Introduction to Data Science

Lecture 5.1 Matplotlib and Seaborn

Instructor: Sangryul Jeon

School of Computer Science and Engineering

srjeonn@pusan.ac.kr

Readings

❖ **Fundamentals of Data Visualization**

- Link: https://clauswilke.com/dataviz/
  - contains the complete author manuscript before final copy-editing and other quality control

❖ **Matplotlib: Visualization with Python**

- **https://matplotlib.org/**

❖ **seaborn: statistical data visualization**

- **https://seaborn.pydata.org/**

Python libraries

pandas

matplotlib

seaborn

NumPy

- ❖ Fundamental package for scientific computing
- ❖ Exceptionally fast – written in C
- ❖ Main data structure:
  - ndarray : n-dimensional arrays of homogeneous data types
- ❖ Data manipulation ≈ NumPy array manipulation
- ❖ Used in other libraries - Matplotlib, pandas, scikit- learn

Pandas

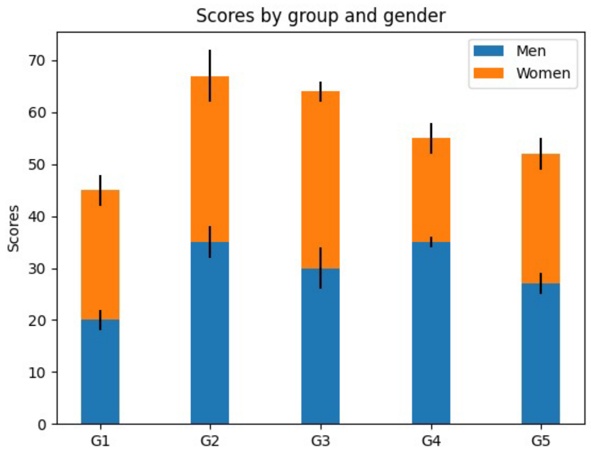
- Fundamental tool for handling and analyzing input data
- Particularly suited for tabular data
- Implements powerful data operations
- Main data structures:
  - DataFrame: A table with rows and columns
  - Series: A single column

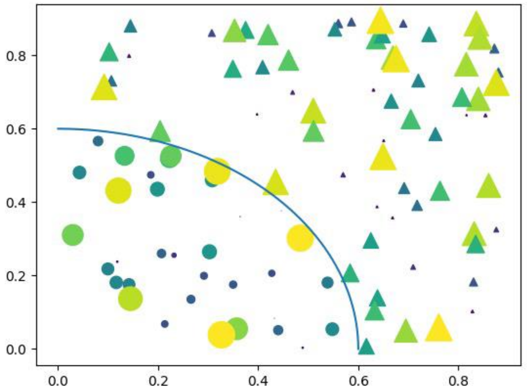
Matplotlib

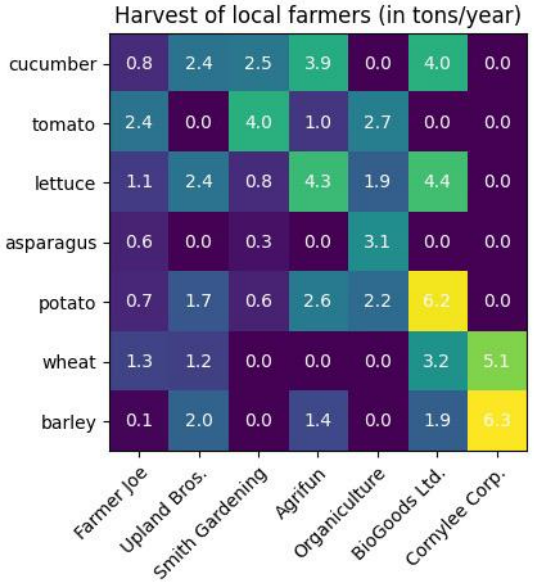
❖ Used for basic plotting

❖ Highly customizable

❖ Works with NumPy and pandas







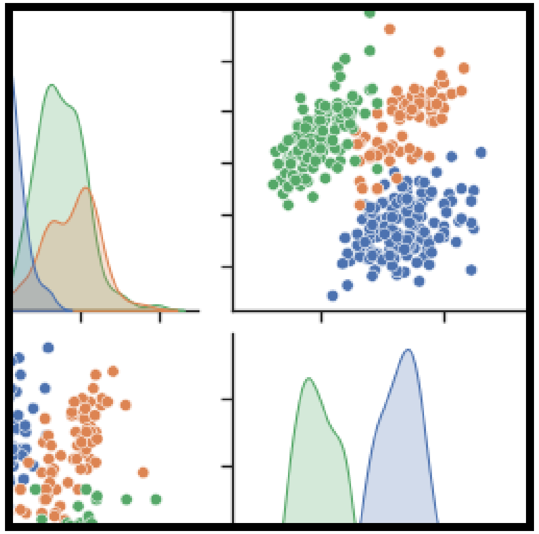
Seaborn

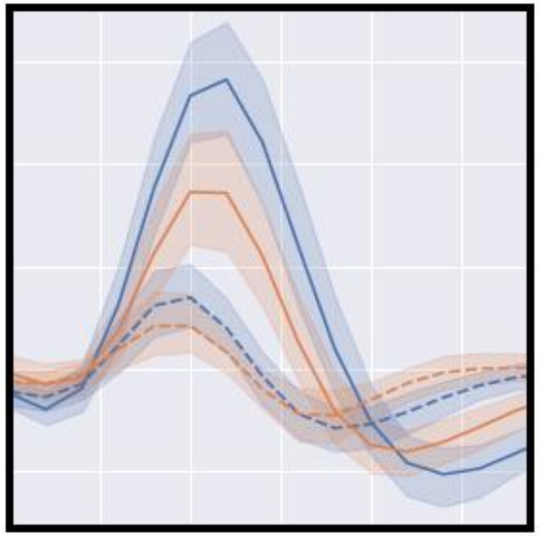
❖ Used for statistical data visualization

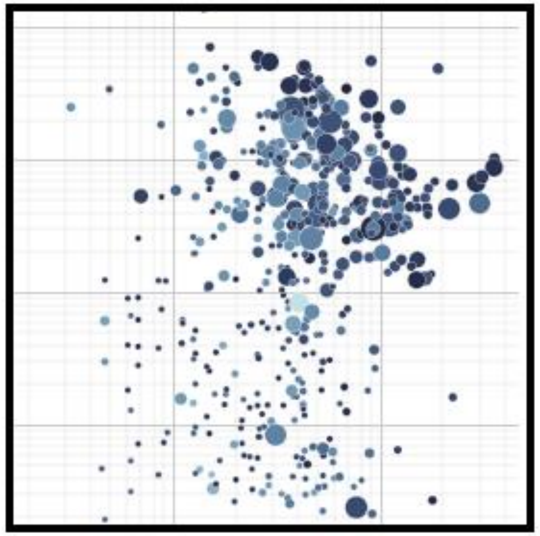
❖ Uses fewer syntax with good default themes

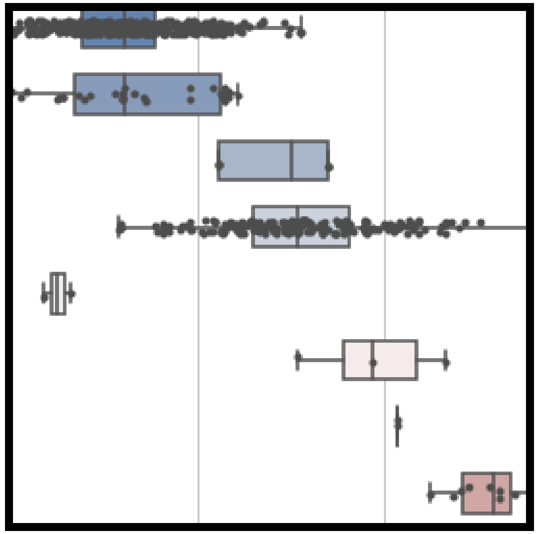
❖ Integrated to work great with pandas data-frame

❖ Uses Matplotlib under the hood

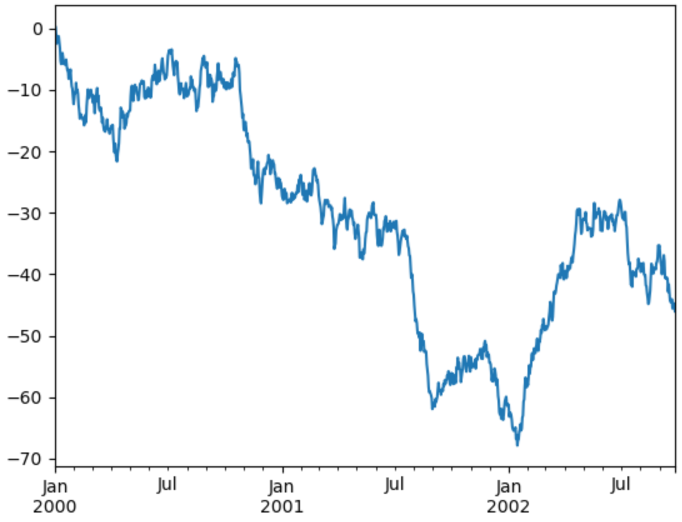


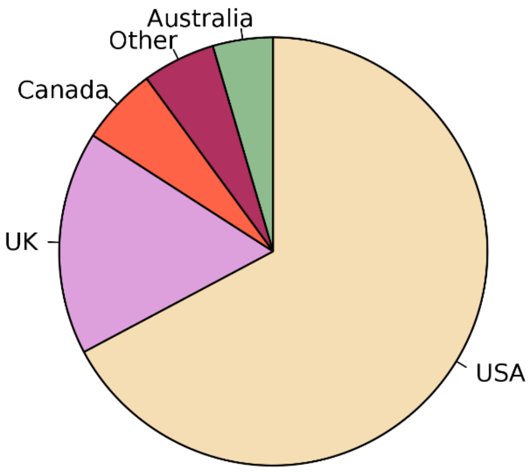


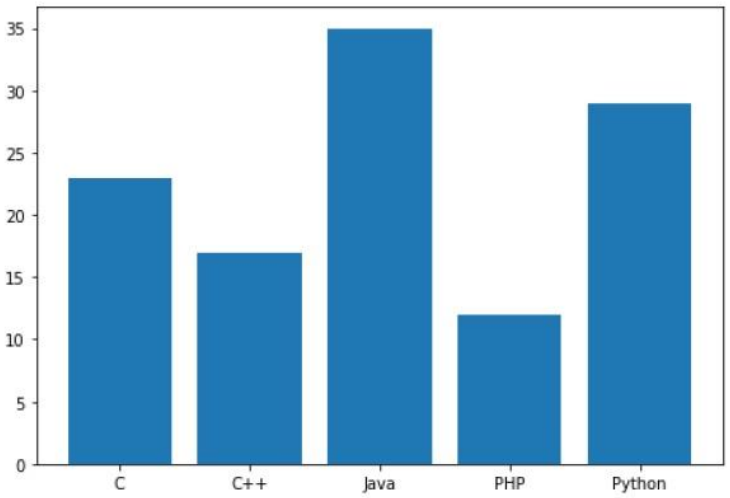


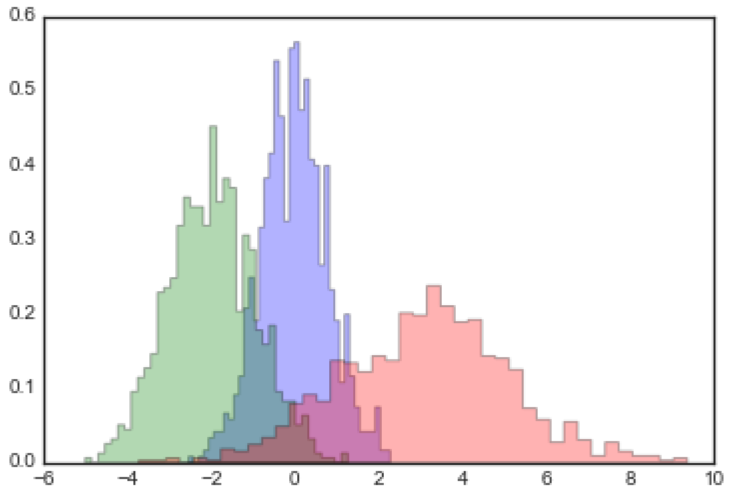


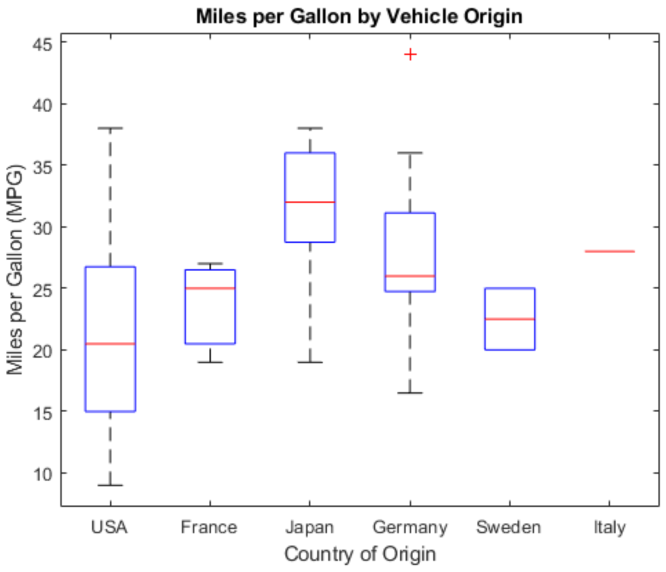
Frequently Used Plots (= Graphs)

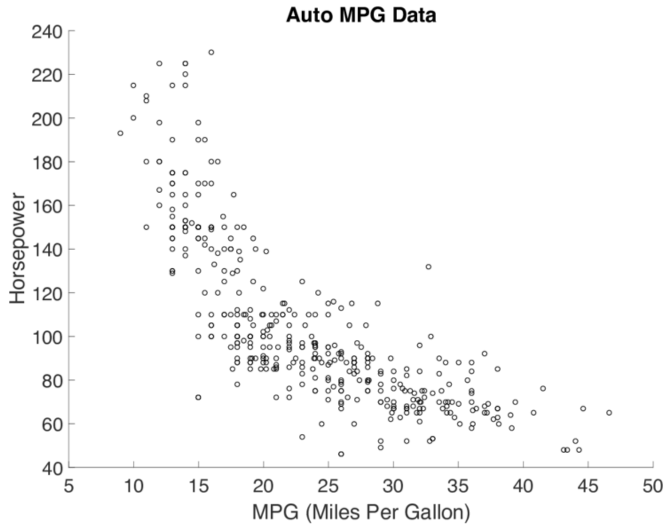






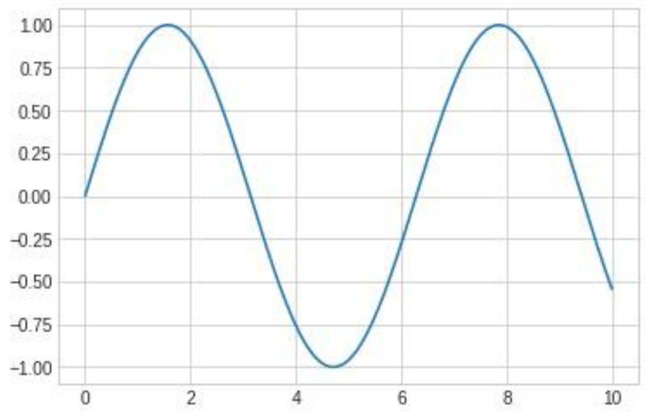






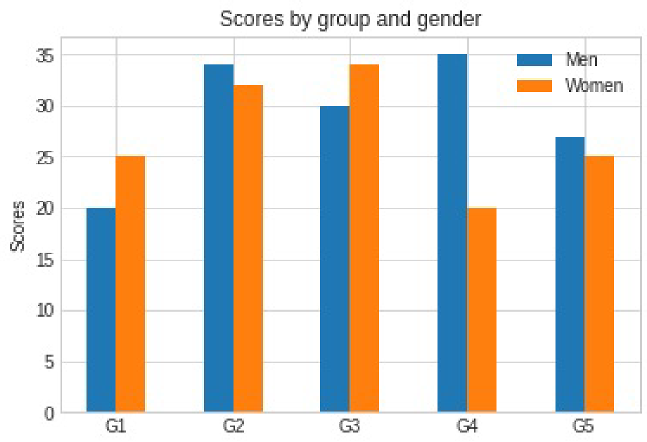
Line plots

- ❖ Used for numeric data
- ❖ Used to show trends
- ❖ Compare two or more different variables over time
- ❖ Could be used to make predictions



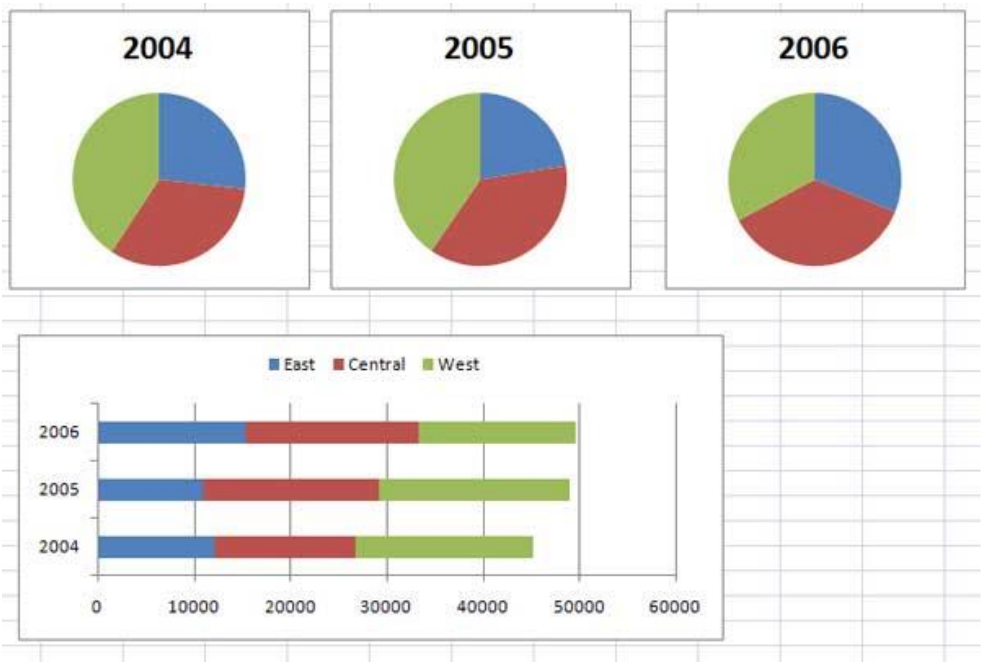
Bar plots

- ❖ Used for nominal or ordinal categories
- ❖ Compare data amongst different categories
- ❖ Ideal for more than 3 categories
- ❖ Can show large data changes over time



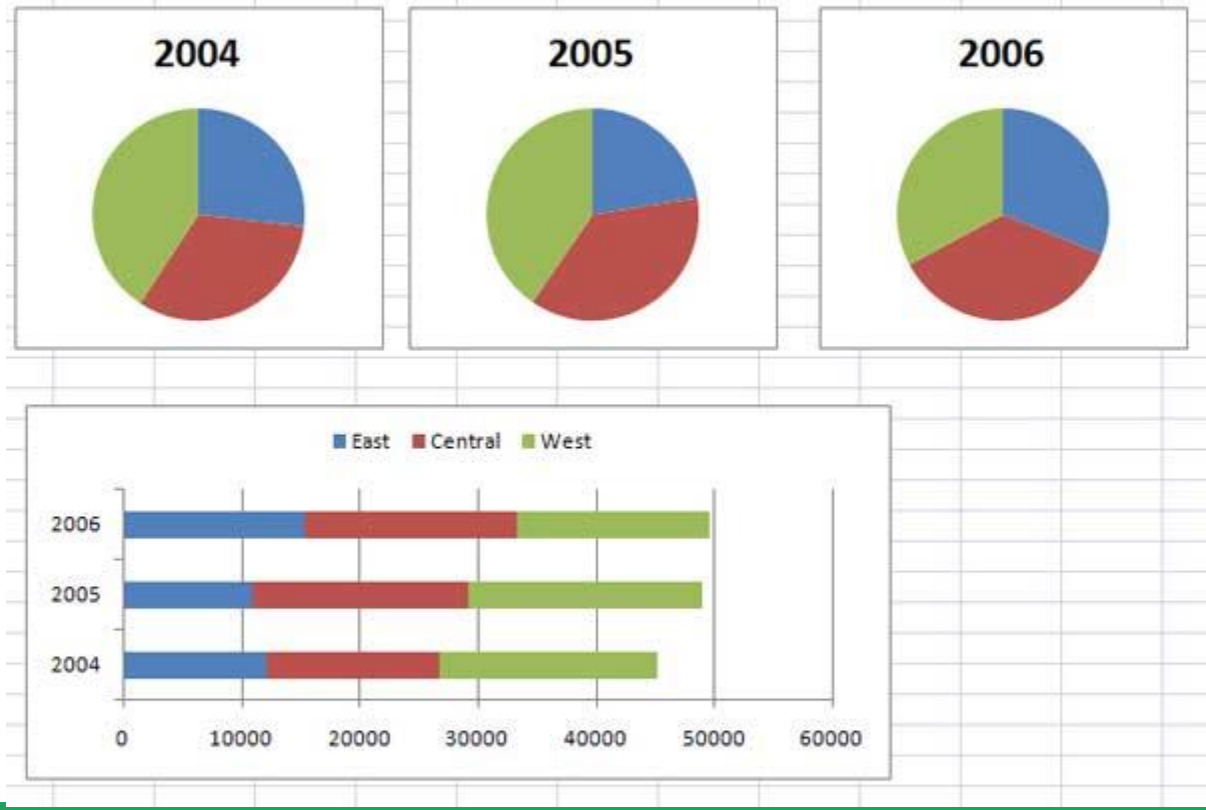
Bar Plots vs. Pie Charts (1/2)

- ❖ Bar plots
  - show the frequency of proportion of categorical variables
- ❖ Pie charts
  - use more space to read and compare

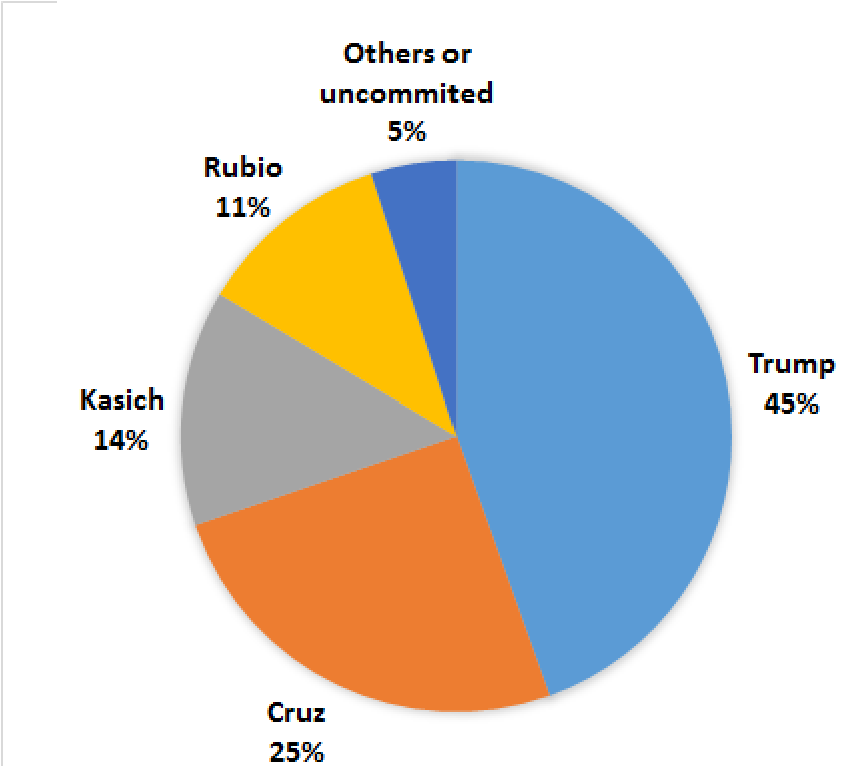


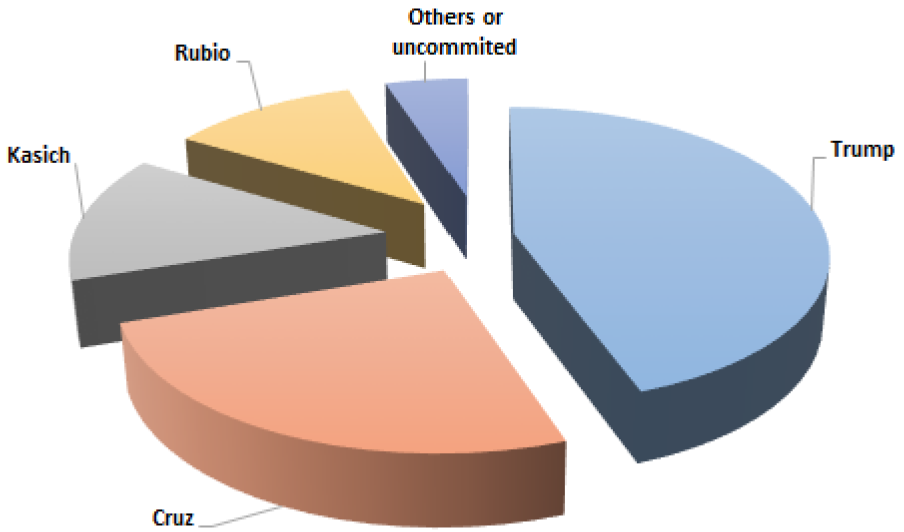
Bar Plots vs. Pie Charts (2/2)

- ❖ Partitioning each bar into pieces yields the stacked bar chart.
- ❖ Pie charts are arguably better for showing percentages of totality, and people do seem to like them, so they may be harmless in small amounts.



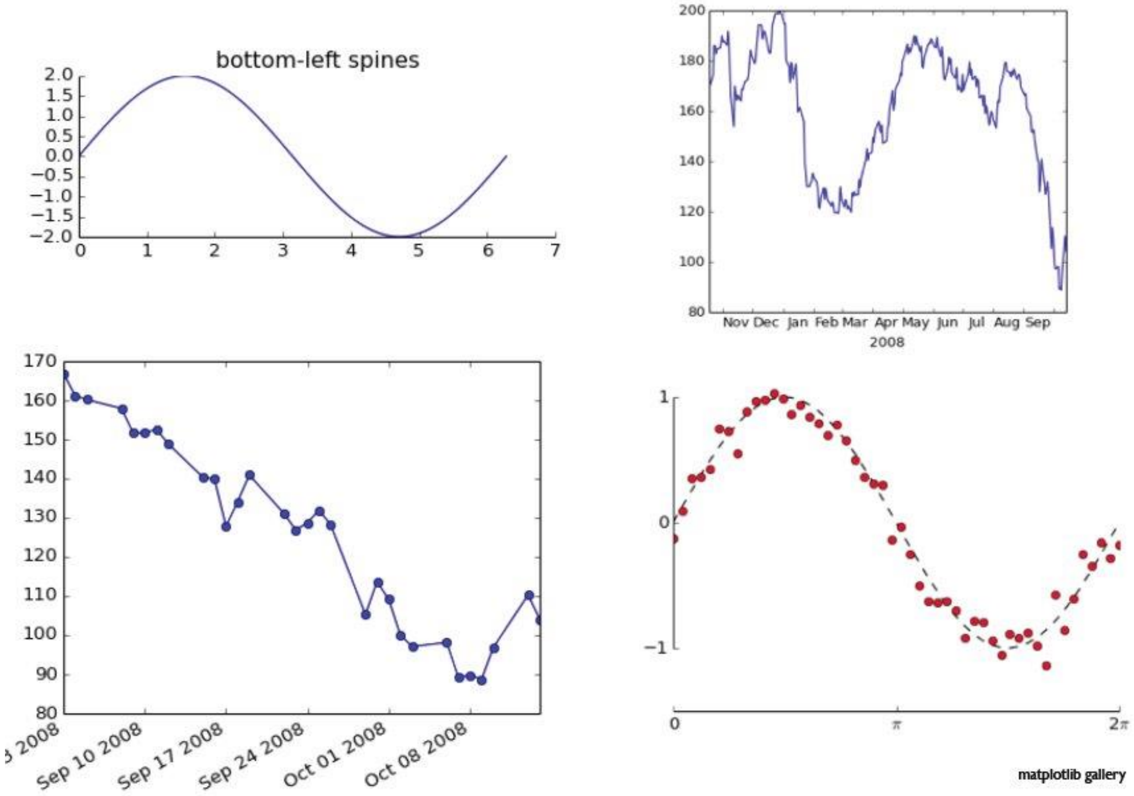
Which Pie Chart is Better?





Line Charts

- Show data points, not just fits.
- Line segments show connections, so do not use in categorical data.
- Connecting points by lines is often chartjunk.
  - Better is usually a trend line or fit with the data points.

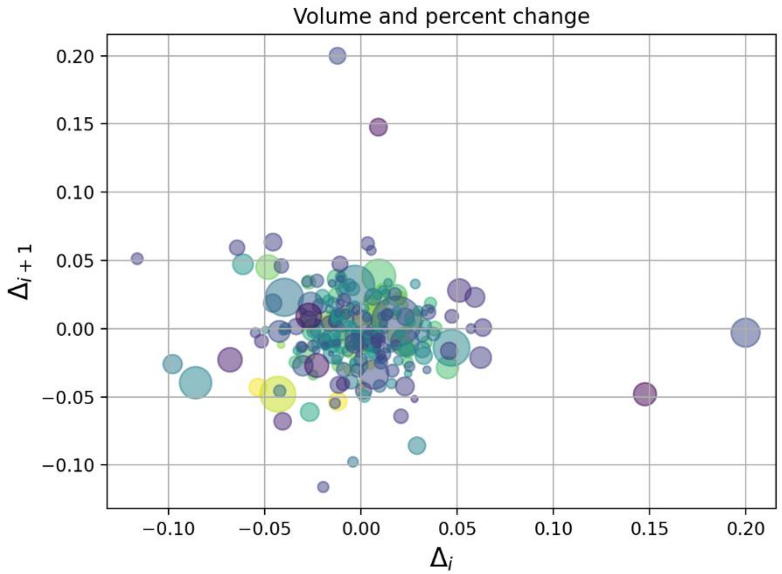


Scatter plots

❖ Used to visualize relation between two numeric variables

❖ Used to visualize correlation in a large data set

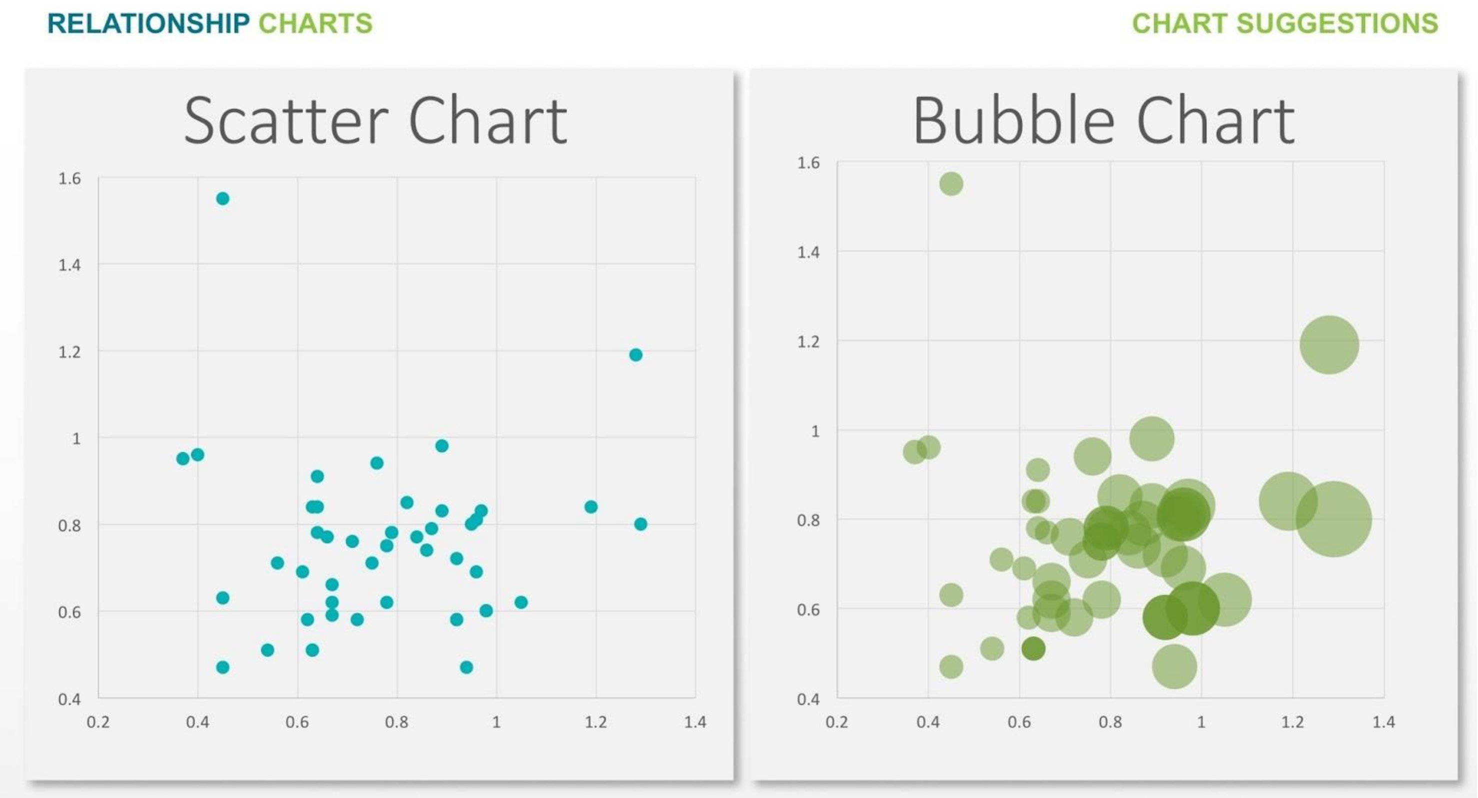
❖ Predict behavior of dependent variable based on the measure of the independent variable.



부산대학교
PUSAN NATIONAL UNIVERSITY

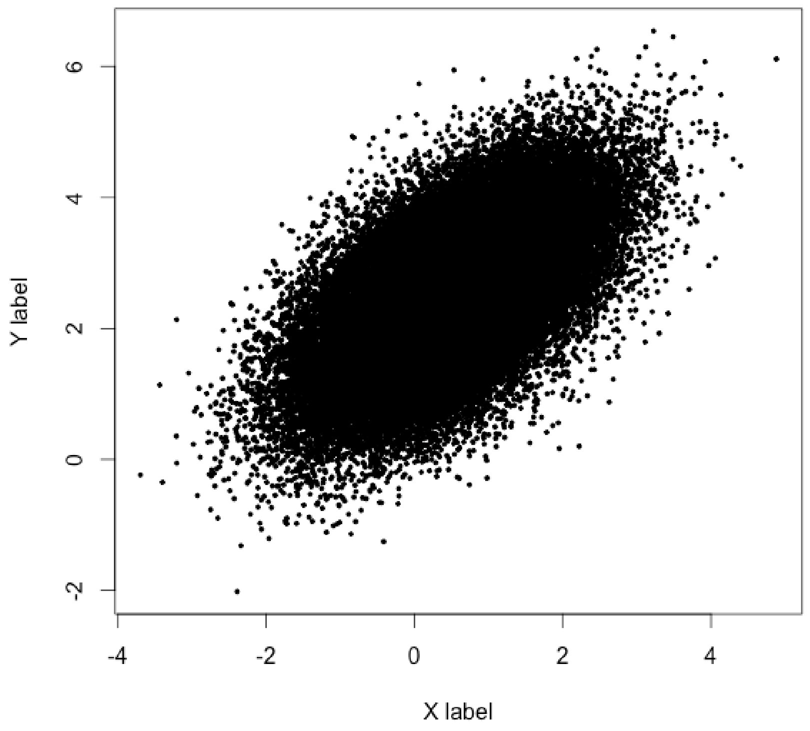
Scatter plots

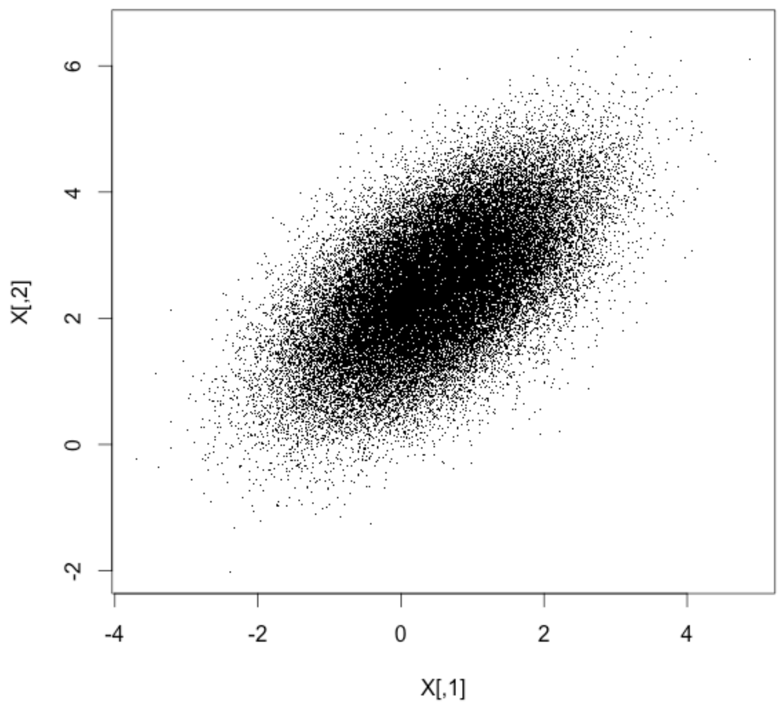
❖ Higher dimensional datasets are often best projected to 2D, through self-organizing maps, principle component analysis or bubble plots.



Scatter plots

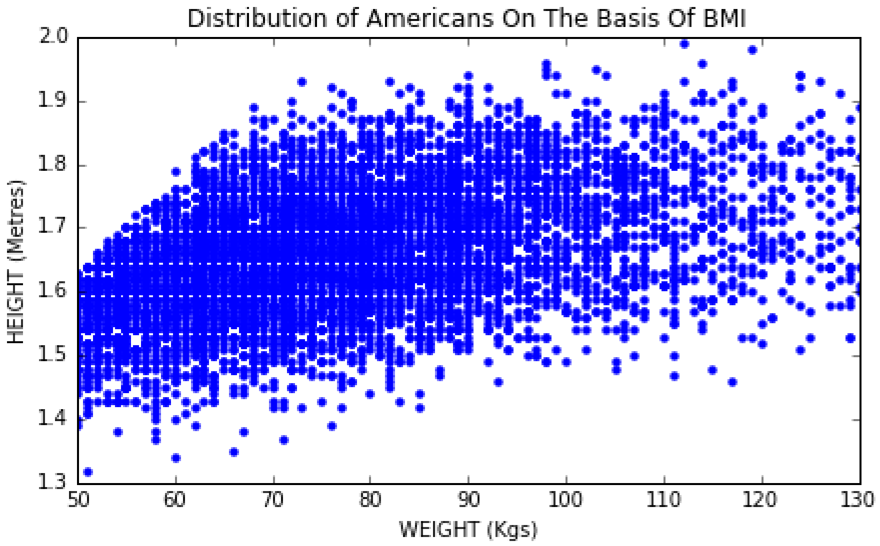
❖ Reduce Overplotting by Small Points

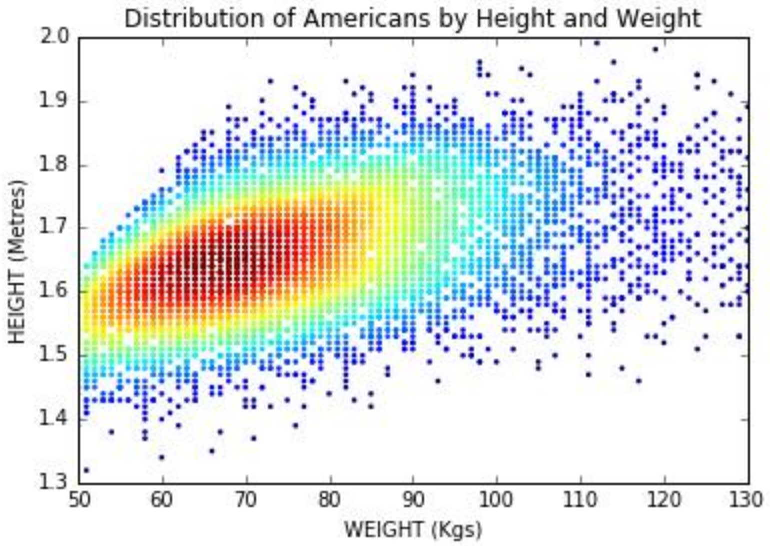




Scatter plots

❖ Heatmaps Reveal Finer Structure





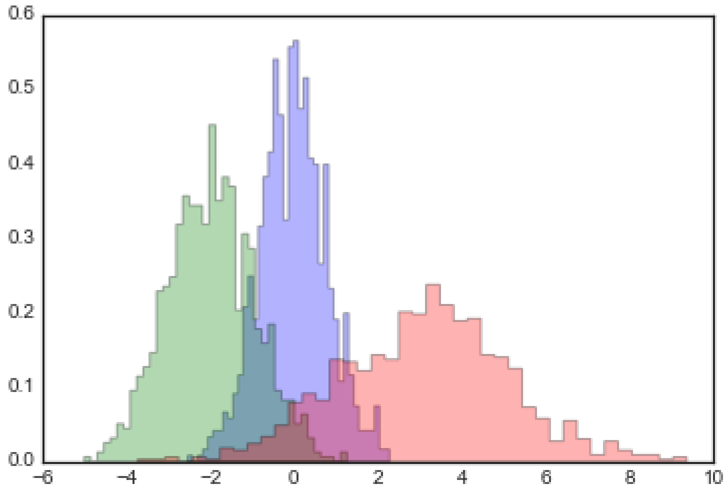
Histograms

❖ Used for continuous data

❖ Displays the frequency distribution (shape)

❖ Summarize large data sets graphically

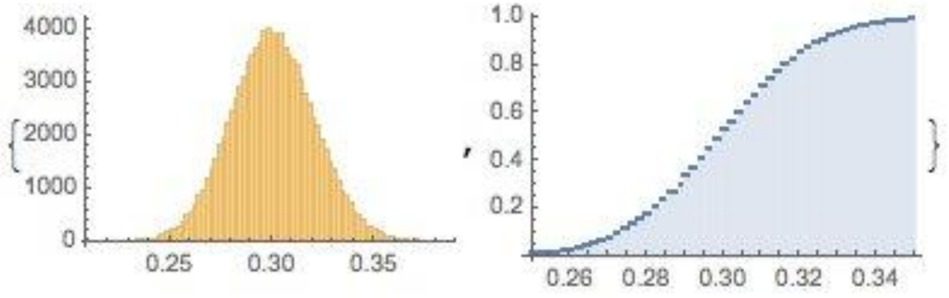
❖ Compare multiple distributions



Histograms

- Histograms (and CDFs) visualize distributions over continuous variables

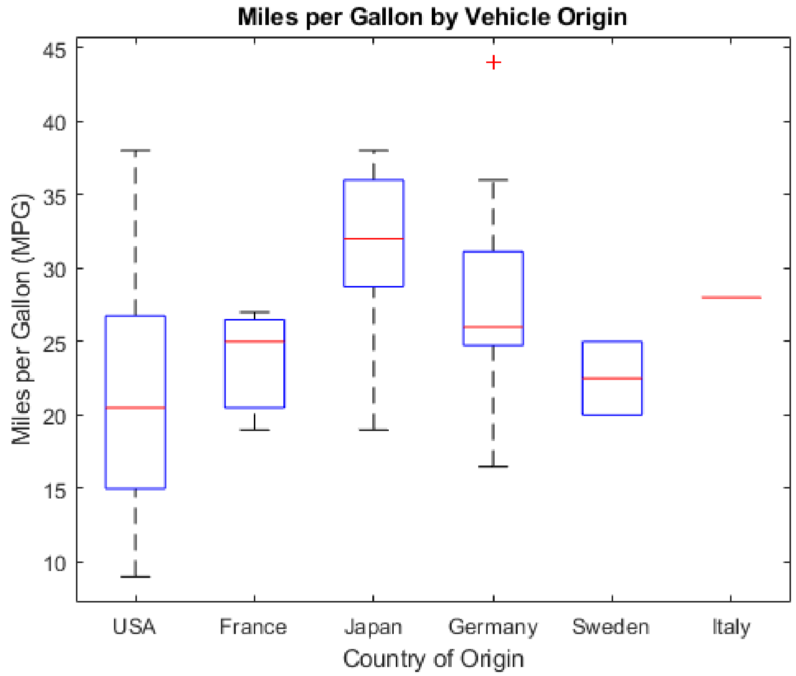
```
{Histogram[ d , 100], DiscretePlot[CDF[d1, x], {x, 0.25, 0.35, 0.001}]}
```



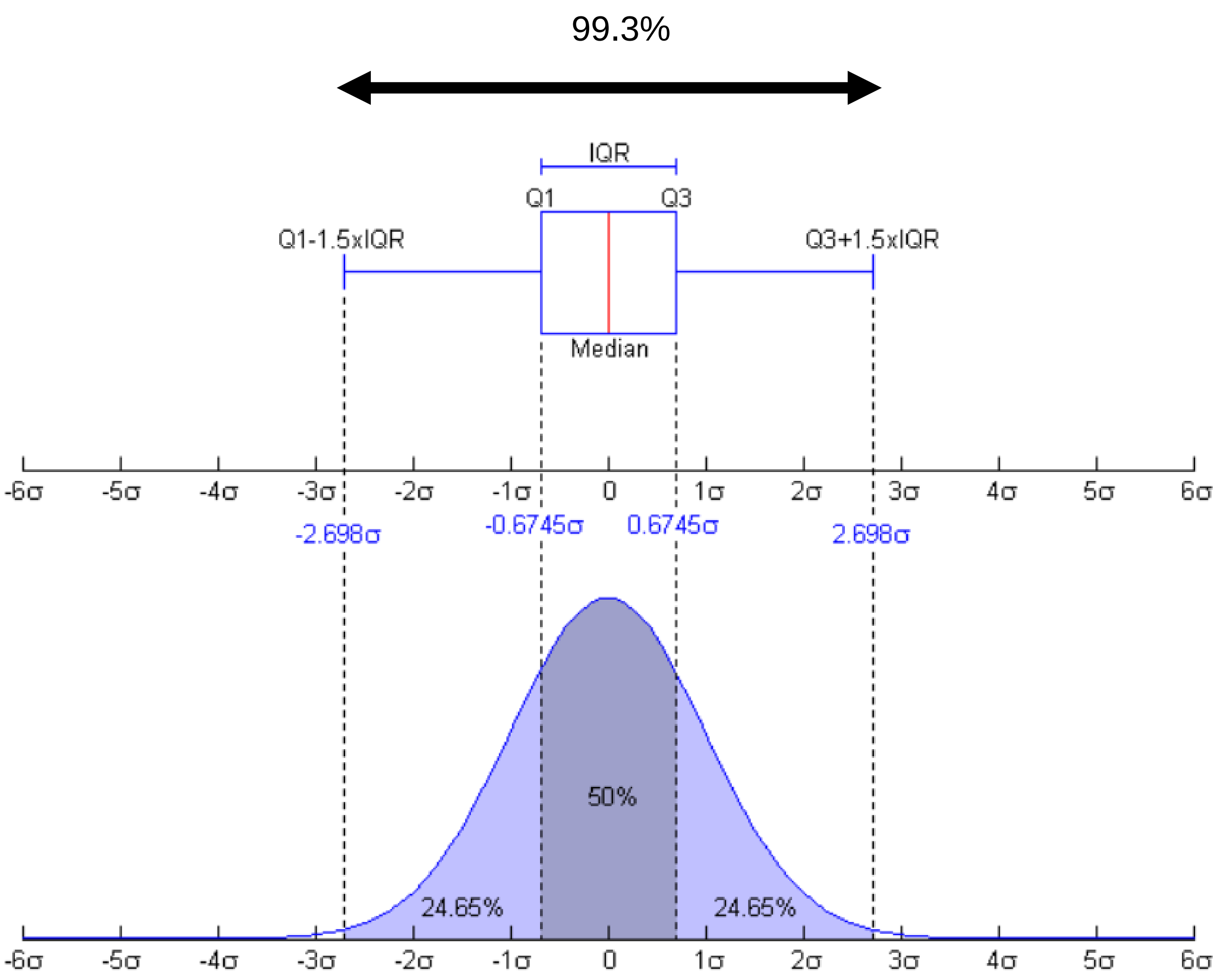
- Histograms are better for displaying peaks, CDFs for showing tails.

Box plots

- aka whisker plot
- Statistical graph used on sets of numerical data
- Shows the range, spread and center
- Used to compare data from different categories



Box plots



99.3%
IQR
Q1
Q3
Q1-1.5xIQR
Q3+1.5xIQR
Median
-6σ -5σ -4σ -3σ -2σ -1σ 0 1σ 2σ 3σ 4σ 5σ 6σ
-2.698σ
-0.6745σ
0.6745σ
2.698σ
50%
24.65%
24.65%
-6σ -5σ -4σ -3σ -2σ -1σ 0 1σ 2σ 3σ 4σ 5σ 6σ

Tabular Data

❖ Tables can have advantages over plots:

- Representation of numerical precision
- Understandable multivariate visualization: each column is a different dimension.
- Representation of heterogeneous data
- Compactness for small numbers of points

Can this Table be Improved?

Country	Area	Density	Birthrate	Population	Mortality	GDP
Russia	17075200	8.37	99.6	142893540	15.39	8900.0
Mexico	1972550	54.47	92.2	107449525	20.91	9000.0
Japan	377835	337.35	99.0	127463611	3.26	28200.0
United Kingdom	244820	247.57	99.0	60609153	5.16	27700.0
New Zealand	268680	15.17	99.0	4076140	5.85	21600.0
Afghanistan	647500	47.96	36.0	31056997	163.07	700.0
Israel	20770	305.83	95.4	6352117	7.03	19800.0
United States	9631420	30.99	97.0	298444215	6.5	37800.0
China	9596960	136.92	90.9	1313973713	24.18	5000.0
Tajikistan	143100	51.16	99.4	7320815	110.76	1000.0
Burma	678500	69.83	85.3	47382633	67.24	1800.0
Tanzania	945087	39.62	78.2	37445392	98.54	600.0
Tonga	748	153.33	98.5	114689	12.62	2200.0
Germany	357021	230.86	99.0	82422299	4.16	27600.0
Australia	7686850	2.64	100.0	20264082	4.69	29000.0

Dimensions for Improvement

- Order rows to invite comparisons.
- Order columns to highlight importance or pairwise relationships.
- Right justify uniform-precision numbers
- Use **emphasis**, *font*, or color to highlight important entries.
- Avoid excessive-length column descriptors.

Improved Tabular Presentation

Country	Population	Area	Density	Mortality	GDP	Birth Rate
Afghanistan	31,056,997	647,500	47.96	**163.07**	700	36.0
Australia	20,264,082	7,686,850	2.64	4.69	29,000	**100.0**
Burma	47,382,633	678,500	69.83	67.24	1,800	85.3
China	**1,313,973,713**	9,596,960	136.92	24.18	5,000	90.9
Germany	82,422,299	357,021	230.86	4.16	27,600	99.0
Israel	6,352,117	20,770	305.83	7.03	19,800	95.4
Japan	127,463,611	377,835	**337.35**	3.26	28,200	99.0
Mexico	107,449,525	1,972,550	54.47	20.91	9,000	92.2
New Zealand	4,076,140	268,680	15.17	5.85	21,600	99.0
Russia	142,893,540	**17,075,200**	8.37	15.39	8,900	99.6
Tajikistan	7,320,815	143,100	51.16	110.76	1,000	99.4
Tanzania	37,445,392	945,087	39.62	98.54	600	78.2
Tonga	114,689	748	153.33	12.62	2,200	98.5
United Kingdom	60,609,153	244,820	247.57	5.16	27,700	99.0
United States	298,444,215	9,631,420	30.99	6.50	**37,800**	97.0

Tabular Data

❖ Some key rules for table layout are the following

1. Do not use vertical lines.
2. Do not use horizontal lines between data rows.
3. Text columns should be left aligned.
4. Number columns should be right aligned and should use the same number of decimal digits throughout.
5. Columns containing single characters are centered.
6. The header fields are aligned with their data, i.e., the heading for a text column will be left aligned and the heading for a number column will be right aligned.

Tabular Data

a — ugly

Rank	Title	Amount
1	*Star Wars: The Last Jedi*	$71,565,498
2	*Jumanji: Welcome to the Jungle*	$36,169,328
3	*Pitch Perfect 3*	$19,928,525
4	*The Greatest Showman*	$8,805,843
5	*Ferdinand*	$7,316,746

b — ugly

Rank	Title	Amount
1	*Star Wars: The Last Jedi*	$71,565,498
2	*Jumanji: Welcome to the Jungle*	$36,169,328
3	*Pitch Perfect 3*	$19,928,525
4	*The Greatest Showman*	$8,805,843
5	*Ferdinand*	$7,316,746

c

Rank	Title	Amount
1	*Star Wars: The Last Jedi*	$71,565,498
2	*Jumanji: Welcome to the Jungle*	$36,169,328
3	*Pitch Perfect 3*	$19,928,525
4	*The Greatest Showman*	$8,805,843
5	*Ferdinand*	$7,316,746

d

Rank	Title	Amount
1	*Star Wars: The Last Jedi*	$71,565,498
2	*Jumanji: Welcome to the Jungle*	$36,169,328
3	*Pitch Perfect 3*	$19,928,525
4	*The Greatest Showman*	$8,805,843
5	*Ferdinand*	$7,316,746

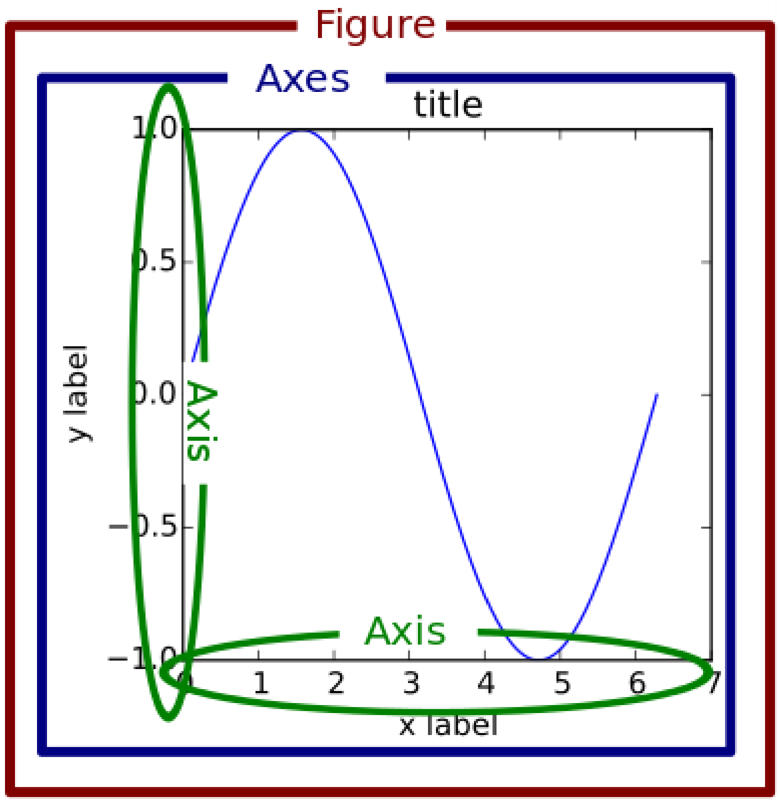
How to use matplotlib

❖ Import

```python
import matplotlib.pyplot as plt
```

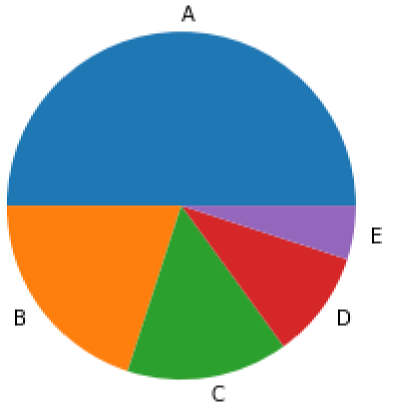
❖ Anatomy of a Matplotlib plot

- A plot is a hierarchy of objects
- Outermost: Figure object
  - Can contain multiple Axes objects
- Axes object
  - Actually represents a 'plot' or 'graph'
  - Contain many smaller object types
- Smaller object types
  - Tick marks, individual lines, legends, text boxes, ...



Pie Chart example (1/3)

```
labels = ['A', 'B', 'C', 'D', 'E']
x = np.array([50, 20, 15, 10, 5])
plt.pie(x, labels=labels)
plt.show()
```



Pie Chart example (2/3)

❖ If you want use a Korean font

- 1) install nanum font

```
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
```

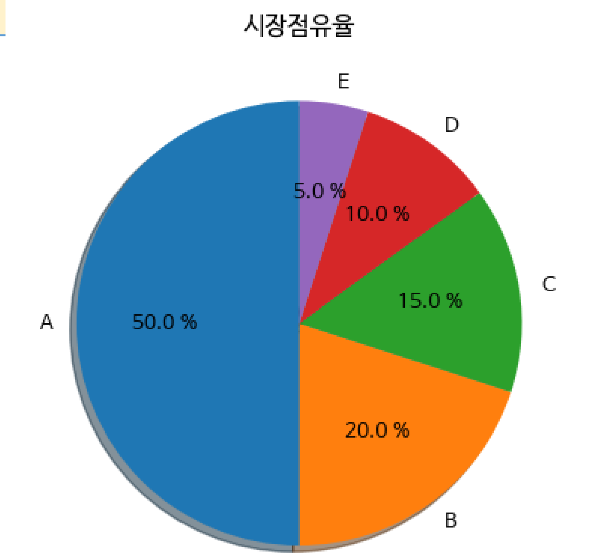
- 2) excute runtime again

Pie Chart example (3/3)

❖ If you want use a Korean font

- 3) use a Korean font

```
plt.rc('font', family='NanumBarunGothic')
plt.pie(x, labels=labels, shadow=True, startangle=90, autopct='%0.1f %%')
plt.title('시장점유율')
plt.show()
```

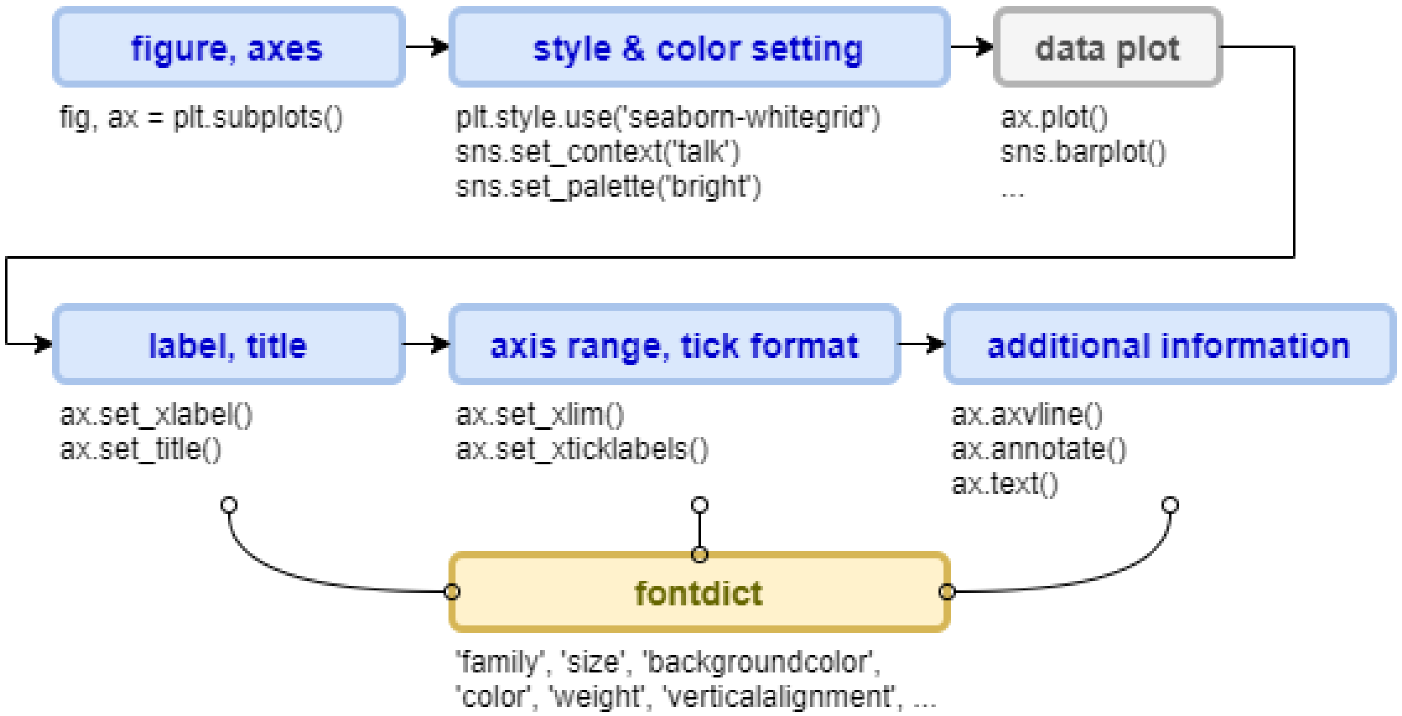


Two Matplotlib application interfaces

- an explicit "Axes" interface (object-oriented)
  - a Figure or Axes object to create other Artists
  - build a visualization step by step.

- An implicit "pyplot" interface (state-based)
  - keeps track of the last Figure and Axes created
  - adds Artists to the object it thinks the user wants.

OO style steps for Matplotlib



subplot() function

```python
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
```

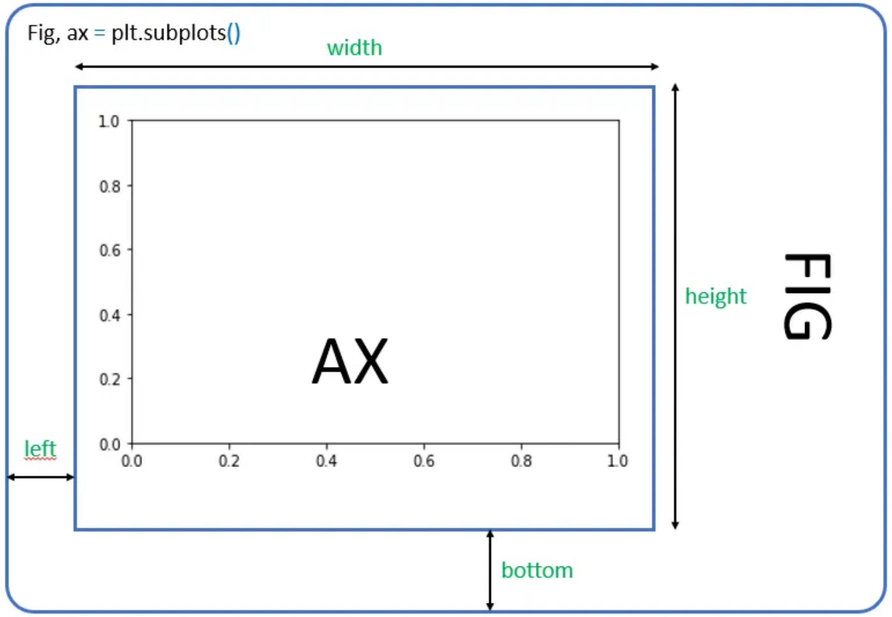
plt.subplots()

fig

Figure: Represents the container of the entire axis system.

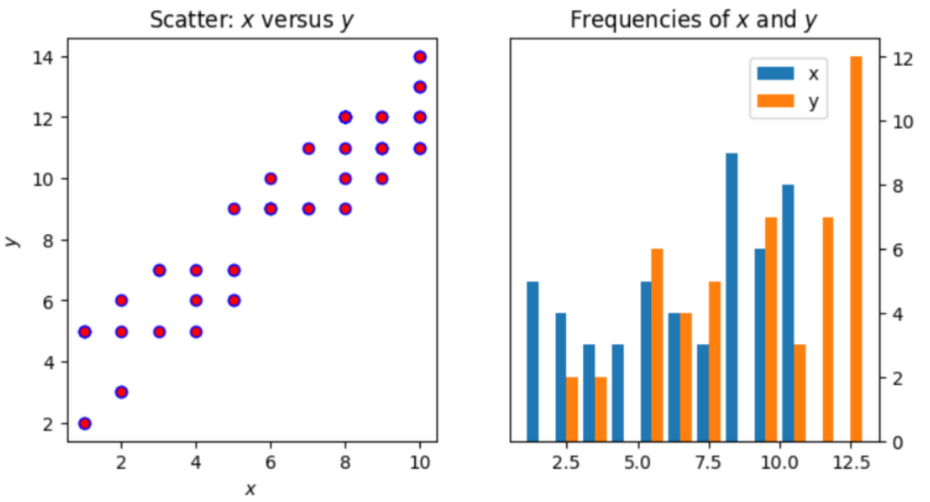
ax

Axis system: Contains information on the type of graph, color, etc.



Axes example

```
d1 = np.random.randint(low=1, high=11, size=50)
d2 = d1 + np.random.randint(1, 5, size=d1.size)
data = np.column_stack((d1, d2))
data
```




```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(8, 4))
ax1.scatter(x=d1, y=d2, marker='o', c='r', edgecolor='b')
ax1.set_title('Scatter: $x$ versus $y$')
ax1.set_xlabel('$x$')
ax1.set_ylabel('$y$')

ax2.hist(data, bins=np.arange(data.min(), data.max()),
          label=('x', 'y'))
ax2.legend(loc=(0.65, 0.8))
ax2.set_title('Frequencies of $x$ and $y$')
ax2.yaxis.tick_right()
```

Types of Attributes

- ❖ All values in a column of a table should be both the same type and be comparable to each other in some way

- ❖ **Numerical** — Each value is from a numerical scale
  - Numerical measurements are ordered
  - Differences are meaningful

- ❖ **Categorical** — Each value is from a fixed inventory
  - May or may not have an ordering
  - Categories are the same or different

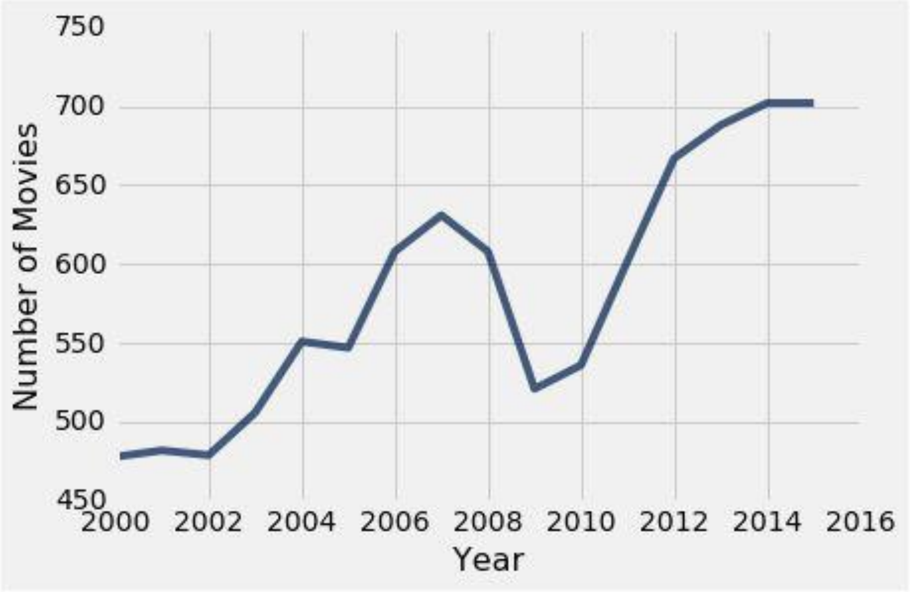
Numerical Attributes

❖ Just because the values are numbers, doesn't mean the attribute is numerical

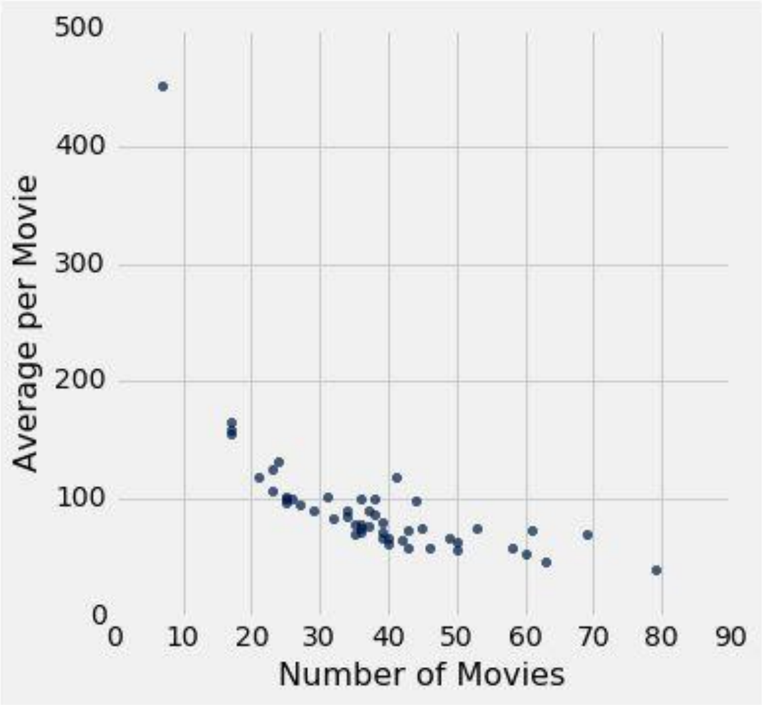
- Census example has numerical SEX code (0, 1, and 2)
- It doesn't make sense to perform arithmetic on these "numbers", e.g. (0+1+2)/3 is meaningless
- The attribute SEX is still categorical, even though numbers were used for the categories

Plotting Two Numerical Variables using Matplotlib

- Line plot
  - plot(x,y)



- Scatter plot
  - Scatter(x,y)



Line vs Scatter Plot Demo (1/2)

❖ Import libs

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```python
# Read csv movies_by_year
movies_by_year = pd.read_csv('https://raw.githubusercontent.com/data-8/textbook/main/assets/data/movies_by_year.csv')
movies_by_year
```

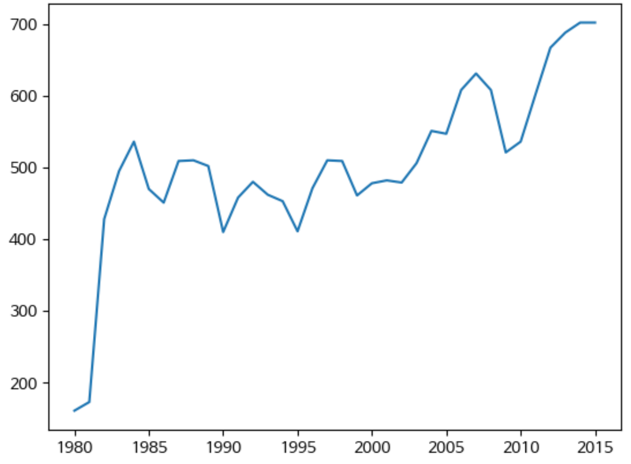
```python
# upload actor.csv file
from google.colab import files
file_uploaded = files.upload()
```

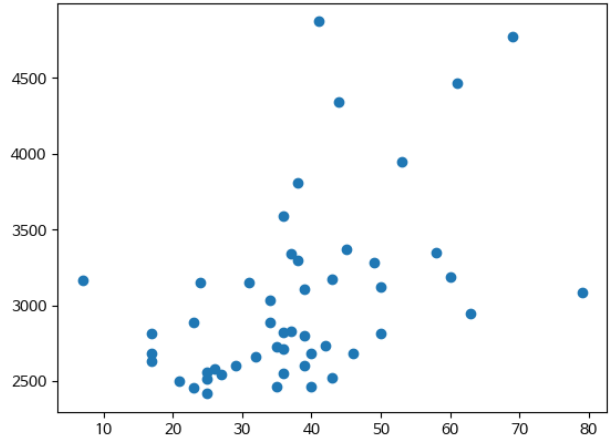
```python
actors = pd.read_csv('actors.csv')
actors
```

Line vs Scatter Plot Demo (2/2)

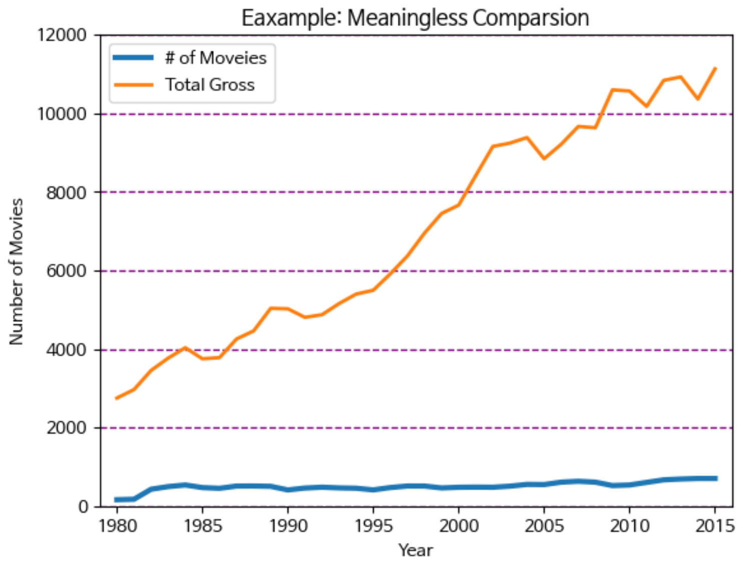
❖ Plotting

```
# plotting a line graph
print("Line graph: ")
plt.plot(movies_by_year["Year"], movies_by_year["Number of Movies"])
plt.show()

# plotting a scatter plot
print("Scatter Plot: ")
plt.scatter(actors["Number of Movies"], actors["Total Gross"])
plt.show(
```





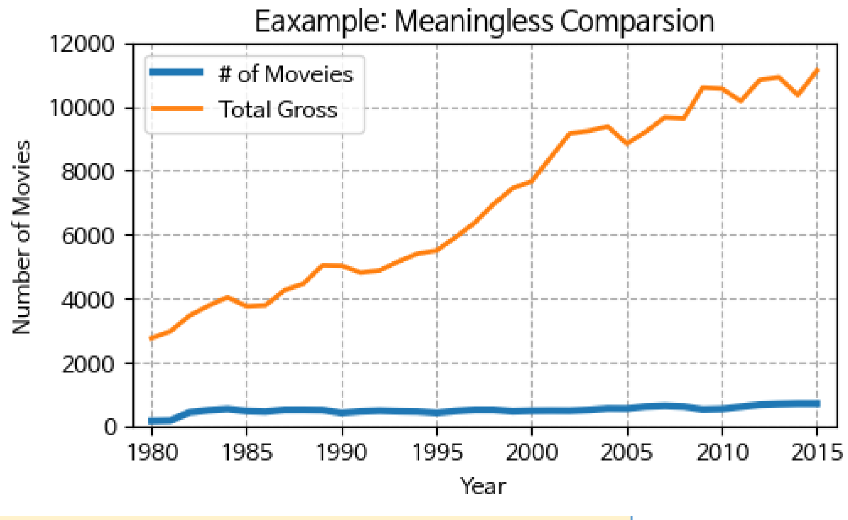
Let's draw a complex(?) graph (1/



❖ Plot style

```
# plot style
plt.plot(movies_by_year["Year"], movies_by_year["Number of Movies"],          linewidth=3.0,
label='# of Moveies')
plt.plot(movies_by_year["Year"], movies_by_year["Total Gross"],      linewidth=2.0,
label='Total Gross')
plt.title("Eaxample: Meaningless Comparsion")
plt.xlabel("Year")
plt.ylabel("Number of Movies")
plt.legend()
plt.axis([1979, 2016, 0, 12000])
plt.grid(axis = 'y', color = 'purple', linestyle = '--', linewidth = 1)
plt.show()
```

Let's draw a complex(?) graph (2



❖ OO style

```
# oo style
# Note that even in the OO-style,
# we use `.pyplot.figure` to create the Figure.
fig, ax = plt.subplots(figsize=(5, 3), layout='constrained')
ax.plot(movies_by_year["Year"], movies_by_year["Number of Movies"],  linewidth=3.0, label='# of Moveies')
ax.plot(movies_by_year["Year"], movies_by_year["Total Gross"],       linewidth=2.0, label='Total Gross')
ax.set_title("Eaxample: Meaningless Comparsion")
ax.set_xlabel("Year")
ax.set_ylabel("Number of Movies")
ax.legend()
ax.set_xlim([1979,2016])
ax.set_ylim([0,12000])
#ax.grid(axis = 'y', color = 'purple', linestyle = '--', linewidth = 1)
ax.grid(True, linestyle = '--')
plt.show()
```

Bar Chart (1/3)

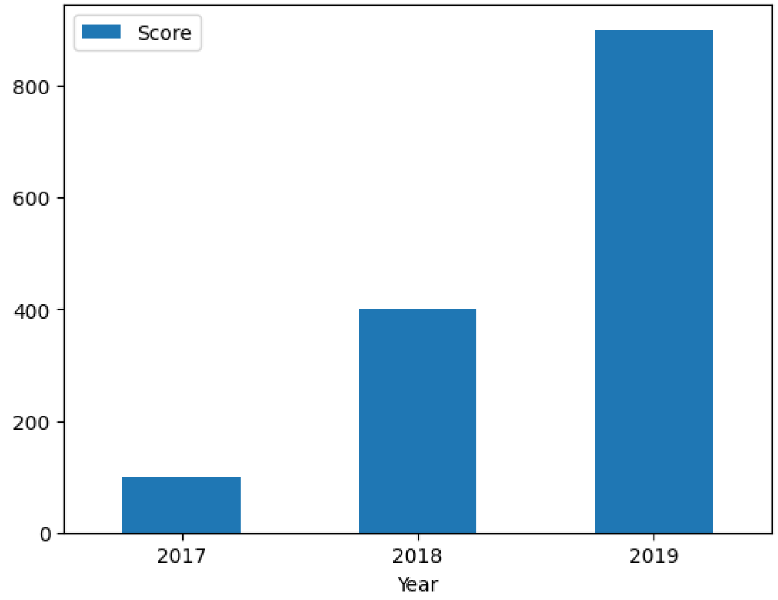
❖ Using DataFrame plot()

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

```
x = np.arange(3) # 0, 1, 2
years = ['2017', '2018', '2019']
values = [100, 400, 900]
```

```
myDF = pd.DataFrame({'Year':years,
        'Score':values})
myDF.head()
```

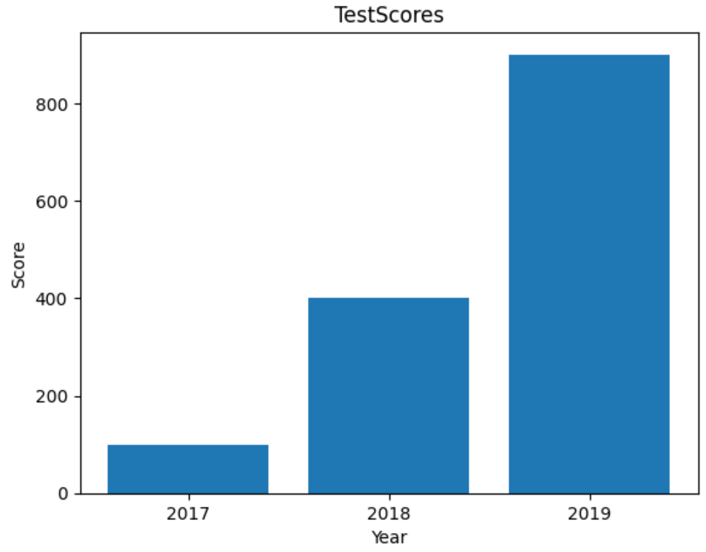
```
myDF.plot.bar(x='Year', y='Score', rot=0)
```



Bar Chart (2/3)

❖ Using matplotlib

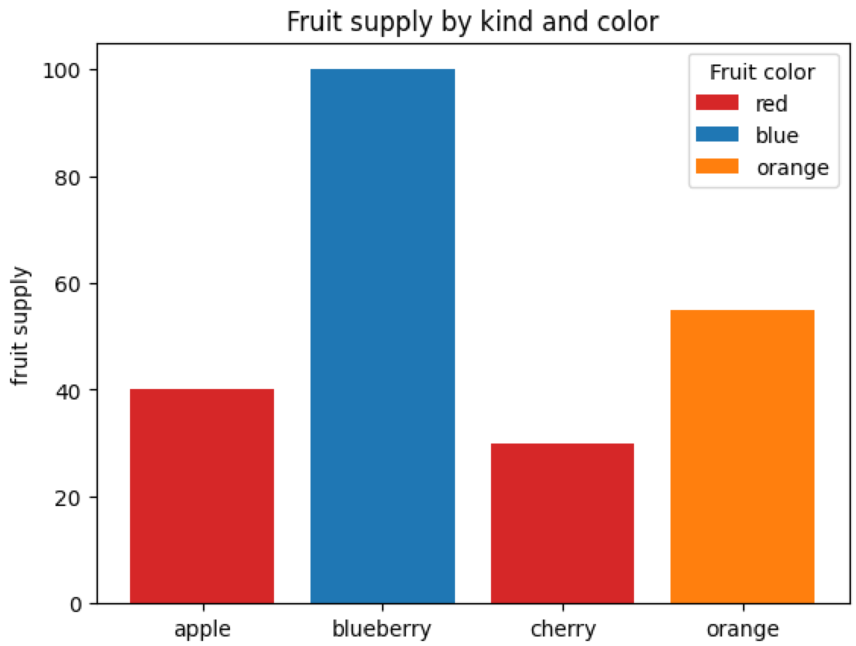
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()

ax.bar(myDF.Year, myDF.Score)
ax.set_title('TestScores')
ax.set_xlabel('Year')
ax.set_ylabel('Score')

plt.show()
```



Bar Chart (3/3)

❖ Bar Color Demo

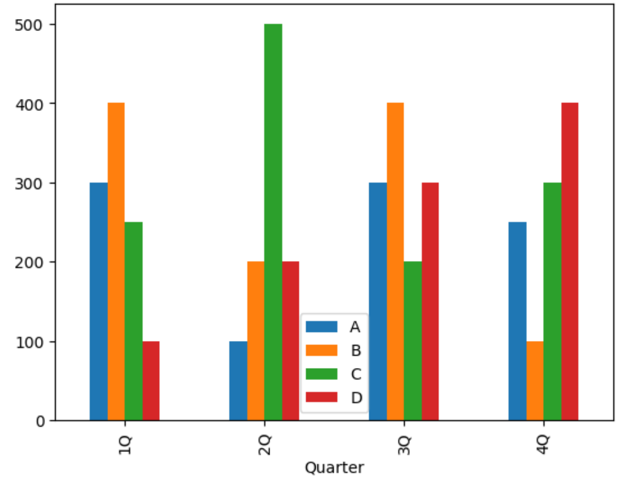
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

fruits = ['apple', 'blueberry', 'cherry', 'orange']
counts = [40, 100, 30, 55]
bar_labels = ['red', 'blue', '_red', 'orange']
bar_colors = ['tab:red', 'tab:blue',
        'tab:red', 'tab:orange']

ax.bar(fruits, counts,
       label=bar_labels, color=bar_colors)

ax.set_ylabel('fruit supply')
ax.set_title('Fruit supply by kind and color')
ax.legend(title='Fruit color')

plt.show()
```



Stacked Bar Chart (1/2)

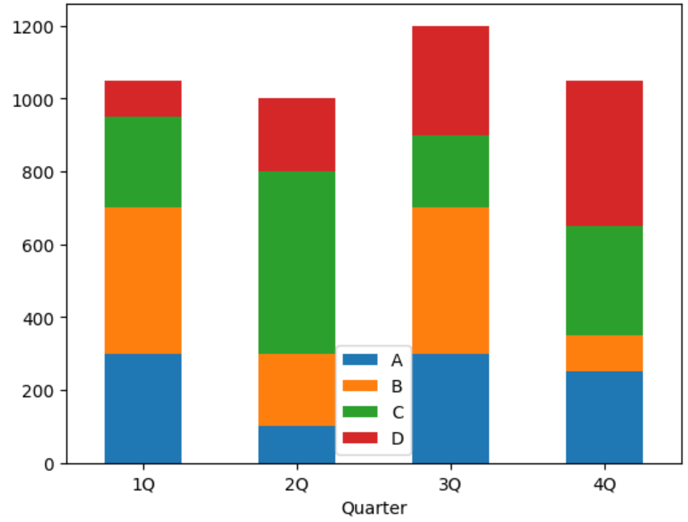
❖ Using Dataframe.plot()

```
df = pd.DataFrame()
df['Quarter'] = ['1Q','2Q','3Q','4Q']
df['A'] = [300,100,300,250]
df['B'] = [400,200,400,100]
df['C'] = [250,500,200,300]
df['D'] = [100,200,300,400]
df.head()
```




```
ax = df.plot.bar(x = 'Quarter')
```

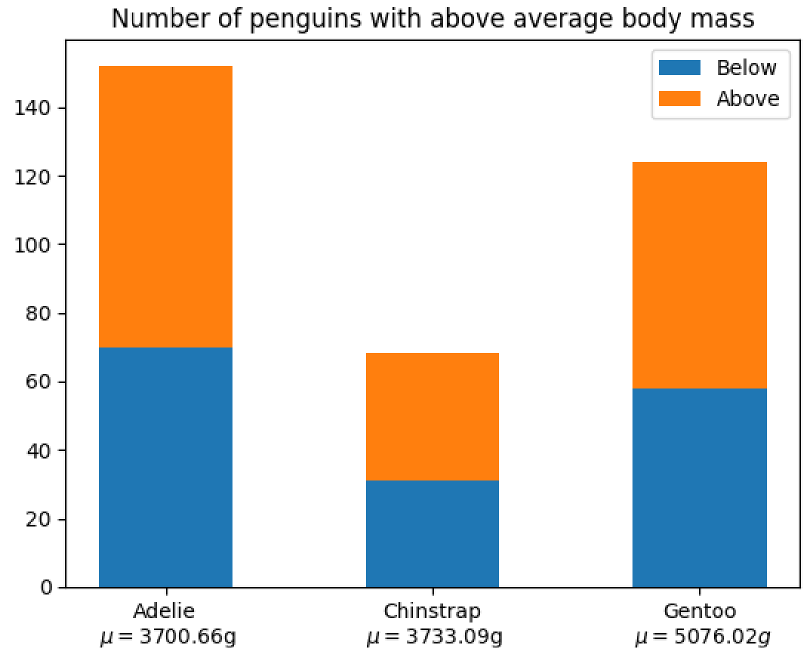
```
ax = df.plot.bar(stacked = True,\
    x = 'Quarter', rot=0)
```



Stacked Bar Chart (2/2)

❖ Using matplotlib

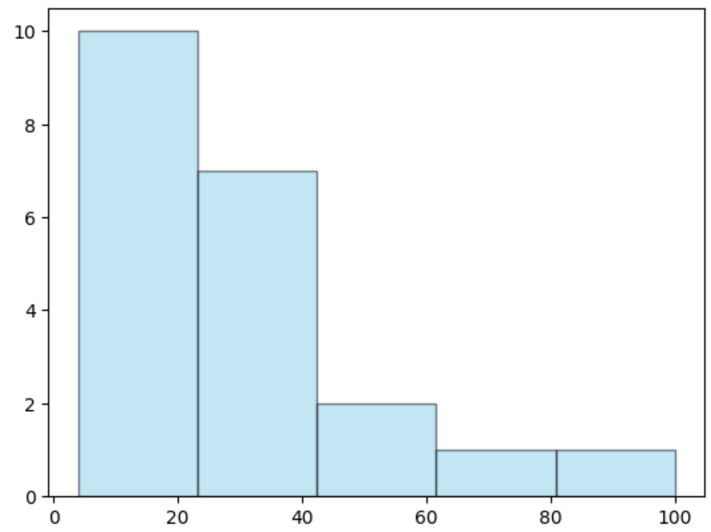
```
species = (
    "Adelie\n $\\mu=$3700.66g",
    "Chinstrap\n $\\mu=$3733.09g",
    "Gentoo\n $\\mu=5076.02g$",
)
weight_counts = {
    "Below": np.array([70, 31, 58]),
    "Above": np.array([82, 37, 66]),
}
width = 0.5

fig, ax = plt.subplots()
bottom = np.zeros(3)

for boolean, weight_count in weight_counts.items():
    p = ax.bar(species, weight_count, width, label=boolean, bottom=bottom)
    bottom += weight_count

ax.set_title("Number of penguins with above average body mass")
ax.legend(loc="upper right")

plt.show()
```



Histogram: Matplotlib (1/3)

❖ Simple example

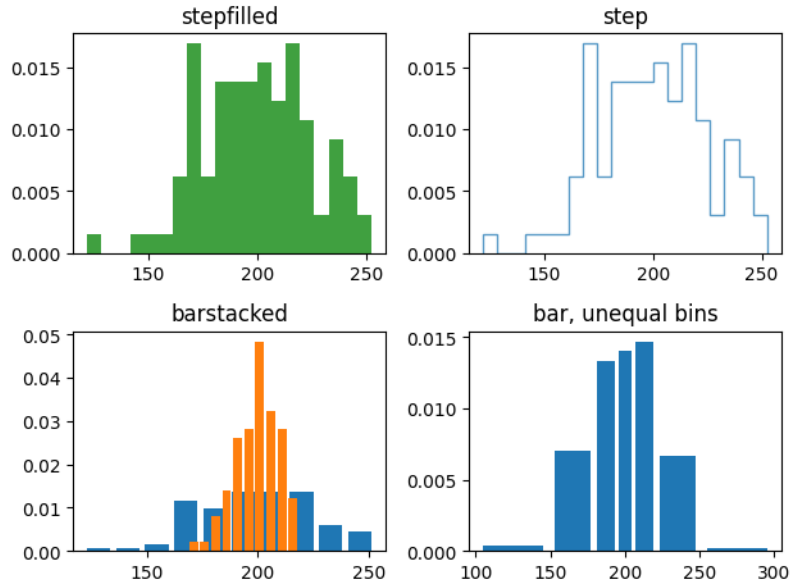
```
import numpy as np
import matplotlib.pyplot as plt

x = [21,22,23,4,5,6,77,8,9,10,31,32,33,34,35,\
        36,37,18,49,50,100]
num_bins = 5

n, bins, patches = plt.hist(x, num_bins, \
        color='skyblue', alpha=0.5,ec='black')
plt.show()
```



Histogram: Matplotlib (2/3)

❖ different histtype settings

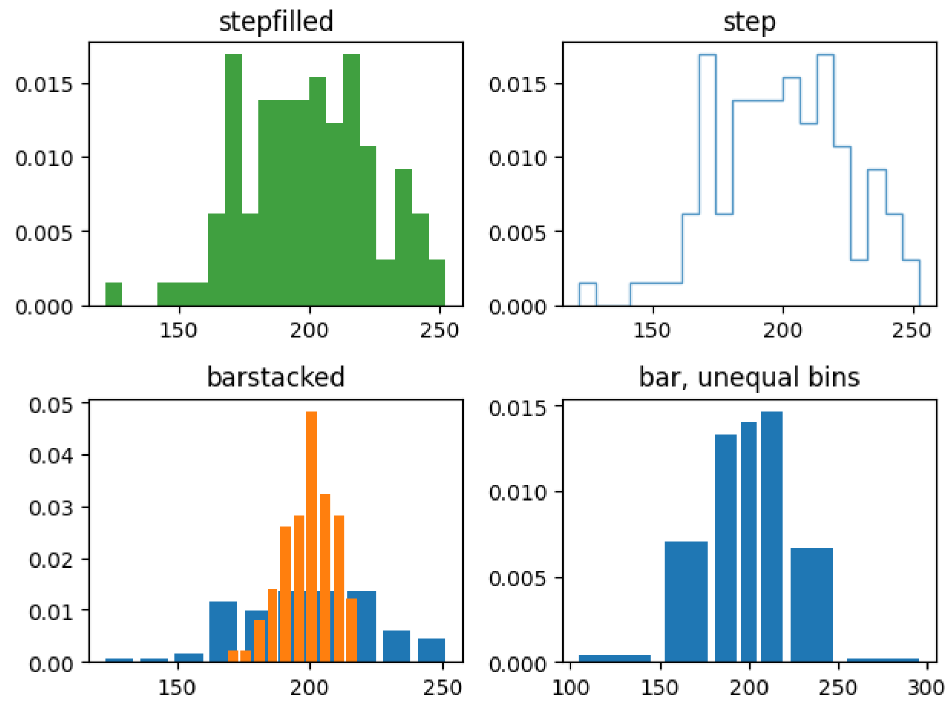
```
np.random.seed(19680801)

mu_x = 200
sigma_x = 25
x = np.random.normal(mu_x, sigma_x, size=100)

mu_w = 200
sigma_w = 10
w = np.random.normal(mu_w, sigma_w, size=100)
```



Histogram: Matplotlib (

❖ different histtype settings (cont'd)

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

axs[0, 0].hist(x, 20, density=True, histtype='stepfilled', \
        facecolor='g',alpha=0.75)
axs[0, 0].set_title('stepfilled')

axs[0, 1].hist(x, 20, density=True, histtype='step', \
        facecolor='g', alpha=0.75)
axs[0, 1].set_title('step')

axs[1, 0].hist(x, density=True, histtype='barstacked', rwidth=0.8)
axs[1, 0].hist(w, density=True, histtype='barstacked', rwidth=0.8)
axs[1, 0].set_title('barstacked')

# Create a histogram by providing the bin edges (unequally spaced).
bins = [100, 150, 180, 195, 205, 220, 250, 300]
axs[1, 1].hist(x, bins, density=True, histtype='bar', rwidth=0.8)
axs[1, 1].set_title('bar, unequal bins')

fig.tight_layout()
plt.show()
```



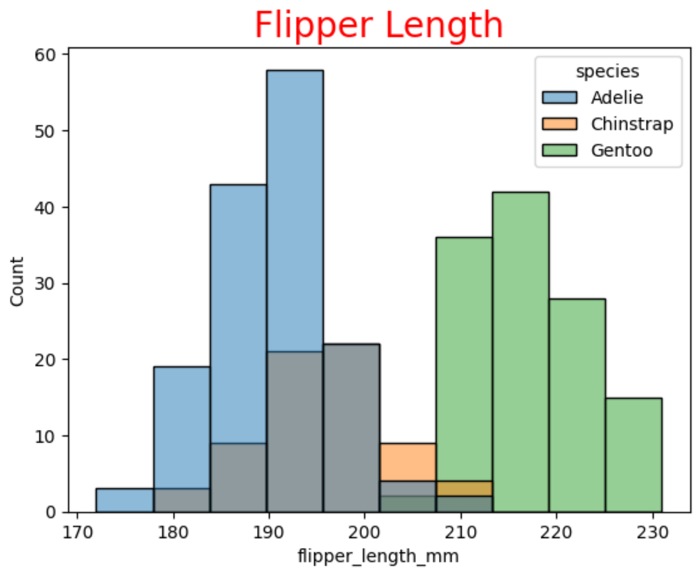
Histogram: Seaborn

❖ Load dataset

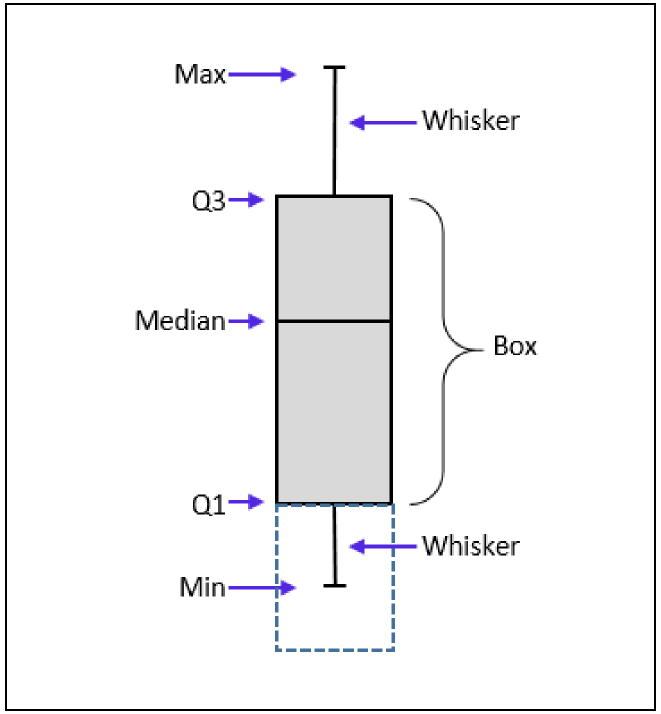
```
import seaborn as sns
import matplotlib.pyplot as plt
penguins = sns.load_dataset("penguins")
penguins
```

❖ Draw histogram

```
#sns.displot(penguins, x="flipper_length_mm")
fig, ax = plt.subplots()
ax=sns.histplot(x = "flipper_length_mm", \
          data = penguins, hue = "species")
plt.title("Flipper Length", size=20, \
          color="red")
plt.show()
```



Box Plot



- **Quartiles**: Q_1 (25^{th} percentile), Q_3 (75^{th} percentile)
- **Inter-quartile range**: $IQR = Q_3 - Q_1$
- **Five number summary**: min, Q_1, median, Q_3, max
- **Boxplot**: Data is represented with a box
  - Q_1, Q_3, IQR: The ends of the box are at the first and third quartiles, i.e., the height of the box is IQR
  - Median (Q_2) is marked by a line within the box
  - Whiskers: two lines outside the box extended to Minimum and Maximum

Box Plot: Matplotlib (1/2)

❖ Load dataset from seaborn

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
# loading dataset
iris = sns.load_dataset('iris')
iris.shape
```

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

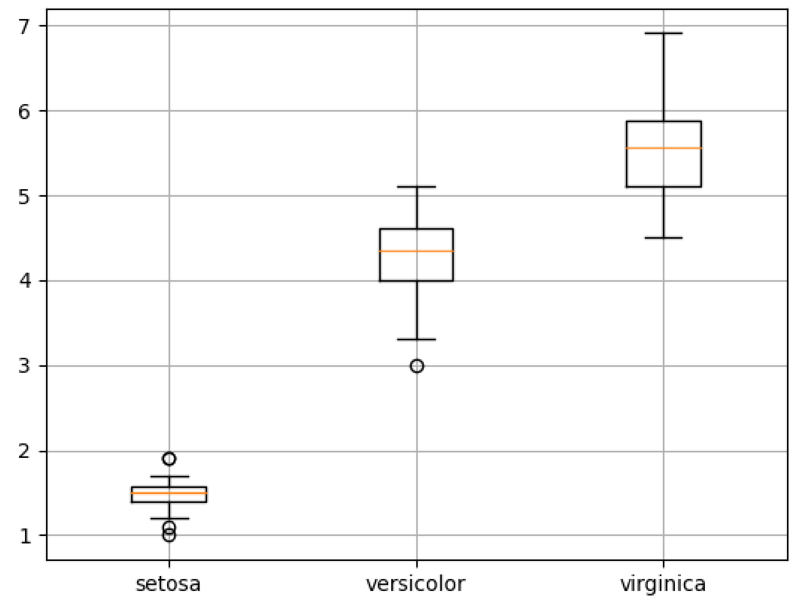
```
iris.groupby('species').count()
```

	sepal_length	sepal_width	petal_length	petal_width
species				
setosa	50	50	50	50
versicolor	50	50	50	50
virginica	50	50	50	50

Box Plot: Matplotlib (2/2)

❖ variables

```
c1 = iris[iris['species'] == 'setosa']
c2 = iris[iris['species'] == 'versicolor']
c3 = iris[iris['species'] == 'virginica']
```

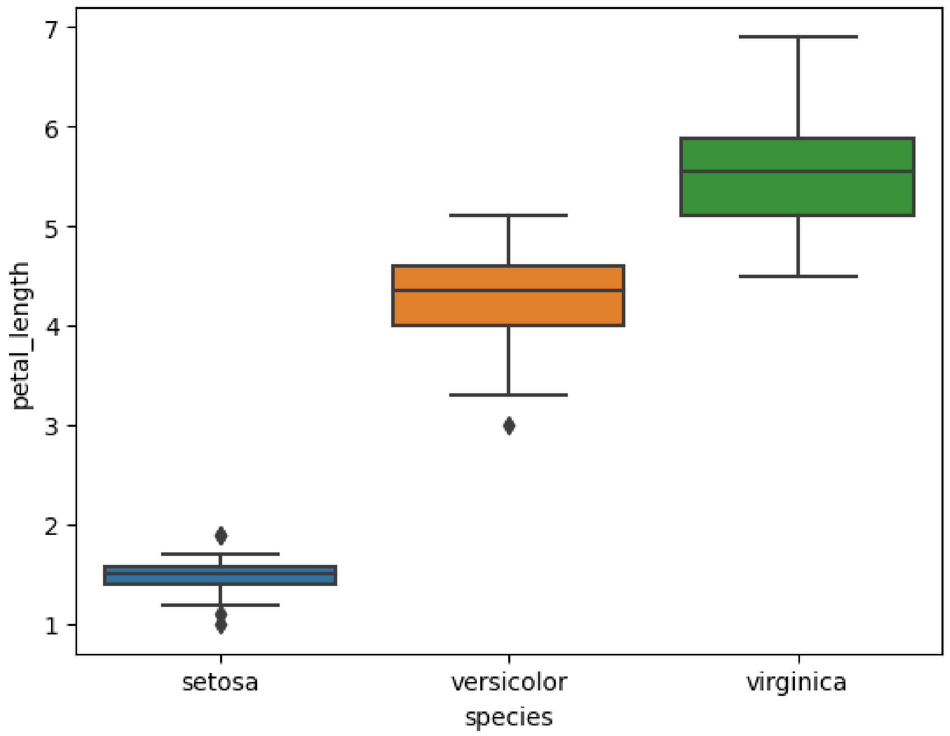


❖

```
plt.boxplot((c1['petal_length'], c2['petal_length'],\
    c3['petal_length']))
plt.xticks([1,2,3],['setosa','versicolor','virginica'])
#plt.grid()
plt.show()
```

Box Plot: Seaborn (1/2)

```
fig, ax = plt.subplots()
ax = sns.boxplot(data=iris, x='species’, \
         y='petal_length')
plt.show()
```



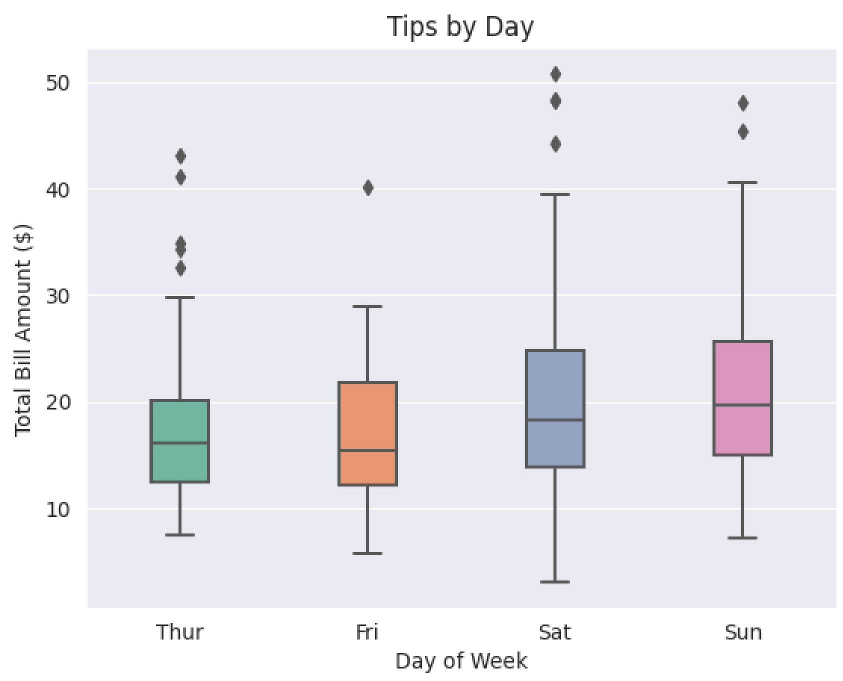
Box Plot: Seaborn (2/2)

❖ Load a 'tips' dataset

```
df = sns.load_dataset('tips')
df.head()
```

❖ Draw a box plot

```
sns.set_style('darkgrid')
sns.set_palette('Set2')

fig, ax = plt.subplots()
ax=sns.boxplot(data=df, x='day', y='total_bill', width=0.3)
ax.set_title('Tips by Day')
ax.set_xlabel('Day of Week')
ax.set_ylabel('Total Bill Amount ($)')
plt.show()
```

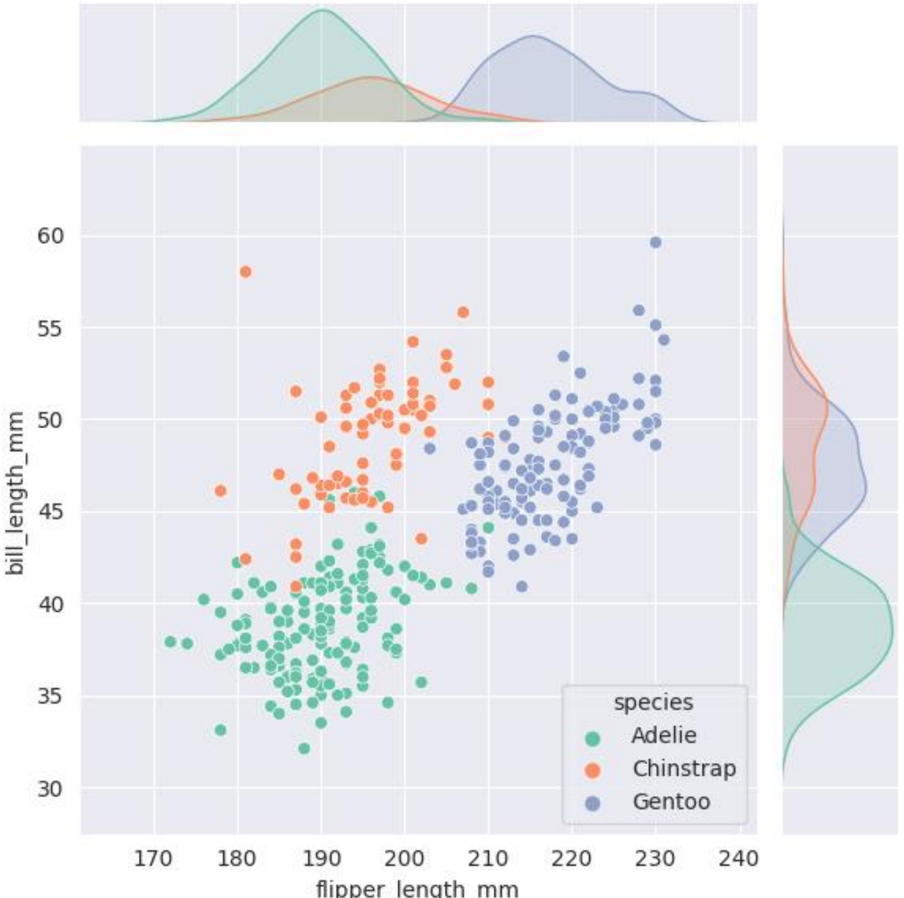


Seaborn: More graphs (1/2)

❖ Jointplot()

- plots the joint distribution between two variables along with each variable's marginal distribution:

```
penguins = sns.load_dataset("penguins")
sns.jointplot(data=penguins, \
        x="flipper_length_mm", \
        y="bill_length_mm", hue="species")
plt.show()
```

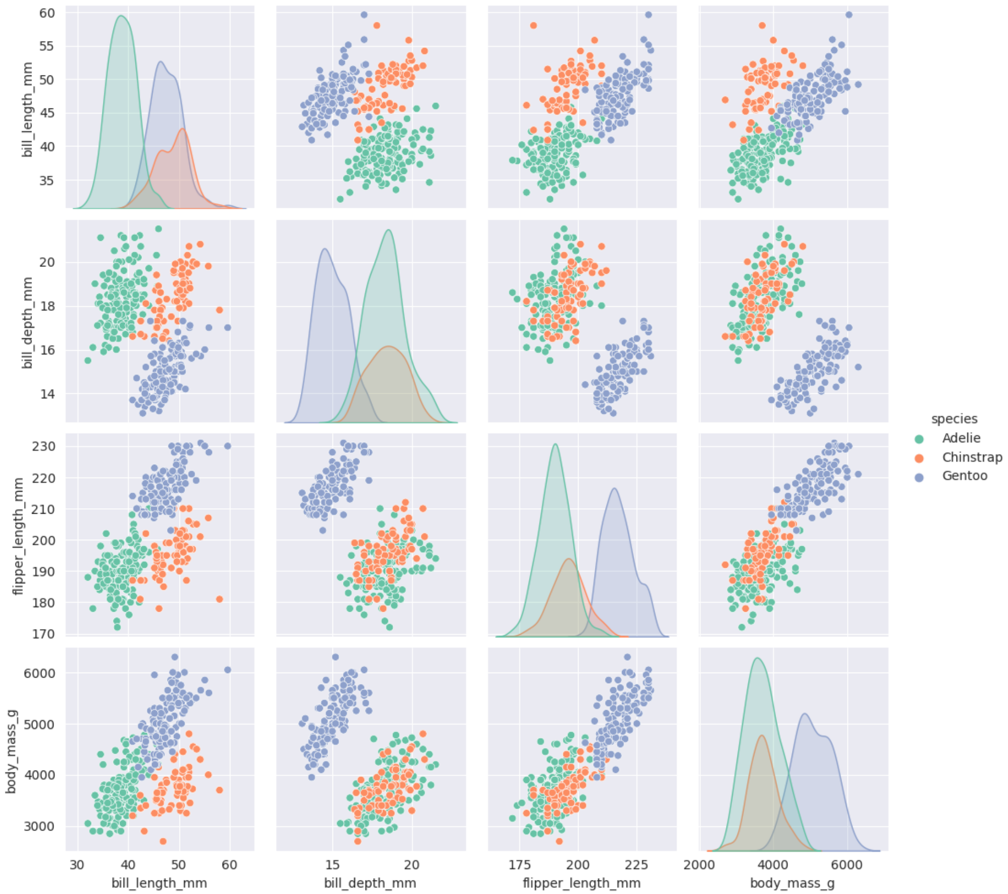


Seaborn: More graphs (2/2)

❖ Pairplot()

- shows joint and marginal distributions for all pairwise relationships and for each variable, respectively

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



Summary

- Matplotlib
  - a comprehensive library for creating static, animated, and interactive visualizations in Python

- Seaborn
  - a Python data visualization library based on matplotlib
    - provides a high-level interface for drawing attractive and informative statistical graphics

Q&A

Prof. Jeon, Sangryul

Computer Vision Lab.

Pusan National University, Korea

Tel: +82-51-510-2423

Web: http://sr-jeon.github.io/

E-mail: srjeonn@pusan.ac.kr