index

January 10, 2022

1 The Probability Mass Function - Lab

In this lab you'll apply what you previously learned about probability mass functions (PMFs) to explore the *class size paradox*. The class size paradox describes apparent contradictory findings where a total allocation of resources is fixed.

The idea behind this paradox is that there is a difference in how events are actually distributed and how events are perceived to be distributed. These types of divergence can have important consequences for data analysis. Probability mass functions can help resolve some of these situations, as you'll learn below.

1.1 Objectives

You will be able to:

- Explain the class size paradox
- Create visualizations to visually compare actual and biased observations
- Calculate the mean from PMFs to identify the expected value

1.2 The Problem

At a university, the expected student-to-teacher ratio is 32.5 : 1. But randomly interviewed students often feel that their average class size is bigger than 32.5. There are two main reasons for this:

1. Students typically take 4 - 5 classes at any given time, but teachers usually only teach 1 or 2 classes.
2. The number of students in a small class is small, and the number of students in a large class is large.

Due to the second fact, while randomly taking feedback from students (and sampling randomly), it is expected we will come across *more* students from larger classes simply because there are more of them.

Let's work through a set of data to recreate and analyze this paradox.

Suppose that a college offers 74 classes in a term. We can start with the following distribution of sizes and counts:

Class size	Class count
15-19	10
20-24	10

Class size	Class count
25-29	18
30-34	6
35-39	8
40-44	10
45-49	5
50-54	3
55-59	4

If the campus manager were asked about the average class size, he would perform the following tasks:

1. Construct a PMF from given data
2. Compute the mean using the PMF

Let's follow the management approach first and see what expected value we get from our PMF. Here is a `size_and_count` dictionary to get you started. Calculate the PMF from this data as we have done before.

To make it slightly more straightforward, we have averaged the class sizes for each class, i.e. for size "15 - 19", we use the average value, 17. This allows us to treat each row of the table above as a single discrete category, represented by the average value of the category.

[85]:
```
size_and_count = {17: 10, 22: 10, 27: 18, 32: 6, 37: 8, 42: 10, 47: 5, 52: 3,␣
↪57: 4}
```

Following the approach seen in the previous lesson, calculate a list of PMF values by normalizing each size.

(Treat the `size_and_count` dictionary as the equivalent of the `counter` variable from the previous lesson — you do not need to count the raw data values because it has already been done for you, but the logic to find the total number of classes will be a bit more elaborate because you don't have access to the raw data.)

We will also use this an an opportunity to practice using pandas, which has convenient built-in methods and broadcasting.

[86]:
```
import numpy as np
import pandas as pd

# Determine total number of classes (integer value)
sum_class = np.sum(np.array(list(size_and_count.values())))
sum_class

# From GitHub
# sum_class = sum(size_and_count.values())

# Create a pandas Series of all possible outcomes (class sizes)
sizes = pd.Series(size_and_count.keys())
```

```
# Divide each class size value by the total number of classes
# to create a pandas Series of PMF values
actual_pmf = pd.Series([item/sum_class for item in size_and_count.values()])

# Display probabilities in a dataframe
pmf_df = pd.concat([sizes, actual_pmf], axis=1)
pmf_df.columns = ["Class Size", "Overall Probability"]
pmf_df.style.hide_index()
```

```
[86]: <pandas.io.formats.style.Styler at 0x7f834b5b0a30>
```

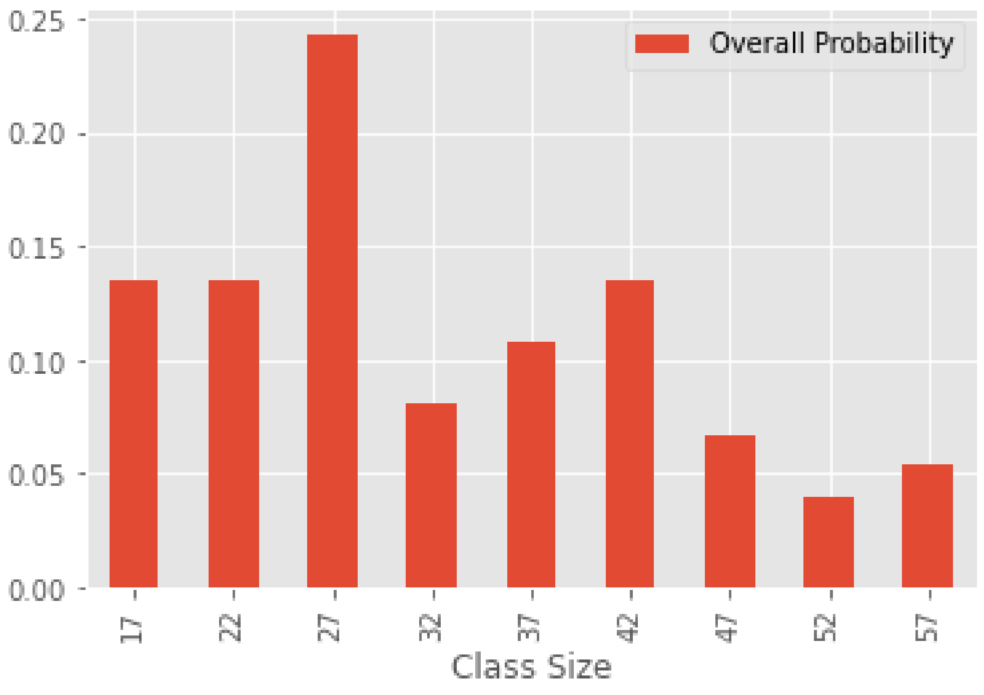
As an additional check, these probability values must sum to 1. Let's check for that. Run the following cell:

```
[87]: # The output should be 1
actual_pmf.sum()
```

```
[87]: 1.0
```

Because this is a dataframe, we can use the built-in `.plot.bar` method to view the class sizes as a bar graph:

```
[88]: import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
pmf_df.plot.bar(x="Class Size", y="Overall Probability");
```



Let's also write the PMF as a Python function `p_actual`. Meaning, it takes in a given x_i value (a class size) and returns the probability of that outcome from the management perspective.

You can use the global variables `size_and_count` and `sum_class`.

[89]:
```python
# def p_actual(x_i):
#     # Your code here
#     return pmf_df.loc[pmf_df["Class Size"]==x_i]["Overall Probability"]

# p_actual(17) # 0.13513513513513514

### From GitHub Solution
def p_actual(x_i):
    return size_and_count[x_i] / sum_class

p_actual(17)
```

[89]:
```
0.13513513513513514
```

1.3 Calculate the Mean or Expected Value $E(X)$

We can now calculate the mean or **Expected Value** for this distribution.

> The mean μ or expected value **E(X)** of a random variable X is the sum of the possible values for X weighted by their respective probabilities.

$$E(X) = \mu = \sum_i p(x_i)x_i$$

In simple terms, you have to multiply each element in the sizes list by their probability of occurrence then sum the resulting values.

We can do this in one line of code using pandas broadcasting. (E.g. `sizes.apply(p_actual)` will result in a series containing all $p(x_i)$ values.)

[90]:
```python
# Calculate the expected value (mu) using formula above
# mu = sum(pmf_df["Class Size"]*pmf_df["Overall Probability"])
# mu

# 32.472972972972975

## From GitHub Solution
mu = (sizes.apply(p_actual) * sizes).sum()
mu
```

[90]:
```
32.472972972972975
```

Recall, we expected the average class size to be 32.5. Indeed, the calculation above confirms this.

1.4 Random Student Survey

Next, we conduct a survey on a random group of students about their class sizes and then compute the mean. Paradoxically, we observed that the average class is bigger than 32.5. How did this happen? Let's see this in action below:

First, let's compute a distribution as a likely observation **by students**, where the probability associated with each class size is "biased" by the **number of students** in the class. If this sounds confusing, think of it this way: instead of calculating a PMF using the counts of class sizes, calculate it using the counts of students.

Perform the following tasks to introduce this bias.

- For each class size x, multiply the class probability by x, the number of students who observe that particular class size
- Get the sum of biased class sizes

The result is a new PMF that represents the biased distribution.

[91]:
```
biased = sizes.apply(p_actual) * sizes
biased
```

[91]:
```
0    2.297297
1    2.972973
2    6.567568
3    2.594595
4    4.000000
5    5.675676
6    3.175676
7    2.108108
8    3.081081
dtype: float64
```

You can now normalize the new biased list with the sum of its values, just like you did before. - Normalize the biased list and calculate the new PMF

[94]:
```
biased_pmf = pd.Series([value/mu for value in biased])
biased_pmf
```

[94]:
```
0    0.070745
1    0.091552
2    0.202247
3    0.079900
4    0.123179
5    0.174782
6    0.097794
7    0.064919
8    0.094881
```

```
dtype: float64
```

You can see that probability values in this PMF are different than our original pmf. Note the differences in the table below:

[95]:
```
pmf_df["Perceived Probability"] = biased_pmf
pmf_df
```

[95]:
```
   Class Size  Overall Probability  Perceived Probability
0          17             0.135135               0.070745
1          22             0.135135               0.091552
2          27             0.243243               0.202247
3          32             0.081081               0.079900
4          37             0.108108               0.123179
5          42             0.135135               0.174782
6          47             0.067568               0.097794
7          52             0.040541               0.064919
8          57             0.054054               0.094881
```

Again, we can represent this as a function, this time called `p_perceived`.

[96]:
```
def p_perceived(x_i):
    return p_actual(x_i)*x_i / mu

p_perceived(17)
```

[96]:
```
0.07074490220557636
```

Just like before, you can calculate the expected value μ. This time, use `p_perceived` instead of `p_actual` in your calculation.

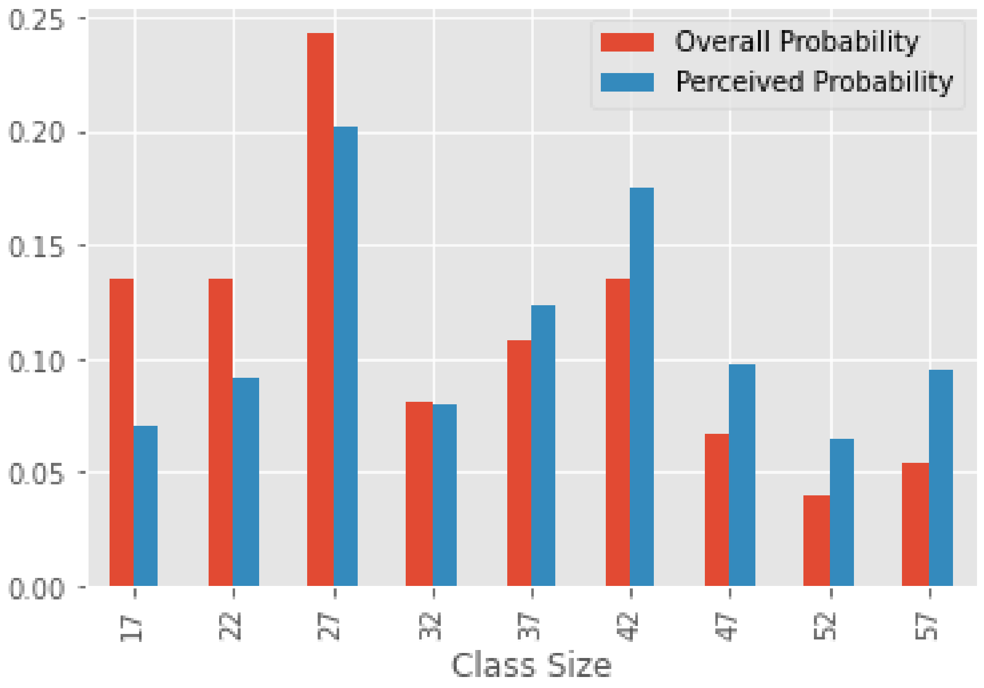
[97]:
```
mu_biased = (sizes.apply(p_perceived)*sizes).sum()
mu_biased

# 36.51310861423221
```

[97]:
```
36.51310861423221
```

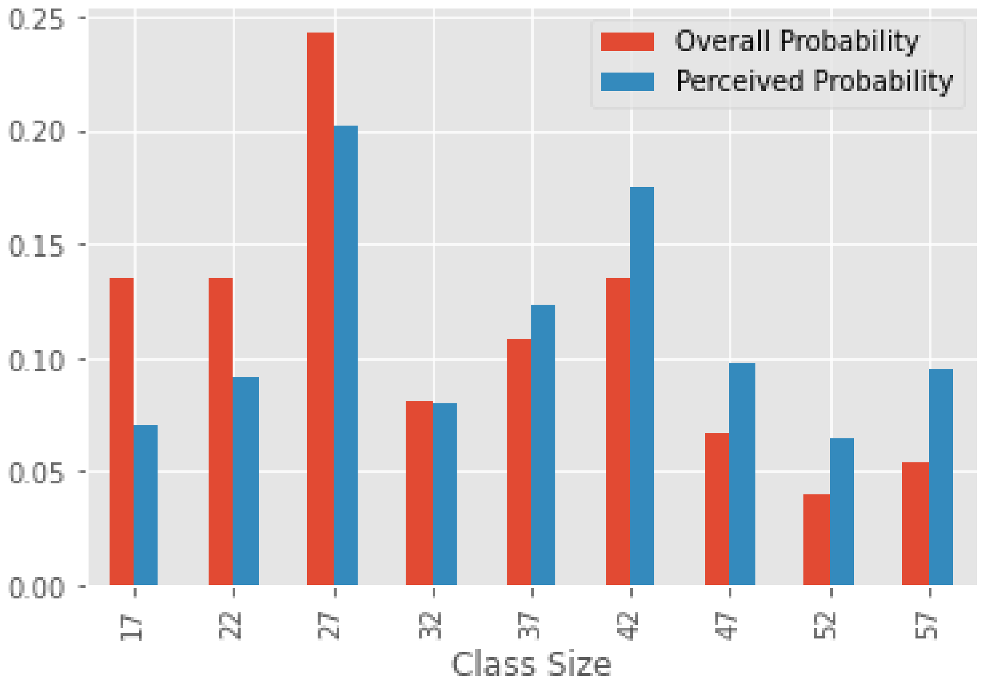
1.5 Here Is the Paradox

Here we see it, the average or expected value of biased results comes out higher than the actual values. In some situations, a paradox like this can be mind-boggling. As an extra measure, inspect both PMFs side by side visually to see the differences.

You can use `.plot.bar` again on `pmf_df`, this time changing the `y` parameter so that both probability distributions will be plotted side-by-side. Your plot should look like this:



[98]:
```
# Your code here
pmf_df.plot(
         x = "Class Size",
         y =["Overall Probability","Perceived Probability"]
        , kind ="bar");
```

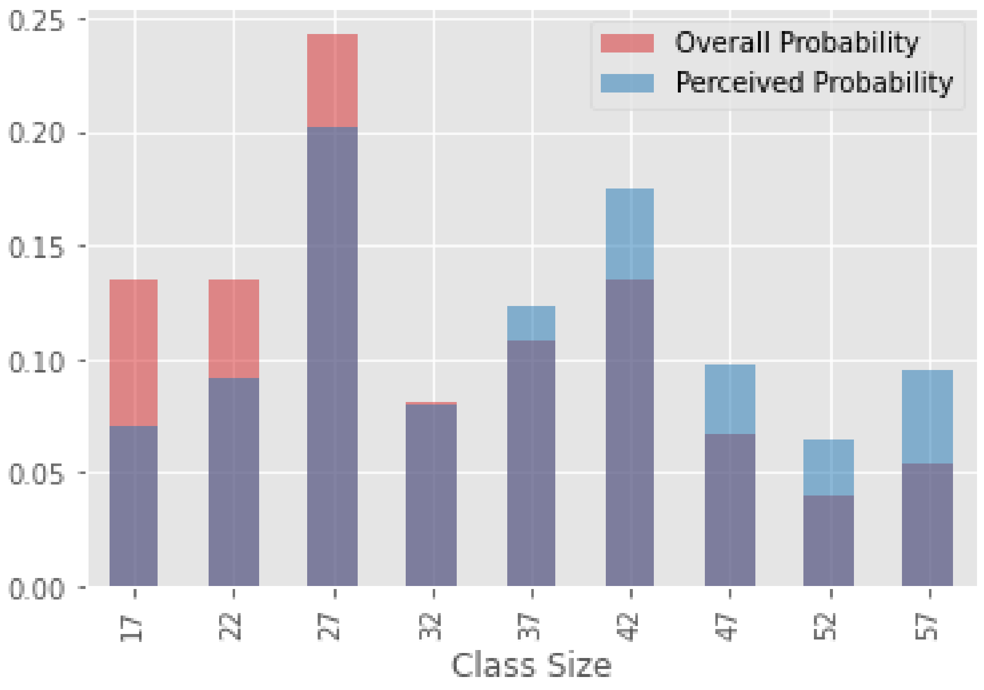


Your results tell you that in the biased distribution there are fewer small classes and more large classes.

The mean of the biased distribution is ~36.5, which is quite a bit higher than the actual mean of ~32.5.

For an alternative comparison where it is easier to see which value is higher, plot these PMFs on top of each other with semi-transparent bar fill.

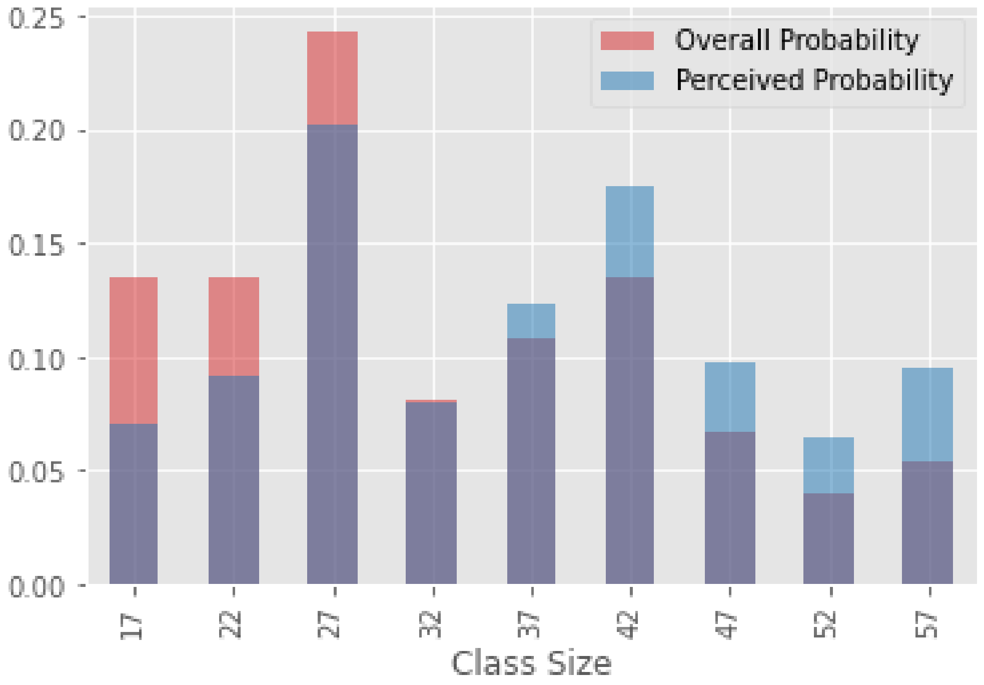
Your plot should look like this:



Hints:

- You will need call `.plot.bar` twice, and pass in `ax`, so that both plots use the same axes
- Change the parameter `alpha` to adjust the transparency
- If you don't specify a color, both will plot with the default red color and you won't be able to tell which is which. In the above version, "Overall Probability" has a `color` of `"tab:red"` and "Perceived Probability" has a `color` of `"tab:blue"`, but you're free to customize it differently!

[117]:
```python
# Setting up shared axes
fig, ax = plt.subplots()


# Your code here
pmf_df.plot.bar(
         x = "Class Size",
         y =["Overall Probability"],
#          kind ="bar",
         color = "tab:red",
         alpha = 0.5,
         ax = ax);

pmf_df.plot.bar(
         x = "Class Size",
         y =["Perceived Probability"],
#          kind ="bar",
         color = "tab:blue",
         alpha = 0.5,
         ax = ax);
```



Here is the key: for smaller class sizes, the probability of coming across a students is lower than the actual probability. For larger classes, the probability of coming across a student is much higher than actual probability. This explains why the paradox takes place!

1.6 Summary

In this lesson, we looked at a common paradox called the "class size paradox", which deals with differences in observation by different people based on their circumstances.

Note that this phenomenon is not just limited to class sizes. It applies to many scenarios where people are grouped together, such as in the context of social networks. This paradox can become really complicated due to the large number of individuals involved and the resulting variations in the probabilities of their observations which arise due to their settings.