CMU Portugal Academy

CMU Portugal
Advanced Training Program

Foundations of Data Science

DAVID SEMEDO

RAFAEL FERREIRA

NOVA SCHOOL OF SCIENCE AND TECHNOLOGY

Carnegie Mellon Portugal · TÉCNICO LISBOA · Ciências ULisboa · NOVA School of Science & Technology

Email: df.semedo@fct.unl.pt

This Week's Topics

1. STATISTICS AND PROBABILITY

2. DATA LOADING, PREPARATION AND PROCESSING WITH PANDAS

3. DATA VISUALIZATION WITH PANDAS AND SEABORN

What will you accomplish today?

- Load and Analyze a Dataset
- Data Pre-processing
- Data Analysis - Plotting

Use-Case Datasets

- **Potential datasets:**
  - Cost of Living Index by Country – Link
  - Red Wine Quality - Link
  - House Rent Prediction – Link
  - Stellar Classification – Link
  - Sleep Efficiency – Link
  - Healthcare Diabetes Dataset - Link

CMU Portugal Academy

01

Statistics and Probability

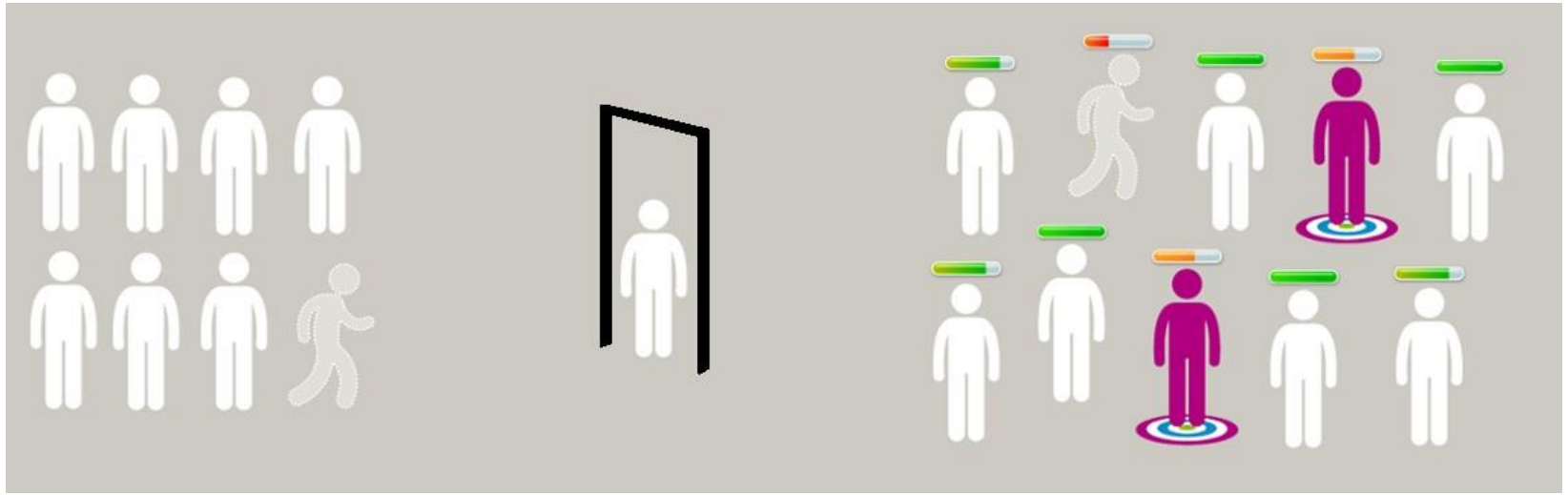
Carnegie Mellon Portugal · TÉCNICO LISBOA · Ciências ULisboa · NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Data Science and Uncertainty

- What is Data Science?
- Where does Uncertainty come from?

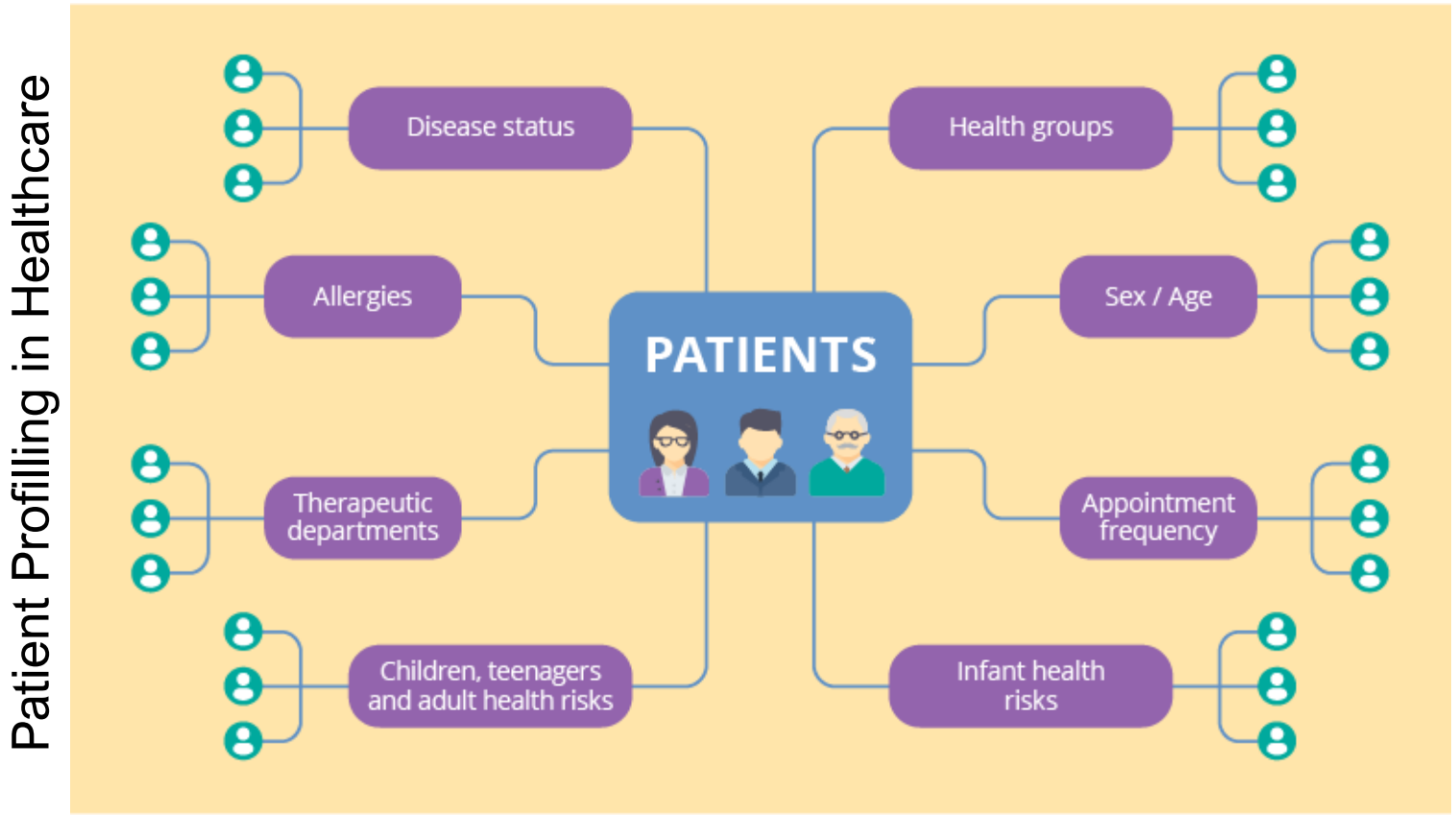
Data Science and Uncertainty

Churn prediction



Source: https://medium.com/@fatihfidan/customer-churn-analysis-55b6ebc8ca68

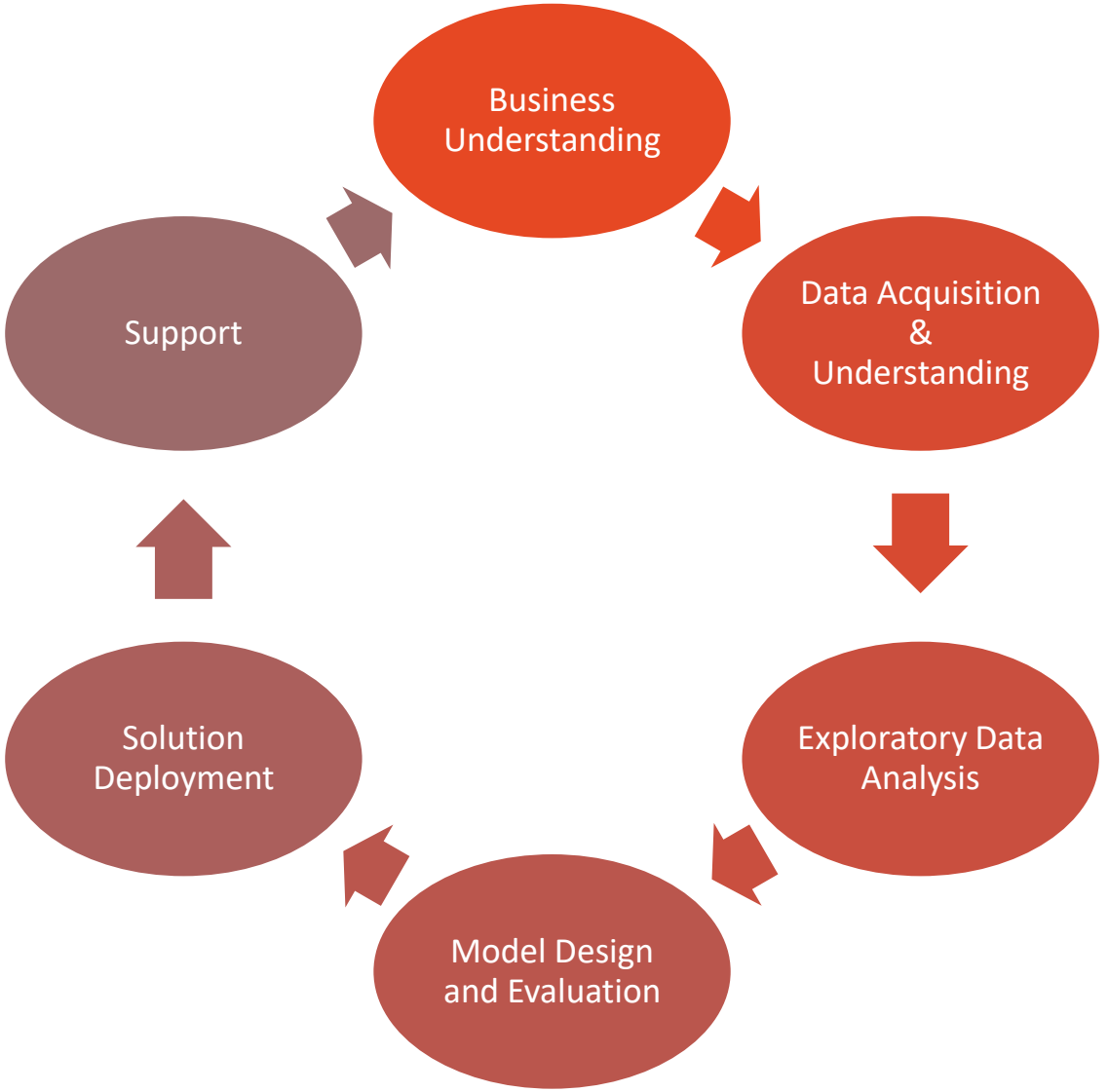
Data Science and Uncertainty



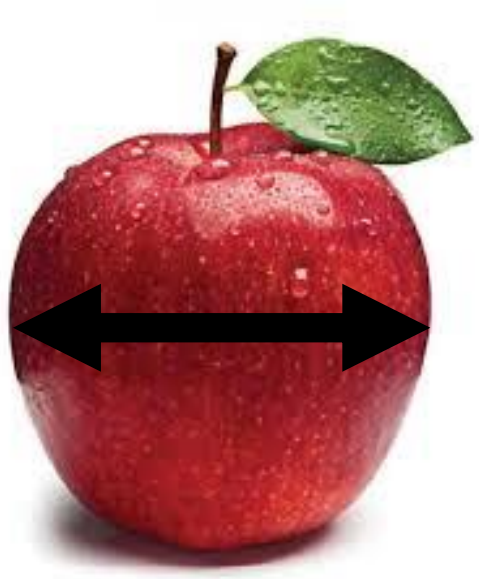
Source: https://www.scnsoft.com/healthcare/assembling-the-puzzle-of-a-patient-health-profile-in-crm

Data Science Lifecyle

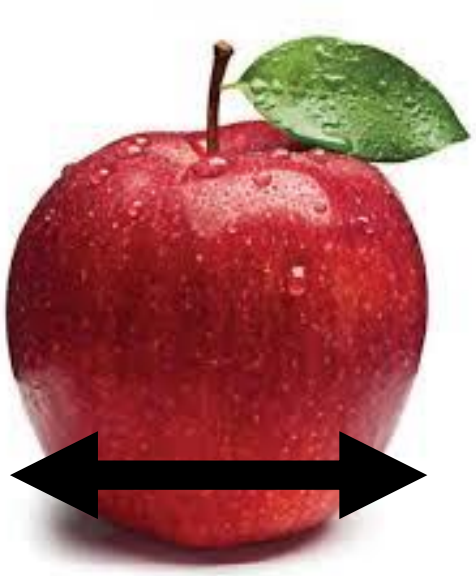
We want to make decisions, from data, under uncertainty!



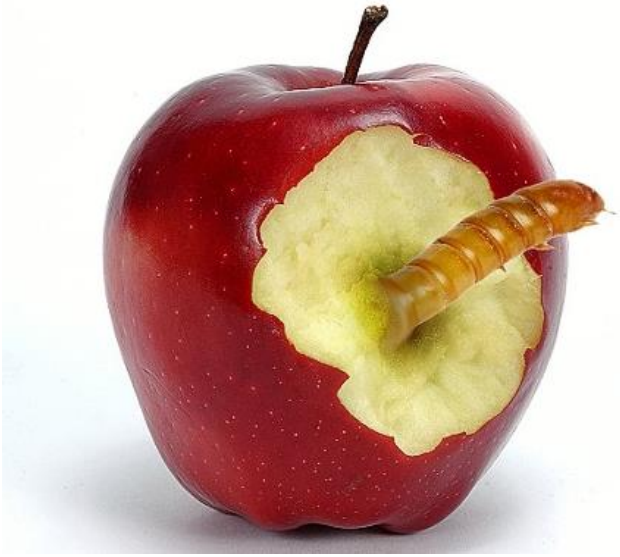
1) Descriptive statistics



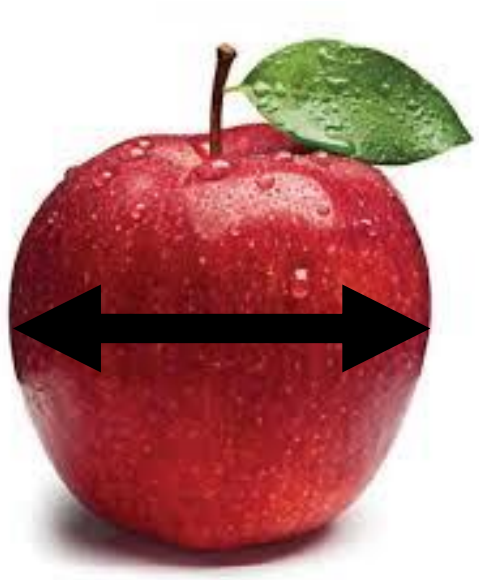
1) Descriptive statistics



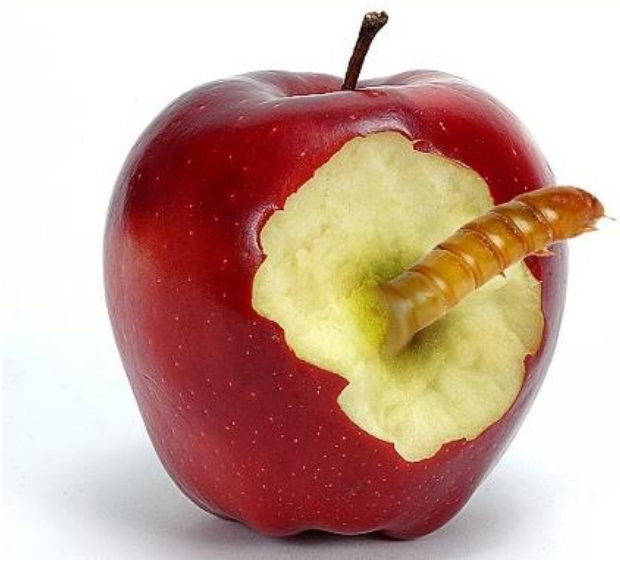
2) Exploratory



1) Descriptive statistics



2) Exploratory



3) Inferential statistics

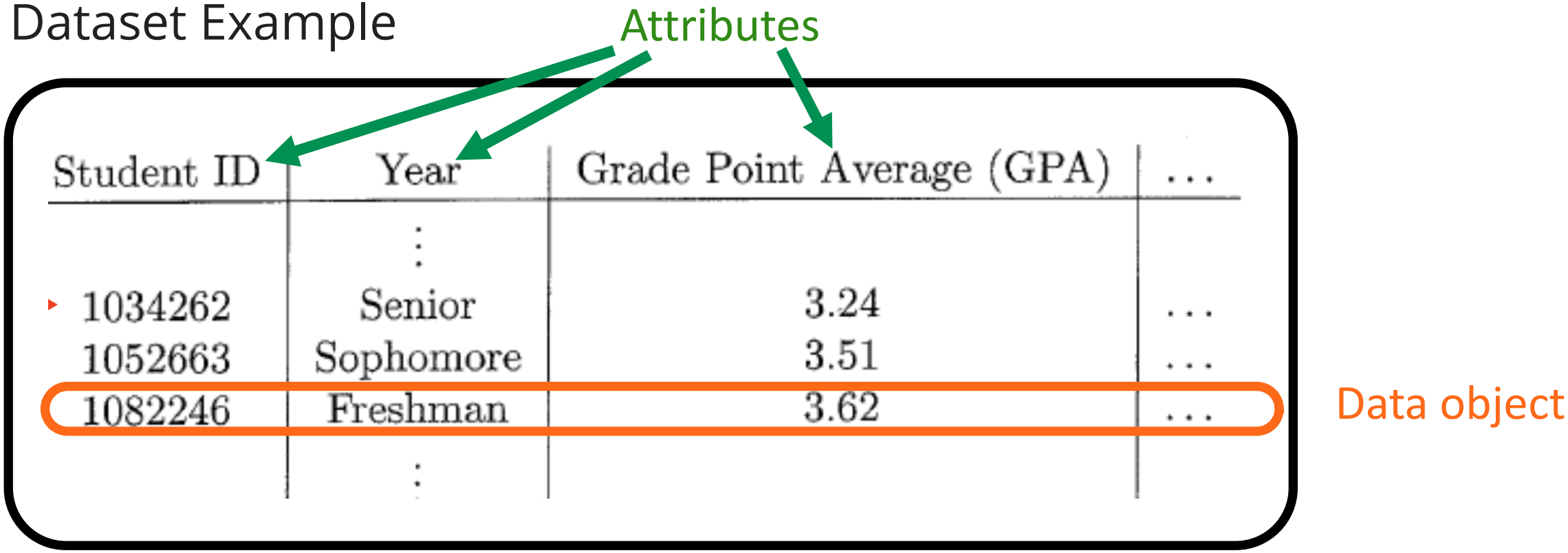
Descriptive Statistics

- Informational coefficients that summarize a data set
- Central tendencies vs. variability (spread)
- Data Distribution
- Univariate, Bivariate, Multivariate

Dataset Example

Student ID	Year	Grade Point Average (GPA)	...
	⋮		
1034262	Senior	3.24	...
1052663	Sophomore	3.51	...
1082246	Freshman	3.62	...
	⋮		

Dataset Example



Data object = record, individual, point, event, observation, vector, entity

Attribute = field, feature, variable, dimension, characteristic

Types of Variables: Categorical or Quantitative

Discrete set of categories

Humans' eye colors

Energy Rating

Categorical variables can be: nominal or ordinal

Humans' eye colors

Energy Rating

Types of Variables: Categorical or Quantitative

Humans' eye colors

Energy Rating

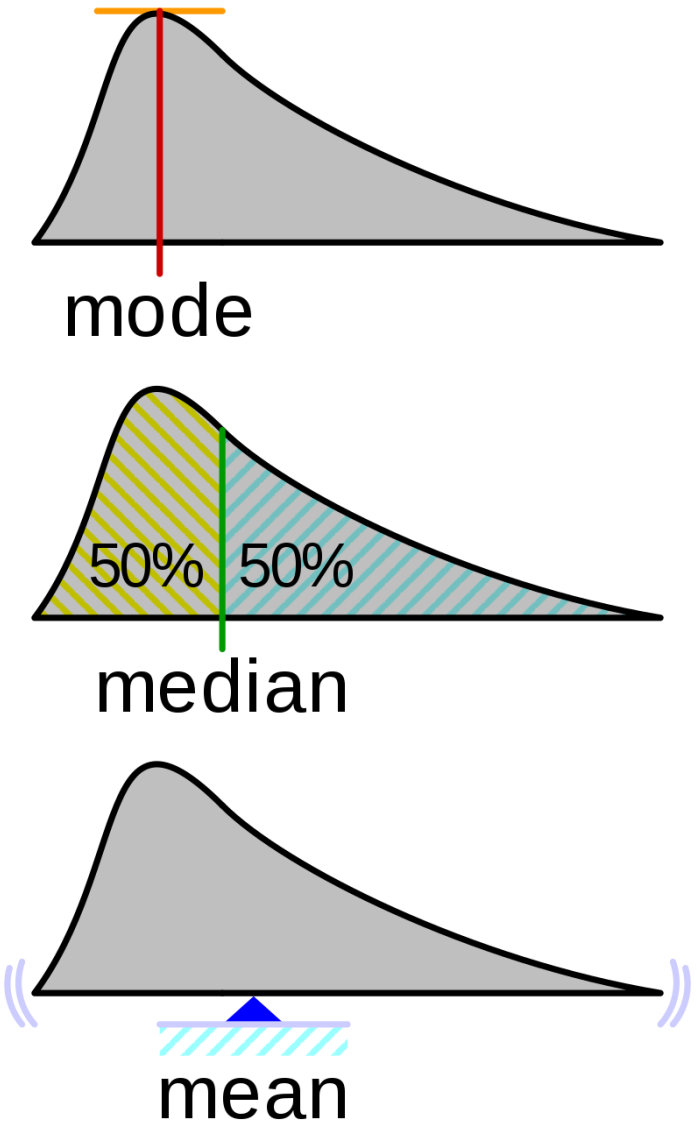
Take values for which arithmetic operations make sense

Quantitative variables can be in an interval or ratio

Humans’ eye colors

Energy Rating

Descriptive Statistics



Statistical Sample Moments

Mean – center

$$m_1 = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Variance – Spread

$$m_2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

Standardized

Skewness – Dispersion asymmetry

$$m_3 = \frac{1}{n\sigma^3}\sum_{i=1}^{n} (x_i - \bar{x})^3$$

Kurtosis – Tail “heaviness”

$$m_4 = \frac{1}{n\sigma^4}\sum_{i=1}^{n} (x_i - \bar{x})^4$$

Descriptive Statistics – Mean

$$\overline{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{1}{n}\sum x_i$$

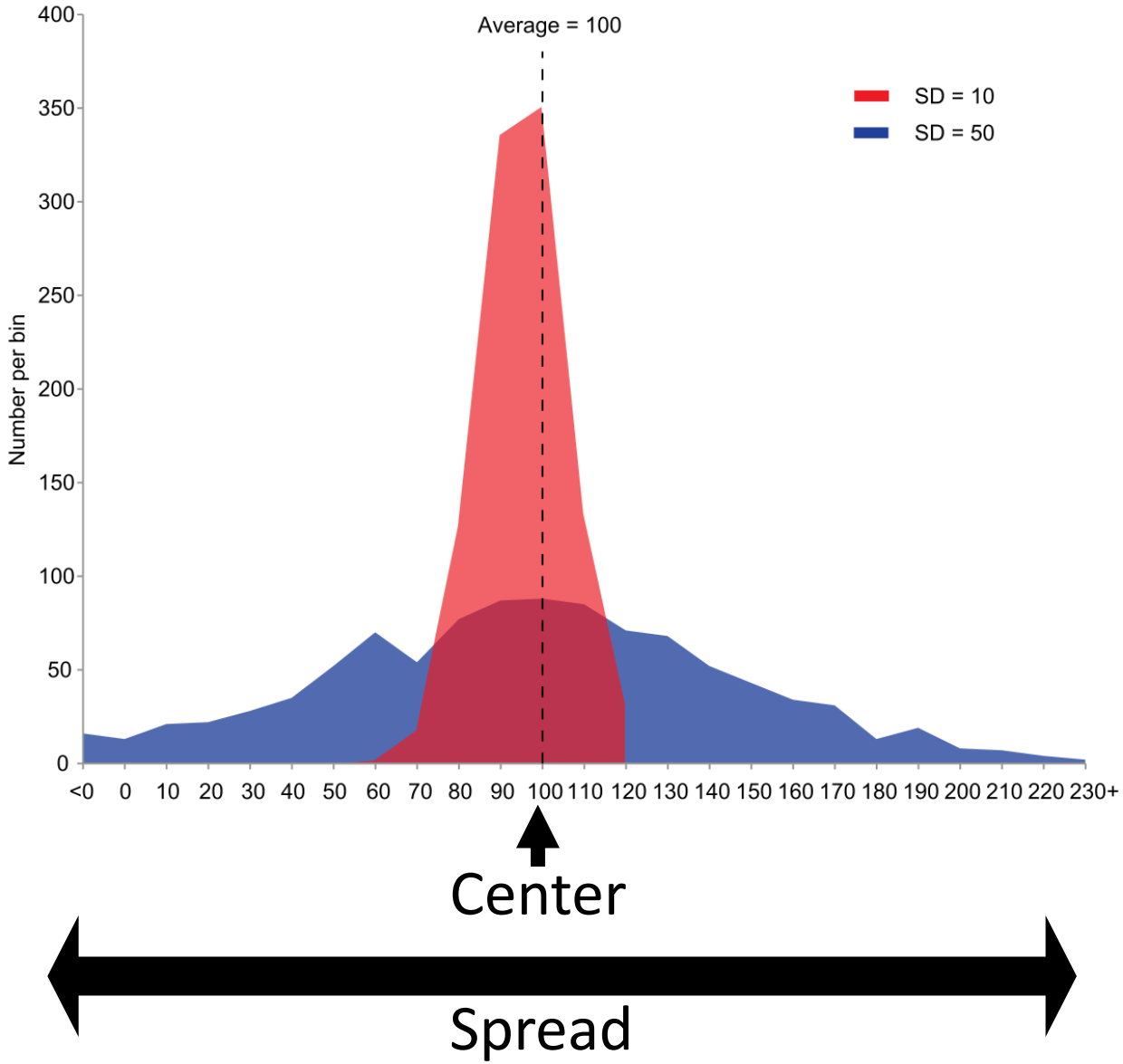
It gives an idea of the "center" of the distribution

Descriptive Statistics – Standard Deviation

Variance: $$s^2 = \frac{(x_1 - \overline{x})^2 + (x_2 - \overline{x})^2 + \cdots + (x_n - \overline{x})^2}{n - 1}$$

$$= \frac{1}{n-1}\sum(x_i - \overline{x})^2$$

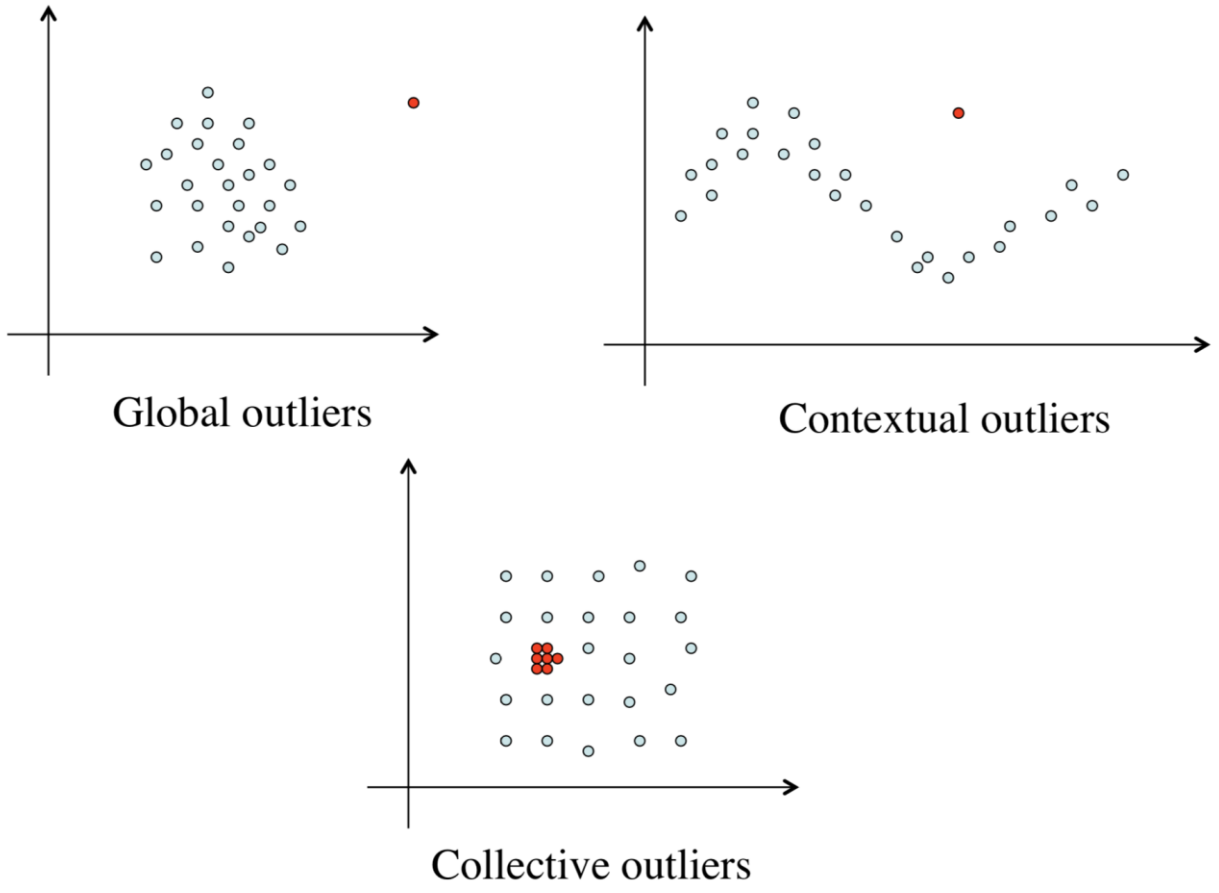
$$s = \sqrt{\frac{1}{n-1}\sum(x_i - \overline{x})^2}$$



https://en.wikipedia.org/wiki/Statistical_dispersion

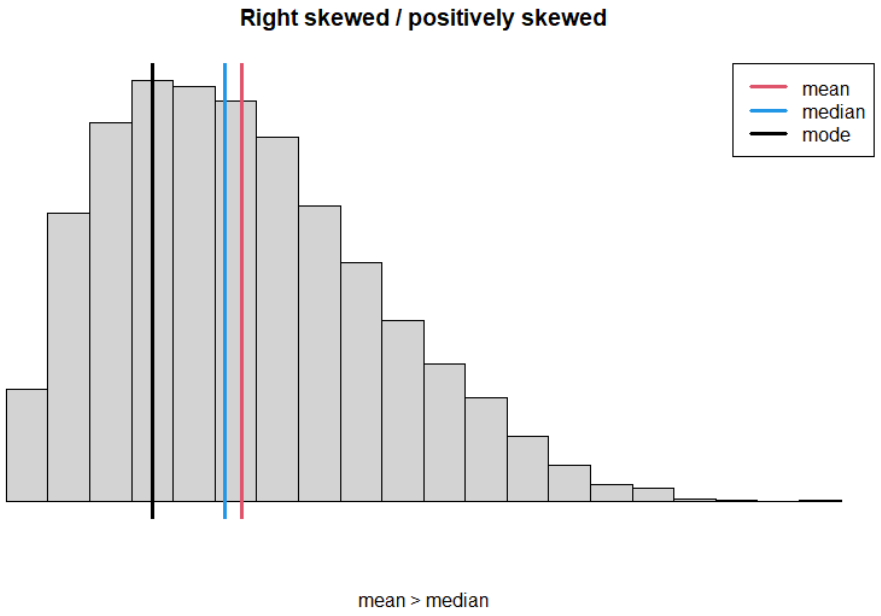
Mean and Variance – Do not tell the full story!

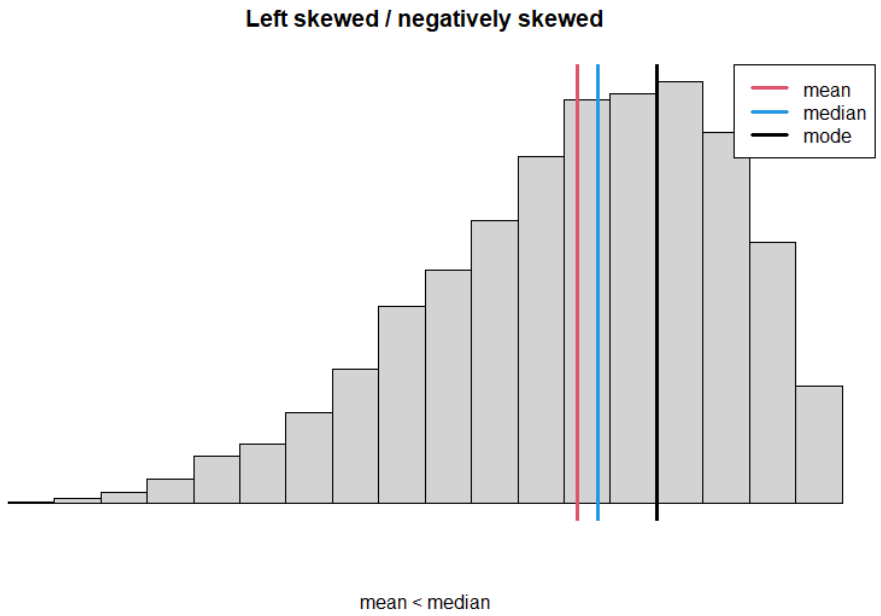
- Not robust to outliers



Mean and Variance – Do not tell the full story!

- Not robust to outliers
- Inadequate for skewed distributions

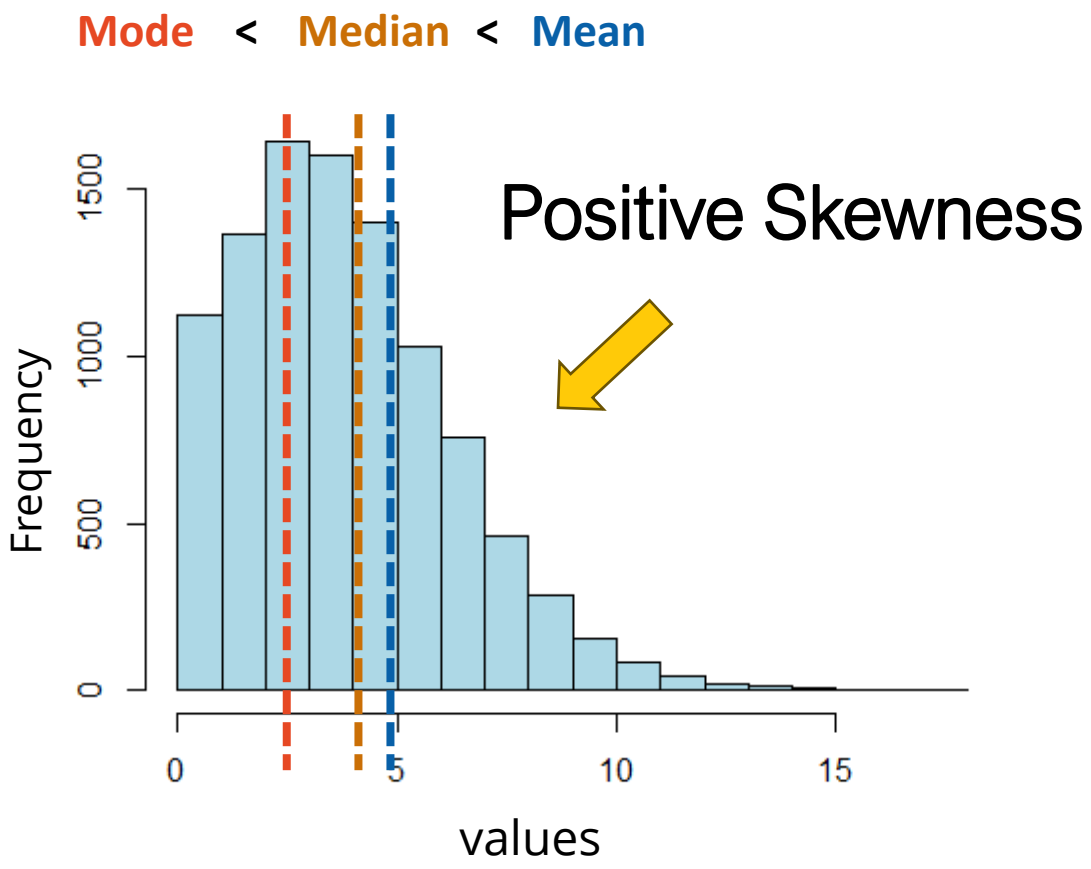




Skewness -3rd Statistical Sample Moment

Measures dispersion asymmetry

$$m_3 = \frac{1}{n\sigma^3}\sum_{i=1}^{n}(x_i - \bar{x})^3$$

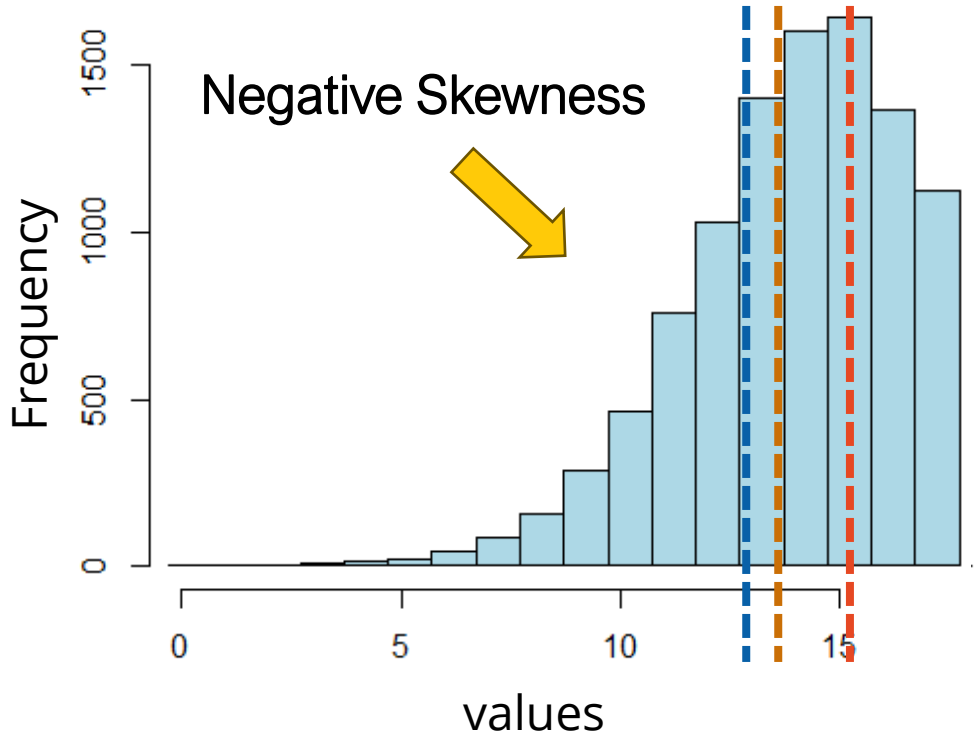


Skewness -3rd Statistical Sample Moment

Mean < Median < Mode

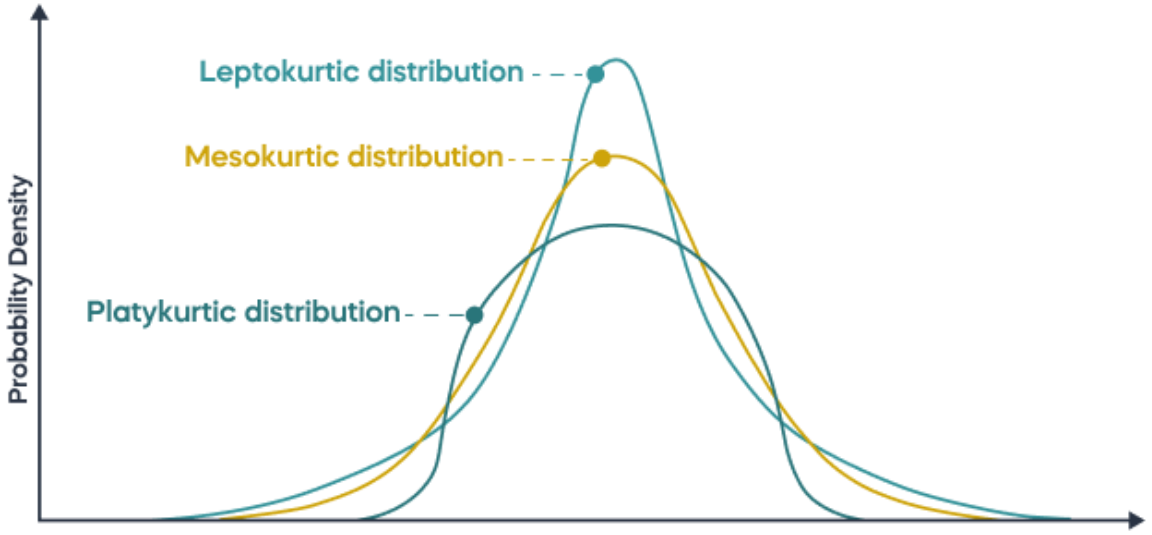
Measures dispersion asymmetry

$$m_3 = \frac{1}{n\sigma^3}\sum_{i=1}^{n}(x_i - \bar{x})^3$$



Kurtosis - 4th Statistical Sample Moment

Kurtosis – Tail "heaviness" – How often outliers occur.



Source: https://365datascience.com/calculators/kurtosis-calculator/

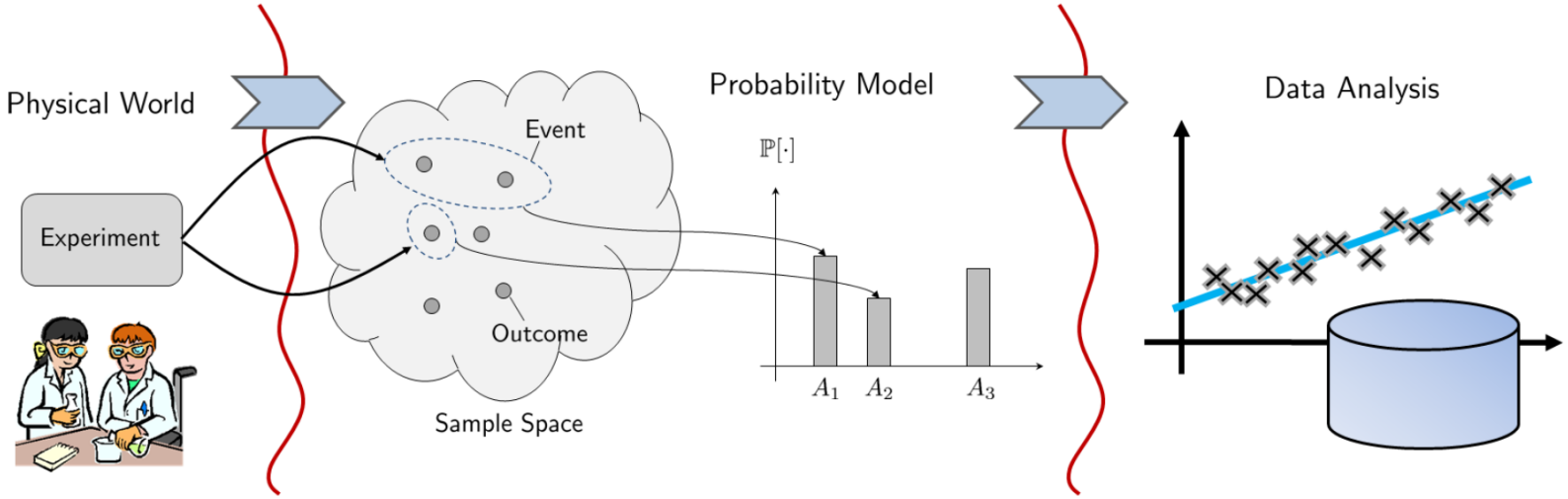
$$m_4 = \frac{1}{n\sigma^4}\sum_{i=1}^{n}(x_i - \bar{x})^4$$

Basic Probability Concepts

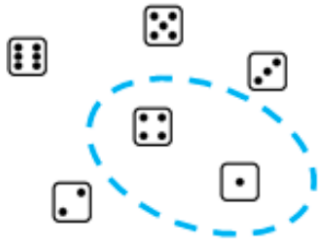
- **Frequentist**: The relative frequency of an outcome
- **Bayesian**: A subjective belief

Basic Probability Concepts

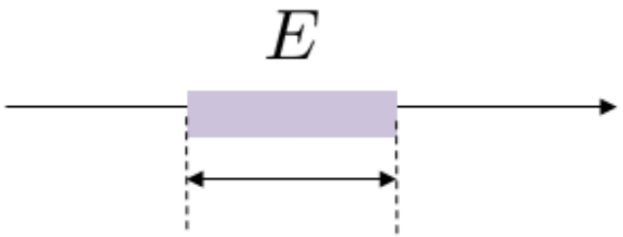
- **Sample space**: Set of all possible outcomes
- **Event space**: Collection of all possible events. One or a combination of outcomes
- **Probability model**: Measures the size of the event.



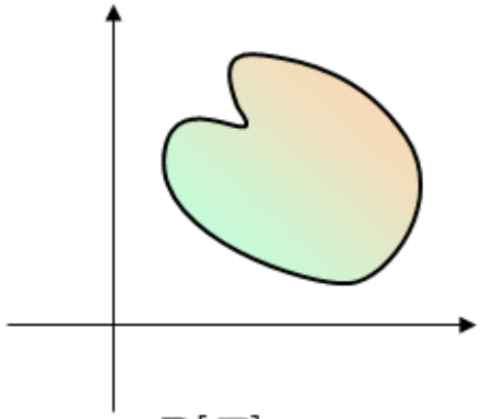
Basic Probability Concepts



$\mathbb{P}[E] = \text{count}$



$\mathbb{P}[E] = \text{length}$



$\mathbb{P}[E] = \text{area}$

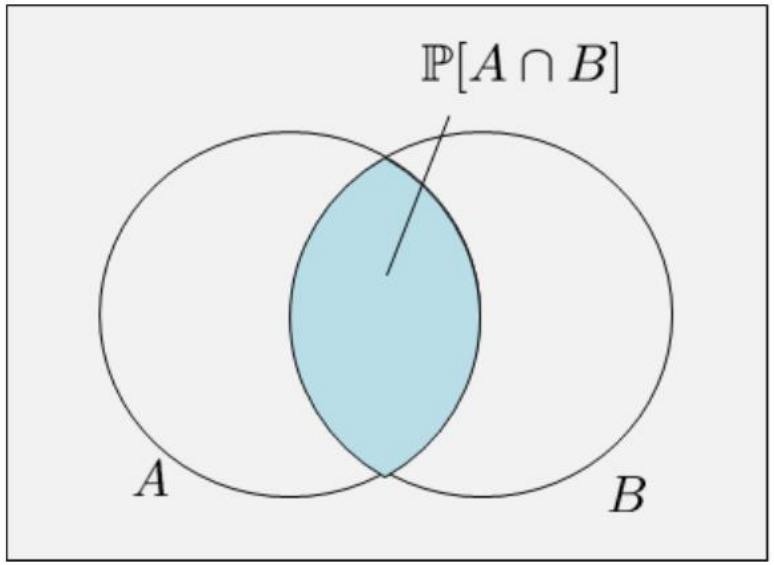
Probability Axioms

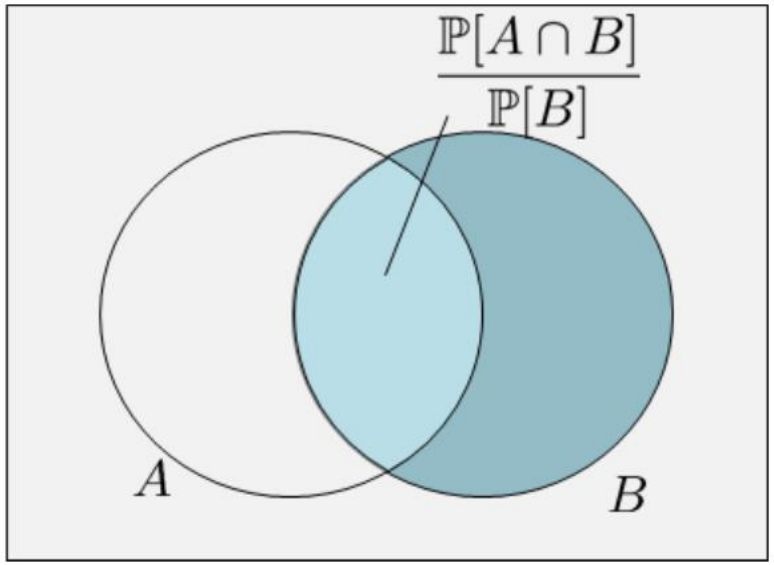
- Non-negativity: $\boldsymbol{P[E] \geq 0}$, for any possible event in $\boldsymbol{\Omega}$.
- Normalization: $\boldsymbol{P[\Omega] \geq 0}$
- Additivity: For any disjoint set of events S $= \{E_1, E_2, E_3\}$,

$$\boldsymbol{P[S] \geq \sum_{i=1}^{\infty} P[E_i]}$$

Conditional Probability – Bayes Theorem

- $\boldsymbol{P[A|\,B] = \frac{P[A \cap B]}{P[B]}}$, with $\boldsymbol{P[B] > 0}$





Probability Independence

- $\boldsymbol{P[A \cap B] = P[A] \cdot P[B]}$

- According to the conditional probability rule,

$$\boldsymbol{P[A|B] = \frac{P[A \cap B]}{P[B]} = \frac{P[A] \cdot P[B]}{P[B]} = P[A]}$$

Deriving Bayes Theorem

- $P[A|B] = \frac{P[A \cap B]}{P[B]}$ and $P[B|A] = \frac{P[B \cap A]}{P[A]}$
- $P[A|B] \cdot P[B] = P[B|A] \cdot P[A]$
- Bayes Theorem: $P[A|B] = \frac{P[B|A] \cdot P[A]}{P[B]}$

Likelihood: How probable is **B**, if the hypothesis **A** is true?
Prior: How probable is the hypothesis **A**, before observing **B**?
Marginal: How probable is **B**, under all possible hypothesis A_i?
Posterior: How probable is the hypothesis, given the evidence **B**?

Likelihood Prior

Posterior

Marginal

$$P[B] = \sum_i P[B|A_i] \cdot P[A_i]$$

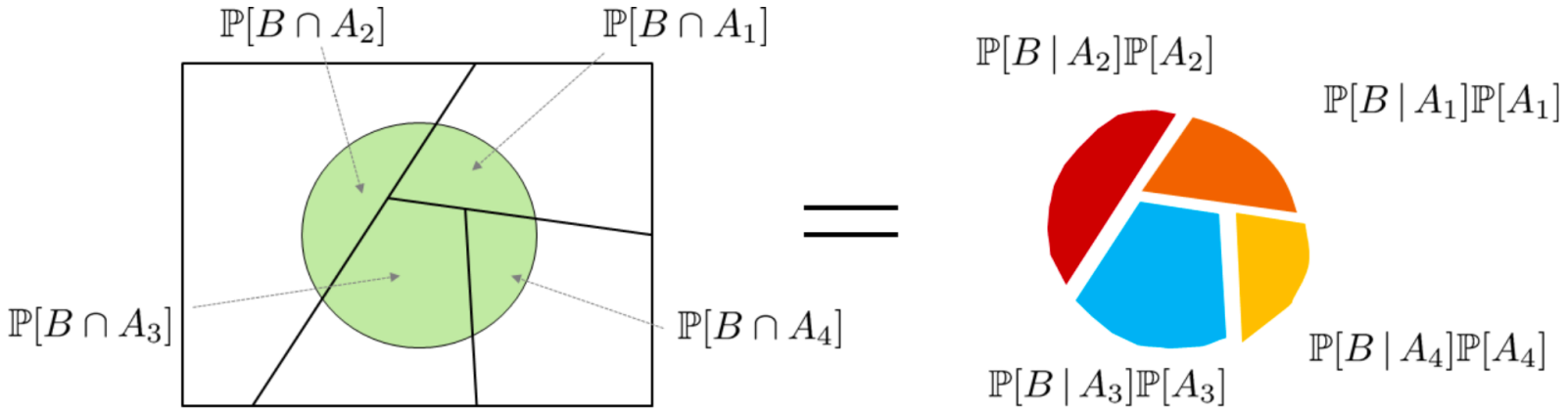
Law of Total Probability

- Bayes Theorem: $P[A|B] = \frac{P[B|A] \cdot P[A]}{P[B]}$

Likelihood, Prior, Posterior, Marginal

$$P[B] = \sum_i P[B|A_i] \cdot P[A_i]$$

A_i are disjoint events



Likelihood: How probable is **B**, if the hypothesis **A** is true?
Prior: How probable is the hypothesis **A**, before observing **B**?
Marginal: How probable is **B**, under all possible hypothesis A_i?
Posterior: How probable is the hypothesis, given the evidence **B**?

Law of Total Probability

- Bayes Theorem: $P[A|B] = \frac{P[B|A] \cdot P[A]}{P[B]}$

Likelihood, Prior, Posterior, Marginal

Likelihood: How probable is **B**, if the hypothesis **A** is true?
Prior: How probable is the hypothesis **A**, before observing **B**?
Marginal: How probable is **B**, under all possible hypothesis A_i?
Posterior: How probable is the hypothesis, given the evidence **B**?

$$P[B] = \sum_i P[B|A_i] \cdot P[A_i]$$

A_i are disjoint events

$$P[A_j|B] = \frac{P[B|A_j] \cdot P[A_j]}{\sum_i P[B|A_i] \cdot P[A_i]}$$

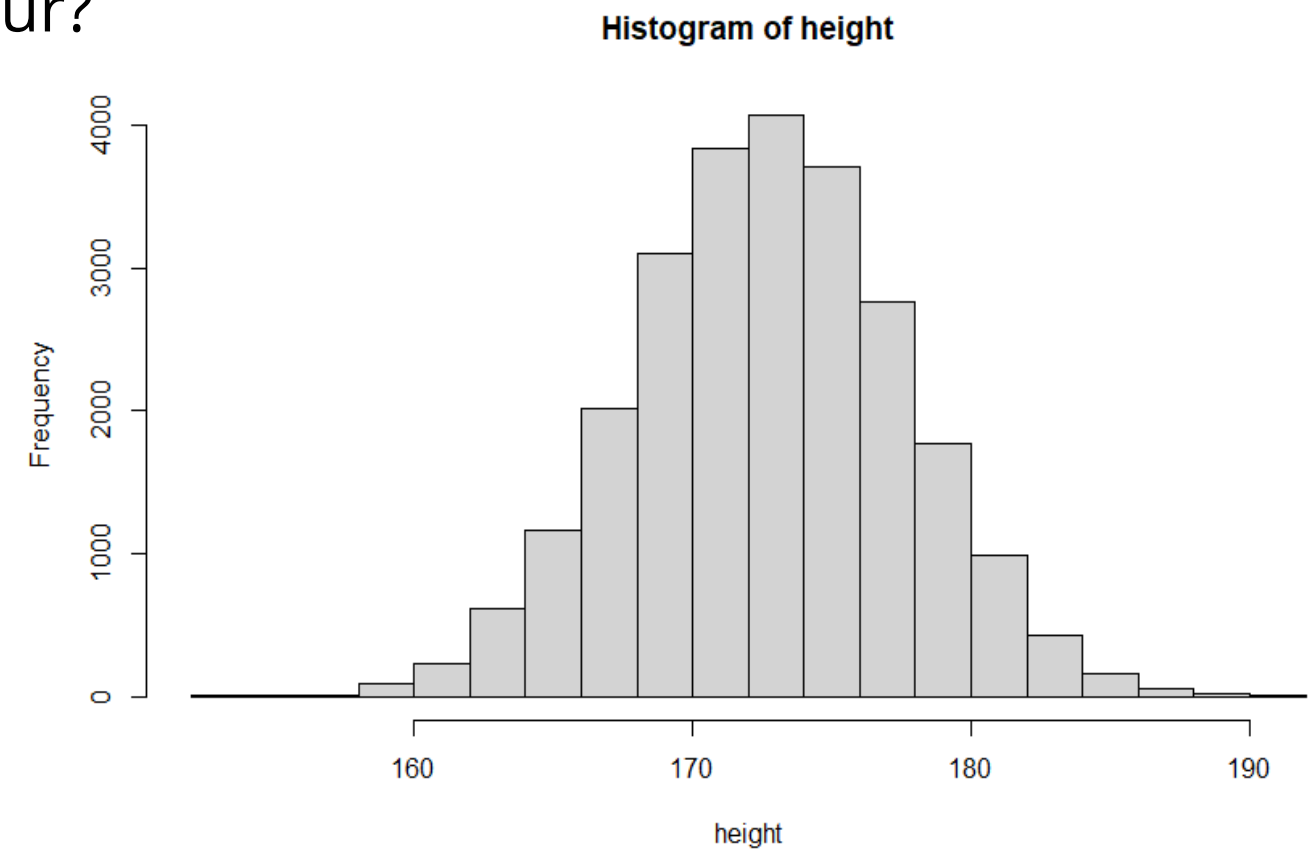
Decomposing an event into smaller events

Distributions

How often do certain values occur?

Discrete
- Bernouli (coin flip)
- Binomial (n coin flips)
- Geometric
- Poisson

Continuous
- Uniform
- Exponential
- Gaussian/Normal

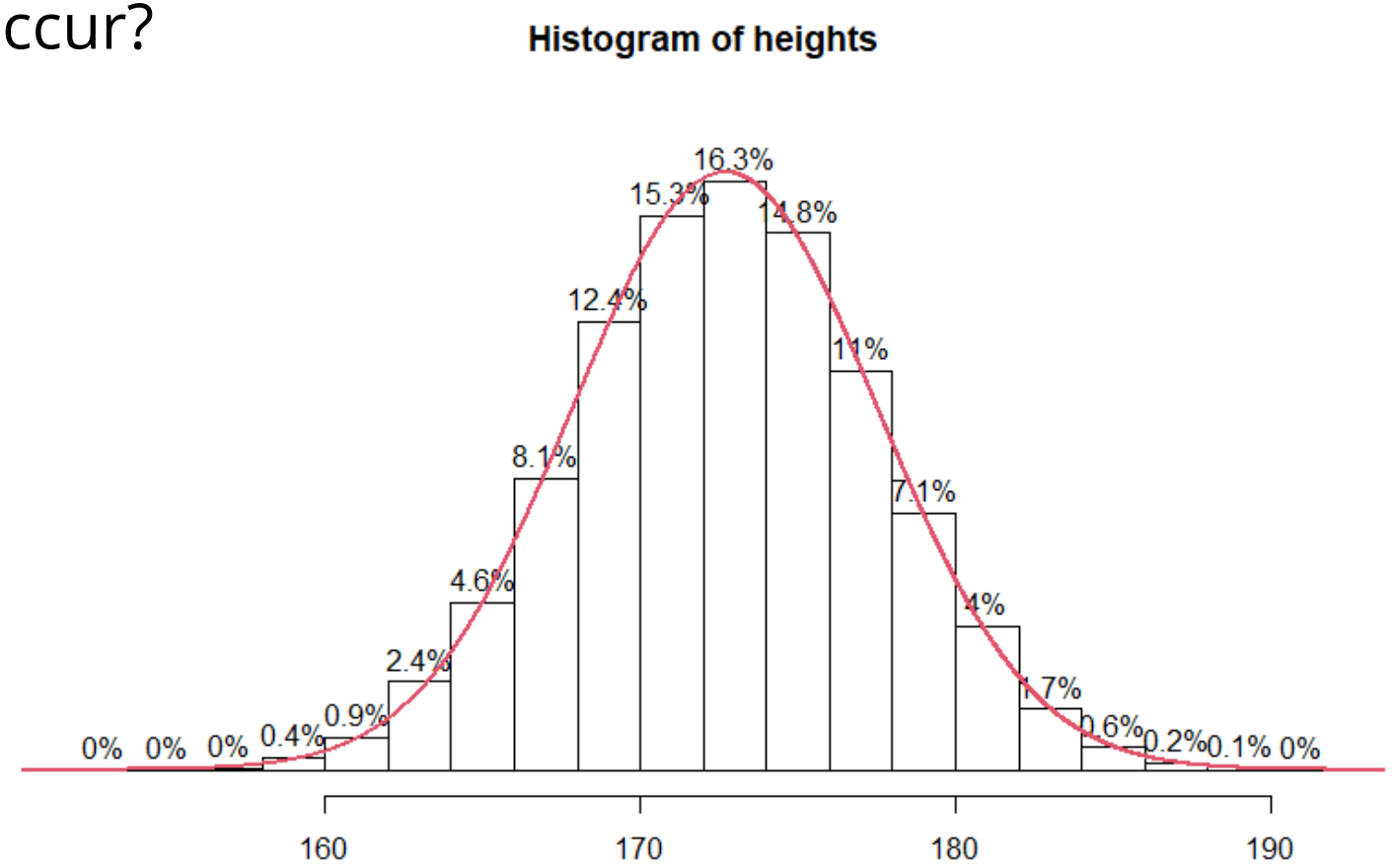


Distributions

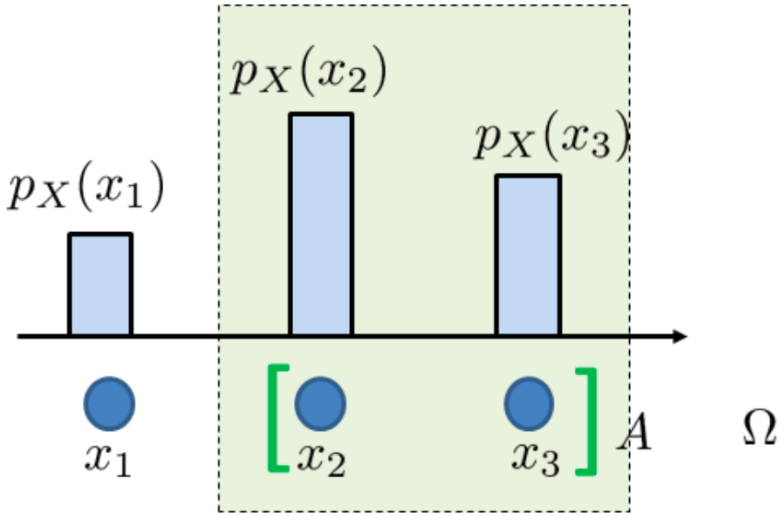
How often do certain values occur?

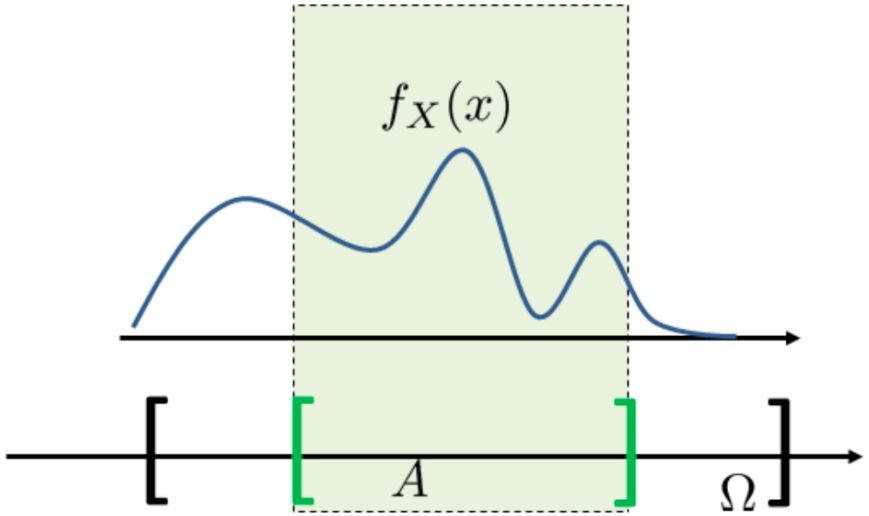
Discrete
- Bernouli (coin flip)
- Binomial (n coin flips)
- Geometric
- Poisson

Continuous
- Uniform
- Exponential
- Gaussian/Normal



Probability Mass vs Densify Functions

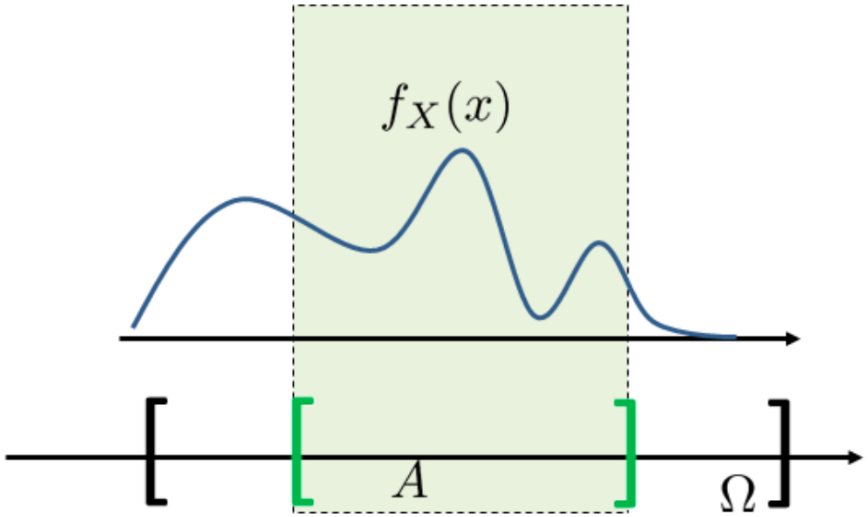




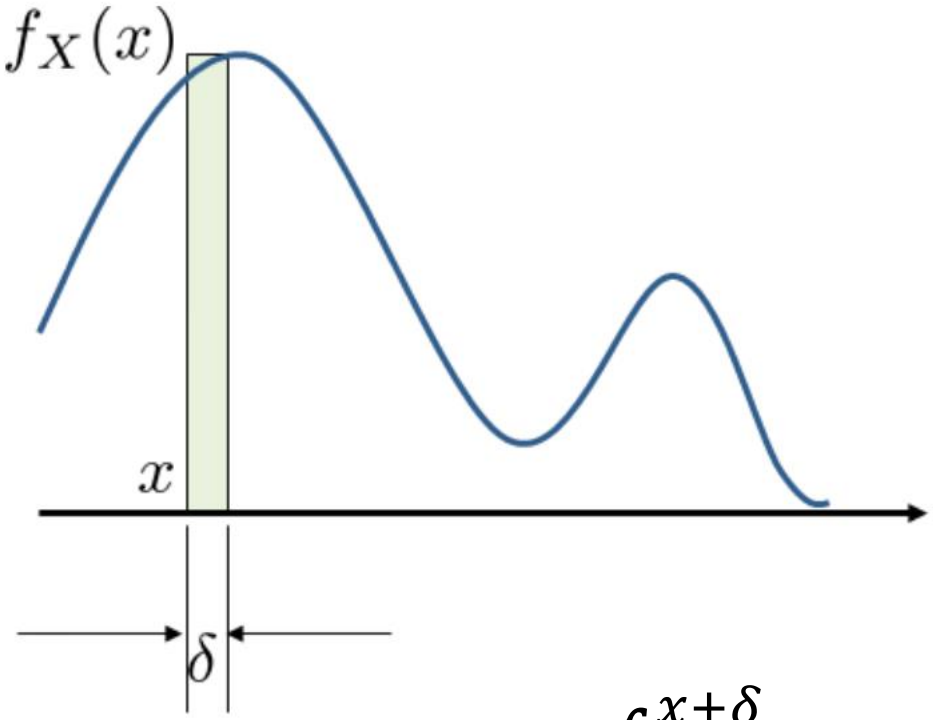
Probability Mass Function (PMF)
Discrete Random Variables

Probability Density Function (PDF)
Continuous Random Variables

From Density Functions to Probability



The probability is the area under the PDF

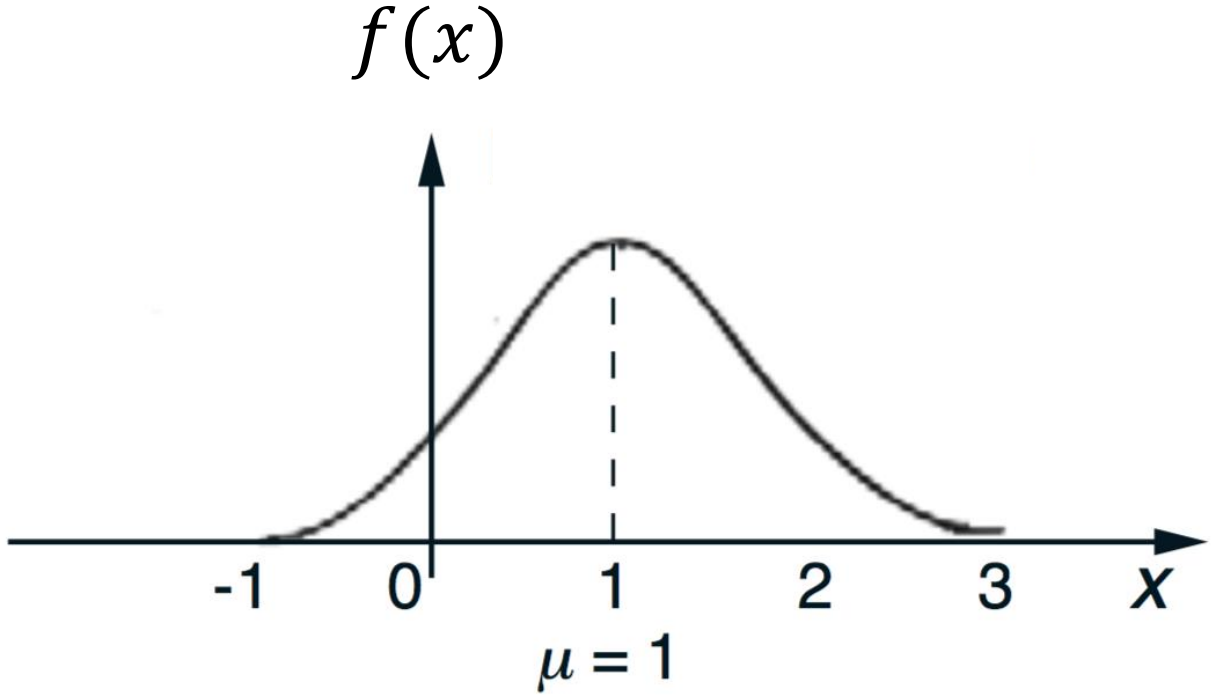


$$P[x \leq X \leq x + \delta] = \int_{x}^{x+\delta} f_X(x)$$

Gaussian/Normal Distribution $X \sim \mathcal{N}(\mu, \sigma^2)$

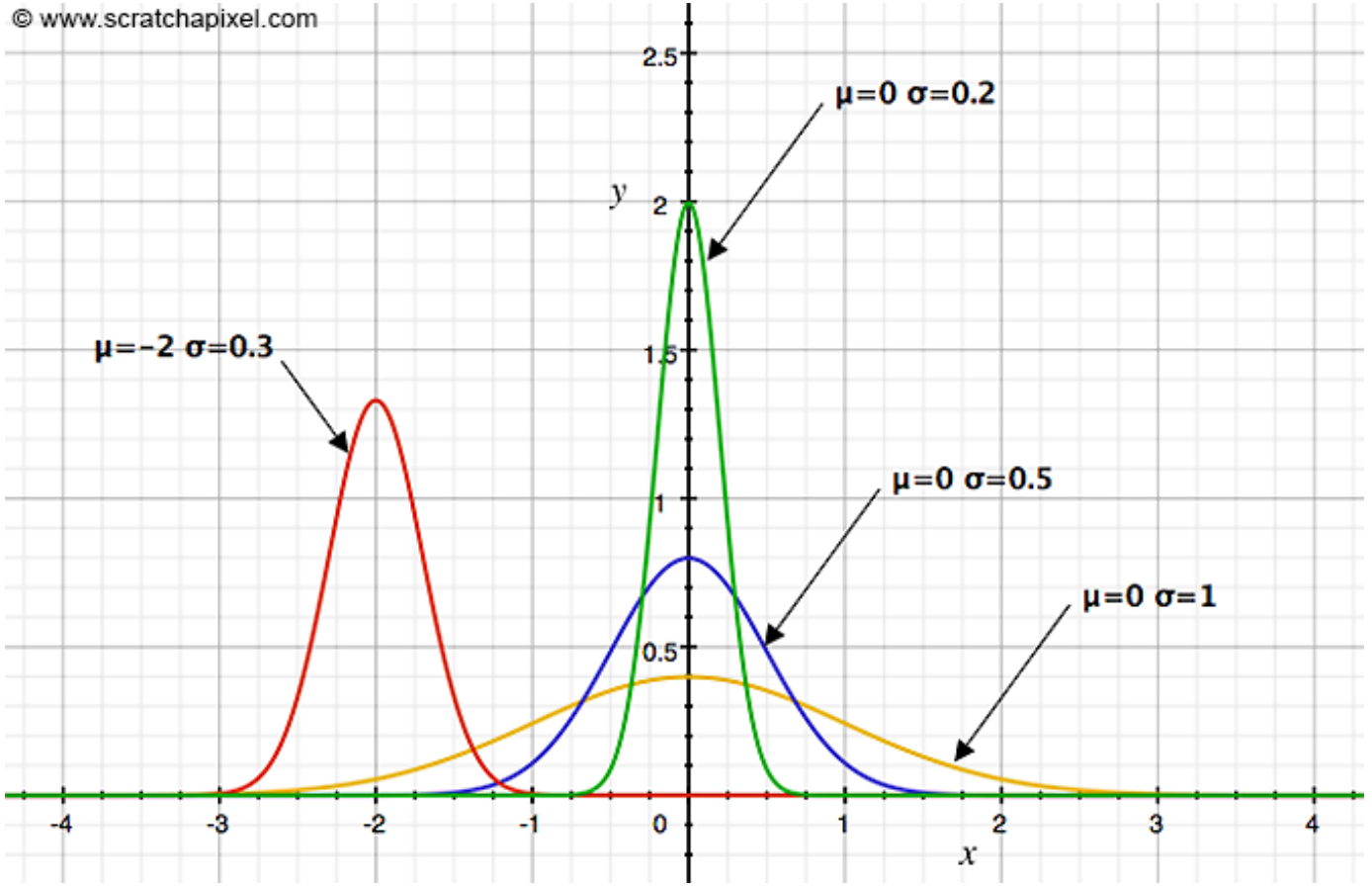
Probability Density Function $f(x)$

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$



Gaussian/Normal Distribution

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

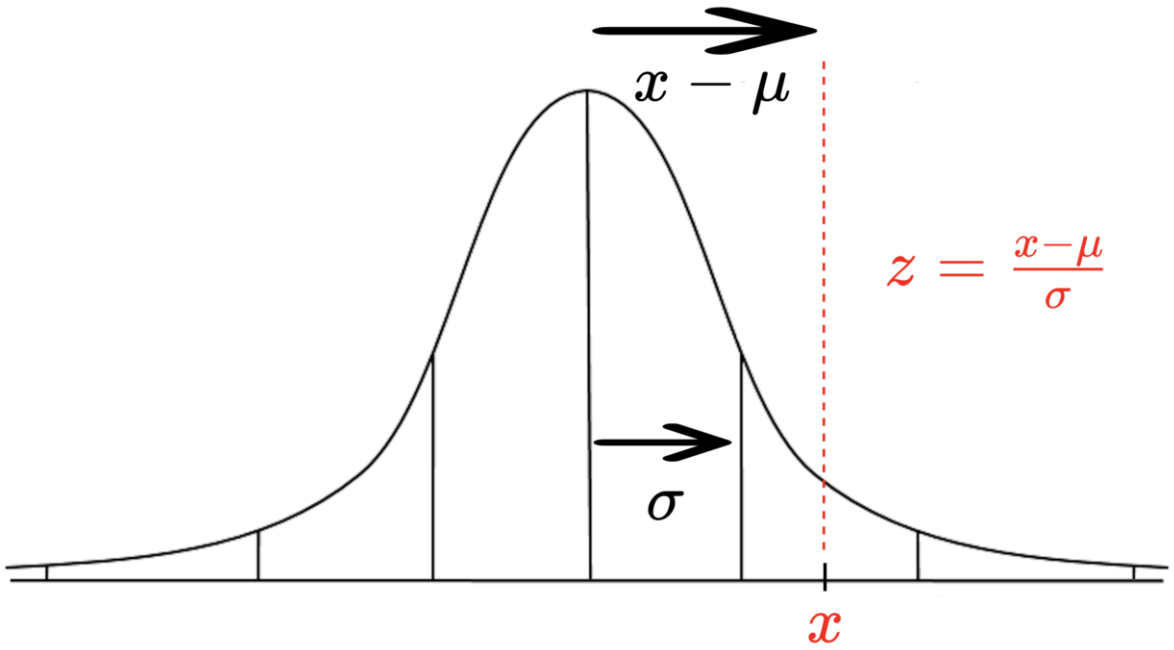


Gaussian/Normal Distribution - Standardization

Turn any normally distributed variable $X \sim \mathcal{N}(\mu, \sigma)$ into $\mathcal{N}(0,1)$ by standardizing it:

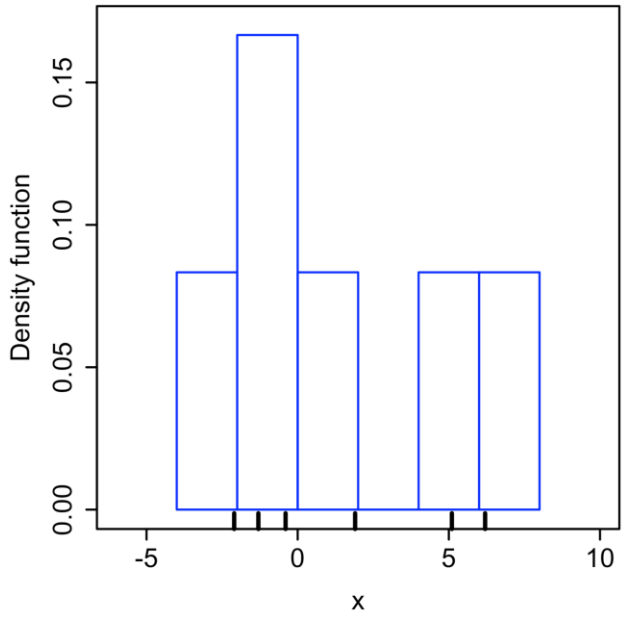
$$Z = \frac{X - \mu}{\sigma}$$

The z-score (standard score) is the number of standard deviations that an observed value x is away from the mean of a reference distribution

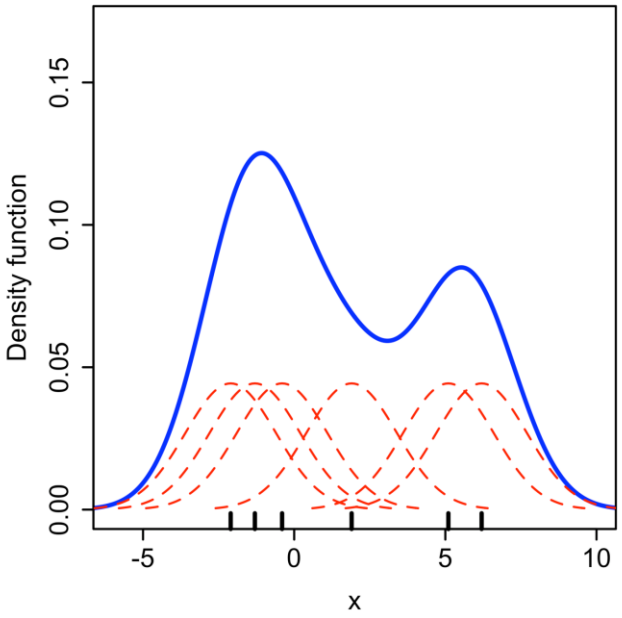


Fitting data to a distribution

Histogram

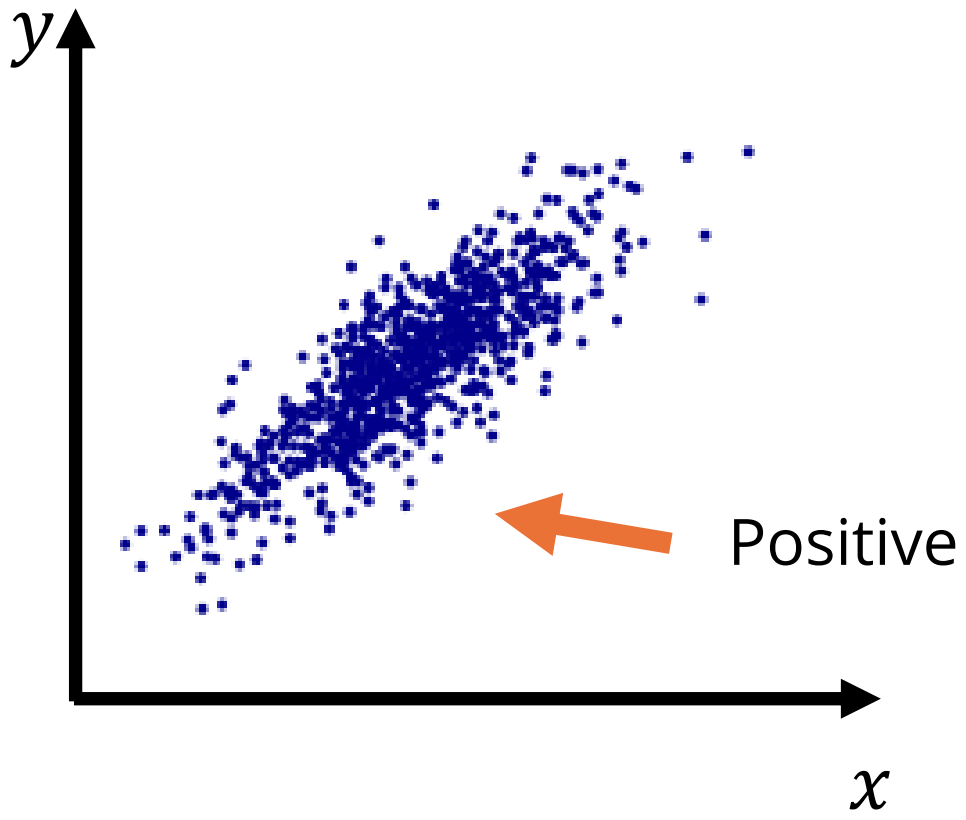


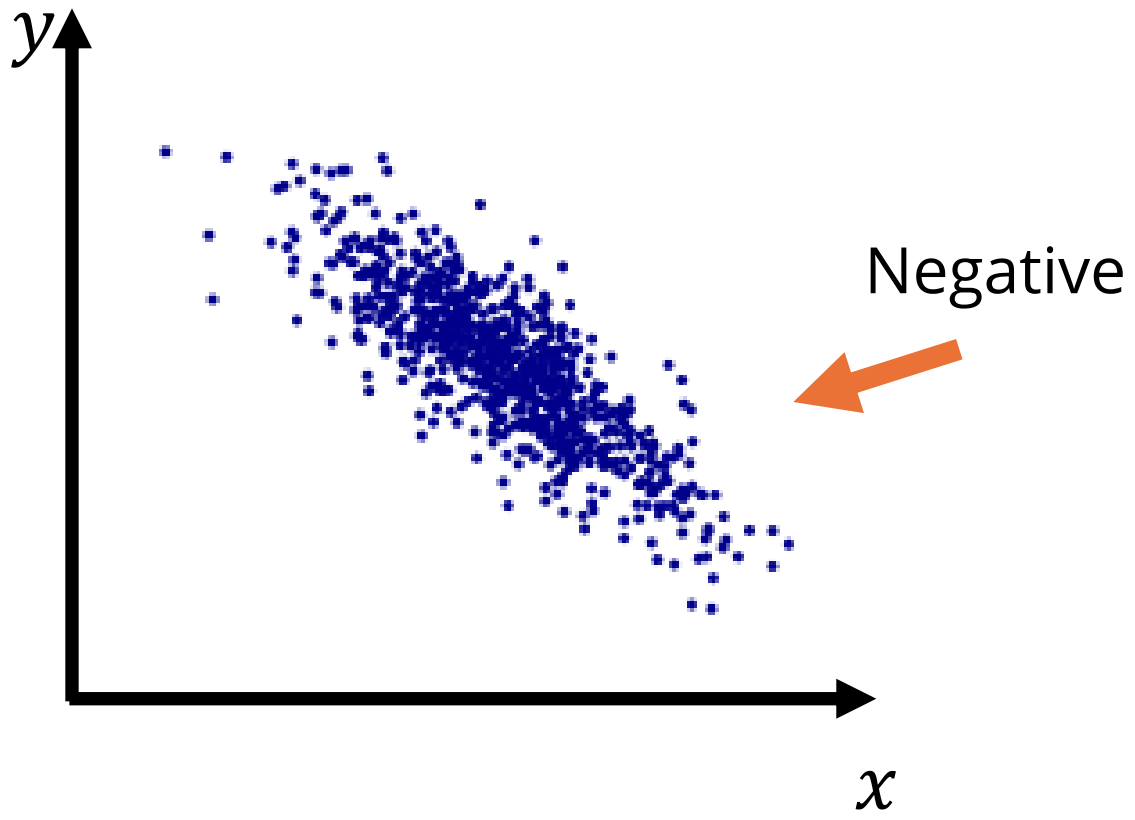
Kernel Density Estimation



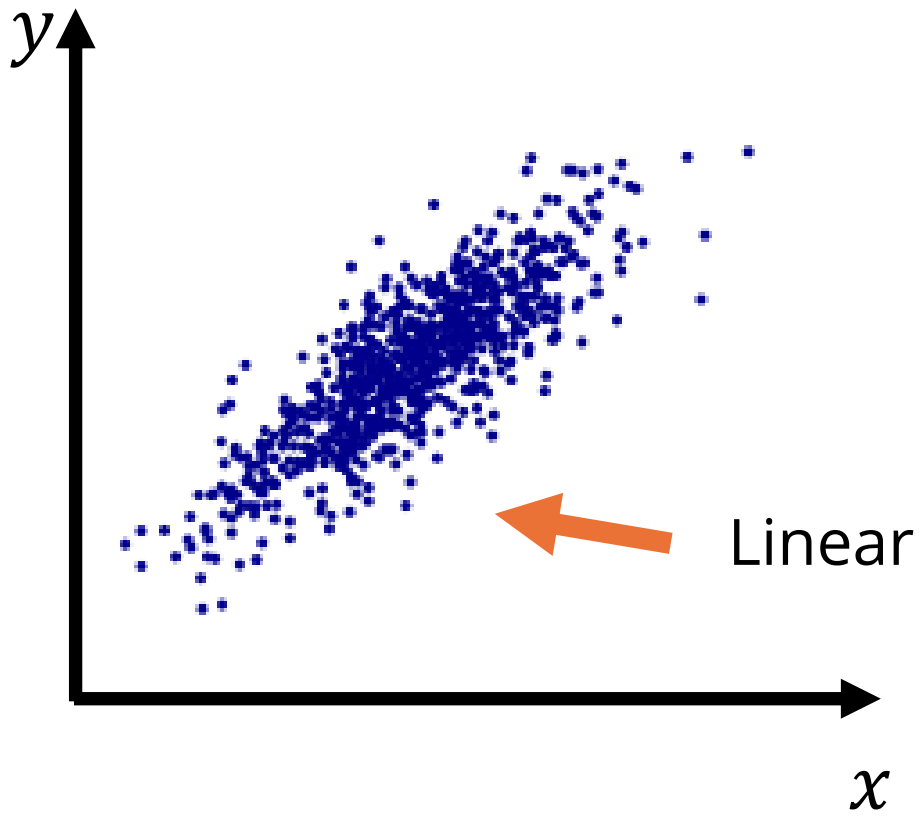
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html
https://en.wikipedia.org/wiki/Kernel_density_estimation

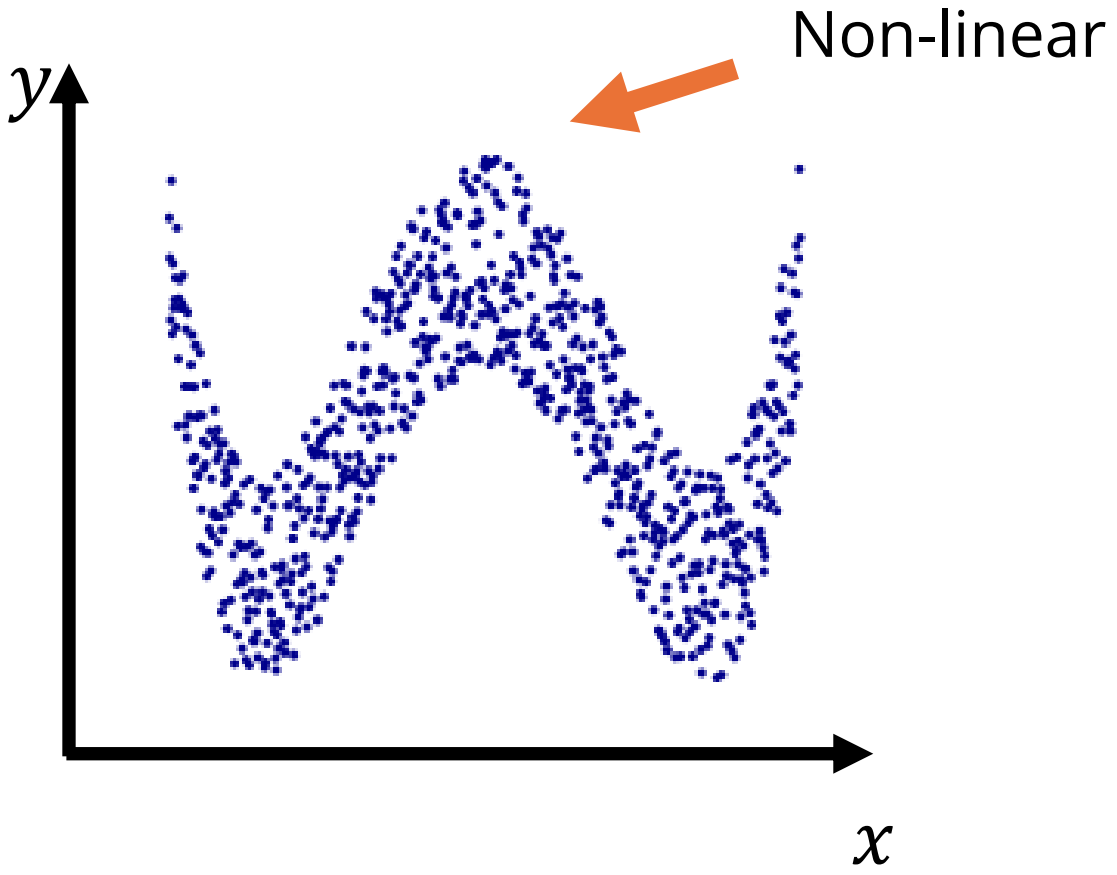
Correlation





Correlation





Pearson Correlation Coefficient r

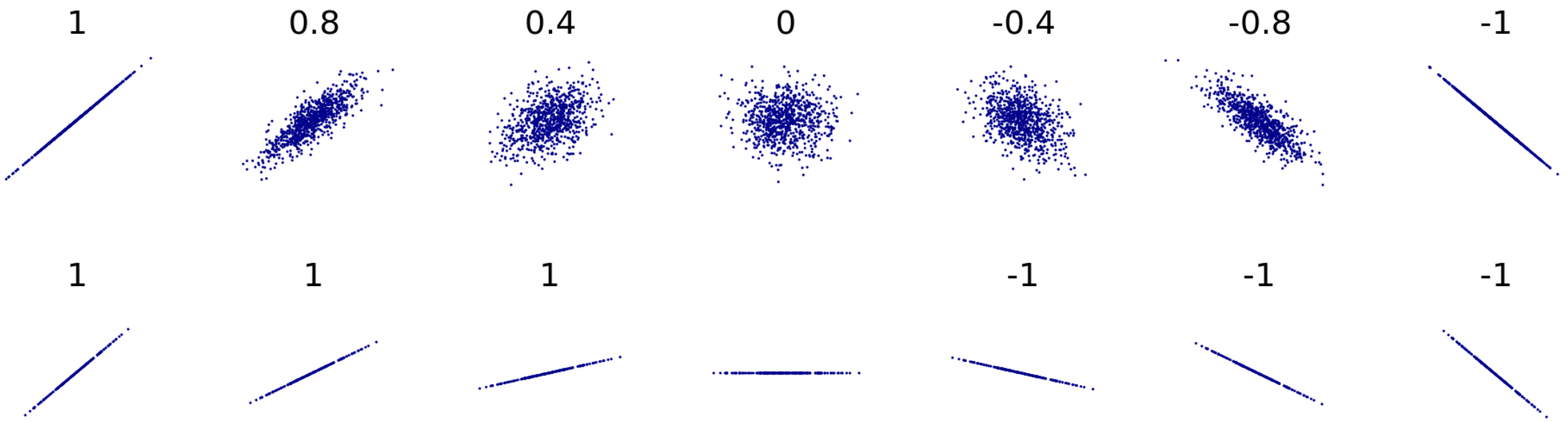
$$r = \frac{1}{n-1}\sum\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right)$$

standardized x values

standardized y values

Therefore r does not change with linear transformations

Pearson Correlation Coefficient r



Source: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

Pearson Correlation Coefficient r

Correlation is an incomplete description of bi-variate data!

Shouldn't be used:

- Outliers
- Non-linear relationships
- Subgroups

Data Visualization!!

Source: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

CMU Portugal Academy

02

Data Loading, Preparation and Processing with Pandas

Carnegie Mellon Portugal · TÉCNICO LISBOA · Ciências ULisboa · NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Pandas

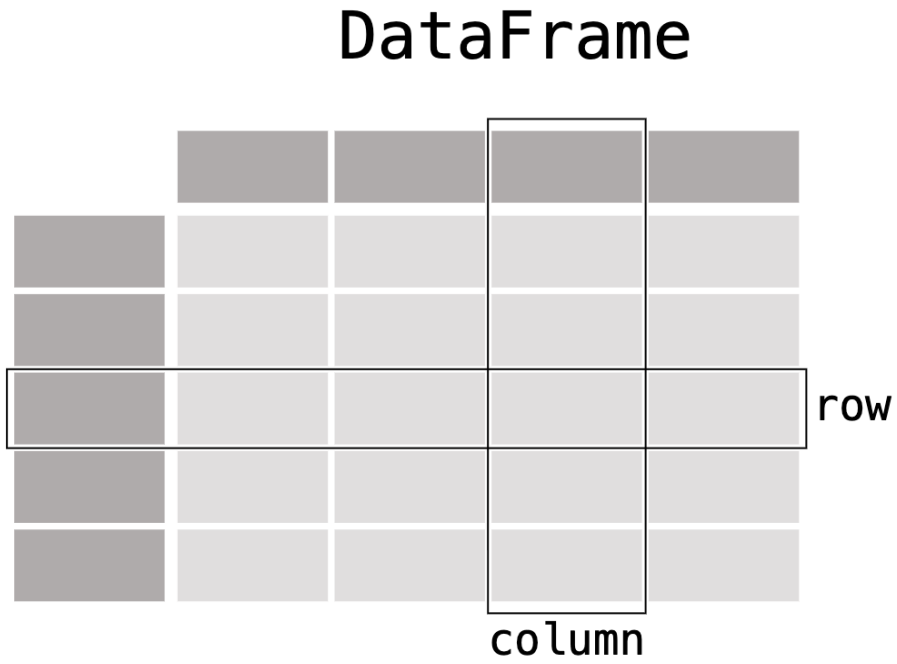

```
$ conda install pandas
```

- Python Library for Data Analysis
- Manipulating Data Tables
- Highly flexible and versatile!
- Leverages NumPy for numerical operations and Plotting Libraries.

Pandas

DataFrame: A 2-dimensional data structure that can store data of different types.

Similar to a spreadsheet or SQL table.

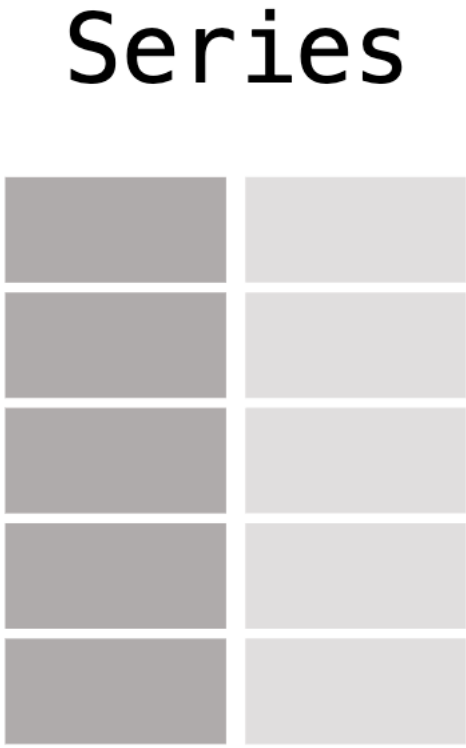


Pandas

Series: Corresponds to a single column of a DataFrame.

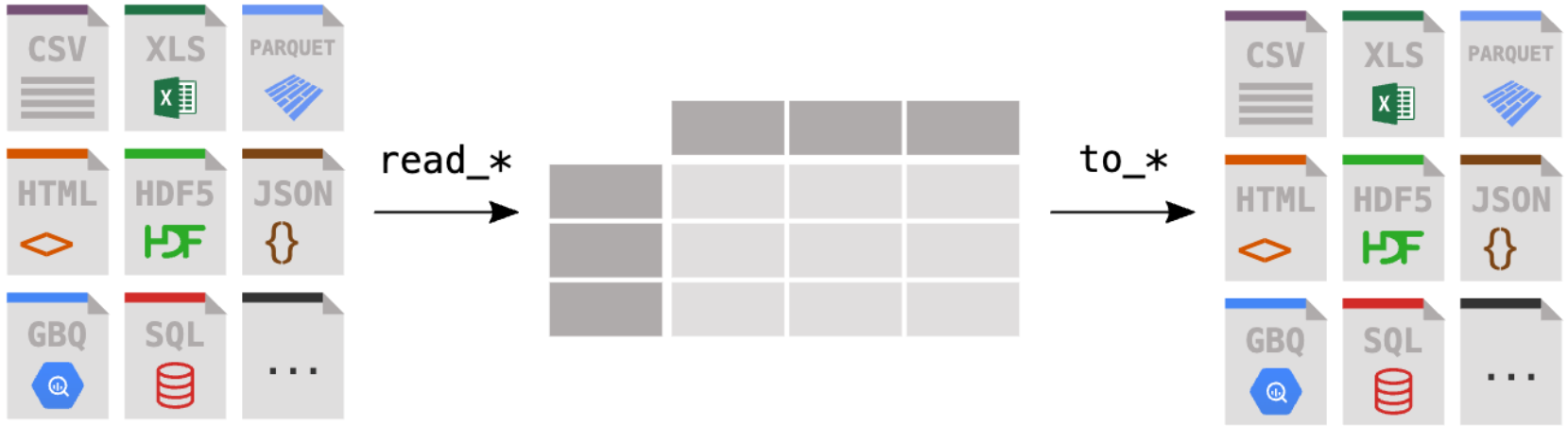
Columns do not have labels

Rows have labels.



Pandas

We can import data from multiple formats to a DataFrame, and export it back



Loading a dataset in CSV – Mental Illness Dataset

Comma Separated Values (CSV)

```
Entity,Code,Year,Schizophrenia disorders (share of population) - Sex: Both - Age: Age-standardized,Depressive dis
Afghanistan,AFG,1990,0.22320578,4.996118,4.713314,0.70302314,0.12770003
Afghanistan,AFG,1991,0.22245377,4.9892898,4.7021,0.7020688,0.123255946
Afghanistan,AFG,1992,0.22175121,4.9813457,4.683743,0.700792,0.11884415
Afghanistan,AFG,1993,0.22098725,4.9769583,4.6735487,0.70008695,0.11508888
Afghanistan,AFG,1994,0.22018303,4.977782,4.67081,0.6998978,0.11181468
Afghanistan,AFG,1995,0.2194088,4.978228,4.6681,0.6997684,0.10850699
Afghanistan,AFG,1996,0.21846454,4.981489,4.6657586,0.6996502,0.10526882
Afghanistan,AFG,1997,0.21728611,4.9875927,4.665013,0.69959736,0.10153082
Afghanistan,AFG,1998,0.21607415,4.9968576,4.6682405,0.69976467,0.09805014
Afghanistan,AFG,1999,0.21506761,5.004257,4.6735573,0.700054,0.095722646
Afghanistan,AFG,2000,0.21451464,5.0084476,4.6768804,0.70025194,0.09487554
Afghanistan,AFG,2001,0.21431455,5.0038724,4.675786,0.70012945,0.09454673
Afghanistan,AFG,2002,0.21418841,4.994327,4.6720624,0.69977945,0.09481004
Afghanistan,AFG,2003,0.21409267,4.9817224,4.6718807,0.69960576,0.09498258
Afghanistan,AFG,2004,0.21406065,4.9731607,4.6727514,0.6995014,0.09553074
Afghanistan,AFG,2005,0.21409707,4.9658794,4.6736326,0.6994395,0.095989875
Afghanistan,AFG,2006,0.214259,4.96284,4.6780405,0.69953,0.09687539
Afghanistan,AFG,2007,0.21459809,4.9597197,4.682732,0.6995203,0.09869648
Afghanistan,AFG,2008,0.21502665,4.9531674,4.688263,0.69948006,0.100904115
Afghanistan,AFG,2009,0.21545461,4.9492145,4.695408,0.6994876,0.10303231
Afghanistan,AFG,2010,0.21580303,4.946899,4.701994,0.6994379,0.10496491
Afghanistan,AFG,2011,0.21621753,4.945379,4.715658,0.69939154,0.10716891
Afghanistan,AFG,2012,0.21681084,4.9445314,4.73926,0.69936365,0.109071545
Afghanistan,AFG,2013,0.21744443,4.9443674,4.7667584,0.6993666,0.11139344
Afghanistan,AFG,2014,0.21797295,4.944629,4.7931795,0.69947726,0.11349482
```

Loading a dataset in CSV – Mental Illness Dataset

```
dataset_path = "datasets/1- mental-illnesses-prevalence.csv"
df = pd.read_csv(dataset_path)
```

[11] ✓ 0.0s Python

```
df
```

[8] ✓ 0.0s Python

	Entity	Code	Year	Schizophrenia disorders (share of population) - Sex: Both - Age: Age-standardized	Depressive disorders (share of population) - Sex: Both - Age: Age-standardized	Anxiety disorders (share of population) - Sex: Both - Age: Age-standardized	Bipolar disorders (share of population) - Sex: Both - Age: Age-standardized	Eating disorders (share of population) - Sex: Both - Age: Age-standardized
0	Afghanistan	AFG	1990	0.223206	4.996118	4.713314	0.703023	0.127700
1	Afghanistan	AFG	1991	0.222454	4.989290	4.702100	0.702069	0.123256
2	Afghanistan	AFG	1992	0.221751	4.981346	4.683743	0.700792	0.118844
3	Afghanistan	AFG	1993	0.220987	4.976958	4.673549	0.700087	0.115089
4	Afghanistan	AFG	1994	0.220183	4.977782	4.670810	0.699898	0.111815
...	...	...	...	...	...	...	...	...
6415	Zimbabwe	ZWE	2015	0.201042	3.407624	3.184012	0.538596	0.095652
6416	Zimbabwe	ZWE	2016	0.201319	3.410755	3.187148	0.538593	0.096662
6417	Zimbabwe	ZWE	2017	0.201639	3.411965	3.188418	0.538589	0.097330
6418	Zimbabwe	ZWE	2018	0.201976	3.406929	3.172111	0.538585	0.097909
6419	Zimbabwe	ZWE	2019	0.202482	3.395476	3.137017	0.538580	0.098295

6420 rows × 8 columns

Pandas – Basic Indexing

Supported Indexing Operations:

- By column (or columns) names;
- By row number (`iloc`)
- By row and column labels (`loc`)
- By a filtering condition, over all or subset of columns

More on this topic later!


jupyter

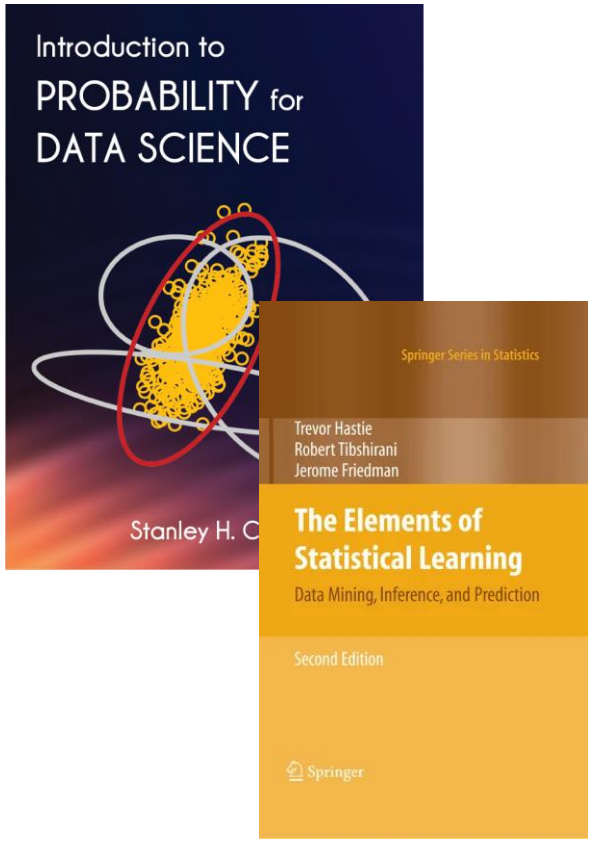
Hands-On Session!

Course Shared Folder

More on Statistics, Probability and Inference

- **Introduction to Probability for Data Science**, Stanley H. Chan, Michigan Publishing

- **The Elements of Statistical Learning:** Data Mining, Inference, and Prediction, Trevor Hastie, Robert Tibshirani, Jerome Friedman, 2nd edition, Springer



CMU Portugal Academy

CMU Portugal
Advanced Training Program

Foundations of Data Science

DAVID SEMEDO
RAFAEL FERREIRA
NOVA SCHOOL OF SCIENCE AND TECHNOLOGY

Carnegie Mellon Portugal · TÉCNICO LISBOA · Ciências ULisboa · NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Email: rah.ferreira@campus.fct.unl.pt

CMU Portugal Academy

CMU Portugal
Advanced Training Program

Foundations of Data Science

DAVID SEMEDO
RAFAEL FERREIRA
NOVA SCHOOL OF SCIENCE AND TECHNOLOGY

Carnegie Mellon Portugal · TÉCNICO LISBOA · Ciências ULisboa · NOVA School of Science & Technology

Email: rah.ferreira@campus.fct.unl.pt

CMU Portugal Academy

Today's Topics

1. PANDAS AGGREGATION
2. PANDAS GROUP BY
3. PANDAS SORTING
4. PANDAS FILTERING
5. INTRODUCTION TO DATA VISUALIZATION
6. MATPLOTLIB
7. PANDAS AND PLOTTING
8. SEABORN

CMU Portugal Academy

03

Pandas

Carnegie Mellon Portugal | TÉCNICO LISBOA | Ciências ULisboa | NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Follow Along...

Course Shared Folder - Google Drive
session_4a_datascience_cmu.ipynb

Recapping the Dataset

```
dataset_path = "datasets/1- mental-illnesses-prevalence.csv"
df = pd.read_csv(dataset_path)
```

[11] ✓ 0.0s Python

```
df
```

[8] ✓ 0.0s Python

	Entity	Code	Year	Schizophrenia disorders (share of population) - Sex: Both - Age: Age-standardized	Depressive disorders (share of population) - Sex: Both - Age: Age-standardized	Anxiety disorders (share of population) - Sex: Both - Age: Age-standardized	Bipolar disorders (share of population) - Sex: Both - Age: Age-standardized	Eating disorders (share of population) - Sex: Both - Age: Age-standardized
0	Afghanistan	AFG	1990	0.223206	4.996118	4.713314	0.703023	0.127700
1	Afghanistan	AFG	1991	0.222454	4.989290	4.702100	0.702069	0.123256
2	Afghanistan	AFG	1992	0.221751	4.981346	4.683743	0.700792	0.118844
3	Afghanistan	AFG	1993	0.220987	4.976958	4.673549	0.700087	0.115089
4	Afghanistan	AFG	1994	0.220183	4.977782	4.670810	0.699898	0.111815
...	...	...	...	...	...	...	...	...
6415	Zimbabwe	ZWE	2015	0.201042	3.407624	3.184012	0.538596	0.095652
6416	Zimbabwe	ZWE	2016	0.201319	3.410755	3.187148	0.538593	0.096662
6417	Zimbabwe	ZWE	2017	0.201639	3.411965	3.188418	0.538589	0.097330
6418	Zimbabwe	ZWE	2018	0.201976	3.406929	3.172111	0.538585	0.097909
6419	Zimbabwe	ZWE	2019	0.202482	3.395476	3.137017	0.538580	0.098295

6420 rows × 8 columns

Renaming Columns

```python
# lets simplify the column names
new_collumn_names = {
    'Schizophrenia disorders (share of population) - Sex: Both - Age: Age-standardized': 'Schizophrenia',
    'Depressive disorders (share of population) - Sex: Both - Age: Age-standardized': 'Depression',
    'Anxiety disorders (share of population) - Sex: Both - Age: Age-standardized': 'Anxiety',
    'Bipolar disorders (share of population) - Sex: Both - Age: Age-standardized': 'Bipolar',
    'Eating disorders (share of population) - Sex: Both - Age: Age-standardized': 'Eating',
}

# rename the columns based on the new_collumn_names dictionary
df = df.rename(columns=new_collumn_names)
```

✓ 0.0s Python

```python
df.head()
```

✓ 0.0s Python

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
0	Afghanistan	AFG	1990	0.223206	4.996118	4.713314	0.703023	0.127700
1	Afghanistan	AFG	1991	0.222454	4.989290	4.702100	0.702069	0.123256
2	Afghanistan	AFG	1992	0.221751	4.981346	4.683743	0.700792	0.118844
3	Afghanistan	AFG	1993	0.220987	4.976958	4.673549	0.700087	0.115089
4	Afghanistan	AFG	1994	0.220183	4.977782	4.670810	0.699898	0.111815

Pandas – Aggregating Results

- Creating a value from a column

```
# get average of depressive disorders
depressive_disorders_average = df['Depression'].mean()
depressive_disorders_average
```

✓ 0.0s

```
3.767035886853579
```

Pandas – Aggregating Results

- Creates a Dataframe by choosing various columns

```
# applying over multiple columns
df[['Depression', 'Anxiety']].mean()
```

✓ 0.0s

```
Depression    3.767036
Anxiety       4.101840
dtype: float64
```

Pandas – Aggregating Results

- Using agg() function:

```
# use agg to get multiple statistics
df.agg(['mean', 'max', 'min'])
```

✓ 0.0s

	Entity	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
max	Zimbabwe	2019.000000	0.462045	7.645899	8.624634	1.506730	1.031688
min	Afghanistan	1990.000000	0.188416	1.522333	1.880237	0.181667	0.044780
mean	NaN	2004.515258	0.266920	3.756903	4.102203	0.637203	0.196632

Pandas – Aggregating Results

- Using **agg()** function:
  - Different functions for each columns

```
# apply different functions to different columns
# it fills with NaN if the function is not applicable to the column
df.agg({
    'Depression': ['mean', 'median'],
    'Anxiety': ['min', 'max'],
})
```

✓ 0.0s

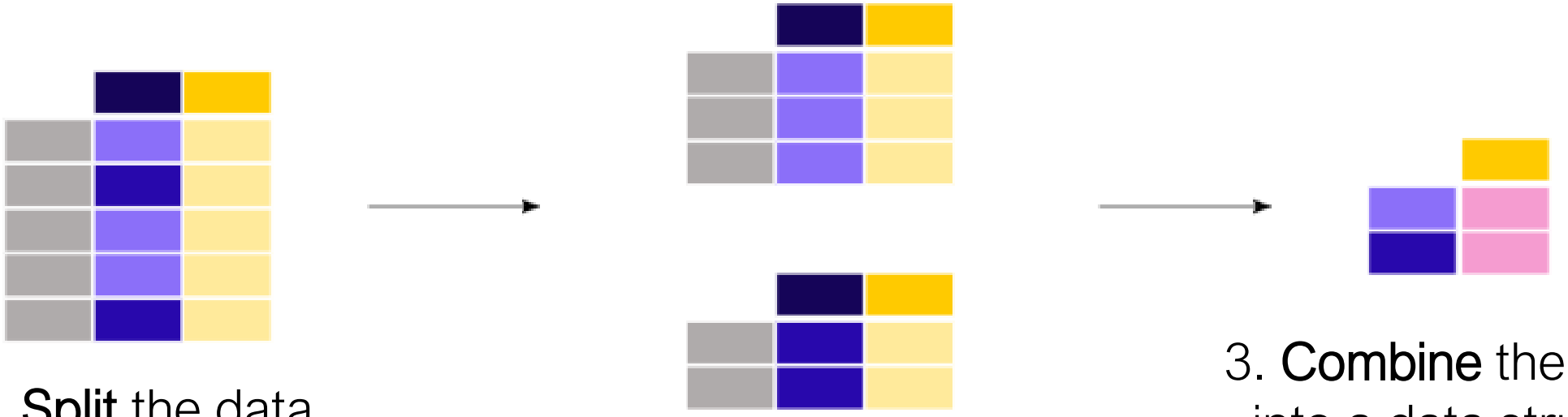
	Depression	Anxiety
max	NaN	8.624634
mean	3.767036	NaN
median	3.636772	NaN
min	NaN	1.879996

Pandas – Aggregating Functions

Operation	Description
sum()	Calculates the sum of values.
mean()	Calculates the average of values.
median()	Calculates the median of values.
mode()	Calculates the mode (most frequent value) of values.
max()	Finds the maximum value.
min()	Finds the minimum value.
std()	Calculates the standard deviation.
count()	Counts the number of non-null values.

Pandas – Grouping by Category

- Follow the **split-apply-combine** pattern:



1. **Split** the data into groups

2. **Apply** a function to each group independently

3. **Combine** the results into a data structure

Pandas – Grouping by Category

Using **grouby()**:

1. **Split** the data into groups
2. **Apply** a function to each group independently
3. **Combine** the results into a data structure

```
# aggregate by category
df.groupby('Year').mean()
```

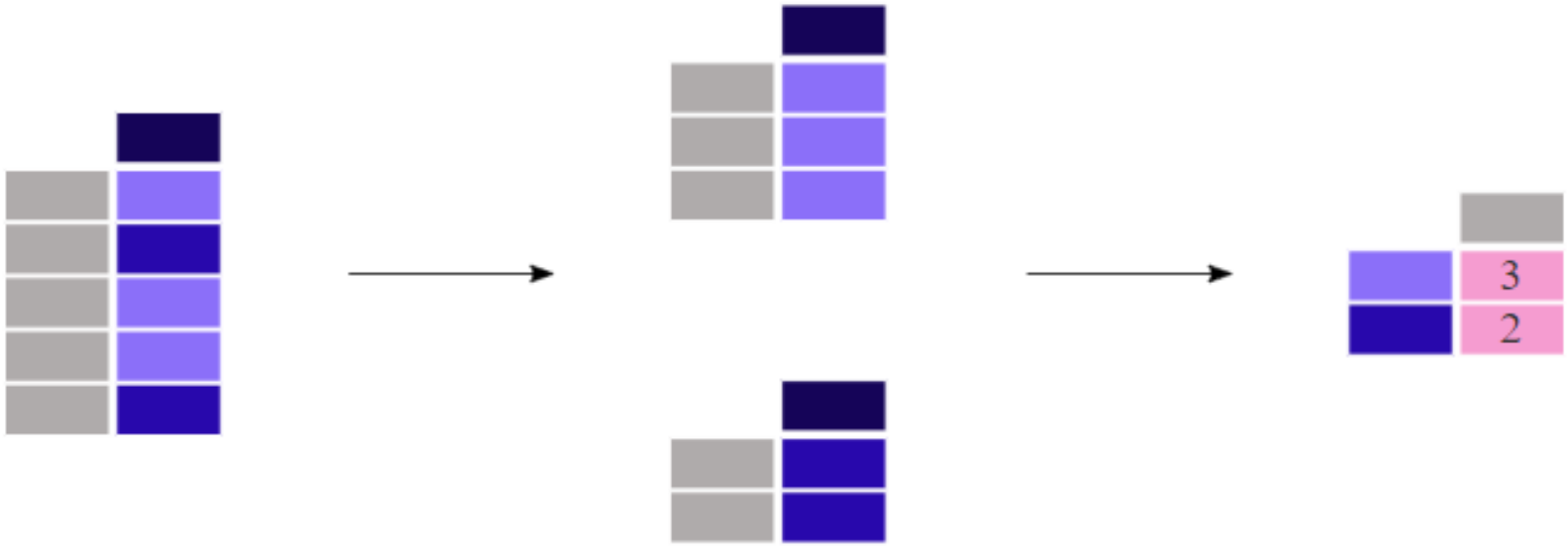
✓ 0.0s

Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
1990	0.265545	3.816706	4.067391	0.635783	0.182725
1991	0.265602	3.814530	4.067576	0.635792	0.182770
1992	0.265644	3.812701	4.067882	0.635811	0.182909
1993	0.265672	3.811056	4.068179	0.635835	0.183126
1994	0.265682	3.809931	4.068545	0.635868	0.183419
1995	0.265671	3.809265	4.068717	0.635902	0.183742
1996	0.265626	3.810262	4.069669	0.635923	0.184201

...

Pandas Counts by Category

- Using the **value_counts()** function:



Pandas Counts by Category

- Using and not using the **value_counts()** function:

```
# counts by category
df["Year"].value_counts()
```

✓ 0.0s

```
2011    169
2001    169
2005    168
2003    167
2012    165
2016    165
2019    164
1991    164
```

```
# value_counts is the same as the groupby and count
df.groupby('Year')["Year"].count()
```

✓ 0.0s

```
Year
1990    162
1991    164
1992    160
1993    149
1994    160
1995    159
1996    162
1997    157
```

value_counts() **changes the order**

Adding a New Column

- Similar to dictionary notation:

```
# adding a new column based on the sum of the other columns
# (ignore the fact that the sum some people might have more than one disorder)
df['Total'] = df['Depression'] + df['Anxiety'] + df['Schizophrenia'] + df['Bipolar'] + df['Eating']

df.head()
```

✓ 0.0s

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating	Total
0	Afghanistan	AFG	1990	0.223206	4.996118	4.713314	0.703023	0.127700	10.763361
1	Afghanistan	AFG	1991	0.222454	4.989290	4.702100	0.702069	0.123256	10.739168
2	Afghanistan	AFG	1992	0.221751	4.981346	4.683743	0.700792	0.118844	10.706476
3	Afghanistan	AFG	1993	0.220987	4.976958	4.673549	0.700087	0.115089	10.686670
4	Afghanistan	AFG	1994	0.220183	4.977782	4.670810	0.699898	0.111815	10.680488

Removing a Column

- Use the **drop()** function:

```
print("Columns Before:", df.columns)
# drop a column
df = df.drop(columns=['Total'])
print("Columns After:", df.columns)

df.head()
```

✓ 0.0s

```
Columns Before: Index(['Entity', 'Code', 'Year', 'Schizophrenia', 'Depression', 'Anxiety',
       'Bipolar', 'Eating', 'Total'],
      dtype='object')
Columns After: Index(['Entity', 'Code', 'Year', 'Schizophrenia', 'Depression', 'Anxiety',
       'Bipolar', 'Eating'],
      dtype='object')
```

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
0	Afghanistan	AFG	1990	0.223206	4.996118	4.713314	0.703023	0.127700
1	Afghanistan	AFG	1991	0.222454	4.989290	4.702100	0.702069	0.123256
2	Afghanistan	AFG	1992	0.221751	4.981346	4.683743	0.700792	0.118844
3	Afghanistan	AFG	1993	0.220987	4.976958	4.673549	0.700087	0.115089
4	Afghanistan	AFG	1994	0.220183	4.977782	4.670810	0.699898	0.111815

Handling Missing Values

- In computing, **NaN** (Not a Number) is a value used to represent a numeric value that is **undefined or unrepresentable**.

```
# set seed for reproducibility
np.random.seed(42)

# create a random column where sometimes it can be NaN
df['random_nan'] = np.random.choice([1, 2, 3, np.nan], size=len(df))

df.head()
```

✓ 0.0s

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating	random_nan
0	Afghanistan	AFG	1990	0.223206	4.996118	4.713314	0.703023	0.127700	3.0
2	Afghanistan	AFG	1992	0.221751	4.981346	4.683743	0.700792	0.118844	NaN
3	Afghanistan	AFG	1993	0.220987	4.976958	4.673549	0.700087	0.115089	1.0
4	Afghanistan	AFG	1994	0.220183	4.977782	4.670810	0.699898	0.111815	3.0
6	Afghanistan	AFG	1996	0.218465	4.981489	4.665759	0.699650	0.105269	3.0

Removing Missing Values

- dropna():
  - drops rows where exists a NaN value

```
print("Size before:", len(df['random_nan']))

# drop rows with NaN
df = df.dropna(subset=['random_nan'])

print("Size After:", len(df['random_nan']))

df.head()
```

✓ 0.0s

```
Size before: 6420
Size After: 4817
```

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating	random_nan
0	Afghanistan	AFG	1990	0.223206	4.996118	4.713314	0.703023	0.127700	3.0
2	Afghanistan	AFG	1992	0.221751	4.981346	4.683743	0.700792	0.118844	1.0
3	Afghanistan	AFG	1993	0.220987	4.976958	4.673549	0.700087	0.115089	3.0
4	Afghanistan	AFG	1994	0.220183	4.977782	4.670810	0.699898	0.111815	3.0
6	Afghanistan	AFG	1996	0.218465	4.981489	4.665759	0.699650	0.105269	1.0

Replacing Missing Values

- fillna():
  - replaces NaN values for the specified value

```
# set seed for reproducibility
np.random.seed(42)

# create a random column where sometimes it can be NaN
df['random_nan'] = np.random.choice([1, 2, 3, np.nan], size=len(df))

print("Mean before:", df['random_nan'].mean())

# fill NaN with the mean
df['random_nan'] = df['random_nan'].fillna(0)

print("Mean After:", df['random_nan'].mean())

# drop random_nan column
df = df.drop(columns=['random_nan'])
```

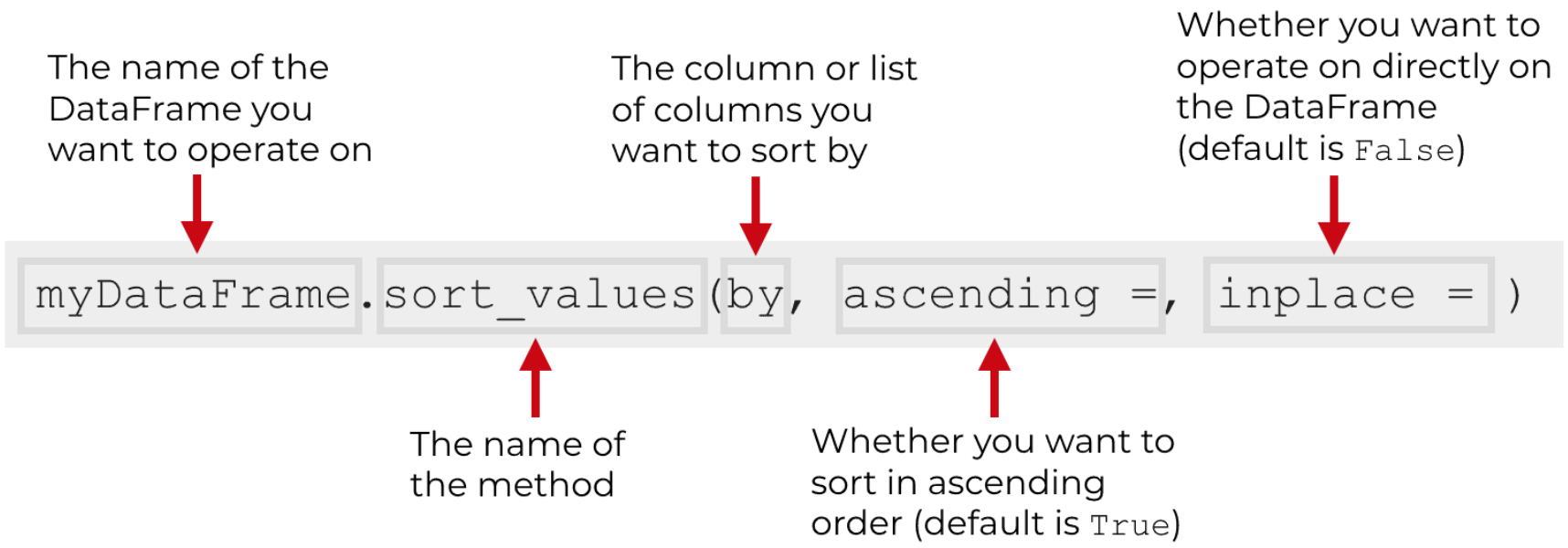
✓ 0.0s

```
Mean before: 1.9761177450708136
Mean After: 1.477268009134316
```

More Functions to Handle Missing Values

df.method()	description
dropna()	Drop missing observations
dropna(how='all')	Drop observations where all cells is NA
dropna(axis=1, how='all')	Drop column if all the values are missing
dropna(thresh = 5)	Drop rows that contain less than 5 non-missing values
fillna(0)	Replace missing values with zeros
isnull()	returns True if the value is missing
notnull()	Returns True for non-missing values

Pandas Sorting



CMU Portugal Academy

Pandas Sorting

```
# sorting by a column
df.sort_values('Depression', inplace=False, ascending=False, na_position='first')

# inplace=False returns a new dataframe
# inplace=True modifies the dataframe in place (i.e. changes the original dataframe)

# na_position='first' puts the NaN values first
# na_position='last' puts the NaN values last
```

✓ 0.0s

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
5950	Uganda	UGA	2000	0.202730	7.645899	3.893322	0.581878	0.078505
5951	Uganda	UGA	2001	0.203101	7.639734	3.896915	0.581884	0.079177
5949	Uganda	UGA	1999	0.202563	7.620522	3.890507	0.581870	0.078026
5953	Uganda	UGA	2003	0.204524	7.597861	3.908152	0.581895	0.080637
5948	Uganda	UGA	1998	0.202391	7.558059	3.887460	0.581864	0.077241
...	...	...	...	...	...	...	...	...
893	Brunei	BRN	2013	0.304498	1.542479	2.868245	0.606702	0.432183
892	Brunei	BRN	2012	0.304932	1.533041	2.866833	0.606739	0.432490
889	Brunei	BRN	2009	0.305295	1.525317	2.870184	0.606810	0.431752
891	Brunei	BRN	2011	0.305267	1.525050	2.866211	0.606769	0.432644
890	Brunei	BRN	2010	0.305376	1.522333	2.867077	0.606802	0.432542

4817 rows × 8 columns

Pandas Filtering

- Similar to numpy create a mask and then filters:

```
# total number of rows
print("Total rows:", len(df))
# filter by a condition
mask = df['Depression'] < 4.0
print(mask)
```

✓ 0.0s

```
Total rows: 4817
0       False
2       False
3       False
4       False
6       False
        ...
6415     True
6416     True
6417     True
6418     True
6419     True
Name: Depression, Length: 4817, dtype: bool
```

```
result = df[mask]
print("Depression below 4:", len(result))

result
```

✓ 0.0s

```
Depression below 4: 3004
```

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
60	Albania	ALB	1990	0.281197	2.381431	3.659145	0.541249	0.103737
61	Albania	ALB	1991	0.280868	2.382247	3.663672	0.541331	0.101877
62	Albania	ALB	1992	0.280622	2.383819	3.668394	0.541427	0.100271
63	Albania	ALB	1993	0.280456	2.387142	3.673226	0.541536	0.099217
64	Albania	ALB	1994	0.280365	2.390369	3.678023	0.541654	0.098728
...	...	...	...	...	...	...	...	...
6415	Zimbabwe	ZWE	2015	0.201042	3.407624	3.184012	0.538596	0.095652
6416	Zimbabwe	ZWE	2016	0.201319	3.410755	3.187148	0.538593	0.096662

Pandas Filtering

- Combine various conditions with:
  - **&** (and)
  - | (or)
  - ~ (not)

```
# filter by multiple conditions
print("Total rows:", len(df))
# can use & for and, | for or, and ~ for not
depression_over_2_and_below_2 = df[(df['Depression'] > 2) & (df['Depression'] < 3)]
print("Depression over 2 and below 3:", len(depression_over_2_and_below_2))

depression_over_2_and_below_2
```

✓ 0.0s

```
Total rows: 4817
Depression over 2 and below 3: 996
```

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
60	Albania	ALB	1990	0.281197	2.381431	3.659145	0.541249	0.103737
61	Albania	ALB	1991	0.280868	2.382247	3.663672	0.541331	0.101877
62	Albania	ALB	1992	0.280622	2.383819	3.668394	0.541427	0.100271
63	Albania	ALB	1993	0.280456	2.387142	3.673226	0.541536	0.099217
64	Albania	ALB	1994	0.280365	2.390369	3.678023	0.541654	0.098728
...	...	...	...	...	...	...	...	...
6294	Vietnam	VNM	2014	0.320686	2.472851	2.253840	0.336163	0.086964
6295	Vietnam	VNM	2015	0.321393	2.471376	2.285857	0.336145	0.088313
6297	Vietnam	VNM	2017	0.321537	2.472004	2.277422	0.336123	0.091739

Pandas Filtering

- Filtering using string operations:
  - .str.<function>()

```
# countries starting with P
filtered_startswith = df[df['Entity'].str.startswith('K')]
filtered_startswith
```

✓ 0.0s

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
2940	Kazakhstan	KAZ	1990	0.274562	3.496055	2.120997	0.507571	0.132850
2941	Kazakhstan	KAZ	1991	0.274427	3.485470	2.120301	0.507551	0.131387
2942	Kazakhstan	KAZ	1992	0.274234	3.476447	2.120074	0.507549	0.130053
2943	Kazakhstan	KAZ	1993	0.274005	3.470038	2.120222	0.507555	0.128486
2945	Kazakhstan	KAZ	1995	0.273496	3.470008	2.121669	0.507609	0.125587
...	...	...	...	...	...	...	...	...
3081	Kyrgyzstan	KGZ	2011	0.261259	3.417319	2.053481	0.515826	0.086764
3083	Kyrgyzstan	KGZ	2013	0.261792	3.411839	2.057651	0.515831	0.088847
3084	Kyrgyzstan	KGZ	2014	0.262066	3.408539	2.059585	0.515829	0.089873
3085	Kyrgyzstan	KGZ	2015	0.262264	3.407313	2.060363	0.515818	0.090874
3086	Kyrgyzstan	KGZ	2016	0.262557	3.398904	2.058070	0.515803	0.091421

116 rows × 8 columns

Pandas Filtering

- Filtering over set of values:
  - isin()

```
# using isin
filtered_isin = df[df['Entity'].isin(['Portugal', 'United States', 'Brazil'])]
filtered_isin
```

✓ 0.0s

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
840	Brazil	BRA	1990	0.274037	4.007267	5.894825	1.112508	0.202025
842	Brazil	BRA	1992	0.274426	3.977750	5.986008	1.112647	0.201797
843	Brazil	BRA	1993	0.274531	3.973795	6.032992	1.112701	0.201734
844	Brazil	BRA	1994	0.274582	3.976075	6.072099	1.112742	0.201927
845	Brazil	BRA	1995	0.274582	3.981849	6.096675	1.112763	0.202034
...	...	...	...	...	...	...	...	...
6084	United States	USA	2014	0.446098	4.404113	5.232689	0.601757	0.422944
6086	United States	USA	2016	0.436417	4.372808	5.168079	0.601451	0.422210
6087	United States	USA	2017	0.431800	4.369103	5.175325	0.601232	0.426212
6088	United States	USA	2018	0.431886	4.369584	5.342051	0.601198	0.426103
6089	United States	USA	2019	0.433649	4.375998	5.697748	0.601298	0.424350

73 rows × 8 columns

Pandas Filtering

- Filtering using a **custom function**

```
# apply filtering with a custom function
def custom_filter(x):
    return x['Depression'] < 3 and x['Anxiety'] > 2

df[df.apply(custom_filter, axis=1)]
```

✓ 0.0s

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
60	Albania	ALB	1990	0.281197	2.381431	3.659145	0.541249	0.103737
61	Albania	ALB	1991	0.280868	2.382247	3.663672	0.541331	0.101877
62	Albania	ALB	1992	0.280622	2.383819	3.668394	0.541427	0.100271
63	Albania	ALB	1993	0.280456	2.387142	3.673226	0.541536	0.099217
64	Albania	ALB	1994	0.280365	2.390369	3.678023	0.541654	0.098728
...	...	...	...	...	...	...	...	...
6294	Vietnam	VNM	2014	0.320686	2.472851	2.253840	0.336163	0.086964
6295	Vietnam	VNM	2015	0.321393	2.471376	2.285857	0.336145	0.088313
6297	Vietnam	VNM	2017	0.321537	2.472004	2.277422	0.336123	0.091739
6298	Vietnam	VNM	2018	0.322204	2.472504	2.279933	0.336112	0.093410
6299	Vietnam	VNM	2019	0.323551	2.473061	2.287339	0.336103	0.094745

1045 rows × 8 columns

Combining Groups and Filters

- Apply a **filter()** after a **groupby()**

```
# combining filters with groupby
df.groupby('Entity').filter(lambda x: x['Depression'].mean() > 3)
```

✓ 0.0s

	Entity	Code	Year	Schizophrenia	Depression	Anxiety	Bipolar	Eating
0	Afghanistan	AFG	1990	0.223206	4.996118	4.713314	0.703023	0.127700
2	Afghanistan	AFG	1992	0.221751	4.981346	4.683743	0.700792	0.118844
3	Afghanistan	AFG	1993	0.220987	4.976958	4.673549	0.700087	0.115089
4	Afghanistan	AFG	1994	0.220183	4.977782	4.670810	0.699898	0.111815
6	Afghanistan	AFG	1996	0.218465	4.981489	4.665759	0.699650	0.105269
...	...	...	...	...	...	...	...	...
6415	Zimbabwe	ZWE	2015	0.201042	3.407624	3.184012	0.538596	0.095652
6416	Zimbabwe	ZWE	2016	0.201319	3.410755	3.187148	0.538593	0.096662
6417	Zimbabwe	ZWE	2017	0.201639	3.411965	3.188418	0.538589	0.097330
6418	Zimbabwe	ZWE	2018	0.201976	3.406929	3.172111	0.538585	0.097909
6419	Zimbabwe	ZWE	2019	0.202482	3.395476	3.137017	0.538580	0.098295

3796 rows × 8 columns

Hands-On Session!

Course Shared Folder - Google Drive
session_4a_datascience_cmu.ipynb - Exercises

CMU Portugal Academy

04

Data Visualization with Pandas, Matplotlib and Seaborn

Data Visualization... but Why?

Raw Data:
Hard to Interpret

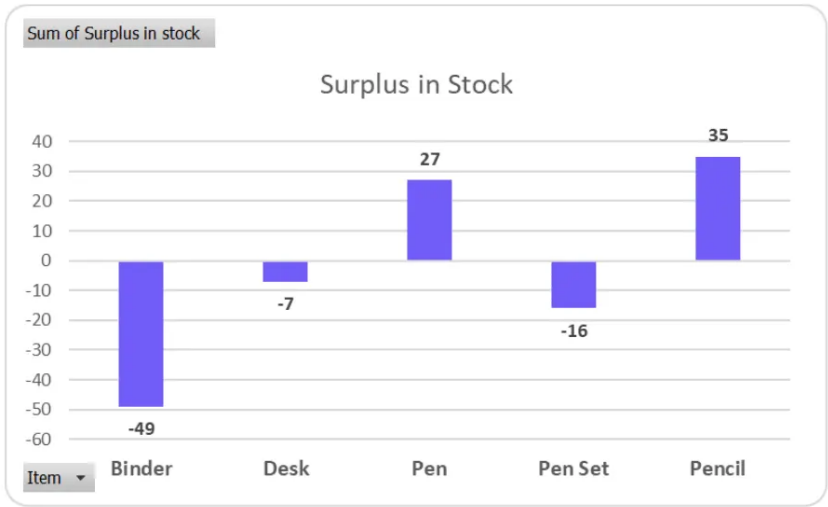
Data Visualization
Clearer Insights

Data

Data Visualization... but Why?

- Makes Complex Data Understandable
  - Raw data can be **overwhelming** and difficult to interpret.
  - Turn it **into visual elements** - charts, graphs, etc.

Order Date	Region	Employee ID	Name	Surname	Full Name	Item	Units In Inventory	Sold Units	Surplus in stock	Unit Cost	Total
1/6/21	East	139708	David	Jones	David Jones	Pencil	100	95	5	1.99	189.05
1/23/21	Central	401971	John	Kivell	John Kivell	Binder	60	50	10	19.99	999.50
2/9/21	Central	281046	Gabriel	Jardine	Gabriel Jardine	Pencil	36	36	0	4.99	179.64
2/26/21	Central	127572	John	Gill	John Gill	Pen	27	27	0	19.99	539.73
3/15/21	West	44645	Andreas	Sorvino	Andreas Sorvino	Pencil	56	56	0	2.99	167.44
4/1/21	East	324744	Bridget	Jones	Bridget Jones	Binder	10	60	-50	4.99	299.40
4/18/21	Central	229150	Rob	Andrews	Rob Andrews	Pencil	75	75	0	1.99	149.25
5/5/21	Central	339307	Gabriel	Jardine	Gabriel Jardine	Pencil	90	90	0	4.99	449.10
5/22/21	West	386044	Dave	Thompson	Dave Thompson	Pencil	32	32	0	1.99	63.68
6/8/21	East	139708	Bridget	Jones	Bridget Jones	Binder	60	60	0	8.99	539.40
6/25/21	Central	370024	TJ	Morgan	TJ Morgan	Pencil	90	90	0	4.99	449.10
7/12/21	East	309647	Len	Howard	Len Howard	Binder	29	29	0	1.99	57.71
7/29/21	East	380608	Steve	Parent	Steve Parent	Binder	81	81	0	19.99	1,619.19
8/15/21	East	452998	Bridget	Jones	Bridget Jones	Pencil	35	35	0	4.99	174.65
9/1/21	Central	264816	Tom	Smith	Tom Smith	Desk	2	2	0	125.00	250.00
9/18/21	East	143055	Bridget	Jones	Bridget Jones	Pen Set	16	16	0	15.99	255.84
10/5/21	Central	426184	TJ	Morgan	TJ Morgan	Binder	70	28	42	8.99	251.72
10/22/21	East	139708	Bridget	Jones	Bridget Jones	Pen	64	64	0	8.99	575.36
11/8/21	East	99377	Steve	Parent	Steve Parent	Pen	15	15	0	19.99	299.85
11/25/21	Central	401971	John	Kivell	John Kivell	Pen Set	57	53	4	4.99	264.47
12/12/21	Central	56018	Jonathan	Smith	Jonathan Smith	Pencil	11	89	-78	1.29	114.81
12/29/21	East	88260	Steve	Parent	Steve Parent	Pen Set	8	23	-15	15.99	367.77
1/15/22	Central	65626	John	Gill	John Gill	Binder	75	31	44	8.99	278.69
2/1/22	Central	474696	Steve	Smith	Steve Smith	Binder	68	50	18	15.00	750.00
2/18/22	East	139708	John	Jones	John Jones	Binder	78	55	23	4.99	274.45
3/7/22	West	436013	Jonathan	Sorvino	Jonathan Sorvino	Binder	52	48	4	19.99	959.52
3/24/22	Central	482378	Steve	Jardine	Steve Jardine	Pen Set	15	1	14	4.99	4.99
4/10/22	Central	229150	John	Andrews	John Andrews	Pencil	84	13	71	1.99	25.87
4/27/22	East	231018	Steve	Howard	Steve Howard	Pen	31	57	-26	4.99	284.43
5/14/22	Central	436791	John	Gill	John Gill	Pencil	40	86	-46	1.29	110.94
5/31/22	Central	436791	Jonathan	Gill	Jonathan Gill	Binder	85	42	43	8.99	377.58
6/17/22	Central	401971	Steve	Kivell	Steve Kivell	Desk	29	3	26	125.00	375.00
7/4/22	East	468311	John	Jones	John Jones	Pen Set	10	65	-55	4.99	324.35
7/21/22	Central	400131	Steve	Morgan	Steve Morgan	Pen Set	62	54	8	12.49	674.46
8/7/22	Central	401971	John	Kivell	John Kivell	Pen Set	60	27	33	23.95	646.65
8/24/22	West	262588	Jonathan	Sorvino	Jonathan Sorvino	Desk	53	55	-2	275.00	15,125.00
9/10/22	Central	436791	Steve	Gill	Steve Gill	Pencil	63	88	-25	1.29	113.52



Data Visualization... but Why?

Reveals Trends and Patterns

Identifies hidden insights.

Shows relationships between variables.

Communicates Findings Effectively

Visual storytelling communicates results clearly to non-technical stakeholders.

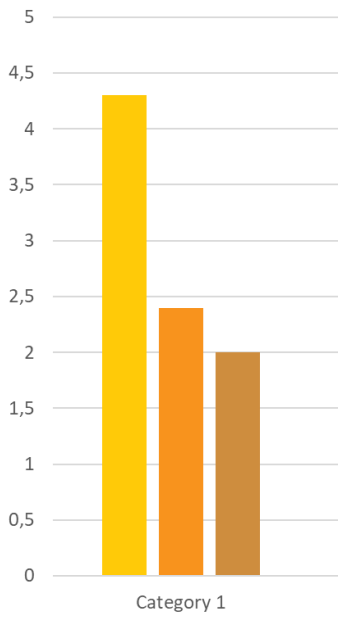
Faster decision-making.

Supports Data-Driven Decision Making

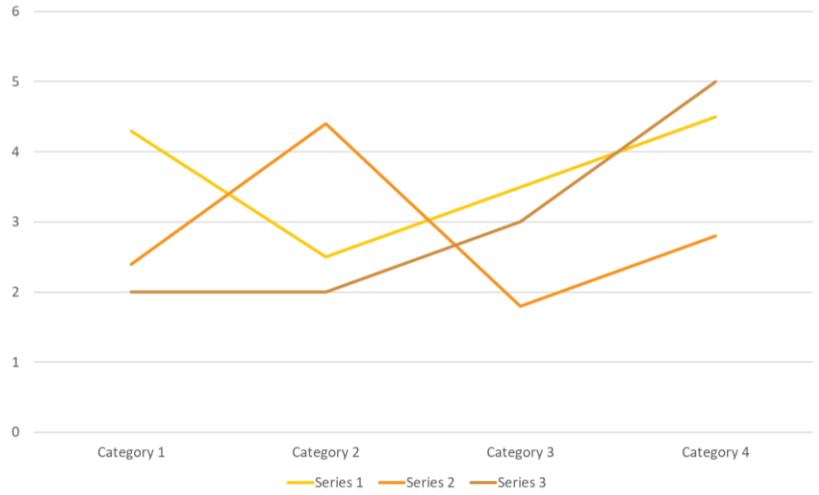
Visual data guides decisions by highlighting actionable insights.

Better decision-making grounded in evidence rather than intuition.

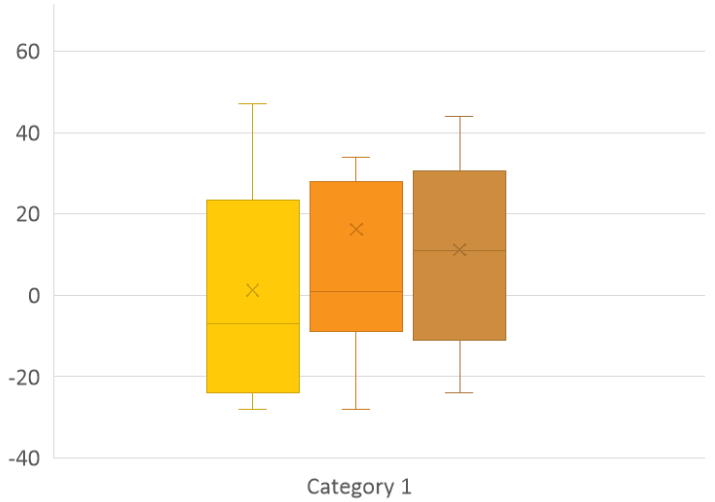
Visualization Types



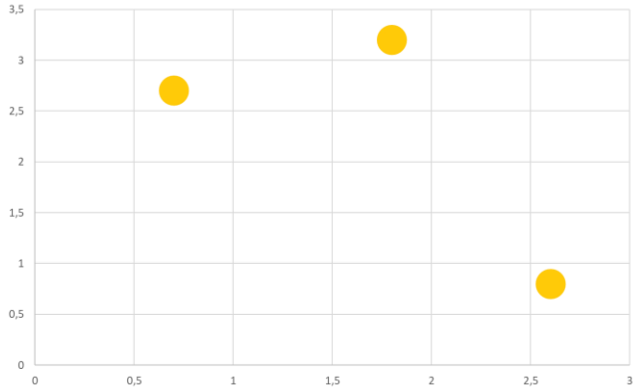
Bar chart



Line chart



Box plot



Scatter plot

Data Visualization: Guidelines

- Choosing the right type of graph depends on the **kind of data** you have and the **insights** you're trying to reveal.

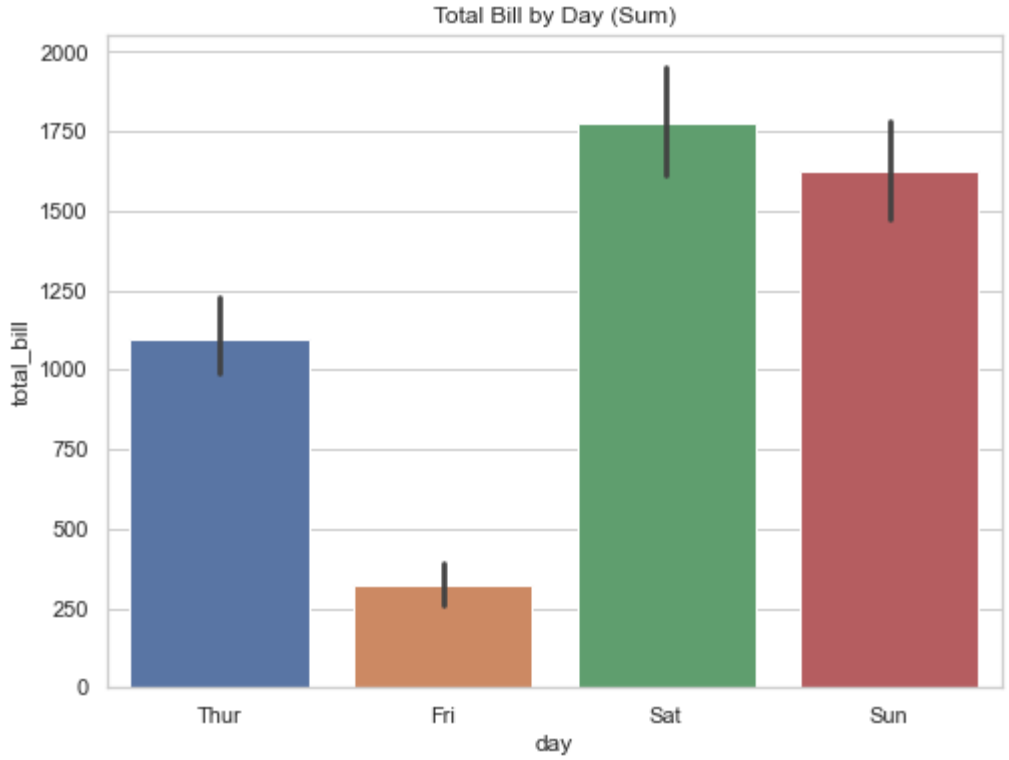
There's no one size fits all

Bar Chart

Categorical data

- **Comparing categories** or discrete groups.
- Showing the difference in values across **different categories**

- **Examples**:
  - Money spent by day;
  - Population across different cities.

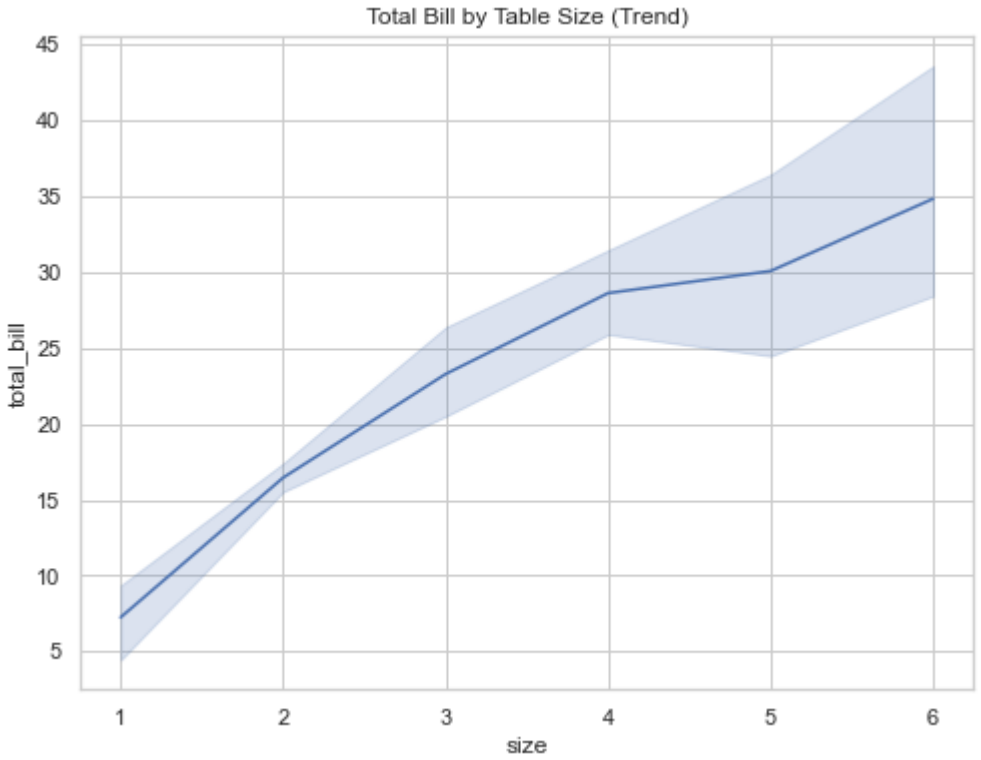


Line Plot

Continuous Variables

- Showing **trends** or patterns over **time**.
- Demonstrating a relationship between **two continuous variables**.

- **Examples**:
  - Total bill by number of people
  - Temperature changes over months.

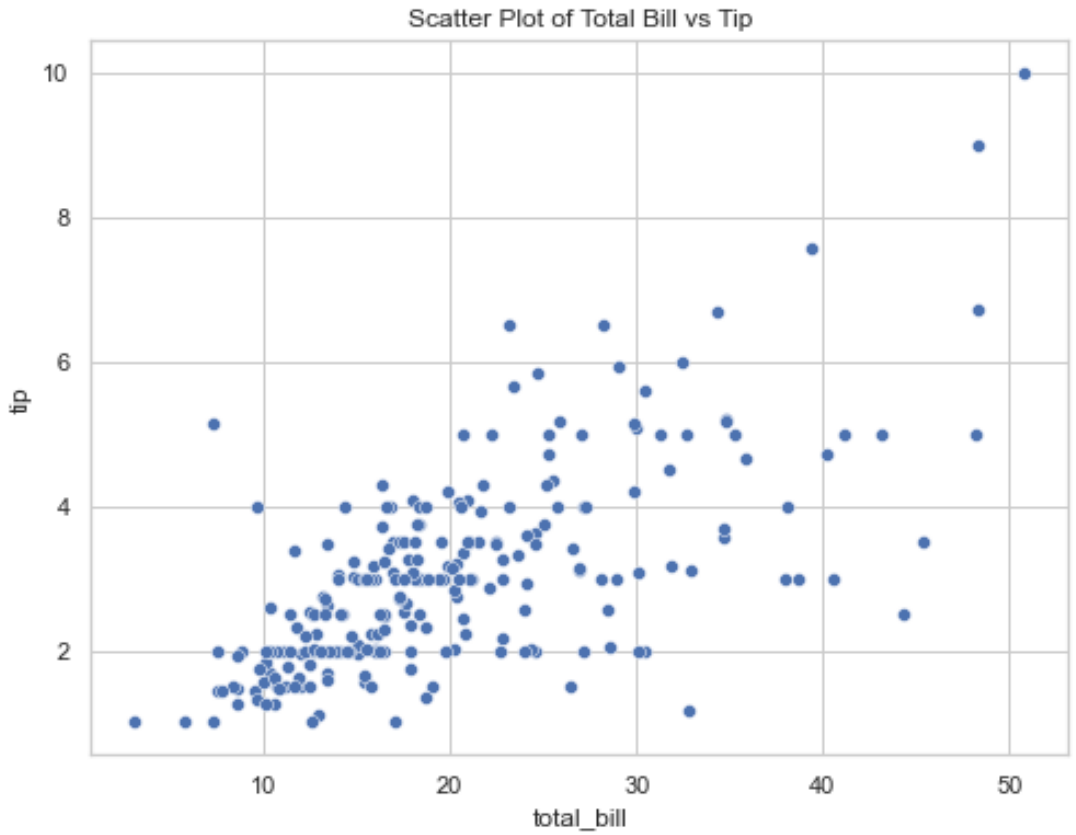


Scatter Plot

Continuous Variables

- Exploring relationships between **two continuous variables**.
- Looking for **correlation** between variables.
- Identifying **outliers** or clusters.

- **Examples**:
  - Distribution of bill by tip.
  - Hours studied vs. exam score.

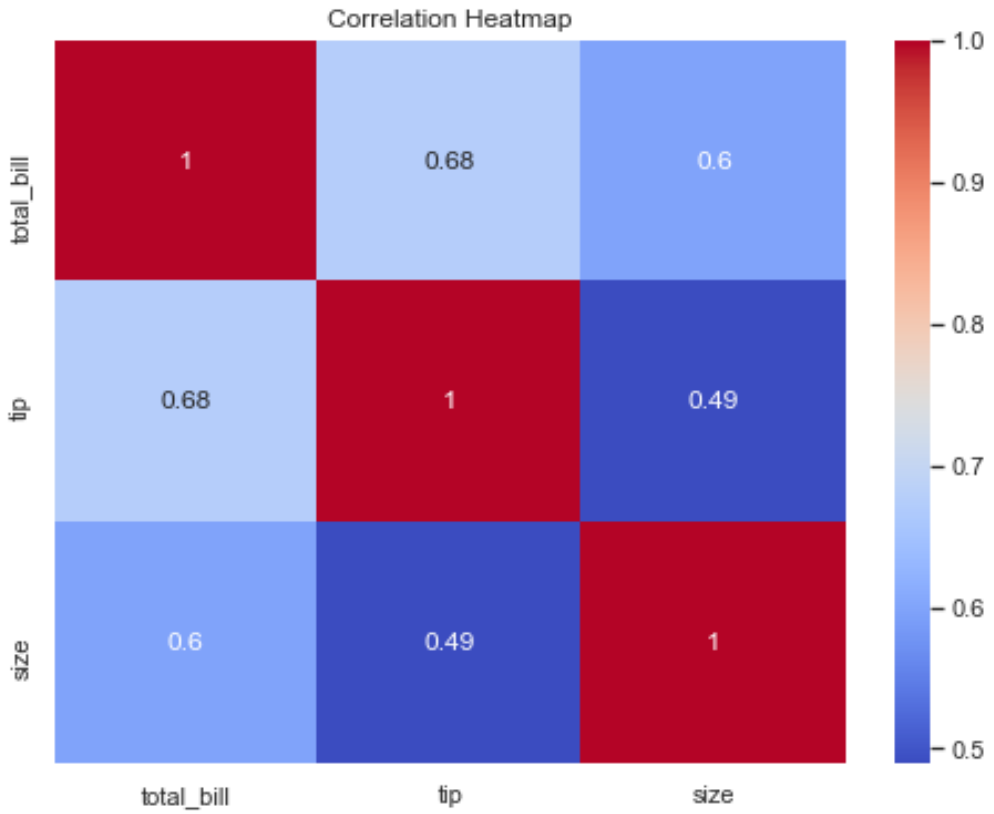


Heatmap

Multiple Variables

- Visualizing **correlations** between multiple variables.
- Displaying a matrix of **values** where color represents magnitude.
- Correlation between features in a dataset.

- **Examples**:
  - Correlation between tip, total bill and size

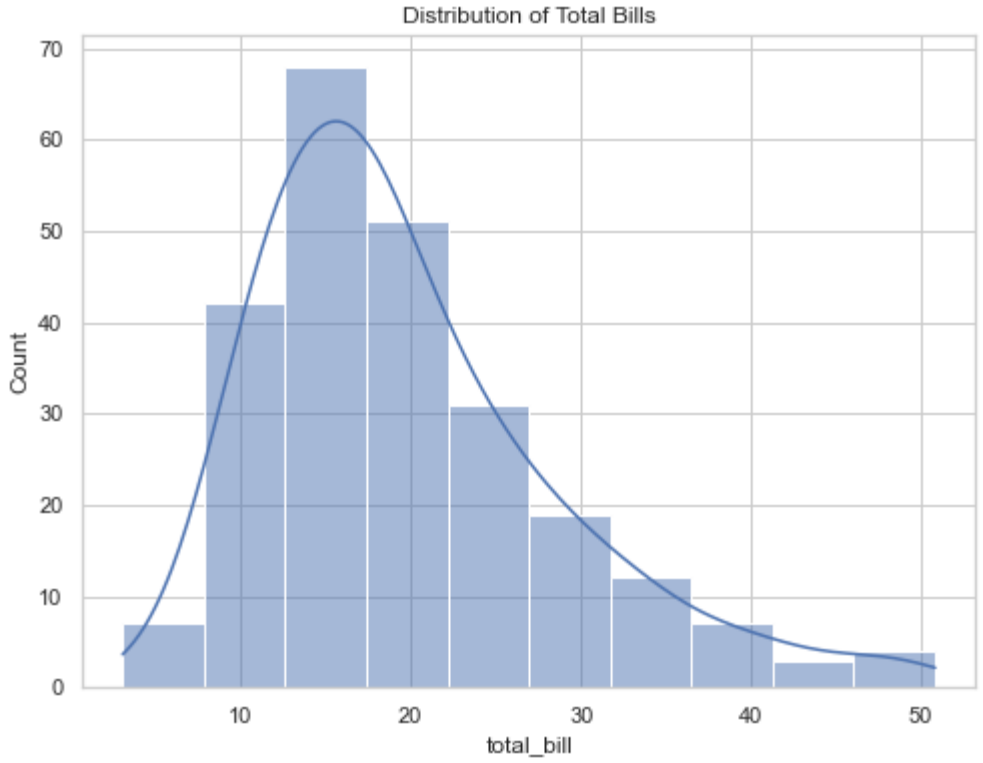


Histogram

- Displaying **frequency** of data points within specific **intervals** (i.e. bins).
- Checking for **skewness** or identifying **outliers**.
- Visualizing the **shape** of the data distribution (normal, skewed, etc.).

- **Examples**:
  - Distribution of the total of bills.
  - Age distribution in a population.

Continuous Variable



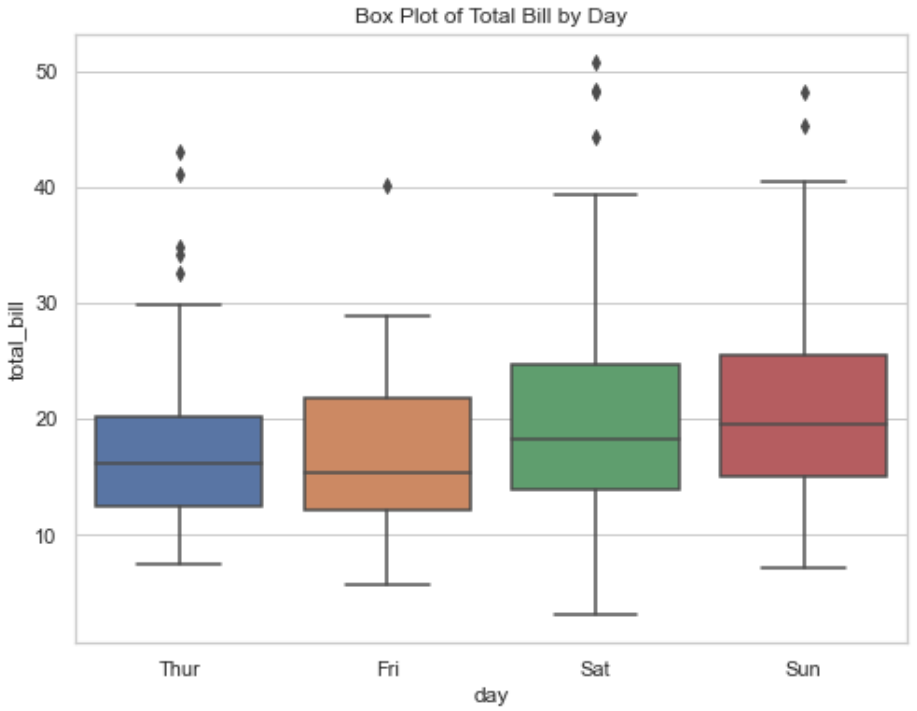
Shape of the distribution

Box Plot

- **Distribution** (spread, range, and outliers) of a continuous variable.
- Identifying **outliers**, **quartiles**, and **median** values.
- Comparing the distribution of data across different categories.

- **Examples**:
  - Bill per day of the week.
  - Salary distributions across different industries.

Continuous Variable



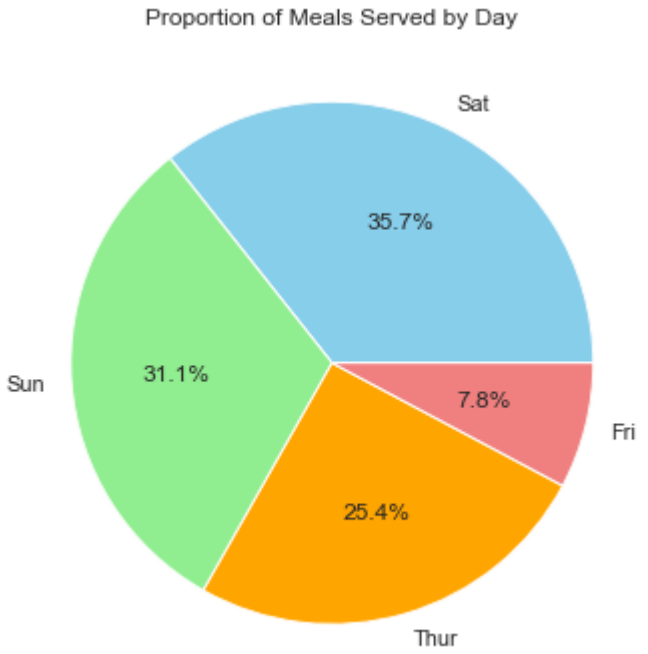
Better when data is **Normally** distributed.

Pie Chart

- Showing the **proportions** of a whole.
- Comparing **parts of a whole**, typically as percentages.
- Displaying simple compositions with **few categories**.

- **Examples**:
  - Percentage of meals served by day.
  - Market share by company.

Numerical over Categories



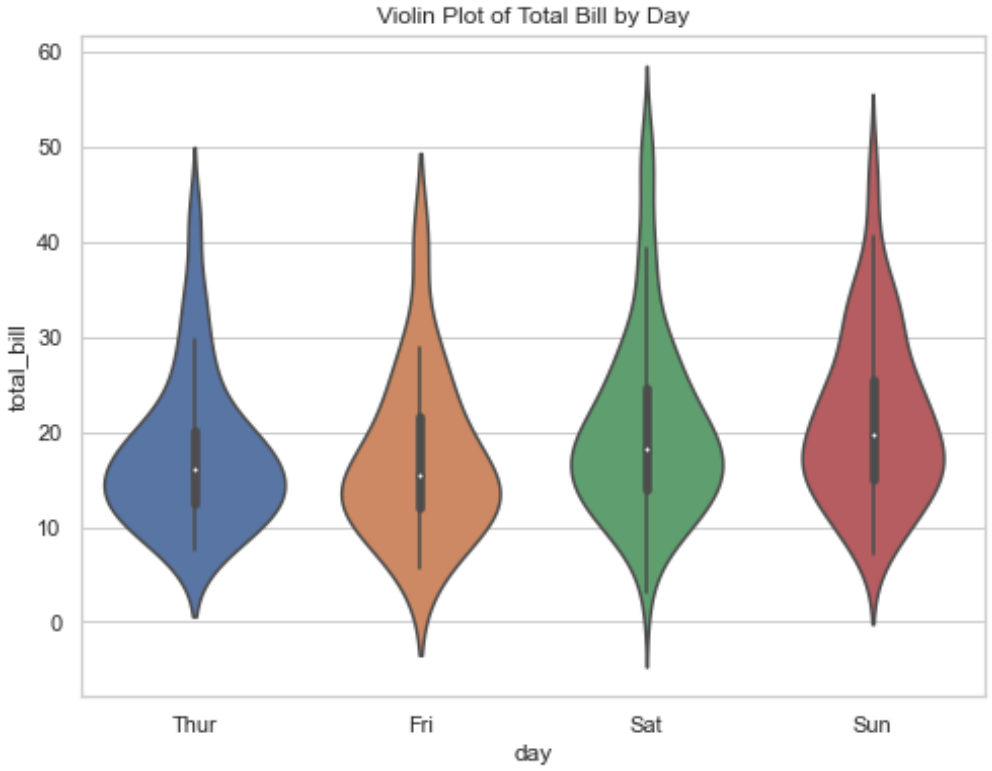
Avoid Using with too many categories

Violin Plot

- Looking for the **density** and **distribution** within categories.

- **Examples**:
  - Total bill by day.
  - Test score distributions across different classes.

Numerical over Categories

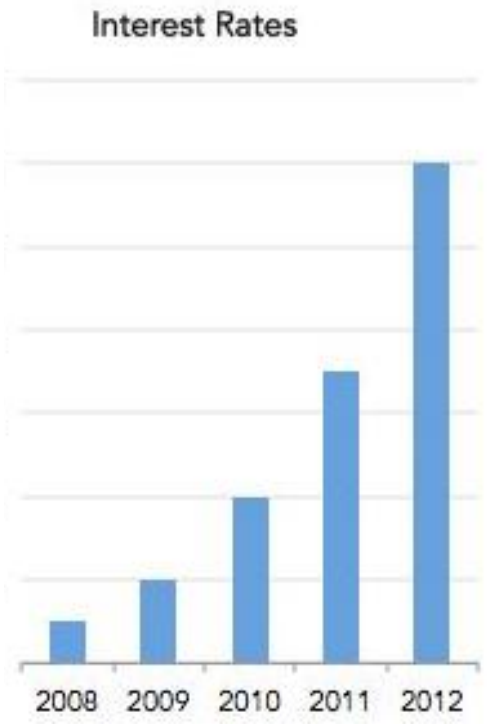


More Detailed than Box Plot

Guidelines Summary

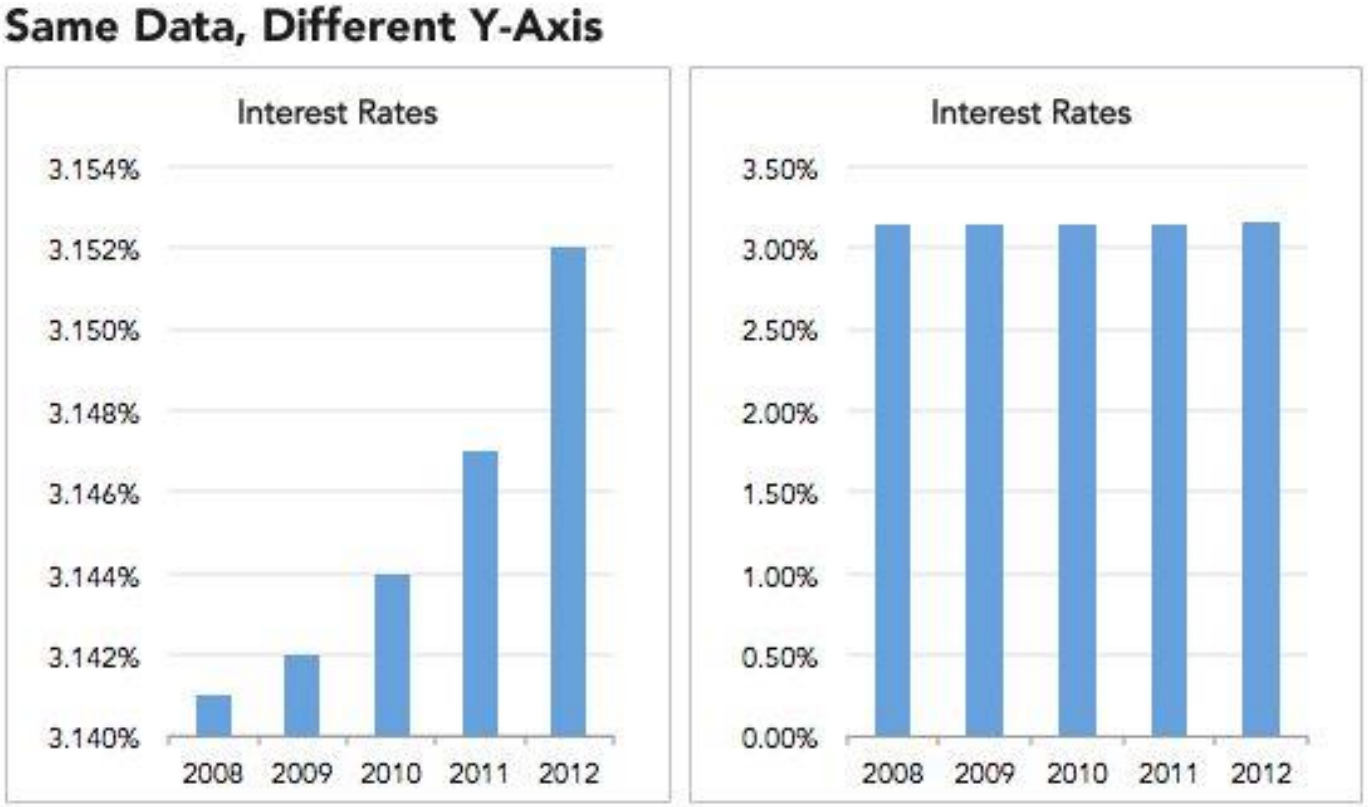
- Comparisons between categories:
  - Bar charts or box plots.
- Showing trends over time:
  - Line charts or area charts.
- Relationships between variables:
  - Scatter plots.
- Distributions:
  - Histograms, box plots, or violin plots.
- Proportions:
  - Pie charts or bar charts.
- Complex relationships:
  - Heatmaps.

Data Visualization



https://i.pinimg.com/564x/0c/ae/61/0cae6187effa5f31171fde69659193eb.jpg

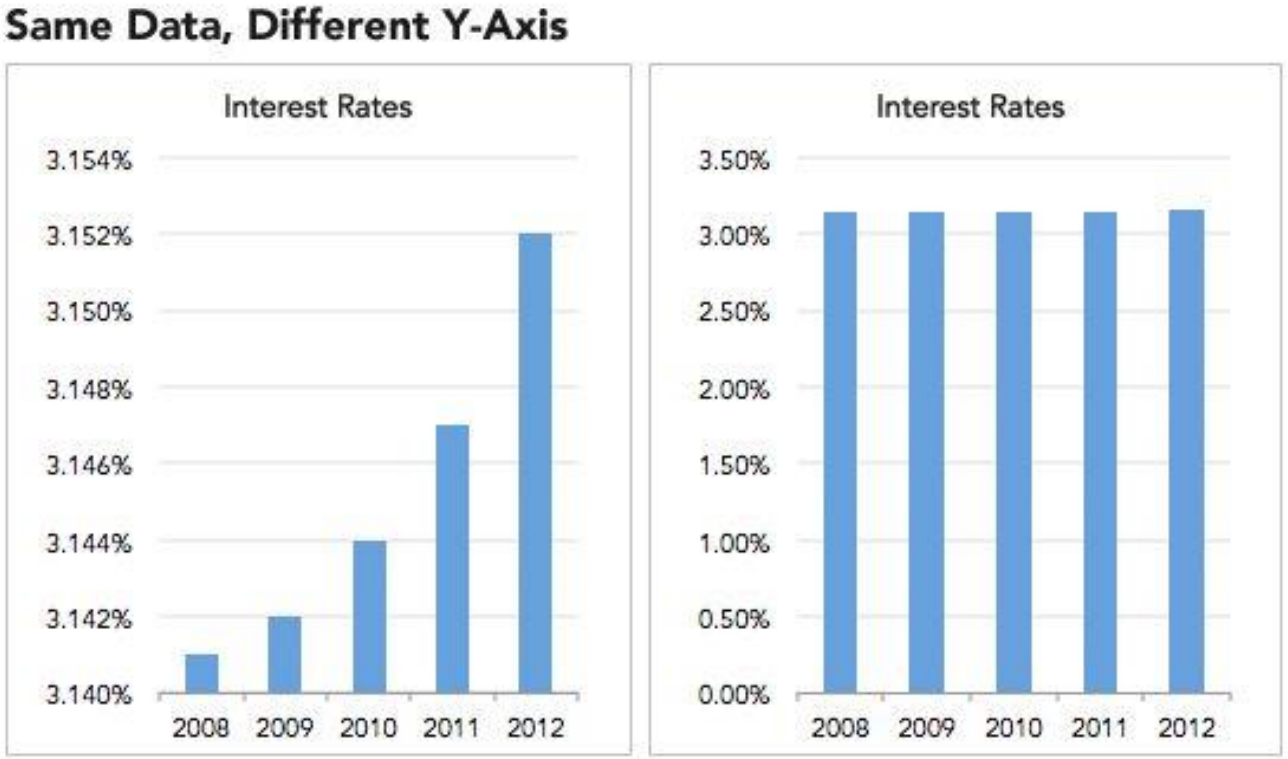
Lying with Data Visualization



https://i.pinimg.com/564x/0c/ae/61/0cae6187effa5f31171fde69659193eb.jpg

Lying with Data Visualization

- Avoiding Misleading Visualizations:
  - Improper scaling
  - Cherry-picking data

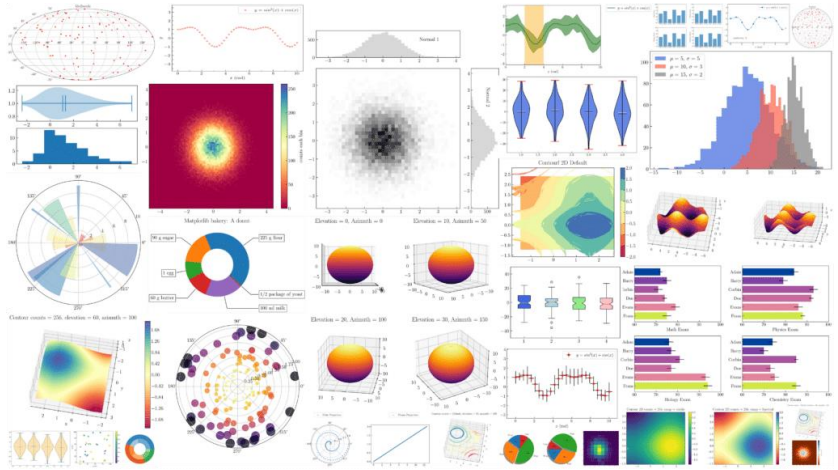


https://i.pinimg.com/564x/0c/ae/61/0cae6187effa5f31171fde69659193eb.jpg

What is Matplotlib?

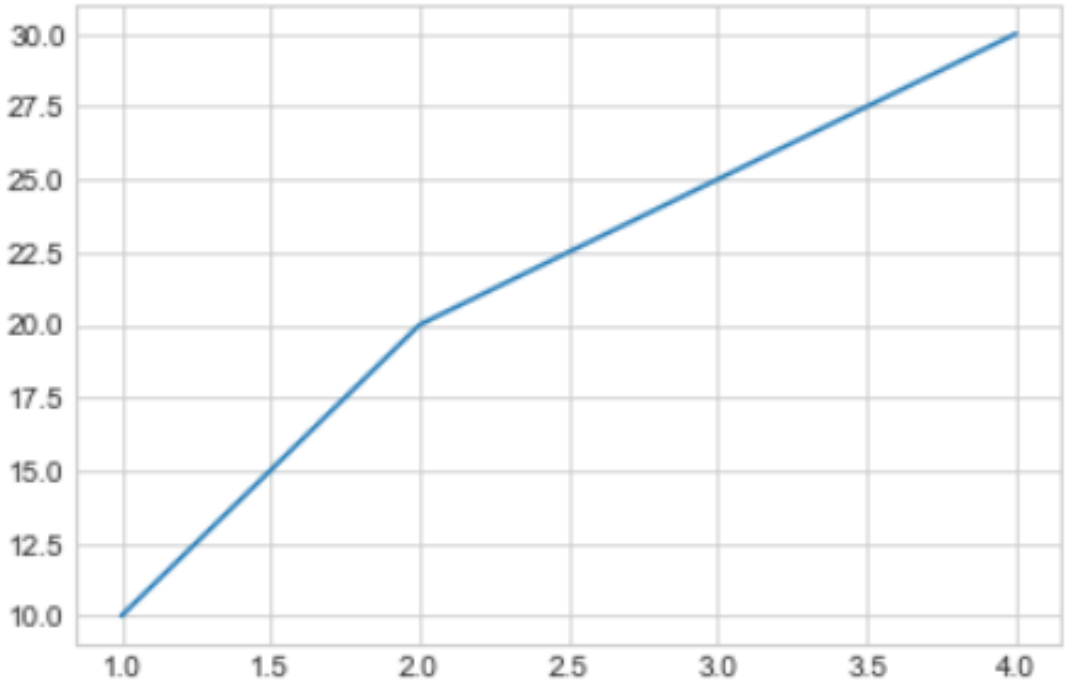
- **Matplotlib** is a Python library for creating **plots**.

- Main points:
  - **Flexible** – various graph types, works with lists, numpy, pandas.
  - **Customizable** – colors, ranges



A Basic Matplotlib Example

```
# Example of a simple line plot
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

# Create a basic plot
plt.plot(x, y)

# Show the plot
plt.show()
```



Matplotlib Flexibility

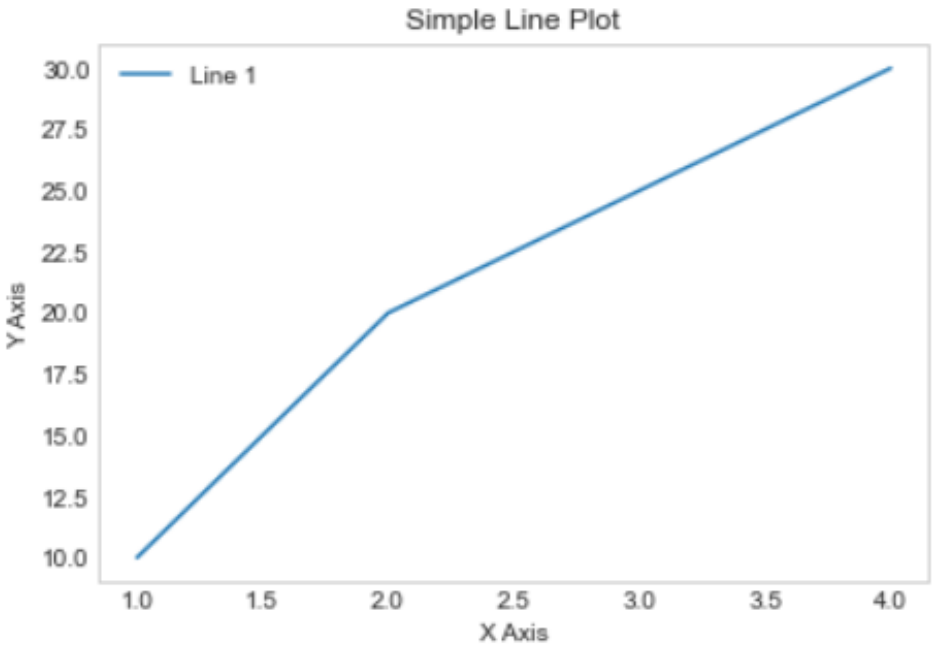
- Supports various types of graphs.

- Line Plot: **plt.plot()**
- Scatter Plot: **plt.scatter()**
- Bar Chart: **plt.bar()**
- Histogram: **plt.hist()**
- Box Plot: **plt.boxplot()**

Matplotlib Customization

- Titles and Labels:
  - plt.title()
  - plt.xlabel()
  - plt.ylabel()
- Legend:
  - plt.legend()
- Grid:
  - plt.grid(True) or plt.grid(False)

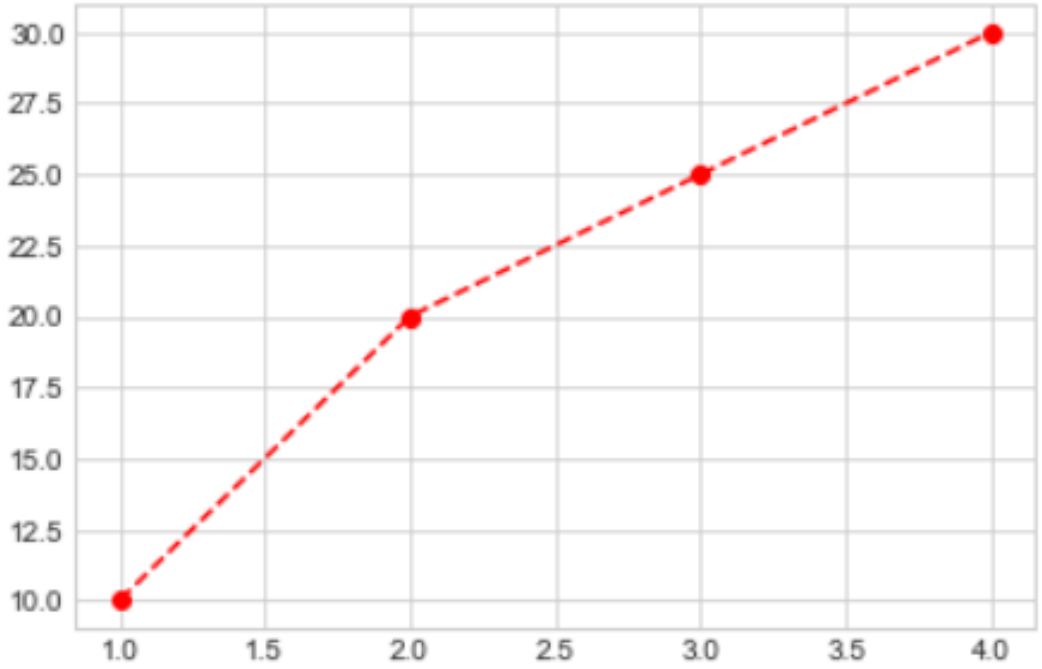
```
# Example of a simple line plot
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

# Create a basic plot
plt.plot(x, y)

# Customize the plot
plt.title("Simple Line Plot")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.legend(["Line 1"],
           loc='upper left')
plt.grid(False)

# Show the plot
plt.show()
```



Matplotlib Customization

- Colors, Line Styles, and Markers.

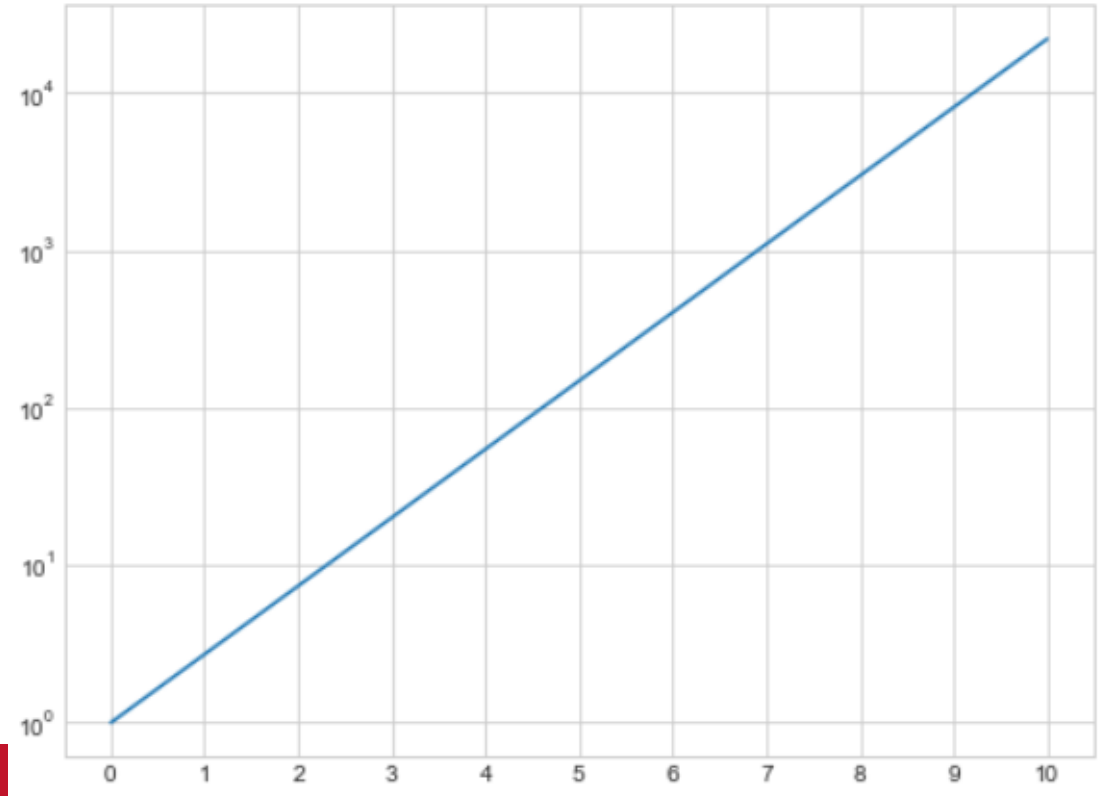
```
# Sample data
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

# Create a basic plot
plt.plot(x, y, marker='o', linestyle='--', color='r')

# other markers: 'o', 's', '^', 'x', 'D', 'p'
# other linestyles: '-', '--', '-.', ':'
# other colors: 'b', 'g', 'r', 'c', 'm', 'y', 'k', 'w'

# Show the plot
plt.show()
```



Matplotlib Customization

- Customizing **ticks** and **scales**.

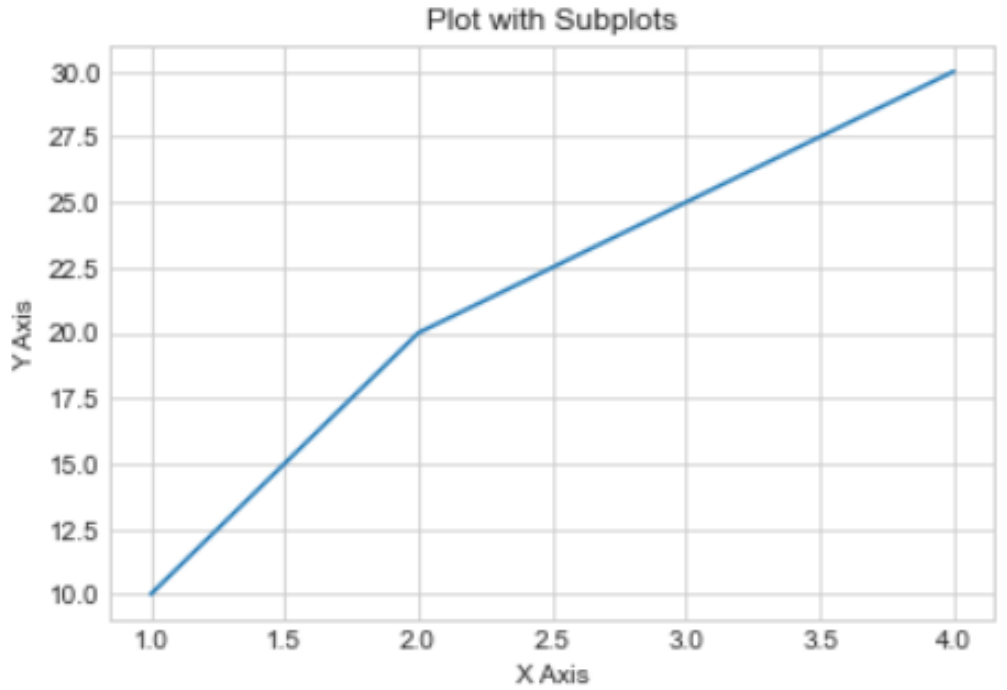
```
# Sample data
x = np.linspace(0, 10, 100)
y = np.exp(x)

# Create the plot
plt.figure(figsize=(8, 6))
plt.plot(x, y, label="Exponential Growth")

# Customizing ticks
# X-axis ticks from 0 to 10 with step 1
plt.xticks(np.arange(0, 11, step=1))
# Custom Y-axis tick values
plt.yticks([0, 5000, 25000, 75000, 200000])
# Setting log scale for y-axis
plt.yscale('log')

# Show the plot
plt.show()
```



Matplotlib Figure and Axes

- **Figure:** The entire plotting area (can contain multiple plots or subplots).
- **Axes:** The individual plot or graph inside a figure (like a subplot).

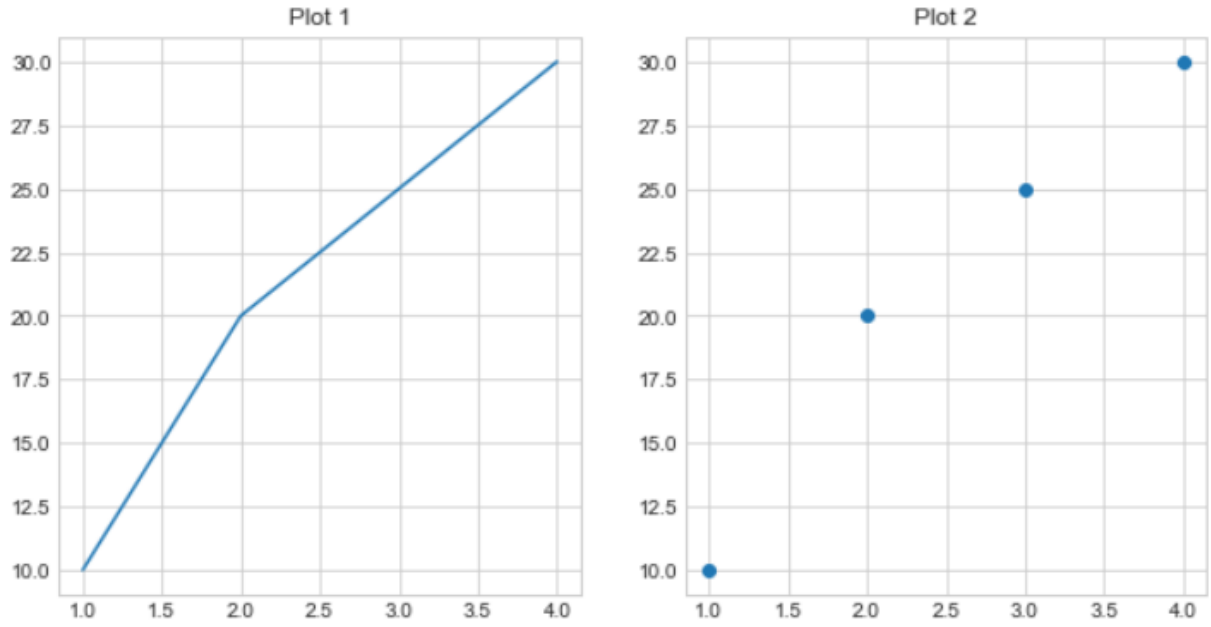
```
# Sample data
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

# Creating a figure and axes
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title("Plot with Subplots")
ax.set_xlabel("X Axis")
ax.set_ylabel("Y Axis")

# Show the plot
plt.show()
```



Subplots

- Create multiple plots in the same figure

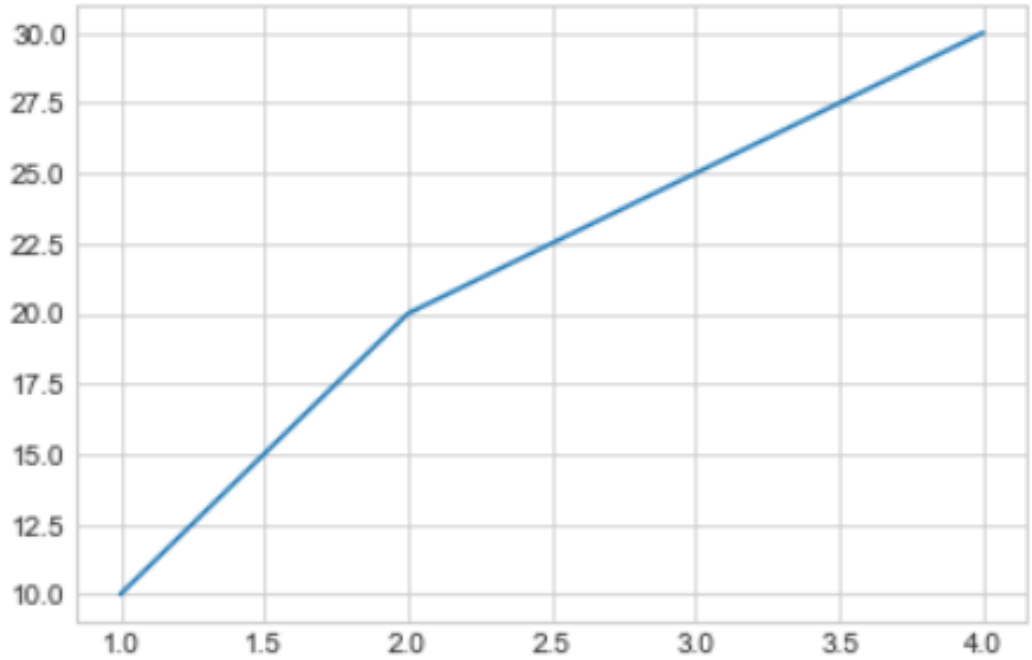
```
# Sample data
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

# Two subplots in one figure
fig, (ax1, ax2) = plt.subplots(1, 2)  # 1 row, 2 columns
# change the figure size
fig.set_size_inches(10, 5)

ax1.plot(x, y)
ax1.set_title("Plot 1")
ax2.scatter(x, y)
ax2.set_title("Plot 2")
plt.show()
```



Saving Plots

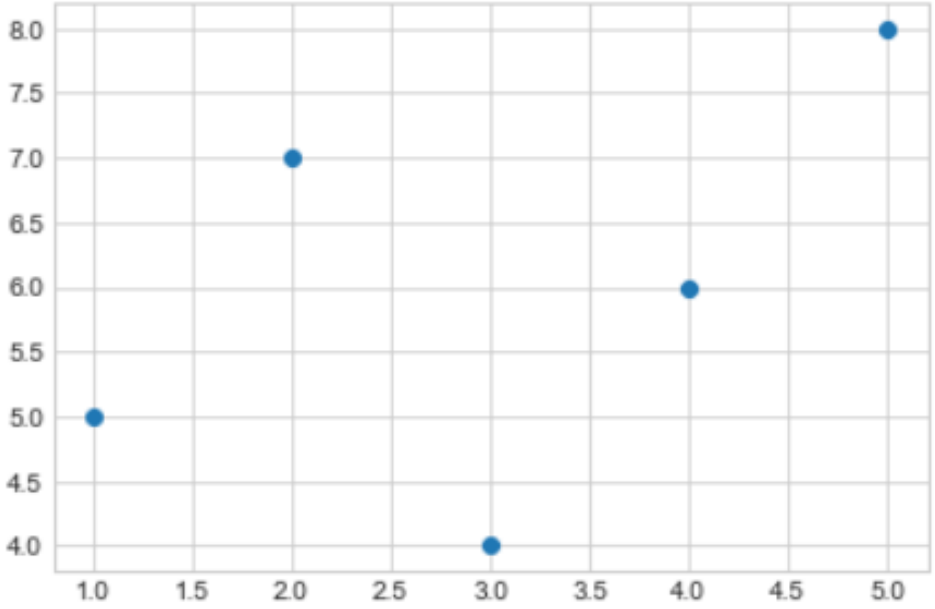
- Save plots to file system – file will be stored at current folder.

```
# Example of a simple line plot
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

# Create a basic plot
plt.plot(x, y)

# saving the plot (need to be before plt.show())
plt.savefig('simple_line_plot.png', # pdf, svg, jpg, etc.
            dpi=300, # resolution
            bbox_inches='tight') # remove extra white space

# Show the plot
plt.show()
```



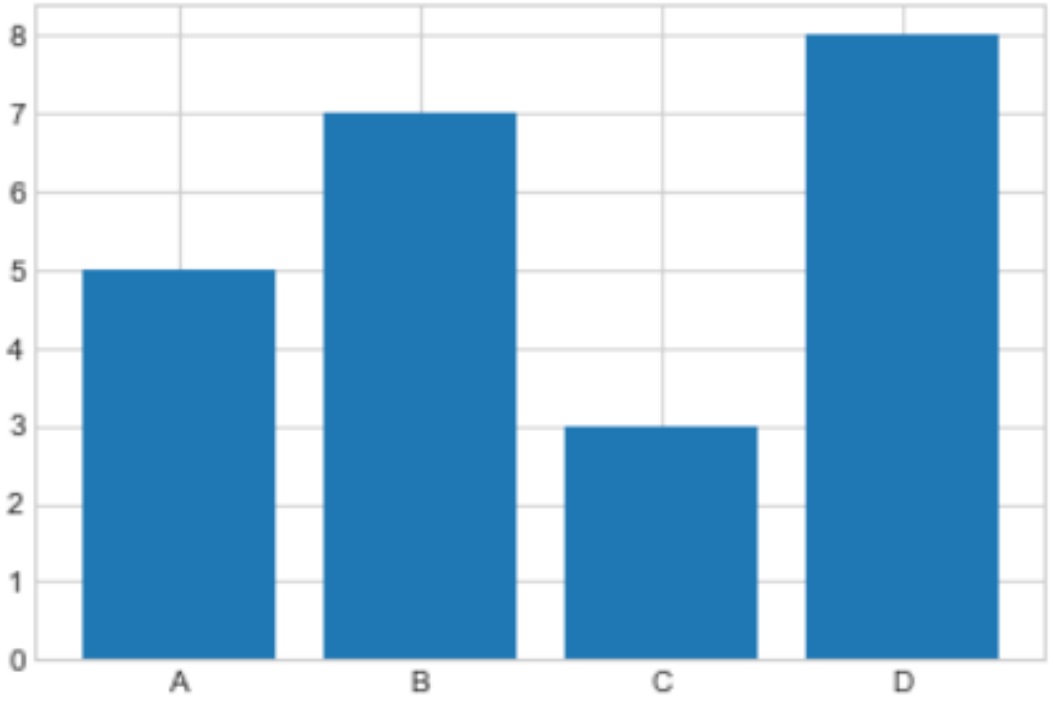
Other Graphs – Scatter Plot

```
# Sample data
x = [1, 2, 3, 4, 5]
y = [5, 7, 4, 6, 8]

# Create a scatter plot
plt.scatter(x, y)
plt.show()
```



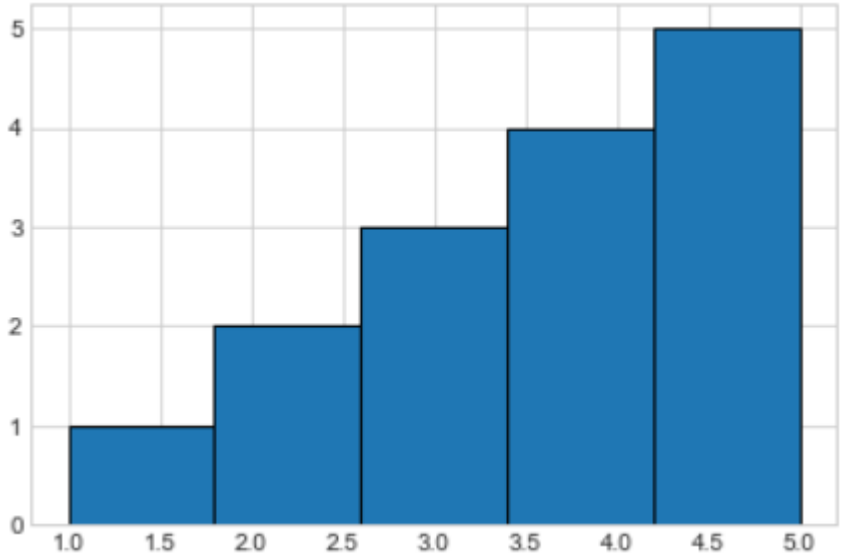
Other Graphs – Bar Chart

```
# Sample data
categories = ['A', 'B', 'C', 'D']
values = [5, 7, 3, 8]

# Create a bar chart
plt.bar(categories, values)
plt.show()
```



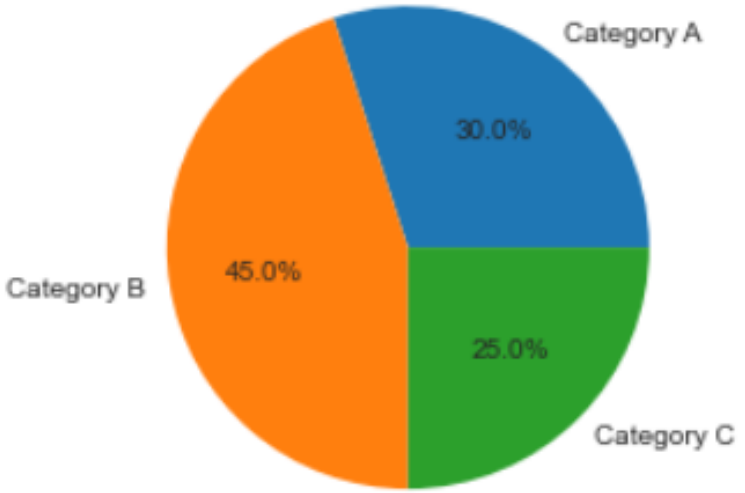
Other Graphs – Histogram

```
# Sample data
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

# Create a histogram
plt.hist(data, bins=5, edgecolor='black')
plt.show()
```



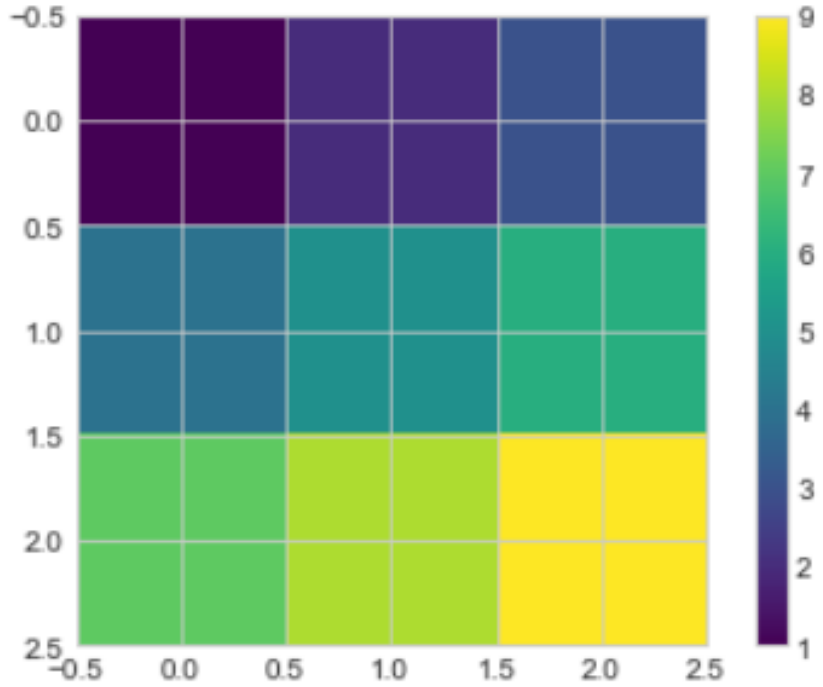
Other Graphs – Pie Chart

```
# Sample data
labels = ['Category A', 'Category B', 'Category C']
sizes = [30, 45, 25]

# Create a pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.show()
```



Other Graphs – Heatmap

```
# Sample data for the heatmap (2D array)
data = np.array([[1, 2, 3],
                 [4, 5, 6],
                 [7, 8, 9]])

# Create a heatmap
plt.imshow(data, cmap='viridis', interpolation='none')

# Add color bar to indicate intensity scale
plt.colorbar()
plt.show()
```



Other Graphs – Box Plot

```
# Sample data
data = [7, 5, 8, 9, 6, 3, 7, 8, 5, 9, 10, 6, 8, 7, 5]

# Create a box plot
plt.boxplot(data)

# Display the plot
plt.show()
```

Combining Pandas and Matplotlib

- **Pandas** leverages **Matplotlib** to create quick, basic visualizations directly from Pandas DataFrames or Series.

Follow Along...

Course Shared Folder - Google Drive
session_4b_datascience_cmu.ipynb

The Tips Dataset

- Information about the **tips** received over a period of a few months in one restaurant.

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

244 rows × 7 columns

- Variables:
  - **tip** in dollars;
  - **total_bill** in dollars;
  - **gender** of the bill payer;
  - whether there were **smokers** in the party;
  - **day** of the week;
  - **time** of day;
  - **size** of the party.

https://rdrr.io/cran/reshape2/man/tips.html

The Tips Dataset

```
df.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object
 3   smoker      244 non-null    object
 4   day         244 non-null    object
 5   time        244 non-null    object
 6   size        244 non-null    int64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

```
df.describe()
```

✓ 0.0s

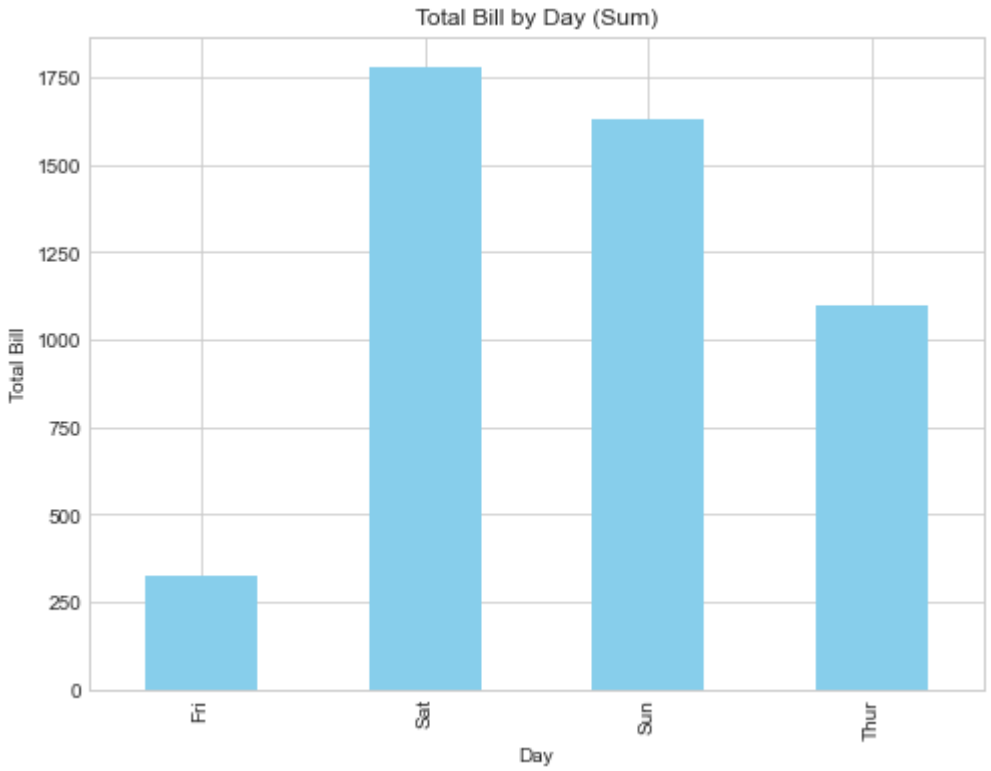
	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

Pandas Bar Chart

- Total bill by day of the week

```
# Bar Chart: Comparing total bill across different days
total_bill_sum = df.groupby('day')['total_bill'].sum()
print(total_bill_sum)

plt.figure(figsize=(8, 6))
total_bill_sum.plot(kind='bar', color='skyblue')
plt.title("Total Bill by Day (Sum)")
plt.xlabel("Day")
plt.ylabel("Total Bill")

plt.show()
```

```
day
Fri      325.88
Sat     1778.40
Sun     1627.16
Thur    1096.33
Name: total_bill, dtype: float64
```

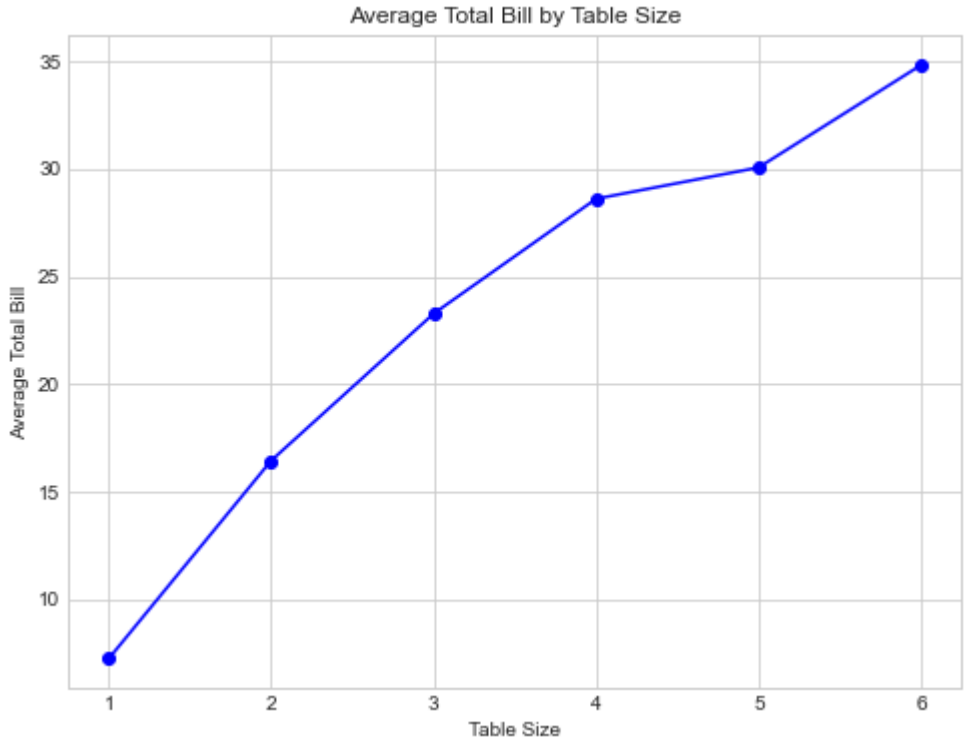


Pandas Line Plot

- Trend of bill over increase in table size

```
# Line Plot: Trends of total bill over table size
plt.figure(figsize=(8, 6))
bill_by_size_mean = df.groupby('size')['total_bill'].mean()
print(bill_by_size_mean)

bill_by_size_mean.plot(kind='line', color='blue', marker='o')
plt.title("Average Total Bill by Table Size")
plt.xlabel("Table Size")
plt.ylabel("Average Total Bill")
plt.show()
```

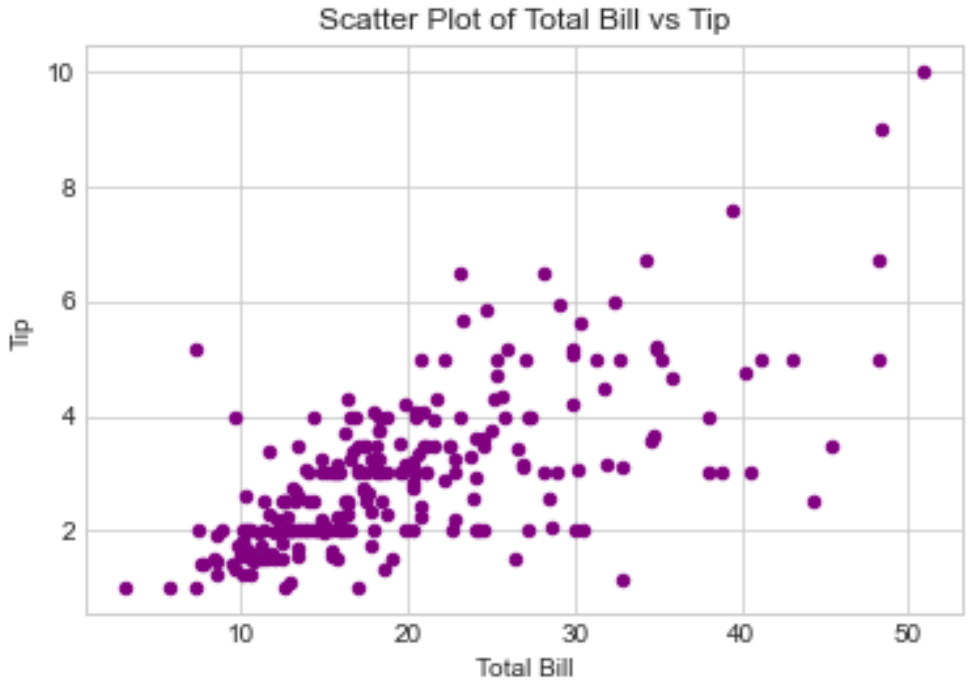
```
size
1     7.242500
2    16.448013
3    23.277632
4    28.613514
5    30.068000
6    34.830000
Name: total_bill, dtype: float64
```



Pandas Scatter Plot

- Relationship between total bill and tip received.

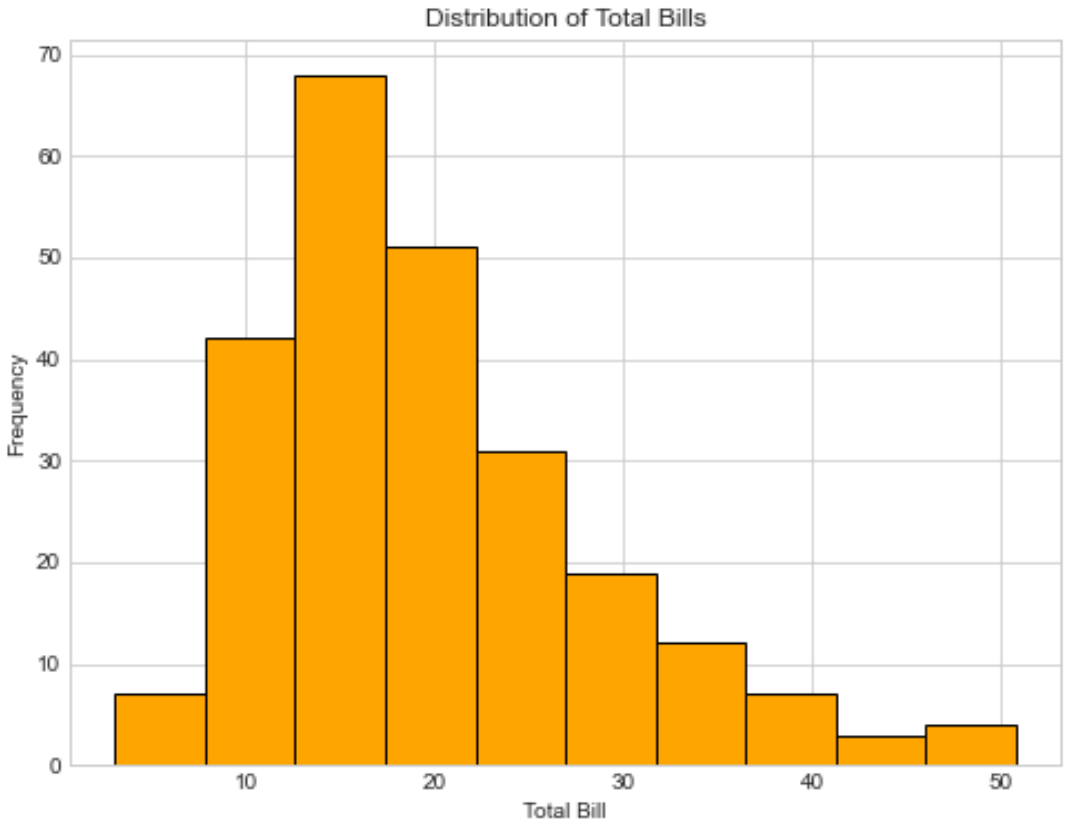
```
# Scatter Plot: Relationship between total bill and tip
plt.figure(figsize=(8, 6))
df.plot(kind='scatter', x='total_bill', y='tip', color='purple')
plt.title("Scatter Plot of Total Bill vs Tip")
plt.xlabel("Total Bill")
plt.ylabel("Tip")
plt.show()
```



Pandas Histogram

- Histogram of bills and their frequency

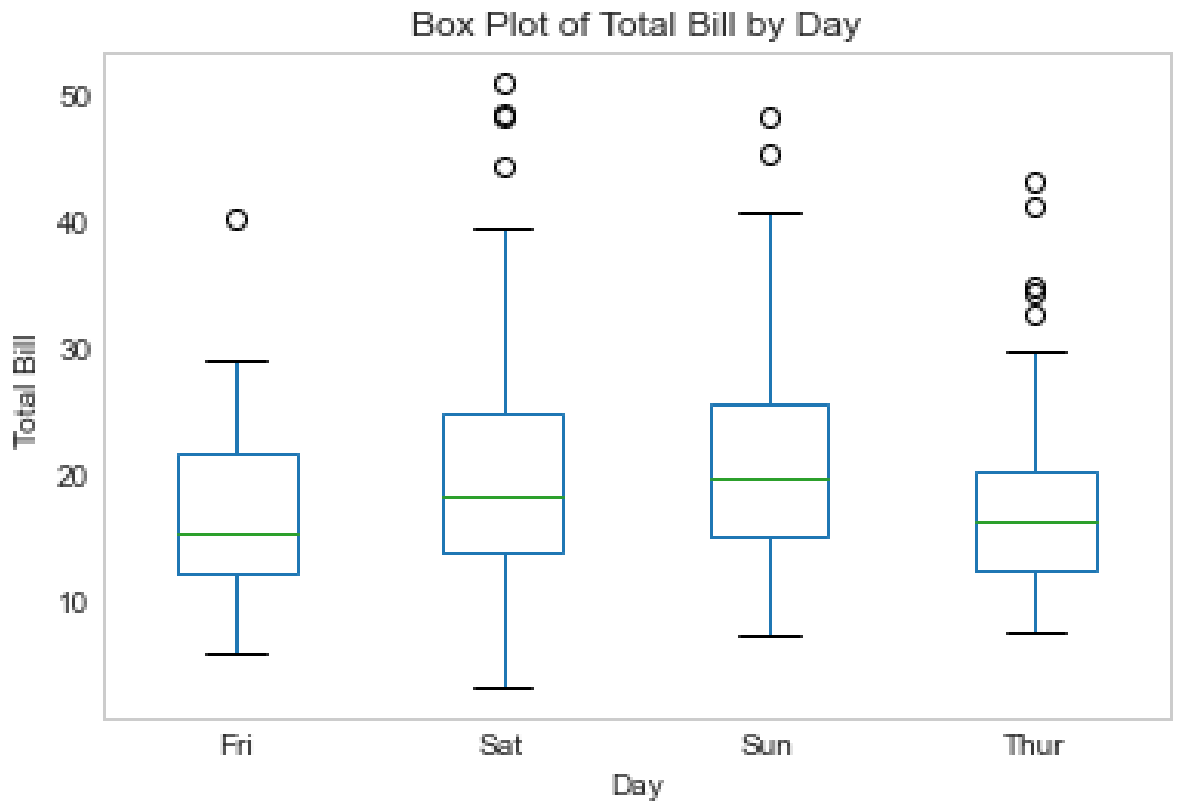
```
# Histogram: Distribution of total bills
plt.figure(figsize=(8, 6))
df['total_bill'].plot(kind='hist', bins=10,
                      color='orange', edgecolor='black')
plt.title("Distribution of Total Bills")
plt.xlabel("Total Bill")
plt.ylabel("Frequency")
plt.show()
```



Pandas Box Plot

- Bills by day

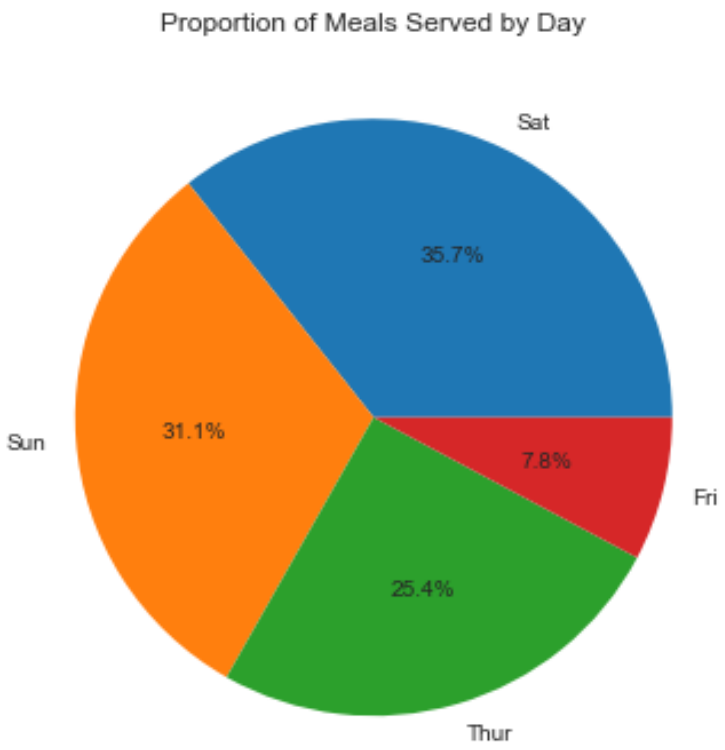
```
# Box Plot: Distribution of total bills by day
plt.figure(figsize=(8, 6))
df.boxplot(column='total_bill', by='day', grid=False)
plt.title("Box Plot of Total Bill by Day")
plt.suptitle('')  # Suppress automatic title
plt.xlabel("Day")
plt.ylabel("Total Bill")
plt.show()
```



Pandas Pie Chart

- Percentage of meals percentage by day of the week.

```
# Pie Chart: Proportion of meals served by day
plt.figure(figsize=(8, 6))
df['day'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title("Proportion of Meals Served by Day")
plt.ylabel('')  # Hide y-label
plt.show()
```



Pandas Heatmap

- Person correlation between all numerical features.:
  - Measure **strength** and **direction** of the linear relationship between two variables
  - **Positive correlation** ($0 < r \leq 1$): Values closer to 1 suggests a strong positive relationship;
  - **Negative correlation** ($-1 \leq r < 0$): Values closer to -1 suggests a strong negative relationship;
  - **No correlation** ($r \approx 0$): Value near 0 imply no linear relationship between the variables.

```
# Correlation between numerical features
plt.figure(figsize=(8, 6))
correlation = df.corr()
print(correlation)
```

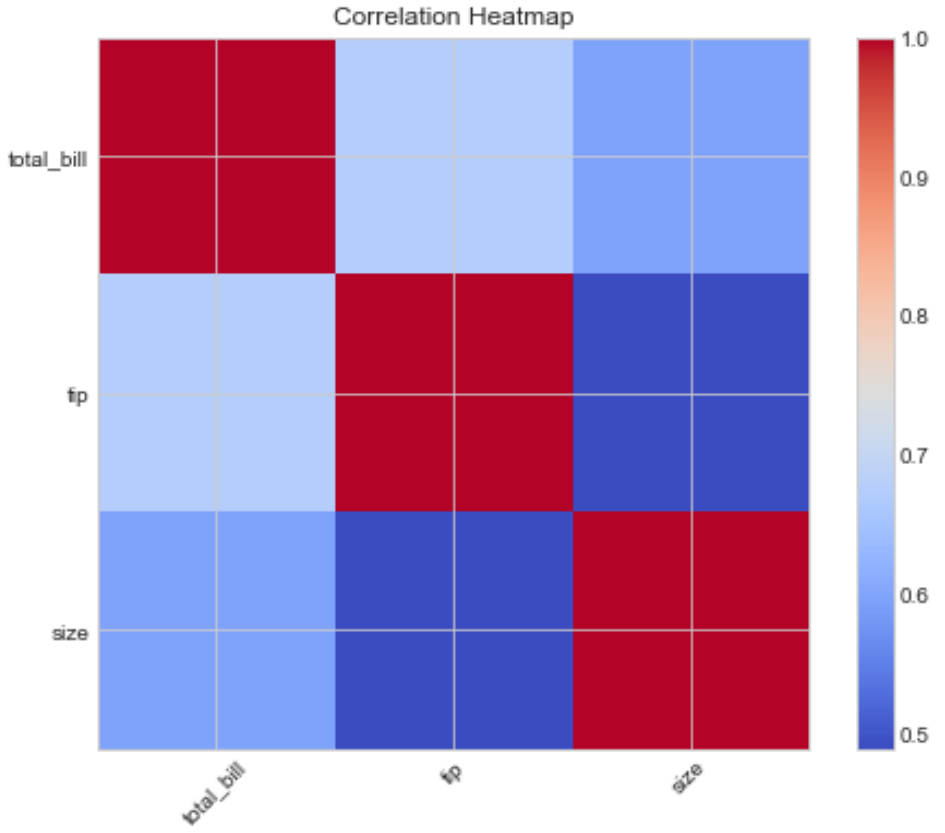
	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

Pandas Heatmap

- Correlation between all numerical variables.

```
# Correlation between numerical features
plt.figure(figsize=(8, 6))
correlation = df.corr()
print(correlation)

plt.imshow(correlation, cmap='coolwarm')
plt.colorbar()
plt.xticks(range(len(correlation)), correlation.columns, rotation=45)
plt.yticks(range(len(correlation)), correlation.columns)
plt.title("Correlation Heatmap")
plt.show()
```

```
              total_bill       tip      size
total_bill      1.000000  0.675734  0.598315
tip             0.675734  1.000000  0.489299
size            0.598315  0.489299  1.000000
```



Why Seaborn?

- Provides an interface for creating graphics:
  - **Higher** level abstraction
  - **Easier** to generate complex plots with minimal code.
  - Better **aesthetic** (by default)

Seaborn and Support For Various Plots

- Bar Chart - sns.barplot()
- Bar Chart with Counts - sns.countplot()
- Line Chart - sns.lineplot()
- Scatter Plot - sns.scatterplot()
- Histogram - sns.histplot().
- Box Plot - sns.boxplot()
- Violin Plot - sns.violinplot()
- Heatmap - sns.heatmap()

Hands-On Session!

Follow Along...

Course Shared Folder - Google Drive

session_4b_datascience_cmu.ipynb

Seaborn Comparison to Pandas – Bar Plot

Pandas Matplotlib

```
# Bar Chart: Comparing total bill across different days
total_bill_sum = df.groupby('day')['total_bill'].sum()
print(total_bill_sum)

plt.figure(figsize=(8, 6))
total_bill_sum.plot(kind='bar', color='skyblue')
plt.title("Total Bill by Day (Sum)")
plt.xlabel("Day")
plt.ylabel("Total Bill")

plt.show()
```

Seaborn

```
# Bar Chart: Comparing total bill across different days
plt.figure(figsize=(8, 6))
sns.barplot(x="day", y="total_bill", data=df,
            estimator=np.sum, color='skyblue', ci=None)
plt.title("Total Bill by Day (Sum)")
plt.show()
```

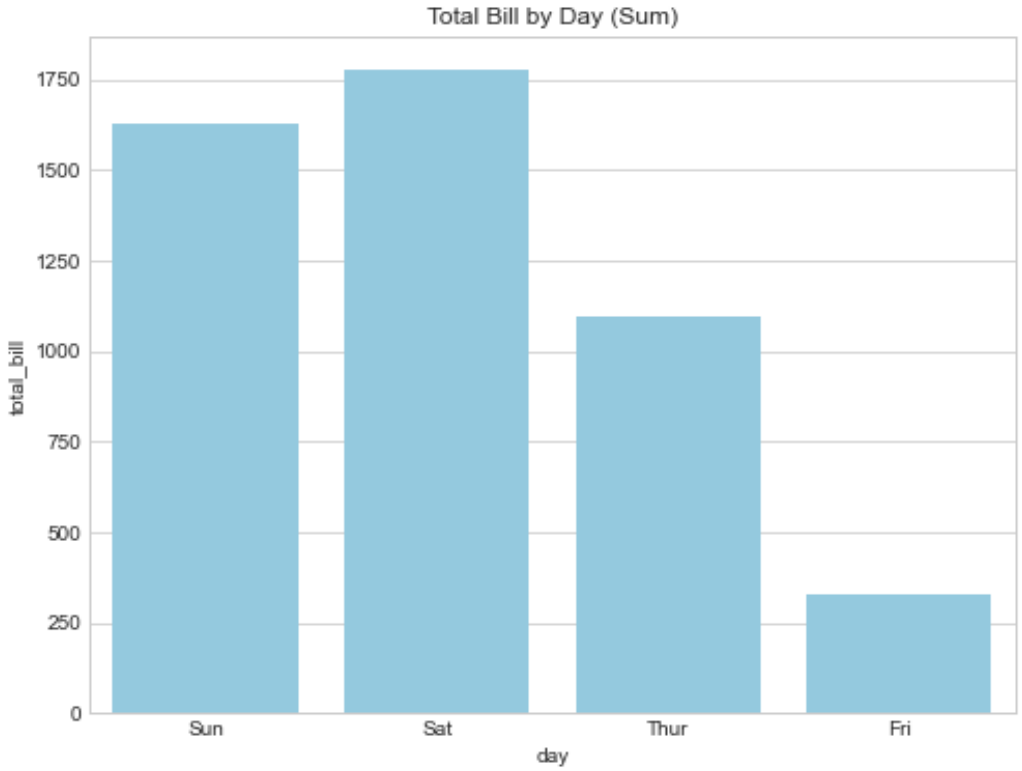
- Performs the **group by** automatically
- Pass any kind of numpy function

Seaborn Comparison to Pandas – Bar Plot

Seaborn

```
# Bar Chart: Comparing total bill across different days
plt.figure(figsize=(8, 6))
sns.barplot(x="day", y="total_bill", data=df,
            estimator=np.sum, color='skyblue', ci=None)
plt.title("Total Bill by Day (Sum)")
plt.show()
```

- Performs the **group by** automatically
- Pass any kind of numpy function



Seaborn Comparison to Pandas – Line Plot

Pandas Matplotlib

```
# Line Plot: Trends of total bill over table size
plt.figure(figsize=(8, 6))
bill_by_size_mean = df.groupby('size')['total_bill'].mean()
print(bill_by_size_mean)

bill_by_size_mean.plot(kind='line', color='blue', marker='o')
plt.title("Average Total Bill by Table Size")
plt.xlabel("Table Size")
plt.ylabel("Average Total Bill")
plt.show()
```

Seaborn

```
# Line Plot: Trends of total bill over table size
plt.figure(figsize=(8, 6))
sns.lineplot(x="size", y="total_bill", data=df, ci=None,
             color='blue', marker='o')
plt.title("Total Bill by Table Size (Trend)")
plt.xlabel("Table Size")
plt.ylabel("Average Total Bill")
plt.show()
```

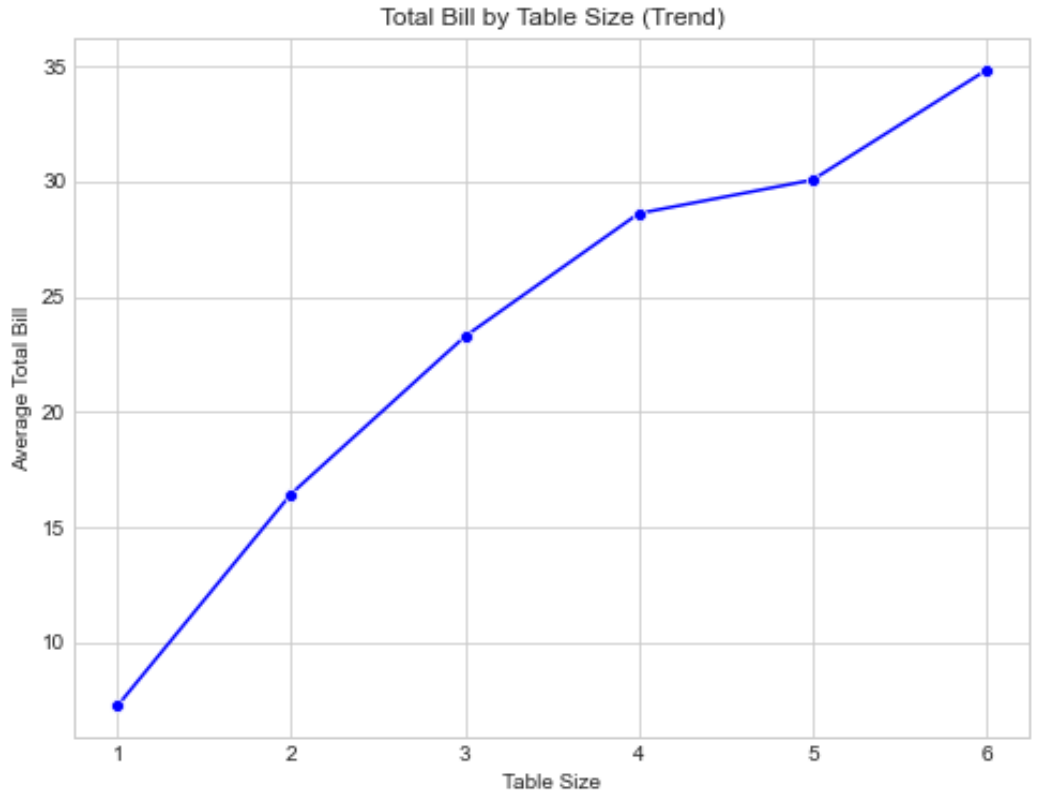
- Performs the **group by** automatically
- Default estimator is mean

Seaborn Comparison to Pandas – Line Plot

Seaborn

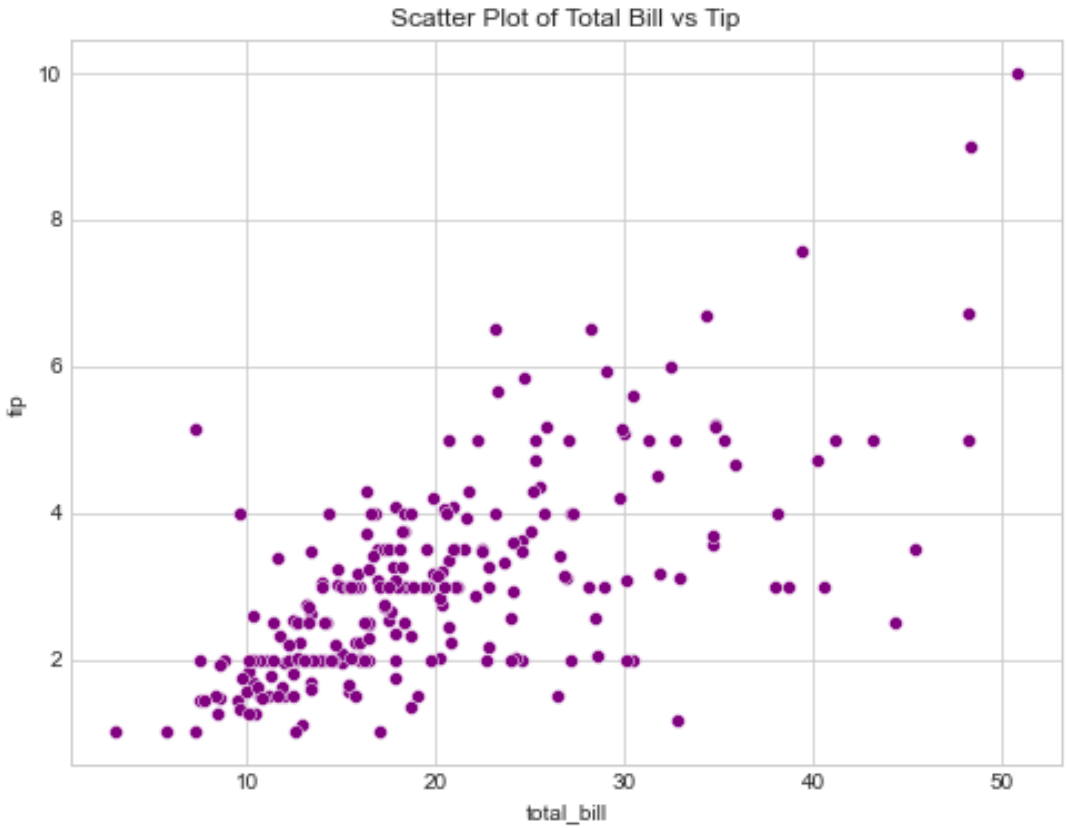
```
# Line Plot: Trends of total bill over table size
plt.figure(figsize=(8, 6))
sns.lineplot(x="size", y="total_bill", data=df, ci=None,
             color='blue', marker='o')
plt.title("Total Bill by Table Size (Trend)")
plt.xlabel("Table Size")
plt.ylabel("Average Total Bill")
plt.show()
```

- Performs the **group by** automatically
- Default estimator is mean



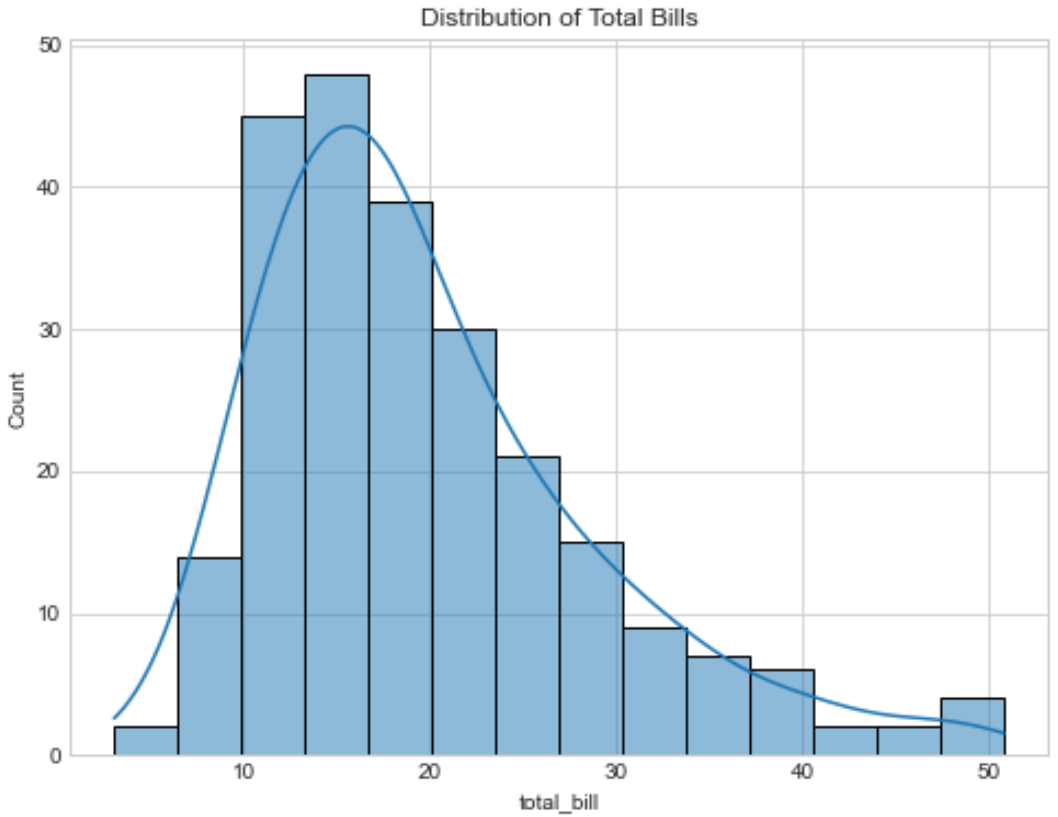
Seaborn – Scatter Plot

```
# Scatter Plot: Relationship between total bill and tip
plt.figure(figsize=(8, 6))
sns.scatterplot(x="total_bill", y="tip", data=df, color='purple')
plt.title("Scatter Plot of Total Bill vs Tip")
plt.show()
```



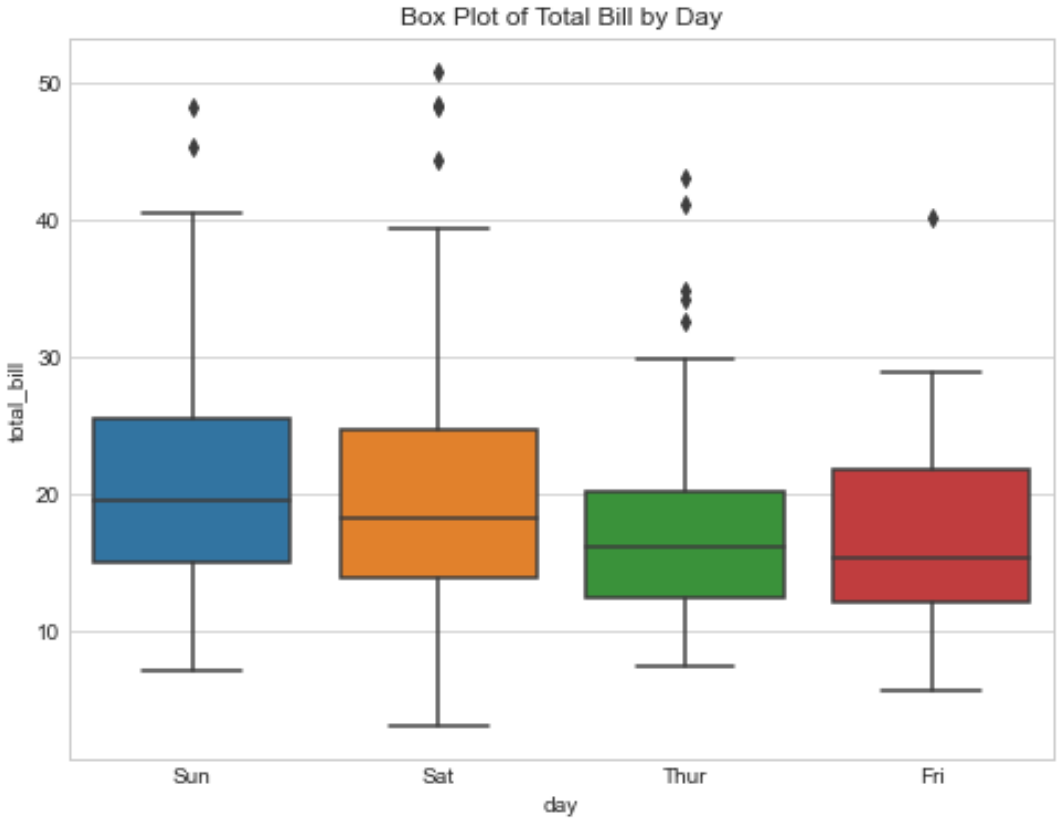
Seaborn – Histogram

```
# Histogram: Distribution of total bills
plt.figure(figsize=(8, 6))
sns.histplot(df['total_bill'], bins="auto", kde=True)
plt.title("Distribution of Total Bills")
plt.show()
```



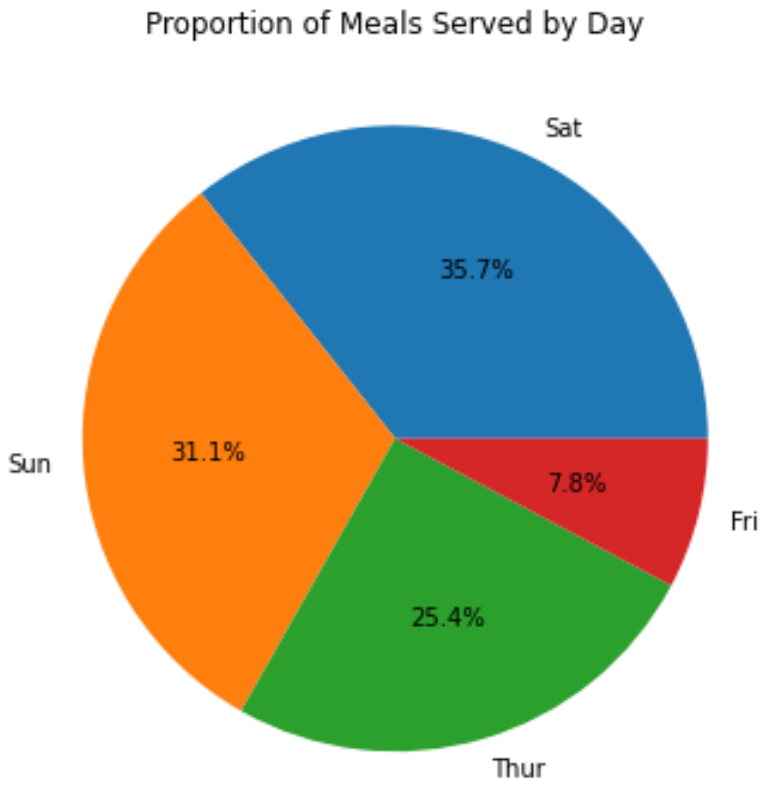
Seaborn – Box Plot

```
# Box Plot: Distribution of total bills by day
plt.figure(figsize=(8, 6))
sns.boxplot(x="day", y="total_bill", data=df)
plt.title("Box Plot of Total Bill by Day")
plt.show()
```



Seaborn – Pie Chart

```
# Pie Chart: Proportion of meals served by day
plt.figure(figsize=(8, 6))
df['day'].value_counts().plot.pie(autopct='%1.1f%%')
plt.title("Proportion of Meals Served by Day")
plt.ylabel('')  # Hide y-label
plt.show()
```



Seaborn Comparison to Pandas – Heatmap

Pandas Matplotlib

```
# Correlation between numerical features
plt.figure(figsize=(8, 6))
correlation = df.corr()
print(correlation)

plt.imshow(correlation, cmap='coolwarm')
plt.colorbar()
plt.xticks(range(len(correlation)), correlation.columns, rotation=45)
plt.yticks(range(len(correlation)), correlation.columns)
plt.title("Correlation Heatmap")
plt.show()
```

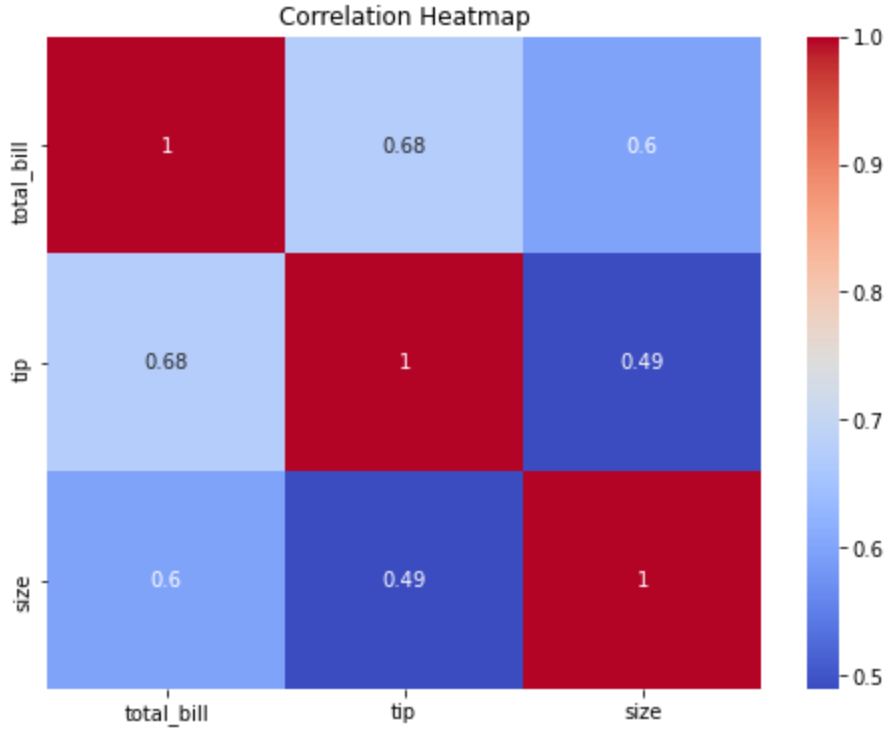
Seaborn

```
# Correlation between numerical features
plt.figure(figsize=(8, 6))
correlation = df.corr()
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

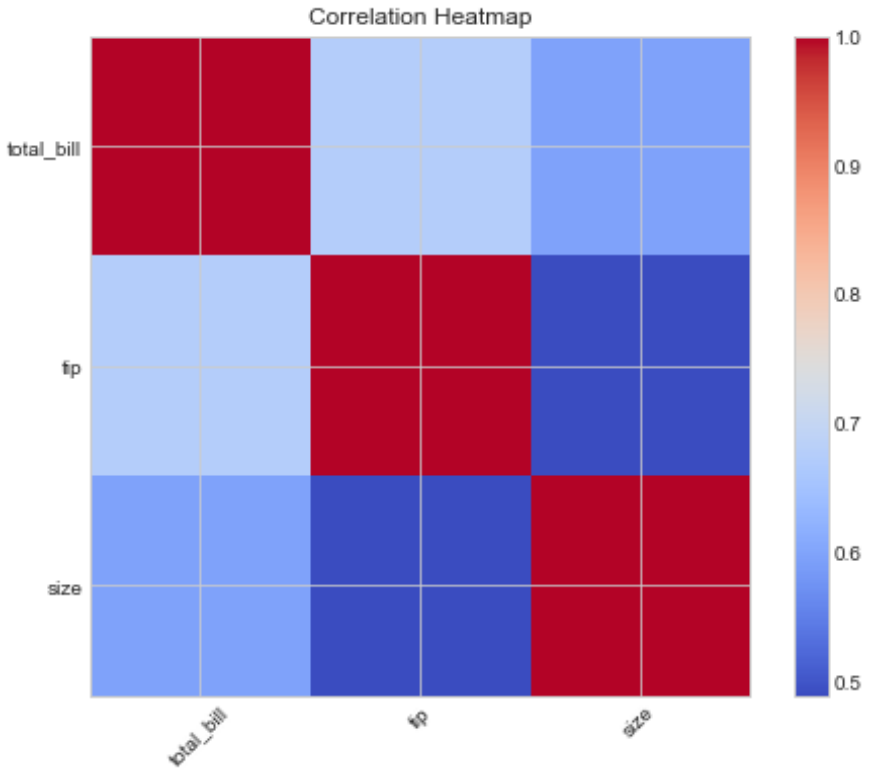
- Performs **ticks** automatically
- Puts **scale (colorbar)** automatically

Seaborn Comparison to Pandas – Heatmap

Seaborn - Better aesthetic



Pandas Matplotlib



Seaborn Comparison to Pandas – Violin Plot

Pandas Matplotlib

```
# Violin Plot: Distribution of total bills by day
plt.figure(figsize=(8, 6))

# Extract unique days dynamically from the dataset
days = df['day'].unique()

# Prepare data for violin plot, grouped by day
data = [df[df['day'] == day]['total_bill'] for day in days]

# Create violin plot
plt.violinplot(data, showmeans=True, showmedians=True)

# Customize the plot
plt.title("Violin Plot of Total Bill by Day")
plt.xlabel("Day")
plt.ylabel("Total Bill")
plt.xticks(range(1, len(days) + 1), days)
plt.show()
```

Seaborn

```
# Violin Plot: Distribution of total bills by day
plt.figure(figsize=(8, 6))
sns.violinplot(x="day", y="total_bill", data=df, color='skyblue')

# Customize the plot
plt.title("Violin Plot of Total Bill by Day")
plt.xlabel("day")
plt.ylabel("total_bill")
plt.show()
```

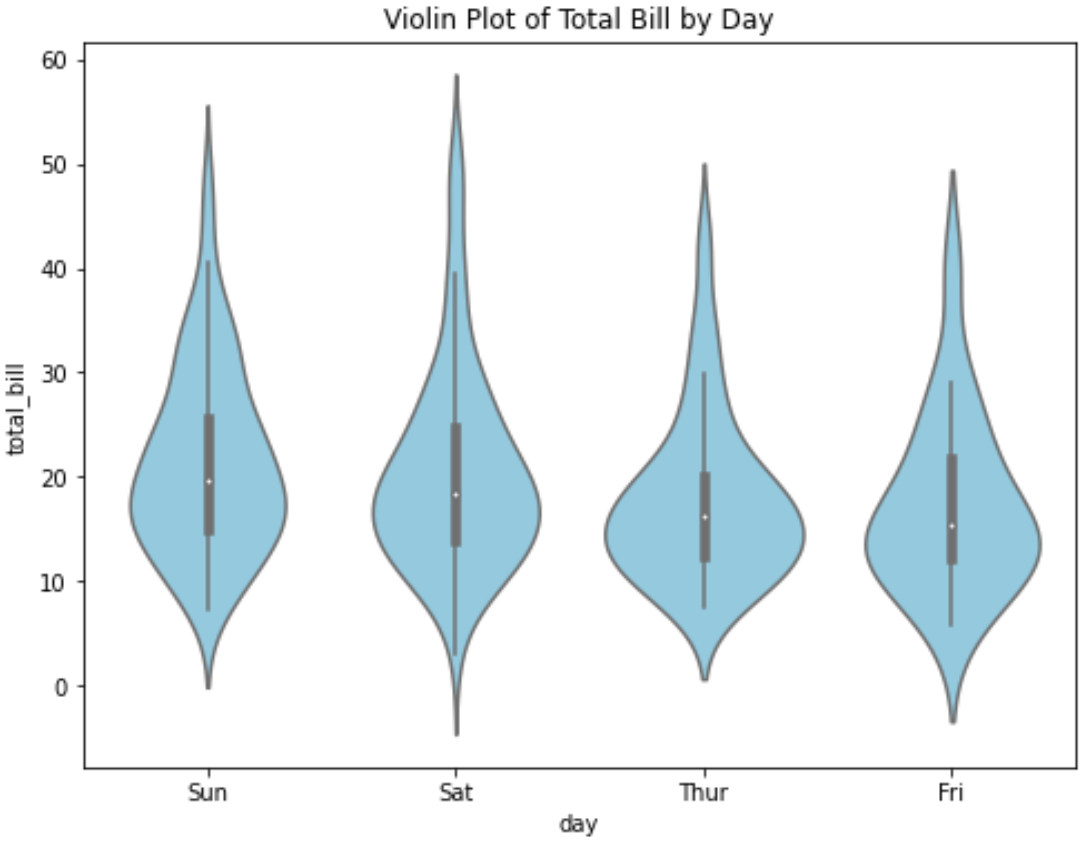
- Performs **group by** automatically
- Puts **ticks** automatically

Seaborn Comparison to Pandas – Violin Plot

Seaborn

```
# Violin Plot: Distribution of total bills by day
plt.figure(figsize=(8, 6))
sns.violinplot(x="day", y="total_bill", data=df, color='skyblue')

# Customize the plot
plt.title("Violin Plot of Total Bill by Day")
plt.xlabel("day")
plt.ylabel("total_bill")
plt.show()
```

- Performs **group by** automatically
- Puts **ticks** automatically



- Use **Seaborn** when you want **quick and attractive** plots with **minimal coding**, good for exploratory data analysis.

- Use **Matplotlib** when you need **more detailed control** over every aspect of the plot.

Plotly?

- **Plotly** is an open-source data visualization library that enables the creation of **interactive, web-based charts**.
- Supports a **wide variety of chart types**, including 3D plots, maps, and statistical graphs.

- **Why Use Plotly?**
  - **Interactivity**: Zoom, hover, and select data points.
  - **Web Integration**: Embed visualizations into web apps or share them online.
  - **Good abstraction level** – similar to seaborn

Follow along...

Course Shared Folder - Google Drive
session_4b_datascience_cmu.ipynb

Conclusion

- In this class, we learned about **data visualization** concepts.
- Selecting the **right type of visualization** is essential to ensure the data conveys the intended message **clearly** and **accurately**.
- We also reviewed popular Python libraries such as **Matplotlib**, **Seaborn**, and **Plotly**, each offering unique strengths.
  - Choosing the appropriate library can improve **efficiency**, **simplify the visualization process**, and **reduce the effort** required for creating insightful plots.

Hands-On Session!

Course Shared Folder - Google Drive
session_4b_datascience_cmu.ipynb - Exercises

Course Use-Case

Apply Data Science tools to work with a specific use-case:

- Groups of 3 students (2 or individual is also possible depending on the case)
- Final presentation in the last course session (18th October)

Course Use-Case

Apply Data Science tools to work with a specific use-case.

Pick an interesting dataset:

- From Kaggle (Link) or UCI (Link)
- Bring your own

Requirements:

- Between 500 to 100k samples
- Tabular file (csv)
- Only categorical and/or numerical features

Use-Case Datasets

- **Potential datasets:**
  - Cost of Living Index by Country – Link
  - Red Wine Quality - Link
  - House Rent Prediction – Link
  - Stellar Classification – Link
  - Sleep Efficiency – Link
  - Healthcare Diabetes Dataset - Link

CMU Portugal Academy

Carnegie Mellon Portugal · TÉCNICO LISBOA · Ciências ULisboa · NOVA SCHOOL OF SCIENCE & TECHNOLOGY