Kelly Betting Simulator

We built this project around a question that sounds simple but has a real risk-management problem inside it: if we think a bet has an edge, how much money should we risk? The app compares several answers to that question under the same simulated futures.

In short, the simulator doesn't try to predict which bet will win or lose but instead tests how different sizing strategies perform across thousands of plausible futures allowing us to see the drawdowns, risk and bankroll volatility before risking real capital.

Reference run used in this report

Input	Value	Why it matters
Starting bankroll	$1,000	The amount being managed
Estimated win probability	55%	The point estimate used in the known-p mode
Decimal odds	1.91	A win returns $1.91 including the original stake
Number of bets	250	The repeated decision horizon
Monte Carlo paths	3,000	Different possible sequences of wins and losses
Seed / ruin threshold	7 / $10	Makes the run repeatable and defines practical ruin

1. The question behind Kelly sizing

The Kelly Criterion is a mathematical formula used to determine the optimal size of a bet over a series of repeated bets. It tries to maximize long-run compounded growth, which means it cares about the effect of losing streaks as well as the average win.

A good edge is not a reason to bet everything you have. The fraction has to be large enough to use the edge, but small enough that a normal losing streak does not destroy the bankroll.

For decimal odds O, the net profit on a win is b = O − 1. If p is the chance of winning and q = 1 − p, the simulator uses this version of the formula where f* is the bet fraction:

The formula we implemented:

$$f^* = max\left(0, \frac{bp - q}{b}\right)$$

When the estimated edge is negative, the app recommends a zero stake instead of inventing a short bet.

What the reference inputs imply

At 1.91 decimal odds, the break-even probability is 52.4%. While the reference model estimate is 55%, so the estimate is a 2.6% advantage above break-even. The expected value is +0.0505 per $1 staked meaning that in the long run, the average gain per dollar wagered is approximately 5.05 cents. Putting this in the formula gives a bet fraction of 5.5%.

This is a useful example because the model estimates a slight advantage above the break-even point. The project therefore has something real to show: a small edge can produce a non-zero stake, but the stake should not be confused with certainty. Additionally, a probability below 50% can still be profitable when the odds are long enough. At 2.50 decimal odds, break-even is 40%, so a 45% estimate can still have positive Kelly sizing.

2. How the project is put together

We kept the project in separate pieces so the user interface does not contain the whole financial model. That makes the math easier to test and makes the engine reusable from a script or another interface.

File / layer	What it does	Why I kept it separate
models.py	Validated wagers, beliefs, configs, strategies, and reports	Creates a clear contract for every input and output
analytics.py	Kelly math, expected growth, Beta quantiles, Wilson intervals, safe exponentiation	Keeps formulas small and directly testable
strategies.py	Cash, full Kelly, fractional, credible-bound, and capped sizing	Lets the engine compare policies without hard-coding them
engine.py	Seeded, chunked, paired Monte Carlo simulation in log wealth	Handles the expensive work and risk state
reporting.py	Turns reports into tables and chart-ready data	Keeps Streamlit formatting out of the model
app.py	Streamlit controls, Plotly chart, strategy table, diagnostics, and caveats	Makes the experiment interactive without owning the math

The simulation flow

1. Validate the inputs, including odds, probabilities, fractions, counts, and the practical-ruin threshold.
2. Choose a probability belief: a fixed KnownProbability or a BetaBelief that represents uncertainty about the true chance.
3. Calculate each strategy's stake fraction. The robust strategy can use a lower Beta percentile instead of the mean.
4. Generate the same latent probability and win/loss sequence for every strategy on a path, so the comparison is fair.
5. Update wealth in log space, track running peaks and drawdowns, and mark paths that cross practical ruin.
6. Convert the results into terminal quantiles, confidence intervals, tables, and charts for the app.

Why log wealth?

Multiplying a bankroll 250 times can overflow or underflow in extreme cases. In log space, a win adds $\log(1 + fb)$ and a loss adds $\log(1 - f)$. The app converts back to dollars only when it is time to display a result.

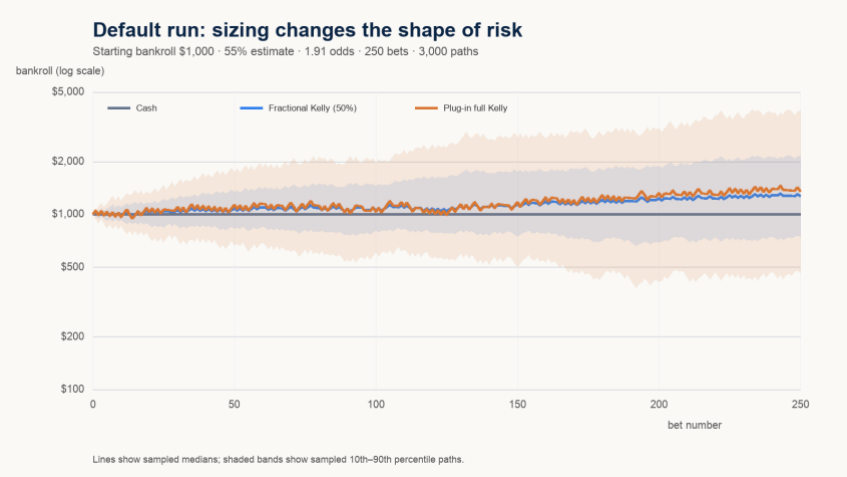

Figure 1. A seeded reference run: the orange full-Kelly line grows faster, but its shaded downside range is wider than half Kelly.

3. The five policies I compare

The app does not present one number as the answer. It puts several strategies next to each other, so the trade-off is visible.

Strategy	How it sizes	Why it is useful
Cash	0%	Zero-risk benchmark. It tells me whether a strategy is earning enough to justify taking risk.
Plug-in full Kelly	Uses the point estimate as if it were exact	The mathematical benchmark, but also the most sensitive to estimation error.
Fractional Kelly (50%)	Half of the full-Kelly fraction	A simple way to trade some theoretical growth for a smoother path.
Credible-bound Kelly	Uses the selected lower probability percentile	Makes uncertainty reduce the stake instead of hiding inside a single average.
Capped Kelly (10% cap)	Uses Kelly, then applies a hard maximum	Prevents one optimistic estimate from creating an oversized first bet.

The words full, fractional, robust, and capped describe different decisions. Full Kelly is not automatically the best choice; it is the cleanest benchmark for showing how much estimation error and drawdown matter.

A small implementation detail that matters
The strategies share the same random outcomes. If full Kelly looks worse than fractional Kelly in a run, that difference comes from the sizing rule—not because one strategy happened to receive a luckier sequence.

What changes when I turn on uncertainty

When you assume the win chance is fixed and known, the safe strategy doesn't have a range of uncertainty to look at, so it uses the same single number as the standard Kelly formula. At the same time, the rule that limits your bet to a maximum of 10% doesn't change anything because standard Kelly only wants to bet 5.5% anyway. Seeing the strategies give the exact same numbers here is completely normal—it means the math is working as intended, not that there is a bug in the code.

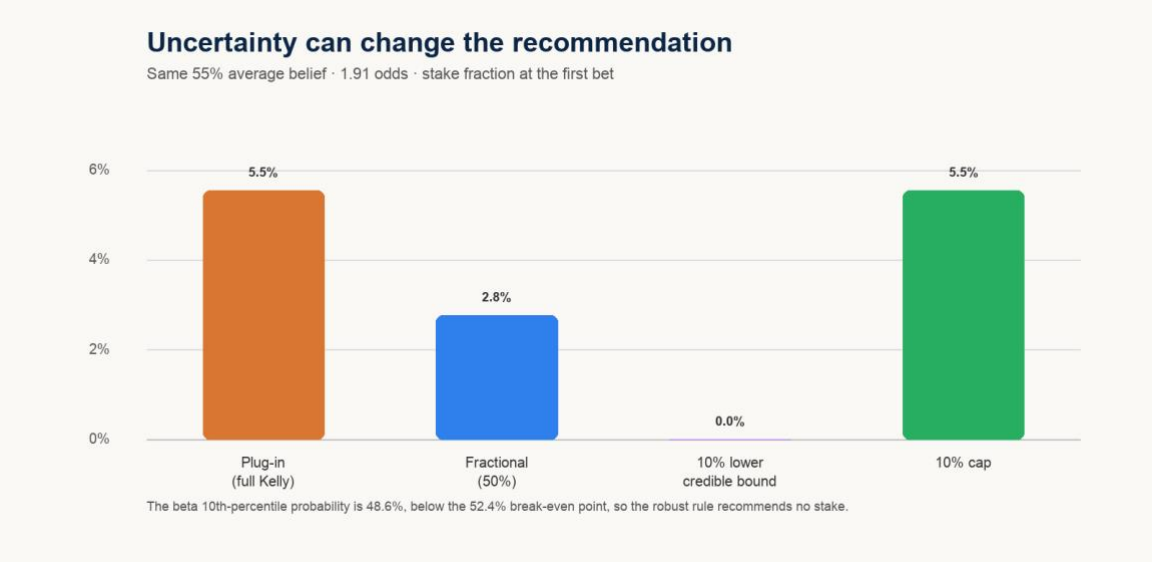

Figure 2. The same average belief can lead to a zero robust stake once the lower tail falls below break-even.

4. Probability uncertainty is the important upgrade

The hardest part of Kelly sizing is usually not the formula. It is deciding whether the probability estimate deserves to be trusted. The project exposes two different answers instead of quietly treating them as the same thing.

Model	What the simulator assumes	What it is good for
Known probability	The supplied p is treated as the true probability on every path	A clean benchmark for understanding the formula and the engine
Beta uncertainty	p itself is drawn from Beta(alpha, beta) once per path, then outcomes are drawn from that p	Showing how limited evidence can widen outcomes and reduce robust sizing

The Streamlit app lets the user enter an estimated probability and an effective sample size. For the uncertainty example in this report, 55% with an effective sample size of 100 becomes Beta(55, 45). The mean is still 55%, but the 10th–90th percentile interval is about 48.6%–61.3% meaning that the there is an 80% chance the true win rate lies around that range.

Why the robust rule goes to zero here
The odds imply a 52.4% break-even probability. The Beta 10th percentile is 48.6%, below that threshold, so Credible-bound Kelly recommends a 0% stake. This does not prove the true probability is 48.6%; it says the lower tail is not strong enough to justify sizing from it.

I also draw the latent probability once per path, not once per wager. That creates a more realistic kind of model uncertainty: one path represents a world where the underlying edge is a little better or worse, and every outcome in that path is conditional on that world. All strategies then see that same world and the same outcomes.

The uncertainty example changes the risk picture

With Beta(55, 45), full Kelly still uses the mean and has a median final bankroll near $1,347, but its 10th percentile falls to about $199 and its 90th percentile rises above $10,000. Half Kelly is less extreme, while the lower-bound policy stays in cash.

5. Reference run: what came out

The table below uses Known probability mode with a fixed seed. It is an illustration of the engine, not a promise about a real market. The 10th and 90th percentiles show how different paths can be even when the inputs are identical.

Strategy	Stake	P10 final	Median	P90 final	> start	P90 DD
Cash	0.0%	$1,000	$1,000	$1,000	0.0%	0%
Full Kelly	5.5%	$465	$1,347	$3,902	64.3%	75%
Half Kelly	2.8%	$745	$1,267	$2,154	73.6%	47%
Lower-bound	5.5%	$465	$1,347	$3,902	64.3%	75%
Capped	5.5%	$465	$1,347	$3,902	64.3%	75%

The main pattern is not simply “more Kelly is better.” Full Kelly has the highest median in this run, but it also has the widest downside range and a 75% P90 maximum drawdown. Half Kelly gives up some upside while bringing that P90 drawdown down to about 47%.

Cash stays at $1,000 because it never risks the bankroll. The lower-bound and capped rows match full Kelly in this known-probability reference because the uncertainty rule has no lower tail to use and the cap is not binding. In Beta mode, those policies separate clearly.

How I read the result

The simulator is not choosing a winner for us. It is showing what we are buying with a larger stake. More exposure to the edge, more exposure to estimation error, and deeper losing streaks when the path is unlucky.

6. How we checked whether it was working correctly

A simulation can return a number even when the code is wrong. I wrote tests around the formula, reproducibility, uncertainty behavior, boundaries, and invalid inputs. The current test suite has 5 passing tests.

Check	What it protects
Odds math and below-50% edge	Confirms break-even, expected value, and Kelly sizing use net payout correctly.
Reproducible shared paths	The same seed produces the same terminal wealth and drawdowns, and cash stays flat.
Beta belief and robust policy	A lower probability percentile cannot silently produce a larger stake than the mean.
Probability boundaries	p = 0 and p = 1 stay finite; safe exponentiation reports overflow instead of returning infinity.
Invalid inputs	Bad odds, NaN probabilities, invalid counts, one-hundred-percent fractions, and similar cases raise errors.

Choices that make the result more reliable

- Log-space wealth updates avoid multiplying a dollar value until it becomes infinity or zero.
- SeedSequence creates separate random streams for outcomes and sampled chart paths, so adding a visual sample does not change the main results.
- The engine processes paths in chunks of 1,024 instead of requiring one huge path matrix in memory.
- Wilson intervals put an uncertainty band around the probability of finishing above the starting bankroll.
- Running peaks, maximum drawdown, and practical ruin are tracked during the simulation rather than reconstructed from a chart.

These tests support the implementation mechanics however they do not validate the user's probability estimate or guarantee a real market opportunity.

7. Limitations, next steps, and what we learned

The model leaves out many things that matter in actual betting or trading that cannot be controlled.

The biggest assumptions

- Outcomes are binary and independent conditional on a path's probability.
- The odds do not move, and fees, taxes, limits, slippage, liquidity, and market impact are not modeled separately.
- The same edge is available for all 250 bets meaning there is no time-varying probability or price.
- The Beta belief is a simple uncertainty model, not a guarantee that the chosen prior or effective sample size is right.

A realistic improvement plan

Priority	Improvement	Why it would matter
1	Add fees, changing odds, and position limits	Turn the clean binary example into a more realistic decision model.
2	Add adaptive Beta updating	Let the model learn from observed outcomes without looking ahead.
3	Add calibration and out-of-sample evaluation	Test whether the probability estimate deserves to be used at all.
4	Allow risk-constrained sizing	Choose a fraction subject to a drawdown or ruin limit instead of only a multiplier.
5	Expand app and engine tests	Cover UI submissions, uncertainty changes, exact benchmarks, and no-lookahead behavior.