1. Text

   Description automatically generated with low confidence
2. CoinResultsNoLoop=function(n)

{

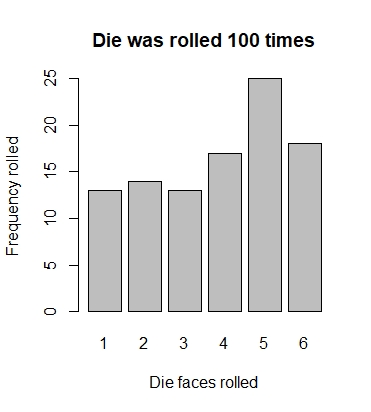
coinList<-sample(c(“Heads”, “Tails”), n, repl=T)

return(coinList)

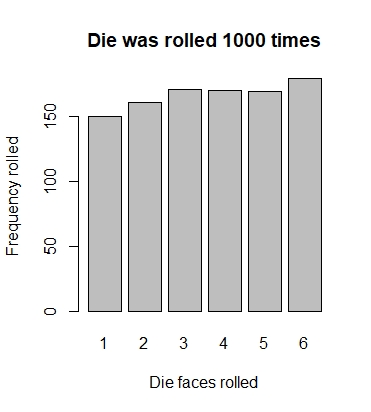
}



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n \ m | 10 | 100 | 1000 | 10000 |
| 10 | Max: 0.8  Min: 0.3 | Max: 0.59  Min: 0.44 | Max: 0.547  Min: 0.486 | Max: 0.5067  Min: 0.4930 |
| 100 | Max: 0.8  Min: 0.1 | Max: 0.66  Min: 0.38 | Max: 0.534  Min: 0.454 | Max: 0.5147  Min: 0.4860 |
| 1000 | Max: 0.9  Min: 0.0 | Max: 0.67  Min: 0.34 | Max: 0.559  Min: 0.445 | Max: 0.5183  Min: 0.4828 |
| 10000 | Max: 1  Min: 0 | Max: 0.69  Min: 0.31 | Max: 0.571  Min: 0.443 | Max: 0.5191  Min: 0.4822 |

1. 

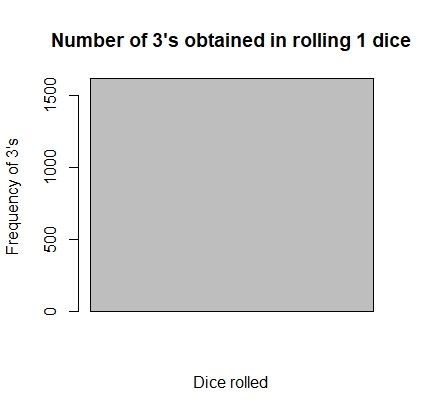
Fewest die rolls, means more skewed results here.



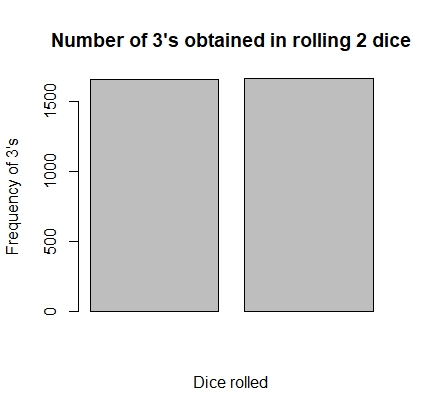
1000 die rolls are a lot more even.



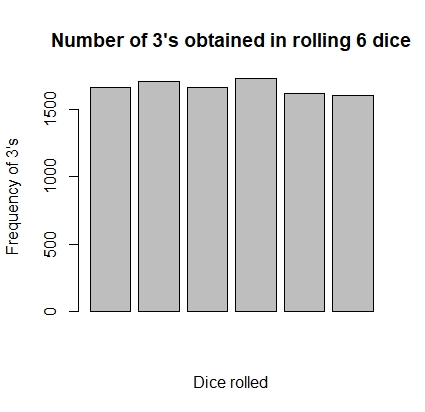
The most die rolls are almost all even. Shows how it is purely random. Two graphs prior show die face 1 with the least frequency, now it is narrowly the most. RANDOM!



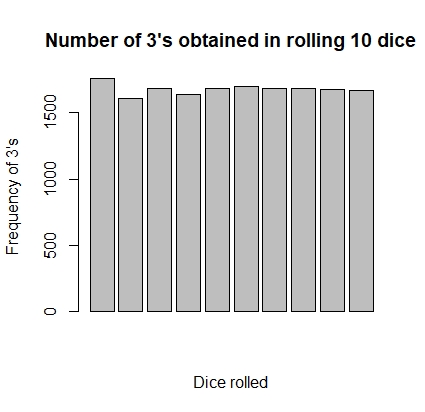
With 1 dice rolled, only 1 big block appears. the number of times this die got 3 looks to be over 1500 times. Which makes sense since 10000 rolls / 6 sides = 1666.67 sides/roll if they were allocated evenly.



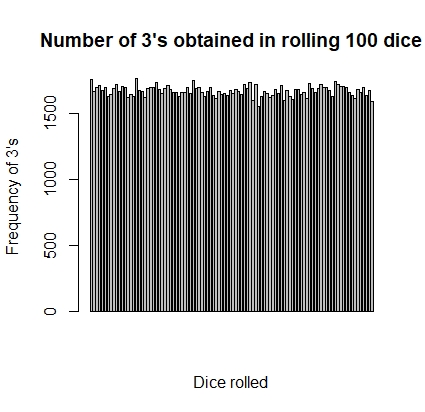
2 bars mean 2 dice were rolled 10000 times. The occurrences that each dice rolled a 3 is virtually the same from this graph. The biggest variation comparison could have been this graph but it is quite close just from luck.



This graph shows a bit more variation for each dice but still very close. I expect all the 3 occurrences to be consistent, especially as the number of dice go up.



The first bars vary a bit but eventually flattens on the latter half. It is to be expected that due to the random nature of rolling dice, that there will be times where the data is more varied. But there are also times where it remains very consistent; not to mention the number of rolls we are dealing with too!



This one is cool. I didn’t how the bar plot would handle so much dice. And again, consistent the whole way through. Some deviation but its not much. Some of the randomness gives highs and lows but they are still in a close range.

1. See Q7

|  |  |  |
| --- | --- | --- |
| m | With replacement | Without replacement |
| 1 | Shape  Description automatically generated with medium confidence |  |
| 5 | A picture containing diagram  Description automatically generated |  |
| 10 | Chart  Description automatically generated |  |
| 30 | A picture containing chart  Description automatically generated | A picture containing timeline  Description automatically generated |
| 50 | A picture containing chart  Description automatically generated | A picture containing text  Description automatically generated |

m = 1: It shows up as a black rectangle. I tested this with a smaller n value, and it looks like it does work normally. Maybe the density between 0 or 1 red card frequency more easily covers the white on the graph. Anyways, the graphs seem the same.

m = 5: These graphs still look largely similar. No real difference yet due to the little number of times the cards are drawn.

m = 10: Beginning to look more different. values without replacement looks scarcer on higher frequencies or red cards.

m = 30: Without replacement graph is having higher frequencies of a red card. With replacement graph is noticeable less frequent rate of red cards drawn

m = 50: Without replacement looks almost filled. With replacement has some more whitespace.

As m increases, the with replacement graph looks to get more consistently towards around half frequency. Without replacement graph, there is no replacements for a sample being taken; therefore, the deck is running out of cards. That means it is pulling all the red cards from the deck and most of the attempts draw 25 or 26 (max) of the red cards. This does not happen on the other graph since that deck does not run out of cards, the cards taken get replaced with the exact same card, so it tends to follow its pattern consistently with more and more attempts.

Note: with n = 10000 it seems to lean over the half threshold for some reason (unless the density covers up some whitespace). I tested with small n values, and it looks like it tends to stick to the middle more which does not make very much sense to me.