Assignment 3

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

1 Poker

If I asked you to compute the probabilities of various poker hands, it would take you just a few seconds find the Wikipedia page 'Poker odds' with all the answers, 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.

2 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 code.

3 The birthday party never ends!

I can't seem to leave the birthday stuff alone, and I had to restrain myself from making *every* problem about birthdays.

3.1 *Somebody* has a birthday today.

When we did the experiment in class on January 25 with a group of 30 people, we found that there was a student whose birthday was that very day. Now, because of our calculations, I knew going in that I would probably find two students with the same birthday. But what is the probability that I would find a student with a birthday on January 25? Give the answer both for 30 students, and in general for k students. How many students would have to be present to make this probability at least $\frac{1}{2}$? You can solve this problem exactly, or use the exponential approximation.

3.2 Births and deaths

Suppose you have a database of biographies of prominent people from the past. Each biography contains a date of birth and a date of death. If there are 1000 records in the database, what is the probability that two of them share both a date of birth and a date of death (we are ignoring the year of birth and the year of death, and just looking at the month and the day)? You should use the exponential estimate for the generalized birthday problem.

3.3 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 above). 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.

You won't be graded on this, but you are invited to speculate on the reasons for these non-uniformities.

(b) You are to estimate, for $k = 1$ to 65, the probability of a coincidental birthday in 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?

Some advice on how to go about this: First, you need some way of generating random birthdays based on this probability model. You can code this by hand using `rand()`, but there is a built-in method in numpy. You have to add to your program

```
import numpy.random as npr
```

You then use a function called `choice`. To see how this works, a call to

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

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 0.3, etc. That is, it will simulate a roll of three loaded dice.

In your problem, you will be working with the probability mass function on the set $\{1, \ldots, 366\}$ determined in (a), 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.

Optional: there is a nice trick for speeding things up: 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.

4 Appendix

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 $\binom{n}{k}$:

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

Contrary to what I believed, `scipy` is not automatically installed when you install matplotlib. The following line, typed in the Terminal on MacOS, or the Command Prompt in Windows should install it:

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

Alternatively, 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
```

How *not* to compute binomial coefficients.

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 Exponential approximation

The tangent line to the graph of $y = e^x$ at $x = 0$ is $y = 1 + x$. This means that for small values of x, $1 + x$ is a very good approximation to e^x, *and vice-versa*. In fact, for $-1 < x < 0$, the error is less than $\frac{x^2}{2,}$ so for example, if $x \approx 0.1$, then the error is less than 0.005. The tangent line lies completely below the graph of $y = e^x$, so that $1 + x < e^x$ for all $x \neq 0$.

We can apply this to get an easy-to-compute approximate solution to the birthday problem. More importantly, we can apply this to get such a solution to problems that have the same structure, like finding the probability of a collision when we hash k items into a hash table of size n, but where the numbers are too large to make an exact computation impractical.

Recall that the probability that no two people in a group of k people share a birthday is

$$\frac{365 \cdot 364 \cdots (365 - k + 1)}{365^k}.$$

We can rewrite this, and apply the approximation, to get

$$\begin{aligned}
\frac{364}{365} \cdot \frac{363}{365} \cdots \frac{365-(k-1)}{365} &= \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdots \left(1 - \frac{k-1}{365}\right) \\
&< e^{\frac{-1}{365}} \cdots e^{-(k-1)/365} \\
&= e^{-(1+2+\cdots+(k-1))/365} \\
&= e^{-k(k-1)/(2\cdot 365)} \\
&< e^{-(k-1)^2/730}.
\end{aligned}$$

Observe that as the number k of people gets larger, this probability gets smaller. If we want to figure out at what point it falls below, say, 0.1, we solve

$$e^{-(k-1)^2/730} = 0.1.$$

We take reciprocals:

$$e^{(k-1)^2/730} = 10,$$

then take natural logs:

$$\frac{(k-1)^2}{730} = \ln 10,$$

so

$$k = 1 + \sqrt{730 \cdot \ln 10} \approx 41.$$

Thus if 41 people are present, the probability of a coincidental birthday is *at least* 0.9. The calculation was made just using the fact that $1 + x < e^{-x}$, so it just gives a lower bound for the probability of a coincidental birthday. But in fact, this is very close to the exact probability: With 41 people present, the probability of a coincidence is 0.903.

4.3 Reading the data file in Python

First download it into the directory in which you are doing your work. To open it for reading, execute the statement

```
f=open('usbirths2000-2003.csv','r')
```

To read the next line from the file and assign it to a string `s`, execute the statement

```
s=f.readline()
```

The first line of the file is a header describing the fields. Every subsequent line consists of 5 fields, separated by commas, and terminated by the
`n` newline character. For example, the third line of the file is

```
2000,1,2,7,8086\n
```

This means that 8086 people were born on January 2, 2000, which was a Sunday (the numbering for days of the week begins with 1 for Monday).

To obtain the separate fields of a line, strip away the newline character and apply the `split` function, so that if you have already assigned the line above to `s`, then a call to

```
 s[:-1].split(',')
```

returns the list

```
 ['2000','1','2','7','8086']
```

For example, the fragment of code below determines the total number of people born in the years 2000-2003, as well as the number of people born on Monday.

```
 #find number of Monday births and total number of births.

f=open('usbirths2000-2003.csv','r')
#read first line
s=f.readline()
totalbirths=0
mondaybirths=0
while(True):
    s=f.readline()
    if s=='':
        break
    fields=s[:-1].split(',')
    totalbirths+=int(fields[4])
    if fields[3]=='1':
        mondaybirths+=int(fields[4])
print totalbirths
print mondaybirths
```