Data Visualization with Python

July 29, 2023

[1]:

```
import pandas as pd
import matplotlib.pyplot as plt

# reading the database
data = pd.read_csv("data/tips.csv")

# printing the top 10 rows
display(data.head(5))
```

```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

0.1 Scatter Plot

Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them.

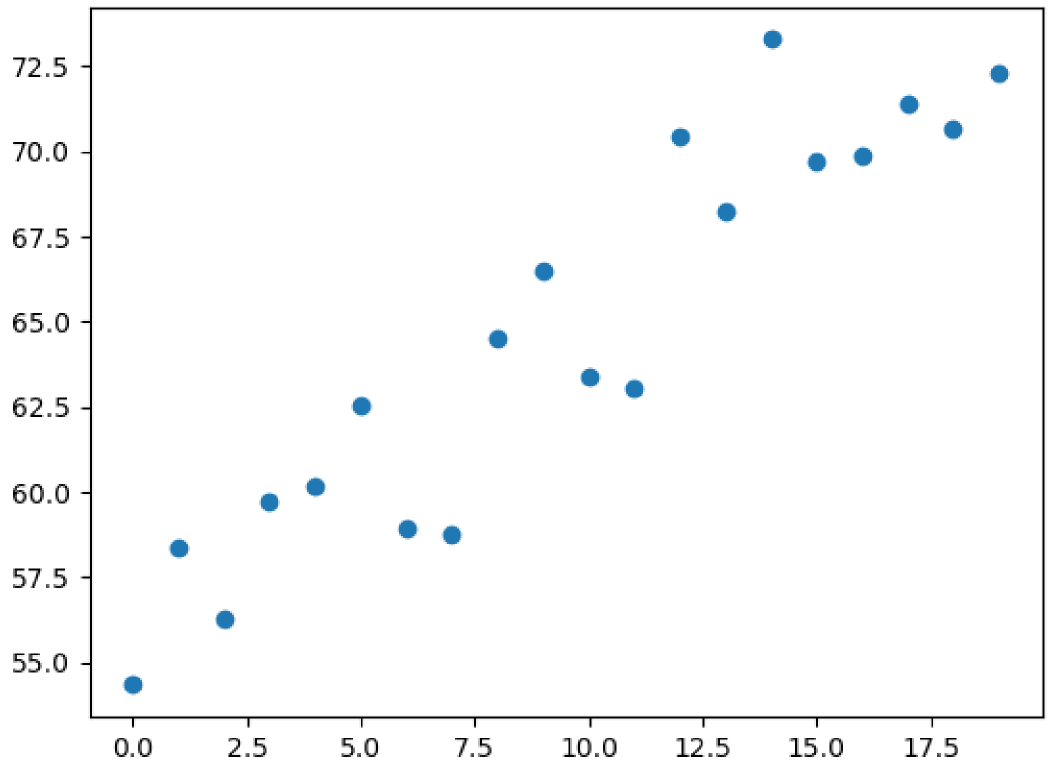
Purpose: Displaying relationships between variables.

matplotlib function: `scatter(x, y)`

- x, y: The values for the two variables.

[2]:

```
import numpy as np

# value gen
x = range(20)
y = np.arange(50, 70) + (np.random.random(20) * 10)

# plot
# adding figure
plt.figure()

plt.scatter(x, y)
plt.show()
```

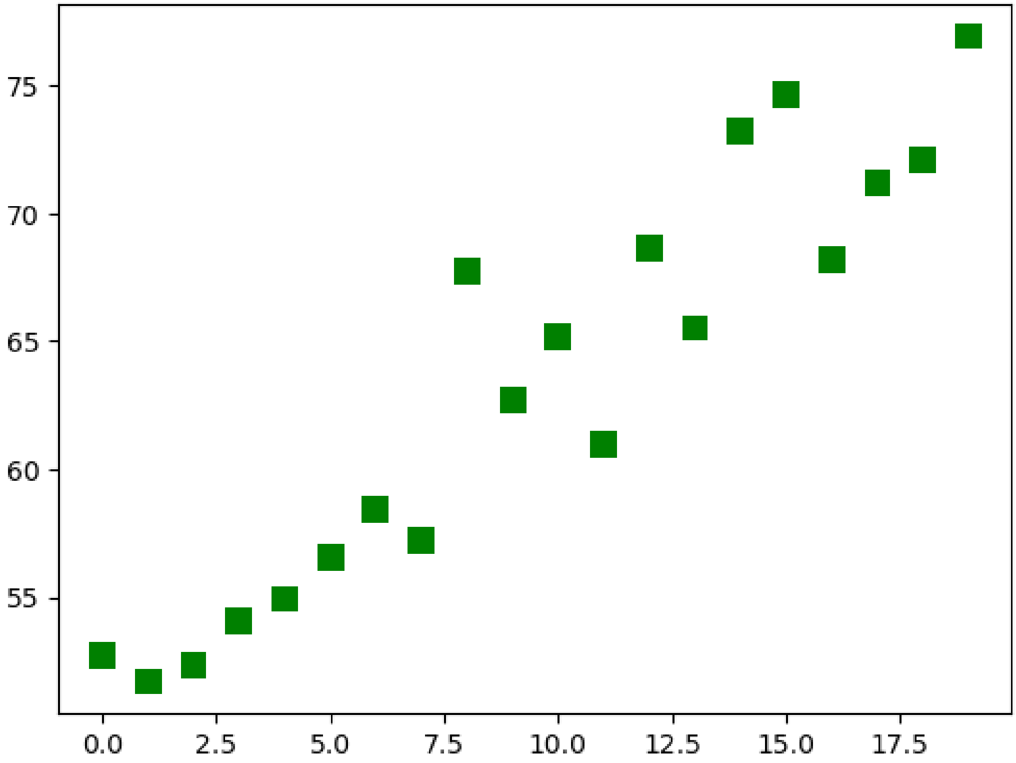


Commonly used parameters:

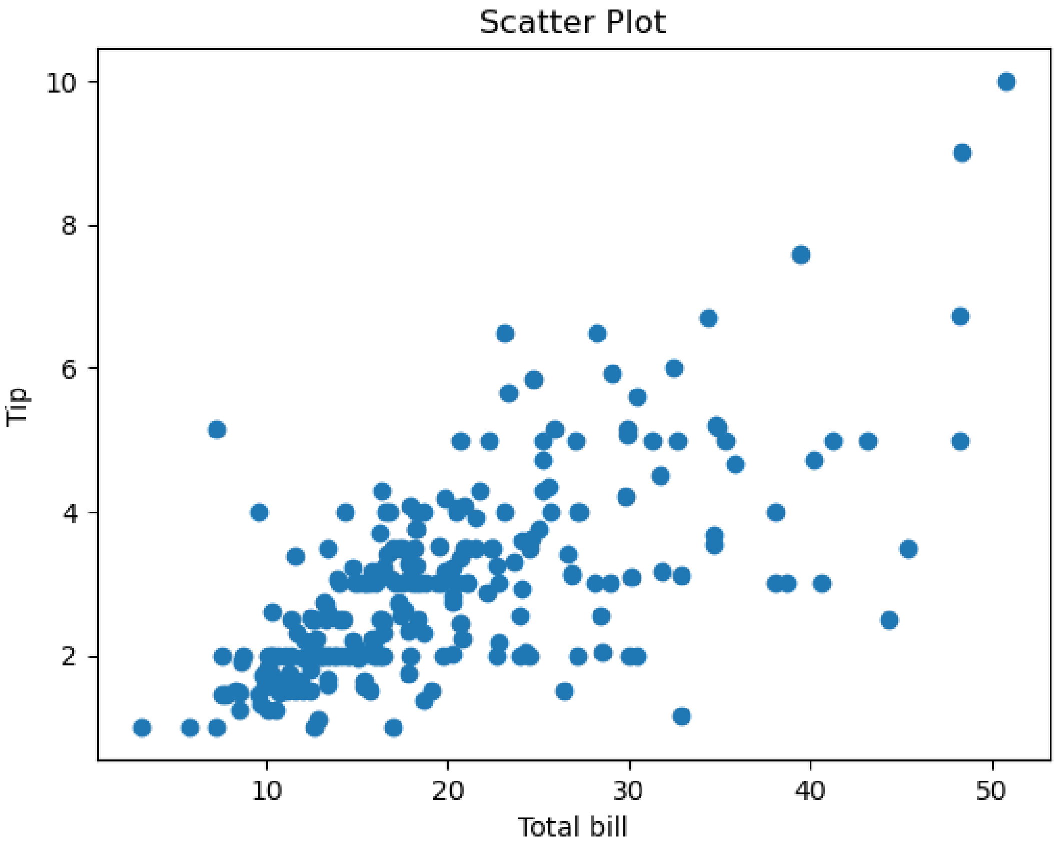
- `c`: Set the color of the markers.
- `s`: Set the size of the markers.
- `marker`: Set the marker style, e.g., circles, triangles, or squares.
- `edgecolor`: Set the color of the lines on the edges of the markers.

[3]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(20)
y = np.arange(50, 70) + (np.random.random(20) * 10.)

plt.figure()

plt.scatter(x,
            y,
            c='green',
            s=100,
            marker='s',
            edgecolor='none')
```

```
plt.show()
```



[4]:
```
import matplotlib.pyplot as plt

plt.figure()
# Scatter plot with day against tip
plt.scatter(data['total_bill'], data['tip'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
plt.xlabel('Total bill')
plt.ylabel('Tip')

plt.show()
```

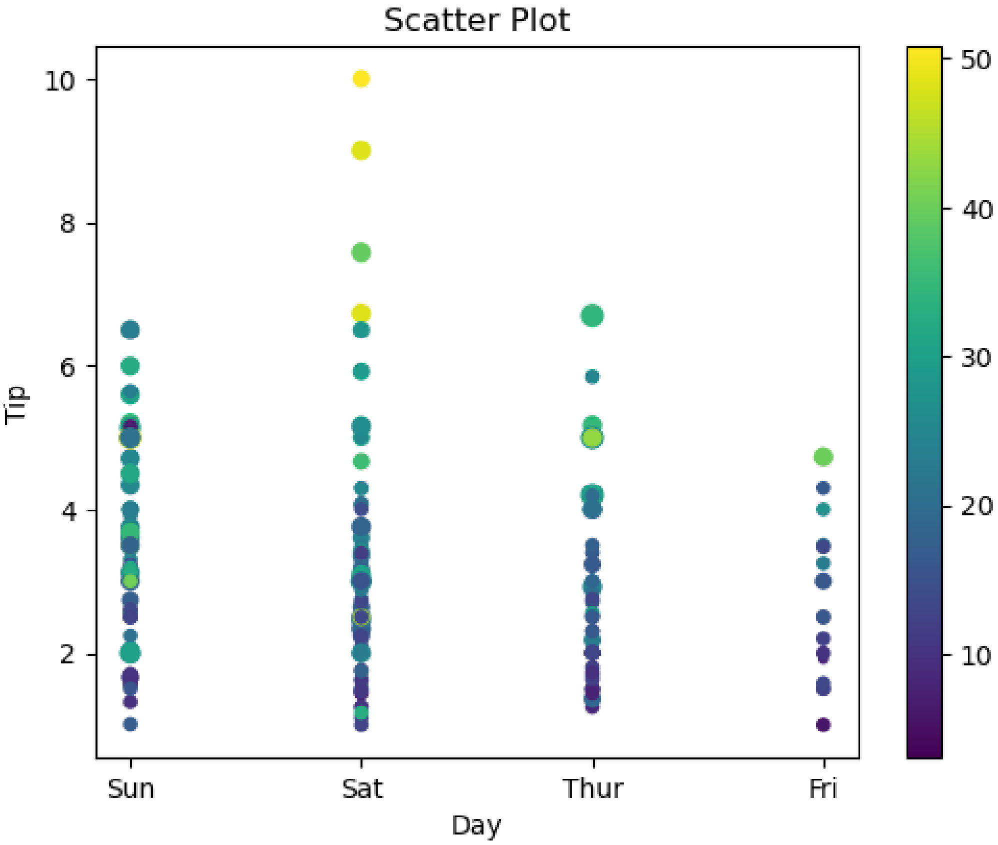


This graph can be more meaningful if we can add colors and also change the size of the points. We can do this by using the c and s parameter respectively of the scatter function. We can also show the color bar using the colorbar() method.

[5]:

```
import pandas as pd
import matplotlib.pyplot as plt


# # reading the database
# data = pd.read_csv("data/tips.csv")

# Scatter plot with day against tip
plt.scatter(data['day'],
            data['tip'],
            s=data['size']*10,
            c=data['total_bill'])

# Adding Title to the Plot
plt.title("Scatter Plot")
```

```
# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

plt.colorbar()

plt.show()
```



1 ## Line Chart

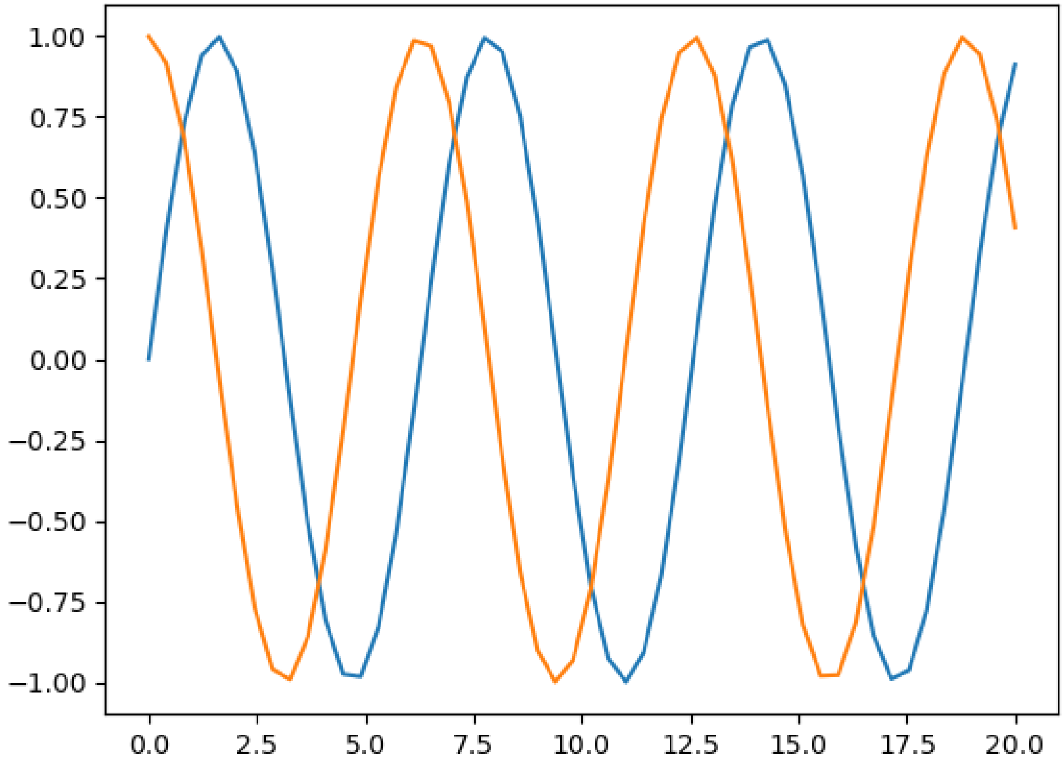
Purpose: Showing trends in data – usually time series data with many time points.

matplotlib function: `plot(x, y)`

- x: The x-coordinates of the lines or markers.
- y: The y-coordinates of the lines or markers.

[6]:
```
import numpy as np
# data gen
x = np.linspace(0, 20)
```

```
y1 = np.sin(x)
y2 = np.cos(x)

# plot
plt.figure()
# line chart
plt.plot(x, y1)
plt.plot(x, y2)
plt.show()
```

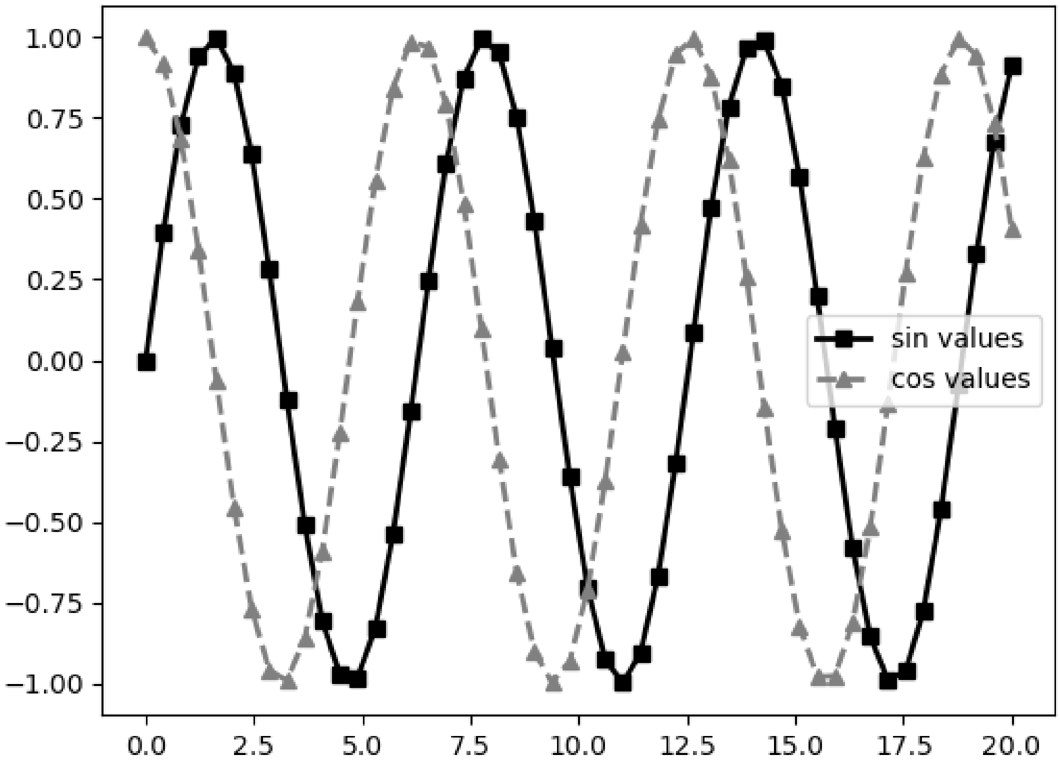


Commonly used parameters:

- `color`: Set the color of the line.
- `linestyle`: Set the line style, e.g., solid, dashed, or none.
- `linewidth`: Set the line thickness.
- `marker`: Set the marker style, e.g., circles, triangles, or none.
- `markersize`: Set the marker size.
- `label`: Set the label for the line that will show up in the legend.

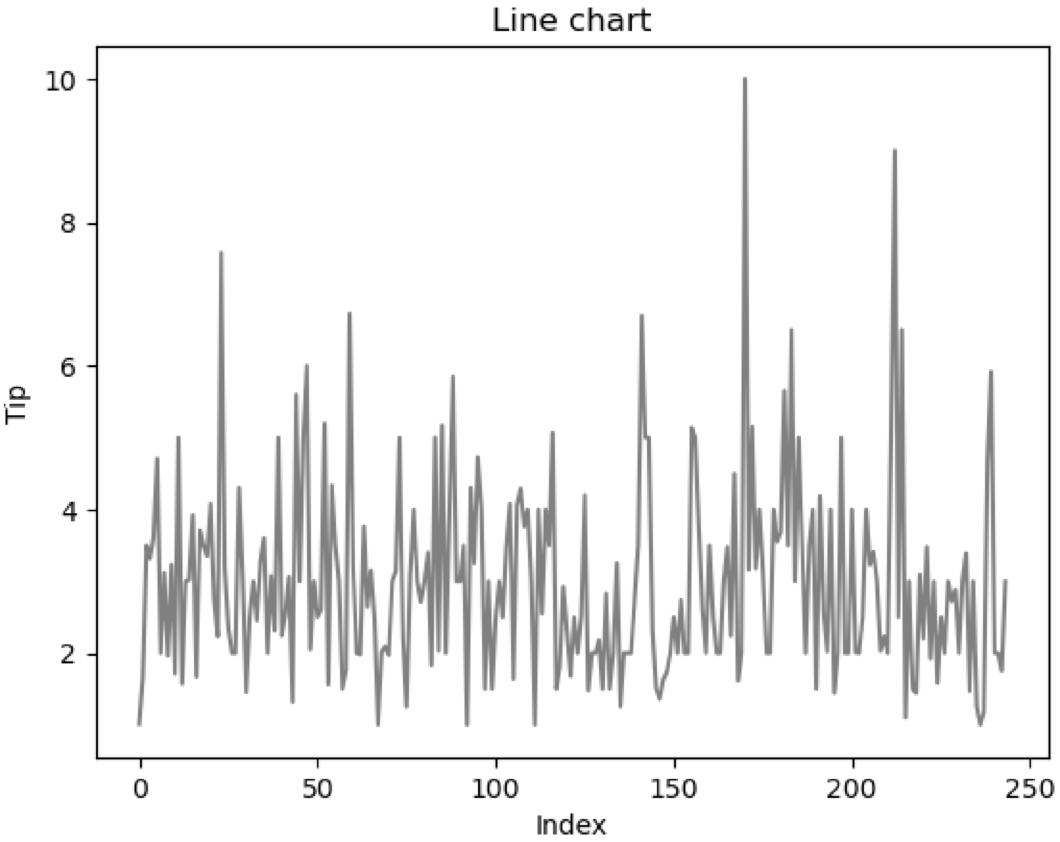
[7]:

```
# data gen.
x = np.linspace(0, 20)
y1 = np.sin(x)
y2 = np.cos(x)

# plot
plt.figure()

plt.plot(x, y1,
         color='black',
         linestyle='-',
         linewidth=2,
         marker='s',
         markersize=6,
         label='sin values')

plt.plot(x, y2,
         color='gray',
         linestyle='--',
         linewidth=2,
         marker='^',
         markersize=6,
         label='cos values')

plt.legend()
plt.show()
```



[8]:

```
import pandas as pd
import matplotlib.pyplot as plt

# # reading the database
# data = pd.read_csv("data/tips.csv")
plt.figure()
# Line plot with day against tip
plt.plot(data['tip'], color='gray')

# Adding Title to the Plot
plt.title("Line chart")

# Setting the X and Y labels
plt.xlabel('Index')
plt.ylabel('Tip')

plt.show()
```



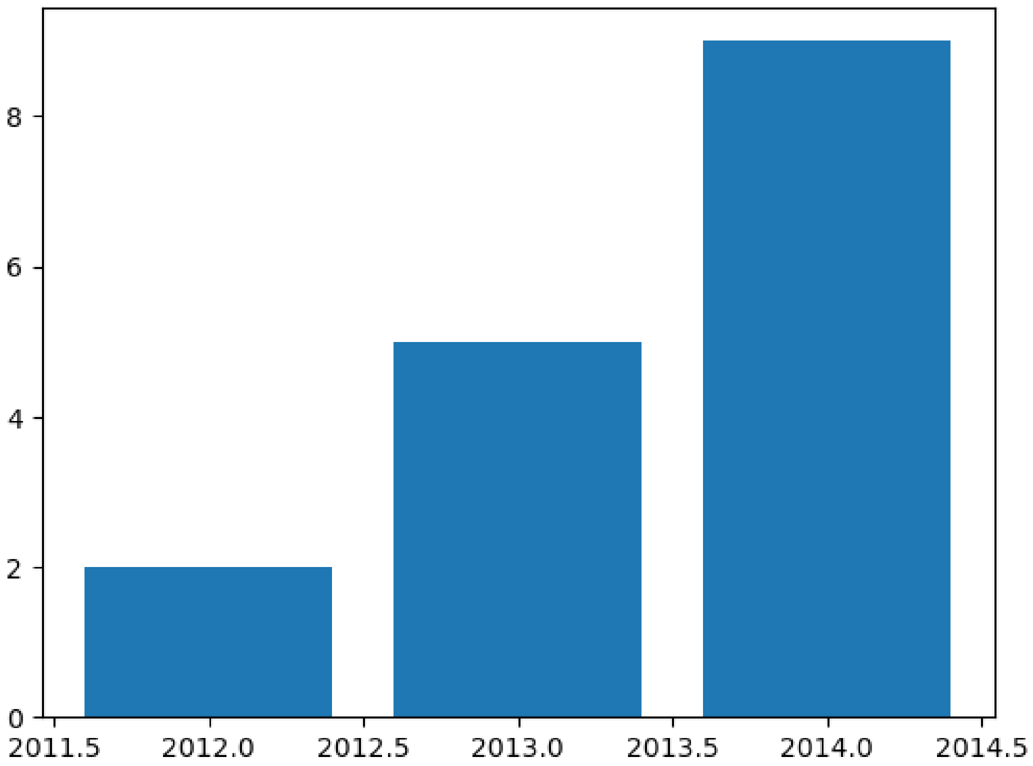
1.1 Bar Chart

Purpose: Comparing categories **OR** showing temporal trends in data with few (< 4) time points.

matplotlib function: `bar(left, height)`

- `left`: The x coordinate(s) of the left sides of the bars.
- `height`: The height(s) of the bars.

[9]:
```
years = np.arange(2012, 2015)
values = [2, 5, 9]

plt.figure()

plt.bar(years, values)

plt.show()
```



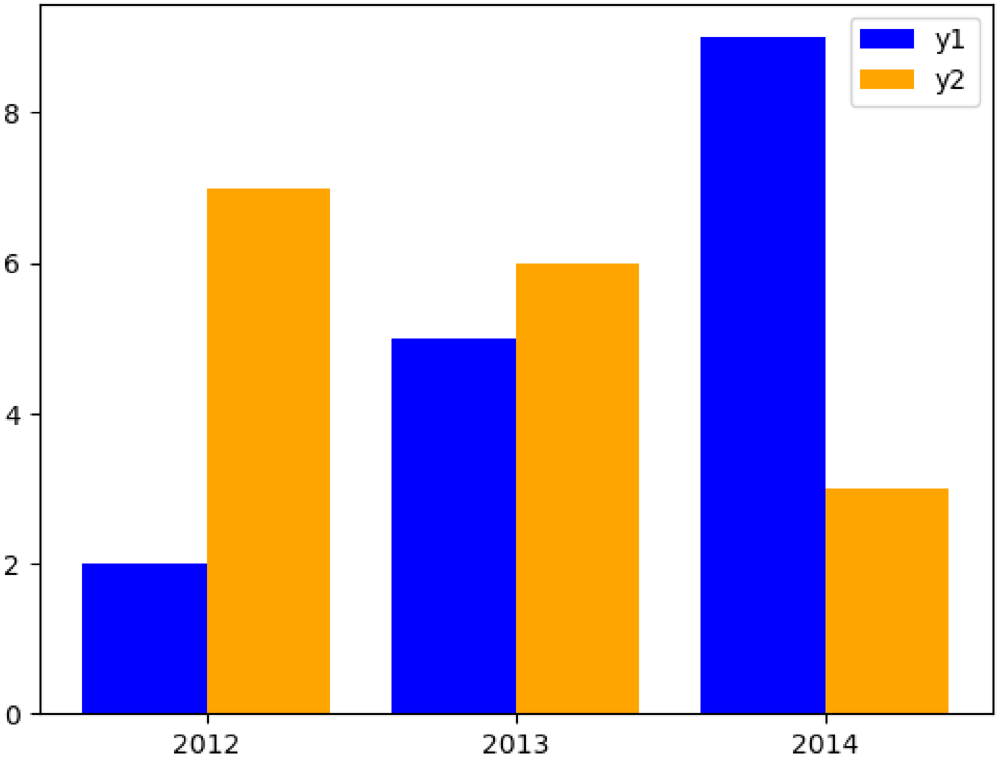
Commonly used parameters:

- `color`: Set the color of the bars.
- `edgecolor`: Set the color of the lines on the edges of the bars.
- `width`: Set the width of the bars.
- `align`: Set the alignment of the bars, e.g., center them on the x coordinate(s).
- `label`: Set the label for the bar that will show up in the legend.

[10]:

```
years = np.arange(2012, 2015)
category1_values = [2, 5, 9]
category2_values = [7, 6, 3]

plt.figure()

plt.bar(years - 0.2,
        category1_values,
        color='blue',
        edgecolor='none',
        width=0.4,
        align='center',
        label='y1')
```

```
plt.bar(years + 0.2,
        category2_values,
        color='orange',
        edgecolor='none',
        width=0.4,
        align='center',
        label='y2')

plt.xticks(years, [str(year) for year in years])

plt.legend()
plt.show()
```

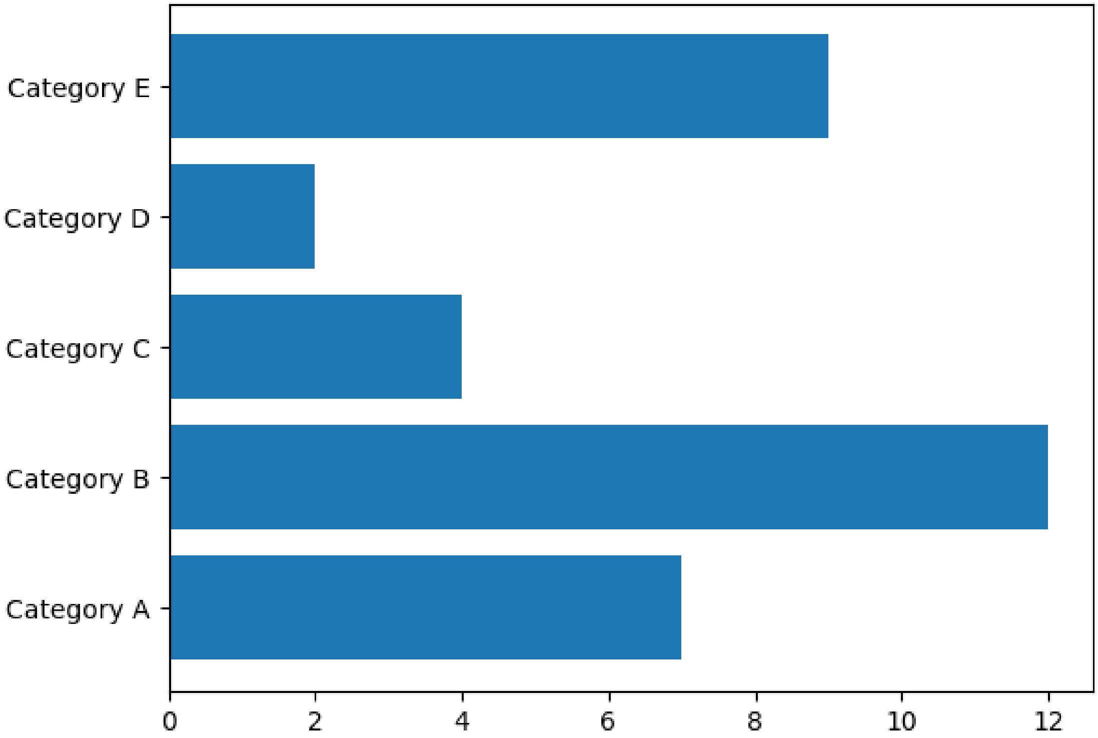


Purpose: Comparing categories.

matplotlib function: `barh(bottom, width)`

- `bottom`: The y coordinate(s) of the bars.
- `width`: The width(s) of the bars.

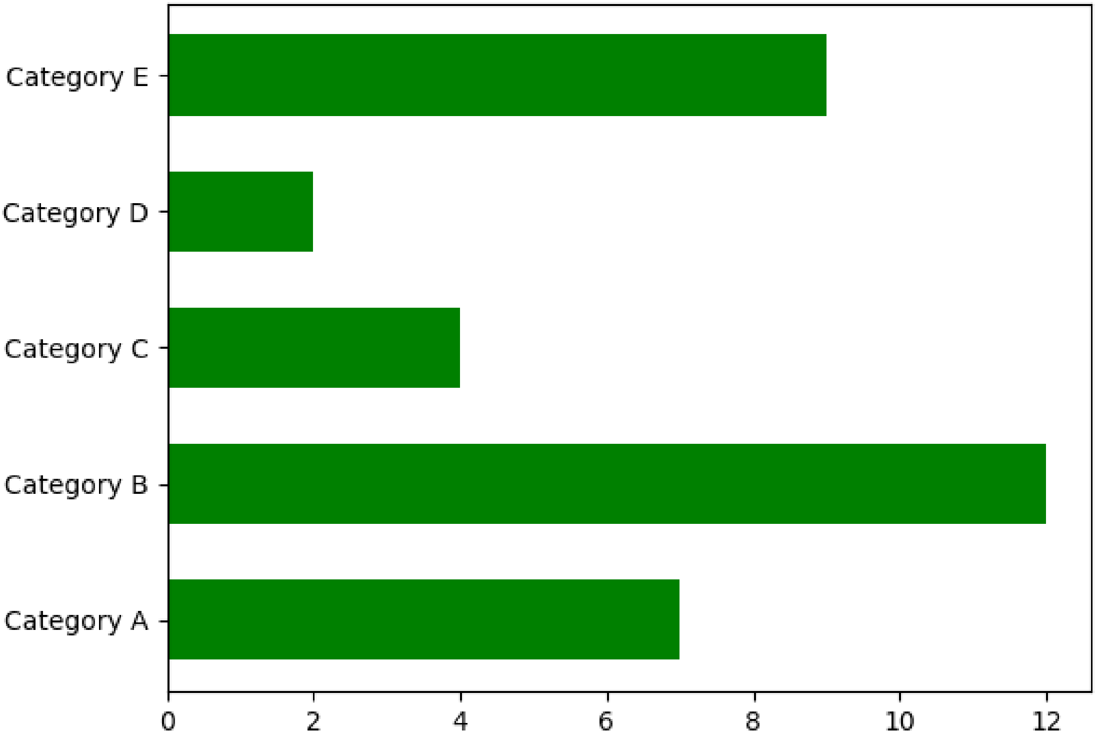
[11]:
```
categories = ['A', 'B', 'C', 'D', 'E']
values = [7, 12, 4, 2, 9]

# print(np.arange(len(categories)))
plt.figure()
plt.barh(np.arange(len(categories)), values)

plt.yticks(np.arange(len(categories)),
           [f'Category {x}' for x in categories])
plt.show()
```



Commonly used parameters:

- `color`: Set the color of the bars.
- `edgecolor`: Set the color of the lines on the edges of the bars.
- `height`: Set the height of the bars.
- `align`: Set the alignment of the bars, e.g., center them on the y coordinate(s).

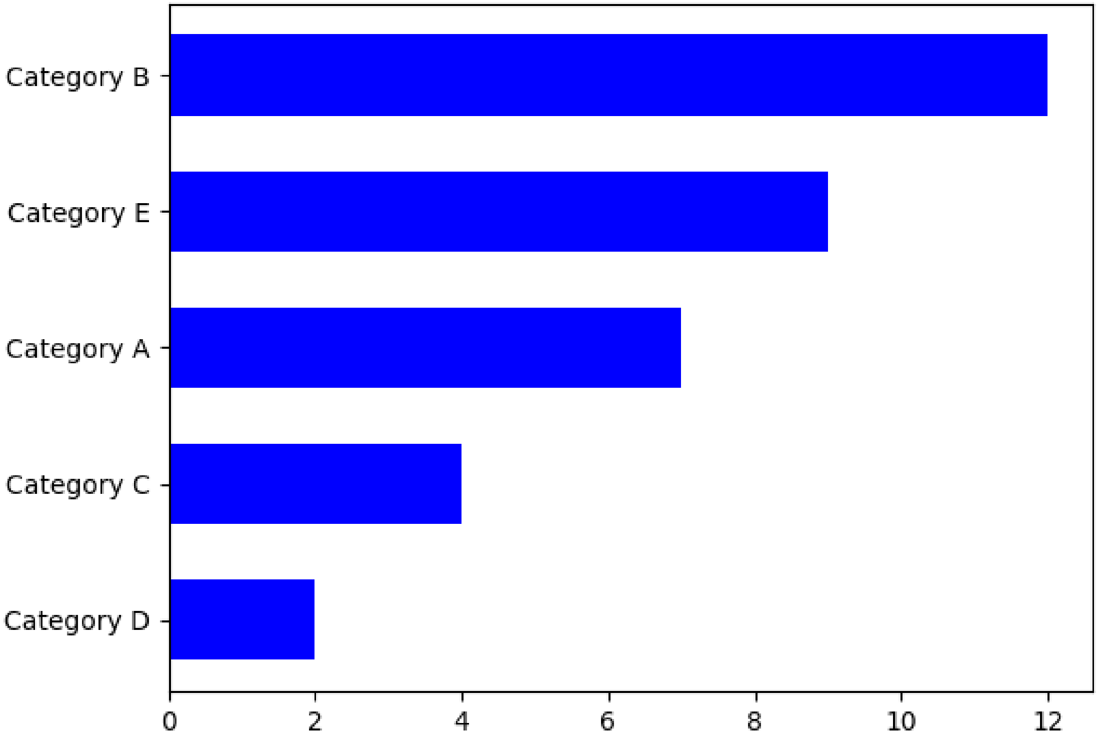
[12]:
```
categories = ['A', 'B', 'C', 'D', 'E']
values = [7, 12, 4, 2, 9]

plt.figure()
plt.barh(np.arange(len(categories)),
         values,
         color='green',
         edgecolor='none',
         height=0.6,
         align='center')

plt.yticks(np.arange(len(categories)),
           ['Category {}'.format(x) for x in categories])
plt.show()
```



[13]:

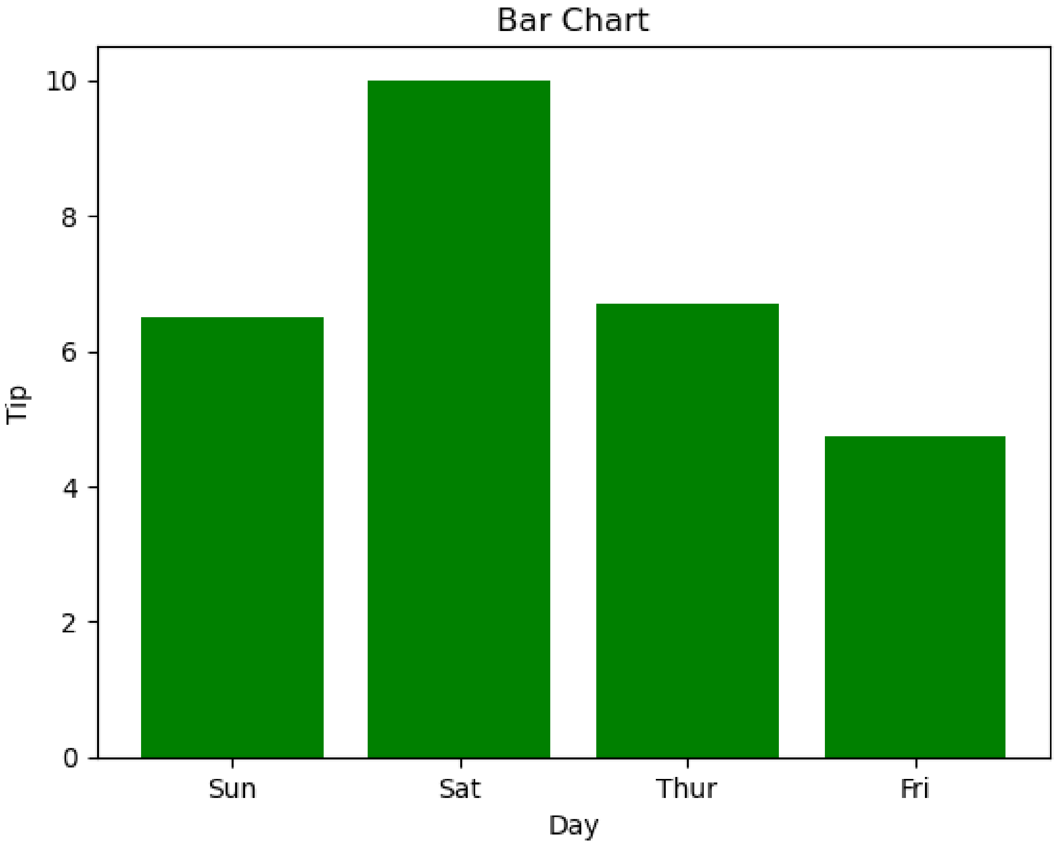
```
import pandas as pd

categories = ['A', 'B', 'C', 'D', 'E']
values = [7, 12, 4, 2, 9]

# convert to dataframe
category_values = pd.DataFrame({'categories': categories,
                                'values': values})

# category values wise sorted
```

```
category_values.sort_values(by='values', ascending=True, inplace=True)

categories = category_values['categories'].values
values = category_values['values'].values

# plot
plt.figure()

plt.barh(np.arange(len(categories)),
         values,
         color='blue',
         edgecolor='none',
         height=0.6,
         align='center')

plt.yticks(np.arange(len(categories)),
           ['Category {}'.format(x) for x in categories])
plt.show()
```



[14]:

```
# Bar chart with day against tip
plt.bar(data['day'], data['tip'], color='green')
plt.title("Bar Chart")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

plt.show()
```



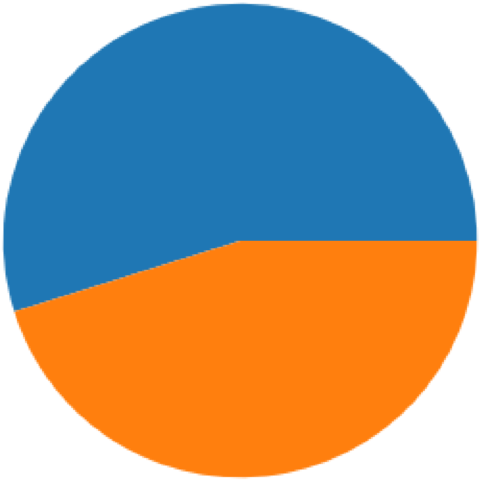
2 Pie charts

Purpose: Displaying a simple proportion.

matplotlib function: `pie(sizes)`

- `sizes`: The size of the wedges as either a fraction or number.

[15]:
```
counts = [17, 14]

plt.figure(figsize=(4, 4))
plt.pie(counts)
plt.show()
```

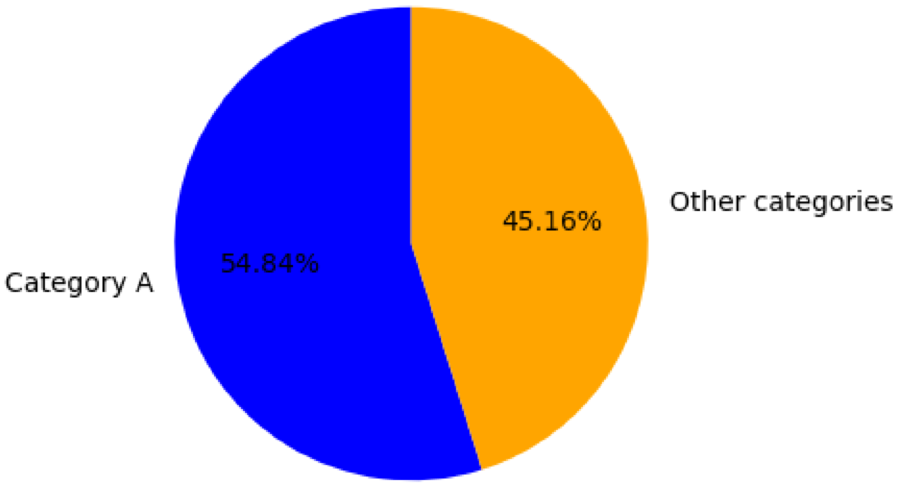


Commonly used parameters:

- `colors`: Set the colors of the wedges.
- `labels`: Set the labels of the wedges.
- `startangle`: Set the angle that the wedges start at.
- `autopct`: Set the percentage display format of the wedges.

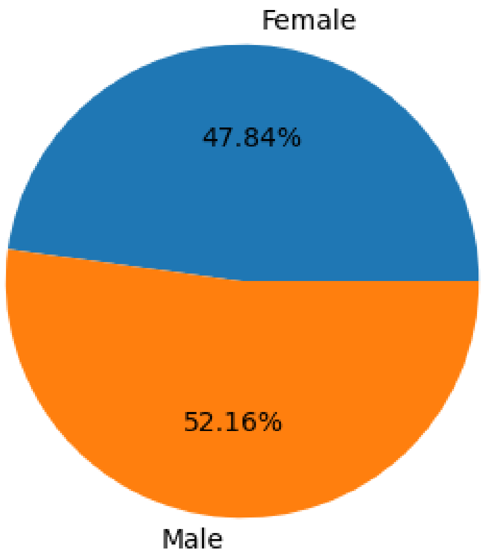
[16]:

```
counts = [17, 14]

plt.figure(figsize=(4, 4))

plt.pie(counts,
        colors=['blue', 'orange'],
        labels=['Category A', 'Other categories'],
        startangle=90,
        autopct='%1.2f%%')
plt.show()
```



```
[17]: grp_data = data.groupby('sex').tip.mean().reset_index()
      print(grp_data)
      plt.figure(figsize=(4, 4))

      plt.pie(grp_data.tip, labels=grp_data.sex, autopct='%1.2f%%')
      plt.show()
```

```
      sex       tip
0  Female  2.833448
1    Male  3.089618
```



2.1 Histogram

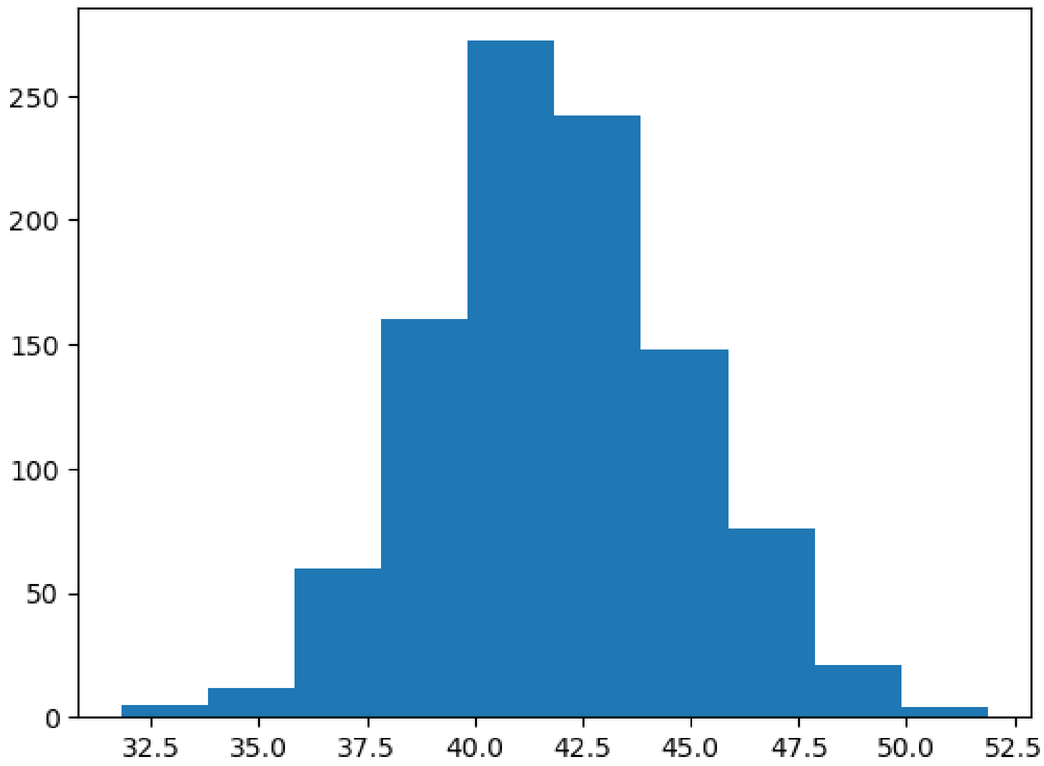
Purpose: Showing the spread of a data column.

matplotlib function: `hist(x)`

- x: List of values to display the distribution of.

[18]:

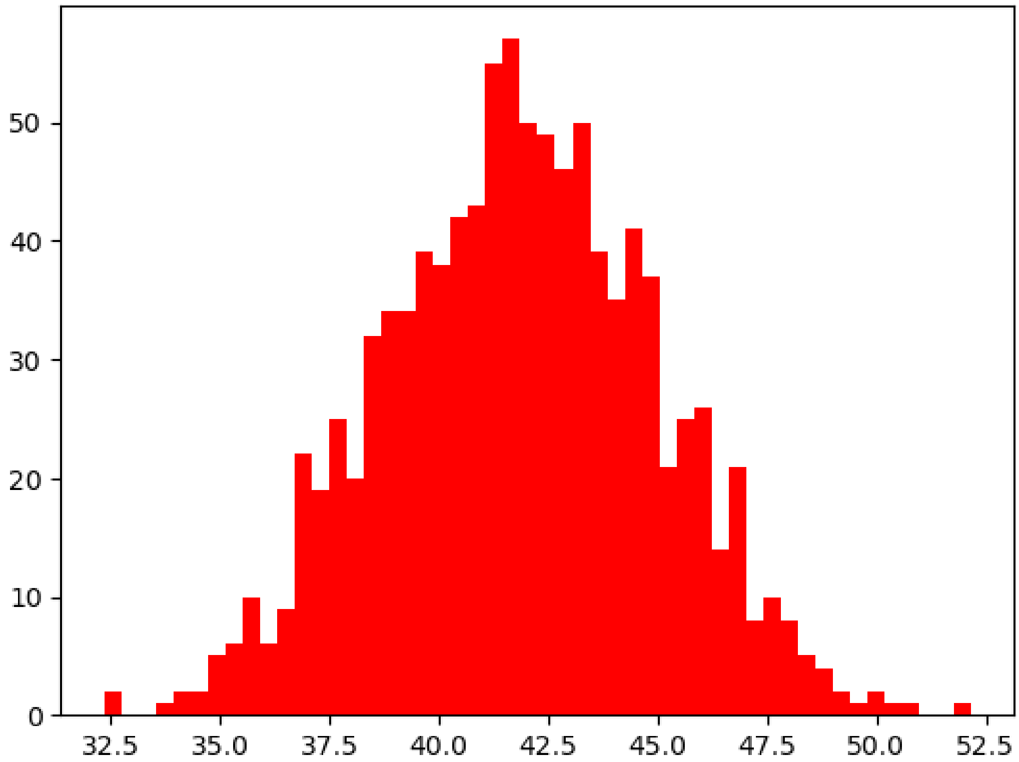
```
column_data = np.random.normal(42, 3, 1000)

plt.figure()

plt.hist(column_data)
plt.show()
```



Commonly used parameters:

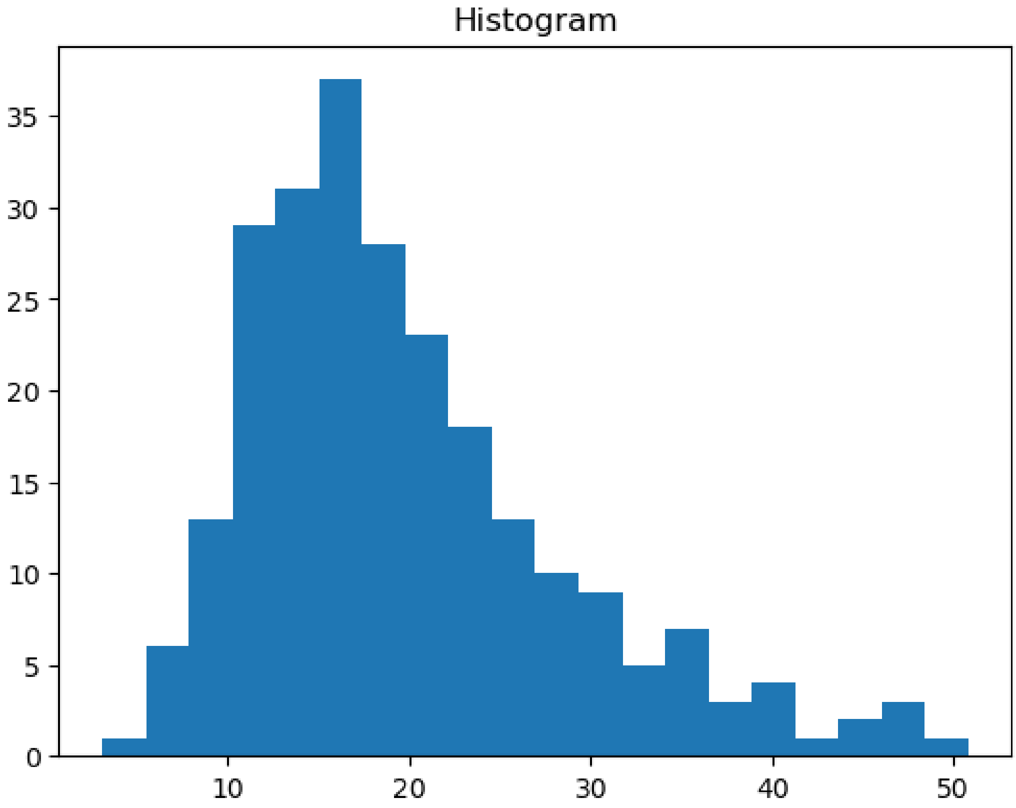
- `color`: Set the color of the bars in the histogram.
- `bins`: Set the number of bins to display in the histogram, or specify specific bins.

[19]:
```
column_data = np.random.normal(42, 3, 1000)

plt.figure()

plt.hist(column_data,
         color='red',
         bins=50)
plt.show()
```



[20]:

```
import pandas as pd
import matplotlib.pyplot as plt


# # reading the database
# data = pd.read_csv("tips.csv")

# hostogram of total_bills
plt.hist(data['total_bill'], bins=20)

plt.title("Histogram")

# Adding the legends
plt.show()
```



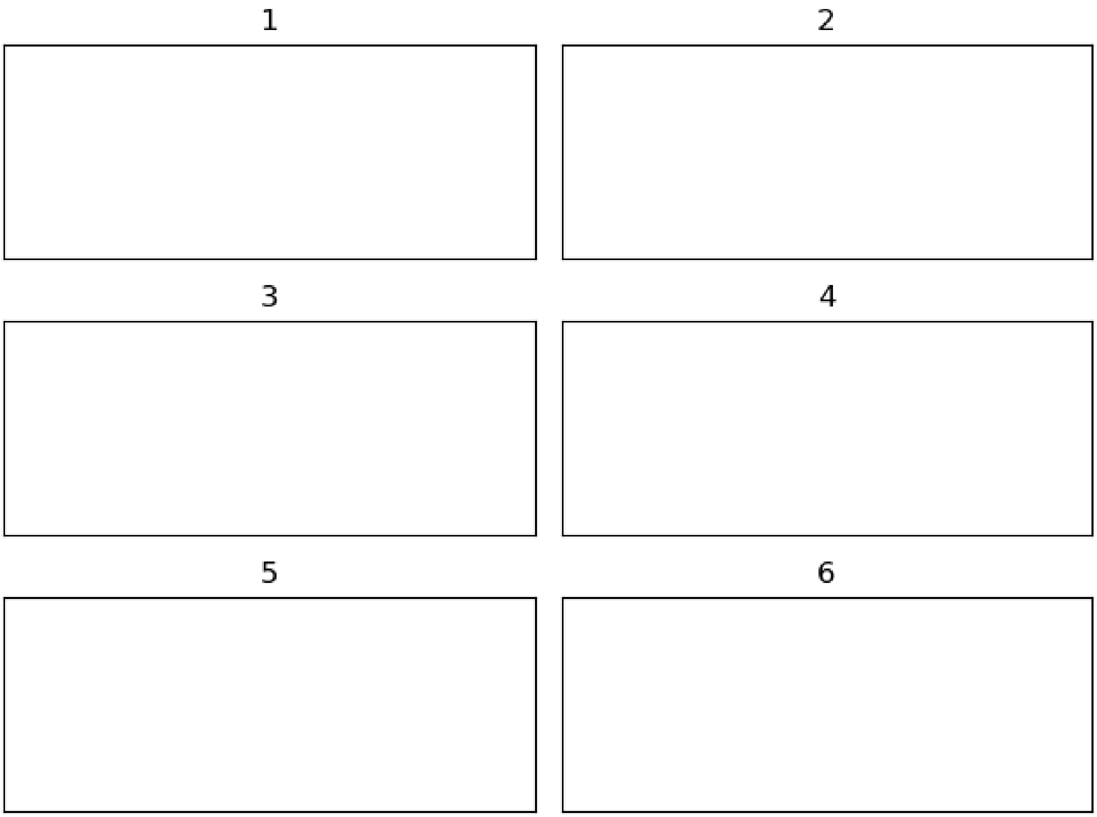
3 Subplots

Purpose: Allows you to place multiple charts in a figure.

matplotlib function: `subplot(nrows, ncols, plot_number)`

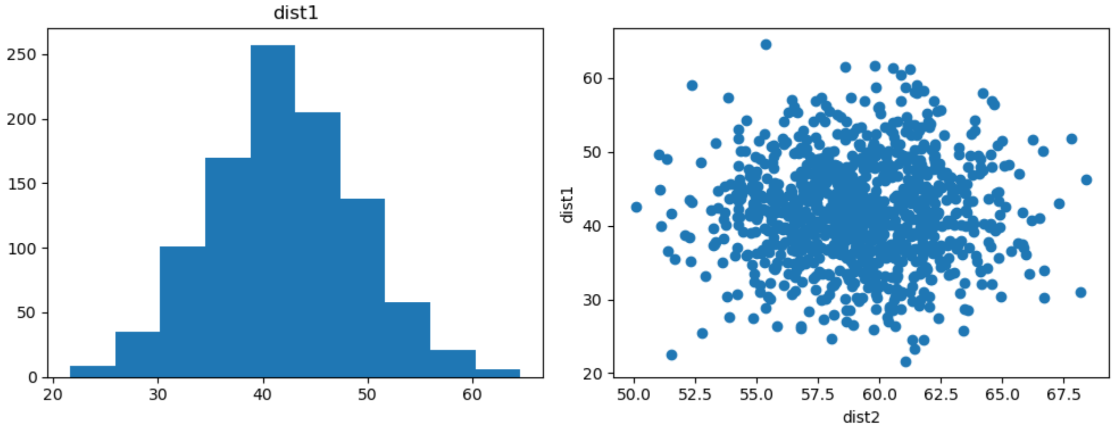
- `nrows`: The number of rows in the figure.
- `ncols`: The number of columns in the figure.
- `plot_number`: The placement of the chart (starts at 1).

[21]:
```
plt.figure()

for i in range(1, 7):
    plt.subplot(3, 2, i)
    plt.title(i)
    plt.xticks([])
    plt.yticks([])

plt.tight_layout()
```



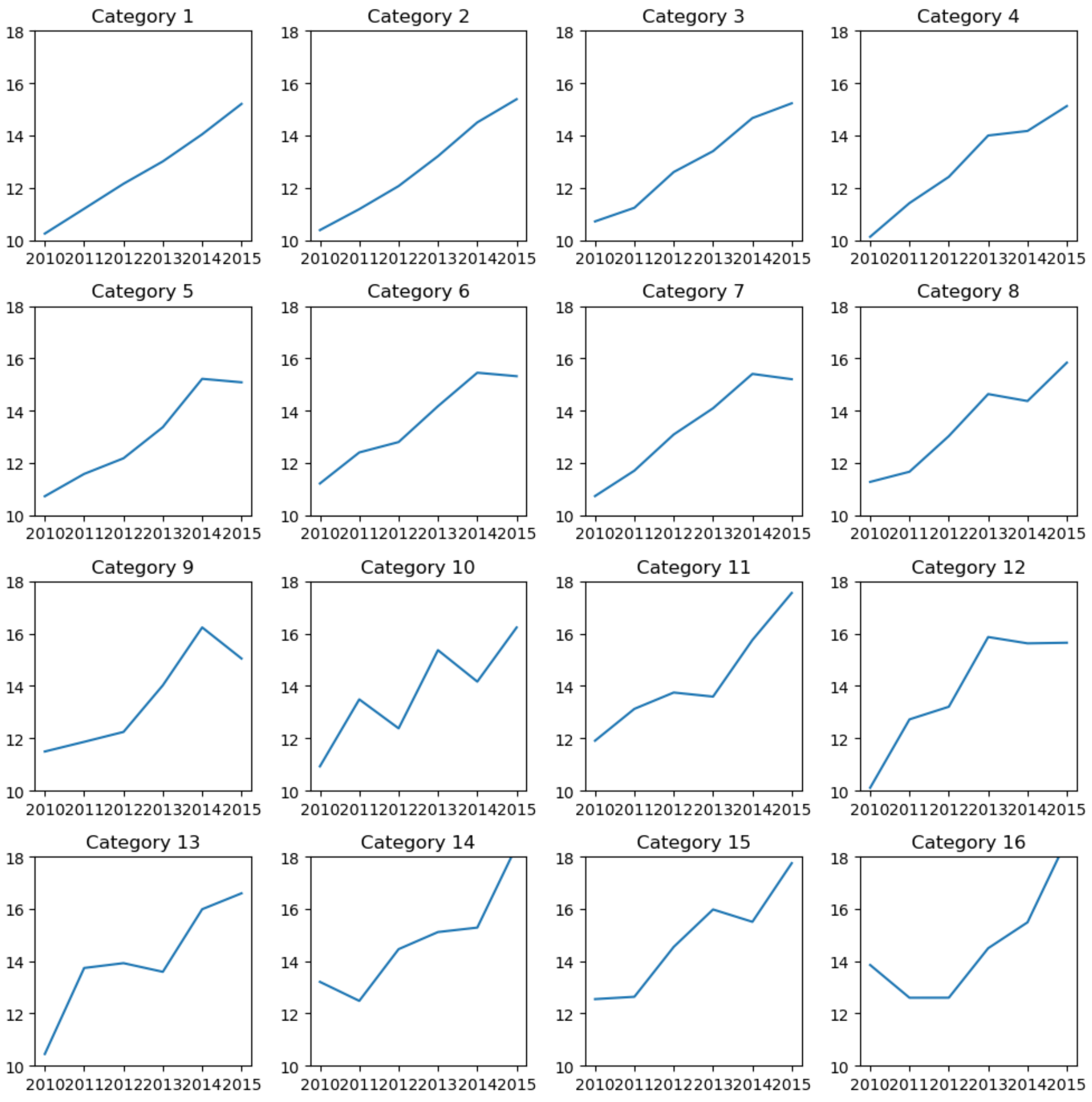
[22]:

```
dist1 = np.random.normal(42, 7, 1000)
dist2 = np.random.normal(59, 3, 1000)

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.hist(dist1)
plt.title('dist1')

plt.subplot(1, 2, 2)
plt.scatter(dist2, dist1)
plt.xlabel('dist2')
plt.ylabel('dist1')

plt.tight_layout()
```



[23]:

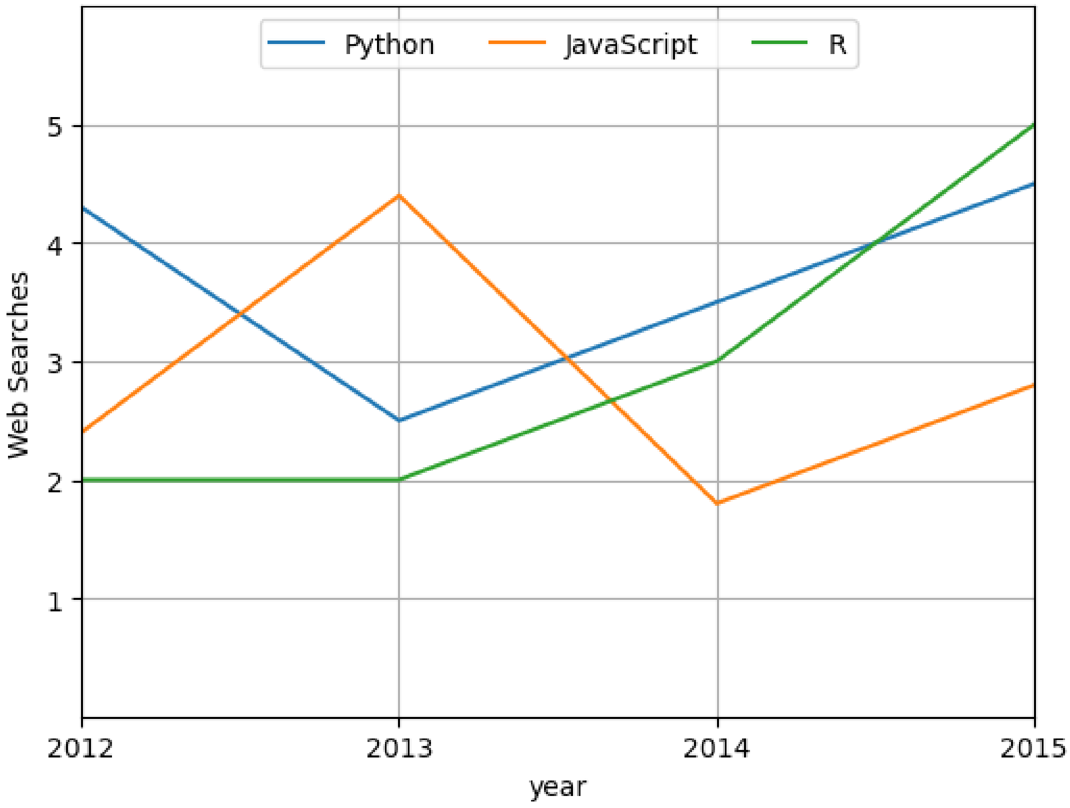
```
years = np.arange(2010, 2016)

plt.figure(figsize=(10, 10))

for category_num in range(1, 17):
    plt.subplot(4, 4, category_num)
    y_vals = np.arange(10, 16) + (np.random.random(6) * category_num / 4.)
    plt.plot(years, y_vals)
    plt.ylim(10, 18)
    plt.xticks(years, [str(year) for year in years])
    plt.title('Category {}'.format(category_num))

plt.tight_layout()
```



3.1 Styles

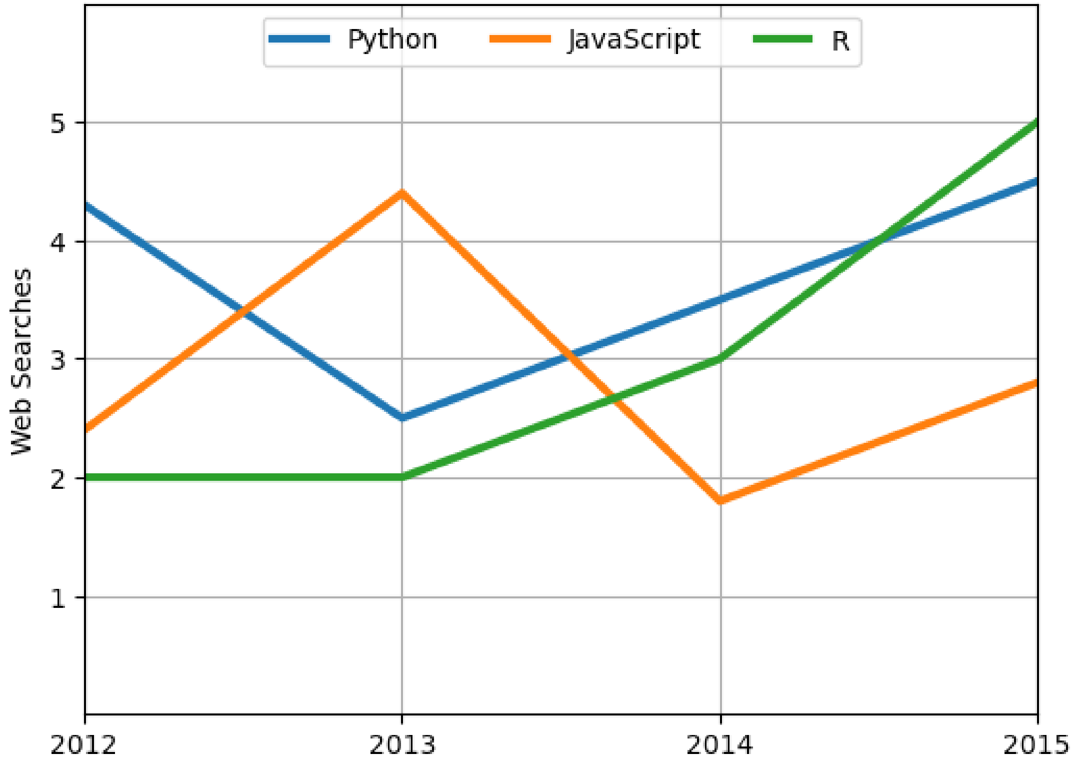
[24]:
```
import matplotlib.pyplot as plt
# data
x1_values = [2012, 2013, 2014, 2015]
y1_values = [4.3, 2.5, 3.5, 4.5]

x2_values = [2012, 2013, 2014, 2015]
y2_values = [2.4, 4.4, 1.8, 2.8]

x3_values = [2012, 2013, 2014, 2015]
y3_values = [2, 2, 3, 5]
```

```
# plot
plt.figure()
plt.plot(x1_values, y1_values, label='Python')
plt.plot(x2_values, y2_values, label='JavaScript')
plt.plot(x3_values, y3_values, label='R')

plt.xlim(2012, 2015)
plt.ylim(0, 6)
plt.xticks([2012, 2013, 2014, 2015], ['2012', '2013', '2014', '2015'])
plt.yticks([1, 2, 3, 4, 5])

plt.xlabel('year')
plt.ylabel('Web Searches')

plt.legend(loc='upper center', ncol=3)
plt.grid(True)

plt.savefig('web-searches.png', dpi=150)
```



[25]:

```
plt.figure()
plt.plot(x1_values, y1_values, label='Python', lw=3, color='#1f77b4')
plt.plot(x2_values, y2_values, label='JavaScript', lw=3, color='#ff7f0e')
plt.plot(x3_values, y3_values, label='R', lw=3, color='#2ca02c')

plt.xlim(2012, 2015)
plt.ylim(0, 6)
plt.xticks([2012, 2013, 2014, 2015], ['2012', '2013', '2014', '2015'])
plt.yticks([1, 2, 3, 4, 5])

plt.xlabel('')
plt.ylabel('Web Searches')

plt.legend(loc='upper center', ncol=3)
# plt.legend(loc='lower center', ncol=3)
plt.grid(True)

plt.savefig('web-searches.png', dpi=150)
```



4 Advanced Visualization

[26]:

```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

[27]:

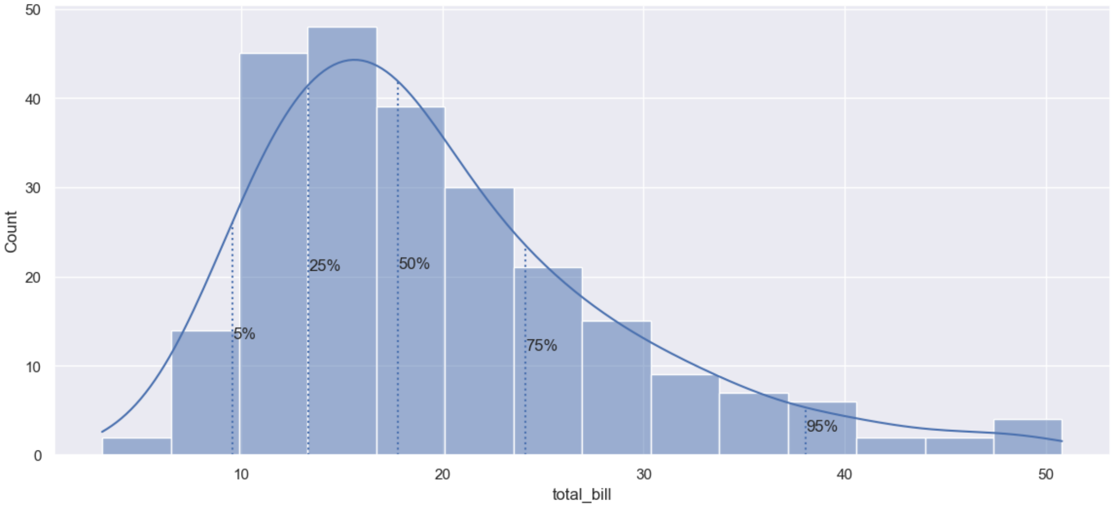
```
df = pd.read_csv('data/tips.csv')
df
```

[27]:

```
     total_bill   tip     sex smoker   day    time  size
0         16.99  1.01  Female     No   Sun  Dinner     2
1         10.34  1.66    Male     No   Sun  Dinner     3
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   Sun  Dinner     2
4         24.59  3.61  Female     No   Sun  Dinner     4
..          …     …       …      …     …       …     …
239       29.03  5.92    Male     No   Sat  Dinner     3
240       27.18  2.00  Female    Yes   Sat  Dinner     2
241       22.67  2.00    Male    Yes   Sat  Dinner     2
242       17.82  1.75    Male     No   Sat  Dinner     2
243       18.78  3.00  Female     No  Thur  Dinner     2

[244 rows x 7 columns]
```

[28]:

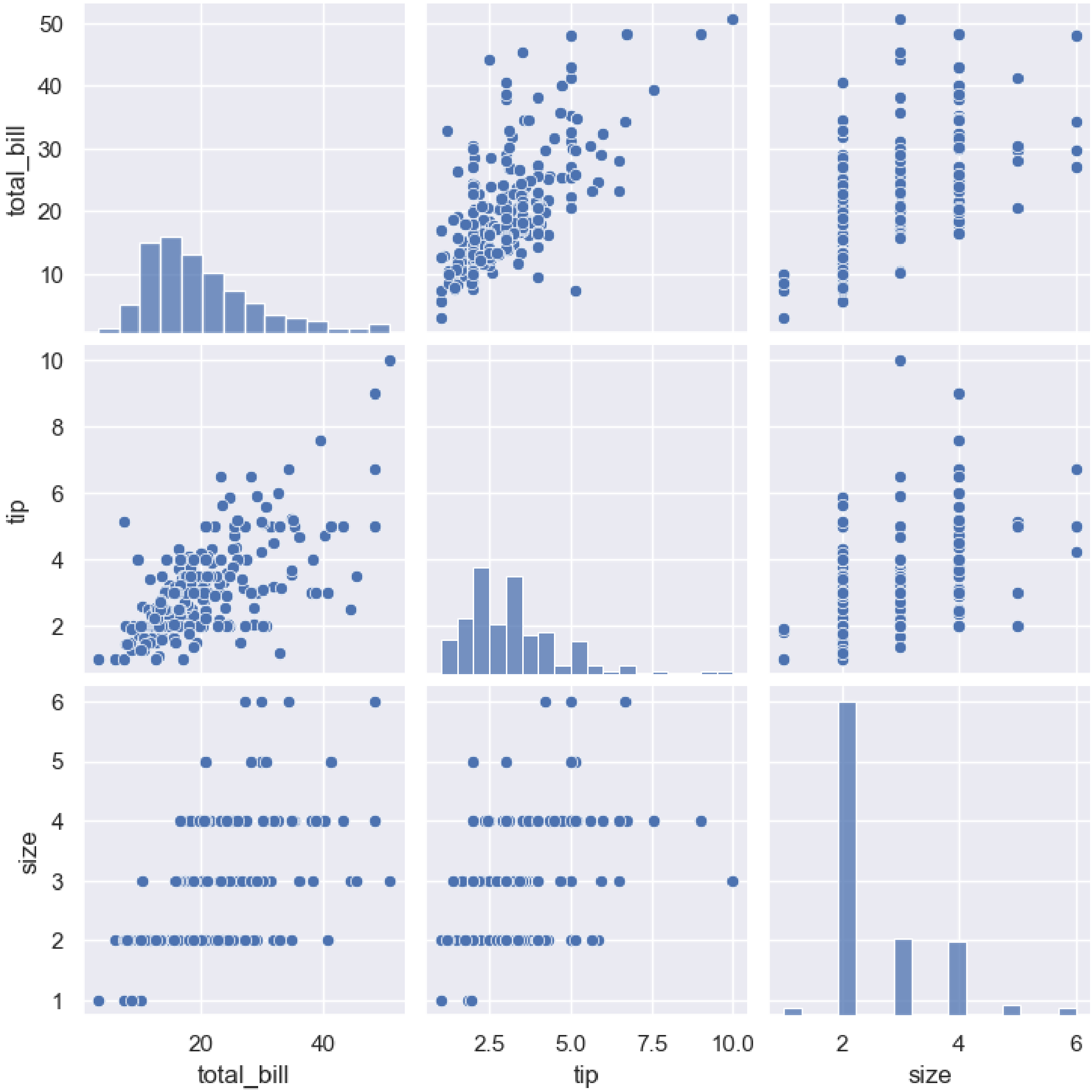
```
import numpy as np

sns.set(rc = {'figure.figsize':(14, 6)})
ax = sns.histplot(x = data['total_bill'], kde=True)

quant_5 = data['total_bill'].quantile(0.05)
quant_25 = data['total_bill'].quantile(0.25)
quant_50 = data['total_bill'].quantile(0.5)
quant_75 = data['total_bill'].quantile(0.75)
quant_95 = data['total_bill'].quantile(0.95)
quant_dict = {'5%': quant_5, '25%': quant_25, '50%': quant_50, '75%': quant_75,␣
 ↪'95%': quant_95}

kdeline = ax.lines[0]
xs = kdeline.get_xdata()
ys = kdeline.get_ydata()

for key, value in quant_dict.items():
    height = np.interp(value, xs, ys)
    ax.vlines(value, 0, height,  ls=':')
    ax.text(value , height * 0.5, key, rotation=0)

plt.show()
```



4.0.1 Pairplot

[29]:

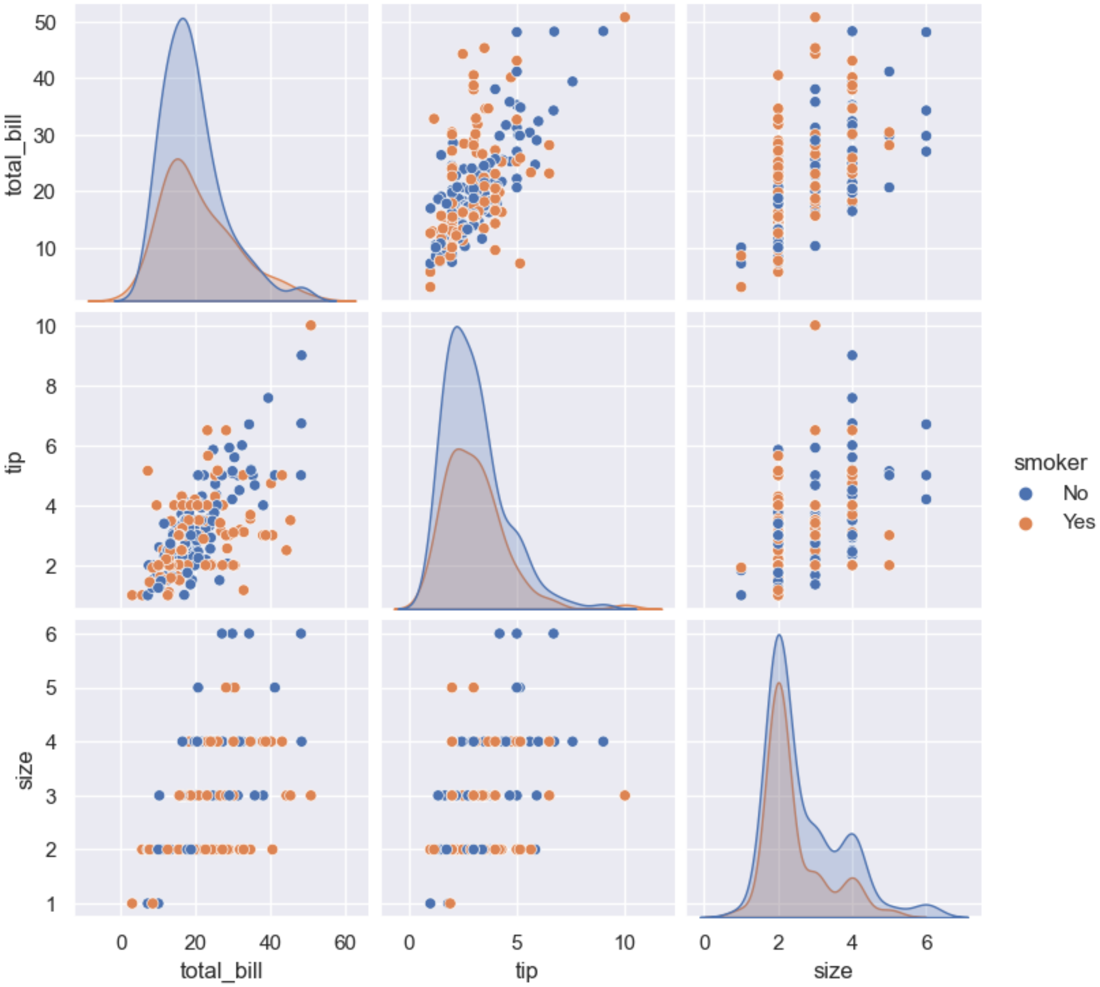
```
sns.pairplot(df)
plt.show()
```



[30]:
```
df.value_counts('smoker')
```

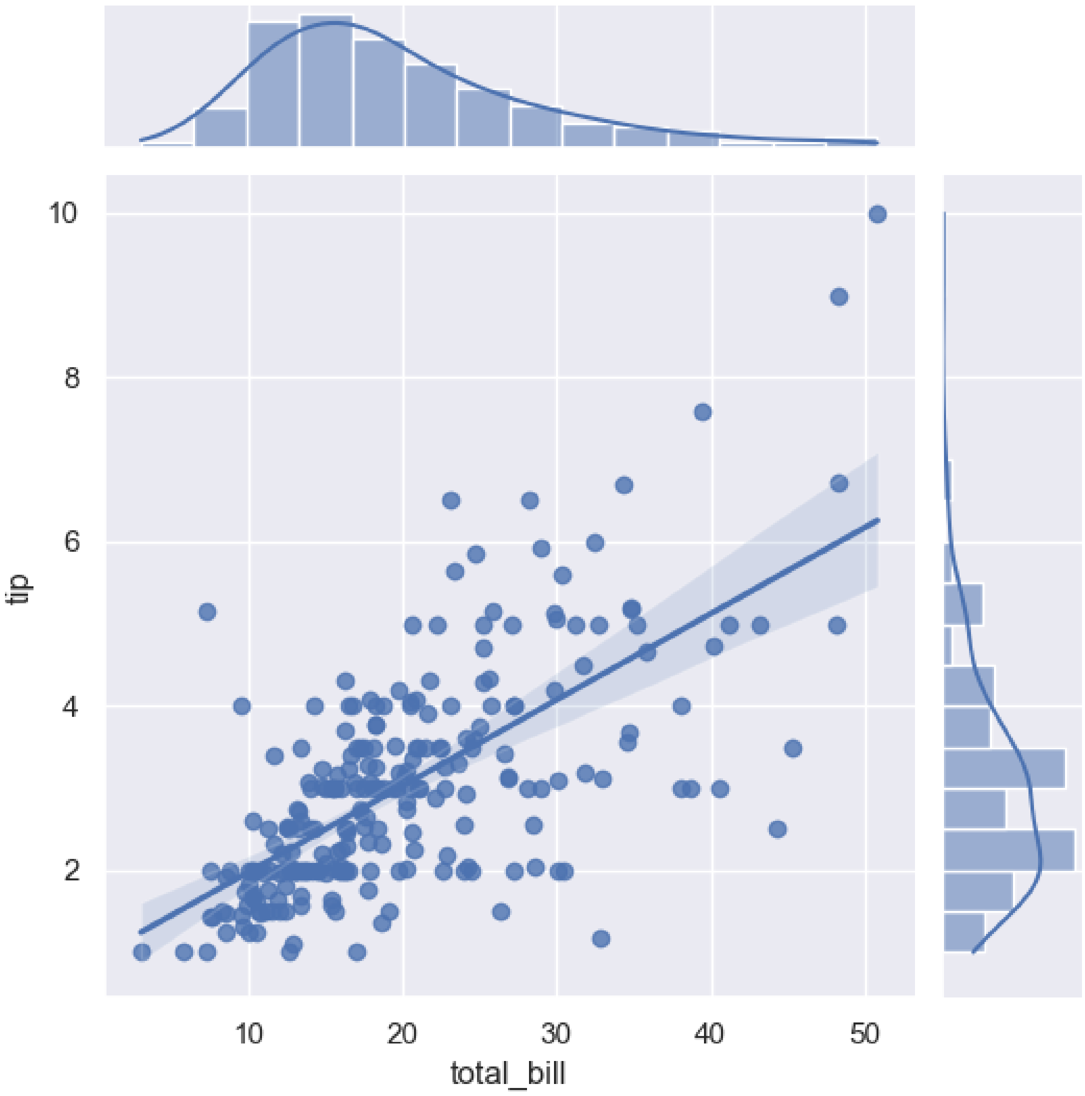
[30]:
```
smoker
No     151
Yes     93
dtype: int64
```

[31]:
```
sns.pairplot(df, hue='smoker')
plt.show()
```



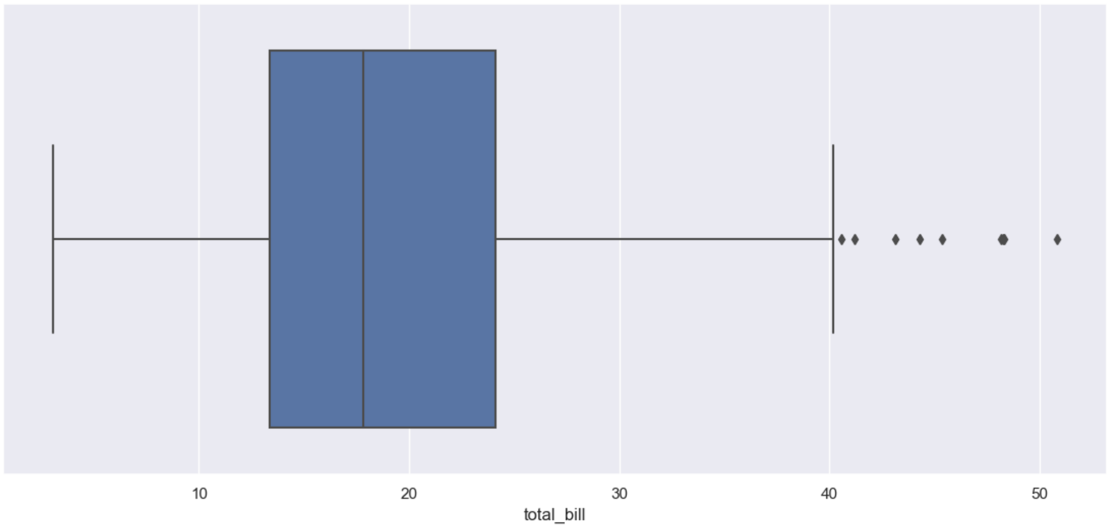
4.0.2 Linear regression with distributions

[32]:
```
sns.jointplot(x="total_bill", y="tip", data=df, kind="reg")
plt.show()
```

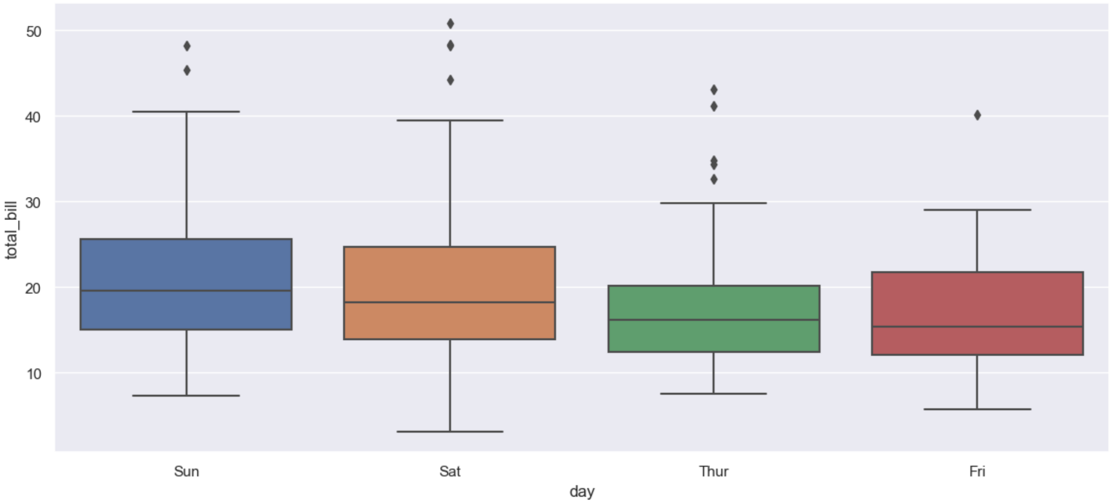


4.0.3 Box Plot

[33]:
```
sns.boxplot(x=df["total_bill"])
plt.show()
```



```
[34]: sns.boxplot(x="day", y="total_bill", data=df)
      plt.show()
```



4.0.4 Heatmap

```
[35]: # Load the example flights dataset and convert to long-form
      flights_long = sns.load_dataset("flights")
      flights_long
```

```
[35]:      year month  passengers
      0    1949   Jan         112
```

```
1    1949   Feb         118
2    1949   Mar         132
3    1949   Apr         129
4    1949   May         121
..    …    …          …
139  1960   Aug         606
140  1960   Sep         508
141  1960   Oct         461
142  1960   Nov         390
143  1960   Dec         432

[144 rows x 3 columns]
```

[36]:

```
flights = flights_long.pivot("month", "year", "passengers")
flights
```

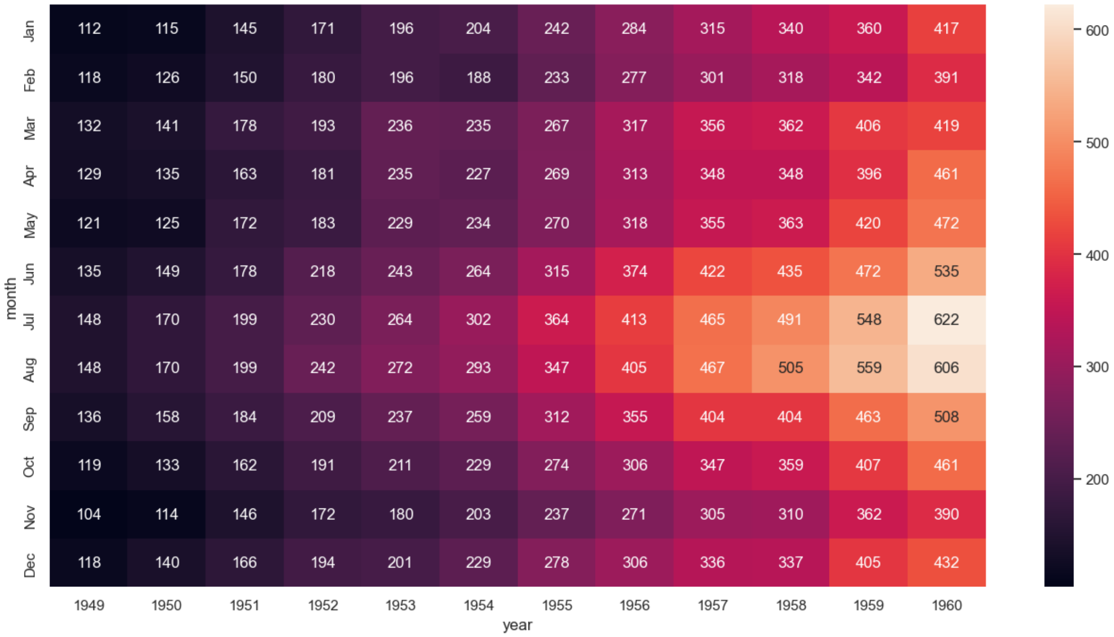
```
C:\Users\88019\AppData\Local\Temp\ipykernel_7132\254108779.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.pivot will be keyword-
only.
  flights = flights_long.pivot("month", "year", "passengers")
```

[36]:

```
year   1949  1950  1951  1952  1953  1954  1955  1956  1957  1958  1959  1960
month
Jan     112   115   145   171   196   204   242   284   315   340   360   417
Feb     118   126   150   180   196   188   233   277   301   318   342   391
Mar     132   141   178   193   236   235   267   317   356   362   406   419
Apr     129   135   163   181   235   227   269   313   348   348   396   461
May     121   125   172   183   229   234   270   318   355   363   420   472
Jun     135   149   178   218   243   264   315   374   422   435   472   535
Jul     148   170   199   230   264   302   364   413   465   491   548   622
Aug     148   170   199   242   272   293   347   405   467   505   559   606
Sep     136   158   184   209   237   259   312   355   404   404   463   508
Oct     119   133   162   191   211   229   274   306   347   359   407   461
Nov     104   114   146   172   180   203   237   271   305   310   362   390
Dec     118   140   166   194   201   229   278   306   336   337   405   432
```

[37]:

```
# Draw a heatmap with the numeric values in each cell
f, ax = plt.subplots(figsize=(16, 8))
sns.heatmap(flights, annot=True, fmt=".0f")
plt.show()
```



Visit this page to learn more -> https://seaborn.pydata.org/examples/index.html