Frank D’Ambrosio

Stat 421 Simulation

Mess with the Dealer Simulation

Introduction:

The game “mess with the dealer” is a game that is played by several players guessing cards. The dealer will look at the top card of the deck and the active player will attempt to guess that card, if they guess correctly the dealer will take a drink and put the card face up on the table. If the active player guesses incorrectly the dealer will tell them and they will get one more try, and if they guess incorrectly again, they will take a drink and the dealer will put the card face up on the table. Once three consecutive players guess incorrectly and must take a drink, the dealer will pass the deck to the next person, and they become the dealer. This continues until the deck is depleted. Each time the 4th value of a card is flipped face up on the table every player must take a drink. In this simulation I would like to look at the distribution of drinks taken by all the players, the probability of players being the dealer at the end of the game, and the players blood alcohol content (BAC) at the end of the game based on their sex (only had formula for male and female), and weight.

Methods:

In this simulation I used functions with nested “while” loops and “if”/”else” statements to simulate the different possible outcomes following the rules of the game. The game simulation calculates the number of drinks each players takes and from there I calculated the BAC of each player using the following formula:

*BAC = [(Alcohol consumed in grams)/ (body weight in grams * r)]*100*

$$\text{Where } r = \begin{cases} .55 \; for\; females \\ .68 \; for\; males \end{cases}$$

I then made a function to calculate body weight in grams using the formula:

*Body weight in grams = (body weight in pounds) * 454*

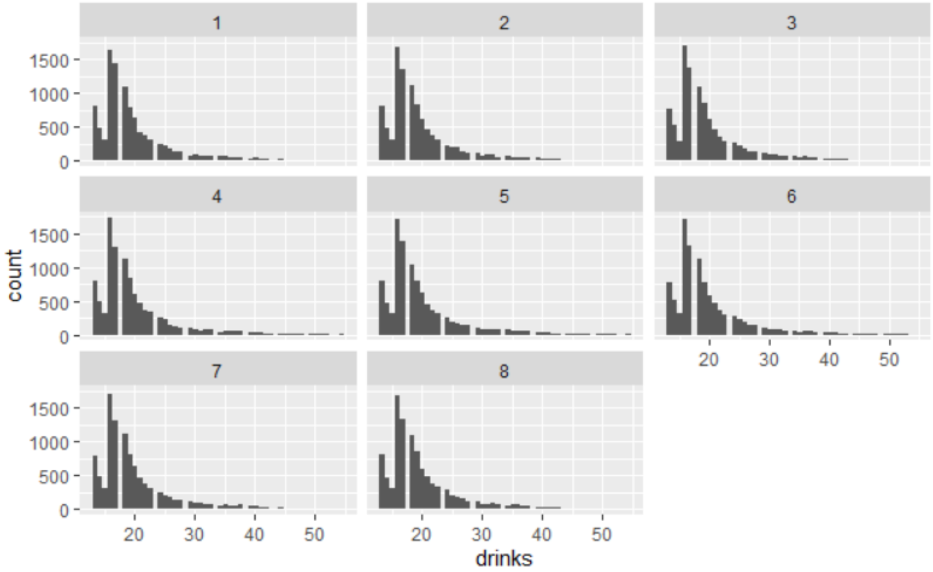
One drink in the United States is considered 14 grams of alcohol. This equates to 1 shot of gin or whiskey, 1 beer, 5 oz of wine, etc. The game itself calculates how many times a player takes a drink, not finishes this much alcohol, so I used the assumption that 1 beer takes 10 “sips” or drinks to finish, so I converted the games count for drinks into the amount of alcohol drunken in grams:

*Grams of alcohol = (# of drinks taken)*1.4*

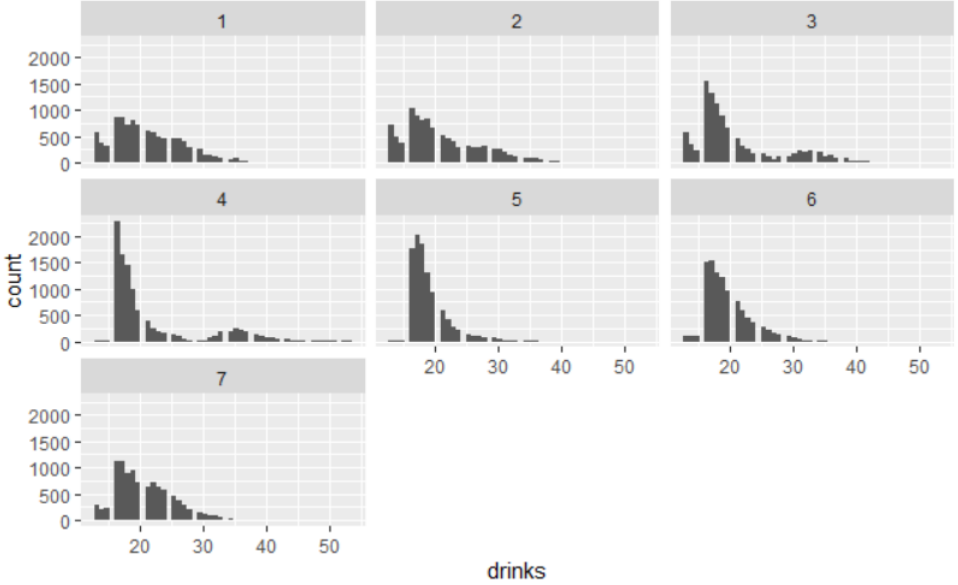
I then combined all of these into one function with the inputs need being drinks taken, weight in pounds of a player, and the sex of the player. I then mad a function that generates a random list of weight, a random list of sexes, then plays the game once, then replicates the game and prints out the BAC results for both the 1 game and for the average across all of the simulated games.

Results:

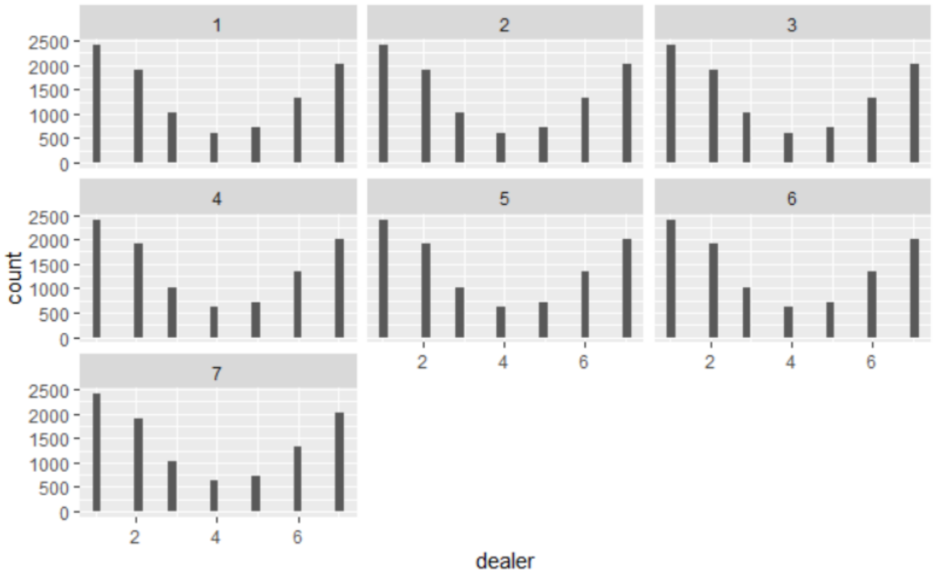
Without the starting dealer being specified and chosen randomly instead, the results of simulating the game many times are as one would expect: The distributions for the drinks each player takes is almost exactly the same.



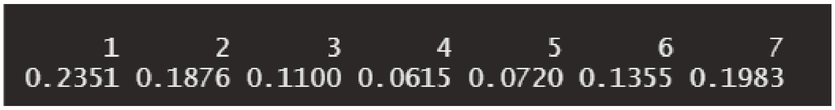
Once we select a starting dealer and simulate the game many times we see the distributions begin to vary. Below is the distribution of 10,000 simulations of the game having 7 players with the starting dealer being the 5th player.



Below is the distribution of times a player ended the game as the dealer with the same metrics as above.

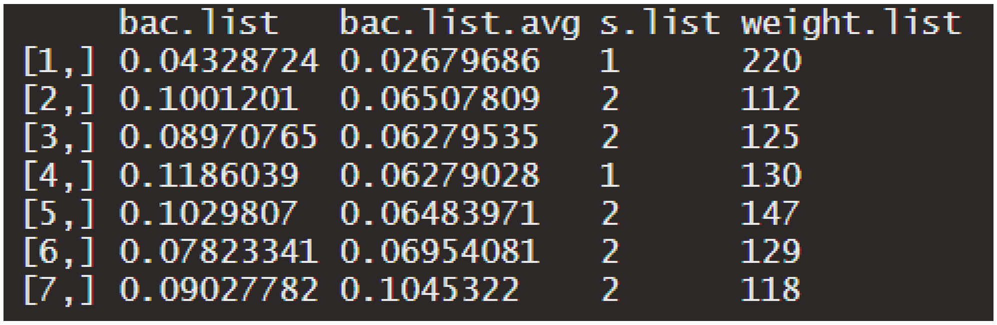


The probability for each player finishing the game as the dealer are as follows.



```
     1      2      3      4      5      6      7 
0.2351 0.1876 0.1100 0.0615 0.0720 0.1355 0.1983 
```

I was surprised at how high the BAC levels are at the end of the game but having played it myself it makes sense. Below is results of simulating the game 10,000 times with 7 players and the 5th player being the starting dealer. The far left column is 1 game, the second column is the average of the 10,000 games, followed by a column defining the sex and lastly a column displaying the weight of each player.



```
     bac.list   bac.list.avg s.list weight.list
[1,] 0.04328724 0.02679686   1      220        
[2,] 0.1001201  0.06507809   2      112        
[3,] 0.08970765 0.06279535   2      125        
[4,] 0.1186039  0.06279028   1      130        
[5,] 0.1029807  0.06483971   2      147        
[6,] 0.07823341 0.06954081   2      129        
[7,] 0.09027782 0.1045322    2      118        
```

Conclusions/Future Work:

I wrote a function to play a game of “mess with the dealer” and then simulated it to look at distributions and probabilities of results along with the BAC levels of the players. Everything for the game is modifiable – I can run this for however many players are playing and can define any starting dealer or can make it completely random. The function at the end generates random players' sex and weight for the sake of running but this can all be entered manually to get game/player specific results. For future work, when a player initially guess incorrectly the dealer must tell the player if the card is higher or lower than their guess. This makes guessing a bit easier and would skew the results towards the ending dealer having more drinks. In my code, the dealer does not indicate this thus the second guess is a bit harder than reality (though in my code I do end the game at 3 cards since when indicating higher or lower, the player cannot be incorrect at this point). Another variation of the game makes it such that if the player guesses wrong twice, the amount they drink is the difference between their guess and the actual card; this would be interesting because while the dealer would get stuck drinking more times, players may have to drink more per event.

Appendix:

Line 8 – packages

Line 24 – create a deck

Line 74 – choose starting/active player

Line 85 – move dealer to next person

Line 103 – make sure dealer cant be the active player

Line 115 – design a turn of the game

Line 157 – put the above together

Line 288 – replicate and plot

Line 318 – designate a starting player instead of random

Line 475 – calculate probability of being dealer at the end of the game

Line 599 – turn pounds to kg

Line 612 – calculate average drinks per game

Line 626 – calculate bmi

Line 643 – calculate bac

Line 812 – allow inputs of weights to the game

Line 848 – final plots