# Exploratory Data Analysis & Visualization

Data Mining

Prof. Sujee Lee

Department of Systems Management Engineering

Sungkyunkwan University

# EDA

# Exploratory Data Analysis

- **What is EDA?**
  - The analysis of datasets based on various numerical methods and graphical tools.
  - Exploring data for patterns, trends, underlying structure, deviations from the trend, anomalies and strange structures.
  - It facilitates discovering unexpected as well as conforming the expected.
  - (Mostly) Techniques for summarizing and visualizing data.

# Summary (or Descriptive) Statistics

## Measure of Location

- The **mean** is the most common measure of the location of a set of points.

$$mean(x) = \bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

- However, the mean is very sensitive to outliers.
- Thus, the **median** or a **trimmed mean** is also commonly used.

$$median(x) = \begin{cases} x_{(r+1)} & \text{if } m \text{ is odd, i.e., } m = 2r+1 \\ (x_{(r)} + x_{(r+1)})/2 & \text{if } m \text{ is even, i.e., } m = 2r \end{cases}$$

$$\alpha\, trimmed\, mean(x) = \frac{1}{m - 2\alpha m}\sum_{i=\alpha m}^{m-\alpha m} x_i$$

- The **mode** is the most frequently occurring value in the set.
  - Useful for discrete variables

$$mode(\{4, 5, 12, 6, 5, 5, 8, 9, 5, 1\}) = 5$$

# Summary Statistics (cont.)

- **Measure of Dispersion**
  - **range** is the difference between the max and min.

$$range(x) = max(x) - min(x)$$

  - The **variance** or **standard deviation** is the most common measure of the spread of a set of points.

$$var(x) = s^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2$$

  - However, this is also sensitive to outliers, so other measures are often used.
  - **average absolute deviation**

$$AAD(x) = \frac{1}{m}\sum_{i=1}^{m}|x_i - \bar{x}|$$

  - **median absolute deviation**

$$MAD(x) = median(\{|x_1 - \bar{x}|, |x_2 - \bar{x}|, \ldots, |x_m - \bar{x}|\})$$

# Summary Statistics (cont.)

- **Measure of Association**
  - **Covariance**

$$Cov(X_j, X_k) = E[(X_j - E[X_j])(X_k - E[X_k])]$$
$$= \frac{1}{m-1}\sum_{i=1}^{m}(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$$

  - **Correlation coefficients**
    - **Pearson correlation**

$$Corr(X_j, X_k) = \frac{Cov(X_j, X_k)}{\sqrt{Var(X_j) \cdot Var(X_k)}}$$

    - **Spearman rank correlation**

$$Corr(X_j, X_k) = 1 - \frac{6}{m(m^2-1)}\sum_{i=1}^{m}\Big(rank(x_{ij}) - rank(x_{ik})\Big)^2$$

    - Many other measures for discrete variables

# Practice Using the Iris Data

- **Iris Data**
  - Many of the exploratory data techniques are illustrated with the Iris Plant data set.
  - Three flower types (classes):
    - Setosa
    - Virginica
    - Versicolour
  - Four (non-class) attributes
    - Sepal width and length
    - Petal width and length
  - 50 instances for each class

# Practice Using the Iris Data

- **Load the dataset**

```
import seaborn as sns
df_iris = sns.load_dataset('iris')
df_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 |

# Practice Using the Iris Data

- **Summary Statistics**

```
df_iris.describe()
```

| | sepal_length | sepal_width | petal_length | petal_width |
|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.057333 | 3.758000 | 1.199333 |
| **std** | 0.828066 | 0.435866 | 1.765298 | 0.762238 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

# Practice Using the Iris Data

- **Summary Statistics**

```
df_iris['sepal_length'].mean()
```

```
5.843333333333334
```

```
import numpy as np
np.mean(df_iris['sepal_length'])
```

```
5.843333333333334
```

# Practice Using the Iris Data

- **Summary Statistics**
  - Percentile

```
df_iris['sepal_length'].quantile(0.95)
```

```
7.254999999999998
```

  - Frequency

```
df_iris['species'].value_counts()
```

```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

# Practice Using the Iris Data

- Sur

```
df_iris['sepal_length'].var()
```

```
0.6856935123042505
```

```
df_iris.iloc[:-1,:-1].cov()
```

| | sepal_length | sepal_width | petal_length | petal_width |
|---|---|---|---|---|
| **sepal_length** | 0.690305 | -0.042699 | 1.282408 | 0.519527 |
| **sepal_width** | -0.042699 | 0.191241 | -0.331360 | -0.122227 |
| **petal_length** | 1.282408 | -0.331360 | 3.125083 | 1.298880 |
| **petal_width** | 0.519527 | -0.122227 | 1.298880 | 0.582478 |

```
np.cov(df_iris.iloc[:-1,:-1])
```

```
array([[4.75      , 4.42166667, 4.35333333, ..., 2.74166667, 2.915     ,
        2.475     ],
       [4.42166667, 4.14916667, 4.055     , ..., 2.82083333, 2.95583333,
        2.50416667],
       [4.35333333, 4.055     , 3.99      , ..., 2.53166667, 2.68833333,
        2.28166667],
       ...,
```

# Practice Using the Iris Data

- **Stratification**
  - When data is divided into several groups with different characteristics, the values in each dimension can also have different representative values and distributions for each group.
  - Stratification: A technique for dividing data into homogeneous groups for analysis.

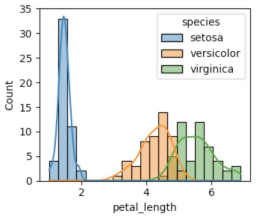
```
df_iris.groupby('species').agg(['mean','std'])
```

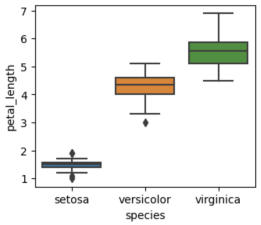
| | sepal_length | | sepal_width | | petal_length | | petal_width | |
|---|---|---|---|---|---|---|---|---|
| | mean | std | mean | std | mean | std | mean | std |
| species | | | | | | | | |
| setosa | 5.006 | 0.352490 | 3.428 | 0.379064 | 1.462 | 0.173664 | 0.246 | 0.105386 |
| versicolor | 5.936 | 0.516171 | 2.770 | 0.313798 | 4.260 | 0.469911 | 1.326 | 0.197753 |
| virginica | 6.588 | 0.635880 | 2.974 | 0.322497 | 5.552 | 0.551895 | 2.026 | 0.274650 |

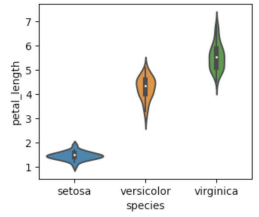
# Data Visualization (Plots)

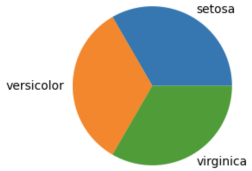
# Data Visualization

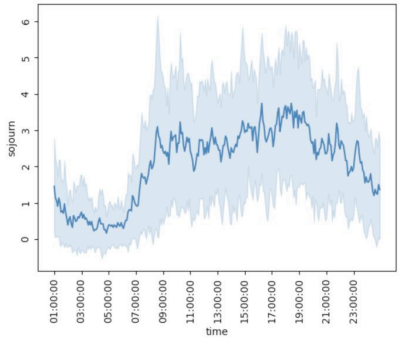
- **Plots in Python**
  - **histogram** (pandas/plt, sns)
  - **box plot** (pandas/plt, sns) / violin plot (plt, sns)
  - **errorbar** (plt), **pointplot** (sns)
  - **pie** (pandas/plt)
  - **Time-series visualization**
    - pointplot (sns), lineplot (sns), seasonal decompose, cumulative
  - **Multivariate visualization**
    - scatterplot, histplot, jointplot, pairplot, heatmap (sns)

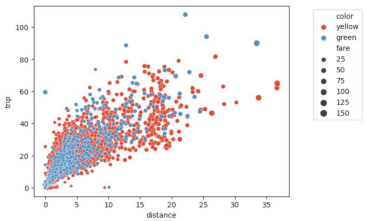


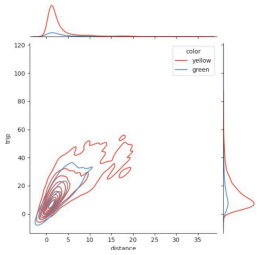



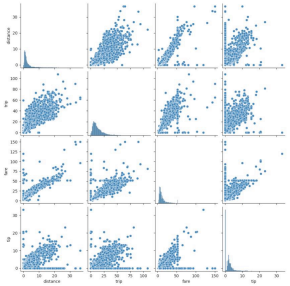



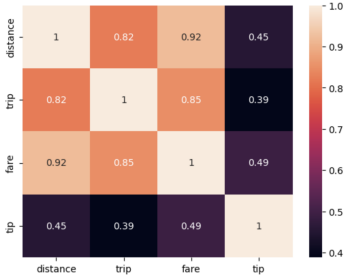



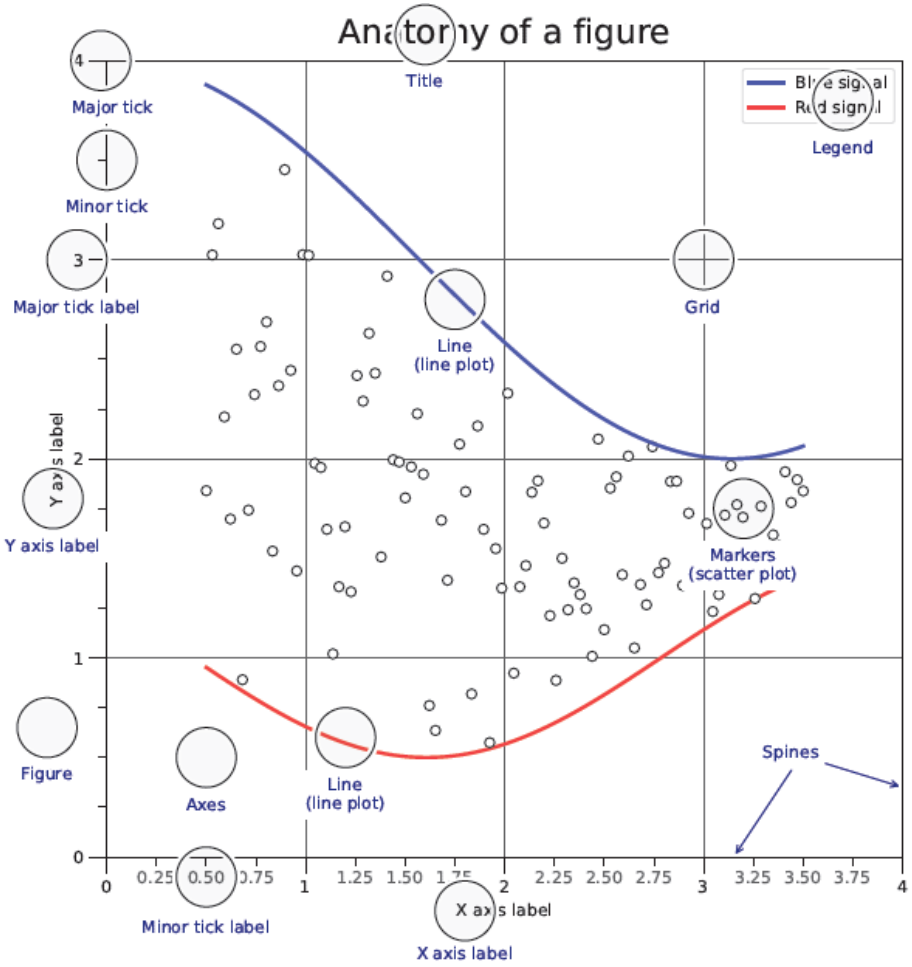





# Matplotlib

- **Matplotlib**
  - One of the most widely used libraries for plotting graphs in Python.
  - Supports various types of graphs:
    - Line graphs, scatter plots, bar charts, histograms, pie charts, and more.
  - Includes various elements needed for creating graphs:
    - Graph titles, labels, legends, axes, and colors.

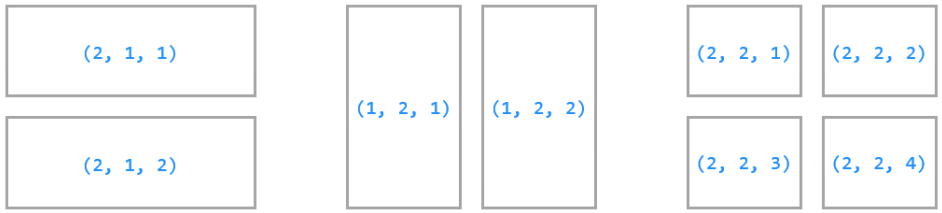
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt


X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X, Y, color='green')

fig.savefig("figure.pdf")
fig.show()
```



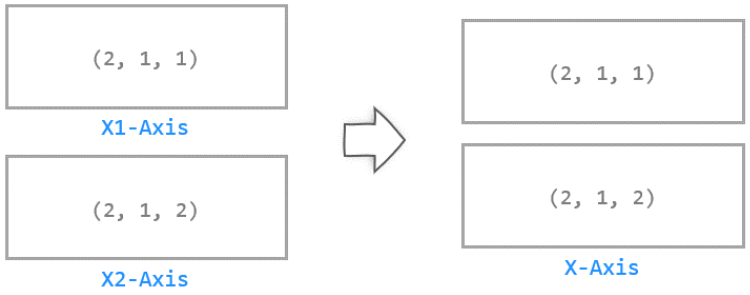

https://matplotlib.org/stable/index.html
https://wikidocs.net/book/5011

# Matplotlib

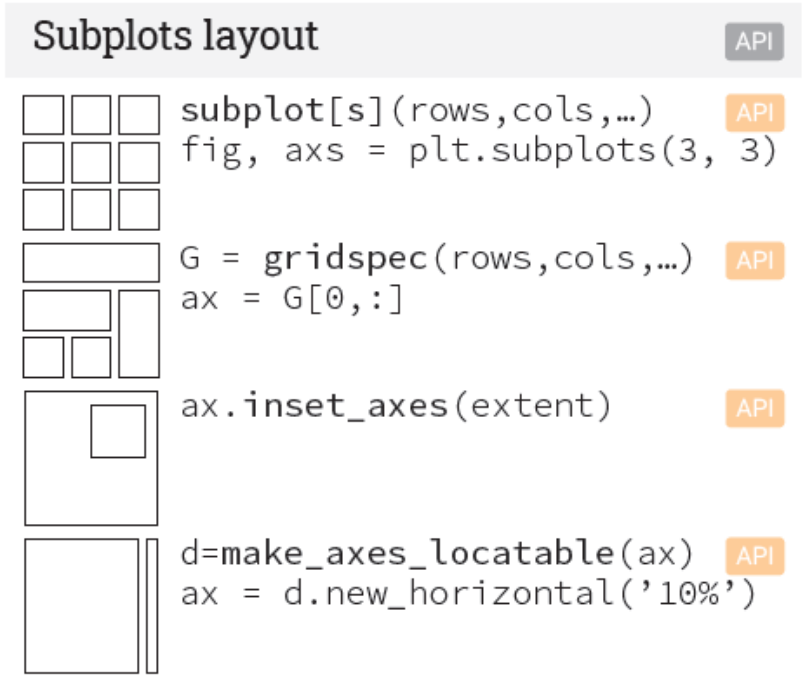
- **Comparing Plots (subplot)**
  - Allows for the comparison of multiple graphs simultaneously




```
plt.subplot(row, column, index)
```




```
plt.subplot(2, 1, 2, sharex=ax1)
```

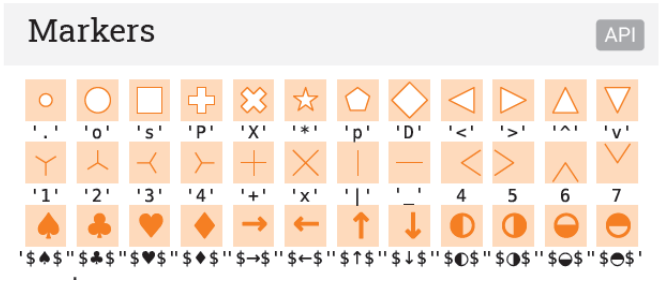


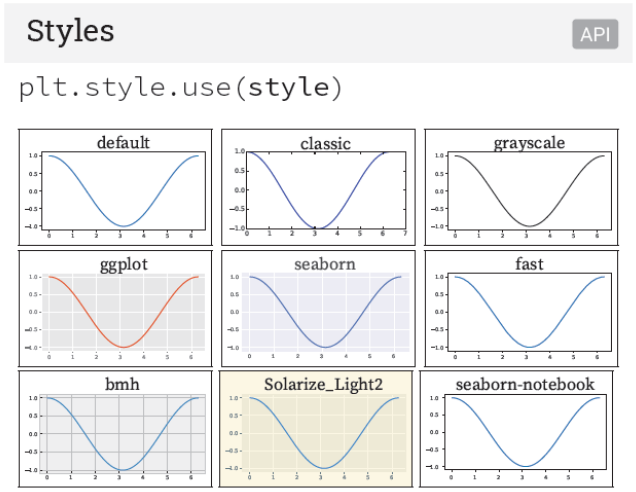
# Matplotlib

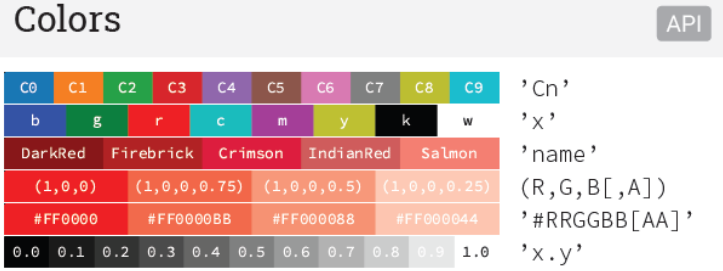
- **Plots**

# Matplotlib

- **Settings (Style, Color)**


# Matplotlib

- **Tips**

### Quick reminder

```
ax.grid()
ax.set_[xy]lim(vmin, vmax)
ax.set_[xy]label(label)
ax.set_[xy]ticks(ticks, [labels])
ax.set_[xy]ticklabels(labels)
ax.set_title(title)
ax.tick_params(width=10, …)
ax.set_axis_[on|off]()

fig.suptitle(title)
fig.tight_layout()
plt.gcf(), plt.gca()
mpl.rc('axes', linewidth=1, …)
[fig|ax].patch.set_alpha(0)
text=r'$\frac{-e^{i\pi}}{2^n}$'
```

### How do I ...

**... resize a figure?**
→ fig.set_size_inches(w, h)
**... save a figure?**
→ fig.savefig("figure.pdf")
**... save a transparent figure?**
→ fig.savefig("figure.pdf", transparent=True)
**... clear a figure/an axes?**
→ fig.clear() → ax.clear()
**... close all figures?**
→ plt.close("all")
**... remove ticks?**
→ ax.set_[xy]ticks([])
**... remove tick labels ?**
→ ax.set_[xy]ticklabels([])
**... rotate tick labels ?**
→ ax.tick_params(axis="x", rotation=90)
**... hide top spine?**
→ ax.spines['top'].set_visible(False)
**... hide legend border?**
→ ax.legend(frameon=False)

**... show error as shaded region?**
→ ax.fill_between(X, Y+error, Y-error)
**... draw a rectangle?**
→ ax.add_patch(plt.Rectangle((0, 0), 1, 1)
**... draw a vertical line?**
→ ax.axvline(x=0.5)
**... draw outside frame?**
→ ax.plot(..., clip_on=False)
**... use transparency?**
→ ax.plot(..., alpha=0.25)
**... convert an RGB image into a gray image?**
→ gray = 0.2989*R + 0.5870*G + 0.1140*B
**... set figure background color?**
→ fig.patch.set_facecolor("grey")
**... get a reversed colormap?**
→ plt.get_cmap("viridis_r")
**... get a discrete colormap?**
→ plt.get_cmap("viridis", 10)
**... show a figure for one second?**
→ fig.show(block=False), time.sleep(1)

# Basic Plots: Line Graphs, Bar Charts, Scatterplots

- **Bar Charts:**
  - Useful for comparing groups with single statistics such as averages, counts, or ratios.
  - The height of the bars represents statistical values, each bar represents a group, and the height (or length for horizontal bars) indicates the value of the variable.
- **Scatterplots:**
  - Used to show the relationship between numerical variables.
  - Help reveal associations such as information redundancy or clustering between two numerical variables in unsupervised learning.
- **Line Graphs**
  - Primarily used to display time series data.
  - The size of the time frame for plotting the graph varies based on the scale of the prediction task and the properties of the data, similar to the time scale.

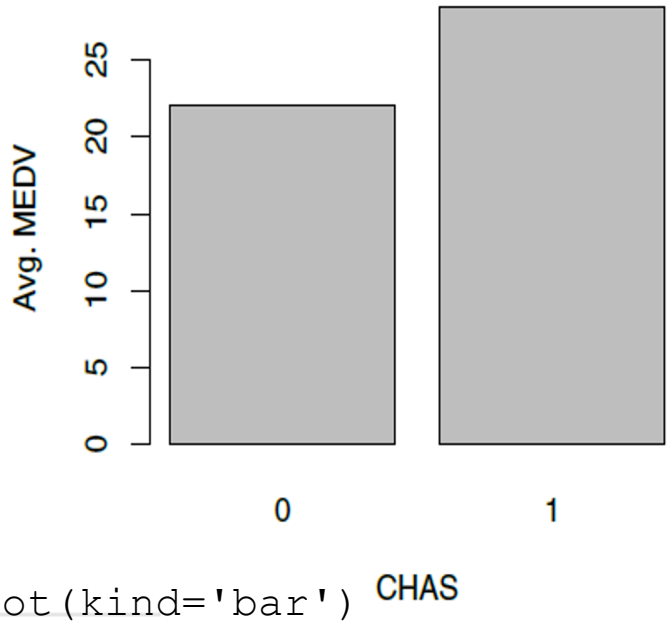
# Basic Plots: Line Graphs, Bar Charts, Scatterplots

- **Load Data to Illustrate Plots**

```
## Load, convert Amtrak data for time series analysis
Amtrak_df = pd.read_csv('Amtrak.csv', squeeze=True)
Amtrak_df['Date'] = pd.to_datetime(Amtrak_df.Month,
    format='%d/%m/%Y')
ridership_ts = pd.Series(Amtrak_df.Ridership.values,
    index=Amtrak_df.Date)

## Boston housing data
housing_df = pd.read_csv('BostonHousing.csv')
housing_df = housing_df.rename(columns={'CAT. MEDV': 'CAT_MEDV'})
```

# Basic Plots: Line Graphs, Bar Charts, Scatterplots

- **Bar Chart for Categorical Variable**
  - Average median neighborhood value for neighborhoods that do and do not border the Charles River




Using `pandas`:

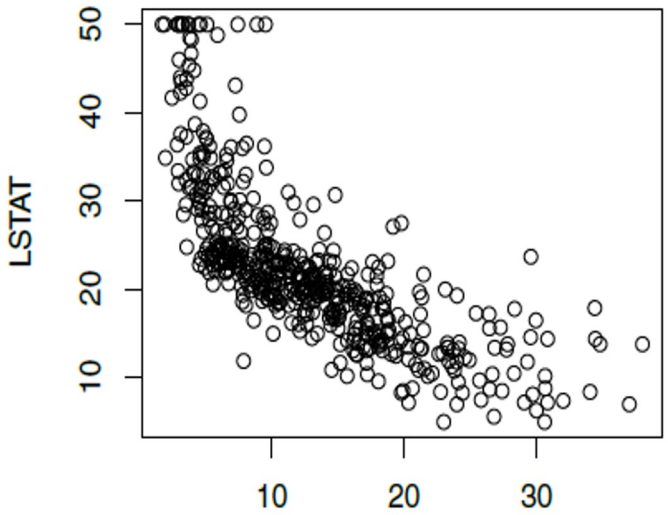
```
# compute mean MEDV per CHAS = (0, 1)
ax = housing_df.groupby('CHAS').mean().MEDV.plot(kind='bar')
ax.set_ylabel('Avg. MEDV')
```

Using `matplotlib`:

```
# compute mean MEDV per CHAS = (0, 1)
dataForPlot = housing_df.groupby('CHAS').mean().MEDV
fig, ax = plt.subplots()
ax.bar(dataForPlot.index, dataForPlot, color=['C5', 'C1'])
ax.set_xticks([0, 1], False)
ax.set_xlabel('CHAS')
ax.set_ylabel('Avg. MEDV')
```

# Basic Plots: Line Graphs, Bar Charts, Scatterplots

- **Scatterplot**
  - Displays relationship between two numerical variables




Using `pandas`:

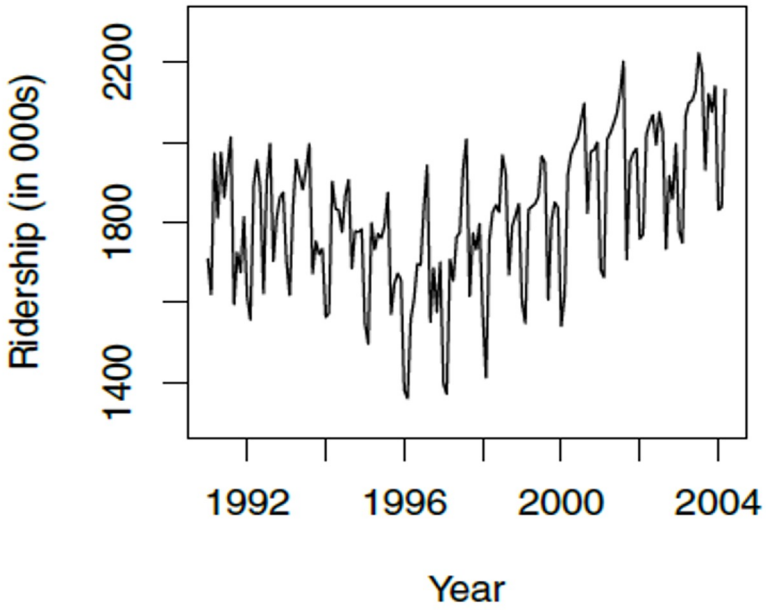
```
## scatter plot with axes names
housing_df.plot.scatter(x='LSTAT', y='MEDV', legend=False)
```

Using `matplotlib`:

```
## Set the color of points and draw as open circles.
plt.scatter(housing_df.LSTAT, housing_df.MEDV, color='C2',
facecolor='none')
plt.xlabel('LSTAT'); plt.ylabel('MEDV')
```

# Basic Plots: Line Graphs, Bar Char

- **Line Plot for Time Series**
  - Amtrak Ridership




Using `pandas`:

```
ridership_ts.plot(ylim=[1300, 2300], legend=False)
plt.xlabel('Year')  # set x-axis label
plt.ylabel('Ridership (in 000s)')  # set y-axis label
```

Using `matplotlib`:

```
plt.plot(ridership_ts.index, ridership_ts)
plt.xlabel('Year')  # set x-axis label
plt.ylabel('Ridership (in 000s)')  # set y-axis label
```

# Distribution Plots: Histograms and Box Plots

- **Distribution Plots:**
  - Display the overall distribution of a numerical variable.
  - Useful for determining data mining methods and variable transformations.

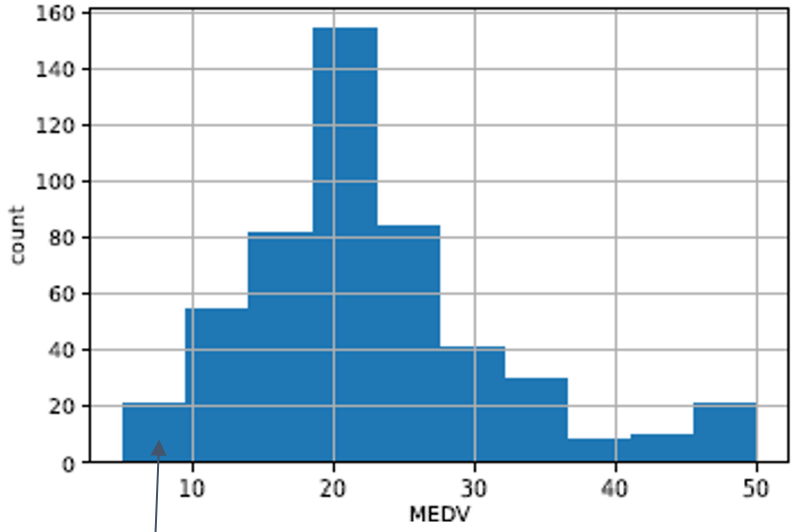
- **Histograms:**
  - Represent the frequency of all x-values using a series of vertically connected bars.

- **Box Plots:**
  - Can be used to compare subgroups side by side or to observe distribution changes over time by generating multiple box plots over different time periods.

# Histograms

- **Histograms**
  - Histogram shows the distribution of the outcome variable (median house value)

Boston Housing example:

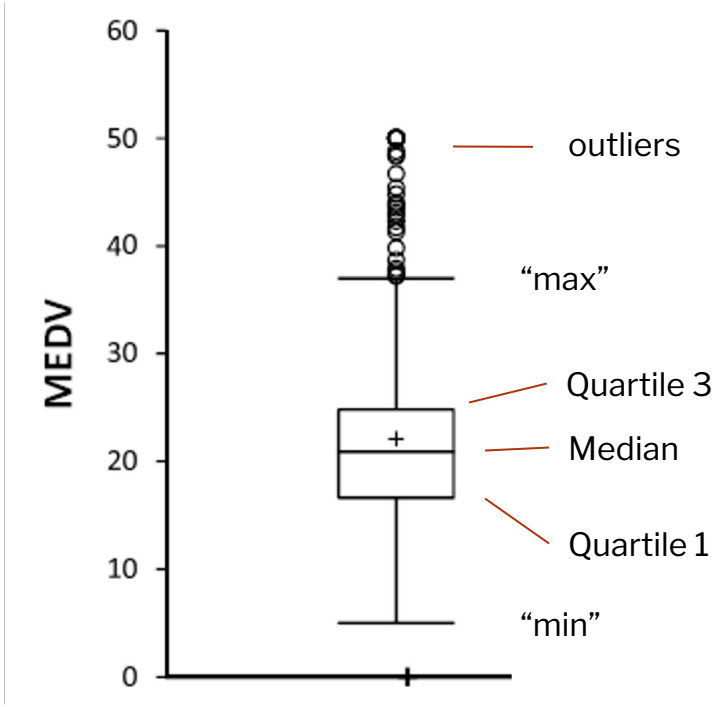



About 20 neighborhoods had a median house value in the lowest bin, about $4000K to $9500 (these data are from mid-20th century)

```
## histogram of MEDV
ax = housing_df.MEDV.hist()
ax.set_xlabel('MEDV'); ax.set_ylabel('count')
```

# Distribution Plots: Histograms and Box Plots

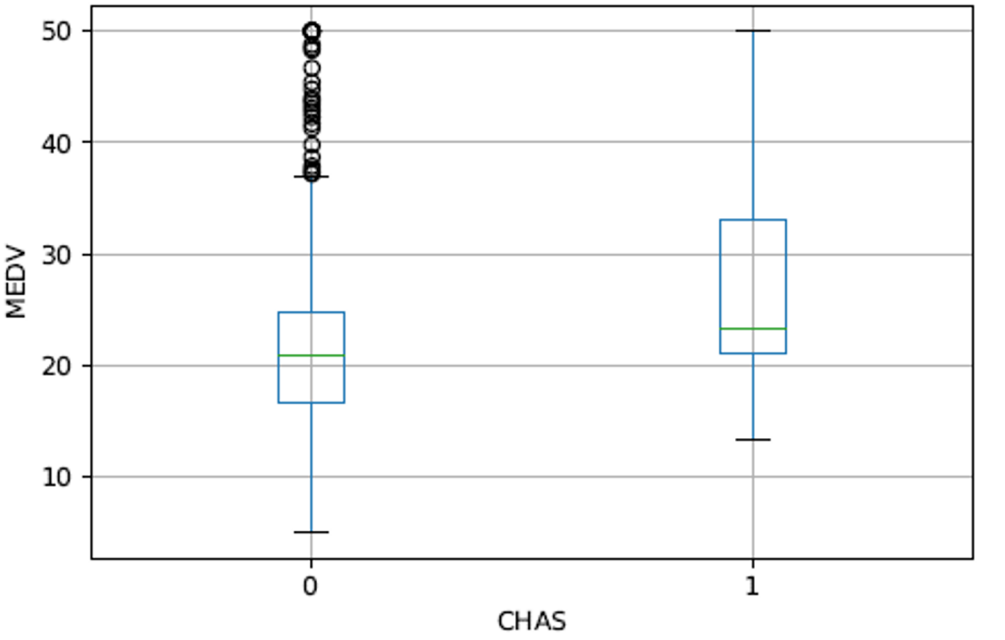
- **Box Plots**
  - Another way of displaying the distribution of data
  - Following figure shows the basic part of a box plot




- Top outliers defined as those above Q3+1.5(Q3-Q1).
- "max" = maximum of non-outliers
- Analogous definitions for bottom outliers and for "min"
- Details may differ across software

# Distribution Plots: Histograms and Box Plots

- **Boxplots**
  - Side-by-side boxplots are useful for comparing subgroups




Houses in neighborhoods on Charles river (1) are more valuable than those not (0)

```
ax = housing_df.boxplot(column='MEDV', by='CHAS')
ax.set_ylabel('MEDV')
plt.suptitle('')  # Suppress the titles
plt.title('')
```

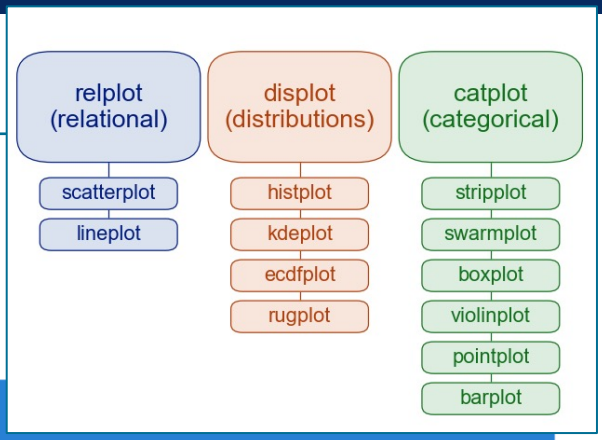
# Plotting with Seaborn

# Seaborn

- **Seaborn**
  - A data visualization library in Python built on top of Matplotlib.
  - Seaborn can generate high-quality graphs with simpler code than Matplotlib and offers various color themes and statistical charts.
    - Provides a variety of color themes.
    - Supports a wide range of graph types.
    - Creates graphs with simpler code compared to Matplotlib.
  - https://seaborn.pydata.org/
  - https://seaborn.pydata.org/examples/index.html
  - https://seaborn.pydata.org/tutorial/properties.html

# Plotting Strategies

- **Plots**

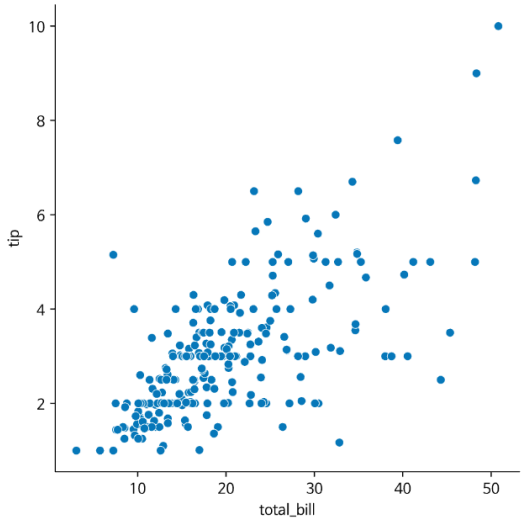



| | 그래프 종류 (대분류) | 그래프 종류 (소분류) | 설명 |
|---|---|---|---|
| 1 | Relational plots | relplot, scatterplot, lineplot | 두가지 변수(x, y)의 관계를 나타내기 위한 그래프 |
| 2 | Distribution plots | displot, hisplot, kdeplot, ecdfplot, rugplot, distplot | 변수 하나(x or y) 혹은 변수 두개 (x,y)의 값 분포를 나타내기 위한 그래프 |
| 3 | Categorical plots | catplot, stripplot, swarmplot, boxplot, violinplot, boxenplot, pointplot, barplot, countplot | 범주형 변수 (ex. Male/Female, Yes/No)와 연속형 변수(숫자) 간의 관계를 나타내기 위한 그래프 |
| 4 | Regression plots | Implot, regplot, residplot | 회귀(regression) 분석 결과를 relational p lots과 함께 나타내주는 그래프 |
| 5 | Matrix plots | heatmap, clustermap | 연속형 변수(숫자) 간의 관계 비율을 2차원 메트릭스로 만들고 그 비율에 따라 색을 입혀서 시각화 |
| 6 | Multi-plot grids | FacetGrid, pairplot, PairGrid, jointplot, JointGrid | 여러 그래프를 함께 그려 한눈에 비교하기 위한 그래프 |

# Relational Plot

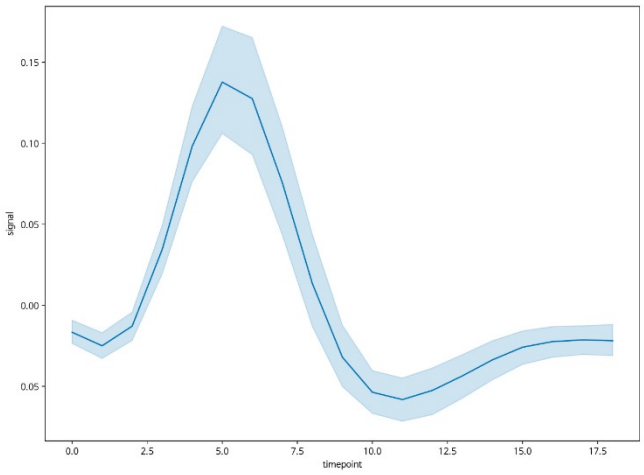
- **Relational Plot**
  - A graph used to show the **relationship** between two variables (continuous variables).
  - Helps to identify linear or non-linear relationships between two variables.
  - Types: scatter plot, lineplot, relplot.

### Scatterplot



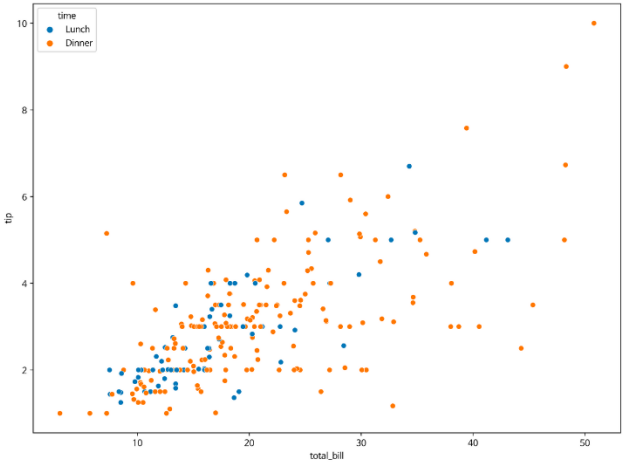

Examine the correlation in a set of continuous data.

### Line plot




Displayed continuously with confidence intervals included.

### Relation Plot

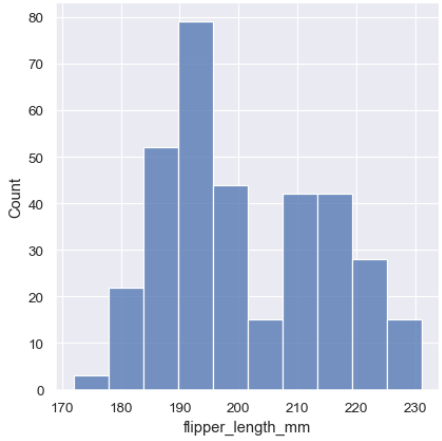



Multiple graphs can be drawn at once, and it combines the features of lineplot and scatterplot.

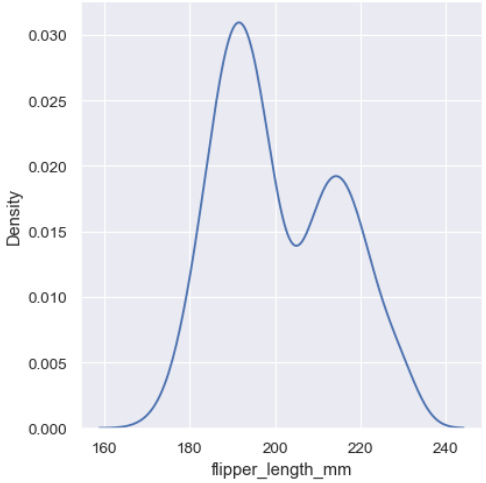
# Distribution Plot

- **Distribution Plot**
  - Graphs to show the distribution of a variable.
  - Helps to understand the shape of the distribution and detect outliers.
  - Can be defined directly using “sns.function_name” or in the form of “sns.displot(kind=‘function_name’)”.
  - Types: histplot, kdeplot, rugplot.

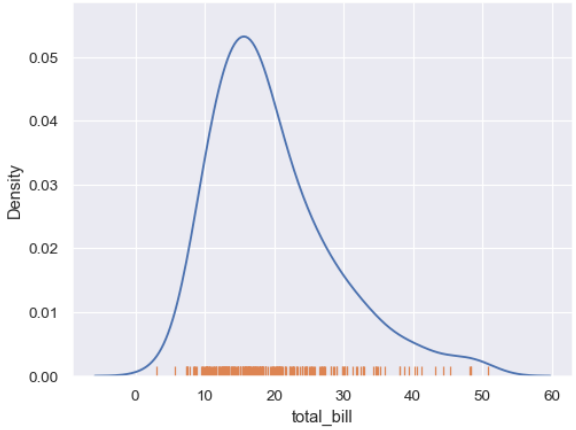
**Histogram**




**KDE plot**




**Rugplot**



# Categorical Plot

- **Categorical Plot**
  - Represents the relationship between categorical and continuous variables.
  - Helps to observe **data changes** in categorical variables.
  - Types: **catplot**, stripplot, swarmplot, boxplot, violinplot, boxedplot, pointplot, barplot, stem plot, countplot.

Categorical scatterplots:

- `stripplot()` (with `kind="strip"`; the default)
- `swarmplot()` (with `kind="swarm"`)

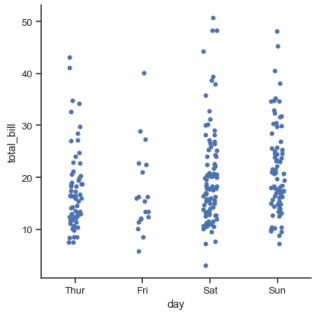
Categorical distribution plots:

- `boxplot()` (with `kind="box"`)
- `violinplot()` (with `kind="violin"`)
- `boxenplot()` (with `kind="boxen"`)

Categorical estimate plots:

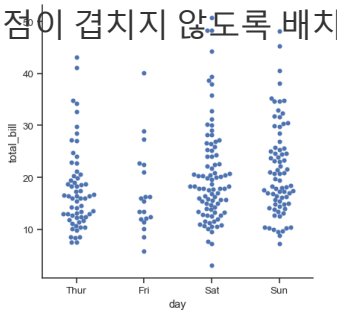
- `pointplot()` (with `kind="point"`)
- `barplot()` (with `kind="bar"`)
- `countplot()` (with `kind="count"`)

**Scatterplot**

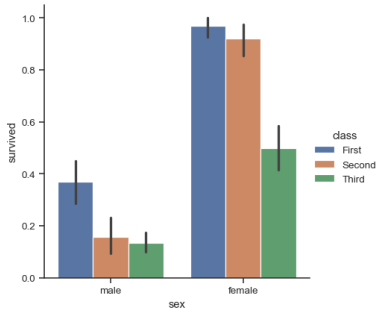



**Swarm plot**

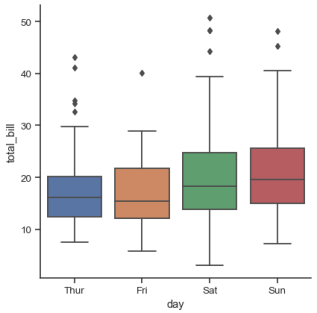
(점이 겹치지 않도록 배치)



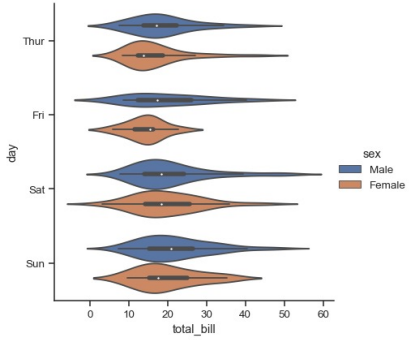

**Bar plot**



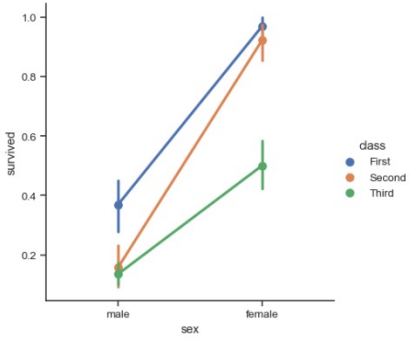

**Box plot**




**Violin plot**



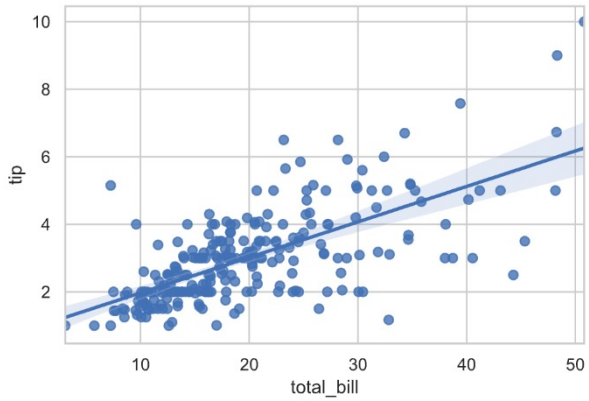

**Point plot**



# Regression Plot

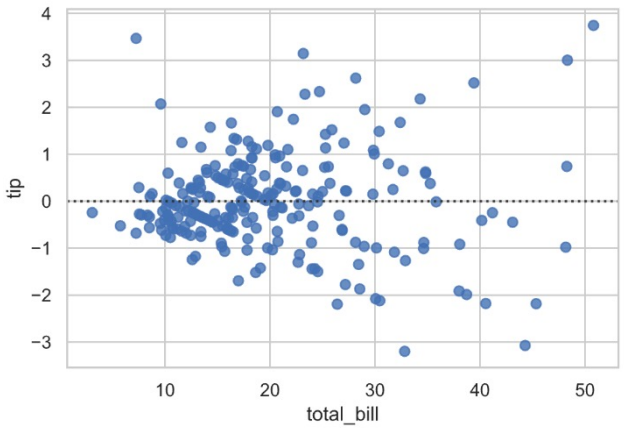
- **Regression Plot**
  - Graph visualizing the results of **regression analysis**, making it easier to understand the outcomes of the analysis.
  - Types: lmplot, regplot, residplot.

**Regplot**



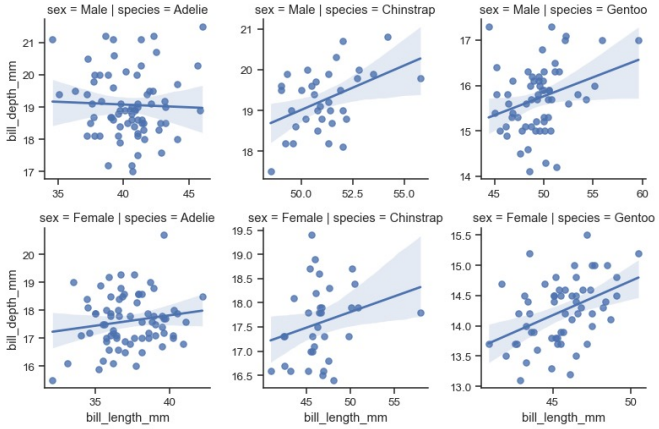

Examine the correlation in a set of continuous data.

**Residplot**




Show the errors of the data points based on the regression line.

**lmplot**

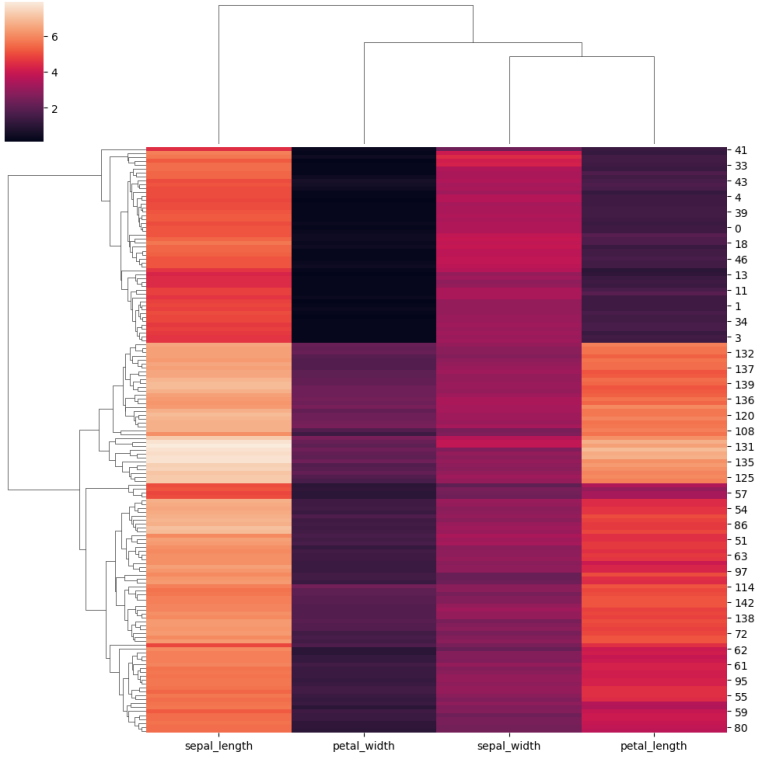



Observe multiple regplots depending on the variables.

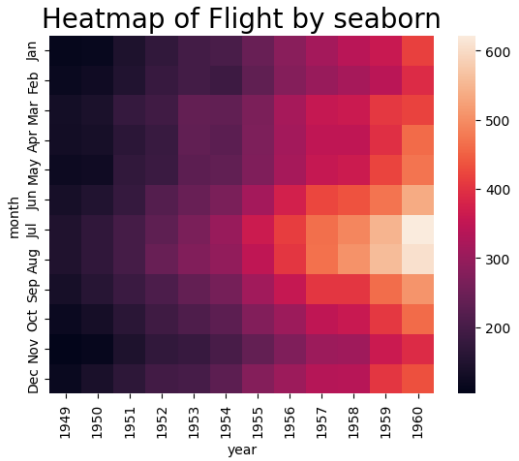
# Matrix Plot

- **Matrix Plot**
  - A type of graph used to display multidimensional data.
  - Helps to understand relationships and patterns among multidimensional data.
  - Types: heatmap, clustermap.

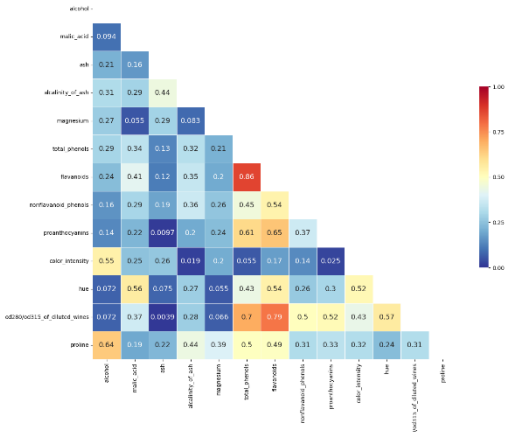
## ClusterMap




## Heatmap



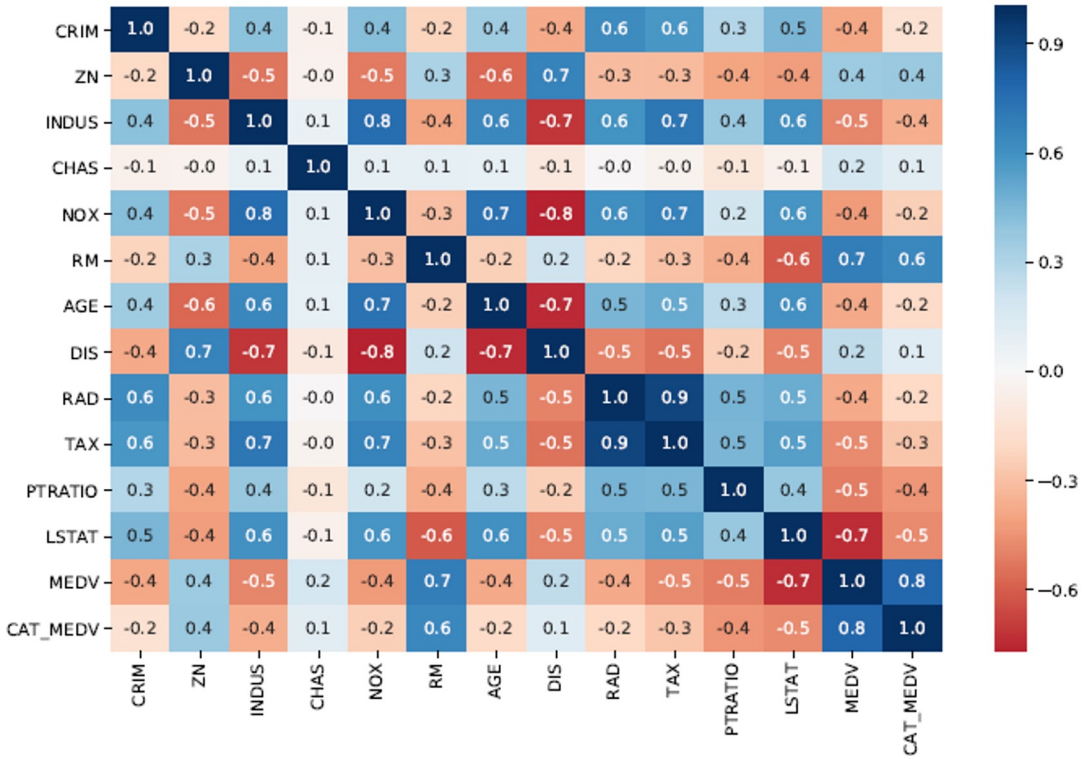

Heatmap showing changes in values according to variables



Heatmap representing the correlation between variables

# Matrix Plot

- **Heat Maps**
  - Color conveys information
  - In data mining, used to visualize
    - Correlations
    - Missing Data




Darker and bluer = stronger positive correlation
Darker & redder = stronger negative correlation

# Matrix Plot

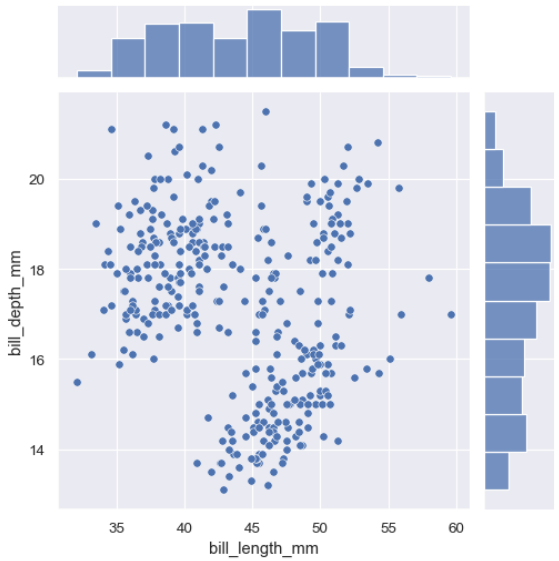
- **Heat Maps**

```
## simple heatmap of correlations (without values)
corr = housing_df.corr()
sns.heatmap(corr, xticklabels=corr.columns,
  yticklabels=corr.columns)

# Change to divergent scale and fix the range
sns.heatmap(corr, xticklabels=corr.columns,
  yticklabels=corr.columns, vmin=-1, vmax=1, cmap="RdBu")

# Include information about values (example demonstrates how to
  control the size of the plot)
fig, ax = plt.subplots()
fig.set_size_inches(11, 7)
sns.heatmap(corr, annot=True, fmt=".1f", cmap="RdBu", center=0,
  ax=ax)
```

# Multi-Plot Grid

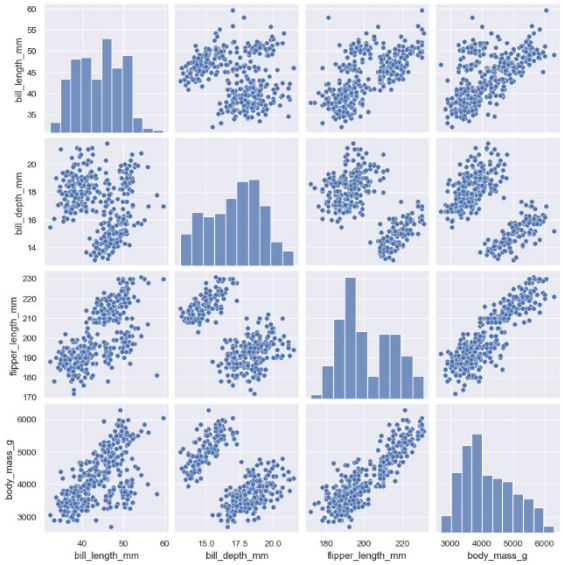
- **Multi-Plot Grid**
  - Allows viewing multiple graphs at once.
  - Combines several graphs to understand multiple aspects of the data simultaneously.
  - Types: facetgrid, pairplot, pairgrid, jointplot, jointgrid.

**Jointplot**



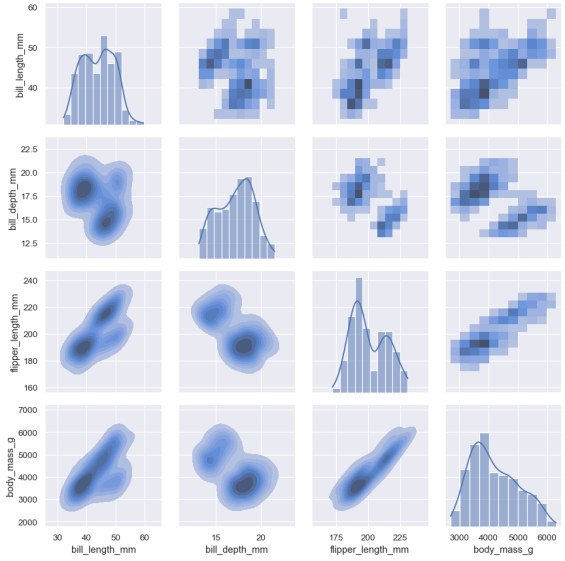

Check both the distribution and the correlation between two variables.

**Pairplot**




Examine the correlations between multiple variables.

**PairGrid**

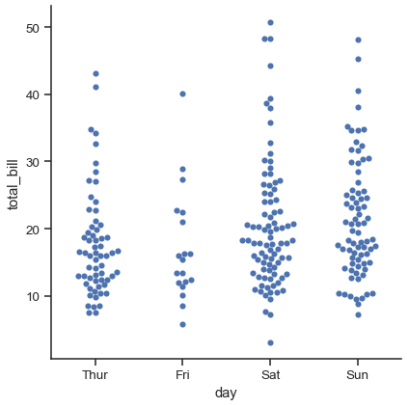


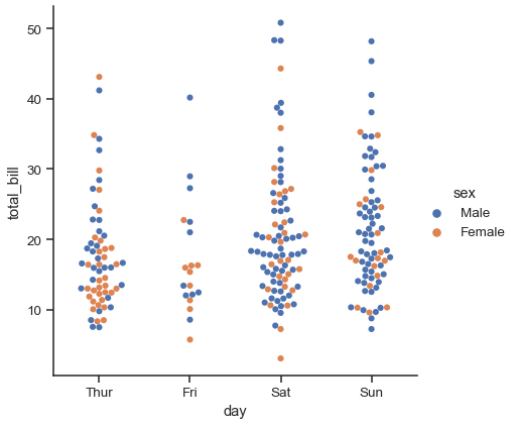

Display correlations and distributions between multiple variables using density plots, scatter plots, etc.

# Other Plots

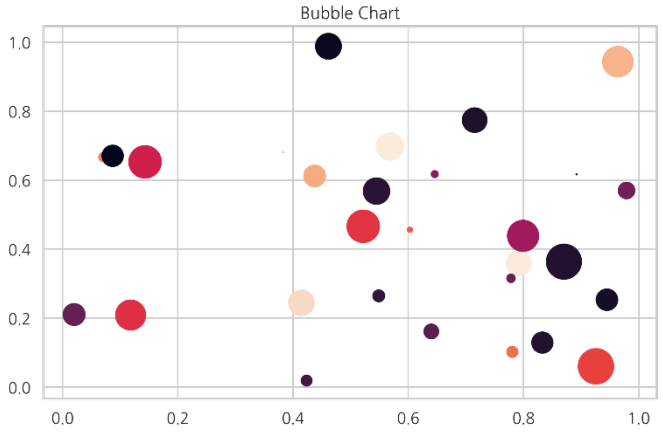
# Adding Dimension to Plots

- **Adding Dimension to plots**
  - Additional variables in data with more than two dimensions can be represented in 2D by adjusting the color or size of visualized points.



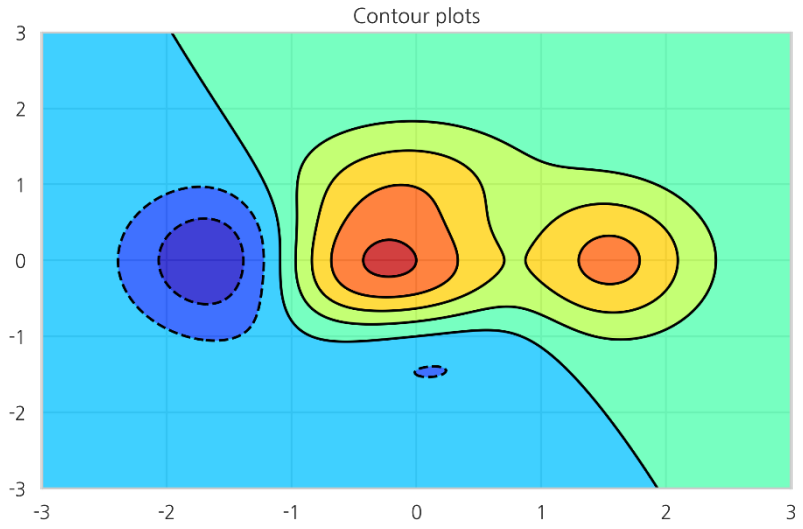




# Other Graphs

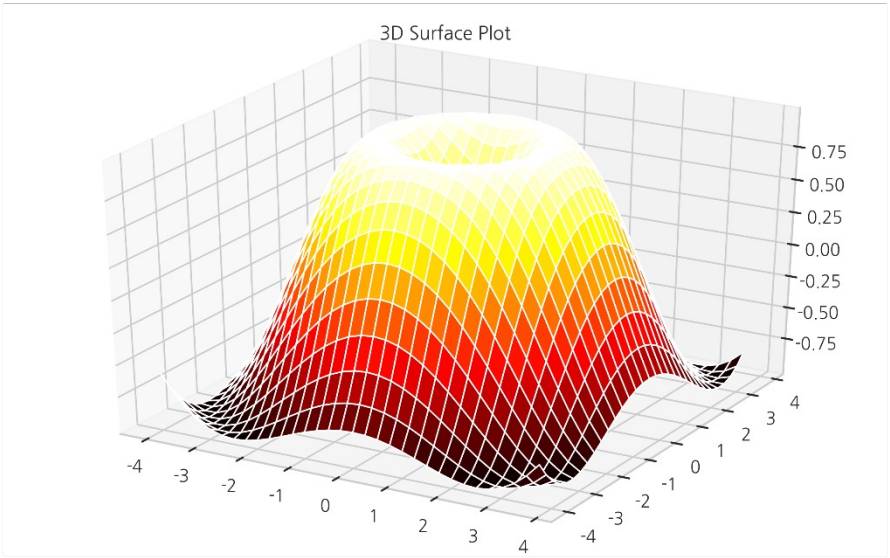
- **Multi-Dimensional Plots : Contour Plot, 3D surface plots**

## Contour Plot




Points with the same value can be connected to observe the density distribution of the data.

## 3D Surface Plot

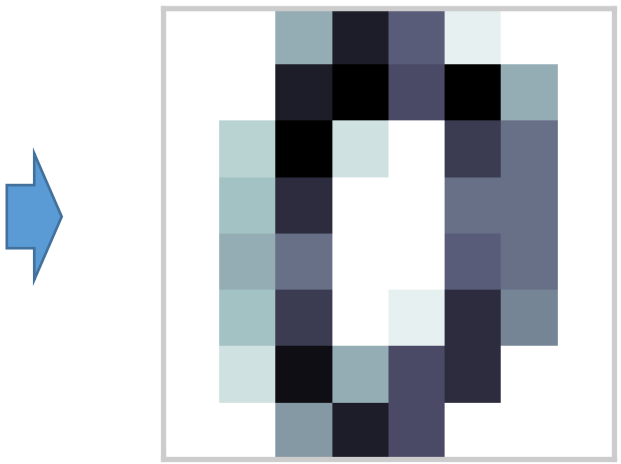



You can examine the shape, characteristics, distribution, and patterns of 3D data.

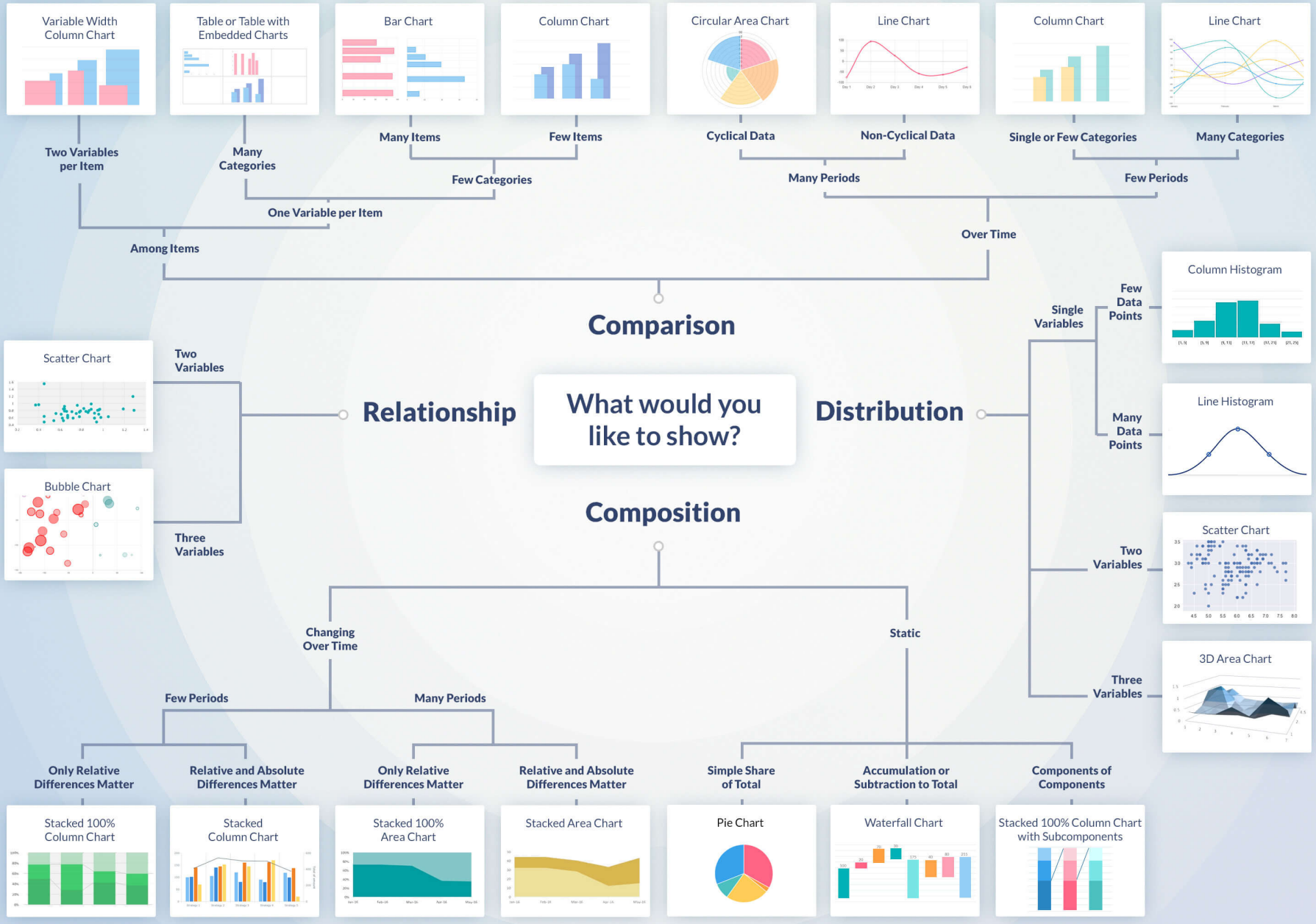
# Other Graphs

- **Image as plots : Imshow**
  - Matrix-like 2D data, such as image data, can be represented using colors to display the rows and columns.

```
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```



# Guided Visualizations for Charts and Graphs



# etc.


80%
75%
50%
25%
A
B
C
ONE
TWO
THREE
FOUR
JAN FEB MAR APR MAY JUN
JAN FEB MAR APR MAY JUN JUL