UNIT I INTRODUCTION 9

Data Science: Benefits and uses – facets of data - Data Science Process: Overview – Defining research goals – Retrieving data – Data preparation - Exploratory Data analysis – build the model– presenting findings and building applications - Data Mining - Data Warehousing – Basic Statistical descriptions of Data

Data Science:

Data science is an interdisciplinary field which is focused on extracting knowledge from Big Data, which are typically large, and applying the knowledge and actionable insights from data to solve problems in a wide range of application domains.

1.1 Need for Data Science

- Big data is a huge collection of data with wide variety of different data set and in different formats. It is hard for the conventional management techniques to extract the data of different format and process them.
- Data science involves using methods to analyse massive amounts of data and extract the knowledge it contains.

Characteristics of Big data

- **Volume** - How much data is there?
- **Variety** - How diverse are different types of data?
- **Velocity** - At what speed is new data generated?
- **Veracity** - How accurate is the data?

Benefits & uses of Data Science & Big Data

- Data science and big data are used almost everywhere in both commercial and non-commercial settings.

Example

- Google AdSense, which collects data from internet users so relevant commercial messages can be matched to the person browsing the internet.
- Human resource professionals use people analytics and text mining to screen candidates, monitor the mood of employees, and study informal networks among coworkers.
- Financial institutions use data science to predict stock markets, determine the risk of lending money, and earn how to attract new clients for their services.

- Many governmental organisations not only rely on internal data scientists to discover valuable information, but also share their data with the public.
- Nongovernmental organisations (NGO s) are also no strangers to using data. They use it to raise money and defend their causes.

1.2 Facets of Data

It is used to represent the various forms in which the data could be represented inside Big Data. The following are the various forms in which the data could be represented.

- Structured
- Unstructured
- Natural Language
- Machine Generated
- Graph Based
- Audio, Video & Image
- Streaming Data

1.Structured

Structured data is data that depends on a data model and resides in a fixed field within a record. As such, it's often easy to store structured data in tables within databases or Excel files. SQL , or Structured Query Language, is the preferred way to manage and query data that resides in databases.

	Indicator ID	Dimension List	Timeframe	Numeric Value	Missing Value Flag	Confidence Inte
2	214390830	Total (Age-adjusted)	2008	74.6%		73.8%
3	214390833	Aged 18-44 years	2008	59.4%		58.0%
4	214390831	Aged 18-24 years	2008	37.4%		34.6%
5	214390832	Aged 25-44 years	2008	66.9%		65.5%
6	214390836	Aged 45-64 years	2008	88.6%		87.7%
7	214390834	Aged 45-54 years	2008	86.3%		85.1%
8	214390835	Aged 55-64 years	2008	91.5%		90.4%
9	214390840	Aged 65 years and over	2008	94.6%		93.8%
10	214390837	Aged 65-74 years	2008	93.6%		92.4%
11	214390838	Aged 75-84 years	2008	95.6%		94.4%
12	214390839	Aged 85 years and over	2008	96.0%		94.0%
13	214390841	Male (Age-adjusted)	2008	72.2%		71.1%
14	214390842	Female (Age-adjusted)	2008	76.8%		75.9%
15	214390843	White only (Age-adjusted)	2008	73.8%		72.9%
16	214390844	Black or African American only (Age-adjusted)	2008	77.0%		75.0%
17	214390845	American Indian or Alaska Native only (Age-adjusted)	2008	66.5%		57.1%
18	214390846	Asian only (Age-adjusted)	2008	80.5%		77.7%
19	214390847	Native Hawaiian or Other Pacific Islander only (Age-adjusted)	2008	DSU		
20	214390848	2 or more races (Age-adjusted)	2008	75.6%		69.6%

2.Unstructured

Unstructured data is data that isn't easy to fit into a data model because the content is context-specific or varying.

3.Natural Language

Natural language is a special type of unstructured data; it's challenging to process because it requires knowledge of specific data science techniques and linguistics.

The natural language processing community has had success in entity recognition, topic recognition, summarization, text completion, and sentiment analysis, but models trained in one domain don't generalise well to other domains.

Example: Emails, mails, comprehensions, essays, articles etc..

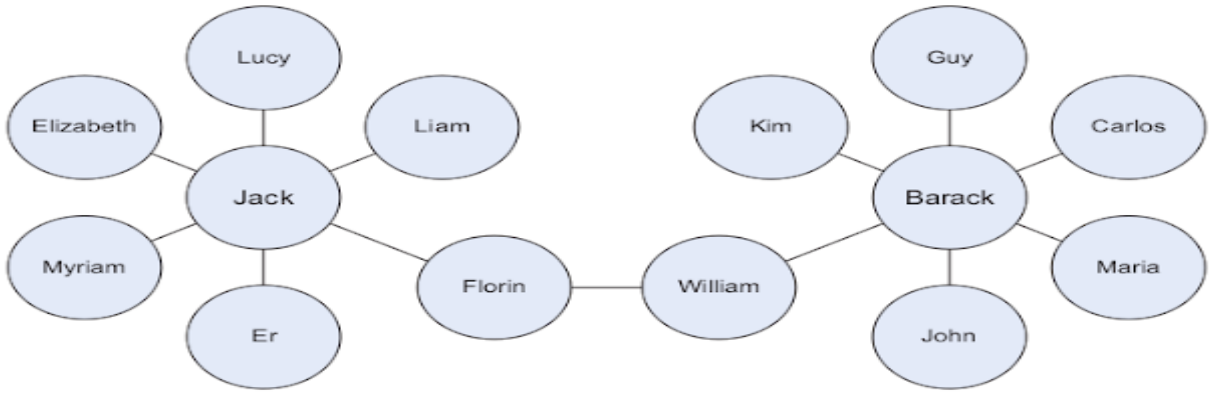
4.Machine Generated

Machine-generated data is information that's automatically created by a computer, process, application, or other machine without human intervention. Machine-generated data is becoming a major data resource and will continue to do so.

```
CSIPERF:TXCOMMIT;313236
2014-11-28 11:36:13, Info                  CSI    00000153 Creating NT transaction (seq
69), objectname [6]"(null)"
2014-11-28 11:36:13, Info                  CSI    00000154 Created NT transaction (seq 69)
result 0x00000000, handle @0x4e54
2014-11-28 11:36:13, Info                  CSI    00000155@2014/11/28:10:36:13.471
Beginning NT transaction commit...
2014-11-28 11:36:13, Info                  CSI    00000156@2014/11/28:10:36:13.705 CSI perf
trace:
CSIPERF:TXCOMMIT;273983
2014-11-28 11:36:13, Info                  CSI    00000157 Creating NT transaction (seq
70), objectname [6]"(null)"
2014-11-28 11:36:13, Info                  CSI    00000158 Created NT transaction (seq 70)
result 0x00000000, handle @0x4e5c
2014-11-28 11:36:13, Info                  CSI    00000159@2014/11/28:10:36:13.764
Beginning NT transaction commit...
2014-11-28 11:36:14, Info                  CSI    0000015a@2014/11/28:10:36:14.094 CSI perf
trace:
CSIPERF:TXCOMMIT;386259
2014-11-28 11:36:14, Info                  CSI    0000015b Creating NT transaction (seq
71), objectname [6]"(null)"
2014-11-28 11:36:14, Info                  CSI    0000015c Created NT transaction (seq 71)
result 0x00000000, handle @0x4e5c
2014-11-28 11:36:14, Info                  CSI    0000015d@2014/11/28:10:36:14.106
Beginning NT transaction commit...
2014-11-28 11:36:14, Info                  CSI    0000015e@2014/11/28:10:36:14.428 CSI perf
trace:
CSIPERF:TXCOMMIT;375581
```

5.Graph Based

"Graph data" can be a confusing term because any data can be shown in a graph. "Graph" in this case points to mathematical graph theory. In graph theory, a graph is a mathematical structure to model pair-wise relationships between objects. Graph or network data is, in short, data that focuses on the relationship or adjacency of objects. The graph structures use nodes, edges, and properties to represent and store graphical data. Graph-based data is a natural way to represent social networks, and its structure allows you to calculate specific metrics such as the influence of a person and the shortest path between two people.



6.Audio, Video & Image

Audio, image, and video are data types that pose specific challenges to a data scientist. Tasks that are trivial for humans, such as recognizing objects in pictures, turn out to be challenging for computers.

Examples: Youtube videos, podcast, music and lots more to add up to.

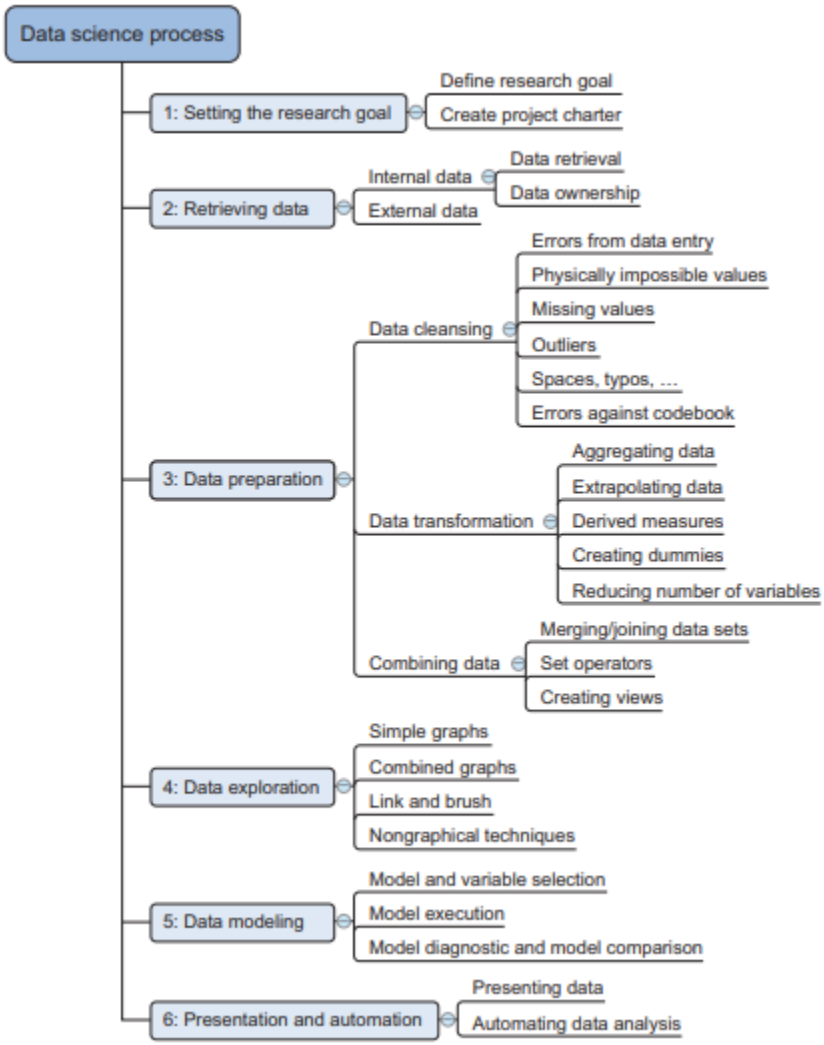
7.Streaming Data

While streaming data can take almost any of the previous forms, it has an extra property. The data flows into the system when an event happens instead of being loaded into a data store in a batch. Although this isn't really a different type of data, we treat it here as such because you need to adapt your process to deal with this type of information.

Examples: Video conferences and live telecasts all work on this basics.

1.3 Data Science Process

The data science process typically consists of six steps



1. **Setting the Research Goal**

The first step of this process is setting a research goal. The main purpose here is making sure all the stakeholders understand the what, how, and why of the project. In every serious project this will result in a project charter.

Defining research goal

1. An essential outcome is the research goal that states the purpose of your assignment in a clear and focused manner.
2. Understanding the business goals and context is critical for project success.

Create project charter

A project charter requires teamwork, and your input covers at least the following:

- A clear research goal

- The project mission and context
- How you're going to perform your analysis
- What resources you expect to use
- Proof that it's an achievable project, or proof of concepts
- Deliverables and a measure of success
- A timeline

2. Retrieving Data

The second phase is data retrieval. You want to have data available for analysis, so this step includes finding suitable data and getting access to the data from the data owner. The result is data in its raw form, which probably needs polishing and transformation before it becomes usable.

Data can be stored in many forms, ranging from simple text files to tables in a database. The objective now is acquiring all the data you need. This may be difficult, and even if you succeed, data is often like a diamond in the rough: it needs polishing to be of any use to you.Start with data stored within the company.The data stored in the data might be already cleaned and maintained inrepositories such as databases, data marts, data warehouses and data lakes.

Don't be afraid to shop around

- If the data is not available inside the organization, look outside your organization walls.

Do data quality checks now to prevent problems later

- Always double check while storing your data if it is an internal data. If it is an external data prepare the data such a way that it could be easily extracted.

3. Data Preparation

The data preparation involves Cleansing, Integrating and transforming data

Cleansing Data

Data cleansing is a sub process of the data science process that focuses on removing errors in your data so your data becomes a true and consistent representation of the processes it originates from.

- **Interpretation error -** Example a age of a person can be greater than 120
- **Inconsistencies-** Example is mentioning the Gender as Female in one column and F in another column but both tend to mention the same

Data Entry Errors - Data collection and data entry are error-prone processes. They often require human intervention, and because humans are only human, they make typos or lose their concentration for a second and introduce an error into the chain.

Redundant White space - White spaces tend to be hard to detect but cause errors like other redundant characters would. White spaces at the beginning of a word or at a end of a word is much hard to identify and rectify.

Impossible values and sanity checks - Here the data are checked for physically and theoretically impossible values.

Outliers - Here the data are checked for physically and theoretically impossible values. An outlier is an observation that seems to be distant from other observations or, more specifically, one observation that follows a different logic or generative process than the other observations.

Dealing with the Missing values - Missing values aren't necessarily wrong, but you still need to handle them separately; certain modelling techniques can't handle missing values.

Techniques used to handle missing data are given below

Technique	Advantage	Disadvantage
Omit the values	Easy to perform	You lose the information from an observation
Set value to `null`	Easy to perform	Not every modeling technique and/or implementation can handle `null` values
Impute a static value such as 0 or the mean	Easy to perform You don't lose information from the other variables in the observation	Can lead to false estimations from a model
Impute a value from an estimated or theoretical distribution	Does not disturb the model as much	Harder to execute You make data assumptions
Modeling the value (nondependent)	Does not disturb the model too much	Can lead to too much confidence in the model Can artificially raise dependence among the variables Harder to execute You make data assumptions

Correct as early as possible

- Decision-makers may make costly mistakes on information based on incorrect data from applications that fail to correct for the faulty data.
- If errors are not corrected early on in the process, the cleansing will have to be done for every project that uses that data.
- Data errors may point to defective equipment, such as broken transmission lines and defective sensors.
- Data errors can point to bugs in software or in the integration of software that may be critical to the company.

Combining data from different sources

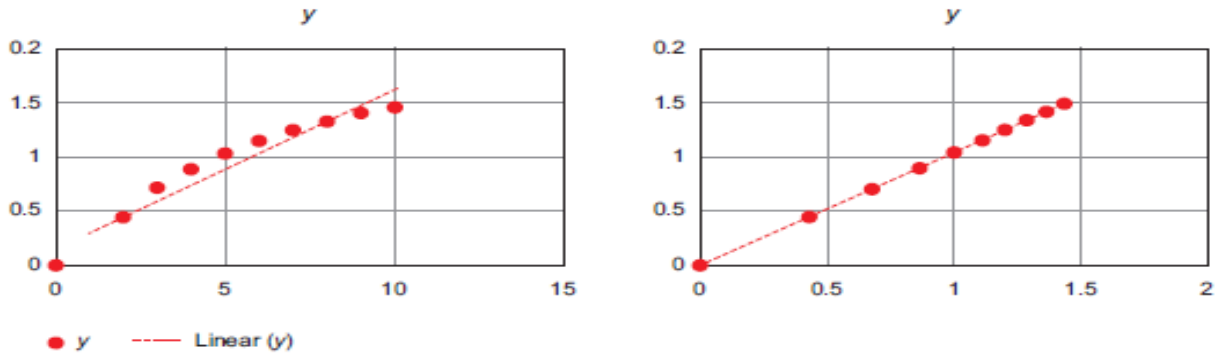
- Data from different model can be combined and stored together for easy cross reference.
- There are different ways of combining the data.
  - Joining Tables
  - Appending Tables
  - Using views to simulate data joins and appends

Transforming Data

Certain models require their data to be in a certain shape.

Data Transformation - Converting a data from linear data into sequential or continuous form of data

x	1	2	3	4	5	6	7	8	9	10
log(x)	0.00	0.43	0.68	0.86	1.00	1.11	1.21	1.29	1.37	1.43
y	0.00	0.44	0.69	0.87	1.02	1.11	1.24	1.32	1.38	1.46



Reducing the number of variables - Having too many variables in your model makes the model difficult to handle, and certain techniques don't perform well when you overload them with too many input variables.

Turning variables into Dummies

Customer	Year	Gender	Sales
1	2015	F	10
2	2015	M	8
1	2016	F	11
3	2016	M	12
4	2017	F	14
3	2017	M	13

M F

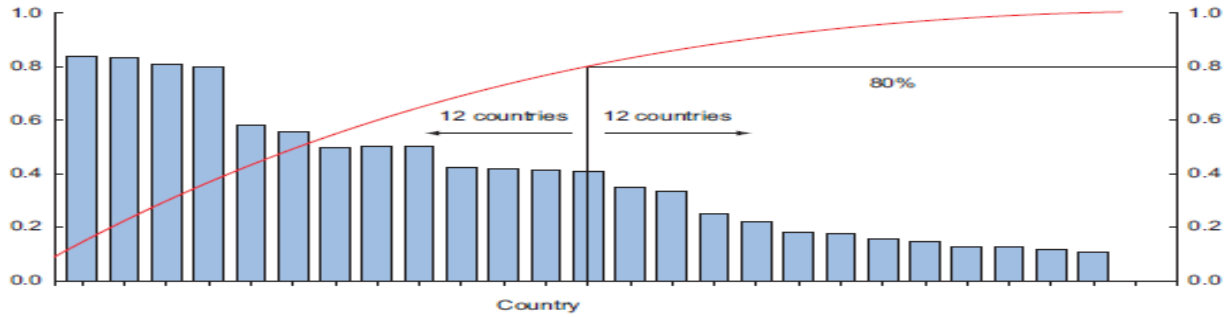
Customer	Year	Sales	Male	Female
1	2015	10	0	1
1	2016	11	0	1
2	2015	8	1	0
3	2016	12	1	0
3	2017	13	1	0
4	2017	14	0	1

4. Data Exploration

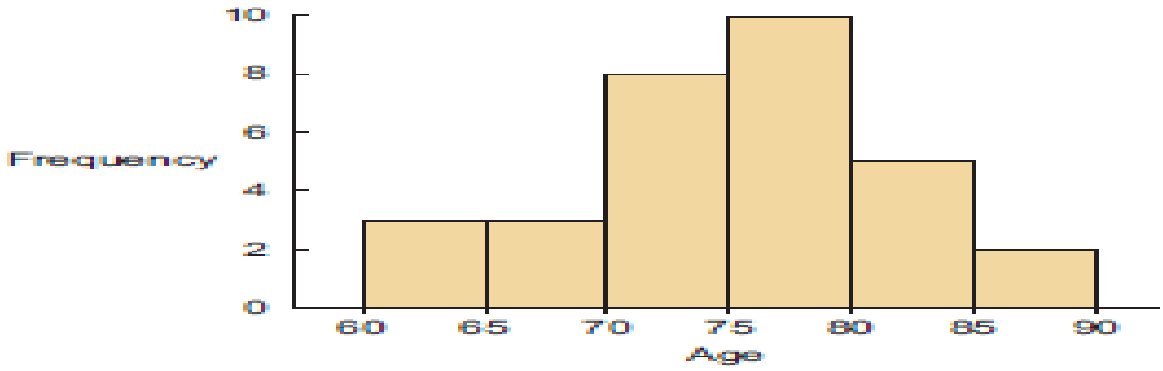
Information becomes much easier to grasp when shown in a picture, therefore you mainly use graphical techniques to gain an understanding of your data and the interactions between variables.

Examples

Pareto diagram is a combination of the values and a cumulative distribution.



Histogram : In it, a variable is cut into discrete categories and the number of occurrences in each category are summed up and shown in the graph.



Boxplot : It doesn't show how many observations are present but does offer an impression of the distribution within categories. It can show the maximum, minimum, median, and other characterising measures at the same time.

5. Data Modelling or Model Building

With clean data in place and a good understanding of the content, you're ready to build models with the goal of making better predictions, classifying objects, or gaining an understanding of the system that you're modelling.

Building a model is an iterative process. Most models consist of the following main steps:

- Selection of a modelling technique and variables to enter in the model
- Execution of the model
- Diagnosis and model comparison

Model and variable selection

The model has to be built upon the following aspects

- Must the model be moved to a production environment and, if so, would it be easy to implement?
- How difficult is the maintenance on the model: how long will it remain relevant if left untouched?
- Does the model need to be easy to explain?

Model Execution - Once you've chosen a model you'll need to implement it in code.

Listing 2.1 Executing a linear prediction model on semi-random data

```
import statsmodels.api as sm
import numpy as np
predictors = np.random.random(1000).reshape(500,2)
target = predictors.dot(np.array([0.4, 0.6])) + np.random.random(500)
lmRegModel = sm.OLS(target,predictors)
result = lmRegModel.fit()
result.summary()
```

Imports required Python modules.

Creates random data for predictors (x-values) and semi-random data for the target (y-values) of the model. We use predictors as input to create the target so we infer a correlation here.

Fits linear regression on data.

Shows model fit statistics.

6. Presentation and Automation

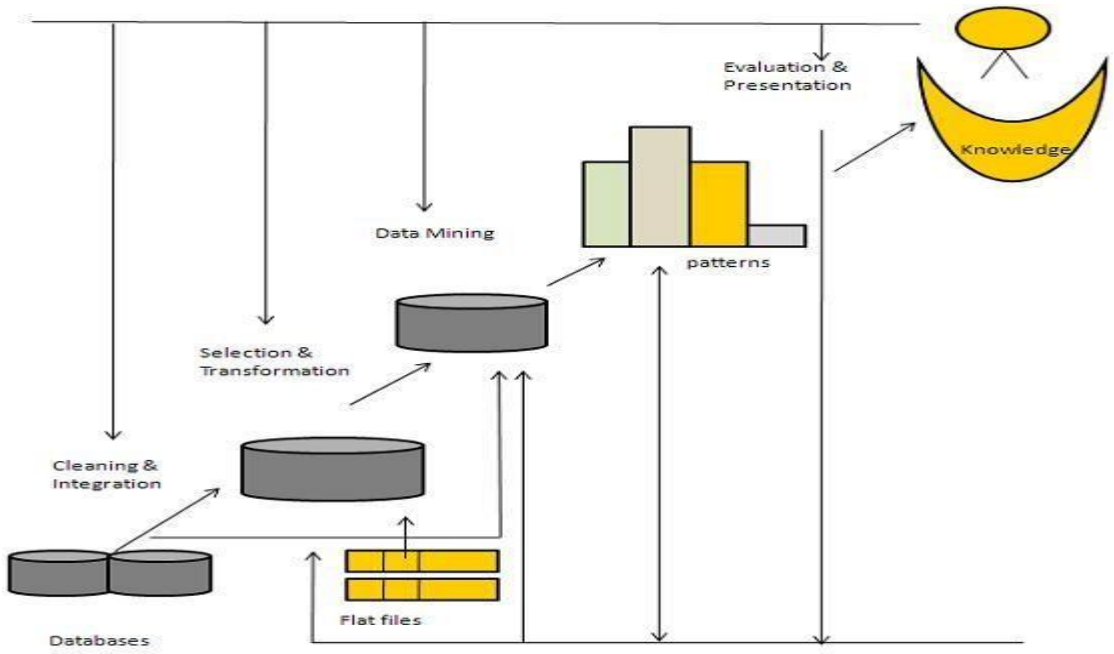
After you've successfully analysed the data and built a well-performing model, you're ready to present your findings to the world. This is an exciting part all your hours of hard work have paid off and you can explain what you found to the stakeholders.

1.4 Data Mining

Data mining refers to extracting or mining knowledge from large amountsof data. The term is actually a misnomer. Thus, data miningshould have been more appropriately named as knowledge mining which emphasis on mining from large amounts of data.

Knowledge Discovery in Databases(KDD)

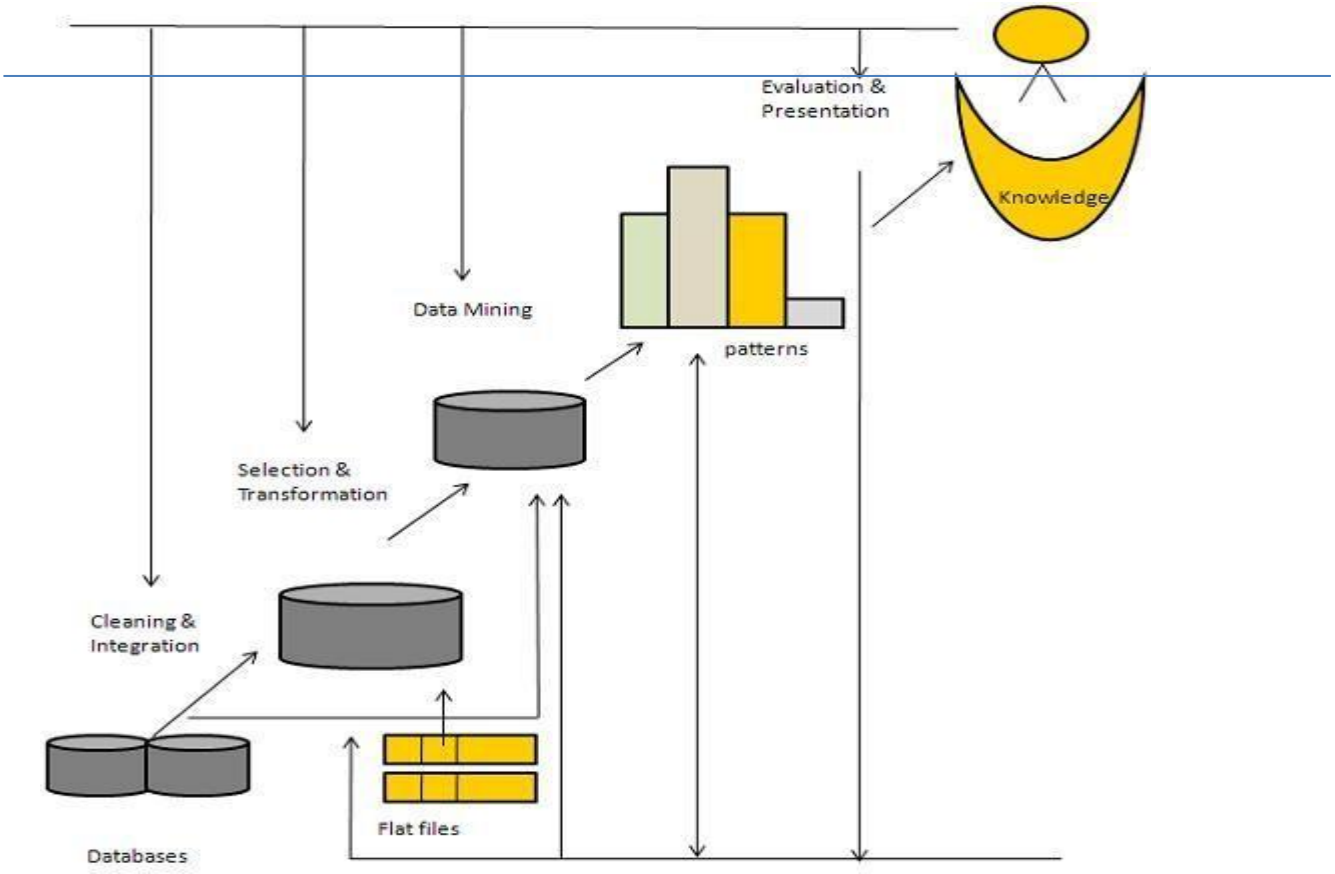
Some people treat data mining same as Knowledge discovery while some



people view data mining essential step in process of knowledge discovery. Here is the list of steps involved inknowledge discovery process:

- **Data Cleaning** - In this step the noise and inconsistent data is removed.
- **Data Integration** - In this step multiple data sources are combined.
- **Data Selection** - In this step relevant to the analysis task are retrieved from the database.
- **Data Transformation** - In this step data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.

- **Data Mining** - In this step intelligent methods are applied in order to extract datapatterns.
- **Pattern Evaluation** - In this step, data patterns are evaluated.
- **Knowledge Presentation** - In this step,knowledge is represented.The following diagram shows the process of knowledge discovery process:



Applications

- Finance
- Telecommunications
- DNA
- Stock Markets
- E-mail

1.5 Data Warehousing

A **Data Warehouse** is separate from DBMS, it stores a huge amount of data, which is typically collected from multiple heterogeneous sources like files, DBMS, etc.

Need for Data Warehouse

An ordinary Database can store MBs to GBs of data and that too for a specific purpose. For storing data of TB size, the storage shifted to Data Warehouse. Besides this, a transactional database doesn't offer itself to analytics.

Advantages

This approach has the following advantages −

- This approach provide high performance.
- The data is copied, processed, integrated, annotated, summarized and restructured in semantic data store in advance.
- Query processing does not require an interface to process data at local sources.

Functions of Data Warehouse Tools and Utilities

The following are the functions of data warehouse tools and utilities −

- **Data Extraction** − Involves gathering data from multiple heterogeneous sources.
- **Data Cleaning** − Involves finding and correcting the errors in data.
- **Data Transformation** − Involves converting the data from legacy format to warehouse format.
- **Data Loading** − Involves sorting, summarizing, consolidating, checking integrity, and building indices and partitions.
- **Refreshing** − Involves updating from data sources to warehouse

Metadata

Metadata is simply defined as data about data. The data that are used to represent other data is known as metadata.

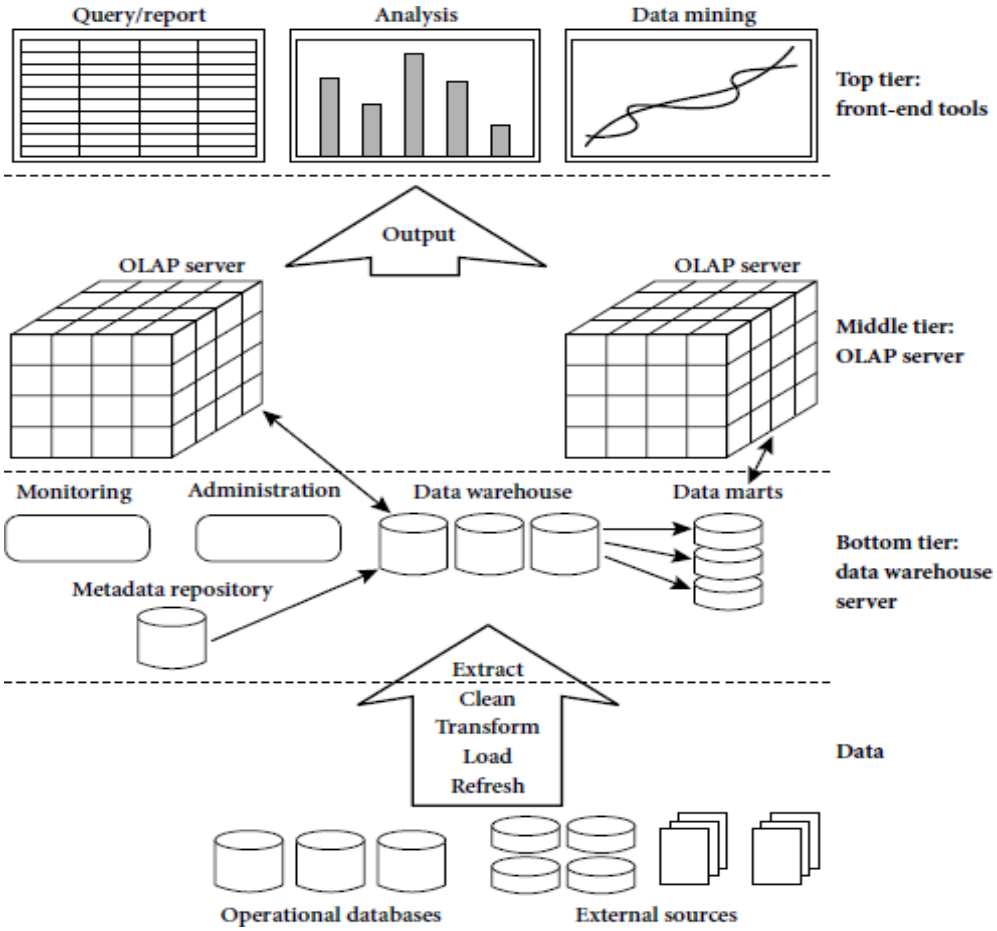
Data Mart

Data marts contain a subset of organization-wide data that is valuable to specific groups of people in an organization

Applications of Data Warehousing

- **Social Media Websites**
- **Banking**
- **Government**

Three Tier Data Warehouse Architecture:



Tier-1:

The bottom tier is a warehouse database server that is almost always a relationaldatabase system. Back-end tools and utilities are used to feed data into the bottomtier from operational databases or other external sources (such as customer profileinformationprovided by external consultants).

Tier-2:

The middle tier is an OLAP server that is typically implemented using either a relational OLAP (ROLAP) model or a multidimensional OLAP.

- OLAP model is an extended relational DBMS thatmaps operations on multidimensionaldata to standard relational operations.
- A multidimensional OLAP (MOLAP) model, that is, a special-purpose server that directly implements multidimensional data and operations.

Tier-3:

The top tier is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

Difference between Data Warehousing & Data Mining

BASIS	DATA WAREHOUSING	DATA MINING
Definition	A huge database which is designed to carry out analytical processes and not transactional application	It is a process of determining hidden relationships an patterns between different sets of data
Meaning	It combines huge sets of related data	It derives useful meaning and insights from a large set of data
Application	Extremely large quantities of data of any organization can be easily stored	It is carried out for the purpose of identifying patterns, relationships and frauds in an organization
Implementation	Before data mining as the data is compiled and stored here in a common database	After warehousing in order to withdraw useful insights
Benefits	Timely data access, enhanced response time and provides consistent data for easy access	Helpful to predict trends, market analysis, financial analysis and recognizing fraudulent
Performed by	Can be performed by engineers	Performed by businessmen with the help of engineers

1.6 Basic Statistical descriptions of Data

Basic Statistical Descriptions of Data

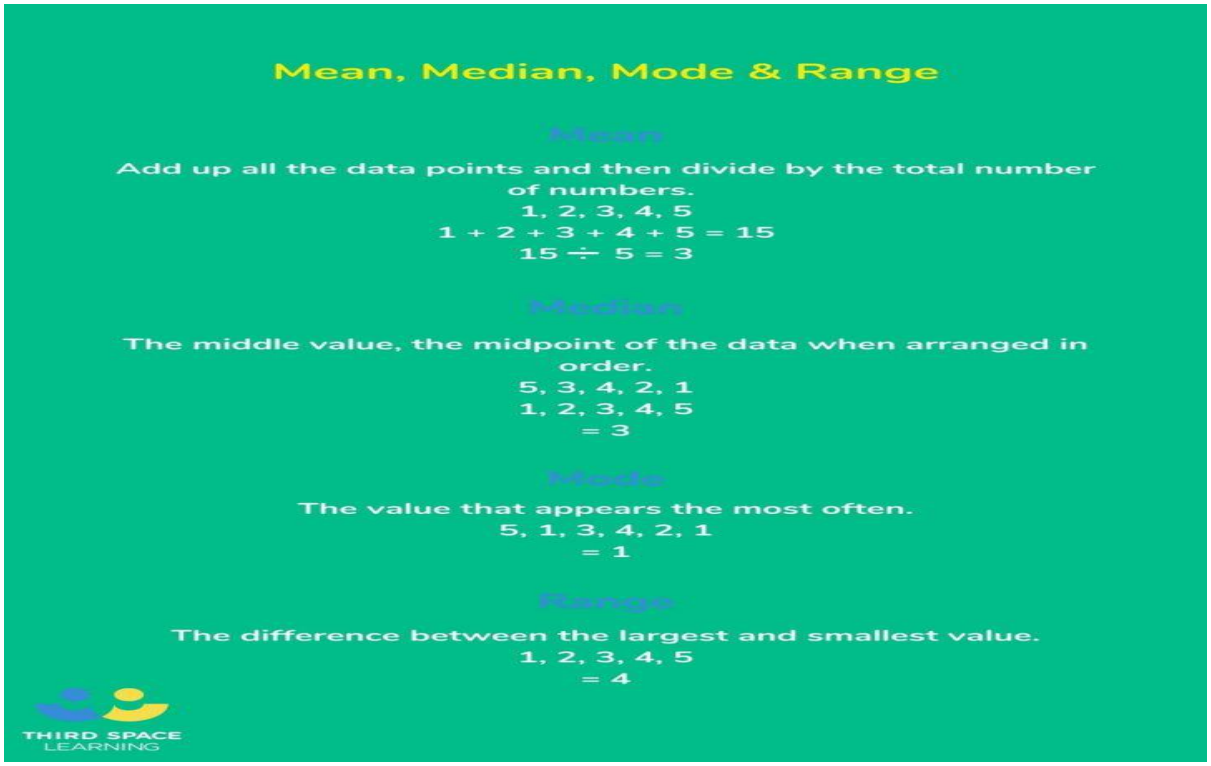
- Motivation
  - To better understand the data: central tendency, variation and spread
- Data dispersion characteristics
  - median, max, min, quantiles, outliers, variance, etc.
- Numerical dimensions correspond to sorted intervals
  - Data dispersion: analyzed with multiple granularities of precision
  - Boxplot or quantile analysis on sorted intervals
- Dispersion analysis on computed measures
  - Folding measures into numerical dimensions
  - Boxplot or quantile analysis on the transformed cube

Measuring the Central Tendency

- Mean (algebraic measure) (sample vs. population): $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \mu = \frac{\sum x}{N}$

  Note: n is sample size and N is population size.
  - Weighted arithmetic mean: $\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$
  - Trimmed mean: chopping extreme values
- Median:
  - Middle value if odd number of values, or average of the middle two values otherwise
  - Estimated by interpolation (for *grouped data*): $median = L_1 + (\frac{n/2 - (\sum freq)l}{freq_{median}})width$
- Mode
  - Value that occurs most frequently in the data
  - Unimodal, bimodal, trimodal
  - Empirical formula: $mean - mode = 3 \times (mean - median)$

age	*frequency*
1–5	200
6–15	450
16–20	300
21–50	1500
51–80	700
81–110	44



Measuring the Dispersion of Data

- Quartiles, outliers and boxplots
  - **Quartiles**: Q_1 (25th percentile), Q_3 (75th percentile)
  - **Inter-quartile range**: IQR = $Q_3 - Q_1$
  - **Five number summary**: min, Q_1, median, Q_3, max
  - **Boxplot**: ends of the box are the quartiles; median is marked; add whiskers, and plot outliers individually
  - **Outlier**: usually, a value higher/lower than 1.5 x IQR
- Variance and standard deviation (*sample: s, population: σ*)
  - **Variance**: (algebraic, scalable computation)

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n-1}[\sum_{i=1}^{n}x_i^2 - \frac{1}{n}(\sum_{i=1}^{n}x_i)^2] \qquad \sigma^2 = \frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2 = \frac{1}{N}\sum_{i=1}^{n}x_i^2 - \mu^2$$

  - **Standard deviation** *s (or σ)* is the square root of variance s^2 *(or* σ^2*)*

Example 1 – Calculation of variance and standard deviation

Let's calculate the variance of the follow data set: 2, 7, 3, 12, 9.

The first step is to calculate the mean. The sum is 33 and there are 5 data points. Therefore, the mean is 33 ÷ 5 = 6.6. Then you take each value in data set, subtract the mean and square the difference. For instance, for the first value:

$(2 - 6.6)^2 = 21.16$

The squared differences for all values are addAed:

21.16 + 0.16 + 12.96 + 29.16 + 5.76 = 69.20

The sum is then divided by the number of data points:

69.20 ÷5 = 13.84

The variance is 13.84. To get the standard deviation, you calculate the square root of the variance, which is 3.72.

UNIT II DESCRIBING DATA

Types of Data – Types of Variables -Describing Data with Tables and Graphs –Describing Data with Averages – Describing Variability – Normal Distributions and Standard (z) Scores

---

2.1 Types of Data

Qualitative Data

1. Nominal
2. Ordinal(Ranked)

Quantitative data

1. Interval
2. Ratio

2.2 Types of Variables

1. Continuous
2. Discrete
3. Independent & Dependent
4. Approximate, Experiment & Observation Study

2.3 Describing Data with Tables and Graphs

Describing Data with Tables

1. Frequency Distribution
2. Grouped & Ungrouped Data
3. Outlier
4. Relative Frequency & Cumulative Frequency & Percentile.

Graphs (Quantitative)

1. Histogram
2. Frequency Polygon
3. Stem & Leaf

Graphs (Qualitative)

Bar graph

2.4 Describing Data with Averages

1. Mean
2. Mode
3. Median

2.5 Describing Variability

1. Range
2. Variance
3. Standard Deviation
4. IQR

2.6 Normal Distributions and Standard (z) Scores

1. Normal Distribution /Curve
2. Z Score

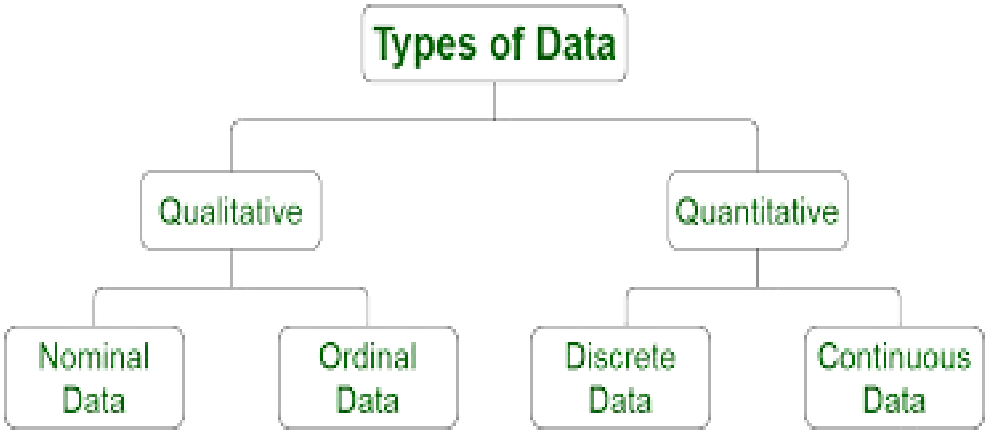
2.1 Types of Data

2.1.1 Data

Any statistical analysis is performed on **data,** A data is a *collection of actual observations or scores in a survey or an experiment.*

There are two types of data: Qualitative and Quantitative data, which are further classified into **four types of data: nominal, ordinal, discrete, and Continuous.**

2.1.2 Types of Data



1.Qualitative Data: Quality: **Categorical variable**

- Measures of 'types' and may be represented by a name, symbol, or a number code.
- Qualitative data are data about categorical variables (e.g. what type).

Ranked Data:

A set of observations where any single observation is a number that indicates relative standing. **Ranked data** consist of numbers (1st, 2nd, . . . 40th place) that represent relative standing within a group

2. Quantitative Data: Quantity: **Numeric variable**

- Measures of values or counts and are expressed as numbers.
- Data about numeric variables (e.g. how many, how much or how often).

Difference between Qualitative Data and Quantitative Data

Qualitative Data	Quantitative Data
1. Qualitative data uses methods like interviews, participant observation, focus on a grouping to gain collective information.	1. Quantitative data uses methods as questionnaires, surveys, and structural observations to gain collective information.
2. Data format used in it is textual. Datasheets are contained of audio or video recordings and notes.	2. Data format used in it is numerical. Datasheets are obtained in the form of numerical values.
3. Qualitative data talks about the experience or quality and explains the questions like 'why' and 'how'.	3. Quantitative data talks about the quantity and explains the questions like 'how much', 'how many .
4. The data is analyzed by grouping it into different categories.	4. The data is analyzed by statistical methods.
5.Qualitative data are subjective and can be further open for interpretation.	5. Quantitative data are fixed and universal.

2.1.3 Levels of Measurement

Specifies the extent to which a number (or word or letter) actually represents some attribute and, therefore, has implications for the appropriateness of various arithmetic operations and statistical procedures.

1.Qualitative Data and Nominal Measurement

Nominal Measurement :Words, letters, or numerical codes of qualitative data that reflect differences in kind based on classification.

2.Ranked Data and Ordinal Measurement

Ordinal Measurement : Relative standing of ranked data that reflects differences in degree based on order

3.Quantitative Data and Interval/Ratio Measurement

Interval/Ratio Measurement : Amounts or counts of quantitative data reflect differences in degree based on equal intervals and a true zero.

Example

1.Indicate whether each of the following terms is qualitative (because it's a word, letter, or numerical code representing a class or category); ranked (because it's a number representing relative standing); or quantitative (because it's a number representing an amount or a count).

(a) ethnic group (b) age (c) family size (d) academic major (e) sexual preference (f) IQ score (g) net worth (dollars) (h) third-place finish (i) gender (j) temperature

Answer

(a) qualitative **(e)** qualitative **(i)** qualitative

(b) quantitative **(f)** quantitative **(j)** quantitative

(c) quantitative **(g)** quantitative

(d) qualitative **(h)** ranked

2. Indicate the level of measurement—nominal, ordinal, or interval/ ratio—attained by the following sets of observations or data. When appropriate, indicate that measurement is only approximately interval.

(a) height (b) religious affiliation (c) score for psychopathic tendency (d) years of education (e) military rank (f) vocational goal (g) GPA (h) marital status

Answer

(a)interval/ratio (e)ordinal

(b)nominal (f) nominal

(c)approximately interval (g)approximately interval

(d)interval/ratio (h) nominal

2.2 TYPES OF VARIABLES

Variable

A characteristic or property that can take on different values

Constant

A characteristic or property that can take on only one value.

Discrete Variable

A variable that consists of isolated numbers separated by gaps

Continuous Variable

A variable that consists of numbers whose values, at least in theory, have no restrictions.

Approximate Numbers

Numbers that are rounded off, as is always the case with values for continuous variables

Example

Indicate whether the following quantitative observations are discrete or continuous.

(a) litter of mice (b) cooking time for pasta (c) parole violations by convicted felons (d) IQ (e) age (f) population of your hometown (g) speed of a jetliner

Answer

(a) discrete (d) continuous (g) continuous

(b) continuous (e) continuous

(c) discrete (f) discrete

Independent and Dependent Variables

Experiment

A study in which the investigator decides who receives the special treatment

Independent Variable

The treatment manipulated by the investigator in an experiment.

Dependent Variable

A variable that is believed to have been influenced by the independent variable

Observational Study

A study that focuses on detecting relationships between variables not manipulated by the investigator

Confounding variable

An uncontrolled variable that compromises the interpretation of a study

Example

For each of the listed studies, indicate whether it is an experiment or an observational study. If it is an experiment, identify the independent variable and note any possible confounding variables.

(a)years of education and annual income (b) prescribed hours of sleep deprivation and subsequent amount of REM (dream) sleep (c) weight loss among obese males who choose to participate either in a weight-loss program or a self-esteem enhancement program (d) estimated study hours and subsequent test score. (e)recidivism among substance abusers assigned randomly to different rehabilitation pro- grams (f) subsequent GPAs of college applicants who, as the result of a housing lottery, live eitheron campus or off campus

Answer

(a) observational study

(b) experiment (independent variable: prescribed hours of sleep deprivation)

(c) experiment (independent variable: two programs; possible confounding variable:self-selection of program)

(d) observational study

(e) experiment (independent variable: different rehabilitation programs)

(f) experiment (independent variable: on campus or off campus)

2.3 Describing Data with Tables and Graphs

2.3.1 Describing Data with Tables

2.3.1.1. Frequency Distributions for Quantitative Data

- **Frequency Distribution**:
  - A collection of observations produced by sorting observations into classes and showing their frequency (f) of occurrence in each class.

1. Frequency Distribution for Ungrouped Data

- A frequency distribution produced whenever observations are sorted into classes of single values.

Example :

Students in a theater arts appreciation class rated the classic film The Wizard of Oz on a 10-point scale, ranging from 1 (poor) to 10 (excellent), as follows:

3	7	2	7	8
3	1	4	10	3
2	5	3	5	8
9	7	6	3	7
8	9	7	3	6

Solution

Data	Frequency
10	1
9	2
8	3
7	5
6	2
5	2
4	1
3	6
2	2
1	1
Total	25

2. Frequency Distribution for Grouped Data

- A frequency distribution produced whenever observations are sorted into classes of more than one value
- Unit of Measurement The smallest possible difference between scores.
- Real Limits Located at the midpoint of the gap between adjacent tabled boundaries.

Example:

The IQ scores for a group of 35 high school dropouts are as follows:

91	85	84	79	80
87	96	75	86	104
95	71	105	90	77
123	80	100	93	108
98	69	99	95	90
110	109	94	100	103
112	90	90	98	89

(a) Construct a frequency distribution for grouped data.

(b) Specify the real limits for the lowest class interval in this frequency distribution.

Solution

(a) Calculating the class width,

$$\frac{123 - 69}{10} \quad \frac{54}{10} \quad 5.4$$

IQ	*f*
120–124	1
115–119	0
110–114	2
105–109	3
100–104	4
95–99	6
90–94	7
85–89	4
80–84	3
75–79	3
70–74	1
65–69	1
Total	**35**

(a) 64.5–69.5

Example 2 :What are some possible poor features of the following frequency distribution?

ESTIMATED WEEKLY TV VIEWING TIME (HRS) FOR 250 SIXTH GRADERS

VIEWING TIME	*f*
35–above	2
30–34	5
25–30	29
20–22	60
15–19	60
10–14	34
5–9	31
0–4	29
Total	250

Answer

Not all observations can be assigned to one and only one class (because of gap between 20–22 and 25–30 and overlap between 25–30 and 30–34). All classes are not equal in width (25–30 versus 30–34). All classes do not have both boundaries (35–above).

2.3.1.2. Relative Frequency Distributions and Cumulative Frequency

- **Relative Frequency Distribution**

A frequency distribution showing the frequency of each class as a fraction of the total frequency for the entire distribution.

$$\text{Relative Frequency} = \frac{\text{Frequency}}{\text{Total no.of Frequency}} * 100$$

Marks	Frequency	Relative Frequency
45 – 50	3	3 / 40 x 100 = 0.075
50 – 55	1	1 / 40 x 100 = 0.025
55 – 60	1	1 / 40 x 100 = 0.075
60 -65	6	6 / 40 x 100 = 0.15
65 – 70	8	8 / 40 x 100 = 0.2
70 – 80	3	3 / 40 x 100 = 0.275
80 -90	11	11 / 40 x 100 = 0.075
90 – 100	7	1 / 40 x 100 = 0.025
Total	40	

- **Cumulative Frequency**

Distribution A frequency distribution showing the total number of observations in each class and all lower-ranked classes.

$$\text{Cumulative Percent} = \frac{\text{Cumulative Frequency}}{\text{Total no.of Frequency}} * 100$$

WEIGHT	*f*	CUMULATIVE *f*	CUMULATIVE PERCENT
240–249	1	53	100
230–239	0	52	98
220–229	3	52	98
210–219	0	49	92
200–209	2	49	92
190–199	4	47	89
180–189	3	43	81
170–179	7	40	75
160–169	12	33	62
150–159	17	21	40
140–149	1	4	8
130–139	3	3	6
Total	53		

Sample Example

Movie ratings reflect ordinal measurement because they can be ordered from most to least restrictive: NC-17, R, PG-13, PG, and G. The ratings of some films shown recently in San Francisco are as follows:

PG	PG	PG	PG-13	G
G	PG-13	R	PG	PG
R	PG	R	PG	R
NC-17	NC-17	PG	G	PG-13

(a) Construct a frequency distribution.

(b) Convert to relative frequencies, expressed as percentages.

(c) Construct a cumulative frequency distribution.

(d) Find the approximate percentile rank for those films with a PG rating.

Solution

MOVIE RATINGS	(a) *f*	(b) RELATIVE *f*(%)	(c) CUMULATIVE *f*
NC-17	2	10	20
R	4	20	18
PG-13	3	15	14
PG	8	40	11
G	3	15	3
Totals	20	100%	

(d) Percentile rank for films with a PG rating is 55 (from $\frac{11}{20}$ multiplied by 100).

Outlier

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population

For example in the scores 25,29,3,32,85,33,27,28 both 3 and 85 are "outliers".

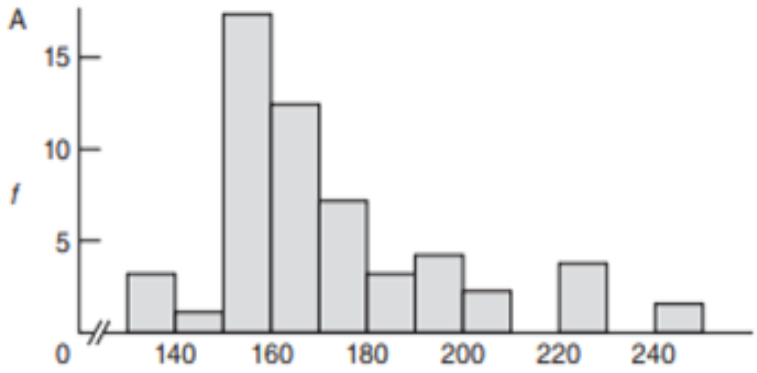
2.3.2 GRAPHS

Data can be described clearly and concisely with the aid of a well-constructed frequency distribution. And data can often be described even more vividly, particularly when you're attempting to communicate with a general audience, by converting frequency distributions into graphs.

GRAPHS FOR QUANTITATIVE DATA

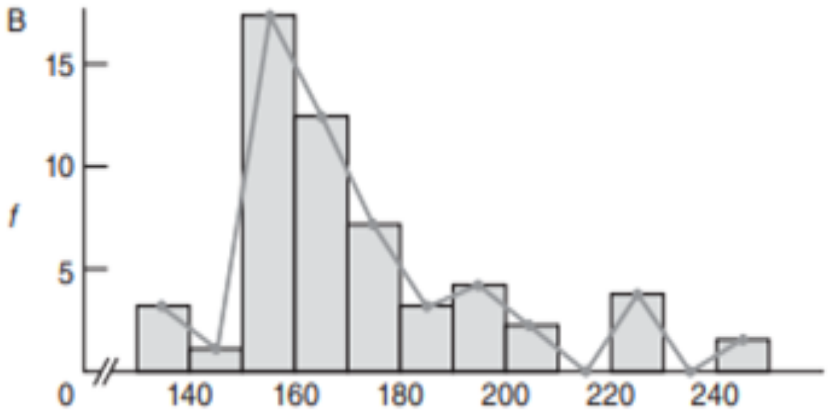
A.Histogram

A bar-type graph for quantitative data. The common boundaries between adjacent bars emphasize the continuity of the data, as with continuous variables.



B.Frequency Polygon

A line graph for quantitative data that also emphasizes the continuity of continuous variables



Stem and Leaf

Display A device for sorting quantitative data on the basis of leading and trailing digits

Example:

Construct a stem and leaf display for the following IQ scores obtained from a group of four-year-old children.

120	98	118	117	99	111
126	85	88	124	104	113
108	141	123	137	78	96
102	132	109	106	143»	

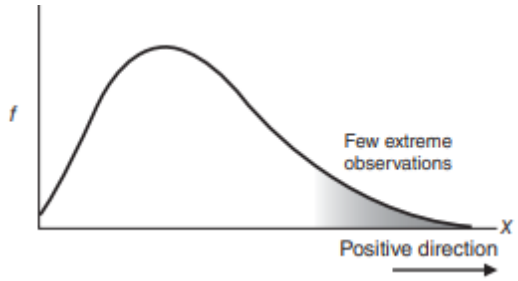
Solution

7	8				
8	5	8			
9	8	9	6		
10	8	2	9	6	4
11	8	7	1	3	
12	0	6	3	4	
13	2	7			
14	1	3			

T Y P I C A L S H A P E S

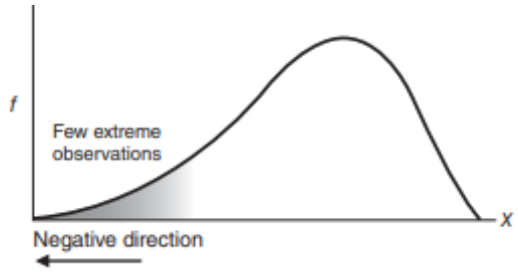
- Positively Skewed

A distribution that includes a few extreme observations in the positive direction (to the right of the majority of observations).



- Negatively Skewed

A distribution that includes a few extreme observations in the negative direction (to the left of the majority of observations).

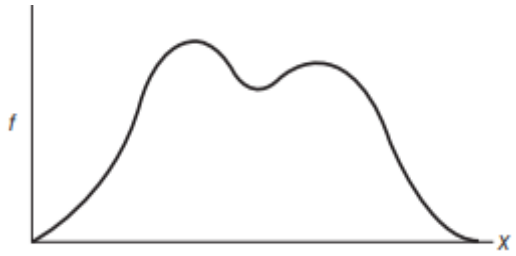


- **Normal**

The familiar bell-shaped silhouette of the normal curve can be superimposed on many frequency distributions, including those for uninterrupted gestation periods of human fetuses, scores on standardized tests, and even the popping times of individual kernels in a batch of popcorn.

- **Bimodal**

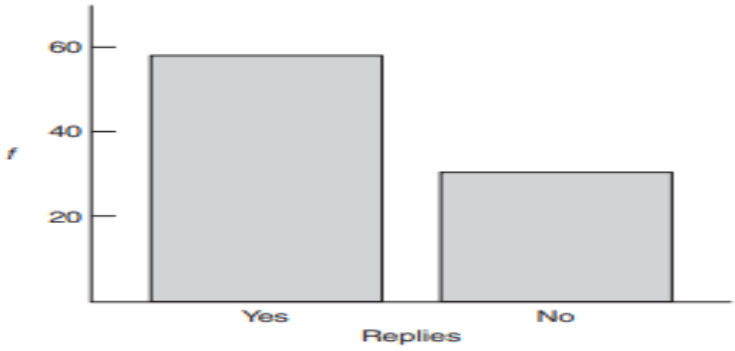
The coexistence of two different types of observations in the same distribution. For instance, the distribution of the ages of residents in a neighborhood consisting largely of either new parents or their infants has a bimodal shape.



A GRAPH FOR QUALITATIVE (NOMINAL) DATA

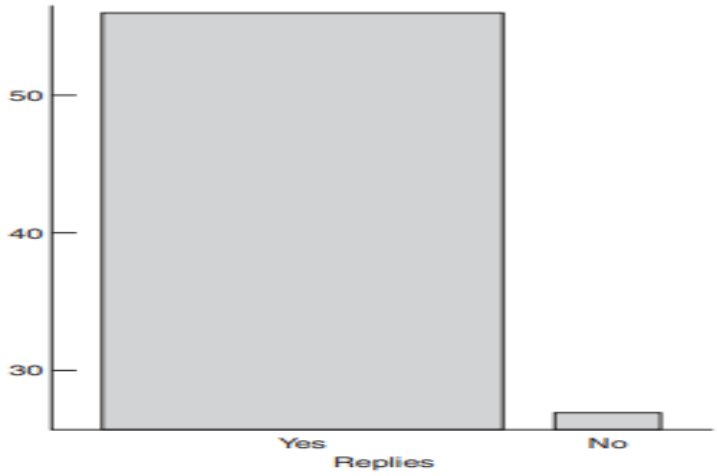
- Bar Graph

A bar-type graph for qualitative data. Gaps between adjacent bars emphasize the discontinuous nature of the data.



M I S L E A D I N G G R A P H S

Graphs can be constructed in an unscrupulous manner to support a particular point of view. Graphs may be misleading by being excessively complex or poorly constructed. Even when constructed to display the characteristics of their data accurately, graphs can be subject to different interpretations, or unintended kinds of data can seemingly and ultimately erroneously be derived



Questions

The number of friends reported by Face book users is summarized in the following frequency distribution:

FRIENDS	*f*
400 – above	2
350 – 399	5
300 – 349	12
250 – 299	17
200 – 249	23
150 – 199	49
100 – 149	27
50 – 99	29
0 – 49	36
Total	200

(a) What is the shape of this distribution?

(b) Find the relative frequencies.

(c) Find the approximate percentile rank of the interval 300–349.

(d) Convert to a histogram.

(e) Why would it not be possible to convert to a stem and leaf display?

2.4 Describing Data with Averages

3.4.1 MODE

3.4.2 MEDIAN

3.4.3 MEAN

Measures of Central Tendency

Numbers or words that attempt to describe, most generally, the middle or typical value for a distribution.

2.4.1 Mode

The value of the most frequent score.

Types of Mode

The different types of Mode are Unimodal, Bimodal, Trimodal, and Multimodal. Let us understand each of these Modes.

1.Unimodal Mode –

A set of data with one Mode is known as a Unimodal Mode.

For example, the Mode of data set A = { 14, 15, 16, 17, 15, 18, 15, 19} is 15 as there is only one value repeating itself. Hence, it is a Unimodal data set.

2.Bimodal Mode

A set of data with two Modes is known as a Bimodal Mode. This means that there are two data values that are having the highest frequencies.

For example, the Mode of data set A = { 8,13,13,14,15,17,17,19} is 13 and 17 because both 13 and 17 are repeating twice in the given set. Hence, it is a Bimodal data set.

3.Trimodal Mode

A set of data with three Modes is known as a Trimodal Mode. This means that there are three data values that are having the highest frequencies.

For example, the Mode of data set A = {2, 2, 2, 3, 4, 4, 5, 6, 5,4, 7, 5, 8} is 2, 4, and 5 because all the three values are repeating thrice in the given set. Hence, it is a Trimodal data set.

4.Multimodal Mode - A set of data with four or more than four Modes is known as a Multimodal Mode.

2.4.2 Median

The middle value when observations are ordered from least to most.

B. EXAMPLES

Set of five scores:

2, 8, 2, 7, 6

1 2, 2, 6, 7, 8

2 $\frac{5+1}{2} = 3$

2, 2, 6, 7, 8

3 1, 2, 3

4 median = 6

Set of six scores:

3, 8, 9, 3, 1, 8

1 1, 3, 3, 8, 8, 9

2 $\frac{6+1}{2} = 3.5$

1, 3, 3, 8, 8, 9

5 1, 2, 3, 4

6 median $= \frac{3+8}{2} = 5.5$

Sample Question

1. Find the median for the following retirement ages: 60, 63, 45, 63, 65, 70, 55, 63, 60, 65, 63.

Answer median = 63

2. Find the median for the following gas mileage tests: 26.3, 28.7, 27.4, 26.6, 27.4, 26.9.

Answer

median = 27.15 (halfway between 26.9 and 27.4)

2.4.3 Mean

The mean is found by adding all scores and then dividing by the number of scores.

$$Mean = \frac{sum\ of\ all\ scores}{number\ of\ scores}$$

Example 1: Find the mean of the data set: 32, 41, 28, 54, 35, 26, 23, 33, 38, 40.

Solution: To find the mean, we have to add all the given values first.

Sum of observations = 32 + 41 + 28 + 54 + 35 + 26 + 23 + 33 + 38 + 40 = 350

Total number of observations = 10

Therefore, the mean of observations is:

Mean = (Sum of all observations)/(Total number of observations)

Mean = 350/10 = 35

Hence, 35 is the required arithmetic mean.

Population : A complete set of scores.

Sample : A subset of scores.

Sample Mean (X) The balance point for a sample, found by dividing the sum for the values of all scores in the sample by the number of scores in the sample.

SAMPLE MEAN

$$\overline{X} = \frac{\sum X}{n}$$

Sample Mean (X) The balance point for a sample, found by dividing the sum for the values of all scores in the sample by the number of scores in the sample.

Sample Size (n) The total number of scores in the sample.

Population Mean (μ) The balance point for a population, found by dividing the sum for all scores in the population by the number of scores in the population.

Population Size (N) The total number of scores in the population.

POPULATION MEAN

$$\mu = \frac{\sum X}{N}$$

2.5 Describing Variability

Measures of Variability

Descriptions of the amount by which scores are dispersed or scattered in a distribution.

2.5.1 Range

The difference between the largest and smallest scores.

Range = Highest Value – Lowest Value

Example 1: Find the range of given observations: 32, 41, 28, 54, 35, 26, 23, 33, 38, 40.

Solution: Let us first arrange the given values in ascending order.

23, 26, 28, 32, 33, 35, 38, 40, 41, 54

Since 23 is the lowest value and 54 is the highest value, therefore, the range of the observations will be;

Range (X) = Max (X) – Min (X)

= 54 – 23

= 31

2.5.2 Variance

The mean of all squared deviation scores.

To calculate the variance follows these steps:

- Work out the Mean (the simple average of the numbers)
- Then for each number: subtract the Mean and square the result (the *squared difference*).
- Then work out the average of those squared differences.

Example: Find the variance of the numbers 3, 8, 6, 10, 12, 9, 11, 10, 12, 7.

Solution:

Given, 3, 8, 6, 10, 12, 9, 11, 10, 12, 7

Step 1: Compute the mean of the 10 values given.

Mean = (3+8+6+10+12+9+11+10+12+7) / 10 = 88 / 10 = 8.8

Step 2: Make a table with three columns, one for the X values, the second for the deviations and the third for squared deviations. As the data is not given as sample data so we use the formula for population variance. Thus, the mean is denoted by μ.

Value X	X – μ	$(X - \mu)^2$
3	-5.8	33.64
8	-0.8	0.64
6	-2.8	7.84
10	1.2	1.44
12	3.2	10.24
9	0.2	0.04
11	2.2	4.84
10	1.2	1.44
12	3.2	10.24
7	-1.8	3.24
Total	0	73.6

Step 3:

$$\sigma^2 = \frac{\sum(X-\mu)^2}{N}$$

$= 73.6 / 10$

$= 7.36$

2.5.3 Standard Deviation

A rough measure of the average (or standard) amount by which scores deviate on either side of their mean.

$$standard\ deviation = \sqrt{variance}$$

Sum of Squares (SS)

The sum of squared deviation scores.

SUM OF SQUARES (*SS*) FOR POPULATION (DEFINITION FORMULA)

$$SS = \sum(X - \mu)^2$$

SUM OF SQUARES (*SS*) FOR POPULATION (COMPUTATION FORMULA)

$$SS = \sum X^2 - \frac{(\sum X)^2}{N}$$

CALCULATION OF POPULATION STANDARD DEVIATION σ (DEFINITION FORMULA)

A. COMPUTATION SEQUENCE

Assign a value to *N* **1** representing the number of *X* scores
Sum all *X* scores **2**
Obtain the mean of these scores **3**
Subtract the mean from each *X* score to obtain a deviation score **4**
Square each deviation score **5**
Sum all squared deviation scores to obtain the sum of squares **6**
Substitute numbers into the formula to obtain population variance, σ^2 **7**
Take the square root of σ^2 to obtain the population standard deviation, σ **8**

B. DATA AND COMPUTATIONS

X	**4** $X - \mu$	**5** $(X - \mu)^2$
13	3	9
10	0	0
11	1	1
7	−3	9
9	−1	1
11	1	1
9	−1	1

1 $N = 7$ **2** $\Sigma X = 70$ **6** $SS = \Sigma (X - \mu)^2 = 22$

3 $\mu = \frac{70}{7} = 10$

7 $\sigma^2 = \frac{SS}{N} = \frac{22}{7} = 3.14$ **8** $\sigma = \sqrt{\frac{SS}{N}} = \sqrt{\frac{22}{7}} = \sqrt{3.14} = 1.77$

CALCULATION OF POPULATION STANDARD DEVIATION (σ) (COMPUTATION FORMULA)

A. COMPUTATIONAL SEQUENCE

Assign a value to *N* representing the number of *X* scores **1**
Sum all *X* scores **2**
Square the sum of all *X* scores **3**
Square each *X* score **4**
Sum all squared *X* scores **5**
Substitute numbers into the formula to obtain the sum of squares, *SS* **6**
Substitute numbers into the formula to obtain the population variance, σ^2 **7**
Take the square root of σ^2 to obtain the population standard deviation, σ **8**

B. DATA AND COMPUTATIONS

X	**4** X^2
13	169
10	100
11	121
7	49
9	81
11	121
9	81

1 $N = 7$ **2** $\sum X = 70$ **5** $\sum X^2 = 722$

3 $(\sum X)^2 = 4900$

6 $SS = \sum X^2 - \frac{(\sum X)^2}{N} = 722 - \frac{4900}{7} = 722 - 700 = 22$

7 $\sigma^2 = \frac{SS}{N} = \frac{22}{7} = 3.14$ **8** $\sigma = \sqrt{\frac{SS}{N}} = \sqrt{\frac{22}{7}} = \sqrt{3.14} = 1.77$

Standard Deviation for Population σ

Population Standard Deviation (σ) A rough measure of the average amount by which scores in the population deviate on either side of their population mean.

VARIANCE FOR POPULATION

$$\sigma^2 = \frac{SS}{N}$$

STANDARD DEVIATION FOR POPULATION

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{SS}{N}}$$

Sum of Squares Formulas for Sample

SUM OF SQUARES (*SS*) FOR SAMPLE (DEFINITION FORMULA)

$$SS = \Sigma(X - \bar{X})^2$$

(COMPUTATION FORMULA)

$$SS = \Sigma X^2 - \frac{(\Sigma X)^2}{n}$$

Sample Standard Deviation (s)

A rough measure of the average amount by which scores in the sample deviate on either side of their sample mean.

Reminder: Replace n with n − 1 only when dividing SS to obtain s2 and s.

VARIANCE FOR SAMPLE

$$s^2 = \frac{SS}{n-1}$$

STANDARD DEVIATION FOR SAMPLE

$$s = \sqrt{s^2} = \sqrt{\frac{SS}{n-1}}$$

CALCULATION OF SAMPLE STANDARD DEVIATION (*S*) (COMPUTATION FORMULA)

A. COMPUTATIONAL SEQUENCE

Assign a value to n representing the number of X scores **1**
Sum all X scores **2**
Square the sum of all X scores **3**
Square each X score **4**
Sum all squared X scores **5**
Substitute numbers into the formula to obtain the sum of squares, SS **6**
Substitute numbers into the formula to obtain the sample variance, s^2 **7**
Take the square root of s^2 to obtain the sample standard deviation, s **8**

B. DATA AND COMPUTATIONS

X	**4** X^2
7	49
3	9
1	1
0	0
4	16

1 $n = 5$ **2** $\Sigma X = 15$ **5** $\Sigma X^2 = 75$
3 $(\Sigma X)^2 = 225$

6 $$SS = \Sigma X^2 - \frac{(\Sigma X)^2}{n} = 75 - \frac{225}{5} = 75 - 45 = 30$$

7 $$s^2 = \frac{SS}{n-1} = \frac{30}{4} = 7.50$$ **8** $$s = \sqrt{\frac{SS}{n-1}} = \sqrt{\frac{30}{4}} = \sqrt{7.50} = 2.74$$

CALCULATION OF SAMPLE STANDARD DEVIATION (*S*) (DEFINITION FORMULA)

A. COMPUTATION SEQUENCE

Assign a value to n **1** representing the number of X scores
Sum all X scores **2**
Obtain the mean of these scores **3**
Subtract the mean from each X score to obtain a deviation score **4**
Square each deviation score **5**
Sum all squared deviation scores to obtain the sum of squares **6**
Substitute numbers into the formula to obtain the sample variance, s^2 **7**
Take the square root of s^2 to obtain the sample standard deviation, s **8**

B. DATA AND COMPUTATIONS

X	**4** $X - \bar{X}$	**5** $(X - \bar{X})^2$
7	4	16
3	0	0
1	−2	4
0	−3	9
4	1	1

1 $n = 5$ **2** $\Sigma X = 15$ **6** $SS = \Sigma(X - \bar{X})^2 = 30$

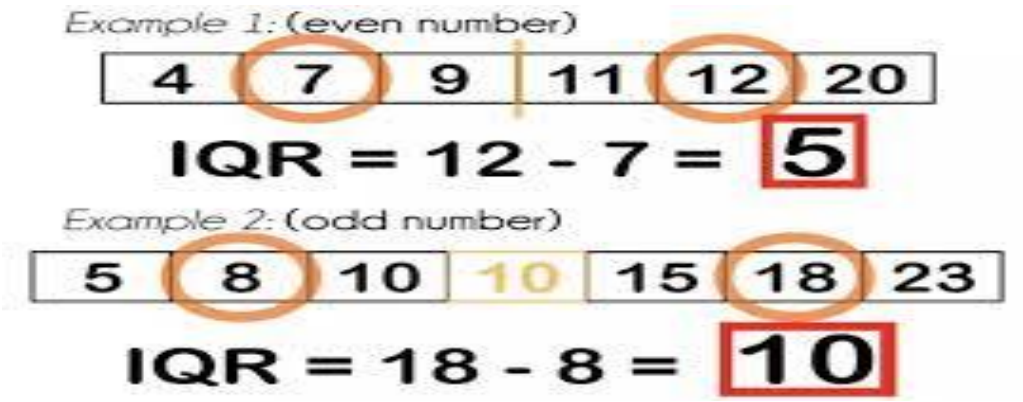
3 $$\bar{X} = \frac{15}{5} = 3$$

7 $$s^2 = \frac{SS}{n-1} = \frac{30}{4} = 7.50$$ **8** $$s = \sqrt{\frac{SS}{n-1}} = \sqrt{\frac{30}{4}} = \sqrt{7.50} = 2.74$$

INTERQUARTILE RANGE (IQR)

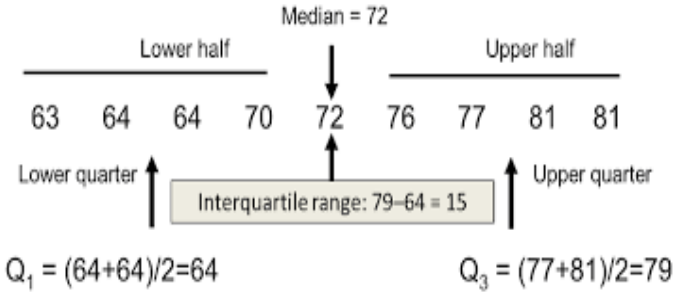
The interquartile range (IQR) is the range of values that resides in the middle of the scores. When a distribution is skewed, and the median is used instead of the mean to show a central tendency, the appropriate measure of variability is the Interquartile range.

Method 1:

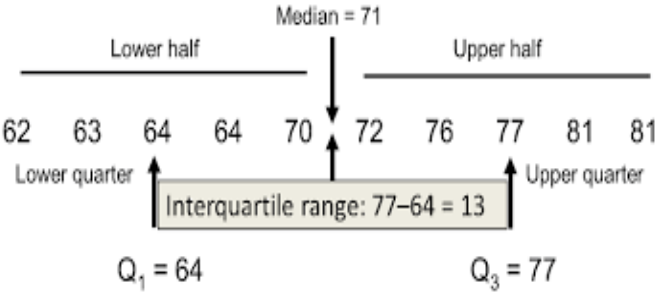


Method 2:

ODD Number



Even Number



$$IQR = Q_3 - Q_1$$

IQR=79-64=15

IQR=77-64=13

Questions

Determine the values of the range and the IQR for the following sets of data.

(a) Retirement ages: 60, 63, 45, 63, 65, 70, 55, 63, 60, 65, 63

(b) Residence changes: 1, 3, 4, 1, 0, 2, 5, 8, 0, 2, 3, 4, 7, 11, 0, 2, 3, 4

Answer

(a) .range = 25; $IQR = 65 - 60 = 5$

(b) range = 11; $IQR = 4 - 1 = 3$

3.6 Normal Distributions and Standard (z) Scores

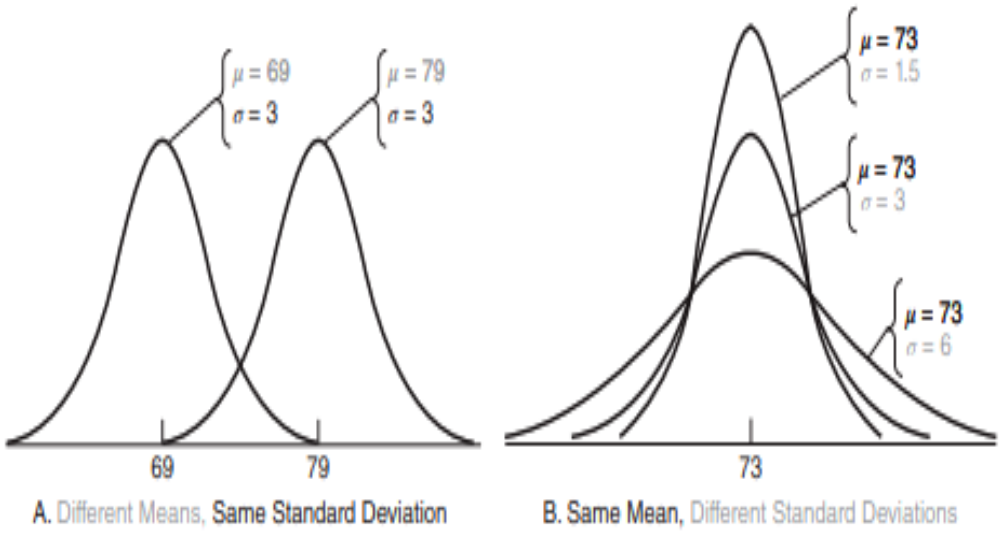
3.6.1 THE NORMAL CURVE

3.6.2 z SCORES

3.6.1 Normal Curve

A theoretical curve noted for its symmetrical bell-shaped form.

Different Normal Curve



A. Different Means, Same Standard Deviation B. Same Mean, Different Standard Deviations

STANDARD NORMAL CURVE

The tabled normal curve for z scores, with a mean of 0 and a standard deviation of 1.

Sample Questions

Express each of the following scores as a z score:

(a) Margaret's IQ of 135, given a mean of 100 and a standard deviation of 15

(b) a score of 470 on the SAT math test, given a mean of 500 and a standard deviation of 100

(c) a daily production of 2100 loaves of bread by a bakery, given a mean of 2180 and a standard deviation of 5

(d) Sam's height of 69 inches, given a mean of 69 and a standard deviation of 3

(e) a thermometer-reading error of –3 degrees, given a mean of 0 degrees and a standard deviation of 2 degrees

Answer:

(a) 2.33 **(d)** 0.00

(b) –0.30 **(e)** –1.50

(c) –1.60

1. Sketch the normal curve and shade in the target area.

Examples: One Area Two Areas

2. Plan the solution in terms of the normal table.

3. Convert *X* to *z*: $z = \frac{X - \mu}{\sigma}$

4. Find the target area by entering either column A or A' with *z*, and noting the corresponding proportion from column B, C, B', or C'.

----------FINDING SCORES----------

1. Sketch the normal curve and, on the correct side of the mean, draw a line representing the target score.

Examples: To Left of Mean (area to left less than .5000) To Right of Mean (area to left more than .5000)

2. Plan the solution in terms of the normal table.

3. Find *z* by locating the entry nearest to that desired in column B, C, B', or C' and reading out the corresponding *z* score.

4. Convert *z* to the target score: $X = \mu + (z)(\sigma)$

3.6.2 z Score

A unit-free, standardized score that indicates how many standard deviations a score is above or below the mean of its distribution.

z SCORE

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

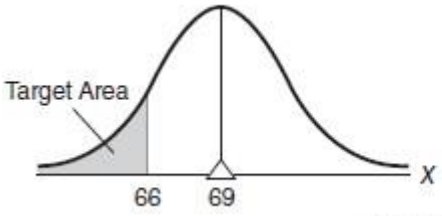
Finding Proportions

Finding Proportions for One Score

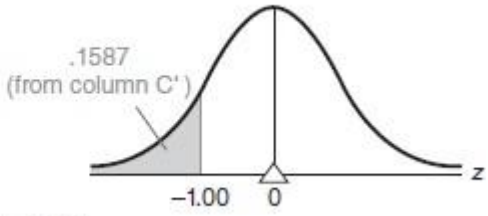
- Sketch a normal curve and shade in the target area,
- Plan your solution according to the normal table.
- Convert *X* to *z*.

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

Find: Proportion Below 66



Solution:



Answer: .1587

Finding Proportions between Two Scores

- Sketch a normal curve and shade in the target area, (example, find proportion between 245to 255)
- Plan your solution according to the normal table.
- Convert *X* to *z* by expressing 255 as

$$z = \frac{255 - 270}{15} = \frac{-15}{15} = -1.00$$

and by expressing 245 as

$$z = \frac{245 - 270}{15} = \frac{-25}{15} = -1.67$$

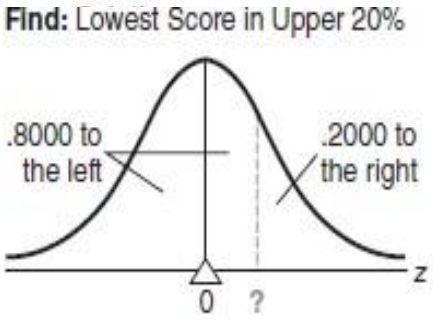
$$z = \frac{255 - 270}{15} = \frac{-15}{15} = -1.00$$

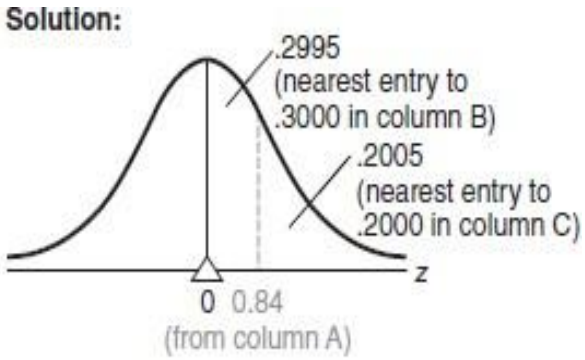
and by expressing 245 as

$$z = \frac{245 - 270}{15} = \frac{-25}{15} = -1.67$$

Finding One Score

- Sketch a normal curve and, on the correct side of the mean, draw a line representing the targetscore, as in figure

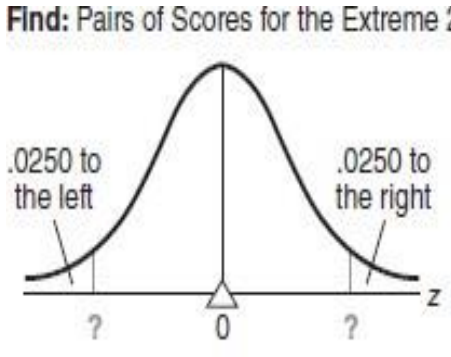


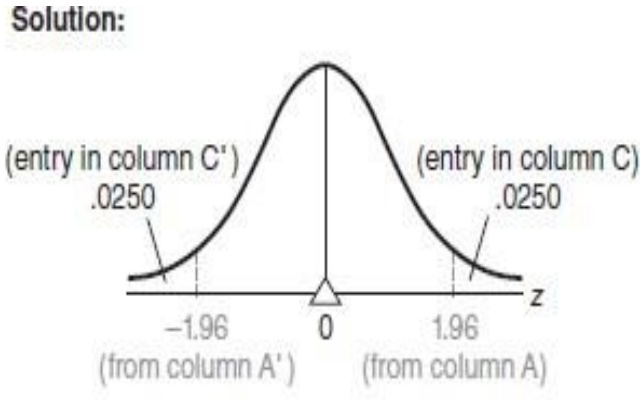


Answer: $X = \mu + (z)(\sigma)$
$= 230 + (0.84)(50)$
$= 230 + 42$
$= 272$

Finding Two Scores

- Sketch a normal curve. On either side of the mean, draw two lines representing the two target scores,as in figure





Answer: $X_{min} = \mu + (z)(\sigma)$
$= 22 + (-1.96)(4)$
$= 22 - 7.84$
$= 14.16$

Answer: $X_{max} = \mu + (z)(\sigma)$
$= 22 + (1.96)(4)$
$= 22 + 7.84$
$= 29.84$

Example

To answer the question about eligible FBI applicants, replace X with 66 (the maximum permissible height), μ with 69 (the mean height), and σ with 3 (the standard deviation of heights) and solve for z as follows:

$$\frac{66-69}{3}=\frac{-3}{3}=-1$$

Question1:

You take the GATE examination and score 500. The mean score for the GATE is 390 and the standard deviation is 45. How well did you score on the test compared to the average test taker?

Solution:

The following data is readily available in the above question statement

Raw score/observed value = X = 500

Mean score = μ = 390

Standard deviation = σ = 45

By applying the formula of z-score,

$z = (X - \mu) / \sigma$

$z = (500 - 390) / 45$

$z = 110 / 45 = 2.44$

This means that your z-score is **2.44**.

Follow the instruction below to find the probability from the table.

Here, **z-score = 2.44**

1. Firstly, map the first two digits 2.4 on the Y-axis.
2. Then along the X-axis, map 0.04
3. Join both axes.

Z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07
1.7	0.95543	0.95637	0.95728	0.95818	0.95907	0.95994	0.9608	0.96164
1.8	0.96407	0.96485	0.96562	0.96638	0.96712	0.96784	0.96856	0.96926
1.9	0.97128	0.97193	0.97257	0.9732	0.97381	0.97441	0.975	0.97558
2	0.97725	0.97778	0.97831	0.97882	0.97932	0.97982	0.9803	0.98077
2.1	0.98214	0.98257	0.983	0.98341	0.98382	0.98422	0.98461	0.985
2.2	0.9861	0.98645	0.98679	0.98713	0.98745	0.98778	0.98809	0.9884
2.3	0.98928	0.98956	0.98983	0.9901	0.99036	0.99061	0.99086	0.99111
2.4	0.9918	0.99202	0.99224	0.99245	0.99266	0.99286	0.99305	0.99324
2.5	0.99379	0.99396	0.99413	0.9943	0.99446	0.99461	0.99477	0.99492
2.6	0.99534	0.99547	0.9956	0.99573	0.99585	0.99598	0.99609	0.99621
2.7	0.99653	0.99664	0.99674	0.99683	0.99693	0.99702	0.99711	0.9972

Question 2:

What is the probability that a student scores between 350 and 400 (with a mean score μ of 390 and a standard deviation σ of 45)?

Solution:

Min score = X1 = 350

Max score = X2 = 400

By applying the formula of z-score,

$z1 = (X1 - \mu) / \sigma$

$z1 = (350 - 390) / 45$

$z1 = -40 / 45 = -0.88$

$z2 = (X2 - \mu) / \sigma$

$z2 = (400 - 390) / 45$

$z2 = 10 / 45 = 0.22$

Since z1 is negative, we will have to look at a negative Z-Table and find that p1, the first probability, is **0.18943**.

z2 is positive, so we use a positive Z-Table which yields a probability p2 of **0.58706**.

The final probability is computed by subtracting p1 from p2: :

$p = p2 - p1$

$p = 0.58706 - 0.18943 = 0.39763$

The probability that a student scores between 350 and 400 is 39.763% **(0.39763 * 100).**

Interpretation

1. If a z-score is equal to -1, then it denotes an element, which is 1 standard deviation less than the mean.
2. If a z score is less than 0, then it denotes an element less than the mean.
3. If a z score is greater than 0, then it denotes an element greater than the mean.
4. If the z score is equal to 0, then it denotes an element equal to the mean.
5. If the z score is equal to 1, it denotes an element, which is 1 standard deviation greater than the mean; a z score equal to 2 signifies 2 standard deviations greater than the mean; etc.

Example 2:

A student appeared for two tests. He secured 80 in the first and 75 in the second. The mean and deviation for the first were 70 and 15 respectively, while for the second it was 64 and 12 respectively. What conclusion can you make on comparing the student's performance for both exams?

First test:

Marks secured, $x_1 = 80$

Standard deviation, $\sigma_1 = 15$

Mean, $\mu_1 1 = 70$

$$\text{z score} = \frac{80-70}{15} = 0.667$$

Second test:

Marks secured, $x_2 2 = 75$

Standard deviation, $\sigma_2 = 12$

Mean, $\mu_2 = 64$

$$\text{z score} = \frac{75-64}{12} = 0.9167$$

Since the z score is more for the second test, the student performed better in the second test.

Answer: The student performed better in the second test.

Sample Question

Assume that SAT math scores approximate a normal curve with a mean of 500 and a standard deviation of 100.

(a) Sketch a normal curve and shade in the target area(s) described by each of the following statements:

(i) more than 570 (ii) less than 515 (iii) between 520 and 540

Part A Questions

1.Differentiate Quantitative and Qualitative Data

2.Define Ranked and Nominal Values

3.Compare Continuous and Discrete values.

4.Differentiate Grouped and Ungrouped data.

5.How to calculate Relative Frequency, Cumulative Frequency and percentile

6.Define class interval.

7.Compute mean Mode and median for following 55,60,60,63,63,63,63,65,65.

8.Construct Histogram and Frequency Polygon for following Example.

VIEWING TIME	*f*
35–above	2
30–34	5
25–30	29
20–22	60
15–19	60
10–14	34
5–9	31
0–4	29
Total	250

9.Define Misleading Graph.

10.Indicate Whether each of the following term in Qualitative :ranked or quantitative types

a.etthic Group b.Temperature c.age d.family Size e.Second Place f.gender

Part B

1.Differentiate Type of data and variable used in data analysis with an example.

2. The number of friends by Face book users is summarized in the following frequency distribution.

Data	f
400-above	2
350-399	5
300-349	12
250-299	17
200-249	23
150-199	49
100-149	27
50-99	29
0-49	36
Total	200

i. What is the shape of this distribution?

ii. Find the relative Frequency and Cumulative Frequency.

iii. Find the approximate percentile rank of interval 300-349

iv. Convert to a histogram

v. Why would it not be possible to convert to a stem and leaf display.

3.What is a frequency distribution? Customers who have purchased a particular product rated the usability of the product on a 10 point scale, ranging from 1 (poor) to 10 (excellent) as follows

3	7	2	7	8
3	1	4	10	3
2	5	3	5	8
9	7	6	3	7
8	9	7	3	6

4. What is Z Score? Outline the steps to obtain a Z score.

5. What is Median? Outline the steps to find the median and find the median for the following scores: first, set of five scores 2,8,2,7,6 and second, set of six scores 3,8,9,3,1,8 with steps.

6.Using computation formula for the sum of squares calculate the population standard deviation for the scores in (a) and sample standard deviation for the scores in (b)

(a) 1,3,7,2,0,4,7,3 (b)10,8,5,0,1,1,7,9,2

7. Determie the values of the range and IQR for the following set of data.

a)Retirement ages:60,63,45,63,65,70,55,63,60,65,63

b)Residence changes : 1,3,4,1,0,2,5,8,0,2,3,4,7,11,0,2,3,4

8.Calculate Measure of tendency:1,3,7,20,4,3,7.

9.Calculate Stem and Leaf for following data 12,22,52,46,14,13,26,41,30,120,112,101,105

10.Express each of the following scores as a Z Score: First Mary's intelligence quotient is 135,given a mean of 100 and standard deviation 15.Second, Mary obtained a score of 470 in the Competitive Examination conducted in April 2022, given a mean of 500 and a standard deviation 0f 100.

Appendix

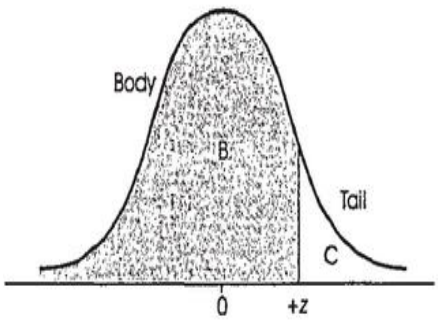
Positive Z score Table

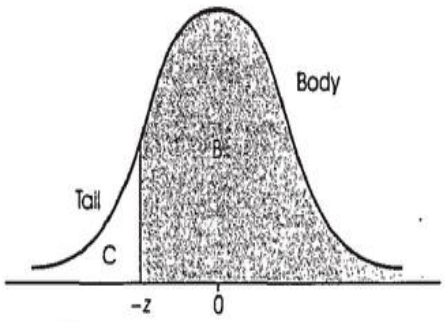
z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998

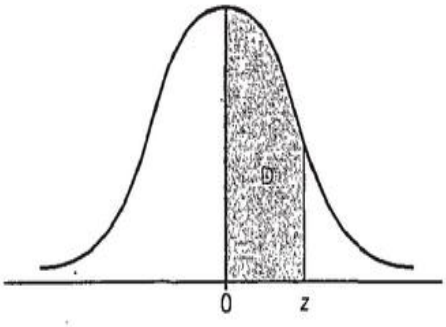
Negative Z score Table

z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
-0	.50000	.49601	.49202	.48803	.48405	.48006	.47608	.47210	.46812	.46414
-0.1	.46017	.45620	.45224	.44828	.44433	.44034	.43640	.43251	.42858	.42465
-0.2	.42074	.41683	.41294	.40905	.40517	.40129	.39743	.39358	.38974	.38591
-0.3	.38209	.37828	.37448	.37070	.36693	.36317	.35942	.35569	.35197	.34827
-0.4	.34458	.34090	.33724	.33360	.32997	.32636	.32276	.31918	.31561	.31207
-0.5	.30854	.30503	.30153	.29806	.29460	.29116	.28774	.28434	.28096	.27760
-0.6	.27425	.27093	.26763	.26435	.26109	.25785	.25463	.25143	.24825	.24510
-0.7	.24196	.23885	.23576	.23270	.22965	.22663	.22363	.22065	.21770	.21476
-0.8	.21186	.20897	.20611	.20327	.20045	.19766	.19489	.19215	.18943	.18673
-0.9	.18406	.18141	.17879	.17619	.17361	.17106	.16853	.16602	.16354	.16109
-1	.15866	.15625	.15386	.15151	.14917	.14686	.14457	.14231	.14007	.13786
-1.1	.13567	.13350	.13136	.12924	.12714	.12507	.12302	.12100	.11900	.11702
-1.2	.11507	.11314	.11123	.10935	.10749	.10565	.10383	.10204	.10027	.09853
-1.3	.09680	.09510	.09342	.09176	.09012	.08851	.08692	.08534	.08379	.08226
-1.4	.08076	.07927	.07780	.07636	.07493	.07353	.07215	.07078	.06944	.06811
-1.5	.06681	.06552	.06426	.06301	.06178	.06057	.05938	.05821	.05705	.05592
-1.6	.05480	.05370	.05262	.05155	.05050	.04947	.04846	.04746	.04648	.04551
-1.7	.04457	.04363	.04272	.04182	.04093	.04006	.03920	.03836	.03754	.03673
-1.8	.03593	.03515	.03438	.03362	.03288	.03216	.03144	.03074	.03005	.02938
-1.9	.02872	.02807	.02743	.02680	.02619	.02559	.02500	.02442	.02385	.02330
-2	.02275	.02222	.02169	.02118	.02068	.02018	.01970	.01923	.01876	.01831
-2.1	.01786	.01743	.01700	.01659	.01618	.01578	.01539	.01500	.01463	.01426
-2.2	.01390	.01355	.01321	.01287	.01255	.01222	.01191	.01160	.01130	.01101
-2.3	.01072	.01044	.01017	.00990	.00964	.00939	.00914	.00889	.00866	.00842
-2.4	.00820	.00798	.00776	.00755	.00734	.00714	.00695	.00676	.00657	.00639
-2.5	.00621	.00604	.00587	.00570	.00554	.00539	.00523	.00508	.00494	.00480
-2.6	.00466	.00453	.00440	.00427	.00415	.00402	.00391	.00379	.00368	.00357
-2.7	.00347	.00336	.00326	.00317	.00307	.00298	.00289	.00280	.00272	.00264
-2.8	.00256	.00248	.00240	.00233	.00226	.00219	.00212	.00205	.00199	.00193
-2.9	.00187	.00181	.00175	.00169	.00164	.00159	.00154	.00149	.00144	.00139
-3	.00135	.00131	.00126	.00122	.00118	.00114	.00111	.00107	.00104	.00100
-3.1	.00097	.00094	.00090	.00087	.00084	.00082	.00079	.00076	.00074	.00071
-3.2	.00069	.00066	.00064	.00062	.00060	.00058	.00056	.00054	.00052	.00050
-3.3	.00048	.00047	.00045	.00043	.00042	.00040	.00039	.00038	.00036	.00035
-3.4	.00034	.00032	.00031	.00030	.00029	.00028	.00027	.00026	.00025	.00024
-3.5	.00023	.00022	.00022	.00021	.00020	.00019	.00019	.00018	.00017	.00017
-3.6	.00016	.00015	.00015	.00014	.00014	.00013	.00013	.00012	.00012	.00011
-3.7	.00011	.00010	.00010	.00010	.00009	.00009	.00008	.00008	.00008	.00008
-3.8	.00007	.00007	.00007	.00006	.00006	.00006	.00006	.00005	.00005	.00005
-3.9	.00005	.00005	.00004	.00004	.00004	.00004	.00004	.00004	.00003	.00003
-4	.00003	.00003	.00003	.00003	.00003	.00003	.00002	.00002	.00002	.00002

Finding Proportion







(A) z	(B) Proportion in Body	(C) Proportion in Tail	(D) Proportion Between Mean and z	(A) z	(B) Proportion in Body	(C) Proportion in Tail	(D) Proportion Between Mean and z
0.00	.5000	.5000	.0000	0.25	.5987	.4013	.0987
0.01	.5040	.4960	.0040	0.26	.6026	.3974	.1026
0.02	.5080	.4920	.0080	0.27	.6064	.3936	.1064
0.03	.5120	.4880	.0120	0.28	.6103	.3897	.1103
0.04	.5160	.4840	.0160	0.29	.6141	.3859	.1141
0.05	.5199	.4801	.0199	0.30	.6179	.3821	.1179
0.06	.5239	.4761	.0239	0.31	.6217	.3783	.1217
0.07	.5279	.4721	.0279	0.32	.6255	.3745	.1255
0.08	.5319	.4681	.0319	0.33	.6293	.3707	.1293
0.09	.5359	.4641	.0359	0.34	.6331	.3669	.1331
0.10	.5398	.4602	.0398	0.35	.6368	.3632	.1368
0.11	.5438	.4562	.0438	0.36	.6406	.3594	.1406
0.12	.5478	.4522	.0478	0.37	.6443	.3557	.1443
0.13	.5517	.4483	.0517	0.38	.6480	.3520	.1480
0.14	.5557	.4443	.0557	0.39	.6517	.3483	.1517
0.15	.5596	.4404	.0596	0.40	.6554	.3446	.1554
0.16	.5636	.4364	.0636	0.41	.6591	.3409	.1591
0.17	.5675	.4325	.0675	0.42	.6628	.3372	.1628
0.18	.5714	.4286	.0714	0.43	.6664	.3336	.1664
0.19	.5753	.4247	.0753	0.44	.6700	.3300	.1700
0.20	.5793	.4207	.0793	0.45	.6736	.3264	.1736
0.21	.5832	.4168	.0832	0.46	.6772	.3228	.1772
0.22	.5871	.4129	.0871	0.47	.6808	.3192	.1808
0.23	.5910	.4090	.0910	0.48	.6844	.3156	.1844
0.24	.5948	.4052	.0948	0.49	.6879	.3121	.1879

UNIT III DESCRIBING RELATIONSHIPS

Correlation –Scatter plots –correlation coefficient for quantitative data – computational formula for correlation coefficient – Regression –regression line –least squares regression line – Standard error of estimate – interpretation of r2 –multiple regression equations –regression towards the mean

3.1 Correlation

Types of Correlation

1.Positive Correlation

2.Negative Correlation

3.No Correlation

3.2 Scatter plots

Types of Correlation with Scatterplot

Linear Relationship

Curvilinear Relationship

3.3 correlation coefficient for quantitative data

3.1 Correlation

Correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate). It's a common tool for describing simple relationships without making a statement about cause and effect.

Correlation coefficient

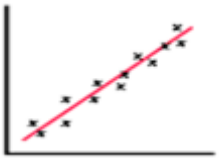
The degree of association is measured by a correlation coefficient, denoted by r. It is sometimes called Pearson's correlation coefficient after its originator and is a measure of linear association.

The correlation coefficient is measured on a scale that varies from + 1 through 0 to – 1. Complete correlation between two variables is expressed by either + 1 or -1.

Types of Correlation

1. **Positive Relationship**

   It Occurs insofar as pairs of scores tend to occupy similar relative positions (high with high and low with low) in their respective distributions

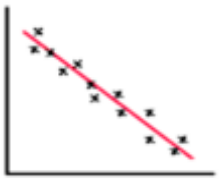


Example

FRIEND	SENT	RECEIVED
Doris	13	14
Steve	9	18
Mike	7	12
Andrea	5	10
John	1	6

2. **Negative Relationship**

   Occurs insofar as pairs of scores tend to occupy dissimilar relative positions (high with low and vice versa) in their respective distributions.

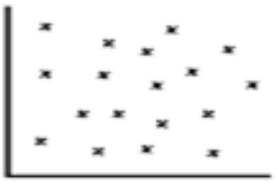


Example

B. NEGATIVE RELATIONSHIP

FRIEND	SENT	RECEIVED
Doris	13	6
Steve	9	10
Mike	7	14
Andrea	5	12
John	1	18

3. **Little or No Relationship**: when there is no



Linear dependence or no relation between the two variables.

Examples

C. LITTLE OR NO RELATIONSHIP

FRIEND	SENT	RECEIVED
Doris	13	10
Steve	9	18
Mike	7	12
Andrea	5	6
John	1	14

3.2 Scatter plots

A graph containing a cluster of dots that represents all pairs of score

Types of correlation

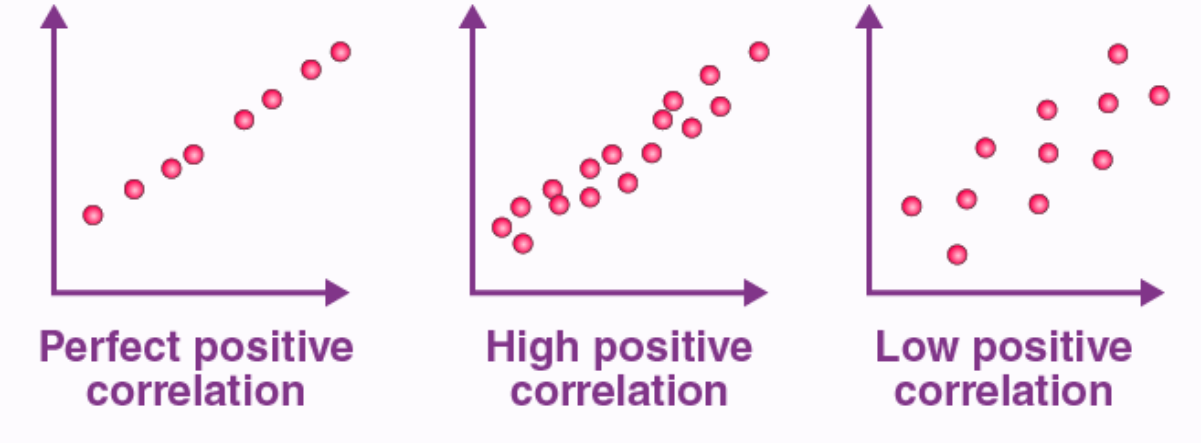
The scatter plot explains the correlation between two attributes or variables. It represents how closely the two variables are connected. There can be three such situations to see the relation between the two variables –

1. Positive Correlation
2. Negative Correlation
3. No Correlation

Positive Correlation

When the points in the graph are rising, moving from left to right, then the scatter plot shows a positive correlation. It means the values of one variable are increasing with respect to another. Now positive correlation can further be classified into three categories:

- **Perfect Positive** – Which represents a perfectly straight line
- **High Positive** – All points are nearby
- **Low Positive** – When all the points are scattered

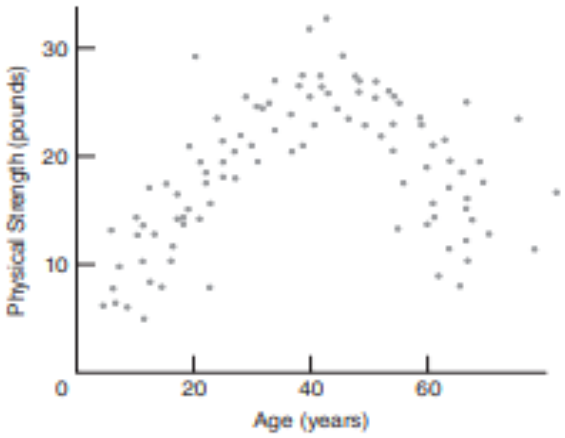


Linear Relationship

A relationship that can be described best with a straight line.

Curvilinear Relationship

A relationship that can be described best with a curved line.



3.3 Correlation coefficient for quantitative data

A number between –1 and 1 that describes the relationship between pairs of variables

Key Properties of r

Named in honor of the British scientist Karl Pearson, the Pearson correlation coefficient, r, can equal any value between –1.00 and +1.00. Furthermore, the following two properties apply:

1. The sign of r indicates the type of linear relationship, whether positive or negative.
2. The numerical value of r, without regard to sign, indicates the strength of the linear relationship

3.4 Computational formula for correlation coefficient

CORRELATION COEFFICIENT (COMPUTATION FORMULA)

$$r = \frac{SP_{xy}}{\sqrt{SS_x SS_y}}$$

where the two sum of squares terms in the denominator are defined as

$$SS_x = \sum \left(X - \bar{X}\right)^2 = \sum X^2 - \frac{\left(\sum X\right)^2}{n}$$

$$SS_y = \sum \left(Y - \bar{Y}\right)^2 = \sum Y^2 - \frac{\left(\sum Y\right)^2}{n}$$

SUM OF PRODUCTS (DEFINITION AND COMPUTATION FORMULAS)

$$SP_{xy} = \sum \left(X - \bar{X}\right)\left(Y - \bar{Y}\right) = \sum XY - \frac{\left(\sum X\right)\left(\sum Y\right)}{n}$$

CALCULATION OF *r*: COMPUTATION FORMULA

A. COMPUTATIONAL SEQUENCE

Assign a value to n (1), representing the number of pairs of scores.
Sum all scores for X (2) and for Y (3).
Find the product of each pair of X and Y scores (4), one at a time, then add all of these products (5).
Square each X score (6), one at a time, then add all squared X scores (7).
Square each Y score (8), one at a time, then add all squared Y scores (9).
Substitute numbers into formulas (10) and solve for SP_{xy}, SS_x, and SS_y
Substitute into formula (11) and solve for r.

B. DATA AND COMPUTATIONS

	CARDS		4	6	8
FRIEND	SENT, X	RECEIVED, Y	XY	X^2	Y^2
Doris	13	14	182	169	196
Steve	9	18	162	81	324
Mike	7	12	84	49	144
Andrea	5	10	50	25	100
John	1	6	6	1	36

1 $n = 5$ 2 $\Sigma X = 35$ 3 $\Sigma Y = 60$ 5 $\Sigma XY = 484$ 7 $\Sigma X^2 = 325$ 9 $\Sigma Y^2 = 800$

10 $$SP_{xy} = \Sigma XY - \frac{(\Sigma X)(\Sigma Y)}{n} = 484 - \frac{(35)(60)}{5} = 484 - 420 = 64$$

$$SS_x = \Sigma X^2 - \frac{(\Sigma X)^2}{n} = 325 - \frac{(35)^2}{5} = 325 - 245 = 80$$

$$SS_y = \Sigma Y^2 - \frac{(\Sigma Y)^2}{n} = 800 - \frac{(60)^2}{5} = 800 - 720 = 80$$

11 $$r = \frac{SP_{xy}}{\sqrt{SS_x SS_y}} = \frac{64}{\sqrt{(80)(80)}} = \frac{64}{80} = .80$$

3.5 REGRESSION

A regression is a statistical technique that relates a dependent variable to one or more independent (explanatory) variables. A

regression model is able to show whether changes observed in the dependent variable are associated with changes in one or more of the explanatory variables.

Regression captures the correlation between variables observed in a data set, and quantifies whether those correlations are statistically significant or not.

3.6 A Regression Line

a regression line is a line that best describes the behavior of a set of data. In other words, it's a line that bestfits the trend of a given data.

Types of regression

The two basic types of regression are

1. Simple linear regression

Simple linear regression uses one independent variable toexplain or predict the outcome of the dependent variable Y

2. Multiple linear regressions

Multiple linear regressions use two or more independentvariables to predict the outcome

Predictive Errors

Prediction error refers to the difference between the predicted values made by some model and theactual values

3.7 LEAST SQUARES REGRESSION LINE

The placement of the regression line minimizes not the total predictive error but the total squared predictive error, that is, the total for all squared predictive errors. When located in this fashion, the regression line is often referred to as the least squares regression line.

The Least Squares Regression Line is the line that minimizes the sum of the residuals squared. The residual is the vertical distance between the observed point and the predicted point, and it is

calculated by subtracting ˆy from y.

LEAST SQUARES REGRESSION EQUATION

$$Y' = bX + a$$

SOLVING FOR *b*

$$b = r\sqrt{\frac{SS_y}{SS_x}}$$

SOLVING FOR *a*

$$a = \overline{Y} - b\overline{X}$$

DETERMINING THE LEAST SQUARES REGRESSION EQUATION

A. COMPUTATIONAL SEQUENCE

Determine values of SS_x, SS_y and r (1) by referring to the original correlation analysis in Table 6.3.
Substitute numbers into the formula (2) and solve for b.
Assign values to $\bar{X}$ and $\bar{Y}$ (3) by referring to the original correlation analysis in Table 6.3.
Substitute numbers into the formula (4) and solve for a.
Substitute numbers for b and a in the least squares regression equation (5).

B. COMPUTATIONS

1 $SS_x = 80^*$

$SS_y = 80^*$

$r = .80$

2 $$b = r\sqrt{\frac{SS_Y}{SS_X}} = .80\sqrt{\frac{80}{80}} = .80$$

3 $\bar{X} = 7^{**}$

$\bar{Y} = 12^{**}$

4 $$a = \bar{Y} - (b)(\bar{X}) = 12 - (.80)(7) = 12 - 5.60 = 6.40$$

5 $$Y' = (b)(X) + a$$
$$= (.80)(X) + 6.40$$

Solving for Y

$$\begin{aligned} Y' &= .80(11) + 6.40 \\ &= 8.80 + 6.40 \\ &= 15.20 \end{aligned}$$

3.8 STANDARD ERROR OF ESTIMATE

Standard Error of Estimate (sy|x) : A rough measure of the average amount of predictive error.

STANDARD ERROR OF ESTIMATE (DEFINITION FORMULA)

$$s_{y|x} = \sqrt{\frac{SS_{y|x}}{n-2}} = \sqrt{\frac{\Sigma(Y-Y')^2}{n-2}}$$

STANDARD ERROR OF ESTIMATE (COMPUTATION FORMULA)

$$s_{y|x} = \sqrt{\frac{SS_y\left(1-r^2\right)}{n-2}}$$

CALCULATION OF THE STANDARD ERROR OF ESTIMATE, $S_{y|x}$

A. COMPUTATIONAL SEQUENCE

Assign values to SS_y and r (1) by referring to previous work with the least squares regression equation in Table 7.1.

Substitute numbers into the formula (2) and solve for $s_{y|x}$.

B. COMPUTATIONS

1 $SS_y = 80$

$r = .80$

2 $s_{y|x} = \sqrt{\frac{SS_y(1-r^2)}{n-2}} = \sqrt{\frac{80\left(1-\left[.80\right]^2\right)}{5-2}} = \sqrt{\frac{80(.36)}{3}} = \sqrt{\frac{28.80}{3}} = \sqrt{9.60}$

$= 3.10$

3.9 I N T E R P R E TAT I O N O F r 2

R-Squared (R^2 or the coefficient of determination) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable. In other words, r-squared shows how well the data fit the regression model (the goodness of fit).

R-squared can take any values between 0 to 1. Although the statistical measure provides some useful insights regarding the regression model, the user should not rely only on the measure in the assessment of a statistical model.

r^2 INTERPRETATION

$$r^2 = \frac{SS_{Y'}}{SS_Y} = \frac{SS_Y - SS_{Y|X}}{SS_Y}$$

$$SS_{y'} = \sum (Y' - \bar{Y})^2$$

3.10 Multiple Regression

Multiple regression analysis is a statistical technique that analyzes the relationship between two or more variables and uses the information to estimate the value of the dependent variables. In multiple regression, the objective is to develop a model that describes a dependent variable y to more than one independent variable.

The multiple regression equation is given by

$$y = a + b_{1\times1} + b_{2\times2} + \ldots\ldots + b_{kxk}$$

where x_1, x_2, $\ldots x_k$ are the k independent variables and y is the dependent variable.

3.11 Regression towards the Mean

A tendency for scores, particularly extreme scores, to shrink toward the mean

The Formula for the Percent of Regression to the Mean

You can estimate exactly the percent of regression to the mean in any given situation. The formula is:

$$Prm = 100(1 - r)$$

where:

- Prm = the percent of regression to the mean
- r = the correlation between the two measures

Consider the following four cases:

- if r = 1, there is no (i.e., 0%) regression to the mean
- if r = .5, there is 50% regression to the mean
- if r = .2, there is 80% regression to the mean
- if r = 0, there is 100% regression to the mean

Table 7.4

REGRESSION TOWARD THE MEAN: BATTING AVERAGES OF TOP 10 HITTERS IN MAJOR LEAGUE BASEBALL DURING 2014 AND HOW THEY FARED DURING 2015

	BATTING AVERAGES*		
TOP 10 HITTERS (2014)	2014	2015	REGRESS TOWARD MEAN?
1. J. Alture	.341	.313	Yes
2. V. Martinez	.335	.282	Yes
3. M. Brantley	.327	.310	Yes
4. A. Beltre	.324	.287	Yes
5. J. Abreu	.317	.290	Yes
6. R. Cano	.314	.287	Yes
7. A. McCutchen	.314	.292	Yes
8. M. Cabrera	.313	.338	No
9. B. Posey	.311	.318	No
10. B. Revere	.306	.306	No

Act

Review Questions

Part A

1.What do You mean by least square method?

2.What is Regression Analysis?

3.What is Correlation?

4.Define Interpretation $R^{2.}$

5.Define SEE with Example.

6.What is Scatterplots?

7. Consider Helen sent 10 greeting card sto her friends and she received back 8 cards, what is the kind of relationship it is? Brief on it.

8.Define Correlation Co efficient.

9.Differentiate simple Regression and Multiple Regression

10.What is Regression Line?

PART B

1.Explain about Scatter plot and Various types of Scatterplot with neat diagram.

2.Problem :Correlation Co efficient (3 Types)

3. a. Calculate the correlation co efficient for the heights of fathers(X) and their sons(y) with the data presented below.

x	**66**	**68**	**68**	**70**	**71**	**72**	**72**
y	**68**	**70**	**69**	**72**	**72**	**72**	**74**

4.The values of x and their corresponding values of y are presented below.

x	**0.5**	**1.5**	**2.5**	**3.5**	**4.5**	**5.5**	**6.5**
y	**2.5**	**3.5**	**5.5**	**4.5**	**6.5**	**8.5**	**10.5**

i) Find the Least square regression line y=ax+b.

ii) Estimate the values of y when x=10.

5. Calculate Standard Error Estimate

Couple	X	Y
A	1	2
B	3	4
C	2	3
D	3	2
E	1	0
F	2	3

6. Estimate whether the following pairs of scores for x and y a positive relationship, negative relationship or no relationship

x	**64**	**40**	**30**	**71**	**55**	**31**	**61**	**42**	**57**
y	**66**	**79**	**98**	**65**	**76**	**83**	**68**	**80**	**72**

a) Construct a scatterplot for x and y verify that scatter does not describe a pronounced curvilinear.

b) Calculate **r** using the Computation formula.

7.Each of the following pairs represents the number of licensed drivers (X) and the number of cars (Y) for seven house in my neighborhood.

Drivers (X)	**Cars (Y)**
5	4
5	3
2	2
2	2
3	2
1	1
2	2

1. Construct a scatterplot to verify a lack of pronounced Curvilinearity.
2. Determine the least squares equation for these data.(Remember, you will first have to calculate r,SSy and SSx)
3. Determine the standard error of estimate ,Sy/x given that n=7.

8.Consider the following dataset with one response variable y and two predictor variables x1 and x2.

y	140	155	159	179	192	200	212	215
x1	60	62	67	70	71	72	75	78
x2	22	25	24	20	15	14	14	11

UNIT IV PYTHON LIBRARIES FOR DATA WRANGLING 9

Basics of Numpy arrays –aggregations –computations on arrays –comparisons, masks, boolean logic – fancy indexing – structured arrays – Data manipulation with Pandas – data indexing and selection – operating on data – missing data – Hierarchical indexing – combining datasets – aggregation and grouping – pivot tables

Python NumPy Array: (Numerical Python)

Numpy array is a powerful N-dimensional array object which is in the form of rows and columns. We can initialize NumPy arrays from nested Python lists and access it elements.

4.1 Basics of Numpy arrays

NumPy array is a powerful N-dimensional array object and its use in linear algebra,Fourier transform, and random number capabilities. It provides an array object much faster than traditional Python lists.

Types of Array

1.Attributes of arrays
Determining the size, shape, memory consumption, and data types of arrays
2.Indexing of arrays
Getting and setting the value of individual array elements
3.Slicing of arrays
Getting and setting smaller sub arrays within a larger array
4.Reshaping of arrays
Changing the shape of a given array
5.Joining and splitting of arrays
Combining multiple arrays into one, and splitting one array into many

Attributes of arrays

Determining the size, shape, memory consumption, and data types of arrays

Single-dimensional Numpy Array:

```
importnumpy as np

a=np.array([1,2,3])

print(a)
```

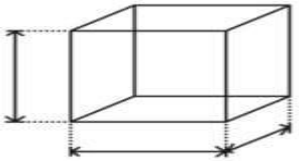
Output – [1 2 3]

Multi-dimensional Array:

```
1 a=np.array([(1,2,3),(4,5,6)])

2 print(a)
```

O/P – [[1 2 3]
[4 5 6]]

ndim:



You can find the dimension of the array, whether it is a two-dimensional array or a single dimensional array. So, let us see this practically how we can find the dimensions. In the below code, with the help of 'ndim' function, I can find whether the array is of single dimension or multi dimension.

```
importnumpy as np
a =np.array([(1,2,3),(4,5,6)])
print(a.ndim)
```

Output – 2

Since the output is 2, it is a two-dimensional array (multi dimension).

- **itemsize**:
  You can calculate the byte size of each element. In the below code, I have defined a single dimensional array and with the help of 'itemsize' function, we can find the size of each element.

```
1  importnumpy as np
2  a =np.array([(1,2,3)])
3  print(a.itemsize)
```

Output – 4

- **dtype:**
  You can find the data type of the elements that are stored in an array. So, if you want to know the data type of a particular element, you can use 'dtype' function which will print the datatype along with the size. In the below code, I have defined an array where I have used the same function.

```
1  importnumpy as np
2  a =np.array([(1,2,3)])
3  print(a.dtype)
```

Output – int32

You can find the size and shape of the array using 'size' and 'shape' function respectively.

```
1  import numpy as np
2  a = np.array([(1,2,3,4,5,6)])
3  print(a.size)
4  print(a.shape)
```

Output – 6 (1,6)

- **linspace**
  This is another operation in python numpy which returns evenly spaced numbers over a specified interval. Consider the below example:

```
1  import numpy as np
2  a=np.linspace(1,3,10)
3  print(a)
```

- Output – [1. 1.22222222 1.44444444 1.66666667 1.88888889 2.11111111 2.33333333 2.55555556 2.77777778 3.]

Array Indexing: Accessing Single Elements

If you are familiar with Python's standard list indexing, indexing in NumPy will feel quite familiar. In a one-dimensional array, you can access the ith value (counting from
zero) by specifying the desired index in square brackets, just as with Python lists:

In[5]: x1
Out[5]: array([5, 0, 3, 3, 7, 9])
In[6]: x1[0]
Out[6]: 5
In[7]: x1[4]

Out[7]: 7
To index from the end of the array, you can use negative indices:
In[8]: x1[-1]
Out[8]: 9
In[9]: x1[-2]
Out[9]: 7
In a multidimensional array, you access items using a comma-separated tuple of indices:
In[10]: x2
Out[10]: array([[3, 5, 2, 4],
[7, 6, 8, 8],
[1, 6, 7, 7]])
In[11]: x2[0, 0]
Out[11]: 3
In[12]: x2[2, 0]
Out[12]: 1
In[13]: x2[2, -1]
Out[13]: 7
You can also modify values using any of the above index notation:
In[14]: x2[0, 0] = 12
x2
Out[14]: array([[12, 5, 2, 4],
[7, 6, 8, 8],
[1, 6, 7, 7]])

Keep in mind that, unlike Python lists, NumPy arrays have a fixed type. This means,for example, that if you attempt to insert a floating-point value to an integer array, thevalue will be silently truncated.

In[15]: x1[0] = 3.14159 *# this will be truncated!*
x1
Out[15]: array([3, 0, 3, 3, 7, 9])

Array Slicing: Accessing Subarrays

- **slicing:**
  As you can see the 'reshape' function has showed its magic. Now, let's take another operation i.e Slicing. Slicing is basically extracting particular set of elements from an array. This slicing operation is pretty much similar to the one which is there in the list as well. Consider the following example:

Before getting into the above example, let's see a simple one. We have an array and we need a particular element (say 3) out of a given array. Let's consider the below example:

```
1 import numpy as np
2 a=np.array([(1,2,3,4),(3,4,5,6)])
3 print(a[0,2])
```

Output – 3

Here, the array(1,2,3,4) is your index 0 and (3,4,5,6) is index 1 of the python numpy array. Therefore, we have printed the second element from the zeroth index.
Taking one step forward, let's say we need the 2nd element from the zeroth and first index of the array. Let's see how you can perform this operation:

```
1  import numpy as np
2  a=np.array([(1,2,3,4),(3,4,5,6)])
3  print(a[0:,2])
```

Output – [3 5]

Here colon represents all the rows, including zero. Now to get the 2nd element, we'll call index 2 from both of the rows which gives us the value 3 and 5 respectively.

Next, just to remove the confusion, let's say we have one more row and we don't want to get its 2nd element printed just as the image above. What we can do in such case?
Consider the below code:

```
1 import numpy as np
2 a=np.array([(8,9),(10,11),(12,13)])
3 print(a[0:2,1])
```

Output – [9 11]

As you can see in the above code, only 9 and 11 gets printed. Now when I have written 0:2, this does not include the second index of the third row of an array. Therefore, only 9 and 11 gets printed else you will get all the elements i.e [9 11 13].

- To access subarrays with the slice notation, marked by the colon (:) character.
- The NumPy slicing syntax follows that of the standard Python list; to access a slice ofan array x, use this:

x[start:stop:step]

If any of these are unspecified, they default to the values start=0, stop=*size of dimension*, step=1. We'll take a look at accessing subarrays in one dimension and in multiple dimensions.

One-dimensional subarrays

```
In[16]: x = np.arange(10)
x
Out[16]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
In[17]: x[:5] # first five elements
Out[17]: array([0, 1, 2, 3, 4])
In[18]: x[5:] # elements after index 5
```

```
Out[18]: array([5, 6, 7, 8, 9])
In[19]: x[4:7] # middle subarray
Out[19]: array([4, 5, 6])
In[20]: x[::2] # every other element
Out[20]: array([0, 2, 4, 6, 8])
In[21]: x[1::2] # every other element, starting at index 1
Out[21]: array([1, 3, 5, 7, 9])
```

A potentially confusing case is when the step value is negative. In this case, the defaults for start and stop are swapped. This becomes a convenient way to reverse an array:

```
In[22]: x[::-1] # all elements, reversed
Out[22]: array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
In[23]: x[5::-2] # reversed every other from index 5
Out[23]: array([5, 3, 1])
```

Multidimensional subarrays

Multidimensional slices work in the same way, with multiple slices separated by commas.

For example:

```
In[24]: x2
Out[24]: array([[12, 5, 2, 4],
[ 7, 6, 8, 8],
[ 1, 6, 7, 7]])
In[25]: x2[:2, :3] # two rows, three columns
Out[25]: array([[12, 5, 2],
[ 7, 6, 8]])
In[26]: x2[:3, ::2] # all rows, every other column
Out[26]: array([[12, 2],
[ 7, 8],
[ 1, 7]])
```

Finally, subarray dimensions can even be reversed together:

```
In[27]: x2[::-1, ::-1]
Out[27]: array([[ 7, 7, 6, 1],
[ 8, 8, 6, 7],
[ 4, 2, 5, 12]])
```

Accessing array rows and columns. One commonly needed routine is accessing singlerows or columns of an array. You can do this by combining indexing and slicing,

using an empty slice marked by a single colon (:):

```
In[28]: print(x2[:, 0]) # first column of x2
[12 7 1]
In[29]: print(x2[0, :]) # first row of x2
[12 5 2 4]
```

In the case of row access, the empty slice can be omitted for a more compact syntax:

```
In[30]: print(x2[0]) # equivalent to x2[0, :]
[12 5 2 4]
```

Subarrays as no-copy views

One important—and extremely useful—thing to know about array slices is that theyreturn views rather than copies of the array data. This is one area in which NumPyarray slicing differs from Python list slicing: in lists, slices will be copies. **Consider ourtwo-dimensional array from before:**

```
In[31]: print(x2)
[[12 5 2 4]
[ 7 6 8 8]
[ 1 6 7 7]]
```

Let's extract a 2×2 subarray from this:

```
In[32]: x2_sub = x2[:2, :2]
print(x2_sub)
[[12 5]
[ 7 6]]
```

Now if we modify this subarray, we'll see that the original array is changed! Observe:

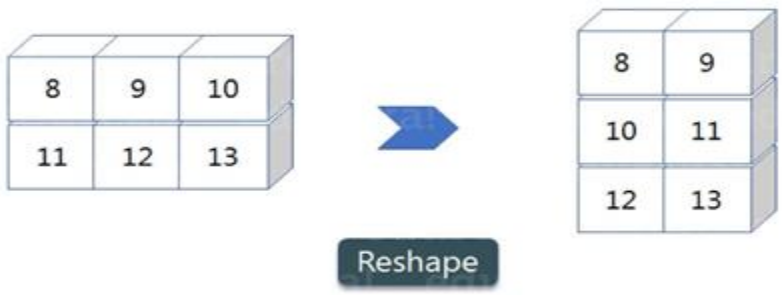
```
In[33]: x2_sub[0, 0] = 99
print(x2_sub)
[[99 5]
[ 7 6]]
In[34]: print(x2)
[[99 5 2 4]
[ 7 6 8 8]
[ 1 6 7 7]]
```

This default behavior is actually quite useful: it means that when we work with largedatasets, we can access and process pieces of these datasets without the need to copythe underlying data buffer.

Reshaping of Arrays

reshape:

Reshape is when you change the number of rows and columns which gives a new view to an object. Now, let us take an example to reshape the below array:



As you can see in the above image, we have 3 columns and 2 rows which has converted into 2 columns and 3 rows. Let me show you practically how it's done.

```
1  import numpy as np
2  a = np.array([(8,9,10),(11,12,13)])
3  print(a)
4  a=a.reshape(3,2)
5  print(a)
```

Output– [[8 9 10] [11 12 13]] [[8 9] [10 11] [12 13]]

Array Concatenation and Splitting

All of the preceding routines worked on single arrays. It's also possible to combinemultiple arrays into one, and to conversely split a single array into multiple arrays.

Concatenation of arrays

Concatenation, or joining of two arrays in NumPy, is primarily accomplished through the routines np.concatenate, np.vstack, and np.hstack. np.concatenate takes a tuple or list of arrays as its first argument, as we can see here:

```
In[43]:x = np.array([1, 2, 3])
y = np.array([3, 2, 1])
np.concatenate([x, y])
```

Out[43]: array([1, 2, 3, 3, 2, 1])

Splitting of Arrays

The opposite of concatenation is splitting, which is implemented by the functions**np.split, np.hsplit, and np.vsplit**. For each of these, we can pass a list of indicesgiving the split points:

```
In[50]: x = [1, 2, 3, 99, 99, 3, 2, 1]
x1, x2, x3 = np.split(x, [3, 5])
print(x1, x2, x3)
```

output:[1 2 3] [99 99] [3 2 1]

The related functions np.hsplitand np.vsplit are similar:

```
In[51]: grid = np.arange(16).reshape((4, 4))
grid
Out[51]: array([[ 0, 1, 2, 3],
                [ 4, 5, 6, 7],
                [ 8, 9, 10, 11],
                [12, 13, 14, 15]])
In[52]: upper, lower = np.vsplit(grid, [2])
print(upper)
```

```
print(lower)
Output :[[0 1 2 3]
         [4 5 6 7]]
```

4.2 Aggregations

The Python numpy aggregate functions are sum, min, max, mean, average, product, median, standard deviation, variance, argmin, argmax, percentile, cumprod, cumsum, and corrcoef.

Functions

min() function returns the item with the lowest value, or the item with the lowest value in an iterable.

If the values are strings, an alphabetically comparison is done.

Input: x = min(5, 10)
Output:5

max() function returns the item with the highest value, or the item with the highest value in an iterable.

If the values are strings, an alphabetically comparison is done.

max(*iterable*)

Input: x = max(5, 10)
Output:10

Mean, Median, and Mode

What can we learn from looking at a group of numbers?

In Machine Learning (and in mathematics) there are often three values that interests us:

- **Mean** - The average value
- **Median** - The mid point value
- **Mode** - The most common value

Mean

```
import numpy

speed=[99,86,87,88,111,86,103,87,94,78,77,85,86]

x=numpy.mean(speed)
```

```
print(x)
```

o/p

```
89.76923076923077
```

Median

```
import numpy

speed=[99,86,87,88,111,86,103,87,94,78,77,85,86]

x=numpy.median(speed)

print(x)
```

o/p
87.0

Mode

```
import numpy
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]
x=numpy.mode(speed)
print(x)
```

o/p
86,87

Standard deviation is a number that describes how spread out the values are.

```
import numpy

speed=[86,87,88,86,87,85,86]

x=numpy.std(speed)

print(x)
```

o/p
0.90

4.3 Computations on Arrays

NumPy'sbroadcasting functionality. Broadcasting is simply aset of rules for applying binary ufuncs (addition, subtraction, multiplication, etc.) onarrays of different sizes.

we can perform other operations such as subtraction, multiplication and division. Consider the below example:

```
1 import numpy as np
```

```
2 x= np.array([(1,2,3),(3,4,5)])
3 y= np.array([(1,2,3),(3,4,5)])
4 print(x-y)
5 print(x*y)
6 print(x/y)
```

Output – [[0 0 0] [0 0 0]]
[[1 4 9] [9 16 25]]
[[1. 1. 1.][1. 1. 1.]

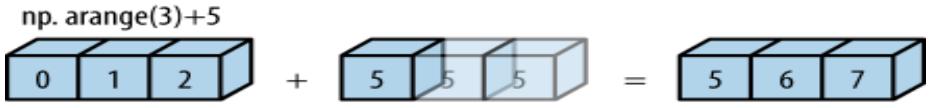
Rules of Broadcasting

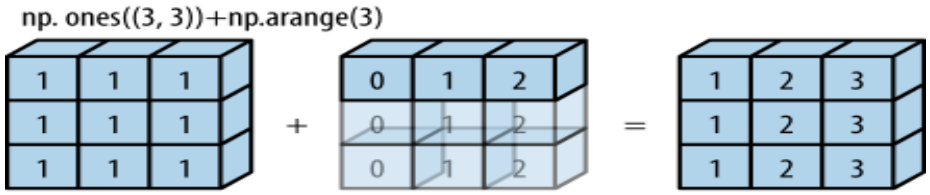
Broadcasting in NumPy follows a strict set of rules to determine the interaction between the two arrays:

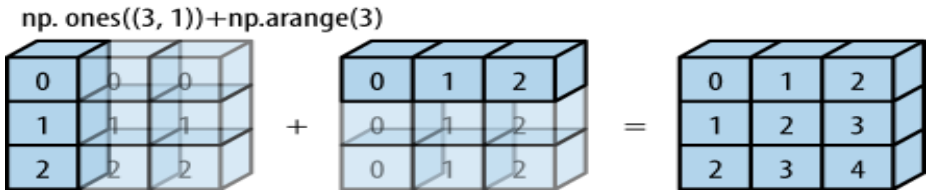
• Rule 1: If the two arrays differ in their number of dimensions, the shape of the one with fewer dimensions is padded with ones on its leading (left) side.

• Rule 2: If the shape of the two arrays does not match in any dimension, the array with shape equal to 1 in that dimension is stretched to match the other shape.

• Rule 3: If in any dimension the sizes disagree and neither is equal to 1, an error is raised.





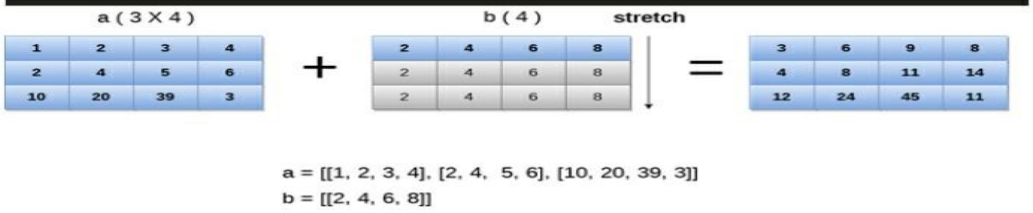


Example

```
import numpy as np
a = np.array([[1,2,3,4],[2,4,5,6],[10,20,39,3]])
b = np.array([2,4,6,8])
print("\nprinting array a..")
print(a)
print("\nprinting array b..")
print(b)
print("\nAdding arrays a and b ..")
c = a + b;
print(c)
```

Output

```
printing array a..
[[ 1  2  3  4]
 [ 2  4  5  6]
 [10 20 39  3]]

printing array b..
[2 4 6 8]

Adding arrays a and b ..
[[ 3  6  9 12]
 [ 4  8 11 14]
 [12 24 45 11]]
```



4.4 Comparisons, Masks, and Boolean Logic

NumPy also implements comparisonoperators such as < (less than) and > (greater than) as element-wise ufuncs.the result of these comparison operators is always an array with a Boolean data type.

All six of the standard comparison operations are available:

```
In[4]: x = np.array([1, 2, 3, 4, 5])
In[5]: x <3 # less than
Out[5]: array([ True, True, False, False, False], dtype=bool)
In[6]: x >3 # greater than
Out[6]: array([False, False, False, True, True], dtype=bool)
In[7]: x <= 3 # less than or equal
Out[7]: array([ True, True, True, False, False], dtype=bool)
In[8]: x >= 3 # greater than or equal
Out[8]: array([False, False, True, True, True], dtype=bool)
In[9]: x != 3 # not equal
Out[9]: array([ True, True, False, True, True], dtype=bool)
In[10]: x == 3 # equal
Out[10]: array([False, False, True, False, False], dtype=bool)
```

Boolean Arrays as Masks

In the preceding section we looked at aggregates computed directly on Boolean arrays. A more powerful pattern is to use Boolean arrays as masks, to select particular subsets of the data themselves. Returning to our x array from before, suppose we want an array of all values in the array that are less than, say, 5:

```
x
array([[5, 0, 3, 3],
       [7, 9, 3, 5],
       [2, 4, 7, 6]])
```

We can obtain a Boolean array for this condition easily, as we've already seen:

```
x< 5
array([[False,  True,  True,  True],
       [False, False,  True, False],
[ True,  True, False, False]], dtype=bool)
```

Now to select these values from the array, we can simply index on this Boolean array; this is known as a masking operation:

Boolean Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Example

```
x = 5
print(x > 3 and x < 10)
```

Output:

True

```
x = 5
```

```
print(x > 3 or x < 4)
```
Output
True

```
x = 5
print(not(x > 3 and x < 10))

# returns False because not is used to reverse the result
```
Output
False

4.5 Fancy Indexing

Fancy Indexing means passing an array of indices to access multiple array elements at once.
Fancy indexing allows you to index a numpy array using the following:

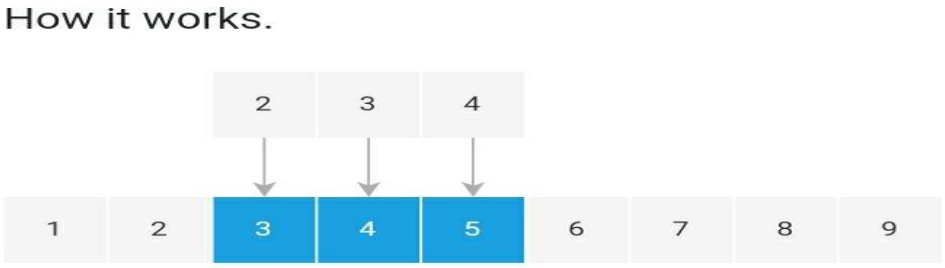
- Another numpy array
- A Python list
- A sequence of integers

Let's see the following example:

```
import numpy as np
a = np.arange(1, 10)
print(a)
indices = np.array([2, 3, 4])
print(a[indices])
```

Output:

```
[1 2 3 4 5 6 7 8 9]
[3 4 5]
```



Numpy Fancy Indexing

First, use the arange() function to create a numpy array that includes numbers from 1 to 9:
[1 2 3 4 5 6 7 8 9]
Second, create a second numpy array for indexing:
indices = np.array([2, 3, 4])
Third, use the indices array for indexing the a array:
print(a[indices])

Summary

Fancy indexing allows you to index an array using another array, a list, or a sequence of integers.

```
In [50]: arr8=np.arange(24).reshape(6,4)

In [51]: arr8
Out[51]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11],
                [12, 13, 14, 15],
                [16, 17, 18, 19],
                [20, 21, 22, 23]])

In [ ]: arr8[[0,2,4]]

Out[52]: array([[ 0,  1,  2,  3],
                [ 8,  9, 10, 11],
                [16, 17, 18, 19]])
```

4.6 NumPy's Structured Arrays

Structure array uses data containers called fields. Each data field can contain data of any type and size. Array elements can be accessed with the help of dot notation.

Properties of Structured array

1.All structs in array have same number of fields.
2.All structs have same fields names.

Creating Structured Arrays

Character	Description	Example
'b'	Byte	np.dtype('b')
'i'	Signed integer	np.dtype('i4') == np.int32
'u'	Unsigned integer	np.dtype('u1') == np.uint8
'f'	Floating point	np.dtype('f8') == np.int64
'c'	Complex floating point	np.dtype('c16') == np.complex128
'S','a'	string	np.dtype('S5')
'U'	Unicode string	np.dtype('U') == np.str_
'V'	Raw data (void)	np.dtype('V') == np.void

Structured array data types can be specified in a number of ways. Earlier, we saw the dictionary method:

In[10]: np.dtype({'names':('name', 'age', 'weight'),

'formats':('U10', 'i4', 'f8')})

Out[10]: dtype([('name', '<U10'), ('age', '<i4'), ('weight', '<f8')])

Example

```
import numpy as np
data_type = [('name', 'S15'), ('class', int), ('height', float)]
students_details = [('James', 5, 48.5), ('Nail', 6, 52.5),('Paul', 5, 42.10), ('Pit', 5, 40.11)]
# create a structured array
students = np.array(students_details, dtype=data_type)
print("Original array:")
print(students)
print("Sort by height")
print(np.sort(students, order='height'))
```

Sample Output:

```
Original array:
[(b'James', 5, 48.5 ) (b'Nail', 6, 52.5 ) (b'Paul', 5, 42.1 )
 (b'Pit', 5, 40.11)]
Sort by height
[(b'Pit', 5, 40.11) (b'Paul', 5, 42.1 ) (b'James', 5, 48.5 )
 (b'Nail', 6, 52.5 )]
```

4.7 PANDAS: Panel Dataset

Pandas is a Python library used for working with data sets.It has functions for analyzing, cleaning, exploring, and manipulating data.The name "Pandas" has a reference to both "Panel Data",

Why Use Pandas?

- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.
- Relevant data is very important in data science.

Installation of Pandas

If you have Python and PIP already installed on a system, then installation of Pandas is very easy.

Install it using this command:

C:\Users\Your Name>py -m pip install pandas

If this command fails, then use a python distribution that already has Pandas installed like, Anaconda, Spyder etc.

Import Pandas

Once Pandas is installed, import it in your applications by adding the import keyword:

import pandas

SERIES

A Pandas Series is like a column in a table.It is a one-dimensional array holding data of any type.

Example

Create a simple Pandas Series from a list:

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a)
print(myvar)
```

Output:

```
0    1
1    7
2    2
dtype: int64
```

DataFrame

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

Example

Create a simple Pandas DataFrame:

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```

Result

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

Load Files Into a DataFrame

If your data sets are stored in a file, Pandas can load them into a DataFrame.

Example
Load a comma separated file (CSV file) into a DataFrame:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df)
```

output

```
Duration  PulseMaxpulse  Calories
0        60   110      130    409.1
1        60   117      145    479.0
2        60   103      135    340.0
3        45   109      175    282.4
4        45   117      148    406.0
..      ...   ...      ...      ...
164      60   105      140    290.8
165      60   110      145    300.4
166      60   115      145    310.2
167      75   120      150    320.4
168      75   125      150    330.4

[169 rows x 4 columns]
```

Named Indexes

With the index argument, you can name your own indexes.

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

Result

```
calories  duration
  day1      420       50
  day2      380       40
  day3      390       45
```

4.8 Data indexing and selection

Data indexing

Selecting values from particular rows and columns in a dataframe is known as Indexing. By using Indexing, we can select all rows and some columns or some rows and all columns.

Program

```
importpandasaspd
data=pd.Series([‘a’,‘b’,’c’],
index=[1,2,3])
data
```

Output

```
1                   a
3                   b
5                   c
dtype: object
```

Now, here Python offers two types of indices

- Explicit
- Implicit

Explicit Indexing:

For the above dataset if we pass the command as,

ds[1] it uses explicit indices

```
# If we pass the above command ds[1], the output will be'a'
```

This is Explicit Indexing. Whereas, if we pass the command ds[1:3] it will use the implicit index style,

The output for the command ds[1:3] will be,

```
3                                                                   b
5                                                                   c
dtype: object
```

These slicing and indexing can lead to some sort of confusion. To avoid this, Python offers some special indexer attributes:

1. **loc**

The loc attribute allows indexing and slicing that always references the explicit index

```
In[14]: data.loc[1]
Out[14]: 'a'
In[15]: data.loc[1:3]
Out[15]: 1 a
3 b
dtype: object
```

2.iloc

The iloc attribute allows indexing and slicing that always references the implicit index style

```
In[16]: data.iloc[1]
Out[16]: 'b'
In[17]: data.iloc[1:3]
Out[17]: 3 b
         5 c
dtype: object
```

- A third indexing attribute, ix, is a hybrid of the two, and for Series objects is equivalentto standard [] based indexing.

Data selection

```
In[7]: # slicing by explicit index
data['a':'c']
Out[7]: a 0.25
b 0.50
c 0.75
dtype: float64
In[8]: # slicing by implicit integer index
data[0:2]
Out[8]: a 0.25
b 0.50
dtype: float64
In[9]: # masking
data[(data >0.3) &(data <0.8)]
Out[9]: b 0.50
c 0.75
dtype: float64
In[10]: # fancy indexing
data[['a', 'e']]
Out[10]: a 0.25
e 1.25
dtype: float64
```

First, let me import the libraries.

```
In [1]:
1 import pandas as pd
2 import numpy as np
```

To index data, you can the square brackets. To show this, let's create a series using the arrange method.

```
In [2]:
1 obj=pd.Series(np.arange(5),
2                index=["a","b","c","d","e"])
```

Let's see the obj variable.

```
In [3]:
1 obj
```

```
Out[3]: a    0
        b    1
        c    2
        d    3
        e    4
        dtype: int32
```

Working with Index in Series

Let's print the value of c.

```
In [4]:
1 obj["c"]
```

```
Out[4]: 2
```

You can do the same thing by entering the index number in square brackets.

```
In [5]:
1 obj[2]
```

```
Out[5]: 2
```

You can slice the data.

```
In [6]:
1 obj[0:3]
```

```
Out[6]: a    0
        b    1
        c    2
        dtype: int32
```

Selecting in Series

Let's select the specific rows.

```
In [7]:  1 obj[["a","c"]]

Out[7]: a    0
        c    2
        dtype: int32
```

You can do the same thing by using the index number.

```
In [8]:  1 obj[[0,2]]

Out[8]: a    0
        c    2
        dtype: int32
```

Filtering in Series

Let's see the values less than 2.

```
In [9]:  1 obj[obj<2]

Out[9]: a    0
        b    1
        dtype: int32
```

You can slice the values.

You can assign a value to the sliced piece.

```
In [10]:  1 obj["a":"c"]
```

```
Out[10]: a    0
         b    1
         c    2
         dtype: int32
```

```
In [11]:  1 obj["b":"c"]=5
          2 obj
```

```
Out[11]: a    0
         b    5
         c    5
         d    3
         e    4
         dtype: int32
```

Selecting in DataFrame

To show how to index in DataFrame, let me create a DataFrame.

```
In [12]:  1 data=pd.DataFrame(
          2     np.arange(16).reshape(4,4),
          3     index=["London","Paris",
          4            "Berlin","Istanbul"],
          5     columns=["one","two","three","four"])
          6 data
```

Out[12]:

	one	two	three	four
London	0	1	2	3
Paris	4	5	6	7
Berlin	8	9	10	11
Istanbul	12	13	14	15

Let's see the column named two.

```
In [13]:  1  data["two"]
```

```
Out[13]: London      1
         Paris       5
         Berlin      9
         Istanbul    13
         Name: two, dtype: int32
```

You can select more than one column.

```
In [14]:  1  data[["one","two"]]
```

Out[14]:

	one	two
London	0	1
Paris	4	5
Berlin	8	9
Istanbul	12	13

You can slice the rows.

```
In [15]:  1  data[:3]
```

Out[15]:

	one	two	three	four
London	0	1	2	3
Paris	4	5	6	7
Berlin	8	9	10	11

Filtering in DataFrame

You can use the boolean expression for selection.

```
In [16]:  1  data[data["four"]>5]
```

Out[16]:

	one	two	three	four
Paris	4	5	6	7
Berlin	8	9	10	11
Istanbul	12	13	14	15

You can assign data to specific values.

```
In [17]:  1  data[data<5]=0
          2  data
```

Out[17]:

	one	two	three	four
London	0	0	0	0
Paris	0	5	6	7
Berlin	8	9	10	11
Istanbul	12	13	14	15

Selecting with iloc and loc methods

You can use the iloc method to select a row using the row's index.

```
In [18]:  1  data.iloc[1]
```

```
Out[18]: one      0
         two      5
         three    6
         four     7
         Name: Paris, dtype: int32
```

You can select the specific columns of the row.

```
In [19]:  1 data.iloc[1,[1,2,3]]
```

```
Out[19]: two      5
         three    6
         four     7
         Name: Paris, dtype: int32
```

You can select specific columns of multiple rows.

```
In [20]:  1 data.iloc[[1,3],[1,2,3]]
```

Out[20]:

	two	three	four
Paris	5	6	7
Istanbul	13	14	15

Let's take a look at the loc method. You need to use names for loc.

```
In [21]:  1 data.loc["Paris",["one","two"]]
```

```
Out[21]: one    0
         two    5
         Name: Paris, dtype: int32
```

Let's select the rows up to Paris and the column named four.

```
In [22]:  1 data.loc[:"Paris","four"]
```

```
Out[22]: London    0
         Paris     7
         Name: four, dtype: int32
```

To show how to use the negative index, let me create a Series named toy_data.

In [23]:
```
toy_data=pd.Series(np.arange(5),
                   index=["a","b","c",
                          "d","e"])
toy_data
```

Out[23]:
```
a    0
b    1
c    2
d    3
e    4
dtype: int32
```

Now, I'm going to use a negative index.

In [24]:
```
toy_data[-1]
```

Out[24]: 4

4.9 OPERATING ON DATA

Loading data

First of all, let's import the libraries.

In [1]:
```
import pandas as pd
import numpy as np
```

Let me create two variables named s1 and s2.

In [2]:
```
s1=pd.Series(np.arange(4),
             index=["a","c","d","e"])
s2=pd.Series(np.arange(5),
             index=["a","c","e","f","g"])
```

Let's take a look at these variables.

In [3]:
```
print(s1)
print(s2)
```

```
a    0
c    1
d    2
e    3
dtype: int32
a    0
c    1
e    2
f    3
g    4
dtype: int32
```

Addition in Pandas

You can add two variables.

In [4]:
```
s1+s2
```

Out[4]:
```
a    0.0
c    2.0
d    NaN
e    5.0
f    NaN
g    NaN
dtype: float64
```

You can do the same for the data frame. To show this, let's create two datasets named df1 and df2.

In [5]:
```
df1=pd.DataFrame(
    np.arange(6).reshape(2,3),
    columns=list("ABC"),
    index=["Tim","Tom"])
df2=pd.DataFrame(
    np.arange(9).reshape(3,3),
    columns=list("ACD"),
    index=["Tim","Kate","Tom"])
```

Let's take a look at data frames.

```
In [6]:
1 print(df1)
2 print(df2)
```

```
      A  B  C
Tim   0  1  2
Tom   3  4  5
       A  C  D
Tim    0  1  2
Kate   3  4  5
Tom    6  7  8
```

You can add two data frames.

```
In [7]:
1 df1+df2
```

Out[7]:

	A	B	C	D
Kate	NaN	NaN	NaN	NaN
Tim	0.0	NaN	3.0	NaN
Tom	9.0	NaN	12.0	NaN

When you can apply arithmetic operations with different indexes, a special value such as zero can be assigned to the missing values.
Let's assign zeros to values that do not match df2 of df1 when adding two data frames.

```
In [8]:
1 df1.add(df2,fill_value=0)
```

Out[8]:

	A	B	C	D
Kate	3.0	NaN	4.0	5.0
Tim	0.0	1.0	3.0	2.0
Tom	9.0	4.0	12.0	8.0

Multiplication & Division in Pandas
You can inverse the values in the dataset.

```
In [9]:  1  1/df1
```

Out[9]:

	A	B	C
Tim	inf	1.00	0.5
Tom	0.333333	0.25	0.2

Let me show multiplication.

```
In [10]:  1  df1*3
```

Out[10]:

	A	B	C
Tim	0	3	6
Tom	9	12	15

You can also use the mul method for multiplication.

```
In [11]:  1  df1.mul(3)
```

Out[11]:

	A	B	C
Tim	0	3	6
Tom	9	12	15

I'm going to handle the df2 again. Let me print this data.

```
In [12]:  1  df2
```

Out[12]:

	A	C	D
Tim	0	1	2
Kate	3	4	5
Tom	6	7	8

Let's take a look at the first row of the df2.

```
In [13]:
1 s=df2.iloc[1]
2 s
```

```
Out[13]: A    3
         C    4
         D    5
         Name: Kate, dtype: int32
```

As you can see, this row is in Serial structure. Let's subtract s from df2.

```
In [14]:
1 df2-s
```

Out[14]:

	A	C	D
Tim	-3	-3	-3
Kate	0	0	0
Tom	3	3	3

To select a column in the dataset, you can use the [] symbol.

```
In [15]:
1 s2=df2["A"]
2 s2
```

```
Out[15]: Tim     0
         Kate    3
         Tom     6
         Name: A, dtype: int32
```

You can perform arithmetic operations with a series and a data frame.

```
In [16]:
1 df2.sub(s2,axis="index")
```

Out[16]:

	A	C	D
Tim	0	1	2
Kate	0	1	2
Tom	0	1	2

Applying a Function in Pandas

You can use NumPy functions with Pandas data structures To demonstrate this, let's first create a dataset named df.

In [17]:
```
df=pd.DataFrame(
    np.random.randn(4,3),
    columns=list("ABC"),
    index=["Kim","Susan","Tim","Tom"])
df
```

Out[17]:

	A	B	C
Kim	-0.173717	-1.126917	-0.595042
Susan	-0.641672	-0.073913	-1.828588
Tim	-0.389124	-1.786140	0.553646
Tom	-0.062436	-0.251933	0.872391

Let's use the abs method in NumPy for df.

In [18]:
```
np.abs(df)
```

Out[18]:

	A	B	C
Kim	0.173717	1.126917	0.595042
Susan	0.641672	0.073913	1.828588
Tim	0.389124	1.786140	0.553646
Tom	0.062436	0.251933	0.872391

You can also apply a function for each column or row. To do this, you can use the apply () method. First, let's create a function.

In [19]:
```
f=lambda x:x.max()-x.min()
```

Let's apply this function to the df using the apply method.

```
In [20]:  1 df.apply(f)
```

```
Out[20]: A    0.579236
         B    1.712227
         C    2.700979
         dtype: float64
```

You can execute this function on each row using axis = 1.

```
In [21]:  1 df.apply(f,axis=1)
```

```
Out[21]: Kim      0.953200
         Susan    1.754676
         Tim      2.339785
         Tom      1.124324
         dtype: float64
```

You can use this function for the data frame. To show that, let's write a simple function.

```
In [22]:  1 def f(x):
          2     return x**2
```

Now, let me apply this function to the df.

```
In [23]:  1 df.apply(f)
```

Out[23]:

	A	B	C
Kim	0.030178	1.269942	0.354075
Susan	0.411743	0.005463	3.343735
Tim	0.151417	3.190294	0.306524
Tom	0.003898	0.063470	0.761066

4.10 MISSING DATA

In DataFrame sometimes many datasets simply arrive with missing data, either because it exists and was not collected or it never existed**.**

In Pandas missing data is represented by two value:

1.None: None is a Python singleton object that is often used for missing data in Python code.

2.NaN : NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation

Loading Data
First of all, let's import the libraries.

In [1]:
```
import pandas as pd
import numpy as np
```

Missing data in the Pandas is represented by the value NaN (Not a Number).

In [2]:
```
s=pd.Series(["Sam",np.nan,"Tim","Kim"])
s
```

Out[2]:
```
0    Sam
1    NaN
2    Tim
3    Kim
dtype: object
```

You can use the isnull method to see missing data in the data.

In [3]:
```
s.isnull()
```

Out[3]:
```
0    False
1     True
2    False
3    False
dtype: bool
```

The notnull method does the opposite of the isnull method. Let me show that.

```
In [4]:  1 s.notnull()
```

```
Out[4]: 0     True
        1    False
        2     True
        3     True
        dtype: bool
```

Let's assign None to a value in data.

```
In [5]:  1 s[3]=None
         2 s.isnull()
```

```
Out[5]: 0    False
        1     True
        2    False
        3     True
        dtype: bool
```

If you want to remove the missing data, you can use the dropna method.

```
In [6]:  1 s.dropna()
```

```
Out[6]: 0    Sam
        2    Tim
        dtype: object
```

Handling Missing Data in Data Frame

Now, let's show how to remove missing data in DataFrame structure. First, let's import nan from Numpy.

```
In [7]:  1 from numpy import nan as NA
```

Let me create a data frame named df.

In [8]:
```
df=pd.DataFrame([[1,2,3],[4,NA,5],
                 [NA,NA,NA]])
df
```

Out[8]:

	0	1	2
0	1.0	2.0	3.0
1	4.0	NaN	5.0
2	NaN	NaN	NaN

By default, the dropna method removes rows with missing data.

In [9]:
```
df.dropna()
```

Out[9]:

	0	1	2
0	1.0	2.0	3.0

The how = all argument removes all rows with missing data.

In [10]:
```
df.dropna(how="all")
```

Out[10]:

	0	1	2
0	1.0	2.0	3.0
1	4.0	NaN	5.0

Let's take a look at df.

In [11]:
```
df
```

Out[11]:

	0	1	2
0	1.0	2.0	3.0
1	4.0	NaN	5.0
2	NaN	NaN	NaN

The axis = 1 argument is used to remove the column with all missing data. First, let's assign the missing data to the second column of the df dataset.

```
In [12]:
1 df[1]=NA
2 df
```

Out[12]:

	0	1	2
0	1.0	NaN	3.0
1	4.0	NaN	5.0
2	NaN	NaN	NaN

Let's remove the columns with missing data.

```
In [13]:
1 df.dropna(axis=1,how="all")
```

Out[13]:

	0	2
0	1.0	3.0
1	4.0	5.0
2	NaN	NaN

Let's take a look at the df dataset again.

```
In [14]:
1 df
```

Out[14]:

	0	1	2
0	1.0	NaN	3.0
1	4.0	NaN	5.0
2	NaN	NaN	NaN

If you want to get rows with a certain number of values, you can use the thresh argument.

```
In [15]:  1 df.dropna(thresh=3)
          2 df
```

Out[15]:

	0	1	2
0	1.0	NaN	3.0
1	4.0	NaN	5.0
2	NaN	NaN	NaN

If you want to assign another value instead of missing data, you can use the fillna method.

```
In [16]:  1 df.fillna(0)
```

Out[16]:

	0	1	2
0	1.0	0.0	3.0
1	4.0	0.0	5.0
2	0.0	0.0	0.0

REPLACE()

```
importpandas as pd
df={

  "Array_1": [49.50, 70],

  "Array_2": [65.1, 49.50]
}
data=pd.DataFrame(df)
print(data.replace(49.50, 60))
```

OUTPUT

```
Array_1  Array_2
0    60.0    65.1
1    70.0    60.0
```

4.11 Hierarchical Indexes

Hierarchical indexing is a method of creating structured group relationships in the dataset. Data frames can have hierarchical indexes. To show this, let me create a dataset.

In [10]:
```
df=pd.DataFrame(
     np.arange(12).reshape(4,3),
     index=[["a","a","b","b"],
            [1,2,1,2]],
     columns=[["num","num","ver"],
              ["math","stat","geo"]])
df
```

Out[10]:

		num		ver
		math	stat	geo
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

Notice that in this dataset, both row and column have hierarchical indexes. You can name hierarchical levels. Let's show this.

In [11]:
```
df.index.names=["class","exam"]
df.columns.names=["field","lesson"]
df
```

Out[11]:

field		num		ver
lesson		math	stat	geo
class	exam			
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

Selecting in Hierarchical Indexing

You can select subgroups of data. For example, let's select the index named num.

```
In [12]:  1  df["num"]
```

Out[12]:

class	exam	lesson	math	stat
a	1		0	1
	2		3	4
b	1		6	7
	2		9	10

What is Swaplevel?

Sometimes, you may want to swap the level of the indexes. You can use the swaplevel method for this. The swaplevel method takes two levels and returns a new object. For example, let's swap the class and exam indexes in the dataset.

```
In [13]:  1  df.swaplevel("class","exam")
```

Out[13]:

		field	num	ver
		lesson	math	stat	geo
exam	class			
1	a	0	1	2
2	a	3	4	5
1	b	6	7	8
2	b	9	10	11

Sorting in Hierarchical Indexing

To sort the indexes by level, you can use the sort_index method. For example, let's sort the dataset by level 1.

```
In [14]:  1  df.sort_index(level=1)
```

Out[14]:

		field	num		ver
		lesson	math	stat	geo
class		exam			
a		1	0	1	2
b		1	6	7	8
a		2	3	4	5
b		2	9	10	11

4.12 Combining Dataset

With pandas, you can **merge**, **join**, and **concatenate** your datasets, allowing you to unify and better understand your data as you analyze it.

- **merge()** for combining data on common columns or indices
- **join()** for combining data on a key column or an index
- **concat()** for combining DataFrames across rows or columns

Merging Datasets in Pandas

A guide on how to merge datasets using the merge, join, concat methods in Pandas

First, let's import Pandas and NumPy.

```
In [1]:  1  import pandas as pd
         2  import numpy as np
```

Merging Data Frames

You can use the merge or join methods to combine rows using one or more keys. First, I'm going to cover the merge method. To show this, let's create two datasets.

```
In [2]:  1  d1=pd.DataFrame(
         2      {"key":["a","b","c","c","d","e"],
         3       "num1":range(6)})
         4  d2=pd.DataFrame(
         5      {"key":["b","c","e","f"],
         6       "num2":range(4)})
```

Let's take a look at these datasets

In [3]:
```
1 print(d1)
2 print(d2)
```

```
  key  num1
0   a     0
1   b     1
2   c     2
3   c     3
4   d     4
5   e     5
  key  num2
0   b     0
1   c     1
2   e     2
3   f     3
```

You can combine these two datasets using the merge function.

In [4]:
```
1 pd.merge(d1, d2)
```

Out[4]:

	key	num1	num2
0	b	1	0
1	c	2	1
2	c	3	1
3	e	5	2

Pandas merges data automatically based on a key column. You can select the column you want to merge with the on parameter.

In [5]:
```
1 pd.merge(d1, d2, on='key')
```

Out[5]:

	key	num1	num2
0	b	1	0
1	c	2	1
2	c	3	1
3	e	5	2

Notice that unmatched data in the two datasets is dropped. You can see these values using how= “outer”.

```
In [8]:  1  pd.merge(d1,d2,how="outer")
```

Out[8]:

	key	num1	num2
0	a	0.0	NaN
1	b	1.0	0.0
2	c	2.0	1.0
3	c	3.0	1.0
4	d	4.0	NaN
5	e	5.0	2.0
6	f	NaN	3.0

You can combine datasets based on a particular key column. To do this, you can use the how = “left” parameter.

```
In [9]:  1  pd.merge(d1,d2,how="left")
```

Out[9]:

	key	num1	num2
0	a	0	NaN
1	b	1	0.0
2	c	2	1.0
3	c	3	1.0
4	d	4	NaN
5	e	5	2.0

Now, let’s use the how = “true” parameter.

```
In [10]:  1  pd.merge(d1,d2,how="right")
```

Out[10]:

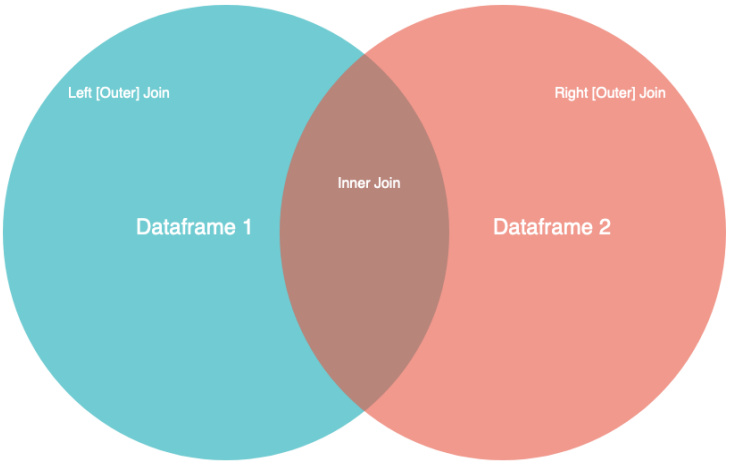
	key	num1	num2
0	b	1.0	0
1	c	2.0	1
2	c	3.0	1
3	e	5.0	2
4	f	NaN	3

You can use the how = “inner” to find common values in two datasets,

```
In [11]:  1  pd.merge(d1, d2, how='inner')
```

Out[11]:

	key	num1	num2
0	b	1	0
1	c	2	1
2	c	3	1
3	e	5	2



Outer Join

Combining with the Join Method

You can use the join method to combine datasets using indexes.

```
In [21]:  1  left.join(right)
```

Out[21]:

	Sam	Kim	Tom	Tim
a	7	8	1.0	2.0
b	9	10	NaN	NaN
e	11	12	NaN	NaN
f	13	14	NaN	NaN

So, the right dataset was added to the left dataset. You can see all the values with how = “outer”.

```
In [22]:  1  left.join(right,how="outer")
```

Out[22]:

	Sam	Kim	Tom	Tim
a	7.0	8.0	1.0	2.0
b	9.0	10.0	NaN	NaN
c	NaN	NaN	3.0	4.0
d	NaN	NaN	5.0	6.0
e	11.0	12.0	NaN	NaN
f	13.0	14.0	NaN	NaN

Concatination() Method:

You can use the axis = 1 to index a hierarchical column.

```
In [32]:  1  x=pd.concat([data1, data2, data4],
          2               axis=1,
          3               keys=['one', 'two', 'three'])
          4  x
```

Out[32]:

	one	two	three
a	0.0	NaN	10.0
b	1.0	NaN	11.0
c	NaN	2.0	12.0
d	NaN	3.0	NaN
e	NaN	4.0	NaN

4.13 Aggregation and Grouping

```
import pandas as pd
df = pd.DataFrame([[9, 4, 8, 9],
                   [8, 10, 7, 6],
                   [7, 6, 8, 5]],
                   columns=['Maths', 'English',
                            'Science', 'History'])
print(df)
```

OUTPUT

	Maths	English	Science	History
0	9	4	8	9
1	8	10	7	6
2	7	6	8	5

Aggregation in Pandas

Aggregation in pandas provides various functions that perform a mathematical or logical operation on our dataset and returns a summary of that function. Aggregation can be used to get a summary of columns in our dataset like getting sum, minimum, maximum, etc. from a particular column of our dataset

Function Description:

- *sum()* :Compute sum of column values
- *min()* :Compute min of column values
- *max()* :Compute max of column values
- *mean()* :Compute mean of column
- *size()* :Compute column sizes
- *describe()* :Generates descriptive statistics
- *first()* :Compute first of group values
- *last()* :Compute last of group values
- *count()* :Compute count of column values
- *std()* :Standard deviation of column
- *var()* :Compute variance of column
- *sem()* :Standard error of the mean of column

1.**df.sum()**

```
Maths      24
English    20
Science    23
History    20
dtype: int64
```

2.df.describe()

	Maths	English	Science	History
count	3.0	3.000000	3.000000	3.000000
mean	8.0	6.666667	7.666667	6.666667
std	1.0	3.055050	0.577350	2.081666
min	7.0	4.000000	7.000000	5.000000
25%	7.5	5.000000	7.500000	5.500000
50%	8.0	6.000000	8.000000	6.000000
75%	8.5	8.000000	8.000000	7.500000
max	9.0	10.000000	8.000000	9.000000

- We used agg() function to calculate the sum, min, and max of each column in our dataset.

3.df.agg(['sum', 'min', 'max'])

	Maths	English	Science	History
sum	24	20	23	20
min	7	4	7	5
max	9	10	8	9

Grouping in Pandas

Grouping is used to group data using some criteria from our dataset. It is used as split-apply-combine strategy.

- Splitting the data into groups based on some criteria.
- Applying a function to each group independently.
- Combining the results into a data structure.

Examples:

- We use groupby() function to group the data on “Maths” value. It returns the

object as result.

```
a = df.groupby('Maths')
a.first()
```

Maths	English	Science	History
7	6	8	5
8	10	7	6
9	4	8	9

```
b = df.groupby(['Maths', 'Science'])
b.first()
```

Maths	Science	English	History
7	8	6	5
8	7	10	6
9	8	4	9

4.14 PIVOT TABLE

Pandas pivot tables offer a powerful tool to perform these analysis techniques with Python. Sometimes the difference between pivot tables and groupby is confusing.

What is the difference between the pivot_table and the groupby?
The groupby method is generally enough for two-dimensional operations, but pivot_table is used for multi-dimensional grouping operations.
How to use the pivot_table?
DataFrame has a pivot_table method. Let's create the table we created with groupby using pivot_table.

In [6]:
```
df.pivot_table(
    "score",
    index="lesson",
    columns="class")
```

Out[6]:

class	A	B	C
lesson			
math	55	75	65
stat	70	60	80

Now let's create a pivot table with hierarchical indexes.

In [7]:
```
df.pivot_table(
    ["sibling","score"],
    index=["class","lesson"],
    columns="sex")
```

Out[7]:

		score		sibling	
	sex	F	M	F	M
class	lesson				
A	math	NaN	55.0	NaN	1.0
	stat	85.0	55.0	1.0	1.0
B	math	NaN	75.0	NaN	2.0
	stat	45.0	75.0	2.0	2.0
C	math	NaN	65.0	NaN	3.0
	stat	65.0	95.0	3.0	3.0

Multi-Level Pivot Tables

You can also create multi-level pivot tables. For example, let's divide the sibling variable into intervals with the cut method.

In [10]:
```
sib=pd.cut(df["sibling"],[0,2,3])
```

Now let's create a multi-level dataset using this sibling variable.

In [11]:
```
df.pivot_table("score",
               ["lesson",sib],
               "class",fill_value=0)
```

Out[11]:

	class	A	B	C
lesson	sibling			
math	(0, 2]	55	75	0
	(2, 3]	0	0	65
stat	(0, 2]	70	60	0
	(2, 3]	0	0	80

UNIT V DATA VISUALIZATION 9

Importing Matplotlib – Line plots – Scatter plots – visualizing errors – density and contour plots – Histograms – legends – colors – subplots – text and annotation – customization – three dimensional plotting – Geographic Data with Basemap – Visualization with Seaborn.

5.1 DATA VISUALIZATION

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

Advantages

- Easily sharing information.
- Interactively explore opportunities.
- Visualize patterns and relationships.

Disadvantages

- Biased or inaccurate information.
- Correlation doesn't always mean causation.
- Core messages can get lost in translation.

5.2 Line plots

Importing Matplotlib

The Pyplot package can be referred to as plt.

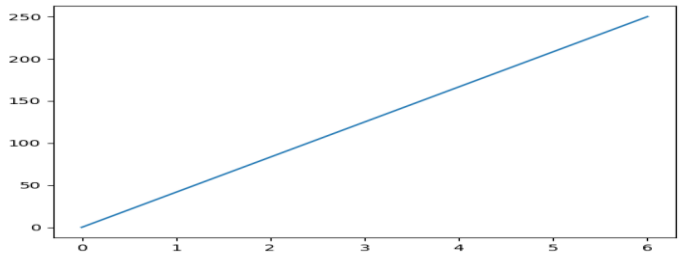
```
import matplotlib.pyplot as plt
```

Example

Draw a line in a diagram from position (0,0) to position (6,250):

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([0, 6])
ypoints = np.array([0, 250])
plt.plot(xpoints, ypoints)
plt.show()
```

Result:



Plotting Without Line

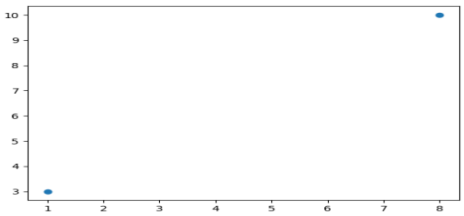
To plot only the markers, you can use *shortcut string notation* parameter 'o', which means 'rings'.

Example

Draw two points in the diagram, one at position (1, 3) and one in position (8, 10):

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints, 'o')
plt.show()
```

Result:

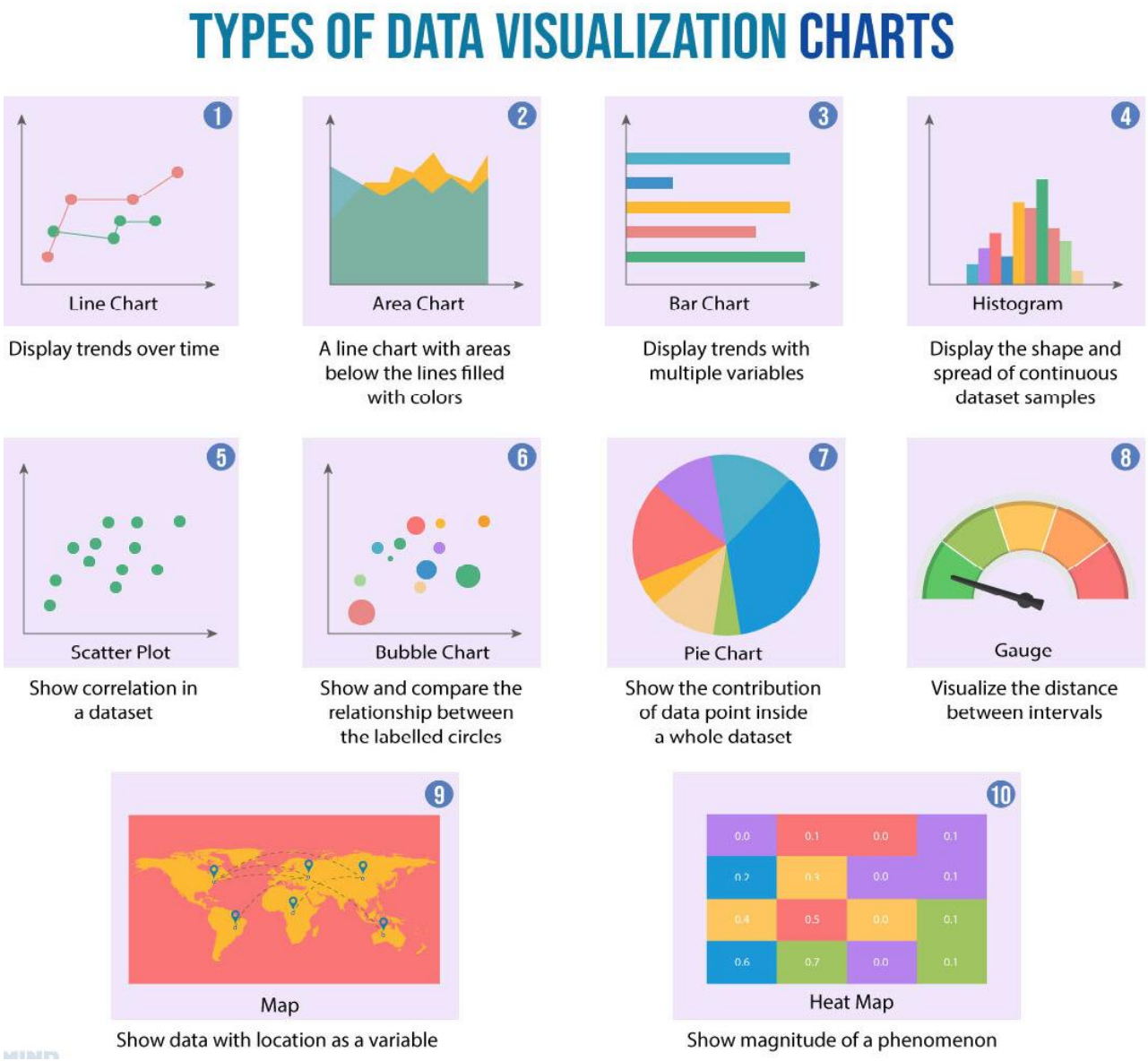


Types of data visualizations

- **Tables**: This consists of rows and columns used to compare variables. Tables can show a great deal of information in a structured way, but they can also overwhelm users that are simply looking for high-level trends.
- **Pie charts and stacked bar charts:** These graphs are divided into sections that represent parts of a whole. They provide a simple way to organize data and compare the size of each component to one other.
- **Line charts and area charts:** These visuals show change in one or more quantities by plotting a series of data points over time and are frequently used within predictive analytics. Line graphs utilize lines to demonstrate these

changes while area charts connect data points with line segments, stacking variables on top of one another and using color to distinguish between variables.

- **Histograms**: This graph plots a distribution of numbers using a bar chart (with no spaces between the bars), representing the quantity of data that falls within a particular range. This visual makes it easy for an end user to identify outliers within a given dataset.
- **Scatter plots**: These visuals are beneficial in reveling the relationship between two variables, and they are commonly used within regression data analysis. However, these can sometimes be confused with bubble charts, which are used to visualize three variables via the x-axis, the y-axis, and the size of the bubble.



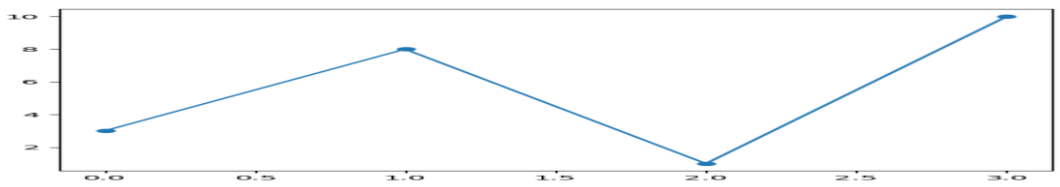
Matplotlib Markers

use the keyword argument marker to emphasize each point with a specified marker:

Example
Mark each point with a circle:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'o')
plt.show()
```

Result:



Matplotlib Line

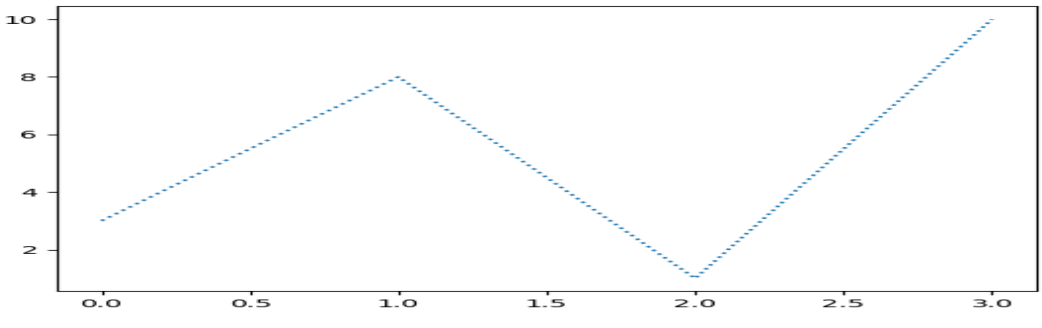
Linestyle

You can use the keyword argument linestyle, or shorter ls, to change the style of the plotted line:

Example
Use a dotted line:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```

Result:

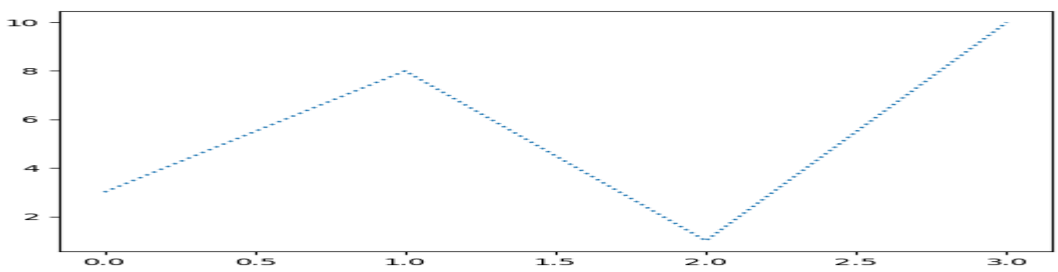


Shorter Syntax
The line style can be written in a shorter syntax:
linestyle can be written as ls.
dotted can be written as :.

dashed can be written as --.

plt.plot(ypoints, ls = ':')

Result:



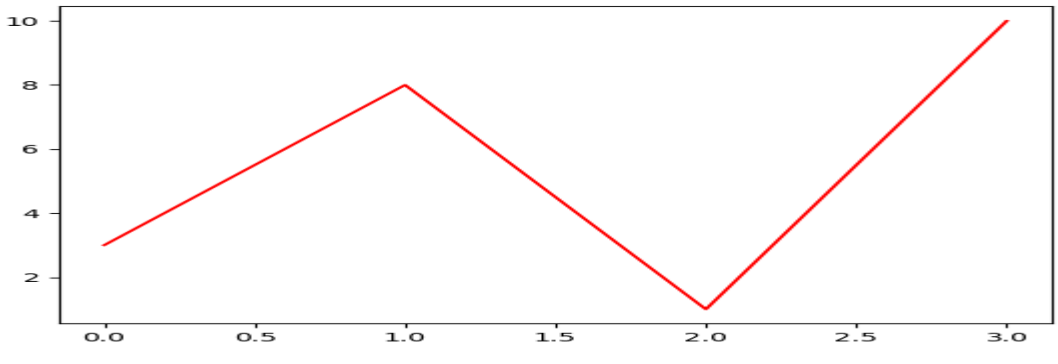
Line Color

You can use the keyword argument color or the shorter c to set the color of the line:

Example
Set the line color to red:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, color = 'r')
plt.show()
```

Result:

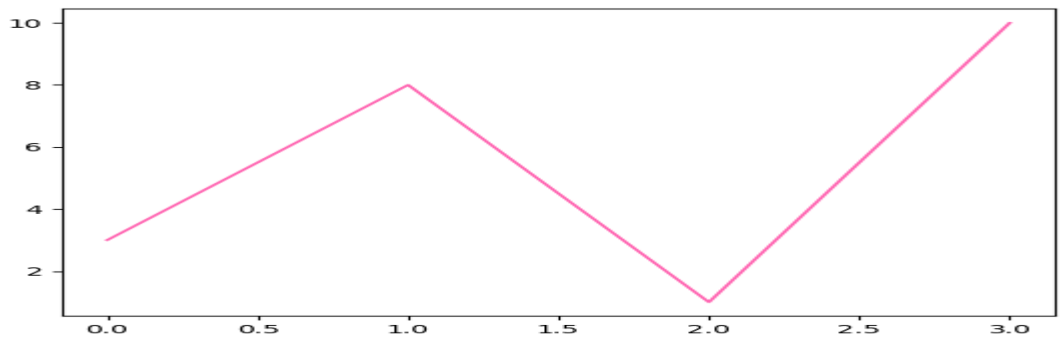


Example

Plot with the color named "hotpink":

```
...
plt.plot(ypoints, c = 'hotpink')
...
```

Result:



Line Width

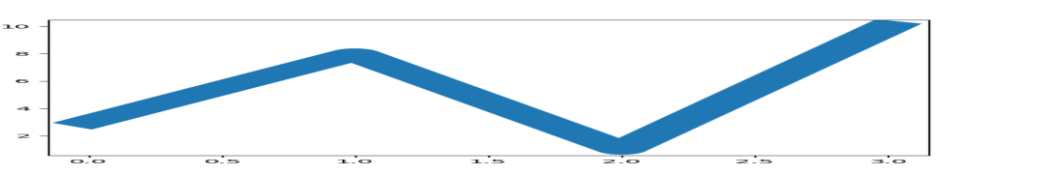
You can use the keyword argument linewidth or the shorter lw to change the width of the line.
The value is a floating number, in points:

Example

Plot with a 20.5pt wide line:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, linewidth = '20.5')
plt.show()
```

Result:



Multiple Lines

You can plot as many lines as you like by simply adding more plt.plot() functions:

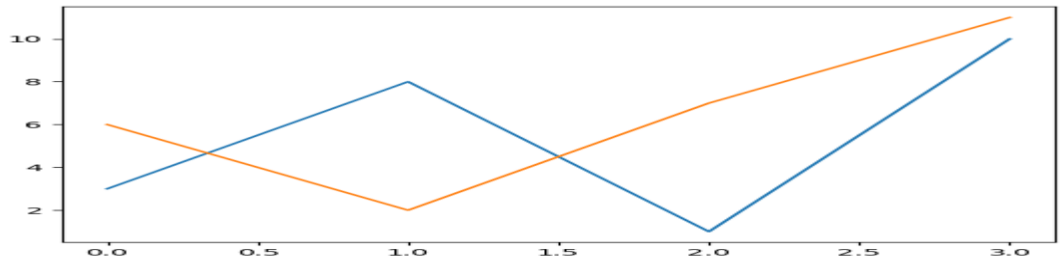
Example

Draw two lines by specifying a plt.plot() function for each line:

```
import matplotlib.pyplot as plt
import numpy as np
y1 = np.array([3, 8, 1, 10])
```

```
y2 = np.array([6, 2, 7, 11])
plt.plot(y1)
plt.plot(y2)
plt.show()
```

Result:



5.3 Scatter plots

Matplotlib Scatter

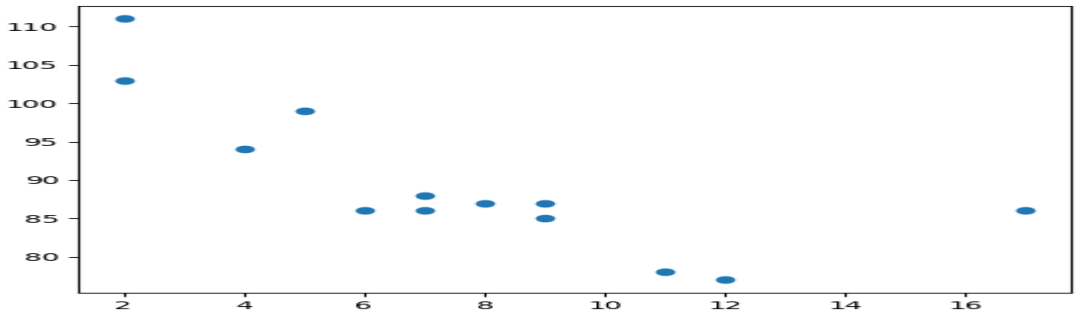
With Pyplot, you can use the scatter() function to draw a scatter plot.

The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

Example
A simple scatter plot:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)
plt.show()
```

Result:



Combine Color Size and Alpha

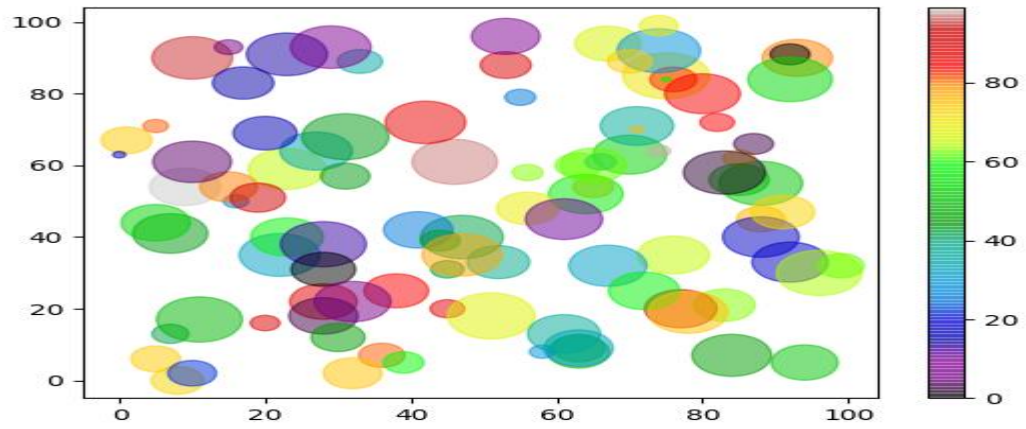
You can combine a colormap with different sizes on the dots. This is best visualized if the dots are transparent:

Example

Create random arrays with 100 values for x-points, y-points, colors and sizes:

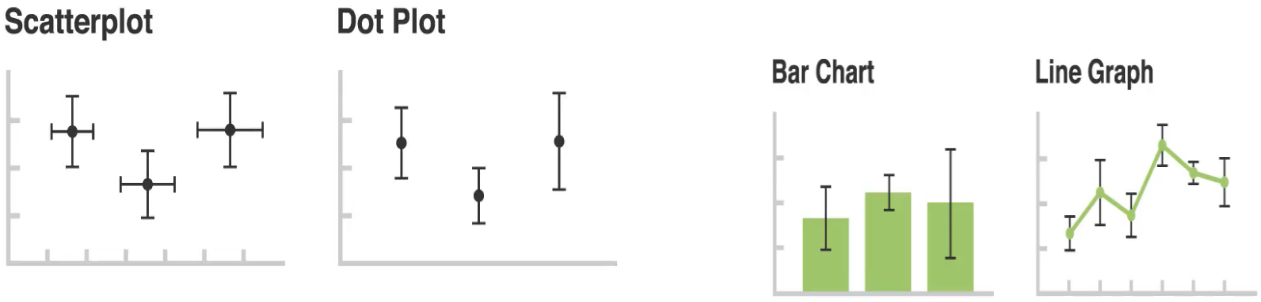
```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.randint(100, size=(100))
y = np.random.randint(100, size=(100))
colors = np.random.randint(100, size=(100))
sizes = 10 * np.random.randint(100, size=(100))
plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='nipy_spectral')
plt.colorbar()
plt.show()
```

Result:

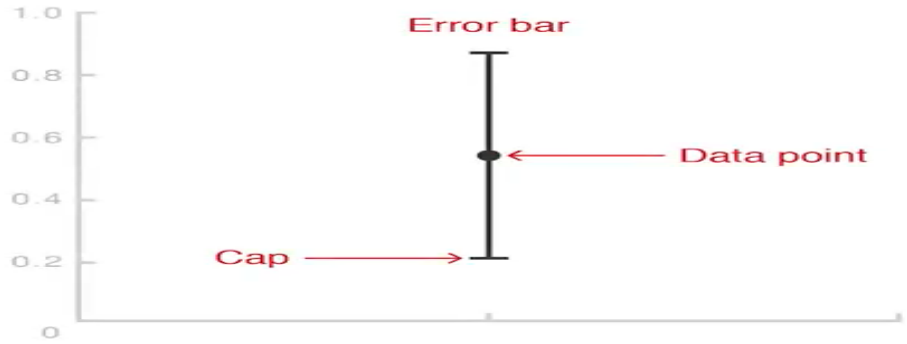


5.4 Visualizing Errors

Error bars function used as graphical enhancement that visualizes the variability of the plotted data on a Cartesian graph. Error bars can be applied to graphs to provide an additional layer of detail on the presented data. As you can see in below graphs.



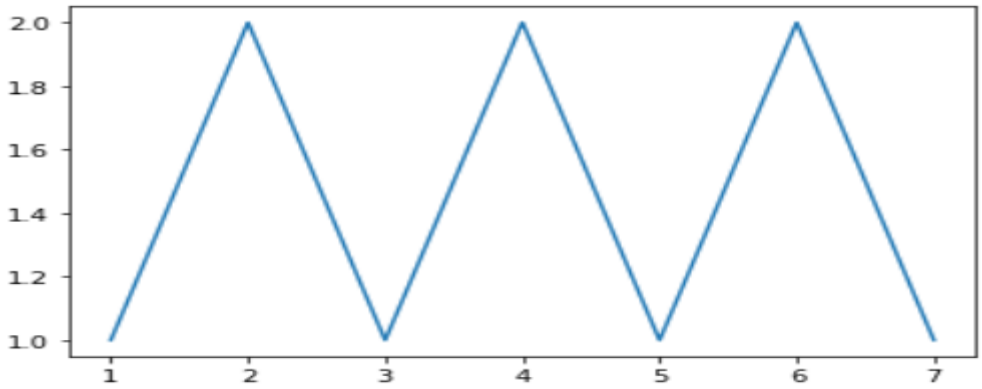
A short error bar shows that values are concentrated signaling that the plotted averaged value is more likely while a long error bar would indicate that the values are more spread out and less reliable. also depending on the type of data. the length of each pair of error bars tends to be of equal length on both sides, however, if the data is skewed then the lengths on each side would be unbalanced.



Creating a Simple Graph.

```
import matplotlib.pyplot as plt
x =[1, 2, 3, 4, 5, 6, 7]
y =[1, 2, 1, 2, 1, 2, 1]
plt.plot(x, y)
```

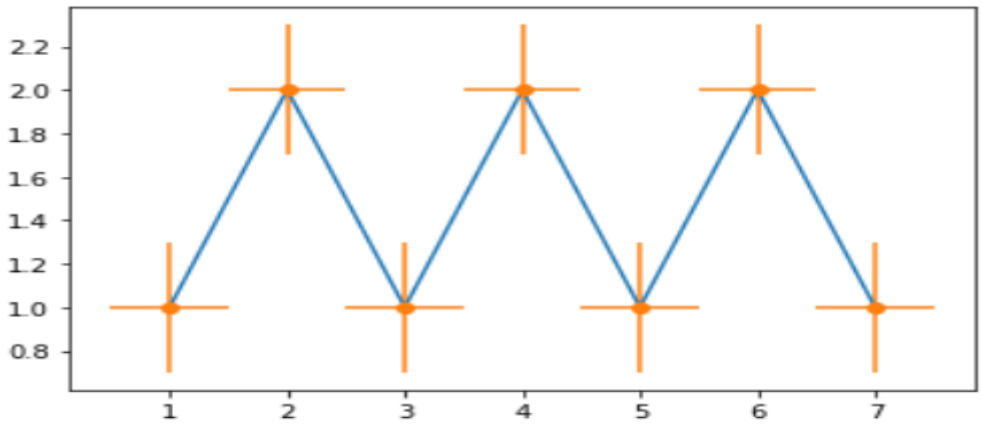
Output:



Adding error in x & y

```
# importing matplotlib
import matplotlib.pyplot as plt
# making a simple plot
x =[1, 2, 3, 4, 5, 6, 7]
y =[1, 2, 1, 2, 1, 2, 1]
# creating error
x_error = 0.5
y_error = 0.3
plt.plot(x, y)
plt.errorbar(x, y,
                    yerr = y_error,
                    xerr = x_error,
                    fmt ='o')
```

Output



5.5 Density and Contour Plots

matplotlib.pyplot.contour

The **matplotlib.pyplot.contour()** are usually useful when **Z = f(X, Y)** i.e Z changes as a function of input X and Y. A contourf() is also available which allows us to draw filled contours.

Syntax: *matplotlib.pyplot.contour([X, Y,] Z, [levels], **kwargs)*

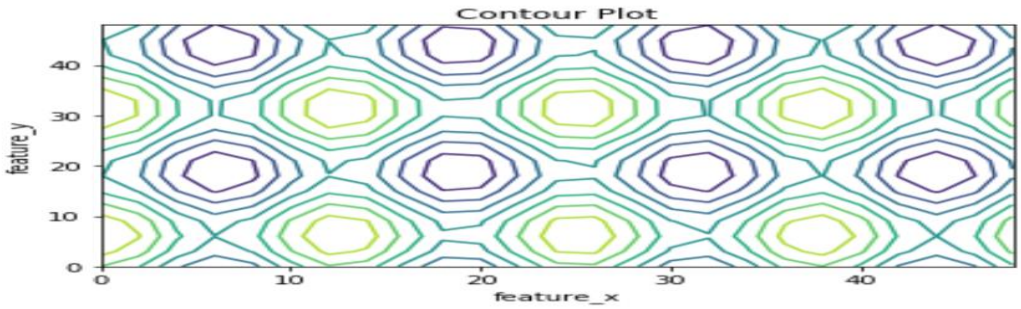
Parameters:

X, Y: *2-D numpy arrays with same shape as Z or 1-D arrays such that len(X)==M and len(Y)==N (where M and N are rows and columns of Z)*

Z: *The height values over which the contour is drawn. Shape is (M, N)*

levels: *Determines the number and positions of the contour lines / regions.*

```
# Implementation of matplotlib function
import matplotlib.pyplot as plt
import numpy as np
feature_x = np.arange(0, 50, 2)
feature_y = np.arange(0, 50, 3)
# Creating 2-D grid of features
[X, Y] = np.meshgrid(feature_x, feature_y)
fig, ax = plt.subplots(1, 1)
Z = np.cos(X / 2) + np.sin(Y / 4)
# plots contour lines
ax.contour(X, Y, Z)
ax.set_title('Contour Plot')
ax.set_xlabel('feature_x')
ax.set_ylabel('feature_y')
plt.show()
```



5.6 Histograms

A histogram is a graph showing *frequency* distributions.

It is a graph showing the number of observations within each given interval

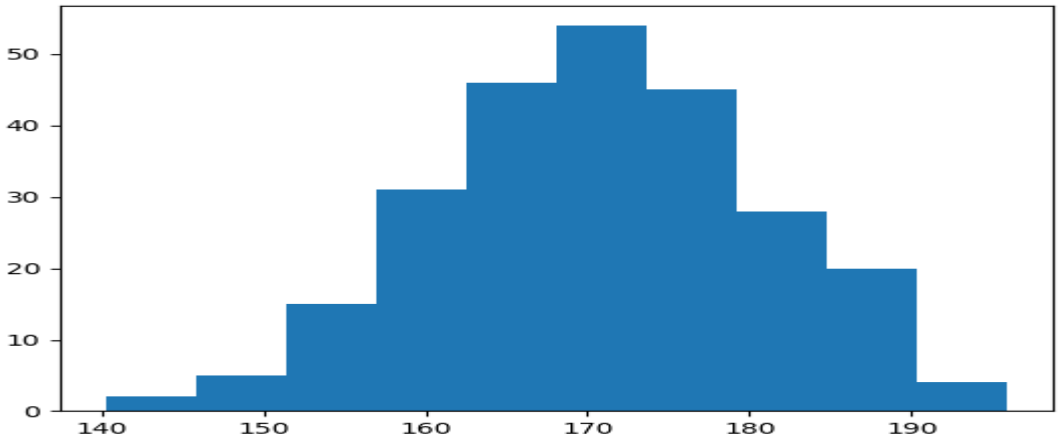
Create Histogram

In Matplotlib, we use the hist() function to create histograms.

The hist() function will use an array of numbers to create a histogram, the array is sent into the function as an argument.

Example
A simple histogram:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.normal(170, 10, 250)
plt.hist(x)
plt.show()
```



Matplotlib Pie Charts

Labels

Add labels to the pie chart with the label parameter.

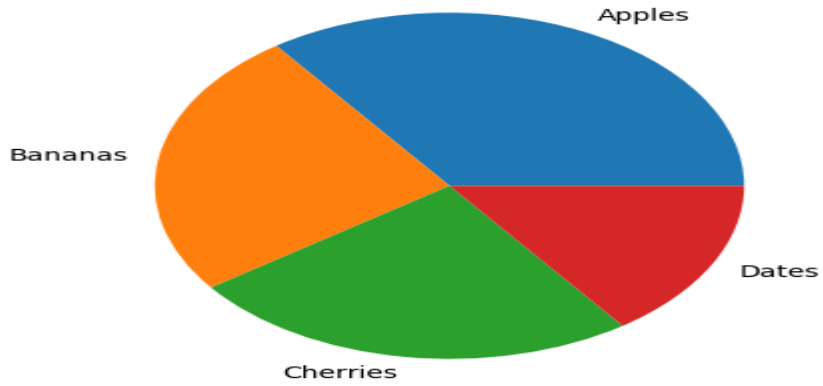
The label parameter must be an array with one label for each wedge:

Example

A simple pie chart:

```
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
plt.pie(y, labels = mylabels)
plt.show()
```

Result:



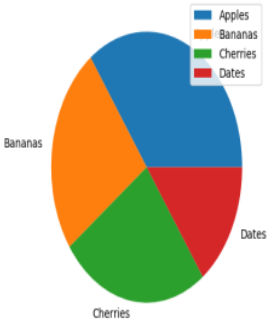
5.7 legends

A legend is an area describing the elements of the graph. In the matplotlib library, there's a function called **legend()** which is used to Place a legend on the axes.

To add a list of explanation for each wedge, use the legend() function:

Example
Add a legend:

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

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
plt.pie(y, labels = mylabels)
plt.legend()
plt.show()
```



Legend With Header

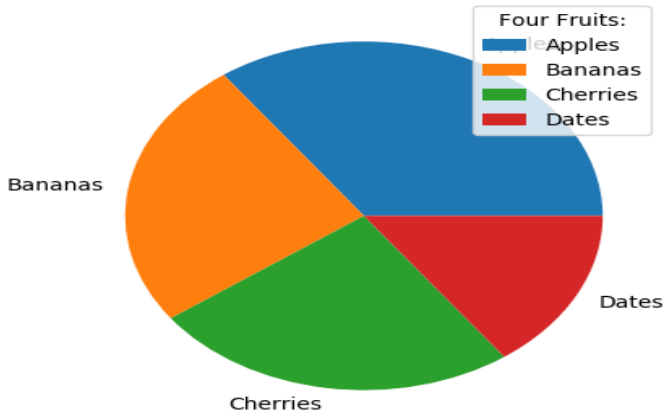
To add a header to the legend, add the title parameter to the legend function.

Example
Add a legend with a header:

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

```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
plt.pie(y, labels = mylabels)
plt.legend(title = "Four Fruits:")
plt.show()
```

OUTPUT:



5.8 Colors

The RGB color model

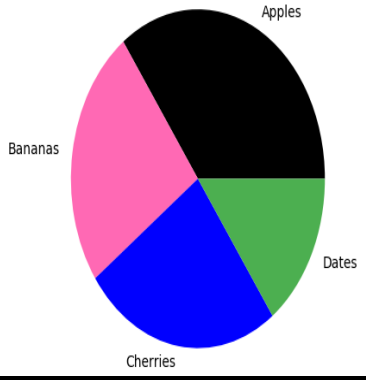
In the RGB color model, any color can be generated by mixing 3 primary colors, namely, Red, Green, and Blue.
In this model, a color can be described by specifying a group of 3 numeric values (typically ranging from 0 to 255),

You can set the color of each wedge with the colors parameter.

The colors parameter, if specified, must be an array with one value for each wedge:

Example
Specify a new color for each wedge:

```
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
mylabels =
["Apples", "Bananas", "Cherries", "Dates"]
mycolors = ["black", "hotpink", "b", "#4CAF50"]
```



```
plt.pie(y, labels = mylabels, colors = mycolors)
plt.show()
```

You can use Hexadecimal color values, any of the 140 supported color names, or one of these shortcuts:

'r' - Red
'g' - Green
'b' - Blue
'c' - Cyan
'm' - Magenta
'y' - Yellow
'k' - Black
'w' - White

5.9 subplots

The subplot() Function

The subplot() function takes three arguments that describes the layout of the figure.

The layout is organized in rows and columns, which are represented by the *first* and *second* argument.

The third argument represents the index of the current plot.

```
plt.subplot(1, 2, 1)
#the figure has 1 row, 2 columns, and this plot is the first plot.
plt.subplot(1, 2, 2)
#the figure has 1 row, 2 columns, and this plot is the second plot.
```

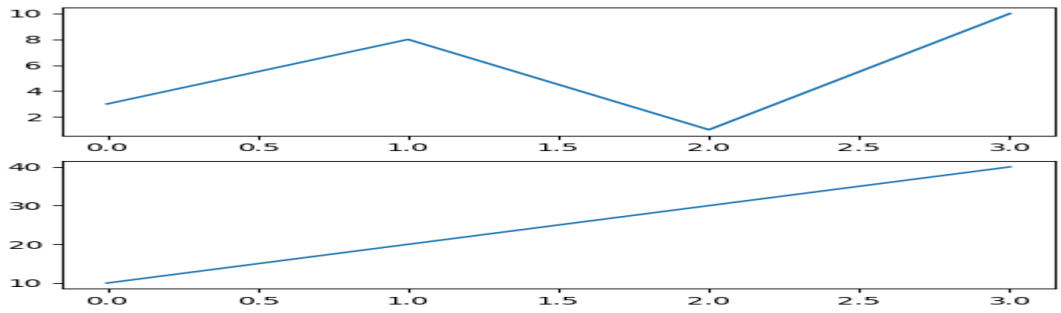
So, if we want a figure with 2 rows an 1 column (meaning that the two plots will be displayed on top of each other instead of side-by-side), we can write the syntax like this:

Example
Draw 2 plots on top of each other:

```
import matplotlib.pyplot as plt
import numpy as np
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 1, 1)
```

```
plt.plot(x,y)
#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 1, 2)
plt.plot(x,y)
plt.show()
```

OUTPUT:



5.10 Text and Annotation

matplotlib.pyplot.annotate() Function

The **annotate() function** in pyplot module of matplotlib library is used to annotate the point xy with text s.

Syntax: angle_spectrum(x, Fs=2, Fc=0, window=mlab.window_hanning, pad_to=None, sides='default', **kwargs)

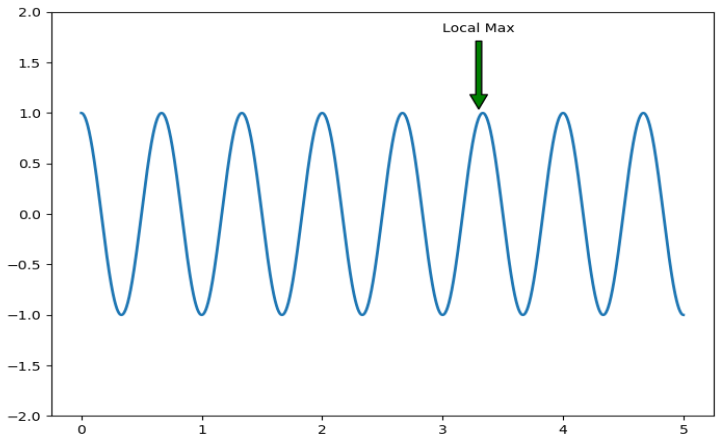
Parameters: This method accept the following parameters that are described below:

- **s:** This parameter is the text of the annotation.
- **xy:** This parameter is the point (x, y) to annotate.
- **xytext:** This parameter is an optional parameter. It is The position (x, y) to place the text at.
- **xycoords:** This parameter is also an optional parameter and contains the string value.
- **textcoords:** This parameter contains the string value.Coordinate system that xytext is given, which may be different than the coordinate system used for xy
- **arrowprops :** This parameter is also an optional parameter and contains dict type.Its default value is None.

- **annotation_clip :** This parameter is also an optional parameter and contains boolean value.Its default value is None which behaves as True.

```
# Implementation of matplotlib.pyplot.annotate()

# function

import matplotlib.pyplot as plt

import numpy as np

fig, geeeks = plt.subplots()

t = np.arange(0.0, 5.0, 0.001)

s = np.cos(3 * np.pi * t)

line = geeeks.plot(t, s, lw = 2)

# Annotation

geeeks.annotate('Local Max', xy =(3.3, 1),

                            xytext =(3, 1.8),

                arrowprops = dict(facecolor ='green', shrink = 0.05),)

geeeks.set_ylim(-2, 2)

# Plot the Annotation in the graph

plt.show()
```

OUTPUT

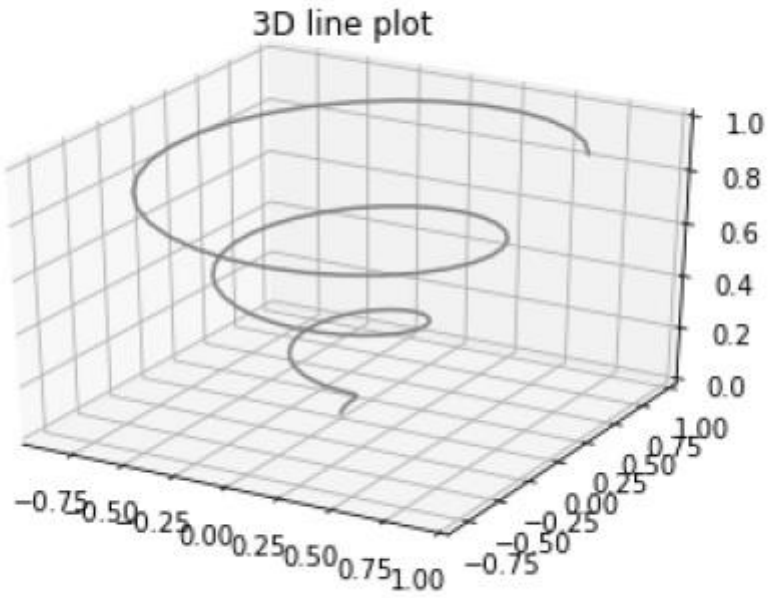


5.11 Three Dimensional Plotting

A three-dimensional axes can be created by passing the keyword projection='3d' to any of the normal axes creation routines.

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes(projection='3d')
z = np.linspace(0, 1, 100)
x = z * np.sin(20 * z)
y = z * np.cos(20 * z)
ax.plot3D(x, y, z, 'gray')
ax.set_title('3D line plot')
plt.show()
```

We can now plot a variety of three-dimensional plot types. The most basic three-dimensional plot is a **3D line plot** created from sets of (x, y, z) triples. This can be created using the ax.plot3D function.



5.12 Geographic Data with Basemap

One common type of visualization in data science is that of geographic data. Matplotlib's main tool for this type of visualization is the Basemap toolkit, which is one of several Matplotlib toolkits which lives under the mpl_toolkits namespace. Admittedly, Basemap feels a bit clunky to use, and often even simple visualizations take much longer to render than you might hope. More modern solutions such as leaflet or the Google Maps API may be a better choice for more intensive map

visualizations. Still, Basemap is a useful tool for Python users to have in their virtual toolbelts. In this section, we'll show several examples of the type of map visualization that is possible with this toolkit.

Installation of Basemap is straightforward; if you're using conda you can type this and the package will be downloaded:

```
$ conda install basemap
```

We add just a single new import to our standard boilerplate:

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
```

Once you have the Basemap toolkit installed and imported, geographic plots are just a few lines away (the graphics in the following also requires the `PIL` package in Python 2, or the `pillow` package in Python 3):

In [2]:
```
plt.figure(figsize=(8, 8))
m = Basemap(projection='ortho', resolution=None, lat_0=50, lon_0=-100)
m.bluemarble(scale=0.5);
```

The meaning of the arguments to `Basemap` will be discussed momentarily.

5.13 Seaborn

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It is built on the top of matplotlib library and also closely integrated into the data structures from pandas.

Installation

For python environment :

pip install seaborn

For conda environment :

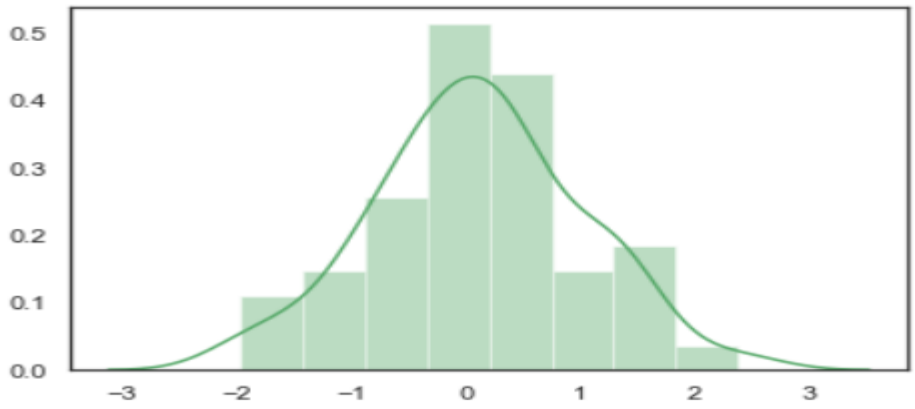
conda install seaborn

Let's create Some basic plots using seaborn:

- Python3

```
# Importing libraries
import numpy as np
import seaborn as sns
# Selecting style as white,
# dark, whitegrid, darkgrid
# or ticks
sns.set( style = "white" )
# Generate a random univariate
# dataset
rs = np.random.RandomState( 10 )
d = rs.normal( size = 50 )
# Plot a simple histogram and kde
# with binsize determined automatically
sns.distplot(d, kde = True, color = "g")
```

Output:



Seaborn: statistical data visualization

Seaborn helps to visualize the statistical relationships, To understand how variables in a dataset are related to one another and how that relationship is dependent on

other variables, we perform statistical analysis. This Statistical analysis helps to visualize the trends and identify various patterns in the dataset.

These are the plot will help to visualize:

- Line Plot
- Scatter Plot
- Box plot
- Point plot
- Count plot
- Violin plot
- Swarm plot
- Bar plot
- KDE Plot

Line plot:

Lineplot Is the most popular plot to draw a relationship between x and y with the possibility of several semantic groupings.

Syntax : *sns.lineplot(x=None, y=None)*

Parameters:

x, y: *Input data variables; must be numeric. Can pass data directly or reference columns in data.*

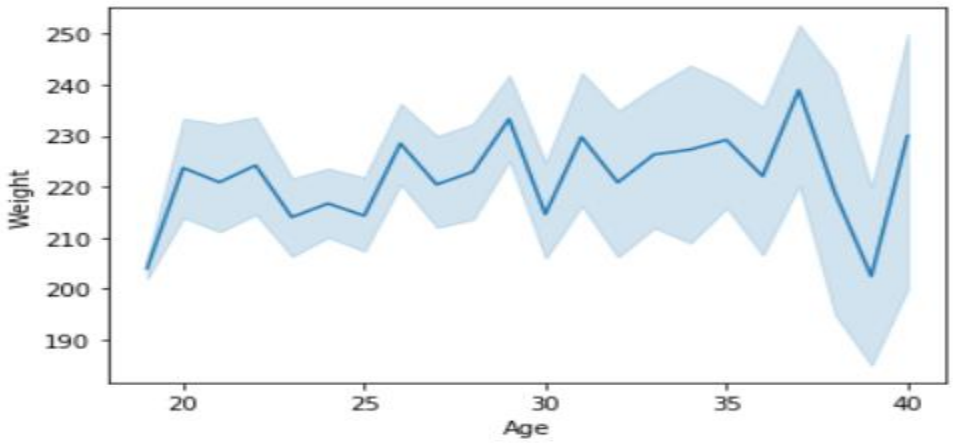
Let's visualize the data with a line plot and pandas:

Example 1:

- Python3

```
# import module
import seaborn as sns
import pandas
# loading csv
data = pandas.read_csv("nba.csv")
# plotting lineplot
sns.lineplot( data['Age'], data['Weight'])
```

Output:

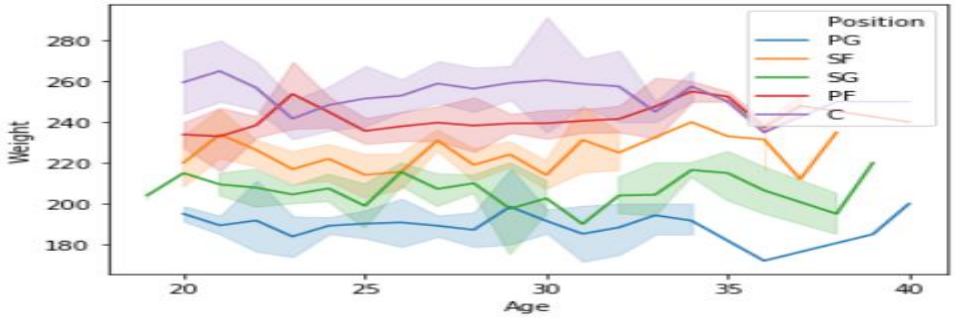


Example 2: Use the hue parameter for plotting the graph.

- Python3

```
# import module
import seaborn as sns
import pandas
 # read the csv data
data = pandas.read_csv("nba.csv")
 # plot
sns.lineplot(data['Age'],data['Weight'], hue =data["Position"])
```

Output:



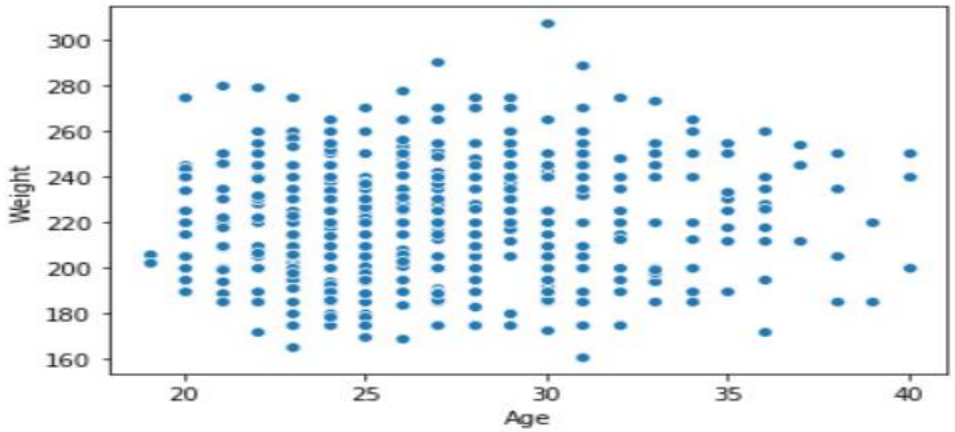
Scatter Plot:

Scatterplot Can be used with several semantic groupings which can help to understand well in a graph against continuous/categorical data. It can draw a two-dimensional graph.
Syntax: *seaborn.scatterplot(x=None, y=None)*
Parameters:
x, y: *Input data variables that should be numeric.*
Returns: *This method returns the Axes object with the plot drawn onto it.*
Let's visualize the data with a scatter plot and pandas:
Example 1:

-

```
# import module
import seaborn
import pandas
 # load csv
data = pandas.read_csv("nba.csv")
 # plotting
seaborn.scatterplot(data['Age'],data['Weight'])
```

Output:

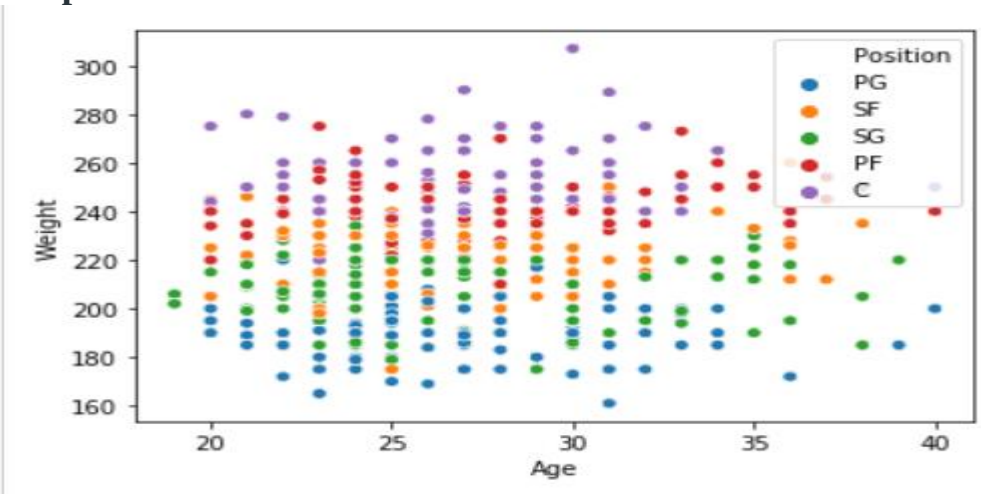


Example 2: Use the hue parameter for plotting the graph.

- Python3

```
import seaborn

import pandas

data = pandas.read_csv("nba.csv")

 seaborn.scatterplot( data['Age'], data['Weight'], hue =data["Position"])
```

Output:



Box plot:

A box plot (or box-and-whisker plot) s is the visual representation of the depicting groups of numerical data through their quartiles against continuous/categorical data. A box plot consists of 5 things.

- Minimum
- First Quartile or 25%
- Median (Second Quartile) or 50%
- Third Quartile or 75%
- Maximum

Syntax:
seaborn.boxplot(x=None, y=None, hue=None, data=None)

Parameters:
- ***x, y, hue:*** *Inputs for plotting long-form data.*
- ***data:*** *Dataset for plotting. If x and y are absent, this is interpreted as wide-form.*

Returns: *It returns the Axes object with the plot drawn onto it.*

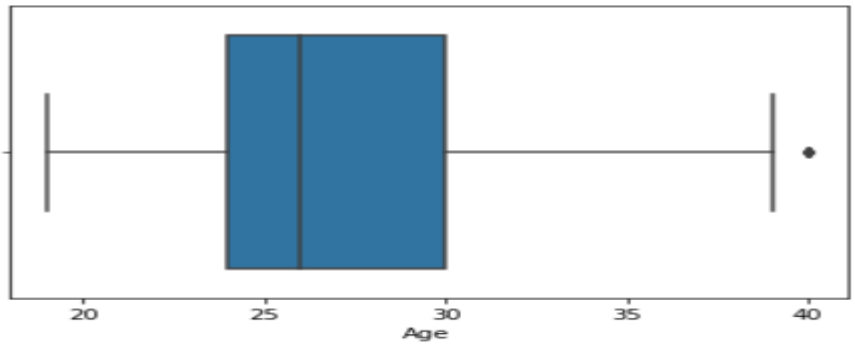
Draw the box plot with Pandas:

Example 1:

- Python3

```
# import module
import seaborn as sns
import pandas
# read csv and plotting
data = pandas.read_csv( "nba.csv" )
sns.boxplot( data['Age'] )
```

Output:

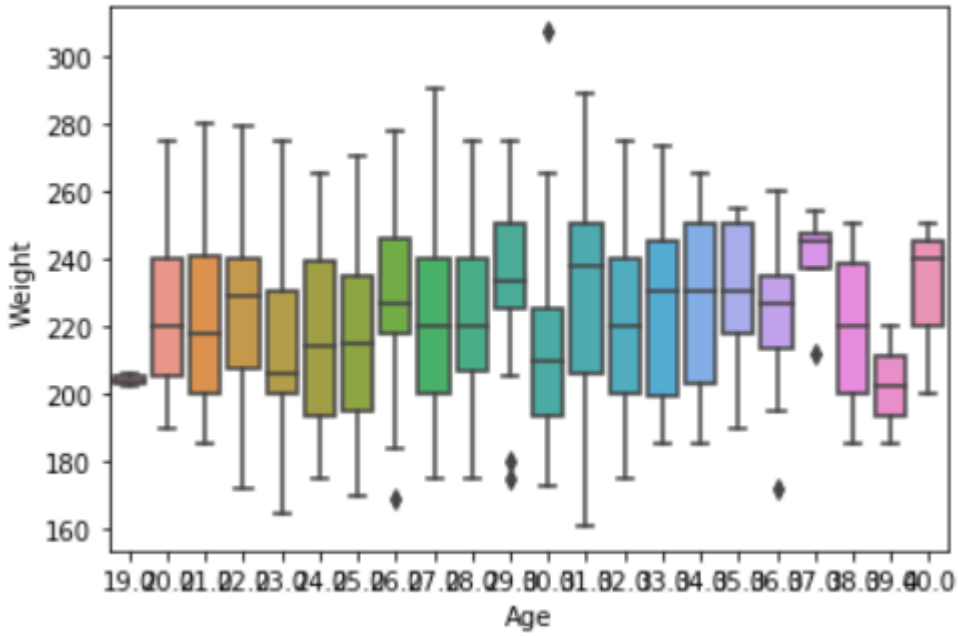


Example 2:

- Python3

```
# import module
import seaborn as sns
import pandas
 # read csv and plotting
data = pandas.read_csv( "nba.csv" )
sns.boxplot( data['Age'], data['Weight'])
```

Output:



Violin Plot:

A violin plot is similar to a boxplot. It shows several quantitative data across one or more categorical variables such that those distributions can be compared.

Syntax: *seaborn.violinplot(x=None, y=None, hue=None, data=None)*
Parameters:

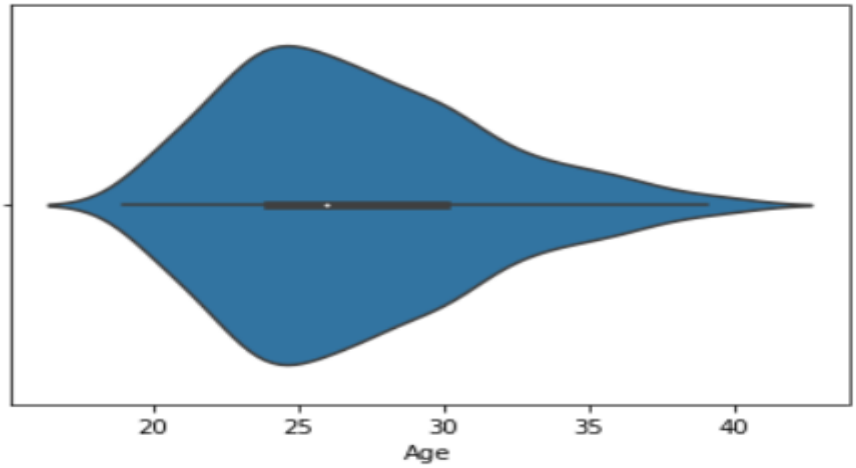
- ***x, y, hue:*** *Inputs for plotting long-form data.*
- ***data:*** *Dataset for plotting.*

Draw the violin plot with Pandas:
Example 1:

- Python3

```
# import module
import seaborn as sns
import pandas
 # read csv and plot
data = pandas.read_csv("nba.csv")
sns.violinplot(data['Age'])
```

Output:

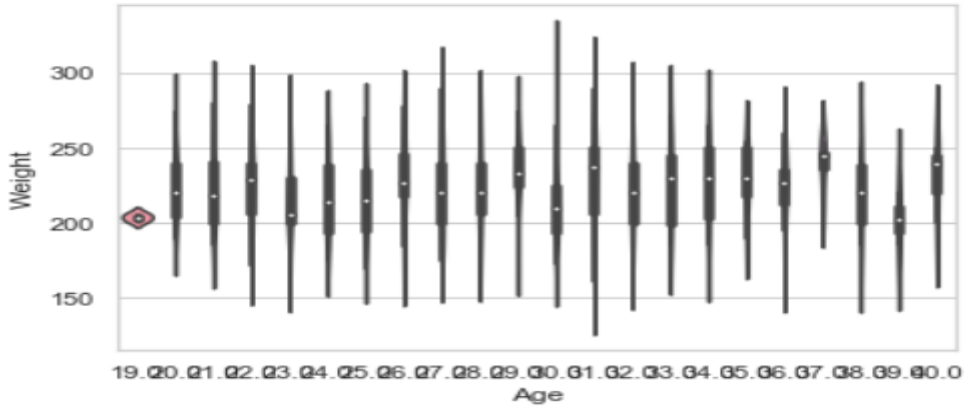


Example 2:

- Python3

```
# import module
import seaborn
 seaborn.set(style = 'whitegrid')
 # read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.violinplot(x ="Age", y ="Weight",data = data)
```

Output:



Swarm plot:

A swarm plot is similar to a strip plot, We can draw a swarm plot with non-overlapping points against categorical data.

Syntax: *seaborn.swarmplot(x=None, y=None, hue=None, data=None)*

Parameters:

- ***x, y, hue:*** *Inputs for plotting long-form data.*

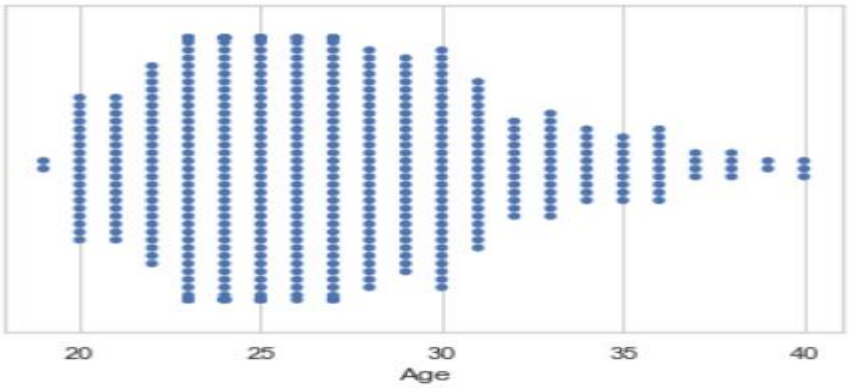
- ***data:*** *Dataset for plotting.*

Draw the swarm plot with Pandas:
Example 1:

- Python3

```
# import module
import seaborn
 seaborn.set(style = 'whitegrid')
 # read csv and plot
data = pandas.read_csv( "nba.csv" )
seaborn.swarmplot(x = data["Age"])
```

Output:

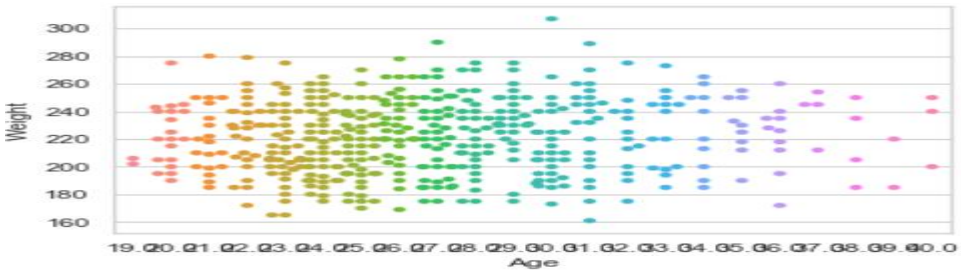


Example 2:

- Python3

```
# import module
import seaborn
 seaborn.set(style = 'whitegrid')
 # read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.swarmplot(x ="Age", y ="Weight",data = data)
```

Output:



Bar plot:

Barplot represents an estimate of central tendency for a numeric variable with the height of each rectangle and provides some indication of the uncertainty around that estimate using error bars.

Syntax : *seaborn.barplot(x=None, y=None, hue=None, data=None)*

Parameters :

- ***x, y :*** *This parameter take names of variables in data or vector data, Inputs for plotting long-form data.*
- ***hue :*** *(optional) This parameter take column name for colour encoding.*
- ***data :*** *(optional) This parameter take DataFrame, array, or list of arrays, Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.*

Returns : *Returns the Axes object with the plot drawn onto it.*

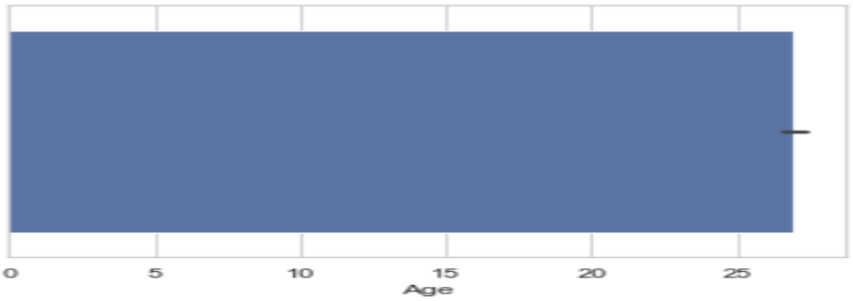
Draw the bar plot with Pandas:

Example 1:

- Python3

```
# import module
import seaborn
 seaborn.set(style = 'whitegrid')
 # read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.barplot(x =data["Age"])
```

Output:

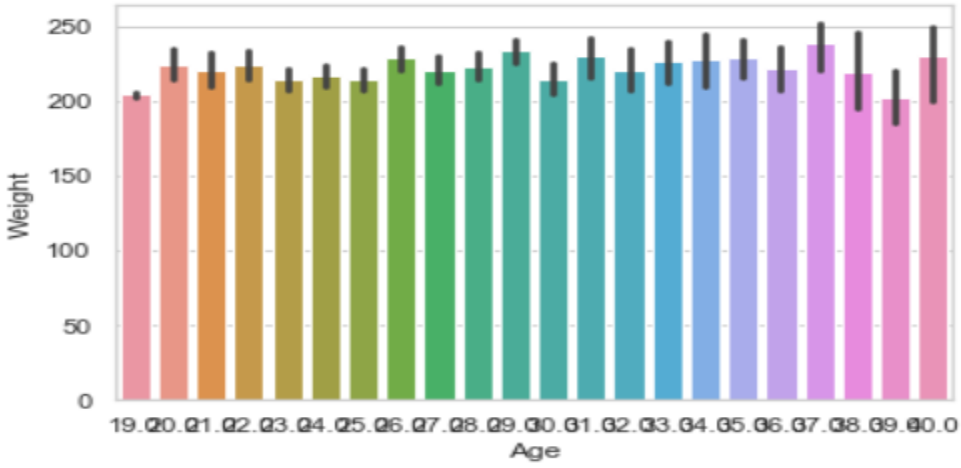


Example 2:

- Python3

```
# import module
import seaborn
 seaborn.set(style = 'whitegrid')
 # read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.barplot(x ="Age", y ="Weight", data = data)
```

Output:



Point plot:

Point plot used to show point estimates and confidence intervals using scatter plot glyphs. A point plot represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars.

Syntax: *seaborn.pointplot(x=None, y=None, hue=None, data=None)*

Parameters:

- ***x, y:*** *Inputs for plotting long-form data.*
- ***hue:*** *(optional) column name for color encoding.*
- ***data:*** *dataframe as a Dataset for plotting.*

Return: *The Axes object with the plot drawn onto it.*

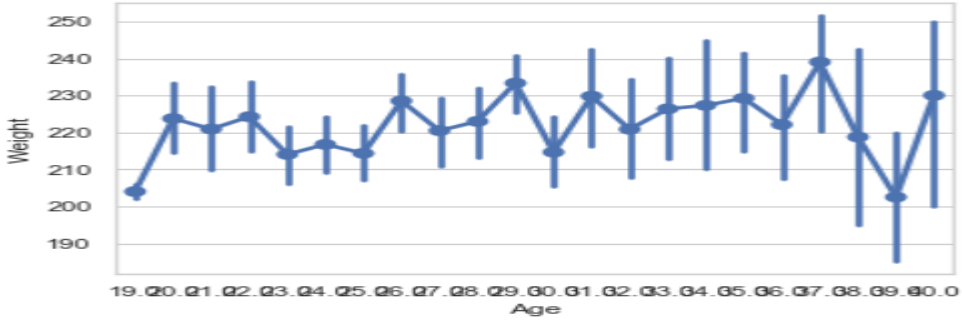
Draw the point plot with Pandas:

Example:

- Python3

```
# import module
import seaborn
 seaborn.set(style = 'whitegrid')
 # read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.pointplot(x = "Age", y = "Weight", data = data)
```

Output:



Count plot:

Count plot used to Show the counts of observations in each categorical bin using bars.

Syntax : *seaborn.countplot(x=None, y=None, hue=None, data=None)*
Parameters :

- ***x, y:*** *This parameter take names of variables in data or vector data, optional, Inputs for plotting long-form data.*
- ***hue :*** *(optional) This parameter take column name for color encoding.*
- ***data :*** *(optional) This parameter take DataFrame, array, or list of arrays, Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise, it is expected to be long-form.*

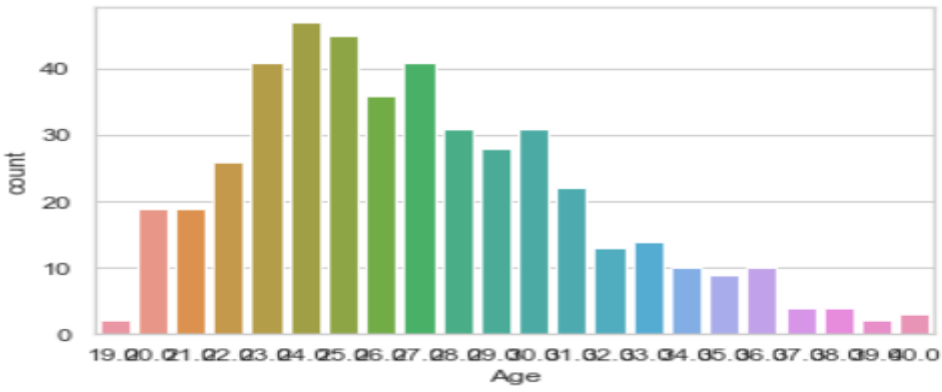
Returns: *Returns the Axes object with the plot drawn onto it.*

Draw the count plot with Pandas:
Example:

- Python3

```
# import module
import seaborn
 seaborn.set(style = 'whitegrid')
 # read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.countplot(data["Age"])
```

Output:



KDE Plot:

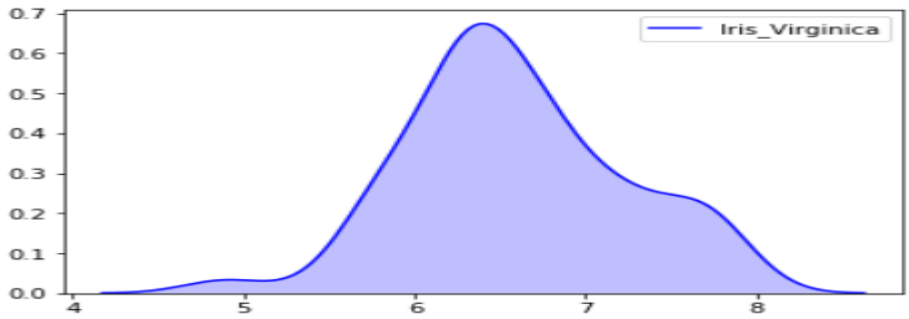
KDE Plot described as **Kernel Density Estimate** is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization.

Syntax: *seaborn.kdeplot(x=None, *, y=None, vertical=False, palette=None, **kwargs)*
Parameters:
x, y : *vectors or keys in data*
vertical : *boolean (True or False)*
data : *pandas.DataFrame, numpy.ndarray, mapping, or sequence*
Draw the KDE plot with Pandas:
Example 1:

- Python3

```
# importing the required libraries
from sklearn import datasets
import pandas as pd
import seaborn as sns

# Setting up the Data Frame
iris = datasets.load_iris()

iris_df = pd.DataFrame(iris.data, columns=['Sepal_Length',
                   'Sepal_Width', 'Patal_Length', 'Petal_Width'])

iris_df['Target'] = iris.target

iris_df['Target'].replace([0], 'Iris_Setosa', inplace=True)
iris_df['Target'].replace([1], 'Iris_Vercicolor', inplace=True)
iris_df['Target'].replace([2], 'Iris_Virginica', inplace=True)

# Plotting the KDE Plot
sns.kdeplot(iris_df.loc[(iris_df['Target'] =='Iris_Virginica'),
          'Sepal_Length'], color = 'b', shade = True, Label ='Iris_Virginica')
```

Output:

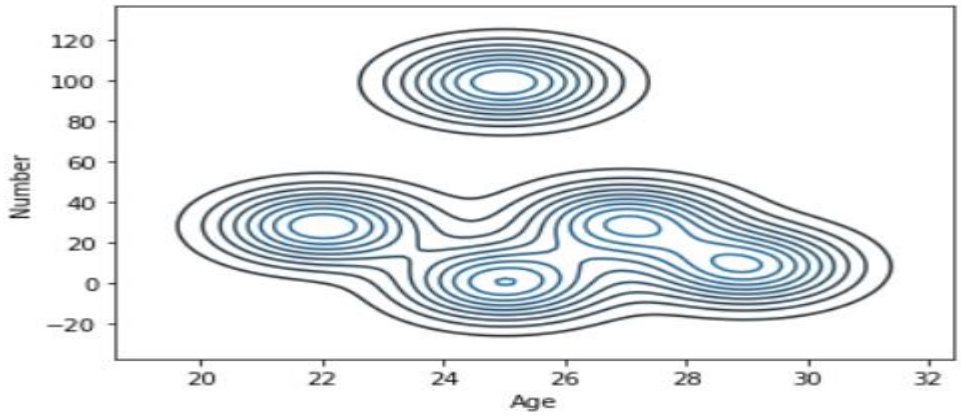


Example 2:

- Python3

```
# import module
import seaborn as sns
import pandas
 # read top 5 column
data = pandas.read_csv("nba.csv").head()
 sns.kdeplot( data['Age'], data['Number'])
```

Output:



Bivariate and Univariate data using seaborn and pandas:

Before starting let's have a small intro of bivariate and univariate data:

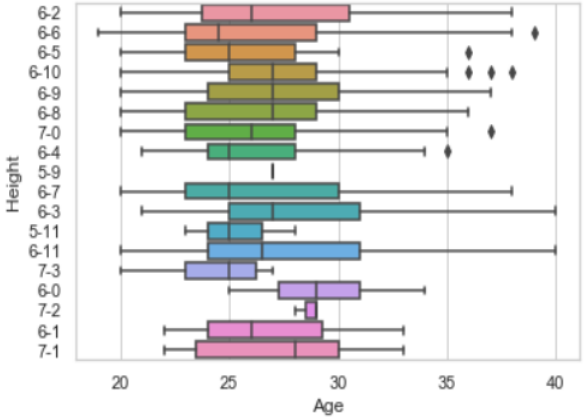
Bivariate data: This type of data involves **two different variables**. The analysis of this type of data deals with causes and relationships and the analysis is done to find out the relationship between the two variables.

Univariate data: This type of data consists of **only one variable**. The analysis of univariate data is thus the simplest form of analysis since the information deals with only one quantity that changes. It does not deal with causes or relationships and the main purpose of the analysis is to describe the data and find patterns that exist within it.

Let's see an example of Bivariate data :

Example 1: Using the box plot.

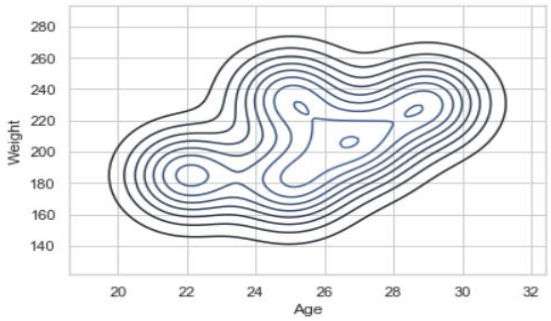
```
# import module
import seaborn as sns
import pandas
 # read csv and plotting
data = pandas.read_csv( "nba.csv" )
sns.boxplot( data['Age'], data['Height'])
```



Example 2: using KDE plot.

- Python3

```
# import module
import seaborn as sns
import pandas
 # read top 5 column
data = pandas.read_csv("nba.csv").head()
 sns.kdeplot( data['Age'], data['Weight'])
```



Let's see an example of univariate data distribution:
Example: Using the dist plot

- Python3

```
# import module
import seaborn as sns
import pandas
 # read top 5 column
data =
pandas.read_csv("nba.csv").head()
 sns.distplot( data['Age'])
```