Jason James

Nickname: OneByte

Matching Birthdays Project

## Birthday Matching Problem:

The given problem is to find the expected percent chance of having at least 3 people with the same birthday in a room of N people. To do this, I first create a variable that will store the amount of experiments that show at least 3 people with the same birthday, which will be called “containsThree”. Then for the Monte Carlo simulation events I will make a for-loop from one to a million. I will create an integer array, called “birthdays[]” which is of size 365, which will store the number of times an individual as a birthday on that given index of the array. For example, if birthdays[0] = 8, then that means there are 8 people within this simulation who share the first day of the year (January 1$^{st}$) as their birthday. In order to obtain the birthday data from all N people in the room, I will first need to generate a random day of the year (a number between 0 and 364) which will be used to access that index number of the birthday array. I will then add 1 to the total number of birthdays at the random generated day within the birthday array. Once all N people’s birthdays have been generated and stored in the birthdays array, I will need to find out if any of the days of the year, stored in the birthdays array, contain at least 3 birthdays. To do this I will iterate through all the elements in the birthday array (from 0 to 364) and if any of them are greater than or equal to 3, then I will increase my previously mentioned variable, named “containsThree”, by 1 and will break out of the iterations, because only one case is needed to prove that there is one day with 3 shared birthdays. This will conclude the logic for one simulation event. After all one-million simulations have been completed, I will return the solution in the form of the percentage of times a simulation returned with the result of containing at least 3 people, via “containsThree”, and divide it by one-million. To put it into percentage form, I will then simply multiply it by 100, which is the solution. For “Part B”, we must find the same thing, but for 4 people instead. To do this, all I have to do is replace my “3”’s with “4”’s.

For “Part A”, my output indicated that N=88 people are needed for at least 50% chance that at least 3 people have the same birthday. This is given by the percentage of 51.2126%. For “Part B”, my output indicated that N=187 people are needed for at least 50% chance that at least 4 people have the same birthday. This is given by the percentage of 50.2359%.

## Code (C++):

```
#include <iostream>
#include <cstdlib>

using namespace std;

const double BIG_NUMBER = 1000000.0; //Monte Carlo loop count

int main()
{
  for (int N = 10; N <= 200; N++)
  {
    int containsThree = 0; //int value to count how many external experiments have at least 3 people
    int containsFour = 0; //int value to count how many external experiments have at least 4 people
    for (int i = 1; i <= BIG_NUMBER; i++) //Externally test N a lot of times
    {
      int birthdays[365] = {0}; //birthdays[] stores the number of birthdays on a given day
      for (int j = 1; j <= N; j++) //Asking N people for their birthday
      {
        int randomDay = rand() % 365; //Will return random int between 0 and 364
        birthdays[randomDay] += 1; //Adds a count number for a person's birthday on the "randomDay" number day of the year
      }
      for (int x = 0; x <= 364; x++) //Iterating through all birthday counts to see if any of them have at least 3 people with that bday
      {
        if (birthdays[x] >= 3) //If a day of the year has at least 3 people with that day as their birthday counted
        {
          containsThree += 1; //Increment number of times an experiment has found 3 people with the same birthday
          break; //Break out of for-loop, because the experiment has already found a day with 3 people that share its birthday
        }
      }
      for (int y = 0; y <= 364; y++) //Iterating through all birthday counts to see if any of them have at least 4 people with that bday
      {
        if (birthdays[y] >= 4) //If a potential birthday has at least 3 people counted
        {
          containsFour += 1; //Increment number of times an experiment has found 3 people with the same bday
          break; //Break out of for-loop, because the experiment has already found a day with 3 people that share its birthday
        }
      }
    }
    double threePercentage = ((double) containsThree / BIG_NUMBER) * 100.0;
    //Percentage of events with at least 3 people who share a birthday
    double fourPercentage = ((double) containsFour / BIG_NUMBER) * 100.0;
    //Percentage of events with at least 4 people who share a birthday
    cout << "3 with " << N << " : " << threePercentage << "%" << endl;
    cout << "4 with " << N << " : " << fourPercentage << "%" << endl << endl;
  }
}
```

Output

| Pop Num | 3 Match | 4 Match |
|---|---|---|
| 10 | 0.0877 | 0.0001 |
| 11 | 0.117 | 0.0006 |
| 12 | 0.1575 | 0.0011 |
| 13 | 0.2071 | 0.0013 |
| 14 | 0.271 | 0.0026 |
| 15 | 0.3259 | 0.0023 |
| 16 | 0.4204 | 0.0044 |
| 17 | 0.5054 | 0.0036 |
| 18 | 0.6009 | 0.0052 |
| 19 | 0.7099 | 0.0083 |
| 20 | 0.8275 | 0.0095 |
| 21 | 0.9857 | 0.0141 |
| 22 | 1.1175 | 0.0149 |
| 23 | 1.2754 | 0.0172 |
| 24 | 1.4544 | 0.0176 |
| 25 | 1.6368 | 0.0247 |
| 26 | 1.8408 | 0.0298 |
| 27 | 2.078 | 0.035 |
| 28 | 2.3098 | 0.0408 |
| 29 | 2.57 | 0.046 |
| 30 | 2.8685 | 0.0509 |
| 31 | 3.1809 | 0.0587 |
| 32 | 3.4759 | 0.0721 |
| 33 | 3.7811 | 0.0772 |
| 34 | 4.1498 | 0.0905 |
| 35 | 4.541 | 0.0994 |
| 36 | 4.9111 | 0.1162 |
| 37 | 5.3386 | 0.1271 |
| 38 | 5.7639 | 0.1366 |
| 39 | 6.2409 | 0.154 |
| 40 | 6.7 | 0.1832 |
| 41 | 7.1945 | 0.1889 |
| 42 | 7.7021 | 0.212 |
| 43 | 8.2742 | 0.2348 |
| 44 | 8.823 | 0.252 |
| 45 | 9.4298 | 0.2721 |
| 46 | 10.0218 | 0.3086 |
| 47 | 10.6379 | 0.3359 |
| 48 | 11.303 | 0.3493 |
| 49 | 11.9729 | 0.3866 |
| 50 | 12.6351 | 0.4278 |
| 51 | 13.3797 | 0.4607 |
| 52 | 14.0737 | 0.4996 |
| 53 | 14.8788 | 0.535 |
| 54 | 15.6182 | 0.5755 |
| 55 | 16.3785 | 0.6196 |
| 56 | 17.21 | 0.6709 |
| 57 | 18.1277 | 0.718 |
| 58 | 18.979 | 0.7731 |
| 59 | 19.9635 | 0.8442 |
| 60 | 20.7877 | 0.8834 |
| 61 | 21.6916 | 0.9554 |
| 62 | 22.6132 | 0.9831 |
| 63 | 23.4745 | 1.0736 |
| 64 | 24.4864 | 1.1496 |
| 65 | 25.494 | 1.1964 |
| 66 | 26.4385 | 1.2874 |
| 67 | 27.5538 | 1.3719 |
| 68 | 28.6173 | 1.4413 |
| 69 | 29.6026 | 1.5458 |
| 70 | 30.6822 | 1.6269 |
| 71 | 31.621 | 1.7068 |
| 72 | 32.8132 | 1.8295 |
| 73 | 33.8946 | 1.9035 |
| 74 | 35.086 | 2.018 |
| 75 | 36.0944 | 2.1331 |
| 76 | 37.2428 | 2.2146 |
| 77 | 38.4458 | 2.3313 |
| 78 | 39.5735 | 2.4515 |
| 79 | 40.705 | 2.6157 |
| 80 | 41.7952 | 2.7307 |
| 81 | 43.0257 | 2.8533 |
| 82 | 44.1568 | 2.989 |
| 83 | 45.3004 | 3.1443 |
| 84 | 46.4629 | 3.2641 |
| 85 | 47.711 | 3.4168 |
| 86 | 48.7665 | 3.6191 |
| 87 | 49.9665 | 3.7507 |
| 88 | 51.2126 | 3.9578 |
| 89 | 52.3306 | 4.1333 |
| 90 | 53.394 | 4.275 |
| 91 | 54.5561 | 4.4408 |
| 92 | 55.7998 | 4.6208 |
| 93 | 56.759 | 4.815 |
| 94 | 58.0881 | 5.0663 |
| 95 | 59.139 | 5.253 |
| 96 | 60.266 | 5.4773 |
| 97 | 61.4204 | 5.6763 |
| 98 | 62.4181 | 5.924 |
| 99 | 63.5289 | 6.1366 |
| 100 | 64.663 | 6.3652 |
| 101 | 65.6276 | 6.5625 |
| 102 | 66.7544 | 6.8798 |
| 103 | 67.7566 | 7.0941 |
| 104 | 68.7447 | 7.377 |
| 105 | 69.8203 | 7.6132 |
| 106 | 70.7005 | 7.8548 |
| 107 | 71.693 | 8.1546 |
| 108 | 72.7277 | 8.4273 |
| 109 | 73.6654 | 8.8017 |
| 110 | 74.6132 | 9.0923 |
| 111 | 75.3643 | 9.3451 |
| 112 | 76.3351 | 9.6494 |
| 113 | 77.1898 | 9.9361 |
| 114 | 77.9881 | 10.2372 |
| 115 | 78.9339 | 10.5883 |
| 116 | 79.7161 | 10.9289 |
| 117 | 80.4611 | 11.3039 |
| 118 | 81.3504 | 11.6262 |
| 119 | 82.0407 | 12.0575 |
| 120 | 82.7698 | 12.3303 |
| 121 | 83.544 | 12.674 |
| 122 | 84.2492 | 13.0848 |
| 123 | 84.94 | 13.5634 |
| 124 | 85.575 | 13.9157 |
| 125 | 86.1941 | 14.3523 |
| 126 | 86.854 | 14.6916 |
| 127 | 87.4322 | 15.1259 |
| 128 | 88.0166 | 15.5893 |
| 129 | 88.4863 | 15.9707 |
| 130 | 89.0931 | 16.4025 |
| 131 | 89.6519 | 16.8742 |
| 132 | 90.1981 | 17.3067 |
| 133 | 90.6454 | 17.6931 |
| 134 | 91.0667 | 18.2051 |
| 135 | 91.5589 | 18.6354 |
| 136 | 91.9903 | 19.1245 |
| 137 | 92.4101 | 19.6208 |
| 138 | 92.8322 | 20.1456 |
| 139 | 93.2089 | 20.6469 |
| 140 | 93.6168 | 21.1528 |
| 141 | 93.9117 | 21.5885 |
| 142 | 94.2817 | 22.0937 |
| 143 | 94.5935 | 22.6667 |
| 144 | 94.9151 | 23.2609 |
| 145 | 95.1961 | 23.7185 |

| | | |
|---|---|---|
| 146 | 95.5128 | 24.238 |
| 147 | 95.7594 | 24.7826 |
| 148 | 96.0256 | 25.2856 |
| 149 | 96.2242 | 25.8662 |
| 150 | 96.4605 | 26.4973 |
| 151 | 96.7274 | 27.0548 |
| 152 | 96.9021 | 27.5879 |
| 153 | 97.0922 | 28.186 |
| 154 | 97.2999 | 28.827 |
| 155 | 97.4684 | 29.4325 |
| 156 | 97.6097 | 29.8952 |
| 157 | 97.8023 | 30.5592 |
| 158 | 97.943 | 31.239 |
| 159 | 98.0696 | 31.7903 |
| 160 | 98.218 | 32.3496 |
| 161 | 98.3409 | 33.0486 |
| 162 | 98.4549 | 33.6154 |
| 163 | 98.5607 | 34.201 |
| 164 | 98.6692 | 34.8739 |
| 165 | 98.7575 | 35.5089 |
| 166 | 98.8513 | 36.1861 |
| 167 | 98.9479 | 36.8075 |
| 168 | 99.0322 | 37.527 |
| 169 | 99.1023 | 38.1593 |
| 170 | 99.174 | 38.7485 |
| 171 | 99.2322 | 39.4911 |
| 172 | 99.297 | 40.1094 |
| 173 | 99.3415 | 40.7051 |
| 174 | 99.3964 | 41.4309 |
| 175 | 99.4418 | 42.1707 |
| 176 | 99.4898 | 42.7224 |
| 177 | 99.5364 | 43.4391 |
| 178 | 99.5763 | 44.0903 |
| 179 | 99.5987 | 44.8668 |
| 180 | 99.6548 | 45.6335 |
| 181 | 99.6747 | 46.0619 |
| 182 | 99.712 | 46.8112 |
| 183 | 99.7352 | 47.7213 |
| 184 | 99.7488 | 48.2059 |
| 185 | 99.7673 | 48.8729 |
| 186 | 99.8045 | 49.6114 |
| 187 | 99.8198 | 50.2359 |
| 188 | 99.8354 | 50.994 |
| 189 | 99.852 | 51.6196 |
| 190 | 99.8594 | 52.3799 |
| 191 | 99.8754 | 53.0587 |
| 192 | 99.8894 | 53.649 |
| 193 | 99.8949 | 54.3157 |
| 194 | 99.9068 | 55.1612 |
| 195 | 99.9199 | 55.7237 |
| 196 | 99.9242 | 56.3676 |
| 197 | 99.9312 | 57.1215 |
| 198 | 99.9394 | 57.805 |
| 199 | 99.9429 | 58.4648 |
| 200 | 99.9527 | 59.1354 |