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

1 Executive Summary

This paper studies online sports gambling in three connected steps [1, 2]. First, we estimate how much income is left after taxes and basic bills. Second, we estimate how much an online sports bettor is likely to lose over one year. Third, we translate those yearly losses into household-level financial harm.

Problem 1 shows that the key issue is not income alone. Age, country, and region also matter because taxes and everyday costs are different across groups. In both the United States and the United Kingdom, basic spending rises as income rises, but it rises more slowly than income. That means higher salaries usually create more budget room, while lower salaries can leave little room at all. In our examples, several lower-income U.S. profiles already have almost no cushion, or even a shortfall, before gambling begins.

Problem 2 asks who is most exposed to yearly gambling losses. We separate two simple questions: how likely is someone to be an active online bettor, and how much is that bettor likely to lose over a year [12, 13, 14]. The model shows that losses are not spread evenly across the public. At moderate risk, the largest population-wide losses in our grid occur for men ages 35 to 49 with at least a bachelor's degree, at about 65 USD per adult in the United States and 28 GBP per adult in the United Kingdom. Among our representative cases, the highest-risk U.S. profile loses about 403 USD per year if he is an active bettor, while the lowest-risk older U.K. example loses about 44 GBP as an active bettor and only about 2 GBP after the lower betting rate is taken into account.

Problem 3 translates those yearly losses into plain-language financial harm. We report how many months of budget cushion are used up, how likely a household is to be pushed into debt in a bad year, and how large the loss becomes over ten years. The most vulnerable profile in our study is a high-risk U.S. South male age 25 with no college degree and a 35,000 USD salary. He has no real cushion left before gambling, a debt-entry probability of 100%, and a ten-year wealth gap of about 8,186 USD. Raising that same profile to 45,000 USD helps, but it still leaves a strained result, with about 1.4 months of cushion absorbed and a ten-year gap of about 4,840 USD. By contrast, a high-risk U.K. England male age 40 with a bachelor's degree and a 45,000 GBP salary remains manageable in the short run, although repeated losses still create a ten-year wealth gap of about 3,750 GBP.

The main public concern is not the average adult. It is the smaller set of active bettors whose yearly losses fall on already thin household budgets. Across all three problems, the same groups remain the most exposed even after sensitivity testing.

Back to contents

Contents

Back to contents

2 Playing With House Money

2.1 Defining the Problem

Problem 1 asks how much money is really left after taxes and necessary expenses for people with different salaries, ages, and locations [1]. In this paper, "playing with house money" means gambling only after regular bills have already been covered, so we define *disposable income* as the part of annual gross salary that remains after taxes and basic living costs are paid. Country and region are included because tax rules and everyday costs differ across places. Using the spending tables in the contest data [2], we build a model that turns salary, age, country, and region into a clear estimate of annual budget room.

2.2 Assumptions

A1. Salary s represents annual gross household income for a household reference person, consistent with the structure of the provided expenditure tables [2].

A2. Essential expenditures include food, housing, utilities, transportation, and healthcare. Categories such as entertainment and recreation are treated as nonessential.

A3. Essential spending grows with income in a smooth way rather than jumping abruptly from one income group to the next. We estimate that smooth relationship from the summary tables in Python [2].

A4. U.S. taxes include federal income tax brackets for 2025 [3] and employee payroll taxes using the Social Security wage base for 2025 [6]. State and local taxes are excluded.

A5. U.K. taxes include Income Tax rates and bands for the 2025 to 2026 tax year [7]. Employee National Insurance uses the 2025 to 2026 thresholds and rates for a standard category A employee [9, 10]. Scotland uses its distinct Income Tax bands in the same tax year [8].

A6. If the model gives $D < 0$, the household does not really have spare money left. That shortfall would have to be covered by savings, borrowing, transfers, or cutting back on spending. In later sections, we therefore replace discretionary funds with $\max(D, 0)$.

2.3 Variables

Table 1 lists the main variables and parameters. Clickable entries in the last column jump to the place where the number, formula, or lookup rule is estimated, assumed, or imported. Spreadsheet-based values link to the raw appendix tables in Appendix A, and bracketed source tags identify the corresponding reference. Age-ordered U.S. values follow the raw-sheet order under 25, 25–34, 35–44, 45–54, 55–64, 65–74, 75+. U.K. age-ordered values follow under 30, 30–49, 50–64, 65–74, over 74. Region and nation orders follow the raw-sheet table order.

2.4 The Model

2.4.1 Disposable income identity

The starting point is simple: money left over equals income minus taxes minus essential spending. In symbols,

$$D(s, a, r) = s - T(s, a, r) - E(s, a, r).$$

So the rest of this section only needs to answer two questions: how much is spent on essentials, and how much is paid in taxes.

2.4.2 Essential expenditure model

The data table gives grouped averages rather than a rule for every exact salary [2]. We therefore use each person's age-region baseline and then scale that baseline smoothly with salary.

Back to contents

Table 1: Variables and parameters in the disposable income model [2]

Symbol	Meaning	Unit	Value used / source
s	annual gross salary	USD or GBP	input
a	age	years	input
r	region	category	input
D	disposable income	USD or GBP	$s - T - E$
T	taxes paid	USD or GBP	2025 U.S./U.K. tax rules
E	essential expenditures	USD or GBP	U.S.: $E_0(s/I_0)^{\alpha_{\mathrm{US}}}$; U.K.: $N_0(s/I_{0,\mathrm{UK}})^{\alpha_{\mathrm{UK}}} + H_0$
$I^{\mathrm{age}}(a)$	average income for the age group containing age a	USD or GBP	48,514; 102,494; 128,285; 141,121; 121,571; 75,460; 56,028 [2]
$E^{\mathrm{age}}(a)$	average essential spending for the age group containing age a	USD	36,977; 56,713; 69,412; 73,230; 64,167; 54,594; 48,174 [2]
$m_I(r)$	U.S. regional income multiplier		1.084; 0.910; 0.879; 1.127 [2]
$m_E(r)$	U.S. regional essential-spending multiplier		1.044; 0.910; 0.898; 1.148 [2]
I_0	reference income used to anchor the fitted power law at the selected age and region or nation	USD or GBP	42,630–159,100; 55,265.6 [2]
E_0	U.S. reference essential spending evaluated at I_0	USD	33,211–84,071 [2]
N	U.K. non-health essentials, defined as food + housing + transport	GBP	$\widehat{N} = 159.94\, I^{0.4072}$ [2]
$N^{\mathrm{age}}(a)$	age-specific U.K. non-health spending from the All column	GBP	15,069.6; 14,601.6; 13,790.4; 10,561.2; 8,424.0 [2]
N_0	U.K. non-health reference essential spending evaluated at the selected age and nation	GBP	7,440–15,355 [2]
$H_0(a)$	U.K. age-specific health baseline added after fitting non-health essentials	GBP	197.6; 348.4; 504.4; 592.8; 546.0 [2]
$m_N(r)$	U.K. nation-specific non-health multiplier		1.019; 0.914; 0.883; 0.962; 1.000 [2]
$I_{0,\mathrm{UK}}$	mean annual U.K. income across the five reported income groups	GBP	55,265.6 [2]
$\beta_{\mathrm{US},0}$	intercept in the U.S. log linear fit		4.6961 [2]
$\beta_{\mathrm{UK},0}$	intercept in the U.K. log linear fit		5.0748 [2]
α_{US}	slope in the U.S. log linear fit		0.5473 [2]
α_{UK}	slope in the U.K. log linear fit		0.4072 [2]
d_{US}	U.S. standard deduction	USD	15,750 [4]
b_{SSA}	Social Security wage base	USD	176,100 [6]

For the age group containing age a, let $I^{\mathrm{age}}(a)$ and $E^{\mathrm{age}}(a)$ be the average income and essential spending for that age group. We then use the region columns to define simple adjustment factors

$$m_I(r) = \frac{I^{\mathrm{reg}}(r)}{\overline{I}^{\mathrm{reg}}}, \qquad m_E(r) = \frac{E^{\mathrm{reg}}(r)}{\overline{E}^{\mathrm{reg}}},$$

with regional multiplier values $m_I = (1.084, 0.910, 0.879, 1.127)$ and $m_E = (1.044, 0.910, 0.898, 1.148)$ in the order Northeast, Midwest, South, West. Summing food, housing, utilities, transportation, and healthcare within each U.S. age column gives

$$E^{\mathrm{age}} = (36{,}977,\ 56{,}713,\ 69{,}412,\ 73{,}230,\ 64{,}167,\ 54{,}594,\ 48{,}174),$$

again in the order under 25, 25–34, 35–44, 45–54, 55–64, 65–74, 75+. where $\overline{I}^{\mathrm{reg}}$ and $\overline{E}^{\mathrm{reg}}$ are the averages

Back to contents

across the four reported U.S. regions. This gives the age-and-region baseline values

$$I_0(a,r) = I^{\text{age}}(a)m_I(r), \qquad E_0(a,r) = E^{\text{age}}(a)m_E(r).$$

Once those anchor points are fixed, we let salary s vary continuously through

$$E_{\text{US}}(s,a,r) = E_0(a,r)\left(\frac{s}{I_0(a,r)}\right)^{\alpha_{\text{US}}}.$$

We use this form because taking logs turns the curve into a straight line:

$$\log E = \beta_{\text{US},0} + \alpha_{\text{US}} \log I + \varepsilon,$$

This keeps predicted spending positive and fits the calibration data without adding extra shape that the table cannot justify.

The U.K. table needs one extra step. Its five income-group columns do not include healthcare spending, so we first fit only the non-health part:

$$N = \text{Food} + \text{Housing} + \text{Transport}.$$

Let $N^{\text{age}}(a)$ be the age-specific non-health spending from the All column, let $H_0(a)$ be the age-specific healthcare spending from the same column, and let

$$m_N(r) = \frac{N^{\text{nation}}(r)}{N^{\text{nation}}(\text{All UK})}$$

be the non-health multiplier for England, Wales, Scotland, or Northern Ireland. If $I_{0,\text{UK}}$ denotes the mean annual income across the five reported income groups, then

$$H_0(a) = 52 \times \text{weekly Health spending in the age block's All column},$$

so the five annual age-based non-health values are

$$N^{\text{age}} = (15{,}069.6,\ 14{,}601.6,\ 13{,}790.4,\ 10{,}561.2,\ 8{,}424.0),$$

and the five annual health values are

$$H_0 = (197.6,\ 348.4,\ 504.4,\ 592.8,\ 546.0),$$

because the raw weekly Health values in Appendix A are 3.8, 6.7, 9.7, 11.4, 10.5, and each one is multiplied by 52. The order is under 30, 30–49, 50–64, 65–74, over 74. Using the nation columns for food, housing, and transport gives

$$m_N = (1.019,\ 0.914,\ 0.883,\ 0.962,\ 1.000)$$

for England, Wales, Scotland, Northern Ireland, and All UK. The five weekly income-group values 272, 544, 850, 1257, and 2391 GBP imply annual values 14,144, 28,288, 44,200, 65,364, and 124,332, whose mean is 55,265.6 GBP.

$$N_0(a,r) = N^{\text{age}}(a)m_N(r), \qquad E_{\text{UK}}(s,a,r) = N_0(a,r)\left(\frac{s}{I_{0,\text{UK}}}\right)^{\alpha_{\text{UK}}} + H_0(a).$$

So the non-health basket is the only part scaled with salary; the Health term is read directly from the age block and then added back. The fitted U.K. line is

$$\log N = \beta_{\text{UK},0} + \alpha_{\text{UK}} \log I + \varepsilon.$$

Back to contents

2.4.3 Tax model

The tax part is more direct than the spending part. For the U.S., we subtract the standard deduction d_{US} from salary, apply the 2025 federal income tax brackets for a single filer, and then add payroll taxes [3, 4]. Payroll taxes include Social Security at 6.2% up to b_{SSA} and Medicare at 1.45% on all wages, with an extra Medicare surtax above the threshold [6, 5].

For the U.K., we apply the 2025 to 2026 Income Tax bands, use Scotland's separate bands when needed, and include employee National Insurance using the published 2025 to 2026 thresholds and rates [7, 8, 9, 10]. For people at or above state pension age, employee National Insurance is set to zero in our simplified model [11].

2.4.4 From fitted model to plotted curves

At this point, the model is ready to plot. Putting the spending rule into the disposable-income equation gives the functions used in the figures:

$$D_{\mathrm{US}}(s,a,r) = s - T_{\mathrm{US}}(s) - E_0(a,r)\left(\frac{s}{I_0(a,r)}\right)^{\alpha_{\mathrm{US}}},$$

$$D_{\mathrm{UK}}(s,a,r) = s - T_{\mathrm{UK}}(s,a,r) - N_0(a,r)\left(\frac{s}{I_{0,\mathrm{UK}}}\right)^{\alpha_{\mathrm{UK}}} - H_0(a).$$

For a fixed age and region, we read the baseline values from the table, compute taxes, and then vary salary s. Figure 2a repeats that process for several U.S. ages at a fixed region, while Table 2 and Figure 2b apply the same formulas to representative profiles.

2.5 Results

2.5.1 Calibration data and fitted functions

Figure 1 shows the actual data points from the spreadsheet together with the fitted curves. In annual local currency units, the fitted relationships are

$$\widehat{\log E}_{\mathrm{US}} = 4.6961 + 0.5473 \log I, \qquad \widehat{E}_{\mathrm{US}} = 109.52\, I^{0.5473},$$

$$\widehat{\log N}_{\mathrm{UK}} = 5.0748 + 0.4072 \log I, \qquad \widehat{N}_{\mathrm{UK}} = 159.94\, I^{0.4072}.$$

The corresponding R^2 values are 0.9087 for the U.S. fit and 0.8209 for the U.K. fit. Both slopes are less than 1, which means essential spending rises with income, but more slowly than income itself.

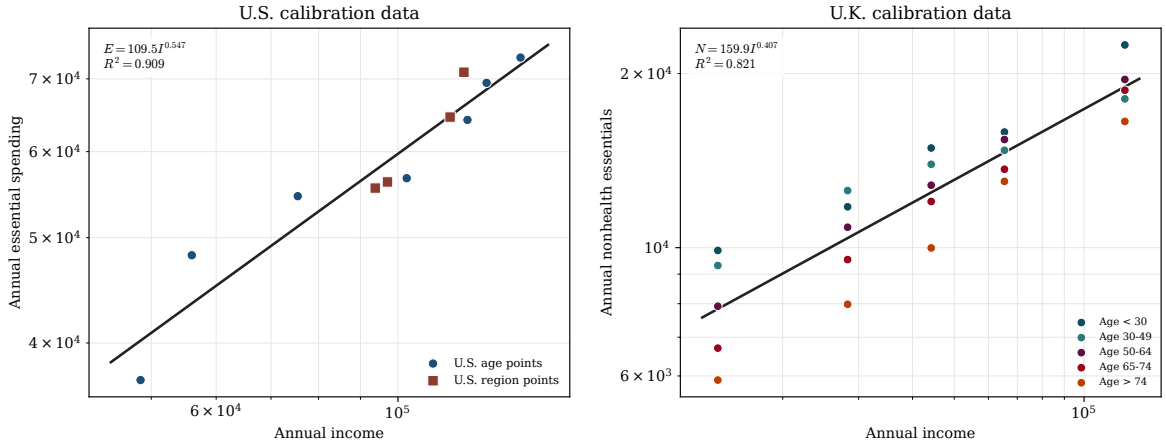


Figure 1: Calibration data points and fitted log linear power laws for the U.S. and U.K. essential spending models [2]

Back to contents

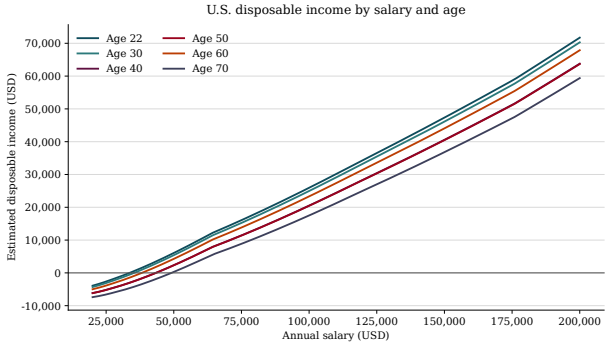
2.5.2 Representative profile estimates

Table 2 gives the required demonstration across varied groups: younger and older households, lower and higher salaries, four U.S. regions, and four U.K. nations [1]. Figure 2a shows how disposable income changes with salary for several U.S. ages, and Figure 2b compares the representative profiles after taxes and essential spending are subtracted. Because U.S. values are in USD and U.K. values are in GBP, later cross-country comparisons focus on ratios and risk measures rather than raw currency amounts.

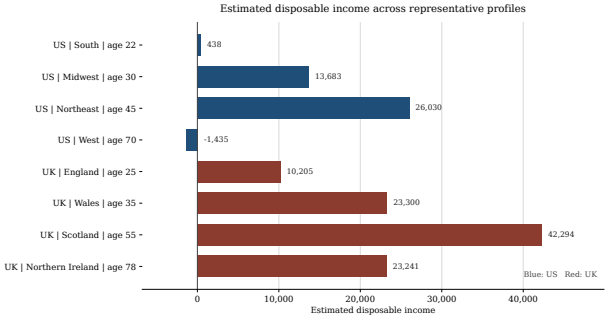
Table 2: Disposable income estimates for representative profiles computed by our model [2]

Profile			Annual Estimate			
Country	Region	Age	Gross Salary	Taxes	Essentials	Disposable Income
US	South	22	35,000	4,749	29,813	438
US	Midwest	30	70,000	12,204	44,113	13,683
US	Northeast	45	120,000	27,047	66,923	26,030
US	West	70	60,000	9,662	51,773	-1,435
UK	England	25	25,000	3,480	11,314	10,205
UK	Wales	35	45,000	9,080	12,620	23,300
UK	Scotland	55	80,000	23,043	14,663	42,294
UK	Northern Ireland	78	35,000	4,486	7,273	23,241

Note. Taxes include modeled national payroll and income taxes. Essentials combine the baseline expenditure basket defined in the model and are reported in local currency units. Negative disposable income is shown with a leading minus sign.



(a) U.S. disposable income as a function of salary for selected ages [2]



(b) Disposable income for representative profiles [2]

2.6 Discussion

Problem 1 mainly measures budget room. Living costs vary sharply by age and location, especially through housing [2], while taxes rise with income and fitted spending rises more slowly than income. As a result, disposable income becomes safely positive only after salary rises far enough above the local baseline. When $D < 0$, regular bills already exceed income, so later sections replace usable gambling cushion with $\max(D, 0)$.

2.7 Sensitivity Analysis

To check whether the answer depends too heavily on a few uncertain inputs, we changed the two least certain spending parameters by as much as 10% and reran the model 1000 times for each representative profile. For trial m,

$$\alpha^{(m)} = \alpha\big(1 + \eta_1^{(m)}\big), \qquad B^{(m)} = B\big(1 + \eta_2^{(m)}\big), \qquad \eta_1^{(m)}, \eta_2^{(m)} \sim \mathrm{Unif}(-0.1, 0.1),$$

where B means E_0 in the U.S. and N_0 in the U.K. After each random change, we recomputed disposable income and measured how much the answer moved as a percentage of salary.

Back to contents

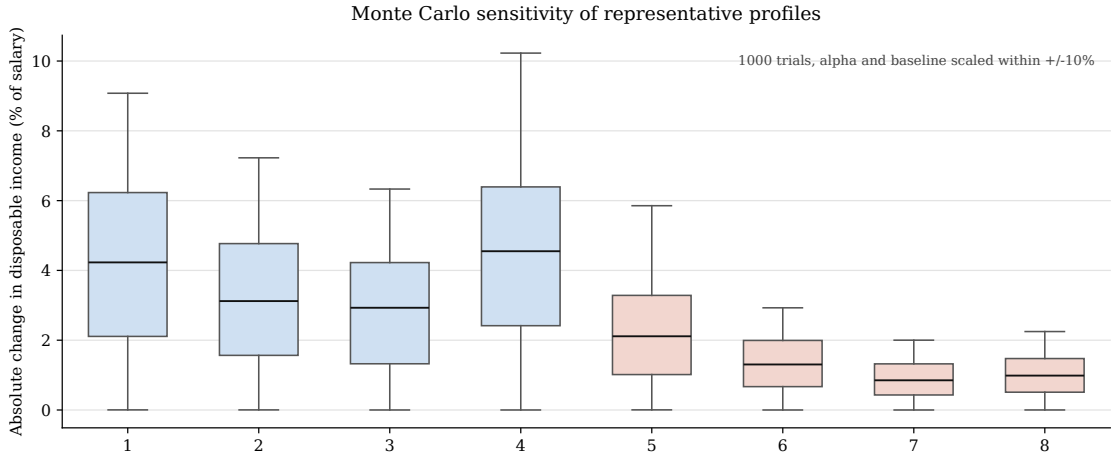


Figure 3: Monte Carlo sensitivity of representative disposable income estimates under 10% parameter jitter [2]

Figure 3 shows that the model reacts more strongly for U.S. profiles with very little budget cushion than for the U.K. examples in Table 2. Averaging across the four representative U.S. profiles, the median absolute change is 3.71% of salary; for the four representative U.K. profiles, it is 1.31%. The most sensitive cases are the South age 22 profile and the West age 70 profile, both of which already sit close to zero disposable income. Their answers switch between positive and negative disposable income in 41.5% and 36.2% of the 1000 runs. By contrast, none of the four representative U.K. profiles changes sign. This matches the derivative

$$\frac{\partial E}{\partial \alpha} = E(s, a, r) \log\left(\frac{s}{I_0}\right),$$

because the model becomes more fragile when salary is far from the baseline level or when the household starts with almost no room in the budget.

2.8 Strengths and Weaknesses

The main strength of the model is transparency. Once salary, age, country, and region are given, the same steps produce the disposable-income estimate, and every step can be traced back to a table or tax rule [2]. The main weakness is that the inputs are broad averages. They miss household size, local price variation, and U.S. state taxes, so the result should be read as a demographic baseline rather than a personalized budget calculator.

2.8.1 Model refinement

If more time and data were available, the first improvement would be to replace group averages with more detailed household data, especially for housing and healthcare. The second would be to add U.S. state taxes, household size, and local housing-cost measures. Those changes would keep the same model structure but would reduce the largest errors caused by broad averaging.

3 Know the Spread

3.1 Defining the Problem

Problem 2 asks who is most exposed to one-year online sports-gambling losses [1]. The spreadsheet does not track individual bettors over time; it only gives broad survey percentages by gender, age, and education, plus country-level market totals [2]. So we build a simple one-year loss model with two outputs:

$$Y_{\text{cond}}(c, d, \rho), \qquad Y(c, d, \rho) = p_{\text{act}}(c, d)\, Y_{\text{cond}}(c, d, \rho).$$

Back to contents

Here c is country, d is the demographic profile, and ρ is risk tolerance. Y_{cond} is the expected one-year loss for an active bettor, while Y averages that loss over the full demographic group after accounting for how many people in that group actually bet. Negative values mean expected losses.

3.2 Assumptions

A1. The separate effects of gender, age, and education can be combined into one overall adjustment without letting any single category dominate. This is necessary because the survey reports those categories separately rather than as one full cross-tabulated table [2].

A2. Because similar U.K. demographic breakdowns are not available in the spreadsheet, we borrow the *relative* demographic pattern from the U.S. survey but keep a separate national baseline for the U.K. [2].

A3. For the open-ended wager bucket 500+, a midpoint of 1000 local currency units is used. This keeps the bucket conservative while still reflecting that this category is substantially larger than the 101 to 500 range.

A4. Risk tolerance ρ summarizes how aggressively a person bets. We use $\rho = 0.65$ for low risk, $\rho = 1.00$ for moderate risk, and $\rho = 1.55$ for high risk.

A5. The 2025 U.S. sportsbook hold rate is represented by national commercial sports betting revenue divided by handle [12].

A6. For the U.K., the baseline share of active bettors is set to 10%, matching the official share of adults who reported betting on sports or race events online or through an app in the past four weeks [13]. The provisional mid-2025 U.K. population estimate is used when turning total revenue into a conservative loss estimate per active bettor [14].

A7. The model predicts average one-year outcomes, not the full spread of possible wins and losses. That spread matters in real life, but the average is the clearest quantity for comparing groups within the page limit.

3.3 Variables

Table 3 lists the main symbols used in the model. Clickable entries in the last column jump to the derivation, assumption, or source entry that introduces the number or formula. Spreadsheet-based values jump to the raw contest-sheet excerpts in Appendix A, and bracketed source tags identify the corresponding reference. For country pairs, the order is U.S.; U.K. For grouped demographic values, the order is male, female — 18–34, 35–49, 50–64, 65+ — no college, B.A.+. For rows with an overall value, that total comes first. For π_{sb}, the five sport triplets are football, basketball, soccer, baseball, and horse racing, each in the bucket order 1–100, 101–500, 500+. For $q_{k,j}$, each four-number block follows once only, once in a while, monthly, weekly+.

3.4 The Model

3.4.1 Model development

We build the P2 model in three steps: estimate who is likely to bet, estimate the yearly loss of a typical active bettor, and then adjust that loss for demographic differences. Figure 4 shows the raw inputs that set the national scale for those steps [2].

3.4.2 Active bettor probability

We begin with a country-wide baseline. In the U.S., the overall chance of having an online sportsbook account and actually betting through it is

$$p^{\text{act}}_{0,\text{US}} = 0.22 \times 0.83 = 0.1826.$$

Back to contents

Table 3: Variables and parameters in the annual gambling outcome model [2, 12, 13, 14]

Symbol	Meaning	Unit	Value used / source
c	country	category	U.S. or U.K.
$d = (g, a, e)$	demographic profile made of gender, age group, and education	category	input
ρ	scenario multiplier showing low, moderate, or high betting aggressiveness		0.65, 1.00, 1.55
$p_{0,c}^{\text{act}}$	country baseline active-bettor probability		0.1826 [2]; 0.10 [13]
p_j^{act}	observed active-bettor percentage for slice j		0.261; 0.115 — 0.301; 0.279; 0.072; 0.035 — 0.168; 0.220 [2]
r_j^{act}	relative participation factor for slice j		1.429; 0.633 — 1.648; 1.527; 0.394; 0.193 — 0.920; 1.205 [2]
$p_{\text{act}}(c, d)$	modeled active-bettor probability		$\min(0.95,\ p_{0,c}^{\text{act}} M_{\text{act}})$
W_{US}	mean monthly U.S. wager for an active bettor	USD	202.7 [2]
π_{sb}	share of respondents in sport s and wager bucket b		65,22,9 — 65,22,10 — 64,21,10 — 62,23,12 — 63,20,12 [2]
m_b	midpoint assigned to wager bucket b	USD	50; 300; 1000
h_{US}	2025 U.S. sportsbook hold rate		0.1016 [12]
χ_{FX}	2025 exchange factor used to convert USD behavioral anchors into GBP	GBP per USD	0.7615 [15]
$F_{\text{US}}, F_{\text{UK}}$	weighted monthly betting-frequency indices for the U.S. and U.K.		7.895; 4.085 [2]
$L_{\text{US}}^{(0)}$	baseline annual U.S. loss for an active bettor before demographic adjustments	USD	247.1 [2], [12]
$L_{\text{UK}}^{(0)}$	baseline annual U.K. loss for an active bettor before demographic adjustments	GBP	192.0 [2], [13], [14], [15]

Continued on next page

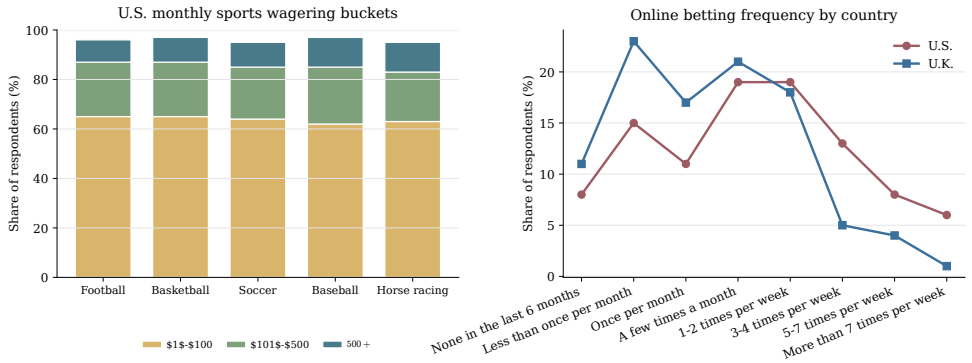


Figure 4: Raw survey inputs used to build the P2 wagering model [2]

Here 0.22 comes from the survey item "Has account with online sportsbook," and 0.83 comes from the separate survey item "Of those with an account, percent that have placed a bet on a sporting event through an online sportsbook?" [2]. Multiplying them gives a simple baseline estimate of the share of U.S. adults who are active online sportsbook users. In the U.K., we use the official active-participation rate

$$p_{0,\text{UK}}^{\text{act}} = 0.10.$$

This 0.10 comes from the official U.K. Gambling Survey result that 10% of adults reported betting on sports or race events online or through an app in the past four weeks [13]. We use that percentage as a practical

Back to contents

Table 3: Variables and parameters in the annual gambling outcome model (continued)

Symbol	Meaning	Unit	Value used / source
$L_{\mathrm{UK,beh}}, L_{\mathrm{UK,rev}}$	U.K. behavior and revenue anchors	GBP	97.4; 378.6 [2], [13], [14], [15]
G_{UK}	2025 U.K. remote-betting revenue	GBP	2.63069×10^9 [2]
N_{UK}	provisional mid-2025 U.K. population	people	6.9487×10^7 [14]
$M_{\mathrm{act}}(d)$	participation adjustment		$(r_g^{\mathrm{act}} r_a^{\mathrm{act}} r_e^{\mathrm{act}})^{1/3}$
D_j	annualized deposit score for slice j		15.80 — 16.31; 14.26 — 16.64; 16.88; 11.99; 5.36 — 16.38; 14.19 [2]
$q_{k,j}$	survey share in deposit bin k for slice j		16,39,22,22 — 15,39,22,23 — 18,39,22,19 — 16,25,25,24 — 12,41,22,24 — 19,49,17,15 — 32,48,13,3 — 14,38,23,23 — 19,40,21,19 [2]
S_j	large-stake adjustment for slice j		1.146; 0.693 — 1.021; 1.095; 0.845; 0.241 — 0.849; 1.205 [2]
r_j^{beh}	relative betting-intensity factor for slice j		1.183; 0.625 — 1.075; 1.170; 0.641; 0.082 — 0.880; 1.083 [2]
$M_{\mathrm{beh}}(d)$	betting-intensity adjustment		$(r_g^{\mathrm{beh}} r_a^{\mathrm{beh}} r_e^{\mathrm{beh}})^{1/3}$
Δ_j, Δ_0	lose-minus-win gap for slice j and the overall sample		0.07 — 0.03; 0.13 — 0.05; 0.07; 0.30; -0.02 — 0.17; -0.08 [2]
κ	coefficient linking the lose-win gap to losses		1.25
r_j^{out}	outcome-based relative factor for slice j		0.950; 1.075 — 0.975; 1.000; 1.287; 0.887 — 1.125; 0.812 [2]
$M_{\mathrm{out}}(d)$	outcome adjustment		$(r_g^{\mathrm{out}} r_a^{\mathrm{out}} r_e^{\mathrm{out}})^{1/3}$
$L_c^{(0)}$	country-specific baseline annual loss before demographic adjustments	local currency	247.1; 192.0 [2], [12], [13], [14], [15]
$Y_{\mathrm{cond}}(c,d,\rho)$	average annual net outcome for an active bettor	local currency	$-L_c^{(0)} M_{\mathrm{beh}} M_{\mathrm{out}} \rho$
$Y(c,d,\rho)$	average annual net outcome for a representative adult	local currency	$p_{\mathrm{act}} Y_{\mathrm{cond}}$

baseline for the share of U.K. adults who are currently active online sports bettors. The spreadsheet also tells us how participation differs by gender, age, and education. For each category $j \in \{g, a, e\}$, we turn the reported participation percentage p_j^{act} into a simple relative factor: Applying the same account-times-bet rule to each reported demographic slice gives

$$p_j^{\mathrm{act}} = (0.261,\ 0.115 \mid 0.301,\ 0.279,\ 0.072,\ 0.035 \mid 0.168,\ 0.220),$$

in the order male, female | 18–34, 35–49, 50–64, 65+ | no college, B.A.+.

$$r_j^{\mathrm{act}} = \frac{p_j^{\mathrm{act}}}{p_{0,\mathrm{US}}^{\mathrm{act}}}.$$

Because the data come as separate broad percentages rather than one detailed joint table, we combine the three effects with a geometric mean rather than multiplying them outright. We use

$$M_{\mathrm{act}}(d) = \left(r_g^{\mathrm{act}}\, r_a^{\mathrm{act}}\, r_e^{\mathrm{act}}\right)^{1/3}.$$

Back to contents

This gives one overall participation adjustment. Table 3 lists the resulting slice-specific values. The resulting chance of being an active bettor is

$$p_{\text{act}}(c,d) = \min\bigl(0.95,\ p_{0,c}^{\text{act}}\, M_{\text{act}}(d)\bigr).$$

3.4.3 Baseline annual loss for an active bettor

Next, we estimate the yearly loss of a typical active bettor before adding demographic adjustments. For the U.S., Figure 4 gives monthly wager buckets. We represent the three reported wager ranges by the midpoints 50, 300, and 1000, corresponding to the buckets \$1 to \$100, \$101 to \$500, and \$500+. We first compute a monthly estimate for each of the five sports shown in the spreadsheet and then take a simple average across those sports, since the table reports the five sports in parallel rather than as parts of one combined handle total. This gives

$$W_{\text{US}} = \frac{1}{5}\sum_{s=1}^{5}\sum_{b=1}^{3}\pi_{sb}m_b = 202.7,$$

where s indexes the five reported sports and π_{sb} is the share of respondents in bucket b for sport s. The official 2025 U.S. sports-betting handle and revenue imply that sportsbooks keep

$$h_{\text{US}} = \frac{16.96}{166.94} = 0.1016$$

[12] of the money wagered. In ordinary language, the hold rate is the sportsbook's average edge. So the baseline annual loss of an active U.S. bettor is

$$L_{\text{US}}^{(0)} = 12\, W_{\text{US}}\, h_{\text{US}} = 247.1.$$

The U.K. data are less direct, so we estimate the baseline in two different ways and then take a balanced middle value between them. The first estimate rescales the U.S. loss by the ratio of monthly betting frequencies from Figure 4 and then converts the result from USD to GBP. To build those frequency indices, we translate the eight survey categories into rough monthly betting counts:

$$w = (0,\ 0.5,\ 1,\ 3,\ 6,\ 14,\ 26,\ 35),$$

for "none in the last 6 months," "less than once per month," "once per month," "a few times a month," "1–2 times per week," "3–4 times per week," "5–7 times per week," and "more than 7 times per week." If $q_{k,c}^{\text{freq}}$ is the reported share in category k for country c, then

$$F_c = \sum_{k=1}^{8} q_{k,c}^{\text{freq}} w_k.$$

Using the survey percentages gives

$$F_{\text{US}} = 7.895, \qquad F_{\text{UK}} = 4.085.$$

These are comparison scores rather than exact counts of bets. For the currency step, we use the 2025 average Federal Reserve monthly U.K.-pound rate, which is 1.3133 USD per GBP, or equivalently

$$\chi_{\text{FX}} = 0.7615 \text{ GBP per USD}$$

[15]. This gives

$$L_{\text{UK,beh}} = L_{\text{US}}^{(0)}\frac{F_{\text{UK}}}{F_{\text{US}}}\chi_{\text{FX}} = 247.1 \cdot \frac{4.085}{7.895} \cdot 0.7615 = 97.4.$$

This behavior-based estimate is paired with a second estimate built from total U.K. remote-betting revenue

Back to contents

divided by a conservative count of active adults:

$$L_{\mathrm{UK,rev}} = \frac{G_{\mathrm{UK}}}{N_{\mathrm{UK}} p_{0,\mathrm{UK}}^{\mathrm{act}}} = \frac{2.63069 \times 10^9}{(6.9487 \times 10^7)(0.10)} = 378.6,$$

where G_{UK} comes from the spreadsheet and N_{UK} is the provisional mid-2025 U.K. population [2, 14]. The first estimate is based on betting behavior, while the second is based on market totals, so we place the final baseline between them using

$$L_{\mathrm{UK}}^{(0)} = \sqrt{L_{\mathrm{UK,beh}} L_{\mathrm{UK,rev}}} = 192.0.$$

The geometric mean gives a middle value without ignoring the gap between the two anchors. Table 4 lists the resulting constants.

Table 4: Main numerical constants used in the P2 model [2, 12, 13, 14]

Quantity	Meaning	Unit	Value
W_{US}	mean monthly U.S. sports wagering per active bettor	USD per month	202.7
h_{US}	2025 U.S. sportsbook hold rate	share of handle	0.102
$L_{\mathrm{US}}^{(0)}$	baseline annual U.S. loss for an active bettor	USD per year	247.1
F_{US}	U.S. monthly online-betting frequency index	score	7.895
F_{UK}	U.K. monthly online-betting frequency index	score	4.085
$L_{\mathrm{UK,beh}}$	U.K. behavior-scaled active-bettor loss anchor	GBP per year	97.4
$L_{\mathrm{UK,rev}}$	U.K. revenue-consistency loss anchor	GBP per year	378.6
$L_{\mathrm{UK}}^{(0)}$	geometric-mean U.K. active-bettor baseline	GBP per year	192.0
$p_{\mathrm{act,US}}^{(0)}$	baseline U.S. active bettor probability	share of adults	0.183
$p_{\mathrm{act,UK}}^{(0)}$	baseline U.K. active bettor probability	share of adults	0.100

Note. F_{US} and F_{UK} are weighted monthly betting-frequency scores built from the survey's broad frequency categories. $L_{\mathrm{UK,beh}}$ scales the U.S. baseline by relative betting frequency and then converts the result from USD to GBP using the 2025 average exchange rate assumption. $L_{\mathrm{UK,rev}}$ comes from official U.K. revenue totals divided by a conservative count of active bettors.

3.4.4 Demographic betting intensity and outcome multipliers

We now adjust that baseline loss for different kinds of people. We use two adjustments. The first reflects how intensely a group tends to gamble: how often it deposits money and how likely it is to place large bets. For each demographic slice j, define a deposit score

$$D_j = 1q_{1,j} + 4q_{2,j} + 12q_{3,j} + 52q_{4,j},$$

where $q_{1,j}, \ldots, q_{4,j}$ are the survey shares for depositing once only, once in a while, monthly, and weekly or more often [2]. The weights 1, 4, 12, and 52 are rough yearly counts for those four frequency levels. Let $s_{100,j}$ and $s_{500,j}$ denote the shares reporting at least 100 and at least 500 wagered in one day. The large-stake factor is

$$S_j = \sqrt{\frac{s_{100,j}}{s_{100,0}} \frac{s_{500,j}}{s_{500,0}}},$$

This is then combined into one overall betting-intensity adjustment:

$$r_j^{\mathrm{beh}} = \frac{D_j}{D_0} S_j, \qquad M_{\mathrm{beh}}(d) = \left(r_g^{\mathrm{beh}} \, r_a^{\mathrm{beh}} \, r_e^{\mathrm{beh}} \right)^{1/3}.$$

The second adjustment is based on whether groups report ending the year ahead or behind. Let $\Delta_j = p_j^{\mathrm{lose}} - p_j^{\mathrm{win}}$ and let Δ_0 be the overall survey value. A larger Δ_j means that the group reports losing more often than winning. We define

$$r_j^{\mathrm{out}} = \max\bigl(0.70,\ 1 + \kappa(\Delta_j - \Delta_0)\bigr), \qquad \kappa = 1.25,$$

Back to contents

$$M_{\text{out}}(d) = \left(r_g^{\text{out}}\, r_a^{\text{out}}\, r_e^{\text{out}}\right)^{1/3}.$$

This keeps the adjustment moderate. Table 3 lists the resulting slice-specific values.

3.4.5 Final annual outcome function

Putting the pieces together gives the final one-year loss functions:

$$Y_{\text{cond}}(c, d, \rho) = -L_c^{(0)}\, M_{\text{beh}}(d)\, M_{\text{out}}(d)\, \rho,$$

$$Y(c, d, \rho) = p_{\text{act}}(c, d)\, Y_{\text{cond}}(c, d, \rho).$$

These are the quantities used in every table and figure below. The first equation describes an active bettor. The second describes the wider population after accounting for the fact that not everyone in that demographic group actually bets. The risk multiplier ρ should be read as a scenario choice, not as a value observed directly in the data: low-risk bettors are assumed to bet more cautiously, while high-risk bettors are assumed to bet more aggressively.

3.5 Results

Figure 5 shows the model's average annual loss per adult at moderate risk for combinations of age, gender, and education in the U.S. and U.K. In both countries, the largest predicted losses occur for men aged 35 to 49 with a bachelor's degree or higher, at roughly 65 USD per adult in the U.S. and 28 GBP per adult in the U.K. The main reason is not just that this group reports losing more often. It is that this group combines higher betting participation with stronger betting intensity.

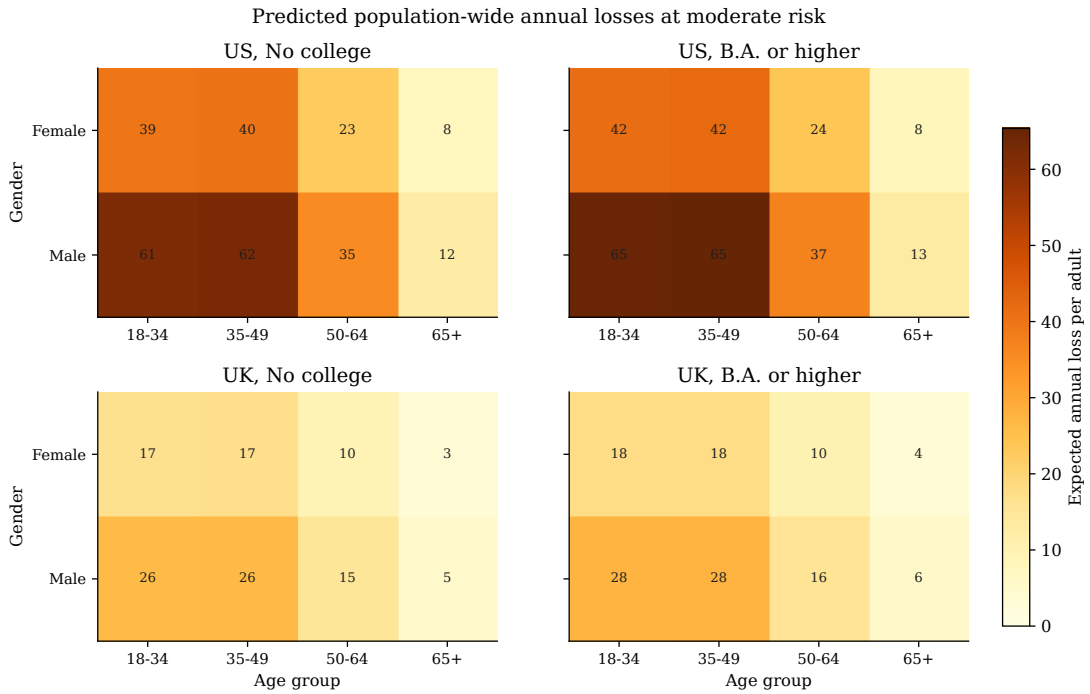


Figure 5: Predicted annual loss per representative adult at moderate risk across demographic groups [2, 13]

Table 5 shows representative profiles from both countries and all three risk levels. The most exposed case is a high-risk U.S. male aged 18 to 34 without a college degree, with a one-year loss of about 403 USD as an active bettor and about 95 USD on a population basis. At the other end, a low-risk U.K. female aged 65 or older with a bachelor's degree or higher loses about 44 GBP as an active bettor and about 2 GBP on a

Back to contents

population basis. The conditional column refers to active bettors only; the final column averages over the whole demographic group.

Table 5: Representative P2 model outputs for varied demographic profiles

Country	Gender	Age group	Education	Risk	p_{act}	$\mathbb{E}[Y \mid \text{active}]$	$\mathbb{E}[Y]$
US	Male	18-34	No college	High	0.236	-403	-95
US	Female	35-49	B.A. or higher	Moderate	0.192	-219	-42
US	Male	50-64	B.A. or higher	Low	0.160	-150	-24
UK	Male	35-49	B.A. or higher	High	0.138	-312	-43
UK	Female	18-34	No college	Moderate	0.099	-170	-17
UK	Female	65+	B.A. or higher	Low	0.053	-44	-2

Note. Y is the predicted annual net gambling outcome in local currency, so a leading minus sign means an expected loss. The conditional column assumes the person is already an active online sports bettor; the final column multiplies that loss by the model's active-bettor probability for the same demographic group.

Figure 6 shows the same contrast visually: the left panel is for active bettors, and the right panel averages over the full demographic group.

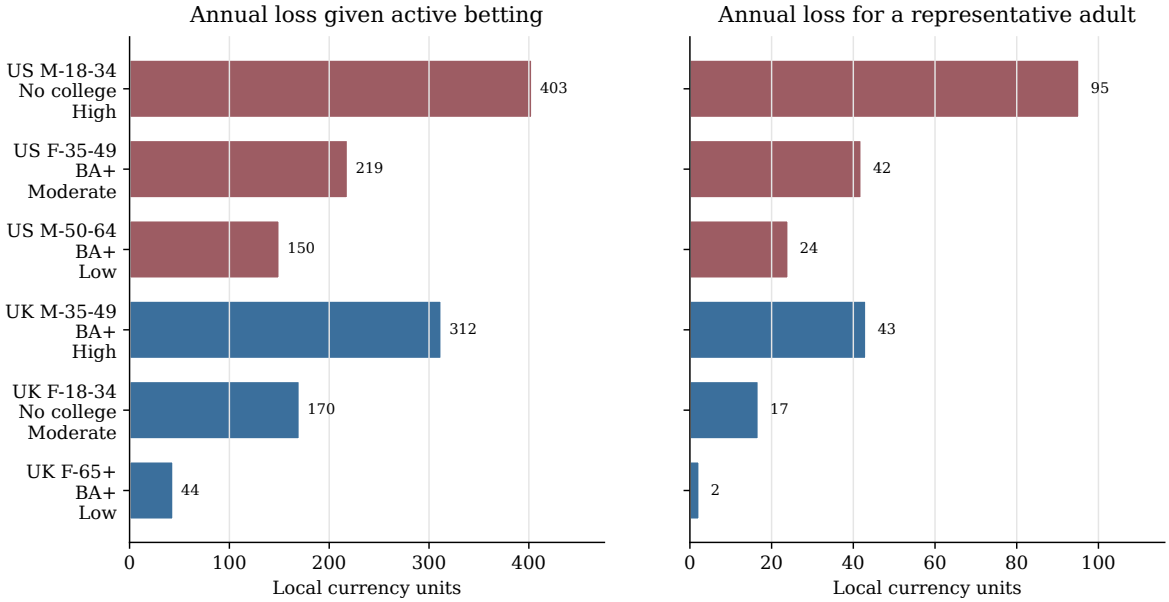


Figure 6: Representative annual losses for active bettors and for the overall population

3.6 Discussion

Two drivers matter most. Population-wide harm depends on both participation and loss size, so groups that both bet more often and bet more heavily create the largest totals. The betting-intensity term M_{beh} does most of the separation across groups, while the lower U.K. baseline keeps comparable U.K. losses below U.S. losses [2, 13, 15].

3.7 Sensitivity Analysis

To test robustness, we changed the most uncertain P2 inputs by as much as 10%: the U.S. hold rate, the midpoint of the open-ended 500+ wager bucket, the U.K. participation baseline, and the outcome coefficient κ. These are the inputs most likely to move if our simplifying choices are too rough. We then reran the model 1000 times for the six representative profiles in Table 5.

Back to contents

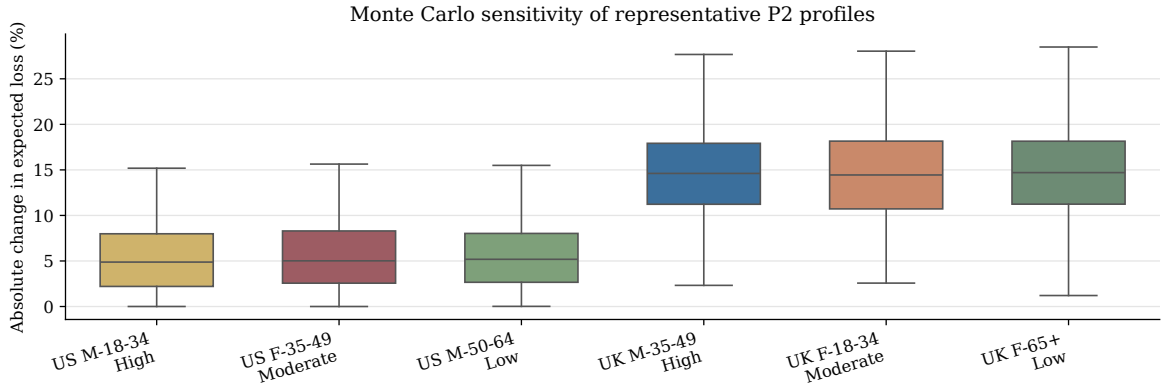


Figure 7: Monte Carlo sensitivity of representative P2 losses under 10% parameter variation. The trials vary the hold rate, the top wager-bucket midpoint, the outcome coefficient, and the U.K. participation baseline.

Figure 7 shows that the U.S. and U.K. models react to different sources of uncertainty. The U.S. profiles depend directly on the hold rate and the top wager bucket, while the U.K. profiles also depend on the exchange-rate conversion used in the behavior-based anchor. Averaging across the three representative U.S. profiles, the median absolute change is 5.03% and the mean 90th-percentile change is 10.65%. For the three representative U.K. profiles, the matching values are 14.58% and 21.17%. The U.K. side is therefore more sensitive to calibration choices, but the ranking of higher-risk and lower-risk groups still stays intact.

3.8 Strengths and Weaknesses

The main strength of the model is transparency: every adjustment comes from a survey percentage or a clearly stated market-based step, so the final function $Y(c, d, \rho)$ stays interpretable. The main weakness is that the survey remains broad. We do not observe one full gender-age-education table or bettor-level records, and the U.K. side uses a rougher calibration, so the results should be read as scale estimates rather than precise household forecasts.

3.8.1 Model refinement

With bettor-level records, the first improvement would be to estimate combined gender-age-education effects directly instead of piecing them together from separate broad tables. The second would be to replace the simple three-level risk multiplier with a model that distinguishes between more expensive bet types, such as same-game parlays, and lower-hold straight bets. Those changes would sharpen the yearly loss estimates without changing the overall logic of the model.

4 Don't Break the Bank

4.1 Defining the Problem

Problem 3 asks the most practical question in the paper: what do the first two models mean for a real household budget [1]? A raw dollar or pound loss is hard to judge on its own, because the same annual loss may be minor for one family and serious for another. So instead of stopping at "expected yearly loss," we translate the outputs of Problems 1 and 2 into three measures that ordinary readers can understand for an *active* online sports bettor:

months of cushion absorbed, probability of entering debt, ten-year wealth gap.

These measures tell the reader how much budget room one year of gambling uses up, how likely a bad year is to push the household into debt, and how much repeated losses can matter over ten years. They also let us

Back to contents

sort profiles into practical tiers such as manageable, strained, precarious, and critical instead of leaving the discussion at an abstract expected loss.

4.2 Assumptions

A1. Problem 3 focuses on *active bettors*, so the loss input is the average annual loss for an active bettor from Problem 2 rather than the population-wide average.

A2. In the ten-year scenario, salary, taxes, essential spending, and average gambling behavior are held constant in real terms. This keeps the long-run comparison easy to interpret and avoids building a second model just for income growth.

A3. Effective gambling cushion is $C = \max(D, 0)$, where D is disposable income from Problem 1. If $D < 0$, the household starts with no real budget buffer before gambling begins.

A4. Annual gambling losses for an active bettor are allowed to vary around the average loss from Problem 2. We model that variation with a right-skewed lognormal curve and link its spread to risk tolerance: 0.60 for low risk, 1.00 for moderate risk, and 1.60 for high risk.

A5. To describe long-run impact, we use a simple savings return of 4% and a simple debt cost of 15% per year. These are comparison assumptions, not forecasts.

A6. Repeated annual losses reduce savings first. If losses go beyond the annual cushion, the excess is treated like revolving debt in the ten-year scenario.

4.3 Variables

Table 6 lists the main symbols in the impact model. Clickable entries in the last column jump to the assumption, formula, or source table that defines them. Appendix A lists the raw contest-sheet excerpts and the official data sources behind Problems 1 and 2. Bracketed source tags identify the corresponding reference. For numeric pairs in the last column, the order is the lower-risk value first and the higher-risk value second unless a row says otherwise.

4.4 The Model

4.4.1 Connecting Problems 1 and 2

Problem 3 simply combines the first two models. For a full profile $x = (c, r, g, a, e, s, \rho)$, we define

$$C(x) = \max\big(D_{\mathrm{P1}}(x), 0\big), \qquad L(x) = -Y_{\mathrm{cond,P2}}(x).$$

The first quantity is the annual budget cushion available before gambling, and the second is the expected one-year loss for an active bettor.

4.4.2 Immediate budget pressure

The first public-facing measure asks: how much of the household's available budget room disappears in one year?

$$M(x) = \begin{cases} 12\,L(x)/C(x), & C(x) > 0, \\ \infty, & C(x) = 0. \end{cases}$$

This is easier to explain than a raw ratio. For example, $M = 4.4$ means one year of gambling losses uses up about 4.4 months' worth of that person's annual budget cushion.

Back to contents

Table 6: Variables and parameters in the P3 bankroll impact model

Symbol	Meaning	Unit	Value used / source
$x = (c, r, g, a, e, s, \rho)$	full bettor profile with country, region, gender, age, education, salary, and risk tolerance	category	input
$D(x)$	disposable income from Problem 1	local currency	$D_{\mathrm{P1}}(x)$
$C(x)$	effective annual gambling cushion $\max(D(x), 0)$	local currency	$\max(D(x), 0)$
$L(x)$	average annual gambling loss for an active bettor from Problem 2	local currency	$-Y_{\mathrm{cond,P2}}(x)$
$M(x)$	months of annual cushion absorbed by one year of gambling	months	$12L/C$ or ∞
ν_ρ	scenario parameter describing how spread out yearly losses are within a risk tier		0.60, 1.00, 1.60
$\widetilde{L}$	realized one-year gambling loss in the spread model	local currency	Lognormal$(\mu_\rho, \sigma_\rho^2)$
σ_ρ	spread parameter in the one-year loss model		0.555; 0.833; 1.127
$\mu_\rho(x)$	location parameter in the one-year loss model		$\log L(x) - \sigma_\rho^2/2$
Φ	cumulative bell-curve function used to compute debt probability		standard normal CDF
$p_{\mathrm{debt}}(x)$	probability that one year of gambling losses exceeds the annual cushion		$1 - \Phi((\log C - \mu_\rho)/\sigma_\rho)$
$A_{10}(r)$	10-year accumulation factor at rate r		12.006; 20.304
$G_{10}(x)$	ten-year wealth gap caused by repeated gambling losses	local currency	piecewise with $A_{10}(r_s)$ and $A_{10}(r_d)$
r_s	annual return on forgone savings in the scenario model		0.04
r_d	annual debt cost in the scenario model		0.15

4.4.3 Probability of entering debt

Average loss is not the whole story, especially for high-risk bettors. Some years will be worse than average. To capture that, we let the realized annual loss $\widetilde{L}$ vary around $L(x)$ using a right-skewed distribution. This choice reflects the practical fact that losses cannot go below zero but can occasionally be much larger than usual. If

$$\sigma_\rho = \sqrt{\log(1 + \nu_\rho^2)}, \qquad \mu_\rho(x) = \log L(x) - \frac{1}{2}\sigma_\rho^2,$$

then $\widetilde{L} \sim \mathrm{Lognormal}(\mu_\rho, \sigma_\rho^2)$ and the chance that one year's gambling losses go beyond the available cushion is

$$p_{\mathrm{debt}}(x) = \begin{cases} 1 - \Phi\left(\dfrac{\log C(x) - \mu_\rho(x)}{\sigma_\rho}\right), & C(x) > 0, \\ 1, & C(x) = 0, \end{cases}$$

For the three risk tiers, this gives $\sigma_\rho = (0.555,\ 0.833,\ 1.127)$. In plain language, $p_{\mathrm{debt}}(x)$ is the chance that one bad year is too large for the household's yearly budget cushion to absorb.

Back to contents

4.4.4 Ten-year wealth gap

The third measure looks beyond one year and asks what repeated losses do to savings. Let

$$A_{10}(r) = \frac{(1+r)^{10} - 1}{r}$$

be the standard 10-year accumulation factor. With $r_s = 0.04$ and $r_d = 0.15$, the values used in Table 6 are $A_{10}(0.04) = 12.006$ and $A_{10}(0.15) = 20.304$. If annual gambling loss stays within the yearly cushion, we treat it as money that otherwise could have been saved:

$$G_{10}(x) = L(x)A_{10}(r_s), \qquad L(x) \le C(x).$$

If annual loss is larger than the cushion, we treat the first $C(x)$ as lost savings and the rest as debt:

$$G_{10}(x) = C(x)A_{10}(r_s) + \big(L(x) - C(x)\big)A_{10}(r_d), \qquad L(x) > C(x).$$

So G_{10} measures the gap between a "no gambling" path and a "repeat this gambling every year" path after ten years.

4.4.5 Public-facing tier system

To summarize the outputs in ordinary language, we place each profile into one of four bankroll-stress tiers:

$$\text{Manageable if } M < 1 \text{ and } p_{\text{debt}} < 0.05,$$

$$\text{Strained if } 1 \le M < 3 \text{ or } 0.05 \le p_{\text{debt}} < 0.20,$$

$$\text{Precarious if } 3 \le M < 12 \text{ or } 0.20 \le p_{\text{debt}} < 0.50,$$

$$\text{Critical if } C = 0 \text{ or } M \ge 12 \text{ or } p_{\text{debt}} \ge 0.50.$$

These are not legal or medical cutoffs. They are simple communication thresholds meant to separate lighter stress from more serious stress in a way that readers can understand quickly.

4.5 Results

Table 7 shows six representative active-bettor profiles. The table covers all four public-facing tiers and reveals a clear pattern. The sharpest harm appears in U.S. profiles with very little budget cushion, while the U.K. examples in our table remain manageable in the short run even when the bettor is high risk. In that table, "Loss" is the one-year active-bettor loss from Problem 2, "Months" shows how much annual budget room is used up, and "Debt risk" means the one-year chance that losses are larger than the available cushion.

Table 7: Representative P3 bankroll impact profiles combining Problems 1 and 2

Country	Region	Demo	Salary	Risk	Disposable	Loss	Months	Debt risk	Tier
US	South	M-25-NC	35,000	High	-114	403	No cushion	100.0%	Critical
US	South	M-25-NC	45,000	High	3,444	403	1.4	0.7%	Strained
US	West	M-38-BA+	60,000	High	1,105	402	4.4	7.2%	Precarious
UK	England	M-40-BA+	45,000	High	21,887	312	0.2	0.0%	Manageable
UK	Wales	F-25-NC	25,000	Moderate	11,353	170	0.2	0.0%	Manageable
UK	N. Ireland	F-70-BA+	35,000	Low	21,488	44	0.0	0.0%	Manageable

Note. Demo abbreviations use gender, age, and education. NC denotes no college and BA+ denotes a bachelor's degree or higher. A leading minus sign in Disposable means the household is already short of money before gambling. Losses are the conditional annual gambling losses for active bettors from Problem 2, and Months measures how many months of annual disposable-income cushion are absorbed by one year of gambling.

The most severe profile is the high-risk U.S. South male age 25 with no college degree and a 35,000 USD salary. Problem 1 already says that this profile is slightly below zero disposable income before gambling

Back to contents

begins, so Problem 3 classifies it as critical immediately. Raising the same profile to 45,000 USD helps, but it does not remove the problem. That bettor still loses about 1.4 months of annual cushion per year and creates a ten-year wealth gap of about 4,840 USD.

Figure 8 shows the short-run and long-run impacts side by side. The left panel reports months of annual cushion absorbed, while the right panel shows the ten-year wealth gap. The precarious U.S. West profile stands out: although it is not immediately critical, one year of gambling still uses about 4.4 months of annual cushion and creates a ten-year gap of about 4,827 USD.

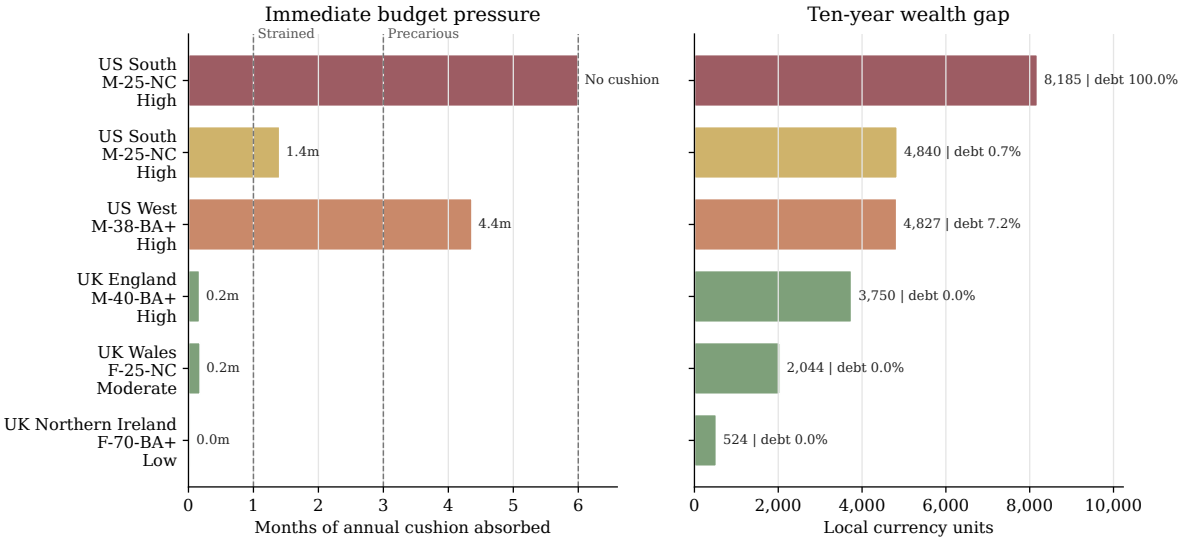


Figure 8: Immediate and long-run financial harm for representative active-bettor profiles. Debt-risk percentages are printed after each right-panel bar.

Figure 9 shows how this immediate budget pressure changes with salary for one high-exposure U.S. demographic and one high-exposure U.K. demographic. In the U.S. case, the high-risk curve crosses into the strained range at lower-middle salaries and becomes critical at the lowest salaries shown. In the U.K. case, all three risk curves stay below one month of annual cushion across the salary range shown, although the losses are still real and still create long-run wealth gaps.

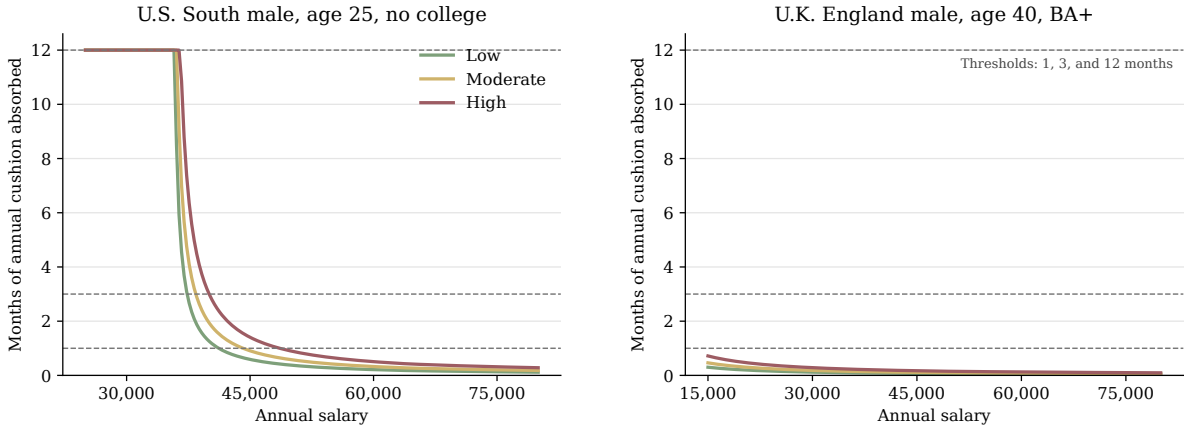


Figure 9: Months of annual disposable-income cushion absorbed as salary changes for two illustrative demographics

4.6 Discussion

The main point is that the same annual loss can be minor for one household and serious for another. The difference is budget cushion. Even manageable profiles still pay a long-run price: for example, the high-risk

Back to contents

U.K. England male profile stays manageable in the short run, yet repeated losses still add up to a ten-year wealth gap of about 3,750 GBP.

4.7 Sensitivity Analysis

To test robustness, we changed the most uncertain P3 inputs by as much as 10%: disposable income from Problem 1, one-year loss from Problem 2, the risk-based spread around losses, the savings return, and the debt cost. These are the inputs most likely to shift if our simplified long-run setup is imperfect. We then reran the model 1000 times for the representative profiles in Table 7.

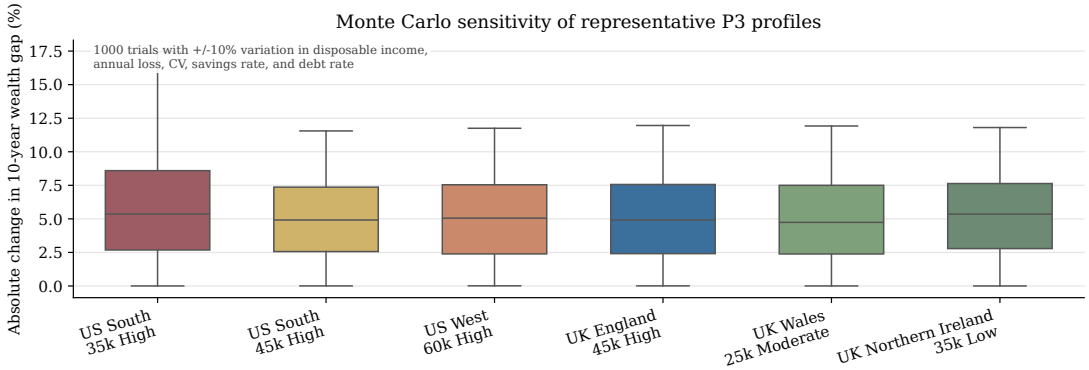


Figure 10: Monte Carlo sensitivity of the ten-year wealth gap under 10% parameter variation

Figure 10 shows that the ten-year gap is fairly stable. Across the six representative profiles, the median absolute change stays between 4.74% and 5.36%, while the 90th-percentile change stays between 8.99% and 11.80%. The most sensitive case is the precarious U.S. West 60,000 USD high-risk profile, but even there the shift is too small to change the main conclusion that low-cushion U.S. profiles are the most exposed.

4.8 Strengths and Weaknesses

The main strength of the model is readability. Months of cushion, debt-entry probability, and ten-year wealth gap are easier to interpret than a raw annual loss, and they still preserve the country and demographic structure built in Problems 1 and 2. The main weakness is that the long-run scenario is stylized: salary, spending, and behavior are held fixed, and the loss spread is imposed rather than estimated from bettor-level records. So this section should be read as a transparent screening tool, not as personalized advice.

4.8.1 Model refinement

If more time and data were available, the first improvement would be to estimate the yearly loss distribution directly from bettor-level transaction records instead of assigning spread values by risk tier. The second would be to let income, spending, and gambling intensity change over time, which would make the ten-year wealth-gap calculation more realistic.

5 Conclusions

Across the three problems, the same message keeps appearing. Online sports gambling does the most harm when yearly losses fall on households that already have very little room in the budget. Problem 1 measures that room, Problem 2 estimates who is most likely to lose money, and Problem 3 translates those losses into budget pressure, debt risk, and long-run wealth gaps. A loss that looks moderate on paper can still be serious when it lands on a household with almost no financial slack.

Back to contents

Further Studies. Further work should start with bettor-level records that track deposits, withdrawals, stakes, and realized losses over time. On the household side, adding U.S. state taxes, household size, and local housing costs would improve the disposable-income model. On the policy side, future work could compare sports gambling with other optional spending and track how repeated losses affect emergency savings and missed payments.

Summary. Society should be concerned, but not in the same way for everyone. The main risk lies in active bettors whose yearly losses may look moderate in dollars or pounds but are large relative to what is left after basic expenses, especially lower-cushion U.S. households under high-risk betting behavior.

For judges who want the implementation details, Appendix A lists the data sets and official releases used in the paper, and the complete Python scripts used to generate the tables and figures are included in the Code Appendix.

6 References

References

[1] MathWorks Math Modeling Challenge. The Rise of Online Gambling: What's at Stake? Problem statement, 2026.

[2] MathWorks Math Modeling Challenge. The Rise of Online Gambling curated data. Spreadsheet, 2026.

[3] Internal Revenue Service. Federal income tax rates and brackets. Accessed February 28, 2026. https://www.irs.gov/filing/federal-income-tax-rates-and-brackets

[4] Internal Revenue Service. IRS provides tax inflation adjustments for tax year 2025. Accessed February 28, 2026. https://www.irs.gov/newsroom/irs-provides-tax-inflation-adjustments-for-tax-year-2025

[5] Internal Revenue Service. Topic no. 751, Social Security and Medicare withholding rates. Accessed February 28, 2026. https://www.irs.gov/taxtopics/tc751

[6] Social Security Administration. Contribution and Benefit Base. Accessed February 28, 2026. https://www.ssa.gov/oact/cola/cbb.html

[7] GOV.UK. Income Tax rates and Personal Allowances. Accessed February 28, 2026. https://www.gov.uk/income-tax-rates

[8] GOV.UK. Income Tax in Scotland. Accessed February 28, 2026. https://www.gov.uk/scottish-income-tax

[9] GOV.UK. Rates and thresholds for employers 2025 to 2026. Accessed February 28, 2026. https://www.gov.uk/guidance/rates-and-thresholds-for-employers-2025-to-2026

[10] GOV.UK. National Insurance rates and categories: Contribution rates. Accessed February 28, 2026. https://www.gov.uk/national-insurance-rates-letters/contribution-rates

[11] GOV.UK. National Insurance and tax after State Pension age. Accessed February 28, 2026. https://www.gov.uk/tax-national-insurance-after-state-pension-age

[12] American Gaming Association. Commercial Gaming Revenue Hits $78.7 Billion in 2025, Driving Record $18.1 Billion in Gaming Taxes Nationwide. Published February 26, 2026. Accessed February 28, 2026. https://www.americangaming.org/commercial-gaming-revenue-hits-78-7-billion-in-2025-driving-record-18-1-billion-in-gaming-taxes-nationwide/

Back to contents

[13] UK Gambling Commission. Statistics on gambling participation, Wave 2, April to July 2025, official statistics. Accessed February 28, 2026. https://www.gamblingcommission.gov.uk/statistics-and-research/publication/statistics-on-gambling-participation-wave-2-april-to-july-2025-official

[14] Office for National Statistics. Provisional population estimate for the UK, mid-2025. Accessed February 28, 2026. https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates/bulletins/provisionalpopulationestimatefortheuk/mid2025

[15] Federal Reserve Board. Foreign Exchange Rates, G.5 monthly release pages. Accessed February 28, 2026. https://www.federalreserve.gov/releases/g5/

Back to contents

A Data Sets Used

This appendix lists the structured data sources behind the paper. In the last column, the links jump to the first place where that source is turned into a model quantity.

Table 8: Contest-provided data sources used in the paper

Data set	Fields used	Main role	Jump targets
M3 workbook: disposable-income sheet	U.S. age and region income/spending; U.K. age, nation, and income-group spending	Builds the P1 spending fits, baseline quantities, and representative disposable-income examples	U.S. age raw, U.S. region raw, U.K. quintiles, U.K. age All, U.K. nation raw
M3 workbook: online-sports-betting sheet	Sportsbook-account use, wager buckets, betting frequency, deposit frequency, large-stake shares, lose/win shares	Builds P2 participation rates, loss anchors, and demographic multipliers; P3 inherits the resulting loss model	account raw, outcome raw, deposit raw, stake raw, frequency raw, wager raw
M3 workbook: revenue sheet	U.K. remote betting revenue, Apr 2024–Mar 2025 row	Supplies the P2 revenue-based U.K. calibration anchor G_{UK}	raw row, U.K. revenue anchor

Table 9: Official public data sources used in the paper

Data set / release	Key value used	Main role	Jump targets
IRS inflation adjustments for tax year 2025	U.S. standard deduction	U.S. disposable-income tax model in P1	P1 variables
SSA Contribution and Benefit Base	Social Security wage base	U.S. payroll-tax cap in P1	P1 variables
GOV.UK tax and National Insurance pages	U.K. tax bands, Scotland bands, NI thresholds, NI rates	U.K. disposable-income tax model in P1	P1 model
AGA 2025 sports-betting totals	U.S. handle and revenue	U.S. sportsbook hold rate in P2	hold rate
UKGC Wave 2 participation release	U.K. active online sports/race betting share	U.K. active-bettor baseline in P2	U.K. active base
ONS mid-2025 population estimate	U.K. population total	U.K. revenue-based loss anchor in P2	U.K. revenue anchor
Federal Reserve G.5 exchange-rate release	2025 average USD/GBP conversion	GBP conversion in the behavior-based U.K. anchor	FX conversion

A.1 Contest Spreadsheet Excerpts

The tables below keep only the workbook blocks that are actually used in the model. Each caption note names the sheet and the exact row block used in the paper.

A.1.1 Problem 1 raw inputs

Table 10: Problem 1 U.S. age-anchor rows

Workbook location. Sheet "Expenditures (U.S.)"; age columns Under 25 through 75 and older; row labels Mean income before taxes, Food, Housing, Utilities, fuel, public services, Transportation, and Healthcare.

Row label	Under 25	25-34	35-44	45-54	55-64	65-74	75 and older
Mean income before taxes	48,514	102,494	128,285	141,121	121,571	75,460	56,028
Food	7215	9630	12460	12772	10214	8483	7168
Housing	16853	26380	30369	30747	27019	22329	21999
Utilities, fuel, public services	2181	4076	5053	5779	5138	4653	4234
Transportation	9243	12802	15581	17184	15085	11414	6855
Healthcare	1485	3825	5949	6748	6711	7715	7918

Back to contents

Table 11: Problem 1 U.S. region-anchor rows

Workbook location. Sheet "Expenditures (U.S.)"; region columns Northeast through West; row labels Mean income before taxes, Food, Housing, Utilities, fuel, public services, Transportation, and Healthcare.

Row label	Northeast	Midwest	South	West
Mean income before taxes	115,770	97,104	93,814	120,365
Food	11372	9677	9003	11746
Housing	29469	23065	23260	32147
Utilities, fuel, public services	5085	4376	4640	4978
Transportation	12341	12901	12768	15463
Healthcare	6289	6246	5870	6660

Table 12: Problem 1 U.K. income-quintile row

Workbook location. Sheet "Expenditures (U.K.)"; quintile columns Lowest 20% through Highest 20%; row label Mean household income per week (GBP).

Field	Lowest 20%	Second 20%	Third 20%	Fourth 20%	Highest 20%
Mean household income per week (GBP)	272	544	850	1257	2391

Table 13: Problem 1 U.K. age All-column rows

Workbook location. Sheet "Expenditures (U.K.)"; All column only for the five age blocks; row labels Food & non-alcoholic drinks, Housing(net), fuel & power, Transport, and Health.

Age block	Food	Housing	Transport	Health
Age ¡ 30	48.9	173.5	67.4	3.8
Age 30-49	71.7	120.7	88.4	6.7
Age 50-64	70.8	93.4	101	9.7
Age 65-74	60.4	72.6	70.1	11.4
Age ¿ 74	54.9	66	41.1	10.5

Table 14: Problem 1 U.K. nation-multiplier rows

Workbook location. Sheet "Expenditures (U.K.)"; nation columns England, Wales, Scotland, Northern Ireland, and All UK; row labels Food & non-alcoholic drinks, Housing(net), fuel & power, and Transport.

Commodity or service	England	Wales	Scotland	Northern Ireland	All UK
Food & non-alcoholic drinks	65.6	62.2	61.5	77.9	65.4
Housing(net), fuel & power	106.5	83.3	81.8	71.3	102.2
Transport	80.8	81.3	75.9	89.5	80.6

A.1.2 Problem 2 raw inputs

Table 15: Problem 2 sportsbook-account rows

Workbook location. Sheet "Online Sports Betting Personal"; columns 2025, Male, Female, 18–34, 35–49, 50–64, 65+, No college, and B.A. or higher; row labels Has account with online sportsbook and Of those with an account, percent that have placed a bet on a sporting event through an online sportsbook?.

Question / response	2025	Male	Fem.	18-34	35-49	50-64	65+	No coll.	B.A.+
Has account with online sportsbook	22	30	15	35	34	10	4	21	25
Of those with an account, percent that have placed a bet on a sporting event through an online sportsbook?	83	87	77	86	82	72	88	80	88

Table 16: Problem 2 win-loss rows

Workbook location. Sheet "Online Sports Betting Personal"; columns 2025, Male, Female, 18–34, 35–49, 50–64, 65+, No college, and B.A. or higher; row labels Win more than lose and Lose more than win.

Question / response	2025	Male	Fem.	18-34	35-49	50-64	65+	No coll.	B.A.+
Win more than lose	30	33	25	33	29	17	25	24	39
Lose more than win	37	36	38	38	36	47	23	41	31

Back to contents

Table 17: Problem 2 deposit-frequency rows

Workbook location. Sheet "Online Sports Betting Personal"; columns 2025, Male, Female, 18–34, 35–49, 50–64, 65+, No college, and B.A. or higher; row labels When I started and have never deposited again, Once in a while, Monthly, and Weekly or more often.

Question / response	2025	Male	Fem.	18-34	35-49	50-64	65+	No coll.	B.A.+
When I started and have never deposited again	16	15	18	16	12	19	32	14	19
Once in a while	39	39	39	25	41	49	48	38	40
Monthly	22	22	22	25	22	17	13	23	21
Weekly or more often	22	23	19	24	24	15	3	23	19

Table 18: Problem 2 large-stake rows

Workbook location. Sheet "Online Sports Betting Personal"; columns 2025, Male, Female, 18–34, 35–49, 50–64, 65+, No college, and B.A. or higher; row labels percent that have bet a total of 100 or more in one day and percent that have bet a total of 500 or more in one day.

Question / response	2025	Male	Fem.	18-34	35-49	50-64	65+	No coll.	B.A.+
Of those with an account AND have placed a bet, percent that have bet a total of $100 or more in one day using an online sportsbook.	56	63	43	56	62	48	26	51	63
Of those with an account AND have placed a bet, percent that have bet a total of $500 or more in one day using an online sportsbook.	24	28	15	25	26	20	3	19	31

Table 19: Problem 2 U.S./U.K. betting-frequency rows

Workbook location. Sheet "Online Sports Betting Personal"; columns U.S. and U.K.; rows None in the last 6 months through More than 7 times per week.

Frequency category	U.S.	U.K.
None in the last 6 months	8	11
Less than once per month	15	23
Once per month	11	17
A few times a month	19	21
1-2 times per week	19	18
3-4 times per week	13	5
5-7 times per week	8	4
More than 7 times per week	6	1

Table 20: Problem 2 monthly sport-wager rows

Workbook location. Sheet "Online Sports Betting Personal"; columns 1–100, 101–500, and 500+; sports Football, Basketball, Soccer, Baseball, and Horse racing.

Sport	$1-$100	$101-$500	$500+
Football	65	22	9
Basketball	65	22	10
Soccer	64	21	10
Baseball	62	23	12
Horse racing	63	20	12

Table 21: Problem 2 U.K. remote-betting revenue row

Workbook location. Sheet "Gambling Industry Revenue (U.S.)"; row Apr 2024 – Mar 2025; column Betting (remote).

Reporting period	Betting (remote)
Apr 2024 - Mar 2025	2630.69

B Code Appendix

B.1 Data Appendix Utility

The following helper script exports the raw spreadsheet excerpts used in Appendix A. It reads the contest workbook and writes the LaTeX tables imported above without manually retyping any of the values.

Listing 1: Utility script used to export the appendix spreadsheet excerpts

```
import os
from pathlib import Path

```

Back to contents

```
import pandas as pd


BASE_DIR = Path(__file__).resolve().parent.parent
DATA_PATH = BASE_DIR / "code" / "M3-Challenge-Problem-Data-2026.xlsx"
TAB_DIR = BASE_DIR / "tables"


def esc(text):
    """Escape LaTeX-special characters while normalizing spreadsheet labels."""
    if pd.isna(text):
        return ""
    value = str(text).replace("\xa0", " ")
    for token in ("\u00b9", "\u00b2", "\u00b3"):
        value = value.replace(token, "")
    value = value.replace("\u00a3", "GBP").replace("\u62e2", "GBP")
    replacements = {
        "\\": r"\textbackslash{}",
        "&": r"\&",
        "%": r"\%",
        "$": r"\$",
        "#": r"\#",
        "_": r"\_",
        "{": r"\{",
        "}": r"\}",
        "~": r"\textasciitilde{}",
        "^": r"\textasciicircum{}",
    }
    for src, dst in replacements.items():
        value = value.replace(src, dst)
    return value


def fmt_cell(value):
    """Preserve spreadsheet values with light cleanup only."""
    if pd.isna(value):
        return ".."
    if isinstance(value, float):
        if value.is_integer():
            return f"{int(value)}"
        return f"{value:g}"
    return esc(value)


def short_header(value):
    """Use compact display labels for wide appendix tables."""
    text = fmt_cell(value)
    mapping = {
        "Female": "Fem.",
        "No college": "No coll.",
        "B.A. or higher": "B.A.+",
    }
    return mapping.get(text, text)


def load_sheet(sheet_name):
    return pd.read_excel(DATA_PATH, sheet_name=sheet_name, header=None)


def make_colspec(num_cols, first_width):
    """Use one consistent table sizing rule across all appendix tables."""
    numeric_cols = num_cols - 1
    remaining = 0.84 - first_width
    width = max(0.045, remaining / max(1, numeric_cols))
    pieces = [f"|>{{\\raggedright\\arraybackslash}}p{{{first_width:.3f}\\linewidth}}|"]
    pieces.extend([f">{{\\centering\\arraybackslash}}p{{{width:.3f}\\linewidth}}|" for _ in range(numeric_cols)])
    return "".join(pieces)


def write_table(handle, target, caption, location, headers, rows, first_width):
    handle.write(f"\\hypertarget{{{target}}}{{}}\n")
    handle.write("\\begin{table}[H]\n")
    handle.write("\\centering\n")
    handle.write(f"\\caption{{{caption}}}\n")
    handle.write(
        f"\\parbox{{0.96\\linewidth}}{{\\footnotesize\\raggedright \\textit{{Workbook location.}} {location}}}\n"
    )
```

Back to contents

```
    handle.write("\\par\\vspace{3pt}\n")
    handle.write("\\scriptsize\n")
    handle.write("\\setlength{\\tabcolsep}{1.9pt}\n")
    handle.write("\\renewcommand{\\arraystretch}{1.08}\n")
    handle.write(f"\\begin{{tabular}}{{{make_colspec(len(headers), first_width)}}}\n")
    handle.write("\\hline\n")
    handle.write(" & ".join(headers) + r" \\" + "\n")
    handle.write("\\hline\n")
    for row in rows:
        handle.write(" & ".join(row) + r" \\" + "\n")
        handle.write("\\hline\n")
    handle.write("\\end{tabular}\n")
    handle.write("\\end{table}\n\n")


def p1_us_age_excerpt():
    us = load_sheet("Expenditures (U.S.)")
    headers = ["Row label"] + [esc(v) for v in us.iloc[2, 1:8].tolist()]
    row_labels = [
        "Mean income before taxes",
        "Food",
        "Housing",
        "Utilities, fuel, public services",
        "Transportation",
        "Healthcare",
    ]
    rows = []
    for label in row_labels:
        idx = us.index[us.iloc[:, 0] == label][0]
        rows.append([esc(label)] + [fmt_cell(v) for v in us.iloc[idx, 1:8].tolist()])
    return headers, rows


def p1_us_region_excerpt():
    us = load_sheet("Expenditures (U.S.)")
    headers = ["Row label"] + [esc(v) for v in us.iloc[2, 8:12].tolist()]
    row_labels = [
        "Mean income before taxes",
        "Food",
        "Housing",
        "Utilities, fuel, public services",
        "Transportation",
        "Healthcare",
    ]
    rows = []
    for label in row_labels:
        idx = us.index[us.iloc[:, 0] == label][0]
        rows.append([esc(label)] + [fmt_cell(v) for v in us.iloc[idx, 8:12].tolist()])
    return headers, rows


def p1_uk_quintiles_excerpt():
    uk = load_sheet("Expenditures (U.K.)")
    headers = ["Field"] + [esc(v) for v in uk.iloc[3, 9:14].tolist()]
    rows = [[esc(uk.iloc[4, 8])] + [fmt_cell(v) for v in uk.iloc[4, 9:14].tolist()]]
    return headers, rows


def p1_uk_age_all_excerpt():
    uk = load_sheet("Expenditures (U.K.)")
    starts = [3, 20, 37, 54, 71]
    labels = [
        "Food & non-alcoholic drinks",
        "Housing(net)\u00b2, fuel & power",
        "Transport",
        "Health",
    ]
    headers = ["Age block", "Food", "Housing", "Transport", "Health"]
    rows = []
    for i, start in enumerate(starts):
        end = starts[i + 1] if i + 1 < len(starts) else len(uk)
        age_label = esc(uk.iloc[start, 0]).replace("Household reference person ", "")
        values = []
        for label in labels:
            idx = next(r for r in range(start, end) if str(uk.iloc[r, 0]) == label)
            values.append(fmt_cell(uk.iloc[idx, 6]))
        rows.append([age_label] + values)
```

Back to contents

```
    return headers, rows


def p1_uk_country_excerpt():
    uk = load_sheet("Expenditures (U.K.)")
    headers = ["Commodity or service"] + [fmt_cell(v) for v in uk.iloc[8, 9:14].tolist()]
    labels = [
        "Food & non-alcoholic drinks",
        "Housing(net)\u00b9, fuel & power",
        "Transport",
    ]
    rows = []
    for label in labels:
        idx = next(r for r in range(9, 19) if str(uk.iloc[r, 8]) == label)
        rows.append([esc(label)] + [fmt_cell(v) for v in uk.iloc[idx, 9:14].tolist()])
    return headers, rows


def p2_account_excerpt():
    survey = load_sheet("Online Sports Betting Personal ")
    cols = [1, 3, 4, 5, 6, 7, 8, 9, 10]
    headers = ["Question / response"] + [short_header(v) for v in survey.iloc[2, cols].tolist()]
    rows = []
    for idx in [3, 16]:
        rows.append([esc(survey.iloc[idx, 0])] + [fmt_cell(v) for v in survey.iloc[idx, cols].tolist()])
    return headers, rows


def p2_outcome_excerpt():
    survey = load_sheet("Online Sports Betting Personal ")
    cols = [1, 3, 4, 5, 6, 7, 8, 9, 10]
    headers = ["Question / response"] + [short_header(v) for v in survey.iloc[2, cols].tolist()]
    rows = []
    for idx in [18, 19]:
        rows.append([esc(survey.iloc[idx, 0])] + [fmt_cell(v) for v in survey.iloc[idx, cols].tolist()])
    return headers, rows


def p2_deposit_excerpt():
    survey = load_sheet("Online Sports Betting Personal ")
    cols = [1, 3, 4, 5, 6, 7, 8, 9, 10]
    headers = ["Question / response"] + [short_header(v) for v in survey.iloc[2, cols].tolist()]
    rows = []
    for idx in [30, 31, 32, 33]:
        rows.append([esc(survey.iloc[idx, 0])] + [fmt_cell(v) for v in survey.iloc[idx, cols].tolist()])
    return headers, rows


def p2_stake_excerpt():
    survey = load_sheet("Online Sports Betting Personal ")
    cols = [1, 3, 4, 5, 6, 7, 8, 9, 10]
    headers = ["Question / response"] + [short_header(v) for v in survey.iloc[2, cols].tolist()]
    rows = []
    for idx in [40, 41]:
        rows.append([esc(survey.iloc[idx, 0])] + [fmt_cell(v) for v in survey.iloc[idx, cols].tolist()])
    return headers, rows


def p2_frequency_excerpt():
    survey = load_sheet("Online Sports Betting Personal ")
    headers = ["Frequency category"] + [fmt_cell(v) for v in survey.iloc[1, 13:15].tolist()]
    rows = []
    for idx in range(2, 10):
        rows.append([esc(survey.iloc[idx, 12])] + [fmt_cell(v) for v in survey.iloc[idx, 13:15].tolist()])
    return headers, rows


def p2_wager_excerpt():
    survey = load_sheet("Online Sports Betting Personal ")
    headers = ["Sport"] + [fmt_cell(v) for v in survey.iloc[13, 13:16].tolist()]
    rows = []
    for idx in range(14, 19):
        rows.append([esc(survey.iloc[idx, 12])] + [fmt_cell(v) for v in survey.iloc[idx, 13:16].tolist()])
    return headers, rows


def p2_uk_revenue_excerpt():
```

Back to contents

```
    revenue = load_sheet("Gambling Industry Revenue (U.S.")
    headers = ["Reporting period", esc(revenue.iloc[17, 7])]
    rows = [[fmt_cell(revenue.iloc[34, 0]), fmt_cell(revenue.iloc[34, 7])]]
    return headers, rows


def build_output():
    out_path = TAB_DIR / "app_spreadsheet_extracts.tex"
    with out_path.open("w", encoding="utf-8") as handle:
        handle.write("% Auto-generated contest spreadsheet excerpts used in the appendix.\n\n")
        handle.write("\\subsection{Contest Spreadsheet Excerpts}\n")
        handle.write(
            "The tables below keep only the workbook blocks that are actually used in the model. "
            "Each caption note names the sheet and the exact row block used in the paper.\n\n"
        )
        handle.write("\\begingroup\\sloppy\n\n")

        handle.write("\\subsubsection{Problem 1 raw inputs}\n")
        headers, rows = p1_us_age_excerpt()
        write_table(
            handle,
            "data-p1-us-age-raw",
            "Problem 1 U.S. age-anchor rows",
            'Sheet "Expenditures (U.S.)"; age columns Under 25 through 75 and older; row labels Mean income before taxes, Food, Housing, Utilities, fuel, public services, Transportation, and Healthcare.',
            headers,
            rows,
            0.28,
        )

        headers, rows = p1_us_region_excerpt()
        write_table(
            handle,
            "data-p1-us-region-raw",
            "Problem 1 U.S. region-anchor rows",
            'Sheet "Expenditures (U.S.)"; region columns Northeast through West; row labels Mean income before taxes, Food, Housing, Utilities, fuel, public services, Transportation, and Healthcare.',
            headers,
            rows,
            0.31,
        )

        headers, rows = p1_uk_quintiles_excerpt()
        write_table(
            handle,
            "data-p1-uk-quintiles",
            "Problem 1 U.K. income-quintile row",
            'Sheet "Expenditures (U.K.)"; quintile columns Lowest 20\\% through Highest 20\\%; row label Mean household income per week (GBP).',
            headers,
            rows,
            0.38,
        )

        headers, rows = p1_uk_age_all_excerpt()
        write_table(
            handle,
            "data-p1-uk-age-all",
            "Problem 1 U.K. age All-column rows",
            'Sheet "Expenditures (U.K.)"; All column only for the five age blocks; row labels Food \\& non-alcoholic drinks, Housing(net), fuel \\& power, Transport, and Health.',
            headers,
            rows,
            0.42,
        )

        headers, rows = p1_uk_country_excerpt()
        write_table(
            handle,
            "data-p1-uk-country",
            "Problem 1 U.K. nation-multiplier rows",
            'Sheet "Expenditures (U.K.)"; nation columns England, Wales, Scotland, Northern Ireland, and All UK; row labels Food \\& non-alcoholic drinks, Housing(net), fuel \\& power, and Transport.',
            headers,
            rows,
            0.33,
        )
```

Back to contents

```
307
308         handle.write("\\subsubsection{Problem 2 raw inputs}\n")
309         headers, rows = p2_account_excerpt()
310         write_table(
311             handle,
312             "data-p2-account-raw",
313             "Problem 2 sportsbook-account rows",
314             'Sheet "Online Sports Betting Personal"; columns 2025, Male, Female, 18--34, 35--49, 50--64, 65+, No college,
        and B.A. or higher; row labels Has account with online sportsbook and Of those with an account, percent that have
        placed a bet on a sporting event through an online sportsbook?.',
315             headers,
316             rows,
317             0.40,
318         )
319
320         headers, rows = p2_outcome_excerpt()
321         write_table(
322             handle,
323             "data-p2-outcome-raw",
324             "Problem 2 win-loss rows",
325             'Sheet "Online Sports Betting Personal"; columns 2025, Male, Female, 18--34, 35--49, 50--64, 65+, No college,
        and B.A. or higher; row labels Win more than lose and Lose more than win.',
326             headers,
327             rows,
328             0.40,
329         )
330
331         headers, rows = p2_deposit_excerpt()
332         write_table(
333             handle,
334             "data-p2-deposit-raw",
335             "Problem 2 deposit-frequency rows",
336             'Sheet "Online Sports Betting Personal"; columns 2025, Male, Female, 18--34, 35--49, 50--64, 65+, No college,
        and B.A. or higher; row labels When I started and have never deposited again, Once in a while, Monthly, and Weekly or
        more often.',
337             headers,
338             rows,
339             0.40,
340         )
341
342         headers, rows = p2_stake_excerpt()
343         write_table(
344             handle,
345             "data-p2-stake-raw",
346             "Problem 2 large-stake rows",
347             'Sheet "Online Sports Betting Personal"; columns 2025, Male, Female, 18--34, 35--49, 50--64, 65+, No college,
        and B.A. or higher; row labels percent that have bet a total of 100 or more in one day and percent that have bet a
        total of 500 or more in one day.',
348             headers,
349             rows,
350             0.40,
351         )
352
353         headers, rows = p2_frequency_excerpt()
354         write_table(
355             handle,
356             "data-p2-frequency-raw",
357             "Problem 2 U.S./U.K. betting-frequency rows",
358             'Sheet "Online Sports Betting Personal"; columns U.S. and U.K.; rows None in the last 6 months through More
        than 7 times per week.',
359             headers,
360             rows,
361             0.38,
362         )
363
364         headers, rows = p2_wager_excerpt()
365         write_table(
366             handle,
367             "data-p2-wager-raw",
368             "Problem 2 monthly sport-wager rows",
369             'Sheet "Online Sports Betting Personal"; columns 1--100, 101--500, and 500+; sports Football, Basketball,
        Soccer, Baseball, and Horse racing.',
370             headers,
371             rows,
372             0.26,
373         )
374
```

Back to contents

```
        headers, rows = p2_uk_revenue_excerpt()
        write_table(
            handle,
            "data-p2-uk-revenue-raw",
            "Problem 2 U.K. remote-betting revenue row",
            'Sheet "Gambling Industry Revenue (U.S.)"; row Apr 2024 -- Mar 2025; column Betting (remote).',
            headers,
            rows,
            0.55,
        )
        handle.write("\\endgroup\n")


if __name__ == "__main__":
    os.makedirs(TAB_DIR, exist_ok=True)
    build_output()
```

B.2 Playing With House Money

The complete Python script for Problem 1 appears below. It is the exact file used to generate the imported tables and figures, and the inline comments document the calibration logic, tax assumptions, and plotting choices.

Listing 2: Complete Python script for Problem 1

```
"""Problem 1 workflow for disposable-income estimation.

The script converts the curated expenditure tables into explicit disposable-
income functions for the United States and the United Kingdom, then writes the
LaTeX-ready figures and tables imported by the paper.
"""

import os
import re

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from matplotlib.ticker import FuncFormatter

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
PROJECT_DIR = os.path.dirname(BASE_DIR)
DATA_PATH = os.path.join(BASE_DIR, "M3-Challenge-Problem-Data-2026.xlsx")
FIG_DIR = os.path.join(PROJECT_DIR, "figures")
TAB_DIR = os.path.join(PROJECT_DIR, "tables")

os.makedirs(FIG_DIR, exist_ok=True)
os.makedirs(TAB_DIR, exist_ok=True)

plt.rcParams.update(
    {
        "figure.facecolor": "white",
        "axes.facecolor": "white",
        "savefig.facecolor": "white",
        "font.family": "serif",
        "font.serif": ["CMU Serif", "Computer Modern Roman", "STIX Two Text", "DejaVu Serif"],
        "mathtext.fontset": "cm",
        "axes.unicode_minus": False,
        "pdf.fonttype": 42,
        "ps.fonttype": 42,
    }
)


def to_float(value):
    """Convert spreadsheet entries to floats while tolerating missing markers."""
    if pd.isna(value):
        return np.nan
    if isinstance(value, (int, float, np.integer, np.floating)):
        return float(value)
    text = str(value).strip().replace(",", "")
    text = text.replace("\u2014", "-").replace("\u2013", "-")
    if text in {"", "-", "..", "..."}:
        return np.nan
```

Back to contents

```
    try:
        return float(text)
    except ValueError:
        return np.nan


def normalize_label(value):
    """Normalize spreadsheet labels before matching them in code."""
    if pd.isna(value):
        return ""
    text = str(value).replace("\xa0", " ").strip()
    text = re.sub(r"[\u00b2\u00b3\u00b9]", "", text)
    text = re.sub(r"\s+", " ", text)
    return text.strip()


def build_label_index(values):
    """Map cleaned labels to row indices."""
    index = {}
    for row_idx, value in values.items():
        label = normalize_label(value)
        if label:
            index[label] = row_idx
    return index


def get_required_index(index_map, label, context):
    """Return the required row index or raise a readable error."""
    key = normalize_label(label)
    if key not in index_map:
        raise KeyError(f"Missing label {label!r} in {context}")
    return index_map[key]


def fit_log_model(incomes, essentials):
    """Fit the log-linear power-law model used for essential expenditures."""
    incomes = np.asarray(incomes, dtype=float)
    essentials = np.asarray(essentials, dtype=float)
    mask = np.isfinite(incomes) & np.isfinite(essentials) & (incomes > 0.0) & (essentials > 0.0)
    x = np.log(incomes[mask])
    y = np.log(essentials[mask])
    A = np.column_stack([np.ones_like(x), x])
    beta, *_ = np.linalg.lstsq(A, y, rcond=None)
    fitted_log = A @ beta
    ss_res = float(np.sum((y - fitted_log) ** 2))
    ss_tot = float(np.sum((y - np.mean(y)) ** 2))
    r2 = 1.0 - ss_res / ss_tot if ss_tot > 0.0 else 1.0
    return {
        "intercept": float(beta[0]),
        "alpha": float(beta[1]),
        "scale": float(np.exp(beta[0])),
        "r2": r2,
        "income": incomes[mask],
        "essential": essentials[mask],
        "fitted": np.exp(fitted_log),
    }


def load_us_tables(xls_path):
    """Load U.S. age and region anchors from the curated spreadsheet."""
    us = pd.read_excel(xls_path, sheet_name="Expenditures (U.S.)")

    age_groups = [normalize_label(v) for v in us.iloc[1, 1:8]]
    age_income = [to_float(v) for v in us.iloc[2, 1:8]]

    regions = [normalize_label(v) for v in us.iloc[1, 8:12]]
    region_income = [to_float(v) for v in us.iloc[2, 8:12]]

    categories = [
        "Food",
        "Housing",
        "Utilities, fuel, public services",
        "Transportation",
        "Healthcare",
    ]
    row_map = build_label_index(us.iloc[:, 0])

```

Back to contents

```
    def essential_sum(col_idx):
        total = 0.0
        for category in categories:
            row_idx = get_required_index(row_map, category, "U.S. expenditure table")
            total += to_float(us.iloc[row_idx, col_idx])
        return total

    age_ess = [essential_sum(col_idx) for col_idx in range(1, 8)]
    region_ess = [essential_sum(col_idx) for col_idx in range(8, 12)]

    age_df = pd.DataFrame(
        {
            "label": age_groups,
            "income": age_income,
            "essential": age_ess,
            "source": "Age group",
        }
    )
    region_df = pd.DataFrame(
        {
            "label": regions,
            "income": region_income,
            "essential": region_ess,
            "source": "Region",
        }
    )
    return age_df, region_df


def load_uk_tables(xls_path):
    """Load U.K. quintile data, age anchors, and nation multipliers."""
    uk = pd.read_excel(xls_path, sheet_name="Expenditures (U.K.)")

    quintile_labels = [normalize_label(v) for v in uk.iloc[2, 9:14]]
    income_quintile_annual = np.array([52.0 * to_float(v) for v in uk.iloc[3, 9:14]], dtype=float)

    starts = [
        row_idx
        for row_idx, value in uk.iloc[:, 0].items()
        if normalize_label(value).startswith("Household reference person")
        and "Age" in normalize_label(value)
    ]

    needed_nonhealth = ["Food & non-alcoholic drinks", "Housing(net), fuel & power", "Transport"]
    fit_rows = []
    age_base_rows = []

    for i, start in enumerate(starts):
        end = starts[i + 1] if i + 1 < len(starts) else len(uk)
        age_label = normalize_label(uk.iloc[start, 0])
        block = uk.iloc[start:end, :7]
        idx_map = build_label_index(block.iloc[:, 0])

        nonhealth_all_week = 0.0
        for label in needed_nonhealth:
            row_idx = get_required_index(idx_map, label, f"U.K. age block starting at row {start}")
            nonhealth_all_week += to_float(uk.loc[row_idx, "Unnamed: 6"])

        health_row = get_required_index(idx_map, "Health", f"U.K. age block starting at row {start}")
        health_all_week = to_float(uk.loc[health_row, "Unnamed: 6"])
        age_base_rows.append(
            {
                "age_group": age_label,
                "nonhealth_all": 52.0 * nonhealth_all_week,
                "health_all": 52.0 * health_all_week,
            }
        )

        for q_idx, col in enumerate(["Unnamed: 1", "Unnamed: 2", "Unnamed: 3", "Unnamed: 4", "Unnamed: 5"]):
            total_week = 0.0
            ok = True
            for label in needed_nonhealth:
                row_idx = get_required_index(idx_map, label, f"U.K. age block starting at row {start}")
                value = to_float(uk.loc[row_idx, col])
                if np.isnan(value):
                    ok = False
                    break
```

Back to contents

```
                total_week += value
            if ok:
                fit_rows.append(
                    {
                        "age_group": age_label,
                        "quintile": quintile_labels[q_idx],
                        "income": income_quintile_annual[q_idx],
                        "essential_nonhealth": 52.0 * total_week,
                    }
                )

    country_cols = {normalize_label(uk.iloc[7, col_idx]): uk.columns[col_idx] for col_idx in range(9, 14)}
    label_to_row = build_label_index(uk.iloc[6:19, 8])
    country_nonhealth = {}
    for country_name, col in country_cols.items():
        total = 0.0
        for label in needed_nonhealth:
            row_idx = get_required_index(label_to_row, label, "U.K. country table")
            total += to_float(uk.loc[row_idx, col])
        country_nonhealth[country_name] = total

    country_nonhealth_mult = {
        name: value / country_nonhealth["All UK"] for name, value in country_nonhealth.items()
    }
    age_base_df = pd.DataFrame(age_base_rows)
    fit_df = pd.DataFrame(fit_rows)
    return fit_df, age_base_df, country_nonhealth_mult, float(np.mean(income_quintile_annual))


# US tax parameters for 2025
US_STD_DED = 15750
US_BRACKETS = [
    (0, 11925, 0.10),
    (11925, 48475, 0.12),
    (48475, 103350, 0.22),
    (103350, 197300, 0.24),
    (197300, 250525, 0.32),
    (250525, 626350, 0.35),
    (626350, float("inf"), 0.37),
]
SSA_WAGE_BASE_2025 = 176100


def us_federal_tax_2025(income):
    """Compute 2025 federal income tax for a simplified single-filer baseline."""
    taxable = max(0.0, income - US_STD_DED)
    tax = 0.0
    for lo, hi, rate in US_BRACKETS:
        if taxable <= lo:
            break
        tax += rate * (min(taxable, hi) - lo)
    return tax


def us_payroll_tax_2025(income):
    """Compute employee-side U.S. payroll taxes for 2025."""
    ss = 0.062 * min(income, SSA_WAGE_BASE_2025)
    medicare = 0.0145 * income
    addl = 0.009 * max(0.0, income - 200000.0)
    return ss + medicare + addl


# UK tax parameters for 2025/26
UK_ALLOW = 12570
UK_BASIC_BAND = 37700
UK_ADDL_THRESHOLD = 125140
UK_PT = 12570
UK_UEL = 50270


def uk_allowance(income):
    """Compute the tapered U.K. personal allowance."""
    if income <= 100000:
        return UK_ALLOW
    return max(0.0, UK_ALLOW - (income - 100000.0) / 2.0)


```

Back to contents

```
def uk_income_tax_2025_26(income):
    """Compute simplified U.K. income tax for the 2025/26 tax year."""
    allow = uk_allowance(income)
    taxable = max(0.0, income - allow)

    basic = min(taxable, UK_BASIC_BAND)
    rest = taxable - basic
    tax = 0.20 * basic

    if income <= UK_ADDL_THRESHOLD:
        tax += 0.40 * max(0.0, rest)
        return tax

    tax = 0.20 * UK_BASIC_BAND + 0.40 * (UK_ADDL_THRESHOLD - UK_BASIC_BAND) + 0.45 * max(
        0.0, income - UK_ADDL_THRESHOLD
    )
    return tax


def uk_nic_2025_26(income, state_pension_age=False):
    """Compute employee National Insurance contributions."""
    if state_pension_age:
        return 0.0
    main = 0.08 * max(0.0, min(income, UK_UEL) - UK_PT)
    addl = 0.02 * max(0.0, income - UK_UEL)
    return main + addl


def us_age_group(age):
    """Map a numeric age to the matching U.S. expenditure-table row."""
    if age < 25:
        return "Under 25"
    if age < 35:
        return "25-34"
    if age < 45:
        return "35-44"
    if age < 55:
        return "45-54"
    if age < 65:
        return "55-64"
    if age < 75:
        return "65-74"
    return "75 and older"


def uk_age_group(age):
    """Map a numeric age to the matching U.K. expenditure-table block."""
    if age < 30:
        return "Household reference person Age < 30"
    if age < 50:
        return "Household reference person Age 30-49"
    if age < 65:
        return "Household reference person Age 50-64"
    if age < 75:
        return "Household reference person Age 65-74"
    return "Household reference person Age > 74"


def plot_fit_diagnostics(us_points, us_fit, uk_points, uk_fit):
    """Plot calibration points together with the fitted power-law curves."""
    fig, axes = plt.subplots(1, 2, figsize=(11.4, 4.9))

    ax = axes[0]
    age_points = us_points[us_points["source"] == "Age group"]
    region_points = us_points[us_points["source"] == "Region"]
    ax.scatter(
        age_points["income"],
        age_points["essential"],
        s=42,
        color="#1f4e79",
        edgecolor="white",
        linewidth=0.5,
        label="U.S. age points",
        zorder=3,
    )
    ax.scatter(
        region_points["income"],
```

Back to contents

```
        region_points["essential"],
        s=52,
        marker="s",
        color="#8c3b2f",
        edgecolor="white",
        linewidth=0.5,
        label="U.S. region points",
        zorder=3,
    )
    income_grid = np.geomspace(us_points["income"].min() * 0.92, us_points["income"].max() * 1.08, 200)
    ax.plot(income_grid, us_fit["scale"] * income_grid ** us_fit["alpha"], color="#222222", linewidth=1.8)
    ax.set_xscale("log")
    ax.set_yscale("log")
    ax.set_xlabel("Annual income")
    ax.set_ylabel("Annual essential spending")
    ax.set_title("U.S. calibration data")
    ax.grid(True, which="both", color="#e6e6e6", linewidth=0.7)
    ax.legend(frameon=False, fontsize=8, loc="lower right")
    ax.text(
        0.03,
        0.96,
        f"$E={us_fit['scale']:.1f}I^{{{{us_fit['alpha']:.3f}}}}$\n$R^2={us_fit['r2']:.3f}$",
        transform=ax.transAxes,
        va="top",
        ha="left",
        fontsize=9,
        bbox={"facecolor": "white", "edgecolor": "none", "alpha": 0.85},
    )

    ax = axes[1]
    age_palette = ["#0f4c5c", "#2c7a7b", "#5f0f40", "#9a031e", "#bb3e03"]
    for color, (age_label, group_df) in zip(age_palette, uk_points.groupby("age_group", sort=False)):
        ax.scatter(
            group_df["income"],
            group_df["essential_nonhealth"],
            s=36,
            color=color,
            edgecolor="white",
            linewidth=0.45,
            label=age_label.replace("Household reference person ", ""),
            zorder=3,
        )
    income_grid = np.geomspace(uk_points["income"].min() * 0.92, uk_points["income"].max() * 1.08, 200)
    ax.plot(income_grid, uk_fit["scale"] * income_grid ** uk_fit["alpha"], color="#222222", linewidth=1.8)
    ax.set_xscale("log")
    ax.set_yscale("log")
    ax.set_xlabel("Annual income")
    ax.set_ylabel("Annual nonhealth essentials")
    ax.set_title("U.K. calibration data")
    ax.grid(True, which="both", color="#e6e6e6", linewidth=0.7)
    ax.legend(frameon=False, fontsize=7, loc="lower right")
    ax.text(
        0.03,
        0.96,
        f"$N={uk_fit['scale']:.1f}I^{{{{uk_fit['alpha']:.3f}}}}$\n$R^2={uk_fit['r2']:.3f}$",
        transform=ax.transAxes,
        va="top",
        ha="left",
        fontsize=9,
        bbox={"facecolor": "white", "edgecolor": "none", "alpha": 0.85},
    )

    fig.subplots_adjust(left=0.08, right=0.98, top=0.88, bottom=0.16, wspace=0.23)
    fig.savefig(os.path.join(FIG_DIR, "p1_fit_diagnostics.pdf"), bbox_inches="tight")
    plt.close(fig)


def plot_us_salary_curves(us_disposable):
    """Plot the disposable-income response curve for several U.S. ages."""
    salaries = np.linspace(20000, 200000, 240)
    ages = [22, 30, 40, 50, 60, 70]
    currency_formatter = FuncFormatter(lambda x, pos: f"{x:,.0f}")

    fig, ax = plt.subplots(figsize=(8.6, 5.2))
    line_colors = ["#0f4c5c", "#2c7a7b", "#5f0f40", "#9a031e", "#bb3e03", "#3d405b"]
    for age, color in zip(ages, line_colors):
        disposable_values = [us_disposable(salary, age, "South")[0] for salary in salaries]
```

Back to contents

```
        ax.plot(salaries, disposable_values, label=f"Age {age}", linewidth=1.8, color=color)
    ax.axhline(0.0, linewidth=1.0, color="#555555")
    ax.set_xlabel("Annual salary (USD)")
    ax.set_ylabel("Estimated disposable income (USD)")
    ax.set_title("U.S. disposable income by salary and age", pad=10)
    ax.xaxis.set_major_formatter(currency_formatter)
    ax.yaxis.set_major_formatter(currency_formatter)
    ax.grid(True, axis="y", color="#d9d9d9", linewidth=0.8)
    ax.grid(False, axis="x")
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.legend(frameon=False, ncol=2, loc="upper left")
    fig.subplots_adjust(left=0.14, right=0.98, top=0.88, bottom=0.14)
    fig.savefig(os.path.join(FIG_DIR, "p1_us_disposable_vs_salary.pdf"), bbox_inches="tight")
    plt.close(fig)


def plot_demo_profiles(demo):
    """Compare representative disposable-income estimates across profiles."""
    currency_formatter = FuncFormatter(lambda x, pos: f"{x:,.0f}")

    def fmt_money(value):
        rounded = round(float(value))
        return f"{rounded:,}"

    fig, ax = plt.subplots(figsize=(11.2, 5.8))
    labels = [f"{row['Country']} | {row['Region']} | age {int(row['Age'])}" for _, row in demo.iterrows()]
    values = demo["Disposable"].to_numpy()
    positions = np.arange(len(labels))
    colors = ["#1f4e79" if country == "US" else "#8c3b2f" for country in demo["Country"]]

    bars = ax.barh(positions, values, height=0.68, color=colors, edgecolor="none")
    ax.axvline(0.0, color="#444444", linewidth=1.0)
    ax.set_yticks(positions, labels)
    ax.invert_yaxis()
    ax.set_xlabel("Estimated disposable income")
    ax.set_title("Estimated disposable income across representative profiles", pad=12)
    ax.xaxis.set_major_formatter(currency_formatter)
    ax.xaxis.grid(True, color="#d9d9d9", linewidth=0.8)
    ax.set_axisbelow(True)
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["left"].set_visible(False)

    max_abs = max(abs(values.min()), abs(values.max()))
    ax.set_xlim(-0.16 * max_abs, 1.18 * max_abs)

    for bar, value in zip(bars, values):
        y = bar.get_y() + bar.get_height() / 2.0
        if value >= 0:
            x = value + 0.018 * max_abs
            ha = "left"
        elif abs(value) < 0.12 * max_abs:
            x = 0.02 * max_abs
            ha = "left"
        else:
            x = value - 0.018 * max_abs
            ha = "right"
        ax.text(x, y, fmt_money(value), va="center", ha=ha, fontsize=9, color="#222222")

    ax.text(
        0.985,
        0.04,
        "Blue: US   Red: UK",
        transform=ax.transAxes,
        ha="right",
        va="bottom",
        fontsize=9,
        color="#444444",
    )
    fig.subplots_adjust(left=0.29, right=0.97, top=0.88, bottom=0.12)
    fig.savefig(os.path.join(FIG_DIR, "p1_demo_profiles.pdf"), bbox_inches="tight")
    plt.close(fig)


def plot_sensitivity_boxplot(sensitivity_metrics, country_tags):
    """Plot the Monte Carlo sensitivity summary for the P1 outputs."""
```

Back to contents

```
    fig, ax = plt.subplots(figsize=(10.8, 4.9))
    box = ax.boxplot(
        sensitivity_metrics,
        patch_artist=True,
        widths=0.6,
        showfliers=False,
        medianprops={"color": "#111111", "linewidth": 1.2},
        whiskerprops={"color": "#555555", "linewidth": 1.0},
        capprops={"color": "#555555", "linewidth": 1.0},
    )
    for patch, country in zip(box["boxes"], country_tags):
        patch.set_facecolor("#cfe0f2" if country == "US" else "#f2d6cf")
        patch.set_edgecolor("#555555")
        patch.set_linewidth(1.0)

    ax.set_ylabel("Absolute change in disposable income (% of salary)")
    ax.set_title("Monte Carlo sensitivity of representative profiles", pad=10)
    ax.grid(True, axis="y", color="#e3e3e3", linewidth=0.8)
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.text(
        0.985,
        0.95,
        "1000 trials, alpha and baseline scaled within +/-10%",
        transform=ax.transAxes,
        ha="right",
        va="top",
        fontsize=8.5,
        color="#444444",
    )
    fig.subplots_adjust(left=0.12, right=0.98, top=0.87, bottom=0.16)
    fig.savefig(os.path.join(FIG_DIR, "p1_sensitivity_boxplot.pdf"), bbox_inches="tight")
    plt.close(fig)


def write_demo_table(demo):
    """Write the representative-profile table imported by LaTeX."""
    def fmt_money(value):
        rounded = round(float(value))
        return f"{rounded:,}"

    tex_lines = [
        r"\small",
        r"\setlength{\tabcolsep}{5.0pt}",
        r"\renewcommand{\arraystretch}{1.14}",
        r"\begin{tabular}{|l|l|r|r|r|r|r|}",
        r"  \hline",
        r"  \multicolumn{3}{c}{Profile} & \multicolumn{4}{c}{Annual Estimate} \\",
        r"  \hline",
        r"  Country & Region & Age & Gross Salary & Taxes & Essentials & Disposable Income \\",
        r"  \hline",
    ]
    rows = list(demo.iterrows())
    for idx, (_, row) in enumerate(rows):
        tex_lines.append(
            "  "
            + f"\\textsc{{{row['Country']}}} & {row['Region']} & {int(row['Age'])} & {fmt_money(row['Salary'])} & "
            + f"{fmt_money(row['Taxes'])} & {fmt_money(row['Essentials'])} & {fmt_money(row['Disposable'])} \\\\"
        )
        if idx != len(rows) - 1:
            tex_lines.append(r"  \hline")
    tex_lines.extend(
        [
            r"  \hline",
            r"\end{tabular}",
            r"\vspace{3pt}",
            r"\parbox{0.94\linewidth}{\footnotesize \textit{Note.} Taxes include modeled national payroll and income taxes.
      Essentials combine the baseline expenditure basket defined in the model and are reported in local currency units.
      Negative disposable income is shown with a leading minus sign.}",
        ]
    )

    with open(os.path.join(TAB_DIR, "p1_demo_table.tex"), "w", encoding="utf-8") as file:
        file.write("\n".join(tex_lines))


def main():
```

Back to contents

```
    """Run the full P1 workflow and write all LaTeX-ready outputs."""
    rng = np.random.default_rng(20260228)

    age_df_us, region_df_us = load_us_tables(DATA_PATH)
    uk_fit_df, uk_age_base_df, uk_country_nonhealth_mult, i0_uk = load_uk_tables(DATA_PATH)

    us_points = pd.concat([age_df_us, region_df_us], ignore_index=True)
    us_fit = fit_log_model(us_points["income"], us_points["essential"])
    uk_fit = fit_log_model(uk_fit_df["income"], uk_fit_df["essential_nonhealth"])

    e_region_avg = float(region_df_us["essential"].mean())
    i_region_avg = float(region_df_us["income"].mean())
    region_mult = {}
    for _, row in region_df_us.iterrows():
        region_mult[row["label"]] = (
            float(row["income"]) / i_region_avg,
            float(row["essential"]) / e_region_avg,
        )

    us_age_base = {}
    for _, row in age_df_us.iterrows():
        us_age_base[row["label"]] = (float(row["income"]), float(row["essential"]))

    uk_age_base = {}
    for _, row in uk_age_base_df.iterrows():
        uk_age_base[row["age_group"]] = (float(row["nonhealth_all"]), float(row["health_all"]))

    def us_reference(age, region):
        age_group = us_age_group(age)
        i_age, e_age = us_age_base[age_group]
        m_income, m_essential = region_mult.get(region, (1.0, 1.0))
        return i_age * m_income, e_age * m_essential

    def us_essential(salary, age, region, alpha_override=None, base_scale=1.0):
        i0, e0 = us_reference(age, region)
        alpha = us_fit["alpha"] if alpha_override is None else alpha_override
        return base_scale * e0 * (salary / i0) ** alpha

    def us_disposable(salary, age, region):
        tax = us_federal_tax_2025(salary) + us_payroll_tax_2025(salary)
        essential = us_essential(salary, age, region)
        return salary - tax - essential, tax, essential

    def uk_reference(age, country):
        age_group = uk_age_group(age)
        nonhealth_age, health_age = uk_age_base[age_group]
        nonhealth_mult = uk_country_nonhealth_mult.get(country, 1.0)
        return nonhealth_age * nonhealth_mult, health_age

    def uk_essential(salary, age, country, alpha_override=None, nonhealth_scale=1.0):
        nonhealth0, health0 = uk_reference(age, country)
        alpha = uk_fit["alpha"] if alpha_override is None else alpha_override
        nonhealth = nonhealth_scale * nonhealth0 * (salary / i0_uk) ** alpha
        return nonhealth + health0

    def uk_disposable(salary, age, country):
        tax = uk_income_tax_2025_26(salary) + uk_nic_2025_26(salary, state_pension_age=(age >= 66))
        essential = uk_essential(salary, age, country)
        return salary - tax - essential, tax, essential

    profiles = [
        {"Country": "US", "Region": "South", "Age": 22, "Salary": 35000},
        {"Country": "US", "Region": "Midwest", "Age": 30, "Salary": 70000},
        {"Country": "US", "Region": "Northeast", "Age": 45, "Salary": 120000},
        {"Country": "US", "Region": "West", "Age": 70, "Salary": 60000},
        {"Country": "UK", "Region": "England", "Age": 25, "Salary": 25000},
        {"Country": "UK", "Region": "Wales", "Age": 35, "Salary": 45000},
        {"Country": "UK", "Region": "Scotland", "Age": 55, "Salary": 80000},
        {"Country": "UK", "Region": "Northern Ireland", "Age": 78, "Salary": 35000},
    ]

    demo_rows = []
    sensitivity_metrics = []
    sensitivity_summary = []
    sensitivity_labels = []
    country_tags = []

```

Back to contents

```
664     for profile in profiles:
665         salary = float(profile["Salary"])
666         age = int(profile["Age"])
667         region = profile["Region"]
668 
669         if profile["Country"] == "US":
670             disposable, taxes, essentials = us_disposable(salary, age, region)
671             alpha_trials = us_fit["alpha"] * (1.0 + rng.uniform(-0.10, 0.10, 1000))
672             scale_trials = 1.0 + rng.uniform(-0.10, 0.10, 1000)
673             i0, e0 = us_reference(age, region)
674             trial_essentials = scale_trials * e0 * (salary / i0) ** alpha_trials
675             trial_disposable = salary - taxes - trial_essentials
676         else:
677             disposable, taxes, essentials = uk_disposable(salary, age, region)
678             alpha_trials = uk_fit["alpha"] * (1.0 + rng.uniform(-0.10, 0.10, 1000))
679             scale_trials = 1.0 + rng.uniform(-0.10, 0.10, 1000)
680             nonhealth0, health0 = uk_reference(age, region)
681             trial_essentials = scale_trials * nonhealth0 * (salary / i0_uk) ** alpha_trials + health0
682             trial_disposable = salary - taxes - trial_essentials
683 
684         sensitivity_metric = 100.0 * np.abs(trial_disposable - disposable) / salary
685         sign_flip_rate = 100.0 * np.mean((trial_disposable >= 0.0) != (disposable >= 0.0))
686         sensitivity_metrics.append(sensitivity_metric)
687         sensitivity_summary.append(
688             {
689                 "Country": profile["Country"],
690                 "Region": region,
691                 "Age": age,
692                 "MedianPctSalary": float(np.median(sensitivity_metric)),
693                 "P90PctSalary": float(np.percentile(sensitivity_metric, 90)),
694                 "SignFlipRate": float(sign_flip_rate),
695             }
696         )
697         sensitivity_labels.append(f"{profile['Country']}\n{region}\nage {age}")
698         country_tags.append(profile["Country"])
699 
700         demo_rows.append(
701             [
702                 profile["Country"],
703                 region,
704                 age,
705                 salary,
706                 taxes,
707                 essentials,
708                 disposable,
709             ]
710         )
711 
712     demo = pd.DataFrame(
713         demo_rows,
714         columns=["Country", "Region", "Age", "Salary", "Taxes", "Essentials", "Disposable"],
715     )
716     sensitivity_df = pd.DataFrame(sensitivity_summary)
717 
718     write_demo_table(demo)
719     plot_fit_diagnostics(us_points, us_fit, uk_fit_df, uk_fit)
720     plot_us_salary_curves(us_disposable)
721     plot_demo_profiles(demo)
722     plot_sensitivity_boxplot(sensitivity_metrics, country_tags)
723 
724     print("Finished")
725     print(f"US fit: log(E) = {us_fit['intercept']:.4f} + {us_fit['alpha']:.4f} log(I), R^2 = {us_fit['r2']:.4f}")
726     print(f"US power law: E = {us_fit['scale']:.2f} * I^{us_fit['alpha']:.4f}")
727     print(f"UK fit: log(N) = {uk_fit['intercept']:.4f} + {uk_fit['alpha']:.4f} log(I), R^2 = {uk_fit['r2']:.4f}")
728     print(f"UK power law: N = {uk_fit['scale']:.2f} * I^{uk_fit['alpha']:.4f}")
729     print(
730         "Sensitivity summary (median %, 90th %, sign flip %):",
731         sensitivity_df.round({"MedianPctSalary": 2, "P90PctSalary": 2, "SignFlipRate": 1}).to_dict("records"),
732     )
733     print("Wrote")
734     print(os.path.join(TAB_DIR, "p1_demo_table.tex"))
735     print(os.path.join(FIG_DIR, "p1_fit_diagnostics.pdf"))
736     print(os.path.join(FIG_DIR, "p1_us_disposable_vs_salary.pdf"))
737     print(os.path.join(FIG_DIR, "p1_demo_profiles.pdf"))
738     print(os.path.join(FIG_DIR, "p1_sensitivity_boxplot.pdf"))
739 
740 
```

Back to contents

```
if __name__ == "__main__":
    main()
```

B.3 Know the Spread

The complete Python script for Problem 2 appears below. It is the exact file used to compute the calibration constants, evaluate representative demographic profiles, and generate the final figures without any omitted sections.

Listing 3: Complete Python script for Problem 2

```
"""Problem 2 workflow for annual online sports-gambling losses.

The script keeps the modeling pipeline explicit:
1. Read the curated survey sheet.
2. Build transparent calibration constants from survey margins and market totals.
3. Evaluate representative demographic profiles under three risk tiers.
4. Write the LaTeX-ready tables and figures used in the paper.
"""

import math
import os

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
PROJECT_DIR = os.path.dirname(BASE_DIR)
DATA_PATH = os.path.join(BASE_DIR, "M3-Challenge-Problem-Data-2026.xlsx")
FIG_DIR = os.path.join(PROJECT_DIR, "figures")
TAB_DIR = os.path.join(PROJECT_DIR, "tables")

os.makedirs(FIG_DIR, exist_ok=True)
os.makedirs(TAB_DIR, exist_ok=True)

plt.rcParams.update(
    {
        "figure.facecolor": "white",
        "axes.facecolor": "white",
        "savefig.facecolor": "white",
        "font.family": "serif",
        "font.serif": ["CMU Serif", "Computer Modern Roman", "STIX Two Text", "DejaVu Serif"],
        "mathtext.fontset": "cm",
        "axes.unicode_minus": False,
        "pdf.fonttype": 42,
        "ps.fonttype": 42,
    }
)


US_HOLD_2025 = 16.96 / 166.94
UK_POP_2025 = 69_487_000.0
UK_SPORTS_ONLINE_PARTICIPATION = 0.10
UK_REMOTE_BETTING_GGR_2025 = 2_630.69e6
GBP_PER_USD_2025 = 1.0 / 1.31325
OUTCOME_COEFF = 1.25

RISK_LEVELS = {
    "Low": 0.65,
    "Moderate": 1.00,
    "High": 1.55,
}


def geometric_mean(values):
    """Combine marginal multipliers without letting one dimension dominate."""
    values = np.asarray(values, dtype=float)
    return float(np.prod(values) ** (1.0 / len(values)))


def fmt_money(value):
    """Format a local-currency amount for tables."""
    rounded = round(float(value))
```

Back to contents

```
    if rounded < 0:
        return f"-{abs(rounded):,}"
    return f"{rounded:,}"


def load_survey_sheet():
    """Load the curated online sports-betting survey data."""
    return pd.read_excel(DATA_PATH, sheet_name="Online Sports Betting Personal ", header=None)


def build_calibration(survey):
    """Extract country baselines and demographic multipliers from the survey."""
    column_groups = {
        "overall": {"Total": 1},
        "gender": {"Male": 3, "Female": 4},
        "age": {"18-34": 5, "35-49": 6, "50-64": 7, "65+": 8},
        "education": {"No college": 9, "B.A. or higher": 10},
    }

    def pct(row_idx, col_idx):
        return float(survey.iloc[row_idx, col_idx]) / 100.0

    base_active = pct(3, 1) * pct(16, 1)
    deposit_weights = {30: 1.0, 31: 4.0, 32: 12.0, 33: 52.0}
    base_deposit = sum(pct(row_idx, 1) * weight for row_idx, weight in deposit_weights.items())
    base_big_100 = pct(40, 1)
    base_big_500 = pct(41, 1)
    base_delta = pct(19, 1) - pct(18, 1)

    active_rel = {}
    behavior_rel = {}
    outcome_mult = {}

    for dim_name, mapping in column_groups.items():
        if dim_name == "overall":
            continue
        active_rel[dim_name] = {}
        behavior_rel[dim_name] = {}
        outcome_mult[dim_name] = {}
        for category, col_idx in mapping.items():
            active_prob = pct(3, col_idx) * pct(16, col_idx)
            active_rel[dim_name][category] = active_prob / base_active

            deposit_score = sum(pct(row_idx, col_idx) * weight for row_idx, weight in deposit_weights.items())
            size_score = math.sqrt((pct(40, col_idx) / base_big_100) * (pct(41, col_idx) / base_big_500))
            behavior_rel[dim_name][category] = (deposit_score / base_deposit) * size_score

            delta = pct(19, col_idx) - pct(18, col_idx)
            outcome_mult[dim_name][category] = max(0.70, 1.0 + OUTCOME_COEFF * (delta - base_delta))

    wager_bucket_midpoints = np.array([50.0, 300.0, 1000.0], dtype=float)
    sport_rows = [14, 15, 16, 17, 18]
    sport_labels = [str(survey.iloc[row_idx, 12]) for row_idx in sport_rows]
    sport_shares = np.array(
        [survey.iloc[row_idx, 13:16].astype(float).to_numpy() / 100.0 for row_idx in sport_rows],
        dtype=float,
    )
    monthly_wagers = sport_shares @ wager_bucket_midpoints
    us_monthly_wager = float(np.mean(monthly_wagers))

    freq_rows = list(range(2, 10))
    freq_labels = [str(survey.iloc[row_idx, 12]) for row_idx in freq_rows]
    freq_weights = np.array([0.0, 0.5, 1.0, 3.0, 6.0, 14.0, 26.0, 35.0], dtype=float)
    us_freq_shares = np.array([pct(row_idx, 13) for row_idx in freq_rows], dtype=float)
    uk_freq_shares = np.array([pct(row_idx, 14) for row_idx in freq_rows], dtype=float)
    us_freq_index = float(np.dot(us_freq_shares, freq_weights))
    uk_freq_index = float(np.dot(uk_freq_shares, freq_weights))

    us_active_loss = us_monthly_wager * 12.0 * US_HOLD_2025
    # Convert the behavior-scaled U.S. loss anchor into GBP before combining
    # it with the revenue-based U.K. anchor.
    uk_behavior_anchor = us_active_loss * (uk_freq_index / us_freq_index) * GBP_PER_USD_2025
    uk_revenue_anchor = UK_REMOTE_BETTING_GGR_2025 / (UK_POP_2025 * UK_SPORTS_ONLINE_PARTICIPATION)
    uk_active_loss = math.sqrt(uk_behavior_anchor * uk_revenue_anchor)

    return {
        "column_groups": column_groups,
```

Back to contents

```
        "base_active_us": base_active,
        "base_active_uk": UK_SPORTS_ONLINE_PARTICIPATION,
        "active_rel": active_rel,
        "behavior_rel": behavior_rel,
        "outcome_mult": outcome_mult,
        "sport_labels": sport_labels,
        "sport_shares": sport_shares,
        "monthly_wagers": monthly_wagers,
        "us_monthly_wager": us_monthly_wager,
        "freq_labels": freq_labels,
        "us_freq_shares": us_freq_shares,
        "uk_freq_shares": uk_freq_shares,
        "us_freq_index": us_freq_index,
        "uk_freq_index": uk_freq_index,
        "us_active_loss": us_active_loss,
        "uk_behavior_anchor": uk_behavior_anchor,
        "uk_revenue_anchor": uk_revenue_anchor,
        "uk_active_loss": uk_active_loss,
    }


def combined_multiplier(lookup, profile):
    """Apply the geometric-mean shrinkage rule to a full profile."""
    values = [
        lookup["gender"][profile["gender"]],
        lookup["age"][profile["age_group"]],
        lookup["education"][profile["education"]],
    ]
    return geometric_mean(values)


def evaluate_profile(calibration, country, gender, age_group, education, risk_label):
    """Evaluate the fitted P2 model for one demographic profile."""
    profile = {
        "gender": gender,
        "age_group": age_group,
        "education": education,
    }
    base_active = calibration["base_active_us"] if country == "US" else calibration["base_active_uk"]
    base_loss = calibration["us_active_loss"] if country == "US" else calibration["uk_active_loss"]

    p_active = min(0.95, base_active * combined_multiplier(calibration["active_rel"], profile))
    behavior = combined_multiplier(calibration["behavior_rel"], profile)
    outcome = combined_multiplier(calibration["outcome_mult"], profile)
    risk = RISK_LEVELS[risk_label]

    expected_loss_active = base_loss * behavior * outcome * risk
    expected_loss_population = p_active * expected_loss_active

    return {
        "Country": country,
        "Gender": gender,
        "Age group": age_group,
        "Education": education,
        "Risk": risk_label,
        "Active probability": p_active,
        "Conditional net": -expected_loss_active,
        "Population net": -expected_loss_population,
        "Behavior multiplier": behavior,
        "Outcome multiplier": outcome,
    }


def write_profile_table(df):
    """Write the representative-profile table imported by LaTeX."""
    lines = [
        r"\small",
        r"\setlength{\tabcolsep}{4.8pt}",
        r"\renewcommand{\arraystretch}{1.14}",
        r"\begin{tabular}{|l|l|l|l|l|r|r|r|}",
        r"  \hline",
        r"  Country & Gender & Age group & Education & Risk & $p_{\mathrm{act}}$ & $\mathbb{E}[Y\mid \mathrm{active}]$ & $\
     mathbb{E}[Y]$ \\",
        r"  \hline",
    ]
    rows = list(df.iterrows())
    for idx, (_, row) in enumerate(rows):
```

Back to contents

```
        lines.append(
            "  "
            + f"\\textsc{{{row['Country']}}} & {row['Gender']} & {row['Age group']} & {row['Education']} & {row['Risk']} &
 "
            + f"{row['Active probability']:.3f} & {fmt_money(row['Conditional net'])} & {fmt_money(row['Population net'])}
 \\\\"
        )
        if idx != len(rows) - 1:
            lines.append(r"  \hline")
    lines.extend(
        [
            r"  \hline",
            r"\end{tabular}",
            r"\vspace{3pt}",
            r"\parbox{0.96\linewidth}{\footnotesize \textit{Note.} $Y$ is the predicted annual net gambling outcome in
 local currency, so a leading minus sign means an expected loss. The conditional column assumes the person is already
 an active online sports bettor; the final column multiplies that loss by the model's active-bettor probability for the
  same demographic group.}",
        ]
    )
    with open(os.path.join(TAB_DIR, "p2_profile_table.tex"), "w", encoding="utf-8") as handle:
        handle.write("\n".join(lines))


def write_calibration_table(calibration):
    """Write the calibration constants table imported by LaTeX."""
    lines = [
        r"\small",
        r"\setlength{\tabcolsep}{4.8pt}",
        r"\renewcommand{\arraystretch}{1.14}",
        r"\begin{tabular}{|l|l|l|r|}",
        r"  \hline",
        r"  Quantity & Meaning & Unit & Value \\",
        r"  \hline",
        r"  $W_{\mathrm{US}}$ & mean monthly U.S. sports wagering per active bettor & USD per month & "
        + f"{calibration['us_monthly_wager']:.1f} \\\\",
        r"  \hline",
        r"  $h_{\mathrm{US}}$ & 2025 U.S. sportsbook hold rate & share of handle & " + f"{US_HOLD_2025:.3f} \\\\",
        r"  \hline",
        r"  $L_{\mathrm{US}}^{(0)}$ & baseline annual U.S. loss for an active bettor & USD per year & "
        + f"{calibration['us_active_loss']:.1f} \\\\",
        r"  \hline",
        r"  $F_{\mathrm{US}}$ & U.S. monthly online-betting frequency index & score & "
        + f"{calibration['us_freq_index']:.3f} \\\\",
        r"  \hline",
        r"  $F_{\mathrm{UK}}$ & U.K. monthly online-betting frequency index & score & "
        + f"{calibration['uk_freq_index']:.3f} \\\\",
        r"  \hline",
        r"  $L_{\mathrm{UK,beh}}$ & U.K. behavior-scaled active-bettor loss anchor & GBP per year & "
        + f"{calibration['uk_behavior_anchor']:.1f} \\\\",
        r"  \hline",
        r"  $L_{\mathrm{UK,rev}}$ & U.K. revenue-consistency loss anchor & GBP per year & "
        + f"{calibration['uk_revenue_anchor']:.1f} \\\\",
        r"  \hline",
        r"  $L_{\mathrm{UK}}^{(0)}$ & geometric-mean U.K. active-bettor baseline & GBP per year & "
        + f"{calibration['uk_active_loss']:.1f} \\\\",
        r"  \hline",
        r"  $p_{\mathrm{act,US}}^{(0)}$ & baseline U.S. active bettor probability & share of adults & "
        + f"{calibration['base_active_us']:.3f} \\\\",
        r"  \hline",
        r"  $p_{\mathrm{act,UK}}^{(0)}$ & baseline U.K. active bettor probability & share of adults & "
        + f"{calibration['base_active_uk']:.3f} \\\\",
        r"  \hline",
        r"\end{tabular}",
        r"\vspace{3pt}",
        r"\parbox{0.95\linewidth}{\footnotesize \textit{Note.} $F_{\mathrm{US}}$ and $F_{\mathrm{UK}}$ are weighted monthly
 betting-frequency scores built from the survey's broad frequency categories. $L_{\mathrm{UK,beh}}$ scales the U.S.
 baseline by relative betting frequency and then converts the result from USD to GBP using the 2025 average exchange
 rate assumption. $L_{\mathrm{UK,rev}}$ comes from official U.K. revenue totals divided by a conservative count of
 active bettors.}",
    ]
    with open(os.path.join(TAB_DIR, "p2_calibration_table.tex"), "w", encoding="utf-8") as handle:
        handle.write("\n".join(lines))


def plot_input_data(calibration):
    """Plot the raw inputs that anchor the scale of the P2 model."""
```

Back to contents

```
    fig, axes = plt.subplots(1, 2, figsize=(11.4, 4.9))

    colors = ["#d8b56a", "#7ea07a", "#487a8a"]
    bucket_names = ["$1$-$100", "$101$-$500", "$500+$"]
    x = np.arange(len(calibration["sport_labels"]))
    bottom = np.zeros(len(calibration["sport_labels"]))
    for idx in range(3):
        heights = calibration["sport_shares"][:, idx] * 100.0
        axes[0].bar(
            x,
            heights,
            bottom=bottom,
            color=colors[idx],
            edgecolor="white",
            width=0.72,
            label=bucket_names[idx],
        )
        bottom += heights
    axes[0].set_xticks(x, calibration["sport_labels"])
    axes[0].set_ylim(0, 100)
    axes[0].set_ylabel("Share of respondents (%)")
    axes[0].set_title("U.S. monthly sports wagering buckets", pad=10)
    axes[0].tick_params(axis="x", labelsize=9)
    axes[0].grid(True, axis="y", color="#e5e5e5", linewidth=0.8)

    x2 = np.arange(len(calibration["freq_labels"]))
    axes[1].plot(x2, calibration["us_freq_shares"] * 100.0, color="#9d5c63", marker="o", linewidth=2.0, label="U.S.")
    axes[1].plot(x2, calibration["uk_freq_shares"] * 100.0, color="#3b6f9c", marker="s", linewidth=2.0, label="U.K.")
    axes[1].set_xticks(x2, calibration["freq_labels"], rotation=22, ha="right")
    axes[1].set_ylabel("Share of respondents (%)")
    axes[1].set_title("Online betting frequency by country", pad=8)
    axes[1].legend(frameon=False)
    axes[1].grid(True, axis="y", color="#e5e5e5", linewidth=0.8)

    for ax in axes:
        ax.spines["top"].set_visible(False)
        ax.spines["right"].set_visible(False)

    axes[0].legend(
        frameon=False,
        ncol=3,
        loc="upper center",
        bbox_to_anchor=(0.5, -0.22),
        fontsize=8,
        columnspacing=1.4,
        handletextpad=0.5,
    )
    fig.subplots_adjust(left=0.07, right=0.985, bottom=0.30, top=0.88, wspace=0.26)
    fig.savefig(os.path.join(FIG_DIR, "p2_input_data.pdf"), bbox_inches="tight")
    plt.close(fig)


def plot_heatmaps(calibration):
    """Plot population-wide loss heatmaps with an external colorbar."""
    age_order = ["18-34", "35-49", "50-64", "65+"]
    gender_order = ["Female", "Male"]
    education_order = ["No college", "B.A. or higher"]
    country_order = ["US", "UK"]

    fig, axes = plt.subplots(2, 2, figsize=(9.6, 6.4))
    vmin = 0.0
    vmax = 0.0
    matrices = {}
    for country in country_order:
        for education in education_order:
            matrix = np.zeros((len(gender_order), len(age_order)))
            for i, gender in enumerate(gender_order):
                for j, age_group in enumerate(age_order):
                    row = evaluate_profile(calibration, country, gender, age_group, education, "Moderate")
                    matrix[i, j] = -row["Population net"]
            matrices[(country, education)] = matrix
            vmax = max(vmax, float(matrix.max()))

    for row_idx, country in enumerate(country_order):
        for col_idx, education in enumerate(education_order):
            ax = axes[row_idx, col_idx]
            matrix = matrices[(country, education)]
```

Back to contents

```
            im = ax.imshow(matrix, cmap="YlOrBr", aspect="auto", vmin=vmin, vmax=vmax)
            ax.set_xticks(range(len(age_order)), age_order)
            ax.set_yticks(range(len(gender_order)), gender_order if col_idx == 0 else ["", ""])
            ax.set_title(f"{country}, {education}", pad=6)
            ax.tick_params(axis="both", labelsize=9)
            for i in range(matrix.shape[0]):
                for j in range(matrix.shape[1]):
                    ax.text(j, i, f"{matrix[i, j]:.0f}", ha="center", va="center", fontsize=8, color="#1f1f1f")
            if row_idx == len(country_order) - 1:
                ax.set_xlabel("Age group")
            if col_idx == 0:
                ax.set_ylabel("Gender")
            ax.spines["top"].set_visible(False)
            ax.spines["right"].set_visible(False)

    cax = fig.add_axes([0.91, 0.16, 0.024, 0.66])
    cbar = fig.colorbar(im, cax=cax)
    cbar.set_label("Expected annual loss per adult")
    fig.suptitle("Predicted population-wide annual losses at moderate risk", y=0.98)
    fig.subplots_adjust(left=0.11, right=0.88, bottom=0.14, top=0.90, wspace=0.18, hspace=0.30)
    fig.savefig(os.path.join(FIG_DIR, "p2_loss_heatmaps.pdf"), bbox_inches="tight")
    plt.close(fig)


def plot_profile_bars(profile_df):
    """Plot representative conditional and population losses without label overlap."""
    colors = {"US": "#9d5c63", "UK": "#3b6f9c"}
    fig, axes = plt.subplots(1, 2, figsize=(11.4, 5.3), sharey=True)

    labels = [
        f"{row['Country']} {row['Gender'][0]}-{row['Age group']}\n{row['Education'].replace('B.A. or higher', 'BA+')}\n{row
      ['Risk']}"
        for _, row in profile_df.iterrows()
    ]
    conditional = -profile_df["Conditional net"].to_numpy()
    population = -profile_df["Population net"].to_numpy()
    bar_colors = [colors[country] for country in profile_df["Country"]]
    y = np.arange(len(labels))

    axes[0].barh(y, conditional, color=bar_colors, edgecolor="white", height=0.72)
    axes[0].set_yticks(y, labels)
    axes[0].invert_yaxis()
    axes[0].set_xlabel("Local currency units")
    axes[0].set_title("Annual loss given active betting", pad=8)
    axes[0].grid(True, axis="x", color="#e5e5e5", linewidth=0.8)

    axes[1].barh(y, population, color=bar_colors, edgecolor="white", height=0.72)
    axes[1].set_xlabel("Local currency units")
    axes[1].set_title("Annual loss for a representative adult", pad=8)
    axes[1].grid(True, axis="x", color="#e5e5e5", linewidth=0.8)
    axes[1].tick_params(axis="y", labelleft=False)

    conditional_max = float(np.max(conditional))
    population_max = float(np.max(population))
    axes[0].set_xlim(0.0, 1.18 * conditional_max)
    axes[1].set_xlim(0.0, 1.22 * population_max)
    for idx, value in enumerate(conditional):
        axes[0].text(value + 0.02 * conditional_max, y[idx], f"{value:.0f}", va="center", ha="left", fontsize=8)
    for idx, value in enumerate(population):
        axes[1].text(value + 0.03 * population_max, y[idx], f"{value:.0f}", va="center", ha="left", fontsize=8)

    for ax in axes:
        ax.spines["top"].set_visible(False)
        ax.spines["right"].set_visible(False)

    fig.subplots_adjust(left=0.29, right=0.985, bottom=0.14, top=0.87, wspace=0.18)
    fig.savefig(os.path.join(FIG_DIR, "p2_profile_losses.pdf"), bbox_inches="tight")
    plt.close(fig)


def run_sensitivity(calibration, profiles):
    """Propagate uncertainty through the representative-profile outputs."""
    rng = np.random.default_rng(20260228)
    base_profile_rows = [
        evaluate_profile(
            calibration,
            row["Country"],
```

Back to contents

```
            row["Gender"],
            row["Age group"],
            row["Education"],
            row["Risk"],
        )
        for _, row in profiles.iterrows()
    ]
    labels = [
        f"{row['Country']} {row['Gender'][0]}-{row['Age group']}\n{row['Risk']}"
        for _, row in profiles.iterrows()
    ]
    data = []
    summary = []

    for idx, (_, row) in enumerate(profiles.iterrows()):
        base_population_loss = -base_profile_rows[idx]["Population net"]
        changes = []
        for _ in range(1000):
            high_bucket_mid = 1000.0 * (1.0 + rng.uniform(-0.10, 0.10))
            us_hold = US_HOLD_2025 * (1.0 + rng.uniform(-0.10, 0.10))
            uk_participation = UK_SPORTS_ONLINE_PARTICIPATION * (1.0 + rng.uniform(-0.10, 0.10))
            outcome_coeff = OUTCOME_COEFF * (1.0 + rng.uniform(-0.10, 0.10))

            sport_shares = calibration["sport_shares"]
            alt_monthly_wager = float(np.mean(sport_shares @ np.array([50.0, 300.0, high_bucket_mid], dtype=float)))
            alt_us_base = alt_monthly_wager * 12.0 * us_hold
            alt_uk_behavior = alt_us_base * (calibration["uk_freq_index"] / calibration["us_freq_index"])
            alt_uk_revenue = UK_REMOTE_BETTING_GGR_2025 / (UK_POP_2025 * uk_participation)
            alt_uk_base = math.sqrt(alt_uk_behavior * alt_uk_revenue)

            profile = {
                "gender": row["Gender"],
                "age_group": row["Age group"],
                "education": row["Education"],
                }

            delta_mult = []
            for dim_name, category in [
                ("gender", profile["gender"]),
                ("age", profile["age_group"]),
                ("education", profile["education"]),
            ]:
                base_mult = calibration["outcome_mult"][dim_name][category]
                implied_delta = (base_mult - 1.0) / OUTCOME_COEFF
                alt_mult = max(0.70, 1.0 + outcome_coeff * implied_delta)
                delta_mult.append(alt_mult)
            alt_outcome = geometric_mean(delta_mult)
            behavior = combined_multiplier(calibration["behavior_rel"], profile)
            active = (
                calibration["base_active_us"]
                if row["Country"] == "US"
                else calibration["base_active_uk"]
            ) * combined_multiplier(calibration["active_rel"], profile)
            active = min(0.95, active)
            base_loss = alt_us_base if row["Country"] == "US" else alt_uk_base
            population_loss = active * base_loss * behavior * alt_outcome * RISK_LEVELS[row["Risk"]]
            changes.append(100.0 * abs(population_loss - base_population_loss) / max(base_population_loss, 1e-9))

        change_array = np.asarray(changes, dtype=float)
        data.append(change_array)
        summary.append(
            {
                "Profile": labels[idx],
                "MedianPct": float(np.median(change_array)),
                "P90Pct": float(np.percentile(change_array, 90.0)),
            }
        )

    return labels, data, pd.DataFrame(summary)


def plot_sensitivity_boxplot(labels, data):
    """Plot the Monte Carlo sensitivity summary for the P2 outputs."""
    fig, ax = plt.subplots(figsize=(10.2, 4.3))
    box = ax.boxplot(data, patch_artist=True, widths=0.62, showfliers=False)
    palette = ["#cfb36b", "#9d5c63", "#7ea07a", "#3b6f9c", "#c9896a", "#6d8b74"]
    for idx, patch in enumerate(box["boxes"]):
```

Back to contents

```
        patch.set_facecolor(palette[idx % len(palette)])
        patch.set_edgecolor("#555555")
        patch.set_linewidth(1.0)
    for part_name in ["whiskers", "caps", "medians"]:
        for artist in box[part_name]:
            artist.set_color("#555555")
            artist.set_linewidth(1.0)

    ax.set_xticklabels(labels, rotation=16, ha="right")
    ax.set_ylabel("Absolute change in expected loss (%)")
    ax.set_title("Monte Carlo sensitivity of representative P2 profiles", pad=8)
    ax.grid(True, axis="y", color="#e5e5e5", linewidth=0.8)
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    fig.subplots_adjust(left=0.08, right=0.985, bottom=0.28, top=0.84)
    fig.savefig(os.path.join(FIG_DIR, "p2_sensitivity_boxplot.pdf"), bbox_inches="tight")
    plt.close(fig)


def main():
    """Run the full P2 workflow and write all LaTeX-ready outputs."""
    survey = load_survey_sheet()
    calibration = build_calibration(survey)

    profiles = pd.DataFrame(
        [
            {"Country": "US", "Gender": "Male", "Age group": "18-34", "Education": "No college", "Risk": "High"},
            {"Country": "US", "Gender": "Female", "Age group": "35-49", "Education": "B.A. or higher", "Risk": "Moderate"},
            {"Country": "US", "Gender": "Male", "Age group": "50-64", "Education": "B.A. or higher", "Risk": "Low"},
            {"Country": "UK", "Gender": "Male", "Age group": "35-49", "Education": "B.A. or higher", "Risk": "High"},
            {"Country": "UK", "Gender": "Female", "Age group": "18-34", "Education": "No college", "Risk": "Moderate"},
            {"Country": "UK", "Gender": "Female", "Age group": "65+", "Education": "B.A. or higher", "Risk": "Low"},
        ]
    )

    profile_rows = [
        evaluate_profile(
            calibration,
            row["Country"],
            row["Gender"],
            row["Age group"],
            row["Education"],
            row["Risk"],
        )
        for _, row in profiles.iterrows()
    ]
    profile_df = pd.DataFrame(profile_rows)

    write_profile_table(profile_df)
    write_calibration_table(calibration)
    plot_input_data(calibration)
    plot_heatmaps(calibration)
    plot_profile_bars(profile_df)
    sens_labels, sens_data, sens_summary = run_sensitivity(calibration, profiles)
    plot_sensitivity_boxplot(sens_labels, sens_data)

    print("Finished")
    print(f"US monthly wager baseline: {calibration['us_monthly_wager']:.2f}")
    print(f"US active-loss baseline: {calibration['us_active_loss']:.2f}")
    print(f"UK behavior anchor: {calibration['uk_behavior_anchor']:.2f}")
    print(f"UK revenue anchor: {calibration['uk_revenue_anchor']:.2f}")
    print(f"UK active-loss baseline: {calibration['uk_active_loss']:.2f}")
    print(
        "Representative profiles:",
        profile_df[
            ["Country", "Gender", "Age group", "Education", "Risk", "Active probability", "Conditional net", "Population
 net"]
        ].round({"Active probability": 3, "Conditional net": 1, "Population net": 1}).to_dict("records"),
    )
    print(
        "Sensitivity summary:",
        sens_summary.round({"MedianPct": 2, "P90Pct": 2}).to_dict("records"),
    )


if __name__ == "__main__":
    main()
```

Back to contents

B.4 Don't Break the Bank

The complete Python script for Problem 3 appears below. It is the exact file used to combine Problems 1 and 2, compute the public-facing harm metrics, and generate the final P3 tables and figures.

Listing 4: Complete Python script for Problem 3

```
"""Problem 3 workflow for translating losses into financial harm.

The script composes the calibrated functions from Problems 1 and 2, evaluates
public-facing harm metrics, and writes the figures and tables imported by the
paper.
"""

import math
import os
import sys

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from matplotlib.ticker import FuncFormatter, MaxNLocator

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
PROJECT_DIR = os.path.dirname(BASE_DIR)
FIG_DIR = os.path.join(PROJECT_DIR, "figures")
TAB_DIR = os.path.join(PROJECT_DIR, "tables")

os.makedirs(FIG_DIR, exist_ok=True)
os.makedirs(TAB_DIR, exist_ok=True)

sys.path.append(BASE_DIR)

import p1_disposable_income as p1  # noqa: E402
import p2_online_sports_loss as p2  # noqa: E402

plt.rcParams.update(
    {
        "figure.facecolor": "white",
        "axes.facecolor": "white",
        "savefig.facecolor": "white",
        "font.family": "serif",
        "font.serif": ["CMU Serif", "Computer Modern Roman", "STIX Two Text", "DejaVu Serif"],
        "mathtext.fontset": "cm",
        "axes.unicode_minus": False,
        "pdf.fonttype": 42,
        "ps.fonttype": 42,
    }
)


SAVE_RATE = 0.04
DEBT_RATE = 0.15
CV_BY_RISK = {"Low": 0.60, "Moderate": 1.00, "High": 1.60}

TIER_COLORS = {
    "Manageable": "#7ea07a",
    "Strained": "#cfb36b",
    "Precarious": "#c9896a",
    "Critical": "#9d5c63",
}


def age_group_for_p2(age):
    """Map an integer age to the age bucket used in Problem 2."""
    if age < 35:
        return "18-34"
    if age < 50:
        return "35-49"
    if age < 65:
        return "50-64"
    return "65+"


def annuity_factor(rate, years=10):
    """Return the annuity factor used in the ten-year wealth-gap scenario."""
    return ((1.0 + rate) ** years - 1.0) / rate
```

Back to contents

```
71
72
73 def normal_cdf(value):
74     """Evaluate the standard normal CDF without a SciPy dependency."""
75     return 0.5 * (1.0 + math.erf(value / math.sqrt(2.0)))
76
77
78 def debt_probability(mean_loss, cushion, risk_label):
79     """Estimate the one-year probability that realized losses exceed cushion."""
80     if cushion <= 0.0:
81         return 1.0
82     if mean_loss <= 0.0:
83         return 0.0
84
85     cv = CV_BY_RISK[risk_label]
86     sigma = math.sqrt(math.log(1.0 + cv * cv))
87     mu = math.log(mean_loss) - 0.5 * sigma * sigma
88     z_score = (math.log(cushion) - mu) / sigma
89     return 1.0 - normal_cdf(z_score)
90
91
92 def wealth_gap_10(loss, cushion, save_rate=SAVE_RATE, debt_rate=DEBT_RATE):
93     """Convert repeated annual losses into a ten-year wealth gap."""
94     if loss <= cushion:
95         return loss * annuity_factor(save_rate)
96     return cushion * annuity_factor(save_rate) + (loss - cushion) * annuity_factor(debt_rate)
97
98
99 def recovery_months(loss, cushion):
100     """Express one year of losses in months of annual discretionary cushion."""
101     if cushion <= 0.0:
102         return math.inf
103     return 12.0 * loss / cushion
104
105
106 def bankroll_tier(months, debt_prob, cushion):
107     """Translate the quantitative outputs into the four public-facing tiers."""
108     if cushion <= 0.0 or months >= 12.0 or debt_prob >= 0.50:
109         return "Critical"
110     if months >= 3.0 or debt_prob >= 0.20:
111         return "Precarious"
112     if months >= 1.0 or debt_prob >= 0.05:
113         return "Strained"
114     return "Manageable"
115
116
117 def build_disposable_functions():
118     """Rebuild the calibrated disposable-income functions from Problem 1."""
119     age_df_us, region_df_us = p1.load_us_tables(p1.DATA_PATH)
120     uk_fit_df, uk_age_base_df, uk_country_nonhealth_mult, i0_uk = p1.load_uk_tables(p1.DATA_PATH)
121     us_points = pd.concat([age_df_us, region_df_us], ignore_index=True)
122     us_fit = p1.fit_log_model(us_points["income"], us_points["essential"])
123     uk_fit = p1.fit_log_model(uk_fit_df["income"], uk_fit_df["essential_nonhealth"])
124
125     e_region_avg = float(region_df_us["essential"].mean())
126     i_region_avg = float(region_df_us["income"].mean())
127     region_mult = {
128         row["label"]: (
129             float(row["income"]) / i_region_avg,
130             float(row["essential"]) / e_region_avg,
131         )
132         for _, row in region_df_us.iterrows()
133     }
134     us_age_base = {row["label"]: (float(row["income"]), float(row["essential"])) for _, row in age_df_us.iterrows()}
135     uk_age_base = {
136         row["age_group"]: (float(row["nonhealth_all"]), float(row["health_all"]))
137         for _, row in uk_age_base_df.iterrows()
138     }
139
140     def us_reference(age, region):
141         age_group = p1.us_age_group(age)
142         i_age, e_age = us_age_base[age_group]
143         income_mult, essential_mult = region_mult.get(region, (1.0, 1.0))
144         return i_age * income_mult, e_age * essential_mult
145
146     def us_disposable(salary, age, region):
147         i0, e0 = us_reference(age, region)
```

Back to contents

```
        essential = e0 * (salary / i0) ** us_fit["alpha"]
        taxes = p1.us_federal_tax_2025(salary) + p1.us_payroll_tax_2025(salary)
        return salary - taxes - essential

    def uk_reference(age, country):
        age_group = p1.uk_age_group(age)
        nonhealth_age, health_age = uk_age_base[age_group]
        return nonhealth_age * uk_country_nonhealth_mult.get(country, 1.0), health_age

    def uk_disposable(salary, age, country):
        nonhealth0, health0 = uk_reference(age, country)
        essential = nonhealth0 * (salary / i0_uk) ** uk_fit["alpha"] + health0
        taxes = p1.uk_income_tax_2025_26(salary) + p1.uk_nic_2025_26(
            salary,
            state_pension_age=(age >= 66),
        )
        return salary - taxes - essential

    return us_disposable, uk_disposable


def evaluate_profile(calibration, us_disposable, uk_disposable, profile):
    """Combine Problems 1 and 2 for one full bettor profile."""
    age_group = age_group_for_p2(profile["Age"])
    if profile["Country"] == "US":
        disposable = us_disposable(profile["Salary"], profile["Age"], profile["Region"])
    else:
        disposable = uk_disposable(profile["Salary"], profile["Age"], profile["Region"])

    p2_row = p2.evaluate_profile(
        calibration,
        profile["Country"],
        profile["Gender"],
        age_group,
        profile["Education"],
        profile["Risk"],
    )
    annual_loss = -float(p2_row["Conditional net"])
    cushion = max(float(disposable), 0.0)
    months = recovery_months(annual_loss, cushion)
    debt_prob = debt_probability(annual_loss, cushion, profile["Risk"])
    gap_10 = wealth_gap_10(annual_loss, cushion)
    tier = bankroll_tier(months, debt_prob, cushion)

    return {
        **profile,
        "Age group": age_group,
        "Disposable": float(disposable),
        "Loss": annual_loss,
        "Cushion": cushion,
        "Recovery months": months,
        "Debt probability": debt_prob,
        "Gap 10y": gap_10,
        "Tier": tier,
    }


def format_money(value):
    """Format a local-currency amount for LaTeX tables."""
    rounded = round(float(value))
    if rounded < 0:
        return f"-{abs(rounded):,}"
    return f"{rounded:,}"


def format_months(value):
    """Format the months-of-cushion metric for LaTeX tables."""
    if not math.isfinite(value):
        return "No cushion"
    return f"{value:.1f}"


def write_profile_table(df):
    """Write the representative-profile table imported by LaTeX."""
    region_display = {"Northern Ireland": "N.\\ Ireland"}
    lines = [
        r"\small",
```

Back to contents

```
        r"\setlength{\tabcolsep}{4.0pt}",
        r"\renewcommand{\arraystretch}{1.14}",
        r"\begin{tabular}{|l|l|l|r|l|r|r|r|r|l|}",
        r"  \hline",
        r"  Country & Region & Demo & Salary & Risk & Disposable & Loss & Months & Debt risk & Tier \\",
        r"  \hline",
    ]
    rows = list(df.iterrows())
    for idx, (_, row) in enumerate(rows):
        demo = f"{row['Gender'][0]}-{int(row['Age'])}-{row['Education short']}"
        region = region_display.get(row["Region"], row["Region"])
        lines.append(
            "  "
            + f"\\textsc{{{row['Country']}}} & {region} & {demo} & {round(float(row['Salary'])):,} & {row['Risk']} & "
            + f"{format_money(row['Disposable'])} & {format_money(row['Loss'])} & {format_months(row['Recovery months'])} &
      "
            + f"{100.0 * row['Debt probability']:.1f}\\% & {row['Tier']} \\\\"
        )
        if idx != len(rows) - 1:
            lines.append(r"  \hline")
    lines.extend(
        [
            r"  \hline",
            r"\end{tabular}",
            r"\vspace{3pt}",
            r"\parbox{0.98\linewidth}{\footnotesize \textit{Note.} Demo abbreviations use gender, age, and education. NC
     denotes no college and BA+ denotes a bachelor's degree or higher. A leading minus sign in Disposable means the
     household is already short of money before gambling. Losses are the conditional annual gambling losses for active
     bettors from Problem 2, and Months measures how many months of annual disposable-income cushion are absorbed by one
     year of gambling.}",
        ]
    )
    with open(os.path.join(TAB_DIR, "p3_profile_table.tex"), "w", encoding="utf-8") as handle:
        handle.write("\n".join(lines))


def plot_profile_impacts(df):
    """Plot immediate and long-run harm without overlapping profile labels."""
    labels = [
        f"{row['Country']} {row['Region']}\n{row['Gender'][0]}-{int(row['Age'])}-{row['Education short']}\n{row['Risk']}"
        for _, row in df.iterrows()
    ]
    colors = [TIER_COLORS[row["Tier"]] for _, row in df.iterrows()]

    months_values = np.array(
        [
            6.0 if not math.isfinite(value) else min(float(value), 6.0)
            for value in df["Recovery months"]
        ],
        dtype=float,
    )
    wealth_gap = df["Gap 10y"].to_numpy(dtype=float)
    debt_pct = 100.0 * df["Debt probability"].to_numpy(dtype=float)
    y = np.arange(len(labels))

    fig, axes = plt.subplots(1, 2, figsize=(11.6, 5.5), sharey=True)

    axes[0].barh(y, months_values, color=colors, edgecolor="white", height=0.72)
    axes[0].set_yticks(y, labels)
    axes[0].invert_yaxis()
    axes[0].set_xlabel("Months of annual cushion absorbed")
    axes[0].set_title("Immediate budget pressure", pad=8)
    for threshold, text in [(1.0, "Strained"), (3.0, "Precarious")]:
        axes[0].axvline(threshold, color="#777777", linewidth=0.9, linestyle="--")
        axes[0].text(threshold + 0.07, -0.55, text, ha="left", va="bottom", fontsize=8, color="#555555")
    axes[0].axvline(6.0, color="#777777", linewidth=0.9, linestyle="--")
    for idx, row in df.iterrows():
        if math.isfinite(float(row["Recovery months"])):
            label = f"{float(row['Recovery months']):.1f}m"
        else:
            label = "No cushion"
        axes[0].text(months_values[idx] + 0.10, y[idx], label, ha="left", va="center", fontsize=8, color="#222222")

    axes[1].barh(y, wealth_gap, color=colors, edgecolor="white", height=0.72)
    axes[1].set_xlabel("Local currency units")
    axes[1].set_title("Ten-year wealth gap", pad=8)
    axes[1].xaxis.set_major_formatter(FuncFormatter(lambda value, pos: f"{value:,.0f}"))
```

Back to contents

```
    axes[1].tick_params(axis="y", labelleft=False)
    wealth_max = float(max(wealth_gap))
    for idx, value in enumerate(wealth_gap):
        axes[1].text(
            value + 0.02 * wealth_max,
            y[idx],
            f"{value:,.0f} | debt {debt_pct[idx]:.1f}%",
            ha="left",
            va="center",
            fontsize=8,
            color="#222222",
        )
    for ax in axes:
        ax.grid(True, axis="x", color="#e5e5e5", linewidth=0.8)
        ax.spines["top"].set_visible(False)
        ax.spines["right"].set_visible(False)

    axes[0].set_xlim(0.0, 6.6)
    axes[1].set_xlim(0.0, 1.25 * wealth_max)
    fig.subplots_adjust(left=0.30, right=0.985, bottom=0.14, top=0.84, wspace=0.18)
    fig.savefig(os.path.join(FIG_DIR, "p3_profile_impacts.pdf"), bbox_inches="tight")
    plt.close(fig)


def plot_budget_curves(calibration, us_disposable, uk_disposable):
    """Plot the salary-response curves used to interpret the tier thresholds."""
    fig, axes = plt.subplots(1, 2, figsize=(10.6, 4.3))

    salary_grid_us = np.linspace(25_000, 80_000, 180)
    salary_grid_uk = np.linspace(15_000, 80_000, 180)
    curve_specs = [
        {
            "country": "US",
            "region": "South",
            "gender": "Male",
            "age": 25,
            "education": "No college",
            "grid": salary_grid_us,
            "ax": axes[0],
            "title": "U.S. South male, age 25, no college",
        },
        {
            "country": "UK",
            "region": "England",
            "gender": "Male",
            "age": 40,
            "education": "B.A. or higher",
            "grid": salary_grid_uk,
            "ax": axes[1],
            "title": "U.K. England male, age 40, BA+",
        },
    ]
    line_colors = {"Low": "#7ea07a", "Moderate": "#cfb36b", "High": "#9d5c63"}

    for spec in curve_specs:
        ax = spec["ax"]
        age_group = age_group_for_p2(spec["age"])
        for risk_label, color in line_colors.items():
            loss = -p2.evaluate_profile(
                calibration,
                spec["country"],
                spec["gender"],
                age_group,
                spec["education"],
                risk_label,
            )["Conditional net"]
            values = []
            for salary in spec["grid"]:
                if spec["country"] == "US":
                    disposable = us_disposable(salary, spec["age"], spec["region"])
                else:
                    disposable = uk_disposable(salary, spec["age"], spec["region"])
                cushion = max(disposable, 0.0)
                months = recovery_months(loss, cushion)
                values.append(12.0 if not math.isfinite(months) else min(months, 12.0))
            ax.plot(spec["grid"], values, color=color, linewidth=2.0, label=risk_label)

```

Back to contents

```
        for threshold in [1.0, 3.0, 12.0]:
            ax.axhline(threshold, color="#777777", linewidth=0.9, linestyle="--")
        ax.set_title(spec["title"], pad=8)
        ax.set_xlabel("Annual salary")
        ax.set_ylabel("Months of annual cushion absorbed")
        ax.xaxis.set_major_formatter(FuncFormatter(lambda value, pos: f"{value:,.0f}"))
        ax.xaxis.set_major_locator(MaxNLocator(5))
        ax.set_ylim(0, 12.6)
        ax.grid(True, axis="y", color="#e5e5e5", linewidth=0.8)
        ax.spines["top"].set_visible(False)
        ax.spines["right"].set_visible(False)

    axes[0].legend(frameon=False, loc="upper right")
    axes[1].text(0.98, 0.93, "Thresholds: 1, 3, and 12 months", transform=axes[1].transAxes, ha="right", va="top", fontsize
     =8, color="#444444")
    fig.subplots_adjust(left=0.08, right=0.985, bottom=0.18, top=0.86, wspace=0.25)
    fig.savefig(os.path.join(FIG_DIR, "p3_budget_curves.pdf"), bbox_inches="tight")
    plt.close(fig)


def run_sensitivity(df):
    """Propagate uncertainty through the ten-year wealth-gap calculation."""
    rng = np.random.default_rng(20260228)
    labels = [
        f"{row['Country']} {row['Region']}\n{int(row['Salary'] / 1000)}k {row['Risk']}"
        for _, row in df.iterrows()
    ]
    data = []
    summary = []

    for _, row in df.iterrows():
        base_gap = float(row["Gap 10y"])
        base_debt = float(row["Debt probability"])
        changes = []
        debt_shift = []
        for _ in range(1000):
            loss = float(row["Loss"]) * (1.0 + rng.uniform(-0.10, 0.10))
            raw_disposable = float(row["Disposable"]) * (1.0 + rng.uniform(-0.10, 0.10))
            cushion = max(raw_disposable, 0.0)
            save_rate = SAVE_RATE * (1.0 + rng.uniform(-0.10, 0.10))
            debt_rate = DEBT_RATE * (1.0 + rng.uniform(-0.10, 0.10))
            cv = CV_BY_RISK[row["Risk"]] * (1.0 + rng.uniform(-0.10, 0.10))
            cv = max(cv, 0.15)

            if cushion <= 0.0:
                alt_debt = 1.0
            else:
                sigma = math.sqrt(math.log(1.0 + cv * cv))
                mu = math.log(max(loss, 1e-9)) - 0.5 * sigma * sigma
                alt_debt = 1.0 - normal_cdf((math.log(cushion) - mu) / sigma)
            alt_gap = wealth_gap_10(loss, cushion, save_rate=save_rate, debt_rate=debt_rate)

            changes.append(100.0 * abs(alt_gap - base_gap) / max(base_gap, 1e-9))
            debt_shift.append(100.0 * abs(alt_debt - base_debt))

        change_array = np.asarray(changes, dtype=float)
        debt_array = np.asarray(debt_shift, dtype=float)
        data.append(change_array)
        summary.append(
            {
                "Profile": f"{row['Country']} {row['Region']} {int(row['Salary'] / 1000)}k {row['Risk']}",
                "MedianGapPct": float(np.median(change_array)),
                "P90GapPct": float(np.percentile(change_array, 90.0)),
                "MeanDebtShiftPP": float(np.mean(debt_array)),
            }
        )

    return labels, data, pd.DataFrame(summary)


def plot_sensitivity_boxplot(labels, data):
    """Plot the Monte Carlo sensitivity summary for the P3 outputs."""
    fig, ax = plt.subplots(figsize=(10.2, 4.2))
    box = ax.boxplot(data, patch_artist=True, widths=0.62, showfliers=False)
    palette = ["#9d5c63", "#cfb36b", "#c9896a", "#3b6f9c", "#7ea07a", "#6d8b74"]
    for idx, patch in enumerate(box["boxes"]):
        patch.set_facecolor(palette[idx % len(palette)])
```

Back to contents

```
        patch.set_edgecolor("#555555")
        patch.set_linewidth(1.0)
    for part_name in ["whiskers", "caps", "medians"]:
        for artist in box[part_name]:
            artist.set_color("#555555")
            artist.set_linewidth(1.0)

    ax.set_xticklabels(labels, rotation=16, ha="right")
    ax.set_ylabel("Absolute change in 10-year wealth gap (%)")
    ax.set_title("Monte Carlo sensitivity of representative P3 profiles", pad=8)
    ax.grid(True, axis="y", color="#e5e5e5", linewidth=0.8)
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.text(
        0.02,
        0.98,
        "1000 trials with +/-10% variation in disposable income,\nannual loss, CV, savings rate, and debt rate",
        transform=ax.transAxes,
        ha="left",
        va="top",
        fontsize=7.8,
        color="#444444",
        bbox={"facecolor": "white", "edgecolor": "none", "alpha": 0.9, "pad": 1.8},
    )
    fig.subplots_adjust(left=0.08, right=0.985, bottom=0.28, top=0.84)
    fig.savefig(os.path.join(FIG_DIR, "p3_sensitivity_boxplot.pdf"), bbox_inches="tight")
    plt.close(fig)


def main():
    """Run the full P3 workflow and write all LaTeX-ready outputs."""
    survey = p2.load_survey_sheet()
    calibration = p2.build_calibration(survey)
    us_disposable, uk_disposable = build_disposable_functions()

    profiles = [
        {"Country": "US", "Region": "South", "Gender": "Male", "Age": 25, "Education": "No college", "Salary": 35000, "Risk": "High"},
        {"Country": "US", "Region": "South", "Gender": "Male", "Age": 25, "Education": "No college", "Salary": 45000, "Risk": "High"},
        {"Country": "US", "Region": "West", "Gender": "Male", "Age": 38, "Education": "B.A. or higher", "Salary": 60000, "Risk": "High"},
        {"Country": "UK", "Region": "England", "Gender": "Male", "Age": 40, "Education": "B.A. or higher", "Salary": 45000, "Risk": "High"},
        {"Country": "UK", "Region": "Wales", "Gender": "Female", "Age": 25, "Education": "No college", "Salary": 25000, "Risk": "Moderate"},
        {"Country": "UK", "Region": "Northern Ireland", "Gender": "Female", "Age": 70, "Education": "B.A. or higher", "Salary": 35000, "Risk": "Low"},
    ]

    rows = [evaluate_profile(calibration, us_disposable, uk_disposable, profile) for profile in profiles]
    df = pd.DataFrame(rows)
    df["Education short"] = df["Education"].map({"No college": "NC", "B.A. or higher": "BA+"})

    write_profile_table(df)
    plot_profile_impacts(df)
    plot_budget_curves(calibration, us_disposable, uk_disposable)
    labels, data, sensitivity_summary = run_sensitivity(df)
    plot_sensitivity_boxplot(labels, data)

    print("Finished")
    print(
        "Representative profiles:",
        df[
            [
                "Country",
                "Region",
                "Gender",
                "Age",
                "Education",
                "Salary",
                "Risk",
                "Disposable",
                "Loss",
                "Recovery months",
                "Debt probability",
                "Gap 10y",
```

Back to contents

```
                "Tier",
            ]
        ]
        .round({"Disposable": 1, "Loss": 1, "Recovery months": 2, "Debt probability": 3, "Gap 10y": 1})
        .to_dict("records"),
    )
    print(
        "Sensitivity summary:",
        sensitivity_summary.round({"MedianGapPct": 2, "P90GapPct": 2, "MeanDebtShiftPP": 2}).to_dict("records"),
    )


if __name__ == "__main__":
    main()
```

Back to contents