Geethanjali College of Engineering and Technology (Autonomous)

Cheeryal (V), Keesara (M), Medchal District, Telangana State– 501 301

BIG DATA ANALYTICS

(20CS41001**)**

COURSE FILE

IV Year B.Tech. CSE– I Semester

A.Y:2024 - 2025

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

(2024-2025)

Course Coordinator
Mrs. P. Usha Sree

HOD-CSE
Dr. A. Sree Lakshmi

Contents

Geethanjali College of Engineering and Technology
(Autonomous)

Cheeryal (V), Keesara (M), Medchal District, Telangana State– 501 301

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Name of the Course: **BIG DATA ANALYTICS**

Subject code: 20CS41001 Programme: **UG**

Branch: CSE	Version No: 1
Year: IV	Document Number: GCET/CSE/BDA/01
Semester: I	No. of Pages:143
Section: IV- CSE A/B/C/D/E	

Classification status (Unrestricted/Restricted): **Restricted**

Distribution List: **Department, Library**

Prepared by:	Updated by:
1) Name: K.Padmaja	1) Name:
2) Sign:	2) Sign:
3) Designation: Sr.Asst.Prof	3) Designation:
4) Date: 12.08.2024	4) Date:

Verified by:	*For Q.C only
1) Name:	1) Name:
2) Sign:	2) Sign:
3) Designation:	3) Designation:
4) Date:	4) Date:

Approved by (HOD):

1) Name: **Dr.A. SreeLaksmi**

2) Sign:

3) Date:

2. Vision of the Institute

Geethanjali visualizes dissemination of knowledge and skills to students, who would eventually contribute to wellbeing of the people of the nation and global community.

3. Mission of the Institute

1. To impart adequate fundamental knowledge in all basic sciences and engineering, technical and Inter-personal skills to students.
2. To bring out creativity in students that would promote innovation, research, and entrepreneurship.
3. To Preserve and promote cultural heritage, humanistic and spiritual values promoting peace and harmony in society.

4. Vision of the Department

To produce globally competent and socially responsible computer science engineers contributing to the advancement of engineering and technology which involves creativity and innovation by providing excellent learning environment with world class facilities.

5. Mission of the Department

1. To be a centre of excellence in instruction, innovation in research and scholarship, and service to the stake holders, the profession, and the public.
2. To prepare graduates to enter a rapidly changing field as a competent computer science engineer.
3. To prepare graduate capable in all phases of software development, possess a firm understanding of hardware technologies, have the strong mathematical background necessary for scientific computing, and be sufficiently well versed in general theory to allow growth within the discipline as it advances.
4. To prepare graduates to assume leadership roles by possessing good communication skills, the ability to work effectively as team members, and an appreciation for their social and ethical responsibility in a global setting.

6. Program Educational Objectives (PEOs)

1. To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in industry and / or to pursue postgraduate studies with an appreciation for lifelong learning.
2. To provide graduates with analytical and problem-solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.
3. To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting-edge technologies of multi-disciplinary nature for societal development.

Program Outcomes (CSE)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering

solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and teamwork**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PSO (Program Specific Outcome):

PSO 1: Demonstrate competency in Programming and problem-solving skills and apply these skills in solving real world problems.

PSO 2: Select appropriate programming languages, Data structures and algorithms in combination with modern technologies and tools, apply them in developing creative and innovative solutions.

PSO 3: Demonstrate adequate knowledge in concepts and techniques of CSE ,apply them in developing intelligent systems to solve real world problems.

7. Syllabus

GEETHANJALI COLLEGE OF ENGINEERING AND TECHNOLOGY
(UGC Autonomous)
Cheeryal (V), Keesara (M), Medchal Dist., Telangana-501301

20CS41001-BIG DATA ANALYTICS

B.Tech. CSE - IV Year, I Sem.

L	T	P/D	C
3	-	-	3

Prerequisite(s):
20CS21002-OBJECT ORIENTED PROGRAMMING
20CS21003-DATABASE MANAGEMENT SYSTEMS

Course Objectives
Develop ability to
1. Know the basic elements of Big Data and Data science to handle huge amount of data.
2. Gain knowledge of basic mathematics behind the Big data.
3. Understand the different Big data processing technologies.
4. Apply the Analytical concepts of Big data using R and Python.
5. Visualize the Big Data using different tools.

Course Outcomes (COs)
At the end of the course, student would be able to
CO1. Describe Big Data elements and Architectures.
CO2. Apply different mathematical models for Big Data.
CO3. Explain all the data processing components of Hadoop architecture by developing different applications.
CO4. Analyze data using R.
CO5. Employ appropriate tools for big data visualization.

UNIT-I
Getting an Overview of Big Data: What is Big Data?, History of Data Management – Evolution of Big Data, Structuring Big Data, Elements of Big Data, Big Data Analytics, Careers in Big Data, Future of Big Data.

UNIT-II
Introducing Technologies for Handling Big Data: Distributed and Parallel Computing for Big Data, Introducing Hadoop, Cloud Computing andBig Data, In-Memory Computing Technology for Big Data, Understanding Hadoop Ecosystem.

UNIT-III
Big Data processing: Big Data technologies, Introduction to Google file system, Hadoop Architecture, Hadoop Storage: HDFS, Common Hadoop Shell commands, NameNode, Secondary NameNode, and DataNode, Hadoop MapReduce paradigm, Map Reduce tasks, Job, Task trackers, Introduction to NOSQL, Textual ETL processing.

UNIT-IV
Big Data analytics: Data analytics life cycle, Data cleaning , Data transformation, Comparing reporting and analysis, Types of analysis, Analytical approaches, Data analytics using R, Exploring basic features of R, Exploring R GUI, Reading data sets, Manipulating and processing data in R, Functions and packages in R, Performing graphical analysis.

UNIT – V

Big Data Visualization: Introduction to Data visualization, Challenges to Big data visualization, Types of data visualization, Visualizing Big Data, Tools used in data visualization, Proprietary Data Visualization tools, Open source data visualization tools, Data visualization with Tableau.

TEXT BOOK(S)

1. Big Data, Black Book, DT Editorial Services, ISBN: 9789351197577, 2016 Edition.

REFERENCES BOOK(S)

1. Algorithmic and Analysis Techniques in Property Testing, Dana Ron, School of EE, 2010.
2. Synopses for Massive Data:Samples, Histograms, Wavelets, Sketches, Foundation and trends in databases, Graham Cormode, Minos Garofalakis, Peter J. Haas and Chris Jermaine, 2012.
3. R for Business Analytics, A.Ohri, Springer, ISBN:978-1-4614-4343-8, 2016.
4. Hadoop in practice, Alex Holmes, Dreamtech press, ISBN:9781617292224, 2015.

8. Course Objectives and Outcomes

Course Objectives:

Develop ability to

1. Know the basic elements of Big Data and Data science to handle huge amounts of data.
2. Gain knowledge about various technologies for handling Big Data.
3. Understand the different big data processing technologies.
4. Apply the Analytical concepts of big data using R and Python.
5. Visualize Big Data using different tools.

Course Outcomes (COs):

At the end of the course, student would be able to:

CO1: Describe Big Data elements and Architectures.

CO2: Understand various technologies for handling Big Data.

CO3: Explain all the data processing components of Hadoop architecture by developing different applications.

CO4: Analyze data using R.

CO5: Employ appropriate tools for big data visualization.

9. Brief notes on the importance of the course and how it fits into the curriculum

a. What role does this course play within the Program?

Data is the new oil in this era. The Big Data technologies and initiatives are rising to analyse this data for gaining insights that can help in making strategic decisions.

b. How is the course unique or different from other courses of the Program?

Big Data analytics is **more specific and concentrated** it focuses various technologies such as Hadoop map reduce programming paradigm, R analytical tool and its importance and finally concentrates on modern tool usage such as tableau.

c. What essential knowledge or skills should they gain from this experience?

Working on Hadoop using map reduce programming.

Critical Thinking.

R Statistical Programming. ...

Data Visualization.

Presentation

Skills. Machine

Learning.

d. What knowledge or skills from this course will students need to have mastered to perform well in future classes or later (Higher Education / Jobs)?

Data Visualization.

Data Cleaning.

R.

Python.

SQL and NoSQL.

Machine Learning.

Linear Algebra and Calculus

e. Why is this course important for students to take?

Big data analytics describes the process of uncovering trends, patterns, and correlations in large amounts of raw data to help make data-informed decisions. These processes use familiar statistical analysis techniques—like clustering and regression—and apply them to more

extensive datasets with the help of newer tools.

f. When students complete this course, what do they need know or be able to do?

Big data analytics is an emerging field that has many opportunities that could impact industries such as finance, government, and healthcare. Data analysts can help change lives by providing valuable insights and real time solutions .This course helps students build vital expertise in Various technologies in Big data and its working, an exposure towards analytical and visualization tools.

g. Is there specific knowledge that the students will need to know in the future?

Analytical

Thinking SQL

Database Decision

Analysis

Mathematical and Statistical

Skills Software Analytics

Programming Skills

Functions and

Formulas

Data Cleaning and

Preparation Quantitative

Skills

Data Visualization

Skills Query Languages

Problem Solving

Domain Knowledge

h. Five years from now, what do you hope students will remember from this course?

Graduates have an opportunity to take specialization in the specified course and an opt for business analyst or can be involved in industry for a data scientist role.

i. What is it about this course that makes it unique or special?

Data analytics is important because it helps businesses optimize their performances. A company can also use data analytics to make better business decisions and help analyze customer trends and satisfaction, which can lead to new—and better—products and services.

j. Why does the program offer this course?

It helps us to understand importance of data and how it can be turned into business revenue if we can handle, process, and analyse in an appropriate way.

l. Why can't this course be "covered" as a sub-section of another course?

Course itself is a specialization in masters.

m. What unique contributions to students' learning experience does this course make?

It helps students to understand the value of data and how the data has become the new oil in the industry.

n. What is the value of taking this course? How exactly does it enrich the program?

Big Data analytics is important **because it helps businesses optimize their performances**. Implementing it into the business model means companies can help reduce costs by identifying more efficient ways of doing business and by storing large amounts of data.

p. What are the major career options that require this course?

Machine learning

engineer. Data architect.

Statistician.

Data analyst.

Chief technology officer (CTO) .

Chief data officer (CDO) .

Application

architect. Project

manager.

10. Prerequisites if any

- 18CS2102 - Object Oriented Programming using Java.
- 18CS2203 - Database Management Systems

.

11. Instructional learning outcomes

S.No	Unit	Contents	Outcomes
1	I	Introduction: Data Science and Big Data	Observe Big Data elements and Architectures.
2	II	Introducing Technologies for Handling Big Data	Learn different big data handling technologies
3	III	Big Data processing	Demonstrate their Big Data skills by developing different applications.
4	IV	Big Data analytics	Apply each learning model for different datasets.
5	V	Big Data Visualization	Analyse needs, challenges, and techniques for big data visualization.

12. Course Mapping with POs

Course	PEOs	POs
BIG DATA ANALYTICS	**PEO1, PEO2, PEO3**	**PO1, PO2, PO3, PO4, PO5, PSO1, PSO2, PSO3**

Mapping of Course outcomes to Program Outcomes:

Course Name BIG DATA ANALYTICS	Program Outcomes														
	1	2	3	4	5	6	7	8	9	10	11	12	PSO 1	PSO 2	PSO 3
CO1: Describe Big Data elements and Architectures.	2	1	1	1									1		1
CO2: Understand various technologies for handling Big Data.	2	1		1									1		1
CO3: Explain all the data processing components of Hadoop architecture by developing different applications.	1	1	2	1	1								2	2	2
CO4: Analyze data using R.	1	2	1	2	2								2	2	2
CO5: Employ appropriate tools for big data visualization.	1	2		2	2								1	1	2

Timetable

Year/Sem/Sec: IV-B.Tech I-Semester B-Section					A.Y : 2024 -25	W.E.F. 12-08-2024		Version : 01
Class Teacher: M.Supriya							Room No :	
Time	**09.00-09.50**	**09.50-10.40**	**10.40-11.30**	**11.30-12.20**	**12.20-1.00**	**01.00-01.50**	**01.50-02.40**	**02.40-03.30**
Period	**1**	**2**	**3**	**4**		**5**	**6**	**7**
Monday					**L U N C H**			
Tuesday				**BDA**				
Wednesday		**BDA**						
Thursday	**BDA**					**<--------BDA LAB -------□**		
Friday								
Saturday								

Individual Time Table

Year/Sem/Sec: IV-B.Tech I-Semester A-Section					A.Y : 2023 -24	W.E.F. 18-07-2023		Version : 01
Class Teacher: Mrs.M.Supriya							Room No :N 2 1 7	
Time	**09.00-09.50**	**09.50-10.40**	**10.40-11.30**	**11.30-12.20**	**12.20-1.00**	**01.00-01.50**	**01.50-02.40**	**02.40-03.30**
Period	**1**	**2**	**3**	**4**		**5**	**6**	**7**
Monday					LUNCH			
Tuesday				BDA				
Wednesday		BDA						
Thursday	BDA					BDA LAB		
Friday								
Saturday								
Monday					LUNCH			
Tuesday	B							
Wednesday								
Thursday								
Friday								
Saturday								

13. Lecture plan with methodology being used/adopted

Lesson Schedule

S.No	Date	No of periods	Topics to be covered	Regular / Additional	Teaching aids used. LCD/OHP/BB
Unit - I					
1.	Day 1	1	Course objectives and Course Outcomes, Introduction to Big Data	Regular	LCD
2.	Day 2	1	What is Big Data, History of Data Management	Regular	LCD
3.	Day 3	1	Evolution of Big Data	Regular	LCD
4.	Day 4,5	2	Structuring Big Data	Regular	LCD
5.	Day 6	1	Elements of Big Data	Regular	LCD
6.	Day 7	1	Big Data Analytics, Careers in Big Data	Regular	LCD
7.	Day 8	1	Future of Big Data		
UNIT-II					
1.	Day 9	1	Introducing Technologies for Handling Big Data	Regular	LCD
2.	Day 10	1	Distributed and Parallel Computing for Big Data	Regular	LCD
3.	Day 11	1	Introducing Hadoop	Regular	LCD
4.	Day 12	1	Cloud Computing and Big Data	Regular	LCD
5.	Day 13	1	In-Memory Computing Technology for Big Data	Regular	LCD
6.	Day 14	1	Understanding Hadoop Ecosystem	Regular	LCD
UNIT-III					
1.	Day 15	1	Big Data processing: Big Data technologies	Regular	LCD
2.	Day 16	1	Introduction to Google file system	Regular	LCD
3.	Day 17,18	2	Hadoop Architecture	Regular	LCD
4.	Day 19,20	2	Hadoop Storage: HDFS, Common Hadoop Shell commands,	Regular	LCD

5.	Day 21,22	2	Name Node, Secondary Name Node, Data Node	Regular	LCD
6.	Day 23	1	Hadoop MapReduce paradigm	Regular	LCD
7.	Day 24	1	Map Reduce tasks, jobs	Regular	LCD
8.	Day 25	1	Job tracker & Task trackers	Regular	LCD
9.	Day 26,27	2	Introduction to NOSQL	Regular	LCD
10.	Day 28	1	Textual ETL processing	Regular	LCD
UNIT-IV					
1.	Day 29.30	2	Data analytics life cycle, Data cleaning, Data transformation	Regular	LCD
2.	Day 30,31	2	Comparing reporting and analysis, Types of analysis, Analytical approaches	Regular	LCD
3.	Day 32,33	2	Data analytics using R, exploring basic features of R, Exploring R GUI		
4.	Day 34,35	2	Reading data sets, Manipulating, and processing data in R		
5.	Day 36,37	2	Functions and packages in R, Performing graphical analysis.	Regular	LCD
UNIT-V					
1.	Day 38,39	2	Introduction to Data visualization, Challenges to Big data visualization	Regular	LCD
2.	Day 40,41,42	3	Types of data visualization, Visualizing Big Data, Tools used in data visualization	Regular	LCD
3.	Day 43,44,45	3	Proprietary Data Visualization tools, Open-source data visualization tools	Regular	LCD
4.	Day 46,47,48	3	Data visualization with Tableau.	Regular	LCD
5.	Day 49	1	Introduction to Spark	Additional	LCD

14. Assignment Questions

Unit 1

1. Flipkart announces its big billion-day sale for this festival week. What are the possible challenges that could be faced by the retailer and how it could be handled?
2. Data is growing at an exponential acceleration since the advent of computers and internet, categorize various kinds of data contributing towards its growth with an example?
3. Elucidate various characteristics of big data with an example.
4. A bookstore owner wants his business to be spread out at 5 locations in India and 5 locations abroad. He wants all information to be available online. Answer the Following:
   i. What are the top challenges he would face initially?
   ii. Is there any way of putting all details on web portal? Justify.

   What are the three top reasons he should consider leveraging big data.
5. Illustrate with relevant examples and diagrams wherever applicable to discuss the classification of analytics.

Unit 2

1. Identify the difference between parallel processing system and distributed system. Discuss how they help to store and process big data.
2. List and Discuss various features of cloud Computing.
3. What assumptions were made in the design of HDFS? Do these assumptions make sense? Discuss why?
4. Explicate various Cloud Deployment models. 5.Explain Hadoop ecosystem.

Unit 3

1. What are the two main phases of the Map Reduce programming? Explain through an example
2. Illustrate the functions of Map Reduce component of Hadoop with explanation to daemons
3. How is a file stored in HDFS to support parallel processing as well as support fault tolerance? 4.What is the role of name node, secondary name node and data node in Hadoop. 5.What are the tasks perform by map reduce in Hadoop?

Unit 4 & 5 planning for case study.

15. Tutorial problems : NA

16. Question Bank

UNIT-1

1. List and discuss the four elements of big data.
2. As an HR manager of a company providing Big Data solutions clients, What characteristics would you look for while recruiting a potential candidate for the position of data analyst.
3. You are planning the market strategy for a new product in your company. Identify and list some limitations of structured data related to this work.
4. Discuss in details different types of analytics.

UNIT-2

5. What is distributed computing? Explain the working of a distributed computing environment. 6.List the difference between parallel and distributed systems.
7. Discuss the techniques of parallel computing. 8.List the important features of Hadoop.
9. Discuss the features of cloud computing that can be used to handle big data.
10. Discuss the role cloud services play in handling big data.
11. Write a short note on In-Memory Computing technology.
12. Write a short note on the Hadoop ecosystem.
13. How does HDFS ensure data integrity in a Hadoop cluster.

UNIT-3

14. Discuss the working of MapReduce.
15. What is role of Name Node in an HDFS cluster. 16.List the main features of MapReduce.
17. Descirbe the working of the MapReduce algorithm.
18. Discuss some techniques to optimize MapReduce Jobs.
19. Discuss the point you need to consider while designing a file system in MapReduce. 20.Write a short note on Input Split.
21. Explain the types of MapReduce applications.
22. Write a short note on FileInputFormat class.
23. Compare relational database with NoSQL database.

24.What is the purpose of sharding? What is difference between replication and sharding?

25.Explain the ways in which data can be distributed.

UNIT-4

26. Explain reporting 27.Explain the following

a. Basic Analytics

b. Advanced Analytics

c. Operational Analytics

d. Monetized Analytics.

28. Explain the working of the c () command in R.

29.Explain the working of the scan () command in R.

30.Compare the write. Table () and write.csv () commands.

31.List some types of data structures available in R.

32.Name some operators used to form data subsets in R.

33. List the functions provided by R to combine different sets of data.

34.List some common characteristics of the R language.

35.Write a script to generate the following list of numerical values with spaces.

36.Write a script to concatenate text and numerical values in R

37. Define text data analysis.

38.What are analytics point solutions.

39.Explain analytical process

40.What are the two main advantages of using functions over scripts?

41.Discuss the syntax of defining a function in R.

42.What do you understand by plotting in R.

43.List the techniques used to draw graphs in R. 44.Write a short note on time series plots.

UNIT-5

45.Discuss some applications of data visualization. 46.Name a few data visualization tools.

47.List and discuss various types of data visualizations. 48.Write a note on tableau products.

49. List and discuss the icons presents on the tableau toolbar.

50. What is Tableau server.

17. Quiz questions

17. Detailed Notes

UNIT I

Getting an Overview of Big Data: What is Big Data?, History of Data Management – Evolution of Big Data, Structuring Big Data, Elements of Big Data, Big Data Analytics, Careers in Big Data, Future of Big Data.

INTRODUCTION TO BIG DATA

The biggest phenomenon that has captured the attention of the modern computing industry today since the "Internet" is "Big Data". These two words combined was first popularized in the paper on this subject by McKinsey & Co., and the foundation definition was first popularized by Doug Laney from Gartner. The fundamental reason why "Big Data" is popular today is because the technology platforms that have emerged along with it, provide the capability to process data of multiple formats and structures without worrying about the constraints associated with traditional systems and database platforms. Big Data represents the lowest raw format of information or knowledge.

The argument presented in the decision-making behaviours and administrative behaviours makes complete sense, as we limit the data in the process of modelling, applying algorithmic applications, and have always been seeking discrete relationships within the data as opposed to the whole picture. However, decision making has always transcended beyond the traditional systems used to aid the process. For example, patient treatment and management is not confined to computers and programs.

These insights provide visible patterns that can be useful in improving quality of care for a given set of diseases. Data warehousing evolved to support the decision-making process of being able to collect, store, and manage data, applying traditional and statistical methods of measurement to create a reporting and analysis platform. The data collected within a data warehouse was highly structured in nature, with minimal flexibility to change with the needs of data evolution. The underlying premise for this comes from the transactional databases that were the sources of data for a data warehouse. This concept applies very well when we talk of transactional models based on activity generated by consumers in retail, financial, or other industries. For example, movie ticket sales is a simple transaction, and the success of a movie is based on revenues it can generate in the opening and following weeks, and in a later stage followed by sales from audio (vinyl to cassette

tapes, CDs', and various digital formats), video ('DVDs and other digital formats), and merchandise across multiple channels.

When reporting sales revenue, population demographics, sentiments, reviews, and feedback were not often reported or at least were not considered as a visible part of decision making in a traditional computing environment. The reasons for this included rigidity of traditional computing architectures and associated models to integrate unstructured, semi-structured, or other forms of data, while these artifacts were used in analysis and internal organizational reporting for revenue activities from a movie. Looking at these examples in medicine and entertainment business management, we realize that decision support has always been an aid to the decision-making process and not the end state itself, as is often confused.

If one were to consider all the data, the associated processes, and the metrics used in any decision making situation within any organization, we realize that we have used information (volumes of data) in a variety of formats and varying degrees of complexity and derived decisions with the data in non-traditional software processes. Before we get to Big Data, let us look at a few important events in computing history. In the late 1980s, we were introduced to the concept of decision support and data warehousing. This wave of being able to create trends, perform historical analysis, and provide predictive analytics and highly scalable metrics created a series of solutions, companies, and an industry in itself.

All these entities have contributed to the consumerization of data, from data creation, acquisition, and consumption perspectives. The business models and opportunities that came with the large-scale growth of data drove the need to create powerful metrics to tap from the knowledge of the crowd that was driving them, and in return offer personalized services to address the need of the moment.

Here are some examples:

- Weather data—there is a lot of weather data reported by governmental agencies around the world, scientific organizations, and consumers like farmers. What we hear on television or radio is an analytic key performance indicator (KPI) of temperature and forecasted conditions based on several factors.
- Contract data—there are many types of contracts that an organization executes every year, and there are multiple liabilities associated with each of them.
- Labor data—elastic labor brings a set of problems that organizations need to solve.

- Maintenance data—records from maintenance of facilities, machines, non-computer-related systems, and more.
- Financial reporting data—corporate performance reports and annual filing to Wall Street.
- Compliance data—financial, healthcare, life sciences, hospitals, and many other agencies that file compliance data for their corporations.
- Clinical trials data—pharmaceutical companies have wanted to minimize the life cycle of processing for clinical trials data and manage the same with rules-based processing; this is an opportunity for Big Data.
- Processing doctors' notes on diagnosis and treatments—another key area of hidden insights and value for disease state management and proactive diagnosis; a key machine learning opportunity.
- Contracts—every organization writes many types of contracts every year, and must process and mine the content in the contracts along with metrics to measure the risks and penalties.

Define Big Data

The term "big data" refers to data that is so large, fast or complex that it's difficult or impossible to process using traditional methods. The act of accessing and storing large amounts of information for analytics has been around a long time.

The History of Big Data

Although the concept of big data itself is relatively new, the origins of large data sets go back to the 1960s and '70s when the world of data was just getting started with the first data centers and the development of the relational database.

Around 2005, people began to realize just how much data users generated through Facebook, YouTube, and other online services. Hadoop (an open-source framework created specifically to store and analyze big data sets) was developed that same year. NoSQL also began to gain popularity during this time.

The development of open-source frameworks, such as Hadoop (and more recently, Spark) was essential for the growth of big data because they make big data easier to work with and cheaper to store. In the years since then, the volume of big data has skyrocketed. Users are still generating huge amounts of data—but it's not just humans who are doing it.

With the advent of the Internet of Things (IoT), more objects and devices are connected to the internet, gathering data on customer usage patterns and product performance. The emergence of machine learning has produced still more data.

While big data has come far, its usefulness is only just beginning. Cloud computing has expanded

big data possibilities even further. The cloud offers truly elastic scalability, where developers can simply spin up ad hoc clusters to test a subset of data.

Types of Big Data

Now that we are on track with what is big data, let's have a look at the types of big data:

a) Structured

Structured is one of the types of big data and By structured data, we mean data that can be processed, stored, and retrieved in a fixed format. It refers to highly organized information that can be readily and seamlessly stored and accessed from a database by simple search engine algorithms. For instance, the employee table in a company database will be structured as the employee details, their job positions, their salaries, etc., will be present in an organized manner.

b) Unstructured

Unstructured data refers to the data that lacks any specific form or structure whatsoever. This makes it very difficult and time-consuming to process and analyze unstructured data. Email is an example of unstructured data. Structured and unstructured are two important types of big data.

c) Semi-structured

Semi structured is the third type of big data. Semi-structured data pertains to the data containing both the formats mentioned above, that is, structured and unstructured data. To be precise, it refers to the data that although has not been classified under a particular repository (database), yet contains vital information or tags that segregate individual elements within the data. Thus we come to the end of types of data.

Characteristics of Big Data

a) Variety: Variety of Big Data refers to structured, unstructured, and semi-structured data that is gathered from multiple sources. While in the past, data could only be collected from spreadsheets and databases, today data comes in an array of forms such as emails, PDFs, photos, videos, audios, SM posts, and so much more. Variety is one of the important characteristics of big data.

b) Velocity: Velocity essentially refers to the speed at which data is being created in real-time. In a broader prospect, it comprises the rate of change, linking of incoming data sets at varying speeds, and activity bursts.

c) Volume: Volume is one of the characteristics of big data. We already know that Big Data indicates huge 'volumes' of data that is being generated daily from various sources like social

media platforms, business processes, machines, networks, human interactions, etc. Such a large amount of data is stored in data warehouses. Thus, comes to the end of characteristics of big data.

Classification of analytics:

Descriptive analytics: Descriptive analytics is a statistical method that is used to search and summarize historical data to identify patterns or meaning.

Predictive analytics: Predictive Analytics is a statistical method that utilizes algorithms and machine learning to identify trends in data and predict future behaviours.

Prescriptive analytics: Prescriptive analytics is a statistical method used to generate recommendations and make decisions based on the computational findings of algorithmic models.

Advantages of Big Data

1. Cost optimization

One of the most major advantages of Big Data technologies is that they reduce the cost of storing, processing, and analysing enormous volumes of data for enterprises. Not only that, but Big Data technologies may help find cost-effective and efficient company practices. The logistics business serves as a good illustration of Big Data's cost-cutting potential. In most cases, the cost of goods returned is 1.5 times the cost of delivery.

2. Helps in Understanding the Market Conditions

By examining Big data, it is possible to have a better knowledge of current market situations. Let's take an example: a corporation can determine the most popular goods by studying a customer's purchase behaviour. It aids in the analysis of trends and client desires. A company can use this to get an advantage over its competition.

3. Improve efficiency

Big Data techniques can dramatically enhance operational efficiency. Big Data technologies may collect vast volumes of usable customer data by connecting with customers/clients and getting their important input.

4. Better Decision Making

The main benefit of using Big Data Analytics is that it has boosted the decision-making process to a great extent. Rather than anonymously making decisions, companies are considering Big Data Analytics before concluding any decision.

5. **Improving Customer Service and Customer Experience**

Big data, machine learning (ML), and artificial intelligence (AI)-powered technical support and helpline services may considerably increase the quality of response and follow-up that firms can provide to their customers.

6. **Fraud and Anomaly Detection**

It's just as vital to know what's going wrong in businesses like financial services or healthcare as it is to know what's going right. With big data, AI and machine learning algorithms can quickly discover erroneous transactions, fraudulent activity signs, and abnormalities in data sets that might indicate a variety of current or prospective problems.

7. **Focused And Targeted Campaigns**

Big data may be used by businesses to give customized products to their target market. Don't waste money on ineffective advertising strategies. Big data enables businesses to do in-depth analyses of customer behavior. Monitoring online purchases and watching point-of-sale transactions are common parts of this investigation.

8. **Innovative Products**

Big data continues to assist businesses in both updating existing goods and developing new ones. Companies can discern what matches their consumer base by gathering enormous volumes of data.

9. **Agile supply chain management**

Surprising, because we usually don't notice our supply networks until they've been severely disrupted. Big data, which includes predictive analytics and is typically done in near real-time, aids in keeping our worldwide network of demand, production, and distribution running smoothly.

10. **Improved operations**

Big data analytics may be used to enhance a variety of business activities, but one of the most exciting and gratifying has been using big data analytics to improve physical operations. For example, using big data and data science to create predictive maintenance plans might help important systems avoid costly repairs and downtime. Start by looking at the age, condition, location, warranty, and servicing information.

Big data analytics applications don’t restrict to one field only .Following are the application of big data analytics

1. Banking and Securities
2. Communications, Media and Entertainment

3. Healthcare Providers
4. Education
5. Manufacturing and Natural Resources
6. Government
7. Insurance
8. Retail and Wholesale trade
9. Transportation
10. Energy and Utilities
11. Big Data & Auto Driving Car
12. Big Data in IoT
13. Big Data in Marketing
14. Big Data in Business Insights

Career in big data analytics:

1. Soaring Demand for Analytics Professionals
2. Huge Job Opportunities & Meeting the Skill Gap
3. Salary Aspects
4. Big Data Analytics: A Top Priority in a lot of Organizations
5. Adoption of Big Data Analytics is Growing
6. Analytics: A Key Factor in Decision Making
7. The Rise of Unstructured and Semi structured Data Analytics
8. Big Data Analytics is Used Everywhere
9. Surpassing Market Forecast / Predictions for Big Data Analytics

Future of Big Data :

As the amount of big data continues to grow, so does the need for trained data analysts to dive in and extract insights. Big data analytics provides exciting opportunities for creating change in industries such as finance, government, and healthcare. Data analysts can help change lives by preventing fraud, allocating resources in a natural disaster, or improving the quality of healthcare.

Big Data examples

1. Fraud detection

For businesses whose operations involve any type of claims or transaction processing, fraud detection is one of the most compelling Big Data application examples. Historically, fraud

detection on the fly has proven an elusive goal. In most cases, fraud is discovered long after the fact, at which point the damage has been done and all that's left is to minimize the harm and adjust policies to prevent it from happening again. Big Data platforms that can analyze claims and transactions in real time, identifying large-scale patterns across many transactions or detecting anomalous behavior from an individual user, can change the fraud detection game.

2. IT log analytics

IT solutions and IT departments generate an enormous quantity of logs and trace data. In the absence of a Big Data solution, much of this data must go unexamined: organizations simply don't have the manpower or resource to churn through all that information by hand, let alone in real time. With a Big Data solution in place, however, those logs and trace data can be put to good use. Within this list of Big Data application examples, IT log analytics is the most broadly applicable. Any organization with a large IT department will benefit from the ability to quickly identify large- scale patterns to help in diagnosing and preventing problems. Similarly, any organization with a large IT department will appreciate the ability to identify incremental performance optimization opportunities.

3. Call center analytics

Now we turn to the customer-facing Big Data application examples, of which call center analytics are particularly powerful. What's going on in a customer's call center is often a great barometer and influencer of market sentiment, but without a Big Data solution, much of the insight that a call center can provide will be overlooked or discovered too late. Big Data solutions can help identify recurring problems or customer and staff behavior patterns on the fly not only by making sense of time/quality resolution metrics, but also by capturing and processing call content itself.

4. Social media analysis

A Big Data solution built to harvest and analyze social media activity, like IBM's Cognos Consumer Insights, a point solution running on IBM's BigInsights Big Data platform, can make sense of the chatter. Social media can provide real-time insights into how the market is responding to products and campaigns. With those insights, companies can adjust their pricing, promotion, and campaign placement on the fly for optimal results.

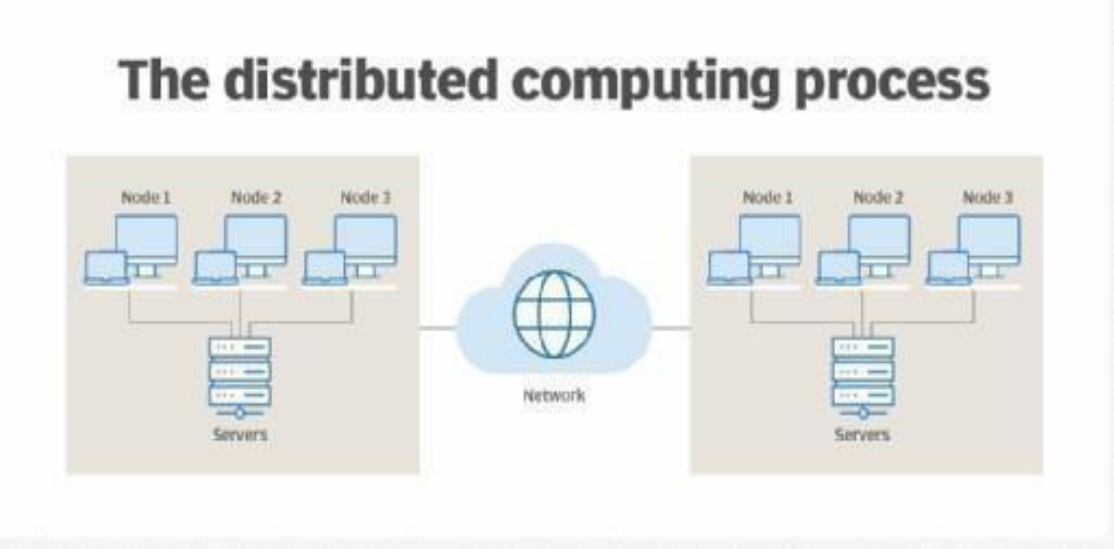
UNIT II

Introducing Technologies for Handling Big Data: Distributed and Parallel Computing for Big Data, Introducing Hadoop, Cloud Computing and Big Data, In-Memory Computing Technology for Big Data, Understanding Hadoop Ecosystem.

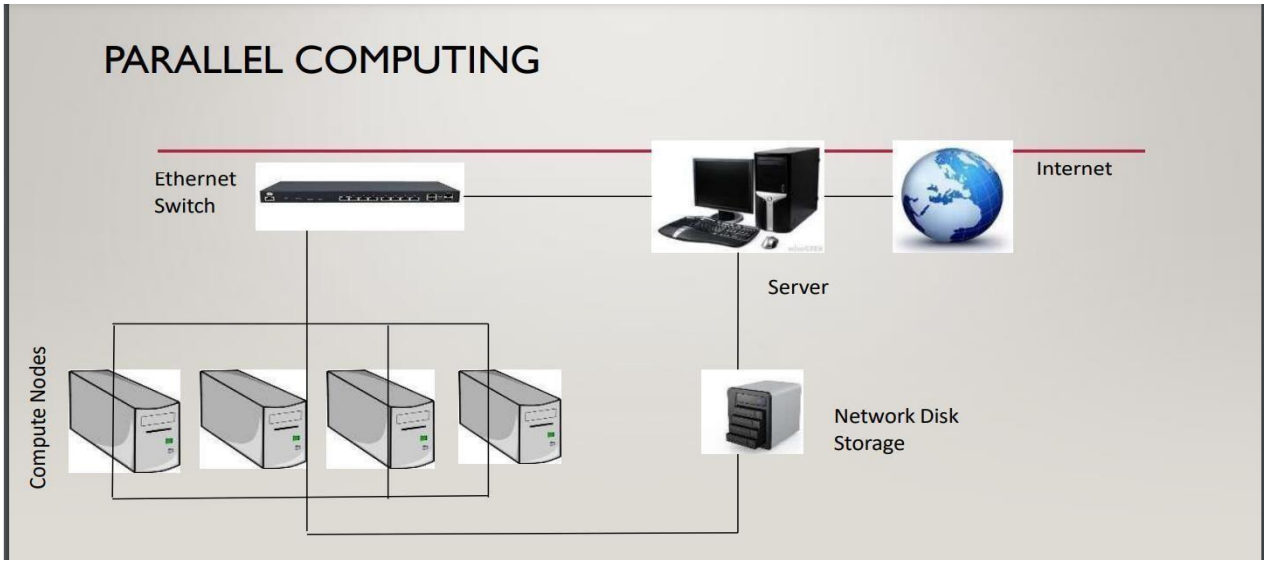
Distributed and parallel Computing for Big Data:

Among the technologies that are used to handle, process, and analyze big data the most popular and effective innovations have been in the field of distributed and parallel processing, Hadoop, in memory computations, big data cloud. Most popular one is Hadoop. Organizations use it to extract maximum output from normal data usage practices at a rapid pace. Cloud computing helps companies to save cost and better manage resources. Big Data can't be handled by traditional data storage and processing systems. For handling such type of data, Distributed and Parallel Technologies are more suitable. Multiple computing resources are connected in a network and computing tasks are distributed across these resources.

- Increases the Speed.
- Increases Efficiency.
- More suitable to process huge amounts of data in a limited time.



Parallel Computing: Also improves the processing capability of a computer system by adding additional computational resources to it. Divide complex computations into subtasks, handled individually by processing units, running in parallel. Concept – processing capability will increase with the increase in the level of parallelism.



BIG DATA PROCESSING TECHNIQUES:

With the increase in data, forcing organizations to adopt a data analysis strategy that can be used for analyzing the entire data in a very short time.

Done by Powerful h/w components and new s/w

programs. The procedure followed by the s/w applications

are:

1) Break up the given task

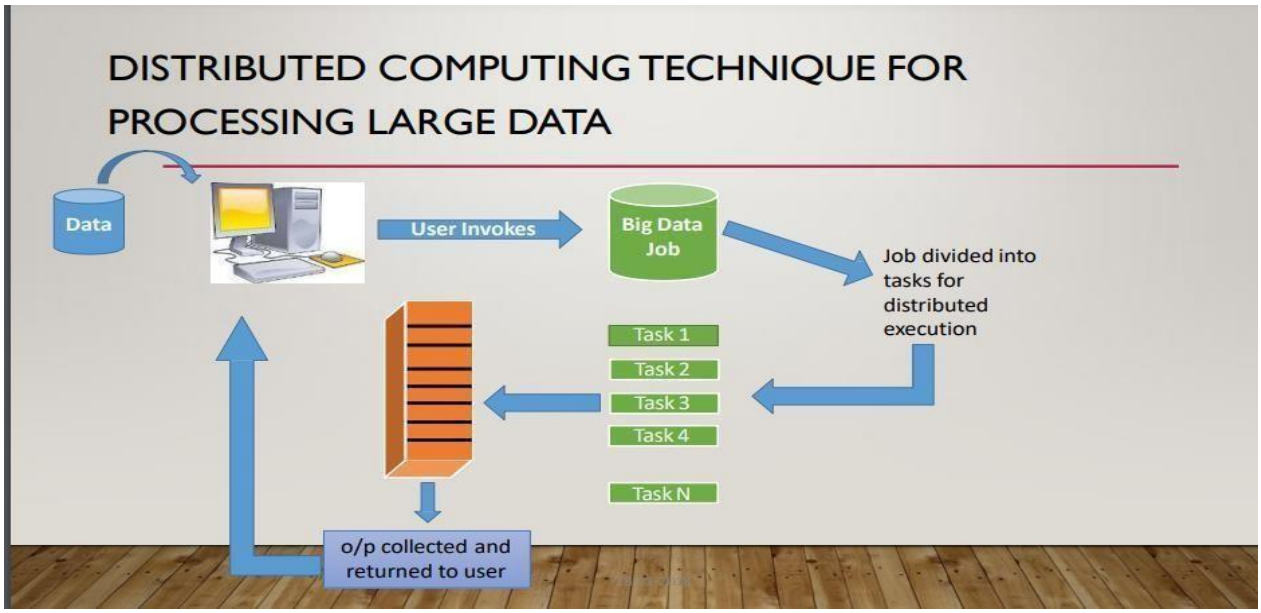
2) Surveying the available resources

3) Assigning the subtask to the nodes

Resources develop some technical problems and fail to respond to virtualization. Some processing and analytical tasks are delegated to other resources.

Latency: can be defined as the aggregate delay in the s/m because of delays in the completion of individual tasks. System delay: Also affects data management and communication.

Affecting the productivity & profitability of an organization.



PARALLEL COMPUTING TECHNIQUES:

Cluster or Grid Computing

• primarily used in Hadoop.

• based on a connection of multiple servers in a network (clusters)

• servers share the workload among them.

• overall cost may be very high. Massively Parallel Processing (MPP)

• used in data warehouses. Single machine working as a grid is used in the MPP platform.

• Capable of handling storage, memory and computing activities.

• Software written specifically for MPP platform is used for optimization.

• MPP platforms, EMC Greenplum, Paracel , suited for high-value use cases.

3) High Performance Computing (HPC) : Offer high performance and scalability by using IMC. Suitable for processing floating point data at high speeds.

Used in research and business organization where the result is more valuable than the cost or where strategic importance of project is of high priority.

HADOOP

Hadoop is a distributed system like distributed database.

Hadoop is a ‘software library’ that allows its users to process large datasets across distributed clusters of computers, thereby enabling them to gather, store and analyze huge sets of data.

It provides various tools and technologies, collectively termed as the Hadoop Ecosystem.

Hadoop cluster consist of single Master Node and a multiple Worker Nodes.

FEATURES OF CLOUD COMPUTING:

- Scalability – addition of new resources to an existing infrastructure. - increase in the amount of data, requires organization to improve h/w components. - The new h/w may not provide complete support to the s/w, that used to run properly on the earlier set of h/w. - Solution to this problem is using cloud services - that employ the distributed computing technique to provide scalability.
- Elasticity – Hiring certain resources, as and when required, and paying for those resources. - no extra payment is required for acquiring specific cloud services. - A cloud does not require customers to declare their resource requirements in advance.
- Resource Pooling - multiple organizations, which use similar kinds of resources to carry out computing practices, have no need to individually hire all the resources.
- Self Service – cloud computing involves a simple user interface that helps customers to directly access the cloud services they want.
- Low Cost – cloud offers customized solutions, especially to organizations that cannot afford too much initial investment. - cloud provides pay-as-you-use option, in which organizations need to sign for those resources only that are essential.
- Fault Tolerance – offering uninterrupted services to customers.

CLOUD DEPLOYMENT MODELS:

Depending upon the architecture used in forming the n/w, services and applications used, and the target consumers, cloud services form various deployment models. They are,

- Public Cloud
- Private Cloud
- Community Cloud
- Hybrid Cloud

• Public Cloud (End-User Level Cloud)

- Owned and managed by a company than the one using it.
- Third party administrator.
- Eg : Verizon, Amazon Web Services, and Rackspace.
- The workload is categorized because of service category, h/w customization is possible to provide optimized performance.

- The process of computing becomes very flexible and scalable through customized h/w resources.

- The primary concern with a public cloud include security and latency.

• Private Cloud (Enterprise Level Cloud)

- Remains entirely in the ownership of the organization using it.

- Infrastructure is solely designed for a single organization.

- Can automate several processes and operations that require manual handling in a public cloud.

- Can also provide firewall protection to the cloud, solving latency and security concerns.

- A private cloud can be either on-premises or hosted externally. on premises: service is exclusively used and hosted by a single organization. hosted externally: used by a single organization and are not shared with other organizations.

• Community Cloud

-Type of cloud that is shared among various organizations with a common tie.

-Managed by third party cloud services.

-Available on or off premises.

Eg. In any state, the community cloud can provided so that almost all govt. organizations of that state can share the resources available on the cloud. Because of the sharing of resources on community cloud, the

data of all citizens of that state can be easily managed by the govt. organizations.

• Hybrid Cloud

-various internal or external service providers offer services to many organizations.

-In hybrid clouds, an organization can use both types of cloud, i.e. public and private together – situations such as cloud bursting.

• Organization uses its own computing infrastructure, high load requirement, access clouds.

The organization using the hybrid cloud can manage an internal private cloud for general use and migrate the entire or part of an application to the public cloud during the peak periods.

CLOUD SERVICES FOR BIG DATA

In big data Iaas, Paas and Saas clouds are used in following manner.

• Iaas:- Huge storage and computational power requirement for big data are fulfilled by limitless storage space and computing ability obtained by iaas cloud.

• Paas:- offerings of various vendors have started adding various popular big data platforms that include MapReduce, Hadoop. These offerings save organizations from a lot of hassles which occur in managing individual hardware components and software applications.

- Saas:- Various organizations require identifying and analyzing the voice of customers particularly on social media. Social media data and platform are provided by SAAS vendors. In addition, private cloud facilitates access to enterprise data which enables these analyses.

IN MEMORY COMPUTING TECHNOLOGY

In-memory computing is a computing approach that stores data in a system's RAM instead of traditional disk storage, enabling faster data processing and real-time analysis. By utilizing the faster speed of RAM, in-memory computing significantly reduces processing time and latency. It allows data to be processed and analyzed directly in a computer's memory, rather than being stored on disk.

This is the way to improve speed and processing power of data.

- IMC is used to facilitate high speed data processing e.g. IMC can help in tracking and monitoring the consumers' activities and behaviors which allow organizations to take timely actions for improving customer services and hence customer satisfaction.
- Data stored on external devices known as secondary storage space. This data had to be accessed from an external source.
- In the IMC technology the RAM or Primary storage space is used for analyzing data. Ram helps to increase computing speed.
- Also, reduction in cost of primary memory has helped to store data in primary memory.

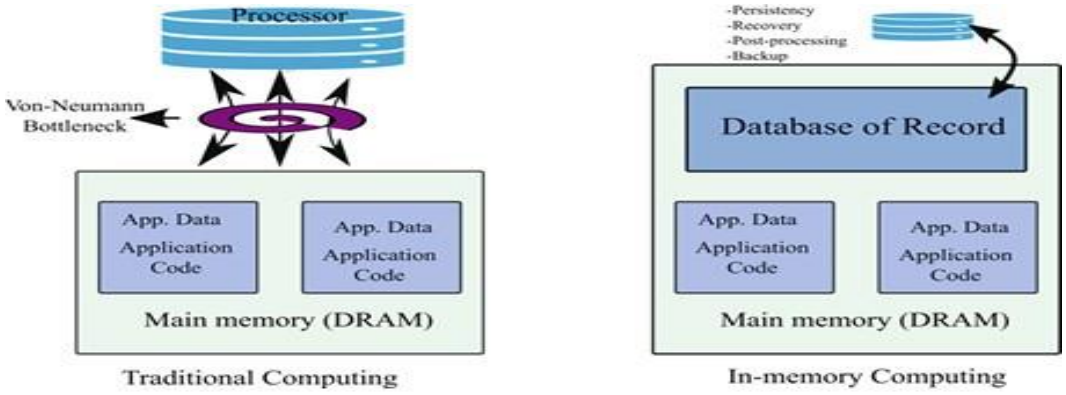


Fig. 1. Von Neumann bottleneck [31].

HADOOP ECOSYSTEM

Hadoop Ecosystem is neither a programming language nor a service, it is a platform or framework which solves big data problems. You can consider it as a suite which encompasses a number of services (ingesting, storing, analyzing and maintaining) inside it.

Below are the Hadoop components, that together form a Hadoop ecosystem.

HDFS -> Hadoop Distributed File System

YARN -> Yet Another Resource Negotiator

MapReduce -> Data processing using programming

Spark -> In-memory Data Processing

PIG, HIVE-> Data Processing Services using Query (SQL-like)

HBase -> NoSQL Database

Mahout, Spark MLlib -> Machine Learning

Apache Drill -> SQL on Hadoop

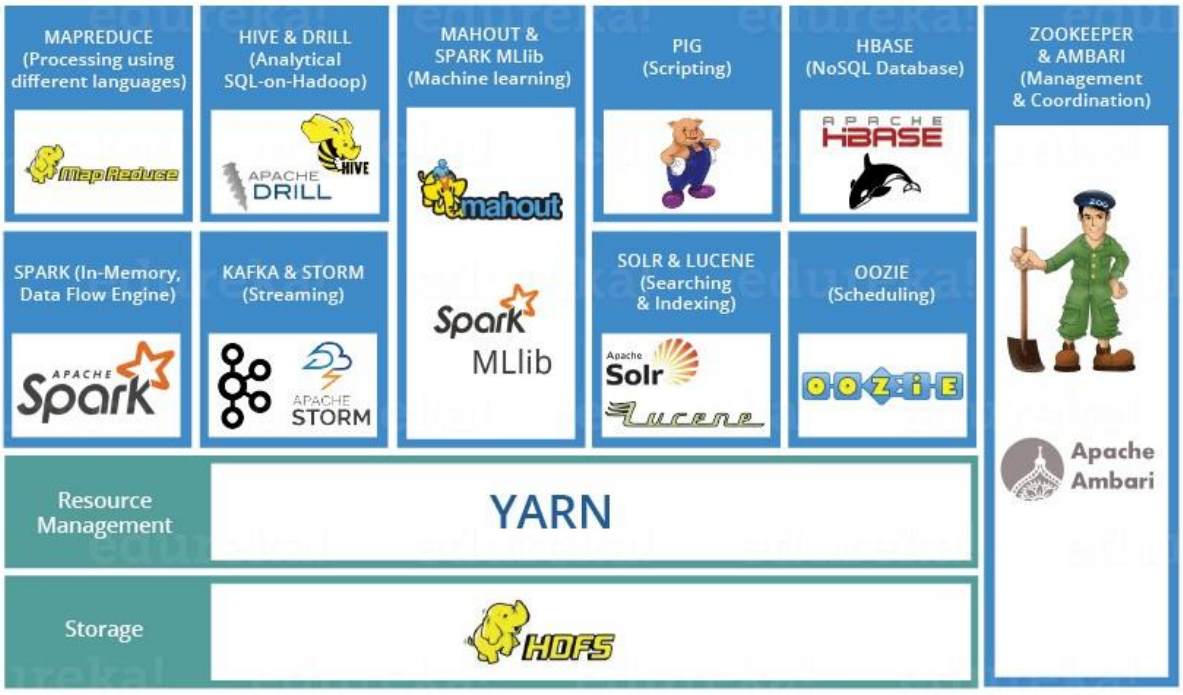
Zookeeper -> Managing Cluster

Oozie -> Job Scheduling

Flume, Sqoop -> Data Ingesting Services

Solr& Lucene -> Searching & Indexing

Ambari -> Provision, Monitor and Maintain cluster



HDFS

Hadoop Distributed File System is the core component or you can say, the backbone of Hadoop Ecosystem.

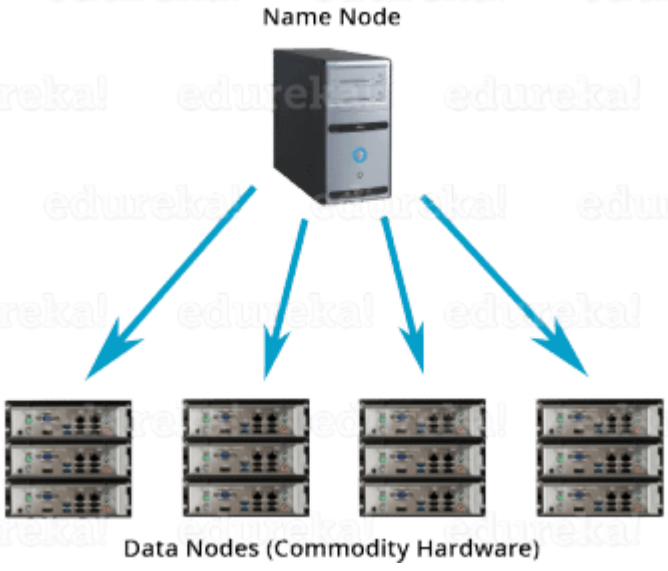
HDFS is the one, which makes it possible to store different types of large data sets (i.e. structured, unstructured and semi structured data).

HDFS creates a level of abstraction over the resources, from where we can see the whole HDFS as a single unit.

It helps us in storing our data across various nodes and maintaining the log file about the stored data (metadata).

HDFS has two core components, i.e. **NameNode and DataNode**.

1. The **NameNode** is the main node and it doesn't store the actual data. It contains metadata, just like a log file or you can say as a table of content. Therefore, it requires less storage and high computational resources.
2. On the other hand, all your data is stored on the **DataNodes** and hence it requires more storage resources. These DataNodes are commodity hardware (like your laptops and desktops) in the distributed environment. That's the reason, why Hadoop solutions are very cost effective.
3. You always communicate to the NameNode while writing the data. Then, it internally sends a request to the client to store and replicate data on various DataNodes.



YARN

Consider YARN as the brain of your Hadoop Ecosystem. It performs all your processing activities by allocating resources and scheduling tasks.

It has two major components, i.e. **Resource Manager and Node Manager**.

1. **Resource Manager** is again a main node in the processing department.
2. It receives the processing requests, and then passes the parts of requests to corresponding Node Managers accordingly, where the actual processing takes place.
3. **Node Managers** are installed on every Data Node. It is responsible for execution of task on every single Data Node.

'

1. **Schedulers:** Based on your application resource requirements, Schedulers perform scheduling algorithms and allocates the resources.
2. **Applications Manager:** While Applications Manager accepts the job submission, negotiates to containers (i.e. the Data node environment where process executes) for executing the application specific Application Master and monitoring the progress. ApplicationMasters are the deamons which reside on DataNode and communicates to containers for execution of tasks on each DataNode.
3. ResourceManager has two components: Schedulers and application manager

MAPREDUCE

It is the core component of processing in a Hadoop Ecosystem as it provides the logic of processing. In other words, MapReduce is a software framework which helps in writing applications that processes large data sets using distributed and parallel algorithms inside Hadoop environment.

In a MapReduce program, **Map() and Reduce()** are two functions.

1. The **Map function** performs actions like filtering, grouping and sorting.
2. While **Reduce function** aggregates and summarizes the result produced by map function.
3. The result generated by the Map function is a key value pair (K, V) which acts as the input for Reduce function.

Student	Department	Count	(Key, Value), Pair
Student 1	D1	1	(D1, 1)
Student 2	D1	1	(D1, 1)
Student 3	D1	1	(D1, 1)
Student 4	D2	1	(D2, 1)
Student 5	D2	1	(D2, 1)
Student 6	D3	1	(D3, 1)
Student 7	D3	1	(D3, 1)

Let us take the above example to have a better understanding of a MapReduce program.

We have a sample case of students and their respective departments. We want to calculate the number of students in each department. Initially, Map program will execute and calculate the students appearing in each department, producing the key value pair as mentioned above. This key value pair is the input to the Reduce function. The Reduce function will then aggregate each department and calculate the total number of students in each department and produce the given result.

Department	Total Student
D1	3
D2	2
D3	2

APACHE PIG

PIG has two parts: **Pig Latin**, the language and **the pig runtime,** for the execution environment. You can better understand it as Java and JVM.
It supports *pig latin* language, which has SQL like command structure.

10 line of pig latin = approx. 200 lines of Map-Reduce Java code

But don't be shocked when I say that at the back end of Pig job, a map-reduce job executes.

The compiler internally converts pig latin to MapReduce. It produces a sequential set of MapReduce jobs, and that's an abstraction (which works like black box).
PIG was initially developed by Yahoo.

It gives you a platform for building data flow for ETL (Extract, Transform and Load), processing and analyzing huge data sets.

How Pig works?

In PIG, first the load command, loads the data. Then we perform various functions on it like grouping, filtering, joining, sorting, etc. At last, either you can dump the data on the screen or you can store the result back in HDFS.

APACHE HIVE

Facebook created HIVE for people who are fluent with SQL. Thus, HIVE makes them feel at home while working in a Hadoop Ecosystem.
Basically, HIVE is a data warehousing component which performs reading, writing and managing large data sets in a distributed environment using SQL-like interface.

HIVE + SQL = HQL

The query language of Hive is called Hive Query Language(HQL), which is very similar like SQL.
It has 2 basic components: **Hive Command Line and JDBC/ODBC driver**.

The **Hive Command line** interface is used to execute HQL commands.

While, Java Database Connectivity (**JDBC**) and Object Database Connectivity (**ODBC**) is used to establish connection from data storage.

Secondly, Hive is highly scalable. As, it can serve both the purposes, i.e. large data set processing (i.e. Batch query processing) and real time processing (i.e. Interactive query processing).
It supports all primitive data types of SQL.

You can use predefined functions, or write tailored user defined functions (UDF) also to accomplish your specific needs.

APACHE MAHOUT

Now, let us talk about Mahout which is renowned for machine learning. Mahout provides an environment for creating machine learning applications which are scalable.

Machine learning algorithms allow us to build self-learning machines that evolve by itself without being explicitly programmed. Based on user behaviour, data patterns and past experiences it makes important future decisions. You can call it a descendant of Artificial Intelligence (AI).

What Mahout does?

It performs **collaborative filtering, clustering and classification.** Some people also consider **frequent item set missing** as Mahout's function. Let us understand them individually:

1. **Collaborative filtering:** Mahout mines user behaviors, their patterns and their characteristics and based on that it predicts and make recommendations to the users. The typical use case is E-commerce website.
2. **Clustering:** It organizes a similar group of data together like articles can contain blogs, news, research papers etc.
3. **Classification**: It means classifying and categorizing data into various sub- departments like articles can be categorized into blogs, news, essay, research papers and other categories.
4. **Frequent item set missing**: Here Mahout checks, which objects are likely to be appearing together and make suggestions, if they are missing. For example, cell phone and cover are brought together in general. So, if you search for a cell phone, it will also recommend you the cover and cases.

Mahout provides a command line to invoke various algorithms. It has a predefined set of library which already contains different inbuilt algorithms for different use cases.

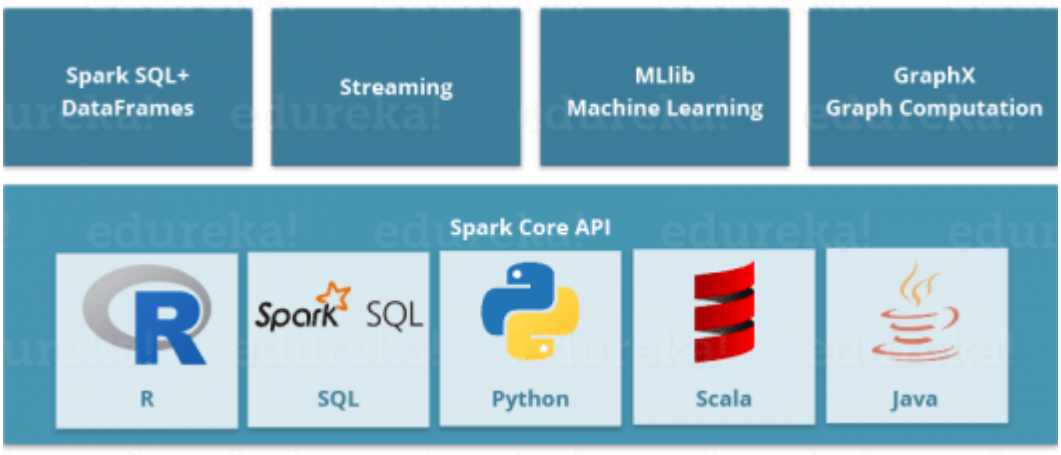
APACHE SPARK

Apache Spark is a framework for real time data analytics in a distributed computing environment.

The Spark is written in Scala and was originally developed at the University of California, Berkeley.

It executes in-memory computations to increase speed of data processing over Map-Reduce.

It is 100x faster than Hadoop for large scale data processing by exploiting in-memory computations and other optimizations. Therefore, it requires high processing power than Map-Reduce.



As you can see, Spark comes packed with high-level libraries, including support for R, SQL, Python, Scala, Java etc. These standard libraries increase the seamless integrations in complex workflow. Over this, it also allows various sets of services to integrate with it like MLlib, GraphX, SQL + Data Frames, Streaming services etc. to increase its capabilities.

. Apache Spark best fits for real time processing, whereas Hadoop was designed to store unstructured data and execute batch processing over it. When we combine, Apache Spark's ability, i.e. high processing speed, advance analytics and multiple integration support with Hadoop's low cost operation on commodity hardware, it gives the best results.

That is the reason why, Spark and Hadoop are used together by many companies for processing and analyzing their Big Data stored in HDFS.

APACHE HBASE

HBase is an open source, non-relational distributed database. In other words, it is a NoSQL database.

It supports all types of data and that is why, it's capable of handling anything and everything inside a Hadoop ecosystem.

It is modelled after Google's BigTable, which is a distributed storage system designed to cope up with large data sets.

The HBase was designed to run on top of HDFS and provides BigTable like capabilities.

It gives us a fault tolerant way of storing sparse data, which is common in most Big Data use cases.

The HBase is written in Java, whereas HBase applications can be written in REST, Avro and Thrift APIs.

For better understanding, let us take an example. You have billions of customer emails and you need to find out the number of customers who has used the word complaint in their emails. The request needs to be processed quickly (i.e. at real time). So, here we are handling a large data set while retrieving a small amount of data. For solving these kind of problems, HBase was designed.

APACHE DRILL

Apache Drill is used to drill into any kind of data. It's an open source application which works with distributed environment to analyze large data sets.

It is a replica of Google Dremel.

It supports different kinds NoSQL databases and file systems, which is a powerful feature of Drill. For example: Azure Blob Storage, Google Cloud Storage, HBase, MongoDB, MapR-DB HDFS, MapR-FS, Amazon S3, Swift, NAS and local files.

So, basically the main aim behind Apache Drill is to provide scalability so that we can process petabytes and exabytes of data efficiently (or you can say in minutes).

The main power of Apache Drill lies in ***combining a variety of data stores just by using a single query.***

Apache Drill basically follows the ANSI SQL.

It has a powerful scalability factor in supporting millions of users and serve their query requests over large scale data.

APACHE ZOOKEEPER

Apache Zookeeper is the coordinator of any Hadoop job which includes a combination of various services in a Hadoop Ecosystem.

Apache Zookeeper coordinates with various services in a distributed environment.

Before Zookeeper, it was very difficult and time consuming to coordinate between different services in Hadoop Ecosystem. The services earlier had many problems with interactions like common configuration while synchronizing data. Even if the services are configured, changes in the configurations of the services make it complex and difficult to handle. The grouping and naming was also a time-consuming factor.

Due to the above problems, Zookeeper was introduced. It saves a lot of time by performing **synchronization, configuration maintenance, grouping and naming.**

Although it's a simple service, it can be used to build powerful solutions.

APACHE OOZIE

Consider Apache Oozie as a clock and alarm service inside Hadoop Ecosystem. For Apache jobs, Oozie has been just like a scheduler. It schedules Hadoop jobs and binds them together as one logical work.

There are two kinds of Oozie jobs:

1. **Oozie workflow**: These are sequential set of actions to be executed**.** You can assume it as a relay race. Where each athlete waits for the last one to complete his part.
2. **Oozie Coordinator**: These are the Oozie jobs which are triggered when the data is made available to it. Think of this as the response-stimuli system in our body. In the same manner as we respond to an external stimulus, an Oozie coordinator responds to the availability of data and it rests otherwise.

APACHE FLUME

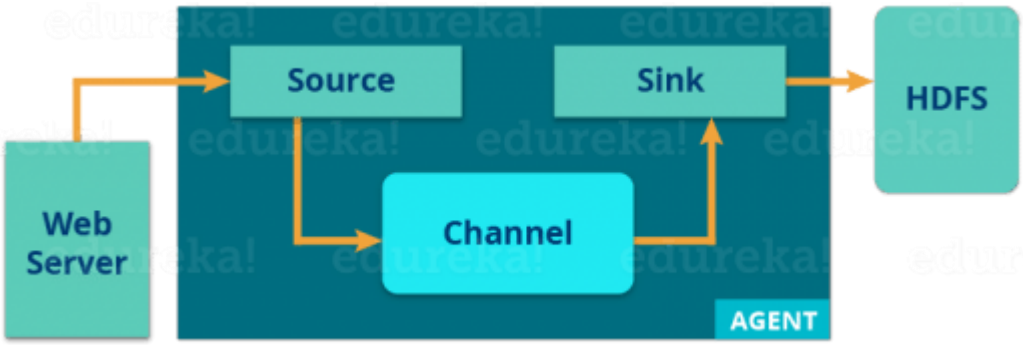
Ingesting data is an important part of our Hadoop Ecosystem.

The Flume is a service which helps in ingesting unstructured and semi-structured data into HDFS.

It gives us a solution which is reliable and distributed and helps us in **collecting, aggregating** and **moving large amount of data sets**.

It helps us to ingest online streaming data from various sources like network traffic, social media, email messages, log files etc. in HDFS.

Now, let us understand the architecture of Flume from the below diagram:



There is a **Flume agent** which ingests the streaming data from various data sources to HDFS. From the diagram, you can easily understand that the web server indicates the data source. Twitter is among one of the famous sources for streaming data.

The flume agent has 3 components: **source, sink and channel**.

1. **Source**: it accepts the data from the incoming streamline and stores the data in the channel.
2. **Channel:** it acts as the local storage or the primary storage. A Channel is a temporary storage between the source of data and persistent data in the HDFS.
3. **Sink:** Then, our last component i.e. Sink, collects the data from the channel and commits or writes the data in the HDFS permanently.

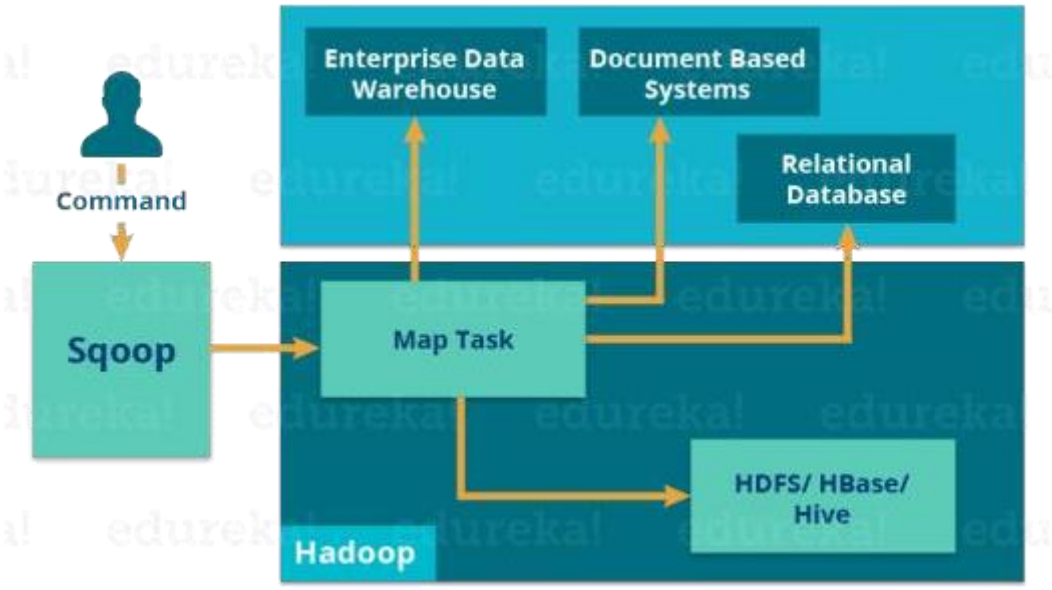
APACHE SQOOP

The major difference between Flume and Sqoop is that:

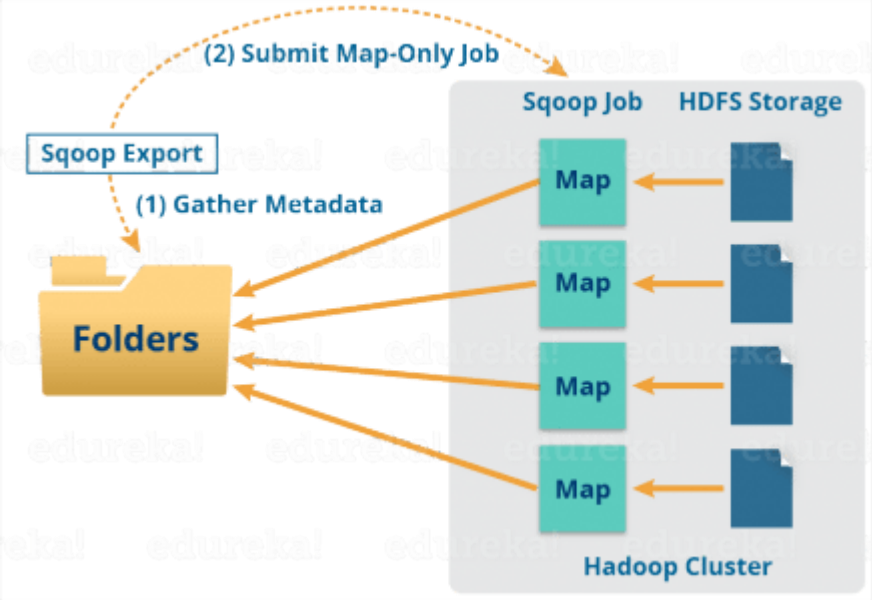
Flume only ingests unstructured data or semi-structured data into HDFS.

While Sqoop can import as well as export structured data from RDBMS or Enterprise data warehouses to HDFS or vice versa.

Let us understand how Sqoop works using the below diagram:



When we submit Sqoop command, our main task gets divided into sub tasks which is handled by individual Map Task internally. Map Task is the sub task, which imports part of data to the Hadoop Ecosystem. Collectively, all Map tasks imports the whole data.



Export also works in a similar manner.

When we submit our Job, it is mapped into Map Tasks which brings the chunk of data from HDFS. These chunks are exported to a structured data destination. Combining all these exported chunks of data, we receive the whole data at the destination, which in most of the cases is an RDBMS (MYSQL/Oracle/SQL Server).

APACHE SOLR & LUCENE

Apache Solr and Apache Lucene are the two services which are used for searching and indexing in Hadoop Ecosystem.

Apache Lucene is based on Java, which also helps in spell checking.

If Apache Lucene is the engine, Apache Solr is the car built around it. Solr is a complete application built around Lucene.

It uses the Lucene Java search library as a core for search and full indexing.

APACHE AMBARI

Ambari is an Apache Software Foundation Project which aims at making Hadoop ecosystem more manageable.

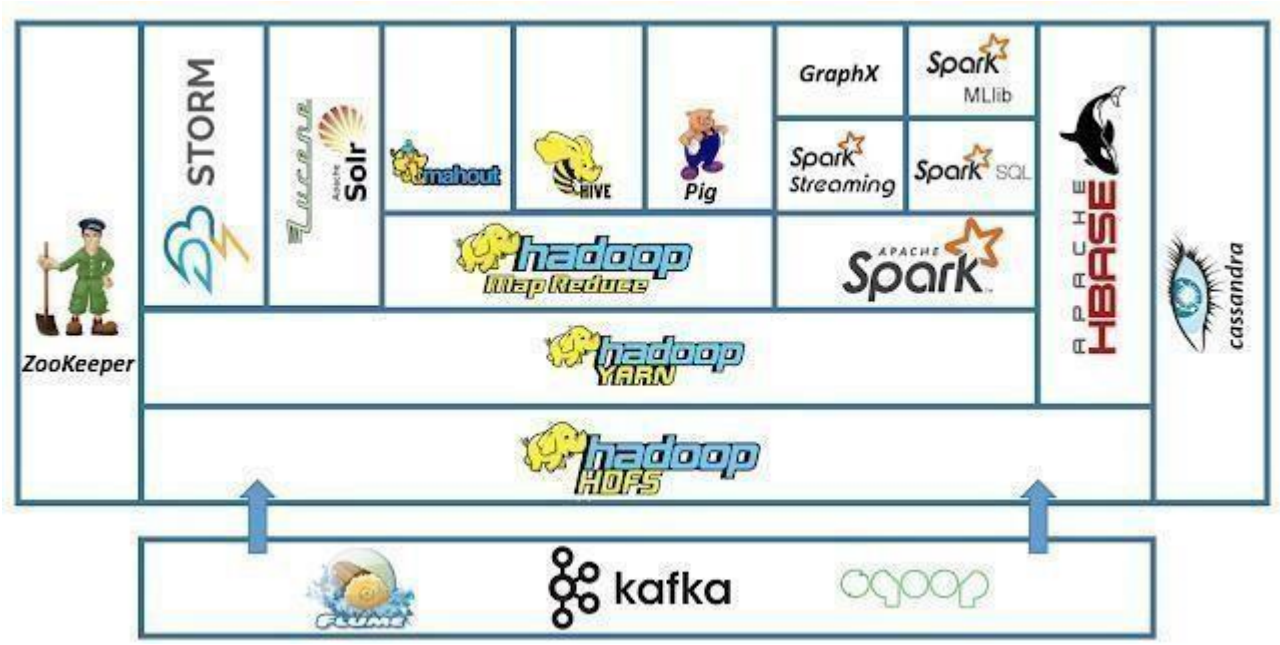
It includes software for **provisioning, managing and monitoring** Apache Hadoop clusters.

The Ambari provides:

1. **Hadoop cluster provisioning**:
   - It gives us step by step process for installing Hadoop services across a number of hosts.
   - It also handles configuration of Hadoop services over a cluster.
2. **Hadoop cluster management:**
   - It provides a central management service for starting, stopping and re-configuring Hadoop services across the cluster.
3. **Hadoop cluster monitoring:**
   - For monitoring health and status, Ambari provides us a dashboard.
   - The **Amber Alert framework** is an alerting service which notifies the user, whenever the attention is needed. For example, if a node goes down or low disk space on a node, etc.

1. Hadoop Ecosystem owes its success to the whole developer community, many big companies like Facebook, Google, Yahoo, University of California (Berkeley) etc. have contributed their part to increase Hadoop's capabilities.
2. Inside a Hadoop Ecosystem, knowledge about one or two tools (Hadoop components) would not help in building a solution. You need to learn a set of Hadoop components, which works together to build a solution.
3. Based on the use cases, we can choose a set of services from Hadoop Ecosystem and create a tailored solution for an organization.

Understanding Hadoop Eco System



Hadoop HDFS - 2007 - A distributed file system for reliably storing huge amounts of unstructured, semi-structured and structured data in the form of files.

Hadoop MapReduce - 2007 - A distributed algorithm framework for the parallel processing of large datasets on HDFS filesystem. It runs on Hadoop cluster but also supports other database formats like Cassandra and HBase.

Cassandra - 2008 - A key-value pair NoSQL database, with column family data representation and asynchronous masterless replication.

HBase - 2008 - A key-value pair NoSQL database, with column family data representation, with master-slave replication. It uses HDFS as underlying storage.

Zookeeper - 2008 - A distributed coordination service for distributed applications. It is based on Paxos algorithm variant called Zab.

Pig - 2009 - Pig is a scripting interface over MapReduce for developers who prefer scripting interface over native Java MapReduce programming.

Hive - 2009 - Hive is a SQL interface over MapReduce for developers and analysts who prefer SQL interface over native Java MapReduce programming.

Mahout - 2009 - A library of machine learning algorithms, implemented on top of MapReduce, for finding meaningful patterns in HDFS datasets.

Sqoop - 2010 - A tool to import data from RDBMS/DataWarehouse into HDFS/HBase and export back.

YARN - 2011 - A system to schedule applications and services on an HDFS cluster and manage the cluster resources like memory and CPU.

Flume - 2011 - A tool to collect, aggregate, reliably move and ingest large amounts of data into HDFS.

Storm - 2011 - A system to process high-velocity streaming data with 'at least once' message semantics.

Spark - 2012 - An in-memory data processing engine that can run a DAG of operations. It provides libraries for Machine Learning, SQL interface and near real-time Stream Processing.

Kafka - 2012 - A distributed messaging system with partitioned topics for very high scalability.

Solr Cloud - 2012 - A distributed search engine with a REST-like interface for full-text search. It uses Lucene library for data indexing.

Hadoop Environment Big Data Analytics Hadoop is changing the perception of handling Big Data especially the unstructured data. Let's know how Apache Hadoop software library, which is a framework, plays a vital role in handling Big Data. Apache Hadoop enables surplus data to be streamlined for any distributed processing system across clusters of computers using simple programming models. It truly is made to scale up from single servers to many machines, each and every offering local computation, and storage space.

Instead of depending on hardware to provide high availability, the library itself is built to detect and handle breakdowns at the application layer, so providing an extremely available service along with a cluster of computers, as both versions might be vulnerable to failures.

Hadoop Community Package Consists of

- File system and OS level abstractions
- A MapReduce engine (either MapReduce or YARN)
- The Hadoop Distributed File System (HDFS)
- Java ARchive (JAR) files
- Scripts needed to start Hadoop
- Source code, documentation and a contribution section

UNIT III

Big Data processing:

Big Data technologies, Introduction to Google file system, Hadoop Architecture, Hadoop Storage: HDFS, Common Hadoop Shell commands, NameNode, Secondary NameNode, and DataNode, Hadoop MapReduce paradigm, Map Reduce tasks, Job, Task trackers, Introduction to NOSQL, Textual ETL processing.

Big Data Technologies

Big data technology and Hadoop is a big buzzword as it might sound. As there has been a huge increase in the data and information domain from every industry and domain, it becomes very important to establish and introduce an efficient technique that takes care of all the needs and requirements of clients and big industries which are responsible for data generation. Earlier the data was being handled by normal programming languages and simple structured query language but now these systems and tools don't seem to do much in case of big data.

Big data technology is defined as the technology and a software utility that is designed for analysis, processing, and extraction of the information from a large set of extremely complex structures and large data sets which is very difficult for the traditional systems to deal with. Big data technology is used to handle both real-time and batch related data. Machine learning has become a very critical component of everyday lives and every industry and therefore managing data through big data becomes very important.

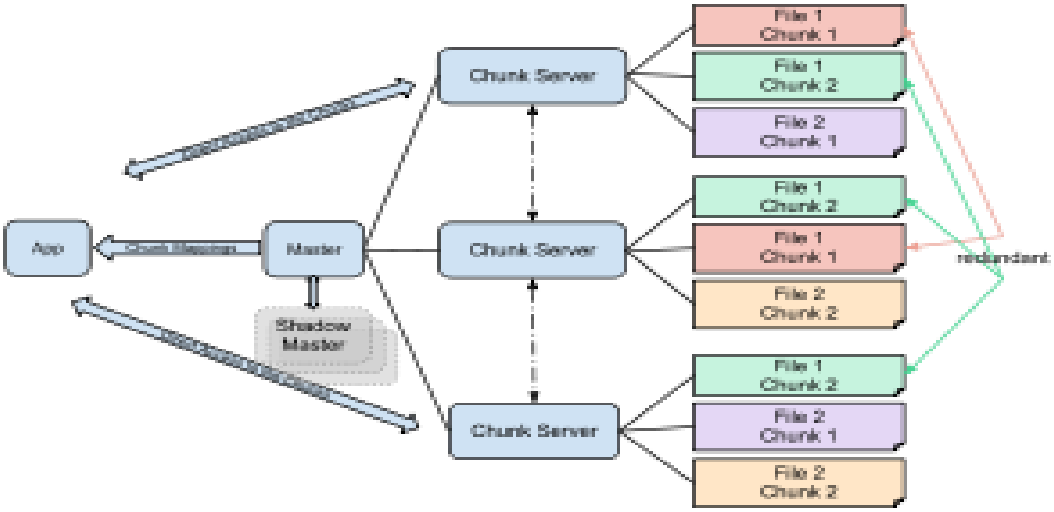
Hadoop:

When it comes to big data, Hadoop is the first technology that comes into play. This is based on map-reduce architecture and helps in the processing of batch related jobs and process batch information. It was designed to store and process the data in a distributed data processing environment along with commodity hardware and a simple programming execution model. It can be used to store and analyze the data present in various different machines with high storage, speed, and low costs. This forms one of the main core components of big data technology which was developed by the Apache software foundation in the year 2011 and is written in Java. Visualization

Google File System

The Google file system (GFS) is a distributed file system (DFS) for **data-centric applications with robustness, scalability, and reliability**. GFS can be implemented in commodity servers to support large-scale file applications with high performance and high reliability.

Google File System (GFS) is **a scalable distributed file system** (DFS) created by Google Inc. and developed to accommodate Google's expanding data processing requirements. GFS provides fault tolerance, reliability, scalability, availability, and performance to large networks and connected nodes.

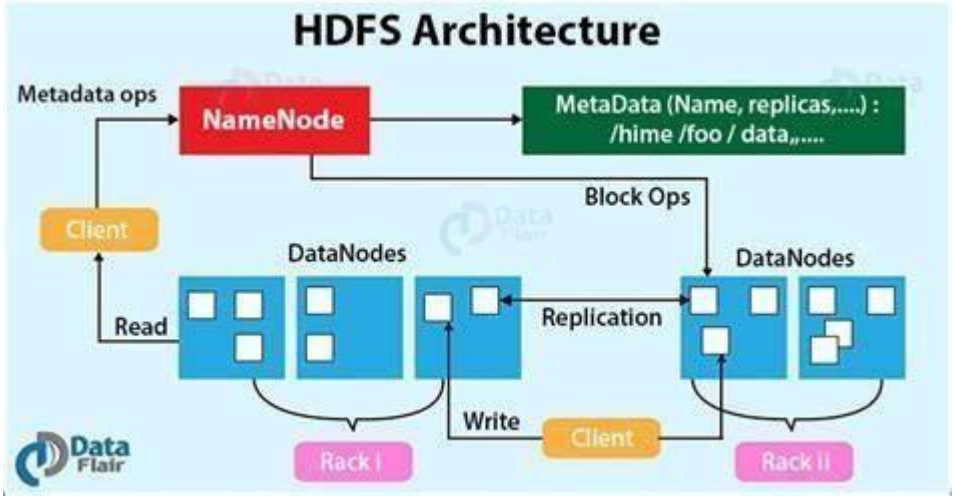


GFS is enhanced for Google's core data storage and usage needs (primarily the search engine), which can generate enormous amounts of data that must be retained; Google File System grew out of an earlier Google effort, "BigFiles", developed by Larry Page and Sergey Brin in the early days of Google, while it was still located in Stanford. Files are divided into fixed-size *chunks* of 64 megabytes, similar to clusters or sectors in regular file systems, which are only extremely rarely overwritten, or shrunk; files are usually appended to or read.

A GFS cluster consists of multiple nodes. These nodes are divided into two types: one *Master* node and multiple *Chunk servers*. Each file is divided into fixed-size chunks. Chunk servers store these chunks. Each chunk is assigned a globally unique 64-bit label by the master node at the time of creation, and logical mappings of files to constituent chunks are maintained. Each chunk is replicated several times throughout the network. At default, it is replicated three times, but this is configurable. Files which are in high demand may have a higher replication factor, while files for which the application client uses strict storage optimizations may be replicated less than three times - in order to cope with quick garbage cleaning policies.

HDFS architecture

HDFS can be presented as the master/slave architecture. HDFS master is named as NameNode whereas slave as DataNode. NameNode is a sever that manages the filesystem namespace and adjusts the access (open, close, rename, and more) to files by the client. It divides the input data into blocks and announces which data block will be store in which DataNode. DataNode is a slave machine that stores the replicas of the partitioned dataset and serves the data as the request comes. It also performs block creation and deletion.



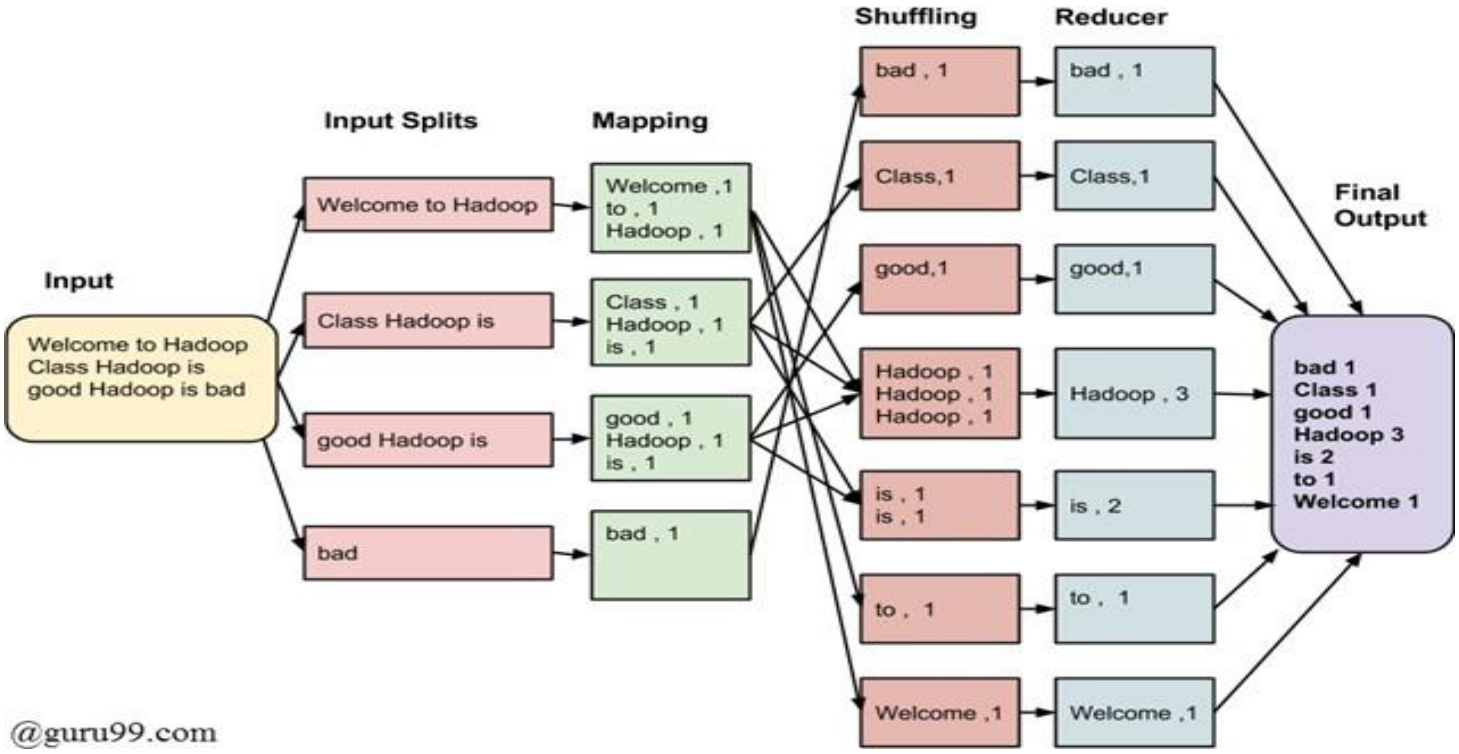
The internal mechanism of HDFS divides the file into one or more blocks; these blocks are stored in a set of data nodes. Under normal circumstances of the replication factor three, the HDFS strategy is to place the first copy on the local node, second copy on the local rack with a different node, and a third copy into different racks with different nodes. As HDFS is designed to support large files, the HDFS block size is defined as 64 MB. If required, this can be increased.

Understanding HDFS components HDFS is managed with the master-slave architecture included with the following components:

- Name Node: This is the master of the HDFS system. It maintains the directories, files, and manages the blocks that are present on the Data Nodes.
- Data Node: These are slaves that are deployed on each machine and provide actual storage. They are responsible for serving read-and-write data requests for the clients.
- Secondary Name Node: This is responsible for performing periodic checkpoints. So, if the Name Node fails at any time, it can be replaced with a snapshot image stored by the secondary Name Node checkpoints.

MapReduce architecture

MapReduce is also implemented over master-slave architectures. Classic MapReduce contains job submission, job initialization, task assignment, task execution, progress and status update, and job completion-related activities, which are mainly managed by the Job Tracker node and executed by Task Tracker. Client application submits a job to the Job Tracker. Then input is divided across the cluster. The Job Tracker then calculates the number of map and reducer to be processed. It commands the Task Tracker to start executing the job. Now, the Task Tracker copies the resources to a local machine and launches JVM to map and reduce program over the data.



Example WordCount Mapreduce architecture

Along with this, the Task Tracker periodically sends update to the Job Tracker, which can be considered as the heartbeat that helps to update JobID, job status, and usage of resources.

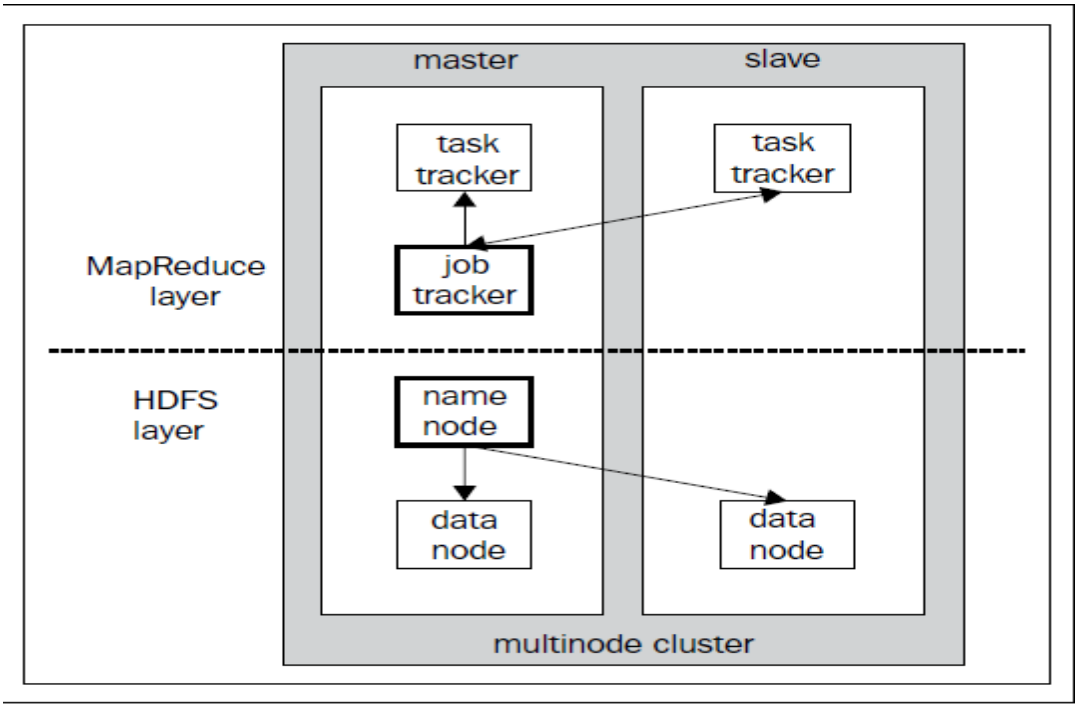
Understanding MapReduce components

MapReduce is managed with master-slave architecture included with the following components:

- JobTracker: This is the master node of the MapReduce system, which manages the jobs and resources in the cluster (TaskTrackers). The JobTracker tries to schedule each map as close to the actual data being processed on the TaskTracker, which is running on the same DataNode as the underlying block.
- TaskTracker: These are the slaves that are deployed on each machine. They are responsible for running the map and reducing tasks as instructed by the JobTracker.

Understanding the HDFS and MapReduce architecture by plot

In this plot, both HDFS and MapReduce master and slave components have been included, where NameNode and DataNode are from HDFS and JobTracker and TaskTracker are from the MapReduce paradigm.Both paradigms consisting of master and slave candidates have their own specific responsibility to handle MapReduce and HDFS operations. In the next plot, there is a plot with two sections: the preceding one is a MapReduce layer and the following one is an HDFS layer.



The HDFS and MapReduce architecture

Hadoop is a top-level Apache project and is a very complicated Java framework. To avoid technical complications, the Hadoop community has developed a number of Java frameworks that has added an extra value to Hadoop features. They are considered as Hadoop subprojects. Here, we are departing to discuss several Hadoop components that can be considered as an abstraction of HDFS or MapReduce.

Hadoop MapReduce fundamentals

Basically, the MapReduce model can be implemented in several languages, but apart from that, Hadoop MapReduce is a popular Java framework for easily written applications. It processes vast amounts of data (multiterabyte datasets) in parallel on large clusters (thousands of nodes) of commodity hardware in a reliable and fault tolerant manner. This MapReduce paradigm is divided into two phases, Map and Reduce, that mainly deal with key-value pairs of data. The Map and Reduce tasks run sequentially in a cluster, and the output of the Map phase becomes the input of the Reduce phase.

All data input elements in MapReduce cannot be updated. If the input (key,value) pairs for mapping tasks are changed, it will not be reflected in the input files. The Mapper output will be piped to the appropriate Reducer grouped with the key attribute as input. This sequential data process will be carried away in a parallel manner with the help of Hadoop MapReduce algorithms as well as Hadoop clusters.

MapReduce programs transform the input dataset present in the list format into output data that will also be in the list format. This logical list translation process is mostly repeated twice in the Map and Reduce phases. We can also handle these repetitions by fixing the number of Mappers and Reducers.

The following are the components of Hadoop that are responsible for performing analytics over Big Data:

- Client: This initializes the job
- JobTracker: This monitors the job
- TaskTracker: This executes the job
- HDFS: This stores the input and output data

The four main stages of Hadoop MapReduce data processing are as follows:

- The loading of data into HDFS
- The execution of the Map phase
- Shuffling and sorting
- The execution of the Reduce phase

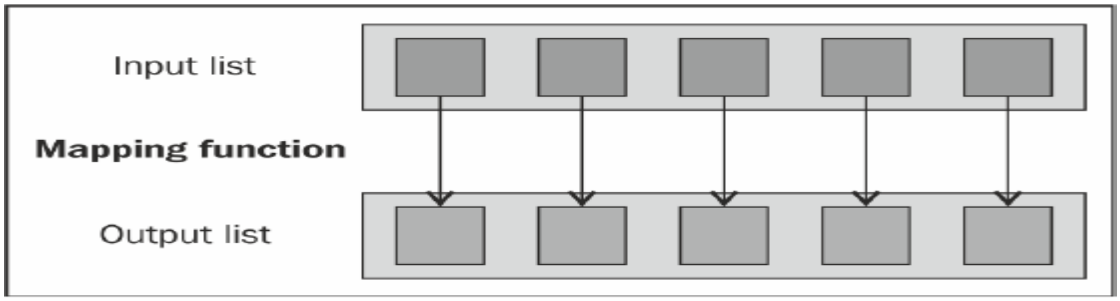
Loading data into HDFS

The input dataset needs to be uploaded to the Hadoop directory so it can be used by MapReduce nodes. Then, Hadoop Distributed File System (HDFS) will divide the input dataset into data splits and store them to Data Nodes in a cluster by taking care of the replication factor for fault tolerance. All the data splits will be processed by Task Tracker for the Map and Reduce tasks in a parallel manner.

Executing the Map phase

Executing the client application starts the Hadoop MapReduce processes. The Map phase then copies the job resources (unjarred class files) and stores it to HDFS, and requests Job Tracker to execute the job. The Job Tracker initializes the job, retrieves the input, splits the information, and creates a Map task for each job.

The Job Tracker will call Task Tracker to run the Map task over the assigned input data subset. The Map task reads this input split data as input (key, value) pairs provided to the Mapper method, which then produces intermediate (key, value) pairs. There will be at least one output for each input (key, value) pair.

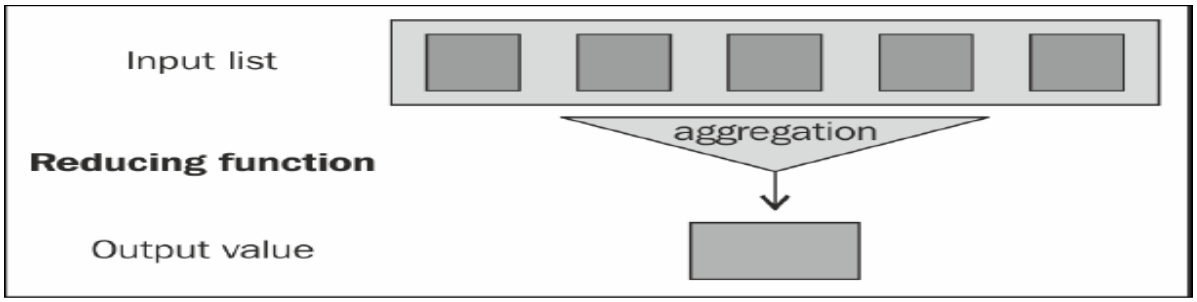


Mapping individual elements of an input list

The list of (key, value) pairs is generated such that the key attribute will be repeated many times. So, its key attribute will be re-used in the Reducer for aggregating values in MapReduce. As far as format is concerned, Mapper output format values and Reducer input values must be the same.

After the completion of this Map operation, the Task Tracker will keep the result in its buffer storage and local disk space (if the output data size is more than the threshold). For example, suppose we have a Map function that converts the input text into lowercase. This will convert the list of input strings into a list of lowercase strings. Reducing phase execution As soon as the Mapper output is available, Task Tracker in the Reducer node will retrieve the available partitioned Map's output data, and they will be grouped together and merged into one large file, which will

then be assigned to a process with a Reducer method. Finally, this will be sorted out before data is provided to the Reducer method. The Reducer method receives a list of input values from an input (key, list (value)) and aggregates them based on custom logic, and produces the output (key, value) pairs.



Reducing input values to an aggregate value as output

The output of the Reducer method of the Reduce phase will directly be written into HDFS as per the format specified by the MapReduce job configuration class. The Hadoop MapReduce fundamentals

• Understand MapReduce objects

• Learn how to decide the number of Maps in MapReduce

• Learn how to decide the number of Reduces in MapReduce

• Understand MapReduce dataflow

• Take a closer look at Hadoop MapReduce terminologies

MapReduce objects

MapReduce operations in Hadoop are carried out mainly by three objects: Mapper, Reducer, and Driver.

• Mapper:

This is designed for the Map phase of MapReduce, which starts MapReduce operations by carrying input files and splitting them into several pieces. For each piece, it will emit a key-value data pair as the output value.

• Reducer:

This is designed for the Reduce phase of a MapReduce job; it accepts key-based grouped data from the Mapper output, reduces it by aggregation logic, and emits the (key, value) pair for the group of values.

• Driver:

This is the main file that drives the MapReduce process. It starts the execution of MapReduce tasks after getting a request from the client application with parameters. The Driver file is responsible for building the configuration of a job and submitting it to the Hadoop cluster. The Driver code will contain the main() method that accepts arguments from the command line. The input and output directory of the Hadoop MapReduce job will be accepted by this program. Driver is the main file for defining job configuration details, such as the job name, job input format, job output format, and the Mapper, Combiner, Partitioner, and Reducer classes. MapReduce is initialized by calling this main() function of the Driver class.

Deciding the number of Maps in Map Reduce

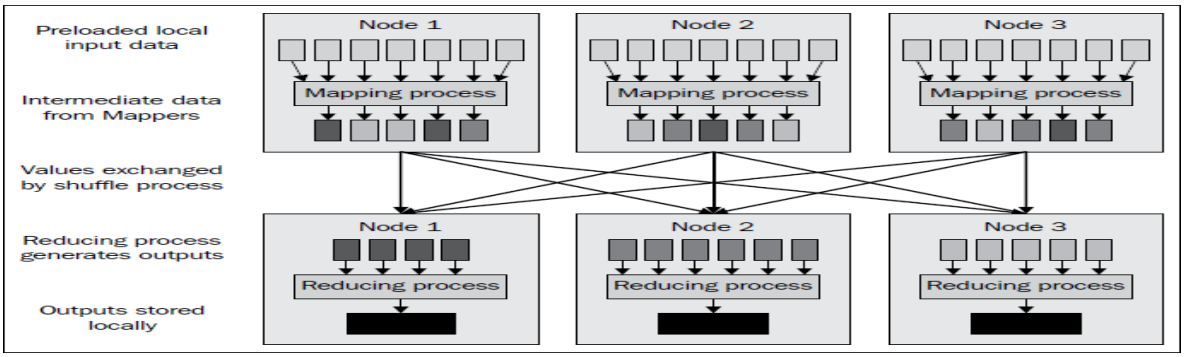
The number of Maps is usually defined by the size of the input data and size of the data split block that is calculated by the size of the HDFS file / data split. Therefore, if we have an HDFS data file of 5 TB and a block size of 128 MB, there will be 40,960 maps present in the file. But sometimes, the number of Mappers created will be more than this count because of speculative execution. This is true when the input is a file, though it entirely depends on the Input Format class. In Hadoop Map Reduce processing, there will be a delay in the result of the job when the assigned Mapper or Reducer is taking a long time to finish. If you want to avoid this, speculative execution in Hadoop can run multiple copies of the same Map or Reduce task on different nodes, and the result from the first completed nodes can be used. From the Hadoop API with the setNumMapTasks(int) method, we can get an idea of the number of Mappers.

Deciding the number of Reducers in MapReduce

Numbers of Reducers are created based on the Mapper's input. However, if you hardcode the number of Reducers in MapReduce, it won't matter how many nodes are present in a cluster. It will be executed as specified in the configuration. Additionally, we can set the number of Reducers at runtime along with the MapReduce command at the command prompt -D mapred reduce. tasks, with the number you want. Programmatically, it can be set via conf. setNumReduceTasks(int).

MapReduce dataflow

Now that we have seen the components that make a basic MapReduce job possible, we will distinguish how everything works together at a higher level. From the following diagram, we will understand MapReduce dataflow with multiple nodes in a Hadoop cluster:



Map reduce data flow

The two APIs available for Hadoop MapReduce are: New (Hadoop 1.x and 2.x) and Old Hadoop (0.20). YARN is the next generation of Hadoop MapReduce and the new Apache Hadoop subproject that has been released for Hadoop resource management.

Hadoop data processing includes several tasks that help achieve the final output from an input dataset. These tasks are as follows:

1. Preloading data in HDFS.

2. Running MapReduce by calling Driver.

3. Reading of input data by the Mappers, which results in the splitting of the data execution of the Mapper custom logic and the generation of intermediate key-value pairs

4. Executing Combiner and the shuffle phase to optimize the overall Hadoop MapReduce process.

5. Sorting and providing of intermediate key-value pairs to the Reduce phase.

The Reduce phase is then executed. Reducers take these partitioned key value pairs and aggregate them based on Reducer logic.

6. The final output data is stored at HDFS.

Here, Map and Reduce tasks can be defined for several data operations as follows:

• Data extraction

• Data loading

• Data segmentation

• Data cleaning

• Data transformation

• Data integration

Common Hadoop Shell commands

ls: This command is used to list all the files. Use *lsr* for recursive approach. It is useful when we want a hierarchy of a folder.

bin/hdfs dfs -ls <path>

mkdir:

To create a directory. In Hadoop *dfs* there is no home directory by default. So let's first create it.

Syntax:

bin/hdfs dfs -mkdir <folder name>

creating home directory:

hdfs/bin -mkdir /user

hdfs/bin -mkdir /user/username -> write the username of your computer

touchz: It creates an empty file.

Syntax:

bin/hdfs dfs -touchz <file_path>

copyFromLocal (or) put: To copy files/folders from local file system to hdfs store. This is the most important command. Local filesystem means the files present on the OS.

Syntax:

bin/hdfs dfs -copyFromLocal <local file path> <dest(present on hdfs)>

cat: To print file contents.

Syntax:

bin/hdfs dfs -cat <path>

copyToLocal (or) get: To copy files/folders from hdfs store to local file system.

Syntax:

bin/hdfs dfs -copyToLocal <<srcfile(on hdfs)> <local file dest>

moveFromLocal: This command will move file from local to hdfs.

Syntax:

bin/hdfs dfs -moveFromLocal <local src> <dest(on hdfs)>

cp: This command is used to copy files within hdfs. Lets copy folder *geeks* to *geeks_copied*.

Syntax:

bin/hdfs dfs -cp <src(on hdfs)> <dest(on hdfs)>

mv: This command is used to move files within hdfs. Lets cut-paste a file *myfile.txt* from *geeks* folder to *geeks_copied*.

Syntax:

bin/hdfs dfs -mv <src(on hdfs)> <src(on hdfs)>

rmr: This command deletes a file from HDFS *recursively*. It is very useful command when you want to delete a *non-empty directory*.

Syntax:

bin/hdfs dfs -rmr <filename/directoryName>

du: It will give the size of each file in directory.

Syntax:

bin/hdfs dfs -du <dirName>

Name Node

NameNode:

This is the master of the HDFS system. It maintains the directories, files, and manages the blocks that are present on the DataNodes.

- **DataNode:** These are slaves that are deployed on each machine and provide actual storage. They are responsible for serving read-and-write data requests for the clients.
- **Secondary NameNode**: This is responsible for performing periodic checkpoints. So, if the NameNode fails at any time, it can be replaced with a snapshot image stored by the secondary NameNode checkpoints.

HADOOP MAPREDUCE PARADIGM

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken

down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The Algorithm

Generally MapReduce paradigm is based on sending the computer to where the data resides MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

Map stage − The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

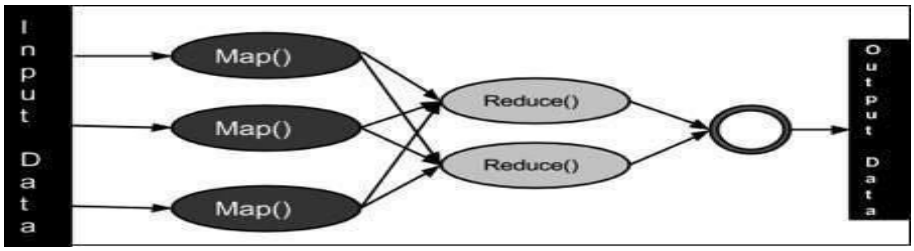
Reduce stage − This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-

Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job − (Input) <k1, v1> → map → <k2, v2> → reduce → <k3, v3>(Output).

	Input	**Output**
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

Terminology

PayLoad − Applications implement the Map and the Reduce functions, and form the core of the job.

Mapper − Mapper maps the input key/value pairs to a set of intermediate key/value pair.

NamedNode − Node that manages the Hadoop Distributed File System (HDFS).

DataNode − Node where data is presented in advance before any processing takes place.

MasterNode − Node where JobTracker runs and which accepts job requests from clients.

SlaveNode − Node where Map and Reduce program runs.

JobTracker − Schedules jobs and tracks the assign jobs to Task tracker.

Task Tracker − Tracks the task and reports status to JobTracker.

Job − A program is an execution of a Mapper and Reducer across a dataset.

Task − An execution of a Mapper or a Reducer on a slice of data.

Task Attempt − A particular instance of an attempt to execute a task on a SlaveNode.

Example Scenario

Given below is the data regarding the electrical consumption of an organization. It contains the monthly electrical consumption and the annual average for various years.

	Jan	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**	**Avg**
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34

1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

If the above data is given as input, we have to write applications to process it and produce results such as finding the year of maximum usage, year of minimum usage, and so on. This is a walkover for the programmers with finite number of records. They will simply write the logic to produce the required output, and pass the data to the application written.

But, think of the data representing the electrical consumption of all the large scale industries of a particular state, since its formation.

When we write applications to process such bulk data,they will take a lot of time to execute.

There will be a heavy network traffic when we move data from source to network server and so on.

To solve these problems, we have the MapReduce framework.

Input Data

The above data is saved as **sample.txt** and given as input. The input file looks as shown below.

1979 23 23 2 43 24 25 26 26 26 26 25 26 25

1980 26 27 28 28 28 30 31 31 31 30 30 30 29

1981 31 32 32 32 33 34 35 36 36 34 34 34 34

1984 39 38 39 39 39 41 42 43 40 39 38 38 40

1985 38 39 39 39 39 41 41 41 00 40 39 39 45

Example Program

Given below is the program to the sample data using MapReduce framework.

```
package hadoop;
import java.util.*;
import java.io.IOException;
import java.io.IOException;
import
org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import
```

```
org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
public class ProcessUnits {
```

```
//Mapper class
public static class E_EMapper extends MapReduceBase implements
Mapper<LongWritable ,/*Input key Type */
Text,                /*Input value Type*/
Text,                /*Output key Type*/
IntWritable>         /*Output value Type*/
{
  public void map(LongWritable key, Text value,
  OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
    String line = value.toString();
    String lasttoken = null;
    StringTokenizer s = new StringTokenizer(line,"\t");
    String year = s.nextToken();
          while(s.hasMoreTokens()) {
      lasttoken = s.nextToken(); }
    int avgprice = Integer.parseInt(lasttoken);
    output.collect(new Text(year), new IntWritable(avgprice));
  } }
  //Reducer class
public static class E_EReduce extends MapReduceBase implements Reducer< Text, IntWritable,
Text, IntWritable > {
    //Reduce function
  public void reduce( Text key, Iterator <IntWritable> values,
  OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
    int maxavg = 30;
    int val = Integer.MIN_VALUE;
            while (values.hasNext())
            {
      if((val = values.next().get())>maxavg) {
        output.collect(key, new IntWritable(val));
      }   } }   }
//Main function
```

```
public static void main(String args[])throws Exception {
```

```
    JobConf conf = new
        JobConf(ProcessUnits.class);
        conf.setJobName("max_eletricityunits");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(E_EMapper.class);
    conf.setCombinerClass(E_EReduce.class);
    conf.setReducerClass(E_EReduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        JobClient.runJob(conf);
  } }
```

Save the above program as **ProcessUnits.java.** The compilation and execution of the program is explained below.

Compilation and Execution of Process Units Program

Let us assume we are in the home directory of a Hadoop user (e.g. /home/hadoop).

Follow the steps given below to compile and execute the above program.

Step 1

The following command is to create a directory to store the compiled java classes.

```
$ mkdir units
```

Step 2

Download **Hadoop-core-1.2.1.jar,** which is used to compile and execute the MapReduce program. Visit the following link mvnrepository.com to download the jar. Let us assume the downloaded folder is **/home/hadoop/.**

Step 3

The following commands are used for compiling the **ProcessUnits.java** program and creating a jar for the program.

```
$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java
$ jar -cvf units.jar -C units/ .
```

Step 4

The following command is used to create an input directory in HDFS.

$HADOOP_HOME/bin/hadoop fs -mkdir input_dir

Step 5

The following command is used to copy the input file named **sample.txt**in the input directory of HDFS.

$HADOOP_HOME/bin/hadoop fs -put /home/hadoop/sample.txt input_dir

Step 6

The following command is used to verify the files in the input directory.

$HADOOP_HOME/bin/hadoop fs -ls input_dir/

Step 7

The following command is used to run the Eleunit_max application by taking the input files from the input directory.

$HADOOP_HOME/bin/hadoop jar units.jar hadoop.ProcessUnits input_dir output_dir

Wait for a while until the file is executed. After execution, as shown below, the output will contain the number of input splits, the number of Map tasks, the number of reducer tasks, etc.

INFO mapreduce.Job: Job job_1414748220717_0002
completed successfully
INFO mapreduce.Job: Counters:
49 File System Counters
FILE: Number of bytes read = 61
FILE: Number of bytes written = 279400
FILE: Number of read operations = 0
FILE: Number of large read operations =
0 FILE: Number of write operations = 0
HDFS: Number of bytes read = 546
HDFS: Number of bytes written = 40
HDFS: Number of read operations = 9
HDFS: Number of large read operations = 0
HDFS: Number of write operations = 2 Job
Counters Launched map tasks = 2
Launched reduce tasks =
1 Data-local map tasks =
2

Total time spent by all maps in occupied slots (ms) = 146137

Total time spent by all reduces in occupied slots (ms) = 441

Total time spent by all map tasks (ms) = 14613

Total time spent by all reduce tasks (ms) = 44120

Total vcore-seconds taken by all map tasks = 146137

Total vcore-seconds taken by all reduce tasks =

44120

Total megabyte-seconds taken by all map tasks = 149644288

Total megabyte-seconds taken by all reduce tasks =

45178880

Map-Reduce

Framework Map input

records = 5 Map output

records = 5 Map output

bytes = 45

Map output materialized bytes =

67 Input split bytes = 208

Combine input records = 5

Combine output records = 5

Reduce input groups = 5

Reduce shuffle bytes = 6

Reduce input records = 5

Reduce output records = 5

Spilled Records = 10

Shuffled Maps = 2

Failed Shuffles = 0

Merged Map outputs =

2

GC time elapsed (ms) = 948

CPU time spent (ms) =

5160

Physical memory (bytes) snapshot = 47749120

Virtual memory (bytes) snapshot = 2899349504

Total committed heap usage (bytes) = 277684224

File Output Format Counters

Bytes Written = 40

Step 8

The following command is used to verify the resultant files in the output folder.

$HADOOP_HOME/bin/hadoop fs -ls output_dir/

Step 9

The following command is used to see the output in **Part-00000** file. This file is generated by HDFS.

$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000

Below is the output generated by the MapReduce program.

1981 34

1984 40

1985 45

Step 10

The following command is used to copy the output folder from HDFS to the local file system for analyzing.

$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000/bin/hadoop dfs get output_dir /home/hadoop

Important Commands

All Hadoop commands are invoked by the **$HADOOP_HOME/bin/hadoop** command. Running the Hadoop script without any arguments prints the description for all commands.

Usage − hadoop [--config confdir] COMMAND

The following table lists the options available and their description.

Sr.No.	Option & Description
1	**namenode -format** Formats the DFS filesystem.
2	**secondarynamenode** Runs the DFS secondary namenode.
3	**namenode** Runs the DFS namenode.
4	**Datanode**

	Runs a DFS datanode.
5	**dfsadmin** Runs a DFS admin client.
6	**mradmin** Runs a Map-Reduce admin client.
7	**fsck** Runs a DFS filesystem checking utility.
8	**fs** Runs a generic filesystem user client.
9	**balancer** Runs a cluster balancing utility.
10	**oiv** Applies the offline fsimage viewer to an fsimage.
11	**fetchdt** Fetches a delegation token from the NameNode.
12	**jobtracker** Runs the MapReduce job Tracker node.
13	**pipes** Runs a Pipes job.
14	**tasktracker** Runs a MapReduce task Tracker node.
15	**historyserver** Runs job history servers as a standalone daemon.
16	**Job**

	Manipulates the MapReduce jobs.
17	**queue** Gets information regarding JobQueues.
18	**version** Prints the version.
19	**jar <jar>** Runs a jar file.
20	**distcp <srcurl> <desturl>** Copies file or directories recursively.
21	**distcp2 <srcurl> <desturl>** DistCp version 2.
22	**archive -archiveName NAME -p <parent path> <src>* <dest>** Creates a hadoop archive.
23	**classpath** Prints the class path needed to get the Hadoop jar and the required libraries.
24	**daemonlog** Get/Set the log level for each daemon

How to Interact with MapReduce Jobs

Usage − hadoop job [GENERIC_OPTIONS]

The following are the Generic Options available in a Hadoop job.

Sr.No.	GENERIC_OPTION & Description
1	**-submit <job-file>** Submits the job.

2	**-status <job-id>** Prints the map and reduce completion percentage and all job counters.
3	**-counter <job-id> <group-name> <countername>** Prints the counter value.
4	**-kill <job-id>** Kills the job.
5	**-events <job-id> <fromevent-#> <#-of-events>** Prints the events' details received by jobtracker for the given range.
6	**-history [all] <jobOutputDir> - history < jobOutputDir>** Prints job details, failed and killed tip details. More details about the job such as successful tasks and task attempts made for each task can be viewed by specifying the [all] option.
7	**-list[all]** Displays all jobs. -list displays only jobs which are yet to complete.
8	**-kill-task <task-id>** Kills the task. Killed tasks are NOT counted against failed attempts.
9	**-fail-task <task-id>** Fails the task. Failed tasks are counted against failed attempts.
10	**-set-priority <job-id> <priority>** Changes the priority of the job. Allowed priority values are VERY_HIGH, HIGH, NORMAL, LOW, VERY_LOW

To see the status of job

$ $HADOOP_HOME/bin/hadoop job -status <JOB-ID>

e.g.

$ $HADOOP_HOME/bin/hadoop job -status job_201310191043_0004

To see the history of job output-dir

```
$ $HADOOP_HOME/bin/hadoop job -history <DIR-NAME>
```

e.g.

$ $HADOOP_HOME/bin/hadoop job -history

/user/expert/output To kill the job

$ $HADOOP_HOME/bin/hadoop job -kill <JOB-ID>

e.g.

$ $HADOOP_HOME/bin/hadoop job -kill job_201310191043_0004

Introduction to NoSQL

A **NoSQL** originally referring to non SQL or non relational is a database that provides a mechanism for storage and retrieval of data. This data is modeled in means other than the tabular relations used in relational databases. Such databases came into existence in the late 1960s, but did not obtain the NoSQL moniker until a surge of popularity in the early twenty-first century.

The suitability of a given NoSQL database depends on the problem it should solve. Data structures used by NoSQL databases are sometimes also viewed as more flexible than relational database tables. Most NoSQL databases offer a concept of eventual consistency in which database changes are propagated to all nodes so queries for data might not return updated data immediately or might result in reading data that is not accurate which is a problem known as stale reads.

Advantages of NoSQL:

There are many advantages of working with NoSQL databases such as MongoDB and Cassandra. The main advantages are high scalability and high availability.

High scalability –

NoSQL database use sharing for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharing. Vertical scaling means adding more resources to the existing machine whereas horizontal scaling means adding more machines to handle the data.

Vertical scaling is not that easy to implement but horizontal scaling is easy to implement. Examples of horizontal scaling databases are MongoDB, Cassandra etc. NoSQL can handle huge amount of data because of scalability, as the data grows NoSQL scale itself to handle that data in efficient manner.

High availability –

Auto replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.

Disadvantages of NoSQL:

NoSQL has the following

disadvantages. Narrow focus –

NoSQL databases have very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.

Open-source –

NoSQL is open-source database. There is no reliable standard for NoSQL yet. In other words two database systems are likely to be unequal.

Management challenge –

The purpose of big data tools is to make management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.

GUI is not available –

GUI mode tools to access the database is not flexibly available in the market.

Backup –

Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.

Large document size –

Some database systems like MongoDB and CouchDB store data in JSON format. Which means that documents are quite large (BigData, network bandwidth, speed), and having descriptive key names actually hurts, since they increase the document size.

Types of NoSQL database:

Types of NoSQL databases and the name of the databases system that falls in that category are: MongoDB falls in the category of NoSQL document based database.

Key value store: Memcached, Redis, Coherence

Tabular: Hbase, Big Table, Accumulo

Document based: MongoDB, CouchDB, Cloudant

When should NoSQL be used:

When huge amount of data need to be stored and retrieved .The relationship between the data you store is not that important. The data changing over time and is not structured. Support of Constraints and Joins is not required at database level. The data is growing continuously and you need to scale the database regular to handle the data.

NoSQL Data Architecture Patterns

Architecture Pattern is a logical way of categorising data that will be stored on the Database. NoSQL is a type of database which helps to perform operations on big data and store it in a valid format. It is widely used because of its flexibilty and wide variety of services.

Architecture Patterns of NoSQL:

The data is stored in NoSQL in any of the following four data architecture patterns.

1. Key-Value Store Database
2. Column Store Database
3. Document Database
4. Graph Database

These are explained as following below.

1. Key-Value Store Database:

This model is one of the most basic models of NoSQL databases. As the name suggests, the data is stored in form of Key-Value Pairs. The key is usually a sequence of strings, integers or characters but can also be a more advanced data type. The value is typically linked or co-related to the key. The key-value pair storage databases generally store data as a hash table where each key is unique. The value can be of any type (JSON, BLOB(Binary Large Object), strings, etc). This type of pattern is usually used in shopping websites or e-commerce applications.

Advantages:

Can handle large amounts of data and heavy

load. Easy retrieval of data by keys.

Limitations:

Complex queries may attempt to involve multiple key-value pairs which may delay performance.

Data can be involving many-to-many relationships which may collide.

Examples:

DynamoDB

Berkeley DB

2. Column Store Database:

Rather than storing data in relational tuples, the data is stored in individual cells which are further grouped into columns. Column-oriented databases work only on columns. They store large amounts of data into columns together. Format and titles of the columns can diverge from one row to other. Every column is treated separately. But still, each individual column may contain multiple other columns like traditional databases.Basically, columns are mode of storage in this type.

Advantages:

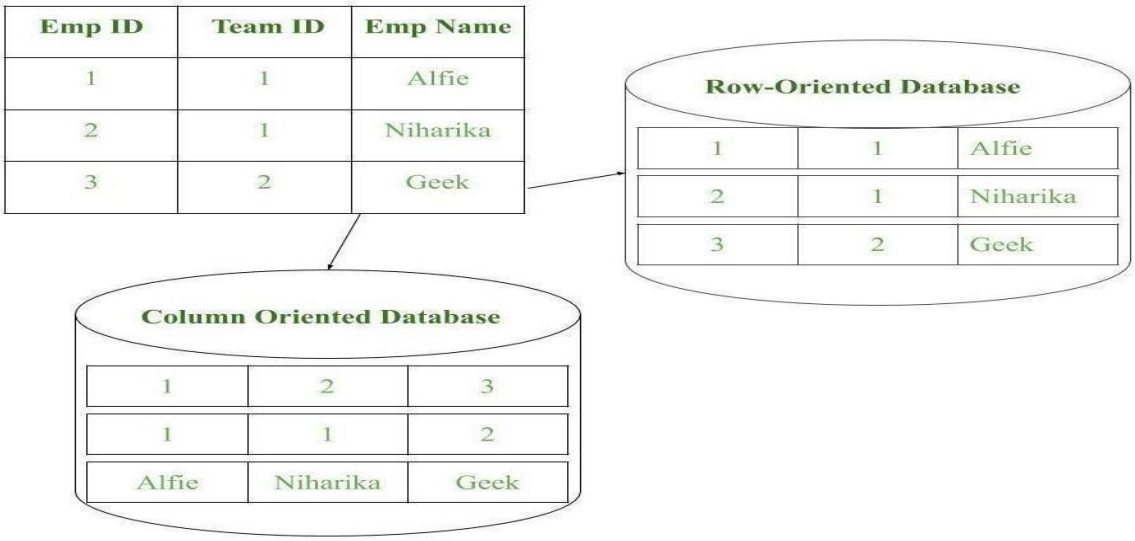
Data is readily available Queries like SUM, AVERAGE, COUNT can be easily performed on columns.

Examples:

HBase

Bigtable by Google

Cassandra



3. Document Database:

The document database fetches and accumulates data in forms of key-value pairs but here, the values are called as Documents. Document can be stated as a complex data structure. Document here can be a form of text, arrays, strings, JSON, XML or any such format. The use of nested documents is also very common. It is very affective as most of the data created is usually in form of JSONs and is unstructured.

Advantages:

This type of format is very useful and apt for semi-structured data.

Storage retrieval and managing of documents is easy.

Limitations:

Handling multiple documents is challenging

Aggregation operations may not work

accurately. Examples:

MongoDB

CouchDB

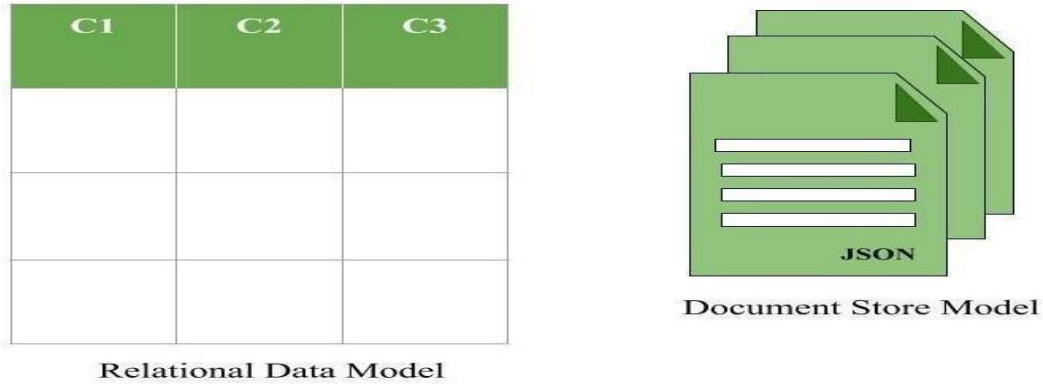


Figure – Document Store Model in form of JSON documents

4. Graph Databases:

Clearly, this architecture pattern deals with storage and management of data in graphs. Graphs are basically structures that depict connections between two or more objects in some data. The objects or entities are called as nodes and are joined together by relationships called Edges. Each edge has a unique identifier. Each node serves as a point of contact for the graph..

Advantages:

Fastest traversal because of connections.

Spatial data can be easily handled.

Limitations:

Wrong connections may lead to infinite loops.

Examples:

Neo4J

FlockDB(Used by Twitter)

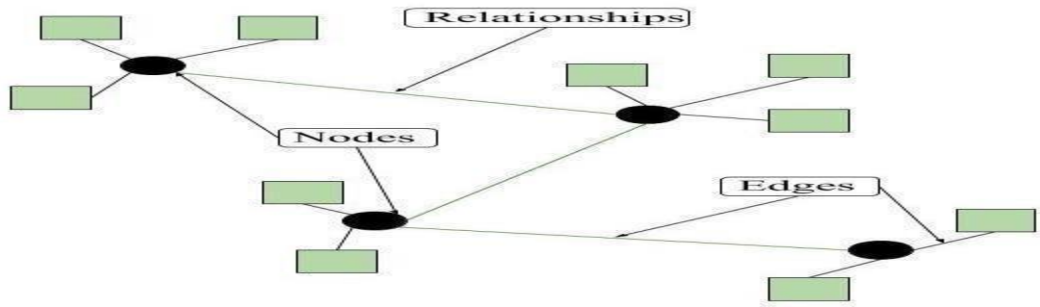
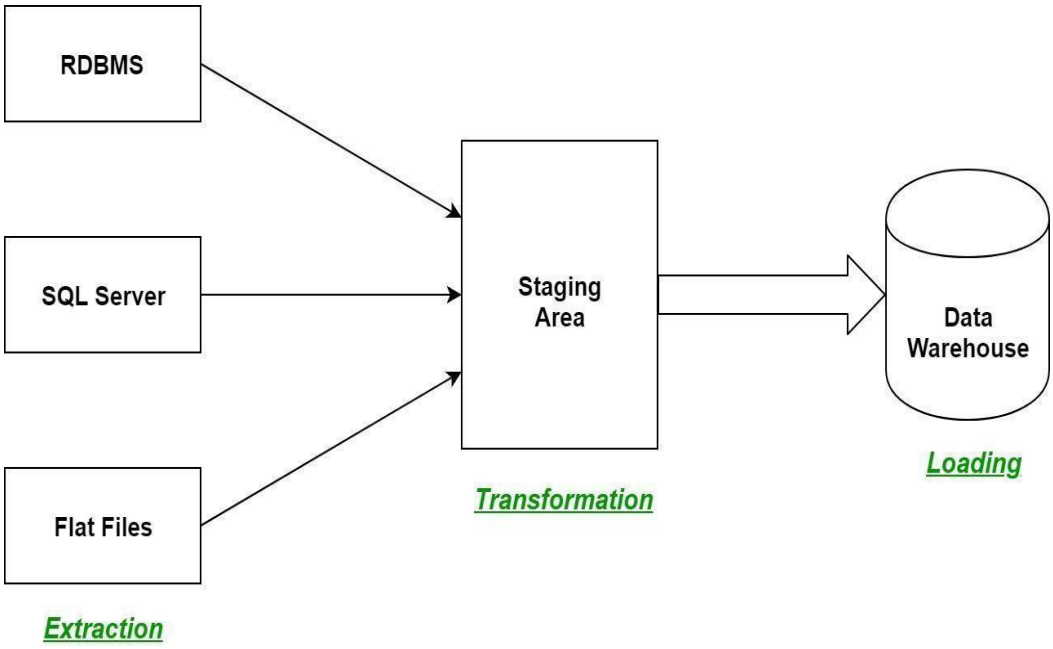


Figure – Graph model format of NoSQL Databases

TEXTUAL ETL PROCESSING

ETL is a process in Data Warehousing and it stands for **Extract**, **Transform** and **Load**. It is a process in which an ETL tool extracts the data from various data source systems, transforms it in the staging area, and then finally, loads it into the Data Warehouse system.



Extraction:

The first step of the ETL process is extraction. In this step, data from various source systems is extracted which can be in various formats like relational databases, No SQL, XML, and flat files into the staging area. It is important to extract the data from various source systems and store it into the staging area first and not directly into the data warehouse because the extracted data is in various formats and can be corrupted also. Hence loading it directly into the data warehouse may damage it and rollback will be much more difficult. Therefore, this is one of the most important steps of ETL process.

Transformation:

The second step of the ETL process is transformation. In this step, a set of rules or functions are applied on the extracted data to convert it into a single standard format. It may involve following processes/tasks:

Filtering – loading only certain attributes into the data warehouse.

Cleaning – filling up the NULL values with some default values, mapping U.S.A, United States, and America into USA, etc.

Joining – joining multiple attributes into one.

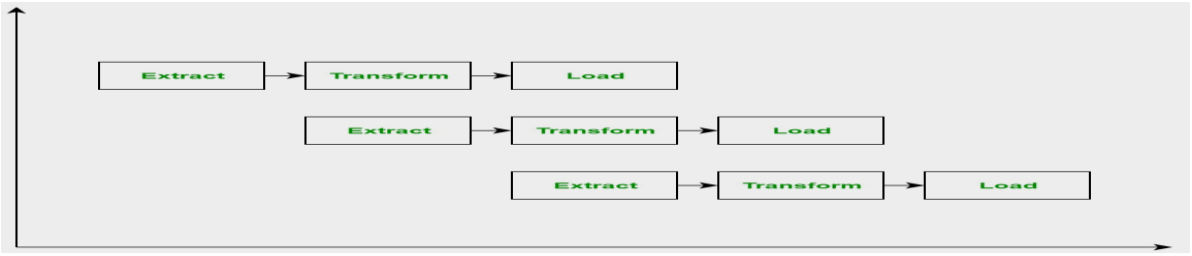
Splitting – splitting a single attribute into multiple attributes.

Sorting – sorting tuples on the basis of some attribute (generally key-attribute).

Loading:

The third and final step of the ETL process is loading. In this step, the transformed data is finally loaded into the data warehouse. Sometimes the data is updated by loading into the data warehouse very frequently and sometimes it is done after longer but regular intervals. The rate and period of loading solely depends on the requirements and varies from system to system.

ETL process can also use the pipelining concept i.e. as soon as some data is extracted, it can transformed and during that period some new data can be extracted. And while the transformed data is being loaded into the data warehouse, the already extracted data can be transformed. The block diagram of the pipelining of ETL process is shown below:



ETL Tools: Most commonly used ETL tools are Sybase, Oracle Warehouse builder, CloverETL, and MarkLogic.

Textual ETL is the **process of reading text and producing a relational data base suitable for analytical processing**. The text can come from any electronic source and the results can go into any relational data base. ... Email typically contains spam and personal email which does not belong in a corporate data base.

UNIT IV

Big Data analytics:

Data analytics life cycle, Data cleaning , Data transformation, Comparing reporting and analysis, Types of analysis, Analytical approaches, Data analytics using R, Exploring basic features of R, Exploring R GUI, Reading data sets, Manipulating and processing data in R, Functions and packages in R, Performing graphical analysis.

Types of analysis

UNIT 4 Big Data Analytics

Analysis of data is a vital part of running a successful business. There are four types of data analysis that are in use across all industries. While we separate these into categories, they are all linked together and build upon each other. As you begin moving from the simplest type of analytics to more complexes, the degree of difficulty and resources required increases. At the same time, the level of added insight and value also increases.

Four Types of Data Analysis

The four types of data analysis are:

- Descriptive Analysis
- Diagnostic Analysis
- Predictive Analysis
- Prescriptive Analysis

Descriptive Analysis

The first type of data analysis is descriptive analysis. It is at the foundation of all data insight. It is the simplest and most common use of data in business today. Descriptive analysis answers the "what happened" by summarizing past data, usually in the form of dashboards.

The biggest use of descriptive analysis in business is to track Key Performance Indicators (KPIs). KPIs describe how a business is performing based on chosen benchmarks.

Business applications of descriptive analysis include:

- KPI dashboards
- Monthly revenue reports
- Sales leads overview

Diagnostic Analysis

Diagnostic analysis takes the insights found from descriptive analytics and drills down to find the causes of those outcomes. Organizations make use of this type of analytics as it creates more connections between data and identifies patterns of behavior.

A critical aspect of diagnostic analysis is creating detailed information. When new problems arise, it is possible you have already collected certain data pertaining to the issue. By already

having the data at your disposal, it ends having to repeat work and makes all problems interconnected.

Business applications of diagnostic analysis include:

- A freight company investigating the cause of slow shipments in a certain region
- A SaaS company drilling down to determine which marketing activities increased trials

Predictive Analysis

This type of analysis is another step up from the descriptive and diagnostic analyses. Predictive analysis uses the data we have summarized to make logical predictions of the outcomes of events. This analysis relies on statistical modeling, which requires added technology and manpower to forecast. It is also important to understand that forecasting is only an estimate; the accuracy of predictions relies on quality and detailed data.

Business applications of predictive analysis include:

- Risk Assessment
- Sales Forecasting
- Using customer segmentation to determine which leads have the best chance of converting
- Predictive analytics in customer success teams

Data Analytics Lifecycle Phases

There's no defined structure of the phases in the life cycle of Data Analytics; thus, there may not be uniformity in these steps. There can be some data professionals that follow additional steps, while there may be some who skip some stages altogether or work on different phases simultaneously.

Phase 1: Data Discovery and Formation

Phase 2: Data Preparation and

Processing Phase 2: Data Preparation

and Processing Phase 4: Model Building

Phase 5: Result Communication and Publication

Phase 6: Measuring of Effectiveness

Data Analytics Lifecycle Example

Consider an example of a retail store chain that wants to optimize its products' prices to boost its revenue. The store chain has thousands of products over hundreds of outlets, making it a highly complex scenario. Once you identify the store chain's objective, you find the data you need, prepare it, and go through the Data Analytics lifecycle process. Observe different types of customers, such as ordinary customers and customers like contractors who buy in bulk. According to you, treating various types of customers differently can give us the solution.

In this case, we need to get the definition, find data, and conduct hypothesis testing to check whether various customer types impact the model results and get the right output. Once we are convinced with the model results, we can deploy the model, and integrate it into the business, and we are all set to deploy the prices you think are the most optimal across the outlets of the store.

Data analytics using R

R

R is a programming language that provides more flexibility than Excel. R is not bound by a spreadsheet, where the data need to be entered in cells. For more complex analyses, Excel's spreadsheet format is too restrictive. And R is *freely* available on-line with new content being uploaded regularly.

RStudio

RStudio is an interface that provides you with a greater ability to conduct your analyses in R. You can think of RStudio as a overlay on the software R to allow you to visually group together in one interface the input window, the output window, the objects in your workspace, and plots.

Benefits of R Analytics

Business analytics in R allows users to analyze business data more efficiently. The following are some of the main benefits realized by companies employing R in their analytics programs:

Democratizing Analytics across the Organization: R can help democratize analytics by enabling business users with interactive data visualization and reporting tools. R can be used for data science by non data scientists so that business users and citizen data scientists can make better business decisions. R analytics can also reduce time spent on data preparation and data wrangling, allowing data scientists to focus on more complex data science initiatives.

Providing Deeper, More Accurate Insights: R can help create powerful models to analyze large amounts of data. With more precise data collection and storage through R analytics, companies can deliver more valuable insights to users. Analytics and statistical engines using R provide deeper, more accurate insights for the business. R can be used to develop very specific, in-depth analyses.

Leveraging Big Data: R can handle big datasets and is arguably as easy if not easier for most analysts to use as any of the other analytics tools available today.

Creating Interactive Data Visualizations: R is also helpful for data visualization and data exploration because it supports the creation of graphs and diagrams. It includes the ability to create interactive visualizations and 3D charts and graphs that are helpful for communicating with business users.

While R programming was originally designed for statisticians, it can be implemented for a variety of uses including predictive analytics, data modeling, and data mining. Businesses can implement R to create custom models for data collection, clustering, and analytics. R analytics can provide a valuable way to quickly develop models targeted at understanding specific areas of the business and delivering tailored insights on day-to-day needs.

R analytics can be used for the following purposes:

- Statistical testing
- Prescriptive analytics
- Predictive analytics
- Time-series analysis
- What-if analysis
- Regression models
- Data exploration
- Forecasting
- Text mining
- Data mining
- Visual analytics
- Web analytics
- Social media analytics
- Sentiment analysis

Exploring basic Features of R:

The R programming language is versatile and can be used for a software development environment for statistical analysis or graphics representation and reporting purposes.The below mentioned are the significant features of the R language:

- R is a simple and effective programming language that has been well-developed, as well as R is data analysis software.
- R has a large, consistent, and incorporated set of tools used for data analysis.
- R contains a suite of operators for different types of calculations on arrays, lists, and vectors.
- R provides highly extensible graphical techniques.
- R graphical techniques for data analysis output either directly display to the computer, or can be print on paper.
- R has an effective data handling and storage facility.
- R is a vibrant online community.
- R is free, open-source, robust, and highly extensible.
- R supports matrix arithmetic
- R language can also be used with several other scripting languages such as python, perl, ruby, F#, and Julia.

Exploring R GUI

R GUI is the standard GUI platform for working in R. The R Console Window forms an essential part of the R GUI. In this window, we input various instructions, scripts and several other important operations. This console window has several tools embedded in it to facilitate ease of operations. This console appears whenever we access the R GUI.

In the main panel of R GUI, go to the '**File**' menu and select the '**New Script**' option. This will create a new script in R. In order to quit the active R session, you can type the following code after the R prompt '>' as follows:

```
>q()
```

Manipulating and processing data in R

Data structures provide the way to represent data in data analytics. We can manipulate data in R for analysis and visualization. One of the most important aspects of computing with data in R is its ability to manipulate data and enable its subsequent analysis and visualization. Let us see few basic data structures in R:

a. VectorsinR

These are ordered container of primitive elements and are used for 1-dimensional data.

Types – integer, numeric, logical, character, complex

b. Matricesin R

These are Rectangular collections of elements and are useful when all data is of a single class that is numeric or characters.

Dimensions – two, three, etc.

c. Listsin R

These are ordered container for arbitrary elements and are used for higher dimension data, like customer data information of an organization. When data cannot be represented as an array or a data frame, list is the best choice. This is so because lists can contain all kinds of other objects, including other lists or data frames, and in that sense, they are very flexible.

d. Data frames

These are two-dimensional containers for records and variables and are used for representing data from spreadsheets etc. It is similar to a single table in the database.

Creating Subsets of Data in R

Data size is increasing exponentially and doing analysis on complete data is very time-consuming. So data is divided into small sized samples and analysis of samples is done. The process of creating samples is called subsetting.

Different methods of subsetting in R are:

a.$

The dollar sign operator selects a single element of data. When you use this operator with a data frame, the result is always a vector.

b.[[

Similar to $ in R, the double square brackets operator in R also returns a single element, but it offers the flexibility of referring to the elements by position rather than by name. It can be used for data frames and lists.

c.[

The single square bracket operator in R returns multiple elements of data. The index within the square brackets can be a numeric vector, a logical vector, or a character vector.

For example: To retrieve 5 rows and all columns of already built in data set iris, below command is used:

1> iris[1:5,]

Creating subsets in vectors

To create subsets of data stored in a vector, the subset( ) function or the [ ] brackets is used.

Example: using the subset( ) function and the [ ] brackets.

```
 # A sample vector
V <-
c(1,5,6,3,2,4,2)
Subset(v,v,4) # using subset function
V[v,4] #using square brackets
# another vector
T ,- c(“one”, “one”, “two”, “three”, “four”,
“two”) #remove “one” entries
Subset(t,
t!=”one)
T[t!=”one”]
```

Creating subsets in Data

Frames Example 1:

```
 # A sample data frame
Data <- read.table(header=T, text=’
Subject    class marks

1              1        99
```

2	2	84
3	1	89

4 2 79

‘)

Subset(data, subject < 3)

Data {data$subject <3,]

Example 2: creating subsets of different logical conditions

Logical AND of two conditions

Subset(data, subject < 3 & class==2)

Data[data$subject <3 & data$calss==2,

] #logical OR of two conditions

Subset(data, subject <3 | class==2)

Data[data$subject < 3 | data$class==2,]

Merging datasets in R

Sometimes, similar datasets obtained from different sources need to be merged together for further processing. R provides the following functions to combine different sets of data :

Merge() Function in R - The merge() function is used to combine data frames on the basis of columns and rows.

Thecbind() function– It is usedto addthecolumnsofdatasetshaving an equalset and identicalorderofrows.

The rbind( ) function– it is usedto addrowsindatasets havinganequalnumberofcolumns. The merge()function takes a large number of arguments, as follows:

- x:A data frame
- y: A data frame
- by, by.x, by.y: Names of the columns common to both x and y. By default, it uses columns with common names between the two data frames.
- all, all.x, all.y: Logical values that specify the type of merge. The default value is all = FALSE…

The merge() function allows four ways of combining data:

a.NaturaljoininR

To keep only rows that match from the data frames, specify the argument all=FALSE

b.Fullouterjoin inR

To keep all rows from both data frames, specify all=TRUE

c.Leftouterjoin inR

To include all the rows of your data frame x and only those from y that match, specify all.x=TRUE

d.RightouterjoininR

To include all the rows of your data frame y and only those from x that match, specify all.y=TRUE

Using the cbind( ) Function:

cbind() function in R Language is used to combine specified Vector, Matrix or Data Frame by columns.

Syntax: cbind(x1, x2, …, deparse.level = 1)

Parameters:
x1, x2: vector, matrix, data frames
deparse.level: This value determines how the column names generated. The default value of deparse.level is 1.

Example 1:

```
# R program to illustrate cbind()

function # Initializing two vectors

x <- 2:7

y <- c(2, 5)

  # Calling cbind() function

cbind(x, y)
```

Output:

```
     x y
[1, ] 2 2

[2, ] 3 5

[3, ] 4 2

[4, ] 5 5
```

[5,] 6 2

[6,] 7 5

Using the rbind( ) Function:

rbind(): The rbind or the row bind function is used to bind or combine the multiple group of rows together.

```
rbind(my_data, new_row)
```

Example 1: rbind Vector to Data Frame

The easiest way of using rbind in R is the combination of a vector and a data frame. First, let's create some example data frame…

```
x1 <- c(7, 4, 4,                # Column 1 of data
9)                              frame
x2 <- c(5, 2, 8,                # Column 2 of data
9)                              frame
x3 <- c(1, 2, 3,                # Column 3 of data
4)                              frame
data_1 <- data.frame(x1, x2, x3)      # Create example data
frame
```

and an example vector:

```
vector_1 <- c(9, 8, 7)          # Create example vector
```

Now, let's rbind this vector to the data frame:

```
rbind(data_1, vector_1)       # rbind vector to data frame
```

Output:

```
## x1 x2
x3 ## 1
7 5 1
```

```
## 2  4  2  2
## 3  4  8  3
## 4  9  9  4
## 5  9  8  7
```

Sorting data:

The Sort( ) function is used to sort the values contained in vector.

Example: vec1,-c(23,45,10,34,89,20,67,99)

sorting of a

vector Sort(vec1)

#reverse sorting

Sort(vec1, descreasing=TRUE)

Ordering data:

the order( ) function is used to organize/arrange values or columns in a dataset.

Example:

#make a data frame

sampleDataFrame <- data. Frame

(id=1:5,weight=c(25,37,14,62,55), size=c(“small”, “large”,

“medium”, “large”, “medium”))

```
sampleDataFrame[ order(sampleDataFrame$weight),

] #sort by size, then by weight

sampleDataFrame[ order(sampleDataFrame$size, sampleDataFrame$weight), ]

#sort by all columns in the data frame, from left to right

sampleDataFrame[ do.call(order, as.list(sampleDataFrame)), ]
```

Transposing Data

t( ) function is used to transpose a matrix or a data frame. This function transposes rows into columns and columns into rows.

Example:

```
sampleDataFrame

t(sampleDataFrame

)
```

Converting data to wide or long formats

R provides the following functions of the reshape package to convert data into wide or long formats:

- use the melt ( ) function to convert wide data into the long format
- use the dcast ( ) function to convert long data into the wide format

melt( )

Syntax:

melt(data, na.rm = FALSE, value.name = “value”)

Cast ( )

Syntax:

cast(data, formula, fun.aggregate)

Managing Data in R using Matrices

A matrix is a collection of data elements arranged in a two-dimensional rectangular layout. In R, the elements that make up a matrix must be of a consistant mode (i.e. all elements must be numeric, or character, etc.). Therefore, a matrix can be thought of as an atomic vector with a dimension attribute. Furthermore, all rows of a matrix must be of same length.

Creating Matrices

Matrices are constructed column-wise, so entries can be thought of starting in the "upper left" corner and running down the columns. We can create a matrix using the matrix() function and specifying the values to fill in the matrix and the number of rows and columns to make the matrix.

```
# numeric matrix
m1 <- matrix(1:6, nrow = 2, ncol = 3)

m1
output
:
##      [,1] [,2] [,3]
## [1,] 1    3    5
## [2,] 2    4    6
```

The underlying structure of this matrix is simply an integer vector with an added 2x3 dimension attribute.

Matrices can also contain character values. Whether a matrix contains data that are of numeric or character type, all the elements must be of the same class.

```
# a character matrix
m2 <- matrix(letters[1:6], nrow = 2, ncol = 3)


m2
##      [,1] [,2] [,3]
```

```
## [1,] "a" "c" "e"
## [2,] "b" "d" "f"
```

Matrices can also be created using the column-bind cbind() and row-bind rbind() functions. However, keep in mind that the vectors that are being binded must be of equal length and mode.

```
v1 <- 1:
            v2)
           v2
           5

## [2,   2  6
## [3,   3  7
## [4,   4  8

rbind(v1,
##    [,1] [,2] [      ,3]
## v1    1    2      [,4]
## v2    5    6        3

                       4
# bind several         vectors
v3 <- 9:               together
                         8

cbind(v1,  v2,      v
##    v1   v2  v     3
## [1,]  1  5 9      )
## [2,]  2  6 1     3
## [3,]  3  7 1
## [4,]  4  8 1
```

Adding on to Matrices

We can leverage the cbind() and rbind() functions for adding onto matrices as well. Again, its important to keep in mind that the vectors that are being binded must be of equal length and mode to the pre-existing matrix.

```
m1 <- cbind(v1,
v2)
## m1
## [1,
## [2,  2  6
## [3,   3  7
## [4,   4  8
# add a new
column
cbind(m1, v3)
##       v  v2 v3
       1
## [1,]  1   5 9
## [2,]  2   6 10
## [3,]  3   7 11
## [4,]  4   8 12
# o
row rbind(m1,
c(4.1, 8.1)) ## v1
v2
## [1,] 1.0 5.0
## [2,] 2.0 6.0
## [3,] 3.0 7.0
## [4,] 4.0 8.0
## [5,] 4.1 8.1
```

Sub setting Matrices

To subset matrices we use the [operator; however, since matrices have 2 dimensions we need to incorporate subsetting arguments for both row and column dimensions. A generic form of matrix subsetting looks like: matrix[rows, columns]. We can illustrate with matrix m2:

```
m2
##       col_1 col_2 col_3
## row_1     1     5     9
## row_2     2     6    10
## row_3     3     7    11
## row_4     4     8    12
```

By using different values in the rows and columns argument of m2[rows, columns], we can subset m2 in multiple ways.

```
# subset for rows 1 and 2 but keep all
columns m2[1:2, ]
##       col_1 col_2 col_3
## row_1     1     5     9
## row_2     2     6    10

# subset for columns 1 and 3 but keep all
rows m2[ , c(1, 3)]
##       col_1 col_3
## row_1     1     9
## row_2     2    10
## row_3     3    11
## row_4     4    12

# subset for both rows and columns
m2[1:2, c(1, 3)]
##       col_1
col_3 ## row_1  1
        9
```

```
## row_2     2    10
```

```
# use a vector t subset
v <- c(1, 2, 4)
m2[v, c(1, 3)]
##       col_ col_3
## row_1     1     9
## row_2     2    10
## row_4     4    12

# use names to subset
m2[c("row_1", "row_3"),
##       col_ col_2 col_3
## row_1     1     5     9
## row_3     3     7    11
```

]

Note that subsetting matrices with the [operator will simplify the results to the lowest possible dimension. To avoid this you can introduce the drop = FALSE argument:

```
# simplifying results in a named vector
m2[, 2]
## row_1 row_2 row_3 row_4
##     5     6     7
8

# preserving results in a 4x1 matrix
m2[, 2, drop = FALSE]
##       col_2
## row_1
5
## row_2     6
## row_3     7
## row_4     8
```

Managing Data in R using Data Frames:

A data frame is a table or a two-dimensional array-like structure in which each column contains

values of one variable and each row contains one set of values from each column.

Following are the characteristics of a data frame.

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factor or character type.
- Each column should contain same number of data items.

Create Data Frame

```
# We assign the data frame to variable df_base

df_base <- data.frame(name =
c("Jane","Sri","Eliza","Joe"),
                      occupation =
```

Note that you simply use the data.frame function, provide names for your columns, and populate the contents of the columns (using the c() vector functionality).

Output:

name <fctr>	occupation <fctr>
Jane	engineer
Sri	designer
Eliza	architect
Joe	engineer

Note that a data frame can hold a number of data types (i.e., the columns can be characters, integers, dates, factors, etc). Above you'll notice that the data types are displayed below the column name; in this case, our two columns were coded as factors.

Accessing Data in the Data Frame

Selecting a Column by Index

If you want to access a particular column by using the column number. Perhaps you want to understand the occupations in our dataset. Here's how that access works:

```
1df_base[,2]
```

r

```
[1] engineer  designer  architect engineer
Levels: architect designer engineer
```

We not only see the values of each row in the second column printed but also the corresponding *levels*. See here for more on what levels are. The syntax is the same when selecting a row from a Tibble, except the levels aren't included because columns with characters aren't automatically coded as factors and only factors have levels (don't get hung-up if you don't understand levels for now). Note that the tibble column prints a little nicer (and is a chr or character, but not in a jokey way).

```
1df_tidy[,2]
```

r

occupation
<chr>
engineer
designer
architect
engineer

Note that in R, when locating a cell, [1,2] refers to the first row and second column, so that [,2] grabs the entire second column.

To actually do something more interesting with this, and count the number of unique jobs, you use the same syntax inside a function:

```
1unique(df_tidy[,2])
```

r

occupation
<chr>
engineer
designer
architect

Selecting a Column by Name

Note that we keep the , syntax since we want the entire column, and select the column by name in quotes.

```
ldf_base[,"occupation"]
```

r

```
[1] engineer  designer  architect engineer
Levels: architect designer engineer
```

This is the same for a tibble (this is the last time we'll make the comparison). Again, notice the levels are gone because the tidyverse defaults to characters instead of factors.

```
ldf_tidy[,"occupation"]
```

r

occupation
<chr>
engineer
designer
architect
engineer

Selecting a Row by Index

Computations usually happen on *columns* or parts of columns; when you're looking at an entire row or a few rows it's usually more for inspection and sanity-checking. Now let's say you ran into an unexpected result of a computation and wanted to examine an entire row in a dataset.

```
ldf_tidy[1,]
```

r

name	occupation
<chr>	<chr>
Jane	engineer

And here's the syntax to grab multiple rows. Note that you span *inclusively* from the first to last row of interest.

```
ldf_tidy[1:2,]
```

r

name	occupation
<chr>	<chr>
Jane	engineer
Sri	designer

name	occupation
<chr>	<chr>
Jane	engineer
Joe	engineer

Packages in R

A package is a collection of R functions, data, and compiled code in a well-defined format. Packages are being stored in the directory called the library. R comes with a standard set of packages. With the help of the *search()* command, you can find all the list of available packages that are installed in your system.

Others are available for download and installation. Once installed, you need to load them into the session to use.

example of search() command as follows:

search()

When the search() command is executed, you can overview the packages that are loaded and are ready for use. You will see the graphics package that carries out routines to create graphs.

There are many packages that are being installed but not loaded by themselves.

For example – Splines package, that contains routines for smoothing curves, is being installed. But this splines package is not loaded by itself.

To see what packages are available, you need to type the following command:

installed.packages()

Installing R Packages for Windows

In Windows, you get the package menu and install option which is very easy.

After selecting a local mirror site, a list of available binary packages is being shown. You can choose the ones you need. Once you have selected the packages you need, you need to click the **OK** button to download and install them into R.

If you download the package files from the internet(as .zip), you need to use the install package(s) in the packages menu. It allows you to select the files you need and again packages are unzipped and installed into R.

How to Install R Packages for Linux

To install R packages on the Linux system, you need to perform the below steps:

- Download the required packages as compressed files from the link: Available packages by name
- Run the following command to install packages:

R CMD INSTALL [options] [l-lib] pkgs

- Use the following command to load the installed package:

library(package)

Installing by the Name of Package

In Linux, you can install the package if you know the name of a package.

Use the following command to install any package:

install.packages('ade4')

R Packages List

The below table specifies the best packages in R with their usage:

Package Name	Use
ade4	Used for analysis in ecological science
amap	Used for multidimensional analysis
ANN	Used for building and analyzing Artificial Neural Networks (ANN)
BayesLogit	Used for logistic regression analysis
C50	Used for developing decision trees and rule-based models
lattice	Used for creating lattice graphics for panel plots or trellis graphs
MASS	Used for modern applied statics using S-PLUS
mgcv	Used for building generalized additive models

How to Use Packages in R

We need to load the package in R after installing them to make them usable.

To load the R language Package, you can use the *library()* command, as follows:

library(package)

In R, you can unload a package by using *detach()* command, as

follows: detach(package:name)

Working with Functions

Movingfrom Scripts to RFunction:

R function provides two major advantages over the script:

- Functions can work with any input. You can provide diverse input data to the functions.
- The output of the function is an object that allows you to work with the result.

How to Create a Script in R?

R supports several editors. So the script can be created in any of the editors like Notepad, MS Word or Word Pad and can be saved with R extension in the current working directory.

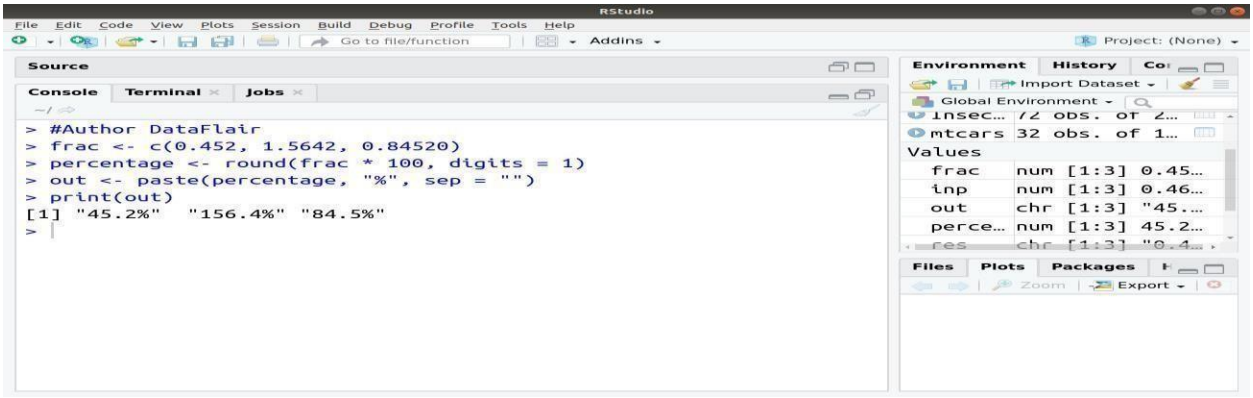
Now to read the file in R, source function can be used.
For example, if we want to read the sample. R script in R, we need to provide below command:
source("sample.R")
In order to create a script, first, open a script file in the editor mode and type the required code.

We will create a script that takes in input in the form of fractions and converts it into a percentage by further rounding it to one decimal digit.

```
> #Author DataFlair
> frac <- c(0.452, 1.5642, 0.84520)
> percentage <- round(frac * 100, digits = 1)
> out <- paste(percentage, "%", sep = "")
> print(out)
```

That is, you need to give values that you want to convert to a percentage as input and then convert it into percentage and round off to required places. Then put the % sign and display the answer.

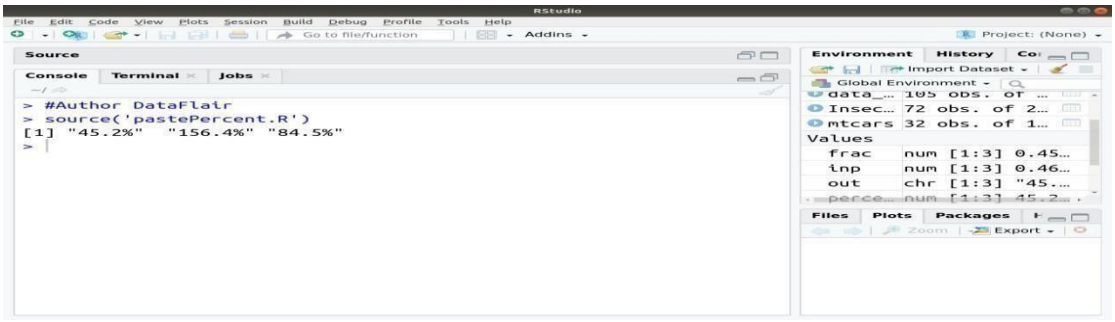
Save the above script as script file with any name for example pastePercent.R.



Now you can call this script on the console with the help of source command which we have already seen.

source('pastePercent.R')

Output:



This is how a script is written and executed in R.

Transforming the Script into R Function

Define a function with a name so that it becomes easier to call an R function and pass arguments to it as input**.**

The R function should be followed by parentheses that act as a front gate for your function and between the parentheses, arguments for the function are provided.

Use the return() statement that acts as a back gate of your function.

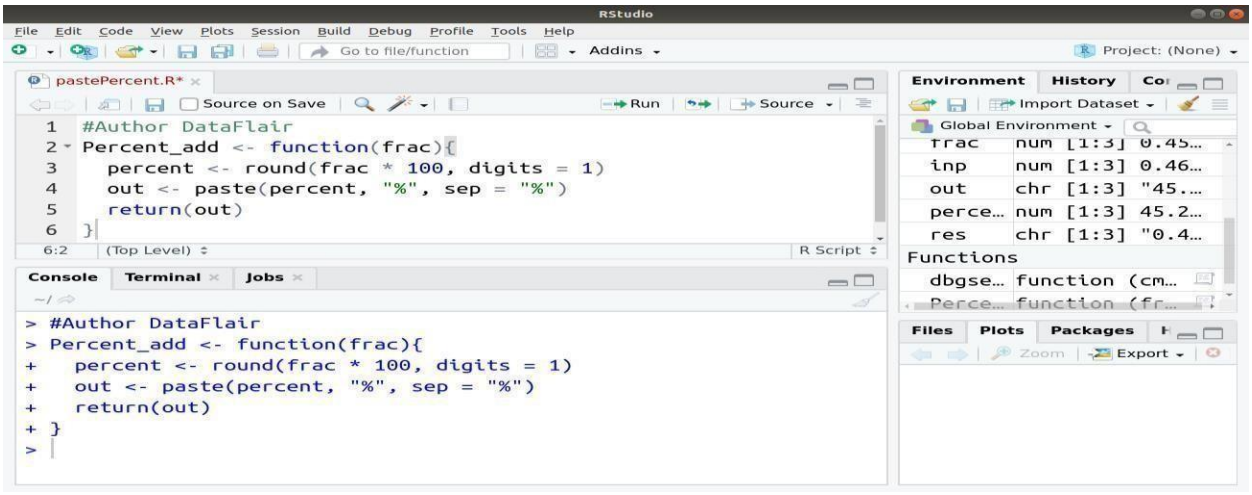
The return() statement provides the final result of the function that is returned to your workspace.

We will see this with an example below.

Let us now see how we can convert the script that we had written earlier to convert values into percentage and round off into an R function.

```
Percent_add <- function(frac){
percent <- round(frac * 100, digits = 1)
out <- paste(percent, "%", sep = "%")
return(out)
}
```

Output:



The keyword **function** defines the starting of function. The **parentheses** after the function form the front gate, or argument list of the function. Between the parentheses are the arguments to the function. In this case, there is only one argument.

The **return** statement defines the end of the function and returns the result. The object put between the parentheses is returned from inside the function to the workspace. Only one object can be placed between the parentheses.

The braces, {} are the walls of the function. Everything between the braces is part of the assembly line or the body of the function. This is how functions are created in R.

Using R Function

After transforming the script into an R function, you need to save it and you can use the function in R again if required.

As R does not let you know by itself that it loaded the function but it is present in the workspace, if you want you can check it by using **ls()** command.

Now as we know what all functions in R are present in the memory and we can use it when required.

For example, if you want to create percentage from values again, you can use add percent function for the same as below:

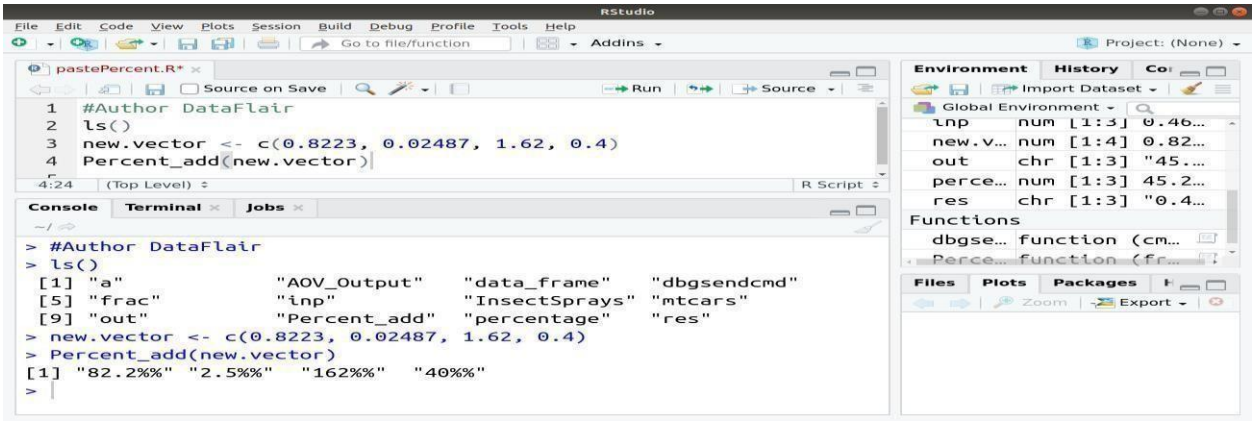
#Author

DataFlair ls()

new.vector <- c(0.8223, 0.02487, 1.62, 0.4)

Percent_add(**new**.vector)

Output:



Using the Function Objects in R

In R, a function is also an object and you can manipulate it as you do for other objects.

You can assign a function to the new object using below command:

percent_paste <- Percent_add

Now **percent paste** is a function as well that does exactly the same as Percent_add. Note that, you do not add it after parentheses Percent_add in this case. If you add the parentheses, you call the function and put the result of that call in percent_paste. If you do not add the parentheses, you refer to the function object itself without calling it.

percent_paste

Percent_add <- **function**(frac){

percentage <- round(frac * 100, digits =

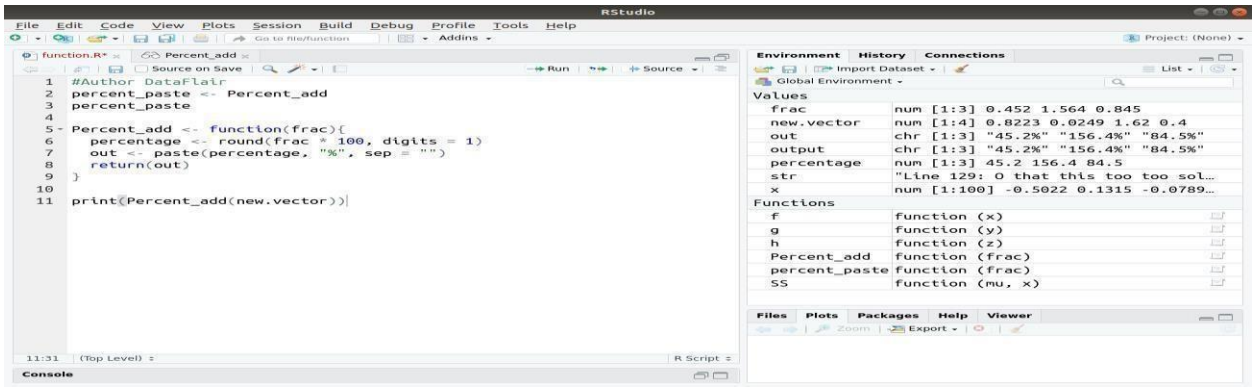
1) out <- paste(percentage, "%", sep = "")

return(out)

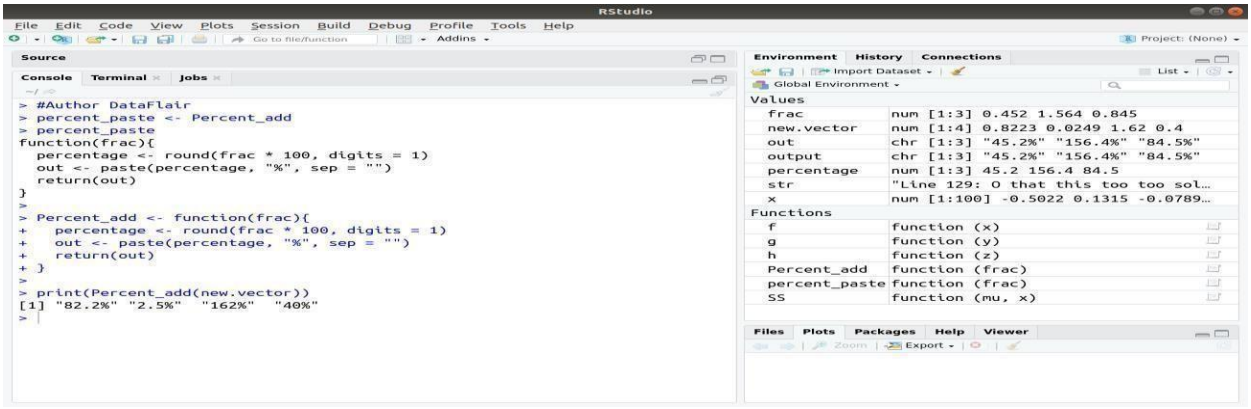
}

print(Percent_add(**new**.vector))

Code Display:



The output of the above code is as follows:



Reducing the Number of Lines in R

There are basically two ways of doing it:

- Returning values by default
- Dropping {}

Let us see the above ways in detail below:

1. Returning Values by Default in R

Till now, in all the above code, we have written *return()* function to return output. But in R, this can be skipped as by default, R returns the value of the last line of code in the R function body.

Now, the above code will be:

#Author DataFlair

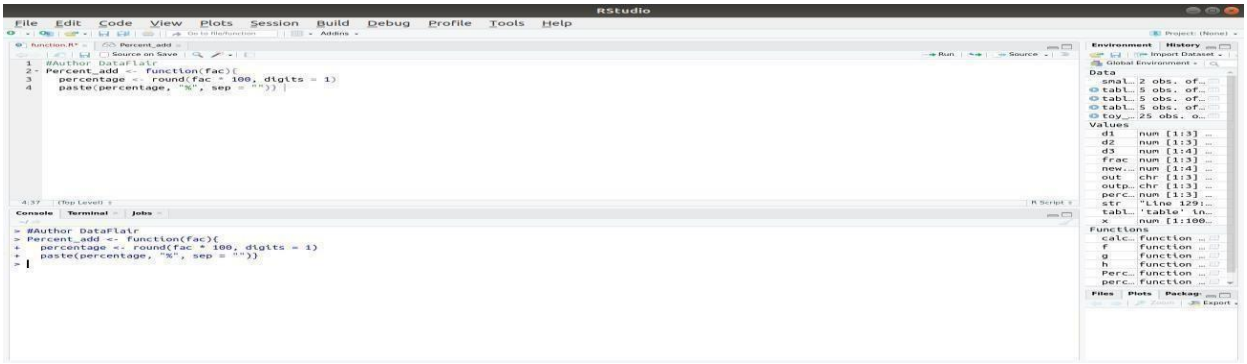
Percent_add <-

function(fac){

```
percentage <- round(fac * 100, digits = 1)
```

paste(percentage, "%", sep = "")}

Output:



You need return if you want to exit the function before the end of the code in the body.

For example, you could add a line to the Percent_add function that checks whether fac is numeric, and if not, returns NULL, as shown in the following table:

#Author DataFlair

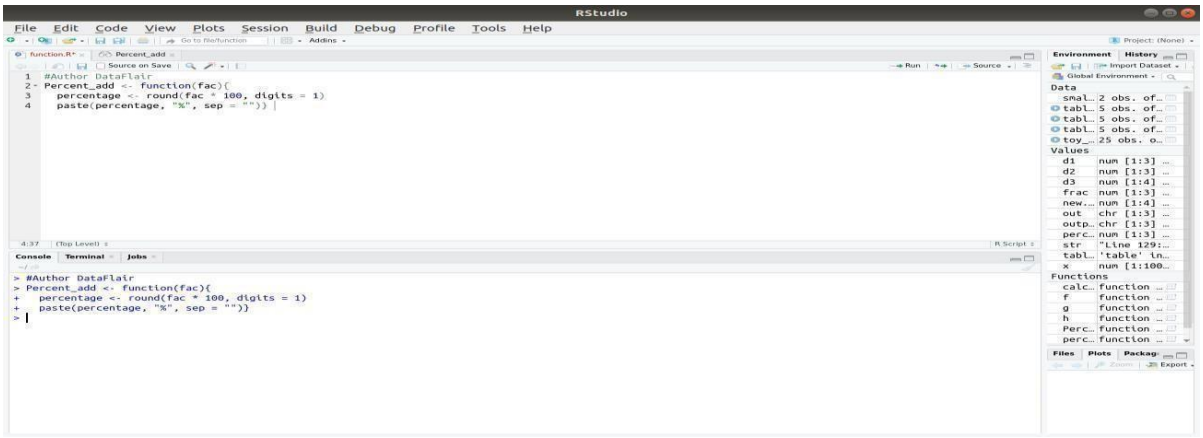
Percent_add <- **function**(frac){

if(!is.numeric(frac)) **return**(**NULL**)

percentage <- round(frac * 100, digits =

1) paste(percentage, "%", sep = "")}

Output:



2. Dropping the {}

You can drop braces in some cases though they form a **proverbial wall** around the function. If a function consists of only one line of code, you can just add that line after the argument list without enclosing it in braces. R will see the code after the argument list as the body of the function.

Suppose, you want to calculate the odds from a proportion. You can write a function without using braces, as shown below:

> odds <- **function**(x) x / (1-x)
Here no braces are used to write the function..

Scope of R Function

Every object you create ends up in this environment, which is also called the global environment. The workspace or global environment is the universe of the R user where everything happens.

There are two types of R functions as explained below:

1. External R Function

If you use an R function, the function first creates a temporary local environment. This local environment is nested within the global environment, which means that, from that local environment, you also can access any object from the global environment. As soon as the function ends, the local environment is destroyed along with all the objects in it.

If R sees any object name, it first searches the local environment. If it finds the object there, it uses that one else it searches in the global environment for that object.

2. Internal R Function

Using global variables in an R function is not considered a good practice. Writing your functions in such a way that they need objects in the global environment is not efficient because you use functions to avoid dependency on objects in the global environment in the first place.

The whole concept behind R strongly opposes using global variables used in different functions. As a functional programming language, one of the main ideas of R is that the outcome of a

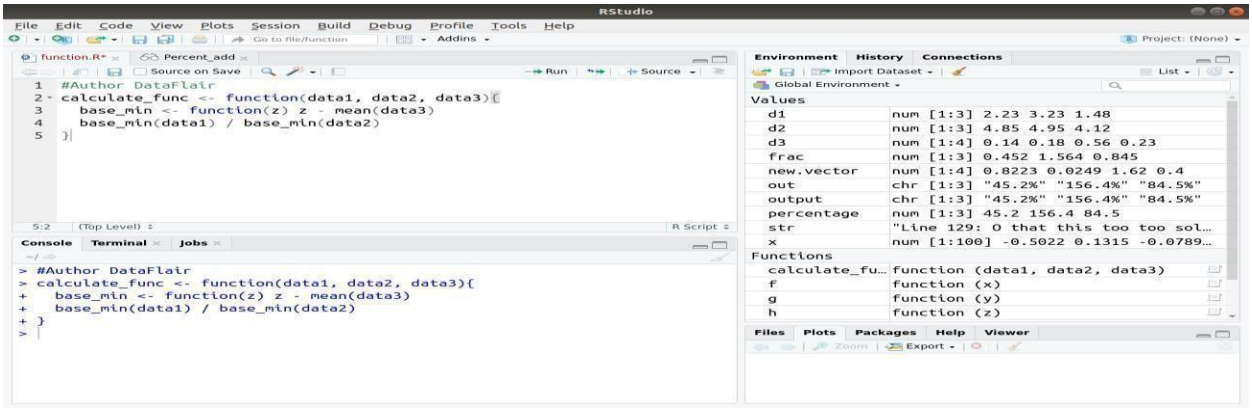
function

should not be dependent on anything but the values for the arguments of that function. If you give the arguments for the same values, you will always get the same results.

The below example shows the usage of internal function:

```
#Author DataFlair
calculate_func <- function(data1, data2, data3){
base_min <- function(z) z - mean(data3)
base_min(data1) / base_min(data2)
}
```

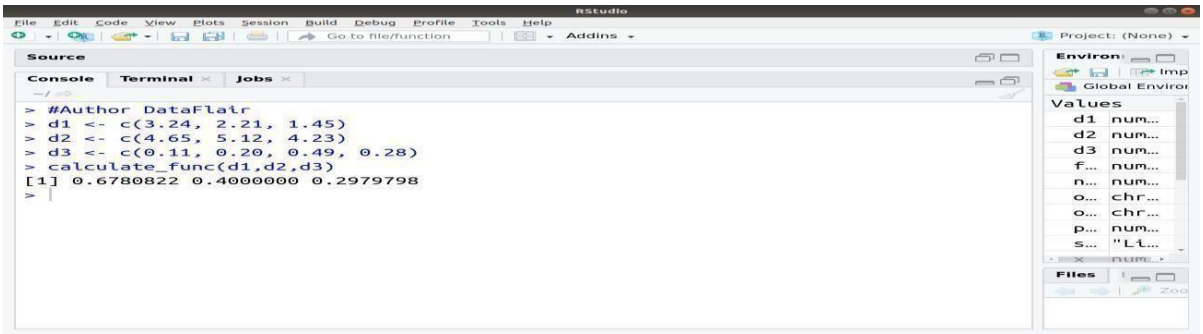
Output:



The code for calculate.eff function is shown below:

```
> #Author DataFlair
> d1 <- c(3.24, 2.21, 1.45)
> d2 <- c(4.65, 5.12, 4.23)
> d3 <- c(0.11, 0.20, 0.49, 0.28)
> calculate_func(d1,d2,d3)
```

Output:



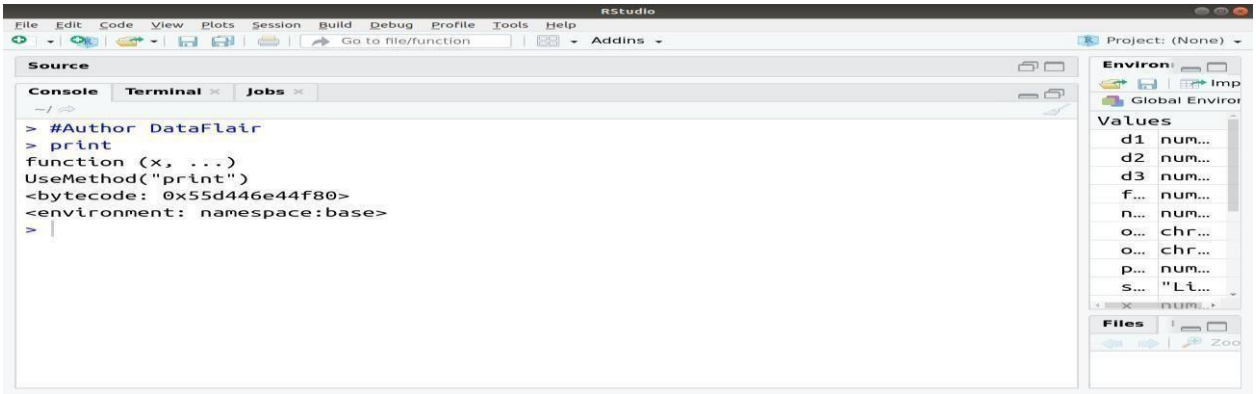
A closer look at the R function definition of base_min() shows that it uses an object control but does not have an argument with that name.

Finding the Methodsbehindthe Function

It is easy to find out the function you used in R. You can just look at the function code of print() by typing its name at the command line.

In order to display the info code of the print() function, we proceed as follows:

Output:



1. The UseMethod() Function

The UseMethod() function contains central function in the main generic function system.

The UseMethod() function moves along and looks for a function that can deal with the type of object that is given as the argument x.

Suppose you have a data frame that you want to print. The object that was passed as an argument will be printed using the print.data.frame() function that will be first looked up by R.

The other functions are searched through a thorough search of another function. The procedure is then started with a print which is then followed by the object type and the dot.

The function print.data.frame() can also be called inside the code.

2. Calling Functions Inside Code

You can also call the function print.data.frame() yourself.

Below is the example for the same:

```
> #Author DataFlair
> small_data <- data.frame(c1 = 1:2, c2 = 2:1)
> print.data.frame(small_data)
```

Output:

Using Default Methods in R

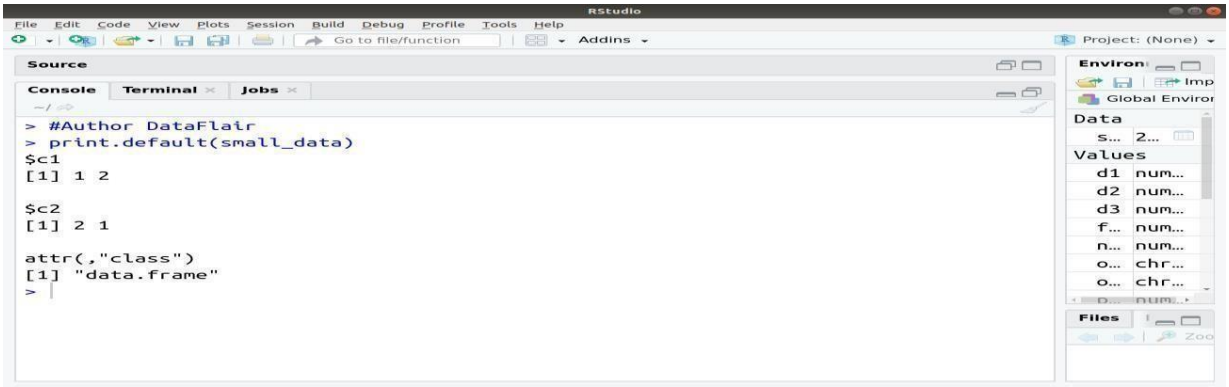
R provides the feature to create an object with the names that are already used by R. It is possible with the use of default keyword.

R will ignore the type of the object in that case and just look for a default method if you use the default keyword with the name of an object.

Below example explains it:

```
> #Author DataFlair
> print.default(small_data)
```

Output:



Performing Graphical Analysis in R

Graphs are useful for non-numerical data, such as colours, flavours, brand names, and more. When numerical measures are difficult or impossible to compute, graphs play an important role.

Statistical computing is done with the aim to produce high-quality graphics.

Various types of plots drawn in R programming are:

- **Plots with Single Variable –** You can plot a graph for a single variable.
- **Plots with Two Variables –** You can plot a graph with two variables.
- **Plots with Multiple Variables –** You can plot a graph with multiple variables.
- **Special Plots –** R has low and high-level graphics facilities.

1. Plots with Single Variable

You may need to plot for a single variable in graphical data analysis with R programming.
For example – A plot showing daily sales values of a particular product over a period of time. You can also plot the time series for month by month sales.
The choice of plots is more restricted when you have just one variable to the plot. There are various plotting functions for single variables in R:

- **hist(y) –** Histograms to show a frequency distribution.
- **plot(y) –** We can obtain the values of y in a sequence with the help of the plot.
- **plot.ts(y) –** Time-series plots.
- **pie(x) –** Compositional plots like pie diagrams.

The types of plots available in R:

- **Histograms** – Used to display the mode, spread, and symmetry of a set of data.
- **Index Plots –** Here, the plot takes a single argument. This kind of plot is especially useful for error checking.
- **Time Series Plots** – When a period of time is complete, the time series plot can be used to join the dots in an ordered set of y values.

- **Pie Charts –** Useful to illustrate the proportional makeup of a sample in presentations.

Histograms have the response variable on the x-axis, and the y-axis shows the frequency of different values of the response. *In contrast, a* bar chart has the response variable on the y-axis and a categorical explanatory variable on the x-axis.

1. Histograms

Histograms display the mode, the spread, and the symmetry of a set of data. The R function *hist()* is used to plot histograms.

The x-axis is divided into which the values of the response variable are distributed and then counted. This is called bins. Histograms are tricky because it depends on the subjective judgments of where exactly to put the bin margins that what graph you will be looking at. Wide bins produce one picture, narrow bins produce a different picture, and unequal bins produce confusion.

Small bins produce multimodality (a combination of audio, textual, and visual modes), whereas broad bins produce unimodality (contains a single-mode). When there are different bin widths, the default in R is for this to convert the counts into densities.

The convention adopted in R for showing bin boundaries is to employ square and round brackets, so that:

- [a,b) means 'greater than or equal to a but less than b' [square than round).
- (a,b] means 'greater than a but less than or equal to b' (round than square].

You need to take care that the bins can accommodate both your minimum and maximum values.

The *cut()* function takes a continuous vector and cuts it up into bins that can then be used for counting.

The *hist()* function in R does not take your advice about the number of bars or the width of bars. It helps simultaneous viewing of multiple histograms with similar range. For small integer data, you can have one bin for each value.

*In R, the **parameter k** of the **negative binomial distribution** is known as **size** and the **mean** is known as **mu**.*

Drawing histograms of continuous variables is a more challenging task than explanatory variables. This problem depends on the density estimation that is an important issue for statisticians. To deal with this problem, you can approximately transform continuous model to a discrete model using a linear approximation to evaluate the density at the specified points.

The choice of bandwidth is a compromise made between removing insignificant bumps and real peaks.

2 Index Plots

For plotting single samples, index plots can be used. The plot function takes a single argument. This is a continuous variable and plots values on the y-axis, with the x coordinate determined by the position of the number in the vector. Index plots are especially useful for error checking.

3 Time Series Plot

The time series plot can be used to join the dots in an ordered set of y values when a period of time is complete. The issues arise when there are missing values in the time series (e.g., if sales values for two months are missing during the last five years), particularly groups of missing values (e.g., if sales values for two quarters are missing during the last five years) and during that period we typically know nothing about the behaviour of the time series.

ts.plot and *plot.ts* are the two functions for plotting time series data in R.

4 Pie Chart

You can use pie charts to illustrate the proportional makeup of a sample in presentations. Here the function *pie* takes a vector of numbers and turns them into proportions. It then divides the circle on the basis of those proportions.

To indicate each segment of the pie, it is essential to use a label. The label is provided as a vector of character strings, here called *data$names*.

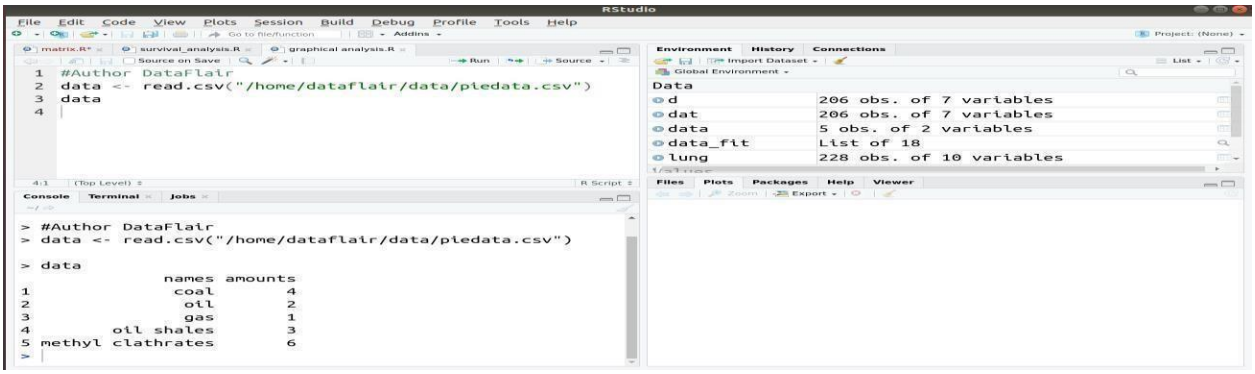
If a names list contains blank spaces then you cannot use *read.table* with a tab-delimited text file to enter the data. Instead, you can save the file called piedata as a comma-delimited file, with a ".csv" extension, and input the data to R using *read.csv* in place of *read.table.*

#Author DataFlair

data <-

read.csv("/home/dataflair/data/piedata.csv") data

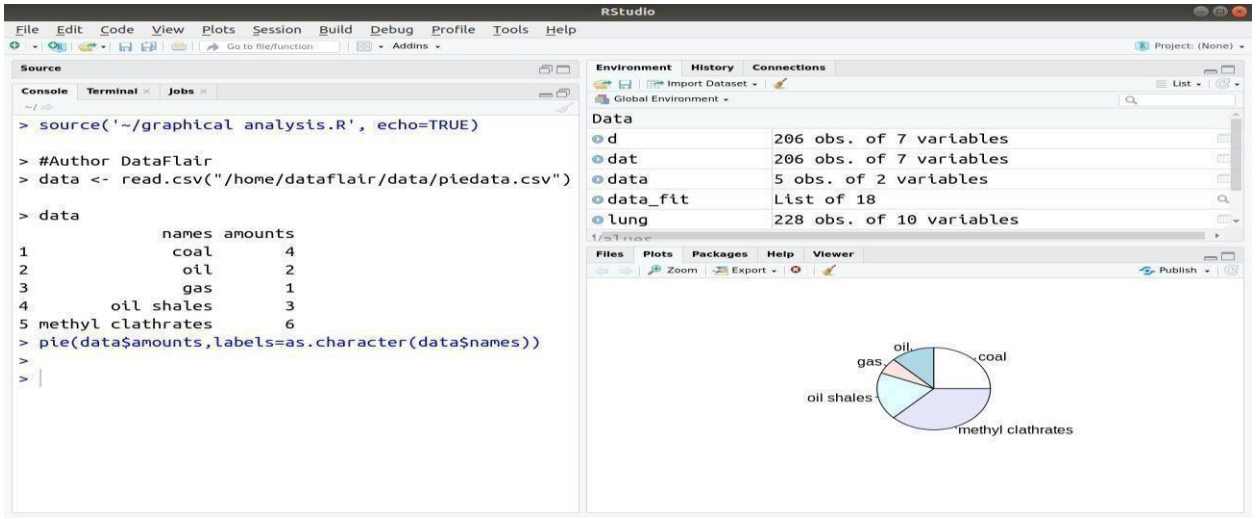
Output:



The pie chart can be created, using the following command:

pie(data$amounts,labels=as.character(data$names))

Output:



2. Plots with Two Variables

The two types of variables used in the graphical data analysis with R:

- Response variable
- Explanatory variable

*The **response variable** is represented on the **y-axis** and the **explanatory variable** is represented on the **x-axis**.*

When an explanatory variable is categorical, like genotype or colour or gender, the appropriate plot is either a box-and-whisker plot or a barplot.

A barplot provides a graphical representation of data in the form of bar charts.

The most frequently used plotting functions for two variables in R:

- **plot(x, y):** Scatterplot of y against x
- **plot(factor, y):** Box-and-whisker plot of y at each factor level.
- **barplot(y):** Heights from a vector of y values (one bar per factor level).

The types of plots available in R:

- **Scatterplots –** When the explanatory variable is a continuous variable.
- **Stepped Lines –** Used to plot data distinctly and provide a clear view.
- **Boxplots –** Boxplots show the location, spread of data and indicate skewness.
- **Barplots –** It shows the heights of the mean values from the different treatments.

Scatterplots

Scatterplots shows a graphical representation of the relationship between two numbered sets. The plot function draws axis and adds a scatterplot of points. You can also add extra points or lines to an existing plot by using the functions, point, and lines.

The points and line functions can be specified in the following two ways:

- **Cartesian plot (x, y) –** A Cartesian coordinate specifies the location of a point in a two-dimensional plane with the help of two perpendicular vectors that are known as an axis. The origin of the Cartesian coordinate system is the point where two axes cut each other and the location of this point is the (0,0).
- **Formula plot (y, x) –** The formula based plot refers to representing the relationship between variables in the graphical form. **For example –** The equation, $y=mx+c$, shows a straight line in the Cartesian coordinate system.

The advantage of the formula-based plot is that the plot function and the model fit look and feel the same. The Cartesian plots build plots using "*x* then *y*" while the model fit uses *"y* then *x*".

The plot function uses the following arguments:

- The name of the explanatory variable.
- The name of the response variable.

The basic syntax of a scatterplot is as follows:

plot(x, y, main, xlab, ylab, xlim, ylim, axes)

where **x** is the data present on the horizontal coordinates.

y is the data that is present on the vertical axis.

main represents the title of our plot.

xlab is the label that denotes the horizontal axis.

ylab is the label for the vertical

axis. **xlim** is the limits of x for

plotting. **ylim** is the limits of y for

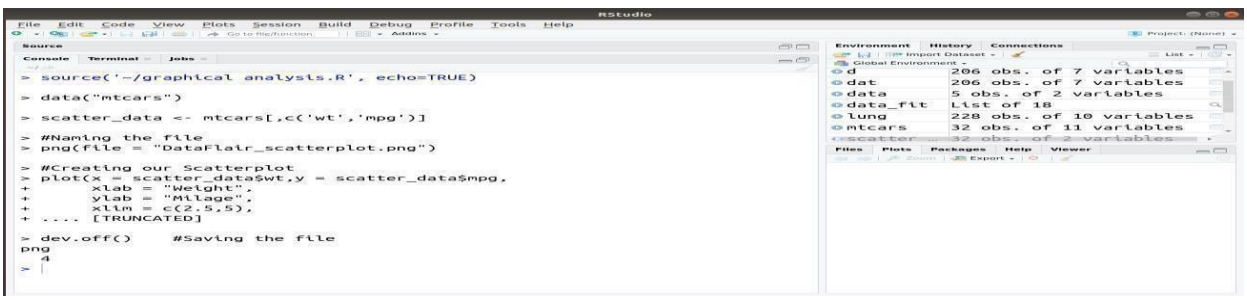
plotting.

axes give an indication that both the axes should be present on the plot.

For creating our scatterplot, we will use the 'mtcars' dataset that is present in the R data library. We plot the graph between the labels 'wt' and 'mpg'.

```
data("mtcars")
scatter_data <- mtcars[,c('wt','mpg')]
#Naming the file
png(file =
"DataFlair_scatterplot.png") #Creating
our Scatterplot
plot(x = scatter_data$wt,y =
scatter_data$mpg, xlab = "Weight",
ylab =
"Milage", xlim
= c(2.5,5), ylim
= c(15,30),
main = "Weight vs Milage"
)
dev.off() #Saving the file
```

Output:



The best way to identify multiple individuals in scatterplots is to use a combination of colours and symbols. A useful tip is to use *as.numeric* to convert a grouping factor into colour and/or a symbol.

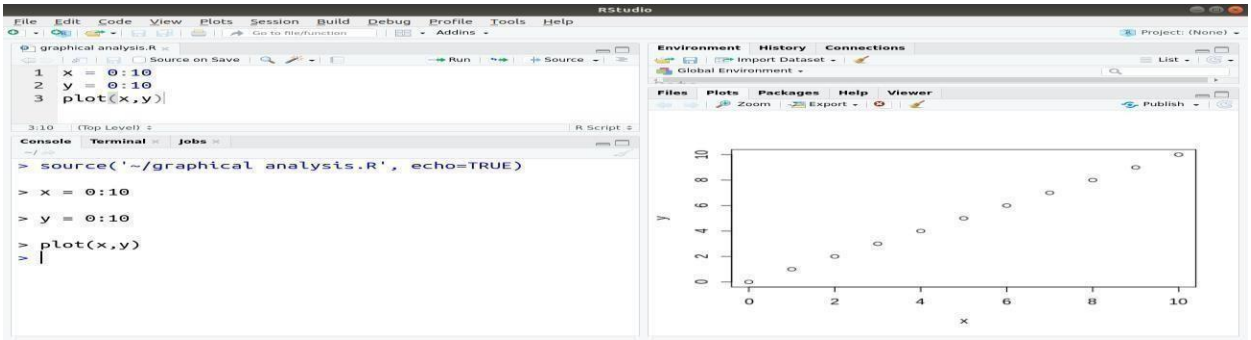
Stepped Lines

Stepped lines can be plotted as graphical representation displays in R. These plots, plot data distinctly and also provide a clear view of the differences in the figures.

While plotting square edges between two points, you need to decide whether to go across and then up, or up and then across. Let's assume that we have two vectors from 0 to 10. We plot these points as follows:

```
x = 0:10
y = 0:10
plot(x,y)
```

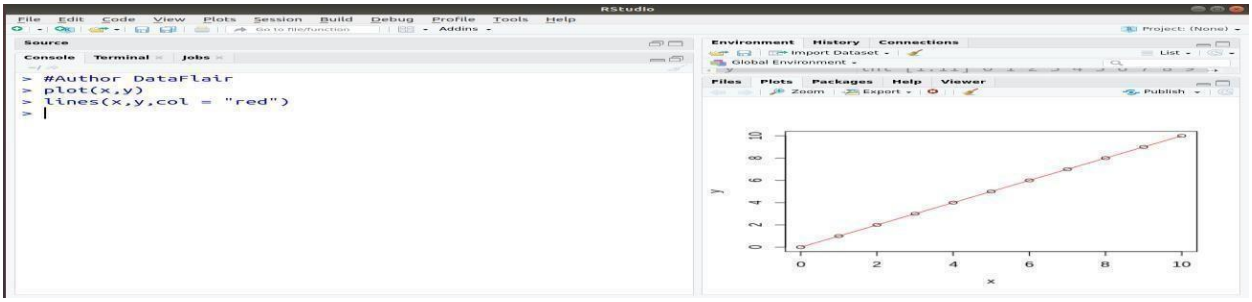
Output:



We can draw a straight line by using the following command:

```
> #Author DataFlair
> plot(x,y)
> lines(x,y,col = "red")
```

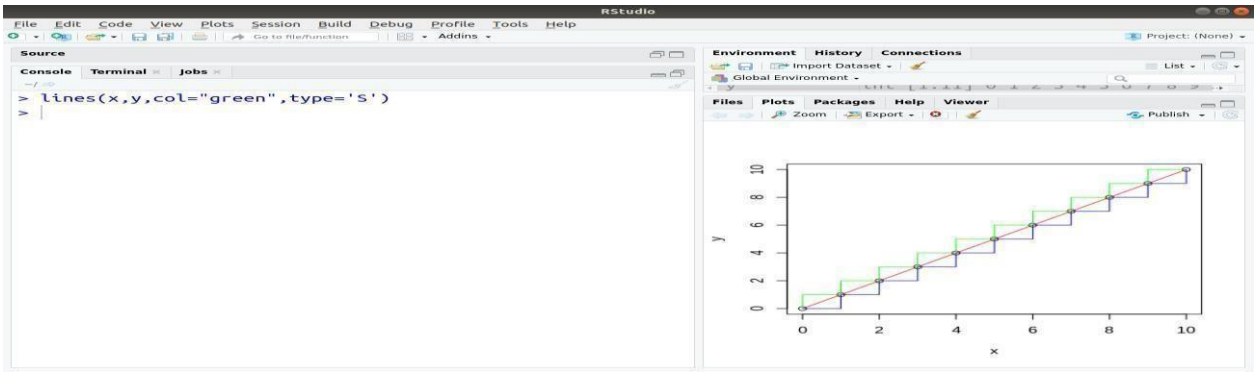
Output:



Also, generate a line by using the upper case "S" as shown below:

```
> lines(x,y,col="green",type='S')
```

Output:



Box and Whisker Plot

A box-and-whisker plot is a graphical means of representing sets of numeric data using quartiles. It is based on the minimum and maximum values, and upper and lower quartiles. Boxplots summarizes the information available. The vertical dash lines are called the 'whiskers'. Boxplots are also excellent for spotting errors in data. The extreme outliers represent these errors.

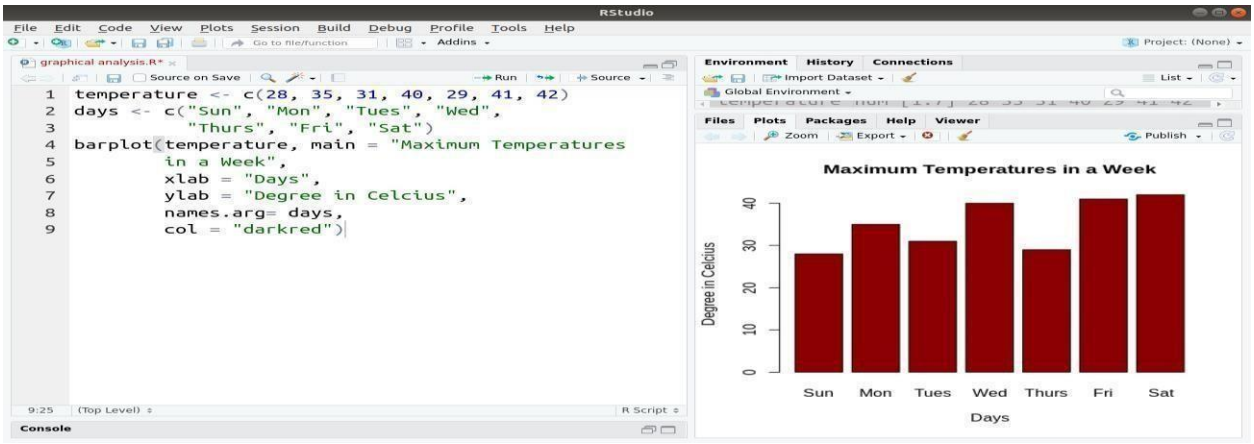
Barplot

Barplot is an alternative to boxplot to show the heights of the mean values from the different treatments. Function *tapply* computes the height of the bars. Thus it works out the mean values for each level of the categorical explanatory variable.

Let us create a toy dataset of temperatures in a week. Then, we will plot a barplot that will have labels.

```
temperature <- c(28, 35, 31, 40, 29, 41, 42)
days <- c("Sun", "Mon", "Tues", "Wed",
"Thurs", "Fri", "Sat")
barplot(temperature, main = "Maximum Temperatures
in a Week",
xlab = "Days",
ylab = "Degree in
Celcius", names.arg= days,
col = "darkred")
```

Output:



3. Plots with Multiple Variables

Initial data inspection using plots is even more important when there are many variables, any one of which might have mistakes or omissions. The principal plot functions that represent multiple variables are:

- **The Pairs Function –** For a matrix of scatterplots of every variable against every other.
- **The Coplot Function –** For conditioning plots where y is plotted against x for different values of z.

It is better to use more specialized commands when dealing with the rows and columns of data frames.

<u>The Pairs Function</u>

For two or more continuous explanatory variables, it is valuable to check for subtle dependencies between the explanatory variables. Rows represent the response variables and columns represent the explanatory variables.

Every variable in the data frame is on the y-axis against every other variable on the x-axis using the pairs function plots. The pairs function needs only the name of the whole data frame as its first argument.

The Coplot Function

The relationship between the two variables may be obscured by the effects of other processes in multivariate data. When you draw a two-dimensional plot of y against x, then all the effects of other explanatory variables are shown onto the plane of the paper. In the simplest case, we have one response variable and just two explanatory variables.

The coplot panels are ordered from lower left to upper right, associated with the values of the conditioning variable in the upper panel from left to right.

Special Plots in Graphical Data Analysis with R

R has extensive facilities for producing graphs. It also has low and high-level graphics facilities as per the requirement.

The low-level graphics are the basic building blocks that can build up graphs step by step, while a high-level facility provides the variety of pre-assembled graphical display.

Apart from the various kinds of graphical plots discussed, R supports the following special plots:

- **Design Plots –** Effective sizes in designed experiments can be visualized using design plots. One can plot the design plots using the *plot.design* function – *plot.design(Growth.rate~Water*Detergent*Daphnia)*
- **Bubble Plots –** Useful for illustrating the variation in the third variable across different locations in the *x–y*.
- **Plots with many Identical Values –** Sometimes, two or more points with count data fall in exactly the same location in a scatterplot. As a result, the repeated values of y are hidden, one beneath the other.

Adding Other Shapes to a Plot

Using the following functions, we can add the extra graphical objects in plots:

- **rect –** For plotting rectangles – *rect(xleft, ybottom, xright, ytop)*

Using the locator function, we can obtain the coordinates of the corners of the rectangle. But the rect function does not accept locator as its argument.

- **arrows –** For plotting arrows and headed bars – The syntax for the arrows function is to draw a line from the point *(xO, yO)* to the point *(x1, y1)* with the arrowhead, by default, at the "second" end *(x1, y1)*.

arrows(xO, yO, xl, yl)

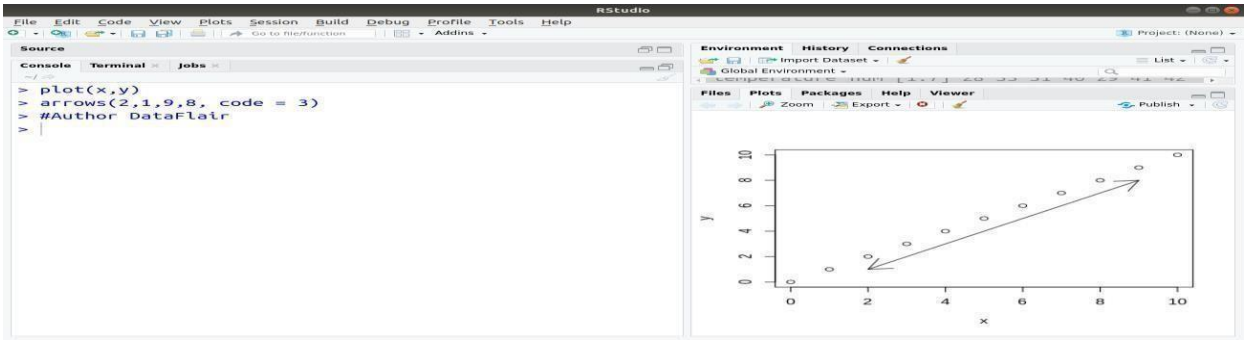
Adding *code = 3* produces a **horizontal double-headed arrow** from *(2,1)* to *(9,8)*, for example:

plot(x,y)

arrows(2,1,9,8, code =

3) #Author DataFlair

Output:



Saving Graphics to File in R

You are likely to want to save each of your plots as a PDF or PostScript file for publication-quality graphics. This is done by specifying the 'device' before plotting, then turning the device off once finished.

The computer screen is the default device, where we can obtain a rough copy of the graph, using the following command:

data <- read.table (filename,

header=T) attach(data)

There are numerous options for the PDF and postscript functions, but width and height are the ones that you will want to change most often. The sizes are in inches. You can specify any nondefault arguments that you want to change using the functions *pdf.options* and *ps.options* before you invoke either PDF or postscript.

Selecting an Appropriate Graph in R

It is also important to select an appropriate type of graph according to the requirements.

Some common graphs and their uses are as follows:

- **Line Graph –** It displays over the time period. It generally keeps the track of records for both, long-time period and short-time period according to requirements. In the case of small change, *the line graph is more common than the bar graph.* In some cases, the line graphs also compare the changes among different groups in the same time period.
- **Pie Chart –** It displays comparison within a group.
  For example – You can compare students in a college on the basis of their streams, such as arts, science, and commerce using a pie chart. One cannot use a pie chart to show changes over the time period.
- **Bar Graph –** Similar to a line graph, the bar graph generally compares different groups or tracking changes over a defined period of time. Thus the difference between the two graphs is that the line graph tracks small changes while a bar graph tracks large changes.
- **Area Graph –** The area graph tracks the changes over the specific time period for one or more groups related to a similar category.
- **X-Y Plot –** The X-Y plot displays a certain relationship between two variables. In this type of variable, the X-axis measures one variable and Y-axis measures another variable. On the one hand, if the values of both variables increase at the same time, a positive relationship exists between variables. On the other hand, if the value of one variable decreases at the time of the increasing value of another variable, a negative relationship exists between variables. It could be also possible that the two variables don't have any relationship. In this case, plotting graph has no meaning.

UNIT – V

Big Data Visualization:

Introduction to Data visualization, Challenges to Big data visualization, Types of data visualization, Visualizing Big Data, Tools used in data visualization, Proprietary Data Visualization tools, Open-source data visualization tools, Data visualization with Tableau.

Data visualization

Data visualization is a graphical representation of any data or information. Visual elements such as charts, graphs, and maps are the few data visualization tools that provide the viewers with an easy and accessible way of understanding the represented information.

Data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

Importance of Data Visualization:

- Easily, graspable information – Data is increasing day-by-day, and it is not wise for anyone to scram through such quantity of data to understand it. Data visualization comes handy then.
- Establish relationships – Charts and graphs do not only show the data but also established co-relations between different data types and information.
- Share – Data visualization is also easy to share with others. You could share any important fact about a market trend using a chart and your team would be more receptive about it.
- Interactive visualization – when technological inventions are making waves in every market segment, regardless of big or small, you could also leverage interactive visualization to dig deeper and segment the different portions of charts and graphs to obtain a more detailed analysis of the information being presented.
- Intuitive, personalized, updatable – Data visualization is interactive. You could click on it and get another big picture of a particular information segment. They are also tailored according to the target audience and could be easily updated if the information modifies.

Example of Data visualization:

Profit and loss – Business companies often resort to pie charts or bar graphs showing their annual profit or loss margin.

Challenges of Big Data Visualization

Scalability and dynamics are two major challenges in visual analytics

The visualization-based methods take the challenges presented by the "four Vs" of big data and turn them into following opportunities.

- *Volume*: The methods are developed to work with an immense number of datasets and enable to derive meaning from large volumes of data.
- *Variety*: The methods are developed to combine as many data sources as needed.
- *Velocity*: With the methods, businesses can replace batch processing with real-time stream processing.
- *Value*: The methods not only enable users to create attractive info graphics and heat maps, but also create business value by gaining insights from big data.

Big data often has unstructured formats. Due to bandwidth limitations and power requirements, visualization should move closer to the data to extract meaningful information efficiently.

Effective data visualization is a key part of the discovery process in the era of big data. For the challenges of high complexity and high dimensionality in big data, there are different dimensionality reduction methods.

There are also following problems for big data visualization:

- *Visual noise:* Most of the objects in dataset are too relative to each other. Users cannot divide them as separate objects on the screen.
- *Information loss:* Reduction of visible data sets can be used, but leads to information loss.
- *Large image perception:* Data visualization methods are not only limited by aspect ratio and resolution of device, but also by physical perception limits.
- *High rate of image change: Users* observe data and cannot react to the number of data change or its intensity on display.
- *High performance requirements:* It can be hardly noticed in static visualization because of lower visualization speed requirements--high performance requirement.

In Big Data applications, it is difficult to conduct data visualization because of the large size and high dimension of big data. Most of current Big Data visualization tools have poor performances in scalability, functionalities, and response time. Uncertainty can result in a great challenge to effective uncertainty-aware visualization and arise during a visual analytics process.

Potential solutions to some challenges or problems about visualization and big data were presented:

- Meeting the need for speed: One possible solution is hardware. Increased memory and powerful parallel processing can be used. Another method is putting data in-memory but using a grid computing approach, where many machines are used.
- Understanding the data: One solution is to have the proper domain expertise in place.
- Addressing data quality: It is necessary to ensure the data is clean through the process of data governance or information management.
- Displaying meaningful results: One way is to cluster data into a higher-level view where smaller groups of data are visible and the data can be effectively visualized.
- Dealing with outliers: Possible solutions are to remove the outliers from the data or create a separate chart for the outliers.

Types of Data Visualization:

Data can be visualized in many ways, such as in the form of 1D,2D, or 3D structures. The below table briefly describes the different types of data visualization:

Name	Description	Tool
1D/Linear	For example, a list of items organized in a predefined manner	Generally, no tool is used for 1D visualization

2D/Planar	For example, choropleth, cartogram, dot distribution map, and proportional symbol map	GeoCommons, Google Fusion Tables, Google Maps, API, Polymaps, Many Eyes, Google Charts, and Tableau Public
3D/ Volumetric	For example, 3D computer models, surface rendering, volume rendering, and computer simulations	TimeFlow, timeline JS, Excel, Timeplot, TimeSearcher, Google charts, Tableau Public, and Google Fusion Tables
Multidimensional	For example, pie chart, histogram, tag cloud, bubble cloud, bar chart, scatter plot, heat map, etc.	Many eyes, google charts, Tableau public, and google fusion tables
Tree/ Hierarchical	For example, dendogram, radial tree, hyperbolic tree, and wedge stack graph	D3,google charts, and network workbench/Sci2
Network	For example, matrix, node link diagram, hive plot, and tube map	Pajek, gephi, nodeXL, VOSviewer, UCINET, GUESS, Network workbench/Sci2, sigma.js,d3/Protovis, Many eyes, and google fusion tables

As shown in the table, the simplest type of data visualization is 1D representation and the most complex data visualization is the network representation. The following is a brief description of each of these data visualizations:

- **1D (Linear) Data Visualization –** In the linear data visualization, data is presented in the form of lists. Hence, we cannot term it as visualization. It is rather a data organization technique. Therefore, no tool is required to visualize data in a linear manner.
- **2D(planar) Data visualization –**This technique presents data in the form of images, diagrams or charts on aplane surface.
- **3D (volumetric) Data visualization –**In this method, data presentation involves exactly three dimensions to show simulations, surface and volume rendering, etc. generally, it is used in scientific studies. Today, many organizations use 3D computer modeling and volume rendering in advertisements to provide users a better feel of their products.
- **Temporal Data Visualization –** sometimes, visualizations are time dependent. To visualize the dependence of analyses on time, the temporal data visualization is used.
- **Multidimensional Data visualization –** in this type of data visualization, numerous dimensions are used to present data.
- **Tree/Hierarchical Data visualization –** sometimes, data relationships need to be shown in the form of hierarchies. To represent such kind of relationships, we use tree or hierarchical data visualizations.
- **Network data visualization –** It is used to represent data relations that are too complex to be represented in the form of hierarchies.

Visualizing Big Data:

Data visualization is a great way to reduce the turn-around time consumed in interpreting big data. Traditional analytical techniques are not enough to capture or interpret the information that big data possesses. Traditional tools are developed by using relational models that works best on static interaction. Big data is highly dynamic in function and therefore, most traditional tools are not able to generate quality results. The response time of traditional tool is quite high, making it unfit for quality interaction.

Deriving Business solution:

The most common notation used for big data is 3 V's- volume, velocity, and variety. But, the most exciting feature is the way in which value is filtered from the haystack of data.

Now a days, IT companies that are using Big Data faces the following challenges:

- □ Most data is in unstructured form
- □ Data is not analyzed in real time
- □ The amount of data generated is huge
- □ There is a lack of efficient tools and techniques

By considering the above factors, IT companies are focusing more on research and development of robust algorithm, software, and tools to analyze the data that is scattered in the internet space.

Turning Data into Information:

Visualization of data produces cluttered images that are filtered with the help of clutter- reduction techniques. Uniform sampling and dimension reduction are two commonly used clutter- reduction techniques.

Visual data reduction process involves automated data analysis to measure density, outliers, and their differences. These measures are then used as quality metrics to evaluate data – reduction activity.

Visual quality metrics can be categorized as:

- □ Size metrics
- □ Visual effectiveness metrics
- □ Feature preservation metrics

A visual analytics tool should be:

- □ Simple enough so that even non- technical users can operate it

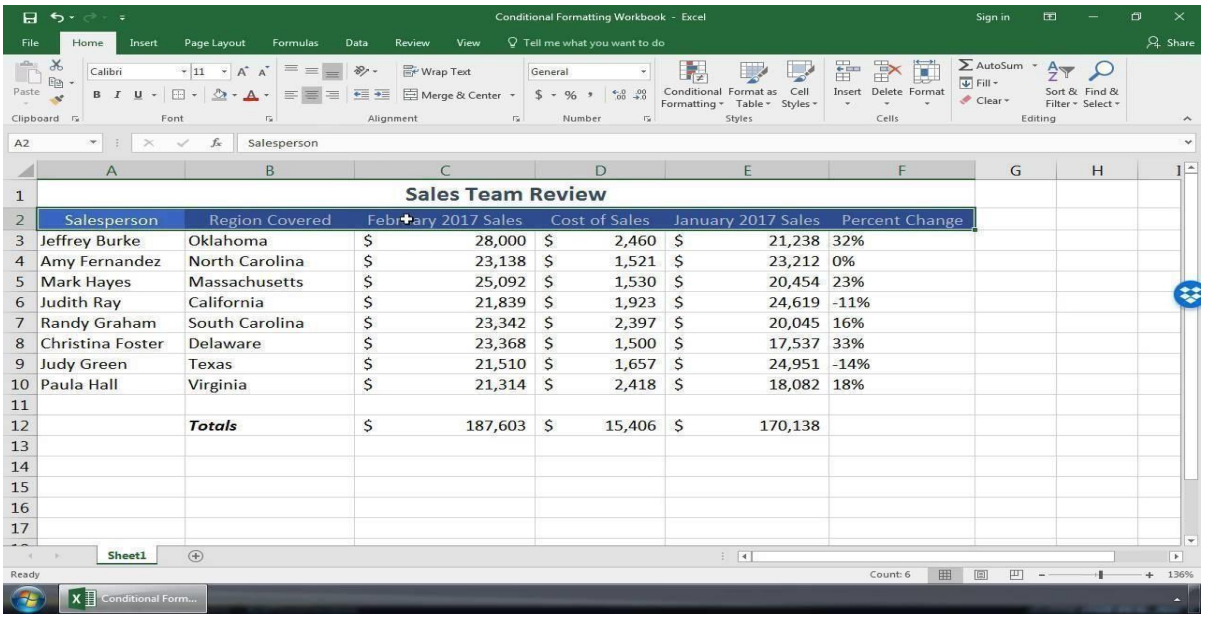
- □ Interactive to connect with different sources of data
- □ Component to create appropriate visuals for interpretations
- □ Able to interpret big data and share information

A part from representing data, a visualization tool must be able to establish links between different data values, restore the missing data, and polish data for further analysis.

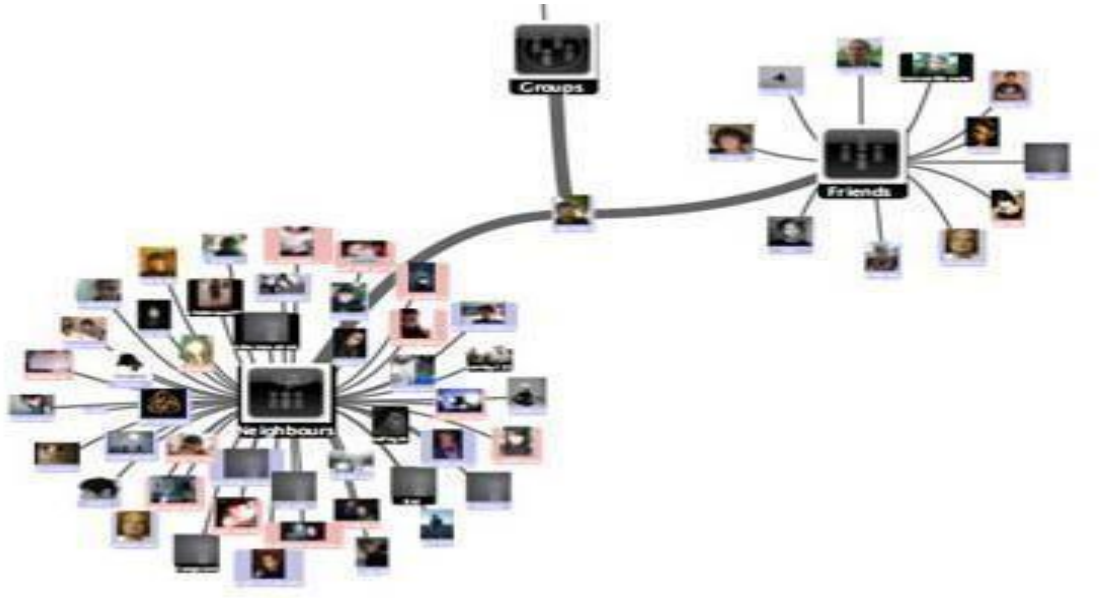
Tools Used in Data Visualization:

Some useful visualization tools are listed as follows:

EXCEL – It is a new tool that is used for data analytics. It helps you to track and visualize data for deriving better insights. This tool provides various ways to share data and analytical conclusions within and across organizations.



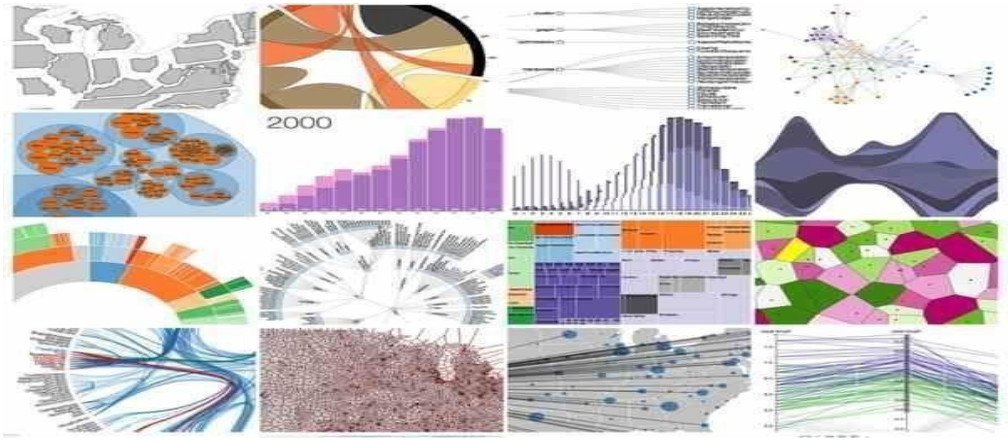
LastForward – It is open – source software provided by last.fm for analyzing and visualizing social music network.



Digg.com – Digg.com provides some of the best web-based visualization tools.

Pics – This tool is used to track the activity of images on the website.

D3 – D3 allows you to bind arbitrary data to a document object model (DOM) and then applies data- driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interactions.



Rootzmap Mapping the internet – It is a tool to generate a series of maps on the basis of the datasets provided by the National Aeronautics and Space Administration (NASA).

Open source data visualization tools

Due to economic and infrastructural limitations, every organization cannot purchase all the applications required for analyzing data. Therefore, to fulfill their requirement of advanced tools and technologies, organizations often turn to open- source libraries. These libraries can be defined as pools of freely available applications and analytical tools. Some examples of open- source tools available for data visualization are VTK, Cave5D, ELKI, Tulip, Gephi, IBM openDX, Tableau public and vis5D.

- Open source tools are easy to use, consistent, and reusable.
- They deliver performance and are complaint with the web as well as mobile web security.

Analytical Techniques used in Big Data Visualization:

Analytical techniques are used to analyze complex relationships among variables. The following are some commonly used analytical techniques for big data solutions:

Regression analysis – it is a statistical tool used for prediction. Regression analysis is used to predict continuous dependent variables from independent variables.

Types of regression analysis are as follows:

- *Ordinary least squares regression* – it is used when dependent variable is continuous and there exists some relationship between the dependent variable and independent variable.
- *Logistic regression* – it is used when dependent variable has only two potential results.
- *Hierarchical linear modeling* – it is used when data is in nested form
- *Duration models* – it is used to measure length of process

Grouping methods – the technique of categorizing observation into significant or purposeful blocks is called grouping. The recognition of features to create a distinction between groups is called discriminant analysis.

Multiple equation models – it is used to analyze casual pathways from independent variables to dependent variables. Types of multiple equation models are as follows :

- Path analysis
- Structural equation modeling

Data Visualization with tableau

Introduction to Tableau:

Tableau is a Data visualization software that allows developers to build interactive dashboards that are easily updated with new data and can be shared with a wider audience. There are various types of Tableau products available in the market. Some of the commonly known products include Tableau Desktop, Tableau Server, Tableau Online, Tableau Reader, and Tableau Public.

The important features of Tableau Software include the following:

- □ Single- click data analytics in visual form
- □ In- depth statistical analysis
- □ Management of metadata
- □ In built, top-class data analytic practices
- □ In built data engine
- □ Big data analytics
- □ Quick and accurate data discovery
- □ Business dashboards creation
- □ Various types of data visualization
- □ Social media analytics, including Facebook and Twitter
- □ Easy and quick integration of R
- □ Business intelligence through mobile
- □ Analysis of time series data
- □ Analysis of data from surveys

Tableau software can be used in various industries and data environments.

Tableau has primarily been used within the following data environments:

- □ All data sources
- □ Amazon redshift
- □ Excel charts and graphs
- □ Google analytics
- □ Google bigquery
- □ Hadoop
- □ HP vertica
- □ SAP
- □ Splunk

Tableau Desktop Workspace

Tableau's desktop environment is simple and easy to learn almost for anyone.

Tableau is an extremely efficient tool, which can answer your questions about data analytics quickly.

To use tableau desktop, you first need to download and install the tool on your computer.

Steps to download and install the tableau software:

1. Open the link http://www.tableau.com/products/desktop in the web browser
2. Download the trail version of the tableau desktop by clicking the TRY IT Free button.
3. Go to the directory of your computer where you have stored the tableau desktop setup and double click the executable file. The tableau desktop installer, showing the tableau version number on top, will appear.
4. Click the I have read and accept the terms of this License Agreement check box to enable the install button
5. Click the install button to start the installation of the tableau desktop.
6. Click the start trail now option. This will ask for registration.
7. Click the register button to open the registration form and provide all the required details.

Click the continue button. The open page of tableau tool appears.

Toolbar Icons:

Tableau is a GUI- oriented drag and drop tool. The following figure shows the icons present on the tableau toolbar.

- **Undo/ Redo –** scrolls backward or forward on the screen. You can retrieve any step by clicking the undo/redo button
- **File save – saves** your work. You need to click this button frequently as tableau does not have the automated save function.
- **Connect to a new data source –** connects you to a data source
- **New dashboard or worksheet –** Adds new page to your worksheet
- **Duplicate sheet-** creates an exact/duplicate copy of a worksheet as well as of the dashboard page that you are working on
- **Clear sheet –** Allows you to clear data of a sheet
- **Auto/ manual update –** generates visual. It is particularly helpful for large datasets where dragging and dropping items consume time
- **Group –** allows you to group data by selecting more than one headers in a table or values in a legend.
- **Pivot worksheet-** allows you to create a pivot table on a new worksheet
- **Ascending/ Descending sort -** Sorts selected items in an ascending or descending order.
- **Label Marks -** turns on or off screen elements.
- **Presentation mode –** Hides/Unhides design shelves. It is particularly used during presentations where you want to use Tableau as a presentation slide deck to keep the slides of the presentation.
- **Reset cards –** provides a menu to turn on or off screen elements, such as caption or summary.

- **Fit Menu –** allows different views of the tableau screen. You can fit the screen either horizontally or vertically.
- **Fit Axis -** Fixes the axis of view. You can zoom in/out charts with this button
- **Highlight control -** compares the selected combinations of dimensions.

Main menu

Main menu of tableau contains following options :

File – contains general functions, such as open, save, and save as. Other functions are print to pdf and Repository location function to review and change the default location of the saved file.

Data – helps to analyze the tabular data on the tableau website. The edit relationships option is used to blend data when the field names in two data sources are not identical

Worksheet – provides option such as export option, excel crosstab, and duplicate as crosstab

Dashboard – Provides the actions menu, which is the most important option on the dashboard menu because all the actions related to tableau worksheets and dashboards are defined within the actions menu. The actions menu is present under the worksheet menu as well.

Story – provides the new story option that is used for explaining the relationship among facts, providing context to certain events, showing the dependency between decisions and outcomes, etc.

Analysis – provides the aggregate measures and stack mark options. To create new measures or dimensions, use create calculated field or edit calculated field.

Map – provides options to change the color scheme and replace the default maps

Format – contains options like cell size and workbook theme

Server – provides options to publish work on tableau server

Window – provides the bookmark menu, which is used to create .tbm files that can be shared with different users

Help – provides options to access tableau's online manual, training videos, and sample workbooks

Tableau Server

In Tableau Server Users can interact with the dashboards on the server without any installation on their machines. Tableau Online is Tableau Server hosted by Tableau on a cloud platform. Tableau server also provides robust security to the dashboards. Tableau Server web-edit feature allows authorized users to download and edit the dashboards. Tableau server allows users to publish and share their data sources as live connections or extracts. Tableau Server is highly secured for visualizing data. It leverages fast databases through live connections

Tableau workbook and data source files

Depending on their utility and the amount of information they contain, tableau saves and shares files as:

- Tableau workbook – it is the default save type when you save your work on the desktop. The extension of such files will be .twb. The files with extension .twbx can be shared with people not having tableau desktop license or those who cannot access the data source.
- Tableau data source – if you frequently connect to a specific data source or if you have manipulated the metadata of any data source, saving the file as tableau data source is of great use. The extension of such a file will be .tds, and it includes server address, password, and metadata.
- Tableau bookmark - if you want to share any specific file with others, use tableau bookmark.
- Tableau data extract – it compresses your extracted data and improves performance by incorporating more formulas and functions. The extension of a tableau data extract file is .tde.

Tableau charts

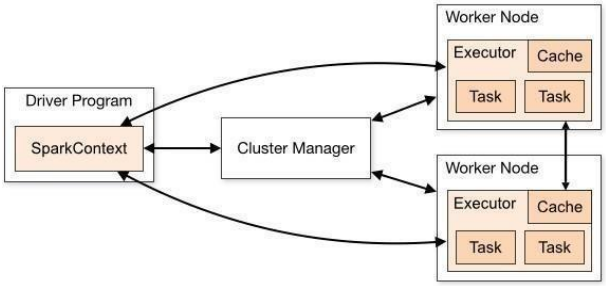
Tableau can create different types of univariate, bivariate, and multivariate charts.

The following are some of the common chart types that tableau can create:

- Tables – tables are an excellent choice of presenting data as they preserve all the information, which in turn minimize the chances of misinterpretation.
- Scatter plots – scatter plots are used to describe the relationship between two variables.
- Trend lines – trend lines are used to analyze the relationship between variables as well as predict the future outcome
- Bullet graph – bullet graph is just like a bar graph and is generally used in qualitative analysis.
- Box plot – box plot represents distribution of data and is used in the comparison of multiple sets of data. It can effectively compute:
    - Minimum and Maximum value
    - Median
    - 25% and 75% quartile
    - Treemap – treemap is one of the best compact techniques to visualize the part to whole relationships as well as hierarchical models.
    - Bubble charts – bubble charts help in categorizing and computing different values and factors in the data with the help of bubbles.
    - Word cloud – similar to bubble charts, the words in a word cloud are sized according to the frequency at which they appear in the content.

18. Additional Topics

Overview of the Spark Architecture



On the left we have a *Driver Program* that runs on the master node. The master node is the one that is responsible for the entire flow of data and transformations across the multiple worker nodes. Usually, when we write our Spark code, the machine to which we deploy acts as the master node. After the *Driver Program*, the very first thing that we need to do is to initiate a *SparkContext*. The *SparkContext* can be considered as a session using which you can use all the features available in Spark. For example, you can consider the *SparkContext* as a database connection within your application. Using that database connection you can interact with the database, similarly, using the SparkContext you can interact with the other functionalities of Spark.

There is also a *Cluster Manager* installed that is used to control multiple worker nodes. The *SparkContext* that we have generated in the previous step works in conjunction with the *Cluster Manager* to manage and control various jobs across all the worker nodes. Whenever a job has to be executed by the *Cluster Manager*, it splits up the entire job into multiple individual tasks and then these tasks are distributed over the worker nodes. This is taken care of by the *Driver Program* and the *SparkContext*. As soon as an RDD is created, it is distributed by the *Cluster Manager* across the multiple worker nodes and cached there.

On the right-hand side of the architecture diagram, you can see that we have two worker nodes. In practice, this can range from two to multiple worker nodes depending on the workload of the system. The worker nodes actually act as the slave nodes that execute the tasks distributed to them by the *Cluster Manager*. These worker nodes return the execution result of the tasks to the *SparkContext*. A key point to mention here is that you can increase your worker nodes such that all the jobs are distributed to each of the worker nodes and as such the tasks can be performed parallelly. This will increase the speed of data processing to a large extent.

19. Known Gaps : Nil

20. Discussion topics

Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

At the present time, Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject). Pig's language layer currently consists of a textual language called Pig Latin, which has the following key properties:

- **Ease of programming.** It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
- **Optimization opportunities.** The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
- **Extensibility.** Users can create their own functions to do special-purpose processing.

21. UNIVERSITY QUESTION PAPERS OF PREVIOUS YEAR

R13

Code No: 117JU

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech IV Year I Semester Examinations, November/December - 2016
BIG DATA ANALYTICS
(Common to CSE, IT)

Time: 3 Hours **Max. Marks: 75**

Note: This question paper contains two parts A and B.
Part A is compulsory which carries 25 marks. Answer all questions in Part A. Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

PART- A

(25 Marks)

1.a)	List the types of accidents.	[2]
b)	Write the elements of data architecture.	[3]
c)	List the stages of OODA Loop.	[2]
d)	What are standard reporting templates?	[3]
e)	What is map reduce?	[2]
f)	What is Key-value data store?	[3]
g)	What are the types of machine learning?	[2]
h)	How do you prepare the input data for an algorithm?	[3]
i)	List Quick Visual Options in Tablue.	[2]
j)	What is role of the workspace in Tablue?	[3]

PART-B

(50 Marks)

2. Explain in detail about Export Job Process. [10]

OR

3. List the Guidelines for identifying and reporting an accident or emergency in detail. [10]

4.a) What is Knowledge Management?
b) Explain about Model Based Techniques. [3+7]

OR

5. Explain about the Kepner-Tregoe Matrix Decision model in detail. [10]

6. List the classification of NoSql Databases and explain about Columns based Database. [10]

OR

7. Explain about the Graph Databases and Descriptive Statistics. [10]

8. Describe Train model using machine learning algorithm, Test model. [10]

OR

9. Explain Knowledge Discovery in Databases task in detail. [10]

10. Explain data visualization in Tablue. [10]

OR

11. Draw insights out of any one visualization tool. [10]

22. References, Journals, websites, and E-links

Websites

1. *https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/what-is-big-data-analytics*
2. https://www.tibco.com/reference-center/what-is-big-data-analytics
3. https://www.databricks.com/glossary/big-data-analytics

REFERENCES

Textbook(s)

1. Big Data, Black Book, DT Editorial Services, ISBN: 9789351197577, 2016 Edition

Reference Book(s)

1. BIG DATA and ANALYTICS, Seema Acharya, Subhasinin Chellappan, Wiley publications
2. Synopses for Massive Data:Samples, Histograms, Wavelets, Sketches, Foundation and trends in databases, Graham Cormode, Minos Garofalakis, Peter J. Haas and Chris Jermaine, 2012.
3. R for Business Analytics, A.Ohri, Springer, ISBN:978-1-4614-4343-8, 2016.
4. Hadoop in practice, Alex Holmes, Dreamtech press, ISBN:9781617292224, 2015.
5. Mining of Massive Datasets, Jure Leskovec Stanford Univ. Anand Rajaraman Milliway Labs Jeffrey D Ullman Stanford University.

Journals

1. The *Journal of Big Data* publishes open-access original research on data science and data analytics.

2. Frontiers in big data is an innovative journal focuses on the power of big data - its role in machine learning, AI, and data mining, and its practical application from cybersecurity to climate science and public health.

3. Big data in research journal aims to promote and communicate advances in big data research by providing a fast and high-quality forum for researchers, practitioners, and policy makers.

4. *Big Data and Cognitive Computing* is an international, scientific, peer-reviewed, open access journal of big data and cognitive computing published quarterly online by MDPI.

5. Journal on Big Data is launched in a new area when the engineering features of big data are setting off upsurges of explorations in algorithms, raising challenges on big data, and industrial development integration.

6. *Big Data Mining and Analytics* (Published by Tsinghua University Press) discovers hidden patterns, correlations, insights and knowledge through mining and analysing large amounts of data obtained from various applications.

23. Quality Control Sheets : Will be submitted End of the semester.

24. Student List

STUDENT NOMINAL ROLL					
Class & Branch : B.Tech (CSE) IV Year I Sem				Section: A	Batch : 2021
Sl.No.	Admn No.	Student Name	Sl.No.	Admn No.	Student Name
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					

25					
26					

Class & Branch : B.Tech (CSE) IV Year I Sem				Section: E	Batch : 2020
Sl.No.	Admn No.	Student Name	Sl.No.	Admn No.	Student Name
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					

25. Group-wise students list for discussion topics : NA