Assignment 3

CSCI2244-Randomness and Computation
Due Friday, February 8, at 11:30PM

1 The birthday party never ends!

1.1 *Somebody* has a birthday today.

When I polled last year's class of 30 students for their birthdays, there turned out to be a student whose birthday was that very day (January 25), as well as two students in the group with the same birthday. (You guys are just way less fun.) What is the probability that in a group of 30 people, someone's birthday is today? In a group of k people? How big does k have to be to make this probability at least one-half? Be sure to describe your calculations carefully, specifying both the sample space and the underlying event. (HINT: As with the original problem, it is best to approach this by first computing the probability of the complementary event that *no one* in the group has a birthday today. For 30 students you can do the calculation exactly, but for an unspecified number k of students, it is best to use the exponential approximation. You should make the same simplifying assumption as earlier—that there are 365 possible birthdays, all equally likely.)

1.2 Births and deaths

Marian the Librarian maintains a carefully curated collection of biographical information about prominent people from the 19th century. She devises the following hashing scheme for indexing her collection: Every person will be given an 8-digit tag, consisting of the date of their birth (month and day, but not year) and the date of their death. For example, Nathaniel Hawthorne, born on July 4, 1804 and died on May 19, 1864, would have the ID tag 07040519.

Marian reasons as follows: I have about one thousand people in my collection. There are $365 \times 365 > 100,000$ different tags, which should be distributed roughly

uniformly, so the probability of two people with the same tag is something like $1/100$. So my scheme will most likely give a unique tag for each person.

What actually *is* the probability that two people in her collection will get the same tag?

Bonus no-extra-credit-has-nothing-to-do-with-the-course-content question. As you have probably guessed, Marian is way off-base. Just exactly how far off is for you to calculate. This of course means that her collection probably does have a collision. Find two prominent 19th century personalities with the same birthday and death day. (No idea how to do this efficiently—if you know some technically interesting Wikipedia-scraping way to solve the problem, I'd be interested in what it is.)

1.3 Real birthdays

As we discussed, our calculation of the probability of a shared birthday in a group of people was based on some idealized assumptions: First, we treated the problem as one of sampling with replacement, but that is not such a big deal (see the elections problem below). Second, there are actually 366 possible birthdays, but one of them (February 29) occurs much less frequently than the others. Third, we assumed that that the distribution of the 365 birthdays is uniform—but are births really uniformly distributed throughout the year?

To help you answer the question, I have posted data on US births in the years 2000-2003. (This was extracted from files posted on GitHub by FiveThirtyEight.com, who in turn got it from records of the Social Security Administration.) I modified the file format ever so slightly, and retained just these four years—the original data was for the 15 years from 2000 to 2014. Thus the data given here contains one leap year and three non-leap years, so February 29 is represented about the right amount.

This is a '.csv' file, which means that it is actually an ordinary text file, with each line consisting of several fields (5 in this case) separated by commas. If you double-click on it, it will probably open with Excel, and, indeed, you could do part (a) below just using Excel stuff. But you will need to open and read the file with Python to do part (b), so I've included in an appendix instructions about how to do this.

(a) Make two scatter plots or stem plots of the data, showing the proportion of the total number of births for each day of the year, both for the year 2000 alone, and for all four years combined. Note that there will be 366 x-values, and you will

need to be careful because February 29 only occurs during the first year. (As a reality check, the plot for 2000-2003 should show February 29 as an outlier, with many fewer births, but for 2000 it will appear as a 'normal' day.)

In both plots you can see the seasonal nonuniformity very clearly. In the plot for a single year, there is another source of non-uniformity, which I find absolutely astonishing and somewhat perplexing: It looks as if there are two or even three entirely separate data series, with roughly the same seasonal variation, but one significantly lower than the other.

(b) You are to estimate, for $k = 1$ to 65, the probability of a coincidental birthday in a group of k people, by performing a simulation based on the probability distribution you computed in (a) for all four years. You should then superimpose this on the plot that computes these probabilities under the assumption of a uniform distribution of 365 days. (The code for this is on the website.) Do you see much difference between the two plots? How adequately does the uniform probability distribution model the real-life version of this problem?

Another bonus no-extra-credit-has-nothing-to-do-with-the-course-content question. Explain the non-uniformities in the distribution of real birthdays, in particular the two mysterious low-probability series in the 2000 data?

2 Poker

As I showed you in class, there is a Wikipedia page 'Poker probabilities' with probabilities of all the 5-card poker hands, complete with the number of relevant outcomes for each hand expressed in terms of binomial coefficients. Please do take a look at it! An excellent exercise is to try to find the correct reasoning behind the results they give, and obtain the same answers.

But I needed to come up with problems whose answers you can't look up on the Web. So I invented some new poker hands, and here I ask you to determine their probabilities. This is the version of poker where we draw 5 cards from a shuffled deck. As in our examples in class, we model this problem with the sample space

$$\{X \subseteq \{1, \ldots, 52\} : |X| = 5\},$$

and assume each 5-element subset has the same probability.

Explain your reasoning carefully, and express your answers both as a formula using binomial coefficients and powers, and as numerical values. It's easy to be led astray here, and a very good way to check your answer is to write a simulation.

You are not required to do this for the homework, but it's not a bad idea if you want to see if you were right.

(a) *A blush* All 5 cards are red.

(b) *A flash.* All four suits occur in the hand. (This means that one suit will occur twice, and the others once, which is a hint for solving the problem. The problem gets more complicated if you attempt to filter out flashes that belong to some other category of poker hand, like pairs or straights. So just assume that any hand containing all four suits is a flash.)

(c) *A royal scandal.* The hand contains two Kings, two Queens, and a Jack.

3 Elections

There are two candidates in an election. Candidate A has received 55% of the votes, candidate B 45%. There is a very large number of voters (several million, let's say). We randomly sample 100 voters. What is the probability that in this sample, candidate B receives more votes? (In other words, that the poll does not correctly predict the outcome of the election.) Express this answer as a formula using the binomial coefficients, and then compute the numerical value.
HINT: Strictly speaking, this is sampling without replacement, since we should not poll the same voter twice! But the voter pool is so large and the sample so small in comparison, that you can treat it as a problem of sampling with replacement, which makes the calculation somewhat easier. You can then think of the problem as one of flipping 100 biased coins in succession. We saw how to express the probability of getting exactly k heads in terms of binomial coefficients, so here you will have a sum of about 50 such probabilities. To obtain a numerical value you will need to write a little bit (like maybe one line) of code.

4 Appendix: Implementation Notes

4.1 How to compute binomial coefficients

Python doesn't have a nice built-in function to compute binomial coefficients, and surprisingly, matplotlib does not seem to either. The function `binom` lives in a library called `scipy.special`, so you can do something like this to compute $\binom{n}{k}$:

```
from scipy.special import binom
binom(n,k)
```

I was under the impression that `scipy` was installed automatically when you installed `numpy` and `matplotlib`, but last year a few students had issues with this. If the above doesn't work, you can do a separate installation of scipy, by typing the following line in the Terminal on MacOS, or the Command Prompt in Windows:

```
python -m pip install -U scipy
```

If all else fails, you can use the following code:

```
def binom(n,k):
    prod = 1.0
    for j in range(k):
        prod = prod*(n-j)/(j+1)
    return prod
```

But *don't* try to implement the formula for $\binom{n}{k}$ directly, as in:

```
1.0*math.factorial(n)/(math.factorial(n-k)*math.factorial(k))
```

This is far more time-consuming and leads to very large values during the computation (which can even cause overflow in some environments). It's even worse if you try to use the Pascal's triangle identity and compute it recursively, with something like

```
binom(n,k)=binom(n-1,k-1)+binom(n-1,k),
```

as this leads to an exponential number of calls to the function.

4.2 Reading from the `.csv` file

I've posted a little Python code that reads the file. The file itself consists of 5 columns of 1461 rows, preceded by a row of column headers. The function I've provided does little more than read from the text file: It returns a list of 1462 lists, each with 5 *strings*. It is up to you to remove the header row, convert the strings to the corresponding integers, and transpose the table, so that you have a list of 5 lists of 1461 integers. You can experiment with the numpy tools for manipulating arrays, or hand-code it, a nice list-manipulation exercise. After that, creating the plots is a piece of cake.

4.3 Simulation with the real birthday data

The `choice` function has an optional argument `p` that lets you specify a non-uniform distribution on the data to be sampled. To see how this works, a call to

```
choice([1,2,3,4,5,6],3,p=[0.3,0.2,0.3,0.1,0.05,0.05])
```

will return an array of 3 values in $\{1, 2, 3, 4, 5, 6\}$ distributed according the probability mass function p: that is, 1 will occur with probability 0.3, 2 with probability 0.2, etc, so this simulates a roll of three identical unfair dice. It is up to you to make sure that the components of the list `p` add up to 1.

In your problem, you will be working with the probability mass function on the set $\{1, \ldots, 366\}$ determined in the first part of the problem, and use the `choice` function and the information read in from the file to randomly generate birthdays. The most straightforward way of solving the problem is to repeatedly generate k birthdays (let's say 1000 times) for $k = 2, \ldots, 65$ and record the proportion of trials in which there was a coincidental birthday. How do you check if the list or array `x` of k birthdays contains a repeat? I don't know the best way to do this, but here is some very simple code: The expression

```
len(set(x))==len(x)
```

has the value `True` if and only if `x` contains no repeats.

Optional: Here is a less straightforward way that will run faster: In each trial, repeatedly sample birthdays using the given distribution, until you find a repeat, and record the number of birthdays you sampled. The proportion of trials that give the result k is approximately

$$q_k = p_k - p_{k-1},$$

where

$$p_k = P(\text{there is a repeated birthday in a group of } k \text{ people }).$$

Thus the sum

$$p_k = \sum_{j=2}^{k} q_j$$

gives the value we're looking for. So you can just collect the numbers obtained for a large number of trials, compute the *cumulative* histogram of these, and plot the result as a line chart. More work for you, but less computation time.