Using Markov Chains to Analyze Roulette Betting Strategies

Isaiah Martinez, Chance Vandergeugten
CSUN

e-mails:
isaiah.martinez.891@my.csun.edu
chance.vandergeugten.704@my.csun.edu

04/03/2021

Contents

1 Introduction

Gambling has been around since the dawn of civilization, ranging from simple six-sided die games to more sophisticated and developed games like roulette or craps. In modern times, gambling has provided a sound business model for entrepreneurs to create wealth as the odds of each game are fixed in a way that will generate revenue for the game operator in the long run. If we take a look at cities like Las Vegas or Atlantic City, it becomes apparent how powerful these simple games are and how strongly favored the house is mathematically.

The amount of gamblers has risen steadily over the past few years despite the odds. That is up until the coronavirus pandemic occurred, where the market size of the casino hotel sector decreased a substantial amount in the United States. As 2021 comes in, we are seeing an increase in players as things begin to return towards a sense of normalcy in this industry.

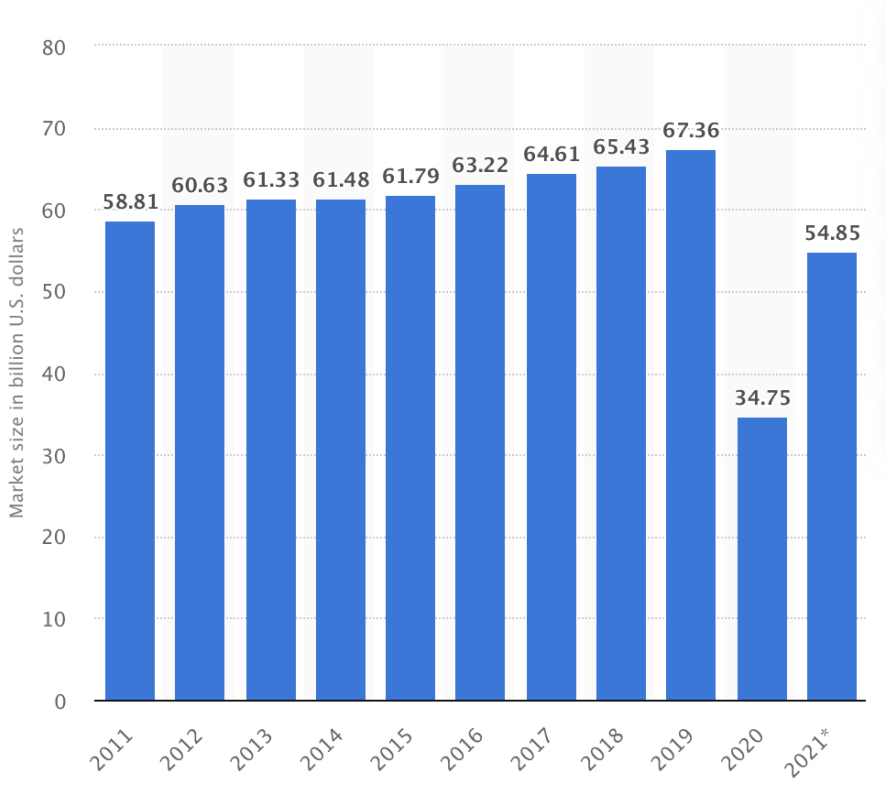

Figure 1: Transition state diagram

What does this mean for returning and new players? Since the adoption of gambling as a business model for owners, patrons of gambling houses and casinos have tried to figure out ways to turn the odds in their favor and make the games more profitable for them. Some have found success in beating the casinos, but most of these methods were not mathematical in nature and relied on either sleight of hand or exploits in equipment used in the games. Since these

discoveries, casinos have become more sophisticated and found ways to prevent these player advantages from existing by implementing new equipment as well as increasing surveillance. Is this the end of the road for gamblers around the world looking for an edge?

This report will focus on the notion that players may not be able to beat the long run odds set up by the casinos, but they can still become aware that the games they play and the bets they choose are not all created equally. Perhaps, giving them the advantage they so seek.

1.1 Interest in the game

After coming across the idea of Markov Chains, we had become curious about their applications towards gambling scenarios. There are lots of games that can be played at a casino, but Markov Chains rely on the probability of certain events to be independent, so we narrowed down our decision to roulette (each spin is random and has no influence on the outcome of the next spin, or any subsequent spin after that).

1.2 Roulette rules

In roulette, a dealer will release a ball into a rotating wheel that has 38 evenly spaced slots. Players will bet on which number(s) they believe the ball will land on. After the ball has landed in a particular slot, all losing bets are taken in by the casino, and all winning bets are payed out to the players. Payouts for each bet covered in this report are in the table below with data found from 888 casino [1]:

Bet	**Win Probability**	**Payout Ratio**
Red/Black	47.37%	1:1
Even/Odd	47.37%	1:1
Low/High	47.37%	1:1
Dozen	31.58%	2:1
Column	15.79%	2:1
Six Line	13.16%	5:1
Straight	2.63%	35:1

Table 1: Bets and Associated Probabilities

1.3 Markov Chains

Markov Chains are mathematical systems that utilize different "states" and the probabilities associated with transitioning between these states in order to draw conclusions about specific events [2]. We can represent this visually by drawing a digraph that contains nodes which will represent each "state" of the Markov Chain process and associated probabilities between connected nodes. In a Markov Chain digraph, each node will have arrows that show the probability of transitioning to another state, and since we are discussing probabilities, all of the transitions from one node must sum to 1. In Markov Chain models,

"absorbing states" have a 100% chance of remaining in that same state (once in this state, it is impossible to transition out).

Below is an example of a three-state Markov Chain digraph connecting three states q0, q1, and q2 (q2 is an absorbing state):

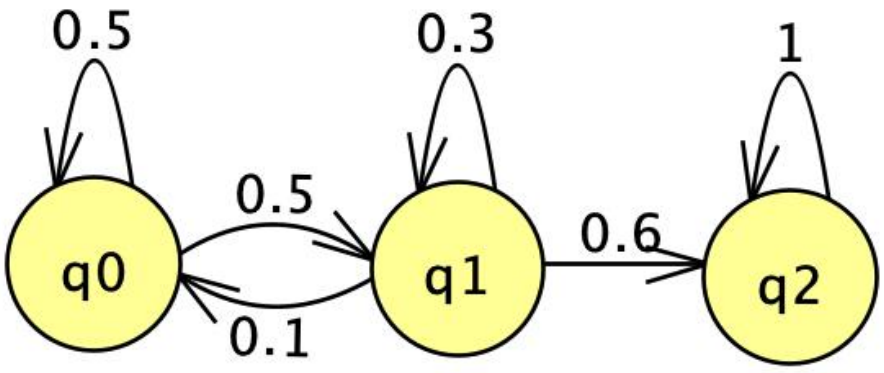

Figure 2: Markov Chain

1.4 Transition Matrices

Besides representing these transitions as a digraph, we can use matrices. By taking into account all the probabilities of transitioning from state to state, we can make a special matrix called a transition matrix. An example may be seen below using the same data from the example Markov Chain digraph:

$$M = \begin{bmatrix} .5 & .5 & 0 \\ .1 & .3 & .6 \\ 0 & 0 & 1 \end{bmatrix}$$

To read this data we denote the row index as the current state we are at, and the column index as the state to move towards. In this case at M(1,1), the way to interpret it would be to say that there is a 50% chance to move from state q0 to state q0, a 50% chance to move from state q0 to state q1, and a 0% chance to move to from q0 to q2.

Note that the rows add up to 1 similar to the total chances of movement from a state aforementioned above. Also notice that q2 is an absorbing state, where the only possible transition in state q2 is to state q2 (itself).

When you multiply a transition matrix with itself, you end up with another transition matrix that represents the probabilities of being in each state after 2 transitions. This idea can be expanded and the original transition matrix can be raised to any power "t", and the resulting matrix will represent the probabilities of being in each state after "t" transitions. If you raise the original transition matrix to infinite(or a really high power), you will end up with a "steady-state" matrix which will give insight about the end behavior of the Markov Chain. In

this report, we will look at a transition matrix from the Markov Chain process and raise it to a really high power. This will allow us to observe the steady-state matrix, which will provide some long-run insight about the betting strategies discussed in this report.

2 Process/Methods

When working with matrices, we wanted to come up with a standardized way of viewing them. In doing so, it would be out hope that the readers would be able to do the calculations and interpret them for themselves. We decided that it would be best to base all calculations and results on the same chip denomination.For simplicity of the model, we assume that a single chip represents $1. Should someone want to calculate the values for $5 chips, then they should multiply all results by 5 and the results would be appropriately adjusted. This allowed us to account for any possible denomination that is used in casinos today.

2.1 Transition Matrix as our focus

Recall M from section 1.4. Say M represents the probabilities on the first spin of a roulette game. M^2 would represent the probabilities that exist when you play this game with the same bet for the second time. In this case,

$$M^2 = \begin{bmatrix} .3 & .4 & .3 \\ .08 & .14 & .78 \\ 0 & 0 & 1 \end{bmatrix}$$

Notice that the absorbing state remains an absorbing state, i.e. it will not have a transition to any other state besides itself.

If we continue to multiply M by itself many times over, we would find the steady-state matrix. For our analysis, we will take into account the actual probabilities of playing roulette with certain bets. This would mean that based on the number of chips the player starts with and how many chips they bet, we will be able to determine the probability of certain events in the long-run. In our case we will look at the chances of going broke or leaving the game with a predetermined amount (playing indefinitely with the same strategy until either event occurs). This predetermined amount of chips will be defined as the Desired Exit Amount, shortened to DEA.

Definition 2.1. *DEA: Desired Exit Amount. The desired exit amount of chips for any particular betting strategy.*

When putting this into the code, they will be represented by input into the functions in the same order as described above: (Chips used to play game, DEA).

2.2 Reading the Matlab results

Once the inputs have been given to the Matlab functions, we can properly analyze what the steady-state results are. We are interested in observing the probabilities of reaching the DEA given a particular betting strategy and starting chip amount for the player. For example, inputting (30) into our red/black Matlab script would represent betting one chip on the red/black bet with a DEA of 30 chips. For the resulting transition matrix: The row value indicates the amount of chips before a given spin and the column indicates the amount of chips after the spin.

Do note that the row number is off by 1, and that in order to understand the results from the code, you must subtract 1 from the row/column in order to

properly interpret the amount of chips. This was a side effect of Matlab starting to index matrices by 1 instead of 0. We correctly represented the results in charts which are in section 4.

After raising the initial transition matrix to a really high power, i.e.

$$M^{100000000}$$

we will be left with a steady-state matrix that only has values in the first column and last column. These are represented by B and W respectively (in the Matlab code). The reason for this is that there will only be the two cases we are interested in, going broke, or reaching the DEA.

We will be comparing the W column vectors to each other for each betting strategy in order to show the probabilities associated with reaching the DEA and see which is best. At this point, a natural question emerged: How will we determine which strategies are best using the resulting steady-state matrix?

2.3 Tipping Point Value

We decided that an appropriate metric to use would be to look at where the probabilities of reaching the DEA are above 50% (within the W column vector). It is at this point where the player has the edge probability wise, although these probabilities rely on the player leaving the game once they reach the DEA. If we take a look at a particular bet on the roulette board, we could determine a value for the DEA and then figure out where the amount of chips the player is required to start the game with in order to have greater than a 50% chance of reaching the DEA.

Because this is an important focus of our project we will define it formally as the Tipping Point Value (TPV). As a player, you will be looking at the odds of losing and eventually find a point where the odds are in your favor to reach the DEA.

Definition 2.2. *TPV: Tipping Point Value. The first value of chips to enter the game with, where the odds of reaching the DEA are above 50% given a particular betting strategy.*

Keep in mind that this is only a probability and not a guarantee to win when entering the game. Additionally, note that for this data to be sound, the player must employ the strategy indefinitely, until an they either go broke or reach the DEA.

By looking at the distance between these TPV and the DEA, we can compare which strategy would be best. If the distance is great, then that means this particular strategy allows the player to start with less chips while maintaining above a 50% of reaching the DEA. If the distance is small, this implies that the player must enter with more chips in order to maintain the advantage.

3 Matlab

For this report, we have written scripts in Matlab that we used to determine the TPV of each betting strategy. This section will describe the code used, how it works, what parameters are considered, and some notes about the scripts.

Before discussing the strategy scripts, it is necessary to talk about two functions that are used to isolate the "busted-state probabilities" and the "winning-state probabilities":

```
function V = win_state(M)
    for ind = 1:length(M)
        V(ind,1) = sum(M(ind,length(M)));
    end
```

Figure 3: Winning Probability Vector

Figure 3 shows the code for the win-state function. This function creates a column vector which has all the values from the last column of our steady-state matrix. The values in this column vector will represent the probability of the player reaching the DEA based on the amount of chips that they enter the game with. This is the vector we used to find the TPV for each betting strategy.

```
function V = bust_state(M)
    for ind = 1:length(M)
        V(ind,1) = sum(M(ind,1:length(M)-1));
    end
```

Figure 4: Bust Probability Vector

Figure 4 shows the code for the bust-state function. This function creates a column vector which has all the values from the first column in our steady-state matrix. The values in this column vector will represent the probability of the player running out of chips to bet based on the amount of chips they start with. Our focus is on the win-state vector, but if the user is curious they can take a look at the bust-state vector to see the complementary probabilities.

NOTE: If the user inspects the transition matrices that are created by each script, they will notice that the parameter values will change the amount of "absorbing states" in the top left region of the matrix. When the user inspects the bottom right region of the transition matrix, they will notice that the same method of changing the amount of absorbing states is different and that instead of having separate absorbing states at the end of the matrix, the final column will encompass all possible "exit scenarios". This is necessary to keep the transition matrix square. For example, if a player makes a 5:1 bet with 19 chips in their bankroll, they will end up with 23 chips on a win. This becomes troublesome if the "defined exit amount" is 20. In the scenario, the last column of the transition matrix "21" will represent the player reaching 20 or more chips.

	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0
4	0.5263	0	0	0	0	0	0.4737	0	0
5	0	0.5263	0	0	0	0	0	0.4737	0
6	0	0	0.5263	0	0	0	0	0	0.4737

Figure 5: Transition Matrix Bust Absorbing States

16	0	0	0.5263	0	0	0	0	0	0.4737	0	0
17	0	0	0	0.5263	0	0	0	0	0	0.4737	0
18	0	0	0	0	0.5263	0	0	0	0	0	0.4737
19	0	0	0	0	0	0.5263	0	0	0	0	0.4737
20	0	0	0	0	0	0	0.5263	0	0	0	0.4737
21	0	0	0	0	0	0	0	0	0	0	1

Figure 6: Transition Matrix Win Absorbing States

3.1 Red/Black Bet

In this section we will outline the code used in the red-black betting strategy. This strategy involves all bets that have a 1:1 payout(Red/Black,Odd/Even,1 to 18/19 to 36).

```
function [M,B,W] = red_black(y)
    %Create Matrix
    M = zeros(y+1);

    %Busted state probs
    M(1,1) = 1;

    %Fill cells with probability of winning a given spin
    for jj = 2:length(M)-1
        M(jj,jj+1) = (18/38);
    end

    %Fill cells with probability of losing a given spin
    for kk = 2:length(M)-1
        M(kk,kk-1) = (20/38);
    end

    %Win state
    M(y+1,y+1) = 1;

    %End behavior of transition matrix
    ESM = M^10000000000000000000000000;

    %Create column vector with busted-state probs
    B = bust_state(ESM);
    %Create column vector with win-state probs
    W = win_state(ESM);
```

Figure 7: 1:1 Bets

This function only has one input of "y" which represents the DEA. Running this script will generate three outputs:

M = Transition matrix based on input parameter.
W = Win-state column vector based on the steady state matrix
B = Bust-state column vector based on the steady state matrix

If the user wants to find the TPV, they should run the program with a chosen parameter and then investigate the "W" vector. The first row that contains a probability greater than .5 will be considered the TPV.

3.2 Dozen/Column Bet

In this section we will outline the code used in the dozen-column betting strategy. This strategy involves all bets that have a 2:1 payout(Dozens or Columns).

```
function [M,B,W] = dozen_column(x,y)
    %Create Matrix
    M = zeros(y+1);

    %Busted state probs
    for ii = 1:x
        M(ii,ii) = 1;
    end

    %Fill cells with probability of winning a given spin
    for jj = x+1:y-2
        M(jj,jj+(3-x)) = ((12*x)/38);
    end

    %Fill cells with probability of losing a given spin
    for kk = x+1:length(M)-1
        M(kk,kk-x) = 1-((12*x)/38);
    end

    %Win state
    M(y-1:y,y+1) = ((12*x)/38);
    M(y+1,y+1) = 1;

    %End behavior of transition matrix
    ESM = M^1000000000000000000000000;

    %Create column vector with busted-state probs
    B = bust_state(ESM);
    %Create column vector with win-state probs
    W = win_state(ESM);
```

Figure 8: 2:1 Bets

This function has two inputs of "x" and "y". "x" represents the number of 2:1 bets made at one time. It wouldn't make sense for the player to bet 3 or more of these 2:1 bets because a win will result in a net gain of 0 chips(gain 2,lose 2). The logical choices for the "x" parameter are either 1 or 2. "y" represents the DEA and good values for this parameter would be between 20 and 500. Running this script will generate three outputs:

M = Transition matrix based on input parameters.

W = Win-state column vector based on the "steady state matrix"
B = Bust-state column vector based on the "steady state matrix"

If the user wants to find the TPV, they should run the program with the chosen parameters and then investigate the "W" vector. The first row that contains a probability greater than .5 will be considered the TPV.

3.3 Six Line Bet

In this section we will outline the code used in the six-line betting strategy. This strategy involves covering 6 numbers from 2 consecutive rows. This bet has a 5:1 payout.

```
function [M,B,W] = six_line(x,y)
    %Create Matrix
    M = zeros(y+1);

    %Busted state probs
    for ii = 1:x
        M(ii,ii) = 1;
    end

    %Fill cells with probability of winning a given spin
    for jj = x+1:y-(5-x)
        M(jj,jj+(6-x)) = ((6*x)/38);
    end

    %Fill cells with probability of losing a given spin
    for kk = x+1:length(M)-1
        M(kk,kk-x) = 1-((6*x)/38);
    end

    %Win state
    M(y-(4-x):y,y+1) = ((6*x)/38);
    M(y+1,y+1) = 1;

    %End behavior of transition matrix
    ESM = M^1000000000000000000000000;

    %Create column vector with busted-state probs
    B = bust_state(ESM);
    %Create column vector with win-state probs
    W = win_state(ESM);
```

Figure 9: 5:1 Bet

This function has two inputs of "x" and "y". "x" represents the number of 5:1 bets made at one time. It wouldn't make sense for the player to bet 6 or more of these 5:1 bets because a win will result in a net gain of 0 chips(gain 5,lose 5). The logical choices for the "x" parameter are from 1 to 5. "y" represents the DEA and good values for this parameter would be between 20 and 500. Running this script will generate three outputs:

M = Transition matrix based on input parameters.
W = Win-state column vector based on the steady state matrix
B = Bust-state column vector based on the steady state matrix

If the user wants to find the TPV, they should run the program with the chosen parameters and then investigate the "W" vector. The first row that contains a probability greater than .5 will be considered the TPV.

3.4 Bet Numbers

In this section we will outline the code used in the numbers betting strategy. This strategy involves covering any amount of the 38 numbers on the board in one bet. The player wins if one of the chosen numbers is picked. The payout on this bet is 35:1. The player will lose all chips that don't correspond to the winning number. For example, if the player bets 18 numbers and one of those numbers is a winner, the player will win 35 chips, but also lose 17 chips for the numbers selected that weren't the winning number.

```
function [M,B,W] = bet_numbers(x,y)
    %Create Matrix
    M = zeros(y+1);

    %Busted state probs
    for ii = 1:x
        M(ii,ii) = 1;
    end

    %Fill cells with probability of winning a given spin
    for jj = x+1:y-(36-x)
        M(jj,jj+(36-x)) = (x/38);
    end

    %Fill cells with probability of losing a given spin
    for kk = x+1:y
        M(kk,kk-x) = 1-(x/38);
    end

    %Since the matrix must be square, fill cells near the
    %end with probability of winning a given spin AND
    %reaching the "win state"
    M((y+1)-(36-x):y,(y+1)) = (x/38);

    %Win state
    M(y+1,y+1) = 1;

    %End behavior of transition matrix
    ESM = M^1000000000000000000000000;

    %Create column vector with busted-state probs
    B = bust_state(ESM);
    %Create column vector with win-state probs
    W = win_state(ESM);
```

Figure 10: 35:1 Bet

This function has two inputs of "x" and "y". "x" represents the number of 35:1 bets made at one time. It wouldn't make sense for the player to bet 36 or more of these 35:1 bets because a win will result in a net gain of 0 chips(gain 35,lose 35). The logical choices for the "x" parameter are from 1 to 35. "y" represents the "defined exit amount" and good values for this parameter would be between 75 and 1500. Running this script will generate three outputs:

M = Transition matrix based on input parameters.
W = Win-state column vector based on the "steady state matrix"
B = Bust-state column vector based on the "steady state matrix"

If the user wants to find the TPV, they should run the program with the chosen parameters and then investigate the "W" vector. The first row that contains a probability greater than .5 will be considered the TPV.

4 Results/Analysis

4.1 Red/Black

Using the red-black Matlab script, we have created a table of betting one of the 1:1 bets with different DEAs (y variable). All of the 1:1 bets cover 18 numbers, so the probability of winning any given spin will be about .47368.

y	TPV
5	3
10	7
15	11
20	15
25	20
30	24
35	29
40	34
50	44
100	94

Table 2: Red/Black Table

Each value in the TPV column is the amount of chips the player would need in order to have greater than a .5 probability of reaching the DEA(y). For the smaller values of the table, we can see that the distance between the TPV and DEA are about 3 chips. As "y" increases, the player has more possible transition states to get through and thus can get away with starting with less chips. One thing to notice here is that once "y" reaches 30, the TPV becomes 6 less than the DEA. This would suggest that using this type of 1:1 bet, the best rule of thumb would be to start with 6 less chips than one wishes to leave the game with (once the DEA is 30 or more).

4.2 Dozen/Column

Using the dozen-column Matlab script, we have created a table of betting 1 or 2 (x variable) of the 2:1 bets with different DEAs (y variable). Betting a dozen or a column involves covering 12 numbers on the roulette board. The player can choose to bet one of these (which covers 12 numbers) or two (which covers 24 numbers). The probability of winning when betting one dozen is about .31579. The probability of winning when betting two dozens is about .63158.

y	**1**	**2**
10	6	6
15	10	10
20	13	14
25	17	19
30	21	23
40	30	33
50	39	43
60	49	53
100	88	93
150	138	143
500	488	493

Table 3: Dozen/Column Table

In our table, "y" represents the the DEA and the numbered columns represent whether the player bets one or two of these bets per spin. When "y" is a small number, both of the bets seem to do just as well, and based on playing off of the TPV the bets are equivalent. As "y" grows larger however, we start to see a disparity between the two bets. The data suggests that betting one dozen is better for the player than betting two dozens when the DEA starts to grow larger than 15. At the bottom of the table, we can see that a player who bets one dozen can start with 12 chips less than the DEA, whereas a player who bets two dozens has to start with 7 less than the DEA.

4.3 Six-Line

Using the six-line Matlab script, we have created a table of betting 1 to 5 (x variable) of the 5:1 bets with different DEAs (y variable). Betting a six line involves covering 6 numbers from consecutive rows on the roulette board. The player can choose to bet up to 5 of these at once. The probability of winning on a given spin based upon the number of bets(x) is computed as follows: (6*x)/38

y	**1**	**2**	**3**	**4**	**5**
20	12	12	14	14	15
30	19	20	21	22	25
40	26	26	28	30	34
50	33	34	36	40	44
75	53	56	60	63	69
100	75	78	82	88	94
150	121	126	132	138	144
200	170	176	182	188	194
400	369	376	382	388	394
600	569	576	583	588	594

Table 4: Six Line Table

Again, at the lower values for "y" we can see that the bets don't really differ by much. The table suggests at first that the best bets to make based on the TPV method are betting either 1 or 2 six-lines. As "y" increases, we can see the intervals between each TPV increasing between all of the different bet types. As we reach the bottom of the table, it becomes evident that the best way to bet a six-line based on our model is to only bet one six-line at a time.

4.4 Numbers

Using the bet-numbers Matlab script, we have created a table of betting 1 to 38 (x variable) of the 35:1 bets with different DEAs (y variable). When betting numbers on the inside of the board (not using any of the previously discussed bets), the player may choose to bet up to 35 numbers at a time (any more than this will result in guaranteed loss). The values of the top row represent the amount of numbers bet by the player each spin. The probability of winning a given spin based on the amount of numbers chosen(x) is computed as follows: (x/38)

y	**10**	**12**	**14**	**16**	**18**	**19**	**20**	**22**	**24**	**26**	**28**	**30**
75	49	51	51	51	54	57	48	52	51	52	55	57
100	64	64	66	64	64	66	68	68	72	70	72	76
150	94	96	98	98	96	101	102	104	102	108	110	120
200	128	128	132	136	144	137	140	142	144	150	156	164
400	284	292	294	300	310	309	312	320	328	340	352	364
600	460	468	476	484	492	500	504	516	528	538	548	564
1000	844	856	866	880	892	896	900	914	928	938	948	964

Table 5: Number Table

This kind of bet becomes interesting because the player has a lot more options than the previous bet. In order to keep the size of the table manageable, we chose to increase "x" by increments of two and to also include 19 since betting 19 numbers has exactly as .5 chance of winning (coin flip). We also chose to start the table at an "x" value of 10 since most house rules require players betting on the inside (betting numbers) to put up a minimum amount of chips (Most casinos will not allow players to bet one chip on the inside). After looking at the three previous tables one would guess that it would be better to bet less numbers each spin. When considering the smaller values of "y", it's interesting that betting 20/22 numbers yields better results than betting 18/19 numbers. As "y" increases though, the table starts to conform to the pattern that the other tables have shown (betting less numbers is better for the player based off of our model). The disparity between betting numbers becomes more apparent when we look at the last line of the table. When "y" is 1000, a player betting 10 numbers can start with 120 less chips than a player betting 30 numbers!

4.5 Comparison of all Bets

This section will compare the four different betting strategies that we have looked at to each other in order to gain more insight about which bets are better for the player to make based on our model.

Red/Black vs Dozen/Column:

y	**Red/Black(y)**	**Dozen/Column(1,y)**
10	7	6
15	11	10
20	15	13
30	24	21
50	44	39

Table 6: Red/Black vs Dozen/Column

This table compares the red/black bet to betting 1 column or dozen. For the lowers values of "y" we can see that the dozen/column bet is slightly better for the player as they can start with 1 or 2 less chips than the player who uses a red/black betting strategy. As "y" grows larger, it becomes more obvious that the dozen/column is a better bet than red/black.

Red/Black vs Six Line:

y	**Red/Black(y)**	**Six Line(1,y)**
20	15	12
30	24	19
40	34	26
50	44	33
75	69	53

Table 7: Red/Black vs Six Line

This table compares the red/black bet to betting 1 six-line. At the lower values of "y" we can already see that the six-line is a better bet for the player, even more so than the dozen/column was. As "y" increases we can really see how much better the six-line bet is than the red/black bet. At "y"=75, the player betting the six-line could start with 16 less chips than the red/black player!

Dozen/Column vs Six Line:

y	**Dozen/Column(1,y)**	**Six Line(2,y)**
20	13	12
30	21	20
40	30	26
50	39	34
75	63	55

Table 8: Dozen/Column vs Six Line

This table compares betting 1 dozen/column vs. 2 six-lines. Both of these bets cover 12 numbers. Again, at the lower "y" values we can see only a slight advantage for betting 2 six-lines over 1 dozen/column. As "y" increases it becomes much more apparent that betting 2 six-lines is better for the player than betting a single dozen/column. At "y"=75, the player betting 2 six-lines was able to start with 8 less chips than the player betting the single dozen/column.

Red/Black vs Numbers:

y	**Red/Black(y)**	**Numbers(18,y)**
50	44	36
75	69	54
100	94	64
150	144	96
200	194	144

Table 9: Red/Blacks vs Numbers

This table compares betting red/black to betting 18 numbers on the inside. These two bets are comparable because they both cover 18 numbers on the board. It becomes clear from looking at the table that betting numbers on the inside is far more favorable for the player than using a red/black bet. At "y"=200, the player betting numbers on the inside was able to start with 50 chips less than the player betting red/black!

Dozen/Column vs Bet Numbers:

y	**Dozen/Column(1,y)**	**Numbers(12,y)**
50	39	36
75	63	51
100	88	64
150	138	96
200	144	128

Table 10: Dozen/Column vs Numbers

This table compares betting 1 dozen/column vs. betting 12 numbers on the inside. These two bets both cover 12 numbers on the board so it would be interesting to compare them. For the lower values of "y", we can see that betting numbers on the inside only provide a slight edge over betting a dozen/column. As "y" grows larger, however, we can see that again betting numbers on the inside is more favorable for the player.

Six Line vs Bet Numbers:

y	**Six Line(1,y)**	**Numbers(6,y)**
50	33	32
75	53	48
100	75	64
150	121	90
200	170	126

Table 11: Dozen/Column vs Numbers

In this table we compare betting 1 six-line vs. betting 6 numbers on the inside. These bets are comparable because they both cover 6 numbers on the board. For the smaller values of "y" we can see that betting numbers on the inside have only a slight advantage over betting a single six-line. As we'd expect, the larger "y" gets, the greater the disparity between the two bets. Betting numbers on the inside is better for the player in this scenario.

5 Conclusions

The entirety of this report is focused on analyzing which bets are best for the player based on our TPV model. Models with different designs might have different results, but in this particular instance we were able to draw concrete conclusions about the four roulette bets outlined in this paper. It is important to note that the TPV metric is the number of starting chips that give the player greater than a 50 percent chance of reaching the DEA before going broke. Due to the nature of randomness, these conclusions in no way guarantee that the player will walk away a winner every time. The intention of these conclusions are to highlight some patterns and relationships between different roulette bets, as well as help players make more informed decisions when placing bets at the table.

It would seem that the higher the payout is for the player, the better the bet is for our model. All of the comparative tables from section 4.5 that involved the red/black betting strategy showed that each of the other bets were more favorable. The difference between the player's starting amounts was huge when comparing red/black to betting numbers on the inside (Table 9). As we compared the other betting strategies, it became obvious that the higher payout bets were better for the player based on our model.

Another fact to take note of is that within each bet type that allowed the player to choose the amount of bets made, it was always more favorable to make less of that particular bet. For example, if a player were to decide whether to bet 1,2,3,4 or 5 six-lines per spin using our model, the best choice for the player would always be to choose 1 six-line at a time (Refer to Table 4). This same conclusion is true for the dozen/column bet, as well as the numbers bet. The downside to this type of betting is that the actual probability of winning a given spin is lower since less numbers are covered. In actual game-play, this might be a little more stressful for the player because they will experience more losses. However, these losses are much smaller than if the player were making more than one of the same bet type. With this being said, if the player can stomach these small losses, they will be better off in the long run based on our model.

These conclusions are interesting because a lot of newer roulette players will tend to stick with the 1:1 bets as they seem more like a coin flip, but won't realize how much worse those bets are than betting numbers on the inside. The bottom line here is that if players are operating based on our model, they should always bet numbers on the inside and it would be most optimal to bet the fewest amount of numbers allowed by the house rules.

Out of these results a natural question arises: If the player plays this strategy repeatedly, will they be able to beat the casino in the long run? The answer to this question is no. Although we have found a point at which the probability of a favorable outcome for the player is above 50%, all we have to do is take a look at the difference between net gains and net losses. For instance, when a player uses this system betting 10 numbers on the inside with a DEA of 600 (Refer to Table 5), slightly more than 50% of the time they will walk away with a net gain of 140 chips. However, slightly less than 50% of the time the player will walk away with a net loss of 600 chips. Since we are analyzing these bets at a point where the odds of reaching the DEA or going broke are almost a coin flip, it becomes apparent why the player will still lose in the long run. The important thing to take away from this report is which bets are better for the player, not

which bets allow the player to beat the casino in the long run.

These conclusions were made based upon the Matlab code used and on the fact that whatever bet type is chosen by the player will be played until they either reach the DEA or don't have enough chips to place another bet of the same type. The Matlab code itself can definitely be improved and more scripts can be written to cover the other various bets in the game of roulette. It is our hope that the readers try to run the code themselves as there is much room to improve and learn from. It would be interesting to analyze combining certain bet types and comparing them (For example, betting 10 numbers on the inside AND a six line), but that lies outside the scope of this particular report.

6 References

[1] Kavouras, Ioannis. (2018) "How to Play Roulette: Rules, Odds and Payouts." Online Casino, 888. Retrieved from: https://www.888casino.com/blog/how-to-play-roulette-beginners-guide

[2] Vicapow. "Markov Chains Explained Visually." Explained Visually. Retrieved from: https://setosa.io/ev/markov-chains/