Mia Watts

**Stats Library Method Descriptions**

**Mean, Median, and Mode**

* Mean – This method accepts an array and calculates the mean based on the values given in the array. The sum of all values is calculated first through a for loop that iterates through all values of the array, and once the loop terminates, the method returns the sum divided by the length of the array which translates into the mean, or average, or expected value of the values in the array.
* Median – This method handles two cases of median calculations: the first is if the number of values in the parameter array is even, and the second is if it’s odd. The method accepts an array as a parameter and returns a double value, returning the value in the center of the array by dividing the length in two if the array is odd in number of values and returns the average of the two middle numbers if the array value count is even. The median checks for whether the array has an odd number of values or even number by doing the modulo operation. If the result of calculating the length mod 2 is 0, then there is an even number of values. Otherwise, the array must have an odd number. If the array is even, then the two middle numbers are found by assigning the first to the value returned when the length is cut in two and assigning the second to the value found by subtracting the halfLength variable by 1 (halfLength being the length of the array divided by 2).
* Mode – The mode method accepts a single parameter which is an array that will contain values of which we want to find the mode. The method uses TreeMaps because we need to keep track of how many times the value is seen to make sure we don’t return a mode when there isn’t one. Using a string return type, the method returns “none” if there is no mode, otherwise the method returns the mode. The method iterates through the items in the array, and if the TreeMap is null, the item is placed in the TreeMap. Otherwise, the maximum value based on the key in the map is obtained, setting the mode to whichever value has the maximum count.

**Classic Variance**

I calculated this a bit differently than usual (maybe), but it produces the correct result as compared to variance calculators and problems we tried in class. This method has a variable n that calculates the array size, the first side of the variance formula which is dividing 1 and n – 1, the second side, which will eventually be the summation, the mean, which is calculated from the previously programmed mean method, and the variance, which is the result that is returned at the end. The first part of the program is a for loop that calculates the sum of the array by adding up all the values in the array. The second part initializes the mean variable with the result of taking the mean of the array. The third part iterates through the array and adds the sum of the squares to the second side variable, and the fourth part computes and returns the variance by multiplying the sides together and returning the result.

**Classic Standard Deviation**

Taking an array as the parameter, this method uses the variance method to find the standard deviation. The first declaration for the standard deviation result variable sets it equal to 0, then sets it equal to the square root of the variance result which is what the standard deviation computes. After, the result is printed.

**mxn Rule**

This method uses the mxn rule calculation formula, taking two parameters and multiplying them together.

**Permutations**

This method computes the permutations of an event, simplified to n pick r, where the items are ordered. The BigInteger factorial method is used to divide the factorial of n and the factorial of n – r. The parameters are n and r, n being the total amount of items and r being how many we are choosing out of the n total amount of items.

**Combinations**

* BigInteger combinations – This version of the combinations method uses BigInteger to calculate the total combinations given a total number of values n and how many we are choosing r. In other words, combinations in general return n choose r, or how many ways we can choose r items from n total items. The method first calculates the numerator, which is the factorial of the total number of items n, then calculates the denominator, which is the factorial of r multiplied by the factorial of the number of total items minus the items we’re choosing.
* Long combinations – This version of the method returns the same result, just with longs instead of BigInteger values. Similar to the previous version, the method calculates the numerator and denominator, this time using the long factorial method instead of the BigInteger one.

**Factorial Method (BigInteger)**

This is the first factorial method programmed in the stats library. It uses the BigInteger classes, sets the result equal to ONE, which is the value of 1 in the BigInteger class, computes the factorial by using a loop to BigInteger multiply each number by the next number (i.e., 4\*3\*2\*1), and returns the result.

**Factorial Method (Longs)**

This is the second factorial method programmed in the stats library. Instead of BigInteger, the method uses longs, simply calculating the result by iterating through all values that would be multiplied when doing a factorial method (i.e., 3\*2\*1). The result is returned, and the original long result is set to 1 at the beginning since true factorial methods technically multiply by 1 twice.

**Conditional Probability**

There are two versions of the conditional probability method:

* A given B has occurred – This version of the method takes the probability of b occurring and the intersection of event a and b as parameters, returning 0 if there is an issue with dividing by 0 to avoid errors and returning the division between the intersection and event b if there is no issue. Returns a double value.
* B given A has occurred – This version of the method takes the probability of a occurring and the intersection of event a and b as parameters, returning 0 if there is an issue with dividing by 0 to avoid errors and returning the division between the intersection and event a if there is not issue. Returns a double value.

**Event Independence or Dependence**

Given the parameters, of a given b has occurred, b given a has occurred, the result of intersecting a and b, and the probabilities of event a and b, this method returns true if the values are independent and returns false otherwise. If any of the independence characteristics are true, those being that a given b equals a, b given a equals b, or the intersection of both equals the multiplied value of a and b, then the event is independent. Returns a boolean value.

**Multiplicative Law of Probability**

This method computes the Multiplicative Law of Probability given two parameters. The result is a multiplication of the numbers returned in double form.

* Method 1 – Returns a multiplied by b given a.
* Method 2 – Returns a multiplied by a given b.
* Method 3 – Returns a multiplied by b.

**General Addition Rule**

This method computes the Additive Law of Probability given three parameters for the first version in which the intersection must be subtracted because otherwise it’d be included in the result twice and the second version in which the intersection doesn’t need to be subtracted because it isn’t already included when adding a and b. Both return double values.

* Method 1 – Adds the probability of event a and the probability of event b together, subtracting the intersection of a and b afterward.
* Method 2 – Adds the probability of event a and the probability of event b.

**Odds of Something Not Occurring (Complement)**

Given the singular parameter of the double variable complement of event a, the method subtracts this value from 1, returning a double variable that represents the complement of the given value.

**Theorem of Total Probability**

This method returns a double variable, housing four total parameters: a given b, the probability of b occurring, the complement of b, and a given the complement of b. The method returns a double that is the result of multiplying a given b and b together and dividing the value by the added value of a given b multiplied by b and a given the complement of b multiplied by the complement of b.

**Bayes Theorem**

This method computes and returns the result of Bayes Theorem. The parameters include the probability of event a occurring, the probability of b occurring, and the probability of a occurring given b has already occurred. The result is the multiplication of a given b and b divided by a. The method returns a double variable.

**Combinatorial PMF**

Not sure if we needed this, but it was programmed anyway using the ball and bag example. The method returns a double variable, computing the combination of the number of red balls choosing y, which is the number of red balls we want, the number of black balls choosing the selected minus y, and the combination of the total amount of items choosing only the selected amount. The first and second combination results are multiplied together, and the result of this is divided by the third combination. This method returns the probability in double form and can be changed for problems similar to the ball and bag problem.

**Binomial Distribution PMF**

Accepting four parameters:

* p – success probability
* q – fail probability
* n – trials
* y – successes

the method returns the binomial distribution result by calculating the combination of n choose y multiplied by p raised to the number of successes and q raised to the number of trials minus the number of successes. Returns the result in double form.

**Expected, Variance, and Standard Deviation for Binomial Distribution**

* Expected – The value is returned as a double, the result being the mean of the binomial distribution event with the formula computing the multiplication of n and p (variables are the same from the above descriptions).
* Variance – The variance is calculated by returning the multiplication of n, p, and q.
* Standard deviation – This returns the square root of the variance in double form.

**Geometric Distribution PMF**

This method returns a double value and takes the following parameters:

* p – success probability
* q – fail probability
* y – number of trials

It calculates geometric distribution using the formula, multiplying p by q to the power of one less than the total trials.

**Expected, Variance, and Standard Deviation for Geometric Distribution**

* Expected – Returns a double result that is calculated by dividing 1 and p, taking p as a parameter (see above for what p is).
* Variance – Returns the variance in the form of a double, calculating it by dividing the complement of p and p to the power of 2.
* Standard deviation – This returns the square root of the variance in double form.

**Hypergeometric Distribution PMF**

This method computes the hypergeometric distribution based on the values given as parameters, which are as follows:

* r – available items of first type
* y – number of second type items chosen
* N – total items
* n – number of items picked

Using these parameters, all being integers, the result is calculated by taking the combination of r choose y, the combination of N – r choose n – y, and the combination of N choose n, multiplying the first two combination results and dividing that result by the third combination result. The method returns a long value since it uses the long combinations method.

**Expected, Variance, and Standard Deviation for Hypergeometric Distribution**

* Expected – Takes three double parameters, n, r, and N, and calculates the expected value by returning the multiplied value of n and r divided by N. See above method description for variable descriptions. Returns a double result.
* Variance – Uses n, r, and N as the parameters to calculate the variance by multiplying n, r divided by N, N minus r divided by N, and N minus n divided by N minus 1. See above description for variable descriptions. Returns a double result.
* Standard deviation – Takes the square root of the variance and returns it as a double value.

**Negative Binomial Distribution PMF**

This method accepts four parameters:

* p – success probability
* q – fail probability
* y – trials
* r – number of “wins”

Both p and q are doubles, and y and r are integers. The method takes these values and returns a double result that is calculated by computing the combination of one less trial choosing one less win and multiplying that by p to the power of the number of wins and q to the power of the trials minus the wins.

**Expected, Variance, and Standard Deviation for Negative Binomial Distribution**

* Expected – Returns the double result of dividing r and p given the parameters of r and p (see above for parameter descriptions).
* Variance – Returns the double result of multiplying r by the complement of p and dividing that result by p squared. See above for parameter descriptions.
* Standard deviation – Returns the square root of the variance value taking p and r as parameters.