Math 321 – Spring 2019 – R Activity 3

SUBMISSION INSTRUCTIONS:
- Submit this assignment via email. DUE DATE: Wednesday, 2/13
- Include all R commands and output that is used to answer any questions or produce any requested graphics.

This R activity will have you simulate some random experiments and comparing the outcomes with the theoretical probabilities.

Coin Flipping

First same the coin flip simulation code in your working directory, and set R to that working directory. Now load the included coin flip simulation code with the R command:

```
> source("coinflipsim.R")
```

Alternatively, you can open the coin flip simulation file into the editor in RStudio and use the keyboard shortcut `Ctrl+Alt+r` to run it and load the coin clip function into R. For this alternative method, it doesn't necessitate that your working directory is set ot the location of the file.

Now simulate a fair coin flip with the `flipcoin()` function:

```
> flipcoin()
```

Now simulate 100 fair coin flips:

```
> flipcoin(N=100)
```

Now let's simulate 100 fair coin flips for a biased coin that has a 75% change of heads and 25% chance of tails, by putting 75 H and 25 T tickets in the box:

```
> flipcoin(N=100, numH=75, numT=25)
```

Question 1: Decide on a number of heads and tails tickets to put in the box. Simulate a few different batches of coin flips, say from 10 flips up to at least 100,000 or more. Do you think the proportions of heads and tails from your simulations reflect the proportion of tickets you put in the box reasonably well? Why or why not? What happened as you increased the number of coinflips in your simulation?

Poker

Here is the Wikipedia page on 5-card poker probability. It which gives all the combinatorial formulas for counting how many ways each type of hand is possible, and their respective probabilities and frequencies:
https://en.wikipedia.org/wiki/Poker_probability#Frequency_of_5-card_poker_hands

You task is to use the provided poker simulation R program to draw a large number of 5-card poker hands and see how many you get of each type.

Question 2: Load the poker simulation code into R. Use the function `drawManyHands(N=10000)` to simulate 10,000 poker hands. Include the output from you simulation. Do the frequencies match what you should expect?

Die rolling

Now simulate rolling a fair 6-sided die roll using the `sample(x, size=n, replace=FALSE)` R function. The first argument `x` is the list that we will randomly choose from, the second argument `size=n` is the sample size, and the third argument tells us whether to draw with `replace=TRUE` or without `replace=FALSE` replacement.

A single die roll:

```
> sample(1:6, size=1)
```

Roll the die 10 times:

```
> sample(1:6, size=10, replace=TRUE)
```

Note that we absolutely need to do the above with replacement, otherwise, after 6 rolls, all the numbers will be used up!

Now we are going to simulate rolling two dice and look a their sum.

```
> Nsims=40
  die1 = sample(1:6, size=Nsims, replace=TRUE)
  die2 = sample(1:6, size=Nsims, replace=TRUE)
  X = die1 + die2
```

Now we have our individual die rolls stored and their pair-wise sum. Check to make sure everything is summing correctly. Let's put the die rolls and their sum into a single array. The `cbind()` R function takes lists (of the same length) and puts them together in a matrix/table as columns ("column bind").

```
> diedata = cbind(die1, die2, X)
```

Rcall that when you save something to a variable name, like we jsut did the "`diedata`," you can retype that name and hit `Enter` to see theNow check that the first column plus the second column equals the third column:

```
> diedata[,1]+diedata[,2]==diedata[,3]
```

You should see a list of `TRUE`'s. If there are any `FALSE`'s, then something went wrong. Rather than looking through the entire, list, we can sum them up, `TRUE=1` and `FALSE=0` for math operations:

```
> sum(diedata[,1]+diedata[,2]==diedata[,3])
```

The result should be the total number of die rolls.

Now we will tabulate the die roll sum data results into a frequency count.

```
> table(X)
```

Now plot as a histogram using the `hist()` function. Set the bins to $\{[1.5, 2.5], \ldots, [11.5, 12.5]\}$ since we know the sum can only take on the values from 2 to 12.

```
> hX = hist(X,breaks=seq(1.5,12.5,by=1))
```

Now let's actually analyze the true sample space.

```
> S = expand.grid(die1=1:6, die2=1:6)
```

Now let us tabulate the sum of the dice from the sample space. This can be accomplished with either of the two following commands:

```
> table(S$die1+S$die2)   (or)
> table(rowSums(S))
```

Now let's turn our tabulation into probabilities:
```
> trueprob = table(rowSums(S))/nrow(S)
```

Now let's turn our simulation data into probabilities using the frequencies form our histogram:
```
> simprob = hX$counts/Nsims
```

Now let's plot them side-by-side in a barplot. This is actually two frequency histograms side-by-side. Since our tabulated results from the true sampls space and our hsitogram counts from our simulation data rae stored as row lists, we will use a `rbind()` function (row bind).
```
> barplot(rbind(trueprob,simprob),beside=TRUE)
```

Question 3: Now simulate different sample sizes of dice rolls and plot the side-by-side barplots/histograms. Do this for one hundred, one thousand, ten thousand, and one hundred thousand dice rolls. Here is a complete script that will do everything you need. Just edit the sample size (number of simulations).

```
> Nsims=10
  die1 = sample(1:6, size=Nsims, replace=TRUE)
  die2 = sample(1:6, size=Nsims, replace=TRUE)
  X = die1 + die2
  hX = hist(X,breaks=seq(1.5,12.5,by=1))
  S = expand.grid(die1=1:6, die2=1:6)
  trueprob = table(rowSums(S))/nrow(S)
  simprob = hX$counts/Nsims
  barplot(rbind(trueprob,simprob),beside=TRUE)
```

Describe what you observe as the sample size is increased.