**EXPERIMENT 1**

**TITLE: Hadoop Installation on Single Node**

**OBJECTIVE:**

1. To Learn and understand the concepts of Hadoop
2. To learn and understand the Hadoop framework for Big Data
3. To understand and practice installation and configuration of Hadoop.

**SOFTWARE REQUIREMENTS:**

1. Ubuntu 14.04 / 14.10
2. Java

**THEORY:**

**Introduction**

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Due to the advent of new technologies, devices, and communication like social networking sites, the amount of data produced by mankind is growing rapidly every year. The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes. The same amount was created in every two days in 2011, and in every ten minutes in 2013. This rate is still growing enormously. Though all this information produced is meaningful and can be useful when processed, it is being neglected

**Big** **Data**

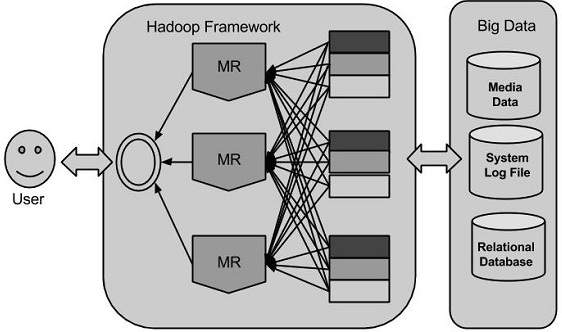
Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, technqiues and frameworks. Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

**Hadoop**

Doug Cutting, Mike Cafarella and team took the solution provided by Google and started an

Open Source Project called HADOOP in 2005 and Doug named it after his son's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.

Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capabale enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data.



Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

**HadoopArchitecture**

Hadoop framework includes following four modules:

 **Hadoop** **Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.

 **Hadoop** **YARN:** This is a framework for job scheduling and cluster resource management.

 **Hadoop** **Distributed** **File** **System** **(HDFS™):** A distributed file system that provides

high-throughput access to application data.

 **Hadoop** **MapReduce:** This is YARN-based system for parallel processing of large data sets.

**Installing Java**

Hadoop framework is written in Java!!

* Update the source list sunita@sunita:~$**sudo apt-get update**
* The OpenJDK project is the default version of Java
* that is provided from a supported Ubuntu repository.

sunita@sunita:~$sudo apt-get install default-jdk

sunita@sunita:~$**java -version** java version "1.7.0\_91"

OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-0ubuntu0.14.04.1) OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)

**Create User for Hadoop**

sunita@sunita:~$sudo addgroup hadoop

sunita@sunita:~$ sudo adduser --ingroup hadoop hduser

sunita@sunita:~$ sudo adduser hduser sudo

sunita@sunita:~$ sudo apt-get install openssh-server

sunita@sunita:~$ su – hduser

**Installing SSH (secure shell)**

**ssh** has two main components:

1. **ssh** : The command we use to connect to remote machines - the client.
2. **sshd** : The daemon that is running on the server and allows clients to connect to the server.

The **ssh** is pre-enabled on Linux, but in order to start **sshd** daemon, we need to install **ssh** first. Use this command to do that :

sunita@sunita:~$sudo apt-get install openssh-server

**Create and Setup SSH Certificates**

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost.

So, we need to have SSH up and running on our machine and configured it to allow SSH public key authentication.

Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands. If asked for a filename just leave it blank and press the enter key to continue.

sunita@sunita:~$**ssh-keygen -t rsa -P ""**

sunita@sunita:~$cat $HOME/.ssh/id\_rsa.pub >> $HOME/.ssh/authorized\_keys

The second command adds the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.

We can check if ssh works:

sunita@sunita:~$**ssh localhost**

**Install Hadoop**

sunita@sunita:~$**wget** http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.**3/hadoop-2.6.3.tar.gz**

sunita@sunita:~$**tar xvzf hadoop-2.6.3.tar.gz**

We want to move the Hadoop installation to the **/usr/local/hadoop** directory using the following command:

sunita@sunita:~$ sudo mv hadoop-2.9.0 /usr/local/hadoop

sunita@sunita:~$ sudo chown -R sunita /usr/local

**Setup Configuration Files**

The following files will have to be modified to complete the Hadoop setup:

* 1. ~/.bashrc
  2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh
  3. /usr/local/hadoop/etc/hadoop/core-site.xml
  4. /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
  5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

1. **~/.bashrc**:

Before editing the **.bashrc** file in our home directory, we need to find the path where Java has been installed to set the **JAVA\_HOME** environment variable using the following command:

Now we can append the following to the end of **~/.bashrc**:

sunita@sunita:~$**gedit .bashrc**

export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64

export HADOOP\_HOME=/usr/local/hadoop

export PATH=$PATH:$HADOOP\_HOME/bin

export PATH=$PATH:$HADOOP\_HOME/sbin

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export YARN\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib"

sunita@sunita:~$**source .bashrc**

This command applies the changes made in the .bashrc file.

**2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh**

We need to set **JAVA\_HOME** by modifying **hadoop-env.sh** file.

sunita@sunita:~$**gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh**

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64 Adding the above statement in the **hadoop-env.sh** file ensures that the value of JAVA\_HOME variable will be available to Hadoop whenever it is started up.

**3. /usr/local/hadoop/etc/hadoop/core-site.xml**:

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up. This file can be used to override the default settings that Hadoop starts with.

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

</property>

**4 sudo gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml**

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop\_tmp/hdfs/namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop\_tmp/hdfs/datanode</value>

</property>

5. sudo gedit /usr/local/hadoop/etc/hadoop/yarn-site.xml

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

6 sudo gedit /usr/local/hadoop/etc/hadoop/mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

sunita@sunita:~$sudo mkdir -p /usr/local/hadoop\_tmp

sunita@sunita:~$sudo mkdir -p sunita@sunita:~$/usr/local/hadoop\_tmp/hdfs/namenode

sunita@sunita:~$sudo mkdir -p sunita@sunita:~$/usr/local/hadoop\_tmp/hdfs/datanode

sunita@sunita:~$sudo chown -R hduser /usr/local/hadoop\_tmp

Open the file and enter the following content in between the

**Format the New Hadoop Filesystem**

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates **current** directory

under **/usr/local/hadoop\_store/hdfs/namenode** folder:

sunita@sunita:~$**hdfs namenode -format**

DEPRECATED: Use of this script to execute hdfs command is

Note that **hadoop namenode -format** command should be executed once before we start using Hadoop. If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system.

**Starting Hadoop**

Now it's time to start the newly installed single node cluster. We can use **start-all.sh** or (**start-dfs.sh** and **start-yarn.sh**)

sunita@sunita:~$**start-all.sh**

We can check if it's really up and running:

sunita@sunita:~$**jps**

9026 NodeManager

7348 NameNode

9766 Jps

8887 ResourceManager

7507 DataNode

The output means that we now have a functional instance of Hadoop running on our VPS (Virtual private server).

**Hadoop Web Interfaces**

Let's start the Hadoop again and see its Web UI:

**Accessing HADOOP through browser**

http://localhost:50070/

**Verify all applications for cluster**

http://localhost:8088/

**CONCLUSION:**

We studied installation of Hadoop installation and configuration.

**EXPERIMENT 2**

**TITLE: Design a distributed application using MapReduce**

**OBJECTIVE:**

1. To explore different Big data processing techniques with use cases.

2. To study detailed concept of Map-Reduced.

**SOFTWARE REQUIREMENTS:**

1. Ubuntu 14.04 / 14.10

2. GNU C Compiler

3. Hadoop

4. Java

**PROBLEM STATEMENT:-** Design a distributed application using MapReduce which processes a log file of a system. List out the users who have logged for maximum period on the system. Use simple log file from the Internet and process it using a pseudo distribution mode on Hadoop platform.

**THEORY:**

**Introduction**

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. 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 major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes.

Under the MapReduce model, the data processing primitives are called mappers and reducers.

Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

**The Algorithm**

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

**Mapstage :** 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.

**Inserting Data into HDFS:**

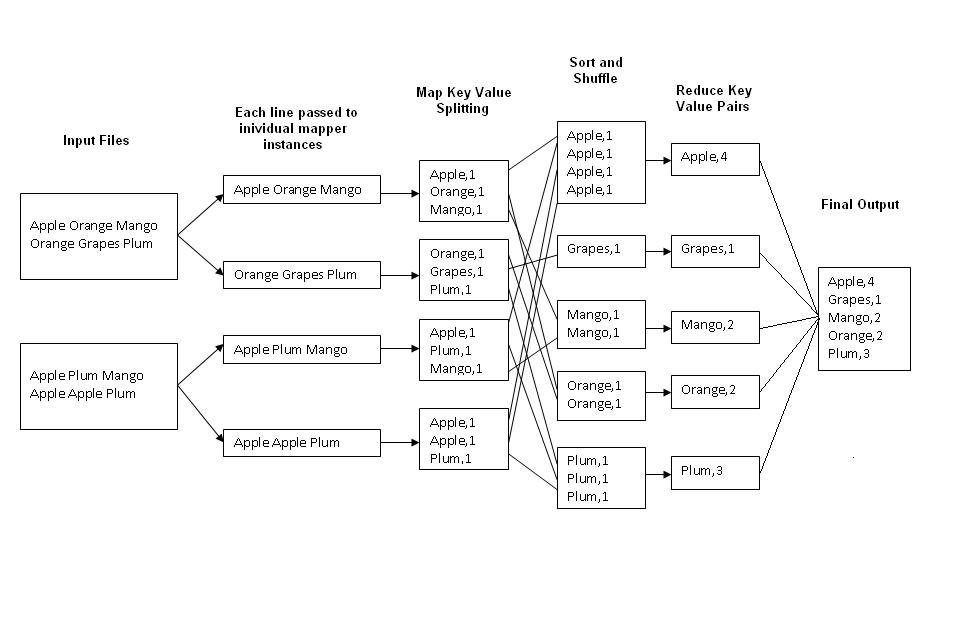


Fig.1 : An Example Program to Understand working of MapReduce Program.

•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).

**Steps for Compilation & Execution of Program:**

#sudo mkdir analyzelogs

ls

#sudo chmod -R 777 analyzelogs/

cd

ls

cd ..

pwd

ls

cd

pwd

#sudo chown -R hduser analyzelogs/

cd

ls

#cd analyzelogs/

ls

cd ..

**Copy the Files (Mapper.java,Reduce.java,Driver.java to Analyzelogs Folder)**

#sudo cp /home/mde/Desktop/count\_logged\_users/\* -/analyzelogs/

Start HADOOP

#start-dfs.sh

#start-yarn.sh

#jps

cd

cd analyzelogs

ls

pwd

ls

#ls -ltr

#ls -al

#sudo chmod +r \*.\*

pwd

#export CLASSPATH="$HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.9.0.jar:$HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.9.0.jar:$HADOOP\_HOME/share/hadoop/common/hadoop-common-2.9.0.jar:~/analyzelogs/SalesCountry/\*:$HADOOP\_HOME/lib/\*"

**Compile Java Files**

# javac -d . SalesMapper.java SalesCountryReducer.java SalesCountryDriver.java

ls

#cd SalesCountry/

ls

cd ..

#sudo gedit Manifest.txt

#jar -cfm analyzelogs.jar Manifest.txt SalesCountry/\*.class

ls

cd

jps

#cd analyzelogs/

**Create Directory on Hadoop**

#sudo mkdir ~/input2000

ls

pwd

#sudo cp access\_log\_short.csv ~/input2000/

# $HADOOP\_HOME/bin/hdfs dfs -put ~/input2000 /

# $HADOOP\_HOME/bin/hadoop jar analyzelogs.jar /input2000 /output2000

# $HADOOP\_HOME/bin/hdfs dfs -cat /output2000/part-00000

# stop-all.sh

# jps

**CONCLUSION:** Thus we have learnt how to design a distributed application using MapReduce and process a log file of a system.

**EXPERIMENT 3**

**TITLE: Design a distributed application using MapReduce**

**OBJECTIVE:**

1. To explore different Big data processing techniques with use cases.

2. To study detailed concept of Map-Reduced.

**SOFTWARE REQUIREMENTS:**

1. Ubuntu 14.04 / 14.10

2. GNU C Compiler

3. Hadoop

4. Java

**PROBLEM STATEMENT: -** Design and develop a distributed application to find the coolest/hottest year from the available weather data. Use weather data from the Internet and process it using MapReduce.

**THEORY:**

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. 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 major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

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.



**Map Function –**It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).

**Example –**(Map function in Word Count)

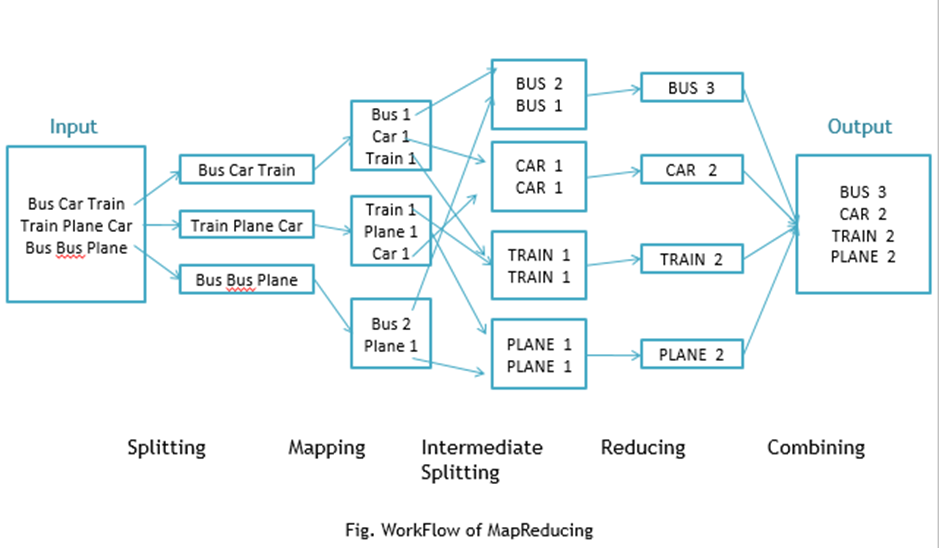
|  |  |  |
| --- | --- | --- |
| **Input** | Set of data | Bus, Car, bus,  car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN |
| **Output** | Convert into another set of data  (Key,Value) | (Bus,1), (Car,1), (bus,1), (car,1), (train,1),  (car,1), (bus,1), (car,1), (train,1), (bus,1),  (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1),  (car,1), (BUS,1), (TRAIN,1) |

 **Reduce Function –**Takes the output from Map as an input and combines those data tuples into a smaller set of tuples.

**Example –**(Reduce function in Word Count)

|  |  |  |
| --- | --- | --- |
| **Input**  **(output of Map function)** | Set of Tuples | (Bus,1), (Car,1), (bus,1), (car,1), (train,1),  (car,1), (bus,1), (car,1), (train,1), (bus,1),  (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1),  (car,1), (BUS,1), (TRAIN,1) |
| **Output** | Converts into smaller set of tuples | (BUS,7),  (CAR,7),  (TRAIN,4) |

### Work Flow of Program



Workflow of MapReduce consists of 5 steps

1. **Splitting** – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line (‘\n’).
2. **Mapping** – as explained above
3. **Intermediate splitting** – the entire process in parallel on different clusters. In order to group them in “Reduce Phase” the similar KEY data should be on same cluster.
4. **Reduce** – it is nothing but mostly group by phase

**Combining** – The last phase where all the data (individual result set from each cluster) is combine together to form a Result**.**

**CONCLUSION:** Thus we have learnt how to design a distributed application using MapReduce and process a Dataset.

**EXPERIMENT 4**

**TITLE: Write an application using HBase and HiveQL for flight information system**

**OBJECTIVE:**

1. To learn NoSQL Databases (Open source) such as Hive/ Hbase

2. To study detailed concept HIVE .

**SOFTWARE REQUIREMENTS:**

1. Ubuntu 14.04 / 14.10

2. GNU C Compiler

3. Hadoop

4. Java

5. HIVE

**PROBLEM STATEMENT: -** Write an application using HBase and HiveQL for flight information system which will include

1) Creating, Dropping, and altering Database tables

2) Creating an external Hive table to connect to the HBase for Customer Information Table

3) Load table with data, insert new values and field in the table, Join tables with Hive

4) Create index on Flight information Table

5) Find the average departure delay per day in 2008.

**THEORY:**

Hive:

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

# Prerequisites: Core Java,

# Database concepts of SQL,

# Hadoop File system, and any of Linux operating system

## Features:

* It stores schema in a database and processed data into HDFS.
* It is designed for OLAP.
* It provides SQL type language for querying called HiveQL or HQL.
* It is familiar, fast, scalable, and extensible.

## Architecture:

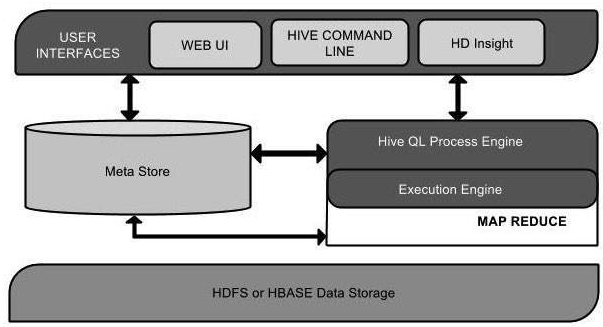


Figure 1: Hive Architecture

This component diagram contains different units. The following table describes each unit:

|  |  |
| --- | --- |
| Unit Name | Operation |
| User Interface | Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server). |
| Meta Store | Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping. |
| HiveQL Process Engine | HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it. |
| Execution Engine | The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce. |
| HDFS or HBASE | Hadoop distributed file system or HBASE are the data storage techniques to store data into file system. |

Table 1: Hive components and their operations.

## Verifying Hadoop Installation :

Hadoop must be installed on system before installing Hive. Verify the Hadoop installation using the following command:

$ hadoop version

If Hadoop is not installed on your system, then download and configure the Hadoop.

## Installing Hive:

The following steps are required for installing Hive.

### Extracting and verifying Hive Archive

The following command is used to verify the download and extract the hive archive:

$ tar zxvf apache-hive-0.14.0-bin.tar.gz

$ ls

On successful download, we get the following response:

apache-hive-0.14.0-bin apache-hive-0.14.0-bin.tar.gz

### Copying files to /usr/local/hive directory

We need to copy the files from the super user “su -”. The following commands are used to copy the files from the extracted directory to the /usr/local/hive” directory.

$ su -

passwd:

# cd /home/user/Download

# mv apache-hive-0.14.0-bin /usr/local/hive

# exit

### Setting up environment for Hive

We can set up the Hive environment by appending the following lines to~/.bashrc file:

export HIVE\_HOME=/usr/local/hive

export PATH=$PATH:$HIVE\_HOME/bin

export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/\*:.

export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/\*:.

The following command is used to execute ~/.bashrc file.

$ source ~/.bashrc

## Configuring Hive:

To configure Hive with Hadoop, we need to edit the hive-env.sh file, which is placed in the $HIVE\_HOME/conf directory. The following commands redirect to Hive config folder and copy the template file:

$ cd $HIVE\_HOME/conf

$ cp hive-env.sh.template hive-env.sh

Edit the hive-env.sh file by appending the following line:

export HADOOP\_HOME=/usr/local/hadoop

Hive installation is completed successfully.

* Creating Database:

Hive is a database technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it.

## Create Database Statement

Create Database is a statement used to create a database in Hive. A database in Hive is a namespace or a collection of tables. The syntax for this statement is as follows:

CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command. The following query is executed to create a database named userdb:

hive> CREATE DATABASE [IF NOT EXISTS] userdb;

or

hive> CREATE SCHEMA userdb;

The following query is used to verify a databases list:

hive> SHOW DATABASES;

default

userdb

## Drop Database Statement

Drop Database is a statement that drops all the tables and deletes the database. Its syntax is as follows:

DROP DATABASE StatementDROP (DATABASE|SCHEMA) [IF EXISTS] database\_name

[RESTRICT|CASCADE];

The following queries are used to drop a database. Let us assume that the database name is userdb.

hive> DROP DATABASE IF EXISTS userdb;

The following query drops the database using CASCADE. It means dropping respective tables before dropping the database.

hive> DROP DATABASE IF EXISTS userdb CASCADE;

The following query drops the database using SCHEMA.

hive> DROP SCHEMA userdb;

## Create Table Statement

Create Table is a statement used to create a table in Hive.

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db\_name.] table\_name

[(col\_name data\_type [COMMENT col\_comment], ...)]

[COMMENT table\_comment]

[ROW FORMAT row\_format]

[STORED AS file\_format]

## Load Data Statement

Generally, after creating a table in SQL, we can insert data using the Insert statement. But in Hive, we can insert data using the LOAD DATA statement. While inserting data into Hive, it is better to use LOAD DATA to store bulk records. There are two ways to load data: one is from local file system and second is from Hadoop file system.The syntax for load data is as follows:

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename

[PARTITION (partcol1=val1, partcol2=val2 ...)]

* LOCAL is identifier to specify the local path. It is optional.
* OVERWRITE is optional to overwrite the data in the table.
* PARTITION is optional.

## Alter Table Statement

It is used to alter a table in Hive.The statement takes any of the following syntaxes based on what attributes we wish to modify in a table.

ALTER TABLE name RENAME TO new\_name

ALTER TABLE name ADD COLUMNS (col\_spec[, col\_spec ...])

ALTER TABLE name DROP [COLUMN] column\_name

ALTER TABLE name CHANGE column\_name new\_name new\_type

ALTER TABLE name REPLACE COLUMNS (col\_spec[, col\_spec ...])

## Drop Table Statement

The syntax is as follows:

DROP TABLE [IF EXISTS] table\_name;

On successful execution of the query, you get to see the following response:

OK

Time taken: 5.3 seconds

hive>

The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore. SELECT statement is used to retrieve the data from a table. WHERE clause works similar to a condition. It filters the data using the condition and gives a finite result. The built-in operators and functions generate an expression, which fulfils the condition.Below is the syntax of the SELECT query:

SELECT [ALL | DISTINCT] select\_expr, select\_expr, ...

FROM table\_reference

[WHERE where\_condition]

[GROUP BY col\_list]

[HAVING having\_condition]

[CLUSTER BY col\_list | [DISTRIBUTE BY col\_list] [SORT BY col\_list]]

**CONCLUSION:** Thus we have learnt how to design a application HBase and HiveQL for flight information system.

**EXPERIMENT 5**

**TITLE: Write an Perform the following operations using R/Python on the Amazon book review and facebook metrics data sets**

**OBJECTIVE:**

1. To understand and apply the Analytical concept of Big data using R/Python.

2. To study detailed concept R .

**SOFTWARE REQUIREMENTS:**

1. Ubuntu 14.04 / 14.10

2. GNU C Compiler

3. Hadoop

4. Java

5. R Studio

**PROBLEM STATEMENT:** Perform the following operations using R/Python on the Amazon book review and facebook metrics data sets

1) Create data subsets

2) Merge Data

3) Sort Data

4) Transposing Data

5) Melting Data to long format

6) Casting data to wide format

**THEORY:**

R was initially written by **Ross Ihaka** and **Robert Gentleman** at the Department of Statistics of the University of Auckland in Auckland, New Zealand. R made its first appearance in 1993.

* A large group of individuals has contributed to R by sending code and bug reports.
* Since mid-1997 there has been a core group (the "R Core Team") who can modify the R source code archive

## Features of R

As stated earlier, R is a programming language and software environment for statistical analysis, graphics representation and reporting. The following are the important features of R −

* R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
* R has an effective data handling and storage facility,
* R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
* R provides a large, coherent and integrated collection of tools for data analysis.
* R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

## R - Data Reshaping

Data Reshaping in R is about changing the way data is organized into rows and columns. Most of the time data processing in R is done by taking the input data as a data frame. It is easy to extract data from the rows and columns of a data frame but there are situations when we need the data frame in a format that is different from format in which we received it. R has many functions to split, merge and change the rows to columns and vice-versa in a data frame.

1. **Joining Columns and Rows in a Data Frame**

We can join multiple vectors to create a data frame using the **cbind()**function. Also we can merge two data frames using **rbind()** function.

> # make two vectors and combine them as columns in a data.frame

> sport <- **c**("Hockey", "Baseball", "Football")

> league <- **c**("NHL", "MLB", "NFL")

> trophy <- **c**("Stanley Cup", "Commissioner's Trophy",

+ "Vince Lombardi Trophy")

> trophies1 <- **cbind**(sport, league, trophy)

> # make another data.frame using data.frame()

> trophies2 <- **data.frame**(sport=**c**("Basketball", "Golf"),

+ league=**c**("NBA", "PGA"),

+ trophy=**c**("Larry O'Brien Championship Trophy",

+ "Wanamaker Trophy"),

+ stringsAsFactors=FALSE)

> # combine them into one data.frame with rbind

> trophies <- **rbind**(trophies1, trophies2)

## Merging Data Frames

We can merge two data frames by using the **merge()** function. The data frames must have same column names on which the merging happens.

**Adding Columns**

To merge two data frames (datasets) horizontally, use the **merge** function. In most cases, you join two data frames by one or more common key variables (i.e., an inner join).

# merge two data frames by ID  
total <- merge(data frameA,data frameB,by="ID")

# merge two data frames by ID and Country  
total <- merge(data frameA,data frameB,by=c("ID","Country"))

**Adding Rows**

To join two data frames (datasets) vertically, use the **rbind** function. The two data frames **must** have the same variables, but they do not have to be in the same order.

total <- rbind(data frameA, data frameB)

1. **Subset**

Subsetting Vectors, Matrices and Data Frames

Return subsets of vectors, matrices or data frames which meet conditions.

Usage

subset(x, …)

# S3 method for default

subset(x, subset, …)

# S3 method for matrix

subset(x, subset, select, drop = FALSE, …)

# S3 method for data.frame

subset(x, subset, select, drop = FALSE, …)

Arguments

x object to be subsetted.

subset logical expression indicating elements or rows to keep: missing values are taken as false.

select expression, indicating columns to select from a data frame.

drop passed on to [ indexing operator.

Details

This is a generic function, with methods supplied for matrices, data frames and vectors (including lists). Packages and users can add further methods.

For ordinary vectors, the result is simply x[subset & !is.na(subset)].

For data frames, the subset argument works on the rows. Note that subset will be evaluated in the data frame, so columns can be referred to (by name) as variables in the expression (see the examples).

The select argument exists only for the methods for data frames and matrices. It works by first replacing column names in the selection expression with the corresponding column numbers in the data frame and then using the resulting integer vector to index the columns. This allows the use of the standard indexing conventions so that for example ranges of columns can be specified easily, or single columns can be dropped (see the examples).

The drop argument is passed on to the indexing method for matrices and data frames: note that the default for matrices is different from that for indexing.

Factors may have empty levels after subsetting; unused levels are not automatically removed. See droplevels for a way to drop all unused levels from a data frame.

e.g subset(airquality, Temp > 80, select = c(Ozone, Temp))

1. **Melting and Casting**

One of the most interesting aspects of R programming is about changing the shape of the data in multiple steps to get a desired shape. The functions used to do this are called **melt()** and **cast()**.

We consider the dataset called ships present in the library called "MASS".

library(MASS)

print(ships)

When we execute the above code, it produces the following result −

type year period service incidents

1 A 60 60 127 0

2 A 60 75 63 0

3 A 65 60 1095 3

4 A 65 75 1095 4

5 A 70 60 1512 6

## Melt the Data

Now we melt the data to organize it, converting all columns other than type and year into multiple rows.

molten.ships <- melt(ships, id = c("type","year"))

print(molten.ships)

When we execute the above code, it produces the following result −

type year variable value

1 A 60 period 60

2 A 60 period 75

3 A 65 period 60

4 A 65 period 75

## Cast the Molten Data

We can cast the molten data into a new form where the aggregate of each type of ship for each year is created. It is done using the **cast()** function.

recasted.ship <- cast(molten.ships, type+year~variable,sum)

print(recasted.ship)

When we execute the above code, it produces the following result −

type year period service incidents

1 A 60 135 190 0

2 A 65 135 2190 7

3 A 70 135 4865 24

4 A 75 135 2244 11

5 B 60 135 62058 68

6 B 65 135 48979 111

7 B 70 135 20163 56

8 B 75 135 7117 18

9 C 60 135 1731 2

## Transpose

Use the t() function to transpose a matrix or a data frame. In the later case, rownames become variable (column) names.

# example using built-in dataset   
mtcars  
t(mtcars)

1. **Sorting Data**

To sort a data frame in R, use the **order( )** function. By default, sorting is ASCENDING. Prepend the sorting variable by a minus sign to indicate DESCENDING order. Here are some examples.

# sorting examples using the mtcars dataset  
attach(mtcars)  
  
# sort by mpg  
newdata <- mtcars[order(mpg),]   
  
# sort by mpg and cyl  
newdata <- mtcars[order(mpg, cyl),]  
  
#sort by mpg (ascending) and cyl (descending)  
newdata <- mtcars[order(mpg, -cyl),]   
  
detach(mtcars)

**CONCLUSION:** Thus we have learnt how to Perform the different reshape operations using R **.**

**EXPERIMENT 6**

**TITLE: - Perform the following operations using R/Python on the Air quality and Heart Diseases data sets**

**OBJECTIVE:**

1. To understand and apply the Analytical concept of Big data using R/Python.

2. To study detailed concept R .

**SOFTWARE REQUIREMENTS:**

1. Ubuntu 14.04 / 14.10

2. GNU C Compiler

3. Hadoop

4. Java

5. R Studio

**PROBLEM STATEMENT:** Perform the following operations using R/Python on the Air quality and Heart Diseases data sets

1) Data cleaning

2) Data integration

3) Data transformation

4) Error correcting

5) Data model building

**THEORY:**

Data cleaning, or data preparation is an essential part of statistical analysis. In fact,

in practice it is often more time-consuming than the statistical analysis itself

**1) Data cleaning & Error correcting**

Applying NA Option

If there are missing values, then the mean function returns NA.

To drop the missing values from the calculation use na.rm=TRUE. which means remove the NA values.

# Create a vector.

x <-c(12,7,3,4.2,18,2,54,-21,8,-5,NA)

# Find mean.

result.mean <-mean(x)

print(result.mean)

# Find mean dropping NA values.

result.mean <-mean(x,na.rm=TRUE)

print(result.mean)

When we execute the above code, it produces the following result

:

[1] NA

[1] 8.22

**2) Data integration**

We can join multiple vectors to create a data frame using the **cbind()**function. Also we can merge two data frames using **rbind()** function.

> # make two vectors and combine them as columns in a data.frame

> sport <- **c**("Hockey", "Baseball", "Football")

> league <- **c**("NHL", "MLB", "NFL")

> trophy <- **c**("Stanley Cup", "Commissioner's Trophy",

+ "Vince Lombardi Trophy")

> trophies1 <- **cbind**(sport, league, trophy)

> # make another data.frame using data.frame()

> trophies2 <- **data.frame**(sport=**c**("Basketball", "Golf"),

+ league=**c**("NBA", "PGA"),

+ trophy=**c**("Larry O'Brien Championship Trophy",

+ "Wanamaker Trophy"),

+ stringsAsFactors=FALSE)

> # combine them into one data.frame with rbind

> trophies <- **rbind**(trophies1, trophies2)

**3) Data transformation**

> head(air)

Ozone Solar.R Wind Temp Month Day

1 41.0 190 7.4 67 5 1

2 36.0 118 8.0 72 5 2

3 12.0 149 12.6 74 5 3

4 18.0 313 11.5 62 5 4

5 31.5 205 14.3 56 5 5

6 28.0 205 14.9 66 5 6

> air$Solar.Danger = air$Solar.R > 100

> head(air)

Ozone Solar.R Wind Temp Month Day Solar.Danger

1 41.0 190 7.4 67 5 1 TRUE

2 36.0 118 8.0 72 5 2 TRUE

3 12.0 149 12.6 74 5 3 TRUE

4 18.0 313 11.5 62 5 4 TRUE

5 31.5 205 14.3 56 5 5 TRUE

6 28.0 205 14.9 66 5 6 TRUE

> brks = c(0,50,100,150,200,250,300,350)

> air$Solar.R = cut(air$Solar.R,breaks=brks,include.lowest = TRUE)

> head(air)

Ozone Solar.R Wind Temp Month Day Solar.Danger

1 41.0 (150,200] 7.4 67 5 1 TRUE

2 36.0 (100,150] 8.0 72 5 2 TRUE

3 12.0 (100,150] 12.6 74 5 3 TRUE

4 18.0 (300,350] 11.5 62 5 4 TRUE

5 31.5 (200,250] 14.3 56 5 5 TRUE

6 28.0 (200,250] 14.9 66 5 6 TRUE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

> air1 = air

> air1$Month = gsub(5,"May",air1$Month)

> head(air1)

Ozone Solar.R Wind Temp Month Day Solar.Danger

1 41.0 (150,200] 7.4 67 May 1 TRUE

2 36.0 (100,150] 8.0 72 May 2 TRUE

3 12.0 (100,150] 12.6 74 May 3 TRUE

4 18.0 (300,350] 11.5 62 May 4 TRUE

5 31.5 (200,250] 14.3 56 May 5 TRUE

6 28.0 (200,250] 14.9 66 May 6 TRUE

> air1$Month = gsub(6,"june",air1$Month)

> air1$Month = gsub(7,"july",air1$Month)

> air1$Month = gsub(8,"August",air1$Month)

> air1$Month = gsub(9,"September",air1$Month)

> head(air1)

Ozone Solar.R Wind Temp Month Day Solar.Danger

1 41.0 (150,200] 7.4 67 May 1 TRUE

2 36.0 (100,150] 8.0 72 May 2 TRUE

3 12.0 (100,150] 12.6 74 May 3 TRUE

4 18.0 (300,350] 11.5 62 May 4 TRUE

5 31.5 (200,250] 14.3 56 May 5 TRUE

6 28.0 (200,250] 14.9 66 May 6 TRUE

**4) Data model building**

# Step1 : Divide the dataset into taining and Testing

library(caTools)

hdata[, c(1)] <- sapply(hdata[, c(1)], as.numeric)

set.seed(123)

split = sample.split(hdata$num, SplitRatio = 2/3)

train\_hdata = subset(hdata, split == TRUE)

test\_hdata = subset(hdata, split == FALSE)

#You can use following code for creating training and testing samples,

#but you cannot get random samples which is possible with above split and subset

function

# train\_hdata=hdata[1:212,]

# test\_hdata=hdata[213:303,]

dim(train\_hdata)

#[1] 212 14

dim(test\_hdata)

#[1] 91 14

# Step 2: Use prediction Model using any of the technique-like regression,

classification and clustering

# here I have used Technique 1-Linear regression, 2-Multiple regression, 3-kNN,

4-Naive Bayes technique for prediction

# Technique 1: Linear regression

# Here for hear disease dataset, Variable age is IV and num is IV for linear

regression model

# fitting simple linear Regression to the training set

library(caTools)

regressor=lm(formula = num~age, data=train\_hdata)

#predicting the test set result using regressor

hd\_age\_predict=predict(regressor, newdata=test\_hdata)

# As the result is not whole number, rounding the result

round\_age=hd\_age\_predict

rage=round(round\_age)

# Displaying the accuracy using confusion Matrix

library(e1071)

library(caret)

df=confusionMatrix(rage,test\_hdata$num)

# Confusion Matrix and Statistics

# Reference

# Prediction 0 1

# 0 35 20

# 1 20 26

#

# Accuracy : 0.604

# 95% CI : (0.5017, 0.6999)

# No Information Rate : 0.5446

# P-Value [Acc > NIR] : 0.1357

#

# Kappa : 0.2016

# Mcnemar's Test P-Value : 1.0000

#

# Sensitivity : 0.6364

# Specificity : 0.5652

# Pos Pred Value : 0.6364

# Neg Pred Value : 0.5652

# Prevalence : 0.5446

# Detection Rate : 0.3465

# Detection Prevalence : 0.5446

# Balanced Accuracy : 0.6008

#

# 'Positive' Class : 0

**CONCLUSION:** Thus we have learnt how to Perform the different Data Cleaning and Data modeling operations using R **.**

**EXPERIMENT 7**

**TITLE: Integrate R/Python and Hadoop and perform the following operations on forest fire dataset**

**OBJECTIVE:**

1. To understand and apply the Analytical concept of Big data using R/Python.

2. To study detailed concept RHadoop .

**SOFTWARE REQUIREMENTS:**

1. Ubuntu 14.04 / 14.10

2. GNU C Compiler

3. Hadoop

4. Java

5. R Studio

**PROBLEM STATEMENT:** Integrate R/Python and Hadoop and perform the following operations on forest fire dataset

1) Text mining in RHadoop

2) Data analysis using the Map Reduce in Rhadoop

3) Data mining in Hive

**THEORY:**

**EXPERIMENT 8**

**TITLE: Visualize the data using R/Python by plotting the graphs for assignment no. 6 and 7**

**OBJECTIVE:**

1. To understand and apply the Analytical concept of Big data using R/Python.

2. To study detailed concept R .

**SOFTWARE REQUIREMENTS:**

1. Ubuntu 14.04 / 14.10

2. GNU C Compiler

3. Hadoop

4. Java

5. R Studio

**PROBLEM STATEMENT:** Visualize the data using R/Python by plotting the graphs for assignment no. 6 and 7

**THEORY:**

# R - Pie Charts

In R the pie chart is created using the **pie()** function which takes positive numbers as a vector input. The additional parameters are used to control labels, color, title etc.

## Syntax

The basic syntax for creating a pie-chart using the R is −

pie(x, labels, radius, main, col, clockwise)

Following is the description of the parameters used −

* **x** is a vector containing the numeric values used in the pie chart.
* **labels** is used to give description to the slices.
* **radius** indicates the radius of the circle of the pie chart.(value between −1 and +1).
* **main** indicates the title of the chart.
* **col** indicates the color palette.
* **clockwise** is a logical value indicating if the slices are drawn clockwise or anti clockwise.

## Example

A very simple pie-chart is created using just the input vector and labels. The below script will create and save the pie chart in the current R working directory.

# Create data for the graph.

x <- c(21, 62, 10, 53)

labels <- c("London", "New York", "Singapore", "Mumbai")

# Give the chart file a name.

png(file = "city.jpg")

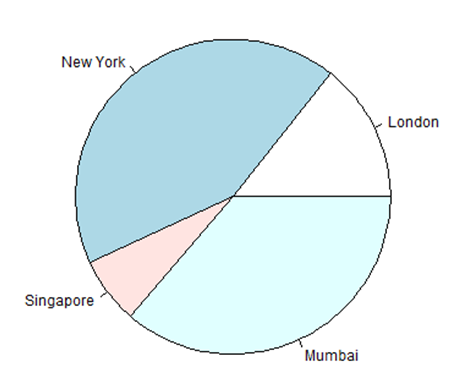
# Plot the chart.

pie(x,labels)

# Save the file.

dev.off()

When we execute the above code, it produces the following result −



# R - Bar Charts

A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function **barplot()** to create bar charts. R can draw both vertical and horizontal bars in the bar chart. In bar chart each of the bars can be given different colors.

## Syntax

The basic syntax to create a bar-chart in R is −

barplot(H, xlab, ylab, main, names.arg, col)

Following is the description of the parameters used −

* **H** is a vector or matrix containing numeric values used in bar chart.
* **xlab** is the label for x axis.
* **ylab** is the label for y axis.
* **main** is the title of the bar chart.
* **names.arg** is a vector of names appearing under each bar.
* **col** is used to give colors to the bars in the graph.

## Example

A simple bar chart is created using just the input vector and the name of each bar.

The below script will create and save the bar chart in the current R working directory.

# Create the data for the chart.

H <- c(7,12,28,3,41)

# Give the chart file a name.

png(file = "barchart.png")

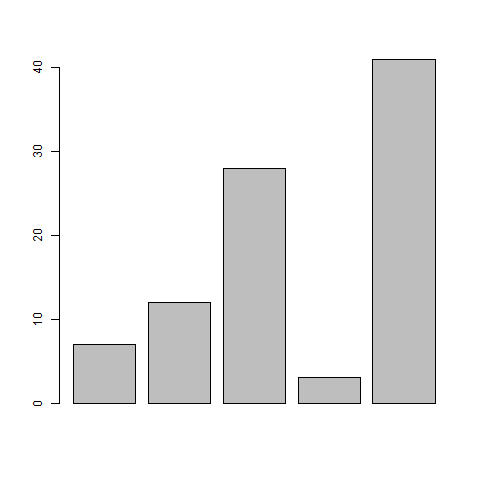
# Plot the bar chart.

barplot(H)

# Save the file.

dev.off()

When we execute the above code, it produces the following result −



# R – Boxplots

Boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.

Boxplots are created in R by using the **boxplot()** function.

## Syntax

The basic syntax to create a boxplot in R is −

boxplot(x, data, notch, varwidth, names, main)

Following is the description of the parameters used −

* **x** is a vector or a formula.
* **data** is the data frame.
* **notch** is a logical value. Set as TRUE to draw a notch.
* **varwidth** is a logical value. Set as true to draw width of the box proportionate to the sample size.
* **names** are the group labels which will be printed under each boxplot.
* **main** is used to give a title to the graph.

## Example

We use the data set "mtcars" available in the R environment to create a basic boxplot. Let's look at the columns "mpg" and "cyl" in mtcars.

input <- mtcars[,c('mpg','cyl')]

print(head(input))

When we execute above code, it produces following result −

mpg cyl

Mazda RX4 21.0 6

Mazda RX4 Wag 21.0 6

Datsun 710 22.8 4

Hornet 4 Drive 21.4 6

Hornet Sportabout 18.7 8

Valiant 18.1 6

## Creating the Boxplot

The below script will create a boxplot graph for the relation between mpg (miles per gallon) and cyl (number of cylinders).

# Give the chart file a name.

png(file = "boxplot.png")

# Plot the chart.

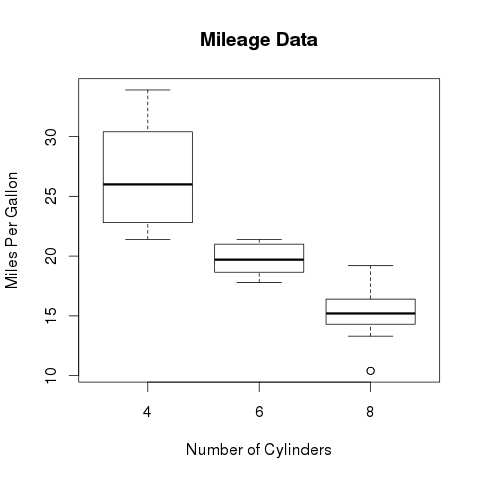
boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders",

ylab = "Miles Per Gallon", main = "Mileage Data")

# Save the file.

dev.off()

When we execute the above code, it produces the following result −



# R - Histograms

A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.

R creates histogram using **hist()** function. This function takes a vector as an input and uses some more parameters to plot histograms.

## Syntax

The basic syntax for creating a histogram using R is −

hist(v,main,xlab,xlim,ylim,breaks,col,border)

Following is the description of the parameters used −

* **v** is a vector containing numeric values used in histogram.
* **main** indicates title of the chart.
* **col** is used to set color of the bars.
* **border** is used to set border color of each bar.
* **xlab** is used to give description of x-axis.
* **xlim** is used to specify the range of values on the x-axis.
* **ylim** is used to specify the range of values on the y-axis.
* **breaks** is used to mention the width of each bar.

## Example

A simple histogram is created using input vector, label, col and border parameters.

The script given below will create and save the histogram in the current R working directory.

# Create data for the graph.

v <- c(9,13,21,8,36,22,12,41,31,33,19)

# Give the chart file a name.

png(file = "histogram.png")

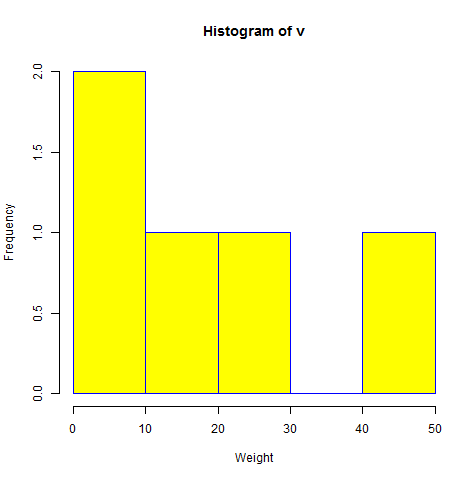
# Create the histogram.

hist(v,xlab = "Weight",col = "yellow",border = "blue")

# Save the file.

dev.off()

When we execute the above code, it produces the following result −



# R - Line Graphs

A line chart is a graph that connects a series of points by drawing line segments between them. These points are ordered in one of their coordinate (usually the x-coordinate) value. Line charts are usually used in identifying the trends in data.

The **plot()** function in R is used to create the line graph.

## Syntax

The basic syntax to create a line chart in R is −

plot(v,type,col,xlab,ylab)

Following is the description of the parameters used −

* **v** is a vector containing the numeric values.
* **type** takes the value "p" to draw only the points, "l" to draw only the lines and "o" to draw both points and lines.
* **xlab** is the label for x axis.
* **ylab** is the label for y axis.
* **main** is the Title of the chart.
* **col** is used to give colors to both the points and lines.

## Example

A simple line chart is created using the input vector and the type parameter as "O". The below script will create and save a line chart in the current R working directory.

# Create the data for the chart.

v <- c(7,12,28,3,41)

# Give the chart file a name.

png(file = "line\_chart.jpg")

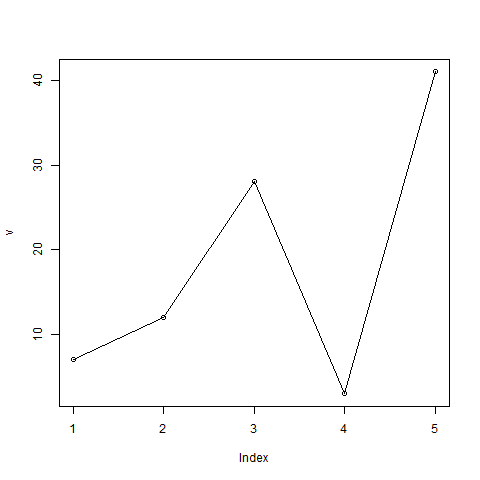
# Plot the bar chart.

plot(v,type = "o")

# Save the file.

dev.off()

When we execute the above code, it produces the following result −



# R – Scatterplots

Scatterplots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.

The simple scatterplot is created using the **plot()** function.

## Syntax

The basic syntax for creating scatterplot in R is −

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

Following is the description of the parameters used −

* **x** is the data set whose values are the horizontal coordinates.
* **y** is the data set whose values are the vertical coordinates.
* **main** is the tile of the graph.
* **xlab** is the label in the horizontal axis.
* **ylab** is the label in the vertical axis.
* **xlim** is the limits of the values of x used for plotting.
* **ylim** is the limits of the values of y used for plotting.
* **axes** indicates whether both axes should be drawn on the plot.

## Example

We use the data set **"mtcars"** available in the R environment to create a basic scatterplot. Let's use the columns "wt" and "mpg" in mtcars.

input <- mtcars[,c('wt','mpg')]

print(head(input))

When we execute the above code, it produces the following result −

wt mpg

Mazda RX4 2.620 21.0

Mazda RX4 Wag 2.875 21.0

Datsun 710 2.320 22.8

Hornet 4 Drive 3.215 21.4

Hornet Sportabout 3.440 18.7

Valiant 3.460 18.1

## Creating the Scatterplot

The below script will create a scatterplot graph for the relation between wt(weight) and mpg(miles per gallon).

# Get the input values.

input <- mtcars[,c('wt','mpg')]

# Give the chart file a name.

png(file = "scatterplot.png")

# Plot the chart for cars with weight between 2.5 to 5 and mileage between 15 and 30.

plot(x = input$wt,y = input$mpg,

xlab = "Weight",

ylab = "Milage",

xlim = c(2.5,5),

ylim = c(15,30),

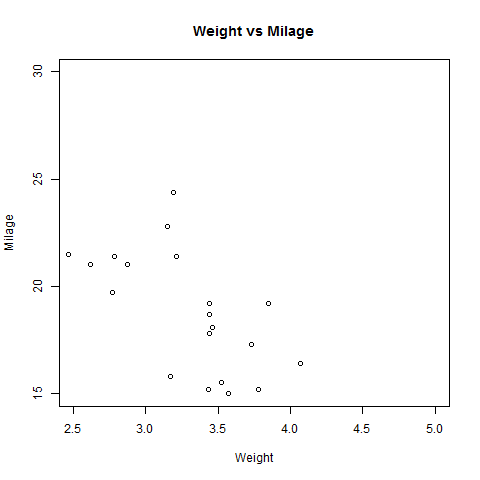
main = "Weight vs Milage"

)

# Save the file.

dev.off()

When we execute the above code, it produces the following result −



# Scatterplot Matrices

When we have more than two variables and we want to find the correlation between one variable versus the remaining ones we use scatterplot matrix. We use **pairs()** function to create matrices of scatterplots.

### Syntax

The basic syntax for creating scatterplot matrices in R is −

pairs(formula, data)

Following is the description of the parameters used −

* **formula** represents the series of variables used in pairs.
* **data** represents the data set from which the variables will be taken.

### Example

Each variable is paired up with each of the remaining variable. A scatterplot is plotted for each pair.

# Give the chart file a name.

png(file = "scatterplot\_matrices.png")

# Plot the matrices between 4 variables giving 12 plots.

# One variable with 3 others and total 4 variables.

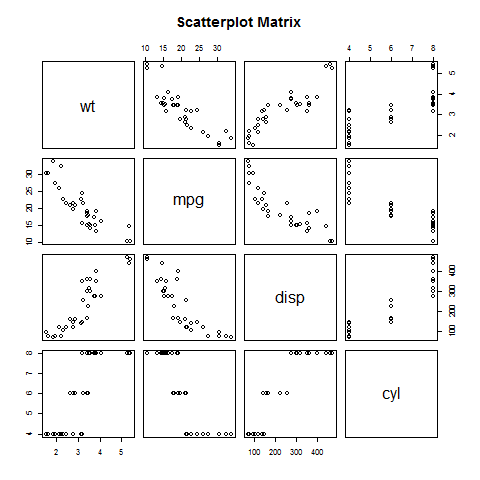
pairs(~wt+mpg+disp+cyl,data = mtcars,

main = "Scatterplot Matrix")

# Save the file.

dev.off()

When the above code is executed we get the following output.



**CONCLUSION:** Thus we have learnt Visualize the data using R/Python by plotting the graphs .

**EXPERIMENT 9**

**TITLE: Perform the following data visualization operations using Tableau on Adult and Iris datasets**

**OBJECTIVE:**

1. To understand and apply the Analytical concept of Big data using Tableau..

2. To study detailed concept of Tableau.

**SOFTWARE REQUIREMENTS:**

1. Ubuntu 14.04 / 14.10

2. GNU C Compiler

3. Hadoop

4. Java

5. Tableau

**PROBLEM STATEMENT:** Perform the following data visualization operations using Tableau on Adult and Iris datasets

1) 1D (Linear) Data visualization

2) 2D (Planar) Data Visualization

3) 3D (Volumetric) Data Visualization

4) Temporal Data Visualization

5) Multidimensional Data Visualization

6) Tree/ Hierarchical Data visualization

7) Network Data visualization

**THEORY:**

**Introduction**

Data visualization or data visualization is viewed by many disciplines as a modern equivalent of [visual communication](https://en.wikipedia.org/wiki/Visual_communication). It involves the creation and study of the [visual](https://en.wikipedia.org/wiki/Visual_system) representation of [data](https://en.wikipedia.org/wiki/Data), meaning "information that has been abstracted in some schematic form, including attributes or variables for the units of [information](https://en.wikipedia.org/wiki/Information)".

Data visualization refers to the techniques used to communicate data or information by encoding it as visual objects (e.g., points, lines or bars) contained in graphics. The goal is to communicate information clearly and efficiently to users. It is one of the steps in [data analysis](https://en.wikipedia.org/wiki/Data_analysis) or [data science](https://en.wikipedia.org/wiki/Data_science)

1D/Linear

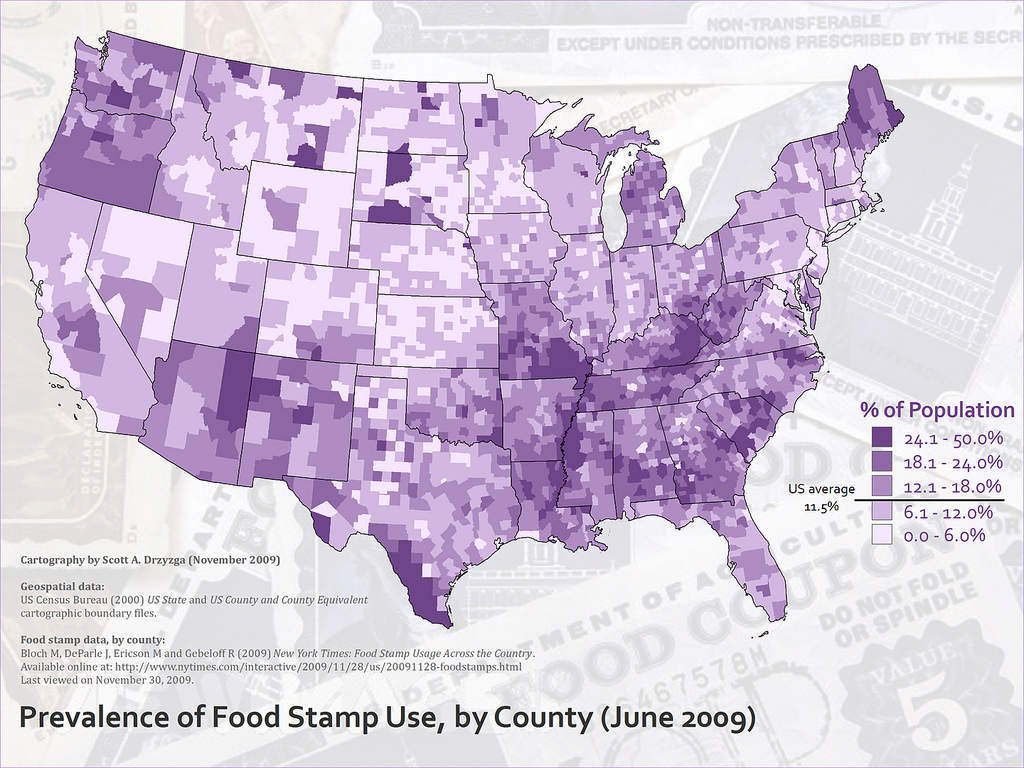
Examples:

* lists of data items, organized by a single feature (e.g., alphabetical order)  
  (not commonly visualized)

## 2D/Planar (especially geospatial)

Examples (geospatial):

* [**choropleth**](http://en.wikipedia.org/wiki/Choropleth)



## 3D/Volumetric

**3D/Volumetric**

Broadly, examples of [scientific visualization](http://en.wikipedia.org/wiki/Scientific_visualization):

* [3D computer models](http://en.wikipedia.org/wiki/3D_modelling)

In [3D computer graphics](https://en.wikipedia.org/wiki/3D_computer_graphics), **3D modeling** (or **three-dimensional modeling**) is the process of developing a mathematical representation of any [*surface*](https://en.wikipedia.org/wiki/Surface_(mathematics)) of an object (either inanimate or living) in [three dimensions](https://en.wikipedia.org/wiki/Three-dimensional_space) via [specialized software](https://en.wikipedia.org/wiki/3D_computer_graphics_software). The product is called a **3D model**. Someone who works with 3D models may be referred to as a **3D artist**. It can be displayed as a two-dimensional image through a process called [*3D rendering*](https://en.wikipedia.org/wiki/3D_rendering) or used in a [computer simulation](https://en.wikipedia.org/wiki/Computer_simulation) of physical phenomena. The model can also be physically created using [3D printing](https://en.wikipedia.org/wiki/3D_printing) devices.

* [surface](http://en.wikipedia.org/wiki/Scientific_visualization#Surface_rendering) and [volume](http://en.wikipedia.org/wiki/Scientific_visualization#Volume_rendering) rendering

[Rendering](https://en.wikipedia.org/wiki/Rendering_(computer_graphics)) is the process of generating an image from a [model](https://en.wikipedia.org/wiki/3D_model), by means of computer programs. The model is a description of three-dimensional objects in a strictly defined language or data structure. It would contain geometry, viewpoint, [texture](https://en.wikipedia.org/wiki/Texture_mapping), [lighting](https://en.wikipedia.org/wiki/Lighting), and shading information. The image is a [digital image](https://en.wikipedia.org/wiki/Digital_image) or [raster graphics](https://en.wikipedia.org/wiki/Raster_graphics) [image](https://en.wikipedia.org/wiki/Image). The term may be by analogy with an "artist's rendering" of a scene. 'Rendering' is also used to describe the process of calculating effects in a video editing file to produce final video output.

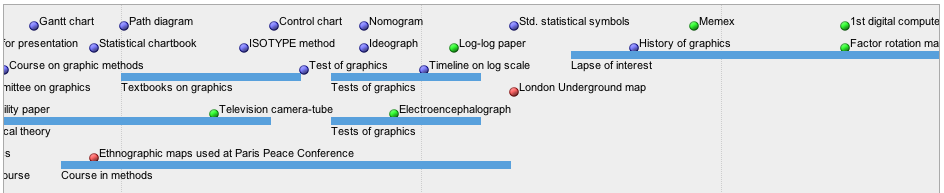
[Volume rendering](https://en.wikipedia.org/wiki/Volume_rendering) is a technique used to display a 2D projection of a 3D discretely [sampled](https://en.wikipedia.org/wiki/Sampling_(signal_processing)) [data set](https://en.wikipedia.org/wiki/Data_set). A typical 3D data set is a group of 2D slice images acquired by a [CT](https://en.wikipedia.org/wiki/Computed_axial_tomography) or [MRI](https://en.wikipedia.org/wiki/Magnetic_resonance_imaging) scanner. Usually these are acquired in a regular pattern (e.g., one slice every millimeter) and usually have a regular number of image [pixels](https://en.wikipedia.org/wiki/Pixel) in a regular pattern. This is an example of a regular volumetric grid, with each volume element, or [voxel](https://en.wikipedia.org/wiki/Voxel) represented by a single value that is obtained by sampling the immediate area surrounding the voxel.

* [computer simulations](http://en.wikipedia.org/wiki/Scientific_visualization#Computer_simulation)

[Computer simulation](https://en.wikipedia.org/wiki/Computer_simulation) is a computer program, or network of computers, that attempts to [simulate](https://en.wikipedia.org/wiki/Simulation) an abstract [model](https://en.wikipedia.org/wiki/Model_(abstract)) of a particular system. Computer simulations have become a useful part of [mathematical modeling](https://en.wikipedia.org/wiki/Mathematical_model) of many natural systems in physics, and computational physics, chemistry and biology; human systems in economics, psychology, and social science; and in the process of engineering and new technology, to gain insight into the operation of those systems, or to observe their behavior.[[6]](https://en.wikipedia.org/wiki/Scientific_visualization#cite_note-6) The simultaneous visualization and simulation of a system is called [visulation](https://en.wikipedia.org/wiki/Visulation" \o "Visulation).

## Temporal

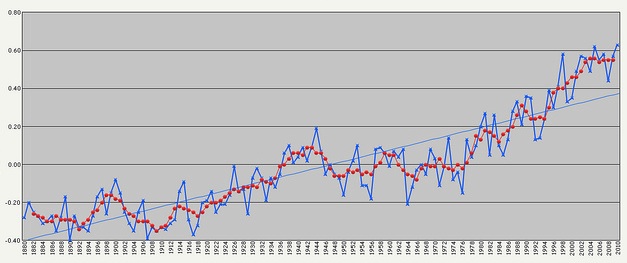
Examples:

* **[](https://lgimages.s3.amazonaws.com/data/imagemanager/62927/milestones-timeline.png)**[**timeline**](http://en.wikipedia.org/wiki/Timeline)

Tools: [SIMILE Timeline](http://www.simile-widgets.org/timeline/), [TimeFlow](https://github.com/FlowingMedia/TimeFlow/wiki/), [Timeline JS](http://timeline.verite.co/), [Excel](http://www.vertex42.com/ExcelArticles/create-a-timeline.html)

Image:  
Friendly, M. & Denis, D. J. (2001). Milestones in the history of thematic cartography, statistical graphics, and data visualization. Web document, <http://www.datavis.ca/milestones/>. Accessed: August 30, 2012.

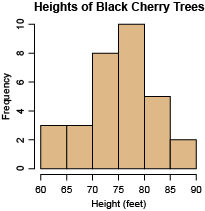
* [**time series**](http://en.wikipedia.org/wiki/Time_series)

[](https://lgimages.s3.amazonaws.com/data/imagemanager/62927/flickr_zcopley_7513082668_cropped.jpg)

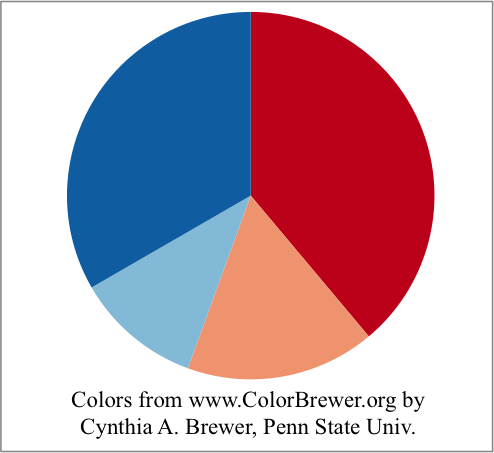
## nD/Multidimensional

Examples (category proportions, counts):

* [**histogram**](http://en.wikipedia.org/wiki/Histogram)

**[](https://lgimages.s3.amazonaws.com/data/imagemanager/62927/black_cherry_tree_histogram.png)**

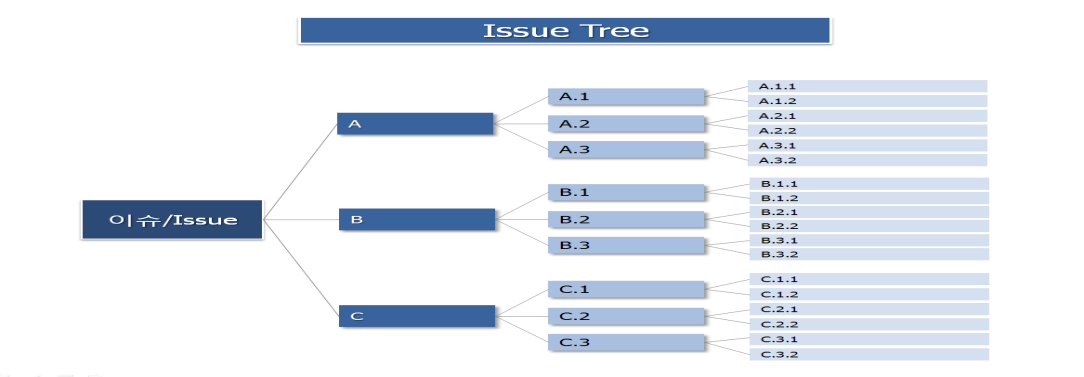
* [**pie chart**](http://en.wikipedia.org/wiki/Pie_chart)

[](https://lgimages.s3.amazonaws.com/data/imagemanager/62927/pie.png)

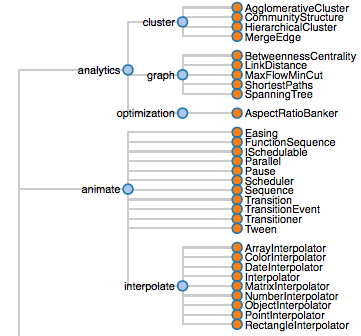
## Tree/Hierarchical

Examples:

* **general tree visualization**

[](https://lgimages.s3.amazonaws.com/data/imagemanager/62927/flickr_phploveme_2769634497.png)

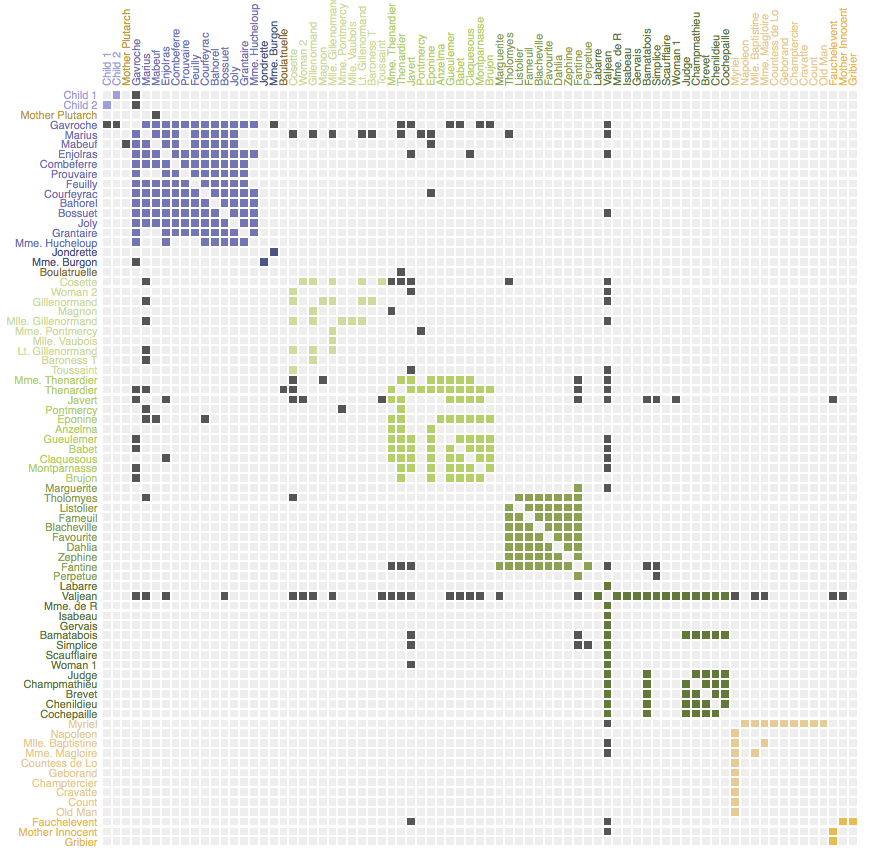
* [**dendrogram**](http://en.wikipedia.org/wiki/Dendrogram)

[](https://lgimages.s3.amazonaws.com/data/imagemanager/62927/dendrogram-protovis.png)

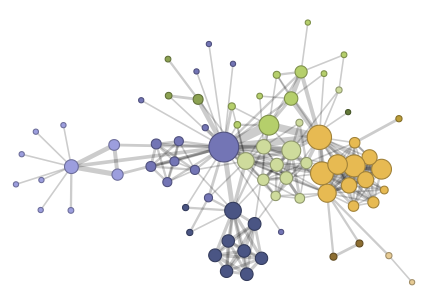
Network

Examples:

* **matrix**



* **node-link diagram (link-based layout algorithm)**



**Tableau:**

Tableau is a Business Intelligence tool for visually analyzing the data. Users can create and distribute an interactive and shareable dashboard, which depict the trends, variations, and density of the data in the form of graphs and charts. Tableau can connect to files, relational and Big Data sources to acquire and process data. The software allows data blending and real-time collaboration, which makes it very unique. It is used by businesses, academic researchers, and many government organizations for visual data analysis. It is also positioned as a leader Business Intelligence and Analytics Platform in Gartner Magic Quadrant.

**Tableau Features:**

Tableau provides solutions for all kinds of industries, departments, and data environments. Following are some unique features which enable Tableau to handle diverse scenarios.

* **Speed of Analysis** − As it does not require high level of programming expertise, any user with access to data can start using it to derive value from the data.
* **Self-Reliant** − Tableau does not need a complex software setup. The desktop version which is used by most users is easily installed and contains all the features needed to start and complete data analysis.
* **Visual Discovery** − The user explores and analyzes the data by using visual tools like colors, trend lines, charts, and graphs. There is very little script to be written as nearly everything is done by drag and drop.
* **Blend Diverse Data Sets** − Tableau allows you to blend different relational, semi structured and raw data sources in real time, without expensive up-front integration costs. The users don’t need to know the details of how data is stored.
* **Architecture Agnostic** − Tableau works in all kinds of devices where data flows. Hence, the user need not worry about specific hardware or software requirements to use Tableau.
* **Real-Time Collaboration** − Tableau can filter, sort, and discuss data on the fly and embed a live dashboard in portals like SharePoint site or Salesforce. You can save your view of data and allow colleagues to subscribe to your interactive dashboards so they see the very latest data just by refreshing their web browser.
* **Centralized Data** − Tableau server provides a centralized location to manage all of the organization’s published data sources. You can delete, change permissions, add tags, and manage schedules in one convenient location. It’s easy to schedule extract refreshes and manage them in the data server. Administrators can centrally define a schedule for extracts on the server for both incremental and full refreshes.

There are three basic steps involved in creating any Tableau data analysis report.

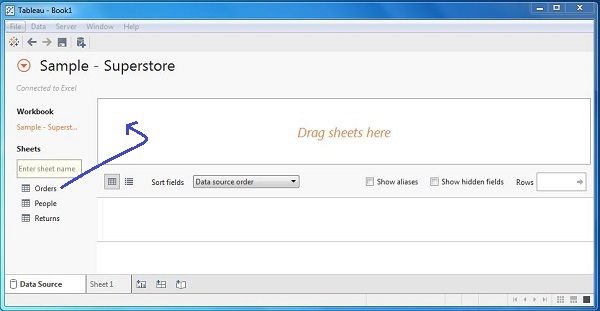
These three steps are −

* **Connect to a data source** − It involves locating the data and using an appropriate type of connection to read the data.
* **Choose dimensions and measures** − This involves selecting the required columns from the source data for analysis.
* **Apply visualization technique** − This involves applying required visualization methods, such as a specific chart or graph type to the data being analyzed.

For convenience, let’s use the sample data set that comes with Tableau installation named sample – superstore.xls. Locate the installation folder of Tableau and go to **My Tableau Repository**. Under it, you will find the above file at **Datasources\9.2\en\_US-US**.

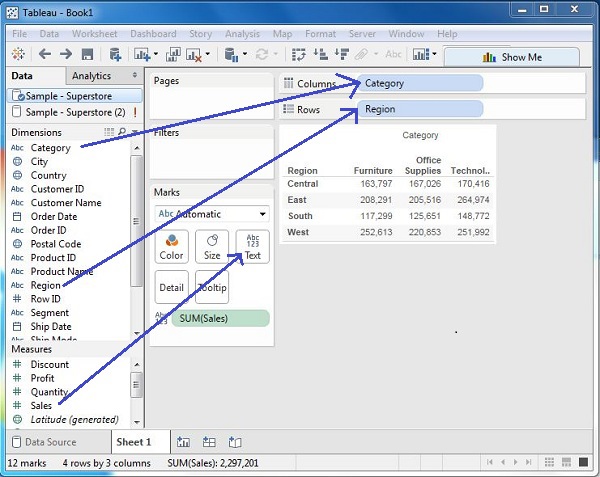
## Connect to a Data Source

On opening Tableau, you will get the start page showing various data sources. Under the header **“Connect”**, you have options to choose a file or server or saved data source. Under Files, choose excel. Then navigate to the file **“Sample – Superstore.xls”** as mentioned above. The excel file has three sheets named Orders, People and Returns. Choose **Orders**.



### Choose the Dimensions and Measures

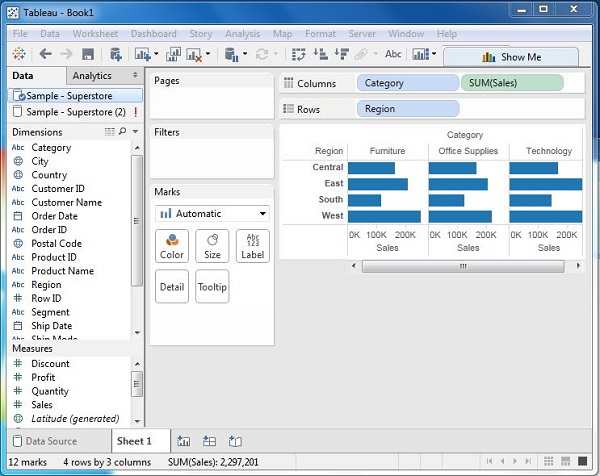
Next, choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help visualize the performance of the dimensional data with respect to the data which are measures. Choose **Category** and **Region** as the dimensions and **Sales** as the measure. Drag and drop them as shown in the following screenshot. The result shows the total sales in each category for each region.



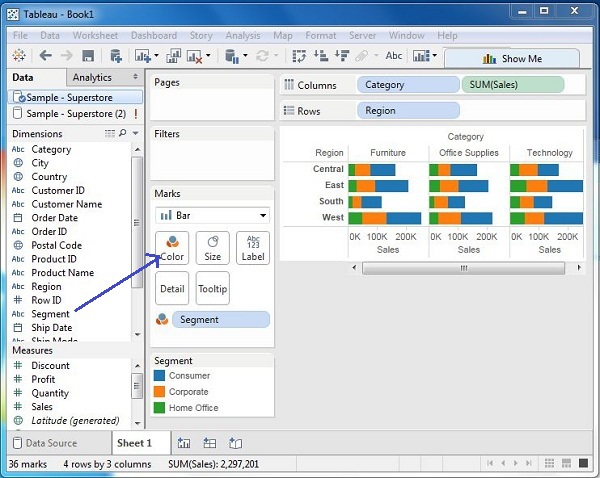
### Apply Visualization Technique

In the previous step, you can see that the data is available only as numbers. You have to read and calculate each of the values to judge the performance. However, you can see them as graphs or charts with different colors to make a quicker judgment.

We drag and drop the sum (sales) column from the Marks tab to the Columns shelf. The table showing the numeric values of sales now turns into a bar chart automatically.



You can apply a technique of adding another dimension to the existing data. This will add more colors to the existing bar chart as shown in the following screenshot.



**Conclusion:** Thus we have learnt how to Visualize the data in different types (1 1D (Linear) Data visualization,2D (Planar) Data Visualization, 3D (Volumetric) Data Visualization, Temporal Data Visualization,Multidimensional Data Visualization, Tree/ Hierarchical Data visualization, Network Data visualization) by using Tableau Software.

**Part C: Case Study Assignment**

**1) Social Media Analytics**

**2) Text Mining/ Text Analytics**

**3) Mobile Analytics**

***Note: 1. Students need to write about theory concepts on each topic and details about at least one tool such as source URL, Open source/Proprietary, installation steps, configurations, features and compatibility etc. with sample case study.***

***2. Single Student/Group can give presentation of 10/15 mins on the tool selected & Case study.***

1. **Social Media Analytics**

**E.g.**

**Tools:**

**Buffer,** [Followerwonk](https://followerwonk.com/analyze), [ViralWoot](http://viralwoot.com/), [Google Analytics](http://analytics.google.com/), [Quintly](https://www.quintly.com/), [Cyfe](http://www.cyfe.com/), [Tailwind](https://www.tailwindapp.com/), [Keyhole](http://keyhole.co/), [Klout](https://klout.com/), [Klear](https://klear.com/), [Audiense](https://audiense.com/twitter-analytics/), [TweetReach](https://tweetreach.com/), [IBM Watson Personality Insights](https://personality-insights-livedemo.mybluemix.net/), [Peakfeed](http://www.peakfeed.com/), [WolframAlpha Facebook Report](https://www.wolframalpha.com/input/?i=facebook+report), [SocialRank](https://socialrank.com/), [WEBSTA](https://websta.me/), [Talkwalker](https://www.talkwalker.com/social-media-analytics-search), [LikeAlyzer](http://likealyzer.com/),

**Dashboard:**

Facebook Insights, Instagram Insights, Twitter analytics, Pinterest analytics, LinkedIn analytics for individuals, LinkedIn analytics for businesses, Google+ Influence

1. **Text Mining/ Text Analytics**

Gate, Rapidminer, KH Coder, VisualTest, Datumbox, TAMS, QDA Miner lite, Carrot2, CAT, TM, GENSIM, NLTK, UIMA, Opennlp, KNIME, Apache Mahuot, LPU, Pattern

1. **Mobile Analytics**

Google Analytics, [iOS](http://www.mobyaffiliates.com/guides/mobile-analytics/) App Analytics, Flurry Analytics, Amazon Mobile Analytics , Tune, AppsFlyer, App Annie, Adjust, Kochava, Localytics, Tenjin, Appsumer, Mixpanel, Appsee, Amplitude, Branch, Singular, Leanplum, Urban Airship, [Appboy](http://www.mobyaffiliates.com/guides/mobile-analytics/), [Apptimize](http://www.mobyaffiliates.com/guides/mobile-analytics/), [GameAnalytics](http://www.mobyaffiliates.com/guides/mobile-analytics/), [Upsight](http://www.mobyaffiliates.com/guides/mobile-analytics/) , [SWRVE](http://www.mobyaffiliates.com/guides/mobile-analytics/), [Taplytics](http://www.mobyaffiliates.com/guides/mobile-analytics/), [Apptentive](http://www.mobyaffiliates.com/guides/mobile-analytics/)