module1_31

type: slides

Quick Viz with Pandas!

Notes: Script here.

If we want to visualize things using different plots we can do that pretty quickly and with little code! Take `manufacturer_freq` object we made in the last slide deck.

```
manufacturer_freq
```

```
## K    23
## G    22
## P     9
## R     8
## Q     8
## N     6
## A     1
## Name: mfr, dtype: int64
```

This would be great to express as a bar chart. But how do we do it?

Notes: Script here.

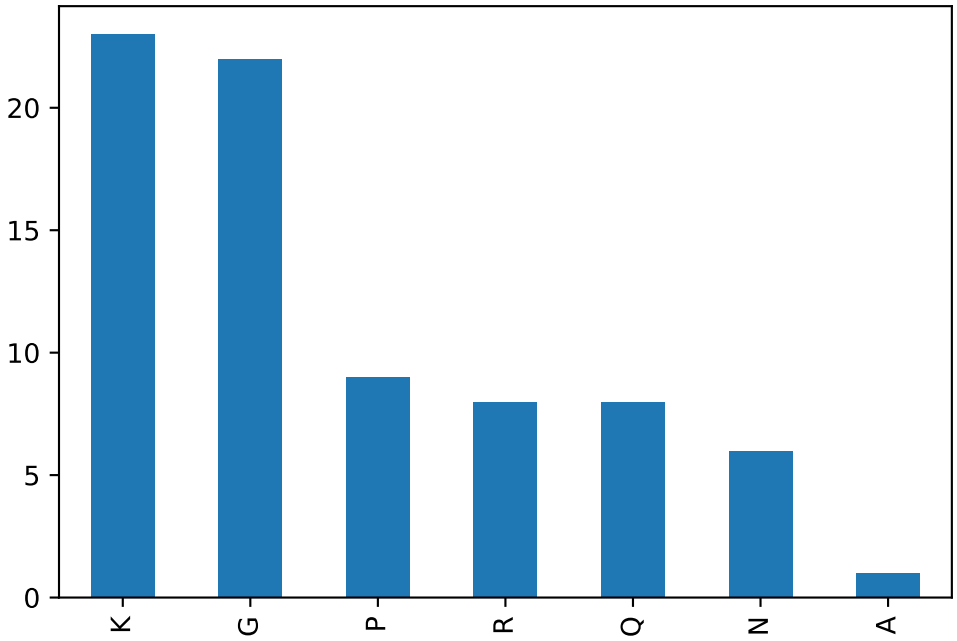
---

1. We take the dataframe we wish to plot, in this case `freq_mfr_df`.

2. Next we add `.plot` since we want to plot it!

3. But what kind of plot do we want?! A bar chart in this case would work nicely so lets add `.bar()` after that.

```
manufacturer_freq.plot.bar()
```

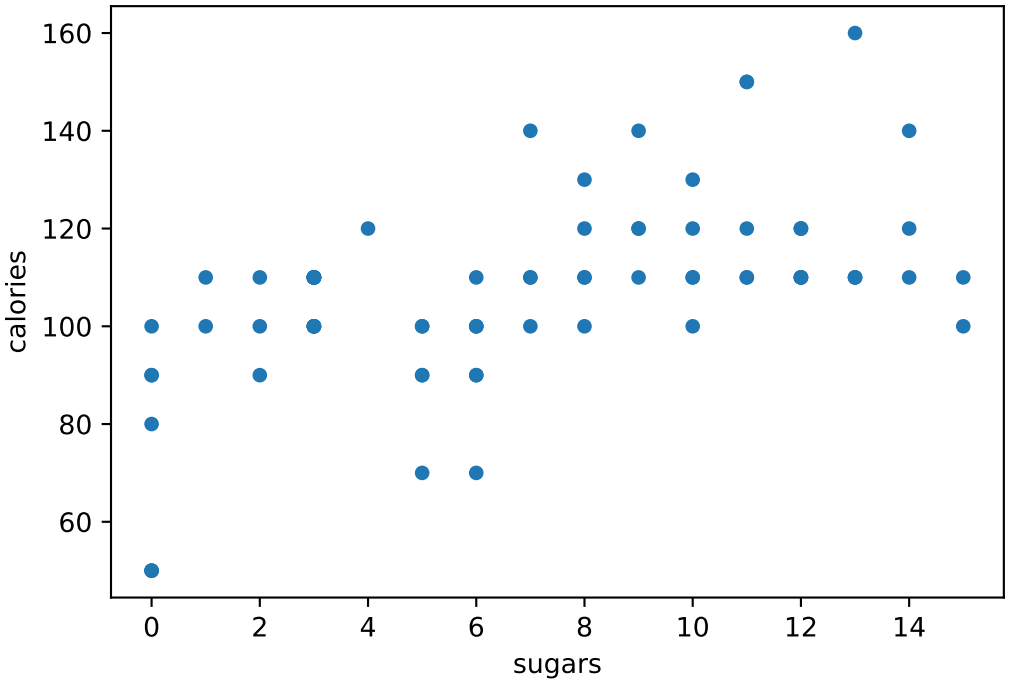


See how quick that was? The important things to notice here is that we want to `.plot` a `.bar()` graph.

Notes: Script here.

What else can we plot from our original cereal dataframe named `df`? Maybe we want to see the relationship between `sugars` and `calories` in cereals?
This would require a `scatter` plot!
In the code we would need to specify the x and y axis which means we would need to specify the column names for each axis; In this case the x-axis is the `sugars` column and the y-axis is the `calories` column.

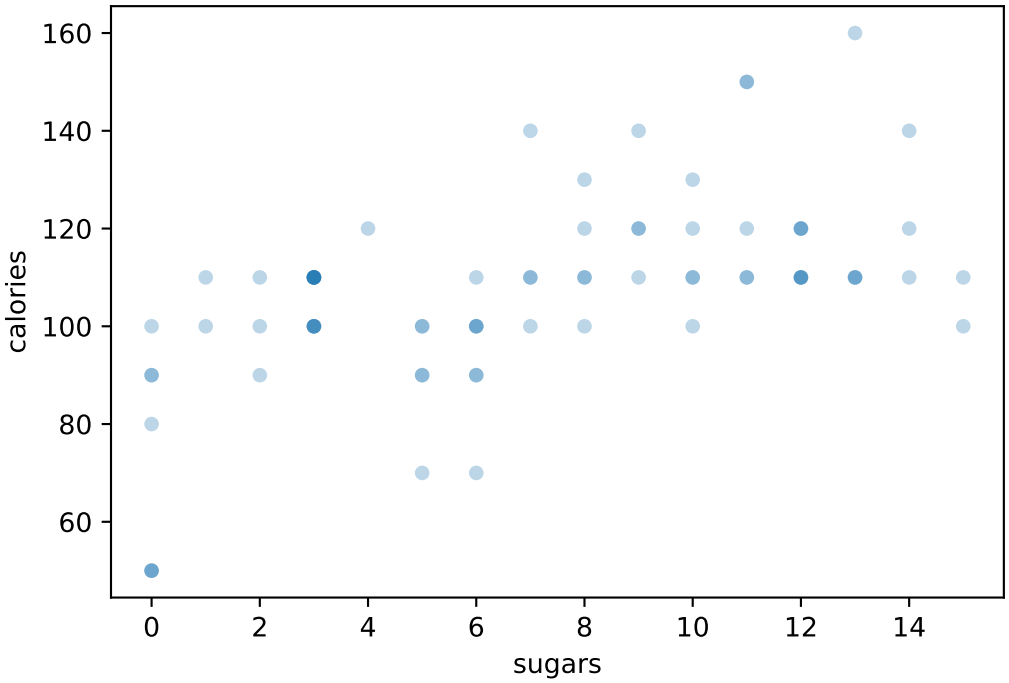
```
df.plot.scatter(x='sugars', y='calories')
```



Notes: Script here.

Something you may have noticed is that there are 77 cereals but there doesn't seem to be 77 data points! That's because some of them are lying on top of each other with the same sugar ar calorie values. It may be of use to set an opacity to the graph to differential those points. Opacity is set with the argument `alpha` and accepts values between 0 and 1, with 1 being full intensity.

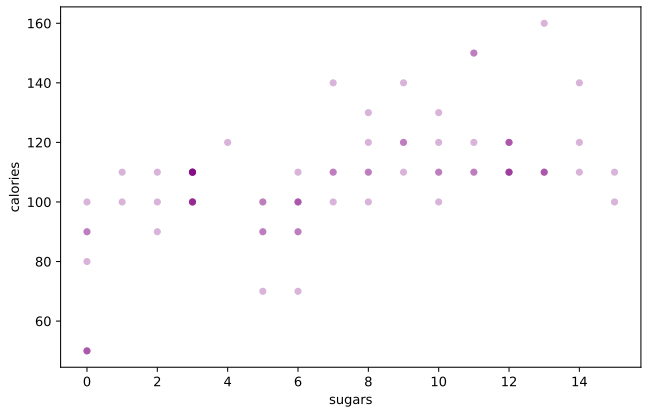
```
df.plot.scatter(x='sugars', y='calories', alpha=0.3)
```



Notes: Script here.

Look at that! Now we can see there are multiple cereals that have 2.5g of sugar with 100 calories. What if we wanted to change the colour to purple? Enter parameter `color`! We can also add a bit of readability by separating the arguments into separate lines.

```
df.plot.scatter(x='sugars',
                y='calories',
                alpha=.3,
                color='purple')
```



Notes: Script here.

Those data points look pretty small. To enlarge them, the argument `s` should do the trick. Also every good graph should have a title! Let's take this opportunity to finish off this graph and set the argument `title` to something as well.

```
df.plot.scatter(x="sugars",
                y="calories",
                alpha= 0.3,
                color="purple",
                s=80,
                title="The relationship between sugar and calories in cereals")
```

```
import altair as alt
import pandas as pd
from altair_saver import save

source = pd.read_csv('cereal.csv')

chart1 = alt.Chart(source).mark_bar().encode(
    x='mfr',
    y='count()'
)
chart1
```

```
## alt.Chart(...)
```

Let's apply what we learned!

Notes: Script here