Zubin Mishra
UID: 604644805
CEE/MAE M20
May 16, 2017

Homework 6

1 The Shared Birthday Problem

1.1 Introduction

The goal in this problem is to develop a program that answers the question: how many people are required in a group before it is more likely than not that any two of the their birthdays occur in the same week? This is accomplished using MATLAB's rand function to create groups of random days to test. It will be discussed if the results seem reasonable and how they might change if a non-uniform distribution was used.

1.2 Model and Methods

The script first sets up the array to hold the 10,000 trials' results. It then enters a for loop to iterate through each trial with an embedded while loop. Before the while loop, the while loop's initial condition is set and the birthday array is set to empty (a reset for each trial). Within the while loop, a birthday is added to the birthday array using the MATLAB function rand:

```
r(length(r)+1) = ceil(365*rand);
```

The while loop then calls the function isMatch to check if there are two days that are within 7 days of each other. When the while loop is exited, the size of the birthday array is placed within the trial results array.

After all the trials are complete and the for loop is exited, the median group size is printed out and the trials are plotted on a histogram.

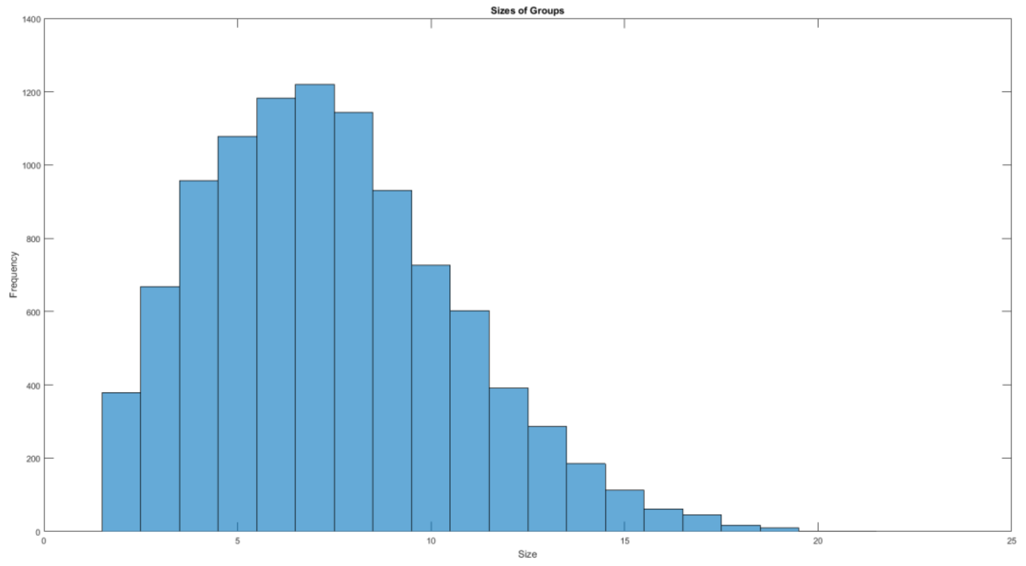
The function, isMatch, takes in an array of birthdays and returns true if there are at least two days that are within 7 days of each other, and false otherwise. It assumes that birthdays are given as integers from 1-365. It functions by first checking if the array is larger than 1. If it is, then a nested for loop is gone through to check if two birthdays are within 7 days of each other:

```
bool = false; %Assume there is no match at first
if length(r) > 1
    for i = 1:length(r)-1
        for j = i+1:length(r)
          if abs(r(i)-r(j)) < 7 || abs((r(i)-365) - r(j)) < 7 ||
             abs((r(j)-365) - r(i)) < 7
              bool = true;
          end
        end
    end
end
```

1.3 Results and Calculations

When the script is run, the following output is produced:

```
Median Number of People = 07
```



The median is consistently found to be 7, and the histogram consistently has the same shape.

1.4 Discussion

This number of 7 seems perfectly reasonable to me. It is easy to show mathematically why it is so. To calculate the probability of at least 2 people in 7 having a birthday within a week of one another, it is easier to calculate the complement and subtract it from 1. With 7 people, there are 21 different connections to be examined (6+5+4+…+1). For each connection, there is a $\frac{(365-13)}{365}$ chance that their birthdays do not fall within a week of one another. For 21 total connections, that means the probability of there not being any birthdays within a week from one another is $\left(\frac{(365-13)}{365}\right)^{21} = 46.7\%$. In other words, the chance to find a match among 7 people is 53.3%. This does assume that birthdays are independent, but it is a good enough approximation.

I would expect the answer to decrease if we accounted for the fact that not all birthdays are equally likely. If I were to think of the most extreme case, where the distribution of birthdays lies on a single day, then the chances of a match are 100% with just a group of two. Smaller deviations from uniform distribution will lead to smaller deviations from 7, but in the same direction as the most extreme case: fewer people would be needed to find a match.

2 Random Walker Collisions

2.1 Introduction

The goal in this problem is to develop a program that simulates two random walkers. Specifically, it is to be seen if a collision is more likely if both walkers are moving or if just one walker is moving. This is accomplished using the rand function to determine the action of each walker. It will be discussed which scenario results in collision sooner and if starting position has any effect on the result.

2.2 Model and Methods

The script begins by setting up the resultant move count array. The script then enters a for loop that will iterate through each of the 5000 trials using a nested while loop. Before the while loop, the while loop condition, initial walker positions, and the count are initialized. The while loop uses the moveRandom function to update the position of the two walkers, and if the positions of A and B match or if 1000 moves have passed, the while loop will end, and the move count will be entered into the resultant move count array.

The moveRandom function functions by finding a random number between 1 and 5 to determine what the walker will do:

```
    % Establish 2D array of directions to choose from
    dir = [[0, 0];[0, 1];[0, -1];[1, 0];[-1, 0]];

    % Get random direction
    dir_P = ceil(5*rand);
```

It then takes that entry from the 2D direction array and adds it to the input point. The result is then tested for validity. If valid, the new point is returned, and the walker performs its action. Otherwise the old point is returned, and it is as though the walker does nothing:

```
% Check if P will be in bounds after moving
    temp_P = P + dir(dir_P,:);
    if temp_P(1) >= -5 && temp_P(1) <= 5 && temp_P(2) >= -5 && temp_P(2) <= 5
        p = temp_P;
    else
        p = P;
    end
```

Once the for loop in the main script completes, the median number of moves is printed out.

2.3 Results and Calculations

With both walkers moving and the starting positions at A = (-5, 0) and B = (5, 0), the following output is produced:

`Median = 219` (consistently around 220)

With only walker A moving and the starting positions at A = (-5, 0) and B = (5, 0), the following output is produced:

`Median = 419` (consistently around 420)

With both walkers moving and the starting positions at A = (-4, 0) and B = (4, 0), the following output is produced:

`Median = 213` (consistently around 210-220)

With only walker A moving and the starting positions at A = (-4, 0) and B = (4, 0), the following output is produced:

`Median = 332` (consistently around 320-340)

With both walkers moving and the starting positions at A = (-3, 0) and B = (3, 0), the following output is produced:

`Median = 197` (consistently around 200)

With only walker A moving and the starting positions at A = (-3, 0) and B = (3, 0), the following output is produced:

`Median = 284` (consistently around 280-285)

2.4 Discussion

If two random walkers become separated, it is faster for both parties to search randomly for each other. This conclusion was shown not to be dependent on the initial conditions of both walkers; having one walker stay in place took more moves regardless of the initial distance between the walkers. These results conflict with the commonly given advice to stay put until help arrives.