

Learn / Courses / Introduction to Data Visua...
Course Outline

Exercise

### Plotting subgroups in line plots

Let's continue to look at the `mpg` dataset. We've seen that the average miles per gallon for cars has increased over time, but how has the average horsepower for cars changed over time? And does this trend differ by country of origin?

Instructions 2/3
35 XP
2
3

- Create different lines for each country of origin (`"origin"`) that vary in both line style and color.

Take Hint (-10 XP)

```

1  # Import
   Matplotlib and
   Seaborn
2  import matplotlib.
   pyplot as plt
3  import seaborn as
   sns
4
5  # Change to
   create subgroups
   for country of
   origin
6  sns.relplot
   (x="model_year",
    y="horsepower",
7
   data=mpg,
   kind="line",
8
   ci=None)
9
10 # Show plot

```

Run Code
Submit Answer

Previous Plot 2/2 Next Plot

IPython Shell
Slides

In [1]:

## Plotting Subgroups in Line Plots with Different Line Styles and Colors

Create different lines for each country of origin ('origin') that vary in both line style and color.

### Full Answer ###

To create subgroups with different line styles and colors, use the 'hue' and 'style' parameters set to 'origin'. Below is the working code:

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

# Create subgroups for country of origin with varying line style and color
sns.relplot(x='model_year', y='horsepower',
            data=mpg,
            kind='line',
            hue='origin',
            style='origin',
            ci=None)

# Show plot
plt.show()
```

### ### Code Explanation ###

1. Import seaborn and matplotlib.pyplot for creating visualizations.
2. Use `sns.relplot()` to create a line plot with:
  - 'x' set to 'model\_year' for the year the car model was produced.
  - 'y' set to 'horsepower' for the engine power.
  - 'kind' set to 'line' to generate a line plot.
  - 'hue' set to 'origin' to vary the line colors based on the country of origin.
  - 'style' set to 'origin' to vary the line styles based on the country of origin.
  - 'ci' set to 'None' to turn off the confidence intervals.
  - 'data' set to `mpg`, the DataFrame containing the data.
3. Use `plt.show()` to render and display the plot.