**StatsLibrary User Manual**

A collection of Java based statistical functions.

Cachary Tolentino

CSCI-3327

**Table of Contents**

**Software Description3**

Detailed Description3.1

System Requirements3.2

**Installation Guide4**

**Class Overview6**

**Software Description**

A collection of Java based statistical functions.

**Detailed Description**

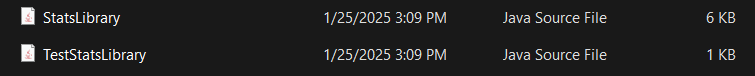
In the StatsLibrary class you can find a multitude of different statistical functions that can be used to calculate various formulas for a given set of inputs. You can find functions such as getMean, getMedian, and getMode which returns the mean, median, and mode. More functions can be found in the **Class Overview** section.

**System Requirements**

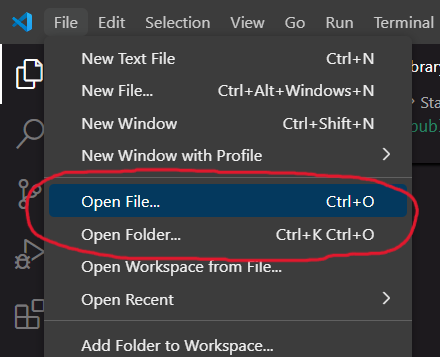
* A working device, primarily a desktop or laptop
* An IDE (ex: VSCode, Eclipse, etc…)
* Java JDK (Ver. 17 & up) & JRE (SE 17 & up)

**Installation Guide**

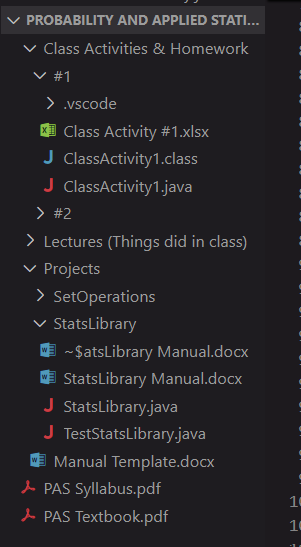
To begin using StatsLibrary, you will need to download two files. One is “StatsLibrary.java” and the other is “TestStatsLibrary.java” (optional).



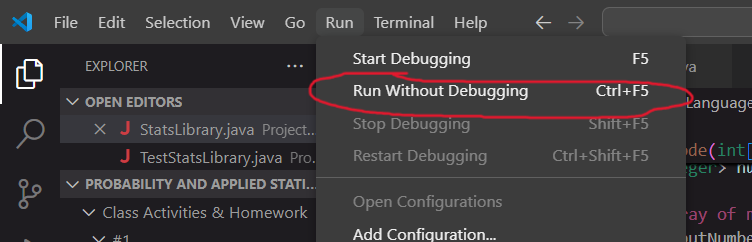
After downloading the files, simply move the files to the folder containing your project. Once done, you can open your preferred IDE (for this example we will be using VSCode). Then you can open the folder or the file itself within your IDE.



If you opened the folder containing the files then it should look similar to the image below.



If you only imported the StatsLibrary file then you can simply start using the class within your own personal project. Otherwise, if you also imported the TestStatsLibrary, then you can open that file and run it.



The result will be displayed on the console, unless there are graphical displays being run.

A screenshot of a computer

AI-generated content may be incorrect.

**Class Overview**

**getMean Function**

The getMean function is a function that takes in one **parameter** and returns a **double.** This function will return the mean of a given array of integers. One must pass an array of integers to use the function properly.

**getMedian Function**

The getMedian function is a function that takes in one **parameter** and returns a **double.** This function will return the median of a given array of integers. One must pass an array of integers to use the function properly. This function allows of odd or even amount of numbers within the array. This function is also implemented using bubbleSort as its sorting algorithm.

**bubbleSort Function**

The bubbleSort function is a function that takes no **parameters** and does not return any value**.** This function will sort a given array of integers into increasing order.

**getMode Function**

The getMode function is a function that takes in one **parameter** and returns a **Set of Integers.** This function will return the mode(s) of a given array of integers. One must pass an array of integers to use the function properly. This function uses a HashMap to find the mode.

**getVariance Function**

The getVariance function is a function that takes in one **parameter** and returns a **double.** This function will return the variance of a given array of integers. One must pass an array of integers to use the function properly.

**getStandardDeviation Function**

The getStandardDeviation function is a function that takes in one **parameter** and returns a **double.** This function will return the sample standard deviation of a given array of integers. One must pass an array of integers to use the function properly.

**Union Function**

The Union Function requires two **ArrayList<String> parameters** and it returns an **ArrayList<String>** value. Each parameter could be a subset of a set. The return value contains the union values of the given pair of parameters. This function only works for String type ArrayLists.

**Intersection Function**

The Intersect Function requires two **ArrayList<String> parameters** and it returns an **ArrayList<String>** value. Each parameter could be a subset of a set.The return value contains the union values of the given pair of parameters. This function only works for String type ArrayLists.

**Complement Function**

The Complement Function requires two **ArrayList<String> parameters** and it returns an **ArrayList<String>** value. The first parameter must be the set of all possible values. The return value contains the union values of the given pair of parameters. This function only works for String type ArrayLists.

**checkAxiomOne Function**

The function checks whether the given Event A (an array list of string) fulfills the Probability Axiom One which is

It first checks if the events in A appears in the space S, if so then it checks the given probabilities if they are greater than 0. It returns the value as a Boolean.

**checkAxiomTwo Function**

The function checks whether the given space S (an array list of string) fulfills the Probability Axiom Two which is

It checks if the total probability of all elements in S add up to 1. It returns the value as a double.

**checkAxiomThree Function**

The function checks whether the given event A (an array list of string) fulfills the Probability Axiom Three which is

It first checks if the events in A appears in the space S, if so then it adds up all the probabilities in event A. It returns the value as a double.

**probMutualExclusive Function**

The probMutualExclusive function takes a string array A, B, double value probA and probB. It will check if the events A and B are mutually exclusive if so then it will return the sum of their probabilities.

**combination Function**

The combination function calculates the combinatorial value of a given pair of integer values using the formula:

It uses the factorial function to find the factorial value. It will then return the value as a BigInteger value.

**permutation Function**

The permutation function calculates the permutation value of a given pair of integer values using the formula:

It uses the factorial function to find the factorial value. It will then return the value as a BigInteger value.

**factorial Function**

The factorial function calculates the factorial value of a given integer value. It takes in an int value as the number to find the factorial value. It returns a BigInteger value of the factorial. This is primarily used as a helper function for combination and permutation functions.

**getMNRule Function**

The getMNRule function calculates the mn value of a given pair of integer values using the formula:

**getMultinomialCoefficient Function**

The getMultinomialCoefficient function calculates the paritioning value of a given integer and array of integer values using the formula:

It uses the factorial function to find the factorial value. It will then return the value as a BigInteger value.

**conditionalProbabilityAB Function**

The function will calculate the conditional probability for the events A and B.

**conditionalProbabilityBA Function**

The function will calculate the conditional probability for the events B and A.

**checkIfIndependent Function**

This function calculates whether the given probabilities of A, B, and A and B are dependent or independent. They are dependent if they fulfill one of the following,

Otherwise, it is dependent. It simply returns the calculated value as a Boolean value whether it is independent or not.

**checkIfDependent Function**

The function simply calls on the checkIfIndependent function, the value return will be the opposite result. It simply returns the Boolean value whether it is dependent or not.

**calculateIndependentorDependentIntersection Function**

The function will calculate the intersection of the given events A and B, depending on whether they are dependent or independent events. The calculation for dependent events are,

But if A and B are independent,

It simply returns the calculated value as a double.

**calculateExclusiveOrNotExclusiveunion Function**

The function will calculate the union of the given events A and B, depending on whether they are exclusive or not. The calculation for non-exclusive events are,

But if A and B are mutually exclusive,

Then,

It simply returns the calculated value as a double.

**partitionOf** **Space Function**

The function will check of the given sets in setsOfValue (arraylist of arraylist of strings) are part of the space when unioned together and if there are no shared values when intersected with one another.

**theoremOfTotalProbability Function**

The function will calculate the total probability by adding all the probability of the given A given B values and the probabilities of B.

**bayesTheorem Function**

The function will calculate the probability using Baye’s theorem with the given probabilities (probA and probB).

**expectedRandom Function**

The function will calculate the expected value for a random variable.

**varianceRandom Function**

The function will calculate the variance value for a random variable.

**standardDeviationRandom Function**

The function will calculate the standard deviation value for a random variable.

**binomialDistribution Function**

The function will calculate the probability for a binomial distribution.

**expectedBinomial Function**

The function will calculate the expected value for a binomial distribution.

**varianceBinomial Function**

The function will calculate the variance value for a binomial distribution.

**standardDeviationBinomial Function**

The function will calculate the standard deviation value for a binomial distribution.

**geometricDistribution Function**

The function will calculate the probability with geometric distribution.

**geometricExpected Function**

The function will calculate the expected value for a geometric distribution.

**geometricVariance Function**

The function will calculate the variance value for a geometric distribution.

**geometricStandardDeviation Function**

The function will calculate the standard deviation value for a geometric distribution.

**onBeforeGeometric Function**

The function will calculate the geometric probability distribution for a success that occurs on or before the nth trial.

**beforeGeometric Function**

The function will calculate the geometric probability distribution for a success that occurs before the nth trial.

**onAfterGeometric Function**

The function will calculate the geometric probability distribution for a success that occurs on or after the nth trial.

**afterGeometric Function**

The function will calculate the geometric probability distribution for a success that occurs after the nth trial.

**hyperGeometricDistribution Function**

The function will calculate the hyper geometric probability distribution.

**hyperExpected Function**

The function will calculate expected value for the hyper geometric probability distribution.

**hyperVariance Function**

The function will calculate variance value for the hyper geometric probability distribution.

**hyperStandardDeviation Function**

The function will calculate standard deviation value for the hyper geometric probability distribution.

**negativeBinomialDistribution Function**

The function will calculate the negative binomial probability distribution.

**expectedNegativeBinomial Function**

The function will calculate the expected value for the negative binomial probability distribution.

**varianceNegativeBinomial Function**

The function will calculate the variance value for the negative binomial probability distribution.

**standardDeviationNegativeBinomial Function**

The function will calculate the standard deviation value for the negative binomial probability distribution.