Discussion 4 Solutions

Note: Starting this week, I'm going to release solutions as an answer document instead of the filled worksheet to have space to explain everything, but just FYI: all of the following is from practice.dsc80.com — the purpose of this is just so you don't have to cross-reference anything yourself!

FA23 Midterm Problem 3

Problem

- A. The researchers chose the 30 donkeys with the largest `'Weight'` values to reweigh.
- B. The researchers drew 30 donkeys uniformly at random without replacement from the donkeys with `BCS` score of 4 or greater.
- C. The researchers set `i` as a number drawn uniformly at random between 0 and 514, then reweighed the donkeys in `donkeys.iloc[i:i+30]`.
- D. The researchers reweighed all the donkeys, but deleted all the values in `'WeightAlt'` except for the 30 lowest values.
- E. The researchers split up the donkeys into the 6 different age groups, then sampled 5 donkeys uniformly at random without replacement within each age group.

Solution

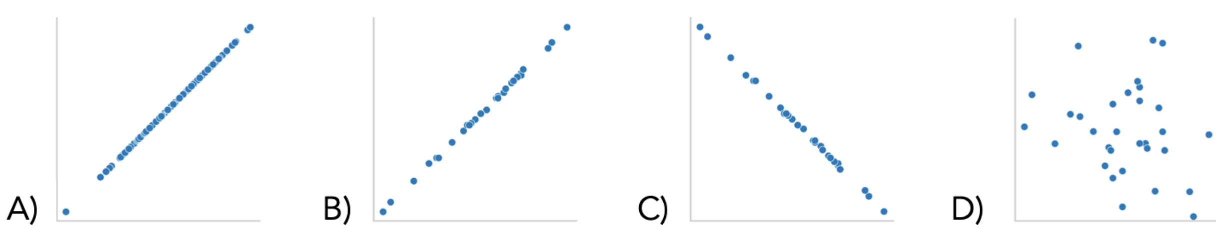
- A. **Missing at random**. This means missing values depend on another column in the DataFrame. In this case, the missing values of `'WeightAlt'` depend on the `'Weight'` column since we select the 30 largest.
- B. **Missing at random**. This means missing values depend on another column in the DataFrame. In this case, the missing values of `'WeightAlt'` depend on the `'BCS'` column since we choose from those with a score of 4 or greater.
- C. **Missing completely at random or, possibly, Missing at Random**. The argument for MAR is as follows: this means missing values depend on another column in the DataFrame. The missing values depend on the index since index 0 can only be selected if `i = 0`, but index 29 could be chosen if `i` is any value between 0 and 29, so it has a higher probability of being chosen. The original solution was MCAR as we did not account for edge case of `i` being small, but it is technically MAR. Credit was given for either answer.

- D. **Not missing at random**. This means missing values depend on the column they're missing from. The missing values here are all values that are not the 30 lowest in `'Weight'`, and so they depend on the column itself.
- E. **Missing completely at random or Missing at random**. If the data was assumed to be evenly distributed, then the data is missing completely at random since the six age groups would all be chosen from uniformly. However, if the data was assumed to possibly have skewed age data, then samples from small sample size age groups had a higher probability of being chosen than those of large sample size age group. Credit was given for either answer.

NOTE: Despite the fact that we accepted multiple answers for a couple of these, you should make *as few assumptions about the data as possible* to get your solutions — but if you're unsure, feel free to ask!

Problem

For this next question, assume that the researchers chose the 30 donkeys to reweigh by drawing a simple random sample of 30 underweight donkeys: donkeys with BCS values of 1, 1.5, or 2. The
researchers weighed these 30 donkeys one day later and stored the results in 'WeightAlt'. Which of the following shows the scatter plot of 'WeightAlt' - 'Weight' on the y-axis and 'Weight' on
the x-axis? Assume that missing values are not plotted.

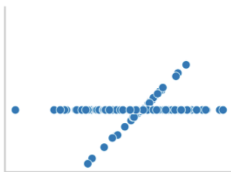


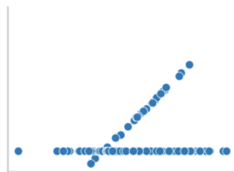
Solution

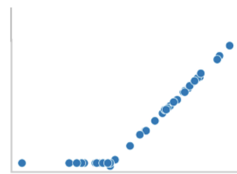
We are measuring the difference in weight from just one day on the y-axis, which means we can't expect any noticeable pattern of weight gain or loss no matter the original weight of the donkey. Therefore, a random scatterplot makes sense. Options A through C all suggest that the single-day weight change correlates with the starting weight, which is not a good assumption.

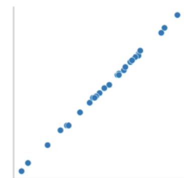
Problem

Suppose we use mean imputation to fill in the missing values in `'WeightAlt'`. Select the scatter plot `'WeightAlt'` on `'Weight'` after imputation.

A) 

B) 

C) 

D) 

Solution

Note we are now plotting `'Weight'` on the y-axis, not the difference of `'WeightAlt' - 'Weight'`. Therefore, it makes sense that we would have 30 data points with a positive slope as the initial weight and re-weight are likely very similar.

Then, mean imputation is the process of filling in missing values with the average of the non-missing values. Therefore, all missing values will be the same, and should be at the center of the sloped line since the line is roughly evenly distributed.

FA23 Final Problem 3

The `bus` table (left) records bus arrivals over 1 day for all the bus stops within a 2 mile radius of UCSD. The data dictionary (right) describes each column.

	time	line	stop	late
0	12pm	201	Gilman Dr & Mandeville Ln	-1.1
1	1:15pm	30	Gilman Dr & Mandeville Ln	2.8
2	11:02am	101	Gilman Dr & Myers Dr	-0.8
3	8:04am	202	Gilman Dr & Myers Dr	NaN
4	9am	30	Gilman Dr & Myers Dr	-3.0

`time`	Time of arrival (`str`). Note that the times are inconsistently entered (e.g. `12pm` vs. `1:15pm`).
`line`	Bus line (`int`). There are multiple buses per bus line each day.
`stop`	Bus stop (`str`).
`late`	The number of minutes the bus arrived after its scheduled time. Negative numbers mean that the bus arrived early (`float`). Some entries in this column are missing.

For each of the following questions, select the correct procedure to simulate a single sample under the null hypothesis, and the correct test statistic for the hypothesis test. Assume that the `time` column of the bus DataFrame has already been parsed into timestamps.

Problem

Are buses equally likely to be early or late? *Note: while the problem says there is only one solution, post-exam two options for the test statistic were given credit. Pick one of the two.*

Simulation procedure:

- A. `np.random.choice([-1, 1], bus.shape[0])`
- B. `np.random.choice(bus['late'], bus.shape[0], replace = True)`
- C. Randomly permute the `'late'` column

Test statistic:

- A. Number of values below 00
- B. `np.mean`
- C. `np.std`
- D. TVD
- E. K-S statistic

Solution

Simulation procedure: The sample we have here is something like 152 early buses, 125 late buses (these numbers are made up – in practice, these two numbers need to add to `bus.shape[0]`). The question is whether this sample looks like it was drawn from a population that is 50-50 (an equal number of early and late buses), which makes this a hypothesis test. In terms of examples from class, this most closely resembles the very first hypothesis testing example we looked at – the "coin flipping" example.

`np.random.choice([-1, 1], bus.shape[0])` will return an array of length `bus.shape[0]`, where each element is equally likely to be either `-1` (late) or `1` (early). (Note that we could also take `-1` to mean early and `1` to mean late – it doesn't really matter.)

Test statistic: Each time we simulate an arrays of `-1`s and `1`s, we'd like to compute a statistic
that helps us differentiate between the number of late (
`-1`) and the number of early (`1`) simulated buses. The number of values below 0 will give us the number of late simulated buses, so we could use that. ~~The mean of the `-1`s and `1`s will give us a value that is negative if there were more late buses and positive if there were more early buses, so we could use that too.~~

NOTE: this problem accepted np.mean for the test statistic, but I am pretty confident that this won't work with some pretty simple assumptions about the data, and I'll see about getting that fixed

Problem

Is the `'late'` column MAR dependent on the `'line'` column?

Simulation procedure:

- `np.random.choice([-1, 1], bus.shape[0])`
- `np.random.choice(bus['late'], bus.shape[0], replace = True)`
- Randomly permute the `'late'` column

Test statistic:

- Absolute difference in means
- Absolute difference in proportions
- TVD
- K-S statistic

Solution

Answer: Simulation procedure: Randomly permute the `'late'` column; Test statistic: TVD

Simulation procedure: To determine if `'late'` is missing at random dependent on the `'line'` column, we conduct a permutation test and compare (1) the distribution of the `'line'` column when the `'late'` column is missing to (2) the distribution of the `'line'` column when the `'late'` column is not missing to see whether they're significantly different. If the distributions are indeed significantly different, then it is likely that the `'late'` column is MAR dependent on `'line'`.

Test statistic: Since we are comparing the distributions of *categorical* data (`'line'` is categorical) for our permutation test, Total Variation Distance is the best test statistic to use.

Discussion 5 Solutions

NOTE: All of these solutions are available on practice.dsc80.com — this is just so you can see everything on one place.

FA23 Midterm Problem 4

Problem

To generate a single sample under his null hypothesis, Alan should

A. Resample 744 donkeys with replacement from `donkeys`.

B. Resample 372 donkeys with replacement from `donkeys` with `'BCS'` < 3, and another 372 donkeys with `'BCS'` >= 3.

C. Randomly permute the `'Weight'` column.

Solution

Answer: C

The null hypothesis is "Donkeys with `'BCS'` >= 3 have the **same** `'Weight'` values on average compared to donkeys that have `'BCS'` < 3". Under the null hypothesis, we should have similar results with a shuffled dataset.

Options A and B shuffle with replacement (bootstrapping), while option C shuffles without replacement (permutation is done without replacement). Bootstrapping is generally used to estimate confidence intervals, while permutation tests are a kind of hypothesis test. In this case, we are performing a hypothesis test, so we want to permute the `'Weight'` column.

Problem

Doris wants to use multiple imputation to fill in the missing values in `'WeightAlt'`. She knows that `'WeightAlt'` is MAR conditional on `'BCS'` and `'Age'`, so she will perform multiple imputation conditional on `'BCS'` and `'Age'` - each missing value will be filled in with values from a random `'WeightAlt'` value **from a donkey with the same `'BCS'` and `'Age'`**. Assume that all `'BCS'` and `'Age'` combinations have observed `WeightAlt` values. Fill in the blanks in the code below to estimate the median of `'WeightAlt'` using multiple imputation conditional on `'BCS'` and `'Age'` with 100 repetitions. A function `impute` is also partially filled in for you, and you should use it in your answer.

Solution

```
def impute(col):
    col = col.copy()
    n = col.isna().sum()
    fill = np.random.choice(col.dropna(), n)
    col[col.isna()] = fill
    return col
results = []
for i in range(100):
    imputed = (donkeys.groupby(['BCS', 'Age'])['WeightAlt'].transform(impu
    results.append(imputed.median())
```

We start with the bottom five blanks as we are not sure what the parameter of `impute(col)` is until we write the function call first. We see that we are using a loop, and seeing that we are doing multiple imputation with 100 reputations, we can fill in `range(100)`. We then define the variable `imputed`, which we can see from the last line of code that calls `imputed.median()` should be a list of `'WeightAlt'` that has imputed values. Since we want to make our imputation conditional on `'BCS'` and `'Age'`, we can fill in the next blank with a `groupby` method and pass in the list of columns we want - `['BCS', 'Age']`. We can see we have then selected the `'WeightAlt'` column in the problem, and so we need to use our `impute` function on that series. We can do so with a `transform` method and then pass in `impute`. Note this can also be done with `apply` and receive credit, but this is our solution.

Now, we can define the `impute` function to impute missing values from `col`. Since we have already aggregated on `['BCS', 'Age']`, we know that our given `col` has samples all of the same `'BCS'` and `'Age'` values. Therefore, to impute as defined in the question, we just need to fill in `NaN` values with any other value from `col`, chosen at random. We can see we will use

`np.random.choice`, which takes in its first parameter possible choices in a list, and in its second parameter the number of choices to make. The number of choices to make we can define as `n`, which is the number of `NaN` values. This is found with `col.isna().sum()`. Then our possible choices are any non-`NaN` values in `col`, which we can use `col.dropna()` to find. Finally, we fill in the `NaN` values in `col` by masking for the `NaN` indices with `col[col.isna()]`, and set it equal to our `fill` values. That will successfully impute values into our `col` and we can then return it.

WI23 Final Problem 1

Problem

The DataFrame `sat` contains one row for **most** combinations of `"Year"` and `"State"`, where `"Year"` ranges between `2005` and `2015` and `"State"` is one of the 50 states (not including the District of Columbia).

The other columns are as follows:

- `"# Students"` contains the number of students who took the SAT in that state in that year.
- `"Math"` contains the mean math section score among all students who took the SAT in that state in that year. This ranges from 200 to 800.
- `"Verbal"` contains the mean verbal section score among all students who took the SAT in that state in that year. This ranges from 200 to 800. (This is now known as the "Critical Reading" section.)

The first few rows of `sat` are shown below (though `sat` has many more rows than are pictured here).

	Year	State	# Students	Math	Verbal
0	2014	Washington	41277	519	510
1	2013	Arizona	22283	529	522
2	2006	Kansas	2545	591	582
3	2011	North Dakota	219	612	586
4	2009	New Mexico	2209	548	553

The data description stated that there is one row in `sat` for ***most*** combinations of `"Year"` (between `2005` and `2015`, inclusive) and `"State"`. This means that for most states, there are 11 rows in `sat` — one for each year between 2005 and 2015, inclusive.

It turns out that there are 11 rows in `sat` for all 50 states, except for one state. Fill in the blanks below so that `missing_years` evaluates to an **array**, sorted in any order, containing the years for which that one state does not appear in `sat`.

Solution

```
state_only = sat.groupby("State").filter(lambda df: df.shape[0] < 11)
merged = sat["Year"].value_counts().to_frame().merge(
    state_only, left_index=True, right_on='Year', how='left'
) # an outer merge also works!
missing_years = merged[merged['# Students'].isna()]['Year'].to_numpy()
```

Blank A

The initial step (in the `state_only` variable) involves identifying the state that has fewer than 11 records in the dataset. This is achieved by the lambda function `lambda df: df.shape[0] < 11`, leaving us with records from only the state that has missing data for certain years.

Blank B

Next, applying `.value_counts()` to `sat["Year"]` produces a Series that enumerates the total occurrences of each year from 2005 to 2015. Converting this Series to a DataFrame with `.to_frame()`, we then merge it with the `state_only` DataFrame. This merging results in a DataFrame (merged) where the years lacking corresponding entries in `state_only` are marked as NaN.

Blank C

Finally, the expression `merged[merged['# Students'].isna()]['Year']` in `missing_years` identifies the specific years that are absent for the one state in the sat dataset. This is determined by selecting years in the merged DataFrame where the `"# Students"` column has NaN values, indicating missing data for those years.

Problem

The following DataFrame contains summary statistics for all SAT takers in New York and Texas from 2005 to 2015. Suppose we want to run a statistical test to assess whether the distributions of the number of students between 2005 and 2015 in New York and Texas are significantly different.

State	mean	median	std
New York	157950.818182	157989.0	3430.986500
Texas	155035.909091	148102.0	22509.092685

Given the information in the above DataFrame, which test statistic is **most likely** to yield a significant difference?

- $mean\,number\;of\;students\;in\,Texas\;-\;mean\,number\;of\;students\;in\;New\;York$
- $|\,mean\,number\;of\;students\;in\,Texas\;-\;mean\,number\;of\;students\;in\;New\;York\,|$
- $|\,median\,number\;of\;students\;in\,Texas\;-\;median\,number\;of\;students\;in\;New\;York\,|$
- The Kolmogorov-Smirnov statistic

Solution

Answer: The Kolmogorov-Smirnov statistic

Here, the means and medians of the two samples are similar, so their observed difference in means and observed difference in medians are both small. This means that a permutation test using either one of those as a test statistic will likely fail to yield a significant difference. However, the standard deviations of both distributions are quite different, which means the shapes of the distributions are quite different. The Kolmogorov-Smirnov statistic measures the distance between two distributions by considering their entire shape, and since these distributions have very different shapes, they will likely have a larger Kolmogorov-Smirnov statistic than expected under the null.