Homework 4

PSTAT 5A: Spring 2023, with Ethan P. Marzban

Instructions

- Please submit your work to Gradescope by no later than **11:59pm on Tuesday, May 2**. As a reminder, late homework will not be accepted.
- Recall that you will be asked to upload a **single** PDF containing your work for *both* the programming and non-programming questions to Gradescope.
  - You can merge PDF files using either Adobe Acrobat, or using adobe's online PDF merger at this link.

Problem 1: Classifying Random Variables

Classify each of the following random variables as either discrete or continuous. Justify your answers.

a. X = the height (in ft) of a randomly selected skyscraper in Los Angeles
b. Y = the number of car crashes we observe at a particular intersection
c. Z = the number of blueberries we find in a randomly-selected cubic-inch of muffin
d. W = the amount of time (in minutes) a person has to spend waiting at a randomly-selected red light

Problem 2: A Discrete Random Variable

Let X be a random variable with the following probability mass function (p.m.f.):

k	-1	-0.5	0	1.4	3
$\mathbb{P}(X = k)$	0.12	0.23	0.34	0.07	a

a. What must the value of a be?
b. What is the probability that X is either 0 or 1?
c. Compute $\mathbb{P}(X \leq 0)$.
d. Compute $\mathbb{E}[X]$, the expected value of X
e. Compute $\text{Var}(X)$, the variance of X.

Problem 3: Binomial, or Not?

For each of the following random variables, determine whether they are Binomially distributed or not. If they are binomially distributed, provide the parameters of the Binomial distribution they follow.

a. X = the number of heads we observe in n independent tosses of a *biased* coin that lands 'heads' with probability 0.8.

b. Y = the number of people who own a car in a pool of 100 UCSB students that is selected *without* replacement.

c. Z = the number of people who own a car in a pool of 100 UCSB students that is selected *with* replacement.

d. You roll a fair 6−sided die, and record the number showing; then, you toss as many coins as there are spots showing on the die (e.g. if the die shows the number 3, you then toss 3 coins). Let W = the number of heads in these coin tosses.

Problem 4: Tracking Defects

It is found that 2% of *GauchoCompute*-brand laptops contain a defect. An inspector selects 52 *GauchoCompute*-brand laptops with replacement, and records the number of defective laptops.

a. Define the random variable of interest.
b. Does X follow the Binomial Distribution? (Be sure to check the conditions!) If so, what are the values of n and p?
c. What is the probability that the inspector observes exactly 34 defective laptops in her sample of 52?
d. What is the probability that the inspector observes at least 34 defective laptops in her sample of 52?
e. What is the expected number of defective laptops the inspector will observe in her sample of 52?

Problem 5: I've Got a Golden Ticket!

At the State Fair, you encounter the following game: a gamemaster has a box containing 4 blue tickets, 3 red tickets, and 1 gold ticket and selects a single ticket at random from this box. If the ticket selected is blue you win a dollar, if it is red you *lose* a dollar, and if it is gold you win 10 dollars. Let W denote your winnings after playing this game once.

a. What is S_W, the state space of W?
b. Find the p.m.f. (probability mass function) of W.
c. What is the expected amount you will win each time you play the game? (I.e. what is $\mathbb{E}[W]$?)

Problem 6: Programming

Part (a): Basic Plotting

! Task 1

Plot the function $f(x) = x\sin(x) + xe^{-x^2}$ between $x = -4\pi$ and $x = 4\pi$. Add appropriate axis labels, and add a title.

! Task 2

Part (b): Copy-paste the following code into a cell, and run the cell:

```
import numpy as np
np.random.seed(5)

x1 = np.random.normal(0, 1, 100)
x2 = (1/x1) + np.random.normal(0, 1, 100)
```

Generate a scatterplot of `x1` vs `x2` (i.e. `x1` should be on the horizontal axis and `x2` should be on the vertical axis), and comment on whether there appears to be an association between `x1` and `x2`. If there is an association, comment on whether it is positive/negative and linear/nonlinear.

Part (b): Superimposing Plots

It will sometimes be necessary to superimpose two or more plots on top of each other. The goal of this problem is to walk through how to do this.

Recall the following: given a function `f()` and a variable `x` that has been assigned a value resulting from a call to `numpy.linspace()`, we generate a graph of `f()` using (assuming `matplotlib.pyplot` has been imported as `plt`):

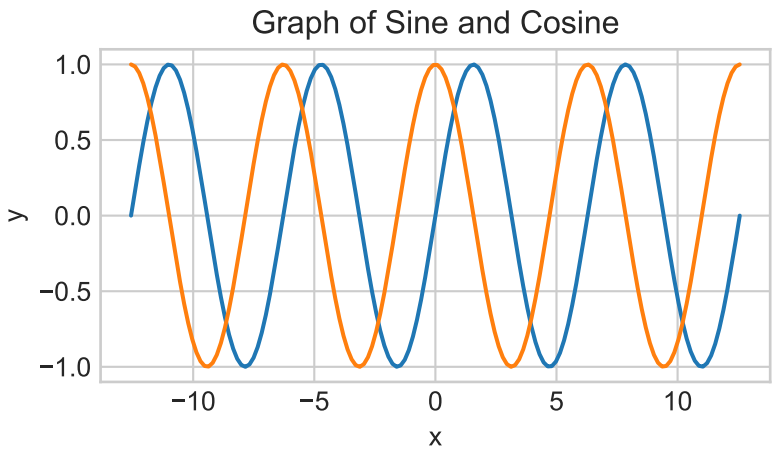
```
plt.plot(x, f(x));
```

It stands to reason, then that given another function `g()` we should be able to superimpose the graph of `g()` onto the graph of `f()` by simply adding another call to `plt.plot()`:

```
plt.plot(x, f(x));
plt.plot(x, g(x));
```

! Task 3

Generate a graph of `sin()`; on top of this graph, superimpose the graph of `cos()`. Restrict the `x` values on the graph to be between -4π and 4π. Your final graph should look like the following (**pay attention to the axis labels and title!**):



Now, as it stands, it's a bit difficult to determine which curve corresponds to the sine curve and which corresponds to the cosine curve. As such, we should add some labels!

! Task 4

Copy your code from Task 3 above into a new code cell, and

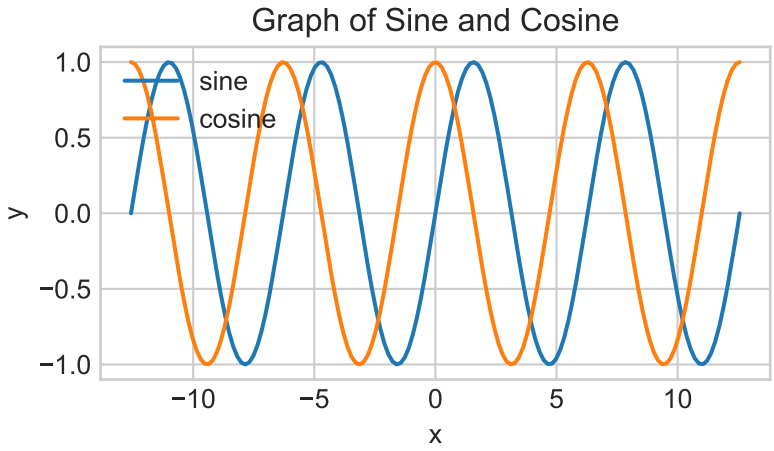
- add `label = "sine"` to your call to `plt.plot()` containing the sine curve
- add `label = "cosine"` to your call to `plt.plot()` containing the cosine curve.

Does this new plot look any different than the plot you generated in Task 3?

Hm, doesn't look like anything changed... That's because we didn't add a **legend** to our plot! To add a legend, we simply tack on a call to `plt.legend()` after our code from above.

! Task 5

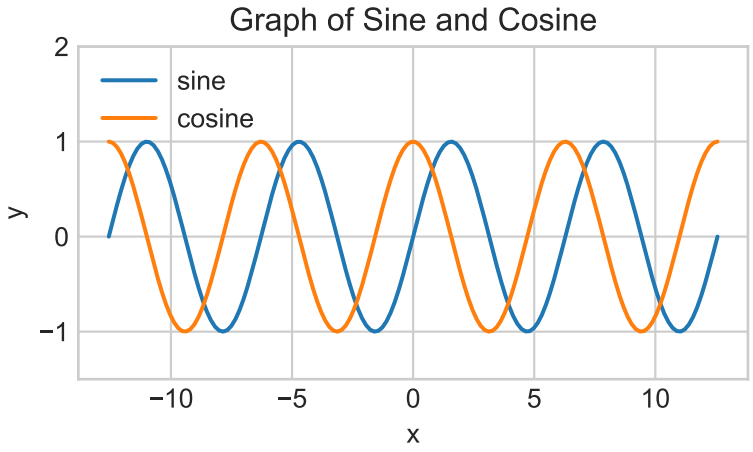
Copy your code from Task 4 above into a new code cell, and add a line underneath it containing a call to `plt.legend()`. Look up the help file to figure out what arguments you need to pass in to obtain the following graph (note the **position** of the legend):



Okay, we're almost there! The only issue is that now the legend is covered up by the actual graphs. One way we can fix this is by extending the $y-$axis further, using the function `plt.ylim()`:

! Task 6

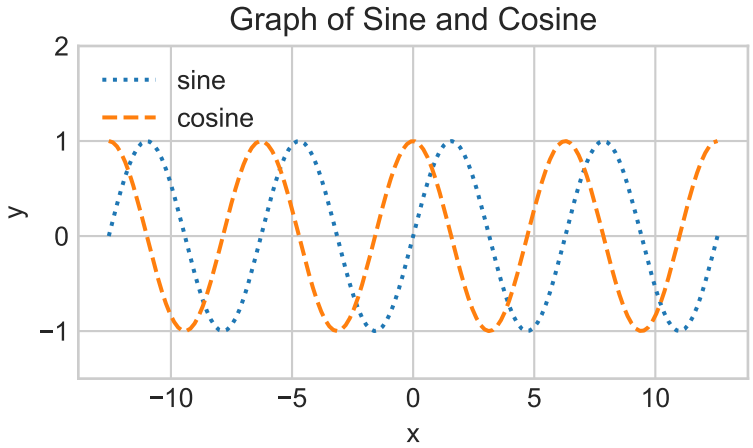
Copy your code from Task 5 above into a new code cell, and add a line underneath it containing a call to `plt.ylim()`. Look up the help file to figure out what arguments you need to pass in to obtain a lower $y-$limit of -1.5 and an upper $y-$limit of 2.0. Your final graph should look like this:



Finally, it is sometimes considered bad form to rely too heavily on colors in plots. This is because doing so alienates readers who are colorblind. One way around this is to rely on different *line types*; e.g. used dashed lines for one graph and dotted lines for another.

! Task 7

Copy your code from Task 6 above into a new code cell. Read the following help file and figure out how to pass in a value to the `linestyle` argument to your two calls to `plt.plot()` to generate the following plot:



Note that the sine curve is now dotted, and the cosine curve is now dashed.