Lecture 14 – Midterm Review

DSC 10, Spring 2023

Announcements

- The Midterm Exam is **this Friday during lecture**. See **this post** for lots of details, including what is covered, what to bring, and how to study.
- The Midterm Project is due on **Tuesday 5/9 at 11:59PM**. Only one partner needs to submit.
- If at least 75% of the class fills out the **Mid-Quarter Survey** by **12PM this Friday**, then everyone will receive 2% of extra credit on the Midterm Exam.
    - We're at 56% as of this morning.
- Homework 2 scores have been released and the Grade Report has been updated; see **this post** for details.

Agenda

- In the 12-12:50PM lecture section, we will cover tricky problems from the Fall 2022 Midterm.
- In the 1-1:50PM lecture section, we will cover tricky problems from the Winter 2023 Midterm.
- Both times, we won't write any code, since you can't run code during the exam. Instead, we'll try to think like the computer ourselves.
- These annotated slides will be posted after both lecture sections are over.
- **Try the problems with me!**

Fall 2022 Midterm

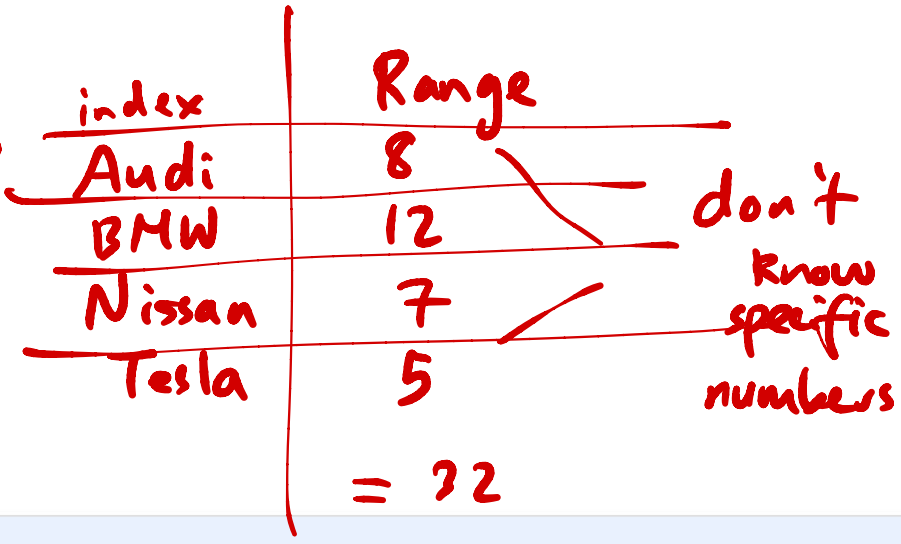
Access the exam here. Make sure to read the data info sheet at the top before starting.

Problem 4

Suppose we've run the following line of code.

```
counts = evs.groupby("Brand").count()
```

DataFrame



Problem 4.1

What value does `counts.get("Range").sum()` evaluate to?

Click to view the solution.

32

Problem 4.2

What value does `counts.index[3]` evaluate to?

Click to view the solution.

"Tesla"

Problem 6

The DataFrame below shows the distribution of `"BodyStyle"` for all `"Brands"` in `evs`, other than Nissan. We will call EVs made by a `"Brand"` other than Nissan "non-Nissan EVs"; there are 24 non-Nissan EVs in evs.

Brand	Hatchback	Liftback	Pickup	SUV	Sedan
Tesla	0	2	3	4	3
BMW	1	0	0	1	1
Audi	0	0	0	8	1

= 7
= 2
= 9

only relevant!

Problem 6.1

Suppose we randomly select one of the non-Nissan EVs and it is either an SUV or a sedan. What is the most likely `"Brand"` of the randomly selected non-Nissan EV?

- Tesla
- BMW
- Audi

Problem 6

The DataFrame below shows the distribution of `"BodyStyle"` for all `"Brands"` in `evs`, other than Nissan. We will call EVs made by a `"Brand"` other than Nissan "non-Nissan EVs"; there are 24 non-Nissan EVs in evs.

Brand	Hatchback	Liftback	Pickup	SUV	Sedan
Tesla	0	2	3	4	3
BMW	1	0	0	1	1
Audi	0	0	0	8	1

= 3 total BMWs

Problem 6.2

Suppose we randomly select two of the non-Nissan EVs without replacement. The probability that both are BMWs is equal to $\frac{1}{k}$, where k is a positive integer. What is k?

$P(\text{both BMWs}) = P(\text{first BMW}) \cdot P(\text{second BMW given first BMW})$

$= \frac{3}{24} \cdot \frac{2}{23} = \frac{6^{1}}{24_{4} \cdot 23} = \frac{1}{23 \cdot 4} = \frac{1}{92}$ → 92

Problem 6

The DataFrame below shows the distribution of `"BodyStyle"` for all `"Brands"` in `evs`, other than Nissan. We will call EVs made by a `"Brand"` other than Nissan "non-Nissan EVs"; there are 24 non-Nissan EVs in evs.

Brand	Hatchback	Liftback	Pickup	SUV	Sedan
Tesla	0	2	3	4	3
BMW	1	0	0	1	1
Audi	0	0	0	8	1

= 13

Problem 6.3

Suppose we randomly select one of the non-Nissan EVs and it is an SUV. What is the probability that it is made by Tesla? Give your answer as a simplified fraction.

$\frac{4}{13}$ ← fraction of SUVs made by Tesla!

Problem 7

Below, we provide the same DataFrame as shown at the start of the previous problem, which contains the distribution of “BodyStyle” for all “Brands” in evs, other than Nissan.

Brand	Hatchback	Liftback	Pickup	SUV	Sedan
Tesla	0	2	3	4	3
BMW	1	0	0	1	1
Audi	0	0	0	8	1

Suppose we’ve run the following few lines of code.

```
tesla = evs[evs.get("Brand") == "Tesla"]
bmw = evs[evs.get("Brand") == "BMW"]
audi = evs[evs.get("Brand") == "Audi"]

combo = tesla.merge(bmw, on="BodyStyle").merge(audi, on="BodyStyle")
```

How many rows does the DataFrame `combo` have?

Brand	Hatchback	Liftback	Pickup	SUV	Sedan
Tesla	0	2	3	4	3
BMW	1	0	0	1	1
Audi	0	0	0	8	1

Tesla.merge(bmw) → 4×1 + 3×1 = 7

.merge(audi) → 32 + 3 = 35

Answer

Tesla

Brand	BodyStyle
Tesla	Liftback
⋮	Liftback
⋮	SUV
⋮	SUV
⋮	SUV
⋮	Sedan
⋮	Sedan
⋮	Sedan
⋮	⋮

BMW

Brand	Bodystyle
BMW	Hatchback
⋮	SUV
⋮	Sedan

Audi

Brand	BodyStyle
Audi	SUV
⋮	SUV
⋮	SUV } 8
⋮	⋮
⋮	SUV
⋮	Sedan

Example $3 \times 3 = 9$

Tesla

Brand	BodyStyle	Model
T	SUV	3
T	SUV	X
T	SUV	Y

Audi

Brand	BodyStyle	Model
A	SUV	R8
A	SUV	SQ5
A	SUV	A4

Problem 8

TritonTrucks is an EV startup run by UCSD alumni. Their signature EV, the TritonTruck, has a subpar battery (the engineers didn't pay attention in their Chemistry courses).

A new TritonTruck's battery needs to be replaced after 500 days, unless it fails first, in which case it needs to be replaced immediately. On any given day, the probability that a given TritonTruck's battery fails is 0.02, independent of all other days.

Fill in the blanks so that `average_days_until_replacement` is an estimate of the **average number of days a new TritonTruck's battery lasts without needing to be replaced**.

```
def days_until_replacement(daily_conditions):
    days = 0
    for i in __(a)__:
        if daily_conditions[i] == True:
            __(b)__
        else:
            return days
    return days

total = 0
repetitions = 10000
for i in np.arange(repetitions):
    # The first element of the first argument to np.random.choice is
    # chosen with probability 0.98
    daily_conditions = np.random.choice(__(c)__, 500, p=[0.98, 0.02])
    total = total + days_until_replacement(daily_conditions)
average_days_until_replacement = total / repetitions
```

np.arange (500)
or np.arange (len(daily_conditions))

days = days + 1
alternative
days += 1

[True, False]

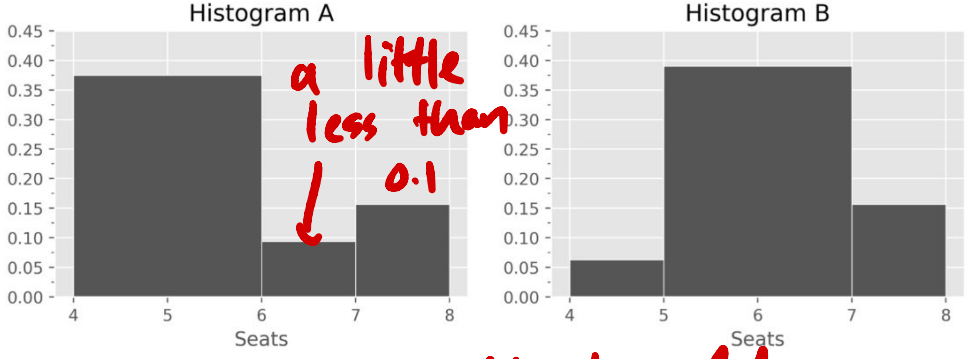
daily_conditions
[True, True, True, True, False
...,
True, False,
...]

What goes in blanks (a), (b), and (c)?

Problem 9

Histograms A and B below both display the distribution of the `"Seats"` column, using different bins. Each histogram includes all 32 rows of `evs`.

Remember: 32 rows in evs



prop = area = width · height
= 1 · 0.1
= 0.1

Problem 9.1

How many EVs in evs have exactly 6 seats?

Click to view the solution.

cars = prop · total
= 0.1 · 32
= 3.2 ↓

3

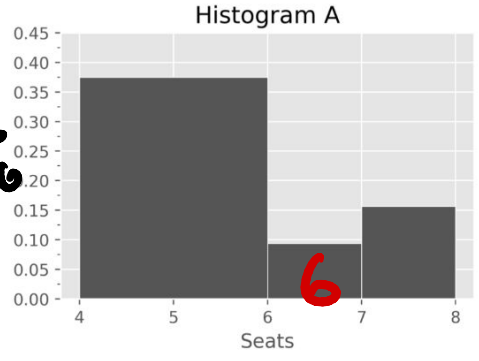
Problem 9.2

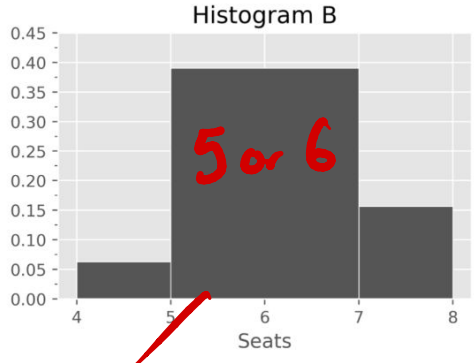
How many EVs in evs have exactly 5 seats?

Click to view the solution.

Problem 9

Histograms A and B below both display the distribution of the `"Seats"` column, using different bins. Each histogram includes all 32 rows of `evs`.





aside

$0.8 \cdot 32$
$= 0.8(30+2)$
$= \frac{4}{5}(30+2)$
$= 24 + 1.6$
$= 25.6$

$8 \cdot 32$
$= 2^3 \cdot 2^5$
$= 2^8$
$= 256$
$\rightarrow 25.6$

Problem 9.1

How many EVs in evs have exactly 6 seats?

Click to view the solution.

3

prop = width · height = $2 \cdot 0.4 = 0.8$

cars = $0.8 \cdot 32 = 25.6 \downarrow 25$

Problem 9.2

How many EVs in evs have exactly 5 seats?

Click to view the solution.

answer = $25 - 3 = 22$

Winter 2023 Midterm

Access the exam here. Make sure to read the data info sheet at the top before starting.

Problem 3.1

The DataFrame `amelia` was created by the code below, and is shown in its entirety.

```
amelia = (storms[(storms.get("Name") == "AMELIA") &
          (storms.get("Year") == 1978)]
          .get(["Year", "Month", "Day", "Time",
          "Status", "Latitude", "Longitude"]))
```

	Year	Month	Day	Time	Status	Latitude	Longitude
3486	1978	7	30	6PM	TD	25.7N	97.0W
3487	1978	7	31	12AM	TS	26.4N	97.4W
3488	1978	7	31	6AM	TS	27.2N	97.8W
3489	1978	7	31	12PM	TS	28.0N	98.2W
3490	1978	7	31	6PM	TD	28.6N	98.7W
3491	1978	8	1	12AM	TD	29.3N	99.2W

Use the space provided to show the DataFrame that results from

```
amelia.groupby("Status").max().
```

The column labels should go in the top row, and the row labels (`index`) should go in the leftmost row. You may not need to use all the rows and columns provided.

	Year	Month	Day	Time	Status	Latitude	Longitude
3486	1978	7	30	6PM	TD	25.7N	97.0W
3487	1978	7	31	12AM	TS	26.4N	97.4W
3488	1978	7	31	6AM	TS	27.2N	97.8W
3489	1978	7	31	12PM	TS	28.0N	98.2W
3490	1978	7	31	6PM	TD	28.6N	98.7W
3491	1978	8	1	12AM	TD	29.3N	99.2W

"6PM"
"6AM"
"12PM"
"12AM"

1
15
199
2
⋮

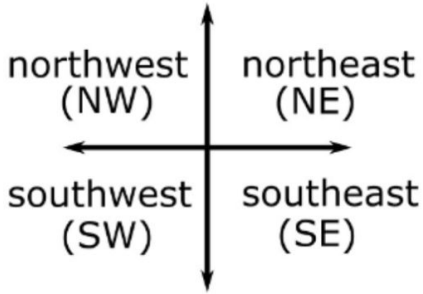
amelia.groupby("Status").max()

index	Year	Month	Day	Time	Latitude	Longitude
TD	1978	8	31	6PM	29.3N	99.2W
TS	1978	7	31	6AM	28.0N	98.2W

sorted in ascending order

The function `direction` takes as input four values: the latitude and longitude of a data entry for one storm, and the latitude and longitude of the next data entry for that same storm. The function should return the direction in which the storm moved in the period of time between the two data entries. The return value should be one of the following strings:

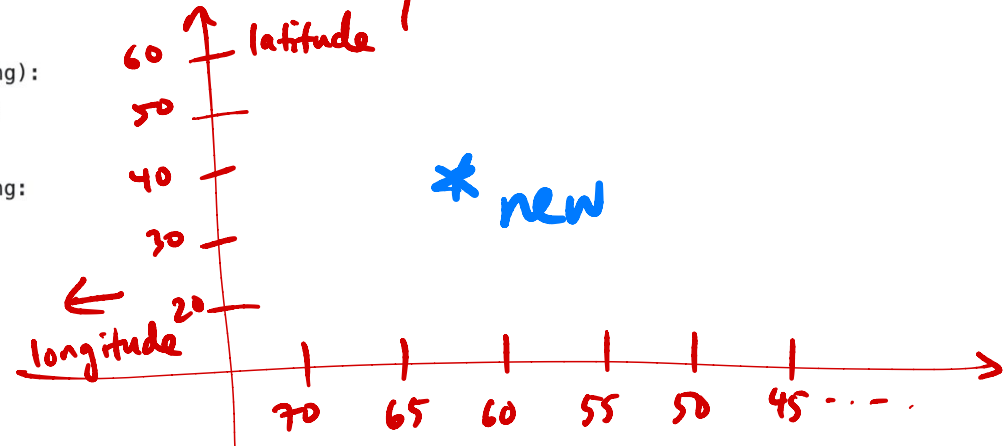
- `"NE"` for Northeastern movement,
- `"NW"` for Northwestern movement,
- `"SW"` for Southwestern movement, or
- `"SE"` for Southeastern movement

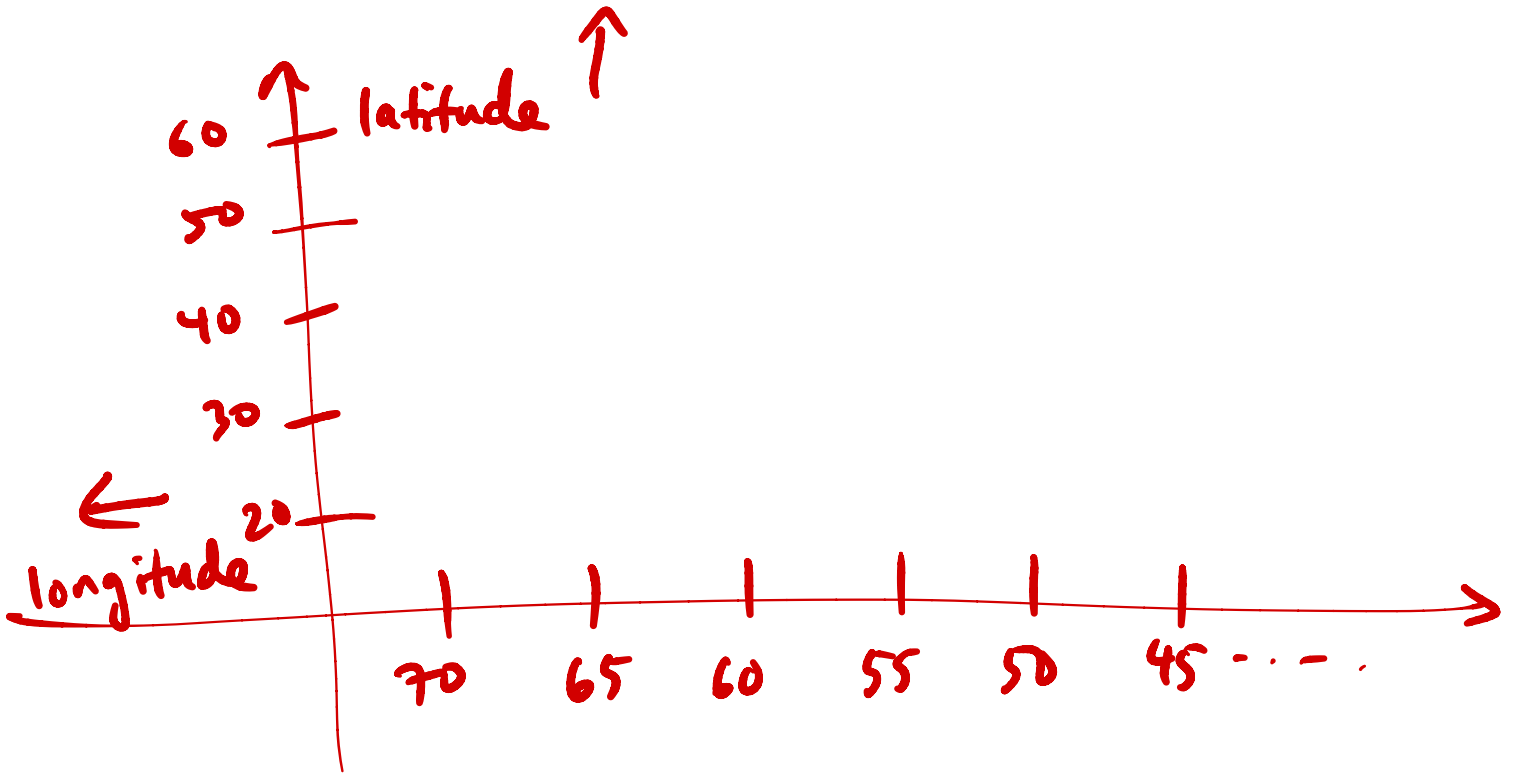


For example, `direction(23.1, 75.1, 23.4, 75.7)` should return `"NW"`. If the storm happened to move *directly* North, South, East, or West, or if the storm did not move at all, the function may return any one of `"NE"`, `"NW"`, `"SW"`, or `"SE"`. Fill in the blanks in the function `direction` below. It may help to refer to the images on Page 5.

```
def direction(old_lat, old_long, new_lat, new_long):
    if old_lat > new_lat and old_long > new_long:
        return ____(a)____
    elif old_lat < new_lat and old_long < new_long:
        return ____(b)____
    elif old_lat > new_lat:
        return ____(c)____
    else:
        return ____(d)____
```

"SE"
"NW"
"SW"
"NE"




latitude
60
50
40
30
20
longitude
70
65
60
55
50
45

Problem 7.1

np.arange(1,7) → [1,2,3,4,5,6]

The most famous Hurricane Katrina took place in August, 2005. The DataFrame `katrina_05` contains just the rows of storms corresponding to this hurricane, which we'll call Katrina'05.

Fill in the blanks in the code below so that `direction_array` evaluates to an array of directions (each of which is `"NE"`, `"NW"`, `"SW"`, or `"SE"`) representing the movement of Katrina '05 between each pair of consecutive data entries in `katrina_05`.

```
direction_array = np.array([])
for i in np.arange(1, ____(a)____):
    w = katrina_05.get("Latitude").____(b)____
    x = katrina_05.get("Longitude").____(c)____
    y = katrina_05.get("Latitude").____(d)____
    z = katrina_05.get("Longitude").____(e)____
    direction_array = np.append(direction_array, direction(w, x, y, z))
```

katrina_05.shape[0]

iloc[i-1]

iloc[i-1]

iloc[i]

iloc[i]

katrina_05	
latitude	longitude
57	25
54	34
47	6
44	7
39	81

old new

What goes in blank (a)?

Problem 8.1

['NE', 'SW', 'SW', 'NE', 'NW', 'SE', 'NW']

Now we want to use `direction_array` to find the number of times that Katrina '05 changed directions, or moved in a different direction than it was moving immediately prior. For example, if `direction_array` contained values `"NW"`, `"NE"`, `"NE"`, `"NE"`, `"NW"`, we would say that there were two direction changes (once from `"NW"` to `"NE"`, and another from `"NE"` to `"NW"`). Fill in the blanks so that `direction_changes` evaluates to the number of times that Katrina '05 changed directions.

```
direction_changes = 0
for j in ____(a)____:
    if ____(b)____:
        direction_changes = direction_changes + 1
```

np.arange(1, len(direction_array))

direction_array[j-1] != direction_array[j]

Problem 8.3

Answer= 32

There are 34 rows in `katrina_05`. Based on this information alone, what is the maxi- mum possible value of `direction_changes`?

katrina_05 : (7,3), (6,4), (9,2), (8,7), (3,5)

direction_array: NE SW SE NW

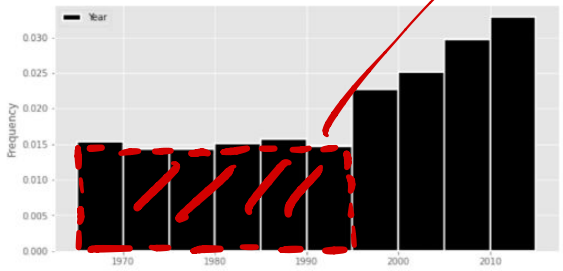
direction_changes: +1 +1 +1 = 3

Problem 10.1

Recall that all the named storms in `storms` occurred between 1965 and 2015, a fifty-year time period.

Below is a histogram and the code that produced it. Use it to answer the questions that follow.

```
(storms.groupby(["Name", "Year"]).count()
    .reset_index()
    .plot(kind="hist", y="Year",
        density=True, ec="w",
        bins=np.arange(1965, 2020, 5)));
```



Red Area = width · height
= (1995 − 1965) · 0.015
= 30 · 0.015
= 0.45 = 45%.

% storms in 1995 or later = (100 − 45)%.
= 55%.

Approximately **(a)** percent of named storms in this fifty-year time period occurred in 1995 or later. Give your answer to the nearest multiple of five.

Problem 12.1

Hurricane forecasters use complex models to simulate hurricanes. Suppose a forecaster simulates 10,000 hurricanes and keeps track of the state where each hurricane made landfall in an array called `landfalls`. Each element of `landfalls` is a string, which is either the full name of a US state or the string `"None"` if the storm did not hit land in the simulation.

The forecaster wants to use the results of their simulation to estimate the probability that a given storm hits Georgia. Write one line of Python code that approximates this probability, using the data in `landfalls`.

landfalls = np.array(["Georgia", "California", "None", ...])

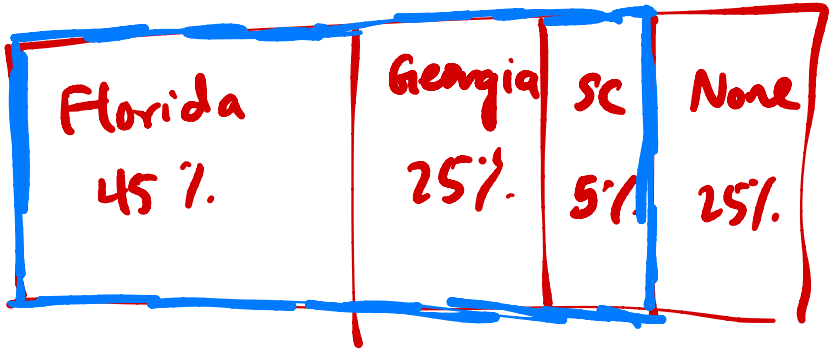
np.count_nonzero(landfalls == "Georgia") / 10000

Problem 12.2

Oh no, a hurricane is forming! Experts predict that the storm has a 45% chance of hitting Florida, a 25% chance of hitting Georgia, a 5% chance of hitting South Carolina, and a 25% chance of not making landfall at all.

Fast forward: the storm made landfall. Assuming the expert predictions were correct, what is the probability that the storm hit Georgia? Give your answer as a **fully simplified fraction** between 0 and 1.

Hint: The answer is not $\frac{1}{4}$, or 25%



$$\frac{25\%}{75\%} = \frac{1}{3}$$