115-1 程式設計與統計軟體

Visual Studio Code 入門與統計程式實作

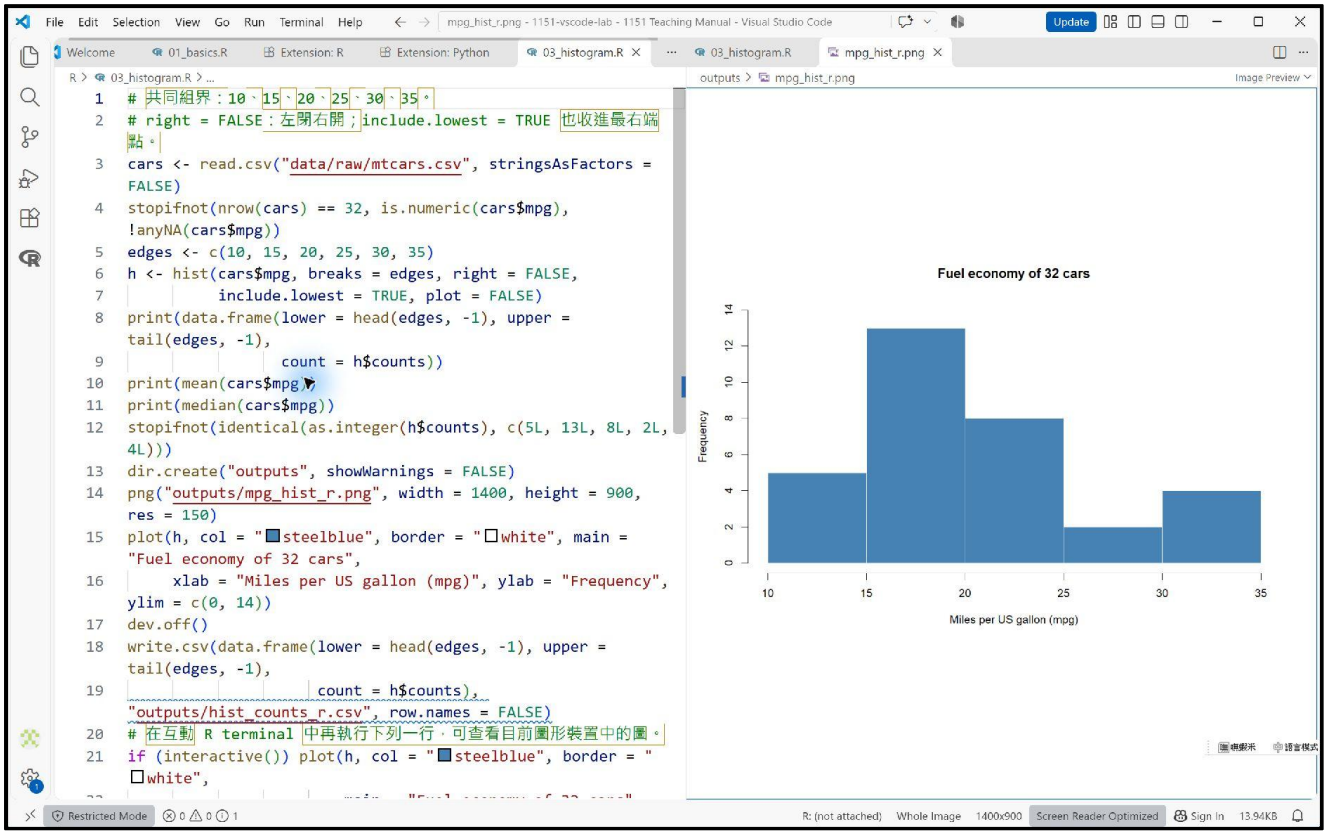
從 R 與 Python 到 AI 協作及版本控制

吳漢銘
國立政治大學統計學系

Windows 操作版　|　教材版本 v1.2　|　修訂日期 2026 年 11 月 12 日

本教材示範版本　Visual Studio Code 1.145.2（Windows）

本手冊帶你建立可重現的統計分析工作流程：整理專案、寫程式、執行與除錯，再使用 AI 改進程式並以 Git 保存修改。每一階段都要能說明程式做了什麼，以及如何驗證結果。



環境設定 → R 實作 → Python 對照 → 編輯與除錯 → AI 協作 → GitHub → Terminal

課程網站：hmwu.idv.tw/115-1_statsoftware/

如何使用這本手冊

先完成一個可以解釋的小成果，再增加工具。第一次學習建議依第 1 至 7 章操作；已熟悉 RStudio 的同學，可先閱讀第 3、4 章的工作區及執行方式，再進入 Python 與 AI。

學習單元	完成時應能做到
環境與 R　第 1–4 章	開啟正確資料夾，建立 .R，完成算術、文字與直方圖。
Python 與操作　第 5–7 章	選對 interpreter，完成相同統計圖，定位及修正錯誤。
AI 與版本　第 8–10 章	提供清楚需求，檢查 AI 程式與差異，以 commit 保存成果。
Terminal 與整合　第 11–12 章	辨認命令輸入位置，操作 Claude Code，整理可重現小專案。

每個操作單元都包含目的、步驟、預期結果及練習。程式碼可直接從 Word / PDF 複製；完整檔案另見隨附範例包。請先解壓縮範例包，再以 File > Open Folder 開啟 1151-vscode-lab。

畫面閱讀方式

標示「實機截圖」者來自本課範例；「官方示範截圖」保留原官方案例及介面。不同版本的圖示位置可能略有差異，因此正文同時列出選單或命令名稱。圖中的 JavaScript 等官方案例只用來辨認操作位置，實作仍使用本課的 R / Python 程式。

程式結果與學習責任

R / Python 範例已使用同一份 mtcars 資料執行驗證。AI 回答每次可能不同；提示語是練習題，不保證產生同一份答案。對照程式、資料、統計方法、數值與圖形後，才把結果納入作業。課堂與考試的 AI 使用方式，依教師當次規定。

版本紀錄

教材 v1.2：本次修訂截圖尺寸、圖前後留白、封面版本資訊及深色佈景建議。教材版次與軟體版本分開記錄；後續小幅修訂可依序使用 v1.3、v1.4，重大改版再改為 v2.0。

本機示範：VS Code 1.145.2、R 4.6.1、R extension 3.4.0、Python extension 2026.10.0。Python 範例驗證環境與套件清單見範例包 requirements-tested.txt。AI 功能及帳號條件以登入時顯示為準。

目錄

如何使用這本手冊 2
第 1 章　認識 VS Code 4
第 2 章　安裝與介面設定 7
第 3 章　資料夾與工作區 13
第 4 章　在 VS Code 寫 R 18
第 5 章　在 VS Code 寫 Python 26
第 6 章　編輯與擴充套件 33
第 7 章　程式執行與除錯 37
第 8 章　用 AI 協作撰寫 R 41
第 9 章　Git 與 GitHub 47
第 10 章　在 VS Code 使用 AI 助手 52
第 11 章　Terminal 與 Claude Code 57
第 12 章　綜合實作與繳交 62
操作速查與學習資源 65
學習資源與引用說明 67

第 1 章　認識 VS Code

把編輯器變成統計程式工作台

學習目的：說明 VS Code 在資料分析流程中的角色，辨認編輯、執行與檢查結果三個階段。

Visual Studio Code，簡稱 VS Code，是用來編輯程式與管理檔案的工具。本課將它當作 R 與 Python 共用的工作台：左側整理資料與程式，中間撰寫分析步驟，下方查看執行訊息。分析完成後，程式、資料與圖檔仍是一般檔案，可以交作業、備份或移到另一臺電腦。

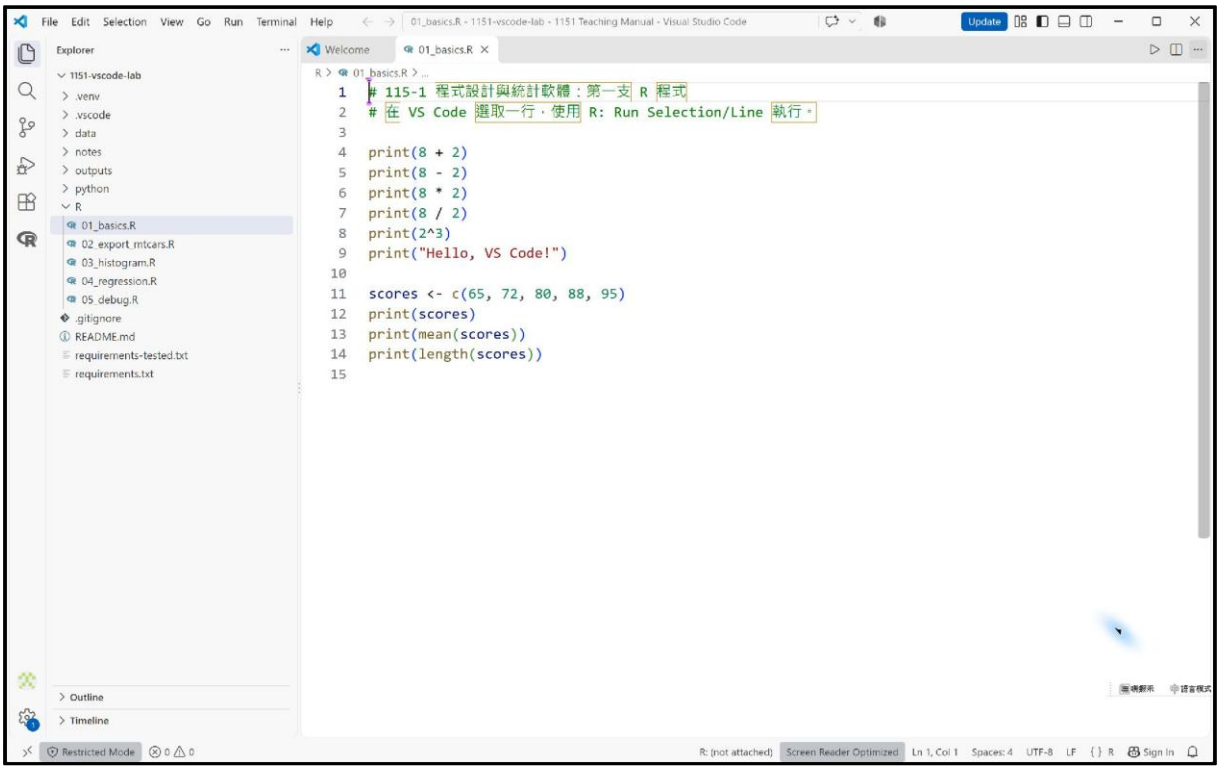


圖 1-1　本機實拍：左側 Explorer 整理檔案，中央 Editor 編輯 R 程式；下方面板目前未展開。

實機截圖：本課範例；畫面可依正文命令操作。

VS Code 本身不等於 R 或 Python。打開 .R 檔只代表看得到程式；要真正計算，電腦還需要 R。打開 .py 檔則需要 Python 解譯器。擴充套件負責讓編輯器認識語言，提供提示、傳送程式及偵錯等整合功能。安裝套件與安裝語言主程式是兩件事。

本手冊從一個資料夾出發，先完成基本計算與直方圖，再加入搜尋、偵錯、AI 協作及版本控制。每次練習都要能指出輸入的程式、採用的資料，以及得到的輸出；有畫面不代表計算正確，還要對照預期結果。

練習與自我檢核　請用一句話回答：若按下執行卻顯示找不到 R，應先檢查語言安裝、還是更換佈景主題？說明理由。

延伸查閱：Get to know Visual Studio Code

第 1 章　認識 VS Code

與 RStudio 的使用經驗接軌

學習目的：把熟悉的 RStudio 工作流程對應到 VS Code，了解功能優點與使用成本。

學過 RStudio 的同學可以沿用「寫程式、送出執行、檢查物件與圖形」的順序。差別在於 VS Code 的語言功能多由擴充套件提供，視窗位置也能重新安排。剛開始不要要求每個畫面都與 RStudio 相同；先確定程式能重跑，再逐步調整習慣。

RStudio 經驗	VS Code 對應與差異
Source 腳本區	Editor 編輯 .R / .py 等文字檔。
Console	R 擴充套件建立的 R 終端；一般 PowerShell 並非 R。
Files / Project	Explorer 管理資料夾；開啟資料夾即可開始課程專案。
Environment / Plots	視語言擴充套件及其設定提供；畫圖也可能開啟外部視窗。

優點是同一套操作可用於不同語言，資料夾、搜尋、差異比較與 Git 能放在同一個視窗；擴充套件也讓個人逐步加入需要的能力。對同時學 R 與 Python 的學生，熟悉一次檔案操作，就能沿用到另一種語言。純文字程式檔容易閱讀與交換，也適合讓 AI 解釋或協助修改。

代價是第一次設定需要分清楚編輯器、語言程式、套件與環境。不同擴充套件的品質、更新速度及鍵盤設定可能不同；安裝太多套件也會讓介面變得複雜。RStudio 的部分統計工作流程比較集中，轉到 VS Code 後可能要額外設定。AI 與遠端服務則可能需要帳號、網路或付費方案，不能把編輯器免費理解成所有附加服務都免費。

本課以完成統計任務為判準，不以安裝最多工具為目標。若只需撰寫短小 R 程式，先使用 R 擴充套件與基礎終端即可；需要跨語言專案，再加入 Python；需要分享修訂歷史，再學 Git。遇到操作不順時，先記錄目前使用的語言與畫面，再尋找對應設定。

選擇工具也要考慮課堂目的。若今天的重點是平均數與直方圖，環境設定應服務這些任務，不必先學完所有功能。完成同一份 R 分析後，再用 Python 重寫一次，可以把注意力放在語言差異；檔案管理、搜尋與版本控制的操作則繼續沿用。這是本課採用 VS Code 的主要學習價值。

練習與自我檢核　列出你在 RStudio 最常用的兩項功能，對照本頁找出在 VS Code 的位置或需要補裝的功能。

延伸查閱：Get to know Visual Studio Code　|　VS Code R language support　|　VS Code Python tutorial

第 1 章　認識 VS Code

用可重跑的分析流程學工具

學習目的：建立儲存、執行、核對及保留成果的習慣，避免把畫面上的結果誤認為完整作業。

一份統計作業至少包含問題、資料、程式與結果。假設我們要查看一組測量值的分布：資料是數值向量或 CSV，程式負責讀取與畫圖，結果是直方圖與文字解釋。VS Code 協助把這些材料放在同一個專案內，讓自己下週或另一位同學都能按說明重新完成分析。

1. 先確認專案根資料夾與檔名，將 R 程式存到 R、Python 程式存到 python 子資料夾。不要只留在 Untitled 分頁。
2. 每完成一小段功能就儲存，執行目前選取的段落，觀察訊息與結果；確認正確後再繼續。
3. 改變輸入後重新執行，檢查數值或圖形是否跟著變化。特別注意終端中可能殘留前一次建立的物件。
4. 準備交作業時，重新啟動語言工作階段，從第一行執行到最後一行，確認程式不依賴先前手動輸入的內容。

容易混淆的動作	它實際完成的事
Save 儲存	把目前檔案內容寫入磁碟；不會自動證明程式可執行。
Run 執行	把程式交給語言處理；結果仍需核對。
Commit 提交	把選定版本記入本機 Git 歷史。
Push 推送	把本機提交傳到遠端；儲存檔案並不等於已上傳。

不要用「沒有紅字」作為唯一正確標準。例如樣本數明明是十筆，程式卻讀到九筆，即使沒有錯誤訊息也要找出原因。直方圖標題、橫軸單位、分組方式與遺漏值處理都會影響解讀。AI 產生程式後仍沿用同樣的核對流程，並能用自己的話說明每一段。

遇到問題時，保留最小可重現範例：使用少量不含個人資料的數字，記下檔名、執行方式、完整錯誤文字與期待的結果。這些資訊比「我的電腦不能跑」更容易讓教師或 AI 找到原因。先讓小範例成功，再放回完整分析。

課堂上請保留「為什麼這樣寫」的說明。例如使用固定分組界線，是為了讓 R 與 Python 直方圖可比較；不是因為圖畫得出來就代表參數選得合適。

練習與自我檢核　建立你的作業檢查習慣：交作業前依序確認「已儲存、可重跑、結果合理、檔案齊全」。在 README 記錄完成情形。

延伸查閱：Tutorial Get started with Visual Studio Code　|　Source control quickstart

第 2 章　安裝與介面設定

完成 Windows 的基本安裝

學習目的：安裝桌面版 VS Code，確認版本與開啟方式，分清後續語言安裝的需求。

本手冊以 Windows 桌面版為主。請從 VS Code 官方下載頁取得安裝程式，依電腦處理器選擇適用版本；一般個人電腦可使用 User Installer。學校電腦若已有安裝，先確認能正常開啟，不必為了跟講義畫面完全相同而重複安裝。

1. 開啟 https://code.visualstudio.com/download，選擇 Windows 適用的安裝檔，執行後閱讀條款及安裝選項。
2. 依安裝精靈完成設定。若有 Add to PATH 選項，保留勾選，日後才能從新開的終端使用 code 指令。
3. 從開始功能表開啟 Visual Studio Code；使用 Help → About 查看版本，並在課堂紀錄寫下版本號。
4. 初次歡迎頁可以稍後再設定帳號。本課先完成資料夾與程式編輯，AI 登入安排在後面章節。

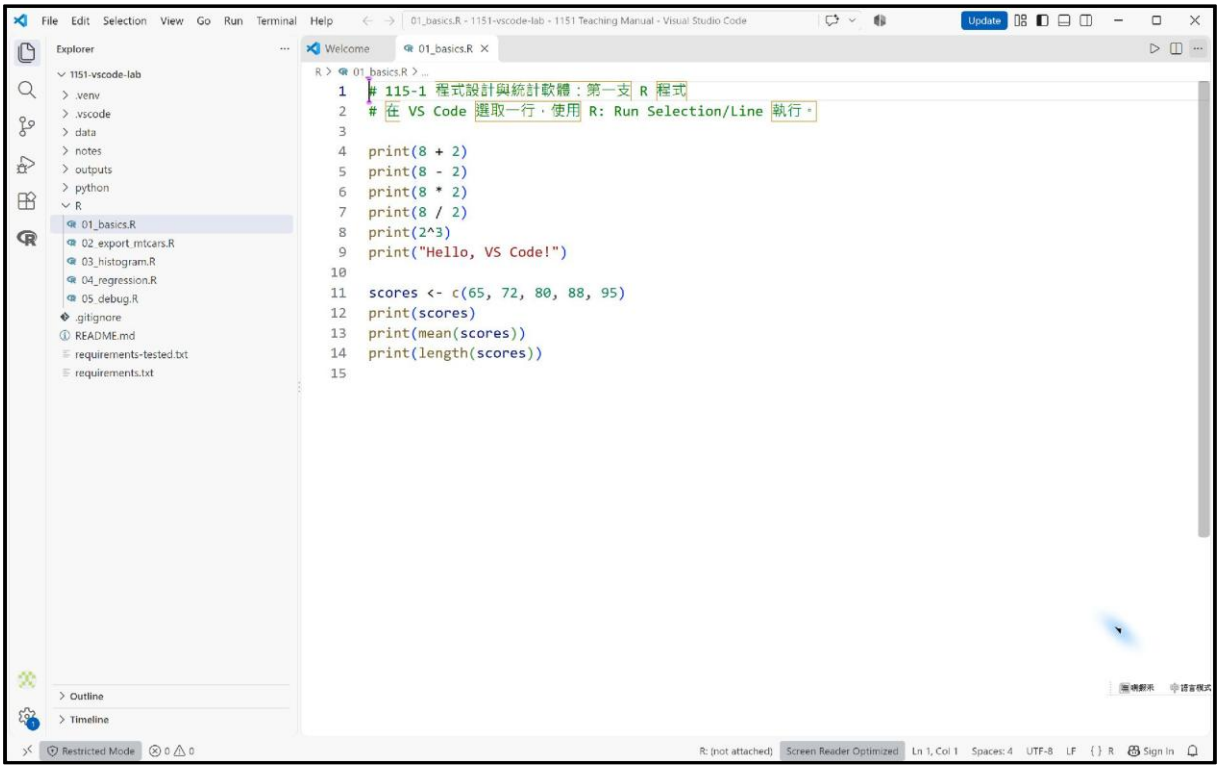


圖 2-1　安裝後能開啟 VS Code 視窗，即可繼續建立練習資料夾。

實機截圖：本課範例；畫面可依正文命令操作。

預期結果是可從開始功能表啟動編輯器。若安裝後終端仍找不到 code，先關閉並重新開啟終端，讓新的環境變數生效。這個問題不妨礙從圖示啟動 VS Code。接著依語言章節安裝 R 與 Python；只裝編輯器，尚不能完成本課的統計計算。

練習與自我檢核　記錄你的 Windows 與 VS Code 版本。若畫面與教材不同，優先搜尋相同的命令名稱，不要只依圖示位置尋找。

延伸查閱：Installing Visual Studio Code on Windows

第 2 章　安裝與介面設定

辨認主要畫面與面板

學習目的：依任務找到 Explorer、Editor、Terminal、Problems 與狀態列。

VS Code 的畫面可依用途分區理解。Activity Bar 是切換功能的入口；主要側邊欄顯示檔案或搜尋結果；中央 Editor 用來編輯；下方 Panel 容納終端與訊息。右側也可能有第二側邊欄顯示 AI 對話。這些位置可移動，所以圖示及名稱比固定方位更可靠。

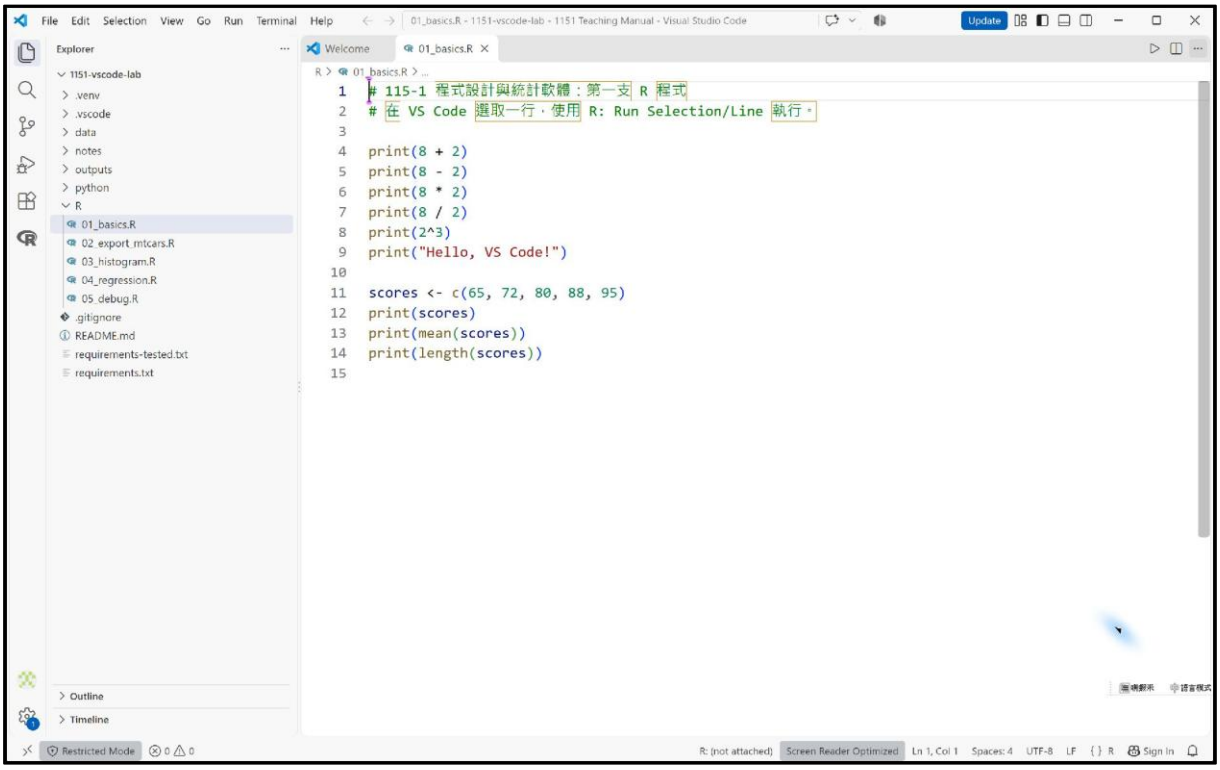


圖 2-2　本機實拍：左側是 Explorer，中央是 Editor，底部是狀態列；按 Ctrl+J 可另行展開 Panel。

實機截圖：本課範例；畫面可依正文命令操作。

區域或檢視	本課用途
Explorer	新增、開啟及整理資料夾與檔案。
Search	找出多個程式檔中相同的名稱。
Source Control	檢查修改、暫存與 Git 提交。
Run and Debug	啟動偵錯並查看變數。
Extensions	尋找、安裝、停用及移除擴充套件。
Panel	Terminal 執行命令；Problems 顯示診斷；Output 顯示工具紀錄。

狀態列通常位於最下方，可看行號欄號、編碼及語言模式等資訊。某些項目由擴充套件加入。若找不到下方面板，可選 View → Appearance → Panel，或按 Ctrl+J 切換顯示；這只是隱藏或顯示，不等於停止正在執行的程式。

練習與自我檢核　依序開啟 Explorer、Search 與 Extensions，再返回 Explorer。指出哪一區域負責輸入程式，哪一區域適合讀錯誤訊息。

延伸查閱：VS Code user interface

第 2 章　安裝與介面設定

用命令選擇區找到功能

學習目的：使用 Command Palette 執行功能，理解它與檔案搜尋及終端的差別。

Command Palette 是依名稱尋找 VS Code 功能的入口。忘記選單位置時，先按 Ctrl+Shift+P，再輸入功能名稱。這裡執行的是編輯器命令，例如切換佈景或選擇 Python 解譯器；不是輸入 R 計算式或 PowerShell 指令的地方。

1. 按 Ctrl+Shift+P，確認頂端出現輸入欄與開頭的 > 符號。
2. 輸入 Color Theme，從結果選擇 Preferences: Color Theme，確認 VS Code 開啟佈景清單。
3. 按 Esc 離開。接著按 Ctrl+P 輸入已存在的檔名，觀察這次清單是在尋找檔案。
4. 再按 Ctrl+Shift+P，輸入 Keyboard Shortcuts，開啟鍵盤快速鍵設定並查詢 Save。

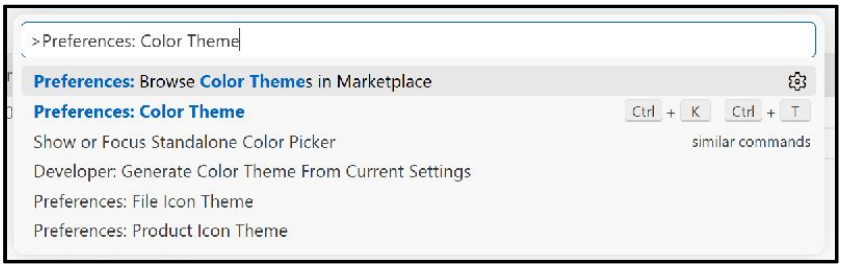


圖 2-3　命令選擇區可以依關鍵字搜尋功能，省去記住每個選單位置的負擔。

實機截圖：本課範例；畫面可依正文命令操作。

功能可能顯示英文或中文，取決於顯示語言。若搜尋不到 R 或 Python 指令，先確認對應擴充套件已安裝且啟用。部分命令還需要先開啟正確副檔名。快捷鍵若被其他軟體攔截，可以從 View → Command Palette 開啟同樣的入口。

預期結果是能不依賴滑鼠位置找到至少一項功能。之後看到教材中的粗體命令名稱，都可以先嘗試由此搜尋；執行前讀清楚結果，避免選到名稱相近但用途不同的命令。

練習與自我檢核　各用一句話說明 Ctrl+P、Ctrl+Shift+P 與 Terminal 的用途，再用命令選擇區開啟 Settings。

延伸查閱：VS Code user interface Command Palette

第 2 章　安裝與介面設定

調整佈景與文字大小

學習目的：把畫面調整到適合閱讀與投影的狀態，區分編輯器字型與整體縮放。

本課建議同學先使用內建 Dark Modern 深色佈景。在較暗的環境中，深色背景減少大面積亮色區塊，畫面的亮度感受通常較柔和；搭配語法上色，可用不同顏色辨識註解、字串與數字，閱讀 R / Python 程式時較容易分辨各段內容。仍須有適當照明，並調整螢幕亮度與文字對比；若不易閱讀，可換用淺色或高對比佈景。佈景只改變顏色，不影響運算結果。

1. 按 Ctrl+Shift+P，執行 Preferences: Color Theme，以方向鍵預覽 Dark Modern，再按 Enter 套用。這是內建佈景，不必另裝擴充套件。
2. 按 Ctrl+, 開啟 Settings，搜尋 editor font size，將 Editor: Font Size 調整到自己能清楚閱讀的大小，例如 16。
3. 若連檔案清單與選單都太小，使用 View → Appearance → Zoom In 調整整體介面；這和只改程式文字大小不同。
4. 以一段含註解、字串與數字的 R 程式檢查語法顏色，並確認選取範圍與錯誤標記清楚。投影時再從同學的觀看距離檢查字體與對比。

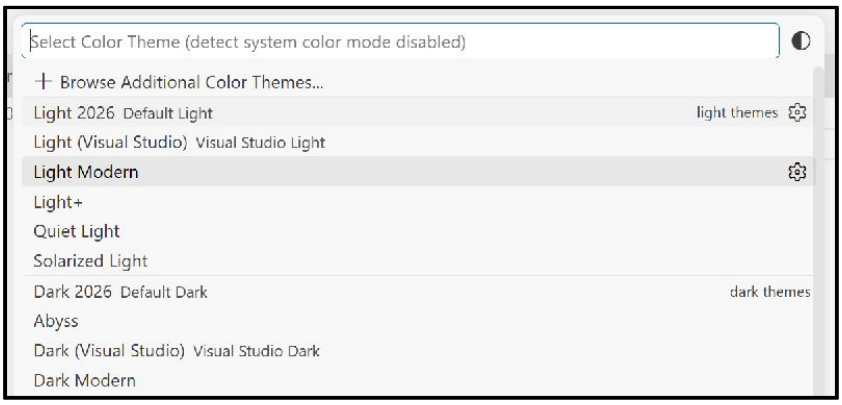


圖 2-4　佈景清單可預覽顏色；選定後確認文字、選取區域與錯誤標記的辨識度。

實機截圖：本課範例；畫面可依正文命令操作。

延伸查閱：Color themes in VS Code　|　Settings editor　|　Accessibility: Zoom and High Contrast theme

確認深色佈景與文字對比

學習目的：確認程式文字、語法顏色與選取範圍清楚，完成個人閱讀與課堂投影的檢查。

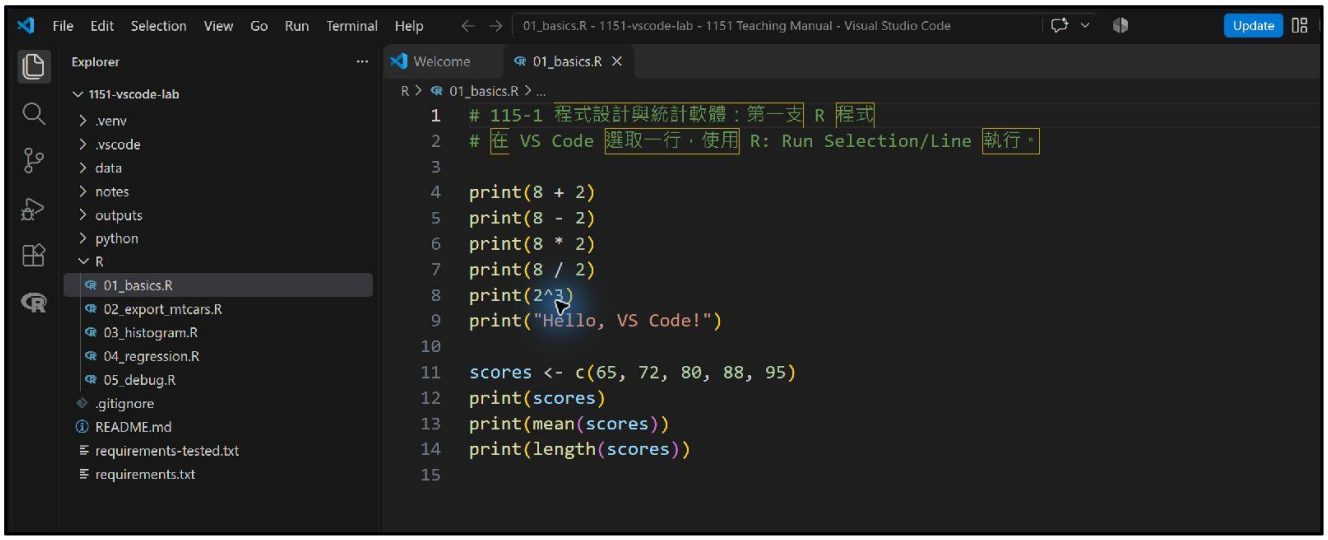


圖 2-5　本機實拍：切換深色佈景後的編輯區；比較文字、註解與選取範圍的對比。

實機截圖：本課範例；畫面可依正文命令操作。

若調整後仍看不清楚，先檢查是否改到另一個設定範圍，或整個視窗仍被縮小。若只想恢復某項設定，使用該設定旁的齒輪選擇 Reset Setting，比刪除全部個人設定更容易控制影響。投影前也可收起暫時不用的側邊欄，保留程式與輸出空間。

練習與自我檢核　先試用 Dark Modern 與適當字型大小，指出一段程式中的註解、字串與數字。請鄰座同學在正常觀看距離讀出一行程式，再依需要調整亮度、字體或佈景。

延伸查閱：Color themes in VS Code　|　Settings editor　|　Accessibility: Zoom and High Contrast theme

第 2 章　安裝與介面設定

設定自動儲存與工作區偏好

學習目的：辨別 User 與 Workspace 設定，建立可控的儲存與顯示習慣。

User 設定套用於自己的編輯環境；Workspace 設定則跟目前專案有關。調整閱讀字型可放在 User，課程專用的格式或檔案設定可放在 Workspace。相同設定在不同範圍可能互相覆寫，所以修改前先確認設定頁上方選中了哪個分頁。

1. 按 Ctrl+, 開啟 Settings，先選 User。搜尋 files auto save，查看 Files: Auto Save 的選項。
2. 初學時可先使用 off 並練習 Ctrl+S；若改成 afterDelay，編輯停止一小段時間後檔案會自動儲存。
3. 搜尋 word wrap，將 Editor: Word Wrap 設為 on，讓長文字在視窗寬度內換行顯示；這不會自行插入程式換行字元。
4. 開啟課程資料夾後切換到 Workspace，再觀察設定範圍。只有課程需要的設定才放進此範圍。



圖 2-6　本機實拍：Settings 上方可選 User 或 Workspace；圖中以 Font Size 示範搜尋，Auto Save 也由同一搜尋欄查找。

實機截圖：本課範例；畫面可依正文命令操作。

預期結果是能解釋何時儲存檔案，以及設定影響哪些專案。分頁上的未儲存標記可用來提醒自己。自動儲存不是版本歷史，也不會自動執行程式；要追蹤一段時間前的版本，後面仍需使用 Git。若設定看似沒有生效，先查 Workspace 是否有同名選項，不要立刻重裝 VS Code。

練習與自我檢核　把 Auto Save 的選擇與原因記在 README。儲存檔案後執行程式，確認你知道這是兩個不同動作。

延伸查閱：User and workspace settings　|　Basic editing Save Auto Save

第 3 章　資料夾與工作區

從資料夾開始一個課程專案

學習目的：理解檔案、資料夾、專案與 Workspace 的關係，先完成最簡單的單資料夾工作區。

本課的「專案」是一組為同一個分析目的準備的資料與程式。VS Code 沒有所有語言共用的必填 New Project 精靈；初學統計程式時，建立資料夾並用 Open Folder 開啟，就能開始。Workspace 指本次視窗開啟的工作範圍，可以是一個資料夾，也可以由多個根資料夾組成。

1. 在 Windows 檔案總管建立課程練習資料夾 1151-vscode-lab，放在自己可讀寫的位置。
2. 回到 VS Code，選 File → Open Folder，選取剛建立的資料夾。
3. 確認 Explorer 的最上層顯示 1151-vscode-lab，再開始新增檔案；不要只把個別 .R 檔拖進空白視窗。
4. 若出現 Workspace Trust 提示，先確認來源。自己建立且內容已知的資料夾可依情況信任；陌生下載資料先閱讀內容。

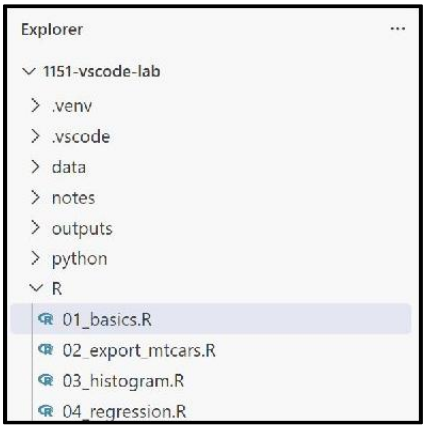


圖 3-1　Explorer 最上層資料夾就是本次練習的工作範圍。

實機截圖：本課範例；畫面可依正文命令操作。

預期結果是新增檔案會存進正確資料夾。若畫面顯示 NO FOLDER OPENED，表示目前可能只開了檔案；重新執行 Open Folder。RStudio 的 .Rproj 不是本課必要條件，VS Code 的 .code-workspace 檔也不是 R 程式，兩者不要當成同一種專案檔。

VS Code 的 Workspace 與 R 的工作空間不同：前者是資料夾與編輯設定的範圍；後者是 R 記憶體中的物件。.RData 可保存 R 物件快照，不會替代 .R 腳本，也不是 .code-workspace。重新分析應以可重跑的程式為主。

練習與自我檢核　指出目前資料夾的完整路徑，並確認 Explorer 顯示的名稱與 Windows 檔案總管一致。

延伸查閱：What is a VS Code workspace　|　Workspace Trust

第 3 章　資料夾與工作區

新增資料夾與程式檔

學習目的：在 Explorer 建立有副檔名的程式檔，能辨認儲存位置與語言模式。

資料夾管理應先求簡單明確。R 程式放 R、Python 程式放 python，原始資料放 data/raw，圖形與表格放 outputs；這樣交作業時較容易檢查哪些是輸入、哪些能由程式重新產生。VS Code 的 Explorer 與磁碟上的資料夾同步，重新命名或刪除也會影響實際檔案。

1. 在 Explorer 的專案根資料夾按滑鼠右鍵，選 New Folder，輸入 R；再建立 python、data、outputs 與 notes，並在 data 內建立 raw。
2. 在 R 按右鍵選 New File，輸入 01_basics.R，注意最後的 .R。若使用教師提供的練習包且已有同名檔，直接開啟查看，不要覆寫。
3. 在新檔輸入一行註解，例如 # My first R script，按 Ctrl+S。檢查分頁不再有未儲存標記。
4. 查看右下角語言模式是否為 R。若顯示 Plain Text，先確認實際檔名不是 01_basics.R.txt，再檢查副檔名或手動選擇語言模式。

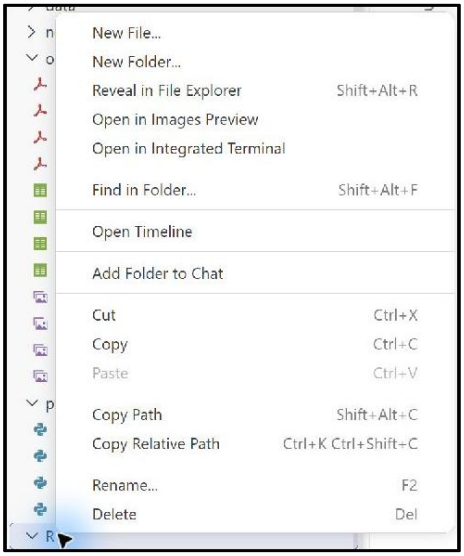


圖 3-2　先選中目標資料夾，再新增有正確副檔名的程式檔。

實機截圖：本課範例；畫面可依正文命令操作。

預期結果是在 R 資料夾中看見程式檔。單擊檔案可能使用預覽分頁，雙擊可保持開啟；分頁被替換不代表檔案消失。若存錯位置，先用 Save As 存到正確資料夾並核對內容，再整理重複版本。不要依賴桌面上多份同名檔案來判斷最新作業。

練習與自我檢核　在 python 資料夾新增 01_basics.py，並在根目錄新增 README.md。說明哪個是 Python 程式，哪個用來寫使用說明。

延伸查閱：Editor tutorial Open a folder and create a file

第 3 章　資料夾與工作區

安排可交接的檔案結構

學習目的：為統計分析建立穩定的輸入與輸出位置，寫出足以讓同學重跑的 README。

建議整學期使用共同命名規則，每份獨立作業保留自己的專案根資料夾。本課示範結構如下。data/raw 用來放原始資料，R 與 python 分別放兩種語言的分析程式，outputs 保存結果，notes 記錄 AI 協作；README.md 記錄問題、執行方式與檔案用途。資料量很少時，也可以先把向量寫在程式內，等學到讀取 CSV 再使用 data。

```
1151-vscode-lab/
  README.md
  R/01_basics.R
  R/03_histogram.R
  python/01_basics.py
  python/02_histogram.py
  data/raw/mtcars.csv
  outputs/mpg_hist_r.png
  outputs/mpg_hist_python.png
  notes/ai_log.txt
  .gitignore
```

根資料夾是閱讀相對路徑的共同起點。例如 data/raw/mtcars.csv 表示根資料夾底下 data/raw 內的檔案。不要把自己電腦的 D 槽完整路徑寫進交作業程式，因為同學可能使用另一個磁碟。相對路徑仍受目前工作目錄影響；開啟根資料夾後，執行時也要確認工作目錄，不能只看程式檔所在位置就推斷讀檔位置。

README 應用平常的話說明：這個專案研究什麼、需要 R 或 Python、先執行哪個檔案、資料放哪裡、結果輸出到哪裡，以及有沒有額外套件。不要只寫「執行即可」。若使用固定隨機種子，也註明這是為了重現結果；若圖形每次略有不同，要能解釋原因。

原始資料與分析結果分開保存，可以減少誤覆寫來源。若要清理資料，程式應讀原檔並輸出另一份整理結果，保留重做的可能。暫存檔與下載的參考資料不必混進 R 或 python。到 Git 章節會再用 .gitignore 排除不適合追蹤的檔案，但它不會替代整潔的資料夾結構。

本課提供 mtcars 範例，資料欄位的名稱與單位也應寫進說明。mpg 代表每加侖英里數，不能把它直接稱為每公升公里數。當 R 與 Python 共用同一份 CSV，就可核對樣本數、平均值及直方圖的分組計數。資料來源一致，才容易判斷差異來自分析程式還是不同輸入。

notes/ai_log.txt 用來記錄向 AI 提出的問題、採用的修改與自行驗證方式。它能協助回想決策，也方便教師了解學習過程。記錄應以必要內容為限，不放登入憑證或不適合分享的資料。

練習與自我檢核　請同學只看 README，說出從哪個檔案開始、輸入和輸出在哪裡。若仍須口頭補充，就把必要資訊補入 README。

延伸查閱：VS Code workspace concepts　|　R working directory reference

第 3 章　資料夾與工作區

讓檔名與路徑容易辨認

學習目的：建立一致命名方式，避免副檔名、大小寫與目前工作目錄造成的執行問題。

檔名應表達內容與順序。初學時建議以英文字母、數字、底線或連字號命名，例如 03_histogram.R；使用兩位數編號可讓 01、02、10 維持自然排序。中文檔名並非一律不能使用，但交換到不同系統或工具時，簡單名稱更容易輸入與查錯。課程說明文字仍可使用中文。

建議做法	例子與理由
保留正確副檔名	R 用 .R；Python 用 .py；資料可用 .csv。
一個意思一個名稱	histogram.R 比 test2_final_ok.R 容易理解。
固定大小寫習慣	data 與 Data 在不同系統可能被視為不同位置。
版本交給 Git	不要靠 final、final2、final_new 猜哪一份才有效。
避開系統禁用字元	Windows 檔名不可含 < > : " / \ | ? *。

路徑中的空白需要特別留意。在終端切換資料夾時，將含空白的整段路徑放在引號內。R 與 Python 字串中的反斜線又有自己的語法規則，因此入門範例優先採相對路徑與正斜線。不要把瀏覽器網址、Windows 路徑與程式中的字串規則混用。資料夾名稱也應保持一致；本課以 outputs 表示成果資料夾，不要混寫 output。

```
data/raw/mtcars.csv
outputs/mpg_hist_r.png
R/03_histogram.R
```

若出現「找不到檔案」，先查三件事：檔案真的存在嗎、拼字與副檔名正確嗎、目前工作目錄是哪裡。Explorer 找得到檔案並不保證語言工作階段會從相同位置搜尋。R 可用 getwd() 查看，Python 可用 pathlib.Path.cwd() 查看；後續語言章節會示範如何核對。移動資料夾後，舊的絕對路徑也可能失效。

重新命名前先儲存內容，再從 Explorer 對檔案按右鍵選 Rename，完成後檢查其他程式是否仍引用舊名稱。選單位置可參考本章「新增資料夾與程式檔」的右鍵選單圖。對資料檔的重新命名不會自動修正所有讀檔字串。若要一次調整多個引用，應先搜尋確認，再逐一修改。

副檔名的功能不只是圖示外觀。把文字檔從 .txt 改為 .R，會讓編輯器依 R 方式處理，但內容仍必須符合 R 語法。相反地，把 Excel 活頁簿改名為 .csv 並不會轉換格式；必須用原軟體另存或由程式匯出。遇到格式問題時，要同時檢查名稱與實際內容。

練習與自我檢核　把含糊的檔名 test_final2.R 改成能說明用途的名稱，並寫下你會搜尋哪些文字以避免讀檔或引用失效。

延伸查閱：VS Code basic editing　|　Microsoft naming files paths and namespaces

第 3 章　資料夾與工作區

儲存與重開多資料夾工作區

學習目的：知道何時需要 .code-workspace，理解它記錄位置與設定而不是封裝所有檔案。

單份作業通常只需 Open Folder。若要同時查看課程範例與自己的作業，才考慮多根工作區：一個視窗列出兩個彼此獨立的資料夾。這樣可共用搜尋與切換操作，又不必把所有檔案搬到一起。工作區檔只記錄資料夾參照及設定，不會把資料與程式自動打包。

1. 先開啟 1151-vscode-lab，再選 File → Add Folder to Workspace，挑選另一個已存在的課程範例資料夾。
2. 確認 Explorer 顯示兩個根資料夾。新增檔案前先選取目標，避免寫進另一份專案。
3. 選 File → Save Workspace As，儲存為 1151-course.code-workspace，位置放在自己容易找到的課程資料夾。
4. 關閉視窗後開啟這個 .code-workspace 檔，確認兩個根資料夾都重新出現。

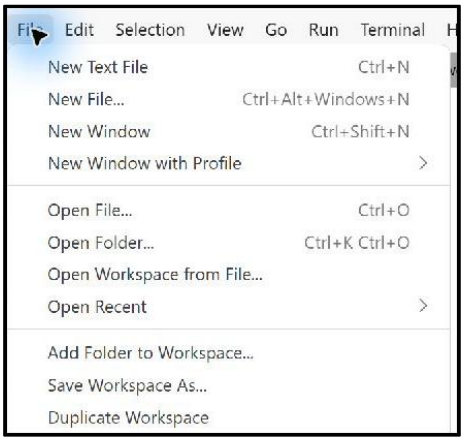


圖 3-3　本機實拍：File 選單包含 Add Folder to Workspace 與 Save Workspace As，分別加入根資料夾及儲存工作區參照。

實機截圖：本課範例；畫面可依正文命令操作。

若日後搬動或重新命名資料夾，原有工作區參照可能找不到位置，需移除失效的參照後重新加入。Remove Folder from Workspace 只從工作區移除該參照，不等於刪除磁碟資料夾。分享作業時應包含實際檔案，不能只交 .code-workspace。

練習與自我檢核　判斷：只分析一份資料是否需要多根工作區？如果同時比較教師範例與自己的修改版，多根工作區能省下哪些操作？

延伸查閱：Multi-root workspaces

第 4 章　在 VS Code 寫 R

安裝 R 與語言支援

學習目的：完成 R 主程式與語言支援安裝，讓 VS Code 能找到 R 並提供程式提示。

如果已在 RStudio 使用 R，可先檢查現有版本，通常不必為 VS Code 重新安裝另一份 R。VS Code、R 主程式與 R 擴充套件分別負責編輯、計算及整合；三者都要準備好。以下以 Windows 64 位元為例。

1. 開啟 CRAN 官方 Windows 下載頁 https://cran.r-project.org/bin/windows/base/，下載教師指定的穩定版安裝程式；不要選開發版。
2. 執行安裝程式，使用預設元件，記錄安裝路徑；保留 Save version number in registry，讓擴充套件較容易找到 R。
3. 安裝後先從 Windows 開始功能表啟動 R，在 R Console 執行下列程式，確認主程式與 languageserver 可用。
4. 回到 VS Code 的 Extensions，搜尋 R，核對發行者 REditorSupport 與識別碼 REditorSupport.r 後安裝。重新開啟 VS Code，再執行 R: Create R terminal。

```
R.version.string
install.packages("languageserver",
                 repos = "https://cloud.r-project.org")
packageVersion("languageserver")
R.home("bin")
```

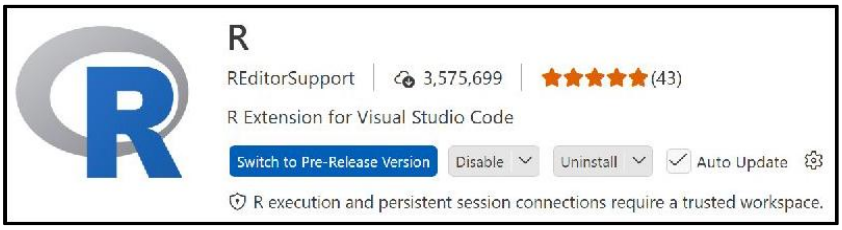


圖 4-1　在 VS Code 安裝 R 擴充套件；R 主程式與 languageserver 要分別安裝。

實機截圖：本課範例；畫面可依正文命令操作。

若仍找不到 R，先確認安裝目錄內的 R.exe 存在。新版擴充套件在 Settings 以 r.consolePath 指定互動終端機使用的 R，r.executablePath 指定背景程序使用的 R；舊版欄位可能不同，以安裝版本的官方說明為準。vscode-R 3 的首次啟動可能提示安裝其 sess 支援套件，請閱讀提示並完成。

練習與自我檢核　記錄 R 版本、R.home("bin") 與 languageserver 版本。確認新終端機顯示的 R 版本與你剛安裝或選擇的一致。

延伸查閱：CRAN Windows 下載　|　REditorSupport Windows 安裝說明

第 4 章　在 VS Code 寫 R

啟動第一個 R 工作階段

學習目的：連接 VS Code 與 R，並確認程式實際在哪裡執行。

本章延續 R 與 RStudio 環境講義的「安裝、設定、存檔、執行」順序。VS Code 負責編輯檔案，R 負責計算；R 擴充套件連接兩者。先確認電腦已安裝 R，再安裝 R 擴充套件。只看到彩色語法，不代表 R 已經可以執行。

1. 以 File > Open Folder 開啟 1151-vscode-lab，確認 Explorer 中可見 R、python、data 與 outputs。
2. 開啟 R/01_basics.R，確認右下角語言模式為 R；若是 Plain Text，點它並改選 R。
3. 開啟 View > Command Palette，輸入並執行 R: Create R terminal。底部應出現 R 啟動資訊與 > 提示字元。
4. 在 R terminal 輸入下列查詢。getwd() 應指向專案根資料夾；若不是，先使用 setwd("實際的專案完整路徑") 設定，再用 getwd() 確認。

```
R.version.string
getwd()
file.exists("R/01_basics.R")
```

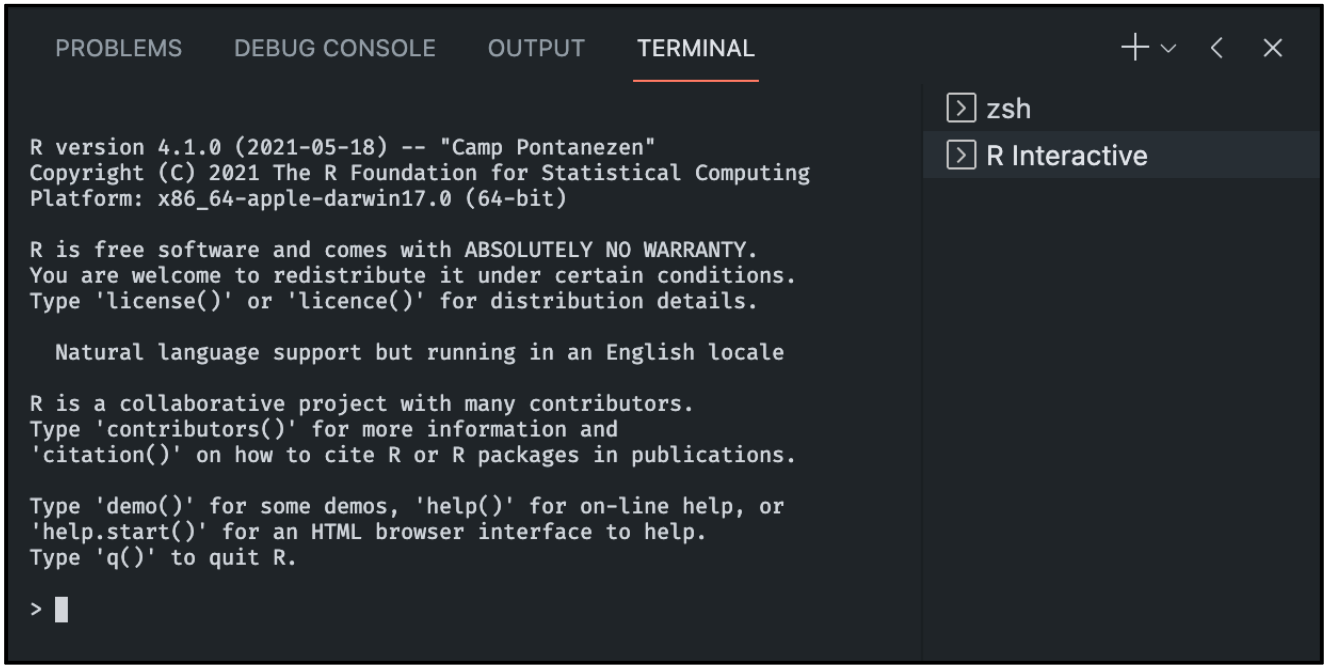


圖 4-2　REditorSupport 官方 R terminal 畫面，使用 macOS 與較早 R 版本；本文操作請使用自己的版本與路徑。

官方示範截圖；R terminal 啟動畫面

預期最後一項為 TRUE。Windows 路徑可使用正斜線，例如 C:/statsoft/1151-vscode-lab。若找不到 R，先確認 R 可獨立啟動，再於 Settings 搜尋 R terminal path，依實際安裝位置設定。不要複製別人的版本資料夾名稱。

練習與自我檢核　關閉目前 R terminal 後重新建立一個，記錄版本與工作目錄。說明「重新開啟程式檔」和「重新啟動 R」有何不同。

延伸查閱：R in Visual Studio Code

第 4 章　在 VS Code 寫 R

四則運算與第一行輸出

學習目的：用明確輸出確認程式執行結果，並建立變數與向量。

開啟 R/01_basics.R。程式中的 # 開始一段註解；註解用來說明目的，不參與計算。先只執行一行，確認結果後再逐步增加。不要將教科書輸出中的 > 或 [1] 複製進程式檔。

```
print(8 + 2)
print(8 - 2)
print(8 * 2)
print(8 / 2)
print(2^3)
print("Hello, VS Code!")
scores <- c(65, 72, 80, 88, 95)
print(mean(scores))
```

1. 將游標放在第一行，開啟命令面板執行 R: Run Selection/Line；預設 Windows 快捷鍵通常為 Ctrl+Enter，可於 Keyboard Shortcuts 核對。
2. 逐行執行四則運算，依序應得到 10、6、16、4；2^3 的結果為 8。
3. 執行文字與 scores 兩行，再執行 mean(scores)，確認平均數為 80。最後按 Ctrl+S 儲存。

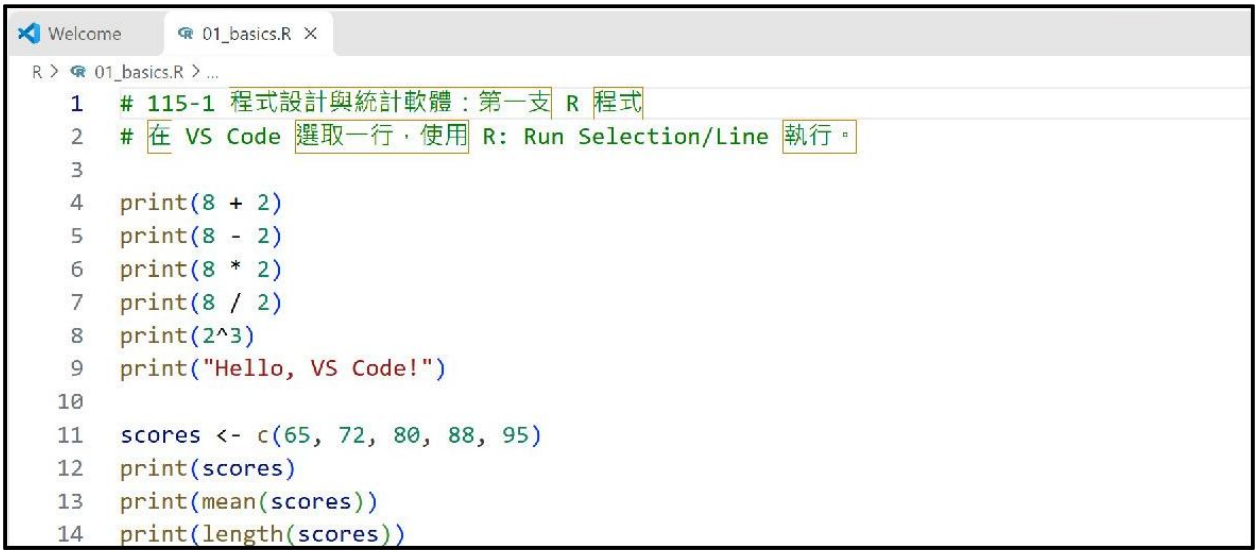


圖 4-3　在 VS Code 編輯本課程的 R 基礎程式；依本文流程將程式送到 R terminal。

實機截圖：本課範例；畫面可依正文命令操作。

文字必須放在成對引號中，變數名稱與函數名稱區分大小寫。scores <- ... 是把右側結果存到左側名稱；c() 把多個數值組成向量。R terminal 的 [1] 表示此行輸出的第一個元素位置，不是計算結果的一部分。

練習與自我檢核　改為自己的五個模擬分數，先手算平均，再執行 mean() 核對。將問候語改為課程名稱，並確認重新開啟檔案後修改仍存在。

延伸查閱：R in Visual Studio Code

第 4 章　在 VS Code 寫 R

逐行執行與整份程式執行

學習目的：分清檔案內容與記憶體中的物件，建立可重跑的習慣。

逐行執行適合探索；整檔執行適合確認流程。R 會保留目前工作階段中建立的物件，因此一個漏掉資料讀取步驟的檔案，可能在你的電腦上暫時跑得動，換到新的 R 工作階段卻失敗。程式應把需要的資料與物件建構步驟寫完整。

1. 在 R/01_basics.R 選取建立 scores 與計算平均的兩行，執行 R: Run Selection/Line。
2. 儲存檔案後，在 R terminal 執行 source("R/01_basics.R")。這會讀取磁碟上的檔案；尚未存檔的變更不會包含在內。
3. 建立新的 R terminal，確認工作目錄，再執行同一個 source()。不靠舊物件也得到相同結果，才算完成。

```
source("R/01_basics.R")
# 整檔執行時，想顯示的值請明確印出。
answer <- mean(c(65, 72, 80, 88, 95))
print(answer)
```

執行方式	適用情境	要注意的事
選取或目前行	試一個公式或檢查資料	先執行它依賴的前面步驟
source("R/檔名.R")	在現有 R 工作階段重跑整檔	先存檔；用 print() 顯示數值
Rscript R/檔名.R	日後從系統 Terminal 執行	使用新程序；路徑與套件仍須正確

Rscript 留到 Terminal 章節再操作。本階段只要會在 R terminal 使用 source()。當提示字元變成 +，通常代表括號、引號或運算式還沒完成；檢查原始碼，不要把更多無關指令接在後面。需要停止時可依終端機的中斷方式終止目前輸入，再重新執行完整一行。

練習與自我檢核　刻意只執行 mean(scores)，觀察新 R 工作階段的錯誤；接著執行建立 scores 的程式，說明錯誤原因。

延伸查閱：R in Visual Studio Code

讀取共用資料並檢查型態

學習目的：用資料列數、欄名、型態與單位確認資料讀取正確。

本手冊使用 R 內建 mtcars，資料描述 32 輛 1973–1974 年車款，包含 11 個數值欄位。範例將車名另存為 model，匯出的 CSV 因此有 12 欄。mpg 表示每美制加侖行駛英里數，數值越大代表同量燃料行駛較遠；wt 的單位是千磅。

1. 在 Explorer 開啟 data/raw/mtcars.csv，確認第一行是欄名，且每列代表一輛車。原始資料作為唯讀的分析起點。
2. 開啟 R/02_export_mtcars.R，閱讀它如何用 write.csv() 建立共用 CSV；教材已附資料，不必每次分析都重新匯出。
3. 在 R terminal 執行下列程式，核對 32 列、12 欄，並檢查 mpg 是否為數值。

```
cars <- read.csv("data/raw/mtcars.csv",
                 stringsAsFactors = FALSE)
dim(cars)
head(cars[, c("model", "mpg", "wt", "cyl")])
str(cars)
summary(cars$mpg)
sum(is.na(cars$mpg))
```

預期 mpg 沒有缺失值，最小值 10.4、最大值 33.9。cars$mpg 取出 mpg 欄；欄名拼錯時要先檢查 names(cars)。檔案存在卻讀不到，先比較 getwd() 與程式中的相對路徑；數字變成文字，則檢查分隔符號、欄位內容及小數表示。

練習與自我檢核　用一行程式計算 mpg 的平均數與中位數，再用文字寫出數值的單位。為什麼不能只說平均數約為 20？

延伸查閱：R datasets mtcars

第 4 章　在 VS Code 寫 R

畫出第一張直方圖

學習目的：指定組界並理解直方圖的計數規則，為跨語言比較做準備。

直方圖描述一個數值變數的分布。本例先指定共同組界 10、15、20、25、30、35，再畫 mpg 的次數直方圖。這比只指定「五組」更清楚，也能讓下一章的 Python 使用相同分組。

```
cars <- read.csv("data/raw/mtcars.csv")
edges <- c(10, 15, 20, 25, 30, 35)
h <- hist(cars$mpg, breaks = edges,
          right = FALSE, include.lowest = TRUE,
          col = "steelblue", border = "white",
          main = "Fuel economy of 32 cars",
          xlab = "Miles per US gallon", ylab = "Frequency")
h$counts
```

1. 在 R terminal 執行資料讀取、組界與 hist()。若已設定互動圖形檢視器，可在對應檢視區查看圖形。
2. 本教材的穩定操作方式是執行 R/03_histogram.R，再於 Explorer 開啟 outputs/mpg_hist_r.png；圖片直接顯示在 VS Code 中。
3. 核對五根柱的次數依序為 5、13、8、2、4，總和為 32。將 col 改成另一個顏色，存檔並重跑。

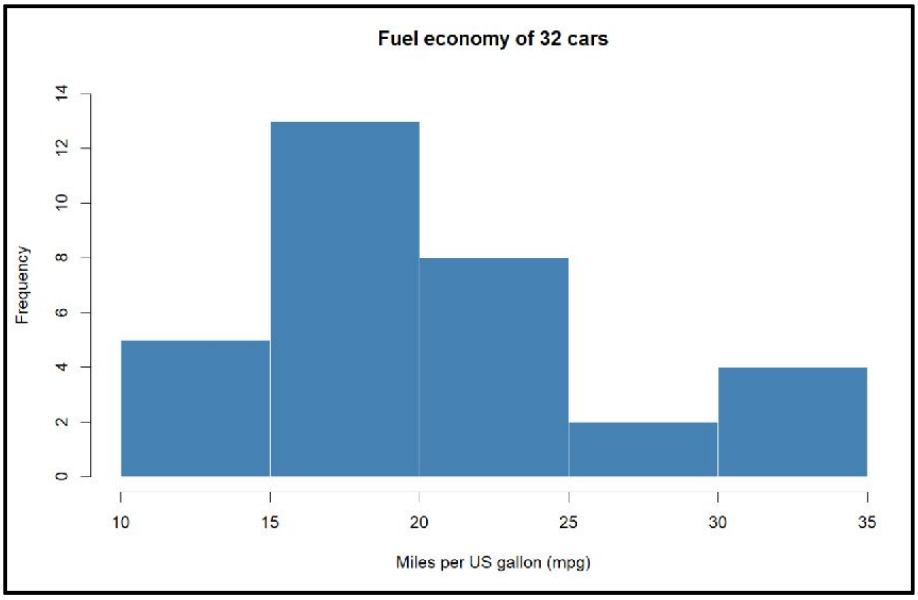


圖 4-4　VS Code 圖片預覽局部：R 程式產生的直方圖，完整保留標題、座標軸及各組次數。

實機截圖：本課範例；畫面可依正文命令操作。

right = FALSE 使前四組採左閉右開，例如 [10,15)；include.lowest = TRUE 在此設定下也收進最右端點，所以最後一組包含 35。頻數圖的縱軸是輛數，不是百分比或機率密度。R 的 breaks 若只給一個數字，實際分組可能經過調整，因此本例使用完整向量。

練習與自我檢核　改用 2.5 的組距，觀察形狀變化。比較前請先列出每一個組界，並核對計數總和仍是 32。

延伸查閱：R graphics hist　|　NumPy histogram

第 4 章　在 VS Code 寫 R

查詢函數與管理 R 套件

學習目的：閱讀函數參數，分清套件安裝與載入，遇到問題時能查證。

函數名稱記不住時，先查說明。以 hist 為例，應讀取用途、參數、回傳值與範例，特別注意 breaks、right 與 freq。VS Code 的語法提示協助輸入，真正的函數行為仍以目前 R 與套件的文件為準。

1. 在 R terminal 輸入 ?hist 或 help("hist")。依環境設定，文件可能出現在 Help 檢視區或瀏覽器。

2. 輸入 args(hist.default) 查看參數名稱；把範例的 col 與 main 改成自己的設定，再執行一次。

3. 若要使用 ggplot2，在 R terminal 安裝一次套件；每個新的 R 工作階段使用前仍需 library(ggplot2)。

```
?hist
args(hist.default)
# 選修：只需安裝一次，更新時再安裝。
install.packages("ggplot2")
library(ggplot2)
packageVersion("ggplot2")
.libPaths()
sessionInfo()
```

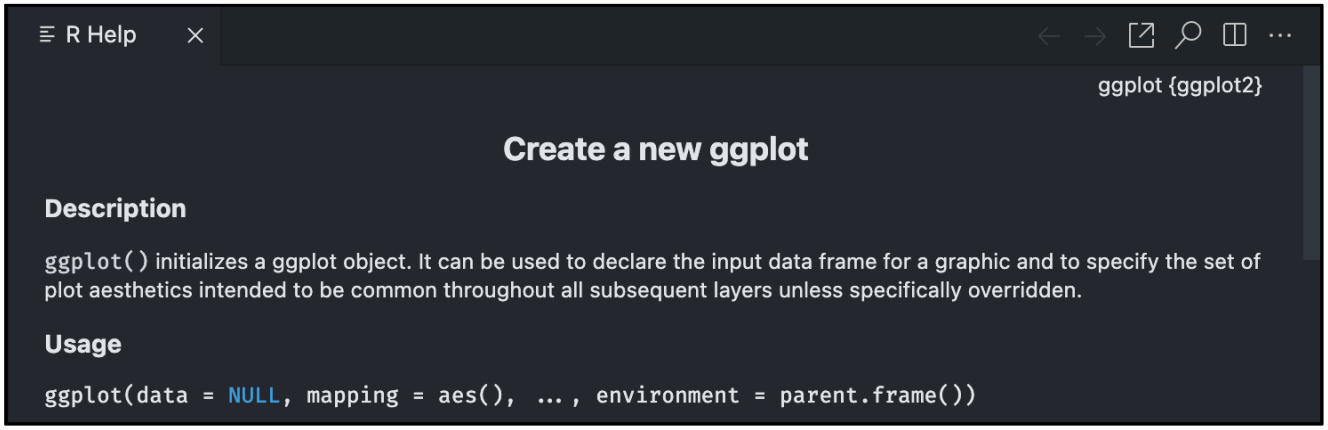


圖 4-5　REditorSupport 官方 Help viewer 畫面，範例為 ggplot2，並非本課程 hist() 的實測畫面。

官方示範截圖；R Help viewer

VS Code 擴充套件在 Extensions 管理；R 套件由 R 管理。安裝 R 擴充套件不會自動安裝 ggplot2。找不到套件時先檢查目前 R 版本與 .libPaths()，因為不同版本可能使用不同套件資料夾。安裝失敗應保留完整錯誤，先排查網路、相依套件和個人套件目錄的寫入權限。

練習與自我檢核　查詢 mean 的 na.rm 參數。對 c(65, 72, NA, 88, 95) 比較預設結果與 na.rm = TRUE，說明忽略缺失值對樣本數的影響。

延伸查閱：R in Visual Studio Code　|　R graphics hist

儲存圖形並閱讀迴歸結果

學習目的：產出可繳交圖檔，並把程式結果和統計解釋連在一起。

互動圖形方便探索，儲存圖檔方便重跑與繳交。R/03_histogram.R 先用 png() 開啟檔案圖形裝置，畫圖後用 dev.off() 完成寫入；少了最後一步，檔案可能尚未完整。圖片存在後，可從 Explorer 直接開啟。

```
dir.create("outputs", showWarnings = FALSE)
png("outputs/my_hist.png", width = 1400, height = 900, res = 150)
hist(cars$mpg, breaks = c(10,15,20,25,30,35),
     right = FALSE, include.lowest = TRUE)
dev.off()
fit <- lm(mpg ~ wt, data = cars)
coef(fit)
summary(fit)
```

1. 執行 R/04_regression.R，開啟 outputs/mpg_wt_r.png。觀察散佈點、迴歸線與平均反應的 95% 信賴帶。
2. 核對截距約 37.2851、wt 係數約 −5.3445、R 平方約 0.7528。這些數字也是之後檢查 AI 修改的基準。

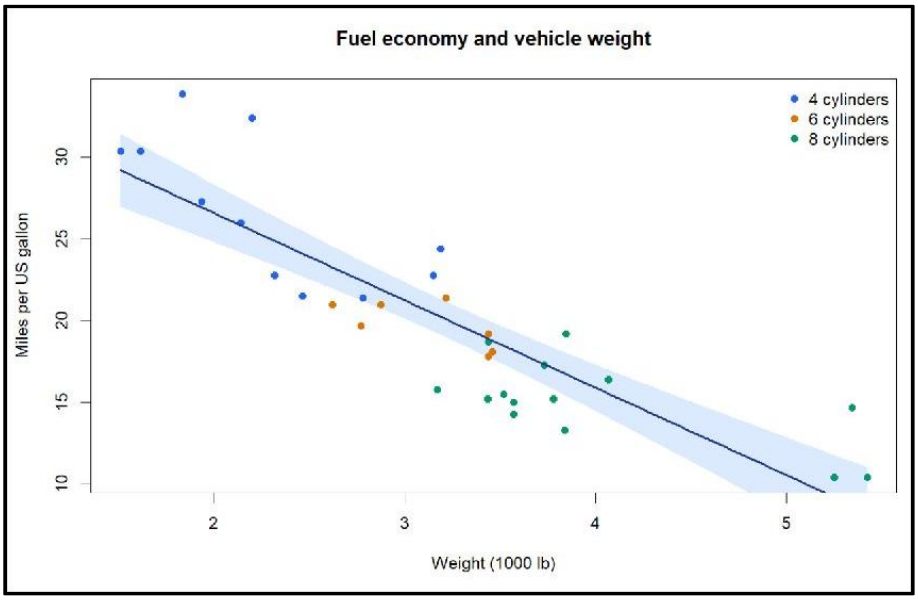


圖 4-6　VS Code 圖片預覽局部：車重與 mpg 散佈圖；顏色表示汽缸數，迴歸線與平均反應的 95% 信賴帶由所有資料共同估計。

實機截圖：本課範例；畫面可依正文命令操作。

模型 mpg ~ wt 只用 wt 當解釋變數。雖然程式將 cyl 轉為 factor 並用顏色區分，並未把 cyl 放進模型。斜率描述車重每增加 1000 磅時，模型平均 mpg 的變化；它不是因果證明。圖中的帶狀區域也不同於單一新車觀測值的預測區間。

練習與自我檢核　將圖形標題改成自己的描述，但保持模型與資料不變。說明單純改標題時，迴歸係數是否應改變。

延伸查閱：R datasets mtcars

第 5 章　在 VS Code 寫 Python

安裝 Python 與選擇執行環境

學習目的：先安裝真正執行程式的 Python，再讓 VS Code 連接它。

Python 主程式不能由單純安裝 VS Code 擴充套件取代。若電腦已有 Python，先查明版本與用途；課程可直接使用合適的既有版本。本頁以 Windows 64 位元安裝程式為例，實際版本依教師及套件相容性安排。

1. 開啟 https://www.python.org/downloads/windows/，找到教師指定的 Stable Release，選 Windows installer (64-bit)。不要選 embeddable package；那是供其他軟體內嵌的精簡套件。
2. 執行安裝程式。若使用傳統安裝介面，勾選 Add python.exe to PATH，選 Install Now 並完成安裝。新版下載頁也提供 Python install manager，使用該路線時依其 Install 流程，勿把兩種安裝畫面的選項混用。
3. 關閉並重新開啟 VS Code，讓新的終端機讀取安裝資訊。在 Terminal 輸入下列版本查詢；若命令無法使用，先回到安裝或 PATH 問題排查。
4. 在 Extensions 安裝發行者 Microsoft 的 Python。開啟 python/01_basics.py，再用 Python: Select Interpreter 選到已安裝的解譯器。

```
python --version
# 若使用 Python install manager，可查已安裝版本：
py list
```

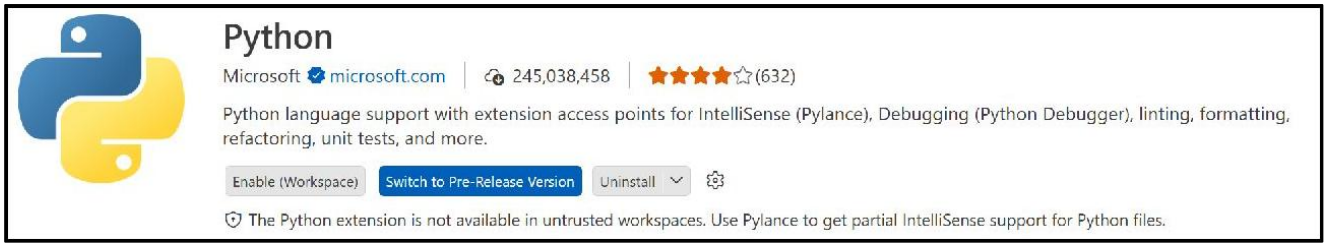


圖 5-1　Microsoft Python 擴充套件提供 VS Code 整合；Python 主程式須先安裝。

實機截圖：本課範例；畫面可依正文命令操作。

舊版 Windows Python launcher 的查詢方式可能是 py -0，不支援新的 py list。若只是查版本，可先使用 python --version。下一節以 Python: Create Environment 建立專案 .venv，再選取該環境。程式內的 sys.executable 是確認實際執行路徑的重要依據。

練習與自我檢核　寫下你採用的是傳統安裝程式或 install manager、Python 版本與執行路徑。確認不是只有 Extensions 顯示 Installed。

延伸查閱：Python Windows 官方下載　|　Using Python on Windows　|　Getting Started with Python in VS Code

第 5 章　在 VS Code 寫 Python

建立並選取 Python 環境

學習目的：知道目前使用哪一個 Python，並把本專案的套件分開管理。

Python 擴充套件與 Python 解譯器是兩個安裝項目。解譯器負責執行程式，擴充套件提供執行入口、語法提示和除錯整合。專案中的 .venv 是虛擬環境，能讓此課程需要的套件與其他專案分開。

1. 確認已安裝 Python 3 與 Microsoft 的 Python 擴充套件。以 VS Code 開啟整個 1151-vscode-lab。
2. 開啟命令面板，執行 Python: Create Environment，選 Venv，再選擇電腦上已安裝的 Python。依畫面選取 requirements.txt 安裝依賴，或稍後於 Terminal 安裝。
3. 開啟 python/01_basics.py，執行 Python: Select Interpreter，選本專案 .venv。Windows 執行檔通常位於 .venv/Scripts/python.exe。
4. 儲存並執行下列程式，核對輸出的執行檔路徑屬於本專案環境。

```
import sys
print(sys.executable)
print(sys.version)
```

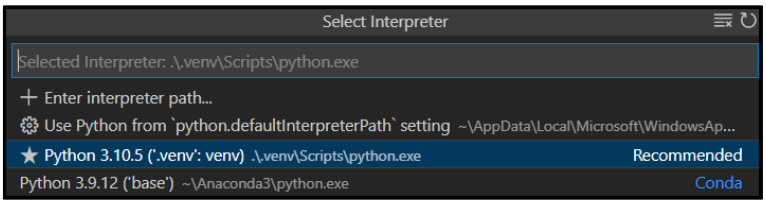


圖 5-2　Microsoft 官方解譯器選擇畫面；路徑與版本是文件範例，請選自己的專案 .venv。

官方示範截圖；Python 解譯器選擇

看見 .venv 資料夾仍不代表目前檔案已使用它。若終端機顯示找不到套件，先比對 sys.executable，再決定安裝位置。不要為了消除錯誤而向每個 Python 重複安裝。教師示範機的絕對路徑只是示例，學生應選自己的安裝位置。

練習與自我檢核　記錄本機 Python 版本與 sys.executable。關閉檔案再開啟，確認狀態列仍選到預期環境。

延伸查閱：Getting Started with Python in VS Code

第 5 章　在 VS Code 寫 Python

用 Python 完成同一個小練習

學習目的：以相同計算比較 R 與 Python，養成整檔執行與明確輸出的習慣。

開啟 python/01_basics.py。先預測每一行結果，再使用編輯器右上方的 Run Python File 或命令面板的 Python: Run Python File in Terminal 執行整份程式。輸出應出現在 Terminal；有多個執行按鈕時，請確認選到 Python 擴充套件的命令。

```
from statistics import mean
print(8 + 2)
print(8 - 2)
print(8 * 2)
print(8 / 2)
print(2 ** 3)
print("Hello, VS Code!")
scores = [65, 72, 80, 88, 95]
print(mean(scores))
```



圖 5-3　Microsoft 官方 Python terminal 輸出示例；畫面是 hello.py，本課程請執行 python/01_basics.py。

官方示範截圖；Python terminal 輸出

依序會得到 10、6、16、4.0、8、問候語及平均數 80。Python 的 / 回傳浮點數，4.0 與 R 顯示的 4 在這裡代表同一數值。次方使用 **；^ 是另一種運算，不能把 R 的次方寫法直接貼進 Python。

概念	R	Python
指定變數	scores <- c(...)	scores = [...]
平均數	mean(scores)	from statistics import mean
元素數量	length(scores)	len(scores)
第一個元素	scores[1]	scores[0]

練習與自我檢核　修改五個分數並重跑。再將字串中的括號或引號故意刪掉一個，讀取錯誤位置後修正；保持 # 註解與程式碼的用途清楚。

延伸查閱：Python tutorial An Informal Introduction

第 5 章　在 VS Code 寫 Python

讀取與 R 相同的 CSV

學習目的：理解文字檔讀取後仍要轉換型態，並讓檔案路徑更穩定。

Python 範例直接讀取前章匯出的 CSV，不另外下載同名資料。內建 csv.DictReader 以欄名取值，讀入的數值起初仍是文字，因此本例用 float() 將 mpg 轉成數字。先理解資料從哪裡來，再使用更高階的資料分析套件。

```
import csv
from pathlib import Path
from statistics import mean, median
project = Path(__file__).resolve().parents[1]
with (project / "data/raw/mtcars.csv").open(
        encoding="utf-8", newline="") as f:
    rows = list(csv.DictReader(f))
mpg = [float(row["mpg"]) for row in rows]
print(len(rows), len(rows[0]))
print(mean(mpg), median(mpg))
```

1. 開啟 python/02_histogram.py，先閱讀程式開頭的資料讀取部分。確認路徑包含 data/raw/mtcars.csv。
2. 用整檔執行方式執行這段程式。__file__ 指目前程式檔，本例從它的上兩層找到專案根目錄；不要直接貼到沒有 __file__ 的互動提示字元。
3. 核對資料有 32 列、12 欄，平均數 20.090625、中位數 19.2；這與 R 結果一致。

with 區塊結束後會關閉檔案；縮排表示區塊範圍。list comprehension 的意思是「逐列取出 mpg，再轉成浮點數」。若有空字串或非數值文字，float() 可能失敗；此時先檢查原始資料，不要不加說明地刪除資料列。FileNotFoundError 則應先檢查路徑和檔名。

練習與自我檢核　另建立 wt 數列，核對長度也是 32。說明一個數字欄位被誤讀成文字時，可能對排序和計算造成什麼影響。

延伸查閱：Python csv module　|　R datasets mtcars

使用 Matplotlib 畫直方圖

學習目的：用明確 bins 產生同一分布的圖，並在 VS Code 中檢視儲存結果。

python/02_histogram.py 用 NumPy 計數、Matplotlib 畫圖；請先在 .venv 安裝兩套件。本例用 Agg 後端儲存 PNG，再由 Explorer 開啟。

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
edges = [10, 15, 20, 25, 30, 35]
fig, ax = plt.subplots()
ax.hist(mpg, bins=edges, color="steelblue", edgecolor="white")
ax.set(xlabel="Miles per US gallon", ylabel="Frequency",
       title="Fuel economy of 32 cars")
fig.tight_layout()
fig.savefig(project / "outputs/mpg_hist_python.png", dpi=150)
plt.close(fig)
```

1. 使用 Python: Run Python File in Terminal 執行完整 python/02_histogram.py；完整檔案包含前頁的讀取資料步驟，以及建立 outputs 的指令。
2. 開啟 outputs/mpg_hist_python.png。將編輯器分割，可把程式和圖片放在兩側對照。
3. 修改顏色、標題或圖形尺寸，儲存並重新執行，確認圖片更新。

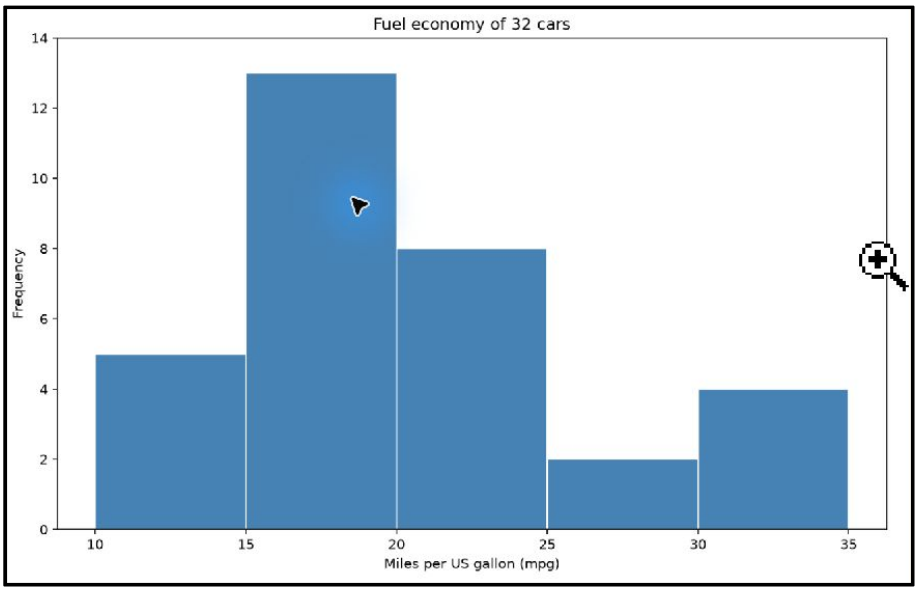


圖 5-4　VS Code 圖片預覽局部：Python 直方圖儲存為 PNG 後開啟，保留相同組界及完整座標軸。

實機截圖：本課範例；畫面可依正文命令操作。

bins=5 只指定組數，不保證組界相同。請先比較完整 bins，再核對柱高；互動視窗則使用 plt.show()，顯示方式會受圖形後端影響。

練習與自我檢核　將 y 軸標籤改為 Number of cars，再確認圖形仍是頻數而非密度。為什麼只改標籤文字不足以改變統計量？

延伸查閱：Matplotlib hist　|　NumPy histogram

跨語言核對結果

學習目的：用資料、分組和數值驗證比較結果，避免只憑圖片外觀判斷。

R 與 Python 畫出的圖可能字型、留白與刻度不同；驗證時應先比較數值。這兩份程式讀取同一 CSV，使用同一組界，計算的是次數而非密度。因此各組次數應一致，平均數與中位數也應相符。

檢查項目	本教材基準	代表什麼
資料列數	32	沒有多讀或漏讀車款
mpg 平均數	20.090625	以所有 32 筆數值計算
mpg 中位數	19.2	排序後中央位置的代表值
五組次數	5、13、8、2、4	組界與左右端點規則一致
次數總和	32	所有數值都落在指定分組內

```
import numpy as np
counts, boundaries = np.histogram(mpg, bins=[10,15,20,25,30,35])
print(counts.tolist())
assert counts.sum() == len(mpg)
assert counts.tolist() == [5, 13, 8, 2, 4]
```

NumPy 的分組是左閉右開，最後一組也包含最右端點。本教材的 R hist() 因此明確指定 right = FALSE、include.lowest = TRUE。若要對照邊界，可用 10、15、20、25、30、35 六個數值測試，兩者應得到 1、1、1、1、2。接近邊界的浮點數還可能受 R 的數值容忍設定影響；真實資料比較應同時保存組界與計數。

進階練習可執行 python/03_regression.py。它使用最小平方法核對 R 的 lm(mpg ~ wt) 係數；截距約 37.28512617、斜率約 −5.34447157、R 平方約 0.75283279。Python 圖只畫散佈點與迴歸線，沒有信賴帶，因此不要把兩圖的視覺元素當成完全相同。

練習與自我檢核　把一個測試值從 14.9 改為 15，指出它移到哪一組。請用計數證明，而不只看柱子的高度。

延伸查閱：NumPy histogram　|　R graphics hist

第 5 章　在 VS Code 寫 Python

安裝套件並留下環境紀錄

學習目的：把「可以在我電腦執行」整理成別人能重現的環境說明。

基本四則運算與 csv、statistics、pathlib 使用 Python 內建功能，不需要另裝套件。繪圖範例才需要 NumPy 與 Matplotlib。安裝前先確認目標解譯器；同一台電腦可能有多個 Python，使用錯誤的 pip 是常見問題。

1. 在已選取 .venv 後建立新的 Terminal，確認 python -c "import sys; print(sys.executable)" 的結果。
2. 於專案根資料夾執行下列套件安裝。若終端機尚未啟用虛擬環境，可在 Windows 改用 .venv/Scripts/python.exe 取代 python。
3. 安裝完成後執行 python/02_histogram.py。再使用下列版本查詢，將結果記到自己的筆記。

```
python -m pip install -r requirements.txt
python -c "import sys; print(sys.executable)"
python -c "import numpy, matplotlib; print(numpy.__version__, matplotlib.__version__)"
```

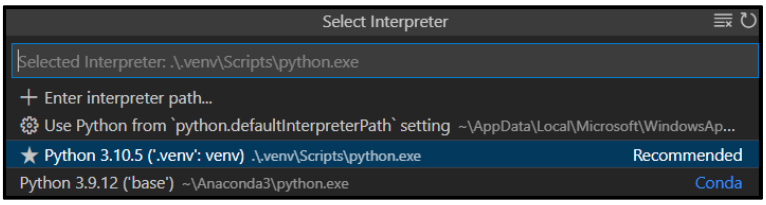


圖 5-5　Microsoft 官方解譯器選擇畫面；路徑與版本是文件範例，請選自己的專案 .venv。

官方示範截圖；Python 解譯器選擇

requirements.txt 表達本教材的相依套件範圍；requirements-tested.txt 記錄製作時實際測試的完整版本。製作環境為 Python 3.12.14、NumPy 2.5.4、Matplotlib 3.11.2。學生的修訂版本不同不代表錯誤，但若結果或安裝出現差異，版本資訊能協助定位。不要把整個 .venv 交給同學，應交付套件清單並讓對方重建環境。

練習與自我檢核　關閉目前 Terminal 後開啟新的 Terminal，再核對 Python 路徑。說明為何「已安裝套件」仍可能出現 ModuleNotFoundError。

延伸查閱：Getting Started with Python in VS Code

第 6 章　編輯與擴充套件

讓提示與大綱協助閱讀程式

學習目的：使用語法提示、滑鼠說明與 Outline 導覽，同時辨別提示所需的語言服務。

程式碼提示可以補完名稱、顯示參數及說明，但內容取決於語言擴充套件與語言服務。R 的語法著色不代表所有提示功能都已備妥；Python 也需要選對解譯器。先開啟正確副檔名，再觀察提示，比在純文字檔裡猜測設定更有效。

1. 開啟 python/04_debug.py，將游標放在 average_score 函數名稱上方，等待語言服務的說明視窗。
2. 在程式中輸入名稱開頭，查看建議清單；必要時按 Ctrl+Space 觸發建議。先讀清楚選項再接受。
3. 展開 Explorer 下方的 Outline，點選 average_score，觀察編輯區跳到函數定義的位置。
4. 按 Ctrl+Shift+O 查看目前檔案符號。若清單空白，先確認語言支援及程式是否包含可辨認的符號。

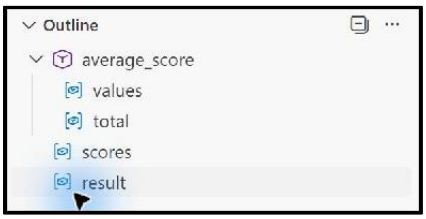


圖 6-1　本機實拍：python/04_debug.py 的 Outline 列出 average_score 函數，可點選名稱跳到定義。

實機截圖：本課範例；畫面可依正文命令操作。

預期結果是找到特定函式與其定義，不必逐行捲動。提示看似合理仍需確認，例如相近函式可能採不同預設值。紅色波浪線也可能來自靜態檢查，不一定就是執行結果。先讀 Problems 的訊息，再決定是語法、套件或環境問題。

練習與自我檢核　在自己寫的分析程式中找出一個函式，說明它的輸入與輸出，再利用 Outline 或符號搜尋返回定義。

延伸查閱：IntelliSense　|　Code navigation

第 6 章　編輯與擴充套件

用分割畫面與快捷鍵修改程式

學習目的：並排比較兩份程式，運用註解與多選取提高編輯效率。

學習 R 與 Python 時，並排閱讀相同任務的程式很有幫助。分割編輯區可以同時顯示兩個檔案，也可以顯示同一檔案的不同位置。這只改變顯示方式；編輯同一個檔案的其中一個視窗，內容也會反映在另一個視窗。

1. 先開啟 R/03_histogram.R，按 Ctrl+\ 分割編輯區，再在右側開啟 python/02_histogram.py。
2. 比較讀取同一份 CSV、設定分組界線與畫直方圖的寫法；修改前先確認游標目前在左側或右側。
3. 選取一行註解說明，按 Ctrl+/ 切換行註解。R 與 Python 都可使用 #，但其他語言的註解方式可能不同。
4. 選取一個變數名稱後按 Ctrl+D，逐次加入下一個相同文字。確認選取範圍後才輸入新名稱；按 Esc 結束多選取。

R：讀取 CSV、指定組界並計算次數

```r
cars <- read.csv("data/raw/mtcars.csv", stringsAsFactors = FALSE)
stopifnot(nrow(cars) == 32, is.numeric(cars$mpg), !anyNA(cars$mpg))
edges <- c(10, 15, 20, 25, 30, 35)
h <- hist(cars$mpg, breaks = edges, right = FALSE,
          include.lowest = TRUE, plot = FALSE)
```

Python：讀取相同 CSV，使用相同組界

```python
project = Path(__file__).resolve().parents[1]
with (project / "data/raw/mtcars.csv").open(encoding="utf-8", newline="") as f:
    rows = list(csv.DictReader(f))
mpg = [float(row["mpg"]) for row in rows]
edges = [10, 15, 20, 25, 30, 35]
counts, _ = np.histogram(mpg, bins=edges)
```

圖 6-2　VS Code 分割編輯器的兩個局部（上：R；下：Python）：對照讀檔、共同組界與直方圖次數計算。

實機截圖：本課範例；畫面可依正文命令操作。

多選取是文字編輯，不保證理解程式語意。若相同字樣也出現在圖標題或另一個變數中，可能被一起改掉。對語言服務支援的符號，可使用 Rename Symbol，並預覽變更；不確定支援情況時，以搜尋後逐項修改較容易核對。誤改可先用 Ctrl+Z 復原，再重新縮小範圍。

練習與自我檢核　把一個說明註解加入 R 與 Python 範例，儲存兩份檔案；比較兩側是否都保留了可執行的程式。

延伸查閱：Basic editing multi cursor and comments　|　VS Code custom layout split editors

先搜尋再取代

學習目的：區分目前檔案與整個工作區的搜尋，安全地修改重複文字。

同一個函數名稱可能出現在定義與呼叫的多個位置。Ctrl+F 搜尋目前檔案，Ctrl+Shift+F 搜尋工作區；這兩個範圍不同。改名之前先查看所有命中位置，避免只改定義卻留下舊的呼叫。以下以本課 R/05_debug.R 與 python/04_debug.py 共有的 average_score 為例，示範搜尋與取代預覽。實際取代前保留副本；練習後恢復 average_score，方便接續第 7 章除錯範例。

1. 開啟 R/05_debug.R，按 Ctrl+F 搜尋 average_score，逐一查看目前檔案中的函數定義與呼叫。
2. 按 Ctrl+Shift+F 搜尋 average_score，確認結果包含 R/05_debug.R 與 python/04_debug.py；點選結果跳到對應程式行。
3. 展開 Replace 欄位並輸入 mean_score，先查看紅色刪除與綠色新增的預覽。此時尚未完成取代，不要把預覽當成已儲存的修改。
4. 確認範圍後逐一取代函數定義與呼叫，儲存兩份檔案，再分別執行 R 與 Python 程式。檢查名稱是否一致，以及結果是否維持正確。

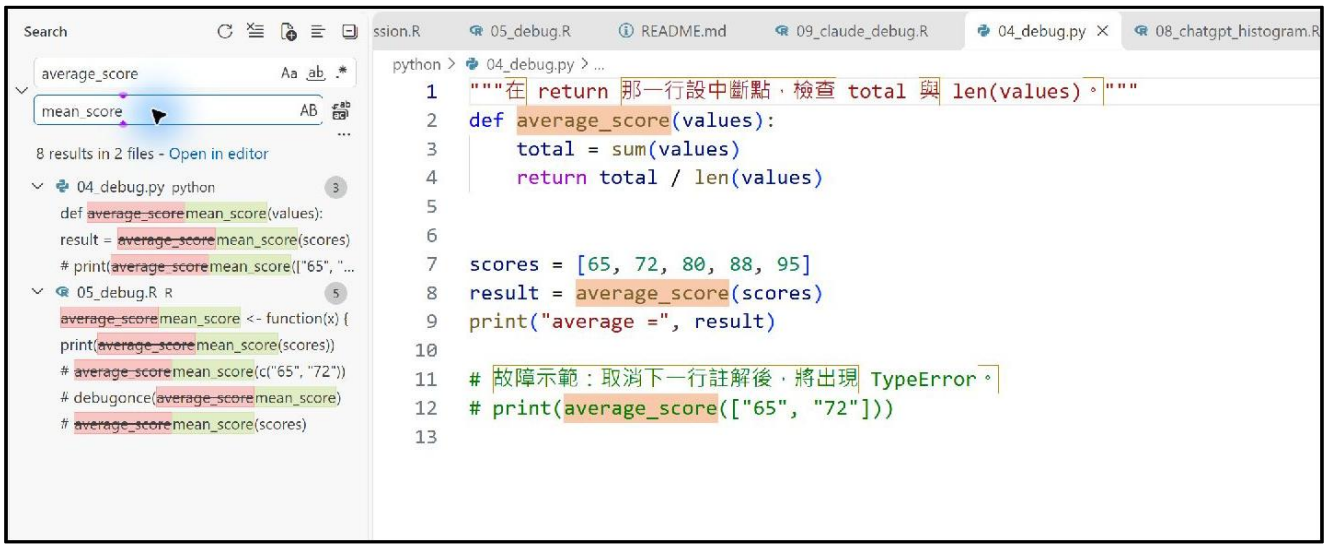


圖 6-3　本機實拍：搜尋 average_score，取代欄輸入 mean_score；紅刪綠加顯示預覽，此畫面尚未套用取代。

實機截圖：本課範例；畫面可依正文命令操作。

Aa 代表區分大小寫，整字比對可避免把名稱片段一起找出，正規表示式則適合進階搜尋。初學先用一般文字搜尋。若找不到已知內容，檢查搜尋範圍、排除規則及檔案是否儲存。取代會修改真正的程式內容，不能只看命中數減少就判定成功。

練習與自我檢核　將一個圖檔名稱改成 outputs/mpg_hist_r_v2.png。先列出所有引用，再逐項更新；確認程式輸出位置與 README 說明一致。

延伸查閱：Search across files

第 6 章　編輯與擴充套件

安裝與整理擴充套件

學習目的：核對套件識別與發行者，完成安裝、停用及問題排查。

擴充套件讓 VS Code 增加語言、偵錯或 AI 能力。搜尋結果可能有相近名稱，不能只看圖示。先核對發行者、完整識別碼與說明，再安裝本課需要的套件。R / Python 的實際執行設定見語言章，Codex 與 Claude Code 的登入及使用見 AI 章。

1. 按 Ctrl+Shift+X 開啟 Extensions，以完整識別碼搜尋套件，點進詳細頁讀取功能與需求。

2. 確認發行者後選 Install；若顯示重新載入提示，儲存檔案後依提示操作。

3. 開啟對應的程式檔，確認語言命令或功能出現。安裝成功不代表 R / Python 主程式已安裝。

4. 要排查衝突時，從套件齒輪選 Disable (Workspace) 只在目前工作區停用；確定不需要才選 Uninstall。

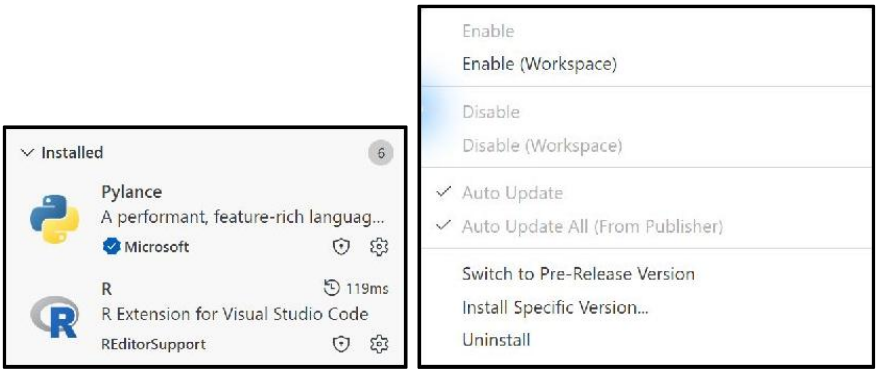


圖 6-4　左：本機實拍：Extensions 的已安裝清單顯示名稱與發行者；點選套件可進入詳細頁。

圖 6-5　右：本機實拍：擴充套件管理選單的局部畫面，可選擇停用或移除；先確認影響範圍再操作。

實機截圖：本課範例；選單項目是否可用會隨套件與工作區狀態改變。

用途	搜尋用識別碼
R	REditorSupport.r
Python	ms-python.python
Codex	openai.chatgpt
Claude Code	anthropic.claude-code

預期結果是能從 @installed 找到已安裝項目。若功能未出現，檢查是否被停用、目前工作區是否受限，以及套件是否要求額外安裝。不要同時安裝多個功能重複的「一鍵執行」套件，否則同一按鈕可能使用不同的執行環境，增加判讀難度。

練習與自我檢核　列出本課目前已安裝的套件及用途；挑出一個暫時用不到的項目，說明停用與移除的差別。

延伸查閱：Extension Marketplace　|　R extension　|　Python extension

第 7 章　程式執行與除錯

先讀錯誤再修改程式

學習目的：區分語法、執行與邏輯問題，建立可說明的除錯步驟。

除錯的第一步是縮小問題。先確認執行的是哪一份檔案、檔案是否已儲存、目前解譯器與工作目錄，再讀取完整錯誤。不要一次改動多個地方，否則即使程式能執行，也難以知道是哪一個修改有效。

類型	例子	先檢查
語法錯誤	引號未成對、括號未關閉、Python 縮排不一致	錯誤行與前一行
執行錯誤	檔案不存在、套件未安裝、文字不能加總	路徑、環境與資料型態
邏輯或統計錯誤	平均數分母寫錯、單位混淆、把數量當密度	手算、摘要統計與分析定義

1. 重跑能產生問題的最小片段，保留完整訊息與行號。
2. 先說明預期結果，再用 print()、str() 或 type() 查看實際資料。
3. 一次修正一個原因，重新執行，最後在新工作階段從頭跑整份程式。

```
x <- c(65, 72, 80, 88, 95)
print(sum(x))
print(length(x))
print(sum(x) / length(x))
```

上例總分為 400，人數為 5，平均數應為 80。若誤寫成 sum(x)/4，程式仍能執行並得到 100；這是邏輯錯誤，通常不會有紅色錯誤提示。Problems 面板與語法檢查能協助找出部分問題，無法替你判斷統計方法是否合理。

練習與自我檢核　把本例分母改成 4，寫出它為何錯誤，再修正。向 AI 求助時一併提供程式、完整錯誤或異常結果，以及你預期的數值。

延伸查閱：R traceback　|　Python debugging in VS Code

第 7 章　程式執行與除錯

用 traceback 追蹤 R 的錯誤

學習目的：辨認出錯函數與傳入的資料，依呼叫過程定位問題。

開啟 R/05_debug.R。本例定義平均分數函數，正常輸入是數值向量；若傳入帶引號的分數，就會在 sum() 發生型態錯誤。請在同一個 R terminal 逐段執行，並在錯誤出現後立即呼叫 traceback()。

```
average_score <- function(x) {
  total <- sum(x)
  total / length(x)
}
average_score(c("65", "72"))
# 等錯誤出現後，再單獨執行：
traceback()
```

1. 先執行函數定義，確認沒有語法錯誤。

2. 單獨執行字串輸入的呼叫，閱讀訊息中關於 character 的描述。

3. 接著執行 traceback()，從呼叫清單找出 average_score() 與 sum() 的關係，再檢查輸入的型態。

4. 改用 average_score(c(65, 72))，應得到 68.5；最後以五個分數重跑，應回到 80。



圖 7-1　VS Code 中的 R 除錯起始檔。取消錯誤呼叫的註解後，依本文在 R terminal 執行與追蹤。

實機截圖：本課範例；畫面可依正文命令操作。

traceback() 描述最近一次未被處理的錯誤；若把錯誤包在 try() 或 tryCatch() 裡，未必能得到這次錯誤的追蹤資訊。不要先用另一個錯誤蓋掉紀錄。看到文字型態也不應盲目全部轉為數字；若欄位本來是車名或類別，轉型會造成新問題。

練習與自我檢核　在呼叫前加上 str(x) 需要放在哪裡？請用自己的話解釋函數內的 x 與外部 scores 之間的關係。

延伸查閱：R traceback

逐步檢查 R 函數中的變數

學習目的：暫停在函數內觀察變數，而不是只看最後結果。

R 自己就提供函數除錯工具，不必先安裝另一套圖形除錯擴充。debugonce() 會讓指定函數的下一次呼叫進入逐步檢查；接著可在 Browse 提示字元查變數、執行下一步或繼續完成。這與一般 R 提示字元的操作情境不同。

```
source("R/05_debug.R")
debugonce(average_score)
average_score(c(65, 72, 80, 88, 95))
# 在 Browse 提示字元依序輸入：
# n
# print(x)
# n
# print(total)
# c
```

1. 先載入 R/05_debug.R，再執行 debugonce() 與函數呼叫；確認提示字元進入 Browse 狀態。

2. 用 n 前進，並用 print(x) 查看五個分數。total 的指定完成後，才檢查 print(total)。

3. 核對 total 為 400、length(x) 為 5。使用 c 繼續，正常結果為 80；需要離開這次除錯時可用 Q。



圖 7-2　在 VS Code 編輯本課程的 R 除錯範例；依本文流程在 R terminal 逐步操作。

實機截圖：本課範例；畫面可依正文命令操作。

debugonce() 只影響下一次呼叫；debug() 則會持續啟用，直到 undebug()。也可以在自己撰寫的函數內加入 browser()，但完成除錯後應移除，避免同學整檔執行時意外停住。圖形化 R 除錯另有擴充套件與相依條件，本入門手冊先採 R 原生方法。

練習與自我檢核　把函數暫時改成 total / 4，用逐步檢查指出輸入和 total 都正確，錯的是哪一個運算。修正後重新執行。

延伸查閱：R debug and debugonce

使用 Python 中斷點

學習目的：在 VS Code 的 Run and Debug 中觀察函數與區域變數。

Python 除錯由 Python Debugger 擴充套件支援。先選取正確解譯器，再開啟 python/04_debug.py。中斷點讓程式在某行執行前暫停；不是刪掉該行，也不是讓程式出錯。

1. 在 return total / len(values) 左側的行號邊緣按一下，看到紅色圓點。
2. 開啟 Run and Debug，選 Python: Debug current file；若首次設定，選 Python Debugger 與目前 Python 檔案。配套專案已提供 .vscode/launch.json。
3. 啟動後確認黃色標示停在 return 行。於 Variables 展開 values，檢查 total 為 400，values 有五個元素。
4. 使用 Step Over 前進，或 Continue 完成；Terminal 應印出 average = 80.0。完成後移除中斷點並再執行一次。

```
def average_score(values):
    total = sum(values)
    return total / len(values)

scores = [65, 72, 80, 88, 95]
result = average_score(scores)
print("average =", result)
```

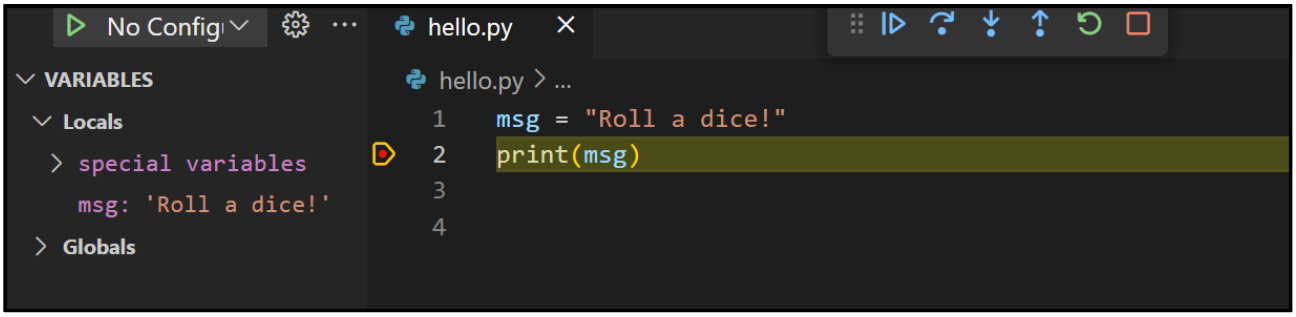


圖 7-3　Microsoft 官方中斷點示例，以 hello.py 顯示 Variables、目前行與除錯工具列；本課程改用平均分數函數。

官方示範截圖；Python 中斷點與區域變數

Step Over 執行目前這一行並移到下一行；Step Into 可進入函數；Continue 跑到下一個中斷點或結束。Debug Console 可在暫停時計算 len(values)，但不要把它當一般 PowerShell。若中斷點呈空心或未停下，先確認檔案已儲存、執行的是目前檔案，且不是一般 Run。

練習與自我檢核　將分母暫時改為 len(values) - 1，在中斷點預測回傳值，再執行核對。修正後保存完整的除錯紀錄。

延伸查閱：Python debugging in VS Code

第 8 章　用 AI 協作撰寫 R

把統計問題寫成清楚的提示語

學習目的：能提供資料、任務、環境與驗收條件，讓 AI 產出的程式可在自己的電腦重現。

本章延續課堂「向 AI 提問，再自己執行與判讀」的練習。ChatGPT、Claude 與 Gemini 都可以透過網頁協助撰寫 R；本章使用一般文字對話，不要求 API 金鑰，也不先安裝三套 VS Code AI 擴充套件。網頁回覆不會自動成為本機的 R 程式，仍須自行存檔、執行並核對結果。各平台的模型與可用額度依帳號及時間而異，練習時記錄實際顯示的名稱即可。

提示語要素	本課的具體寫法
資料	使用 R 內建 mtcars；不下載資料，不模擬另一組資料。
任務	畫 mpg 直方圖，並報告樣本數與平均數。
環境	Windows、VS Code、R；只使用 base R。
驗收	可從頭執行；有標題與座標軸單位；繁體中文說明。

```
我是 R 初學者，使用 Windows 與 VS Code。
請使用內建 mtcars，計算 mpg 的筆數與平均數，
再用 base R 畫直方圖。不要安裝套件或下載資料。
請給完整可執行程式，加入繁體中文註解，
並說明每一段的作用、預期結果及如何檢查。
```

先要求一個可理解的小成果，再逐步加入修改。若回覆出現看不懂的函數，可追問「請先解釋這一行的輸入與輸出，不要改寫整份程式」。若 AI 提議改用 Python、改換資料或增加套件，先確認是否符合本題。教材中的提示語是課堂練習；接下來的參考程式由教材編寫，並非宣稱某平台一定會回覆相同文字。

練習與自我檢核　課堂練習：把「幫我畫漂亮的圖」改寫成包含上述四項要素的提示語。與同學交換，檢查對方能否只憑提示語判定成果是否完成。

延伸查閱：Use ChatGPT　|　Get started with Claude　|　Gemini Apps 帳號條件

第 8 章　用 AI 協作撰寫 R

使用 ChatGPT 完成 mpg 直方圖

學習目的：把網頁對話中的程式帶回 VS Code，完成存檔、執行與結果核對。

1. 開啟 https://chatgpt.com，依畫面登入，建立新對話。貼上前頁提示語；本題不必上傳個人檔案。
2. 閱讀回覆，確認使用 mtcars$mpg 與 hist()。複製程式區塊的內容，不要把 Markdown 的三個反引號或「以下是程式」一起複製。
3. 回到 1151-vscode-lab，開啟配套檔 R/08_chatgpt_histogram.R；未使用範例包才新增此檔。貼上完整程式取代練習起始內容，儲存並確認右下角語言為 R。
4. 以命令選擇區執行 R: Create R terminal。先執行資料與摘要，再執行繪圖；選取程式後按 Ctrl+Enter。
5. 核對筆數為 32，mpg 平均數約 20.09，圖的橫軸為每加侖英里數。把提示語、工具名稱、日期、修改處寫入 notes/ai_log.txt。

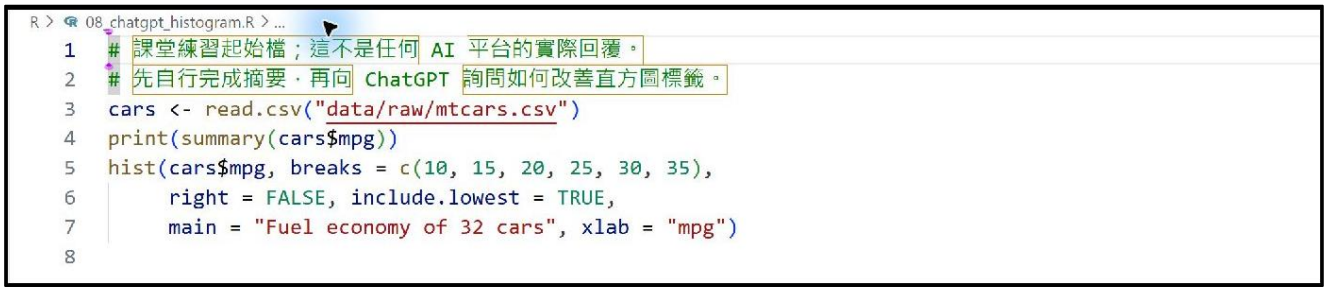


圖 8-1　VS Code 中的直方圖練習起始程式；畫面為本課原始碼，不是 ChatGPT 實際回覆或執行結果

實機截圖：本課範例；畫面可依正文命令操作。

預期成果是一個可重新開啟的 .R 檔與可重現的圖形。若只有文字回覆，尚未完成執行；若圖未出現，先看 R 終端機是否有錯誤，再檢查圖形檢視器。顯示正常但平均數不同時，檢查是否誤用了其他欄位、篩選了資料，或沿用前次對話的物件。

練習與自我檢核　追問範例：「請只把直方圖改為淺藍色，保留資料、計算方式與其他程式，並指出改了哪一行。」修改後重新執行，確認筆數與平均數沒有改變。

延伸查閱：Use ChatGPT　|　R in Visual Studio Code　|　R 官方 mtcars 資料說明

第 8 章　用 AI 協作撰寫 R

使用 Claude 協助修正 R 錯誤

學習目的：提供最小程式與完整錯誤訊息，判斷 AI 建議是否真正解決問題。

先開啟配套檔 R/09_claude_debug.R；未使用範例包才新增此檔。在新 R 工作階段逐段執行下列有意安排的錯誤，再將實際訊息帶到 https://claude.ai 的新對話。此處使用 Claude 網頁，與第十章的 Claude Code 擴充套件不同。

```
cars <- read.csv("data/raw/mtcars.csv")
hist(mpg, main = "Fuel economy", xlab = "Miles per gallon")

我在 Windows 的 VS Code 執行 R，想畫 mtcars 的 mpg 直方圖。
程式如下：[貼上完整兩行程式]
錯誤訊息如下：[貼上 R 實際顯示的訊息]
請先解釋原因，再給最小修改。不要使用 attach()，
不要安裝套件，最後說明如何驗證修正成功。
```

1. 將方括號提示換成實際內容後送出。不要只問「怎麼不能跑」。

2. 檢查建議是否明確從讀入的 cars 取出 mpg，例如 cars$mpg。先說明原因，再改一處並儲存。

3. 在 VS Code 重新執行兩行程式。確認沒有錯誤且圖形出現，並把錯誤原因用自己的話記錄。

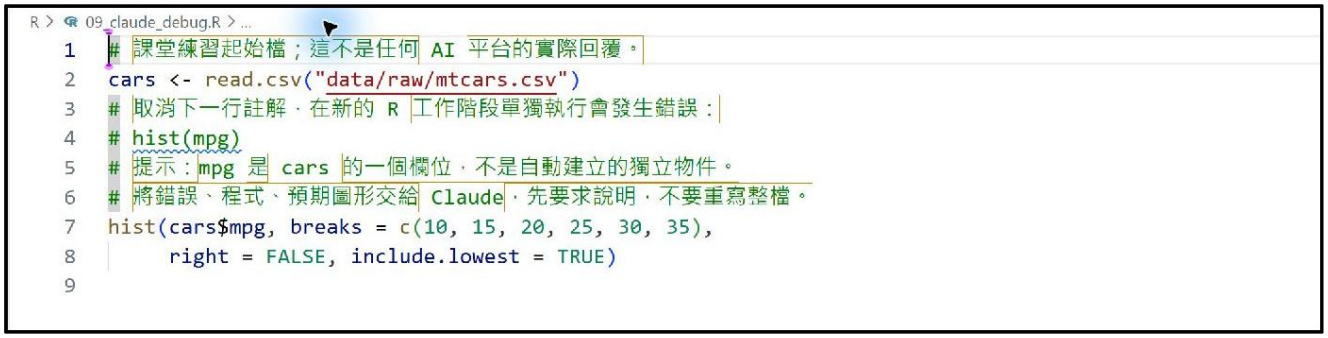


圖 8-2　VS Code 中的 R 除錯練習原始碼；故障呼叫以註解保存，依本文逐段執行後再複製實際錯誤訊息

實機截圖：本課範例；畫面可依正文命令操作。

預期理解：資料框中的欄位不一定是全域環境中獨立的物件。若先前剛好建立過 mpg，這段程式可能不報錯卻使用舊資料；因此應開新 R 工作階段重做。AI 回覆若只建議重裝軟體，先檢查它是否處理了物件名稱與資料來源。

延伸查閱：Get started with Claude　|　R in Visual Studio Code

第 8 章　用 AI 協作撰寫 R

使用 Gemini 撰寫散佈圖程式

學習目的：以同一份資料探索兩個變數的關係，並核對單位與圖形解讀。

1. 開啟 https://gemini.google.com，依畫面登入並建立新對話。學校帳號若無法使用，可能需要校方管理員啟用服務。
2. 送出下列提示語，先閱讀程式是否保留 mtcars，並確認橫軸是 wt、縱軸是 mpg。
3. 開啟配套檔 R/10_gemini_scatter.R；未使用範例包才新增此檔。貼上完整程式並儲存，在 R 終端機執行；資料有 32 個觀測，重疊的點可能看起來較少。

```
請用 R 內建 mtcars 和 base R 畫散佈圖。
橫軸 wt，標明 1000 lbs；縱軸 mpg，標明 miles per US gallon。
使用實心藍色點，加上清楚標題，不加入迴歸線。
請提供完整 R 程式與繁體中文註解，再用兩句話描述圖形。
描述只談關聯，不把觀察到的關係說成因果。
```



圖 8-3　VS Code 中的 wt 與 mpg 散佈圖練習原始碼；不含迴歸線，執行後再檢查座標軸與觀測點

實機截圖：本課範例；畫面可依正文命令操作。

預期成果：車重較大的觀測通常有較低的 mpg，這是此資料中的關聯描述。若 AI 將 wt 標成公斤，應要求更正；wt 的原始單位是千磅。若兩軸放反，修正 plot() 的 x、y 或公式順序，重新執行後確認。

版本提醒：Google 已公告自 2026 年 6 月 18 日停止 Gemini Code Assist 個人帳號的 IDE 擴充套件與 Gemini CLI 服務。本課採網頁產碼、VS Code 執行的流程；不要照舊影片假設免費 Gemini 擴充套件仍可使用。

延伸查閱：Gemini Apps 帳號條件　|　Gemini Code Assist 個人帳號停用公告　|　R 官方 mtcars 資料說明

用參考程式驗證 AI 的結果

學習目的：以可重現的資料摘要與圖形，建立獨立於 AI 回覆的檢查標準。

下列是教材參考程式，不是任何 AI 平台的實際回覆。開啟配套檔 R/11_ai_reference.R；未使用範例包才新增並貼上程式，再從頭執行。它只使用 R 內建資料與函數；先驗證資料，再畫圖，可減少舊物件或不完整執行造成的誤判。

```
# 教材參考程式：先檢查，再畫圖
data(mtcars)
n <- nrow(mtcars)
mean_mpg <- mean(mtcars$mpg)
print(n)
print(mean_mpg)
stopifnot(n == 32, !anyNA(mtcars$mpg))

hist(mtcars$mpg, breaks = seq(10, 35, by = 5),
     right = FALSE, include.lowest = TRUE,
     main = "Fuel economy of 32 cars",
     xlab = "Miles per US gallon", ylab = "Frequency",
     col = "lightblue", border = "white")

plot(mpg ~ wt, data = mtcars, pch = 19,
     col = "steelblue", main = "Weight and fuel economy",
     xlab = "Weight (1000 lbs)",
     ylab = "Miles per US gallon")
```

應得到 n = 32、mean_mpg = 20.090625。直方圖使用固定邊界及左含右不含分組（最後一組包含最右端），五組次數依序為 5、13、8、2、4，與本課 Python 範例一致。逐張執行繪圖指令，或使用圖形檢視器的歷史功能查看兩圖。

比較 AI 產出的圖時，先對齊資料、分組邊界與縱軸類型。hist() 預設分組可能不同；縱軸若是 Density，柱高也不能直接當次數。圖的顏色不同不代表結果錯誤，資料範圍、單位或總筆數不同才需要追查。

練習與自我檢核　驗證題：執行下列計數程式，核對五組次數，再解釋為何分組邊界與端點規則不同會影響柱高。此處不傳入顏色或標題等繪圖參數。

```
h <- hist(mtcars$mpg, breaks = seq(10, 35, 5),
          right = FALSE, include.lowest = TRUE, plot = FALSE)
h$counts
```

延伸查閱：R 官方 mtcars 資料說明　|　R hist 函數文件

第 8 章　用 AI 協作撰寫 R

比較三個 AI 並留下學習紀錄

學習目的：用相同標準評估程式品質，說清楚自己接受、修改與驗證了哪些內容。

選一個相同的直方圖提示語，分別交給當下可使用的 ChatGPT、Claude 與 Gemini。請保留原始要求，不要先替其中一個工具補充更多資訊。若遇到帳號限制或額度不足，可與同學分工，或將該欄標為「本次未測試」，不猜測結果。比較的是本次任務的可用性，不是為模型做永久排名。

檢核項目	完成的證據
資料與方法	確實使用 mtcars$mpg；沒有另造資料。
能夠重跑	在新的 R 工作階段從頭執行成功。
統計數字	筆數 32，平均數約 20.09。
圖形與單位	標題、橫軸、縱軸、分組方式合理。
可解釋性	能說明 hist()、資料欄位與主要參數。
修改紀錄	可指出自己改過的行及修改理由。

日期與時間：
工具與畫面顯示的模型：
完整提示語：
存放的 R 檔名：
第一次執行是否成功與錯誤訊息：
我的修改及理由：
重新執行的結果：
我能解釋的一個函數與仍有疑問的地方：

在 notes/ai_log.txt 為每次練習追加一筆紀錄，避免覆蓋前一次內容。貼給 AI 的訊息只需包含本題程式、公開資料及必要錯誤；不需要帳號密碼或無關學生資料。交作業時同時保留可執行程式與簡短紀錄，讓教師能看出你的理解與修正過程。AI 說「已測試」不等於你的電腦也已成功，最後仍須自己重跑。

練習與自我檢核　課堂成果：提交一份修正後的直方圖程式、圖形與 AI 紀錄。用約 100 字說明哪個回覆最容易理解、你修改了什麼，以及你如何確認圖形正確。

延伸查閱：R 官方 mtcars 資料說明

第 9 章　Git 與 GitHub

開始追蹤本機版本

學習目的：區分 Git 與 GitHub，在練習資料夾建立本機儲存庫。

Git 用來記錄檔案的版本；GitHub 是可存放 Git 儲存庫的線上服務。只使用本機 Git 不需要 GitHub 帳號。VS Code 已提供 Git 操作介面，但電腦仍須安裝 Git。這一節先練習本機提交，能清楚說明每一步的作用後，再了解雲端分享。

1. 從 git-scm.com 安裝 Windows 適用的 Git，完成後重新開啟 VS Code；選 Terminal → New Terminal 開啟 PowerShell，輸入 git --version 確認能顯示版本。
2. 若尚未設定提交作者，用下列指令設定自己的名稱與電子郵件。它們是提交紀錄中的作者資訊，不是登入密碼。
3. 以 Open Folder 開啟 1151-vscode-lab，確認不是誤開整個磁碟或使用者家目錄。
4. 按 Ctrl+Shift+G 開啟 Source Control，選 Initialize Repository；若按鈕不見，可由命令選擇區執行 Git: Initialize Repository。

```
git config --global user.name "Your Name"
git config --global user.email "your-email@example.com"
```

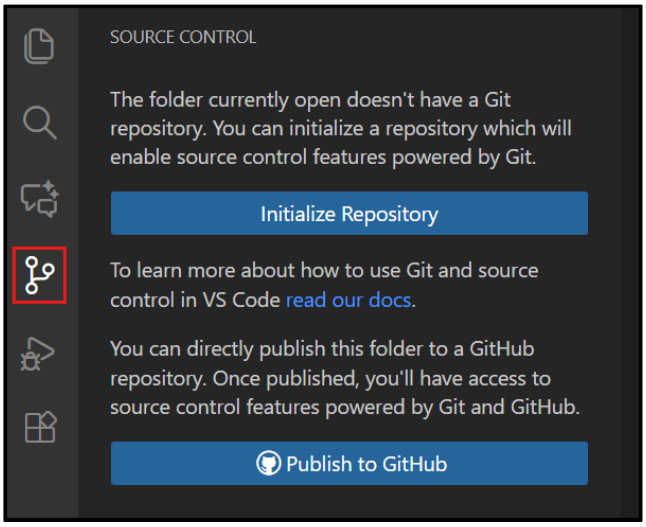


圖 9-1　官方畫面示意：Initialize Repository 建立本機版本庫；下方 Publish to GitHub 是另一個動作。

官方示範截圖；Source Control 的 Initialize Repository 按鈕

請把範例姓名與信箱換成自己的資料；若日後公開提交，作者資訊也會公開。預期結果是資料夾成為本機儲存庫，尚未有任何上傳。若看不到 Git 功能，先查 Git 安裝與工作區狀態，不必先申請 GitHub 帳號。

練習與自我檢核　回答：Initialize Repository 是否會把程式放到網路？目前提交作者的設定又與 GitHub 密碼有何不同？

延伸查閱：Source control quickstart　|　Git downloads

檢查修改並建立一次提交

學習目的：完成儲存、檢查差異、暫存與提交，讓每次版本紀錄有清楚意義。

儲存讓修改寫入檔案；Stage 選定下一次提交的內容；Commit 才把它記入本機歷史。用直方圖作業練習時，可以把「新增基本分析」與「修正圖形標題」分成兩次提交。這樣日後回看，較容易理解每次修改的目的。

1. 在 README.md 寫一行專案說明並儲存，返回 Source Control，查看 Changes 中的 U（未追蹤）或 M（已修改）。
2. 點選檔案檢查內容；對已追蹤檔案可看到差異。綠色通常表示新增、紅色表示刪除，但仍要讀文字確認。
3. 把游標移到要提交的檔案，按 +（Stage Changes），確認它移到 Staged Changes。
4. 輸入符合內容的訊息，例如 Add project README，再按 Commit。完成後已提交檔案應不再列為待處理修改。

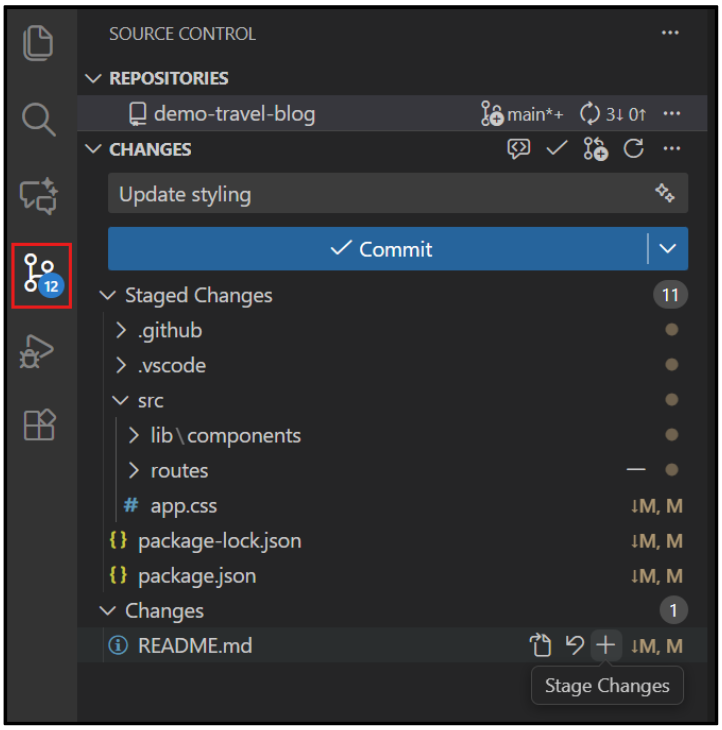


圖 9-2　官方暫存區局部畫面：Changes 與 Staged Changes 分別列出；以 + 暫存，再輸入訊息並提交。

官方示範截圖；暫存區與程式差異檢視

若暫存後又修改同一檔案，新的修改不一定已包含在暫存版本中，提交前要再檢查。按 – 取消暫存會保留編輯內容；Discard Changes 會丟棄修改，兩者不可混淆。若提交失敗且要求 user.name 或 user.email，回到前頁完成作者設定。

練習與自我檢核　為程式加入一段用途註解並提交。用自己的話說明這次提交包含哪些檔案，以及為何沒有把其他檔案一起提交。

延伸查閱：Staging and committing changes

查看歷史與排除暫存檔

學習目的：檢查提交紀錄，理解 .gitignore 的用途與限制。

提交後應確認歷史中真的留下紀錄。Source Control Graph 可查看整個儲存庫的提交與分支關係；Explorer 的 Timeline 則偏向某個檔案。查看舊版本只是閱讀，不必把工作資料夾切換到舊時間點，初學時先使用比較功能即可。

1. 在 Source Control 展開 Graph，找到剛建立的提交訊息；點選提交，再選檔案查看該次變更。

2. 再修改一行註解、儲存與提交，觀察圖上多出一筆紀錄。對照兩則訊息，確認能說明先後順序。

3. 在根資料夾新增 .gitignore，填入下列練習規則並儲存；查看原本未追蹤的暫存檔是否不再出現。

```
.venv/
__pycache__/
.Rhistory
.RData
.env
```

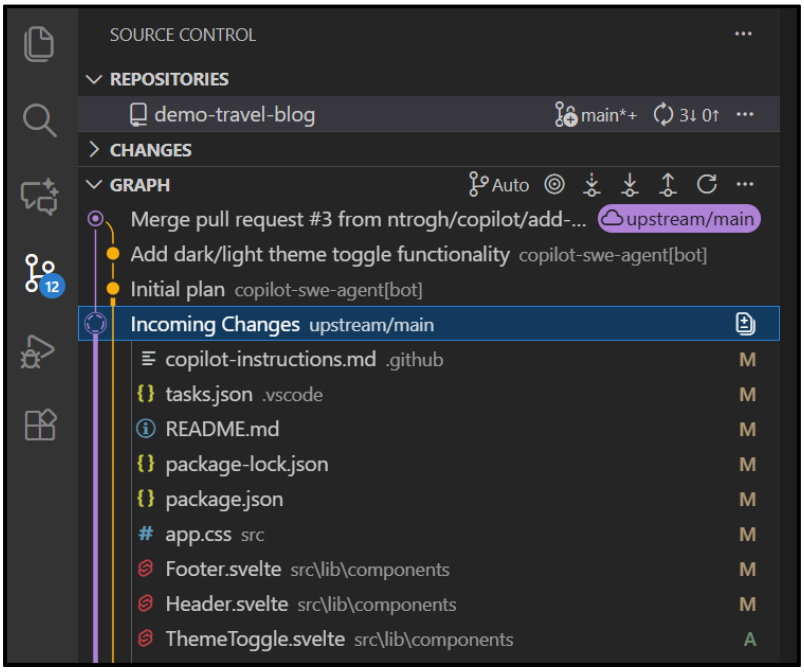


圖 9-3　官方畫面示意：Graph 顯示提交與分支；選取提交可以檢查相關檔案。

官方示範截圖；Source Control Graph 的提交紀錄與分支關係

這些規則排除虛擬環境、快取、R 指令歷史、R 物件快照與 .env 設定檔；程式碼和必要的套件需求仍要保存。小型教學圖檔是否追蹤可依作業規定決定，不必一律排除 outputs。.gitignore 只處理未追蹤檔案；已提交的內容不會因加入規則就從歷史消失。不要把密碼或金鑰寫入程式後再期待忽略規則補救。

練習與自我檢核　找到修改 README 與加入程式註解的提交，分別檢查其內容。確認要交出的程式與 README 都有被追蹤，而 .venv 沒有被納入。

延伸查閱：View source control history　|　gitignore documentation

第 9 章　Git 與 GitHub

理解發布與下載專案

學習目的：知道 Publish、Clone 與登入授權的作用，能規劃自己的 GitHub 練習。

準備線上分享時，先確認內容可分享、README 說明完整，並已完成本機提交。Publish to GitHub 會建立遠端儲存庫並上傳提交；Clone 則把已存在的遠端儲存庫複製到自己的電腦。兩者方向與用途不同。以下是操作教學，圖為官方示意。

1. 若要發布自己的練習，開啟命令選擇區執行 Publish to GitHub，依提示登入並返回 VS Code。選擇可用的名稱，課堂練習可先選 Private。
2. 發布前再次檢查待分享檔案；完成後開啟 GitHub 儲存庫頁面，核對 README、程式與提交訊息。
3. 若要下載教師提供的專案，使用 Git: Clone，貼上教師提供的儲存庫網址，選擇本機存放的上層資料夾。
4. 複製完成後開啟該資料夾，先讀 README 再執行。公開儲存庫通常可直接複製，能下載不代表有上傳修改的權限。

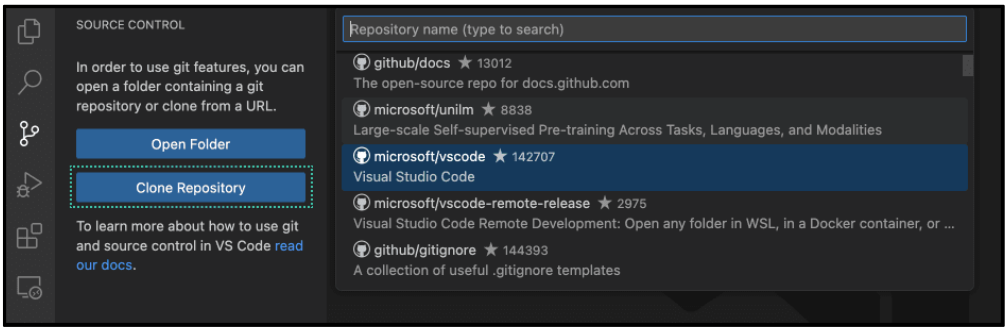


圖 9-4　官方畫面示意：Clone 可輸入來源或從 GitHub 清單選擇儲存庫。

官方示範截圖；Git Clone 的 GitHub 儲存庫選擇欄

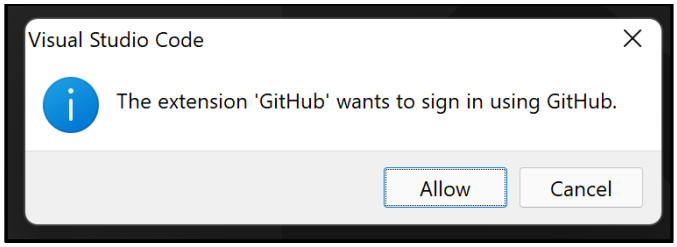


圖 9-5　官方畫面示意：登入流程可能要求在瀏覽器完成授權，再回到 VS Code。

官方示範截圖；VS Code 的 GitHub 登入授權提示

若按登入後沒有返回編輯器，檢查瀏覽器是否仍等待確認。不要把登入密碼或權杖貼進 .R 檔。Private 也需要管理分享權限；若教師要收作業，依課程指定方式提供存取。下載 ZIP 只能取得檔案快照，與保留 Git 歷史的 Clone 不完全相同。

延伸查閱：Repositories and remotes　|　Working with GitHub in VS Code

第 9 章　Git 與 GitHub

同步之前先辨認方向

學習目的：選擇 Fetch、Pull 或 Push，遇到衝突時先保留工作並理解原因。

第一次發布後，之後的 Commit 仍先留在本機。Push 才上傳本機提交；Pull 先取得遠端內容，再整合到目前分支；Fetch 只更新遠端資訊，適合先查看同學做了什麼。Sync Changes 會先 Pull 再 Push，不能把它當成單純的儲存按鈕。

1. 作業開始前，在 Source Control 的 … 選單選 Fetch，查看 Graph 或狀態列的遠端資訊。
2. 若要納入遠端更新，先提交自己的修改或妥善保存，再選 Pull；完成後重新執行分析。
3. 完成一段工作後檢查差異、Stage 與 Commit，再由 … → Push 上傳；最後到 GitHub 確認最新提交。
4. 若發生衝突，先讀出衝突檔案與雙方內容，決定要保留或合併哪些分析步驟，處理後執行程式再提交。

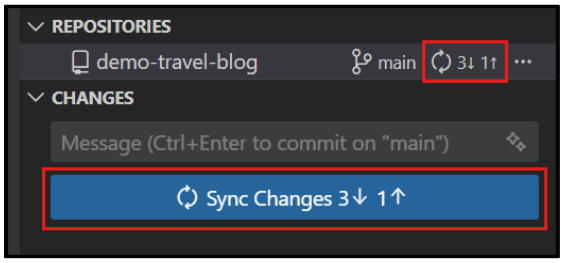


圖 9-6　官方畫面示意：向下箭頭表示待納入的提交，向上箭頭表示待推送的提交；Sync 會做雙向操作。

官方示範截圖；Source Control 的 Sync Changes 按鈕

箭頭數量是相對於上次取得的遠端狀態，想知道最新狀況應先 Fetch。Push 被拒絕時，常見原因是遠端已有新的提交或帳號沒有寫入權限。先看錯誤訊息，再決定拉取或檢查帳號；不要用 Force Push 當作通用修復方式。若不確定如何合併，保留現有檔案並請教師協助判讀。

練習與自我檢核　口頭演練：同學改了圖標題，你改了資料篩選條件，兩人都提交後，為何不能只按儲存就認為對方看得到？

延伸查閱：Push pull and sync　|　Resolve merge conflicts　|　Source control troubleshooting

第 10 章　在 VS Code 使用 AI 助手

分清 AI 工具的三種入口

學習目的：辨認網頁聊天、VS Code 擴充套件與 CLI，避免把安裝成功誤認為已取得服務權限。

入口	操作位置	適合本課的用途
網頁聊天	ChatGPT、Claude、Gemini 網站	產生範例後複製到 .R 檔。
擴充套件	VS Code 的側邊欄或編輯器	引用檔案、解釋程式、檢閱修改。
CLI	VS Code 的 Terminal 內	在專案目錄以文字與 Claude Code 協作。

本章安裝 Codex、Claude Code 並認識 Copilot；練習時一次選一個工具即可。擴充套件是服務的操作入口，訂閱、登入、組織規則與額度另行決定可用功能。ChatGPT 訂閱、Claude 訂閱與 GitHub Copilot 權益不互通。R 程式仍由本機 R 執行，AI 助手不會取代 R 主程式。

1. 按 Ctrl+Shift+X 開啟 Extensions。安裝前核對名稱與發行者，優先由官方文件的安裝連結進入。
2. Codex 搜尋「Codex」，核對 OpenAI 與識別碼 openai.chatgpt；Claude Code 核對 Anthropic 與 anthropic.claude-code。按 Install，必要時重新載入視窗。
3. 依工具頁面完成登入。若沒有合適帳號，先完成第八章網頁練習，將本章視為教師示範。

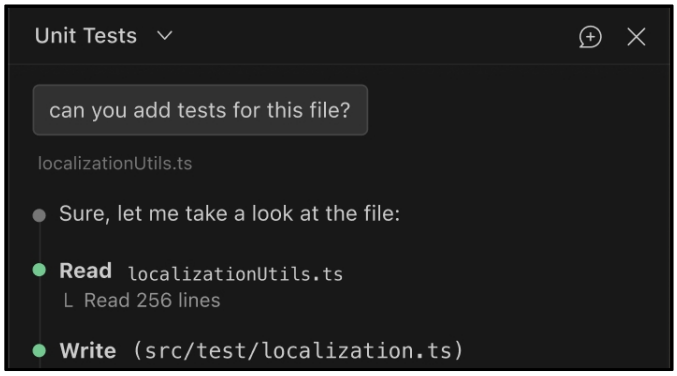


圖 10-1　Claude Code 官方 VS Code 對話區局部；示範提示、讀取檔案及工具操作的位置。

官方示範截圖；Claude Code 擴充套件對話面板

安裝後找不到面板時，先確認 extension 已啟用，再由命令選擇區搜尋工具名稱。不要因為找不到按鈕就安裝名稱相近、來源不明的替代品。

延伸查閱：Codex IDE extension　|　Use Claude Code in VS Code　|　Set up GitHub Copilot in VS Code

使用 Codex 解釋並修改一個 R 檔

學習目的：提供明確檔案脈絡，讓 AI 完成範圍小而可檢查的修改。

1. 先開啟 1151-vscode-lab，儲存 R/03_histogram.R。若已完成第九章，先建立 Git commit 作為修改前的紀錄。
2. 安裝 OpenAI 的 Codex 擴充套件後，按 Codex 圖示；找不到時按 Ctrl+Shift+P，執行 Codex: Open Codex Sidebar。依畫面登入。
3. 開啟 R/03_histogram.R，選取繪圖段落，送出下列第一個要求。讀懂解釋後，再送出第二個小修改。

第一步：請用繁體中文解釋目前選取的繪圖程式。
指出資料來源、橫軸、縱軸與分組方式。先不要修改檔案。

第二步：只修改 `R/03_histogram.R` 的圖形標題與顏色。
標題改為 `Fuel economy of 32 cars`，顏色改為 `lightblue`。
保留資料與統計計算，完成後列出改動及驗證方式。

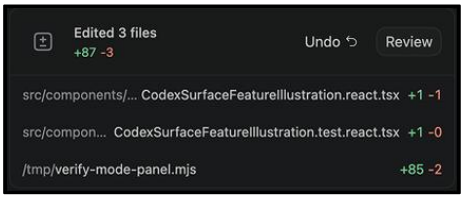


圖 10-2　Codex 官方修改檢閱區局部；檔案清單、增刪計數，以及 Undo / Review 按鈕可用於核對本次變更。

官方示範截圖；Codex 在編輯器內檢視修改差異

預期只改動指定 R 檔的繪圖參數。開啟修改差異，確認沒有刪除資料處理或改動其他檔案；回到 R 終端機重新執行，筆數與平均數應保持原值。若 Codex 沒有引用到檔案，重新附加目前檔案或明確寫出相對路徑。圖中的程式為官方示例，不是本課 R 回應。

延伸查閱：Codex IDE extension

使用 Claude Code 引用選取程式

學習目的：把問題綁定到檔案與行數，先獲得說明再進行小幅修改。

1. 在 Extensions 安裝 Anthropic 的 Claude Code。開啟 R/04_regression.R，再由命令選擇區搜尋 Claude Code，選擇開啟面板或新分頁。
2. 依畫面登入合適的 Claude 帳號。免費 Claude 網頁對話與 Claude Code 的使用權益不同，實際可用性以登入畫面為準。
3. 選取散佈圖的程式；在 Claude Code 輸入框以 @ 選取檔案，或用 Alt+K 加入選取範圍。先要求解釋，確認它引用的檔名與行數。
4. 準備修改時選擇 Manual / Ask before editing 等逐次檢閱模式，閱讀提案後才接受本次小修改。

```
請解釋這段 R 程式為何以 wt 為橫軸、mpg 為縱軸。
核對 mtcars 的原始單位。先不要改檔，也不要新增套件。
接著只提出修改座標軸標籤的建議。
```

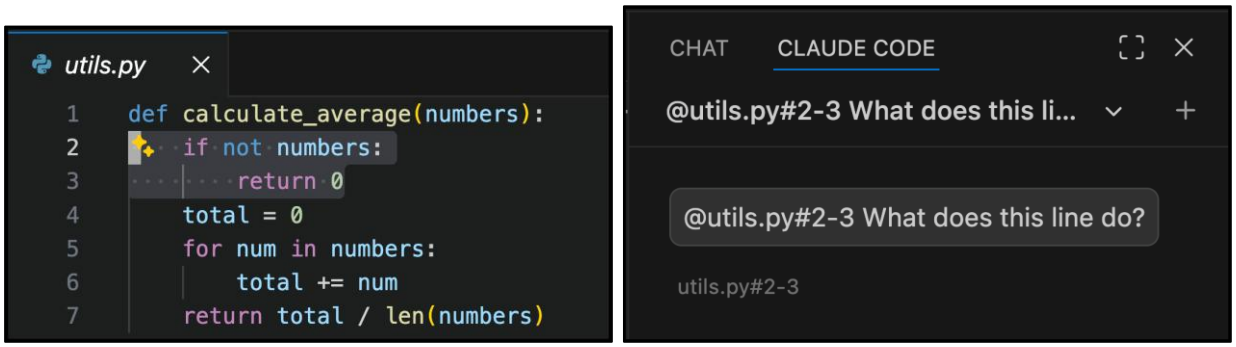


圖 10-3　Claude Code 官方畫面局部：左側為選取的程式，右側以 @ 檔名與行數引用該範圍；本課操作時改為選取 R 程式。

官方示範截圖；Claude Code 引用選取程式碼

預期回覆能對應到你選取的繪圖指令。若回答的是另一份檔案，先修正引用，再繼續提問。版本較新時模式名稱與按鈕位置可能改變，應查看輸入框附近的模式標示。本章面板能使用，不代表 PowerShell 已能執行 claude；獨立 CLI 安裝見第十一章。

延伸查閱：Use Claude Code in VS Code　|　R 官方 mtcars 資料說明

第 10 章　在 VS Code 使用 AI 助手

使用 GitHub Copilot 提示與對話

學習目的：分辨一般程式提示與 AI 建議，並練習接受一個可理解的補全。

1. 在 VS Code 狀態列找到 Copilot 圖示，選擇 Use AI Features 或 Sign in to use Copilot，依畫面登入 GitHub 帳號。
2. 開啟一個 R 練習檔，寫下描述任務的註解，稍候觀察是否出現淡色建議。先閱讀整段；要採用才按 Tab，不採用可按 Esc。
3. 開啟 Chat，附加目前 R 檔，詢問「請解釋這段程式的資料來源與輸出，先不要改檔」。對照本機程式核對回答。

```
# 使用 mtcars 的 mpg 畫直方圖，只用 base R
# 橫軸標示 Miles per US gallon，縱軸標示 Frequency
```



圖 10-4　Copilot 官方登入入口局部；由狀態列 Signed out 開啟選單後，選取 Sign in to use Copilot。

官方示範截圖；GitHub Copilot 登入入口

IntelliSense 通常列出函數、參數或物件名稱；Copilot 的淡色建議可能一次補出多行程式。兩者都只是編輯協助，是否合理要靠你檢查。AI 建議若引用不存在的變數，先修正名稱；若沒有出現建議，檢查登入、可用額度、功能是否啟用與目前語言，不必反覆按 Tab。

符合資格的帳號可使用 Copilot Free 的有限額度，組織帳號也可能受管理規則影響。畫面出現可用模型不代表其他服務的訂閱已互通；本課不要求購買特定方案。

練習與自我檢核　把一段可理解的補全存檔、執行，並在 AI 紀錄中寫明你接受了哪幾行、檢查了哪些變數。

延伸查閱：Set up GitHub Copilot in VS Code　|　Copilot inline suggestions

檢閱 AI 修改並保留可恢復版本

學習目的：把 AI 的建議轉成可追溯的程式修改，完成差異檢查與重新執行。

AI 改完檔案後，先看差異再執行。紅色通常表示刪除，綠色通常表示新增；顏色只說明變化，不保證正確。工具內的 Keep、Accept 或同意修改，也不等同 Git commit。若修改已寫入磁碟，尚未按 Keep 仍可能影響執行結果。

1. 閱讀受影響檔案清單，確認範圍與提示語一致；開啟每個檔案的差異。
2. 逐段核對資料欄位、篩選條件、圖形參數與輸出路徑。若改動超出要求，提出具體修正或拒絕該改動。
3. 儲存後從頭執行 R 程式。核對 32 筆資料、mpg 平均數約 20.09，以及圖形單位。
4. 將完成的修改與說明記錄在 notes/ai_log.txt；確認結果後再依第九章流程建立 Git commit。

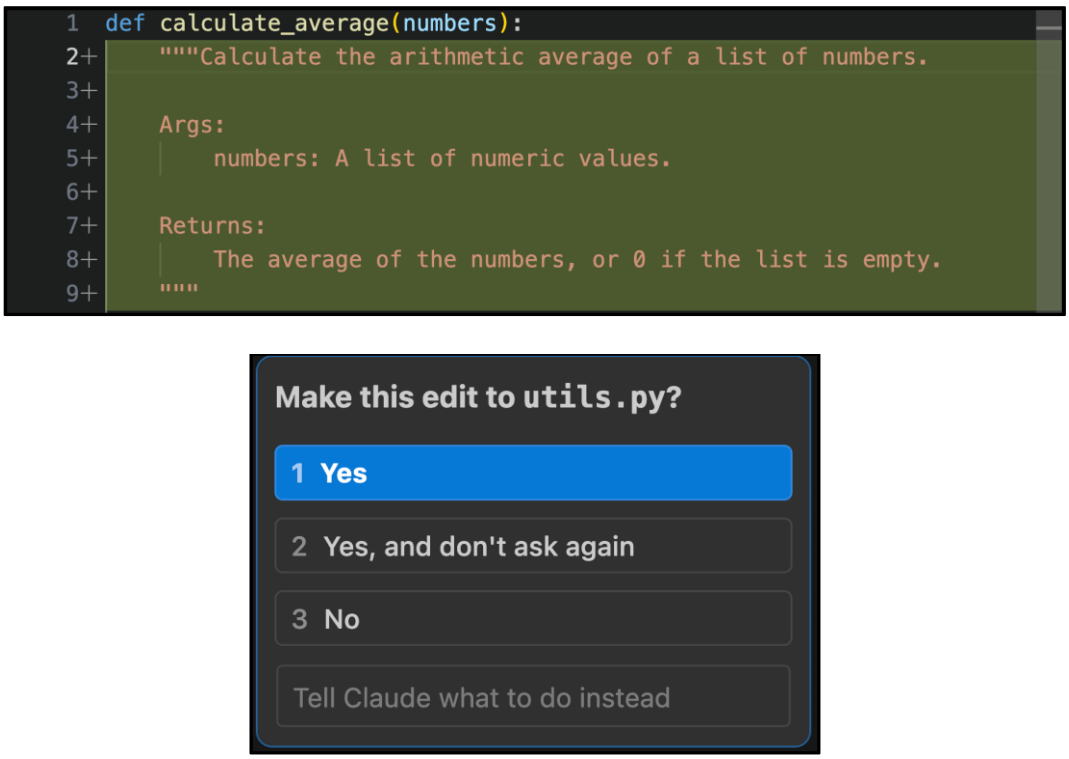


圖 10-5　Claude Code 官方畫面局部：上方綠色區域為新增內容，下方為逐次確認對話框；先閱讀修改，再決定是否採用。

官方示範截圖；Claude Code 檢閱建議修改

需要還原時，先確認要還原的範圍。小段文字可用 Undo；有 Git 紀錄的檔案可從 Source Control 比較並恢復指定變更。不要為了撤銷 AI 的一行修改而丟棄同學或自己其他尚未保存的工作。各工具的檢查點功能不同，不能把它當成所有外部動作的復原鍵。

練習與自我檢核　請 AI 只改直方圖的顏色，觀察差異後再還原；重新執行並確認圖回到原樣。說明「還原程式」與「刪除對話」為何是兩件事。

延伸查閱：Review and revert agent changes　|　Use Claude Code in VS Code

第 11 章　Terminal 與 Claude Code

開啟 Terminal 並辨認輸入環境

學習目的：分清 PowerShell、R 與 Claude Code 的提示位置，避免把不同語言混在一起。

1. 選單選擇 Terminal > New Terminal，或按 Ctrl+Shift+反引號鍵新增終端機。Ctrl+反引號鍵則用來顯示或隱藏終端機面板，不一定會新增工作階段。反引號鍵通常位於 Esc 下方。
2. 按新增終端機「 + 」旁的下拉箭頭，選 PowerShell。若預設環境不同，可由 Select Default Profile 選擇後另開新終端機。
3. 先查看頁籤名稱與提示字元，再決定要輸入哪一類內容。R 的 > 與 PowerShell 路徑末端的 > 不是同一種環境。

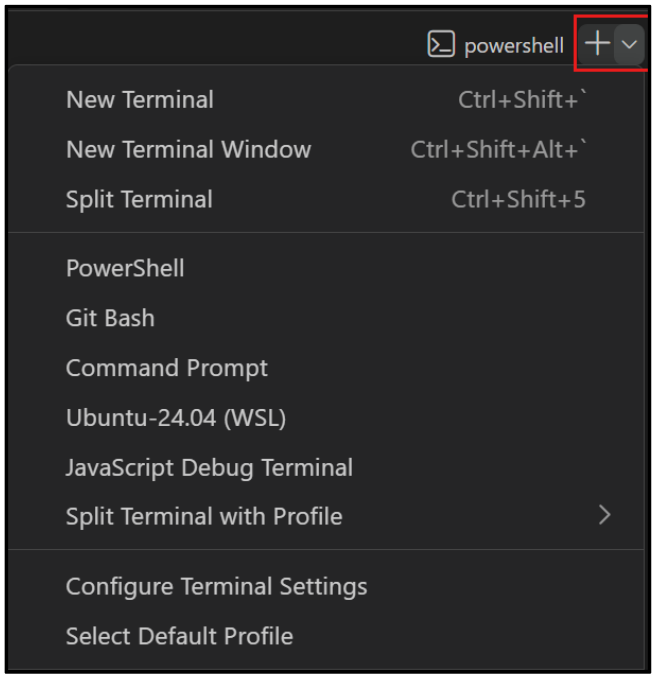


圖 11-1　VS Code 官方終端機設定檔選單局部；由新增按鈕旁的箭頭選擇 PowerShell，或設定 Select Default Profile。

官方示範截圖；選擇 PowerShell 等終端機設定檔

所在環境	適合輸入	本課例子
PowerShell	檔案指令、啟動程式	Get-Location、claude --version
R 終端機	R 語法	mean(mtcars$mpg)
Claude Code 對話	自然語言、工具命令	請解釋 R/03_histogram.R、/help

預期看到可以輸入指令的 PowerShell。若把 mean(...) 打進 PowerShell，或把 Get-Location 送進 R，錯誤來自輸入環境，不一定是程式本身有問題。先切到正確頁籤再重做。若快捷鍵與輸入法衝突，使用選單即可；Mac 常用 Control+反引號鍵開關終端機，shell 通常是 zsh。

延伸查閱：Terminal Basics　|　Getting started with the terminal

第 11 章　Terminal 與 Claude Code

確認目前資料夾與相對路徑

學習目的：知道指令在哪個資料夾執行，避免輸出寫到不預期的位置。

1. 在 Explorer 對 1151-vscode-lab 根資料夾按右鍵，選 Open in Integrated Terminal。不要先對 R 子資料夾操作。
2. 在 PowerShell 依序輸入下列指令，每行按 Enter。先讀位置，再讀目錄內容；確認可見 R、python、data、outputs 與 notes。
3. 若開在其他位置，使用 Set-Location 指定自己的完整專案路徑。路徑有空白時加引號；範例路徑須依自己的電腦修改。

```
Get-Location
Get-ChildItem
Test-Path .\R\03_histogram.R
# 需要切換時才執行，先改成自己的路徑
Set-Location "D:\course\1151-vscode-lab"
```

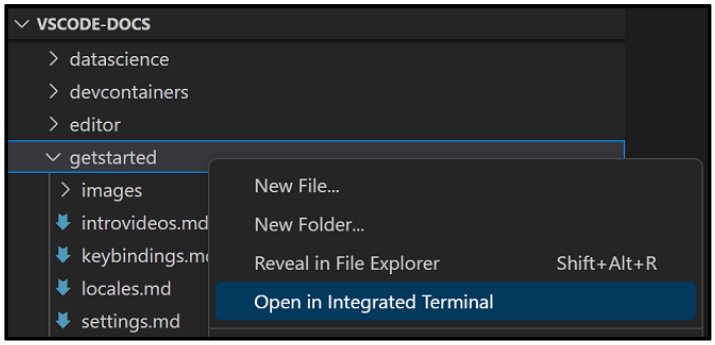


圖 11-2　VS Code 官方 Explorer 右鍵選單局部；Open in Integrated Terminal 可在所選資料夾開啟終端機。

官方示範截圖；在指定資料夾開啟整合式終端機

預期 Test-Path 回傳 True。若為 False，檢查資料夾層級與檔名，不要立即新建重複檔案。絕對路徑從磁碟代號開始；相對路徑 .\R\03_histogram.R 則從目前工作目錄出發。開啟編輯器中的 R 檔，不一定會自動把既有終端機切到同一資料夾。

不同 shell 使用的指令與引號規則可能不同，本頁以 PowerShell 為準。不要把網路上的 Bash 指令直接當成 PowerShell 指令；先確認教學所使用的環境。

練習與自我檢核　先記錄 Get-Location 的結果，再切進 R 子資料夾並列出檔案，最後用 Set-Location .. 返回根資料夾。

延伸查閱：Terminal Basics　|　PowerShell Get-Location

安裝並檢查 Claude Code CLI

學習目的：讓 PowerShell 能找到獨立的 claude 指令，了解它與擴充套件的關係。

本頁是需要合適帳號的進階練習。Claude Code 擴充套件自帶供面板使用的程式，但 Terminal 內的 claude 指令需另裝 CLI。Windows 原生環境可以使用 PowerShell；本課不要求先安裝 WSL。已安裝 Git for Windows 可提供 Git Bash，但並非此流程的必備條件。

1. 先在 PowerShell 執行 claude --version。若已有版本資訊，可跳過安裝。若顯示找不到指令，先核對是否曾安裝，再依下一步處理。
2. 可用 WinGet 時，執行下列安裝指令並閱讀畫面的套件資訊。若找不到 winget，改依官方安裝頁的 Windows PowerShell 原生安裝方法，不混用 CMD 指令。
3. 完成後關閉舊終端機、另開 PowerShell，再檢查版本。需要時重開 VS Code，讓它取得更新後的程式搜尋路徑。

```
winget install Anthropic.ClaudeCode
# 安裝完成後，在新開的 PowerShell 檢查
claude --version
claude doctor
```

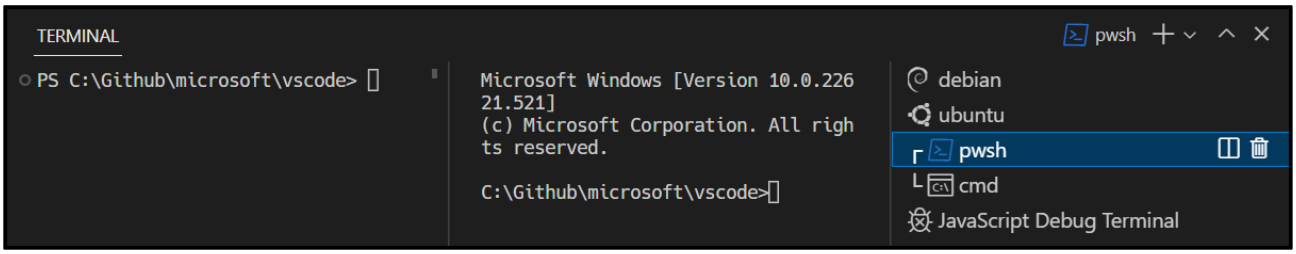


圖 11-3　VS Code 官方多終端機與分割畫面局部；操作時可保留 R 頁籤，另開 PowerShell 檢查 CLI。

官方示範截圖；多個終端機與分割畫面

預期 claude --version 印出版本號；這只證明指令可啟動，尚不代表帳號已登入或有可用額度。若仍找不到，查官方安裝頁的 PATH 說明。若安裝成功但登入受限，先確認帳號方案與組織設定，不要重裝 R 或改動課程程式來嘗試解決。

延伸查閱：Claude Code 安裝設定　|　Use Claude Code in VS Code　|　Claude Code CLI reference

在專案中啟動 Claude 並提出任務

學習目的：先確認工作目錄，再讓 Claude Code 讀取指定 R 檔並提出可檢查的修改計畫。

1. 在 VS Code 的 PowerShell 確認目前位於 1151-vscode-lab，先儲存所有編輯中的檔案。依第九章保留 Git 紀錄。
2. 執行下列指令，以 Plan 模式開始。第一次使用時依畫面完成登入與專案信任確認；登入由學生本人操作。
3. 看到 Claude 的文字輸入區後，送出下列任務。先讓它辨認的檔案與計畫；這一步不需要它自動安裝套件。

```
Get-Location
claude --permission-mode plan
```

```
請閱讀 R/03_histogram.R，只規劃以下修改：
把 mpg 直方圖另存成 outputs/mtcars_hist_ai.png。
使用 base R，保留原有計算，不改動其他資料。
請說明要加入哪些指令，以及如何檢查輸出成功。
先不要修改或執行程式，等我確認。
```

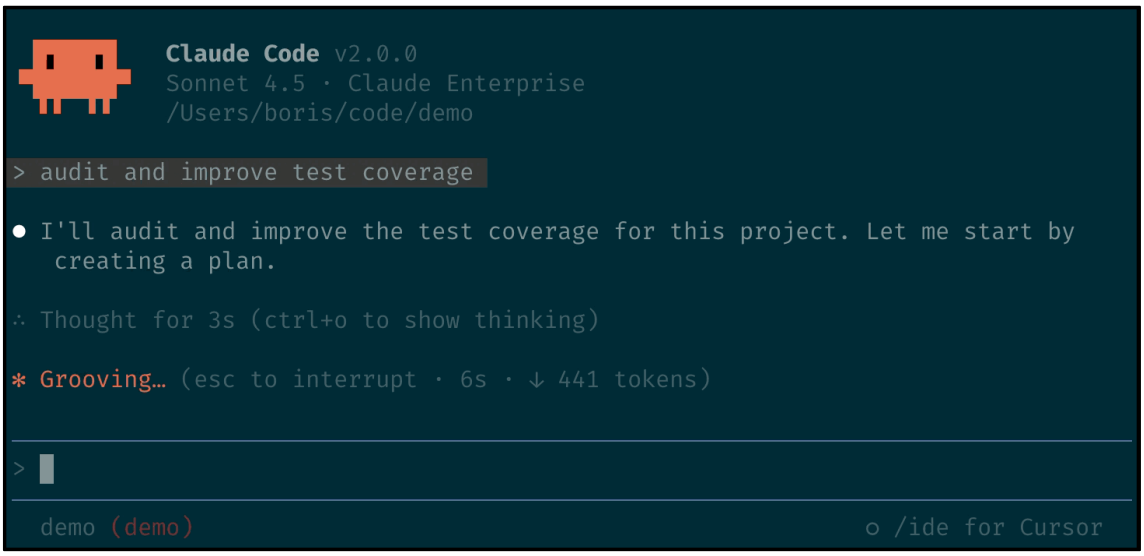


圖 11-4　Claude Code 官方終端機示範的文字對話區局部（macOS、舊版介面；用於辨識對話與輸入區，不代表本課實測回應）。

官方示範截圖；Claude Code 的終端機文字對話

預期得到具體的檔案與修改步驟。若 AI 回答找不到檔案，先回頭確認目錄及檔名；若要求操作其他資料夾，先釐清理由。圖為官方舊版 macOS 終端機示例，用於辨識對話結構；本課在 VS Code 的 PowerShell 內操作，版本與回覆不必完全相同。

延伸查閱：Claude Code CLI reference　|　Claude Code Quickstart　|　Claude Code 官方示範

第 11 章　Terminal 與 Claude Code

執行驗證與結束 Claude 工作階段

學習目的：完成一次計畫、檢閱、執行、保存的協作循環，並分清中止與關閉。

1. 讀完計畫後，如同意進行本次修改，可依模式選單或按 Shift+Tab 切到 Manual，確認模式標示後，明確要求只修改指定 R 檔。逐次閱讀提案；不知道的指令先請它解釋。
2. 回到編輯器查看差異，再在 R 終端機從頭執行。確認 outputs/mtcars_hist_ai.png 可開啟，圖形仍使用共用 CSV 的 mpg 欄位，即 cars$mpg。
3. 檢查 R 的圖形輸出是否成對呼叫 png() 與 dev.off()。若檔案不存在，確認工作目錄與 outputs 資料夾；若檔案空白，檢查繪圖指令是否位於兩者之間。
4. 把結果與你的驗證寫入 notes/ai_log.txt，再建立 Git commit。要結束時在 Claude 輸入 exit，回到 PowerShell。

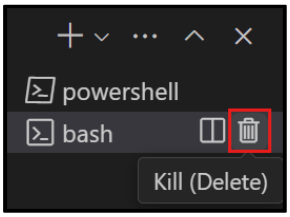


圖 11-5　VS Code 官方終端機關閉控制局部：垃圾桶會結束該終端機；右上角 X 通常只是隱藏面板。

官方示範截圖；終端機的關閉控制

情況	處理方式
Claude 正在做不需要的工作	按 Esc 中止，讀取已完成的修改後重新指示。
想查看 Claude 指令	在 Claude 對話輸入 /help。
要回到 PowerShell	在 Claude 輸入 exit，確認回到 shell 提示位置。
執行後圖檔仍不存在	核對工作目錄、輸出路徑與 dev.off()。

中止回覆不會自動還原已完成的檔案修改，關閉面板也不等於結束執行中的程式。每次協作最後都應查看變更、重跑程式及保存紀錄；這是讓 AI 協助可用於課堂作業的必要步驟。

延伸查閱：Claude Code interactive mode | Claude Code terminal guide | Getting started with the terminal

第 12 章　綜合實作與繳交

建立自己的統計小專案

學習目的：把環境設定、檔案管理、R 與 Python 分析串成可以獨立完成的工作流程。

本實作以「車重與燃油使用表現」為主題，建議安排 60–90 分鐘。先獨立完成必要部分，再使用 AI 協助改進。學習成果包括可重跑的程式、兩張直方圖、一份驗證表，以及能說明人工判斷的協作紀錄。

1. 複製課堂範例為自己的作業資料夾，以 File > Open Folder 開啟。保留 R、python、data/raw、outputs、notes 等目錄。
2. 選定 R 與 Python 環境，記錄版本。讀取共用 mtcars.csv，核對資料列數、欄位與單位。
3. 先用 R 完成基礎運算、平均數、中位數與直方圖；再用 Python 完成同一計算。
4. 把圖表標題和配色改成自己的版本，保持組界一致。所有程式都應從檔案讀取資料，並輸出到 outputs。

```
VS Code 入門手冊配套範例

以 VS Code 的 File > Open Folder 開啟整個 1151-vscode-lab 資料夾。
R 範例以本資料夾為工作目錄；先執行 R/02_export_mtcars.R 可重建 CSV。
R: 依序執行 R/01_basics.R、R/03_histogram.R、R/04_regression.R。
Python: 建立並選取專案 .venv，再使用 python -m pip install -r requirements.txt。
Python: 依序執行 python/01_basics.py、python/02_histogram.py、python/03_regression.py。
debug 範例刻意保留錯誤呼叫的註解，請依手冊指示操作。

共用資料 data/raw/mtcars.csv 來自 R datasets::mtcars。
32 輛汽車；mpg 為每美制加侖行駛英里數，wt 為千磅，cyl 為汽缸數。
原資料說明：https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html
程式產生圖片及表格至 outputs/；不必將 .venv 或 output 提交 Git。
AI 協作紀錄放在 notes/ai_log.txt，保留提示、人工修改及驗證紀錄。
```

圖 12-1　VS Code 中的 README 局部，列出開啟專案、執行順序、資料來源與輸出位置。

實機截圖：本課範例；畫面可依正文命令操作。

進階部分執行 mpg ~ wt 的簡單線性迴歸，寫一段 100–150 字的結果解釋，必須包含單位、係數意義與不能直接推論因果的限制。若使用 AI，先請它檢查你的步驟或解釋錯誤，再決定是否採用修改。不要只交付 AI 產生的圖片。

練習與自我檢核　完成前請同學只看你的 README.md 和程式，嘗試重新產生一張圖；記錄他在哪一個步驟需要詢問你。

延伸查閱：R datasets mtcars

第 12 章　綜合實作與繳交

驗證並記錄 AI 協作

學習目的：把輸出檢查與 AI 使用紀錄整合到作業中。

AI 可以協助修正程式，最後仍需由你確認資料處理與統計解釋。使用第 8 至第 11 章任一工具，挑選一個具體任務，例如「改善圖形標籤」「解釋型態錯誤」或「檢查是否漏掉資料」。保留原始程式與修改後版本，方便比較。

驗證層次	作業中要留下的證據
資料	32 列、mpg 為數值、無缺失值；註明單位
計算	平均 20.090625、中位數 19.2、五組次數一致
程式	新工作階段從頭執行成功，輸出路徑可用
圖形	標題、單位、次數軸與共同組界皆正確
統計	若做迴歸，解釋斜率並區分關聯與因果
AI 協作	提示語、採用內容、人工修改與驗證方法

將紀錄寫入 notes/ai_log.txt。可以採用以下格式：「我先寫了什麼；遇到什麼問題；提供給 AI 的內容；AI 建議了什麼；我保留或刪除了什麼；用哪個數值或文件驗證」。工具名稱與使用日期有助於理解環境，無需抄錄與本次作業無關的長對話。

任務：檢查 `R` 與 `Python` 直方圖為何不一樣。
我提供：兩段程式、相同 `CSV` 的欄位摘要、組界。
我要求：先指出分組差異，不要直接重寫整個專案。
我驗證：修正後的次數皆為 `5`、`13`、`8`、`2`、`4`。
我說明：顏色不同不影響次數，組界不同則會影響。

若 AI 給出的數字與基準不同，先檢查是否讀錯資料、刪除資料列、用不同組界或做了單位轉換，不要為了符合答案而手動改輸出。保存差異與排查過程，比只留下成功截圖更能呈現學習。

練習與自我檢核　請同學選一行你採用的 AI 程式，口頭詢問其作用；若不能解釋，回到文件查證後再繳交。

延伸查閱：R graphics hist | NumPy histogram

整理繳交內容與自我檢核

學習目的：交付一份清楚、可重跑且能說明來源的分析專案。

交作業前先以新的 R 工作階段與 Python 程序各執行一次，確認結果不依賴記憶體中的舊變數。將圖檔和簡短說明放進作業壓縮檔；Git 儲存庫則依課程要求管理來源碼與筆記。配套 .gitignore 忽略 outputs，故不要以為圖檔會自動進入 GitHub。

項目	建議交付內容
程式與資料	R/、python/、data/raw/mtcars.csv
分析成果	outputs/ 中兩張直方圖、必要的迴歸圖與計數表
說明與紀錄	README.md、notes/ai_log.txt、環境版本與結果解釋
相依套件	requirements.txt，必要時附實測版本紀錄
不必打包	.venv、暫存檔、密碼、API 金鑰與個人設定

1. README.md 寫明資料來源、各檔用途、執行順序，以及輸出會出現在哪裡。

2. 檢查所有相對路徑，避免只在自己的電腦有效的絕對路徑。

3. 若使用 Git，檢查變更清單與差異，完成有意義的 commit；依教師指定方式繳交儲存庫網址或壓縮檔。

4. 重新打開壓縮檔或下載副本，確認程式、CSV、圖形與筆記完整，再核對檔名。

自我評量可依五個面向各 20 分檢核：環境與結構、R 實作、Python 實作、驗證與統計解釋、AI 紀錄與可重現性。能執行只是第一步；能說清楚資料、參數與結果，並讓別人重跑，才是本實作的完成標準。實際成績與繳交格式仍依本課程公告。

練習與自我檢核　請用三句話回答：我的分析問題是什麼？我用哪些檢查確定結果合理？若同學重跑失敗，第一個應檢查什麼？

延伸查閱：Getting Started with Python in VS Code

操作速查與學習資源

Windows 常用快捷鍵

學習目的：以少量常用快捷鍵建立穩定操作習慣，遇到差異時能查詢命令名稱。

快捷鍵會受作業系統、鍵盤配置及使用者設定影響。本表列 Windows 常見預設值；若沒有反應，先用命令選擇區搜尋相同功能，或開啟 Keyboard Shortcuts 查詢。macOS 通常改用 Command，但不是所有快捷鍵都能直接替換。

工作	Windows 常見快捷鍵	辨認方式
儲存目前檔案	Ctrl+S	未儲存標記消失。
搜尋檔案	Ctrl+P	依檔名選取。
命令選擇區	Ctrl+Shift+P	搜尋編輯器命令。
開啟設定	Ctrl+,	檢查 User / Workspace。
目前檔案搜尋	Ctrl+F	只搜尋目前檔。
目前檔案取代	Ctrl+H	先確認取代內容。
工作區搜尋	Ctrl+Shift+F	列出多個檔案。
Explorer	Ctrl+Shift+E	回到資料夾樹。
Extensions	Ctrl+Shift+X	搜尋或管理套件。
Source Control	Ctrl+Shift+G	檢查 Git 修改。
切換下方面板	Ctrl+J	隱藏不代表停止程式。
切換終端	Ctrl+`	`` ` `` 是反引號鍵。
分割編輯區	Ctrl+\	左右比較檔案。
行註解	Ctrl+/	依語言切換註解。
下一個相同選取	Ctrl+D	逐次加入選取範圍。
目前檔案符號	Ctrl+Shift+O	導覽函式等符號。

Ctrl+K Ctrl+S 表示先按 Ctrl+K，放開後再按 Ctrl+S，是兩段式快捷鍵。它會開啟 Keyboard Shortcuts。不要一次把所有鍵壓住，也不要與單獨 Ctrl+S 儲存混淆。

在 R 或 Python 中執行程式的快捷鍵可能由擴充套件定義，請依對應語言章節操作，並查看命令名稱。終端取得焦點時，有些按鍵會交給正在執行的程式處理；若想編輯檔案，先點回編輯區。

練習與自我檢核　先熟悉儲存、命令選擇區、檔案搜尋與工作區搜尋四項，再逐步增加。任選一項在鍵盤設定中查出其命令名稱。

延伸查閱：Keyboard shortcuts　|　Windows shortcut reference

操作速查與學習資源

從訊息定位常見問題

學習目的：依現象逐層排查環境、路徑及程式，不以重裝所有軟體作為第一步。

排錯先分層：看不到檔案通常是資料夾或路徑問題；找不到命令通常是安裝或環境問題；執行後報錯可能是語法、資料或套件問題；結果不合理則要回到統計方法與輸入資料。先抄下完整訊息，再找對應層次，才能避免反覆改動無關設定。

現象	先檢查什麼	下一個可驗證動作
Explorer 沒有作業	是否開對根資料夾	File → Open Folder 重新選取。
檔案沒有語法顏色	副檔名與語言模式	確認不是 .R.txt / .py.txt。
找不到 R / Python	主程式與套件是否都已安裝	查看語言章的路徑或解譯器設定。
找不到資料檔	目前工作目錄、拼字與檔案存在性	用 getwd() 或 Path.cwd() 查看。
找不到套件	執行環境是否與安裝環境一致	先確認正在用哪個 R / Python。
提示或 Outline 空白	語言服務、套件啟用與檔案內容	開正確檔案，讀 Problems / Output。
Git 無法提交	作者設定與暫存區	設定姓名信箱，確認 Staged Changes。
GitHub Push 失敗	帳號權限或遠端有新提交	讀完整訊息，先 Fetch 檢查。
AI 程式能跑但結果怪	資料、參數與統計解釋	用小例子手算或對照已知結果。

請教師或 AI 協助時，提供四項資訊：要完成的任務、最小程式、完整錯誤文字、預期與實際結果。再附上作業系統、VS Code 與語言版本。若要附圖，讓錯誤訊息和相關程式行同時可讀；單拍一個紅色圖示通常不足以判斷原因。

修正後從乾淨工作階段重新執行，確認問題確實消失，再繼續原本的分析。一次只改一個主要因素，保留原始錯誤與修正說明，下一次遇到類似情況就能快速定位。不要公開含個人資料、金鑰或權杖的畫面。

若同一段程式在教師電腦成功、自己的電腦失敗，先比較語言版本與套件版本，再檢查路徑和環境。不要假設兩個名為 Python 的終端就使用相同解譯器；也不要把另一個環境安裝套件的成功訊息當成目前環境可用的證據。

練習與自我檢核　挑一個本週遇過的錯誤，用「問題、原因、修正、驗證」四句話記錄。確認另一位同學能依紀錄重現並修好它。

延伸查閱：VS Code troubleshooting | Python environments | R in VS Code

學習資源與引用說明

本教材銜接原有 R 與 AI 講義，將編輯器教學轉化為統計程式操作。影片列出適合搭配的主題，技術步驟對照官方文件；Springer 章節列為延伸閱讀。所有連結的查閱日期為 2026-11-12。

1. Getting started with Visual Studio Code

Microsoft / Visual Studio Code；用於介面入口、開啟資料夾、儲存與常用功能的概念對照；教材的 R 與 Python 操作為本課改編。

https://code.visualstudio.com/docs/introvideos/basics

2. VS Code for Absolute Beginners | 2026

Coder Coder；2025-12-22；48:27；以新手逐步完成檔案、設定、搜尋與整合工具的順序作為參考；網頁範例改為統計分析。

https://www.youtube.com/watch?v=lWEKiak0WVU

3. Visual Studio Code Tutorial for Beginners - Introduction

Academind；2018-05-24；33:48；參考編輯器定位、Activity Bar、資料夾、終端與快捷鍵的解說；以目前版本重新核對流程。

https://www.youtube.com/watch?v=VqCgcpAypFQ

4. Learn Visual Studio Code in 15 minutes: 2026 Official Beginner Tutorial

Visual Studio Code；2026-02-04；15:28；參考介面導覽、新增檔案、Chat 與 Source Control 的概念；MCP 等內容列為延伸。

https://www.youtube.com/watch?v=f8_uF_IDV50

5. How to Use VS Code: The Ultimate Beginner's Guide (2026)

JavaScript King；2026-07-14；22:58；參考安裝到日常工作流程的銜接，改編為資料與統計程式專案。

https://www.youtube.com/watch?v=6tQ6MS8cHMk

6. Tutorial: Get started with Visual Studio Code

Microsoft / Visual Studio Code；作為開資料夾、新增檔案、設定、編輯與工具串接的步驟核對來源；本課使用自己的程式與資料。

https://code.visualstudio.com/docs/editing/getting-started/editor-tutorial

7. Introduction to Visual Studio Code

Sufyan bin Uzayr；2021-09-21；以公開摘要支持編輯器功能的背景，全文列為延伸閱讀。

https://link.springer.com/chapter/10.1007/978-1-4842-7344-9_1

8. Visual Studio Code: A Guide to Microsoft’ s Free Code Editor

Coursera Staff；2026-07-01；參考編輯器使用情境與功能分類；技術步驟另以官方文件核對。

https://www.coursera.org/articles/visual-studio-code

9. Get to know Visual Studio Code

Microsoft / Visual Studio Code；用於解釋專案從資料夾開始、編輯器與語言執行環境的分工，以及 AI 的選擇性使用。

https://code.visualstudio.com/docs/getstarted/what-is-vscode

10. 為什麼全球開發者都離不開 VS Code 這個軟體？新手必學的完整入門指南，一次搞懂 7 大核心功能！

PAPAYA 電腦教室；2026-07-22；23:32；作為教師指定七大核心功能的覆蓋檢查表，依本課先 R 後進階工具的順序重新編排。

https://www.youtube.com/watch?v=2RbXlT5wl9c

11. AI 编程前必须学会的 VS Code｜零基础工作流入门【Python / Git / Terminal / GitHub】新手教程 2026 VS Code Beginner Guide

乐心 / lexinlabs；2026-05-14；19:53；參考 Python 執行、終端種類、虛擬環境與 Git 的銜接；教學本文使用繁體中文與統計案例。

https://www.youtube.com/watch?v=qbtRYEV7uCs

技術文件與圖片來源

1. R in VS Code　開啟文件
2. Getting Started with Python in VS Code　開啟文件
3. Source control quickstart　開啟文件
4. Installing Visual Studio Code on Windows　開啟文件
5. VS Code user interface　開啟文件
6. VS Code user interface Command Palette　開啟文件
7. Color themes in VS Code　開啟文件
8. User and workspace settings　開啟文件
9. Accessibility: Zoom and High Contrast theme　開啟文件
10. Basic editing Save Auto Save　開啟文件
11. VS Code workspace concepts　開啟文件
12. Workspace Trust　開啟文件
13. R working directory reference　開啟文件
14. Basic editing multi cursor and comments　開啟文件
15. Microsoft naming files paths and namespaces　開啟文件
16. Multi-root workspaces　開啟文件
17. CRAN Windows 下載　開啟文件
18. REditorSupport Windows 安裝說明　開啟文件
19. R datasets mtcars　開啟文件
20. R graphics hist　開啟文件
21. NumPy histogram　開啟文件
22. Python Windows 官方下載　開啟文件
23. Using Python on Windows　開啟文件
24. Python tutorial An Informal Introduction　開啟文件
25. Python csv module　開啟文件
26. Matplotlib hist　開啟文件
27. IntelliSense　開啟文件
28. Code navigation　開啟文件
29. VS Code custom layout split editors　開啟文件
30. Search across files　開啟文件
31. Extension Marketplace　開啟文件
32. R extension　開啟文件
33. Python extension　開啟文件
34. R traceback　開啟文件
35. Python debugging in VS Code　開啟文件
36. R debug and debugonce　開啟文件
37. Use ChatGPT　開啟文件
38. Get started with Claude　開啟文件

39. Gemini Apps 帳號條件　開啟文件
40. Gemini Code Assist 個人帳號停用公告　開啟文件
41. Git downloads　開啟文件
42. Staging and committing changes　開啟文件
43. View source control history　開啟文件
44. gitignore documentation　開啟文件
45. Repositories and remotes　開啟文件
46. Working with GitHub in VS Code　開啟文件
47. Push pull and sync　開啟文件
48. Resolve merge conflicts　開啟文件
49. Source control troubleshooting　開啟文件
50. Codex IDE extension　開啟文件
51. Use Claude Code in VS Code　開啟文件
52. Set up GitHub Copilot in VS Code　開啟文件
53. Copilot inline suggestions　開啟文件
54. Review and revert agent changes　開啟文件
55. Terminal Basics　開啟文件
56. Getting started with the terminal　開啟文件
57. PowerShell Get-Location　開啟文件
58. Claude Code 安裝設定　開啟文件
59. Claude Code CLI reference　開啟文件
60. Claude Code Quickstart　開啟文件
61. Claude Code 官方示範　開啟文件
62. Claude Code interactive mode　開啟文件
63. Claude Code terminal guide　開啟文件
64. Keyboard shortcuts　開啟文件
65. Windows shortcut reference　開啟文件
66. VS Code troubleshooting　開啟文件
67. Python environments　開啟文件
68. 暫存區與程式差異檢視　開啟文件
69. R terminal 啟動畫面　開啟文件
70. R Help viewer　開啟文件

教師原有教材：A01-hmwu_R-Introduction.pptx；115-1_StatSoft_week01_RStudio&AI.docx。保留其學習順序、AI 驗證與使用紀錄觀念，操作介面與安裝方式依本次資料重寫。