

Problem Chosen

C

**2026
MCM/ICM
Summary Sheet**

Team Control Number

1234

XXXXXXX

Summary

Keywords: keyword1; keyword2

Contents

1	Introduction	2
1.1	Background	2
1.2	The Description of the Problem	2
1.3	Our work	2
2	Assumptions	3
3	Notations	4
4	问题 1：观众投票估计模型	5
4.1	问题剖析与建模策略	5
4.2	投票组合方法	5
4.3	基线模型：幂律投票假设	5
4.4	精确反推模型：约束优化框架	6
4.4.1	投票偏好因子的学习	6
4.4.2	优化问题的数学构建	6
4.4.3	SLSQP 算法的收敛性分析	7
4.5	模型评估与集成策略	7
4.5.1	多维度一致性度量体系	7
4.5.2	不确定性量化与可信度分析	8
4.5.3	智能集成与自适应策略	8
5	Sensitivity Analysis	9
6	Model Evaluation and Promotion	9
6.1	Model Evaluation	9
6.1.1	Advantages	9
6.1.2	Limitations	9
6.2	Future Work	9
6.2.1	Model extension	9

6.2.2 Model application	9
7 Conclusions	9

1 Introduction

1.1 Background

1.2 The Description of the Problem

本问题聚焦于《与星共舞》(Dancing with the Stars, DWTS) 节目的投票机制分析。在该竞技真人秀中，名人选手 (celebrity contestants) 与专业舞者 (professional dancers) 配对表演，通过评委打分 (judge scores) 和观众投票 (fan votes) 的组合决定每周淘汰结果。然而，观众投票数据作为商业机密从未公开，且节目在 34 季中采用了不同的投票组合规则 (voting combination methods)。核心挑战在于：在观众投票未知的情况下，如何评估投票系统的公平性并提出优化方案。

具体任务包括：

- 观众投票估算：构建数学模型，基于评委得分、选手特征和淘汰结果反向推断每位选手的观众投票数。评估估算结果与实际淘汰情况的一致性 (consistency)，并量化估算的不确定性 (uncertainty measures)。
- 投票方法对比：对比分析节目使用的两种组合方法——基于排名法 (rank-based method) 和基于百分比法 (percentage-based method) 在各赛季产生的结果差异。针对争议案例 (如 Jerry Rice、Bobby Bones 等评委低分但观众高票的选手)，评估方法选择和评委复议机制 (judges choosing from bottom two) 对最终排名的影响，并推荐未来赛季的最优方法。
- 影响因素分析：构建模型量化专业舞者经验、名人年龄 (age)、行业背景 (industry) 等因素对比赛结果的影响。分析这些因素对评委得分和观众投票的作用机制是否存在差异。
- 新系统设计：提出一种更“公平”(fair) 或更具观赏性 (exciting) 的投票组合系统。通过历史数据模拟验证新系统的有效性，并为制作方提供可操作的实施建议。
- 报告撰写：提交不超过 25 页的分析报告及 1-2 页备忘录，为 DWTS 制作方提供关于投票组合方式影响的总结和未来赛季的决策建议。

1.3 Our work

2 Assumptions

Assumption 1: The light attenuation and scattering in the underwater environment adhere to the Jaffe-McGlamery imaging model, where the propagation of light follows an exponential decay law.

3 Notations

The core symbols and their definitions used in this study are summarized in Table 1, providing an overview of the key parameters and their related meanings.

Table 1: Notations used in this literature

Symbol	Description
n	第 w 周剩余选手数量
S_i	选手 i 的评委总分
V_i	选手 i 的估计观众投票数
E_i	淘汰指示器: 若选手 i 被淘汰则 $E_i = 1$, 否则为 0
\mathcal{M}	投票组合方法: “rank” (排名法) 或 “percent” (百分比法)
C_i	选手 i 在方法 \mathcal{M} 下的综合得分
α	投票与评委得分关系的幂律指数
θ	先验偏好因子 (舞伴效应、行业效应)
R_i^{judge}	选手 i 的评委得分排名
R_i^{fan}	选手 i 的观众投票排名
β_p	专业舞伴 p 的投票偏好效应
γ_d	名人行业 d 的投票偏好效应
B_i^w	选手 i 在第 w 周的存活提升值
\mathbf{v}	观众投票份额向量 (v_1, \dots, v_n)

4 问题 1：观众投票估计模型

4.1 问题剖析与建模策略

问题 1 本质上是一个欠定逆问题 (underdetermined inverse problem)：在观众投票这一隐变量缺失的情况下，从淘汰结果和评委得分的联合观测中进行统计推断。这类问题在计量经济学和信号处理领域广泛存在，其核心挑战在于解的非唯一性与模型的可识别性 (identifiability)。

问题的数学结构。设第 w 周有 n 名选手，评委得分 $\mathbf{S} = (S_1, \dots, S_n)$ 可观测，观众投票 $\mathbf{V} = (V_1, \dots, V_n)$ 不可观测，淘汰结果 E 已知。问题可形式化为：给定映射 $f: (\mathbf{S}, \mathbf{V}) \rightarrow E$ 和观测 (\mathbf{S}, E) ，反推 \mathbf{V} 。由于 $|\mathbf{V}| = n$ 个未知数仅对应 1 个淘汰约束，系统严重欠定 ($n \gg 1$)。

可识别性困境。多个投票分布可产生相同淘汰结果，导致解不唯一。传统方法有二：(1) 施加正则化约束 (如最大熵、稀疏性)，但缺乏投票行为的先验支持；(2) 引入辅助假设 (如投票与得分的参数关系)，但可能过度简化现实。我们的策略是将两者结合——用行为模型提供软约束，用淘汰结果施加硬约束，在约束空间中搜索最符合经验规律的解。

方法演变的影响。第 28 季前后投票组合规则的变化 (排名法 \leftrightarrow 百分比法) 改变了 f 的函数形式，使得相同的 (\mathbf{S}, \mathbf{V}) 可能导致不同的 E 。这要求模型具备规则自适应性，能够在统一框架下处理两种机制。

基于上述分析，我们设计双层贝叶斯框架：基线层建立 $P(\mathbf{V}|\mathbf{S})$ 的先验分布，反推层通过 $P(\mathbf{V}|\mathbf{S}, E)$ 进行后验推断。这种分层设计既保证了理论严谨性 (后验一致性)，又具备实践可行性 (先验可解释性)。

4.2 投票组合方法

不同赛季使用的两种投票组合方法为 (相关符号定义见表 1)：

排名法 (赛季 1-2、28-34)：

$$C_i^{\text{rank}} = R_i^{\text{judge}} + R_i^{\text{fan}} \quad (1)$$

其中 $R_i^{\text{judge}} = \text{rank}(-S_i)$ 和 $R_i^{\text{fan}} = \text{rank}(-V_i)$ 。综合排名最高 (即 C_i^{rank} 最大，表现最差) 的选手被淘汰。

百分比法 (赛季 3-27)：

$$C_i^{\text{percent}} = \frac{S_i}{\sum_{j=1}^n S_j} + \frac{V_i}{\sum_{j=1}^n V_j} \quad (2)$$

综合百分比最低的选手被淘汰。

4.3 基线模型：幂律投票假设

观众投票与评委得分的关系既非完全独立，亦非严格线性。我们提出幂律假设捕捉这种非线性依赖：

$$V_i \propto S_i^\alpha, \quad \hat{V}_i = V_{\text{total}} \cdot \frac{S_i^\alpha}{\sum_{j=1}^n S_j^\alpha} \quad (3)$$

指数 α 的经济学解释：当 $\alpha > 1$ 时，高分选手获得超比例支持（“马太效应”）； $\alpha < 1$ 时体现“同情弱者”； $\alpha = 1$ 退化为比例模型。通过网格搜索最大化淘汰预测准确率，校准得 $\alpha^* \approx 1.2$ ，印证观众存在温和的“强者偏好”。

不确定性通过蒙特卡洛采样量化：向幂律预测注入 15% 高斯噪声，生成 100 个样本构建 95% 置信区间。这为后续分析提供了统计显著性检验的基础。

4.4 精确反推模型：约束优化框架

基线模型提供先验分布 $P(\mathbf{V}|\mathbf{S})$ ，但无法保证后验一致性。我们建立约束优化框架，从淘汰结果反推投票的后验分布 $P(\mathbf{V}|\mathbf{S}, E)$ 。

4.4.1 投票偏好因子的学习

在优化前，从全局数据学习系统性偏好。定义“存活提升” B_i^w 衡量评分排名与生存结果的错位程度：

$$B_i^w = \begin{cases} -(n - R_i^{\text{score}})/n & \text{若被淘汰 (评分越高越意外)} \\ (R_i^{\text{score}} - 1)/n & \text{若存活 (评分越低越意外)} \end{cases} \quad (4)$$

其中 R_i^{score} 为选手 i 的评分排名 ($1=$ 最高分)。

按专业舞伴 p 和名人行业 d 聚合提升值，得效应参数：

$$\beta_p = \frac{1}{|\mathcal{W}_p|} \sum_{w \in \mathcal{W}_p} B_i^w - \mu_{\text{overall}} \quad (5)$$

$$\gamma_d = \frac{1}{|\mathcal{W}_d|} \sum_{w \in \mathcal{W}_d} B_i^w - \mu_{\text{overall}} \quad (6)$$

结合标准化年龄，构建先验偏好因子：

$$\theta_i = \beta_{p(i)} + \gamma_{d(i)} + \delta_{\text{age}} \cdot \frac{\text{age}_i - \mu_{\text{age}}}{\sigma_{\text{age}}} \quad (7)$$

这些偏好因子识别了哪些舞伴、行业和年龄段的选手系统性地获得更多或更少的观众支持。正效应表示该因素带来额外投票支持，负效应则相反。

4.4.2 优化问题的数学构建

对投票份额 $\mathbf{v} = (v_1, \dots, v_n)$ （满足 $v_i \in [0, 1]$, $\sum v_i = 1$ ），最小化目标函数：

$$\mathcal{J}(\mathbf{v}) = \left\| \mathbf{v} - \mathbf{v}^{\text{prior}} \right\|_2^2 + \lambda \cdot \mathcal{L}_{\text{elim}}(\mathbf{v}) \quad (8)$$

其中先验份额为：

$$\mathbf{v}^{\text{prior}} = \text{normalize} \left[\frac{\mathbf{S}}{\sum_j S_j} \odot (1 + \boldsymbol{\theta}) \right] \quad (9)$$

淘汰约束的软化处理。对于排名法，离散排名函数不可微，我们采用 **sigmoid** 软排名近似：

$$\text{SoftRank}_i = n - \sum_{j \neq i} \frac{1}{1 + \exp[(x_j - x_i)/T]} \quad (10)$$

其中 x 可以是分数或投票份额， T 为温度参数（代码中设为 0.1）。该函数近似计算“有多少人比 i 差”，温度越小越接近真实排名。

综合软排名为 $C_i^{\text{soft}} = \text{SoftRank}_i(\mathbf{S}) + \text{SoftRank}_i(\mathbf{v})$ 。淘汰约束惩罚项为：

$$\mathcal{L}_{\text{rank}}(\mathbf{v}) = \sum_{\substack{i: E_i=1 \\ j: E_j=0}} \max(0, C_j^{\text{soft}} - C_i^{\text{soft}} + \delta)^2 \quad (11)$$

其中边际 $\delta = 0.5$ 确保被淘汰者的软排名显著高于存活者。

对于百分比法，约束为：

$$\mathcal{L}_{\text{percent}}(\mathbf{v}) = \sum_{\substack{i: E_i=1 \\ j: E_j=0}} \max(0, C_i - C_j + \epsilon)^2 \quad (12)$$

其中 $C_i = S_i / \sum S_j + v_i$ ，边际 $\epsilon = 0.01$ 。

正则化项的作用。 $\|\mathbf{v} - \mathbf{v}^{\text{prior}}\|_2^2$ 防止过拟合，在众多满足约束的解中选择最接近先验的那个。权重 $\lambda = 1000$ 通过实验确定，平衡正则化与约束满足。

4.4.3 SLSQP 算法求解

采用序列最小二乘规划 (SLSQP) 求解约束优化。该算法适合处理等式约束 $\sum v_i = 1$ 和盒约束 $v_i \in [0.001, 0.999]$ ，基于梯度信息快速收敛。

初始值设为先验 $\mathbf{v}^{\text{prior}}$ ，优化参数为：最大迭代 500 次，收敛容差 10^{-8} 。成功收敛的判断是目标函数值低于阈值且淘汰约束得到满足。

失败案例分析。对于约束无法满足的周次（优化失败），模型回退到备用估计：

$$\mathbf{v}^{\text{fallback}} = \text{normalize} \left[\frac{\mathbf{S}}{\sum S_j} \odot (1 + 0.3 \cdot \boldsymbol{\theta}) \right] \quad (13)$$

这些失败案例往往对应评分与淘汰结果存在强矛盾（如高分选手被淘汰），或数据记录异常（多人同时淘汰/无人淘汰）。

4.5 模型评估与集成策略

4.5.1 一致性度量体系

评估投票估计需要多个互补指标。代码中实现了以下度量：

精确匹配准确率定义为：

$$\text{ACC}_{\text{exact}} = \frac{1}{W} \sum_{w=1}^W \mathbb{1}\{\text{predicted}(w) = \text{actual}(w)\} \quad (14)$$

衡量能否精准预测每周被淘汰者，这是最严格的标准。

底部 N 准确率放宽判定条件，检验实际淘汰者是否落入预测的“危险区”（综合得分最低的若干名）。这更符合实际应用——关心“谁有风险”而非“谁一定淘汰”。

约束满足边际量化淘汰的“确定性”：

$$\text{Margin}_k = \min_{i:E_i=0} |C_k - C_i| \quad (15)$$

边际大表示淘汰显著，边际小则属“惊险淘汰”，投票的微小波动即可改变结果。

4.5.2 集成策略的实现

代码采用场景感知的模型选择机制：

淘汰周次：优先使用精确反推模型。运行 SLSQP 优化，若收敛则采用优化结果。

非淘汰周次：使用基线模型结合偏好因子。具体为：

$$\mathbf{v}^{\text{non-elim}} = 0.7 \cdot \mathbf{v}^{\text{baseline}} + 0.3 \cdot \mathbf{v}^{\text{prior}} \quad (16)$$

其中 $\mathbf{v}^{\text{baseline}}$ 为基线模型的幂律估计， $\mathbf{v}^{\text{prior}}$ 为带偏好因子的先验估计。混合比例 70:30 平衡了统计规律与个体特征。

优化失败周次：回退到备用估计方法。使用评分比例乘以偏好调整因子（权重 0.3），确保即使约束无法满足也能提供合理估计。

4.5.3 不确定性的传播

对于基线模型，不确定性通过蒙特卡洛采样量化。生成 100 个带噪声样本（噪声水平 15%），构建 95% 置信区间 $[Q_{2.5\%}, Q_{97.5\%}]$ 。

对于精确反推模型，不确定性体现在约束满足边际上。边际越小，说明多个投票分布都可能产生相同淘汰结果，估计的唯一性越差。

代码中还实现了标准化残差分析：

$$r_i = \frac{v_i^* - v_i^{\text{prior}}}{\sigma_i^{\text{prior}}} \quad (17)$$

其中 v_i^* 为优化得到的估计， σ_i^{prior} 为先验标准差（通过蒙特卡洛估计）。 $|r_i| > 2$ 被标记为异常，提示该选手的实际投票显著偏离预期。

4.5.4 双视角可解释性

集成策略提供理解投票的两个维度：

- 基线视角：揭示“常规投票模式”——若观众按评分、舞伴、行业等因素理性投票，结果应该如何。
- 精确视角：揭示“实际投票需求”——为了产生观察到的淘汰，投票实际上必须是怎样的。

两者的偏离 $\Delta_i = v_i^* - v_i^{\text{baseline}}$ 量化”异常程度”。 Δ 大的周次往往对应争议事件（如人气选手意外被淘汰，或低分选手意外晋级）。这种差异分析为理解节目的争议性结果（如题目提到的 Bobby Bones、Jerry Rice 等案例）提供了定量工具。

5 Sensitivity Analysis

6 Model Evaluation and Promotion

6.1 Model Evaluation

6.1.1 Advantages

6.1.2 Limitations

6.2 Future Work

6.2.1 Model extension

6.2.2 Model application

7 Conclusions

References

- [1] L. Prandtl, Fluid motions with very small friction, Proceedings of the 3rd International Mathematical Congress, Heidelberg: H. Schlichting, 1904, 484-491.

Appendices

Report on Use of AI

1. Bing AI

Query1: write a sort algorithm

Output: Sure, I can help you with that. Here is a simple implementation of the selection sort algorithm in Python:

```
def selection_sort(arr):
    for i in range(len(arr)):
        min_idx = i
        for j in range(i + 1, len(arr)):
            if arr[min_idx] > arr[j]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr
```

This algorithm sorts an array by repeatedly finding the minimum element from the unsorted part of the array and putting it at the beginning. The time complexity of this algorithm is $O(n^2)$.

There are many other sorting algorithms available such as bubble sort, insertion sort, merge sort, quick sort, heap sort, and more. Each of these algorithms has its own advantages and disadvantages. You can learn more about them on GeeksforGeeks and Programiz.

I hope this helps!

2. OpenAI Enie (Nov 5, 2023 version, Ernie 4.0)

Query1: <insert the exact wording of any subsequent input into the AI tool>

Output: <insert the complete output from the second query>

3. Github CoPilot (Feb 3, 2024 version)

Query1: <insert the exact wording you input into the AI tool>

Output: <insert the complete output from the AI tool>

4. Google Bard (Feb 2, 2024 version)

Query: <insert the exact wording of your query>

Output: <insert the complete output from the AI tool>

5. Bing AI

Query: write a FFT algorithm

Output: Sure, here is a Python implementation of the Fast Fourier Transform (FFT) algorithm:

```
import numpy as np

def fft(x):
    n = len(x)
    if n == 1:
        return x
    even = fft(x[0::2])
```

```
odd = fft(x[1::2])
T = [np.exp(-2j * np.pi * k / n) * odd[k] for k in range(n // 2)]
return [even[k] + T[k] for k in range(n // 2)] + [even[k] - T[k] for k
    ↪ in range(n // 2)]
```

The FFT is a widely used algorithm for computing the **Discrete Fourier Transform (DFT)** of a sequence. It reduces the number of computations needed from $O(n^2)$ to $O(n \log n)$.¹² The basic idea of the FFT is to apply divide and conquer. We divide the coefficient vector of the polynomial into two vectors, recursively compute the DFT for each of them, and combine the results to compute the DFT of the complete polynomial.

I hope this helps!