The Martingale

CMS 430, Spring 2017

Due Thursday, February 2, at 11:59 PM

Description

A *martingale* is a classic gambling strategy where the player doubles his bet after every loss. It's traditionally applied to roulette, craps, baccarat, or other games that have even money bets where the odds of winning are close to 50%. Some have characterized it as an "unbeatable" gambling strategy, but that's ridiculous.

Let's consider a simple coin-flipping game:

- The player makes a wager and flips a coin
- If the coin comes up heads the player wins even money (that is, the player wins the value of his wager)
- If the coin comes up tails the player loses his wager

A player using a martingale starts by betting the minimum amount: let's say it's $1. If he wins, he keeps $1 and bets $1 again on the next round.

If he loses, he doubles the bet on the next round to $2. If he wins the $2 bet, he will have recovered the $1 he lost in the first round, plus an additional $1 profit. After winning, he resets his bet amount to $1.

If he loses the $2 bet, he doubles again to $4. If he wins this round, he will recovered all previous losses ($1 from the first round and $2 from the second round) plus an additional $1.

This strategy continues, with the player doubling his stakes after every loss and resetting his bet to the minimum after a win. If the player ever reaches a point where the required bet is larger than his remaining money, he simply bets everything he has and either wins or goes broke.

Simulation

Is the martingale a good strategy? Clearly, if the player has a large enough bankroll to sustain several losses in a row, it's likely that a win will occur,

wiping out all previous losses and giving the player a small profit. This fact has led some unsophisticated observers to think that the martingale is "unbeatable," because the probability of several consecutive losses must be impossibly small. Right?

In reality, all betting strategies fall before the relentless onslaught of the house's advantage. It doesn't matter if you put all your money on a single spin of the roulette wheel or dribble it out over a longer series of bets, the long-run expected outcome will always tend towards a loss, with the margin of that loss determined by the house's edge.

To verify this, let's **simulate** the martingale strategy, under the following assumptions:

- The player begins with $255
- The minimum bet is $1
- The player doubles his bet after each loss and resets it to $1 after each win
- Wins are added to the player's total bankroll, so they can be used in future bets; they're not set aside in a separate pile
- The player will play for 100 rounds, or until he goes bankrupt
- The player is running the simple coin-flipping game with a fair coin

Write a Python program that simulates **1 million** instances of the gambling process described above. Each instance should be independent and run for either 100 rounds or until the player goes bankrupt. Keep track of the player's net outcome at the end of each instance, which will be in the range (-255, 355).

When the 1 million trials are complete, **plot a histogram** showing the distribution of outcomes generated by the martingale. Your histogram should have enough bins to clearly show the structure of the distribution, labeled axes, and a title.

Submission and Grading

You will upload your solution, including your Python script and your final histogram plot, to a personal GitHub repository.

There are two ways to upload your work to GitHub:

- Through the GitHub web interface
- From the command line in Cloud 9

I've included instructions on using the command line in the starter files for this project.

I'll grade your project using the following breakdown:

- Correct results: 50 points (your histogram should show that most outcomes lead to a small net gain but these are balanced by a fraction of outcomes with catastrophic losses)
- Simulation design and program organization: 40 points (correct use of randomization, modular design, etc.)
- Style and clarity: 10 points (variable names, comments, function docstrings, etc.)