CMPT 435 - Fall 2021 - Dr. Labouseur

Semester Project

Nicholas Petrilli

December 15, 2021

1 Results

	Number of Tests Administered
1,000 People	255
10,000 People	2,560
100,000 People	25,686
1,000,000 People	256,206

Listed in the table above is an average of 10 runs for each population size. For a disease with an infection rate of 2 percent these are values that we expect when running the simulation. The slight difference in results when running the simulation for larger population sizes is a result of the distribution method used in this simulation. This simulation models binomial distribution, or selection without replacement which results in the same probability for each trial. When randomly choosing which people in the population get infected, it would be better to remove the infected people to eliminate the possibility of them being infected again. This is the subtle difference between binomial distribution and hypergeometric distribution. Hypergeometric distrubution uses selection with replacement, so when one person gets infected they are already accounted for and the probability slightly increases for the others in the population to be infected. In other words, in binomial distribution the events are indepedent of eachother, whereas in hypergeometric distribution the probability is dependent on the previous draw. This difference in distributions would make the simulation more like real life, which is the overall goal. Another way to improve the simulation would be to account for the fact that the tests aren't perfect and may result in false positives, or even worse false negatives. Also instead of hardcoding the infection rate and group size, letting the user input both of those values will give this simulation more use for different diseases.

2 PooledTesting Class

Listing 1: Pooled Testing with Main Method

```
1
2  import java.util.*;
3
4  public class PooledTesting {
5
6      public static int infectionRate = 2; // represents 2% infection rate
7      public static int groupSize = 8;
8      public static int numTests = 0;
9
10
11     public static void main(String[] args) {
12
13         System.out.println("Welcome to my Pooled Testing Simulation! In this
14         System.out.println("Enter a population size that you want to test on.
15         System.out.print("Population size: ");
16
17         Scanner input = new Scanner(System.in);
18         int populationSize = input.nextInt();
19
20         List<Person> peopleList = new ArrayList<>(populationSize);
21         ListPeople listPeople = new ListPeople(peopleList);
22
23         listPeople.addPeople(populationSize);
24         input.close();
25
26         System.out.println("\n--Running testing simulation for " + population
27
28         infect(peopleList, listPeople);
29
30         test(peopleList, listPeople);
31
32         System.out.println("\nNumber of tests needed for " + populationSize +
33
34
35
36
37
38
39
40     }
41     public static void infect(List<Person> peopleList, ListPeople listPeople)
42         //infect population with disease with 2% infection rate
43         listPeople.giveDisease(infectionRate);
44         int infectionCount = 0;
```

```
45          for (int i = 0; i < listPeople.size(); i++) {
46              if (peopleList.get(i).getIsSick() == 1) {
47                  infectionCount++;
48                  //printing out for greater than 10,000 is too many
49                  if (peopleList.size() <= 10000) {
50                      System.out.println("Person " + i + " has been infected");
51                  }//if
52
53              }//if
54          }//for
55          System.out.println("The total number of people infected for populatio
56      }
57
58      public static void test(List<Person> peopleList, ListPeople listPeople) {
59          /*
60          testing group size at a time (8) so split up list into groups of 8
61          if infection is found
62              split into two lists
63              if one group shows infection and other does not
64                  everyone in infect group tested individually, other group cle
65              else both groups show infection
66                  test all members of both groups
67          */
68
69
70          List<List<Person>> listOfLists = splitInGroups(listPeople.getList(),
71          //iterate through each list of 8
72          for (List<Person> list: listOfLists) {
73              numTests++;
74              //then iterate through each person in list
75              for (Person person: list) {
76                  //test if anyone of the 8 are sick
77                  if (person.getIsSick() == 1) {
78                      //need to split this list into two groups of 4 here
79                      List<List<Person>> splitList = new ArrayList<>();
80                      splitList = splitInTwo(list);
81                      //iterate through new list of lists with the new groups o
82                      //increment test
83                      for (int i = 0; i < splitList.size(); i++) {
84                          numTests++;
85                      }
86                      //now need to iterate through the two lists of 4
87                      for (List<Person> splitGroup: splitList) {
88                          //iterate through the individual people in the lists
89                          for (Person personInSplitGroup: splitGroup) {
90                              //if anyone is sick, need to test every person in
```

```
91                                  if (personInSplitGroup.getIsSick() == 1) {
92                                      for (int j = 0; j < splitGroup.size(); j++) {
93                                          numTests++;
94                                      }//for
95                                  }//if
96                              }//for
97                          }//for
98
99                      }//if
100                 }//for
101             }//for
102         }
103
104         //splits original list up into groups of 8
105         //returns a list of lists (of 8 each)
106         public static <T> List<List<T>> splitInGroups(List<T> list, int groupSize
107             List<List<T>> listOfLists = new ArrayList<List<T>>();
108             for (int i = 0; i < list.size(); i += groupSize) {
109                 //adds sublist of original list from i to groupSize
110                 //i + groupSize exceeds the list size for the last list, so the m
111                 listOfLists.add(new ArrayList<T>(list.subList(i, Math.min(list.siz
112             }
113             return listOfLists;
114         }
115
116         //split list of 8 into two lists of 4
117         //also returns list of list (the two lists of 4)
118         public static <T> List<List<T>> splitInTwo(List<T> list) {
119             List<List<T>> listsOfFour = new ArrayList<List<T>>();
120
121             int size = list.size();
122
123             List<T> first = new ArrayList<>(list.subList(0, (size + 1) / 2));
124             List<T> second = new ArrayList<>(list.subList((size + 1) / 2, size));
125
126             listsOfFour.add(first);
127             listsOfFour.add(second);
128
129             return listsOfFour;
130
131
132         }
133         public static void resetTests() {
134             numTests = 0;
135         }
```

3 ListPeople Class

Listing 2: ListPeople Class

```
1
2  import java.util.*;
3
4  public class ListPeople {
5
6      private List<Person> listPeople = new ArrayList<>();
7
8      //List of people constructor
9      public ListPeople(List<Person> listPeople) {
10         this.listPeople = listPeople;
11     }
12
13     //Adds specified number of people into list to represent population
14     public void addPeople(int numPeople) {
15         for (int i = 0; i < numPeople; i++) {
16             Person person = new Person(0);
17             listPeople.add(person);
18         }
19     }
20
21     //Infection rate is set to 2%, so this infects 2% of the population
22     //Generates random number between 0 and 100, if its less than 2 the perso
23     public void giveDisease(int infectionRate) {
24         for (int i = 0; i < listPeople.size(); i++) {
25             Random rand = new Random();
26             int percentSick = rand.nextInt(100);
27             if (percentSick < infectionRate) {
28                 listPeople.get(i).setIsSick(1);
29             }
30         }
31     }
32
33     public int size() {
34         return listPeople.size();
35     }
36
37     public List<Person> getList() {
38         return listPeople;
39     }
40
41 }
```

4 Person Class

Listing 3: Person Class

```
1
2  public class Person {
3
4      //0 represents not sick, 1 reprents sick
5      private int isSick = 0;
6
7      //construct person object
8      public Person(int isSick) {
9          this.isSick = isSick;
10     }
11
12     public void setIsSick(int isSick) {
13         this.isSick = isSick;
14     }
15     public int getIsSick() {
16         return this.isSick;
17     }
18
19 }
```