

Jeopardy! Wagering Strategy: A Practical Model for Daily Double and Final Jeopardy Bets

Version 2.2 — 2026-10-13 (Tier 2 now breaks ties in win share by expected winnings instead of taking the smallest wager, so some recommended wagers move and no win share does. Version 2.1 recalibrates the simulated rivals to modern-era play.)

Abstract

This paper documents the research, math, and design decisions behind a wagering-strategy tool for Jeopardy!'s two decision points that carry real stakes: the Daily Double and Final Jeopardy. It covers three things in one place: what's already known about optimal wagering (from a peer-reviewed source and from expert practitioner writing), the exact math this project built on top of that foundation, and which inputs are published, which are calibrated privately, and which are our own assumption. The intent is that a reader can verify every claim here against either a citation or a worked example, not take it on faith.

Correction (2.0.0). Versions before 2.0 scored a tie for first as a full win, on the grounds that co-champions both return. That rule ended with the 24 November 2014 episode: a tie for first after Final Jeopardy now goes to a tiebreaker clue, and only the winner returns (The Ringer, 2021). The engines now score a k-way tie for first as $1/k$ of a win, and nobody wins at \$0 or less. Every 1.x statement that ties count as wins is withdrawn, and every Tier 2 and Daily Double figure computed under it has been removed from this paper. The visible symptom was a leader told to bet \$6,000, a bet that only ties second place, when the \$6,001 cover bet wins outright (section 5.4). Version 2.0 also replaces the 1.x placeholder opponent model (ADR-0002) with simulated contestant behavior (ADR-0003, section 5.3).

Change in 2.1. Version 2.0 took opponent accuracy, the outcome correlation, the leader's cover rate and all Daily Double behavior from Tesauro et al. (2013), whose data end in 2012. Play changed in the 2010s: contestants now hunt Daily Doubles and bet them much harder. Version 2.1 recalibrates those inputs to Seasons 38-41 (2021-2025), privately, on J! Archive season statistics whose terms forbid republication (ADR-0004, section 3). Final Jeopardy accuracy now depends on score rank, and a future Daily Double now falls at a random point among the remaining clues instead of after all of them. Every 2.0 Tier 2 and Daily Double figure is replaced by a rerun of the 2.1 engine. The question each figure answers is the same.

Change in 2.2. A user reported that a solo game ("Just you") at 100% confidence got a Smart Bet of \$0. A solo player wins whatever they bet, so every wager tied on win share, and the 2.1 tiebreak, the smallest wager, chose \$0. Tier 2 now ranks wagers by win share first and money second: among wagers tied on win share, it takes the one with the highest expected winnings, then the smallest (section 5.2). It also tries every breakpoint wager, so Python and the app give identical recommendations. The Daily Double engine applies the money rule to exact ties only (section 6.4). No win share moves. Recommended wagers inside a tied range do: the featured case moves from \$6,001 to \$6,599 at the same 91.3% (section 5.4).

1. Motivation and scope

Jeopardy! has two moments where a contestant chooses a dollar amount rather than just answering a clue: a Daily Double (DD), found while a category is in play, and Final Jeopardy (FJ), at the end of the game. Both decisions are mathematically tractable — the legal wager bounds are fixed by the rules, the scores are known, and the outcome is binary (right or wrong) — which makes them a good target for a model, as opposed to buzzer speed or trivia knowledge, which aren't.

The scope is a 3-player game, since that's the standard format; 2-player and 1-player (practice) games are treated as reduced cases of the same underlying model, not separate logic.

Two distinct questions get asked here, and they are not the same question:

1. **"What wager can't hurt me if I'm right?"** — a guarantee against the worst case, needing no assumption about anyone's actual odds of being correct. This project calls this **Tier 1**.
2. **"Given how confident I actually am, and how real contestants tend to bet and answer, what wager gives me the best real odds of winning?"** — a probability-weighted decision. This project calls this **Tier 2** for Final Jeopardy, and the **Daily Double engine** for the mid-game bet (which additionally has to account for however much of the board is left to play).

2. Methodology: how this was researched

The starting point was a web search for existing quantitative work on Jeopardy! wagering strategy, followed by a deliberate check on where the underlying historical game data could legally come from, since the obvious source — J-Archive, the fan-run historical database at j-archive.com — bans scraping and republication of its content in its own site terms.

That search surfaced three kinds of material, each treated differently:

- **A peer-reviewed academic source:** Tesauro, Gondek, Lenchner, Fan, and Prager (2013), published in the *Journal of Artificial Intelligence Research*. This is the IBM Watson team's own writeup of the statistical and game-theoretic strategy engine built for Watson's Jeopardy! matches. It mined roughly 3,000 historical episodes from J-Archive and published the *resulting aggregate statistics* — not the raw data — which makes those numbers citable without the reuse restriction that applies to scraped data itself. This is the paper this project leans on hardest, because it's the only source that is both rigorous and legally clean to cite from.
- **Expert practitioner writing, not peer-reviewed but influential:** Keith Williams' "The Final Wager" (a tutorial series by a former College Jeopardy! Champion) and Chuck Dupee's 1998 wagering analysis, both of which independently arrived at heuristic rules — the cover bet, the two-thirds rule — that the Tesauro paper later confirmed empirically describe how real contestants tend to bet (imperfectly).
- **Pre-scraped historical datasets on GitHub** (e.g., jwolle1's `jeopardy_clue_dataset`, 538,845 clues with Daily Double wager amounts and round-end scores). These looked, at first glance, like the ideal calibration source. Reading the dataset's own README changed that: it explicitly asks that the data

not be used to build a public-facing product. That single sentence is why this project does not use that dataset (or others like it) anywhere in a shipped artifact — only, if at all, for private internal validation. See `docs/adr/0001-data-source-strategy.md` for the full reasoning.

From there, the project's design followed directly from what the Tesauro paper says about how *they* approached the same problem: rather than compute a game-theoretically perfect equilibrium (which assumes all players are perfectly rational and know it — not true of real contestants, and expensive to compute), they computed a **best-response** strategy against an empirically-fitted model of how real opponents actually wager. This project follows the same two-layer structure: a guarantee-based layer that needs no behavioral assumptions (Tier 1), and a best-response layer that does (Tier 2 / the DD engine). Every behavioral number in the second layer is labeled CALIBRATED (set privately from modern-era season statistics, ADR-0004), PUBLISHED (taken from Tesauro et al., 1990s-2012 play, kept only where no modern source exists) or ASSUMED (our choice where no source gives a number).

3. Data and sourcing decision

Summarized from ADR-0001 (`docs/adr/0001-data-source-strategy.md`), as amended by ADR-0004 (`docs/adr/0004-modern-era-calibration.md`):

Source	Used for	Restriction
J! Archive per-season statistics pages, Seasons 38-41	Private calibration of the CALIBRATED model values; read by hand, one page per season, no scraping	Site terms forbid scraping and forbid republishing data collected or derived from J! Archive
J! Archive game pages	Nothing (not scraped)	Same terms
jwolle1/jeopardy_clue_dataset	Nothing shipped; internal validation only, if used at all	README forbids public-facing product use
J! Archive's own Wagering Calculator	Reference point to sanity-check against	N/A — a tool, not a data source
Tesauro et al. (2013), JAIR	PUBLISHED values cited in this paper (1990s-2012 play)	None — published research

Decision: this paper, the app and the deck cite only the Tesauro et al. published statistics. The modern-era source statistics are not kept in the repository and are not quoted, tabulated or paraphrased as numbers in any document. This paper describes the CALIBRATED values in words only; the method is in `engine/behavior.py` and ADR-0004. The values themselves are model settings in the engine and the app, which is an accepted terms-of-use risk (section 8).

4. Final Jeopardy, Tier 1: guarantee-based wagering

4.1 The question this answers

Given the scores of up to three players entering Final Jeopardy, what wager guarantees the best possible finish, assuming the worst realistic case: every opponent bets their entire legal maximum and answers

correctly?

This needs no input about confidence, category, or opponent psychology — it's a minimax guarantee, not a probability estimate.

4.2 The math

Let a player's current score be `S`. The legal maximum Final Jeopardy wager is:

```
legal_max(S) = max(S, 0)
```

(A player at zero or a negative score has no wager available — there's nothing to stake.)

For any player, their **worst-case ceiling** — the highest score they could possibly reach — is what happens if they bet everything and answer correctly:

```
ceiling(S) = 2S    (if S > 0)
ceiling(S) = S     (if S <= 0, since no wager is possible)
```

For a player with several opponents, the **binding constraint** is whichever opponent has the single highest ceiling. This is the key simplification that makes the model work cleanly for three players instead of needing pairwise patchwork: every opponent can reach their own ceiling in the same game, so the true worst case for you is exactly "every opponent simultaneously achieves their own individual best outcome," and beating the largest of those ceilings automatically beats all the smaller ones too.

```
binding_ceiling = max(ceiling(opponent_1), ceiling(opponent_2), ...)
```

Lockout: if your current score already exceeds `binding_ceiling`, no opponent can catch you regardless of the Final Jeopardy outcome. The correct wager is \$0 (or anything — it doesn't change the outcome).

Cover bet: if you're not in lockout, the minimum wager that guarantees an outright win (assuming you answer correctly) is:

```
required = binding_ceiling - your_score + 1
```

capped at your legal maximum. If `required` exceeds your legal maximum, an outright guarantee is not possible — the model reports that plainly rather than pretending otherwise.

Lock-tie: a special case. If your score is *exactly half* of the binding opponent's score, and that opponent is themselves already safe against everyone else (so they have no reason to wager anything), then betting everything and answering correctly guarantees at least a tie for first. Under the current rule a tie for first goes to a tiebreaker clue, so a lock-tie guarantees a place in the tiebreaker, not a win. It matters only when no outright cover is possible: then it is the best guarantee Tier 1 can offer. Tier 1's premise that the leader

bets nothing is a worst-case convenience, not a prediction; section 5.4 shows what the lock-tie is worth against simulated opponents.

4.3 Worked examples

These are the exact hand-derived cases used to test the implementation (`tests/test_fj_wager.py`), shown here so the math can be checked directly:

- **A = \$12,000, B = \$5,000.** `ceiling(B) = 10,000`. Since $12,000 > 10,000$, A is in lockout. Correct wager: **\$0**.
- **A = \$9,000, B = \$5,000.** `ceiling(B) = 10,000`. Not lockout. `required = 10,000 - 9,000 + 1 = 1,001`. Correct wager: **\$1,001**.
- **A = \$10,000, B = \$6,000, C = \$3,000.** `ceiling(B) = 12,000`, `ceiling(C) = 6,000`. B is the binding constraint (larger ceiling), not C, even though this is a three-player game. `required = 12,000 - 10,000 + 1 = 2,001`. Correct wager: **\$2,001**. (Note this is not a pairwise patchwork of "beat B" and "beat C" separately — it's a single formula that happens to reduce to the pairwise answer here, because one opponent's ceiling dominates the other's.)
- **A = \$10,000, B = \$5,000, C = \$1,000** (from B's perspective). B's binding opponent is A, with `ceiling(A) = 20,000` — B cannot cover that even betting everything (\$5,000 max). An outright guarantee isn't possible. But: A's score (10,000) already exceeds `ceiling(C) = 2,000`, so A has no need to wager anything against C. And B's score is exactly half of A's ($5,000 \times 2 = 10,000$). This is a lock-tie: betting everything (\$5,000) and answering correctly ties A for first and forces a tiebreaker if A bets nothing.

4.4 Corollary: mathematically irrelevant opponents

A direct consequence of section 4.2's "binding constraint" logic: an opponent whose ceiling never exceeds the current binding ceiling has *no effect whatsoever* on the correct wager. If a third player is far enough behind that even doubling their score plus winning couldn't reach the threshold set by whoever the actual binding opponent is, that player can be dropped from the calculation entirely with no loss of correctness — the model already behaves this way, since it only ever looks at `max(ceiling(...))` across opponents, never a sum or an average.

This means a "mathematically eliminated" rival doesn't need special-case handling in the underlying math — they're automatically ignored. What the current tool does *not* yet do is *tell the user* this explicitly at the interface level (e.g., "Rival 2's score doesn't affect this wager"), which would be a natural, low-risk usability enhancement. It's logged as a backlog item rather than built, since confirming this was a design question, not a math gap.

5. Final Jeopardy, Tier 2: confidence-weighted best response

5.1 The question this answers

Given an actual confidence level (not worst-case paranoia) and a model of how real contestants wager and answer, what wager maximizes the expected share of a win?

Tier 2 scores a game the way the show does. A player's result is a **win share**: 1 for an outright win, $1/k$ for a k -way tie for first, 0 otherwise. Nobody wins if the top score is \$0 or less. Giving each tied player an equal $1/k$ chance in the tiebreaker clue is an assumption: in reality buzzer skill decides it.

Tier 2 can recommend a different wager from Tier 1 for the same scores. The reason is the confidence input and the opponent model, not the tie rule: Tier 1 assumes every opponent bets everything and is right; Tier 2 weighs what opponents actually tend to do.

5.2 The math

Win share of one wager. Let S be the user's score, w the wager, p the user's confidence. Final Jeopardy outcomes are correlated across players (a hard clue is hard for everyone), so we use a **one-factor Gaussian copula**. Each player i has a latent value

$$Z_i = \sqrt{r} * F + \sqrt{1 - r} * E_i, \quad F, E_i \sim N(0, 1) \text{ independent}$$

and answers right when $Z_i < \Phi^{-1}(p_i)$. F is the shared clue difficulty. Given $F = f$, players are independent, with

$$p_i(f) = \Phi(\Phi^{-1}(p_i) - \sqrt{r} * f) / \sqrt{1 - r})$$

The correlation is CALIBRATED to modern-era play (ADR-0004). It is lower than the value Tesauro et al. (2013, section 2.3) report for 1990s-2012 games. We read it as the correlation of the binary right/wrong outcomes and convert it to the latent correlation with Sheppard's formula (Sheppard, 1899), $\text{rho_binary} = (2/\pi) * \arcsin(r)$, so

$$r = \sin(\text{rho_binary} * \pi / 2)$$

The formula is exact at 50% accuracy and approximate elsewhere (section 8). Each opponent's p_i is the CALIBRATED accuracy for their score rank.

The expected win share is

$$E[\text{share} | w] = \text{integral } \phi(f) * [p(f) * E[\text{share}(S + w) | f] + (1 - p(f)) * E[\text{share}(S - w) | f]] df$$

Inside the bracket, the engine enumerates every combination of each opponent's (wager $\times$ right/wrong) branch, weights it by its probability given f , and records the best opponent score and how many opponents share it. The user's share against that is 1 (strictly ahead and above \$0), $1/(m+1)$ (tied with m opponents at a positive top score), or 0.

Integration. The integral over f is a trapezoid rule on $[-6, 6]$ with step 0.25 (49 nodes), with the normal density weights normalized to sum to 1. There is no sampling, so there is no sampling error. The result is computed in IEEE-754 double precision. On the one hand-derived correlated case (section 7, which uses an illustrative binary correlation of 0.3 as an explicit input), the quadrature result differs from the exact value by $1.4e-10$.

Search. The engine evaluates every \$100 from \$0 to the legal maximum, plus the legal maximum, the Tier 1 cover wager and one dollar below it, the Tier 1 lock-tie wager, and every breakpoint. For own score S and each possible opponent final score f , the breakpoints are $f - S$, $f - S + 1$, $S - f$ and $S - f - 1$: the wagers that tie or just pass f when right, and that tie or just stay above f when wrong. Win share changes only at these points, so the best wager in each flat stretch is always tried.

Choice: win share first, money second. Among wagers whose win share is within $1e-9$ of the best, the engine recommends the one with the highest expected winnings, then the smallest. Expected winnings are the final score times the win share, $E[\text{share}(S \pm w) \times (S \pm w)]$, because the winner keeps their total; they are compared to six decimal places. A solo player, or a leader in a lockout, wins whatever they bet, so only the money separates the wagers. Before 2.2 the engine took the smallest tied wager, which told a solo player at 100% confidence to bet \$0.

Version 1.x also had a "strict win" secondary objective, added to stop the engine preferring a tie bet over a cover bet. It is removed in 2.0. Under the tiebreaker rule a tie is already worth less than a win, so the objective function itself prefers the cover bet.

5.3 The opponent model and its honest limits

All values below live in one configuration block in `engine/behavior.py` (mirrored in the `BEHAVIOR` object in `app/index.html`). Full decision record: `docs/adr/0003-simulated-contestant-behavior.md`, with the modern-era recalibration in `docs/adr/0004-modern-era-calibration.md`. Each opponent is given a role by score: leader (A), second (B), third (C).

CALIBRATED values are described in words only. Their source statistics cannot be republished (section 3); the method is in `engine/behavior.py` and ADR-0004.

Quantity	Value	Status
B wager mix	all-in 26%; keep C out (bet below $B-2C$, or cover with $2C-B+1$) 27%; overtake A ($A-B+1$) 15%; two-thirds ($3B-2A-1$) 8%; random 24%	PUBLISHED from 1990s-2012 games, for the regime $B \geq 3/4 A$ and $B \geq 2C$ (T13 section 2.3). No modern source exists. ASSUMED to hold in the other five regimes and in the modern era.

Quantity	Value	Status
A wager mix, not locked out	the cover bet 2B-A+1 most of the time; small shares for the tie-cover 2B-A, all-in and a random bet	Share of bets that cover second place: CALIBRATED (leaders almost always cover). The split between bet types: ASSUMED.
A wager mix, lockout (A > 2B)	\$0 50%; largest safe bet A-2B-1 50%	ASSUMED.
C wager	random	PUBLISHED that C's bets do not depend on the game state (T13 section 2.3, note 5; 1990s-2012). The flat shape is ASSUMED.
"Random" bet	equal weight on 0, 1/4, 1/2, 3/4 and all of the score	ASSUMED.
Missing bet types	weight shared out in proportion among the bet types that exist	ASSUMED.
Opponent FJ accuracy	by score rank entering Final Jeopardy: the leader is right more often than second place, and second place more often than third	CALIBRATED.
Outcome correlation	positive, across all players including the user; lower than the 1990s-2012 value in T13	CALIBRATED. The binary-scale reading is ASSUMED.
Tiebreaker odds	equal, 1/k	ASSUMED.

T13 is Tesauro et al. (2013). Example: for the scores \$10,000 / \$8,000 / \$3,400, the model gives B (\$8,000) this wager distribution: \$8,000 30.8%, \$1,199 27%, \$2,001 15%, \$3,999 8%, and \$0, \$2,000, \$4,000, \$6,000 at 4.8% each. C (\$3,400) bets \$0, \$850, \$1,700, \$2,550 or \$3,400 with 20% each.

Honest limits:

- Five of the six B regimes, A's split between bet types and C's shape are our assumptions, not fitted values.
- The published B weights come from groups of only a few hundred games each (T13 section 2.3), so even they carry wide error bars. They describe 1990s-2012 play; second place's wager mix, the largest behavioral component in Final Jeopardy, has no modern source.
- The correlation is read as a binary correlation and converted with Sheppard's formula, which is exact only at 50% accuracy. Modern average accuracy is below 50%, so the conversion is approximate.
- The user's own confidence p is self-reported and is not calibrated.

5.4 Worked examples

Hand-derived: a tie bet is worth less than a cover bet. Self \$9,000, one opponent at \$5,000 who bets \$0 or \$5,000 with probability 0.5 each. Both players are right 50% of the time, independently. The opponent

finishes at \$5,000 (0.5), \$10,000 (0.25) or \$0 (0.25).

- Tie bet, \$1,000. Right: \$10,000 beats \$5,000 and \$0 (0.75) and ties \$10,000 ($0.25 \times 1/2 = 0.125$), so 0.875. Wrong: \$8,000 beats \$5,000 and \$0, so 0.75. Win share = $0.5 \times 0.875 + 0.5 \times 0.75 = \mathbf{0.8125}$.
- Cover bet, \$1,001. Right: \$10,001 beats every outcome, so 1. Wrong: \$7,999, so 0.75. Win share = $0.5 \times 1 + 0.5 \times 0.75 = \mathbf{0.875}$.

The extra dollar is worth 6.25 points of win share. Under the 1.x rule both bets scored 0.875; that equality was the root of the defect.

The cover bet is not the only wager worth 0.875. Every wager w from \$1,001 to \$3,999 wins when right ($9,000 + w$ is above \$10,000) and, when wrong, lands on \$5,001 to \$7,999: above \$5,000 and \$0, below \$10,000, so 0.75. At \$4,000 a wrong answer ties \$5,000 and the win share falls to 0.75. Within the tied range, expected winnings are $0.5 \times (9,000 + w) + 0.375 \times (9,000 - w)$, which rise by \$0.125 for each dollar of wager. Tier 2 therefore recommends **\$3,999**; 2.1 recommended \$1,001, the smallest tied wager. Tier 1 says \$1,001. The tie bet, \$1,000, is still worth only 0.8125. This case is

`test_tie_bet_is_worth_less_than_cover_bet` in `tests/test_fj_bestresponse.py`.

Shipped defaults: the reported case. Leader at \$10,000, opponents at \$8,000 and \$3,400, leader 89% confident, all 2.1 defaults (simulated opponents, accuracy by score rank, calibrated correlation).

Wager	Win share
\$0	69.7%
\$6,000 (ties B if both double)	84.9%
\$6,001 (Tier 1 cover)	91.3% (0.912976)
\$6,599 (recommended)	91.3% (0.912976)
\$6,600	91.0%
\$10,000 (all-in)	89.0%

Tier 1 says \$6,001. The 1.x engine said \$6,000, a bet that only ties B; 2.0 and 2.1 said \$6,001. In 2.2, Tier 2 recommends **\$6,599**. Every wager from \$6,001 to \$6,599 has the same win share, 0.912976: each one covers B's doubled \$16,000, and a miss still leaves at least \$3,401, above C's \$3,400. At \$6,600 a miss leaves \$3,400, which can tie C, and the share drops to 91.0%. Within the tied range a larger wager wins more money when right, so the money tiebreak picks the top of the range. The JavaScript app gives the same \$6,599 and 91.3%. With the correlation set to 0, Tier 2 also recommends \$6,599, at 90.4%.

The same scores at 50% confidence give a different answer: Tier 2 recommends \$2, at 74.0% (the \$6,001 cover is worth 58.6%). A \$2 bet that is right reaches \$10,002, which clears the \$10,000 and \$10,001 finishes that some simulated B wagers produce (B betting \$2,000, or the \$2,001 overtake bet, and answering right), without risking much on a coin-flip clue. Every tried wager from \$2 to \$800 scores 74.0%. Here expected winnings fall slightly as the wager grows (\$7,396.99 at \$2, \$7,396.30 at \$800), so the smallest tied wager is also the money choice. The 2.1 engine said \$100, because its \$100 grid did not try \$2.

The lock-tie against simulated opponents. B at \$5,000 against A at \$10,000 and C at \$1,000 (section 4.3). Tier 2 also recommends the \$5,000 lock-tie wager, but it is worth much less than a guarantee: 20.8% at 50% confidence and 41.9% at 89% (betting \$0 gives 4.9%). The reason is that the simulated leader rarely bets nothing here: A is not locked out ($A = 2B$ exactly), so A usually makes the \$1 cover bet. B's all-in then wins only when A misses.

6. Daily Double wagering: Monte Carlo simulation

6.1 Why this needs simulation, not a closed form

Unlike Final Jeopardy, a Daily Double decision has consequences that unfold over the *rest* of the game, not just the next thirty seconds: there may be more clues to play, possibly another hidden Daily Double, and then Final Jeopardy after all of that. There's no clean closed-form "cover bet" formula for this, because the future is genuinely uncertain in a way Final Jeopardy — a single terminal event — is not. This project's approach is to simulate the rest of the game forward, many times, for each candidate wager, and see which one has the highest average win share.

6.2 What's modeled, named explicitly

Every simplification is named, with its status:

- **Regular clues.** Each remaining clue is "won" by a uniformly random player with probability 0.6. ASSUMED placeholder. The model does not simulate buzzer competition, and a wrong answer does not pass the clue to another player.
- **Future Daily Doubles.** Each one falls at a uniformly random point among the remaining clues, because modern players find Daily Doubles early. The finder is drawn by score rank, with the leader the most likely finder. The finder wagers a fixed, round-specific fraction of their score and is right at a fixed rate. Finder weights, wager fractions and accuracy are CALIBRATED (ADR-0004). Real finders choose their wager from the game state; this model does not.
- **Final Jeopardy inside a rollout.** The advised player uses the fast Tier 1 rule (lockout \$0, else lock-tie, else cover, else all-in) at the CALIBRATED all-player accuracy by default; running the full Tier 2 search in every trial would be too slow. Opponents draw a wager from the simulated behavior in section 5.3, at the CALIBRATED accuracy for their score rank. Right/wrong outcomes are drawn from the same copula and correlation as Tier 2.
- **Result.** Each rollout returns the advised player's win share (1, $1/k$ or 0) under the tiebreaker rule.

6.3 A methodological fix worth documenting: common random numbers

An early version of this simulator compared candidate wagers using independently drawn random numbers for each one. That's a subtle bug, not just an inefficiency: with a finite number of simulated trials, pure sampling noise can make a genuinely worse wager look better purely by luck, especially when two wagers are close in real quality.

The fix is a standard variance-reduction technique called **common random numbers**: for each simulated trial, the *same* underlying random sequence (who wins each remaining clue, whether a future Daily Double lands, every player's Final Jeopardy wager and correctness) is replayed for every candidate wager being compared in that trial. Only the wager itself differs between candidates within a trial — the "luck" of the rest of the game is held fixed. This turns a noisy independent comparison into a fair, paired one, and it's what makes the engine's basic sanity checks — for example, "if you're guaranteed to answer wrong, the optimal wager is \$0" — hold reliably instead of only approximately.

6.4 Selecting a wager from noisy estimates

Common random numbers (6.3) reduce sampling noise between candidates; they do not eliminate it. Every candidate wager's simulated win share is still a Monte Carlo *estimate*. Selecting a wager from these estimates is a statistical decision, and this project got it wrong twice before getting it right. The history is in `CHANGELOG.md` (1.1.0 and 1.3.0) and `docs/reviews/statistical-review-2026-07-23-v2.md`; the dollar figures quoted there were computed under the 1.x tie rule and opponent model and are not repeated here.

- **First error.** The original engine took the bare argmax of noisy estimates, so a two-point change in stated confidence could swing the recommendation across the whole wager range.
- **Second error.** The first fix compared wagers with the *independent* two-sample standard error $\sqrt{\text{se}_w^2 + \text{se}_{\text{best}}^2}$. Under common random numbers the estimates are strongly positively correlated, and the correct variance of the difference is $\text{Var}(p_w) + \text{Var}(p_{\text{best}}) - 2\text{Cov}(p_w, p_{\text{best}})$. Dropping the covariance gave a spuriously wide plateau and systematic under-betting.

Current method. The comparison standard error is the **paired** per-trial difference between a candidate wager and the empirical best, estimated directly from the per-trial win shares. Because k candidate wagers are each compared against the single empirical best, this is a *multiple-comparisons-with-the-best* problem (Hsu, 1996); the band is the Bonferroni value $z^* = \Phi^{-1}\{1 - \alpha/k\}$ with $\alpha = 0.05$. A wager is on the plateau if $p_{\text{best}} - p_w \leq z^* \times \text{SE}_{\text{paired}}(w)$. The recommendation is the smallest wager on the plateau.

Exact ties (2.2). Some wagers have the same outcome as the recommendation on every simulated trial; a solo game, which is always won, is the plain case. No amount of simulation can separate them on win share, so the engine picks by money: the Daily Double adds $(2p - 1) \times \text{wager}$ on average, so when $p > 0.5$ it recommends the largest such wager. Otherwise the plateau rule is unchanged.

Current results (Monte Carlo estimates; 2.2 engine, rerun 2026-10-13; every figure is the same as under 2.1):

Scenario	Trials per wager, seed	Recommended	Best estimate
Clear signal: \$3,000 vs \$4,000 vs \$2,000, Double Jeopardy, 100% confident, nothing left before FJ, \$300 steps (11 candidates, $z^* = 2.609$)	800; seeds 1, 2, 3	\$3,000 (all-in) on every seed; plateau is \$3,000 alone	70.0%, 70.2%, 68.5%

Scenario	Trials per wager, seed	Recommended	Best estimate
Reported case: \$4,200 vs \$6,000 vs \$3,400, Double Jeopardy, four \$1,000 clues and one DD left, \$100 steps (43 candidates, $z^* = 3.045$), 42% confident	3,000; seed 42	\$0 (plateau: 14 of 43 wagers, between \$0 and \$4,200)	\$4,100 at 25.7% (\$4,200: 25.5%; \$0: 23.5%)
Same, 44% confident	3,000; seed 42	\$0 (plateau: 7 of 43 wagers, between \$0 and \$4,200)	\$4,100 at 26.5% (\$4,200: 26.4%; \$0: 23.5%)
Same, 42% and 44%	300; seed 42	\$0 at both (plateau: all 43 wagers)	—

What these numbers do **not** prove: the reported-case runs show only that the engine cannot separate \$0 from a near all-in bet at 3,000 trials when the estimated gap is about 2 to 3 points; they do not show that \$0 is the better bet. Each row is one seed (three for the clear-signal row), so they do not measure seed-to-seed stability beyond that. No row has been checked against real games. All figures depend on the placeholder clue rate and the CALIBRATED future-DD and Final Jeopardy values in section 6.2; a different calibration would move them.

One honesty note: the reported "best estimate" is the in-sample maximum over candidate wagers, so it is a mildly *optimistic* (selection-biased) estimate. The recommended *wager* is well insulated from this, because it is the smallest wager on a multiplicity-controlled plateau, not the raw argmax (risk #11 in `RISK_REGISTER.md`).

6.5 Legal wager bounds

A Daily Double wager may be up to the greater of the player's current score or the maximum clue value showing on the board for that round: \$1,000 in Single Jeopardy, \$2,000 in Double Jeopardy, under the rules current as of this writing.

7. Validation

Method. 49 automated Python tests (`cd tests && python -m pytest -q`, 49 passed on 2026-10-13): 9 for Tier 1, 11 for Tier 2, 11 for the behavior model, 12 for the Daily Double engine, 6 for the rollout simulator. A separate harness, `node tests/js_parity.js`, runs 27 checks against the JavaScript port in `app/index.html` (27 passed on 2026-10-13). CI runs both on every push and pull request.

- **Exact parts (Tier 1, Tier 2, scoring, behavior).** Expected values are derived by hand and written in the test comments, not captured from the code's output. Examples: the tie bet 0.8125 vs cover bet 0.875 (section 5.4), with the money tiebreak choosing \$3,999 inside the tied range; a two-way tie worth 1/2 and a three-way tie 1/3; nobody wins at \$0; a third-place player at 100% confidence against two leaders with fixed wager distributions wins exactly 1/12 (only when both leaders bust) and bets all \$4,000, because every wager ties on win share; a solo player bets everything at 100%

confidence and \$0 at 30%; a locked-out leader who is sure bets everything; the published B weights reproduced in their regime; opponent accuracy that follows score rank; the frequency with which each rank finds a future Daily Double; the future-DD wager fraction by round; and a correlated case with a closed form. That case uses explicit inputs, not the model defaults: self \$9,000 bets \$1,001, opponent \$5,000 always bets all, both 50%, and an illustrative binary correlation of 0.3 (an input chosen for the hand derivation, not the model's calibrated value). Then $P(\text{both wrong}) = 0.25 + 0.3 \times 0.25 = 0.325$, so the win share is $0.5 + 0.325 = 0.825$; the quadrature gives 0.825 within $1.4e-10$.

- **Shipped-default regression.** For \$10,000 / \$8,000 / \$3,400 and \$12,000 / \$9,000 / \$3,000 at 89%, Tier 2 must not recommend the tie-only bet 2B-A.
- **Monte Carlo parts (Daily Double).** These cannot be hand-derived, so the tests check invariants: legal bounds; guaranteed-wrong recommends \$0; a guaranteed-correct player is never penalized for betting more; determinism under a fixed seed; probabilities in $[0, 1]$; paired standard error never above the independent formula; the Bonferroni band widens with more candidates; a clear-signal case recommends a positive wager, identical across seeds 1, 2 and 3; among exactly tied wagers, a solo player who is likely right bets the most.
- **JavaScript parity.** The exact Tier 2 values, win-share rule, behavior weights, correlation check, shipped-default regression and statistical reference values match Python. Python and JS use different random number generators (Mersenne Twister vs. mulberry32), so Daily Double outputs are checked only for invariants, never trial for trial.

What the tests do not prove. They prove the code computes the model as specified. They do not prove the model matches real contestants: no recommendation or win share has been checked against the outcomes of real games. The published and calibrated inputs come from season-level aggregate statistics, not from a fit to individual games, and the assumed inputs have no empirical check at all. The tests check that the calibrated values are wired in correctly; they cannot check them against their source, which is not in the repository.

8. Known limitations

In order of how much they matter:

1. **Much of the opponent model is assumed** (section 5.3): five of six B regimes, A's split between bet types, C's shape, the random grid, and equal tiebreaker odds. Treat Tier 2 and Daily Double win shares as model outputs, not validated statistics.
2. **Second place is still pre-2012.** Second place's wager mix, the largest behavioral component in Final Jeopardy, comes from 1990s-2012 games (Tesauro et al., 2013), on small samples. No modern source exists.
3. **Terms-of-use risk, accepted (ADR-0004).** The CALIBRATED values are model settings visible in `engine/behavior.py` and in the app's public source. A strict reading of J! Archive's terms could treat rounded model settings as data derived from J! Archive. We publish no statistics, tables or quotes, and would remove the values on a takedown request.
4. **Readers cannot reproduce the calibrated values.** The method is described; the source data are not shipped, so a reader cannot check a CALIBRATED value against its source.

5. **Score rank by proxy.** Who finds a Daily Double is reported by lectern, not by score rank. The returning champion's lectern stands in for the leader.
6. **The correlation conversion is approximate.** Sheppard's formula reads the correlation as binary and is exact only at 50% accuracy. Modern average accuracy is below 50%.
7. **The Daily Double simulator's rest-of-game model is simplified** (section 6.2): no buzzer competition, no pass-through, a placeholder clue rate, and future-DD wagers that are a fixed fraction of score rather than a response to the game state. Trust it most for the very next decision.
8. **The reported Daily Double win share is selection-biased upward** (section 6.4, risk #11). The recommended wager is well insulated.
9. **The JavaScript app is a manually maintained port** with a different random number generator; the parity harness covers the exact functions only (risk #8).
10. **Earlier figures are not comparable or are superseded.** Any win percentage from a version before 2.0 scored ties as wins against a different opponent model. Every 2.0 Tier 2 and Daily Double figure is superseded by the 2.1 recalibration. Tier 2 recommended wagers from 2.1 and earlier used the smallest-tied-wager rule; 2.2 replaces them (the win shares are unchanged).
11. **This tool is a study and post-game analysis aid, not a live-assist device.** Using it during an actual taping would violate the show's rules.

9. References

- Tesauro, G., Gondek, D., Lenchner, J., Fan, J., & Prager, J. M. (2013). Analysis of Watson's strategies for playing Jeopardy!. *Journal of Artificial Intelligence Research*, 47, 205–251. <https://arxiv.org/pdf/1402.0571>
- The Ringer (2021, January 23). Article on the Jeopardy! tiebreaker scenario and its rule changes (title not reproduced here). Source for the November 2014 change from co-champions to a tiebreaker clue. <https://www.theringer.com/2021/01/23/tv/jeopardy-tiebreaker-scenario-rules-changes>
- Sheppard, W. F. (1899). On the application of the theory of error to cases of normal distribution and normal correlation. *Philosophical Transactions of the Royal Society A*, 192, 101–167. (Binary-to-latent correlation, section 5.2.)
- Hsu, J. C. (1996). *Multiple Comparisons: Theory and Methods*. Chapman & Hall. (Multiple-comparisons-with-the-best, section 6.4.)
- Williams, K. "How to wager in Final Jeopardy!" *The Final Wager*. <https://thefinalwager.wordpress.com/tutorial/>
- Dupee, C. (1998). Wagering strategy analysis for Final Jeopardy!, as cited and summarized in Tesauro et al. (2013).
- jwolle1. jeopardy_clue_dataset [dataset]. GitHub. Referenced for its reuse restriction, not used as a data source in this project. https://github.com/jwolle1/jeopardy_clue_dataset
- J! Archive Wagering Calculator. <https://j-archive.com/wageringcalculator.php>
- J! Archive. Per-season Daily Double and Final Jeopardy! statistics, Seasons 38-41. Used for private calibration only; the site's terms of use forbid republication, so no figure from it appears in this paper. <https://j-archive.com/>

Reproducing the figures

Every figure in sections 5 and 6 comes from the shipped engine, run from `engine/`:

```
python -c "import fj_bestresponse as br; r = br.analyze([9000, 5000], 0, 0.5,
opponent_model_fn=lambda s, i: [(0, 0.5), (5000, 0.5)], opponent_correctness=0.5,
correlation=0.0); d = dict(r.wager_win_probabilities); print(r.recommended_wager,
r.win_probability, d[1000], d[1001], d[4000])"
python -c "import fj_bestresponse as br; r = br.analyze([10000, 8000, 3400], 0, 0.89); d =
dict(r.wager_win_probabilities); print(r.recommended_wager, r.win_probability, [d[w] for w
in (0, 6000, 6001, 6599, 6600, 10000)])"
python -c "import fj_bestresponse as br; r = br.analyze([10000, 8000, 3400], 0, 0.89,
correlation=0.0); print(r.recommended_wager, r.win_probability)"
python -c "import fj_bestresponse as br; r = br.analyze([10000, 8000, 3400], 0, 0.5); d =
dict(r.wager_win_probabilities); print(r.recommended_wager, r.win_probability, d[6001], [w
for w, p in r.wager_win_probabilities if p >= r.win_probability - 1e-9])"
python -c "import behavior; print(behavior.wager_distribution([10000, 8000, 3400], 1),
behavior.wager_distribution([10000, 8000, 3400], 2))"
python -c "import fj_bestresponse as br; [print(p, br.analyze([10000, 5000, 1000], 1,
p).recommended_wager, dict(br.analyze([10000, 5000, 1000], 1, p).wager_win_probabilities)
[5000], dict(br.analyze([10000, 5000, 1000], 1, p).wager_win_probabilities)[0]) for p in
(0.5, 0.89)]"
python -c "import dd_wager; from simulate import RolloutConfig as C; [print(s, (r :=
dd_wager.analyze([3000, 4000, 2000], 0, 2, 1.0, config=C(), wager_step=300,
trials_per_wager=800, seed=s)).recommended_wager, r.win_probability, r.plateau) for s in
(1, 2, 3)]"
python -c "import dd_wager; from simulate import RolloutConfig as C; r =
dd_wager.analyze([4200, 6000, 3400], 0, 2, 0.42, config=C(remaining_clue_values=[1000]*4,
remaining_dd_count=1), trials_per_wager=3000, seed=42); d =
dict(r.wager_win_probabilities); print(r.recommended_wager, r.win_probability,
len(r.plateau), d[0], d[4200])"
```

Change `0.42` to `0.44`, and `3000` to `300`, for the other Daily Double rows.

Related project documents

- `docs/adr/0001-data-source-strategy.md` — the data-sourcing decision
- `docs/adr/0003-simulated-contestant-behavior.md` — the opponent model and tie rule (supersedes ADR-0002)
- `docs/adr/0004-modern-era-calibration.md` — the private modern-era calibration and its accepted risks (amends ADR-0001 and ADR-0003)
- `docs/adr/0002-opponent-model-assumptions.md` — the superseded 1.x placeholder model, kept for history
- `docs/explanation/wagering-theory-research.md` — the original research memo
- `docs/explanation/tiers-explained.md` — a plain-language companion to the Tier 1/Tier 2 discussion

- `docs/reviews/statistical-review-2026-07-23-v2.md` — the review behind section 6.4's current method
- `docs/tutorial/`, `docs/how-to/`, `docs/reference/` — task-oriented documentation (Diátaxis)
- `RISK_REGISTER.md` — tracked risks referenced in this paper
- `CHANGELOG.md` — the record of every change