Final Project

Probability and Applied Statistics

Contents

[Part 1: Stats Problem Solves 2](#_Toc121517493)

[Section 1.2: Characterizing a Set of Measurements: 2](#_Toc121517494)

[Section 1.3: Characterizing a Set of Measurements 3](#_Toc121517495)

[Section 2.3: A review of Set Notation 3](#_Toc121517496)

[Section 2.4: A probabilistic Model for an Experiment: The Discrete Case 3](#_Toc121517497)

[Section 2.5: Calculating the Probability of an Event: The Sample-Point Method 3](#_Toc121517498)

[Section 2.6: Tools for Counting Sample Points 4](#_Toc121517499)

[Section 2.7: Conditional Probability and the Independence of Events 4](#_Toc121517500)

[Section 2.8: Two Laws of Probability 5](#_Toc121517501)

[Section 2.9: Calculating the Probability of an Event: Event Composition. 5](#_Toc121517502)

[Section 2.10: The Law of Total Probability and Bayes Rule 5](#_Toc121517503)

[Section 3.2: The Probability Distribution for a Discrete Random Variable 5](#_Toc121517504)

[Section 3.3: The Expected Value of a Random Variable or a Function of a Random Variable 5](#_Toc121517505)

[Section 3.4: Binomial Probability Distribution 6](#_Toc121517506)

[Section 3.5: The Geometric Probability Distribution 6](#_Toc121517507)

[Section 3.7: The Hypergeometric Probability Distribution 6](#_Toc121517508)

[Section 3.8: The Poisson Probability Distribution 7](#_Toc121517509)

[Section 3.11: Tchebyscheff’s Theorem 7](#_Toc121517510)

[Section 4.2: The Probability Distribution for a Continuous Random Variable 7](#_Toc121517511)

[Section 4.3: Expected Values for Continuous Random Variables 7](#_Toc121517512)

[Section 4.4: The Uniform Probability Distribution 7](#_Toc121517513)

[Section 5.2: Bivariate and Multivariate Probability Distributions 8](#_Toc121517514)

[Section 5.3: Marginal and Conditional Probability Distribution 8](#_Toc121517515)

[Part 2: Statistics Equation Library 9](#_Toc121517516)

[Part 3: Formal Project Report 12](#_Toc121517517)

[Section 1: Make and Solve Stats Problems 12](#_Toc121517518)

[Section 2: Salting and Smoothing with JFreeCharts and Apache 13](#_Toc121517519)

[Section 3: Learning how to use Octave 15](#_Toc121517520)

[Section 4: Normal, Beta, and Gamma Distributions 18](#_Toc121517521)

[Section 5: Poker Project 21](#_Toc121517522)

# Part 1: Stats Problem Solves

The dataset I chose to do my final project on is Covid cases throughout 2020 as reported by the New York Times. My dataset had to be cleaned up a little bit, so I took out the places and summed up the total cases by date, instead of having an abundance of location data, which would make it harder to solve in a meaningful way. The war data can be found here: <https://github.com/nytimes/covid-19-data/blob/master/us-counties-2020.csv>

You will find my Scrubbed data set in the assignment submission.

### Section 1.2: Characterizing a Set of Measurements:

Measuring the rate of infection of covid was extremely important in the early days of the pandemic before we had testing and any kind of vaccine or therapeutic treatment. We had no sort of immunity against this disease. The rate of infection was directly related to how many people ended up in the hospital; we had to tailor public policy to ensure we didn’t overrun the hospital system and keep the economy open enough so that society wouldn’t collapse. I have constructed a relative histogram for all the months in 2020 and the covid cases in those months:

Chart, histogram

Description automatically generated

Table

Description automatically generated

The chart shows that covid cases went up exponentially, this is probably for many reasons, such as not having tests yet or because the initial spread was probably much quicker than the later spread of the illness. The highest frequency on this chart was in December with 525,973,531 cases.

### Section 1.3: Characterizing a Set of Measurements

The Standard Deviation, Mean, and Variance are 5167288, 4992165, and 26700900000000. This variance figure seems incorrect, but because a lot of numbers are very far away from the mean, it makes sense to have a large variance.

The mean of a set of data identifies the center of it and doesn’t necessarily categorize it. The variance describes how different the data can be from each other. Sets with a large Variance, such as this set, are going to have data points that are very far away from each other.

### Section 2.3: A review of Set Notation

It appears that my dataset is cumulative, which means that all the covid cases in November is a subset of December, so . This relation would be a proper subset because all of November’s cases are counted in December, but December has cases that aren’t counted in November.

### Section 2.4: A probabilistic Model for an Experiment: The Discrete Case

Assume that a person is equally as likely to get covid on any given day. The sample space for this event would be S = {E0, E1}, because there are only 2 possible outcomes. The P(S)=1/2. The Venn diagram of the sample space is as follows:

S

* E2
* E1

### Section 2.5: Calculating the Probability of an Event: The Sample-Point Method



Assume that there is an equally likely chance that 3 people test positive for covid. The three people are tested back-to-back, what is the probability that exactly 2 people have covid?

There are 8 sample events:

E1 = +++, E2 = ++-, E3 = +-+, E4=-++, E5+--, E6=-+-, E7=--+, E8=---

So out of the 8 possible outcomes, we are only looking for ones that contain 2 positive values. That is E2, E3 and E4. Because there are 8 of these sample events, and each of them are equally as likely, there is a 1/8 chance that any one of these events are chosen, since we have identified 3 that match what we are looking at, we can conclude that (1/8)+(1/8)+(1/8) = 3/8

### Section 2.6: Tools for Counting Sample Points

The problem in section 2.5 can have the sample events represented by the mn rule. This states that one can calculate the number of sample events by taking the discrete ways that event A can happen and multiplying it by the number of discrete ways that B can happen. This effectively simulates a matrix multiplication event. Because the events had 2 different outcomes, and we were looking at sets of exactly 3 outcomes, the mn rule would be 2\*2\*2 or 8 different possible events. This is reflected in how every event was represented as (1/8).

### Section 2.7: Conditional Probability and the Independence of Events

Consider the following table:

Table

Description automatically generated

This table shows the Sum of Covid Cases as the year progresses and the Sum of all covid deaths as the year progresses. Presumably, one is not able to die of covid if they don’t contract it. This shows that these variables are independent or dependent mathematically.

Assume variable A is Sum of Cases and Variable B is Sum of Deaths. Assuming that about half the population got covid in 2020 means A=1/2. It appears that about 2% of the people who got covid, died (these numbers were double checked from the New York Times, The Number of cases alone is 15 times the size of the US population). It appears that 1/100 of the general population died from Covid and 2/100 if A has occurred to the subject.

As stated before P(A) = ½, P(B) = 1/100 or 2/100. Since these probabilities are equal to the probability of A, these events are independent.

### Section 2.8: Two Laws of Probability

Because the example from 2.7 are independent, then

### Section 2.9: Calculating the Probability of an Event: Event Composition.

Of the population in the United States, about half of the population has gotten covid. Of the general population, 1/100 people have died from covid. Let P(D) = Died from Covid. Let P(C) represent people who have had covid. And Let P(L) represent people who have never had covid. Therefore

P(C) = ½, P(L)= ½. P(D|C) = 2/100, P(D|L)=0.

The probability of someone belonging to the general population dying of covid is 1%

### Section 2.10: The Law of Total Probability and Bayes Rule

A particular strain of covid causes a foot rash in 72% of people, while the old strain caused a foot rash in 32% of people. If a person from the population was chosen at random, what is the chance that they experienced the foot rash? This can be represented from the book by If we plug in (.28)(.68)+(.72)(.32)=.4208 or 42% chance that the person experienced the foot rash.

### Section 3.2: The Probability Distribution for a Discrete Random Variable

A doctor is attempting to determine how likely it is for him to instruct a nurse to work with a covid patient, when the nurse hasn’t had covid and has no natural immunity him/herself. The doctor knows there are 12 nurses on his staff, 8 of them have had the disease and 4 of them have not. The doctor needs 2 nurses to assist him with a covid patient. What is the probability that he picks 0,1 or 2 nurses who have NOT had the disease?

### Section 3.3: The Expected Value of a Random Variable or a Function of a Random Variable

Covid researchers are attempting to find the efficacy of a vaccination against covid, their findings were y = weeks after taking the vaccine and p(y) is a successful measurement of antibodies in the blood stream. The distribution is given by this chart:

|  |  |
| --- | --- |
| y | P(y) |
| 0 | 1/16 |
| 1 | 3/16 |
| 2 | 4/16 |
| 3 | 6/16 |
| 4 | 4/16 |

Find the variance.

And

and

### Section 3.4: Binomial Probability Distribution

One common complication from covid is pneumonia. It occurs in 31% of the population. If 15 people who have had covid were selected at random, what is the probability that less than 6 of them suffered from pneumonia during covid?

This number almost rounds to zero.

### Section 3.5: The Geometric Probability Distribution

Take the example from 3.4 but instead find the first person who suffered from covid with a geometric probability distribution.

4.8% is the probability of finding the first person who suffered from pneumonia after covid on the 6th trial.

### Section 3.7: The Hypergeometric Probability Distribution

Suppose that amongst a shipment of 100 covid tests, 50 of them are bad. If a nurse takes 75 of them to resupply a particular testing site, what is the probability that all 50 of the bad ones get taken by the nurse.

Such a miniscule amount that it is almost impossible

### Section 3.8: The Poisson Probability Distribution

The distribution of covid cases are randomly disbursed across the entire US, with the mean density of cases = 2 per square yard. If you circled an area of 100 square yards on a map of the US, what is the probability that the area you circled will have 4 covid cases?

Or 9% chance that the area circled would have 4 covid cases.

### Section 3.11: Tchebyscheff’s Theorem

The number of patients taking up hospital beds has been found to be a mean of = 50 and standard deviation of 6. What’s the Probability that tomorrow, the number of patients taking up hospital beds are 50<y<62.

### Section 4.2: The Probability Distribution for a Continuous Random Variable

The probability that a person in the U.S. has had covid more than once can be represented by the expression:

Find F(y):

### Section 4.3: Expected Values for Continuous Random Variables

Find the expected value of the probability density function from 4.2:

Thus, the integral does not exist. We cannot calculate the Expected Value from this function.

### Section 4.4: The Uniform Probability Distribution

Suppose that during the ventilator shortage, the probability that someone would need to be put on and someone who was coming off was proportional only to the length of time. So every 2 minutes, 3 people needed to go on a ventilator and 3 ventilators became available. This would follow a uniform distribution. Calculate the probabilities for

Therefore, these probabilities have the same values.

### Section 5.2: Bivariate and Multivariate Probability Distributions

A hospital has 5 beds, 4 patients arrive at the hospital at different times and are not ever in the hospital at the same time. Each patient is put in a bed randomly. Find the joint probability function of Y3 and Y4

Let Y1 be the patient who used bed 1,

Let Y2 be the patient who used bed 2,

Let y3 be the patient who used bed 3,

Let Y4 be the patient who used bed 4,

Let y5, be the patient who used bed 5

Our sample space is

S = [{1,1},{1,2},{1,3},{1,4},{2,1},{2,2},{2,3},{2,4},{3,1},{3,2},{3,3},{3,4},{4,1},{4,2},{4,3},{4,4}]

|  |  |  |  |
| --- | --- | --- | --- |
| Y3 |  | Y4 |  |
|  | 0 | 1 | 2 |
| 0 | 1/4 | 1/4 | 1/16 |
| 1 | 1/4 | 1/8 | 0 |
| 2 | 1/16 | 0 | 0 |
|  |  |  |  |

### Section 5.3: Marginal and Conditional Probability Distribution

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Y3 | Y4 | | |  |
| 0 | 1 | 2 | Total |
| 0 | 1/4 | 1/4 | 1/16 | 9/16 |
| 1 | 1/4 | 1/8 | 0 | 6/16 |
| 2 | 1/16 | 0 | 0 | 1/16 |
| Total | 9/16 | 6/16 | 1/16 | 1 |

Find the probability that bed 4 is used 0, 1 and 2 times.

This problem is fairly easy to solve because we’ve done all the hard work to get our answer. All we have to do is add up all the probabilities for the Y value we want, and we will know how much bed 4 is used.

P(0) = 1/4+1/4+1/16 = 9/16

P(1)= 1/4+1/8+0=6/16

P(2) = 1/16

## Part 2: Statistics Equation Library

Mean

Variance

Standard deviation

Permutation

Combination

Conditional Probability of event A, given that event B occurred

The multiplicative law of probability

A success occurs on or before the nth trial

A success occurs before the nth trial

A success occurs on or after the nth trial

A success occurs after the nth trial

Probability distribution for a discrete variable y

Expected Value of discrete random variable

Variance of a discrete random variable

Binomial distribution of a random variable based on n trials a success probability p

Mean of Binomial Distribution

Variance of Binomial Distribution

Geometric Probability Distribution

Geometric Probability Distribution Mean

Geometric Probability Distribution Variance

Hypergeometric Probability Distribution

Poisson probability distribution

Tchebysheff’s Theorem

Preconditions of a Distribution Function

How to convert distribution function to probability density function (pdf)

Expected Value of a continuous random variable Y is

Uniform Distribution Function

If Y1 and Y2 are jointly continuous random variables with a joint density function f(y1,y2)

Marginal Probability Functions of y1 and y2

Conditional Discrete Probability Function

## Part 3: Formal Project Report

### Section 1: Make and Solve Stats Problems

The dataset I chose for this project is the New York Times covid 19 case data. In hindsight, this was not a good dataset because it didn’t give me much flexibility when I was writing statistics problems. I often found myself writing problems that didn’t really have to do with my dataset, even though I did try to stick to the covid theme.

Another issue I had with my dataset was that the numbers seemed off, so much so that I went back, and triple checked, not only my math, but also the source. These numbers are legitimately from the New York Times, and they claim it’s case data from the US, but the number of cases reaches the billions if I’m reading it correctly, which doesn’t make sense because the population of the US is about 340 million people so, there does seem to be something incorrect about that data.

My statistics had to improve for me to complete get through the problems in this book. I found myself rereading and comprehending things I didn’t quite understand before.

### Section 2: Salting and Smoothing with JFreeCharts and Apache

This program was certainly harder than I anticipated to write. I assumed that there was an Apache library that I could just use to call a data salter and a data smoother function. Unfortunately, no such library exists. One of the first hurdles was that my IDE wouldn’t quite work correctly with the JFreeCharts library or the Apache commons math library. It appears that IntelliJ has some sort of bug with one of those libraries which causes you to have to sometimes rebuild the project. That was the first hurdle.

Another problem that I ran into was finding which part of the libraries I should use, especially Apache, the documentation didn’t mention anything about a graph smoother or graph salter. One great thing I found was that JFreeCharts has this object called TimeSeriesCollection. This worked well for me because one of my data points is a Date. This feature allowed the chart to format itself properly with the dates instead of treating every x axis label as if it were it’s own string datatype.

The salted data I got had to write by myself. I could not find any useful Apache math function that would salt the graph data for me. My Salter is very simple. It takes a piece of data and adds a random number between =1 million and 1 million to it. I have this parameter set so high because my data set gets up to almost 2 billion. One wouldn’t notice the salting at all if the parameter wasn’t so high.

For the Graph Smoother, I wrote a lot of the logic that manages the ArrayList objects, and I extract the values from them and feed them into an Apache mean function. Oddly enough, this function only took an array of doubles to average, and I had to make another array of all 1’s because it needed a weight parameter. In the end, it probably would have been less messy, and performed better if I had written the algorithm.

One class I did implement in sort of a weird way was a class I called manageData. This class has a constructor that takes a filename and reads the first two columns from the CSV which was supplied into 2 ArrayLists of different types. Once the constructor has run the data is stored in that class, but it still needs to be parsed into a datatype that JFreeCharts can understand. There’s a method in that class called addToDataset that takes the arrayLists global variables from when that method is constructed, and it creates JfreeChart data object and adds them to the dataset.

The Salt and Smoother function takes ArrayList of type integer as parameters, and they have a getter so we can access that array list. This is just a pointer though, when the data is being Salted and Smoothed it is passing the pointer of the original ArrayList object and they are editing the original arrayList. The getter method and ArrayList return type are not always necessary in this code, only when one is going from one object to another and the other object can’t resolve the symbol used in the first object.

The graph that I produced is:

Graphical user interface, chart

Description automatically generated

You’ll notice that there are three lines, a red line, a green Line and a blue line. The red line represents the data without anything done to it. The blue line is the data after it has been salted by adding a random number between -1000000 and 1000000 to the y data point. Once it has been salted, you’ll see the green line. The green line is the salted data after it’s been run through a smoother. I suspect I could get the graph much closer to the original if I used more data points, but 10 is enough for demonstration purposes. It works well enough.

The Y axis labels on this chart were nice and easy to implement. Before I was using the Date object from JFreeCharts to create the data set, the Y axis was unreadable because it treated every single Y axis label as a String that had to be present on the graph. This date format automatically decides what should and what should not go on the axis label. The x axis label also automatically scaled, but it is just Integer values, so that’s expected behavior.

JFreeCharts is something I’ve never used before, and I would say it was easy to learn and once you pick it up it is very intuitive, which is not something you commonly find with the java programming language. I especially liked how easy it is to go back and plot other lines on the same chart by just adding another series of data.

### Section 3: Learning how to use Octave

Daniel Woods

GNU Octave Tutorial report

My initial experience with Octave was that it seemed clunky to use and the syntax was a bit on the strange size. It seems like it is very particular about where the spaces are placed, and it won’t work if you put them in the incorrect places. I had a little trouble picking up the syntax at first, but while I was figuring the program out it became very intuitive and easier. I chose from 2 different tutorials, and I will link them at the bottom.

One of the problems I had initially was that in both tutorials I chose, they were both working in the command line version of Octave and I was trying to write my code in scripts so I could run them multiple times and debug them. This was a problem because when you run code in Octave, the console window doesn’t pop up automatically. I didn’t realize that it was at the bottom at first, so I ran into some trouble when my code wouldn’t run and I had to guess what was wrong.

The first function I decided to write after reading much of the tutorial was to simply plot x from – 10 to 10 and y as the sin(x). I added a variable to the function so that one could specify the coefficient of x as their own custom number. When the function is not supplied to the variable, it defaults to 1. The resulting chart is below:

Chart, line chart

Description automatically generated

The second function that I wrote was from the video tutorial. The function takes no input, and it creates a matrix out of 2 vector variables. The matrix, z, is x transposed and multiplied by y, which is equal to x. It appears that y is extraneous, and z could have been instantiated by z=x’ \* x instead of also creating y, but I shadowed the author of the tutorial. The contour chart can be seen below:

Chart, background pattern, radar chart

Description automatically generated

Some of the shorthand I learned during the tutorial I picked up directly from the Octave documentation. Some important things to remember are that if you’re declaring a matrix manually, you separate rows with a semi-colin. A vector is declared by putting numbers in brackets, separated by spaces. Most math problems are the same syntax as regular math. A couple notable exceptions are exponents which are represented by the base and the exponent separated by \*\*. As mentioned before, you can transpose a matrix by putting an apostrophe right after the variable representing a matrix.

The third script I wrote on Octave was the script that generates a plot from a CSV. I didn’t find detailed instructions in either of the 2 tutorials I was using, so I found a specific tutorial just for this. My implementation uses a method called fopen to generate a file ID required by fscanf and it reads the data into a matrix, which I manipulate and use to plot the data in the CSV file. You’ll see the output pasted below:

Chart, line chart

Description automatically generated

The tutorials I used are:

For introduction: <https://www.youtube.com/playlist?list=PL1A2CSdiySGJ6oZe6XB-TTCFuHc5Fs1PO>

For syntax: <http://www-mdp.eng.cam.ac.uk/web/CD/engapps/octave/octavetut.pdf>

For CSV: <https://www.youtube.com/watch?v=cLo2UOBU5yY>

### Section 4: Normal, Beta, and Gamma Distributions

Daniel Woods

Professor Byron Hoy

Continuous random variable probability distributions can take on certain shapes depending on the probability of the variable being measured. One can determine which class of distribution that a particular distribution belongs to by checking its probability density function (pdf) and comparing it to the formulas and preconditions that exist for each class. One can also look at the shape of the distribution and determine, with a good degree of accuracy, which class it belongs to. Probability distributions that can be attributed to one of these classes have the benefit of being able to solve for certain things more easily.

NORMAL PROBABILITY DISTRIBUTION

The Normal probability distribution is the most common shape that is most often thought of as “the bell curve”. This probability distribution has most of its probability occurring right at the mean. If a distribution is a normal distribution, the density function of Y is:

Unfortunately, with a normal distribution there is no closed form method of finding the integral of an equation of that form. This necessitates the use of a table or computer assistance to find the area (probability) of a certain range of Y (continuous probability variable). To find areas under a normal distribution curve, we utilize the concept of z-scores. Z-scores are calculated as follows:

This equation will give you the area of probability in units of standard deviations. For example, if we would like to find the probability that on a distribution where the mean is 70 and the standard deviation is 8, then we would first calculate the z scores as follows:

This means that our Y values are 6 and 2 standard deviations below the mean respectively. If we consult a z score table, we will find that the areas correspond to those z scores, (that is the area between them and Y=0) are .00009 corresponding to -3.75 and .10565 corresponding to -1.25. Because these areas are corresponding to the left, we must calculate the area by subtracting .00009 from .10565 which equals .10556, or a 10.6 percent change of Y being between 40 and 60 on a standard curve.

Examples of data sets that measure things like height, IQ, and other things where there is a strong average population.

GAMMA PROBABILITY DISTRIBUTION

A gamma probability distribution is said to be skewed to the right. This is seen much less commonly than a normal probability distribution. Most of the area of the curve is located on the left side of the curve, and closer to 0. A gamma distribution function will follow the following form:

The value of can be simplified to , if *n* is an integer. is called the gamma function. is the variable that significantly changes the shape of the curve of the gamma function, and it is known as the shape parameter. is the parameter that is multiplied by and is referred to as the scale parameter because it produces a curve of the same shape, but on a different scale.

When is not an integer and 0 < c < d < , then it is not possible to calculate the area under the curve by evaluating the integral in conventional ways, except when . You must use a computer or a chart, just as in a normal distribution. The expression of area of a gamma distribution between the points c and d is given by the expression:

While the area of a gamma function is hard to calculate, the mean and variance are easier to calculate:

A random variable with a chi-square distribution is called a chi-square random variable if the random variable Y is a gamma-distributed random variable where where v is a positive integer. V is referred to as degrees of freedom. The mean and variance of a chi-square random variable can be calculated as follows:

Chi-square distributions are preferable to standard gamma distributions because it is much easier to find tables for probability of chi-square distributions than regular gamma distributions. If the of a gamma distribution can be represented as , then can be represented by and is a chi-square distribution.

If a gamma density function has parameter , then that function is said to be an exponential density distribution. A function can be classified as an exponential distribution if and the equation matches the following:

BETA PROBABILITY DISTRIBUTION

A beta density distribution can take on a wide variety of shapes and cannot easily be determined by looking at the curve. These distributions exist only if the density function is:

Because y is defined as being between or equal to 0 and 1, you may define a new variable y\* as: if .

The cumulative distribution function of a beta random variable is known as the incomplete beta function. Values of the cumulative distribution function of a beta distribution can be found on a table of values. Using a computer is the most efficient way to get probabilities from a beta distribution

### Section 5: Poker Project

The poker project was by far the most time-consuming part of this project. For whatever reason, I had such a difficult time chasing bugs in my code and it was extremely time consuming. My Poker Project consists of 8 classes. The tester class only institutes the player class, and the player class calls almost everything that needs to be called. The player class provides output and takes player input. The first thing player does is instantiate the dealer class.

The dealer class is a class that performs all the judgements and is used to ensure that card objects get from the players hand to the deck, instead of being discarded. The dealer object is meant to do most of the work under the hood. When the dealer is instantiated, a new deck is also instantiated and declared, this deck is what the dealer uses throughout the game. The deck is not discarded after every turn, so that the experience would be more authentic. When the dealer is instantiated, there are also a lot of global variable Booleans that are set to false, more on that later.

When the player has indicated that they’re ready to play, the play class will call the draw 5 class of the dealer. When this method is called, it instantiates a hand object and removes the first 5 cards from the top of the deck. Throughout the game players may remove cards from their hands or the game may do it automatically throughout the game, this is done through the returnCards method; which not only removes the card from the players hand, but it puts it back in the deck allowing it to be drawn by a different player. The toString in the hand object iterates through all the cards in the hand and calls a method in card called printCard. All the other methods in hand should only be used by the dealer.

The dealer evaluate function is the method where all the fun is when programming. Admittedly, I did implement this in a peculiar way. The evaluate method calls each method that determines what a hand is in a very particular order. Some of the comparison methods remove cards from the hand and put them back in the deck. I did this with the isPair, isTriple and isQuadruple methods because if I don’t remove the cards from the player’s hand, then when other methods call those methods later, it sets off false positives that aren’t easy to check for. The evaluate method adds the results in a particular order to an ArrayList of type Boolean which gets fed into the scrubResults method. This method figures out what hand you have and classifies it. It will set a global String variable which can be obtained from the toString, or referenced directly, because it’s public. There is also a resultID method that returns an int which represents a hand and is very easy to compare. A reset method must be called between evaluating hands so that all the strings can be set back to false.

One thing that is strange in the dealer method is that there is a lot of passing hands back and forth. I am aware that I am passing object references and that everything that alters the hand does it everywhere, but as I was going to change it, I figured if I ever write code in the future, it is not a costly enough operation to impact the performance of my program.

The deck method is instantiated when the dealer method is instantiated. The deck method has 2 for each of the loops that loop through the values of my rank and suit enumerated types to create 52 cards in an ArrayList of type card. The deck class has a shuffle method, but there is a shuffle method in both the dealer class and the play class, this one is only for those classes to use. There is also a draw method that is only used by the dealer, a toString method which should only be used for diagnostics, and putBack method; which again should only be used by the dealer.

The card method is instantiated 52 times when the deck is initialized. The card constructor takes a Rank value and a Suit value from the enumerated types and stores those values within the object. The card class has plenty of methods that aren’t really supposed to be used except by the deck, or when you’re getting the value for comparison (even then, most of that is done by the hand class). Other than the getters and setters, and toString which produces the cards information in text format. I have a printCard method which prints the Unicode suit symbol for the card as well as their value number or letter. This makes the game a little easier to follow because it removes a level of abstraction when playing with familiar symbols instead of just text. This printCard method is called by the hand and concatenated to print out a symbol and card value for every card in the hand.

The two enumerated types I made for this game were Suit and Rank enumerated types. I chose to do this because it was easy to customize them and it makes the programming a little more of a high-level language, as opposed to representing the suit with an integer and doing all the comparisons and computations with that method. It’s also good to get some practice working with them.

My game has 3 modes that you will be prompted to play. The first option is to play 2 different hands against each other. The dealer will let you see your cards and ask if you would like to return any of them to the deck. Once that’s all over, it will evaluate the hands and declare a winner.

The second option is the Monte Carlo Statistics mode. This will play the game as many times as you would like, it tabulates the data and prints them out in a ASCII chart with percentages. I have run it 100,000,000 times with no issues.

The third option is the extra credit arcade mode. You start off with 1000 coins and everytime you get nothing, you lose 100 coins. Different hands have different payouts. The game keeps running until you run out of money, or you close the program.

**Monte Carlo Statistics Mode:**

Graphical user interface

Description automatically generated with medium confidence

**Arcade Mode:**

Text

Description automatically generated