SI 305 Discussion 11: Visualization in Seaborn

Arpitha / Coulton / Haley

What is seaborn

Flexible visualization package built on top of matplotlib

- Many functions in seaborn use similar syntax to their equivalents in matplotlib

Supports more kinds of plots that matplotlib

Like with matplotlib, there are many ways to create the same chart in seaborn. You're welcome to experiment with other plotting techniques in addition to the ones we cover today.

What is seaborn

```
# Import seaborn
import seaborn as sns
```

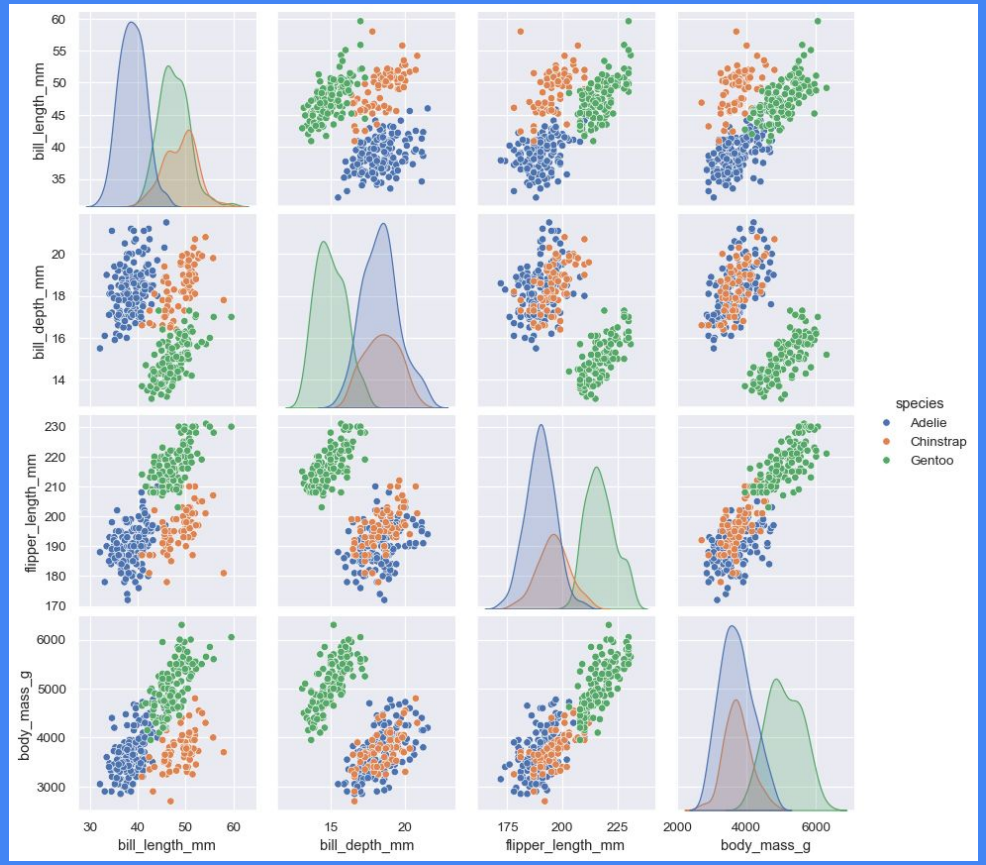
Useful charts: pairplot

Good for quickly exploring the relationships in your dataset

Creates a grid of the variables in your dataset

- Diagonals are the distribution of a variable
- Off-diagonals are a scatterplot showing the relationship between two variables

```
sns.pairplot(data=penguins, hue="species")
```

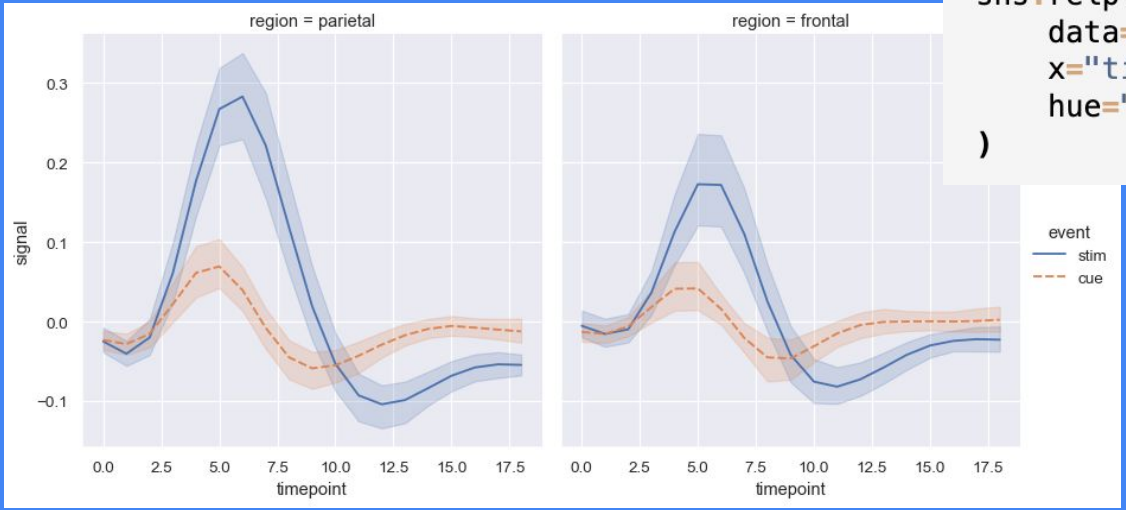


Useful charts: replot

Creates a line chart with confidence intervals

Seaborn uses bootstrapping to construct confidence intervals for the point estimates

```
fmri = sns.load_dataset("fmri")
sns.relplot(
    data=fmri, kind="line",
    x="timepoint", y="signal", col="region",
    hue="event", style="event",
)
```

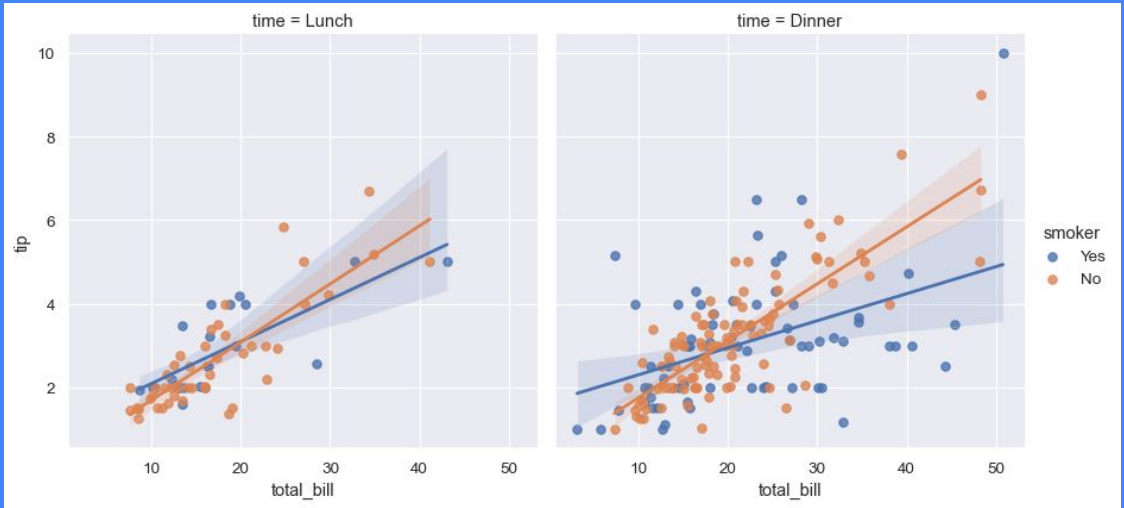


Useful charts: lmplot

Plots a linear regression line over a scatter plot

Will run a separate linear regression for each value of the hue parameter

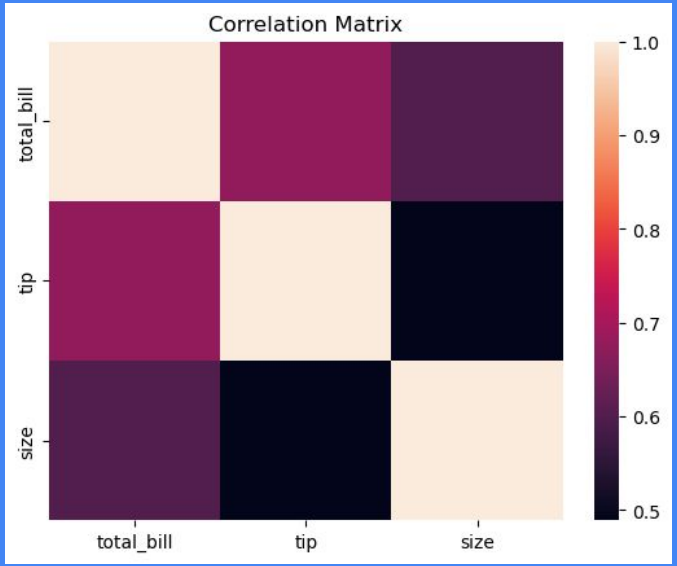
```
sns.lmplot(data=tips, x="total_bill", y="tip", col="time", hue="smoker")
```



Useful charts: heatmap

Plots the magnitude of individual variables

These are often used to visualize correlation matrices




```
subset = tips[['total_bill', 'tip', 'size']]
correlation_matrix = subset.corr()

f = sns.heatmap(correlation_matrix)
f.set(title = 'Correlation Matrix')
```

Customizing your chart

You can use matplotlib to set your title and axes labels

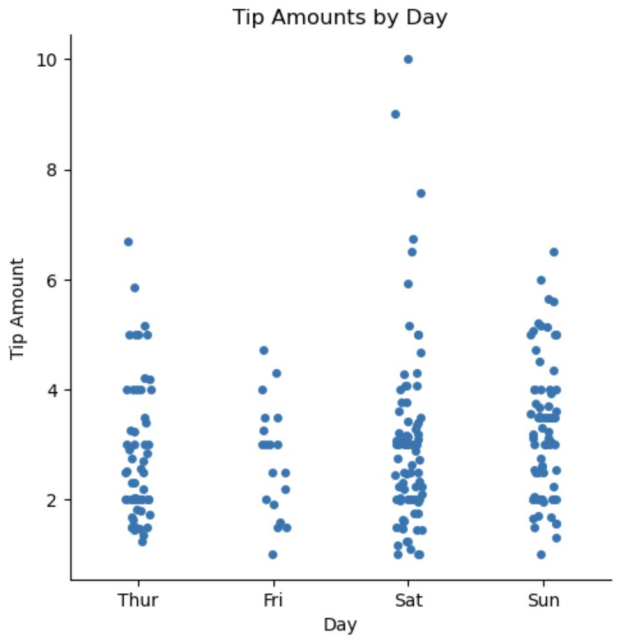
You can also use the .set() method in seaborn

Some customizations are easier to do in seaborn, others are in easier in matplotlib. You can use whichever package you want

Customizing your chart

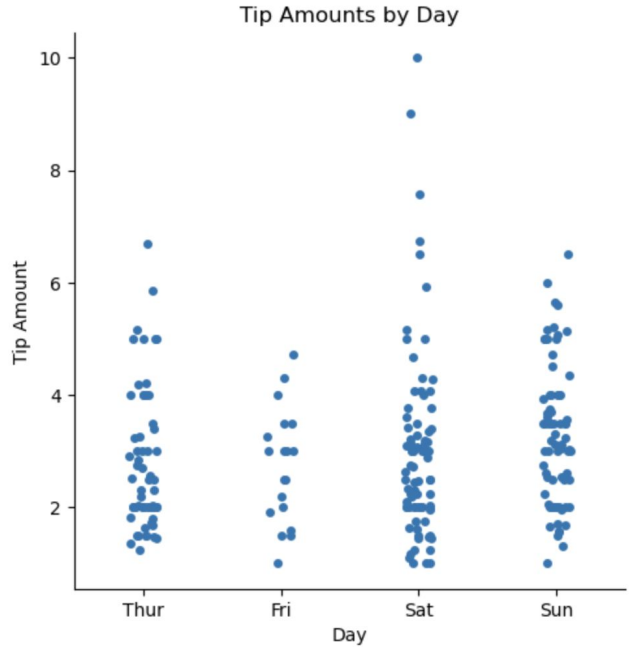
```
sns.catplot(data = tips, x = 'day', y = 'tip')
plt.title("Tip Amounts by Day")
plt.xlabel("Day")
plt.ylabel("Tip Amount")
```

```
Text(13.819444444444445, 0.5, 'Tip Amount')
```




```
f = sns.catplot(data = tips, x = 'day', y = 'tip')
f.set(title = "Tip Amounts by Day", xlabel = "Day", ylabel = "Tip Amount")
```

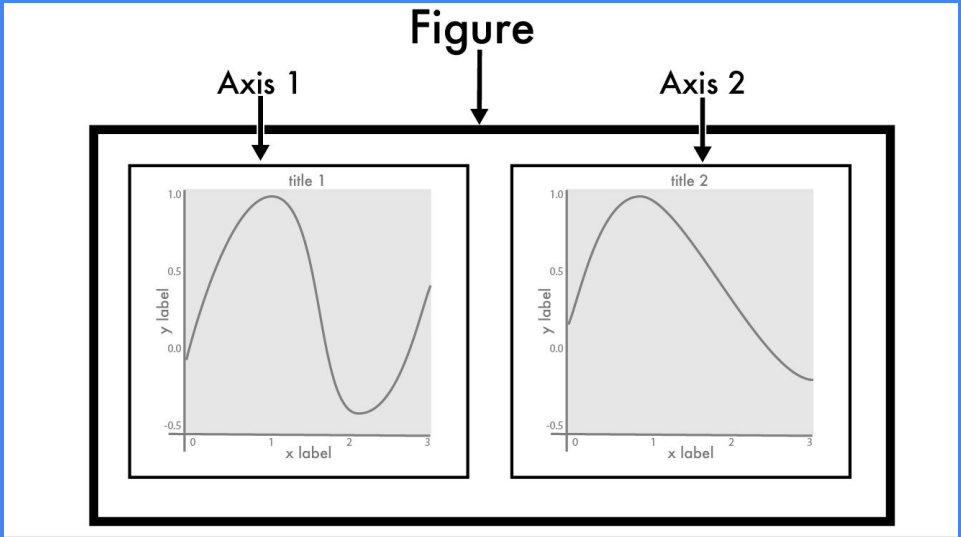
```
<seaborn.axisgrid.FacetGrid at 0x14b51ef10>
```



Subplots

If you are trying to analyze the results of two plots together, it’s often helpful to create subplots

- A subplot puts multiple axes on the same figure



Subplots

1) Generate subplots

```
fig, axs = plt.subplots(2)
```

2) Add charts to axes

Matplotlib

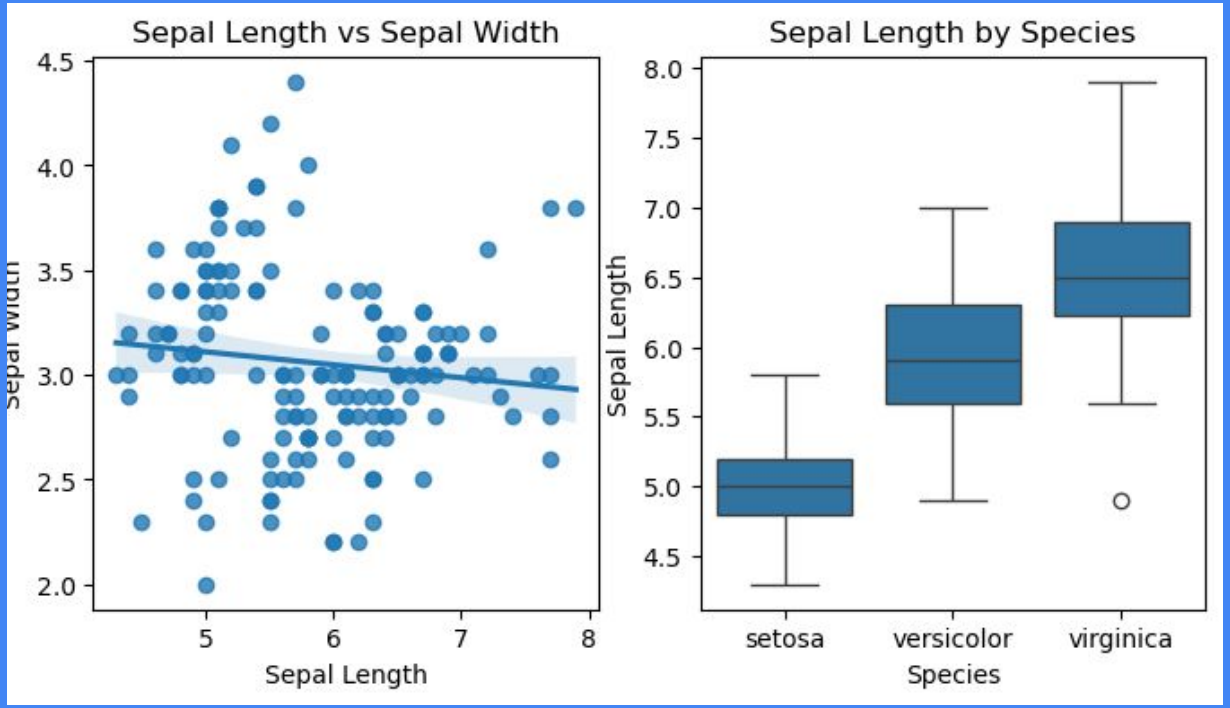
```
axs[0].plot(x, y)
```

```
axs[1].scatter(x, -y)
```

Seaborn

```
sns.boxplot(  y="b", x= "a", data=df,  orient='v' , ax=axes[0])
sns.boxplot(  y="c", x= "a", data=df,  orient='v' , ax=axes[1])
```

Lets recreate this chart



Lets recreate this chart

Step 1: Create subplots

```
fig, axes = plt.subplots(1, 2, figsize = (8, 4))
```

Lets recreate this chart

Step 1: Create subplots

```
fig, axes = plt.subplots(1, 2, figsize = (8, 4))
```

Step 2: Add plot to first axis

```
fig1 = sns.regplot(data = iris, x = "sepal_length",
                   y = "sepal_width", ax = axes[0])
```

Lets recreate this chart

Step 1: Create subplots

```
fig, axes = plt.subplots(1, 2, figsize = (8, 4))
```

Step 2: Add plot to first axis

```
fig1 = sns.regplot(data = iris, x = "sepal_length",
                   y = "sepal_width", ax = axes[0])
```

Step 3: Add plot to second axis

```
fig2 = sns.boxplot(data = iris, x = "species",
                   y = "sepal_length", ax = axes[1])
```

Lets recreate this chart

Step 1: Create subplots

```
fig, axes = plt.subplots(1, 2, figsize = (8, 4))
```

Step 2: Add plot to first axis

```
fig1 = sns.regplot(data = iris, x = "sepal_length",
                   y = "sepal_width", ax = axes[0])
```

Step 3: Add plot to second axis

```
fig2 = sns.boxplot(data = iris, x = "species",
                   y = "sepal_length", ax = axes[1])
```

Step 4: Add labels

```
fig1.set(xlabel = "Sepal Length", ylabel = "Sepal Width",
         title = "Sepal Length vs Sepal Width")
_ = fig2.set(xlabel = "Species", ylabel = "Sepal Length",
             title = "Sepal Length by Species")
```