An Introduction to seaborn

Dr. Atif Tahir, Dr. Abdul Aziz and Dr. Nuaman Durrani

Introduction

- Seaborn is a complement to matplotlib and not a replacement for it
- It is built on top of matplotlib
- Seaborn comes with a number of customized themes and a high-level interface for controlling the look of matplotlib figures

Cont…

- Functions that visualize matrices of data and use clustering algorithms to discover structure in those matrices
- A function to plot statistical time series data with flexible estimation and representation of uncertainty around the estimate
- High-level abstractions for structuring grids of plots that let you easily build complex visualizations

- For reference and more details, please visit: https://seaborn.pydata.org/introduction.html

Commands to Play with Seaborn

1- Show seaborn Plots:

plt.show() to display seaborn plots

2- Use seaborn With Matplotlib Defaults:

set(), set_style(), set_context(), and set_palette()

3- Use seaborn’s colors as a colormap in matplotlib:

color_palette() with n_colors as the number of colors with the argument

4- Scale Seaborn Plots For Other Contexts:

set_context() to control the plot elements.

The four predefined contexts are “paper”, “notebook”, “talk” and “poster”.

5- Set The Figure Size in Seaborn:

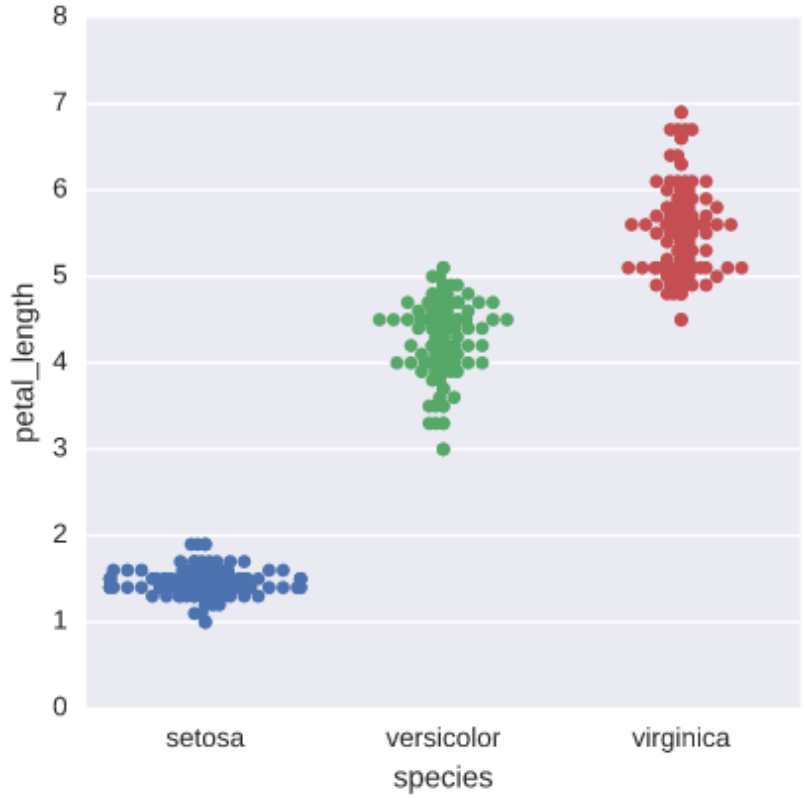
plt.subplots()

6- Add A Title:

set_title()

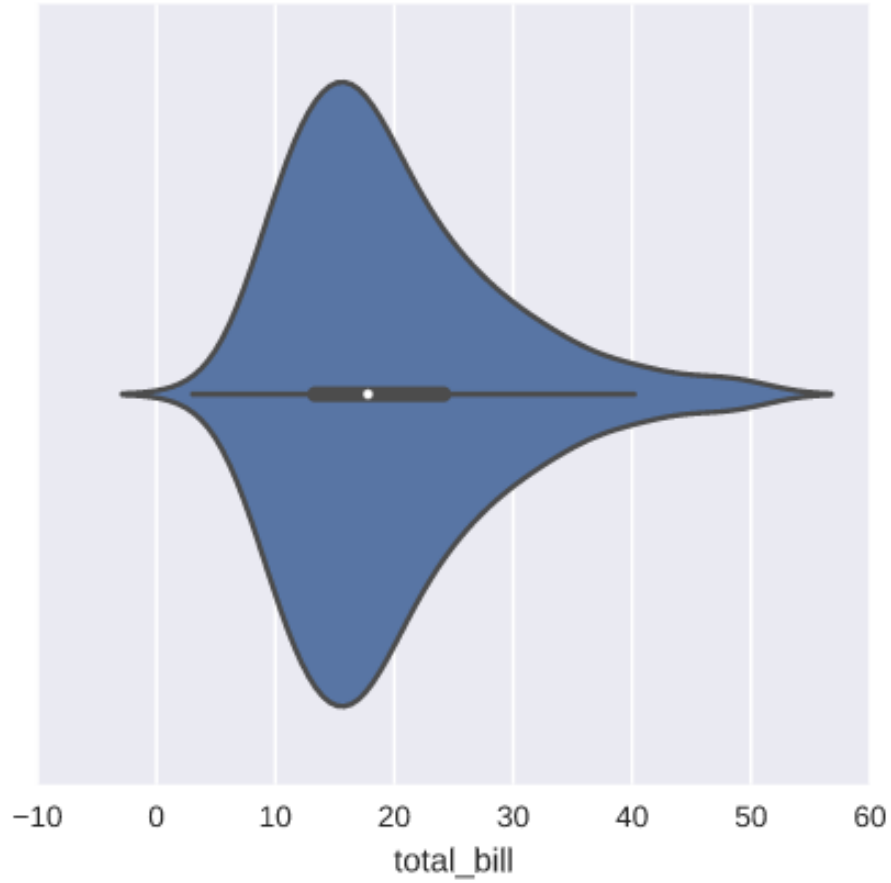
Loading A Built-in Seaborn Data Set

```
import seaborn as sns
import matplotlib.pyplot as plt
# Load iris data
        iris = sns.load_dataset("iris")
# Construct iris plot
        sns.swarmplot(x="species", y="petal_length", data=iris)
# Show plot
        plt.show()
```



Loading Your Data from Pandas DataFrame

```
import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

# Load in data

tips = pd.read_csv("https://raw.githubuserconten
data/master/tips.csv")

# Create violinplot

        sns.violinplot(x = "total_bill", data=tips)

# Show the plot

        plt.show()
```



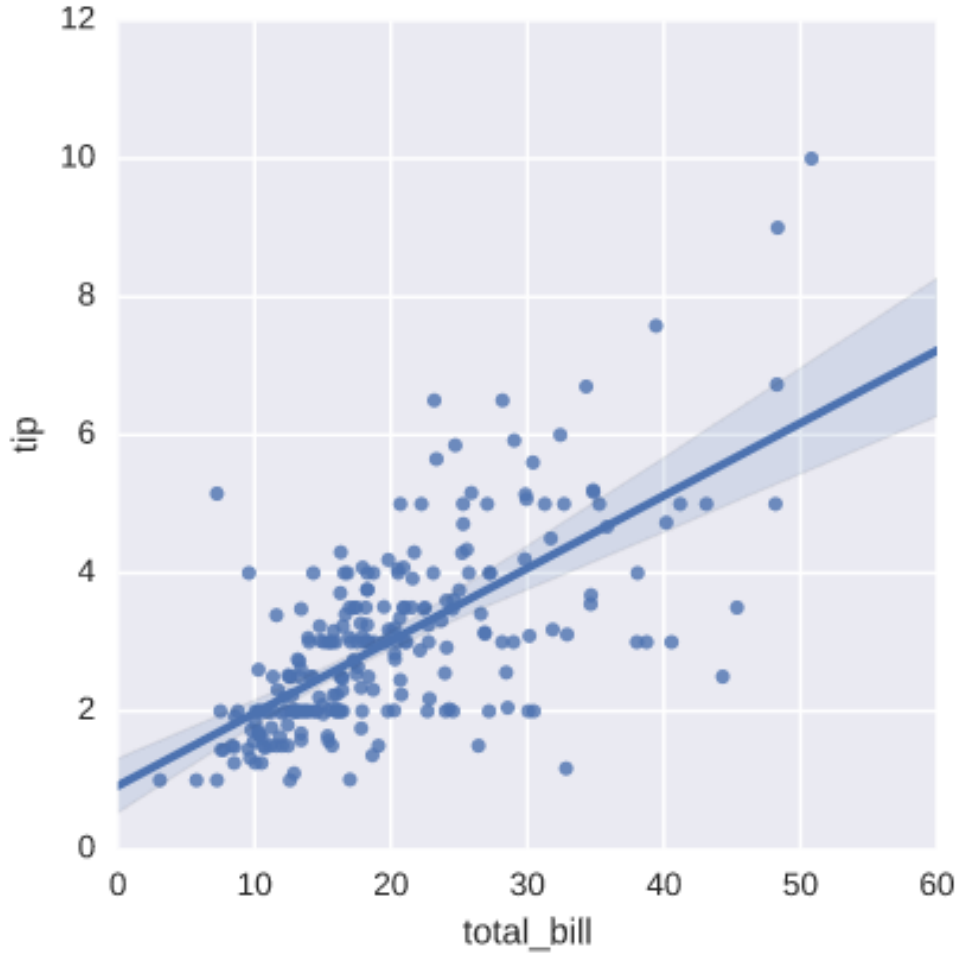
Regression Plot

- Many datasets contain multiple quantitative variables, and the goal of an analysis is often to relate those variables to each other

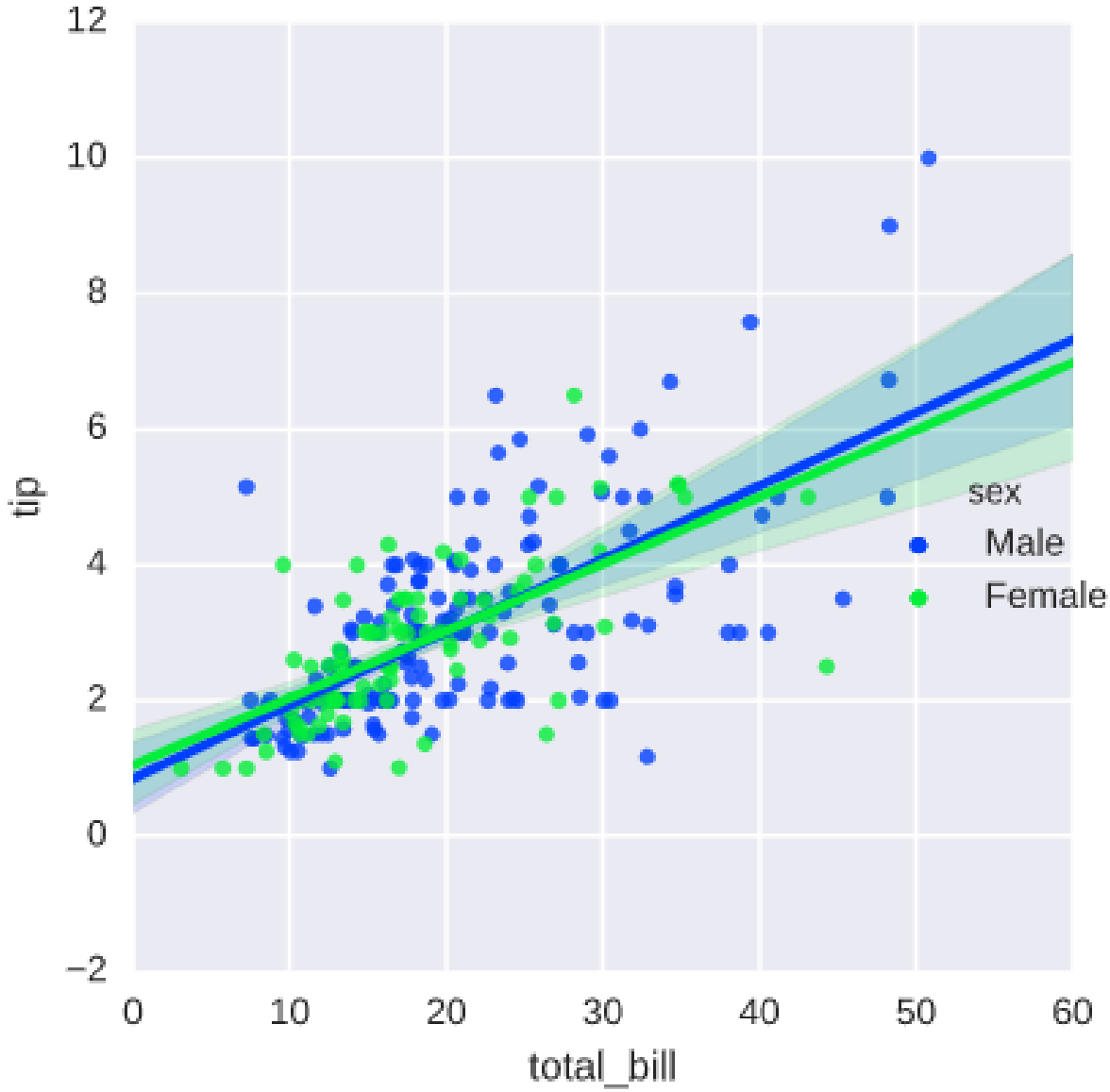
- to visualize a linear relationship as determined through regression

1- Regression Plot

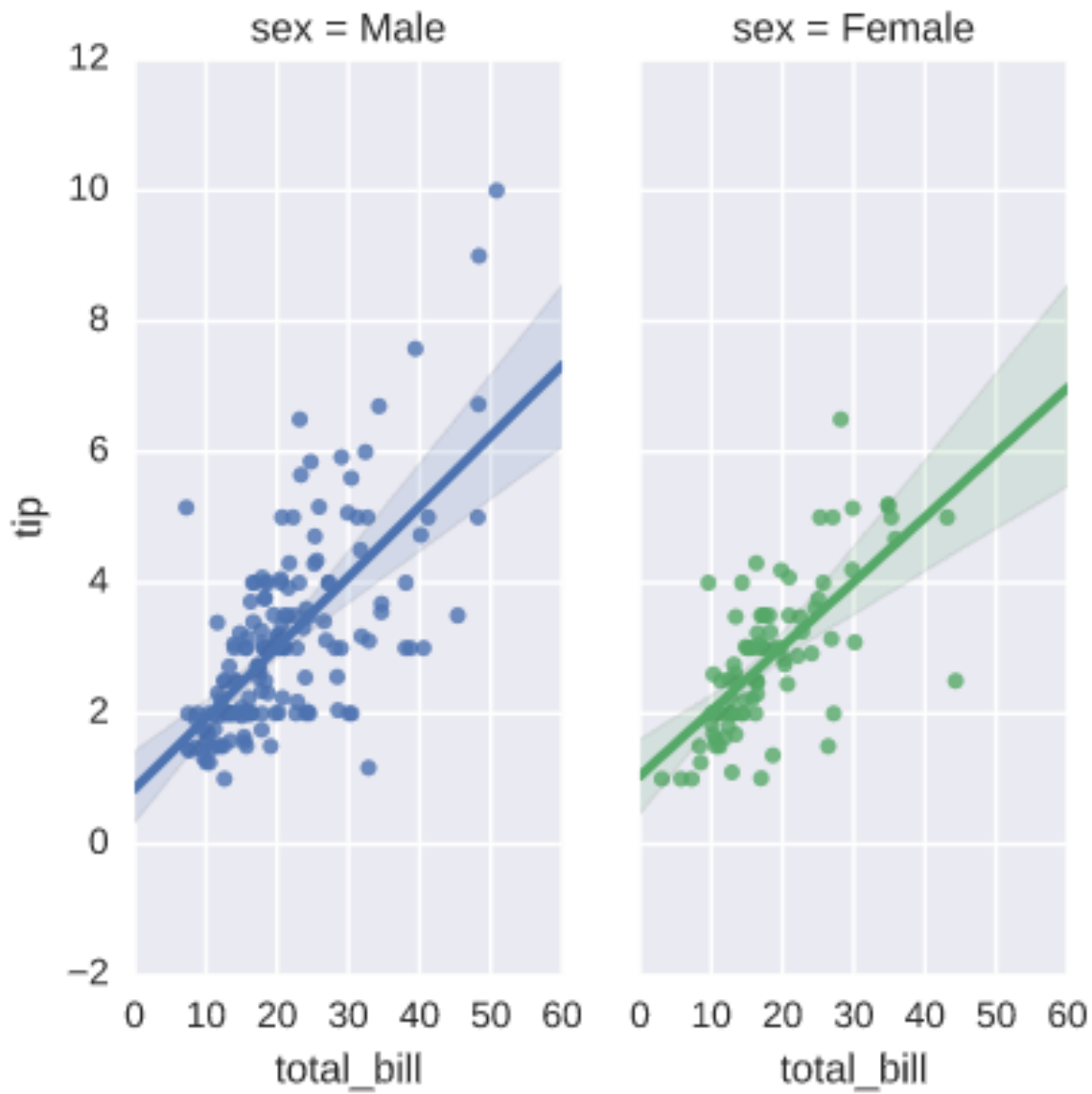
```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.lmplot(x='total_bill', y='tip', data=tips)
plt.show()
```



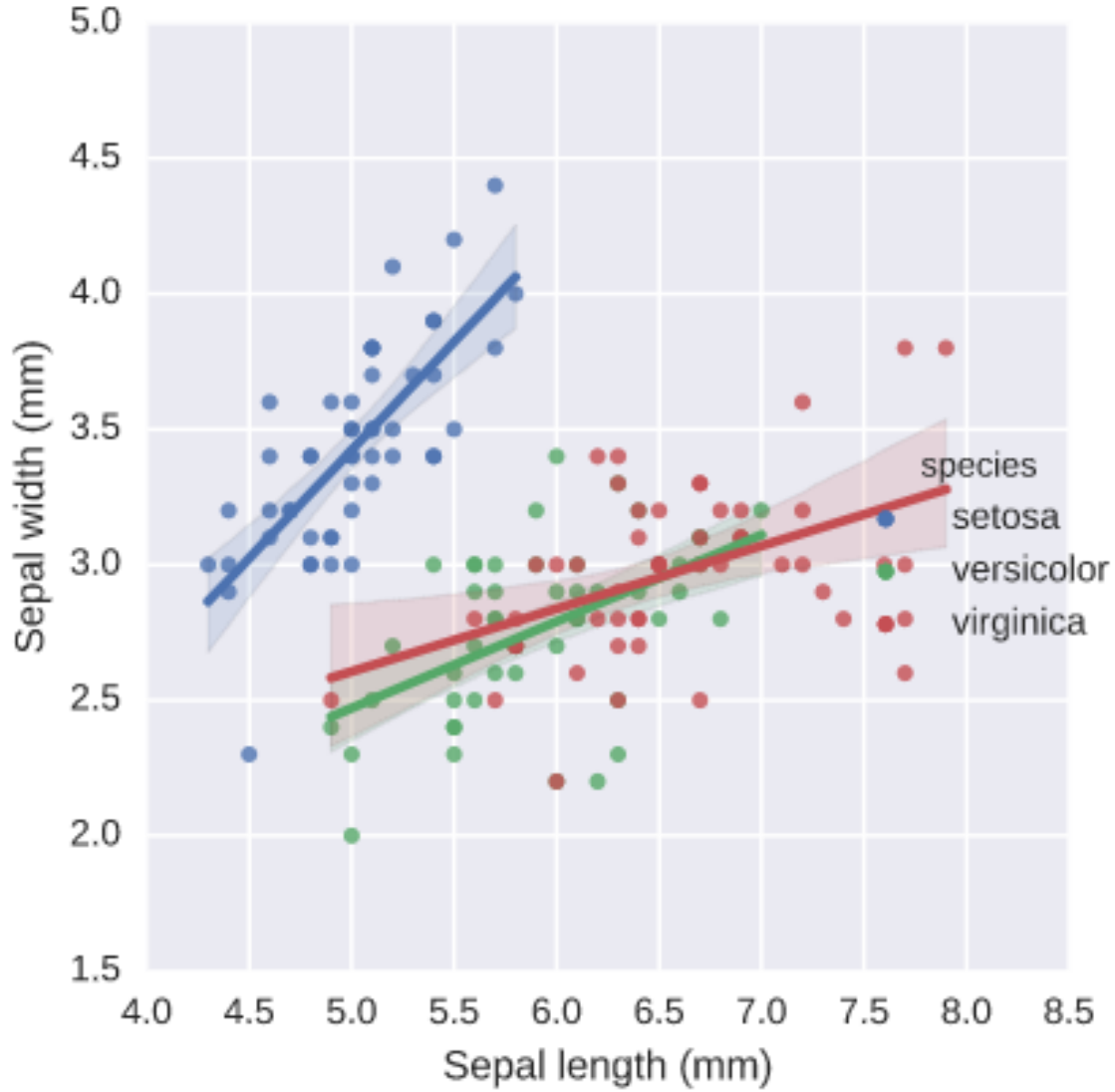
```
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset('tips')sns.

lmplot(x='total_bill', y='tip', hue='sex', data=tips, palette = 'bright')

plt.show()
```


12
10
8
6
4
2
0
−2
tip
0
10
20
30
40
50
60
total_bill
sex
Male
Female

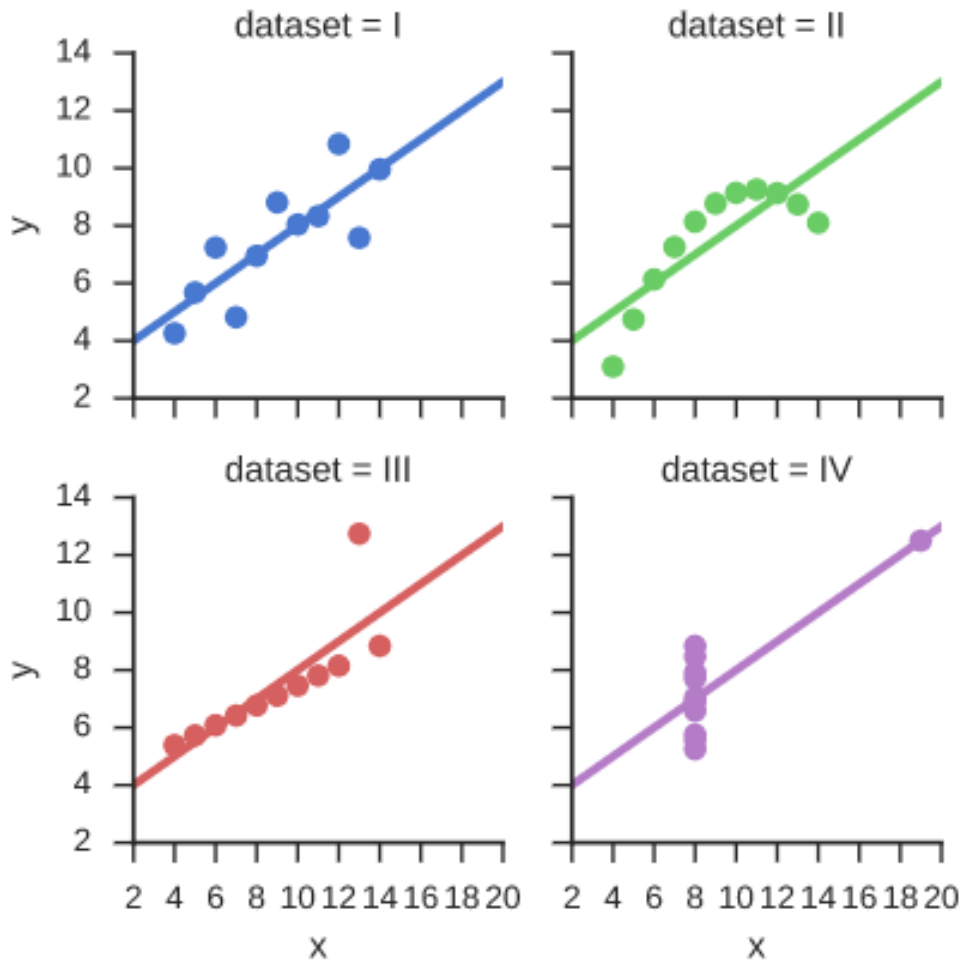
```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')sns.
lmplot(x='total_bill', y='tip', hue='sex', data=tips, col = 'sex')
plt.show()
```


sex = Male
sex = Female
tip
total_bill
total_bill
12
10
8
6
4
2
0
−2
0
10
20
30
40
50
60

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()# Load the example tips dataset
iris = sns.load_dataset("iris")
# Plot tip as a function of toal bill across days
g = sns.lmplot(x="sepal_length", y="sepal_width",
hue="species",truncate=True, size=5, data=iris)
# Use more informative axis labels than are provided by default
g.set_axis_labels("Sepal length (mm)", "Sepal width (mm)")
plt.show()
```

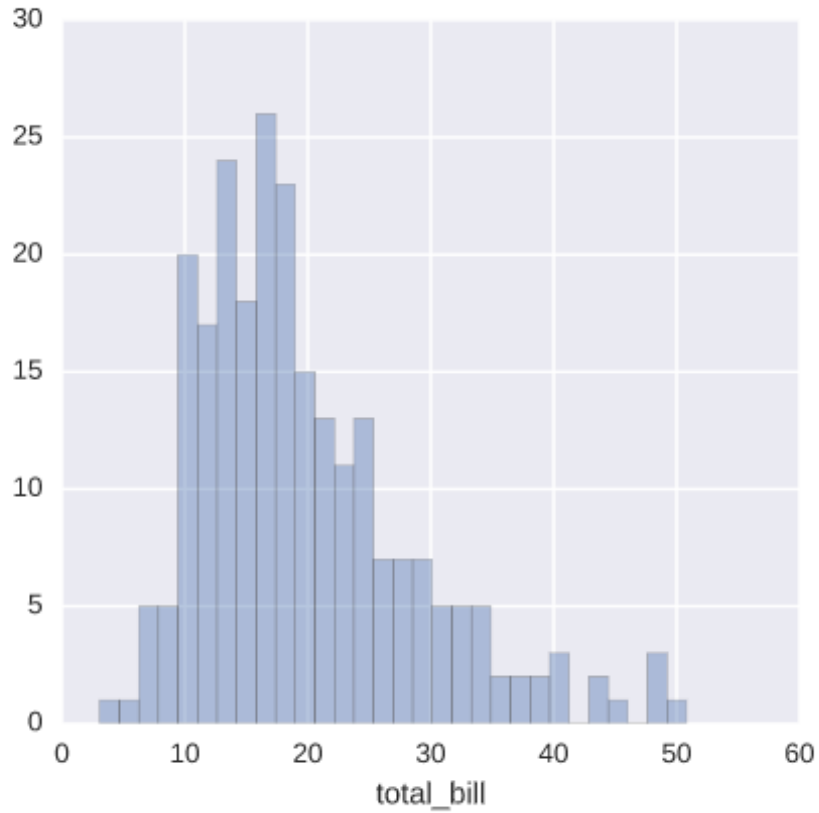

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
Sepal width (mm)
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
Sepal length (mm)
species
setosa
versicolor
virginica

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="ticks")
# Load the example dataset for Anscombe's quartet
df = sns.load_dataset("anscombe")
# Show the results of a linear regression within each dataset
sns.lmplot(x="x", y="y", col="dataset", hue="dataset", data=df,col_wrap=2,
           ci=None, palette="muted", size=4,scatter_kws={"s": 50, "alpha": 1})
plt.show()
```

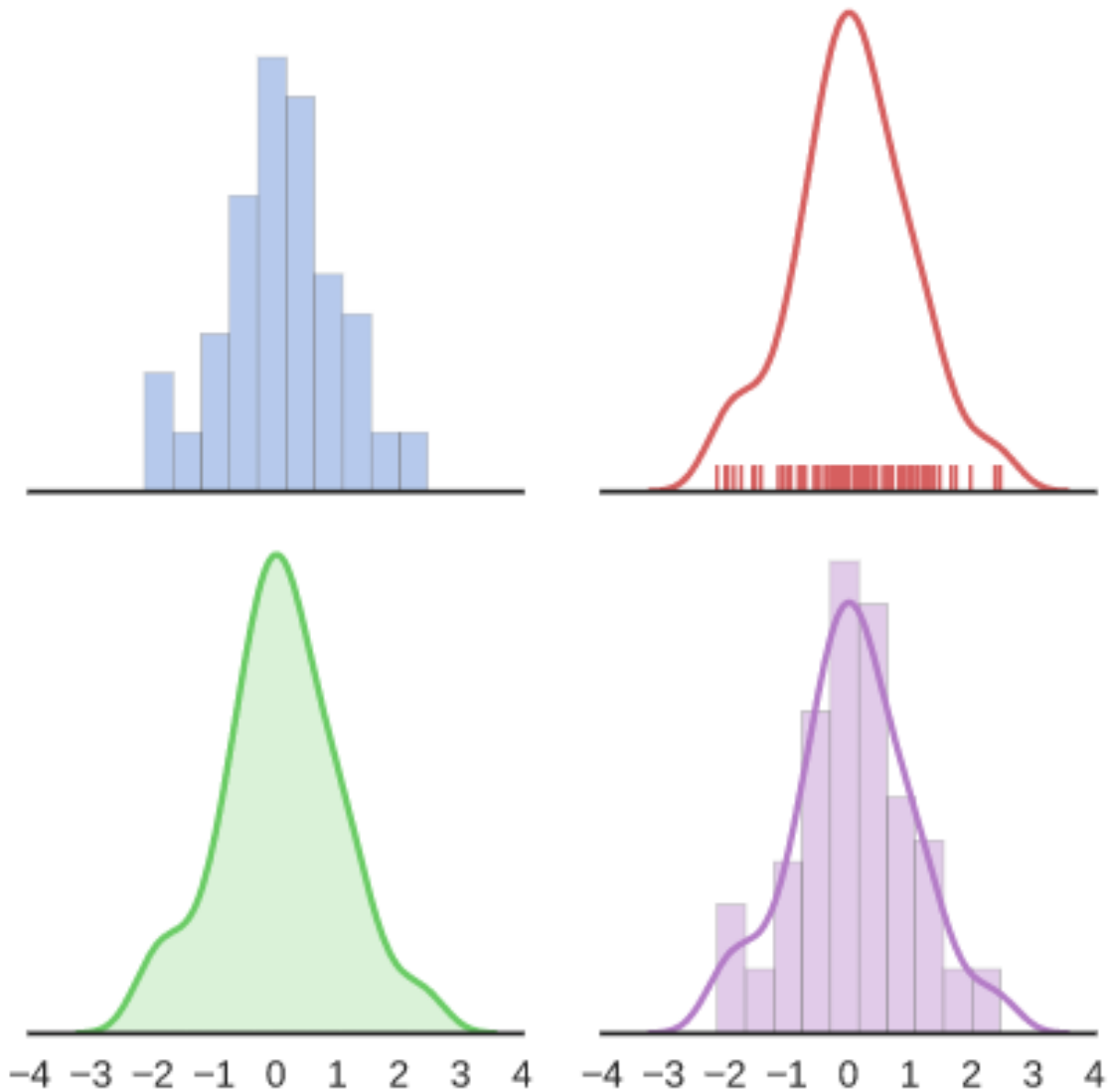

dataset = I
dataset = II
dataset = III
dataset = IV
y
x
14
12
10
8
6
4
2
2 4 6 8 10 12 14 16 18 20

2- Distribution Plot

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.distplot(tips['total_bill'], kde=False, bins=30)
plt.show()
```



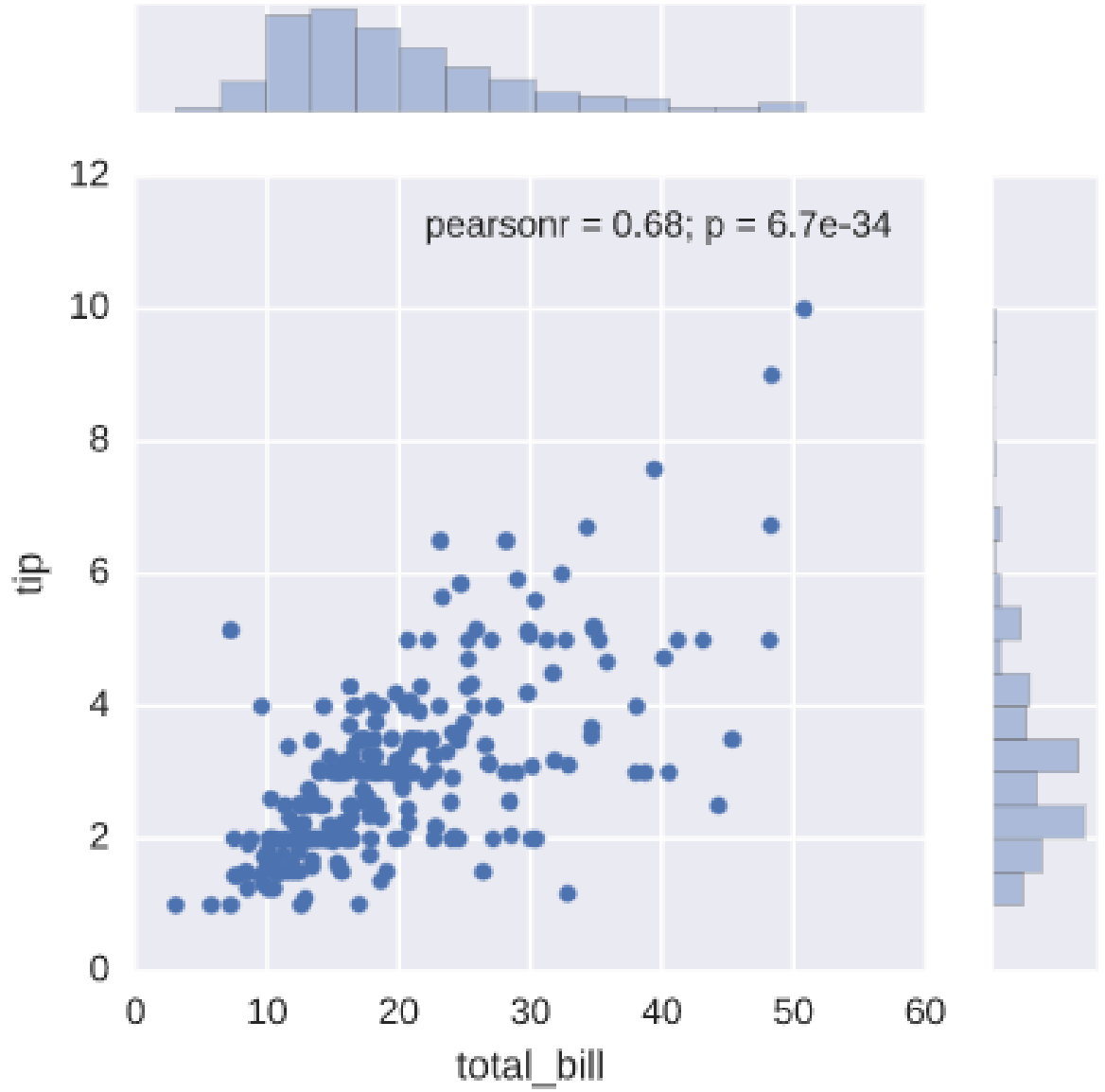
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="white", palette="muted", color_codes=True)
rs = np.random.RandomState(10)
# Set up the matplotlib figure
f, axes = plt.subplots(2, 2, figsize=(7, 7), sharex=True)
sns.despine(left=True)
# Generate a random univariate dataset
d = rs.normal(size=100)
# Plot a simple histogram with binsize determined automatically
sns.distplot(d, kde=False, color="b", ax=axes[0, 0])
# Plot a kernel density estimate and rug plot
sns.distplot(d, hist=False, rug=True, color="r", ax=axes[0, 1])
# Plot a filled kernel density estimate
sns.distplot(d, hist=False, color="g", kde_kws={"shade": True}, ax=axes[1, 0])
# Plot a historgram and kernel density estimate
sns.distplot(d, color="m", ax=axes[1, 1])
plt.setp(axes, yticks=[])
plt.tight_layout()
plt.show()
```


−4 −3 −2 −1 0 1 2 3 4
−4 −3 −2 −1 0 1 2 3 4

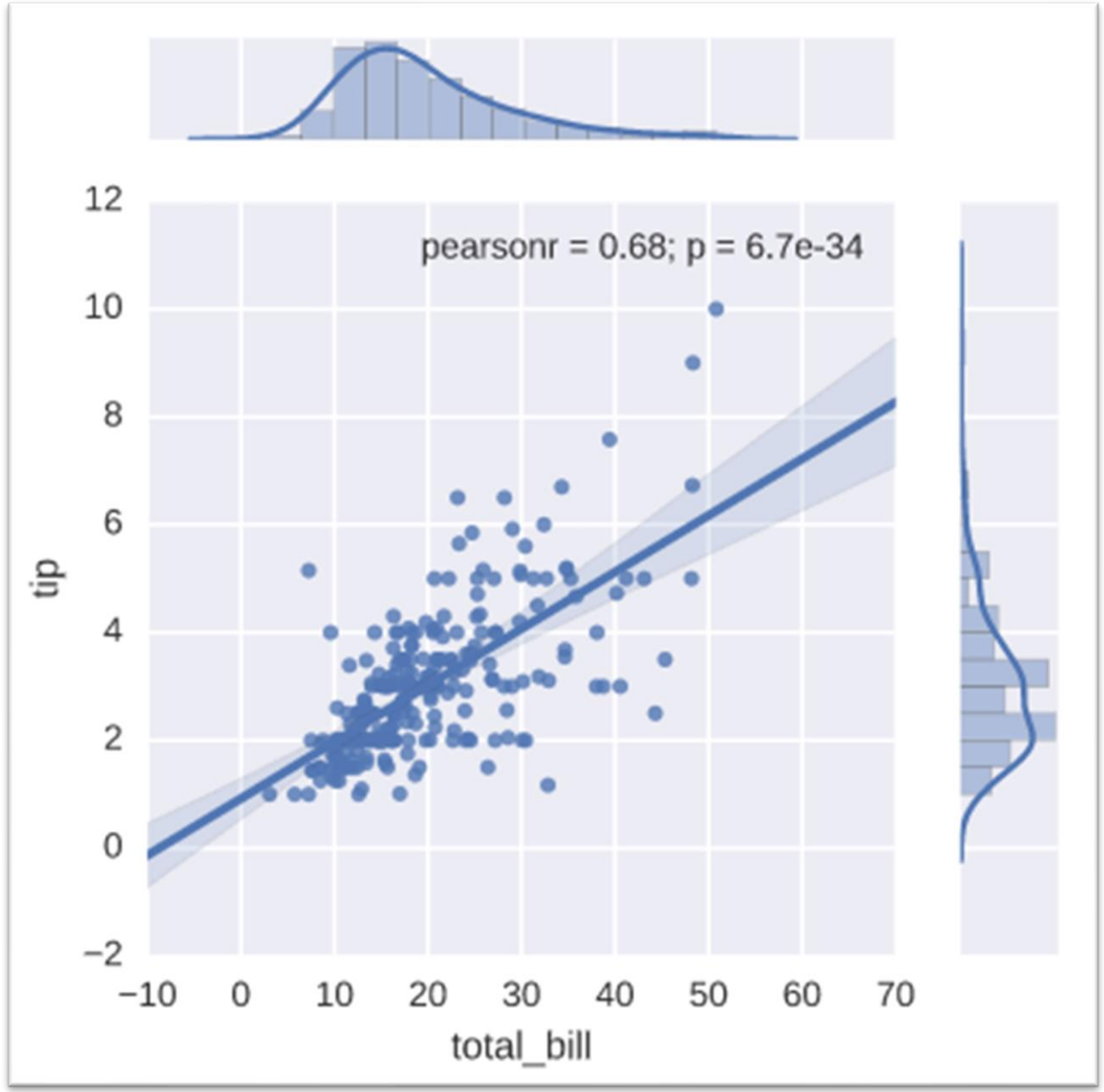
3- Joint Plot

- We can combine different types of plots together to create a join plot such as scatter, hex, regression…etc.

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.jointplot(x= 'total_bill', y = 'tip', data=tips, kind='scatter‘)
plt.show()
```


pearsonr = 0.68; p = 6.7e-34
tip
total_bill
0
2
4
6
8
10
12
0
10
20
30
40
50
60

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.jointplot(x= 'total_bill', y = 'tip', data=tips, kind='regression')
plt.show()
```


pearsonr = 0.68; p = 6.7e-34
tip
total_bill
12
10
8
6
4
2
0
−2
−10
0
10
20
30
40
50
60
70

Pair Plot

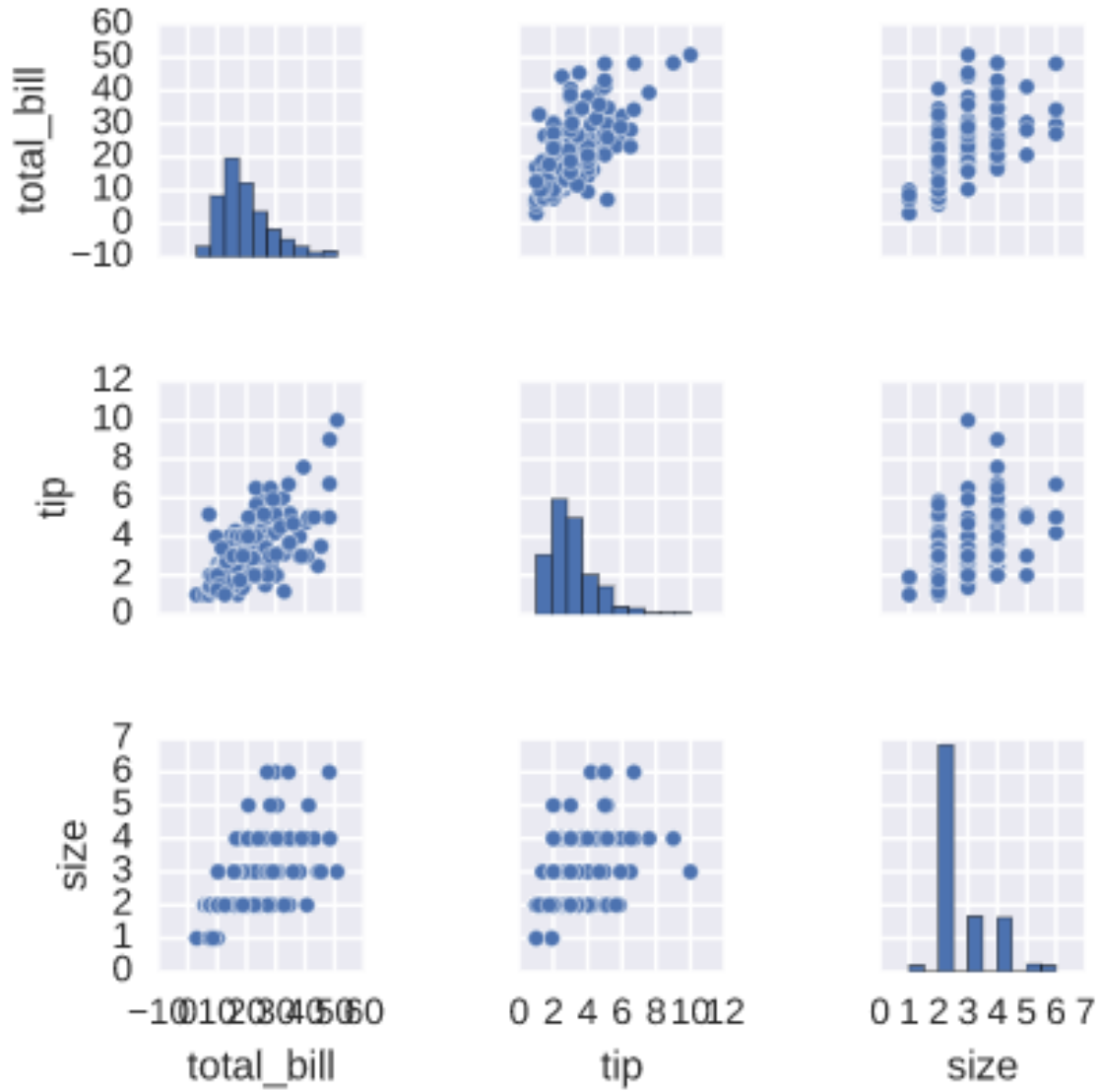
- **Pair Plots** are a really simple (one-line-of-code simple!) way to visualize relationships between each variable

- It produces a matrix of relationships between each variable in your data for an instant examination of our data

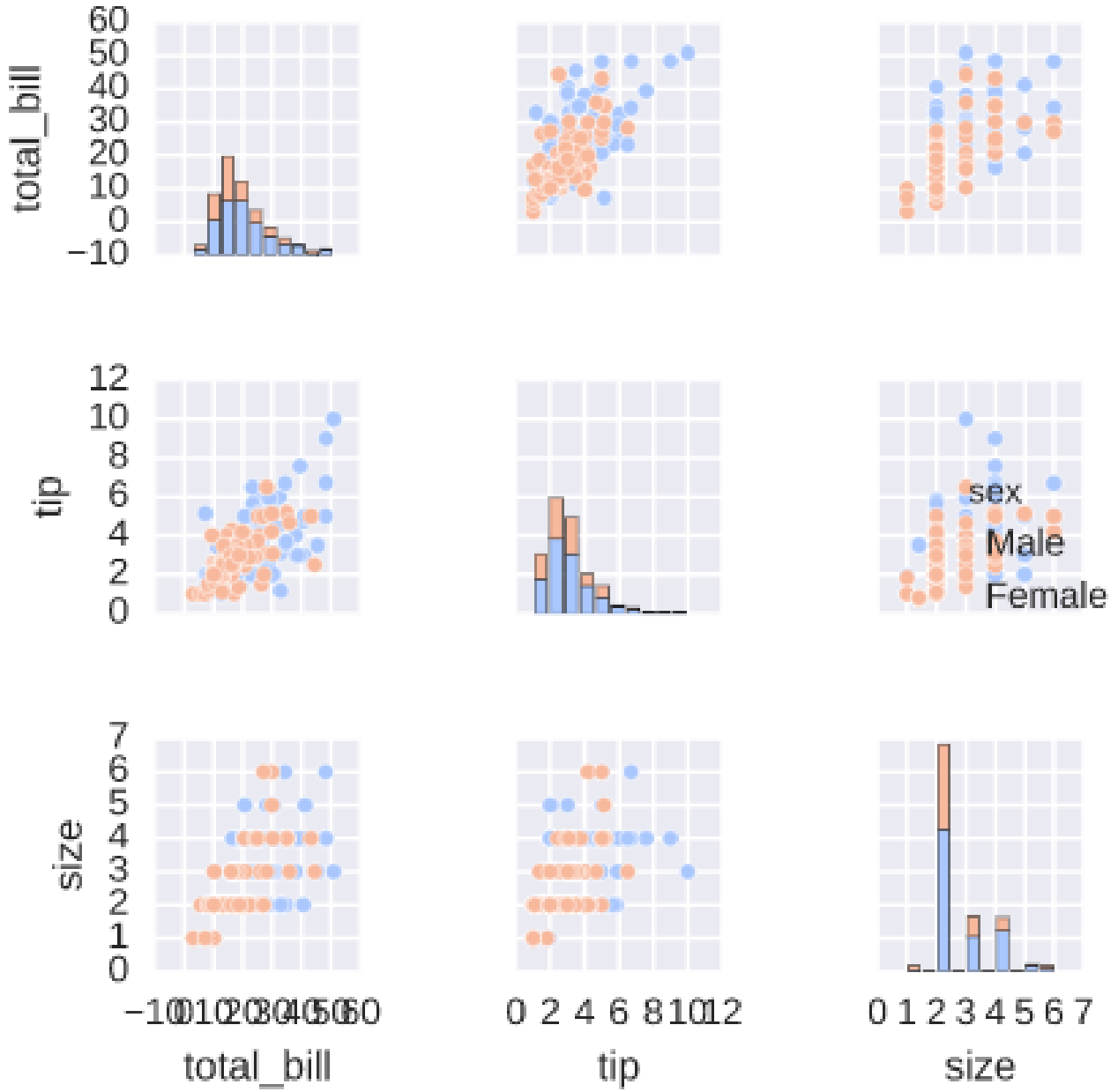
- It can also be a great jumping off point for determining types of regression analysis to use.

4- Pair Plot

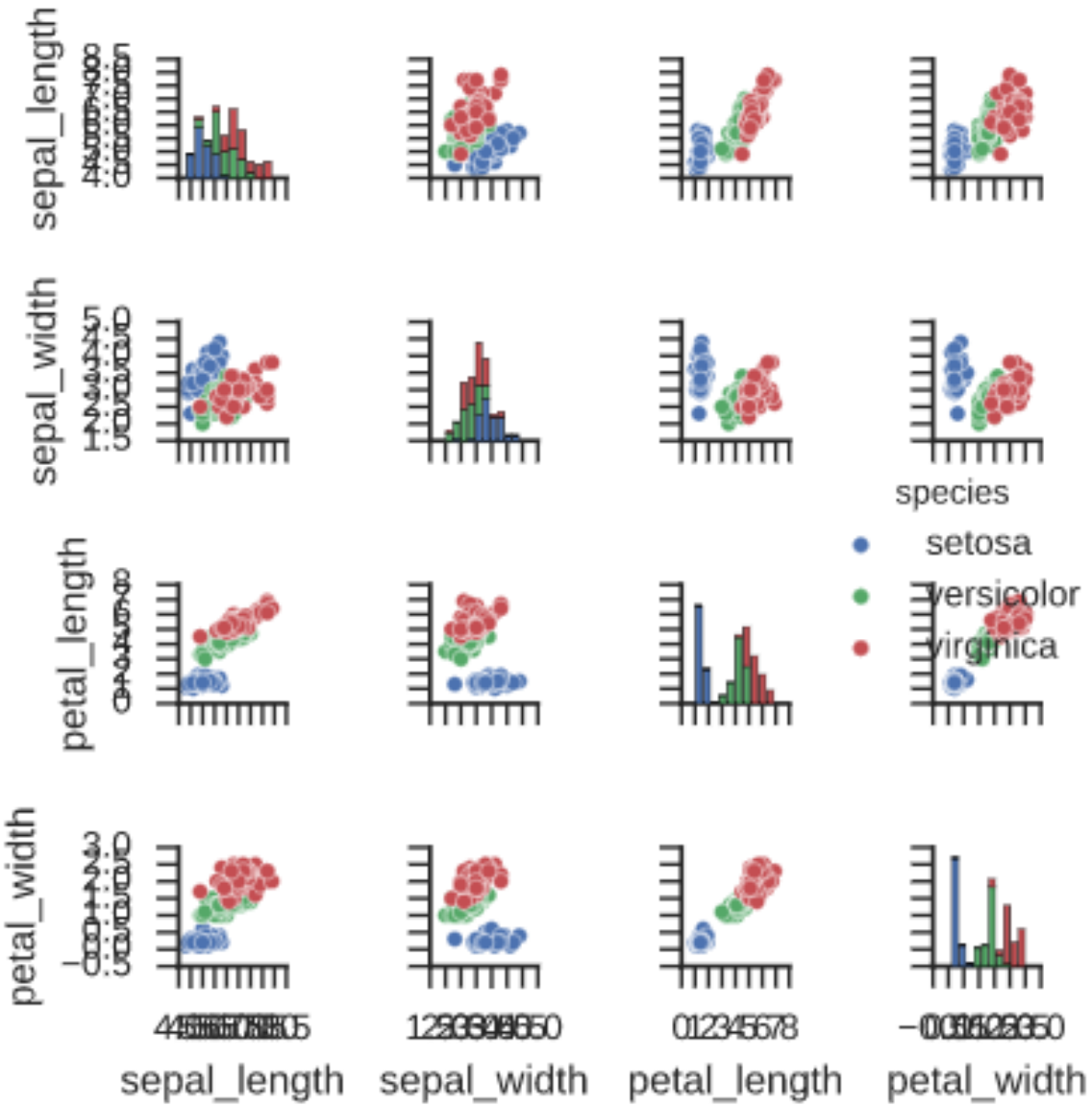
```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.pairplot(tips)
plt.show()
```


total_bill
tip
size
60
50
40
30
20
10
0
−10
12
10
8
6
4
2
0
7
6
5
4
3
2
1
0
−10 0 10 20 30 40 50 60
0 2 4 6 8 10 12
0 1 2 3 4 5 6 7
total_bill
tip
size

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.pairplot(tips, hue='sex', palette='coolwarm')
plt.show()
```

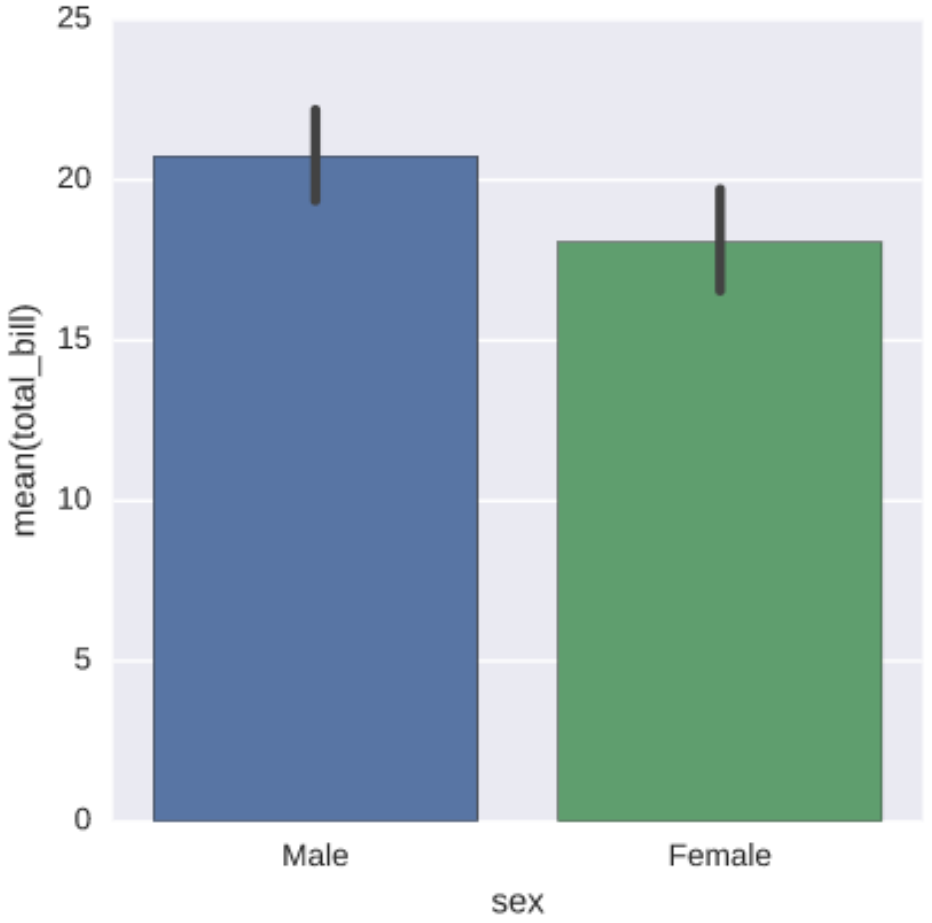

total_bill
tip
size
sex
Male
Female
60
50
40
30
20
10
0
−10
12
10
8
6
4
2
0
7
6
5
4
3
2
1
0
−10 0 1 0 2 0 3 0 4 0 5 0 6 0
0 2 4 6 8 10 12
0 1 2 3 4 5 6 7
total_bill
tip
size

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="ticks")
df = sns.load_dataset("iris")
sns.pairplot(df, hue="species")
plt.show()
```

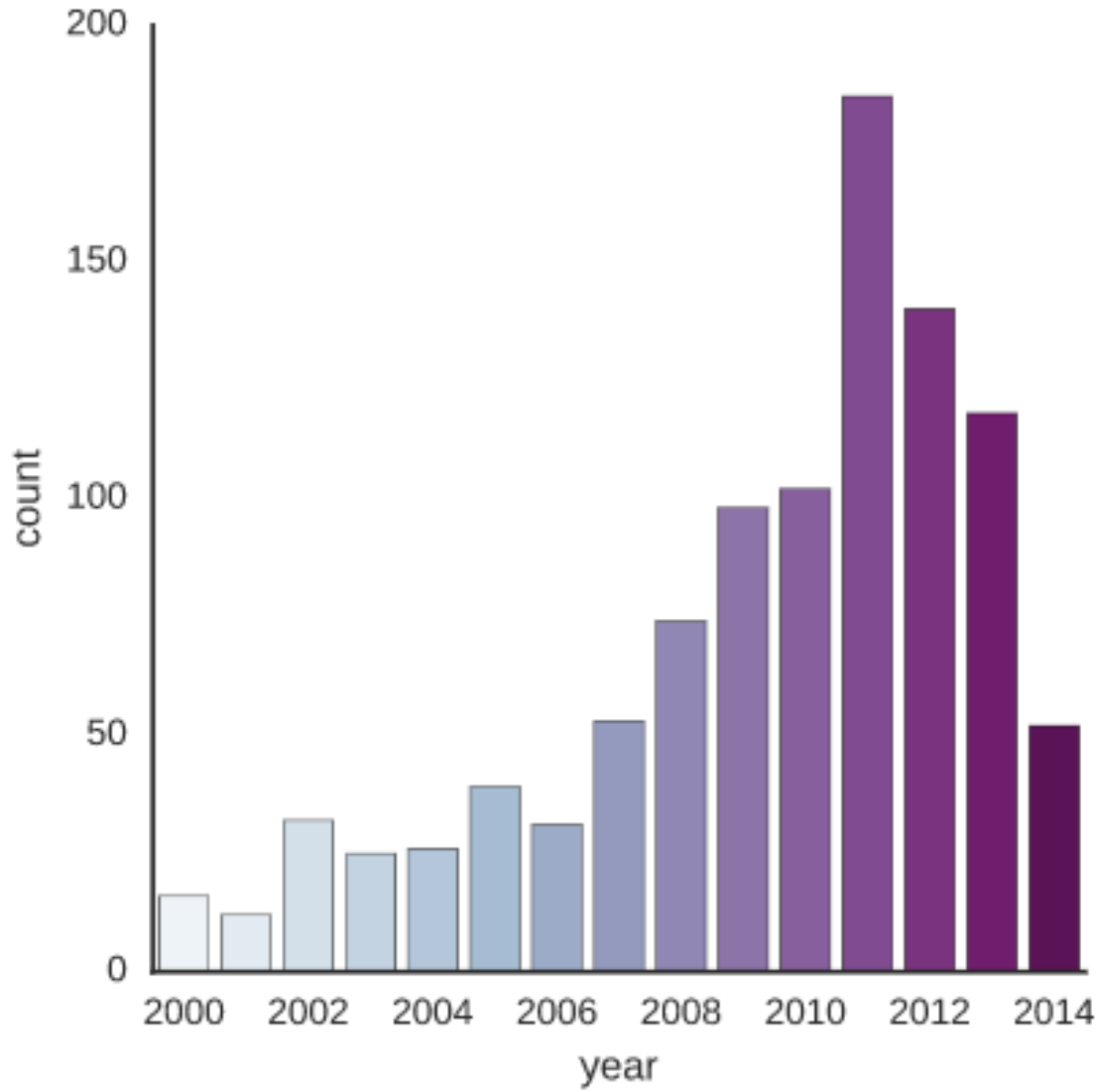

sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

5- Bar Plot

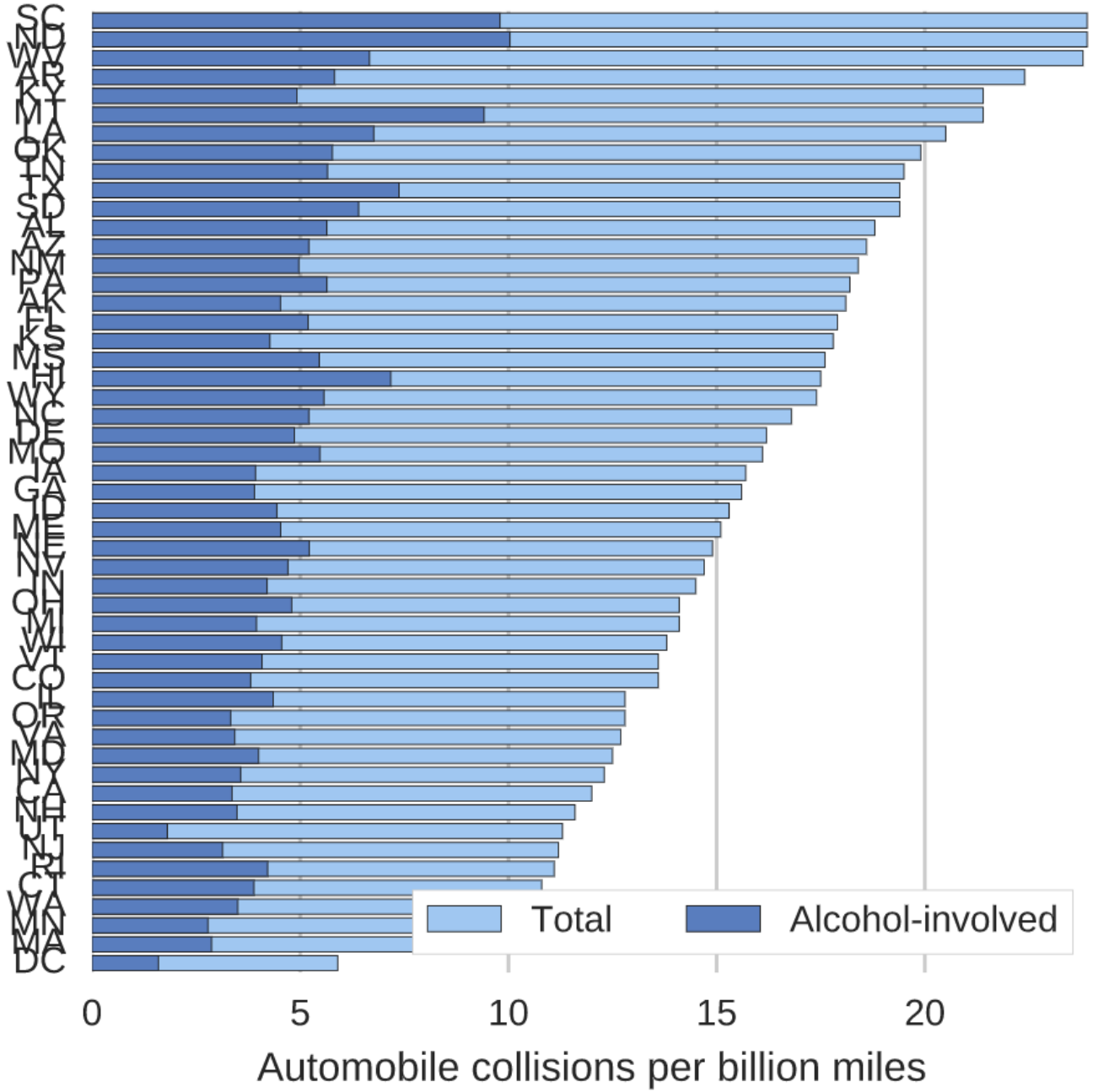
```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.barplot(x = 'sex', y='total_bill', data=tips)
plt.show()
```



```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="white")
# Load the example planets dataset
planets = sns.load_dataset("planets")
# Make a range of years to show categories with no observations
years = np.arange(2000, 2015)
# Draw a count plot to show the number of planets discovered each year
g = sns.factorplot(x="year", data=planets,
kind="count",palette="BuPu", size=6, aspect=1.5, order=years)
g.set_xticklabels(step=2)
plt.show()
```

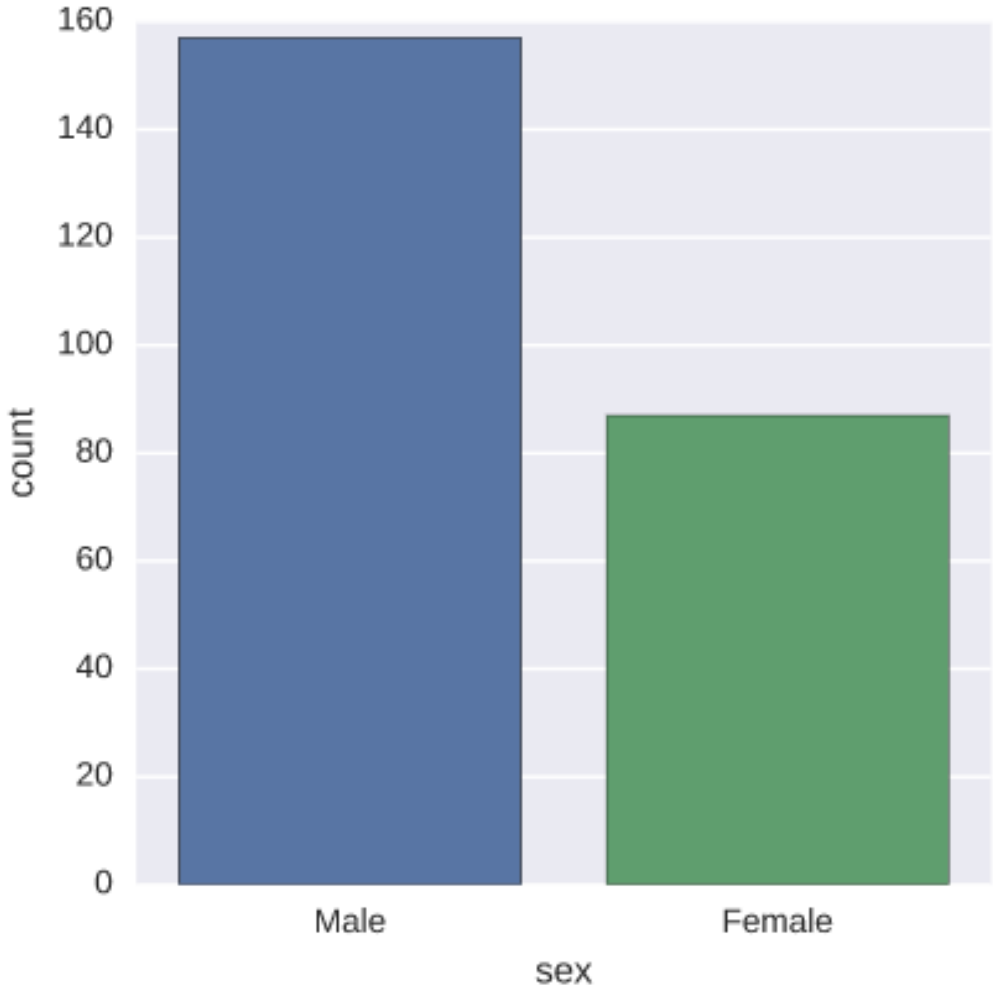

200
150
100
50
0
count
2000
2002
2004
2006
2008
2010
2012
2014
year

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="whitegrid")
# Initialize the matplotlib
figuref, ax = plt.subplots(figsize=(6, 15))
# Load the example car crash dataset
crashes = sns.load_dataset("car_crashes").sort_values("total", ascending=False)
# Plot the total crashes
sns.set_color_codes("pastel")
sns.barplot(x="total", y="abbrev", data=crashes,label="Total", color="b")
# Plot the crashes where alcohol was involved
sns.set_color_codes("muted")
sns.barplot(x="alcohol", y="abbrev", data=crashes,label="Alcohol-involved", color="b")
# Add a legend and informative axis label
ax.legend(ncol=2, loc="lower right", frameon=True)
ax.set(xlim=(0, 24), ylabel="",xlabel="Automobile collisions per billion miles")
sns.despine(left=True, bottom=True)
plt.show()
```


Total
Alcohol-involved
0
5
10
15
20
Automobile collisions per billion miles

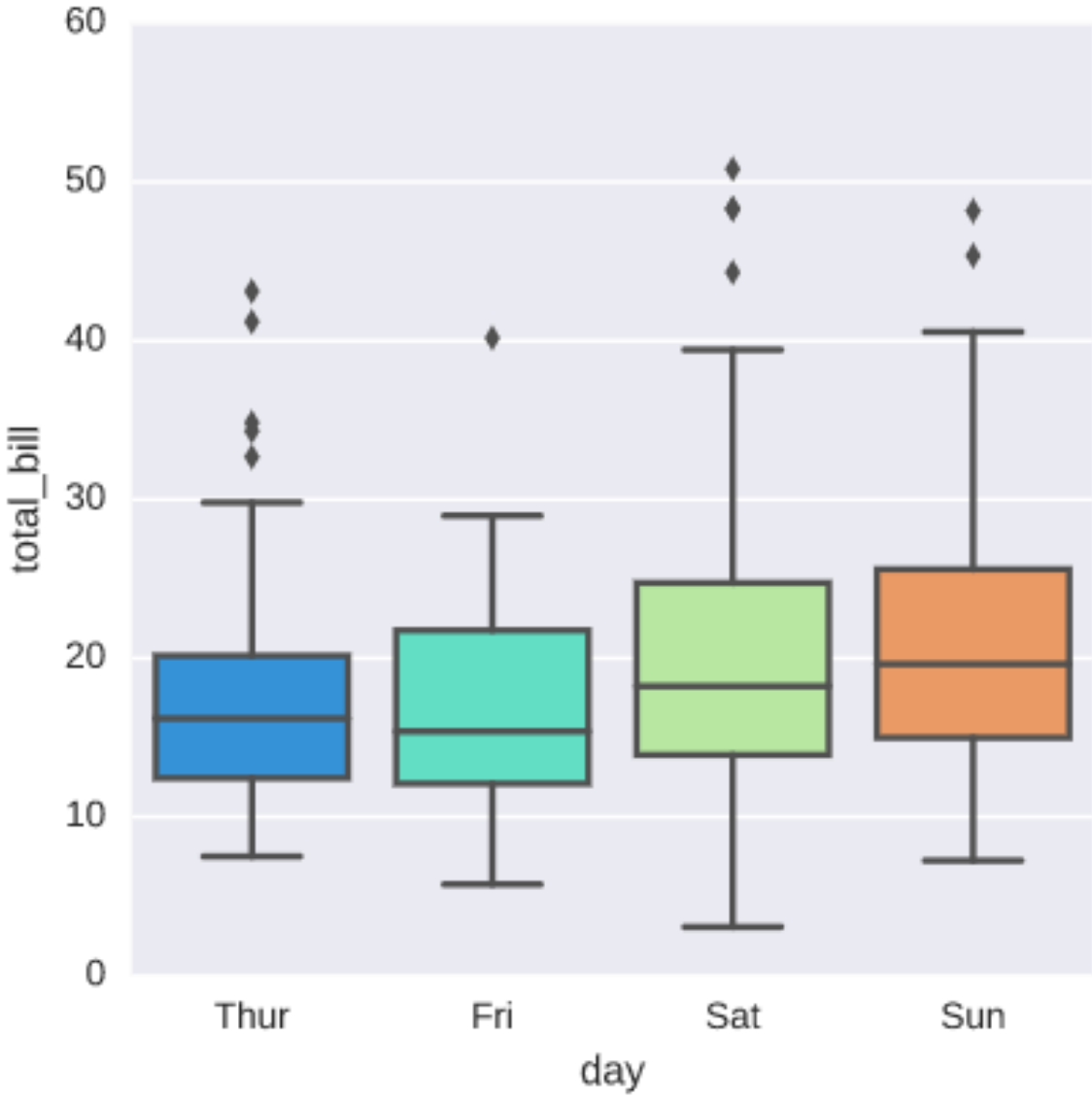
6- Count Plot

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.countplot(x = 'sex', data = tips)
plt.show()
```

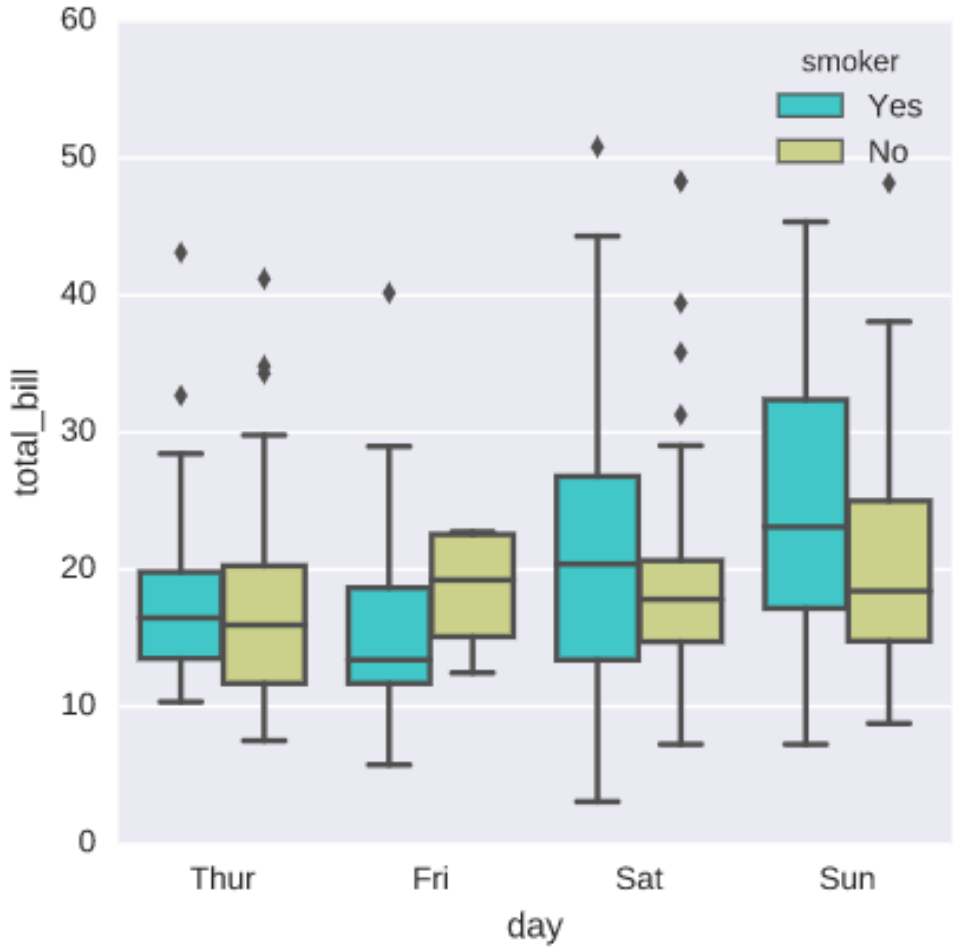


7- Box Plot

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.boxplot(x = 'day', y='total_bill', (
plt.show()
```



```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.boxplot(x = 'day', y='total_bill', hue='smoker', data=tips,
palette='rainbow')
plt.show()
```

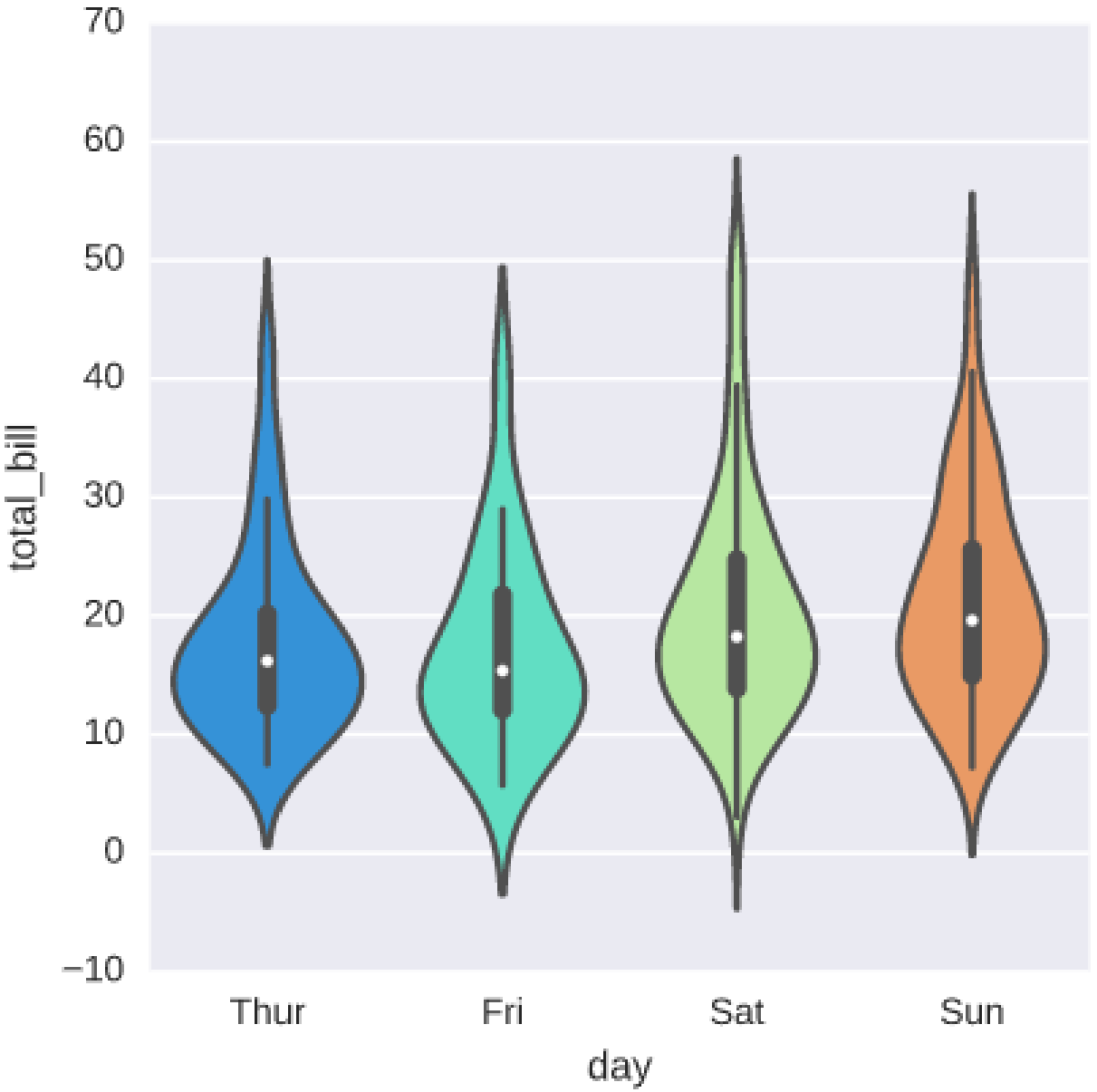


Violin Plot

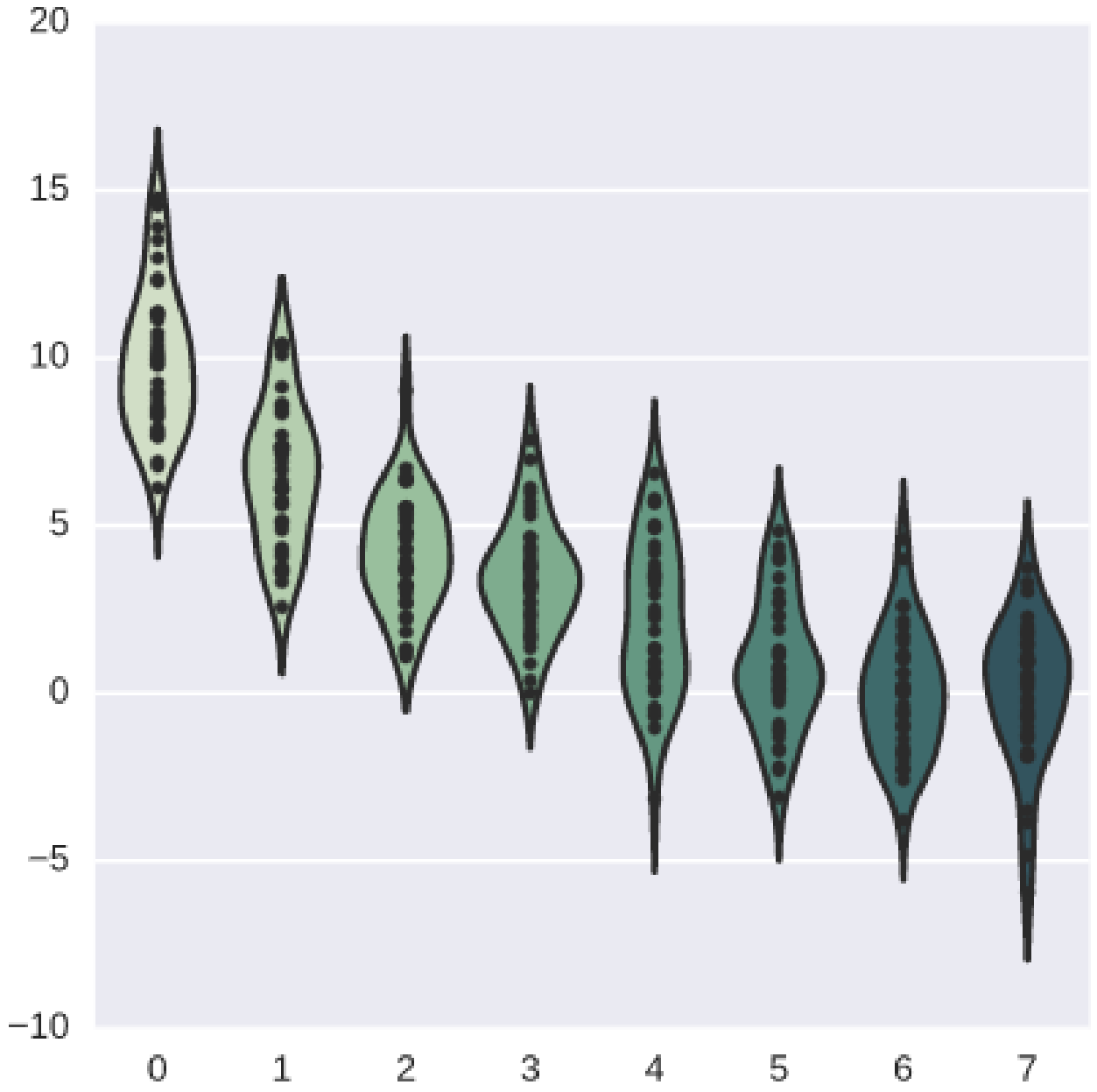
- A **violin plot** is a method of **plotting** numeric data
- It is similar to a **box plot**, with the addition of a rotated kernel density **plot** on each side
- **Violin plots** are similar to **box plots**, except that they also show the probability density of the data at different values, usually smoothed by a kernel density estimator.

8- Violin Plot

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.violinplot(x = 'day', y='total_bill
plt.show()
```



```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()
# Create a random dataset across several variables
rs = np.random.RandomState(0)
n, p = 40, 8
d = rs.normal(0, 2, (n, p))
d += np.log(np.arange(1, p + 1)) * -5 + 10
# Use cubehelix to get a custom sequential palette
pal = sns.cubehelix_palette(p, rot=-.5, dark=.3)
# Show each distribution with both violins and points
sns.violinplot(data=d, palette=pal, inner="points")
plt.show()
```


20
15
10
5
0
−5
−10
0
1
2
3
4
5
6
7

Heap Map

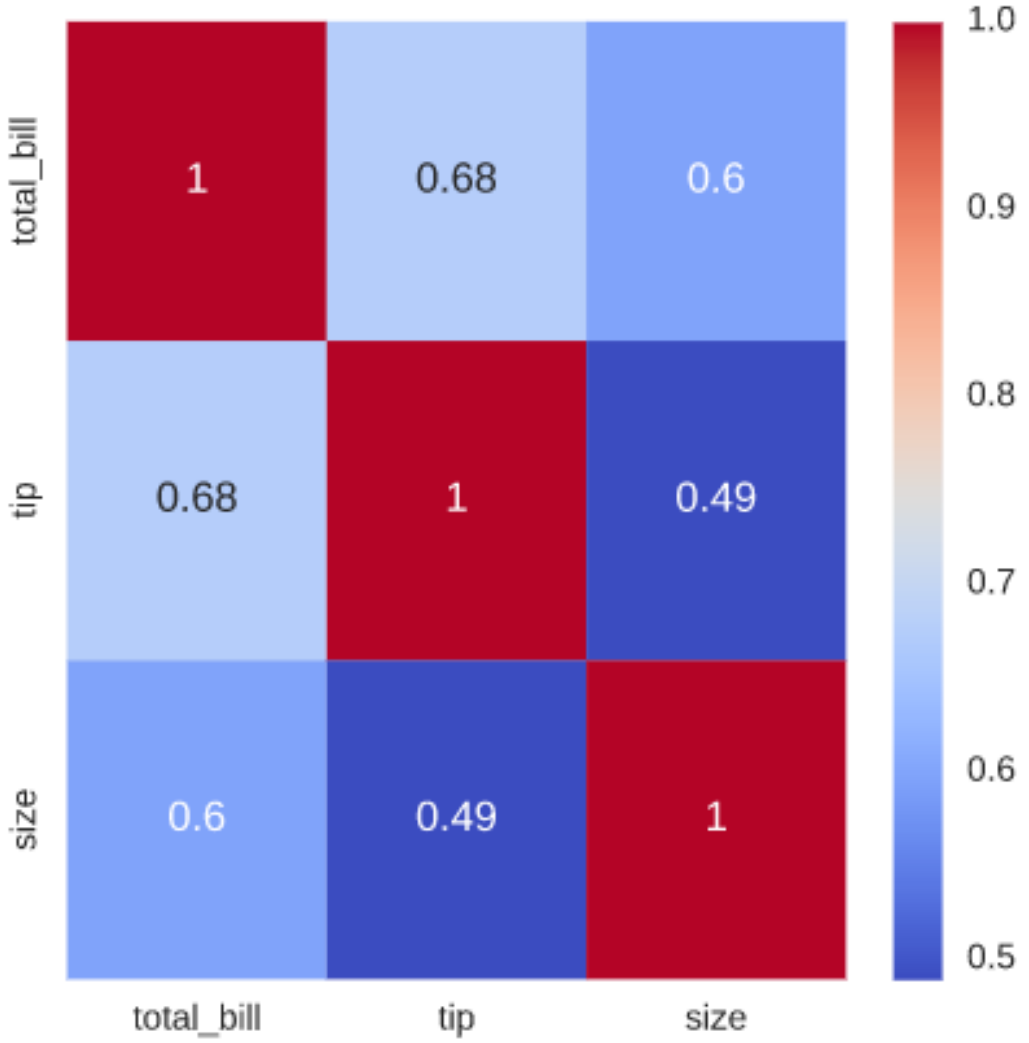
- A **heatmap** is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colors

- Heat Maps are graphical representations of data that utilize color-coded systems.

- The primary purpose of Heat Maps is to better visualize the volume of locations/events within a dataset and assist in directing viewers towards areas on data visualizations that matter most.

9- Heat Plot

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
tips.corr()
sns.heatmap(tips.corr(), cmap='coolwarm
plt.show()
```



```
import seaborn as sns
import matplotlib.pyplot as plt
flights = sns.load_dataset('flights')
pvflights = flights.pivot_table(values='passengers', index='month',
        columns='year')
sns.heatmap(pvflights,
        linecolor='white', linewidths=1)
plt.show()
```