Choosing a Plot for Discrete Data

If you want to plot a **discrete quantitative variable**, it is possible to select either a histogram or a bar chart to depict the data.

- Here, the **discrete** means non-continuous values. In general, a discrete variable can be assigned to any of the limited (countable) set of values from a given set/range, for example, the number of family members, number of football matches in a tournament, number of departments in a university.
- The **quantitative** term shows that it is the outcome of the measurement of a quantity.

The histogram is the most immediate choice since the data is numeric, but there's one particular consideration to make regarding the bin edges. Since data points fall on set values (bar-width), it can help to reduce ambiguity by putting bin edges between the actual values taken by the data.

An example describing the ambiguity

For example, assume a given bar falls in a range [10-20], and there is an observation with value 20. This observation will lie on the *next* bar because the given range [10-20] does not include the upper limit 20. Therefore, your readers may not know that values on bin edges end up in the bin to their right, so this can bring potential confusion when they interpret the plot.

Compare the two visualizations of 100 random die rolls below (in die_rolls), with bin edges *falling on* the observation values in the left subplot, and bin edges *in between* the observation values in the right subplot.

Preparatory Step

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb

%matplotlib inline


die_rolls = pd.read_csv('die_rolls.csv')


# A fair dice has six-faces having numbers [1-6].

# There are 100 dices, and two trials were conducted.

# In each trial, all 100 dices were rolled down, and the outcome [1-6] was recorded.

# The `Sum` column represents the sum of the outcomes in the two trials, for each given dice.
```

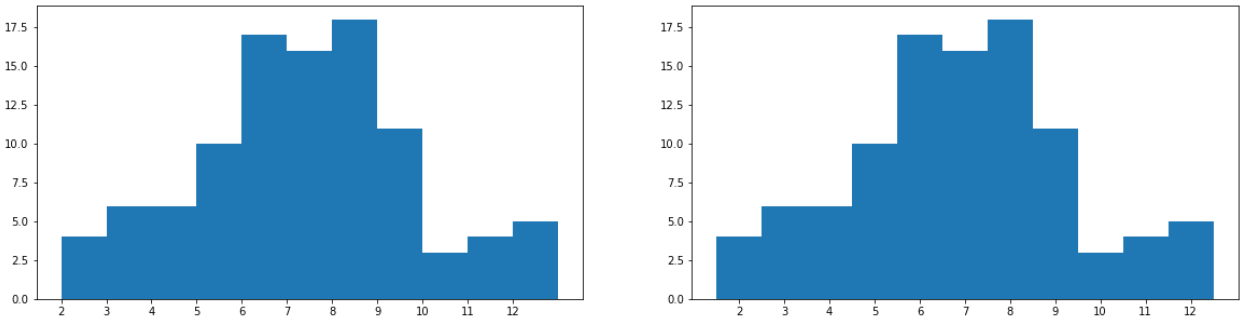
```
die_rolls.head(10)
```

The die_rolls.csv file is available to download at the bottom of this page.

	Dice	Trial 1	Trial 2	Sum
0	1	4	1	5
1	2	4	5	9
2	3	2	6	8
3	4	6	3	9
4	5	3	6	9
5	6	6	6	12
6	7	3	3	6
7	8	3	2	5
8	9	2	6	8
9	10	6	6	12

Example 1. Shifting the edges of the bars can remove ambiguity in the case of Discrete data

```
plt.figure(figsize = [20, 5])

# Histogram on the left, bin edges on integers

plt.subplot(1, 2, 1)

bin_edges = np.arange(2, 12+1.1, 1) # note `+1.1`, see below

plt.hist(data=die_rolls, x='Sum', bins = bin_edges);

plt.xticks(np.arange(2, 12+1, 1));


# Histogram on the right, bin edges between integers

plt.subplot(1, 2, 2)

bin_edges = np.arange(1.5, 12.5+1, 1)

plt.hist(data=die_rolls, x='Sum', bins = bin_edges);

plt.xticks(np.arange(2, 12+1, 1));
```



The same data is plotted in both subplots, but the alignment of the bin edges is different.

You'll notice for the left histogram, in a deviation from the examples that have come before, I've added 1.1 to the max value (12) for setting the bin edges, rather than just the desired bin width of 1. Recall that data that is equal to the rightmost bin edge gets lumped in to the last bin. This presents a potential problem in perception when a lot of data points take the maximum value, and so is especially relevant when the data takes on discrete values. The 1.1 adds an additional bin to the end to store the die rolls of value 12 alone, to avoid having the last bar catch both 11 and 12.

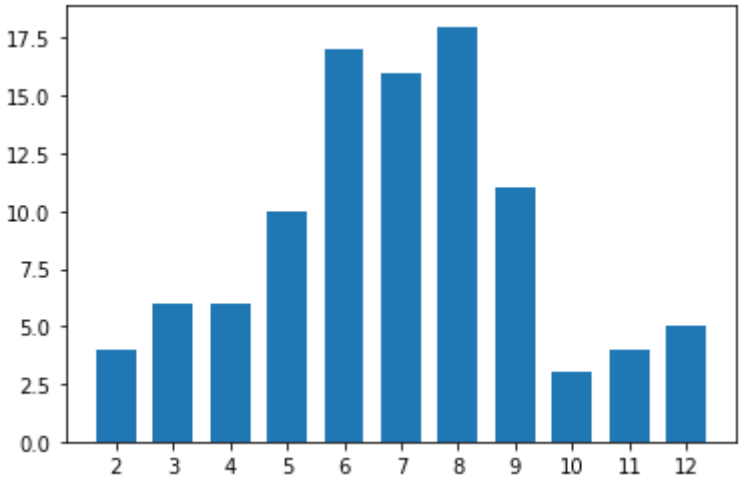
Alternatively to the histogram, consider if a bar chart with non-connected bins might serve your purposes better. The plot below takes the code from before, but adds the "rwidth" parameter to set the proportion of the bin widths that will be filled by each histogram bar.

Example 2. Making gaps between individual bars

```
bin_edges = np.arange(1.5, 12.5+1, 1)
```

```
plt.hist(data=die_rolls, x='Sum', bins = bin_edges, rwidth = 0.7)

plt.xticks(np.arange(2, 12+1, 1));
```

With "rwidth" set to 0.7, the bars will take up 70% of the space allocated by each bin, with 30% of the space left empty. This changes the default display of the histogram (which you could think of as "rwidth = 1") into a bar chart.



Gaps between bars makes it clear that the data is discrete in nature.

By adding gaps between bars, you emphasize the fact that the data is discrete in value. On the other hand, plotting your quantitative data in this manner might cause it to be interpreted as ordinal-type data, which can have an effect on overall perception.

For continuous numeric data, you should not make use of the "rwidth" parameter, since the gaps imply discreteness of value. As another caution, it might be tempting to use seaborn's countplot function to plot the distribution of a discrete numeric variable as bars. Be careful about doing this, since each unique numeric value will get a bar, regardless of the spacing in values between bars. (For example, if the unique values were {1, 2, 4, 5}, missing 3, countplot would only plot four bars, with the bars for 2 and 4 right next to one another.) Also, even if your data is technically discrete numeric, you should probably not consider either of the variants depicted on this page unless the number of unique values is small enough to allow for the half-unit shift or discrete bars to be interpretable. If you have a large number of unique values over a large enough range, it's better to stick with the standard histogram than risk interpretability issues.

While you might justify plotting discrete numeric data using a bar chart, you'll be less apt to justify the opposite: plotting ordinal data as a histogram. The space between bars in a bar chart helps to remind the reader that values are not contiguous in an 'interval'-type fashion: only that there is an order in levels. With that space removed as in a histogram, it's harder to remember this important bit of interpretation.

Supporting Materials

- die_rolls.csv