Homework 1: Random Events and Probability

UA CSC 380: Principles of Data Science

Homework due at 11:59pm on Sep 9, 2023

Deliverables Your must make two submissions: (1) your homework as a SINGLE PDF file by the stated deadline to the gradescope (Homework 1). Include your code and output of the code as texts in the PDF. and (2) your codes in HW01.ipynb file to a separate submission (Homework 1 code). Each subproblem is worth 10 points. More instructions:

- You can hand-write your answers and scan them to make it a PDF. If you use your phone camera, I recommend using TurboScan (smartphone app) or similar ones to avoid uploading a slanted image or showing the background. Make sure you rotate it correctly.
- Watch the video and follow the instruction for the submission: `https://youtu.be/KMPoby5g_nE`
- **Show all work along with answers to get the full credit**.
- **Paste all your codes and outputs in the report to get full credit.**
- Place your final answer into an 'answer box' that can be easily identified.
- Map the questions with your solutions when submitting. Points will be deducted if not following this.
- There will be no late days. Late homeworks result in zero credit.

Failure to follow the submission instruction will result in a minor penalty in credit.

You can choose to work individually or in pairs.

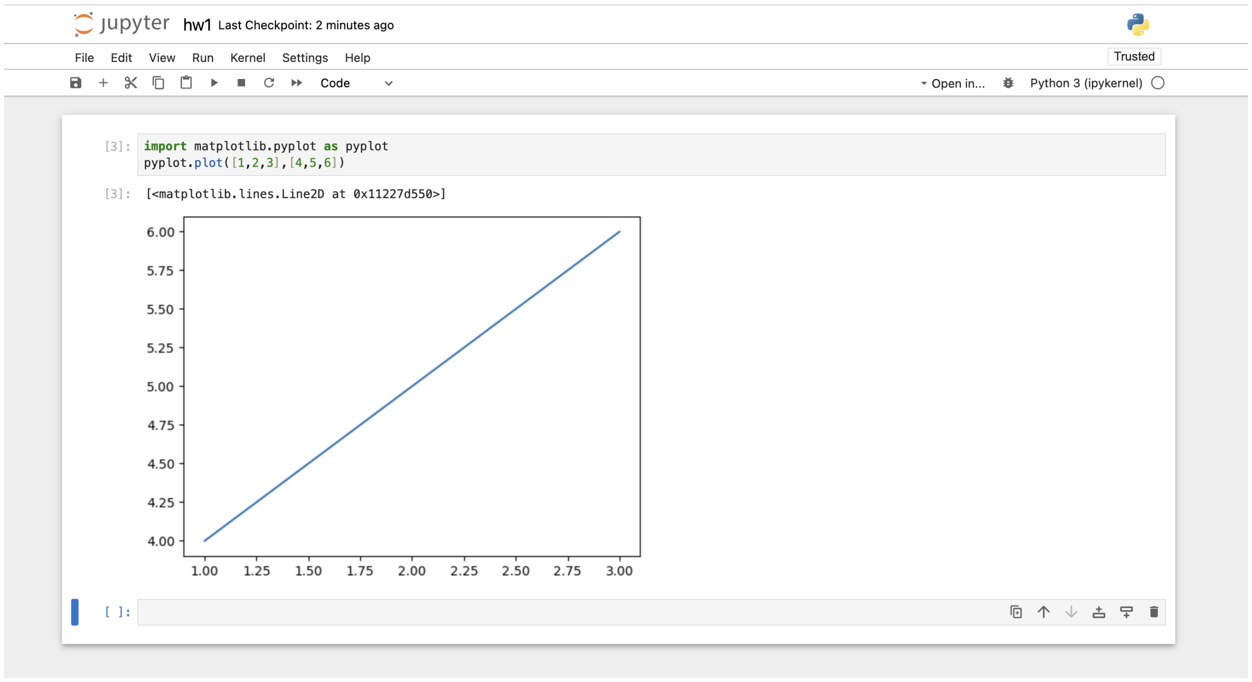
- If you choose to work in pairs, you are free to discuss whatever you want with your partner; please make only one submission per group.
- Please do not discuss with people outside your group about the homework (refer to the academic integrity policy in Lecture 1).
- If you have clarification questions, please feel free to post on Piazza so that it can promote discussion.

Problem 1: Programming Setup

In this problem, let's set up the python environment using Conda. You do not need to submit any code for this problem.

Install Python 3 with Conda. Create your own virtual environment (see https://numpy.org/install/ for an example), and in your virtual environment, install numpy, scipy, matplotlib. Install Jupyter lab. Start the jupyter lab and start a jupyter notebook. Import matplotlib.pyplot and plot anything in there (e.g., pyplot.plot([1,2,3],[4,5,6])).

Please screen capture the jupyter notebook screen (the code and the shown plot) and report it in your submission. Once you are done, you may want to deactivate your own virtual environment. If you have already been using Jupyter Lab, no need to reinstall; paste the screen shots in your report.



```
import matplotlib.pyplot as pyplot
pyplot.plot([1,2,3],[4,5,6])
```

```
[<matplotlib.lines.Line2D at 0x11227d550>]
```

Problem 2: Random Dice

This problem will compare the theoretical properties of a fair die to empirical results from simulation. It will further familiarize you with the `numpy.random` library.

a) Assume that we roll two fair six-sided dice. Let E be the event that the two dice's outcomes sum to 3. What is the theoretical probability of E?

There are 36 possible outcomes in total.
The outcomes where two dice sum to 3 are:
(1,2), (2,1).
So, there are 2 outcomes out of 36 total possible outcomes where event E happens.
Therefore,

$$E = \frac{2}{36} = \frac{1}{18}$$

b) Initialize the random seed to 2023 using `numpy.random.seed`. Using `numpy.random.randint`, simulate 1,000 throws of two fair six-sided dice. Paste your code here.

```
>>> import numpy as np
>>> np.random.seed(2023)
>>> dice1 = np.random.randint(6, size=1000) + 1
>>> dice2 = np.random.randint(6, size=1000) + 1
>>> sim = [(dice1[i],dice2[i]) for i in range(len(dice1))]
>>> sum_three = list(filter(lambda x: x[0] + x[1] == 3, sim))
>>> emp_freq = len(sum_three) / 1000
>>> print("%.3f" % emp_freq)
0.055
```

From these simulations, what is the empirical frequency of E (i.e., the percentage of times this event occurred in simulation)?

Empirical frequency = 0.055

c) Reset the random seed to 2023 and repeat the above simulation a total of 10 times and report the empirical frequency of E for each run. Paste your code here.

```
>>> np.random.seed(2023)
>>> for i in range(10):
...     dice1 = np.random.randint(6, size=1000) + 1
...     dice2 = np.random.randint(6, size=1000) + 1
...     sim = [(dice1[i],dice2[i]) for i in range(len(dice1))]
...     sum_three = list(filter(lambda x: x[0] + x[1] == 3, sim))
...     emp_freq = len(sum_three) / len(sim)
...     print("run %2d: %.3f" % (i + 1, emp_freq))
```

```
...
run  1: 0.055
run  2: 0.051
run  3: 0.056
run  4: 0.061
run  5: 0.060
run  6: 0.055
run  7: 0.052
run  8: 0.061
run  9: 0.068
run 10: 0.059
```

d) The empirical frequency of E from each simulation will differ. Why do these numbers differ? Yet, the probability of E is fixed and was calculated in part (a) above. Why does the probability disagree with the empirical frequencies?

The numbers for the empirical frequencies differ because the empirical frequencies are formed from a collection of random obeservations.
In this case, the random observations are the random integers (in the range of 1 to 6 inclusive) generated by numpy.
Therefore, each the empirical frequency for each run will slightly differ as each is comprised of 1000 random observations.
More specificially, any given run can randomly have more or less rolls that sum up to 3 than another run.
The theoretical frequency of E may not match all the emprical frequencies becuase the theoretical frequency is calculated as the number of possible outcomes that we are looking for divided by the total number of possible outcomes. It is the expected frequency based on math and what we know of the situation. On the other hand with empirical frequncy, we are collecting a large number of observations and then making the calculation based on what we observed. Thus, the expected frequency may vary from what we observe due to the randomness of the data we are collecting.

e) In the above we have estimated the probability of an event by performing $1,000$ rolls of two dice each. We generated 10 different estimates by repeating this procedure. How do our results change if we instead perform $10,000$ rolls, and repeat 10 times? Try it, report the difference, and discuss why. Paste your code here.

```
>>> np.random.seed(2023)
>>> for i in range(10):
...     dice1 = np.random.randint(6, size=10000) + 1
...     dice2 = np.random.randint(6, size=10000) + 1
...     sim = [(dice1[i],dice2[i]) for i in range(len(dice1))]
...     sum_three = list(filter(lambda x: x[0] + x[1] == 3, sim))
...     emp_freq = len(sum_three) / len(sim)
...     print("run %2d: %.4f" % (i + 1, emp_freq))
```

```
...
run  1: 0.0546
run  2: 0.0566
run  3: 0.0551
run  4: 0.0558
run  5: 0.0546
run  6: 0.0564
run  7: 0.0560
run  8: 0.0586
run  9: 0.0550
run 10: 0.0555
```

With 10,000 rolls, our 10 empirical frequencies are closer to the theoretical frequency E than with 1000 rolls. This is because the empirical frequency gets closer the theoritical (expected) frequency with the increasing number of observations. This is according to the law of large numbers. Intuitively, we can also understand that with a larger sample size, all the (extreme) outliers in the data will balance each other out since they occur equally enough at both ends of the spectrum. Thus, we approach the expected frequency.

Problem 3: Coinflips

Suppose we flip a fair coin 10 times. What is the probability that the following events occur: I recommend that you use the code like Problem 1 to debug your answers (but this debugging itself is not part of the evaluation).

a) The number of heads and the number of tails are equal

R code $P(X = 5)$**:**
dbinom(5, 10, 0.5)
0.2460938

b) There are strictly more heads than tails

R code $1 - P(X <= 5)$**:**
1-pbinom(5,10,0.5)
0.3769531

c) The number of heads and the number of tails are equal, but now with the assumption that the head probability is 0.2 (unfair coin).

R code $P(X <= 5)$**:**
dbinom(5, 10, 0.2)
0.02642412

Problem 4: Conditional Probability

a) Assume that we roll two fair six-sided dice. What is $P(\text{sum is 5} \mid \text{first die is 2})$? What is $P(\text{sum is 5} \mid \text{first die is 5})$?

$P(\textbf{sum is 5} \mid \textbf{first die is 2}) = \frac{1}{6}$
If the first die is 2, there are 6 possible rolls with first die as 2. Out of those 6 possible rolls, only 3 on the second die gives a sum of 5.
$P(\textbf{sum is 5} \mid \textbf{first die is 5}) = 0$
If the first die is 5, there is no chance the sum adds up to 5 since the second die has to be greater than 0.

b) Assume that we roll two fair four-sided dice. What is $P(\text{sum is at least 4})$? What is $P(\text{First die is 1})$? What is $P(\text{sum is at least 4} \mid \text{first die is 1})$?

$P(\textbf{sum is at least 4}) = \frac{13}{16}$
Writing out 4x4 grid of possbilities shows this (not shown).
$P(\textbf{First die is 1}) = \frac{1}{4}$
Again, 4x4 grid shows this.
$P(\textbf{sum is at least 4} \mid \textbf{first die is 1}) = \frac{1}{2}$
Look at top row of 4x4 grid.

c) Suppose two players each roll a die, and the one with the highest roll wins. Each roll is considered a "round" and further suppose that ties magically don't happen (or those rounds are simply ignored) so there is always a winner. The best out of 7 rounds wins the match (in other words the first to win 4 rounds wins the match). Let W be the event that you win the whole match. Let $S = (i, j)$ be the current score where you have i wins and the opponent has j wins. Compute the probability that you win the match, given the current score, i.e., $a_{i,j} := P(W \mid S = (i, j))$ for each of the 16 possible values of $S = (i, j)$, $i, j \in \{0, 1, 2, 3\}$.

$(0, 0), (1, 1), (2, 2), (3, 3) = \frac{1}{2}$

To help you out a bit with part (c) above, I am providing the following hints:

1) If $i = 4$ and $j < 4$, $a_{i,j} = 1$. OTOH, if $i < 4$ and $j = 4$, $a_{i,j} = 0$. Can you see why?

2) Let R_i be a random variable where $R_i = 1$ if you win round i and $R_i = 0$ if you lose that round. Note that $P(R_i = 0) = P(R_i = 1) = \frac{1}{2}$.

3) Recall that by the law of total probability $P(W \mid S = (i, j)) = P(W, R_{i+j+1} = 1 \mid S = (i, j)) + P(W, R_{i+j+1} = 0 \mid S = (i, j))$.

4) By the probability chain rule $P(W, R_{i+j+1} \mid S = (i, j)) = P(W \mid R_{i+j+1}, S = (i, j))P(R_{i+j+1} \mid S = (i, j))$.

5) Although it requires rigorous argument, for this problem, you can take it as given that $P(W \mid R_{i+j+1} = 1, S = (i,j)) = P(W \mid S = (i+1, j))$ and $P(W \mid R_{i+j+1} = 0, S = (i,j)) = P(W \mid S = (i, j+1))$. (Can you see why, intuitively?)

6) Find a way to write down $a_{i,j}$ as a function of $a_{i+1,j}$ and $a_{i,j+1}$. This will help you compute the answers in a recursive manner.

7) As a sanity check, when the current score is equal–that is $S = (k,k)$–then there should be equal chance of either player winning the match, $P(W \mid S = (k,k)) = \frac{1}{2}$ for $k = 0, 1, 2, 3$.