Piece 1

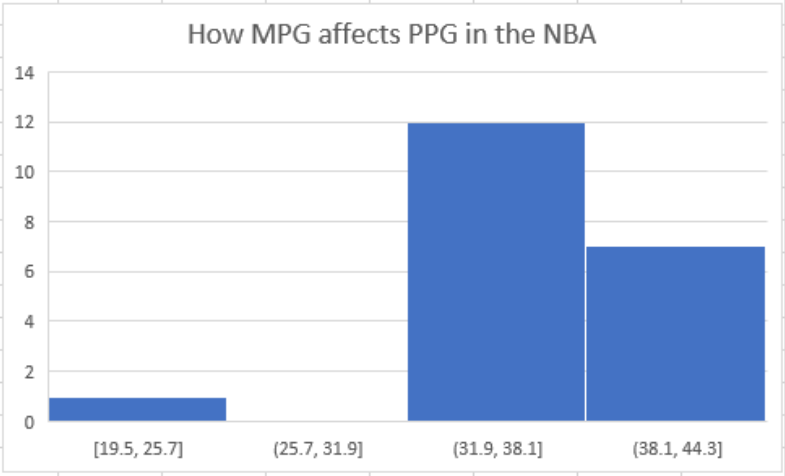
Dataset Problems from Top 20 NBA Players

In this piece of my final project, I will be taking problems from each section of the textbook and changing a question from each section to correlate with my personal dataset. My dataset contains statistics for my favorite twenty NBA players from the 2021-2022 season. The textbook that I am using the question from is Mathematical Statistics with Applications 7th Edition. Within the textbook, I will be taking examples from chapters one through five. The dataset was found online from ESPN's public database. This database is fact checked constantly by ESPN employees as well as NBA employees to ensure that these statistics are accurate.

Section 1.2 Characterizing a Set of Measurements: Graphical Methods:

For this question, I decided that I would create a relative frequency histogram that would highlight the points per game of the player with how many minutes they play. I believe that this graph would be useful because there are many NBA players that play a variety of different minutes however, I believe that the more minutes you play, the more points you would score. Therefore, this relative frequency histogram would show that with more minutes, you score more buckets!

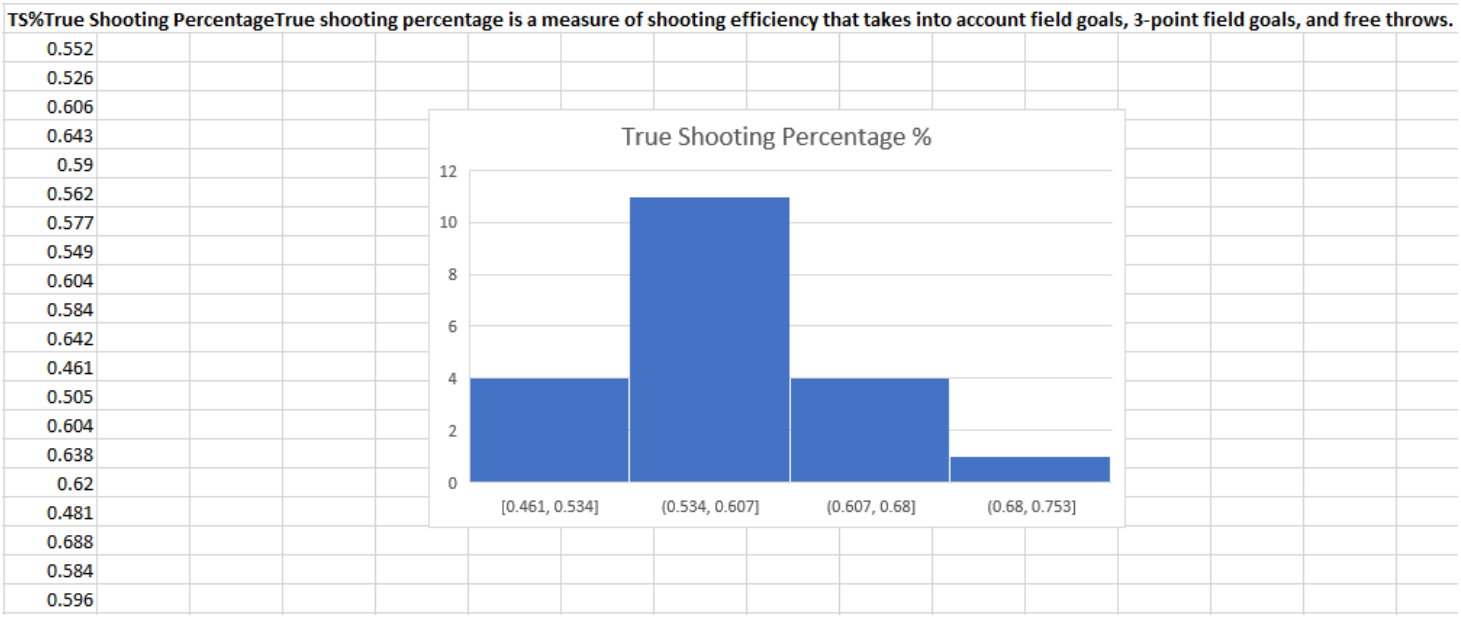
MPG	PPGPointsPoints per game.
37.3	31.7
44	26.3
34.7	27.4
34.2	31
38.5	23.6
41	25.6
36.8	31.7
37.5	27.1
37	27.4
39.9	18.6
37	21.8
37.3	15.4
38.3	25.5
37.8	25.2
34.1	14.8
19.5	8
40.7	20.8
32.8	12
39.3	27
37.7	10.7



Section 1.3 Characterizing a Set of Measurements: Numerical Methods:

When we take a look at the stats of an NBA player, we tend to overlook the true shooting percentage of a player. True shooting percentage is the percent value of all the players attempted shots from the free throw line, any two-point attempts, and three-point attempts. If you take all their attempts and divide it by the number of attempts made, you receive the true shooting percentage. This is something that a lot of NBA players care about because the better the percentage, the better the paycheck. Therefore, to make this histogram, I took the true shooting percentage, highlighted it, and inserting a histogram. However, to complete this section, we also must solve for the standard deviation. In order to solve for the standard deviation, we have to insert our data into the following formula √ (∑ $(x-x1)$^2/ $n-1$). After solving for the standard

deviation with our data, the answer is 0.055331184697239 which means that there is not a lot of variances between these players true shooting percentage. Since these players are some of the best players in the league, it does not surprise me that they are close to each other's true shooting percentage.



Section 2.3: A Review of Set Notation

In street basketball (3 versus 3) there are typically three different positions that can play on the court at the same time. You can have a point guard (otherwise known as the ball handler PG), a forward (typically the bucket getter F), and a big (larger player otherwise known as a center B). However, the game of street basketball is not locked to have one point guard, one forward, or one big. You can have three bigs, three forwards, or any combination of positions in the three-on-three game. Define the super set and the subsets.

A = if there is one point guard in the game.

B = if one big is in the game.

Find the Union and the Intersection of A and B:

S = {(PG, F, B), (PG, F, F), (PG, B, B), (PG, PG, PG), (PG, PG, F), (PG, PG, B), (F, B, B), (F, F, B), (F, F, F), (B, B, B)

A = {(PG, F, F), (PG, B, B), (PG, B, F)}

B = {(B, F, F), (Pg, Pg, B), (B, F, PG)}

$A \cup B$ = {(F, F, PG), (PG, B, B), B, F, PG), (B, F, F), (PG, PG, B)}

$A \cap B$ = {(B, F, PG)

Section 2.4 A Probabilistic Model for an Experiment: The Discrete Case

On a team in the NBA, you need to always have five players on the court. Generally, there are specifically five general positions that most teams follow. Each basketball team's starting five positions tend to be point guard, shooting guard, small forward, power forward, and center. However, these positions are either offensive (denoted by a +) or defensive (denoted by a *). List the sample space for the starting line up of an NBA game.

S = {(PG+), (SG+), (SF+), (PF+), (C+), (PG*), (SG*), (SF*), (PF*), (C*)}

What this sample space is telling us is that although most of the time, the power forward and the center are the defensive players, any of the positions on the court can be an offensive or defensive player. An example of a defensive point guard would be Jose Alvarado because although he doesn't score the most points, he is able to guard players very well. However, an example of an offensive center would be Joel Embiid or Giannis.

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

In the NBA, the starting line up for the point guard and shooting guard positions tend to be very valuable and earn a lot of praise. Most NBA teams tend to have up to five players with a point/shooting guard position. Since there are only two starting spots, what is the probability that the point guard is a defensive point guard?

The guards that we are looking at are:

1. Stephen Curry (+)
2. Jose Alvarado (*)
3. Trae Young (+)
4. Luka Doncic (+)
5. Jayson Tatum (*)

S = {(SC, JA), (SC, TY), (SC, LD), (SC, JT), (JA, TY), (JA, LD), (JA, JT), (TY, LD), (TY, JT), (LD, JT)

Because we know that all the five players have an equal chance of being selected, we know that the probability of them getting selected is 1/5. We have six different sets that have a mainly defense guard. Therefore, there is a 6/10 or 60% chance of a defensive guard.

Section 2.6 Tools for Counting Sample Points

For a home game, the New York Knicks are going to get picked up by the team shuttle before the game so that they can practice. The last eight players need to get picked up locally from around Madison Square Garden. Four of the players are in the Bronx. Another two players are in the heart of New York City, and the rest are in Manhattan. How many different ways does the team shuttle have to go to pick up all of the players.

$$\frac{8!}{(4!2!2!)} = \frac{40320}{96} = 420$$

Section 2.8 Two Laws of Probability

A study on Kevin Durant's 2021-2022 statistics determined that he had a free throw percentage of 89.5% and a two-point percentage of 40.3%. Since free throws and two-pointers are independent of each other, find P(A ∪ B), P(A' ∩ B'), P(A' ∪ B').

P(A ∪ B) = .895 + .403 – (.895 * .403) = 1.298 - 0.360685 = 0.937315 or 93.7%

P(A' ∩ B') = P(A ∪ B)' = 0.062685

P(A' ∪ B') = .105 + .597 - .062685 = 0.639315

Section 2.9 Calculating the Probability of an Event: The Event-Composition Method

In the NBA, 84% of the jerseys are made by the organization themselves and 16% are outsourced by another company. 1.1% of the NBA made jerseys are defective and 8.9% of the third party jerseys are defective. If a jersey is selected at random, what is that chance that the jersey will not be defective.

A = jerseys created by the NBA

B = jerseys created by a third party

X = not defective

P(X) = P(X ∩ (A ∪ B)) = P(X ∩ A) + P(X ∩ B) = .89(.84) + .911(.84) = 1.51284

Section 2.10 The Law of Total Probability and Bayes' Rule

From the majority of basketball fans, 52% of women like to watch mainstream teams such as the Los Angeles Laker or the Miami Heat. While 39% of men like watching mainstream teams like the Boston Celtics. Out of a group of 10 people, 7 females and 3 males, the response was negative towards mainstream teams. What is the probability that it was a female?

P(Female) = 5.2/10 = .52

P(Male) = 3/10 = .3

P(Negative | Female) = 1 - .72 = 0.28

P(Negative | Male) = 1 – 0.39 = 0.61

P(Female | Negative) = .48 * .3 / .28 * .7 + .48 * .3 = 4.235%

Section 3.2 The Probability Distribution for a Discrete Random Variable

A fan of the sport wants to select their favorite three players. There are four forwards and six centers. If the fan selects players at random, what are the odds that the fan selects no centers, one center, two centers, or three centers?

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

$$P(3) = \frac{\binom{4}{0}\binom{6}{3}}{\binom{10}{3}} = \frac{20}{120} = 0.1\overline{6}$$

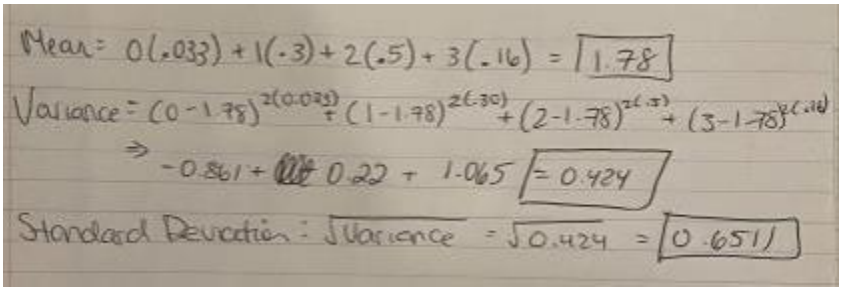
$$P(2) = \frac{\binom{4}{1}\binom{6}{2}}{\binom{10}{3}} = \frac{60}{120} = 0.50$$

$$P(1) = \frac{\binom{4}{2}\binom{6}{1}}{\binom{10}{3}} = \frac{36}{120} = 0.30$$

$$P(0) = \frac{\binom{4}{3}\binom{6}{0}}{\binom{10}{3}} = \frac{4}{120} = 0.0\overline{33}$$

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

Using the previous answers from above, what is the mean, variance, and the standard deviation of Y?



Section 3.4 The Binomial Probability Distribution

Stephen Currey is the greatest three point scorer of all time. He has a 39.7% chance to make every three-point shot he takes. Identify the event favor that Steph's shot as a success S. The probability of S on trial 1 is .397. Consider the event B that S occurs on the second trial. B can occur in two ways, either he makes both shots or the first shot is a make and the second shot is a miss. Show that P(B) = .397. What is P(B|trial 1 in S)? Does this conditional probability differ markedly from P(B)?

The probability of first trial being a failure is 1 - .397 = 0.603

The probability of the second trial becoming a success is .397.

The probability of back-to-back successes = 0.157609

The probability of a failure then back-to-back is = .603 * .397 = 0.239391

Total probability of two trials becoming both successes or the first trial is a failure and the second is a success = 0.157609 + 0.239391 = 0.397

P(B) does equal .397 because the conditional doesn't differ at all from P(B).

Section 3.5 Geometric Probability Distribution

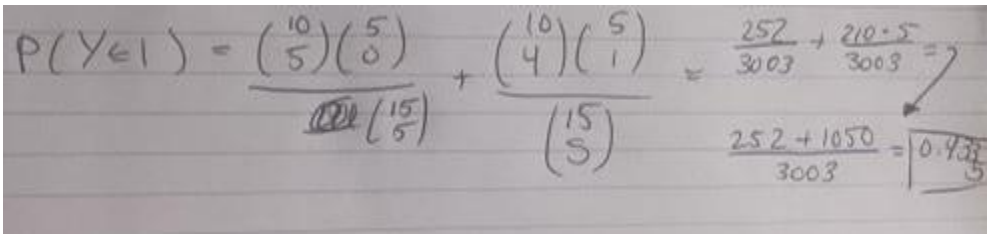
Jose Alvarado has a true shooting percentage of 62%. If we took four NBA fans and told them the true percentage of Jose, what is the probability that the first person thinks Jose would make the three and the fourth player also agrees that he would make his fourth three.

P(X=4) = .62 * (1-.62)^4-1

= .62 * 0.054872 = **0.034**

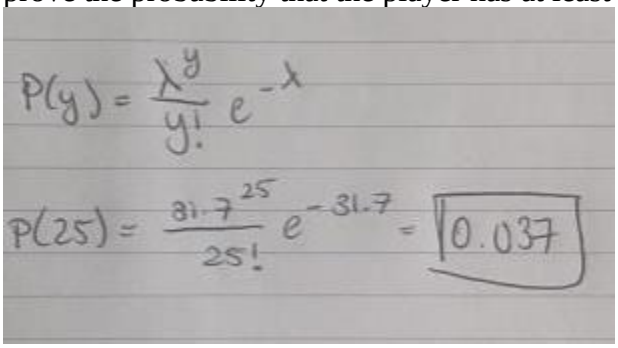
Section 3.7 Hypergeometric Probability Distribution

Five NBA fans were selected by the NBA to select their favorite NBA player from a group of 15 total fans. Of the total fans, 10 of them selected Stephen Curry to be the best player in the NBA and 5 fans selected Giannis Antetokounmpo. The lot was randomly selected and only one Giannis fan was selected? Is there any doubt on the randomness of this selection?



Section 3.8 Poisson Probability Distribution

Luka Doncic scores 31.7 points-per game on average. Use the Poisson Distribution to prove the probability that the player has at least 25 points per game.



Section 3.11 Tchebysheff's Theorem

14.8% of NBA players wear Kobe Bryant's Kobe shoes. If the same proportion of 90 G-League players are interviewed on what shoes they preferred,

a. What is the expected number of Americans who prefer Kobes?

P = .148, n = 90

E(x) = np = 90(.148) = 13.32

b. What is the standard deviation of the number Y who would prefer Kobes?

$\sigma = \sqrt{V(x)} = \sqrt{npq} = \sqrt{90(.148)(.852)} = 3.3687$

c. Is it likely that the number of Americans who preferred Kobes exceeds 13 people?

It is likely because 13.32 + 3.3687 = 16.6887

Section 4.2 Probability Distribution of a Continuous Random Variable

In the past 5 years, the NBA has sold over 110 (in thousands) jerseys in the United States. The total amount of jerseys sold in a 5 year span is a random variable Y with a probability density function given by:

$$\begin{cases} y, & 0 \le y \le 1 \\ 5 - y & 1 \le y \le 2 \\ 0, & elsewhere \end{cases}$$

a. Find F(y): For 0 < y < 1 $F(y) = \int_0^y t dt = \frac{y^2}{2}$

   For 1 ≤ y < 2, $F(y) = \int_0^1 t dt + \int_1^y (5 - t) dt = 5y - \frac{y^2}{2} - 1.$

b. Find the probability that the sales will be between 100 and 120 copies in a 5 year span.

$$P(.92 \le 1.1) = F(1.1) - F(.92) = 4.445$$

Section 4.4 Uniform Probability Distribution

Giannis Antetokounmpo dribble twice to get to the end of full court. If he dribbles the ball one time, which half of the court would he be on? (A is the half with the opponents basket, B is his basket that he is defending). Find the probability that he is closer to the opponent's basket than his defending basket.

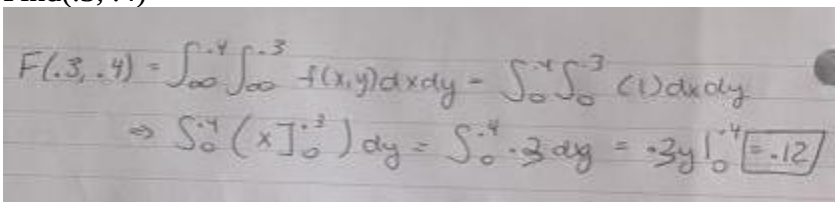
If Giannis is in the opponent's half of the court, he is in the interval (A, $\frac{(A+B)}{2}$). This would be half of the total interval length; therefore, the probability of this happening is 0.50 or 50%.

Section 5.2 Bivariate and Mulitvariate Probability Distribution (was unable to use database, wrote example from textbook and solved it for Chapter 5)

Suppose that a radioactive particle is randomly located in a square with sides of unit length. That is, if two regions within the unit square and of equal area are considered, the particle is equally likely to be in either. Let X and Y denote the coordinates of the particle's location. A reasonable model for the relative frequency histogram for X and Y is the bivariate analogue of the univariate uniform density function:

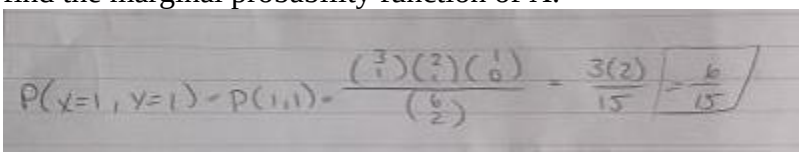
$$f(x, y) = \begin{cases} 1, & 0 \le x \le 1, 0 \le y \le 1 \\ 0, & elsewhere \end{cases}$$

a. Find(.3, .4)



Section 5.3 Marginal and Conditional Probability Distribution

From a group of three Republicans, two Democrats, and one independent, a committee of two people is to be randomly selected. Let X denote the number of Republicans and Y denote the number of Democrats on the committee. Find the joint probability function of X and Y and then find the marginal probability function of X.

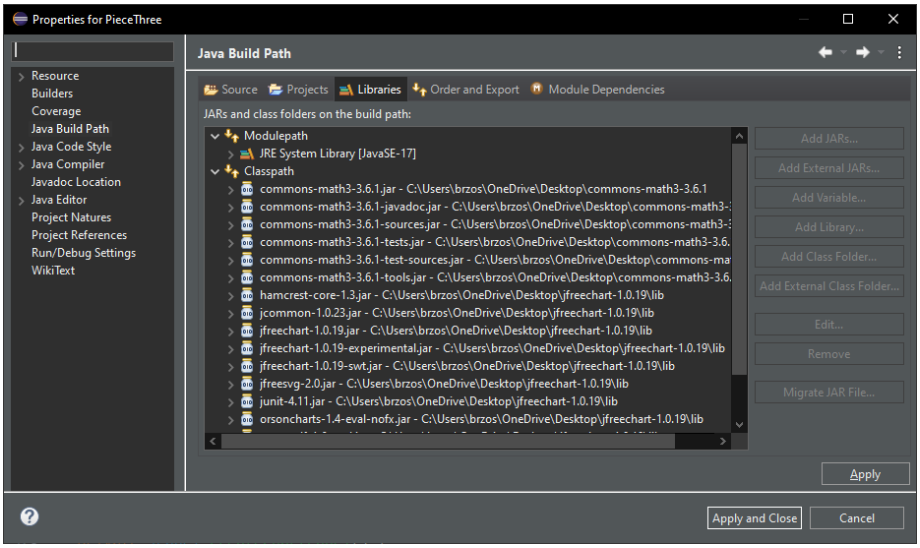


Piece 2

Plot, Salt, Smooth Data

With JFreeCharts and Apache Stats Libraries

At first, this piece of the project was giving me a lot of trouble because I was not importing the two libraries to the classpath. Instead, I was having the .jar files imported into the modeulepath. This was the first time that I was importing a library therefore, it took me a few minutes to determine that maybe it should be in the classpath instead of the module path. After moving JFreeCharts and Apache stats library to the classpath, I was able to import anything from those libraries so that I could complete the assignment. However, first I needed to understand how both the libraries work so that I could program anything with them.



Prior to showing the graph of my results, I wanted to explain how I was able to program this piece of the project and get it to plot the original, the salted, and the smooth graphs of the function y = 9x + ½. I chose to do the same function in piece 3 where I used octave to graph the original and then also graph the salted version as well as the smoothed graph. First, I extended Plot to JFrame and I would then create the JFreeChart. The documentation on JFreeCharts was very helpful since I was able to easily create the JFreeChart that I wanted. I was able to set the size to 800 x 800 and allowed the button on the top right (top left for MAC users) to close the application.

```
public Plot(int userValue) { //initialize the JFreeChart window
    super("Piece 2"); //names the GUI to "Piece 2"
    this.value = userValue;
    JPanel panel = createPanel(); //creates the panel
    add(panel, BorderLayout.CENTER);

    setSize(800, 800); //initializes the size to 800 x 800
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //allow the button to close the graph
    setLocationRelativeTo(null);
}
```

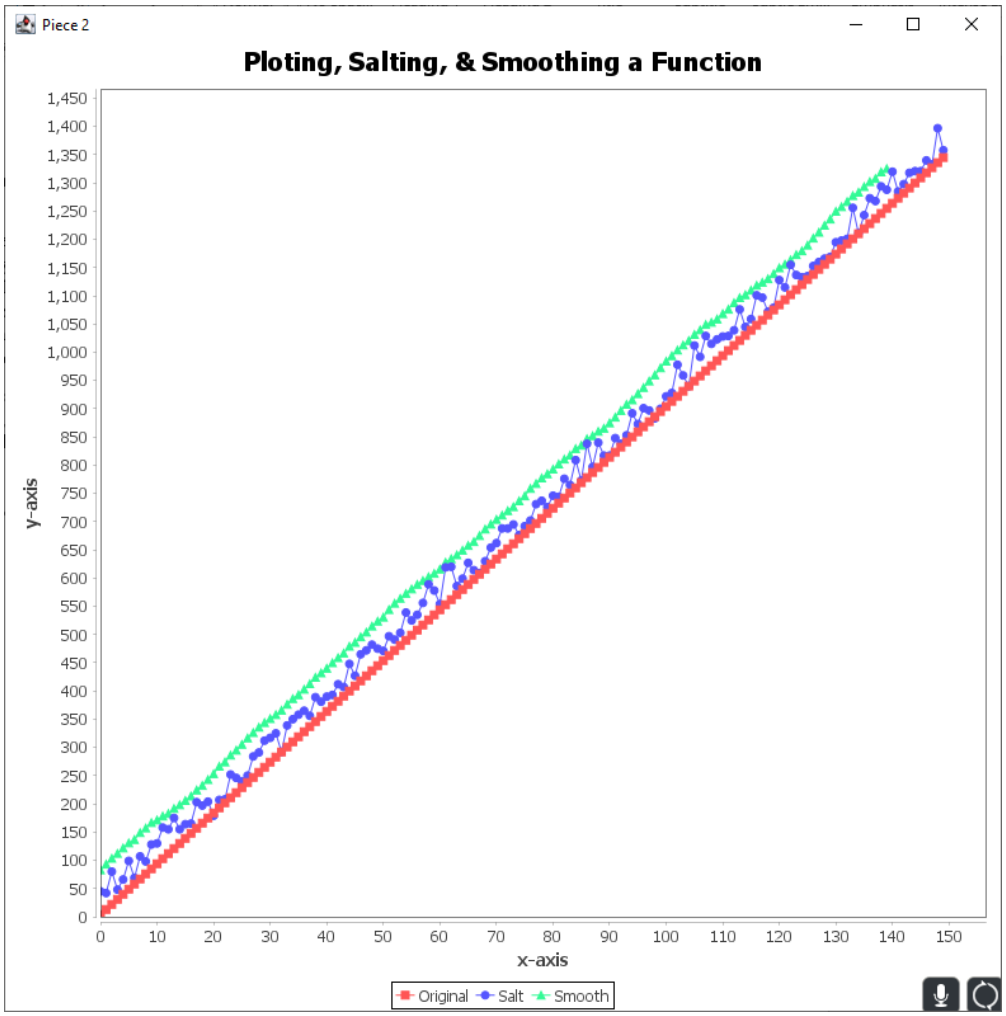
Next, I would make a method that would create the panel for the chart. Using JPanel, I was able to create the panel for the graph. This method allowed me to set the title of the chart, set the x-axis title, and the y-axis title. I was able to create a new XYDataset dataset equal to the generateDataset() method which would fill in the data regarding the original graph, the salted, and smoothed graphs. After initializing the plot, I was able to set a few other

```
private JPanel createPanel() {
    String title = "Ploting, Salting, & Smoothing a Function"; //sets the title
    String xLabel = "x-axis"; //labels the x-axis
    String yLabel = "y-axis"; //labels the y-axis

    XYDataset dataset = generateDataset(); //calls the generateDataset() method
    JFreeChart chart = ChartFactory.createXYLineChart(title, xLabel, yLabel, dataset);
    //creates the chart and includes the title, xLabel, yLabel, and the dataset.
    XYPlot myPlot = chart.getXYPlot();
    XYLineAndShapeRenderer renderer = (XYLineAndShapeRenderer) myPlot.getRenderer();
    renderer.setBaseShapesVisible(true);
    chart.getXYPlot().setBackgroundPaint(Color.WHITE); //sets the background color
    myPlot.getRenderer().setSeriesPaint(2, new Color(50, 250, 150)); //sets the color of functions
    return new ChartPanel(chart); //returns the chart
}
```

things such as the color of the background in the graph which I set to white. I also was able to change the individual colors of the plotted functions as well.

The generateDataset() method is another method and the first few lines are relatively self-explanatory. You are able to see that a random number generator is initialized as well as a graph variable. Then, I was able to create the Arrays for the x and y variables. This would allow me to have the coordinates for the point that I would be graphing. I would create a for loop so that I can populate the x[] and y[] arrays with the proper values. After I added values to both x[] and y[], I was able to add x[i] and y[i] to the graph.

To salt the data, I created a for loop that would have three if statements. In these if statements, if the random value is 0, 1, or 2 then, it would add a random value to the y value. This would be different values for the three if statements. To smooth the data, I created a for loop that would add 10 increments from the mean. Finally, I added the original graph, the saltData graph, and the smoothData graph to the dataset. Within my tester class, I was able to simply call the plot from the SwingUtilities library in javax.swing. When doing this final step, you can set the size of the x-axis. I personally set my graph to 150 x values because I wanted to have the functions shown more visibly. Below is my screenshot with the results of this piece of the project.

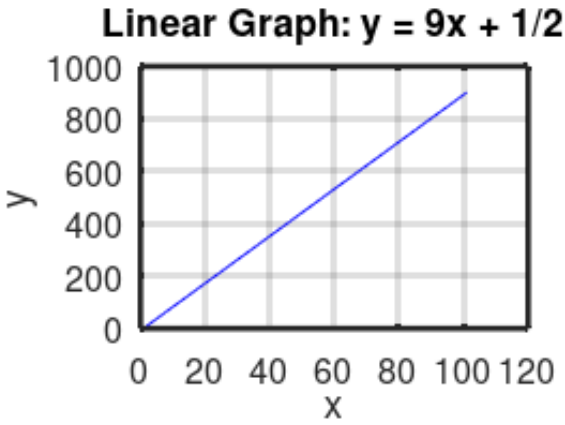


Piece 3

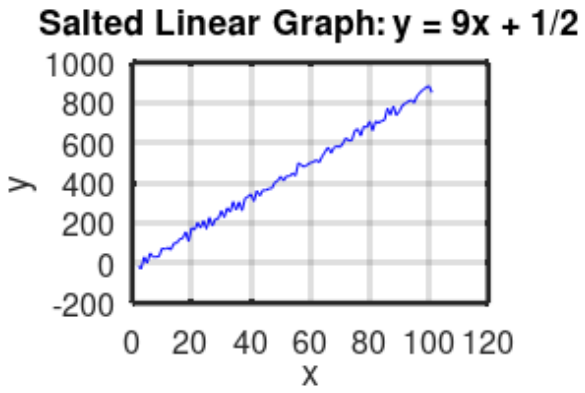
Octave Tutorial - Plot, Salt, Smooth Data

The tutorial that I decided to use so that I can learn about Octave/Matlab is the one and a half hour video by Mr. STEM EDU TV. It is titled “Octave Tutorial for Absolute Beginners: Learn Octave in 1hr and 30 min”. During this video, it taught me the bare basics regarding the programming language ranging from how to install octave on my desktop to writing functions in octave. What was interesting to me was that since I already learned a few different programming languages, this one was a lot easier to learn and grasp the concepts of.

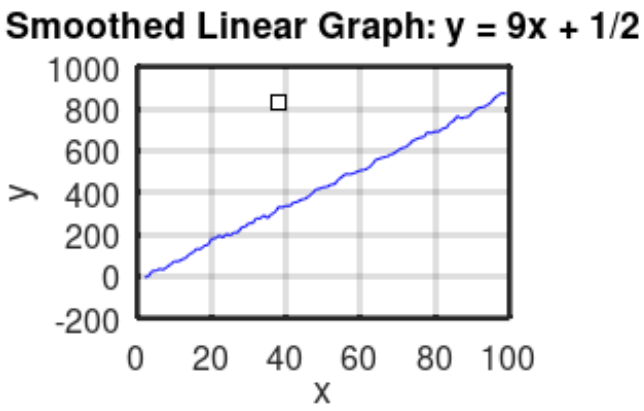
For this piece of the project, we had to pick a function and then plot, salt, and smooth the data. When I was deciding what function to plot, I decided to pick a more interesting function rather than the traditional y=mx+b. In my program, I plotted the function y=9x+1/2. To create the first graph, we can plot the function. The simple way to plot the function would be to set two variables. We would set x to x = 0:10. This would populate the x variable with [0 1 2 3 4 5 6 7 8 9 10]. For us to create the y variable, we would have y = 9*x + ½. Therefore, if we are plotting the point x = 4, we would have y(4) = 9*4 + ½.



I wrote a function named createPlot() that would create empty vertices (e.g. x = []). For x, I wrote a for loop that you populate x with numbers from 0 to 100. You can write this as x = 0:100. For y values, we would have a y = (9*x) + ½. This would populate all the variable in y with the corresponding values to the x values. Once those two variables are created and populated with values, we can run a few lines of code that will add things such as x-axis labels, y-axis labels, displaying the x values and the y values, turning on the grid, and altering the linewidth and the fontsize so that the graph is easier to read. There is also a plot title. Once the graph is created, towards the end of the function I would call the saltData(x,y) function.



The next function that I wrote was so that I can salt the data. We would have the data x values populate in the same way and then we would have the y values populate by creating a for loop for all the variables in x. Therefore, the y values would be populated by random values on the y axis. This graph also has similar code to the one above other than populating the x and y variables. This graph has code to display y, plot the salted x and y variables, add a title to the graph, add a x label on the x-axis, add a y label on the y-axis, turn the grid on, and then it calls the smoothData(x,y) function.



The final function that I wrote is a smoothData(x,y) function that will create a variable s and have it equal to an empty vertices []. I then would create a loop that will input values into s that is (y(i) + y(i+1) + y(i+2))/2. That creates the values that is required for s so that the smoothData function works properly. Then, the same lines of code would be written for the graph title, the x-axis label, the y-axis label, turning on the grid, and changing the linewidth and the fontsize.

Overall, my thoughts on this portion of the project was that this was an interesting and relatively easier piece of the final project. I believe that this is a piece that should be kept for the future because although I may not even specifically use octave or matlab again, I was able to learn a new programming language easier than any of the previous languages that I have used. I believe that it also gives me a great insight to what matlab/octave really are because in my previous classes such as discrete math, foundations, and calculus my professors have always demonstrated graphs to us using this programming language.

I did find that learning Octave was more enjoyable than other programming languages. I wish that more programming languages used similar syntax because I believe that this language was very self-explanatory and beginner friendly. If you were to do this project again for the next semester, I think that your future students would benefit from learning this language and watching a tutorial.

```
function createPlot()
  x = 0:100
  for i = 1:length(x)
    y = (9*(i) + 1/2)
  endfor

  subplot(2,2,1)
  plot(x,y,"b")
  title("Linear Graph=9x+1/2")
  hold on
  disp(x)
  disp(y)
  xlabel('x');
  ylabel('y');
```

```
function saltData(x,y)
  counter = 1
  for i = 1:length(x)
    a = randi(15,1)
    b = randi(50,1)
    if(mod(counter,2) == 1)
    y(i) += (a-b)
  else
    y(i) -= (a+b)
  endif
  counter++
  endfor
    disp(y)
```

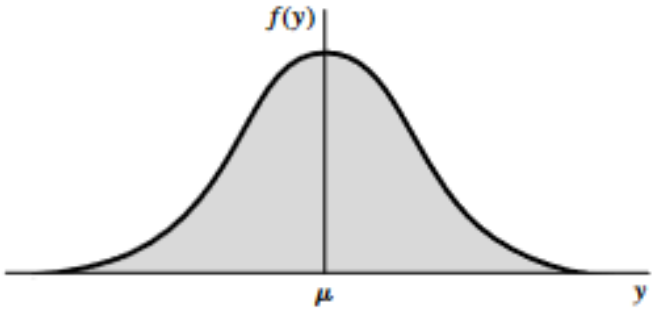
```
function smoothData(x,y)
  s = []
  for i = 1.length(x)-2
    s = (y(i) + y(i+1) + y(i+2))/3
  endfor

  x(end) = [];
  y(end) = [];
  disp(s)
  subplot(2,2,3)
  plot(x,s,'b')
  title('Smoothed Linear Graph: y=9x+1/2')
  xlabel('x');
  ylabel('y');
```

Piece 4

Normal, Beta, and Gamma Distributions Report

The first section in chapter 4 that is covered would be section 4.5. These sections go over the normal, beta, and gamma distributions. Prior to this chapter, I was not the best at double integrals. However, after I read about this chapter, I was able to understand more regarding integration as well as the several distributions. This section goes over the Normal Probability Distribution. This most widely used continuous probability distribution has a bell-shaped curve or an upside-down parabola. The normal probability distribution is a random variable Y and is said to have to a *normal probability distribution* if and only if, a > 0 and -∞, the density function of Y is:

$$f(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-\mu)^2/(2\sigma^2)}, \qquad -\infty < y < \infty.$$

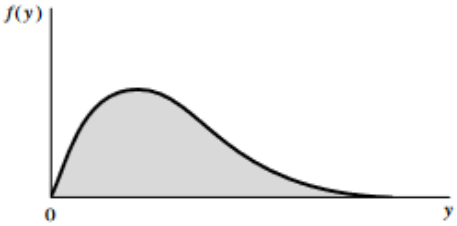


Normal Distribution is one of the most used distributions within the entire textbook. On the top of the curve is where the mean, the median, and the mode are located. Since we are looking at the normal probability distribution, all three of them are in the same spot.

$$\int_a^b \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-\mu)^2/(2\sigma^2)}\, dy.$$

To find the normal distribution, you would need to solve the areas under the function using integration. For the areas that are under the distribution that correspond to P(a <= Y <=b) would require the evaluation for the following integral:

The next distribution is the Gamma Distribution which is when random variables are always nonnegative. Therefore, the yield distributions of the data are skewed to the right so most of the density of the function would be near the origin on the graph. The Gamma Probability Distribution and the Normal Probability Distribution are very different due to the graphs. The Gamma Distribution contains two parameters. The first parameter is the alpha parameter which can also be named the scale parameter.



A random variable Y is said to have a *gamma distribution with parameters* $\alpha > 0$ and $\beta > 0$ if and only if the density function of Y is

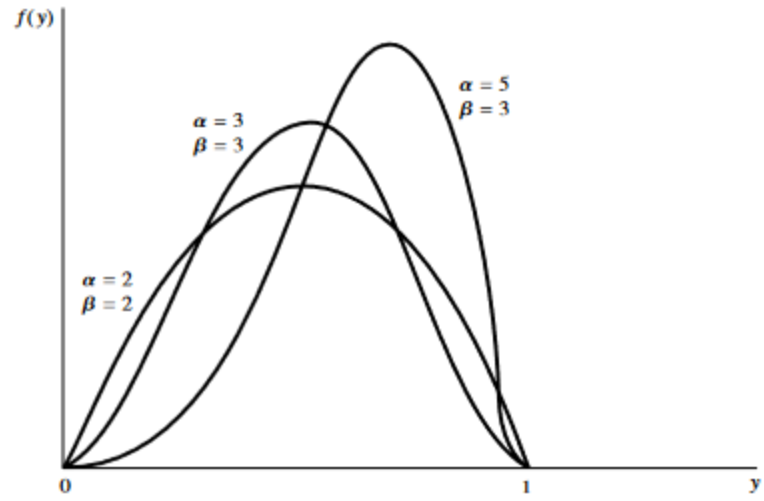
$$f(y) = \begin{cases} \dfrac{y^{\alpha-1}e^{-y/\beta}}{\beta^{\alpha}\Gamma(\alpha)}, & 0 \le y < \infty, \\ 0, & \text{elsewhere,} \end{cases}$$

where

$$\Gamma(\alpha) = \int_0^{\infty} y^{\alpha-1}e^{-y}\, dy.$$

The last distribution in the chapter in the Beta Probability Distribution. This distribution is a density function that has two parameters over the closed interval of zero and one. Beta Probability Distribution unlike the Gamma Probability Distribution and the Normal Probability Distribution, has different shapes of how the Beta Probability Distribution graph looks due to the two parameters. The highest point of the graph could be like the Gamma Probability Distribution where it is in the middle of the distribution, or it could be like the Normal Probability Distribution. The apex can also be all the way to the right unlike any of the other distributions that we have gone through. Similarly, to the Gamma Probability Distribution, the names of the parameters are also alpha and beta.

A random variable Y is said to have a *beta probability distribution with parameters* $\alpha > 0$ *and* $\beta > 0$ if and only if the density function of Y is



Overall, I found that double integrals are difficult to do and figure out the correct bounds for. At times I found that double integrals would make me sit there and ponder about what the bounds could be if it wasn't obvious in the question. However, chapter 4 allowed me to practice my integrals a lot more and I feel pretty confident with solving them. This is one of the five chapters that we covered that I struggled with the most in my opinion. I tried my hardest to understand that proofs and the solutions to the examples in the class however, I would sometimes get lost and not understand why we would get a certain bound in the integrals.

Piece 5
Java Poker Project

This java project was created by making the Card object first. My logic was that in order to have a Card, the card would be a suit and a rank. Therefore, to make a Card, the parameters that it requires is Card(suit, rank). After the Card was object was created, I had to create a deck that contained 52 cards. I knew that the deck had to be 52 cards and that each card would have a rank and a suit. I created a method named generateCards() that would take the Card and give it two characters ("HDSC" and "23456789TTQKA"). This would allow me to have the deck filled with Card objects that contained the rank and the suit.

```
public Card(char inputRank, char inputSuit) {
    rank = inputRank;
    suit = inputSuit;

}
public Deck() {
    deck = new ArrayList<>();
    generateCards();
}

public void generateCards() {
    ArrayList<Card> cards = new ArrayList<Card>();
    String temp = "HDSC";
    String temp2 = "23456789TJQKA";

    for (int j = 0; j < 4; j++)
        for (int i = 0; i < 13; i++)
            cards.add(new Card(temp2.charAt(i), temp.charAt(j)));

    deck = cards;
}
```

Next, I went ahead and created a method to draw(). This method would take a card from the deck, remove it from the deck and put it into my hand. Once this was working, I was able to create the drawHand() method that would call the draw method five times and put the cards into an ArrayList named hand. An idea that I had that would make my life a lot easier was to sort the ArrayList hand prior to checking the hand. This is because if I have a straight for example, then I know that the second card in my hand would be the first card plus 1. Another example is that if I have a royal flush and the hand is sorted, I know that the 3rd card in my hand (otherwise known as the middle card) would have to be a Queen.

```
public Card draw() {
    if(deck.isEmpty()) {
        return null;
    }

    Random random = new Random();
    int random1 = random.nextInt(deck.size());

    return deck.remove(random1);
}

public ArrayList<Card> drawHand() {
    ArrayList<Card> hand = new ArrayList<>();
    for(int i = 0; i < 5; i++) {
        hand.add(draw());
    }
    return hand;
}
```

However, the issue was that if I had a card and its rank was 2 and another card with the rank of Q, I couldn't compare an integer to a character that way. Therefore, inside of the Card class, I had the Card rank and suit change from 'A' to 14, 'K' to 13, and so forth. Since the Card ranks and the suits were now able to be compared to each other, I was able to start creating methods like isPair(). IsPair will sort the hand prior to checking anything. Then, I check if the first two cards are the same and the third and fourth cards. If they are the same, then it is a twoPair instead of one pair. If the first two cards are not the same, it would check the 3rd and 4th card, and then the fourth and the fifth card.

```
public int checkRank() {
    if(2 <= Character.getNumericValue(rank) && Character.getNumericValue(rank) <= 9) {
        return Character.getNumericValue(rank);
    }
    int newRank = -1;
    if(rank == 'T') {
        newRank = 10;
    }
    if(rank == 'J') {
        newRank = 11;
    }
    if(rank == 'Q') {
        newRank = 12;
    }
    if(rank == 'K') {
        newRank = 13;
    }
    if(rank == 'A') {
        newRank = 14;
    }
    return newRank;
}

public int checkSuit() {

    int newSuit = -1; //HDSC
    if(suit == 'H') {
        newSuit = 1;
    }
    if(suit == 'D') {
        newSuit = 2;
    }
    if(suit == 'S') {
        newSuit = 3;
    }
    if(suit == 'C') {
        newSuit = 4;
    }

    return newSuit;
}
```

Then, I would check twoPair() because the logic of it was similar to isPair(). I would check if the first two cards are the same and if the third and fourth cards are the same. If the first two cards are not the same, I would check the second card with the third card and then the fourth card with the fifth card. However, using Collections.frequency() saved my life because I was able to check how many times that card is in the hand. Therefore, I knew that if the first card and

the second card were a pair but, I was receiving a frequency of 3 then I knew that I had three of a kind rather than twoPair.

```
public boolean isPair(ArrayList<Card> newList) {
    ArrayList<Integer> tempList = sorted(newList);
    if (tempList.get(0) == tempList.get(1)) {
        if (tempList.get(2) == tempList.get(3)) {
            return false;
        }
        if (tempList.get(2) == tempList.get(4)) {
            return false;
        }
        if (tempList.get(3) == tempList.get(4)) {
            return false;
        }
        if (Collections.frequency(tempList, tempList.get(0)) == 2) {
            return true;
        }
        return false;
    }

    if (tempList.get(1) == tempList.get(2)) {
        if (tempList.get(3) == tempList.get(4)) {
            return false;
        }
        if (Collections.frequency(tempList, tempList.get(1)) == 2) {
            return true;
        }
        return false;
    }

    if (tempList.get(2) == tempList.get(3)) {
        if (Collections.frequency(tempList, tempList.get(2)) == 2) {
            return true;
        }
        return false;
    }

    if (tempList.get(3) == tempList.get(4)) {
        return true;
    }

    return false;
}
```

The isThree() method would check to see if the first three cards are equal to each other. If they are, then it would check if the fourth and fifth card are a pair or not. If they are, then it would be three of a kind. However, if it is not the first three cards, we could check the middle three cards out of the five in the hand. If they are equal and had a frequency of three, then we have a three of a king. Lastly, I would check the last three cards to see if they are equal to each other which in that case we would have a straight as well.

The isFour() method would work in a similar fashion. If the first card is equal to the fourth card, then it is a four of a kind as long as the fifth card is not equal as well. I did this by checking the frequency of the first card equal to 4. If it was anything else, it was not a four of a kind. I would then check the fifth card for a frequency of four that would determine whether or not it was a straight.

In order to check a straight, I created the straight() method that would check if the hand is in order. Therefore, I would check to see if the fifth card is a 14 (Ace). If it is 14, then I would check to see if the first card is a two. Then I would check if the second card is the first card plus one. If that holds true for the rest of the hand, then it is a straight. Otherwise, if the fifth card is not an ace, then I would check to see if the second card is the first card plus one all the way through the hand. If it holds true, then it is a straight.

A flush was very easy to create. I would call the sortSuit() method so that I could have all of the suits change to numerical values (1,2,3,4,5). If the suit in the hand has a Collections.frequency of 5, the it would be a flush. There is nothing else to check in this method other than the frequency of a single card in the hand. To check whether the hand is a full house, we would take bits from both isPair() and isThree(). We would check to see if the first three cards are equal and that the last two cards are equal however, we made sure that the third card is not equal to the fourth card. If that did not work, then we would check to see if the third, fourth, and fifth cards are equal to each other. If they are, then it is not a full house. Otherwise, if it is all true, it would be a full house.

To check if the hand is a straight flush, I would use both methods that I created prior to check. I would check the hand to see if it is a straight and then a flush. If both of those methods return true, then it is a straight flush. To check if the hand is a royal flush, you would check to see if the hand is not a straight flush. If it is a straight flush, you would check to see that the third card is a Queen. If it is, then it is a straight flush.

Overall, this is one of the hardest projects that I have done at Stockton. I believe that this took me a few weeks to finish however, I am very proud of the results that I received. I didn't know as much about Object oriented programming as I though and I was able to teach myself a few things that I didn't know about before such as Collections.frequency. I found it very difficult to compare the actual Card objects to each other. It took me a while until I figured out that I

could just change the characters to numerical values and then compare them that way. Once I was able to figure that out, the rest of the project was relatively easier that the initial set up. This is a project that I plan on working on a little more after this course so that I can include it on my resume. I don't think that I've ever worked so hard on a project other than this one.