Michael Chillemi

2/13/2022

Probability & Applied Stats

Professor Hoy

Explaining The Submissions

This paper is going to be explaining all the submissions pieces in project 1 bundle. In this bundle is all our programming assignments we went over in class. To start our first program was to start creating our very own statistics library. The first set of statistics tools we had to add to our library was functions that were able to compute the mean, median and mode of a list. To complete this task all functions needed to take in a list as an input. To complete this we were instructed to use arrays. For creating this function, I created a variable called “answer” and set this variable equal to zero. My game plan was to use this variable later in the to be returned as the final answer. After I did a basic if statement to verify that the list was populated to perform the calculation. If the list size was zero, I would just return 0. But if that was not the case, I created a for loop that went through the whole list and added every value to the variable “answer”. After the loop was finished, I took the answer variable and divided by the size of the list. The last step was to return my answer variable which would be the answer. Next was the calculate the median of a list. I took a similar approach to my mean method and created a variable that I was going to return in the end as my final answer. I also created an if statement to check the size of the list. After I created an if statement to check if the list size was even or odd. If the list was even, then the function would add the two-middle number in the list and divide by 2 which would be set to my answer variable. If the list was not even, then the else case would hit, and you would return the middle value of the list giving you the correct answer. Lastly, we had to create a method that would return the mode of the list. To complete this take, I repeated the same beginning steps but additionally created a variable count to keep track the number of times a number was repeated and then after traversing the list was complete it would return that number. Once we drove deeper into statics, we go introduced to more formulas that we needed to add to our library. These functions were the standard deviation and variance. To complete this we had to create a method that takes in a list. After we needed the mean of that list once we got that we needed to go through the list again and subtract every value by the mean value and square up each value. After that step we had to add up all the values and divide by the size of the list minus one. That value would be our variance so both methods are basically identical. The only difference was for standard deviation we had to take the variance and take the square root to achieve our answer. One of the next topics we learned about was permutations and combination probability. These methods were very simple and very easy to implement it was basically plug and play. Now I know I could have used the factorial method in the math library but as a little bit of extra credit I created my own factorial method to use in this case and it worked perfectly. The last methods we added to our stats library for the time being was methods that would return the union of two list, the intersection of two list and the compliment of a list. To complete these goals and make it easier we used array lists. Also, I decided to use lists of days of the week to make it easier to accomplish my goals. For my union method it took in an input of two array list. What I did was I created a new array list that I was going to return at the end as the result. After I made a for loop that went through the first list and added every value in the new list. Once that loop was broken, I created another for loop to go through the second list and all those values as well. But there is a twist before the values were added I created an if statement to check if the current value in the second list was not already in my new list. If it was not in the new list add it, if the value was in the list, then don’t do anything and continue going through the list. At the end return the new list to get your answer. The next method was not difficult to implement. All I did was create another list to return at the end and then I traversed through the first list. To help keep the method simple I utilized the contains method array lists have. I just created an if statement to check if the current element in list 1 was in list 2 and if it was then added it to the new list. If it was not in the second list, then do nothing and move on. After the loop was complete, I returned the new list. The last method was the compliment this method only took in one array list. To complete this I predefined an array list with all days of the week. Yet again I created a new list to return at the end. All I did to complete this function was like the previous methods. I loop through the list I created of the days of the week. If the current element was not in the list that was inputted, then add it to the new list and then when the loop was completed return the new list with the compliment values. After I just created other methods to populate array lists to use in my tester class to showcase my work.

On one of the fist days of class we learned how to create a histogram. We used data from our textbook. The task was not hard we just had to follow the steps the Professor provided for us. Later in the semester we leaned how to create a program that would write out text files from whatever the user inputted. The task we were given was to create a program that created a CSV file with one column of ID’s going from zero to one thousand. The next column was a random number from one to one thousand. This task was not difficult at all I did was took the example we did in class and modify it to be able to put those values into a CSV file. Once that step was completed, we need to replicate what we did one the first day and take those values and create a histogram.

The next program we had to create was recreating the monte hall game show problem. This program was not terrible either. First, I created some necessary variables that I would be able to implement later. This program has two methods keep Door and change Door. Keep door was simple all I did was use a random number generator to generate a random number from 0-2. After I made a loop to run the test ten thousand times. I just created one if statement if the contestant’s door was the same as the prize door, then add to the counter. After the loop was completed take the average and return that value. Change door was slightly difficult at first. So, I began the same as keep door and set the two values to a random number. Then created a variable called goat door which was going to be the door that the host essentially shows the contestant before they decide to change the door. To implement this, I crated a while loop to basically loop through and change the value of goat door until it was not the same as the contestant door or the prize door. To simulate the contestant changing the door all I did was three minus the goat door value plus the contestant door value to get the value of the switched door. To wrap up this method I did a simple if statement to check if switched door was the winning door and if it was added it to the counter and return the average just like keep door.

Finally, the last program we had to create was to simulate the probability that two people in the same room had the same birthday. For this program first I created a Person class, and every person had a random birthday which can be simulated as a random number from 1-365. Next a created a birthday probably class to do all the necessary calculations. This class only had 3 methods add people to class, compare birthdays, and share birthdays. The add person to class was made to take in an array of Person objects and add a new person objects to the array. Another method to compare the birthdays of everyone in the array. This was implemented by a basic double for loop. Once a match was hit increase my counter variable and break out of the loops. My last method was very simple. The inputs for this method were the size of the class and the number of runs. I created an array with the size from the input. Next, I made a for loop with the amount of runs that was one of the inputs. After, I called my add people to class method to populate it. Then my compare birthdays method to generate the number of times it hits. After the loop ended I simply calculate the average and returned the answer.