Project 1: Martingale (CS7646)

Stella Lai-Hoong Soh

lsoh3@gatech.edu

Question 1: In Experiment 1, based on the experiment results *calculate and provide the estimated probability of winning $80 within 1000 sequential bets*. *Thoroughly explain your reasoning for the answer using the experiment output*. Your explanation should NOT be based on estimates from visually inspecting your plots.

In my martingale.py, winnings is a 2-d array that is generated by calling on build_winnings(), which in turn calls on winning_strategy(). In the interpreter, if I were to run these commands:

```
>>> winnings = build_winnings(1000, 18/38.0, 80, -sys.maxsize - 1)
>>> winnings
array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
      [ 1., -1., -1., ..., -1.,  1., -1.],
      [ 0., -3.,  1., ...,  1.,  0., -3.],
      ...,
      [80., 80., 80., ..., 80., 80., 80.],
      [80., 80., 80., ..., 80., 80., 80.],
      [80., 80., 80., ..., 80., 80., 80.]])    ---> winnings is a 1001 x 1000 array

>>> mask = winnings >= 80.0
>>> number_of_trues = mask.sum()
>>> number_of_trues
831217
```

The estimated probability of winning $80 within 1000 successive bets = 83,1217 / (1001 x 1000)

= 0.8303 or ~83%

Question 2: In Experiment 1, *what is the estimated expected value of winnings after 1000 sequential bets? Thoroughly explain your reasoning for the answer.*

In the interpreter, if one were to find the outcomes of the following commands:

```
>>> df2 = pd.DataFrame(data=winnings)
>>> mean_episodes_2 = df2.mean(axis=1)
>>> mean_episodes_2
0        0.000
1       -0.014
2       -0.015
3       -0.216
:
        ...
:
995     80.000
996     80.000
997     80.000
998     80.000
999     80.000
1000    80.000
Length: 1001, dtype: float64
```

one can see that the value in mean_episodes_2 stabilizes to $80. The estimated expected value of winnings after 1000 sequential bets is, therefore, $80.

Question 3: In Experiment 1, *do the mean upper standard deviation line and mean lower standard deviation lines reach a maximum (or minimum) value and then stabilize? Do the standard deviation lines converge as the number of sequential bets increases? Thoroughly explain why it does or does not.*

In my code, the mean upper standard deviation line is defined by:

```
line_above_mean = mean_episodes_2 + df2.std(axis=1)
```

winnings is a 2-d 1001 x 1000 array. One can see that standard deviation stabilizes at 0.0:

```
>>> standard_deviation = df2.std(axis=1)

>>> standard_deviation

0         0.000000
1         1.000402
2         1.848836
3         3.077469
```

```
:
:
998       0.000000
999       0.000000
1000      0.000000
Length: 1001, dtype: float64
```

mean_episodes_2 converges to 80.0. Since df.std(axis=1) converges to 0.0, this translates to line_above_mean reaching a value of 80.0, and getting stabilized there. Similarly,

```
line_below_mean = mean_episodes_2 - df2.std(axis=1)
```

Since mean_episodes_2 converges to 80.0, and df.std(axis=1) converges to 0.0, this translates to line_below_mean reaching a maximum value of 80.0 and getting stabilized there.

Looking at Figure 2 on Page 6 , we see that the line_above_mean (in orange) and the line_below_mean(in green) converge to 80.0, and thus confirm our observation above. Standard deviation of a sample is defined as sqrt [(Σ(x_i -μ)^2 / n-1] where μ = mean. If Σ(x_i -μ)^2 = 0, this means for every i, $x_i = \mu$, which translates to every data value equaling the mean. Now, if every data value equals the mean, there is no variation, and therefore the standard deviation converges to 0.0 as the number of sequential bets increases.

Question 4: In Experiment 2, based on the experiment results *calculate and provide the estimated probability of winning $80 within 1000 sequential bets*. *Thoroughly explain your reasoning for the answer using the experiment output.* Your explanation should NOT be based on estimates from visually inspecting your plots.

In Experiment 2, the gambler has a $256 bankroll. In the interpreter:

>>> **winnings** = **build_winnings(1000, 18/38.0, 80, -256)**

>>> **winnings**

```
array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
      [-1., -1., -1., ..., -1.,  1., -1.],
      [ 1., -3., -3., ...,  1.,  0., -3.],
      ...,
      [80., 80., 80., ..., 80., 80., 80.],
      [80., 80., 80., ..., 80., 80., 80.]])
```

```
>>> mask = winnings >= 80.
>>> number_of_trues = mask.sum()
>>> number_of_trues
533040
```

The estimated probability of winning $80 within 1000 sequential bets = 53,3040/ (1001 x 1000)

= 0.5325 or ~53.3 %

Question 5: In Experiment 2, what is the estimated expected value of our winnings after 1000 sequential bets? Thoroughly explain your reasoning for the answer.

From the winnings array one obtains from Question 4, one creates a dataframe, df4:

```
>>> df4 = pd.DataFrame(data=winnings)
>>> mean_episodes_4 = df4.mean(axis=1)
>>> mean_episodes_4
0        0.000
1       -0.040
2       -0.134
3       -0.296
:
:
998    -37.600
999    -37.600
1000   -37.600
Length: 1001, dtype: float64
```

As shown in the logs above, the expected value of winnings after 1000 successive bets would be -$37.60.

Question 6: In Experiment 2, do the mean upper standard deviation line and mean lower standard deviation lines reach a maximum (or minimum) value and then stabilize? Do the standard deviation lines converge as the number of sequential bets increases? Thoroughly explain why it does or does not.

Question 5 above shows that the mean_episodes_4 stabilizes at -$37.60. In the interpreter, the outcomes when I run the following commands are as follows:

```
>>> standard_deviation2 = df4.std(axis=1)
>>> standard_deviation2
0       0.000000
1       0.999700
2       1.930225
:
        ...
:
997     160.341977
998     160.341977
999     160.341977
1000    160.341977
Length: 1001, dtype: float64
```

The outcome above proves that standard_deviation2 stabilizes at $160.341977. From my martingale.py code, line_above_mean_4 = mean_episodes_4 + df4.std(axis=1)

= -$ 37.60 + $ 160.341977

= $122.74

line_below_mean_4 = mean_episodes_4 - df4.std(axis=1)

= -$37.60 - $160.341977

= - $197.94

The mean upper standard deviation (i.e. line_above_mean_4) and the mean lower standard deviation (i.e. line_below_mean_4) lines reach a maximum value of $122.74 and a minimum value of -$197.94 respectively. At these 2 values, both lines are parallel to each other, and do not converge as the number of sequential bets increase. The reason could be that once the winning target $80 is reached, the gambler stops betting, and the $80 value persists.

Question 7: *What are some of the benefits of using expected values when conducting experiments instead of simply using the result of one specific random episode?*

In this project, the probability of getting a black in American Roulette is 18/38 or 0.4736. The expected value or the number of blacks one gets in 10 successive bets = (0.4736) * 10

= 4.7 or ~5

As such, the expected value represents the average winnings of the roulette game. When we increase the number of successive bets - perhaps doing 1000 successive bets, we see that the expected value or the number of blacks one gets = 0.4736 * 1000 or 473.

Expected value is also typically divided into **neutral**, **positive** or **negative**-valued numbers. (Soni Devin, 2018)

American roulette, whereby the casino advantage is 5.25%, gives a minuscule expected value to the gambler. This is an example of a **negative** expected value scenario. Being informed of whether we are heading into **neutral** (scissors-paper-stone game), **positive** (investing in the stock market) or **negative** expected value scenarios helps us in making decisions as to whether we want to invest time, money or effort in that endeavor.

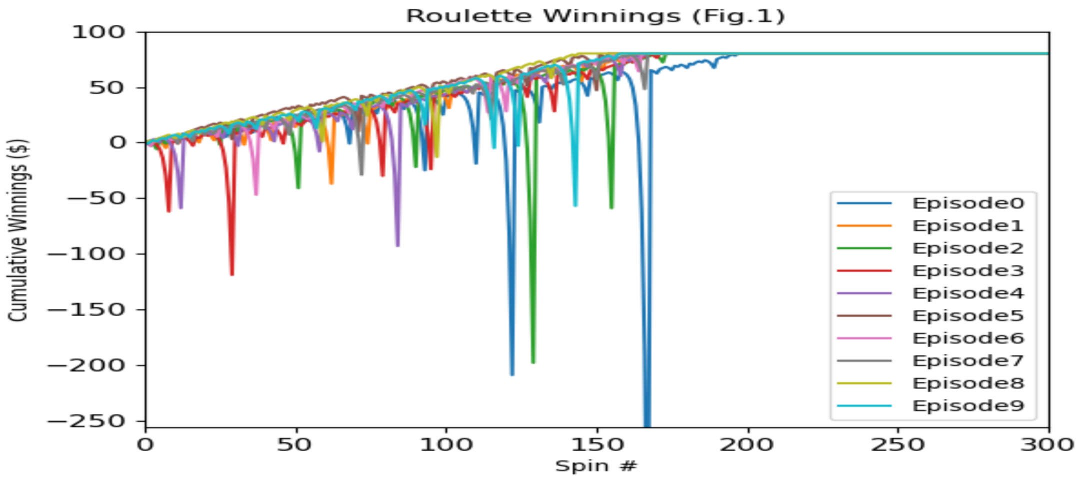

Figure 1 —The number of spins vs. Cumulative winnings ($)

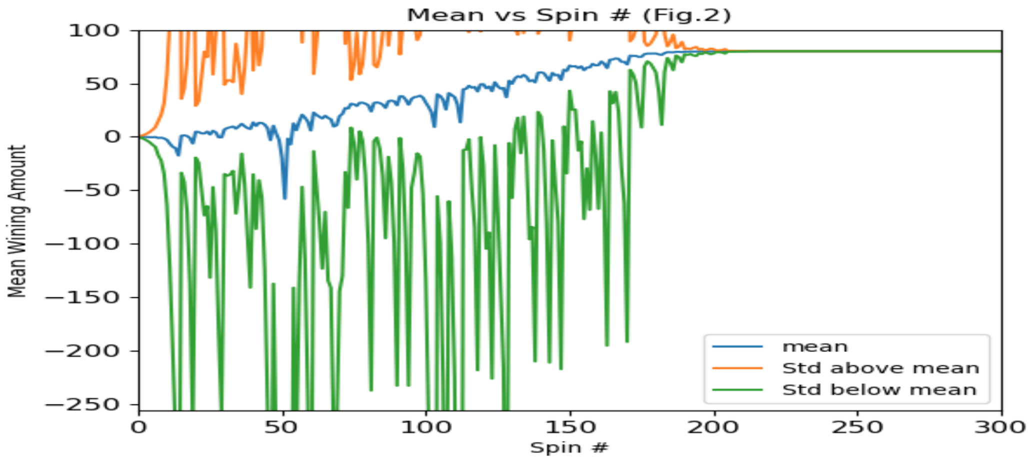

Figure 2 —The number of spins vs. Mean Winning Amount ($)

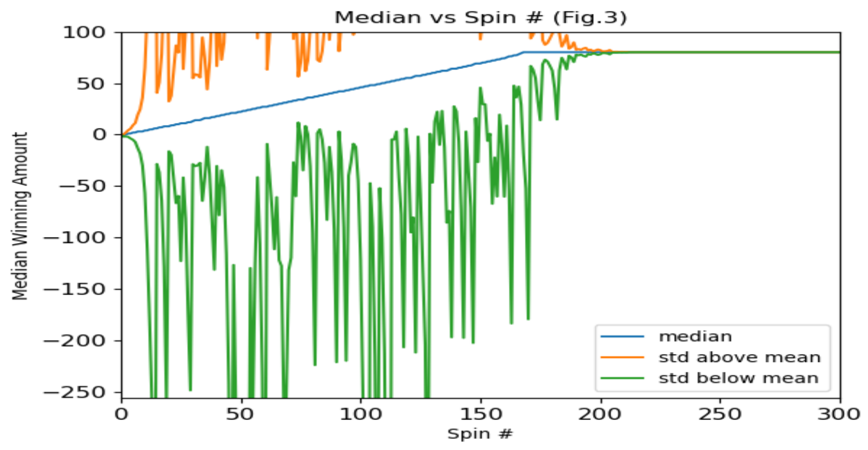

Figure 3 —The number of spins vs. Median Winning Amount ($)

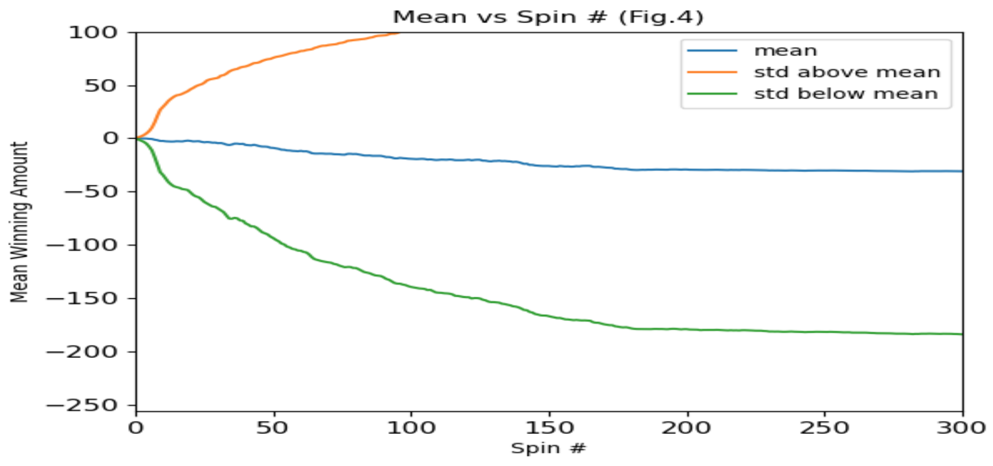

Figure 4 —The number of spins vs. Mean Winning Amount ($)

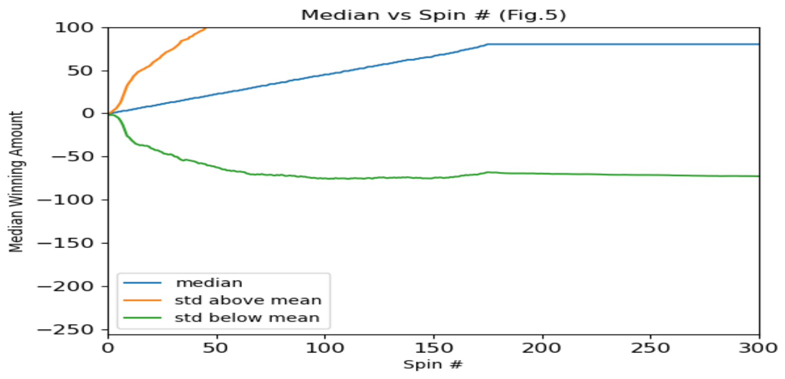

Figure 5 —The number of spins vs. Median Winning Amount ($)

REFERENCE

1. Soni, Devin.(2018). What Is Expected Value?