Natalie Belmonte

30 April 2024

StatsLibrary Methods and Testing- Part 2

Imported Methods from StatsLibrary 1: longFactorial() and combination()

The overlap between the two portions of this program requires some methods to be carried over. The first, longFactorial(), is used to find the factorial of an integer using Java.Math's Long datatype. It is used to determine the factorial value of Y for the Poisson distribution. Secondly, combination() is used in almost every aspect of joint cumulative distributions. It uses the aforementioned factorial method to calculate the combination of two integers.

```
run:
The factorial of 6 is 720
The possible combinations of 6 choose 2 is 15
```

Lambda() and Poisson()

Lambda() is a simple method, used when finding the Poisson distribution as well as it's expected value and its variance. Leaving this division outside of poisson() allows a user to determine the lambda (λ) value without calculating poisson(). The Poisson distribution utilizes exponents and the irrational number *e*, both of which can be calculated using Java.Math.

```
run:
Poisson distribution: assume y = 2, k = 5, n = 4. Result: 0.183940
Using the same values, the expected and variance are both 1.25.
```

The Poisson value has been rounded using printf for readability.

Chebyshev()

Chebyshev's rule is used to determine the percentage of values that will be a certain distance from the mean. Though the formula differs depending on the value of each variable, this method is built to calculate the more common formula ($1 - \frac{1}{k^2}$). Because of the varying formatting of Chebyshev's rule problems, this method takes the mean, within value, and standard deviation as parameters instead of calculating it from scratch.

```
run:
Tchebysheff's theorem: assume mean = 151, standard deviation = 14, and both values are within 28 of the mean.
So, 75.0% of values are 2 units from the mean.
```

For this print statement only, the returned value has been multiplied by 100 in order to display properly as a percentage.

ContUniform(), contUniformExpected(), and contUniformVariance()

This method is limited in its usage, as it can only be utilized in situations when each given bound is a double instead of an equation. It is most useful in situations when hand-solving is complex or tedious because of the constants given. A similar situation is true when finding the expected and variance, which take two of the same bounds. While variance uses the Math.pow function, expected is a straightforward addition and division formula.

```
run:
Continuous uniform distribution: Assume that first event occurs within 0 and 40 seconds.
The probability of the event occuring in less than 15 seconds is 0.375
The expected value, on the interval (40,0): 20.0
The variance, on the interval (40,0): 133.333333
```

The variance has been rounded using printf for readability.

jointCumulativeDistribution()

Modeled after a problem with three quantities chosen from three dependent groups of items, this method takes six parameters: the total quantities, the first two amounts chosen, and the total amount of items chosen. Then, utilizing combination(), it follows a process similar to the hypergeometric distribution of independent variables to find the probability of a certain situation occurring.

```
run:
Joint cummulative distribution: Assume a = 2, b = 3, and c = 4.
The probability of chosing 1 from a, 1 from b, and 2 for c: 0.285714
```

The probability value has been rounded using printf for readability.

Which formulas are missing?

Because of how Java processes information, calculating integrals is more difficult than it is through programs meant for graphing, like Octave. Several statistical processes that handle functions contain integrals in their processes, such as gamma function distributions, marginal density functions, and multivariable conditional probability. Because of the increased complexity of integrating within Java, these distributions have been omitted from the StatsLibrary.