Visualization with Multi-Plot Grids

In this lesson, we will discuss how to use Seaborn to create multi-plot grids.

WE'LL COVER THE FOLLOWING

Introduction to multi-plot grids

Seaborn also has some multi-plot grid capabilities.

I don't think I could explain it any better than the Seaborn documentation:

When exploring medium-dimensional data, a useful approach is to draw multiple instances of the same plot on different subsets of your dataset. This technique is sometimes called either "lattice" or "trellis" plotting, and it is related to the idea of "small multiples". It allows a viewer to quickly extract a large amount of information about complex data. Matplotlib offers good support for making figures with multiple axes; seaborn builds on top of this to directly link the structure of the plot to the structure of your dataset.

Multi-plot grids in Python

FacetGrid and map

I think the easiest way to understand, is to use an example in Python from the flight's dataset.

First, we will look at the `FacetGrid()` and `map()` combination in Seaborn.

The parameters we will use for `FacetGrid` are:

- `data` which is the first parameter and specifies the dataframe to use.
- `row` which is the column name from your dataframe you want to use as the rows of your grid.
- `col` which is the column name from your dataframe you want to use as the columns of your grid.
- `margin_titles` which if `True`, the titles for the row variable are drawn to the right of the last column.

The parameters we will use for `map` are:

- `func` which is the first argument and is the plotting function you wish to use.
- `args` is the second parameter and is the column name for the variable you wish to plot.
- `color` which allows you to specify the plot color.

Let's take a look:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
flights = sns.load_dataset("flights")
# Subset the data to years >= 1956 to more easily fit on the plot
flights = flights[flights.year >= 1956]

g = sns.FacetGrid(flights, row="year", margin_titles=True)
g.map(plt.plot, "passengers", color="steelblue")
```

First, we created a FacetGrid by passing our dataframe, which in this example is called flights. We then pass the `row` variable (**year**) we want to use in the resulting plot. You can see in our plot above that we have one plot per year. Margin titles tell the plots to add the titles in the margin.

Once we have our FacetGrid, we call the **map** function in order to map data onto our grid. First, we tell it what type of plot we want to use. In this example, we use a line plot from `Matplotlib`. We then pass the variable we want plotted, **passengers**.

This creates a line plot for each row for our variable of interest.

PairGrid

Another common multi-plot function in Seaborn is `PairGrid()`. This function plots pairwise relationships in a grid. You generally pass one parameter which is a dataframe consisting of the columns you wish to plot.

You then call the `map()` function which maps the pairwise combinations of your columns onto the plot of your choice. The plotting function you wish to use is the parameter you pass to `map()`.

Here is an example:

```
from sklearn.datasets import load_boston
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# Set the palette and style to be more minimal
sns.set(style='ticks', palette='Set2')

# Load data as explained in introductory lesson
boston_data = load_boston()
boston_df = pd.DataFrame(boston_data.data, columns=boston_data.feature_names)

# Create and map the PairGrid
g = sns.PairGrid(boston_df[['CRIM', 'NOX', 'INDUS']])
g.map(plt.scatter);

# Remove excess chart lines and ticks for a nicer looking plot
sns.despine()
```

The result, as you see, is all of the pairwise scatterplots of your variables allowing you to quickly scan for potential relationships. Since the diagonal of your grid is the scatter plot of variables with themselves, you will always get data on the 45-degree line.

You have a lot of ways you can experiment with the FacetMap and map combination, so have fun! You can learn more from the Seaborn documentation.

Now that you are familiar with techniques to visualize any dataset, the next lesson brings some challenges for you to solve.