**Lab 6: High Roller**

**(a.k.a. Baby needs a new pair of shoes, or a new version of MATLAB)**

1. **Background Information**

One of the powerful uses of computers is to perform simulations of experiments or systems prior to going to the lab or building a prototype to prove that an idea might work. Many companies are moving in this direction, such as P&G, which has partnered with the University of Cincinnati on the UC Simulation Center (<http://www.min.uc.edu/ucsc>), which employs students, faculty, and researchers to work on simulations for a variety of P&G projects. The reason computers are so useful in these situations is that they can perform complex calculations quickly and efficiently and can perform multiple rounds of simulation to gather additional data. While it might take weeks to perform and analyze a series of crash tests on a new vehicle, a computer can simulate thousands of crashes under different conditions in a fraction of the time (without the need to destroy a car in the process!).

In this lab, you will be running a series of simulations to better understand a fairly simple system: rolling dice. In part B, you will simulate rolling a single die many times and determine the probability of rolling each number (1 through 6). In part C, you will simulate rolling two dice and determine the probability of rolling the combined values (2 through 12). Last, in part C, you will simulate rolling five dice and determine the probability of achieving a pair, three of a kind, four of a kind, and five of a kind.

|  |  |
| --- | --- |
| https://upload.wikimedia.org/wikipedia/commons/thumb/3/36/Two_red_dice_01.svg/2000px-Two_red_dice_01.svg.png | http://www.math.ttu.edu/~aledet/images/ordinary-die.gif |
| **(a)** | **(b)** |
| **Figure 1: (a) a typical pair of dice and (b) a projection showing the 6 sides of a typical die** | |

1. **Rolling a Single Die**

Write a script that simulates the rolling of a single die a user-specified number of times.

1. Start off your script by seeding the random number generator using the following command:  
     
   rng('shuffle');  
     
   This will ensure that each time the simulation is run, a new set of values will be produced for the die rolls.
2. Create a way to store the counts of how many times each number on the die appears
3. Prompt the user for the number of times to simulate rolling the die
4. Set up a for loop to simulate the rolling of the die the number of times specified by the user and keep track of how many times each number appears
   1. To simulate a die roll, use the randi command as follows:  
        
      roll = randi([1 6],1);  
        
      This command will generate a single random integer between 1 and 6 and store that value in the variable roll.
5. Compute and display the probabilities (as a percentage) of rolling each of the six numbers on a die.

**Based on your own knowledge and experience with dice, what should the probability be for rolling each number on the die?**

|  |  |
| --- | --- |
| **Value** | **Expected Probability** |
| 1 | 16.67% |
| 2 | 16.67% |
| 3 | 16.67% |
| 4 | 16.67% |
| 5 | 16.67% |
| 6 | 16.67% |

**Now, test your hypothesis by running your script with the following number of rolls and record the probabilities you calculate:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of Rolls: 10** | | **Number of Rolls: 100** | | **Number of Rolls: 10000** | |
| **Combination** | **Probability** | **Combination** | **Probability** | **Combination** | **Probability** |
| 1 | 10% | 1 | 19% | 1 | 16.52% |
| 2 | 30% | 2 | 13% | 2 | 16.79% |
| 3 | 20% | 3 | 17% | 3 | 16.31% |
| 4 | 30% | 4 | 18% | 4 | 17.01% |
| 5 | 0% | 5 | 21% | 5 | 17.09% |
| 6 | 10% | 6 | 12% | 6 | 16.28% |

**How do the results of your simulations compare to the values you expected initially?**

|  |
| --- |
| The results of only rolling 10 times is not very accurate but the 10000 roll data comes fairly close with the 100 roll data being not nearly as accurate as 10000 but still much closer than only 10 rolls. |

**Paste your script below:**

%Models Lab 6

clear; clc;

rng('shuffle');

%start the counts at 0 and index them as the die is rolled

count1 = 0;

count2 = 0;

count3 = 0;

count4 = 0;

count5 = 0;

count6 = 0;

count\_rolls = input('How many times should the die be rolled? ');

%roll the dice and index the variables

for num\_rolls = 1:count\_rolls

die1 = randi([1 6],1);

switch die1

case 1

count1 = count1 + 1;

case 2

count2 = count2 + 1;

case 3

count3 = count3 + 1;

case 4

count4 = count4 + 1;

case 5

count5 = count5 + 1;

case 6

count6 = count6 + 1;

end

end

%find the percentages

percent1 = count1/count\_rolls\*100;

percent2 = count2/count\_rolls\*100;

percent3 = count3/count\_rolls\*100;

percent4 = count4/count\_rolls\*100;

percent5 = count5/count\_rolls\*100;

percent6 = count6/count\_rolls\*100;

%print out the percentages

fprintf('Roll percentages - 1: %05.2f, 2: %0.2f, 3: %0.2f, 4: %0.2f, 5: %0.2f, 6: %0.2f\n',percent1,percent2,percent3,percent4,percent5,percent6);

1. **Rolling Two Dice**

Now that you’ve explored the rolling of a single die, let’s increase the complexity slightly and add a second die into the mix. Write a script that simulates the rolling of a two dice a user-specified number of times. You may want to use your script from Part B as a starting point, but don’t overwrite it as you’ll need to turn it in with your submission for this lab.

1. Start off your script by seeding the random number generator using the following command:  
     
   rng('shuffle');
2. Create a way to store the counts of how many times the combined total of the dice appears (2 – 12)
3. Prompt the user for the number of times to simulate rolling the dice
4. Set up a for loop to simulate the rolling of the dice the number of times specified by the user and keep track of how many times each combined value appears
   1. Since you are rolling two dice now instead of the single die, you will need to generate two random numbers, which can be done either by using two randi commands or by using a single randi command and creating a vector with two random integers
5. Compute and display the probabilities (as a percentage) of rolling each of the 11 combinations on a die.

Before running your simulations, think about the situation and what you expect to happen by filling in the following tables.

**How many different combinations are there to achieve the each of the possible rolls? Do not count the same value on different die as two combinations. For instance, if you roll a 3 on the first die and a 3 on the second die, do not count this as two combinations. However, you should count rolling a 4 and a 3 and rolling a 3 and a 4 on the first and second dice, respectively, as two separate combinations.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Value** | **Number of Combinations** | **Value** | **Number of Combinations** |
| 2 | 1 | 7 | 6 |
| 3 | 2 | 8 | 5 |
| 4 | 3 | 9 | 4 |
| 5 | 4 | 10 | 3 |
| 6 | 5 | 11 | 2 |
|  |  | 12 | 1 |

**Given this information, which combination(s) would you expect to have the highest probability and which would you expect to have the lowest probability?**

|  |
| --- |
| 7 has the highest probability. 2 and 12 have the lowest probabilities. |

**Now, run your script for the following number of rolls and record the probabilities of achieving each combination:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of Rolls: 10** | | **Number of Rolls: 100** | | **Number of Rolls: 10000** | |
| **Combination** | **Probability** | **Combination** | **Probability** | **Combination** | **Probability** |
| 2 | 0% | 2 | 0% | 2 | 2.80% |
| 3 | 0% | 3 | 3% | 3 | 5.54% |
| 4 | 10% | 4 | 6% | 4 | 7.61% |
| 5 | 20% | 5 | 15% | 5 | 11.36% |
| 6 | 10% | 6 | 21% | 6 | 14.43% |
| 7 | 0% | 7 | 19% | 7 | 16.81% |
| 8 | 10% | 8 | 15% | 8 | 13.36% |
| 9 | 40% | 9 | 7% | 9 | 11.07% |
| 10 | 10% | 10 | 7% | 10 | 8.57% |
| 11 | 0% | 11 | 5% | 11 | 5.58% |
| 12 | 0% | 12 | 2% | 12 | 2.87% |

**How do the results of your simulations compare to the values you expected initially?**

|  |
| --- |
| Similarly to before as the sample size of rolls gets higher, the data becomes much more accurate. Here however, the 100 rolls were not adequately representative. The 10000 rolls do maintain fairly accurate proportions for the different numbers according to my predictions. |

**Paste your script below:**

%Models Lab 6 Problem 2

clear; clc;

rng('shuffle');

%start the counts at 0 and index them as the dice are rolled

count2 = 0;

count3 = 0;

count4 = 0;

count5 = 0;

count6 = 0;

count7 = 0;

count8 = 0;

count9 = 0;

count10 = 0;

count11 = 0;

count12 = 0;

count\_rolls = input('How many times should the die be rolled? ');

%roll the dice and index the variables

for num\_rolls = 1:count\_rolls

die1 = randi([1 6],1);

die2 = randi([1 6],1);

dice\_roll = die1 + die2;

switch dice\_roll

case 2

count2 = count2 + 1;

case 3

count3 = count3 + 1;

case 4

count4 = count4 + 1;

case 5

count5 = count5 + 1;

case 6

count6 = count6 + 1;

case 7

count7 = count7 + 1;

case 8

count8 = count8 + 1;

case 9

count9 = count9 + 1;

case 10

count10 = count10 + 1;

case 11

count11 = count11 + 1;

case 12

count12 = count12 + 1;

end

end

%find the percentages

percent2 = count2/count\_rolls\*100;

percent3 = count3/count\_rolls\*100;

percent4 = count4/count\_rolls\*100;

percent5 = count5/count\_rolls\*100;

percent6 = count6/count\_rolls\*100;

percent7 = count7/count\_rolls\*100;

percent8 = count8/count\_rolls\*100;

percent9 = count9/count\_rolls\*100;

percent10 = count10/count\_rolls\*100;

percent11 = count11/count\_rolls\*100;

percent12 = count12/count\_rolls\*100;

%print out the percentages

fprintf('Roll percentages - 2: %0.2f, 3: %0.2f, 4: %0.2f, 5: %0.2f, 6: %0.2f,\n 7: %0.2f, 8: %0.2f, 9: %0.2f, 10: %0.2f, 11: %0.2f, 12: %0.2f\n',percent2,percent3,percent4,percent5,percent6,percent7,percent8,percent9,percent10,percent11,percent12);

1. **Rolling Multiple Dice**

While rolling single and pairs of dice are fairly common in many games, such as Craps or Over 7 Under 7, there are a number of games that require the use of more dice. For instance, Yahtzee requires the players to roll 5 dice and each player scores points based on the different combinations they roll. Particularly, you will explore the probability of rolling a pair, three-of-a-kind, four-of-a-kind, and five-of-a-kind when rolling 5 dice.

Write a new script that will perform this simulation a user-specified number of times. Again, you may want to use one of your previous scripts as a starting point, but don’t overwrite them!

1. Start off your script by seeding the random number generator using the following command:  
     
   rng('shuffle');
2. Create a way to store the counts of how many times the combination of dice produces one of the four outcomes in which we are interested (pair, three-of-a-kind, etc.)
3. Prompt the user for the number of times to simulate rolling the dice
4. Set up a for loop to simulate the rolling of the dice the number of times specified by the user and keep track of how many times each combined value appears
   1. Now that you are dealing with 5 dice, it may be easier to create a vector of random numbers rather than five individual variables for each die; use the following command to create a vector of 5 random integers:  
        
      rolls = randi([1 6],[1 5]);
   2. When it comes time to determine what you have rolled, first sort the dice into ascending order; this can be done using the sort command:  
        
      rolls = sort(rolls);
   3. With a sorted vector of dice values, you can fairly easily determine which outcome you have achieved (if any) by thinking about the length of consecutive values that must be the same in order to achieve the outcome  
        
      **Hints:**  
      **Five-of-a-kind** means you have all five values the same, so if the first value and the last value in the ***sorted vector*** are the same, then you know the roll must be five-of-a-kind

**Four-of-a-kind** is a bit trickier as there are a couple of possibilities for the sorted

vector:

- The first four values could be the same (e.g. [ 2 2 2 2 6])

**or**

- The last four values could be the same (e.g. [4 3 3 3 3])

You now have two conditions to check for four-of-a-kind

You can think of similar patterns for the other two outcomes that might help you to determine which outcome you have achieved

***NOTE: While a five-of-a-kind could also count as a four-of-a-kind, a three-of-a-kind, or a pair, you should only count it as a five-of-a-kind 🡪 essentially, you should only count the highest outcome for a given roll***

1. Compute and display the probabilities (as a percentage) of rolling each of the 4 combinations on a die.

**Now, run your script for the following number of rolls and record the probabilities of achieving each combination:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of Rolls: 10** | | **Number of Rolls: 100** | | **Number of Rolls: 10000** | |
| **Combination** | **Probability** | **Combination** | **Probability** | **Combination** | **Probability** |
| Pair | 80% | Pair | 63% | Pair | 69.54% |
| 3-of-a-Kind | 0% | 3-of-a-Kind | 22% | 3-of-a-Kind | 18.76% |
| 4-of-a-Kind | 0% | 4-of-a-Kind | 1% | 4-of-a-Kind | 2.02% |
| 5-of-a-Kind | 0% | 5-of-a-Kind | 0% | 5-of-a-Kind | 0.09% |

**Paste your script below:**

%Models Lab 6 Problem 3

clear; clc;

rng('shuffle')

%start the counts at 0 and index them as the dice are rolled

count\_pair = 0;

count\_3kind = 0;

count\_4kind = 0;

count\_5kind = 0;

count\_rolls = input('How many times should the die be rolled? ');

%roll the dice and index the variables

for num\_rolls = 1:count\_rolls

dice\_roll = randi([1 6],[1 5]);

dice\_roll = sort(dice\_roll);

if dice\_roll(1) == dice\_roll(5)

count\_5kind = count\_5kind + 1;

elseif dice\_roll(1) == dice\_roll(4) || dice\_roll(2) == dice\_roll(5)

count\_4kind = count\_4kind + 1;

elseif dice\_roll(1) == dice\_roll(3) || dice\_roll(2) == dice\_roll(4) || dice\_roll(3) == dice\_roll(5)

count\_3kind = count\_3kind + 1;

elseif dice\_roll(1) == dice\_roll(2) || dice\_roll(2) == dice\_roll(3) || dice\_roll(3) == dice\_roll(4) || dice\_roll(4) == dice\_roll(5)

count\_pair = count\_pair + 1;

end

end

%find the percentages

percent\_pair = count\_pair/count\_rolls\*100;

percent\_3kind = count\_3kind/count\_rolls\*100;

percent\_4kind = count\_4kind/count\_rolls\*100;

percent\_5kind = count\_5kind/count\_rolls\*100;

%print out percentages

fprintf('Roll percentages - Pair: %0.2f, Three-of-a-kind: %0.2f, Four-of-a-kind: %0.2f, Five-of-a-kind: %0.2f\n',percent\_pair,percent\_3kind,percent\_4kind,percent\_5kind);

1. **To be turned in:**

* You will need to upload this word document with all tables, questions, and figures included and the m-file for your script.