**STU22004 Applied Probability**

**Group Project**

Abigail Cisowska - 20333898

Daniel Penrose - 20331752

Niall Connolly - 20332921

C Okafor - 20332400

# Question 1

The aim of this question is to simulate 4 possible games or systems of betting on the roulette wheel and comparing the data under certain criteria from these simulations between each game.

Below is an analysis of the data received from simulating these games 10,000 times in excel.

## Game 1: Betting on Red

Seeing as there are a total of 37 slots on the roulette wheel, and 18 of them are coloured red, the probability of the ball landing on red is or 0.486 rounded to the 3 decimal places.

Using this information, we can use the RAND() function in excel to generate random values between 0 and 1, and by checking if these values are less than or equal to the calculated probability of the ball landing on red (0.486), we can determine whether the bet is lost or won. Using this, we can also determine which specific colour the ball landed on.

Using this method of simulation, we can repeat the bets 10,000 times and calculate the 5 criteria for the game.

Here are 5 simulations of the game to show as an example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Reps** | **RAND()** | **Colour** | **Result** | **$ Won** | **Cumulative Winnings** |
| 1 | 0.138 | Red | Win | 1 | 1 |
| 2 | 0.776 | Black | Loss | -1 | 0 |
| 3 | 0.220 | Red | Win | 1 | 1 |
| 4 | 0.515 | Black | Loss | -1 | 0 |
| 5 | 0.854 | Black | Loss | -1 | -1 |

### **Tasks (a) and (b)**

#### C1. The expected winnings per game

By totalling the amount of games won over the 10,000 simulations we can calculate the expected winnings per game by using the following formula:

Where represents winning the bet.

Using this formula, we get a value for the expected winnings per game of 0.99$ (this is just for one simulation of 10,000 bets, this number can vary between simulations).

We are also asked to calculate the exact winnings per game for this game, which turns out to be 0.97$.

We can calculate the percentage error of these values by using the following formula:

Where represents the exact value and represents the estimated value.

Using this formula with the values calculated we get a percentage error of 2.05% rounded to 2 decimal places.

#### C2. The proportion of games you win

The proportion of games won can be calculated simply by dividing the total amount of games won out of the 10,000 simulated games by the total amount of games played, which would be 10,000.

The resulting value for this simulation is 0.488 rounded to 3 decimal places. This value can change depending on the simulation results.

The exact value for this criteria is 0.486, which is also the probability of winning this game.

Using the same formula as above we get a percentage error for the proportion of games won of 0.393%.

#### C3. The expected playing time per game

Since this game involves placing only one bet per game, the expected playing time per game for betting on red is 1 bet.

#### C4. The maximum amount of money you can lose

Each game only takes the time of 1 bet, meaning that the maximum amount of money you can lose is the bet you placed, which is 1$. Over the 10,000 repetitions of the game, the maximum amount of money you can lose is 1,000$.

#### C5. The maximum amount of money you can win

Each game only takes the time of 1 bet, meaning that the maximum amount of money you can win is 2$. Over the 10,000 repetitions of the game, the maximum amount of money you can win is 2,000$.

### **Task (c)**

I believe that the estimates are reliable based on the plot above, as they correlate to the variance of winnings calculated. We can also see that the graph initially shows large spikes, but flattens out and normalises as the simulations are repeated.

### **Task (d)**

The variance of winnings calculated was 2.246, which correlates with the graph shown above. The variance is fairly low, which is to be expected since in this game you can either lose 1$ or win 2$.

The variance of wins is 0.000171.

Since the playing time for this game is 1 bet, the variance of expected playing time is 0.

## Game 2: Betting on a Number

There are a total of 37 slots on the roulette wheel, and all of them are numbered, which means that if you are to bet on a specific number, the probability of you winning the bet would be or 0.027 rounded to 3 decimal places.

The simulation uses the RANDBETWEEN() function in excel to determine which number is being bet, and which number the ball lands on. Comparing these two numbers in each repetition, we can determine a win or loss.

Here is a table representing 5 repetitions of the game:

(The number chosen to bet on for these repetitions was 24)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reps** | **Number** | **Result** | **$ Won** | **Cumulative Winnings** |
| 1 | 28 | Loss | -1 | -1 |
| 2 | 5 | Loss | -1 | -2 |
| 3 | 18 | Loss | -1 | -3 |
| 4 | 19 | Loss | -1 | -4 |
| 5 | 35 | Loss | -1 | -5 |

### **Tasks (a) and (b)**

#### C1. The expected winnings per game

Much like the previous game, we can total the amount of games won from the simulation and use the formula to calculate the estimated expected winnings per game. This turns out to be 0.97$.

We can also calculate the exact expected winnings per game the same way as previously, which gives us a value of 0.95$.

The percentage error of the expected winnings per game between the estimated and exact values is 2.11% rounded to 2 decimal places.

#### C2. The proportion of games won

The estimated proportion of games won for this simulation is 0.028, while the exact proportion is 0.027.

This means the percentage error for this criteria is 3.70% rounded to 2 decimal places.

#### C3. The expected playing time per game

Since this game involves placing only one bet per game, the expected playing time per game for betting on a number is 1 bet.

#### C4. The maximum amount of money you can lose

Each game only takes the time of 1 bet, meaning that the maximum amount of money you can lose is the bet you placed, which is 1$. Over the 10,000 repetitions of the game, the maximum amount of money you can lose is 1,000$.

#### C5. The maximum amount of money you can win

Each game only takes the time of 1 bet, meaning that the maximum amount of money you can win is 35$. Over the 10,000 repetitions of the game, the maximum amount of money you can win is 35,000$.

### **Task (c)**

I believe that the estimates are reliable based on the plot above as they line up with the calculated variance of winnings. Once again we can see large spikes for the variance at the beginning of the simulations, which flattens out as the simulations continue.

### **Task (d)**

The calculated variance of winnings is 34.1698. This shows that the discrepancy in winnings across games played is much higher than game 1, since you can either lose 1$ or win 35$.

The variance of wins is 0.0000154508.

Once again, the variance of expected playing time is 0, since the playing time for each game is 1 bet.

## Game 3: Martingale System

This system used multiple bets per game. There are a total of 37 slots on the roulette wheel, and 18 of them are coloured red, the probability of the ball landing on red is or 0.486 rounded to the 3 decimal places. For this system we will be betting on red every time. This simulation uses the RAND() function in excel to determine the outcome of each bet. When a bet is lost, the amount bet is doubled. When a bet is won the amount bet goes back to 1.

Below is a table showing 5 simulations of the game

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Games** | **Result** | **No. of Bets** | **$ Won / Lost** | **Cumulative Winnings** |
| 1 | Win | 19 | $10 | $10 |
| 2 | Win | 17 | $10 | $20 |
| 3 | Win | 20 | $10 | $30 |
| 4 | Win | 27 | $10 | $40 |
| 5 | Win | 10 | $10 | $50 |

### **Task (a)**

#### C1. The expected winnings per game

The expected value of the winnings per game was calculated using the formula mentioned in previous games. Using this formula, we get the expected value to be $9.06 during these 10,000 repetitions.

#### C2. The proportion of games won

The proportion of games won in this case came out to be 0.9056. This indicates that approximately 90.56% of the time, using this system will result in a win.

#### C3. The expected playing time per game

The expected playing time was again calculated using the expected value formula across all 10,000 games. This came out to be 19.3958, meaning that on average one can expect to make 19 bets per game.

#### C4. The maximum amount of money you can lose

The maximum amount of money that can be lost with this system can be calculated by losing every bet placed. Since this value would be cumulative (as the loss adds up), this becomes the sum of 1 and every multiple of 2 upto and including 128, since the game is lost when the amount bet reaches $100. The maximum amount of money you can lose is $255. Across all 10,000 games, this results in a loss of $2,550,000.

#### C5. The maximum amount of money you can win

Since the game is won when the total winnings reaches 10, the maximum winnings for the martingale system is $10.

### **Task (c)**

I believe that the estimates are reliable based on the plot, as they line up with the calculated variance of winnings. As before there is a large spike early in the simulations which flattens out.

### 

### **Task (d)**

The variance of winnings, as shown in the graph, came out to be 3142.207. This large variance indicates a sizable discrepancy in winnings across games played.

The variance of wins is approximately 3.960E-05.

The variance of expected playing time comes out to be approximately 16.544. This indicates that the number of bets made does not have a large amount of variability.

## Game 4: Labouchere System

This system used multiple bets per game. Loops are difficult to use while using regular Excel due to “circular references”, so it was instead decided to use Microsoft Visual Basic for Applications (hereinafter known as VBA), a programming language within Excel. These values were then written to the main spreadsheet using VBA’s *Worksheets.Range* property. Due to this, the game could be repeated 10,000 times and information about the game (winnings, amount of bets, win or loss, and amount spent on betting) could be saved. All further calculations were made using these four columns of data. For the bets, it was decided that a random colour (red or black) would be selected by the player, and a random colour would also be selected by the computer.

Here are the first 5 simulations of the game.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bet turn** | **Winnings** | **# Bets** | **W/L** | **Amount spent on betting** | **Total after bet** |
| 1 | 10 | 2 | 1 | -10 | 20 |
| 2 | 239 | 23 | 1 | -10 | 249 |
| 3 | 10 | 2 | 1 | -10 | 20 |
| 4 | 35 | 8 | 1 | -10 | 45 |
| 5 | 38 | 8 | 1 | -10 | 48 |

### **Task (a)**

#### C1. The expected winnings per game

The expected value of the winnings per game was calculated using the formula mentioned in previous games. Using this formula, we get the expected value to be 58.6508 during these 10,000 repetitions.

#### C2. The proportion of games won

The proportion of games won in this case came out to be 0.9856. This indicates that approximately 98.6% of the time, using this system will result in a win.

#### C3. The expected playing time per game

The expected playing time was again calculated using the expected value formula across all 10,000 games. This came out to be 8.2771, meaning that on average, one can expect to make 8 bets per game.

#### C4. The maximum amount of money you can lose

The maximum amount of money that can be lost with this system can be calculated by losing every bet placed. Since this value would be cumulative (as the loss adds up), this becomes the sum of every number from 5 to 100. Since the game is lost when the amount bet reaches $101 (which is not counted), the maximum amount of money you can lose amounts to $5040. Across all 10,000 games, this results in a loss of $50,400,000.

#### C5. The maximum amount of money you can win

The maximum amount of money that can be won is calculated similarly to the method above. This can be calculated by first losing every game until the betting amount is equal to $99. This value never reaches $100 as that would result in the next bet exceeding the stated limit of $100, resulting in a lost game. Then, every game afterwards must be won until there are no values left in the list. As the winnings are also cumulative, the maximum amount of money that can be won is the sum of every number from 1 to 99, which results in a maximum winning of $4950. Across all 10,000 games, this results in a loss of $49,500,000.

An interesting thing to note is that losing every game until the bet amount is equal to $99 results in a loss of $4940, as this would be the sum of every value from 5 to 99. This means that, while the maximum amount of money that can be won is $4950, the amount one would actually win would be $10, since the winnings don’t take into account the amount spent on bets. This happens to be the sum of the original four numbers (1 to 4).

### **Task (c)**

### Chart

I believe the estimates received to be reliable based on the plot. The running variance appears to tend towards 8750, and the final result for variance is known to be about 8505. Similarly, the running average appears to tend towards 60, and the final result for the expected value is known to be about 58. Based on this, I believe that the estimates are reliable.

### **Task (d)**

The variance of winnings, as shown in the graph, came out to be 8505.775. This large variance indicates a large discrepancy in winnings across games played.

The variance of wins is approximately 0.0142. Using the Labouchere system in the way specified by the assignment sheet makes it remarkably easy to win games, even if a large number of games only result in an actual winning of $10. The player essentially has an infinite amount of money to start, so even though they may lose a game, there is no clear indication of whether or not this affects them.

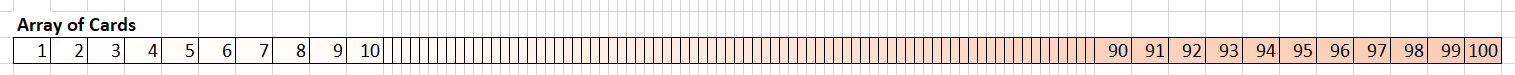
The variance of expected playing time comes out to be approximately 52.102. This indicates that the number of games played does not have a large amount of variability, but still differs somewhat.

Compared to the other three betting systems, the Labouchere system is the most variable when considering both winnings and playing time.

# Question 2

### **The cards**

#### Representing the cards

I first make an array of 100 values, 1 for each card.

*[Fig. 1] The 100 cards. Numbers 11-89 are hidden for readability. Using conditional formatting, all numbers have been assigned a colour ranging from white (1) to light orange (100). This will come up again later.*

#### Shuffling the cards

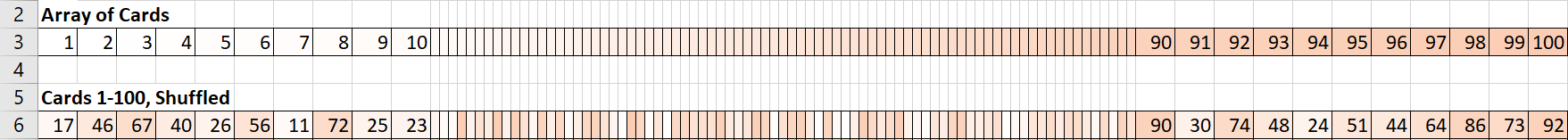
Now that we have a representation of our cards, we need to shuffle them. I first generate 100 random numbers using **RAND()**.

*[Fig. 2] Some of the random numbers contained in the cells. Not all of these are visible, for readability.*

Then I put the following formula into 100 cells. (**B6**) refers to one of the 100 cells in fig. 2:

**=INDEX($CZ$3:$GU$3, RANK(B6, $B6:$CW6))**

This will generate the numbers 1-100 in each cell, with no repeats, then use that number as an index to return the value of the card found at that position in the card array.

Using this formula, we now have the 100 cards in a random order!

*[Fig. 4] Cards 1-100 in an array, followed by them shuffled. For readability, the middle 79 columns are hidden. Assigning each number a colour, as we did in fig. 1, makes it easy to see that the numbers are randomised.*

### **The rules**

In the game, you draw all 100 cards from the deck, one at a time. The player keeps track of how many cards have been drawn. If a card drawn has a value equal to the number of cards drawn, that counts as a “hit”. So how do we represent this in excel?

Simple! If we check any shuffled card in fig. 4 to the card in the card array directly above it, we can see whether or not it “hits”.

**=IF(CZ6=CZ$3, 1,0)** returns 1 if (**CZ6**) “hits”, otherwise it returns 0. **CZ6** is the first card in the 100 shuffled cards we generated in fig. 2, and **CZ$3** is the first card in our card array from fig. 1. Using autofill on the next 99 cells, we can now see which cards have “hit”!  


*[Fig. 5] Array containing “hit” data. For readability, the middle 79 columns are hidden. Using conditional formatting, every cell containing a “hit” is coloured in green.*

### **The results**

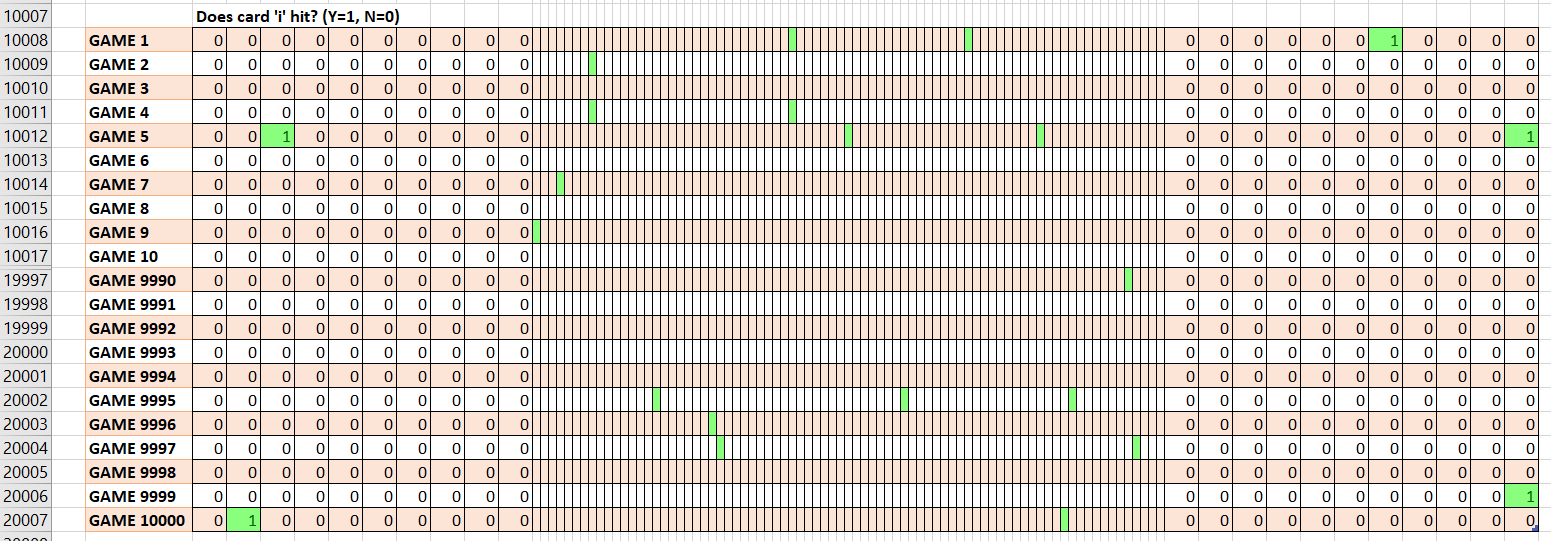
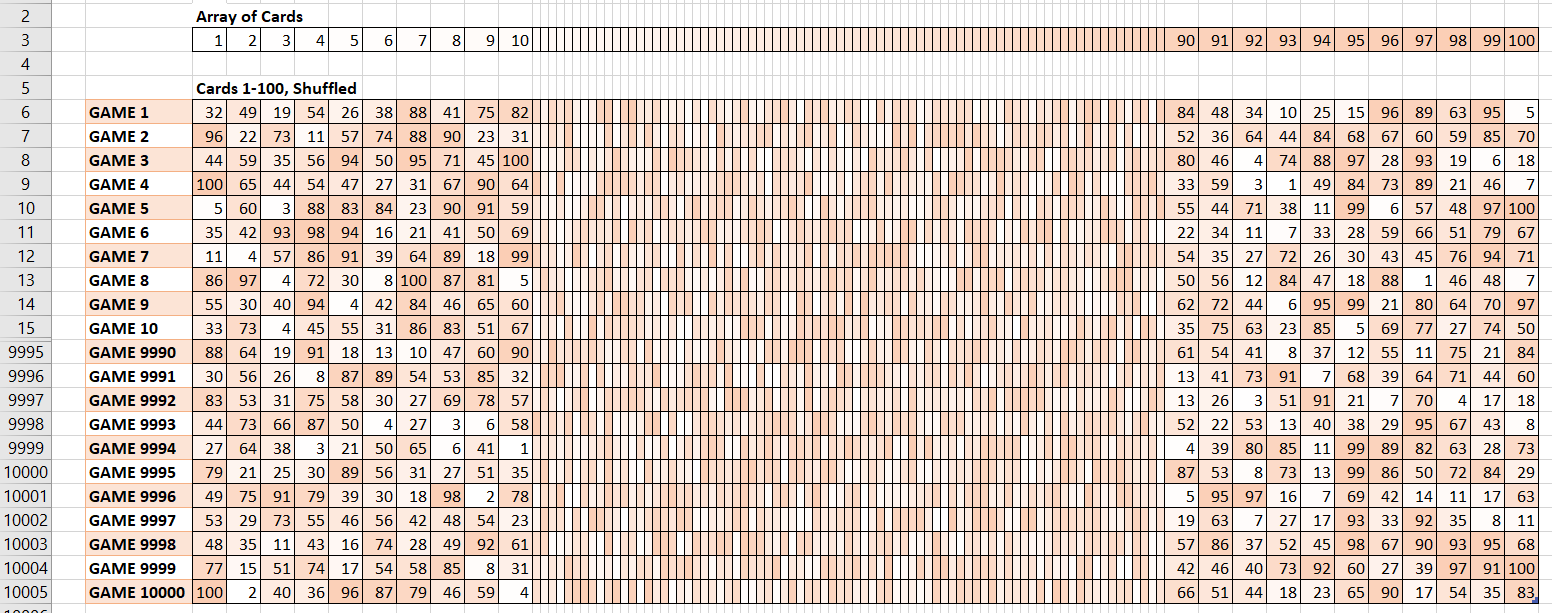
#### Total hits

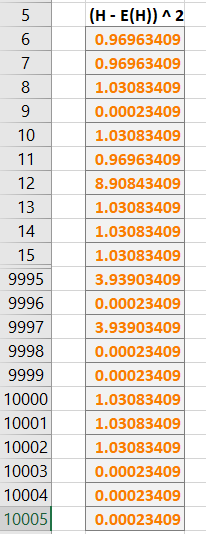
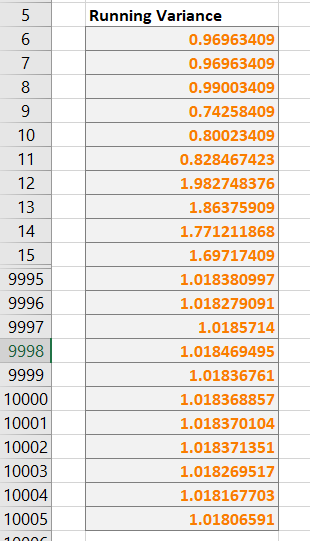
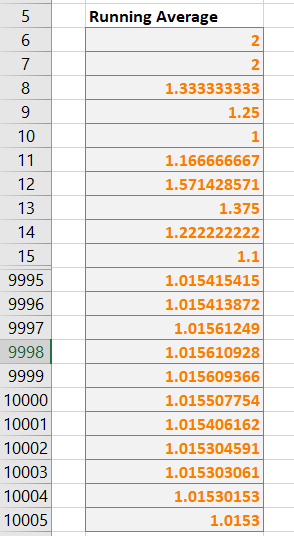
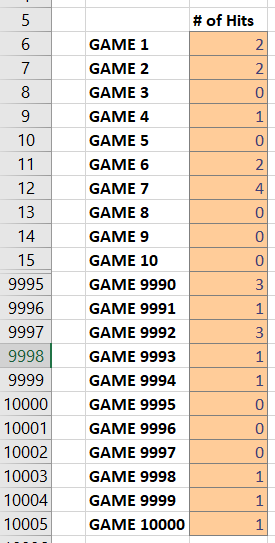
Now we need to find how many hits we’ve got. This can be done by just adding up all of the values found in the array in fig. 5. Passing them into **SUMPRODUCT()** does just that.



*[Fig. 6] The total number of hits.*

#### Running it 10,000 times

*[Fig. 7,8] What 10,000 runs of the simulation look like in excel.*

**

*[Fig. 9] Total number of hits for each game.  
[Fig. 10] Running average of numbers found in fig. 9.  
[Fig. 11] Running variance of numbers found in fig. 9.  
[Fig. 12] Calculations used for fig. 11*

Expected number of hits

The expected number of hits should be equal to the average total number of hits. Therefore, we can find it by making a running average. Excel has a built in function called **AVERAGE()** that we can use to find a running average.

If we have a column containing every hit count that starts on **GX6**, we can write **AVERAGE($GX$6:GX6)** into one cell, and copy it downward 10,000 times, as I did in fig. 10. This will give us a running average.

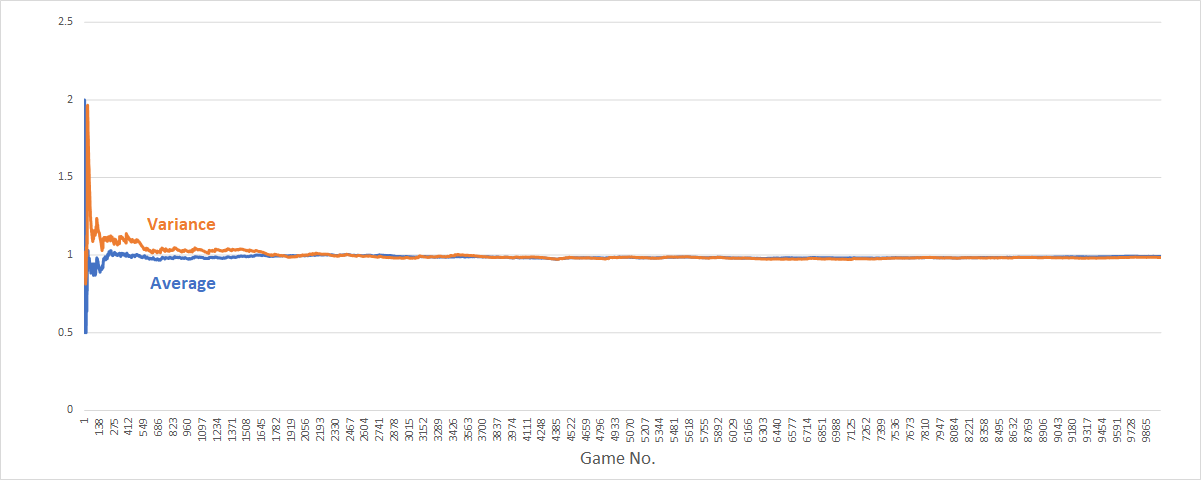
#### Hit variance

Next we’ll find the variance of hit totals.

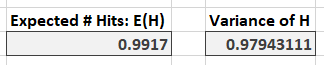
First we need to do a small calculation. Given one of the hit totals, ‘H’, and its expected value, ‘E(H)’, subtract them, then square the result. Do this for every hit total, as seen in fig. 12.

Then for each value in fig. 12, find the sum of every value in fig. 12 up to that point and divide it by which row in the array you’re in. I wrote that as **SUM($HB$6:$HB6)/(ROW($GW6)-5)**.

The final value should be your variance.



*[Fig. 13] Running average & variance for hit totals. As you can see, they both converge towards 1.*

*[Fig. 14] The expected value for H and the variance of H.*

#### Hit/Miss Ratio

**SUMPRODUCT(HIT\_TOTALS) / (10000\*100)  
&":"& ((10000 \*100)-SUMPRODUCT(HIT\_TOTALS)) / (10000\*100)**

#### Largest number of hits

**INDEX(GAME\_NUMBER\_ARRAY, MATCH(MAX(HIT\_TOTALS), HIT\_TOTALS, 0))  
&", "& MAX(HIT\_TOTALS) &" hits"**



*[Fig. 14] Hit:Miss ratio & the largest number of hits.*

### **Observations**

* The expected number of hits converges towards 1.
* The variance for the number of hits converges towards 1.
* The Hit/Miss ratio converges towards 0.01:1, which makes sense, as each card has a 1/100 chance of "Hitting".