GBDi
Government Big Data Institute
สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ (สวข.)

depa

โครงการอบรมหลักสูตร Hand-on Data Science and Machine Learning

Exploratory Data Analysis (EDA) with Python

Kanyawee Pornsawangdee
Data scientist
Government Big Data Institute (GBDi)

Agreements

1. ท้ายคาบจะให้ส่งลิงค์ colab ผ่าน Form โดยจะต้องมีการทำ lab ทุก lab
   (ตั้งชื่อไฟล์ ว่า Lab-EDA-DS6502-xxx)
1. กรณีมีปัญหาให้ทักถาม TA หรือถามในห้องแชทได้เลย
2. เราจะไปกันอย่างช้าๆ รอคนที่ไม่ทันด้วยน้า :D (ถ้าใครรู้สึกว่าเริ่มไม่ทันพิมพ์ ! มาในแชทได้เลย)

Pirommas Techitnutsarut

TA ฝน

Khwansiri Sirimangkhala

TA ขวัญ

Ananwat Tippawat

TA หมู

Ravi Laohasurayodhin

TA โน้ต

Titirat Boonchuaychu

TA มิ้ว

Pre-test (10 mins)

https://forms.office.com/r/p0Rw3JjmAi

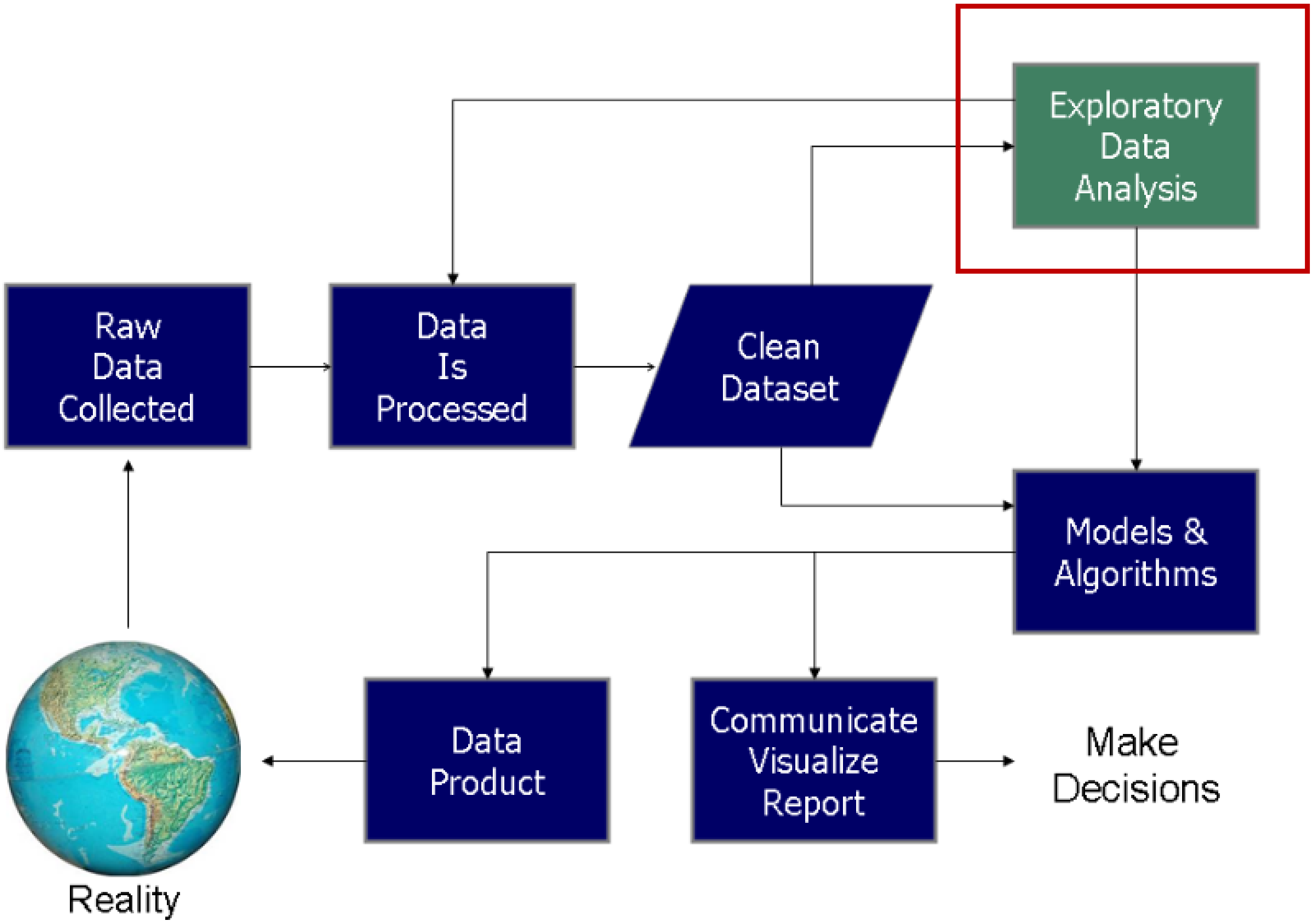
Overview

- **Learning Outcome**
  - Understand the basic concepts of exploratory data analysis
  - Understand the role of statistics in data exploration
  - Choose appropriate data analysis techniques to explore and analyze data
- **Agenda**
  - Basic concepts of exploratory data analysis (EDA)
  - EDA techniques

Agenda

9:00 – 10:30	Data Preprocessing
10:30 – 10:45	Break
10:45 – 12:00	Introduction to EDA
	EDA: Univariate Analysis
12:00 – 13:00	Lunch
13:00 – 14:30	EDA: Multivariate Analysis
14:30 – 14:45	Break
14:45 – 16:00	Workshop

Data Science Process

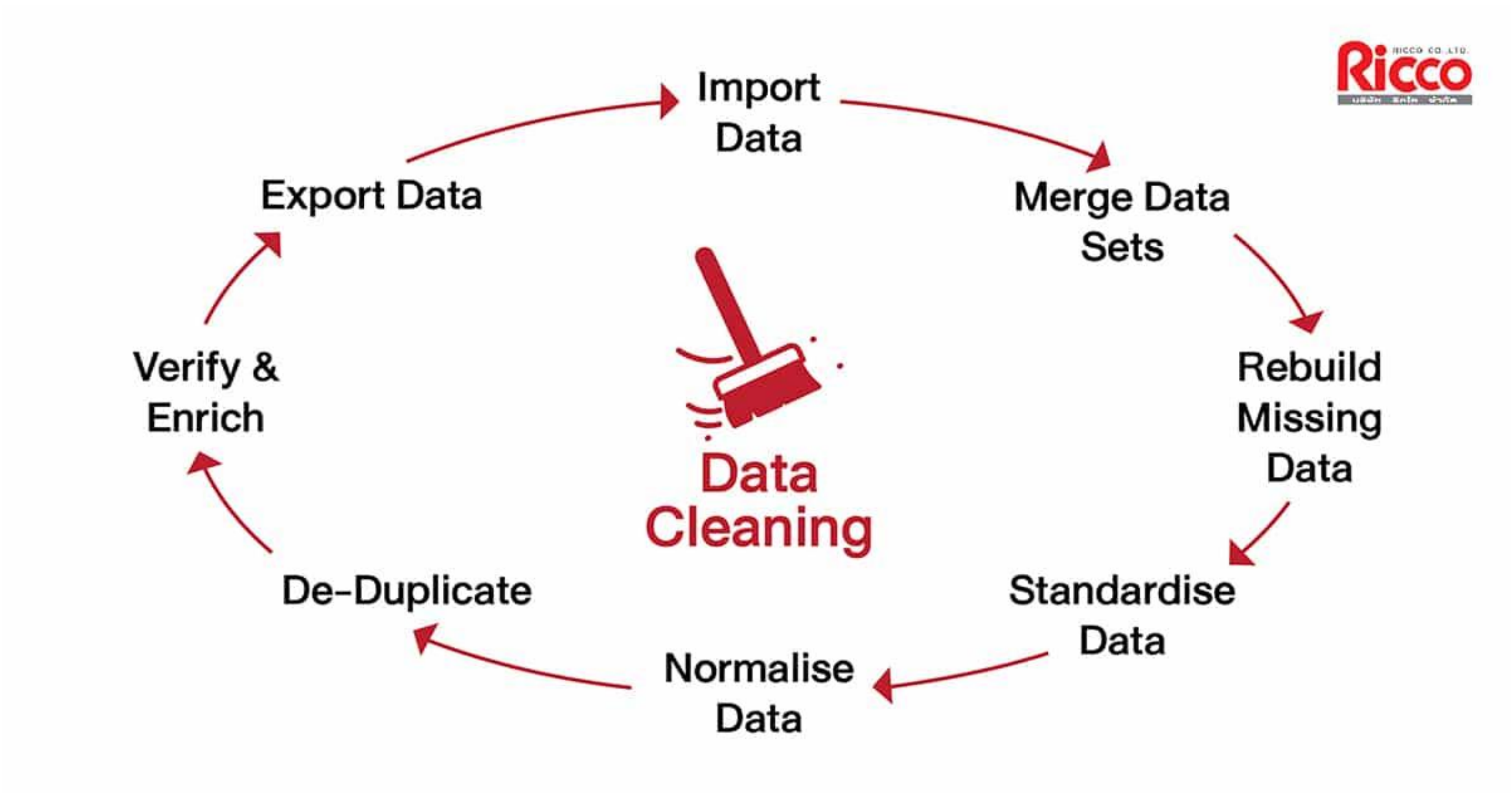


Data Preprocessing

Pre-Processing Data

- It is the process of converting or mapping data from a raw format to more manageable format for later analysis.
- Also known as “**Data Cleaning**”
- ~ **80 %** of data science process

Data Cleaning



Data Preprocess Libraries in Python

- pandas
- scikit learn
- statsmodels
- matplotlib
- seaborn

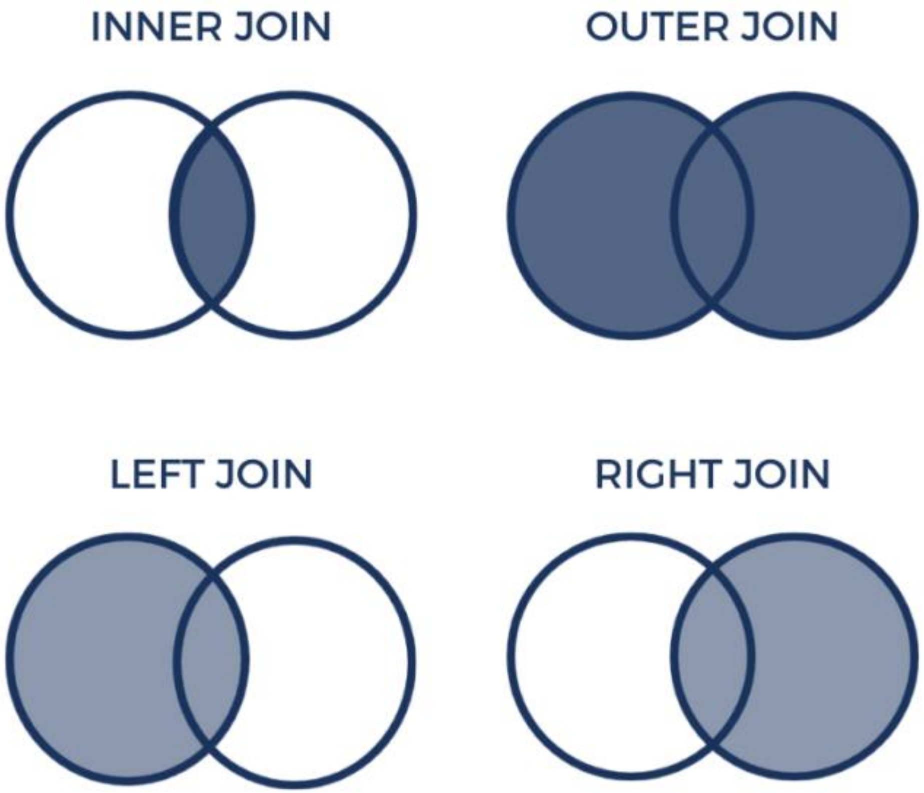
Data Preprocessing Concepts

- Combine Data
- Handling Missing Values
- Dealing with Duplicate Data
- Dealing with Outlier
- Data Formatting
- Data Scaling
- Grouping Numerical Data into Classes
- Converting Categorical Variables to Numeric Variables

Combine Data with Merge and Join

- Both join and merge can be used to **combines two dataframes** but the
- **Join** method combines two dataframes **on the basis of their indexes**
- **Merge** method is more versatile and **allows us to specify columns** beside the index to join on for both dataframes.

Join Methods



Handling missing values

- **Missing Values** is a data entry that is left empty in the data set.
- It can be reflected with “?”, Zero or a blank cell.

	Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent
0	10000	Person	Cash	120000.0	150000.0	240	Canada	America
1	10001	Company	Cash	NaN	651750.0	1043	Japan	Asia
2	10002	Company	Credit Card	451000.0	563750.0	902	Mexico	America
3	10003	Company	Transfer	565000.0	706250.0	1130	Spain	Europe
4	10004	Person	Transfer	512300.0	NaN	1024	Argentina	America

Missing Values Strategies

- Common Strategies
  - Review source data and fill in missing values.

  - Remove missing values.
    - Delete the entire variable.
    - Delete the entry with the missing data.

  - Replace missing values.
    - Replace with the average of the entire variable.
    - Replace with mode if categorical variable.
    - Replace based on collected data.

  - Stay with missing values.

Missing Values in Python

- Remove missing values in Python.

- dropna()
  - axis = 0
  - Delete whole row
  - axis = 1
  - Remove entire column

	Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent
0	10000	Person	Cash	120000.0	150000.0	240	Canada	America
1	10001	Company	Cash	NaN	651750.0	1043	Japan	Asia
2	10002	Company	Credit Card	451000.0	563750.0	902	Mexico	America
3	10003	Company	Transfer	565000.0	706250.0	1130	Spain	Europe
4	10004	Person	Transfer	512300.0	NaN	1024	Argentina	America

```
df_test.dropna(subset=["Purchases"], axis=0, inplace = True)
df_test.head()
```

	Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent
0	10000	Person	Cash	120000.0	150000.0	240	Canada	America
2	10002	Company	Credit Card	451000.0	563750.0	902	Mexico	America
3	10003	Company	Transfer	565000.0	706250.0	1130	Spain	Europe
4	10004	Person	Transfer	512300.0	NaN	1024	Argentina	America
5	10005	Person	Transfer	415500.0	519375.0	0	Canada	America

Missing Values in Python

- Replace missing values in Python.
- replace()

```
my_mean = df_test["Sales"].mean()
df_test["Sales"].replace(np.nan, int(my_mean), inplace = True)
df_test.head()
```

	Customer	Customer Type	Payment Type	Purchases	Sales
0	10000	Person	Cash	120000.0	150000.0
2	10002	Company	Credit Card	451000.0	563750.0
3	10003	Company	Transfer	565000.0	706250.0
4	10004	Person	Transfer	512300.0	NaN

	Customer	Customer Type	Payment Type	Purchases	Sales
0	10000	Person	Cash	120000.0	150000.0
2	10002	Company	Credit Card	451000.0	563750.0
3	10003	Company	Transfer	565000.0	706250.0
4	10004	Person	Transfer	512300.0	553509.0

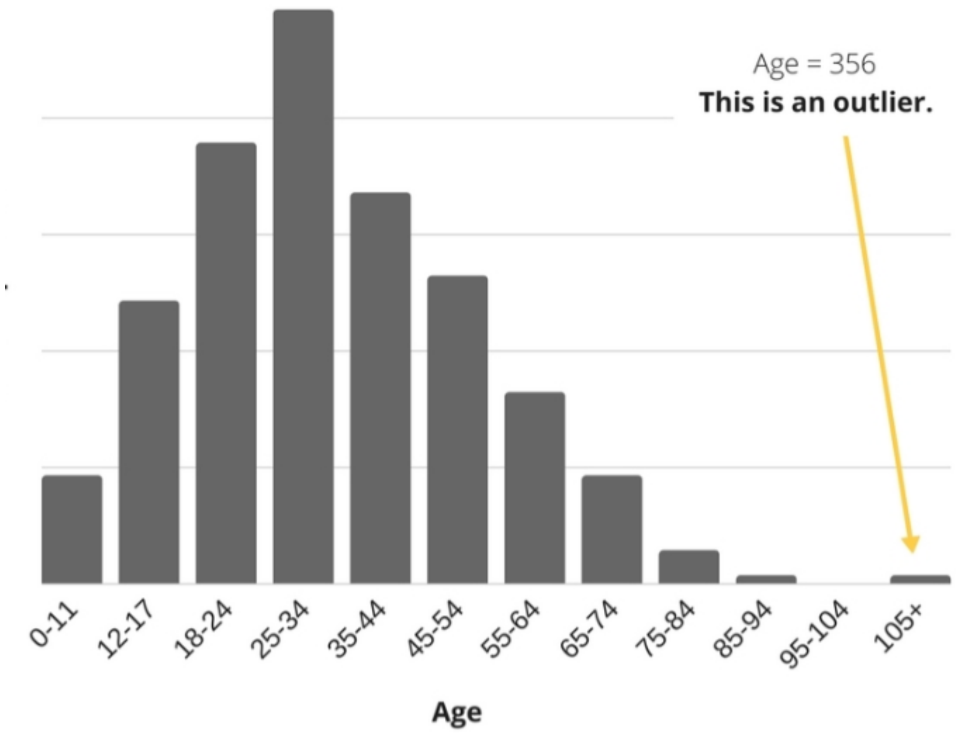
Dealing with duplicate data

- Duplicate data

id	first_name	last_name	email
1	Carine	Schmitt	carine.schmitt@verizon.net
4	Janine	Labrune	janine.labrune@aol.com
6	Janine	Labrune	janine.labrune@aol.com
2	Jean	King	jean.king@me.com
12	Jean	King	jean.king@me.com
5	Jonas	Bergulfsen	jonas.bergulfsen@mac.com
10	Julie	Murphy	julie.murphy@yahoo.com
11	Kwai	Lee	kwai.lee@google.com
3	Peter	Ferguson	peter.ferguson@google.com
9	Roland	Keitel	roland.keitel@yahoo.com
14	Roland	Keitel	roland.keitel@yahoo.com
7	Susan	Nelson	susan.nelson@comcast.net
13	Susan	Nelson	susan.nelson@comcast.net
8	Zbyszek	Piestrzeniewicz	zbyszek.piestrzeniewicz@att.net

Dealing with outlier

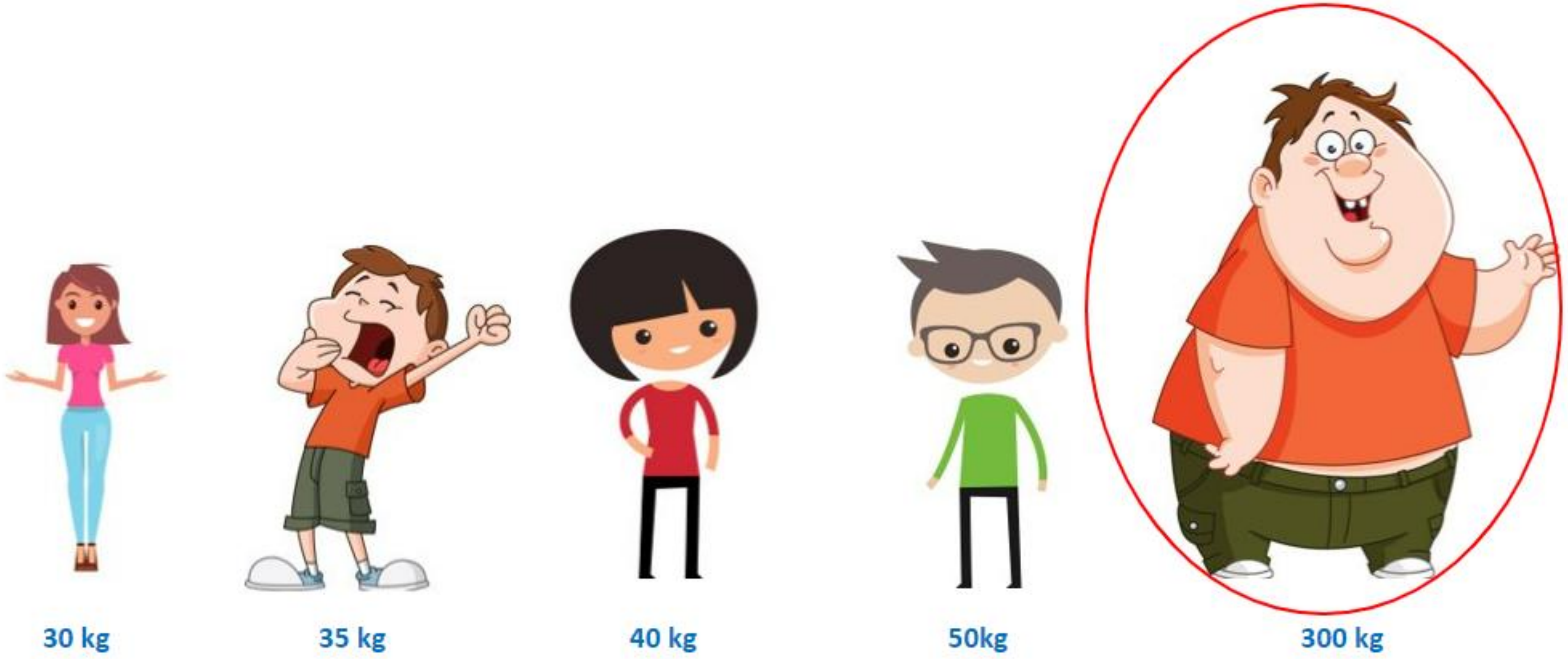
- **Outlier** is an observation in a given dataset that lies far from the rest of the observations.



Dealing with outlier

- Affect of outlier data

Average weight of first 4 kids = (30 + 35 + 40 + 50)/4 = 38.75 kg
Average weight of all kids = (30 + 35 + 40 + 50 + 300)/5 = 91 kg



Dealing with outlier

- Example
  - sample= [15, 101, 18, 7, 13, 16, 11, 21, 5, 15, 10, 9]

```
+---------------------+--------------------+
| with outlier        | without outlier    |
+---------------------+--------------------+
| Mean: 20.08         | Mean: 12.72        |
| Median: 14.0        | Median: 13.0       |
| Mode: 15            | Mode: 15           |
| Variance: 614.74    | Variance: 21.28    |
| Std dev: 24.79      | Std dev: 4.61      |
+---------------------+--------------------+
```

Data Formatting

- Different origins usually lead to different formats.
- A common data format makes data easy to compare and interpret.

Gender
0
1
Man
Woman
Male
Female
M
F

Gender
Male
Female
Male
Female
Male
Female
Male
Female

Data Formatting

- Converting numeric variables.

```
df_test["Purchases in thousands"] = df_test["Purchases"]/1000
df_test["Sales in thousands"] = df_test["Sales"]/1000
df_test["Refunds in thousands"] = df_test["Refunds"]/1000

df_test.head()
```

	Customer	Customer Type	Payment Type	Purchases	Sales
0	10000	Person	Cash	120000.0	150000.0
2	10002	Company	Credit Card	451000.0	563750.0
3	10003	Company	Transfer	565000.0	706250.0
4	10004	Person	Transfer	512300.0	553509.0
5	10005	Person	Transfer	415500.0	519375.0

Purchases in thousands	Sales in thousands	Refunds in thousands
120.0	150.000	0.240
451.0	563.750	0.902
565.0	706.250	1.130
512.3	553.509	1.024
415.5	519.375	0.000

Data Formatting

- Correcting data type.
- dtypes

```
df_test.dtypes
```

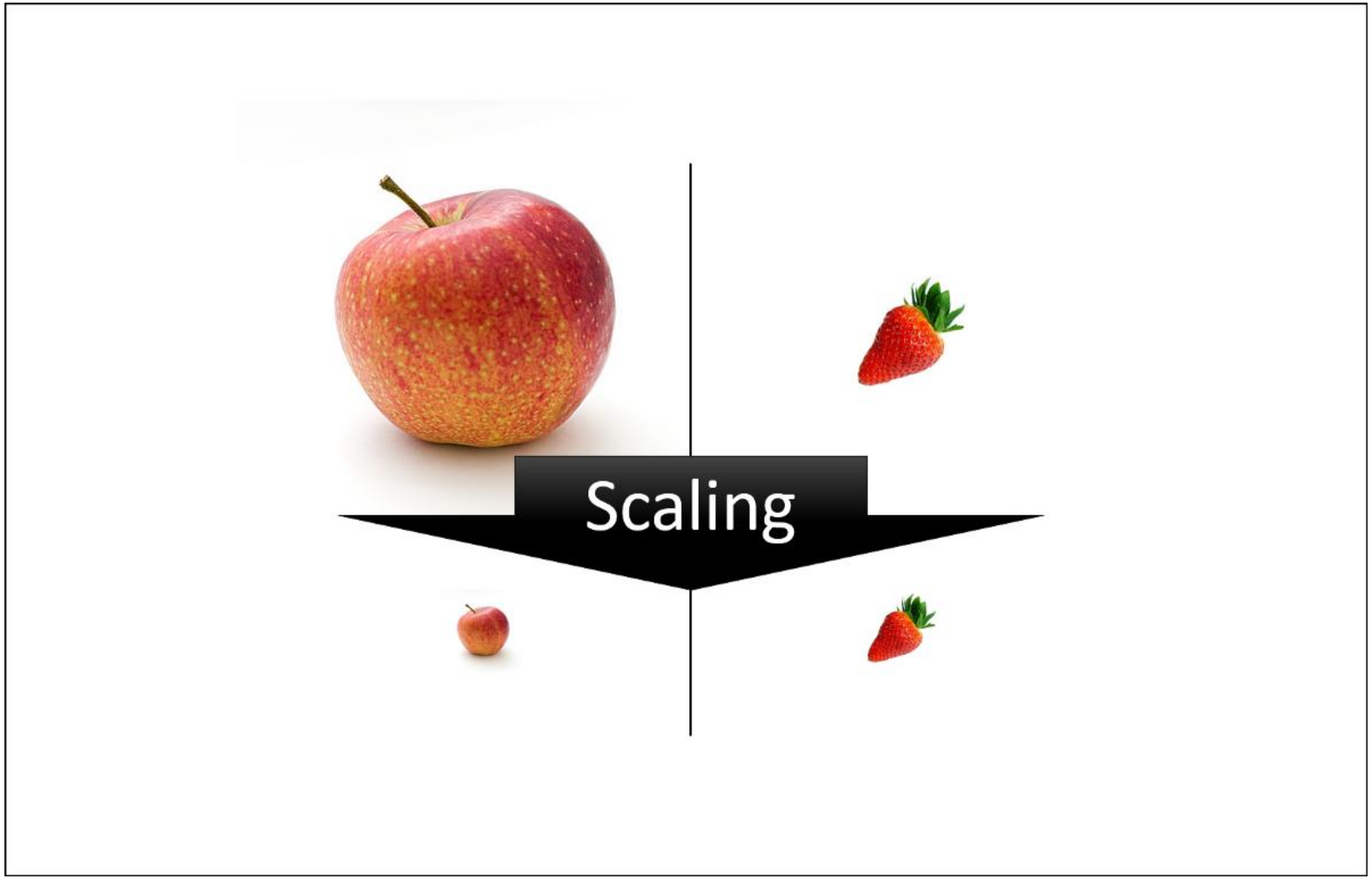
```
Customer                      int64
Customer Type                object
Payment Type                 object
Purchases                   float64
Sales                       float64
Refunds                       int64
Country                      object
Continent                    object
Purchases in thousands      float64
Sales in thousands          float64
Refunds in thousands        float64
dtype: object
```

- astype()

```
df_test = df_test.astype({'Customer':'object'})
df_test.dtypes
```

```
Customer                     object
Customer Type                object
Payment Type                 object
Purchases                   float64
Sales                       float64
Refunds                       int64
Country                      object
Continent                    object
Purchases in thousands      float64
Sales in thousands          float64
Refunds in thousands        float64
dtype: object
```

Data Scaling



Data Scaling

- It consists of putting the data in a **similar range** to be able to compare them.

	No	Employee ID	First Name	Last Name	Age	Worked years	Salary	Status	Grade
0	1	1000001	John	Denver	23	1	500	Single	Elementary
1	2	1000002	Peter	Hank	30	3	900	Married	High School
2	3	1000003	Jack	Sullivan	27	2	900	Married	High School
3	4	1000004	Marco	Aurelio	40	8	1500	Married	Master Degree
4	5	1000005	Claudia	Perez	35	5	1300	Single	Master Degree
5	6	1000006	Sally	Royal	19	1	1400	Single	Graduate
6	7	1000007	Peter	Miller	33	4	600	Married	Graduate
7	8	1000008	Susan	Gordon	35	10	2000	Married	Master Degree

Data Scaling

- Data Scaling Methods.
  - Simple Feature Scaling

$$x_{new} = \frac{x_{current}}{x_{maximum}}$$

  - Min-Max Scaling

$$x_{new} = \frac{x_{current} - x_{minimum}}{x_{maximum} - x_{minimum}} \qquad 0 \leq x_{new} \leq 1$$

  - Standard or Z-score Scaling

$$x_{new} = \frac{x_{current} - Mean}{Standard\ Deviation} \qquad -3 \leq x_{new} \leq 3$$

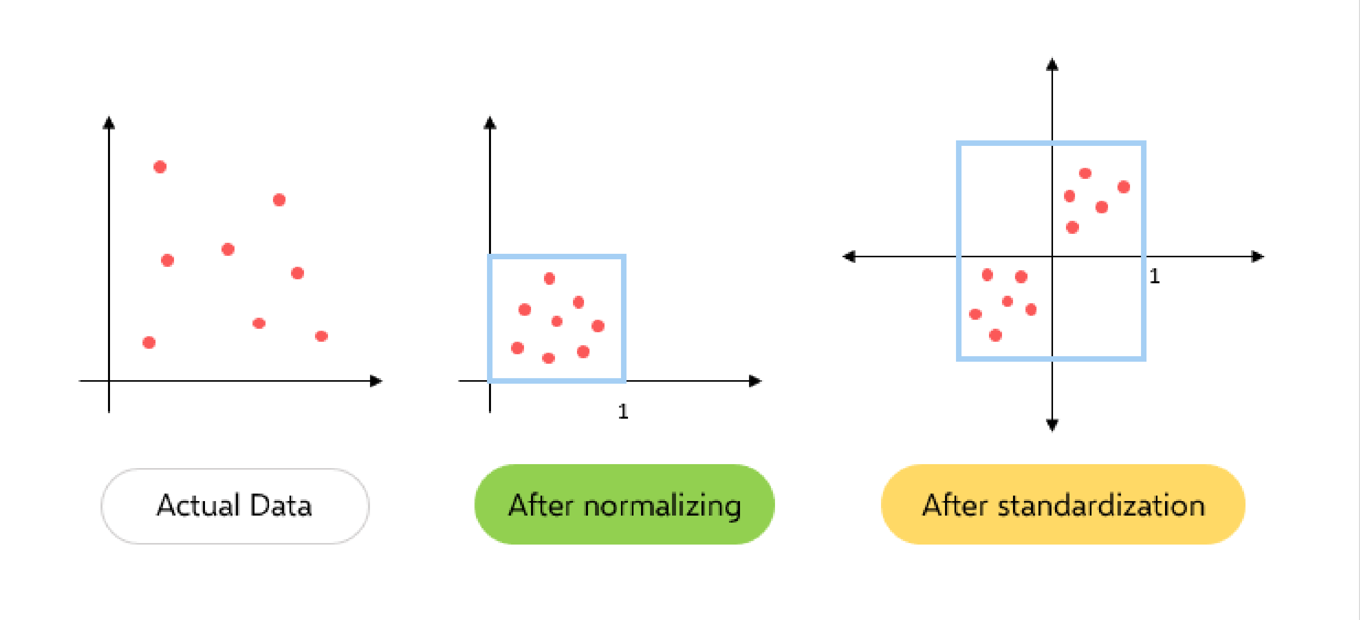
Data Scaling

Standardisation

	Age	Salary
0	0.758874	7.494733e-01
1	-1.711504	-1.438178e+00
2	-1.275555	-8.912655e-01
3	-0.113024	-2.532004e-01
4	0.177609	6.632192e-16
5	-0.548973	-5.266569e-01
6	0.000000	-1.073570e+00
7	1.340140	1.387538e+00
8	1.630773	1.752147e+00
9	-0.258340	2.937125e-01

Max-Min Normalization

	Age	Salary
0	0.739130	0.685714
1	0.000000	0.000000
2	0.130435	0.171429
3	0.478261	0.371429
4	0.565217	0.450794
5	0.347826	0.285714
6	0.512077	0.114286
7	0.913043	0.885714
8	1.000000	1.000000
9	0.434783	0.542857



Data Scaling

- Applying Simple Feature Scaling method in Python.

```
df_employees[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	23	1	500
1	30	3	900
2	27	2	900
3	40	8	1500
4	35	5	1300

```
df_norm1 = df_employees

df_norm1["Age"] = df_norm1["Age"] / df_norm1["Age"].max()
df_norm1["Worked years"] = df_norm1["Worked years"] / df_norm1["Worked years"].max()
df_norm1["Salary"] = df_norm1["Salary"] / df_norm1["Salary"].max()

df_norm1[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	0.575	0.1	0.25
1	0.750	0.3	0.45
2	0.675	0.2	0.45
3	1.000	0.8	0.75
4	0.875	0.5	0.65

Data Scaling

- Applying Min-Max method in Python.

```
df_employees[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	23	1	500
1	30	3	900
2	27	2	900
3	40	8	1500
4	35	5	1300

```
df_norm2 = df_employees

df_norm2["Age"] = (df_norm2["Age"] - df_norm2["Age"].min()) / (df_norm2["Age"].max() - df_norm2["Age"].min())
df_norm2["Worked years"] = (df_norm2["Worked years"] - df_norm2["Worked years"].min()) / (df_norm2["Worked years"].max() - df_norm2["Worked years"].min())
df_norm2["Salary"] = (df_norm2["Salary"] - df_norm2["Salary"].min()) / (df_norm2["Salary"].max() - df_norm2["Salary"].min())

df_norm2[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	0.190476	0.000000	0.000000
1	0.523810	0.222222	0.266667
2	0.380952	0.111111	0.266667
3	1.000000	0.777778	0.666667
4	0.761905	0.444444	0.533333

Data Scaling

- Applying Z-score method in Python.

```
df_employees[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	23	1	500
1	30	3	900
2	27	2	900
3	40	8	1500
4	35	5	1300

```
df_norm3 = df_employees

df_norm3["Age"] = (df_norm3["Age"] - df_norm3["Age"].mean()) / df_norm3["Age"].std()
df_norm3["Worked years"] = (df_norm3["Worked years"] - df_norm3["Worked years"].mean()) / df_norm3["Worked years"].std()
df_norm3["Salary"] = (df_norm3["Salary"] - df_norm3["Salary"].mean()) / df_norm3["Salary"].std()

df_norm3[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	-1.044119	-0.989598	-1.264654
1	-0.036004	-0.380615	-0.471146
2	-0.468053	-0.685106	-0.471146
3	1.404160	1.141844	0.719117
4	0.684078	0.228369	0.322363

Grouping Numerical Data into Classes

- Grouping the data into classes.
- Sales have a range that goes from 100,000 to a little more than 900,000

Sales
150000
651750
563750
706250
640375
519375
870375
926250
676250
103750
567925

Classes	Classes Range
Low	0 - 299,999
Medium	300,000 - 599,999
High	600,000 - 999,999

Grouping into Classes

- Grouping with Python.

```
my_class = np.linspace(min(df_test["Sales"]), max(df_test["Sales"]), 4)
group_names = ["Low","Medium","High"]

df_test["Sales Category"] = pd.cut(df_test["Sales"], my_class, labels = group_names, include_lowest = True)

df_test[['Sales','Sales Category']].head()
```

	Sales	Sales Category
0	150000.0	Low
2	563750.0	Medium
3	706250.0	High
4	553509.0	Medium
5	519375.0	Medium

Converting Categorical Variables to Numeric Variables

- Converting categorical variables to numeric variables.
- get_dummies()

```
df_dummies = pd.get_dummies(df_test['Payment Type'])
df_test2 = pd.concat([df_test, df_dummies], axis = 1, sort = False)

df_test2.head()
```

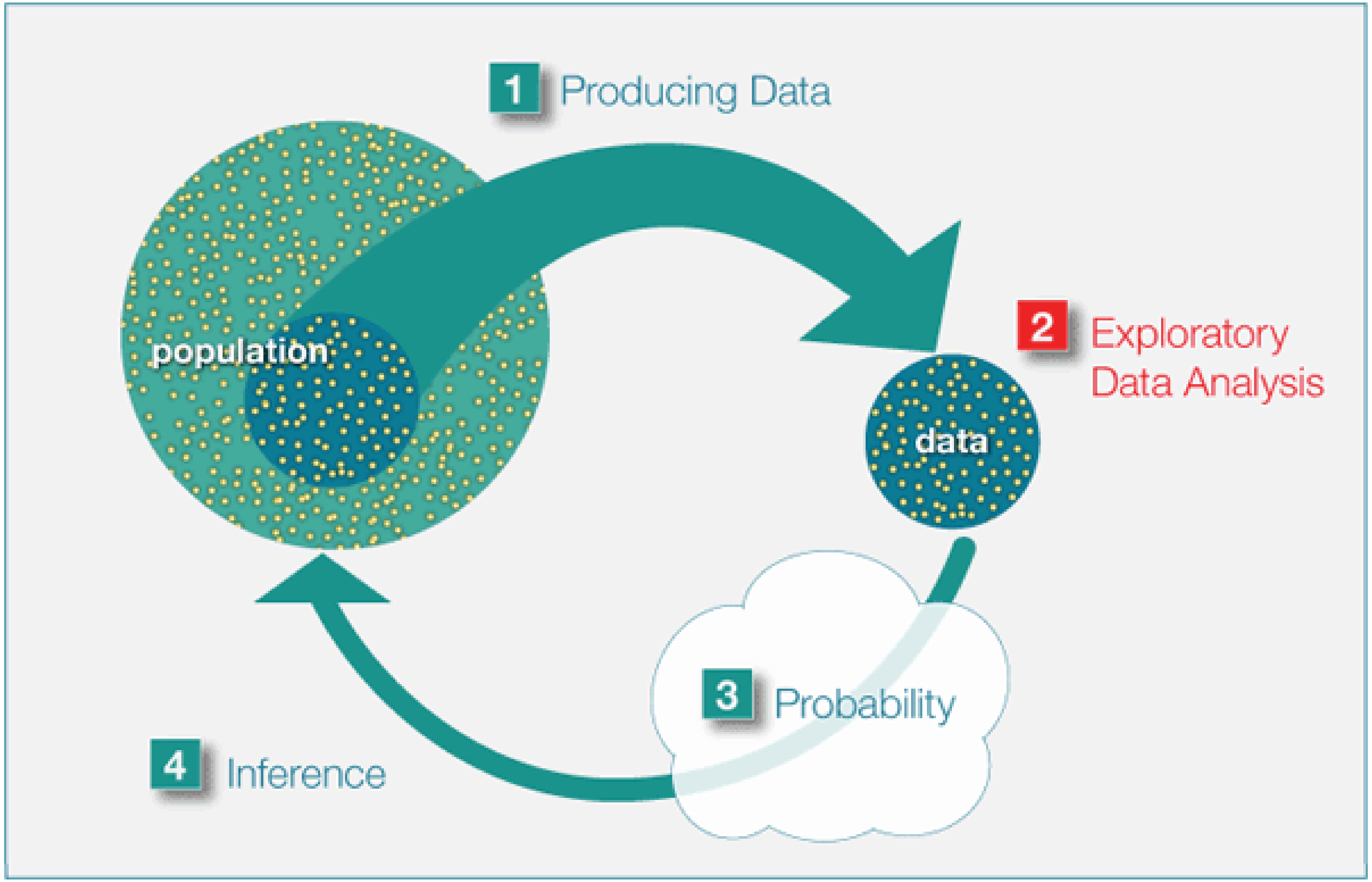
Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent	Purchases in thousands	Sales in thousands	Refunds in thousands	Sales Category	Cash	Credit Card	Transfer
10000	Person	Cash	120000.0	150000.0	240	Canada	America	120.0	150.000	0.240	Low	1	0	0
10002	Company	Credit Card	451000.0	563750.0	902	Mexico	America	451.0	563.750	0.902	Medium	0	1	0
10003	Company	Transfer	565000.0	706250.0	1130	Spain	Europe	565.0	706.250	1.130	High	0	0	1
10004	Person	Transfer	512300.0	553509.0	1024	Argentina	America	512.3	553.509	1.024	Medium	0	0	1
10005	Person	Transfer	415500.0	519375.0	0	Canada	America	415.5	519.375	0.000	Medium	0	0	1

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA)

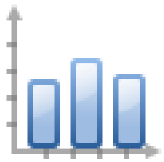
- Exploratory data analysis or "EDA" is a critical step in analyzing the data
- The main reasons are
  - detection of mistakes, outliers or abnormalities
  - checking of assumptions
  - preliminary selection of appropriate models
  - determining relationships among the explanatory variables
  - assessing the direction and rough size of relationships between explanatory and outcome variables

Exploratory Data Analysis



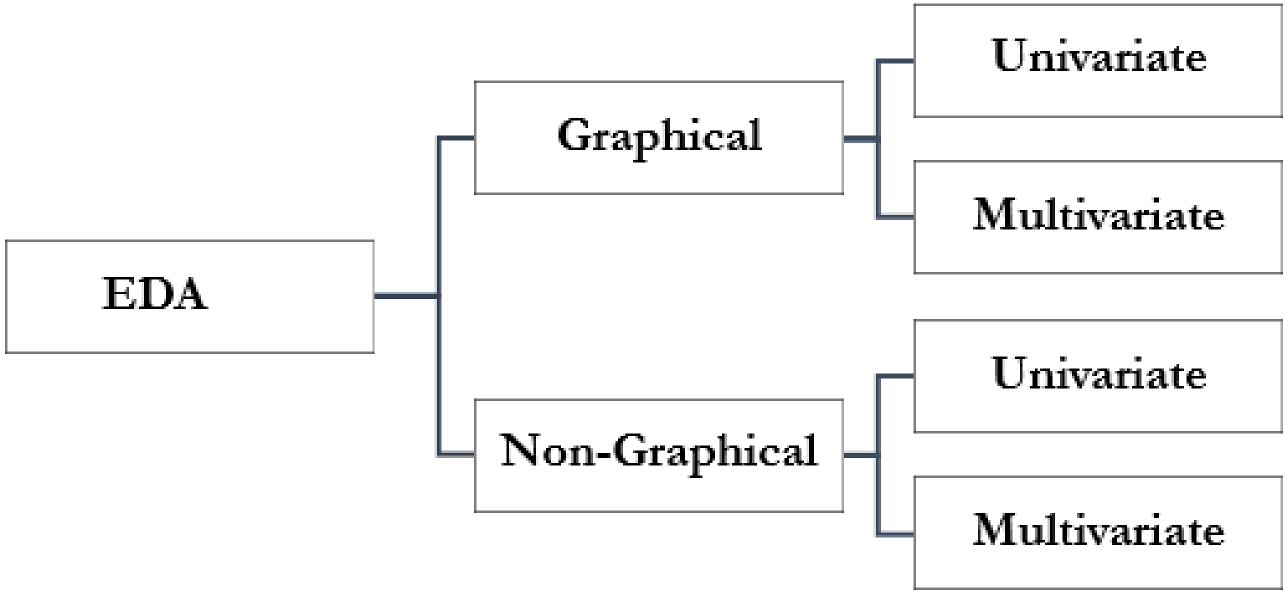
Types of EDA

- Graphical or Non-graphical
  - **Non-graphical** methods usually involve with calculation of summary statistics
  - **Graphical** methods obviously summarize the data in a diagrammatic or pictorial way
- Univariate or Multivariate
  - **Univariate** methods look at one variable (column) at a time while
  - **Multivariate** methods look at two or more variables at a time

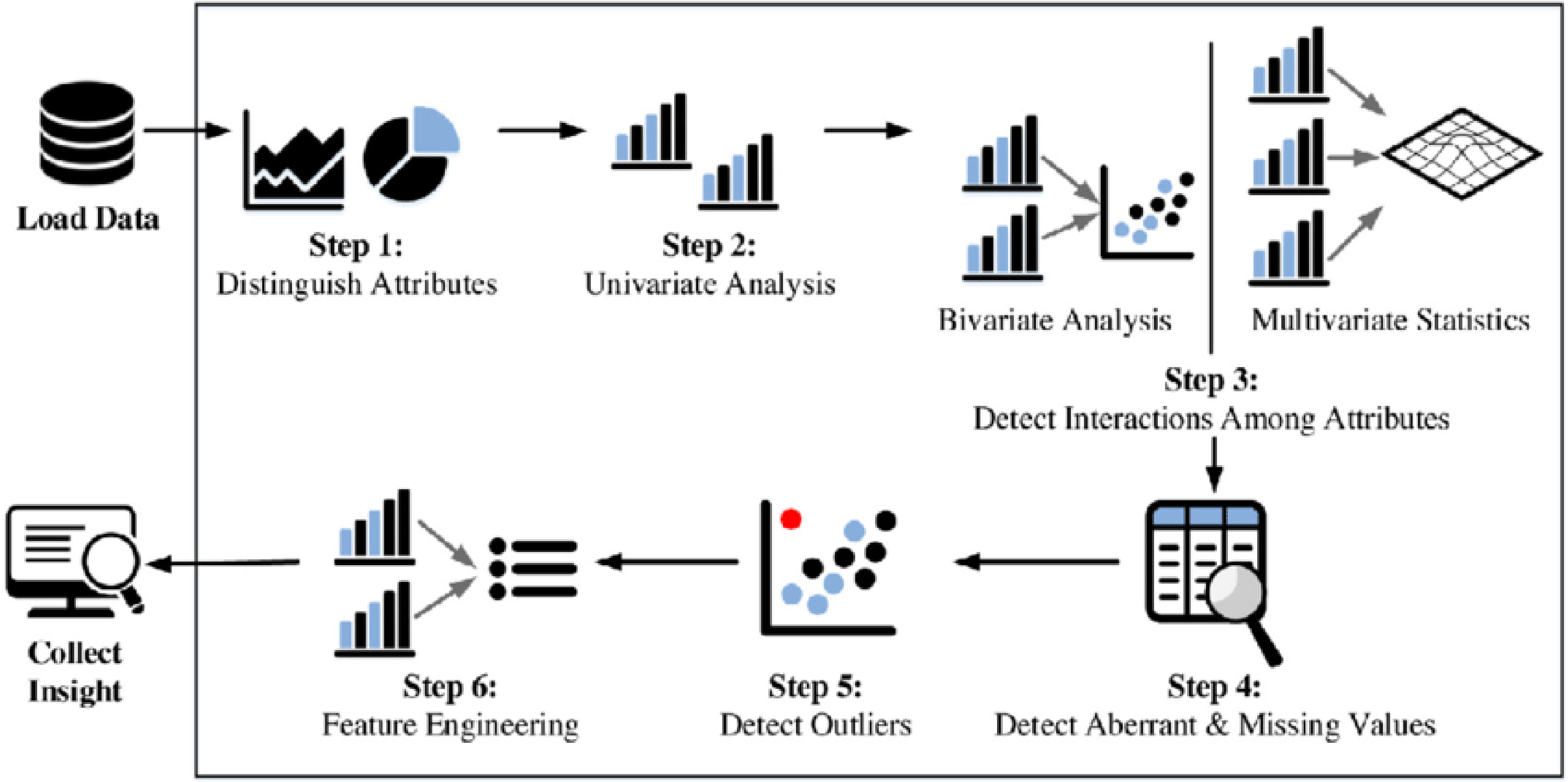


Four basic types of EDA

- Univariate non-graphical
- Multivariate non-graphical
- Univariate graphical
- Multivariate graphical



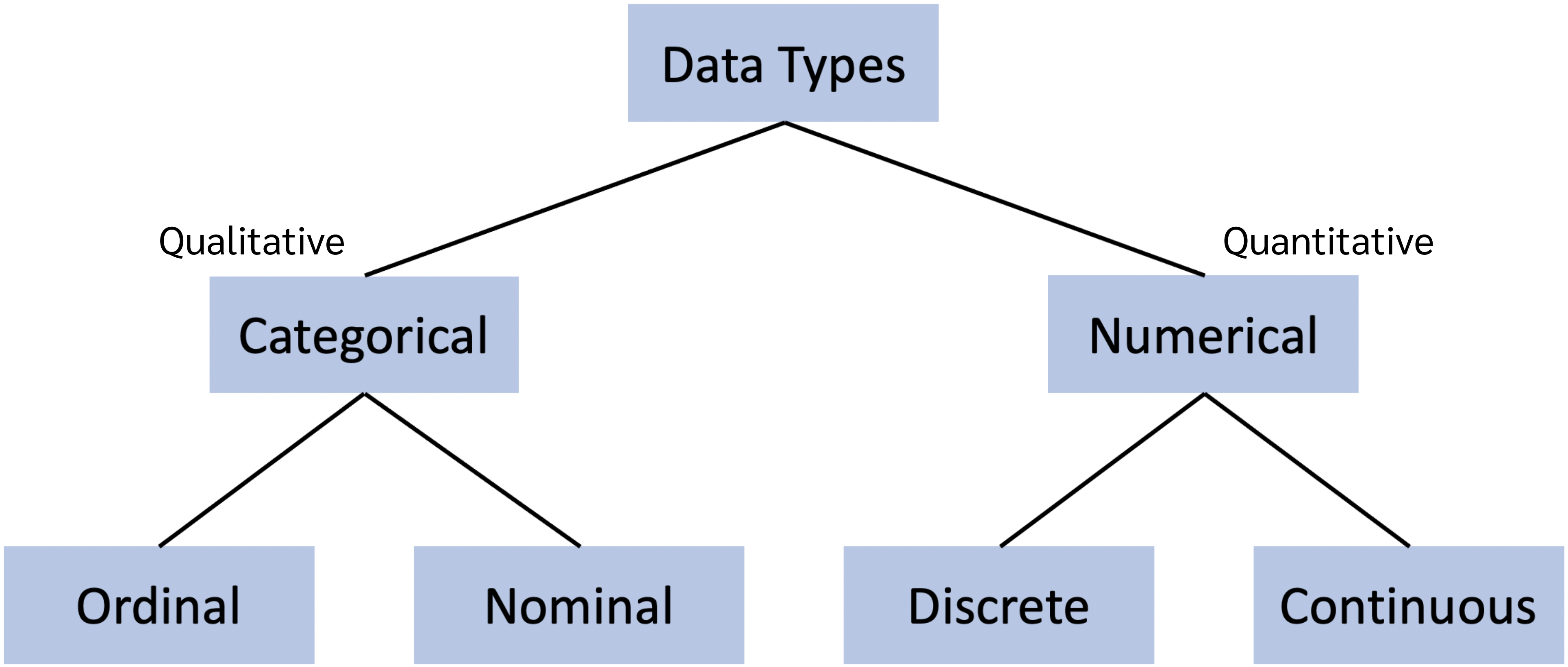
EDA Process Overview



Data format

- Data from either experiments or operations are generally collected in databases (e.g. spreadsheet)
- One row per record and one column for each identifiers, outcome variables, and explanatory variables
- Each column contains the **numerical value** of a particular quantitative variable or the levels for a **categorical variable**

Data Types


Data Types
Qualitative
Quantitative
Categorical
Numerical
Ordinal
Nominal
Discrete
Continuous

Categorical Data: Nominal vs Ordinal

Nominal

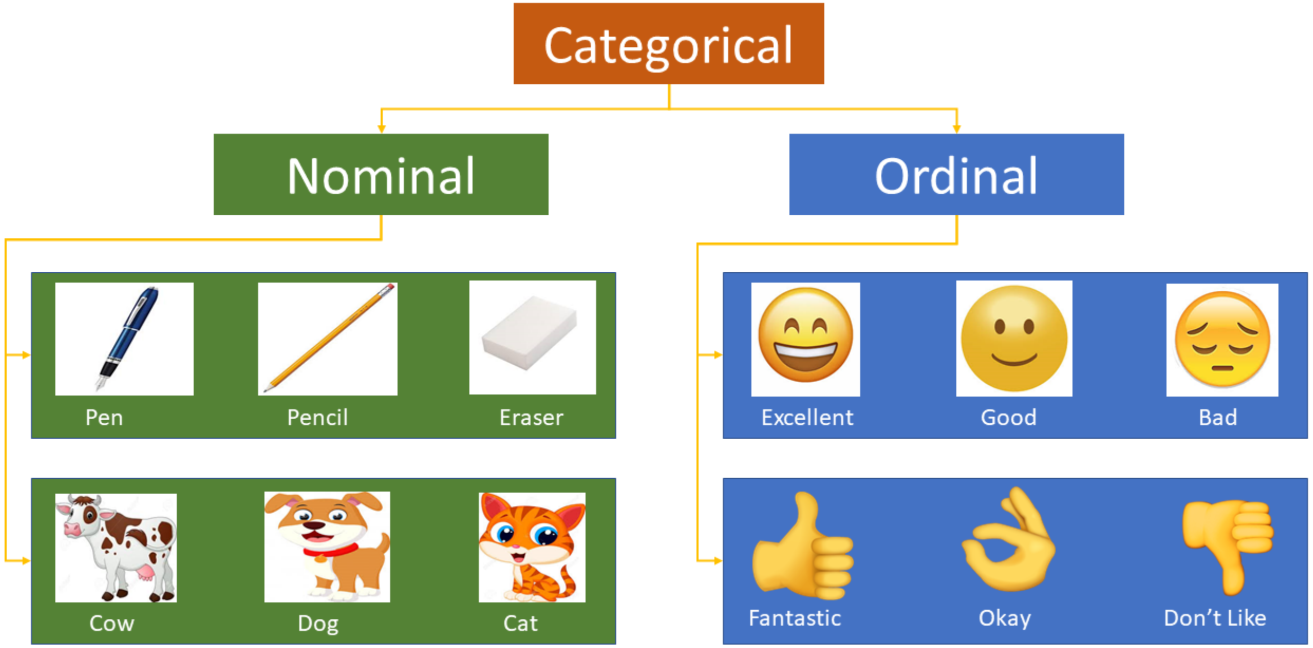
Characteristics can be **distinguished**

A D C B

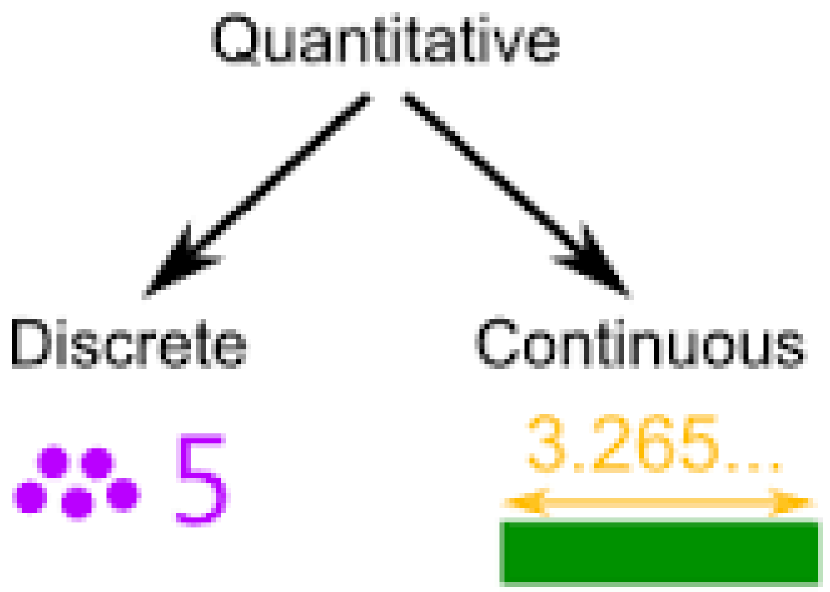
Ordinal

Characteristics can be **sorted**

A < B < C < D



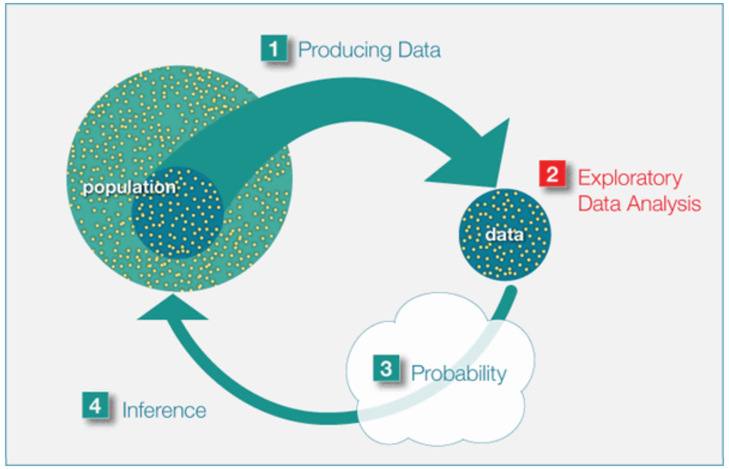
Numerical Data: Nominal vs Ordinal



Univariate non-graphical EDA

Univariate non-graphical EDA

- This is to **measure certain characteristics** (e.g. age, gender, speed at a task, or response to a stimulus) of data of all subjects/records
- We should think of measurements as representations of a "**sample distribution**", which in turn more or less representing the "**population distribution**"
- The goal is to **better understand the "sample distribution"** and **make some conclusion about the "population distribution"**



Univariate non-graphical EDA

Categorical data

- The characteristics of interest for a categorical variable are simply the range of values and the frequency (or relative frequency) of occurrence for each value.
- Therefore, the only useful univariate non-graphical techniques for categorical variables is some form of **tabulation of the frequencies**, usually along with calculation of the fraction (or percent) of data that falls in each category.

Univariate non-graphical EDA

Categorical data

- Sample Data:
  - 4 categories (H&SS, MCS, SCS, and other), take 20 samples
  - True population -> H&SS, H&SS, MCS, other, other, SCS, MCS, other, H&SS, MCS, SCS, SCS, other, MCS, MCS, H&SS, MCS, other, H&SS, SCS.
  - EDA would look like this

Statistic/College	H&SS	MCS	SCS	other	Total
Count	5	6	4	5	20
Proportion	0.25	0.30	0.20	0.25	1.00
Percent	25%	30%	20%	25%	100%

Univariate non-graphical EDA

Categorical data

- describe()

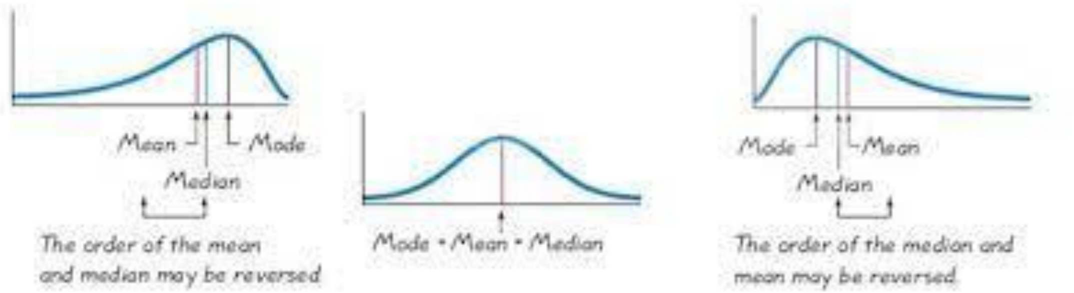
```
df_operations.describe(include = [object])
```

	Customer Type	Payment Type	Country	Continent
count	19	19	19	19
unique	2	3	8	3
top	Company	Cash	EEUU	America
freq	10	8	6	14

Univariate non-graphical EDA

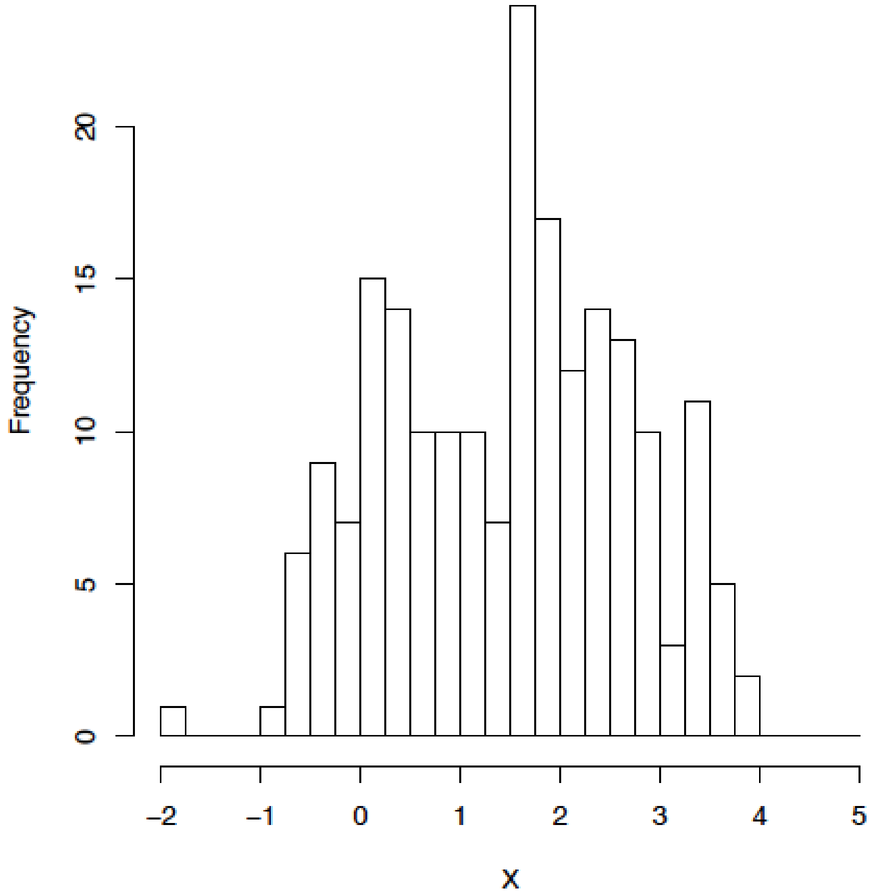
Quantitative data

- Univariate EDA for a quantitative variable is a way to make preliminary assessments about the **population distribution** of the variable using the data of the observed sample.
- The **characteristics of the population distribution** of a quantitative variable are its
  - Center
  - Spread
  - Shape
  - Outliers



Univariate non-graphical EDA

Quantitative data



Central tendency
Spread
Skewness
Etc.

What can you see from this histogram?

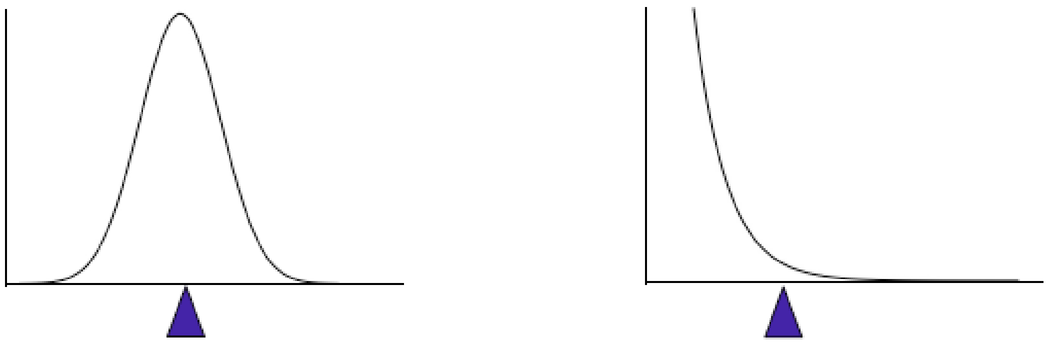
Central tendency (Middle Value)

Mean

I. The Mean

To calculate the average $\bar{x}$ of a set of observations, add their value and divide by the number of observations:

$$\bar{x} = \frac{x_1 + x_2 + x_3 + \ldots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i$$



Central tendency

Median

- **Median** – the exact middle value

- **Calculation**:
  - If there are an odd number of observations, find the middle value

  - If there are an even number of observations, find the middle two values and average them

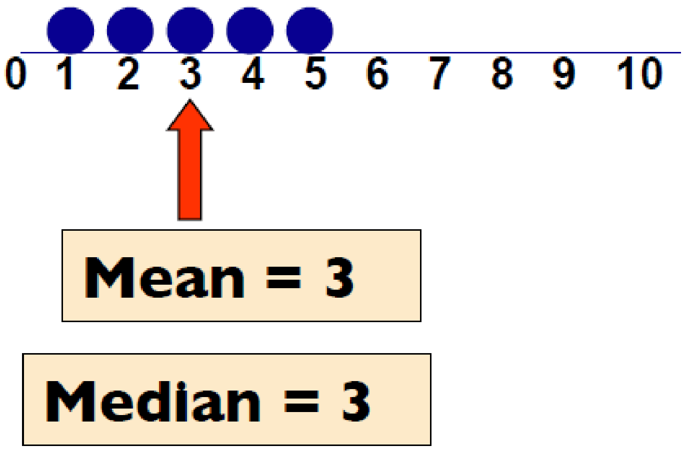
- **Example**

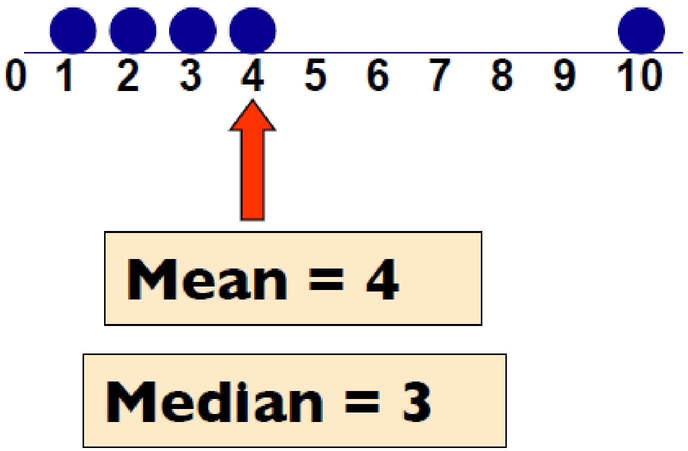
  Some data:
  Age of participants: 17 19 21 22 23 23 23 38

  Median = (22+23)/2 = 22.5

Which location measure is the best?

- Mean is best for symmetric distributions without outliers
- Median is useful for skewed distributions or data with outliers





Scale: Variance Standard measurement of Spread

- Average of squared deviations of values from the mean

$$\hat{\sigma}^2 = \frac{\sum_{i}^{n}(x_i - \bar{x})^2}{n-1}$$

Why squared deviations?

- Absolute values do not have nice mathematical properties (non-linear)
- Squares eliminate the negatives

- Results:
  - Increasing contribution to the variance as you go farther from the mean

Scale: Variance

- Variance is somewhat arbitrary
- What does it mean to have a variance of 8.9? Or 1.5? Or 1245.34? Or 0.00001?
- Nothing. But if you could “standardize” that value, you could talk or compare about any variance (i.e. deviation) in equivalent terms
- Standard deviations are simply the square root of the variance

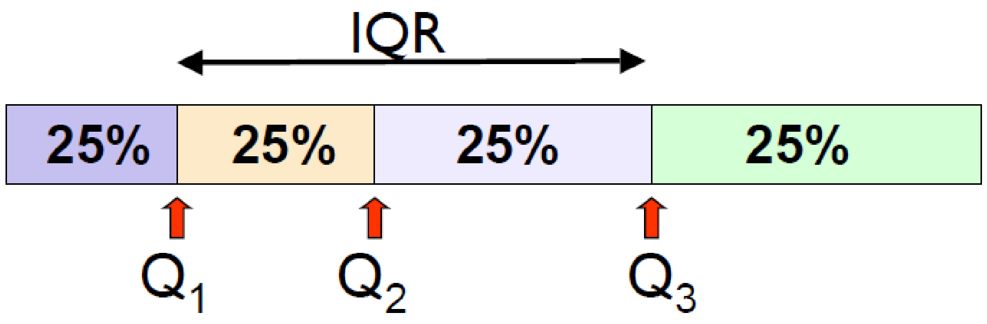
Scale: Standard deviation

$$\hat{\sigma} = \sqrt{\frac{\sum_{i}^{n}(x_i - \bar{x})^2}{n-1}}$$

For Normally distributed data, approximately 95% of the values lie within 2 SD of the mean.

1. Score (in the units that are meaningful)
2. Mean
3. Each score's deviation from the mean
4. Square that deviation
5. Sum all the squared deviations (Sum of Squares)
6. Divide by n-1
7. Square root – now the value is in the units we started with!!!

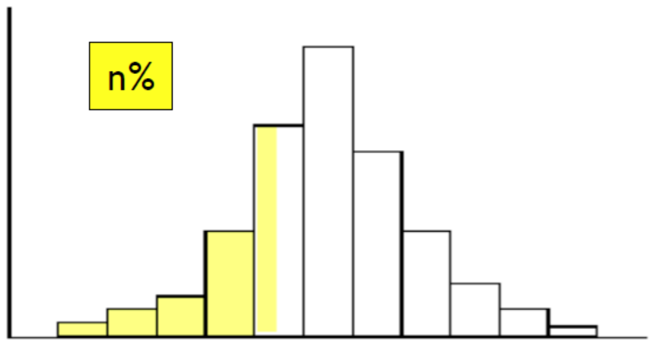
Scale: Quartiles and IQR



- The first quartile, Q_1, is the value for which 25% of the observations are smaller and 75% are larger
- Q_2 is the same as the median (50% are smaller, 50% are larger)
- Only 25% of the observations are greater than the third quartile

Percentiles (aka Quantiles)

In general the **n^{th} percentile** is a value such that n% of the observations fall at or below or it

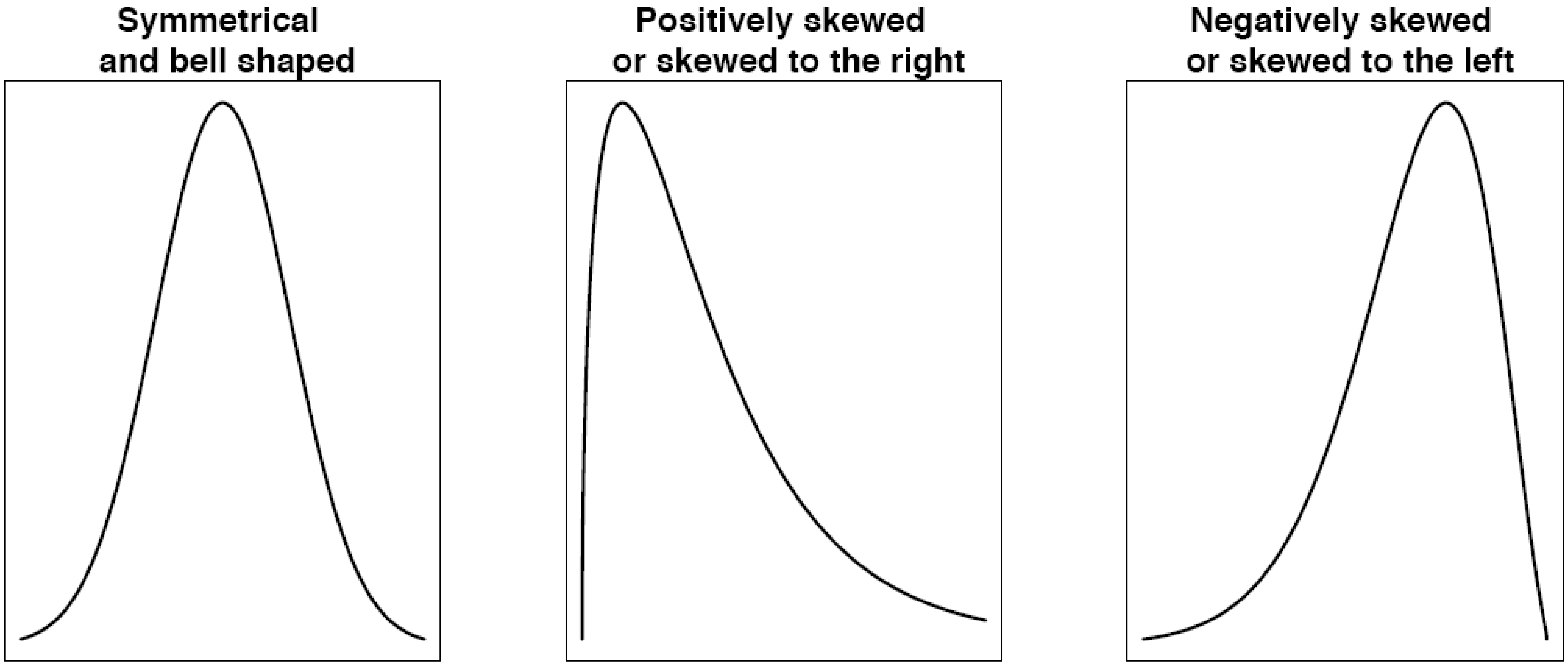


Q_1 = 25^{th} percentile

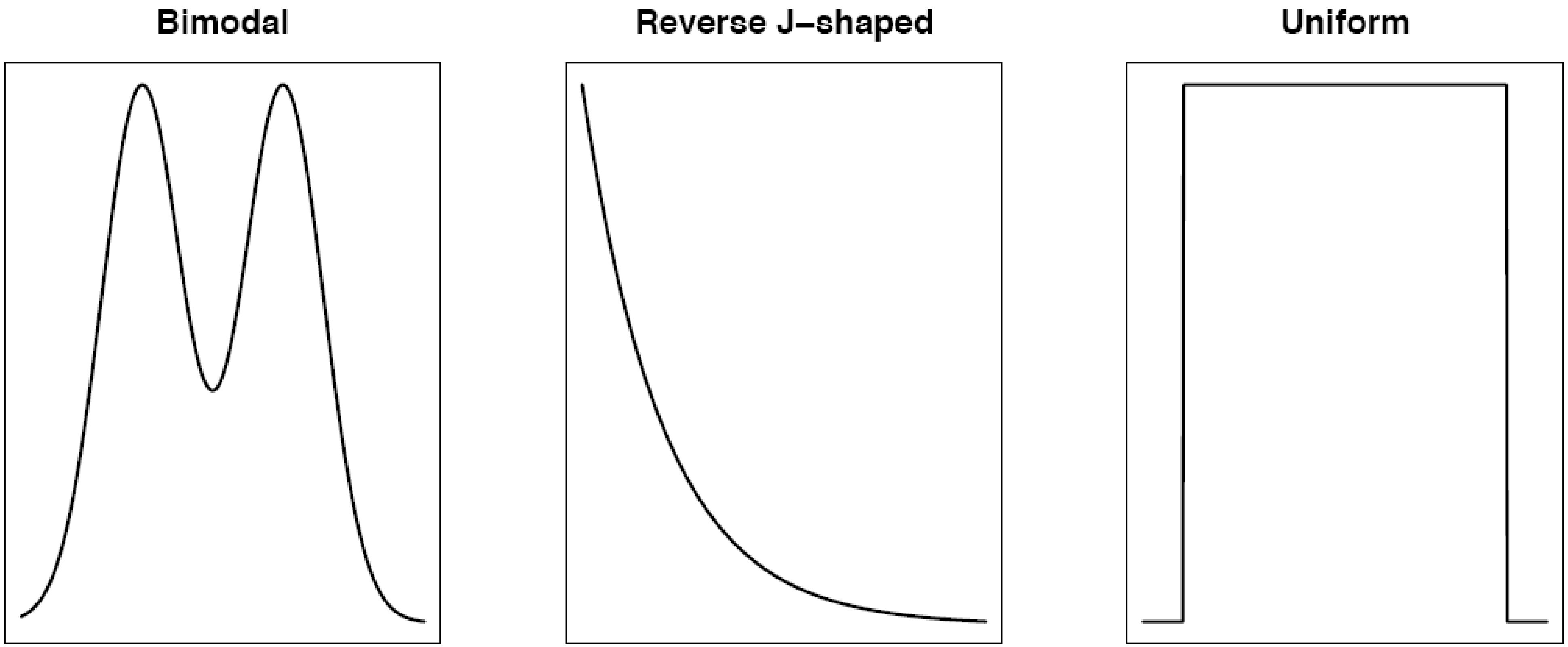
Median = 50^{th} percentile

Q_2 = 75^{th} percentile

Common distribution shapes



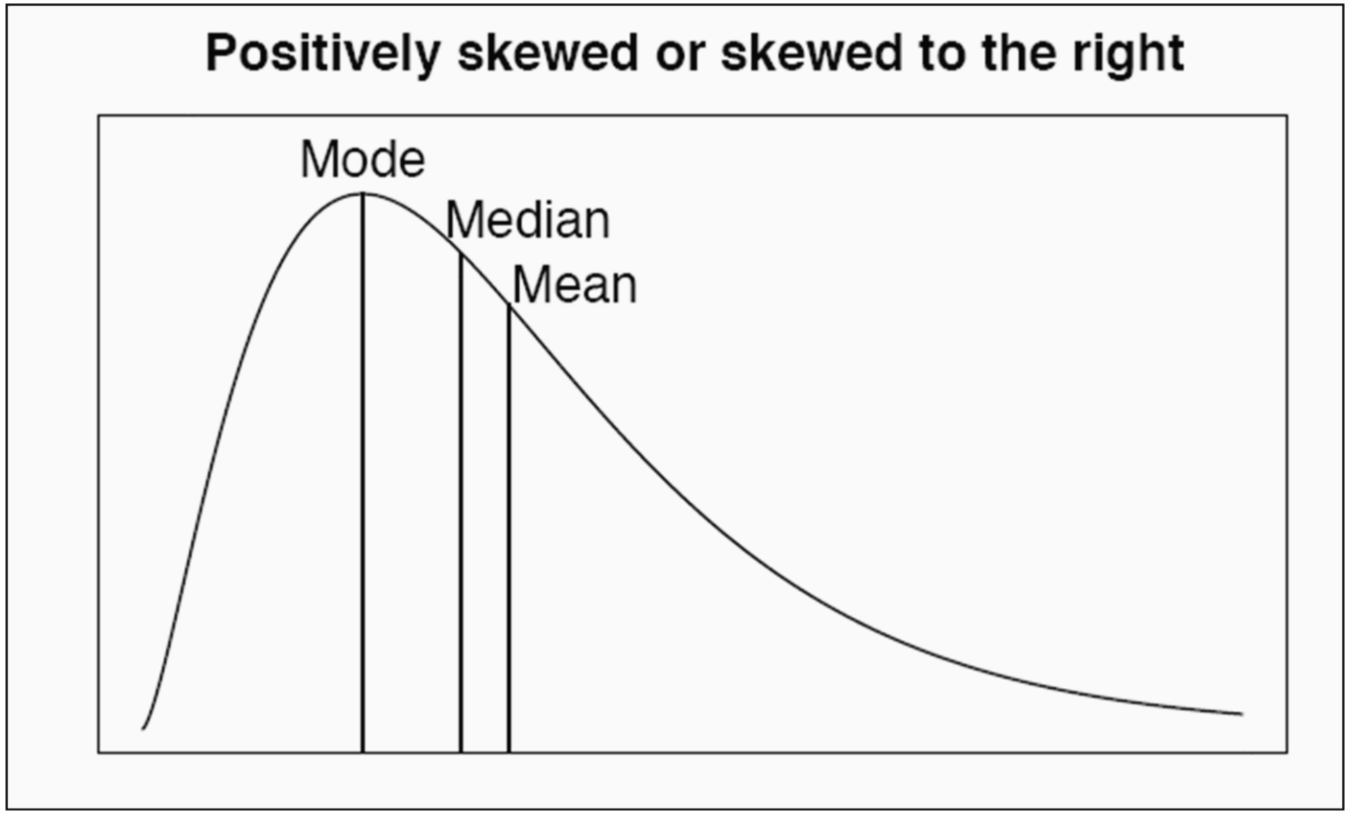
Other distribution shapes



Skewness I

Positively skewed

- Longer tail in the high value
- Mean > Median > Mode



Univariate non-graphical EDA

Quantitative data In Python

- describe()

```
df_operations.describe()
```

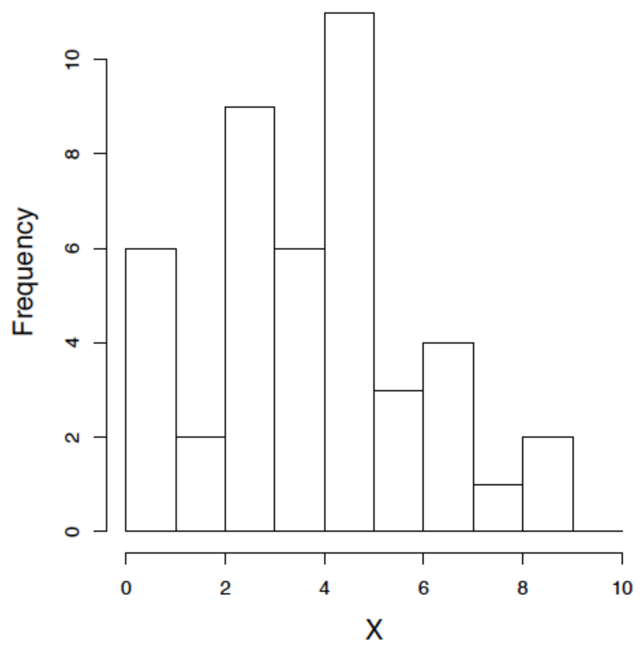
	Customer	Purchases	Sales	Refunds
count	19.000000	19.000000	19.000000	19.000000
mean	10009.000000	450589.210526	563252.210526	819.210526
std	5.627314	167280.787361	209101.355900	439.467554
min	10000.000000	83000.000000	103750.000000	0.000000
25%	10004.500000	388850.000000	486062.500000	592.000000
50%	10009.000000	454100.000000	567925.000000	910.000000
75%	10013.500000	531200.000000	664000.000000	1062.500000
max	10018.000000	741000.000000	926250.000000	1482.000000

Univariate Graphical EDA

Univariate Graphical EDA

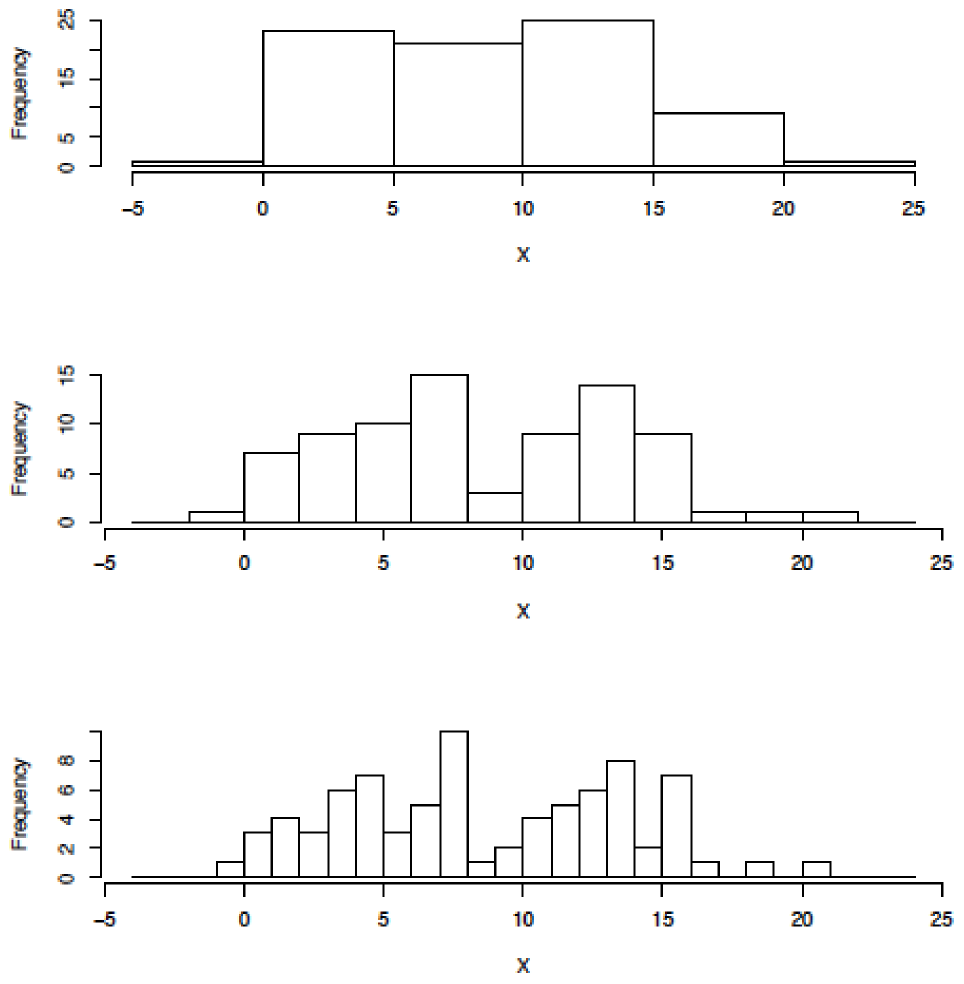
Histogram

- **Histogram** is a graphical representation of the **distribution of numerical data**
- It provides a view of data **density** and the **shape** of data distribution
- To construct a histogram, the first step is to
  - bin the range of values
  - count how many values fall into each interval
- The **bins** are usually specified as consecutive, non-overlapping intervals of a variable.
- The bins (intervals) must be adjacent and are usually equal size.



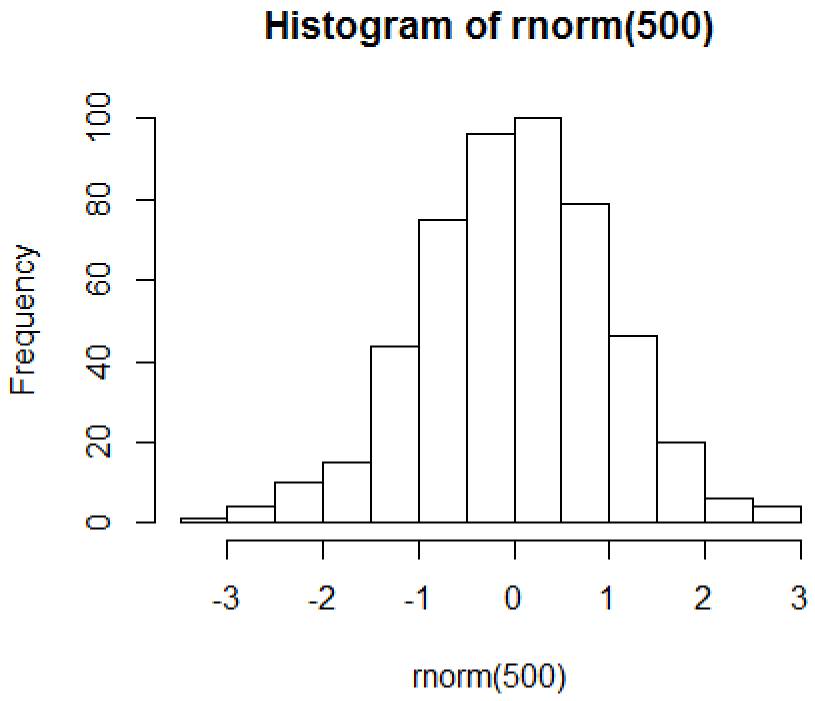
Univariate Graphical EDA

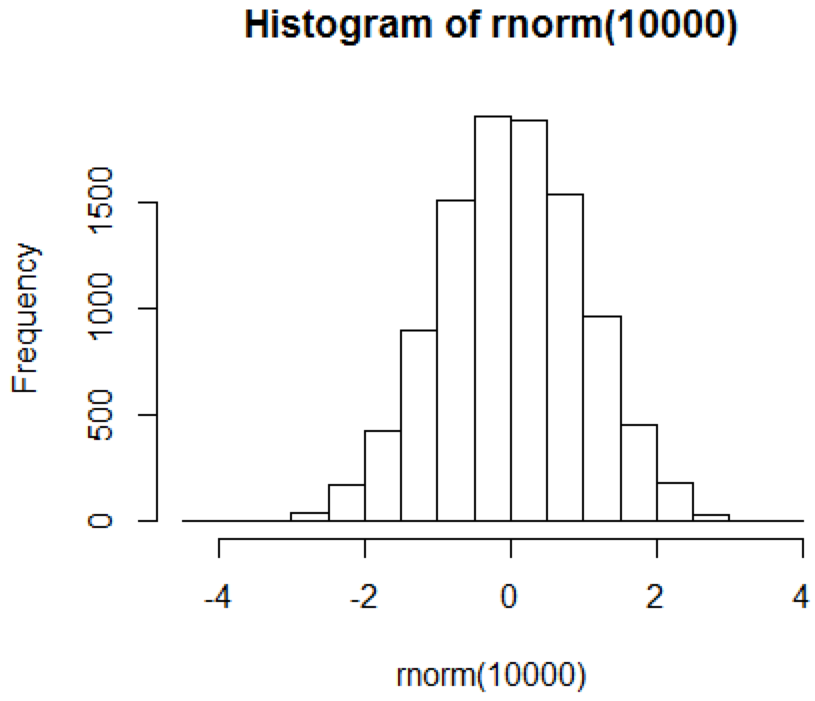
Effects of Histogram Bin



Univariate Graphical EDA

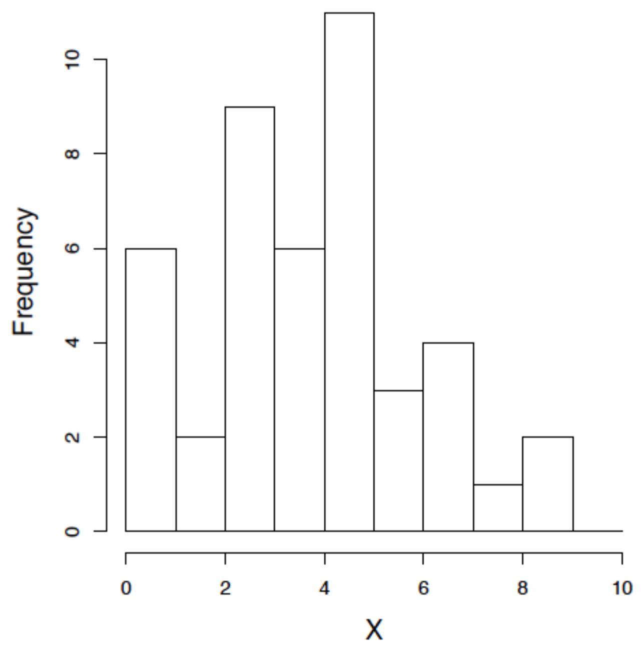
Effects of Number of Samples





Univariate Graphical EDA

Histogram



- **Histogram** is a graphical representation of the **distribution of numerical data**
- It provides a view of data **density** and the **shape** of data distribution
- To construct a histogram, the first step is to
  - bin the range of values
  - count how many values fall into each interval
- The **bins** are usually specified as consecutive, non-overlapping intervals of a variable.
- The bins (intervals) must be adjacent and are usually equal size.

Univariate Graphical EDA

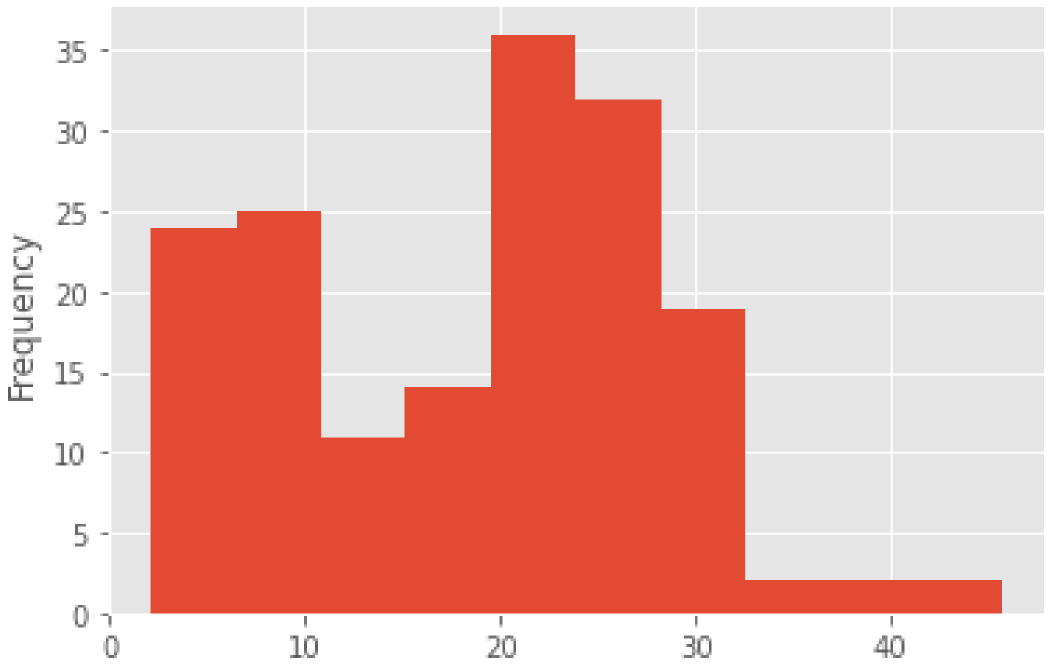
Histogram in Python

- Histrogram.
  - plot() method.

```
Country
Afghanistan            4.5
Albania               22.3
Algeria               26.6
Angola                 6.8
Antigua and Barbuda   19.1
Argentina             28.5
Armenia               20.9
Australia             30.4
Austria               21.9
Azerbaijan            19.9
Name: Obesity, dtype: float64
```

```python
# An easy way to create the histogram.
df_fat['Obesity'].plot.hist()
```

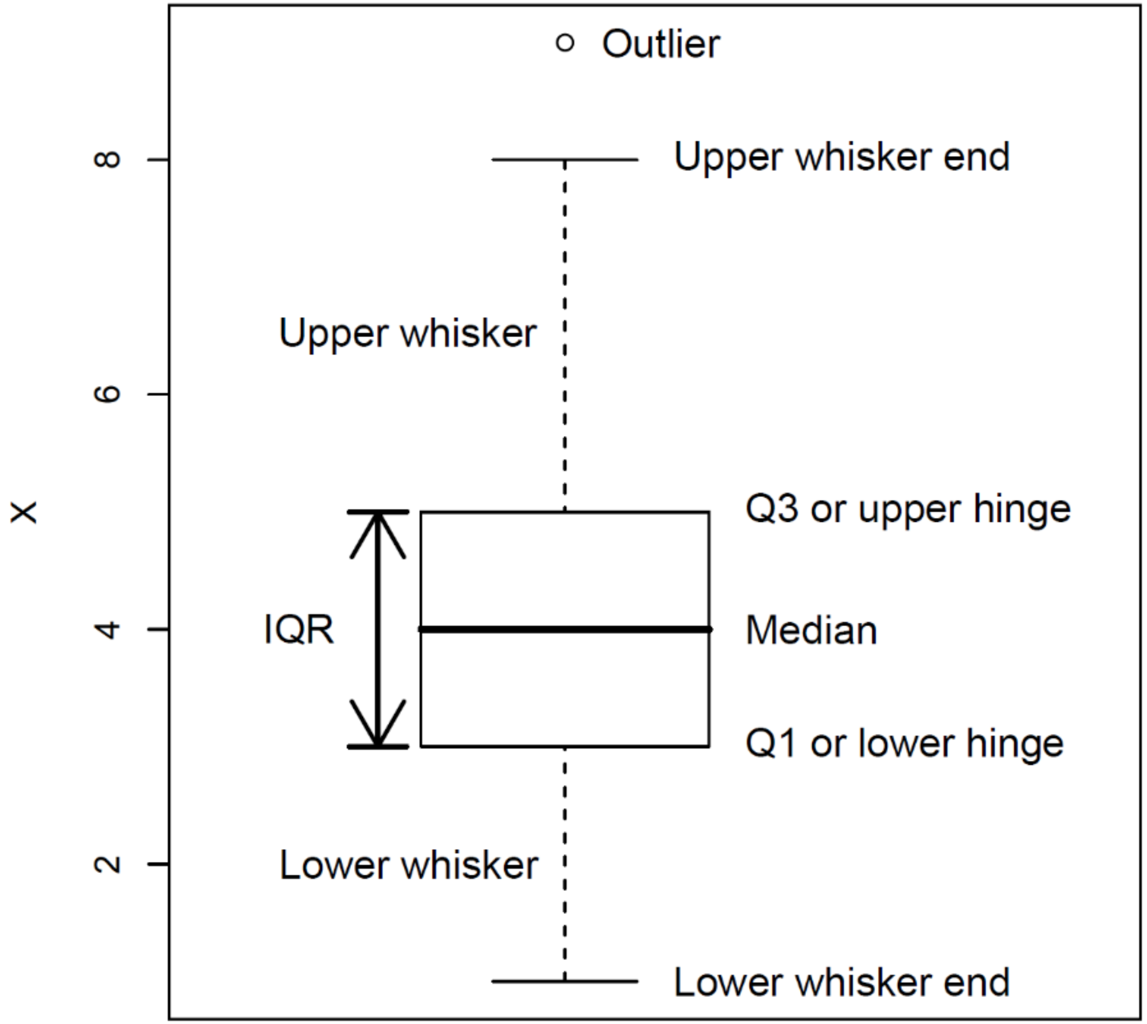
```
<AxesSubplot:ylabel='Frequency'>
```



Univariate Graphical EDA

Boxplot

- The box in boxplot represents the middle 50% o
- The middle line indicates median
- Whiskers can be designated as either
  - Max/Min
  - Outlier boundaries
    - Upper = Q3 + 1.5*IQR
    - Lower = Q1 – 1.5*IQR

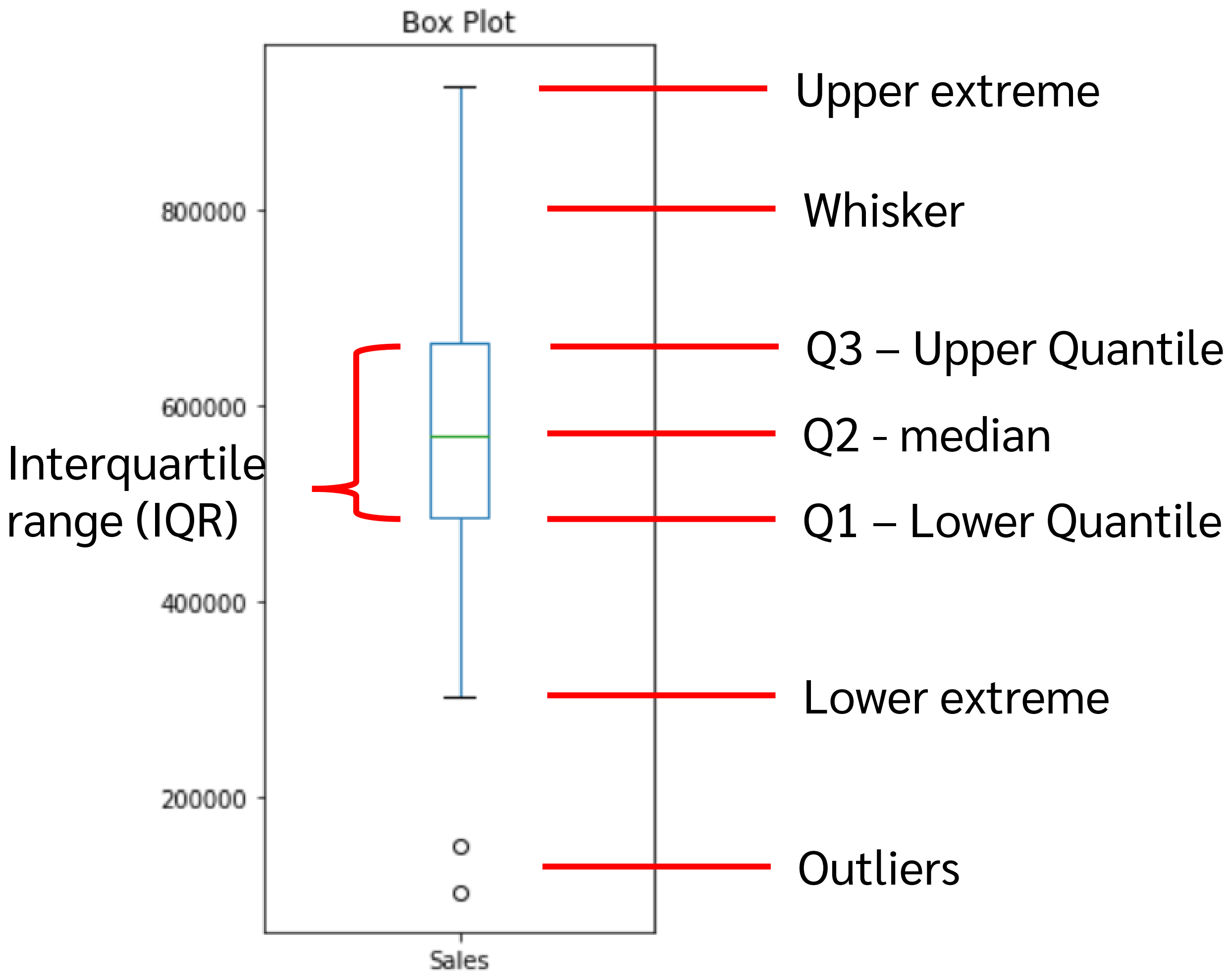


Boxplot in Python

- Box Plots with matplotlib library

```
#We only take the variable sales.
df_oper_sales = df_operations.loc[:,'Sales']

df_oper_sales.plot(kind='box', figsize=(3,7))
plt.title('Box Plot')
plt.show()
```



Multivariate Graphical EDA

Multivariate Non-Graphical EDA

- Multivariate non-graphical EDA techniques show the **relationship between two or more variables** in the form of either <u>cross-tabular</u> or <u>statistics</u>

Multivariate Non-Graphical EDA

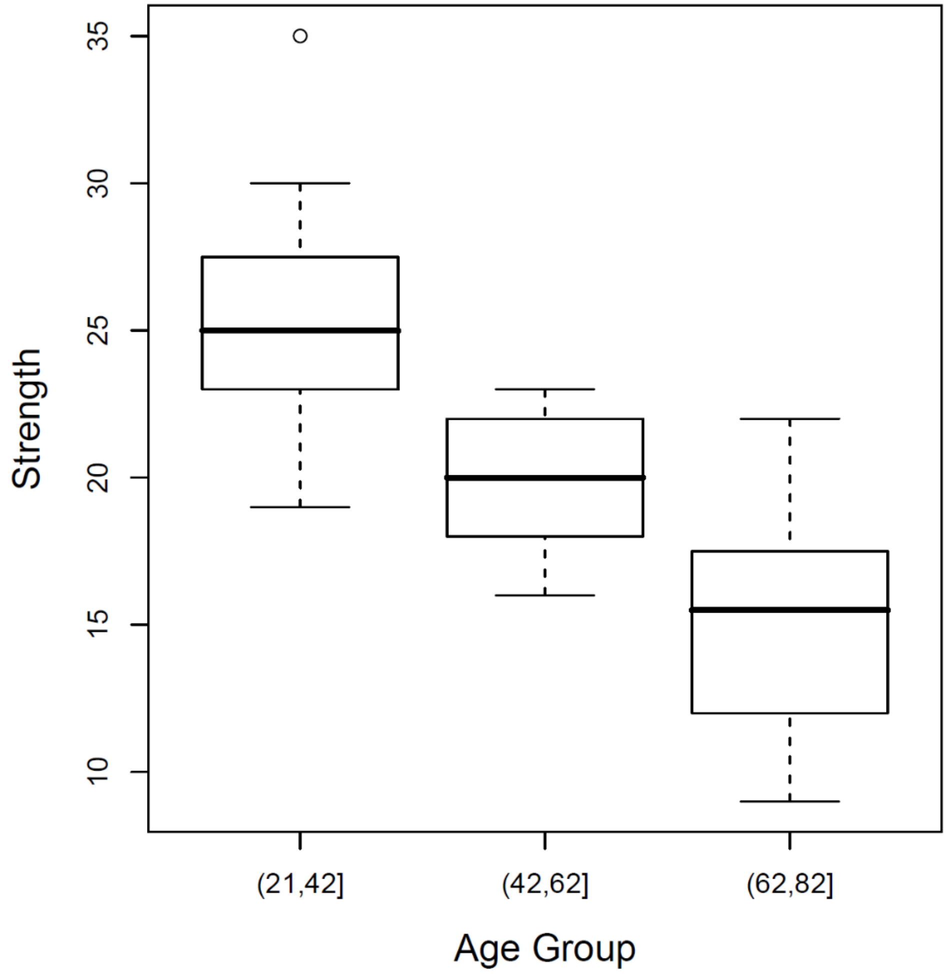
Cross-Tabulation

Subject ID	Age Group	Sex
GW	young	F
JA	middle	F
TJ	young	M
JMA	young	M
JMO	middle	F
JQA	old	F
AJ	old	F
MVB	young	M
WHH	old	F
JT	young	F
JKP	middle	M

Age Group / Sex	Female	Male	Total
young	2	3	5
middle	2	1	3
old	3	0	3
Total	7	4	11

Multivariate Graphical EDA

side-by-side boxplot



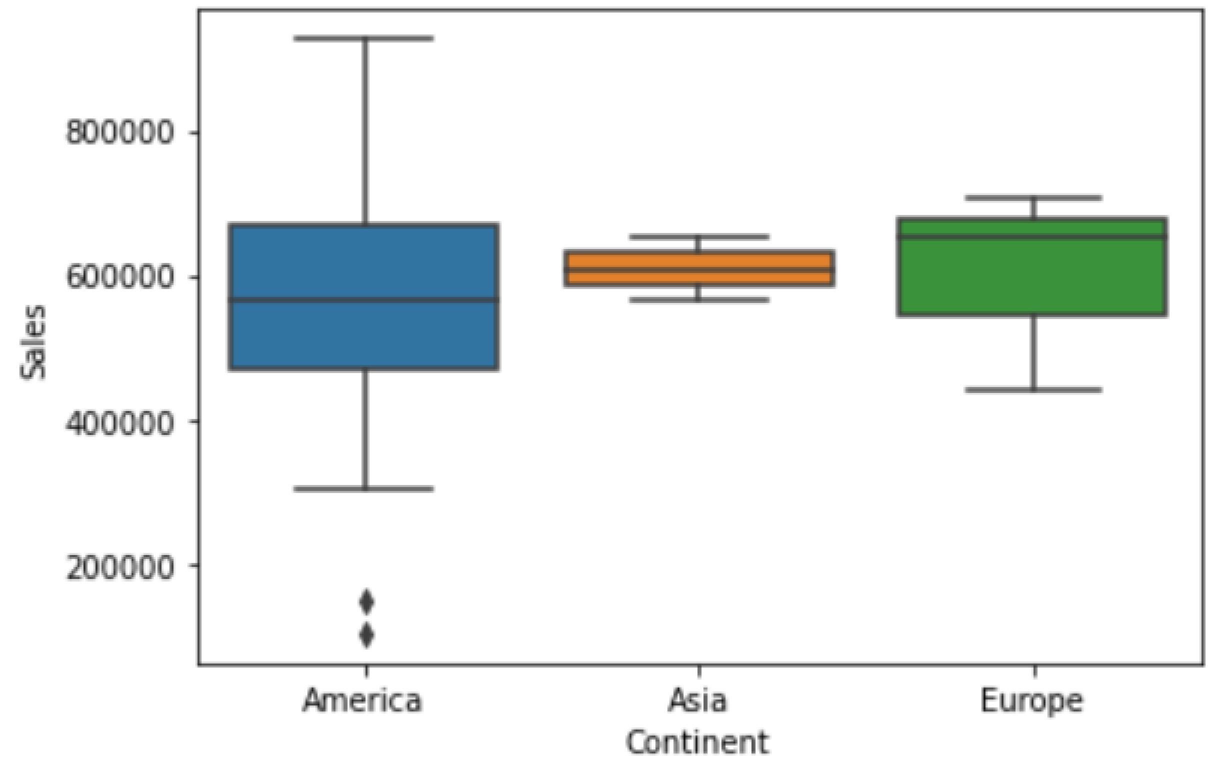
Side-by-side boxplots are good for examining the relationship between a categorical variable and a quantitative variable.

Multivariate Graphical EDA

side-by-side boxplot in Python

- Box Plots with seaborn library

```
sns.boxplot(x="Continent", y="Sales", data=df_operations)
```

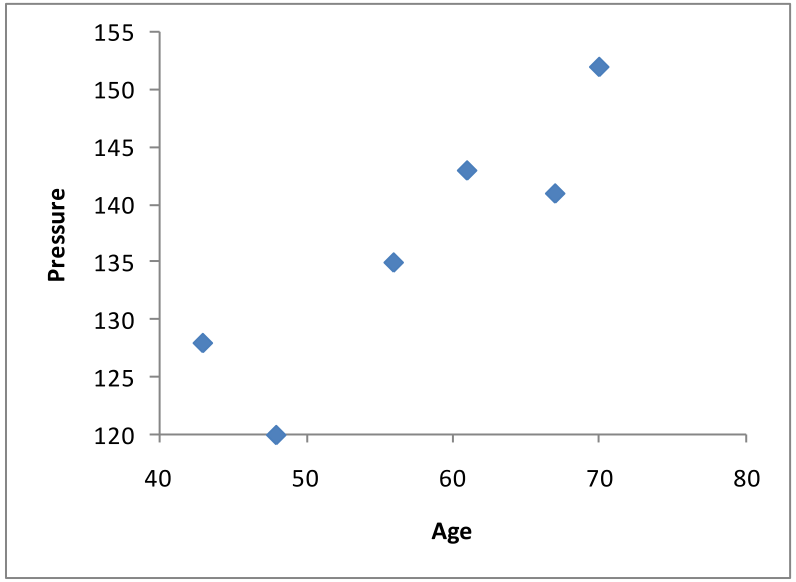


Multivariate Graphical EDA

Scatter plot

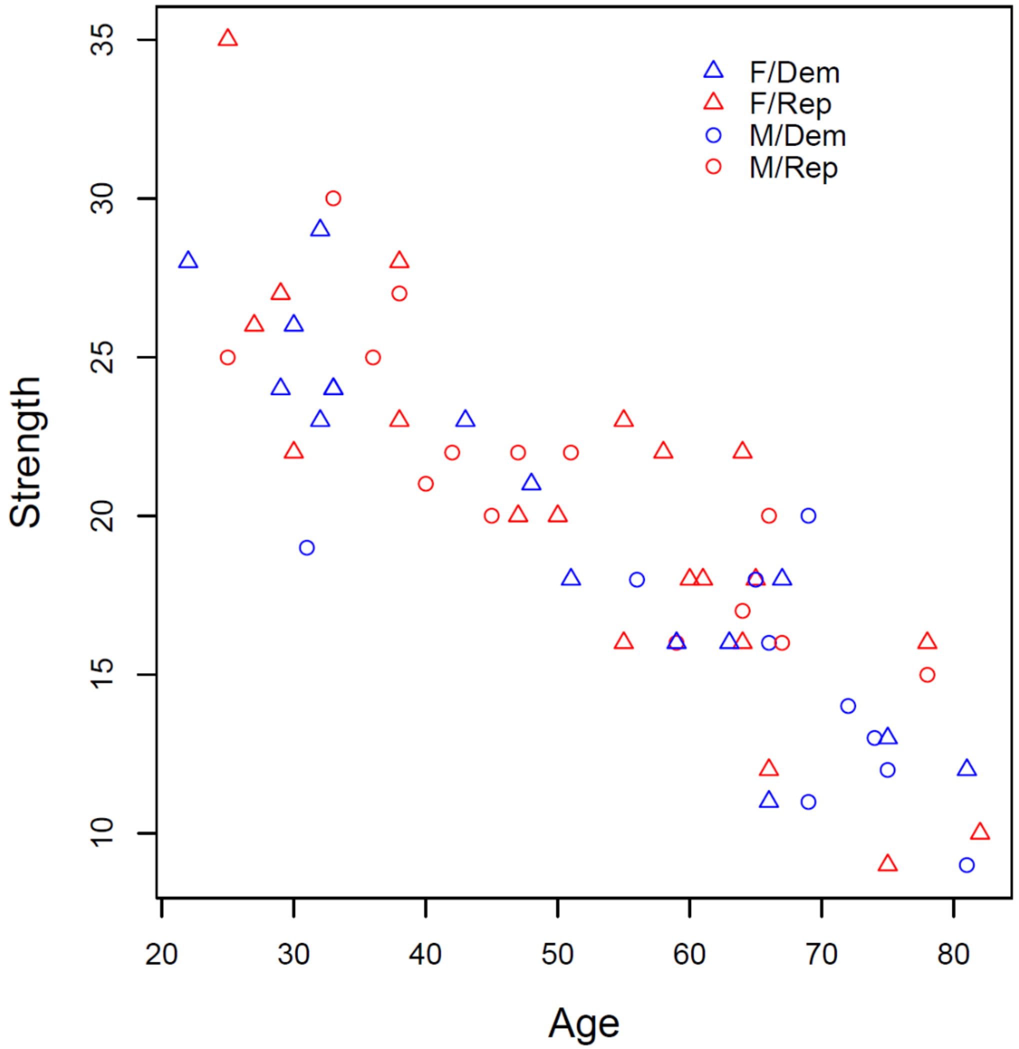
- A **scatter plot** is a graph of the ordered pairs (x,y) of numbers consisting of the **independent variable** x and the **dependent variable** y.

Subject	Age x	Pressure y
A	43	128
B	48	120
C	56	135
D	61	143
E	67	141
F	70	152



Multivariate Graphical EDA

Scatter plot with categorical data



- **One or two** additional **categorical variables** can be accommodated on the scatterplot
- Encoding the additional information in the **symbol type** and/or **color.**
- Here, colors code political party and gender

Summary EDA Types

	UNIVARIATE	MUTIVARIATE

Graphical	• Quantitative Variable:
	  • Histogram
	  • Boxplots
	  • Normal QQ-plot
	• Categorical Variable: Bar Charts
	• Time data – Line Plot
		• One Categorical and One Quantitative Variable:
		  • Side-by-side Boxplots
		• Two or More Categorical Variables:
		  • Grouped Bar Chart
		• Two or More Quantitative Variables:
		  • Scatterplot
		  • Correlation Heatmap
		  • Pairplot
		• Missing Data Detection

Non-Graphical	• Categorical Variable: tabular representation of frequency (*or relative frequency*)
	• Quantitative Variable:
	  • Location (*mean, median*)
	  • Spread (*IQR, std dev, range*)
	  • Modality (*mode*)
	  • Shape (*skewness, kurtosis*)
	  • Outliers
	• Missing Data Detection
		• One Categorical and One Quantitative Variable: ***standard univariate nongraphical statistics for the quantitative variables separately for each level of the categorical variable.***
		  • Mean
		  • Median
		  • Range and Spread measures
		• Two or More Quantitative Variables:
		  • Correlation
		  • Covariance
		  • Descriptive stat per
		• Missing Data Detection

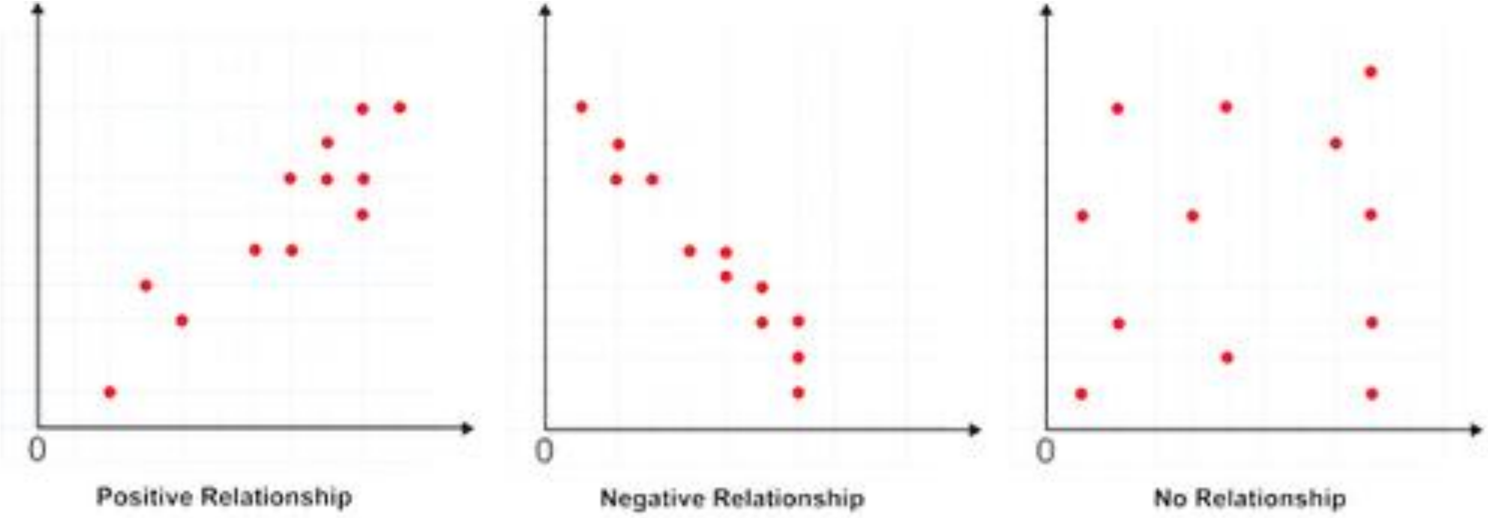
Relationship Between Variable

The study of the relationship between variable

Subject ID	Age	Strength	Age-50	Str-19	product
GW	38	20	-12	+1	-12
JA	62	15	+12	-4	-48
TJ	22	30	-28	+11	-308
JMA	38	21	-12	+2	-24
JMO	45	18	-5	-1	+5
JQA	69	12	+19	-7	-133
AJ	75	14	+25	-5	-125
MVB	38	28	-12	+9	-108
WHH	80	9	+30	-10	-300
JT	32	22	-18	+3	-54
JKP	51	20	+1	+1	+1
Total			0	0	-1106

Positive and negative relationship

- Simple relationships can also be positive or negative
- A **positive relationship** exists when both variables increase or decrease at the same time.
- In a **negative relationship**, as one variable increases, the other variable decreases, and vice versa.



Correlation Analysis

- Pearson's Correlation Coefficient.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

$$r = \frac{(20 * 4213.468) - (190.3998 * 361.6)}{\sqrt{[(20 * 2501.446) - 190.3998^2][(20 * 8568.5) - 361.6^2]}}$$

$$r = \frac{559552262.4}{23654.85706} = 0.651908$$

n = 20

Meat	Obesity			
x	y	xy	x^2	y^2
6.1244	4.5	27.5598	37.50828	20.25
8.7428	22.3	194.9644	76.43655	497.29
3.8961	26.6	103.6363	15.1796	707.56
11.0268	6.8	74.98224	121.5903	46.24
14.3202	19.1	273.5158	205.0681	364.81
19.2693	28.5	549.1751	371.3059	812.25
10.8165	20.9	226.0649	116.9967	436.81
11.6002	30.4	352.6461	134.5646	924.16
8.1099	21.9	177.6068	65.77048	479.61
11.9993	19.9	238.7861	143.9832	396.01
17.4941	32.1	561.5606	306.0435	1030.41
1.8407	3.4	6.25838	3.388176	11.56
13.1382	24.8	325.8274	172.6123	615.04
11.5636	26.6	307.5918	133.7168	707.56
5.6817	24.5	139.2017	32.28171	600.25
10.3435	22.4	231.6944	106.988	501.76
3.2849	8.2	26.93618	10.79057	67.24
21.1476	18.7	395.4601	447.221	349.69
190.3998	361.6	4213.468	2501.446	8568.5

Correlation Analysis in Python

- Correlation Coefficient with p value.

Country	Alcoholic Beverages	Animal Products	Animal fats	Aquatic Products, Other	Cereals - Excluding Beer	Eggs	Fish, Seafood	Fruits - Excluding Wine	Meat	Miscellaneous	...	Vegetable Oils	Vegetables	Obesity
Afghanistan	0.0	21.6397	6.2224	0.0	8.0353	0.6859	0.0327	0.4246	6.1244	0.0163	...	17.0831	0.3593	4.5
Albania	0.0	32.0002	3.4172	0.0	2.6734	1.6448	0.1445	0.6418	8.7428	0.0170	...	9.2443	0.6503	22.3
Algeria	0.0	14.4175	0.8972	0.0	4.2035	1.2171	0.2008	0.5772	3.8961	0.0439	...	27.3606	0.5145	26.6
Angola	0.0	15.3041	1.3130	0.0	6.5545	0.1539	1.4155	0.3488	11.0268	0.0308	...	22.4638	0.1231	6.8
Antigua and Barbuda	0.0	27.7033	4.6686	0.0	3.2153	0.3872	1.5263	1.2177	14.3202	0.0898	...	14.4436	0.2469	19.1

```
from scipy import stats
```

```
pearson_coef, p_value = stats.pearsonr(df_fat['Meat'], df_fat['Obesity'])
print("Pearson's correlation coefficient: ", pearson_coef, " p value: ", p_value)
```

```
Pearson's correlation coefficient:  0.219919023772617  p value:  0.00429447433848602
```

Correlation Analysis in Python

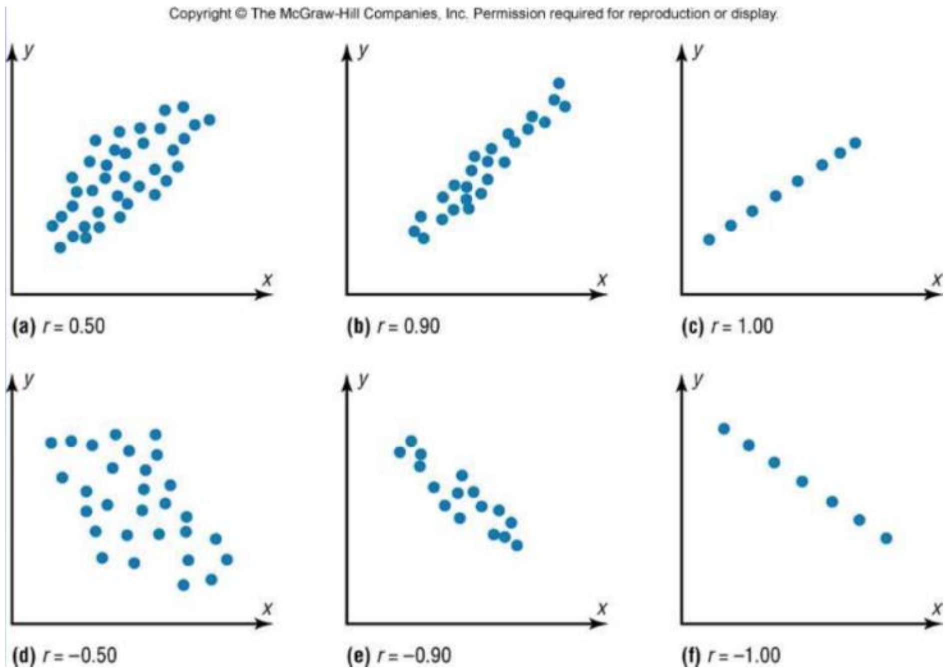
- Correlation Matrix.

```
df_fat[['Animal Products','Meat','Cereals - Excluding Beer','Sugar & Sweeteners','Obesity']].corr()
```

	Animal Products	Meat	Cereals - Excluding Beer	Sugar & Sweeteners	Obesity
Animal Products	1.000000	0.738702	-0.459064	-0.016549	0.417490
Meat	0.738702	1.000000	-0.270603	0.095534	0.219919
Cereals - Excluding Beer	-0.459064	-0.270603	1.000000	-0.003046	-0.488142
Sugar & Sweeteners	-0.016549	0.095534	-0.003046	1.000000	-0.163192
Obesity	0.417490	0.219919	-0.488142	-0.163192	1.000000

- Determination Coefficient:
  - $r^2 = (0.219919)^2 = 0.048364366561 = 0.4\%$

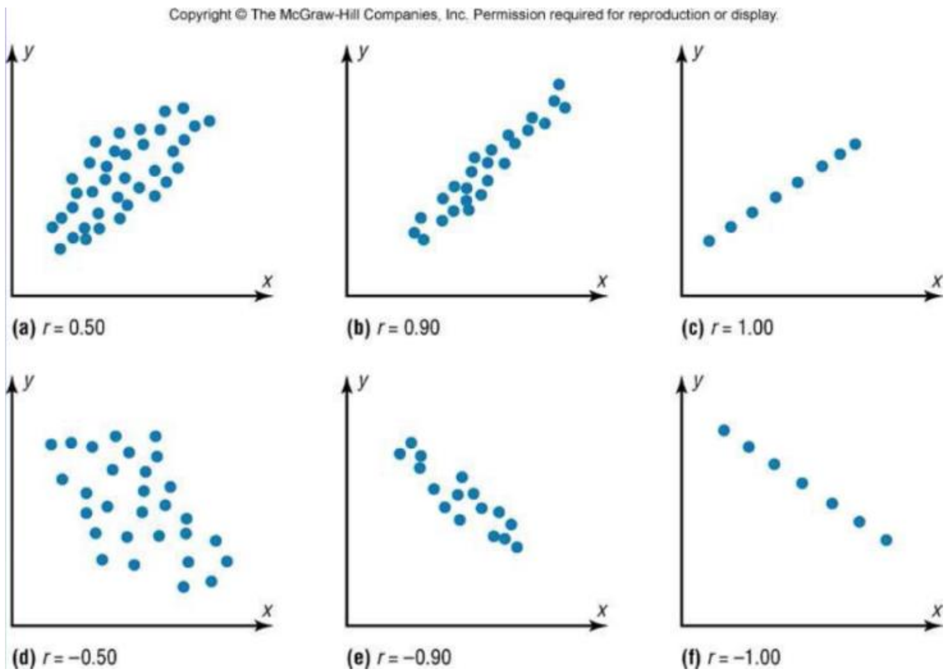
Correlation coefficient and scatter plot



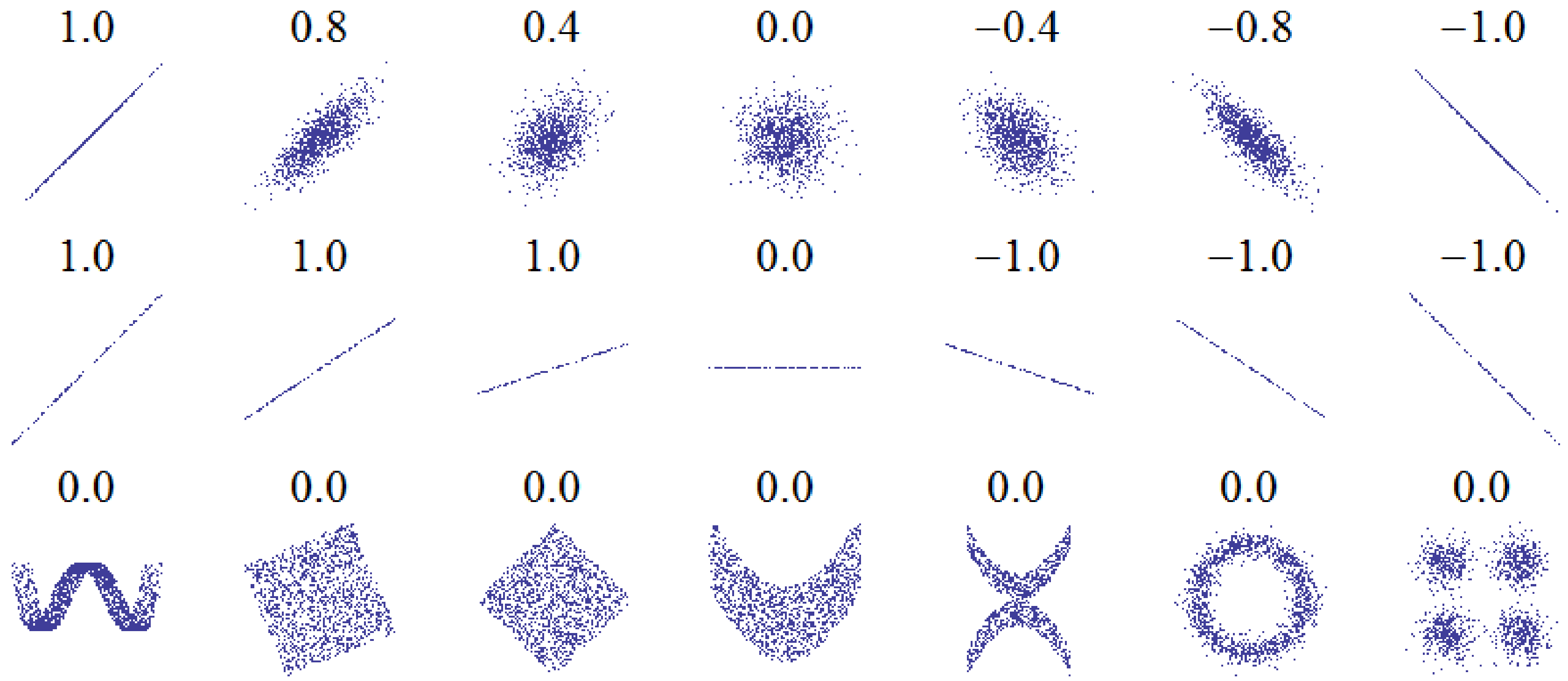
- The range of the correlation coefficient is from -1 to 1.
- If there is a strong positive linear relationship between the variables, the value of *r* will be close to 1.
- If there is a strong negative linear relationship, the value of *r* will be close to -1.
- When there is no linear relationship between the variables or only a weak one, the value of *r* will be close to 0

Strong correlation?

- A frequently asked question is: "what can it be said that there is a strong correlation between variables, and when is the correlation weak?"
- A reasonable rule of thumb is to say that the correlation is
  - weak if $0 \le |r| \le 0.5$
  - strong $0.8 \le |r| \le 1$
  - moderate otherwise

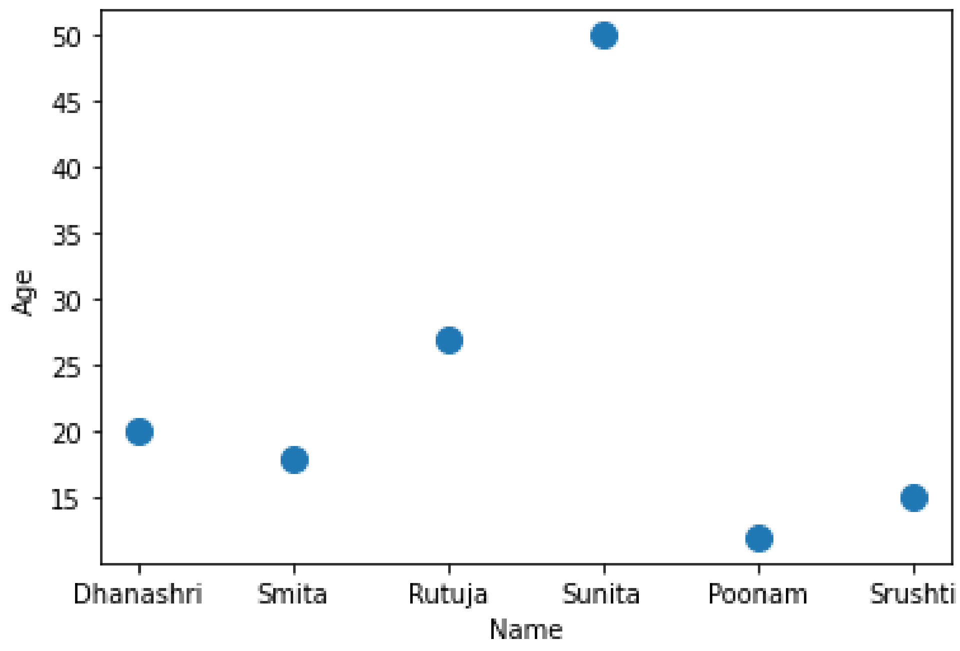


Correlation and shape

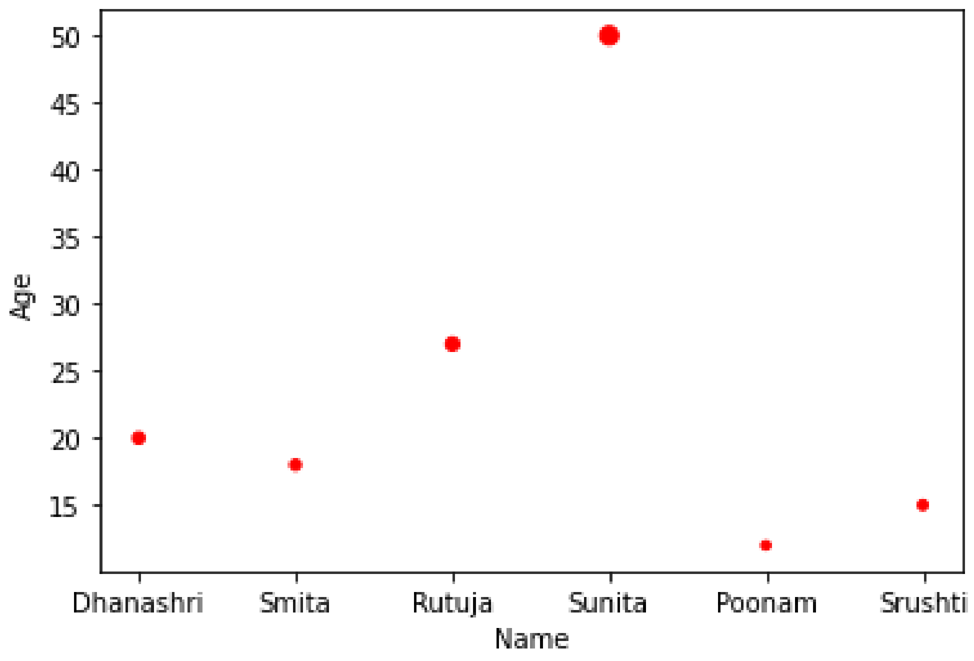


Scatter Plot in Python

```
df.plot.scatter(x = 'Name', y = 'Age', s = 100);
```




```
df.plot.scatter(x = 'Name', y = 'Age', s = 'Age', c = 'red');
```



Remarks on correlation

- Any of the following five relationships between variables can exist
  1. There is a direct cause-and-effect relationship between the variables. That is x causes y.
  2. There is a reverse cause-and-effect relationship between the variables. That is y causes x.
  3. The relationship between the variables may be caused by a third variable
  4. The may be a complexity of interrelationships among many variables
  5. The relationship may be coincidental

Summary

- You should always perform appropriate EDA before further analysis of your data. Perform whatever steps are necessary to become more familiar with your data
- Check for obvious mistakes
- Learn about variable distributions and relationships between variables.

Workshop

Example: cars dataset

Effects of features on the price

How does the brand affect the price?

What cars can be considered overpriced?

Price VS. popularity?

Car Features and MSRP

Includes features such as make, model, year, and engine type to predict price

Try all by yourself

Workshop (20 mins):

1. สร้างไฟล์ colab ใหม่ (ตั้งชื่อไฟล์ ว่า Workshop-EDA-DS6502-xxx)
2. จากชุดข้อมูลที่ได้รับไปให้ทำการทำความสะอาดข้อมูล และ ทำ EDA
3. ทำสรุปอธิบายกระบวนการทำความสะอาดข้อมูล และผลการวิเคราะห์จาก EDA ใส่ใน Colab โดยอธิบายที่มาที่ไปในการดำเนินการแต่ละ Step
4. ขออาสาสมัครตัวแทนนำเสนอ 3 ท่าน (ถ้า ไม่มีจะสุ่มเลือก) มาเล่าสิ่งที่ได้มาจากการทำ EDA ให้เพื่อนๆฟัง
5. คนที่เสร็จคนแรก ที่พร้อมจะนำเสนอ มีรางวัลพิเศษให้

Post-test และส่งงาน (10 mins)

https://forms.office.com/r/dkmxNJZqq4

Follow us on

govbigdata

Twitter

Blockdit

YouTube

Government Big Data Institute (GBDi)

Line Official

@gbdi

GBDi

gbdi.depa.or.th