



RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS19P16 - DATA ANALYTICS LAB

LABORATORY RECORD NOTEBOOK

NAME :

BRANCH : COMPUTER SCIENCE AND ENGINEERING

REGISTER NO :

SEMESTER : VII

ACADEMIC YEAR : 2022– 2023



RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

BONAFIDE CERTIFICATE

NAME ACADEMIC YEAR 2022 – 2023

SEMESTER VII BRANCH COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY REGISTER NO.

Certified that this is the bonafide record of work done by the above student in the CS19P16 - DATA ANALYTICS Laboratory during the Academic year 2022 – 2023.

Signature of faculty – in Charge

Submitted for the Practical Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER



RAJALAKSHMI
ENGINEERING COLLEGE
 An AUTONOMOUS Institution
 Affiliated to ANNA UNIVERSITY, Chennai

CS19P16-DATA ANALYTICS LABORATORY

INDEX

S.NO	DATE	TOPIC	Page No.	SIGNATURE
1		Install, configure and run Hadoop and HDFS		
2		Implementation of word count programs using MapReduce		
3		Implementation of Linear Regression		
4		Implementation of Logistic Regression		
5		Implementation of SVM classification techniques		
6		Implementation of Decision tree classification techniques		
7		Implementation of clustering techniques – Hierarchical		
8		Implementation of clustering techniques K – Means		
9		Implement Data visualization techniques and visualize data using any plotting framework 1. Line chart 2.Bar chart 3.Box plot 4.Scatter plot		
10		Implement Data visualization techniques and visualize data using any plotting framework. 1.Area Chart 2.Heat Map 3.Correlogram		
11		Implementation of sample script by using PIG and HIVE command		
12		Implement An Application That Stores Data In MongoDB		

Ex.No 1

INSTALL, CONFIGURE AND RUN HADOOP AND HDFS

Date:

Step by step Hadoop 2.8.0 installation on Windows 10 Prepare:

These software's should be prepared to install Hadoop 2.8.0 on window 10 64 bits.

1) *Download Hadoop2.8.0*

(Link: <http://www.apache.org/dist/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz> OR
<http://archive.apache.org/dist/hadoop/core/hadoop-2.8.0/hadoop-2.8.0.tar.gz>)

2) *Java JDK1.8.0.zip*

(Link: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>)

Set up:

- 1) Check either Java 1.8.0 is already installed on your system or not, use "Javac - version" to check Java version
- 2) If Java is not installed on your system then first install java under "C:\JAVA" Java setup
- 3) Extract files Hadoop 2.8.0.tar.gz or Hadoop-2.8.0.zip and place under "C:\Hadoop-2.8.0" hadoop
- 4) Set the path HADOOP_HOME Environment variable on windows10 (see Step1, 2, 3 and 4 below) hadoop
- 5) Set the path JAVA_HOME Environment variable on windows 10 (see Step 1, 2, 3 and 4 below) java
- 6) Next we set the Hadoop bin directory path and JAVA bin directory path

Configuration

a) File C:/Hadoop-2.8.0/etc/hadoop/core-site.xml, paste below xml paragraph and save this file.

```
<configuration>

  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

b) Rename "mapred-site.xml.template" to "mapred-site.xml" and edit this file C:/Hadoop-2.8.0/etc/hadoop/mapred-site.xml, paste below xml paragraph and save this file.

```
<configuration>

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

c) Create folder "data" under "C:\Hadoop-2.8.0"

1) Create folder "datanode" under "C:\Hadoop-2.8.0\data"

2) Create folder "namenode" under "C:\Hadoop-2.8.0\data\data"

d) Edit file C:\Hadoop-2.8.0/etc/hadoop/hdfs-site.xml, paste below xml paragraph and save this file.

```
<configuration>

  <property>
```

```
<name>dfs.replication</name>

<value>1</value>

</property>

<property>

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

<value>C:\hadoop-2.8.0\data\namenode</value>

</property>

<property>

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

<value>C:\hadoop-2.8.0\data\datanode</value>

</property>

</configuration>
```

e) Edit file C:/Hadoop-2.8.0/etc/hadoop/yarn-site.xml, paste below xml paragraph and save this file.

```
<configuration>

<property>

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

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

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

</property>

</configuration>
```

f) Edit file C:/Hadoop-2.8.0/etc/hadoop/hadoop-env.cmd by closing the command line "JAVA_HOME=%JAVA_HOME%" instead of set "JAVA_HOME=C:\Java" (On C:\java this is path to filejdk.18.0)

Hadoop Configuration

- 7) Download file Hadoop Configuration.zip (Link: <https://github.com/MuhammadBilalYar/HADOOP-INSTALLATION-ON-WINDOW-10/blob/master/Hadoop%20Configuration.zip>)
- 8) Delete file bin on C:\Hadoop-2.8.0\bin, replaced by file bin on file just download (from Hadoop Configuration.zip).
- 9) Open **cmd** and typing command "hdfs namenode -format" .You will see hdfs namenode -format

Testing

- 10) Open cmd and change directory to "C:\Hadoop-2.8.0\sbin" and type "start-all.cmd" to start apache.
- 11) Make sure these apps are running.
 - Namenode
 - Hadoop datanode
 - YARN ResourceManager
 - YARN Node Manager hadoopnodes
- 12) Open:<http://localhost:8088>
- 13) Open:<http://localhost:50070>

RESULT:

EX NO:02
DATE:

IMPLEMENTATION OF WORD COUNT PROGRAMS USING MAPREDUCE

1. *Open cmd in Administrative mode and move to "C:/Hadoop-2.8.0/sbin" and start cluster*
2. *Create an input directory inHDFS.*

hadoop fs -mkdir /input_dir

3. *Copy the input text file named input_file.txt in the input directory(input_dir) ofHDFS.*

hadoop fs -put C:/input_file.txt /input_dir

4. *Verify input_file.txt available in HDFS input directory(input_dir).*
- hadoop fs -ls /input_dir/**

5. *Verify content of the copiedfile.*

hadoop dfs -cat /input_dir/input_file.txt

6. *Run MapReduceClient.jar and also provide input and outdirectories.*

hadoop jar C:/MapReduceClient.jar wordcount /input_dir /output_dir

7. *Verify content for generated output file.*

hadoopdfs -cat /output_dir/*

Some Other useful commands

8) To leave Safe mode

hadoopdfsadmin –safemode leave

9) To delete file from HDFS directory

hadoop fs -rm -r /iutput_dir/input_file.txt

10) To delete directory from HDFS directory

hadoop fs -rm -r /iutput_dir

RESULT:

EX NO: 3

IMPLEMENTATION OF LINEAR REGRESSION

Date :

AIM:

To implement Linear Regression in R studio.

EXPLANATION:

The aim of linear regression is to model a continuous variable Y as a mathematical function of one or more X variable(s), so that we can use this regression model to predict the Y when only the X is known. This mathematical equation can be generalized as follows:

$$Y = \beta_1 + \beta_2 X + \epsilon$$

SOURCE CODE:

```
head(cars)
scatter.smooth (x=cars$speed, y=cars$dist, main = "LinearRegression")
linearMod<-lm(dist~speed,cars)
print(linearMod)
summary(linearMod)
```

OUTPUT :

```
Call:
lm(formula = dist ~ speed, data = cars)

Coefficients:
(Intercept)      speed 
   -17.579       3.932 

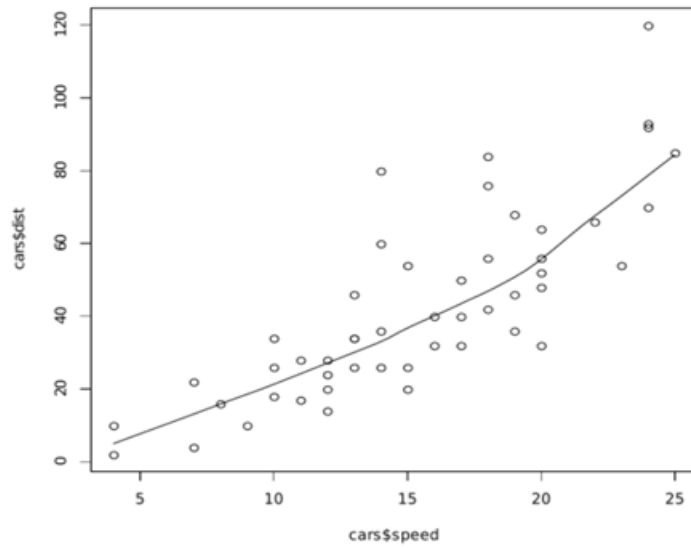
Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min       1Q   Median       3Q      Max 
-29.069  -9.525  -2.272   9.215  43.201 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791    6.7584   -2.601  0.0123 *
speed         3.9324    0.4155    9.464 1.49e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438 
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

PLOT OUTPUT :



RESULT :

EX NO: 4

IMPLEMENTATION OF LOGISTIC REGRESSION

DATE :

AIM:

To implement a Logistic Regression model in R studio.

LOGISTIC REGRESSION:

- The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of the response variable based on the mathematical equation relating it with the predictor variables.
- The general mathematical equation for logistic regression is

$$- y = 1/(1+e^{-(a+b_1x_1+b_2x_2+b_3x_3+\dots)})$$

- Following are the description of the parameters used –
 - y is the response variable.
 - x is the predictor variable.
 - a and b are the coefficients which are numeric constants.
- The function used to create the regression model is the glm() function.

SOURCE CODE:

```
# Select some columns from mtcars
input <- mtcars[,c("am","cyl","hp","wt")]
am.data = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)
print(summary(am.data))
```

OUTPUT:

```
Call:
glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.17272  -0.14907  -0.01464   0.14116   1.27641

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) 19.70288    8.11637   2.428  0.0152 *
cyl          0.48760    1.07162   0.455  0.6491
hp           0.03259    0.01886   1.728  0.0840 .
wt          -9.14947    4.15332  -2.203  0.0276 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.2297  on 31  degrees of freedom
Residual deviance:  9.8415  on 28  degrees of freedom
AIC: 17.841

Number of Fisher Scoring iterations: 8
```

RESULT:

EX No: 5

IMPLEMENTATION OF SVM CLASSIFICATION

DATE :

TECHNIQUES

AIM:

To implement a Support Vector Machine (SVM) classification technique using R.

EXPLANATION:

- Iris is a dataset with 5 Variables namely Sepal.Length, Sepal.Width, Petal.Length, Petal.Width and Species.
- Import the dataset 'iris'.
- Plot the graph against Petal.Length, Petal.Width and Species.
- Import library 'e1071' which contains the SVM function.
- Build the SVM model
- Draw the SVM classification plot
- Display the confusion matrix

SOURCE CODE

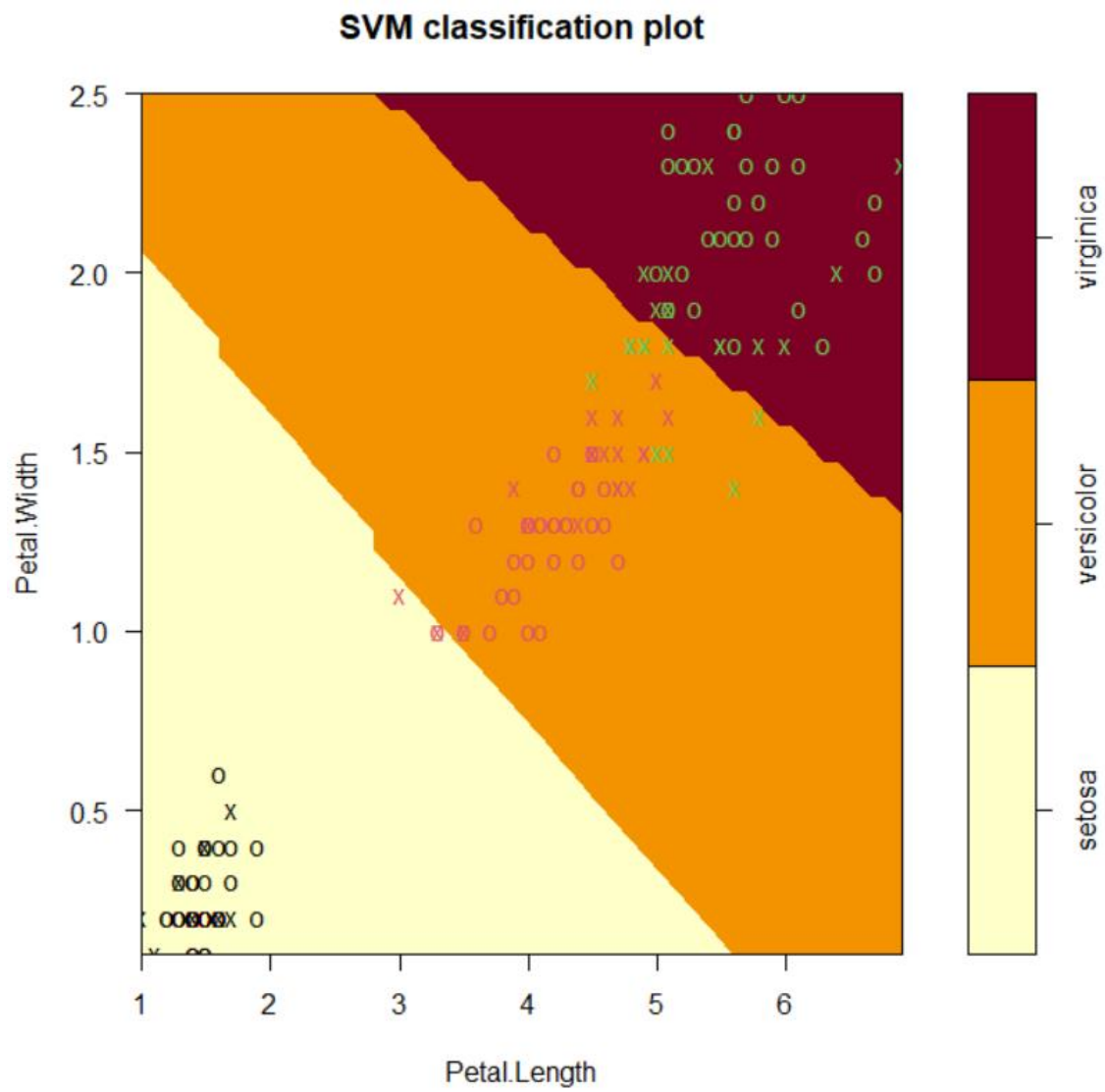
```
library("e1071")
data(iris)
svm_model<- svm(Species ~ ., data=iris, kernel="radial")
plot(svm_model, data=iris, Petal.Width~Petal.Length, slice =list(Sepal.Width=3, Sepal.Length=4))
pred = predict(svm_model, iris)
tab = table(Predicted=pred, Actual = iris$Species)
tab
```

OUTPUT:

	Actual		
Predicted	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	2
virginica	0	2	48

Degree 5

	truth		
predict	setosa	versicolor	virginica
setosa	11	0	0
versicolor	0	21	3
virginica	0	0	10



RESULT:

EX No :6

DATE :

IMPLEMENTATION OF DECISION TREE CLASSIFICATION TECHNIQUES

AIM:

To implement a decision tree classification technique for gender classification using python.

EXPLANATION:

- Import tree from sklearn.
- Call the function DecisionTreeClassifier() from tree
- Assign values for X and Y.
- Call the function predict for Predicting on the basis of given random values for each given feature.
- Display the output.

SOURCE CODE:

```
from sklearn import tree
#Using DecisionTree classifier for prediction
clf = tree.DecisionTreeClassifier()

#Here the array contains three values which are height,weight and shoe size
X = [[181, 80, 91], [182, 90, 92], [183, 100, 92], [184, 200, 93], [185, 300, 94], [186, 400, 95],
[187, 500, 96], [189, 600, 97], [190, 700, 98], [191, 800, 99], [192, 900, 100], [193, 1000, 101]]
Y = ['male', 'male', 'female', 'male', 'female', 'male', 'female', 'male', 'female', 'male', 'female',
'male' ]
clf = clf.fit(X, Y)

#Predicting on basis of given random values for each given feature
predictionf = clf.predict([[181, 80, 91]])
predictionm = clf.predict([[183, 100, 92]])

#Printing final prediction
print(predictionf)
print(predictionm)
```


OUTPUT:

['male']

['female']

RESULT:

EX No : 7

IMPLEMENTATION OF CLUSTERING TECHNIQUES

DATE :

HIERARCHICAL

AIM:

To implement a hierarchical clustering technique using R.

EXPLANATION:

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. The similar objects are grouped together into a cluster, and each cluster is distinct from each other.

- Import the iris dataset.
- Call the function “hclust” using complete linkage.
- Obtain the dendrogram.
- Compute agglomerative coefficients.
- Plot the dendrogram.
- Display the output.

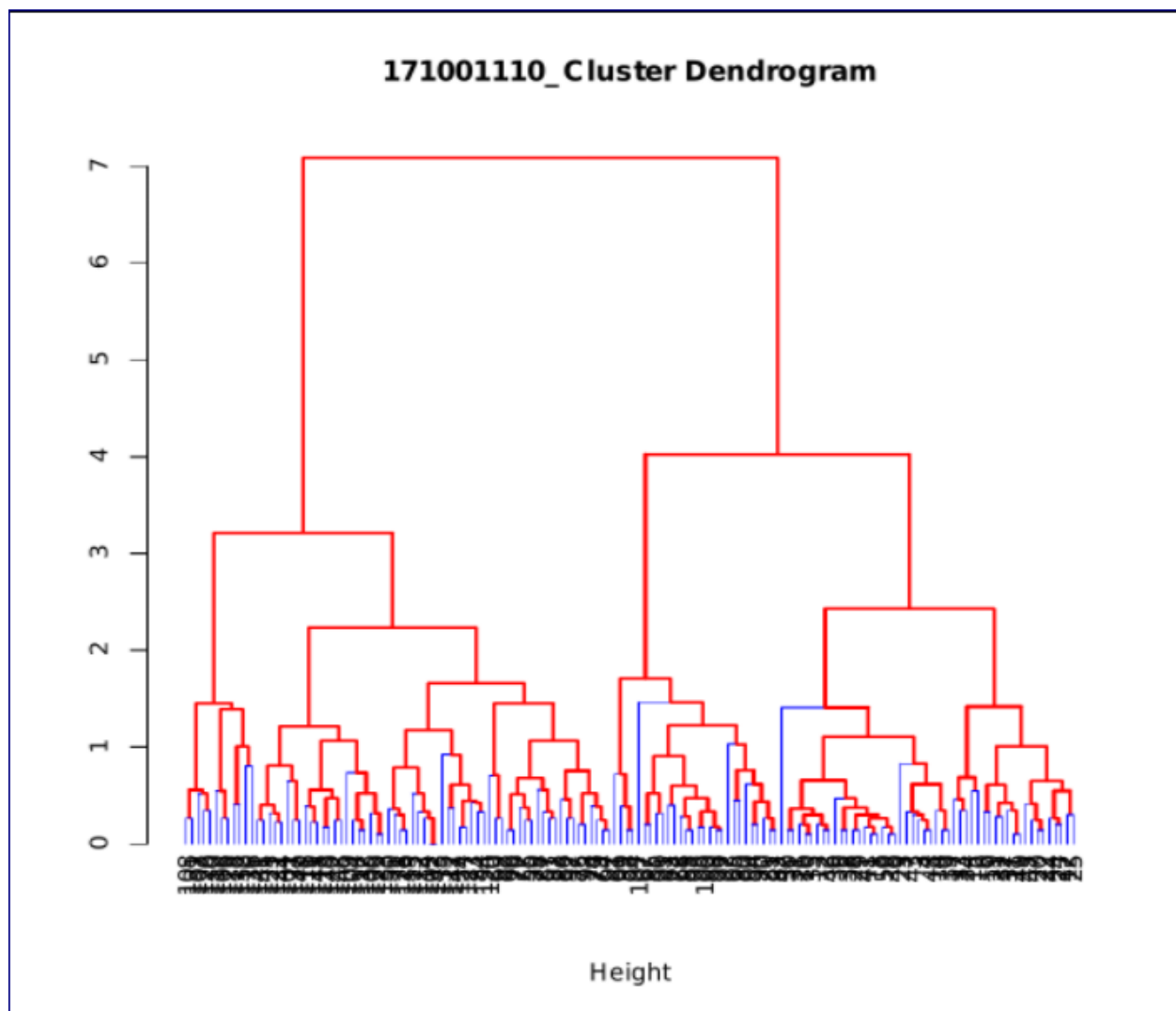
SOURCE CODE:

```
# Import dataset
data <- dist(iris[, 1:4])

# Hierarchical clustering using Complete Linkage
hc1uster <- hclust(data )
hcd<- as.dendrogram(hc1uster)

# Plot the obtained dendrogram
plot(hcd, xlab="Height", main="Cluster Dendrogram", edgePar= list (col=c("red","blue"),
lwd=2:1))
```

OUTPUT:



RESULT:

EX No :8

IMPLEMENTATION OF CLUSTERING TECHNIQUES

DATE :

K - Means

AIM:

To implement a K - Means clustering technique using python language.

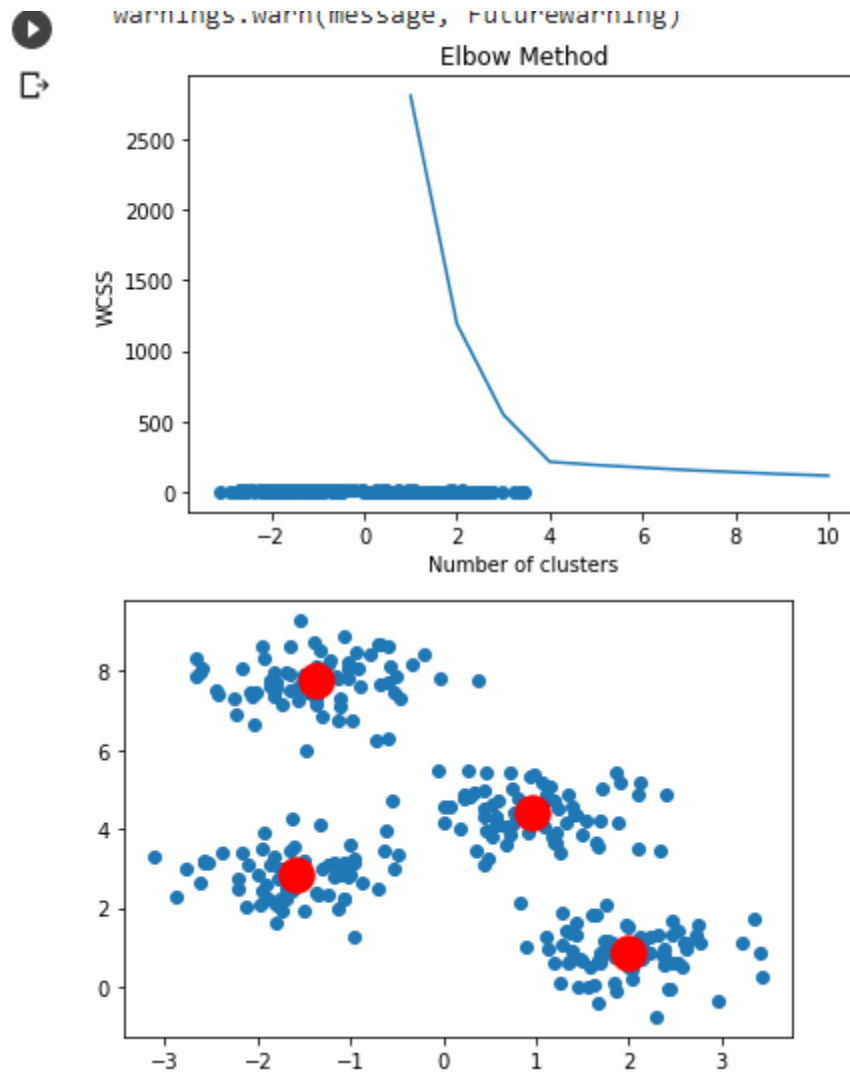
EXPLANATION:

- Import KMeans from sklearn.cluster
- Assign X and Y.
- Call the function KMeans().
- Perform scatter operation and display the output.

SOURCE CODE:

```
Import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import KMeans
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
plt.scatter(X[:,0], X[:,1])
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=300, n_init=10, random_state=0)
pred_y = kmeans.fit_predict(X)
plt.scatter(X[:,0], X[:,1])
plt.scatter(kmeans.cluster_centers_[0], kmeans.cluster_centers_[1], s=300, c='red')
plt.show()
```

OUTPUT :



RESULT:

**EX No :9 IMPLEMENT DATA VISUALIZATION TECHNIQUES AND
DATE : VISUALIZE DATA USING ANY PLOTTING FRAMEWORK**

1.LINE CHART 2.BAR CHART 3.BOX PLOT 4.SCATTER PLOT

AIM:

To implement Data visualization techniques and visualize data using any plotting framework such as 1. Line chart , 2.Bar chart, 3.Box plot , 4.Scatter plot

1. LINE CHART

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

2. BAR CHART

- A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R can draw both vertical and Horizontal bars in the bar chart.
- R uses the function barplot() to create bar charts.

3. BOX PLOT

- 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.

4. SCATTER PLOT

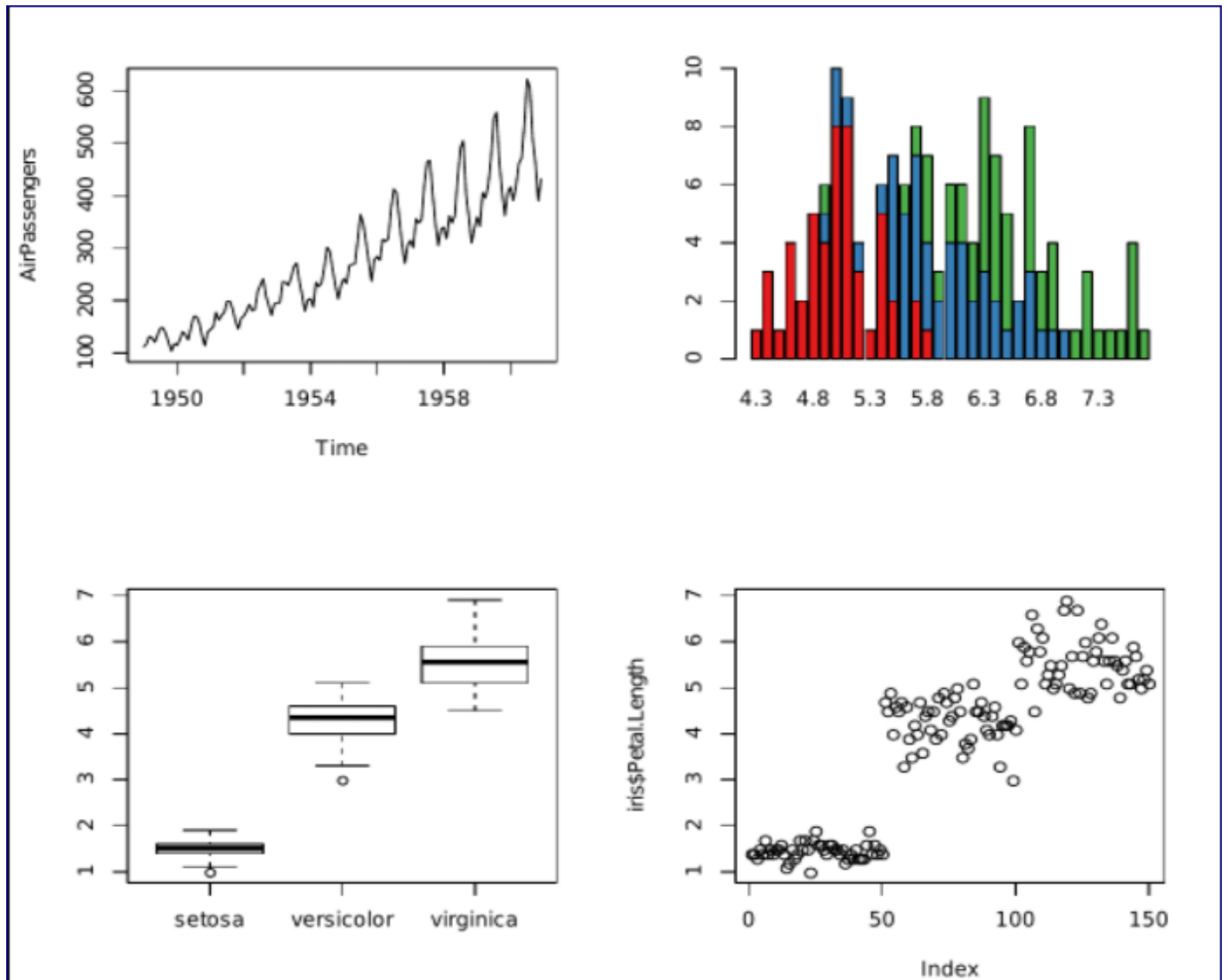
- Scatterplots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.
- The simple scatterplot is created using the plot() function.

SOURCE CODE:

```
library(RColorBrewer)
#Plot Multiple graphs in a page
par(mfrow=c(2,2))
#Line Plot
plot(AirPassengers,Type="l")
#BarPlot
barplot(table(iris$Species, iris$Sepal.Length), col=brewer.pal(3,"Set1"))
```

```
#BoxPlot
boxplot(iris$Petal.Length ~ iris$Species)
#ScatterPlot
plot(x=iris$Petal.Length)
```

OUTPUT:



RESULT:

**EX No :10 IMPLEMENTATION DATA VISUALIZATION TECHNIQUES AND
DATE: VISUALIZE DATA USING ANY PLOTTING FRAMEWORK**

1.AREA CHART 2.HEAT MAP 3.CORRELOGRAM

AIM:

To implement Data visualization techniques and visualize data using any plotting framework such as 1.Area chart 2.Heat Map 3.Correlogram

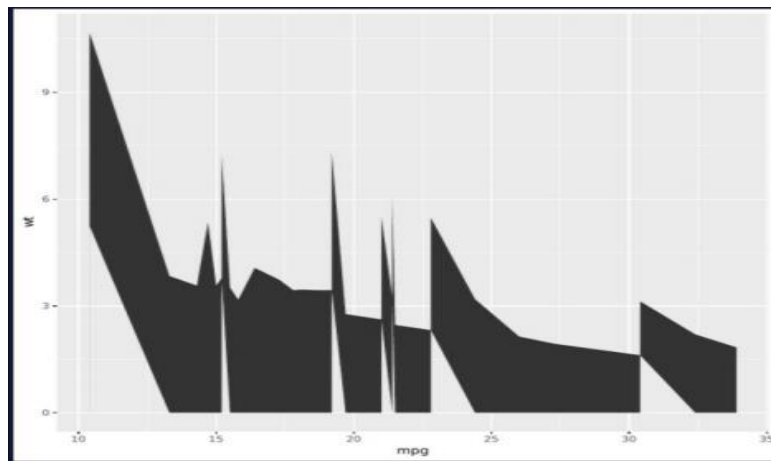
1. AREA CHART

- An area chart represents the evolution of a numeric variable. It is very close to a line chart. The stacked area charts are used to represent multiple variables against some parameters. Stacked area charts are the most used area chart types in the analysis.
- R offers the standard function `geom_area()` to plot the area charts.

SOURCE CODE:

```
data("mtcars")  
df <- mtcars  
library(ggplot2)  
ggplot(df, aes(x=mpg, y=wt)) + geom_area()
```

OUTPUT:



2. HEATMAP

- The `heatmap()` function is natively provided in R. It produces high quality matrices and offers statistical tools to normalize input data, run clustering algorithms and visualize the result with dendrograms.
- A heat map is a false color image (basically `image(t(x))`) with a dendrogram added to the left side and to the top.

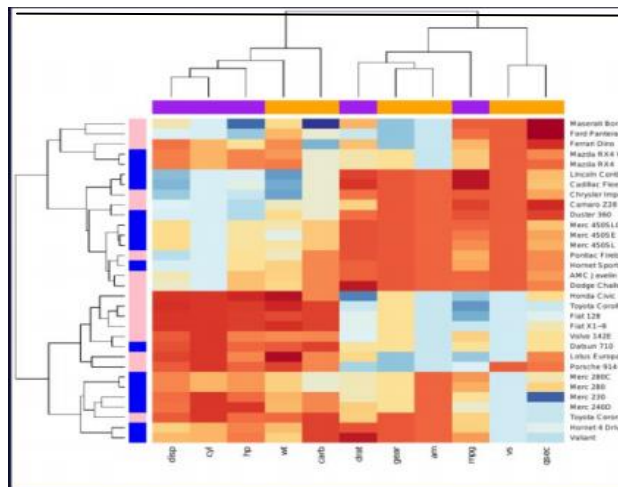
SOURCE CODE:

```
df <- scale(mtcars)  
library("RColorBrewer")
```



```
col <- colorRampPalette(brewer.pal(10, "RdYlBu"))(256)
heatmap(df, scale = "none", col = col, RowSideColors = rep(c("blue", "pink"), each = 16),
ColSideColors = c(rep("purple", 5), rep("orange", 6)))
```

OUTPUT:



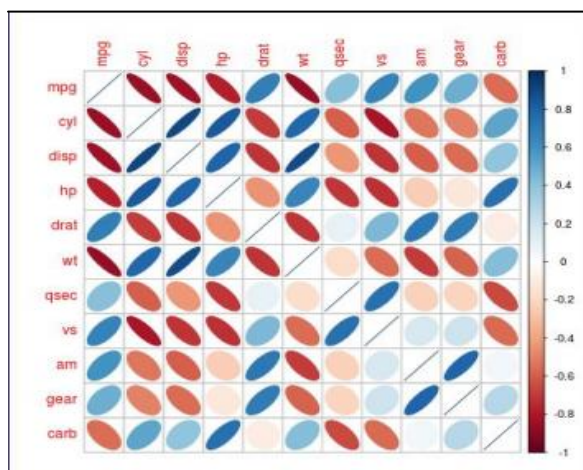
3. CORRELOGRAM:

- A graph of the correlation matrix is known as Correlogram. This is generally used to highlight the variables in a data set or data table that are correlated most.
- The correlation coefficients in the plot are colored based on the value. Based on the degree of association among the variables, we can reorder the correlation matrix accordingly.

SOURCE CODE:

```
library("corrplot")
M <- cor(mtcars)
corrplot(M, method="ellipse")
```

OUTPUT:



RESULT

EX No :11 IMPLEMENTATION OF SAMPLE SCRIPT BY USING PIG AND HIVE COMMAND

DATE:

AIM:

To implement script by using PIG and HIVE

HIVE COMMANDS

```
[root@sandbox Desktop]# vi bdanames.txt
```

```
[root@sandbox Desktop]# cat bdanames.txt
```

```
0    Rajesh
1    Babu
2    Mani
3    Kumar
4    Poonkuzhalihive
5    Ganesh
6    Raja
7    Kannan
8    Jeyalakshmi
```

```
[root@sandbox Desktop]# hive
```

```
16/06/15 12:43:33 WARN conf.HiveConf: HiveConf of name hive.optimize.mapjoin.mapreduce does not exist
```

```
.....
```

```
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
```

```
hive> dfs -ls /apps/hive/warehouse;
```

```
Found 3 items
```

```
drwxr-xr-x - hue hdfs      0 2014-12-16 19:30 /apps/hive/warehouse/sample_07
drwxr-xr-x - hue hdfs      0 2014-12-16 19:30 /apps/hive/warehouse/sample_08
drwxr-xr-x - hive hdfs     0 2014-12-16 19:50 /apps/hive/warehouse/xademo.db
```

SAMPLE CODE 1for creating/listing/moving data table to hadoop

CREATING TABLE [recnames]

```
hive> create table bdanameshive(id int,name string) ROW FORMAT DELIMITED FIELDS  
TERMINATED BY '\t';
```

```
OK
```

```
Time taken: 4.111 seconds
```

```
hive> dfs -put names.txt /apps/hive/warehouse/bdanameshive/;
```

```
hive> select *from bdanameshive;
```

```
OK
```

```
0    Rajesh
1    Babu
2    Mani
3    Kumar
```

```

4      Poonkuzhali
5      Ganesh
6      Raja
7      Kannan
8      Jeyalakshmi
Time taken: 1.093 seconds, Fetched: 9 row(s)
hive> dfs -ls /apps/hive/warehouse/bdanameshive;
Found 1 items
-rw-r--r--  1 root hdfs      93 2016-06-15 12:47 /apps/hive/warehouse/bdanameshive/bdanames.txt
hive>
hive> describe bdanameshive;
OK
id          int
name        string
Time taken: 0.753 seconds, Fetched: 2 row(s)
hive> show databases;
OK
default
raj
xademo
Time taken: 0.03 seconds, Fetched: 3 row(s)

```

FIG:

```

[root@sandbox Desktop]# hadoop fs -mkdir bdamoviereview
[root@sandbox Desktop]# hadoop fs -put bdamovies.txt bda
[root@sandbox Desktop]# hadoop fs -ls bda
Found 2 items
-rw-r--r--  1 root root      351 2016-06-15 12:11 bda/bdamovies.txt
-rw-r--r--  1 root root      60 2016-06-15 11:55 bda/bdastudent.txt
[root@sandbox Desktop]# pig
...
org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system
at: hdfs://sandbox.hortonworks.com:8020
grunt> ls
hdfs://sandbox.hortonworks.com:8020/user/root/.Trash    <dir>
hdfs://sandbox.hortonworks.com:8020/user/root/whouse    <dir>
grunt> cd bda
grunt> ls
hdfs://sandbox.hortonworks.com:8020/user/root/bda/bdamovies.txt<r 1>  351
hdfs://sandbox.hortonworks.com:8020/user/root/bda/bdastudent.txt<r 1>  60
grunt> pwd
hdfs://sandbox.hortonworks.com:8020/user/root/bda

```

Loading and Describe files:Movie Review)

```
grunt>      Movies      =      LOAD      'bdamovies.txt'      USING      PigStorage(',')      as
(id:int,name:chararray,year:int,rating:float, duration:int);
grunt> describe Movies;
Movies: {id: int,name: chararray,year: int,rating: float,duration: int}
```

Displaying the contents of Movies:

```
grunt> DUMP Movies;
```

Success!

Counters:

Total records written : 10

Total bytes written : 371

Spillable Memory Manager spill count : 0

Total bags proactively spilled: 0

Total records proactively spilled: 0

(NIGHTMARE BEFORE CHRISTMAS)

(THE MUMMY)

(ORPHANS OF THE STORM)

(THE OBJECT OF BEAUTY)

(NIGHT TIDE)

(ONE MAGIC CHRISTMAS)

(MURIEL'S WEDDING)

(MOTHER'S BOYS)

(NOSFERATU: ORIGINAL VERSION)

(NICK OF TIME)

GROUP Movies By Ratings:

```
grunt> GroupMovies = GROUP Movies BY rating;
```

2016-06-15 04:02:38,449 [main] WARN org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 1 time(s).

```
grunt> DUMP GroupMovies;
```

(2.8,{(5,Night Tide,1963,2.8,5126),(4,The Object of Beauty,1991,2.8,6150)})

(3.2,{(3,Orphans of the Storm,1921,3.2,9062)})

(3.4,{(10,Nick of Time,1995,3.4,5333),(8,Mother's Boys,1994,3.4,5733)})

(3.5,{(9,Nosferatu: Original Version,1929,3.5,5651),(7,Muriel's Wedding,1994,3.5,6323),(2,The Mummy,1932,3.5,4388)})

(3.8,{(6,One Magic Christmas,1985,3.8,5333)})

(3.9,{(1,Nightmare Before Christmas,1993,3.9,4568)})

RESULT:

EX No :12 IMPLEMENT AN APPLICATION THAT STORES DATA IN MONGODB
DATE:

AIM:

To implement an that stores big data in Hbase / MongoDB / Pig using Hadoop / R.

Procedure:

MongoDB with R

1. To use MongoDB with R, first, we have to download and install MongoDB Next, start MongoDB.
2. Add the MongoDB location C:\Program Files\MongoDB\Server\5.0\bin to the path in environmental variables.
3. Go to C, create a new folder named data and create a subfolder inside it named db.
4. Verify mongodb installation by running mongod on command prompt
5. Go to R, then install the packages given in the observation using `install.packages("the package name")` command
6. Install the package "tidyverse" using `install.packages` command
7. Then, import the packages using `library()` command.
8. Go to <https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2> and click on export and download the chicago crimes list as csv file.
9. Install the data.table packages using `install.packages("data.table")` command
10. Execute the command : `crimes = data.table::fread("Crimes_-_2001_to_Present.csv")`
11. Once done, execute all the commands given in the observation

`library(tidyverse)`

```
> library(tidyverse)
-- Attaching packages ----- tidyverse 1.3.1 --
v tibble 3.1.4      v purrr 0.3.4
v tidyr 1.1.3      v stringr 1.4.0
v readr 2.0.1      v forcats 0.5.1
-- Conflicts ----- tidyverse_conflicts() --
x lubridate::as.difftime() masks base::as.difftime()
x gridExtra::combine()    masks dplyr::combine()
x lubridate::date()        masks base::date()
x dplyr::filter()          masks stats::filter()
x lubridate::intersect()   masks base::intersect()
x dplyr::lag()             masks stats::lag()
x purrr::map()             masks maps::map()
x lubridate::setdiff()     masks base::setdiff()
```

STEP 1:Let's insert the crimes data from data.gov to MongoDB. The dataset reflects reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago since 2001.

```
library (ggplot2)
library (dplyr)
library (maps)
library (ggmap)
library (mongolite)
library (lubridate)
library (gridExtra)
```



```
R Console

> library(ggplot2)
> library (dplyr)

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

  filter, lag

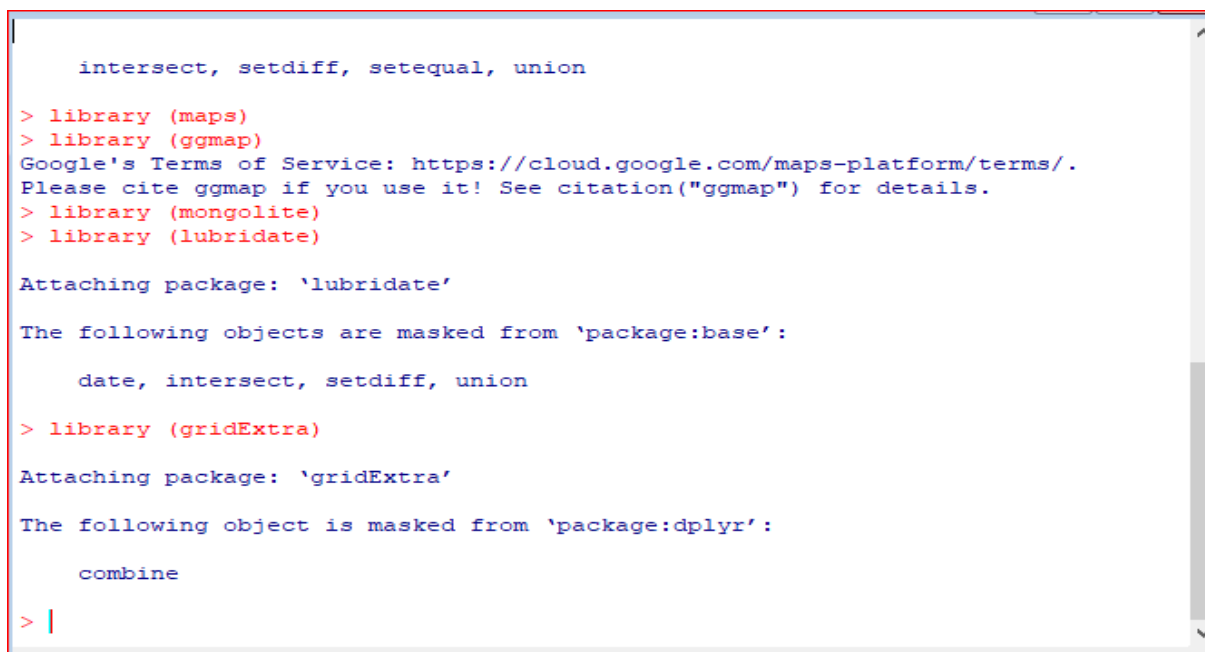
The following objects are masked from 'package:base':

  intersect, setdiff, setequal, union

> library (maps)
> library (ggmap)
Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
Please cite ggmap if you use it! See citation("ggmap") for details.
> library (mongolite)
> library (lubridate)

Attaching package: 'lubridate'

The following objects are masked from 'package:base':
```



```
  intersect, setdiff, setequal, union

> library (maps)
> library (ggmap)
Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
Please cite ggmap if you use it! See citation("ggmap") for details.
> library (mongolite)
> library (lubridate)

Attaching package: 'lubridate'

The following objects are masked from 'package:base':

  date, intersect, setdiff, union

> library (gridExtra)

Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

  combine

> |
```

```
crimes=data.table::fread("Crimes_-_2001_to_Present.csv")
```

```
> crimes=data.table::fread("Crimes_-_2001_to_Present.csv")
|-----|
|=====|
> |
```

names (crimes)

```
> names (crimes)
[1] "ID"           "Case Number"    "Date"
[4] "Block"        "IUCR"           "Primary Type"
[7] "Description"   "Location Description" "Arrest"
[10] "Domestic"     "Beat"           "District"
[13] "Ward"         "Community Area" "FBI Code"
[16] "X Coordinate" "Y Coordinate"   "Year"
[19] "Updated On"   "Latitude"       "Longitude"
[22] "Location"
> |
```

STEP 2: Let's use the insert function from the mongolite package to insert rows to a collection in MongoDB. Let's create a database called Chicago and call the collection crimes.

```
my_collection = mongo(collection = "crimes", db = "Chicago")
my_collection$insert(crimes)
```

```
> my_collection = mongo(collection = "crimes", db = "Chicago")
> my_collection$insert(crimes)
List of 5
 $ nInserted  : num 7405504
 $ nMatched   : num 0
 $ nRemoved   : num 0
 $ nUpserted  : num 0
 $ writeErrors: list()
> |
```

STEP 3: Let's check if we have inserted the "crimes" data
my_collection\$count()

```
> my_collection$count()
[1] 29637460
> |
```

STEP 4: First, let's look what the data looks like by displaying one record:
my_collection\$iterate()\$one()

```
> my_collection$iterate()$one()
$ID
[1] 10224738

$CaseNumber
[1] "HY411648"

$Date
[1] "09/05/2015 01:30:00 PM"

$Block
[1] "043XX S WOOD ST"

$IUCR
[1] "0486"

$PrimaryType
[1] "BATTERY"

$Description
[1] "DOMESTIC BATTERY SIMPLE"

$LocationDescription
[1] "RESIDENCE"

$Arrest
[1] FALSE

$Domestic
[1] TRUE

$Beat
[1] 924
|
```

```
$District
[1] 9

$Ward
[1] 12

$CommunityArea
[1] 61

$FBICode
[1] "08B"

$XCoordinate
[1] 1165074

$YCoordinate
[1] 1875917

$Year
[1] 2015

$UpdatedOn
[1] "02/10/2018 03:50:01 PM"

$Latitude
[1] 41.81512

$Longitude
[1] -87.67

$Location
[1] "(41.815117282, -87.669999562)"

> |
```


STEP 5: How many distinct “Primary Type” do we have?

```
length(my_collection$distinct("PrimaryType"))
```

```
> length(my_collection$distinct("PrimaryType"))
[1] 36
> |
```

STEP 6: Now, let's see how many domestic assaults there are in the collection

```
my_collection$count({'PrimaryType':"ASSAULT", "Domestic" : "true" })
```

```
> my_collection$count({'PrimaryType':"ASSAULT", "Domestic" : "true" })
[1] 0
> |
```

STEP 7: To get the filtered data and we can also retrieve only the columns of interest.

```
query1= my_collection$find({'PrimaryType' : "ASSAULT", "Domestic" : "true" })
```

```
query2= my_collection$find({'PrimaryType' : "ASSAULT", "Domestic" : "true" },
```

```
fields = {'_id':0, "PrimaryType":1, "Domestic":1})
```

```
ncol(query1)
```

```
ncol(query2)
```

```
> my_collection$count({'PrimaryType':"ASSAULT", "Domestic" : "true" })
[1] 0
> query1= my_collection$find({'PrimaryType' : "ASSAULT", "Domestic" :
+ "true" })
> query2= my_collection$find({'PrimaryType' : "ASSAULT", "Domestic" :
+ "true" },
+ fields = {'_id':0, "PrimaryType":1, "Domestic":1})
> ncol(query1)
[1] 0
> ncol(query2)
[1] 0
```

STEP 8: To find out “Where do most crimes take place?” use the following command

```
my_collection$aggregate(['{$group':{'_id':"$LocationDescription", "Count":
```

```
{ "$sum":1 } }'])%>%na.omit()%>%
```

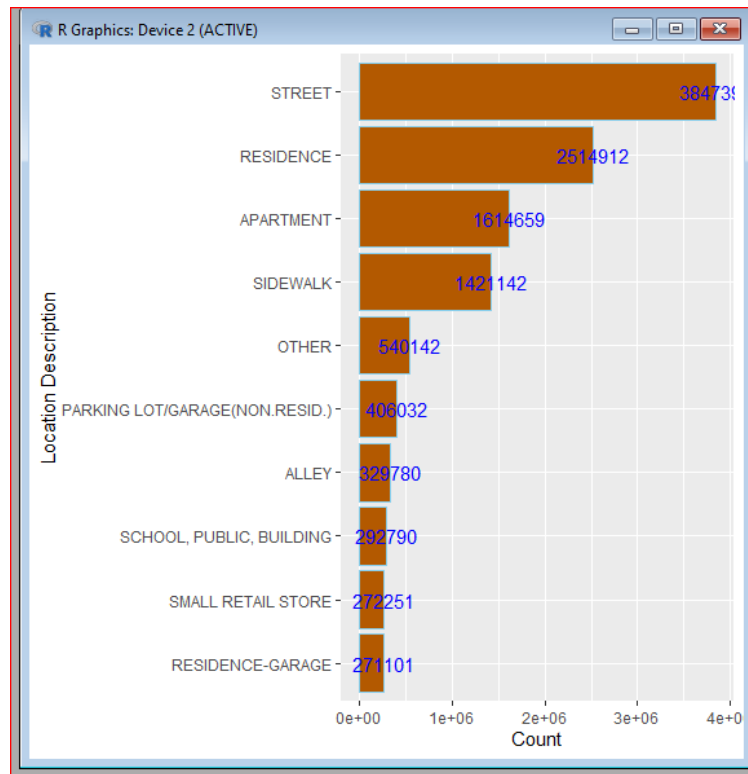
```
arrange(desc(Count))%>%head(10)%>%
```

```
ggplot(aes(x=reorder(`_id`,Count),y=Count))+
```

```
geom_bar(stat="identity",color="skyblue",fill="#b35900")+geom_text(aes(label =
```

```
Count), color = "blue") +coord_flip()+xlab("Location Description")
```

```
> my_collection$aggregate(['{$group':{'_id':"$LocationDescription", "Count":
+ {"$sum":1 } }'])%>%na.omit()%>%
+ arrange(desc(Count))%>%head(10)%>%
+ ggplot(aes(x=reorder(`_id`,Count),y=Count))+
+ geom_bar(stat="identity",color="skyblue",fill="#b35900")+geom_text(aes(label $
+ Count), color = "blue") +coord_flip()+xlab("Location Description")
> |
```



RESULT: