Discussion 2

DSC 80

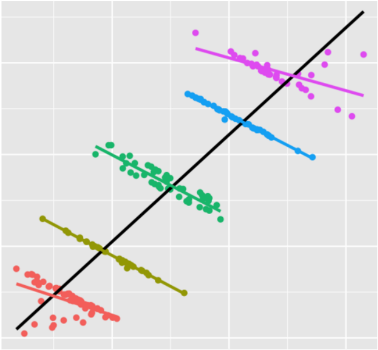
2024-04-11

1. WI24 Midterm Problem 3
2. FA23 Midterm Problem 1
3. FA23 Final Problem 1

Section 1

WI24 Midterm Problem 3

Simpson's Paradox



Credit: Skew the Script

Data

	Beagle		Cocker Spaniel	
	Mean Weight	**Count**	**Mean Weight**	**Count**
District 1	25	3	20	2
District 2	45	1	x	y

This table gives us information about a group of dogs, grouped by breed and location.

Part 1

What is the mean weight of all beagles in the table above, across both districts?

- This should be pretty straightforward – we just have to take a weighted mean

Part 1

What is the mean weight of all beagles in the table above, across both districts?

- This should be pretty straightforward – we just have to take a weighted mean
- $\frac{25 \cdot 3 + 45 \cdot 1}{3+1} = \frac{120}{4} = 30$

Part 1

What is the mean weight of all beagles in the table above, across both districts?

- This should be pretty straightforward – we just have to take a weighted mean
- $\frac{25 \cdot 3 + 45 \cdot 1}{3+1} = \frac{120}{4} = 30$
- This is going to be important for the next part!

Part 2

Notice that the table above has two unknowns, x and y Find positive integers x and y such that the mean weight of all beagles is equal to the mean weight of all cocker spaniels, where x is as small as possible.

Basically: we're trying to induce the pattern that makes Simpson's paradox occur!

- First, given the mean weight of the two cocker spaniels in District 1, what can we say about the possible values of x?

Part 2

Notice that the table above has two unknowns, x and y Find positive integers x and y such that the mean weight of all beagles is equal to the mean weight of all cocker spaniels, where x is as small as possible.

Basically: we're trying to induce the pattern that makes Simpson's paradox occur!

- First, given the mean weight of the two cocker spaniels in District 1, what can we say about the possible values of x?
- So if x has to be as small as possible, what's the first value we might try?

Part 2 (cont.)

And now, it's time for algebra.

$$\frac{20 \cdot 2 + 31y}{2 + y} = 30$$

$$40 + 31y = 60 + 30y \implies y = 20$$

So, the smallest possible x that could raise the mean weight of all cocker spaniels to 30 has an integer solution for y, and there's our answer.

We can now see that Simpson's paradox is occurring between beagles and cocker spaniels!

`The average score on this problem was 51%.`

Bonus

Is it possible for there to be a value of x and y such that Simpson's paradox happens in two different groupings (district and breed)?

Why/why not?

Section 2

FA23 Midterm Problem 1

FA23 Midterm Problem 1

	date	name	food	weight
0	2023-01-01	Sam	Ribeye	0.20
1	2023-01-01	Sam	Pinto beans	0.10
2	2023-01-01	Lauren	Mung beans	0.25
3	2023-01-02	Lauren	Lima beans	0.30
4	2023-01-02	Sam	Sirloin	0.30

This DataFrame contains information about people's daily food intake, with `weight` measured in kilograms. (Apparently, Sam eats steak every night.)

Problem

Find the total kg of food eaten for each day and each person in `df` as a Series.

- Even without the blanks, you should be able to guess that this question involves a `groupby` call...

Problem

Find the total kg of food eaten for each day and each person in `df` as a Series.

- Even without the blanks, you should be able to guess that this question involves a `groupby` call. . .
- . . . and that "total kg" means that we want to sum the `weight` column in each group.

Solution

```
df.groupby(['date', 'name'])['weight'].sum()
```

People who don’t eat beans

Find all the unique people who did not eat any food containing the word “beans”.

The blanks for this question are kind of a hint – we know we have to use `groupby`, and then pass the function `foo` *as an argument* to a `groupby` function.

Hint: If `food` is a Series of strings, `food.str.contains(<str>)` returns a Series of booleans, corresponding to which elements contain the substring.

- What function might we pass `foo` into?

People who don’t eat beans

Find all the unique people who did not eat any food containing the word “beans”.

The blanks for this question are kind of a hint – we know we have to use `groupby`, and then pass the function `foo` *as an argument* to a `groupby` function.

Hint: If `food` is a Series of strings, `food.str.contains(<str>)` returns a Series of booleans, corresponding to which elements contain the substring.

- What function might we pass `foo` into?
- `.filter(foo)` returns a subset of the DataFrame with only groups for which function `foo` returns `True`

People who don't eat beans

Find all the unique people who did not eat any food containing the word "beans".

The blanks for this question are kind of a hint – we know we have to use `groupby`, and then pass the function `foo` *as an argument* to a `groupby` function.

Hint: If `food` is a Series of strings, `food.str.contains(<str>)` returns a Series of booleans, corresponding to which elements contain the substring.

- What function might we pass `foo` into?
- `.filter(foo)` returns a subset of the DataFrame with only groups for which function `foo` returns `True`
- How do we structure the function `foo` (what sort of value is x)?

Solution

```python
def foo(x):
    return x['food'].str.contains('beans').sum() == 0

df.groupby('name').filter(foo)['name'].unique()
```

Notes:

- The function you pass into `.filter()` should take in a DataFrame, and return a boolean of whether you want to keep this group.
- If you want to do more complicated processing on `GroupBy` objects, there's an `apply` function.

`The average score on this problem was 39%.`

Section 3

FA23 Final Problem 1

Data

	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.

	line	stop	next
0	201	Gilman Dr & Mandeville Ln	VA Hospital
1	201	VA Hospital	La Jolla Village Dr & Lebon Dr
2	30	VA Hospital	Villa La Jolla Dr & Holiday Ct
3	30	UTC	NaN

`line` Bus line (`int`).

`stop` Bus stop (`str`).

`next` The next bus stop for a particular bus line (`str`). For example, the first row of the table shows that after the `201` stops at `Gilman Dr & Mandeville Ln`, it will stop at the `VA Hospital` next. A missing value represents the end of a line.

Next Stop

Compute the number of buses in bus whose next stop is `UTC`.

```
x = stop.merge(_______, on = ________, how = ________)
x[_______].shape[0]
```

Solution

```
x = stop.merge(bus, on = ['line', 'stop'], how = 'inner')
x[x['next'] == 'UTC'].shape[0]
```

- A `right` merge would work the exact same – why?
- We could even do a `left` merge, if we filter the outcome more!

Two Stops Away

Compute the number of unique pairs of bus stops that are exactly two stops away from each other.

For example, if you only use the first four rows of the stop table, then your code should evaluate to the number 2, since you can go from `'Gilman Dr & Mandeville Ln'` to `'La Jolla Village Dr & Lebon Dr'` and from `'Gilman Dr & Mandeville Ln'` to `'Villa La Jolla Dr & Holiday Ct'` in two stops.

Hint: The `suffixes = (1, 2)` argument to merge appends a 1 to column labels in the left table and a 2 to column labels in the right table whenever the merged tables share column labels.

Solution

We can merge a DataFrame with itself!

```
m = stop.merge(stop, left_on = 'next', right_on = 'stop',
  how = 'inner', suffixes=(1, 2))
(m[['stop1', 'next2']].drop_duplicates().shape[0])
```