DS-100 Midterm Exam A

Fall 2018

Name: _____

Email: _____@berkeley.edu

Student ID: _____

Instructions:

- This midterm exam must be completed in the **110 minute** time period ending at **10:00**, unless you have accommodations supported by a DSP letter.
- Note that some questions have 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**.
- When selecting your choices, you must **fully shade** in the box/circle. Check marks will likely be mis-graded.
- You may use a one-sheet (two-sided) study guide.

Honor Code:

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: _____

Syntax Reference

Regular Expressions

"|" matches expression on either side of symbol. Has lowest priority.

"\" match the following character literally.

"?" match preceding literal or sub-expression 0 or 1 times.

"+" match preceding literal or sub-expression *one* or more times.

"*" match preceding literal or sub-expression *zero* or more times.

"." match any character except new line.

"[ ]" match any one of the characters inside, accepts a range, e.g., `"[a-c]"`. All characters inside treated literally.

"( )" used to create a sub-expression.

"{n}" preceding expression repeated n times.

Some useful Python functions and syntax

`re.findall(pattern, st)` returns the list of all sub-strings in `st` that match `pattern`.

`np.random.choice(a, replace, size)` Generates a random sample from `a` consisting of `size` values (with replacement if `replace=True`). `a` can be 1-D array-like or int.

Useful Pandas Syntax

```
df.loc[row_selection, col_list]  # row selection can be boolean
df.iloc[row_selection, col_list] # row selection can be boolean
pd.get_dummies(data) # Convert categorical variable into indicator values
df.groupby(group_columns)[['colA', 'colB']].agg(agg_func)
df.groupby(group_columns)[['colA', 'colB']].filter(filter_func)
```

Variance and Expected Value

The expected value of X is $\mathbb{E}[X] = \sum_{j=1}^{m} x_j p_j$. The variance of X is $Var[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$. The standard deviation of X is $SD[X] = \sqrt{Var[X]}$.

Sampling

1. Below is shown the first 5 rows of the table `restaurants`:

name	cuisine	size
Marufuku	Japanese	2241
Jack in the Box	Fast Food	1592
Thai Basil	Thai	820
Tako Sushi	Japanese	1739
McDonald's	Fast Food	1039

The table `restaurants` contains information regarding different restaurants. The name and cuisine columns contain strings, and the size column contains integers. **The name column is the primary key of the table and therefore contains unique values.** *This is a preview of the first 5 rows of the table. You may assume it has many more rows than what is shown, with the same structure and no missing data.*

Throughout problem 1 and 2, use the keywords and numbers from the answer bank below to fill in the blanks. Note that the same keyword/number can be used multiple times; some keywords may not be used at all. The documentation for some terms appears on the first page of this exam. This answer bank also appears on the back of your answer sheet.

1	2	3	4	5	6
7	8	9	10	11	12
<	>	<=	>=	==	
`restaurants`	`min_rating_df`	**pivot**	**agg**	**loc**	index
iloc	**sort_values**	**size**	**filter**	**groupby**	
barplot	**lineplot**	**jointplot**	**boxplot**		
'name'	'cuisine'	'size'	'rating'		
max	**min**	**median**	**mean**	**first**	**last**
np.array	**sample**	**pd.Series**	**list**	values	
True	False	n	x		

(a) Complete the following function `sample`, which takes in a series and a sample size n and returns a **simple random sample** of n values in that series. Recall that a SRS is drawn without replacement. The result should be a **list** of the n values that are in the sample. For example, `sample(restaurants['name'], 10)` should return a simple random sample of 10 restaurant names with no duplicates. The documentation for `np.random.choice` can be found on the first page of this exam.

```
def sample(series, n):
    return _<i>_______(np.random.choice(_<ii>______._<iii>______,
    size=_<iv>___________, replace=_<v>_____________))
```

Solution:

```
def sample(series, n):
    return list(np.random.choice(series.values,
    size=n, replace=False))
```

(b) Suppose that the probability that Jack in the Box appears in the simple random sample is $\frac{1}{10}$. What is the probability that McDonald's appears in the sample? **There is only one correct answer.**

◯ A. $< \frac{1}{10}$
◯ **B.** $\frac{1}{10}$
◯ C. $> \frac{1}{10}$
◯ D. Not enough information

Solution: Each restaurant has the same chance of being chosen in a simple random sample, so the probability that McDonalds appears is also $\frac{1}{10}$.

(c) What type of sample does `restaurants.groupby('cuisine')['name'].first()` collect? **Select all that apply.**

☐ A. Simple Random Sample
☐ B. Stratified Random Sample
☐ C. Cluster Sample
☐ D. Probability Sample
☐ **E. None of the above**

Solution: This would collect the same sample every time, so this is not a probability sample.

(d) Josh wants to collect a stratified random sample of restaurant names where the strata are the cuisine type, and he wants to collect 2 restaurant names per strata. Complete the following line of code to collect Josh's desired stratified random sample.

```
restaurants._<i>_____(_<ii>_____)[_<iii>_____]._<iv>_____(
lambda x: _<v>_____(_<vi>_____, _<vii>_____))
```

Solution:

```
restaurants.groupby('cuisine')['name'].agg(
lambda x: sample(x, 2))
```

(e) Suppose there are 10 unique cuisine types with 15 fast food restaurants, 20 Japanese Restaurants, and 5 Thai Restaurants. There are 100 restaurants overall in the table with at least 2 restaurants for each cuisine type. What is the probability that McDonald's is picked in Josh's stratified sample from the previous problem?

Solution: We can find the probability that McDonald's does not appear in the stratified sample and subtract this from one. $1 - \frac{14}{15} * \frac{13}{14} = \frac{2}{15}$

(f) Fernando wants to collect a cluster sample, where each cluster is a cuisine type. Suppose Fernando wants to have 2 clusters in his cluster sample. Which of the following lines of code would create Fernando's desired cluster sample? **Select all that apply.** At least one answer is correct. All of the code in all three possible answers is syntactically correct.

☐ A.
```
restaurants[restaurants['cuisine'].isin(
    np.random.choice(restaurants['cuisine'].unique(),
    size=2, replace=True))]['name']
```

☐ **B.**
```
restaurants[restaurants['cuisine'].isin(
    np.random.choice(restaurants['cuisine'].unique(),
    size=2, replace=False))]['name']
```

☐ C.
```
restaurants[restaurants['cuisine'].isin(
    np.random.choice(restaurants['cuisine'].values,
    size=2, replace=False))]['name']
```

Solution:

☐ A. False. Having `replace=True` in `np.random.choice` makes it possible for two of the same cluster to be chosen.

☐ **B. True. By selecting the unique values in the cuisine column, we make sure that each cuisine has a equal chance of being chosen.**

☐ C. False. Having `restaurants['cuisine'].values` makes it possible for multiple cuisines to appear, which means that it is possible for two of the same cluster to be chosen.

(g) With the same assumptions as in part (e), what is the probability that McDonald's appears in Fernando's cluster sample?

Solution: We can find the probability that McDonald's does not appear in the cluster sample and subtract this from one. $1 - \frac{9}{10} * \frac{8}{9} = \frac{2}{10} = \frac{1}{5}$

(h) Manana goes for a third sampling strategy. She decides to take a cluster sample (just like Fernando) of 2 cuisines, but rather than collecting information about every restaurant in those two clusters, she then collects a SRS of one restaurant from each of her two randomly selected clusters, for a total of two restaurants. What type of sample has Manana collected? **Select all that apply.**

☐ A. Simple Random Sample
☐ B. Stratified Random Sample
☐ C. Cluster Sample
☐ **D. Multi-Stage sample**
☐ E. None of the above

Solution: As discussed in discussion 1, this is a multi-stage sample. We are taking a cluster sample and then taking a SRS of each cluster we have chosen.

Pandas

2. For the following problems, suppose we add a new column to our restaurants table which contains the average rating of each restaurant by users of a restaurant review service. **Remember to only use terms from the table in the previous section (or on the back of your answer sheet)!**

name	cuisine	size	rating
Marufuku	Japanese	2241	4.5
Jack in the box	Fast Food	1592	3.4
Thai Basil	Thai	820	4.7
Tako Sushi	Japanese	1739	2.3
McDonald's	Fast Food	1039	3.5

(a) Let a "lowest rank restaurant" be a restaurant that has the lowest rating for a given cuisine. Create a table of all lowest rank restaurants (i.e. one row for each cuisine). Include the restaurant's name, size, and average rating.

```
min_rating_df = restaurants._<i>_____(_<ii>_____)
                            ._<iii>_____(_<iv>_____)
                            ._<v>_____()
```

Solution:

```
min_rating_df = restaurants.sort_values('rating')
                .groupby('cuisine')
                .first()
```

(b) Create a chart that is useful for visualizing the ratings for the lowest rank restaurants.

```
sns._<i>_____(min_rating_df.index, _<ii>_____[_<iii>_____])
```

Solution:

```
sns.barplot(min_rating_df.index, min_rating_df['rating'])
```

(c) Change the ratings of all Japanese restaurants to be 5.

```
restaurants.loc[_<i>_____[_<ii>_____]_<iii>_____ 'Japanese',
    _<iv>_____] = _<v>_____
```

Solution:

```
restaurants.loc[restaurants['cuisine']=='Japanese', 'rating'] = 5
```

(d) Return a list of all cuisines whose restaurants have an average size greater than or equal to 1000.

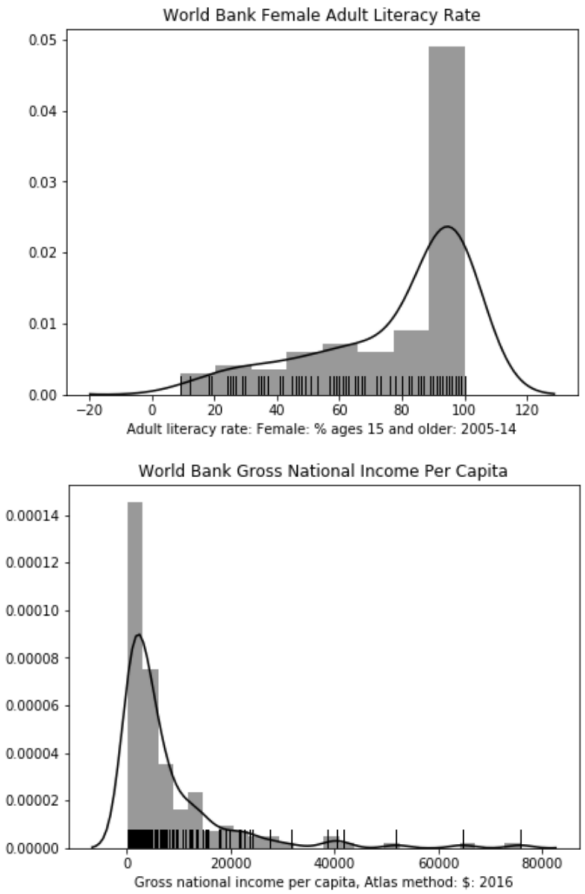
```
list(restaurants._<i>___________(_<ii>___________)
            ._<iii>_____(
            lambda x:x[_<iv>________]._<v>________() >= 1000)
            [_<vi>___________].unique())
```

Solution:

```
restaurants.groupby('cuisine').filter(lambda x: x['size']
.mean() >= 1000)['cuisine'].unique()
```

Visualization

3. Suppose we have the following histograms.



(a) Which of the histograms above is right skewed? **There is only one correct answer.**

◯ A. World Bank Female Adult Literacy Rate

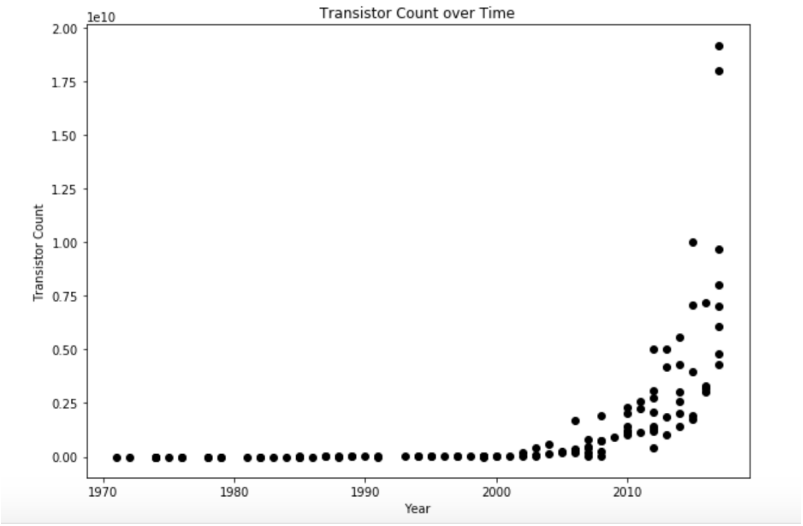
◯ **B. World Bank Gross National Income Per Capita**

(b) Which histogram would look more symmetric with a log transformation applied? **There is only one correct answer.**

◯ A. World Bank Female Adult Literacy Rate

◯ **B. World Bank Gross National Income Per Capita**

4. Consider the scatter plot shown below.



Which of the following transformations would make the relationship between x and y more linear, i.e. if we plotted $f_y(y)$ vs. $f_x(x)$, which would look most linear? **There is only one correct answer.**

- ◯ A. $f_x(x) = x^2$ $f_y(y) = y^2$
- ◯ B. $f_x(x) = log(x)$ $f_y(y) = y$
- ◯ **C.** $f_x(x) = x$ $f_y(y) = log(y)$
- ◯ D. $f_x(x) = log(x)$ $f_y(y) = y^2$

5. For each of the following relationships between x and y, select the appropriate transformation so that the transformed values are linearly related. In other words, select the transformations such that if we plotted $f_y(y)$ vs. $f_x(x)$, we'd expect to get a straight line. **There is only one correct answer in each part.**

(a) $y = ab^x$

- ◯ A. $f_y(y) = log(y)$ $f_x(x) = log(x)$
- ◯ B. $f_y(y) = y$ $f_x(x) = log(x)$
- ◯ **C.** $f_y(y) = log(y)$ $f_x(x) = x$
- ◯ D. $f_y(y) = \frac{1}{y}$ $f_x(x) = \frac{1}{x}$
- ◯ E. The relationship is already linear.

(b) $y = \frac{x}{a+bx}$

- ◯ A. $f_y(y) = \frac{1}{y}$ $f_x(x) = x$
- ◯ B. $f_y(y) = y$ $f_x(x) = \frac{1}{x}$
- ◯ **C.** $f_y(y) = \frac{1}{y}$ $f_x(x) = \frac{1}{x}$
- ◯ D. None of the transformations above create a linear relationship.
- ◯ E. The relationship is already linear.

Regular Expressions

6. Recall that the `re.findall(regex, string)` method returns a list of all matching strings, e.g. `re.findall('cow', 'A cow = cow.')` would return `['cow', 'cow']`.

   For the following regular expression, which of the following strings would result in exactly one match? By one match, we mean the list returned by `findall` is of length 1.

   `'[a-z][aueoi][a-z]+'`

   - ☐ A. `'ba'`
   - ☐ **B. `'bat'`**
   - ☐ **C. `'batch'`**
   - ☐ **D. `'batches'`**
   - ☐ E. `'batches␣of␣bees'`
   - ☐ F. None of the above

7. For the following regular expression, which of the following strings would result in exactly one match?

   `'(burrito|dog){2}'`

   - ☐ **A. `'dogdog'`**
   - ☐ **B. `'burritodog'`**
   - ☐ **C. `'dogburrito'`**
   - ☐ D. `'burrito␣dog'`
   - ☐ E. None of the above

8. How many matches would be returned by the code below (note the **square** brackets!): `re.findall("[Cow|Man]{2}", "CowManCowCowManMan999")`?

   ◯ A. 0 ◯ B. 1 ◯ C. 2 ◯ D. 3 ◯ E. 4 ◯ F. 6 ◯ G. 8 √ **H. 9**
   ◯ I. 12 ◯ J. 17 ◯ K. 18

9. What are the start and end positions for the match returned by `re.search` below? Use Python's 0-indexing and semi-open intervals [a, b) notation. If you believe that no match is returned, answer with the empty interval [0, 0).

   `re.search(r'\..*', 'pic.jpg.*bak')`

   (a) The inclusive start position is: ◯ A. 0 √ **B. 3** ◯ C. 4 ◯ D. 7 ◯ E. 8

   (b) The exclusive end position is: ◯ A. 0 ◯ B. 3 ◯ C. 7 ◯ D. 8 √ **E. 11**