Set: The Game ® – A Simulation Approach

Kimberly Stanke
kmstanke@mtu.edu
Applied and Computational Mathematics
Michigan Technological University

Amanda Stenzelbarton
adstenze@mtu.edu
General Mathematics
Michigan Technological University

1. Introduction

1.1 History

Set: The Game, designed by Marsha Falco in 1974, evolved out of a coding system she used while trying to connect groups of traits in her job as a geneticist. Sixteen years later, in 1990, Set: The Game was published by Set Enterprises. Over the years, Set: The Game has earned 29 best game awards, including the 1991 MENSA Select Award and the 1995 *Deutscher Spiele Preis* (German Game Prize). [1]

1.2 Rules

The game of Set is played by one or more players. There is a deck of 81 cards that come with the game. Each card has a picture that holds four properties: number of items, color, shape, and fill. There could be one, two, or three items in the picture. The items could be red, purple, or green. As far as shape, they could be diamonds, ovals, or squiggles, and they could be solid-filled, striped, or empty. So as an example, one card could have three red striped ovals. (For visual representations, see Figures 1-3 on the next page.) Each card in the deck is different; the deck consists of every combination of the four properties (3 representations of 4 properties means there are $3^4 = 81$ cards).

To play the game, the deck is shuffled and 12 cards are laid face-up on the table. Then players try to find groups of three cards that go together to find a "set". A "set" is a collection of three cards such that when each property is compared between the three cards, it can be said that either all three share the same characteristic or all three are different. Another way of looking at it would be to say (as the game instructions state), "If 2 cards are the same and 1 card is different in any feature, then it is not a ***SET***." Examples of sets are displayed in section 1.2.1. If, at any time, players agree that there is not a set in the cards face-up on the table, then the dealer can lay out three more cards. If the total number of cards on the table is ever above 12 and a set is found, new cards are not laid out. The goal of the game is to be the player to find the most sets.

1.2.1 Examples of Sets

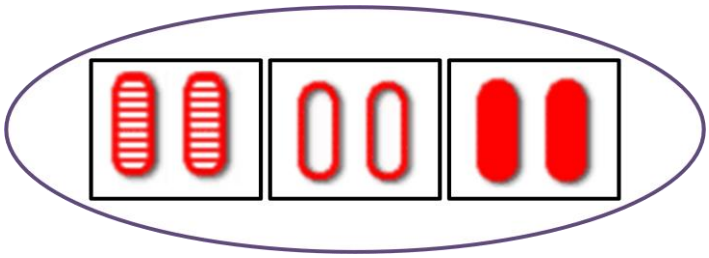


Figure 1: An example of a set – every card has two red ovals and each card has a different fill.

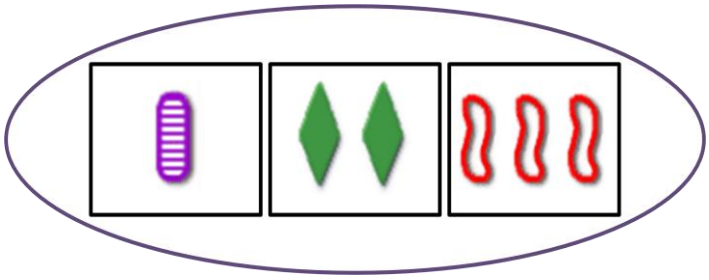


Figure 2: An example of a set - each card is a different color, contains a different fill, has a different number of objects, and is composed of different shapes.

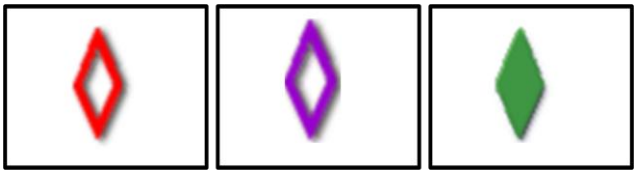


Figure 3: An example of a non-set - two cards have the same fill, but the third card has a different fill.

1.3 Goals

When discussing Set: The Game, some interesting questions arose before any work was done on the project: How many cards are left in the end of the game? Given a random order of cards, is there a "best" way to select sets based on the remaining cards so as to try to use all the cards? Are some sets easier to see than others? What kind of strategies are there to the game?

Throughout the project, other questions became relevant: If a card is lost, what happens? What if two cards are lost? Does it matter which set is chosen?

In preparation for pursuing the project, the internet was searched for strategies for Set: The Game. There was a lack of widely available strategy, so it's possible that simulation may be the only viable way to find answers to these goal questions.

Because of a lack of widely available strategy, simulation may be the only viable way to meet these goals.

2. Organization

2.1 Software

Wolfram Mathematica 8 software was used to simulate Set: The Game.

2.2 Encoding

2.2.1 Deck

Cards were represented as strings of four numbers: one number used to designate each property. Because every possible unique card is involved in Set: The Game, the deck was created by generating all possible strings of four numbers from one to three, as shown in Figure 4. To shuffle the deck, the built-in Mathematica function RandomSample to reorder the cards without changing the identities of the cards, as shown in Figure 5.

```
SetDeck = Table[
   {i, j, k, m},
   {i, 1, 3},
   {j, 1, 3},
   {k, 1, 3},
   {m, 1, 3}];
```

Figure 4: Code indicating how the deck was created.

```
SetDeck = RandomSample[SetDeck];
```

Figure 5: Code indicating how the deck was shuffled.

2.2.2 Standard Game Play

The IsSet function, shown in Figure 6, reduces modulo 3 to determine if any given three cards are a set. If the three cards a set, adding by index and reducing modulo 3 yields zero, the function returns true; if the three cards are not a set, the function does not yield zero and returns false.

```
IsSet[{c1_, c2_, c3_}] := Module[
  {},
  Max[Mod[c1 + c2 + c3, 3]] == 0
 ]
```

Figure 6: Code showing the IsSet function.

The FindSet function, shown in Figure 7, maps all possible arrangements of three each of the cards on the table and calls the IsSet function to determine which of the arrangements are sets. The function returns the number of sets on the table and a random set.

```
FindSet[CT_] := Module[
  {PossibleSets, CheckSet, SetFound, SetPosition},
  PossibleSets = Subsets[CT, {3}];
  CheckSet = Map[IsSet, PossibleSets];
  CheckSet = Position[CheckSet, True];
  If[ CheckSet == {}, Return[{0, {}}]];
  SetPosition = RandomChoice[CheckSet];
  SetFound = Flatten[PossibleSets[[SetPosition]], 1];
  {Length[CheckSet], SetFound}
 ]
```

Figure 7: Code showing the FindSet function.

2.2.3 Differences

2.2.3.1 Number of Cards

InitializeSetDeck (Figure 8) shows the coding that creates the set deck. The variables i, j, k, and m represent the four characteristics on each card, and each is encoded as a 1, 2, or 3 based on which representation of that property it is holding. The parameters n and d represent the number of cards face-up on the table (such as the 12 of a normal game) and the number of cards that have been "dropped" or lost respectively. It was encoded this way so as to easily be able to change the game play.

"81 card gameplay" involves playing with the standard 81 card deck. This was coded by setting n equal to 12 and d equal to 0. This way, the play is normal and all cards are present. In an 80 card game, d is set equal to 1 (because one card is lost), and similarly for 79 cards, d equals 2. This was done so a comparison could be made between the final number of cards on the table for the different game variants.

```
InitializeSetDeck[n_, d_] := Module[
  {SetDeck},
  SetDeck = Table[
    {i, j, k, m},
    {i, 1, 3},
    {j, 1, 3},
    {k, 1, 3},
    {m, 1, 3}];
  SetDeck = Drop[Flatten[SetDeck, 3], d];
  SetDeck = RandomSample[SetDeck];
  {SetDeck[[1 ;; (81 - d) - n]], SetDeck[[(81 - d) + 1 - n ;; (81 - d)]], {}}
 ]
```

Figure 8: Code showing how the drop cards are dropped for game variations.

2.2.3.2 Popularity

Another game variant that was looked at came from the updated goal question, "Does it matter which set is chosen?" The original code used FindSet (from Figure 7) to locate all the sets on the table at a given time. It then chose a random set from those, removed it and returned the set as SetFound. When looking at the popularity variant, FindSet was used to locate all the sets. However, for "popular game play," the code was set up so that the card that was in the most sets was labeled as the "popular card." (If there was more than one, they were all considered to be the popular card.) From there, one set was randomly selected from the sets that the popular card was involved in to be the one removed. There was also a "reject game play" in which instead of the card involved in the most sets being the one to base sets off of, the one involved in the fewest was the basis.

2.2.4 Repeated Game Play

The function SetPlayRepeat played Set: The Game for a length of 50 "turns". It was found that the maximum number of turns a game could take was 49, so it was safe to make the number of turns be 50. This 49 was calculated by counting the worst-case scenario of turns by hand. (It counted as a turn to put out 3 new cards if a set was not on the table.) This counting was counted by putting the number of cards on the table up to 21 and then let it go back down to 12. According to Davis and Maclagan, the maximum number of cards that can ever be face up without there being a set present is 20 (3). [2] So therefore, once the total gets up to 21, there must be a set present.

SetPlayRepeat was then altered so as to accommodate for the other game variants. PopularSetPlayRepeat, RejectSetPlayRepeat, MissingOneSetPlayRepeat, etc., each have their respective game rules encoded and play a full game with the specific guidelines in place.

Because the maximum number of cards that can be left at the end of a game is 18 in a normal game, 20 in a game with one card missing and 19 in a game missing two cards, the code makes it so any times there are 18 or more cards left at the end of the game, the remaining cards are written out to a text file so they can be reviewed later.

```
SetPlayRepeat[] := Module[
   {SetGame, CardsRemaining},
   {CD, CT, CF} = InitializeSetDeck[12, 0];
   SetGame = Do[{CD, CT, CF} = SetPlay[{CD, CT, CF}], {50}];
   {CDEnd, CTEnd, CFEnd} = Map[Length, {CD, CT, CF}];
   If[CTEnd == 18, PutAppend[CT, "18.txt"]; Print["Got One"]];
   {CDEnd, CTEnd, CFEnd}
  ]
```

Figure 9: Code showing the SetPlayRepeat function

3. Data

3.1 Overview

One million runs were made of each of the 9 types of game (all combinations of the set choice methods [standard, popular, reject] and deck size [81, 80, 79]). Each type of game was run 10,000 times, then was written out to a .csv (Comma Separated Value) file and posted online with an appropriate name (79PopularChoice_1 for instance) so as to be easily accessible. There were 100 of each file, giving a total of one million runs per game type. On average it took about one hour and forty minutes to run the nine game variants 10,000 times each, so to get the one million runs of each it took about one week.

3.2 Generation

To generate data, a do-loop was used to run each variation and export each file to the set directory. To import the data, StringForm was used to pull each set of 10,000 data points from the directory and the built-in functions Table and Flatten were used to create single strings of 1,000,000 million numbers.

```
Do[
 Data1 = Transpose[Table[SetPlayRepeat[], {10^4}]][[2]] / 3;
 FileName1 = ToString[StringForm["81RandomChoice_``.csv", n]];

 Data2 = Transpose[Table[PopularSetPlayRepeat[], {10^4}]][[2]] / 3;
 FileName2 = ToString[StringForm["81PopularChoice_``.csv", n]];
```

Figure 10: Code showing how the data is exported.

```
Data81Random = Table[
   Data81RaIn =
    ToString[
     StringForm[
      "http://www.mathlab.mtu.edu/~struther/Courses/2990/SET/April12/81RandomChoice_``.csv", n]];
   Import[Data81RaIn],
   {n, 1, 100}];
Data81Random = Flatten[Data81Random];
```

Figure 11: Code showing how the data is imported.

4. Results

4.1 81 Cards

Table 1: Cards Left in a standard 81 Card Game

Number of Cards Left	Percentage of Occurrence
0	0.12%
3	0%
6	46.9%
9	44.5%
12	0.73%
15	0.078%
18	0.0001%

```
SetHistogram81[Data81Random]
```

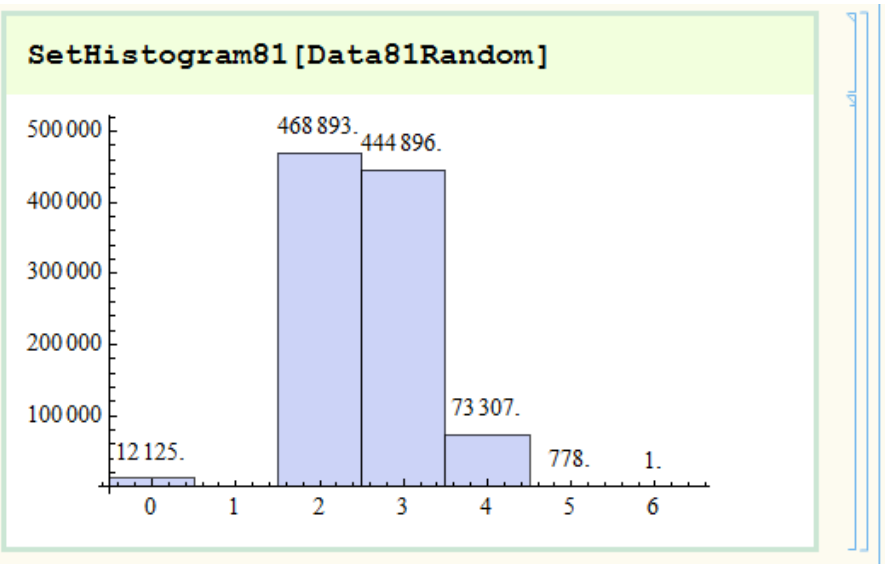


Figure 12: Histogram of the data representing a standard game of Set: The Game.

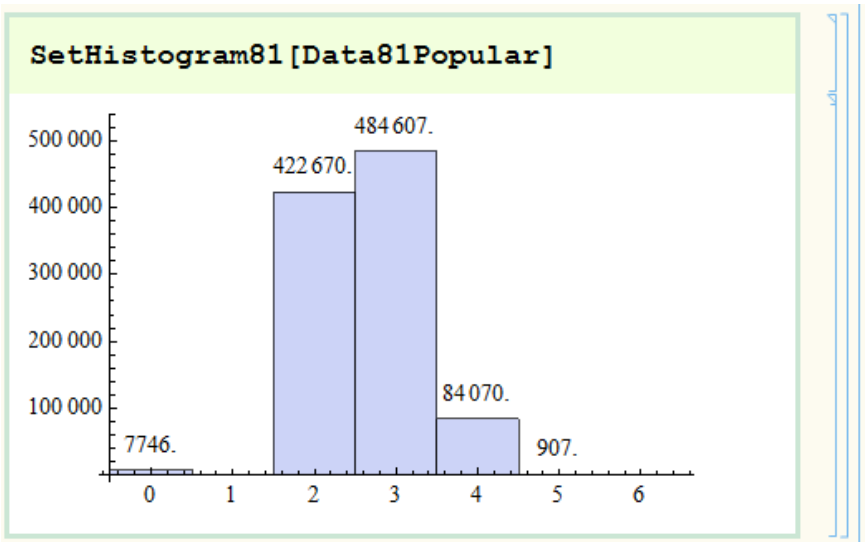


Figure 13: A histogram showing a game of Set: The Game with the most popular set being chosen.

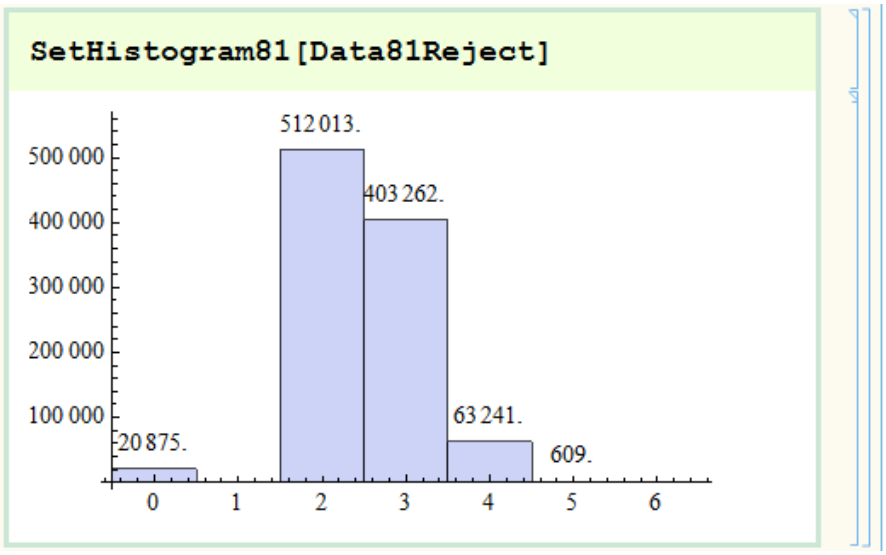


Figure 14: A histogram of the data representing a game of Set: The Game with the least popular set being chosen.

4.2 80 Cards

Table 2: Cards Left in a standard 80 Card Game

Number of Cards Left	Percentage of Occurrence
2	1.3%
5	25.2%
8	56.3%
11	16.6%
14	0.51%
17	0.0003%

```
SetHistogram80[Data80Random]
```

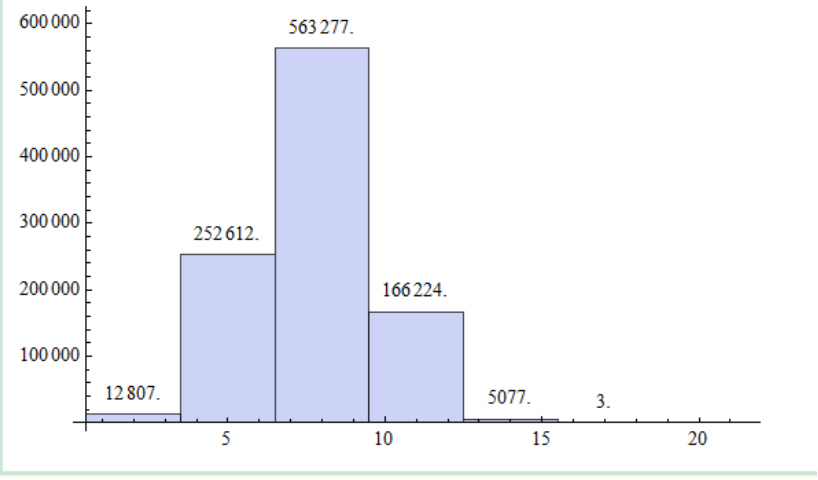
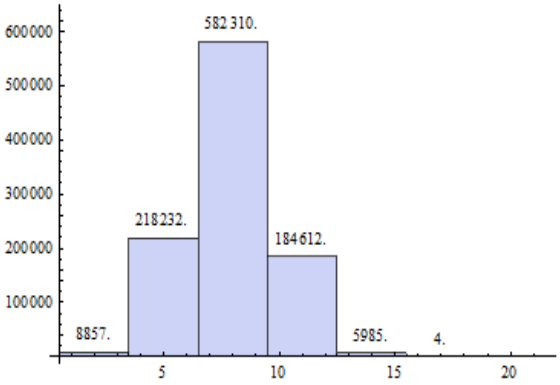


Figure 15: A histogram of the data representing a standard game of Set: The Game with one card missing.

```
SetHistogram80[Data80Popular]|
```




```
SetHistogram80[Data80Reject]
```

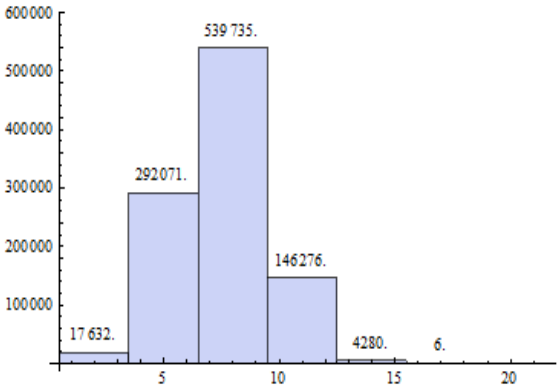


Figure 16: A histogram of the data representing a game of Set: The Game with the most popular set being chosen (above) with one card missing. A histogram of the data representing a game of Set: The Game with the least popular set being chosen (below) with one card missing.

4.3 79 Cards

Table 3: Cards Left in a standard 79 Card Game

Number of Cards Left	Percentage of Occurrence
1	0.20%
4	12.3%
7	54.1%
10	31.2%
13	2.2%
16	0.0074%

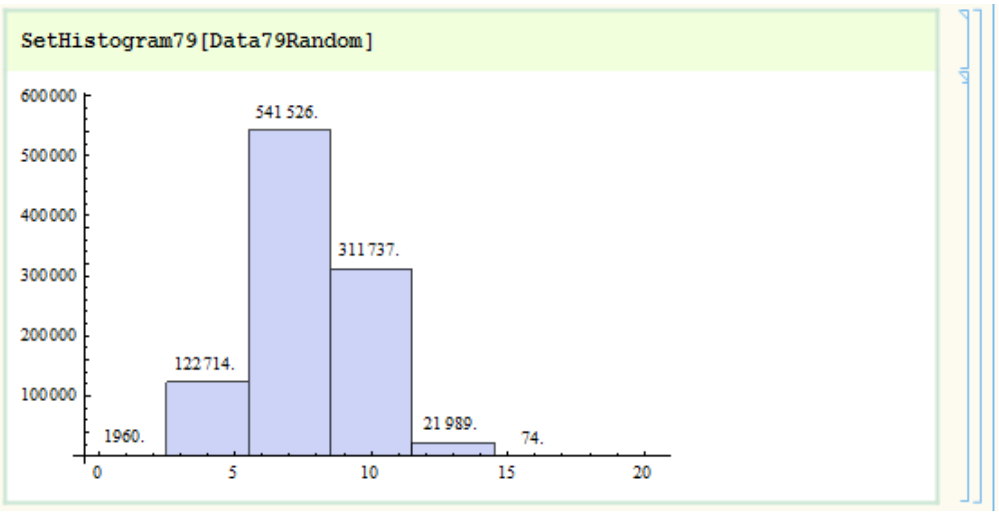


Figure 17: A histogram of the data representing a standard game of Set: The Game with two cards missing.

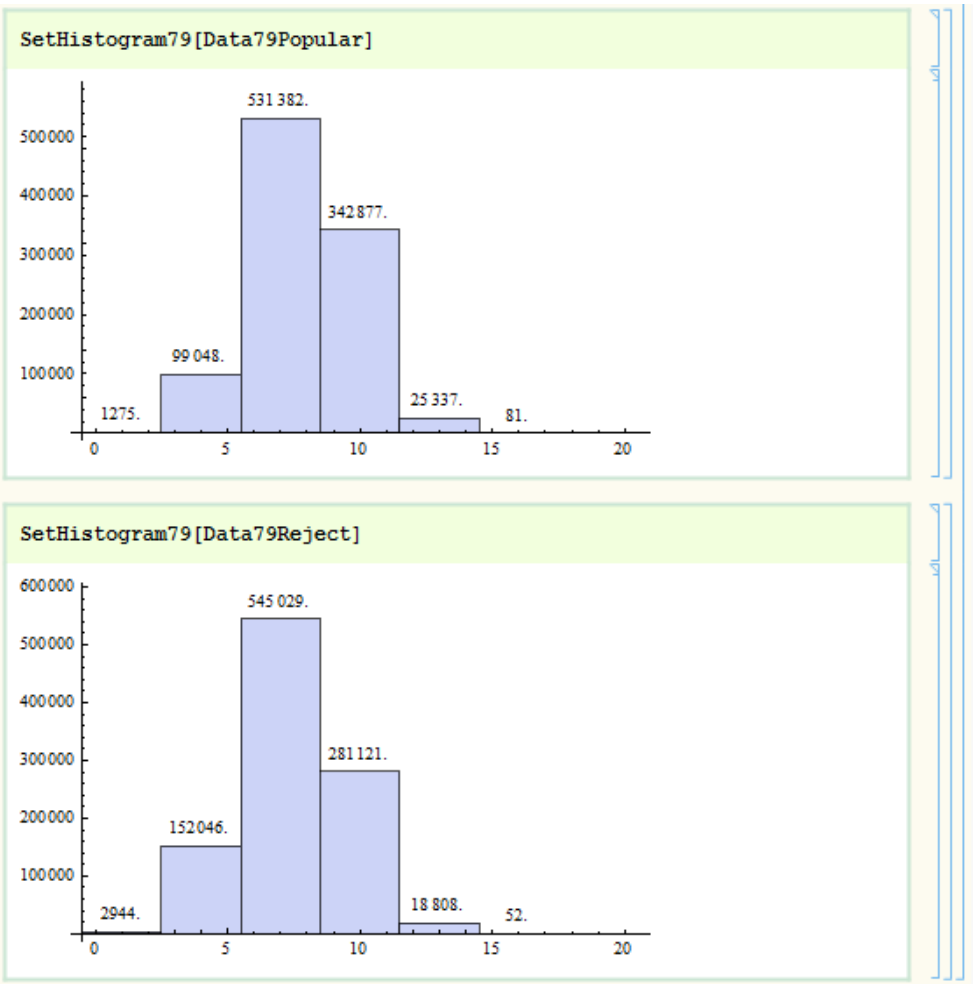


Figure 18: A histogram of the data representing a game of Set: The Game with the most popular set being chosen with two cards missing (above). A histogram of the data representing a game of Set: The Game with the least popular set being chosen with two cards missing (below).

4.4 Popular vs. Reject

To determine if the popular and reject were similar, as shown in Figure 19, the built in Mathematica function DistributionFitTest was used to test the distributions against each other. The results can be seen in Table 4.

Ho: Popular Data = Reject Data	
Ha: Popular Data ≠ Reject Data	
α=.01	
Cards	**P-Value**
81	4.88498×10^{-14}
80	3.37508×10^{-14}
79	3.41949×10^{-14}

```
DistributionFitTest[Data81Popular, Data81Reject]
```

Figure 19: Code showing the build-in Mathematica function used to determine if the distributions were similar.

5. Future Work

Some items that came up as possible points of focus for future work included the following:

- What if there were four different representations of each property? And if this were the case, would each set need to be four cards?
- What if three cards were lost? These cards could either be a set or not. Does it matter whether or not they are?
- How often are there 21 cards on the table at some point during the game?
- Are some sets easier to "see" than others?

6. Notes

Thank you to Dr. Allan Struthers, Michigan Technological University, for running and storing our data.

Data can be accessed at:
<http://www.mathlab.mtu.edu/~struther/Courses/2990/SET/April12/>.

7. References

[1] Falco, Marsha J. "SETÂ® Card Game." *SET® Card Game*. Web. 19 Apr. 2012. <http://www.setgame.com/set/index.html>.

[2] Davis, Benjamin L., and Diane Mclagan. "The Card Game Set." *Mathematics Department*. Rutgers School of Arts & Sciences. Web. 25 Apr. 2012. <http://www.math.rutgers.edu/~maclagan/papers/set.pdf>.