VERTICALS

CCS346 EXPLORATORY DATA ANALYSIS L T P C
2 0 2 3

COURSE OBJECTIVES:

- To outline an overview of exploratory data analysis.
- To implement data visualization using Matplotlib.
- To perform univariate data exploration and analysis.
- To apply bivariate data exploration and analysis.
- To use Data exploration and visualization techniques for multivariate and time series data.

UNIT I EXPLORATORY DATA ANALYSIS 6

EDA fundamentals – Understanding data science – Significance of EDA – Making sense of data – Comparing EDA with classical and Bayesian analysis – Software tools for EDA - Visual Aids for EDA- Data transformation techniques-merging database, reshaping and pivoting, Transformation techniques.

UNIT II EDA USING PYTHON 6

Data Manipulation using Pandas – Pandas Objects – Data Indexing and Selection – Operating on Data – Handling Missing Data – Hierarchical Indexing – Combining datasets – Concat, Append, Merge and Join – Aggregation and grouping – Pivot Tables – Vectorized String Operations.

UNIT III UNIVARIATE ANALYSIS 6

Introduction to Single variable: Distribution Variables - Numerical Summaries of Level and Spread - Scaling and Standardizing – Inequality.

UNIT IV BIVARIATE ANALYSIS 6

Relationships between Two Variables - Percentage Tables - Analysing Contingency Tables - Handling Several Batches - Scatterplots and Resistant Lines.

UNIT V MULTIVARIATE AND TIME SERIES ANALYSIS 6

Introducing a Third Variable - Causal Explanations - Three-Variable Contingency Tables and Beyond – Fundamentals of TSA – Characteristics of time series data – Data Cleaning – Time-based indexing – Visualizing – Grouping – Resampling.

30 PERIODS

PRACTICAL EXERCISES: 30 PERIODS

1. Install the data Analysis and Visualization tool: R/ Python /Tableau Public/ Power BI.
2. Perform exploratory data analysis (EDA) with datasets like email data set. Export all your emails as a dataset, import them inside a pandas data frame, visualize them and get different insights from the data.
3. Working with Numpy arrays, Pandas data frames , Basic plots using Matplotlib.
4. Explore various variable and row filters in R for cleaning data. Apply various plot features in R on sample data sets and visualize.
5. Perform Time Series Analysis and apply the various visualization techniques.
6. Perform Data Analysis and representation on a Map using various Map data sets with Mouse Rollover effect, user interaction, etc..

TABLE OF CONTENTS

UNIT I

UNIT II

UNIT III

UNIT IV

UNIT V

UNIT I

1 Exploratory Data Analysis

Syllabus

EDA fundamentals - Understanding data science - Significance of EDA – Making sense of data - Comparing EDA with classical and Bayesian analysis - Software tools for EDA - Visual Aids for EDA - Data transformation techniques-merging database, reshaping and pivoting, Transformation techniques - Grouping Datasets - data aggregation - Pivot tables and cross-tabulations.

Contents

1.1 Data and EDA Fundamentals

1.1.1 Data and Information

- Data (Singular Datum), is collection of different facts, figures, objects, symbols and events that are capable of providing informative pieces usually formatted in a particular manner. Data is a collection of facts, such as numbers, words, measurements, observations or just descriptions of things.
- Data can be qualitative or quantitative. **Qualitative data** is descriptive information (it describes something). **Quantitative data** is numerical information (numbers).
- **Quantitative data** can be discrete or continuous. **Discrete data** can only take certain values (like whole numbers). **Continuous data** can take any value (within a range). **Discrete data** is counted, **continuous data** is measured.

Examples of Data

Qualitative

- My friend's favorite holiday destination.
- The most common names in India.
- How people describe the smell of a new deodorant.

Quantitative

- Height (Continuous).
- Weight (Continuous).
- Leaves on a tree (Discrete).
- Customers in a shop (Discrete).

Information

- Information is defined as classified or organized data that has some meaningful value for the user. Information is also the processed data used to make decisions and take action. Processed data must meet the following criteria for it to be of any significant use in decision-making :
  - **Accuracy** - The information must be accurate.
  - **Completeness** - The information must be complete.
  - **Timeliness** - The information must be available when it's needed.

1.1.2 Data Collection Methods

- Data collection is defined as a method of **collecting, analyzing data** for the purpose of validation and research using some techniques. Data collection is done to analyze a problem and learn about its outcome and future trends. When there is a need to arrive at a solution for a question, data collection methods help to make assumptions about the result in the future.
- Data collection methods can be classified as,
  - **Primary :** This is original, first-hand data collected by the data researchers. This process is the initial information gathering step, performed before anyone carries out any further or related research. Primary data results are highly accurate provided the researcher collects the information. However, there is a downside, as first-hand research is potentially time-consuming and expensive.
  - **Secondary :** Secondary data is second-hand data collected by other parties and already having undergone statistical analysis. This data is either information that the researcher has tasked other people to collect or information the researcher has looked up. Simply put, it is second-hand information. Although it is easier and cheaper to obtain than primary information, secondary information raises concerns regarding accuracy and authenticity. Quantitative data makes up a majority of secondary data.

Types of data collection :

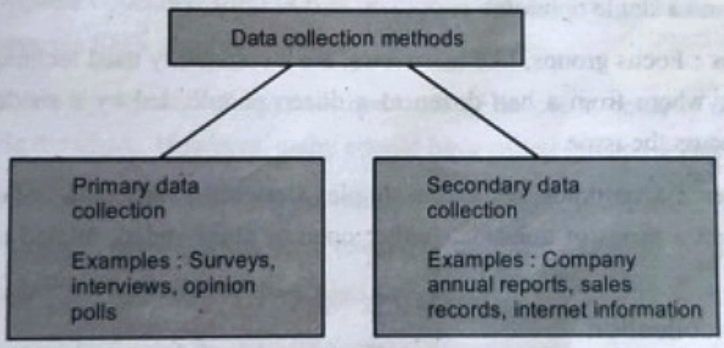


Fig. 1.1.1 : Data collection methods

- Below are widely used data collection methods,

1. Interviews
2. Questionnaires and surveys
3. Observations
4. Documents and records
5. Focus groups
6. Oral histories.

- Some of the above methods are quantitative, dealing with something that can be counted. Others are qualitative, meaning that they consider factors other than numerical values. In general, questionnaires, surveys and documents and records are quantitative, while interviews, focus groups, observations and oral histories are qualitative. There can also be crossover between the two methods.

Primary data collection

- **Interviews :** The researcher asks questions of a large sampling of people, either by direct interviews or means of mass communication such as by phone or mail. This method is by far the most common means of data gathering.
- **Projective Technique :** Projective data gathering is an indirect interview, used when potential respondents know why they are being asked questions and hesitate to answer. For instance, someone may be reluctant to answer questions about their phone service if a cell phone carrier representative poses the questions. With projective data gathering, the interviewees get an incomplete question and they must fill in the rest, using their opinions, feelings and attitudes.
- **Delphi Technique :** The Oracle at Delphi, according to Greek mythology, was the high priestess of Apollo's temple, who gave advice, prophecies and counsel. In the realm of data collection, researchers use the Delphi technique by gathering information from a panel of experts. Each expert answers questions in their field of specialty and the replies are consolidated into a single opinion.
- **Focus Groups :** Focus groups, like interviews, are a commonly used technique. The group consists of anywhere from a half-dozen to a dozen people, led by a moderator, brought together to discuss the issue.
- **Questionnaires :** Questionnaires are a simple, straightforward data collection method. Respondents get a series of questions, either open or close-ended, related to the matter at hand.

Secondary data collection

- Unlike primary data collection, there are no specific collection methods. Instead, since the information has already been collected, the researcher consults various data sources, such as :
  - Financial Statements
  - Sales Reports
  - Retailer / Distributor / Deal Feedback

- Customer Personal Information (e.g., name, address, age, contact info)
- Business Journals
- Government Records (e.g., census, tax records, Social Security info)
- Trade/Business Magazines
- The Internet.

Data collection tools : Below are widely used tools for data collection.

- **Word Association :** The researcher gives the respondent a set of words and asks them what comes to mind when they hear each word.
- **Sentence Completion :** Researchers use sentence completion to understand what kind of ideas the respondent has. This tool involves giving an incomplete sentence and seeing how the interviewee finishes it.
- **Role-Playing :** Respondents are presented with an imaginary situation and asked how they would act or react if it was real.
- **In-Person Surveys :** The researcher asks questions in person.
- **Online / Web Surveys :** These surveys are easy to accomplish, but some users may be unwilling to answer truthfully, if at all.
- **Mobile Surveys :** These surveys take advantage of the increasing proliferation of mobile technology. Mobile collection surveys rely on mobile devices like tablets or smartphones to conduct surveys via SMS or mobile apps.
- **Phone Surveys :** No researcher can call thousands of people at once, so they need a third party to handle the chore. However, many people have called screening and won't answer.
- **Observation :** Sometimes, the simplest method is the best. Researchers who make direct observations collect data quickly and easily, with little intrusion or third-party bias. Naturally, it is only effective in small-scale situations.

1.1.3 Common Issues / Problems in Data

Inconsistent data

- When working with various data sources, it is conceivable that the same information will have discrepancies between sources. The differences could be in formats, units or occasionally spellings. The introduction of inconsistent data might also occur during firm mergers or relocations. Inconsistencies in data have a tendency to accumulate and reduce

the value of data if they are not continually resolved. Organizations that have heavily focused on data consistency do so because they only want reliable data to support their analytics.

Data downtime

- Data is the driving force behind the decisions and operations of data-driven businesses. However, there may be brief periods when their data is unreliable or not prepared. Customer complaints and subpar analytical outcomes are only two ways that this data unavailability can have a significant impact on businesses. A data engineer spends about 80 % of their time updating, maintaining and guaranteeing the integrity of the data pipeline. In order to ask the next business question, there is a high marginal cost due to the lengthy operational lead time from data capture to insight.
- Schema modifications and migration problems are just two examples of the causes of data downtime. Data pipelines can be difficult due to their size and complexity. Data downtime must be continuously monitored and it must be reduced through automation.

Ambiguous data

- Even with thorough oversight, some errors can still occur in massive databases or data lakes. For data streaming at a fast speed, the issue becomes more overwhelming. Spelling mistakes can go unnoticed, formatting difficulties can occur and column heads might be deceptive. This unclear data might cause a number of problems for reporting and analytics.

Duplicate data

- Streaming data, local databases and cloud data lakes are just a few of the sources of data that modern enterprises must contend with. They might also have application and system silos. These sources are likely to duplicate and overlap each other quite a bit. For instance, duplicate contact information has a substantial impact on customer experience. If certain prospects are ignored while others are engaged repeatedly, marketing campaigns suffer. The likelihood of biased analytical outcomes increases when duplicate data are present. It can also result in ML models with biased training data.

Too much data

- While the data-driven analytics and its advantages are prominent, a data quality problem with excessive data exists. There is a risk of getting lost in an abundance of data when searching for information pertinent to analytical efforts. Data scientists, data analysts and business users devote 80 % of their work to finding and organizing the appropriate data. With an increase in data volume, other problems with data quality become more serious, particularly when dealing with streaming data and big files or databases.

Inaccurate data

- For highly regulated businesses like healthcare, data accuracy is crucial. Given the current experience, it is more important than ever to increase the data quality for COVID-19 and later pandemics. Inaccurate information does not provide a true picture of the situation and cannot be used to plan the best course of action. Personalized customer experiences and marketing strategies underperform if the customer data is inaccurate.
- Data inaccuracies can be attributed to a number of things, including data degradation, human mistake and data drift. Worldwide data decay occurs at a rate of about 3 % per month, which is quite concerning. Data integrity can be compromised while being transferred between different systems and data quality might deteriorate with time.

Hidden data

- The majority of businesses only utilize a portion of their data, with the remainder sometimes being lost in data silos or discarded in data graveyards. For instance, the customer service team might not receive client data from sales, missing an opportunity to build more precise and comprehensive customer profiles. Missing out on possibilities to develop novel products, enhance services and streamline procedures is caused by hidden data.

Finding relevant data

- Finding relevant data is not so easy. There are several factors that are to be considered to consider while trying to find relevant data, which include -
  - Relevant domain
  - Relevant demographics
  - Relevant time period and so many more factors that one needs to consider while trying to find relevant data.
- Data that is not relevant to the study in any of the factors render it obsolete and one cannot effectively proceed with its analysis. This could lead to incomplete research or analysis, re-collecting data again and again or shutting down the study.

Deciding the data to collect

- Determining what data to collect is one of the most important factors while collecting data and should be one of the first factors while collecting data. One must choose the subjects the data will cover, the sources used to gather it and the required quantity of information. The responses to these queries will depend on the aims or what is expected to achieve utilizing the data. As an illustration, one may choose to gather information on the categories of articles that website visitors between the ages of 20 and 50 most frequently access. One can also decide to compile data on the typical age of all the clients who made a purchase from the business over the previous month.

Dealing with big data

- Big data refers to exceedingly massive data sets with more intricate and diversified structures. These traits typically result in increased challenges while storing, analyzing and using additional methods of extracting results. Big data refers especially to data sets that are quite enormous or intricate that conventional data processing tools are insufficient. The overwhelming amount of data, both unstructured and structured, that a business faces on a daily basis.
- The amount of data produced by healthcare applications, the internet, social networking sites social, sensor networks and many other businesses are rapidly growing as a result of recent technological advancements. Big data refers to the vast volume of data created from numerous sources in a variety of formats at extremely fast rates. Dealing with this kind of data is one of the many challenges of data collection and is a crucial step toward collecting effective data.

1.1.4 Exploratory Data Analysis (EDA)

- **Exploratory Data Analysis (EDA)**, a term defined by John W. Tukey, is a process of examining or understanding the data and extracting insights or main characteristics of the data. EDA is generally classified into two methods, that is, **graphical** analysis and non-**graphical** analysis.
- "EDA" is a critical first step in analyzing the data from an experiment. Any method of looking at data that does not include formal statistical modeling and inference falls under the term exploratory data analysis.
- It is also known as **visual analytics or descriptive statistics**. It is the practice of inspecting and exploring data, before stating hypotheses, fitting predictors and other expected inferential goals. It typically includes the computation of simple summary statistics which capture some property of interest in the data and **visualization**. EDA can be thought of as an assumption-free, purely algorithmic practice.
- Exploratory Data Analysis (EDA) is the crucial process of using summary statistics and graphical representations to perform preliminary investigations on data in order to uncover patterns, detect anomalies, test hypotheses and verify assumptions.
- EDA is an approach to data analysis that postpones the usual assumptions about what kind of model the data follow with the more direct approach of allowing the data itself to reveal its underlying structure and model. EDA is not a mere collection of techniques; EDA is a philosophy that dissects a data set; what is to look for; how

to look; and how to interpret. It is true that EDA heavily uses the collection of techniques that are called as "**statistical graphics**", but it is not identical to statistical graphics.

- Basic purpose of EDA is to spot problems in data (as part of data wrangling) and understand variable properties like,
  - Central trends (mean)
  - Spread (variance)
  - Skew
  - Outliers and anomalies.
- **Below are the most prominent reasons to use EDA**

1. Detection of mistakes.
2. Examine the data distribution.
3. Handling missing values of the dataset(a most common issue with every dataset) .
4. Handling the outliers.
5. Removing duplicate data.
6. Encoding the categorical variables.
7. Normalizing and scaling.
8. Checking of assumptions.
9. Preliminary selection of appropriate models.
10. Determining relationships among the explanatory variables.
11. Assessing the direction and rough size of relationships between explanatory and outcome variables.

1.2 Understanding Data Science

- Data science is the domain of study that deals with vast volumes of data using modern tools and techniques to find unseen patterns, derive meaningful information and make business decisions.
- Data science combines math and statistics, specialized programming, advanced analytics, Artificial Intelligence (AI) and machine learning with specific subject matter expertise to uncover actionable insights hidden in an organization's data. These insights can be used to guide decision making and strategic planning.

- Data science is the field of study that combines domain expertise, programming skills and knowledge of mathematics and statistics to extract meaningful insights from data. Data science practitioners apply machine learning algorithms to numbers, text, images, video, audio and more to produce Artificial Intelligence (AI) systems to perform tasks that ordinarily require human intelligence. In turn, these systems generate insights which analysts and business users can translate into tangible business value.
- The data used for analysis can come from many different sources and presented in various formats.

The data science lifecycle

- Data science's lifecycle consists of five distinct stages, each with its own tasks :
  1. **Capture** - Data acquisition, data entry, signal reception, data extraction - This stage involves gathering raw structured and unstructured data.
  2. **Maintain** - Data warehousing, data cleansing, data staging, data processing, data architecture - This stage covers taking the raw data and putting it in a form that can be used.
  3. **Process** - Data mining, clustering/classification, data modeling, data summarization - Data scientists take the prepared data and examine its patterns, ranges and biases to determine how useful it will be in predictive analysis.
  4. **Analyze** - Exploratory/confirmatory, predictive analysis, regression, text mining, qualitative analysis - This stage involves performing the various analyses on the data.
  5. **Communicate** - Data reporting, data visualization, business intelligence, decision making - In this final step, analysts prepare the analyses in easily readable forms such as charts, graphs and reports.

Data science tools

- Below are various data science tools that can be used in various stages of the data science process.
  - Data Analysis - SAS, Jupyter, R Studio, MATLAB, Excel, RapidMiner
  - Data Warehousing - Informatica / Talend, AWS Redshift
  - Data Visualization - Jupyter, Tableau, Cognos, RAW
  - Machine Learning - Spark MLib, Mahout, Azure ML studio.

Use of data science

1. Data science may detect patterns in seemingly unstructured or unconnected data, allowing conclusions and predictions to be made.
2. Tech businesses that acquire user data can utilize strategies to transform that data into valuable or profitable information.
3. Data science has also made inroads into the transportation industry, such as with driverless cars. It is simple to lower the number of accidents with the use of driverless cars. For example, with driverless cars, training data is supplied to the algorithm and the data is examined using data science approaches, such as the speed limit on the highway, busy streets, etc.
4. Data science applications provide a better level of therapeutic customization through genetics and genomics research.
5. Data science has found its applications in almost every industry.

Applications of data science

1. **Healthcare :** Healthcare companies are using data science to build sophisticated medical instruments to detect and cure diseases. Machine learning models and other data science components are used by hospitals and other healthcare providers to automate X-ray analysis and assist doctors in diagnosing illnesses and planning treatments based on previous patient outcomes.
2. **Gaming :** Video and computer games are now being created with the help of data science and that has taken the gaming experience to the next level.
3. **Image recognition :** Identifying patterns in images and detecting objects in an image is one of the most popular data science applications.
4. **Recommendation systems :** Netflix and Amazon give movie and product recommendations based on what people like to watch, purchase or browse on their platforms.
5. **Logistics :** Data science is used by logistics companies to optimize routes to ensure faster delivery of products and increase operational efficiency.
6. **Fraud detection :** Banking and financial institutions use data science and related algorithms to detect fraudulent transactions.

7. **Internet search :** When one thinks of search, immediately Google comes into mind. However, there are other search engines, such as Yahoo, Duckduckgo, Bing, AOL, Ask and others, that employ data science algorithms to offer the best results for the search query in a matter of seconds. Given that Google handles more than 20 petabytes of data per day. Google would not be the 'Google', widely popular today, if data science did not exist.

8. **Speech recognition :** Speech recognition is dominated by data science techniques. There are excellent applications of these algorithms in daily lives. A virtual speech assistant like Google assistant, Alexa or Siri are few speech recognition applications to name. Its voice recognition technology is operating behind the scenes, attempting to interpret and evaluate the words and delivering useful results from the use. Image recognition may also be seen on social media platforms such as Facebook, Instagram and Twitter. When one submits a picture of their own with someone onone's list, these applications will recognise them and tag them.

9. **Targeted advertising :** From displaying banners on various websites to digital billboards at airports, data science algorithms are utilised to identify almost anything. This is why digital advertisements have a far higher CTR (Call-Through Rate) than traditional marketing. They can be customized based on a user's prior behaviour. That is why one may see adverts for data science training programs while another person sees an advertisement for clothes in the same region at the same time.

10. **Airline route planning :** As a result of data science, it is easier to predict flight delays for the airline industry, which is helping it grow. It also helps to determine whether to land immediately at the destination or to make a stop in between, such as a flight from Delhi to the United States of America or to stop in between and then arrive at the destination.

11. **Virtual reality :** A virtual reality headset incorporates computer expertise, algorithms and data to create the greatest viewing experience possible. The popular game Pokemon GO is a minor step in that direction. The ability to wander about and look at Pokemon on walls, streets and other non-existent surfaces. The makers of this game chose the locations of the Pokemon and gyms using data from ingress, the previous app from the same business.

12. **To evaluate client in retail shopping :** Retailers evaluate client behaviour and purchasing trends in order to provide individualized product suggestions as well as targeted advertising, marketing and promotions. Data science also assists them in managing product inventories and supply chains in order to keep items in stock.

13. **Entertainment :** Da consumer's views, v algorithms are also u of a user.

14. **Finance :** Banks an activities, manage f order to uncover up

15. **Manufacturing :** management and d probable equipmer

1.3 Significa

- Exploratory Data A identify trends in d model is a feasible
- EDA helps data sc or hypothesis test so as to gain insig

Importance of usi

- EDA helps iden EDA helps detec and the relations
- Different fields data primarily i made using the
- It is practically points without collected data are distinctive
- Exploratory da visualize data exploratory an in a data mini

13. **Entertainment :** Data science enables streaming services to follow and evaluate what consumer's views, which aids in the creation of new TV series and films. Data-driven algorithms are also utilised to provide tailored suggestions based on the watching history of a user.

14. **Finance :** Banks and credit card firms mine and analyse data in order to detect fraudulent activities, manage financial risks on loans and credit lines and assess client portfolios in order to uncover upselling possibilities.

15. **Manufacturing :** Data science applications in manufacturing include supply chain management and distribution optimization, as well as predictive maintenance to anticipate probable equipment faults in facilities before they occur.

1.3 Significance of EDA

- Exploratory Data Analysis (EDA) involves using statistics and visualizations to analyze and identify trends in data sets. The primary intent of EDA is to determine whether a predictive model is a feasible analytical tool for business challenges or not.
- EDA helps data scientists gain an understanding of the data set beyond the formal modeling or hypothesis testing task. Exploratory data analysis is essential for any research analysis, so as to gain insights into a data set.

Importance of using EDA for analyzing data sets is,

- EDA helps identify errors in data sets. EDA gives a better understanding of the data set. EDA helps detect outliers or anomalous events. EDA helps to understand data set variables and the relationship among them.
- Different fields of science, economics, engineering and marketing accumulate and store data primarily in electronic databases. Appropriate and well-established decisions should be made using the data collected.
- It is practically impossible to make sense of datasets containing more than a handful of data points without the help of computer programs. To be certain of the insights that the collected data provides and to make further decisions, data mining is performed where there are distinctive analysis processes.
- Exploratory data analysis is key and usually the first exercise in data mining. It allows us to visualize data to understand it as well as to create hypotheses for further analysis. The exploratory analysis centers around creating a synopsis of data or insights for the next steps in a data mining project.

- EDA actually reveals ground truth about the content without making any underlying assumptions. This is the fact that data scientists use this process to actually understand what type of modeling and hypotheses can be created. Key components of exploratory data analysis include summarizing data, statistical analysis and visualization of data.
- Python provides expert tools for exploratory analysis, with pandas for summarizing; scipy, along with others, for statistical analysis; and matplotlib and plotly for visualizations.
- Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions. EDA can help answer questions about standard deviations, categorical variables and confidence intervals. Once EDA is complete and insights are drawn, its features can then be used for more sophisticated data analysis or modeling, including machine learning.

Importance of EDA in Data Processing and Modeling

- EDA makes it simple to comprehend the structure of a dataset, making data modeling easier. The primary goal of EDA is to make data 'clean' implying that it should be devoid of redundancies. It aids in identifying incorrect data points so that they may be readily removed and the data cleaned. Furthermore, it aids us in comprehending the relationship between the variables, providing us with a broader view of the data and allowing us to expand on it by leveraging the relationship between the variables. It also aids in the evaluation of the dataset's statistical measurements.
- Outliers or abnormal occurrences in a dataset can have an impact on the accuracy of machine learning models. The dataset might also contain some missing or duplicate values. EDA may be used to eliminate or resolve all of the dataset's undesirable qualities.

Data Cleaning and Preprocessing

- Data preprocessing and cleansing are critical components of EDA. Understanding the variables and the structure of the dataset is the initial stage in this process. The data must then be cleaned. The dataset may contain redundancy such as irregular data, missing values or outliers that may cause the model to overfit or underfit during training. Removing or resolving these redundancies is known as **data cleaning**. The last part is analysing the relationship between the variables.

1.4 Types of Exploratory Data Analysis

- There are four primary types of EDA namely,
  1. **Univariate non-graphical :** This is the simplest form of data analysis, where the data being analyzed consists of just one variable. Since it is a single variable, it does not deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it.
  2. **Univariate graphical :** Non-graphical methods do not provide a full picture of the data. Graphical methods are therefore required. Common types of univariate graphics include :
     - Stem-and-leaf plots, which show all data values and the shape of the distribution.
     - Histograms, a bar plot in which each bar represents the frequency (count) or proportion (count/total count) of cases for a range of values.
     - Box plots, which graphically depict the five-number summary of minimum, first quartile, median, third quartile and maximum.
  3. **Multivariate non graphical :** Multivariate data arises from more than one variable. Multivariate non-graphical EDA techniques generally show the relationship between two or more variables of the data through cross-tabulation or statistics.
  4. **Multivariate graphical :** Multivariate data uses graphics to display relationships between two or more sets of data. The most used graphic is a grouped bar plot or bar chart with each group representing one level of one of the variables and each bar within a group representing the levels of the other variable.
- Other common types of multivariate graphics include :
  - Scatter plot, which is used to plot data points on a horizontal and a vertical axis to show how much one variable is affected by another.
  - Multivariate chart, which is a graphical representation of the relationships between factors and a response.
  - Run chart, which is a line graph of data plotted over time.
  - Bubble chart, which is a data visualization that displays multiple circles (bubbles) in a two-dimensional plot.
  - Heat map, which is a graphical representation of data where values are depicted by color.

1.5 Making Sense of Data

- In today's digital era data is a powerful tool. Just having possession of data is not sufficient. One should be able to analyze data in the proper manner to get insight into the data.
- Making sense of data means finding meanings from data, generating inference from data. Data analysis is a process of making meaning. Making sense of data educates owners or readers on the steps and issues that need to be considered in order to successfully complete a data analysis or data mining project.
- Consider the process of creating something out of a set of raw materials. For example, given a bolt of fabric and some notions (thread, zippers, buttons and so forth), different people might end up making very different kinds of creations. One person might make a baby dress, another a dance costume and a thirda diwali lamp. But what they come up with must at least be plausibly created from the raw materials. It would be awfully difficult to make a working car or an edible meal out of a bolt of fabric.

Basic steps to make sense of data can be as follows,

1. **Managing data** - Initially in the process of analyzing qualitative data, first task is to arrange or organize data so that one can begin to make sense of it. That is, one has to do some arrangements for holding data and keep it in a certain format before starting the analysis. One should make a comprehensive list of all the data materials. At one level, this is a fairly mechanical process of gathering all the materials and devising a filing system or other way of organizing them so that one can easily access it.
   - Separating different types of data would help to manage the data. If there is a variety of data one can classify it as per need. This classification will help to access the data easily.
   - Keeping data in chronological order makes data access fast. If applicable data can be arranged in certain order so as to access it quickly.
   - Organizing data in topic wise or document wise helps quick access.
   - While maintaining data one should always give a thought to the fact that how many copies of data are to be stored. It would also be important to keep in details of updations of data.
2. **Getting familiar with data** - Based on data collection and its size, it might take long time to get a complete data set. Over the time user may lose on tiny details about data. Hence it is necessary that once the data set is ready the user should browse through data and get well acquainted with data.

3. **Making sense of data** - Once data is collected and a well organized analysis process can start, that would make sense of data. For analyzing data there is a need to have data in certain presentable or viewable format. Data is best seen using tables, charts, graphs, patterns. This would help to analyze the data set.

1.6 Comparing EDA with Classical and Bayesian Analysis

- There are three widely used approaches for data analysis namely,
  1. Classical data analysis
  2. Exploratory data analysis
  3. Bayesian data analysis.
- These three approaches are similar in that they all begin with a general science/engineering problem and all yield science/engineering conclusions. The difference is the sequence and focus of the intermediate steps.
- Classical data analysis follows below steps,

  Problem -> Data -> Model -> Analysis -> Conclusions
- Exploratory data analysis follows below steps,

  Problem -> Data -> Analysis -> Model -> Conclusions
- Bayesian data analysis follows below steps,

  Problem -> Data -> Model -> Prior Distribution -> Analysis -> Conclusions
- That is the in classical analysis, the data collection is followed by the imposition of a model (normality, linearity, etc.) and the analysis, estimation and testing that follows are focused on the parameters of that model.
- For EDA, the data collection is not followed by a model imposition; rather it is followed immediately by analysis with a goal of inferring what model would be appropriate.
- For a Bayesian analysis, the analyst attempts to incorporate scientific/engineering knowledge/expertise into the analysis by imposing a data - independent distribution on the parameters of the selected model; the analysis thus consists of formally combining both the prior distribution on the parameters and the collected data to jointly make inferences and/or test assumptions about the model parameters. In the real world, data analysts freely mix elements of all of the above three approaches and if required other approaches as well.

Classical Analysis Vs EDA

- Classical analysis approach and EDA can be compared on the basis of below parameters

1. **Model**
   - The classical approach imposes models (both deterministic and probabilistic) on the data. Deterministic models include, for example, regression models and analysis of variance (ANOVA) models. The most common probabilistic model assumes that the errors about the deterministic model are normally distributed - This assumption affects the validity of the ANOVA F tests.
   - The exploratory data analysis approach does not impose deterministic or probabilistic models on the data. On the contrary, the EDA approach allows the data to suggest admissible models that best fit the data.
2. **Focus**
   - The two approaches differ drastically in focus. For classical analysis, the focus is on the model estimating parameters of the model and generating predicted values from the model.
   - For exploratory data analysis, the focus is on the data - Its structure, outliers and models suggested by the data.
3. **Techniques**
   - Classical techniques are generally quantitative in nature. They include ANOVA, t tests, chi-squared tests and F tests. EDA techniques are generally graphical. They include scatter plots, character plots, box plots, histograms, bi-histograms, probability plots, residual plots and mean plots.
4. **Rigor**
   - Classical techniques serve as the probabilistic foundation of science and engineering. The most important characteristic of classical techniques is that they are rigorous, formal and "objective". EDA techniques do not share in that rigor or formality. EDA techniques make up for that lack of rigor by being very suggestive, indicative and insightful about what the appropriate model should be. EDA techniques are subjective and depend on interpretation which may differ from analyst to analyst, although experienced analysts commonly arrive at identical conclusions.

5. **Data treatment**

- Classical estimation techniques have the characteristic of taking all of the data and mapping the data into a few numbers ("estimates"). This is both a virtue and a vice. The virtue is that these few numbers focus on important characteristics (location, variation, etc.) of the population. The vice is that concentrating on these few characteristics can filter out other characteristics (skewness, tail length, autocorrelation, etc.) of the same population. In this sense there is a loss of information due to this "**filtering**" process.
- Whereas, the EDA approach often makes use of (and shows) all of the available data. In this sense there is no corresponding loss of information.

6. **Assumptions**

- The classical approach tests based on classical techniques that are usually very sensitive- that is, if a true shift in location, say, has occurred, such tests frequently have the power to detect such a shift and to conclude that such a shift is "**statistically significant**". But classical tests depend on underlying assumptions (e.g., normality) and hence the validity of the test conclusions becomes dependent on the validity of the underlying assumptions. Further the issue is, the exact underlying assumptions may be unknown to the analyst or if known, untested. Thus the validity of the scientific conclusions becomes intrinsically linked to the validity of the underlying assumptions. In practice, if such assumptions are unknown or untested, the validity of the scientific conclusions becomes suspect. Many EDA techniques make little or no assumptions they present and show the data - all of the data - as is, with fewer encumbering assumptions.

1.7 Software Tools for EDA

Python, R, Excel are some of the popular EDA tools.

1. **R** - An open-source programming language and free software environment for statistical computing and graphics supported by the R foundation for statistical computing. The R language is widely used among statisticians in developing statistical observations and data analysis.
2. **Python** - An interpreted, object-oriented programming language with dynamic semantics. Its high-level, built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together. Python and EDA can be used together to identify missing values in a data set, which is important so one can decide how to handle missing values for machine learning.

EDA can be done using python for identifying the missing value in a data set. Other functions that can be performed are the description of data, handling outliers, getting insights through the plots. Its high-level, built-in data structure and dynamic typing and binding make it an attractive tool for EDA. Analyzing a dataset is a hectic task that takes a lot of time. Python provides certain open-source modules that can automate the whole process of EDA and help in saving time.

3. **Excel / Spreadsheet :** A very long running and dependable tool excel remains an indispensable part of analytics industry. Most of the problems faced in analytics projects are solved using this software. It supports all the important features like summarizing data, visualizing data, data wrangling etc. which are powerful enough to inspect data from all possible angles. Microsoft Excel is paid but there are various other spreadsheet tools like open office, google docs, which are certainly worth using.
4. **Weka -** Weka is an easy to learn, machine learning tool, having an intuitive interface to get the job done quickly. It provides options for data pre-processing, classification, regression, clustering, association rules and visualization. Most of the steps while model building can be achieved using Weka. It is built on Java.
5. **Tableau public -** Tableau is a data visualization software. Its a fast visualization software which allows exploring of data, every observation using various possible charts. It uses intelligent algorithms figure out by self about the type of data, best method available etc. For understanding data in real time, tableau can get the job done. In a way, tableau imparts a colorful life to data and allows sharing work with others.

1.8 Visual Aids for EDA

1. Univariate Plots (Used for univariate data - the data containing one variable)

- Univariate plots show the frequency or the distribution shape of a variable. Below are visual tools used to analyze univariate data.

i. Histograms

- Histograms are two-dimensional plots in which the x-axis divides into a range of numerical bins or time intervals. The y-axis shows the frequency values, which are counts of occurrences of values for each bin. Bar graphs have gaps between the bars to indicate that they compare distinct groups, but there are no gaps in histograms. Hence, they convey if the distribution is left/positively skew (most of the data falls to the right side), right/negatively skewed (most of the data falls to the left side), bi-modal (graphs having two distinct peaks), normal (perfectly symmetrical without skew) or uniform (almost all the bins have similar frequency).

- **Example :** Below is the waiting time of the customer at the cash counter of the grocessary shop during peak hours, which was observed by the cashier.

Customer Waiting Time (in mins)
2.30
5.00
3.55
2.50
5.10
4.21
3.33
4.10
2.55
5.07
3.45
4.10
5.12

- For this data a histogram can be created using five bins with five different frequencies, as seen in the chart below. On the Y-axis, it's the average number of customers falling in that particular category. On the X-axis, there is a range of waiting times. For example, the 1st bin range is 2.30 mins to 2.86 mins. It can be noted that the count is three for that category from the table and as seen in the below graph.

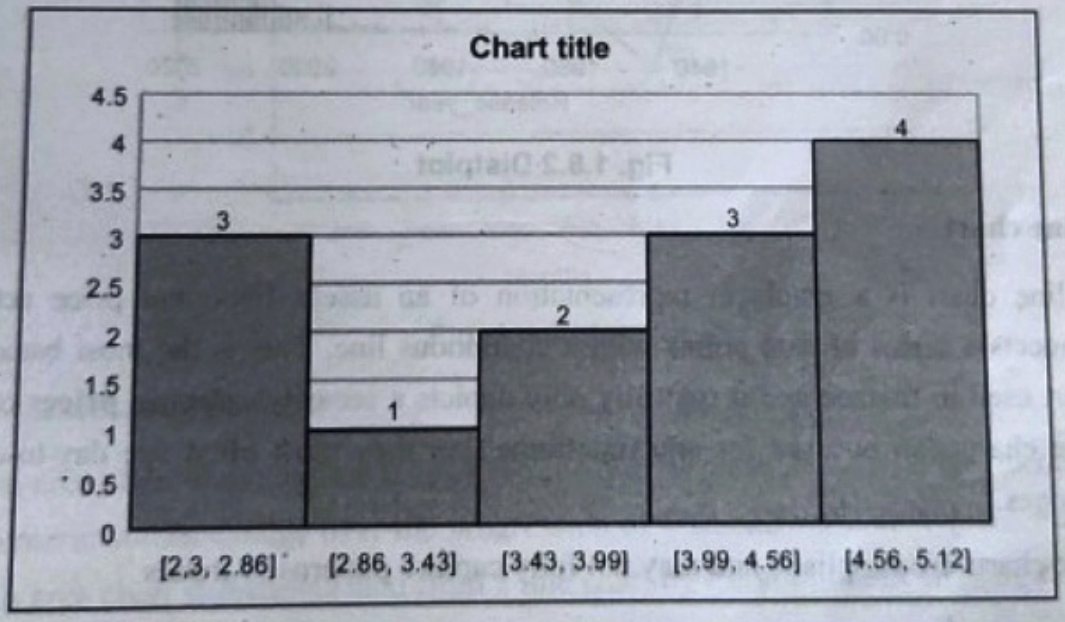


Fig. 1.8.1 Histogram

- It is a random distribution, which is a type of distribution that has several peaks and lacks an apparent pattern.

ii. Distplot

- Distplot is also known as the **second histogram** because it is a slight improved version of the histogram. Distplot gives us a KDE(Kernel Density Estimation) over histogram which explains PDF(Probability Density Function) which means what is the probability of each value occurring in this column. Below is the distplot of distribution of when news channel is most active with new additions over the year.

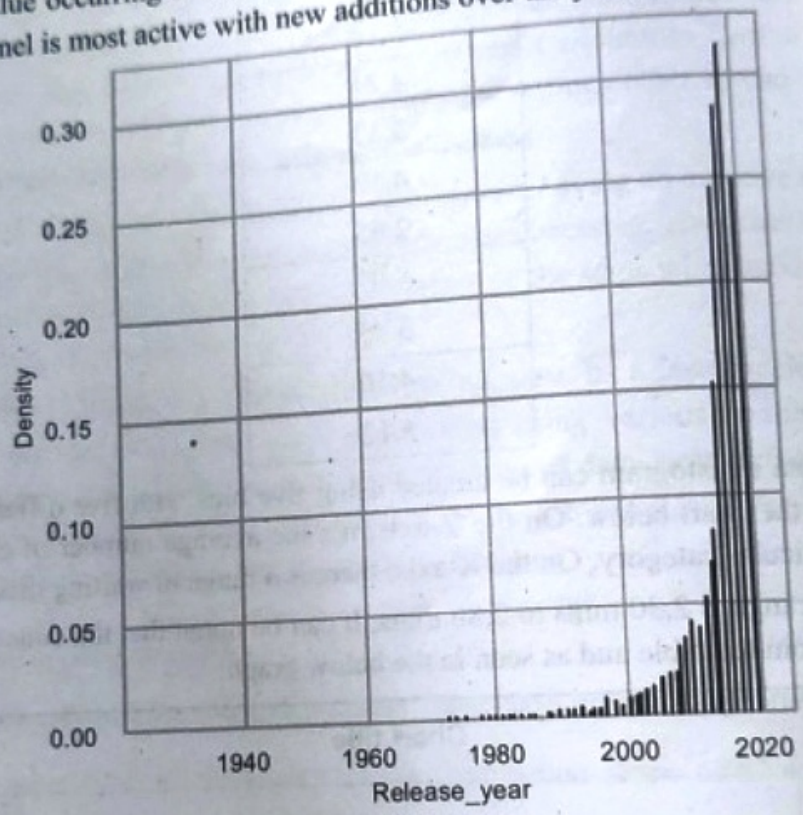


Fig. 1.8.2 Distplot

iii. Line chart

- A line chart is a graphical representation of an asset's historical price action that connects a series of data points with a continuous line. This is the most basic type of chart used in finance and it typically only depicts a security's **closing prices** over time. Line charts can be used for any timeframe, but they most often use day-to-day price changes.
- Line charts are simplistic and may not fully capture patterns or trends.

- There are different types of line charts. They are :
  - **Line chart** - It shows the trend over time (years, months, days) or other categories. It is used when the order of time or types is important.
  - **Line chart with markers** - It is similar to the line chart, but it will highlight data points with markers.
  - **Stacked line chart** - This is a line chart where lines of the data points do not get overlapped because they will be cumulative at each point.
  - **Stacked line chart with markers** - This is similar to the stacked line chart but will highlight data points with markers.
  - **100 % stacked line chart** - It shows the percentage contribution to a whole-time or category.
  - **100 % stacked line chart with markers** - This is similar to a 100 % stacked line chart, but markers will highlight data points.
- Below line chart shows number of houses sold in particular months.

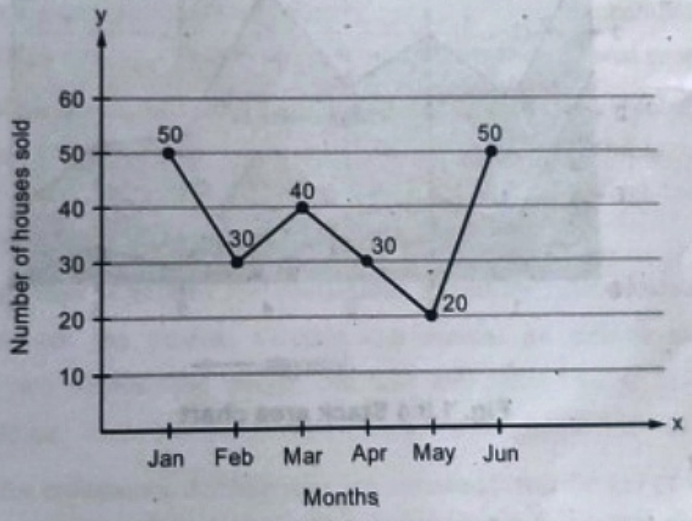


Fig. 1.8.3 Lineplot

iv. Stacked area plot/chart

- An area chart combines the **line chart** and **bar chart** to show how one or more group's numeric values change over the progression of a second variable, typically that of time. An area chart is distinguished from a line chart by the addition of shading between lines and a baseline, like in a bar chart.

- **Stacked area chart is plotted in the form of several area series stacked on top of one another.** The height of each series is determined by the value in each data point. A typical use case for stacked area charts is analyzing how each of several variables, whose their totals vary, on the same graphic.
- A stacked area graph is useful for comparing multiple variables changing over an interval.
- **Example :** Below stack area chart shows data plotting of three data sets.

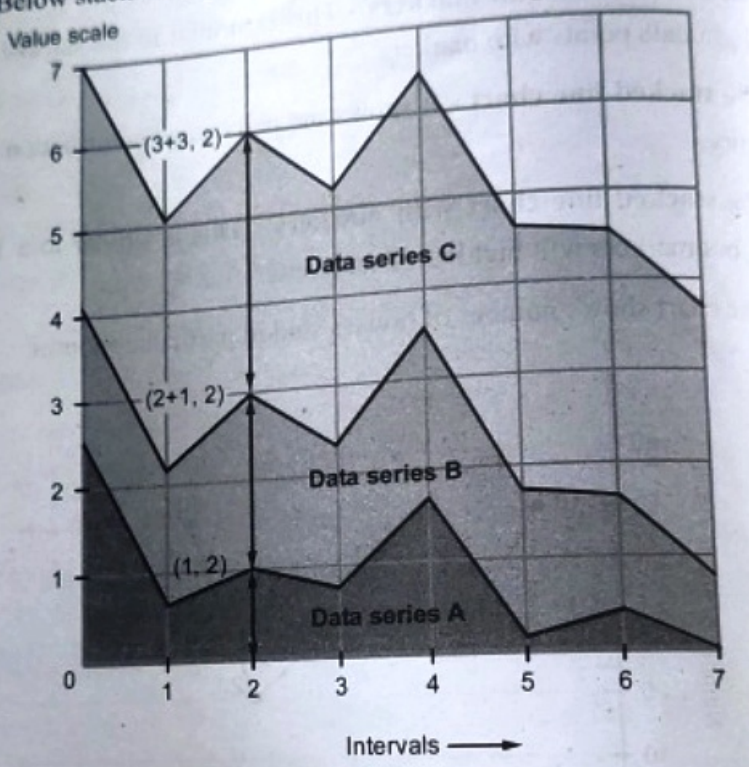


Fig. 1.8.4 Stack area chart

v. Table chart

- **A table chart is** a means of arranging data in rows and columns. **The use of tables is pervasive throughout all communication, research and data analysis. Tables appear in print media, handwritten notes, computer software, architectural ornamentation, traffic signs and many other places.**
- Tables are meant to be read, so they are ideal **when there is data that cannot easily be presented visually or when the data requires more specific attention.**

vi. Probability distribution plots

- Probability distributions are mathematical functions that deseribe all the possible values that a random variable can assume within a given range. They help model random phenomena, allowing us in order to estimate the probability of a particular event. This type of distribution is helpful to know the likely outcomes and the spread of potential values.
- For a single random variable, probability distributions can be divided into two types :

Types :

- **Discrete probability distributions for discrete variables :** Also known as probability mass functions, the random variable can assume a discrete number of values like the number of reviews; it can be 100 or 101, but nothing in between. It returns probabilities; hence the output is between 0 and 1. There are a variety of discrete probability distributions that can be used to model different types of data.
- **Binomial distribution :** There are two possible outcomes in this distribution - success or failure and multiple trials are carried out. The probability of success and failure is the same for all trials. The sum of all probabilities must equal one.

- **Success probability :** So, let's say that there is a success probability of 0.8 of manufacturing a perfect car engine part. What is the probability of having seven successes in 10 attempts ? The probability of success is 0.8 and failure is 0.2. The number of trials is ten and the number of successes is 7.
- **Probability density functions for continuous variables :** Also known as **probability density functions**, the random variable can assume an infinite number of values between any two values; like weight can take any value like 45.3, 45.36. 45.369 or 45.3698 and so on.
- Probabilities for continuous distributions are measured over ranges of values rather than single points. A probability indicates the likelihood that a value will fall within an interval. The entire area under the distribution curve equals 1. For instance, the proportion of the area under the curve that falls within a range of values along the X-axis is the likelihood that a value will fall within that range.

Probability Density Function

- Suppose there is a dataset of adult's heights in a town and the data follows a norm distribution. The mean equals 5.5 and the standard deviation is 1. The shaded p shows the probability that a randomly picked person's height will be smaller than 4 and is approximately equal to 0.15 or 15 %.

vii. Run sequence plots

- A run chart, also known as a **run-sequence plot**, displays observed data in a tim sequence. So, often, the data displayed represents some aspect of a business proce output or performance. It is, therefore, a form of a line chart. They are often analyzed order to locate anomalies in data that suggest shifts in a process over time. Changes location and scale and outliers can easily be detected.

2. Bivariate Plots (Used for bivariate data - the data containing two variables)

- Bivariate plots display the relationship between two variables in exploratory data analysis.

i. Bar graphs

- Bar charts can be used to compare nominal or ordinal data. They are helpful for recognizing trends.

ii. Scatter plots

- Scatter plots are commonly used in statistical analysis in order to visualize numerical relationships. So, they are used in order to determine whether two measures are correlated by plotting them on the x and y-axis. They are suitable for recognizing trends.

iii. Box plots

- These charts show the distribution of values along an axis. Rectangular boxes are used in order to bucket the data, giving us an idea of how the data points are spread out. These boxes are also called **quartiles** which represent a quarter of a data set. Boxes can be drawn vertically or horizontally. One can also easily spot the outliers, which are usually treated as abnormal values and affect the data set's overall observation due to their very high or low values.
- Box plots are suitable for identifying outliers.

iv. Correlation plots (Heat maps)

- For instance, correlation heat maps show the interrelationship between variables - areas as shaded as per the data's values. So, Color differences can easily spot similar and different values and make sense of the data variation. They are usually helpful when there is a large amount of data. They are used during A/B testing to see which parts of a web page are accessed by users on a website. The number of reviews generated every hour or to analyze a cricket match to understand where a batsman is scoring the bulk of his runs or where the bowler is pitching his ball.

v. Cluster map

- One can also use a cluster map to understand the relationship between two categorical variables. A cluster map basically plots a dendrogram that shows the categories of similar behavior together.

3. Special purpose plots

i. Pair plots

- Pair plots are a simple way in order to visualize relationships between multiple variables. So, It produces a matrix of relationships between variables in the data for a direct examination of the data.
- This plot shows how registered and casual users are using bike rentals. It also shows the effect of temperature, humidity and wind speed on bike rentals. This gives an overview of the correlation between multiple variables.

ii. Contour plots

- The contour plot can be used for representing a 3D surface in a 2D format. Contour plots are generally used for continuous variables rather than categorical data.
- The contour maps are inspired by seismic data analysis. They can explain where the data density is high, explore deep learning error functions or gradient analysis.

iii. Density plots

- A density plot is a smoothed, continuous version of a histogram estimated from the data. The most common form of estimation is the kernel density plot. In this method, a continuous curve (the kernel) is drawn at every individual data point. All of these curves are then combined to make a single smooth density estimation.

- So, The y-axis in a density plot is the probability density function for the kernel density estimation and not a probability. The difference is that the probability density is the probability per unit on the x-axis.
- While comparing the distributions of one variable across multiple categories, histograms have issues with readability. Density plots are useful in this scenario.

iv. Polar / Spider / Radar charts

- Polar charts are **circular charts that use values and angles to show information as polar coordinates**. Polar charts are useful for showing scientific data. One can specify a default measure.
- A polar chart is a graphical way of displaying multivariate data of three or more quantitative variables represented on axes starting from the same point. It helps demonstrate a dominant variable.
- **Polar chart** is a common variation of circular graphs. It is useful when relationships between data points can be visualized most easily in terms of radians and angles.
- In polar charts, a series is represented by a closed curve that connects points in the polar coordinate system. Each data point is determined by the distance from the pole (the radial coordinate) and the angle from the fixed direction (the angular coordinate).

v. Lollipop chart

- The lollipop chart is **a composite chart with bars and circles**. It is a variant of the bar chart with a circle at the end, to highlight the data value. Like a bar chart, a lollipop chart is used to compare categorical data. For this kind of composite chart, there are more visual elements to convey information.
- Lollipop Chart (LC) is a handy variation of a bar chart where the bar is replaced with a line and a dot at the end. Just like bar graphs, lollipop plots are used to make **comparisons** between different items or categories. They are also used **for ranking** or for showing **trends over time**. Only one numerical variable is compared per item or category. They are not suitable for relationships, distribution or composition analysis.
- Lollipop charts are two-dimensional with two axes : One axis shows categories or a time series, the other axis shows numerical values.
- LCs are preferred to bar charts when one has to display a large number of similar high values. In that case with a standard bar plot, one may get a cluttered chart and experience an optical effect called a **Moiré pattern** (The Moiré effect is a visual perception that occurs when viewing a set of lines or bars that is superimposed on another set of lines or bars, where the sets differ in relative size, angle or spacing.).

ction for the kernel density
probability density is the

tiple categories, histograms
enario.

to show information as
fic data. One can specify a

e data of three or more
ne same point. It helps

seful when relationships
dians and angles.

nects points in the polar
nce from the pole (the
ular coordinate).

It is a variant of the bar
a bar chart, a lollipop
nposite chart, there are

bar is replaced with a
ots are used to make
o used for ranking or
compared per item or
mposition analysis.

ws categories or a time

umber of similar high
a cluttered chart and
iré effect is a visual
t is superimposed on
gle or spacing.).

vi. Lag plots

- A relationship between an observation and the previous observation is beneficial in time series modeling. Previous observations in a time series are lags, with the observation at one previous time step. It is known as lag1, the observation at two previous steps lag 2 and so on.
- A lag plot is a useful type of plot in order to explore each observation's relationship and a lag of that observation and is displayed as a scatter plot. If the points cluster along a diagonal line from the bottom-left to the plot's top-right, it suggests a positive correlation relationship. If the points cluster along a diagonal line from the top-left to the bottom-right, it means a negative correlation relationship.
- Lag plots can help compare observations simultaneously in the last week or last month or the previous year by using corresponding lag values.
- The plot here shows the count of bike rentals compared to the previous day's count and it displays a relatively strong positive correlation.

vii. Auto-correlation plots

- The correlation between observations and their lag values in a time series name autocorrelation. Correlation coefficients are plotted on an autocorrelation plot.
- A correlation coefficient is a correlation value between observations and their lag 1 values and results in a number between – 1 and + 1. A value close to zero suggests a weak correlation, whereas a value closer to – 1 or 1 indicates a strong correlation. It helps better understand how this relationship changes over the lag. It shows the lag on the x-axis and the correlation on the y-axis.

viii. Lognormal plots

- A normal distribution can be converted to a lognormal distribution using logarithmic mathematics. The lognormal distribution plots the log of random variables from a normal distribution curve. It displays the Probability Density Function (PDF) and is of particular interest when the variable must be positive as log values are always positive.
- Many examples follow lognormal distribution like the concentration of elements and their radioactivity in the Earth's crust, latent periods of infectious diseases, the distribution of particles, chemicals and organisms in the environment, the length of comments posted on social media website discussion forums or fluctuations in the stock markets.

ix. Violin plot

- A violin plot is a hybrid of a box plot and a kernel density plot, which shows peaks in the data. It is used to visualize the distribution of numerical data. Unlike a box plot that can only show summary statistics, violin plots depict summary statistics and the density of each variable.

x. Joint plot

- While the pair plot provides a visual insight into all possible correlations, the joint plot provides bivariate plots with univariate marginal distributions.

xi. Pie chart

- The pie chart is also the same as the countplot, only gives additional information about the percentage presence of each category in data which means category is getting how much weightage in data.

4. Multivariate Visualization (Used for multivariate data - The data containing more than two variables)

- When dealing with multiple variables, it is tempting to make three dimensional plots. Such data can be viewed with scatter plots of the relation between each variable.
- Combined charts also are handy ways to visualize data, since each chart is simple to understand on its own.
- For very large dimensionality, one can reduce the dimensionality using principal component analysis or other techniques and then make a plot of the reduced variables. This is particularly important for high dimensionality data and has applications in deep learning such as visualizing natural language or images.

Text data

- For example with text data, one could create a world cloud, where the size of each word is based on its frequency in the text. To remove the words which add noise to the dataset, the documents can be grouped using topic modeling and only the important words can be displayed.

Image data

- When doing image classification, it is common to use decomposition and remove the dimensionality of the data.
- Instead of blindly using decomposition, a data scientist could plot the result :
  - By looking at the contrast (black and white) in the images, one can see there is an importance with the locations of the eyes, nose and mouth, along with the head shape.

1.9 Data Transformation Techniques

- Data transformation techniques refer to all the actions that help to transform the raw data into a clean and ready-to-use dataset.
- There are different types of data transformation techniques that offer a unique way of transforming the data and there is a chance that there will not be a need for all these techniques on every projects. Below are basic data transformation techniques that can be used in analysis project or data pipelines.

1. Data smoothing

- Smoothing is a technique where an algorithm is applied in order to remove noise from the dataset when trying to identify a trend. Noise can have a bad effect on the data and by eliminating or reducing it one can extract better insights or identify patterns that would not be seen otherwise.
- There are three algorithm types that help with data smoothing :
  - **Clustering :** Where one can group similar values together to form a cluster while labeling any value out of the cluster as an outlier.
  - **Binning :** Using an algorithm for binning will help to split the data into bins and smooth the data value within each bin.
  - **Regression :** Regression algorithms are used to identify the relation between two dependent attributes and help to predict an attribute based on the value of the other.

2. Attribution construction

- Attribution construction is one of the most common techniques in data transformation pipelines. Attribution construction or feature construction is the process of creating new features from a set of the existing features/attributes in the dataset.
- Imagine working in marketing and trying to analyze the performance of a campaign. One may have all the impressions that the campaign generated the total cost for the given time frame. Instead of trying to compare these two metrics across all of the campaigns, one can construct another metric to calculate the cost per million impressions or CPM.

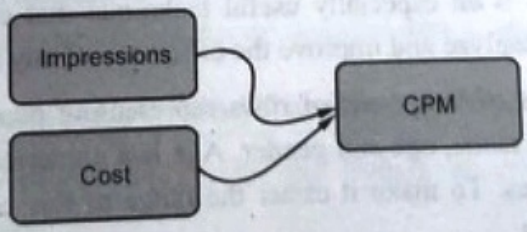


Fig. 1.9.1 Attribution construction

- This will make data exploration and analysis process a lot easier, as one can compare the campaign performance on a single metric rather than two separate metrics.

3. Data generalization

- Data generalization refers to the process of transforming low-level attributes into high-level ones by using the concept of hierarchy. Data generalization is applied to categorical data where they have a finite but large number of distinct values.
- This is something that everyone is already doing without noticing and it helps to get a clearer picture of the data. Suppose there are four categorical attributes in the database,

1. City
2. Street
3. Country
4. State / province.

- One can define a hierarchy between these attributes by specifying the total ordering among them at the schema level, for example :

street < city < state/province < country.

4. Data aggregation

- Data aggregation is possibly one of the most popular techniques in data transformation. When data aggregation is applied to the raw data, it is essentially a process of storing and presenting data in a summary format.
- This is ideal when one wants to perform statistical analysis over the data as one might want to aggregate the data over a specific time period and provide statistics such as average, sum, minimum and maximum.
- For instance, raw data can be aggregated over a given time period to provide statistics such as average, minimum, maximum, sum and count. After the data is aggregated and written as a report, one can analyse the aggregated data to gain insights about particular resources or resource groups. There are two types of data aggregation : Time aggregation and spatial aggregation.

5. Data discretization

- Data discretization refers to the process of transforming continuous data into a set of data intervals. This is an especially useful technique that can help to make the data easier to study and analyze and improve the efficiency of any applied algorithm.
- Imagine having tens of thousands of rows representing people in a survey providing their first name, last name, age and gender. Age is a numerical attribute that can have a lot of different values. To make it easier the range of this continuous attribute can be divided into intervals.

- Mapping this attribute to a higher-level concept, like youth, middle-aged and senior, can help a lot with the efficiency of the task and improve the speed of the algorithms applied.
- There are a wide variety of discretization methods starting with naive methods such as equal-width and equal-frequency to much more sophisticated methods such as MDLP.

6. Data normalization

- Data normalization is the process of scaling the data to a much smaller range, without losing information in order to help minimize or exclude duplicated data and improve algorithm efficiency and data extraction performance.
- There are three methods to normalize an attribute :
  - **Min-max normalization :** Where one performs a linear transformation on the original data.
  - **Z-score normalization :** In z-score normalization (or zero-mean normalization) one is normalizing the value for attribute A using the mean and standard deviation.
  - **Decimal scaling :** Where one can normalize the value of attribute A by moving the decimal point in the value.
- Normalization methods are frequently used when there are values that skew the dataset and it is hard to extract valuable insights.

7. Integration

- Data integration is a crucial step in data pre-processing that involves combining data residing in different sources and providing users with a unified view of these data. It includes multiple databases, data cubes or flat files and works by merging the data from various data sources. There are mainly two major approaches for data integration : Tight coupling approach and loose coupling approach.

8. Manipulation

- Data manipulation is the process of changing or altering data to make it more readable and organised. Data manipulation tools help identify patterns in the data and transform it into a usable form to generate insights on financial data, customer behaviour etc.

1.10 Merging Database (Using Pandas Library)

- Pandas is a software library written for the Python programming language for data manipulation and analysis.

- The Series and DataFrame objects in pandas are powerful tools for exploring and analyzing data. Part of their power comes from a multifaceted approach to combining separate datasets. With pandas, one can **merge**, **join** and **concatenate** datasets, allowing to unify and better to understand the data being analyzed.
- The **Pandas DataFrame** is a structure that contains **two-dimensional data** and its corresponding **labels**. Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). DataFrames are widely used in data science, machine learning, scientific computing and many other data-intensive fields.
- DataFrames are similar to SQL tables or the spreadsheets that one works with in Excel or Calc. In many cases, DataFrames are faster, easier to use and more powerful than tables or spreadsheets because they are an integral part of the Python and NumPy ecosystems.
- A Pandas Series is like a column in a table. It is a one-dimensional array holding data of any type.

1.10.1 Pandas Merge(): Combining Data on Common Columns or Indices

- Merge()can be used when functionality similar to a database's **join** operations is required. It is the most flexible operation that can be applied to data.
- When one wants to combine data objects based on one or more keys, similar to what is done in a relational database, merge() is the tool one can use. More specifically, merge() is most useful when one wants to combine rows that share data.
- One can achieve both **many-to-one** and **many-to-many** joins with merge(). In a many-to-one join, one of the datasets will have many rows in the merge column that repeat the same values. For example, the values could be 1, 1, 3, 5 and 5. At the same time, the merge column in the other dataset won't have repeated values. Take 1, 3 and 5 as an example.
- As it can be seen, in a many-to-many join, both of the merge columns will have repeated values. These merges are more complex and result in the cartesian product of the joined rows.
- This means that, after the merge, there will be every combination of rows that share the same value in the key column.
- What makes merge() so flexible is the sheer number of options for defining the behavior of the merge.

- Formerge() two arguments are required,
  1. The left DataFrame
  2. The right DataFrame.
- After that, it requires other optional arguments mentioned below, to define how the datasets are merged.
  - **how** defines what kind of merge to make. It defaults to 'inner', but other possible options include 'outer', 'left' and 'right'.
  - **on** tells merge() which columns or indices, also called **key columns or key indices**, one wants to join on. This is optional. If it isn't specified and left_index and right_index (covered below) are False, then columns from the two DataFrames that share names will be used as join keys. If on is used, then the specified column or index must be present in both objects.
  - **left_on** and **right_on** specify a column or index that's present only in the left or right object that is being merged. Both default to **None**.
  - **left_index** and **right_index** both default to False, but if it uses the index of the left or right object to be merged, then one can set the relevant argument to True.
  - **suffixes** is a tuple of strings to append to identical column names that aren't merge keys. This allows us to keep track of the origins of columns with the same name.
- These are some of the most important parameters to pass to merge().

Using merge()

- Before getting into the details of how to use merge(), one should first understand the various forms of joins :
  - Inner
  - Outer
  - Left
  - Right

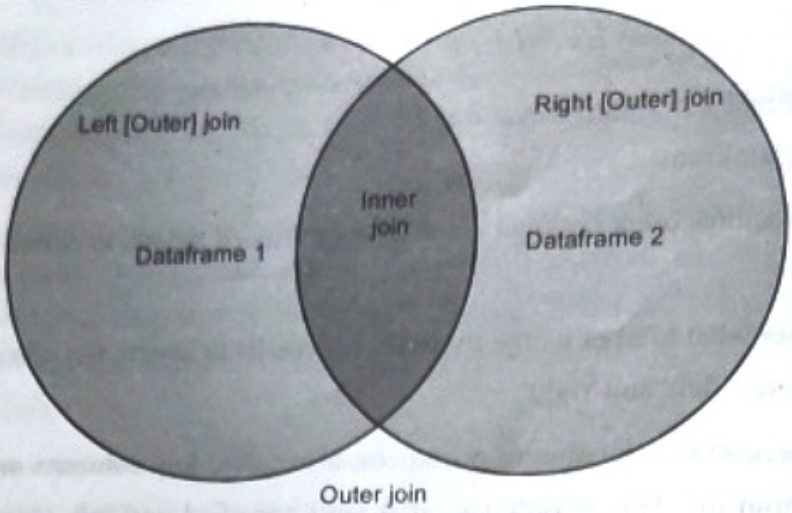


Fig. 1.10.1 Merge and Joins

- In this image, the two circles are the two datasets and the labels point to which part or parts of the datasets is expected to be seen.

1.10.2 Pandas .join() : Combining Data on a Column or Index

- While merge() is a **module function**, .join() is an **instance method** that lives on the DataFrame. This enables you to specify only one DataFrame, which will join the DataFrame call .join() on.
- Under the hood, .join() uses merge(), but it provides a more efficient way to join DataFrames than a fully specified merge() call.

Using .join()

- By default, .join() will attempt to do a left join on indices. If one wants to join on columns similar to merge(), then one needs to set the columns as indices.
- Like merge(), .join() has a few parameters that give more flexibility in the joins operations. However, with .join(), the list of parameters is relatively short,
  - **Other** is the only required parameter. It defines the other DataFrame to join. One can also specify a list of DataFrames here, allowing to combine a number of datasets in a single .join() call.
  - **On** specifies an optional column or index name for the left DataFrame to join the other DataFrame's index. If it's set to none, which is the default, then one will get an index-on-index join.
  - **How** has the same options as how from merge(). The difference is that it's index-based unless the columns are also specified.

- **lsuffix** and **rsuffix** are similar to suffixes in merge(). They specify a suffix to add to any overlapping columns but have no effect when passing a list of other DataFrames.
- **Sort** can be enabled to sort the resulting DataFrame by the join key.

Example Program - 1

```
#using merge function
import pandas as pd
# First DataFrame
left = pd.DataFrame(

  {
    "key": ["K0", "K1", "K2", "K3"],

    "A": ["A0", "A1", "A2", "A3"],

    "B": ["B0", "B1", "B2", "B3"],

  }
)

# Second DataFrame
right = pd.DataFrame(

  {

    "key": ["K0", "K1", "K2", "K3"],

    "C": ["C0", "C1", "C2", "C3"],

    "D": ["D0", "D1", "D2", "D3"],

  }
)
result = pd.merge(left, right, on="key")
print(result)
```

Example Program - 1 Output

key	A	B	C	D
0 K0	A0	B0	C0	D0
1 K1	A1	B1	C1	D1
2 K2	A2	B2	C2	D2
3 K3	A3	B3	C3	D3

1.10.3 Pandas concat() : Combining Data across Rows or Columns

- Concatenation is a bit different from the merging techniques. With merging, one can expect the resulting dataset to have rows from the parent datasets mixed in together, often based on some commonality. Depending on the type of merge, one might also lose rows that don't have matches in the other dataset.
- With concatenation, the datasets are just stitched together along an **axis** either the **row axis** or **column axis**. Visually, a concatenation with no parameters along rows would look as shown below,

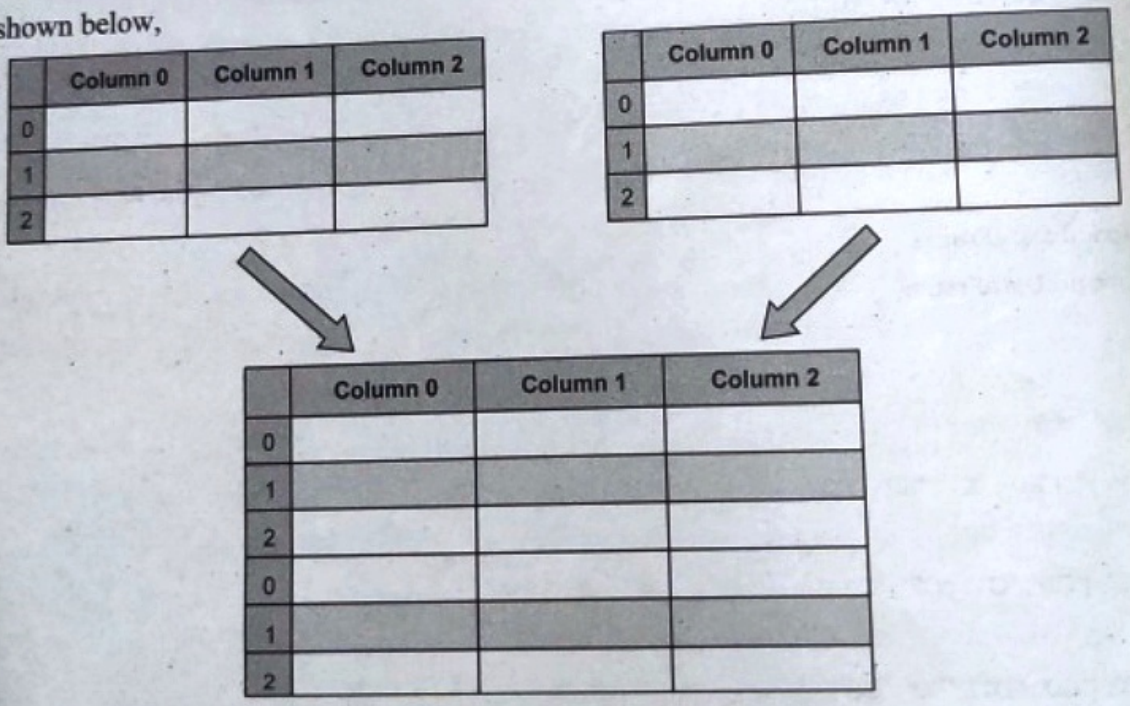


Fig. 1.10.2 Concatenation operation

Using concat()

- As it can be seen, concatenation is a simpler way to combine datasets. It's often used to form a single, larger set to do additional operations on.
- While concatenating datasets, one can specify the axis along which one wants to concatenate.

- By default, a concatenation results in **a set union**, where all data is preserved with the same operations like merge() and .join() as an outer join, with the required join parameter.
- If this parameter is used, then the default is outer, but there is an inner option available, which will perform an inner join or **set intersection**.
- As with the case of other inner joins, some data loss can occur when an inner join with concat() is carried out. Only here the axis labels match preserves rows or columns.
- Below are some of the other parameters that concat() takes,
  - **Objs** takes any sequence typically a list of Series or DataFrame objects to be concatenated. One can also provide a dictionary. In this case, the keys will be used to construct a hierarchical index.
  - **Axis** represents the axis that concatenate along. The default value is 0, which concatenates along the index or row axis. Alternatively, a value of 1 will concatenate vertically, along columns. One can also use the string values "index" or "columns".
  - **Join** is similar to the how parameter in the other techniques, but it only accepts the values inner or outer. The default value is outer, which preserves data, while inner would eliminate data that doesn't have a match in the other dataset.
  - **Ignore_index** takes a **Boolean** True or False value. It defaults to False. If True, then the new combined dataset won't preserve the original index values in the axis specified in the axis parameter. This gives entirely new index values.
  - **Keys** allows to construct a hierarchical index. One common use case is to have a new index while preserving the original indices so that one can tell which rows, for example, come from which original dataset.
  - **Copy** specifies whether one wants to copy the source data. The default value is True. If the value is set to False, then pandas won't make copies of the source data.

Example Program - 2

```
#Using concat function
import pandas as pd
# First DataFrame
df1 = pd.DataFrame({'id': ['T01', 'T02', 'T03', 'T04'],
                    'Name': ['Jiya', 'Riya', 'Maya', 'Piya']})
# Second DataFrame
df2 = pd.DataFrame({'id': ['R05', 'R06', 'R07', 'R08'],
                    'Name': ['Raam', 'Raaj', 'Naaz', 'Saaj']})
```

```
frames = [df1, df2]
result = pd.concat(frames)
print(result)
```

Example Program - 2 Output

```
   id  Name
0 T01  Jiya
1 T02  Riya
2 T03  Maya
3 T04  Piya
0 R05  Raam
1 R06  Raaj
2 R07  Naaz
3 R08  Saaj
```

1.11 Reshaping and Pivoting

1.11.1 Joining and Splitting Data - melt(), split(), pivot()

- In Pandas data reshaping means the transformation of the structure of a table or vector (i.e DataFrame or Series) to make it suitable for further analysis. Some of Pandas reshaping capabilities do not readily exist in other environments (e.g. SQL or bare bone R).
- Pandas has two methods namely, **melt()** and **pivot()**, to reshape the data.

melt()

- This method flattens/melts tabular data such that the specified columns and their respective values are transformed into key-value pairs. 'keys' are the column names of the dataset before transformation and 'values' are the values in the respective columns. After transformation, 'keys' are stored in a column named 'variable' and 'values' are stored in another column named 'value', by default.
- The columns of the data frame are transformed into key-value pairs.
- Unpivot a DataFrame from wide format to long format, optionally leaving identifier variables set.
- This function is useful to massage a DataFrame into a format where one or more columns are identifier variables (id_vars), while all other columns, considered measured variables (value_vars), are "unpivoted" to the row axis, leaving just two non-identifier columns, 'variable' and 'value'.

Syntax

- pandas.melt(frame, id_vars=None, value_vars=None, var_name=None, value_name='value', col_level=None)

Parameters :

Name	Description	Type	Required / Optional
frame	DataFrame	DataFrame - Contains list, numbers, strings	Required
id_vars	Column(s) to use as identifier variables.	tuple, list or ndarray	Optional
value_vars	Column(s) to unpivot. If not specified, uses all columns that are not set as id_vars.	tuple, list or ndarray	Optional
var_name	Name to use for the 'variable' column. If None it uses frame.columns.name or 'variable'.	scalar	Required
value_name	Name to use for the 'value' column.	scalar, default 'value'	Required
col_level	If columns are a MultiIndex then use this level to melt.	int or string	Optional

Returns : Unpivoted DataFrame.

Example Program - 3

```
#using melt() function
import pandas as pd
df1 = {"Name": ["Ravi", "Aniket", "Ana"], "ID": [1, 2, 3], "Role": ["CEO", "Editor", "Author"]}
df = pd.DataFrame(df1)
print(df)
df_melted = pd.melt(df, id_vars=["ID"], value_vars=["Name", "Role"])
print(df_melted)
```

Example Program - 3 Output

	Name	ID	Role
0	Ravi	1	CEO
1	Aniket	2	Editor
2	Ana	3	Author

	ID	variable	value
0	1	Name	Ravi
1	2	Name	Aniket
2	3	Name	Ana
3	1	Role	CEO
4	2	Role	Editor
5	3	Role	Author

pivot()

- This method does the reverse of what melt() did. It transforms the key-value pairs into columns.
- It returns a reshaped DataFrame organized by given index / column values.
- Reshape data (produce a "pivot" table) based on column values. Uses unique values from specified index / columns to form axes of the resulting DataFrame. This function does not support data aggregation, multiple values will result in a MultiIndex in the columns.
- pandas.pivot(data, index=None, columns=None, values=None)[source]

Parameters :

Name	Description	Type	Required / Optional
Data	DataFrame	DataFrame - Contains list, numbers, strings	Required
Index	Column to use to make a new frame's index. If None, uses existing index.	String or object	Optional
Columns	Column to use to make new frame's columns.	String or object	Required
Values	Column(s) to use for populating new frame's values. If not specified, all remaining columns will be used and the result will have hierarchically indexed columns.	String, object or a list of the previous	Optional

Returns : Returns reshaped DataFrame.

Raises : ValueError : When there are any index, column combinations with multiple values.

Example Program - 4

```
import pandas as pd

d1 = {"Name": ["Ravi", "Aniket", "Ana"], "ID": [1, 2, 3], "Role": ["CEO", "Editor", "Author"]}

df = pd.DataFrame(d1)

# print(df)

df_melted = pd.melt(df, id_vars=["ID"], value_vars=["Name", "Role"], var_name="Attribute",
value_name="Value")
print(df_melted)
# unmelting using pivot()
df_unmelted = df_melted.pivot(index='ID', columns='Attribute')
print(df_unmelted)
```

Program Output - 4

```
   ID  Attribute  Value
0   1       Name    Ravi
1   2       Name  Aniket
2   3       Name     Ana
3   1       Role     CEO
4   2       Role  Editor
5   3       Role  Author


        Value
Attribute  Name    Role
ID
1          Ravi     CEO
2        Aniket  Editor
3           Ana  Author
```

1.11.2 Transformation Techniques

1. Grouping data sets

- **Pandas groupby** is used for grouping the data according to the categories and applying a function to the categories. It also helps to aggregate data efficiently.
- Pandas **dataframe.groupby()** function is used to split the data into groups based on some criteria. pandas objects can be split on any of their axes. The abstract definition of grouping is to provide a mapping of labels to group names.
- groupby() is a very powerful function with a lot of variations. It makes the task of splitting the dataframe over some criteria really easy and efficient.
- Any **groupby** operation involves one of the following operations on the original object. They are,
  - **Splitting** the object
  - **Applying** a function
  - **Combining** the results.

Syntax

- DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)

Parameters :

- **By :** mapping, function, str or iterable
- **Axis :** int, default 0
- **Level :** If the axis is a MultiIndex (hierarchical), group by a particular level or levels
- **As_index :** For aggregated output, return an object with group labels as the index. Only relevant for DataFrame input. as_index = False is effectively "SQL-style" grouped output
- **Sort :** Sort group keys. Get better performance by turning this off. Note this does not influence the order of observations within each group. groupby preserves the order of rows within each group.
- **Group_keys :** When calling apply, add group keys to index to identify pieces
- **Squeeze :** Reduce the dimensionality of the return type if possible, otherwise return consistent type

Returns : GroupBy object

- In many situations, the data is split into sets and some functionality can be applied on each subset. In the applied functionality, one can perform the following operations.
  - **Aggregation** - Computing a summary statistic
  - **Transformation** - Perform some group-specific operation
  - **Filtration** - Discarding the data with some condition.

2. Split data into groups

- Pandas objects can be split into any of their objects. There are multiple ways to split an object like -
  - obj.groupby('key')
  - obj.groupby(['key1','key2'])
  - obj.groupby(key,axis=1)

3. Data aggregations

- An aggregated function returns a single aggregated value for each group. Once the **group by** object is created, several aggregation operations can be performed on the grouped data.

Applying multiple aggregation functions at once

- With grouped series, one can also pass a **list** or **dict of functions** to do aggregation with and generate DataFrame as output

4. Transformations

- Transformation on a group or a column returns an object that is indexed the same size as that of that is being grouped. Thus, the transform should return a result that is the same size as that of a group chunk.

5. Filtration

- Filtration filters the data on a defined criteria and returns the subset of data. The **filter()** function is used to filter the data.

Example Program - 5

```
#grouping data
import pandas as pd
punepl_data = {'Team': ['Warriors', 'Fighters', 'Supers', 'Fighters', 'Warriors',
 'Fighters', 'Fighters', 'Supers', 'Supers', 'Fighters', 'Warriors', 'Supers'],
 'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
```

```
  'Year': [2004,2005,2004,2005,2004,2005,2006,2007,2006,2004,2005,2007],
  'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
df = pd.DataFrame(punepl_data)
print (df)
print (df.groupby('Team').groups)

#iterating through groups
grouped = df.groupby('Year')
for name,group in grouped:
  print (name)
  print (group)
#applying grouping and then applying aggregate function
grouped = df.groupby('Year')
print (grouped['Points'].agg(np.mean))
#applying multiple aggregate functions
grouped = df.groupby('Team')
print (grouped['Points'].agg([np.sum, np.mean, np.std]))
#applying filtration
# filter condition, return the teams which have participated four or more times in IPL
print (df.groupby('Team').filter(lambda x: len(x)>=4))
```

Program Output - 5

	Team	Rank	Year	Points
0	Warriors	1	2004	8761
1	Fighters	2	2005	7892
2	Supers	2	2004	8633
3	Fighters	3	2005	6734
4	Warriors	3	2004	7415
5	Fighters	4	2005	8126
6	Fighters	1	2006	7567
7	Supers	1	2007	7888
8	Supers	2	2006	6949
9	Fighters	4	2004	7010
10	Warriors	1	2005	8014
11	Supers	2	2007	6902

```
{'Fighters': [1, 3, 5, 6, 9], 'Supers': [2, 7, 8, 11], 'Warriors': [0, 4, 10]}
```

2004

	Team	Rank	Year	Points
0	Warriors	1	2004	8761
2	Supers	2	2004	8633
4	Warriors	3	2004	7415
9	Fighters	4	2004	7010

2005

	Team	Rank	Year	Points
1	Fighters	2	2005	7892
3	Fighters	3	2005	6734
5	Fighters	4	2005	8126
10	Warriors	1	2005	8014

2006

	Team	Rank	Year	Points
6	Fighters	1	2006	7567
8	Supers	2	2006	6949

2007

	Team	Rank	Year	Points
7	Supers	1	2007	7888
11	Supers	2	2007	6902

Year

2004 7954.75

2005 7691.50

2006 7258.00

2007 7395.00

Name : Points, dtype: float64

	sum	mean	std
Team			
Fighters	37329	7465.800000	585.456403
Supers	30372	7593.000000	828.822860
Warriors	24190	8063.333333	674.354753

	Team	Rank	Year	Points
1	Fighters	2	2005	7892
2	Supers	2	2004	8633
3	Fighters	3	2005	6734
5	Fighters	4	2005	8126
6	Fighters	1	2006	7567
7	Supers	1	2007	7888
8	Supers	2	2006	6949
9	Fighters	4	2004	7010
11	Supers	2	2007	6902

6. Pivot-tables and cross tabulations

- A pivot table is a table of statistics that helps summarize the data of a larger table by "pivoting" that data. Microsoft Excel popularized the pivot table, where they are known as **PivotTables**.
- *Pivot table in pandas is a tool to summarize one or more numeric variable based on two other categorical variables.* Pandas gives access to creating pivot tables in Python using the .pivot_table() function.
- **pandas.pivot_table(data, values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All')** create a spreadsheet-style pivot table as a DataFrame.
- Levels in the pivot table will be stored in MultiIndex objects (hierarchical indexes) on the index and columns of the result DataFrame.

Parameters :

- **Data :** DataFrame
- **Values :** Column to aggregate, optional
- **Index :** Column, Grouper, array or list of the previous
- **Columns :** Column, Grouper, array or list of the previous
- **Aggfunc :** Function, list of functions, dict, default numpy.mean.

  -> If list of functions passed, the resulting pivot table will have hierarchical columns whose top level are the function names.

  -> If dict is passed, the key is column to aggregate and value is function or list of functions.

- **fill_value[scalar, default None]** : Value to replace missing values with **margins[boolean, default False]** : Add all row / columns (e.g. for subtotal / grand totals)
- **dropna[boolean, default True]** : Do not include columns whose entries are all NaN **margins_name[string, default 'All']** : Name of the row / column that will contain the totals when margins is True.
- **Returns :** DataFrame

Example Program - 5

```
import pandas as pd
import numpy as np

df = pd.DataFrame({'First Name': ['Mina', 'Sima', 'Rima', 'Sita', 'Rita', ],
          'Last Name': ['Das', 'Wani', 'Kapur', 'Pande', 'Kumar'],
          'Type': ['Full-time Employee', 'Intern', 'Full-time Employee',
               'Part-time Employee', 'Full-time Employee'],
          'Department': ['Administration', 'Technical', 'Administration',
                    'Technical', 'Management'],
          'YoE': [2, 3, 5, 7, 6],
          'Salary': [20000, 5000, 10000, 10000, 20000]})
print (df)
#creating pivot table
output = pd.pivot_table(data=df,
               index=['Type'],
               columns=['Department'],
               values='Salary',
               aggfunc='mean')
print (output)

# Pivot table with multiple aggfuncs
output = pd.pivot_table(data=df, index=['Type'],
               values='Salary',
               aggfunc=['sum', 'mean', 'count'])
print (output)
```

```
# Calculate row and column totals (margins)
output = pd.pivot_table(data=df, index=['Type'],
                        values='Salary',
                        aggfunc=['sum', 'mean', 'count'],
                        margins=True,
                        margins_name='Grand Total')
print (output)
```

Program Output - 6

	First Name	Last Name	Type	Department	YoE	Salary
0	Mina	Das	Full-time Employee	Administration	2	20000
1	Sima	Wani	Intern	Technical	3	5000
2	Rima	Kapur	Full-time Employee	Administration	5	10000
3	Sita	Pande	Part-time Employee	Technical	7	10000
4	Rita	Kumar	Full-time Employee	Management	6	20000

Department	Administration	Management	Technical
Type			
Full-time Employee	15000.0	20000.0	NaN
Intern	NaN	NaN	5000.0
Part-time Employee	NaN	NaN	10000.0

	sum	mean	count
	Salary	Salary	Salary
Type			
Full-time Employee	50000	16666.666667	3
Intern	5000	5000.000000	1
Part-time Employee	10000	10000.000000	1

	sum	mean	count
	Salary	Salary	Salary
Type			
Full-time Employee	50000	16666.666667	3
Intern	5000	5000.000000	1
Part-time Employee	10000	10000.000000	1
Grand Total	65000	13000.000000	5

- **The Pandas crosstab function is one of the many ways in which Pandas allows to customize data**. On the surface, it appears to be quite similar to the Pandas pivot table function.

Review Questions with Answers

1. *Explain various data collection methods.* **(Refer section 1.1)**
2. *What is EDA ? What is significance of EDA ?* **(Refer section 1.1)**
3. *Define data science.* **(Refer section 1.2)**
4. *What do you mean by making sense of data ?* **(Refer section 1.5)**
5. *Discuss various visual aids for EDA.* **(Refer section 1.8)**

1.12 Two Marks Questions and Answers

Q.1 List widely used data collection tools.

Ans. : Below are some of the widely used tools for data collection,

- **Word Association :** The researcher gives the respondent a set of words and asks them what comes to mind when they hear each word.
- **Sentence Completion :** Researchers use sentence completion to understand what kind of ideas the respondent has. This tool involves giving an incomplete sentence and seeing how the interviewee finishes it.
- **Role-Playing :** Respondents are presented with an imaginary situation and asked how they would act or react if it was real.
- **In-Person Surveys :** The researcher asks questions in person.
- **Online / Web Surveys :** These surveys are easy to accomplish, but some users may be unwilling to answer truthfully, if at all.
- **Mobile Surveys :** These surveys take advantage of the increasing proliferation of mobile technology. Mobile collection surveys rely on mobile devices like tablets or smartphones to conduct surveys via SMS or mobile apps.

Q.2 Define exploratory data analysis.

Ans. : Exploratory Data Analysis (EDA) is defined by John W. Tukey, is a process of examining or understanding the data and extracting insights or main characteristics of the data. EDA is generally classified into two methods, that is, **graphical** analysis and **non-graphical** analysis.

It is also known as **visual analytics or descriptive statistics**. It is the practice of inspecting and exploring data, before stating hypotheses, fitting predictors and other expected inferential goals. It typically includes the computation of simple summary statistics which capture some property of interest in the data and **visualization**. EDA can be thought of as an assumption-free, purely algorithmic practice.

Q.3 Explain multivariate graphical methods for data analysis.

Ans. : Multivariate data uses graphics to display relationships between two or more sets of data. The most used graphic is a grouped bar plot or bar chart with each group representing one level of one of the variables and each bar within a group representing the levels of the other variable.

Common types of multivariate graphics include,

- Scatter plot, which is used to plot data points on a horizontal and a vertical axis to show how much one variable is affected by another.
- Multivariate chart, which is a graphical representation of the relationships between factors and a response.
- Run chart, which is a line graph of data plotted over time.

Q.4 Write a note on box plots and bar plots.

Ans. : Box plots show the distribution of values along an axis. Rectangular boxes are used in order to bucket the data, giving us an idea of how the data points are spread out. These boxes are also called **quartiles** which represent a quarter of a data set. Boxes can be drawn vertically or horizontally. One can also easily spot the outliers, which are usually treated as abnormal values and affect the data set's overall observation due to their very high or low values. Box plots are suitable for identifying outliers.

Bar charts can be used to compare nominal or ordinal data. They are helpful for recognizing trends.

Q.5 Write a Python program to demonstrate use merge() function.

Ans. : Refer example program 1.

❑❑❑

TRANSFORMATION TECHNIQUES

Other types of data transformations including cleaning, filtering, deduplication, and others.

1. Performing data deduplication

It is very likely that your dataframe contains duplicate rows. Removing them is essential to enhance the quality of the dataset. This can be done with the following steps:

1. Let's consider a simple dataframe, as follows:

```
frame3 = pd.DataFrame({'column 1': ['Looping'] * 3 + ['Functions'] * 4,
'column 2': [10, 10, 22, 23, 23, 24, 24]})
frame3
```

The preceding code creates a simple dataframe with two columns. You can clearly see from the following screenshot that in both columns, there are some duplicate entries:

	column 1	column 2
0	Looping	10
1	Looping	10
2	Looping	22
3	Functions	23
4	Functions	23
5	Functions	24
6	Functions	24

2. The pandas dataframe comes with a duplicated() method that returns a Boolean series stating which of the rows are duplicates:

```
frame3.duplicated()
```

The output of the preceding code is pretty easy to interpret:

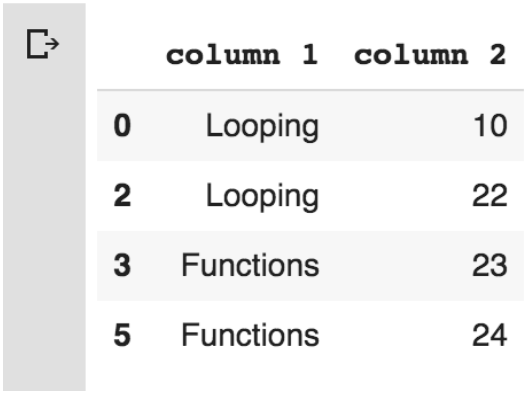
```
0    False
1     True
2    False
3    False
4     True
5    False
6     True
dtype: bool
```

The rows that say True are the ones that contain duplicated data.

3. Now, we can drop these duplicates using the drop_duplicates() method:

```
frame4 = frame3.drop_duplicates()
frame4
```

The output of the preceding code is as follows:



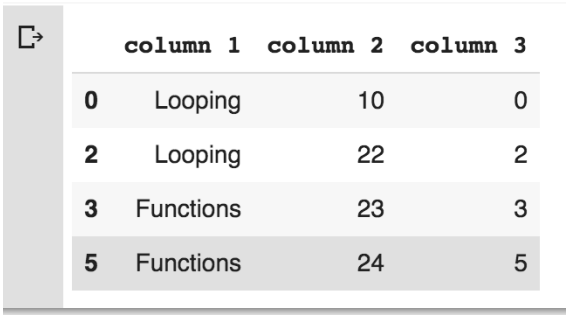
	column 1	column 2
0	Looping	10
2	Looping	22
3	Functions	23
5	Functions	24

Note that rows 1, 4, and 6 are removed. Basically, both the duplicated() and drop_duplicates() methods consider all of the columns for comparison. Instead of all the columns, we could specify any subset of the columns to detect duplicated items.

4. Let's add a new column and try to find duplicated items based on the second column:

```
frame3['column 3'] = range(7)
frame5 = frame3.drop_duplicates(['column 2'])
frame5
```

The output of the preceding snippet is as follows:



	column 1	column 2	column 3
0	Looping	10	0
2	Looping	22	2
3	Functions	23	3
5	Functions	24	5

Note that both the duplicated and drop duplicates methods keep the first observed value during the duplication removal process. If we pass the take_last=True argument, the methods return the last one.

2. Replacing values

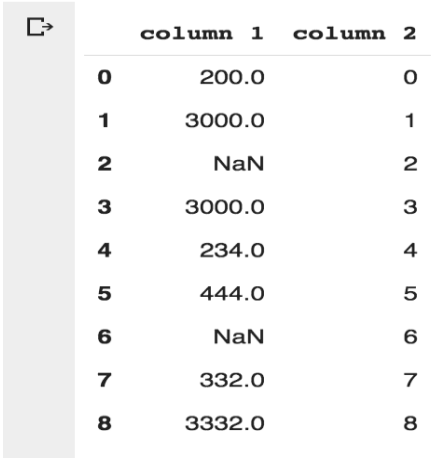
Often, it is essential to find and replace some values inside a dataframe. This can be done with the following steps:

1. We can use the replace method in such cases:

```
import numpy as np
replaceFrame = pd.DataFrame({'column 1': [200., 3000., -786., 3000.,
```

```
234., 444., -786., 332., 3332. ], 'column 2': range(9)})
replaceFrame.replace(to_replace =-786, value= np.nan)
```

The output of the preceding code is as follows:



	column 1	column 2
0	200.0	0
1	3000.0	1
2	NaN	2
3	3000.0	3
4	234.0	4
5	444.0	5
6	NaN	6
7	332.0	7
8	3332.0	8

Note that we just replaced one value with the other values. We can also replace multiple values at once.

2. In order to do so, we display them using a list:

```
replaceFrame = pd.DataFrame({'column 1': [200., 3000., -786., 3000.,
234., 444., -786., 332., 3332. ], 'column 2': range(9)})
replaceFrame.replace(to_replace =[-786, 0], value= [np.nan, 2])
```

In the preceding code, there are two replacements. All -786 values will be replaced by NaN and all 0 values will be replaced by 2. That's pretty straightforward, right?

3. Handling missing data

Whenever there are missing values, a NaN value is used, which indicates that there is no value specified for that particular index. There could be several reasons why a value could be NaN:

- It can happen when data is retrieved from an external source and there are some incomplete values in the dataset.
- It can also happen when we join two different datasets and some values are not matched.
- Missing values due to data collection errors.
- When the shape of data changes, there are new additional rows or columns that are not determined.
- Reindexing of data can result in incomplete data.

Let's see how we can work with the missing data:

1. Let's assume we have a dataframe as shown here:

```
data = np.arange(15, 30).reshape(5, 3)
dfx = pd.DataFrame(data, index=['apple', 'banana', 'kiwi', 'grapes',
```

```
'mango'], columns=['store1', 'store2', 'store3'])
dfx
```

And the output of the preceding code is as follows:

	store1	store2	store3
apple	15	16	17
banana	18	19	20
kiwi	21	22	23
grapes	24	25	26
mango	27	28	29

Assume we have a chain of fruit stores all over town. Currently, the dataframe is showing sales of different fruits from different stores. None of the stores are reporting missing values.

2. Let's add some missing values to our dataframe:

```
fx['store4'] = np.nan
dfx.loc['watermelon'] = np.arange(15, 19)
dfx.loc['oranges'] = np.nan
dfx['store5'] = np.nan
dfx['store4']['apple'] = 20.
dfx
```

And the output will now look like the following screenshot:

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

Note that we've added two more stores, store4 and store5, and two more types of fruits, watermelon and oranges. Assume that we know how many kilos of apples and watermelons were sold from store4, but we have not collected any data from store5. Moreover, none of the stores reported sales of oranges. We are quite a huge fruit dealer, aren't we?

Note the following characteristics of missing values in the preceding dataframe:

- An entire row can contain NaN values.
- An entire column can contain NaN values.
- Some (but not necessarily all) values in both a row and a column can be NaN.

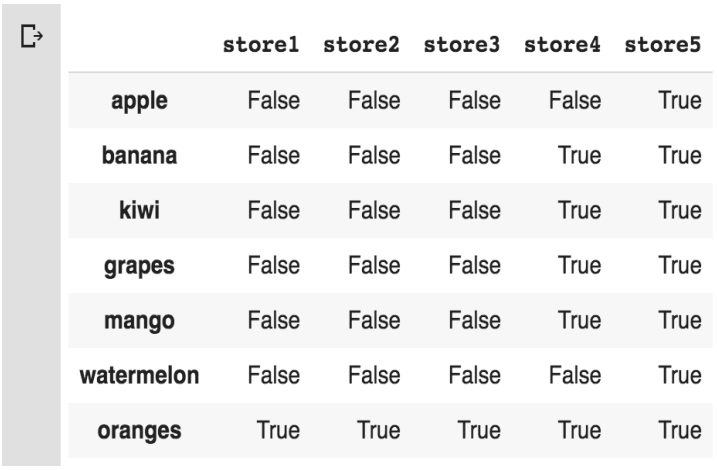
Based on these characteristics, let's examine NaN values in the next section.

NaN values in pandas objects

We can use the isnull() function from the pandas library to identify NaN values:

1. Check the following example:
   dfx.isnull()

The output of the preceding code is as follows:

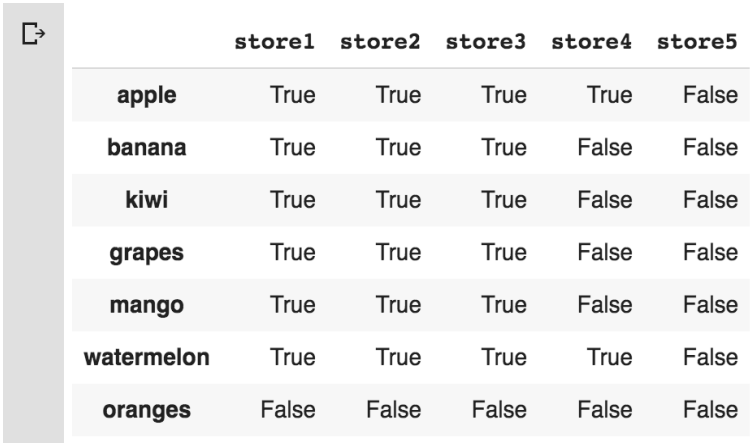


	store1	store2	store3	store4	store5
apple	False	False	False	False	True
banana	False	False	False	True	True
kiwi	False	False	False	True	True
grapes	False	False	False	True	True
mango	False	False	False	True	True
watermelon	False	False	False	False	True
oranges	True	True	True	True	True

Note that the True values indicate the values that are NaN. Alternatively, we can also use the notnull() method to do the same thing. The only difference would be that the function will indicate True for the values which are not null.

2. Check it out in action:
   dfx.notnull()

And the output of this is as follows:



	store1	store2	store3	store4	store5
apple	True	True	True	True	False
banana	True	True	True	False	False
kiwi	True	True	True	False	False
grapes	True	True	True	False	False
mango	True	True	True	False	False
watermelon	True	True	True	True	False
oranges	False	False	False	False	False

Compare these two tables. These two functions, notnull() and isnull(), are the complement to each other.

3. We can use the sum() method to count the number of NaN values in each store. How does this work, you ask? Check the following code:

```
dfx.isnull().sum()
```

And the **output** of the preceding code is as follows:

```
store1 1
store2 1
store3 1
store4 5
store5 7
dtype: int64
```

The fact that *True* is 1 and *False* is 0 is the main logic for summing. The preceding results show that one value was not reported by store1, store2, and store3. Five values were not reported by store4 and seven values were not reported by store5.

4. We can go one level deeper to find the total number of missing values:

```
dfx.isnull().sum().sum()
```

And the **output** of the preceding code is as follows:

```
15
```

This indicates 15 missing values in our stores. We can use an alternative way to find how many values were actually reported.

5. So, instead of counting the number of missing values, we can count the number of reported values:

```
dfx.count()
```

And the **output** of the preceding code is as follows:

```
store1 6
store2 6
store3 6
store4 2
store5 0
dtype: int64
```

4. Dropping missing values

One of the ways to handle missing values is to simply remove them from our dataset. We have seen that we can use the isnull() and notnull() functions from the pandas library to determine null values:

```
dfx.store4[dfx.store4.notnull()]
```

The **output** of the preceding code is as follows:

```
apple 20.0
watermelon 18.0
Name: store4, dtype: float64
```

The output shows that store4 only reported two items of data. Now, we can use the **dropna()** method to remove the rows:

```
dfx.store4.dropna()
```

The **output** of the preceding code is as follows:

```
apple 20.0
watermelon 18.0
Name: store4, dtype: float64
```

Note that the dropna() method just returns a copy of the dataframe by dropping the rows with NaN. The original dataframe is not changed.

If dropna() is applied to the entire dataframe, then it will drop all the rows from the dataframe, because there is at least one NaN value in our dataframe:

```
dfx.dropna()
```

The output of the preceding code is an empty dataframe.

We can also drop rows that have NaN values. To do so, we can use the `how=all` argument to drop only those rows entire values are entirely NaN:

```
dfx.dropna(how='all')
```

The output of the preceding code is as follows:

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN

Note that only the orange rows are removed because those entire rows contained NaN values.

5. Dropping by columns

Furthermore, we can also pass axis=1 to indicate a check for NaN by columns.

Check the following example:

```
dfx.dropna(how='all', axis=1)
```

And the output of the preceding code is as follows:

	store1	store2	store3	store4
apple	15.0	16.0	17.0	20.0
banana	18.0	19.0	20.0	NaN
kiwi	21.0	22.0	23.0	NaN
grapes	24.0	25.0	26.0	NaN
mango	27.0	28.0	29.0	NaN
watermelon	15.0	16.0	17.0	18.0
oranges	NaN	NaN	NaN	NaN

Note that `store5` is dropped from the dataframe. By passing in `axis=1`, we are instructing pandas to drop columns if all the values in the column are `NaN`. Furthermore, we can also pass another argument, `thresh`, to specify a minimum number of NaNs that must exist before the column should be dropped:

```
dfx.dropna(thresh=5, axis=1)
```

And the output of the preceding code is as follows:

	store1	store2	store3
apple	15.0	16.0	17.0
banana	18.0	19.0	20.0
kiwi	21.0	22.0	23.0
grapes	24.0	25.0	26.0
mango	27.0	28.0	29.0
watermelon	15.0	16.0	17.0
oranges	NaN	NaN	NaN

Compared to the preceding, note that even the store4 column is now dropped because it has more than five NaN values

6. Mathematical operations with NaN

The pandas and numpy libraries handle NaN values differently for mathematical operations.

Consider the following example:

```
ar1 = np.array([100, 200, np.nan, 300])
ser1 = pd.Series(ar1)
ar1.mean(), ser1.mean()
```

The **output** of the preceding code is the following:

```
(nan, 200.0)
```

Note the following things:

- When a NumPy function encounters NaN values, it returns NaN.
- Pandas, on the other hand, ignores the NaN values and moves ahead with processing. When performing the sum operation, NaN is treated as 0. If all the values are NaN, the result is also NaN.

Let's compute the total quantity of fruits sold by store4:

```
ser2 = dfx.store4
ser2.sum()
```

The **output** of the preceding code is as follows:

```
38.0
```

Note that store4 has five NaN values. However, during the summing process, these values are treated as 0 and the result is 38.0.

Similarly, we can compute averages as shown here:

```
ser2.mean()
```

The **output** of the code is the following: 19.0

Note that NaNs are treated as 0s. It is the same for cumulative summing:

```
ser2.cumsum()
```

And the **output** of the preceding code is as follows:

```
apple 20.0
banana NaN
kiwi NaN
grapes NaN
mango NaN
watermelon 38.0
oranges NaN
Name: store4, dtype: float64
```

Note that only actual values are affected in computing the cumulative sum.

7. Filling missing values

We can use the fillna() method to replace NaN values with any particular values.

Check the following example:

```
filledDf = dfx.fillna(0)
filledDf
```

The **output** of the preceding code is shown in the following screenshot:

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	0.0
banana	18.0	19.0	20.0	0.0	0.0
kiwi	21.0	22.0	23.0	0.0	0.0
grapes	24.0	25.0	26.0	0.0	0.0
mango	27.0	28.0	29.0	0.0	0.0
watermelon	15.0	16.0	17.0	18.0	0.0
oranges	0.0	0.0	0.0	0.0	0.0

Note that in the preceding dataframe, all the `NaN` values are replaced by `0`. Replacing the values with `0` will affect several statistics including mean, sum, and median.

Check the difference in the following two examples:

```
dfx.mean()
```

And the output of the preceding code is as follows:

```
store1 20.0
store2 21.0
store3 22.0
store4 19.0
store5 NaN
dtype: float64
```

Now, let's compute the mean from the filled dataframe with the following command:

```
filledDf.mean()
```

And the **output** we get is as follows:

```
store1 17.142857
store2 18.000000
store3 18.857143
store4 5.428571
store5 0.000000
dtype: float64
```

Note that there are slightly different values. Hence, filling with `0` might not be the optimal solution.

8. Backward and forward filling

NaN values can be filled based on the last known values. To understand this, let's consider taking our store dataframe as an example.

We want to fill store4 using the forward-filling technique:

```
dfx.store4.fillna(method='ffill')
```

And the **output** of the preceding code is the following:

```
apple 20.0
banana 20.0
kiwi 20.0
grapes 20.0
mango 20.0
watermelon 18.0
oranges 18.0
Name: store4, dtype: float64
```

Here, from the forward-filling technique, the last known value is 20 and hence the rest of the NaN values are replaced by it.

The direction of the fill can be changed by changing method='bfill'. Check the following example:

```
dfx.store4.fillna(method='bfill')
```

And the **output** of the preceding code is as follows:

```
apple 20.0
banana 18.0
kiwi 18.0
grapes 18.0
mango 18.0
watermelon 18.0
oranges NaN
Name: store4, dtype: float64
```

Note here that the NaN values are replaced by 18.0.

9. Interpolating missing values

The pandas library provides the interpolate() function both for the series and the dataframe. By default, it performs a linear interpolation of our missing values. Check the following example:

```
ser3 = pd.Series([100, np.nan, np.nan, np.nan, 292])
ser3.interpolate()
```

And the output of the preceding code is the following:

```
0 100.0
1 148.0
2 196.0
3 244.0
4 292.0
dtype: float64
```

The first value before and after any sequence of the NaN values. In the preceding series, ser3, the first and the last values are 100 and 292 respectively. Hence, it calculates the next value as *(292-100)/(5-1) = 48*. So, the next value after 100 is *100 + 48 = 148*.

10. Renaming axis indexes

Consider the example from the *Reshaping and pivoting* section. Say you want to transform the index terms to capital letters:

```
dframe1.index = dframe1.index.map(str.upper)
dframe1
```

The **output** of the preceding code is as follows:

	Bergen	Oslo	Trondheim	Stavanger	Kristiansand
RAINFALL	0	1	2	3	4
HUMIDITY	5	6	7	8	9
WIND	10	11	12	13	14

Note that the indexes have been capitalized. If we want to create a transformed version of the dataframe, then we can use the `rename()` method. This method is handy when we do not want to modify the original data. Check the following example:

```
dframe1.rename(index=str.title, columns=str.upper)
```

And the **output** of the code is as follows:

	BERGEN	OSLO	TRONDHEIM	STAVANGER	KRISTIANSAND
Rainfall	0	1	2	3	4
Humidity	5	6	7	8	9
Wind	10	11	12	13	14

The rename method does not make a copy of the dataframe.

11. Discretization and binning

Often when working with continuous datasets, we need to convert them into discrete or interval forms. Each interval is referred to as a bin, and hence the name *binning* comes into play:

1. Let's say we have data on the heights of a group of students as follows:

```
height = [120, 122, 125, 127, 121, 123, 137, 131, 161, 145, 141, 132]
```

And we want to convert that dataset into intervals of 118 to 125, 126 to 135, 136 to 160, and finally 160 and higher.

2. To convert the preceding dataset into intervals, we can use the cut() method provided by the pandas library:

```
bins = [118, 125, 135, 160, 200]
category = pd.cut(height, bins)
category
```

The **output** of the preceding code is as follows:

```
[(118, 125], (118, 125], (118, 125], (125, 135], (118, 125], ..., (125, 135], (160,
200], (135, 160], (135, 160], (125, 135]] Length: 12 Categories (4,
interval[int64]): [(118, 125] < (125, 135] < (135, 160] < (160, 200]]
```

If you look closely at the output, you'll see that there are mathematical notations for intervals. Do you recall what these parentheses mean from your elementary mathematics class? If not, here is a quick recap:

- A parenthesis indicates that the side is open.
- A square bracket means that it is closed or inclusive.

From the preceding code block, `(118, 125]` means the left-hand side is open and the right-hand side is closed. This is mathematically denoted as follows:

	Bergen	Oslo	Trondheim	Stavanger	Kristiansand
RAINFALL	0	1	2	3	4
HUMIDITY	5	6	7	8	9
WIND	10	11	12	13	14

Hence, `118` is not included, but anything greater than `118` is included, while `125` is included in the interval.

3. We can set a right=False argument to change the form of interval:

```
category2 = pd.cut(height, [118, 126, 136, 161, 200], right=False)
category2
```

And the **output** of the preceding code is as follows:

[[118, 126), [118, 126), [118, 126), [126, 136), [118, 126), ..., [126, 136), [161, 200), [136, 161), [136, 161), [126, 136)] Length: 12 Categories (4, interval[int64]): [[118, 126) < [126, 136) < [136, 161) < [161, 200)]

Note that the output form of closeness has been changed. Now, the results are in the form of *right-closed, left-open*.

4. We can check the number of values in each bin by using the pd.value_counts() method:

```
pd.value_counts(category)
```

And the **output** is as follows:

```
(118, 125] 5
(135, 160] 3
(125, 135] 3
(160, 200] 1
dtype: int64
```

The output shows that there are five values in the interval `[118-125)`.

5. We can also indicate the bin names by passing a list of labels:

```
bin_names = ['Short Height', 'Average height', 'Good Height', 'Taller']
pd.cut(height, bins, labels=bin_names)
```

And the **output** is as follows:

```
[Short Height, Short Height, Short Height, Average height, Short Height, ..., Average height, Taller, Good Height, Good Height, Average height]
Length: 12
Categories (4, object): [Short Height < Average height < Good Height < Taller]
```

Note that we have passed at least two arguments, the data that needs to be discretized and the required number of bins. Furthermore, we have used a `right=False` argument to change the form of interval.

6. Now, it is essential to note that if we pass just an integer for our bins, it will compute equal-length bins based on the minimum and maximum values in the data. Okay, let's verify what we mentioned here:

```
import numpy as np
pd.cut(np.random.rand(40), 5, precision=2)
```

In the preceding code, we have just passed `5` as the number of required bins, and the **output** of the preceding code is as follows:

```
[(0.81, 0.99], (0.094, 0.27], (0.81, 0.99], (0.45, 0.63], (0.63, 0.81], ..., (0.81, 0.99], (0.45, 0.63], (0.45, 0.63], (0.81, 0.99], (0.81, 0.99]] Length: 40
Categories (5, interval[float64]): [(0.094, 0.27] < (0.27, 0.45] < (0.45, 0.63] < (0.63, 0.81] < (0.81, 0.99]]
```

Pandas provides a `qcut` method that forms the bins based on sample quantiles. Let's check this with an example:

```
randomNumbers = np.random.rand(2000)
category3 = pd.qcut(randomNumbers, 4) # cut into quartiles
category3
```

And the **output** of the preceding code is as follows:

```
[(0.77, 0.999], (0.261, 0.52], (0.261, 0.52], (-0.000565, 0.261], (-0.000565, 0.261], ..., (0.77, 0.999], (0.77, 0.999], (0.261, 0.52], (-0.000565, 0.261], (0.261, 0.52]]
Length: 2000
Categories (4, interval[float64]): [(-0.000565, 0.261] < (0.261, 0.52] < (0.52, 0.77] < (0.77, 0.999]]
```

Note that based on the number of bins, which we set to 4, it converted our data into four different categories. If we count the number of values in each category, we should get equal-sized bins as per our definition. Let's verify that with the following command:

```
pd.value_counts(category3)
```

And the **output** of the command is as follows:

```
0.77, 0.999] 500
(0.52, 0.77] 500
(0.261, 0.52] 500
(-0.000565, 0.261] 500
dtype: int64
```

Our claim is hence verified. Each category contains an equal size of 500 values. Note that, similar to cut, we can also pass our own bins:

```
pd.qcut(randomNumbers, [0, 0.3, 0.5, 0.7, 1.0])
```

And the **output** of the preceding code is as follows:

```
[(0.722, 0.999], (-0.000565, 0.309], (0.309, 0.52], (-0.000565, 0.309], (-0.000565, 0.309], ..., (0.722, 0.999], (0.722, 0.999], (0.309, 0.52], (-0.000565, 0.309], (0.309, 0.52]] Length: 2000
Categories (4, interval[float64]): [(-0.000565, 0.309] < (0.309, 0.52] < (0.52, 0.722] < (0.722, 0.999]]
```

Note that it created four different categories based on our code. Congratulations! We successfully learned how to convert continuous datasets into discrete datasets.

12. Outlier detection and filtering

Outliers are data points that diverge from other observations for several reasons. During the EDA phase, one of our common tasks is to detect and filter these outliers. The main reason for this detection and filtering of outliers is that the presence of such outliers can cause serious

issues in statistical analysis. In this section, we are going to perform simple outlier detection and filtering. Let's get started:

1. Load the dataset that is available from the GitHub link as follows:

   ```
   df = pd.read_csv('https://raw.githubusercontent.com/PacktPublishing/hands-on-exploratory-data-analysis-with-python/master/Chapter%204/sales.csv')
   df.head(10)
   ```

The dataset was synthesized manually by creating a script. If you are interested in looking at how we created the dataset, the script can be found inside the folder named Chapter 4 in the GitHub repository shared with this book.

The output of the preceding df.head(10) command is shown in the following screenshot:

	Account	Company	Order	SKU	Country	Year	Quantity	UnitPrice	transactionComplete
0	123456779	Kulas Inc	99985	s9-supercomputer	Aruba	1981	5148	545	False
1	123456784	GitHub	99986	s4-supercomputer	Brazil	2001	3262	383	False
2	123456782	Kulas Inc	99990	s10-supercomputer	Montserrat	1973	9119	407	True
3	123456783	My SQ Man	99999	s1-supercomputer	El Salvador	2015	3097	615	False
4	123456787	ABC Dogma	99996	s6-supercomputer	Poland	1970	3356	91	True
5	123456778	Super Sexy Dingo	99996	s9-supercomputer	Costa Rica	2004	2474	136	True
6	123456783	ABC Dogma	99981	s11-supercomputer	Spain	2006	4081	195	False
7	123456785	ABC Dogma	99998	s9-supercomputer	Belarus	2015	6576	603	False
8	123456778	Loolo INC	99997	s8-supercomputer	Mauritius	1999	2460	36	False
9	123456775	Kulas Inc	99997	s7-supercomputer	French Guiana	2004	1831	664	True

2. Now, suppose we want to calculate the total price based on the quantity sold and the unit price. We can simply add a new column, as shown here:

   ```
   df['TotalPrice'] = df['UnitPrice'] * df['Quantity']
   df
   ```

This should add a new column called TotalPrice, as shown in the following screenshot:

	Account	Company	Order	SKU	Country	Year	Quantity	UnitPrice	transactionComplete	TotalPrice
0	123456779	Kulas Inc	99985	s9-supercomputer	Aruba	1981	5148	545	False	2805660
1	123456784	GitHub	99986	s4-supercomputer	Brazil	2001	3262	383	False	1249346
2	123456782	Kulas Inc	99990	s10-supercomputer	Montserrat	1973	9119	407	True	3711433
3	123456783	My SQ Man	99999	s1-supercomputer	El Salvador	2015	3097	615	False	1904655
4	123456787	ABC Dogma	99996	s6-supercomputer	Poland	1970	3356	91	True	305396
5	123456778	Super Sexy Dingo	99996	s9-supercomputer	Costa Rica	2004	2474	136	True	336464
6	123456783	ABC Dogma	99981	s11-supercomputer	Spain	2006	4081	195	False	795795
7	123456785	ABC Dogma	99998	s9-supercomputer	Belarus	2015	6576	603	False	3965328
8	123456778	Loolo INC	99997	s8-supercomputer	Mauritius	1999	2460	36	False	88560
9	123456775	Kulas Inc	99997	s7-supercomputer	French Guiana	2004	1831	664	True	1215784

Now, let's answer some questions based on the preceding table.

Let's find the transaction that exceeded 3,000,000:

```
TotalTransaction = df["TotalPrice"]
TotalTransaction[np.abs(TotalTransaction) > 3000000]
```

The **output** of the preceding code is as follows:

```
2 3711433
7 3965328
13 4758900
15 5189372
17 3989325
      ...
9977 3475824
9984 5251134
9987 5670420
9991 5735513
9996 3018490
Name: TotalPrice, Length: 2094, dtype: int64
```

Note that, in the preceding example, we have assumed that any price greater than 3,000,000 is an outlier.

Display all the columns and rows from the preceding table if TotalPrice is greater than 6741112, as follows:

```
df[np.abs(TotalTransaction) > 6741112]
```

The **output** of the preceding code is the following:

	Account	Company	Order	SKU	Country	Year	Quantity	UnitPrice	transactionComplete	TotalPrice
818	123456781	Gen Power	99991	s1-supercomputer	Burkina Faso	1985	9693	696	False	6746328
1402	123456778	Will LLC	99985	s11-supercomputer	Austria	1990	9844	695	True	6841580
2242	123456770	Name IT	99997	s9-supercomputer	Myanmar	1979	9804	692	False	6784368
2876	123456772	Gen Power	99992	s10-supercomputer	Mali	2007	9935	679	False	6745865
3210	123456782	Loolo INC	99991	s8-supercomputer	Kuwait	2006	9886	692	False	6841112
3629	123456779	My SQ Man	99980	s3-supercomputer	Hong Kong	1994	9694	700	False	6785800
7674	123456781	Loolo INC	99989	s6-supercomputer	Sri Lanka	1994	9882	691	False	6828462
8645	123456789	Gen Power	99996	s11-supercomputer	Suriname	2005	9742	699	False	6809658
8684	123456785	Gen Power	99989	s2-supercomputer	Kenya	2013	9805	694	False	6804670

Note that in the output, all the TotalPrice values are greater than 6741112. We can use any sort of conditions, either row-wise or column-wise, to detect and filter outliers.

13. Permutation and random sampling

Well, now we have some more mathematical terms to learn: *permutation* and *random sampling*. Let's examine how we can perform permutation and random sampling using the pandas library:

1. With NumPy's numpy.random.permutation() function, we can randomly select or permute a series of rows in a dataframe. Let's understand this with an example:

```
dat = np.arange(80).reshape(10,8)
df = pd.DataFrame(dat)
df
```

And the output of the preceding code is as follows:

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
3	24	25	26	27	28	29	30	31
4	32	33	34	35	36	37	38	39
5	40	41	42	43	44	45	46	47
6	48	49	50	51	52	53	54	55
7	56	57	58	59	60	61	62	63
8	64	65	66	67	68	69	70	71
9	72	73	74	75	76	77	78	79

2. Next, we call the np.random.permutation() method. This method takes an argument – the length of the axis we require to be permuted – and gives an array of integers indicating the new ordering:

```
sampler = np.random.permutation(10)
sampler
```

The **output** of the preceding code is as follows:

```
array([1, 5, 3, 6, 2, 4, 9, 0, 7, 8])
```

3. The preceding output array is used in ix-based indexing for the take() function from the pandas library. Check the following example for clarification:

```
df.take(sampler)
```

The **output** of the preceding code is as follows:

	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
5	40	41	42	43	44	45	46	47
3	24	25	26	27	28	29	30	31
6	48	49	50	51	52	53	54	55
2	16	17	18	19	20	21	22	23
4	32	33	34	35	36	37	38	39
9	72	73	74	75	76	77	78	79
0	0	1	2	3	4	5	6	7
7	56	57	58	59	60	61	62	63
8	64	65	66	67	68	69	70	71

It is essential that you understand the output. Note that our sampler array contains array([1, 5, 3, 6, 2, 4, 9, 0, 7, 8]). Each of these array items represents the rows of the original dataframe. So, from the original dataframe, it pulls the first row, then the fifth row, then the third row, and so on. Compare this with the original dataframe output and it will make more sense.

Random sampling without replacement

To compute random sampling without replacement, follow these steps:

1. To perform random sampling without replacement, we first create a permutation array.
2. Next, we slice off the first *n* elements of the array where *n* is the desired size of the subset you want to sample.
3. Then we use the df.take() method to obtain actual samples:

```
df.take(np.random.permutation(len(df))[:3])
```

The **output** of the preceding code is as follows:

	0	1	2	3	4	5	6	7
9	72	73	74	75	76	77	78	79
2	16	17	18	19	20	21	22	23
0	0	1	2	3	4	5	6	7

Note that in the preceding code, we only specified a sample of size 3. Hence, we only get three rows in the random sample.

Random sampling with replacement

To generate random sampling with replacement, follow the given steps:

1. We can generate a random sample with replacement using the numpy.random.randint() method and drawing random integers:

```
sack = np.array([4, 8, -2, 7, 5])
sampler = np.random.randint(0, len(sack), size = 10)
sampler
```

We created the sampler using the np.random.randint() method. The **output** of the preceding code is as follows:

```
array([3, 3, 0, 4, 0, 0, 1, 2, 1, 4])
```

2. And now, we can draw the required samples:

```
draw = sack.take(sampler)
draw
```

The **output** of the preceding code is as follows:

```
array([ 7,  7,  4,  5,  4,  4,  8, -2,  8,  5])
```

Compare the index of the sampler and then compare it with the original dataframe. The results are pretty obvious in this case.

14. Computing indicators/dummy variables

Often, we need to convert a categorical variable into some dummy matrix. Especially for statistical modeling or machine learning model development, it is essential to create dummy variables. Let's get started:

1. Let's say we have a dataframe with data on gender and votes, as shown here:

```
df = pd.DataFrame({'gender': ['female', 'female', 'male', 'unknown', 'male',
'female'], 'votes': range(6, 12, 1)})
df
```

The output of the preceding code is as follows:

	gender	votes
0	female	6
1	female	7
2	male	8
3	unknown	9
4	male	10
5	female	11

So far, nothing too complicated. Sometimes, however, we need to encode these values in a matrix form with 1 and 0 values.

2. We can do that using the pd.get_dummies() function:
   pd.get_dummies(df['gender'])

And the **output** of the preceding code is as follows:

	female	male	unknown
0	1	0	0
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0
5	1	0	0

Note the pattern. There are five values in the original dataframe with three unique values of male, female, and unknown. Each unique value is transformed into a column and each original value into a row. For example, in the original dataframe, the first value is female, hence it is added as a row with 1 in the female value and the rest of them are 0 values, and so on.

3. Sometimes, we want to add a prefix to the columns. We can do that by adding the prefix argument, as shown here:
   dummies = pd.get_dummies(df['gender'], prefix='gender')
   dummies

The **output** of the preceding code is as follows:

	gender_female	gender_male	gender_unknown
0	1	0	0
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0
5	1	0	0

Note the gender prefix added to each of the column names. Not that difficult, right? Great work so far.

Benefits of data transformation

- Data transformation promotes interoperability between several applications. The main reason for creating a similar format and structure in the dataset is that it becomes compatible with other systems.
- Comprehensibility for both humans and computers is improved when using better-organized data compared to messier data.
- Data transformation ensures a higher degree of data quality and protects applications from several computational challenges such as null values, unexpected duplicates, and incorrect indexings, as well as incompatible structures or formats.
- Data transformation ensures higher performance and scalability for modern analytical databases and dataframes.

Challenges

- It requires a qualified team of experts and state-of-the-art infrastructure. The cost of attaining such experts and infrastructure can increase the **cost of the operation**.
- Data transformation requires data cleaning before data transformation and data migration. This process of cleansing can be expensively **time-consuming**.
- Generally, the activities of data transformations involve batch processing. This means that sometimes, we might have to wait for a day before the next batch of data is ready for cleansing. This can be very **slow**.

GROUPING DATASETS

- During data analysis, it is often essential to cluster or group data together based on certain criteria. For example, an e-commerce store might want to group all the sales that were done during the Christmas period or the orders that were received on Black Friday.
- These grouping concepts occur in several parts of data analysis.
- Different groupby() mechanics that will accumulate our dataset into various classes that we can perform aggregation on.

In this chapter, we will cover the following topics:

- Understanding groupby()
- Groupby mechanics
- Data aggregation
- Pivot tables and cross-tabulations

Technical requirements

- The code for this chapter can be found in this book's GitHub repository, https://github.com/PacktPublishing/hands-on-exploratory-data-analysis-with-python
- The dataset we'll be using in this chapter is available under open access through Kaggle. It can be downloaded from https://www.kaggle.com/toramky/automobile-dataset.

Understanding groupby()

- During the data analysis phase, categorizing a dataset into multiple categories or groups is often essential. We can do such categorization using the pandas library.
- The pandas groupby function is one of the most efficient and time-saving features for doing this. Groupby provides functionalities that allow us to split-apply-combine throughout the dataframe;
- that is, this function can be used for splitting, applying, and combining dataframes.
- Similar to the **Structured Query Language** (**SQL**), we can use pandas and Python to execute more complex group operations by using any built-in functions that accept the pandas object or the numpy array.

Groupby mechanics

While working with the pandas dataframes, our analysis may require us to split our data by certain criteria. Groupby mechanics amass our dataset into various classes in which we can perform exercises and make changes, such as the following:

- Grouping by features, hierarchically
- Aggregating a dataset by groups
- Applying custom aggregation functions to groups
- Transforming a dataset groupwise

The pandas groupby method performs two essential functions:

- It splits the data into groups based on some criteria.
- It applies a function to each group independently.

To work with groupby functionalities, we need a dataset that has multiple numerical as well as categorical records in it so that we can group by different categories and ranges.

1. Let's start by importing the required Python libraries and datasets:

```
import pandas as pd
df = pd.read_csv("/content/automobileEDA.csv")
df.head()
```

The **output** of the preceding code is as follows:

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	0.909722	52.4	2823	ohcv	six	152
3	2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	0.919444	54.3	2337	ohc	four	109
4	2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	0.922222	54.3	2824	ohc	five	136

As you can see, there are multiple columns with categorical variables.

2. Using the groupby() function lets us group this dataset on the basis of the body-style column:

```
df.groupby('body-style').groups.keys()
```

The **output** of the preceding code is as follows:

```
dict_keys(['convertible', 'hardtop', 'hatchback', 'sedan', 'wagon'])
```

From the preceding output, we know that the body-style column has five unique values, including convertible, hardtop, hatchback, sedan, and wagon.

3. Now, we can group the data based on the body-style column. Next, let's print the values contained in that group that have the body-style value of convertible. This can be done using the following code:

```
# Group the dataset by the column body-style
style = df.groupby('body-style')

# Get values items from group with value convertible
style.get_group("convertible")
```

The **output** of the preceding code is as follows:

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130
69	3	142	mercedes-benz	std	two	convertible	rwd	front	96.6	0.866410	0.979167	50.8	3685	ohcv	eight	234
125	3	122	porsche	std	two	convertible	rwd	rear	89.5	0.811629	0.902778	51.6	2800	ohcf	six	194
168	2	134	toyota	std	two	convertible	rwd	front	98.4	0.846708	0.911111	53.0	2975	ohc	four	146
185	3	122	volkswagen	std	two	convertible	fwd	front	94.5	0.765497	0.891667	55.6	2254	ohc	four	109

In the preceding example, we have grouped by using a single body-style column.

Selecting a subset of columns

To form groups based on multiple categories, we can simply specify the column names in the groupby() function. Grouping will be done simultaneously with the first category, the second category, and so on.

Let's groupby using two categories, body-style and drive wheels, as follows:

```
double_grouping = df.groupby(["body-style","drive-wheels"])
double_grouping.first()
```

The **output** of the preceding code is as follows:

body-style	drive-wheels	symboling	normalized-losses	make	aspiration	num-of-doors	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size	fue syst
convertible	fwd	3	122	volkswagen	std	two	front	94.5	0.765497	0.891667	55.6	2254	ohc	four	109	m
	rwd	3	122	alfa-romero	std	two	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130	m
hardtop	fwd	2	168	nissan	std	two	front	95.1	0.780394	0.886111	53.3	2008	ohc	four	97	2b
	rwd	0	93	mercedes-benz	turbo	two	front	106.7	0.901009	0.976389	54.9	3495	ohc	five	183	
hatchback	4wd	2	83	subaru	std	two	front	93.3	0.755887	0.886111	55.7	2240	ohcf	four	108	2b
	fwd	2	121	chevrolet	std	two	front	88.4	0.678039	0.837500	53.2	1488	l	three	61	2b
	rwd	1	122	alfa-romero	std	two	front	94.5	0.822681	0.909722	52.4	2823	ohcv	six	152	m
sedan	4wd	2	164	audi	std	four	front	99.4	0.848630	0.922222	54.3	2824	ohc	five	136	m
	fwd	2	164	audi	std	four	front	99.8	0.848630	0.919444	54.3	2337	ohc	four	109	m
	rwd	2	192	bmw	std	two	front	101.2	0.849592	0.900000	54.3	2395	ohc	four	108	m
wagon	4wd	0	85	subaru	std	four	front	96.9	0.834214	0.908333	54.9	2420	ohcf	four	108	2b
	fwd	1	122	audi	std	four	front	105.8	0.925997	0.991667	55.7	2954	ohc	five	136	m
	rwd	-1	93	mercedes-benz	turbo	four	front	110.0	0.917347	0.976389	58.7	3750	ohc	five	183	

Not only can we group the dataset with specific criteria, but we can also perform arithmetic operations directly on the whole group at the same time and print the output as a series or dataframe.

There are functions such as max(), min(), mean(), first(), and last() that can be directly applied to the GroupBy object in order to obtain summary statistics for each group.

Max and min

Let's compute the maximum and minimum entry for each group. Here, we will find the maximum and minimum for the normalized-losses column:

```
# max() will print the maximum entry of each group
style['normalized-losses'].max()
```

```
# min() will print the minimum entry of each group
style['normalized-losses'].min()
```

The **output** of the preceding code is as follows:

```
body-style
convertible 122
hardtop 93
hatchback 65
sedan 65
wagon 74
Name: normalized-losses, dtype: int64
```

As illustrated in the preceding output, the minimum value for each category is presented.

Mean

We can find the mean values for the numerical column in each group. This can be done using the df.mean() method.

The code for finding the mean is as follows:

```
# mean() will print mean of numerical column in each group
style.mean()
```

The **output** of the preceding code is as follows:

body-style	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower
convertible	2.833333	127.333333	92.700000	0.818757	0.910880	51.433333	2801.666667	157.166667	3.491667	3.043333	8.933333	131.666667
hardtop	1.875000	128.625000	98.500000	0.850252	0.925174	52.850000	2810.625000	176.250000	3.608750	3.322500	10.725000	142.250000
hatchback	1.617647	130.897059	95.435294	0.799078	0.904228	52.133824	2322.852941	112.852941	3.236015	3.280312	9.042941	97.768473
sedan	0.329787	120.893617	100.750000	0.855583	0.921070	54.387234	2625.893617	131.691489	3.345106	3.270638	10.965957	103.808511
wagon	-0.160000	98.560000	102.156000	0.871235	0.920222	56.728000	2784.240000	123.840000	3.406400	3.175600	10.316000	98.010246

get the average of each column by specifying a column, as follows:

```
# get mean of each column of specific group
style.get_group("convertible").mean()
```

The output of the preceding code is as follows:

```
symboling               2.833333
normalized-losses     127.333333
wheel-base             92.700000
length                  0.818757
width                   0.910880
height                 51.433333
curb-weight          2801.666667
engine-size           157.166667
bore                    3.491667
stroke                  3.043333
compression-ratio       8.933333
horsepower            131.666667
peak-rpm             5158.333333
city-mpg               20.500000
highway-mpg            26.000000
price               21890.500000
city-L/100km           11.745886
diesel                  0.000000
gas                     1.000000
dtype: float64
```

Next, we can also **count the number of symboling/records** in each group. To do so, use the following code:

```
# get the number of symboling/records in each group
style['symboling'].count()
```

The **output** of the preceding code is as follows:

```
body-style
convertible 6
hardtop 8
hatchback 68
sedan 94
wagon 25
Name: symboling, dtype: int64
```

DATA AGGREGATION

Aggregation is the process of implementing any mathematical operation on a dataset or a subset of it. Aggregation is one of the many techniques in pandas that's used to manipulate the data in the dataframe for data analysis.

The Dataframe.aggregate() function is used to apply aggregation across one or more columns. Some of the most frequently used aggregations are as follows:

- sum: Returns the sum of the values for the requested axis
- min: Returns the minimum of the values for the requested axis
- max: Returns the maximum of the values for the requested axis

We can apply aggregation in a DataFrame, df, as df.aggregate() or df.agg().

Since aggregation only works with numeric type columns, let's take some of the numeric columns from the dataset and apply some aggregation functions to them:

```
# new dataframe that consist length,width,height,curb-weight and price
new_dataset = df.filter(["length","width","height","curb-weight","price"],axis=1)
new_dataset
```

The **output** of the preceding code snippet is as follows:

	length	width	height	curb-weight	price
0	0.811148	0.890278	48.8	2548	13495.0
1	0.811148	0.890278	48.8	2548	16500.0
2	0.822681	0.909722	52.4	2823	16500.0
3	0.848630	0.919444	54.3	2337	13950.0
4	0.848630	0.922222	54.3	2824	17450.0
...	...	...	...	...	...
196	0.907256	0.956944	55.5	2952	16845.0
197	0.907256	0.955556	55.5	3049	19045.0
198	0.907256	0.956944	55.5	3012	21485.0
199	0.907256	0.956944	55.5	3217	22470.0
200	0.907256	0.956944	55.5	3062	22625.0

201 rows × 5 columns

Next, let's apply a single aggregation to get the mean of the columns. To do this, we can use the agg() method, as shown in the following code:

```
# applying single aggregation for mean over the columns
new_dataset.agg("mean", axis="rows")
```

The **output** of the preceding code is as follows:

```
length 0.837102
width 0.915126
height 53.766667
curb-weight 2555.666667
price 13207.129353
dtype: float64
```

We can aggregate more than one function together. For example, we can find the sum and the minimum of all the columns at once by using the following code:

```
# applying aggregation sum and minimum across all the columns
new_dataset.agg(['sum', 'min'])
```

The **output** of the preceding code is as follows:

	length	width	height	curb-weight	price
sum	168.257568	183.940278	10807.1	513689	2654633.0
min	0.678039	0.837500	47.8	1488	5118.0

The output is a dataframe with rows containing the result of the respective aggregation that was applied to the columns. To apply aggregation functions across different columns, you can pass a dictionary with a key containing the column names and values containing the list of aggregation functions for any specific column:

```
# find aggregation for these columns
new_dataset.aggregate({"length":['sum', 'min'],
        "width":['max', 'min'],
        "height":['min', 'sum'],
        "curb-weight":['sum']})
# if any specific aggregation is not applied on a column
# then it has NaN value corresponding to it
```

The **output** of the preceding code is as follows:

	length	width	height	curb-weight
max	NaN	1.0000	NaN	NaN
min	0.678039	0.8375	47.8	NaN
sum	168.257568	NaN	10807.1	513689.0

Check the preceding output. The maximum, minimum, and the sum of rows present the values for each column. Note that some values are NaN based on their column values.

Group-wise operations in data aggregation

- The most important operations groupBy implements are aggregate, filter, transform and apply.
- An efficient way of implementing aggregation functions in the dataset is by doing so after grouping the required columns.
- The aggregated function will return a single aggregated value for each group.
- Once these groups have been created, several aggregation operations are applied to that grouped data.
- It is possible to group the DataFrame, df, by passing a dictionary of aggregation functions:

```
# Group the data frame df by body-style and drive-wheels and extract stats
from each group
df.groupby( ["body-style","drive-wheels"]).agg(
   { 'height':min, # minimum height of car in each group
      'length': max, # maximum length of car in each group
      'price': 'mean', # average price of car in each group })
```

The **output** of the preceding code is as follows:

		height	length	price
body-style	drive-wheels			
convertible	fwd	55.6	0.765497	11595.000000
	rwd	48.8	0.866410	23949.600000
hardtop	fwd	53.3	0.780394	8249.000000
	rwd	51.6	0.957232	24202.714286
hatchback	4wd	55.7	0.755887	7603.000000
	fwd	49.4	0.896684	8396.387755
	rwd	49.6	0.881788	14337.777778
sedan	4wd	54.3	0.848630	12647.333333
	fwd	50.6	0.925997	9811.800000
	rwd	47.8	1.000000	21711.833333
wagon	4wd	54.9	0.834214	9095.750000
	fwd	53.0	0.925997	9997.333333
	rwd	54.1	0.955790	16994.222222

- The preceding code groups the dataframe according to body-style and then driver-wheels.
- The aggregate functions are applied to the height, length, and price columns, which return the minimum height, maximum length, and average price in the respective groups.
- We can make an aggregation dictionary of functions we want to perform in groups, and then use it later:

```
# create dictionary of aggregations
aggregations=({
        'height':min, # minimum height of car in each group
        'length': max, # maximum length of car in each group
        'price': 'mean', # average price of car in each group})
# implementing aggregations in groups
df.groupby(["body-style","drive-wheels"]).agg(aggregations)
```

The **output** of the preceding code is as follows:

		height	length	price
body-style	drive-wheels			
convertible	fwd	55.6	0.765497	11595.000000
	rwd	48.8	0.866410	23949.600000
hardtop	fwd	53.3	0.780394	8249.000000
	rwd	51.6	0.957232	24202.714286
hatchback	4wd	55.7	0.755887	7603.000000
	fwd	49.4	0.896684	8396.387755
	rwd	49.6	0.881788	14337.777778
sedan	4wd	54.3	0.848630	12647.333333
	fwd	50.6	0.925997	9811.800000
	rwd	47.8	1.000000	21711.833333
wagon	4wd	54.9	0.834214	9095.750000
	fwd	53.0	0.925997	9997.333333
	rwd	54.1	0.955790	16994.222222

We can use numpy functions in aggregation as well:

```
# import the numpy library as np
import numpy as np
# using numpy libraries for operations
df.groupby(["body-style","drive-wheels"])["price"].agg([np.sum, np.mean, np.std])
```

The **output** of the preceding code is as follows:

		sum	mean	std
body-style	drive-wheels			
convertible	fwd	11595.0	11595.000000	NaN
	rwd	119748.0	23949.600000	11165.099700
hardtop	fwd	8249.0	8249.000000	NaN
	rwd	169419.0	24202.714286	14493.311190
hatchback	4wd	7603.0	7603.000000	NaN
	fwd	411423.0	8396.387755	3004.675695
	rwd	258080.0	14337.777778	3831.795195
sedan	4wd	37942.0	12647.333333	4280.814681
	fwd	539649.0	9811.800000	3519.517598
	rwd	781626.0	21711.833333	9194.820239
wagon	4wd	36383.0	9095.750000	1775.652063
	fwd	119968.0	9997.333333	3584.185551
	rwd	152948.0	16994.222222	4686.703313

Renaming grouped aggregation columns

- We can perform aggregation in each group and rename the columns according to the operation performed.

- This is useful for understanding the output dataset:

```
df.groupby(["body-style","drive-wheels"]).agg(
    # Get max of the price column for each group
    max_price=('price', max),
    # Get min of the price column for each group
    min_price=('price', min),
    # Get sum of the price column for each group
    total_price=('price', 'mean') )
```

The **output** of the preceding code is as follows:

		max_price	min_price	total_price
body-style	drive-wheels			
convertible	fwd	11595.0	11595.0	11595.000000
	rwd	37028.0	13495.0	23949.600000
hardtop	fwd	8249.0	8249.0	8249.000000
	rwd	45400.0	8449.0	24202.714286
hatchback	4wd	7603.0	7603.0	7603.000000
	fwd	18150.0	5118.0	8396.387755
	rwd	22018.0	8238.0	14337.777778
sedan	4wd	17450.0	9233.0	12647.333333
	fwd	23875.0	5499.0	9811.800000
	rwd	41315.0	6785.0	21711.833333
wagon	4wd	11694.0	7898.0	9095.750000
	fwd	18920.0	6918.0	9997.333333
	rwd	28248.0	12440.0	16994.222222

- As shown in the preceding screenshot, we only selected two categories: body-style and drive-wheels. For each row in these categories, the maximum price, the minimum price, and the total price is computed in the successive columns.

PIVOT TABLES AND CROSS-TABULATIONS TECHNIQUES

- Pandas offer several options for grouping and summarizing data.
- Like groupby, aggregation, and transformation, there are other options available, such as pivot_table and crosstab.

Pivot tables

- The pandas.pivot_table() function creates a spreadsheet-style pivot table as a dataframe.
- The levels in the pivot table will be stored in MultiIndex objects (hierarchical indexes) on the index and columns of the resulting dataframe.
- The simplest pivot tables must have a dataframe and an index/listof the index.

1. Creating a pivot table of a new dataframe that consists of the body- style, drive-wheels, length, width, height, curb-weight, and price columns:

new_dataset1=df.filter(["body-style","drive wheels","length","width","height","curbweight","price"],axis=1)

#implest pivot table with dataframe df and index body-styletable = pd.pivot_table(new_dataset1, index =["body-style"]) table

Output:

body-style	curb-weight	height	length	price	width
convertible	2801.666667	51.433333	0.818757	24079.550000	0.910880
hardtop	2810.625000	52.850000	0.850252	24429.350000	0.925174
hatchback	2322.852941	52.133824	0.799078	10953.185294	0.904228
sedan	2625.893617	54.387234	0.855583	15905.730851	0.921070
wagon	2784.240000	56.728000	0.871235	13609.156000	0.920222

- The output table is similar to grouping a dataframe with respect to body-style.

2. The values in the preceding table are the mean of the values in thecorresponding category.Design a pivot table with the new_dataset1 dataframe and make body-style and drive-wheels as an index.
   - Providing multiple indexes will make a grouping of the dataframe first and then summarize the data

#pivot table with dataframe df and index body-style and drivewheels table = pd.pivot_table(new_dataset1, index =["body-style","drivewheels"])table

Output:

body-style	drive-wheels	curb-weight	height	length	price	width
convertible	fwd	2254.000000	55.600000	0.765497	12754.500000	0.891667
	rwd	2911.200000	50.600000	0.829409	26344.560000	0.914722
hardtop	fwd	2008.000000	53.300000	0.780394	9073.900000	0.886111
	rwd	2925.285714	52.785714	0.860232	26622.985714	0.930754
hatchback	4wd	2240.000000	55.700000	0.755887	8363.300000	0.886111
	fwd	2181.551020	52.442857	0.787818	9236.026531	0.898214
	rwd	2712.111111	51.094444	0.832132	15771.555556	0.921605
sedan	4wd	2573.000000	54.300000	0.833894	13912.066667	0.912963
	fwd	2313.018182	53.956364	0.828404	10792.980000	0.908182
	rwd	3108.305556	55.052778	0.898913	23883.016667	0.941435
wagon	4wd	2617.500000	57.000000	0.824844	10005.325000	0.895833
	fwd	2464.333333	56.008333	0.843064	10997.066667	0.910185
	rwd	3284.888889	57.566667	0.929414	18693.644444	0.944444

- The output is a pivot table grouped by body-style and drive- wheels.
- It contains the average of the numerical values of the corresponding columns.
- The syntax for the pivot table takes some arguments, such as c, values, index, column, and aggregation function.
- We can apply the aggregation function to a pivot table at the same time.
- We can pass the aggregation function, values, and columns that aggregation will be applied to, in order to create a pivot table of a summarized subset of a dataframe:

```
import numpy as np
new_dataset3 = df.filter(["body-style","drivewheels","price"],axis=1)
table = pd.pivot_table(new_dataset3, values='price', index=["bodystyle"],
columns=["drivewheels"],
aggfunc=np.mean,fill_value=0)table
```

In terms of syntax, the preceding code represents the following:

- A pivot table with a dataset called new_dataset3.
- The values are the columns that the aggregation function is to beapplied to.
- The index is a column for grouping data.
- Columns for specifying the category of data.
- aggfunc is the aggregation function to be applied.
- fill_value is used to fill in missing values.

Output:

drive-wheels	4wd	fwd	rwd
body-style			
convertible	0.000000	12754.500000	26344.560000
hardtop	0.000000	9073.900000	26622.985714
hatchback	8363.300000	9236.026531	15771.555556
sedan	13912.066667	10792.980000	23883.016667
wagon	10005.325000	10997.066667	18693.644444

- The preceding pivot table represents the average price of cars with different body-style and available drive-wheels in those body-style.

3. A different aggregation function can also be applied to differentcolumns:

```
table = pd.pivot_table(new_dataset1,values=['price','height','width'],
index =["body-style","drive-wheels"], aggfunc={'price': np.mean,'height':
[min,max],'width': [min, max]}, fill_value=0)
table
```

Output:

		height		price	width	
		max	min	mean	max	min
body-style	drive-wheels					
convertible	fwd	55.6	55.6	12754.500000	0.891667	0.891667
	rwd	53.0	48.8	26344.560000	0.979167	0.890278
hardtop	fwd	53.3	53.3	9073.900000	0.886111	0.886111
	rwd	55.4	51.6	26622.985714	1.000000	0.902778
hatchback	4wd	55.7	55.7	8363.300000	0.886111	0.886111
	fwd	56.1	49.4	9236.026531	0.925000	0.837500
	rwd	54.8	49.6	15771.555556	0.948611	0.888889
sedan	4wd	54.3	54.3	13912.066667	0.922222	0.908333
	fwd	56.1	50.6	10792.980000	0.991667	0.868056
	rwd	56.7	47.8	23883.016667	0.995833	0.858333
wagon	4wd	59.1	54.9	10005.325000	0.908333	0.883333
	fwd	59.8	53.0	10997.066667	0.991667	0.883333
	rwd	58.7	54.1	18693.644444	0.976389	0.923611

- This pivot table represents the maximum and minimum of the height and width and the average price of cars in the respective categories mentioned in the index.

Cross-tabulations

- The pandas dataframe can be customized with another techniquecalled cross-tabulation.
- This allows us to cope with groupby and aggregation for better dataanalysis.
- Pandas have the crosstab function, which helps to build a cross-tabulation table.
- The cross-tabulation table shows the frequency with which certaingroups of data appear.
- To identify how many different body styles cars are made by differentmakers, use pd.crosstab()

```
pd.crosstab(df[“make”], df[“body-style”])
```

Output

body-style make	convertible	hardtop	hatchback	sedan	wagon
alfa-romero	2	0	1	0	0
audi	0	0	0	5	1
bmw	0	0	0	8	0
chevrolet	0	0	2	1	0
dodge	0	0	5	3	1
honda	0	0	7	5	1
isuzu	0	0	1	1	0
jaguar	0	0	0	3	0
mazda	0	0	10	7	0
mercedes-benz	1	2	0	4	1
mercury	0	0	1	0	0

Apply margins and margins_name attribute to display the row wise andcolumn wise sum of the cross table

```
pd.crosstab(df["make"], df["bodystyle"], margins=True,margins_name="Total Made")
```

Output:

body-style make	convertible	hardtop	hatchback	sedan	wagon	Total Made
alfa-romero	2	0	1	0	0	3
audi	0	0	0	5	1	6
bmw	0	0	0	8	0	8
chevrolet	0	0	2	1	0	3
dodge	0	0	5	3	1	9
honda	0	0	7	5	1	13
isuzu	0	0	1	1	0	2
jaguar	0	0	0	3	0	3
mazda	0	0	10	7	0	17
mercedes-benz	1	2	0	4	1	8
mercury	0	0	1	0	0	1
mitsubishi	0	0	9	4	0	13
nissan	0	1	5	9	3	18
peugot	0	0	0	7	4	11

- Applying multiple columns in the crosstab function for the row index or column index or both will print the output with grouping automatically.

2. Data distribution by the body-type and drive_wheels columns within the maker of car and their door type in a crosstab:

```
pd.crosstab([df["make"],df["num-of-doors"]], [df["bodystyle"], df["drive-wheels"]], margins=True,margins_name="Total Made")
```

Output:

	body-style	convertible		hardtop		hatchback			sedan			wagon			Total Made
	drive-wheels	fwd	rwd	fwd	rwd	4wd	fwd	rwd	4wd	fwd	rwd	4wd	fwd	rwd	
make	num-of-doors														
alfa-romero	two	0	2	0	0	0	0	1	0	0	0	0	0	0	3
audi	four	0	0	0	0	0	0	0	1	3	0	0	1	0	5
	two	0	0	0	0	0	0	0	0	1	0	0	0	0	1
bmw	four	0	0	0	0	0	0	0	0	0	5	0	0	0	5
	two	0	0	0	0	0	0	0	0	0	3	0	0	0	3
chevrolet	four	0	0	0	0	0	0	0	0	1	0	0	0	0	1
	two	0	0	0	0	0	2	0	0	0	0	0	0	0	2
dodge	four	0	0	0	0	0	1	0	0	3	0	0	1	0	5
	two	0	0	0	0	0	4	0	0	0	0	0	0	0	4
honda	four	0	0	0	0	0	0	0	0	4	0	0	1	0	5
	two	0	0	0	0	0	7	0	0	1	0	0	0	0	8
isuzu	four	0	0	0	0	0	0	0	0	0	1	0	0	0	1

#

Rename the columns and row index for better understanding of crosstab

```
pd.crosstab([df["make"],df["num-of-doors"]], [df["bodystyle"],df["drive-wheels"]], rownames=['Auto Manufacturer', "Doors"], colnames=['Body Style', "Drive Type"], margins=True,margins_name="Total Made").head()
```

Output:

	Body Style	convertible		hardtop		hatchback			sedan			wagon			Total Made
	Drive Type	fwd	rwd	fwd	rwd	4wd	fwd	rwd	4wd	fwd	rwd	4wd	fwd	rwd	
Auto Manufacturer	Doors														
alfa-romero	two	0	2	0	0	0	0	1	0	0	0	0	0	0	3
audi	four	0	0	0	0	0	0	0	1	3	0	0	1	0	5
	two	0	0	0	0	0	0	0	0	1	0	0	0	0	1
bmw	four	0	0	0	0	0	0	0	0	0	5	0	0	0	5
	two	0	0	0	0	0	0	0	0	0	3	0	0	0	3

- The pivot table syntax of pd.crosstab also takes some arguments, such as dataframe columns, values, normalize, and the aggregation function.
- The aggregation function can be applied to a cross table at the same time.
- Passing the aggregation function and values, which are the columns that aggregation will be applied to, gives a cross table of a summarized subset of the dataframe.

3. The average curb-weight of cars made by different makers with respect to their body-style by applying the mean() aggregation function to the crosstable:

```
# values are the column in which aggregation function is to be applied
# aggfunc is the aggregation function to be applied round() to round output

    pd.crosstab(df["make"], df["body-style"],values=df["curb-weight"],
    aggfunc='mean').round(0)
```

Output:

body-style	convertible	hardtop	hatchback	sedan	wagon
make					
alfa-romero	2548.0	NaN	2823.0	NaN	NaN
audi	NaN	NaN	NaN	2720.0	2954.0
bmw	NaN	NaN	NaN	2929.0	NaN
chevrolet	NaN	NaN	1681.0	1909.0	NaN
dodge	NaN	NaN	2132.0	2056.0	2535.0
honda	NaN	NaN	1970.0	2289.0	2024.0
isuzu	NaN	NaN	2734.0	2337.0	NaN
jaguar	NaN	NaN	NaN	4027.0	NaN
mazda	NaN	NaN	2254.0	2361.0	NaN
mercedes-benz	3685.0	3605.0	NaN	3731.0	3750.0
mercury	NaN	NaN	2910.0	NaN	NaN
mitsubishi	NaN	NaN	2377.0	2394.0	NaN
nissan	NaN	2008.0	2740.0	2238.0	2452.0
peugot	NaN	NaN	NaN	3143.0	3358.0

- A normalized crosstab will show the percentage of time each combination occurs.
- This can be accomplished using the normalize parameter, as follows:

- Cross-tabulation techniques is useful to analyze two or more variables.
- This helps in inspecting the relationships between them.

```
pd.crosstab(df["make"], df["body-style"],normalize=True).head(10)
```

Output:

body-style make	convertible	hardtop	hatchback	sedan	wagon
alfa-romero	0.009950	0.00000	0.004975	0.000000	0.000000
audi	0.000000	0.00000	0.000000	0.024876	0.004975
bmw	0.000000	0.00000	0.000000	0.039801	0.000000
chevrolet	0.000000	0.00000	0.009950	0.004975	0.000000
dodge	0.000000	0.00000	0.024876	0.014925	0.004975
honda	0.000000	0.00000	0.034826	0.024876	0.004975
isuzu	0.000000	0.00000	0.004975	0.004975	0.000000
jaguar	0.000000	0.00000	0.000000	0.014925	0.000000
mazda	0.000000	0.00000	0.049751	0.034826	0.000000
mercedes-benz	0.004975	0.00995	0.000000	0.019900	0.004975

UNIT III

3 Univariate Analysis

Syllabus

Introduction to Single variable : Distributions and Variables - Numerical Summaries of Level and Spread - Scaling and Standardizing - Inequality - Smoothing Time Series.

Contents

3.1 Introduction to Single Variable

- Exploratory data analysis is cross-classified in two different ways where each method is either graphical or non-graphical and then, each method is either univariate, bivariate or multivariate.
- **Univariate analysis** is simplest analysis of statistical data. The term **univariate analysis** refers to the analysis of one variable. The prefix "uni" means "one." The purpose of univariate analysis is to understand the distribution of values for a single variable. Univariate analysis explores each variable in a data set, separately.
- In other words in univariate analysis data has only one variable. It doesn't deal with causes or relationships and it's major purpose is to describe data; it takes data, summarizes that data and finds patterns in the data.
- Some ways one can describe patterns found in univariate data include central tendency (mean, mode and median) and dispersion : Range , variance, maximum, minimum, quartiles (including the interquartile range) and standard deviation.
- Univariate analysis works by examining the effects of a singular variable on a set of data. For example, a frequency distribution table is a form of univariate analysis as frequency is the only variable being measured. Alternative variables may be age, height, weight, etc., however it is important to note that as soon as a secondary variable is introduced it becomes bivariate analysis. With three or more variables, it becomes multivariate analysis.

3.1.1 Univariate Statistics

- Univariate analysis can be performed in a statistical setting. Two types of statistics can be used for analysis namely, descriptive and inferential.

Descriptive statistics

- As the name suggests, descriptive statistics are used to describe data. The statistics used here are commonly referred to as summary statistics.
- Descriptive statistics can be used for calculating things like missing value proportions, upper and lower limits for outliers, level of variance through the coefficient of variance, etc.

Inferential statistics

- Often, the data one is dealing with is a subset (sample) of the complete data (population). Thus, the common question here is -
  - Can the findings of the sample be extrapolated to the population ? That is, is the sample representative of the population or has the population changed ? Such questions are answered using specific hypothesis tests designed to deal with such univariate data-based problems.

- **Hypothesis tests** help to answer crucial questions about the data and their relation with the population from where they are drawn. Several hypotheses or univariate testing mechanisms come in handy here, such as -
  1. **Z Test** - Used for numerical (quantitative) data where the sample size is greater than 30 and the population's standard deviation is known.
  2. **One-Sample t-Test** - Used for numerical (quantitative) data where the sample size is less than 30 or the population's standard deviation is unknown.
  3. **Chi-Square Test** - Used with ordinal categorical data.
  4. **Kolmogorov-Smirnov Test** - Used with nominal categorical data.
- There are below common methods for performing univariate analysis,
  1. Summary statistics
  2. Frequency distributions
  3. Charts
  4. Univariate tables.

1. Summary statistics

- The most common way to perform the univariate analysis is to use summary statistics to describe a variable. There are two kinds of summary statistics :
  1. **Measures of central tendency** - These values describe where the dataset's center or middle value is located. The mean, mode and median are examples.
  2. **Dispersion measures** - These numbers describe how evenly distributed the values are in the dataset. The range, standard deviation and variance are some examples.
  3. **Measure of shape** - The shape of the data distribution can explain a great deal about the data as the shape can help in identifying the type of distribution followed by the data. Each of these distributions has specific properties that can be used to one's advantage. By analyzing the shapes, one will know if the data is symmetrical, non-symmetrical, left or right-skewed, is suffering from positive or negative kurtosis, among other things.

2. Frequency distributions

- A frequency distribution describes how frequently different values occur in a dataset. This acts as another way to perform univariate analysis.

3. Charts

- Another method for performing univariate analysis is to create charts that show the distribution of values for a specific variable.
- Various types of graphs can be used to understand data. The standard type of graphs include -

1. **Histograms :** A histogram displays the frequency of each value or group of values (bins) in numerical data. This helps in understanding how the values are distributed.
2. **Boxplot :** A boxplot provides several important information such as minimum, maximum, median, 1st and 3rd quartiles. It is beneficial in identifying outliers in the data.
3. **Density curve :** The density curve helps in understanding the shape of the data's distribution. It helps answer questions such as if the data is bimodal, normally distributed, skewed, etc.
4. **Bar chart :** Bar charts, mainly frequency bar charts, is a univariate chart used to find the frequency of the different categories of categorical data.
5. **Pie chart :** Frequency Pie charts convey similar information to bar charts. The difference is that they have a circular formation with each slice indicating the share of each category in the data.

ivariate tables

Tables help in univariate analysis and are typically used with categorical data or numerical data with limited cardinality. Different types of tables include :

1. **Frequency tables :** Each unique value and its respective frequency in the data is shown through a table. Thus, it summarizes the frequency the way a histogram, frequency bar or pie chart does but in a tabular manner.
2. **Grouped tables :** Rather than finding the count of each unique value, the values are binned or grouped and the frequency of each group is reflected in the table. It is typically used for numerical data with high cardinality.
3. **Percentage (Proportion) tables :** Rather than showing the frequency of the unique values (or groups), such a table shows their proportion in the data (in percentage).
4. **Cumulative proportion tables :** It is similar to the proportion table, with the difference being that the proportion is shown cumulatively. It is typically used with binned data having a distinct order (or with

3.1.2 Variable and Distribution in Univariate Analysis

- A variable in univariate analysis is a condition or subset that data falls into. Variable can be thought of as a "category." For example, the analysis might work on a variable "height" or it might work on "weight". Univariate analysis can be carried out on any of the individual variables in the dataset to gain a better understanding of its distribution of values.

Univariate data examples

- The salaries of employees in a specific industry; the variable in this example is employee's salaries.
- The heights of ten students in a class are measured; the variable here is the student's heights.
- A veterinarian wants to weigh 20 cats; the variable, in this case, is the weight of the cats.
  - Finding the average height of a country's men from a sample.
  - Calculate how reliable a batsman is by calculating the variance of their runs.
  - Finding which country is the most frequent in winning Olympic Gold Medal by creating a frequency bar chart or frequency table.
  - Understanding the income distribution of a county by analyzing the distribution's shape. A right-skewed distribution can indicate an unequal society.
  - Checking if the price of sugar has statistically significantly risen from the generally accepted price by using sample survey data. Hypothesis tests such as the Z or t-test solve such questions.
  - Assessing the predictive capability of a variable by calculating the coefficient of variance.

Distribution and variables

- **Types of variables :** Variables can be one of two types : Categorical or numerical.

Categorical Data

- Categorical data classify items into groups. This type of data can be further broken down into nominal, ordinal and binary values.
  - Ordinal values have a set order. An example here could be a ranking of low to high.
  - Nominal values have no set order. Examples include the superhero's gender and alignment.
  - Binary data has only two values. This could be represented as true / false or 1/0.

- A common way to summarize categorical variables is with a frequency table.
- Columns holding categorical data : Gender, Married, BankCustomer, Industry, Ethnicity, PriorDefault, Employed, DrivingLicense, Citizen, Approved.

Numerical data

- Numerical data are values that one can perform mathematical operations on. They are further broken down into continuous and discrete data types.
  - Discrete variables have to be an integer. An example is number of superheroes.
  - Continuous can be any value. Examples here include height and weight.
- Numerical data can be visualized with a histogram. Histograms are a great first analysis of continuous data. Four main aspects to consider here are shape, center, spread and outliers.
  - Shape is the overall appearance of the histogram. It can be symmetric, skewed, uniform or have multiple peaks.
  - Center refers to the mean or median.
  - Spread refers to the range or how far the data reaches.
  - Outliers are data points that fall far from the bulk of the data.
- **Columns holding numerical and continuous data :** Age, debt, YearsEmployed, CreditScore, Income.

3.2 Storing and Importing Data using Python

- There are various methods to import data in Python. One of the way to import data is using Pandas library.
- In most simplest form data can be stored in a CSV file. CSV stands for "Comma Separated Values." It is the simplest form of storing data in tabular form as plain text.
- CSV file structure is very simple in which, the first line of a CSV file is the header and contains the names of the fields/features separated by "comma". After the header, each line of the file is an data set value/observation/record. The values of a record are separated by "comma."

Steps to import using Pandas

1. Get the correct and full file path

- Firstly, capture the full path where CSV file is stored.
- For example, a CSV file is stored under the following path :

```
D:\Tech_data\myteam.csv
```

- File name - It should made sure that the file name specified in the code matches with the actual file name.
- File extension - The file extension should always be '.csv' when importing CSV files.

Example program - 1

```
# Python code to import the file

import pandas as pd

#read the csv file (put 'r' before the path string to address any special #characters in the path, such as '\').
df = pd.read_csv (r'D:\Tech_data\myteam.csv')
print (df)
```

Example program 1 - Output

Person	Name	City	Plays	Score
0	Lucky	Pune	Cricket	500
1	Aniket	Mumbai	Cyclist	900
2	Lahu	Kolhapur	Tennis	900
3	Shital	Amarawati	Badminton	700
4	Raj	Nashik	Cricket	800
5	Mohit	Ratnagiri	Tennis	700
6	Jaya	Anagar	Badminton	600
7	Dev	Radhangari	Cyclist	800
8	Anita	Vashi	Tennis	800
9	Rashmi	Dhule	Badminton	600

3.3 Numerical Summaries of Level and Spread

- The two main types of summary are, summaries of the center of the distribution and of spread.
- The three major characteristics of the distribution for a quantitative variable that are of primary interest are the center of the distribution, the amount of dispersion in the distribution and the shape of the distribution. A numerical summary is a number used to describe a specific characteristic about a data set.

Below are some of the useful numerical summaries

- **Center :** Mean, median, mode
- **Quantiles :** Percentiles, five number summaries
- **Spread :** Standard deviation, variance, interquartile range
- **Outliers**
- **Shape :** Skewness, kurtosis
- **Concordance :** Correlation, quantile-quantile plots.

Mean

- This is the point of balance, describing the most typical value for normally distributed data. By "normally distributed" data it means it is highly influenced by outliers.
- The mean adds up all the data values and divides by the total number of values, as follows :

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

- The 'x-bar' is used to represent the sample mean (the mean of a sample of data). 'Σ' (sigma) implies the addition of all values up from 'i=1' until 'i=n' ('n' is the number of data values). The result is then divided by 'n'.

Median

- This is the "middle data point", where half of the data is below the median and half is above the median. It's the 50^{th} percentile of the data It's also mostly used with skewed data because outliers won't have a big effect on the median.
- There are two formulas to compute the median. The choice of which formula to use depends on n (number of data points in the sample or sample size) if it's even or odd.

$$\text{Median} = \frac{x_{(n/2)} + x_{(n/2+1)}}{2}$$

- When n is **even**, there is no "middle" data point, so the middle two values are averaged.

$$\text{Median} = x_{((n+1)/2)}$$

- When n is **odd**, the middle data point is the median.

Mode

- The mode returns the most commonly occurring data value.

Percentile

- The percent of data that is equal to or less than a given data point. It's useful for describing where a data point stands within the data set. If the percentile is close to zero, then the observation is one of the smallest. If the percentile is close to 100, then the data point is one of the largest in the data set.

Quartiles (Five-number summary)

- Quartiles measure the center and it's also great to describe the spread of the data. Highly useful for skewed data. There are four quartiles and they compose the five-number summary(combined with the minimum). The Five-number summary is composed of :
  1. Minimum
  2. 25^{th} percentile (lower quartile)
  3. 50^{th} percentile (median)
  4. 75^{th} percentile (upper quartile)
  5. 100^{th} percentile (maximum)

Standard deviation

- Standard deviation is extensively used in statistics and data science. It measures the amount of variation or dispersion of a data set, calculating how spread out the data are from the mean. Small values mean the data is consistent and close to the mean. Larger values indicate the data is highly variable.
- **Deviation :** The idea is to use the mean as a reference point from which everything varies. A deviation is defined as the distance an observation lies from the reference point. This distance is obtained by subtracting the data point (x_i) from the mean (x-bar).

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

- **Calculating the standard deviation :** The average of all the deviations will always turn out to be zero, so one can square each deviation and sum up the results. Then, one can divide it for 'n – 1' (called degrees of freedom). Further, square root the final result to undo the squaring of the deviations.
- The standard deviation is a representation of all deviations in the data. It's never negative and it's zero only if all the values are the same.

Variance

- Variance is almost the same calculation of the standard deviation, but it stays in squared units. So, if taken the square root of the variance, one gets the standard deviation. Note that it's represented by 's-squared', while the standard deviation is represented by 's'.

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

Range

- The difference between the maximum and minimum values. Useful for some basic exploratory analysis, but not as powerful as the standard deviation.

$$x_{(n)} - x_{(1)}$$

Proportion

- It's often referred to as "percentage". Defines the percent of observations in the data set that satisfy some requirements.

$$\hat{p} = \frac{x}{n}$$

Correlation

- Defines the strength and direction of the association between two quantitative variables. It ranges between – 1 and 1. Positive correlations mean that one variable increases as the other variable increases. Negative correlations mean that one variable decreases as the other increases. When the correlation is zero, there is no correlation at all. As closest to one of the extreme the result is, stronger is the association between the two variables.

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

Example program - 2

```
#calculate mean, median
import pandas as pd
import numpy as np
df = pd.DataFrame([('Indian Cinema', 'Restaurant', 289.0),
                   ('RamKrishna', 'Restaurant', 224.0),
                   ('Zingo', 'Juice bar', 80.5),
                   ('The Place', 'Play Club', np.nan)],
```

```
columns=('name', 'type', 'AvgBill')
        )
print(df)
print('AvgBill Mean = ', df['AvgBill'].mean())
print('AvgBill Median = ', df['AvgBill'].median())
```

Example program - 2 Output

name	type	AvgBill
Indian Cinema	Restaurant	289.0
RamKrishna	Restaurant	224.0
Zingo	Juice bar	80.5
The Place	Play Club	NaN

AvgBill Mean = 197.83333333333334
AvgBill Median = 224.0

Example program - 3

```
import pandas as pd
data = pd.DataFrame({'col1':[5, 2, 7, 3, 4, 4, 2, 3, 2, 1, 2, 5],    # Create pandas DataFrame
                     'col2':['y', 'x', 'x', 'z', 'x', 'y', 'y', 'x', 'z', 'x', 'z', 'x'],
                     'group':['A', 'C', 'B', 'B', 'A', 'C', 'A', 'A', 'C', 'B', 'B', 'A']})
print(data)
print('Col2 Mode = ', data['col1'].mode())
```

Example program - 3 Output

	col1	col2	group
0	5	y	A
1	2	x	C
2	7	x	B
3	3	z	B
4	4	x	A
5	4	y	C
6	2	y	A
7	3	x	A
8	2	z	C
9	1	x	B
10	2	z	B
11	5	x	A

Col2 Mode = 0 2
dtype: int64

Example program - 4

```
# calculate a 5-number summary
from numpy import percentile
from numpy.random import rand

# generate data sample
data = rand(1000)

# calculate quartiles
quartiles = percentile(data, [25, 50, 75])
# calculate min/max
data_min, data_max = data.min(), data.max()
# print 5-number summary
print('Min: %.3f' % data_min)
print('Q1: %.3f' % quartiles[0])
print('Median: %.3f' % quartiles[1])
print('Q3: %.3f' % quartiles[2])
print('Max: %.3f' % data_max)
```

Example program - 4 Output

```
Min : 0.001
Q1 : 0.269
Median : 0.509
Q3 : 0.762
Max : 0.999
```

3.4 Scaling and Standardizing

- **Feature scaling** (also known as **data normalization**) is the method used to standardize the range of features of data. Since, the range of values of data may vary widely, it becomes a necessary step in data preprocessing while exploring and visualizing data.
- Scaling of data may be useful and/or necessary under certain circumstances (e.g. when variables span different ranges). There are several different versions of scaling, the most important of which are listed below. Scaling procedures may be applied to the full data matrix or to parts of the matrix only (e.g. column-wise).

Range scaling

- Range scaling transforms the values to another range which usually includes both a shift and a change of the scale (magnification or reduction). In scaling (also called **min-max scaling**), one transforms the data such that the features are within a specific range e.g. [0, 1].
- Scaling is important in the algorithms such as Support Vector Machines (SVM) and k-nearest neighbors (KNN) where distance between the data points is important. For example, in the dataset containing prices of products; without scaling, SVM might treat 1 ₹ equivalent to 1 Euro though 1 Euro = 90 INR.
- The data samples are transformed according to the following equation :

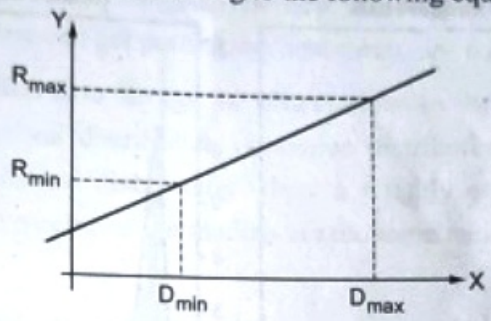


Fig. 3.4.1 Range scaling

$$Y = X\frac{R_{max} - R_{min}}{D_{max} - D_{min}} + \frac{R_{min}\,D_{max} - R_{max}\,D_{min}}{D_{max} - D_{min}}$$

Example program - 5

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import minmax_scale, scale

# set seed for reproducibility
np.random.seed(0)

# generate random data points from an exponential distribution
x = np.random.exponential(size=1000)

# mix-max scaling
scaled_data = minmax_scale(x)
```

```
# scaled_data = (x-x.min())/(x.max()-x.min())
# plot both together to compare
f, ax = plt.subplots(1,2)
sns.distplot(x, ax=ax[0], color='g')
ax[0].set_title("Original data")
sns.distplot(scaled_data, ax=ax[1], color='r')
ax[1].set_title("Scaled data")
plt.show()
```

Example program - 5 Output

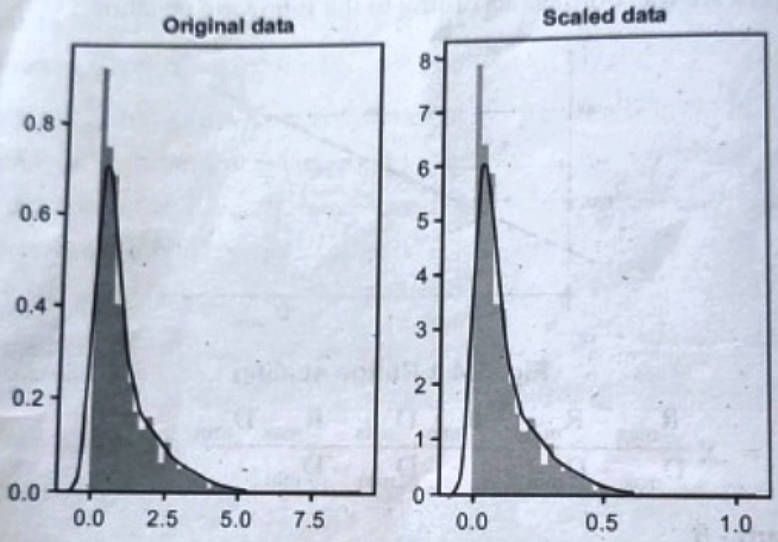


Fig. 3.4.2 Original and scaled data

Mean centering

- Subtracting the mean of the data is often called "**mean centering**". It results in a shift of the data towards the mean. The mean of the transformed data thereafter equals zero :

$$Y = X - \mu$$

Standardization and Normalization

- Standardization (sometimes also called autoscaling or z-transformation, **z-score normalization**) is the scaling procedure which results in a zero mean and unit variance of any descriptor variable. For every data value the mean μ has to be subtracted and the result has to be divided by the standard deviation σ (note that the order of these two operations must not be reversed) :

$$Y = \frac{(X - \mu)}{\sigma}$$

where X is the original feature vector, μ is the mean of that feature vector and σ is its standard deviation.

The z-score comes from statistics, defined as,

$$z = \frac{x - \mu}{\sigma}$$ where μ is the mean.

- By subtracting the mean from the distribution, it is essentially being shifted towards left or right by amount equal to mean i.e. if there is a distribution of mean 100 and one subtracts mean 100 from every value, then one is shifting the distribution left by 100 without changing its shape. Thus, the new mean will be 0. When it is divided by standard deviation σ, the shape of distribution is changed. The new standard deviation of this standardized distribution is 1 which one can get putting the new mean, $\mu = 0$ in the z-score equation.
- The point of normalization is to change the observations so that they can be described as a normal distribution. Normal distribution (Gaussian distribution), also known as the **bell curve**, is a specific statistical distribution where a roughly equal observations fall above and below the mean, the mean and the median are the same and there are more observations closer to the mean.
- **Standardization**(also called) transforms the data such that the resulting distribution has a mean of 0 and a standard deviation of 1.

$$x' = \frac{x - x_{mean}}{\sigma}$$

- There is a need to normalize the data if data is going to get used in machine learning or data analysis techniques that assume that data is normally distributed e.g. t-tests, ANOVAs, linear regression, Linear Discriminant Analysis (LDA) and Gaussian Naive Bayes.

Example program - 6

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import minmax_scale, scale
x = np.random.exponential(size=1000)
# standardization
standardized_data = scale(x)
# plot
fig, ax=plt.subplots(1,2)
```

```
sns.distplot(x, ax=ax[0], color='g')
ax[0].set_title("Original data")
sns.distplot(standardized_data, ax=ax[1], color='r')
ax[1].set_title("Standardized data")
plt.show()
```

Example program - 6 Output

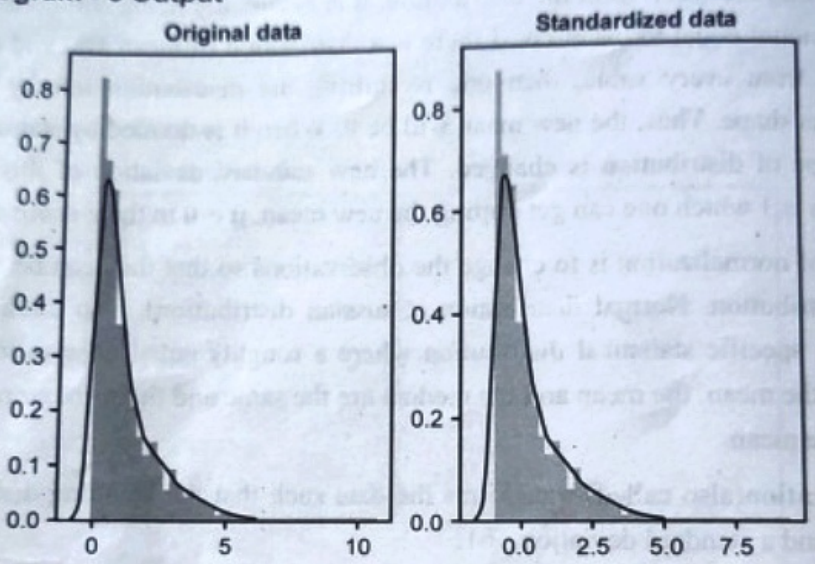


Fig. 3.4.3 Original and Standardized data

Example program - 7

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import minmax_scale, scale
x = np.random.exponential(size=1000)
# normalization
normalized_data = (x-x.mean())/(x.max()-x.min())
# plot
fig, ax=plt.subplots(1,2)
sns.distplot(x, ax=ax[0], color='y')
ax[0].set_title("Original data")
sns.distplot(normalized_data, ax=ax[1])
ax[1].set_title("Normalized data")
plt.show()
```

Example program - 7 Output

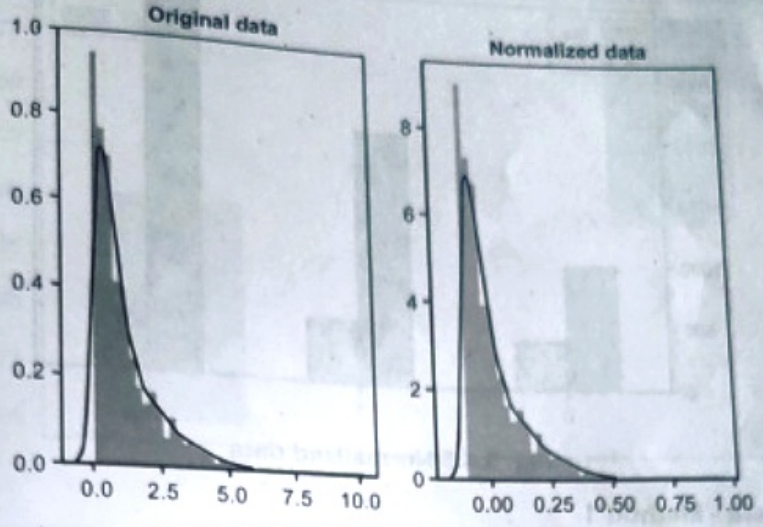


Fig. 3.4.4 Original and Normalized data

Example program - 8

```
import pandas as pd
Survey = pd.DataFrame({'No': [1000, 2000, 3000],
                       'Yes': [400, 500, 600]
                      })
df = pd.DataFrame(Survey)
print(df)
df.plot(kind = 'bar',color = 'red')
df_normalized = (df - df.mean() ) / df.std()
print('Normalized Data - Method 1')
print(df_normalized)
df_normalized.plot(kind = 'bar',color = 'blue')
normalized_df=df.apply(lambda x: (x-x.mean())/ x.std(), axis=0)
print('Normalized Data - Method 2')
print(normalized_df)
normalized_df.plot(kind = 'bar',color = 'green')
```

Example program - 8 Output

	No	Yes
0	1000	400
1	2000	500
2	3000	600

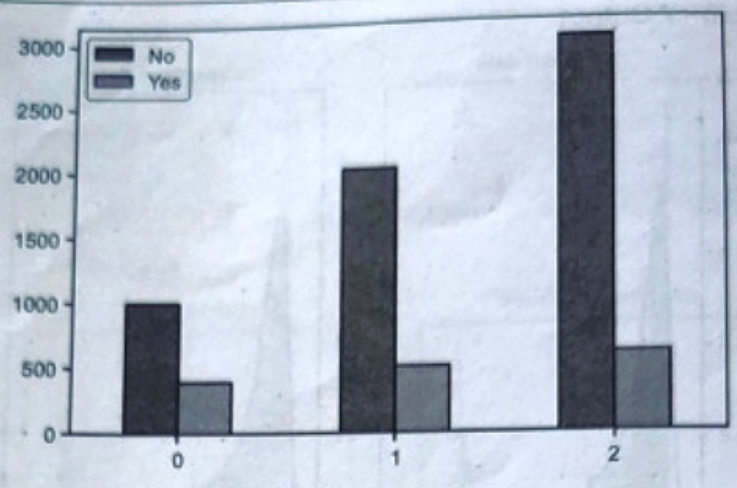


Fig. 3.4.5 Normalized data

Normalized Data - Method 1

	No	Yes
0	– 1.0	– 1.0
1	0.0	0.0
2	1.0	1.0

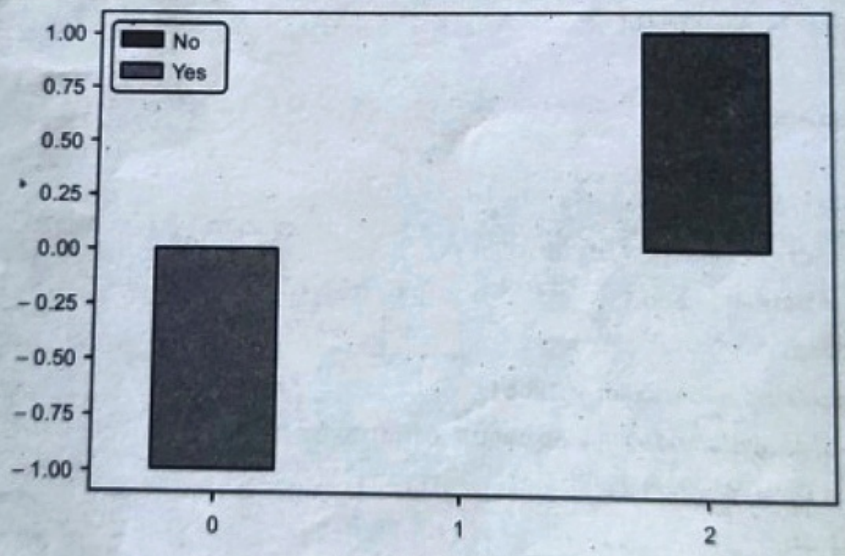


Fig. 3.4.6

Normalized Data - Method 2

	No	Yes
0	– 1.0	– 1.0
1	0.0	0.0
2	1.0	1.0

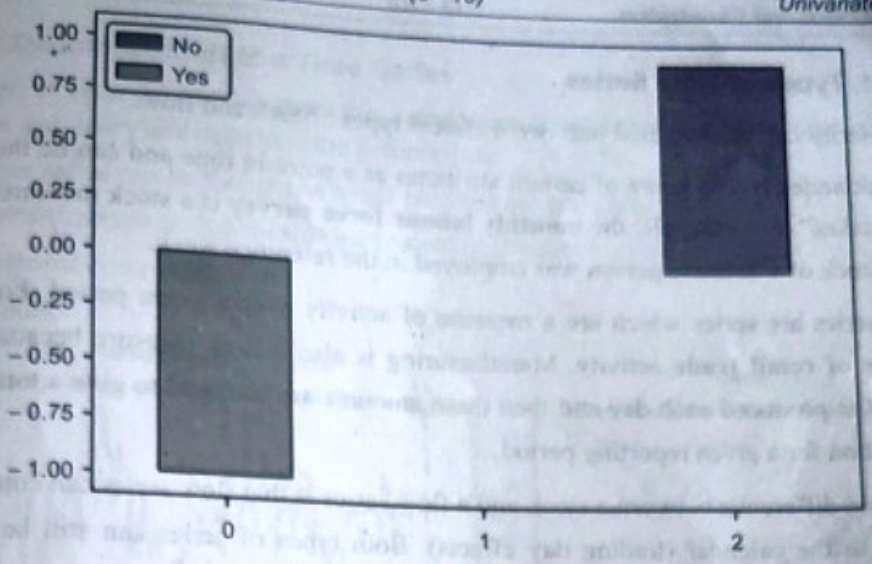


Fig. 3.4.7 Normalized data

3.5 Time Series and Smoothing Time Series

- A time series is a collection of observations of well-defined data items obtained through repeated measurements over time. Time series data, also referred to as time-stamped data, is a sequence of data points indexed in time order. These data points typically consist of successive measurements made from the same source over a fixed time interval and are used to track change over time. Time series data is a collection of observations obtained through repeated measurements over time.
- A time series is a data sequence ordered (or indexed) by time. It is discrete and the interval between each point is constant.
- Below are the examples of time series analysis,
  - Electrical - impulse activity in the brain
  - Rainfall measurements
  - Stock prices
  - Number of sunspots
  - Annual retail sales
  - Monthly subscribers
  - Heartbeats per minute
  - Sound frequency of audio.

3.5.1 Types of Time Series

- Time series can be classified into two different types : Stock and flow.
- A stock series is a measure of certain attributes at a point in time and can be thought of as "stocktakes". For example, the **monthly labour force survey** is a stock measure because it takes stock of whether a person was employed in the reference week.
- Flow series are series which are a measure of activity over a given period. For example, surveys of **retail trade** activity. Manufacturing is also a flow measure because a certain amount is produced each day and then these amounts are summed to give a total value for production for a given reporting period.
- The main difference between a stock and a flow series is that flow series can contain effects related to the calendar (trading day effects). Both types of series can still be seasonally adjusted using the same seasonal adjustment process.

3.5.2 Properties of Time Series

- An observed time series can be decomposed into three components : The trend (long term direction), the seasonal (systematic, calendar related movements) and the irregular (unsystematic, short term fluctuations).
  - ***Trend (deterministic)*** - A long-term increase or decrease in the data. This can be seen as a slope (it doesn't have to be linear) roughly going through the data.
  - ***Seasonality* (deterministic)** - A time series is said to be seasonal when it is affected by seasonal factors (hour of day, week, month, year, etc.). Seasonality can be observed with nice cyclical patterns of fixed frequency.
  - ***Cyclicity(deterministic)*** - A cycle occurs when the data exhibits rises and falls that are not of a fixed frequency. These fluctuations are usually due to economic conditions and are often related to the "business cycle". The duration of these fluctuations is usually at least 2 years.
- **Irregular components/ remainder (stationary process) / Residuals** - Each time series can be decomposed in two parts namely,
  - A forecast, made up of one or several forecasted values
  - Residuals : They are the difference between an observation and its predicted value at each time step.

Value of series at time t = Predicted value at time t + Residual at time t

3.5.3 Decomposition of a Time Series

- In order to remove the deterministic components one can decompose time series into separate stationary and deterministic components.
- Each time series can be thought as a mix between several parts,
  - A trend (upward or downwards movement of the curve on the long term)
  - A seasonal component
  - Residuals.

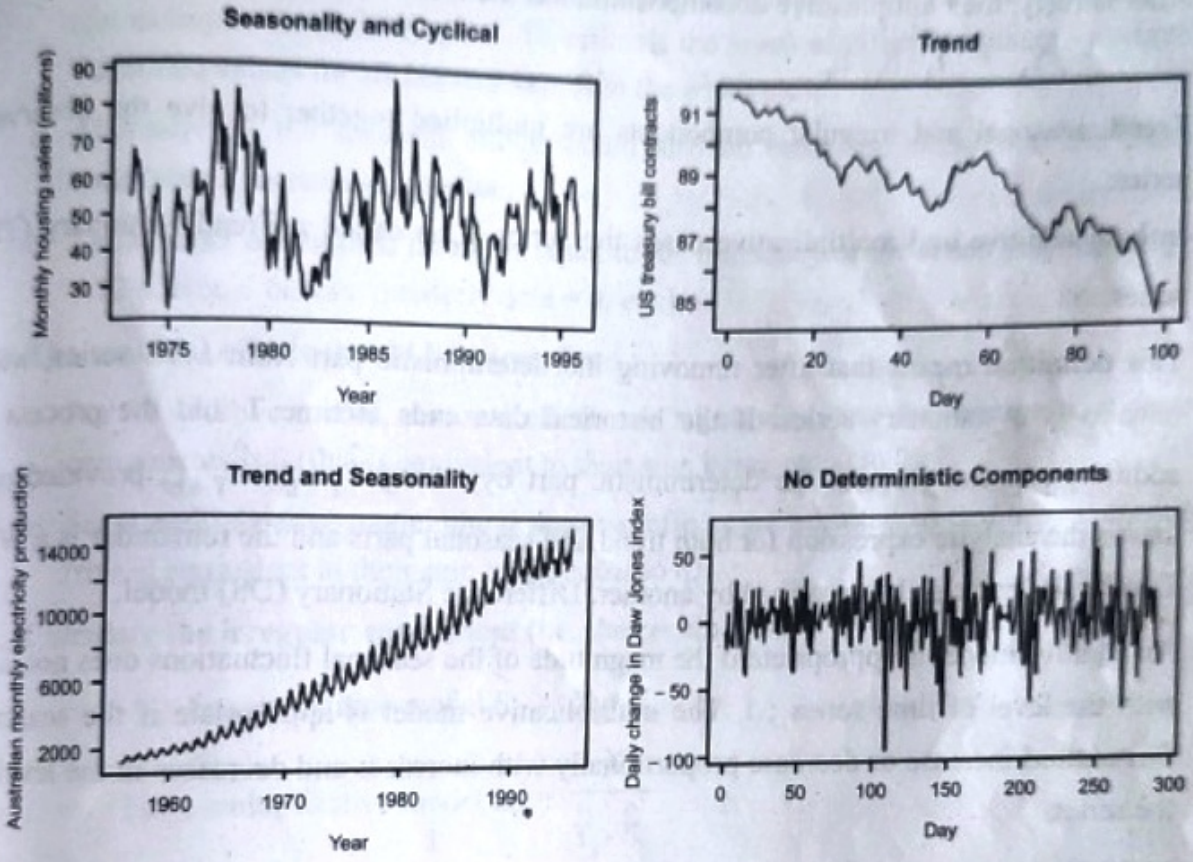


Fig. 3.5.1 Time series properties

- ***Stationarity*** is the property of exhibiting constant statistical properties (mean, variance, autocorrelation. If the mean of a time-series increases over time, then it's not stationary.
- The general mathematical representation of the decomposition approach :

$$Y_t = f(T_t, S_t, E_t)$$

Where Y_t is the time series value (actual data) at period t;

T_t is a deterministic trend-cycle or general movement component;

S_t is a deterministic seasonal component

E_t is the irregular (remainder or residual) (stationary) component.

- The exact functional form of f (·) depends on the decomposition method used.

3.5.4 Trend Stationary Time Series

- A common approach is to assume that the equation has an additive form :

$$Y_t = T_t + S_t + E_t$$

- Trend, seasonal and irregular components are simply added together to give the observed series.
- Alternatively, the multiplicative decomposition has the form :

$$Y_t = T_t \cdot S_t \cdot E_t$$

- Trend, seasonal and irregular components are multiplied together to give the observed series.
- In both additive and multiplicative cases the series Y_t is called a Trend Stationary (TS) series.
- This definition means that after removing the deterministic part from a TS series, what remains is a stationary series. If the historical data ends at time T and the process is additive, one can forecast the deterministic part by taking, $\hat{T}_{T+h} + \hat{S}_{T+h}$, provided one knows the analytic expression for both trend and seasonal parts and the remainder is a WN. Time series can also be described by another, Difference Stationary (DS) model.
- An additive model is appropriate if the magnitude of the seasonal fluctuations does not vary with the level of time series ; I. The multiplicative model is appropriate if the seasonal fluctuations increase or decrease proportionally with increases and decreases in the level of the series.
- Multiplicative decomposition is more prevalent with economic series because most seasonal economic series do have seasonal variations which increase with the level of the series.
- Rather than choosing either an additive or multiplicative decomposition, one should transform the data beforehand.

Basic steps in decomposition (1)

1. **Estimate the trend. Two approaches :**
   - Using a smoothing procedure;
   - Specifying a regression equation for the trend;

2. **De-trending the series :**
   - For an additive decomposition, this is done by subtracting the trend estimates from the series;
   - For a multiplicative decomposition, this is done by dividing the series by the estimated trend values.
3. **Estimating the seasonal factors from the detrended series :**
   - Calculate the mean (or median) values of the detrended series for each specific period (for example, for monthly data - To estimate the seasonal effect of January - average the detrended values for all January values in the series etc);
   - Alternatively, the seasonal effects could also be estimated along with the trend by specifying a regression equation.
   - The number of seasonal factors is equal to the frequency of the series (e.g. monthly data = 12 seasonal factors, quarterly data = 4, etc.).
4. **The seasonal effects should be normalized :**
   - For an additive model, seasonal effects are adjusted so that the average of d seasonal components is 0 (this is equivalent to their sum being equal to 0);
   - For a multiplicative model, the d seasonal effects are adjusted so that they average to 1 (this is equivalent to their sum being equal to d);
5. **Calculate the irregular component (i.e. the residuals) :**

$$\text{For an additive model } \hat{E}_t = Y_t - \hat{T}_t - \hat{S}_t$$

$$\text{For a multiplicative model } \hat{E}_t = \frac{Y_t}{\hat{T}_t \cdot \hat{S}_t}$$

   - Analyze the residual component. Whichever method was used to decompose the series, the aim is to produce stationary residuals.
6. Choose a model to fit the stationary residuals (e.g. see ARMA models).
7. Forecasting can be achieved by forecasting the residuals and combining with the forecasts of the trend and seasonal components.

Estimating the trend, T_t

- There are various ways to estimate the trend T_t at time t but a relatively simple procedure which does not assume any specific form of T_t is to calculate a moving average centered on t.

- A moving average is an average of a specific number of time series values around each value of t in the time series, with the exception of the first few and last few terms (this procedure is available in Python with the decompose function). This method smoothes the time series. The estimation depends on the seasonality of the time series :
  - If the time series has no seasonal component;
  - If the time series contains a seasonal component;
- Smoothing is usually done to help to better see patterns (like the trend) in the time series by smoothing out the irregular roughness to see a clearer signal. For seasonal data, one might smooth out the seasonality so that one can identify the trend.
- Estimating T_t if the time series has no seasonal component In order to estimate the trend, one can take any odd number, for example, if $l = 3$, one can estimate an additive model,

$$\hat{T}_t = \frac{Y_{t-1} + Y_t + Y_{t+1}}{3}, \text{ (two - sided averaging)}$$

$$\hat{T}_t = \frac{Y_{t-2} + Y_{t-1} + Y_t}{3}, \text{ (one - sided averaging)}$$

- In this case, the average is calculated either,
  - Centered around t - One element to the left (past) and one element to the right (future),
  - Or alternatively - Two elements to the left of t (past values at $t - 1$ and $t - 2$).
- Estimating T_t if the time series contains a seasonal component.
- If the time series contains a seasonal component and one wants to average it out, the length of the moving average must be equal to the seasonal frequency (for monthly series, one would take $l = 12$). However, there is a slight hurdle. Suppose, the time series begins in January ($t = 1$) and one can average up to December ($t = 12$). This averages corresponds to a time $t = 6.5$ (time between June and July). While estimating seasonal effects, there is a need of moving average at integer times. This can be achieved by averaging the average of January to December and the average of February ($t = 2$) up to January ($t = 13$). This average of the two moving averages corresponds to $t = 7$ and the process is called **centering**.

Thus, the

$$\hat{T}_t = \frac{(Y_{t-6} + \ldots + Y_{t+5})/12 + (Y_{t-5} + \ldots + Y_{t+6})/12}{2}$$

$$= \frac{(1/2)\,Y_{t-6} + Y_{t-5} \ldots + Y_{t+5} + (1/2)\,Y_{t+6}}{12}$$

Where, $t = 7, \ldots, T - 6$.

- By using the seasonal frequency for the coefficients in the moving average, the procedure generalizes for any seasonal frequency (i.e. quarterly, weekly, etc. series), provided the condition that the coefficients sum up to unity is still met.

Estimating the seasonal component, S_t

- An estimate of S_t at time t can be obtained by subtracting $\hat{T}_t$:

$$\hat{S}_t = Y_t - \hat{T}_t$$

- By averaging these estimates of the monthly effects for each month (January, February etc.), one can obtain a single estimate of the effect for each month. That is, if the seasonality period is d, then :

$$S_t = S_{t+d}$$

- Seasonal factors can be thought of as expected variations from trend throughout a seasonal period, so one would expect them to cancel each other out over that period - i.e., they should add up to zero.

$$\sum_{t-1}^{d} S_t = 0$$

- It should be noted that this applies to the additive decomposition.

Estimating the seasonal component, S_t

- If the estimated (average) seasonal factors $\tilde{S}_t$ do not add up to zero, then one can correct them by dividing the sum of the seasonal estimates by the seasonality period and adjusting each seasonal factor. For example, if the seasonal period is d, then

1. Calculate the total sum : $\sum_{t-1}^{d} \tilde{S}_t$
2. Calculate the value $w = \dfrac{\sum_{t-1}^{d} \tilde{S}_t}{d}$
3. Adjust each period $\hat{S}_t = \tilde{S}_t - w$

- Now, the seasonal components add up to zero :

$$\sum_{t-1}^{d} \hat{S}_t = 0$$

- It is common to present economic indicators such as unemployment percentages as seasonally adjusted series. This highlights any trend that might otherwise be masked by seasonal variation (for example, to the end of the academic year, when schools and university graduates are seeking work). If the seasonal effect is additive, a seasonally adjusted series is given by, $Y_t - \hat{S}_t$.
- The described moving-average procedure usually quite successfully describes the time series in question, however it does not allow to forecast it.
- To decide upon the mathematical form of a trend, one must first draw the plot of the time series.
- If the behavior of the series is rather 'regular', one can choose a parametric trend - usually it is a low order polynomial in t, exponential, inverse or similar functions.
- In any case, the smoothing method is **acceptable** if the residuals $\hat{\epsilon}_t = Y_t - \hat{T}_t - \hat{S}_t$ constitute a stationary process.
- If there are a few competing trend specifications, the best one can be chosen by AIC, BIC or similar criterions.
- An alternative approach is to create models for all but some T0 end points and then choose the model whose forecast fits the original data best. To select the model, one can use such characteristics as :

Root Mean Square Error, $$RMSE = \sqrt{\frac{1}{T_0} \sum_{t-T-T_0}^{T} \hat{\epsilon}_t^2}$$

Mean Absolute Percentage Error, $$MAPE = \frac{100}{T_0} \sum_{t-T-T_0}^{T} \left| \frac{\hat{\epsilon}_t}{Y_t} \right|$$

and similar statistics.

3.5.5 Transforms used for Stationarizing Data

- ***Detrending*** - One can remove the underlying trend in the series. This can be done in several ways, depending on the nature of data,
- ***Indexed* data :** Data measured in currencies are linked to a price index or related to inflation. Dividing the series by this index (that is **deflating**) element-wise is therefore the solution to de-trend the data.

- **Non-indexed data :** Is it necessary to estimate if the trend is constant, linear or exponential. The first two cases are easy, for the last one it is necessary to estimate a growth rate (inflation or deflation) and apply the same method as for indexed data.

- **Differencing** - Seasonal or cyclical patterns can be removed by subtracting periodical values. If the data is 12-month seasonal, subtracting the series with a 12-lag difference series will give a "flatter" series.

- **Logging** - In the case where the compound rate in the trend is not due to a price index (i.e. the series is not measured in a currency), logging can help linearize a series with an exponential trend (recall that log(exp(x)) = x). It does not remove an eventual trend whatsoever, unlike deflation.

3.5.6 Checking Stationarity

Plotting rolling statistics

- Plotting rolling means and variances is a first good way to visually inspect the series. If the rolling statistics exhibit a clear trend (upwards or downwards) and show varying variance (increasing or decreasing amplitude), then one might conclude that the series is very likely not to be stationary.

Augmented Dickey-Fuller test

- This test is used to assess whether or not a time-series is stationary. Without getting into too much details about hypothesis testing, one should know that this test will give a result called a "test-statistic", based on which one can say, with different levels (or percentage) of confidence, if the time-series is stationary or not.

KPSS

- The KPSS (Kwiatkowski-Phillips-Schmidt-Shin) tests for the null hypothesis that the series is trend stationary. In other words, if the p-value of the test statistic is below the X % confidence threshold, this means one can reject this hypothesis and that the series is not trend-stationary with X % confidence. A p-value higher than the threshold will lead to accept this hypothesis and conclude that the series is trend-stationary.

Autocorrelation plots (ACF & PACF)

- An autocorrelation (ACF) plot represents the autocorrelation of the series with lags of itself. A partial autocorrelation (PACF) plot represents the amount of correlation between a series and a lag of itself that is not explained by correlations at all lower-order lags. Ideally, one would want no correlation between the series and lags of itself. Graphically speaking, one would like all the spikes to fall in the blue region.

Choosing a model

- Exponential smoothing methods are appropriate for non-stationary data (ie data with a trend and seasonal data). ARIMA (**Autoregressive Integrated Moving Average**) models should be used on stationary data only. One should therefore remove the trend of the data (via deflating or logging) and then look at the differenced series.

3.5.7 Smoothing Methods

- The smoothing technique is a family of time-series forecasting algorithms, which utilizes the weighted averages of a previous observation to predict or forecast a new value. This technique is more efficient when time-series data is moving slowly over time. It harmonizes errors, trends and seasonal components into computing smoothing parameters.
- Smoothing methods work as weighted averages. Forecasts are weighted averages of past observations. The weights can be uniform (this is a moving average) or following an exponential decay - This means giving more weight to recent observations and less weight to old observations. More advanced methods include other parts in the forecast, like seasonal components and trend components.

1. Simple exponential smoothing

- Simple Exponential Smoothing (SES) is one of the minimal models of the exponential smoothing algorithms. SES is the method of time series forecasting used with univariate data with no trend and no seasonal pattern. It needs a single parameter called **alpha** (α), also known as the **smoothing factor**. Alpha controls the rate at which the influence of past observations decreases exponentially. The parameter is often set to a value between 0 and 1. This method can be used to predict series that do not have trends or seasonality.
- The simple exponential smoothing formula is given by,

$$s_t = \alpha x_t + (1-\alpha)s_{t-1} = s_{t-1} + \alpha(x_t - s_{t-1})$$

here,

s_t = Smoothed statistic (simple weighted average of current observation x_t)

s_{t-1} = Previous smoothed statistic

α = Smoothing factor of data; $0 < \alpha < 1$

t = Time period

- Simple exponential smoothing should be used when there are few data points, irregular data, no seasonality or trend.

2. Double exponential smoothing

- This method is known as Holt's trend model or second-order exponential smoothing. Double exponential smoothing is used in time-series forecasting when the data has a linear trend but no seasonal pattern. The basic idea here is to introduce a term that can consider the possibility of the series exhibiting some trend.
- In addition to the alpha parameter, double exponential smoothing needs another smoothing factor called **beta** (β), which controls the decay of the influence of change in trend. The method supports trends that change in additive ways (smoothing with linear trend) and trends that change in multiplicative ways (smoothing with exponential trend).
- The double exponential smoothing formulas are,

$$S_1 = x_1$$

$$B_1 = x_1 - x_0$$

For $t > 1$,

$$s_t = \alpha x_t + (1 - \alpha)(s_{t-1} + b_{t-1})$$

$$\beta_t = \beta(s_t - s_{t-1}) + (1 - \beta)b_{t-1}$$

here,

b_t = Best estimate of the trend at time t

β = Trend smoothing factor; $0 < \beta < 1$

- Double exponential smoothing should be used when there is a trend in data and seasonality.

3. Triple exponential smoothing

- This method is the variation of exponential smoothing that's most advanced and is used for time series forecasting when the data has linear trends and seasonal patterns. The technique applies exponential smoothing three times - Level smoothing, trend smoothing and seasonal smoothing. A new smoothing parameter called **gamma** (γ) is added to control the influence of the seasonal component.
- The triple exponential smoothing method is called **Holt-Winters Exponential Smoothing**, named after its contributors, Charles Holt and Peter Winters.
- Holt-Winters Exponential Smoothing has two categories depending on the nature of the seasonal component :
  - Holt-Winter's Additive Method - For seasonality that is addictive.
  - Holt-Winter's Multiplicative Method - For seasonality that is multiplicative.

3.5.8 Configuration of Exponential Smoothing

- To configure exponential smoothing, analysts need to specify all the model hyperparameters explicitly. However, this can be challenging for both beginners and experts.
- Instead, numerical optimization is commonly used to search for and fund the smoothing factors (alpha, beta, gamma, phi) for the model resulting in the most negligible error.
- An exponential smoothing method can obtain values for unknown parameters by estimating them from the observed data. The initial values and unknown parameters can be estimated by minimizing the sum of the squared errors (SSE).
- The parameters that indicate the kind of change in trend or seasonality (for example, whether they are additive or multiplicative or whether they should be dampened) need to be specified explicitly.

3.5.9 Exponential Smoothing in Python

- The Statsmodels Python library provides the implementations of Exponential Smoothing in Python.

Single exponential smoothing

The SimpleExpSmoothingStatsmodels class enables implementation of Single Exponential Smoothing or simple smoothing in Python.

First, an instance of SimpleExpSmoothing is instantiated and passed the training data. Next, the fit() function is called, giving the fit configuration, especially the alpha value. The fit() function returns an instance of the HoltWintersResults class containing the learned coefficients. The forecast() or the predict() function is then called on the result object to make a forecast.

Double and triple exponential smoothing

The SimpleExpSmoothingStatsmodels class also enables the implementation of Double and Triple Exponential Smoothing in Python.

First, an instance of SimpleExpSmoothing is instantiated, specifying training data and model configuration. One must define the configuration parameters for trend, damped, seasonal and seasonal_periods. The fit() function is then called to fit the model on the training data.

The fit() function returns an instance of the HoltWintersResults class containing the learned coefficients. The forecast() or the predict() function is then called on the result object to

ARIMA

- ARIMA models (which include ARMA, AR and MA models) are a general class of models to forecast stationary time series. ARIMA models are made of three parts :
  - A weighted sum of lagged **values** of the series (**Auto-regressive (AR)** part)
  - A weighted sum of lagged **forecasted errors** of the series (**Moving-average (MA)** part)
  - A **difference** of the time series (**Integrated (I)** part)
- An ARIMA model is often noted **ARIMA(p, d, q)** where **p** represents the order of the **AR** part, **d** the order of differencing ("I" part) and q the order of the **MA** term.

1) Choosing the differencing order

- The first step of fitting an ARIMA model is to determine the differencing order to stationarize the series. To do that, ACF and PACF plots can be used and one should keep in mind these two rules,
- **Rule 1 -** If the series has positive autocorrelations out to a high number of lags, then it probably needs a higher order of differencing.
- **Rule 2 -** If the lag-1 autocorrelation is zero or negative or the autocorrelations are all small and patternless, then the series does not need a higher order of differencing. If the lag-1 autocorrelation is – 0.5 or more negative, the series may be over differenced.

2) Choosing the MA order

- To include a 1^{st} order difference in the model, one needs to choose the moving-average order. This is done by looking at the differenced series.
- If the lag-1 autocorrelation of the differenced series **ACF** is negative and/or there is a sharp cutoff, then choose a MA order of 1".

3) Choosing the AR order

- One should add an AR term in the case where,
- If the lag-1 autocorrelation of the differenced series **PACF** is negative and/or there is a sharp cutoff, then choose a AR order of 1".

3.5.10 Time Series Data Visualization

- Visualization plays an important role in time series analysis and forecasting.
- Plots of the raw sample data can provide valuable diagnostics to identify temporal structures like trends, cycles and seasonality that can influence the choice of model.
- A problem is that many novices in the field of time series forecasting stop with line plots.
- Below are widely used six different types of visualizations that can be used on the time series data.
- These techniques are applicable to univariate time series as well as the to multivariate time series, when there are more than one observation at each time step,

1. Line plots.
2. Histograms and density plots.
3. Box and whisker plots.
4. Heat maps.
5. Lag plots or scatter plots.
6. Autocorrelation plots.

Line plot

```
#Time Series line plot
```

Example program - 9

```
#plot dates and respective temperatures
#csv file has dates and respective temperatures
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('d:\Tech_Data\dateandtemps.csv', parse_dates=['Date'], index_col='Date')
df.plot(figsize=(10,10))
plt.show()
```

Example program - 9 Output

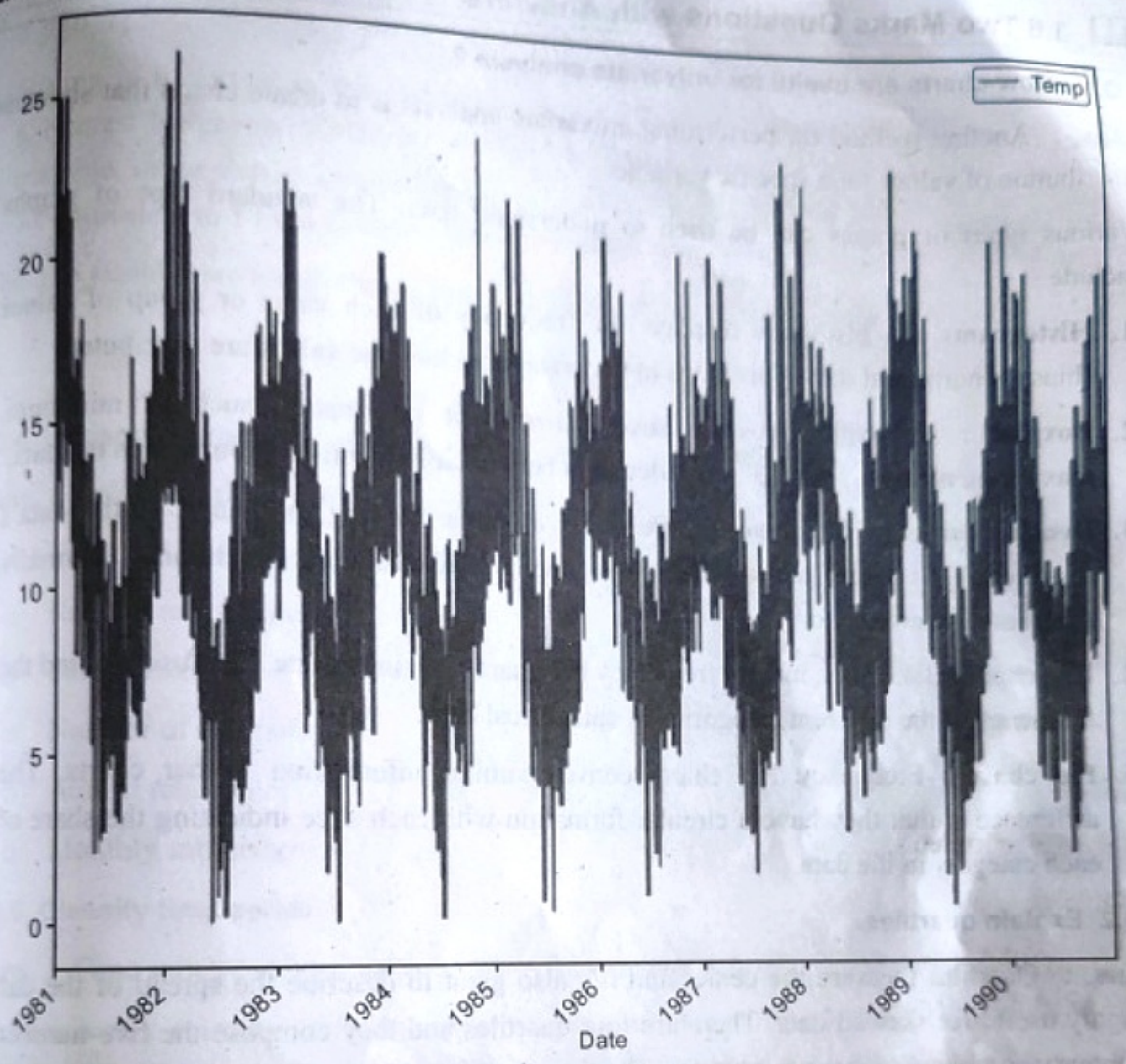


Fig. 3.5.2 Time series plot

Review Questions with Answers

1. *Discuss with example how univariate analysis is helpful ?* **(Refer section 3.1)**
2. *What are univariate tables ?* **(Refer section 3.1)**
3. *Explain numerical summaries level and spread used in univariate analysis.* **(Refer section 3.3)**
4. *What is feature scaling ?* **(Refer section 3.4)**
5. *How time series are useful in data analysis ?* **(Refer section 3.5)**

3.6 Two Marks Questions with Answers

Q.1 How charts are useful for univariate analysis ?

Ans. : Another method for performing univariate analysis is to create charts that show the distribution of values for a specific variable.

Various types of graphs can be used to understand data. The standard type of graphs include -

1. **Histograms :** A histogram displays the frequency of each value or group of values (bins) in numerical data. This helps in understanding how the values are distributed.
2. **Boxplot :** A boxplot provides several important information such as minimum, maximum, median, 1st and 3rd quartiles. It is beneficial in identifying outliers in the data.
3. **Density curve :** The density curve helps in understanding the shape of the data's distribution. It helps answer questions such as if the data is bimodal, normally distributed, skewed, etc.
4. **Bar chart :** Bar charts, mainly frequency bar charts, is a univariate chart used to find the frequency of the different categories of categorical data.
5. **Pie chart :** Frequency Pie charts convey similar information to bar charts. The difference is that they have a circular formation with each slice indicating the share of each category in the data.

Q.2 Explain quartiles.

Ans. : Quartiles measure the center and it's also great to describe the spread of the data. Highly useful for skewed data. There are four quartiles and they compose the five-number summary(combined with the minimum). The five-number summary is composed of :

1. Minimum
2. 25th percentile (lower quartile)
3. 50th percentile (median)
4. 75th percentile (upper quartile)
5. 100th percentile (maximum)

Q.3 Brief about range scaling.

Ans. : • Range scaling transforms the values to another range which usually includes both a shift and a change of the scale (magnification or reduction). In scaling (also called **min-**

max scaling), one transforms the data such that the features are within a specific range e.g. [0, 1].

- Scaling is important in the algorithms such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) where distance between the data points is important. For example, in the dataset containing prices of products; without scaling, SVM might treat 1 ₹ equivalent to 1 Euro though 1 Euro = 90 INR.

The data samples are transformed according to the following equation :

$$Y = X \frac{R_{max} - R_{min}}{D_{max} - D_{min}} + \frac{R_{min} D_{max} - R_{max} D_{min}}{D_{max} - D_{min}}$$

Q.4 List some examples of time series data.

Ans. : Below are the examples of time series analysis,

- Electrical - impulse activity in the brain
- Rainfall measurements
- Stock prices
- Number of sunspots
- Annual retail sales
- Monthly subscribers

Q.5 Classify time series.

Ans. : Time series can be classified into two different types : Stock and flow.

A stock series is a measure of certain attributes at a point in time and can be thought of as "stocktakes". For example, the **monthly labour force survey** is a stock measure because it takes stock of whether a person was employed in the reference week.

Flow series are series which are a measure of activity over a given period. For example, surveys of **retail trade** activity. Manufacturing is also a flow measure because a certain amount is produced each day and then these amounts are summed to give a total value for production for a given reporting period.

The main difference between a stock and a flow series is that flow series can contain effects related to the calendar (trading day effects). Both types of series can still be seasonally adjusted using the same seasonal adjustment process.

❑❑❑

UNIT IV

4 Bivariate Analysis

Syllabus

Relationships between Two Variables - Percentage Tables - Analyzing Contingency Tables - Handling Several Batches - Scatterplots and Resistant Lines - Transformations.

Contents

4.1 Relationship between Two Variables

- The term **bivariate analysis** refers to the analysis of two variables. It is a methodical statistical technique applied to a pair of variables (features/ attributes) of data to determine the empirical relationship between them. In order words, it is meant to determine any concurrent relations (usually over and above a simple correlation analysis).
- **Bivariate analysis** is performed to find the relationship between each variable in the dataset and the target variable of interest (or) using 2 variables and finding the relationship between them. For example, Box plot, Violin plot.
- **Bivariate analysis** can be thought of as simple as creating a scatter plot by plotting one variable against another on a Cartesian plane (think X and Y axis) can sometimes give a picture of what the data is trying to show. If the data seems to fit a line or curve then there is a relationship or correlation between the two variables. For example, one might choose to plot caloric intake versus weight.
- There are three common ways to perform bivariate analysis :
  1. **Scatter plots** - This gives an idea of the patterns that can be formed using the two variables.
  2. **Correlation coefficients** - The coefficient helps to know if the data in question are related. When the correlation coefficient is zero then this means that the variables are not related. If the correlation coefficient is a positive or a negative 1 then this means that the variables are perfectly correlated.
  3. **Simple linear regression** - This uses a wide range of tools to determine how the data post could be related. The post may follow an exponential curve. The regression analysis gives the equation for a line or curve. It also helps to find the correlation coefficient.
- In the context of supervised learning, it can help determine the essential predictors when the bivariate analysis is done keeping one of the variables as the dependent variable (Y) and the other ones as independent variables (X_1, X_2, … and so on) hence plot all Y, X_s pairs. So essentially, it is a way of feature selection and feature prioritization.

Comparison of correlation and causality

- It is a widespread fallacy to assume that if one variable is observed to vary with a change in values of another empirically, then either of them is "causing" the other to change or leading the other variable to change. In bivariate analysis, it might be observed that one variable (especially the X_s) is causing Y to change. Still, in actuality, it might just be an indicator and not the actual driver.

Types of variable and bivariate analysis

- The type of bivariate analysis is dependent on the kind of attributes and variables that is used to analyze the data. The variables may be ordinal, categorical or numeric. The independent variable is categorical like a brand of a pencil. In this case, prob it regression or log it regression is used. If the dependent and the independent variables are both ordinal which means that they have a ranking or position then the rank correlation coefficient is measured.
- In case the dependent attribute is ordinal then the ordered probit or the ordered logit is used. It is possible that the dependent attribute could be internal or a ratio like the scale of temperature. This is where regression is measured. Below are the kinds of bivariate data correlation.
  1. **Numerical and Numerical :** In this kind of variable both the variables of the bivariate data which includes the dependent and the independent variable have a numerical value.
  2. **Categorical and Categorical :** When both the variables in the bivariate data are in the static form then the data is interpreted and statements and predictions are made about it. During the research, the analysis will help to determine the cause and impact to conclude that the given variable is categorical.
  3. **Numerical and Categorical :** This is when one of the variables is numerical and the other is categorical. Bivariate analysis is a kind of statistical analysis when two variables are observed against each other. One of the variables will be dependent and the other is independent. The variables are denoted by X and Y. The changes are analyzed between the two variables to understand to what extent the change has occurred.
- Further, there are two types of variables in data - Categorical and continuous (numerical). Therefore, for bivariate analysis, there are 3 possible combinations for analysis that could be carried out namely, categorical and categorical, categorical and continuous, continuous and continuous.

1. Categorical and categorical variables combination

- This is used in case both the variables being analyzed are categorical. In the case of classification, models say, for example, classifying a credit card fraud or not as Y variables and then checking if the customer is at his hometown or away or outside the country. Another example can be age vs gender and then counting the number of customers who fall in that category. It is important to note that the visualization / summary shows the count or some mathematical or logical aggregation of a 3rd variable / metric like revenue or cost and the like in all such analyses. It can be done using Crosstabs (heatmaps) or Pivots in Python.

- **Crosstabs :** It is used to count between categories or get summaries between two categories. Pandas library has this functionality.
- **Pivots :** Another useful functionality that can be applied to Pandas dataframes to get Excel like Pivot tables. This can work for 2+ categorical variables when placed in the proper hierarchy.

2. Categorical and continuous (numerical) variables combination :

- In this type the variance plotting of a numerical variable in a class is performed. For example, how age varies in each segment or how do income and expenses of a household vary by loan re-payment status.
- Categorical plot for aggregates of continuous variables : Used to get total or counts of a numerical variable eg revenue for each month. Also, this can be used for counts of another categorical variable too instead of the numerical.
- Plots for distribution of continuous (numerical) variables : Use to see the range and statistics of a numerical variable across categories.
- Plots used are - box plot, violin plot, swarm plot.

3. Continuous and continuous variable combination :

- This is the most common use case of bivariate analysis and is used for showing the empirical relationship between two numerical (continuous) variables. This is usually more applicable in regression cases.
- In case there is large datasets with 30 - 70+ features (variables), there might not be sufficient time to run each pair of variables through bivariate analysis one by one. One could use the pair plot or pair grid from the seaborn library in such cases. It makes a grid where each cell is a bivariate graph and Pair grid also allows customizations.

4.2 Percentage Tables

A bivariate table addresses the joint distribution of two variables. Bivariate table is a table that illustrates the relationship between two variables by displaying the distribution of one variable across the categories of a second variable.

To detect association within bivariate tables one can calculate percentages within the categories of the independent variable or can compare percentages across the categories of the independent variable or can perform a Chi Square test of Independence formally determines the statistical significance.

- **Cross-tabulation is a** technique used to explore the relationship between two variables that have been organized in a table. **Column variable is** a variable whose categories comprise the columns of a bivariate table. **Row variable is** a variable whose categories comprise the rows of a bivariate table. **Cell is** the intersection of a row and a column in a bivariate table. **Marginals is** the row and column totals in a bivariate table.
- A bivariate table displays the distribution of one variable across the categories of another variable. The independent variable usually goes in the columns, while the dependent variable goes in the rows. Rows and columns intersect at cells. The row and column totals of a bivariate table are called marginals.
- Bivariate relationships come in several different flavors. When the variation in the dependent variable can be attributed only to the independent variable, the relationship is said to be direct. When a third variable affects both the independent and dependent variables, the relationship is said to be spurious. When the independent variable affects the dependent variable only by way of a mediating variable (sort of like a chain reaction), it is said to be an intervening relationship.

The common percent is an extremely useful measure.

- One can use a percent with any kind of data : Nominal, ordinal, interval and ratio. A percent is a standardized measure. Percent means "per 100 cases."Because the percent is standardized one can use it to compare results from different population bases that have different sizes or total case - bases. For example, one could compare the percentage of home computer ownership among people with living in villages, urban areas and metro cities.
- The **first step** in calculating a percent is to **isolate the case base of interest**.
- The **second step** is to **identify the category of interest**.
- The **third step** is to **locate the number of cases in the category of interest ONLY for the group**.
- The **fourth step** is to **take the frequency in category of interest and divide that frequency by the total number of cases in my case - base of interest**.

Constructing a bivariate table :

- Percentages can be computed indifferent ways namely,
  1. Column percentages - Column totals as base.
  2. Row percentages - Row totals as base.
- Typically percentages are provided for the independent variables.

Elaboration

- Elaboration is a process designed to further explore bivariate relationships by introducing additional variables called control variables.

Limitations of elaboration

- Elaboration can be useful, but it also has its limitations. First, it tends to be a little bit tedious, especially if done by hand. Second, it's not the most precise form of analysis. Elaboration allows to compare the distribution of one variable across the categories of another, but there are other measures of association that do a better job of quantifying the relationship between two variables.

Crosstab function Python

- The crosstab() function is **used to compute a simple cross tabulation of two (or more) factors**. By default computes a frequency table of the factors unless an array of values and an aggregation function are passed.

pandas.crosstab(parameters)

index	array - like, Series or list of arrays / Series Values to group by in the rows.
columns	array - like, Series or list of arrays / Series Values to group by in the columns.
values	array - like, optional Array of values to aggregate according to the factors. Requires `aggfunc` be specified.
rownames	sequence, default None If passed, must match a number of row arrays passed.
colnames	sequence, default None If passed, must match a number of column arrays passed.
aggfunc	function, optional If specified, requires `values` be specified as well.
margins	bool, default False Add row / column margins (subtotals).
margin_name	str, default 'All' Name of the row / column that will contain the totals when margins is True.
dropna	bool, default True Do not include columns whose entries are all NaN.
normalize	bool, {'all', 'index', 'columns'} or {0,1}, default False Normalize by dividing all values by the sum of values.

#Program to build crosstable and build bar chart

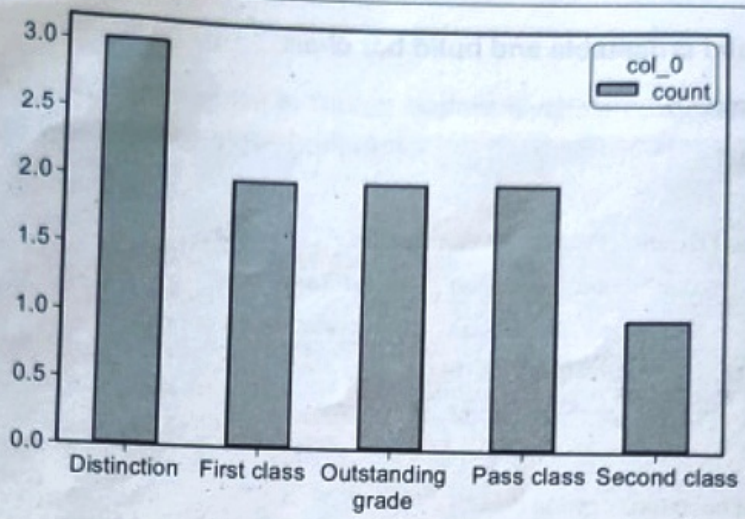
Example program - 1

```
import pandas as pd
#create data
df = pd.DataFrame({'Grade': ['FirstClass','Outstanding','Outstanding',
                             'Distinction','FirstClass', 'SecondClass',
                             'PassClass', 'PassClass', 'Distinction', 'Distinction'],
                   'Age': [18, 18, 18, 19, 19, 20, 18, 18, 19, 19],
                   'Gender': ['M','M', 'F', 'F', 'F', 'M', 'M', 'F', 'M', 'F']})
print(df)
#find frequency of each letter grade
grdcrosstb = pd.crosstab(index=df['Grade'], columns='count')
print(grdcrosstb)
# Creatingbarplot
barplot = grdcrosstb.plot.bar(rot=0)
```

Example program - 1 Output

	Age	Gender	Grade
0	18	M	FirstClass
1	18	M	Outstanding
2	18	F	Outstanding
3	19	F	Distinction
4	19	F	FirstClass
5	20	M	SecondClass
6	18	M	PassClass
7	18	F	PassClass
8	19	M	Distinction
9	19	F	Distinction

col_0	count
Grade	
Distinction	3
FirstClass	2
Outstanding	2
PassClass	2
SecondClass	1



#Building tables with crosstab function

Example program - 2

```
import pandas as pd
res_names = ['Vaishali', 'Rupali', 'Lucky', 'Gorge', 'BlueNyle', 'Goodluck', 'SPs']
purchase_type = ['Food', 'Food', 'Food', 'Drink', 'Food', 'Drink', 'Drink']
price = [12, 25, 32, 10, 15, 22, 18]
print ('Restaurant Names: {}'.format(res_names))
print ('Purchase Type: {}'.format(purchase_type))
print ('Price: {}'.format(price))
rescrtb1 = pd.crosstab(index=[res_names], columns=[purchase_type])
print (rescrtb1)
rescrtb2 = pd.crosstab(index=[res_names], columns=[purchase_type], values=price, aggfunc=sum)
print(rescrtb2)
rescrtb3 = pd.crosstab(index=[res_names],
columns=[purchase_type],
values=price,
aggfunc=lambda x: x.sum()**2, # Setting a custom agg function
rownames=["Restaurants"], # Giving a title to my rows
colnames=['Food Types'], # Giving a title to my columns
margins=True, # Adding margins (Subtotals on the ends)
        #margins_name="Totals
        ) # Give my subtotals a title
print(rescrtb3)
```

program - 2 Output

```
t Names: ['Vaishali', 'Rupali', 'Lucky', 'Gorge', 'BlueNyle', 'Goodluck', 'SPs']
Type: ['Food', 'Food', 'Food', 'Drink', 'Food', 'Drink', 'Drink']
 25, 32, 10, 15, 22, 18]
```

	Drink	Food
	0	1
	1	0
	1	0
	0	1
	0	1
	1	0
	0	1

	Drink	Food
aN	15.0	
	22.0	NaN
	0.0	NaN
Nal	N	32.0
	25.0	
.0	NaN	
N	12.0	

s	Drink	Food	All
s			
	NaN	225.0	225.0
	484.0	NaN	484.0
	100.0	NaN	100.0
	NaN	1024.0	1024.0
	NaN	625.0	625.0
	324.0	NaN	324.0
	NaN	144.0	144.0
	2500.0	7056.0	17956.0

Example program - 3

```
import pandas as pd
# Dictionary
df1 = {
    'Name': ['Dashrath', 'Ram', 'Bharat',
             'Laxman', 'Maruti', 'Shatru',
             'Bibhi'],
    'Math_score': [52, 87, 49,
                   74, 38, 59,
                   48]}
# Create a DataFrame
df1 = pd.DataFrame(df1,
columns = ['Name',
                        'Math_score'])
# Calculating Percentage
df1['percent'] = (df1['Math_score'] /
df1['Math_score'].sum()) * 100
# Show the dataframe
print(df1)
```

Example program - 3 Output

	Name	Math_score	percent
0	Dashrath	52	13.098237
1	Ram	87	21.914358
2	Bharat	49	12.342569
3	Laxman	74	18.639798
4	Maruti	28	7.052897
5	Shatru	59	14.861461
6	Bibhi	48	12.090680

4.3 Analyzing Contingency Tables

- **Contingency table** is the techniques for exploring two or even more variables. I basically a tally of counts between two or more categorical variables.
- A contingency table depicts the distribution of one variable in rows and another variabl columns. It is used to study the correlation between the two variables. It is a multiway t which describes a dataset in which each observation belongs to one category for eac

several variables. It is basically a tally of counts between two or more categorical variables. Contingency tables are also called crosstabs or two - way tables, used in statistics to summarize the relationship between several categorical variables.

- The contingency coefficient is a coefficient of association which tells whether two variables or datasets are independent or dependent of each other, also known as Pearson's Coefficient
- For example, suppose that there is data of 200 people whether they prefer Potato Vada or Misal and whether they prefer Curd or Coke. This data can be assembled into a contingency table, which might look like :

	Potato Vada	Misal
Curd	40	85
Coke	50	25

- When determining whether two variables are associated, it can be helpful to look at a contingency table of proportions. Contingency tables are often given in frequencies and can be converted to proportions by dividing each frequency by the total number of observations.

Marginal proportions

- A marginal proportion in a contingency table is the proportion of observations in a single category of one variable. If given a contingency table of proportions, the marginal proportion can be calculated by taking the row and column sums. If given a contingency table of frequencies, the marginal proportion can be calculated by dividing the row or column sum by the total number of observations.
- Contingency tables are a quite popular method during EDA (Exploratory Data Analysis) as they show relationship between two categorical fields. They are often used to see the distribution of a categorical field with respect to the class label in classification tasks. They are also used in statistical tests like chi-squared test which test for the association between two categorical variables.

Example program - 4

```
#Build contingency table using Python
import pandas as pd
# create a dataframe
df = pd.DataFrame({
'Jeans_id': [1, 2, 3, 4, 5, 6],
```

```
    'color': ['Red', 'Blue', 'Gray', 'Blue', 'Red', 'Red'],
    'size': ['M', 'S', 'L', 'L', 'S', 'M']
})
# display the dataframe
print(df)
# contingecy table between "color" and "size"
print(pd.crosstab(df['color'], df['size']))
# contingecy table with row and column totals
print(pd.crosstab(df['color'], df['size'], margins=True))
```

Example program - 4 Output

```
    Jeans_id  color  size
0          1    Red     M
1          2   Blue     S
2          3   Gray     L
3          4   Blue     L
4          5    Red     S
5          6    Red     M
size   L  M  S
color
Blue   1  0  1
Gray   1  0  0
Red    0  2  1
size   L  M  S  All
color
Blue   1  0  1  2
Gray   1  0  0  1
Red    0  2  1  3
All    2  2  2  6
```

4.4 Handling Several Batches

- Many analytics applications require frequent batch processing, which allows them to process data in batches at varying interval. Batch systems must be built to scale for all sizes of data and scale seamlessly to the size of the dataset being processed at various job runs.
- Such data received from batch system can be referred as big data as it is a large data set.

- While dealing with such data sets below techniques need to be applied,
- **Batch Processing** is a technique for processing large amounts of data in a repeatable manner. When computational resources are available, the batch technique allows users to process data with little or no human intervention. Simply described, batch processing is the method through which a computer completes batches of work in a nonstop, sequential manner, typically simultaneously. It's also a command that guarantees huge jobs are broken down into smaller chunks for debugging efficiency.

1. **Reduce memory usage by optimizing data types**
   - When using Pandas to load data from a file, it will automatically infer data types unless told otherwise. Most of the times it works fine but the inferred type is not necessarily optimized. Moreover, if a numerical column contains missing values than the inferred type will automatically be float. For particular case, specifying the data types led to an important reduction of the used memory.
2. **Split data into chunks**
   - When data is too large to fit into memory, one can use Pandas' *chunksize* option to split the data into chunks instead of dealing with one big block. Using this option creates an ***iterator*** object that can be used to go through the different chunks and perform filtering or analysis just like one would do when loading the full dataset.
   - Chunking can be used from initial exploratory analysis to model training and requires very little extra setup.
3. **Take advantage of lazy evaluation**
   - Lazy evaluation refers to the strategy which delays evaluation of an expression until the value is actually needed. Lazy evaluation is an important concept (used especially in functional programming).
   - Lazy evaluation is the basis on which distributed computation frameworks are built. Although they were designed to work on clusters one can still take advantage of them to handle large datasets on personal computer.
   - The main difference with respect to Pandas is that they do not load the data directly in memory. Instead what happens during the read command is that they scan the data, infer dtypes and split it into partitions (so far nothing new). Computational graphs are built for these partitions independently and they are executed only when really needed (hence the laziness).

- Steps to work with **Python Batch Processing** using **Joblib**. Joblib is a suite of Python utilities for lightweight pipelining. It contains unique optimizations for NumPy arrays and is built to be quick and resilient on large data. It is released under the Berkeley Source Distribution (BSD) license.
- Some of the key features of **Joblib** are :
  - **Transparent Disk-Caching of Functions and Lazy Re-Evaluation :** A memoize or make - like feature for Python functions that work with any Python object, including very big NumPy arrays. By expressing the operations as a collection of steps with well-defined inputs and outputs : Python functions, one can separate persistence and flow-execution logic from the domain logic or algorithmic code. Joblib can save their computation to disc and repeat it only if required.
  - **Simple Parallel Computing :** Joblib makes it easy to write readable parallel code and easily debug it.
  - **Fast Compressed Persistence :** An alternative for a pickle that works well with Python objects that contain a lot of data (joblib.dump&joblib.load).

enefits of Python Batch Processing

Speed and Low Costs : Batch Processing can lessen a company's dependency on other pricey pieces of technology, making it a comparatively low - cost option that saves money and time. Batch procedures are executed most efficiently and feasibly without the risk of user mistakes. As a result, managers have more time to focus on day-to-day operations and can analyze data more quickly and accurately.

Offline Features : Batch Processing systems work in a stand - alone mode. This process is still going strong at the end of the day. To prevent overloading a system and disturbing regular tasks, Managers can restrict when a process begins. The software can be configured to execute specific batches overnight, which is a practical option for firms that don't want jobs like automated downloads to disturb their daily operations.

Efficiency : When computers or other resources are readily accessible, Batch Processing allows a corporation to handle jobs. Companies can plan batch operations for activities that aren't as urgent and prioritize time - sensitive jobs. Batch systems can also run in the background to reduce processor burden.

Simplicity : Batch Processing, is a less sophisticated system that does not require particular hardware or system support for data entry. It requires less maintenance after it is set up than stream processing system.

mproved Data Quality : Batch Processing reduces the chances of mistakes by automating ost or all components of a processing operation and minimizing user contact. To achieve greater level of data quality, precision and accuracy are enhanced.

4.5 Scatter Plots and Resistant Lines

4.5.1 Bi-Variate Analysis using Scatter Plot

- A scatter plot can be used to visually inspect whether there is an association between two quantitative variables. If there is a pattern in the plot, the variables are associated; if there is no pattern, the variables are not associated. For example, this plot shows a pair of associated variables; children who are older tend to weigh more.

#Scatter plot for Bi-variate analysis - 2 Variables

Example program - 5

```
import pandas as pd
import matplotlib.pyplot as plt
#create DataFrame
df = pd.DataFrame({'HoursWalked': [1, 1, 1, 2, 2, 2, 3, 3, 3, 3,
                   3, 4, 4, 5, 5, 6, 6, 6, 7, 8],
          'Heartbeatscore': [75, 66, 68, 74, 78, 72, 85, 82, 90, 82,
                   80, 88, 85, 90, 92, 94, 94, 88, 91, 96]})

#View first five rows of DataFrame
df.head()
print(df)
#create scatterplot of hours vs. score
plt.scatter(df.HoursWalked, df.Heartbeatscore)
plt.title('Hours Walked vs. Heartbeat Score')
plt.xlabel('Hours Walked')
plt.ylabel('Heartbeat Score')
```

Example program - 5 Output

Heartbeat	scoreHours	Walked
0	75	1
1	66	1
2	68	1
3	74	2
4	78	2
5	72	2

4.5 Scatter Plots and Resistant Lines

4.5.1 Bi-Variate Analysis using Scatter Plot

- A scatter plot can be used to visually inspect whether there is an association between two quantitative variables. If there is a pattern in the plot, the variables are associated; if there is no pattern, the variables are not associated. For example, this plot shows a pair of associated variables; children who are older tend to weigh more.

#Scatter plot for Bi-variate analysis - 2 Variables

Example program - 5

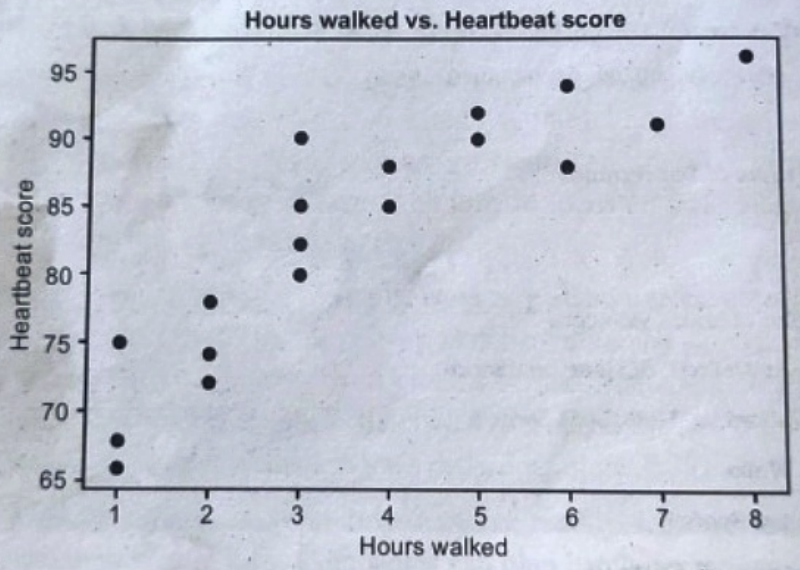
```
import pandas as pd
import matplotlib.pyplot as plt
#create DataFrame
df = pd.DataFrame({'HoursWalked': [1, 1, 1, 2, 2, 2, 3, 3, 3, 3,
                   3, 4, 4, 5, 5, 6, 6, 6, 7, 8],
          'Heartbeatscore': [75, 66, 68, 74, 78, 72, 85, 82, 90, 82,
                   80, 88, 85, 90, 92, 94, 94, 88, 91, 96]})

#View first five rows of DataFrame
df.head()
print(df)
#create scatterplot of hours vs. score
plt.scatter(df.HoursWalked, df.Heartbeatscore)
plt.title('Hours Walked vs. Heartbeat Score')
plt.xlabel('Hours Walked')
plt.ylabel('Heartbeat Score')
```

Example program - 5 Output

Heartbeat	scoreHours	Walked
0	75	1
1	66	1
2	68	1
3	74	2
4	78	2
5	72	2

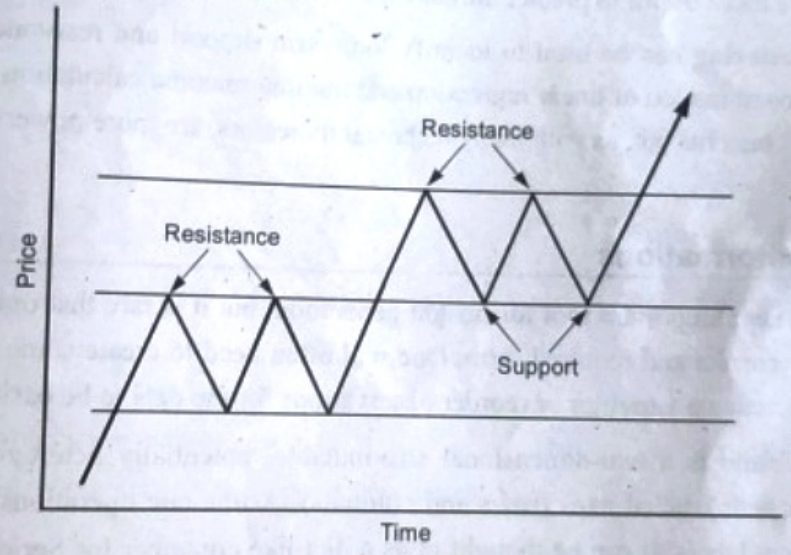
6	85	3
7	82	3
8	90	3
9	82	3
10	80	3
11	88	4
12	85	4
13	90	5
14	92	5
15	94	6
16	94	6
17	88	6
18	91	7
19	96	8



4.5.2 Bivariate Analysis Resistant Lines

- Resistant lines helps to find trend in the data and to identify support and resistance level in the data
- Resistance lines are technical indication tools used to determine the trend of a specific variable. They are very useful in predicting the probable movement of two variables. Resistance lines are usually drawn on a high-to-low basis. They help estimate resistance and support levels. A resistance line in an uptrend movement marks the support area and a resistance line in a downtrend movement marks the resistance area.

- Support and resistance levels are popular measures in technical analysis for stock trading. Support levels reflect price ranges at which a certain stock has trouble exceeding while resistance levels are those at which a stock's price tends not to fall below.
- Support and resistance levels are used in technical analysis to predict reversals in price trends. A falling price *might* be likelier to stop falling when it nears a support level. Conversely, a rising stock price *might* be likelier to stop increasing when it nears a resistance level. Support and resistance levels are not infallible and that, determining such price ranges is no simple task.
- Support levels become resistance levels once broken and likewise resistance levels become support levels when they are broken. As such, the same price levels can be either support *or* resistance depending on price action.



- In above figure it can be seen that a rising price "breaking through" a previous level to find a new range of resistance, after which the previous resistance level becomes a new support level. This figure illustrates how support and resistance levels can predict price reversals.
- For identifying the support and resistance levels, below are some common points to consider :
  - Horizontal vs. Diagonal
  - Intraday vs. Long Term
  - Major vs. Minor levels
  - Multiple Re-tests

- How one chooses to incorporate each of these considerations greatly influences the nature of how support and resistance levels are calculated. For example, diagonal support and resistance can be powerful in helping predict small pullbacks during an uptrend. In this case, "breaking support" can be identified as a potential trend reversal.

Calculating Support, Resistance and Trendlines

- There are various ways to calculate support and resistance. Below are two primary means :
  1. Long-term support and resistance levels, drawn as horizontal lines.
  2. Shorter-term trendlines, drawn as either diagonal or horizontal lines.
- **Long-term levels** are used to help predict large price reversals marking the start and completion of price movements on longer timelines such as the daily or weekly charts. Trendlines are more useful to predict intraday movements or shorter daily movements.
- **K-Means clustering** can be used to identify long-term support and resistance levels. For trendlines, a combination of linear regression and minima-maxima calculation is used. Each offer different benefits but, as with many technical indicators, are more powerful when used together.

4.6 Transformations

- Visualization is an important tool for insight generation, but it is rare that one gets the data in exactly the correct and required form. One will often need to create some new variables or summaries, rename variables or reorder observations for the data to be easier to manage.
- Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. It can be thought of as a dict-like container for Series objects. This is the primary data structure of the Pandas.
- Pandas**DataFrame.transform()** function call func on self producing a DataFrame with transformed values and that has the same axis length as self.

Pandas**DataFrame.transform()**Syntax

```
DataFrame.transform(func, axis=0, *args, **kwargs)
```

- **func :** It is a function that is used for transforming the data.
- **axis :** Refers to 0 or 'index', 1 or 'columns', default value 0.
- ***args :** It is a positional arguments that is to be passed to a func.
- ****kwargs :** It is a keyword arguments that is to be passed to a func.

Example program - 6

```
# importing pandas as pd
import pandas as pd

# Creating the DataFrame
df = pd.DataFrame({"Data1":[12, 4, 5, None, 1],
                   "Data2":[7, 2, 54, 3, None],
                   "Data3":[20, 16, 11, 3, 8],
                   "Data4":[14, 3, None, 2, 6]})

# Create the index
index_ = ['Row_1', 'Row_2', 'Row_3', 'Row_4', 'Row_5']

# Set the index
df.index = index_

# Print the DataFrame
print(df)

# add 10 to each element of the dataframe
result = df.transform(func = lambda x : x + 10)
#DataFrame.transform() function has successfully added 10 to each element of the given Dataframe.

# Print the result
print(result)
```

Example program 6 - Output

	Data1	Data2	Data3	Data4
Row_1	12.0	7.0	20	14.0
Row_2	4.0	2.0	16	3.0
Row_3	5.0	54.0	11	NaN
Row_4	NaN	3.0	3	2.0
Row_5	1.0	NaN	8	6.0

	Data1	Data2	Data3	Data4
Row_1	22.0	17.0	30	24.0
Row_2	14.0	12.0	26	13.0
Row_3	15.0	64.0	21	NaN
Row_4	NaN	13.0	13	12.0
Row_5	11.0	NaN	18	16.0

Review Questions with Answers

1. *What are common ways to perform bivariate analysis ?* **(Refer section 4.1)**
2. *Explain various types of bivariate analysis.* **(Refer section 4.1)**
3. *Discuss with example percentage tables.* **(Refer section 4.2)**
4. *What are the uses of contigency tables ?* **(Refer section 4.3)**
5. *How resistent lines are used in bivariate analysis ?* **(Refer section 4.5)**

4.7 Two Marks Questions with Answers

Q.1 What are possible types of correlation between two variables in bivariate analysis.

Ans. : Bivariate data correlation,

1. **Numerical and Numerical :** In this kind of variable both the variables of the bivariate data which includes the dependent and the independent variable have a numerical value.
2. **Categorical and Categorical :** When both the variables in the bivariate data are in the static form then the data is interpreted and statements and predictions are made about it. During the research, the analysis will help to determine the cause and impact to conclude that the given variable is categorical.
3. **Numerical and Categorical :** This is when one of the variables is numerical and the other is categorical. Bivariate analysis is a kind of statistical analysis when two variables are observed against each other. One of the variables will be dependent and the other is independent. The variables are denoted by X and Y. The changes are analyzed between the two variables to understand to what extent the change has occurred.

Q.2 Explain elaboration.

Ans. : Elaboration

Elaboration is a process designed to further explore bivariate relationships by introducing additional variables called control variables.

Limitations of elaboration

Elaboration can be useful, but it also has its limitations. First, it tends to be a little bit tedious, especially if done by hand. Second, it's not the most precise form of analysis. Elaboration allows to compare the distribution of one variable across the categories of another, but there are other measures of association that do a better job of quantifying the relationship between two variables.

Q.3 What are benefits of Python batch processing ?

Ans. : Benefits of Python batch processing are,

- **Speed and Low Costs :** Batch Processing can lessen a company's dependency on other pricey pieces of technology, making it a comparatively low - cost option that saves money and time. Batch procedures are executed most efficiently and feasibly without the risk of user mistakes. As a result, managers have more time to focus on day-to-day operations and can analyze data more quickly and accurately.
- **Offline Features :** Batch Processing systems work in a stand - alone mode. This process is still going strong at the end of the day. To prevent overloading a system and disturbing regular tasks, Managers can restrict when a process begins. The software can be configured to execute specific batches overnight, which is a practical option for firms that don't want jobs like automated downloads to disturb their daily operations.
- **Efficiency :** When computers or other resources are readily accessible, Batch Processing allows a corporation to handle jobs. Companies can plan batch operations for activities that aren't as urgent and prioritize time - sensitive jobs. Batch systems can also run in the background to reduce processor burden.
- **Simplicity :** Batch Processing, is a less sophisticated system that does not require particular hardware or system support for data entry. It requires less maintenance after it is set up than a stream processing system.
- **Improved Data Quality :** Batch Processing reduces the chances of mistakes by automating most or all components of a processing operation and minimizing user contact. To achieve a greater level of data quality, precision and accuracy are enhanced.

Q.4 What is cross tabulation ?

Ans. : Cross-tabulation is a technique used to explore the relationship between two variables that have been organized in a table. **Column variable** is a variable whose categories comprise the columns of a bivariate table. **Row variable** is a variable whose

categories comprise the rows of a bivariate table. **Cell is** the intersection of a row and a column in a bivariate table. **Marginals** is the row and column totals in a bivariate table.

A bivariate table displays the distribution of one variable across the categories of another variable. The independent variable usually goes in the columns, while the dependent variable goes in the rows. Rows and columns intersect at cells. The row and column totals of a bivariate table are called marginals.

The crosstab() function is **used to compute a simple cross tabulation of two (or more) factors**. By default computes a frequency table of the factors unless an array of values and an aggregation function are passed.

Q.5 What is marginal proportion in contingency table ?

Ans. : A marginal proportion in a contingency table is the proportion of observations in a single category of one variable. If given a contingency table of proportions, the marginal proportion can be calculated by taking the row and column sums. If given a contingency table of frequencies, the marginal proportion can be calculated by dividing the row or column sum by the total number of observations.

Contingency tables are a quite popular method during EDA (Exploratory Data Analysis) as they show relationship between two categorical fields. They are often used to see the distribution of a categorical field with respect to the class label in classification tasks. They are also used in statistical tests like chi-squared test which test for the association between two categorical variables.

❑❑❑

UNIT V

5 Multivariate and Time Series Analysis

Syllabus

Introducing a Third Variable - Causal Explanations - Three-Variable Contingency Tables and Beyond - Longitudinal Data - Fundamentals of TSA - Characteristics of time series data - Data Cleaning - Time-based indexing - Visualizing - Grouping - Resampling.

Contents

5.1 Introducing a Third Variable

- The data analysis involving three variables is termed as three variable analysis or more precisely it is termed as **multivariate analysis**. Multivariate analysis is defined as a process of involving multiple dependent variables resulting in one outcome. This explains that the majority of the problems in the real world are multivariate. For example, one cannot predict the weather of any year based on the season. There are multiple factors like pollution, humidity, precipitation and other related factor.
- **Multivariate analysis (MVA)** is a statistical procedure for analysis of data involving more than one type of measurement or observation. It may also mean solving problems where more than one dependent variable is analyzed simultaneously with other variables.

Five common relationships among three variables in a statistical model

- In a statistical model there is generally one way that a predictor X and a response Y can relate :

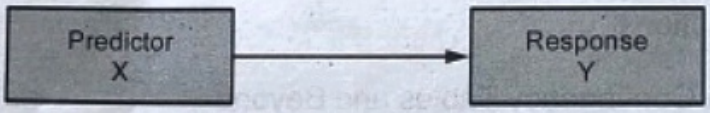


Fig. 5.1.1 Statistical relationship - 1

- This relationship can take on different forms, of course, like a line or a curve, but there is really only one relationship here to measure.
- Usually the point is to model the predictive ability, the effect of X on Y.
- In other words, there is a clear response variable*, although not necessarily a causal relationship. One could have switched the direction of the arrow to indicate that Y predicts X or used a two-headed arrow to show a correlation, with no direction, but that is a whole other story.
- In current text Y is the response variable and X the predictor.
- But a third variable another predictor can relate to X and Y in a number of different ways. How this predictor relates to X and Y changes the fact that how the relationship between X and Y is interpreted. These relationships have common names, but the names sometimes differ across fields, so one might be is familiar with a different names. In any case, the names are less important than :

1. Make sure that relationships being tested will answer the analysis questions.
2. The relationship reflects the data.

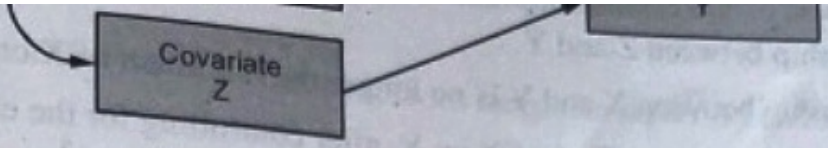


Fig. 5.1.2 Statistical relationship - 2

Because X and Z are not independent, there will usually be some joint effect on Y - some part of the relationship between X and Y that can not be distinguished from the relationship between Z and Y.

The relationship between X and Y is no longer the full effect of X on Y.

It is the marginal, unique effect of X on Y, after controlling for the effect of Z.

While the model fit as a whole will include both the joint and the unique effects of both X and Z on Y, the regression coefficient for X will only include its unique effect.

When both X and Z are observed variables, this is nearly always the situation.

As long as the correlation is moderate, it is still possible to measure the unique effect of X. If it gets too high, however, one will start to hit a point of multicollinearity in which the model has problems calculating estimates.

A good example of this kind of relationship would be in a study that measures the nutritional composition of soil cores at different altitudes and moisture levels.

Because water flows downhill, lower altitudes (X) tend to be more moist (Z). So while one can't completely separate altitude from moisture levels, as long as they are only moderately correlated, one should be able to find the unique effect of altitude on potassium levels.

Test this effect by including X and Z in a model together. To understand how much Z and X overlap in their explanation of Y, rerun the model without Z and look at how much X's coefficient changes.

Covariate correlated with X

- In this model, the covariate, Z is correlated with X and both predict Y.

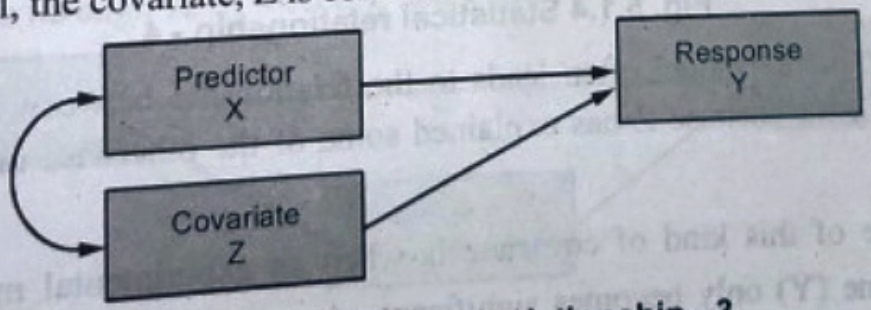


Fig. 5.1.3 Statistical relationship - 3

- Because X and Z are not independent, there will usually be some joint effect on Y - some part of the relationship between X and Y that can not be distinguished from the relationship between Z and Y.
- The relationship between X and Y is no longer the full effect of X on Y.
- It is the marginal, unique effect of X on Y, after controlling for the effect of Z.
- While the model fit as a whole will include both the joint and the unique effects of both X and Z on Y, the regression coefficient for X will only include its unique effect.
- When both X and Z are observed variables, this is nearly always the situation.
- As long as the correlation is moderate, it is still possible to measure the unique effect of X. If it gets too high, however, one will start to hit a point of multicollinearity in which the model has problems calculating estimates.
- A good example of this kind of relationship would be in a study that measures the nutritional composition of soil cores at different altitudes and moisture levels.
- Because water flows downhill, lower altitudes (X) tend to be more moist (Z). So while one can not completely separate altitude from moisture levels, as long as they are only moderately correlated, one should be able to find the unique effect of altitude on potassium levels.
- Test this effect by including X and Z in a model together. To understand how much Z and X overlap in their explanation of Y, rerun the model without Z and look at how much X's coefficient changes.

. Covariate independent of X

- When a covariate Z is NOT related to X, it has a slightly different effect. It needs to be able to predict Y as well to be useful in the model, but the effects of X and Z do not overlap.

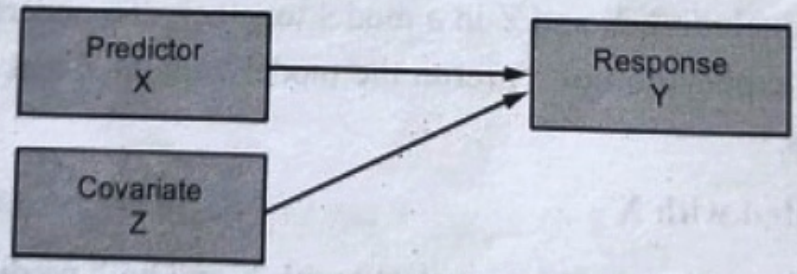


Fig. 5.1.4 Statistical relationship - 4

- Including Z in the model often leads to the relationship between X and Y becoming *more* significant because Z has explained some of the otherwise unexplained variance in Y.
- An example of this kind of covariate is when an experimental manipulation (X) on response time (Y) only becomes significant when one can control for finger dexterity levels (Z).

- If finger dexterity has a large effect on response time and one do not account for it, all of that variation due to dexterity will go into unexplained error - The denominator of the test statistic.
- Controlling for that variance means it is no longer unexplained and is removed from the denominator.
- Test this type of effect by running hierarchical regression. (add each predictor in a separate step)

3. Spurious relationship

- A confounding variable Z creates a spurious relationship between X and Y because Z is related to both X and Y.

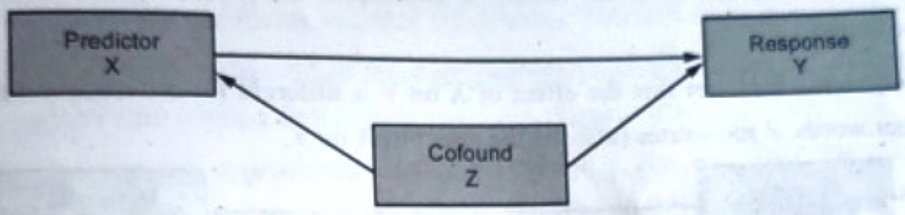


Fig. 5.1.5 Statistical relationship - 5

- This is the relationship seen in most "correlation is not causation" examples : The amount of ice cream consumption (X) in a month predicts number of shark attacks (Y). Do sharks like eating ice-cream laden people ? No.
- This spurious relationship is created by a confounder that leads to increases in both ice-cream consumption and shark attacks : Temperature (Z). People both eat more ice-cream and swim in the ocean in hotter months.
- It can be difficult to distinguish between this relationship and #1 above through testing alone. This is a situation where one has to entertain spuriousness as a possibility and question the results.

4. Mediation

- Mediation indicates a specific causal pathway. It occurs when at least part of the reason X affects Y is *through* Z. X affects Z and Z affects Y.

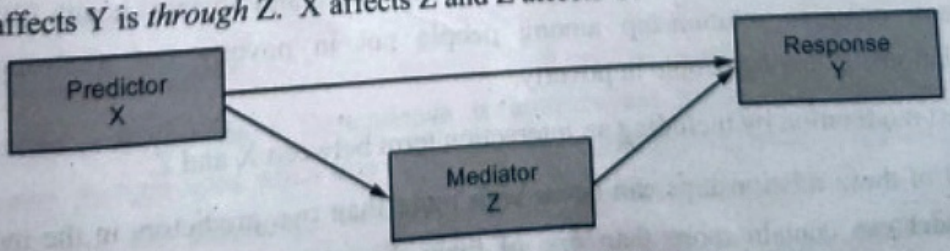


Fig. 5.1.6 Statistical relationship - 6

- There are then two effects of X on Y : The indirect effect of X on Y through Z and the direct effect of X on Y.
- An example here would be if the relationship between stress (X) and depression (Y) was mediated through increased levels of stress hormones (Z).
- There are a number of ways to test for mediation, including running a series of regression models and through path analysis. More modern approaches recommend testing the strength of the indirect effect.

5. Moderation

- Despite the similarity in the names, a moderation is an entirely different beast than mediation.
- Moderation indicates that the effect of X on Y is different for different values of Z. In other words, Z moderates (affects) the effect of X on Y.

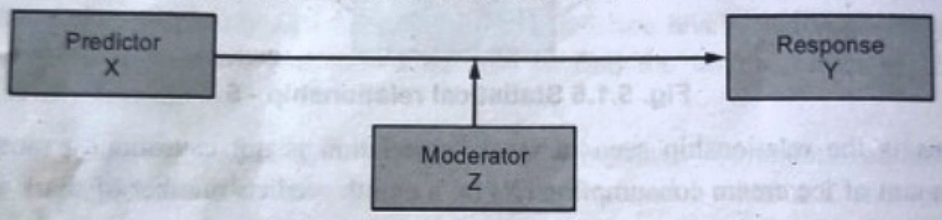


Fig. 5.1.7 Statistical relationship - 7

- Maybe when Z is large, there is a strong positive relationship between X and Y but when Z is small, there is no relationship between X and Y.
- So essentially, it can be said that X predicts Y in different ways, depending on the value of Z.
- X and Z can be correlated or not.
- One example of moderation can be seen in the relationship between depression (X), physical health scores (Y) and poverty status (Z). Poverty status (Yes/No) would be said to moderate the relationship between depression and physical health if there was a weak negative relationship among people not in poverty but a strong negative relationship among people in poverty.
- Test moderation by including an interaction term between X and Z.
- All of these relationships can occur with more than two predictors in the model and a model can contain more than one of them. Figuring out the potential relationships among variables is often *the fun part* of data analysis.

The objective of multivariate analysis

- Multivariate analyses are used principally for four reasons, i.e. to see patterns of data, to make clear comparisons, to discard unwanted information and to study multiple factors at once. Applications of multivariate analysis are found in almost all the disciplines which make up the bulk of policy-making, e.g. economics, healthcare, pharmaceutical industries, applied sciences, sociology and many more. Multivariate analysis has particularly enjoyed a traditional stronghold in the field of behavioural sciences like psychology, psychiatry and allied fields because of the complex nature of the discipline. Below listed are central goals of multivariate analysis.

1) **Data reduction or structural simplification :** This helps data to get simplified as possible without sacrificing valuable information. This will make interpretation easier.
2) **Sorting and grouping :** When there are multiple variables, groups of "similar" objects or variables are created, based upon measured characteristics.
3) **Investigation of dependence among variables :** The nature of the relationships among variables is of interest. Are all the variables mutually independent or are one or more variables dependent on the others.
4) **Prediction relationships between variables :** Must be determined for the purpose of predicting the values of one or more variables based on observations on the other variables.
5) **Hypothesis construction and testing :** Specific statistical hypotheses, formulated in terms of the parameters of multivariate populations, are tested. This may be done to validate assumptions or to reinforce prior convictions.

Multivariate analysis techniques

- Multivariate analysis technique can be classified into two broad categories, depending upon, the question - Are the involved variables dependent on each other or not ?
  - Dependence techniques
  - Interdependence techniques.

Dependence Techniques

- **Dependence technique :** Dependence techniques are types of multivariate analysis techniques that are used when one or more of the variables can be identified as dependent variables and the remaining variables can be identified as independent. Dependence methods are used when one or some of the variables are dependent on others. Dependence looks at cause and effect; in other words, can the values of two or more independent

variables be used to explain, describe or predict the value of another, dependent variable? To give a simple example, the dependent variable of "weight" might be predicted by independent variables such as "height" and "age."

- In machine learning, dependence techniques are used to build predictive models. The analyst enters input data into the model, specifying which variables are independent and which ones are dependent in other words, which variables they want the model to predict and which variables they want the model to use to make those predictions.

1. Multiple regression

- **Multiple regression analysis :** Multiple regression is an extension of simple linear regression. It is used when one wants to predict the value of a variable based on the value of two or more other variables. The variable one wants to predict is called the **dependent variable** (or sometimes, the outcome, target or criterion variable). Multiple regression uses multiple "x" variables for each independent variable : (x1)1, (x2)1, (x3)1, Y1)

2. Conjoint analysis

- **'Conjoint analysis'** is a survey-based statistical technique used in market research that helps determine how people value different attributes (feature, function, benefits) that make up an individual product or service. The objective of conjoint analysis is to determine the choices or decisions of the end-user, which drives the policy/product/service. Today it is used in many fields including marketing, product management, operations research, etc.
- It is used frequently in testing consumer response to new products, in acceptance of advertisements and in-service design. Conjoint analysis techniques may also be referred to as multi-attribute compositional modeling, discrete choice modeling or stated preference research and is part of a broader set of trade-off analysis tools used for systematic analysis of decisions.
- There are multiple conjoint techniques, few of them are CBC (Choice-based conjoint) or ACBC (Adaptive CBC).

3. Multiple discriminant analysis

- The objective of discriminant analysis is to determine group membership of samples from a group of predictors by finding linear combinations of the variables which maximize the differences between the variables being studied, to establish a model to sort objects into their appropriate populations with minimal error.

- Discriminant analysis derives an equation as a linear combination of the independent variables that will discriminate best between the groups in the dependent variable. This linear combination is known as the **discriminant function**. The weights assigned to each independent variable are corrected for the interrelationships among all the variables. The weights are referred to as discriminant coefficients.
- The discriminant equation :

$$F = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon$$

where, F is a latent variable formed by the linear combination of the dependent variable, $X_1, X_2, \ldots X_p$ is the p independent variable, ε is the error term and $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ is the discriminant coefficients.

4. A linear probability model

- A Linear Probability Model (LPM) is a regression model where the outcome variable is binary and one or more explanatory variables are used to predict the outcome. Explanatory variables can themselves be binary or be continuous. If the classification involves a binary dependent variable and the independent variables include non-metric ones, it is better to apply linear probability models.
- Binary outcomes are everywhere : Whether a person died or not, broke a hip, has hypertension or diabetes, etc.
- One typically want to understand what the probability of the binary outcome is given explanatory variables.
- One could actually use the linear model to do so, it is very simple to understand why. If Y is an indicator or dummy variable, then E[Y | X] is the proportion of 1s given X, which is interpreted as a probability of Y given X.

$$\text{Died}_i = \beta_0 + \beta_1 \text{age}_i + \varepsilon_i$$

- One can then interpret the parameters as the change in the probability of Y when X changes by one unit or for a small change in X. For example, if one models , one could interpret β_1 as the change in the probability of death for an additional year of age.

5. Multivariate analysis of variance and covariance

- Multivariate analysis of variance (MANOVA) is an extension of a common analysis of variance (ANOVA). In ANOVA, differences among various group means on a single-response variable are studied.

- Multivariate analysis of variance (MANOVA) is used to measure the effect of multiple independent variables on two or more dependent variables. With MANOVA, it is important to note that the independent variables are **categorical**, while the dependent variables are **metric** in nature. A categorical variable is a variable that belongs to a distinct category for example, the variable "employment status" could be categorized into certain units, such as "employed full-time," "employed part-time," "unemployed" and so on. A metric variable is measured quantitatively and takes on a numerical value.
- In MANOVA analysis, one is looking at various combinations of the independent variables to compare how they differ in their effects on the dependent variable.
- In MANOVA, the number of response variables is increased to two or more. The hypothesis concerns a comparison of vectors of group means. A MANOVA has one or more factors (each with two or more levels) and two or more dependent variables. The calculations are extensions of the general linear model approach used for ANOVA.
- ANOVA remains one of the most widely used statistical models in academia. Of the several types of ANOVA models, there is one subtype that is frequently used because of the factors involved in the studies. Traditionally, it has found its application in behavioural research, i.e. Psychology, Psychiatry and allied disciplines. This model is called the **Multivariate Analysis of Variance (MANOVA)**. It is widely described as the multivariate analogue of ANOVA, used in interpreting univariate data.

6. Canonical correlation analysis

- Canonical correlation analysis is the study of the linear relations between two sets of variables. It is the multivariate extension of correlation analysis.
- CCA is used for two typical purposes :
  - Data reduction
  - Data interpretation.
- One can compute all correlations between variables from the one set (p) to the variables in the second set (q), however interpretation is difficult when pq is large.
- Canonical correlation analysis allows us to summarize the relationships into a lesser number of statistics while preserving the main facets of the relationships. In a way, the motivation for canonical correlation is very similar to principal component analysis.

7. Structural equation modelling

- Structural equation modeling is a multivariate statistical analysis technique that is used to analyze structural relationships. It is an extremely broad and flexible framework for data analysis, perhaps better thought of, as a family of related methods rather than as a single technique.
- SEM in a single analysis can assess the assumed causation among a set of dependent and independent constructs i.e. validation of the structural model and the loadings of observed items (measurements) on their expected latent variables (constructs) i.e. validation of the measurement model. The combined analysis of the measurement and the structural model enables the measurement errors of the observed variables to be analyzed as an integral part of the model and factor analysis combined in one operation with the hypotheses testing.

Interdependence technique

- Interdependence techniques are a type of relationship that variables cannot be classified as either dependent or independent.
- It aims to unravel relationships between variables and/or subjects without explicitly assuming specific distributions for the variables. The idea is to describe the patterns in the data without making (very) strong assumptions about the variables.
- Interdependence methods are used to understand the structural makeup and underlying patterns within a dataset. In this case, no variables are dependent on others, so one is not looking for causal relationships. Rather, interdependence methods seek to give meaning to a set of variables or to group them together in meaningful ways.
- So : One is about the effect of certain variables on others, while the other is all about the structure of the dataset.

1. Factor analysis

- Factor analysis is a way to condense the data in many variables into just a few variables. For this reason, it is also sometimes called "**dimension reduction**". It makes the grouping of variables with high correlation. Factor analysis includes techniques such as principal component analysis and common factor analysis.
- This type of technique is used as a pre-processing step to transform the data before using other models. When the data has too many variables, the performance of multivariate techniques is not at the optimum level, as patterns are more difficult to find. By using factor analysis, the patterns become less diluted and easier to analyze.

2. Cluster analysis

- Cluster analysis is a class of techniques that are used to classify objects or cases into relative groups called **clusters**. In cluster analysis, there is no prior information about the group or cluster membership for any of the objects.
  - While doing cluster analysis, one first partitions the set of data into groups based on data similarity and then assign the labels to the groups.
  - The main advantage of clustering over classification is that it is adaptable to changes and helps single out useful features that distinguish different groups.
- Cluster analysis used in outlier detection applications such as detection of credit card fraud. As a data mining function, cluster analysis serves as a tool to gain insight into the distribution of data to observe the characteristics of each cluster.

3. Multidimensional scaling

- Multidimensional scaling (MDS) is a technique that creates a map displaying the relative positions of several objects, given only a table of the distances between them. The map may consist of one, two, three or even more dimensions. The program calculates either the metric or the non-metric solution. The table of distances is known as the **proximity matrix**. It arises either directly from experiments or indirectly as a correlation matrix.

4. Correspondence analysis

- Correspondence analysis is a method for visualizing the rows and columns of a table of non-negative data as points in a map, with a specific spatial interpretation. Data are usually counted in a cross-tabulation, although the method has been extended to many other types of data using appropriate data transformations. For cross-tabulations, the method can be considered to explain the association between the rows and columns of the table as measured by the Pearson chi-square statistic. The method has several similarities to principal component analysis, in that it situates the rows or the columns in a high-dimensional space and then finds a best-fitting subspace, usually a plane, in which to approximate the points.
- A correspondence table is any rectangular two-way array of non-negative quantities that indicates the strength of association between the row entry and the column entry of the table. The most common example of a correspondence table is a contingency table, in which row and column entries refer to the categories of two categorical variables and the quantities in the cells of the table are frequencies.

Interpretation of results

- Interpretation of results is probably the most difficult part in the technique. The relevant results are generally summarized in a table with an associated text. Appropriate information must be highlighted regarding :
  - Multivariate test statistics used
  - Degrees of freedom
  - Appropriate test statistics used
  - Calculated p-value ($p < x$).
- Reliability and validity of the test are the most important determining factors in such techniques.

Multivariate analysis applications

- Multivariate analysis is used in several disciplines. One of its most distinguishing features is that it can be used in parametric as well as non-parametric tests.
  1. **Parametric tests :** Tests which make certain assumptions regarding the distribution of data, that is, within a fixed parameter.
  2. **Non-parametric tests :** Tests which do not make assumptions with respect to distribution. On the contrary, the distribution of data is assumed to be free of distribution.

Sr. No.	Parametric tests	Non parametric tests
1.	Based on interval / ratio scale.	Based on nominal / ordinal scale.
2.	Outliers absent.	Outliers present.
3.	Uniformly distributed data.	Nonuniform data.
4.	Equal variance.	Unequal variance.
5.	Sample size is usually large.	Small sample size.

5.2 Causal Explanations

- Science is a deterministic approach that searches for causal relationships (explanations) as it seeks to discover whether X causes Y or whether the independent variable causes changes in the dependent variable.
- The causal explanation is referring not so much to the logic of a theory but rather to the explanation of the internal physical mechanism of phenomenon. "explaining the world and what is going on in it means, accordingly, laying bare its inner working, its underlying causal mechanisms."

- The principle of causal explanation requires investigation into the causal mediating mechanisms that underlie a relationship of interest. Causal explanation is valuable because it explains how and why an effect occurs and consequently, provides information regarding when and where the relationship can be replicated.
- Causal explanations involve cause and effect. Single events from the past or sequences of events from the past are explained by citing other events. The event to be explained is the "**effect**" and the other events included in the explanation are "**causes**". Causes generally occur prior to the effect. Causal explanations assert that the effect occurred because of the prior occurrence of the cause. For every effect, there are an indefinite number of causes.
- Causal explanations are usually expressed in terms of probability.
- Historians can never assert that an effect will always happen but only that it will probably happen. Historians therefore seldom use the term "cause" but instead employ terms like "influence," "may cause" etc.
- Causal explanations use equivocal terms to express the degree of uncertainty embedded in their causal explanation.
- Logically demonstrating that the occurrence of an event caused a later event to happen is a process fraught with all kinds of dangers.
- The biggest danger revolves around the issue of connections. Just because Event A occurred before Event B, it does not necessarily follow that A caused B to occur.

Establishing a causal relationships

- For establishing causal relationships below points must be considered,
  1. There must be a correlation between the two events. A correlation is the degree to which the occurrence of one event is associated with the occurrence of another event. It is the degree of interactivity between two or more events.
  2. There must be a proper temporal relationship in the occurrence of events. The cause must occur before the effect.
  3. There must be some sort of theoretical generalization to connect the two events. Theoretical generalizations are almost always implicit rather than explicit in causal explanations. But they are nonetheless an essential part of every logically valid causal explanation.

Logical pitfalls in causal relationships

- Causal relationships may prove to be incorrect under certain situations. Therefore below points needs to be considered while inferring from causal relationships.

1. **Mistaking correlation for cause (cum hoc, propter hoc) :** Correlation by itself can never establish cause. It can disestablish a cause but a correlation can occur without a cause. Correlation is only a necessary component of a causal explanation. It is not sufficient in itself to explain cause.
2. **Putting effect before the cause (pro hoc, propter hoc) :** The effect must always follow the cause and it must be separated by an interval of time (even if it is very short). Cause and effect cannot happen at exactly the same time.
3. **Reductionism :** It happens when a historian reduces complexity or diversity to uniformity in a causal explanation. Most common form is the confusion of a probable or possible cause with the prime cause. Mistakes one event in a chain of causative factors as the main cause of an effect.
4. **Indiscriminate pluralism :** Occurs when a number of causal factors are put forward to explain an event but they are neither carefully defined nor weighted. Just thrown at the reader and the result is hopeless confusion.
5. **The fallacy of identity :** There should not be the assumption that a cause must resemble its effect.
6. The fallacy of responsibility as a cause merges two different questions and demands a single answer. "why did it happen ?" with "who is to blame ?". For example the historians must always be careful not to allow their moral inclination to blame someone for an event to confuse their empirical attempt to identify the cause of the event.
7. Fallacy of mechanistic cause break down the components of a causal complex (i.e., isolate the various causes for an event) and analyze them independently from each other. Problem with this is that, most of time, various causes interact with each other to produce an event. Independently, they might not be that important but when they interact with other factors their importance becomes manifest.

Examples of social causal claims

- Population increase causes technological innovation.
- A free press causes a low incidence of famine.
- Transport systems cause patterns of commerce and habitation.
- New market conditions cause changes in systems of norms.

5.3 Three Variable Contingency Tables and Beyond

- Three-way contingency tables involve three binary or categorical variables.
- A three-way contingency table is a cross-classification of observations by the levels of three categorical variables. More generally, k-way contingency tables classify observations by levels of k categorical variables. Levels may be ordinal or nominal.
- A three-way contingency table is a cross-classification of observations by the levels of three categorical variables. Suppose there are three categorical variables,
  - X takes possible values 1, 2, ..., I
  - Y takes possible values 1, 2, ..., J
  - Z takes possible values 1, 2, ..., K
- If a triplet (X, Y, Z) is collected for each individual in a sample of size 'n' then the data can be summarized as a three-dimensional table.
- Let n_{ijk} be the number o units for which X = i, Y = j and Z = k. Then the vector of cell counts (n_{111}, n_{112}, ... n_{IJK}) can be arranged in a table whose dimensions are $I \times J \times K$.
- The two variable or also termed as two-way contingency tables are tables having I rows and J columns, forming a $I \times J$ matrix.
- If a third variable is included, the three-way contingency table is described as $I \times J \times K$, where I is the number of rows, J is the number of columns and K is the number of super ordinate columns each level containing a 2×2 matrix. The indexes i, j and k are used to enumerate the rows or columns for each dimension. Each within-cell count or joint proportion has three subscripts now, n_{ijk} and p_{ijk}.
- There are two general types of marginal counts (or proportions), with three specific marginals for each type.
- The first type of marginal count combines one of the three dimensions. Using the + notation from two-way tables, the marginals combining counts across either I, J or K dimensions are n_{+jk}, n_{i+k} or n_{ij+}, respectively. The second general type of marginal collapses two dimensions and is given as n_{i++}, n_{+j+} or n_{++k}. The total count (sample size) is n_{+++} (or sometimes just n). The simplest case is a $2 \times 2 \times 2$ shown below to illustrate the notation.

k = 1

n_{111}	n_{121}	n_{1+1}
n_{211}	n_{221}	n_{2+1}
n_{+11}	n_{+21}	n_{++1}

k = 2

n_{112}	n_{122}	n_{1+2}
n_{212}	n_{222}	n_{2+2}
n_{+12}	n_{+22}	n_{++2}

- It should be noted that marginal collapsing across levels of K are not depicted in above table.
- The three variables are referred to by letter names X,Y and Z. Depending on data set and need of the analysis, one may not need to assign explanatory (independent) and response dependent) roles to the variables for all hypotheses that might be tested, it can be easier to put the table into more familiar terms by imagining one response variable Y (with its J number of levels) and two explanatory variables, X and Z (with I and K number of levels, respectively). If considered this way, then this is a the conceptual equivalent of a ANOVA design, where Y is the (binary) dependent variable and X and Z are two independent variables. One common hypothesis of interest is to examine the X effect on Y at different levels of Z, which is the same question asked in the test of the two-way factorial ANOVA (e.g., is the X group difference on Y the same for Z_1 and Z_2?). It is important to note that this is the real analogy to ANOVA for at least one possible or common hypothesis that can be tested with three-way contingency table.

Partial tables

- To display three-way tables, we typically use a set of two-way tables. These are referred to as **partial tables**. There are three ways to do this :
  - Consider I, Y × Z tables for each level of X.
  - Consider J, X × Z tables for each level of Y.
  - Consider K, X × Y tables for each level of Z.
- There are two 2 × 2 tables above that make up the three-way table an X × Y table within Z_1 and an X × Y table within Z_2. These are known as partial tables.
- Although partial tables can be constructed for levels of the X or levels the Y variable also, it is most common to set up three-way tables in this manner and discuss the X × Y partial tables within levels (or strata) of Z. Marginals for the two-way partial tables can be expressed as conditional proportions, similar to the simple conditional proportion in the two-way case.
- The notation $p_{ij|k}$ denotes one type of conditional proportion, which is the within-cell count relative to the number of cases within one level of Z, $p_{ij|k} = n_{ijk} / n_{++k}$. One can also compute other conditional proportions, such as p_{ijk} or $p_{jk|i}$ and so on. If the two I × J contingency tables are considered separately, one would compute odds ratios as before, adding to the notation to indicate that the odds ratio is specific to one level of the Z variable. Following the notation X, Y and Z and the Greek letter θ ("theta") are used for the odds ratio.

$$\theta_{XY|Z1} = \frac{odds_{X-1|YZ1}}{odds_{X-0|YZ1}} = \frac{n_{211}/n_{221}}{n_{111}/n_{121}} = \frac{n_{111}\,n_{221}}{n_{211}\,n_{121}} = \frac{p_{X-1|YZ1}}{p_{X-0|YZ1}}$$

- To calculate these odds ratios for each level of Z and then average them taking into account the different number of cases in each stratum of Z, one should get a common odds ratio called the Mantel-Haenszel (MH) odds ratio.

$$\theta_{MH} = \frac{\sum_{k-1}^{k} (n_{11k}\,n_{22k}/n_{++k})}{\sum_{k-1}^{k} (n_{12k}\,n_{21k}/n_{++k})}$$

- The Mantel-Haenszel is the common odds ratio and assumes that the odds ratio does not differ across the levels of Z, so it may not be of interest itself in many circumstances. It can be used, however, if the test of whether the X and Y are associated overall by assessing the weighted average of this relationship over the levels of Z is of interest and the odds do not differ substantially across the levels.
- For example, one can look at the two-way (partial) table between sex and admission status for each department in the university admission data. This representation corresponds to a **conditional distribution** because the department is fixed within each table. For the first two departments,

Department A

		Admission status	
		Admitted	Rejected
Sex	Male	512	313
	Female	89	19

Department B

		Admission status	
		Admitted	Rejected
Sex	Male	353	207
	Female	17	8

Marginal Tables

- "+" is used to indicate summation over a subscript;

$$n_{ij+} = \sum_{k=1}^{k} n_{ijk}$$

- Then the vector of counts (n_{11+}, n_{12+}, ... , n_{1J+}) can be arranged into a table of I × J dimensions, referred to as the **marginal table** of X and Y. There are three possible two-way marginal tables resulting from one three-way table. Essentially, by summing over one variable, ignoring its association with each of the other variables. This idea can even extend to the marginal table of a single variable by summing over both of the other variables.

Example program - 1

```
#Build Crosstable
import pandas as pd
# create a dataframe
df = pd.DataFrame({
  'Country': ['India', 'UK', 'India', 'Japan', 'Turkey', 'India','UK',
        'Japan', 'Turkey', 'UK'],
  'World_Champion': ['Anil', 'Charles', 'Sachin', 'Chang', 'Ferit', 'Mahendra',
            'William', 'Bruce', 'Baris', 'Thomas'],
  'Game': ['Cricket', 'Badminton', 'Cricket', 'Karate', 'Tennis', 'Football',
       'Badmiton', 'Foostball','Tennis', 'Badmiton'],
  'Score_Points' : [900,850,950,750,775,800,980,750,875,780]
})
# display the dataframe
print('\n\n\n')
print(df)
df.head().T
df.columns

#Select data for the crosstab

#To analyze data and select the values which best represent the problem or question.
#Typical benefits of using crosstab or pivot tables are:
#   Data summary
#   Data aggregation
#   Grouping
#   Quick Reports
#   Data patterns
```

```
#Create cross-tabulation table
#To build crosstab.
#The cross-tabulation includes many different options and parameters
#which make it really powerful tool for data analysis.
print('\n\n\n')
print('CrossTab - 2 Variable')
crtab = pd.crosstab(df['World_Champion'], df['Country'])
print(crtab)

print('\n\n\n')
print('CrossTab - 3 Variable')
crtab1 = pd.crosstab([df['World_Champion'], df['Game']],df['Country'] )
print(crtab1)

print('\n\n\n')
print('CrossTab - Using Percentage and Total')
crtab2 = pd.crosstab(df['World_Champion'], df['Country'], margins=True,  normalize='index')
print(crtab2)
print('\n\n\n')
print('Use values from another column and aggregation function like sum, average')
importnumpy as np
crtab3 = pd.crosstab(df['World_Champion'], df['Country'], values=df.Score_Points, aggfunc=np.sum)
print(crtab3)
print('\n\n\n')
print('Using Group By')
cols = ['World_Champion', 'Country']
print(df.groupby(cols)[cols].count())
```

Example program - 1 Output

	Country	Game	Score_Points	World_Champion
0	India	Cricket	900	Anil
1	UK	Badminton	850	Charles
2	India	Cricket	950	Sachin
3	Japan	Karate	750	Chang

4	Turkey	Tennis	775	Ferit
5	India	Football	800	Mahendra
6	UK	Badmiton	980	William
7	Japan	Foostball	750	Bruce
8	Turkey	Tennis	875	Baris
9	UK	Badmiton	780	Thomas

CrossTab - 2 Variable

Country	India	Japan	Turkey	UK
World_Champion				
Anil	1	0	0	0
Baris	0	0	1	0
Bruce	0	1	0	0
Chang	0	1	0	0
Charles	0	0	0	1
Ferit	0	0	1	0
Mahendra	1	0	0	0
Sachin	1	0	0	0
Thomas	0	0	0	1
William	0	0	0	1

CrossTab - 3 Variable

Country		India	Japan	Turkey	UK
World_Champion	Game				
Anil	Cricket	1	0	0	0
Baris	Tennis	0	0	1	0
Bruce	Foostball	0	1	0	0
Chang	Karate	0	1	0	0
Charles	Badminton	0	0	0	1
Ferit	Tennis	0	0	1	0
Mahendra	Football	1	0	0	0
Sachin	Cricket	1	0	0	0
Thomas	Badmiton	0	0	0	1
William	Badmiton	0	0	0	1

CrossTab - Using Percentage and Total

Country	India	Japan	Turkey	UK
World_Champion				
Anil	1.0	0.0	0.0	0.0
Baris	0.0	0.0	1.0	0.0
Bruce	0.0	1.0	0.0	0.0
Chang	0.0	1.0	0.0	0.0
Charles	0.0	0.0	0.0	1.0
Ferit	0.0	0.0	1.0	0.0
Mahendra	1.0	0.0	0.0	0.0
Sachin	1.0	0.0	0.0	0.0
Thomas	0.0	0.0	0.0	1.0
William	0.0	0.0	0.0	1.0
All	0.3	0.2	0.2	0.3

Use values from another column and aggregation function like sum, average

Country	India	Japan	Turkey	UK
World_Champion				
Anil	900.0	NaN	NaN	NaN
Baris	NaN	NaN	875.0	NaN
Bruce	NaN	750.0	NaN	NaN
Chang	NaN	750.0	NaN	NaN
Charles	NaN	NaN	NaN	850.0
Ferit	NaN	NaN	775.0	NaN
Mahendra	800.0	NaN	NaN	NaN
Sachin	950.0	NaN	NaN	NaN
Thomas	NaN	NaN	NaN	780.0
William	NaN	NaN	NaN	980.0

Using Group By

World_Champion Country

World_Champion Country

Anil	India	1	1
Baris	Turkey	1	1

Bruce	Japan	1	1
Chang	Japan	1	1
Charles	UK	1	1
Ferit	Turkey	1	1
Mahendra	India	1	1
Sachin	India	1	1
Thomas	UK	1	1
William	UK	1	1

5.4 Longitudinal Data

- A dataset is **longitudinal** if it tracks the same type of information on the same subjects at multiple points in time. For example, a longitudinal study refers to an investigation where participant outcomes and possibly treatments or exposures are collected at multiple follow-up times.
- For example, part of a longitudinal dataset could contain specific students and their standardized test scores in six successive years.

Student Name	Grade 1 (2001) Raw Score	Grade 2 (2002) Raw Score	Grade 3 (2003) Raw Score	Grade 4 (2004) Raw Score	Grade 5 (2005) Raw Score	Grade 6 (2006) Raw Score
Ravi	339	350	361	366	381	390
Ani	332	343	350	351	351	355
Lahu	360	380	400	420	430	438

- The primary advantage of longitudinal databases is that they can measure ***change***. So one can estimate, for example, the effect of various factors on *improvement* in student achievement. One can also estimate the overall effectiveness of individual teachers by examining the performance of successive classes of students they teach, as well as examine the extent to which teacher effectiveness changes with experience or the composition of their class.
- The longitudinal data extend into the past as well as the present. So one can evaluate the effect of a specific policy by looking at, say, student performance or teacher turnover *before* as well as *after* the policy was introduced. Longitudinal data also allow us to use sophisticated **analytic strategies** to measure the impact of various policies with reasonable precision.

Benefits of longitudinal studies :

1. Incident events are recorded. A prospective longitudinal study measures the new occurrence of event. The timing of event onset can be correlated with recent changes in other factor impacting the event.
2. Prospective ascertainment of exposure/impact factors. For example in a prospective study participants can have their exposure status recorded at multiple follow-up visits. This can alleviate recall bias where subjects who subsequently experience disease are more likely to recall their exposure (a form of measurement error). In addition the temporal order of exposures and outcomes is observed.
3. Measurement of individual change in outcomes. A key strength of a longitudinal study is the ability to measure change in outcomes and/or exposure at the individual level. Longitudinal studies provide the opportunity to observe individual patterns of change.
4. Separation of time effects - Cohort, period, age. When studying change over time there are many time scales to consider. A longitudinal study with measurements at times $t_1, t_2, \ldots t_n$ can simultaneously characterize multiple time scales such as age and cohort effects using covariates derived from the calendar time of visit and the participant's birth year : The age of subject i at time t_j is age $i_j = (t_j - \text{birth}_i)$; and their cohort is simply $\text{cohort}_{ij} = \text{birth}_i$.

5.5 Fundamental of Time Series Analysis (TSA) and Characteristics of Time Series Data

- A time series is a collection of observations of well-defined data items obtained through repeated measurements over time. For example, measuring the value of retail sales each month of the year would comprise a time series. This is because sales revenue is well defined and consistently measured at equally spaced intervals. Data collected irregularly or only once are not time series.
- An observed time series can be decomposed into three components : The trend (long term direction), the seasonal (systematic, calendar related movements) and the irregular (unsystematic, short term fluctuations).
- Time series data is data that is collected at different points in time. This is opposed to cross-sectional data which observes individuals, companies, etc. at a single point in time.
- Because data points in time series are collected at adjacent time periods there is potential for correlation between observations. This is one of the features that distinguishes time series data from cross-sectional data.

- Time series forecasting is the use of statistical methods to predict future behavior based on historical data.
- Examples of time series analysis in action include :
  - Weather data
  - Rainfall measurements
  - Temperature readings
  - Heart rate monitoring (EKG)
  - Brain monitoring (EEG)
  - Quarterly sales
  - Stock prices
  - Automated stock trading
  - Industry forecasts
  - Interest rates.

Important characteristics to consider first

- Some important questions to first consider when first looking at a time series are :
  - Is there **a trend**, meaning that, on average, the measurements tend to increase (or decrease) over time ?
  - Is there **seasonality**, meaning that there is a regularly repeating pattern of highs and lows related to calendar time such as seasons, quarters, months, days of the week and so on ?
  - Are there **outliers** ? In regression, outliers are far away from the line. With time series data, outliers are far away from other data.
  - Is there a **long-run cycle** or period unrelated to seasonality factors ?
  - Is there **constant variance** over time or is the variance non-constant ?
  - Are there any **abrupt changes** to either the level of the series or the variance ?
- Time series can be classified into two different types : Stock and flow.
- A stock series is a measure of certain attributes at a point in time and can be thought of as "stocktakes". For example, the **monthly labour force survey** is a stock measure because it takes stock of whether a person was employed in the reference week.
- Flow series are series which are a measure of activity over a given period. For example, surveys of **retail trade** activity. Manufacturing is also a flow measure because a certain amount is produced each day and then these amounts are summed to give a total value for production for a given reporting period.

- The main difference between a stock and a flow series is that flow series can contain effects related to the calendar (trading day effects). Both types of series can still be seasonally adjusted using the same seasonal adjustment process.
- A time series is a sequence of data points recorded through time.
- Thus, when dealing with time series data, order matters. Specifically, values in a time series express a dependency on time. Consequently, if one changes the order of a time series, one may change the meaning of the data.
- Time series data have two important properties :
  - Data is measured sequentially.
  - Each time unit has at most one data measurement.
- Time series data can also be regular or irregular. For regular time series, the measurements are equally spaced in time. On the other hand, for irregular time series, the measurements may not happen at fixed time intervals. For instance, this typically happens in IoT applications, where sensors and networks might fail and missing data is expected to some degree.
- In addition, when doing time series forecasting, one usually have two goals :
  - First, to identify patterns that explain the behavior of the time series.
  - Second, to use these patterns to forecast (predict) new values.

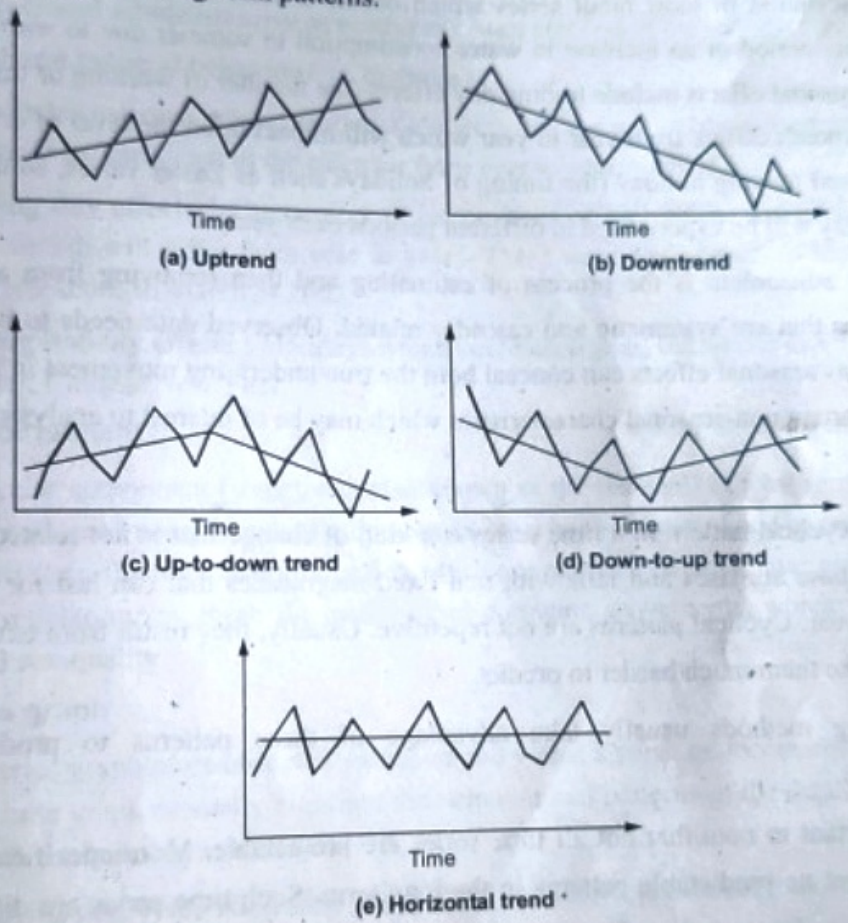
Simple forecast methods

- Time series forecasting has a rich family of algorithms. Some of the most basic ones include :
  - Average method
  - Moving average method
  - Naive method.
- These algorithms are simple to understand. Each takes a different assumption to predict new values.
- The average method assumes that a future event is best described by the average of all past events.
- The moving average method builds on simple average methods. Instead of using the average of all past events, it predicts a new event as the average over a predefined number of recent values. Lastly, the Naive method assumes that the next event will be equal to the most recent one.

Time series patterns

Trend

- The trend describes the general behavior of a time series. If a time series manifests a positive long-term slope over time, it has an upward trend. If instead, it describes a general negative slope, it has a downtrend.
- The overall trend may also change direction. There can be an up-to-down trend or a down-to-up trend. Lastly, a stationary or horizontal trend defines a time series with neither positive nor negative long-term patterns.



(a) Uptrend (b) Downtrend

(c) Up-to-down trend (d) Down-to-up trend

(e) Horizontal trend

Fig. 5.5.1 Trend in time series

Seasonal effects and seasonal adjustment

- A seasonal pattern is any kind of fluctuation (change) in a time series that is caused by calendar-related events.
- These events can be the time of year (like winter or summer) or the time of day or the week. Seasonality always has fixed frequencies. That is, a seasonal pattern always starts and ends in the same period of a week, year, etc.

- Take a data center as an example. If one considers the cooling system as the primary source of energy consumption, it is easy to imagine that in the summer, energy costs probably go up, while winter might show a decrease in energy consumption.
- Also, a clothing store that sells heavy coats might observe higher selling rates during winter, as opposed to the summer.
- A seasonal effect is a systematic and calendar related effect. Some examples include the sharp escalation in most retail series which occurs around December in response to the Christmas period or an increase in water consumption in summer due to warmer weather. Other seasonal effects include trading day effects (the number of working or trading days in a given month differs from year to year which will impact upon the level of activity in that month) and moving holiday (the timing of holidays such as Easter varies, so the effects of the holiday will be experienced in different periods each year).
- Seasonal adjustment is the process of estimating and then removing from a time series influences that are systematic and calendar related. Observed data needs to be seasonally adjusted as seasonal effects can conceal both the true underlying movement in the series, as well as certain non-seasonal characteristics which may be of interest to analysts.

Cycle

- Lastly, a cyclical pattern in a time series is a kind of change that is not related to seasonal factors. These are rises and falls with non-fixed magnitudes that can last for more than a calendar year. Cyclical patterns are not repetitive. Usually, they result from external factors which make them much harder to predict.
- Forecasting methods usually take advantage of these patterns to produce reliable predictions.
- It is important to note that not all time series are predictable. More specifically, some of them present no predictable patterns in the long term. Such time series are difficult, if not impossible, to forecast since future movements are equally likely to be up or down.
- To forecast this kind of data, one usually uses the random walk model. This model assumes that the next event is completely uncorrelated from the previous one. Hence, forecasts from a random walk model are equal to the last observation plus some noise. Random walk models are typically used with financial and economic data.

Seasonality

- The seasonal component consists of effects that are reasonably stable with respect to timing, direction and magnitude. It arises from systematic, calendar related influences such as :
  - **Natural conditions :** Weather fluctuations that are representative of the season (uncharacteristic weather patterns such as snow in summer would be considered irregular influences).
  - **Business and administrative procedures :** Start and end of the school term.
  - **Social and cultural behaviour :** Christmas.
- It also includes calendar related systematic effects that are not stable in their annual timing or are caused by variations in the calendar from year to year, such as :
  - **Trading day effects :** The number of occurrences of each of the day of the week in a given month will differ from year to year - There were 4 weekends in March in 2000, but 5 weekends in March of 2002.
  - **Moving holiday effects :** Holidays which occur each year, but whose exact timing shifts - Easter, Chinese New Year.

Irregular component

- The irregular component (sometimes also known as the residual) is what remains after the seasonal and trend components of a time series have been estimated and removed. It results from short term fluctuations in the series which are neither systematic nor predictable. In a highly irregular series, these fluctuations can dominate movements, which will mask the trend and seasonality.

Time series graph

- A time series graph plots observed values on the y-axis against an increment of time on the x-axis. These graphs visually highlight the behavior and patterns of the data and can lay the foundation for building a reliable model.
- More specifically, visualizing time series data provides a preliminary tool for detecting if data :
  - Is mean-reverting or has explosive behavior.
  - Has a time trend.
  - Exhibits seasonality.
  - Demonstrates structural breaks.
- This, in turn, can help guide the testing, diagnostics and estimation methods used during time series modeling and analysis.

Mean reverting data

- Mean reverting data returns, over time, to a time-invariant mean. It is important to know whether a model includes a non-zero mean because it is a prerequisite for determining appropriate testing and modeling methods.
- For example, unit root tests use different regressions, statistics and distributions when a non-zero constant is included in the model.
- A time series graph provides a tool for visually inspecting if the data is mean-reverting and if it is, what mean the data is centered around. While visual inspection should never replace statistical estimation, it can help to decide whether a non-zero mean should be included in the model.
- For example, the data in the figure above varies around a mean that lies above the zero line. This indicates that the models and tests for this data must incorporate a non-zero mean.

Structural breaks

- Sometimes time series data shows a sudden change in behavior at a certain point in time. For example, many macroeconomic indicators changed sharply in 2008 after the start of the global financial crisis. These sudden changes are often referred to as structural breaks or non-linearities.
- These structural breaks can create instability in the parameters of a model. This, in turn, can diminish the validity and reliability of that model.
- Though statistical methods and tests should be used to test for structural breaks, time series plots can help for preliminary identification of structural breaks in data.
- Structural breaks in the mean of a time series will appear in graphs as sudden shifts in the level of the data at certain breakpoints. For example, in the time series plot above there is a clear jump in the mean of the data around the start of 1980.

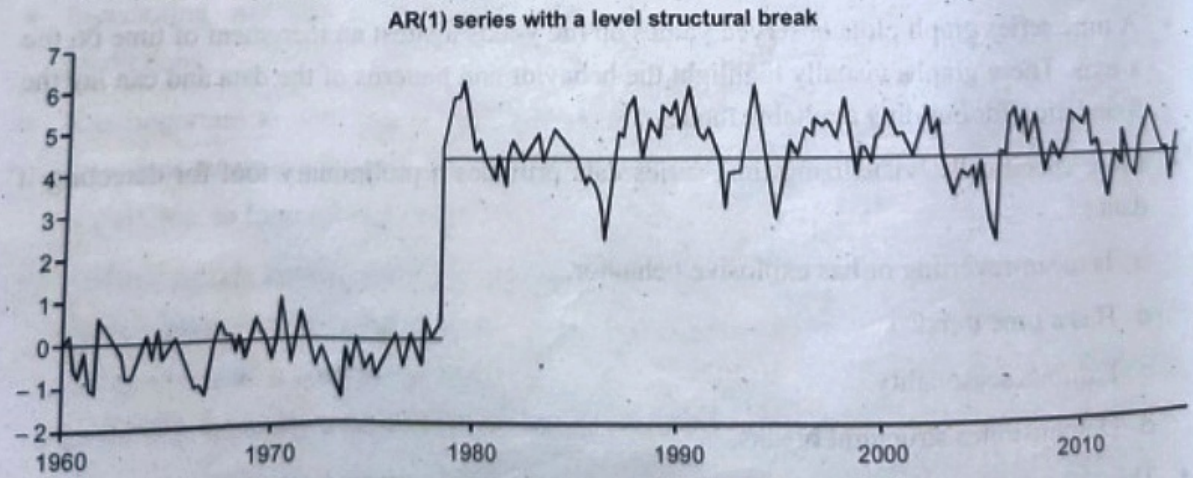


Fig. 5.5.2 Structural breaks in time series

Modeling time series data

- Time series models are used for a variety of reasons - Predicting future outcomes, understanding past outcomes, making policy suggestions and much more. These general goals of time series modeling do not vary significantly from modeling cross-sectional or panel data. However, the techniques used in time series models must account for time series correlation.

Time-domain versus frequency domain models

- Two broad approaches have been developed for modeling time series data, the time-domain approach and the frequency-domain approach.
- The time-domain approach models future values as a function of past values and present values. The foundation of this approach is the time series regression of present values of a time series on its own past values and past values of other variables. The estimates of these regressions are often used for forecasting and this approach is popular in time series econometrics.
- Frequency domain models are based on the idea that time series can be represented as a function of time using sines and cosines. These representations are known as Fourier representations. Frequency domain models utilize regressions on sines and cosines, rather than past and present values, to model the behavior of the data.

Univariate versus multivariate time series models

- Time series models may also be split into univariate time series models and multivariate time series models. Univariate time series models are models used when the dependent variable is a single time series. Trying to model an individual's heart rate per minute using only past observations of heart rate and exogenous variables is an example of a univariate time series model.
- Multivariate time series models are used when there are multiple dependent variables. In addition to depending on their own past values, each series may depend on past and present values of the other series. Modeling U.S. gross domestic product, inflation and unemployment together as endogenous variables is an example of a multivariate time series model.

Univariate model examples	Multivariate model examples
Univariate Generalized autoregressive conditional heteroscedasticity (GARCH).	Vector Autoregressive Models (VAR).
Seasonal Autoregressive Integrated Moving Average (SARIMA) Models.	Vector Error Correction Model (VECM).
Univariate unit root tests.	Multivariate unit root tests.

- When autocorrelation is present, there are two options for finding robust standard errors. The first approach estimates an OLS model and modifies the standard errors afterward. The Newey-West (1987) method is the standard approach for modifying the OLS standard errors to produce heteroskedastic and autocorrelation consistent standard errors.
- The alternative for dealing with autocorrelation in time series data is to re-weight the data prior to estimation. One method for doing this is generalized least squares which applies least squares to data that has been transformed by weights. Generalized least squares requires that the true parameters of autocorrelation be known.
- More generally, the true parameters of autocorrelations are unknown and must be estimated using Feasible Generalized Least Squares (FGLS). Feasible generalized least squares requires estimation of the covariance matrix.
- When the covariance structure is assumed the method is known as weighted least squares. Alternatively, the covariance structure can be estimated using iterative methods.

Detect autocorrelation

- Detecting autocorrelation in time series data can be done in a number of ways. One preliminary measure for detecting autocorrelation is a time series graph of residuals versus time. If no autocorrelation is present the residuals should appear random and scattered around zero. If there is a pattern in the residuals present, then autocorrelation is likely.

The Durbin-Watson test

- If the time series plot suggests autocorrelation, then further statistical tests can be used to formally test for autocorrelation.
- The Durbin-Watson test is a test of the null hypothesis of no first-order autocorrelation against the alternative that the error term in one period is negatively or positively correlated with the error term in the previous period.
- The Durbin-Watson test :

1. Estimate the model using ordinary least squares.
2. Predict the dependent variable using parameter estimates from step one.
3. Compute the residuals by subtracted predicted dependent variables from the observed dependent variable.
4. Square and sum the residuals.
5. Compute the difference between the residual at each time period, t and the previous time period, t – 1. Then square the differences and find the sum.

6. Compute the Durbin-Watson statistic by dividing the sum from Step Five by the sum in step four.

- Upper and lower critical values for the Durbin-Watson statistic depend on the number of independent variables.

Time series technique : The Box-Jenkins ARIMA method

- The Box-Jenkins method for ARIMA modeling is a procedure for consistently identifying, estimating and evaluating ARIMA models.

What Is an ARIMA Model ?

- The Autoregressive Integrated Moving Average Model (ARIMA) is a fundamental univariate time series model. The ARIMA model is made up of three key components :
  - The *autoregressive component* is the relationship between the current dependent variable the dependent variable at lagged time periods.
  - The *integrated component* refers to the use of transforming the data by subtracting past values of a variable from the current values of a variable in order to make the data stationary.
  - The *moving average* component refers to the dependency between the dependent variable and past values of a stochastic term.
- The ARIMA data is described by the order of each of these components with the notation ARIMA(p, d, q) where :
  - p is the number of autoregressive lags included in the model.
  - d is the order of differencing used to make the data stationary.
  - q is the number of moving average lags included in the model.

What is the Box-Jenkins method for ARIMA models ?

- The Box-Jenkins method for estimating ARIMA models is made up of several steps :

1. Transform data so it meets the assumption of stationarity.
2. Identify initial proposals for p, d and q.
3. Estimate the model using the proposed p, d and q.
4. Evaluate the performance of the proposed p, d and q.
5. Repeat steps 2 - 4 as needed to improve model fit.

Transforming the data for stationarity

- The first step of the Box-Jenkins model involves :
  1. Performing unit root tests to confirm the stationarity of the time series data.
  2. Taking the proper order of differencing in the case that the raw data is not stationary.
  3. Retesting for stationarity after the data has been differenced.

How to identify the order of an ARIMA model

- Identifying the autoregressive and moving average orders of the ARIMA model can be done using a variety of statistical tools :
  - Patterns in the autocorrelation function (ACF) and the partial autocorrelation function (PACF).
  - The Box-Pierce and Box-Ljung tests of joint significance of autocorrelation coefficients.
  - The Akaike Information Criterion (AIC) and Bayesian (Schwarz) criterion (BIC or SIC).

What are the methods for estimating ARIMA models ?

- Techniques for ARIMA model include :
  - Least squares nonlinear and linear regression.
  - Maximum likelihood methods.
  - Generalized method of moments.

Cross-covariance and cross-correlation functions

- Two key characteristics of the univariate time series model are the autocorrelation function and the covariance. The autocorrelation function measures the correlation of a univariate series with its own past values. The covariance measure the joint variability of the dependent time series with other variables.
- The analogies of these in the multivariate time series model are the cross-covariance and the cross-correlation. These measures provide insight into how the individual series in a group of time series are related.

Cross-correlation function

- The time series cross-correlation function measures the correlation between one series at various points in time with the values in another series at various points in time.

- **The cross-correlation function :**
  - Is scaled with values between – 1 and 1;
  - Measures how strongly two time series are related;
  - Shows how the relationship between variables changes across time.

Cross-covariance function

- The time series cross-covariance measures the covariance between values in one time series with values of another time series.
- **The cross-covariance function :**
  - Is an unscaled measure;
  - Reflect the direction and scale of comovement between two series.
- **Box-Jenkins multivariate models :** Multivariate models are used to analyze more than one time-dependent variable, such as temperature and humidity, over time.
- **Holt-Winters method :** The Holt-Winters method is an exponential smoothing technique. It is designed to predict outcomes, provided that the data points include seasonality.

5.6 Data Cleaning

5.6.1 Data Cleaning Concept and Methods

- In multivariate analysis there is always a margin for potential error that does not relate to the design of the study, its conduct or its implementation of various strategies aimed at preventing mistakes. These errors slip through the system and have the potential of obfuscating the results and leading to wrong conclusions. While it is impossible to eliminate every error that could potentially affect the research, there are several data-cleaning strategies that could correct these errors or minimize their negative impact on the study.
- Data cleaning is the process of identifying and correcting inaccurate records from a dataset along with recognizing unreliable or irrelevant parts of the data. Understanding data, cleansing data and dataset generation are important first steps in exploratory data analysis.
- The difference between data cleaning and error-prevention strategies lie in the fact that the former eliminates data-related problems once they have occurred, while the latter tries to prevent errors from happening in the first place. A standard data-cleaning strategy for multivariate data involves a three-stage process of screening, diagnosing and editing or eliminating any suspected data abnormalities. This process can be initiated at any stage of

the analysis. While it is possible to correct errors while the research is being performed, on the spot, it is recommended to actively and systematically screen for any errors in a planned way, in order to ensure that all data is adequately scanned and analyzed. Data cleaning process can be done manually or with the assistance of specialized software.

Data quality criteria

- Before data can be cleaned one needs to be able to identify if it needs to be cleaned. This can be done be examining the data quality criteria,
  - Validity
  - Accuracy
  - Completeness
  - Consistency
  - Uniformity.

1. Validity

- It is possible for data to be correct, but invalid. For example this is a correct date 1900-01-01. However if this was the date a document was received it is likely to be incorrect, one is conducting historical data analysis of the beginning of the twentieth century.

2. Accuracy

- To determine if data is accurate one should have an external data source to compare it to. For example, to know if an address reference code (postal code) is correct for an address the data would need to be compared to a database of addresses and code.

3. Completeness

- Data may have missing values. If the data was derived from another source one may be able to go back and try to get the required data or at least understand why it was missing. Otherwise one can either leave it blank or try to replace it with a value derived from statistical analysis of all the other values in this field.

4. Consistency

- Consistency is a measure of how the same data collected by different systems is the same or different. For example a Bank will collect a person's age in many different financial products and situations. If the person's age is the same in all systems in all places then it is consistent. However a person may give a different age when talking to someone in a call centre than when buying a pension where the retirement age is important to them.

5. Uniformity

- Different units of measure are used in different countries. Dates also have different formats and may even refer to different calendars. When data is brought together, they must all conform to a common unit of measure to be processed together.

6. Irrelevant data

- By understanding the data and framing the business problem one can identify fields and rows that cannot contribute to the final answer. So if one is searching for ideal colour combinations in girl's winter wears then one can probably safely drop all records for boys.

7. Duplicates

- Repeated data can appear when combining datasets from different systems or from human input. The record or fields may not be completely identical. One will need to compare records on important fields or with a fuzzy match.

8. Type conversion

- Many data types may appear as text. Often this is not as useful as if they were in their original format for example a number in a numeric format can be statistically analysed, but this cannot be done if it is a textual.

9. Syntax errors

- Syntax errors often appear in manually entered data sources where there can be some scope to enter correct data in a variety of ways. Validation in the user interface should minimise this.

10. Standardise

- Ensure all fields have consistent units. This is particularly important with internationally derived data where different measuring systems are used such as metric weights and measures.

11. Missing values

- Missing values and outliers can distort results and therefore effect how the analysis model makes predictions. By understanding the data one can find out the reason behind the data's absence. The empty field could be an optional field from customer data input or it could be a mutually exclusive field paired with another where only one contains data. A field with missing data could be null or have an indicator to show the data is absent for example :

- NaN (Not a Number)
- NA (Not applicable)
- None
- ?

- Missing data could be ignored or the whole record could be dropped. However this could distort the results. The alternative is to replace the data. There are four methods that can be employed to replace missing data :
  - Mean - Use the average value to replace missing values.
  - Median - Use the middle value of all values to replace missing values.
  - Mode - Use the most common value to replace missing values.
  - Use a machine learning algorithm to predict the missing values depending on other values in the data.

Detecting outliers

- Outliers are extreme values on one variable (univariate outlier) or a combination of variables (multivariate outlier), that distort or obscure the results of analyses.
- **Outliers can be the result of :**
  - Data entry errors
  - Invalid missing data coding
  - The participant did not complete the task/measure correctly
  - Case sampled is not from the intended population
  - Case sampled is from the intended population, but simply represents an extreme value within that population i.e. a genuine outlier.

Multivariate outliers

- Outliers across two or more variables.
- Best assessed using formal statistical procedures rather than graphical methods of detection.
- Mahalanobis distance is the distance of a case from the centroid of the remaining cases (centroid is the intersection of the variable means).
- Larger values are more outlying.
- The MD is tested using a chi squared (χ^2) distribution, with a conservative value of alpha (usually $p < .001$)

Steps in data cleaning

1. Screening phase

- During the data screening process in a multivariate analysis, one has to detect and distinguish five basic types of anomalous data : Lacking data, excessive data, outliers and inconsistencies, strange patterns and distributions, unexpected analysis results and other types of interferences and potential errors. Screening methods can be statistical and non-statistical. Nonconformities in outliers are often detected when compared to prior expectations based on experience, evidence in the literature, previous studies or common sense.
- Detecting erroneous inliers or data that falls within the expected range is harder. Erroneous inliers often escape the screening process. In order to detect erroneous inliers in a multivariate research, they need to be viewed in relation to other variables using regression analyses and consistency checks. Remeasurement is a recommended strategy to deal with erroneous inliers, but it is not always feasible. Another strategy involves examination of a multitude of inliers in order to approximately estimate the number of potential errors in the analysis.
- Below are some of the screening methods,
  - Checking data for double entry.
  - Graphical presentation and exploration of the datasets using histograms, diagrams, scatterplots, etc.
  - Frequency distributions.
  - Cross-tabulations.
  - Statistical methods for outlier detection.

2. Diagnostic phase

- After the suspected erroneous data points have been identified in the screening phase, it is necessary to determine their nature. There are several potential outcomes of this process - The data points could be erroneous, normal (if the expectations were incorrect), true extreme or undecided (without any explanation or reason for the extremity to occur). Some of these data points can be singled out on the grounds of being logically or biologically impossible. In many cases, several diagnostic procedures may be required in order to determine the true nature of every troublesome data pattern.
- Depending on the number and nature of errors in a multivariate data, the analyst might be required to reconsider their expectations for the results. In addition, quality control procedures may need to be reviewed and adjusted. This is why the diagnostic phase is considered to be a very labor-intensive and expensive procedure. The costs of data diagnostics can be lowered if the data-cleaning process is implemented throughout the entire research, rather than after it has been concluded. Data diagnostic software may be implemented to speed up the process.

atment phase

After the diagnostic stage is completed, all of the possible suspected errors in a multivariate lata are identified either as actual errors, missing values or true values. There are only three ptions to deal with each correcting, deleting or keeping the data unchanged. The latter is mplemented towards true values, as removing true extremes from the research would nevitably obfuscate results. Values that are physically or logically impossible are never left nchanged. They should be either removed or corrected, if possible. In the case with two ata points measured within a short period of time that only have small variations between ach other, it is recommended to take an average between the two, in order to enhance data ccuracy.

is possible to correct the measurements of every individual component in order to ensure at any suspicious data is either accurate or erroneous. Extremities should be kept in the search if there have been no errors in the test samples or the measurements, as they present a percentage of statistical probability that would otherwise obfuscate the research. owever, in order to ensure that the results of the research are accurate, an additional reening must be made for any extraneous influences that could have been unaccounted .

.6.2 Data Cleaning using Time-based Indexing, Visualizing, Grouping and Resampling

ne-based indexing can help to classify data based on time slices/range to indentify orrect data. **Grouping** of data can help to find similarity and dissimilarity or non-ociation among the data. If data has multiple errors then it could is good idea to **ample** the data to get data again in required form.

analyzing and **visualizing** the data using statistical methods such as mean, standard iation, range or quantiles, one can find values that are unexpected and thus erroneous.

example, by visualizing the average income across the countries, one might see there some **outliers**. Some countries have people who earn much more than anyone else. se outliers are worth investigating and are not necessarily incorrect data.

ating visualization can help to understand the data in greater detail and allow to spot erlying relationships that one would not be able to see by just describing the data. alizations can also reveal outliers that could potentially skew the results.

Review Questions with Answers

1. *Discuss common relationships that exist among 3 variables.* **(Refer section 5.1)**
2. *What do you understand by causal explanations ?* **(Refer section 5.2)**
3. *Explain the concept of longitudinal data.* **(Refer section 5.4)**
4. *List and explain characteristics of time series data.* **(Refer section 5.5)**
5. *Differentiate univariate and multivariate time series analysis ?* **(Refer section 5.5)**

5.7 Two Marks Questions and Answers

Q.1 Explain covariate independence of one variable in multivariate analysis.

Ans. :

- When a covariate Z is NOT related to X, it has a slightly different effect. It needs to be able to predict Y as well to be useful in the model, but the effects of X and Z do not overlap.

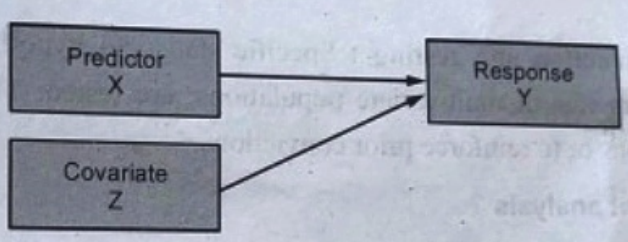


Fig. 5.7.1

- Including Z in the model often leads to the relationship between X and Y becoming more significant because Z has explained some of the otherwise unexplained variance in Y.
- An example of this kind of covariate is when an experimental manipulation (X) on response time (Y) only becomes significant when one can control for finger dexterity levels (Z).
- If finger dexterity has a large effect on response time and one do not account for it, all of that variation due to dexterity will go into unexplained error - The denominator of the test statistic.
- Controlling for that variance means it is no longer unexplained and is removed from the denominator.

- Test this type of effect by running hierarchical regression (add each predictor in a separate step).

Q.2 What are some of the central goals of multivariate analysis ?

Ans. :

- Below listed are central goals of multivariate analysis :

1) **Data reduction or structural simplification :** This helps data to get simplified as possible without sacrificing valuable information. This will make interpretation easier.
2) **Sorting and grouping :** When there are multiple variables, groups of "similar" objects or variables are created, based upon measured characteristics.
3) **Investigation of dependence among variables :** The nature of the relationships among variables is of interest. Are all the variables mutually independent or are one or more variables dependent on the others.
4) **Prediction relationships between variables :** Must be determined for the purpose of predicting the values of one or more variables based on observations on the other variables.
5) **Hypothesis construction and testing :** Specific statistical hypotheses, formulated in terms of the parameters of multivariate populations, are tested. This may be done to validate assumptions or to reinforce prior convictions.

Q.3 What is conjoint analysis ?

Ans. :

- **'Conjoint analysis'** is a survey-based statistical technique used in market research that helps determine how people value different attributes (feature, function, benefits) that make up an individual product or service. The objective of conjoint analysis is to determine the choices or decisions of the end-user, which drives the policy/product/service. Today it is used in many fields including marketing, product management, operations research, etc.
- It is used frequently in testing consumer response to new products, in acceptance of advertisements and in-service design. Conjoint analysis techniques may also be referred to as multi-attribute compositional modeling, discrete choice modeling or stated preference research and is part of a broader set of trade-off analysis tools used for systematic analysis of decisions.

- There are multiple conjoint techniques, few of them are CBC (Choice-based conjoint) or ACBC (Adaptive CBC).

Q.4 List applications of multivariate analysis.

Ans. : Multivariate analysis is used in several disciplines. One of its most distinguishing features is that it can be used in parametric as well as non-parametric tests.

1. **Parametric tests :** Tests which make certain assumptions regarding the distribution of data, that is, within a fixed parameter.
2. **Non-parametric tests :** Tests which do not make assumptions with respect to distribution. On the contrary, the distribution of data is assumed to be free of distribution.

Q.5 What is seasonality component in time series?

Ans. :

- The seasonal component consists of effects that are reasonably stable with respect to timing, direction and magnitude. It arises from systematic, calendar related influences such as :
  - **Natural conditions :** Weather fluctuations that are representative of the season (uncharacteristic weather patterns such as snow in summer would be considered irregular influences).
  - **Business and administrative procedures :** Start and end of the school term.
  - **Social and cultural behaviour :** Christmas.
- It also includes calendar related systematic effects that are not stable in their annual timing or are caused by variations in the calendar from year to year, such as :
  - **Trading day effects :** The number of occurrences of each of the day of the week in a given month will differ from year to year - There were 4 weekends in March in 2000, but 5 weekends in March of 2002.
  - **Moving holiday effects :** Holidays which occur each year, but whose exact timing shifts - Easter, Chinese New Year.

❑❑❑

SOLVED MODEL QUESTION PAPER

[As Per New Syllabus]

Data Exploration and Visualization

Semester - III (AI&DS)

Time : Three Hours] **[Maximum Marks : 100**

Answers ALL Questions

PART - A (10 × 2 = 20 Marks)

Q.1 *Explain multivariate graphical methods for data analysis.* **(Refer Two Marks Q.3 of Chapter - 1)**

Q.2 *Write a Python program to demonstrate use merge() function.* **(Refer Two Marks Q.5 of Chapter - 1)**

Q.3 *What is pyplot ?* **(Refer Two Marks Q.1 of Chapter - 2)**

Q.4 *What are contour plots ?* **(Refer Two Marks Q.5 of Chapter - 2)**

Q.5 *Brief about range scaling.* **(Refer Two Marks Q.3 of Chapter - 3)**

Q.6 *Classify time series.* **(Refer Two Marks Q.5 of Chapter - 3)**

Q.7 *What are possible types of correlation between two variables in bivariate analysis.* **(Refer Two Marks Q.1 of Chapter - 4)**

Q.8 *What is cross tabulation ?* **(Refer Two Marks Q.4 of Chapter - 4)**

Q.9 *What are some of the central goals of multivariate analysis ?* **(Refer Two Marks Q.2 of Chapter - 5)**

Q.10 *What is seasonality component in time series ?* **(Refer Two Marks Q.5 of Chapter - 5)**

PART - B (5 × 13 = 65 Marks)

Q.11 a) *What is EDA ? What is significance of EDA ?* **(Refer section 1.1)** **[7]**

b) *Discuss various visual aids for EDA.* **(Refer section 1.8)** **[6]**

OR

a) *Explain various data collection methods.* **(Refer section 1.1)** **[7]**

b) *What do you mean by making sense of data ?* **(Refer section 1.5)** **[6]**

Q.12 a) *What are applications of Matplotlib ?* **(Refer section 2.1)** **[6]**

b) *Discuss various simple plots for univariate data analysis.* **(Refer section 2.2)** [7]

OR

a) *How to visualize errors ?* **(Refer section 2.3)** [7]

b) *What are various customized markers in scatter plots ?* **(Refer section 2.5)** [6]

Q.13 a) *Discuss with example how univariate analysis is helpful ?* **(Refer section 3.1)** [7]

b) *What are univariate tables ?* **(Refer section 3.1)** [6]

OR

a) *Explain numerical summaries level and spread used in univariate analysis.* **(Refer section 3.3)** [6]

b) *What is feature scaling ?* **(Refer section 3.4)** [7]

Q.14 a) *Explain categorial variables and plotting with categorial variables.* **(Refer section 2.2)** [7]

b) *How time series are useful in data analysis ?* **(Refer section 3.5)** [6]

OR

a) *What are common ways to perform bivariate analysis ?* **(Refer section 4.1)** [7]

b) *What do you understand by causal explanations ?* **(Refer section 5.2)** [6]

Q.15 a) *Explain various types of bivariate analysis.* **(Refer section 4.1)** [6]

b) *Discuss with example percentage tables.* **(Refer section 4.2)** [7]

OR

a) *What are the uses of contigency tables ?* **(Refer section 4.3)** [7]

b) *How resistent lines are used in bivariate analysis ?* **(Refer section 4.5)** [6]

PART - C (1 × 15 = 15 Marks)

Q.16 a) *Discuss common relationships that exist among 3 variables.* **(Refer section 5.1)** [10]

b) *Explain the concept of longitudinal data.* **(Refer section 5.4)** [5]

OR

a) *List and explain characteristics of time series data.* **(Refer section 5.5)** [10]

b) *Differentiate univariate and multivariate time series analysis ?* **(Refer section 5.5)** [5]

❑❑❑