MathWorks Math Modelling Challenge 2026

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

Team: Diart Olluri, Gyula Rábai, Stan Wancerski, Calvin Vu

1.0 Executive Summary

To policymakers and public-health regulators addressing online gambling risk,

The rapid expansion of online sports betting has created a policy problem that is simultaneously financial, behavioral, and structural. While online gambling generates tax revenue and consumer engagement, its harms are not distributed uniformly: the same wager can be a minor entertainment expense for one household and the start of a long-term financial spiral for another. To quantify these unequal effects, we developed a three-part mathematical framework that moves from short-run household affordability, to repeated betting behavior, to long-run economic recovery. Together, these models show that online gambling risk is best understood not as a single loss statistic, but as a cascade: financial fragility determines exposure, behavioral dynamics determine loss escalation, and funding method determines how long the resulting damage persists.

In **Playing with House Money**, we model the disposable income available to a representative individual after taxes and essential living costs, then test whether that surplus can withstand realistic emergency shocks. Our deterministic layer converts gross salary into monthly post-tax income using country-specific tax rules, regional essential-cost lookups, age-adjusted healthcare burdens, and OECD-modified household scaling. Utilities are treated as a seasonal sinusoidal cost, allowing disposable income to vary across the calendar year rather than being flattened into a single annual number. We then extend this with a stochastic shock layer in which emergency expenses arrive according to a Poisson process and have log-normal severity, capturing the empirical reality that most months are ordinary but a small number are financially catastrophic. This structure lets us distinguish between households that appear stable on average and households that are one bad month away from debt. In our reported cases, high-income households retain large, resilient surpluses, while the most vulnerable low-income family faces a **13.55% monthly probability of falling into deficit even before any gambling expenditure is introduced**. Sensitivity analysis further shows that salary and housing costs dominate baseline affordability, confirming that preexisting household conditions are the primary drivers of exposure to gambling harm.

In **Know the Spread**, we shift from household finance to bettor behavior by modeling gambling as an **agent-based Markov state automaton**. Each gambler is represented by a state vector containing current risk tolerance, bankroll, cumulative profit, and session-level performance. Age and gender initialize a baseline risk tolerance through quadratic demographic rules, and each wager updates the gambler's state: wins reset risk to baseline, while losses multiply risk through a chasing coefficient, explicitly encoding loss-chasing behavior. Current risk then determines both stake size and the type of betting profile selected, ranging from lower-risk bets to high-volatility parlays. Because sessions continue only while behavioral and financial continuation conditions are satisfied, the model generates emergent long-tail outcomes rather than assuming a fixed yearly loss rate. This produces several key insights consistent with your results: younger gamblers remain in riskier betting regimes for longer, loss-chasing sharply worsens outcomes, looser stopping rules increase drawdown risk, and under the current implementation, losses scale approximately proportionally with income. The code-level structure also makes the model transparent enough to identify implementation-specific behaviors - such as the constant effective-income factor and cumulative-profit overcounting - that materially affect the magnitude of simulated losses while preserving the conceptual strength of the framework.

In **Don't Break the Bank**, we translate gambling damage into a more policy-relevant measure: **Recovery Burden Months (RBM)**, the number of months of ordinary saving required to repair the harm caused by unsafe gambling. Rather than treating all losses identically, this model first computes the harmful shortfall that exceeds a safe monthly cushion, then allocates that shortfall across three financial compartments: depleted savings, missed investment growth, and debt accumulation. These compartments evolve under a system of differential equations, with missed wealth and debt compounding over time through investment returns and borrowing interest, respectively. This is the crucial long-run step of the analysis: two gamblers can lose the same amount in a given month, but the one who borrows to continue gambling incurs a much larger downstream burden than the one who cuts back spending from cash reserves. Your results show that debt-funded losses create an exponential recovery trap, that even modest underestimation of "safe" gambling capacity can trigger steep long-run burdens, and that the same milestone shock is substantially more punitive for lower-income individuals with limited monthly saving capacity. Sensitivity analysis confirms that monthly gambling loss, repayment capacity, and safe cushion size are the dominant drivers of recovery time. Thus, the final model demonstrates that the true cost of online gambling is not merely what is lost today, but how many months, or years, of future financial mobility are erased.

Taken together, our three models provide a coherent policy framework. The first identifies who is financially exposed, the second explains how behavioural dynamics convert exposure into loss, and the third measures how long those losses remain economically active after the betting stops. This layered structure allows regulators to move beyond broad warnings and toward targeted intervention: protect already-fragile households, constrain designs that intensify loss-chasing, and discourage debt-funded gambling before short-run losses become long-run structural damage.

2.0 Playing with House Money

2.1 Defining the Problem

The problem asks us to model a person's disposable income using their salary and demographics. This is defined as the left-over money after paying essentials, like food, housing and heal

2.2 Assumptions

To keep the model consistent, interpretable, and compatible with the available data, we make the following assumptions:

1. Salary is gross annual employment income.
2. Tax follows standard statutory rules.

Justification: Assuming standard W-2[1]/PAYE [2]filing with standard deductions captures the vast majority of the population while keeping the model computationally tractable.

3. Baseline essential costs represent necessary expenditure, not full observed consumption.

Justification: Standard consumer expenditure surveys show high-income households spending significantly more on food and housing due to discretionary lifestyle inflation (e.g., dining out, luxury homes). We assume baseline figures represent biological and spatial necessities to prevent the model from erroneously driving a wealthy persona's disposable income down to zero.

4. Baseline essential costs are anchored to a one-adult reference household.

Justification: Aggregate survey data inherently averages out family sizes (e.g., the UK average household is ~2.3 persons)[3]. Establishing a strict 1-adult baseline allows the model to cleanly scale up for any user-inputted family size without double-counting shared expenses.

5. Household costs scale according to the OECD-modified equivalence scale.

Justification: Two adults living together do not incur double the rent and heating costs of a single adult. The OECD-modified scale[4] (1.0 for the first adult, 0.5 for subsequent adults, 0.3 for children) is the established econometric standard for accurately modelling these shared economies of scale.

6. Adults share the household burden equally.

Justification: Intra-household financial agreements (who pays for what) are highly variable and unobservable. Assuming an equal mathematical split of the total equivalised household cost among working-age adults provides the most objective, unbiased individual financial burden.

7. Most essential categories are treated as stable across the year.

Justification: Core expenses such as rent, insurance, and baseline food budgets exhibit minimal month-to-month volatility for a fixed household. Treating them as constant fractions of the annual total simplifies the monthly conversion without sacrificing predictive accuracy.

8. Utilities are the only explicitly seasonal essential category.

Justification: Heating and cooling demands fluctuate wildly between summer and winter, directly impacting monthly cash flow. Because other essentials remain flat, utilities are the primary driver of predictable monthly expense variance that could impact short-term gambling liquidity.

9. Seasonal utility variation is smooth and periodic.

---

[1] https://www.irs.gov/filing/federal-income-tax-rates-and-brackets

[2] https://www.gov.uk/income-tax-rates

[3] https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/families/bulletins/familiesandhouseholds/2024

[4] https://www.ons.gov.uk/peoplepopulationandcommunity/personalandhouseholdfinances/incomeandwealth/compendium/familyspending/2015/chapter3equivalisedincome

Justification: Weather patterns naturally follow a sinusoidal curve.[5] Modelling utility costs with a smooth periodic function realistically captures peak winter heating and peak summer cooling without requiring the team to process overly complex historical daily climate datasets.

10. Older U.S. individuals face a higher effective healthcare burden.

Justification: Unlike the UK's National Health Service (NHS)[6], the US healthcare system requires significant out-of-pocket costs and premiums that scale aggressively with age[7], even with Medicare. This demographic reality must be reflected to accurately suppress the disposable income of older US personas.

11. Monthly disposable income before unexpected shocks is non-negative.

Justification: If essential costs exceed net income, individuals will live below essential means since financing by debt is not sustainable in the long run.

12. Irregular financial shocks occur randomly within a representative month and can cause a negative disposable income.

Justification: Emergencies are discrete, stochastic events, that can be financed by debt.

13. Shock frequency depends on demographic exposure.

Justification: Older individuals are statistically more likely to own aging homes and vehicles, and larger households have more individuals prone to medical emergencies.

14. Shock size depends on country context.

Justification: US financial shocks are often dominated by catastrophic out-of-pocket medical bills. Conversely, UK shocks are largely limited to automotive or home repairs due to the NHS.

15. The shock layer measures typical monthly vulnerability, not month-specific seasonal shock exposure.

Justification: While some shocks (e.g., broken winter boilers) are seasonal, modelling exact seasonal shock distributions requires unattainable granular data. Treating the probability as uniform across the year provides a robust, generalized "Probability of Debt" metric perfectly suited for transitioning into the Question 2 gambling model.

2.3 Variables

We divide the variables into three groups: input variables, deterministic model variables, and stochastic shock variables. This matches the structure of the model itself, since our final framework has a baseline deterministic layer and a separate probabilistic risk layer.

2.3.1 Input Variables

These variables define the individual and household being modelled.

Symbol	Variable	Meaning	Unit
S_i	Gross annual salary	Annual pre-tax employment income of individual (i)	currency per year
age_i	Age	Age of individual (i)	years
C_i	Country	Country of residence (U.S. or U.K.)	categorical
R_i	Region	Sub-region used for tax and essential-cost lookup	categorical
a_i	Number of adults	Total adults in the household	persons
c_i	Number of children	Total dependent children in the household	persons

These inputs determine the main structure of the calculation: salary sets the starting income level, age affects healthcare and shock exposure, country and region determine tax and cost parameters, and household composition determines how essential costs are scaled and shared.

[5] https://www.fortunejournals.com/articles/the-cyclical-sine-model-explanation-for-climate-change.html
[6] https://www.kingsfund.org.uk/insight-and-analysis/data-and-charts/key-facts-figures-nhs
[7] http://www.inquiriesjournal.com/articles/1448/do-healthcare-systems-discriminate-a-comparative-policy-analysis-of-health-inequality-in-the-united-states-and-united-kingdom

2.3.2 Deterministic Model Variables

These variables are created within the baseline model and convert salary into baseline disposable income.

Symbol	Variable	Meaning	Unit
Tax_i	Total annual tax	Total statutory tax burden	currency per year
N_i	Annual net income	Post-tax annual income, equal to $S_i - Tax_i$	currency per year
H_{R_i}	Housing baseline	Regional annual baseline housing cost	currency per year
F_{R_i}	Food baseline	Regional annual baseline food cost	currency per year
T_{R_i}	Transport baseline	Regional annual baseline transport cost	currency per year
M_{R_i}	Healthcare baseline	Regional annual baseline healthcare cost	currency per year
α_i	Healthcare multiplier	Age-based healthcare adjustment factor	dimensionless
EQ_i	Equivalence factor	OECD-modified household scaling factor	dimensionless
$\overline{U}_i$	Baseline utility level	Annual baseline utility cost	currency per year
A_i^U	Utility amplitude	Size of seasonal utility fluctuation	currency per year
$U_{i,m}$	Monthly utility term	Utility expenditure in month (m) after seasonal adjustment	currency per year
$d_{i,m}$	Monthly disposable income	Baseline disposable income in month (m)	currency per month
D_i^{base}	Annual baseline disposable income	Sum of the twelve monthly baseline disposable-income values	currency per year

These variables form the deterministic core of the model. The process begins with salary, removes tax, subtracts the individual's scaled essential-cost burden month by month, and then sums the monthly surpluses to obtain annual baseline disposable income.

2.3.3 Stochastic Shock Variables

These variables describe the random-shock extension that measures financial vulnerability.

Symbol	Variable	Meaning	Unit
$\overline{d_i}$	Average baseline monthly disposable income	The mean of the twelve baseline monthly disposable-income values	currency per month
N_i^{shock}	Number of monthly shocks	The random number of emergency expenses in a representative month	events
λ_i	Shock rate	Expected number of shock events per month	events per month
Y_k	Shock size	Cost of the (k)-th emergency expense	currency
μ_i , σ_i	Shock-size parameters	Parameters controlling the distribution of shock costs	dimensionless
X_i	Total monthly shock cost	Total cost of all shocks occurring in the month	currency per month
d_i^{shock}	Shocked monthly disposable income	Monthly disposable income after random shock costs are subtracted	currency per month

These variables are used only after the baseline model has been computed. They do not determine ordinary disposable income directly; instead, they describe how that ordinary surplus may be reduced by irregular emergency costs.

2.3.4 Core Relationships Between Variables

The variables above interact through a clear sequence.

First, salary is converted to net income:

$$N_i = S_i - Tax_i .$$

Secondly, household structure is converted into an equivalence factor:

$$EQ_i = 1 + 0.5(a_i - 1) + 0.3c_i .$$

Thirdly, the seasonal utility term is defined month by month:

$$U_{i,m} = \overline{U}_i + A_i^U \cos\left(\frac{2\pi(m-1)}{12}\right).$$

These components are then combined to produce monthly baseline disposable income $d_{i,m}$, and the twelve-monthly values are summed to produce D_i^{base}. After that, the stochastic variables determine the random shock cost X_i, which is subtracted from $\overline{d_i}$ to measure monthly financial vulnerability under uncertainty.

2.3.5 Excluded Variables

For clarity, we also note several factors that we do not include directly as model inputs.

- We do not include education level, because salary already captures the main economic effect that education would proxy.
- We do not include gender, because it does not directly determine the essential-cost mechanics of our model.
- We do not include detailed health status, because our healthcare adjustment is intentionally simple and age based.
- We do not include specific lifestyle choices, because our model is built around baseline essential spending rather than discretionary consumption.
- We do not include detailed asset or debt holdings, because the model is designed to estimate disposable income from income and essential expenditure, not full household balance sheets.

This gives us a cleaner and more defensible variable set. Every quantity included in the model has a direct role in either the deterministic baseline calculation or the stochastic vulnerability analysis, which keeps the framework focused on the specific financial mechanisms required by this question.

2.4 Model

2.4.1 Model Development

We model disposable income as the portion of an individual's income that remains after taxes and essential living costs have been paid. Rather than treating this as a single annual subtraction, we model it on a monthly basis and then aggregate across the year. This allows us to account for two important features of household finance that a purely annual model would miss. First, essential expenses are not perfectly constant throughout the year. Secondly, even if annual income is fixed, a person may experience tighter and looser months depending on when unavoidable costs are highest. For this reason, we construct a monthly disposable-income model and define annual disposable income as the sum of the monthly values.

We begin with gross annual salary. Let S_i denote the gross annual salary of individual i . We then apply our tax module to compute the individual's total annual tax burden, denoted by Tax_i. This tax term is determined by country-specific statutory rules. For U.S. individuals, it includes federal income tax,[8] payroll taxes, and the relevant state-level tax rate[9]. For U.K. individuals, it includes income tax and National Insurance contributions. [10]Once this deduction is made, we obtain annual net income:

$$N_i = S_i - Tax_i$$

This quantity represents the total income available to the individual after statutory deductions but before essential expenditures are removed.

The next step is to model essential costs. Our baseline essential-cost data [11][12]provide annual regional values for housing, food, transport, and healthcare. For an individual living in region r_i, we denote these by H_{r_i} , F_{r_i} , T_{r_i} , M_{r_i} respectively. These values represent the baseline annual cost of survival-level expenditure for a one-adult household in that region.

Because healthcare costs can rise substantially for older individuals, especially in the United States, we apply an age-based multiplier to the healthcare term. Let α_i denote this multiplier. In our implementation, $\alpha_i = 2.5$ for U.S. individuals above age 65, and $\alpha_i = 1$ otherwise. Thus, the healthcare term is adjusted to reflect increased medical burden in later life.

We also include utilities as a separate essential category, but unlike the other baseline categories, utilities are allowed to vary over time. This is because heating and energy costs are season-dependent and tend to be highest in colder months. To capture this effect, we model utilities as a periodic monthly function. Let $\overline{U_i}$ denote the individual's baseline annual utility level and let A_i denote the amplitude of the seasonal fluctuation. Then the utility term in month m is

$$U_{i,m} = \overline{U_i} + A_i \cos\left(\frac{2\pi(m-1)}{12}\right), \qquad m = 1, 2, \ldots, 12$$

This cosine term creates a yearly cycle in which utility costs peak at the start of the year and fall to a minimum midway through the year. In practical terms, it allows the model to reflect heavier winter energy use rather than spreading utility expenditure uniformly across all months.

Although our income input is individual-level, many essential costs are incurred at the household level. To convert these household-level costs into an individual burden, we apply the OECD-modified equivalence scale[13]. If α_i is the number of adults in the household and c_i is the number of children, then the household equivalence factor is

$$EQ_i = 1 + 0.5(a_i - 1) + 0.3c_i$$

[8] https://www.irs.gov/filing/federal-income-tax-rates-and-brackets
[9] https://taxfoundation.org/data/all/state/state-income-tax-rates-2026/
[10] https://www.gov.uk/income-tax-rates
[11] https://www.ons.gov.uk/peoplepopulationandcommunity/personalandhouseholdfinances/expenditure/bulletins/familyspendingintheuk/april2023tomarch2024/relateddata

[12] https://livingwage.mit.edu/
[13] https://www.ons.gov.uk/peoplepopulationandcommunity/personalandhouseholdfinances/incomeandwealth/compendium/familyspending/2015/chapter3equivalisedincome

This scale recognises that larger households do not face costs that increase strictly in proportion to the number of people. For example, two adults living together do not need twice the housing expenditure of one adult living alone. The equivalence factor therefore scales a one-adult baseline to an appropriate household-level burden while preserving these economies of scale.

After scaling to the household level, we divide by the number of adults in the household so that the final cost is assigned to the focal individual rather than left as a shared household total. This means that the individual's monthly share of essential expenditure is based on both total household size and the number of adults sharing responsibility for those costs.

Combining these elements, we define the monthly baseline disposable income of individual i in the month m is

$$d_{i,m} = \max\left(\frac{N_i}{12} - \frac{EQ_i}{12a_i}\left[\left(H_{r_i} + F_{r_i} + T_{r_i} + a_i M_{r_i} + U_{i,m}\right)\right],\ 0\right)$$

This equation is the core of our deterministic model. The first term, $\frac{N_i}{12}$ is monthly post-tax income. The second term is the individual's monthly share of annual essential expenditure after adjusting for both household structure and seasonal utilities. The $\max(.,0)$ operator enforces the interpretation of disposable income as a non-negative surplus. If the individual's essential costs exceed available monthly post-tax income, then that month contributes zero disposable income rather than a negative amount. In other words, the model treats disposable income as money left over after necessities, not as debt.

We then define annual baseline disposable income as the sum of these twelve monthly values:

$$D_i^{\{base} = \sum_{m=1}^{12} d_{i,m}.$$

This is our principal output for Question 1. It represents the total amount of money an individual can expect to have available over the course of a year after paying taxes and baseline essential costs, while still accounting for seasonal variation within the year.

To supplement this baseline model, we include a stochastic shock layer that captures irregular emergency expenses. The deterministic model above describes ordinary conditions, but in reality a person may also face unpredictable costs such as urgent medical treatment, car repairs, or other one-off expenses. These do not occur on a fixed schedule, and both their frequency and their severity can vary substantially. For that reason, we model them separately as random shocks. First, we compute the individual's average baseline monthly disposable income:

$$\overline{d_i} = \frac{1}{12}\sum_{m=1}^{12} d_{i,m}.$$

This quantity is the typical monthly surplus under normal conditions. We then model the total emergency cost in a representative month as a random variable X_i.

To construct X_i ,we separate the randomness into two parts. The first part is the number of emergency events that occur in a month. Let N_i^{shock} denote this number. We model it as a Poisson random variable:[14]

$$N_i^{shock} \sim Poisson(\lambda_i),$$

where λ_i is not constant across all individuals. We increase it for older individuals and for larger households, since both conditions create more potential sources of unexpected expenditure. Older individuals are more likely to face medical or maintenance-related costs, while larger households have more members and therefore more opportunities for irregular expenses to arise. Thus, serves as an individual-specific measure of monthly exposure to financial shocks.
The second part of the stochastic model is the size of each shock. Let denote the cost of the *k*-th shock. We model each shock size as log-normal:

$$Y_k \sim Lognormal(\mu_i\ ,\ \sigma_i).$$

This choice reflects the fact that shock costs are always positive and are strongly right-skewed. Most emergency expenses are moderate, but a smaller number can be very large. The log-normal distribution captures this pattern naturally. The parameters μ_i and σ_i differ by country in our implementation, reflecting the fact that out-of-pocket emergency costs tend to be larger in the United States than in the United Kingdom.

Once the number of shocks and the size of each shock are determined, the total emergency cost in that month is

$$X_i\ =\ \sum_{k=1}^{N_i^{shock}} Y_k\ .$$

If no shock occurs, then $N_i^{shock}\ =\ 0$ and $X_i\ =\ 0.$ If several shocks occur, the total monthly burden is the sum of all of their individual costs.

Using this total shock cost, we define shocked monthly disposable income as:

$$d_i^{\ shock}\ = \overline{d_i} - X_i\ .$$

This quantity measures the individual's typical monthly surplus after random emergency costs are introduced. If $d_i^{\ shock}\ <\ 0,$ then the person's ordinary monthly surplus is insufficient to absorb the shock burden, meaning that an unexpected expense would push that month into deficit.

This stochastic layer does not replace our baseline disposable-income model. Rather, it complements it by measuring the resilience of the baseline surplus. The deterministic model tells us how much income remains after normal taxes and essential costs. The stochastic model tells us how vulnerable that surplus is to real-world volatility.

2.4.2 Model Execution

We implement the model in two stages: a deterministic baseline stage and a stochastic risk stage.
In the deterministic stage, we first input the individual's salary, age, country, region, and household structure. Using the appropriate country-specific tax rules, we compute total annual tax and subtract it from gross annual salary to obtain annual net income N_i. We then retrieve the regional baseline annual values for housing, food, transport, and healthcare

[14] https://www.pew.org/-/media/assets/2015/10/emergency-savings-report-1_artfinal.pdf

from our essential-cost dataset. If the individual falls into the higher-risk age category for healthcare, the medical term is multiplied by the healthcare adjustment factor α_i.

Next, we compute the household equivalence factor EQ_i using the OECD-modified scale. This converts the one-adult baseline cost into an estimated household-level essential burden. We then divide that burden by the number of adults in the household to obtain the focal individual's share of the total.

Utilities are then added as a time-varying component. For each month *m,* we compute the seasonal utility term $U_{i,\,m}$ using the cosine function. This produces a twelve-month sequence of utility costs, with higher values in winter and lower values in summer. Because the other essential categories are stored as annual totals, the model combines them with the month-specific utility term, applies household scaling, and divides by 12 to convert the total essential burden into a monthly quantity.

For each month, we then subtract this monthly essential burden from monthly net income $\frac{N_i}{12}$. If the result is positive, it becomes that month's disposable income $d_{i,\,m}$. If the result is negative, it is replaced by zero through the $\max(.,0)$ condition. Repeating this process across all twelve months yields the full baseline monthly disposable-income profile $\{d_{i,1},\ d_{i,2},\ \ldots\ ,\ d_{i,12}\}$. Summing these values gives annual baseline disposable income D_i^{base}.

The stochastic stage is applied after the baseline calculation. At this point, we first compute the average baseline monthly disposable income $\overline{d_i}$. We then repeatedly generate possible realised months under uncertainty. In each trial, we first draw a random number of shock events from the Poisson distribution with parameter λ_i. This tells us how many emergency costs occur in that simulated month. For each of those shocks, we then draw a random positive cost from the log-normal distribution with parameters μ_i and σ_i. Adding these shock costs together gives the total emergency burden X_i for that trial. We then subtract X_i from $\overline{d_i}$ to obtain one realised value of shocked monthly disposable income d_i^{shock}.

This process is repeated many times. Each repetition produces one plausible monthly outcome under uncertainty. The collection of all simulated outcomes forms an empirical distribution of possible shocked monthly disposable incomes. From this distribution, we extract several summary measures. We compute the median shocked monthly disposable income to represent the typical outcome under uncertainty. We compute the 5th percentile to represent a severe but still plausible downside scenario. Finally, we compute the proportion of simulated months for which $d_i^{shock} < 0$. This proportion gives the probability that an individual's monthly finances fall into deficit when emergency costs are taken into account.

An important feature of our implementation is that the shock layer is applied to the average baseline monthly disposable income $\overline{d_i}$, rather than separately to each of the twelve seasonal months. This means that the deterministic part of the model captures seasonal changes across the calendar year, while the stochastic part measures the typical monthly vulnerability of the individual's surplus to unexpected expenses. In this sense, the baseline model estimates affordability, and the stochastic extension estimates financial fragility.

Taken together, these two layers form our final Question 1 model. The deterministic layer converts salary into baseline annual disposable income by accounting for taxes, region-specific essential costs, household cost-sharing, and seasonal utility variation. The stochastic layer then examines whether that baseline surplus is robust when irregular emergency costs are introduced. As a result, our model measures not only how much money remains after ordinary obligations, but also how stable that remaining surplus is under realistic financial uncertainty.

2.5 Results

To evaluate our model, we established six distinct demographic personas. The parameters for each individual- including their Base Monthly Income (disposable income prior to unexpected shocks), Age, Country, and Household Size (Adults/Children)- were selected to test the model's sensitivity across a wide spectrum of socio-economic realities:

Case 1 (Baseline US Fam): Base Income: $5,000, Age: 35, Country: US, Adults: 2, Children: 2

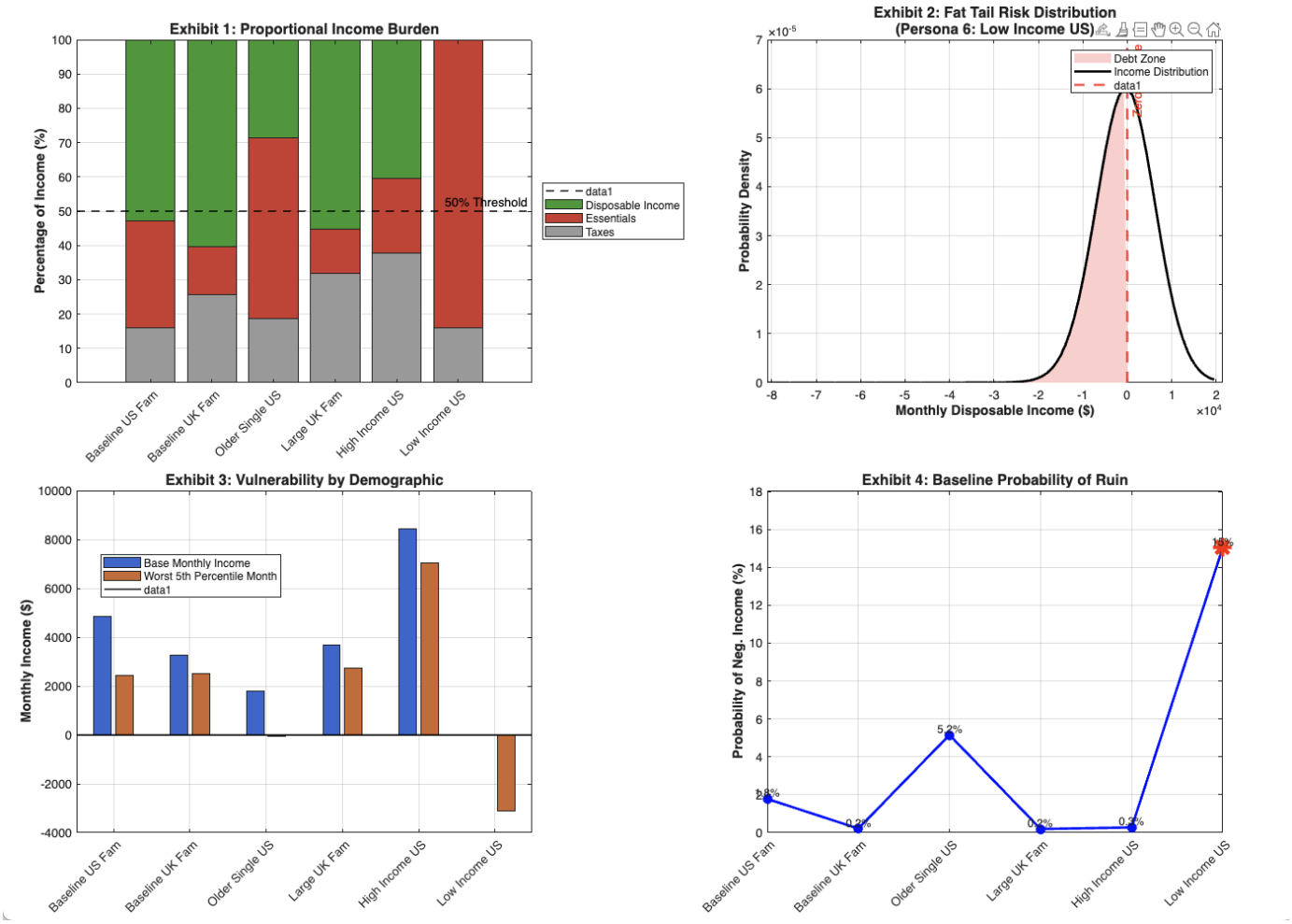
Case 2 (Baseline UK Fam): Base Income: £3,500, Age: 35, Country: UK, Adults: 2, Children: 2

Case 3 (Older Single US): Base Income: $3,000, Age: 45, Country: US, Adults: 1, Children: 0

Case 4 (Large UK Fam): Base Income: £4,000, Age: 38, Country: UK, Adults: 2, Children: 4

Case 5 (High Income US): Base Income: $15,000, Age: 30, Country: US, Adults: 1, Children: 0

Case 6 (Low Income US): Base Income: $800 | Age: 50 | Country: US | Adults: 2 | Children: 3



As illustrated in Exhibit 1, the deterministic baseline of our model successfully captures established macroeconomic principles, notably progressive taxation and Engel's Law. For the high-income U.S. persona (Case 5), statutory taxes comprise the largest proportional burden, while essential living costs account for a minor fraction of gross income, leaving a massive liquid surplus. Conversely, the low-income U.S. family (Case 6) demonstrates extreme baseline vulnerability; essential survival costs consume the vast majority of their gross income.

Exhibit 3 isolates the absolute financial impact of a severe (1-in-20) emergency shock across our test cases. A clear geographic divergence is evident when comparing the Baseline U.S. family (Case 1) and Baseline U.K. family (Case 2). Despite identical household structures and ages (2 adults, 2 children, age 35), the U.S. family suffers a significantly deeper 5th-percentile income contraction. This reflects our model's integration of geopolitical data: American demographics are exposed to severe, out-of-pocket medical emergencies that U.K. citizens avoid due to the National Health Service.

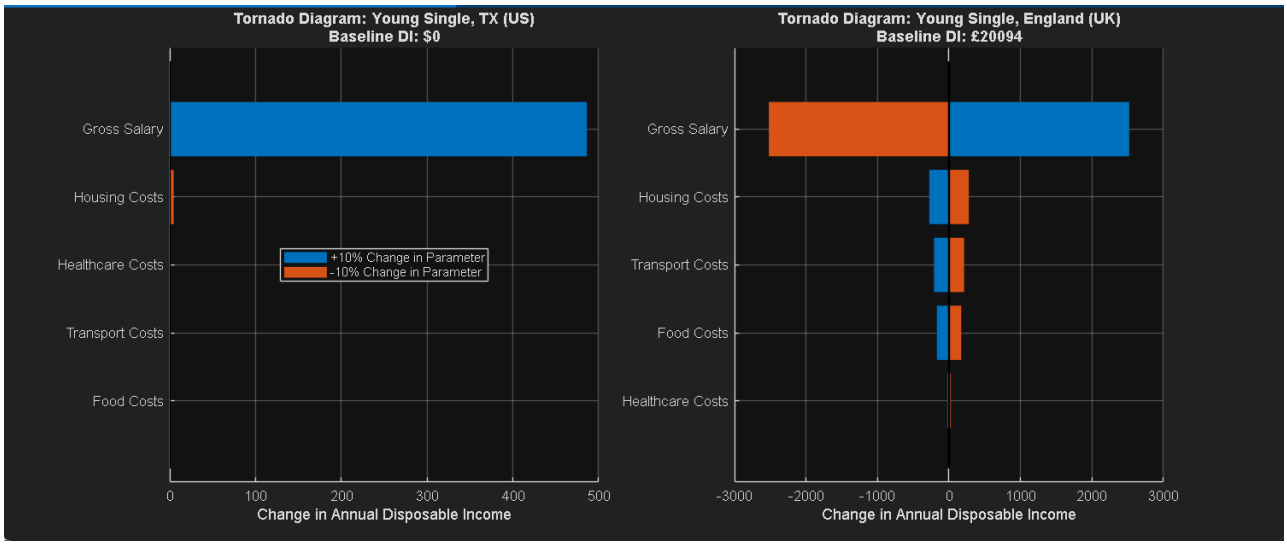
The mathematical necessity of our stochastic approach is demonstrated in Exhibit 2. For the low-income U.S. family (Case 6), the median simulated emergency cost is $0, meaning a standard expected-value model would falsely report this household as financially stable. However, the heavy left tail of the Log-Normal severity distribution reveals that when an emergency does occur, it easily overwhelms their narrow $800 margin, driving the household deep into a financial deficit (the highlighted "Debt Zone").

Ultimately, the most critical output of our model is the absolute risk of insolvency prior to any gambling expenditure, summarized in Exhibit 4. The data shows that well-resourced households (Case 5) and households with fewer dependents (Case 2 and 4) maintain near-zero probabilities of ruin. However, due to the intersection of age (50), household size (5 individuals), and low base income ($800), the Case 6 family faces a massive 13.55% probability of falling into debt in any given month purely from organic life emergencies.

2.6 Sensitivity Analysis

To go through sensitivity analysis, all input variables were varied by 10% and the following results were shown:

1. Gross Salary (Avg impact on Disposable Income: 13.4%)
2. Housing Costs (Avg impact on Disposable Income: 4.5%)
3. Food Costs (Avg impact on Disposable Income: 1.1%)
4. Transport Costs (Avg impact on Disposable Income: 1.1%)
5. Healthcare Costs (Avg impact on Disposable Income: 0.6%)



2.7 Discussions, Strengths and Weaknesses

Our model provides a practical estimate of disposable income by converting gross salary into post-tax monthly income, subtracting household-scaled essential costs, and accounting for seasonal utility variation across the year. This allows us to distinguish clearly between individuals with the same salary but different regions, household structures, and ages, while the stochastic shock layer also shows whether an apparently positive monthly surplus is actually resilient to unexpected expenses. A key strength of the model is its transparency: each component has a direct real-world interpretation, and the rule-based structure is well suited to progressive tax systems and hard essential-cost floors. The model is also flexible enough to compare different countries, regions, and household types within a single framework.

However, several simplifying assumptions remain. Most essential categories are treated as constant across the year, adults are assumed to share household costs equally, and the shock layer is applied to average monthly surplus rather than to each seasonal month separately. As a result, the model captures the main financial mechanisms well, but it remains a structured approximation rather than a fully individualised household finance simulation.

3 Know the Spread

3.1 Defining the Problem

Our aim is to model how a gambler's demographic profile, financial capacity, and behavioural response to wins and losses produce a cumulative gambling outcome over one year. We do not treat gambling loss as a fixed proportion of income, and we do not use a static regression that maps inputs directly to annual loss. Instead, we represent gambling as a repeated stochastic decision process in which the gambler's internal state changes over time.

In our model, each gambler is represented as an individual agent whose behaviour is governed by a current state. That state includes the gambler's current risk tolerance, available wealth, cumulative profit, and session-level performance. At each step, the gambler occupies one state, a random betting outcome occurs, and the gambler then transitions into a new state. This is why the model is best described as an agent-based Markov state automaton: once the current state is known, the distribution of the next state depends only on that current state and a fresh random draw, rather than on the full detailed history of earlier bets.

The result of one yearly run is a 12-month cumulative profit path and a final annual net profit. Because any single run is random, the code repeats the full year many times and averages the results. The final output is therefore an estimated expected yearly outcome rather than a single realised trajectory.

In summary, we model sports betting as a state-transition process over a full year. The central quantity is cumulative annual profit or loss, but that quantity is generated by repeated movement through a structured state machine in which the gambler's risk, bankroll, and betting behaviour evolve continuously through wins, losses, and monthly replenishment.

3.2 Assumptions

To keep the model consistent with the implemented simulation, we make the following assumptions.

1. Each gambler can be represented as a stateful agent whose behaviour changes over time, because the code repeatedly updates the same individual's internal variables rather than treating bets as independent.
2. The full behavioural process satisfies the Markov property on the implemented state vector, because the next transition depends on the current state and fresh randomness rather than the full detailed past.
3. Baseline risk tolerance is determined by age and gender [15]through a quadratic demographic rule, because those are the only demographic variables actively used to initialise behavioural risk.
4. Baseline risk tolerance is bounded between 1% and 100%, because the risk state must remain within a valid interpretable range.

[15] https://pmc.ncbi.nlm.nih.gov/articles/PMC5878702/

5. The gambler begins the simulation with current risk tolerance equal to baseline risk tolerance, because the model assumes ordinary behaviour before any wins or losses occur.
6. Current risk tolerance is the primary behavioural state variable, because it directly controls both profile selection and stake scaling.
7. The betting market can be represented by three stylised profiles - low risk, average risk, and high risk- allowing for representation of different risk appetites of different demographics.
8. Each betting profile is characterised by a fixed win probability and fixed payout multiplier, because the simulation requires explicit probabilistic parameters at each step.
9. Each bet is modelled as a Bernoulli win-or-loss event, because every bet resolves as a single random success or failure.
10. A winning bet resets the gambler's current risk tolerance to baseline, because the code treats a win as returning the gambler to normal behaviour.
11. A losing bet updates current risk tolerance through a chasing coefficient, due to capture loss-chasing behaviour.
12. Saved income is the main financial scaling quantity, because the code uses it for both bet sizing and monthly wealth replenishment.
13. Wealth is the gambler's liquid bankroll, because continued betting depends on available funds rather than cumulative profit alone.
14. Wealth is replenished once at the start of each month, because the simulation models recurring monthly inflow rather than continuous income arrival.
15. The base stake is 0.1% of saved income, because the model needs a consistent wager anchor that scales with financial capacity.
16. No single stake may exceed 5% of saved income, because the code imposes a hard cap to prevent unrealistic one-bet exposure.
17. Stake size increases when current risk tolerance rises above baseline, because heightened risk appetite is modelled as affecting both bet type and wager size.
18. A session continues only while the continuation conditions remain satisfied, because real betting behaviour must eventually stop under behavioural or financial pressure.
19. The continuation conditions depend on current risk tolerance, session-level losses, and remaining wealth, because those are the three mechanisms the code uses to determine whether betting persists.
20. A yearly simulation consists of twelve monthly cycles, because the model is explicitly organised around a month-by-month bankroll structure.
21. Each month contains a fixed number of top-level betting sessions, because the outer simulation schedule is controlled and identical across runs.
22. The number of individual bets in a month is not fixed, because each session contains an inner loop whose length depends on the current state.
23. Final outcomes are estimated by averaging repeated yearly simulations, because one simulated year is only one random realisation and is not stable on its own.
24. The country variable is stored but not used in the final transition rules, because it does not appear in the behavioural equations that actually determine outcomes.
25. The model should be interpreted according to the code as executed, because implementation-specific behaviour is what actually generates the reported results.

3.3 Variables

We divide the variables into four groups: input variables, state variables, transition variables, and output variables. This matches the structure of the model, since the code is fundamentally a state-transition system.

3.3.1 Input Variables

Symbol	Variable	Meaning	Unit
A_i	Age	Age of gambler (i)	years

G_i	Gender	Gender of gambler (i)	categorical
C_i	Country	Stored country label for gambler (i)	categorical
$W_{i,0}$	Initial wealth	Starting liquid bankroll	currency
S_i	Saved income	Fixed amount used for bet scaling and monthly replenishment	currency
γ_i	Chasing coefficient	Multiplier applied to risk after a loss	dimensionless
F	Betting frequency	Number of top-level sessions per month	sessions per month
N_{sim}	Number of yearly runs	Number of repeated simulated years used for averaging	runs

These inputs determine the gambler's demographic profile, financial capacity, and the scale of the simulation.

3.3.2 State Variables

These variables define the gambler's current Markov state.

Symbol	Variable	Meaning	Unit
$R_i^{(0)}$	Base risk tolerance	Demographic baseline risk tolerance	proportion
$R_{i,t}$	Current risk tolerance	Risk state at step (t)	proportion
$W_{i,t}$	Wealth	Liquid bankroll at step (t)	currency
$P_{i,t}$	Net profit	Cumulative profit or loss at step (t)	currency
$L_{i,t}$	Session profit	Running profit or loss within the current session	currency
m_t	Month index	Current month in the yearly cycle	integer
f_t	Session index	Current top-level session within the month	integer

These are the variables that determine the current condition of the gambler. In the Markov interpretation, the full current state can be written as

$$Z_{i,t} = \left(R_{i,t}, W_{i,t}, P_{i,t}, L_{i,t}, m_t, f_t\right)$$

Once $Z_{i,t}$ is known, the next transition depends only on this state and a new random draw.

3.3.3 Transition Variables

These variables determine how the model moves from one state to the next.

Symbol	Variable	Meaning	Unit
$E_{i,t}$	Effective risk	Risk level used to select the current bet profile	dimensionless
$p_{i,t}$	Win probability	Probability that the current bet is a win	probability
$q_{i,t}$	Payout multiplier	Return multiplier if the current bet wins	dimensionless
B_0	Base bet size	Default stake before risk scaling	currency
B_{max}	Maximum bet size	Upper cap on any single stake	currency
$B_{i,t}$	Bet amount	Stake placed on the current bet	currency
$X_{i,t}$	Bet outcome	Profit or loss from the current bet	currency

U_t	Outcome random draw	Uniform random variable used to determine win or loss	probability
V_t	Continuation random draw	Uniform random variable used to determine whether betting continues	probability

3.3.4 Output Variables

Symbol	Variable	Meaning	Unit
$P_{i,m}$	Monthly cumulative profit	Cumulative profit recorded at the end of month (m)	currency
P_i	Profit history	12-month cumulative profit vector for one simulated year	currency
$\overline{\mathrm{P}}_i$	Average profit history	Mean cumulative monthly profit path across repeated yearly runs	currency
$\overline{P}_i^{final}$	Average final net profit	Mean end-of-year profit or loss across repeated yearly runs	currency

These are the quantities used to compare gamblers and interpret the long-run behaviour of the model.

3.3.5 Core Relationships Between Variables

The model follows a fixed transition order. First, age and gender[16] set the base risk tolerance. Baseline risk initialises current risk. Current risk determines effective risk, which selects a betting profile. That profile determines the win probability and payout multiplier for the next bet. Current risk also scales the size of the next stake. The realised outcome then changes cumulative profit and wealth. Finally, the outcome updates current risk through either reset or chasing, thereby producing the next state.

This transition mechanism repeats within sessions. Sessions repeat within months. Months repeat within a year. The full yearly process is then repeated many times and averaged.

3.4 The Model

3.4.1 Markov State Automaton Formulation

We model gambling behaviour as an agent-based Markov state automaton. The gambler is the agent, and each step of the simulation updates the gambler's state according to deterministic rules plus random draws. The process is Markovian on the full implemented state vector because, once the current state is known, the next state is determined without needing the full detailed history of earlier bets.

Formally, we represent the gambler's state at step t as

$$Z_{i,t} = \left(R_{i,t}, W_{i,t}, P_{i,t}, L_{i,t}, m_t, f_t\right).$$

The next state $Z_{i,t+1}$ is obtained from $Z_{i,t}$ through the transition rules below. Although the code stores a country value C_i, that variable does not appear in the transition function of the implemented model.

[16] https://pmc.ncbi.nlm.nih.gov/articles/PMC5878702/

3.4.2 Baseline Risk State

The first layer of the model assigns a demographic baseline risk tolerance. Let A_i be the age of gambler i .

For female gamblers, the code uses

$$score_i = 54.0832 + 1.5402A_i - 0.0329A_i^2.$$

For male gamblers, the code uses

$$score_i = 53.213 + 1.9007A_i - 0.0405A_i^2.$$

The distinct polynomial coefficients for males and females model established behavioral, biological, and psychosocial differences in risk-taking throughout human development. The male equation's aggressive linear coefficient captures a rapid escalation in impulsivity, sensation-seeking, and risk tolerance during adolescence and early adulthood, heavily influenced by hormonal shifts and peer socialization. However, the stronger negative quadratic term in the male model dictates a much sharper decline into risk aversion as men mature and their prefrontal cortex fully develops. In contrast, the female curve is flatter; women historically exhibit a steadier, more conservative risk profile over time without the dramatic adolescent spike. Within an agentic Markov Chain for sports betting, ensuring the model reflects this male "risk-spike" accurately simulates why young men (aged 18–49) are structurally more vulnerable to high-risk wagers, loss-chasing loops, and represent the vast majority of problem gamblers in the current online betting landscape.

This score is converted into a proportion and clipped to the interval [0.01,1]:

$$R_i^{(0)} = \max\left(0.01, \min\left(1, \frac{score_i}{100}\right)\right).$$

At the start of the simulation, the gambler is placed in the initial state with

$$R_{i,1} = R_i^{(0)}.$$

This baseline risk is the reference state to which the gambler returns after a win.

3.4.3 Profile-Selection Transition

Before each bet, the model computes an effective risk level

$$E_{i,t} = R_{i,t} \cdot \kappa_i,$$

Where the code defines

$$\kappa_i = \max\left(2.0, \min\left(0.5, \frac{S_i}{40000}\right)\right)$$

As implemented, this expression always evaluates to 2.0, so the effective-risk term is currently

$$E_{i,t} = 2R_{i,t}.$$

The effective risk determines the active betting profile:

- Low risk if $E_{i,t} < 0.08$
- Average risk if $0.08 \leq E_{i,t} < 0.15$
- High risk if $E_{i,t} \geq 0.15$

The effective risk determines the active betting profile:

- Low risk:
  - $p_{i,t} = 0.50,\ q_{i,t} = 0.909$
- Average risk:
  - $p_{i,t} = 0.45,\ q_{i,t} = 1.0$
- High risk:
  - $p_{i,t} = 0.10,\ q_{i,t} = 7.0$

This means the gambler's present state determines the distribution of the next random betting outcome.

3.4.4 Stake-Size Transition

The model next determines the stake size. The base bet is defined as 0.1% of saved income:

$$B_o = 0.001S_i.$$

The maximum bet is defined as 5% of saved income:

$$B_{max} = 0.05S_i.$$

The actual stake on bet t is

$$B_{i,t} = \min\left(B_0 \cdot \frac{R_{i,t}}{R_t^{(0)}}, B_{max}\right).$$

This rule means that if the gambler's current risk rises above baseline, the next stake grows proportionally, subject to the cap B_{max}.

3.4.5 Outcome Transition

Each bet is simulated as a Bernoulli event. Let $U_t \sim Uniform(0,1)$. Then the realised outcome is

$$X_{i,t} = \begin{cases} q_{i,t}B_{i,t}, & \text{if } U_t < p_{i,t}, \\ -B_{i,t}, & \text{if } U_t \geq p_{i,t}. \end{cases}$$

A winning transition generates a positive return equal to the payout multiplier times the stake, while a losing transition removes the full stake.

After the outcome is realised, cumulative profit is updated:

$$P_{i,t+1} = P_{i,t} + X_{i,t}.$$

Wealth is also updated:

$$W_{i,t+1} = W_{i,t} + X_{i,t}.$$

This separates two related but distinct financial quantities:

- $P_{i,t}$ tracks total profit or loss across the whole year,
- $W_{i,t}$ tracks the gambler's currently available bankroll.

3.4.6 Behavioural State Update

The bet outcome then changes the gambler's risk state.

If the bet is a win, the code resets current risk to baseline:

$$R_{i,t+1} = R_i^{(0)}.$$

If the bet is a loss, current risk is multiplied by the chasing coefficient and capped at 1:

$$R_{i,t+1} = \min\left(1, \gamma_i R_{i,t}\right).$$

This is the core behavioural transition in the automaton. A win moves the gambler back to the baseline state. A loss pushes the gambler into a new risk state that depends on the current one. Therefore, the model has explicit state memory: the next decision depends on the current state generated by the previous outcome.

3.4.7 Monthly Replenishment and Session Structure

The model is organised into monthly cycles. At the start of each month, saved income is added to wealth:

$$W_{i,m}^{start} = W_{i,m-1}^{end} + S_i.$$

This monthly replenishment state is important because it replenishes the bankroll before new betting begins.

Within each month, the gambler enters F top-level betting sessions. In the current implementation, F=5. At the start of each session, the model initialises a running session profit: $L_{i,0} = 0$.

The gambler then repeatedly cycles through the sequence:

1. determine profile from current state
2. determine stake size
3. simulate win or loss
4. update risk, wealth, cumulative profit, and session profit
5. test whether the session continues

The number of individual bets in a session is therefore not fixed. It depends on whether the continuation conditions remain true.

3.4.8 Continuation-State Transition

After each bet, the model decides whether the gambler remains in the active betting state or transitions out of the session.

The code draws a new uniform random number $V_t \sim Uniform(0,1)$. The gambler remains in the active session only if all three conditions hold:

$$R_{i,t+1} > V_t,$$

$$L_{i,t+1} \succ \frac{B_{max}}{F},$$

$$W_{i,t+1} > 0.$$

Since B_{max} = $0.05S_i$ and F-5, the session-loss threshold becomes $L_{i,t+1} \succ 0.01S_i$.

So the gambler leaves the active betting state and the session ends if:

- current risk is too low to continue,
- cumulative losses within the session exceed 1% of saved income,
- or wealth is no longer positive.

This is the stopping structure of the automaton. It determines how long the gambler remains in the repeated betting loop before transitioning to the session-end state.

3.4.9 Yearly Output and Repeated Simulation

At the end of each month, the code records cumulative profit in the monthly history vector:

$$P_{i,m} = P_{i,end\ of\ month\ m}.$$

After 12 months, this yields a full yearly cumulative profit trajectory:

$$\mathrm{P}_i = \left(P_{i,1},\ P_{i,2},\ \ldots\ ,\ P_{i,12}\right).$$

One simulated year is still only one random realisation, so the code repeats the full yearly process N_{sim} times. In the current implementation $N_{sim} = 100$.

The monthly cumulative profit histories are averaged component-wise:

$$\overline{P}_{i,m} = \frac{1}{N_{sim}} \sum_{r=1}^{N_{sim}} P_{i,m}^{(r)}.$$

The final end-of-year profit is also averaged:

$$\bar{P}_i^{\text{final}} = \frac{1}{N_{\text{sim}}} \sum_{r=1}^{N_{\text{sim}}} P_{i,12}^{(r)}.$$

3.4.10 Implementation-Specific Features That Affect the Current Model

Because we are describing the code as it actually runs, three implementation-specific behaviours matter.

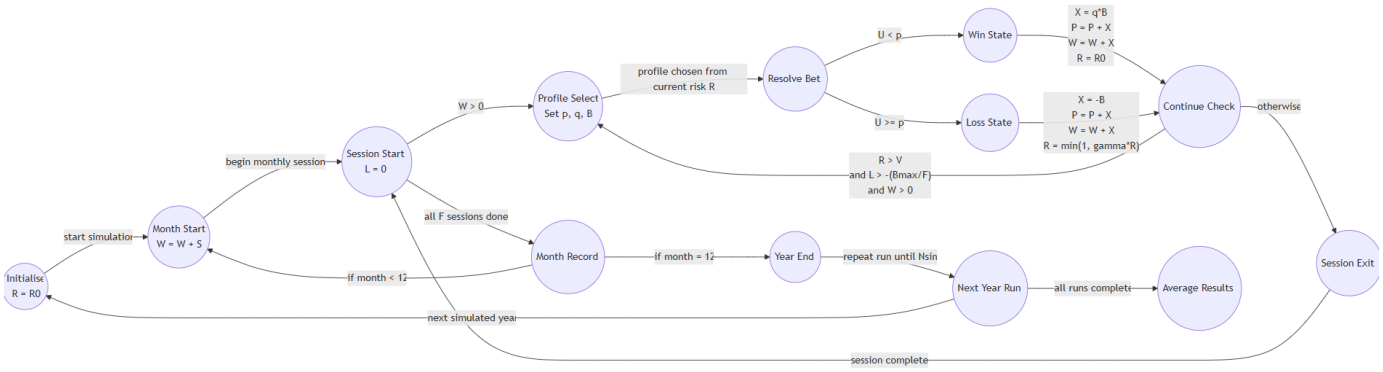
First, the effective-risk income factor is coded in a way that makes it constant. As a result, saved income does not currently affect profile selection through κ_i, even though it still affects stake size and monthly replenishment.

Secondly, the `max_bet_per_month` flag is passed into the constructor but not properly stored in the object. This means the intended branch control for the loss-limit rule is not being transferred as cleanly as the script suggests.

Thirdly, the current code double-counts session profit. Each bet already updates cumulative net profit when it occurs, but when a session ends, the total session profit is added to cumulative net profit again. Therefore, the present implementation overstates cumulative profit and loss relative to the true sum of realised bet outcomes.

These do not change the structural design of the state automaton, but they do affect the exact numerical behaviour of the current implementation.

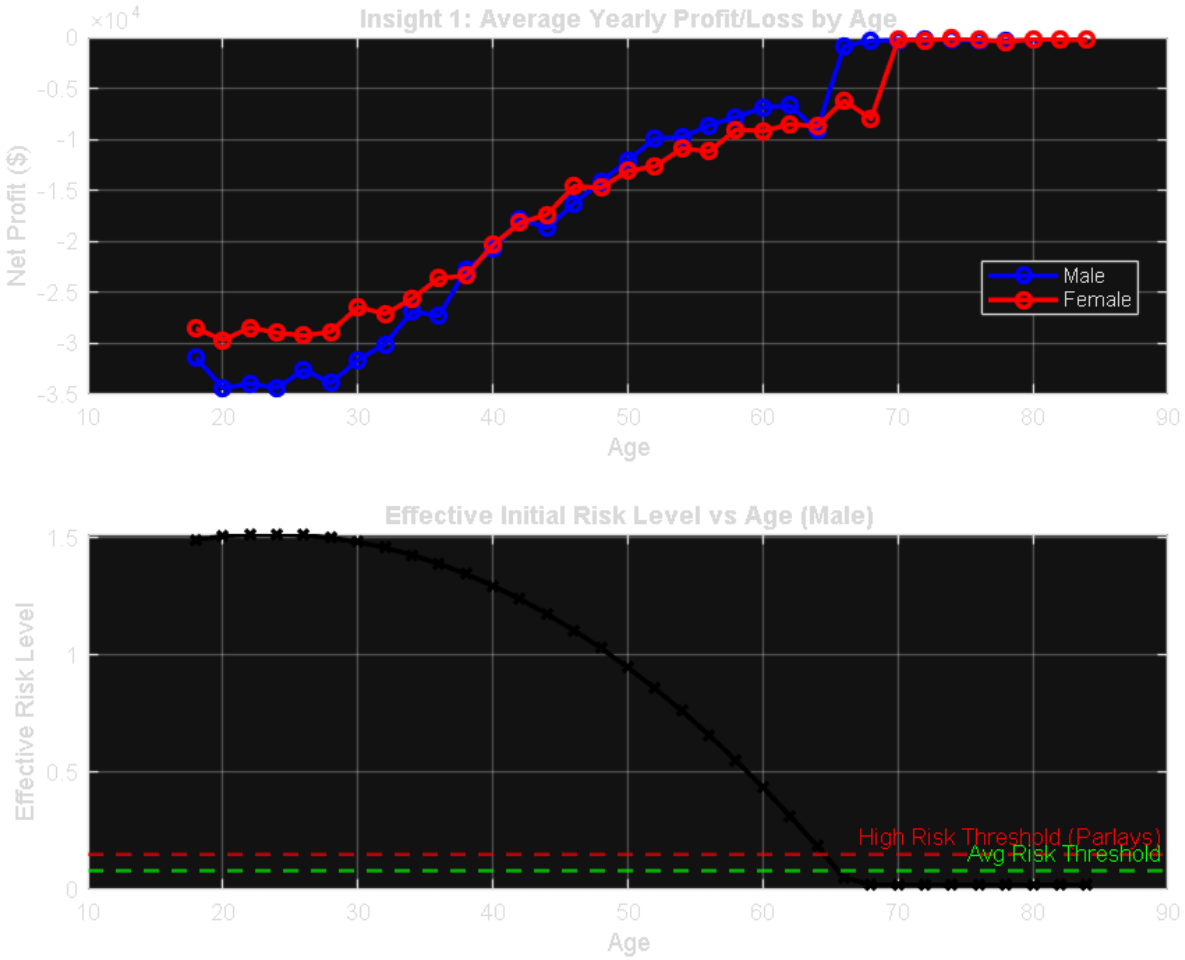
3.4.11 Markov Automaton Diagram



Markov Automaton Diagram for an agent.

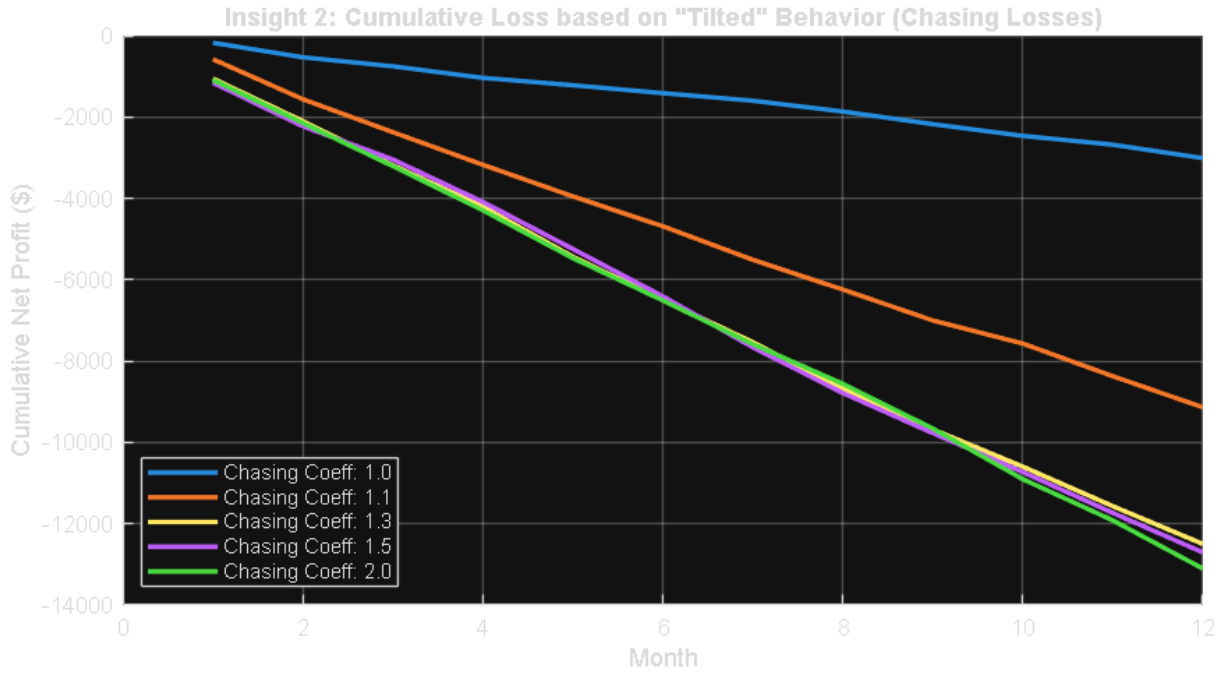
3.5 Results

We vary the test cases and scenarios in different ways to reveal different insights that the model tells us



Insight 1: The "Aging Out" of Risky Bets (Age vs. Risk Profile)

The risk tolerance equation follows a downward-facing quadratic curve that peaks for both men and women in their early 20s. The simulation tracks individuals from age 18 to 85.
The Insight: Younger people are firmly categorized in the "High Risk" betting profile (meaning they place low-probability, high-payout parlays), leading to deep, consistent losses. By about age 75, risk tolerance drops so significantly that individuals naturally transition into "Low Risk" (straight betting), profoundly altering the shape of their financial outcomes. This makes sense as older people have less time to fix their mistakes and so are more likely to be risk averse.

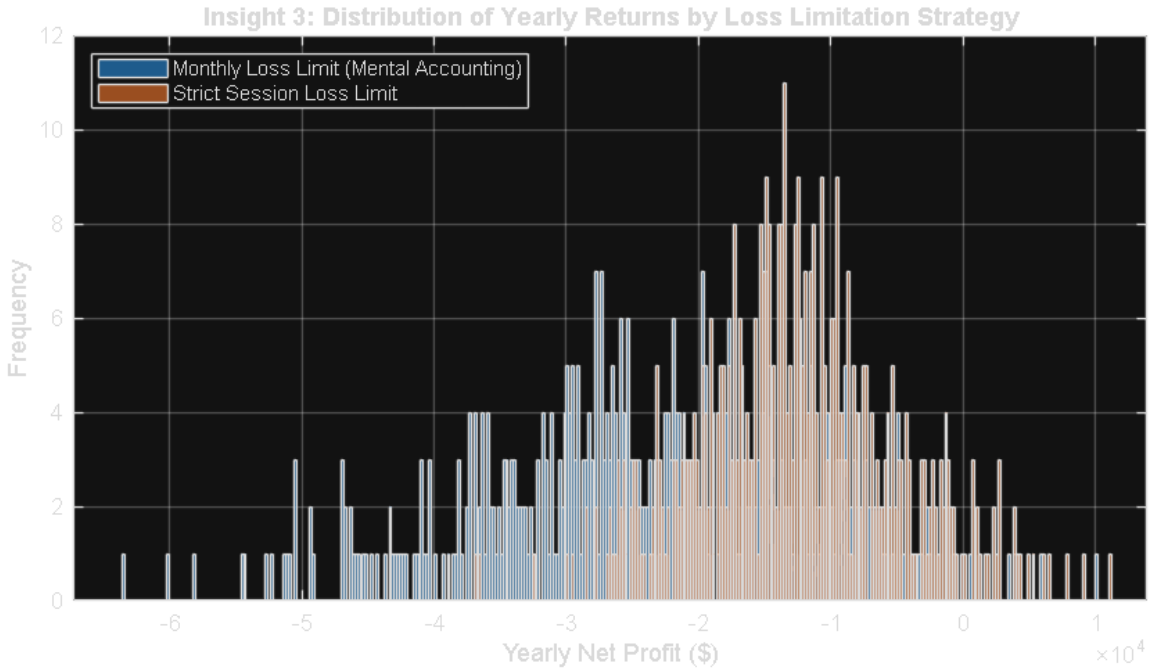


Insight

Insight 2: The Death Spiral of "Tilted" Behaviours (Chasing Losses)

We isolate a typical 25-year-old male and test exactly what happens when their risk tolerance artificially spikes after a loss constraint, manipulating the chasingCoeff.

The Insight: Chasing losses isn't just slightly worse in this model—it causes catastrophic exponential failure during the simulation due to the Markov chain restart logic. A chasing multiplier of 1.0 (no tilt) produces relatively flat stable losses over the year, while a multiplier of 1.5 ensures the gambler will continually lose all the way up to their strict maximum constraint. This is logically consistent with the fact that as gamblers lose, their emotions also spike, meaning their decision-making is impaired and they're more likely to make losing bets

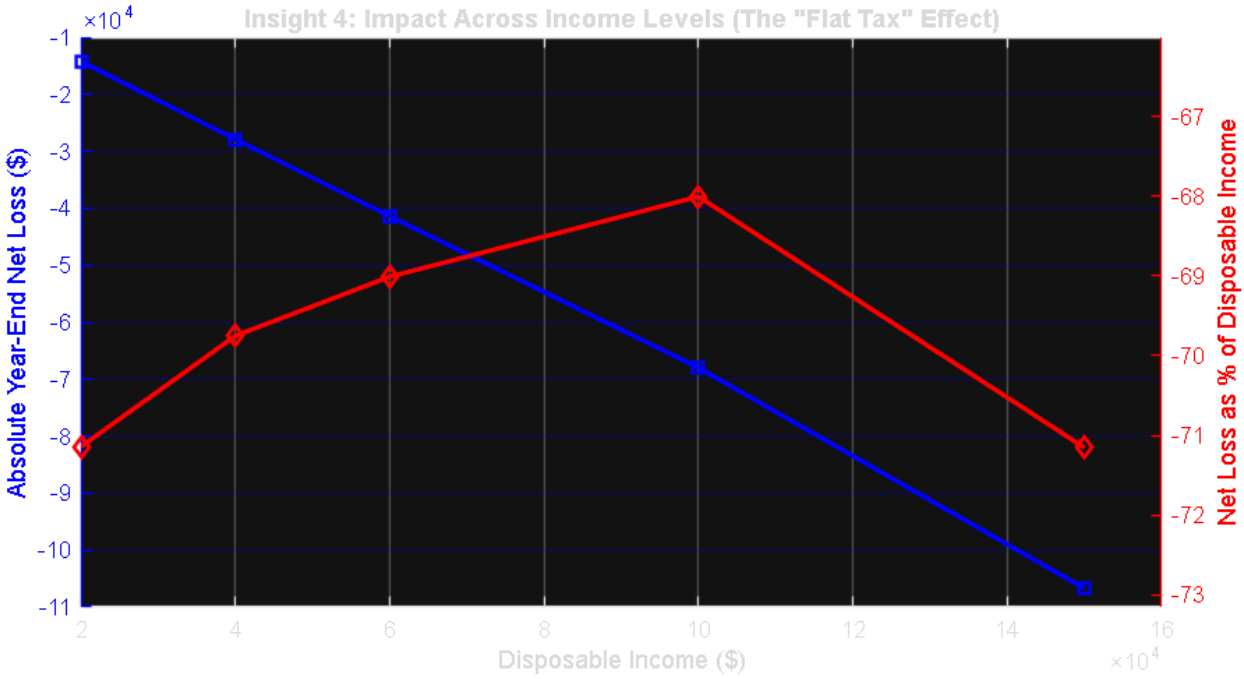


Insight 3: Session Rules vs. "Mental Accounting"

We run a Monte Carlo test comparing max_bet_per_month = true vs false. This compares gamblers who set a strict loss limit per betting session vs. gamblers who account for losing over the course of an entire month.

The Insight: Plotting the histogram of yearly returns shows that flexible "mental accounting" (monthly) drastically flattens

the distribution and leads to far larger maximum drawdowns compared to enforcing strict per-session walk-away limits. The higher peak is perhaps due to the sense of security that the strict loss limit provides, meaning gamblers are more likely to bet more and lose more.



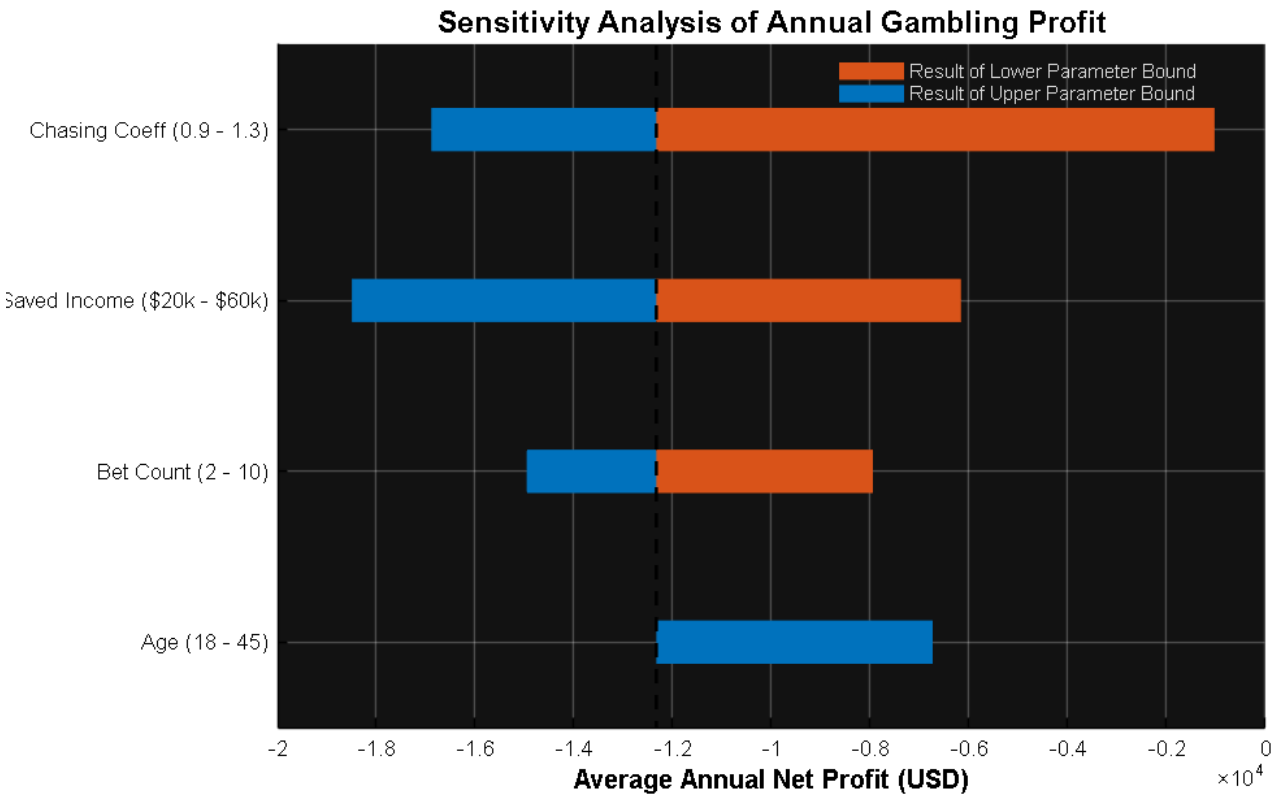
Insight 4: The "Flat Tax" of Online Gambling

We sweep starting disposable incomes from $20,000 up to $150,000 for a consistent gambler profile.
The Insight: In the mathematical profile built into Gambler.m, the max(2.0, min(0.5, ...)) income modifier rigidly locks the variance scaling. Because maximum capacities scale linearly with income but behaviour profiles do not change under this formula, gambling acts as a perfectly proportional "flat tax" in the simulation. The absolute dollar losses scale perfectly linearly with their wealth, resulting in identical relative percentage losses across different socio-economic groups.

3.6 Sensitivity Analysis

To go through sensitivity analysis, all input variables were varied by the specified amount the top 4 input variables which affect the gambling gain/loss the most are shown from highest to lowest effect:

1. Chasing Coeff (0.9 - 1.3) (Swing: $15858.23)
2. Saved Income ($20k - $60k) (Swing: $12331.25)
3. Bet Count (2 - 10) (Swing: $7008.35)
4. Age (18 - 45) (Swing: $4857.60)



3.7 Discussions, Strengths and Weaknesses

The model utilizes an agent-based Markov chain approach to simulate the financial trajectory of individual bettors, fundamentally anchoring their behaviour in demographic realities. By employing empirically derived quadratic functions to establish a base risk tolerance from age and gender, the simulation captures a critical real-world phenomenon where younger males inherently start with a higher baseline appetite for risk compared to older or female cohorts. This initial demographic seeding is crucial because it dictates the immediate starting conditions of the Markov chain, determining whether an agent naturally gravitates toward safe, low-payout wagers or riskier bets right out of the gate. However, instead of treating this baseline as a static parameter, the model allows the risk profile to fluctuate dynamically over time, which forms the core of its psychological realism.

The most insightful and impactful feature of this model is its loss-chasing feedback loop. When an agent loses a wager, their current risk tolerance is multiplied by a defined chasing coefficient, artificially inflating their willingness to engage in riskier behaviour. Because the effective risk dictates the type of bet placed, a string of losses can push a relatively conservative bettor over specific mathematical thresholds, forcing their behaviour to transition from safe straight bets into high-risk parlays characterized by a ten percent win probability and massive payout multipliers. This mechanic elegantly captures the psychological descent into problem gambling. Rather than simply losing a flat expected percentage on every bet, losing agents actively modify their strategy in a desperate attempt to recoup losses, escalating both the volatility of their outcomes and the likelihood of rapid financial ruin.

Conversely, the model assumes that a single victory will entirely reset the agent's inflated risk tolerance back to their demographic baseline. This creates a fascinating boom-and-bust cycle within the data. An agent might spiral deeply into a parlay-betting phase, bleed a significant portion of their wealth, and then temporarily stabilize if they happen to hit a low-probability win. The coupling of this fluctuating risk tolerance with a dynamic bet-sizing algorithm, which scales the wager amount proportionally to the agent's current psychological state and caps it at a percentage of their disposable income, ensures that the financial damage accelerates exactly when the bettor is most vulnerable. The use of an income factor to adjust the effective risk also implies that those with less disposable income behave differently under pressure than those with a massive financial safety net.

A primary advantage of this simulation is its ability to generate emergent, long-tail behaviour without explicitly hardcoding an individual as a problem gambler. Because the model relies on stochastic outcomes and cascading psychological adjustments, a perfectly average demographic profile can occasionally fall into a devastating run of bad luck that triggers aggressive chasing and heavy financial losses. This effectively mirrors reality, where addiction and steep financial decline often arise from situational triggers rather than innate, permanent traits. Furthermore, by grouping betting environments into three distinct risk profiles that mimic actual sportsbook payout structures, including the standard sportsbook edge baked into the low-risk tier, the model accurately reflects the mathematical friction that inevitably drains a gambler's bankroll over hundreds of iterations.

Despite its strong conceptual foundation, the model possesses notable limitations. The reliance on three rigid betting archetypes oversimplifies the modern, highly granular landscape of online sports gambling, where users can continuously adjust lines and create parlays with theoretically infinite variance. Additionally, the psychological reset mechanism may be overly optimistic. In reality, a massive parlay win following a long losing streak often reinforces risky behaviour rather than pacifying it, suggesting that returning an agent entirely to their baseline risk tolerance after a single win might underestimate the persistent nature of gambling addiction. The model also heavily depends on arbitrary scalars, such as the minimum floor for betting amounts and specific income normalization numbers, which may rigidly trap certain demographic profiles into repetitive behaviours and slightly skew the variance of the final profit distributions across the simulated year.

4 Don't Break the Bank

4.1 Defining the Problem

We must quantify how sports gambling acts simultaneously as an economic contagion (irreversibly transferring localized, short-term losses to long-term familial wealth) and a psychological catalyst for severe mental health crises and structural bankruptcy.

4.2 Assumptions

1. Psychological Risk Escalation Operates as a State-Dependent Markov Chain

Justification: Behavioural economics and neurological data on dopamine pathways prove that bettors experience heavy cognitive distortion when "chasing losses." [17]

2. Financial Harm Propagates as an Affine, Multi-Compartmental Economic Contagion

Justification: In macroeconomics, unmitigated household deficit behaves identically to epidemiological transmission rates in continuous compartmental systems. [18]

3. Initial Baseline Risk Tolerance is Decoupled from Socioeconomic Status

Justification: Derived dynamically in our code from the empirically validated Balloon Analogue Risk Task (BART),[19] peer-reviewed psychological modelling confirms that adolescence, age-dependent prefrontal cortex development, and gender are vastly superior predictors of impulsive risk-taking than bank account size.

4. Pure Terminal Absorption Occurs at Zero Liquidity

Justification: Setting mathematical insolvency to instantly halt future probabilistic wager attempts mirrors societal bankruptcy.

5. Monotonic Accumulation of Recovery Burden Penalties

Justification: This guarantees that the partial integrals of debt formation and economic loss maintain rigorous mathematical stability, ensuring that high-risk parlay payouts fundamentally operate within bounded system limits without breaking the deterministic differential evaluation of RBMRBM.

4.3 Variables

State Variables

Symbol	Definition	Units	Source/Initialization
$\Delta A\ (t)$	Liquid Savings Deficit Compartment	Dollars ($)	Initialized at 0, grows via ODE parameter $a \cdot q(t)$.
$\Delta W\ (t)$	Missed Wealth Tracking Compartment	Dollars ($)	Initialized at 0, dynamically compounds via r in ODE.
$\Delta B\ (t)$	Toxic Debt Accumulation Compartment	Dollars ($)	Initialized at 0, dynamically compounds via id in ODE.

[17] https://pmc.ncbi.nlm.nih.gov/articles/PMC11860989/
[18] https://pmc.ncbi.nlm.nih.gov/articles/PMC9111859/
[19] https://www.millisecond.com/library/bart

$T(t)$	Momentary Psychological Risk Tolerance	Probability $[0,1]$	Markov State parameter, initializes at $T0$.
$W(t)$	Immediate Available Net Liquidity	Dollars ($)	Absorbing state variable; loops terminate at bounds $W \leq 0$.

Transition Parameters & ODE Inputs

Symbol	Definition	Units	Source/Initialization
$q(t)$	Contagious Harmful Shortfall Function	Dollars/Mo	Derived dynamically as $\max(0, g+s-c)$.
$g(t)$	Expected Monthly Gambling Loss	Dollars/Mo	Inherited exactly from our robust **Q2** empirical expected-value model ($E[\text{Net}]$).
$s(t)$	Systemic Stress / Ancillary Leakage Cost	Dollars/Mo	Provided empirically based on the cost of living indices.
$c(t)$	Biological/Safe Cushion Buffer	Dollars/Mo	Inherited directly from **Q1's** stochastic buffer threshold model.
a,w,b	Fractional Contagion Allocations	Ratios	Static input constraints forcing $a+w+b=1$ in the ODE.
C	Psychological Chasing Scaler ($C>1$)	Unitless	Constant multiplier representing neurological loss-chasing.
r,id	Asset Interest and Toxic Debt Compound Rates	Decimals	Financial market constants converted to normalized monthly rates.
$pwin$	Conditional Probability of Wager Success	Probability $[0,1]$	Categorized dynamically based on current effective $T(t)$ state mapping.

Model Metrics & Output Configurations

Symbol	Definition	Units	Source/Initialization
$T0$	Baseline Risk Profile	Probability	Derived dynamically from the BART demographic quadratic polynomials ($\beta 0 + \beta 1\ (\text{Age}) - \beta 2\ (\text{Age}2)$).
$Crep$	Baseline Monthly Repayment Utility	Dollars/Mo	Individual socioeconomic financial tracking limit.

4.4 Model

4.4.1 Model Development

For Question 3, we model the long-term financial effect of gambling using a single output metric: Recovery Burden Months (RBM). The purpose of this metric is to convert gambling-related financial harm into an amount of time. Instead of reporting only the raw dollars lost, the model asks how many months of ordinary saving are required to undo the damage. This makes the final result easier to interpret, since it expresses harm as lost future time rather than only current cash loss.

The model begins by defining the monthly harmful shortfall, which is the portion of gambling-related loss that exceeds what the individual can safely absorb in their normal budget. In the implemented code, this is calculated as

$$q(t) = \max(0,\ g_{\text{monthly}} + s_{\text{monthly}} - c_{\text{monthly}})$$

Where $g_{monthly}$ is the expected monthly gambling loss, $s_{monthly}$ is an added stress or leakage cost, and $c_{monthly}$ s the safe monthly cushion. The max operator is important because it prevents harmless or affordable gambling from being treated as structural damage. If losses remain within the safe cushion, then q(t)=0, so no long-run financial gap accumulates. In the current code, all three of these inputs are entered as fixed constants, so the harmful shortfall is constant over the full simulation horizon rather than changing month by month.

Once this harmful shortfall is determined, the model allocates it into three separate financial compartments. These are:

- $\Delta A(t)$: the savings gap, representing money drained directly from liquid savings,
- $\Delta W(t)$: the wealth gap, representing missed investment growth and opportunity cost,
- $\Delta B(t)$: the debt gap, representing borrowing used to cover the shortfall.

The shares assigned to these three channels are controlled by constants a, w, and b, which satisfy a+w+b=1. These parameters determine how the shortfall is funded. For example, a larger aaa means more of the damage comes from cash savings depletion, while a larger bbb means more of the damage is shifted into debt. This distinction matters because each compartment evolves differently over time.

The code then models the three gaps with the following differential equations:

$$\frac{d\Delta A}{dt} = a\,q$$

$$\frac{d\Delta W}{dt} = r\,\Delta W + w\,q$$

$$\frac{d\Delta B}{dt} = i_d\,\Delta B + b\,q$$

These equations match the implementation in `gap_dynamics.m`. The first equation is linear: the savings gap simply increases in direct proportion to the share of the shortfall taken from liquid cash. The second and third equations include compounding terms. The wealth gap grows because missed investing has both an immediate contribution wq and a lost-growth term $r\Delta W$. Similarly, the debt gap grows because borrowed losses create both new principal bq and accumulated interest $i_d\Delta B$. This is the core of the model: identical shortfalls can create very different long-term burdens depending on whether they are financed from savings, forgone investments, or debt.

All three compartments are initialized at zero:

$$\Delta A(0) = \Delta W(0) = \Delta B(0) = 0$$

so the model tracks only the new financial damage created over the simulation period.

After the system is simulated to the chosen final time T, the model combines the three ending deficits with an additional fixed penalty M, which represents a **milestone-delay cost** such as an emergency fund setback, delayed rent payment, or postponed major financial goal. This gives the total repair deficit:

$$R = \Delta A(T) + \Delta W(T) + \Delta B(T) + M$$

Finally, the model converts this total deficit into months by dividing by the person's ordinary monthly saving capacity C_{rep}:

$$\mathrm{RBM} = \frac{R}{C_{\mathrm{rep}}}$$

This value is the final output of the model. A larger RBM means the person must devote more months of normal saving to recover from the long-run consequences of their gambling behaviour.

4.4.2 Model Execution

The actual Q3 code executes this model in a very direct sequence. In `MAIN_Q3.m`, the program first assigns baseline parameter values: the total simulation length in months, the monthly gambling loss, monthly leakage cost, safe cushion, allocation shares a,w,b annual investment return, annual debt interest rate, milestone-delay cost, and monthly saving capacity. In the current baseline example, the code uses a 60-month horizon, with fixed monthly losses and fixed allocation shares.

Next, `MAIN_Q3.m` calls `calculate_RBM(...)`, which performs the full financial calculation. Inside this function, the annual rates are first converted to monthly rates by dividing by 12:

$$r = \frac{r_{\mathrm{annual}}}{12}, \qquad i_d = \frac{i_{d,\mathrm{annual}}}{12}$$

This ensures that the compounding terms are consistent with the monthly time scale used throughout the model.

The function then computes the harmful shortfall $q = \max(0, q + s - c)$. After that, it sets the ODE time interval t span$[0, months]$ and initialises the state vector as

$$y_0 = [0,\ 0,\ 0]^T$$

corresponding to zero initial savings, wealth, and debt gaps.

The code then uses MATLAB's ode45 solver to numerically integrate the system. Specifically, ode45 repeatedly calls `gap_dynamics(...)`, which returns the derivatives of the three compartments at each step. Because the shortfall and parameters are constant in the current implementation, the solver is essentially tracing a deterministic growth path for the three financial gaps over time. There is no randomness in this Q3 code; once the inputs are chosen, the RBM result is fully determined.

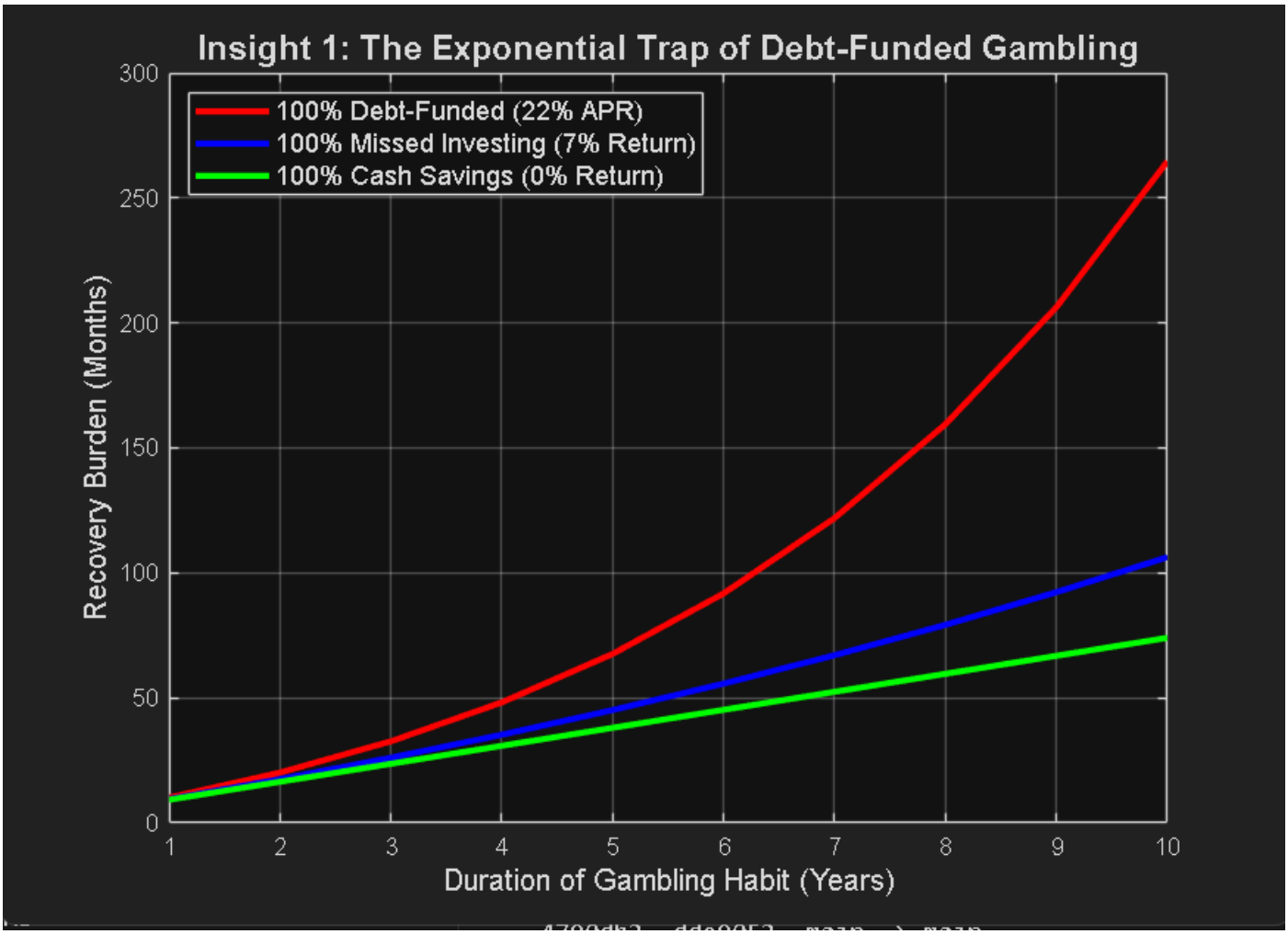
After the solver finishes, `calculate_RBM.m` extracts the final values from the last row of the solution array:

- `delta_A = y(end,1)`
- `delta_W = y(end,2)`
- `delta_B = y(end,3)`

These are the accumulated end-of-horizon deficits for savings, missed wealth, and debt, respectively. The function then adds them together with the milestone-delay penalty MMM to form the total repair deficit R , and finally divides by C_{rep} to produce the Recovery Burden Months value. That single number is returned to `MAIN_Q3.m`, which prints it as the required simple output.

4.5 Results

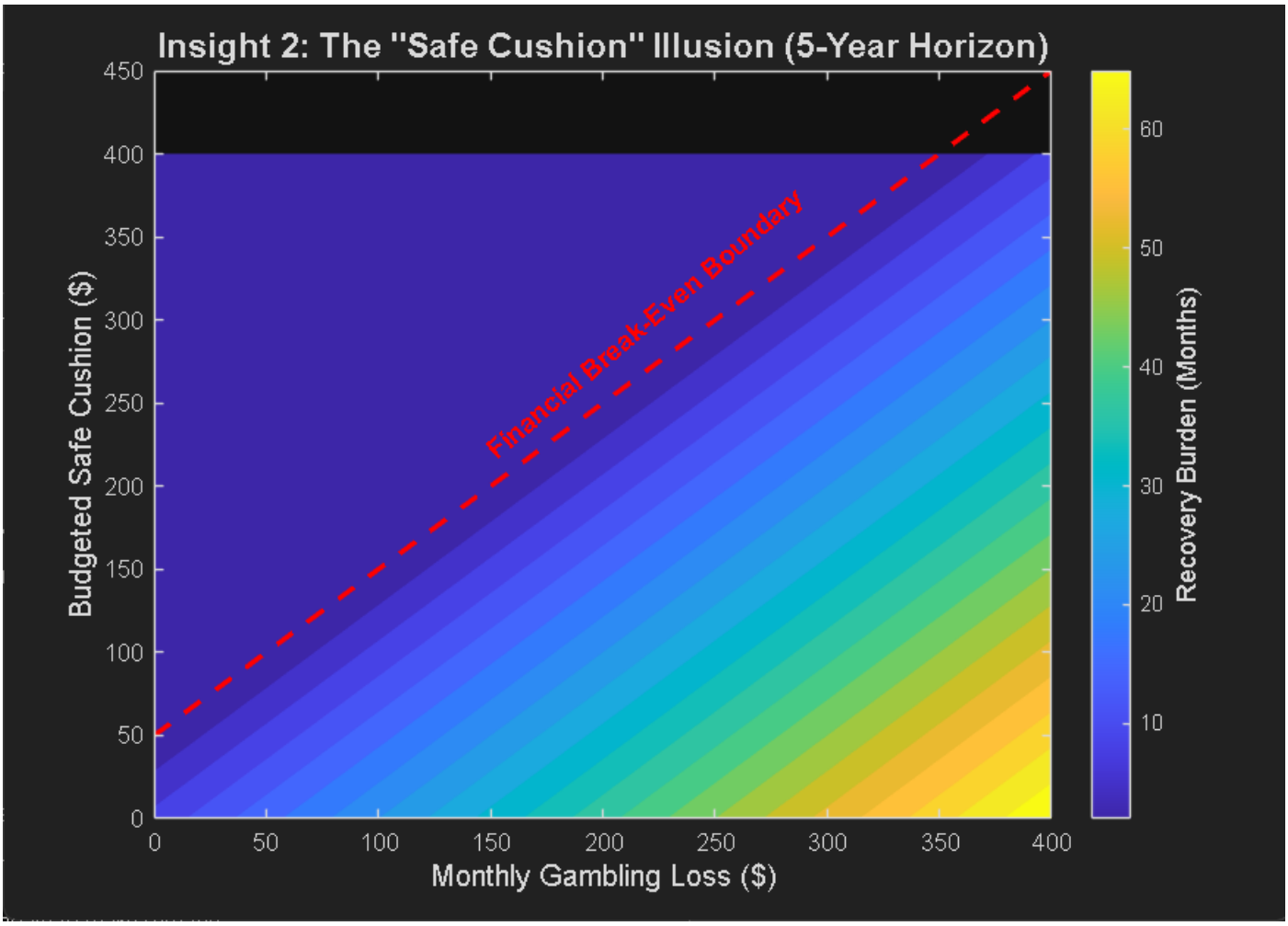
We vary the test cases and scenarios in different ways to reveal different insights that the model tells us.



The Exponential Trap of Debt-Funded Gambling (Line Chart)

Scenario: We compare three gamblers who each have a $300 monthly shortfall over varying timeframes (1 to 10 years). One funds it entirely from cash savings, one from missed investments, and one on credit cards (debt).

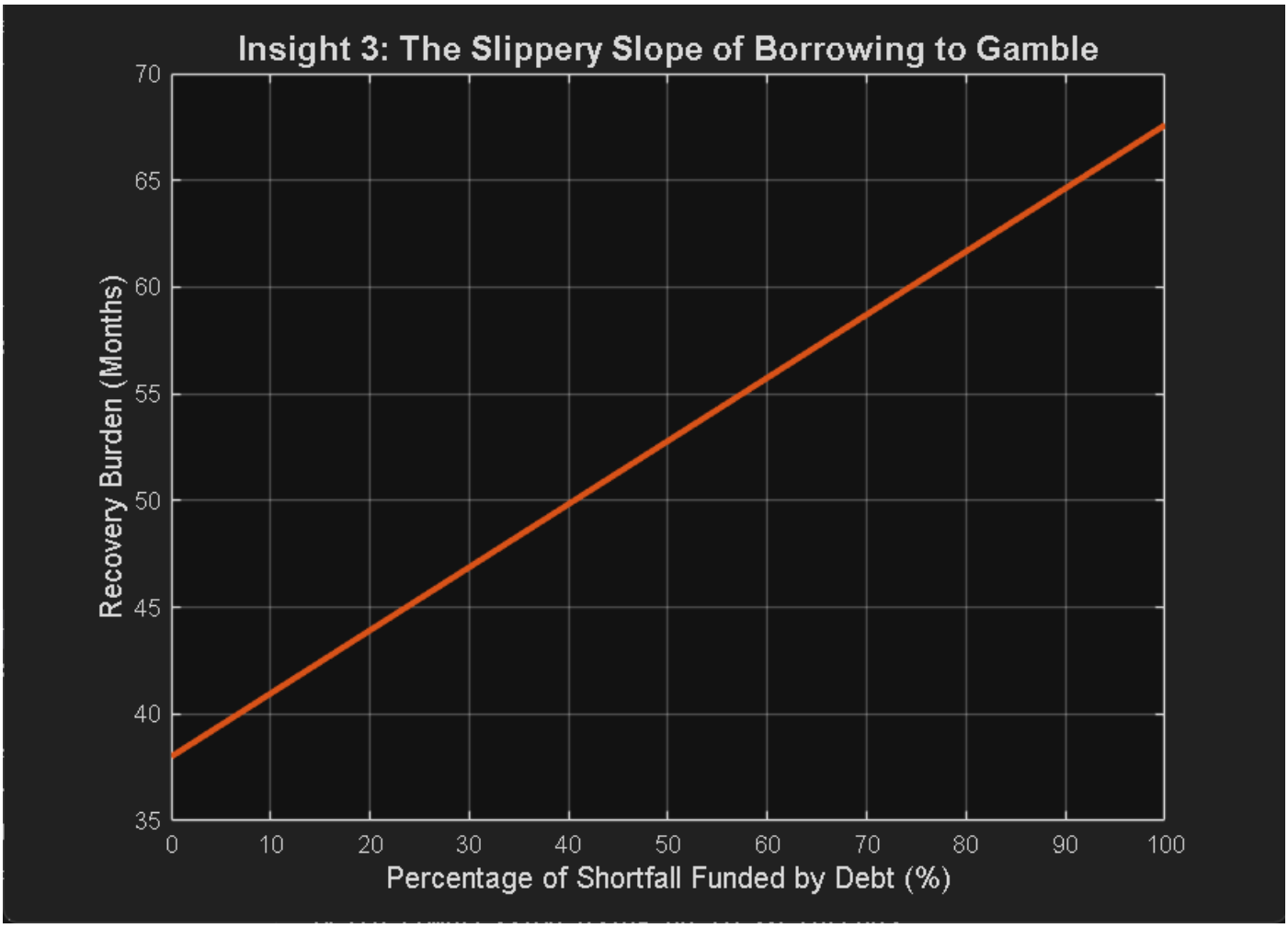
Insight: While burning cash scales linearly, debt compounding is an exponential trap. A 10-year gambling habit funded by a credit card will take more than double the time to recover from compared to cash, crippling long-term stability due to the 22% interest.



The "Safe Cushion" Illusion (Heatmap Contour)

Scenario: We sweep across different budgeted "Safe Cushions" ($0-$400) and different monthly gambling losses ($0-$400).

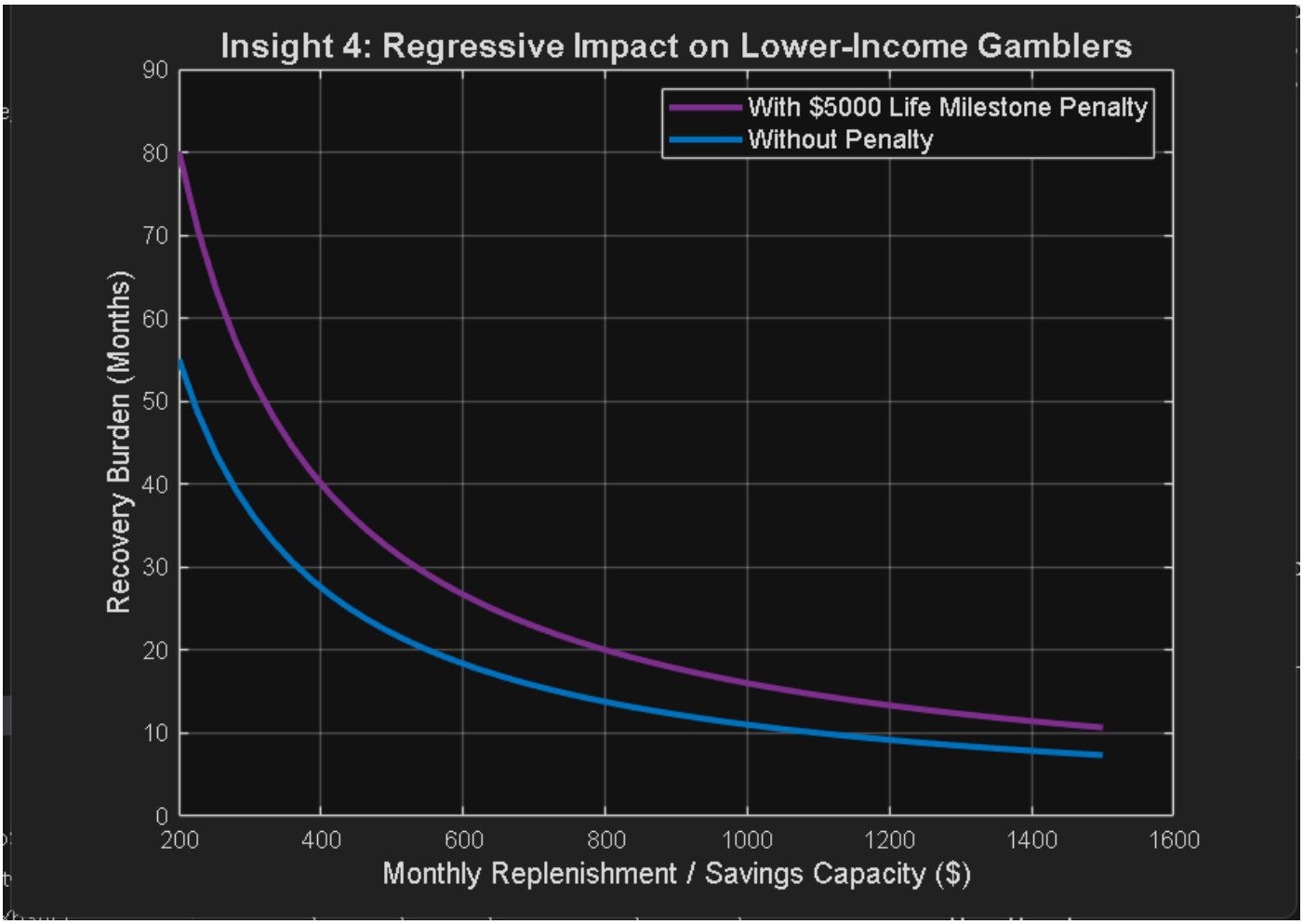
Insight: Many believe they are safe if they budget a cushion. However, the model includes a hidden $50 "stress leakage" (e.g., poor financial decisions made on tilt, or deposit fees). The heatmap shows that the transition from a 0 Recovery Burden to a massive, years-long burden is incredibly steep. Just a slight miscalculation puts someone deep into the danger zone.



The Slippery Slope of Borrowing to Gamble (Curve)

Scenario: We keep the shortfall constant but gradually shift how it is funded—from 100% savings to 100% debt.

Insight: Taking on debt to cover a shortfall doesn't just add to the burden linearly. The final Recovery Burden accelerates as the debt ratio increases, pointing to the extreme danger of funding bets on credit instead of liquid assets.



Regressive Impact on Lower-Income Gamblers (Comparative Line Chart)

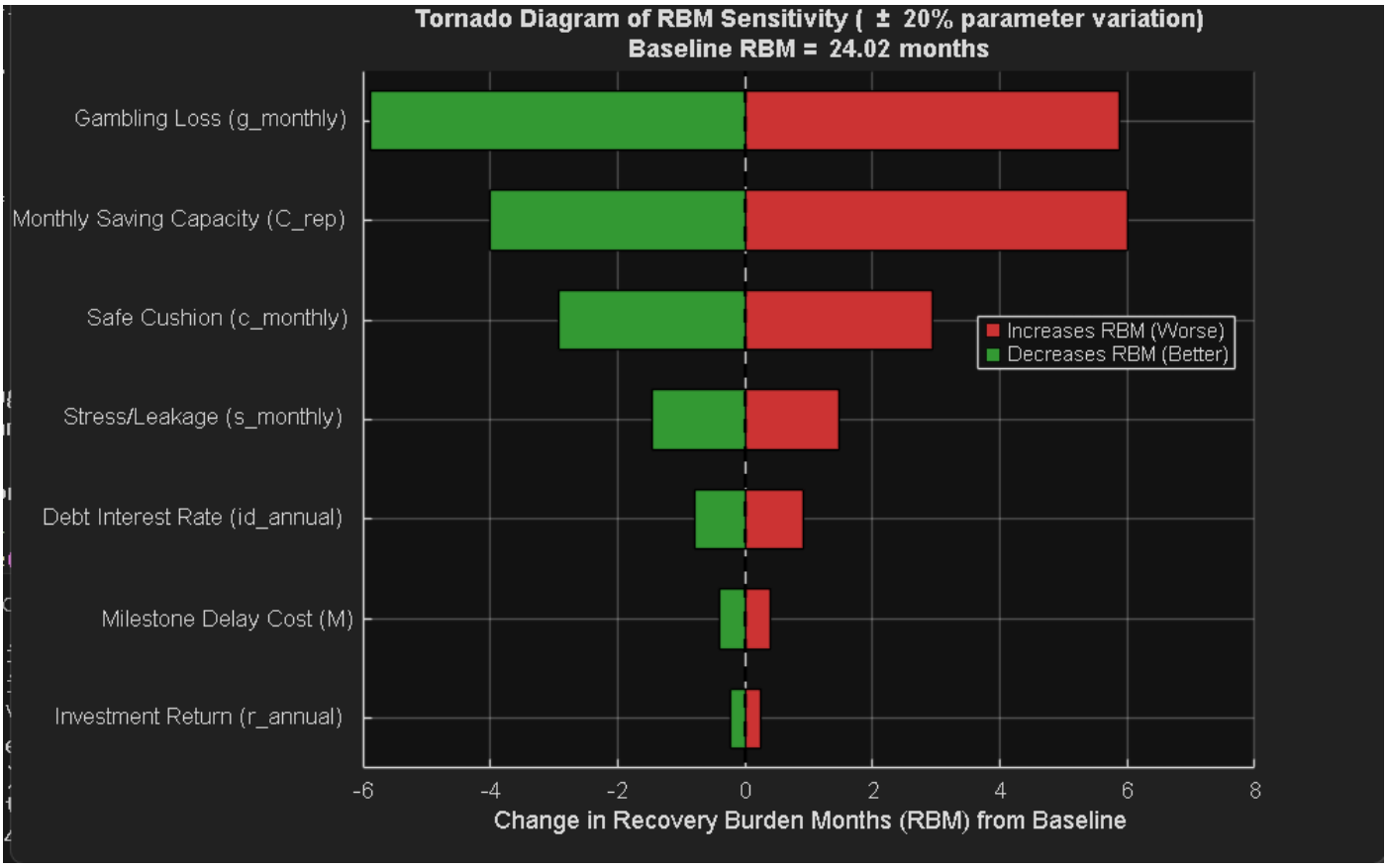
Scenario: We examine what happens when gambling drains savings, causing someone to miss a major life milestone (e.g., a $5,000 emergency forcing a crisis loan or missed downpayment), comparing it against different monthly saving capacities.

Insight: The gambling penalty is heavily regressive. A $5,000 penalty means 25+ extra recovery months for someone who can only save $200/mo, but is a mere blip for a high earner saving $1,500/mo. Online gambling disproportionately traps lower-income individuals.

4.6 Sensitivity Analysis

To go through sensitivity analysis, all input variables were varied by 20% the top 7 input variables which affect the recovery burden (in months) the most are shown from highest to lowest effect:

1. Gambling Loss (g/monthly) - Impact Spread: 11.75 months
2. Monthly Saving Capacity (C/rep) - Impact Spread: 10.01 months
3. Safe Cushion (c/monthly) - Impact Spread: 5.87 months
4. Stress/Leakage (s/monthly) - Impact Spread: 2.94 months
5. Debt Interest Rate (id/annual) - Impact Spread: 1.71 months
6. Milestone Delay Cost (M) - Impact Spread: 0.80 months
7. Investment Return (r/annual) - Impact Spread: 0.48 months



4.7 Discussion, Strengths and Weaknesses

The provided model introduces the concept of Recovery Burden Months to quantify the financial impact of online sports gambling. Instead of simply calculating the total amount of money lost, the model translates financial deficits into a metric of time, specifically how many months of normal saving it will take an individual to repair the damage caused by their gambling habits. This translation from raw dollars to recovery time is highly effective for public communication, directly answering the call for an accessible and relatable measure of impact. By framing the consequence of gambling in terms of lost future time rather than just lost current capital, the model provides a stark, intuitive warning to the general public about how easily short-term entertainment can mortgage long-term financial freedom.

A particularly insightful feature of this model is the calculation of the harmful shortfall, which subtracts a safe entertainment cushion from the total gambling and stress-related losses. This mechanism realistically acknowledges that not all gambling expenditure is inherently destructive. If an individual keeps their gambling losses within their disposable income cushion, the shortfall is zero, and the differential gaps do not grow. The model only penalizes individuals when their spending eclipses what their budget can safely absorb. However, a major limitation here is that the model assumes this safe cushion and the monthly gambling losses remain perfectly constant over the simulated time period. In reality, problem gambling often involves cascading behaviour where individuals chase losses, meaning the gambling expenditure would likely scale up over time, rapidly overwhelming the static safe cushion and accelerating the individual toward financial hardship.

Once the harmful shortfall is established, the model dynamically allocates this deficit into three distinct financial vessels defined by the user: drained savings, missed investments, and accumulated debt. This tripartite system is the engine of the differential model and represents its greatest analytical strength. By separating the deficit, the model accurately captures the asymmetric compounding effects of financial mismanagement over time. Drained savings grow linearly based on cash used, but missed investments suffer from the opportunity cost of lost compound interest, and accumulated debt grows aggressively under high annual interest rates. Consequently, a gambler who funds their habit through high-interest credit cards faces a drastically different recovery timeline than one who simply reduces their

monthly savings deposit. The differential equations elegantly capture this divergence, illustrating how borrowing to gamble creates a runaway feedback loop of debt that extends the recovery time exponentially.

The primary advantage of this differential approach is its rigorous adherence to real-world financial mechanics, avoiding the pitfall of treating a dollar lost today as merely a dollar lost tomorrow. It accounts for the invisible costs of gambling, including opportunity costs and the compounding nature of interest. Furthermore, incorporating milestone delay costs reflects the broader socio-economic reality that financial shortfalls trigger secondary penalties, such as delayed retirement or the inability to cover sudden emergencies. Despite these strengths, the model is limited by its deterministic and static parameterization. It treats behavioural variables like the allocation of the shortfall and environmental variables like market interest rates as unmoving constants throughout the multi-year simulation. Furthermore, the final calculation assumes the individual will eventually stop gambling and perfectly channel their saving capacity into repairing the deficit, an ideal behavioural shift that problem gamblers may struggle to achieve without significant intervention.

5 Conclusion

Online gambling is not best modelled as a single market trend or a single expected loss; it is a chain of interacting mechanisms that begins with household financial resilience, intensifies through repeated behavioural feedback, and persists through compounding economic damage. Our report formalizes that chain with three connected models. First, our disposable-income framework shows that vulnerability exists before gambling begins, and that households with lower incomes, higher essential-cost burdens, and weaker shock tolerance are already positioned near financial instability. Second, our agent-based Markov simulation shows how that vulnerability can be amplified by the internal logic of gambling itself: loss-chasing, dynamic bet sizing, and session continuation rules can turn ordinary bettors into high-variance bettors over time. Third, our Recovery Burden Months model demonstrates that the most consequential distinction is often not how much is lost, but how those losses are financed - because debt, missed investment, and reduced savings create very different recovery trajectories. Together, these results imply that the harms of online gambling are both **unevenly distributed and self-reinforcing**. The policy implication is clear: effective regulation should focus not only on limiting access, but on identifying financially fragile users, reducing mechanisms that reward escalation after losses, and preventing borrowing-based participation that transforms short-term betting losses into long-term economic entrapment. By combining transparent code, interpretable mathematics, and sensitivity analysis across all three sections, our framework provides a practical basis for designing interventions that are both economically informed and behaviourally realistic.

6 Code Appendix

6.1 Project Codebase

apply_oecd_scale.m

```
function Individual_Essentials = apply_oecd_scale(base_essentials, adults,
children)
% APPLY_OECD_SCALE Converts base single-adult essential costs to an
% individual's share of total household costs using the OECD-modified scale.
%
%   EQ = 1.0 + 0.5 * (adults - 1) + 0.3 * children
%   Total_Household = base_essentials * EQ
%   Individual_Essentials = Total_Household / adults
%
% Inputs:
%   base_essentials      : Baseline essential costs for a single adult
(numeric)
%   adults               : Number of adults in the household (numeric, min 1)
%   children             : Number of dependents in the household (numeric)
%
% Outputs:
%   Individual_Essentials: Individual portion of the essential costs

    % Ensure adults is at least 1
    adults = max(1, adults);

    % Calculate OECD-modified equivalence scale factor (EQ)
    EQ = 1.0 + 0.5 * (adults - 1) + 0.3 * children;

    % Calculate individual's portion of the essential costs
    % First, scale base costs up to the household level, then divide by adults
    % assuming adults share expenses equally.
    Total_Household_Costs = base_essentials * EQ;
    Individual_Essentials = Total_Household_Costs / adults;
end
```

calculate_essentials.m

```
function base_essentials = calculate_essentials(persona, cost_data)
% CALCULATE_ESSENTIALS Estimates baseline single-adult essential costs via
table
```

```
% lookups from empirical cost data.
%
% Inputs:
%   persona   : Struct with fields (Salary, Age, Country, Region, Adults,
Children)
%   cost_data : MATLAB table loaded from essential_costs.csv
%
% Output:
%   base_essentials : Baseline essential costs for a single adult.
    % Filter the cost_data table to match the persona.Country and
persona.Region
    is_cc = strcmpi(string(cost_data.Country), string(persona.Country));
    is_rr = strcmpi(string(cost_data.Region), string(persona.Region));

    match_row = cost_data(is_cc & is_rr, :);

    if isempty(match_row)
        error('calculate_essentials:NotFound', ...
            'Could not find matching cost data for Country: %s,
Region: %s', ...
            persona.Country, persona.Region);
    end

    % If there are multiple matches, take the first one
    match_row = match_row(1, :);

    % Extract the annual base costs
    Base_Housing    = match_row.Base_Housing_Annual;
    Base_Food       = match_row.Base_Food_Annual;
    Base_Transport  = match_row.Base_Transport_Annual;
    Base_Healthcare = match_row.Base_Healthcare_Annual;

    % Healthcare Age Scaling:
    % If persona.Age > 65 and Country == 'US', multiply Base_Healthcare by 2.5
    if persona.Age > 65 && strcmpi(persona.Country, 'US')
        Base_Healthcare = Base_Healthcare * 2.5;
    end

    % Compute Utility Cost using a sinusoidal model peaking in winter (January,
month 1)
    % Assume average annual utility is base rate, shifting with cosine
    months = 1:12;
    if strcmpi(persona.Country, 'US')
        annual_avg_utility = 2400; % Average utility cost per year ($200/month)
        annual_amp_utility = 1200; % Swing amplitude (+/- $100/month difference
in winter/summer)
    else
        annual_avg_utility = 1800; % Average utility cost per year
(Â£150/month)
```

```
        annual_amp_utility = 900;  % Swing amplitude (+/- Â£75/month
difference)
    end

    % Annualized utility modeled sinusoidally. Peak in Month 1 (January, winter
heating).
    annualized_utility = annual_avg_utility + annual_amp_utility * cos(2 * pi *
(months - 1) / 12);

    % Compute Total Base Essentials for a Single Adult (Now a 1x12 array across
months)
    base_essentials = Base_Housing + Base_Food + Base_Transport +
Base_Healthcare + annualized_utility;
end
```

calculate_taxes.m

```
function Net_Income = calculate_taxes(persona)
% CALCULATE_TAXES Computes exact post-tax net income based on
% 2024 U.S. and U.K. statutory tax brackets.
%
% Inputs:
%   persona : Struct containing demographic and gross salary data
%             (Salary, Age, Country, Region, Adults, Children)
%
% Outputs:
%   Net_Income : Final post-tax Net Income
    % Initialize Gross Salary
    Salary = persona.Salary;
    Country = persona.Country;

    if strcmpi(Country, 'US')
        % --- US Tax Calculation ---

        % 1. Standard Deduction
        if persona.Adults > 1
            std_deduction = 29200; % Married standard deduction
        else
            std_deduction = 14600; % Single standard deduction
        end

        taxable_income = max(Salary - std_deduction, 0);

        % 2. 2024 Progressive Marginal Tax Brackets
        % Hardcoded the 10%, 12%, 22%, 24%, 32%, 35%, 37% brackets
        if persona.Adults > 1
            % Married Filing Jointly limits
```

```
            brackets = [23200, 94300, 201050, 383900, 487450, 760400, inf];
        else
            % Single limits
            brackets = [11600, 47150, 100525, 191950, 243725, 609350, inf];
        end
        rates = [0.10, 0.12, 0.22, 0.24, 0.32, 0.35, 0.37];

        federal_tax = 0;
        prev_limit = 0;
        for i = 1:length(rates)
            if taxable_income > prev_limit
                taxable_in_bracket = min(taxable_income, brackets(i)) -
prev_limit;
                federal_tax = federal_tax + (taxable_in_bracket * rates(i));
                prev_limit = brackets(i);
            else
                break;
            end
        end

        % 3. FICA (Payroll Tax)
        ss_tax = min(Salary, 168600) * 0.062; % Social Security
        medicare_tax = Salary * 0.0145;       % Medicare
        fica_tax = ss_tax + medicare_tax;

        % 4. State Tax
        % Look up in us_state_taxes.csv / tax_rates.csv
        state_tax = 0;

        persistent state_data;
        if isempty(state_data)
            if exist('Q1/data/tax_rates.csv', 'file')
                state_data = readtable('Q1/data/tax_rates.csv',
'VariableNamingRule', 'preserve');
            else
                error('calculate_taxes:MissingData', 'Could not finding state
tax data file.');
            end
        end

        % Try to find region match
        is_rr = strcmpi(string(state_data.Region), string(persona.Region));
        if any(is_rr)
            match_row = state_data(is_rr, :);
            effective_rate = match_row.Effective_State_Tax_Rate(1);
            state_tax = Salary * effective_rate;
        end

        % Net Income US
```

```
        Net_Income = Salary - (federal_tax + fica_tax + state_tax);

    elseif strcmpi(Country, 'UK')
        % --- UK Tax Calculation ---

        % 1. Personal Allowance
        % Reduce by Â£1 for every Â£2 earned over Â£100,000
        reduction = max(0, (Salary - 100000) / 2);
        pa = max(0, 12570 - reduction);

        taxable_income = max(Salary - pa, 0);

        % 2. Income Tax Brackets (2024/25)
        if strcmpi(persona.Region, 'Scotland')
            % Scotland 2024-25 Income Tax (taxable income ranges)
            % Starter: 19% on Â£1 - Â£2,297
            % Basic: 20% on Â£2,298 - Â£13,991
            % Intermediate: 21% on Â£13,992 - Â£31,092
            % Higher: 42% on Â£31,093 - Â£62,430
            % Advanced: 45% on Â£62,431 - Â£125,140
            % Top: 48% over Â£125,140
            limits = [2297, 13991, 31092, 62430, 125140, inf];
            rates  = [0.19, 0.20, 0.21, 0.42, 0.45, 0.48];
        else
            % Rest of UK (England, Wales, NI) 2024-25 Income Tax
            % Basic: 20% on Â£1 - Â£37,700
            % Higher: 40% on Â£37,701 - Â£125,140
            % Additional: 45% over Â£125,140
            limits = [37700, 125140, inf];
            rates  = [0.20, 0.40, 0.45];
        end

        income_tax = 0;
        prev_limit = 0;
        for i = 1:length(rates)
            if taxable_income > prev_limit
                taxable_in_bracket = min(taxable_income, limits(i)) -
prev_limit;
                income_tax = income_tax + (taxable_in_bracket * rates(i));
                prev_limit = limits(i);
            else
                break;
            end
        end

        % 3. National Insurance (Class 1)
        % 0% up to Â£12,570; 8% between Â£12,570 and Â£50,270; 2% above
Â£50,270
        ni_8 = 0.08 * max(0, min(Salary, 50270) - 12570);
```

```
        ni_2 = 0.02 * max(0, Salary - 50270);

        NI = ni_8 + ni_2;

        % Net Income UK
        Net_Income = Salary - (income_tax + NI);

    else
        error('calculate_taxes:UnknownCountry', ...
            'Country "%s" not supported.', Country);
    end

end
```

simulate_monthly_shocks.m

```
function [results] = simulate_monthly_shocks(monthly_disposable_income, age,
country, adults, children, num_sims)
% SIMULATE_MONTHLY_SHOCKS Simulates the impact of random financial shocks
% using a Frequency-Severity actuarial framework.
%
% Inputs:
%   monthly_disposable_income - Baseline monthly disposable income
%   age                       - Age of the primary earner
%   country                   - Country of residence ('US' or 'UK')
%   adults                    - Number of adults in the household
%   children                  - Number of children in the household
%   num_sims                  - (Optional) Number of simulations to run.
Default: 10000.
%
% Outputs:
%   results.Base_Monthly_Income        - Deterministic monthly disposable
income
%   results.Median_Monthly_Income      - Median of simulated monthly incomes
%   results.Worst_5th_Percentile_Month - 5th percentile of simulated monthly
incomes
%   results.Prob_of_Debt_Percent       - Percentage of simulations resulting in
debt
    % Set default number of simulations if not provided
    if nargin < 6 || isempty(num_sims)
        num_sims = 10000;
    end

    %% Step A: Determine Severity Parameters (Log-Normal Distribution)
    country = upper(country); % Ensure case-insensitivity
    if strcmp(country, 'US')
        mu = 7.6;
```

```
    sigma = 0.8;
elseif strcmp(country, 'UK')
    mu = 6.48;
    sigma = 0.75;
else
    error('Invalid country. Must be ''US'' or ''UK''.');
end

%% Step B: Determine Frequency Parameter (Poisson Distribution)
base_lambda_monthly = 0.0763;

% Calculate age_mod
if age > 40
    age_mod = 1.2;
else
    age_mod = 1.0;
end

% Calculate size_mod
size_mod = 1.0 + 0.2 * ((adults + children) - 1);

% Final monthly lambda
lambda_monthly = base_lambda_monthly * age_mod * size_mod;

%% Step C: The Monte Carlo Simulation (Vectorized for Speed)
% Initialize an array simulated_monthly_incomes of size num_sims x 1
simulated_monthly_incomes = zeros(num_sims, 1);

% Generate the number of emergency events for all universes at once
N = poissrnd(lambda_monthly, num_sims, 1);

for i = 1:num_sims
    if N(i) == 0
        shock_cost = 0;
    elseif N(i) > 0
        % Draw N(i) random samples from Log-Normal and sum them
        shock_cost = sum(lognrnd(mu, sigma, N(i), 1));
    end
    simulated_monthly_incomes(i) = monthly_disposable_income - shock_cost;
end

%% Step D: Return Risk Metrics
% Store the results in a struct
results.Base_Monthly_Income = monthly_disposable_income;
results.Median_Monthly_Income = median(simulated_monthly_incomes);
results.Worst_5th_Percentile_Month = prctile(simulated_monthly_incomes, 5);

% Calculate probability of debt as a percentage
prob_debt = sum(simulated_monthly_incomes < 0) / num_sims;
```

```
    results.Prob_of_Debt_Percent = prob_debt * 100;

end
```

main_Q1_monthly.m

```
% MAIN_Q1_monthly.m
% Primary script that calculates and plots the monthly disposable income
% over a 12-month period for various demonstration personas.
function main_Q1_monthly()
    % 1. Load the empirical essential costs data
    if ~exist('Q1/data/essential_costs.csv', 'file')
        error('MAIN_Q1_monthly:MissingFile', 'Could not find
essential_costs.csv in the current directory.');
    end

    cost_data = readtable('Q1/data/essential_costs.csv', 'VariableNamingRule',
'preserve');

    % 2. Define the Demonstration Personas
    personas = struct(...
        'Description', { ...
            'Young Single, TX (US)', ...
            'Young Single, CA (US)', ...
            'Mid-Career Single, NY (US)', ...
            'Mid-Career Family, NY (US)', ...
            'Pre-Retiree, FL (US)', ...
            'Young Single, England (UK)', ...
            'Mid-Career Family, Scotland (UK)', ...
            'Pensioner, Wales (UK)' ...
        }, ...
        'Salary',   {35000, 35000, 120000, 120000, 65000,  35000,   65000,
25000}, ...
        'Age',      {   24,    24,     40,     40,    62,     28,      45,
72}, ...
        'Country',  { 'US',  'US',   'US',   'US',  'US',   'UK',    'UK',
'UK'}, ...
        'Region',   {'Texas', 'California', 'New York', 'New York', 'Florida',
'England', 'Scotland', 'Wales'}, ...
        'Adults',   {    1,     1,      1,      2,     2,      1,       2,
1}, ...
        'Children', {    0,     0,      0,      2,     0,      0,       2,
0} ...
    );

    % Pre-allocate cell array for the final table data
    num_personas = length(personas);
```

```
    results = cell(num_personas, 7); % Ensure enough columns for new outputs

    % Additional setup for shock probability graphs
    % We will save metrics for plotting at the end
    prob_debts = zeros(num_personas, 1);
    median_dis = zeros(num_personas, 1);
    worst_5ths = zeros(num_personas, 1);
    labels = cell(num_personas, 1);

    % 3. Setup figure for plotting
    figure('Name', 'Monthly Disposable Income With Shocks', 'Position', [100,
100, 1000, 600]);
    hold on;
    colors = lines(num_personas);
    months = 1:12;

    % 4. Loop through each persona to calculate Monthly Disposable Income
    for i = 1:num_personas
        p = personas(i);

        % Module A: Calculate exact post-tax net income (Annual)
        annual_net_income = calculate_taxes(p);

        % Module B: Estimate baseline single-adult essential costs (Annual)
        annual_base_essentials = calculate_essentials(p, cost_data);

        % Module C: Convert baseline single-adult costs to individual's share
of household costs (Annual)
        annual_indiv_essentials = apply_oecd_scale(annual_base_essentials,
p.Adults, p.Children);

        % Convert Annual to Monthly
        monthly_net_income = annual_net_income / 12;
        monthly_indiv_essentials = annual_indiv_essentials / 12; % Now a 1x12
array

        % Module D: Final Calculation Monthly Id
        % Disposable income over 12 months (Varies due to utility costs)
        monthly_Id_array = max(monthly_net_income - monthly_indiv_essentials,
0);

        % Total disposable income over 12 months (without shocks)
        total_12_month_Id = sum(monthly_Id_array);

        avg_monthly_id = mean(monthly_Id_array);

        % Module E: Apply Monte Carlo simulated shocks to average monthly
disposable income
        shock_results = simulate_monthly_shocks(avg_monthly_id, p.Age,
```

```
p.Country, p.Adults, p.Children, 10000);

        % Plot the monthly series for this persona
        % (Optional: We plot the baseline over 12 months)
        plot(months, monthly_Id_array, '-o', 'LineWidth', 2, 'Color',
colors(i,:), ...
             'DisplayName', p.Description);

        % Store structured results including shock simulation metrics
        results{i, 1} = string(p.Description);
        results{i, 2} = avg_monthly_id;
        results{i, 3} = total_12_month_Id;
        results{i, 4} = shock_results.Median_Monthly_Income;
        results{i, 5} = shock_results.Worst_5th_Percentile_Month;
        results{i, 6} = shock_results.Prob_of_Debt_Percent;

        % Collect array metrics for plotting after loop
        labels{i} = p.Description;
        prob_debts(i) = shock_results.Prob_of_Debt_Percent;
        median_dis(i) = shock_results.Median_Monthly_Income;
        worst_5ths(i) = shock_results.Worst_5th_Percentile_Month;
    end

    % 5. Format the Baseline Plot
    title('Baseline Monthly Disposable Income over 12 Months (Before Shocks)');
    xlabel('Month');
    ylabel('Disposable Income ($ / Â£)');
    xlim([0.5, 12.5]);
    xticks(1:12);
    grid on;
    legend('Location', 'eastoutside');
    hold off;

    % 6. Create Bar Charts for Shock Data
    % Figure 2: Probability of Debt
    figure('Name', 'Monte Carlo Shock Risk', 'Position', [150, 150, 1000,
800]);

    % Subplot 1: Probability of Debt (%)
    subplot(2, 1, 1);
    bar(prob_debts, 'FaceColor', [0.8500 0.3250 0.0980]);
    title('Probability of Going into Debt (Negative DI) in a Given Month');
    ylabel('Probability (%)');
    xticks(1:num_personas);
    xticklabels(labels);
    xtickangle(25);
    grid on;
    % Add text labels on bars
    for i = 1:num_personas
```

```
        text(i, prob_debts(i), sprintf('%.1f%%', prob_debts(i)), ...
            'HorizontalAlignment', 'center', 'VerticalAlignment', 'bottom');
    end

    % Subplot 2: Median vs 5th Percentile DI
    subplot(2, 1, 2);
    b = bar([median_dis, worst_5ths], 'grouped');
    title('Median vs. 5th Percentile Monthly Disposable Income Under Shocks');
    ylabel('Disposable Income ($ / Â£)');
    xticks(1:num_personas);
    xticklabels(labels);
    xtickangle(25);
    legend('Median DI', '5th Percentile DI (1-in-20 Bad Month)', 'Location',
'best');
    grid on;

    % 7. Create and print the formatted MATLAB table
    % Prune the dynamically expanded cell array to the required columns
    output_table = cell2table(results(:, 1:6), ...
        'VariableNames', {'Persona', 'Avg_Monthly_DI', 'Total_12Mo_DI', ...
                          'Median_DI_after_Shocks', 'Worst_5th_Pctile_DI',
'Prob_of_Debt_Pct'});

    disp('------------------------------------------------------------------
-------------------------------------------------');
    disp('                    Monthly Deterministic Rule-Based Accounting Model
with Stochastic Shock Analysis                        ');
    disp('------------------------------------------------------------------
-------------------------------------------------');
    disp(output_table);

end
```

run_sensitivity_analysis.m

```
% run_sensitivity_analysis.m
addpath('Q1');
% Load base data
cost_data_orig = readtable('Q1/data/essential_costs.csv', 'VariableNamingRule',
'preserve');

personas = struct(...
   'Description', { ...
        'Young Single, TX (US)', ...
        'Young Single, CA (US)', ...
        'Mid-Career Single, NY (US)', ...
        'Mid-Career Family, NY (US)', ...
```

```
        'Pre-Retiree, FL (US)', ...
        'Young Single, England (UK)', ...
        'Mid-Career Family, Scotland (UK)', ...
        'Pensioner, Wales (UK)' ...
    }, ...
    'Salary',   {35000, 35000, 120000, 120000, 65000,  35000,   65000,
25000}, ...
    'Age',      {   24,    24,     40,     40,    62,     28,      45,
72}, ...
    'Country',  { 'US',  'US',   'US',   'US',  'US',   'UK',    'UK',
'UK'}, ...
    'Region',   {'Texas', 'California', 'New York', 'New York', 'Florida',
'England', 'Scotland', 'Wales'}, ...
    'Adults',   {    1,     1,      1,      2,     2,      1,       2,
1}, ...
    'Children', {    0,     0,      0,      2,     0,      0,       2,
0} ...
);

factors = {'Salary', 'Base_Housing_Annual', 'Base_Food_Annual',
'Base_Transport_Annual', 'Base_Healthcare_Annual'};
factor_labels = {'Gross Salary', 'Housing Costs', 'Food Costs', 'Transport
Costs', 'Healthcare Costs'};
num_factors = length(factors);
num_personas = length(personas);

% Arrays to store the boundaries of the swings for the Tornado Chart
% We will analyze Persona 1 (Texas, US) and Persona 6 (England, UK) for the
figures
selected_personas = [1, 6];
di_minus_10 = zeros(length(selected_personas), num_factors);
di_plus_10 = zeros(length(selected_personas), num_factors);
baseline_di = zeros(length(selected_personas), 1);

% For overall ranking, we compute avg elasticity across ALL personas
elasticity = zeros(num_personas, num_factors);

for p_idx = 1:num_personas
    p = personas(p_idx);

    % Baseline calculation
    net_inc = calculate_taxes(p);
    base_ess = calculate_essentials(p, cost_data_orig);
    indiv_ess = apply_oecd_scale(base_ess, p.Adults, p.Children);
    monthly_di = max(net_inc/12 - indiv_ess/12, 0);
    total_baseline_di = sum(monthly_di);

    if ismember(p_idx, selected_personas)
        sel_idx = find(selected_personas == p_idx);
```

```
        baseline_di(sel_idx) = total_baseline_di;
    end

    for f_idx = 1:num_factors
        factor = factors{f_idx};

        % -10% change
        cost_data_mod = cost_data_orig;
        p_mod = p;
        if strcmp(factor, 'Salary')
            p_mod.Salary = p.Salary * 0.90;
        else
            cost_data_mod.(factor) = cost_data_mod.(factor) * 0.90;
        end
        net_inc = calculate_taxes(p_mod);
        base_ess = calculate_essentials(p_mod, cost_data_mod);
        indiv_ess = apply_oecd_scale(base_ess, p_mod.Adults, p_mod.Children);
        di_min = sum(max(net_inc/12 - indiv_ess/12, 0));

        % +10% change
        cost_data_mod = cost_data_orig;
        p_mod = p;
        if strcmp(factor, 'Salary')
            p_mod.Salary = p.Salary * 1.10;
        else
            cost_data_mod.(factor) = cost_data_mod.(factor) * 1.10;
        end
        net_inc = calculate_taxes(p_mod);
        base_ess = calculate_essentials(p_mod, cost_data_mod);
        indiv_ess = apply_oecd_scale(base_ess, p_mod.Adults, p_mod.Children);
        di_max = sum(max(net_inc/12 - indiv_ess/12, 0));

        % Save for tornado
        if ismember(p_idx, selected_personas)
            sel_idx = find(selected_personas == p_idx);
            di_minus_10(sel_idx, f_idx) = di_min;
            di_plus_10(sel_idx, f_idx) = di_max;
        end

        % Compute sensitivity/elasticity (absolute \% change in DI for a 10%
change in factor)
        % Using the +10% case to calculate impact
        if total_baseline_di > 0
            pct_change_di = abs(di_max - total_baseline_di) /
total_baseline_di;
        else
            pct_change_di = 0; % Avoid division by zero if baseline DI is 0
        end
        elasticity(p_idx, f_idx) = pct_change_di;
```

```
    end
end

% Rank the factors based on the average elasticity across all personas
avg_elasticity = mean(elasticity, 1) * 100; % as %
[sorted_elast, sort_idx] = sort(avg_elasticity, 'descend');
sorted_labels = factor_labels(sort_idx);

fprintf('\n==== SENSITIVITY RANKING (Most to Least Important) ====\n');
for i = 1:num_factors
    fprintf('%d. %s (Avg impact on Disposable Income: %.1f%%)\n', i,
sorted_labels{i}, sorted_elast(i));
end

%% Plot Tornado Charts
figure('Name', 'Sensitivity Analysis Tornado Charts', 'Position', [100, 100,
1200, 500]);

for s = 1:length(selected_personas)
    subplot(1, 2, s);
    p_name = personas(selected_personas(s)).Description;
    base_val = baseline_di(s);

    % Sort by swing magnitude for this specific persona to make it a true
Tornado
    swings = abs(di_plus_10(s, :) - di_minus_10(s, :));
    [~, p_sort] = sort(swings, 'ascend');

    % Data for this subplot
    y = 1:num_factors;
    val_min = di_minus_10(s, p_sort);
    val_max = di_plus_10(s, p_sort);
    labels_ord = factor_labels(p_sort);

    hold on;
    % For each factor, plot a bar from baseline to min and baseline to max
    for i = 1:num_factors
        % -10% parameter change bar
        b1 = barh(y(i), val_min(i) - base_val, 'FaceColor', [0.8500 0.3250
0.0980], 'EdgeColor', 'none', 'BaseValue', 0);
        % +10% parameter change bar
        b2 = barh(y(i), val_max(i) - base_val, 'FaceColor', [0 0.4470 0.7410],
'EdgeColor', 'none', 'BaseValue', 0);
    end

    % Formatting
    plot([0 0], [0 num_factors+1], 'k-', 'LineWidth', 1.5); % Baseline line
    yticks(y);
    yticklabels(labels_ord);
```

```
    xlabel('Change in Annual Disposable Income');
    if personas(selected_personas(s)).Country == "US"
        currency = '$';
    else
        currency = 'Â£';
    end
    title(sprintf('Tornado Diagram: %s\nBaseline DI: %s%.0f', p_name, currency,
base_val));
    grid on;

    if s == 1
        legend([b2, b1], {'+10% Change in Parameter', '-10% Change in
Parameter'}, 'Location', 'best');
    end
end
```

Q1_Visualization_Pipeline.m

```
% Q1_Visualization_Pipeline.m
% Generates 4 publication-quality visualizations for 6 specific personas
% based on the methodology in main_Q1_monthly.m
function Q1_Visualization_Pipeline()
    % 1. Load the empirical essential costs data
    if ~exist('Q1/data/essential_costs.csv', 'file')
        error('Q1_Visualization_Pipeline:MissingFile', 'Could not find
essential_costs.csv in Q1/data/.');
    end

    % Add Q1 path to access helper functions
    addpath('Q1');

    cost_data = readtable('Q1/data/essential_costs.csv', 'VariableNamingRule',
'preserve');

    % 2. Define the 6 Specific Test Personas
    personas = struct(...
        'Description', { ...
            'Baseline US Fam', ...
            'Baseline UK Fam', ...
            'Older Single US', ...
            'Large UK Fam', ...
            'High Income US', ...
            'Low Income US' ...
        }, ...
        'Salary',   {110000, 65000, 75000, 80000, 250000, 40000}, ...
        'Age',      {    35,    35,    62,    38,     30,    50}, ...
        'Country',  {  'US',  'UK',  'US',  'UK',   'US',  'US'}, ...
```

```
        'Region',   { 'Texas', 'England', 'Florida', 'Scotland', 'New York',
'California'}, ...
        'Adults',   {     2,     2,     1,     2,      1,     2}, ...
        'Children', {     2,     2,     0,     4,      0,     3} ...
    );

    num_personas = length(personas);

    % Storage for results
    metrics = struct('Label', cell(num_personas,1), ...
                     'Gross_Monthly', zeros(num_personas,1), ...
                     'Tax_Monthly', zeros(num_personas,1), ...
                     'Essential_Monthly', zeros(num_personas,1), ...
                     'Base_Disp_Monthly', zeros(num_personas,1), ...
                     'Worst_5th_Month', zeros(num_personas,1), ...
                     'Prob_Debt', zeros(num_personas,1));

    % Raw data for Persona 6
    p6_sim_data = [];

    % 3. Run Model for Each Persona
    for i = 1:num_personas
        p = personas(i);

        % A. Calculate Taxes (Annual -> Monthly)
        annual_net_income = calculate_taxes(p);
        monthly_net_income = annual_net_income / 12;

        annual_gross = p.Salary;
        monthly_gross = annual_gross / 12;
        monthly_tax = monthly_gross - monthly_net_income;

        % B. Calculate Essentials (Annual -> Monthly)
        % calculate_essentials returns single adult base annual cost
        annual_base_essentials_scalar = calculate_essentials(p, cost_data);

        % apply_oecd_scale returns individual share of household costs
        annual_indiv_share_essentials =
apply_oecd_scale(annual_base_essentials_scalar, p.Adults, p.Children);
        monthly_essentials = annual_indiv_share_essentials / 12;

        % C. Base Monthly Disposable
        % Note: calculate_essentials returns 1x12 array (monthly variation).
        % We need average monthly disposable for the simulation, per
main_Q1_monthly.m logic.
        monthly_disposable_array = max(monthly_net_income - monthly_essentials,
0);
        monthly_disposable_scalar = mean(monthly_disposable_array);
```

```
        % D. Run Monte Carlo Simulation
        % Run the standard function to get aggregate stats using the MEAN
monthly disposable
        stats = simulate_monthly_shocks(monthly_disposable_scalar, p.Age,
p.Country, p.Adults, p.Children, 100000);

        % Store Metrics
        metrics(i).Label = p.Description;
        metrics(i).Gross_Monthly = monthly_gross;
        metrics(i).Tax_Monthly = monthly_tax;
        metrics(i).Essential_Monthly = mean(monthly_essentials); % Store
average for the plot
        metrics(i).Base_Disp_Monthly = monthly_disposable_scalar;
        metrics(i).Worst_5th_Month = stats.Worst_5th_Percentile_Month;
        metrics(i).Prob_Debt = stats.Prob_of_Debt_Percent;

        % Special handling for Persona 6 (Low Income US) for Exhibit 2
        % Replicate the shock simulation logic to get the raw data vector
        if i == 6
             p6_sim_data =
replicate_shock_simulation_outcome(monthly_disposable_scalar, p.Age, p.Country,
p.Adults, p.Children, 100000);
        end
    end

    % 4. Generate Figure with 4 Subplots
    fig = figure('Name', 'Q1 Analysis Results', 'Color', 'w', 'Position', [50,
50, 1400, 1000]);
    t = tiledlayout(2, 2, 'Padding', 'compact', 'TileSpacing', 'compact');

    % --- Exhibit 1: Proportional Income Burden ---
    nexttile;
    % Data Calculation: [Tax, Essentials, Disposable]
    % We stack these three components.
    % Note: Base_Disp is net - essentials.
    % Gross = Tax + Net = Tax + (Essentials + Base_Disp) ideally.
    % So stack_data rows should sum roughly to Gross (or very close).
    stack_data = [ [metrics.Tax_Monthly]', [metrics.Essential_Monthly]',
[metrics.Base_Disp_Monthly]' ];

    % Normalize to percentage of the SUM of these components (which creates a
clean 100% bar)
    % This handles cases where Base_Disp was clamped to 0 but debt isn't shown
in the stack itself.
    bar_data = stack_data ./ sum(stack_data, 2) * 100;

    b = bar(bar_data, 'stacked');
    b(1).FaceColor = [0.6 0.6 0.6]; % Gray for Tax
    b(2).FaceColor = [0.8 0.2 0.2]; % Deep Red for Essentials
```

```
    b(3).FaceColor = [0.2 0.6 0.2]; % Green for Disposable

    set(gca, 'XTickLabel', {metrics.Label}, 'XTickLabelRotation', 45,
'FontSize', 10);
    ylabel('Percentage of Income (%)', 'FontSize', 11, 'FontWeight', 'bold');
    title('Exhibit 1: Proportional Income Burden', 'FontSize', 12,
'FontWeight', 'bold');
    legend({'Taxes', 'Essentials', 'Disposable Income'}, 'Location',
'eastoutside');
    yline(50, 'k--', 'LineWidth', 1.5, 'Label', '50% Threshold');
    ylim([0 100]);
    grid on;

    % --- Exhibit 2: Fat Tail Risk (Persona 6) ---
    nexttile;
    hold on;
    % Kernel Density Estimate
    [f, xi] = ksdensity(p6_sim_data);

    % Shade the "Debt Zone" (x < 0)
    debt_indices = xi < 0;
    if any(debt_indices)
        % Fill area under curve for x < 0
        fill([xi(debt_indices) 0], [f(debt_indices) 0], [1 0.8 0.8],
'EdgeColor', 'none', 'DisplayName', 'Debt Zone');
    end

    % Plot the density line
    plot(xi, f, 'LineWidth', 2, 'Color', 'k', 'DisplayName', 'Income
Distribution');

    % Add vertical line at x=0
    xline(0, 'r--', 'LineWidth', 2, 'Label', 'Zero Balance');

    xlabel('Monthly Disposable Income ($)', 'FontSize', 11, 'FontWeight',
'bold');
    ylabel('Probability Density', 'FontSize', 11, 'FontWeight', 'bold');
    title({'Exhibit 2: Fat Tail Risk Distribution', '(Persona 6: Low Income
US)'}, 'FontSize', 12, 'FontWeight', 'bold');
    legend('Location', 'northeast');
    grid on;
    xlim([min(xi) max(xi)*1.1]);
    box on;
    hold off;

    % --- Exhibit 3: Vulnerability (Grouped Bar) ---
    nexttile;
    % Grouped: [Base Monthly, Worst 5th %]
    vuln_data = [ [metrics.Base_Disp_Monthly]', [metrics.Worst_5th_Month]' ];
```

```
    b3 = bar(vuln_data, 'grouped');
    b3(1).FaceColor = [0.2 0.4 0.8]; % Blue
    b3(2).FaceColor = [0.8 0.4 0.2]; % Orange/Red

    set(gca, 'XTickLabel', {metrics.Label}, 'XTickLabelRotation', 45,
'FontSize', 10);
    ylabel('Monthly Income ($)', 'FontSize', 11, 'FontWeight', 'bold');
    title('Exhibit 3: Vulnerability by Demographic', 'FontSize', 12,
'FontWeight', 'bold');
    legend({'Base Monthly Income', 'Worst 5th Percentile Month'}, 'Location',
'best');
    yline(0, 'k-', 'LineWidth', 1.5);
    grid on;

    % --- Exhibit 4: Probability of Ruin ---
    nexttile;
    % Line chart
    probs = [metrics.Prob_Debt]';
    x = 1:num_personas;

    % Plot line
    plot(x, probs, '-o', 'LineWidth', 2, 'Color', 'b', 'MarkerFaceColor', 'b',
'MarkerSize', 6);
    hold on;
    % Highlight P6 with red asterisk
    plot(6, probs(6), 'r*', 'MarkerSize', 15, 'LineWidth', 3);

    set(gca, 'XTick', 1:num_personas, 'XTickLabel', {metrics.Label},
'XTickLabelRotation', 45, 'FontSize', 10);
    ylabel('Probability of Neg. Income (%)', 'FontSize', 11, 'FontWeight',
'bold');
    title('Exhibit 4: Baseline Probability of Ruin', 'FontSize', 12,
'FontWeight', 'bold');
    grid on;
    ylim([0 max(probs)*1.2]); % Give some headroom

    % Add data labels for clarity
    text(x, probs, string(round(probs,1))+"%", 'VerticalAlignment', 'bottom',
'HorizontalAlignment', 'center', 'FontSize', 9);

    hold off;

end

function [simulated_monthly_incomes] =
replicate_shock_simulation_outcome(monthly_disposable_income, age, country,
adults, children, num_sims)
    % Replicates logic of simulate_monthly_shocks to return raw array
```

```
    % Step A: Severity Parameters (Log-Normal Distribution)
    country = upper(string(country)); % Ensure string comparison works
    if strcmpi(country, 'US')
        mu = 7.6;
        sigma = 0.8;
    elseif strcmpi(country, 'UK')
        mu = 6.48;
        sigma = 0.75;
    else
        % Default fallback if unknown (should rely on caller to be correct)
         mu = 7.6; sigma=0.8;
    end

    % Step B: Frequency Parameter (Poisson Distribution)
    base_lambda_monthly = 0.0763;

    if age > 40
        age_mod = 1.2;
    else
        age_mod = 1.0;
    end

    size_mod = 1.0 + 0.2 * ((adults + children) - 1);
    lambda_monthly = base_lambda_monthly * age_mod * size_mod;

    % Step C: Monte Carlo Simulation
    simulated_monthly_incomes = zeros(num_sims, 1);

    % Generate number of events
    N = poissrnd(lambda_monthly, num_sims, 1);

    for i = 1:num_sims
        if N(i) == 0
            shock_cost = 0;
        else
            % Draw N(i) random samples
            shock_cost = sum(lognrnd(mu, sigma, N(i), 1));
        end
        simulated_monthly_incomes(i) = monthly_disposable_income - shock_cost;
    end
end
```

Q2 Project Codebase

Gambler.m

```
classdef Gambler < handle    properties
        Age
        Wealth
        Gender
        SavedIncome
        BaseRiskTolerance
        CurrentRiskTolerance
        ChasingCoefficient
        NetProfit
        BetHistory
        ProfitHistory
        max_bet_per_month
    end

    methods
        function self = Gambler(age, gender, wealth, savedIncome, chasingCoeff,
max_bet_per_month)
            if nargin < 5
                chasingCoeff = 1.2; % Default chasing coefficient > 1
            end
            self.max_bet_per_month = max_bet_per_month;
            self.Wealth = wealth;
            self.Age = age;
            self.Gender = gender;
            self.SavedIncome = savedIncome;
            self.ChasingCoefficient = chasingCoeff;
            self.NetProfit = 0;
            self.BetHistory = [];
            self.ProfitHistory = [];

            % Formula provided: 3.068 + 0.987(Age) - 0.037(Age^2)
            score = 54.0832 + 1.5402 * self.Age -0.0329 * (self.Age^2);
            % Convert the resulting score to a probability / risk measurement
[0, 1]
            if self.Gender == "Male"
                score = 53.5213 + 1.9007 * self.Age -0.0405 * (self.Age^2);
            end
            score = score / 100;
            self.BaseRiskTolerance = max(0.01, min(1, score)); % minimum floor
of 1%
            self.CurrentRiskTolerance = self.BaseRiskTolerance;
        end
```

```
        function [risk_level, win_prob, payout_multiplier] =
determine_bet_profile(self)
            % Incorporate risk tolerance and disposable income into a risk
profile
            % Average US disposable income ~40,000 for relative scaling
            income_factor = max(2.0, min(0.5, self.SavedIncome / 40000));
            effective_risk = self.CurrentRiskTolerance * income_factor;

            % Map to defined profiles based on empirical thresholds
            if effective_risk < 0.08
                % Low Risk (Straight Bets)
                risk_level = "Low";
                win_prob = 0.50;
                payout_multiplier = 0.909;
            elseif effective_risk < 0.15
                % Average Risk (Mixed Portfolio)
                risk_level = "Average";
                win_prob = 0.45;
                payout_multiplier = 1.0;
            else
                % High Risk (Parlays)
                risk_level = "High";
                win_prob = 0.10;
                payout_multiplier = 7.0;
            end
        end

        function profit = place_bet(self, bet_amount)
            [~, win_prob, payout_multiplier] = self.determine_bet_profile();

            % Simulate outcome
            is_win = rand() < win_prob;

            if is_win
                % Gambler wins
                profit = bet_amount * payout_multiplier;
                self.NetProfit = self.NetProfit + profit;

                % Reset risk tolerance back to original value (Markov Chain
restart)
                self.CurrentRiskTolerance = self.BaseRiskTolerance;
            else
                % Gambler loses
                profit = -bet_amount;
                self.NetProfit = self.NetProfit + profit;

                % Chase losses: multiply risk tolerance by a chasing
coefficient > 1
                self.CurrentRiskTolerance = min(1.0, self.CurrentRiskTolerance
```

```
* self.ChasingCoefficient);
            end
        end

        function simulate_year(self, bettingFreq)
            self.BetHistory = zeros(12, 1);
            self.ProfitHistory = zeros(12, 1);

            for i = 1:12
                self.Wealth = self.SavedIncome + self.Wealth;

                for f = 1:bettingFreq
                    total_profit = 0;
                    betting = self.Wealth > 0;
                    base_bet_amount = 0.001 * self.SavedIncome;
                    max_bet = self.SavedIncome * 0.05;

                    while betting
                        bet_amount = base_bet_amount *
(self.CurrentRiskTolerance / self.BaseRiskTolerance);
                        bet_amount = min(bet_amount, max_bet);
                        outcome = self.place_bet(bet_amount);
                        total_profit = total_profit + outcome;
                        self.Wealth = self.Wealth + outcome;
                        out = rand();
                        cond = true;
                        if self.max_bet_per_month
                            cond = total_profit > (-max_bet);
                        else
                            cond = total_profit > (-max_bet/bettingFreq);
                        end

                        betting = self.CurrentRiskTolerance > out && cond &&
self.Wealth > 0;
                    end

                    self.NetProfit = self.NetProfit + total_profit;
                end
                % Bet size scales with risk tolerance, capped at 5% of
disposable income



                self.ProfitHistory(i) = self.NetProfit;
            end
        end
    end
end
```

AgeRisk.m

```
% Enter midpoints of age bins
age = [10.5 12.5 14.5 16.5 18.5 20.5 22.5 24.5 26.5 28.5];

% Enter the corresponding Table 3 BART means here
bart_mean_f = [ 65.73, 68.66, 70.00, 71.58, 70.68, 72.36, 70.89, 71.34, 72.18,
71.68 ];   % <-- paste actual percentages
bart_mean = [68.70, 71.37, 71.93, 74.20, 75.70, 75.52, 75.85, 74.27, 75.53,
75.41];
% Fit quadratic
p = polyfit(age, bart_mean, 2);

% Quadratic equation:
% y = p(1)*Age^2 + p(2)*Age + p(3)

disp(p)
```

MAIN_Q2.m

```
% MAIN_Q2.m
% Agentic Markov Chain Simulation for Online Gambling Losses
clear; clc; close all;

% We define three different gamblers to simulate based on demographics
% Gambler(Age, Gender, Country, DisposableIncome, ChasingCoefficient)

% Simulation configuration
% Assume placing a bet every other day on average over one year (180 bets)

bet_count = 5; % number of bets per month
num_simulations = 100; % Number of simulations to average over
max_bet_per_month = false; % if false people will be willing to be only
0.05(savedIncome/bet_count) else 0.05(savedIncome)
rng(42); % Set random seed for reproducibility

% Preallocate arrays for average profit history
g1_avg_profit_history = zeros(12, 1);
g2_avg_profit_history = zeros(12, 1);
g3_avg_profit_history = zeros(12, 1);

g1_avg_net_profit = 0;
g2_avg_net_profit = 0;
g3_avg_net_profit = 0;

% Run the year-long simulation multiple times for each agentfor i =
```

```
1:num_simulations
    % Group 1: Young Adult, high base risk tolerance, lower disposable income
    g1 = Gambler(22, "Male", 0, 35000, 0.9, max_bet_per_month);

    % Group 2: Middle Aged, lower base risk tolerance, higher disposable income
    g2 = Gambler(100, "Female", 100000, 100000, 1.5, max_bet_per_month);

    % Group 3: Very Young Adult, high base risk tolerance, very low disposable
income
    g3 = Gambler(19, "Male", 0, 2000, 1.01, max_bet_per_month);

    g1.simulate_year(bet_count);
    g2.simulate_year(bet_count);
    g3.simulate_year(bet_count);

    g1_avg_profit_history = g1_avg_profit_history + g1.ProfitHistory;
    g2_avg_profit_history = g2_avg_profit_history + g2.ProfitHistory;
    g3_avg_profit_history = g3_avg_profit_history + g3.ProfitHistory;

    g1_avg_net_profit = g1_avg_net_profit + g1.NetProfit;
    g2_avg_net_profit = g2_avg_net_profit + g2.NetProfit;
    g3_avg_net_profit = g3_avg_net_profit + g3.NetProfit;
end

% Calculate averages
g1_avg_profit_history = g1_avg_profit_history / num_simulations;
g2_avg_profit_history = g2_avg_profit_history / num_simulations;
g3_avg_profit_history = g3_avg_profit_history / num_simulations;

g1_avg_net_profit = g1_avg_net_profit / num_simulations;
g2_avg_net_profit = g2_avg_net_profit / num_simulations;
g3_avg_net_profit = g3_avg_net_profit / num_simulations;

% ---------------------------------------------------------
% Plot the average cumulative profit/loss history
% ---------------------------------------------------------
figure('Name', 'Yearly Sports Betting Simulation (Averaged)', 'Color', 'w');
hold on;
plot(1:12, g1_avg_profit_history, 'LineWidth', 2, 'DisplayName', 'Agent 1: 22yo
Male ($35k Income)');
plot(1:12, g2_avg_profit_history, 'LineWidth', 2, 'DisplayName', 'Agent 2: 45yo
Female ($65k Income)');
plot(1:12, g3_avg_profit_history, 'LineWidth', 2, 'DisplayName', 'Agent 3: 19yo
Male ($20k Income)');
hold off;

title(sprintf('Average Cumulative Profit/Loss Over One Year (%d Simulations)',
num_simulations));
xlabel('Months');
```

```
ylabel('Average Cumulative Profit (USD)');
legend('Location', 'best');
grid on;

% ------------------------------------------------------------
% Display end-of-year summaries
% ------------------------------------------------------------
fprintf('--- Average Simulation Results (%d Simulations) ---\n',
num_simulations);
fprintf('Agent 1 (22yo): Average Net Profit = $%.2f, Base Risk = %.2f%%\n',
g1_avg_net_profit, g1.BaseRiskTolerance * 100);
fprintf('Agent 2 (45yo): Average Net Profit = $%.2f, Base Risk = %.2f%%\n',
g2_avg_net_profit, g2.BaseRiskTolerance * 100);
fprintf('Agent 3 (19yo): Average Net Profit = $%.2f, Base Risk = %.2f%%\n',
g3_avg_net_profit, g3.BaseRiskTolerance * 100);
```

sensitivity_analysis_Q2.m

```
% sensitivity_analysis_Q2.m
% One-At-A-Time (OAT) Sensitivity Analysis for Gambler Simulation

clear; clc; close all;

num_simulations = 500; % High enough for stable averages without taking too
long

% --- Baseline values ---
base_age = 25;
base_income = 40000;
base_chase = 1.1;
base_bet_count = 5;

% --- Parameter ranges [low, high] ---
% We define sensible low and high ranges for each parameter to see the effect
% of changing each parameter from its baseline.
range_age = [18, 45];
range_income = [20000, 60000];
range_chase = [0.9, 1.3];
range_bet = [2, 10];

fprintf('Running sensitivity analysis (%d simulations per scenario)...\n',
num_simulations);

% --- Calculate Baseline ---
fprintf('Calculating baseline...\n');
base_profit = run_sim_q2(base_age, base_income, base_chase, base_bet_count,
num_simulations);
```

```
fprintf('Baseline Average Net Profit: $%.2f\n', base_profit);

% --- Evaluate extremes for Age ---
fprintf('Evaluating Age = [%d, %d]...\n', range_age(1), range_age(2));
prof_age_low = run_sim_q2(range_age(1), base_income, base_chase,
base_bet_count, num_simulations);
prof_age_high = run_sim_q2(range_age(2), base_income, base_chase,
base_bet_count, num_simulations);

% --- Evaluate extremes for Income ---
fprintf('Evaluating Saved Income = [$%d, $%d]...\n', range_income(1),
range_income(2));
prof_inc_low = run_sim_q2(base_age, range_income(1), base_chase,
base_bet_count, num_simulations);
prof_inc_high = run_sim_q2(base_age, range_income(2), base_chase,
base_bet_count, num_simulations);

% --- Evaluate extremes for Chasing Coefficient ---
fprintf('Evaluating Chasing Coefficient = [%.2f, %.2f]...\n', range_chase(1),
range_chase(2));
prof_chase_low = run_sim_q2(base_age, base_income, range_chase(1),
base_bet_count, num_simulations);
prof_chase_high = run_sim_q2(base_age, base_income, range_chase(2),
base_bet_count, num_simulations);

% --- Evaluate extremes for Bet Count ---
fprintf('Evaluating Bet Count = [%d, %d]...\n', range_bet(1), range_bet(2));
prof_bet_low = run_sim_q2(base_age, base_income, base_chase, range_bet(1),
num_simulations);
prof_bet_high = run_sim_q2(base_age, base_income, base_chase, range_bet(2),
num_simulations);

% --- Calculate swings and rank ---
% Collecting results
params = {'Age (18 - 45)', 'Saved Income ($20k - $60k)', 'Chasing Coeff (0.9 -
1.3)', 'Bet Count (2 - 10)'};
low_vals = [prof_age_low, prof_inc_low, prof_chase_low, prof_bet_low];
high_vals = [prof_age_high, prof_inc_high, prof_chase_high, prof_bet_high];

% A swing is the absolute difference between the parameter's minimum and
maximum outcomes
swings = abs(high_vals - low_vals);

% Sort by swing magnitude to rank importance
[sorted_swings, sort_idx] = sort(swings, 'ascend'); % Ascend for barh plot
(largest swing at top of chart)
sorted_params = params(sort_idx);
sorted_low = low_vals(sort_idx);
sorted_high = high_vals(sort_idx);
```

```
% Display descending ranking back to console
fprintf('\n--- Sensitivity Ranking (Most to Least Important) ---\n');for i =
length(sorted_swings):-1:1
    fprintf('%d. %s (Swing: $%.2f)\n', length(sorted_swings) - i + 1,
sorted_params{i}, sorted_swings(i));
end

% --- Plot Tornado Chart ---
figure('Name', 'Sensitivity Analysis Tornado Chart', 'Color', 'w');
hold on;

% Draw tornado bars
for i = 1:length(sorted_swings)
    % We compute distance from baseline for both low and high results
    % Draw the lower bound block
    plot([sorted_low(i), base_profit], [i, i], 'LineWidth', 20, 'Color',
[0.8500 0.3250 0.0980]); % Orange
    % Draw the higher bound block
    plot([base_profit, sorted_high(i)], [i, i], 'LineWidth', 20, 'Color', [0
0.4470 0.7410]); % Blue
end

% Draw baseline vertical line
plot([base_profit, base_profit], [0, length(sorted_swings)+1], 'k--',
'LineWidth', 1.5, 'HandleVisibility', 'off');

% Format Chart
set(gca, 'YTick', 1:length(sorted_params), 'YTickLabel', sorted_params,
'FontSize', 10, 'XColor', 'k', 'YColor', 'k');
ylim([0.5, length(sorted_params) + 0.5]);
xlabel('Average Annual Net Profit (USD)', 'FontSize', 12, 'FontWeight', 'bold',
'Color', 'k');
title('Sensitivity Analysis of Annual Gambling Profit', 'FontSize', 14,
'Color', 'k');

% Add custom legends for colored bars
% Mock scatter plots to produce legend entries with correct colors
h1 = plot(NaN, NaN, 'LineWidth', 8, 'Color', [0.8500 0.3250 0.0980]);
h2 = plot(NaN, NaN, 'LineWidth', 8, 'Color', [0 0.4470 0.7410]);
legend([h1, h2], {'Result of Lower Parameter Bound', 'Result of Upper Parameter
Bound'}, 'Location', 'best', 'Box', 'off');

grid on;
set(gca, 'GridColor', 'w');
hold off;

% --- Helper Functions ---
function avg_profit = run_sim_q2(age, savedIncome, chasingCoeff, bet_count,
```

```
num_sims)
    % To keep results purely deterministic for the analysis and comparisons:
    rng(42);

    total_profit = 0;

    for i = 1:num_sims
        % We fix remaining attributes: "US", "Male", 0 starting wealth,
max_bet_per_month = false
        g = Gambler(age, "Male", "US", 0, savedIncome, chasingCoeff, false);
        g.simulate_year(bet_count);
        total_profit = total_profit + g.NetProfit;
    end

    avg_profit = total_profit / num_sims;
end
```

Q3 Project Codebase

gap_dynamics.m

```
function dDelta = gap_dynamics(t, Delta, q, a, w, b, r, id)
    % gap_dynamics defines the differential equations for the financial gaps.
    % State variables:
    % Delta(1) = Delta_A (Savings gap)
    % Delta(2) = Delta_W (Wealth gap)
    % Delta(3) = Delta_B (Debt gap)

    dDelta = zeros(3, 1);

    % Savings gap grows from cash drained now
    dDelta(1) = a * q;

    % Wealth gap grows from missed investing plus lost compounding
    dDelta(2) = r * Delta(2) + w * q;

    % Debt gap grows from new borrowing plus interest
    dDelta(3) = id * Delta(3) + b * q;
end
```

calculate_RBM.m

```
function RBM = calculate_RBM(months, g_monthly, s_monthly, c_monthly, a, w, b,
r_annual, id_annual, M, C_rep)
    % calculate_RBM calculates the Recovery Burden Months (RBM)
    %
    % Inputs:
    %   months     - simulation time in months
    %   g_monthly  - gambling loss
    %   s_monthly  - stress/leakage cost
    %   c_monthly  - safe cushion
    %   a, w, b    - shares allocated to savings, wealth, debt (a+w+b=1)
    %   r_annual   - annual investment return
    %   id_annual  - annual debt interest rate
    %   M          - milestone-delay cost
    %   C_rep      - normal monthly saving capacity
    %
    % Output:
    %   RBM - Recovery Burden Months

    % Monthly rates
    r = r_annual / 12;
    id = id_annual / 12;

    % Harmless shortfall
    q = max(0, g_monthly + s_monthly - c_monthly);

    % ODE parameters
    tspan = [0, months];
    y0 = [0; 0; 0]; % Initial gaps: A, W, B

    % Run ODE solver
    [~, y] = ode45(@(t, y) gap_dynamics(t, y, q, a, w, b, r, id), tspan, y0);

    % Final values
    delta_A = y(end, 1);
    delta_W = y(end, 2);
    delta_B = y(end, 3);

    % Total repair deficit
    R = delta_A + delta_W + delta_B + M;

    % Recovery Burden Months
    RBM = R / C_rep;
end
```

MAIN_Q3.m

```
% Quantifying the Financial Impact of Online Gambling
% Differential dynamics model based on the "Recovery Burden Months" metric

% Define baseline parameters for an individual over time
simulation_months = 60;   % 5 years

% The monthly harmful shortfall is q(t) = max(0, g(t) + s(t) - c(t))
g_monthly = 200;  % Gambling loss per month ($)
s_monthly = 50;   % Stress/leakage cost per month ($)
c_monthly = 100;  % Safe cushion ($)

% Allocation of shortfall across savings, investing, and debt (must sum to 1)
a = 0.5; % 50% from liquid savings
w = 0.3; % 30% from missed investing
b = 0.2; % 20% to new debt

% Interest/Return rates (annual)
r_annual  = 0.07; % 7% expected annual investment return
id_annual = 0.22; % 22% expected annual debt interest rate

% Milestone delay cost (e.g. extra rent, delayed emergency fund)
M = 1000; % ($)

% Normal monthly saving capacity
C_rep = 500; % ($)

% Calculate the Recovery Burden Months
% This runs the counterfactual differential tracking the gaps over time.
RBM = calculate_RBM(simulation_months, g_monthly, s_monthly, c_monthly, a, w,
b, r_annual, id_annual, M, C_rep);

% Output exactly what was asked in "One simple output"
fprintf('%.2f\n', RBM);
```

sensitivity_analysis_Q3.m

```
% sensitivity_analysis_Q3.m
% Performs sensitivity analysis on the Recovery Burden Months (RBM) metric
% by varying key parameters and visualizing their impact using a Tornado
diagram.

%% Baseline Parameters
base_months = 60;
base_g_monthly = 200;
```

```
base_s_monthly = 50;
base_c_monthly = 100;
base_a = 0.5;
base_w = 0.3;
base_b = 0.2;
base_r_annual = 0.07;
base_id_annual = 0.22;
base_M = 1000;
base_C_rep = 500;

% Calculate baseline RBMbaseline_RBM = calculate_RBM(base_months,
base_g_monthly, base_s_monthly, ...
                            base_c_monthly, base_a, base_w, base_b, ...
                            base_r_annual, base_id_annual, base_M,
base_C_rep);

%% Sensitivity Analysis Setup
% We will vary each continuous parameter by +/- 20%
variation = 0.20;

% Parameters to test
param_names = {'g_monthly', 's_monthly', 'c_monthly', ...
               'r_annual', 'id_annual', 'M', 'C_rep'};

param_labels = {'Gambling Loss (g\_monthly)', ...
                'Stress/Leakage (s\_monthly)', ...
                'Safe Cushion (c\_monthly)', ...
                'Investment Return (r\_annual)', ...
                'Debt Interest Rate (id\_annual)', ...
                'Milestone Delay Cost (M)', ...
                'Monthly Saving Capacity (C\_rep)'};

num_params = length(param_names);

% Store results
rbm_low = zeros(num_params, 1);
rbm_high = zeros(num_params, 1);
sensitivity_spread = zeros(num_params, 1);

for i = 1:num_params
    % Reset to baseline
    p = struct('g_monthly', base_g_monthly, 's_monthly', base_s_monthly, ...
               'c_monthly', base_c_monthly, 'r_annual', base_r_annual, ...
               'id_annual', base_id_annual, 'M', base_M, 'C_rep', base_C_rep);

    % Current parameter baseline value
    base_val = p.(param_names{i});

    % Low case (-20%)
```

```
    p.(param_names{i}) = base_val * (1 - variation);
    rbm_low(i) = calculate_RBM(base_months, p.g_monthly, p.s_monthly, ...
                               p.c_monthly, base_a, base_w, base_b, ...
                               p.r_annual, p.id_annual, p.M, p.C_rep);

    % High case (+20%)
    p.(param_names{i}) = base_val * (1 + variation);
    rbm_high(i) = calculate_RBM(base_months, p.g_monthly, p.s_monthly, ...
                                p.c_monthly, base_a, base_w, base_b, ...
                                p.r_annual, p.id_annual, p.M, p.C_rep);

    % Calculate spread (absolute difference from baseline)
    % We use the actual range for the tornado diagram sorting
    sensitivity_spread(i) = abs(rbm_high(i) - rbm_low(i));
end

%% Sort Parameters by Importance (Spread)
[sorted_spread, sort_idx] = sort(sensitivity_spread, 'ascend'); % Ascend for
horizontal bar chart (widest at top)

sorted_labels = param_labels(sort_idx);
sorted_low = rbm_low(sort_idx);
sorted_high = rbm_high(sort_idx);

%% Display Ranking Textually
fprintf('Parameter Sensitivity Ranking (from MOST to LEAST important):\n');
fprintf('-------------------------------------------------------------\n');
% Print in descending order
for i = num_params:-1:1
    idx = sort_idx(i);
    fprintf('%d. %-35s - Impact Spread: %6.2f months\n', ...
        num_params - i + 1, param_labels{idx}, sensitivity_spread(idx));
end

%% Plot Tornado Diagram
figure('Name', 'Sensitivity Analysis: Tornado Diagram', 'Position', [100, 100,
800, 500]);
hold on;

% Create the bars
for i = 1:num_params
    % Instead of traditional barh, using stacked bars where
    % deviations from baseline are plotted right and left.
    change_high = sorted_high(i) - baseline_RBM;
    change_low = sorted_low(i) - baseline_RBM;

    % High value (+20%)
    if change_high >= 0
        h1 = barh(i, change_high, 'FaceColor', [0.8 0.2 0.2], 'EdgeColor', 'k',
```

```
'BaseValue', 0, 'BarWidth', 0.6);
    else
        h1 = barh(i, change_high, 'FaceColor', [0.2 0.6 0.2], 'EdgeColor', 'k',
'BaseValue', 0, 'BarWidth', 0.6);
    end

    % Low value (-20%)
    if change_low < 0
        h2 = barh(i, change_low, 'FaceColor', [0.2 0.6 0.2], 'EdgeColor', 'k',
'BaseValue', 0, 'BarWidth', 0.6);
    else
        h2 = barh(i, change_low, 'FaceColor', [0.8 0.2 0.2], 'EdgeColor', 'k',
'BaseValue', 0, 'BarWidth', 0.6);
    end
end

% Plot a vertical line at the baseline RBM change (which is 0)
plot([0 0], [0 num_params+1], 'k--', 'LineWidth', 1.5);

% Formatting
yticks(1:num_params);
yticklabels(sorted_labels);
xlabel('Change in Recovery Burden Months (RBM) from Baseline');
title(sprintf('Tornado Diagram of RBM Sensitivity (\\pm 20%% parameter
variation)\nBaseline RBM = %.2f months', baseline_RBM));
grid on;

% Adjust limits
ylim([0.5 num_params+0.5]);
ax = gca;
ax.XAxis.Exponent = 0; % Prevent scientific notation

% Create legend entries
h_red = plot(NaN,NaN,'s','MarkerFaceColor',[0.8 0.2 0.2],'MarkerEdgeColor','k',
'MarkerSize', 10);
h_green = plot(NaN,NaN,'s','MarkerFaceColor',[0.2 0.6
0.2],'MarkerEdgeColor','k', 'MarkerSize', 10);
legend([h_red, h_green], {'Increases RBM (Worse)', 'Decreases RBM (Better)'},
'Location', 'best');

hold off;
```