PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2019-20

LAB MANUAL

DEPARTMENT: INFORMATION TECHNOLOGY

CLASS: T.E. **SEMESTER: VI**

Subject Name: Software Laboratory VI

INDEX OF LAB EXPERIMENTS

LAB EXPT. NO.	PROBLEM STATEMENT	LAST DATE OF COMPLETION
	Part A : Assignments based on the Hadoop	
1.	Study of Hadoop Installation. • Single Node • Multiple Node	**4TH WEEK OF DEC**
2.	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.	**1ST WEEK OF JAN**
3.	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.	**2ND WEEK OF JAN**
4.	Write an application using HBase and HiveQL for flight information system which will include • Creating, Dropping, and altering Database tables • Creating an external Hive table to connect to the HBase for Customer Information Table • Load table with data, insert new values and field in the table, Join tables with Hive • Create index on Flight information Table 5) Find the average departure delay per day in 2008.	**3RD WEEK OF JAN**
	Part B : Assignments based on R and Python	
5.	Perform the following operations using R/Python on the Amazon book review and facebook metrics data sets • Create data subsets • Merge Data • Sort Data • Transposing Data • Melting Data to long format • Casting data to wide format	**4TH WEEK OF JAN**

6.	Perform the following operations using R/Python on the Air quality and Heart Diseases data sets a. Data cleaning b. Data integration c. Data transformation d. Error correcting e. Data model building	**1ST WEEK OF FEB**
7.	Integrate R/Python and Hadoop and perform the following operations on forest fire dataset a. Text mining in RHadoop b. Data analysis using the Map Reduce in Rhadoop c. Data mining in Hive	**3RD WEEK OF FEB**
8.	Visualize the data using R/Python by plotting the graphs for assignment no. 6 and 7.	**4TH WEEK OF FEB**
9.	Perform the following data visualization operations using Tableau on Adult and Iris datasets a. 1D (Linear) Data visualization b. 2D (Planar) Data Visualization c. 3D (Volumetric) Data Visualization d. Temporal Data Visualization e. Multidimensional Data Visualization f. Tree/ Hierarchical Data visualization g. Network Data visualization	**1ST WEEK OF MAR**
	Part C : Case Study Assignments	
10.	Social Media Analytics	**2ND WEEK OF MAR**
11.	Text Mining/ Text Analytics	**3RD WEEK OF MAR**
12.	Mobile Analytics	**4TH WEEK OF MAR**
	Part D: Mini Project in Data Science	
13.	Design and implement any Data Science application using R/Python. Obtain data, scrub data (Data cleaning). Explore data, Prepare and validate data model and interpret data (Data visualization). Visualize data using any visualization tool like Matplotlib, ggplot, Tableau etc. Prepare Project Report.	**1ST WEEK OF APR**

R. B. Murumkar
Subject Coordinator **Head of Department**

Part A

Assignments based on the Hadoop

Assignment: 1

AIM: Study of Hadoop Installation.

PROBLEM STATEMENT / DEFINITION:

1. Study of Hadoop Installation on Single Node.
2. Study of Hadoop Installation on Multiple Nodes.

OBJECTIVE:

I. To understand Installation & Configuration of Hadoop on single Node.
II. To understand Installation & Configuration of Hadoop on multiple nodes.

THEORY:

a) THE Single Node:

Introduction

Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common and should be automatically handled by the framework. The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part called MapReduce. Hadoop splits files into large blocks and distributes them across nodes in a cluster. To process data, Hadoop transfers packaged code for nodes to process in parallel based on the data that needs to be processed. This approach takes advantage of data locality— nodes manipulating the data they have access to— to allow the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking. The base Apache Hadoop framework is composed of the following modules:

- **Hadoop Common** – contains libraries and utilities needed by other Hadoop modules;
- **Hadoop Distributed File System (HDFS)** – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- **Hadoop YARN** – a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications; and
- **Hadoop MapReduce** – an implementation of the MapReduce programming model for large scale data processing.

Steps:

Open Terminal

 # Update the source list
aaditya@laptop:~$ **sudo apt-get update**

The OpenJDK project is the default version of Java
Install JDK

aaditya@laptop:~$ **sudo apt-get install default-jdk`**

aaditya@laptop:~$ **java -version**
java version "1.7.0_95"

Adding a dedicated Hadoop user
aaditya@laptop:~$ **sudo addgroup hadoop**
Adding group `hadoop' (GID 1002) ...
Done.

aaditya@laptop:~$ **sudo adduser --ingroup hadoop hduser**
(Press Enter)

Is the information correct? [Y/n] Y

Enter new password for new hduser

aaditya@laptop:~$ **sudo adduser hduser sudo**
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.

Logout from current user and log in to hduser

Installing SSH

aaditya@laptop:~$ **su hduser**

hduser@laptop:~$ **sudo apt-get install ssh**

Create and Setup SSH Certificates

hduser@laptop:~$ **cd /home/hduser/**

hduser@laptop:~$ **ssh-keygen -t rsa -P ""**
Generating public/private rsa key pair.

Enter file in which to save the key (/home/hduser/.ssh/id_rsa): **(**press enter)**
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3 hduser@laptop

The key's randomart image is:
```
+--[ RSA 2048]----+
|        .oo.o    |
|       . .o=. o  |
|      . + .  o . |
|       o =    E  |
|        S +      |
|         . +     |
|          O +    |
|           O o   |
|            o..  |
+-----------------+
```

hduser@laptop:$ **cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/.ssh/authorized_keys**

hduser@laptop:$ **ssh localhost**

hduser@laptop:$ **exit**

hduser@laptop:$ **cd /home/hduser**

Download Hadoop

hduser@laptop:~$

wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz

Or from http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz

hduser@laptop:~$ **tar xvzf hadoop-2.8.3.tar.gz**

hduser@laptop:~$ **cd hadoop-2.8.3**

hduser@laptop:~$ **sudo mkdir /usr/local/hadoop**

hduser@laptop:~/hadoop-2.7.2$ **sudo mv * /usr/local/hadoop**
hduser@laptop:~/hadoop-2.7.2$ **sudo chown -R hduser:hadoop /usr/local/hadoop**

hduser@laptop:~/hadoop-2.7.2$ **sudo apt-get install vim**

Check Java

hduser@laptop:~$ **readlink -f /usr/bin/javac**
/usr/lib/jvm/java-8-openjdk-amd64/bin/javac *** for 64 bit

/usr/lib/jvm/java-8-openjdk-i386/bin/javac ***for 32 bit

Set-up the Configuration Files

hduser@laptop:~$ **vim ~/.bashrc**
#HADOOP VARIABLES START ****Append this at end of file*******

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

Save the file and Exit (**esc:wq!** to save the file in vim)
hduser@laptop:~$ **source ~/.bashrc**

hduser@laptop:~$ **vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh**

(Change existing JAVA_HOME to **/usr/lib/jvm/java-7-openjdk-amd64 if machine is 64 Bit** or

If machine is 32 bit change it to /usr/lib/jvm/java-7-openjdk-i386)

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
Save and Exit

hduser@laptop:~$ **vim /usr/local/hadoop/etc/hadoop/yarn-site.xml**

Open the file and enter the following in between <configuration></configuration> tag and save :

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
```

Create Temporary Directory to store App-data

hduser@laptop:~$ **sudo mkdir -p /app/hadoop/tmp**

hduser@laptop:~$ **sudo chown hduser:hadoop /app/hadoop/tmp**

hduser@laptop:~$ **vim /usr/local/hadoop/etc/hadoop/core-site.xml**

Open the file and enter the following in between the <configuration></configuration> tag:

```
<property>
 <name>hadoop.tmp.dir</name>
 <value>/app/hadoop/tmp</value>
 <description>A base for other temporary directories.</description>
</property>
<property>
 <name>fs.default.name</name>
 <value>hdfs://localhost:54310</value></property>
```

hduser@laptop:~$ **cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml**

hduser@laptop:~$ **vim /usr/local/hadoop/etc/hadoop/mapred-site.xml**

Add these properties in <configuration></configuration> tag:

```
<property>
 <name>mapred.job.tracker</name>
 <value>localhost:54311</value>
</property>

<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

Save and Exit

hduser@laptop:~$ **sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode**

hduser@laptop:~$ **sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode**
hduser@laptop:~$ **sudo chown -R hduser:hadoop /usr/local/hadoop_store**

hduser@laptop:~$ **vim /usr/local/hadoop/etc/hadoop/hdfs-site.xml**
Open the file and add the following properties between the <configuration></configuration> tag:

```
<property>
 <name>dfs.replication</name>  <value>1</value>
```

```
 </property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
```

hduser@laptop:~$ **hadoop namenode -format**

Starting Hadoop

hduser@laptop$: **/usr/local/hadoop/sbin/start-all.sh**

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

hduser@laptop$: **jps**

9026 NodeManager

7348 NameNode

9766 SecondaryNameNode

8887 ResourceManager

7507 DataNode

Hadoop UI visible in browser at localhost:50070

MR Job progress and history UI visible at localhost:8088

To Stop Hadoop

hduser@laptop$: **/usr/local/hadoop/sbin/stop-all.sh**

**********Done**********

Conclusion: In this way single node Hadoop was installed & configured on Ubuntu for BigData analytics.

Reference: http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/SingleCluster.html

b) Multiple Nodes:

Install a Multi Node Hadoop Cluster on Ubuntu 16.04

This article is about multi-node installation of Hadoop cluster. You would need minimum of 2 ubuntu machines or virtual images to complete a multi-node installation. If you want to just try out a single node cluster, follow this article on Installing Hadoop on Ubuntu 16.04.

I used Hadoop Stable version 2.6.0 for this article. I did this setup on a 3 node cluster. For simplicity, i will designate one node as **master,** and 2 nodes as **slaves (slave-1, and slave-2)**. Make sure all slave nodes are reachable from master node. To avoid any unreachable hosts error, make sure you add the slave hostnames and ip addresses in **/etc/hosts** file. Similarly, slave nodes should be able to resolve **master** hostname.

Installing Java on Master and Slaves

```
$ sudo add-apt-repository ppa:webupd8team/java

$ sudo apt-get update

$ sudo apt-get install oracle-java7-installer

# Updata Java runtime

$ sudo update-java-alternatives -s java-7-oracle
```

Disable IPv6

As of now Hadoop does not support IPv6, and is tested to work only on IPv4 networks. If you are using IPv6, you need to switch Hadoop host machines to use IPv4. The Hadoop Wiki link provides a one liner command to disable the IPv6. If you are not using IPv6, skip this step:

```
sudo sed -i 's/net.ipv6.bindv6only\ =\ 1/net.ipv6.bindv6only\ =\ 0/' \

/etc/sysctl.d/bindv6only.conf && sudo invoke-rc.d procps restart
```

Setting up a Hadoop User

Hadoop talks to other nodes in the cluster using no-password ssh. By having Hadoop run under a specific user context, it will be easy to distribute the ssh keys around in the Hadoop cluster. Lets's create a user **hadoopuser** on **master** as well as **slave** nodes.

```
# Create hadoopgroup

$ sudo addgroup hadoopgroup

# Create hadoopuser user

$ sudo adduser —ingroup hadoopgroup hadoopuser
```

Our next step will be to generate a ssh key for password-less login between master and slave nodes. Run the following commands only on **master** node. Run the last two commands for each slave node. Password less ssh should be working before you can proceed with further steps.

```
# Login as hadoopuser

$ su - hadoopuser

#Generate a ssh key for the user

$ ssh-keygen -t rsa -P ""

#Authorize the key to enable password less ssh

$ cat /home/hadoopuser/.ssh/id_rsa.pub >> /home/hadoopuser/.ssh/authorized_keys

$ chmod 600 authorized_keys

#Copy this key to slave-1 to enable password less ssh

$ ssh-copy-id -i ~/.ssh/id_rsa.pub slave-1

#Make sure you can do a password less ssh using following command.

$ ssh slave-1
```

Download and Install Hadoop binaries on Master and Slave nodes

Pick the best mirror site to download the binaries from Apache Hadoop, and download the stable/hadoop-2.6.0.tar.gz for your installation. **Do this step on master and every slave node.** You can download the file once and the distribute to each slave node using scp command.

```
$ cd /home/hadoopuser

$ wget http://www.webhostingjams.com/mirror/apache/hadoop/core/stable/hadoop-2.2.0.tar.gz

$ tar xvf hadoop-2.2.0.tar.gz

$ mv hadoop-2.2.0 hadoop
```

Setup Hadoop Environment on Master and Slave Nodes

Copy and paste following lines into your .bashrc file under /home/hadoopuser. **Do this step on master and every slave node.**

```
# Set HADOOP_HOME

export HADOOP_HOME=/home/hduser/hadoop

# Set JAVA_HOME

export JAVA_HOME=/usr/lib/jvm/java-7-oracle

# Add Hadoop bin and sbin directory to PATH

export PATH=$PATH:$HADOOP_HOME/bin;$HADOOP_HOME/sbin
```

Update hadoop-env.sh on Master and Slave Nodes

Update JAVA_HOME in /home/hadoopuser/hadoop/etc/hadoop/hadoop_env.sh to following. **Do this step on master and every slave node.**

```
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
```

Common Terminologies

Before we start getting into configuration details, lets discuss some of the basic terminologies used in Hadoop.

- **Hadoop Distributed File System**: A distributed file system that provides high-throughput access to application data. A HDFS cluster primarily consists of a NameNode that manages the file system metadata and DataNodes that store the actual data. If you compare HDFS to a traditional storage structures (e.g. FAT, NTFS), then NameNode is analogous to a Directory Node structure, and DataNode is analogous to actual file storage blocks.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

Update Configuration Files

Add/update core-site.xml on **Master and Slave nodes** with following options. Master and slave nodes should all be using the same value for this property **fs.defaultFS,** and should be pointing to master node only.

/home/hadoopuser/hadoop/etc/hadoop/core-site.xml (Other Options)

```
<property>
 <name>hadoop.tmp.dir</name>
 <value>/home/hadoopuser/tmp</value>
 <description>Temporary Directory.</description>
</property>

<property>
 <name>fs.defaultFS</name>
 <value>hdfs://master:54310</value>
 <description>Use HDFS as file storage engine</description>
</property>
```

Add/update mapred-site.xml on **Master node only** with following options.

/home/hadoopuser/hadoop/etc/hadoop/mapred-site.xml (Other Options)

```
<property>
 <name>mapreduce.jobtracker.address</name>
 <value>master:54311</value>
 <description>The host and port that the MapReduce job tracker runs
 at. If "local", then jobs are run in-process as a single map
 and reduce task.
</description>
</property>
<property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
 <description>The framework for running mapreduce jobs</description>
</property>
```

Add/update hdfs-site.xml on **Master and Slave Nodes.** We will be adding following three entries to the file.

- **dfs.replication**– Here I am using a replication factor of 2. That means for every file stored in HDFS, there will be one redundant replication of that file on some other node in the cluster.
- **dfs.namenode.name.dir –** This directory is used by Namenode to store its metadata file. Here i manually created this directory /hadoop-data/hadoopuser/hdfs/namenode on master and slave node, and use the directory location for this configuration.
- **dfs.datanode.data.dir –** This directory is used by Datanode to store hdfs data blocks. Here i manually created this directory /hadoop-data/hadoopuser/hdfs/datanode on master and slave node, and use the directory location for this configuration.

/home/hadoopuser/hadoop/etc/hadoop/hdfs-site.xml (Other Options)

```
<property>

<name>dfs.replication</name>

<value>2</value>

<description>Default block replication.

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

</description>

</property>

<property>

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

<value>/hadoop-data/hadoopuser/hdfs/namenode</value>

<description>Determines where on the local filesystem the DFS name node should store the name table(fsimage). If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.

</description>

</property>

<property>
```

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

<value>/hadoop-data/hadoopuser/hdfs/datanode</value>

<description>Determines where on the local filesystem an DFS data node should store its blocks. If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices. Directories that do not exist are ignored.

</description>

</property>
```

Add yarn-site.xml on **Master and Slave Nodes**. This file is required for a Node to work as a Yarn Node. Master and slave nodes should all be using the same value for the following properties, and should be pointing to master node only.

/home/hadoopuser/hadoop/etc/hadoop/yarn-site.xml

```
<property>

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

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>master:8030</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>master:8032</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>master:8088</value>
```

```
</property>
<property>
  <name>yarn.resourcemanager.resource-tracker.address</name>
  <value>master:8031</value>
</property>
<property>
  <name>yarn.resourcemanager.admin.address</name>
  <value>master:8033</value>
</property>
```

Add/update **slaves** file on Master node only. Add just name, or ip addresses of master and all slave node. If file has an entry for localhost, you can remove that. This file is just helper file that are used by hadoop scripts to start appropriate services on master and slave nodes.

/home/hadoopuser/hadoop/etc/hadoop/slave

```
master

slave-1

slave-2
```

Format the Namenode

Before starting the cluster, we need to format the Namenode. Use the following command only on **master node**:

```
$ hdfs namenode -format
```

Start the Distributed Format System

Run the following on **master node** command to start the DFS.

```
$ ./home/hadoopuser/hadoop/sbin/start-dfs.sh
```

You should observe the output to ascertain that it tries to start datanode on slave nodes one by one. To validate the success, run following command on master nodes, and slave node.

```
$ su - hadoopuser

$ jps
```

The output of this command should list ***NameNode, SecondaryNameNode, DataNode*** on **master** node, and ***DataNode*** on all slave nodes. If you don't see the expected output, review the log files listed in Troubleshooting section.

Start the Yarn MapReduce Job tracker

Run the following command to start the Yarn mapreduce framework.

```
$ ./home/hadoopuser/hadoop/sbin/start-yarn.sh
```

To validate the success, run **jps** command again on master nodes, and slave node.The output of this command should list ***NodeManager, ResourceManager*** on **master** node, and ***NodeManager,*** on all **slave** nodes. If you don't see the expected output, review the log files listed in Troubleshooting section.

Review Yarn Web console

If all the services started successfully on all nodes, then you should see all of your nodes listed under Yarn nodes. You can hit the following url on your browser and verify that:

```
http://master:8088/cluster/nodes
```

Lets's execute a MapReduce example now

You should be all set to run a MapReduce example now. Run the following command

```
$ hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar pi 30 100
```

Once the job is submitted you can validate that its running on the cluster by accessing following url.

```
http://master:8088/cluster/apps
```

Troubleshooting

Hadoop uses $HADOOP_HOME/logs directory. In case you get into any issues with your installation, that should be the first point to look at. In case, you need help with anything else, do leave me a comment.

REFERENCE BOOK:

http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html

CONCLUSION:

Thus we have successfully installed and tested single and multimode cluster.

Assignment: 2

AIM: Design a distributed application using MapReduce.

PROBLEM STATEMENT /DEFINITION

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.

OBJECTIVE:

- To understand the concept of Map Reduce.
- To understand the details of Hadoop File system
- To understand the technique for log file processing
- Analyze the performance of hadoop file system
- To understand use of distributed processing

THEORY:

What is MapReduce?

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

- 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.

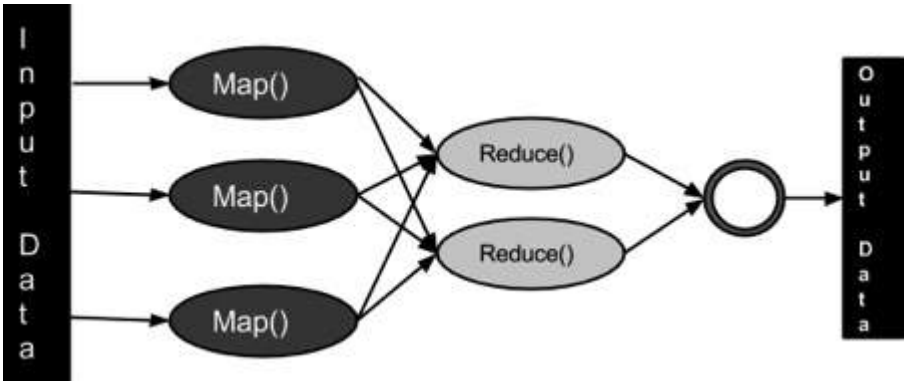


Fig.1. Map Reduce job processing

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 largescale 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*/
  {

    //Map function
    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 http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1to 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

14/10/31 06:02:52

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.

Options	Description
namenode -format	Formats the DFS filesystem.
secondarynamenode	Runs the DFS secondary namenode.
namenode	Runs the DFS namenode.
datanode	Runs a DFS datanode.
dfsadmin	Runs a DFS admin client.
mradmin	Runs a Map-Reduce admin client.
fsck	Runs a DFS filesystem checking utility.
fs	Runs a generic filesystem user client.
balancer	Runs a cluster balancing utility.
oiv	Applies the offline fsimage viewer to an fsimage.
fetchdt	Fetches a delegation token from the NameNode.
jobtracker	Runs the MapReduce job Tracker node.

pipes	Runs a Pipes job.
tasktracker	Runs a MapReduce task Tracker node.
historyserver	Runs job history servers as a standalone daemon.
job	Manipulates the MapReduce jobs.
queue	Gets information regarding JobQueues.
version	Prints the version.
jar <jar>	Runs a jar file.
distcp <srcurl> <desturl>	Copies file or directories recursively.
distcp2 <srcurl> <desturl>	DistCp version 2.
archive -archiveName NAME -p	Creates a hadoop archive.
<parent path> <src>* <dest>	
classpath	Prints the class path needed to get the Hadoop jar and the required libraries.
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.

GENERIC_OPTIONS	Description
-submit <job-file>	Submits the job.
-status <job-id>	Prints the map and reduce completion percentage and all job counters.
-counter <job-id> <group-name> <countername>	Prints the counter value.
-kill <job-id>	Kills the job.
-events <job-id> <fromevent-#> <#-of-events>	Prints the events' details received by jobtracker for the given range.
-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.
-list[all]	Displays all jobs. -list displays only jobs which are yet to complete.
-kill-task <task-id>	Kills the task. Killed tasks are NOT counted against failed attempts.
-fail-task <task-id>	Fails the task. Failed tasks are counted against failed attempts.
-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
```

2. MapReduce (Log File)

Use Hadoop to Analyze Java Logs (Tomcat catalina.out)

One of the Java applications I develop deploys in Tomcat and is load-balanced across a couple dozen servers. Each server can produce gigabytes of log output daily due to the high volume. This post demonstrates simple use of hadoop to quickly extract useful and relevant information from catalina.out files using Map Reduce. I followed Hadoop: The Definitive Guide for setup and example code.

Installing Hadoop

Hadoop in standalone mode was the most convenient for initial development of the Map Reduce classes. The following commands were executed on a virtual server running RedHat Enterprise Linux 6.3. First verify Java 6 is installed:

```
[watrous@myhost ~]$ java -version
java version "1.6.0_24"
OpenJDK Runtime Environment (IcedTea6 1.11.5) (rhel-1.50.1.11.5.el6_3-x86_64)
OpenJDK 64-Bit Server VM (build 20.0-b12, mixed mode)
```

Next, download and extract Hadoop. Hadoop can be downloaded using a mirror. Hadoop can be setup and run locally and does not require any special privileges. Always verify that you have a good download.

```
[watrous@myhost ~]$ wget http://download.nextag.com/apache/hadoop/common/stable/hadoop-2.2.0.tar.gz
[watrous@myhost ~]$ md5sum hadoop-2.2.0.tar.gz
25f27eb0b5617e47c032319c0bfd9962  hadoop-2.2.0.tar.gz
[watrous@myhost ~]$ tar xzf hadoop-2.2.0.tar.gz
[watrous@myhost ~]$ hdfs namenode -format
```

That last command creates an HDFS file system in the **tmp** folder. In my case it was created here: **/tmp/hadoop-watrous/dfs/**.

Environment variables were added to **.bash_profile** for JAVA_HOME and HADOOP_INSTALL, as shown. These can also be run locally each time you login.

```
export JAVA_HOME=/usr/lib/jvm/jre
export HADOOP_INSTALL=/home/watrous/hadoop-2.2.0
export PATH=$PATH:$HADOOP_INSTALL/bin
```

I can now verify that Hadoop is installed and ready to run.

```
[watrous@myhost ~]$ hadoop version
Hadoop 2.2.0
Subversion https://svn.apache.org/repos/asf/hadoop/common -r 1529768
Compiled by hortonmu on 2013-10-07T06:28Z
Compiled with protoc 2.5.0
From source with checksum 79e53ce7994d1628b240f09af91e1af4
This command was run using /home/watrous/hadoop-2.2.0/share/hadoop/common/hadoop-common-2.2.0.jar
```

Get some seed data

Now that Hadoop is all setup, I need some seed data to operate on. For this I just reached out and grabbed a log file from one of my production servers.

```
[watrous@myhost ~]$ mkdir input
[watrous@myhost ~]$ scp watrous@mywebhost.com:/var/lib/tomcat/logs/catalina.out ./input/
```

Creating Map Reduce Classes

The most simple operation in Hadoop requires a Mapper class, a Reducer class and a third class that identifies the Mapper and Reducer including the datatypes that connect them. The examples below required two jars from the release downloaded above:

- hadoop-2.2.0.tar.gz\hadoop-2.2.0.tar\hadoop-2.2.0\share\hadoop\common\hadoop-common-2.2.0.jar
- hadoop-2.2.0.tar.gz\hadoop-2.2.0.tar\hadoop-2.2.0\share\hadoop\mapreduce\hadoop-mapreduce-client-core-2.2.0.jar

I also use regular expressions in Java to analyze each line in the log. Regular expressions can be more resilient to variations and allow for grouping, which gives easy access to specific data elements. As always, I used Kodos to develop the regular expression.

In the example below, I don't actually use the log value, but instead I just count up how many occurrences there are by key.

Mapper class

```
import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class TomcatLogErrorMapper extends Mapper<LongWritable, Text, Text, Text> {

   String pattern = "([0-9]{4}-[0-9]{2}-[0-9]{2}\\s[0-9]{2}:[0-9]{2}:[0-9]{2}.[0-9]{3})\\s*([a-zA-Z]+)\\s*([a-zA-Z.]+)\\s*-\\s*(.+)$";
   // Create a Pattern object
   Pattern r = Pattern.compile(pattern);

   @Override
   public void map(LongWritable key, Text value, Context context)
         throws IOException, InterruptedException {
      String line = value.toString();

      Matcher m = r.matcher(line);
      if (m.find()) {
         // only consider ERRORs for this example
         if (m.group(2).contains("ERROR")) {
            // example log line
       // 2013-11-08 04:06:56,586 DEBUG component.helpers.GenericSOAPConnector  - Attempting to connect to: https://remotehost.com/app/rfc/entry/msg  status
//             System.out.println("Found value: " + m.group(0)); //complete line
//             System.out.println("Found value: " + m.group(1)); // date
//             System.out.println("Found value: " + m.group(2)); // log level
//             System.out.println("Found value: " + m.group(3)); // class
//             System.out.println("Found value: " + m.group(4)); // message
         context.write(new Text(m.group(1)), new Text(m.group(2) + m.group(3) + m.group(4)));
         }
      }
   }
}
```

Reducer class

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class TomcatLogErrorReducer extends Reducer<Text, Text, Text, IntWritable> {
```

```
    @Override
    public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
        int countValue = 0;
        for (Text value : values) {
            countValue++;
        }
        context.write(key, new IntWritable(countValue));
    }
}
```

Job class with main

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class TomcatLogError {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: TomcatLogError <input path> <output path>");
            System.exit(-1);
        }
        Job job = new Job();
        job.setJarByClass(TomcatLogError.class);
        job.setJobName("Tomcat Log Error");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setMapperClass(TomcatLogErrorMapper.class);
        job.setReducerClass(TomcatLogErrorReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Running Hadoop

In netbeans I made sure that the Main Class was TomcatLogError in the compiled jar. I then ran Clean and Build to get a jar which I transferred up to the server where I installed Hadoop.

```
[watrous@myhost ~]$ hadoop jar HadoopExample.jar input/catalina.out ~/output
13/11/11 19:20:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
13/11/11 19:20:52 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
...
13/11/11 18:36:57 INFO mapreduce.Job: Job job_local1725513594_0001 completed successfully
13/11/11 18:36:57 INFO mapreduce.Job: Counters: 27
        File System Counters
                FILE: Number of bytes read=430339145
```

```
        FILE: Number of bytes written=1057396
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
    Map-Reduce Framework
        Map input records=1101516
        Map output records=105
        Map output bytes=20648
        Map output materialized bytes=20968
        Input split bytes=396
        Combine input records=0
        Combine output records=0
        Reduce input groups=23
        Reduce shuffle bytes=0
        Reduce input records=105
        Reduce output records=23
        Spilled Records=210
        Shuffled Maps =0
        Failed Shuffles=0
        Merged Map outputs=0
        GC time elapsed (ms)=234
        CPU time spent (ms)=0
        Physical memory (bytes) snapshot=0
        Virtual memory (bytes) snapshot=0
        Total committed heap usage (bytes)=1827143680
    File Input Format Counters
        Bytes Read=114455257
    File Output Format Counters
        Bytes Written=844
```

The output folder now contains a file named **part-r-00000** with the results of the processing.

```
[watrous@c0003913 ~]$ more output/part-r-00000
2013-11-08 04:04:51,894 2
2013-11-08 05:04:52,711 2
2013-11-08 05:33:23,073 3
2013-11-08 06:04:53,689 2
2013-11-08 07:04:54,366 3
2013-11-08 08:04:55,096 2
2013-11-08 13:34:28,936 2
2013-11-08 17:32:31,629 3
2013-11-08 18:51:17,357 1
2013-11-08 18:51:17,423 1
2013-11-08 18:51:17,491 1
2013-11-08 18:51:17,499 1
2013-11-08 18:51:17,500 1
2013-11-08 18:51:17,502 1
2013-11-08 18:51:17,503 1
2013-11-08 18:51:17,504 1
2013-11-08 18:51:17,506 1
2013-11-08 18:51:17,651 6
```

```
2013-11-08 18:51:17,652 23
2013-11-08 18:51:17,653 25
2013-11-08 18:51:17,654 19
2013-11-08 19:01:13,771 2
2013-11-08 21:32:34,522 2
```

Based on this analysis, there were a number of errors produced around the hour **18:51:17**. It is then easy to change the Mapper class to emit based on a different key, such as Class or Message to identify more precisely what the error is, now that I know when the errors happened.

References:

https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

CONCLUSION:

Understand the uses of distributed data processing using Map reduce.

Assignment: 3

AIM: Design and develop a distributed application using MapReduce.

PROBLEM STATEMENT /DEFINITION

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.

OBJECTIVE:

- To understand use of various aggregate operation in Map Reduce.
- To understand use of various Key Value passing techniques in Map Reduce.
- To understand preprocessing on input data file.

THEORY:

Download dataset.zip file (Weather data)
It contains NCDC weather data from year 1901 to year 1920.
Copy and extract dataset.zip in your home folder
Open terminal
whoami
It will display your user name, we will use it later.
Open eclipse->new java project->project name
exp7
->new class->
MaxTemperatureMapper
Add following code in that class

```
Package exp7;
Import java.io.IOException;
Import org.apache.hadoop.io.IntWritable;
```

```
Import org.apache.hadoop.io.LongWritable;
Import org.apache.hadoop.io.Text;
Import org.apache.hadoop.mapreduce.Mapper;
Public class MaxTemperatureMapper extends
Mapper<LongWritable,Text,Text,IntWritable>
{
Private static final int MISSING = 9999;
@Override public void map(LongWritable key ,Text value , Context context)
throws IOException,InterruptedException
{
String line = value.toString();
String year = line.substring(15, 19);
intairTemperature;
if (line.charAt(87)=='+')
{
airTemperature= Integer.parseInt(line.substring(88, 92));
}
else
{
airTemperature= Integer.parseInt(line.substring(87, 92));
}
String quality= line.substring(92, 93);
if (airTemperature != MISSING&& quality.matches("[01459]"))
{
context.write(new Text(year), new IntWritable(airTemperature));
}
}
}
```

Save the file

It will display some errors, so we are going to import two jar files in our project.

Copy hadoop-mapreduce-client-core-2.7.1.jar from ~/hadoop/share/hadoop/mapreduce directory

In eclipse-> right click on exp7 project- >paste

Right click on pasted hadoop-mapreduce-client-core-2.7.1.jar-> Buid path-> add to buid path

#Copy hadoop-common-2.7.1.jar from ~/hadoop/share/hadoop/common directory

In eclipse-> right click on exp7 project- >paste

Right click on pasted hadoop-common-2.7.1.jar-> Buid path-> add to buid path
Right click on project exp7->new class-> MaxTemperatureReducer
Add following code in that class

```
Package exp7;
Import java.io.IOException;
Import org.apache.hadoop.io.IntWritable;
Import org.apache.hadoop.io.Text;
Import org.apache.hadoop.mapreduce.Reducer;
Public classMaxTemperatureReducer extends Reducer<Text,IntWritable, Text, IntWritable>
{
@Override public void reduce(Text key , Iterable<IntWritable> values , Context  context )
Throws IOException, InterruptedException
{
Int maxValue = Integer.MIN_VALUE;
For (IntWritable  value :  values )
{
maxValue = Math. Max ( maxValue, value.get());
}
Context.write(key, new IntWritable(maxValue));
}
}
```

Save the file
Right click on project exp7->new class-> MaxTemperature
Add following code in that class (replace your_user_name by your own username)
hdfs port number here is 1234, replace it with your port no (if different).

```
Package exp7;
Import java.io.BufferedReader;
Import java.io.File;
Import java.io.FileReader;
Import java.util.ArrayList;
Import java.util.List;
Import java.util.Scanner;
Import org.apache.hadoop.conf.Configuration;
Import org.apache.hadoop.fs.FileStatus;
Import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
```

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class MaxTemperature
{
Public static void main(String[] args)
Throws Exception
{
If (args.length != 2)
{
System.err.println("Usage:MaxTemperature<input path> <output path>");
System.exit(-1);
}
@SuppressWarnings
(
"deprecation"
)
Job job= new Job();
Job.setJarByClass(MaxTemperature.class);
Job.setJobName("Max temperature");
FileInputFormat.addInputPath(job,newPath(args[0]));
FileOutputFormat.setOutputPath(job,newPath(args[1]));
job.setMapperClass(MaxTemperatureMapper.class);
job.setReducerClass(MaxTemperatureReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.waitForCompletion(true);
Configuration conf= newConfiguration();
conf.set("fs.defaultFS", "hdfs://localhost:1234/user/your_user_name/");
FileSystem fs= FileSystem.get(conf);
FileStatus[] status= fs.listStatus(newPath(args[1]));
//copy hdfsoutput file to local folder
for(int i=0;i<status.length;i++)
{
System.out.println(status[i].getPath());
fs.copyToLocalFile(false, status[i].getPath(), newPath("/home/your_user_name/"+args[1]));
}
System.out.println("\nYear\tTemperature\n");
//display contents of local file
BufferedReader br= newBufferedReader(newFileReader("/home/your_user_name/"+args[1]));
String line= null;
while((line= br.readLine()) != null)
{
System.out.println(line);
}
```

```
br.close();
Scanner s= newScanner(newFile("/home/your_user_name/"+args[1]));
List<Integer> temps= newArrayList<Integer>();
List<String> years= newArrayList<String>();
while(s.hasNext())
{
years.add(s.next());
temps.add(Integer.parseInt(s.next()));
}
Int max_temp=0,min_temp=999,i=0,j=0;
String hottest_year="", coolest_year="";
for(inttemp: temps)
{
if(temp>max_temp)
{
max_temp=temp;
hottest_year=years.get(i);
}
i++;
}
Float max_temp1=max_temp;
System.out.println("Hottest Year:"+hottest_year);
System.out.println("\tTemperature:"+max_temp1/10+" Degree Celcius");
for(int temp: temps)
{
if(temp<min_temp)
{
min_temp=temp;
coolest_year=years.get(j);
}
j++;
}
Float min_temp1=min_temp;
System.out.println("Coolest Year:"+coolest_year);
System.out.println("\tTemperature:"+min_temp1/10+" Degree Celcius");
s.close();
}
}
```

Save the file

In eclipse->Right click on project exp7-> export->java->jar file->next-> select the export destination -> /home/ your_user_name /exp7.jar

-> next -> next -> select main class ->browse -
>
MaxTemperature -
>
finish
#
exp7.jar
file will be created in your home folder
Open terminal
Now Start NameNode daemon and DataNode daemon:
~/hadoop/sbin/start-dfs.sh
Make the HDFS directories required to execute MapReduce jobs (if not already done)
~/hadoop/bin/hdfs dfs -mkdir /user
~/hadoop/bin/hdfs dfs -mkdir /user/
your_user_name
Put NCDC weather dataset in hdfs
~/hadoop/bin/hdfs dfs -put ~/dataset input_dataset
Perform MapReduce job
~/hadoop/bin/hadoop jar ~/exp7.jar input_dataset output_dataset
Output
Stop haddop
~/hadoop/sbin/stop-dfs.sh
jps

REFERENCES:

https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

CONCLUSION:

Understand to use various aggregate functions using map reduce.

Assignment: 4

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

PROBLEM STATEMENT /DEFINITION

Write an application using HBase and HiveQL for flight information system which will include

a. Creating, Dropping, and altering Database tables
b. Creating an external Hive table to connect to the HBase for Customer Information Table
c. Load table with data, insert new values and field in the table, Join tables with Hive

Create index on Flight information Table 5) Find the average departure delay per day in 2008.

OBJECTIVE

- To understand various NOSQL database
- To understand the integration of NOSQL database with Hadoop.
- To analyze the performance of distributed processing with NOSQL.

THEORY:

HBase via Hive

This is the second of two posts examining the use of Hive for interaction with HBase tables. This is a hands-on exploration so the first post isn't required reading for consuming this one. Still, it might be good context.
"Nick!" you exclaim, "that first post had too many words and I don't care about JIRA tickets. Show me how I use this thing!"

This is post is exactly that: a concrete, end-to-end example of consuming HBase over Hive. The whole mess was tested to work on a tiny little 5-node cluster running HDP-1.3.2, which means Hive 0.11.0 and HBase 0.94.6.1.

Grab some data and register it in Hive

We'll need some data to work with. For this purpose, grab some traffic stats from wikipedia. Once we have some data, copy it up to HDFS.

```
$ mkdir pagecounts ; cd pagecounts
$ for x in {0..9} ; do wget "http://dumps.wikimedia.org/other/pagecounts-raw/2008/2008-10/pagecounts-20081001-0${x}
0000.gz" ; done
$ hadoop fs -copyFromLocal $(pwd) ./
```

For reference, this is what the data looks like.

```
$ zcat pagecounts-20081001-000000.gz | head -n5
aa.b Special:Statistics 1 837
aa Main_Page 4 41431
aa Special:ListUsers 1 5555
aa Special:Listusers 1 1052
aa Special:PrefixIndex/Comparison_of_Guaze%27s_Law_and_Coulomb%27s_Law 1 4332
```

As I understand it, each record is a count of page views of a specific page on Wikipedia. The first column is the language code, second is the page name, third is the number of page views, and fourth is the size of the page in bytes. Each file contains an hour's worth of aggregated data. None of the above pages were particularly popular that hour.

Now that we have data and understand its raw schema, create a Hive table over it. To do that, we'll use a DDL script that looks like this.

```
$ cat 00_pagecounts.ddl
-- define an external table over raw pagecounts data
CREATE TABLE IF NOT EXISTS pagecounts (projectcode STRING, pagename STRING, pageviews STRING, bytes STRING)
ROW FORMAT
  DELIMITED FIELDS TERMINATED BY ' '
  LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/ndimiduk/pagecounts';
```

Run the script to register our dataset with Hive.

```
$ hive -f 00_pagecounts.ddl
OK
```

```
Time taken: 2.268 seconds
```

Verify that the schema mapping works by calculating a simple statistic over the dataset.

```
$ hive -e "SELECT count(*) FROM pagecounts;"
Total MapReduce jobs = 1
Launching Job 1 out of 1
...
OK
36668549
Time taken: 25.31 seconds, Fetched: 1 row(s)
```

Hive says the 10 files we downloaded contain just over 36.5mm records. Let's just confirm things are working as expected by getting a second opinion. This isn't that much data, so confirm on the command line.

```
$ zcat * | wc -l
36668549
```

The record counts match up – excellent.

Transform the schema for HBase

The next step is to transform the raw data into a schema that makes sense for HBase. In our case, we'll create a schema that allows us to calculate aggregate summaries of pages according to their titles. To do this, we want all the data for a single page grouped together. We'll manage that by creating a Hive view that represents our target HBase schema. Here's the DDL.

```
$ cat 01_pgc.ddl
-- create a view, building a custom hbase rowkey
CREATE VIEW IF NOT EXISTS pgc (rowkey, pageviews, bytes) AS
SELECT concat_ws('/',
        projectcode,
        concat_ws('/',
          pagename,
          regexp_extract(INPUT__FILE__NAME, 'pagecounts-(\\d{8}-\\d{6})\\..*$', 1))),
      pageviews, bytes
FROM pagecounts;
```

The `SELECT` statement uses hive to build a compound rowkey for HBase. It concatenates the project code, page name, and date, joined by the `'/'` character. A handy trick: it uses a simple regex to extract the date from the source file names. Run it now.

```
$ hive -f 01_pgc.ddl
OK
Time taken: 2.712 seconds
```

This is just a view, so the `SELECT` statement won't be evaluated until we query it for data. Registering it with hive doesn't actually process any data. Again, make sure it works by querying Hive for a subset of the data.

```
$ hive -e "SELECT * FROM pgc WHERE rowkey LIKE 'en/q%' LIMIT 10;"
Total MapReduce jobs = 1
Launching Job 1 out of 1
...
OK
en/q:Special:Search/Blues/20081001-090000       1       1168
en/q:Special:Search/rock/20081001-090000        1       985
en/qadam_rasul/20081001-090000  1       1108
en/qarqay/20081001-090000       1       933
en/qemu/20081001-090000 1       1144
en/qian_lin/20081001-090000     1       918
en/qiang_(spear)/20081001-090000        1       973
en/qin_dynasty/20081001-090000  1       1120
en/qinghe_special_steel_corporation_disaster/20081001-090000    1       963
en/qmail/20081001-090000        1       1146
Time taken: 40.382 seconds, Fetched: 10 row(s)
```

Register the HBase table

Now that we have a dataset in Hive, it's time to introduce HBase. The first step is to register our HBase table in Hive so that we can interact with it using Hive queries. That means another DDL statement. Here's what it looks like.

```
$ cat 02_pagecounts_hbase.ddl
-- create a table in hbase to host the view
CREATE TABLE IF NOT EXISTS pagecounts_hbase (rowkey STRING, pageviews STRING, bytes STRING)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,f:c1,f:c2')
TBLPROPERTIES ('hbase.table.name' = 'pagecounts');
```

This statement will tell Hive to go create an HBase table named `pagecounts` with the single column family `f`. It registers that HBase table in the Hive metastore by the name `pagecounts_hbase` with 3 columns: `rowkey`, `pageviews`, and `bytes`. The SerDe property `hbase.columns.mapping` makes the association from Hive column to HBase column. It says the Hive column `rowkey` is mapped to the HBase table's rowkey, the Hive

column `pageviews` to the HBase column `f:c1`, and `bytes` to the HBase column `f:c2`. To keep the example simple, we have Hive treat all these columns as the `STRING` type.
In order to use the HBase library, we need to make the HBase jars and configuration available to the local Hive process (at least until HIVE-5518 is resolved). Do that by specifying a value for the `HADOOP_CLASSPATH` environment variable before executing the statement.

```
$ export HADOOP_CLASSPATH=/etc/hbase/conf:/usr/lib/hbase/hbase-0.94.6.1.3.2.0-111-security.jar:/usr/lib/
zookeeper/zookeeper.jar
$ hive -f 02_pagecounts_hbase.ddl
OK
Time taken: 4.399 seconds
```

Populate the HBase table

Now it's time to write data to HBase. This is done using a regular Hive `INSERT` statement, sourcing data from the view with `SELECT`. There's one more bit of administration we need to take care of though. This `INSERT` statement will run a mapreduce job that writes data to HBase. That means we need to tell Hive to ship the HBase jars and dependencies with the job.
Note that this is a separate step from the classpath modification we did previously. Normally you can do this with an `export` statement from the shell, the same way we specified the `HADOOP_CLASSPATH`. However there's a bug in HDP-1.3 that requires me to use Hive's `SET` statement in the script instead.

```
$ cat 03_populate_hbase.hql
-- ensure hbase dependency jars are shipped with the MR job
-- Should export HIVE_AUX_JARS_PATH but this is broken in HDP-1.3.x
SET hive.aux.jars.path = file:///etc/hbase/conf/hbase-site.xml,file:///usr/lib/hive/lib/hive-hbase-handler-0.11.0.1.3.2.0-111.jar,file:
///usr/lib/hbase/hbase-0.94.6.1.3.2.0-111-security.jar,file:///usr/lib/zookeeper/zookeeper-3.4.5.1.3.2.0-111.jar;

-- populate our hbase table
FROM pgc INSERT INTO TABLE pagecounts_hbase SELECT pgc.* WHERE rowkey LIKE 'en/q%' LIMIT 10;
```

Note there's a big ugly bug in Hive 0.12.0 which means this doesn't work with that version. Never fear though, we have a patch in progress. Follow along at HIVE-5515.
If you choose to use a different method for setting Hive's auxpath, be advised that it's a tricky process – depending on how you specify it (`HIVE_AUX_JARS_PATH`, `--auxpath`), Hive will interpret the argument differently. HIVE-2349seeks to remedy this unfortunate state of affairs.

```
$ hive -f 03_populate_hbase.hql
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
```

```
...
OK
Time taken: 40.296 seconds
```

Be advised also that this step is currently broken on secured HBase deployments. Follow along HIVE-5523 if that's of interest to you.

Query data from HBase-land

40 seconds later, you now have data in HBase. Let's have a look using the HBase shell.

```
$ echo "scan 'pagecounts'" | hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.94.6.1.3.2.0-111, r410a7a1c151ca953553eae68aa84e2a9f0d6e4ca, Mon Aug 19 19:00:12 PDT 2013

scan 'pagecounts'
ROW                                          COLUMN+CELL
 en/q:Pan%27s_Labyrinth/20081001-080000      column=f:c1, timestamp=1381534232485, value=1
 en/q:Pan%27s_Labyrinth/20081001-080000      column=f:c2, timestamp=1381534232485, value=1153
 en/q:Special:Search/Jazz/20081001-080000    column=f:c1, timestamp=1381534232485, value=1
 en/q:Special:Search/Jazz/20081001-080000    column=f:c2, timestamp=1381534232485, value=980
 en/q:Special:Search/peinture/20081001-080000 column=f:c1, timestamp=1381534232485, value=1
 en/q:Special:Search/peinture/20081001-080000 column=f:c2, timestamp=1381534232485, value=989
 en/q:Special:Search/rock/20081001-080000    column=f:c1, timestamp=1381534232485, value=1
 en/q:Special:Search/rock/20081001-080000    column=f:c2, timestamp=1381534232485, value=980
 en/qadi/20081001-080000                     column=f:c1, timestamp=1381534232485, value=1
 en/qadi/20081001-080000                     column=f:c2, timestamp=1381534232485, value=1112
 en/qalawun%20complex/20081001-080000        column=f:c1, timestamp=1381534232485, value=1
 en/qalawun%20complex/20081001-080000        column=f:c2, timestamp=1381534232485, value=942
 en/qalawun/20081001-080000                  column=f:c1, timestamp=1381534232485, value=1
 en/qalawun/20081001-080000                  column=f:c2, timestamp=1381534232485, value=929
 en/qari'/20081001-080000                    column=f:c1, timestamp=1381534232485, value=1
 en/qari'/20081001-080000                    column=f:c2, timestamp=1381534232485, value=929
 en/qasvin/20081001-080000                   column=f:c1, timestamp=1381534232485, value=1
 en/qasvin/20081001-080000                   column=f:c2, timestamp=1381534232485, value=921
 en/qemu/20081001-080000                     column=f:c1, timestamp=1381534232485, value=1
 en/qemu/20081001-080000                     column=f:c2, timestamp=1381534232485, value=1157
10 row(s) in 0.4960 seconds
```

Here we have 10 rows with two columns each containing the data loaded using Hive. It's now accessible in your online world using HBase. For example, perhaps you receive an updated data file and have a corrected value for one of the stats. You can update the record in HBase with a regular `PUT` command.

Verify data from from Hive

The HBase table remains available to you Hive world; Hive's `HBaseStorageHandler` works both ways, after all.
Note that this command expects that the `HADOOP_CLASSPATH` is still set and `HIVE_AUX_JARS_PATH` as well if your query is complex.

```
$ hive -e "SELECT * from pagecounts_hbase;"
OK
en/q:Pan%27s_Labyrinth/20081001-080000  1       1153
en/q:Special:Search/Jazz/20081001-080000        1       980
en/q:Special:Search/peinture/20081001-080000    1       989
en/q:Special:Search/rock/20081001-080000        1       980
en/qadi/20081001-080000 1       1112
en/qalawun%20complex/20081001-080000    1       942
en/qalawun/20081001-080000      1       929
en/qari'/20081001-080000        1       929
en/qasvin/20081001-080000       1       921
en/qemu/20081001-080000 1       1157
Time taken: 2.554 seconds, Fetched: 10 row(s)
```

Continue using Hive for analysis

Since the HBase table is accessible from Hive, you can continue to use Hive for your ETL processing with mapreduce. Keep in mind that the auxpath considerations apply here too, so I've scripted out the query instead of just running it directly at the command line.

```
$ cat 04_query_hbase.hql
-- ensure hbase dependency jars are shipped with the MR job
-- Should export HIVE_AUX_JARS_PATH but this is broken in HDP-1.3.x
SET hive.aux.jars.path = file:///etc/hbase/conf/hbase-site.xml,file:///usr/lib/hive/lib/hive-hbase-handler-0.11.0.1.3.2.0-111.jar,file:///usr/

-- query hive data
SELECT count(*) from pagecounts_hbase;
```

Run it the same way we did the others.

```
$ hive -f 04_query_hbase.hql
```

```
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
...
OK
10
Time taken: 19.473 seconds, Fetched: 1 row(s)
```

REFERENCES:

https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

CONCLUSION:

Understand to use various aggregate functions using map reduce.

Part B

Assignments based on R and Python

Assignment: 5

AIM: Perform different operations using R/Python

PROBLEM STATEMENT /DEFINITION

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

a. Create data subsets
b. Merge Data
c. Sort Data
d. Transposing Data
e. Melting Data to long format

Casting data to wide format

OBJECTIVE:

To learn R/Python programming
To learn different data preprocessing techniques.

THEORY:

Subsetting Data

R has powerful indexing features for accessing object elements. These features can be used to select and exclude variables and observations. The following code snippets demonstrate ways to keep or delete variables and observations and to take random samples from a dataset.

Selecting (Keeping) Variables

```
# select variables v1, v2, v3
myvars <- c("v1", "v2", "v3")
newdata <- mydata[myvars]

# another method
myvars <- paste("v", 1:3, sep="")
newdata <- mydata[myvars]

# select 1st and 5th thru 10th variables
newdata <- mydata[c(1,5:10)]
```

To practice this interactively, try the selection of data frame elements exercises in the Data frames chapter of this introduction to R course.

Excluding (DROPPING) Variables

```
# exclude variables v1, v2, v3
myvars <- names(mydata) %in% c("v1", "v2", "v3")
newdata <- mydata[!myvars]

# exclude 3rd and 5th variable
newdata <- mydata[c(-3,-5)]

# delete variables v3 and v5
mydata$v3 <- mydata$v5 <- NULL
```

Selecting Observations

```
# first 5 observations
newdata <- mydata[1:5,]

# based on variable values
newdata <- mydata[ which(mydata$gender=='F'
& mydata$age > 65), ]

# or
attach(mydata)
newdata <- mydata[ which(gender=='F' & age > 65),]
detach(mydata)
```

Selection using the Subset Function

The **subset( )** function is the easiest way to select variables and observations. In the following example, we select all rows that have a value of age greater than or equal to 20 or age less then 10. We keep the ID and Weight columns.

```
# using subset function
newdata <- subset(mydata, age >= 20 | age < 10,
select=c(ID, Weight))
```

In the next example, we select all men over the age of 25 and we keep variables weight *through* income (weight, income and all columns between them).

```
# using subset function (part 2)
newdata <- subset(mydata, sex=="m" & age > 25,
select=weight:income)
```

To practice the **subset()** function, try this this interactive exercise. on subsetting data.tables.

Random Samples

Use the **sample( )** function to take a **random sample of size n** from a dataset.

```
# take a random sample of size 50 from a dataset mydata
# sample without replacement
mysample <- mydata[sample(1:nrow(mydata), 50,
   replace=FALSE),]
```

Merging Data

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)
```

If data frameA has variables that data frameB does not, then either:

1. Delete the extra variables in data frameA or
2. Create the additional variables in data frameB and set them to NA (missing) before joining them with **rbind( )**.

Going Further

To practice manipulating data frames with the dplyr package, try this interactive course on data frame manipulation in R.

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)
```

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)
```

The Reshape Package

Hadley Wickham has created a comprehensive package called reshape to massage data. Both an introduction and article are available. There is even a video!

Basically, you "melt" data so that each row is a unique id-variable combination. Then you "cast" the melted data into any shape you would like. Here is a very simple example.

mydata

id	time	x1	x2
1	1	5	6
1	2	3	5
2	1	6	1
2	2	2	4

```
# example of melt function
library(reshape)
mdata <- melt(mydata, id=c("id","time"))
```

newdata

id	time	variable	value

1	1	x1	5
1	2	x1	3
2	1	x1	6
2	2	x1	2
1	1	x2	6
1	2	x2	5
2	1	x2	1
2	2	x2	4

```
# cast the melted data
```

```
# cast(data, formula, function)
```

```
subjmeans <- cast(mdata, id~variable, mean)
```

```
timemeans <- cast(mdata, time~variable, mean)
```

subjmeans

id	x1	x2
1	4	5.5
2	4	2.5

timemeans

time	x1	x2
1	5.5	3.5
2	2.5	4.5

Melting

There are many situations where data is presented in a format that is not ready to dive straight to exploratory data analysis or to use a desired statistical method. The **reshape2** package for **R** provides useful functionality to avoid having to hack data around in a spreadsheet prior to import into **R**.

The **melt** function takes data in wide format and stacks a set of columns into a single column of data. To make use of the function we need to specify a data frame, the id variables (which will be left at their settings) and the measured variables (columns of data) to be stacked. The default assumption on measured variables is that it is all columns that are not specified as id variables.

Consider the following set of data:

```
> dat
  FactorA FactorB     Group1     Group2     Group3      Group4
1     Low     Low -1.1616334 -0.5228371 -0.6587093  0.45064563
2  Medium     Low -0.5991478 -1.0461138 -0.1942979  2.47985577
3    High     Low  0.8420797 -1.5413266  0.6318852 -0.98948125
4     Low  Medium  1.6225569 -1.2706469 -0.8026467 -0.32332181
5  Medium  Medium -0.3450745 -1.3377985  1.4988363  0.36541918
6    High  Medium  1.6025044  0.7631882 -0.5375833  0.85028148
7     Low    High -1.2991011 -0.2223622 -0.6321478 -1.57284216
8  Medium    High -0.4906400 -1.1802192  0.1235253  0.09891793
9    High    High  0.3897769 -0.3832142  0.6671101  0.23407257
```

There four groups are to used as part of a statistical analysis so we want to stack them into a single column and create an factor variable to indicate which group the measurement corresponds to and the **melt** function does the trick:

```
> melt(dat)
Using FactorA, FactorB as id variables
   FactorA FactorB variable       value
1      Low     Low   Group1 -1.16163338
2   Medium     Low   Group1 -0.59914783
3     High     Low   Group1  0.84207974
4      Low  Medium   Group1  1.62255690
5   Medium  Medium   Group1 -0.34507455
6     High  Medium   Group1  1.60250438

...
36    High    High   Group4  0.23407257
```

Consider a second set of data where there are two groups but we only want to retain the FactorB variable in the molten data set:

```
   FactorA   FactorB   Group1   Group2
1      Low  Very Low 6.851828 3.061329
2   Medium  Very Low 7.352169 1.303077
3     High  Very Low 6.918091 2.477875
4      Low       Low 7.402351 2.450527
5   Medium       Low 6.928385 4.334323
6     High       Low 7.400626 3.074158
7      Low    Medium 8.312145 5.725185
```

```
8   Medium    Medium 8.251806 4.384492
9     High    Medium 8.339398 3.443789
10     Low      High 5.127386 2.868952
11  Medium      High 8.561181 3.616898
12    High      High 6.993838 3.450634
13     Low Very High 7.880877 2.950622
14  Medium Very High 9.439892 3.220295
15    High Very High 8.799447 3.106060
```

We now need to specify both the **id.vars** and **measure.vars** arguments in the **melt** function to get the desired output:

```
> melt(dat, id.vars = "FactorB", measure.vars = c("Group1", "Group2"))
     FactorB variable    value
1   Very Low   Group1 6.851828
2   Very Low   Group1 7.352169
3   Very Low   Group1 6.918091
4        Low   Group1 7.402351
5        Low   Group1 6.928385
6        Low   Group1 7.400626
...
30 Very High   Group2 3.106060
```

REFERENCES:

https://www.statmethods.net/

CONCLUSION:

Understand different data preprocessing techniques

Assignment: 6

AIM: Perform different data cleaning operations using R/Python

PROBLEM STATEMENT /DEFINITION

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

a. Data cleaning
b. Data integration
c. Data transformation
d. Error correcting
e. Data model building

OBJECTIVE:

To learn data processing methods
To learn building data model

THEORY:

Statistical analysis in five steps In this tutorial a statistical analysis is viewed as the result of a number of data processing steps where each step increases the ``value'' of the data* . Raw data . Technically correct data Consistent data Statistical results Formatted output type checking, normalizing fix and impute estimate, analyze, derive, etc. tabulate, plot data cleaning

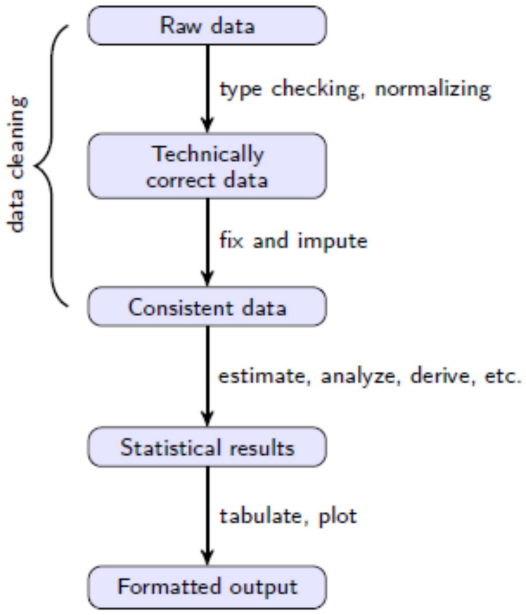


Fig. 1 Statistical analysis value chain

Figure 1 shows an overview of a typical data analysis project. Each rectangle represents data in a certain state while each arrow represents the activities needed to get from one state to the other. The first state (Raw data) is the data as it comes in. Raw data files may lack headers, contain wrong data types (e.g. numbers stored as strings), wrong category labels, unknown or unexpected character encoding and so on. In short, reading such files into an R data.frame directly is either difficult or impossible without some sort of preprocessing. Once this preprocessing has taken place, data can be deemed technically correct. That is, in this state data can be read into an R data.frame, with correct names, types and labels, without further trouble. However, that does not mean that the values are error-free or complete. For example, an age variable may be reported negative, an under-aged person may be registered to possess a driver's license, or data may simply be missing. Such inconsistencies obviously depend on the subject matter * In fact, such a value chain is an integral part of Statistics Netherlands business architecture. An introduction to data cleaning with R 7 that the data pertains to, and they should be ironed out before valid statistical inference from such data can be produced. Consistent data is the stage where data is ready for statistical inference. It is the data that most statistical theories use as a starting point. Ideally, such theories can still be applied without taking previous data cleaning steps into account. In practice however, data cleaning methods like imputation of missing values will influence statistical results and so must be accounted for in the following analyses or interpretation thereof.

1.2.1 Variable types and indexing techniques

If you had to choose to be proficient in just one R-skill, it should be indexing. By indexing we mean all the methods and tricks in R that allow you to select and manipulate data using logical, integer or named indices. Since indexing skills are important for data cleaning, we quickly review vectors, data.frames and indexing techniques. The most basic variable in R is a vector. An R vector is a sequence of values of the same type. All basic operations in R act on vectors (think of the element-wise arithmetic, for example). The basic types in R are as follows. numeric Numeric data (approximations of the real numbers, $\mathbb{R}$) integer Integer data (whole numbers, $\mathbb{Z}$) factor Categorical data (simple classifications, like gender) ordered Ordinal data (ordered classifications, like educational level) character Character data (strings) raw Binary data All basic operations in R work element-wise on vectors where the shortest argument is recycled if necessary. This goes for arithmetic operations (addition, subtraction,…), comparison operators (==, <=,…), logical operators (&, |, !,…) and basic math functions like sin, cos, exp and so on. If you want to brush up your basic knowledge of vector and recycling properties, you can execute the following code and think about why it works the way it does.

```
# vectors have variables of _one_ type
c(1, 2, "three")
# shorter arguments are recycled
(1:3) * 2
(1:4) * c(1, 2)
# warning! (why?)

(1:4) * (1:3)
```

Each element of a vector can be given a name. This can be done by passing named arguments to the c() function or later with the names function. Such names can be helpful giving meaning to your variables. For example compare the vector

```
x <- c("red", "green", "blue")
```

with the one below.

```
capColor = c(huey = "red", duey = "blue", louie = "green")
```

Obviously the second version is much more suggestive of its meaning. The names of a vector need not be unique, but in most applications you'll want unique names (if any). Elements of a vector can be selected or replaced using the square bracket operator [ ]. The square brackets accept either a vector of names, index numbers, or a logical. In the case of a logical, the index is recycled if it is shorter than the indexed vector. In the case of numerical indices, negative indices omit, in stead of select elements. Negative and positive indices are not allowed in the same index vector. You can repeat a name or an index number, which results in multiple instances of the

same value. You may check the above by predicting and then verifying the result of the following statements.

```
capColor["louie"]

names(capColor)[capColor == "blue"]

x <- c(4, 7, 6, 5, 2, 8)

I <- x < 6

J <- x > 7

x[I | J]

x[c(TRUE, FALSE)]

x[c(-1, -2)]
```

Replacing values in vectors can be done in the same way. For example, you may check that in the following assignment

```
x <- 1:10

x[c(TRUE, FALSE)] <- 1
```

every other value of x is replaced with 1. A list is a generalization of a vector in that it can contain objects of different types, including other lists. There are two ways to index a list. The single bracket operator always returns a sub-list of the indexed list. That is, the resulting type is again a list. The double bracket operator ([[ ]]) may only result in a single item, and it returns the object in the list itself. Besides indexing, the dollar operator $ can be used to retrieve a single element. To understand the above, check the results of the following statements.

```
L <- list(x = c(1:5), y = c("a", "b", "c"), z = capColor)

L[[2]]

L$y

L[c(1, 3)]

L[c("x", "y")]
```

```
L[["z"]]
```

Especially, use the class function to determine the type of the result of each statement. A data.frame is not much more than a list of vectors, possibly of different types, but with every vector (now columns) of the same length. Since data.frames are a type of list, indexing them with a single index returns a sub-data.frame; that is, a data.frame with less columns. Likewise, the dollar operator returns a vector, not a sub-data.frame. Rows can be indexed using two indices in the bracket operator, separated by a comma. The first index indicates rows, the second indicates columns. If one of the indices is left out, no selection is made (so everything is returned). It is important to realize that the result of a two-index selection is simplified by R as much as possible. Hence, selecting a single column using a two-index results in a vector. This behaviour may be switched off using drop=FALSE as an extra parameter. Here are some short examples demonstrating the above.

```
d <- data.frame(x = 1:10, y = letters[1:10], z = LETTERS[1:10])

d[1]

d[, 1]

d[, "x", drop = FALSE]

d[c("x", "z")]

d[d$x > 3, "y", drop = FALSE]

d[2, ]
```

1.2.2 Special values Like most programming languages, R has a number of Special values that are exceptions to the normal values of a type. These are NA, NULL, ±Inf and NaN. Below, we quickly illustrate the meaning and differences between them. NA Stands for not available. NA is a placeholder for a missing value. All basic operations in R handle NA without crashing and mostly return NA as an answer whenever one of the input arguments is NA. If you understand NA, you should be able to predict the result of the following R statements.

```
NA + 1

sum(c(NA, 1, 2))

median(c(NA, 1, 2, 3), na.rm = TRUE)

length(c(NA, 2, 3, 4))

3 == NA

NA == NA

TRUE | NA
```

The function is.na can be used to detect NA's. NULL You may think of NULL as the empty set from mathematics. NULL is special since it has no class (its class is NULL) and has length 0 so it does not take up any space in a vector. In particular, if you understand NULL, the result of the following statements should be clear to you without starting R.

```
length(c(1, 2, NULL, 4))

sum(c(1, 2, NULL, 4))

x <- NULL

c(x, 2)
```

The function is.null can be used to detect NULL variables. Inf Stands for infinity and only applies to vectors of class numeric. A vector of class integer can never be Inf. This is because the Inf in R is directly derived from the international standard for floating point arithmetic 1 . Technically, Inf is a valid numeric that results from calculations like division of a number by zero. Since Inf is a numeric, operations between Inf and a finite numeric are well-defined and comparison operators work as expected. If you understand Inf, the result of the following statements should be clear to you.

```
pi/0

2 * Inf

Inf - 1e+10

Inf + Inf

3 < -Inf

Inf == Inf
```

NaN Stands for not a number. This is generally the result of a calculation of which the result is unknown, but it is surely not a number. In particular operations like 0/0, Inf-Inf and Inf/Inf result in NaN. Technically, NaN is of class numeric, which may seem odd since it is used to indicate that something is not numeric. Computations involving numbers and NaN always result in NaN, so the result of the following computations should be clear.

```
NaN + 1

exp(NaN)
```

The function is.nan can be used to detect NaN's.

c.Data Transformations

A number of reasons can be attributed to when a predictive model crumples such as:

- Inadequate data pre-processing
- Inadequate model validation
- Unjustified extrapolation
- Over-fitting

(Kuhn, 2013)

Before we dive into data preprocessing, let me quickly define a few terms that I will be commonly using.

- *Predictor/Independent/Attributes/Descriptors* – are the different terms that are used as input for the prediction equation.
- *Response/Dependent/Target/Class/Outcome* – are the different terms that are referred to the outcome event that is to be predicted.

In this article, I am going to summarize some common data pre-processing approaches with examples in R

1. Centering and Scaling

Variable centering is perhaps the most intuitive approach used in predictive modeling. To center a predictor variable, the average predictor value is subtracted from all the values. as a result of centering, the predictor has zero mean.
To scale the data, each predictor value is divided by its standard deviation (sd). This helps in coercing the predictor value to have a *sd* of one. Needless to mention, centering and scaling will work for continuous data.
The drawback of this activity is loss of interpretability of the individual values.
An R example:

```
# Load the default datasets
> library(datasets)
> data(mtcars)
> dim(mtcars)
32 11
> str(mtcars)
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
> cov(mtcars$disp, mtcars$cyl) # check for covariance
[1] 199.6603
> mtcars$disp.scl<-scale(mtcars$disp, center = TRUE, scale = TRUE)  > mtcars$cyl.scl<- s
check for covariance in scaled data
```

```
          [,1]
[1,] 0.9020329
```

2. Resolving Skewness

Skewness is a measure of shape. A common appraoch to check for skewness is to plot the predictor variable. As a rule, negative skewness indicates that the mean of the data values is less than the median, and the data distribution is left-skewed. Positive skewness would indicates that the mean of the data values is larger than the median, and the data distribution is right-skewed.

- If the skewness of the predictor variable is 0, the data is perfectly symmetrical,
- If the skewness of the predictor variable is less than -1 or greater than +1, the data is highly skewed,
- If the skewness of the predictor variable is between -1 and -1/2 or between +1 and +1/2 then the data is moderately skewed,
- If the skewness of the predictor variable is -1/2 and +1/2, the data is approximately symmetric.

I will use the function skewness from the e1071 package to compute the skewness coefficient
An R example:

```
> library(e1071)
> engine.displ<-skewness(mtcars$disp) > engine.displ
[1] 0.381657
```

So the variable displ is moderately positively skewed.

3. Resolving Outliers
The outliers package provides a number of useful functions to systematically extract outliers. Some of these are convenient and come handy, especially the outlier() and scores() functions.
Outliers
The function outliers() gets the extreme most observation from the mean.
If you set the argument opposite=TRUE, it fetches from the other side.
An R example:

```

> set.seed(4680) # for code reproducibility
> y<- rnorm(100) # create some dummy data > library(outliers) # load the library
> outlier(y)
[1] 3.581686
> dim(y)<-c(20,5) # convert it to a matrix > head(y,2)# Look at the first 2 rows of t
          [,1]       [,2]      [,3]      [,4]       [,5]
[1,] 0.5850232  1.7782596  2.051887  1.061939 -0.4421871
[2,] 0.5075315 -0.4786253 -1.885140 -0.582283  0.8159582
> outlier(y) # Now, check for outliers in the matrix
[1] -1.902847 -2.373839  3.581686  1.583868  1.877199
> outlier(y, opposite = TRUE)
[1]  1.229140  2.213041 -1.885140 -1.998539 -1.571196

```

There are two aspects the the scores() function.
Compute the normalised scores based on “z”, “t”, “chisq” etc
Find out observations that lie beyond a given percentile based on a given score.

```
> set.seed(4680)
> x = rnorm(10)
> scores(x)  # z-scores => (x-mean)/sd
[1]  0.9510577  0.8691908  0.6148924 -0.4336304 -1.6772781...
> scores(x, type="chisq")  # chi-sq scores => (x - mean(x))^2/var(x)
[1] 0.90451084 0.75549262 0.37809269 0.18803531 2.81326197 . . .
> scores(x, type="t")  # t scores
[1]  0.9454321  0.8562050  0.5923010 -0.4131696 -1.9073009
> scores(x, type="chisq", prob=0.9)  # beyond 90th %ile based on chi-sq
[1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
> scores(x, type="chisq", prob=0.95)  # beyond 95th %ile
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> scores(x, type="z", prob=0.95)  # beyond 95th %ile based on z-scores
[1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
> scores(x, type="t", prob=0.95)  # beyond 95th %ile based on t-scores
[1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
```

3.1 Outlier Treatment
Once the outliers are identified, you may rectify it by using one of the following approaches.

- A. Imputation

  Imputation with mean / median / mode.

- B. Capping

  For missing values that lie outside the 1.5 * IQR limits, we could cap it by replacing those observations outside the lower limit with the value of 5th %ile and those that lie above the upper limit, with the value of 95th %ile. For example, it can be done like this as shown;

```
> par(mfrow=c(1, 2)) # for side by side plotting
> x <- mtcars$mpg > plot(x)
> qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
> caps <- quantile(x, probs=c(.05, .95), na.rm = T)
> H <- 1.5 * IQR(x, na.rm = T)
> x[x < (qnt[1] - H)] <- caps[1]
> x[x > (qnt[2] + H)] <- caps[2]
> plot(x)
```

4. Missing value treatment

- A. Impute Missing values with median or mode
- B. Impute Missing values based on K-nearest neighbors

Use the library DMwR or mice or rpart. If using DMwR, for every observation to be imputed, it identifies 'k' closest observations based on the euclidean distance and computes the weighted average (weighted based on distance) of these 'k' obs. The advantage is that you could impute all the missing values in all variables with one call to the function. It takes the whole data frame as the argument and you don't even have to specify which variabe you want to impute. But be cautious not to include the response variable while imputing.

There are many other types of transformations like treating colinearity, dummy variable encoding, covariance treatment

REFERENCES:

https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf

https://www.r-bloggers.com/data-transformations/

CONCLUSION:

Understand different data preprocessing techniques

Assignment: 7

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

1) Text mining in R

2) Data analysis using the Map Reduce in R

Theory:

Text Mining:

Natural languages (English, Hindi, Mandarin etc.) are different from programming languages. The semantic or the meaning of a statement depends on the context, tone and a lot of other factors. Unlike programming languages, natural languages are ambiguous.

Text mining deals with helping computers understand the "meaning" of the text. Some of the common text mining applications include sentiment analysis e.g if a Tweet about a movie says something positive or not, text classification e.g classifying the mails you get as spam or ham etc.

R is succinctly described as “a language and environment for statistical computing and graphics,” which makes it worth knowing if you’re dabbling in the data science/art of statistics and exploratory data analysis. R has a wide variety of useful packages.

In R the packages useful in understanding and extracting insights from the text and text mining packages are as follow:

1. RSQLite, ‘SQLite’ Interface for R
2. tm, framework for text mining applications
3. SnowballC, text stemming library
4. Wordcloud, for making wordcloud visualizations
5. Syuzhet, text sentiment analysis
6. ggplot2, one of the best data visualization libraries
7. quanteda, N-grams

Text preprocessing

Before we dive into analyzing text, we need to preprocess it. Text data contains white spaces, punctuations, stop words etc. These characters do not convey much information and are hard to process. For example, English stop words like “the”, “is” etc. do not tell you much information about the sentiment of the text, entities mentioned in the text, or relationships between those entities. Depending upon the task at hand, we deal with such characters differently. This will help isolate text mining in R on important words.

- Convert the text to lower case, so that words like “write” and “Write” are considered the same word for analysis
- Remove numbers
- Remove English stopwords e.g “the”, “is”, “of”, etc
- Remove punctuation e.g “,”, “?”, etc
- Eliminate extra white spaces
- Stemming our text

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. E.g changing “car”, “cars”, “car’s”, “cars’” to “car”. This can also help with different verb tenses with the same semantic meaning such as digs, digging, and dig. One very useful library to perform the aforementioned steps and text mining in R is the “tm” package. The main structure for managing documents in tm is called a Corpus, which represents a collection of text documents.

Cleaning text in R

```
# Transform and clean the text
library("tm")
docs <- Corpus(VectorSource(emailRaw)) # emailRaw is sample file name
```

Transformations are done via the **tm_map() function** which applies a function to all elements of the corpus. Basically, all transformations work on single text documents and tm_map() just applies them to all documents in a corpus.

A **document term matrix** is an important representation for text mining in R tasks and an important concept in text analytics. Each row of the matrix is a document vector, with one column for every term in the entire corpus.

Naturally, some documents may not contain a given term, so this matrix is sparse. The value in each cell of the matrix is the term frequency.

tm makes it very easy to create the term-document matrix. With the document term matrix made, we can then proceed to build a word cloud for Hillary's emails, highlighting which words the most are frequently made.

WORD CLOUD

A word cloud is a simple yet informative way to understand textual data and to do text analysis.

Word Cloud is another way of representing the frequency of terms in a document. Here, the size of a word indicates its frequency in the document corpus. Load the wordcloud package, as follows:

library(wordcloud)

For Word Cloud comprising terms with a frequency greater than 30, use the following command:

wordcloud(names(freq), freq, min.freq=30,colors=brewer.pal(3,"Dark2"))

For a Word Cloud for the 50 words that occur most often, use the command given below:

wordcloud(names(freq), freq, max.words=50,colors=brewer.pal(6,"Dark2"))

Text processing is about extracting useful information from text, which includes basic steps of pre-processing data, stemming the data, representing the corpus using the document term matrix and obtaining the associations between terms. R provides several libraries and functions to efficiently carry out these tasks.

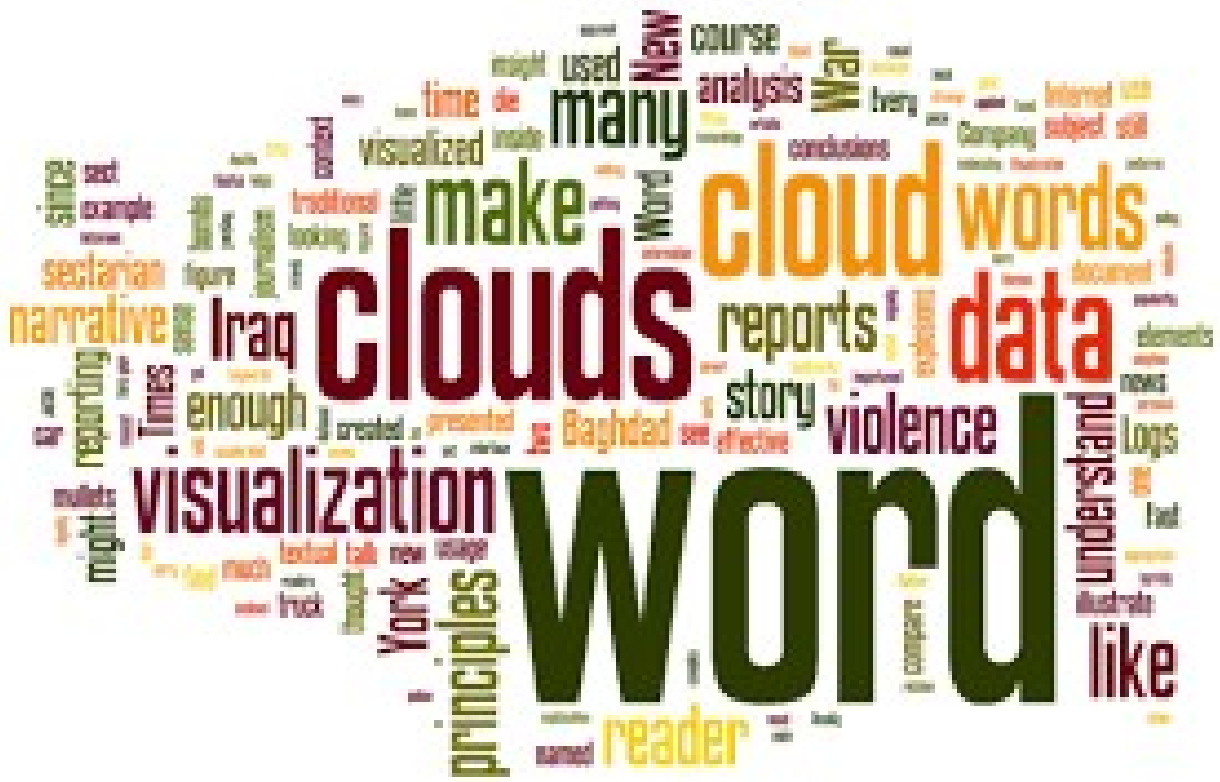


Fig 1. Word Cloud

Assignment: 8

Aim: Visualize the data using R/Python by plotting the graphs for assignment no. 2 and 3

OBJECTIVE:

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

2. To study detailed data visualization techniques in R programing.

SOFTWARE REQUIREMENTS:

1. Ubuntu 16.04 / 18.04

2. R Studio

THEORY:

1) 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”)

42

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 –

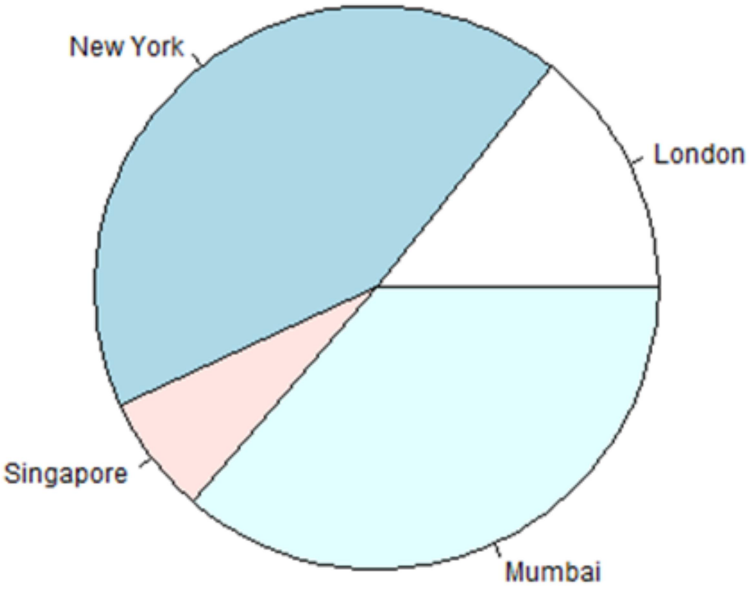


Fig 1. Pie Chart

2) 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 –

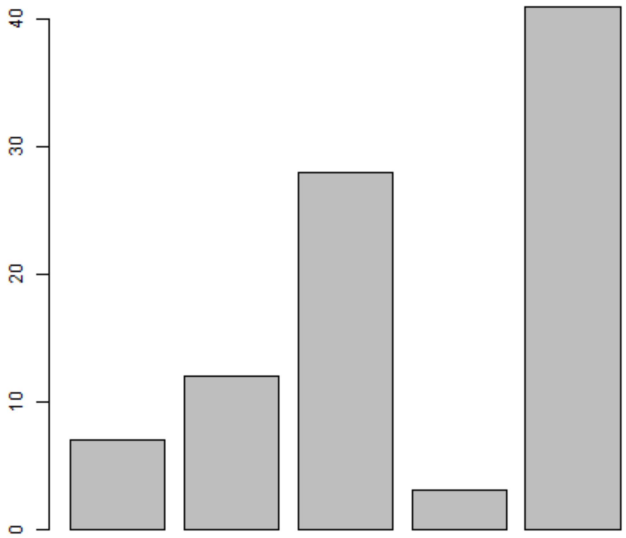


Fig. 2. Bar Chart

3) 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).

45

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 –

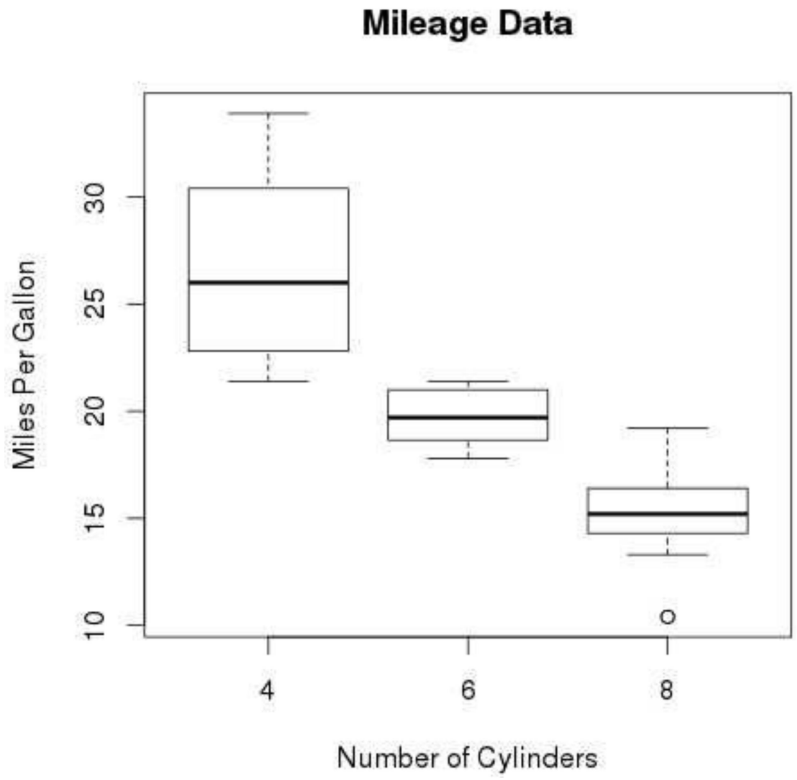


Fig. 3. Box Plot

4) 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 –

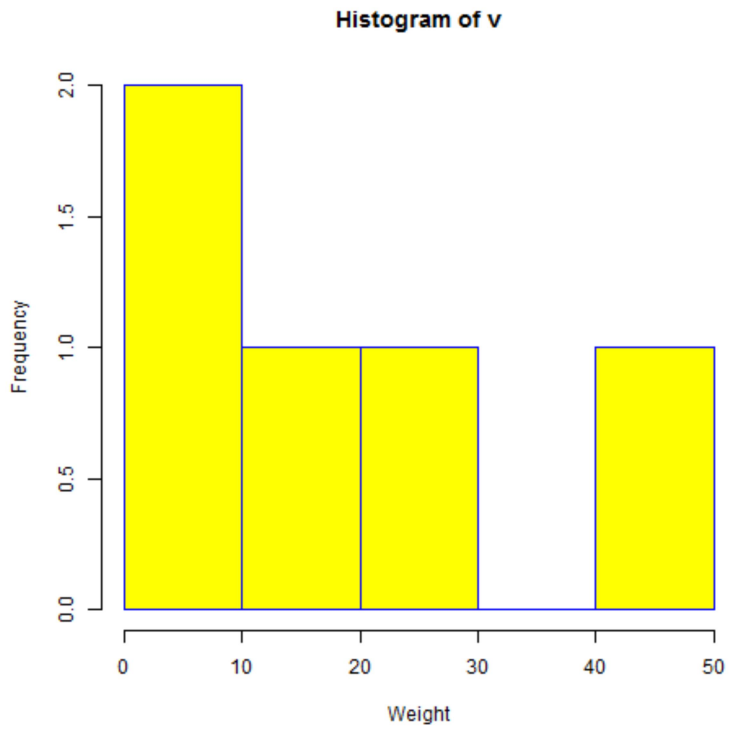


Fig. 4 Hostogram

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 –

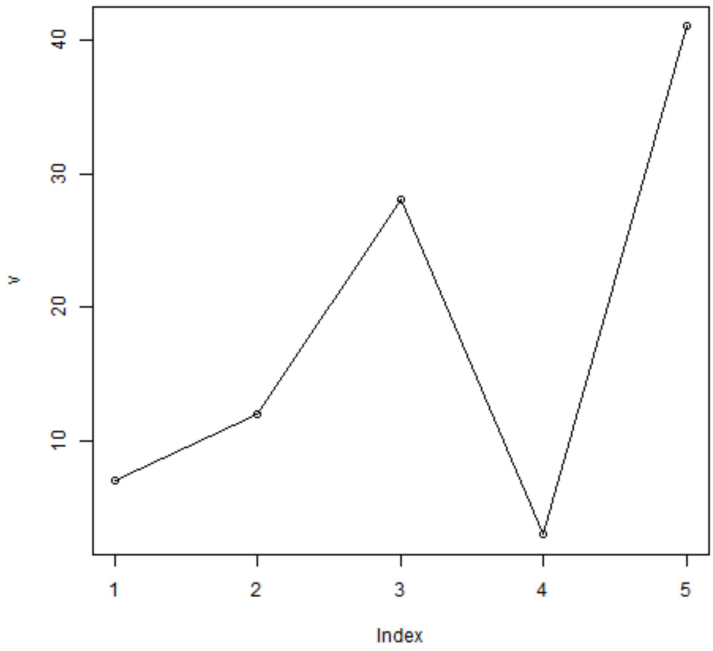


Fig. 5. Line Chart

6) R – Scatter plots

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 scatter plot 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 Scatter plot

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 –

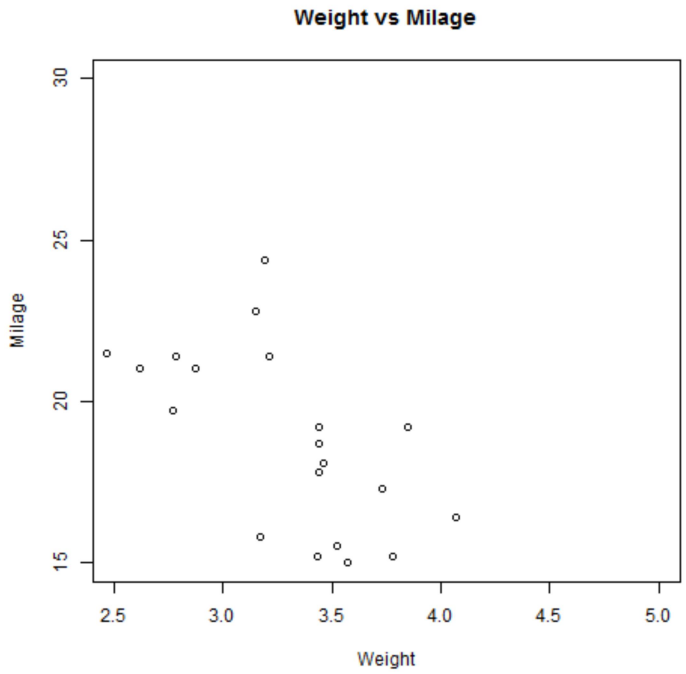


Fig. 6 Scatter Plot

7) Scatter plot 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.

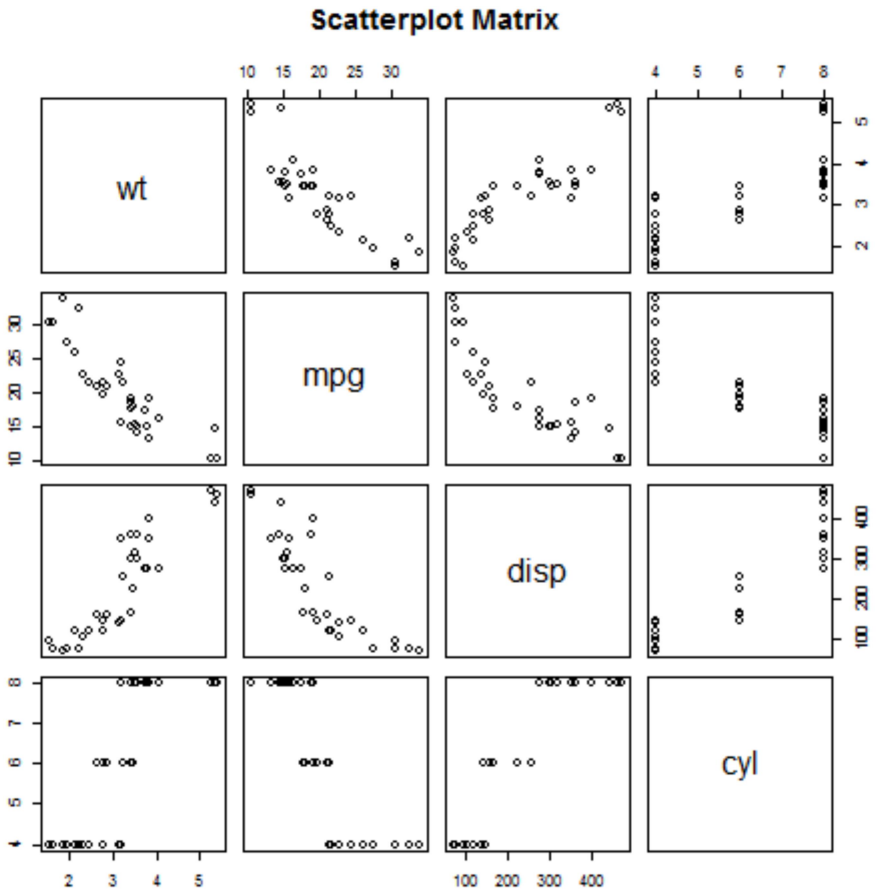


Fig. 7 Scatter Plot Matrix

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

Assignment: 9

AIM: Perform different data visualization operations using Tableau

PROBLEM STATEMENT /DEFINITION

Perform the following data visualization operations using Tableau on Adult, Iris datasets or retail dataset.

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

OBJECTIVE:

To learn specialized data visualization tools.
To learn different types of data visualization techniques.

SOFTWARE REQUIREMENTS:

1. Ubuntu 16.04 / 18.04

2. Tableau

THEORY:

Introduction:

Data visualization or data visualization is viewed by many disciplines as a modern equivalent of visual communication. It involves the creation and study of the visual representation of data, meaning "information that has been abstracted in some schematic form, including attributes or variables for the units of 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 or 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**

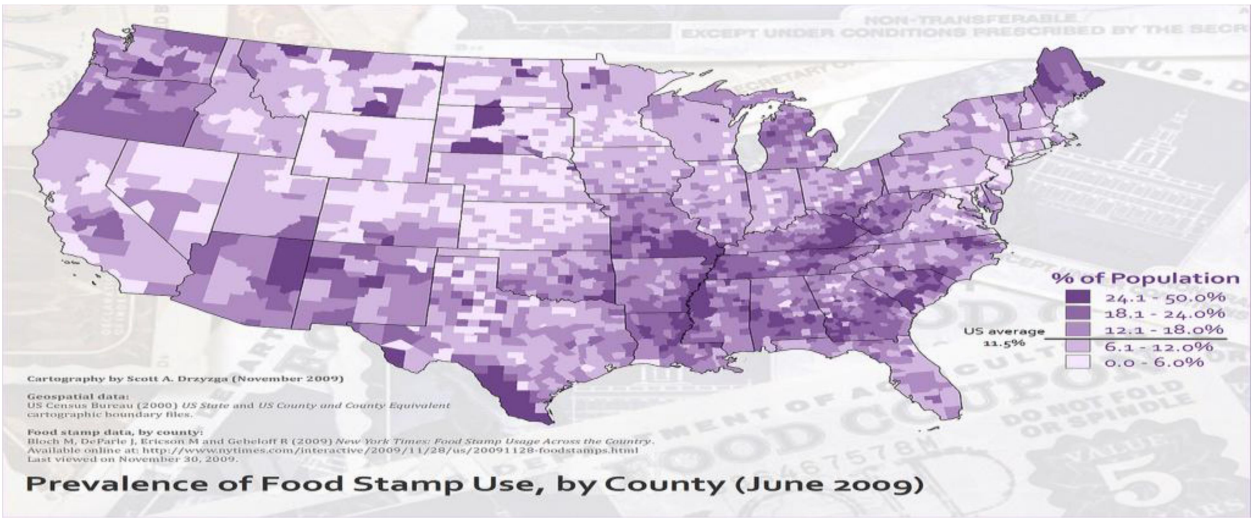


Fig. 1. Geospatial Graph

3D/Volumetric:

Broadly, examples of scientific visualization:

- 3D computer models

  In 3D computer graphics, **3D modeling** (or **three-dimensional modeling**) is the process of developing a mathematical representation of any *surface* of an object (either inanimate or living) in three dimensions via specialized 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* or used in a computer simulation of physical phenomena. The model can also be physically created using 3D printing devices.

- surface and volume rendering

Rendering is the process of generating an image from a 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, lighting, and shading information. The image is a digital image or raster graphics 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 is a technique used to display a 2D projection of a 3D discretely sampled data set. A typical 3D data set is a group of 2D slice images acquired by a CT or MRI scanner. Usually these are acquired in a regular pattern (e.g., one slice every millimeter) and usually have a regular number of image pixels in a regular pattern. This is an example of a regular volumetric grid, with each volume element, or voxel represented by a single value that is obtained by sampling the immediate area surrounding the voxel.

- Computer Simulations

Computer simulation is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system. Computer simulations have become a useful part of mathematical modeling 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] The simultaneous visualization and simulation of a system is called visualization.

Temporal

Examples:

- **Timeline**

Fig.2 Timeline Graph

Tools: SIMILE Timeline, TimeFlow, Timeline JS, Excel

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:**

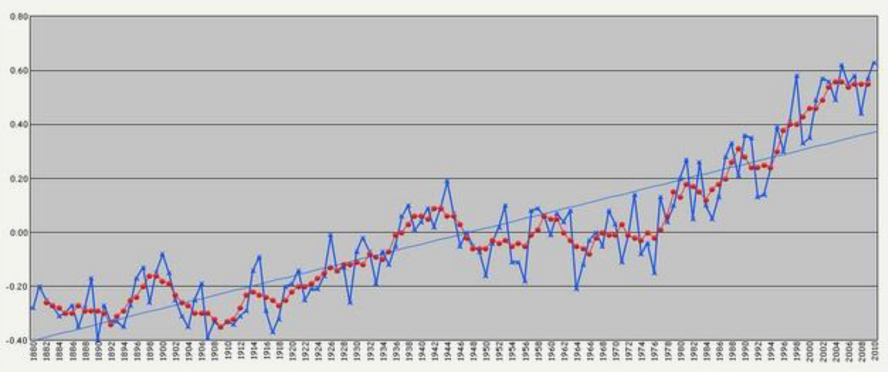


Fig.3Time Series

Multidimensional:

Examples (category proportions, counts):

- **Histogram**

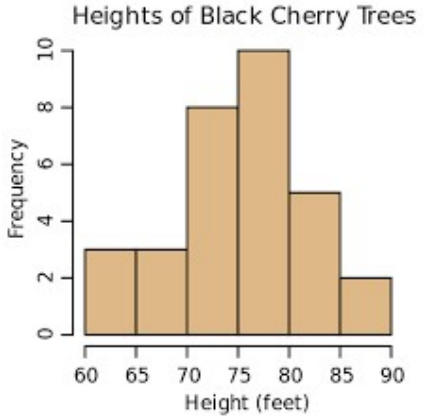


Fig. 4 Histogram

- **pie chart**

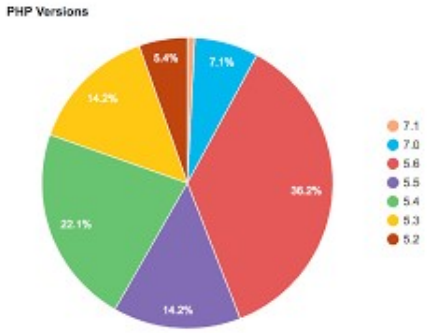


Fig. 5 Pie Chart

Tree/Hierarchical

Examples:

- **general tree visualization**

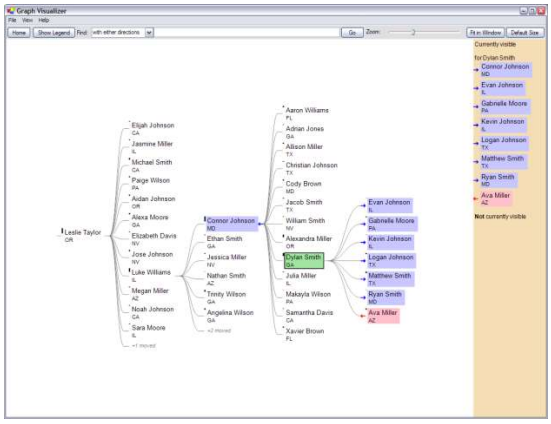


Fig. 6 Tree Graph

- **Hierarchical Visualization:**

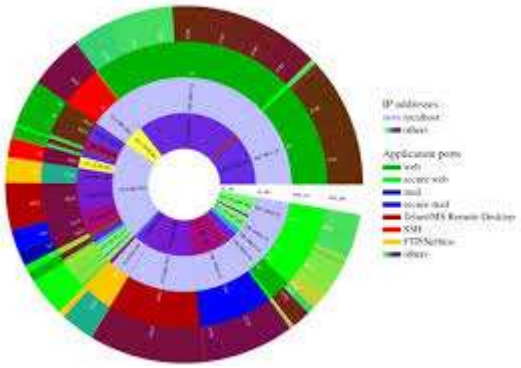


Fig. 7 Hierarchical Graph

- **Dendrogram:**

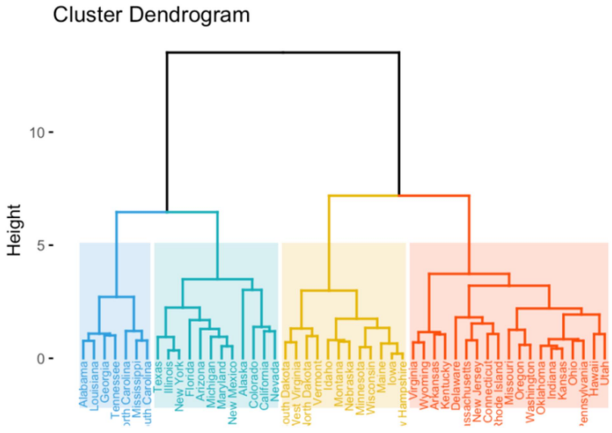


Fig. 8 Dendogram

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

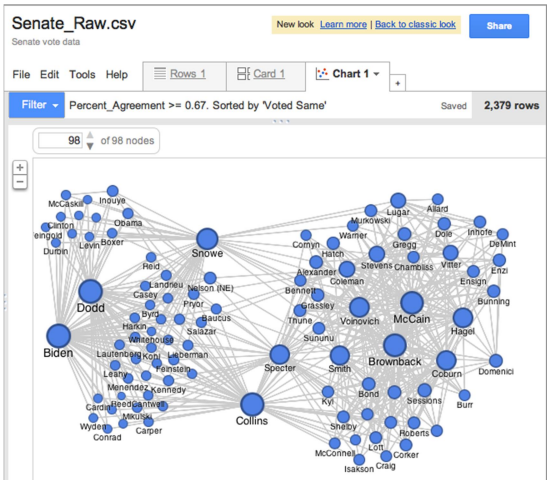


Fig. 9 Network Graph

  - **matrix:**

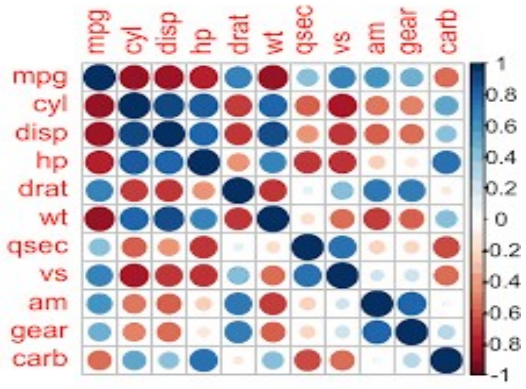


Fig. 10 Matrix Graph

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

Fig. 11 Node Link Graph

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.

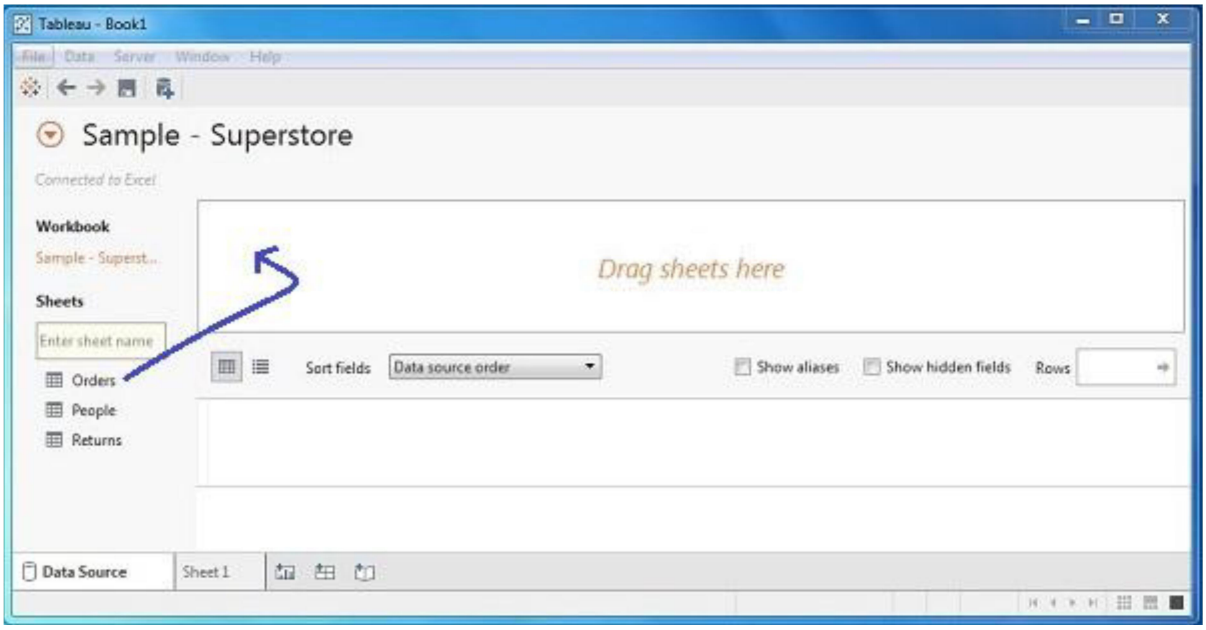
These three steps are −

- **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.**

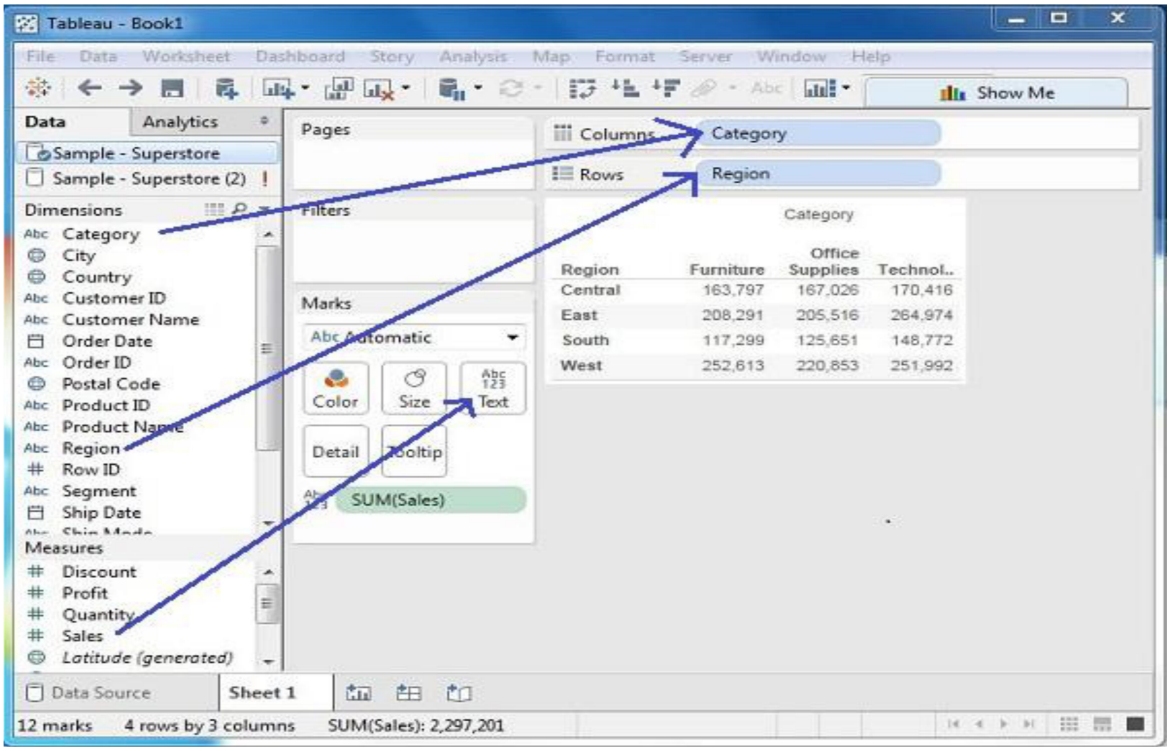
Connect to a Data Source

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

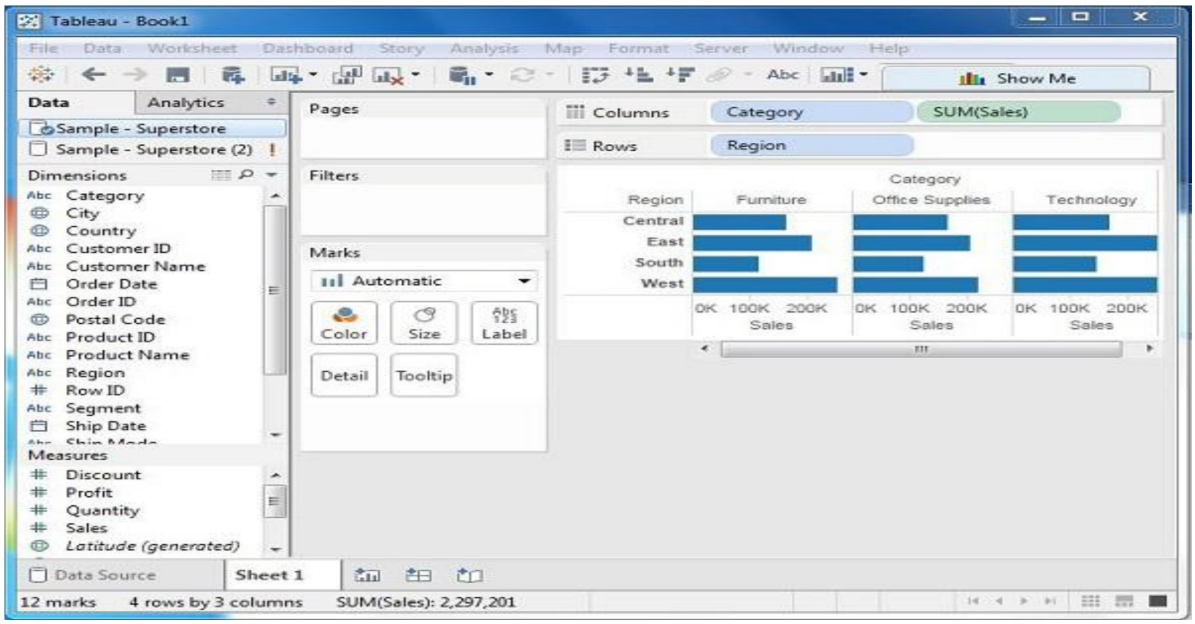
Next, choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help visualize the performance of the dimensional data with respect to the data which are measures. Choose **Category** and **Region** 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.



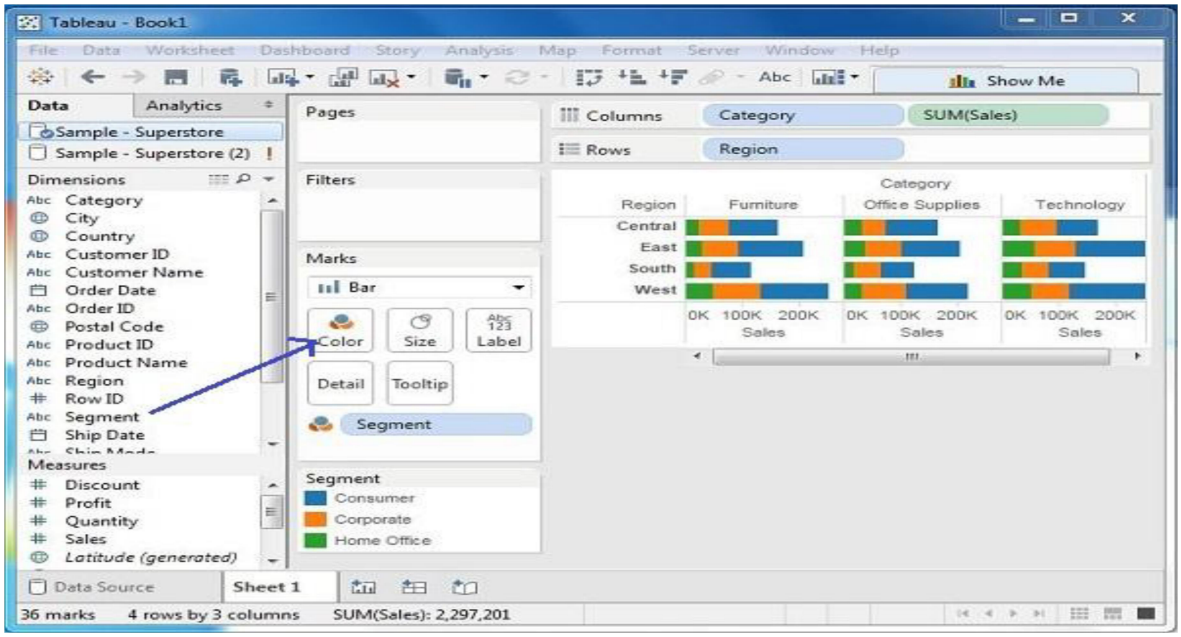
Apply Visualization Technique

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 Assignments

Assignment: 10

Aim: Case study of any one social media analytics tool

Theory:

Social media analytics is the process of gathering and analyzing data from social networks such as Facebook, Instagram, and Twitter. It is commonly used by marketers to track online conversations about products and companies. **Social media analytics** also defined as, "the art and science of extracting valuable hidden insights from vast amounts of semi-structured and unstructured social media data to enable informed and insightful decision making."

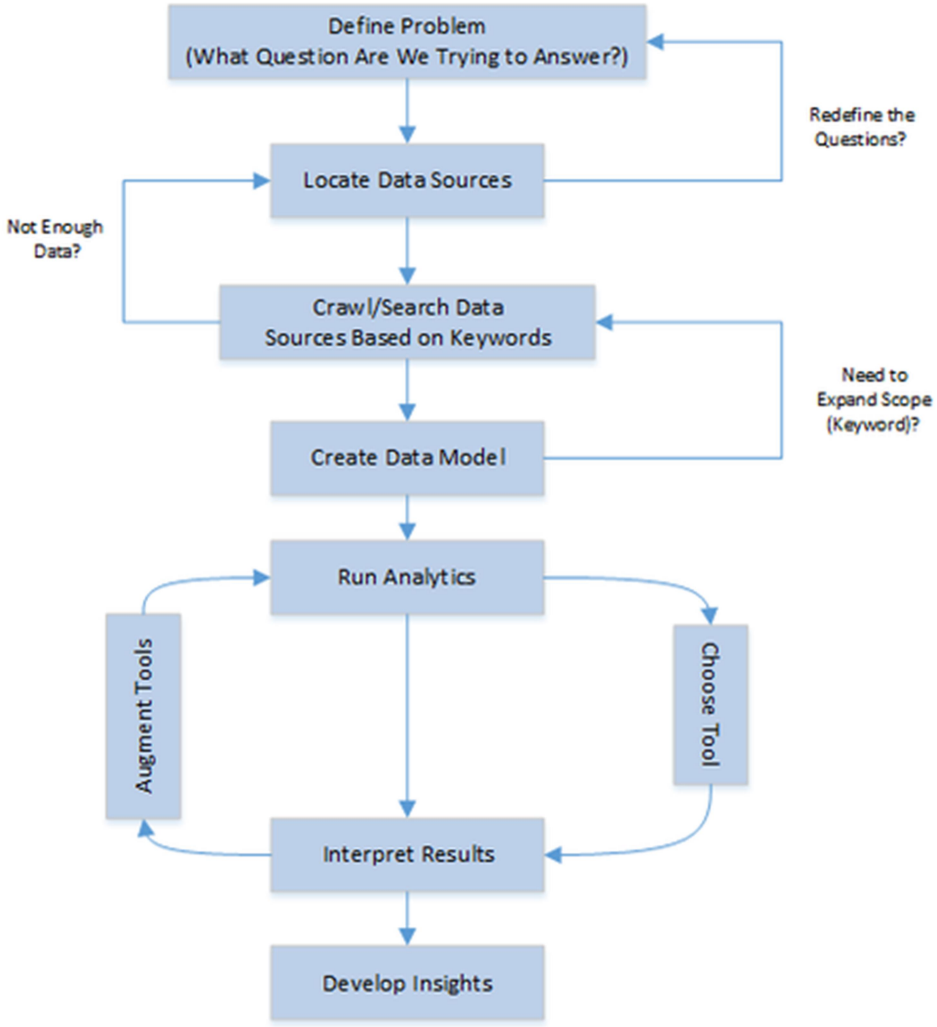


Fig 1 Social Media Analytics Process

Students need to download / use any one of the social media analytics tool and prepare report.

Few sample free and paid social media analytics tools.

1. Buffer Analyze
2. Sprout Social
3. Hootsuite
4. Zoho Social
5. Sendible
6. Keyhole
7. Rival IQ
8. Social Report
9. Socialbakers
10. Iconosquare
11. Tailwind
12. Likealyzer
13. Cyfe
14. Union Metrics
15. Quintly
16. Followerwonk
17. SparkToro
18. Audiense
19. Klear
20. Talkwalker
21. Google Analytics

Apart from these third party social media tools, we have the social networks' own analytics toolkits:

1. Facebook Insights
2. Instagram Insights
3. Twitter Analytics
4. Pinterest Analytics
5. LinkedIn Analytics
6. YouTube Analytics

Conclusion: Thus we have understood the practical use of analytical tool.

Assignment: 11

Aim: Case study of any one text mining / text analytics tool

Theory:

Text mining is "the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources." Written resources can be websites, books, emails, reviews, articles.

High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output. 'High quality' in text mining usually refers to some combination of relevance, novelty, and interest. Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling (*i.e.*, learning relations between named entities).

Text Analytics:

The term **text analytics** describes a set of linguistic, statistical, and machine learning techniques that model and structure the information content of textual sources for business intelligence, exploratory data analysis, research, or investigation.

Text Analytics Processes:

- Dimensionality reduction is important technique for pre-processing data. Technique is used to identify the root word for actual words and reduce the size of the text data.
- Information retrieval or identification of a corpus is a preparatory step: collecting or identifying a set of textual materials, on the Web or held in a file system, database, or content corpus manager, for analysis.
- Although some text analytics systems apply exclusively advanced statistical methods, many others apply more extensive natural language processing, such as part of speech tagging, syntactic parsing, and other types of linguistic analysis.
- Named entity recognition is the use of gazetteers or statistical techniques to identify named text features: people, organizations, place names, stock ticker symbols, certain abbreviations, and so on.
- Disambiguation—the use of contextual clues—may be required to decide where, for instance, "Ford" can refer to a former U.S. president, a vehicle manufacturer, a movie star, a river crossing, or some other entity.
- Recognition of Pattern Identified Entities: Features such as telephone numbers, e-mail addresses, and quantities (with units) can be discerned via regular expression or other pattern matches.
- Document clustering: identification of sets of similar text documents.
- Co-references: identification of noun phrases and other terms that refer to the same object.

- Relationship, fact, and event Extraction: identification of associations among entities and other information in text
- Sentiment analysis involves discerning subjective (as opposed to factual) material and extracting various forms of attitudinal information: sentiment, opinion, mood, and emotion. Text analytics techniques are helpful in analyzing sentiment at the entity, concept, or topic level and in distinguishing opinion holder and opinion object.
- Quantitative text analysis is a set of techniques stemming from the social sciences where either a human judge or a computer extracts semantic or grammatical relationships between words in order to find out the meaning or stylistic patterns of, usually, a casual personal text for the purpose of psychological profiling etc.

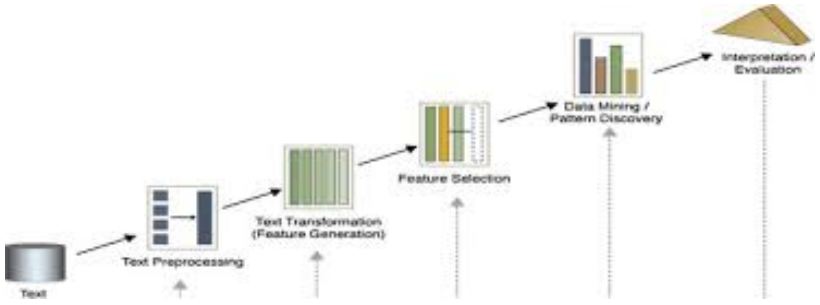


Fig. 1 Text mining process

Students need to download / use any one of the social media analytics tool and prepare report.

Few sample free text mining/ text analytics tools.

- Carrot2 – text and search results clustering framework.
- GATE – general Architecture for Text Engineering, an open-source toolbox for natural language processing and language engineering.
- Gensim - large-scale topic modelling and extraction of semantic information from unstructured text (Python).
- Natural Language Toolkit (NLTK) – a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for the Python programming language.
- OpenNLP - natural language processing.
- Orange with its text mining add-on.
- Stanbol - an open source text mining engine targeted at semantic content management.
- The programming language R provides a framework for text mining applications in the package *tm*.[3] The Natural Language Processing task view contains *tm* and other text mining library packages.[4]
- The KNIME Text Processing extension.
- The PLOS Text Mining Collection.[5]
- Voyant Tools - a web-based text analysis environment, created as a scholarly project.
- spaCy - open-source Natural Language Processing library for Python

Few sample commercial text mining/ text analytics tools.

- Angoss – Angoss Text Analytics provides entity and theme extraction, topic categorization, sentiment analysis and document summarization capabilities via the embedded
- AUTINDEX - is a commercial text mining software package based on sophisticated linguistics by IAI (Institute for Applied Information Sciences), Saarbrücken.
- Autonomy – text mining, clustering and categorization software
- Averbis – provides text analytics, clustering and categorization software, as well as terminology management and enterprise search
- Basis Technology – provides a suite of text analysis modules to identify language, enable search in more than 20 languages, extract entities, and efficiently search for and translate entities.
- DigitalMR - social media listening & text+image analytics tool for market research
- Endeca Technologies – provides software to analyze and cluster unstructured text.
- FICO Score – leading provider of analytics.
- General Sentiment - Social Intelligence platform that uses natural language processing to discover affinities between the fans of brands with the fans of traditional television shows in social media. Standalone text analytics to capture social knowledge base on billions of topics stored to 2004.
- IBM LanguageWare - the IBM suite for text analytics (tools and Runtime).
- IBM SPSS - provider of Modeler Premium (previously called IBM SPSS Modeler and IBM SPSS Text Analytics), which contains advanced NLP-based text analysis capabilities (multi-lingual sentiment, event and fact extraction), that can be used in conjunction with Predictive Modeling. Text Analytics for Surveys provides the ability to categorize survey responses using NLP-based capabilities for further analysis or reporting.
- Inxight – provider of text analytics, search, and unstructured visualization technologies. (Inxight was bought by Business Objects that was bought by SAP AG in 2008).
- Language Computer Corporation – text extraction and analysis tools, available in multiple languages.
- Lexalytics - provider of a text analytics engine used in Social Media Monitoring, Voice of Customer, Survey Analysis, and other applications. Salience Engine. The software provides the unique capability of merging the output of unstructured, text-based analysis with structured data to provide additional predictive variables for improved predictive models and association analysis.
- Linguamatics – provider of natural language processing (NLP) based enterprise text mining and text analytics software, I2E, for high-value knowledge discovery and decision support.
- Mathematica – provides built in tools for text alignment, pattern matching, clustering and semantic analysis. See Wolfram Language, the programming language of Mathematica.
- MATLAB offers Text Analytics Toolbox for importing text data, converting it to numeric form for use in machine and deep learning, sentiment analysis and classification tasks.[1]
- Medallia - offers one system of record for survey, social, text, written and online feedback.
- NetOwl – suite of multilingual text and entity analytics products, including entity extraction, link and event extraction, sentiment analysis, geotagging, name translation, name matching, and identity resolution, among others.

- RapidMiner with its Text Processing Extension – data and text mining software.
- SAS – SAS Text Miner and Teragram; commercial text analytics, natural language processing, and taxonomy software used for Information Management.
- Sketch Engine - a corpus manager and analysis software which providing creating text corpora from uploaded texts or the Web including part-of-speech tagging and lemmatization or detecting a particular website.[2]
- Smartlogic – Semaphore; Content Intelligence platform containing commercial text analytics, natural language processing, rule-based classification, ontology/taxonomy modelling and information visualization software used for Information Management.
- Sysomos - provider social media analytics software platform, including text analytics and sentiment analysis on online consumer conversations.
- WordStat - Content analysis and text mining add-on module of QDA Miner for analyzing large amounts of text data.

Conclusion: Thus we have understood the practical use of text mining / text analytics tool.

Assignment: 12

Aim: Case study of any one mobile analytics tool

Theory:

Mobile web analytics studies the behavior of mobile website visitors in a similar way to traditional web analytics. In a commercial context, mobile web analytics refers to the use of data collected as visitor's access a website from a mobile phone. It helps to determine which aspects of the website work best for mobile traffic and which mobile marketing campaigns work best for the business, including mobile advertising, mobile search marketing, text campaigns, and desktop promotion of mobile sites and services.

Data collected as part of mobile analytics typically includes page views, visits, visitors, and countries, as well as information specific to mobile devices, such as device model, manufacturer, screen resolution, device capabilities, service provider, and preferred user language. This data is typically compared against key performance indicators for performance and return on investment, and is used to improve a website or mobile marketing campaign's audience response.

The majority of modern smartphones are able to browse websites, some with browsing experiences similar to those of desktop computers. The W3C Mobile Web Initiative identifies best practices to help websites support mobile phone browsing. Many companies use these guidelines and mobile-specific code like Wireless Markup Language or HTML5 to optimize websites for viewing on mobile devices.

Students need to download / use any one of the social media analytics tool and prepare report.

Few sample text mining / text analytics tools.

1. Firebase
2. UXCam
3. Mixpanel
4. Flurry
5. Amplitude
6. Countly
7. Apple iOS Analytics
8. Adobe Analytics
9. Facebook Analytics
10. Localytics

Conclusion: Thus we have understood the practical use of mobile analytics tool.

Part D

Mini Project in Data Science

Assignment: 13

AIM: Design and Implement Mini Project in Data Science

PROBLEM STATEMENT /DEFINITION

Design and Implement any Data science Application using R/Python. Obtain Data, Scrub data (Data cleaning), Explore data, Prepare and validate data model and Interpret data (Data Visualization). Visualize data using any visualization tool like Matplotlib, ggplot, Tableau etc. Prepare Project Report.

OBJECTIVE:

1. To explore the Data science project life cycle.
2. To identify need of project and define problem statement.
3. To extract and process data.
4. To interpret and analyze results using data visualization.

Mini Project Report Format:

Abstract

Acknowledgement

List of Tables & Figures

Contents

1. Introduction

1.1 Purpose, Problem statement

1.2 Scope, Objective

1.3 Definition, Acronym, and Abbreviations

1.4 References

2. Literature Survey

2.1 Introduction

2.2 Detail Literature survey

2.4 Findings of Literature survey

3. System Architecture and Design

3.1 Detail Architecture

3.2 Dataset Description

3.3 Detail Phases

3.4 Algorithms

4. Experimentation and Results

4.1 Phase-wise results

4.2 Explanation with example

4.3 Comparison of result with standard

4.4 Accuracy

4.5 Visualization

4.6 Tools used

5. Conclusion and Future scope

5.1 Conclusion

5.2 Future scope

References

Annexure:

A. GUIs / Screen Snapshot of the System Developed
B. Implementation / code