和 AI 一起做研究

用 FRED 完成資料、估計、圖表、工作規則與六頁簡報

林世揚

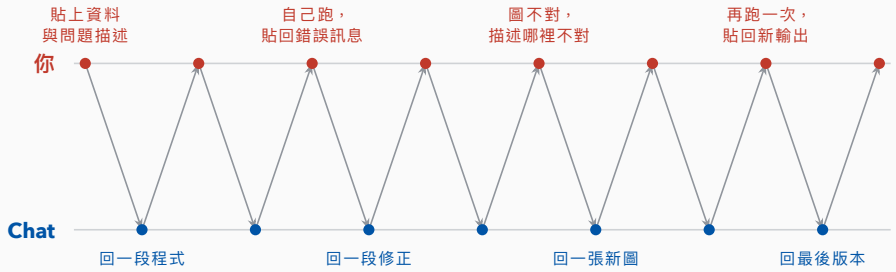
Aug. 19th, 2026

東華大學經濟系

從你昨天的對話說起

在談任何模型之前，先看一次你自己每天在做的事。

你昨天的 ChatGPT 對話，長什麼樣子？



一次很普通的分析：**你動手 7 次**、Chat 回 7 次。**14 次來回，每一次的上下文都是你親手搬過去的。**

而且這 14 次之間你不能離開螢幕——每一步都在等你把結果接到下一步。

這 14 次裡，有幾次用到了你的經濟學判斷？

你整晚實際在做的事

- 複製資料、貼上輸出
- 轉述錯誤訊息
- 用文字描述圖哪裡不對
- 確認檔案路徑與欄位名稱
- 重跑一次再貼回去

這些事，實驗室裡任何一個人都能做。

你沒空做的事

- CPI 該用 headline 還是 core？
- 樣本要不要含 2020–2021？
- 這條斜率能不能解讀成因果？
- 2010 年代近乎水平，是真的嗎？
- 這個結果值不值得寫成一篇論文？

這些事，全場只有你能做。

你把 100% 的注意力花在左邊，而你的全部價值在右邊。

接下來 30 分鐘要說明的是：這不是「AI 還不夠好」的問題，是你被放錯位置的問題。

知識分工的經濟學

為什麼「誰處理哪一種問題」決定了整個團隊的產出。

知識經濟與知識階層理論

Worker
第一線工作者 / RA

- 處理常規工作 ($x \leq z$)
- 遇瓶頸向上求助

Independent
獨立生產者

- 單打獨鬥包辦全程
- 不求助也不指導人

記住這一格。

Solver
高階專家 / 教授

- 不參與常規工作
- 提供高階解答與槓桿

- **能力與難度**：每人擁有 1 單位時間與知識 $z \in [0, 1]$，能獨立解決難度 $x \leq z$ 的生產問題。
- **隱性知識 (Tacit Knowledge)**：知識無法被 100% 編碼成 SOP 手冊，每次求助消耗 Solver h 的時間。
- **職業分工與組織**：時間與知識是核心瓶頸，透過雙層企業分工可放大產能並創造經濟剩餘。

Pre-AI 知識經濟：教授與研究生的知識分工

教授 / 學者 (s) 頂層 Solver（高階知識 $s \in [0, 1]$）

- 決定研究方向與模型架構
- 解決罕見、困難的理論/方法盲點
- 最終審核與負責論文品質

研究生 / RA (z) 第一線 Worker（基層知識 $z < s$）

- 下載清理資料、執行常規迴歸
- 處理簡單技術問題（難度 $x \leq z$）
- 遇瓶頸（$x > z$）前往 Office Hour 求助

Garicano (2000), *Journal of Political Economy*。

驅動整個模型的兩個核心公式與雙層組織

(A) 單一 RA 計畫的期望品質：完全由教授知識 s 保底

$$\underbrace{z}_{\text{RA 獨立解決}} + \underbrace{(1-z)}_{\text{RA 無法解決}} \cdot \underbrace{\frac{s-z}{1-z}}_{\text{教授解答率 } \Pr(x \le s \mid x > z)} = \mathbf{s}$$

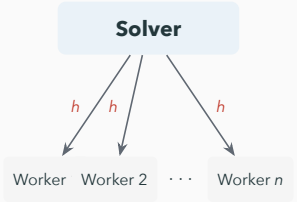
假設 $x \sim U[0,1]$。**注意右邊沒有 h**：溝通成本再高再低，單一計畫品質都是 s。

(B) 教授控制跨度（Span of Control）

RA 求助討論消耗教授 h 時間（溝通成本）：

$$h \cdot n(z) \cdot (1-z) = 1$$

$$\Longrightarrow \quad \mathbf{n(z) = \frac{1}{h(1-z)}}$$



Worker 人數 $n(z)$ 取決於 h 和 z

Garicano (2000), *Journal of Political Economy*。

Pre-AI 合作的經濟獲利：創造經濟大餅

單打獨鬥的瓶頸

- **RA 單獨做**：一個計畫，期望品質只有自身知識水準 z。
- **教授單獨做**：1 單位時間只能親手推進 1 個計畫，期望品質 s。

師生合作的剩餘（Surplus）

- **單一計畫品質**：由教授保底，升至 s。
- **教授控制跨度**：RA 的 z 越高，求助頻率 $(1-z)$ 越低。

$$Y_{\text{Pre-AI}} = \underbrace{n(z)}_{\text{計畫數}} \cdot \underbrace{s}_{\text{每個計畫的期望品質}} = \underbrace{\frac{s}{h(1-z)}}_{\text{團隊總產出（經濟大餅）}}$$

師生雙贏：RA 成果提升 ($w(z) > z$)，教授研究規模槓桿放大 ($w(s) > s$)。

Garicano (2000)；Garicano & Rossi-Hansberg (2006)。

你其實站在哪一格？

同一個模型，套用到你現在使用 AI 的方式。

同一個 AI，三種用法，決定你站在哪一格

不用 AI
自己從頭做到尾

- 常規工作：**你**
- 步驟銜接：**你**
- 卡住時：自己查

Independent
$n = 1$

用 Chat
它寫，你搬運

- 常規工作：AI 寫，**你跑**
- 步驟銜接：**你**（14 次）
- 卡住時：你轉述

仍然是 Independent
$n = 1$

用 Agent
它自己跑完迴圈

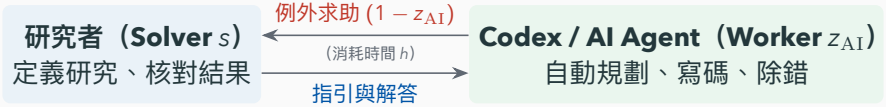
- 常規工作：**AI**
- 步驟銜接：**AI**
- 卡住時：才問你

回到 Solver
$n \gg 1$

Chat 讓你打字更快，但沒有把你從第一線移開。

Independent 的產出函數裡沒有 n 這一項。要讓 n 出現，必須有人在你之下**獨立跑完**整個迴圈。

Post-AI 時代：Codex 成為你的「數位 RA」



$$Y_{\text{Post-AI}} = \frac{s}{h(1 - z_{\mathrm{AI}})}$$

1. AI 基礎能力 z_{AI}

z_{AI} 越高，常規問題出錯率 $(1 - z_{\mathrm{AI}})$ 越低，減少打擾研究者的頻率。

2. 人機溝通成本 h

h 越小，研究者解答與交接所耗費的時間越少，槓桿放大的空間越大。

Ide & Talamàs (2025, JPE), *Artificial Intelligence in the Knowledge Economy*。

Chat vs. Codex：管理 RA 的溝通成本 h 差異

Chat

(Non-autonomous Copilot)

- 手動複製、寫長指令、貼回結果
- 每一步都需人工參與轉傳
- **高溝通成本 h_{Chat}**

$$n_{\text{Chat}} = \frac{1}{h_{\text{Chat}}(1-z_{\text{AI}})} \approx \mathbf{1}$$

人類被困在第一線盯著 1 個視窗

Codex

(Autonomous Agent)

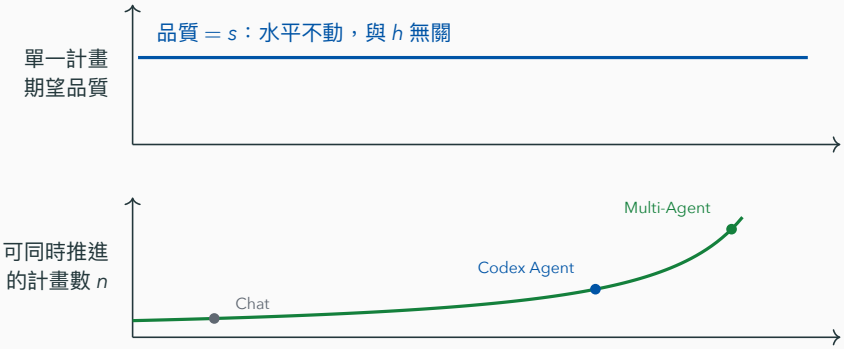
- 自動讀取 `AGENTS.md` 上下文
- 自動跑迴圈、執行工具與除錯
- **低溝通成本 h_{Codex}（例外管理）**

$$n_{\text{Codex}} = \frac{1}{h_{\text{Codex}}(1-z_{\text{AI}})} \gg \mathbf{1}$$

人類可同時指揮多個 Agent 並行流水線

Codex 的核心價值：將溝通耗損 h 降至極低，釋放人類 Solver 的管理跨度 n！

你的 s 動不了，但你的 n 沒有上界



溝通效率提升（$h \to 0$，例如把規則一次寫進 `AGENTS.md`）

地板由你保底，天花板被打開。這才是模型真正在說的事。

AI 再快也拉不高你的 s，AI 出錯也拉不低它——因為最後審稿的是你。

模型的預測：你的 s 越高，Agent 給你的槓桿越大

自主 AI 的獲益，**隨使用者自身的判斷力 s 遞增**。

因為品質由 s 保底、產量由 n 放大，兩者相乘後 s 越高、放大的絕對量越大。

很多人的直覺

「AI 是年輕人的東西，它會把我幾十年累積的判斷力變得不值錢。」

模型的結論

正好相反：s 是唯一**不能**被外包的東西，而它現在能乘上一個更大的 n。

反過來說：死守 Chat，等於自願讓自己的 s 只乘上 $n = 1$。

Ide & Talamàs (2025, JPE), *Artificial Intelligence in the Knowledge Economy*.

我怎麼同時跑三個專案：s 不變，n 放大

低涉入·只在關鍵點出現

Project A：檢定方法的蒙地卡羅模擬

- 新的統計檢定方法，需要蒙地卡羅測試表現
- AI → 撰寫程式、監控進度、整理結果

高涉入·幾乎全程在場

Project B：整理結果、寫成論文

- 結果已經完成，進入寫作階段
- 需要更頻繁的往返，AI 逐次修改表格、圖形

低涉入·一次把規格定案

Project C：開學後的上課投影片

- 我已經明確定義好規格
- AI → 從課本、習題、考古題產生投影片與作業

正因為每個階段需要我的程度不同，AI 才能接走「低涉入」的部分——讓我能同時把三個專案一起往前推。

呼應上一頁：s 沒有被稀釋，只是我現在能把它乘上更大的 n。

我知道你在想什麼

你的疑慮	模型怎麼回答
「它會亂編，我不敢放手。」	公式 (A)：品質由**你的** s **保底**。你的工作從「寫」變成「審」——驗證的成本 $\leq$ 推導。
「我不會寫程式，看不懂怎麼驗證。」	你要驗證的不是程式碼，是**經濟學結論**：資料對不對、期間合不合理、這句話能不能說成因果。
「學這個要花多少時間？」	一次性成本。規則寫進 `AGENTS.md` 之後，h 只付一次，之後每個新任務都免費繼承。
「我已經有真的 RA 了。」	模型不排斥——n 是可加的。Agent 補的是連 RA 都排不進去的那些常規段落。
「資料能不能放上去？」	敏感資料、未公開資料、審稿中的稿件，仍須依所屬單位規範判斷。

那要怎麼把 h 降下來？

把常規工作交出去、把規格講清楚、讓工具替 Agent 保證精確度。

哪些常規任務（$x \leq z_{\mathrm{AI}}$）可以全權交給 Agent？

1. 資料處理與清理

- 讀取 CSV / API 抓檔
- 日期與缺值處理
- 西元/民國與跨表合併

2. 程式繪圖與估計

- 時序圖與散布圖繪製
- 執行 OLS/Panel 迴歸
- 匯出標準 TeX 估計表

3. 工作流與簡報排版

- 專案資料夾結構自動化
- 產出可重跑分析腳本
- 成果轉 Beamer/PPTX

規則明確、邊界清晰的常規步驟（$x \leq z_{\mathrm{AI}}$），交給 Agent 獨立完成免打擾！

學者只需在關鍵例外節點（$x > z_{\mathrm{AI}}$）介入驗證，即可達成極大化的時間槓桿。

AI 的獨特之處：能處理不可編碼的認知工作

		知識是否可編碼	
		可編碼	不可編碼
工作類型	手動	Robots	Physical AI
	認知	企業軟體	**AI**

AI 的定義特徵：它能執行仰賴「不可編碼（tacit）知識」的認知工作。

這使 AI 有別於自動化可編碼認知工作的企業軟體，以及自動化手動工作的機器人；也不同於「physical AI」——未來可能自動化不可編碼手動工作（如照護、營建）的機器人構想。

Ide & Talamàs (2025, JPE), *Artificial Intelligence in the Knowledge Economy*, table 1.

重塑工作流程的核心：結合 AI 與企業軟體（SOP）

1. 企業軟體 / 工具的價值（SOP）

- **品質與精確度保證**：OLS 估計值、t 檢定值 100% 精確，免去計算出錯風險。
- **極致速度與省時**：避免每項常規任務都讓 AI 從頭重新刻程式或造輪子。

2. AI Agent 的角色（Orchestrator）

- **靈活調用與銜接**：理解人類意圖，自動選擇並正確調用對應的軟體 SOP。
- **例外處理與除錯**：當軟體執行報錯或遇到邊界狀況時，自主修正與重跑。

最佳工作流：用 AI 作為大腦進行調度，讓既有軟體與 SOP 負責高效精確執行！

之後想再加速：往「AI 友善」的工具移動

今天不用換任何工具，你現在的環境就能開始。

以下是之後想再加速時，一次換一項的選項。

1. AI 友善軟體的三大特徵

- **純文字與開放格式**：無二進位黑盒子，AI 可直接讀寫、可版本控制。
- **CLI / API 介面**：提供命令列或 Local API 供 Agent 自主呼叫。
- **程式化與可重跑**：腳本自動執行，而非人工點擊 GUI。

2. 之後可以一次換一項

- **文獻管理**：EndNote → **Zotero**
- **知識筆記**：Word → **Obsidian**
- **論文簡報**：GUI 排版 → **LaTeX / Typst**
- **統計分析**：點選軟體 → **Python / R**

每換一項，Agent 能替你獨立跑完的範圍就大一塊，*h* 就再低一點。

人的角色從逐步執行上移到關鍵判斷

1. AI 自主規劃與執行（$x \leq z_{\mathrm{AI}}$）

- 拆解並執行已定義的常規任務
- 檔案處理、程式編寫與自動除錯
- 跨工具與軟體 SOP 自動運轉

2. 人類的核心判斷與驗證（s）

- 尋找研究問題與建立核心假說
- 判斷成果與資料可信度 (Verification)
- 負責最終的經濟直覺與因果解釋

重新思考人機邊界：各學科更上層的知識 s 是什麼？AI 時代該培養學生什麼？

當執行與工具操作成本降至零，教育與研究的焦點必須從「步驟操作」上移至「問題定義與批判研判」。

Ide & Talamàs (2025, JPE) 主結果：Autonomous AI 讓具備高階判斷力（High s）的解答者獲益最大。

Agent 越自主，事前溝通規格的報酬越高

已知已知 你已經寫進 prompt 的邊界與實作計畫	**已知未知** 請 AI 訪談你、逐一提問補齊規格
未知已知 腦力激盪、原型與參考範例，喚醒隱性標準	**未知未知** 請 AI 指出盲點、事前審查計畫

只有左上角「已知已知」是你初始寫出的內容；另外三格全靠事前對話挖出來！

同一個模糊的規格，在 Chat 只錯一步；但在自主 Agent 會被放大成 *N* 步的偏離。

Shihipar (2026), *A Field Guide to Claude Fable: Finding Your Unknowns*。事前對話降低溝通成本：Crémer, Garicano & Prat (2007, QJE).

象限一【已知已知】：直接指派明確指令與 SOP

學者輸入的 Prompt（指令明確）

「請讀取 `data/fred_cpi_unrate.csv`，計算通膨率 π_t 與失業率 u_t 的 OLS 迴歸，並匯出符合 AEA 規範的 LaTeX 估計表格。」

AI Agent 的自動執行與產出

- 自動編寫 Python/R 迴歸腳本
- 呼叫既有工具庫（如 `statsmodels`）
- 自動產生 `phillips_table.tex` 估計表

Known Knowns 直接交辦！規格在 Prompt 裡寫明，AI 便能自動執行完整 SOP。

這是最基本的代理情境：研究者已知需求與細節，Agent 獨立完成無偏差。

象限二【已知未知】：透過專案對話與 AI 訪談（Interviews）

學者輸入的 Prompt（開啟訪談）

「我想分析飛利浦曲線的結構變遷，但我不確定細節。請作為計量專家逐步訪談我，提出優先會影響架構設計的 3 個關鍵問題。」

AI Agent 的引導式提問

- **斷點位置**：已知特定年份（Chow）或自動搜尋未知斷點（Sup-Wald）？
- **資料頻率**：月資料還是季資料？
- **控制變數**：是否納入進口物價衝擊？

Known Unknowns 用訪談釐清！讓 AI 逐一提問，填補你意識到尚未說明的規格。

當你知道目標但尚待決定架構時，讓 AI 擔任訪談者能最快收斂模糊需求。

象限三【未知已知】：製作原型（Prototypes）喚醒隱性知識

學者輸入的 Prompt（要求原型）

「我說不出想要的圖表形式，但我看到就知道！請先生成 3 種截然不同的視覺原型（散布圖 +Loess、滾動迴歸、子分期比較），讓我反應。」

AI Agent 產出的多樣原型

- **原型 A**：非線性平滑曲線 (Loess Fit)
- **原型 B**：10 年滾動 OLS 係數變化圖
- **原型 C**：分年代 (Pre/Post-2008) 對比圖

Unknown Knowns 用原型對齊！將學者腦中隱性的直覺審美轉化為具體標準。

在寫死正式程式碼前先產出輕量原型，能大幅節省事後大規模重構與修改成本。

象限四【未知未知】：發動盲點審查（Blind Spot Pass）

學者輸入的 Prompt（盲點審查）

「這是我研究飛利浦曲線的分析計畫。請執行 Blind Spot Pass，扮演嚴格審稿人找出我完全沒考慮到的 Unknown Unknowns 與陷阱。」

AI Agent 挖掘出的隱形盲點

- **非平穩性**：序列為 *I*(1)，未做單根與共整合檢定，會有偽迴歸風險
- **前瞻偏誤**：季資料歷史修正未對齊即時版本
- **NAIRU 漂移**：未考量自然失業率動態變化

Unknown Unknowns 用盲點審查排除！在發動 Agent 批量執行前先設護欄。

事前調用「Blind Spot Pass」提示詞，是避免 Agent 在錯誤假設下越走越遠的關鍵。

統合四象限：三階段 Agentic 工作流程

1. 實作前對齊

- **Blind Spot Pass**
  排除未知未知
- **Prototypes**
  喚醒未知已知
- **Interviews**
  補齊已知未知

2. 實作中執行

- **寫入 AGENTS.md**
  固定規格與規則
- **發動 Agent**
  自主寫程式與除錯
- **執行 SOP**
  處理已知已知

3. 實作後驗證

- **變更與報告**
  檢視改動紀錄
- **Quiz / Q&A**
  確保深度掌控細節
- **最終品質把關**
  確定符合學術標準

先對齊邊界，再發動 Agent！將對話轉化為高效且可控的自動化流程。

這套三階段架構，能讓研究者既享高度時間槓桿，又 100% 確保成果品質。

動手：從空資料夾到飛利浦曲線

接下來每一步，都可以打開、執行、播放，並且要求 Codex 修改。

你會完成一套從資料到簡報的可重跑分析

- 每一階段都有可以打開、執行或播放的成果。
- 每一階段也都能要求 Codex 根據你的判斷修改。

150 分鐘：示範、分段練習、成果發表

75

分鐘示範

講師走完一次完整工作流。

45

分鐘實作

每組完成六段任務。

30

分鐘發表

最後播放生成的簡報。

課程順序：**問出未知** → **看見走勢** → **估計關係** → **比較年代** → **檢查解讀** → **製作簡報**。

現在建立你的第一個 Codex 專案

1. 在桌面建立一個新的空白資料夾，命名為 `fred-demo`。
2. 開啟 ChatGPT 桌面 App，從左側切換到 **Codex**。
3. 選擇 **Open folder**，開啟剛才建立的 `fred-demo`。
4. 在新的 Codex thread 貼上下面的提示，再按下送出。

請先查看目前資料夾，告訴我裡面有哪些檔案。先不要建立或修改任何東西。

預期結果：Codex 告訴你資料夾目前是空的，且尚未建立任何檔案。

OpenAI, `https://learn.chatgpt.com/docs/app` and `https://learn.chatgpt.com/docs/integrated-terminal`.

美國通膨率與失業率從 2000 年以來如何變化？

CPIAUCSL

CPI，月資料，季節調整。
用來計算 12 個月通膨率。

UNRATE

失業率，月資料，季節調整。
單位為百分比。

$$\text{通膨率}_t = \left(\frac{\text{CPI}_t}{\text{CPI}_{t-12}} - 1 \right) \times 100.$$

想一想： headline 還是 core CPI？季調還是未季調？這是你的「已知未知」，還是根本沒想到的「未知未知」？

FRED series pages: `CPIAUCSL` and `UNRATE`.

先讓 Codex 問，再把任務交辦下去

第一步：先問（J_s）

在動手之前，請先問我 5 個最可能影響結果的問題，並指出你認為我沒想到的地方。先不要寫任何程式。

值得保留的回答，稍後寫進 `AGENTS.md`。

第二步：交辦

成果	下載兩個 series、計算通膨、合併、畫圖並寫 README。
位置	原始資料放在 `data`；圖表與結果放在 `output`。
限制	從 2000 年開始；原始資料不手動修改。
停點	開始前先說明計畫；需要網路或套件時先說明原因。

指定「什麼算完成」，不要替 Codex 寫每一行程式；但先讓它把你的未知問出來。

Direct CSV example: `https://fred.stlouisfed.org/graph/fredgraph.csv?id=CPIAUCSL`

本課直接下載 CSV，不使用 API key

	工作坊直接 CSV	正式 FRED API
API key	不需要	需要
上手成本	低	較高
格式	CSV	XML / JSON / XLSX / CSV
本課角色	立即完成第一個工作流	第二次進階課

直接下載範例：

```
https://fred.stlouisfed.org/graph/fredgraph.csv?id=CPIAUCSL
```

FRED API documentation: `https://fred.stlouisfed.org/docs/api/fred/series_observations.html`.

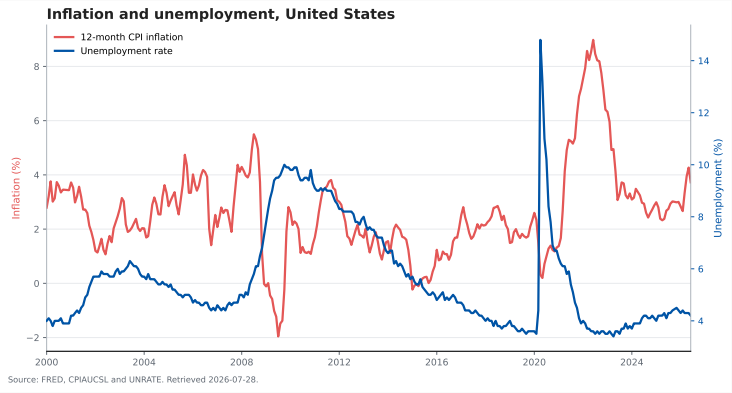
資料夾結構讓成果可以重跑

```
fred-demo/
|-- data/
|   |-- CPIAUCSL.csv
|   `-- UNRATE.csv
|-- output/
|   `-- inflation_unemployment.png
|-- analysis.py
`-- README.md
```

先確認四件事

- 原始資料有保留
- 分析程式可執行
- 圖表有固定位置
- README 說明如何重跑

先檢查圖表，再解讀走勢



先問自己

- 若拿掉右側刻度，兩條線還能正確比較嗎？
- 確認期間從 2000 年開始，且兩個單位清楚。
- 確認 series ID 與下載日可以追溯。

用一次明確修改改善圖表

請不要使用雙 Y 軸。

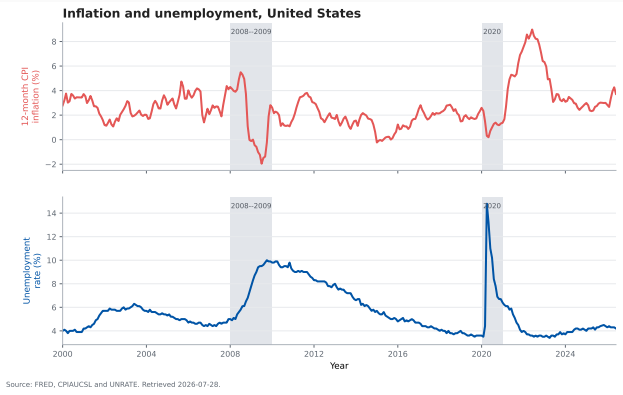
改成上下排列的兩張圖，並標示 2008–2009 年與 2020 年。

這次修改要求

- 分開尺度，降低誤讀
- 標出關鍵期間
- 重新執行並更新輸出

人提出判斷 → Codex 修改 → 人重新檢查

修改後的圖表更容易比較



- 兩個變數各自使用清楚的尺度，不需要雙 Y 軸。
- 2008–2009 與 2020 的標示讓讀者更快定位關鍵變化。

下一步：把這些檢查標準寫入專案規則，避免每次重新交代。

「寫給 AI 研究助理看的工作守則」

回到模型：這就是把 h 一次付清。 寫一次規則，之後每個新 thread 都免費繼承——邊際溝通成本趨近於零。

全域

個人長期偏好。

專案

本課建立的 `AGENTS.md`。

子目錄

更專門的局部規則。

本課只做一件事：在 `fred-demo` 根目錄建立短而明確的規則。

OpenAI, `https://learn.chatgpt.com/docs/agent-configuration/agents-md`.

四組規則讓成果更容易檢查

資料
原始 FRED 資料放在 `data`，禁止手動修改。

輸出
圖表與結果放在 `output`，標示來源、期間與單位。

措辭
相關性不寫成因果關係。

重現
程式可從頭執行；完成後列出異動檔案。

用新 thread 讓規則帶入下一項分析

請延伸目前分析，估計一條描述性的飛利浦曲線，並比較不同年代。

檢查 Codex 是否自動遵守

- 不修改原始資料
- 輸出放在 `output`
- 圖表標示來源與單位
- 結果不宣稱因果
- 最後列出異動檔案

注意這裡：我一個字都沒有重講，規則自己跟過來了。

這是 Chat 結構性做不到的事——每開一個新對話，*h* 就要重付一次。短提示只說這次的新任務，資料與輸出規則由 `AGENTS.md` 延續。

從時間序列走勢，進一步估計條件平均關係

$$\pi_t = \alpha + \beta u_t + \varepsilon_t$$

- π_t CPI 的 12 個月通膨率
- u_t 當月失業率
- β 散布圖中的描述性斜率

本課的最小估計

- 月資料
- 普通最小平方法
- 全期與分年代
- 報告斜率、R^2、N

這不是結構式飛利浦曲線；$\hat{\beta}$ 不代表失業率對通膨的因果效果。

用自然語言指定估計、比較與輸出

請在現有 Python 分析中加入描述性飛利浦曲線。

以月資料估計 $\pi_t = \alpha + \beta u_t + \varepsilon_t$，其中 π_t 是 CPI 12 個月通膨率，u_t 是失業率。先估計 2000 年至今，再分成 2000–2009、2010–2019、2020–最新資料。

請產生全期散布圖、三個年代使用相同座標尺度的比較圖，以及包含截距、斜率、R^2、樣本數的表格。標示資料來源，並把結果稱為描述性關係，不要宣稱因果。

想一想： 為什麼三個年代要使用相同的 X、Y 軸範圍？

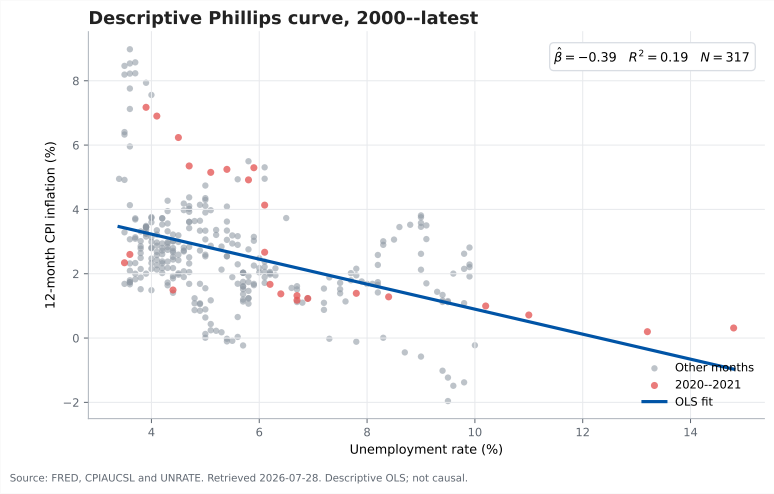
延伸分析會增加圖表與估計表

```
fred-demo/
|-- data/
|   |-- CPIAUCSL.csv
|   `-- UNRATE.csv
|-- output/
|   |-- inflation_unemployment.png
|   |-- phillips_curve_overall.png
|   |-- phillips_curve_by_era.png
|   `-- phillips_curve_estimates.csv
|-- analysis.py
|-- README.md
`-- AGENTS.md
```

新增成果

- 一張全期散布圖
- 一張三年代比較圖
- 一份估計結果表
- 可重跑的更新程式

全期斜率只是 317 個月份的平均描述



讀圖順序

1. 先看散點分布
2. 再看斜率方向
3. 檢查 R^2 與樣本數
4. 找出特殊月份

紅點標出 2020-2021，不代表可以直接刪除。

分年代後，估計斜率並不穩定

Period	Intercept	Slope $\hat{\beta}$	R^2	N
2000–latest	4.79	-0.39	0.19	317
2000–2009	6.41	-0.69	0.50	120
2010–2019	1.25	0.08	0.04	120
2020–latest	6.62	-0.56	0.27	77

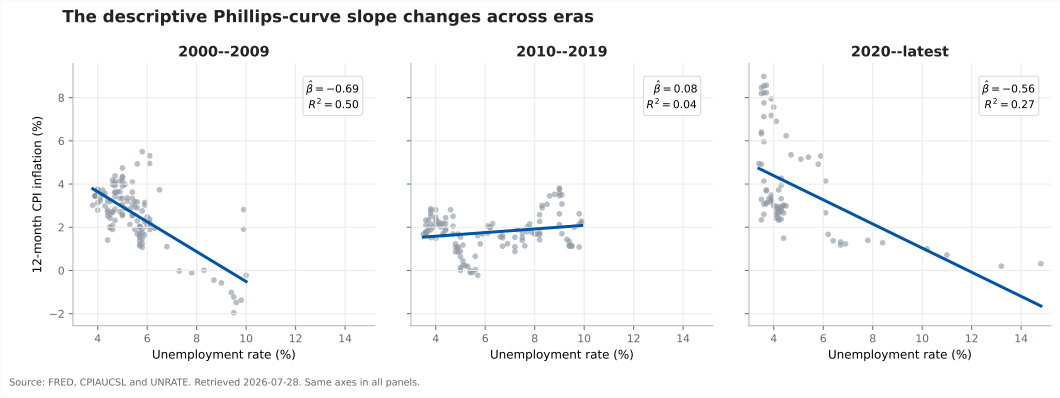
2000-2009
斜率約 -0.69

2010-2019
斜率約 $+0.08$

2020-最新
斜率約 -0.56

FRED `CPIAUCSL` and `UNRATE`; retrieved 2026-07-28. Descriptive OLS.

相同尺度讓年代差異可以直接比較



2010–2019 的線近乎水平；全期負斜率不能代表每一個年代。

估計完成後，先寫限制再寫故事

這個結果能說什麼？

- 樣本中的同期相關方向
- 全期與分年代的斜率差異
- 哪些月份可能影響平均線

這個結果不能排除什麼？

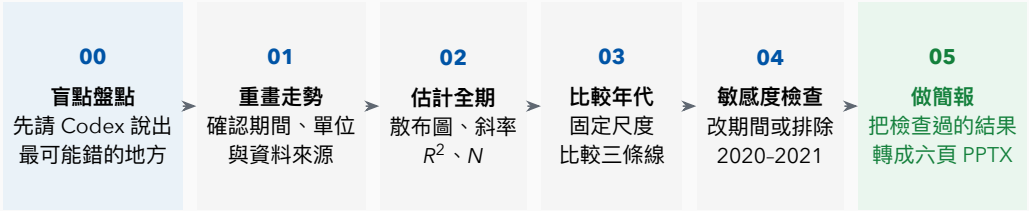
- 通膨預期與供給衝擊
- 序列相關與共同趨勢
- 政策反應與制度變化

想一想： 2010–2019 的斜率接近零，是「沒有飛利浦曲線」，還是簡單模型漏掉重要變數？

換你動手，然後把結果變成簡報

45 分鐘實作，每一段都留下可以檢查的輸出。

六段練習從盲點盤點推進到可交付的簡報



每完成一段就先檢查輸出，再把下一段要求交給 Codex。

第 00 段是事前的 J_s：先把未知問出來，後面五段的驗證成本才會下降。

練習的重點是提出檢查，而不是接受第一版

練習 0：盲點盤點
請說出這個分析最可能錯的三個地方。先不要改任何程式。

練習 A：改變樣本
將起始年改為 1985 年，重新估計全期與分年代結果。哪些結論改變？

練習 B：檢查特殊期
排除 2020–2021 後重估，與原結果並列比較，不要覆蓋原圖。

練習 C：轉成簡報
把已檢查過的結果做成六頁 PPTX，不重新計算、不補造數字。

每組至少回答

1. 哪個結果最穩定？
2. 哪個結果對期間最敏感？
3. 圖表與表格是否一致？
4. 哪一句解讀需要降級？
5. **練習 0 列出的盲點，哪幾個真的被 A、B 證實了？**

進階選做：改用核心 CPI `CPILFESL`，比較 headline 與 core inflation。

75 分鐘示範：先完成分析，最後說明交付格式

0-10	**你昨天的對話：14 次手動銜接，你的判斷力全程閒置。**
10-20	知識分工的經濟學：品質由 *s* 保底，跨度由 *h* 決定。
20-32	**你站在哪一格：Chat 讓你留在 Independent；並回應五個常見疑慮。**
32-40	怎麼把 *h* 降下來：交辦範圍、AI 與 SOP 分工、四象限與三階段。
40-45	建立並開啟空白資料夾，先讓 Codex 提問再交辦。
45-56	下載 FRED、建立分析、畫圖，並要求一次明確修改。
56-62	**把規則寫進 `AGENTS.md`，開新 thread 驗證規則自動繼承。**
62-72	估計全期與分年代飛利浦曲線，比較斜率並討論限制。
72-75	說明六段練習與最後六頁簡報。

45 分鐘實作：每一段都有可檢查的輸出

5 分鐘	7 分鐘	9 分鐘	9 分鐘	6 分鐘	9 分鐘
盲點盤點 先問再動手	**資料與走勢** 重跑與核對	**全期估計** 散點與 OLS	**年代比較** 固定尺度	**敏感度** 改樣本重估	**做簡報** 六頁 PPTX

完成標準：一份盲點清單、程式可重跑、三張分析圖、一份估計表、一項敏感度檢查、六頁簡報。

分析結果確認後，才把它轉成簡報

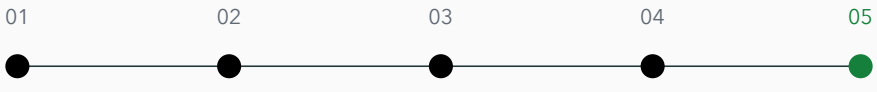
資料與定義 → 時間序列圖 → 全期估計 → 年代比較 → 限制

⇓

把已經檢查過的結果整理成六頁簡報

簡報不重新計算、不補造數字；它只讀取分析程式已產生的圖表與表格。

簡報工作流包含建立、渲染、檢查與修正



六頁簡報依序回答研究問題與限制

01 研究問題：2000 年以來通膨與失業如何變化？

02 資料與定義：series ID、期間、頻率、單位與通膨公式。

03 時間序列：上下排列圖與一句主要觀察。

04 全期飛利浦曲線：散布圖、斜率、R^2 與樣本數。

05 年代比較：三個時期使用相同尺度並列。

06 發現與限制：哪些關係穩定、哪些對期間敏感、為何不能因果解讀。

本課使用可逐頁檢查的 PPTX 工作流

本課使用

Presentations / Slides skill

- 建立原生、可編輯的 `.pptx`
- 建立後渲染並逐頁檢查
- 可要求重新排版或縮短內容

備選與後續

Quarto 可重現，適合進階研究流程。

PptxGenJS 版面控制高，背後程式較複雜。

Google Slides 協作方便，但增加連線與權限設定。

實作時，以 Codex 環境中實際顯示的簡報 skill 名稱為準。

用一個提示生成並檢查六頁簡報

請根據目前資料夾中的分析結果，建立一份六頁 PowerPoint。

內容：研究問題｜資料與定義｜時間序列｜全期飛利浦曲線｜分年代比較｜主要發現與限制。

規則：只使用分析程式已產生的圖表與表格；每頁一個重點；不得把描述性斜率寫成因果效果。

完成：存入 `output`；渲染逐頁檢查；修正溢出、重疊與字型；列出 series ID 與異動檔案。

若環境可用，請使用 Presentations / Slides skill 建立可編輯的 `.pptx`。

回到你身上

Agent 拿走了執行，沒有拿走判斷——那正是它拿不走的部分。

三分鐘發表：問題、證據、判斷

第 1 分鐘

研究問題

使用哪些 FRED series？想觀察什麼？

第 2 分鐘

展示成果

播放六頁簡報，指出年代比較的主要差異。

第 3 分鐘

反思判斷

哪個結果最敏感？哪一句不能因果解讀？

四項判斷仍由經濟學者負責

01　資料選擇

series、樣本與修訂狀態。

02　變數定義

單位、轉換與缺值處理。

03　因果推論

描述性關係不等於因果。

04　經濟解釋

制度背景與替代機制。

想一想： 如果程式成功執行，但通膨定義錯了，這份結果算完成嗎？

Dell'Acqua et al. (2023), *Navigating the Jagged Technological Frontier*：在 AI 不擅長的任務上，使用 AI 的受試者表現反而更差。

回到那一格：你要當 Independent，還是 Solver？

讓 Codex 採取行動，留下可以檢查的工作；
讓經濟學者負責定義、因果與解釋。

1. Chat 需要人銜接每一步，所以 n 永遠是 1；Codex 讓 AI 自己跑完迴圈。
2. 不必先會寫程式，也能建立可執行、可修改的分析。
3. `AGENTS.md` 把 h 一次付清，人的研究責任不會因此消失。

今天唯一要你改變的，不是你的研究，是你在自己計畫裡站的位置。