Semester Final – LaTeX Sorts

Daniel Bilotto
danielbilotto1@marist.edu

December 16, 2021

1 Main class

This is the one and only class that does the work. Let me run you through it; so we start off with a size that we want in this case 1000 (line 10), then we set the infection rate (line 12). Then we set up our main array called pool. We also set up some other variables that we will get to later.

So lines 24 - 33, is setting up the array with the population and infection rate, the way I decided to do that was to turn the first 2 percent into "true" which in my case is infected. And then shuffle the array.

Then at line 46 I split the main pool into a group of 8 by making a temp array called "test" and populate it with the 8 people in the main pool. At line 64 I see if the group of 8 has a infected person, if they do then I know i need at least 3 tests so I add 3 tests to the count. Then I split that group of 8 into two groups of 4 and then I test those 2 and see if they have a infected person and if they do I will add 4 to the test count. If there is no infected person in the group of 8 then I will instead add one test to the count as seen in line 121. Also in this chunk of code I will see how many tests are done based on what case it is as well.

The only thing left is the output which prints out the sample that was given as the expected and then the actual with my data.

```
1
2 import java.util.ArrayList;
3 import java.util.Arrays;
4 import java.util.List;
5 import java.util.Random;
6
7 public class Main {
8
```

```
9   public static void main(String[] args) {
10    int size = 1000;
11    double theSize = size;
12    double infectSize = size * .02;
13    boolean[] pool = new boolean[size];
14    int testNums = 0;
15    int count = 0;
16    double caseOne = 0;
17    double caseTwo = 0;
18    double caseThree = 0;
19    double onePer = 0;
20    double twoPer = 0;
21    double threePer = 0;
22
23
24    for (int i = 0; i < size; i++)
25    {
26      pool[i] = false;
27    }
28
29
30    for (int k = 0; k < infectSize; k++)
31    {
32      pool[k] = true;
33    }
34
35    shuffle(pool);
36
37    /*
38    for (int j = 0; j < pool.length; j++)
39    {
40      System.out.println(pool[j]);
41    }
42    */
43
44
45
46    for (int e = 0; e < size/8; e++)
47    {
48      Boolean[] test = new Boolean[8];
49
50
51      for (int g = 0; g < 8; g++)
52      {
53
54        //System.out.println(count);
55        test[g] = pool[count];
56        count++;
57        //System.out.println(test[g]);
58      }
59
60      //System.out.println("hi");
61
62      List<Boolean> boolTest = new ArrayList<Boolean>(Arrays.asList
    (test));
63
64      if (boolTest.contains(true))
```

```
65        {
66          int pop = 0;
67          Boolean[] one = new Boolean[4];
68          Boolean[] two = new Boolean[4];
69          for (int t = 0; t < 3; t++)
70            testNums++;
71
72
73          for (int b = 0; b < 4; b++)
74          {
75            one[b] = test[pop];
76            two[b] = test[pop + 4];
77
78            //System.out.println(Arrays.toString(one));
79            //System.out.println(Arrays.toString(two));
80            //System.out.println("hi2");
81            pop++;
82
83
84          }
85
86          List<Boolean> testOne = new ArrayList<Boolean>(Arrays.
        asList(one));
87          List<Boolean> testTwo = new ArrayList<Boolean>(Arrays.
        asList(two));
88
89          if (testOne.contains(true))
90          {
91            for (int o = 0; o < 4; o++)
92            {
93              testNums++;
94              caseTwo++;
95              //System.out.println("hi2");
96            }
97
98
99          }
100
101         if (testTwo.contains(true))
102         {
103           for (int t = 0; t < 4; t++)
104           {
105             testNums++;
106             caseTwo++;
107             //System.out.println("hi3");
108           }
109         }
110
111         if (testOne.contains(true) && testTwo.contains(true))
112         {
113           for (int t = 0; t < 4; t++)
114           {
115             caseThree++;
116           }
117         }
118
119       }//if
```

```
120
121       else
122       {
123         testNums++;
124         caseOne++;
125         //System.out.println("hi4");
126
127       }
128
129
130
131     }
132     onePer = caseOne/theSize;
133     twoPer = caseTwo/theSize;
134     threePer = caseThree/theSize;
135
136     System.out.println("Expected:");
137     System.out.println("Case 1: 125 x 0.8500 = 106.25 instances
        requiring 107 tests (since there are no partial tests)");
138     System.out.println("Case 2: 125    0.1496 = 18.70 instances
        requiring 131 tests");
139     System.out.println("Case 3: 125    0.0004 = 0.05 round up to 1
        instance requiring 11 tests");
140     System.out.println("That's 249 tests to screen a population of
        1000 people for a disease with an infection rate of 2%.");
141     System.out.println("Actual:");
142     System.out.println("Case 1: " + size/8 + " x " + onePer + " = "
         + ((size/3) * (onePer)) + " instances requiring " + caseOne +
        " tests (since there are no partial tests)");
143     System.out.println("Case 2: " + size/8 + " x " + twoPer + " = "
         + ((size/3) * (twoPer)) + " instances requiring " + caseTwo +
        " tests");
144     System.out.println("Case 3: " + size/8 + " x " + threePer + " =
         " + ((size/3) * (threePer)) + " instances requiring " +
        caseThree + " tests");
145     System.out.println("That's " + testNums + " tests to screen a
        population of " + size + " people for a disease with an
        infection rate of 2%.");
146   }//main
147
148
149   static boolean[] shuffle(boolean[] array)
150   {
151     Random random = new Random();
152     for (int i = 0; i < array.length; i++)
153     {
154       int randInt = random.nextInt(array.length);
155       boolean temp = array[randInt];
156       array[randInt] = array[i];
157       array[i] = temp;
158     }
159
160     return array;
161   }//shuffle
162
163 }
```

2 Results

So let's talk results! For a size of 1000, my tests were right in line of the sample data. And with 10,000 they are similar to 1000 if you added a extra digit. As for100,000 to one million I can't really tell a difference in the data as the percentages are similar to the sample but maybe that's because I'm tired and need to look at it with new eyes.

After some research, while both hypergeometric distributions and binomial distributions tell the number of times an event happens in a fixed number of trials. I found that the difference between hypergeometric distributions and binomial distributions is that binomial distribution's percentage stays the same while hypergeometric distributions has different percentages. So I can conclude that this project was closer and probably was hypergeometric distribution.

Lastly as for things that I could of done better with this project to make it more efficient. I think my run time is a bit long. I'm using the java built in .contains to see if the pool of 8 has a infect person which since I didn't make it means I don't know how long it takes as well as I don't know if it's the most efficient. And the other thing I could do is maybe use a test method instead of making main do it all. That would make the code cleaner and just slightly more professional but I don't think it would make the program that much more efficent.