2025 台灣棒球數據分析競賽

美國職棒大聯盟
最終排名預測模型之建構

Presented by NCKU STAT

Team：藍色球隊，藍到顯著

Members：林宇軒、洪毅荃、郭懿蓼

特別感謝

Advisor：張聖岳🏉、家任

Best Friend： 張芮其、余振瑋

Contents

- Motivation & Methods
- Data Content & Data Source
- **Colley's** Bias Free **Ranking Method**
- Extension of the **Elo rating system** to MOV(margin of victory)
- **Beta Regression** – Result1, 2
- Comparison
- Conclusion, Suggestion and Future Work
- Reference & Our Github

Motivation (Background)

Question ? Why use Comprehensive Strength

在棒球比賽中，僅依靠打擊指標與防守指標往往難以準確的預測比賽結果，這可能源於棒球比賽本身具有較高的隨機性，也可能是因為單一面向的數據僅反映影響勝負的部分因素，代表性不足。

洛杉磯道奇 Follow team

1th 國聯 West | 93-69

打擊率	得分	防禦率	WHIP
5th	2th	16th	14th
.253 AVG	825 R	3.95 ERA	1.26 WHIP

多倫多藍鳥 Follow team

1th 美聯 East | 94-68

打擊率	得分	防禦率	WHIP
1th	4th	19th	16th
.265 AVG	798 R	4.19 ERA	1.27 WHIP

克里夫蘭守護者 Follow team

1th 美聯 Central | 88-74

打擊率	得分	防禦率	WHIP
29th	28th	4th	14th
.226 AVG	643 R	3.70 ERA	1.26 WHIP

德州遊騎兵 Follow team

3th 美聯 West | 81-81

打擊率	得分	防禦率	WHIP
26th	22th	1th	1th
.234 AVG	684 R	3.47 ERA	1.18 WHIP

舉例來說：

有些球隊打擊強、投手弱；有些球隊防守佳、得分卻不穩定，使得傳統的勝率或單指標難以衡量球隊的整體實力。

基於上述可能原因，本研究採用「**綜合實力指標（ Comprehensive Strength ）**」去預測賽季最終的球隊排名，以取得較全面且穩定的預測結果。

Motivation and Methods

本研究的主要動機在於建立一套能夠有效**預測美國職棒大聯盟（MLB）2025 年球季整體排名**的模型。**不同於僅以球隊勝率爲依據的傳統預測方式**，本研究聚焦於**最終整體排名的預測表現**，探討在不同綜合實力指標下的預測能力與穩定性。

為了衡量球隊的綜合實力，本研究採用三種方法進行分析：

1. **WIN Rate**
2. **Colley’s Rating Method**
3. **Extension of the Elo Rating System to Margin of Victory（MOV-Elo）**

綜上所述，本研究的核心在於：與其依賴單一面向的球隊數據，不如改採更全面的「**綜合實力指標**」來預測整季的最終排名。因此，本研究比較 Win Rate（勝率）、Colley Rating 與 MOV-Elo 三種方法，檢驗不同方法在整體排名預測上的表現與穩定性。

Methods

- Comparison of Comprehensive Strength

三種方法的主要差異為：

- **Win Rate**：
  - **不考慮賽程強度，也不考慮勝分差**。
  - 只反映「結果」，但無法反映球隊在不同強度賽程下的真實競爭力。
- **Colley's Rating** ：
  - **考慮賽程強度，但不考慮勝分差**，
  - 僅依據勝敗紀錄建立評分矩陣，使評分更平滑、較不受單場極端結果影響，提供**更穩定**、**無偏**的隊伍排名。
- **MOV-Elo**：
  - **考慮賽程強度與勝分差**。
  - 在傳統 Elo 模型中加入「勝分差（Margin Of Victory, MOV）加權函數」，使評分更新能反映比賽結果的差距，進而更細緻地捕捉球隊間的實際競爭強度。

Data – Data Content

本研究之資料取自**美國職棒大聯盟（MLB）2025 年賽季**之比賽紀錄。

以比賽為單位（Game-based data），每筆紀錄對應一場比賽，並包含以下變數：

- **日期（Date）**：比賽進行之日期與星期。
- **主場 / 客場（Home/Away）**：比賽場地資訊。
- **隊伍（Team）**：主場球隊。
- **對手（Opponent）**：客場球隊。
- **狀態（Result）**：主場球隊之勝（Win）或敗（Loss）。
- **得分（Score）**：包含「大」與「小」兩隊之分數。

日期	主場/客場	對手	狀態	大	小	隊伍
03/18（週二）	@	小熊	勝	4	1	道奇
03/19（週三）	@	小熊	勝	6	3	道奇
03/28（週五）	vs	老虎	勝	5	4	道奇
03/29（週六）	vs	老虎	勝	8	5	道奇
03/30（週日）	vs	老虎	勝	7	3	道奇
04/01（週二）	vs	勇士	勝	6	1	道奇
04/02（週三）	vs	勇士	勝	3	1	道奇
04/03（週四）	vs	勇士	勝	6	5	道奇
04/05（週六）	@	費城人	敗	3	2	道奇
04/06（週日）	@	費城人	勝	3	1	道奇
04/07（週一）	@	費城人	敗	8	7	道奇
04/08（週二）	@	國民	敗	6	4	道奇
04/09（週三）	@	國民	敗	8	2	道奇
04/10（週四）	@	國民	勝	6	5	道奇
04/12（週六）	vs	小熊	勝	3	0	道奇

Data – Data Source

資料以**自動化爬蟲**方式收集，內容涵蓋每場比賽的基本資訊。（以 2025 年為例）

```
# -*- coding: utf-8 -*-
import os
import time
import pandas as pd
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager

# --- 設定 ---
teams = [
    "光芒", "藍鳥", "金鷲", "紅襪", "洋基",
    "守護者", "皇家", "老虎", "雙城", "白襪",
    "太空人", "運動家", "遊騎兵", "天使", "水手",
    "勇士", "大都會", "國民", "費城人", "馬林魚",
    "海盜", "釀酒人", "紅雀", "小熊", "紅人",
    "響尾蛇", "道奇", "落磯", "教士", "巨人"
]

base_url = "https://tw.sports.yahoo.com/mlb/teams/{team}/schedule/?season=2025&scheduleType=list"
output_file = r"您的電腦位置"

# --- 初始化 Chrome ---
options = Options()
options.add_argument("--headless")
options.add_argument("--disable-gpu")
options.add_argument("--no-sandbox")

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)

all_data = []
```

```
for team in teams:
    url = base_url.format(team=team)
    print(f"抓取 {team} 賽程中... {url}")
    driver.get(url)
    time.sleep(1)

    try:
        rows = driver.find_elements(By.CSS_SELECTOR, "table tbody tr")

        for row in rows:
            text = row.text.strip()
            if not text:
                continue

            parts = text.split()
            if len(parts) < 5:
                continue

            date = parts[0]
            home_away = parts[1]
            opponent = parts[2]
            result = parts[3]
            score = " ".join(parts[4:])

            all_data.append({
                "球隊": team,
                "日期": date,
                "主/客": home_away,
                "對手": opponent,
                "勝敗": result,
                "比分": score
            })

    except Exception as e:
        print(f"{team} 抓取失敗：{e}")
        continue

driver.quit()

df = pd.DataFrame(all_data)
print(f"共抓取 {len(df)} 場比賽資料")

if not df.empty:
    if not os.path.exists(output_file):
        with pd.ExcelWriter(output_file, engine="openpyxl", mode="w") as writer:
            df.to_excel(writer, index=False, sheet_name="工作表1")
        print(f"已建立新檔案 {output_file}")
    else:
        from openpyxl import load_workbook
        book = load_workbook(output_file)
        if "工作表1" not in book.sheetnames:
            with pd.ExcelWriter(output_file, engine="openpyxl", mode="a") as writer:
                df.to_excel(writer, index=False, sheet_name="工作表1")
        else:
            with pd.ExcelWriter(output_file, engine="openpyxl", mode="a", if_sheet_exists="overlay") as writer:
                startrow = writer.sheets["工作表1"].max_row
                df.to_excel(writer, index=False, sheet_name="工作表1", header=False, startrow=startrow)
        print(f"已成功寫入 {output_file}")
else:
    print("沒有抓到任何資料！")
```

Colley's Bias Free Ranking Method

STEP1 — **建立評分方法/公式**

Laplace's Method

當賽季剛開始，還沒有任何數據時，最合理的猜測是評分（勝率）落在任一處的機率相等，即採用「$\mathbf{U(0,1)}$」的 uniform Bayesian prior，$\pi(\hat{r}) = 1$，表示「任何勝率都有相同的可能性」，其期望值為 0.5，也就是在沒有任何比賽資料時，**每支球隊的預期勝率（expected win probability）被設定 0.5**，代表各隊實力在賽季初被視為勢均力敵。

⚾ 舉例來說：

觀察一場比賽後，結果為1勝0負。

- 將一場比賽視為 Bernoulli 試驗，這筆資料出現的機率為：

$$L(\hat{r}) = P(1\ \text{win} \mid \hat{r}) = \hat{r}^1(1-\hat{r})^0 = \hat{r}$$

- 套用至貝氏更新，得到後驗機率為：

$$f(\hat{r} \mid 1\ \text{win}) \propto L(\hat{r})\pi(\hat{r}) = \hat{r} \times 1 = \hat{r}$$

- 正規化後得 $f(\hat{r}) = 2\hat{r}$，則期望值：

$$E[\hat{r}] = \frac{2}{3}$$

根據此例，可以發現在一般情況下，當觀測為 w 勝 ℓ 負時，後驗機率為：

$$\boldsymbol{f(\hat{r}) \propto \hat{r}^w(1-\hat{r})^l}$$

即 $\boldsymbol{Beta(w+1, \ell+1)}$，則期望值為：

$$\boldsymbol{E[\hat{r}] = \frac{w+1}{w+\ell+2} = \frac{1+n_{w,i}}{2+n_{tot,i}}}$$

Colley's Bias Free Ranking Method

STEP1 建立評分方法/公式

STEP2 **校正賽程強度**

理想情況：假設對手都是平均球隊

首先，把勝場拆成兩部分，前項是「淨勝場的一半」，後項是「把每一場都當成對上**平均球隊**，平均貢獻 $\frac{1}{2}$ 勝」，可得下式：

$$n_{w,i} = \frac{n_{w,i} - n_{l,i}}{2} + \frac{n_{tol,i}}{2} \; where \; \frac{n_{tol,i}}{2} = \sum_{j=1}^{n_{tol,i}} \frac{1}{2}$$

實際情況：對手強弱不同

實際上，對手的實力並**不平均**，所以把每一場的 $\frac{1}{2}$ 換成「**該場對手的評分 r_j^i**」，接著，把上式改為有效勝場：

$$n_{w,i}^{eff} = \frac{n_{w,i} - n_{l,i}}{2} + \sum_{j=1}^{n_{tot,i}} r_j^i$$

其中，

- r_j^i：球隊 i 的第 j 個對手的評分。

Colley's Bias Free Ranking Method

迭代法

現實比賽中，因為對手並非隨機配對，且彼此可能面臨共同對手，球隊評分彼此相依。
因此，需透過多次修正才能收斂至穩定的結果。

STEP1

建立評分方法/公式

根據 Laplace's Method，假設所有對手初始評分（勝率）均為：$r = \frac{1}{2}$

STEP2

校正賽程強度

利用修正後的勝場公式，依「對手評分」計算各隊的賽程強度修正：$n_{w,i}^{eff} = \frac{n_{w,i} - n_{l,i}}{2} + \sum_{j=1}^{n_{tot,i}} r_j^i$

STEP3

重新計算評分

根據新的有效勝場數，更新球隊評分：$r_i = \frac{1 + n_{w,i}^{eff}}{2 + n_{tot,i}}$

此為 Colley 方法的核心評分公式，
會同時反映勝負與賽程強度

STEP4

Rating 計算方法 – 迭代法

重覆 STEP2–3。

若每一步的變動愈來愈小，直到所有球隊的變動小到可忽略（收斂），此時的評分即為最終解。

由於迭代法在計算過程中較為繁瑣且效率有限，
因此本研究改以 **Colley 矩陣法（Colley Matrix Method）** 直接求解，以獲得更穩定且高效的結果。

Colley's Bias Free Ranking Method

Colley 矩陣法（Colley Matrix Method）

根據 STEP1 的 Laplace's Method 以及 STEP2 的賽程強度修正，可得每隊的一條線性方程式：

$$(2 + n_{tot,i})r_i - \sum_{j=1}^{n_{tot,i}} r_j^i = 1 + \frac{n_{w,i} - n_{l,i}}{2}$$

將所有球隊的方程式整合後，可用矩陣形式表示為：

$$\boldsymbol{C\vec{r} = \vec{b}}$$

其中：

- N：球隊總數
- $C \in [c_{ij}]_{N\text{x}N}$ where $c_{ii} = 2 + n_{tot,i}$ and $c_{ij} = -n_{ij}(i \neq j)$
- $b_i = 1 + \frac{n_{w,i} - n_{l,i}}{2}$

此矩陣 C 即為 Colley Matrix。

求解聯立方程組 $Cr = b$ 所得的向量 r，即為各隊的最終評分（Ranking Scores）。

Colley's Bias Free Ranking Method

Colley 矩陣法（Colley Matrix Method）

目標

證明 Colley 矩陣 C 是**對稱正定矩陣**（Symmetric Positive Definite），即可保證聯立方程式：

$$C\vec{r} = \vec{b}$$

存在且僅有一個**唯一解** (Unique Solution)。

想法

為了證明 Colley 矩陣 C 是對稱正定，我們需要確認：矩陣 C 對任何非零向量 $\vec{v}$，皆滿足：

$$\vec{v}^{\mathsf{T}}(C\vec{v}) > 0$$

Colley Matrix

根據矩陣 $\boldsymbol{C}$ 的結構可寫成：

$$C = 2I + \sum_{k \in \text{all game}} G^{(k)}$$

其中每一場比賽 k（隊伍 i 對隊伍 j）會貢獻一個 $G^{(k)}$：

- 在 i,i與 j,j位置各 +1；
- 在 i,j與 j,i位置各 −1；
- 其餘元素皆為 0。

$$G^{(k)} = \begin{bmatrix} & \vdots & & \vdots & \\ \cdots & 1 & \cdots & -1 & \cdots \\ & \vdots & & \vdots & \\ \cdots & -1 & \cdots & 1 & \cdots \\ & \vdots & & \vdots & \end{bmatrix}$$

故每個 $G^{(k)}$的**每列總和為 0**，且對稱。

Colley's Bias Free Ranking Method

Colley 矩陣法（Colley Matrix Method）

證明

已知：

$$C = 2I + \sum_k G^k$$

根據矩陣乘法的分配律，可得：

$$\vec{v}^{\mathsf{T}}(C\vec{v}) = \underbrace{\vec{v}^{\mathsf{T}}(2I\vec{v})}_{2\,\|\,\vec{v}\,\|^2 > 0} + \sum_k \vec{v}^{\mathsf{T}}\left(G^k\vec{v}\right) > 0, \forall \text{非零向量} \vec{v}$$

單場比賽的貢獻 $G^{(k)}$：

若第 k場是隊伍 i 對隊伍 j，在計算 $\vec{v}' = G^k\vec{v}$ 後，僅得兩個非零分量：

$$v_i' = v_i - v_j \text{ and } v_j' = v_j - v_i$$

因此：

$$\vec{v}^{\mathsf{T}}\left(G^k\vec{v}\right) = v_i(v_i - v_j) + v_j(v_j - v_i) = (v_i - v_j)^2 \geq 0.$$

故 C正定（對稱且正定 ⇒ 可逆，Cholesky 成立）

結論

C 為對稱正定矩陣 (SPD)，則可保證

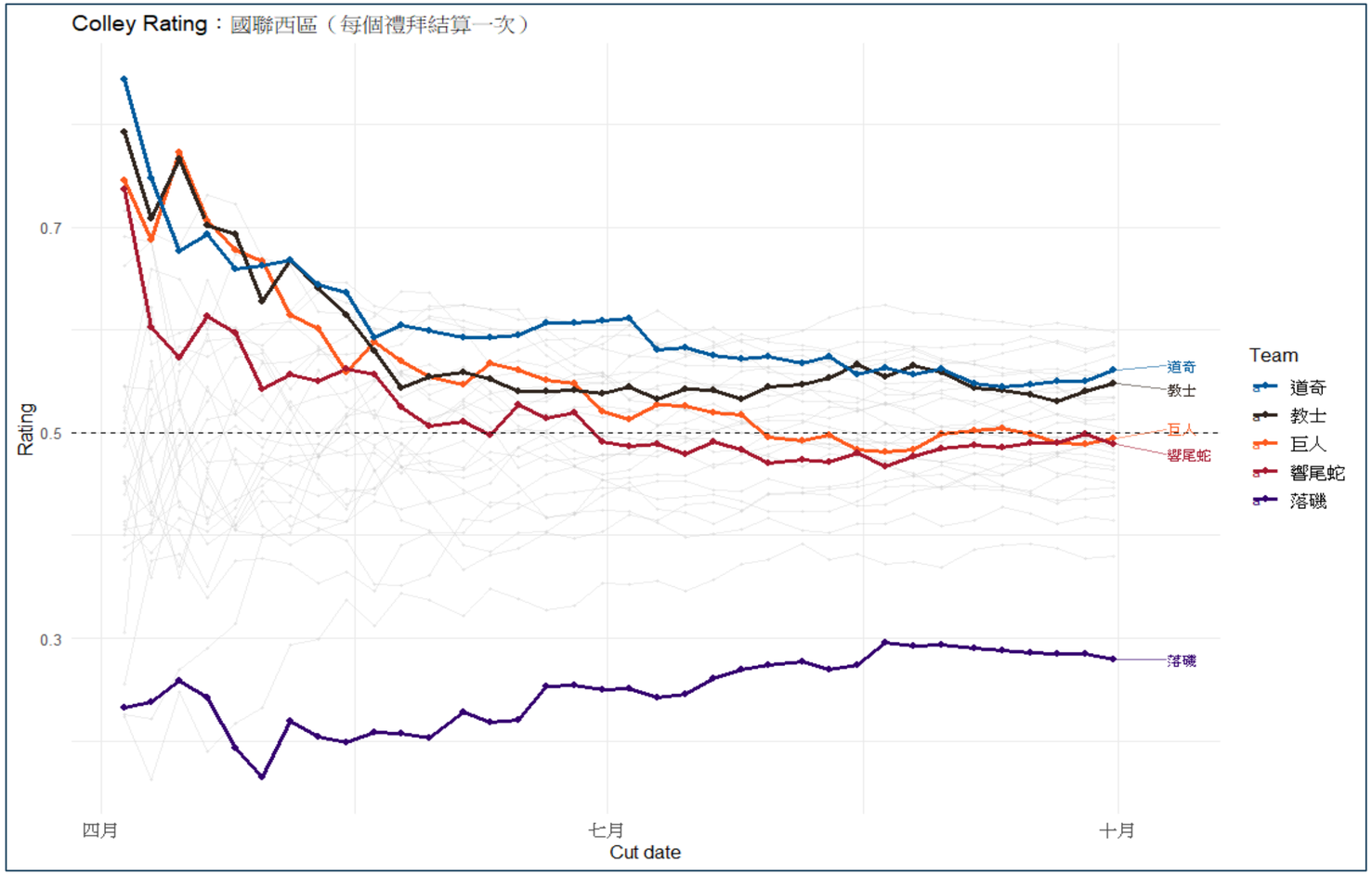
$$C\vec{r} = \vec{b}$$

存在且僅有**唯一解**。

接著，利用 **Cholesky 分解** (Cholesky Decomposition) 高效求解，可以**一次性獲得所有隊伍的公平評分**，同時考慮勝負與賽程強度，避免迭代的複雜運算，以確保 Colley 評分結果的穩定性與一致性。

Colley's Bias Free Ranking Method

下圖為利用 **Colley's Method** 進行預測後的視覺化結果，以國聯西區為例，並將預測的動態評分趨勢與右上圖之 **2025 年球季實際最終排名** 進行對照。



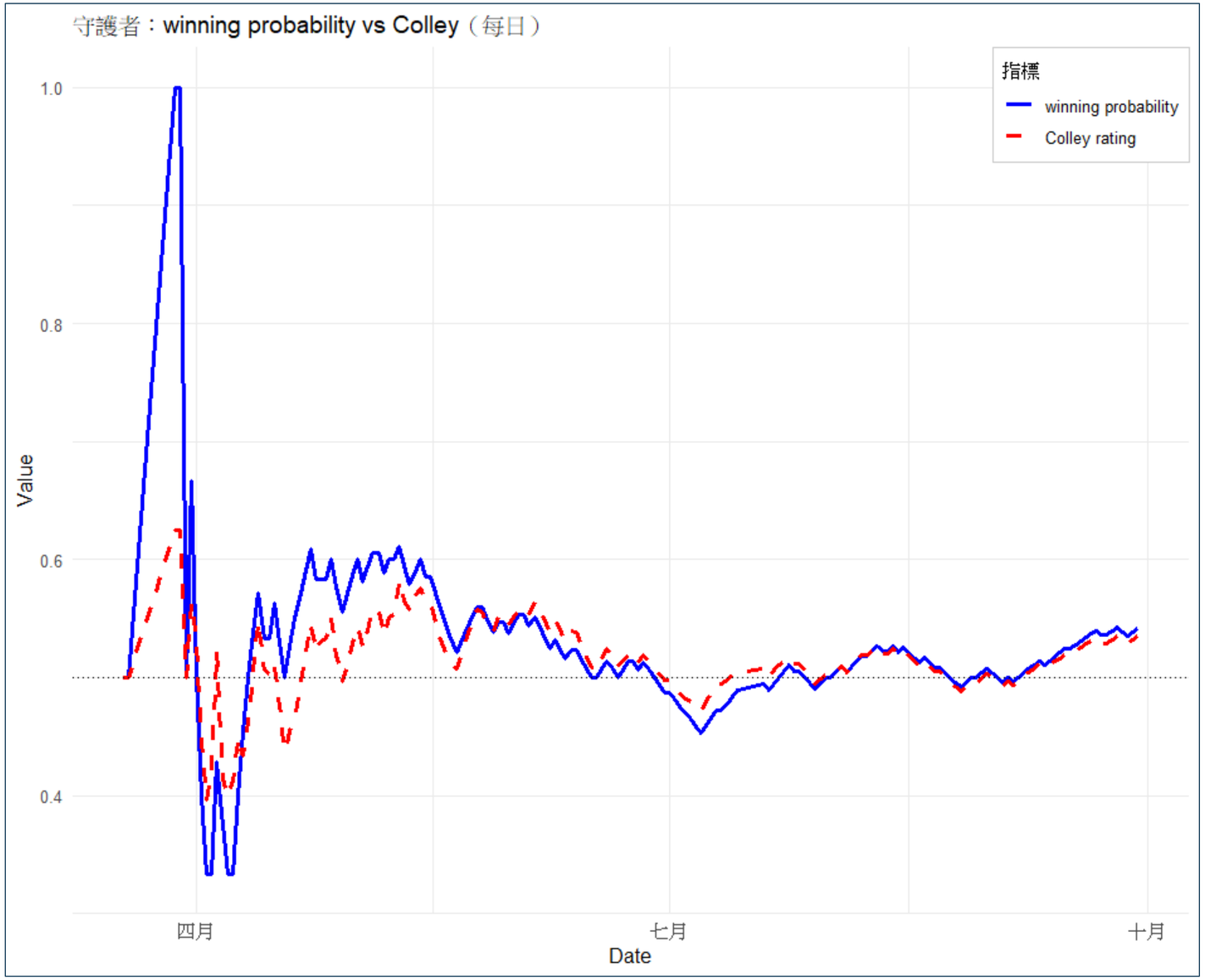
NL WEST	W	L	PCT
Los Angeles Dodgers y	93	69	.574
San Diego Padres w	90	72	.556
San Francisco Giants	81	81	.500
Arizona Diamondbacks	80	82	.494
Colorado Rockies	43	119	.265

由左圖可見，Colley's Method 隨比賽累積逐步收斂，並反映各隊在 2025 年國聯西區的整體實力變化趨勢：

- **道奇（Dodgers）與教士（Padres）** 整季評分大致維持領先，顯示其長期穩定的勝率與較強的賽程強度表現。
- **巨人（Giants）與響尾蛇（Diamondbacks）** 位於中段區間，呈現一定程度的波動，反映其時強時弱的競爭態勢。
- **洛磯（Rockies）** 之 Colley Rating 明顯低於其他隊伍，顯示其相對較弱的整體實力。

整體而言，Colley Method 在不考慮勝分差的情況下，透過勝負與賽程強度有效呈現球隊實力的長期趨勢，並提供穩定的排名判斷依據。

Colley Rating vs. winning probability



以守護者球隊為例，右圖比較了球隊每日的**Colley Rating** 與 **勝率（winning probability）**的變化趨勢。

可以發現，在賽季初期，由於各隊賽程強度落差較大，勝率會出現明顯波動。但隨著球季推進，各隊賽程難度逐漸接近，使得勝率走勢變較平穩。相較之下，**Colley Rating 曲線更為平滑**，主要因其考量了對手強度，因此能有效降低單場比賽結果波動所造成的雜訊，更能呈現球隊**整體且穩定的競爭實力趨勢**。

Extension Of Elo To MOV

Standard Elo Rating System

Elo 系統由 Arpad Elo 在 1960 年代提出，最初用於國際西洋棋排名。

它是一種基於**比賽結果**的動態評分模型，每場比賽後根據勝負結果更新雙方的 Elo 分數，其更新僅基於輸贏的**二元結果**，因此，無法區分大比分勝出或險勝的差異。

STEP1

每隊分配起始 Elo 分數（通常為1500）

STEP2

雙方預測勝率，以 i vs j 的第 t 場比賽為例：

$$\widehat{P}_{ijt} = \frac{1}{1 + 10^{(R_j - R_i)/400}}$$

STEP3

賽後，根據勝負結果調整 Elo 分數，以 i 隊為例：

$$R_{i(t+1)} = R_{it} + K(W_{ijt} - \widehat{P}_{ijt})$$

STEP4

每場比賽後重複計算 Elo 分數，下一場再用新的 Elo 分數繼續預測，回到 STEP2。

Extension Of Elo To MOV

Extension – Margin of Victory Elo (MOV-Elo)

MOV 模型是標準 Elo 系統的擴充，旨在納入**勝分差（Margin of Victory, MOV）**資訊。例如一場網球直落盤勝利與一場苦戰決勝盤勝利，對評分應該影響不同。

本次分析採用的模型是 **Joint Additive MOV-Elo 模型**，其主要目標同樣在於預測比賽勝負結果。此模型與其他以勝負為焦點的模型不同，因為在更新選手評分時，它會以實際的勝負分差的函數取代觀察到的勝負結果（win result）。

比較

- **標準 Elo**：根據勝負更新，假設所有勝利的價值相同。
- **MOV-Elo**：加入得分差資訊，使模型更靈敏，更準確反映隊伍的實際實力。

Extension Of Elo To MOV

Elo Rating System

- **預測勝率（Expected Score）**

  隊伍 i 對隊伍 j 的預期勝率為：

  $$\hat{P}_{ijt} = \frac{1}{1 + 10^{(R_j - R_i)/400}}$$

- **更新規則（Update Rule）**

  比賽結束後：

  $$R_{i(t+1)} = R_{it} + K(W_{ijt} - \hat{P}_{ijt})$$

  其中：

  - R_{it}：隊伍 i 第 t 場比賽後的 Elo 分數
  - K：學習率（K-factor，控制更新幅度）
  - W_{ijt}：實際結果（勝 = 1，負 = 0，平 = 0.5）
  - $\hat{P}_{ijt}$：預測第 $t+1$ 場比賽的勝率

Joint Additive MOV-Elo

- **預測勝率（Expected Score）**

  隊伍 i 對隊伍 j 的預期勝率為：

  $$\hat{P}_{ijt} = \frac{1}{1 + 10^{(R_j - R_i)/\sigma_2}}$$

- **更新規則（Update Rule）**

  比賽結束後：

  $$R_{i(t+1)} = R_{it} + K_1(M_{ijt} - \hat{M}_{ijt}) + K_2(W_{ijt} - \hat{P}_{ijt})$$

  其中：

  - M_{ijt}：以 i vs j 的第 t 場比賽的實際勝分差
  - $\hat{M}_{ijt} = \frac{R_{it} - R_{jt}}{\sigma_1}$：$i$ vs j 的第 t 場比賽的預期勝分差
  - σ_1：兩隊實際得分的標準差
  - σ_2：兩隊 Elo 分數的標準差

Extension Of Elo To MOV

The procedure of Joint Additive MOV-Elo

STEP1 每隊分配起始 Elo 分數（通常為1500）

STEP2 計算雙方預測勝率，以隊伍 i 對戰隊伍 j 的第 t 場比賽為例：

$$\widehat{P}_{ijt} = \frac{1}{1 + 10^{(R_j - R_i)/\sigma_2}}$$

STEP3 賽後，根據實際勝負結果調整 Elo 分數，以隊伍 i 為例：

$$R_{i(t+1)} = R_{it} + K_1\left(M_{ijt} - \widehat{M}_{ijt}\right) + K_2(W_{ijt} - \widehat{P}_{ijt})$$

STEP4 每場比賽後重複計算 Elo 分數，下一場再用新的 Elo 分數繼續預測，回到 STEP2。

- M_{ijt}：以 i vs j 的第 t 場比賽的實際勝分差
- $\widehat{M}_{ijt} = \frac{R_{it} - R_{jt}}{\sigma_1}$ ： i vs j 的第 t 場比賽的預期勝分差
- σ_1：兩隊實際得分的標準差
- σ_2：兩隊 Elo 分數的標準差
- K_1：控制「分差殘差」的學習強度 (Margin learning rate)
- K_2：控制「勝負殘差」的學習強度 (Win-loss learning rate)

因此，我們需要估計的未知參數有：

$$\sigma_1、\sigma_2、K_1、K_2$$

Extension Of Elo To MOV

$$R_{i(t+1)} = R_{it} + K_1\left(M_{ijt} - \hat{M}_{ijt}\right) + K_2\left(W_{ijt} - \hat{P}_{ijt}\right)$$

Joint Additive MOV-Elo 的目標，是同時讓模型符合兩件事：

1. **分差預測要準確**

   → 實際分差 M_{ijt} 與預測分差 $\hat{M}_{ijt}$ 要接近

2. **勝負機率預測要合理**

   → 即預測勝率 $\hat{P}_{ijt}$ 能解釋實際勝負

因此在估計未知參數：

$$\sigma_1、\ \sigma_2、\ K_1、\ K_2$$

時，透過一個聯合目標函數（joint objective function），把**分差誤差（橘色）**與**勝率對數概似（藍色）**一起考量。

目標函數

$$\mathcal{L}(\theta) = \frac{1}{N}\left[\frac{\sum_{i,j,t}(\hat{M}_{ijt}(\theta) - M_{ijt})^2}{3SD(M)} - \sum_{I,j,t} log(\hat{P}_{ijt}(\theta))\right]$$

$$\hat{M}_{ijt} = \frac{R_{it} - R_{jt}}{\sigma_1} \qquad \hat{P}_{ijt} = \frac{1}{1 + 10^{(R_j - R_i)/\sigma_2}}$$

其中：

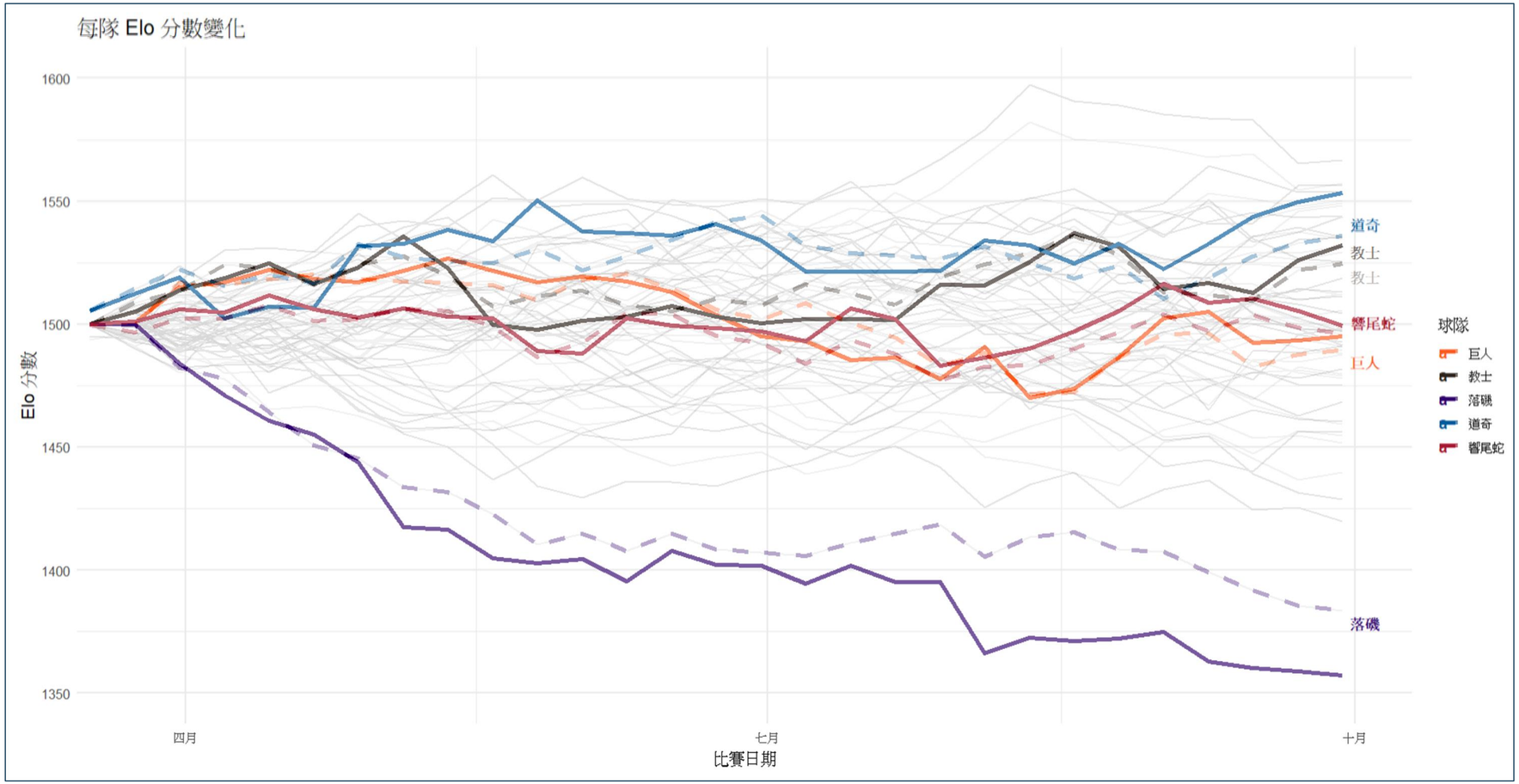
- 左半（橘色）逼使預測分差 $\hat{M}_{ijt}$ 更貼近 實際分差 M_{ijt}
- 右半（藍色）逼使預測勝率 $\hat{P}_{ijt}$ 更能解釋實際勝負結果

藉由最小化聯合目標函數，可以同時找到能最佳化「勝負」與「分差」預測的參數組合，使 Elo 更新式能更完整反映比賽的實際競爭強度。

Extension Of Elo To MOV

下圖為國聯西區 **MOV-ELO 分數逐日表現**。

虛線代表 **標準 ELO**，實線則為加入勝分差（Margin of Victory, MOV）後的 **MOV-ELO**。



根據圖中曲線可以發現與標準 ELO 相比 **MOV-ELO 曲線起伏更明顯**，能**捕捉到比分差距造成的強弱變化**，反映不同隊伍每場比賽的實際競爭強度。

⚾ 舉例來說：

- 落磯隊的 MOV-ELO 持續明顯下降，反映其整季較弱的表現。
- 道奇與教士的 MOV-ELO 則較穩定並維持較高評分。

整體而言，**MOV-ELO 提供的資訊比標準 ELO 更細緻**，能更準確反映比賽中「競爭力差距」的變化，讓最終排名預測更貼近實際比賽強弱。

Beta Regression

◆ 比較 Elo / MOV-Elo / Colley 與真實勝率排序之差異

PROCEDURE

- **STEP0** 以 7/15 前資料作訓練集，7/15 後資料作測試集
- **STEP1** 使用 **合適的模型** 來預測勝率
- **STEP2** 依預測勝率排序 → 得到預測排名
- **STEP3** 使用 **Spearman 等級相關係數**評估預測準確度

合適的模型？

① **Linear Regression**
- 應變數：連續型
- 預測值：$(-\infty, \infty)$

② **Logistic Regression**
- 應變數：0/1 （例如: 輸/贏）
- 預測值：最終勝率（介於 0~1 之間）

③ **Beta Regression**
- 應變數：最終勝率 （介於 0~1 之間）
- 預測值：最終勝率（介於 0~1 之間）

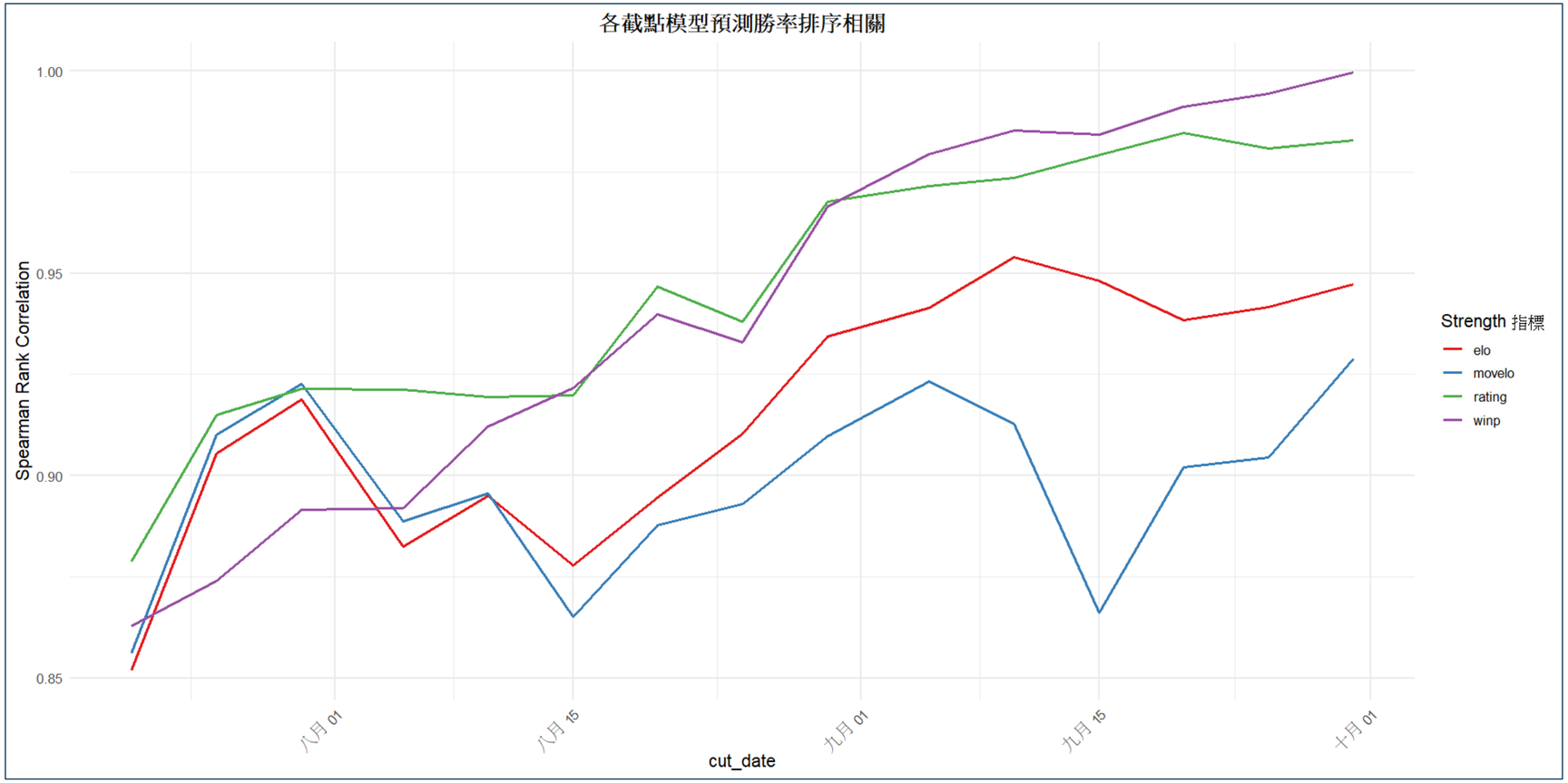
- 使用 linear regression : 預測值不一定介於 0~1 之間
- 使用 logistic regression : 應變數並非 0/1

結論

使用 Beta Regression（ Beta 分配的定義域是介於 0~1 之間 ）

Beta Regression – Result 1

◆ 分別對四個綜合實力指標直接配適 Beta Regression，不考慮其他變數。



預測準確度的整體趨勢：

- 7 月以前（賽季前半）：ELO、MOV-ELO、Colley 的預測力普遍優於 Win-rate。
- 8–9 月（賽季中期）：Colley 與 Win-rate 的表現逐漸上升，在這段期間成為預測力最好的兩個方法。
- 9 月之後（賽季末期）：Win-rate 穩定表現最佳，因為此時勝率已能最直接反映球隊最終戰力。

Predict the Final Rank by using Beta Regression

◆ 本研究以 Beta Regression 建模最終勝率，並同時考慮**球隊強度**、**運氣因素**與**主客場比例**，以提升最終 排名預測的準確度。

Variables

1. **球隊整體實力（Comprehensive Strength）**

   → 使用 Win Rate、Colley、ELO score 以及 MOV ELO score 作為代表球隊強度的指標。

2. **luck_winp - Win Portion（幸運勝場比率）**

   → 衡量「運氣好時的小分差勝利比例」（運氣好 → 較多兩分差勝；運氣差 → 較少兩分差勝）

3. **Unluck – Lose Portion（不幸敗場比率）**

   → 衡量「運氣不佳時的小分差落敗比例」（運氣差 → 較多兩分差輸；運氣好 → 較少兩分差輸）

4. **主客場比例（Home-Away Factor）**

   → 考慮球隊賽程中主場比率可能對勝率造成的結構性影響。

5. **最終勝率（應變數：Y）**

   → 介於 0 ~ 1 之間，因此適合使用 **Beta Regression** 作為模型。

Predict the Final Rank by using Beta Regression

球隊整體實力
Ex: 勝率（winp）

rating is from Colley Method

運氣因子
luck_winp、unluck_losep

主客場比率
home_percent

	cut_date	team	total_games	total_wins	winp	rating	elo	movelo	luck_wins	unluck_lose	luck_winp	unluck_losep	home_games	home_percent	final_winp
611	2025-07-15	水手	96	51	0.5312500	0.5365210	1514.640	1522.684	27	20	0.2812500	0.2083333	46	0.4791667	0.556
612	2025-07-15	洋基	96	53	0.5520833	0.5491885	1512.041	1539.354	18	26	0.1875000	0.2708333	49	0.5104167	0.580
613	2025-07-15	海盜	97	39	0.4020619	0.4222056	1462.401	1483.608	19	27	0.1958763	0.2783505	47	0.4845361	0.438
614	2025-07-15	白襪	97	32	0.3298969	0.3464278	1428.463	1442.757	12	36	0.1237113	0.3711340	50	0.5154639	0.370
615	2025-07-15	皇家	97	47	0.4845361	0.4786717	1486.661	1486.094	26	21	0.2680412	0.2164948	50	0.5154639	0.506
616	2025-07-15	紅人	97	50	0.5154639	0.5102050	1510.611	1511.356	21	21	0.2164948	0.2164948	50	0.5154639	0.512
617	2025-07-15	紅襪	98	53	0.5408163	0.5264040	1526.464	1539.775	27	23	0.2755102	0.2346939	52	0.5306122	0.549
618	2025-07-15	紅雀	97	51	0.5257732	0.5227849	1505.650	1500.500	25	19	0.2577320	0.1958763	49	0.5051546	0.481
619	2025-07-15	老虎	97	59	0.6082474	0.6013603	1540.513	1529.305	23	11	0.2371134	0.1134021	50	0.5154639	0.537
620	2025-07-15	落磯	96	22	0.2291667	0.2449523	1392.108	1383.652	17	32	0.1770833	0.3333333	46	0.4791667	0.265
621	2025-07-15	藍鳥	96	55	0.5729167	0.5764661	1541.363	1520.190	30	19	0.3125000	0.1979167	48	0.5000000	0.580
622	2025-07-15	費城人	96	55	0.5729167	0.5657945	1529.807	1540.329	24	23	0.2500000	0.2395833	46	0.4791667	0.593
623	2025-07-15	遊騎兵	97	48	0.4948454	0.4910639	1496.799	1533.023	21	24	0.2164948	0.2474227	45	0.4639175	0.500
624	2025-07-15	運動家	98	41	0.4183673	0.4302537	1469.196	1441.931	20	20	0.2040816	0.2040816	51	0.5204082	0.469
625	2025-07-15	道奇	97	58	0.5979381	0.5833975	1538.708	1524.462	25	20	0.2577320	0.2061856	50	0.5154639	0.574
626	2025-07-15	釀酒人	96	56	0.5833333	0.5741169	1549.548	1559.661	23	20	0.2395833	0.2083333	50	0.5208333	0.599
627	2025-07-15	金鶯	95	43	0.4526316	0.4533308	1485.684	1462.639	16	15	0.1684211	0.1578947	47	0.4947368	0.463

Beta Model (Win-Rate)

Significance

```
> fit <- betareg(final_winp ~ winp + luck_winp + unluck_losep +
+                  home_percent, data = elo_train)
> summary(fit)

Call:
betareg(formula = final_winp ~ winp + luck_winp + unluck_losep + home_percent,
    data = elo_train)

Quantile residuals:
    Min      1Q  Median      3Q     Max
-3.2781 -0.5065 -0.0165  0.6017  3.0425

Coefficients (mean model with logit link):
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.38535    0.08637 -16.039  < 2e-16 ***
winp          2.20570    0.09305  23.704  < 2e-16 ***
luck_winp    -0.36236    0.12417  -2.918  0.00352 **
unluck_losep  0.26905    0.13945   1.929  0.05369 .
home_percent  0.60183    0.07364   8.173 3.02e-16 ***

Phi coefficients (precision model with identity link):
      Estimate Std. Error z value Pr(>|z|)
(phi)  101.860      5.711   17.83   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

VIF（檢查共線性）

```
> vif(fit)
       winp    luck_winp unluck_losep home_percent
   2.037952     1.603252     1.710392     1.084168
```

- 顯著的變數:

  winp、luck_winp、home_percent
- 有無共線性: 無

Beta Model (Colley)

Significance

```
> fit <- betareg(final_winp ~ rating + luck_winp + unluck_losep +
+                  home_percent, data = elo_train)
> summary(fit)

Call:
betareg(formula = final_winp ~ rating + luck_winp + unluck_losep + home_percent,
    data = elo_train)

Quantile residuals:
    Min      1Q  Median      3Q     Max
-3.5436 -0.6011  0.0188  0.6997  2.8733

Coefficients (mean model with logit link):
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.28562    0.08657 -14.851  < 2e-16 ***
rating        2.41348    0.10775  22.398  < 2e-16 ***
luck_winp    -0.36690    0.12778  -2.871  0.00409 **
unluck_losep -0.14920    0.13569  -1.100  0.27153
home_percent  0.38982    0.07343   5.309  1.1e-07 ***

Phi coefficients (precision model with identity link):
      Estimate Std. Error z value Pr(>|z|)
(phi)   96.660      5.418   17.84   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

VIF（檢查共線性）

```
> vif(fit)
      rating     luck_winp unluck_losep home_percent
    1.775925      1.619788      1.544162      1.035797
```

- 顯著的變數:

  rating、luck_winp、home_percent
- 有無共線性: 無

Beta Model (ELO)

Significance

```
> fit <- betareg(final_winp ~ elo + luck_winp + unluck_losep +
+                  home_percent, data = elo_train)
> summary(fit)

Call:
betareg(formula = final_winp ~ elo + luck_winp + unluck_losep + home_percent,
    data = elo_train)

Quantile residuals:
    Min      1Q  Median      3Q     Max
-4.0981 -0.5988 -0.0416  0.7297  3.1205

Coefficients (mean model with logit link):
               Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.014e+01  4.105e-01 -24.698  < 2e-16 ***
elo           6.644e-03  2.656e-04  25.011  < 2e-16 ***
luck_winp    -1.786e-01  1.169e-01  -1.527    0.127
unluck_losep  8.606e-02  1.311e-01   0.656    0.512
home_percent  3.871e-01  6.942e-02   5.576 2.46e-08 ***

Phi coefficients (precision model with identity link):
      Estimate Std. Error z value Pr(>|z|)
(phi)   107.87       6.05   17.83   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

VIF（檢查共線性）

```
vif(fit)
       elo    luck_winp unluck_losep home_percent
  1.707367     1.516663     1.611318     1.034098
```

- 顯著的變數:

  Elo、home_percent
- 有無共線性: 無

Beta Model (MOV-ELO)

Significance

```
> fit <- betareg(final_winp ~ movelo + luck_winp + unluck_losep +
+                  home_percent, data = elo_train)
> summary(fit)

Call:
betareg(formula = final_winp ~ movelo + luck_winp + unluck_losep + home_percent,
    data = elo_train)

Quantile residuals:
    Min      1Q  Median      3Q     Max
-3.2945 -0.5939 -0.0064  0.6700  2.9428

Coefficients (mean model with logit link):
               Estimate Std. Error z value Pr(>|z|)
(Intercept)  -9.7835553  0.3836132 -25.504  < 2e-16 ***
movelo        0.0064331  0.0002488  25.858  < 2e-16 ***
luck_winp     0.1884714  0.1111890   1.695   0.0901 .
unluck_losep -0.2954935  0.1237116  -2.389   0.0169 *
home_percent  0.3127744  0.0684253   4.571 4.85e-06 ***

Phi coefficients (precision model with identity link):
      Estimate Std. Error z value Pr(>|z|)
(phi)  111.272      6.242   17.83   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

VIF（檢查共線性）

```
> vif(fit)
     movelo    luck_winp unluck_losep home_percent
   1.321453     1.408867     1.471765     1.025579
```

- 顯著的變數:

  movelo、unluck_losep、home_percent
- 有無共線性: 無

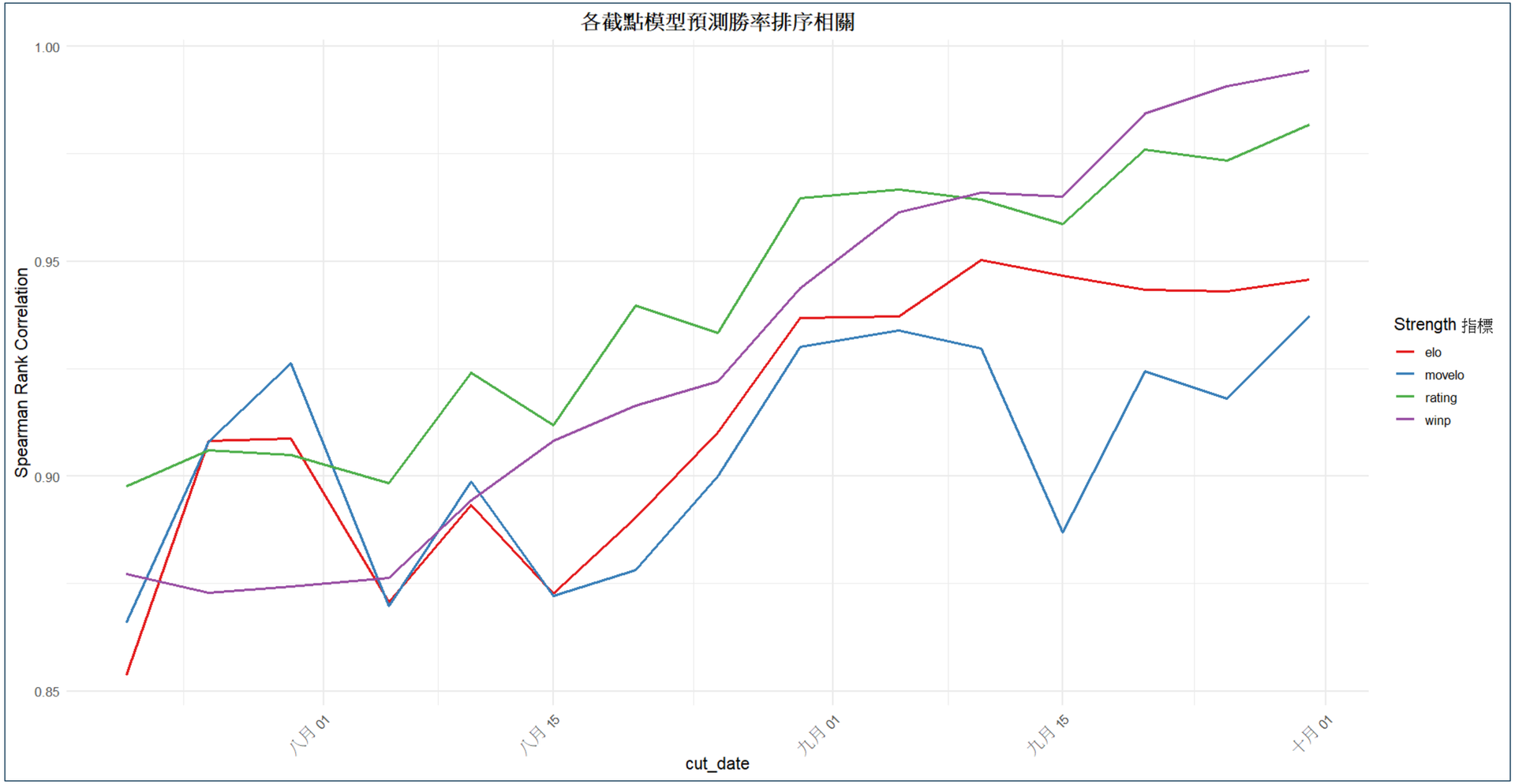
Beta Regression – Result 2

不同指標（ELO、MOV-ELO、Colley、Win-rate）在 Beta Regression 中所加入的顯著變數不同，因此每條線代表的模型並不相同。

下圖比較的是各模型「隨切割日期的預測能力變化」，而非模型之間的直接比較。

各條線代表的是「不同的 Beta Regression 模型」，每條線的模型架構（變數組合）都不一樣。

- ELO 模型加入：
  Elo、home_percent
- MOV-ELO 模型加入：
  movelo、unluck_losep、home_percent
- Colley 模型加入：
  rating、luck_winp、home_percent
- Win-rate 模型加入：
  winp、luck_winp、home_percent



Comparison

◆ 使用 **Spearman 等級相關係數**評估預測準確度

⚾ 為評估該月份的預測表現，此為該月份預測準確率的平均。

考慮其他變數

	勝率	Colley	ELO	MOV-ELO
7 月	0.875	**0.903**	0.890	0.900
8 月	0.910	**0.929**	0.896	0.891
9 月	**0.977**	0.970	0.944	0.922

不考慮其他變數

	勝率	Colley	ELO	MOV-ELO
7 月	0.876	**0.905**	0.892	0.896
8 月	0.927	**0.936**	0.899	0.890
9 月	**0.989**	0.979	0.945	0.906

結論

① 是否加入其他變數對預測整體名次的準確度影響不大

② 無論加入與否：

- 7 月與 8 月皆由 Colley Method 取得最高相關係數 → 最能貼近真實排序
- 9 月（賽季後期）則以勝率表現最佳，因為球隊戰力趨於穩定、數據集中度增高

Comparison

◆ 比較不同方法預測出**晉級季後賽比率**

考慮其他因素

	勝率	Colley	ELO	MOV-ELO
7 月	0.833	0.861	0.861	**0.917**
8 月	0.875	0.875	**0.889**	**0.889**
9 月	0.986	**1**	0.931	0.875

不考慮其他因素

	勝率	Colley	ELO	MOV-ELO
7 月	0.806	0.806	0.861	**0.917**
8 月	0.889	0.889	**0.903**	**0.903**
9 月	**0.986**	0.972	0.917	0.875

結論

加入其他因素後，模型的預測能力 僅有些微提升，**整體影響並不明顯**。

以「是否能晉級季後賽」作為預測目標時，各月份表現如下：

7 月：MOV-ELO 表現最佳 → 此時賽季仍在中段，加入勝分差能更有效捕捉球隊真實強度差距，因此 MOV-ELO 的預測最準確。

8 月：MOV-ELO 與 ELO 表現最佳 → 賽季逐漸穩定，傳統 ELO 與 MOV-ELO 趨於一致，兩者皆能穩定反映球隊整體強弱。

9 月：Colley 與勝率表現最佳 → 接近季末，各隊勝負趨於集中、戰力輪廓明顯，因此 不需要考慮勝分差，Colley 與單純勝率反而預測最準確。

Conclusion

① 賽季前中期，調整賽程強度或勝分差的指標較有效

→ 表示此階段可能存在**賽程不平衡**的問題，因此需要補正。

② 越接近賽季末，賽程已趨於平衡

→ 此時直接使用當下勝率（Win Rate）即可代表球隊實力，預測準確度最高。

③ ELO 與 MOV-ELO 難以分出優劣

→ 顯示勝分差（Margin of Victory）對預測最終勝率的幫助有限

④ ELO、MOV-ELO 曲線起伏較大，且在賽末表現較差

→ 因為模型需估計未知參數，參數不穩定，導致預測波動提高，影響賽末表現。

⑤ 納入運氣因素（luck/unluck）與主客場因素後，訓練資料顯著，但預測無提升

→ 代表這些因素對預測實際沒有幫助，或在賽季中後段其影響已逐漸消失。

Suggestion

- 明星賽後預測最終排名時，不需再額外考慮運氣成分與主客場因素。
- 賽季中後段→ 建議使用 Colley Rating。
- 賽季末→ 建議直接使用當下勝率（Win Rate）。賽程完全平衡後，勝率能最直接反映球隊最終實力。

Future Work

改良 ELO 系統中未知參數（如 K、σ 等）的估計方法。

目前的 MOV-ELO 與 ELO 因需估參數而不穩定，未來可透過：

- Bayesian Estimation
- Cross-Validation 自動搜尋最佳參數
- Hierarchical / Regularized ELO

來提升模型的穩定性，使 ELO 類模型更具代表性與泛化能力。

Reference

- Colley, W. N. (Ph.D., Princeton University). **Colley's Bias Free College Football Ranking Method: The Colley Matrix Explained.**
- Kovalchik, S. (Zelus Analytics, Austin, TX, USA). **Extension of the Elo Rating System to Margin of Victory.**
- https://tw.sports.yahoo.com/mlb/teams/{team}/schedule/?season=2025&scheduleType=list

Our Github

- https://github.com/r26131109-a11y/2025-Taiwan-baseball-data-analysis-competition

MLB2025

THANK YOU