SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

PRACTICAL RECORD NOTE

STUDENT NAME :

REGISTER NUMBER : RA_____

CLASS : MCA Section: F

YEAR & SEMESTER : I YEAR & II SEMESTER

SUBJECT CODE : PCA20S02J

SUBJECT TITLE : DATA ANALYSIS USING R

APRIL 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

SRM Nagar, Kattankulathur – 603 203

CERTIFICATE

Certified to be the bonafide record of practical work done by _____

Register No. _____ *of MCA Degree course for* PCA20S02J – DATA ANALYSIS USING R *in the Computer lab in SRM Institute of Science and Technology during the academic year 2023-2024.*

Staff In-charge *Head of the Department*

Submitted for Semester Practical Examination held on _____.

Internal Examiner *External Examiner*

INDEX

Ex. No:1	Installation of R and R-Studio
Date: 13.12.2023	

INSTALLATION OF R

AIM

To install R and R-Studio

PROCEDURE

Step 1: Go to CRAN R project website - https://cran.r-project.org/

Step 2: Click on the Download R for Windows link.

Step 3: Click on the base subdirectory link or install R for the first time link.

Step 4: Click Download R 4.2.1 for Windows and save the executable .exe file.

Step 5: Run the .exe file and follow the installation instructions.

Step 6: Select the desired language and then click Next.

Step 7: Read the license agreement and click Next.

Step 8: Select the components you wish to install. Click Next.

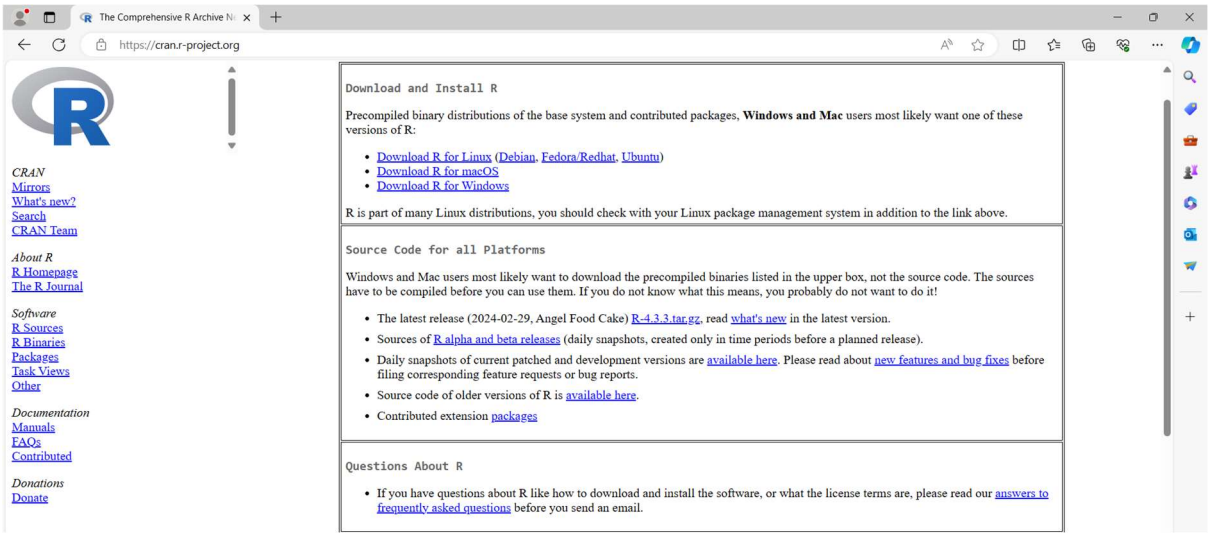
Step 9: Browse the folder/path you wish to install R into and then confirm by clicking Next.

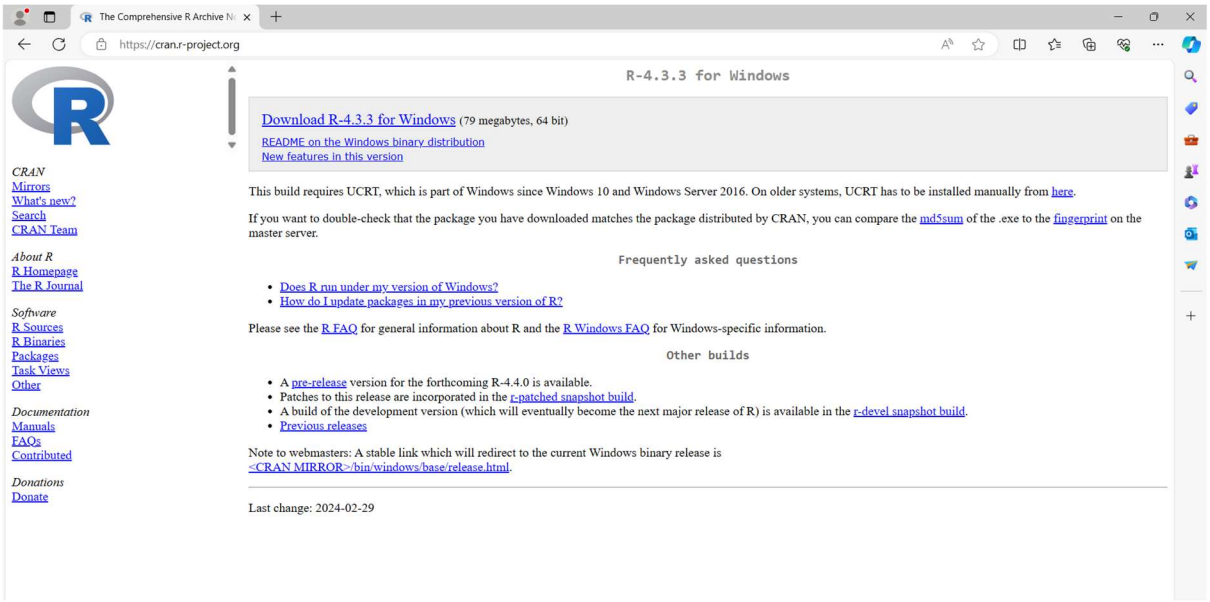
Step 10: Select additional tasks like creating desktop shortcuts etc. then click Next.

Step 11: Wait for the installation process to complete.

Step 12: Click on Finish to complete the installation.

OUTPUT


The Comprehensive R Archive Ne
https://cran.r-project.org
CRAN
Mirrors
What's new?
Search
CRAN Team
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Task Views
Other
Documentation
Manuals
FAQs
Contributed
Donations
Donate
Download and Install R
Precompiled binary distributions of the base system and contributed packages, Windows and Mac users most likely want one of these versions of R:
Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
Download R for macOS
Download R for Windows
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.
Source Code for all Platforms
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!
The latest release (2024-02-29, Angel Food Cake) R-4.3.3.tar.gz, read what's new in the latest version.
Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
Source code of older versions of R is available here.
Contributed extension packages
Questions About R
If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.


The Comprehensive R Archive Ne
https://cran.r-project.org
CRAN
Mirrors
What's new?
Search
CRAN Team
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Task Views
Other
Documentation
Manuals
FAQs
Contributed
Donations
Donate
R-4.3.3 for Windows
Download R-4.3.3 for Windows (79 megabytes, 64 bit)
README on the Windows binary distribution
New features in this version
This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from here.
If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server.
Frequently asked questions
Does R run under my version of Windows?
How do I update packages in my previous version of R?
Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.
Other builds
A pre-release version for the forthcoming R-4.4.0 is available.
Patches to this release are incorporated in the r-patched snapshot build.
A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
Previous releases
Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.html.
Last change: 2024-02-29

INSTALLATION OF R-STUDIO

PROCEDURE

Step 1: With R-base installed, let's move on to installing R-Studio. To begin, go to download RStudio - https://posit.co/downloads and click on the download button for R-Studio desktop.

Step 2: Click on the link for the windows version of R-Studio and save the .exe file.

Step 3: Run the .exe and follow the installation instructions.

Step 4: Click Next on the welcome window.

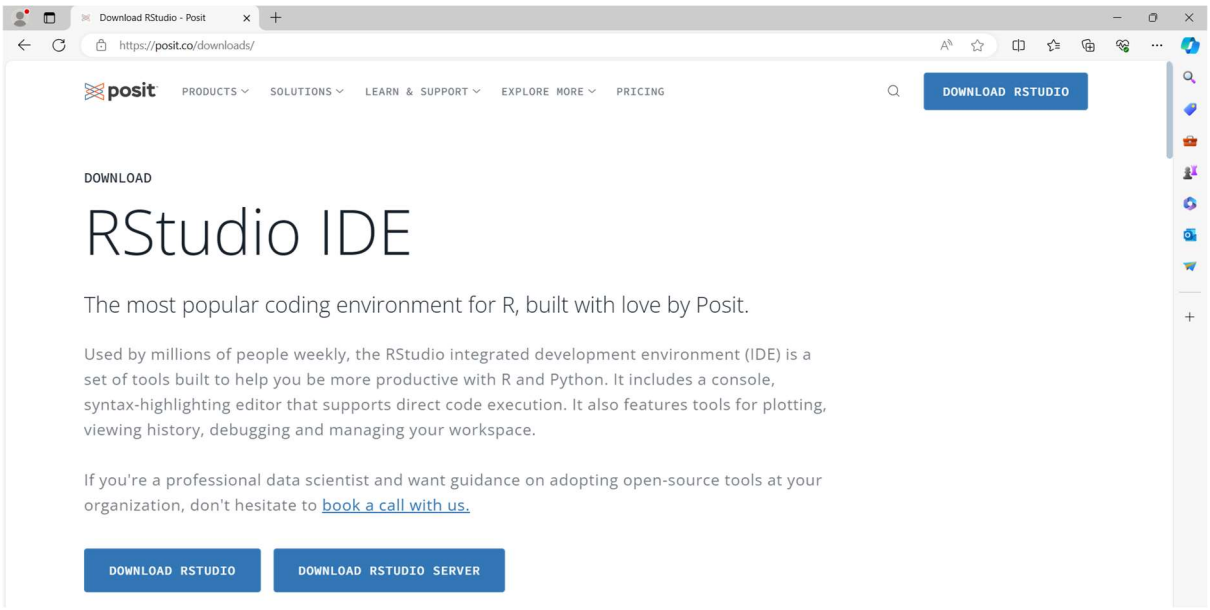
Step 5: Enter/browse the path to the installation folder and click Next to proceed.

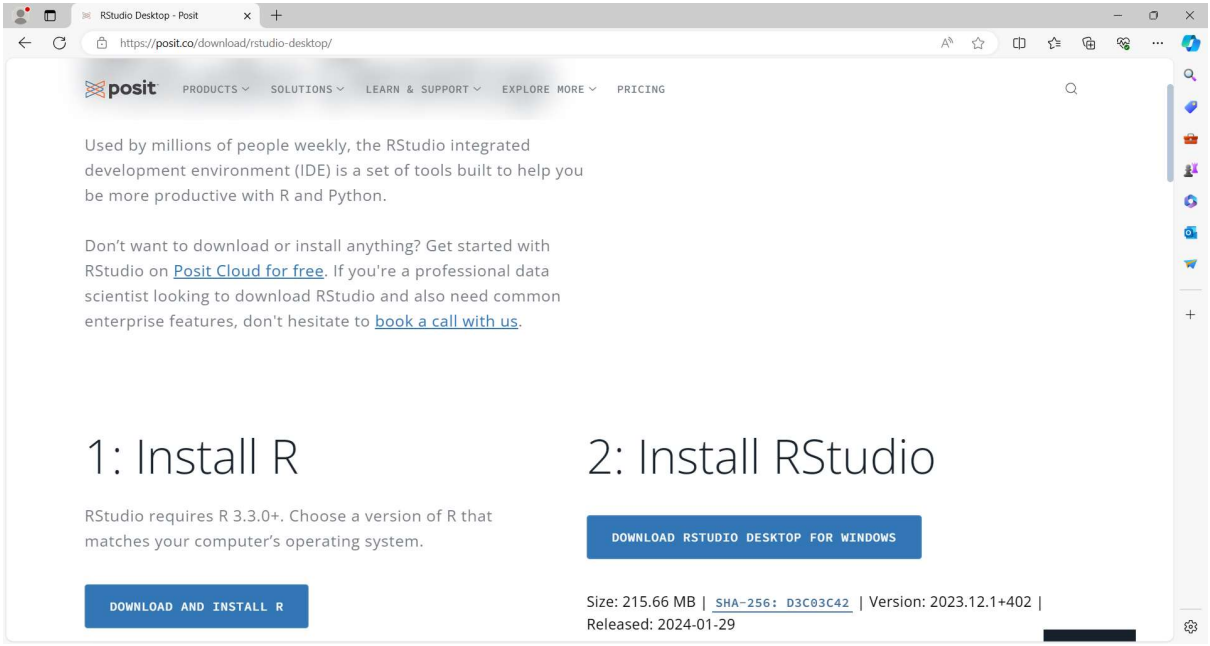
Step 6: Select the folder for the start menu shortcut and then click Next.

Step 7: Wait for the installation process to complete.

Step 8: Click on Finish to complete the installation.

OUTPUT


Download RStudio - Posit
https://posit.co/downloads/
posit
PRODUCTS
SOLUTIONS
LEARN & SUPPORT
EXPLORE MORE
PRICING
DOWNLOAD RSTUDIO
DOWNLOAD
RStudio IDE
The most popular coding environment for R, built with love by Posit.
Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution. It also features tools for plotting, viewing history, debugging and managing your workspace.
If you're a professional data scientist and want guidance on adopting open-source tools at your organization, don't hesitate to book a call with us.
DOWNLOAD RSTUDIO
DOWNLOAD RSTUDIO SERVER


RStudio Desktop - Posit
https://posit.co/download/rstudio-desktop/
posit
PRODUCTS
SOLUTIONS
LEARN & SUPPORT
EXPLORE MORE
PRICING
Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python.
Don't want to download or install anything? Get started with RStudio on Posit Cloud for free. If you're a professional data scientist looking to download RStudio and also need common enterprise features, don't hesitate to book a call with us.
1: Install R
RStudio requires R 3.3.0+. Choose a version of R that matches your computer's operating system.
DOWNLOAD AND INSTALL R
2: Install RStudio
DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS
Size: 215.66 MB | SHA-256: D3C03C42 | Version: 2023.12.1+402 |
Released: 2024-01-29

Ex. No:2	Working with R Packages
Date: 22.12.2023	

AIM

To install, load and work with packages

PACKAGES

A package is basically just a big collection of functions, data sets and other R objects that are all grouped together under a common name. A package must be installed before it can be loaded. A package must be loaded before it can be used.

Install R Packages from CRAN

install.packages ( ) - This function is used to install a required package in the R programming language.

Example : To install ggplot2 package

```
install.packages("ggplot2")
```

Uninstall R Packages

remove.packages ( ) - This function is used to uninstall apackage in the Rprogramming language.

Example : To uninstall ggplot2 package

```
remove.packages("ggplot2")
```

Loading of R Packages

library ( ) - It is used to load and list all the packages in the R Programminglanguage.

Example : To load ggplot2 package

```
library ( ggplot2 )
```

To list all the packages installed

```
library ( )
```

Updating R Packages

old.packages ( ) - It is used to check which packages need an update in R.

Example : To check an update

```
old.packages( )
```

update.packages ( ) - It is used to update all the packages in R

Example : To update Packages

```
update.packages( )
```

Listing the Packages that are Installed

install.packages ( ) - It is used to list out all packages which are installed in computer

Example : To list out installed packages

```
installed.packages( )
```

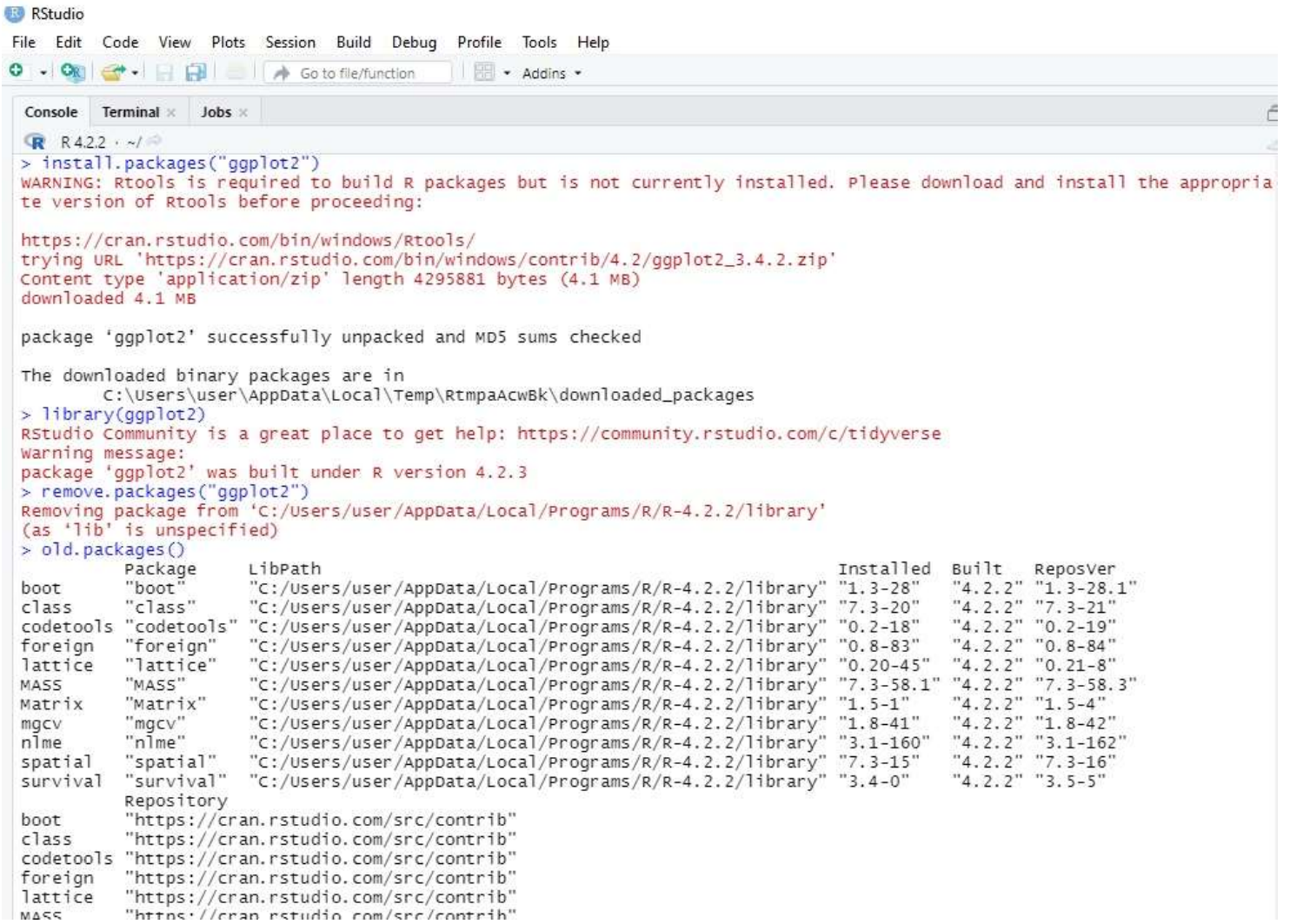
Get help pages about Packages

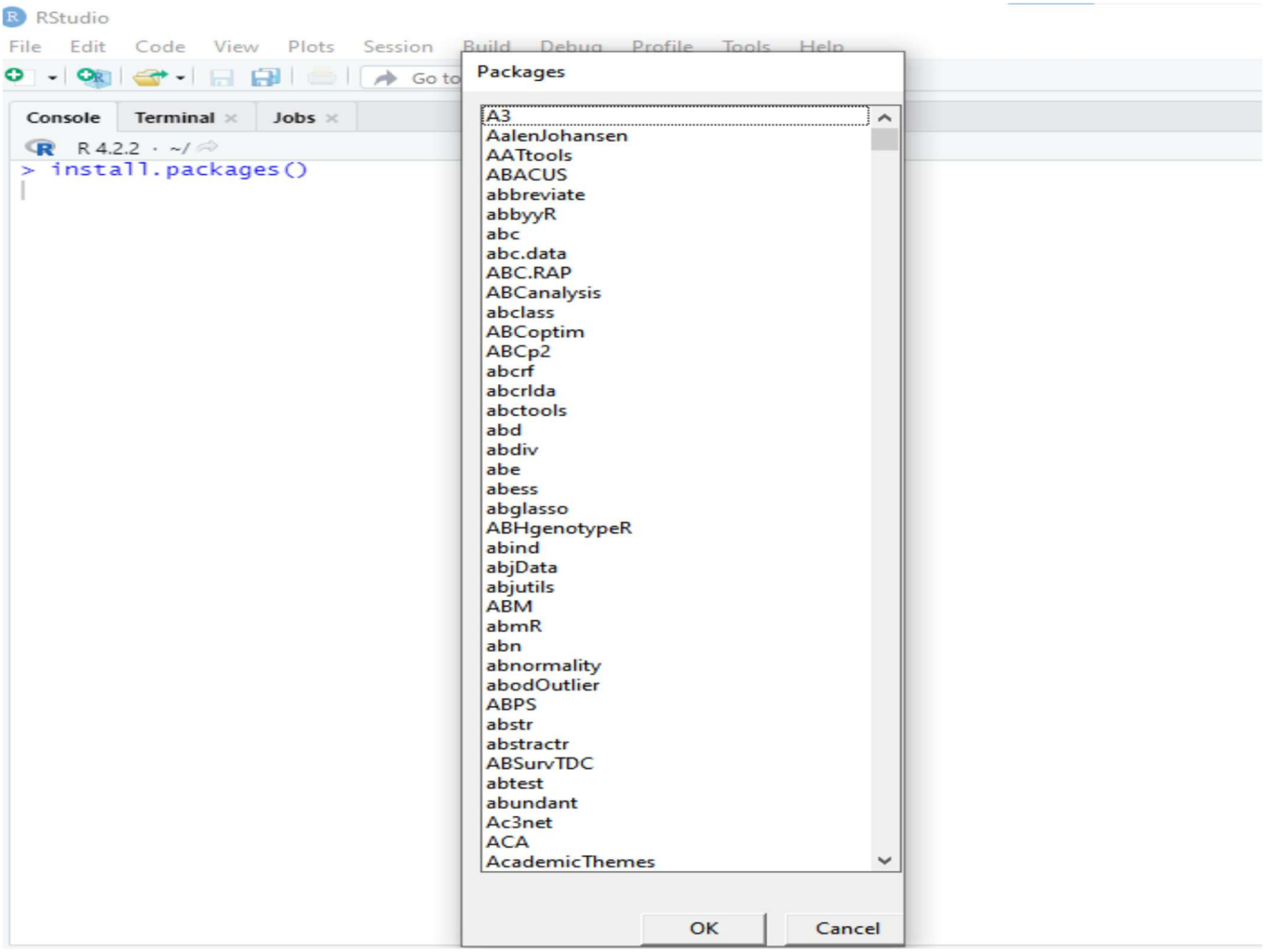
Help ( ) and (?)- They provide access to the documentation pages for R functions, data sets, and other objects, both for packages in the standard R distribution and for contributed packages.

Example : To get more description about ggplot2

```
help("ggplot2") and ?ggplot2
```

OUTPUT


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Console
Terminal
Jobs
R 4.2.2 · ~/
> install.packages("ggplot2")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropria
te version of Rtools before proceeding:
https://cran.rstudio.com/bin/windows/Rtools/
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/ggplot2_3.4.2.zip'
Content type 'application/zip' length 4295881 bytes (4.1 MB)
downloaded 4.1 MB
package 'ggplot2' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\user\AppData\Local\Temp\RtmpaAcwBk\downloaded_packages
> library(ggplot2)
RStudio Community is a great place to get help: https://community.rstudio.com/c/tidyverse
Warning message:
package 'ggplot2' was built under R version 4.2.3
> remove.packages("ggplot2")
Removing package from 'C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library'
(as 'lib' is unspecified)
> old.packages()
Package LibPath Installed Built Reposver
boot "boot" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "1.3-28" "4.2.2" "1.3-28.1"
class "class" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "7.3-20" "4.2.2" "7.3-21"
codetools "codetools" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "0.2-18" "4.2.2" "0.2-19"
foreign "foreign" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "0.8-83" "4.2.2" "0.8-84"
lattice "lattice" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "0.20-45" "4.2.2" "0.21-8"
MASS "MASS" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "7.3-58.1" "4.2.2" "7.3-58.3"
Matrix "Matrix" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "1.5-1" "4.2.2" "1.5-4"
mgcv "mgcv" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "1.8-41" "4.2.2" "1.8-42"
nlme "nlme" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "3.1-160" "4.2.2" "3.1-162"
spatial "spatial" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "7.3-15" "4.2.2" "7.3-16"
survival "survival" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "3.4-0" "4.2.2" "3.5-5"
Repository
boot "https://cran.rstudio.com/src/contrib"
class "https://cran.rstudio.com/src/contrib"
codetools "https://cran.rstudio.com/src/contrib"
foreign "https://cran.rstudio.com/src/contrib"
lattice "https://cran.rstudio.com/src/contrib"


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal Jobs
R 4.2.2 · ~/
> install.packages()
Packages
A3
AalenJohansen
AATtools
ABACUS
abbreviate
abbyyR
abc
abc.data
ABC.RAP
ABCanalysis
abclass
ABCoptim
ABCp2
abcrf
abcrlda
abctools
abd
abdiv
abe
abess
abglasso
ABHgenotypeR
abind
abjData
abjutils
ABM
abmR
abn
abnormality
abodOutlier
ABPS
abstr
abstractr
ABSurvTDC
abtest
abundant
Ac3net
ACA
AcademicThemes
OK
Cancel

Ex. No:3	Built-in Functions in R
Date:04.01.2024	

AIM

To show the working of built-in functions.

BUILT - IN FUNCTIONS

Functions	Description
sum()	Returns the sum of all the input vector.
prod()	Returns the multiplication result of all the input vector.
max()	Returns the maximum value of input vector.
min()	Returns the minimum value of input vector.
unique()	Returns the unique value.
sort()	Returns sorted value from input vector.
rev()	Returns reversed order of input vector.
rbind()	Combines vector and matrix row-wise.
cbind()	Combines vector and matrix column-wise.
setdiff()	Returns differences between two vectors.
cumsum()	Returns the cumulative sum of two vector.
abs()	Returns the absolute value of input vector.
sqrt()	Returns the square root of input vector.
ceiling()	Returns the smallest integer which is larger than or equal to input vector.
floor()	Returns the largest integer, which is smaller than or equal to input vector.
trunc()	Returns the truncate value of input vector.
round()	Returns round value of input vector.
cos(), sin(), tan()	Returns cos(), sin(), tan() value of input vector.
log()	Returns natural logarithm of input vector.
log10()	Returns common logarithm of input vector.
exp()	Returns exponent of input vector.

1. Create a Vector ‘v’ with the values 1,5,8,10,4,5,3,9,8,10,12 and do the following

a. Find Sum, Mean and Product of the Vector

Solution

```
v =c(1,5,8,10,4,5,3,9,8,10,12)
sum(v)
mean(v)
prod(v)
```

b. Find the minimum and the maximum of the Vector.

Solution

```
min(v)
max(v)
```

c. Sort the Vector in ascending and descending order.

Solution

```
sort(v)
sort(v,decreasing=TRUE)
```

d. List the distinct values in the vector

Solution

```
unique(v)
```

e. Reverse the order of the vector.

Solution

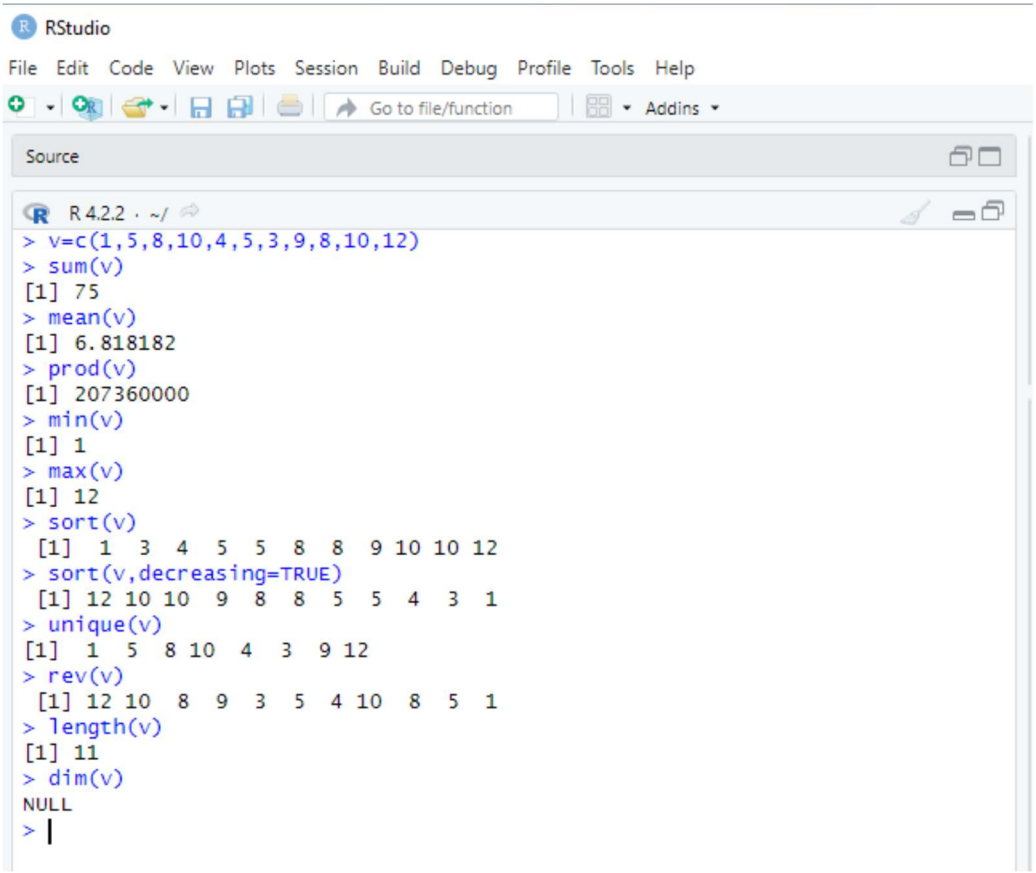
```
rev(v)
```

f. Find the length and the dimension of the vector.

Solution

```
length(v)
dim(v)
```

OUTPUT



2. Create two vectors as below:

A = 0,2,4,15

B = 3,12,4,11

a. Combines these two vectors by column wise and row wise.

Solution

A = c(0,2,4,15)

B = c(3,12,4,11)

rbind(A,B)

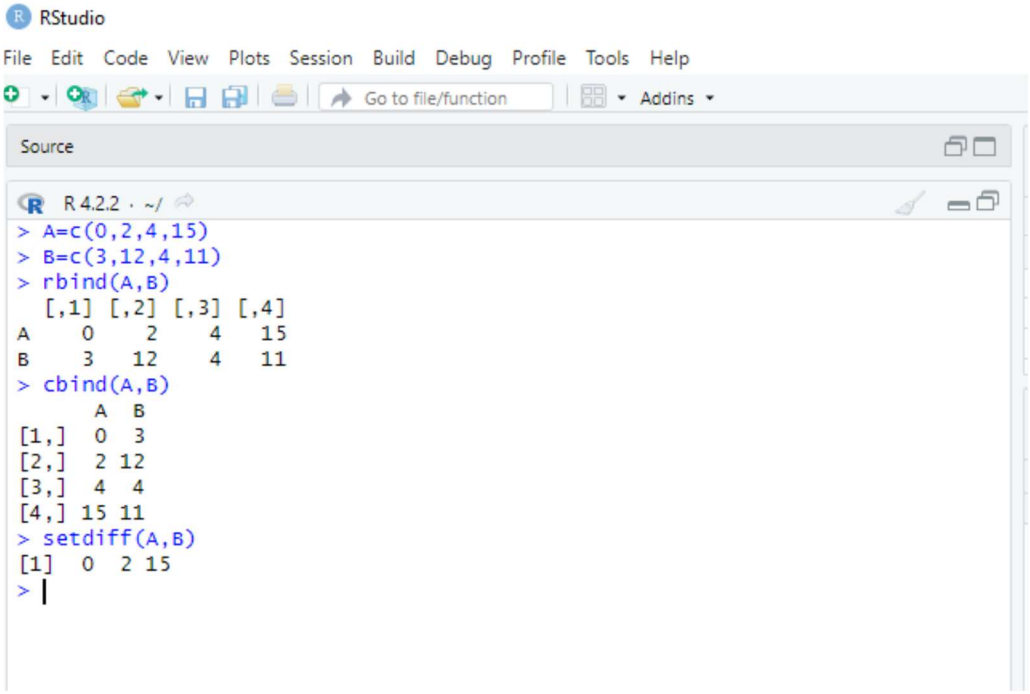
Cbind(A,B)

b. Find the elements of 'A' vector that are not in 'B' vector.

Solution

Setdiff(A,B)

OUTPUT



3. Your cell phone bill varies from month to month.

46 33 39 37 46 30 48 32 49 35 30 48

a. Represent the above information in a vector 'bill'

Solution

Bill = c(46,33,39,37,46,30,48,32,49,35,30,48)

b. Find the total amount you spent this year on the cell phone

Solution

sum(Bill)

c. Display the smallest amount you spent in a month

Solution

min(Bill)

d. Display the largest amount you spent in a month

Solution

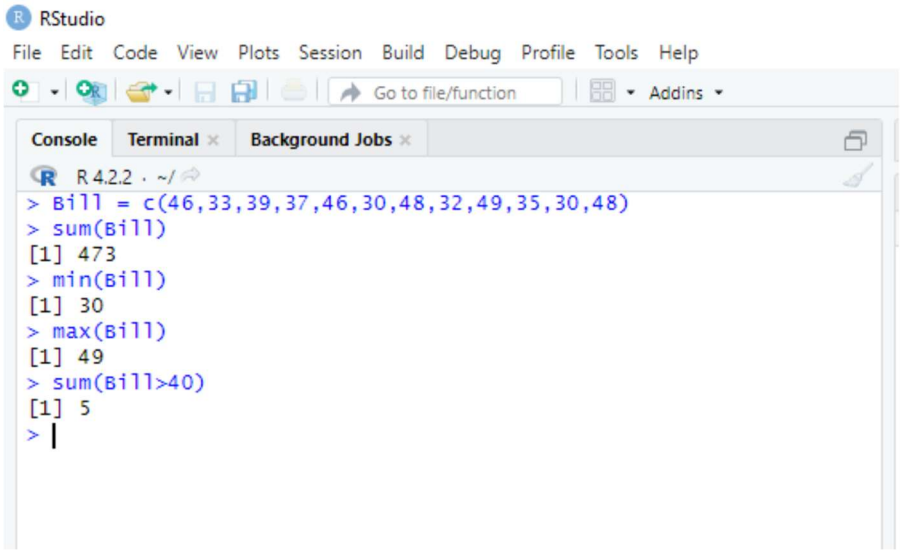
max(Bill)

e. How many months was the amount greater than $40

Solution

sum(Bill>40)

OUTPUT



4. A survey asks people if they smoke or not. The data is yes, no, no, yes, yes

a. Represent the above information in a vector

Solution

S=c(“yes”,”no”,”no”,”yes”,”yes”)

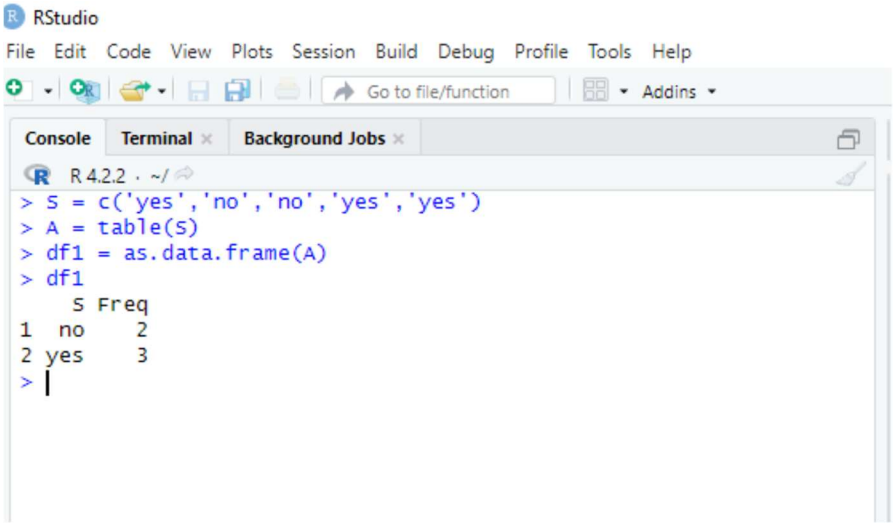
b. Display frequency table for above information

Solution

A =table(S)

df1 = as.data.frame(A)

OUTPUT


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Console
Terminal
Background Jobs
R 4.2.2 · ~/
> S = c('yes','no','no','yes','yes')
> A = table(S)
> df1 = as.data.frame(A)
> df1
S Freq
1 no 2
2 yes 3
>

Ex. No:4	Control Statements
Date: 08.01.2024	

1. Finding biggest among three numbers using if else

AIM

Find and print the greatest among three numbers.

PROCEDURE

Step 1: Take three numbers x, y, and z as input from the user.

Step 2: Compare if x is greater than y and x is greater than z.

Step 3: Print the greatest number is x.

Step 4: Else if y is greater than z, print the greatest number is y.

Step 5: If neither condition holds, then print the greatest number is z.

Step 6: End of the program.

CODE

```
x<-as.integer(readline(“Enter first number:”))
Enter first number:10
y<-as.integer(readline(“Enter second number:”))
Enter second number:20
z<-as.integer(readline(“Enter third number:”))
Enter third number:30
if (x > y && x > z) {
    print(paste("The greatest number is:", x))
 } else if (y > z) {
    print(paste("The greatest number is:", y))
 } else {
    print(paste("The greatest number is:", z))
 }
```

OUTPUT

```
[1] "The greatest number is: 30"
```

SWITCH CASE

AIM

Determine the season corresponding to the input month number and print it.

PROCEDURE

Step 1: Take an integer input representing the month number from the user.

Step 2: Use the switch function to match the input month number with corresponding seasons:

i. For months 11, 12, 1, assign "Winter" to the variable season.
ii. For months 2, 3 assign "Spring" to the variable season.
iii. For months 4, 5, 6, 7 assign "Summer" to the variable season.
iv. For months 8, 9, 10 assign "Rainy" to the variable season.

Step 3: Print the determined season corresponding to the input month number using print season.

Step 4: The program execution ends after printing the determined season.

CODE

```
month <- as.integer(readline(“Enter month number:”))
Enter month number:5
season <- switch(month,
"Winter","Spring","Spring","Summer","Summer"," Summer ","Summer",
"Rainy","Rainy","Rainy","Winter","Winter")
print(season)
```

OUTPUT

```
[1] "Summer"
```

Ex. No:5	Looping Structures
Date: 18.01.2024	

1. Find Sum of n Numbers using for loop

AIM

Calculate the sum of natural numbers up to a given integer input by the user and print the result.

PROCEDURE

Step 1: Take an integer input n from the user representing the upper limit for the sum of natural numbers.

Step 2: Initialize a variable sum to 0 to store the cumulative sum of natural numbers.

Step 3: Use for loop to iterate through natural numbers from 1 to n.

Step 4: Add each number i to the variable sum.

Step 5: Print the final sum of natural numbers using print sum.

Step 6: The program execution ends after printing the sum of natural numbers up to the given integer n.

CODE

```
n <- as.integer(readline("Enter a number:")
Enter a number:5
sum <- 0
for(i in 1:n){
 sum=sum+i
}
print(sum)
```

OUTPUT

```
[1] 15
```

2. Find Sum of n Numbers using while loop

AIM

Calculate the sum of natural numbers up to a given integer input by the user using a while loop, and print the result.

PROCEDURE

Step 1: Take an integer input n from the user representing the upper limit for the sum of natural numbers.

Step 2: Initialize a variable sum to 0 to store the cumulative sum of natural numbers.

Step 3: Initialize a counter i to 1.

Step 4: Use while loop to iterate, while i is less than or equal to n, add the current value of i to the variable sum.

Step 5: Print the final sum of natural numbers and store in the variable sum.

Step 6: Increment the value of i by 1 in each iteration.

Step 7: The program execution ends after printing the sum of natural numbers up to the given integer n.

CODE

```
n <- as.integer(readline("Enter a number:")
Enter a number:6
sum <- 0
i <- 1
while(i<=n){
 sum=sum+i
 i=i+1
}
print(sum)
```

OUTPUT

```
[1] 21
```

Ex. No:6	Factorial of a Given Number
Date: 23.01.2024	

AIM

Compute the factorial of a given number and display the result.

PROCEDURE

Step 1: Take an integer input n from the user for which the factorial has to be computed.

Step 2: Initialize a variable factorial=1.

Step 3: Handle special cases:

- If n is 0, the factorial is 1.
- If n is 1, the factorial is 1.

Step 4: Use a loop typically for loop to compute the factorial

- Multiply the factorial result by each integer from 1 to n.

Step 5: Display the computed factorial.

CODE

```
num=readline("Enter a Number:")
Enter a Number:10
num=as.integer(num)
factorial=1
if(num<0)
 {
  print("Factorial does not exisit")
 } else if(num==0){
 print("Factorial of 0 is 1")
 } else{
 for(i in 1:num)
 {
 factorial=factorial*i
 }
 print(paste("Factorial of the Given Number:" ,factorial))
 }
```

OUTPUT

[1] "Factorial of the Given Number: 3628800"

Ex. No:7	Fibonacci Series Generation
Date: 02.02.2024	

AIM

Generate the Fibonacci sequence up to a specified number of terms and display the sequence.

PROCEDURE

Step 1: Define the number of terms in the Fibonacci sequence.

Step 2: Initialize variables to store the first two Fibonacci numbers 0 and 1.

Step 3: Print or store these initial Fibonacci numbers.

Step 4: Use a loop typically for or while loop to generate the subsequent Fibonacci numbers

Step 5: Calculate the next Fibonacci number as the sum of the previous two.

Step 6: Update the variables storing the previous two Fibonacci numbers.

Step 7: Repeat this process until the desired number of terms is reached.

Step 8: Display the generated Fibonacci sequence.

CODE

```
num<-readline("Enter the Number:")
Enter the Number:10
num<-as.integer(num)
num1<-0
num2<-1
count=2
if(num<=0)
+ {
+ print("Enter a Positive Number")
+ }else {
+ if(num==1)
+ {
+ print("Fibonacci Sequence:")
+ print(num1)
```

```
+ }else{
+ print("Fibonacci Sequence:")
+ print(num1)
+ print(num2)
+ while(count<num)
+ {
+ nxt=num1+num2
+ print(nxt)
+ num1=num2
+ num2=nxt
+ count=count+1
+ }
+ }
+ }
```

OUTPUT

```
[1] "Fibonacci Sequence:"
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
[1] 13
[1] 21
[1] 34
```

Ex. No:8	Check the Given Number is Prime or not
Date: 07.02.2024	

AIM

Determine whether a given input number by the user is prime or not, and then display the result.

PROCEDURE

Step 1: Take input from the user for the number to be checked.

Step 2: Initialize a variable flag to 0.

Step 3: Check if the input number is greater than 1.

Step 4: If the number is greater than 1, start a loop from 2 to num-1 to check for factors.

Step 5: If the remainder of the division of the input number by any number within this range is 0, then it is not a prime number.

Step 6: Set flag to 0 and break out of the loop.

Step 7: If the input number is 2, set flag to 1 as 2 is a prime number.

Step 8: If flag is 1, print the number is a prime number, else print the number is not a prime number.

CODE

```
num = as.integer(readline(prompt="Enter a number: "))
Enter a number: 10
flag = 0
# prime numbers are greater than 1
if(num > 1) {
   # check for factors
   flag = 1
   for(i in 2:(num-1)) {
      if ((num %% i) == 0) {
         flag = 0
         break
      }
```

```
    }
}
if(num == 2)    flag = 1
if(flag == 1) {
    print(paste(num,"is a prime number"))
} else {
    print(paste(num,"is not a prime number"))
}
```

OUTPUT

```
[1] "10 is not a prime number"
```

Ex. No:9	Implementation of Decision Tree Algorithm using Titanic Dataset
Date: 21.02.2024	

AIM

To show the implementation of Decision Tree using titanic dataset.

PROCEDURE

Step 1: Load the dataset

Step 2: Create train and test set.

i. train <- train_test_split(titanic, 0.8, train = TRUE)
ii. test <- train_test_split(titanic, 0.8, train = FALSE)

Step 3: Build the model.

i. Install Package **rpart**
ii. Load the package
iii. fit <- rpart(survived~., data = train, method ='class')

Step 4: Make prediction.

predicted = predict(fit, test, type = 'class')

Step 5: Plot the decision tree for visualization.

Step 6: Make predictions on the test set and evaluate model accuracy.

CODE

```
install.packages(“dplyr”)
library(dplyr)
install.packages(“caret”)
library(caret)
install.packages(“ggplot2”)
library("ggplot2")
install.packages(“rpart.plot”)
library(“rpart.plot”)
titanic<-read.csv("E:/R Programs/titanic_data.csv")
head(titanic)
tail(titanic)
```

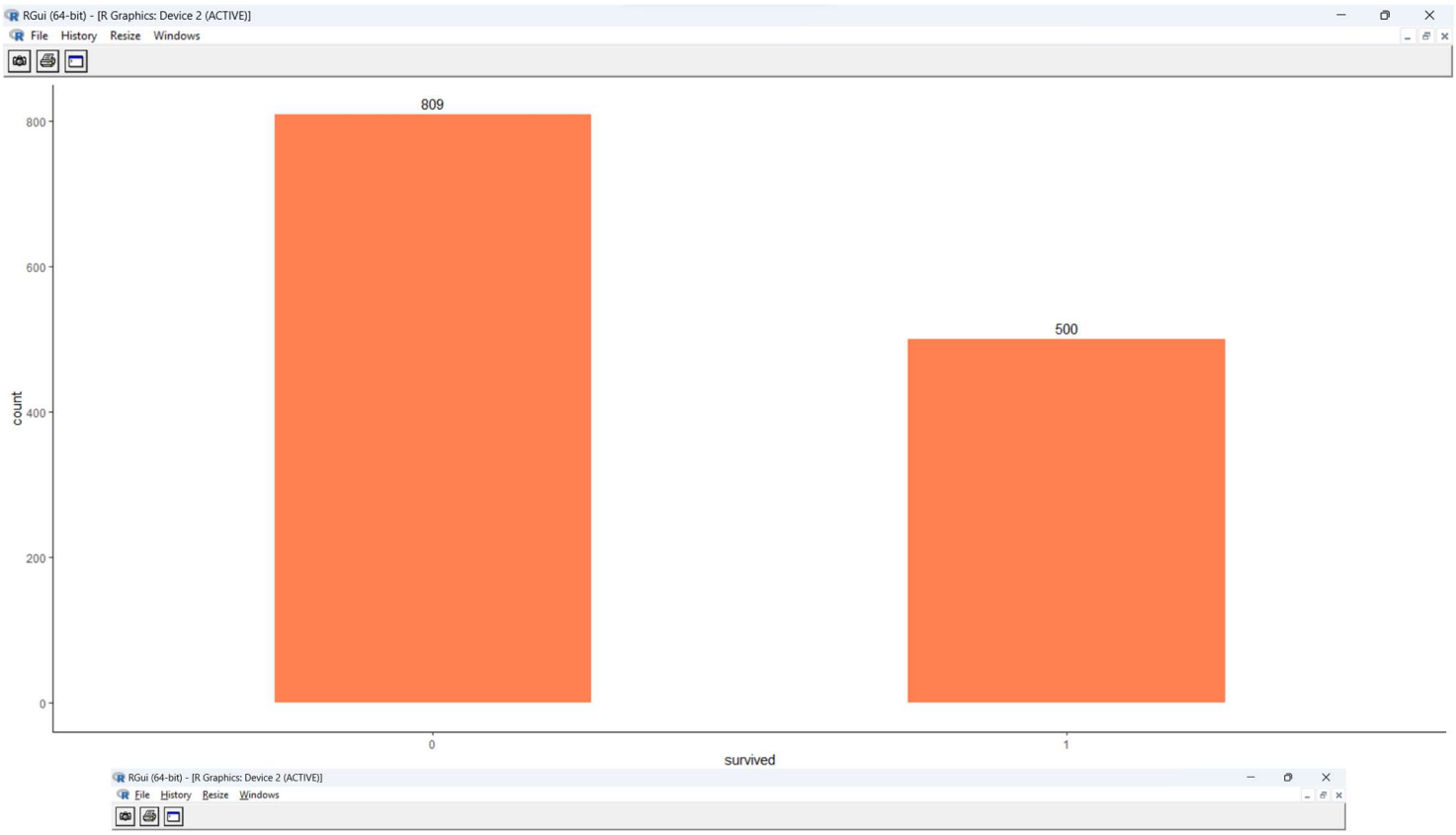
```
titanic=select(titanic,survived,pclass,sex,sibsp,parch)
titanic=na.omit(titanic)
str(titanic)
```

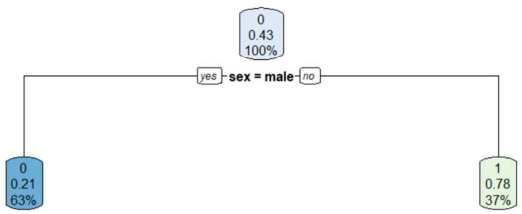
```
'data.frame':   1309 obs. of  5 variables:
 $ survived: Factor w/ 2 levels "0","1": 2 2 1 1 1 2 2 1 2 1 ...
 $ pclass  : Ord.factor w/ 3 levels "3"<"2"<"1": 3 3 3 3 3 3 3 3 3 3 ...
 $ sex     : chr  "female" "male" "female" "male" ...
 $ sibsp   : int  0 1 1 1 1 0 1 0 2 0 ...
 $ parch   : int  0 2 2 2 2 0 0 0 0 0 ...
```

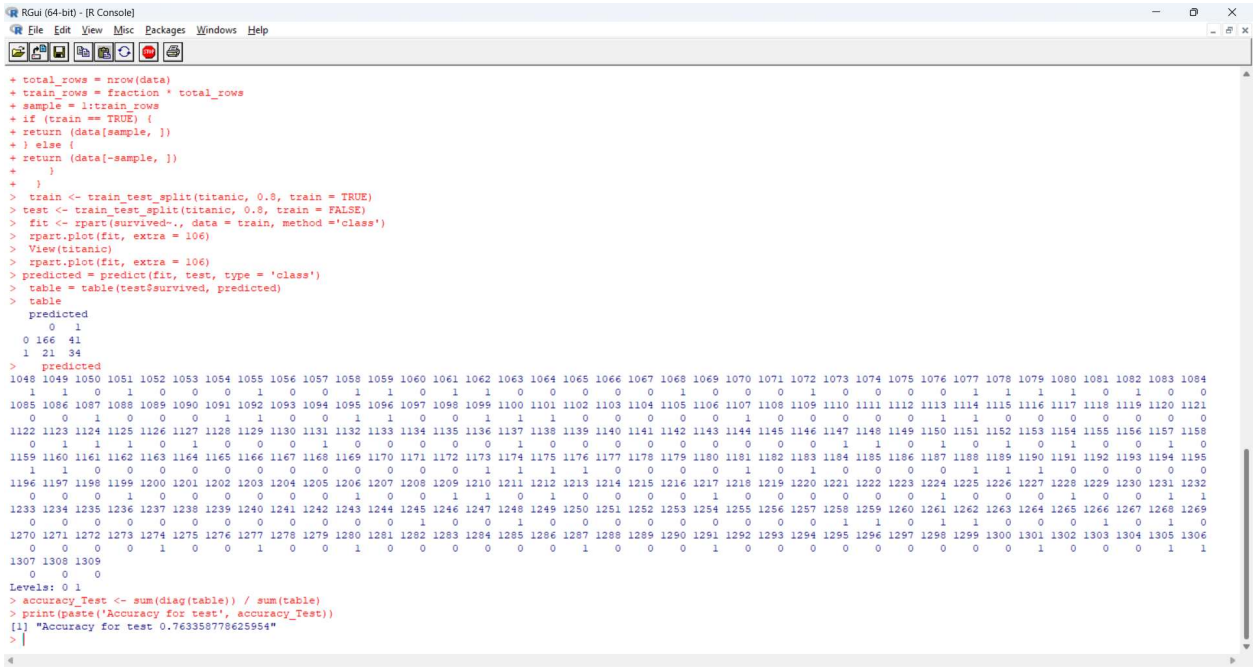
```
titanic$survived = factor(titanic$survived)
titanic$pclass = factor(titanic$pclass, order=TRUE, levels = c(3, 2, 1))
ggplot(titanic,aes(x = survived))+
geom_bar(width=0.5, fill = "coral") +
 geom_text(stat='count', aes(label=stat(count)), vjust=-0.5) +
theme_classic()

train_test_split = function(data, fraction = 0.8, train = TRUE) {
total_rows = nrow(data)
train_rows = fraction * total_rows
sample = 1:train_rows
if (train == TRUE) {
return (data[sample, ])
} else {
return (data[-sample, ])
 }
}
train <- train_test_split(titanic, 0.8, train = TRUE)
test <- train_test_split(titanic, 0.8, train = FALSE)
fit <- rpart(survived~., data = train, method ='class')
rpart.plot(fit, extra = 106)
predicted = predict(fit, test, type = 'class')
table = table(test$survived, predicted)
table
predicted
    0   1
0 166  41
1  21  34

accuracy_Test <- sum(diag(table)) / sum(table)
print(paste('Accuracy for test', accuracy_Test))
```

OUTPUT

[1] "Accuracy for test 0.763358778625954"






RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
+ total_rows = nrow(data)
+ train_rows = fraction * total_rows
+ sample = 1:train_rows
+ if (train == TRUE) {
+ return (data[sample,])
+ } else {
+ return (data[-sample,])
+ }
+ }
> train <- train_test_split(titanic, 0.8, train = TRUE)
> test <- train_test_split(titanic, 0.8, train = FALSE)
> fit <- rpart(survived~., data = train, method ='class')
> rpart.plot(fit, extra = 106)
> View(titanic)
> rpart.plot(fit, extra = 106)
> predicted = predict(fit, test, type = 'class')
> table = table(test$survived, predicted)
> table
predicted
0 1
0 166 41
1 21 34
> predicted
1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084
1 1 0 1 0 0 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0
1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121
0 0 1 0 0 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0
1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158
0 1 1 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0
1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195
1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0
1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232
0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1
1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0
1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306
0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1
1307 1308 1309
0 0 0
Levels: 0 1
> accuracy_Test <- sum(diag(table)) / sum(table)
> print(paste('Accuracy for test', accuracy_Test))
[1] "Accuracy for test 0.763358778625954"
>

Ex. No:10	Implementation of Titanic Survival Prediction using Naive Bayes Algorithm
Date: 01.03.2024	

AIM

To show the implementation of Naive Bayes algorithm using titanic dataset.

PROCEDURE

Step 1: Load the dataset.

Step 2: Create train and test set.

i) train <- train_test_split(titanic, 0.8, train = TRUE)
ii) test <- train_test_split(titanic, 0.8, train = FALSE)

Step 3: Build the model.

i. Install Package e1071
ii. Load the package
iii. nb_model <- naiveBayes(survived ~ ., data = train)

Step 4: Make prediction.

nb_predict <- predict(nb_model, test)

Step 5: Measure performance evaluation by confusion matrix and calculate accuracy.

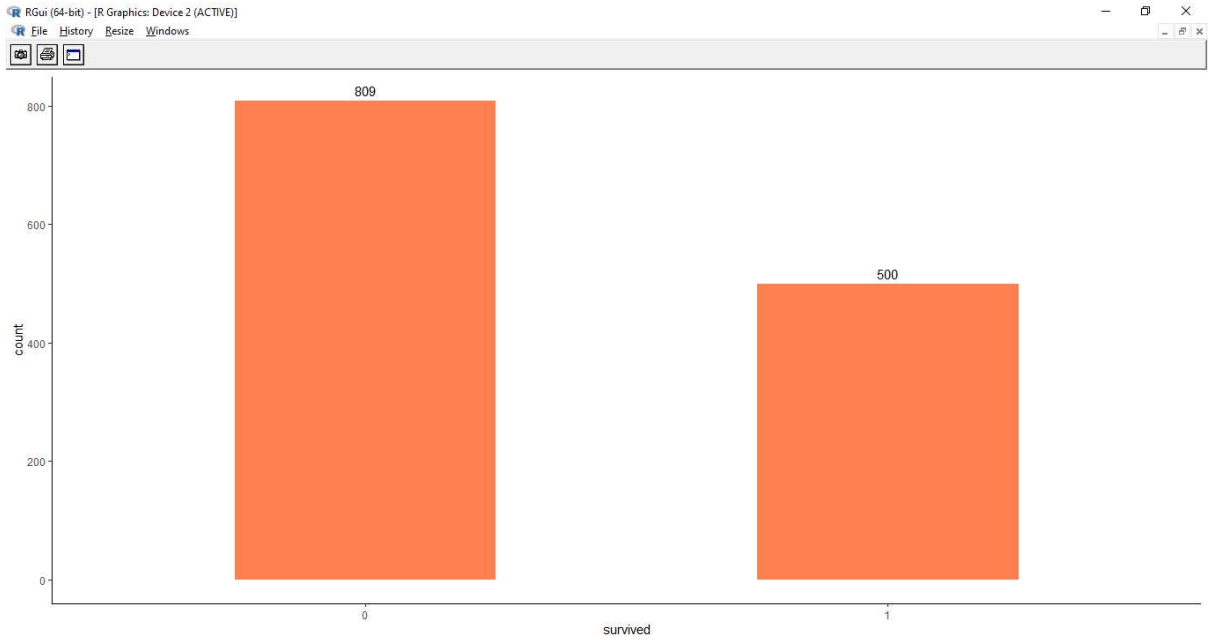
CODE

```
install.packages(“dplyr”)
library(dplyr)
install.packages(“caret”)
library(caret)
install.packages("e1071")
library("e1071")
install.packages(“ggplot2”)
library("ggplot2")
titanic<-read.csv("D:/Faculty/R Programming/titanic_data.csv")
head(titanic)
tail(titanic)
titanic=select(titanic,survived,pclass,sex,sibsp,parch)
titanic=na.omit(titanic)
str(titanic)
'data.frame':   1309 obs. of  5 variables:
 $ survived: int  1 1 0 0 0 1 1 0 1 0 ...
```

```
 $ pclass  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ sex     : chr  "female" "male" "female" "male" ...
 $ sibsp   : int  0 1 1 1 1 0 1 0 2 0 ...
 $ parch   : int  0 2 2 2 2 0 0 0 0 0 ...

titanic$survived = factor(titanic$survived)
titanic$pclass = factor(titanic$pclass, order=TRUE,levels = c(3, 2, 1))
ggplot(titanic,aes(x = survived))+  geom_bar(width=0.5, fill = "coral") +
geom_text(stat='count', aes(label=stat(count)), vjust=-0.5) + theme_classic()
train_test_split = function(data, fraction = 0.8, train = TRUE)
+ {
+     total_rows = nrow(data)
+     train_rows = fraction * total_rows
+     sample = 1:train_rows
+     if (train == TRUE) {
+       return (data[sample, ])
+     } else {
+       return (data[-sample, ])
+     }
+   }
train <- train_test_split(titanic, 0.8, train = TRUE)
test <- train_test_split(titanic, 0.8, train = FALSE)
nb_model = naiveBayes(survived ~., data=train)
nb_predict = predict(nb_model,test)
table_mat = table(nb_predict, test$survived)
table_mat

nb_predict   0   1
         0 173  21
         1  34  34

nb_accuracy = sum(diag(table_mat)) / sum(table_mat)
paste("The accuracy is : ", nb_accuracy)
```

OUTPUT

[1] "The accuracy is : 0.790076335877863"



Ex. No:11	Implementation of K Nearest Neighbor using Iris dataset
Date: 06.03.2024	

AIM

To show the Implementation of KNN using iris dataset.

PROCEDURE

Step 1: Load the dataset

Step 2: Install the e1071, caTools, and class packages

Step 3: Split the data into train and test sets

i. split <- sample.split(iris, SplitRatio = 0.7)

ii. train_cl <- subset(iris, split == "TRUE")

iii. test_cl <- subset(iris, split == "FALSE")

Step 4: Build the KNN Model

classifier_knn <- knn(train = train_scale, test = test_scale, cl = train_cl$Species, k = 1)

Step 5: Generate a confusion matrix to evaluate performance of the model on the test set.

Step 6: Calculate out of Sample error for the model.

Step 7: Evaluate Different Values of K (3, 5, 7, 15, 19) for accuracy.

Step 8: Use ggplot2 to create a line plot showing model accuracy for different values of k

CODE

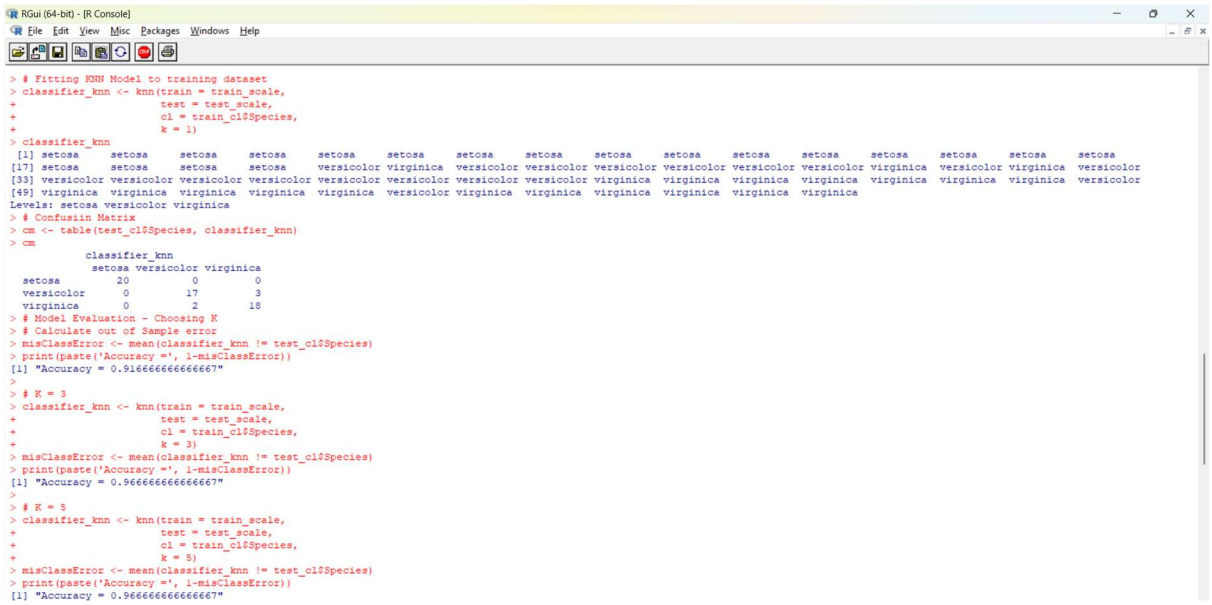
```
iris<-read.csv("D:/R programming/iris_dataset.csv")
# Loading data
data(iris)
 head(iris)
 tail(iris)
# Structure
str(iris)
# Installing Packages
install.packages("e1071")
install.packages("caTools")
install.packages("class")
# Loading package
```

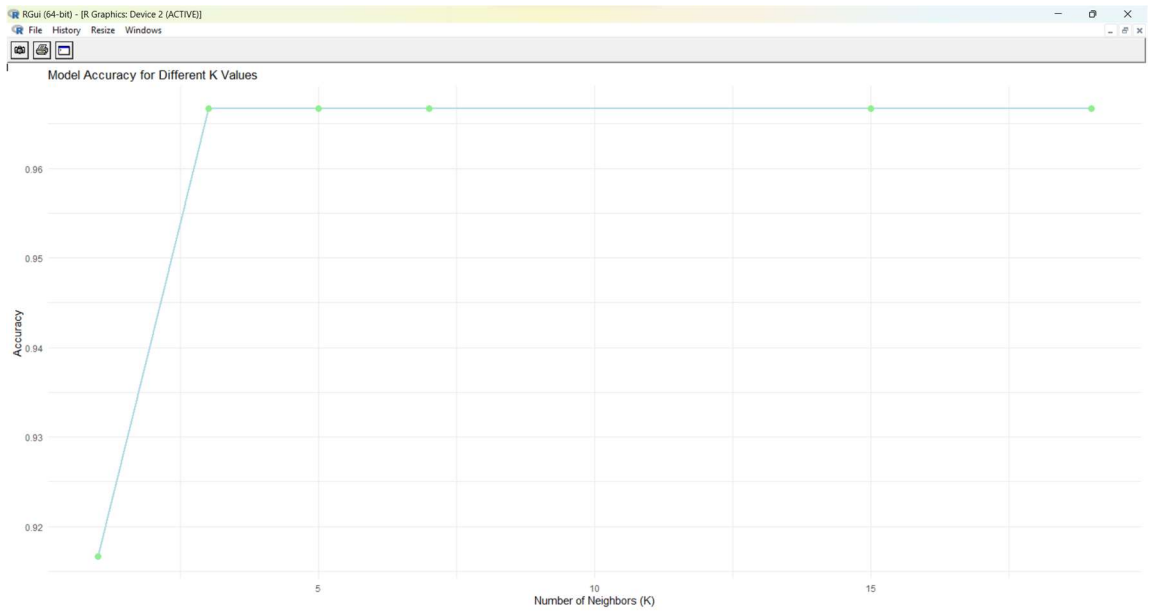
```
library(e1071)
library(caTools)
library(class)

# Splitting data into train and test data
split <- sample.split(iris, SplitRatio = 0.7)
train_cl <- subset(iris, split == "TRUE")
test_cl <- subset(iris, split == "FALSE")
# Feature Scaling
train_scale <- scale(train_cl[, 1:4])
test_scale <- scale(test_cl[, 1:4])
head(train_scale)
head(test_scale)

# Fitting KNN Model to training dataset
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 1)
classifier_knn
# Confusion Matrix
cm <- table(test_cl$Species, classifier_knn)
cm

# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 3
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 3)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 5
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 5)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))
```

```
# K = 7
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 7)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 15
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 15)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 19
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 19)
misClassError <- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

library(ggplot2)

# Data preparation
k_values <- c(1, 3, 5, 7, 15, 19)

# Calculate accuracy for each k value
accuracy_values <- sapply(k_values, function(k) {
  classifier_knn <- knn(train = train_scale,
                        test = test_scale,
                        cl = train_cl$Species,
                        k = k)
  1 - mean(classifier_knn != test_cl$Species)
})

# Create a data frame for plotting
accuracy_data <- data.frame(K = k_values, Accuracy = accuracy_values)


# Plotting
```

```
ggplot(accuracy_data, aes(x = K, y = Accuracy)) +
  geom_line(color = "lightblue", size = 1) +
  geom_point(color = "lightgreen", size = 3) +
  labs(title = "Model Accuracy for Different K Values",
       x = "Number of Neighbors (K)",
       y = "Accuracy") +
  theme_minimal()
```

OUTPUT





Ex. No:12	Implementation of Random Forest Algorithm using Iris Dataset
Date: 08.03.2024	

AIM

To show the Implementation of Random Forest using iris dataset.

PROCEDURE

Step 1: Load the dataset

Step 2: Create train and test set.

i. split <- sample.split(iris, SplitRatio = 0.7)

ii. train <- subset(iris, split == "TRUE")

iii. test <- subset(iris, split == "FALSE")

Step 3: Build the model.

i. Set control parameters for model training

ii. Set seed for reproducibility

iii. Train the Random Forest model

rf <- train(Species ~ ., data=train, method="rf", metric=metric, tuneLength=15, trControl=control)

Step 4: Use Grid Search for Hyperparameter Tuning

tunegrid <- expand.grid(.mtry=c(1:4))

Step 5: Plot the grid search.

CODE

```
iris<-read.csv("D:/R programming/iris_dataset.csv")
# Loading data
data(iris)
head(iris)
tail(iris)
# Structure
str(iris)

# Installing package
install.packages(“caTools”)
```

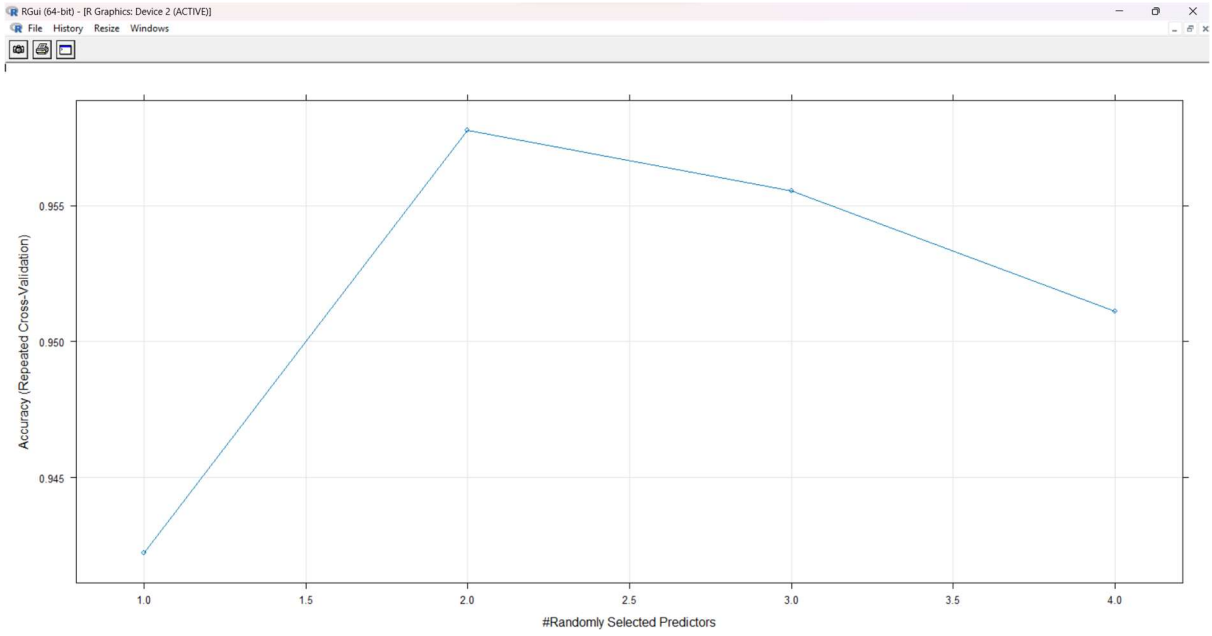
```
library(“caTools”)
install.packages(“randomForest”)
library(“randomForest”)
install.packages(“caret”)
library("caret")
# Splitting data in train and test data
split <- sample.split(iris, SplitRatio = 0.7)
split
[1] FALSE  TRUE  TRUE  TRUE FALSE

train <- subset(iris, split == "TRUE")
test <- subset(iris, split == "FALSE")
# Fitting Random Forest to the train dataset
control <- trainControl(method="repeatedcv", number=10, repeats=3)
seed <- 7
metric <- "Accuracy"
set.seed(seed)
rf <- train(Species~., data=iris, method="rf", metric=metric, tuneLength=15,
trControl=control)
print(rf)
```

```
Random Forest

150 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9555556  0.9333333
  3     0.9577778  0.9366667
  4     0.9600000  0.9400000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 4.
```

```
# Grid Search
tunegrid <- expand.grid(.mtry=c(1:4))
rf_gridsearch <- train(Species~., data=iris, method="rf", metric=metric,
tuneGrid=tunegrid, trControl=control)
print(rf_gridsearch)
```

```
Random Forest

150 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  1     0.9422222  0.9133333
  2     0.9577778  0.9366667
  3     0.9555556  0.9333333
  4     0.9511111  0.9266667

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

plot(rf_gridsearch)



Ex. No:13	Implementation of K Means Clustering
Date: 20.03.2024	

AIM

To show the implementation of K Means Algorithm using USArrest Dataset

PROCEDURE

Step 1: Load the dataset

Step 2: Choose the number K clusters.

Step 3: Select at random K points

Step 4: Assign each data point to closest centroid that forms K clusters

Step 5: Compute and place the new centroid of each centroid

Step 6: Reassign each data point to new cluster.

CODE

```
mydata<-read.csv("D:/R programming/USArrests.csv")
head(mydata)
```

```
     rownames Murder Assault UrbanPop Rape
1     Alabama   13.2     236       58 21.2
2      Alaska   10.0     263       48 44.5
3     Arizona    8.1     294       80 31.0
4    Arkansas    8.8     190       50 19.5
5  California    9.0     276       91 40.6
6    Colorado    7.9     204       78 38.7
```

```
tail(mydata)
```

```
         rownames Murder Assault UrbanPop Rape
45        Vermont    2.2      48       32 11.2
46       Virginia    8.5     156       63 20.7
47     Washington    4.0     145       73 26.2
48  West Virginia    5.7      81       39  9.3
49      Wisconsin    2.6      53       66 10.8
50        Wyoming    6.8     161       60 15.6
```

```
mydata$rownames
```

```
 [1] "Alabama"       "Alaska"       "Arizona"      "Arkansas"     "California"   "Colorado"     "Connecticut"  "Delaware"     "Florida"      "Georgia"
[11] "Hawaii"        "Idaho"        "Illinois"     "Indiana"      "Iowa"         "Kansas"       "Kentucky"     "Louisiana"    "Maine"        "Maryland"
[21] "Massachusetts" "Michigan"     "Minnesota"    "Mississippi"  "Missouri"     "Montana"      "Nebraska"     "Nevada"       "New Hampshire" "New Jersey"
[31] "New Mexico"    "New York"     "North Carolina" "North Dakota" "Ohio"         "Oklahoma"     "Oregon"       "Pennsylvania" "Rhode Island" "South Carolina"
[41] "South Dakota"  "Tennessee"    "Texas"        "Utah"         "Vermont"      "Virginia"     "Washington"   "West Virginia" "Wisconsin"    "Wyoming"
```

```
install.packages("factoextra")
library(factoextra)
#Scales/standardizes the data
USdata <- scale(mydata[,-1])
head(USdata)
```

```
          Murder   Assault   UrbanPop         Rape
[1,] 1.24256408 0.7828393 -0.5209066 -0.003416473
[2,] 0.50786248 1.1068225 -1.2117642  2.484202941
[3,] 0.07163341 1.4788032  0.9989801  1.042878388
[4,] 0.23234938 0.2308680 -1.0735927 -0.184916602
[5,] 0.27826823 1.2628144  1.7589234  2.067820292
[6,] 0.02571456 0.3988593  0.8608085  1.864967207
```

#K-means clustering with 2, 3, 4 and 5 centers
kmeans.cluster_2 <- kmeans(USdata, centers = 2, nstart = 20)
kmeans.cluster_2

```
K-means clustering with 2 clusters of sizes 30, 20

Cluster means:
     Murder    Assault   UrbanPop       Rape
1 -0.669956 -0.6758849 -0.1317235 -0.5646433
2  1.004934  1.0138274  0.1975853  0.8469650

Clustering vector:
 [1] 2 2 2 1 2 2 1 1 2 2 1 1 2 1 1 1 1 2 1 2 1 2 1 2 2 1 1 2 1 1 2 2 2 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 56.11445 46.74796
 (between_SS / total_SS =  47.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

kmeans.cluster_3 <- kmeans(USdata, centers = 3, nstart = 20)
kmeans.cluster_3

```
K-means clustering with 3 clusters of sizes 17, 13, 20

Cluster means:
      Murder    Assault   UrbanPop       Rape
1 -0.4469795 -0.3465138  0.4788049 -0.2571398
2 -0.9615407 -1.1066010 -0.9301069 -0.9667633
3  1.0049340  1.0138274  0.1975853  0.8469650

Clustering vector:
 [1] 3 3 3 1 3 3 1 1 3 3 1 2 3 1 2 1 2 3 2 3 1 3 2 3 3 2 2 3 2 1 3 3 3 2 1 1 1 1 1 3 2 3 3 1 2 1 1 2 2 1

Within cluster sum of squares by cluster:
[1] 19.62285 11.95246 46.74796
 (between_SS / total_SS =  60.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

kmeans.cluster_4 <- kmeans(USdata, centers = 4, nstart = 20)
kmeans.cluster_4

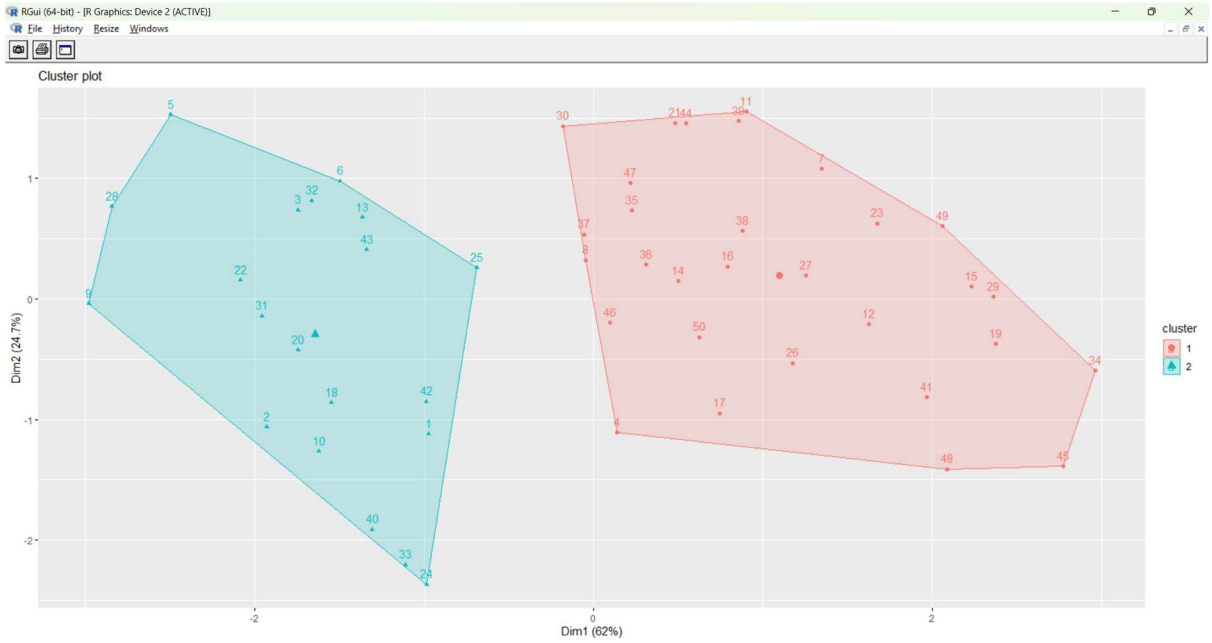
```
K-means clustering with 4 clusters of sizes 13, 13, 16, 8

Cluster means:
      Murder    Assault   UrbanPop        Rape
1 -0.9615407 -1.1066010 -0.9301069 -0.96676331
2  0.6950701  1.0394414  0.7226370  1.27693964
3 -0.4894375 -0.3826001  0.5758298 -0.26165379
4  1.4118898  0.8743346 -0.8145211  0.01927104

Clustering vector:
 [1] 4 2 2 4 2 2 3 3 2 4 3 1 2 3 1 3 1 4 1 2 3 2 1 4 2 1 1 2 1 3 2 2 4 1 3 3 3 3 3 4 1 4 2 3 1 3 3 1 1 3

Within cluster sum of squares by cluster:
[1] 11.952463 19.922437 16.212213  8.316061
 (between_SS / total_SS =  71.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

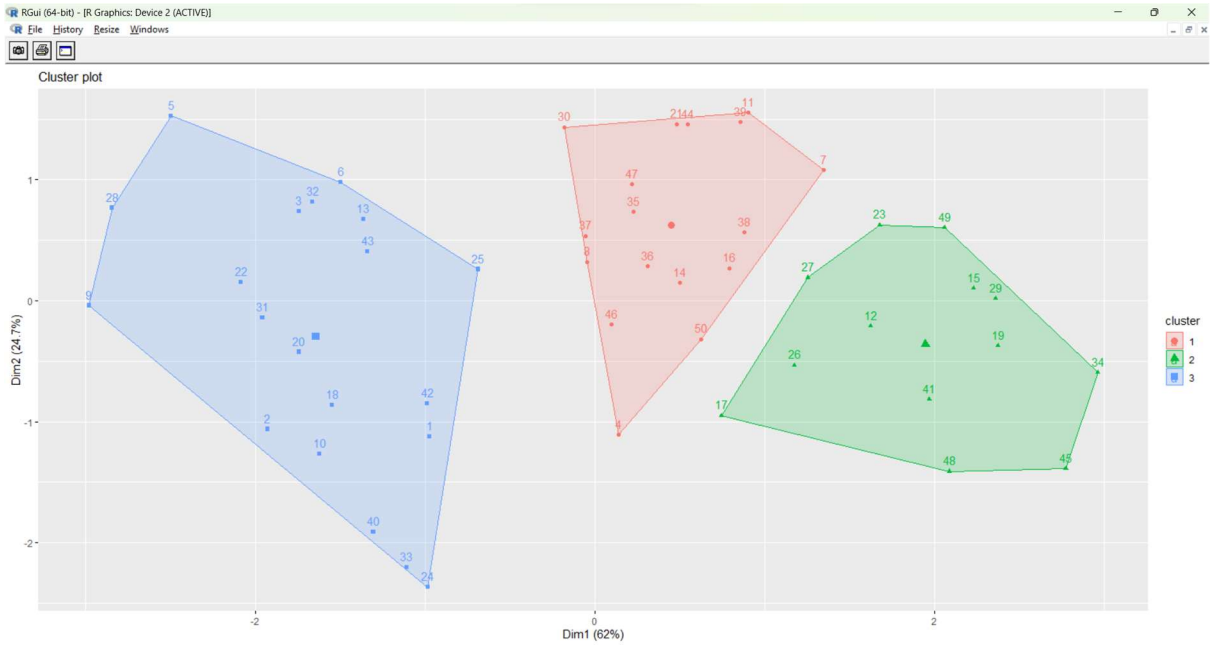
kmeans.cluster_2$cluster
[1] 2 2 2 1 2 2 1 1 2 2 1 1 2 1 1 1 1 2 1 2 1 2 1 2 2 1 1 2 1 1 2 2 2 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1
kmeans.cluster_3$cluster
[1] 3 3 3 1 3 3 1 1 3 3 1 2 3 1 2 1 2 3 2 3 1 3 2 3 3 2 2 3 2 1 3 3 3 2 1 1 1 1 1 3 2 3 3 1 2 1 1 2 2 1
kmeans.cluster_4$cluster
[1] 4 2 2 4 2 2 3 3 2 4 3 1 2 3 1 3 1 4 1 2 3 2 1 4 2 1 1 2 1 3 2 2 4 1 3 3 3 3 3 4 1 4 2 3 1 3 3 1 1 3
kmeans.cluster_5$cluster
[1] 4 1 1 5 1 1 2 2 1 4 2 3 1 5 3 5 5 4 3 1 2 1 3 4 5 5 5 1 3 2 1 1 4 3 2 5 5 2 2 4 3 4 1 2 3 5 2 3 3 5

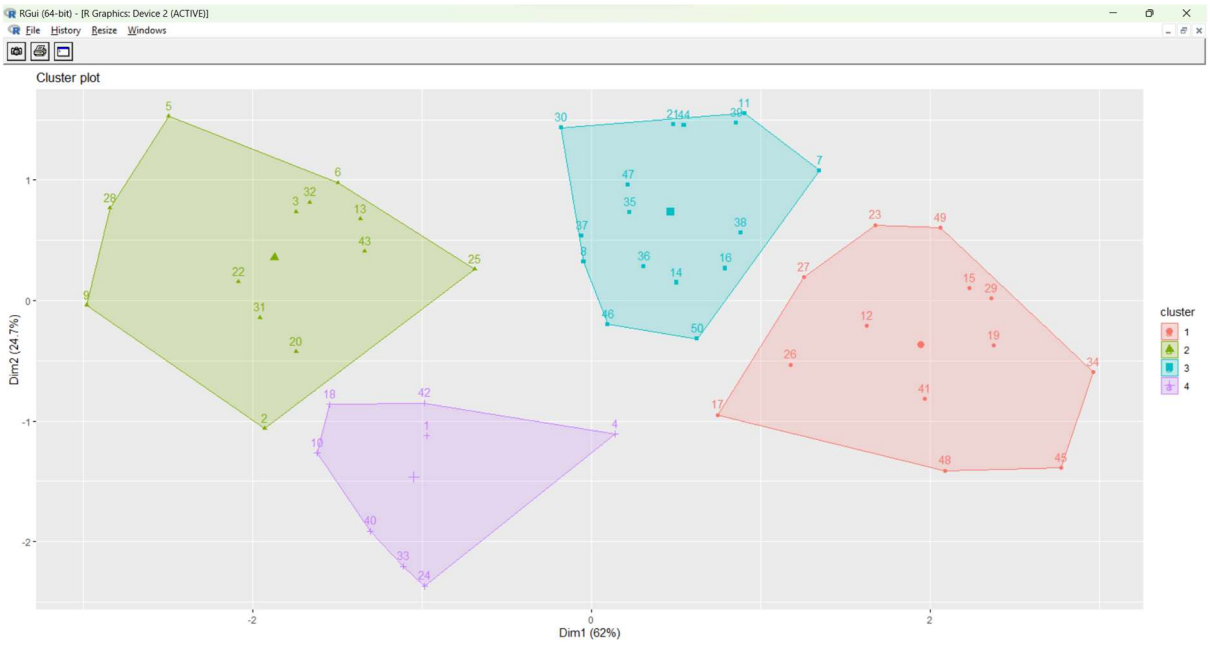
fviz_cluster(kmeans.cluster_2, data = USdata)
fviz_cluster(kmeans.cluster_3, data = USdata)
fviz_cluster(kmeans.cluster_4, data = USdata)
fviz_cluster(kmeans.cluster_5, data = USdata)

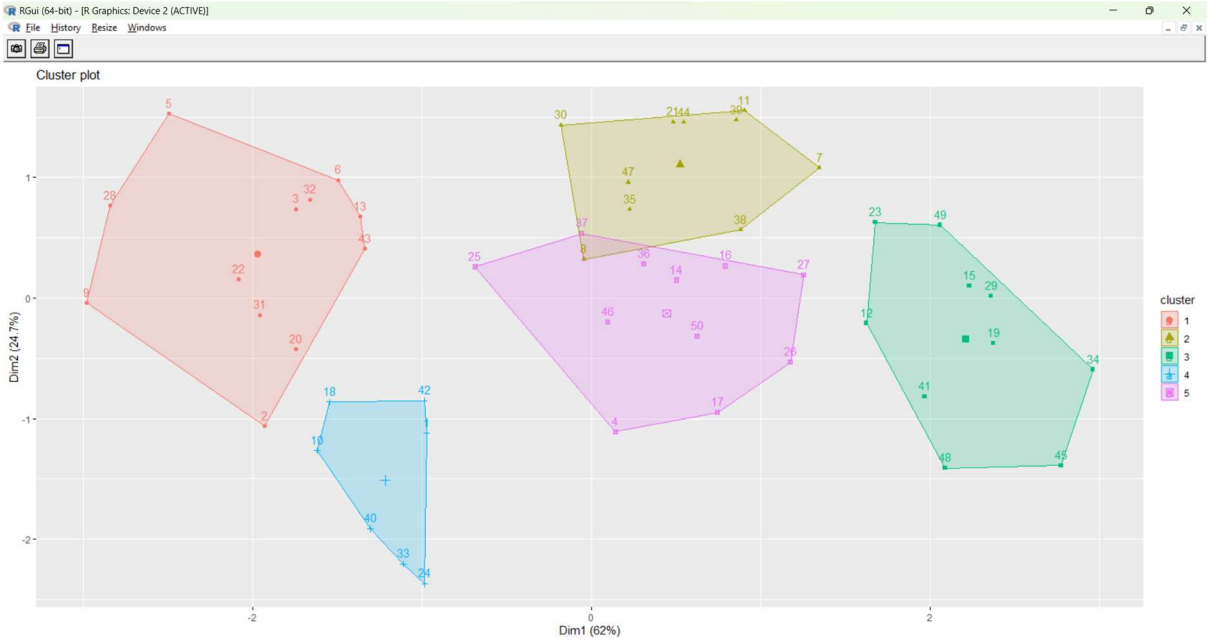
#Creates a plot showing the silhouette and elbow methods in determining the amount of optimal centers.
fviz_nbclust(USdata, FUN = kmeans, method = "silhouette")

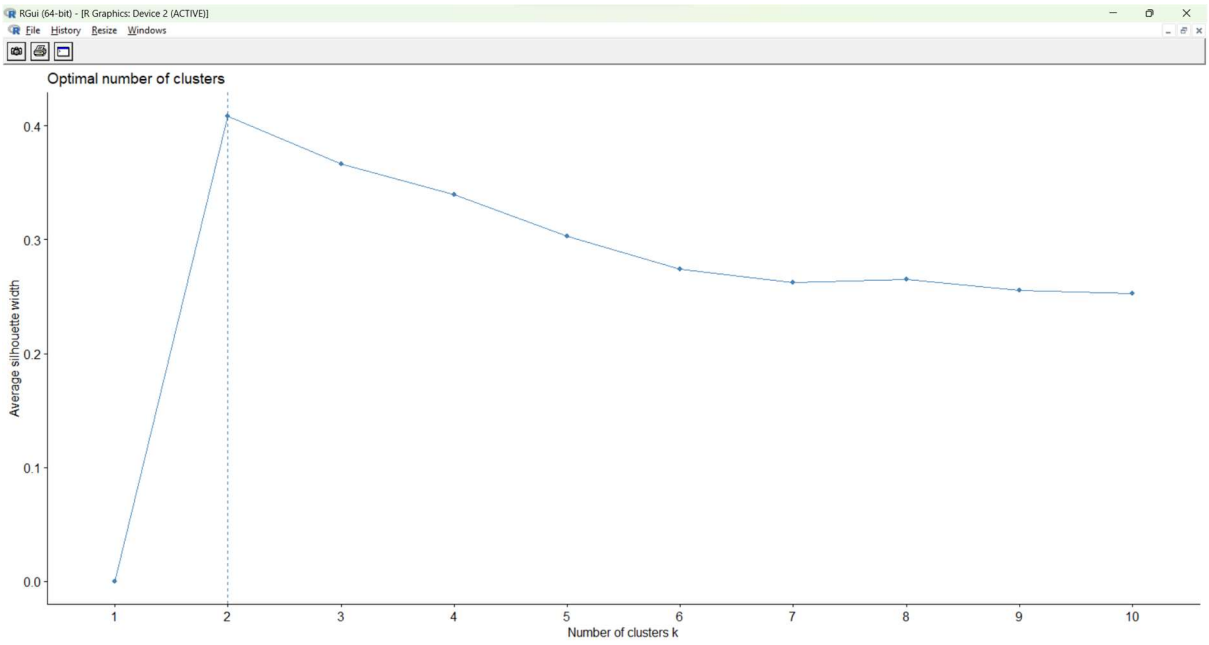
OUTPUT


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Cluster plot
Dim2 (24.7%)
Dim1 (62%)
cluster
1
2


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Cluster plot
Dim2 (24.7%)
Dim1 (62%)
cluster
1
2
3


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Cluster plot
Dim2 (24.7%)
Dim1 (62%)
cluster
1
2
3
4


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Cluster plot
Dim2 (24.7%)
Dim1 (62%)
cluster
1
2
3
4
5


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Optimal number of clusters
Average silhouette width
0.0
0.1
0.2
0.3
0.4
1
2
3
4
5
6
7
8
9
10
Number of clusters k

Ex. No:14	Data Visualization in R
Date: 27.03.2024	

AIM

To illustrate the concept of data visualization in R using air quality dataset.

PROCEDURE

Step 1: Horizontal bar plot displays categorical data using horizontal bars, with the length of each bar representing the frequency or value of the corresponding category.

Step 2: Vertical Bar Plot represents categorical data using vertical bars, with the height of each bar indicating the frequency or value of the corresponding category.

Step 3: Histogram illustrates the distribution of numerical data by dividing it into bins and displaying the frequency of observations within each bin as vertical bars.

Step 4: Box plot represents average wind speed

Step 5: Multiple Box plots, each represents an Air Quality Parameter

Step 6: Scatter plot illustrates Ozone Concentration per month

Step 7: Visualize data patterns using heatmap

Step 8: Prepare variables, add Titles and Labeling Axes and plot the 3D surface.

CODE

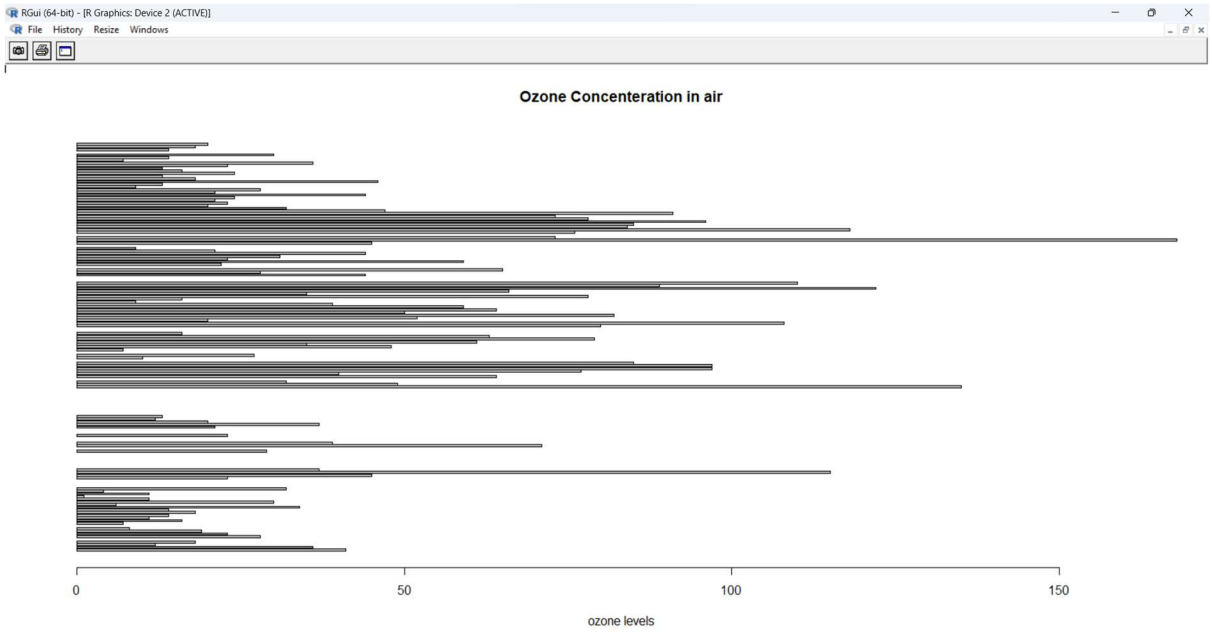
```
mydata<-read.csv("D:/R programming/airquality.csv")
str(mydata)
# Horizontal Bar Plot for Ozone concentration in air
barplot(airquality$Ozone,
        main = 'Ozone Concenteration in air',
        xlab = 'ozone levels', horiz = TRUE)


# Vertical Bar Plot for Ozone concentration in air
barplot(airquality$Ozone, main = 'Ozone Concenteration in air',
          xlab = 'ozone levels', col ='blue', horiz = FALSE)
```

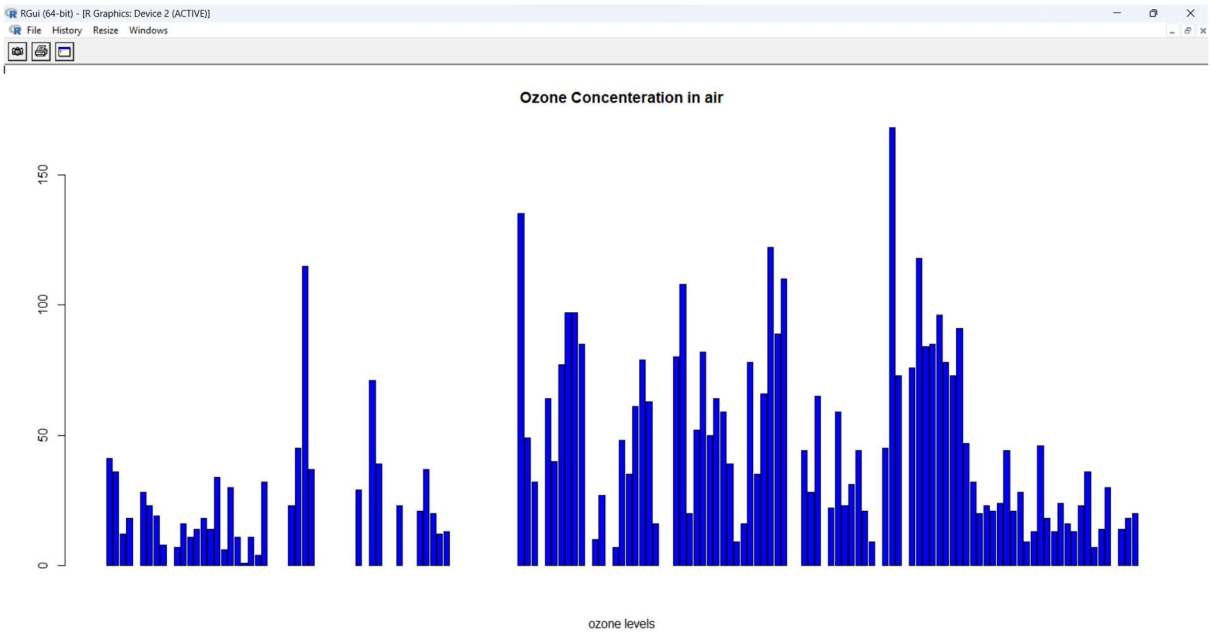
```
# Histogram for Maximum Daily Temperature
data(airquality)
hist(airquality$Temp, main ="La Guardia Airport's\
Maximum Temperature(Daily)",
   xlab ="Temperature(Fahrenheit)",
   xlim = c(50, 125), col ="yellow",
   freq = TRUE)


# Box plot for average wind speed
data(airquality)
boxplot(airquality$Wind, main = "Average wind speed\
at La Guardia Airport",
       xlab = "Miles per hour", ylab = "Wind",
       col = "orange", border = "brown",
       horizontal = TRUE, notch = TRUE)


# Multiple Box plots, each representing an Air Quality Parameter
boxplot(airquality[, 0:4],
       main ='Box Plots for Air Quality Parameters')


# Scatter plot for Ozone Concentration per month
data(airquality)
plot(airquality$Ozone, airquality$Month,
    main ="Scatterplot Example",
   xlab ="Ozone Concentration in parts per billion",
   ylab =" Month of observation ", pch = 19)


# Set seed for reproducibility
# set.seed(110)
# Create example data
data <- matrix(rnorm(50, 0, 5), nrow = 5, ncol = 5)
# Column names
colnames(data) <- paste0("col", 1:5)
rownames(data) <- paste0("row", 1:5)
# Draw a heatmap
heatmap(data)
```

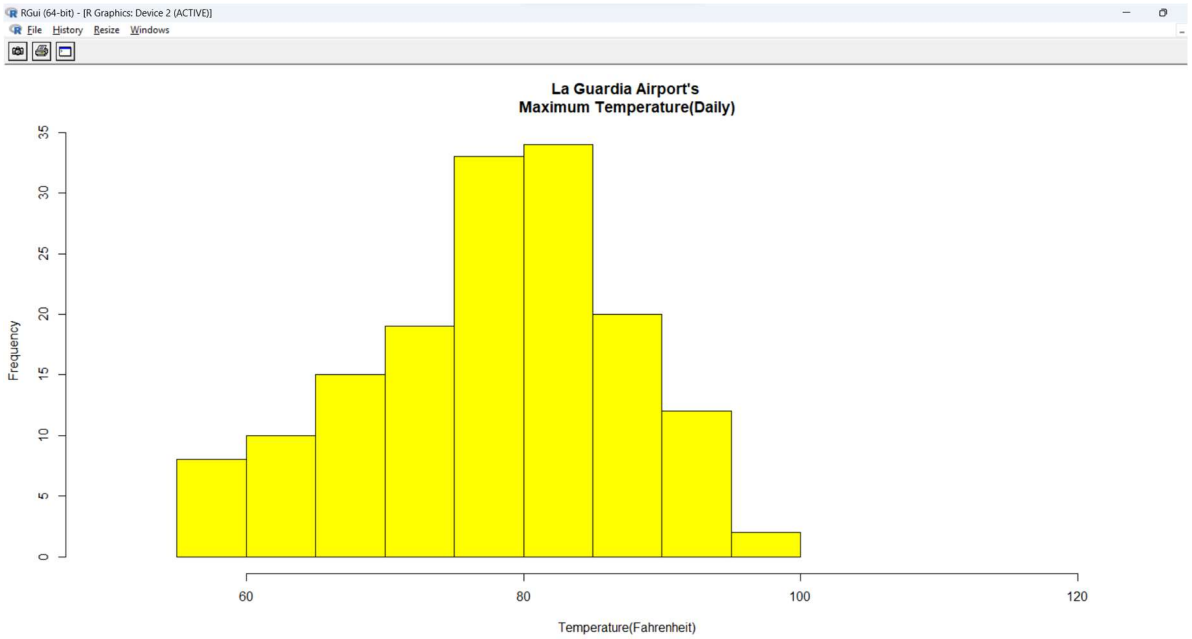
3D Graphs in R

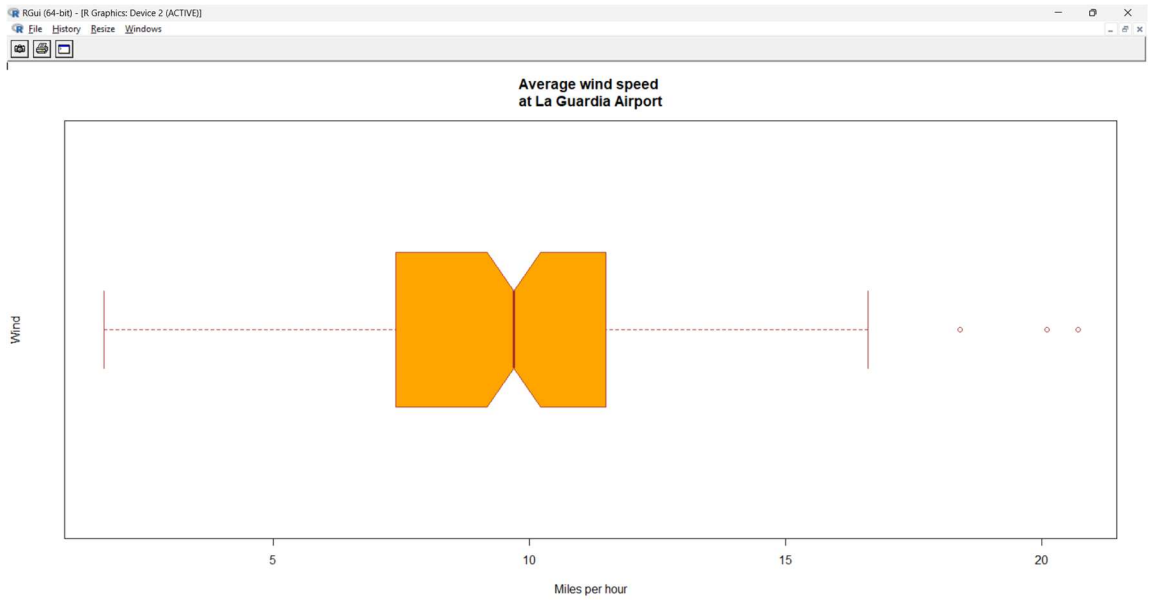
```
# Adding Titles and Labeling Axes to Plot
cone <- function(x, y){
sqrt(x ^ 2 + y ^ 2)
}

# prepare variables.
x <- y <- seq(-1, 1, length = 30)
z <- outer(x, y, cone)
# plot the 3D surface
# Adding Titles and Labeling Axes to Plot
persp(x, y, z,
main="Perspective Plot of a Cone",
zlab = "Height",
theta = 30, phi = 15,
col = "orange", shade = 0.4)
```

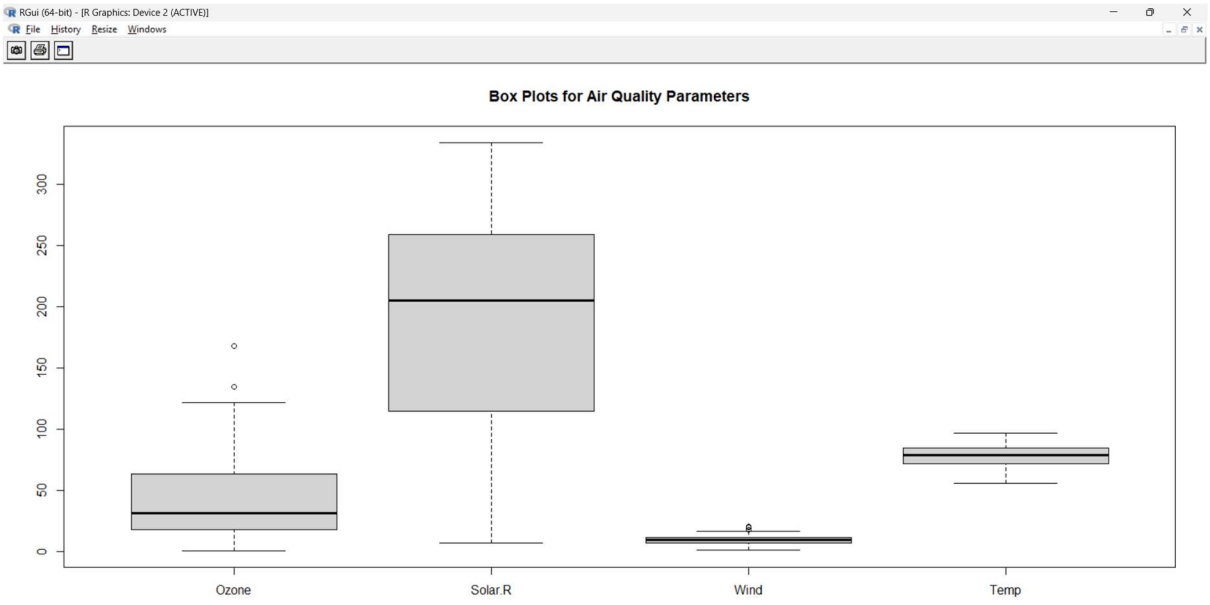
OUTPUT

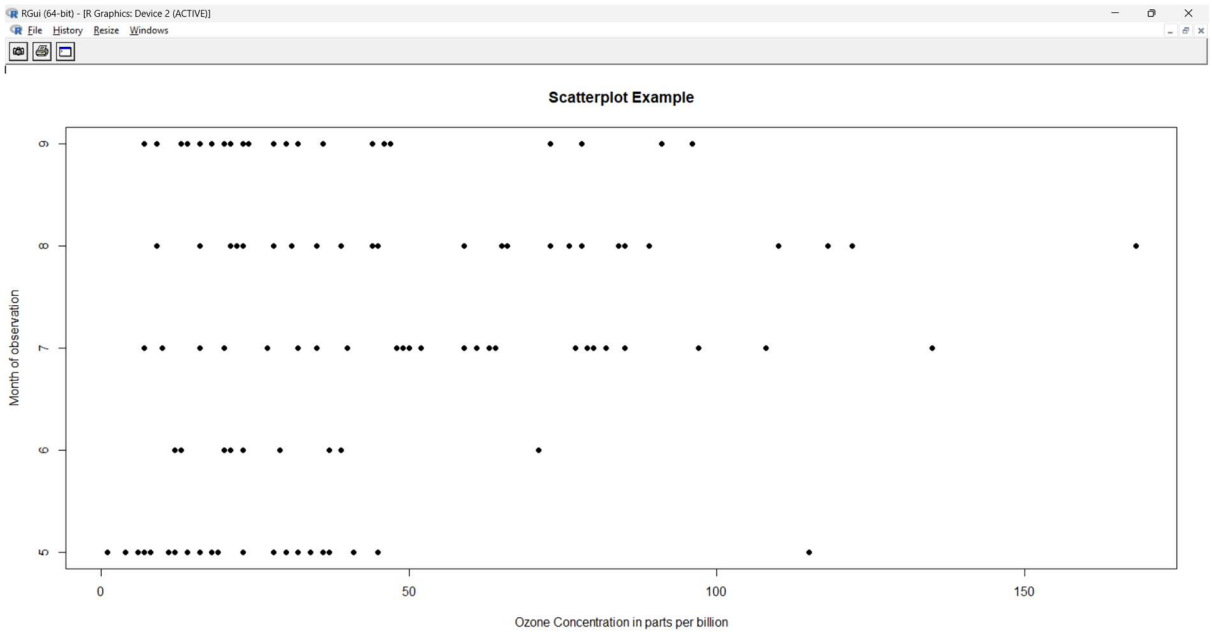


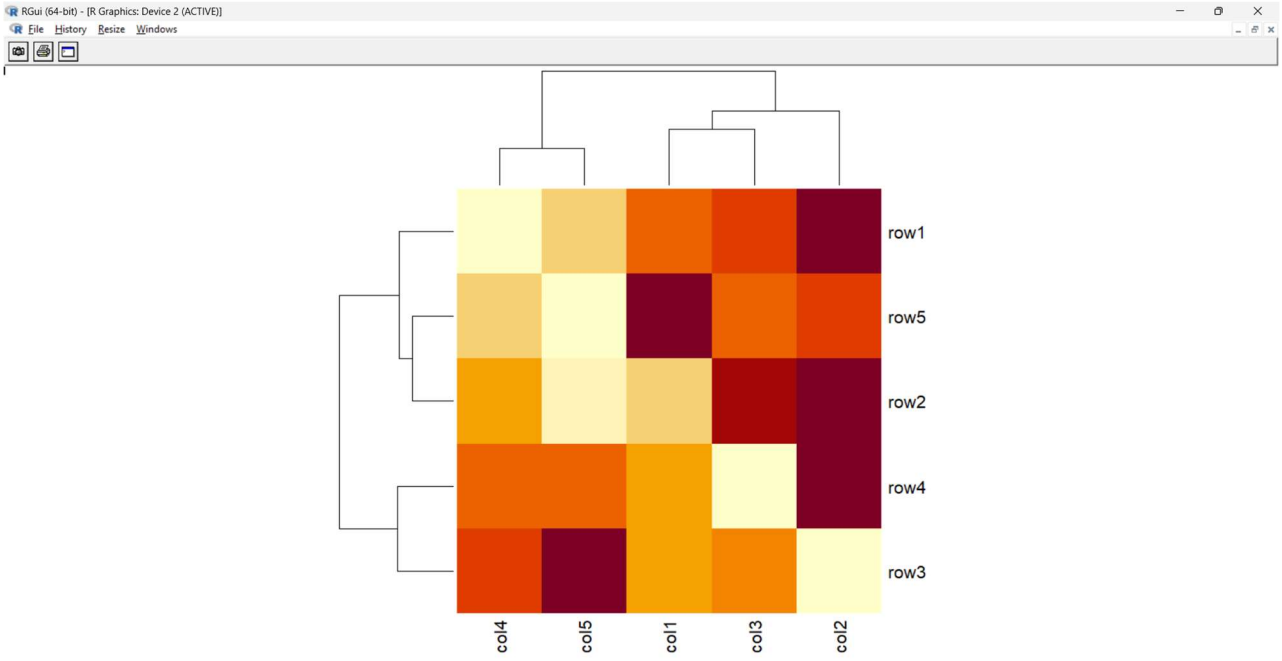

RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Ozone Concenteration in air
0
50
100
150
ozone levels

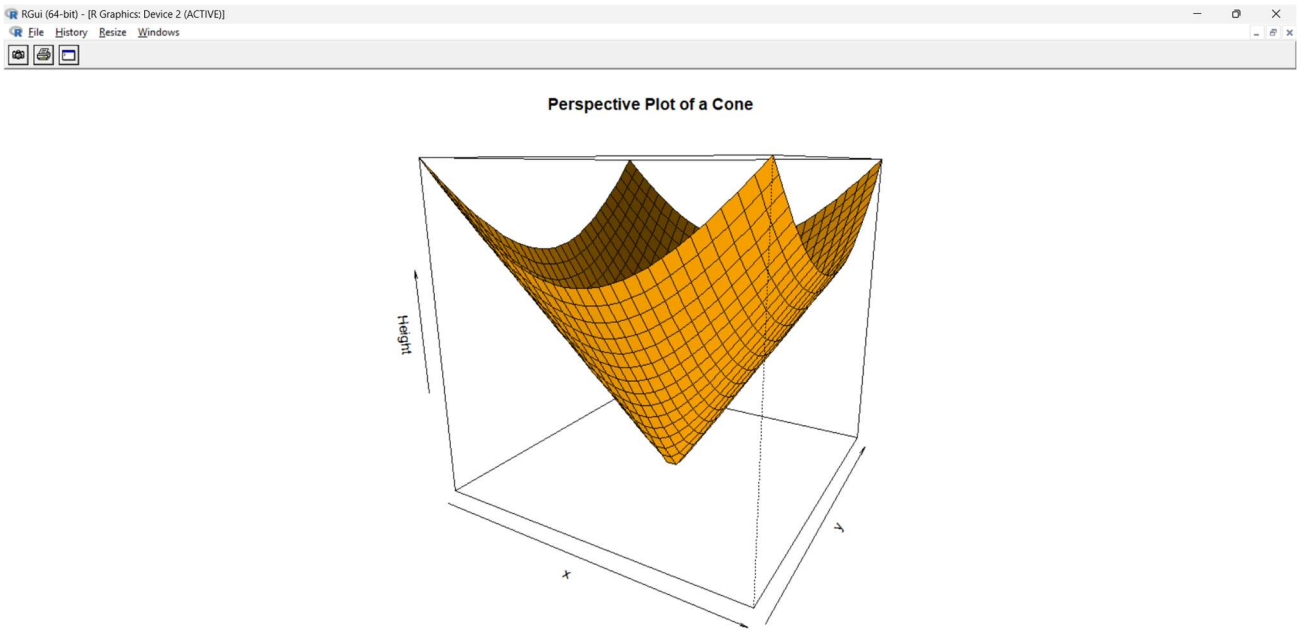

RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
La Guardia Airport's
Maximum Temperature(Daily)
Frequency
0
5
10
15
20
25
30
35
60
80
100
120
Temperature(Fahrenheit)


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Average wind speed
at La Guardia Airport
Wind
5
10
15
20
Miles per hour


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Box Plots for Air Quality Parameters
0
50
100
150
200
250
300
Ozone
Solar.R
Wind
Temp


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Scatterplot Example
Month of observation
Ozone Concentration in parts per billion
5
6
7
8
9
0
50
100
150


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
row1
row5
row2
row4
row3
col4
col5
col1
col3
col2


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
Perspective Plot of a Cone
Height
x
y