Natalie Belmonte

28 October 2024

StatsLibraryCode Methods and Testing

findMean(), findMedian(), and findMode()

Each of these methods is given a list of integers as a parameter, which they analyze to find the mean, median, and mode. These operations are well-known and simple to complete on paper with small sample sizes. However, their complexity increases as the dataset increases in size. When generalized as methods, the process to find these values can be used on integer lists with any number of elements.

Because there is no default median when a list is of an even length, findMedian() uses the mean method to find the value between the two middle numbers.

```
run:
A is a set of numbers: [1,2,2,2,3,3,4,5,6,7,8,9,10]
Mean value of A: 4.769230769230769
Median value of A: 4
Mode value of A: 2
```

Since mode selects a value from the list, it returns an integer whole number. Mean almost always returns a decimal, and median will return either depending on list length.

remove() and printArray() [set utility methods]

Abstracted out of union, intersection, and complement, these methods are useful for computing and displaying sets of numbers. remove() uses a for loop to overwrite removable elements, rearranging the remaining elements accordingly. printArray() uses print statements along with a loop to display each element in a set as a readable list of values.

```
run:
A is a set of numbers:
[1,2,2,2,3,3,4,5,6,7,8,9,10,]
After removing 4:
[1,2,2,2,3,3,5,6,7,8,9,10,0,]
```

union(), intersection(), and complement()

These set operation methods take two integer arrays as parameters for two sets. Union combines the two sets, eliminating duplicates, using several for loops and the remove method. Conversely, intersection finds and stores only the shared values. Complement returns each value that is found in the first set, but not the second. In this way, the second array given acts as a sample space for the first set.

This method was written referencing this website: https://stackoverflow.com/questions/51113134/union-or-intersection-of-java-sets

```
run:
A is a set of numbers: [1,2,2,2,3,3,4,5,6,7,8,9,10]
B is a set of numbers: [4,6,5,2,19,30,22,7]
Union of A and B:
[1,2,3,4,5,6,7,8,9,10,5,19,30,22,0,0,0,0,0,0,0,0,]
Intersection of A and B:
[2,2,2,4,5,6,7,0,0,0,0,0,0,0,]
Complement of B, with A as sample:
[1,2,3,3,0,0,0,0,0,0,0,0,0,0,0,]BUILD SUCCESSFUL (total time: 0 seconds)
```

By storing each value in a temporary array that then gets returned, the original arrays will always be left intact. This is useful when performing several operations using the same sets.

findVariance() and findStandardDev()

Using a for loop and Math functions, the mean of an input set is subtracted from each element and squared. The sum of these values is then divided by the length of the set minus one. A higher number indicates that the values are, in general, further from the mean, whereas a low number indicates that the values are closely related to the dataset's center. Because standard deviation is the square root of the variance, findStandardDev() takes the square root of findVariance()'s return value.

```
run:
A is a set of numbers: [1,2,2,2,3,3,4,5,6,7,8,9,10]
B is a set of numbers: [4,6,5,2,19,30,22,7]
Variance of A: 8.858974358974358
Standard deviation of A: 2.9764029228204905
```

Because variance and standard deviation receive their parameters independently, different sets can be handled by each simultaneously. The above shows both methods given the same input set (A).

conditionalProb()

The conditional probability of two sets is found using the probability values of the intersection of the sets and the probability value of one set on its own. The probability of a particular set can be determined by the number of elements it holds compared to the total number of elements available (in this case, all values in A+B). Though the intersection function has already been established, each probability calculation is done within this method manually.

```
run:
A is a set of numbers: [1,2,2,2,3,3,4,5,6,7,8,9,10]
B is a set of numbers: [4,6,5,2,19,30,22,7]
Conditional probability (A|B): 0.0
```

Ideally, (A|B) should return a value between 0 and 1. A result of 0.0 indicates an impossible event or, in this case, an error in the computation.

isIndependent()

To determine if sets are dependent or independent, three conditions must be considered: the value of (A|B), the value of (B|A), and the value of (A∩B). Similarly to how they would be found on paper, this method computes and tests the values of each condition with if-else statements. Because all three values must be true for independence, this method checks that each inequality is false, instead of proving equalities true.

```
run:
A is a set of numbers: [1,2,2,2,3,3,4,5,6,7,8,9,10]
B is a set of numbers: [4,6,5,2,19,30,22,7]
Are these sets independent? false
```

Based on this output, A and B are dependent sets.

bigFactorial() and longFactorial()

These methods are used to calculate the factorial of an integer, processed through the java classes BigInteger and Long respectively. They utilize identical for loops and temporary variables to calculate and return each factorial. Each starting number is initialized at 1, both to prevent divide by zero errors and to follow the factorial equation 0! = 1.

```
run:
Factorial of 6 by BigInteger: 720
Factorial of 6 by longs: 720
```

permutation() and combination()

Calling the longFactorial() method, these methods calculate the permutation and combination of two integer parameters, n and r. Since the above factorial methods are interchangeable, longFactorial() is used as the default throughout this program.

```
Permutation of n = 6 and r = 5: 720
Combination of n = 7 and r = 2: 21
```

Because the edge cases of n-r <= 0 are handled by the factorial functions, they do not cause errors for these methods.

binomialDist(), binomialExpected(), and binomialVariance()

To calculate the binomial distribution given four parameters, binomialDist() calls the combination() method and Java.lang's Math.pow() function. It two integers (n and y), as well as two doubles representing the probability of success/failure. The expected and variance formulas both use integer-double multiplication to calculate their results. Unlike the above methods, all three of these return doubles.

```
run:
Binomial distribution of n = 10, y = 4, p = 0.7, q = 0.3: 0.03675690899999998
Binomial expected of n = 10 and p = 0.7: 7.0
Binomial varaince of n = 10, p = 0.7, q = 0.3: 2.1
```

Binomial distribution returns a probability value, rounded automatically.

geometricDist(), geometricExpected(), and geometricVariance()

Geometric distribution handles the integer number of trials required for a success, as well as the probability of success and failure (as doubles). Both the expected and variance formulas only handle one double: the probability of success value. These methods utilize Math.pow(), double division, and double multiplication.

```
run:
Geometric distribution of y = 5, p = 0.6, q = 0.4: 0.015360000000000002
Geometric expected of p = 0.6: 1.6666666666666667
Geometric varaince of p = 0.6 and q = 0.4: 1.1111111111111112
```

Expected and variance both display repeated decimals that can be converted to fractional values (5/3 and 10/9).