為 Claude 4.5 終端機 AI 代理打造「最佳 UX」的視覺化架構藍圖

I. 終端機環境下的「最佳 UX」視覺化原則

本章節將奠定我們的方法論基礎。在我們談論「如何實現」之前，我們必須嚴格定義 AI 代理需要達成的「目標是什麼」。在終端機的限制下，「最佳 UX」是一種在「資訊密度」、「傳真度」和「執行速度」之間的精確平衡。

1.1 重新定義終端機 UX：超越純文字的「資料-墨水比例」

傳統的終端機 UX 優先考慮文字的簡潔性。然而，對於資料分析，「最佳 UX」意味著「最佳洞察」。我們將借鑒 Edward Tufte 的核心原則：最大化「資料-墨水比例」（data-ink ratio）[5]。

AI 代理的輸出不應該只是 ASCII 表格。它必須有能力（在技術允許的情況下）提供高解析度的圖形表示，將終端機從一個「指令介面」提升為一個「分析畫布」。

在自動化生成中，避免「圖表垃圾」（chartjunk）[6] 比手動繪圖更為重要。AI 代理必須被明確指示刪除不必要的格線、花俏的背景、以及無意義的裝飾 [6]。如 [15] 的評論一針見血地指出：「漸層不是你的朋友，圓餅圖是戰爭罪」。代理生成的圖表應優先考慮清晰度和簡潔性，去除所有不傳達資料的視覺元素。

1.2 代理的「情境感知」：基於意圖的圖表選擇

「最佳 UX」始於選擇正確的圖表類型 [7]。AI 代理的智慧不應體現在它能畫出多少種圖表，而應體現在它能根據使用者的意圖，準確地選擇那一種圖表。

代理應具備以下意圖映射（Intent-to-Chart Mapping）能力：

- 比較 **(Comparison):** 當使用者詢問「A vs B」，代理應自動選擇條形圖 (Bar Chart) [8]。
- 趨勢 **(Trend):** 當使用者詢問「隨時間變化」，代理應自動選擇折線圖 (Line Chart) [8]。
- 分佈 **(Distribution):** 代理應選擇直方圖 (Histogram) 或箱型圖 (Box Plot) [9]。
- 關聯 **(Relationship):** 代理應選擇散點圖 (Scatter Plot) [8]。

Claude 4.5 的角色至關重要。代理的內部提示詞 (prompt) 應利用 Claude 4.5 的「思考」能力 [11]，在生成視覺化之前，先明確地陳述它所識別的「使用者意圖」以及「選擇該圖表類型的理由」。

1.3 AI 代理的視覺化戒律：嚴格禁止低效圖表

某些圖表類型在認知上是有缺陷的。AI 代理絕不能在追求「視覺吸引力」時犧牲「資料準確性」。

第一戒：嚴禁 3D 圖表。
3D 圖表（特別是 3D 圓餅圖和 3D 條形圖）會因透視失真 (perspective distortion) 而誤導使用者，使其無法準確比較數值 12。
第二戒：嚴禁圓餅圖 (Pie Charts)。
14 尖銳地指出「我痛恨圓餅圖」。人類的眼睛不擅長準確比較二維空間的面積或角度 13。代理的系統提示詞應包含一條規則：「如果使用者要求圓餅圖，總是禮貌地拒絕，並提供一個排序後的條形圖作為更清晰的替代方案。」15。

1.4 終端機美學：為 CLI 優化的深色模式調色盤

您的代理在終端機中運行，這意味著它極有可能在「深色模式」(Dark Mode) 介面中顯示。錯誤的顏色選擇會導致嚴重的可讀性問題。

#121212 原則：
最佳的深色模式背景不是純黑色 (#000000)，因為純黑與純白的對比度過高，會引起眼睛疲勞 16。Google Material Design 推薦的 #121212 是一個更柔和、更優雅的深灰色背景 16。
去飽和度 (Desaturation)：
在深色背景上，明亮、飽和的顏色會「發光」，分散注意力且難以閱讀 17。代理在生成圖表規格（例如 Vega-Lite）時，必須同時生成一個 config 物件 18，將背景設為 #121212，並為資料系列選擇一組「去飽和度」的調色盤。

II. 視覺化文法：為何 Vega-Lite 是 AI 代理的首選

本章節將進行核心技術選型。您的 AI 代理需要一個「目標語言」來描述視覺化。在「命令式」和「聲明式」之間，這個選擇將從根本上決定您工作流程的可靠性和可擴展性。

2.1 典範之爭：聲明式 (Vega-Lite) vs 命令式 (ECharts, Chart.js)

- **命令式 (Imperative):** 像 Apache ECharts [19] 或 Chart.js [21] 這樣的函式庫，要求您編寫 JavaScript 程式碼來「逐步執行」圖表的構建：「創建一個畫布，添加一個 X 軸，循環資料並繪製每個條形...」。
- **聲明式 (Declarative):** 像 Vega-Lite [9] 這樣的文法，您只需提供一個 JSON 檔案來「描述」最終結果：「這是一個條形圖 (mark)，X 軸對應『銷售額』(encoding)，Y 軸對應『類別』。」編譯器會負責處理所有繪圖細節 [25]。

2.2 為何 Vega-Lite 是 AI 生成的必然選擇

對於人類開發者來說，命令式程式碼提供了極致的靈活性 [20]。但對於 AI 代理，這種靈活性是一場「惡夢」。Vega-Lite 的聲明式特性使其成為 Claude 4.5 代理在架構上的必然選擇。

1. JSON 原生：LLM 的「母語」
Claude 4.5 26 和其他 LLM 本質上是強大的「結構化資料」生成器。JSON 是它們最容易、最可靠的輸出格式 28。要求 LLM 生成一個 Vega-Lite JSON 規格，是一個「資料生成」任務，其成功率遠高於「程式碼編寫」任務。
2. 抽象性：專注於「什麼」，而非「如何」
使用 ECharts，Claude 4.5 需要正確處理 JavaScript 語法、非同步回調、DOM 互動 30，以及 ECharts 特有的 option 物件結構 19。這非常脆弱。使用 Vega-Lite，Claude 4.5 只需要專注於「資料欄位」和「視覺通道」之間的映射 24。這極大地「縮小了錯誤範圍」。
3. 文法一致性與學術嚴謹性
Vega-Lite 是一個「視覺化文法」(A Grammar of Interactive Graphics) 8。其 API 具有高度的一致性和紀律性，且其設計基於人類視覺感知的學術研究 31。這意味著，一個學習了 Vega-Lite 文法的 AI 代理，可以更可靠地將在「散點圖」上學到的編碼知識（例如 $color: { "field": "category", "type": "nominal" }$）應用到「地圖」上，而無需學習一個全新的 API。

2.3 提示工程：指導 Claude 4.5 生成 Vega-Lite JSON

我們的目標是讓 Claude 4.5 可靠地生成有效的 Vega-Lite JSON。我們應該使用結構化提示（如 XML 標籤 [11] 或 JSON 格式 [28]）來約束 AI。

提示範例 **(Prompt Example)**：

XML

```
<system_prompt>
您是一位專精於 Vega-Lite v5 (https://vega.github.io/vega-lite/docs/) 的資料視覺化專家。
您將嚴格遵循 Tufte 原則，最大化資料-墨水比例，並避免不必要的「圖表垃圾」。
根據 HCI 原則，您絕不允許生成 3D 圖表或圓餅圖 (Pie Charts)，因為它們會扭曲資料感知 [12, 14]。
您的所有圖表都必須使用針對終端機優化的深色模式主題。
</system_prompt>

<user_prompt>
我需要分析以下資料，比較每個「區域」(region) 的「平均銷售額」(avg_sales)。
資料：
<data>

</data>

請為此任務生成一個 Vega-Lite JSON 規格。
在您的 <thinking> 標籤中，請先執行以下步驟：
1. 識別使用者的核心意圖 (例如：比較、趨勢、分佈)。
2. 根據意圖選擇最佳的 'mark' 類型。
3. 定義 'x' 和 'y' 的編碼 (encoding)，包括 'field', 'type', 和任何必要的 'aggregate'。
4. 加入一個 'config' 物件，以實現 HCI 最佳實踐的深色模式：
  - 'background' 應為 '#121212'。
  - 'view' 的 'stroke' 應為 null (移除邊框)。
  - 'axis' 和 'legend' 的 'titleColor' 和 'labelColor' 應為淺灰色 (例如 '#CCCCCC')。
  - 選擇一組適合深色背景的去飽和度調色盤 (例如 'tableau10' 或 'set2')。
</thinking>
<final_response>
請只回傳 <json_output> 標籤中的 JSON 區塊。
</final_response>
```

上述的 <thinking> 步驟利用了「引導式發現」(Guided Discovery) [32] 和「鷹架」(scaffolding) 的概念

，它強迫 LLM 在回答之前先進行自我反思和規劃 [11]，確保了語義的準確性。

III. 策略一 (低傳真度)：TUI 內建圖表渲染

這是您的「基礎層」或「回退方案」。當高傳真度不可行或不需要時，TUI 提供了最快、最具可攜性的視覺化。

3.1 概念：在終端機中「繪製」

此策略完全不依賴圖形介面。它利用 ASCII、Unicode 區塊字元、或更先進的盲文 (Braille) 字元 [34] 來直接在終端機緩衝區中「繪製」圖表。

- 優點：100% 跨平台（適用於任何 SSH 會話、任何終端機）、零外部依賴、速度極快。
- 缺點：解析度極低、資訊密度有限、完全沒有互動性。

3.2 關鍵函式庫：Python 的 plotext

plotext [36] 是此領域的佼佼者。它提供了一個與 matplotlib 極為相似的 API [38]，使其易於學習和使用。它支援散點圖 [36]、條形圖 [39]、直方圖，甚至串流資料 [35]。

[35] 指出，使用盲文點陣（Braille dots）的函式庫（如 plotille，plotext 也具備此能力）可以提供「非常好的解析度」，這比基本的 ASCII 區塊字元 [35] 提供了顯著更高的資訊密度 [34]。

3.3 代理工作流程與提示 (Agent Workflow & Prompt)

在此策略中，AI 代理不是生成圖表；它是生成「執行 plotext 的 Python 程式碼」。

提示範例 **(Prompt Example)**：

XML

<system_prompt>
您是一個 Python 程式碼生成器，專精於使用 'plotext' 函式庫 在終端機中繪圖。
您生成的程式碼必須是獨立的、可執行的。
</system_prompt>

<user_prompt>
使用 plotext [42]，為以下資料創建一個水平條形圖：
資料：
{
"regions":,
"sales": [1, 2, 3, 4]
}
</user_prompt>

<claude_response>
```python
import plotext as plt

try:
    regions =
    sales = [1, 2, 3, 4]

    plt.bar(regions, sales, orientation='horizontal')
    plt.title("Regional Sales")

    plt.show()

except Exception as e:
    print(f"Error generating plot: {e}")
```

(基於 36 的範例)
</claude_response>

3.4 TUI 作為「回退機制」的戰略價值

TUI 策略不應被視為「過時」或「低級」。在一個強大的 AI 代理架構中，它扮演著至關重要的「回退」角色。

一個設計精良的 AI 代理，其*首要*目標應該是提供高傳真度影像（策略二、三）。但是，高傳真度策略依賴於特定的終端機模擬器（如 iTerm2 [44]）。因此，代理在啟動時，應*首先*檢測使用者的環境（例如，檢查 `TERM_PROGRAM` 環境變數）。

如果檢測到不相容的終端機（例如一個標準的 `xterm` 或 `ssh` 會話），代理不應「失敗」，而應*自動*將其視覺化策略降級為「策略一」。TUI 確保您的代理在 100% 的情況下都能提供「視覺化」輸出，即使是在最受限的環境中。這實現了「彈性 UX」(Resilient UX)。

IV. 策略二 (高傳真度)：圖像編譯工作流程

這是在單一機器上實現高傳真度 UX 的「手動」或「開發者驅動」的工作流程。它為策略三（MCP）奠定了技術基礎。

4.1 概念：兩步驟的「生成 -> 顯示」

這是一個解耦的流程，模仿了開發者的典型工作：

1. **生成 (Generate):** 使用 AI 代理（或手動）創建一個 `chart.vl.json` 檔案。
2. **編譯 (Compile):** 在終端機中手動執行一個命令，將該 JSON 轉換為 PNG 影像。
3. **顯示 (Display):** 手動執行另一個命令，在終端機中內嵌顯示該 PNG。

4.2 核心工具鏈 (The Core Toolchain)

* **步驟 1 - 生成 (Claude 4.5):**
  使用 II.3 中詳述的提示策略，生成 Vega-Lite JSON 規格。

* **步驟 2 - 編譯 (vl-convert):**
  `vl-convert` 是 Vega-Lite 生態系統的「瑞士軍刀」。它是一個基於 Rust 的二進位檔案，封裝了 JavaScript V8 引擎和 Vega 函式庫，允許*無頭* (headless) 轉換。
  * `vl-convert vl2png -i chart.vl.json -o chart.png`
  * `vl-convert vl2svg -i chart.vl.json -o chart.svg` [25, 45]
  * `vl-convert vl2html -i chart.vl.json -o chart.html`

* **步驟 3 - 顯示 (imgcat / timg):**
  這是實現「終端機內」高傳真度 UX 的關鍵，但*並非*所有終端機都支援。
  * **iTerm2 (macOS):** 提供了 `imgcat` 工具，它使用一個專有協議 [44, 46] 來顯示內嵌影像。
  * **Kitty / WezTerm:** 提供了類似的 `icat` [47, 48] 或 `wezterm imgcat` [49, 50]。
  * **`timg` (推薦的抽象工具):** 介紹了 `timg`，這是一個使用者友好的檢視器，它會*自動*檢測終端機是支援 Sixel、Kitty 還是 iTerm2 圖形協議，並使用最佳方法。如果都不支援，它會回

退到 ASCII 。

完整工作流程 (CLI):

```bash
# 1. (AI 代理已創建 chart.vl.json)

# 2. 編譯
$ vl-convert vl2png -i chart.vl.json -o chart.png

# 3. 顯示
$ timg chart.png
```

4.3 評估：功能強大，但摩擦力大

- 優點：實現了 100% 的圖表傳真度。所有複雜的圖表（如地理空間圖）都可以完美渲染。
- 缺點：這是一個「手動」工作流程。使用者需要執行多個步驟，並且需要管理中間檔案 (chart.png)。這不符合您對「最佳 UX」所要求的「低摩擦」標準。

4.4 從「手動」到「自動」的橋樑

策略二本身不是最終解決方案，但它提供了實現真正最佳 UX（策略三）的所有必要「元件」。它證明了「JSON -> PNG -> 終端機顯示」的技術可行性 [45]。

其主要缺點是「手動操作」和「多步驟」。如果我們能將「編譯」(vl-convert) 和「顯示」(timg) 這兩個步驟，封裝到一個可以遠端調用的單一服務中，並使這個服務成為 AI 代理可以調用的「工具」，那麼我們就可以將這個多步驟的手動流程，轉化為一個對使用者來說「單一步驟」的自動化體驗。這正是「模型上下文協議 (MCP) 伺服器」所要解決的問題。

V. 策略三 (代理原生)：Vega-Lite MCP 伺服器工作流程

（核心推薦架構）

本章節詳述了「最佳 UX」的推薦實現。此策略將 AI 代理從一個「程式碼生成器」轉變為一個「工具

協調器」，利用 Claude 4.5 最新的「代理原生」能力 [11] 來提供一個真正無縫的視覺化體驗。

5.1 概念：將視覺化作為一種「工具服務」

在此架構中，AI 代理不再生成需要在本地執行的程式碼。相反，它執行一個「工具調用」(Tool Call)。

代理工作流程的轉變 [53]：

- 舊工作流程（策略一/二）：
  使用者 -> 代理 -> 生成程式碼 -> 使用者（複製/貼上/執行）-> 輸出
- 新工作流程（策略三）：
  使用者 -> 代理 -> 工具調用(JSON) -> MCP 伺服器 -> 影像成品 -> 代理（直接顯示） -> 輸出

如 [53] 所述，這「關閉了代理循環」。使用者無需離開對話，AI 代理就能端到端地完成「分析」和「展示」，這才是真正的「最佳 UX」。

5.2 什麼是模型上下文協議 (MCP)？

模型上下文協議 (MCP) 是一個「開放標準」[54]，旨在解決 LLM 與外部世界（工具、API、檔案）通訊的「混亂、臨機操作的局面」[55]。

MCP [55] 定義了一個三部分架構：

1. 主機 **(Host):** 您正在運行的 AI 應用程式（例如您的終端機代理）。
2. 客戶端 **(Client):** Host 內部的一個元件，它知道如何使用 JSON-RPC 2.0 [56] 格式發送標準化的 MCP 請求。
3. 伺服器 **(Server):** 一個獨立的（本地或遠端）進程，它接收 MCP 請求，執行「工具」，並回傳「成品」(Artifacts)。

MCP 不是一個函式庫，它是一個「協議」——就像 AI 的 HTTP。[58]、[79] 和 [80] 都強調了這一點：它是一個「統一的介面」，讓您的代理可以與任何 MCP 伺服器（檔案系統、資料庫、Playwright [57]、或我們的 Vega-Lite 伺服器）對話。

5.3 建立 Vega-Lite MCP 伺服器 (The mcp-vegalite-server)

[81]、[58]、[58] 和 [59] 提供了多個開源 Vega-Lite MCP 伺服器的範例（例如 Marko Mitranic 或 Isaac Wasserman 的專案）。這正是您需要的「工具」。

其工作原理 [58]：

1. 伺服器啟動並監聽（例如在本地的 stdio 或 TCP 埠上）[56]。
2. Claude 4.5 代理（作為 Host/Client）向伺服器發送一個 JSON-RPC 請求，其中包含一個 visualize 工具調用，其參數是完整的 Vega-Lite JSON 規格。
3. MCP 伺服器（在內部）執行我們在策略二中詳述的 vl-convert vl2png... [45] 命令。
4. 伺服器將生成的 chart.png 讀取為二進位資料。
5. 伺服器回傳一個 JSON-RPC 回應，其中包含一個「成品」(Artifact)，該成品是 PNG 影像的 Base64 編碼字串 [58]。

設定指南 [58]：

1. git clone https://github.com/markomitranic/mcp-vegalite-server.git [58]
2. cd mcp-vegalite-server
3. （使用 uv）uv --directory. run mcp_server_vegalite --output-type png [58]
4. 配置 **Claude** 代理：您的代理需要知道這個工具的存在。這通常是通過一個設定檔（如 claude_desktop_config.json [58] 或 VS Code 中的 mcp.json [54]）來完成的，您在其中聲明了 MCP 伺服器的位置和啟動命令。

5.4 代理工作流程的「終極版」(The "Ultimate" Agentic Workflow)

Claude 4.5 的新能力 [11] 允許我們建立一個「自我校驗」的視覺化流程。

1. 使用者提出請求：「顯示我的銷售趨勢。」
2. Claude 4.5 代理（利用 [11] 的「子代理協調」能力）開始規劃。
3. 步驟 **A**：代理生成 Vega-Lite JSON 規格。
4. 步驟 B：代理並行 (parallel execution) 52 調用兩個 MCP 伺服器：
   a. visualize_tool：調用 vega-lite-mcp-server 58，請求一個 PNG 成品。
   b. feedback_tool：調用 playwright-mcp-server 57，該伺服器將相同的 JSON 渲染在真實的無頭瀏覽器中，並回傳一張「螢幕截圖」。
5. 步驟 C (思考 [11])：代理現在擁有了兩個成品：一個原始 PNG 和一個「真實世界」的截圖。它使用其多模態（視覺）能力來比較這兩者，或分析截圖是否存在明顯的 UX 問題（例如，標籤重疊、文字太小）。
6. 步驟 D (迴圈 53)：

a. 如果 (if) 截圖看起來有問題，代理會啟動一個內部迴圈：「視覺回饋 57 顯示 Y 軸標籤重疊。我將修改 Vega-Lite 規格，增加 $axis: { "labelAngle": -45 }$ 並重試。」然後跳回步驟 A。
b. 如果 (else) 截圖看起來很好，代理才將已驗證的 PNG 影像（來自步驟 Ba）呈現給使用者。

這是一個強大的「閉環」系統 [53]，它利用 Claude 4.5 的協調 [11]、並行 [52] 和視覺回饋 [57] 能力，確保使用者第一次就能看到一個高品質、經過驗證的圖表。

VI. 策略四 (進階互動)：生成獨立式網頁神器

當終端機的限制真正成為束縛時，「最佳 UX」就是承認這些限制，並無縫地將使用者轉移到一個更強大的媒介：網頁瀏覽器。

6.1 概念：從「顯示」到「交付」

策略一到三專注於在終端機內「顯示」圖表。策略四專注於「交付」一個豐富的、可互動的、可分享的報告。

工作流程是：AI 代理生成一個 report.html 檔案 [23]，並在系統的預設瀏覽器中自動開啟它。

6.2 選項 A：獨立式互動儀表板

Claude 4.5 在生成「專業品質的文件、試算表和簡報」方面表現出色 [11]，這包括完整的 HTML/CSS/JS 儀表板 [23]。

提示工程：

1. 佈局 **(Layout):** 明確要求使用 CSS Grid [64] 或 Flexbox [66] 來創建響應式佈局。
2. 響應式 **(Responsive):** 要求「行動優先」(mobile-first) [67] 和 RWD 原則 [69]。
3. 獨立 **(Standalone):** 關鍵指令：「生成一個單一的、獨立的 *HTML* 檔案。所有 CSS 和 JavaScript 必須是內聯的。所有資料必須作為 JSON 變數嵌入。該檔案不應有任何外部依賴。」

4. 技術棧：要求使用 vegaEmbed [24] 並將 Vega-Lite 規格作為 JS 變數嵌入。

6.3 選項 B：「最佳 UX」的極致：AI 驅動的 Scrollytelling 報告

這是「最佳 UX」的旗艦級實現。與其給使用者一個儀表板讓他們自己去「探索」，AI 代理不如主動引導他們完成一個「資料故事」。

什麼是 Scrollytelling？
這是一種流行的數位敘事格式 71，當使用者滾動瀏覽敘述性文本時，旁邊（或背景）的視覺化會動態更新 72。
技術棧：Scrollama.js + vegaEmbed
Scrollama.js 是一個輕量級的 JavaScript 函式庫，用於偵測使用者滾動到某個「步驟」(step) 71。它的核心事件是 scroller.onStepEnter 71。

6.4 代理的 Scrollytelling 工作流程

Claude 4.5 [52] 首次使「全自動 Scrollytelling 生成」成為可能。

1. 步驟 **1 -** 敘事規劃 **(NoT)**：使用者提供一個複雜的資料集。代理首先使用「敘事思維」(Narrative-of-Thought, NoT) [77] 來分析資料，找出一個有意義的「故事線」（例如，「總銷售額在增長，但深入挖掘後，我們發現增長僅來自北方地區，而南方地區正在萎縮。」）[78]。
2. 步驟 **2 -** 結構生成：代理將这个故事線分解為 3-5 個「步驟」。
3. 步驟 **3 -** 視覺化狀態生成：對於每一個敘事步驟，代理生成一個相應的 Vega-Lite JSON 規格，以視覺方式僅突出該步驟的觀點。
4. 步驟 4 - HTML 構建：代理生成一個單一的 HTML 檔案，其中包含：
   a. HTML 結構（一個 scroll__graphic 區域和多個 scroll__text.step 元素）71。
   b. Scrollama.js 和 vegaEmbed 的 CDN 連結 24。
   c. 一個 JavaScript 陣列 chartStates，其中包含所有步驟的 Vega-Lite JSON。
   d. 一個 handleStepEnter 回調函數，其核心邏輯如下：

```
JavaScript
// (由 Claude 4.5 生成的程式碼)

// 步驟 3 中由 AI 生成的、包含所有圖表狀態的陣列
const chartStates =;

const scroller = scrollama(); // 初始化 [71]
```

```
function handleStepEnter(response) {
 // response.index 是當前步驟的索引 [75]
 const stepIndex = response.index;

 // 從我們預先生成的陣列中獲取對應的圖表規格
 const vlSpec = chartStates[stepIndex];

 // 使用 vegaEmbed 重新渲染圖表
 // 確保圖表容器的 ID 是 '#vis'
 if (vlSpec) {
   vegaEmbed('#vis', vlSpec, { "actions": false, "theme": "dark" });
 }
}

// 初始化 Scrollama [71]
scroller.setup({
 container: '#scroll',
 graphic: '.scroll__graphic',
 text: '.scroll__text',
 step: '.scroll__text.step',
 offset: 0.5, // 觸發器位於視窗中點 [71]
 debug: false
})

.onStepEnter(handleStepEnter); // 綁定事件 [75]

// 處理視窗大小調整
window.addEventListener('resize', scroller.resize);
```

5. 結論：代理交付的不僅是一個圖表，而是一個完整的、客製化的分析簡報。這將 AI 代理從一個「分析師」提升為一個「首席溝通者」。

VII. 綜合比較與最終建議

本報告分析了四種從「低傳真度、高可攜性」到「高傳真度、高互動性」的架構。您的最終選擇取決於您的代理需要平衡的具體情境、成本和 UX 目標。

7.1 終端機 AI 代理視覺化架構決策矩陣

此表格將前面六章的詳細分析濃縮為一個高層次的決策工具，使您能夠根據您的具體需求（UX 傳真度、實施複雜性、可攜性、代理整合度、互動性）快速權衡四種策略的利弊。

終端機 **AI** 代理視覺化架構決策矩陣

策略 **(Strategy)**	**UX** 傳真度 **(UX Fidelity)**	實施複雜性 **(Complexity)**	跨平台可攜性 **(Portability)**	代理整合度 **(Agent Integration)**	互動潛力 **(Interactivity)**
策略一：**TUI** 內建 (plotext) [36]	低 (ASCII/Braille)	低 (單一 Python 函式庫)	極高 (適用於任何終端機)	低 (純程式碼生成)	無
策略二：圖像編譯 (vl-convert + timg) [45]	高 (PNG 影像)	中 (需安裝多個 CLI 工具)	低 (僅限 iTerm2/Kitty/WezTerm)	低 (手動多步驟流程)	無
策略三：**MCP** 伺服器 (mcp-vegalite-server) [58]	高 (PNG 影像)	高 (需設定伺服器進程)	低 (繼承策略二的終端機限制)	極高 (原生工具使用)	有限 (透過參數)

策略四：網頁神器 (Scrollama.js) [23, 71]	極高 (全互動式 HTML)	極高 (需生成完整網頁應用)	極高 (適用於任何瀏覽器)	高 (代理作為「報告生成器」)	無限

7.2 首席架構師的最終建議

1. 立即實施「策略一」作為彈性回退 (Resilient Fallback)。
   您的代理必須立即整合 plotext 36。在您的代理啟動腳本中，加入一個終端機能力檢測。如果環境不支援圖形協議，自動切換到此策略。這確保了 100% 的可靠性 51。
2. 積極投資「策略三」作為核心工作流程 (Core Workflow)。
   這是您尋求的「最佳 UX」的真正答案。它在「傳真度」（高）和「摩擦力」（零）之間取得了完美平衡。使用 Claude 4.5 11 卻不使用其「工具使用」和「代理協調」能力，是一種戰略浪費。策略三是唯一一個充分利用了您 AI 模型全部潛力的架構。開始建構並運行 mcp-vegalite-server 58，並採用 5.4 中詳述的「自我校驗」工作流程 57，以確保交付給使用者的所有圖表都已通過 AI 的品質保證。
3. 保留「策略四」作為旗艦級「交付」功能 (Premium Deliverable)。
   當使用者的請求不僅僅是「查看資料」，而是「為我建立一份關於...的報告」時，您的代理應觸發策略四。專注於 6.4 中的 Scrollytelling 工作流程 24。讓 Claude 4.5 扮演「資料敘事者」的角色，為您生成一個完整的、由 AI 驅動的、可直接呈報給執行長的互動式網頁報告。

總結：您的終端機代理不應只有一種視覺化模式。它應該是一個「分層的」系統：策略一確保「普遍可用性」，策略三提供「即時的最佳 UX」，而策略四則作為「最終的、可交付的分析產品」。

Works cited

1. Chart Junk Removal: Maximizing Data-Ink Ratio - Dev3lop, accessed November 5, 2025, https://dev3lop.com/chart-junk-removal-maximizing-data-ink-ratio/
2. Data Visualization Design, Part 4: Removing Chart Junk | by Data Lass | Medium, accessed November 5, 2025, https://medium.com/@LauraHKahn/data-visualization-design-part-4-removing-chart-junk-28b3bdd0faa1
3. 7 Best Practices of Data Visualization | by Tharini Iddamalgoda | Medium, accessed November 5, 2025, https://medium.com/@tharini.iddamalgoda/7-best-practices-of-data-visualization-6f95ef610713
4. Best Practices for Creating Charts with Vega and Vega-Lite - TurboLine.ai, accessed November 5, 2025, https://turboline.ai/best-practices-for-creating-charts-with-vega-and-vega-lite/
5. Overview - Vega-Lite, accessed November 5, 2025,

https://vega.github.io/vega-lite/docs/
6. A Quick review of LLM for Data Visualization - vizGPT, accessed November 5, 2025, https://vizgpt.ai/docs/blog/llm-for-viz
7. Prompting best practices - Claude Docs, accessed November 5, 2025, https://docs.claude.com/en/docs/build-with-claude/prompt-engineering/claude-4-best-practices
8. Data Visualization: Why 3D charts are a terrible idea | by Praveen Purohit | Medium, accessed November 5, 2025, https://medium.com/@purohitpraveen/data-vizualization-why-3d-charts-are-a-terrible-idea-32657fbb928e
9. Data Visualization: Best Practices That Drive Better Business Decisions - CodingCops, accessed November 5, 2025, https://codingcops.com/data-visualization/
10. death to pie charts - storytelling with data, accessed November 5, 2025, https://www.storytellingwithdata.com/blog/2011/07/death-to-pie-charts
11. The "ugly first draft" method completely changed how I approach dashboards - Reddit, accessed November 5, 2025, https://www.reddit.com/r/datavisualization/comments/1msfc7f/the_ugly_first_draft_method_completely_changed/
12. How to Design Accessible Dark Mode Interfaces - Medium, accessed November 5, 2025, https://medium.com/@tundehercules/designing-effective-dark-mode-interfaces-17f38ecea2e9
13. 10 Dark Mode UI Best Practices & Principles for 2025, accessed November 5, 2025, https://www.designstudiouiux.com/blog/dark-mode-ui-design-best-practices/
14. How Can You Configure Your Vega-Lite Charts? - vizGPT, accessed November 5, 2025, https://vizgpt.ai/docs/vega-lite/config
15. [Proposal] Consolidate chart-rendering to Vega-Lite #2385 - GitHub, accessed November 5, 2025, https://github.com/opensearch-project/OpenSearch-Dashboards/issues/2385
16. Ask HN: What's the best charting library for customer-facing dashboards? - Hacker News, accessed November 5, 2025, https://news.ycombinator.com/item?id=40196880
17. Using LLM to Generate Data for D3.js Force Directed Graph (FDG) - Medium, accessed November 5, 2025, https://medium.com/@junjunzaragosa2309/using-llm-to-generate-data-for-d3-js-force-directed-graph-c490382d1172
18. How to Create Custom Interaction Mode for Tooltip in Chart JS 4 - YouTube, accessed November 5, 2025, https://www.youtube.com/watch?v=VvrdL6BfkYg
19. Create Dashboards in Seconds With AI (It's Mind-Blowing) - YouTube, accessed November 5, 2025, https://www.youtube.com/watch?v=VIFyXj31iYE
20. Introduction to Vega-Lite, accessed November 5, 2025, https://vega.github.io/vega-lite/tutorials/getting_started.html
21. Compiling Vega-Lite to Vega, accessed November 5, 2025,

https://vega.github.io/vega-lite/usage/compile.html
22. Getting the most out of Sonnet 4.5 in Claude.ai, accessed November 5, 2025, https://support.claude.com/en/articles/12439373-getting-the-most-out-of-sonnet-4-5-in-claude-ai
23. Top 30 Claude Sonnet 4.5 Prompts for Coding, Reasoning & Long Tasks - Sider, accessed November 5, 2025, https://sider.ai/blog/ai-tools/top-30-claude-sonnet-4_5-prompts-for-coding-reasoning-long-tasks
24. Anyone using JSON Prompting with LLMs? : r/GithubCopilot - Reddit, accessed November 5, 2025, https://www.reddit.com/r/GithubCopilot/comments/1mb7lpn/anyone_using_json_prompting_with_llms/
25. Effectively Use JSON in Generative AI LLM Prompts - YouTube, accessed November 5, 2025, https://www.youtube.com/watch?v=j3VuroS5M04
26. Create an interactive custom tooltip for chartjs - Stack Overflow, accessed November 5, 2025, https://stackoverflow.com/questions/56969606/create-an-interactive-custom-tooltip-for-chartjs
27. Why I'm backing Vega-Lite as our default tool for data visualisation - Robin Linacre, accessed November 5, 2025, https://robinlinacre.medium.com/why-im-backing-vega-lite-as-our-default-tool-for-data-visualisation-51c20970df39
28. hyungkwonko/chart-llm: Vega-Lite Chart Dataset and NL Generation Framework using LLMs, accessed November 5, 2025, https://github.com/hyungkwonko/chart-llm
29. Natural Language Dataset Generation Framework for Visualizations Powered by Large Language Models - YouTube, accessed November 5, 2025, https://www.youtube.com/watch?v=OTbnkT_T8FA
30. smar10s/pytui: Python TUI library - GitHub, accessed November 5, 2025, https://github.com/smar10s/pytui
31. How to Create Stunning Graphs in the Terminal with Python | by Sourav De - Medium, accessed November 5, 2025, https://medium.com/@SrvZ/how-to-create-stunning-graphs-in-the-terminal-with-python-2adf9d012131
32. python - How to plot a chart in the terminal - Stack Overflow, accessed November 5, 2025, https://stackoverflow.com/questions/37288421/how-to-plot-a-chart-in-the-terminal
33. How to Plot in the Terminal with Python and Textualize, accessed November 5, 2025, https://www.blog.pythonlibrary.org/2024/08/19/how-to-plot-in-the-terminal-with-python-and-textualize/
34. Plotext: Plotting in the Terminal - Python Snacks, accessed November 5, 2025, https://www.pythonsnacks.com/p/plotext-terminal-plotting
35. Tuesday Tooling: Visualise Data in the Terminal - bigl.es, accessed November 5,

2025, https://bigl.es/tuesday-tooling-plotext/
36. plotext 2.3.1 - PyPI, accessed November 5, 2025, https://pypi.org/project/plotext/2.3.1/
37. plotext 3.0.1 - PyPI, accessed November 5, 2025, https://pypi.org/project/plotext/3.0.1/
38. Making Plots In Your Terminal With Plotext - Pybites, accessed November 5, 2025, https://pybit.es/articles/terminal-plotting-with-plotext/
39. piccolomo/plotext: plotting on terminal - GitHub, accessed November 5, 2025, https://github.com/piccolomo/plotext
40. vega/vl-convert: Utilities for converting Vega-Lite specs from the command line and Python, accessed November 5, 2025, https://github.com/vega/vl-convert
41. hzeller/timg: A terminal image and video viewer. - GitHub, accessed November 5, 2025, https://github.com/hzeller/timg
42. Harnessing Claude Sonnet 4.5 for Smarter Coding Workflows and AI Agent Development, accessed November 5, 2025, https://www.cloudthat.com/resources/blog/harnessing-claude-sonnet-4-5-for-smarter-coding-workflows-and-ai-agent-development
43. My favorite MCP use case: closing the agentic loop : r/ClaudeAI - Reddit, accessed November 5, 2025, https://www.reddit.com/r/ClaudeAI/comments/1mwjbdo/my_favorite_mcp_use_case_closing_the_agentic_loop/
44. Use MCP servers in VS Code, accessed November 5, 2025, https://code.visualstudio.com/docs/copilot/customization/mcp-servers
45. Model Context Protocol (MCP): A Beginner's Guide | by Alaa Dania ..., accessed November 5, 2025, https://medium.com/infinitgraph/model-context-protocol-mcp-a-beginners-guide-d7977b52570a
46. Build an MCP server - Model Context Protocol, accessed November 5, 2025, https://modelcontextprotocol.io/docs/develop/build-server
47. Building agents with the Claude Agent SDK - Anthropic, accessed November 5, 2025, https://www.anthropic.com/engineering/building-agents-with-the-claude-agent-sdk
48. The Ultimate Guide to the Vega-Lite MCP Server: Turn Your AI into a Data Visualization Pro, accessed November 5, 2025, https://skywork.ai/skypage/en/vega-lite-mcp-server-data-visualization/1980471103827070976
49. mcp-vegalite-server - MCP Server Registry - Augment Code, accessed November 5, 2025, https://www.augmentcode.com/mcp/mcp-vegalite-server
50. Vega-Lite Data Visualization MCP server for AI agents - Playbooks, accessed November 5, 2025, https://playbooks.com/mcp/markomitranic-vegalite
51. Generate Dashboards with AI in seconds - Prototypr.ai, accessed November 5, 2025, https://www.prototypr.ai/dashboards
52. Creating AI-Powered Interactive Dashboards - Coconote, accessed November 5, 2025, https://coconote.app/notes/d17fe717-5129-4474-a8af-06da5e1718c0

53. LLM Tools for Design and Visual Content Creation: Tips, Best Practices & Prompt Library, accessed November 5, 2025, https://www.kalungi.com/blog/syntropy/llm-tools-for-design-and-visual-content-creation-tips-best-practices-prompt-library
54. AI Build a Responsive Grid Layout with CSS Grid Prompts | Taskade, accessed November 5, 2025, https://www.taskade.com/prompts/coding/build-a-responsive-grid-layout-with-css-grid
55. Using LLMs to generate UX Wireframes - Sony Interactive Entertainment, accessed November 5, 2025, https://sonyinteractive.com/en/news/blog/using-llms-to-generate-ux-wireframes/
56. The Ultimate CSS Grid Layout for Dashboards - YouTube, accessed November 5, 2025, https://www.youtube.com/watch?v=055AgeHNkHU
57. How to write effective prompts for AI-powered UI design - Updivision, accessed November 5, 2025, https://updivision.com/blog/post/how-to-write-effective-prompts-for-ai-powered-ui-design
58. From Code to Creation: AI Prompts Every Front-End Developer Should Know in 2025, accessed November 5, 2025, https://bluetickconsultants.medium.com/from-code-to-creation-ai-prompts-every-front-end-developer-should-know-in-2025-1c13ebba9d62
59. Responsive web design - Learn web development - MDN Web Docs - Mozilla, accessed November 5, 2025, https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Responsive_Design
60. Usage | Vega, accessed November 5, 2025, https://vega.github.io/vega/usage/
61. An Introduction to Scrollama.js - The Pudding, accessed November 5, 2025, https://pudding.cool/process/introducing-scrollama/
62. Learning Scrollama for Interactive Data Visualizations | by Michela Tjan Sakti Effendie, accessed November 5, 2025, https://medium.com/@tjanmichela/learning-scrollama-for-interactive-data-visualizations-66bc11b2179b
63. Scrollytelling demo using scrollama.js and d3.js - GitHub, accessed November 5, 2025, https://github.com/edriessen/scrollytelling-scrollama-d3-demo
64. An introduction to scrollytelling: data storytelling using scrollama.js, d3.js and html/css, accessed November 5, 2025, https://www.edriessen.com/2023/04/24/an-introduction-to-scrollytelling-data-storytelling-using-scrollama-js-d3-js-and-html-css/
65. russellsamora/scrollama: Scrollytelling with IntersectionObserver. - GitHub, accessed November 5, 2025, https://github.com/russellsamora/scrollama
66. Scrollama, changing charts - Stack Overflow, accessed November 5, 2025, https://stackoverflow.com/questions/54243895/scrollama-changing-charts
67. Narrative-of-Thought (NoT) - Learn Prompting, accessed November 5, 2025, https://learnprompting.org/docs/new_techniques/narrative_of_thought
68. Prompt Engineering For storytelling: From Chaos to Characters Building | by

Karthikeya Suppa | Medium, accessed November 5, 2025, https://medium.com/@karthikeyasuppa01/prompt-engineering-for-storytelling-from-chaos-to-characters-building-6550cd35ee7d
69. Build AI's Future: Model Context Protocol (MCP) with Spring AI in Minutes, accessed November 5, 2025, https://www.youtube.com/watch?v=MarSC2dFA9g
70. Build Agents using Model Context Protocol on Azure | Microsoft Learn, accessed November 5, 2025, https://learn.microsoft.com/en-us/azure/developer/ai/intro-agents-mcp
71. Unlocking AI-Driven Visuals: A Deep Dive into the Vega-Lite Data Visualization MCP Server, accessed November 5, 2025, https://skywork.ai/skypage/en/ai-visualization-vega-lite/1980509238833303552