Figuring out the Game 'Hangman' *by Misha Jhaveri*

Hangman has always been a fascinating game for me because of the aspect of probability attached to a word game. If a word is x letters long, then the range of possible permutations and combinations can be analyzed in a fashion similar to analyzing an x letter long string of number, with 26 existing letters. However, the interesting twist is that there are only a certain set of "allowed" combinations since the string of letters need to be a dictionary certified word. My research revolved around trying to figure out if there was any way to **"beat" the game of hangman.** This essentially means correctly guessing the word before the number of wrong guesses allowed runs out.

My main methodology was through computer algorithms on the statistical program R. The R package enabled me to make user-defined functions to work through the game step by step.

Initially I wanted to try to create a program that "cheated" and changed the word every time the played guessed a correct letter, and keep going until there was only one possibility left. But I slowly realized that with the limitations of R it was more feasible to try to be the played and beat the game.

My first step was to prepare a database with all possible word. I then created a function to take one number (the number of letters) as the input and return a string of all words that were that long. This restricted the database to the number of possibilites for that turn. The next step I accomplished was to establish the preliminary "guess" or the most common letter in words of that particular length.

After figuring out and making a guess the next step was the last function to establish and the rest was to be applied recursively. There were two possibilities, either the guess was correct or it was wrong. If the guess was wrong, then the function removed all words that contained that letter and then another guess was made with the previous function but on the further restricted database. If the initial guess was correct, the function had to restrict the database to only thos words with the guessed letter in that "position". I hit a wall at this spot and making a function to accomplish this was very difficult due to the many possibilities. However my function works not but hits glitches some times when the words has the letter three times or more. But this case is rare and hence it took a lot of reruns before the glitch was discovered.

For the most part running and rerunning both sets of functions has resulted in a success rate of over ninety percent. I would love to continue working on this over summer to be able to figure out full proof functions and to potentially venture into the "cheat" version of the game.