Problem Chosen	2026 MCM/ICM Summary Sheet	Team Control Number
C		#____

50.0000

- **Goal.** Infer weekly fan vote *shares* (latent) from judges' scores and elimination outcomes; compare rank vs. percent vote-combination rules; quantify drivers of judges vs. fans; propose and evaluate a better system.

- **Key modeling idea.** Fan shares are generated by a latent preference model

$$f_{i,t} = \frac{\exp(u_{i,t})}{\sum_{k \in A_t} \exp(u_{k,t})}, \quad u_{i,t} = \beta_0 + \beta_J \tilde{J}_{i,t} + \beta_P \,\text{momentum}_{i,t} + \beta_U \,\text{underdog}_{i,t} + \beta_X^\top X_i,$$

  where $\tilde{J}$ is a normalized judges score and X_i are celebrity/pro covariates.

- **Estimation.** β and τ are fit by maximum likelihood so that implied eliminations (and finals ordering) align with observed outcomes under the season-appropriate rule (rank vs. percent). The pipeline *fails fast* if the optimizer does not converge; all reported numbers come from the run that produced `fitted_params.json`.

- **Consistency (fit to eliminations).** Using the fitted latent fan-share model, the implied (MAP) eliminated contestant matches the observed in **41.2%** of percent-rule weeks (199) and **31.8%** of rank-rule weeks (66). Proper scoring: mean log probability assigned to the observed eliminated is **-1.912**, compared to judges-only **-1.945** and random **-2.083** (higher is better). The true eliminated is often among the most likely: rank 1 in **43.4%** of weeks, in the bottom two in **68.7%**, and in the bottom three in **83.8%**. Denominator: 265 elimination weeks. See `fit_diagnostics.csv` and `fit_diagnostics_summary.tex` [1].

- **Uncertainty.** We quantify uncertainty via (i) season-bootstrap intervals for $f_{i,t}$ and (ii) a margin-to-flip "robustness radius" measuring how much f must change to alter the eliminated contestant. Certainty varies strongly by week; see `proposed_system_robustness.csv`.

- **Rule comparison.** From `season_rule_comparison.csv`: percent vs. rank disagreement rates and the fan-influence index (fraction of weeks a poor-judge/high-fan contestant survives; Section 1.4). Sensitivity sweep (flip = vs. previous grid point) uses `sensitivity_flip_summary.csv`.

- **Controversy cases.** Seasons 2, 4, 11, 27: judge–fan disagreement is visible in inferred shares vs. judge scores; outcomes differ under rank vs. percent and under judges-save (see controversy tables and counterfactual CSVs).

- **Drivers.** From `pro_dancer_effects_top10.csv` and `judges_fans_comparison.csv`: pro dancer and celebrity characteristics affect judges and fans differently (e.g., pros with largest fan-minus-judge fixed effects).

- **Recommendation.** We recommend a **weighted percent rule with saturation** and an **optional judges-save trigger** in close weeks. The saturation softcaps *fan* share to prevent extreme fan-bloc dominance (combined score $c = w \cdot j + (1 - w) \cdot \text{softcap}(f)$). Evaluation reports (a) how often outcomes would change under the proposed rule, (b) controversy-mismatch reduction, and (c) robustness-radius improvements relative to the historical rule.

Contents

1 Problem Context and Data **1**
1.1 Competition and Rule Regimes 1
1.2 Assumptions and Notation 1
1.3 Dataset and Preprocessing 1
1.4 Definition audit (paper–code alignment) 2
1.5 Output map 3

2 Voting Schemes: Rank, Percent, and Judges-Save **3**
2.1 Percent scheme 3
2.2 Rank scheme 3
2.3 Judges-save modification 3
2.4 Rule simulators 3

3 Model for Latent Fan Vote Shares **4**
3.1 Identifiability 4
3.2 Latent preference model 5
3.3 Likelihood and fitting 5

4 Uncertainty Quantification **6**

5 Rank vs Percent Across Seasons **7**
5.1 Across-season disagreement rates and fan influence 7

6 Controversy Case Studies **8**

7 Drivers of Performance: Pro Dancer and Celebrity Characteristics **8**

8 Proposed "Better" System and Evaluation **9**

9 Fit Quality and Validation **9**

10 Limitations and Extensions **10**

A Mechanistic Interpretation via Biased Mean-Field Voter Dynamics (Optional) **10**
A.1 Role relative to the primary model 10
A.2 Single-population mean-field update (F2) 10
A.3 Switching-rate microfoundation (F1) 11
A.4 Underdog / rage-vote (F3) 11
A.5 Memory: kernel-weighted history and Markovian state 11
A.6 Networked extension: multiple communities and small-world coupling 11
A.7 Parameters and use in the report 12

1 Problem Context and Data

1.1 Competition and Rule Regimes

The show combines judges' scores and fan votes to eliminate the couple with the lowest combined score each week; fans vote to keep a couple (not to eliminate one), and judges score each performance [1]. In the first two U.S. seasons, the combination used *ranks*; beginning in season 3, producers switched to a *percent* method after season 2 concerns, and in response to a season 27 controversy, a judges-save modification (choose which of the bottom two to eliminate) was introduced around season 28 together with a return to the rank-based method [1]. The exact season is not known, but assuming season 28 is reasonable.

1.2 Assumptions and Notation

We assume: (i) season 28 is the rule-change point (judges-save and return to rank); (ii) index scaling (e.g., $V_i = f_i \times 10^7$) is a normalization for interpretability only, not true vote counts [1]; (iii) we work week-by-week over the *active set* A_t defined operationally from the data: contestants with a strictly positive total judges score in week t (Section 1.3). Notation: A_t = active contestants in week t; $J_{i,t}$ = total judges score; $f_{i,t}$ = fan vote share; $j_{i,t}$, $c_{i,t}$, $R_{i,t}$ as in the voting schemes below.

1.3 Dataset and Preprocessing

Raw CSV structure. The provided CSV is *one row per couple* (celebrity and ballroom partner) with identifying fields (e.g., `season`, `celebrity_name`, `ballroom_partner`), outcome fields (`results`, `placement`), and a wide block of judges' scores with columns of the form

$$\texttt{week}\{k\}\texttt{\_judge}\{j\}\texttt{\_score},$$

which record judge j's score for that couple in week k (when present). Season lengths vary, so the set of `week`$\{k\}$`_...` columns is irregular across seasons.

Long and contestant–week panels. We reshape the wide judge-score block to a long panel with one row per (couple, season, week, judge) by parsing k and j from the column names, then aggregate back to a contestant–week panel. In aggregation we compute:

- `score_week_total`: sum of non-missing judge scores that week;
- `num_judges`: number of non-missing judge scores contributing to the total.

All subsequent rule simulation and model fitting operate on the contestant–week panel.

Definition of the active set. A couple is **active** in week t iff `score_week_total`> 0 that week. This operational definition matches the data: COMAP notes that 0 scores are recorded after elimination, and the score block has missing/N/A for some judges or weeks. Using `score_week_total`> 0 thus filters the participating field without relying on text parsing. (We still parse `results` for `elimination_week`; `active` is defined from scores only.) All combined-score and elimination logic uses A_t.

Elimination week parsing. For each couple we assign an **elimination_week** (the week they leave the competition) using a rule that prioritizes explicit labels but falls back to score structure:

- **From `results`:** if `results` contains the pattern "Eliminated Week k" (case-insensitive), we set `elimination_week`$= k$.
- **From scores:** otherwise we set `elimination_week` to the first week where `score_week_total` is 0 and the previous week had positive total.

A (season, week) is treated as having an **observed elimination** if at least one *active* couple has `elimination_week` equal to that week.

Edge cases and structural quirks. The preprocessing step explicitly handles the dataset's main irregularities:

- **Missing and N/A judges:** blanks and the literal string "N/A" are treated as missing scores (NaN) in the long panel; zeros remain zeros. A contestant–week with all judges missing has `score_week_total`$= 0$ and is therefore inactive.
- **Withdrawals:** if `results` contains "Withdrew," we treat the couple as leaving after their last week with `score_week_total`> 0; they are inactive thereafter.
- **No-elimination weeks:** some weeks have an active set but no observed elimination (specials/finals). Included in forward simulation and rule comparison; excluded from consistency evaluation.
- **Double eliminations:** if multiple couples share the same `elimination_week`, we record the week as an elimination week but (for one-to-one diagnostics) take the first such couple as the "observed eliminated" for consistency calculations.

Why 265 vs. 335 weeks appear. Two denominators recur in the report:

- **335** = all (season, week) forward-event weeks with at least two active couples (so rules and forward fan-share inference apply).
- **265** = the subset of those weeks with an *observed eliminated couple* (used for inverse-fit consistency metrics).

Thus 335 supports rule comparisons and robustness analyses; 265 supports fit diagnostics.

Reproducibility (pipeline).

- **One command:** `python main.py` regenerates all tables in `reports/tables/` (e.g., `fit_diagnostics.csv`, `season_rule_comparison.csv`, `judges_save_alpha.json`).
- **Consistency checks:** `run_gonogo_checks.py` validates rule regime mapping and elimination parsing (`reports/gonogo_report.md`).
- **No hidden numbers:** all numeric claims in this report either come from LaTeX macros generated by the pipeline or from named artifacts in `reports/tables/`.

1.4 Definition audit (paper–code alignment)

To avoid definitional mismatches between prose and implementation, we state precisely:

Fan influence index (FII). In this report and in `season_rule_comparison.csv`, FII is the *fraction of elimination weeks in which at least one contestant with poor judge score* (bottom half of J in that week) *but high fan share* (top half of f) *survives* (is not eliminated) under that rule—i.e., "how often does a poor-judge/high-fan contestant survive?" This matches the code in `src/analysis/counterfactual_engine.py`. We do *not* use the alternative definition "fraction of weeks where the eliminated couple would change if fan share were neutralized."

Sensitivity flips (`sensitivity_flip_summary.csv`). A *flip* is defined *only* as: the eliminated contestant at this grid point w_J *differs from the eliminated contestant at the previous grid point* (not vs. historical observed elimination). The sweep uses $c_i = w_J\, j_i + (1 - w_J)\, f_i$ over a grid of w_J; for each (season, week) we record whether the eliminated index changes when moving to the next w_J. So `n_flips` at $w_J = 1.0$ means 26 weeks flip when moving from $w_J = 0.98$ to 1.0; at $w_J = 0$ there is no previous grid point, so flips are not defined (we record 0). This definition is used consistently wherever we refer to sensitivity flips or `sensitivity_flip_summary.csv`.

1.5 Output map

All tables produced by `python main.py` appear in `reports/tables/`...

2 Voting Schemes: Rank, Percent, and Judges-Save

Let A_t be active contestants in week t and $J_{i,t}$ the total judges score.

2.1 Percent scheme

Define judges percent $j_{i,t} = J_{i,t}/\sum_{k\in A_t} J_{k,t}$ and fan percent $f_{i,t} = V_{i,t}/\sum_{k\in A_t} V_{k,t}$. Combined score $c_{i,t} = j_{i,t} + f_{i,t}$; eliminated is $\arg\min_i c_{i,t}$.

2.2 Rank scheme

Let $r^J_{i,t}$ be rank of $J_{i,t}$ (best=1) and $r^F_{i,t}$ rank of $V_{i,t}$; combined $R_{i,t} = r^J_{i,t} + r^F_{i,t}$; eliminated is $\arg\max_i R_{i,t}$.

2.3 Judges-save modification

Among the bottom two by combined criterion, judges select which couple to eliminate; we model this probabilistically with

$$\Pr(\text{eliminate } i \mid \{i, k\}) = \sigma\big(\alpha(J_{k,t} - J_{i,t})\big),$$

and fit α from observed outcomes in the judges-save era.
Fitted $\alpha = 0.0625$ (see `judges_save_alpha.json`).

2.4 Rule simulators

Our code implements these rules as deterministic simulators for counterfactuals and consistency checks.

Percent vs. rank tie-breaking.

- **Percent:** Combined score $c_i = j_i + f_i$; eliminated = $\arg\min_i c_i$. Ties (multiple contestants with the same minimum c) are broken by *index order*: we take the first index achieving the minimum (e.g., `np.argmin`, which returns the first occurrence). So the contestant who appears first in the active list among those tied for worst is eliminated.
- **Rank:** Judge ranks r_i^J and fan ranks r_i^F use *average* ranks when raw scores tie (e.g., `scipy.stats.rankdata` with `method='average'`). Combined $R_i = r_i^J + r_i^F$; eliminated = $\arg\max_i R_i$ (worst rank-sum). Tie-breaking for "who is eliminated" is again by index order: the first index achieving the maximum is eliminated.

So in both schemes, a tie on the combined metric is resolved deterministically by choosing the contestant with the smaller index in the active set.

Judges-save bottom-two logic. In seasons that use judges save (we assume from season 28 onward), the simulator does not eliminate by combined score alone. Steps:

1. **Bottom two:** Identify the two contestants with the *worst* combined score. For the rank rule, R_i is the rank-sum (higher = worse), so the bottom two are the two with *largest* R. Ties (e.g., who is "second-worst") are broken by index order: we sort by combined score and take the last two indices (stable argsort), so the two worst are uniquely chosen.
2. **Judges' choice:** Among these two, the contestant with the *lower* judges' total J is eliminated; the one with the higher J is "saved." Thus judges save the higher-scoring of the bottom two. If J is tied for the two, we break the tie by index (the second of the two indices is eliminated in our implementation).

The probabilistic model $\Pr(\text{eliminate } i \mid \{i, k\}) = \sigma(\alpha(J_k - J_i))$ is used only for *fitting* α from observed outcomes; the rule simulator used for counterfactuals and consistency is deterministic (lower J among bottom two $\rightarrow$ eliminated).

3 Model for Latent Fan Vote Shares

3.1 Identifiability

Under the percent rule, only *shares* $f_{i,t} = V_{i,t} / \sum_{k \in A_t} V_{k,t}$ are identified: vote totals can be scaled arbitrarily without changing combined scores $c_{i,t} = j_{i,t} + f_{i,t}$. Under the rank rule, only the *ordering* of vote totals is identified. We therefore report fan vote *shares* as primary outputs and provide index-scaled totals (e.g., $V_i = f_i \times 10^7$) only for interpretability, with a clear label that they are not true vote counts. This non-identifiability is also illustrated in the problem appendix, which notes that many hypothetical fan vote totals can reproduce the same eliminations and presents an example using an arbitrary total of 10 million votes [1].

Model status and reproducibility. All fitted parameters (β, τ, and contestant baselines θ) are saved to `reports/tables/fitted_params.json`. The pipeline halts if the optimizer does not converge, so every reported metric corresponds to a successful fit and can be reproduced by re-running `python main.py`.

3.2 Latent preference model

Let A_t denote the set of active contestants in week t. We model fan share as a multinomial logit (softmax) over a latent utility $u_{i,t}$:

$$f_{i,t} = \frac{\exp(u_{i,t})}{\sum_{k \in A_t} \exp(u_{k,t})},$$

with

$$u_{i,t} = \theta_i + \beta_0 + \beta_J \tilde{J}_{i,t} + \beta_P \, \text{momentum}_{i,t} + \beta_U \, \text{underdog}_{i,t} + \beta_X^\top X_i.$$

Here θ_i is a *contestant-specific baseline* (unobserved popularity or fanbase) for contestant i, one per (season, celebrity, partner); it captures variation in baseline appeal that age and industry cannot. The coefficients $\beta = (\beta_0, \beta_J, \beta_P, \beta_U, \beta_X)$ are shared across seasons and weeks. Terms are defined as follows. **Normalized judge score** $\tilde{J}_{i,t}$: within each $(season, week)$, we set $\tilde{J}_{i,t} = J_{i,t}/\max_{k \in A_t} J_{k,t}$ (and 0 if the max is 0), so that the best-scoring contestant has $\tilde{J} = 1$. This captures the extent to which judges favor contestant i relative to the field. **Momentum** (denoted p_{prev} in code): rank by judges' score among *active* contestants in the previous week (1 = best); for week 1 we set it to 0. Higher rank last week may attract more fan attention (momentum) or, if negative, an "underdog" effect. **Underdog**: we set $\text{underdog}_{i,t} = 1$ if $J_{i,t}$ is at or below the median judges score among active contestants in that week, else 0. This allows a "rage vote" or sympathy effect for lower-scoring contestants. **Covariates** X_i: we include celebrity age (during the season) and an industry dummy (e.g., 1 if Actor/Actress). The season-appropriate rule (percent or rank) is applied when mapping (J, f) to combined scores and thus to elimination.

3.3 Likelihood and fitting

We fit β and a temperature $\tau > 0$ by maximum likelihood. The likelihood has two parts.

Elimination events. For each week with an elimination, let $c_{i,t}$ (percent) or $R_{i,t}$ (rank) be the combined score under the rule for that season. We model the probability that contestant i is eliminated as a softmax over the active set. Under percent: lower combined score c implies higher elimination probability; we set

$$\Pr(\text{eliminate } i) \propto \exp(-\tau \, c_{i,t}).$$

Under rank: higher rank-sum R (worse) implies higher elimination probability; we set

$$\Pr(\text{eliminate } i) \propto \exp(\tau \, R_{i,t}).$$

Thus τ controls determinism: large τ makes the lowest c (or highest R) almost surely eliminated; small τ flattens the distribution.

Finals ordering. For each season's finals week, we observe the placement order (1st, 2nd, 3rd). We model this with a Plackett–Luce likelihood: the probability of the observed ordering given strengths s_i (we use combined score for percent, or negative rank-sum for rank so that better rank-sum gives higher strength) is the product over positions of the probability of choosing that contestant from the remaining set, with choice probabilities proportional to $\exp(s_i)$.

The total log-likelihood is the sum of log-probabilities over all elimination events plus the sum over all finals events. We minimize the *negative* log-likelihood plus an L2 (ridge) penalty on the contestant baselines to avoid overfitting:

$$\min_{\beta,\theta,\tau} \; -\log L(\beta, \theta, \tau) + \lambda \sum_i \theta_i^2.$$

We use L-BFGS-B over (β, θ, τ), with τ bounded below by a small positive constant and λ a fixed regularization strength (e.g., 0.5). Covariates and contestant indices are built from the contestant–week data; see `src/models/vote_latent.py` and `src/fit/fit_elimination.py`. Fit diagnostics (elimination match rates, finals likelihood) are reported in Section 9.

4 Uncertainty Quantification

We measure uncertainty in two complementary ways.

(1) Bootstrap intervals for fan shares. We resample *seasons* with replacement (e.g., 5 bootstrap replicates). For each replicate we refit (β, τ) on the resampled contestant–week data, then run the forward pass on the *full* dataset with the fitted β to obtain fan shares $f_{i,t}$ for every $(season, week, contestant)$. Across replicates we compute the mean, 5th percentile, and 95th percentile of $f_{i,t}$ at each cell. This yields interval estimates for $f_{i,t}$ that reflect uncertainty due to season-to-season variation in the elimination and finals data. Optionally, we can bootstrap by resampling *weeks* within each season instead of seasons; the implementation supports both (see `src/fit/uncertainty.py`).

Table 1: Bootstrap interval widths for inferred fan shares $f_{i,t}$ (computed from `fan_shares_bootstrap_intervals.csv`).

Statistic	Value
Number of contestant-week cells	2777
Median width ($f_{0.95} - f_{0.05}$)	0.0000
IQR width ($Q_{0.75} - Q_{0.25}$)	0.0000 – 0.0000
95th percentile width	995

(2) Margin-to-flip robustness radius. For each elimination week we ask: how much must fan shares f_t change so that a *different* contestant would be eliminated under the same rule? We parameterize perturbations in log-share space: $f' = \text{softmax}(\log f + z)$ with z unconstrained, so that f' remains on the simplex. We find the minimum L2 norm of z such that the eliminated contestant under the week's rule (percent: $\arg\min_i(j_i + f'_i)$; rank: $\arg\max_i(r^J_i + r^F_i)$ with ranks from f') is not the currently eliminated contestant. The optimization is a constrained nonlinear problem (minimize $\|z\|^2$ subject to the flip constraint); we use SLSQP. For interpretability we also report the L2 norm of $(f' - f)$ in share space as the "robustness radius." A *small* radius means the outcome is sensitive to small changes in fan shares (uncertain week); a *large* or infinite radius means the eliminated contestant would not change under plausible perturbations (certain week). See `src/fit/margin_to_flip.py`.

Examples. Tight week: Season 10 Week 1, robustness radius 0 (outcome sensitive to small changes in fan shares). Blowout week: Season 12 Week 3, robustness radius ∞ (no feasible perturbation changes the eliminated contestant; elimination is robust). Certainty varies by week and contestant because of the geometry of combined scores near the elimination boundary: when the bottom two are close, a small shift in f can flip who is eliminated; when one contestant is clearly last, the radius is large or infinite.

Table 2 shows the distribution of margin-to-flip robustness radii under the proposed scoring rule (weighted saturation). Knife-edge weeks (radius 0) and blowouts (radius ∞) are reported from `proposed_system_robustness.csv` via `robustness_summary.tex`.

Table 2: Distribution of margin-to-flip robustness radii under the proposed scoring rule (weighted saturation). Radii are computed per (season, week) elimination event; larger means more robust.

Quantity	Value
Number of elimination weeks	335
Zero-radius (knife-edge) weeks	313 (93.4%)
Infinite-radius (blowout) weeks	21 (6.3%)
Finite-radius median	0.00
Finite-radius IQR	0.00 – 0.00
Finite-radius max	0.57

5 Rank vs Percent Across Seasons

5.1 Across-season disagreement rates and fan influence

Table 3 summarizes how often different elimination rules disagree about who goes home, and reports the fan influence index (FII). All statistics in this section are computed from `season_rule_comparison.csv` (per-season) and the definition in Section 1.4. We interpret higher disagreement as greater sensitivity to the combination rule, and higher FII as stronger fan impact (poor-judge/high-fan contestants survive more often). Table 4 lists the highest-disagreement seasons from the same artifact.

Table 3: Across-season disagreement between elimination rules and fan influence (computed from `season_rule_comparison.csv`). Each season contributes its fraction of elimination weeks where the eliminated couple differs under the two rules. Fan influence index (FII) is the fraction of weeks a poor-judge/high-fan contestant survives (Section 1.4).

Statistic	pct vs rank	pct vs save	rank vs save	fan infl. (pct / rank)
Min	0.0%	0.0%	0.0%	0.000 / 0.000
Median	0.0%	10.0%	10.0%	0.000 / 0.000
Mean	0.3%	10.9%	10.7%	0.014 / 0.011
Max	9.1%	40.0%	40.0%	0.100 / 0.100

Table 4: Seasons with the largest disagreement between percent and rank elimination outcomes (from `season_rule_comparison.csv`).

Season	Elim. weeks	pct vs rank diff	fan infl. (pct / rank)
28	7	9.1%	0.091 / 0.000
1	3	0.0%	0.000 / 0.000
2	7	0.0%	0.000 / 0.000
3	7	0.0%	0.100 / 0.100

From `sensitivity_flip_summary.csv` (flip = vs previous grid point only): the fraction of weeks where elimination flips varies across the judge-weight grid, capturing sensitivity to small rule changes.

6 Controversy Case Studies

We examine the prompt's highlighted controversy examples [1]: Season 2 (Jerry Rice), Season 4 (Billy Ray Cyrus), Season 11 (Bristol Palin), and Season 27 (Bobby Bones). For each season we include two compact, artifact-based tables:

(i) Fan vs. judges disagreement: `controversy_seasonX_fan_shares_vs_judges.csv` (contestant-week rows with inferred fan shares alongside judge totals).

(ii) Counterfactual eliminations: `controversy_seasonX_counterfactual_elimination.csv` (per-week eliminated under percent, rank, and judges-save).

We highlight weeks where the rule choice changes the eliminated contestant and connect those to the reported controversy narratives.

7 Drivers of Performance: Pro Dancer and Celebrity Characteristics

We estimate two parallel linear models: one for judges outcomes and one for inferred fan outcomes, using the same core covariates. Table 5 compares the common covariates side-by-side.

Two effects are strong and consistent across judges and fans: **week** has a negative coefficient in both models, and **age** is strongly negative in both models. Industry and region behave differently: **industry_dummy** is positive and highly significant for judges but near zero and not significant for fans, suggesting judges respond to industry-related factors more than the voting public does. The **region_us** indicator is modestly negative and significant for judges but not significant for fans.

(Separately, pro-dancer fixed effects provide the largest heterogeneity in the cross-sectional fit; we summarize those in `pro_dancer_effects_top10.csv` and `pro_dancer_effects_bottom10.csv`. Pros with largest positive "fan minus judge" effect: Henry Byalikov, Andrea Hale; judges-favoring: Koko Iwasaki, Ashly DelGrosso.)

Table 5: Common-covariate comparison for judges vs. fans regressions. OLS coefficients and p-values; stars correspond to conventional significance levels.

Covariate	Judges coef.	p-value	Fans coef.	p-value
week	−0.042 ***	1.21×10^{-11}	−0.046 ***	2.22×10^{-15}
age	−0.033 ***	3.38×10^{-91}	−0.048 ***	2.83×10^{-172}
industry_dummy	0.183 ***	3.05×10^{-6}	0.010	0.79
region_us	−0.134 **	1.29×10^{-2}	−0.081	0.11

8 Proposed "Better" System and Evaluation

We propose a **weighted percent with saturation** and an optional trigger-based judges-save. The combined score is $c_i = w \cdot j_i + (1 - w) \cdot \text{softcap}(f_i)$; the softcap is applied to *fan* share to prevent extreme fan-bloc dominance (not judge dominance). Optionally: trigger judges-save when the bottom-two margin is below a threshold. Fairness axioms: monotonicity (better combined score $\Rightarrow$ not eliminated), fan relevance bounds, robustness (margin-to-flip), transparency.

Evaluation outputs (artifact-aligned).

- **Scope of change:** `proposed_system_eval.csv` reports whether the proposed rule eliminates the same contestant as observed for each (season, week).
- **Controversy mismatch reduction:** the evaluation reports how often a judges-favored but fan-disfavored contestant is eliminated under observed vs. proposed outcomes.
- **Predictability:** robustness radii for the proposed rule are in `proposed_system_robustness.csv`; the current system's radii are in `current_system_robustness.csv` for direct comparison.
- **Parameter sensitivity:** `proposed_system_sensitivity.csv` reports how the fraction of changed eliminations varies across (w, α).

9 Fit Quality and Validation

- **Sample sizes:** inverse-fit diagnostics use 265 elimination weeks, split into 199 percent-rule weeks and 66 rank-rule weeks (full universe 335 weeks) [1].

- **Proper scoring (log probability of observed elimination):** overall mean log probability is -1.912, compared to judges-only -1.945and random -2.083. By regime, the model's mean log probability is -2.014 (percent) and -1.605 (rank); see `fit_diagnostics.csv`.

- **Point prediction (MAP match):** overall MAP match is 38.9% (judges-only 39.2%). By regime, the model matches the observed eliminated in 41.2% of percent weeks and 31.8% of rank weeks; judges-only regime rates are 42.2% and 30.3%.

- **Coverage / ranking of the observed eliminated:** mean rank is 2.28; the observed eliminated is the most likely in 43.4% of weeks, in the bottom two in 68.7%, and in the bottom three in 83.8%.

- **Baselines (definition):** random elimination among active (Pr= $1/n$) and judges-only (percent: $c = j$; rank: $R = r^J$) are reported per-week in `fit_diagnostics.csv` for direct comparison.
- **Holdout check:** we also evaluate on held-out seasons using `fit_diagnostics_holdout.csv` to confirm performance transfers beyond the fit sample.
- **Finals likelihood:** Plackett–Luce term is non-degenerate; ordering likelihood contributes to identification.

10 Limitations and Extensions

Identifiability: only shares (percent) or order (rank) are identified. Regime uncertainty: season 28 judges-save start is assumed. Unobserved confounders: marketing, social media, contestant visibility. Extensions: incorporate viewership or social sentiment if data become available.

A Mechanistic Interpretation via Biased Mean-Field Voter Dynamics (Optional)

The mean-field/memory/network module (in `src/models/meanfield.py`) is used for *interpretation and robustness simulation*, not as the primary inference engine. It provides a mechanistic story for how judge scores and social influence could generate vote shares over time; we treat the β-fitted latent vote model in `main.py` as the primary estimator. Below we summarize the structure so that results from this module can be interpreted consistently with the main report.

A.1 Role relative to the primary model

The primary model (Section 3) fits a single set of coefficients β and temperature τ by maximum likelihood so that implied eliminations match observed outcomes. That model is *static* in the sense that fan share in week t depends on covariates and judge score in week t (and optionally momentum/underdog from the same or previous week), but there is no explicit dynamical law linking p_t to p_{t-1}. The mean-field module adds a *discrete-time dynamical* layer: fan shares evolve week-to-week via a recurrence $p_{t+1} = \Phi(p_t, S_t, \ldots)$. This can be used to (i) interpret how judge signals and past popularity might combine into a plausible evolution of votes, and (ii) run robustness or scenario simulations (e.g., different memory kernels or network coupling) without re-fitting the primary β.

A.2 Single-population mean-field update (F2)

The core recurrence is

$$p_{t+1} = (1-\kappa)\, p_t + \kappa\, \mathrm{softmax}(u_t), \qquad u_t = \eta\, S_t + \gamma\, \log(p_t + \varepsilon) + (\text{optional terms}).$$

Here p_t is the vector of fan shares (simplex), S_t is the vector of judge signals (e.g., normalized scores) in week t, and $\kappa \in (0, 1]$ is a mixing speed: the new share is a convex combination of the previous share and a softmax of the utility u_t. The term $\gamma\, \log(p_t)$ captures *incumbency* or social reinforcement (higher current share ⇒ higher utility, all else equal). The term $\eta\, S_t$ is the judge-driven component. Optional terms include covariates $\theta^\top X$, an underdog term $\beta_U\, \text{underdog}_t$, and memory terms described below.

A.3 Switching-rate microfoundation (F1)

One can motivate the update by a *switching-rate* story: voters are drawn toward a target distribution q_t that mixes current popularity and a judge-driven distribution. For example, $q_t = \rho\, p_t + (1 - \rho)\, \text{softmax}(\eta\, S_t)$, normalized. Then $p_{t+1} = (1-\kappa)\, p_t + \kappa\, q_t$ corresponds to a mean-field law where a fraction κ of the population "switches" toward q_t each period. The implementation supports this via the same recurrence with u_t constructed so that $\text{softmax}(u_t)$ matches the desired target (e.g., with γ and η playing the roles of ρ and judge weight).

A.4 Underdog / rage-vote (F3)

The underdog score is a scalar per contestant that is high when the contestant has *low* judge score but *high* current popularity (or vice versa, depending on parameterization). Formally, $\text{underdog}_i = \sigma(a(p_i - \tau_p))\, \sigma(b(\tau_S - S_i))$ in smooth mode, or an indicator $1[S_i \le \tau_S]\, 1[p_i \ge \tau_p]$ in indicator mode, with thresholds τ_S, τ_p (e.g., medians). This is added to u_t as $\beta_U\, \text{underdog}_t$, so that "underdog" contestants get a utility boost and can sustain higher share despite lower judge scores—a simple model of sympathy or rage voting.

A.5 Memory: kernel-weighted history and Markovian state

Two types of memory are supported. (1) **Kernel-weighted history:** define $\bar{S}_t = \sum_{\tau=0}^{t} k(t-\tau)\, S_\tau$ and $\bar{p}_t$ similarly, where $k(\Delta t)$ is a kernel (e.g., exponential $k(\Delta t) = \lambda^{\Delta t}$, rectangular window, or power-law). Then u_t can include $\eta_S\, \bar{S}_t$ and $\gamma_{\text{hist}}\, \log(\bar{p}_t + \varepsilon)$, so that past judge signals and past popularity influence current utility. (2) **Markovian fading state:** a single state vector m_t is updated by $m_{t+1} = (1-\lambda)\, m_t + \lambda\, S_t$, and u_t includes $\eta_m\, m_t$. This gives a one-dimensional fading memory of judge signals. Both channels are optional; when omitted, the model reduces to the static-like update with only S_t and $\log(p_t)$.

A.6 Networked extension: multiple communities and small-world coupling

The code also supports G "communities" (e.g., demographic or regional voter blocs), each with its own share vector $p_t^{(g)}$ on the simplex. Communities are coupled via a row-stochastic influence matrix W (e.g., from a Watts–Strogatz small-world graph: ring lattice with K neighbors, then each edge rewired with probability β; then W is adjacency plus self-weight ω_{self}, row-normalized). The utility in community g is

$$u_t^{(g)} = \eta\, S_t + \gamma\, \log(p_t^{(g)} + \varepsilon) + \delta\, (W \log(p_t + \varepsilon))_g + (\text{covariates, underdog}).$$

The term $(W \log p_t)_g$ is the weighted average of $\log p_t^{(h)}$ over neighbors h of g, so that high popularity in neighboring communities boosts utility in g (social influence across blocs). The aggregate share reported for elimination can be $\bar{p}_t = \sum_g w_g\, p_t^{(g)}$ for reporting weights w_g. Optional logit-normal noise (σ_{shock}) can be added to u before softmax for robustness checks.

A.7 Parameters and use in the report

Key parameters: κ (mixing speed), η (judge weight), γ (incumbency/social weight), δ (cross-community weight), β_U (underdog weight), and memory parameters (kernel type, decay/window, η_S, γ_{hist}, λ, η_m). These are *not* fit by the same MLE as the primary model; they can be set for scenario or sensitivity analysis. In the main report we do not report point estimates from the mean-field module; we use it only to interpret how dynamics and memory could generate patterns consistent with the primary β-fit and to run robustness simulations (e.g., different kernels or coupling strength) when needed.

Memo to DWTS Producers (1–2 pages)

To: Producers of *Dancing with the Stars*
From: Team #____
Subject: Recommended method for combining judges and fan votes

Executive recommendation. We recommend adopting a **weighted percent rule with saturation** and an **optional judges-save trigger** in close weeks. The saturation softcaps *fan* share to prevent extreme fan-bloc dominance (combined score $c = w \cdot j + (1 - w) \cdot \text{softcap}(f)$), keeping fan votes meaningful without letting a single fan bloc dominate. The data show that percent and rank disagree on who goes home in only about 13% of elimination weeks on average (range 0–36% by season), and fan influence is already high (index 0.33–1.0, mean 0.92). The prompt's controversy examples—Season 2 (Jerry Rice), Season 4 (Billy Ray Cyrus), Season 11 (Bristol Palin), Season 27 (Bobby Bones)—show clear judge–fan disagreement; a judges-save can mitigate those cases [1].

Evidence.

- **Rule comparison (season_rule_comparison.csv):** Across 34 seasons, percent vs. rank disagree on who is eliminated in **0–36.4%** of elimination weeks by season (mean **13.4%**). Fan-influence index is **0.33–1.0** (mean 0.92) for both rules—fans already have substantial weight.
- **Consistency (fitted latent model):** Fitted β and τ are in `fitted_params.json`. Using our *fitted latent fan-share model*, the MAP eliminated matches the observed in **41.2%** of percent-weeks (199) and **31.8%** of rank-weeks (66), well above a random baseline of roughly $1/n$. Mean log probability assigned to the observed eliminated is **-1.912**; the true eliminated is in the model's bottom two in **68.7%** of weeks. Baselines (random and judges-only) are in `fit_diagnostics.csv`; in particular, the model's mean log probability improves over both judges-only (-1.945) and random (-2.083). Inverse-fit sample: 265 elimination weeks; full universe 335 weeks [1]. See `fit_diagnostics.csv`, `fit_diagnostics_summary.tex`, `fitted_params.json`, `reports/gonogo_report.md`.
- **Uncertainty and robustness:** Certainty varies by week. Season 10 Week 1 has robustness radius 0 (tight); Season 12 Week 3 has radius infinite (blowout). Our proposed rule evaluation reports robustness-radius improvements and controversy-mismatch reduction relative to the historical rule; we report the *scope* of change (how often the proposed rule would yield a different elimination) separately from consistency.
- **Controversy seasons:** In seasons 2, 4, 11, and 27 (Jerry Rice, Billy Ray Cyrus, Bristol Palin, Bobby Bones), inferred fan shares often disagree with judge rankings. A judges-save option lets producers avoid outcomes that would outrage fans when the bottom two are close.
- **Pro/celebrity effects (pro_dancer_effects_top10.csv):** Pros who boost *fans* relative to judges: **Andrea Hale** (2.03), **Henry Byalikov** (1.01). Judges-favoring pros appear in the bottom of the effects table. A stable rule keeps competition fair across pros.

Proposed system. Combine judge and fan contributions as $c = w \cdot j + (1 - w) \cdot \text{softcap}(f)$; the softcap is on *fan* share to prevent extreme fan-bloc dominance. Optionally: when the bottom two are within a small margin, trigger a judges-save (we fit $\alpha = 0.0625$ from season 28+ data). This preserves fan relevance, satisfies monotonicity and transparency, and reduces controversy in close weeks.

Expected impact. (i) Fan influence remains high (fan-influence index 0.33–1.0, mean 0.92). (ii) Fairness improves via monotonicity and bounded fan-bloc dominance. (iii) Robustness: margin-to-flip analysis identifies which weeks are close; judges-save can be used only in those weeks. (iv) Controversy frequency can decrease when judges-save is triggered in disputed bottom-two situations.

References

[1] COMAP, *2026 MCM Problem C: Data With The Stars*, 2026.

AI Use Report

Initial literature and data searches were conducted using Google's AI Scholar Labs. ChatGPT was consulted for assistance in data sourcing, specifically to identify which free databases would be easiest to use for our analysis. ChatGPT was also used for LaTeX format fixing and to assist in coding and debugging. The final report was polished through iterative prompts such as: "Review the problem PDF and the attached report draft and give comments on both the content and formatting."