# Midterm Exam Solutions - DSC 80, Spring 2024

**Instructions:**

- This exam consists of 7 questions. A total of 50 points are available.
- Write name in the top right of each page in the space provided.
- Please write neatly in the provided answer boxes. We will not grade work that appears elsewhere.
- Completely fill in bubbles and square boxes.
  - ◯ A bubble means that you should only **select one choice**.
  - ☐ A square box means you should **select all that apply**.
- You may refer to one 8.5" × 11" sheet of notes of your own creation. No other resources or technology (including calculators) are permitted.
- Do not turn the page until instructed to do so.

| | |
|---|---|
| Last name | |
| First name | |
| Student ID number | |
| UCSD email | |
| Name of the person to your left | |
| Name of the person to your right | |
| *All the work on this exam is my own.* **(please sign)** | |

This page is intentionally left blank, but feel free to use it as scratch paper.

Name: ______________________________

**Question 1** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . *12 points*

Fill in Python code below so that the last line of each part evaluates to each desired result using the tables `h`, `o`, and `j` as shown on the Reference Sheet.

(a) (2 points) Find the median duration of outages that happened in the early morning (before 8am).

```
o.loc[ o['time'].dt.hour < 8 , 'duration' ].median()
```

(b) (3 points) A Series containing the mean outage duration for outages that happened on the weekend and outages that happened on weekdays. *Hint:* If `s` is a Series of timestamps, `s.dt.dayofweek` returns a Series of integers where 0 is Monday and 6 is Sunday.

```
(o.assign( is_weekend=o['time'].dt.dayofweek >= 5 )
.groupby( 'is_weekend' )[ 'duration' ].mean() )
```

(c) (4 points) A DataFrame containing the proportion of 4-digit address numbers for each unique street in `h`.

```
def foo(x):
    lengths = x.astype(str).str.len()

    return (lengths == 4).mean()

h.groupby( 'street' ). agg (foo)
```

(d) (3 points) What does the following code compute?

```
a = h.merge(j, left_index=True, right_on='hid', how='left')
a.loc[a['oid'].isna(), 'hid'].shape[0]
```

◯ The number of addresses with exactly one outage.
◯ The number of addresses with at least one outage.
◯ **The number of addresses with no outages.**
◯ The total number of addresses affected by all power outages.
◯ The number of power outages.
◯ The number of power outages that affected exactly one address.
◯ The number of power outages that affected at least one address.
◯ The number of power outages that affected no addresses.
◯ 0
◯ The code will raise an error.
◯ None of the above.

Name: ____________________

**Question 2** .......... *8 points*

Consider the following code:

```
whoa = (h.merge(j, left_index=True, right_on='hid', how='left')
        .merge(o, left_on='oid', right_index=True, how='right')
        .reset_index(drop=True))
```

Consider the following variables:

```
a = j['hid'] <= 50
b = j['hid'] > 50
c = j['oid'] <= 100
d = j['oid'] > 100
e = (j[j['hid'] <= 50]
     .groupby('hid')
     .filter(lambda x: all(x['oid'] > 100))
     ['hid']
     .nunique())
f = (j[j['oid'] <= 100]
     .groupby('oid')
     .filter(lambda x: all(x['hid'] > 50))
     ['oid']
     .nunique())
g = len(set(h.index) - set(j['hid']))
i = len(set(o.index) - set(j['oid']))
```

Write a **single expression** that evaluates to the number of rows in `whoa`. In your code, you may only use the variables `a`, `b`, `c`, `d`, `e`, `f`, `g`, `i` as defined above, arithmetic and bitwise operators (`+`, `-`, `/`, `*`, `&`, `|`), and the `np.sum()` function. **You may not use any other variables or functions.** Your code might not need to use all of the variables defined above.

Show your work in the space below and draw a box around your final answer.

**Solution:** We know that `h` has the numbers 1-50 as unique integers in its index, and `o` has the numbers 1-100 as unique integers in its index. However, the `hid` and `oid` columns in `j` have values outside these ranges. To approach this problem, it's easiest to come up smaller versions of `h`, `j`, and `o`, then perform the join by hand. For example, consider the following example `h`, `j`, and `o` tables:

| hid |
|---|
| 1 |
| 2 |
| 3 |

| hid | oid |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 2 | 10 |
| 2 | 11 |
| 10 | 3 |
| 11 | 3 |

| oid |
|---|
| 1 |
| 2 |
| 3 |

In this example, `whoa` would look like the following (omitting other columns besides `hid` and `oid` for brevity):

| hid | oid |
|---|---|
| 1 | 1 |
| 2 | 1 |
| NaN | 2 |
| NaN | 3 |

There are 3 cases where rows will be kept for `whoa`:

1. When both `hid` and `oid` match in the three tables (when `a` and `c` are both true). In the example above, this corresponds to the first two rows of `whoa`.
2. When the `oid` in `o` doesn't appear at all in `j` (calculated by `i`). In the example above, this corresponds to the third row of `whoa`.
3. When the `oid` in `o` does appear in `j`, but none of the `hid` values appear in `h` (calculated by `f`). In the example above, this corresponds to the fourth row of `whoa`.

Therefore, the number of rows in `whoa` is:

```
np.sum(a & c) + f + i
```

Name: ____________________

**Question 3** .................................................................... *7 points*

Consider the following code which defines a DataFrame named `df`:

```
def hour(df):       return df.assign(hour=df['time'].dt.hour)
def is_morning(df): return df.assign(is_morning=df['hour'] < 12)

df = (h.merge(j, left_index=True, right_on='hid', how='inner')
      .merge(o, left_on='oid', right_index=True, how='inner')
      .reset_index(drop=True)
      .pipe(hour)
      .pipe(is_morning))
```

The first few rows of `df` are shown below.

| | number | street | hid | oid | time | duration | hour | is_morning |
|---|---|---|---|---|---|---|---|---|
| **0** | 7370 | Torrey Pines Rd | 1 | 60 | 2024-04-09 13:14:00 | 70 | 13 | False |
| **1** | 4758 | Mission Blvd | 32 | 60 | 2024-04-09 13:14:00 | 70 | 13 | False |

Suppose we define a DataFrame `p` and functions `a`, `b`, `c`, and `d` as follows:

```
p = df.pivot_table(index='street', columns='hour', values='duration',
                   aggfunc='count', fill_value=0)

def a(n):    return p[n].sum()
def b(s):    return p.loc[s].sum()
def c():     return p.sum().sum()
def d(s, n): return p.loc[s, n]
```

Write a single expression to compute each of the probabilities below. **Your code can only use the functions `a`, `b`, `c`, `d`, and arithmetic operators (`+`, `-`, `/`, `*`).**

(a) (2 points) The probability that a randomly selected row from `df` has the street `Mission Blvd`.

**Solution:**

```
b('Mission Blvd') / c()
```

(b) (2 points) The probability that a randomly selected row from `df` has the street `Gilman Dr` given that its hour is 21.

**Solution:**

```
d('Gilman Dr', 21) / a(21)
```

(c) (3 points) The probability that a randomly selected row from `df` either has the street `Mission Blvd` or the hour 12.

**Solution:**

```
(b('Mission Blvd') + a(12) - d('Mission Blvd', 12)) / c()
```

Name: ______________________________

**Question 4** ........................................................................ *6 points*

(a) (3 points) Consider the following pivot table created using the `df` table from Question 3 which shows the average duration of power outages split by street name and whether the outage happened before 12pm.

| is_morning | False | True |
|---|---|---|
| **street** | | |
| **El Cajon Blvd** | 48.00 | 57.58 |
| **Gilman Dr** | 44.85 | 59.29 |
| **La Jolla Village Dr** | 40.62 | 54.56 |
| **Mission Blvd** | 44.86 | 44.93 |
| **Torrey Pines Rd** | 52.78 | 55.04 |

Given only the information in this pivot table and the Reference Sheet, is it possible to observe Simpson's paradox for this data if we don't split by street? In other words, is it possible that the average duration of power outages before 12pm is lower than the average duration of power outages after 12pm?

◯ **Yes**
◯ No
◯ Need more information to determine

(b) (3 points) Consider the following pivot table created using the `o` table which shows the average duration of power outages split by whether the outage happened on the weekend and whether the outage happened before 12pm.

| is_morning | False | True |
|---|---|---|
| **is_weekend** | | |
| **False** | 43.40 | 53.09 |
| **True** | 51.67 | 58.64 |

Given only the information in this pivot table and the Reference Sheet, is it possible to observe Simpson's paradox for this data if we don't split by `is_weekend`? In other words, is it possible that the average duration of power outages before 12pm is lower than the average duration of power outages after 12pm?

◯ Yes
◯ **No**
◯ Need more information to determine

**Solution:** By the same logic as the previous part, the overall average when `is_morning=True` must be between (53.09, 58.64). The overall average when `is_morning=False` must be between (43.40, 51.67). This implies that Simpson's paradox cannot happen, since the overall average when `is_morning=False` will never be greater than the overall average when `is_morning=True`.

Name: ______________________________

**Question 5** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . *9 points*

Praveen wants to answer the following questions using hypothesis tests on the power outages data, so he adds a `hour` and `is_morning` column to the `o` DataFrame. The first few rows of the new `o` DataFrame are shown below. For this problem, assume that some of the `duration` values are missing.

| oid | time | duration | hour | is_morning |
|---|---|---|---|---|
| 1 | 2024-04-07 03:21:00 | 3 | 3 | True |
| 2 | 2024-04-20 16:35:00 | 70 | 16 | False |

For each test, select the **one** correct procedure to simulate a single sample under the null hypothesis, and select **all** test statistics that can be used for the hypothesis test among the choices given.

(a) (3 points) Null: Every hour of the day (0, 1, 2, etc.) has an equal probability of having a power outage. Alternative: At least one hour is more prone to outages than others.

Simulation procedure:

- ○ `np.random.multinomial(100, [1/2] * 2)`
- ○ **`np.random.multinomial(100, [1/24] * 24)`**
- ○ `o['hour'].sample(100)`
- ○ `np.random.permutation(o['duration'])`

Test statistic:

- ☐ Difference in means
- ☐ Absolute difference in means
- ■ **Total variation distance**
- ■ **K-S test statistic**

(b) (3 points) Null: The proportion of outages that happen in the morning is the same for both recorded durations and missing durations.

Alternative: The outages are more likely to happen in the morning for missing durations than for recorded durations.

Simulation procedure:

- ○ `np.random.multinomial(100, [1/2] * 2)`
- ○ `np.random.multinomial(100, [1/24] * 24)`
- ○ `o['hour'].sample(100)`
- ○ **`np.random.permutation(o['duration'])`**

Test statistic:

- ■ **Difference in means**
- ☐ Absolute difference in means
- ☐ Total variation distance
- ☐ K-S test statistic

(c) (3 points) Null: The distribution of hours is the same for both recorded durations and missing durations. Alternative: The distribution of hours is different for recorded durations and missing durations.

Simulation procedure:

- ○ `np.random.multinomial(100, [1/2] * 2)`
- ○ `np.random.multinomial(100, [1/24] * 24)`
- ○ `o['hour'].sample(100)`
- ○ **`np.random.permutation(o['duration'])`**

Test statistic:

- ☐ Difference in means
- ■ **Absolute difference in means**
- ■ **Total variation distance**
- ■ **K-S test statistic**

**Question 6** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ***8 points***

After loading in the DataFrame `df` from Question 3, Sam realizes that his puppy Bentley ate some of his data! The first few rows of `df` are shown below for convenience.

| | number | street | hid | oid | time | duration | hour | is_morning |
|---|---|---|---|---|---|---|---|---|
| **0** | 7370 | Torrey Pines Rd | 1 | 60 | 2024-04-09 13:14:00 | 70 | 13 | False |
| **1** | 4758 | Mission Blvd | 32 | 60 | 2024-04-09 13:14:00 | 70 | 13 | False |

(a) (2 points) Suppose that Sam sorted `df` by `is_morning`, and then Bentley ate the first five values from the `duration` column. What is the missingness mechanism for the `duration` column?

- ◯ Missing by design
- ◯ MNAR
- ◯ MAR on `is_morning` only
- ◯ MAR on `is_morning` and `hour` only
- ◯ **MAR on `is_morning`, `hour`, and `time` only**
- ◯ MCAR

(b) (6 points) Sam believes that the data are MAR on `hour` only, so he decides to use probabilistic imputation to fill in the missing values. He uses the following code copied from Lecture 8 (line numbers shown in parentheses):

```
(1)  def impute(s):
(2)      s = s.copy()
(3)      n = s.isna().sum()
(4)      fill = np.random.choice(s.dropna(), n)
(5)      s[s.isna()] = fill
(6)      return s
(7)  df.groupby('hour')['duration'].transform(impute)
```

i. Even though this code is copied from lecture, it can raise an error on Sam's data if a certain condition is met. Which of these, if true, would cause to code to error?

- ◯ The missing values in `duration` are actually NMAR.
- ◯ The missing values in `duration` are actually MAR on `street`, not `hour`.
- ◯ There are no missing values in `duration`.
- ◯ At least one `hour` value doesn't have any missing `duration` values.
- ◯ **At least one `hour` value only has missing `duration` values.**
- ◯ There are no rows where `hour == 12`.

ii. Which line in the code would raise the error?

- ◯ Line 1
- ◯ Line 2
- ◯ Line 3
- ◯ **Line 4**
- ◯ Line 5
- ◯ Line 6
- ◯ Line 7

Name: ____________________

**Question 7** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ***0 points***

Optional: Draw a Picture About UCSD Data Science (or use this page for scratch work)

This page is intentionally left blank, but feel free to use it as scratch paper.