Data C100/C200, Midterm Exam

Spring 2023

Name: _____

Email: _____@berkeley.edu

Student ID: _____

Name of the student to your left: _____

Name of the student to your right: _____

Instructions:

Do not open the examination until instructed to do so.

This exam consists of **80 points** spread out over **7 questions** and must be completed in the **110 minutes** time period on March 9, 2023, from 7:10 PM to 9:00 PM unless you have pre-approved accommodations otherwise.

Note that some questions have circular bubbles to select a choice. This means that you should only **select one choice**. Other questions have boxes. This means you should **select all that apply**. Please shade in the box/circle to mark your answer.

There is space to write your student ID number (SID) in the upper right-hand corner of each page of the exam. **Make sure to write your SID on each page** to ensure that your exam is graded.

Honor Code [1 pt]:

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I am the person whose name is on the exam, and I completed this exam in accordance with the Honor Code.

Signature: _____

1 Pandas Pandemonium [17 Pts]

Bella and Kanu are opening a zoo full of their favorite animal – pandas! To organize their zoo logistics, they've created a DataFrame `pandas_df` to store information about each of their pandas. The first five rows of `pandas_df` are shown below. You may assume that each panda has a unique name.

	Name	Age	Favorite Food	Height	Weight	Birthday
0	Minh	4	bamboo	2.5	170.34	January 5 2019
1	Shiangyi	9	cotton candy	NaN	160.56	October 12 2013
2	Vasanth	10	potato chips	3.2	200.23	November 15 2012
3	Shiny	4	bamboo	NaN	205.68	December 28 2018
4	Rohan	5	ice cream	2.1	180.61	September 20 2017

(a) [3 Pts] Bella wants to perform exploratory data analysis on the dataset. To familiarize herself with the data, she first identifies the variable types of important columns in `pandas_df`. Determine the **variable type** that best describes each of the following columns in `pandas_df`. Assume `"Age"` values are rounded to the nearest integer.

	Quantitative Continuous	Quantitative Discrete	Qualitative Ordinal	Qualitative Nominal
(i) `"Name"`	○	○	○	●
(ii) `"Age"`	○	●	○	○
(iii) `"Weight"`	●	○	○	○

Solution: (i) `"Name"` is qualitative nominal because it describes non-ordered, non-numeric data (ii) `"Age"` is quantitative discrete because it represents a quantity (number of years) that can only take on whole values (iii) `"Weight"` is quantitative continuous because it describes a numerical quantity that can take on decimal values

(b) [1 Pt] What is the **primary key** of `pandas_df`?

- ● **`"Name"`**
- ○ `"Age"`
- ○ `["Name", "Age"]`
- ○ `["Name", "Age", "Height"]`

The remainder of this question involves coding. All code for each part, where applicable, must be written in Python. Assume that the `pandas` library has been imported as `pd` and the `numpy` library has been imported as `np`. ***In each blank, you can write as much code as is necessary that fits the skeleton code.***

(c) [1 Pt] Kanu wants to sort the records based on the `"Age"` column in **descending order**. Fill in the blank below to achieve this.

```
pandas_df = pandas_df._____A_____
```

Fill in blank `A`:

Solution: `sort_values("Age", ascending = False)`

(d) [2 Pts] Kanu notices that the *only* column that has `NaN` values is `"Height"`. He wants to replace these missing values through imputation. Fill in the blanks below to **replace any `NaN` values with the mean of the `"Height"` column**.

```
pandas_df = pandas_df.___A___(_____B_____)
```

(i) Fill in blank `A`:

Solution:

`fillna`

(ii) Fill in blank `B`:

Solution:

`pandas_df["Height"].mean()` or
`np.mean(pandas_df["Height"])`

(e) [2 Pts] Bella wants to know the birth month of each panda. She notices that the `"Birthday"` column of `pandas_df` *always* contains the birth month, birth date, and birth year of each panda, in order, separated by spaces. Fill in the blank to add a new column to `pandas_df` called `"Birth Month"` that contains each panda's month of birth.

```
pandas_df["Birth Month"] = \
            pandas_df["Birthday"].str._______A_______
```

The first few rows of `pandas_df` after executing parts (a) through (e) are shown below.

	Name	Age	Favorite Food	Height	Weight	Birthday	Birth Month
2	Vasanth	10	potato chips	3.2	200.23	November 15 2012	November
1	Shiangyi	9	cotton candy	2.6	160.56	October 12 2013	October
4	Rohan	5	ice cream	2.1	180.61	September 20 2017	September
0	Minh	4	bamboo	2.5	170.34	January 5 2019	January
3	Shiny	4	bamboo	2.6	205.68	December 28 2018	December

You should use `str.split`, and the documentation for the function is shown below. You may fill in blank `A` with **as many** calls to `str` as you believe are necessary.

Fill in blank `A`:

Solution: Two solutions are possible:
`split(" ").str[0]` or
`split(" ", expand=True)[0]`

pandas.Series.str.split

`Series.str.split(pat=None, expand=False)`

Split strings around given separator/delimiter.

Splits the string in the Series/Index from the beginning, at the specified delimiter string.

Parameters: **pat** : *str or compiled regex, optional*
String or regular expression to split on. If not specified, split on whitespace.

expand : *bool, default False*
Expand the split strings into separate columns.
- If `True`, return DataFrame/MultiIndex expanding dimensionality. DataFrame columns are labeled 0, 1, etc. for each split string.
- If `False`, return Series/Index, containing lists of strings.

(f) [2 Pts] Kanu wants to investigate the behaviors of pandas by diet. Fill in the blanks below to create a **new** DataFrame called `foods_df` that has the same columns as `pandas_df`, but only contains data for pandas who have a `"Favorite Food"` contained in `SELECT_FOODS`.

```
SELECT_FOODS = ["potato chips", "ice cream", "cotton candy"]
foods_df = pandas_df.loc[__________A__________, ___B___]
```

The first few rows of `foods_df` after executing parts (a) through (f) are shown below.

	Name	Age	Favorite Food	Height	Weight	Birthday	Birth Month
2	Vasanth	10	potato chips	3.2	200.23	November 15 2012	November
1	Shiangyi	9	cotton candy	2.6	160.56	October 12 2013	October
4	Rohan	5	ice cream	2.1	180.61	September 20 2017	September

(i) Fill in blank `A`:

Solution: `pandas_df["Favorite Food"].isin(SELECT_FOODS)`

(ii) Fill in blank `B`:

Solution: `:`

(g) [2 Pts] Bella wonders how a panda's diet and birth month relate to their physical growth. She wants to find the **median height** of all pandas in `foods_df` who share the same `"Favorite Food"` and `"Birth Month"`. Which option produces the DataFrame below?

Birth Month	April	December	February	January	March	November	October	September
Favorite Food								
cotton candy	2.4	5.6	2.8	2.4	2.6	0.0	2.8	2.5
ice cream	2.9	5.4	2.6	0.0	0.0	2.3	2.8	2.8
potato chips	2.9	5.1	2.8	2.0	2.2	2.9	2.6	2.8

◯
```
foods_df.groupby("Favorite Food")["Birth Month"].agg("median")
```

◯
```
foods_df.pivot_table(columns="Birth Month", \
  index="Favorite Food", aggfunc="median")
```

◯
```
foods_df.groupby( \
  ["Favorite Food", "Birth Month","Height"]).agg("median")
```

◯
```
foods_df.pivot_table(columns="Birth Month", \
  index="Favorite Food", aggfunc="median", values="Height")
```

(h) [4 Pts] Kanu notices that pandas born in certain months have abnormally large heights. He decides to note these outlier months, and then exclude them from subsequent analysis.

Fill in the blanks below to produce a **DataFrame that has the same index/columns** as `foods_df` but **only** rows where the **mean** `"Height"` of all pandas born in the same `"Birth Month"` is **less than 5 (exclusive)**. Example: If the mean `"Height"` of all pandas born in December is 5 or greater, all rows with a `"Birth Month"` of December should be removed.

```
foods_df.groupby(___A___).___B___(lambda sf: ________C________)
```

(i) Fill in blank `A`:

Solution: `"Birth Month"`

(ii) Fill in blank `B`:

Solution: `filter`

(iii) Fill in blank `C`:

Solution: `sf["Height"].mean() < 5`

2 Regularly Extra [10 Pts]

In this question, the Python regular expression library has been imported as `re`. Note that `\` is used to break strings across code lines without inserting newline characters, as below:

```
In [1]: import re

        oneline_str = "This is a string \
        with no newline characters"
        oneline_str
```

```
Out[1]: 'This is a string with no newline characters'
```

Samantha is an aspiring influencer debuting on the new video-sharing platform TokTik. As a data scientist, she wants to analyze data and understand how her audience respond to her videos. She uses regular expressions to analyze comments made by viewers of her videos.

(a) [2 Pts] Samantha wants to understand what users love about her videos. To do this, she decides to write a regex pattern that will capture the **next two words** that appear **after any word containing "love"** in a comment.

```
In [2]: pattern = # Your selected answer choice

        comment_1 = "I love the animations and all the effects she added!"
        re.findall(pattern, comment_1)
```

```
Out[2]: ['the animations']
```

```
In [3]: comment_2 = "I disliked how she explained that lolol but I loved \
        the interview part."
        re.findall(pattern, comment_2)
```

```
Out[3]: ['the interview']
```

Select the regex pattern below that will achieve this goal. The correct answer should satisfy the test cases above when the regex pattern is assigned to the variable `pattern`.

- ◯ `r"love\s(\w\w)"`
- ◯ `r"(love) [A-Za-z]+[A-Za-z]+"`
- ◯ `r"love[A-Za-z]*\s(\w+\s\w+)"`
- ◯ `r"love ([A-Za-z]+[A-Za-z]+)"`

(b) [2 Pts] Samantha has heard that internet users will convey strong emotion in comments by ending sentences with repeated punctuation, e.g., multiple periods (`.`), exclamation points (`!`), or question marks (`?`). Samantha decides to write a regex pattern that will match **any full sentence** that **ends with more than one period, exclamation point, or question mark**.

```
In [4]: pattern = # Your selected answer choice

        comment_3 = "I love this!!! So super cool!!"
        re.findall(pattern, comment_3)
```

```
Out[4]: ['I love this!!!', 'So super cool!!']
```

```
In [5]: comment_4 = 'what?!? this is crazy. My mind is blown...'
        re.findall(pattern, comment_4)
```

```
Out[5]: ['what?!?', 'My mind is blown...']
```

Select the regex pattern below that will achieve this goal. The correct answer should satisfy the test cases above when the regex pattern is assigned to the variable `pattern`.

- ◯ `r".+[.!?]{2,}"`
- ◯ `r"\w+[^.!?]*[.!?]{2,}"`
- ◯ `r"^\w+[.!?]{2,}"`
- ◯ `r"\w+\s[.!?]{2,}"`

(c) [6 Pts] Samantha's well-meaning friend tries to help by writing some regex patterns to analyze the comment data. In each of the following parts, select the output generated by running the below cell when `friend_pattern` is assigned to the corresponding regex pattern.

```
In [ ]: friend_pattern = # pattern specified in each subpart below

        comment_5 = "whos watching in 2023? Im with Samantha before \
        shes $rich and #famous haha"
        re.findall(friend_pattern, comment_5)[0]
```

(i) `friend_pattern = r"2023?"`

- ◯ `'2023'`
- ◯ `'whos watching in 2023?'`
- ◯ `'2023?'`
- ◯ `IndexError`

(ii) `friend_pattern = r"\w+$"`

- ◯ `' $'`
- ◯ `'haha'`
- ◯ `'$rich'`
- ◯ `''`
- ◯ `'shes $rich'`
- ◯ `IndexError`

(iii) `friend_pattern = r"#\w+\s+"`

- ◯ `'whos '`
- ◯ `' haha'`
- ◯ `'#famous '`
- ◯ `''`
- ◯ `'#famous'`
- ◯ `IndexError`

3 Hydration Station [10 Pts]

In the distant future, a molecular compound KSKD2 has been discovered in one of the Berkeley residential water sources. Water quality researchers have determined that while ingesting the compound is completely harmless, the compound's presence in the body could actually improve the immune systems of people who have the DNA sequence CNAM-10. As part of the research team, you want to conduct a research study to investigate how regular ingestion of water with the KSKD2 molecular compound impacts Berkeley residents that have the CNAM-10 DNA sequence.

(a) [1 Pt] What is the population you are interested in studying?

- ◯ All Berkeley residents who have the CNAM-10 DNA sequence.
- ◯ All Berkeley residents who currently drink from the water source with the KSKD2 molecule.
- ◯ **All Berkeley residents who have the CNAM-10 DNA sequence and regularly drink from the water source with the KSKD2 molecule.**
- ◯ All Berkeley residents who have the CNAM-10 DNA sequence and have drunk from the water source with the KSKD2 molecule at some point in time.

Solution: The population can be determined from the last sentence in the question setup. The keyword here is "regular ingestion of water with the KSKD2 molecular compound", so our population is those individuals that regularly drink from the water source (not necessarily those that have drunk from it at some point in time).

(b) [1 Pt] The City of Berkeley has provided you with a list of phone numbers of all Berkeley residents who drink from the water source with the KSKD2 molecular compound. You decide to call *every 100th resident* on your list and ask if they are interested in participating in your research study.

What type of sample is this? Select the choice that best describes this process:

- ◯ **Probability Sample**
- ◯ Random Sample without Replacement
- ◯ Stratified Sample
- ◯ Random Sample with Replacement

Solution:
Probability Sampling: We can find the specific chance that any set of individuals will be in the sample. The probability of every 100th individual being in the sample is 1; for all others, it's 0.
Stratified Sampling: There are no stratas of the population being considered (also, this form of sampling was not covered during the Spring 2023 semester).
Random Sample without Replacement: This is not a random sample, as not every individual has the same probability of being chosen.

Random Sample with Replacement: This is not a random sample, as not every individual has the same probability of being chosen. Additionally, this is not being conducted with replacement.

(c) [2 Pts] What are some forms of error that we may encounter with our sampling strategy from above? **Select all that apply**.

☐ **Chance Error** ☐ **Selection Bias**

☐ **Non-Response Bias** ☐ None of the above

Solution:
Chance Error: Exists when the sample is only a subset of the population, which is true here.
Non-response Bias: Exists because individuals may be unresponsive to our phone call.
Selection Bias: Exists because our sampling strategy is flawed – we are sampling individuals outside of our intended population.

(d) [2 Pts] Suppose we use Part (b)'s sampling strategy and find Candidate X is interested in participating. Which of the following **could potentially** be true? **Select all that could apply.**

Candidate X...

☐ ***is in* our sampling frame and *is in* our population of interest.**

☐ ***is in* our sampling frame but *is not in* our population of interest.**

☐ *is not in* our sampling frame but *is in* our population of interest.

☐ *is not in* our sampling frame and *is not in* our population of interest.

Solution: By nature of being sampled, Candidate X is in our sampling frame. However, they may or may not be in the population of interest because our sampling frame includes individuals outside the population of individuals with the CNAM-10 mutation.

(e) [2 Pts] Suppose that Candidate X is only interested in participating if their Berkeley roommate Candidate Y (who also drinks from the same water source) is participating. Based on Part (b)'s sampling strategy, you did not choose Candidate Y's phone number to call. However, Candidate Y reports that they have the CNAM-10 DNA sequence. Which of the following **must be** true?

Candidate Y...

- ◯ *is in* our sampling frame and *is in* our population of interest.
- ◯ *is in* our sampling frame but *is not in* our population of interest.
- ◯ ***is not in* our sampling frame but *is in* our population of interest.**
- ◯ *is not in* our sampling frame and *is not in* our population of interest.

Solution: Candidate Y is not in our sampling frame because they were not on our list of individuals to call (and thus, are not in a position that is a multiple of 100 on our phone list). However, they are in our population of interest as they carry the CNAM-10 mutation and drink from the water source with the KSKD2 molecular compound.

(f) [2 Pts] Ultimately, your team decides on a different strategy: Contact every resident in the population of interest, and ask them if they agree to participate in the study. This recruits a **participant group**. Your team then randomly assigns exactly half of this participant group to Group A and the rest to Group B. Will these two groups be **simple random samples** of the population of interest?

For full credit, **please explain your reasoning**. You will receive partial credit for a correct answer without giving a correct explanation.

- ◯ Yes
- ◯ **No**

Solution: No, the two groups are not a true simple random sample of the population of interest. By asking individuals whether or not they agree to participate in the study, there exists some selection bias in our sampling technique, as individuals who disagree to participate do not have a similar chance of being in the sample as those who agree to participate.

4 Pass It Along [12 Pts]

You are on a research team studying how Berkeley students interact with Snackpass, a food take-out app. Your team has collected and cleaned 1000 records of Snackpass data in the DataFrame `snackpass_df`, the first five rows of which are shown below.

	Order ID	Restaurant	Amount Paid	Wait Time	Calories
0	255334	Thai Basil	15.89	12.76	565.39
1	736425	Rice and Bones	5.00	2.34	475.13
2	853635	La Burrita	13.29	7.59	373.89
3	134546	Stuffed Inn	12.50	20.55	360.23
4	953624	Yifang	4.89	0.64	332.21

(a) [1 Pt] What is the **granularity** of each record in the `snackpass_df` DataFrame?

◯ One Snackpass restaurant ◯ One Snackpass user

◯ One day of Snackpass orders ◯ **One Snackpass order**

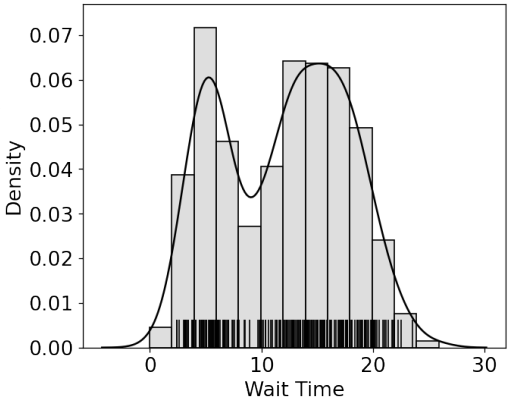
(b) [2 Pts] The researchers would like to visualize the distribution of the `"Wait Time"` column. Which of the following visualizations is/are appropriate to represent this distribution? **Select all that apply.**

☐ Bar chart ☐ Hexplot ☐ Count plot

☐ **Violin plot** ☐ **KDE plot** ☐ None of the above

(c) [2 Pts] One of the researchers decides to use a **histogram** to plot the distribution of the `"Wait Time"` column. You would like to go one step further and overlay both a **rugplot** and **kernel density estimate (KDE)** on the variable histogram.

Write code to produce a plot similar to the example. Assume Matplotlib and Seaborn are respectively imported as `plt` and `sns`. Do not worry about stylistic details like color, text size, or setting axis labels.



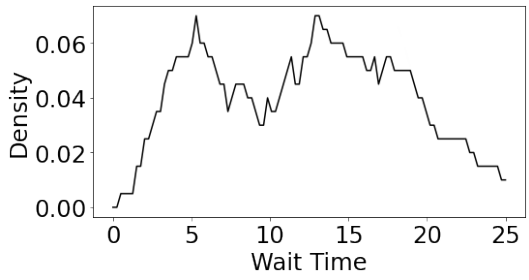
Solution:

```
sns.displot(data=snackpass_df, x="Wait Time",
    rug=True, kde=True, stat="density")
```

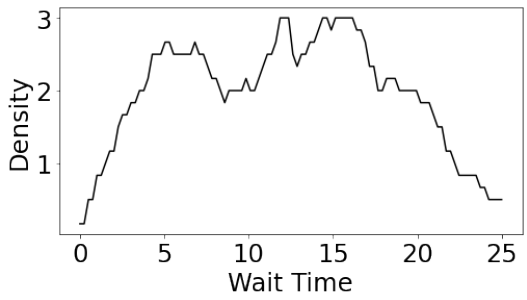
(d) [2 Pts] Another researcher decides to manually create a KDE curve. However, they produce several incorrect drafts. Which of the below plots depicts a **valid** KDE curve created using a **Boxcar Kernel**? The boxcar kernel K_α is defined as follows:

$$K_a(x, z) = \begin{cases} \frac{1}{\alpha}, & \text{if } -\frac{\alpha}{2} \leq (x - z) \leq \frac{\alpha}{2} \\ 0, & \text{otherwise} \end{cases}$$

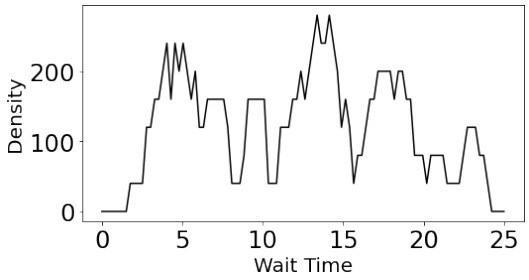
◯ A.



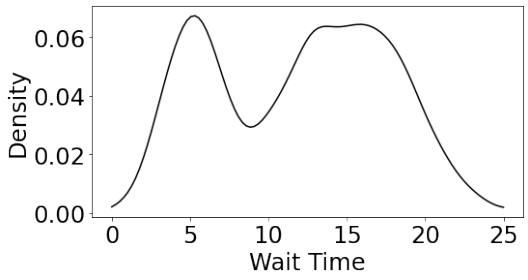
◯ B.



◯ C.



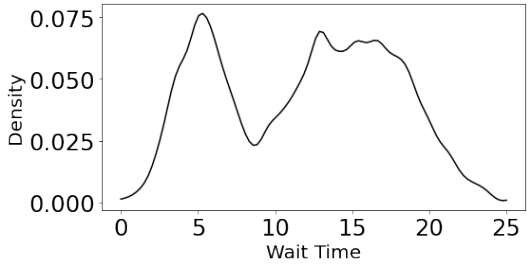
◯ D.



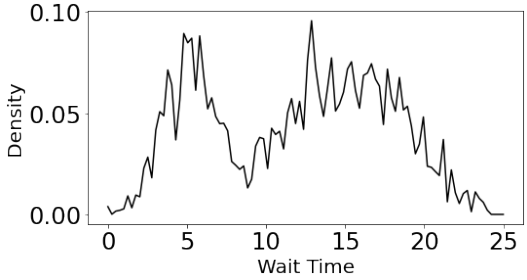
Solution: The solution is (A). Choices (B) and (C) depict non-normalized densities, as can be seen from the large density values on the vertical axis. The total area under the curve would integrate to greater than one. Choice (D) uses a Gaussian kernel, as can be inferred by the smooth curves in the KDE line.

(e) [2 Pts] The researcher decides to try again, this time using a **Gaussian Kernel**. Which of the following KDE plots were likely generated using the largest value of bandwidth parameter?

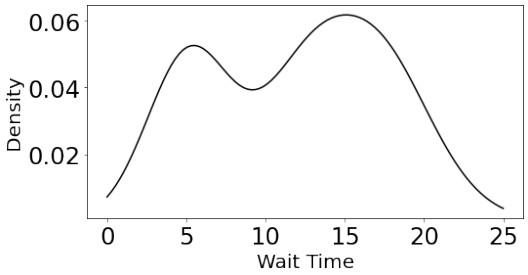
◯ A.



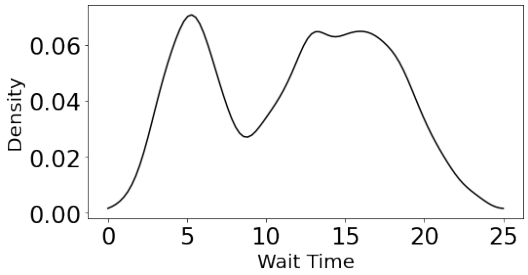
◯ B.



◯ C.

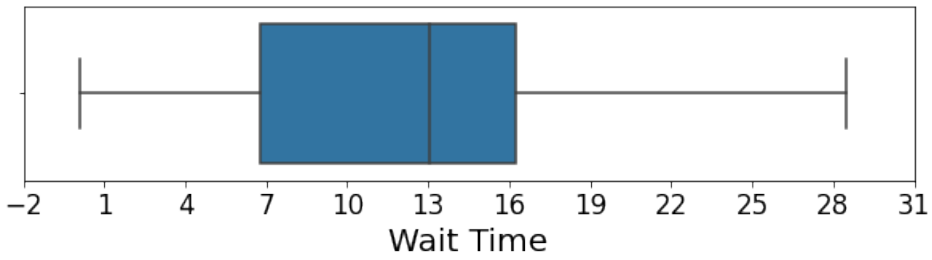


◯ D.



Solution: The correct choice is (C). The bandwidth parameter (α) of a Gaussian kernel is the standard deviation of the normal curve superimposed on each data point. A large value of α corresponds to a large standard deviation, and hence a broad curve. Because of this, a KDE generated using a large bandwidth parameter should have broad, smooth peaks. Option (C) is the smoothest of the available options.

(f) [3 Pts] A different researcher wants to use a boxplot to visualize the distribution of the `"Wait Time"` column of `snackpass_df`. You help generate the following plot:



In the subparts below, approximate by rounding to the nearest integer.

(i) What is the **median** of the `"Wait Time"` distribution? If there is not enough information to determine the median, write "N/A" below.

Solution: 13

(ii) What is the **mean** of the distribution of `"Wait Time"`? If there is not enough information to determine the mean, write "N/A" below.

Solution: N/A

(iii) In the space below, compute an approximate value for the **interquartile range** of the distribution. If there is not enough information to determine the interquartile range,

write "N/A" below.

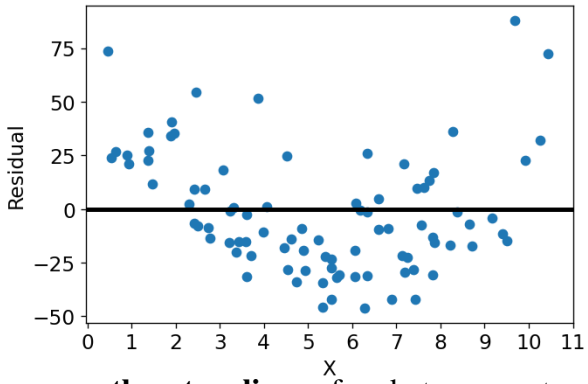
Solution: $IQR = 16 - 7 = 9$

5 Waste Not Want Not [10 Pts]

Pizza Pete, the owner of Pizza Pete's Pizzeria, wants to understand the food waste of his restaurant. Pete first records the daily amount of tomato sauce used (x) and the amount of food waste (y). He then fits a simple linear regression model to the data to predict the amount of food waste, $\hat{y}$.

Below (left) is the first few rows of the `leftovers` DataFrame that records x, y, and $\hat{y}$ for each day. Also below (right) is a residual plot that visualizes the residuals of Pizza Pete's model.

	x	y	y_hat
0	4.06	40.56	39.43
1	7.19	61.88	91.40
2	5.65	33.87	65.88
3	2.30	12.64	10.10
4	2.46	67.39	12.72



(a) [3 Pts] In the box below, write **no more than two lines** of code to generate a residual plot from `leftovers`, similar to the plot shown above. Your plot should include both the residual scatter plot and the horizontal line, which represents a residual value of 0. You don't need to match the exact styling of the plot (color, scatter point size, etc.), nor include the same axis labels.

Hint: You may find helpful the `plt.axhline` documentation at the bottom of this page.

Solution: To generate the scatterplot, either syntax below is valid:

```
plt.scatter(leftovers["x"], leftovers["y"] - leftovers["y_hat"])
sns.scatterplot(leftovers["x"], leftovers["y"]-leftovers["y_hat"])
```

To generate the horizontal line, either syntax below is valid:

```
plt.axhline(0) #xmin and xmax parameters are optional
plt.plot([0, 10], [0, 0]) #endpoint of x is arbitrary
```

matplotlib.pyplot.axhline(*y=0*, *xmin=0*, *xmax=1*, ***kwargs*)

Add a horizontal line across the Axes.

Parameters:

y : *float, default: 0*

y position in data coordinates of the horizontal line.

xmin : *float, default: 0*

Should be between 0 and 1, 0 being the far left of the plot, 1 the far right of the plot.

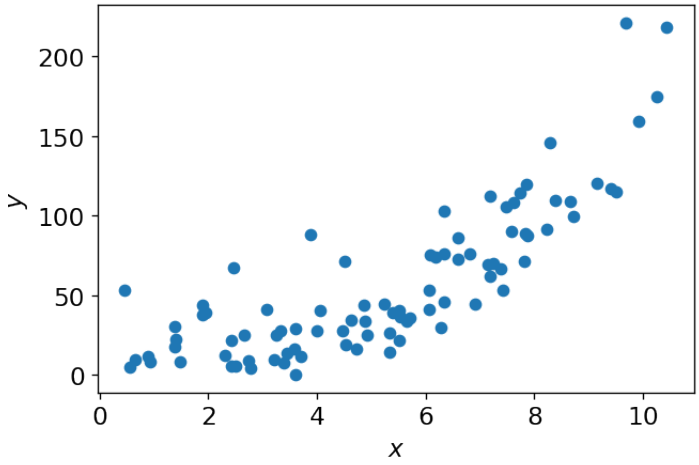
xmax : *float, default: 1*

Should be between 0 and 1, 0 being the far left of the plot, 1 the far right of the plot.

(b) [2 Pts] Pizza Pete realizes that this residual plot shows that the model is not a good fit for the data. **Referring to properties of the residual plot** on the previous page, **briefly** explain how you know that the model is not appropriate for the data. One to two sentences of explanation is sufficient.

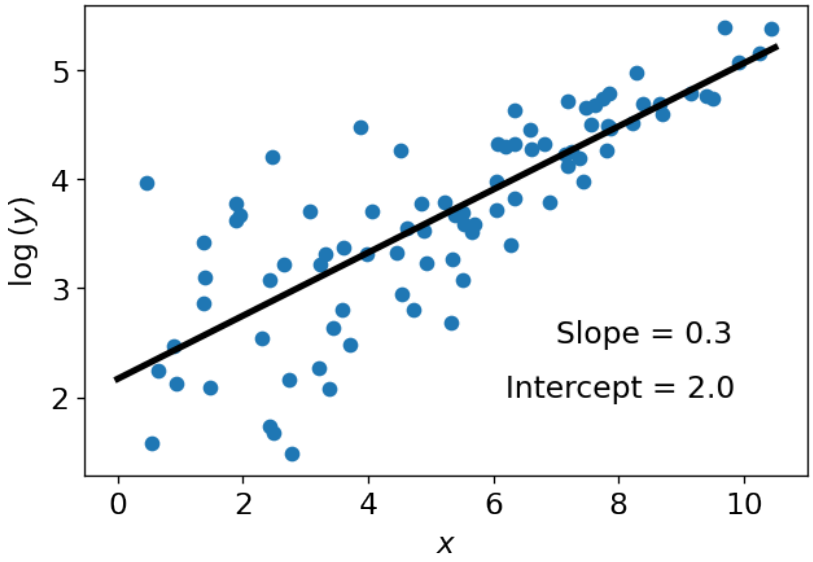
Solution: We expect the residual plot of a good-performing linear model to display no clear trends. However, there is a clear pattern present in the residual plot (the scatter points are in a roughly parabolic shape).

(c) [2 Pts] Pizza Pete creates the scatter plot of y against x shown below. Which of the following transformations is most appropriate to linearize the data?



- ◯ $\sqrt{x}$
- ◯ x^3
- ◯ y^3
- ◯ y^2

(d) [3 Pts] Pizza Pete takes the **natural logarithm (base e)** of all y values and replots the data. He then determines the simple linear regression line relating x to $\log(y)$. A scatterplot of the transformed dataset, as well as the slope and intercept of the regression line, are displayed below.



Given the visualization and the information about the simple linear regression line, derive a relationship that expresses the **original, untransformed variable y as a function of the variable x**. Your answer should take the form $y = f(x)$, where f is some function of x that you define. You don't need to simplify your final answer.

Solution:

$$\log y = 0.3x + 2$$

$$y = e^{0.3x+2}$$

6 Modeling and Loss [12 Pts]

(a) [3 Pts] For a Linear Regression model with parameter(s) $\theta = [\theta_0, \theta_1, \ldots, \theta_p]$, which of the following are true regarding Mean Squared Error (MSE) and Mean Absolute Error (MAE)? **Select all that apply.**

- ☐ **MAE is less sensitive to outliers than MSE.**
- ☐ MAE and MSE are equally sensitive to outliers.
- ☐ **Sensitivity to outliers in both MAE and MSE depends on the dataset size.**
- ☐ There is always a unique solution to the optimal parameters $\hat{\theta}$ that minimize MAE.
- ☐ There is never a unique solution to the optimal parameters $\hat{\theta}$ that minimize MAE.
- ☐ There is always a unique solution to the optimal parameters $\hat{\theta}$ that minimize MSE.

(b) [2 Pts] Suppose we have a dataset $\{(x_{i1}, x_{i2}, y_i)\}_{i=1}^n$, where for datapoint i, features 1 and 2 are x_{i1} and x_{i2}, respectively, and the target output is y_i. Is there a unique solution to the optimal parameters $\hat{\theta}$ that minimizes Mean Squared Error (MSE) for the model $\hat{y} = f_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2(8x_1 - 1) + \theta_3(3x_2 + 5)$?

For full credit, **you must explain your reasoning**. You will receive partial credit for a correct answer without giving a correct explanation.

- ◯ Yes
- ◯ **No**

Solution: The design matrix $\mathbb{X} \in \mathbb{R}^{n \times 4}$ has linearly dependent columns (the column $8x_1 - 1$ is a linear transformation of the column x_1) and is not a full rank; therefore, $\mathbb{X}^T\mathbb{X}$ is not invertible and a unique solution to the normal equation $\mathbb{X}^T\mathbb{X}\hat{\phi} = \mathbb{X}^T\mathbb{Y}$ does not exist.

(c) [5 Pts] Suppose we have a dataset $\{(x_i, y_i)\}_{i=1}^n$ and we model some particular observation y_i as $\hat{y}_i = f_{\theta_1}(x_i) = \theta_1 \ln(x_i)$. What is the minimizer $\hat{\theta}_1$ that minimizes MSE? Define MSE as $\frac{1}{n}\sum_{i=1}^n (y_i - f_{\theta_1}(x_i))^2$. You must show your work in the box on the following page to get full credit. The critical point you find is guaranteed to be a minimum, so you do **not** need to prove that this point is a minimum.

For full credit, **you must show your work on the following page**. You will receive partial credit for a correct answer without showing a correct approach.

- ◯ $\frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}$
- ◯ $\frac{\sum_{i=1}^n \ln(x_i) y_i}{\sum_{i=1}^n \ln(x_i^2)}$
- ◯ $\frac{\sum_{i=1}^n \ln(x_i) y_i}{\sum_{i=1}^n (\ln(x_i))^2}$
- ◯ $\frac{\sum_{i=1}^n y_i / x_i}{\sum_{i=1}^n x_i^2}$

Solution:

Solution 1:

$$R(\theta_1) = \frac{1}{n}\sum_{i=1}^{n}(\theta_1 \ln(x_i) - y_i)^2$$

$$\frac{\partial}{\partial \theta_1} R(\theta_1) = \frac{2}{n}\sum_{i=1}^{n}\left[(\theta_1 \ln(x_i) - y_i)\ln(x_i)\right] = \frac{2}{n}\sum_{i=1}^{n}\left[\theta_1(\ln(x_i))^2 - y_i \ln(x_i)\right]$$

$$= \frac{2}{n}\left[\theta_1 \sum_{i=1}^{n}(\ln(x_i))^2 - \sum_{i=1}^{n} y_i \ln(x_i)\right]$$

Setting to 0 and solving for θ_1:

$$0 = \frac{2}{n}\left[\theta_1 \sum_{i=1}^{n}(\ln(x_i))^2 - \sum_{i=1}^{n} y_i \ln(x_i)\right] \rightarrow \theta_1 \sum_{i=1}^{n}(\ln(x_i))^2 = \sum_{i=1}^{n} y_i \ln(x_i)$$

$$\theta_1 = \frac{\sum_{i=1}^{n}\ln(x_i)y_i}{\sum_{i=1}^{n}(\ln(x_i))^2}$$

Solution 2:

Alternatively, we can recognize this problem as HW5B Q2 which we derived the solution to the model $y = \theta_1 x$ to be $\theta_1 = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$. $y = \theta_1 \ln(x)$ is just a transformation of x, so we can replace x_i with $\ln(x_i)$, giving us $\theta_1 = \frac{\sum_{i=1}^{n}\ln(x_i)y_i}{\sum_{i=1}^{n}(\ln(x_i))^2}$.

(d) [2 Pts] Given the provided information about the data, model, and loss function, why is the critical point found in Part (c) guaranteed to be a minimum and not a maximum? Explain your reasoning in 1-2 sentences.

Note: In place of a full mathematical proof, a succinct and precise explanation will suffice.

Solution: The critical point of a convex function is always a minimum. Because MSE is a convex function, the critical point occurring at the value of $\hat{\theta}$ calculated in Part (c) is guaranteed to be a minimum.

7 Flight Delays [8 Pts]

Narges is building a flight delay linear regression model using a dataset of historical flight information. Suppose the flight information dataset contains 400 flight records and 10 columns. Of the 10 columns, 1 contains the observed flight delays in minutes, and 9 columns have non-constant numerical features related to the flight (quantitative variables). Assume that we find the least squares estimate for parameter vector θ (with intercept) as $\hat{\theta} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$ for a design matrix $\mathbb{X}$ and true output vector $\mathbb{Y}$.

(a) [1 Pt] What is the shape of the design matrix $\mathbb{X}$?

Shape of $\mathbb{X}$ = ($\square$ $\times$ $\square$)

Solution: Shape of $\mathbb{X} = 400 \times 10$. The matrix contains a bias column.

(b) [1 Pt] What is the shape of the parameter vector θ?

Shape of θ = ($\square$ $\times$ $\square$)

Solution: Shape of $\theta = 10 \times 1$.

(c) [1 Pt] What is the shape of the true output vector $\mathbb{Y}$?

Shape of $\mathbb{Y}$ = ($\square$ $\times$ $\square$)

Solution: Shape of $\mathbb{Y} = 400 \times 1$

(d) [1 Pt] What is the shape of the prediction vector $\hat{\mathbb{Y}}$?

Shape of $\hat{\mathbb{Y}}$ = ($\square$ $\times$ $\square$)

Solution: Shape of $\mathbb{Y} = 400 \times 1$

(e) [2 Pts] What is the residual vector? Write down the equation for calculating the residual vector using $\mathbb{Y}$, $\mathbb{X}$, and $\hat{\theta}$.

Solution: The residual vector is an $n \times 1$ vector, with n being the number of samples, representing the errors in prediction per sample. It can be calculated by $e = \mathbb{Y} - \mathbb{X}\theta$.

(f) [2 Pts] What is **always** true about the residuals in any Ordinary Least Squares (OLS) regression? **Select all that apply.**

- ☐ **They are orthogonal to the column space of the design matrix ($\mathbb{X}$).**
- ☐ Their sum is equal to the mean squared error.
- ☐ Their sum is equal to zero.
- ☐ **They are orthogonal to the vector of predictions ($\hat{\mathbb{Y}}$).**

8 Congratulations [0 Pts]

Congratulations! You have completed the Midterm.

- **Make sure that you have written your student ID number on *each page* of the exam.** You may lose points on pages where you have not done so.
- Also ensure that you have **signed the Honor Code** on the cover page of the exam for 1 point.

[Optional, 0 pts] Draw your favorite Data 100 TA as a panda!

Spring 2023 Data C100/C200 Midterm Reference Sheet

Pandas

Suppose `df` is a DataFrame; `s` is a Series. `import pandas as pd`

Function	Description
`df[col]`	Returns the column labeled `col` from `df` as a Series.
`df[[col1, col2]]`	Returns a DataFrame containing the columns labeled `col1` and `col2`.
`s.loc[rows] / df.loc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their index values.
`s.iloc[rows] / df.iloc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their positions.
`s.isnull() / df.isnull()`	Returns boolean Series/DataFrame identifying missing values
`s.fillna(value) / df.fillna(value)`	Returns a Series/DataFrame where missing values are replaced by `value`
`s.isin(values) / df.isin(values)`	Returns a Series/DataFrame of booleans indicating if each element is in `values`.
`df.drop(labels, axis)`	Returns a DataFrame without the rows or columns named `labels` along `axis` (either 0 or 1)
`df.rename(index=None, columns=None)`	Returns a DataFrame with renamed columns from a dictionary `index` and/or `columns`
`df.sort_values(by, ascending=True)`	Returns a DataFrame where rows are sorted by the values in columns `by`
`s.sort_values(ascending=True)`	Returns a sorted Series.
`s.unique()`	Returns a NumPy array of the unique values
`s.value_counts()`	Returns the number of times each unique value appears in a Series
`pd.merge(left, right, how='inner', on='a')`	Returns a DataFrame joining DataFrames `left` and `right` on the column labeled `a`; the join is of type `inner`
`left.merge(right, left_on=col1, right_on=col2)`	Returns a DataFrame joining DataFrames `left` and `right` on columns labeled `col1` and `col2`.
`df.pivot_table(index, columns, values=None, aggfunc='mean')`	Returns a DataFrame pivot table where columns are unique values from `columns` (column name or list), and rows are unique values from `index` (column name or list); cells are collected `values` using `aggfunc`. If `values` is not provided, cells are collected for each remaining column with multi-level column indexing.
`df.set_index(col)`	Returns a DataFrame that uses the values in the column labeled `col` as the row index.
`df.reset_index()`	Returns a DataFrame that has row index 0, 1, etc., and adds the current index as a column.

Let `grouped = df.groupby(by)` where `by` can be a column label or a list of labels.

Function	Description
`grouped.count()`	Return a Series containing the size of each group, excluding missing values
`grouped.size()`	Return a Series containing size of each group, including missing values
`grouped.mean()`/`grouped.min()`/`grouped.max()`	Return a Series/DataFrame containing mean/min/max of each group for each column, excluding missing values
`grouped.filter(f)` `grouped.agg(f)`	Filters or aggregates using the given function f

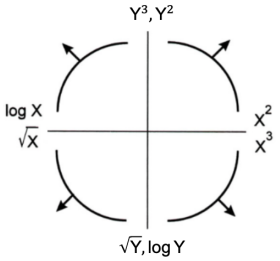
Function	Description
`s.str.len()`	Returns a Series containing length of each string
`s.str[a:b]`	Returns a Series where each element is a slice of the corresponding string indexed from `a` (inclusive, optional) to `b` (non-inclusive, optional)
`s.str.lower()`/`s.str.upper()`	Returns a Series of lowercase/uppercase versions of each string
`s.str.replace(pat, repl)`	Returns a Series that replaces occurences of substrings matching the regex `pat` with string `repl`
`s.str.contains(pat)`	Returns a boolean Series indicating if a substring matching the regex `pat` is contained in each string
`s.str.extract(pat)`	Returns a Series of the first subsequence of each string that matches the regex `pat`. If `pat` contains one group, then only the substring matching the group is extracted

Visualization

Matplotlib: `x` and `y` are sequences of values. `import matplotlib.pyplot as plt`

Function	Description
`plt.plot(x, y)`	Creates a line plot of `x` against `y`
`plt.scatter(x, y)`	Creates a scatter plot of `x` against `y`
`plt.hist(x, bins=None)`	Creates a histogram of `x`; `bins` can be an integer or a sequence
`plt.bar(x, height)`	Creates a bar plot of categories `x` and corresponding heights `height`

Tukey-Mosteller Bulge Diagram.



Seaborn: `x` and `y` are column names in a DataFrame `data`. `import seaborn as sns`

Function	Description
`sns.countplot(data, x)`	Create a barplot of value counts of variable `x` from `data`
`sns.histplot(data, x, stat='count', kde=False)` `sns.displot(x, data, stat='count', rug=True, kde=True)`	Creates a histogram of `x` from `data`, where bin statistics `stat` is one of `'count'`, `'frequency'`, `'probability'`, `'percent'`, and `'density'`; optionally overlay a kernel density estimator. `displot` is similar but can optionally overlay a rug plot
`sns.boxplot(data, x=None, y)` `sns.violinplot(data, x=None, y)`	Create a boxplot of `y`, optionally factoring by categorical `x`, from `data`. `violinplot` is similar but also draws a kernel density estimator of `y`
`sns.rugplot(data, x)`	Adds a rug plot on the x-axis of variable `x` from `data`
`sns.scatterplot(data, x, y)`	Create a scatterplot of `x` versus `y` from `data`
`sns.lmplot(x, y, data, fit_reg=True)`	Create a scatterplot of `x` versus `y` from `data`, and by default overlay a least-squares regression line
`sns.jointplot(x, y, data, kind)`	Combine a bivariate scatterplot of `x` versus `y` from `data`, with univariate density plots of each variable overlaid on the axes; `kind` determines the visualization type for the distribution plot, can be `scatter`, `kde` or `hist`

Regular Expressions

Operator	Description	Operator	Description
`.`	Matches any character except `\n`	`*`	Matches preceding character/group zero or more times
`\`	Escapes metacharacters	`?`	Matches preceding character/group zero or one times
`|`	Matches expression on either side of expression; has lowest priority of any operator	`+`	Matches preceding character/group one or more times
`\d`, `\w`, `\s`	Predefined character group of digits (0-9), alphanumerics (a-z, A-Z, 0-9, and underscore), or whitespace, respectively	`^`, `$`	Matches the beginning and end of the line, respectively
`\D`, `\W`, `\S`	Inverse sets of `\d`, `\w`, `\s`, respectively	`( )`	Capturing group used to create a sub-expression
`{m}`	Matches preceding character/group exactly `m` times	`[ ]`	Character class used to match any of the specified characters or range (e.g. `[abcde]` is equivalent to `[a-e]`)
`{m, n}`	Matches preceding character/group at least `m` times and at most `n` times. If either `m` or `n` are omitted, set lower/upper bounds to 0 and ∞, respectively	`[^ ]`	Invert character class; e.g. `[^a-c]` matches all characters except `a`, `b`, `c`

Modified lecture example for capture groups:

```
import re
lines = '169.237.46.168 - - [26/Jan/2014:10:47:58 -0800] "GET ... HTTP/1.1"'
re.findall(r'\[\d+\/(\w+)\/\d+:\d+:\d+:\d+ .+\]', line) # returns ['Jan']
```

Function	Description
`re.match(pattern, string)`	Returns a match if zero or more characters at beginning of `string` matches `pattern`, else None
`re.search(pattern, string)`	Returns a match if zero or more characters anywhere in `string` matches `pattern`, else None
`re.findall(pattern, string)`	Returns a list of all non-overlapping matches of `pattern` in `string` (if none, returns empty list)
`re.sub(pattern, repl, string)`	Returns `string` that replaces all occurrences of `pattern` with `repl`

Modeling

Concept	Formula	Concept	Formula
Variance, σ_x^2	$\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2$	Correlation r	$r=\frac{1}{n}\sum_{i=1}^{n}\frac{x_i-\bar{x}}{\sigma_x}\frac{y_i-\bar{y}}{\sigma_y}$
L_1 loss	$L_1(y,\hat{y})=\mid y-\hat{y}\mid$	Linear regression estimate of y	$\hat{y}=\theta_0+\theta_1 x$
L_2 loss	$L_2(y,\hat{y})=(y-\hat{y})^2$	Least squares linear regression	$\hat{\theta}_0=\bar{y}-\hat{\theta}_1\bar{x}$ $\quad$ $\hat{\theta}_1=r\frac{\sigma_y}{\sigma_x}$
Empirical risk with loss L	$R(\theta)=\frac{1}{n}\sum_{i=1}^{n}L(y_i,\hat{y}_i)$		

Ordinary Least Squares

Multiple Linear Regression Model: $\hat{\mathbb{Y}}=\mathbb{X}\theta$ with design matrix $\mathbb{X}$, response vector $\mathbb{Y}$, and predicted vector $\hat{\mathbb{Y}}$. If there are p features plus a bias/intercept, then the vector of parameters $\theta=[\theta_0,\theta_1,\ldots,\theta_p]^T\in\mathbb{R}^{p+1}$. The vector of estimates $\hat{\theta}$ is obtained from fitting the model to the sample $(\mathbb{X},\mathbb{Y})$.

Concept	Formula	Concept	Formula
Mean squared error	$R(\theta)=\frac{1}{n}\|\mathbb{Y}-\mathbb{X}\theta\|_2^2$	Normal equation	$\mathbb{X}^T\mathbb{X}\hat{\theta}=\mathbb{X}^T\mathbb{Y}$
		Least squares estimate, if $\mathbb{X}$ is full rank	$\hat{\theta}=(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$
Residual vector, e	$e=\mathbb{Y}-\hat{\mathbb{Y}}$	Multiple R^2 (coefficient of determination)	$R^2=\frac{\text{variance of fitted values}}{\text{variance of } y}$