Extended Statistical Java Gaming Simulation

Timothy Liu, Mason Chen, and Joseph Jang
Lynbrook High School, Milpitas Christian School, and Monta Vista High School
San Jose, CA 95132, USA
timothys.new.email@gmail.com, mason.chen.training@gmail.com, and jangyeongshinn@gmail.com

Abstract

Our team has redesigned the 3 chips game into the 4 chips game. 4 chips game strategy is more complex than the previous 3 chips game. To do that extension, we fully utilized our knowledge of JAVA to create a program that could quickly and accurately represent each of our hypothesis. Three cases including the random pick, with 3 chips rules and with 4 chips rules are explored. We then compared the results of the program to our predictive model for further improvement. Our team still went through the standard Forming, Storming, Norming, and Performing phases, though the time spent on each phase was greatly reduced. This project is excellent practice for how to apply Java programming and statistics for the complex real-world problems.

Keywords

Java, Statistics, Probability, Predictive Modeling, Six Sigma

1.0 Introduction

The purpose of this project is to expand the model our team created last year. This project, unlike our last one, forces us to fully utilize our resources, like the computer, since it is impossible for us to calculate everything by hand. In the previous paper, we had hoped that the program we had created could be used in medical research, but that was not possible. However, since we could have more experience with the computer and JAVA this year, our program has become more advanced, and is able to incorporate more rules than before.

2.0 Design of the "4-chips" Game

For a quick review of the previous 3 chip game. The basic law is each 2 players play take turns to pick the one of more chips from the same color and who pick the last one is the winner. The sample space for the (6, 8, 10) 3 chips game is 480 and we can manually calculate the probability for each case. When we extend the game in to 4 chip game, for example, (6, 8, 10, 12) the sample space is 12X. The rules are more complex than 3 chips and the results are harder to predict. When we extend the game, the problem is more like real-world problem.

Therefore, we have redesigned the 3-chips game to the 4-chips game and here are the basic laws of the 4 chips game:

1. There are four groups of chips with different number and color in each group as the initial game condition (e.g. 12 Blue chips, 10 Red chips, 8 Yellow chips, 6 Green chips). The initial condition can be randomly assigned as long as there is NO identical number of chips in any two groups such as (X, X, Y, Z).
2. There are three types of players (Player Type A, Player Type B and Player Type C) who will play each other. One player will go first, and then two players will take turns until completed the game. Player Type A is not aware of any game rules, which means it's chip selection is completely random. Player Type B is aware of the 3 chips game rule, and Player C is aware of both 3 chip game rules and the additional 4 chip rules.

3. During each round, the player will decide one group (could be Blue, Red, Yellow, or Green) and remove at least one chip up to all of the remaining chips from that particular group.
4. The player that picks the overall last chip will be the loser of the game.

We already had four patterns in place for our 3-chips game which can be applied to 4 chips game directly when the 4th group is 0 number of chips:

- **Rule #1**: you will lose the game eventually if you will be the first player removed any group completely like (X, Y, 0) if the remaining two groups have different number ($X \neq Y$) of the remaining chips
- **Rule #2**: if your opponent removed any group completely such as (X, Y, 0), you will win the game if you can keep the remaining two groups with same number, i.e., (X, X, 0), except (1, 1, 0). You can keep this pair pattern until (1, 1, 0) to win the game eventually.
- **Rule #3**, if your opponent removed any group completely and the remaining two groups have one group with 1 chip and the other group with more than 1 chips (X, 1, 0), you will win the game if you can remove all the chips from that group to make (0, 1, 0). Your opponent will pick the last chip and lost the game.
- **Rule #4**, if your opponent will make two group have the same number of chips and the third group with different number like (X, X, Y) and $X \neq 1$, you can win by removing the entire Y group to make (X, X, 0) like Rule #2 concept.

The above Rule #1~4 is basically the same as our previous 3 chip game project by adding a dummy 0 number of chips to the 4th color in the 4 chips game.

However, after playing our 4-chips game, we discovered multiple pattern:
Despite being the simplest 4 chip rules, a much larger program is needed to properly compute them comparing to previous 3 chip game project. We needed to use the sorting function to consistently keep track of the numbers in an order. For example: 6, 2, 4, 5 → 2, 4, 5, 6. This helped us to develop the cases without having to worry about the number's positions

- **Rule #5**:
    - One way to win the game is to make the chips (0, 1, 1, 1), which forces the other player to pick the last chip. The possible combinations that can lead to this result is (0, 1, 1, y), and (1, 1, 1, y), so by making y into 1 and 0 respectively we can achieve 0, 1, 1, 1
    - Another way to win is to make the chips (0, 1, 2, 3), which backs the other player into a dead end. No matter what he or she picks, now, it will violate one of the winning patterns and result in a loss. For example, if he or she were to make the remaining chips (0,1,1,2), then you would be able to set up the winning pattern of (0,1,1,1), resulting in a win. The following are possible combinations that you wanna make to be (0, 1, 2, 3). ex:(1, x, 2, 3), (1, 2, x, 3), (1, 2, 3, x), (0, x, 2, 3), (0, 1, x, 3), (0, 1, 2, x), (0, 1, 3, x).
    - Another way for the player to win, is to keep a pattern when you get the chips (1, x, x+1, y) and change it to (1, x, x+1, 0), and keep this pattern until (0, 1, 2, 3) or (1, 1, 1, 0), both are cases that we previously explained that ensures the player's victory. For example, if someone were to get (1, 5, 6, 10), they can follow the pattern and make it (1, 5, 6, 0). Then when applicable, continue to follow the pattern to make it (1, x, x+1, 0). If the other player were to make the combination (1, 4, 6, 0), you can make the chips (1, 4, 5, 0), which continues to follow the pattern that we previously stated. In this example, if this were to continue, the player could end up with (1, 2, 5, 0), then he can proceed to make it (1, 2, 3, 0), which if played correctly, means the player wins. Another possible case is that the combination doesn't have the 1 to begin with, like (0, 2, 3, 7). In this case, the player can make the combination follow the previous pattern by changing 7 into 1. this means the chip will continue to follow the pattern of (1, x, x+1, 0), and the player will win. As long as the player changes the amount of chips in the game to match the pattern explained, if played correctly, the player will most likely win.

2.1 Two Main Hypotheses

Our team continued to use the same hypothesis from last year's project, with a slight change:

- Does going first in the game affect the winning probability as significant as it did in 3-Chip game?
- How does the rules including the 3 chips rules (Rule 1~4) and 4 chips rule (Rule 5) we designed affect the winning probability?

2.2 Project Research

Before team would build the predictive model, team has searched the Google Internet and has found some Poker Statistics but could not directly apply the Poker statistics associated with our 3 Chip-game. With the research already done before, we decided to build upon the previous 3 Chip game data and statistical research to redesign 4 Chips game.

2.3 Explore Basic Statistics and Probability

Team has decided to setup the initial condition as (6, 8, 10, and 12) to build the Statistical Prediction Model. We will simulate the winning % based on 3x1 scenarios in Table 1:

Table 1: 1x3 Matrix Move Rules

		Go Second		
		A	**B**	**C**
Go First	**A**	**I**	**II**	**III**

I. Player A vs Player A (Random)
II. Player A vs Player B (Rule # 1-4)
III. Player A vs Player C (Rule # 1-5)

Case I: Player A (Go First) vs. Player A (Go Second)

In Case I, we want to check whether the playing sequence will make any impact on the winning probability. If based on statistics, there should be no bias on winning the game by who will go first. The winning probability should be close to 50% among two players. We will use Java programming to verify this random probability at 50% winning probability.

Case II: Player A (Go First) vs. Player B (Go Second)

Initial condition is (6, 8, 10, 12), and player A will pick the first move. Player A can decide which group to pick and pick chips blindly by NOT following the game rules provided earlier.

Player A has 12 possible choices (after first move, the remaining chips can be from 0 chip to 11 chips). Among 12 choices, in contrast of the last game, where Player A can immediately lose, the 4 chip game model is more complex

and therefore can't lead to an instant loss choice at the start of the game. But if the player picks one of the whole groups, the next player, who will most likely know the rules might be able to apply a rule. But if a group is not immediately removed, it'll take a few turn before any rules can be applied. But if one of the chip group is completely removed, Player B will most likely be able to apply the rules that he knows.

- Player A will pick chips blindly, and Player B will pick the least chip(s) by following the four Game rules.

- If Player A could fortunately survive any round and move to the next round, we would assume Player A picked the least chip(s) in that particular round to simulate the best winning scenario for Player A.

- Both Players will not pick the chip(s) from the smallest group to advance to the next round. We assume both Players are very conservative and avoid taking the chips from the smallest group.

Based on the above scenario, Player A has a 0% chance to immediately lose the game, and since the chip set has more chips compared to the last project, the game will be mostly composed of random picks and further leads to a win rate of 50% chance. Unless one group is completely removed, since then the 3 chip game rules can be applied, and we can just refer to our last 3-chip game project's case modeling.

Case III: Player A (Go First) vs. Player C (Go Second)

Initial condition (6, 8, 10, 12), with Player A randomly picking and Player C using the 5 rules from above.

Similarly like the last case, the rules cannot be directly applied, and leads to a random picking situation. Despite knowing the fifth rule, Player C will most likely end up with a random picking situation before one whole group is removed, or in this case, when the number is reduced low enough. ex: (0, x, y, z) or (1, x, y, z)

So before that happens, the idea of randomly picking can be applied from the case before.

Since both players will mostly likely do random pick, the fifth rule with created specifically focuses on the ending cases and ensuring a higher chance of player C winning as the possible cases we simulated in our java program mostly composed of a lower amount of chips, focusing towards the end of the game.

3. Design Java Programming

The Java program flowchart shown in Figure 1 is the same as previous project. We added an 4 chips rule in the same way. Java program will take only 2 seconds to decide a game result, total less than 1 hour to complete all 3 cases.

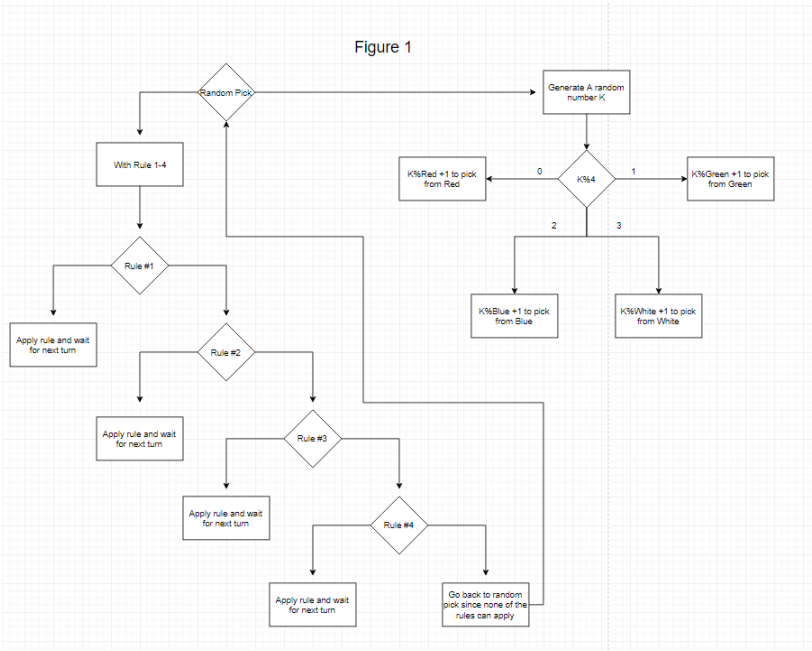

Figure 1: Java Flowchart

Before developing Java programming, our team has brainstormed the following Programming Flowchart to lay out the Java programming modules based on the four game rules mentioned previously:

```
import java.util.Random;
import java.util.Arrays;
import java.util.ArrayList;
public class Chip{
  public int r;  public int g; public int b; public int m;
  private int [] sorting;
  private ArrayList<String> result;
  private String counter;
  private int winnerSumA;
  private int winnerSumB;
 public void Sort(){
  int i=r;
  int j=g;
  int k=b;
  int p=m;
  sorting = new int[4];
  sorting[0] = i; sorting[1] = j;  sorting[2] = k; sorting[3] = p;
  Arrays.sort(sorting);
  r = sorting[0]; g = sorting[1]; b = sorting[2]; m = sorting[3];
 }
 public boolean GameOver(){
  if (r==0 && g==0 && b==0 && m==0)
  return true;
  return false;
```

```
  public boolean Rule3(){
    if(g==b && g>1 && r==0) {
   m=0; return true;
 }
else if(m==b && m>1 && g>0 && r==0) {
  g=0; return true;}
    return false; }public boolean Rule4(){
 if(r == 0 && g == 1 && b==1 && m >1){
 m = 1; return true; }
else if(r ==1 && g ==1 && b==1){
 m = 0; return true; }
else if( r == 1 && b ==2 && m == 3){
 g = 0; return true;}
else if( r == 1 && g ==2 && m == 3){
b = 0; return true;}
else if( r == 1 && g ==2 && b == 3){
 m = 0; return true;}
else if( r == 0 && b ==2 && b == 3){
 g = 1;return true; }
else if( r == 0 && g == 1 && m == 3){
b = 2;return true;}
else if( r == 0 && g == 1 && b == 2 && m > 3){
m = 3; return true;}
else if( r == 0 && g == 1 && b == 3){
```

```
 }
 Chip(int i, int j, int k, int l) {
  Math.max(i,k));
  result = new ArrayList<String>();
  r = i; g = j; b = k; m = l;
 }
 public void RandPick(String player){
   while(RandPickSubfunction(player, false) == false){
   }
 }
public boolean RandPickSubfunction(String player, boolean
debug){
    Random rn = new Random();
    int pick = rn.nextInt(1000);
    if(pick%4 == 0 && r > 0){
      r-=pick%r + 1;
      if(debug)
     }else if(pick%4 == 1 && g>0){
      g-=pick%g + 1;
      if(debug)
     }else if(pick%4 == 2 && b>0){
      b-=pick%b + 1;
      if(debug)
     }else if(pick%4 == 3 && m>0){
      m-=pick%m + 1;
      if(debug)
      }else{
      return false;
    }
    Sort();
    return true;
  }
 public boolean Rule12(){
    if(r==0 && g==0){
      if(b>1 && b<m){
      m=b; return true;
      }
     else if(b==1) {
      m=0;  return true;
      }
      else if(b==0 && m>1) {
   m=1; return true;
  }
 }return false;
 }
```

```
 m = 2; return true; }
 else if(r==1 && g > 0 && b == g+1 && m != 0){
  m = 0;return true; }
 else if(r == 1 &&  g > 1 && b == m-1){
 g = 0; return true;}
else if(r == 0 && g > 1 && b == g+1 && m > 1){
 m = 1; return true;}else if( r == 0 && g >1 && m == b+1
){g = 1; return true; }
else if(r == 0 && g == 1 && m > b + 1){
 m = b+1; return true;}
return false;}
public void Pick(String player){
    if( Rule12() || Rule3() || Rule4()){}
    else RandPick(player);   Sort();  }
  public static void main(String[] args) {
    int x=6, y=8, z=10, r=12;
  Chip list=new Chip(x,y,z, r);
   list.run();}
  public void run(){
 for(int i = 0; i <= 208; i++){
              playGame();result.add(i,counter);
              if(result.get(i).equals("A")){winnerSumA ++;}
else if(result.get(i).equals("B")){winnerSumB ++;}
r = 6;g = 8;b = 10;m = 12;} }
  public String playGame(){
while(true){
RandPick("Player A");  Sort();
 if(GameOver()){
  counter = "B";
  return counter;
 }
 Pick("Player B");
 Sort();
 if(GameOver()){
counter = "A";
return counter;
   }
  }
 }
}
```

Figure 2: Java Program

3.2 Analyze Java Result

After collecting all our data, it showed that the playing sequence does not really matter in the 4 chip game in contracts of the 3 chip one.

What made the difference between Case II and Case III is the addition of the fifth rule. In Case II the 3 chip rules that only player B knows are used towards the end of the game. But as it assumes one chip group is empty, which

means it does not regard the whole group, the plays in which it affected was not as significant compared to 3 chip game model. As for Case III, with Player C knowing the newly added rule 5, which unlike rule 1-4, took in consideration of the whole chip set. With it evaluating the whole group, the chance of it being applied is greater than Case II's rules.

- Sequential Bias isn't as significant with a larger data space (chips)
- The 3 chip game rules that only applied in the end affected the winning probability very much. As both players are randomly picking beforehand.
- The 4 chip game rule, used in Case III in addition to 3 chip game rules showed more of a significant impact on the winning probability as it has more chance of being used, unlike the old 3 chip rules.

Table 2: Case I/II/III Result

Winning Matches	Player A	PlayerA/B/C
Case I Player A vs Player A	1036	1054
Case II Player A vs Player B	122	1968
Case III Player A vs Player C	8	2082

Figure 3: Winning Probability Bar Chart

4. Conclusion

Team has successfully built a predictive model to simulate the winning probability on three Cases.

- There is no significant evidence showing the playing sequence would impact the winning result. This result is making sense since we are assuming all events are independent. This independency should be more accurate when we have more chips in the pool.
- Player B (knowing four rules) has a much higher winning probability (> 95% chance) over Play A (playing blindly). Our predictive model can accurately predict the winning probability if we can take 5 or 6 rounds.
- Team has conducted the sample size calculation in order to draw a statistical conclusion to verify the two hypotheses. Developing a Java programming has significantly reduced our effort to collect data to validate our predictive model.

5. Future Work Opportunities

Team has built a very basic Java model to simulate the Powerball Probability. Team could have done it better on the following future opportunities:

- Consider the Prize Model to adjust the expected value and the probability uniformity across bigger prizes
- Search the historical tickets-sold amount distribution to more accurately simulate the No-Jackpot probability
- Analyze the Roll-Over pattern on the tickets sold amount to more accurately predict the Jackpot Amount distributions
- Build a model of two Mega Balls to create an even bigger Jackpot.
- Compare Powerball to other Lotto like Mega Millions or Super Lotto

Acknowledgements
Need to acknowledge Dr. Charles Chen here for the guidance and mentorship here….

References
1. Toupo D., and Strogatz S., *Nonlinear dynamics of the rock-paper-scissors game with mutations*, Physical Review E, Published 11 May 2015.
2. Maciejeweski W., Fu, F., and Hauert, C., Evolutionary Game Dynamics in Populations with Heterogenous Structures, *PLOS Computational Biology*, published April 24, 2014.
3. Liu T., Chen M., and Jang J., Statistical JAVA Gaming Simulation, Accepted by IEOM Conference, Rabat, Morocco, 2017.

Biography

Timothy Liu is currently a student in the Lynbrook High School. He has been learning Java Programming and AP Statistics. He has certified IBM SPSS Statistics Certificate, IBM Modeler Data Analysis and Data Mining Certificates. He has also received a solid Six Sigma Yellow Belt Training this summer and familiar with DMAIC Six Sigma Problem Solving Methodology. Timothy got invited and presented his JAVA Project in the Local American Society for Quality (ASQ) Statistics and Reliability Group. He is taking part in Lynbrook wrestling. He also won the 2nd place 2016 Shuai Jiao Teen open at the regional competition. At his spare time, he enjoys comic drawings. He is the co-author of 3 books: "Mindy: The Tropical Trio" ISBN 9781622280001, "Mindy: Friend or Foe?" ISBN 9781537301792, and "Spellbound Story" ISBN 9781492715856.

Mason Chen is currently a student in the Milpitas Christian School. Mason has certified IASSC (International Association of Six Sigma Certificate) Lean Six Sigma Yellow Belt, Green Belt, and Black Belt Certificates. He has also certified IBM SPSS Statistics Certificate, IBM Modeler Data Analysis and Data Mining Certificates. He also won the 1st Place Award this year on the Mental Math and Abacus Math contests in the North California Region,

USA. Mason Chen is familiar with Lego Robotics/EV3, Six Sigma DMAIC, DMADOV, Lean Production, Minitab, SPSS Statistics, SPSS Modeler CRISP Data Mining, AP Statistics, Java Programming, and ASQ (American Society for Quality) Quality Engineering. Mason got invited to present his five ASA (American Statistical Association) team statistics projects for 90mins in the local ASQ Statistics and Reliability Group this summer. He will also be invited to ASA CSP (Conference of Statistical Practice) next year in Florida. He is the founder of Mason Chen Consulting which organization helps develop young kids on Big Data Analytics and STEM Projects. Recently, he is also learning "Computational Biology" which integrates Biology, Chemistry, Physics, Mathematics, Statistics, and Computer Science fields.

Joseph Jang is currently a student in the MontaVista High School. He is currently taking AP Computer Science and Ap Statistics. He has certified IBM SPSS Statistics Certificate. Joseph got invited and presented his JAVA Project in the Local American Society for Quality (ASQ) Statistics and Reliability Group. He was also part of a relay team and won second place at the regional school competition.