PHSX815 - Project 3

Ryan Scott

April 12, 2021

1 Introduction

This project is meant to study how well we can determine a value G which modifies a categorical distribution of data in a card game using likelihoods. The game utilizes a deck of N_{cards} cards, distributed among two players, played against each other. The higher valued card designates the winner. We utilize some weighted and unweighted Bernoulli distributions (the card values) which affect a categorical distribution of number of wins per game (N_{wins}). For this experiment, we want to look at how modifying the number of cards, (N_{cards}), the number of sets (or experiments, N_{sets}), and the number of games in each set (N_{games}) affects how well we can determine our G value. This G value is a variable named gimme which allows one of the players to use cards that are guaranteed to win. Changing the G value modifies the data in unique ways, which will be explained in a later section.

The goal: Study how well we can determine G by varying N_{games}, N_{cards}, and N_{sets} using the likelihood.

We first describe how the game works in **Game Mechanics**, then we take a look at the algorithms and calculations used in the study in **Algorithms and Calculations**. Next, the analysis begins when we discuss how our variables affect G in **Studying the Distribution**. Finally, we conclude with a summary of results in **Conclusion**. Figures are located at the end of the document.

2 Game Mechanics

The standard game has $N_{cards} = 16$. The 16 cards are randomly dealt to two players, A and B. The two players then simultaneously show the top card of the deck and the player with the high value card records a win. After all cards are played ($n = {}^{N_{cards}}/_{2}$ rounds). The wins associated with player A are recorded for analysis. This constitutes one game. Multiple *games* make up one *set*.

The game is considered "fair" when $G = 0$: the cards are randomly distributed and the opportunity for winning is normally distributed around 50%. We can modify this value, allowing player A to use unfair cards. If a normal deck of 16 cards are valued 1-16, the cards for the gimme deck are valued at 17+, guaranteed to win. When $G > 0$, the player randomly loses a number of cards equal to G (in order to keep the number of cards in their own hand constant), guaranteeing that the cheater cards are played.

3 Algorithms and Calculations

We use two python scripts to accumulate and analyze data. The first is CardgameSim.py. As its name suggests, it is used to simulate the game and iterate over a number of games and sets from a given input. The user also has the ability to set the number of cards, gimme value, and name of the output file. When run, this script will shuffle all of the cards, randomly distribute them between two players, and return a value from 0 to n. This is determined quasi-randomly based on how the cards were dealt and on the G value. If it returns 0, player A did not win any games; if 8, player A won 8 games, etc. When G is nonzero, player A receives G cards that are guaranteed to win. This suggests the returned win value for cheated games must be G or greater. The domain of our win distribution shifts from [0,8] to [2,8], for example, if $N_{cards} = 16$ and $G = 2$.

The probability of winning any given hand in a game is given by $p_1 = \frac{2G}{N_{cards}} + .5(1 - \frac{2G}{N_{cards}})$. The first expression gives the number of hands that will win based on G, the second expression gives a standard $p = .5$ probability that the rest of the hands will be won. This value also determines the probability distribution of winning X number of games for any given G value. The probability distribution is determined by the equation: $\binom{n-G}{i-G}(.5)^{i-G}(.5)^{n-i}$, where n is the number of hands in any game, i is the indexed value for winning i games, and G is the variable we are attempting to determine, which ranges from [0,n].

The number of hands won per game is listed in the user designated text file as a single value. We then separate each game with a comma (,) and each set is a new line of data. The text file is then composed of N_{games} rows by N_{sets} columns.

Additionally, this script will also save the user designated variables in a text file as "rules_userinput.txt". That file contains four lines of data that indicate the values for N_{cards}, N_{games}, N_{sets}, and G, respectively. This is recorded for the analysis script: CardgameAnalysis.py.

We use CardgameAnalysis to read the data from two text files, organize it, and return plots of interest. This script first gets all of the text file data (iterated games and the associated rules) and converts it into data that we can manipulate and study. Our experiment focuses on studying the probability distribution of G from the likelihoods. Our script includes relevant code to determine the "pull" of the data, which is used to give confidence intervals of our distribution. The pull is discussed later.

The plots that we will investigate are determined by the likelihood of the data. For each set of data, we have data points equal to N_{games}. The number of wins (N_{wins}) for each game reveals information about which G value was used, because different G values means different probability distributions. We use the likelihood, which associates each value of N_{wins} with a probability for any possible G value. If we take the product of all of these values, we obtain our likelihood for that particular set. This means that each set will increase the number of possible likelihood values for our distribution. Once we have a distribution of likelihoods for any one set, we can determine which G value maximizes the likelihood. We record that value for our distribution.

A quick note: we use the likelihoods because they are an important part of developing the uniformly most powerful test statistic, as indicated by the Neyman Pearson Lemma for testing simple (point) hypotheses.

Once we have our likelihoods, we can determine our 1-σ range by looking at where the distribution of values exceeds .5 after normalized with $log(L(G)/L()\hat{G}))$ where $\hat{G}$ is the G value which maximizes the likelihood. There are two values of G on either side of $\hat{G}$ which will cause this function to exceed .5, which gives our 1-σ band. This value of σ is then fed into the pull function, $(pull = G - \hat{G})/\sigma$. This gives a confidence interval to our distribution.

4 Studying the Distribution

Instead of looking at each variable and determining how that affects the distribution of G, we are going to look at a number of arbitrary plots and make our conclusions. Plots are included at the end of the document.

We see that N_{sets} determines the number of data points we use. Larger N_{sets} means more data points in our distribution. N_{games} determines how accurate our measurements are; when there are more games, there is more data to determine a maximum likelihood. Conversely, N_{cards} will widen the distribution. When we increase the number of cards, we lose clarity on which G value we are using.

Lastly, we see that varying the G value does not seem to reveal any direct increase in clarity until G approaches very close n. When that occurs, the number of possible outcomes has become very small $(n-G)$ until there is only one outcome at $G = n$.

5 Conclusion

In summary, we have seen that we can use the likelihood function to determine which G value was chosen. More often than not, this method reveals an accurate value of G in experiments. The number of sets, games, and cards all affect how well G can be predicted. Finally, we can see how accurate our predictions are by observation of the pull plot associated with each distribution.

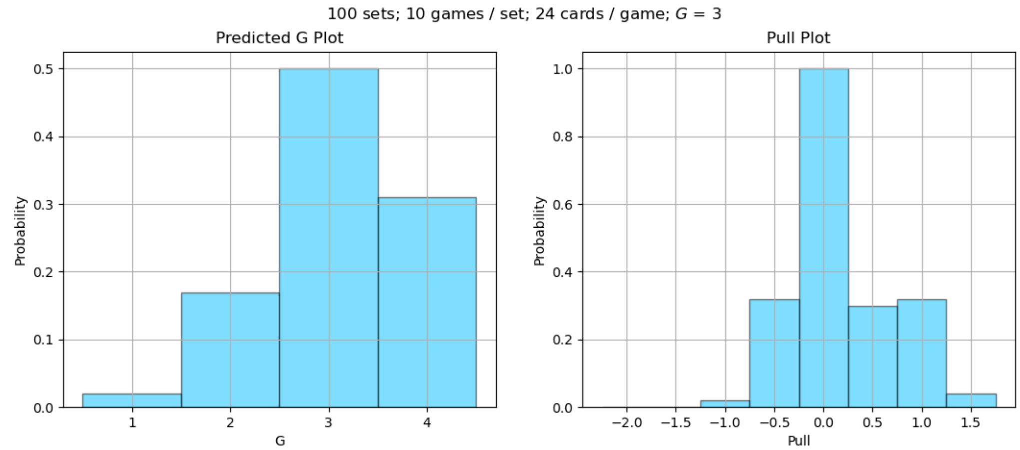

Figure 1: Control Game

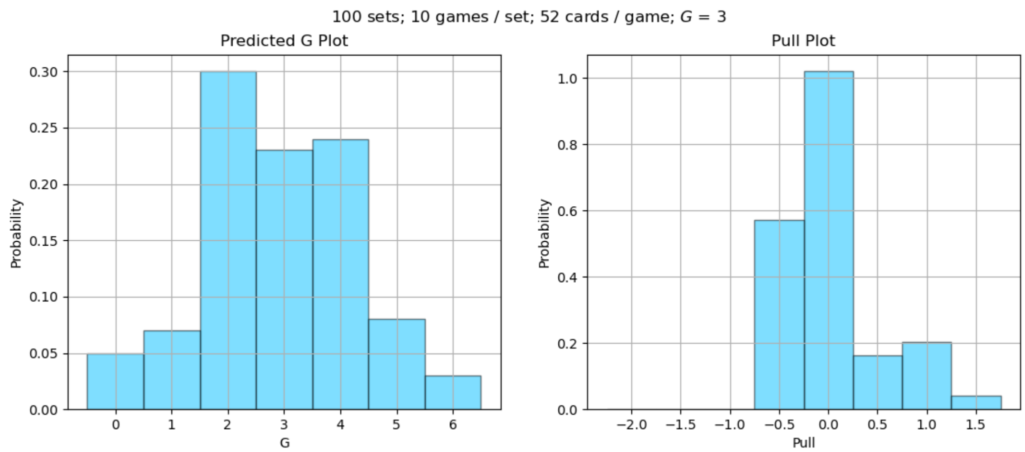

Figure 2: $N_{cards} = 52$

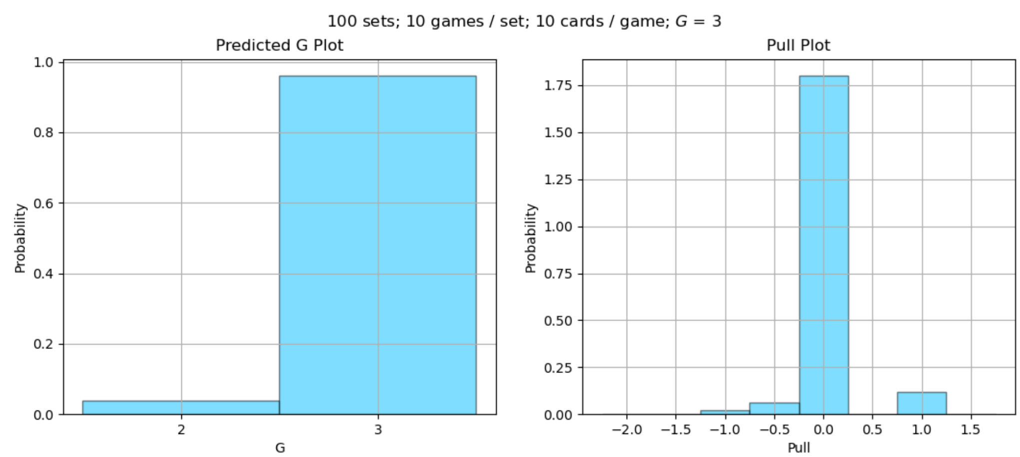

Figure 3: $N_{cards} = 10$

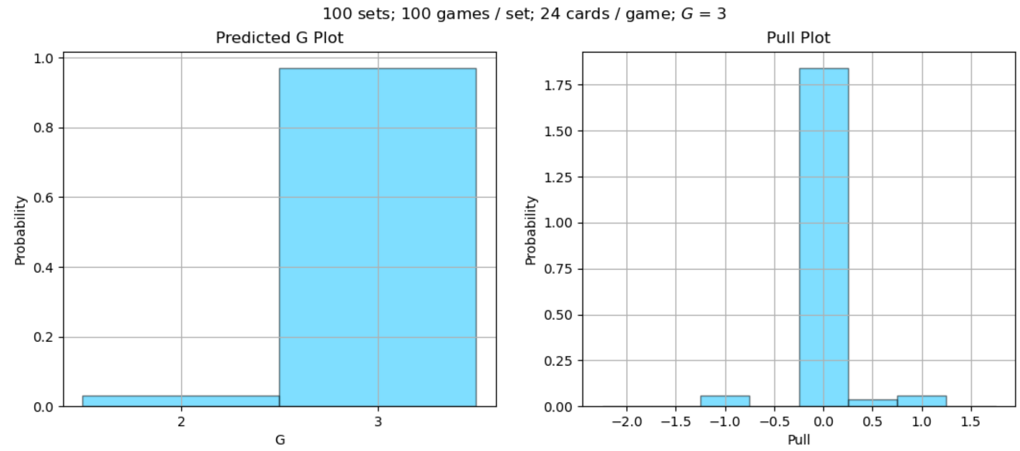

Figure 4: $N_{games} = 100$

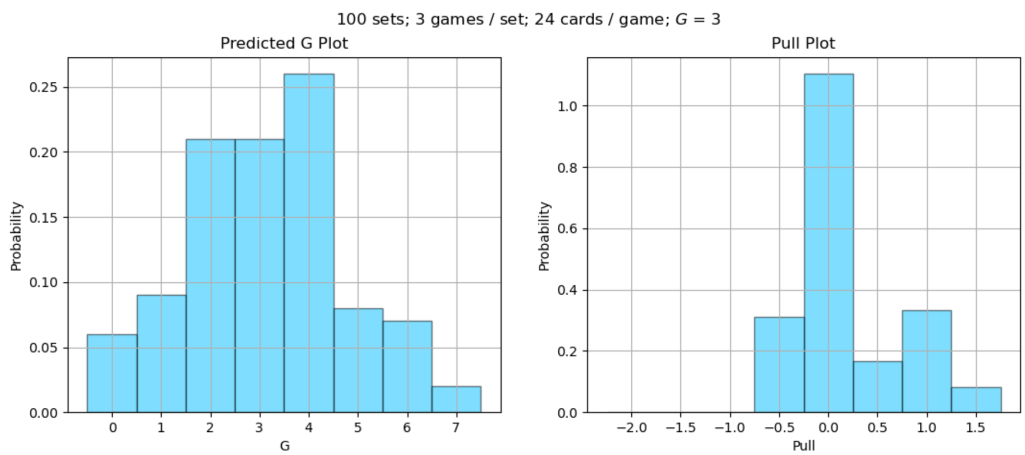

Figure 5: $N_{games} = 3$

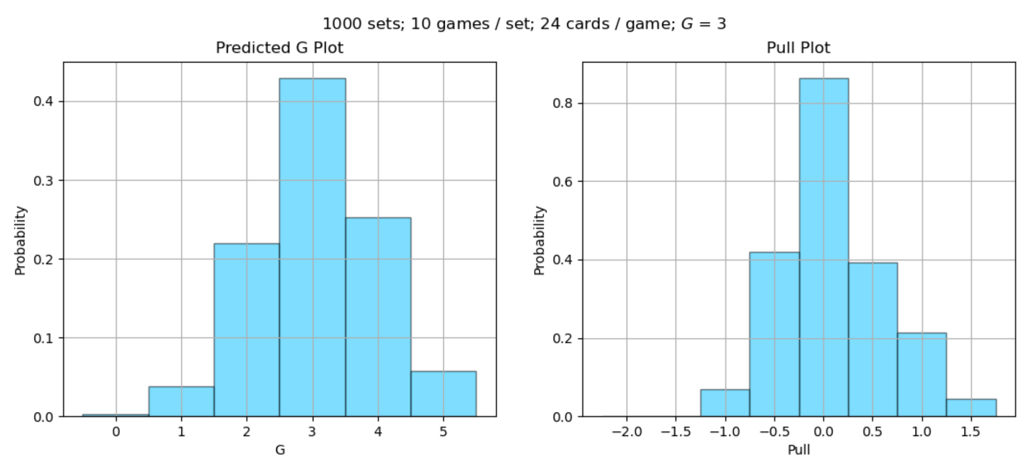

Figure 6: $N_{sets} = 1000$

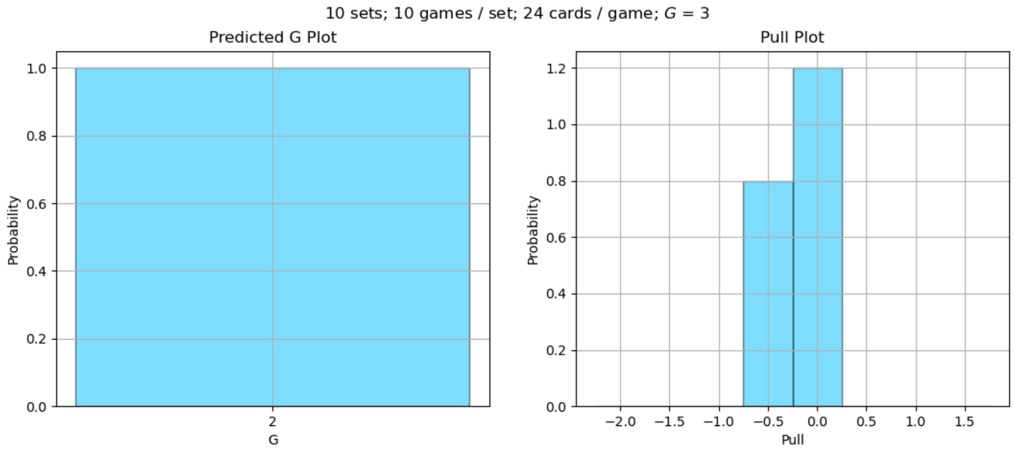

Figure 7: $N_{sets} = 10$

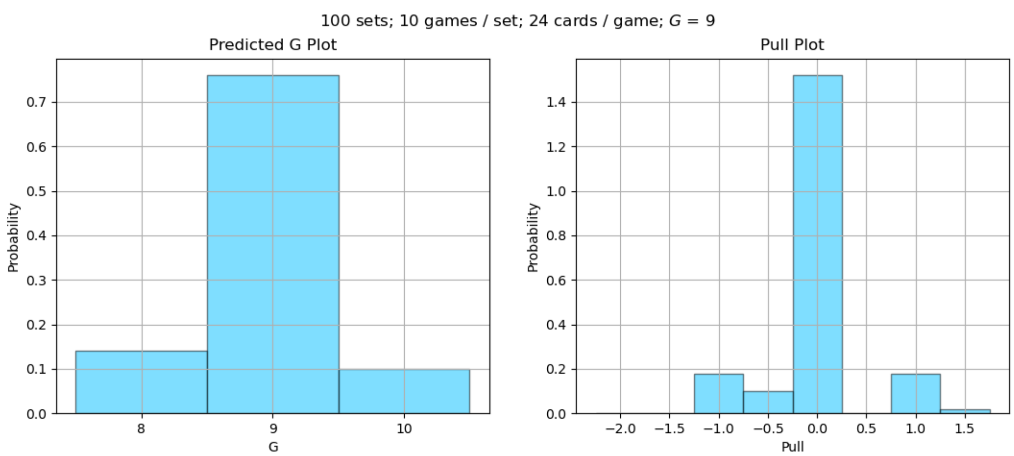

Figure 8: $G = 9$