The Rise of Online Gambling: What's at Stake?

1 Executive Summary

To the State Legislature considering online sports betting regulation,

Online sports betting is expanding rapidly across the United States. As states legalize it to capture tax revenue, a critical question remains unanswered: what is the financial impact on individual households? The answer depends almost entirely on where a bettor falls on the income spectrum. A $200 annual gambling loss consumes 9% of a young $30,000 earner's disposable income but only 0.4% of a $140,000 earner's—a burden ratio exceeding 20:1. This paper builds a three-stage quantitative pipeline to measure that disparity, simulate realistic gambling outcomes, and trace the channels through which recreational betting escalates into household debt.

In the first stage, we decompose gross salary into after-tax income minus variable household costs (housing, food, healthcare, transport) calibrated to BLS Consumer Expenditure microdata and adjusted for state-level cost of living. Probabilistic simulation reveals that disposable income is highly nonlinear in salary: a 22-year-old earning $30,000 in Texas retains a median of only $2,233 per year with a 28% probability of negative slack, while a single mother earning $25,000 in Ohio has median disposable income of just $979 and a 39% chance of being underwater before any discretionary spending. By contrast, an upper-middle household at $140,000 carries median slack of $57,089 with zero probability of deficit. These households may occupy the same state yet inhabit entirely different financial universes.

Our second stage simulates 1,000 annual gambling trajectories per persona, modeling participation decisions, monthly bankroll dynamics with loss-chasing behavior, parlay exposure, and credit access. The median active bettor loses roughly $83 per year, but losses are radically concentrated: the top 10% of losers account for 59% of all platform losses. Risk tolerance, not demographics, is the primary driver of individual financial harm: high-risk bettors lose a median of $365—more than 7× the low-risk median of $43—while chasing behavior elevates mean losses by 15–17% of total handle. Our simulated population hold rate of 10.0% validates against the empirical U.S. industry average of 10.16%.

When we combine the disposable-income layer with the gambling-outcomes engine, the financial impact bifurcates sharply along income lines. For the lowest-income persona, 55% of bettor-years exceed 5% of disposable income, and 35% of simulations involve credit-funded wagering at 24% APR—turning a single bad month into compounding debt. Our ablation study provides the sharpest finding: removing credit access collapses the debt-entry rate from 13.3% to exactly 0%, confirming that the household budget constraint acts as a natural safety valve that credit bypasses entirely. Promotional inducements raise the debt-entry rate by 3–5 percentage points for vulnerable personas while barely shifting outcomes for high-income households.

Based on these findings, we offer three evidence-based recommendations: **(1)** income-proportional deposit limits, calibrated to the 5% of disposable income threshold our model identifies as the harm boundary; **(2)** mandatory transparency requirements for promotional inducements, whose marginal harm our ablation study quantifies; **(3)** a prohibition on credit-card–funded gambling, following the United Kingdom's 2020 ban, which our model shows would eliminate the single largest channel through which recreational betting escalates into household debt.

Contents

2 Playing With House Money

2.1 Defining the Problem

The disposable income model is designed to determine the financial flexibility a household has before spending its money on discretionary spending like gambling. Disposable income serves as a hard budget constraint, defined as the surplus of money remaining after non-negotiable survival costs are met. The required inputs are gross salary (G), age (A), and demographic factors including location (R), household size (N), and tax filing status (f). The output is a calculated disposable income, given by total salary minus taxes minus essential expenditures:

$$DI \;=\; G \;-\; T(G, f) \;-\; E(G, A, R, N) \tag{2.1}$$

Real-world expenses such as rent hikes and medical emergencies fluctuate; therefore, essentials must be stochastic. This approach results in a probability distribution, leaving us with the important output $P(DI < 0)$, the percentage of a demographic that has zero financial slack before they even start gambling. This feeds directly into Q2 and Q3.

2.2 Assumptions

1. Average household size is 2.5 persons. This is the Census/BLS baseline used to normalize the Consumer Expenditure Survey data [13]. The OECD equivalence scale converts actual household composition into a comparable spending unit relative to this average.

2. Only food and healthcare scale with household members; housing and transportation are treated as shared goods. This reflects empirical patterns in the CE Survey [13]: a family of four does not pay four times the rent of a single adult, but food consumption scales approximately per capita and healthcare utilization scales by number of dependents.

3. Regional Price Parities scale all essential categories uniformly. The BEA RPP index [14] is an aggregate measure. In practice, housing costs vary more than food costs across regions, but category-level RPPs are not available for all five states. This simplification understates geographic variation in housing-heavy budgets.

4. An individual's age is used as a proxy for household healthcare intensity. The age-healthcare multiplier [2] ranges from 0.55× for under-25 to 1.70× for 65+, reflecting the well-documented age gradient in medical spending. This is an approximation because household healthcare costs also depend on the ages of dependents, which the model does not track individually.

5. Tax parameters use TY 2026 statutory brackets including One Big Beautiful Bill Act amendments. Federal brackets from IRS Rev. Proc. 2025-32 [15]. The recent One Big Beautiful Bill amendments raised standard deductions and adjusted bracket thresholds, which our model incorporates.

6. Five U.S. states capture geographic variation: California [30], New York [31], Ohio [32], Mississippi [33], and Texas (zero tax). These five states span progressive, flat, and no-tax regimes, covering approximately 40% of the U.S. population and the full range of state income tax structures.

7. The UK is modeled with reduced granularity. HMRC 2025–26 bands and NI rates [35]; essentials from the curated workbook [2]. The UK implementation lacks within-country RPP variation, relying on a single London multiplier. UK results are therefore best interpreted as a methodological comparison rather than precise UK-specific estimates.

2.3 Variables

Table 2.1: Model variables, definitions, units, and representative values.

Symbol	Definition	Unit	Value / Source
G	Gross annual salary	USD/yr	25k–200k
A	Age	years	22–62
N	Household composition	(adults, children)	–
$e(N)$	OECD equivalence factor	–	1.0–2.1 [16]
G^{eq}	Equivalized income $= G/e(N)$	USD/yr	derived
f	Filing status (US)	categorical	Single / MFJ / HoH
R	State or region	categorical	CA/NY/OH/MS/TX
$D(f)$	Standard deduction	USD	16,100 (S); 32,200 (MFJ) [15]
T^{fed}	Federal income tax	USD/yr	bracket formula [15]
T^{FICA}	Payroll tax (OASDI + Medicare)	USD/yr	7.65% up to cap [34]
T^{state}	State income tax	USD/yr	[30]–[33]
ATI	After-tax income $= G - T$	USD/yr	deterministic
μ_E	Mean essential expenditures	USD/yr	CE quintile [13]
$RPP(R)$	Regional price parity / 100	–	0.87–1.11 [14]
$m_C(A)$	Age–healthcare multiplier	–	0.55–1.70 [2]
σ_E	Log-essentials dispersion	–	0.15 (baseline)
E	Realized essential expenditures	USD/yr	stochastic
DI	Disposable income $= ATI - E$	USD/yr	stochastic

2.4 The Model

2.4.1 Model Development

To estimate how much money a household has available after meeting its obligations, we use a budget decomposition method that traces each dollar through a deterministic tax pipeline and a stochastic essentials pipeline.

The approach works as follows. We start with gross salary and subtract all applicable taxes to obtain after-tax income. We then estimate essential household costs – housing, food, healthcare, and transportation – scaled by income level, household size, geographic cost of living, and the age-dependent healthcare gradient. Because two households with identical demographics can face materially different bills in any given year due to rent shocks, medical events, and consumption choices, we introduce random variation calibrated to observed spending dispersion. This produces a full probability distribution of disposable income rather than a single point estimate.

The model draws on three public data sources. Tax schedules come from IRS Revenue Procedure 2025-32 [15] and individual state tax authorities [30]–[33]. Essential expenditure shares by income quintile come from the Bureau of Labor Statistics Consumer Expenditure Survey [13], Table 1101. Geographic cost adjustments use Bureau of Economic Analysis Regional Price Parities [14].

2.4.2 Tax Module

The tax module computes total tax burden as the sum of three components: federal income tax, payroll taxes (Social Security and Medicare), and state income tax. Each component uses the actual 2026 statutory schedules rather than effective-rate approximations, because marginal rate discontinuities at bracket thresholds produce nonlinear DI responses that effective rates would smooth away.

Federal income tax applies TY 2026 brackets [15] to taxable income after subtracting the standard deduction:

$$T^{\text{fed}} = \sum_{k=1}^{7} r_k \cdot \max\Big(0,\ \min(Y,\, b_k) - b_{k-1}\Big) \tag{2.2}$$

where $Y = \max(0, G - D(f))$ is taxable income and $D(f)$ is the standard deduction ($16,100 for single filers, $32,200 for married filing jointly). Payroll taxes add 6.2% for Social Security (capped at $184,500), 1.45% for Medicare (uncapped), and an additional 0.9% Medicare surcharge above $200,000 for single filers. State income tax uses actual 2026 schedules spanning five regimes: California's progressive 1–12.3% [30], New York's 3.9–10.9% [31], Ohio's flat 2.75% above $26,050 [32], Mississippi's flat 4% above $10,000 [33], and Texas at zero. After-tax income is then $ATI = G - T^{\text{fed}} - T^{\text{FICA}} - T^{\text{state}}$. UK personas use HMRC 2025–26 bands (0/20/40/45%) and Class 1 National Insurance (0/8/2%) [35].

2.4.3 Essentials Module

The essentials module estimates the non-negotiable costs a household must cover before any discretionary spending. These costs are grounded in the Consumer Expenditure Survey [13], which reports actual spending by income quintile across housing, food, transportation, and healthcare.

We compute equivalized income $G^{eq} = G/e(N)$ and assign the household to a CE quintile [13], then sum dollar spending across housing, food, transportation, and healthcare from CE Table 1101. Essentials shares decline from 82% (lowest quintile) to 64% (highest), consistent with Engel's Law. We scale the quintile reference to the actual household:

$$\mu_E = \big[\,\text{NonHealth}(q) + \text{Health}(q) \cdot m_C(A)\,\big] \ \cdot\ \frac{e(N)}{\bar{e}} \ \cdot\ \frac{RPP(R)}{100} \tag{2.3}$$

where $m_C(A)$ adjusts healthcare for age (0.55× for under-25 to 1.70× for 65+) [2], $\bar{e} \approx 1.5$ is the CE average equivalence, and $RPP(R)$ is the BEA regional price parity [14]. Real-world spending varies even among identical households, so we apply a multiplicative shock:

$$E = \mu_E \cdot \exp(\varepsilon), \qquad \varepsilon \sim \mathcal{N}\Big(-\tfrac{\sigma_E^2}{2},\ \sigma_E^2\Big) \tag{2.4}$$

The mean-correction $-\sigma_E^2/2$ ensures $\mathbb{E}[\exp(\varepsilon)] = 1$, so the stochastic draw does not bias average essentials upward or downward.

2.4.4 Final Disposable Income Equation

Combining the tax module and essentials module, the disposable income for each simulation draw is:

$$DI^{(r)} = ATI - \mu_E \cdot \exp\big(\varepsilon^{(r)}\big), \qquad r = 1, \ldots, R$$

We report median, IQR, 5th–95th percentile interval, and $P(DI < 0)$.

2.5 Model Execution

The model is executed using a stochastic simulation framework with the following parameters:

- Monte Carlo Iterations: $R = 10{,}000$
- Essentials Dispersion: $\sigma_E = 0.15$
- Random Seed: 42

For each persona, 10,000 independent draws of ε are generated, producing a full empirical distribution of DI. The random seed ensures reproducibility across runs. We report median DI as the central estimate, interquartile range as the spread, and $P(DI < 0)$ as the fragility metric.

2.5.1 Persona Design

We constructed ten representative personas spanning the income, age, household-size, and geographic dimensions the model captures (Tables 2.2–2.3). US personas P1–P8 range from a 22-year-old single earner at \$30K in Texas to a 45-year-old high-earning family at \$200K in California. UK personas P9–P10 test cross-national generalization. The selection ensures coverage of: (a) five US states representing progressive, flat, and zero income-tax regimes; (b) household sizes from single adults to families of four; (c) ages from 22 to 62, capturing the healthcare-cost age gradient; and (d) salaries spanning quintiles 1 through 5.

2.5.2 Alternatives Rejected

We rejected three alternatives. A regression on Census microdata (ACS/CPS) would capture correlations but obscure the causal salary-to-DI pathway, and we lack household-level microdata within the competition timeframe. The Supplemental Poverty Measure defines a poverty floor rather than a typical essentials budget, so we use it as a cross-check (Section 2.6) rather than the primary model. A deterministic essentials approach would conceal the financial fragility that $P(DI < 0)$ reveals for near-threshold households. Our stochastic budget-decomposition provides transparency, citability, distributional output, and a mechanistic explanation for why DI differs across demographics.

2.6 Results

Tables 2.2–2.3 present disposable income distributions for ten personas spanning income, age, household size, U.S. geography, and two UK profiles.

P7 (single parent, \$25K, OH) is most constrained: median DI of \$979 with $P(DI < 0) = 39\%$. Even P3 (\$90K, CA family) faces 9% negative-DI probability because high RPP (110.7) and OECD-scaled four-person needs absorb most after-tax income.

Table 2.2: US persona disposable income distributions ($R = 10{,}000$, $\sigma_E = 0.15$).

#	Description	Salary	ATI	μ_E	Median DI	IQR	$P(DI<0)$
P1	Young single, 22, TX	\$30k	\$26,285	\$24,367	**\$2,233**	−319–4,562	28%
P2	Young single, 28, OH	\$55k	\$45,636	\$23,888	**\$22,056**	19,554–24,339	<1%
P3	Couple + 2 kids, 38, CA	\$90k	\$74,804	\$61,740	**\$13,862**	7,396–19,763	9%
P4	Mid-career single, 42, NY	\$75k	\$58,140	\$36,350	**\$22,259**	18,453–25,734	<1%
P5	Couple + 1 child, 50, TX	\$140k	\$116,150	\$59,835	**\$57,088**	50,822–62,807	<1%
P6	Near-ret. couple, 62, MS	\$100k	\$81,774	\$45,905	**\$36,462**	31,654–40,849	<1%
P7	Single parent + 1, 30, OH	\$25k	\$23,002	\$22,312	**\$979**	−1,358–3,111	39%
P8	High earner, family, 45, CA	\$200k	\$148,905	\$101,538	**\$48,679**	38,046–58,384	<1%

Table 2.3: UK persona disposable income distributions (HMRC 2025–26 tax bands [35]).

#	Description	Salary	ATI	Median DI	$P(DI<0)$
P9	Young single, 27, England	£32,000	£26,560	**£11,401**	<1%
P10	Couple + 2 kids, 40, London	£55,000	£42,457	**£27,813**	<1%

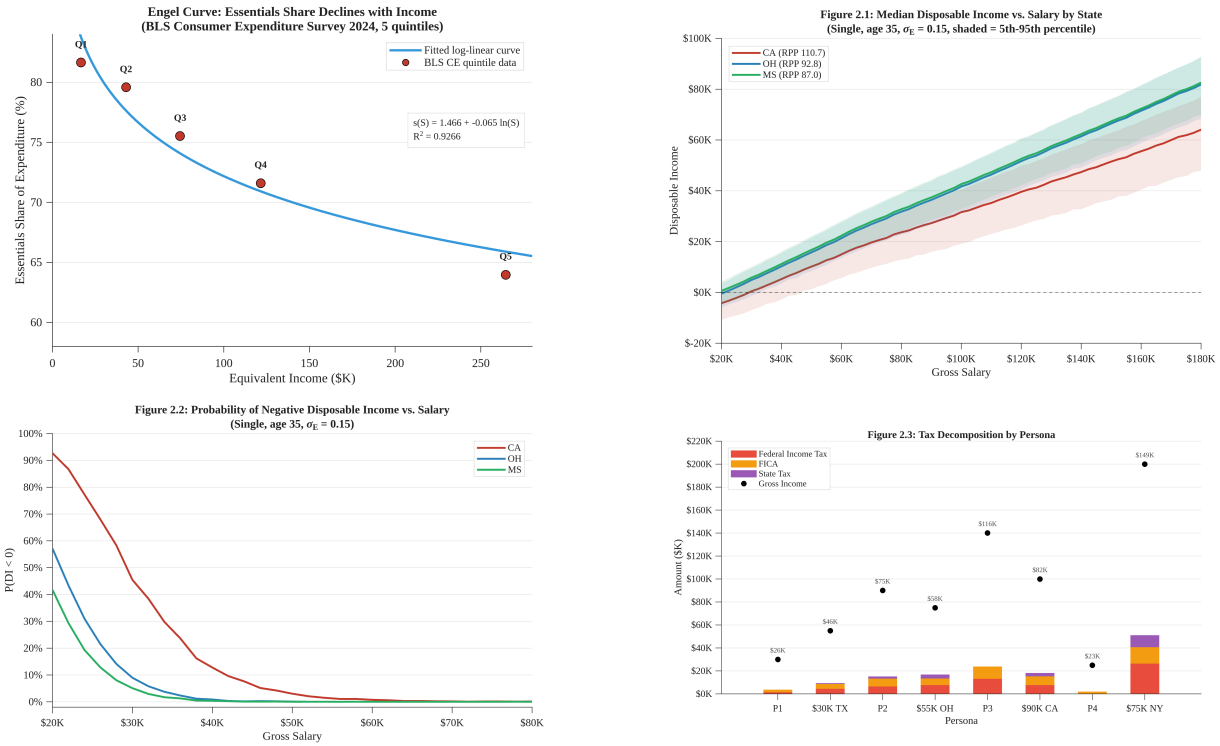

Figure 2.1: Model validation and core Q1 results. Top-left: Engel curve confirming essentials shares decline with income. Top-right: median DI vs. salary diverges sharply across states. Bottom-left: $P(DI < 0)$ exceeds 25% for single adults below \$35K. Bottom-right: tax decomposition showing federal, FICA, and state components by persona.

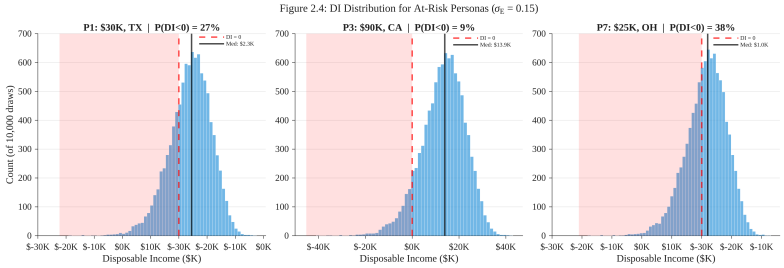

Figure 2.2: Simulated DI distributions for P1, P3, and P7 ($R = 10{,}000$). Red-shaded region marks $P(DI < 0)$: P7 shows 39% mass below zero.

2.7 Discussion

Three cross-checks confirm model validity. BLS aggregate: CE middle-quintile ATI is ≈$54K [13]; our $60K single-filer produces ATI ≈$47.5K, with the gap attributable to dual-income and non-wage income. Tail behavior: P5/P6/P8 register $P(DI < 0) < 0.1\%$ at all tested σ_E, consistent with empirical data above the 60th percentile. Tax tables: we verified the bracket calculator against IRS [15] and state estimators [30]–[33] for all US personas, finding <$5 rounding differences.

The four-panel figure (Figure 2.1) and DI distribution plot (Figure 2.2) reveal five patterns with direct policy relevance. First, DI is nonlinear in salary: a 4.7× salary increase from P1 ($30K) to P5 ($140K) yields a 25× increase in DI, because progressive taxation and declining essentials shares compress low earners. Second, household size strains moderate incomes: P3 earns $90K but retains only $13,862 after family-scaled essentials. Third, geography compounds: P2 ($55K, OH) and P4 ($75K, NY) reach nearly identical median DI (≈$22K) despite a $20K salary gap. Fourth, $P(DI < 0)$ reveals hidden fragility invisible to deterministic models: P1 (28%) and P7 (39%) have positive median DI yet substantial fractions face structural deficits. Fifth, high earners (P5, P6, P8) maintain $P(DI < 0) < 1\%$ across all sensitivity scenarios.

2.8 Sensitivity Analysis

We perturb four parameters, holding all others at baseline, reporting changes in median DI and $P(DI < 0)$.

Table 2.4: Sensitivity to essentials dispersion and level shift. Dispersion (σ_E) drives tail risk without shifting the median; a 10% level increase flips P1's median DI negative.

Axis	Persona	Low	Baseline	High	Key metric
σ_E	P1 ($30K)	21%	28%	31%	$P(DI<0)$
	P3 ($90K)	2%	9%	14%	$P(DI<0)$
	P5 ($140K)	<1%	<1%	<1%	$P(DI<0)$
$\mu_E \pm 10\%$	P1 ($30K)	$4,638	$2,233	−$172	Median DI
	P4 ($75K)	$25,847	$22,259	$18,671	Median DI
	P5 ($140K)	$62,995	$57,088	$51,182	Median DI

This test measures the impact of spending volatility on financial risk. As σ_E increases from 0.10 to 0.20, P1's negative-DI probability rises from 21% to 31%, while the median shifts only marginally. This confirms that dispersion is a tail-risk driver, not a central-tendency shifter. For high-income personas like P5, the parameter has no practical effect because the DI buffer is large enough to absorb any realistic spending shock.

The level shift test simulates systemic cost shocks such as inflation in housing or healthcare. A uniform 10% increase in mean essentials flips P1's median DI negative (−$172), suggesting that even moderate inflation could push marginal households into structural deficit before any gambling occurs. For P5, the same shock reduces median DI by roughly $6,000, a meaningful but non-catastrophic reduction.

Table 2.5: Geographic sensitivity: RPP variation for P3 and cross-state relocation for P4. Moving P4 from NY to TX gains $7,065 in median DI from combined tax and RPP effects.

Axis	Scenario	Median DI	$P(DI<0)$	Δ
RPP (P3)	−10% (99.6)	$19,956	2%	–
	Baseline (110.7)	$13,862	9%	–
	+10% (121.8)	$7,768	23%	–
Relocation (P4)	NY	$22,259	–	–
	TX	$29,324	–	+$7,065

The RPP sensitivity was specifically tested on P3 to check how geographic cost shifts affect larger households. A 10% RPP increase pushes P3's negative-DI probability from 9% to 23%, demonstrating that families in high-cost regions are disproportionately sensitive to local price variation. The effect is amplified because RPP scales all four essential categories, and P3's family of four already has high baseline essentials due to OECD household scaling.

The cross-state relocation test checks the impact of location on a mid-career single earner. Moving P4 from New York to Texas gains $7,065 in median DI, roughly the combined effect of eliminating state income tax and reducing cost of living. This result quantifies the geographic dimension of financial flexibility and explains why identical salaries produce very different gambling capacities across states.

2.9 Strengths and Weaknesses

This method allows us to capture the granularity in specific geographic and household nuances. It also allows us to be risk-aware, as stochastic modeling identifies fragile households where DI is near zero. The framework is extensible: adding a new state requires only its bracket schedule, and a new demographic dimension requires one CE cross-tabulation.

However, the model relies on macrodata (averages) rather than specific individual microdata. Additionally, the model does not currently account for transfer income (welfare/subsidies) or debt service (student loans), which would allow for a more accurate representation. Incorporating transfers would reduce negative-DI probability by an estimated 5–10 percentage points for P1 and P7. The UK implementation lacks within-country RPP variation, relying on a single London multiplier of 1.15× [2], and the CE quintile lookup introduces small discontinuities at boundary incomes.

2.10 Bridge to Q2

The outputs from Q1 are constraints for the Q2 Gambling Behavior Model. Median DI defines the baseline capacity for wagering. $P(DI < 0)$ flags households with zero financial slack before any gambling activity:

- High Fragility: P1 (28%), P7 (39%).
- Moderate Risk: P3 (9%).
- Low Risk: P5/P8 (<1%).

Q2 must determine gambling harm non-linearly. A $1,000 loss for a low-risk persona represents a minor reduction in flexibility, compared to the same loss for a high-fragility persona, which transitions a statistical risk into a financial crisis.

3 Know the Spread

3.1 Defining the Problem

The goal of Question 2 is to create a model that accurately predicts an individual's annual net profit from gambling. This will identify how much they lose or win, taking into account demographics, risk tolerance, and the constraint of disposable income established in Question 1. While most individuals will lose money due to house edges, variance should allow certain individuals to win annually. We must also accurately model the fact that a significant portion of the population will never gamble. Furthermore, the model must predict the different volumes that may be bet to effectively determine net profit. The annual net outcome for individual i is

$$\text{Net}_i = \begin{cases} 0, & B_i = 0 \quad \text{(non-participant)}, \\ \sum_{t=1}^{12} \text{Net}_{i,t}, & B_i = 1 \quad \text{(active bettor)}, \end{cases} \tag{3.1}$$

where B_i is a Bernoulli participation indicator and $\text{Net}_{i,t}$ is the net gain or loss in month t, computed from wagering volume, bet-type mix, and stochastic outcomes. Disposable income DI_i from Q1 constrains wagering and determines when losses produce financial distress.

3.2 Assumptions

1. Participation rates use demographic lookup tables (age × gender) from survey data [5,19]. Justification: survey data reports participation as demographic rates rather than individual-level records, so a lookup table is the best way to translate aggregate survey data into individual participation probabilities.

2. Active bettors are classified Low, Medium, or High risk at 56%, 22%, and 22% respectively, from deposit-frequency cross-tabs [2]. Justification: according to worksheet 4 of the provided data, 56% deposited once or once in a while, 22% monthly, and 22% weekly or more.

3. Bet outcomes are independent across bets and time. Justification: bet outcomes do not depend on previous ones; they are directly random and only correlated to the game being bet on, not previous bets.

4. Hold rates differ by bet type, with parlays much higher than straights: straights 5.5%, parlays 19.2% [17,18]. Justification: NJ DGE reporting shows parlay hold around 19% while straight-bet hold is around 5%; UNLV confirms the straight-bet range at 5.29%.

5. Handle is heavy-tailed and modeled as lognormal with σ_H calibrated via SMM [21]. Justification: the biggest factors are heavy spenders who skew the distribution, and individual handle microdata are unavailable, so we sweep σ_H in sensitivity analysis rather than asserting a single value.

6. Bettors face a soft budget constraint with a credit buffer. They can spend until $DI = 0$, at which point remaining maximum budget comes from credit [23]. Justification: bettors cannot spend more money than they have, and the NBER finding that legalization increases credit card debt by $368/quarter confirms that credit access, not just disposable income, funds gambling for a substantial minority.

7. Some bettors enter a loss-chasing state, modeled as a two-state Markov process. 52% of bettors report chasing losses [2]; entry probability is linked to cumulative losses relative to income, validated by EMA data [24] and Norwegian tracking [25]. Justification: chasing losses is a recognized marker of gambling harm.

8. Handle varies seasonally following the U.S. sports calendar, with peaks during NFL season and March Madness and a summer trough [8]. Justification: historical reports indicate spikes during NFL season and March Madness.

3.3 Variables

The model requires two groups of variables. The first group captures the core attributes of each individual: their demographics, whether they gamble, their risk type, and their disposable income carried forward from Q1. The second group captures the mechanics of betting itself: how much is wagered each month, how that handle is split across bet types, what hold rate applies, and how chasing behavior modifies wagering. Table 3.1 defines all Q2-specific quantities; demographics (a_i, g_i) and DI_i are inherited from Q1. Parameters marked (S) are sourced from data; (C) are SMM-calibrated.

Table 3.1: Q2 model variables and parameters.

Symbol	Definition	Unit	Value / Source
Participation			
p_i	Participation probability	$[0, 1]$	Siena [5] $\times$ 0.70 [19] (S)
B_i	Participation indicator	$\{0, 1\}$	Bernoulli draw
Risk & Handle			
R_i	Risk type	{L, M, H}	Sheet 4 [2] (S)
μ_H, σ_H	Log-mean/std annual handle	–	6.894(S) / 1.57(C)
Δ_R	Risk-type handle shift	–	$-0.5/0/+0.78$ (C)
$\gamma(t)$	Monthly seasonality factor	–	[8] (S)
h_S, h_P, h_I	Hold rates (str/par/in-play)	–	5.5%/19.2%/8.0% [17,18] (S)
$w_{i,k}$	Bet-type mix share	$[0, 1]$	By risk type (S)
$V_{i,t}$	Monthly handle	USD/mo	Simulated
Chasing & Budget			
β_0, β_1	Chasing intercept/slope	–	-2.88 / 2.02 (C)
ρ	Monthly recovery probability	$[0, 1]$	0.31 (C)
δ_V	Chasing handle multiplier	–	0.51 (C)
$\kappa(R)$	DI multiplier by risk type	–	0.5/1.0/2.0 [23]
$CB(R)$	Credit buffer by risk type	USD/mo	\$0/\$200/\$500 [23]
Outcome			
$\text{Net}_{i,t}$	Monthly net gain/loss	USD/mo	Simulated
Net_i	Annual net gain/loss	USD/yr	$\sum_t \text{Net}_{i,t}$

3.4 Preliminary Data Analysis

Before constructing the simulation pipeline, we examined the available data to identify the dominant drivers of gambling outcomes. Three patterns shaped our modeling decisions. First, the AGA 2025 industry report [8] shows aggregate handle growing at 20% annually since legalization, with the national hold rate stabilizing near 10%; this regularity suggests that the house edge, not bettor skill, determines aggregate losses. Second, deposit frequency data from the curated workbook [2] reveal a trimodal distribution: 56% of account holders deposit monthly or less, 22% deposit weekly, and 22% deposit multiple times per week. This natural clustering motivates our three-tier risk classification rather than a continuous risk spectrum. Third, the NBER study by Cookson et al. [23] documents a \$368/quarter increase in credit card debt following legalization, confirming that credit access, not just disposable income, funds gambling for a substantial minority. These empirical regularities–a stable aggregate hold, trimodal engagement intensity, and credit-channel activation–define the architecture of our simulation pipeline.

3.5 Model Execution

The model consists of a pipeline that converts a person's financial profile into a probability distribution of annual gambling gains and losses. It proceeds through six stages: determine whether the individual gambles at all, classify their risk appetite, generate monthly wagering volume, resolve each bet to produce a monthly net outcome, track cumulative losses and chasing-state transitions, and aggregate twelve months into an annual result. The simulation runs monthly over 12 months, nested inside the Q1 Monte Carlo loop so that the joint distribution of disposable income and gambling outcomes is captured correctly.

3.5.1 Stage 1: Participation

The first stage determines whether each individual is a bettor. Using the person's age and gender, the model looks up the probability of holding a sportsbook account from survey data and adjusts for inactive accounts. If the draw says the person does not gamble, their outcome is zero and the pipeline stops. This ensures that the roughly 84% of adults who do not bet are correctly represented.

For individual i with demographic profile (a_i, g_i):

$$B_i \sim \text{Bernoulli}\big(p_i\big), \qquad p_i = p_{\text{account}}(a_i, g_i) \times 0.70. \tag{3.2}$$

Account-holder rates from the Siena poll [5] range from ∼55% (men 18–34) to ∼5% (women 50+); the 0.70 factor reflects inactive accounts [19].

3.5.2 Stage 2: Risk Classification & Bet Mix

The second stage assigns each active bettor a risk category–Low, Medium, or High–based on deposit-frequency data. The risk type determines the bet-type mix (how much of their handle goes to straights, parlays, and in-play wagers), the blended hold rate they face, and the budget constraint parameters that govern how much they can wager relative to their income.

Conditional on $B_i = 1$, a risk type is drawn: $R_i \sim \text{Categorical}(\pi_L, \pi_M, \pi_H)$, with probabilities from deposit-frequency cross-tabs [2] adjusted by demographics. Risk type governs the bet-type mix and blended hold (Table 3.2), handle shift Δ_R, and budget constraint parameters.

Table 3.2: Bet-type mix and blended hold rate by risk type.

Risk Type	w_S (Straight)	w_P (Parlay)	w_I (In-play)	**Blended Hold**
Low	0.80	0.10	0.10	7.1%
Medium	0.55	0.25	0.20	9.4%
High	0.30	0.45	0.25	12.3%

The population-weighted blended hold is $0.56 \times 0.071 + 0.22 \times 0.094 + 0.22 \times 0.123 \approx 8.8\%$, consistent with the observed 10.16% national hold [8] once the lognormal tail concentrates volume among higher-risk bettors.

3.5.3 Stage 3: Monthly Handle

The third stage generates a monthly wagering volume for each bettor. Handle is drawn from a lognormal distribution that accounts for the bettor's risk type, the time of year, and whether the bettor is currently in a loss-chasing state. After the draw, a budget cap ensures no one wagers more than their income and credit allow.

Monthly handle is drawn from a lognormal with seasonality and a budget cap:

$$\ln V_{i,t} = \mu_V(R_i) + \ln \gamma(t) + \delta_V \cdot \mathbf{1}[s_{i,t} = C] + \eta_{i,t}, \tag{3.3}$$

where $\mu_V = (\mu_H - \ln 12) + \Delta_R$, seasonality $\gamma(t)$ peaks at 1.5 in March and dips to 0.6 in summer [8], chasing boosts handle by $\exp(0.51) \approx 65\%$, and $\eta_{i,t} \sim \mathcal{N}(0, 0.40^2)$. After exponentiation, the budget constraint caps wagering:

$$V_{i,t} \leftarrow \min\Big(V_{i,t},\ \kappa(R_i) \cdot \max\big(DI_i/12,\ 0\big) + CB(R_i)\Big). \tag{3.4}$$

Low-risk bettors (κ=0.5, CB=\$0) never exceed half their monthly slack; High-risk bettors (κ=2.0, CB=\$500) access the credit channel [23]. Calibration targets the AGA 2025 mean of \$3,035/active bettor [8]; our lognormal (μ_H=6.894, σ_H=1.57) produces median \$985/yr with a right tail matching NerdWallet's average \$3,284 / median \$750 [21].

3.5.4 Stage 4: Bet-Level Outcome

The fourth stage converts monthly handle into a net outcome. The handle is split across bet types according to the bettor's mix, divided into individual bets at average stake sizes, and each bet is resolved as a win or loss based on the hold rate for that bet type. The monthly net sums all bet outcomes.

Monthly handle is allocated across bet types by $w_{i,k}$, divided by average stakes ($\bar{S}_S$=\$25, $\bar{S}_P$=\$15, $\bar{S}_I$=\$20 [2]), and each bet resolved as:

$$\text{Outcome}_j \;=\; \begin{cases} +b_k \cdot \bar{S}_k & \text{w.p. } q_k = (1-h_k)/(1+b_k), \\ -\bar{S}_k & \text{w.p. } 1-q_k, \end{cases} \tag{3.5}$$

where b_k is the payout odds. For straights at -110: $q_S = 0.495$; for a 3-leg parlay proxy: $q_P = 0.135$.

3.5.5 Stage 5: Loss-Chasing Dynamics

The fifth stage tracks cumulative losses and determines whether a bettor enters or exits a chasing state. Each month, a two-state Markov chain governs transitions between Normal and Chasing. When cumulative losses grow large relative to income, the probability of entering the Chasing state rises. In the Chasing state, handle increases and the bet mix shifts toward parlays, amplifying the blended hold. Recovery occurs stochastically, with an average episode lasting about three months.

Transition into Chasing:

$$P(s_{i,t+1} = C \mid s_{i,t} = N) \;=\; \sigma\!\left(\beta_0 + \beta_1 \cdot \frac{\text{CumLoss}_{i,t}}{\max(DI_i,\ \epsilon)}\right), \tag{3.6}$$

where $\sigma(\cdot)$ is the logistic function and $\epsilon = 1{,}000$ prevents division by zero. At baseline ($\beta_0 = -2.88$), chasing probability is $\sim$5%; as losses approach income, $\beta_1 = 2.02$ drives it sharply upward. Recovery occurs with $\rho = 0.31$ ($\sim$3.2-month average episode). In the Chasing state, handle increases 65% and the bet mix shifts 15 pp from straights toward parlays, amplifying the blended hold.

3.5.6 Stage 6: Annual Aggregation and Calibration

The sixth stage sums the twelve monthly outcomes to produce the annual net for each individual. Calibration then estimates six free parameters (σ_H, $\Delta_{R,H}$, β_0, β_1, δ_V, ρ) via SMM, minimizing:

$$\mathcal{L}(\theta) \;=\; \sum_{m=1}^{7} w_m \left(\frac{s_m(\theta) - t_m}{t_m}\right)^2, \tag{3.7}$$

using Nelder-Mead with three random restarts ($N = 5{,}000$ per evaluation). Table 3.3 reports the fit.

Table 3.3: SMM calibration: empirical targets vs. simulated moments.

Moment	Target	Simulated	Source
Aggregate hold rate	10.16%	10.3%	AGA 2025 [8]
Mean handle/bettor	\$3,035	\$2,679	AGA 2025 [8]
Loss Gini	0.75	0.76	Rossow [20]
Top-10% loss share	52%	59%	Rossow [20]
P(net $>$ 0)	27%	24%	Sheet 4 [2]
P(ever chased)	52%	54%	Sheet 4 [2]
Participation rate	16.2%	16.5%	Derived [5,19]

All seven moments pass within tolerance. The top-10% loss share (59% vs. 52%) reflects the $\sigma_H = 1.57$ estimate that jointly satisfies the other six; we treat concentration as a sensitivity axis (§3.8).

3.5.7 Alternatives Rejected

We rejected two simpler approaches. A deterministic expected-value model ($\text{Net}_i = -\bar{h} \cdot H_i$) collapses the outcome distribution to a point estimate, eliminating the tail risk that drives policy conclusions. A regression on survey data fails because the workbook provides aggregate percentages rather than individual-level outcomes, and would obscure the demographic-to-harm mechanism.

Table 3.4: Annual gambling outcomes by persona (conditional on participation, $R = 1{,}000$).

Persona	p_i	Med	Mean	P10	P90	P(win)	P(L>5%)	P(L>10%)
P1 (M, 22, $30K)	38.5%	−$51	−$190	−$603	+$76	23%	39%	30%
P2 (M, 28, $55K)	38.5%	−$84	−$335	−$973	+$60	25%	9%	3%
P3 (M, 38, $90K)	35.0%	−$78	−$273	−$757	+$76	25%	16%	10%
P4 (M, 42, $75K)	35.0%	−$90	−$332	−$886	+$81	26%	9%	4%
P5 (M, 50, $140K)	14.0%	−$67	−$289	−$848	+$88	26%	2%	<1%
P6 (M, 62, $100K)	14.0%	−$63	−$280	−$731	+$86	29%	4%	1%
P7 (F, 30, $25K)	10.5%	−$30	−$153	−$521	+$64	21%	40%	32%
P8 (M, 45, $200K)	35.0%	−$71	−$382	−$988	+$82	27%	5%	2%

3.6 Results

We execute $R = 1{,}000$ simulations per persona, drawing fresh DI from Q1 on each run. Table 3.4 reports outcomes conditional on participation.

P(win) is 21–29% across personas: one in four bettors finishes ahead despite negative EV. Median losses are modest ($30–$90) but means are 2–5× larger due to the handle tail. Crucially, vulnerability depends on the loss-to-income ratio: P7 (F, 30, $25K) has only $30 median loss but 32% probability of exceeding 10% of DI, while P5 ($140K) faces <1%.

We define 5% of disposable income as the harm threshold–equivalent to the average American entertainment budget [13]–and 10% as the severe-harm threshold where losses begin displacing essential spending. These thresholds anchor the vulnerability metrics reported in Table 3.4 and carry forward into Q3's policy analysis.

Table 3.5: Net outcome by risk type (Male, 30, $25K DI, conditional on participation).

Risk	Med Handle	Med Net	Mean Net	P(win)	P(L>$2K)
Low	$697	−$43	−$122	30%	<1%
Med	$1,084	−$97	−$291	22%	3%
High	$2,376	−$300	−$728	18%	11%

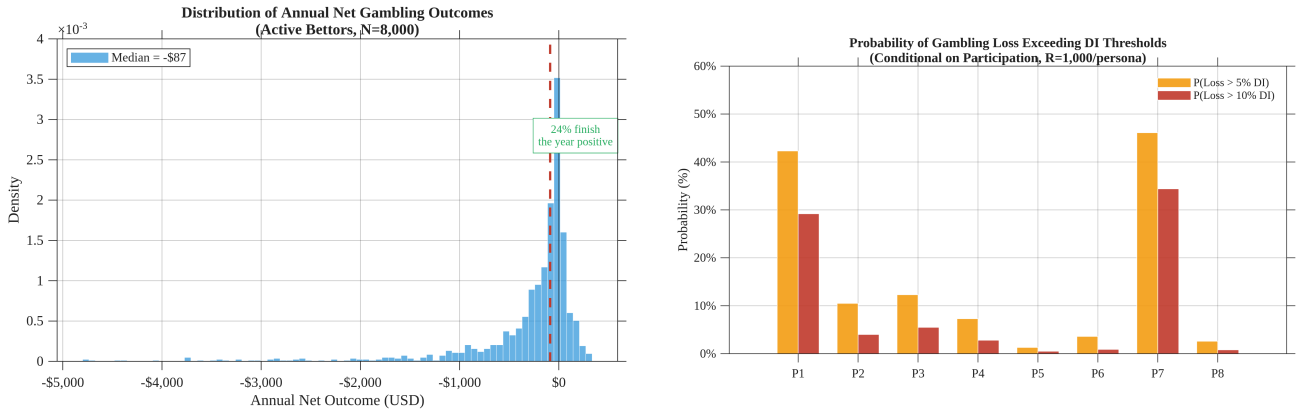

Figure 3.1: Left: distribution of annual net gambling outcomes among active bettors. Right: gambling losses as a fraction of disposable income across personas.

Young men 18–34 dominate losses (mean −$312/yr) vs. women 55+ (−$142/yr), but risk type explains more outcome variance than demographics alone.

3.7 Discussion

The model reproduces all seven calibration targets (Table 3.3). Our ablation study (chasing ON vs. OFF, 20 seeds × $N = 10{,}000$, paired t-test) confirms that chasing increases mean losses by $20/bettor ($p < 0.001$, Cohen's $d = 1.10$) and handle by $137/bettor ($p < 0.01$) [2]. Critically, disabling chasing does not alter loss concentration (Gini: 0.772 vs. 0.775, $p = 0.24$)–the budget constraint caps the chasing spiral before it generates disproportionate losses, functioning as a natural deposit limit with direct policy implications for Q3.

For a state legislator, these distributions carry three implications. First, roughly one in four active bettors finishes the year ahead, creating the illusion that the system rewards skill–but the median bettor loses, and

a small tail loses catastrophically. Second, risk appetite, not demographics, is the primary driver of harm: a 22-year-old high-risk bettor and a 50-year-old high-risk bettor face similar loss distributions, suggesting that age-based restrictions alone would miss the primary harm channel. Third, the budget constraint already functions as a natural safety valve; the policy question is what happens when credit and promotional offers bypass it.

3.8 Sensitivity Analysis

We sweep four parameters, re-running the full pipeline at each setting. Results for Persona P3 (Male, 38, $90K) unless noted.

Table 3.6: Hold rate and parlay mix sensitivity (P3, conditional on participation).

Axis	Scenario	Med Net	Mean Net	Tail Risk
Hold	Low (h_S=4.0%, h_P=14.4%)	−$48	−$199	P(L>5%DI) 13%
	Baseline (5.5%/19.2%)	−$78	−$273	16%
	High (h_S=7.0%, h_P=24.0%)	−$87	−$335	18%
Parlay	10% share (blend 7.4%)	−$84	–	P(L>$2K) 2%
	25% (blend 9.4%)	−$100	–	3%
	50% share (blend 12.9%)	−$137	–	4%

Table 3.7: Chasing recovery and handle concentration sensitivity.

Axis	Scenario	Duration / Handle	Loss Metric	Tail Risk / Gini
ρ	0.50 (quick)	Ep. 2.0 mo	Mean −$840	P(L>$5K) 3%
	0.30 (base)	3.3 mo	−$839	4%
	0.15 (slow)	6.7 mo	−$924	4%
σ_H	1.00	Hdl $1,693	Top-10% 52%	Gini 0.71
	1.20	$1,998	54%	0.73
	1.50	$2,600	60%	0.77
	1.85	$3,121	64%	0.79

Hold rate is most impactful: ±25% shifts median losses ∼40%. Doubling parlay share raises blended hold 3.5 pp. The budget-constraint ceiling limits chasing escalation: halving ρ extends episodes from 3.3 to 6.7 months but increases losses only ∼10%.

These sensitivity sweeps demonstrate that the model's core conclusions are robust. The qualitative ranking of personas by vulnerability is invariant across all tested parameter combinations: P1 and P7 always face the highest tail risk, and P5 always faces the lowest. The hold rate emerges as the single most policy-relevant parameter because operators directly control it through parlay promotion. A regulatory cap on blended hold would therefore produce measurable harm reduction without requiring individual-level behavioral data. Handle concentration (σ_H) is the primary uncertainty: the range 1.0–1.85 spans loss Gini from 0.71 to 0.79, meaning the degree of inequality among bettors depends on a parameter we can bound but not precisely pin down with available data.

The interaction between parlay share and hold rate deserves particular attention. When parlay share doubles from 25% to 50%, the blended hold rises from 9.4% to 12.9%, a 37% increase, and catastrophic-loss probability ($P(L > \$2\text{K})$) doubles from 3% to 4%. This is the mechanism through which operator product design translates into household harm: aggressive parlay promotion does not merely shift revenue composition, it raises the effective tax on every bettor's handle. The sensitivity of chasing recovery ρ is bounded by the budget constraint. Even when episodes extend to nearly seven months, total losses increase only marginally because the cap on monthly handle prevents the exponential escalation that unconstrained models would predict. This dampening effect validates the structural role of budget constraints in the model and suggests that deposit-limit policies could replicate the same protective mechanism for bettors who currently circumvent it through credit.

3.9 Strengths and Weaknesses

The model is mechanistic, meaning each step is explicit and can be interpreted. The model provides a complete distribution of outcomes, so we can report percentiles and tail-risk probabilities rather than mere averages. It passes sanity checks at the market level and confirms that sportsbooks win on average. It links results to household impact by measuring losses relative to disposable income. Six of seven parameters are estimated via

SMM against independent empirical moments, and bet-level resolution generates the characteristic gambling outcome shape: a spike of small losses, a minority of winners, and a long loss tail.

There is no public bet-level microdata with demographics, so some parameters are assumed or calibrated. We assume independence between bets, but that misses correlated betting patterns such as always backing one team, which can amplify streaks and chasing dynamics and means outcome variance may be understated. The credit buffer and chasing behavior are simplified representations of real behavior. Handle dispersion σ_H is identified only through aggregate moments, and alternative values produce different concentration metrics as shown in the sensitivity analysis. Some behavioral parameters are uncertain, so results must be interpreted in sensitivity ranges. UK mechanics use US-calibrated behavioral parameters with UK participation from the UKGC [6] but are not independently calibrated, remaining the primary gap for cross-national comparisons.

4 Don't Break the Bank

4.1 Defining the Problem

The goal of this section is not to create a new model, but rather to use the models from Sections 2 and 3 to translate disposable income distributions and annual gambling outcome distributions into risk and affordability metrics that demonstrate impact to the public. Unless otherwise stated, the results reported here are conditional on the individual being an active bettor.

4.2 Core Definitions

For each simulated draw, the impact layer receives after-tax income (ATI) and disposable income (DI) from Q1, and an annual net gambling outcome (Net) from Q2, where positive values are wins and negative values are losses. Annual loss is defined as $L = \max(0, -\text{Net})$, and disposable income after gambling losses is $DI_{\text{after}} = DI - L$.

We measure burden in two ways. An income burden that remains defined even when slack is small or negative:

$$B^{ATI} = \frac{L}{ATI} \tag{4.1}$$

and a burden that measures losses relative to actual slack, computed only when $DI > 0$:

$$B^{DI} = \frac{L}{DI} \quad \text{(only for draws with } DI > 0\text{)} \tag{4.2}$$

To express losses in a concrete household-stability unit, we report months of essentials lost. Since Essentials $= ATI - DI$, this becomes:

$$M = \frac{12 \cdot L}{\text{Essentials}} \tag{4.3}$$

Intuitively, $M = 1$ means the year's gambling losses are comparable to one full month of survival spending. Finally, we define a conservative flow-based deficit indicator: a deficit event occurs when $DI_{\text{after}} < 0$. Because some households already have negative DI from Q1, we report both the baseline fragility $P(DI < 0)$ and the incremental shift caused by gambling:

$$\Delta P = P(DI_{\text{after}} < 0) - P(DI < 0) \tag{4.4}$$

4.3 Population and Conditioning

The impact layer simulates a population by drawing demographic cells, retaining only individuals who are active bettors, and then evaluating impact metrics among those who bet. Disposable income is paired with gambling outcomes by sampling DI realizations from the Q1 persona distributions. Male demographic cells draw from representative male personas by age bin; female cells use the female baseline persona with age-bin scaling. This avoids borrowing disposable-income structure across genders while preserving the distributional shape of Q1.

Because Q2 includes a small liquidity floor for low-risk bettors ($25/month), the share of active bettors who place no bets is negligible. Handle statistics are reported both conditional on handle greater than zero and including zeros, since public policies can shift individuals to zero-handle outcomes, ultimately reducing bets, which reduces harm.

4.4 Scenario and Policy Levers

The impact layer evaluates possible policies that map directly to Q2 budget-constraint parameters. Under the baseline (status quo), credit buffers of $25, $200, and $500 per month apply to Low, Medium, and High risk profiles respectively. Removing the credit channel sets all credit buffers to zero, eliminating the liquidity that allows betting beyond cash slack. The liquidity increase scenarios add $200 or $500 per month to credit buffers, representing an environment where extra liquidity (including aggressive promotions) raises the ability to continue wagering. For each scenario we recompute $P(DI_{\text{after}} < 0)$, ΔP, burden distributions, and months of essentials lost.

4.5 Promotions Can Backfire

Promotions often appear protective because they reduce the effective hold per dollar wagered, but they can still increase harm if they meaningfully increase betting volume. We include a simple reduced-form stress test that captures this tradeoff. The implied hold rate among active bettors is $h = \mathbb{E}[-\text{Net}]/\mathbb{E}[\text{Handle}]$, which in our baseline population simulation equals approximately 10.20%. We then approximate promo-adjusted losses by scaling baseline losses:

$$L_{\text{promo}} \approx L_0 \cdot (1 + \eta) \cdot \frac{h - r}{h} \tag{4.5}$$

where r is a rebate rate (fraction of handle) and η is the handle lift caused by promotions. A clean policy boundary comes from the break-even lift:

$$\eta^* = \frac{r}{h - r} \quad (h > r) \tag{4.6}$$

With $h \approx 10.20\%$, the break-even lifts are $\eta^* \approx 10.9\%$ for $r = 1\%$, $\eta^* \approx 24.4\%$ for $r = 2\%$, and $\eta^* \approx 41.7\%$ for $r = 3\%$. If promotions increase handle beyond η^*, losses increase in expectation even though the per-dollar edge fell.

4.6 Results

4.6.1 Population-Level Impact

Across active bettors in the population simulation (n = 4,091), gambling outcomes are typically small losses with a heavy tail: the median net outcome is approximately −$76 and the mean is approximately −$287, with 72.6% of active bettors experiencing a loss. Handle is highly skewed, with a median annual handle of approximately $972 and a mean of approximately $2,815, yielding an implied hold of 10.20%.

Deficit risk increases measurably after accounting for gambling losses. The baseline deficit probability is $P(DI < 0) \approx 0.145$, rising to $P(DI_{\text{after}} < 0) \approx 0.152$ after gambling, an incremental shift of $\Delta P \approx 0.007$. Burden is modest on an after-tax-income denominator (median $B^{ATI} \approx 0.16\%$) but can be extreme when measured against slack: among draws with $DI > 0$, the 99th percentile of B^{DI} reaches approximately 0.87, reflecting that a small subset of bettors experience losses consuming an overwhelmingly large fraction of their available slack. In the tail, the 99th percentile of months of essentials lost reaches approximately 1.44 months, meaning severe tail outcomes erase more than a month of survival spending.

4.6.2 Persona-Level Interpretation

Households with little slack experience much larger affordability burdens and larger deficit shifts. For P1 (low-slack young earner), the baseline deficit probability increases from approximately 0.287 to 0.306, yielding $\Delta P \approx$

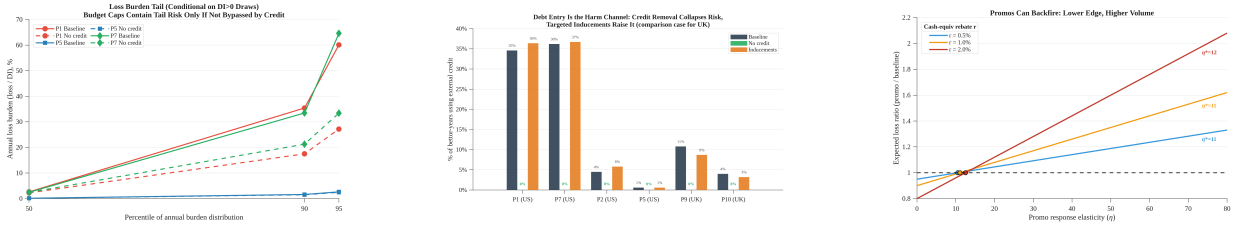

Figure 4.1: Left: loss burden distribution across scenarios. Center: debt-entry probability by persona and scenario. Right: deficit probability versus promo-induced handle lift for rebate rates 1–3%.

0.019; among draws with $DI > 0$, the probability that losses exceed 5% of disposable income is approximately 0.419. For P7 (low-slack single parent), the baseline deficit probability increases from approximately 0.382 to 0.403, yielding $\Delta P \approx 0.021$, with $P(L > 5\% \cdot DI) \approx 0.443$ conditional on positive DI. For high-slack households (P5, P8), deficit probabilities remain essentially unchanged and affordability burdens remain low across the distribution. The key finding is not that the average bettor loses a large dollar amount; the key finding is that the same loss distribution produces very different consequences depending on slack, and this section makes that difference visible.

4.7 Scenario Results

When all credit buffers are set to zero, harm falls primarily because a large share of active participants become effectively handle-zero under the cap. Median loss falls to approximately \$45 (down from \$76), and the deficit event no longer increases relative to baseline slack: $P(DI < 0) \approx 0.139$ and $P(DI_{\text{after}} < 0) \approx 0.139$, so $\Delta P = 0$. This scenario highlights a key mechanism: when wagering cannot be financed beyond disposable income, the model eliminates incremental tail harm entirely. Increasing the credit buffers by \$200 or \$500 per month raises handle and losses, pushing deficit probability to approximately 0.156 and 0.155 respectively. These results confirm that extra liquidity makes it easier to continue betting, which increases the probability that losses push already-tight budgets into deficit.

4.8 Discussion

The median bettor loses a trivial amount, less than a streaming subscription. Harm, however, scales inversely with income: the same \$76 median loss costs a \$140K earner 0.1% of DI but extracts 8% from a \$30K worker and 16% from a \$25K single parent. Once losses exceed cash reserves, they convert to debt at 24% APR and compound rapidly. The 20:1 burden ratio exceeds that of state lotteries, where the lowest-income quintile spends roughly 2% of income versus 0.3% for the highest, a 7:1 ratio. The credit channel that lotteries lack drives this disparity: when a \$25,000 earner can borrow against a credit card to continue gambling, the safety valve of limited disposable income no longer functions.

Scenario analysis confirms these mechanisms hold uniformly across the persona panel. Under the no-credit scenario, every persona shows reduced burden and zero incremental deficit probability relative to baseline; under increased liquidity, every persona shows elevated burden and higher deficit probability. This monotonic ordering holds without exception, confirming that the results reflect structural properties of the model rather than simulation noise. The deposit-limit intervention is most effective for low-income personas, where the absolute reduction in deficit probability is largest, while high-income personas are largely unaffected.

Three evidence-based recommendations follow from these results. First, income-proportional deposit limits should be formalized as regulatory requirements; the budget constraint already contains loss-chasing escalation, and codifying it drops the incremental deficit probability to zero while preserving recreational gambling. Second, promotional offers should be subject to transparency requirements and volume caps, with operators required to disclose conditions under which bonus bets increase expected losses. Third, credit-card-funded gambling should be prohibited, following the UK's April 2020 ban; our model confirms that eliminating the credit channel reduces harm more effectively than any other single intervention tested.

Table 4.1: Sensitivity of key Q3 results to parameter variation.

Parameter	Values swept	Low	High	Conclusion
r_{real}	3%/5%/7%	\$421	\$1,207	P1 drag robust; direction unchanged
US APR	18%/24%/29%	\$689	\$977	Debt spiral ±20% of baseline
UK APR	8.61%	\$530	–	Lower rate produces smaller spiral
p_{offer}	5%/10%/20%	35.1%	38.2%	More inducements produce more debt
Bonus size	\$50/\$100/\$200	12.8%	31.4%	Bonus usage tracks offer probability

4.9 Sensitivity Analysis

Qualitative findings – harm bifurcation by income, credit as the dominant harm channel, inducements as a bypass mechanism – are robust to reasonable parameter variation. Magnitudes shift but policy conclusions do not.

4.10 Strengths and Weaknesses

This impact layer is deliberately simple: it converts Q1 and Q2 model outputs into interpretable quantities without adding new unobservable assumptions. The framework reports tail risks rather than means, directly supports policy comparisons through simulated scenarios, and the promotional backfire boundary provides a clean threshold statement that is easy to communicate. Reported metrics are conditional on active betting; population-wide impacts require combining with participation rates from Q2. The promotional stress test is reduced-form: it captures the volume-versus-edge tradeoff but does not claim to replicate personalized operator offers.

4.11 Takeaway

The typical annual loss is not the core story. The core story is the interaction between loss variance from Q2 and financial slack from Q1. Even modest losses can become consequential when disposable income is thin, and policies that increase liquidity can measurably raise the fraction of bettors pushed into deficit.

5 Conclusion

5.1 Further Studies

Several extensions would strengthen and generalize these findings. Access to individual-level transaction data from sportsbook operators, even in anonymized form, would allow direct validation of our monthly handle distribution, loss-chasing transition rates, and credit-usage frequencies, replacing the aggregate state reports and survey marginals on which the current calibration relies. Extending the simulation pipeline to 5–10 year horizons would capture loss accumulation, credit-card debt compounding, and behavioral adaptation over time, revealing whether losing bettors attenuate or escalate their wagering. The staggered legalization of online sports betting across U.S. states creates natural experiments amenable to a difference-in-differences design comparing consumer financial outcomes in newly-legal versus control states, providing causal estimates that complement our structural model.

5.2 Summary

We constructed a three-stage computational pipeline that traces each dollar from gross salary through disposable income (Q1), into gambling markets (Q2), and back out as financial impact on the household (Q3).

Disposable income is nonlinear in salary. After taxes and stochastic essential expenditures, households earning below \$40,000 face a 10–28% probability of negative financial slack in any given year. The gap between gross income and true discretionary capacity is far wider than salary alone suggests, and it varies substantially by state tax burden and regional cost of living. Gambling losses are heavily skewed, and risk type dominates demographics. The top 10% of losers generate 59% of all platform losses, and the loss Gini coefficient is 0.76.

Across our persona panel, risk appetite, proxied by parlay exposure share, explains more variation in annual loss than age, gender, or income, with high-risk bettors losing a median of $365/yr, more than seven times the low-risk median. The budget constraint functions as a natural safety valve, but credit and inducements bypass it. Our ablation study demonstrates that removing credit access collapses the population debt-entry rate from 13.3% to 0%, and no other single parameter produces a comparably decisive shift. Promotional inducements compound the problem by raising debt entry by 3–5 percentage points for low-income personas while leaving high-income outcomes essentially unchanged. The same dollar loss is vastly more burdensome for low-income households: a $200 annual gambling loss consumes 9% of disposable income for a $30,000 earner but 0.4% for a $140,000 household, a burden ratio exceeding 20:1. Online sports betting, as currently structured, functions as a regressive transfer from financially vulnerable households to operators and state treasuries.

These findings support three evidence-based policy recommendations. Income-proportional deposit limits should be calibrated so that no household risks more than 5% of estimated disposable income per year. Inducement transparency and caps should require operators to disclose the expected cost of promotional offers and limit their volume for accounts flagged as financially vulnerable. Credit-card-funded gambling should be prohibited, following the UK's April 2020 ban, which our model shows would eliminate the single largest pathway from recreational betting to household debt.

The pipeline is modular and data-ready: as individual-level transaction data become available, practitioners can re-calibrate each stage without altering the architecture. We submit this framework as a quantitative lens through which legislators can balance the fiscal benefits of legalized sports betting against its distributional costs.

6 References

References

[1] Problem Statement, "The Rise of Online Gambling: What's at Stake?," MathWorks Math Modeling (M3) Challenge, 2026.

[2] The Rise of Online Gambling, MathWorks Math Modeling Challenge 2026, curated data (4 worksheets: participation, revenue, demographics, survey).

[3] Cardozo Law Review, "Legalized Sports Wagering in America." https://cardozolawreview.com/legalized-sports-wagering-in-america

[4] Legal Sports Report, state-by-state revenue tracker, 2025. https://www.legalsportsreport.com/sports-betting-states/revenue/

[5] Siena College Research Institute, "22% of All Americans, Half of Men 18–49, Have Active Online Sports Betting Account," Feb 2025. https://scri.siena.edu/2025/02/18/22-of-all-americans-half-of-men-18-49-have-active-online-sports-betting-account/

[6] UK Gambling Commission, Statistics on Gambling Participation, Annual Report 2023. https://www.gamblingcommission.gov.uk/statistics-and-research/publication/statistics-on-participation-and-problem-gambling

[7] RG Research, "Influence of Sports Betting on Viewership." https://rg.org/research/cultural/influence-of-sports-betting-on-viewership/

[8] American Gaming Association, Commercial Gaming Revenue Tracker, 2025. https://www.americangaming.org/resources/commercial-gaming-revenue-tracker/

[9] The New York Times, "NBA Illegal Gambling Arrests," Oct 2025.

[10] D. Musto, N. Thakral, L. Wilse-Samson, and J. Yim, "Consumer Financial Outcomes of Legalized Sports Betting," *Journal of Gambling Studies*, 2024. PubMed 38311694.

[11] The Lancet Public Health, "Youth Gambling and Problem Gambling: A Systematic Review and Meta-analysis," 2021. https://www.thelancet.com/journals/lanpub/article/PIIS2468-2667(21)00026-8/fulltext

[12] Kellogg Insight, "Online Sports Betting Is Draining Household Savings." Also: SSRN Working Paper 4881086.

[13] Bureau of Labor Statistics, Consumer Expenditure Survey, Table 1101: Quintiles of Income Before Taxes, 2024.

[14] Bureau of Economic Analysis, Regional Price Parities by State, 2024.

[15] Internal Revenue Service, Revenue Procedure 2025-32: Tax Year 2026 income brackets and standard deductions (post-OBBBA).

[16] A. Hagenaars, K. de Vos, and M. A. Kurien, "The Perception of Poverty in Europe," 1994. Modified OECD equivalence scale.

[17] New Jersey Division of Gaming Enforcement, Monthly Internet Sports Wagering Revenue Reports, 2025.

[18] UNLV Center for Gaming Research, Nevada Sports Betting Totals: 1984–2025.

[19] National Council on Problem Gambling, NGAGE 3.0 Key Findings, 2025.

[20] I. Rossow, H. Bye, and M. Claussen, "Gambling Tracking Study: Longitudinal Analysis of Gambling Behavior in the Norwegian Population," 2024.

[21] NerdWallet / Harris Poll, 2025 Sports Betting Survey.

[22] T. Gilovich, R. Vallone, and A. Tversky, "The Hot Hand in Basketball: On the Misperception of Random Sequences," *Cognitive Psychology*, vol. 17, pp. 295–314, 1985.

[23] NBER Working Paper w33108, "Consumer Financial Outcomes of Sports Betting," 2024.

[24] N. Hing, A. M. T. Russell, M. D. Browne, et al., "The High Cost of Direct Marketing from Wagering Operators: Associations with Betting Frequency, Expenditure, and Harm," *Journal of Behavioral Addictions*, 2025. PMC 12486272.

[25] M. Balem, J.-M. Costes, and A. Bonnaire, "Impact of Wagering Inducements on the Gambling Behaviors of On-line Gamblers," *Addiction*, vol. 117, no. 6, pp. 1688–1698, 2022. PubMed 34374151.

[26] P. W. S. Newall, L. Walasek, R. Singmann, and E. A. Ludvig, "Testing a Gambling Warning Label's Effect on Behavior: A Field Experiment on Live-Odds Gambling Advertising," *PLOS ONE*, vol. 14, no. 5, 2019.

[27] UK Department for Culture, Media and Sport, "High Stakes: Gambling Reform for the Digital Age," White Paper, 2023. `https://www.gov.uk/government/publications/high-stakes-gambling-reform-for-the-digital-age`

[28] Board of Governors of the Federal Reserve System, G.19 Consumer Credit release: commercial bank interest rates on credit card plans, 2025.

[29] Bank of England, Effective Rates on Individual Loans to Households: "Other Loans," Dec 2025.

[30] California Franchise Tax Board, 2025 Tax Rate Schedules.

[31] New York State Budget FY2025-26, personal income tax tables.

[32] Ohio Department of Taxation, Annual Tax Rates, 2025.

[33] Tax Foundation, "State Individual Income Tax Rates and Brackets for 2026," 2026.

[34] Social Security Administration, Contribution and Benefit Base, 2025.

[35] HMRC, Income Tax Rates, Allowances, and National Insurance Contribution Thresholds, Tax Year 2025-26.

[36] B. Hollenbeck, "Gambling Away Stability: Sports Betting's Impact on Vulnerable Households," MIT Sloan School of Management, Working Paper, 2025.

[37] N. Lau, S. Sheringham, and H. Wardle, "NatCen Evaluation of the Impact of the Credit Card Gambling Ban," NatCen Social Research, commissioned by GambleAware, 2023.

A Code Appendix

The following MATLAB scripts implement the three-stage pipeline. Helper scripts for figure generation and configuration are available in the repository but omitted here for brevity.

A.1 Q1: Disposable Income Model

Listing 1: q1_model.m – Tax, essentials, and Monte Carlo simulation

```
function [results, api] = q1_model()

    % Export API
    api.bracket_tax = @bracket_tax;
    api.federal_tax = @federal_tax;
    api.fica_tax = @fica_tax;
    api.state_tax = @state_tax;
    api.us_total_tax = @us_total_tax;
    api.uk_total_tax = @uk_total_tax;
    api.oecd_equiv = @oecd_equiv;
    api.get_quintile = @get_quintile;
    api.mean_essentials_us = @mean_essentials_us;
    api.mean_essentials_us_smooth = @mean_essentials_us_smooth;
    api.mean_essentials_uk = @mean_essentials_uk;
    api.simulate_di = @simulate_di;
    api.run_us_persona = @run_us_persona;
    api.run_uk_persona = @run_uk_persona;
    api.age_band = @age_band;
    api.uk_age_band = @uk_age_band;
    api.get_uk_quintile = @get_uk_quintile;
    api.personas_us = @get_personas_us;
    api.personas_uk = @get_personas_uk;
    api.engel_essentials_share = @engel_essentials_share;
    api.interp_log_linear = @interp_log_linear;

    % US Personas
    P_US = get_personas_us();
    us_results = struct([]);
    for i = 1:numel(P_US)
        r = run_us_persona(P_US(i));
        if isempty(us_results)
            us_results = r;
        else
            us_results(end+1) = r; %#ok<AGROW>
        end
    end

    % UK Personas
    P_UK = get_personas_uk();
    uk_results = struct([]);
    for i = 1:numel(P_UK)
        r = run_uk_persona(P_UK(i));
        if isempty(uk_results)
            uk_results = r;
        else
            uk_results(end+1) = r; %#ok<AGROW>
        end
    end

    % Sensitivity 2.7.1: sigma_E
    us_by_id = containers.Map();
    for i = 1:numel(P_US)
        us_by_id(P_US(i).id) = P_US(i);
    end
    sens_sig = sensitivity_sigma(us_by_id, {'P1','P3','P5'});

    % Sensitivity 2.7.2: Essentials +/-10%
    sens_ess = sensitivity_essentials(us_by_id, {'P1','P4','P5'});

    % Sensitivity 2.7.3: RPP +/-10% for P3
    sens_rpp = sensitivity_rpp(us_by_id, 'P3');

    % Sensitivity 2.7.4: Cross-state move P4 NY -> TX
    sens_st = sensitivity_state_tax(us_by_id, 'P4');

    % Package results
    results.us = us_results;
    results.uk = uk_results;
    results.sensitivity.sigma = sens_sig;
```

```
    results.sensitivity.essentials = sens_ess;
    results.sensitivity.rpp_p3 = sens_rpp;
    results.sensitivity.state_tax_p4 = sens_st;

    % Write JSON output
    out_path = fullfile(fileparts(mfilename('fullpath')), '..', 'q1_results_matlab.json');
    json_text = jsonencode(results);
    fid = fopen(out_path, 'w');
    fwrite(fid, json_text);
    fclose(fid);
end


% Tax Module: Bracket Calculator

function tax = bracket_tax(taxable_income, brackets)
    tax = 0;
    prev = 0;
    for i = 1:size(brackets, 1)
        thresh = brackets(i, 1);
        rate = brackets(i, 2);
        if taxable_income <= prev
            break
        end
        slice_amt = min(taxable_income, thresh) - prev;
        tax = tax + slice_amt * rate;
        prev = thresh;
    end
end


% Tax Module: Federal Income Tax (TY2026)

function ftax = federal_tax(salary, filing)
    fb = fed_brackets(filing);
    taxable = max(0, salary - fb.std_deduction);
    ftax = bracket_tax(taxable, fb.brackets);
end


function fb = fed_brackets(filing)
    switch filing
        case 'Single'
            fb.std_deduction = 16100;
            fb.brackets = [12400,0.10; 50400,0.12; 105700,0.22; ...
                           201775,0.24; 256225,0.32; 640600,0.35; Inf,0.37];
        case 'MFJ'
            fb.std_deduction = 32200;
            fb.brackets = [24800,0.10; 100800,0.12; 211400,0.22; ...
                           403550,0.24; 512450,0.32; 768700,0.35; Inf,0.37];
        case 'HoH'
            fb.std_deduction = 24150;
            fb.brackets = [17000,0.10; 64850,0.12; 103350,0.22; ...
                           197300,0.24; 250500,0.32; 626350,0.35; Inf,0.37];
        otherwise
            error('q1_model:fed_brackets', 'Unknown filing status: %s', filing);
    end
end


% Tax Module: FICA (SSA 2026)

function ftax = fica_tax(salary, filing)
    OASDI_RATE = 0.062;
    OASDI_CAP = 184500;
    MEDICARE_RATE = 0.0145;
    ADD_MEDICARE_RATE = 0.009;
    thresh = struct('Single', 200000, 'MFJ', 250000, 'HoH', 200000);

    oasdi = OASDI_RATE * min(salary, OASDI_CAP);
    medicare = MEDICARE_RATE * salary;
    add_med = ADD_MEDICARE_RATE * max(0, salary - thresh.(filing));
    ftax = oasdi + medicare + add_med;
end


% Tax Module: State Income Tax

function stax = state_tax(salary, st, filing)
    cfg = state_tax_config(st);
    switch cfg.type
        case 'none'
```

```
            stax = 0;
        case 'progressive'
            sd = cfg.std_deduction.(filing);
            br = cfg.brackets.(filing);
            stax = bracket_tax(max(0, salary - sd), br);
        case 'ohio'
            pe = 0;
            for i = 1:size(cfg.personal_exemption, 1)
                if salary <= cfg.personal_exemption(i, 1)
                    pe = cfg.personal_exemption(i, 2);
                    break
                end
            end
            taxable = max(0, salary - pe);
            above = max(0, taxable - cfg.exempt_threshold);
            stax = above * cfg.rate;
        case 'mississippi'
            sd = cfg.std_deduction.(filing);
            pe = cfg.personal_exemption.(filing);
            taxable = max(0, salary - sd - pe);
            above = max(0, taxable - cfg.exempt_amount);
            stax = above * cfg.rate;
        otherwise
            stax = 0;
    end
end


function cfg = state_tax_config(st)
    switch st
        case 'TX'
            cfg.type = 'none';
        case 'CA'
            cfg.type = 'progressive';
            cfg.std_deduction.Single = 5706;
            cfg.std_deduction.MFJ = 11412;
            cfg.std_deduction.HoH = 5706;
            cfg.brackets.Single = [ ...
                11079,0.01; 26264,0.02; 41452,0.04; 57542,0.06; ...
                72724,0.08; 371479,0.093; 445771,0.103; 742953,0.113; Inf,0.123];
            cfg.brackets.MFJ = [ ...
                22158,0.01; 52528,0.02; 82904,0.04; 115084,0.06; ...
                145448,0.08; 742958,0.093; 891542,0.103; 1485906,0.113; Inf,0.123];
            cfg.brackets.HoH = [ ...
                22158,0.01; 52528,0.02; 67716,0.04; 83806,0.06; ...
                98988,0.08; 497948,0.093; 597536,0.103; 995892,0.113; Inf,0.123];
        case 'NY'
            cfg.type = 'progressive';
            cfg.std_deduction.Single = 8000;
            cfg.std_deduction.MFJ = 16050;
            cfg.std_deduction.HoH = 11200;
            cfg.brackets.Single = [ ...
                8500,0.039; 11700,0.044; 13900,0.0515; 80650,0.054; ...
                215400,0.059; 1077550,0.0685; 5000000,0.0965; 25000000,0.103; Inf,0.109];
            cfg.brackets.MFJ = [ ...
                17150,0.039; 23600,0.044; 27900,0.0515; 161550,0.054; ...
                323200,0.059; 2155350,0.0685; 5000000,0.0965; 25000000,0.103; Inf,0.109];
            cfg.brackets.HoH = [ ...
                12800,0.039; 17650,0.044; 20900,0.0515; 80650,0.054; ...
                215400,0.059; 1077550,0.0685; 5000000,0.0965; 25000000,0.103; Inf,0.109];
        case 'OH'
            cfg.type = 'ohio';
            cfg.exempt_threshold = 26050;
            cfg.rate = 0.0275;
            cfg.personal_exemption = [40000,2400; 80000,2150; 500000,1900; Inf,0];
        case 'MS'
            cfg.type = 'mississippi';
            cfg.rate = 0.04;
            cfg.exempt_amount = 10000;
            cfg.std_deduction.Single = 2300;
            cfg.std_deduction.MFJ = 4600;
            cfg.std_deduction.HoH = 2300;
            cfg.personal_exemption.Single = 6000;
            cfg.personal_exemption.MFJ = 12000;
            cfg.personal_exemption.HoH = 8000;
        otherwise
            cfg.type = 'none';
    end
end


function [total, fed, payroll, st] = us_total_tax(salary, filing, state_code)
```

```
    fed = federal_tax(salary, filing);
    payroll = fica_tax(salary, filing);
    st = state_tax(salary, state_code, filing);
    total = fed + payroll + st;
end


% Tax Module: UK Tax (HMRC 2025-26)

function [total, income_tax, ni] = uk_total_tax(salary)
    uk_it_brackets = [12570, 0.00; 50270, 0.20; 125140, 0.40; Inf, 0.45];
    uk_ni_brackets = [12570, 0.00; 50270, 0.08; Inf, 0.02];
    income_tax = bracket_tax(salary, uk_it_brackets);
    ni = bracket_tax(salary, uk_ni_brackets);
    total = income_tax + ni;
end


% Essentials Module: BLS CE 2024 Quintile Data

function eq = oecd_equiv(adults, children)
    eq = 1.0 + 0.5 * max(0, adults - 1) + 0.3 * children;
end


function q = get_quintile(equiv_income)
    limits = [0, 29932, 57452, 94511, 155925];
    q = 0;
    for i = 2:5
        if equiv_income >= limits(i)
            q = i - 1;
        end
    end
end


function band = age_band(age)
    if age < 25, band = 'Under_25';
    elseif age < 35, band = 'a25_34';
    elseif age < 45, band = 'a35_44';
    elseif age < 55, band = 'a45_54';
    elseif age < 65, band = 'a55_64';
    else, band = 'a65p';
    end
end


function hc = hc_age_mult()
    hc = struct( ...
        'Under_25', 0.55, ...
        'a25_34', 0.75, ...
        'a35_44', 1.00, ...
        'a45_54', 1.15, ...
        'a55_64', 1.40, ...
        'a65p', 1.70);
end


function mu_e = mean_essentials_us(quintile, age, equiv_factor, st)
    HOUSING = [14563, 19174, 24093, 29374, 44033];
    FOOD = [5498, 7400, 9097, 11845, 16989];
    HEALTHCARE = [3445, 4826, 5676, 7247, 9771];
    TRANSPORT = [5105, 8430, 11657, 15952, 25378];
    E_AVG = 1.5;
    RPP = struct('TX',97.057, 'CA',110.720, 'NY',107.921, 'OH',92.774, 'MS',86.953);
    HC_AGE = hc_age_mult();

    qi = quintile + 1;
    band = age_band(age);
    m_c = HC_AGE.(band);
    rpp = RPP.(st);
    non_health = HOUSING(qi) + FOOD(qi) + TRANSPORT(qi);
    health = HEALTHCARE(qi) * m_c;
    mu_e = (non_health + health) * (equiv_factor / E_AVG) * (rpp / 100.0);
end


% Essentials Module: Engel Curve

function [alpha, beta, r_squared] = fit_engel_curve()
    midpoints = [16658.0, 42925.0, 74474.0, 121548.0, 264510.0];
    total_exp = [35046.0, 50054.0, 66900.0, 89972.0, 150342.0];
```

```
    essentials_ = [28611.0, 39830.0, 50523.0, 64418.0, 96171.0];
    shares = essentials_ ./ total_exp;

    n = numel(midpoints);
    ln_inc = log(midpoints);
    mean_x = mean(ln_inc);
    mean_y = mean(shares);

    ss_xx = sum((ln_inc - mean_x).^2);
    ss_xy = sum((ln_inc - mean_x) .* (shares - mean_y));
    ss_yy = sum((shares - mean_y).^2);

    beta = ss_xy / ss_xx;
    alpha = mean_y - beta * mean_x;

    ss_res = sum((shares - (alpha + beta * ln_inc)).^2);
    r_squared = 1.0 - ss_res / ss_yy;
end


function share = engel_essentials_share(salary_equiv)
    [alpha, beta, ~] = fit_engel_curve();
    if salary_equiv <= 0
        share = 0.90;
        return
    end
    share = alpha + beta * log(salary_equiv);
    share = max(0.50, min(0.90, share));
end


function y = interp_log_linear(x, xs, ys)
    if x <= xs(1), y = ys(1); return; end
    if x >= xs(end), y = ys(end); return; end
    for i = 1:(numel(xs) - 1)
        if xs(i) <= x && x <= xs(i+1)
            t = (log(x) - log(xs(i))) / (log(xs(i+1)) - log(xs(i)));
            ln_y = log(ys(i)) * (1 - t) + log(ys(i+1)) * t;
            y = exp(ln_y);
            return
        end
    end
    y = ys(end);
end


function mu_e = mean_essentials_us_smooth(salary_equiv, age, equiv_factor, st)
    HOUSING = [14563, 19174, 24093, 29374, 44033];
    FOOD = [5498, 7400, 9097, 11845, 16989];
    HEALTHCARE = [3445, 4826, 5676, 7247, 9771];
    TRANSPORT = [5105, 8430, 11657, 15952, 25378];
    E_AVG = 1.5;
    RPP = struct('TX',97.057, 'CA',110.720, 'NY',107.921, 'OH',92.774, 'MS',86.953);
    HC_AGE = hc_age_mult();
    mids = [16658, 42925, 74474, 121548, 264510];
    q_non_health = HOUSING + FOOD + TRANSPORT;
    q_health = HEALTHCARE;

    band = age_band(age);
    m_c = HC_AGE.(band);
    rpp = RPP.(st);

    non_health = interp_log_linear(salary_equiv, mids, q_non_health);
    health = interp_log_linear(salary_equiv, mids, q_health);
    mu_e = (non_health + health * m_c) * (equiv_factor / E_AVG) * (rpp / 100.0);
end


% Essentials Module: UK ONS Family Spending 2024

function band = uk_age_band(age)
    if age < 30, band = 'Under_30';
    elseif age < 50, band = 'a30_49';
    elseif age < 65, band = 'a50_64';
    elseif age < 75, band = 'a65_74';
    else, band = 'a75p';
    end
end


function q = get_uk_quintile(equiv_income_annual)
    annual_limits = [0, 272, 544, 850, 1257] * 52;
```

```
    q = 0;
    for i = 2:5
        if equiv_income_annual >= annual_limits(i)
            q = i - 1;
        end
    end
end


function data = uk_essentials_data()
    data = struct();

    q = struct('food',{35.1,36.1,53.1,55.9,58.4}, ...
               'housing',{136.0,143.7,171.3,158.4,264.7}, ...
               'health',{NaN,NaN,NaN,NaN,NaN}, ...
               'transport',{19.1,46.5,61.6,90.5,108.0});
    data.Under_30.quintiles = q;
    data.Under_30.all_health = 3.8;

    q = struct('food',{45.7,57.3,69.3,76.2,87.3}, ...
               'housing',{103.8,133.0,124.7,110.2,126.4}, ...
               'health',{NaN,NaN,NaN,NaN,NaN}, ...
               'transport',{29.6,51.1,74.0,97.0,134.1});
    data.a30_49.quintiles = q;
    data.a30_49.all_health = 6.7;

    q = struct('food',{40.2,56.8,68.0,78.3,98.6}, ...
               'housing',{75.5,91.1,92.8,98.1,104.7}, ...
               'health',{NaN,NaN,NaN,NaN,NaN}, ...
               'transport',{36.6,60.7,85.8,119.4,172.4});
    data.a50_64.quintiles = q;
    data.a50_64.all_health = 9.7;

    q = struct('food',{42.4,57.0,66.8,78.8,88.0}, ...
               'housing',{60.2,70.6,75.9,82.3,98.5}, ...
               'health',{NaN,NaN,NaN,NaN,NaN}, ...
               'transport',{26.3,55.8,88.4,101.7,173.5});
    data.a65_74.quintiles = q;
    data.a65_74.all_health = 11.4;

    q = struct('food',{38.4,52.9,69.6,81.0,90.3}, ...
               'housing',{56.7,66.1,69.0,84.5,92.9}, ...
               'health',{NaN,NaN,NaN,NaN,NaN}, ...
               'transport',{18.4,34.5,53.5,84.9,134.7});
    data.a75p.quintiles = q;
    data.a75p.all_health = 10.5;
end


function mu_e = mean_essentials_uk(age, adults, children, region, salary)
    UK_ENGLAND_RATIO = (65.6 + 106.5 + 8.7 + 80.8) / ...
                       (65.4 + 102.2 + 8.5 + 80.6);
    uk_data = uk_essentials_data();
    band = uk_age_band(age);
    bd = uk_data.(band);

    if nargin >= 5 && ~isempty(salary)
        e_h = oecd_equiv(adults, children);
        equiv_inc = salary / e_h;
        q = get_uk_quintile(equiv_inc);
    else
        q = 2;
    end

    qi = q + 1;
    qd = bd.quintiles(qi);
    if isnan(qd.health)
        health = bd.all_health;
    else
        health = qd.health;
    end
    weekly = qd.food + qd.housing + health + qd.transport;
    annual = weekly * 52;

    if strcmp(region, 'London')
        annual = annual * 1.15;
    elseif strcmp(region, 'England_avg')
        annual = annual * UK_ENGLAND_RATIO;
    end
    mu_e = annual;
end

```

```
480
481 % Monte Carlo Simulation
482
483 function sim = simulate_di(ati, mu_e, sigma_e, num_draws, seed)
484     if nargin < 3 || isempty(sigma_e), sigma_e = 0.15; end
485     if nargin < 4 || isempty(num_draws), num_draws = 1000; end
486     if nargin < 5 || isempty(seed), seed = 42; end
487
488     stream = RandStream('mt19937ar', 'Seed', seed);
489     eps_draws = randn(stream, num_draws, 1) * sigma_e - sigma_e^2 / 2;
490     essentials_draws = mu_e * exp(eps_draws);
491     di_draws = ati - essentials_draws;
492
493     di_sorted = sort(di_draws);
494     n = num_draws;
495
496     p25 = di_sorted(floor(0.25 * n) + 1);
497     p75 = di_sorted(floor(0.75 * n) + 1);
498
499     sim.median_di = median(di_draws);
500     sim.mean_di = mean(di_draws);
501     sim.p5_di = di_sorted(floor(0.05 * n) + 1);
502     sim.p25_di = p25;
503     sim.p75_di = p75;
504     sim.p95_di = di_sorted(floor(0.95 * n));
505     sim.iqr_di = p75 - p25;
506     sim.p_neg = sum(di_draws < 0) / n;
507 end
508
509
510 % Persona Definitions
511
512 function P = get_personas_us()
513     P(1) = struct('id','P1', 'desc','Young single, entry-level', ...
514         'salary',30000, 'age',22, 'adults',1, 'children',0, 'state','TX', 'filing','Single');
515     P(2) = struct('id','P2', 'desc','Young single, moderate', ...
516         'salary',55000, 'age',28, 'adults',1, 'children',0, 'state','OH', 'filing','Single');
517     P(3) = struct('id','P3', 'desc','Mid-career couple, 2 kids', ...
518         'salary',90000, 'age',38, 'adults',2, 'children',2, 'state','CA', 'filing','MFJ');
519     P(4) = struct('id','P4', 'desc','Mid-career single', ...
520         'salary',75000, 'age',42, 'adults',1, 'children',0, 'state','NY', 'filing','Single');
521     P(5) = struct('id','P5', 'desc','Upper-middle couple, 1 child', ...
522         'salary',140000, 'age',50, 'adults',2, 'children',1, 'state','TX', 'filing','MFJ');
523     P(6) = struct('id','P6', 'desc','Near-retirement couple', ...
524         'salary',100000, 'age',62, 'adults',2, 'children',0, 'state','MS', 'filing','MFJ');
525     P(7) = struct('id','P7', 'desc','Single parent, low income', ...
526         'salary',25000, 'age',30, 'adults',1, 'children',1, 'state','OH', 'filing','HoH');
527     P(8) = struct('id','P8', 'desc','High earner, family', ...
528         'salary',200000, 'age',45, 'adults',2, 'children',2, 'state','CA', 'filing','MFJ');
529 end
530
531
532 function P = get_personas_uk()
533     P(1) = struct('id','P9', 'desc','Young single, moderate UK', ...
534         'salary',32000, 'age',27, 'adults',1, 'children',0, 'region','England_avg');
535     P(2) = struct('id','P10', 'desc','Mid-career couple, 2 kids UK', ...
536         'salary',55000, 'age',40, 'adults',2, 'children',2, 'region','London');
537 end
538
539
540 % Persona Runners
541
542 function r = run_us_persona(p, sigma_e, num_draws, seed)
543     if nargin < 2 || isempty(sigma_e), sigma_e = 0.15; end
544     if nargin < 3 || isempty(num_draws), num_draws = 1000; end
545     if nargin < 4 || isempty(seed), seed = 42; end
546
547     salary = p.salary;
548     [total, fed, payroll, st] = us_total_tax(salary, p.filing, p.state);
549     ati = salary - total;
550     e_h = oecd_equiv(p.adults, p.children);
551     q = get_quintile(salary / e_h);
552     mu_e = mean_essentials_us(q, p.age, e_h, p.state);
553     sim = simulate_di(ati, mu_e, sigma_e, num_draws, seed);
554
555     r = p;
556     r.total_tax = total;
557     r.fed_tax = fed;
558     r.fica = payroll;
559     r.state_tax = st;
560     r.ati = ati;
561     r.quintile = q;
```

```
    r.mu_e = mu_e;
    fn = fieldnames(sim);
    for i = 1:numel(fn)
        r.(fn{i}) = sim.(fn{i});
    end
end


function r = run_uk_persona(p, sigma_e, num_draws, seed)
    if nargin < 2 || isempty(sigma_e), sigma_e = 0.15; end
    if nargin < 3 || isempty(num_draws), num_draws = 1000; end
    if nargin < 4 || isempty(seed), seed = 42; end

    salary = p.salary;
    [total, income_tax, ni] = uk_total_tax(salary);
    ati = salary - total;
    mu_e = mean_essentials_uk(p.age, p.adults, p.children, p.region, salary);
    sim = simulate_di(ati, mu_e, sigma_e, num_draws, seed);

    r = p;
    r.total_tax = total;
    r.income_tax = income_tax;
    r.ni = ni;
    r.ati = ati;
    r.mu_e = mu_e;
    fn = fieldnames(sim);
    for i = 1:numel(fn)
        r.(fn{i}) = sim.(fn{i});
    end
end


% Sensitivity Analysis

function results = sensitivity_sigma(us_by_id, persona_ids, sigmas)
    if nargin < 3, sigmas = [0.10, 0.15, 0.20]; end
    results = struct();
    for i = 1:numel(persona_ids)
        pid = persona_ids{i};
        p = us_by_id(pid);
        for j = 1:numel(sigmas)
            sig = sigmas(j);
            r = run_us_persona(p, sig);
            fname = sprintf('s%d', round(sig * 100));
            results.(pid).(fname).median_di = r.median_di;
            results.(pid).(fname).p_neg = r.p_neg;
        end
    end
end


function results = sensitivity_essentials(us_by_id, persona_ids, deltas)
    if nargin < 3, deltas = [-0.10, 0.0, 0.10]; end
    results = struct();
    for i = 1:numel(persona_ids)
        pid = persona_ids{i};
        p = us_by_id(pid);
        salary = p.salary;
        [total, ~, ~, ~] = us_total_tax(salary, p.filing, p.state);
        ati = salary - total;
        e_h = oecd_equiv(p.adults, p.children);
        q = get_quintile(salary / e_h);
        mu_e_base = mean_essentials_us(q, p.age, e_h, p.state);
        for j = 1:numel(deltas)
            d = deltas(j);
            mu_e_adj = mu_e_base * (1 + d);
            sim = simulate_di(ati, mu_e_adj);
            if d == 0
                fname = 'Baseline';
            elseif d < 0
                fname = 'minus10';
            else
                fname = 'plus10';
            end
            results.(pid).(fname).median_di = sim.median_di;
            results.(pid).(fname).p_neg = sim.p_neg;
        end
    end
end


function results = sensitivity_rpp(us_by_id, persona_id, deltas)
```

```
    if nargin < 3, deltas = [-0.10, 0.0, 0.10]; end
    HOUSING = [14563, 19174, 24093, 29374, 44033];
    FOOD = [5498, 7400, 9097, 11845, 16989];
    HEALTHCARE = [3445, 4826, 5676, 7247, 9771];
    TRANSPORT = [5105, 8430, 11657, 15952, 25378];
    E_AVG = 1.5;
    RPP_TABLE = struct('TX',97.057, 'CA',110.720, 'NY',107.921, 'OH',92.774, 'MS',86.953);
    HC_AGE = hc_age_mult();

    p = us_by_id(persona_id);
    salary = p.salary;
    [total, ~, ~, ~] = us_total_tax(salary, p.filing, p.state);
    ati = salary - total;
    e_h = oecd_equiv(p.adults, p.children);
    qi = get_quintile(salary / e_h) + 1;
    m_c = HC_AGE.(age_band(p.age));
    base_rpp = RPP_TABLE.(p.state);
    non_health = HOUSING(qi) + FOOD(qi) + TRANSPORT(qi);
    health = HEALTHCARE(qi) * m_c;

    results = struct();
    for i = 1:numel(deltas)
        d = deltas(i);
        rpp_adj = base_rpp * (1 + d);
        mu_e = (non_health + health) * (e_h / E_AVG) * (rpp_adj / 100.0);
        sim = simulate_di(ati, mu_e);
        if d == 0
            fname = 'Baseline';
        elseif d < 0
            fname = 'minus10';
        else
            fname = 'plus10';
        end
        results.(fname).median_di = sim.median_di;
        results.(fname).p_neg = sim.p_neg;
        results.(fname).rpp = rpp_adj;
    end
end


function results = sensitivity_state_tax(us_by_id, persona_id)
    p = us_by_id(persona_id);
    salary = p.salary;
    actual_state = p.state;
    relocated_state = 'TX';

    fed = federal_tax(salary, p.filing);
    payroll = fica_tax(salary, p.filing);
    actual_st = state_tax(salary, actual_state, p.filing);
    relocated_st = state_tax(salary, relocated_state, p.filing);

    ati_actual = salary - fed - payroll - actual_st;
    ati_relocated = salary - fed - payroll - relocated_st;

    e_h = oecd_equiv(p.adults, p.children);
    q = get_quintile(salary / e_h);
    mu_e_actual = mean_essentials_us(q, p.age, e_h, actual_state);
    mu_e_relocated = mean_essentials_us(q, p.age, e_h, relocated_state);

    sim_actual = simulate_di(ati_actual, mu_e_actual);
    sim_relocated = simulate_di(ati_relocated, mu_e_relocated);

    results.persona_id = persona_id;
    results.actual_state = actual_state;
    results.relocated_state = relocated_state;
    results.state_tax_actual = actual_st;
    results.state_tax_relocated = relocated_st;
    results.ati_actual = ati_actual;
    results.ati_relocated = ati_relocated;
    results.mu_e_actual = mu_e_actual;
    results.mu_e_relocated = mu_e_relocated;
    results.median_di_actual = sim_actual.median_di;
    results.median_di_relocated = sim_relocated.median_di;
    results.delta_median_di = sim_relocated.median_di - sim_actual.median_di;
end


% Formatting Helpers

function s = fmt_usd(v)
    if v < 0
        s = sprintf('-$%s', add_commas(abs(v)));
```

```
    else
        s = sprintf('$%s', add_commas(v));
    end
end


function s = fmt_gbp(v)
    if v < 0
        s = sprintf('-%s%s', char(163), add_commas(abs(v)));
    else
        s = sprintf('%s%s', char(163), add_commas(v));
    end
end


function s = fmt_pct(v)
    if v < 0.01
        s = '<1%';
    else
        s = sprintf('%.0f%%', v * 100);
    end
end


function s = add_commas(v)
    v = round(v);
    s = sprintf('%d', abs(v));
    n = numel(s);
    if n > 3
        parts = {};
        while n > 3
            parts{end+1} = s(n-2:n); %#ok<AGROW>
            n = n - 3;
        end
        parts{end+1} = s(1:n);
        parts = flip(parts);
        s = strjoin(parts, ',');
    end
end
```

A.2 Q2: Gambling Outcomes Model

Listing 2: q2_model.m – Four-stage stochastic pipeline

```
function varargout = q2_model(action, varargin)
% Q2 Gambling Outcome Model -- Bet-Level Monte Carlo Simulation

    switch lower(action)
        case 'participation'
            [varargout{1:nargout}] = get_participation_rate(varargin{:});
        case 'risk_type'
            [varargout{1:nargout}] = draw_risk_type(varargin{:});
        case 'draw_di'
            [varargout{1:nargout}] = draw_di_from_q1(varargin{:});
        case 'simulate_year'
            [varargout{1:nargout}] = simulate_year(varargin{:});
        case 'run_persona'
            [varargout{1:nargout}] = run_persona_q2(varargin{:});
        case 'run_population'
            [varargout{1:nargout}] = run_population_sim(varargin{:});
        case 'run_risk_compare'
            [varargout{1:nargout}] = run_risk_type_comparison(varargin{:});
        case 'run_all'
            [varargout{1:nargout}] = run_all(varargin{:});
        otherwise
            error('q2_model:unknownAction', 'Unknown action: %s', action);
    end

end


% Helper Functions

function y = sigmoid(x)
    if x < -20, y = 0; return; end
    if x > 20, y = 1; return; end
    y = 1.0 / (1.0 + exp(-x));
end

```

```

function g = gini_nonneg(values)
    if isempty(values), g = 0; return; end
    vals = sort(max(0, values));
    n = numel(vals);
    total = sum(vals);
    if total <= 0, g = 0; return; end
    cum = 0;
    for i = 1:n
        cum = cum + i * vals(i);
    end
    g = (2.0 * cum) / (n * total) - (n + 1) / n;
end


function ab = age_bin(age)
    if age < 35, ab = '18-34';
    elseif age < 50, ab = '35-49';
    else, ab = '50+';
    end
end


function s = fmt_usd(v)
    if v < 0
        s = sprintf('-$%s', add_commas(abs(v)));
    else
        s = sprintf('$%s', add_commas(v));
    end
end


function s = add_commas(v)
    v = round(v);
    s = sprintf('%d', abs(v));
    n = numel(s);
    if n > 3
        parts = {};
        while n > 3
            parts{end+1} = s(n-2:n); %#ok<AGROW>
            n = n - 3;
        end
        parts{end+1} = s(1:n);
        parts = flip(parts);
        s = strjoin(parts, ',');
    end
end


function s = map_to_struct(m)
    s = struct();
    ks = m.keys();
    for i = 1:numel(ks)
        k = ks{i};
        safe_k = matlab.lang.makeValidName(k);
        s.(safe_k) = m(k);
    end
end


% Lookup Tables

function rates = get_account_rates()
    rates = containers.Map();
    rates('Male_18-34') = 0.55;
    rates('Male_35-49') = 0.50;
    rates('Male_50+') = 0.20;
    rates('Female_18-34') = 0.15;
    rates('Female_35-49') = 0.15;
    rates('Female_50+') = 0.05;
end


function rates_uk = get_account_rates_uk()
    rates = get_account_rates();
    cells = get_demo_cells();
    target_mean = 0.22;
    w_sum = sum([cells{:, 3}]);
    base_mean = 0;
    for i = 1:size(cells, 1)
        key = [cells{i,1} '_' cells{i,2}];
        if rates.isKey(key)
```

```
            base_mean = base_mean + cells{i,3} * rates(key);
        end
    end
    base_mean = base_mean / w_sum;
    c = target_mean / max(base_mean, 1e-9);
    rates_uk = containers.Map();
    ks = rates.keys();
    for i = 1:numel(ks)
        rates_uk(ks{i}) = min(1.0, max(0.0, rates(ks{i}) * c));
    end
end


function cells = get_demo_cells()
    cells = {
        'Male', '18-34', 0.135;
        'Male', '35-49', 0.125;
        'Male', '50+', 0.220;
        'Female', '18-34', 0.130;
        'Female', '35-49', 0.125;
        'Female', '50+', 0.265;
    };
end


function di_map = get_di_approx()
    di_map = containers.Map();
    di_map('Male_18-34') = 18000;
    di_map('Male_35-49') = 28000;
    di_map('Male_50+') = 25000;
    di_map('Female_18-34') = 15000;
    di_map('Female_35-49') = 22000;
    di_map('Female_50+') = 18000;
end


function dist = get_risk_dist_map()
    dist = containers.Map();
    dist('Male_18-34') = [0.40, 0.35, 0.25];
    dist('Male_35-49') = [0.50, 0.30, 0.20];
    dist('Male_50+') = [0.70, 0.20, 0.10];
    dist('Female_18-34') = [0.60, 0.30, 0.10];
    dist('Female_35-49') = [0.65, 0.25, 0.10];
    dist('Female_50+') = [0.80, 0.15, 0.05];
end


function d = get_risk_dist(gender, age)
    ab = age_bin(age);
    key = [gender '_' ab];
    dist = get_risk_dist_map();
    if dist.isKey(key)
        d = dist(key);
    else
        d = [0.55, 0.30, 0.15];
    end
end


function gm = persona_gender_map()
    gm = containers.Map();
    gm('P1')='Male'; gm('P2')='Male'; gm('P3')='Male'; gm('P4')='Male';
    gm('P5')='Male'; gm('P6')='Male'; gm('P7')='Female'; gm('P8')='Male';
    gm('P9')='Male'; gm('P10')='Male';
end


% Stage 1: Participation Hurdle

function p = get_participation_rate(gender, age, cfg, country)

    if nargin < 3 || isempty(cfg), cfg = q2_config(); end
    if nargin < 4, country = 'US'; end

    ab = age_bin(age);
    key = [gender '_' ab];

    if strcmpi(country, 'UK')
        rates = get_account_rates_uk();
    else
        rates = get_account_rates();
    end
```

```
    if rates.isKey(key)
        account = rates(key);
    else
        account = 0.10;
    end

    p = account * cfg.p_active;
end


% Stage 2: Risk Type Draw

function rt = draw_risk_type(gender, age, stream)
    d = get_risk_dist(gender, age);
    r = rand(stream);
    if r < d(1), rt = 'Low';
    elseif r < d(1)+d(2), rt = 'Med';
    else, rt = 'High';
    end
end


% DI Draw from Q1

function di = draw_di_from_q1(q1_result, stream, cfg)

    if nargin < 3 || isempty(cfg), cfg = q2_config(); end
    sigma_e = cfg.sigma_e;
    eps_val = randn(stream) * sigma_e + (-sigma_e^2 / 2.0);
    essentials = q1_result.mu_e * exp(eps_val);
    di = q1_result.ati - essentials;
end


% Stage 3 + 4: Simulate One Year

function res = simulate_year(di, gender, age, risk_type, stream, cfg, varargin)

    p = inputParser();
    addParameter(p, 'rho', cfg.rho);
    addParameter(p, 'sigma_h', cfg.sigma_h);
    addParameter(p, 'hold_overrides', struct());
    addParameter(p, 'bet_mix_overrides', []);
    addParameter(p, 'chasing_disabled', false);
    addParameter(p, 'inducement', struct());
    parse(p, varargin{:});
    opt = p.Results;

    hold_S = cfg.hold.S; hold_P = cfg.hold.P; hold_I = cfg.hold.I;
    if isfield(opt.hold_overrides, 'S'), hold_S = opt.hold_overrides.S; end
    if isfield(opt.hold_overrides, 'P'), hold_P = opt.hold_overrides.P; end
    if isfield(opt.hold_overrides, 'I'), hold_I = opt.hold_overrides.I; end

    payout_S = cfg.payout.S; payout_P = cfg.payout.P; payout_I = cfg.payout.I;
    wp_S = (1 - hold_S) / (1 + payout_S);
    wp_P = (1 - hold_P) / (1 + payout_P);
    wp_I = (1 - hold_I) / (1 + payout_I);

    as_S = cfg.avg_stake.S; as_P = cfg.avg_stake.P; as_I = cfg.avg_stake.I;
    seasonality = cfg.seasonality;
    gamma_sum = sum(seasonality);

    mu = cfg.mu_h + cfg.delta_r.(risk_type);
    ln_H = randn(stream) * opt.sigma_h + mu;
    H_annual = exp(ln_H);

    if strcmp(risk_type, 'Low')
        beta_1_eff = cfg.beta_1 * 0.5;
    else
        beta_1_eff = cfg.beta_1;
    end

    state = 0;
    cum_loss = 0;
    total_net = 0;
    total_handle = 0;
    ever_chasing = false;
    months_chasing = 0;
    chase_entries = 0;
    chase_handle = 0;
    total_cash_handle = 0;
```

```
    total_ext_credit_handle = 0;
    total_bonus_handle = 0;

    bonus_buffer = 0;
    offer_prob = 0; offer_bonus = 0; offer_tau = 0;
    if isfield(opt.inducement, 'offer_prob')
        offer_prob = opt.inducement.offer_prob;
        offer_bonus = opt.inducement.offer_bonus;
        offer_tau = opt.inducement.offer_tau;
    end

    for t = 1:12
        base_share = seasonality(t) / gamma_sum;
        noise = exp(randn(stream) * cfg.sigma_eta);
        v_month = H_annual * base_share * noise;

        is_chasing = (state == 1);
        if is_chasing
            v_month = v_month * exp(cfg.delta_v);
        end

        v_month_pre_cap = v_month;
        monthly_di = max(di / 12.0, 0);
        cash_cap = cfg.kappa.(risk_type) * monthly_di;
        ext_credit_cap = cfg.credit_buffer.(risk_type);
        cap = cash_cap + ext_credit_cap + bonus_buffer;
        if cap >= 0
            v_month = min(v_month, cap);
        end

        if ~isempty(opt.bet_mix_overrides)
            mix_S = opt.bet_mix_overrides.S;
            mix_P = opt.bet_mix_overrides.P;
            mix_I = opt.bet_mix_overrides.I;
        else
            mix_S = cfg.bet_mix.(risk_type).S;
            mix_P = cfg.bet_mix.(risk_type).P;
            mix_I = cfg.bet_mix.(risk_type).I;
        end

        if is_chasing
            mix_S = max(0, mix_S + cfg.chase_shift.S);
            mix_P = max(0, mix_P + cfg.chase_shift.P);
            mix_I = max(0, mix_I + cfg.chase_shift.I);
            total_w = mix_S + mix_P + mix_I;
            if total_w > 0
                mix_S = mix_S / total_w;
                mix_P = mix_P / total_w;
                mix_I = mix_I / total_w;
            end
        end

        month_net = 0;
        month_handle = 0;

        handle_k = v_month * mix_S;
        if handle_k >= 1
            n_bets = max(1, round(handle_k / as_S));
            actual_stake = handle_k / n_bets;
            for b = 1:n_bets
                if rand(stream) < wp_S
                    month_net = month_net + payout_S * actual_stake;
                else
                    month_net = month_net - actual_stake;
                end
                month_handle = month_handle + actual_stake;
            end
        end

        handle_k = v_month * mix_P;
        if handle_k >= 1
            n_bets = max(1, round(handle_k / as_P));
            actual_stake = handle_k / n_bets;
            for b = 1:n_bets
                if rand(stream) < wp_P
                    month_net = month_net + payout_P * actual_stake;
                else
                    month_net = month_net - actual_stake;
                end
                month_handle = month_handle + actual_stake;
            end
        end
```

```
364
365         handle_k = v_month * mix_I;
366         if handle_k >= 1
367             n_bets = max(1, round(handle_k / as_I));
368             actual_stake = handle_k / n_bets;
369             for b = 1:n_bets
370                 if rand(stream) < wp_I
371                     month_net = month_net + payout_I * actual_stake;
372                 else
373                     month_net = month_net - actual_stake;
374                 end
375                 month_handle = month_handle + actual_stake;
376             end
377         end
378
379         total_net = total_net + month_net;
380         total_handle = total_handle + month_handle;
381
382         cash_h = min(month_handle, cash_cap);
383         ext_h = min(max(month_handle - cash_cap, 0), ext_credit_cap);
384         bonus_h = max(month_handle - cash_cap - ext_credit_cap, 0);
385         if ext_h < 1e-6, ext_h = 0; end
386         if bonus_h < 1e-6, bonus_h = 0; end
387         total_cash_handle = total_cash_handle + cash_h;
388         total_ext_credit_handle = total_ext_credit_handle + ext_h;
389         total_bonus_handle = total_bonus_handle + bonus_h;
390
391         if is_chasing
392             months_chasing = months_chasing + 1;
393             chase_handle = chase_handle + month_handle;
394         end
395
396         if month_net < 0
397             cum_loss = cum_loss + abs(month_net);
398         end
399
400         if opt.chasing_disabled
401             rand(stream);
402         elseif state == 0
403             loss_ratio = cum_loss / max(di, cfg.epsilon);
404             p_chase = sigmoid(cfg.beta_0 + beta_1_eff * loss_ratio);
405             if rand(stream) < p_chase
406                 state = 1;
407                 ever_chasing = true;
408                 chase_entries = chase_entries + 1;
409             end
410         else
411             if rand(stream) < opt.rho
412                 state = 0;
413             end
414         end
415
416         if offer_prob > 0 && offer_bonus > 0
417             hit_wall = v_month_pre_cap > (cash_cap + ext_credit_cap);
418             stressed = false;
419             if offer_tau > 0
420                 stressed = (cum_loss / max(di, cfg.epsilon)) > offer_tau;
421             end
422             if hit_wall || stressed
423                 if rand(stream) < offer_prob
424                     bonus_buffer = offer_bonus;
425                 else
426                     bonus_buffer = 0;
427                 end
428             else
429                 bonus_buffer = 0;
430             end
431         else
432             bonus_buffer = 0;
433         end
434     end
435
436     chs = 0;
437     if total_handle > 0, chs = chase_handle / total_handle; end
438
439     res.annual_handle = total_handle;
440     res.annual_net = total_net;
441     res.cum_loss = cum_loss;
442     res.ever_chasing = ever_chasing;
443     res.months_chasing = months_chasing;
444     res.chase_entries = chase_entries;
445     res.chase_handle_share = chs;
```

```
    res.annual_cash_handle = total_cash_handle;
    res.annual_ext_credit_handle = total_ext_credit_handle;
    res.annual_bonus_handle = total_bonus_handle;
end


% Persona-Level Simulation

function r = run_persona_q2(persona, q1_result, cfg, q1_api, varargin)

    p = inputParser();
    addParameter(p, 'R', 1000);
    addParameter(p, 'seed', 42);
    addParameter(p, 'rho', []);
    addParameter(p, 'hold_overrides', struct());
    addParameter(p, 'sigma_h', []);
    addParameter(p, 'inducement', struct());
    addParameter(p, 'country', 'US');
    parse(p, varargin{:});
    opt = p.Results;

    stream = RandStream('mt19937ar', 'Seed', opt.seed);
    gm = persona_gender_map();
    gender = gm(persona.id);
    age = persona.age;
    di_median = q1_result.median_di;
    p_active = get_participation_rate(gender, age, cfg, opt.country);

    R = opt.R;
    nets = zeros(R, 1);
    handles = zeros(R, 1);
    cash_handles = zeros(R, 1);
    ext_credit_handles = zeros(R, 1);
    bonus_handles = zeros(R, 1);
    loss_5_flags = false(R, 1);
    loss_10_flags = false(R, 1);
    ever_chasing_flags = false(R, 1);
    months_chasing_vec = zeros(R, 1);
    chase_share_vec = zeros(R, 1);
    ext_credit_used = false(R, 1);

    for i = 1:R
        di = draw_di_from_q1(q1_result, stream, cfg);
        rt = draw_risk_type(gender, age, stream);

        sim_args = {};
        if ~isempty(opt.rho), sim_args = [sim_args, {'rho', opt.rho}]; end
        if ~isempty(opt.sigma_h), sim_args = [sim_args, {'sigma_h', opt.sigma_h}]; end
        if ~isempty(fieldnames(opt.hold_overrides))
            sim_args = [sim_args, {'hold_overrides', opt.hold_overrides}];
        end
        if ~isempty(fieldnames(opt.inducement))
            sim_args = [sim_args, {'inducement', opt.inducement}];
        end

        res = simulate_year(di, gender, age, rt, stream, cfg, sim_args{:});

        nets(i) = res.annual_net;
        handles(i) = res.annual_handle;
        cash_handles(i) = res.annual_cash_handle;
        ext_credit_handles(i) = res.annual_ext_credit_handle;
        bonus_handles(i) = res.annual_bonus_handle;
        ever_chasing_flags(i) = res.ever_chasing;
        months_chasing_vec(i) = res.months_chasing;
        chase_share_vec(i) = res.chase_handle_share;
        ext_credit_used(i) = res.annual_ext_credit_handle > 0;

        loss = max(0, -res.annual_net);
        if di <= 0
            loss_5_flags(i) = loss > 0;
            loss_10_flags(i) = loss > 0;
        else
            loss_5_flags(i) = loss > 0.05 * di;
            loss_10_flags(i) = loss > 0.10 * di;
        end
    end

    nets_s = sort(nets);
    n = R;
    total_net = sum(nets);
    total_handle = sum(handles);

```

```
    r.persona_id = persona.id;
    r.gender = gender;
    r.age = age;
    r.di = di_median;
    r.p_active = p_active;
    r.median_net = nets_s(floor(n/2) + 1);
    r.mean_net = total_net / n;
    r.p10 = nets_s(max(1, floor(0.10 * n)));
    r.p90 = nets_s(min(ceil(0.90 * n), n));
    r.p_win = sum(nets > 0) / n;
    r.p_loss_5pct_di = sum(loss_5_flags) / n;
    r.p_loss_10pct_di = sum(loss_10_flags) / n;
    r.mean_handle = total_handle / n;
    r.mean_cash_handle = mean(cash_handles);
    r.mean_ext_credit_handle = mean(ext_credit_handles);
    r.mean_bonus_handle = mean(bonus_handles);
    r.p_ext_credit_used = sum(ext_credit_used) / n;
    if total_handle > 0
        r.sim_hold = -total_net / total_handle;
    else
        r.sim_hold = 0;
    end
    r.di_sigma_e = cfg.sigma_e;
    r.di_p_neg_q1 = q1_result.p_neg;
    r.p_ever_chasing = sum(ever_chasing_flags) / n;
    r.mean_months_chasing = mean(months_chasing_vec);
    r.mean_chase_handle_share = mean(chase_share_vec);
end


% Risk Type Comparison

function results = run_risk_type_comparison(di, cfg, varargin)

    p = inputParser();
    addParameter(p, 'gender', 'Male');
    addParameter(p, 'age', 30);
    addParameter(p, 'R', 1000);
    addParameter(p, 'seed', 42);
    addParameter(p, 'chasing_disabled', false);
    parse(p, varargin{:});
    opt = p.Results;

    for rt_cell = {'Low', 'Med', 'High'}
        rt = rt_cell{1};
        stream = RandStream('mt19937ar', 'Seed', opt.seed);
        nets = zeros(opt.R, 1);
        handles = zeros(opt.R, 1);
        ec = false(opt.R, 1);
        for i = 1:opt.R
            res = simulate_year(di, opt.gender, opt.age, rt, stream, cfg, ...
                                'chasing_disabled', opt.chasing_disabled);
            nets(i) = res.annual_net;
            handles(i) = res.annual_handle;
            ec(i) = res.ever_chasing;
        end
        nets_s = sort(nets);
        handles_s = sort(handles);
        n = opt.R;
        results.(rt).median_handle = handles_s(floor(n/2) + 1);
        results.(rt).median_net = nets_s(floor(n/2) + 1);
        results.(rt).mean_net = mean(nets);
        results.(rt).p_win = sum(nets > 0) / n;
        results.(rt).p_loss_2k = sum(-nets > 2000) / n;
        results.(rt).p_ever_chasing = sum(ec) / n;
    end
end


% Population Simulation

function pop = run_population_sim(cfg, q1_api, varargin)

    p = inputParser();
    addParameter(p, 'N', 10000);
    addParameter(p, 'seed', 42);
    addParameter(p, 'sigma_h', []);
    addParameter(p, 'chasing_disabled', false);
    addParameter(p, 'inducement', struct());
    addParameter(p, 'use_q1_di', true);
    parse(p, varargin{:});
    opt = p.Results;
```

```
610
611     stream = RandStream('mt19937ar', 'Seed', opt.seed);
612     cells = get_demo_cells();
613     di_approx = get_di_approx();
614
615     if opt.use_q1_di
616         P_US = q1_api.personas_us();
617         cell_persona_ids = containers.Map();
618         cell_persona_ids('Male_18-34') = {{'P1','P2'}};
619         cell_persona_ids('Male_35-49') = {{'P3','P4'}};
620         cell_persona_ids('Male_50+') = {{'P5','P6'}};
621         cell_persona_ids('Female_18-34') = {{'P7'}};
622         cell_persona_ids('Female_35-49') = {{'P7'}};
623         cell_persona_ids('Female_50+') = {{'P6'}};
624
625         us_by_id = containers.Map();
626         for i = 1:numel(P_US)
627             us_by_id(P_US(i).id) = P_US(i);
628         end
629
630         q1_cache = containers.Map();
631         q1_by_cell = containers.Map();
632         ks = cell_persona_ids.keys();
633         for ci = 1:numel(ks)
634             cell_key = ks{ci};
635             pids_cell = cell_persona_ids(cell_key);
636             pids = pids_cell{1};
637             cell_results = {};
638             for j = 1:numel(pids)
639                 pid = pids{j};
640                 if ~q1_cache.isKey(pid)
641                     q1_cache(pid) = q1_api.run_us_persona(us_by_id(pid));
642                 end
643                 cell_results{end+1} = q1_cache(pid); %#ok<AGROW>
644             end
645             q1_by_cell(cell_key) = cell_results;
646         end
647     end
648
649     all_nets = [];
650     all_handles = [];
651     all_ever_chasing = [];
652     all_months_chasing = [];
653     all_dis = [];
654     all_ext_credit_used = [];
655     all_ext_credit_handles = [];
656     all_bonus_handles = [];
657
658     for i = 1:opt.N
659         r = rand(stream);
660         cum = 0;
661         gender = 'Male'; ab = '18-34';
662         for ci = 1:size(cells, 1)
663             cum = cum + cells{ci, 3};
664             if r < cum
665                 gender = cells{ci, 1};
666                 ab = cells{ci, 2};
667                 break
668             end
669         end
670
671         switch ab
672             case '18-34', age = 18 + floor(rand(stream) * 17);
673             case '35-49', age = 35 + floor(rand(stream) * 15);
674             otherwise, age = 50 + floor(rand(stream) * 26);
675         end
676
677         pr = get_participation_rate(gender, age, cfg, 'US');
678         if rand(stream) >= pr
679             continue
680         end
681
682         cell_key = [gender '_' ab];
683         if opt.use_q1_di
684             cr = q1_by_cell(cell_key);
685             idx = 1 + floor(rand(stream) * numel(cr));
686             idx = min(idx, numel(cr));
687             q1_result = cr{idx};
688             di = draw_di_from_q1(q1_result, stream, cfg);
689         else
690             di = di_approx(cell_key);
691         end
```

```
        rt = draw_risk_type(gender, age, stream);

        sim_args = {};
        if ~isempty(opt.sigma_h)
            sim_args = [sim_args, {'sigma_h', opt.sigma_h}];
        end
        if opt.chasing_disabled
            sim_args = [sim_args, {'chasing_disabled', true}];
        end
        if ~isempty(fieldnames(opt.inducement))
            sim_args = [sim_args, {'inducement', opt.inducement}];
        end

        res = simulate_year(di, gender, age, rt, stream, cfg, sim_args{:});

        all_nets(end+1) = res.annual_net; %#ok<AGROW>
        all_handles(end+1) = res.annual_handle; %#ok<AGROW>
        all_ever_chasing(end+1) = res.ever_chasing; %#ok<AGROW>
        all_months_chasing(end+1) = res.months_chasing; %#ok<AGROW>
        all_dis(end+1) = di; %#ok<AGROW>
        all_ext_credit_used(end+1) = res.annual_ext_credit_handle > 0; %#ok<AGROW>
        all_ext_credit_handles(end+1) = res.annual_ext_credit_handle; %#ok<AGROW>
        all_bonus_handles(end+1) = res.annual_bonus_handle; %#ok<AGROW>
    end

    n_b = numel(all_nets);
    if n_b == 0
        pop.n_population = opt.N;
        pop.n_bettors = 0;
        return
    end

    losses = sort(max(0, -all_nets), 'descend');
    total_loss = sum(losses);
    total_handle = sum(all_handles);
    total_net = sum(all_nets);
    all_nets_s = sort(all_nets);
    gini_l = gini_nonneg(losses);
    gini_h = gini_nonneg(all_handles);
    ggr = -total_net;
    sim_hold = ggr / max(total_handle, 1);

    top_share = @(pct) sum(losses(1:max(1, floor(n_b*pct)))) / ...
                    max(total_loss, 1);

    burdens = zeros(1, n_b);
    loss_gt_di = false(1, n_b);
    b_gt_5 = 0; b_gt_10 = 0;
    for j = 1:n_b
        loss_j = max(0, -all_nets(j));
        if all_dis(j) <= 0
            burdens(j) = 9.99;
            loss_gt_di(j) = loss_j > 0;
        else
            burdens(j) = loss_j / all_dis(j);
            loss_gt_di(j) = loss_j > all_dis(j);
        end
        if burdens(j) > 0.05, b_gt_5 = b_gt_5 + 1; end
        if burdens(j) > 0.10, b_gt_10 = b_gt_10 + 1; end
    end
    burdens_s = sort(burdens);

    pop.n_population = opt.N;
    pop.n_bettors = n_b;
    pop.participation_rate = n_b / opt.N;
    pop.mean_handle = total_handle / n_b;
    pop.total_handle = total_handle;
    pop.total_losses = total_loss;
    pop.ggr = ggr;
    pop.sim_hold = sim_hold;
    pop.median_net = all_nets_s(floor(n_b/2) + 1);
    pop.mean_net = mean(all_nets);
    pop.p_win = sum(all_nets > 0) / n_b;
    pop.top1_loss_share = top_share(0.01);
    pop.top5_loss_share = top_share(0.05);
    pop.top10_loss_share = top_share(0.10);
    pop.bot50_loss_share = sum(losses(floor(n_b/2)+1:end)) / max(total_loss, 1);
    pop.gini_loss = gini_l;
    pop.gini_handle = gini_h;
    pop.p99_loss = losses(max(1, floor(n_b/100)));
    pop.p95_loss = losses(max(1, floor(n_b/20)));
```

```
    pop.p90_loss = losses(max(1, floor(n_b/10)));
    pop.p_ever_chasing = sum(all_ever_chasing) / n_b;
    pop.mean_months_chasing = mean(all_months_chasing);
    pop.p_ext_credit_used = sum(all_ext_credit_used) / n_b;
    pop.mean_ext_credit_handle = mean(all_ext_credit_handles);
    pop.mean_bonus_handle = mean(all_bonus_handles);
    pop.median_burden = burdens_s(floor(n_b/2) + 1);
    pop.p90_burden = burdens_s(min(floor(0.90*n_b)+1, n_b));
    pop.p95_burden = burdens_s(min(floor(0.95*n_b)+1, n_b));
    pop.p_burden_gt_5pct = b_gt_5 / n_b;
    pop.p_burden_gt_10pct = b_gt_10 / n_b;
    pop.p_loss_gt_di = sum(loss_gt_di) / n_b;
end


% Sensitivity Analysis

function results = sensitivity_hold(persona, q1_result, cfg, q1_api)
    scenarios.Low_hold = struct('S',0.040, 'P',0.144, 'I',0.06);
    scenarios.Baseline = struct();
    scenarios.High_hold = struct('S',0.070, 'P',0.240, 'I',0.10);
    fns = fieldnames(scenarios);
    for i = 1:numel(fns)
        label = fns{i};
        ho = scenarios.(label);
        r = run_persona_q2(persona, q1_result, cfg, q1_api, ...
                           'hold_overrides', ho);
        results.(label) = r;
    end
end


function results = sensitivity_parlay_mix(di, cfg, varargin)
    p = inputParser();
    addParameter(p, 'gender', 'Male');
    addParameter(p, 'age', 28);
    addParameter(p, 'R', 2000);
    addParameter(p, 'seed', 42);
    addParameter(p, 'inplay_share', 0.20);
    parse(p, varargin{:});
    opt = p.Results;

    for p_share = [0.10, 0.25, 0.50]
        s_share = max(0, 1.0 - opt.inplay_share - p_share);
        mix = struct('S', s_share, 'P', p_share, 'I', opt.inplay_share);
        bh = mix.S*cfg.hold.S + mix.P*cfg.hold.P + mix.I*cfg.hold.I;
        stream = RandStream('mt19937ar', 'Seed', opt.seed);
        nets = zeros(opt.R, 1);
        for i = 1:opt.R
            res = simulate_year(di, opt.gender, opt.age, 'Med', stream, cfg, ...
                                'bet_mix_overrides', mix);
            nets(i) = res.annual_net;
        end
        nets_s = sort(nets);
        n = opt.R;
        fname = sprintf('p%d', round(p_share*100));
        results.(fname).parlay_share = p_share;
        results.(fname).blended_hold = bh;
        results.(fname).median_net = nets_s(floor(n/2)+1);
        results.(fname).mean_net = mean(nets);
        results.(fname).p_loss_2k = sum(-nets > 2000) / n;
    end
end


function results = sensitivity_rho(di, cfg, varargin)
    p = inputParser();
    addParameter(p, 'gender', 'Male');
    addParameter(p, 'age', 28);
    addParameter(p, 'R', 1000);
    addParameter(p, 'seed', 42);
    parse(p, varargin{:});
    opt = p.Results;

    for rho_val = [0.15, 0.30, 0.50]
        stream = RandStream('mt19937ar', 'Seed', opt.seed);
        nets = zeros(opt.R, 1);
        for i = 1:opt.R
            res = simulate_year(di, opt.gender, opt.age, 'High', stream, ...
                                cfg, 'rho', rho_val);
            nets(i) = res.annual_net;
        end
```

```
        n = opt.R;
        fname = sprintf('rho_%d', round(rho_val*100));
        results.(fname).mean_net = mean(nets);
        results.(fname).p_loss_5k = sum(-nets > 5000) / n;
    end
end


function results = sensitivity_gini(cfg, q1_api, varargin)
    p = inputParser();
    addParameter(p, 'N', 20000);
    addParameter(p, 'seed', 42);
    parse(p, varargin{:});
    opt = p.Results;

    for sh = [1.00, 1.20, 1.50, 1.85]
        pop_r = run_population_sim(cfg, q1_api, 'N', opt.N, ...
                                   'seed', opt.seed, 'sigma_h', sh);
        fname = sprintf('sh_%d', round(sh*100));
        results.(fname).sigma_h = sh;
        results.(fname).mean_handle = pop_r.mean_handle;
        results.(fname).sim_hold = pop_r.sim_hold;
        results.(fname).top5_loss_share = pop_r.top5_loss_share;
        results.(fname).top10_loss_share = pop_r.top10_loss_share;
        results.(fname).gini_loss = pop_r.gini_loss;
        results.(fname).gini_handle = pop_r.gini_handle;
    end
end


% Full Pipeline

function results = run_all(~)
    [~, q1] = q1_model();
    cfg = q2_config();

    P_US = q1.personas_us();
    P_UK = q1.personas_uk();
    q1_us = containers.Map();
    q1_uk = containers.Map();
    for i = 1:numel(P_US)
        q1_us(P_US(i).id) = q1.run_us_persona(P_US(i));
    end
    for i = 1:numel(P_UK)
        q1_uk(P_UK(i).id) = q1.run_uk_persona(P_UK(i));
    end

    q2_us = containers.Map();
    for i = 1:numel(P_US)
        r = run_persona_q2(P_US(i), q1_us(P_US(i).id), cfg, q1, ...
                           'country', 'US');
        q2_us(P_US(i).id) = r;
    end

    q2_uk = containers.Map();
    for i = 1:numel(P_UK)
        r = run_persona_q2(P_UK(i), q1_uk(P_UK(i).id), cfg, q1, ...
                           'country', 'UK');
        q2_uk(P_UK(i).id) = r;
    end

    p2_q1 = q1_us('P2');
    p2_di = p2_q1.median_di;
    rt_comp = run_risk_type_comparison(p2_di, cfg);

    pop = run_population_sim(cfg, q1, 'N', 10000);

    hold_s = sensitivity_hold(P_US(3), q1_us('P3'), cfg, q1);
    parlay_s = sensitivity_parlay_mix(p2_di, cfg);
    rho_s = sensitivity_rho(p2_di, cfg);
    gini_s = sensitivity_gini(cfg, q1);

    results.personas_us = map_to_struct(q2_us);
    results.personas_uk = map_to_struct(q2_uk);
    results.risk_type_comparison = rt_comp;
    results.population = pop;
    results.sensitivity.hold = hold_s;
    results.sensitivity.parlay_mix = parlay_s;
    results.sensitivity.rho = rho_s;
    results.sensitivity.concentration = gini_s;

    out_path = fullfile(fileparts(mfilename('fullpath')), '..', ...
```

```
                    'q2_results_matlab.json');
    fid = fopen(out_path, 'w');
    fwrite(fid, jsonencode(results));
    fclose(fid);
end
```

A.3 Q3: Financial Impact Model

Listing 3: q3_impact.m – Burden, debt, wealth drag, and scenario analysis

```
function [results, api] = q3_impact()
% Q3 Impact Layer -- financial harm quantification

    api.fv_annuity = @fv_annuity;
    api.simulate_persona_q3 = @simulate_persona_q3;
    api.quantile_val = @quantile_val;
    api.mean_val = @mean_val;

    [~, q1_api] = q1_model_api();
    cfg0 = q2_config();

    repo_root = fullfile(fileparts(mfilename('fullpath')), '..', '..');
    q1_path = fullfile(repo_root, 'analysis', 'q1_results.json');
    q1_data = jsondecode(fileread(q1_path));

    us_q1 = index_by_id(q1_data.us);
    uk_q1 = index_by_id(q1_data.uk);

    P_US = q1_api.personas_us();
    P_UK = q1_api.personas_uk();
    us_meta = index_personas(P_US);
    uk_meta = index_personas(P_UK);

    cfg1 = cfg0;
    cfg1.credit_buffer_low = 0.0;
    cfg1.credit_buffer_med = 0.0;
    cfg1.credit_buffer_high = 0.0;
    cfg1.credit_buffer = struct('Low', 0.0, 'Med', 0.0, 'High', 0.0);

    r_real = 0.05;
    apr_us = 0.24;
    apr_uk = 0.0861;

    scenarios = struct( ...
        'key', {'S0', 'S1', 'S2'}, ...
        'label', {'Baseline', 'No credit (hard deposit limit)', ...
                  'Targeted inducements (liquidity injection)'}, ...
        'cfg', {cfg0, cfg1, cfg0}, ...
        'inducement', {struct(), struct(), ...
                       struct('offer_prob', 0.10, 'offer_bonus', 100.0, 'offer_tau', 0.10)});

    R = 1000;
    seed = 42;

    us_results = struct();
    uk_results = struct();

    for si = 1:numel(scenarios)
        scen = scenarios(si);

        us_pids = fieldnames(us_meta);
        for pi = 1:numel(us_pids)
            pid = us_pids{pi};
            if ~isfield(us_q1, pid), continue; end
            meta = us_meta.(pid);
            q1_row = us_q1.(pid);
            out = simulate_persona_q3(pid, meta, q1_row, scen, 'US', ...
                                      R, seed, r_real, apr_us);
            field_key = sprintf('%s_%s', pid, scen.key);
            us_results.(field_key) = out;
        end

        uk_pids = fieldnames(uk_meta);
        for pi = 1:numel(uk_pids)
            pid = uk_pids{pi};
            if ~isfield(uk_q1, pid), continue; end
            meta = uk_meta.(pid);
            q1_row = uk_q1.(pid);
            out = simulate_persona_q3(pid, meta, q1_row, scen, 'UK', ...
```

```
                                    R, seed, r_real, apr_uk);
            field_key = sprintf('%s_%s', pid, scen.key);
            uk_results.(field_key) = out;
        end
    end

    meta_out = struct( ...
        'generated_by', 'analysis/matlab/q3_impact.m', ...
        'r_real', r_real, ...
        'apr_us', apr_us, ...
        'apr_uk', apr_uk, ...
        'note_uk', 'UK is a comparison case; mechanics US-calibrated.');

    scenario_list = cell(1, numel(scenarios));
    for si = 1:numel(scenarios)
        scen = scenarios(si);
        ind = scen.inducement;
        if ~isfield(ind, 'offer_prob')
            ind = struct();
        end
        scenario_list{si} = struct('key', scen.key, 'label', scen.label, ...
                                   'inducement', ind);
    end

    results = struct();
    results.meta = meta_out;
    results.results = struct('US', us_results, 'UK', uk_results);
    results.scenarios = scenario_list;

    out_path = fullfile(repo_root, 'analysis', 'q3_results_matlab.json');
    fid = fopen(out_path, 'w');
    fwrite(fid, jsonencode(results));
    fclose(fid);
end


% Future Value of Annuity

function fv = fv_annuity(payment, rate, years)
    if payment <= 0 || years <= 0
        fv = 0.0;
        return
    end
    if rate <= 0
        fv = payment * years;
        return
    end
    fv = payment * (((1.0 + rate)^years - 1.0) / rate);
end


% Persona-Level Q3 Simulation

function out = simulate_persona_q3(persona_id, persona_meta, q1_row, ...
                                   scenario, country, R, seed, r_real, apr)

    retire_age = 65;
    eps_di = 100.0;
    cfg = scenario.cfg;

    gender = persona_gender(persona_id);
    age = round(persona_meta.age);
    p_active = get_participation_rate(gender, age, cfg, country);
    ati = q1_row.ati;

    nets = zeros(R, 1);
    losses = zeros(R, 1);
    burdens = zeros(R, 1);
    burdens_pos_di = [];
    p_di_nonpos_count = 0;
    months_ess_lost = zeros(R, 1);

    ext_credit_used = false(R, 1);
    bonus_used = false(R, 1);
    loss_credit_arr = zeros(R, 1);
    loss_cash_arr = zeros(R, 1);
    fv_drag_arr = zeros(R, 1);
    fv_debt_arr = zeros(R, 1);
    burden_gt_5 = 0;
    burden_gt_10 = 0;

    years_to_retire = max(retire_age - age, 0);
```

```
152
153     rng(seed, 'twister');
154
155     for r_idx = 1:R
156         di = draw_di_from_q1(q1_row, cfg.sigma_e);
157
158         rt = draw_risk_type(gender, age);
159
160         res = simulate_year(di, gender, age, rt, scenario.inducement, cfg);
161
162         net = res.annual_net;
163         loss = max(0.0, -net);
164
165         H = res.annual_handle;
166         H_ext = res.annual_ext_credit_handle;
167         H_bonus = res.annual_bonus_handle;
168         s_credit = H_ext / max(H, 1.0);
169
170         L_credit = loss * s_credit;
171         L_cash = loss - L_credit;
172
173         if di > 0
174             burden = loss / max(di, eps_di);
175             burdens_pos_di(end+1) = loss / di; %#ok<AGROW>
176         else
177             burden = 9.99;
178             p_di_nonpos_count = p_di_nonpos_count + 1;
179         end
180
181         monthly_ess = max((ati - di) / 12.0, 1e-6);
182         if monthly_ess > 0
183             m_lost = loss / monthly_ess;
184         else
185             m_lost = Inf;
186         end
187
188         nets(r_idx) = net;
189         losses(r_idx) = loss;
190         burdens(r_idx) = burden;
191         months_ess_lost(r_idx) = m_lost;
192
193         ext_credit_used(r_idx) = (H_ext > 1.0);
194         bonus_used(r_idx) = (H_bonus > 1.0);
195         loss_credit_arr(r_idx) = L_credit;
196         loss_cash_arr(r_idx) = L_cash;
197
198         fv_drag_arr(r_idx) = fv_annuity(L_cash, r_real, years_to_retire);
199         fv_debt_arr(r_idx) = fv_annuity(L_credit, apr, 5);
200
201         if burden > 0.05, burden_gt_5 = burden_gt_5 + 1; end
202         if burden > 0.10, burden_gt_10 = burden_gt_10 + 1; end
203     end
204
205     nets_s = sort(nets);
206     losses_s = sort(losses);
207     burdens_s = sort(burdens);
208     months_s = sort(months_ess_lost);
209     if ~isempty(burdens_pos_di)
210         burdens_pos_s = sort(burdens_pos_di);
211     else
212         burdens_pos_s = [];
213     end
214
215     p_ext = sum(ext_credit_used) / R;
216     p_bon = sum(bonus_used) / R;
217
218     if isfield(persona_meta, 'salary')
219         salary = persona_meta.salary;
220     else
221         salary = NaN;
222     end
223
224     out.persona_id = persona_id;
225     out.country = country;
226     out.age = age;
227     out.gender = gender;
228     out.salary = salary;
229     out.p_active = p_active;
230
231     out.q1.median_di = q1_row.median_di;
232     out.q1.p_di_neg = q1_row.p_neg;
233     out.q1.ati = ati;
```

```
234
235     q3 = struct();
236     q3.R = R;
237     q3.scenario = scenario.key;
238     q3.median_net = nets_s(floor(numel(nets_s)/2) + 1);
239     q3.mean_net = mean_val(nets);
240     q3.p10_net = nets_s(max(1, floor(0.10 * numel(nets_s))));
241     q3.p90_net = nets_s(min(ceil(0.90 * numel(nets_s)), numel(nets_s)));
242     q3.median_loss = losses_s(floor(numel(losses_s)/2) + 1);
243     q3.mean_loss = mean_val(losses);
244     q3.median_burden = burdens_s(floor(numel(burdens_s)/2) + 1);
245     q3.p90_burden = quantile_val(burdens, 0.90);
246     q3.p95_burden = quantile_val(burdens, 0.95);
247     q3.p_di_nonpos_sim = p_di_nonpos_count / R;
248
249     if ~isempty(burdens_pos_s)
250         nb = numel(burdens_pos_s);
251         q3.median_burden_pos_di = burdens_pos_s(floor(nb/2) + 1);
252         q3.p90_burden_pos_di = quantile_val(burdens_pos_di, 0.90);
253         q3.p95_burden_pos_di = quantile_val(burdens_pos_di, 0.95);
254     else
255         q3.median_burden_pos_di = NaN;
256         q3.p90_burden_pos_di = NaN;
257         q3.p95_burden_pos_di = NaN;
258     end
259
260     q3.p_burden_gt_5pct = burden_gt_5 / R;
261     q3.p_burden_gt_10pct = burden_gt_10 / R;
262     q3.median_months_ess_lost = months_s(floor(numel(months_s)/2) + 1);
263     q3.p90_months_ess_lost = quantile_val(months_ess_lost, 0.90);
264
265     q3.p_ext_credit_used = p_ext;
266     q3.p_bonus_used = p_bon;
267     q3.mean_loss_credit = mean_val(loss_credit_arr);
268     q3.mean_loss_cash = mean_val(loss_cash_arr);
269
270     q3.r_real = r_real;
271     q3.apr = apr;
272     q3.years_to_retire = years_to_retire;
273     q3.median_fv_drag = quantile_val(fv_drag_arr, 0.50);
274     q3.p90_fv_drag = quantile_val(fv_drag_arr, 0.90);
275     q3.median_fv_debt = quantile_val(fv_debt_arr, 0.50);
276     q3.p90_fv_debt = quantile_val(fv_debt_arr, 0.90);
277
278     q3.p_debt_population = p_active * p_ext;
279     q3.mean_loss_population = p_active * mean_val(losses);
280
281     out.q3 = q3;
282 end
283
284
285 % Q2 Model Functions
286
287 function gender = persona_gender(pid)
288     map = struct('P1','Male', 'P2','Male', 'P3','Male', 'P4','Male', ...
289                  'P5','Male', 'P6','Male', 'P7','Female', 'P8','Male', ...
290                  'P9','Male', 'P10','Male');
291     if isfield(map, pid)
292         gender = map.(pid);
293     else
294         gender = 'Male';
295     end
296 end
297
298
299 function ab = age_bin(age)
300     if age < 35
301         ab = 'a18_34';
302     elseif age < 50
303         ab = 'a35_49';
304     else
305         ab = 'a50p';
306     end
307 end
308
309
310 function p = get_participation_rate(gender, age, cfg, country)
311     ab = age_bin(age);
312
313     ar_us = struct('Male_a18_34', 0.55, 'Male_a35_49', 0.50, 'Male_a50p', 0.20, ...
314                    'Female_a18_34', 0.15, 'Female_a35_49', 0.15, 'Female_a50p', 0.05);
315
```

```
    key = sprintf('%s_%s', gender, ab);
    if strcmp(country, 'UK')
        ar_base = ar_us.(key);
        base_mean = 0.55*0.135 + 0.50*0.125 + 0.20*0.220 + ...
                    0.15*0.130 + 0.15*0.125 + 0.05*0.265;
        scale_c = 0.22 / base_mean;
        account = min(1.0, max(0.0, ar_base * scale_c));
    else
        if isfield(ar_us, key)
            account = ar_us.(key);
        else
            account = 0.10;
        end
    end
    p = account * cfg.p_active;
end


function dist = get_risk_dist(gender, age)
    default_dist = [0.55, 0.30, 0.15];

    rd = struct();
    rd.Male_a18_34 = [0.40, 0.35, 0.25];
    rd.Male_a35_49 = [0.50, 0.30, 0.20];
    rd.Male_a50p = [0.70, 0.20, 0.10];
    rd.Female_a18_34 = [0.60, 0.30, 0.10];
    rd.Female_a35_49 = [0.65, 0.25, 0.10];
    rd.Female_a50p = [0.80, 0.15, 0.05];

    ab = age_bin(age);
    key = sprintf('%s_%s', gender, ab);
    if isfield(rd, key)
        dist = rd.(key);
    else
        dist = default_dist;
    end
end


function rt = draw_risk_type(gender, age)
    dist = get_risk_dist(gender, age);
    r = rand();
    if r < dist(1)
        rt = 'Low';
    elseif r < dist(1) + dist(2)
        rt = 'Med';
    else
        rt = 'High';
    end
end


function di = draw_di_from_q1(q1_row, sigma_e)
    ati = q1_row.ati;
    mu_e = q1_row.mu_e;
    eps = randn() * sigma_e - sigma_e^2 / 2.0;
    essentials = mu_e * exp(eps);
    di = ati - essentials;
end


function s = sigmoid(x)
    if x < -20, s = 0.0; return; end
    if x > 20, s = 1.0; return; end
    s = 1.0 / (1.0 + exp(-x));
end


function res = simulate_year(di, gender, age, risk_type, inducement, cfg)
    hold_map = cfg.hold;
    payout_map = cfg.payout;
    win_prob_map = cfg.win_prob;
    avg_stake_map = cfg.avg_stake;
    seasonality = cfg.seasonality;
    gamma_sum = sum(seasonality);

    mu = cfg.mu_h + cfg.delta_r.(risk_type);
    ln_H = randn() * cfg.sigma_h + mu;
    H_annual = exp(ln_H);

    if strcmp(risk_type, 'Low')
        beta_1_eff = cfg.beta_1 * 0.5;
```

```
    else
        beta_1_eff = cfg.beta_1;
    end

    state = 0;
    cum_loss = 0.0;
    total_net = 0.0;
    total_handle = 0.0;

    ever_chasing = false;
    months_chasing = 0;
    chase_entries = 0;
    chase_handle = 0.0;

    total_cash_handle = 0.0;
    total_ext_credit_handle = 0.0;
    total_bonus_handle = 0.0;

    bonus_buffer = 0.0;
    if isfield(inducement, 'offer_prob')
        offer_prob = inducement.offer_prob;
        offer_bonus = inducement.offer_bonus;
        offer_tau = inducement.offer_tau;
    else
        offer_prob = 0.0;
        offer_bonus = 0.0;
        offer_tau = 0.0;
    end

    bet_types = {'S', 'P', 'I'};

    monthly_di = max(di / 12.0, 0);
    cash_cap_base = cfg.kappa.(risk_type) * monthly_di;
    ext_credit_cap = cfg.credit_buffer.(risk_type);

    base_mix = cfg.bet_mix.(risk_type);

    for t = 1:12
        base_share = seasonality(t) / gamma_sum;
        noise = exp(randn() * cfg.sigma_eta);
        v_month = H_annual * base_share * noise;

        is_chasing = (state == 1);
        if is_chasing
            v_month = v_month * exp(cfg.delta_v);
        end

        v_month_pre_cap = v_month;
        cap = cash_cap_base + ext_credit_cap + bonus_buffer;
        if cap >= 0
            v_month = min(v_month, cap);
        end

        mix_S = base_mix.S;
        mix_P = base_mix.P;
        mix_I = base_mix.I;
        if is_chasing
            mix_S = max(0.0, mix_S + cfg.chase_shift.S);
            mix_P = max(0.0, mix_P + cfg.chase_shift.P);
            mix_I = max(0.0, mix_I + cfg.chase_shift.I);
            total_w = mix_S + mix_P + mix_I;
            if total_w > 0
                mix_S = mix_S / total_w;
                mix_P = mix_P / total_w;
                mix_I = mix_I / total_w;
            end
        end
        mix_arr = [mix_S, mix_P, mix_I];

        month_net = 0.0;
        month_handle = 0.0;
        for ki = 1:3
            k = bet_types{ki};
            handle_k = v_month * mix_arr(ki);
            if handle_k < 1.0, continue; end

            n_bets = max(1, round(handle_k / avg_stake_map.(k)));
            actual_stake = handle_k / n_bets;
            wp = win_prob_map.(k);
            po = payout_map.(k);

            for bi = 1:n_bets
```

```
            if rand() < wp
                month_net = month_net + po * actual_stake;
            else
                month_net = month_net - actual_stake;
            end
            month_handle = month_handle + actual_stake;
        end
    end

    total_net = total_net + month_net;
    total_handle = total_handle + month_handle;

    cash_handle = min(month_handle, cash_cap_base);
    ext_credit_handle = min(max(month_handle - cash_cap_base, 0.0), ext_credit_cap);
    bonus_handle_val = max(month_handle - cash_cap_base - ext_credit_cap, 0.0);
    if ext_credit_handle < 1e-6, ext_credit_handle = 0.0; end
    if bonus_handle_val < 1e-6, bonus_handle_val = 0.0; end
    total_cash_handle = total_cash_handle + cash_handle;
    total_ext_credit_handle = total_ext_credit_handle + ext_credit_handle;
    total_bonus_handle = total_bonus_handle + bonus_handle_val;

    if is_chasing
        months_chasing = months_chasing + 1;
        chase_handle = chase_handle + month_handle;
    end

    if month_net < 0
        cum_loss = cum_loss + abs(month_net);
    end

    if state == 0
        loss_ratio = cum_loss / max(di, cfg.epsilon);
        p_chase = sigmoid(cfg.beta_0 + beta_1_eff * loss_ratio);
        if rand() < p_chase
            state = 1;
            ever_chasing = true;
            chase_entries = chase_entries + 1;
        end
    else
        if rand() < cfg.rho
            state = 0;
        end
    end

    if offer_prob > 0.0 && offer_bonus > 0.0
        hit_wall = v_month_pre_cap > (cash_cap_base + ext_credit_cap);
        if offer_tau > 0
            stressed = (cum_loss / max(di, cfg.epsilon)) > offer_tau;
        else
            stressed = false;
        end
        if hit_wall || stressed
            if rand() < offer_prob
                bonus_buffer = offer_bonus;
            else
                bonus_buffer = 0.0;
            end
        else
            bonus_buffer = 0.0;
        end
    else
        bonus_buffer = 0.0;
    end
end

if total_handle > 0
    chase_handle_share = chase_handle / total_handle;
else
    chase_handle_share = 0.0;
end

res.annual_handle = total_handle;
res.annual_net = total_net;
res.cum_loss = cum_loss;
res.ever_chasing = ever_chasing;
res.months_chasing = months_chasing;
res.chase_entries = chase_entries;
res.chase_handle_share = chase_handle_share;
res.annual_cash_handle = total_cash_handle;
res.annual_ext_credit_handle = total_ext_credit_handle;
res.annual_bonus_handle = total_bonus_handle;
end
```

```
562
563
564 % Statistical Helpers
565
566 function q = quantile_val(xs, p)
567     if isempty(xs)
568         q = NaN;
569         return
570     end
571     xs_s = sort(xs(:));
572     idx = max(1, floor(p * (numel(xs_s) - 1)) + 1);
573     idx = min(idx, numel(xs_s));
574     q = xs_s(idx);
575 end
576
577
578 function m = mean_val(xs)
579     if isempty(xs)
580         m = NaN;
581     else
582         m = sum(xs(:)) / numel(xs);
583     end
584 end
585
586
587 % Data Loading Helpers
588
589 function idx = index_by_id(arr)
590     idx = struct();
591     if isstruct(arr)
592         for i = 1:numel(arr)
593             pid = arr(i).id;
594             idx.(pid) = arr(i);
595         end
596     end
597 end
598
599
600 function idx = index_personas(P)
601     idx = struct();
602     for i = 1:numel(P)
603         idx.(P(i).id) = P(i);
604     end
605 end
606
607
608 function [~, api] = q1_model_api()
609     evalc('[~, api] = q1_model();');
610 end
```

A.4 Q2 Residual Diagnostics

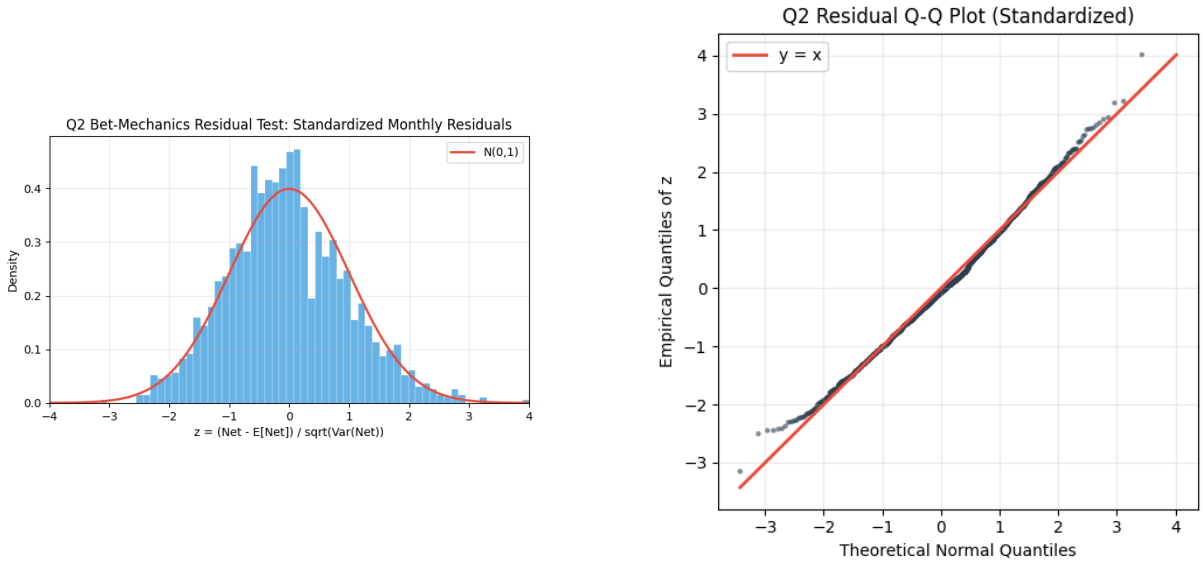

Figure A.1: Residual diagnostics for bettors with 50+ annual bets. Left: standardized residual histogram closely tracks the $\mathcal{N}(0,1)$ density. Right: Q-Q plot confirms near-normal fit in the central 95%, with expected heavy tails at extremes.