1.Implementation of Data partitioning through Hash, Range and List partitioning

Range Partitioning:- Range Partitioning is a database partitioning technique where data is divided into subsets (partitions) based on a specified range of values. Each partition contains rows whose values fall within a particular range, typically of a numeric or date-based column.

Hash Partitioning:- Hash Partitioning is a database partitioning technique where data is distributed across multiple partitions using a hash function. This method applies a hash function to a partitioning key (typically a column in the table) to determine which partition a particular row belongs to.

List Partitioning:- List Partitioning is a database partitioning technique where data is divided into partitions based on predefined, discrete sets of values for a specific column. Each partition contains rows that match a particular list of values for the partitioning key (i.e., the column being used to partition the table). The values in the list are explicitly specified by the database administrator.

Aim1 Implementation of Data partitioning through Range.

A. Create table transactions containing 3 columns as id, name and purchased. Create 5 partitions on that table.

write a query to check whether the partition has dropped.

Range Partitioning:-

CREATE database STUDENT; USE STUDENT;

CREATE TABLE tr (id INT, name VARCHAR(50), purchased DATE) PARTITION BY RANGE(YEAR(purchased)) (

PARTITION p0 VALUES LESS THAN (1990), PARTITION p1 VALUES LESS THAN (1995), PARTITION p2 VALUES LESS THAN (2000), PARTITION p3 VALUES LESS THAN (2005), PARTITION p4 VALUES LESS THAN (2010), PARTITION p5 VALUES LESS THAN (2015));

B. Insert 10 records in the transaction table.

INSERT INTO tr VALUES(1, 'desk organiser', '2003-10-15'),(2, 'alarm clock', '1997-11-05'),(3, 'chair', '2009-03-10'),(4, 'bookcase', '1989-01-10'),(5, 'exercise bike', '2014-05-09'),(6, 'sofa', '1987-06-05'),(7, 'espresso maker', '2011-11-22'),(8, 'aquarium', '1992-08-04'),(9, 'study desk', '2006-09-16'), (10, 'lava lamp', '1998-12-25');

C. Select each partition

select * from tr;

id	name	purchased
1	desk organiser	2003-10-15
2	alarm clock	1997-11-05
3	chair	2009-03-10
4	bookcase	1989-01-10
5	exercise bike	2014-05-09
6	sofa	1987-06-05
7	espresso maker	2011-11-22
8	aquarium	1992-08-04
9	study desk	2006-09-16
10	lava lamp	1998-12-25

D. **Select all partition names.**

select * from tr PARTITION (p2);

id	name	purchased
2	alarm clock	1997-11-05
10	lava lamp	1998-12-25

select * from tr PARTITION (p1);

id	name	purchased
8	aquarium	1992-08-04

select PARTITION_NAME, TABLE_ROWS from INFORMATION_SCHEMA.PARTITIONS where TABLE_NAME='tr';

PARTITION_NAME	TABLE_ROWS
p0	2
p1	1
p2	2
p3	1
p4	2
p5	2

D. Select all partition names.

select PARTITION_NAME, TABLE_ROWS from INFORMATION_SCHEMA.PARTITIONS where TABLE_NAME='tr';

	PARTITION_NAME	TABLE_ROWS
▶	p0	2
	p1	1
	p2	2
	p3	1
	p4	2
	p5	2

E. Drop a MySQL partition

ALTER table tr drop PARTITIONp0;

select PARTITION_NAME, TABLE_ROWS from INFORMATION_SCHEMA.PARTITIONS where

TABLE_NAME='tr';

	PARTITION_NAME	TABLE_ROWS
▶	p1	0
	p2	2
	p3	0
	p4	2
	p5	2

id	name	purchased
4	bookcase	1989-01-10
6	sofa	1987-06-05
8	aquarium	1992-08-04
2	alarm clock	1997-11-05
10	lava lamp	1998-12-25
1	desk organiser	2003-10-15
3	chair	2009-03-10
9	study desk	2006-09-16
5	exercise bike	2014-05-09
7	espresso maker	2011-11-22

	id	name	purchased
▶	8	aquarium	1992-08-04

Aim 2:- Implementation of Data partitioning through Hash partition.

A. Create a table employee containing 7 columns as id, fname, lname, hired, separated, job_code, store_id . Create 4 partitions on that table.

Hash Partitioning Code:

```
create database MCA; use MCA;


Create table employees( id INT NOT NULL,

fname VARCHAR(30), lname VARCHAR(30),

job_code INT, store_id INT),

PARTITION BY HASH(store_id) PARTITIONS 4;


insert into employees values(1, 'Deepak','Teja','2017-02-19','2010-2-23',1,1),

(2, 'Raheem','Khan','2020-03-25','2010-3-30',1,1),

(3, 'Rahman','Shaikh','2020-03-25','2021-5-21',4,2),
```

(4, 'Sani','Bandal','2020-03-25','2028-3-28',2,4),

(5, 'Pranita','Panchal','2020-03-25','2033-2-27',3,5),

(6, 'Yash','Gaikwad','2020-03-25','2033-8-03',3,1),

(7, 'Hiren','Jha','2020-03-25','2025-1-25',3,5);

B. Select each partition

select * from employees;

select * from employees partition(p1);

id	fname	lname	hired	seperated	job_code	store_id
1	Deepak	Teja	2017-02-19	2010-02-23	1	1
2	Raheem	Khan	2020-03-25	2010-03-30	1	1
5	Pranita	Panchal	2020-03-25	2033-02-27	3	5
6	Yash	Gaikwad	2020-03-25	2033-08-03	3	1
7	Hiren	Jha	2020-03-25	2025-01-25	3	5

id	fname	lname	hired	seperated	job_code	store_id
1	Deepak	Teja	2017-02-19	2010-02-23	1	1
2	Raheem	Khan	2020-03-25	2010-03-30	1	1
6	Yash	Gaikwad	2020-03-25	2033-08-03	3	1
3	Rahman	Shaikh	2020-03-25	2021-05-21	4	2
4	Sani	Bandal	2020-03-25	2028-03-28	2	4
5	Pranita	Panchal	2020-03-25	2033-02-27	3	5
7	Hiren	Jha	2020-03-25	2025-01-25	3	5

C. Select all partition

PARTITION BY HASH(store_id) (

partition p1, partition p2, partition p3, partition p4);

id	fname	lname	hired	seperated	job_code	store_id
4	Sani	Bandal	2020-03-25	2028-03-28	2	4

D.Drop a MYSQL Partition

Drop Table employee

Aim3 :- Implementation of Data partitioning through List partition.

A. **Create table stores with the columns cust_name, bill_no, store_id, bill_date and amount and partition the table as pEast, pWest, pNorth and pSouth with required constraints.**

Drop any of the partition

write a query to check whether the partition has droped.

List Partitioning:-

```
CREATE TABLE Stores (
cust_name VARCHAR(40), bill_no VARCHAR(20) NOT NULL,
store_id INT PRIMARY KEY NOT NULL,
bill_date DATE NOT NULL, amount DECIMAL(8,2) NOT NULL)
PARTITION BY LIST(store_id)
(PARTITION pEast VALUES IN (101, 103, 105),
PARTITION pWest VALUES IN (102, 104, 106),
PARTITION pNorth VALUES IN (107, 109, 111),
PARTITION pSouth VALUES IN (108, 110, 112));
```

B.Insert 10 records

```
insert into Stores values('Suresh','B101',101,'2023-10-29', 1900.00);
insert into Stores values('Ali','B102',102,'2024-08-15', 1800.00);
insert into Stores values('Qunut','B103',103,'2022-10-11', 1950.00);
insert into Stores values('Arshnaz','B104',104,'2019-11-10', 18000.00);
insert into Stores values('Habib','B105',105,'2019-12-01', 5800.00);
insert into Stores values('Pratik','B106',106,'2020-07-2', 6548.50);
insert into Stores values('Neha','B107',107,'2022-04-6', 19874.00);
insert into Stores values('Sheetal','B108',108,'2021-03-18', 69253.73);
insert into Stores values('Gaurav','B109',109,'2021-02-27', 15000.00);
insert into Stores values('Balasaheb','B110',110,'2019-01-20', 19000.00);
insert into Stores values('Sudha','B111',112,'2020-05-25', 30000.00);
select * from Stores;
```

cust_name	bill_no	store_id	bill_date	amount
Suresh	B101	101	2023-10-29	1900.00
Ali	B102	102	2024-08-15	1800.00
Qunut	B103	103	2022-10-11	1950.00
Arshnaz	B104	104	2019-11-10	18000.00
Habib	B105	105	2019-12-01	5800.00
Pratik	B106	106	2020-07-02	6548.50
Neha	B107	107	2022-04-06	19874.00
Sheetal	B108	108	2021-03-18	69253.73
Gaurav	B109	109	2021-02-27	15000.00
Balasaheb	B110	110	2019-01-20	19000.00
Sudha	B111	112	2020-05-25	30000.00
NULL	NULL	NULL	NULL	NULL

C. Select each partition.

select * from Stores partition(pNorth);

cust_name	bill_no	store_id	bill_date	amount
Neha	B107	107	2022-04-06	19874.00
Gaurav	B109	109	2021-02-27	15000.00
NULL	NULL	NULL	NULL	NULL

D. **Drop any of the partition**

select * from Stores partition(pNorth);

Output:-

1) select * from Stores;

cust_name	bill_no	store_id	bill_date	amount
Suresh	B101	101	2023-10-29	1900.00
Qunut	B103	103	2022-10-11	1950.00
Habib	B105	105	2019-12-01	5800.00
Ali	B102	102	2024-08-15	1800.00
Arshnaz	B104	104	2019-11-10	18000.00
Pratik	B106	106	2020-07-02	6548.50
Neha	B107	107	2022-04-06	19874.00
Gaurav	B109	109	2021-02-27	15000.00
Sheetal	B108	108	2021-03-18	69253.73
Balasaheb	B110	110	2019-01-20	19000.00
Sudha	B111	112	2020-05-25	30000.00
NULL	NULL	NULL	NULL	NULL

2. Implementation of Analytical queries like Roll_UP, CUBE, First, Last, Lead, Lag, Rank AND Dense_rank and Windowing functions preceding rows, following and rows between

Roll-Up:- Roll-Up is a data aggregation operation used in databases, data warehouses, and OLAP (Online Analytical Processing) systems to summarize or consolidate detailed data into higher-level summaries. It typically involves moving from a lower level of granularity (detailed data) to a higher level of granularity (summary data), often by performing aggregation operations like **SUM**, **COUNT**, **AVG**, or other statistical functions on groups of records.

CUBE:- CUBE is an operation used in multidimensional databases, OLAP (Online Analytical Processing) systems, and data warehouses to generate **all possible aggregations** of a set of dimensions. It is an extension of the **GROUP BY** clause (often seen in SQL) and provides a way to compute summaries for all combinations of dimensions. The CUBE operation helps generate a multidimensional view of the data, allowing analysts to perform more complex and comprehensive data analysis.

Rank:- Rank refers to an operation that assigns a ranking or ordinal position to each row in a result set based on a specified ordering of the rows. It is often used in scenarios where you need to identify the relative position of each row within a group or across the entire dataset, based on some criteria (such as values in a particular column).

DENSE_RANK():- Dense Rank is a ranking function in SQL and analytical databases that assigns a unique rank to each row in a result set based on a specified ordering, but **does not leave gaps in the ranking** when there are ties. In other words, if two rows are tied for a particular rank, they will share the same rank, and the next row will receive the **immediately following rank** (without skipping any numbers).

window functions:- to define the range of rows that should be included in the calculation for each row in the result set. Specifically, it refers to the rows **preceding the current row** (i.e., rows that come before the current row in the ordered result set).

Aim1: Implementation of Analytical query using ROLLUP.

A. Create table payment with 4 columns as customer_id, sales_amount, payment_mode and payment_date.

Code:

#Create Table and Insert Values

```
create table payment(

customer_id number,

sales_amount number,

payment_mode varchar2(20),

payment_date Date );
```

B. Insert 15 records into it

```
insert into payment values(1,60,'Cash', '24-SEP-2020');

insert into payment values(2,30,'Credit Card', '27-APR-2020');
```

insert into payment values(3,90,’Credit Card’, '7-JUL-2020');

insert into payment values(4,50,’Debit Card’, '12-FEB-2020');

insert into payment values(5,40,’Mobile Payment’, '20-NOV-2020');

insert into payment values(6,40,’Debit Card’,'28-JUN-2021');

insert into payment values(7,10,’Cash’,'25-AUG-2021');

insert into payment values(8,30,’Mobile Payment’,'17-JUN-2021');

insert into payment values(9,80,’Cash’,'25-AUG-2021');

insert into payment values(10,50,’Mobile Payment’,'3-NOV-2021');

insert into payment values(11,70,’Cash’,'1-NOV-2022');

insert into payment values(12,60,’Netbanking’,'11-SEP-2022');

insert into payment values(13,30,’Netbanking’,'10-DEC-2022');

insert into payment values(14,50,’Credit Card’,'14-MAY-2022');

insert into payment values(15,70,’Credit Card’,'25-SEP-2022');

c. **Write an Analytical using Rollup on the columns sales_id and sales_amount.**

#ROLLUP

SELECT customer_id,sales_amount, SUM(sales_amount) AS Totalsal from payment GROUP BY ROLLUP(customer_id,sales_amount);

```
SQL> SELECT customer_id,sales_amount, SUM(sales_amount) AS Totalsal from payment GROUP BY ROLLUP(customer_id,sales_amount);

CUSTOMER_ID SALES_AMOUNT   TOTALSAL
----------- ------------ ----------
          1           60         60
          1                      60
          2           30         30
          2                      30
          3           90         90
          3                      90
          4           50         50
          4                      50
          5           40         40
          5                      40
          6           40         40
          6                      40
          7           10         10
          7                      10
          8           30         30
          8                      30
          9           80         80
          9                      80
         10           50         50
         10                      50
         11           70         70
         11                      70
         12           60         60
         12                      60
         13           30         30
         13                      30
         14           50         50
         14                      50
         15           70         70
         15                      70
                                760

31 rows selected.

SQL>
```

2. Implementation of Analytical query using RANK() and DENSE_RANK.

select customer_id, sales_amount,

ROW_NUMBER() OVER (ORDER BY customer_id) as "ROW_NUMBER",

RANK() OVER(ORDER BY customer_id) as "RANK",

DENSE_RANK() OVER(ORDER BY customer_id) as "DENSE_RANK"

FROM payment;

```
SQL> select customer_id, amount,
  2  ROW_NUMBER() OVER (ORDER BY customer_id) as "ROW_NUMBER",
  3  RANK() OVER(ORDER BY customer_id) as "RANK",
  4  DENSE_RANK() OVER(ORDER BY customer_id) as "DENSE_RANK",
  5  PERCENT_RANK() OVER(ORDER BY customer_id) as "PERCENT_RANK"
  6  FROM payment;

CUSTOMER_ID     AMOUNT ROW_NUMBER       RANK DENSE_RANK PERCENT_RANK
----------- ---------- ---------- ---------- ---------- ------------
          1         60          1          1          1            0
          2         30          2          2          2   .071428571
          3         90          3          3          3   .142857143
          4         50          4          4          4   .214285714
          5         40          5          5          5   .285714286
          6         40          6          6          6   .357142857
          7         10          7          7          7   .428571429
          8         30          8          8          8           .5
          9         80          9          9          9   .571428571
         10         50         10         10         10   .642857143
         11         70         11         11         11   .714285714
         12         60         12         12         12   .785714286
         13         30         13         13         13   .857142857
         14         50         14         14         14   .928571429
         15         70         15         15         15            1

15 rows selected.

SQL>
```

3. FIRST and LAST.

select customer_id, sales_amount,

ROW_NUMBER() OVER (ORDER BY sales_amount) as "ROW_NUMBER",

RANK() OVER(ORDER BY sales_amount) as "RANK",

DENSE_RANK() OVER(ORDER BY sales_amount) as "DENSE_RANK"

FROM payment;

```
SQL> select customer_id, sales_amount,
  2  ROW_NUMBER() OVER (ORDER BY sales_amount) as "ROW_NUMBER",
  3  RANK() OVER(ORDER BY sales_amount) as "RANK",
  4  DENSE_RANK() OVER(ORDER BY sales_amount) as "DENSE_RANK"
  5  FROM payment;

CUSTOMER_ID SALES_AMOUNT ROW_NUMBER       RANK DENSE_RANK
----------- ------------ ---------- ---------- ----------
          7           10          1          1          1
          2           30          2          2          2
         13           30          3          2          2
          8           30          4          2          2
          5           40          5          5          3
          6           40          6          5          3
         10           50          7          7          4
          4           50          8          7          4
         14           50          9          7          4
         12           60         10         10          5
          1           60         11         10          5
         15           70         12         12          6
         11           70         13         12          6
          9           80         14         14          7
          3           90         15         15          8

15 rows selected.

SQL>
```

4 & 5.LEAD and LAG

Replace OVER clauses with MySQL's equivalent:

SELECT REGION, PRODUCT, YEAR, SALES_AMOUNT,LEAD(SALES_AMOUNT) OVER(PARTITION BY

REGION ORDER BY YEAR)

AS NEXT_YEAR_SALES,LAG(SALES_AMOUNT)

OVER (PARTITION BY

REGION ORDER BY YEAR) AS PREV_YEAR_SALES FROM SALES;

```
17 rows in set (0.00 sec)

mysql> SELECT REGION, PRODUCT, YEAR, SALES_AMOUNT,LEAD(SALES_AMOUNT) OVER (PARTITION BY REGION ORDER BY YEAR) AS NEXT_YE
AR_SALES,LAG(SALES_AMOUNT) OVER (PARTITION BY REGION ORDER BY YEAR) AS PREV_YEAR_SALES FROM SALES;
+--------+---------+------+--------------+-----------------+-----------------+
| REGION | PRODUCT | YEAR | SALES_AMOUNT | NEXT_YEAR_SALES | PREV_YEAR_SALES |
+--------+---------+------+--------------+-----------------+-----------------+
| East   | Tablet  | 2022 |     20000.00 |        30000.00 |            NULL |
| East   | Tablet  | 2023 |     30000.00 |            NULL |        20000.00 |
| North  | Laptop  | 2022 |     40000.00 |        50000.00 |            NULL |
| North  | Laptop  | 2023 |     50000.00 |            NULL |        40000.00 |
| South  | Mobile  | 2022 |     45000.00 |        60000.00 |            NULL |
| South  | Mobile  | 2023 |     60000.00 |            NULL |        45000.00 |
| West   | Laptop  | 2022 |     60000.00 |        70000.00 |            NULL |
| West   | Laptop  | 2023 |     70000.00 |            NULL |        60000.00 |
+--------+---------+------+--------------+-----------------+-----------------+
8 rows in set (0.02 sec)

mysql>
```

6.Windowing Functions

SELECT REGION, PRODUCT, YEAR, SALES_AMOUNT,SUM(SALES_AMOUNT) OVER (PARTITION BY REGION ORDER BY YEAR ROWS BETWEEN 1 PRECEDING AND

CURRENT ROW) AS RUNNING_TOTAL FROM SALES;

```
mysql> SELECT REGION, PRODUCT, YEAR, SALES_AMOUNT,SUM(SALES_AMOUNT) OVER (PARTITION BY REGION ORDER BY YEAR ROWS BETWEEN
 1 PRECEDING AND CURRENT ROW) AS RUNNING_TOTAL FROM SALES;
+--------+---------+------+--------------+---------------+
| REGION | PRODUCT | YEAR | SALES_AMOUNT | RUNNING_TOTAL |
+--------+---------+------+--------------+---------------+
| East   | Tablet  | 2022 |     20000.00 |      20000.00 |
| East   | Tablet  | 2023 |     30000.00 |      50000.00 |
| North  | Laptop  | 2022 |     40000.00 |      40000.00 |
| North  | Laptop  | 2023 |     50000.00 |      90000.00 |
| South  | Mobile  | 2022 |     45000.00 |      45000.00 |
| South  | Mobile  | 2023 |     60000.00 |     105000.00 |
| West   | Laptop  | 2022 |     60000.00 |      60000.00 |
| West   | Laptop  | 2023 |     70000.00 |     130000.00 |
+--------+---------+------+--------------+---------------+
8 rows in set (0.00 sec)

mysql>
```

3. Implementation of ORDBMS concepts like ADT (Abstract Data Types), Object table and Inheritance.

An **Abstract Data Type (ADT)** is a mathematical model or a conceptual framework for data structures, defined by a set of operations that can be performed on them and the rules that govern those operations, **independently of any implementation details**.

An **object table** is a table in a relational database that stores **objects** or **instances of object types**, typically used in **Object-Oriented Databases (OODBs)** or in **Object-Relational Mapping (ORM)** systems. The concept of object tables blends the features of **relational databases** (which organize data into tables) and **object-oriented programming** (where data is organized as objects).

Inheritance is a fundamental concept in **object-oriented programming (OOP)** that allows one class to inherit the properties and behaviors (fields and methods) of another class. It promotes **code reuse** and establishes a relationship between the **parent class** (also called **superclass** or **base class**) and the **child class** (also called **subclass** or **derived class**).

User-Defined Types/ADT

1. Example-1

A. Create type person_type having attributes name, phone and a member function get_areacode().
B. Create table person_table of person_type.
C. Insert values in the table.
D. Display the content of the table.
E. Display records applying various conditions.
F. Drop type person_type.
G. Drop table of type_person.

2. Example-2 (Nested Table)

A. Create a user defined type Items_Purchased_Typ with attributes Item_ID, Quantity, Price to model the items that are purchased by a customer in a supermarket.
B. Create a table type Items_Purchased_Tab as a table of the type Items_Purchased_Typ
C. Created a table Bills to store the information regarding purchase of all customers' bills. Bills table has the attribute Billed_Items as nested table type.
D. Insert the data into Bills table
E. Display records from Bills table

3. Example-3 (Oracle Type Inheritance)

A. Create type person_type with the attributes ssn, name, address.
B. Create type student_type under person_type.
C. Create type employee_type under person_type.
D. Create table persons of person_type.
E. Insert values in the persons table.
F. select data from various tables.
G. drop all types and tables.

Code:-

Step 1: Create an Abstract Data Type (ADT)

```
CREATE OR REPLACE TYPE PersonType AS OBJECT (

person_id NUMBER,     name VARCHAR2(50),
```

dob DATE,

MEMBER FUNCTION age RETURN NUMBER) NOT FINAL;

CREATE OR REPLACE TYPE BODY PersonType AS

MEMBER FUNCTION age RETURN NUMBER IS

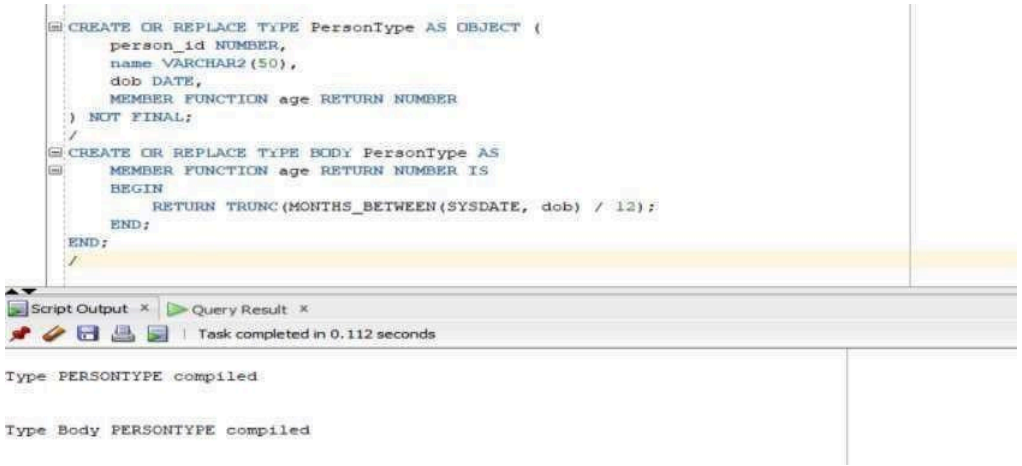
BEGIN

RETURN TRUNC(MONTHS_BETWEEN(SYSDATE, dob) / 12);

END;

END;

Output:-



Step 2: Create a Subtype for Inheritance

-- Define a subtype inheriting from PersonType

CREATE OR REPLACE TYPE EmployeeType UNDER

PersonType (employee_id NUMBER, department VARCHAR2(50), salary NUMBER);

Step 3: Create an Object Table:

CREATE TABLE PersonTable OF PersonType (person_id PRIMARY KEY);

-- Create a table for EmployeeType CREATE TABLE EmployeeTable OF EmployeeType

(person_idPRIMARY KEY) OBJECT IDENTIFIER IS PRIMARY KEY;

Output:-

Step 4: Insert Data into Object Tables

-- Insert data into PersonTable

INSERT INTO PersonTable VALUES (

PersonType(1, 'Alice', TO_DATE('1990-05-15', 'YYYY-MM-DD')));

Output:-

#Insert data into EmployeeTable

INSERT INTO EmployeeTable VALUES (

EmployeeType(2, 'Bob', TO_DATE('1985-10-20', 'YYYY-MM-DD'), 101, 'IT', 75000));

```
INSERT INTO EmployeeTable
VALUES (EmployeeType(7, 'Fiona', TO_DATE('1993-01-30', 'YYYY-MM-DD'), 103, 'Finance', 60000));

INSERT INTO EmployeeTable
VALUES (EmployeeType(8, 'George', TO_DATE('1990-09-12', 'YYYY-MM-DD'), 104, 'Marketing', 55000));

INSERT INTO EmployeeTable
VALUES (EmployeeType(9, 'Helen', TO_DATE('1987-03-18', 'YYYY-MM-DD'), 105, 'Operations', 65000));

INSERT INTO EmployeeTable
VALUES (EmployeeType(10, 'Ian', TO_DATE('1984-11-22', 'YYYY-MM-DD'), 106, 'IT', 80000));
```

Script Output | Query Result

Task completed in 0.154 seconds

row inserted.

row inserted.

```
select * from employeetable;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.01 seconds

	PERSON_ID	NAME	DOB	EMPLOYEE_ID	DEPARTMENT	SALARY
1	2	Bob	20-10-85	101	IT	75000
2	7	Fiona	30-01-93	103	Finance	60000
3	8	George	12-09-90	104	Marketing	55000
4	9	Helen	18-03-87	105	Operations	65000
5	10	Ian	22-11-84	106	IT	80000

Step 5: Query Data with Advanced Features Query for PersonTable

SELECT p.name, p.age() AS age FROM PersonTable p;

```
SELECT person_id, name, dob, p.age() AS age
FROM PersonTable p;
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.085 seconds

	PERSON_ID	NAME	DOB	AGE
1	1	Alice	15-05-90	34
2	4	shreya	25-08-95	29
3	5	srushti	15-12-92	31
4	6	samruddhi	05-07-88	36

#Query EmployeeTable with inheritance

SELECT e.name, e.age() AS age, e.department, e.salary FROM EmployeeTable e;

```
SELECT e.name, e.age() AS age, e.department, e.salary
FROM EmployeeTable e;
```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0.033 seconds

	NAME	AGE	DEPARTMENT	SALARY
1	Bob	39	IT	75000

#**Query using object-relational relationships**

SELECT e.person_id, e.name, e.dob, e.employee_id, e.department, e.salary FROM EmployeeTable e;

```
SELECT
    e.person_id,
    e.name,
    e.dob,
    e.employee_id,
    e.department,
    e.salary
FROM EmployeeTable e;
```

Script Output | Query Result | SQL | All Rows Fetched: 5 in 0.045 seconds

	PERSON_ID	NAME	DOB	EMPLOYEE_ID	DEPARTMENT	SALARY
1	2	Bob	20-10-85	101	IT	75000
2	7	Fiona	30-01-93	103	Finance	60000
3	8	George	12-09-90	104	Marketing	55000
4	9	Helen	18-03-87	105	Operations	65000
5	10	Ian	22-11-84	106	IT	80000

4. Implementation of ETL transformation with Pentaho like Copy data from Source (Table/Excel/ Oracle) and store it to Target (Table / Excel/ Oracle), Adding sequence, Adding Calculator, Concatenation of two fields, splitting of two fields, Number Range, String Operations, Sorting data, Implement the merge join transformation on tables, Implement data validations on the table data.

ETL stands for **Extract, Transform, Load**, which is a process used in data integration and data warehousing to move and manipulate data from different sources to a central repository or data warehouse.

The ETL process involves three main steps:

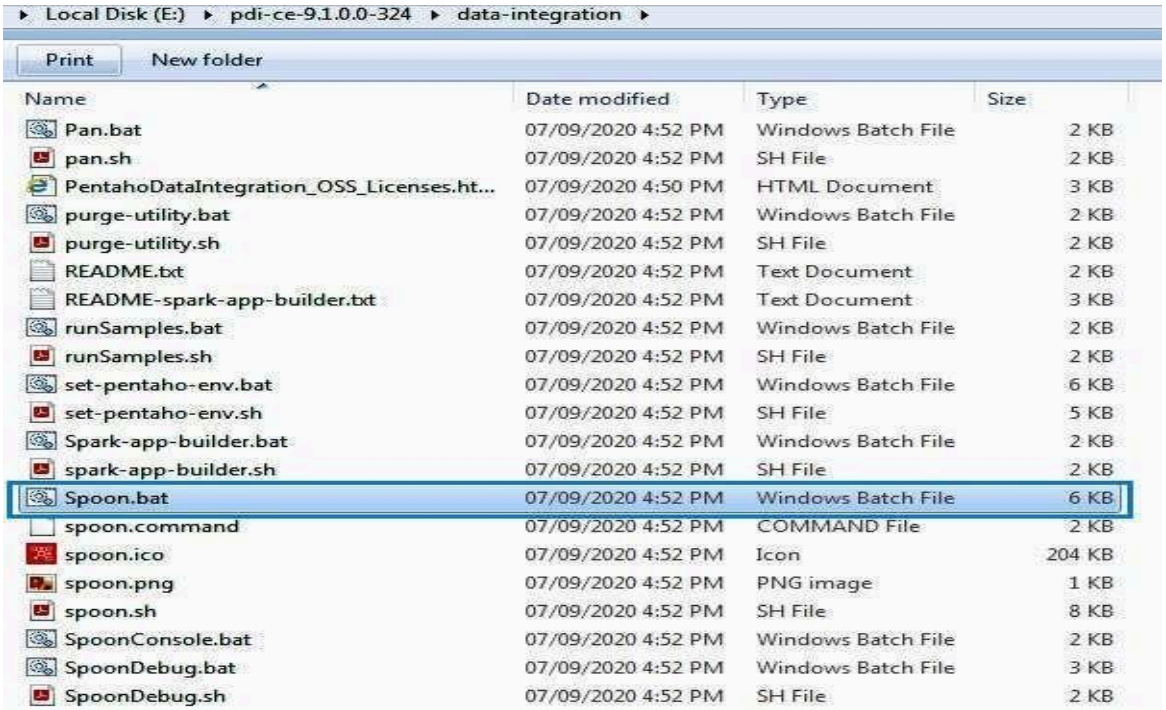
1.Extract:- The **Extract** phase involves retrieving data from various source systems. These sources could be databases, flat files, APIs, web services, or other external systems.

2.Transform:- In the **Transform** phase, the extracted data is cleaned, processed, and transformed into a format that is suitable for the destination database or data warehouse.

3.Load:- The **Load** phase is where the transformed data is loaded into the target system, which could be a **data warehouse**, **data lake**, or another storage system designed for reporting and analytics.

Pentaho Data Integration Software

The file that runs the app is called "Spoon.bat".

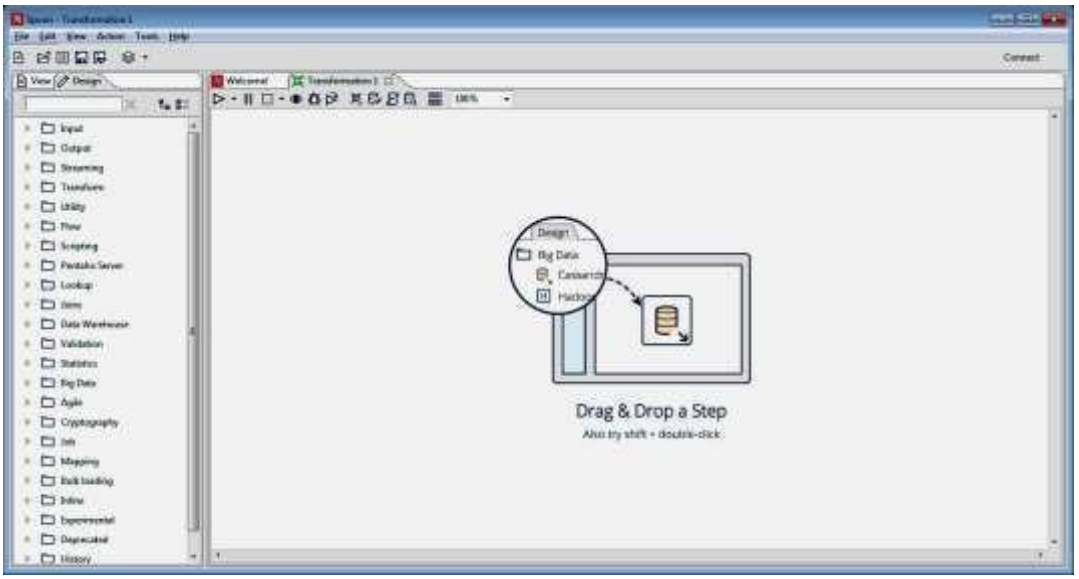


Double click this file to open the Pentaho Data Integration app.

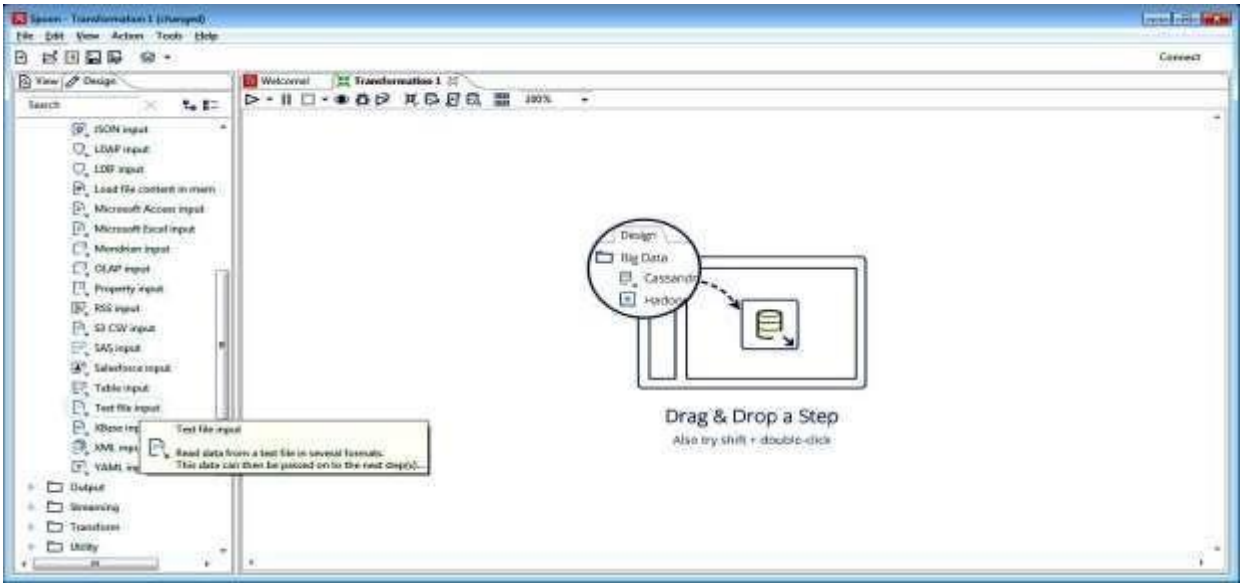
Create a new transformation:-

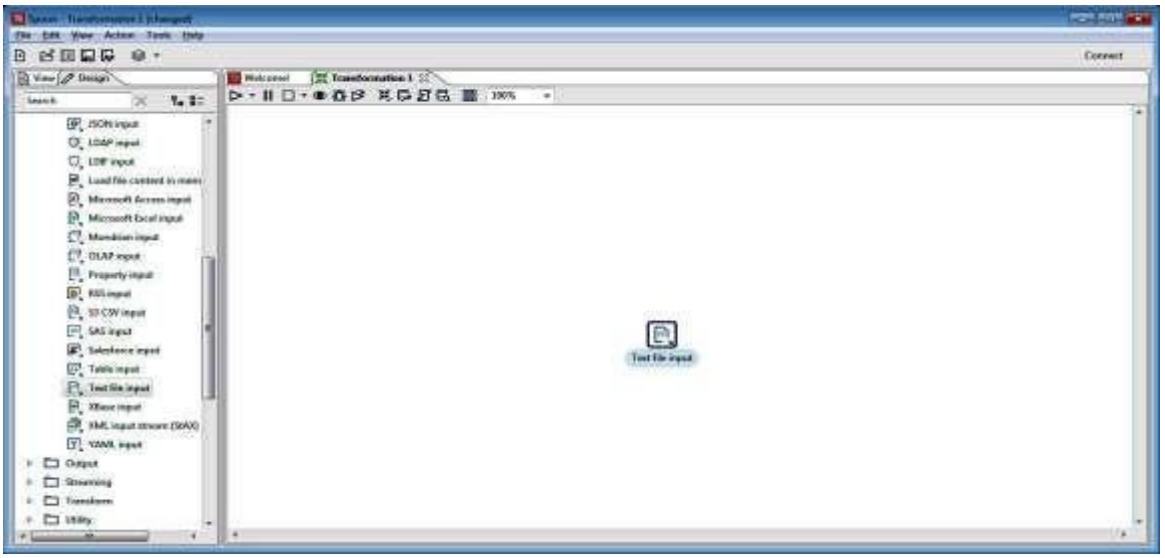
Follow these steps to create a new transformation.

1. Select File New Transformation in the upper left corner of the PDI window.

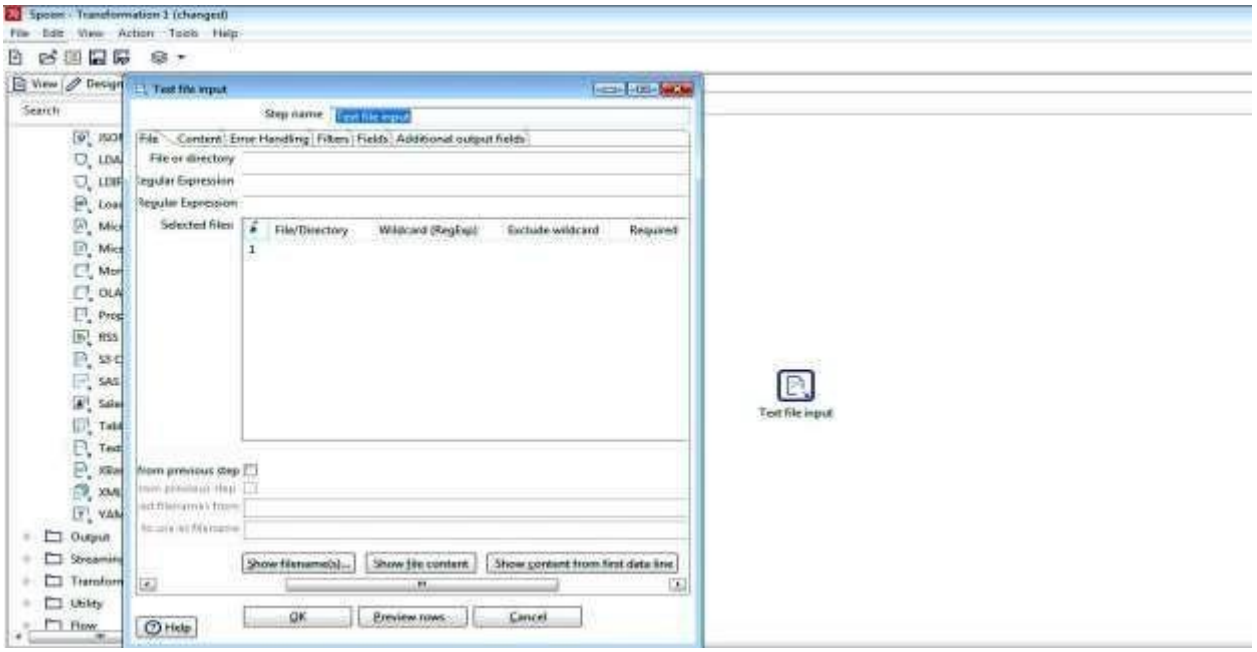


2. **Under the Design tab, expand the Input node, then select and drag a Text File Input step onto the canvas.**

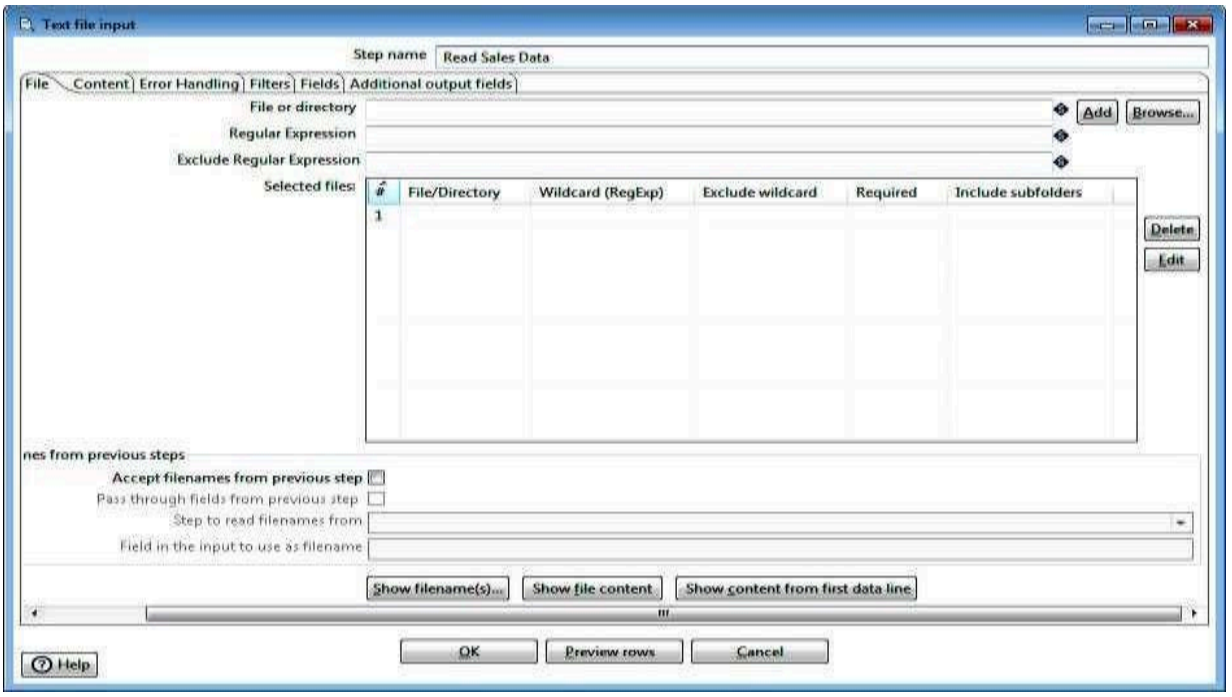




3.Double-click the Text File input step. It will open following window. you can set the step's various properties.



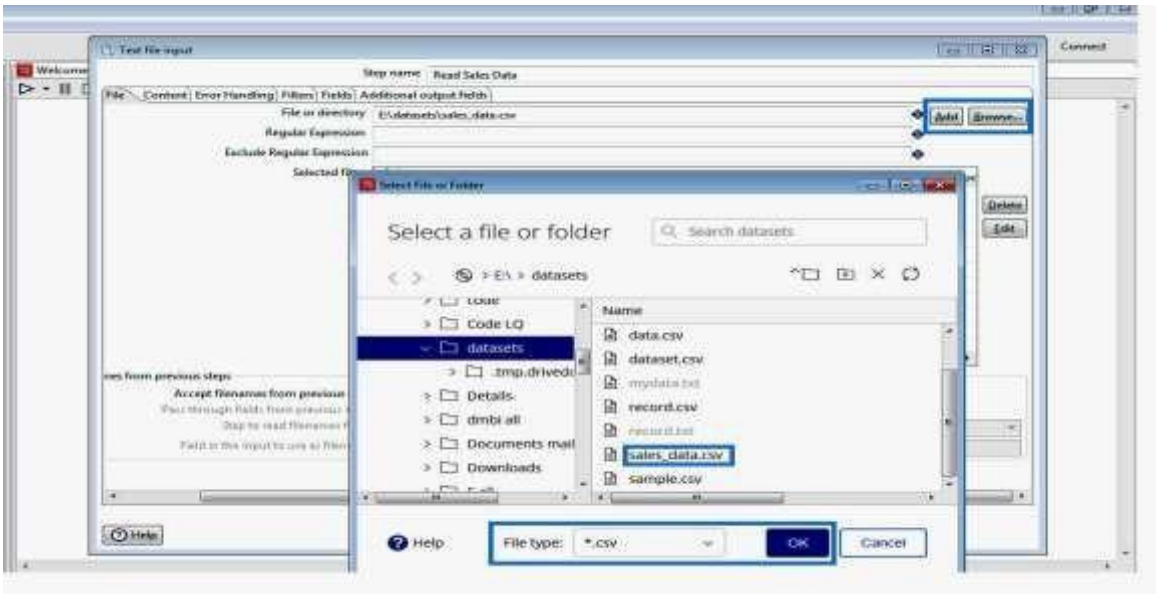
Now, In Step Name field, type Read Sales Data. The Step name: Text file input step is now renamed to Read Sales Data.



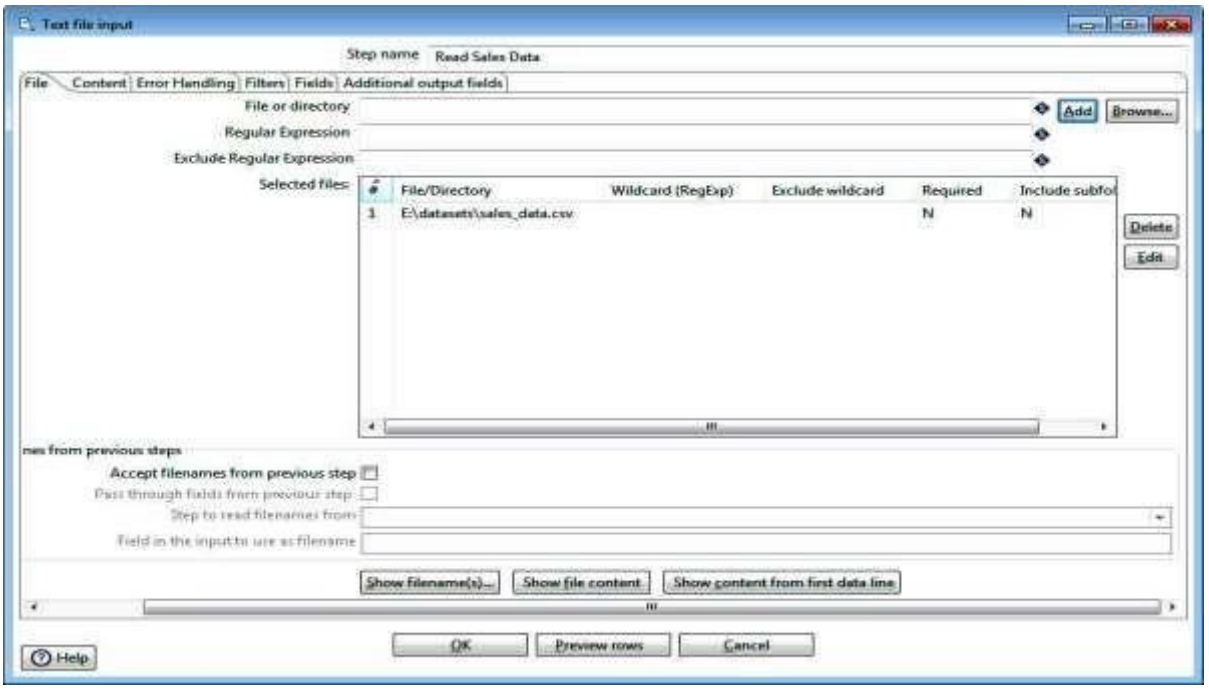
Click Browse to locate the source file, sales_data.csv, in the E:\datasets\sales_data.csv. The Browse button appears in the top right side of the window near the File or Directory field.

4.**Change File type to *.csv. Select sales_data.csv, then click OK.**

The path to the source file appears in the File or directory field.



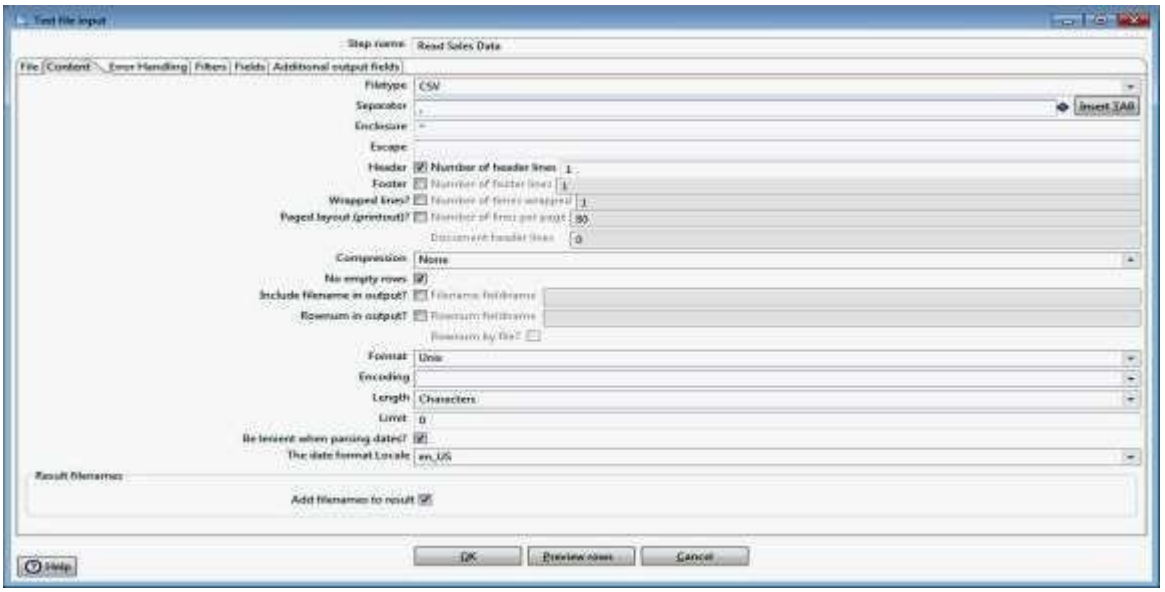
2. Click Add. The path to the file appears under Selected Files.



Edit and save the transformation:-

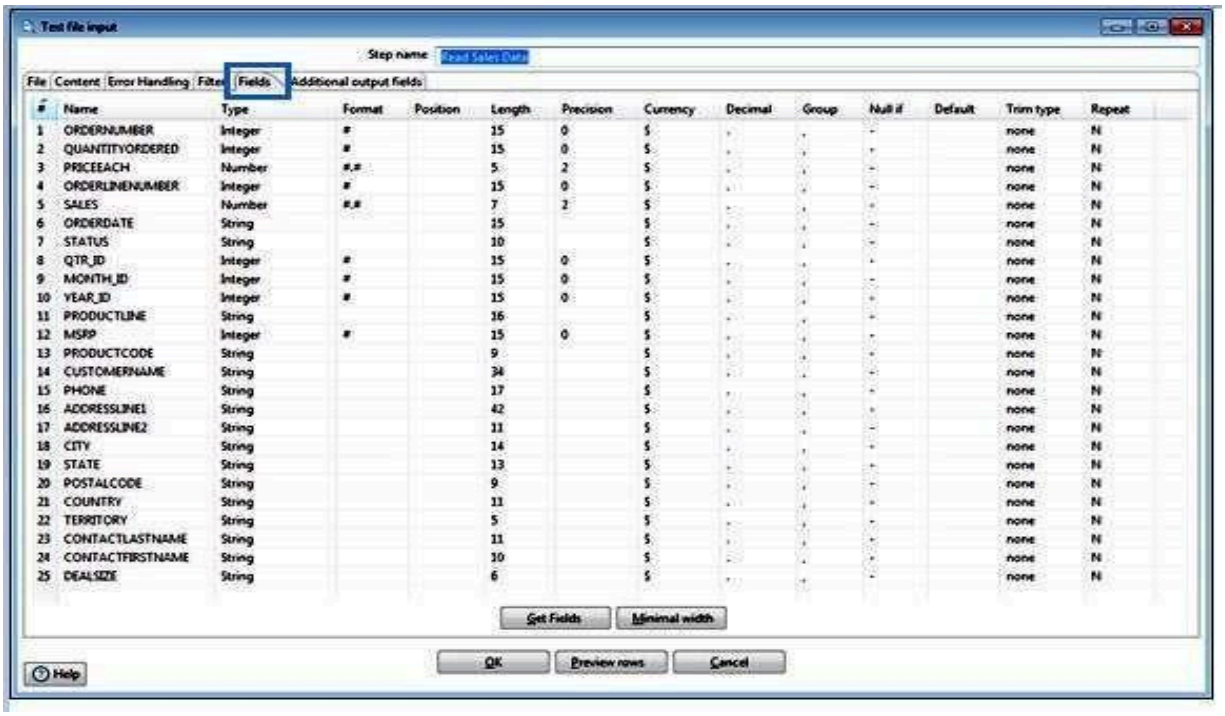
Follow these steps to provide information about the data's content.

1. **Click the Content tab. The fields under the Content tab allow you to define how your data is formatted.**
2. **Verify that the Separator is set to comma (,) and the Enclosure is set to quotation mark ("). Enable Header because there is one line of header rows in the file and set the Format field to Unix.**

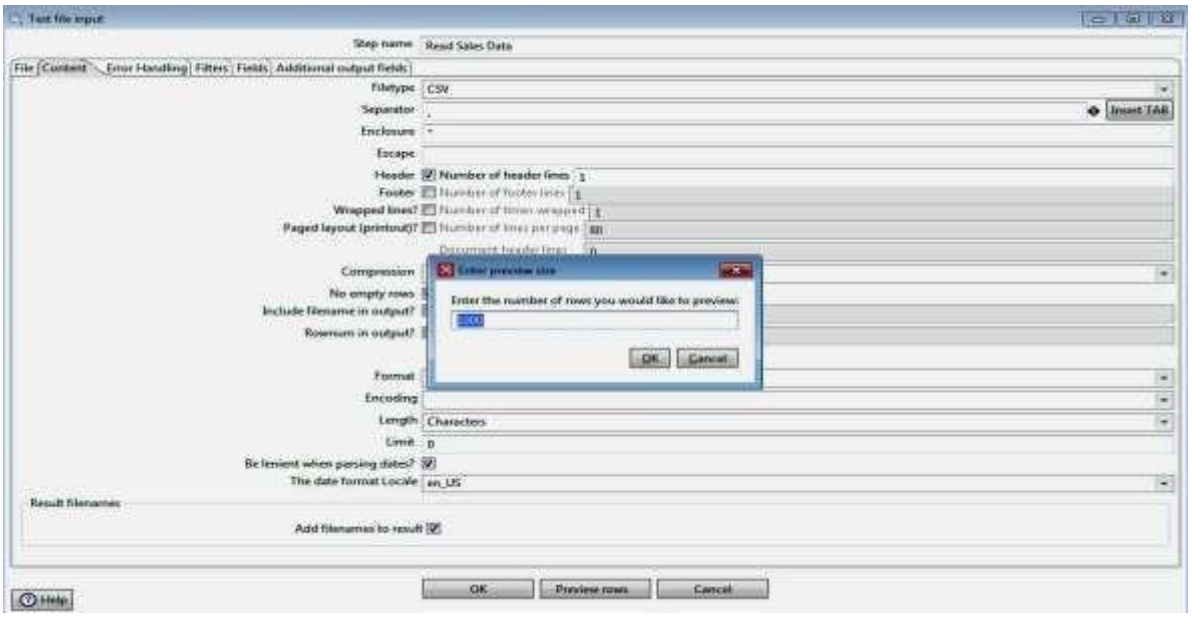


3. **Click the Fields tab and click Get Fields to retrieve the input fields from your source file.**

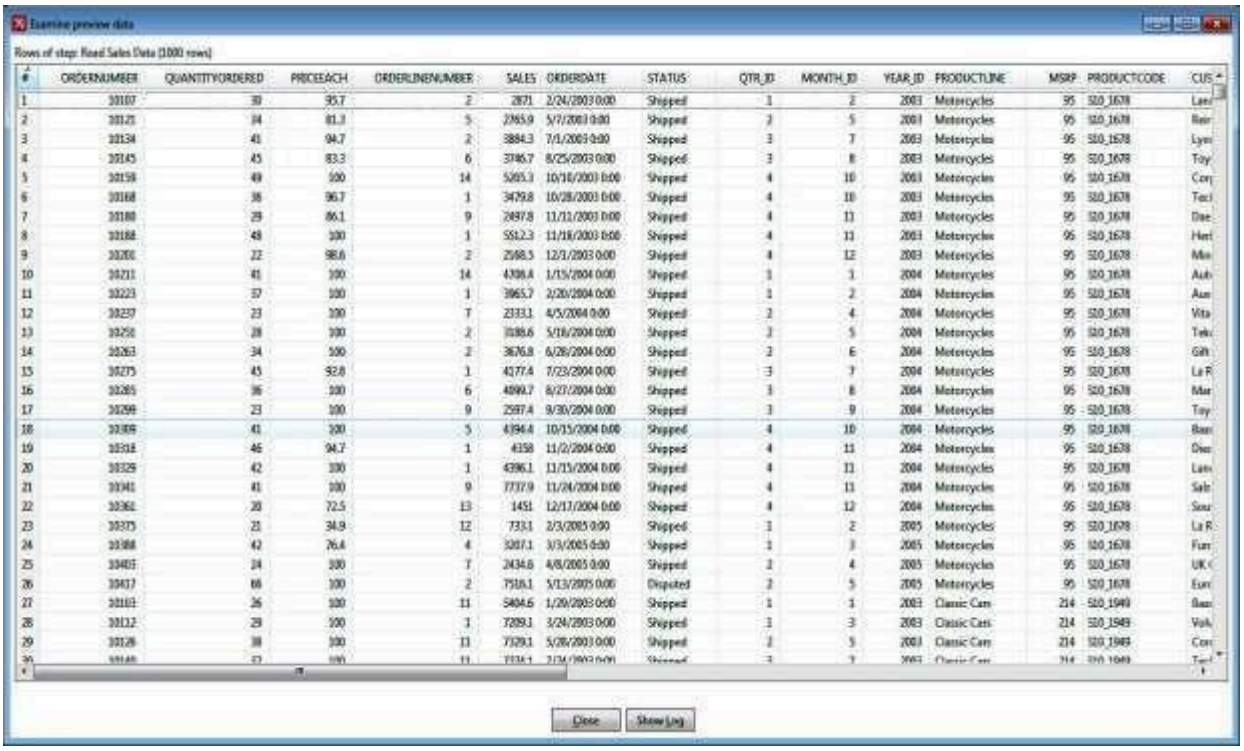
When the Number of lines to sample window appears, enter 0 in the field then click OK.



4. **If the Scan Result window displays, click Close to close the window.**
5. **To verify that the data is being read correctly, click the Content tab, then click Preview rows.**
6. **In the Enter the number of rows you would like to preview window, click OK to accept the default.**

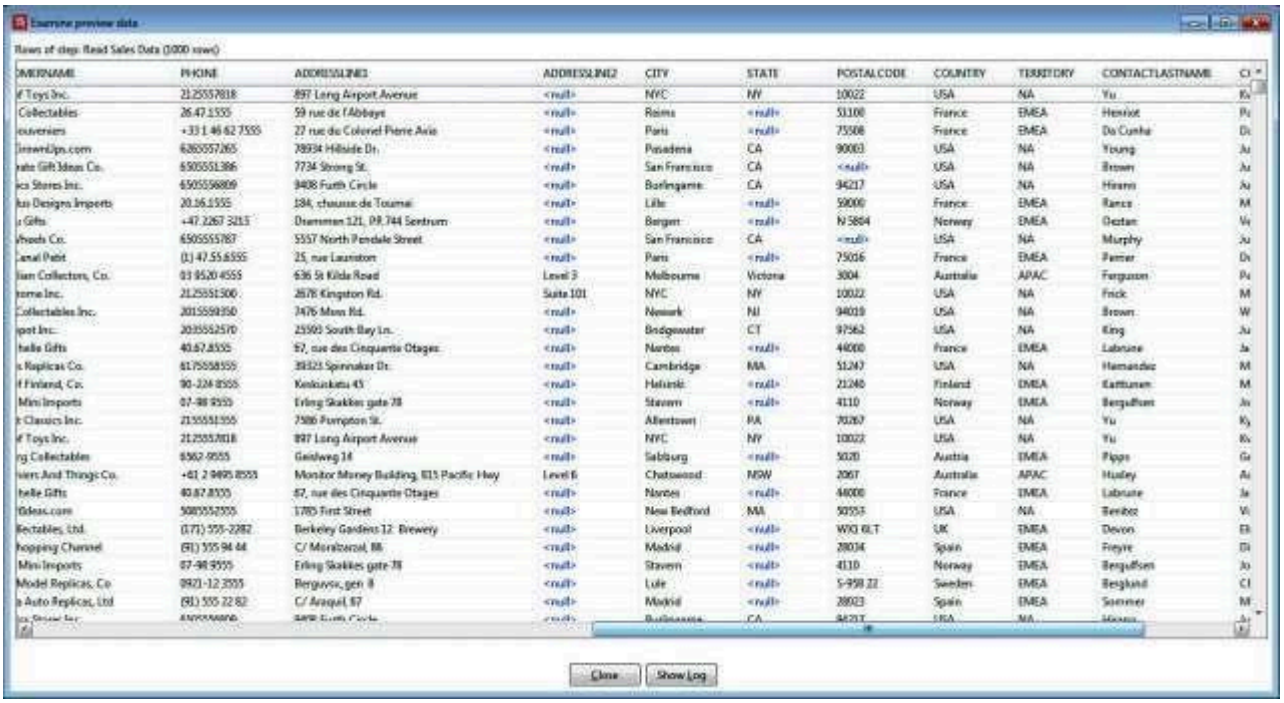


The Examine preview data window appears.

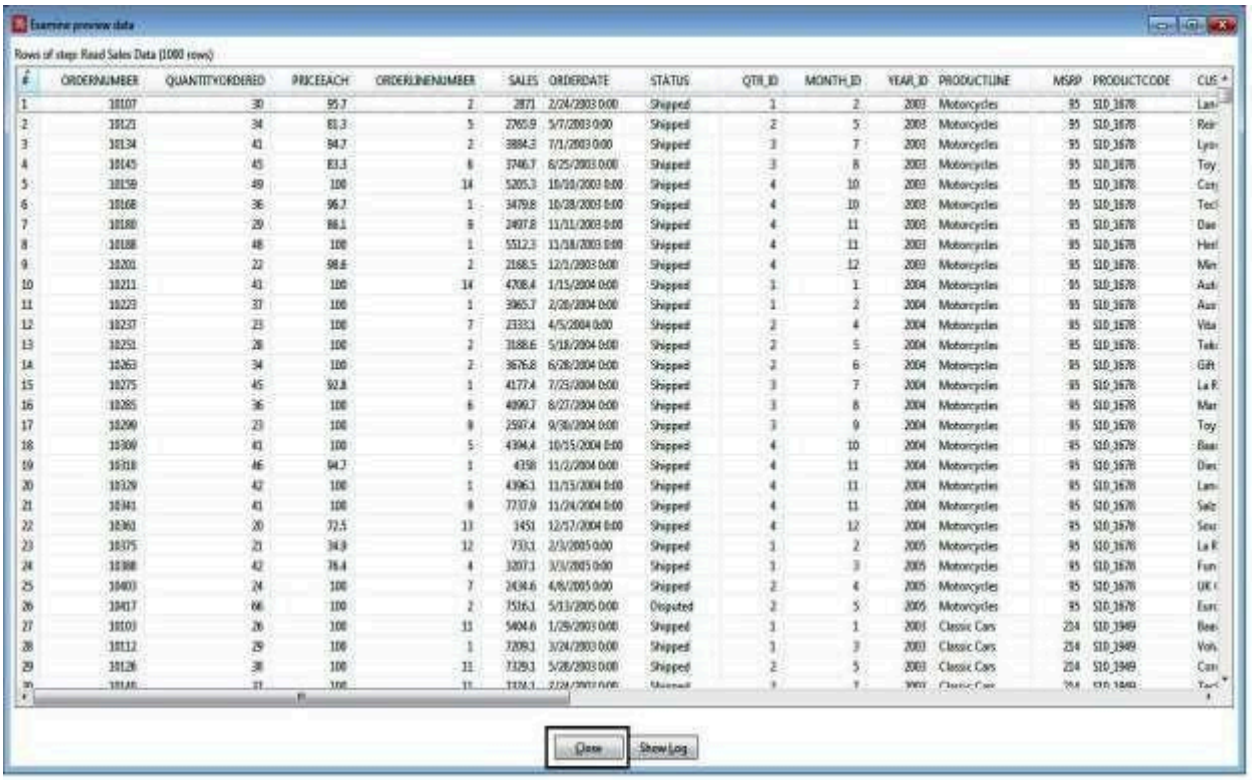


7. **Review the data. Do you notice any missing, incomplete, or variations of the data?**

ADDRESSLINE2, STATE & POSTALCODE contains <null> value.



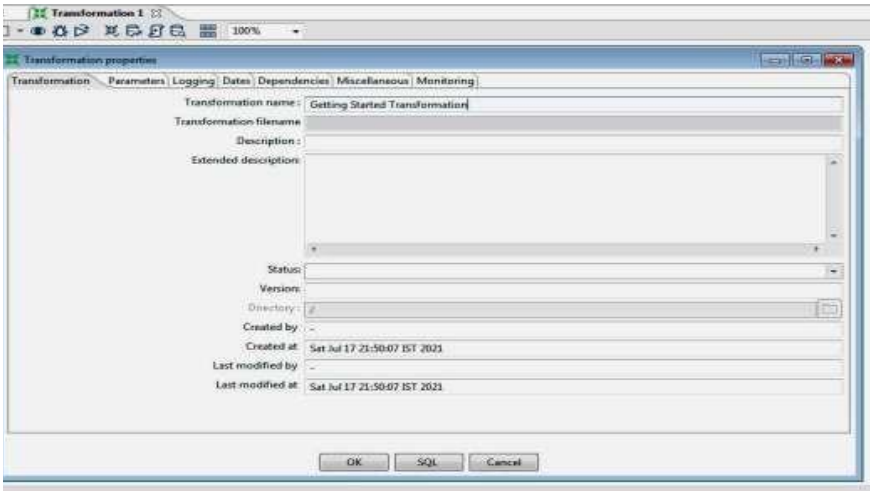
8. Click Close to Examine preview data window. Click OK to save the information that you entered in the step i.e Text file input window.



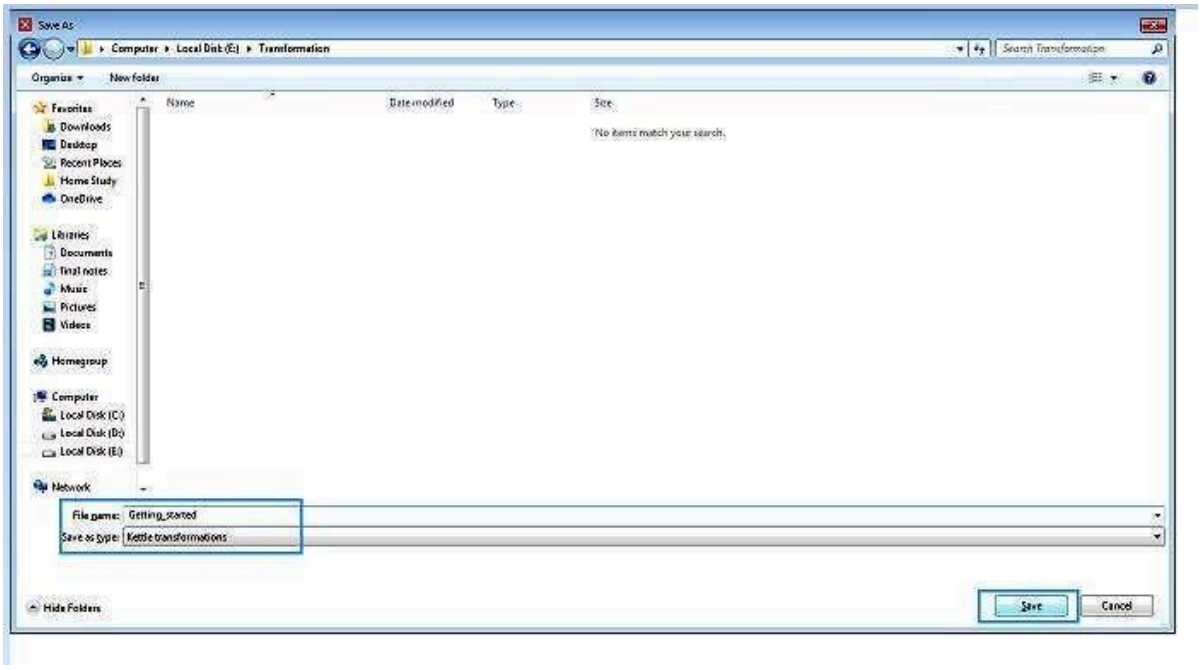
Give the transformation a name and provide additional properties using the Transformation Properties window. There are multiple ways to open the Transformation Properties window.

a. **Right-click on any empty space on the canvas and select properties.**
b. **Double-click on any empty space on the canvas to select properties.**
8. **In the Transformation Name field, type: Getting Started Transformation. Below the name you will see that the filename is empty.**

9. Click OK to close the Transformation Properties window.



To save the transformation, select File Save. This is the first time you are saving transformation so you will be prompted for a file location and name of your choice. You will also see that.ktr is the usual file extension for transformations.



5.Introduction to basics of R programming: - Install packages, loading packages, Data types, checking type of variable, printing variable and objects (Vector, Matrix, List, Factor, Data frame, Table) cbind-ing and rbinding, Reading and Writing data. setwd (), getwd (), data (), rm (), Attaching and Detaching data. Reading data from the consol. Loading data from different data sources. (CSV, Excel).

R is a **programming language** and **software environment** specifically designed for statistical computing, data analysis, and graphical representation of data. It is widely used by statisticians, data analysts, and data scientists for data manipulation, statistical modeling, and producing high-quality visualizations.

A **vector** is a **one-dimensional array** or **list** used to store a collection of elements that are typically of the same type (such as integers, floats, or characters). Vectors are a fundamental data structure in many programming languages, especially in mathematical, statistical, and data science applications.

A **matrix** is a two-dimensional array of numbers, symbols, or expressions arranged in rows and columns. Matrices are a fundamental concept in mathematics, especially in **linear algebra**, and are widely used in various fields such as **computer graphics**, **data science**, **machine learning**, and **physics**. A matrix can be used to represent systems of linear equations, transformations, and other mathematical objects.

A **list** is a data structure used to store an ordered collection of items. Lists are commonly found in many programming languages, and they can hold elements of various types (depending on the language). In general, a list is a flexible container that allows you to add, remove, and access elements, and the elements in the list are usually indexed, allowing for easy retrieval.

A **DataFrame** is a two-dimensional, size-mutable, and potentially heterogeneous tabular data structure, commonly used in data analysis and manipulation. It is one of the most widely used data structures in data science, especially in **R** and **Python (pandas)**, and is analogous to a table in a relational database, an Excel spreadsheet, or a SQL table.

1. Installing packages(xlsx & csv)

2. loading packages(xlsx & csv)

3. Data types: A. checking type of variable. B. printing variable and objects (Vector, Matrix, List, Factor, Data frame, Table) .
4. cbind-ing and rbind-ing
5. Reading and Writing data: A. setwd () B. getwd () C. data () rm ()
6. Attaching and Detaching data: A. Reading data from the consol, B. Loading data from different data sources. (CSV, Excel)

Aim:- Factors in Data Frame:-

Code:-

```
# Create the vectors for the data frame.

 height <- c(132,151,162,139,166,147,122)

weight <- c(48,49,66,53,67,52,40)

gender <-  c("male","male","female","female","male","female","male")



# Create the data frame.

  input_data <-data.frame(height,weight,gender)

print(input_data)

# Test if the gender column is a factor.

print(is.factor(input_data$gender))

 # Print the gender column so see the levels. print(input_data$gender)
```

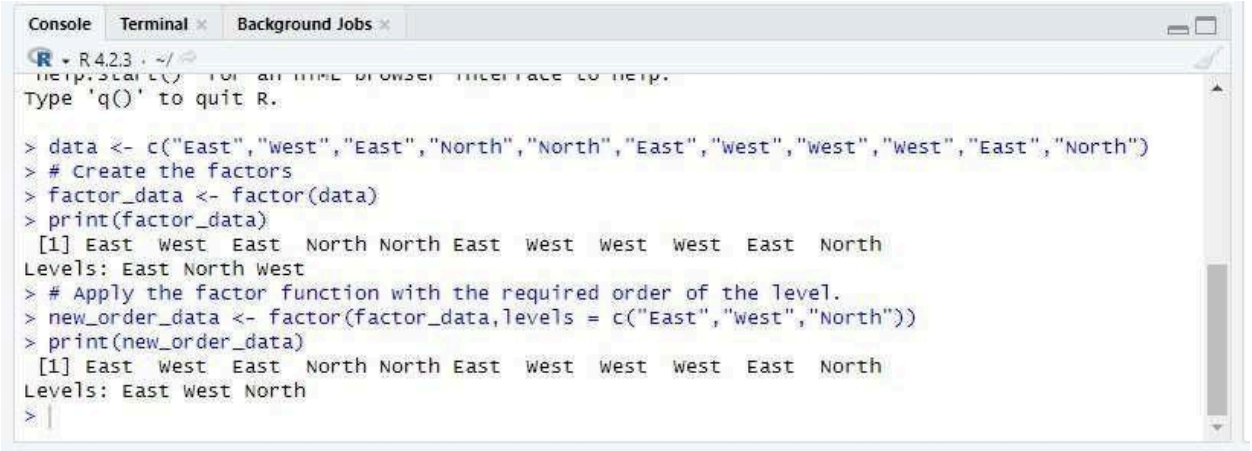
Output:-

Aim:- Changing the Order of Levels:-

Code:-

```
data <- c("East","West","East","North","North","East","West","West","West","East","North")

# Create the factors factor_data <- factor(data) print(factor_data)

# Apply the factor function with the required order of the level.

new_order_data <- factor(factor_data,levels = c("East","West","North")) print(new_order_data)
```

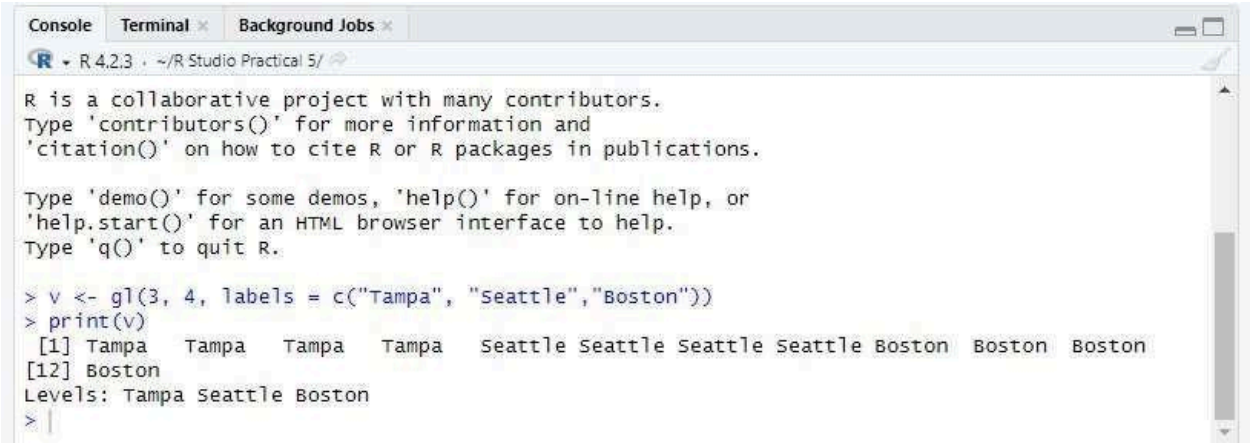
Output:-



Generating Factor Levels

```
v <- gl(3, 4, labels = c("Tampa", "Seattle","Boston"))  print(v)
```

Output:-



R - Data Frames

Create the data frame.

emp.data <- data.frame(emp_id = c (1:5), emp_name =c("Rick","Dan","Michelle","Ryan","Gary"),

salary = c(623.3,515.2,611.0,729.0,843.25),

start_date = as.Date(c("2012-01-01", "2013-09-23","2014-11-15", "2014-05-11","2015-03-27")),
stringsAsFactors = FALSE)

Print the data frame.

print(emp.data)

Output:-

Get the Structure of the Data Frame

Create the data frame.

emp.data <- data.frame(emp_id= c (1:5),

emp_name =c("Rick","Dan","Michelle","Ryan","Gary"),

salary = c(623.3,515.2,611.0,729.0,843.25),

start_date = as.Date(c("2012-01-01", "2013-09-23","2014-11-15", "2014-05-11", "2015-03-27")),

stringsAsFactors = FALSE)

Get the structure of the data frame.

str(emp.data)

Output:-

Summary of Data in Data Frame

Create the data frame.

emp.data <- data.frame(emp_id = c (1:5),

emp_name= c("Rick","Dan","Michelle","Ryan","Gary"), salary = c(623.3,515.2,611.0,729.0,843.25),

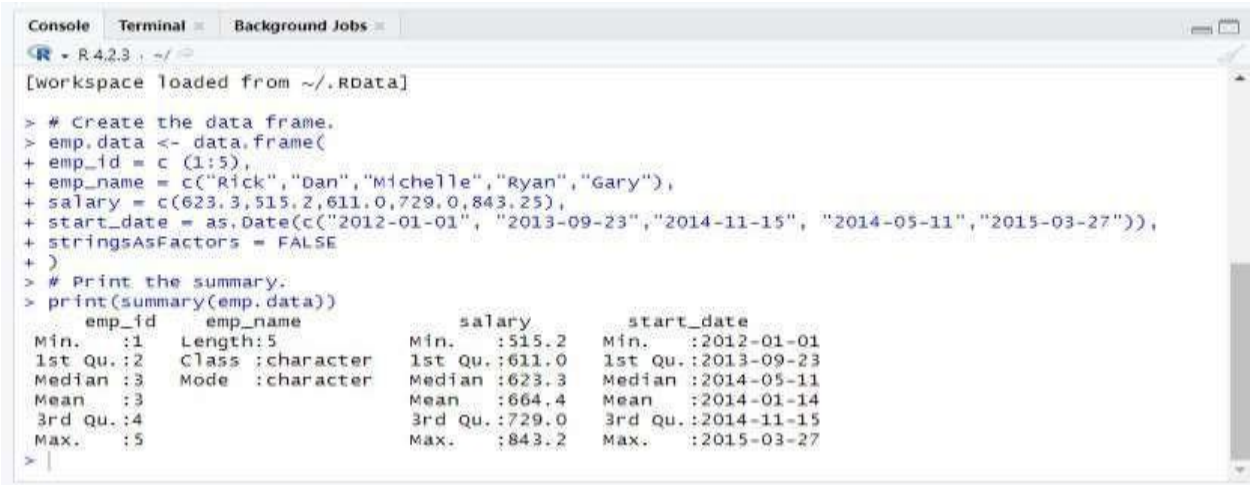
start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15","2014-05-11", "2015-03-27")),

stringsAsFactors = FALSE)

Print the summary.

print(summary(emp.data))

Output:-

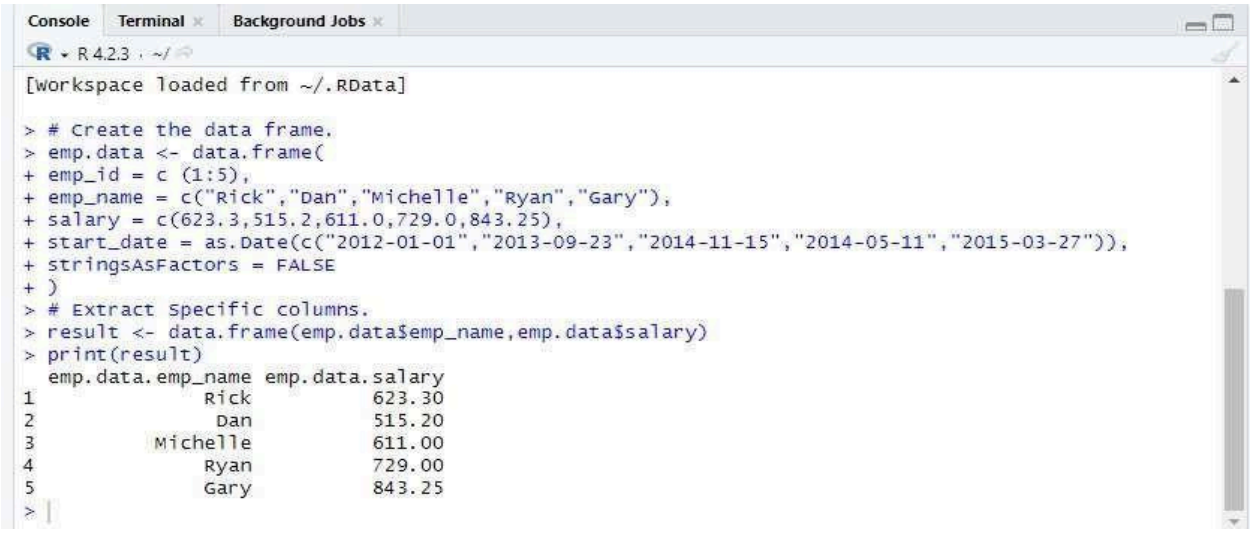

Console
Terminal
Background Jobs
R 4.2.3 · ~/
[Workspace loaded from ~/.RData]
> # Create the data frame.
> emp.data <- data.frame(
+ emp_id = c (1:5),
+ emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
+ salary = c(623.3,515.2,611.0,729.0,843.25),
+ start_date = as.Date(c("2012-01-01", "2013-09-23","2014-11-15", "2014-05-11","2015-03-27")),
+ stringsAsFactors = FALSE
+)
> # Print the summary.
> print(summary(emp.data))
emp_id emp_name salary start_date
Min. :1 Length:5 Min. :515.2 Min. :2012-01-01
1st Qu.:2 Class :character 1st Qu.:611.0 1st Qu.:2013-09-23
Median :3 Mode :character Median :623.3 Median :2014-05-11
Mean :3 Mean :664.4 Mean :2014-01-14
3rd Qu.:4 3rd Qu.:729.0 3rd Qu.:2014-11-15
Max. :5 Max. :843.2 Max. :2015-03-27
>

Aim:- Extract Data from Data Frame

Code:-

```
# Create the data frame. emp.data <- data.frame( emp_id = c (1:5), emp_name =
c("Rick","Dan","Michelle","Ryan","Gary"), salary = c(623.3,515.2,611.0,729.0,843.25), start_date =
as.Date(c("2012-01-01","2013-09-23","2014-11-15","2014-05-11",

"2015-03-27")), stringsAsFactors = FALSE

)

# Extract Specific columns. result <- data.frame(emp.data$emp_name,emp.data$salary) print(result)
```

Output:-

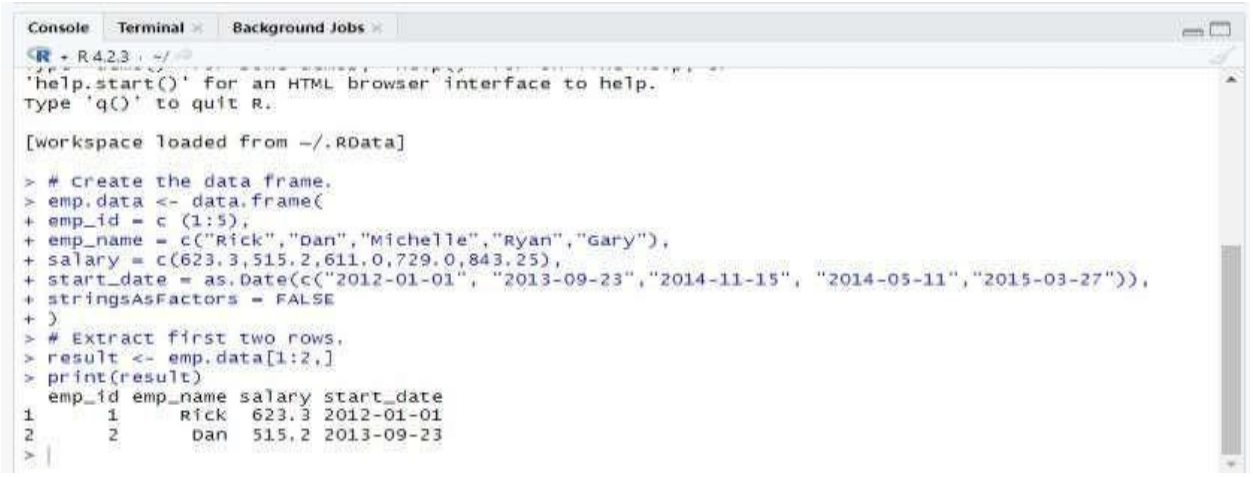


Aim:- Extract the first two rows and then all columns

Code:-

```
# Create the data frame.

emp.data <- data.frame( emp_id = c (1:5),

emp_name = c("Rick","Dan","Michelle","Ryan","Gary"), salary = c(623.3,515.2,611.0,729.0,843.25),

start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11", "2015-03-27")),

 stringsAsFactors = FALSE

)

# Extract first two rows. result <- emp.data[1:2,] print(result)
```

Output:-



Aim:- Extract 3rd and 5th row with 2nd and 4th column

Code:-

Create the data frame

emp.data <- data.frame(emp_id = c (1:5),

emp_name =c("Rick","Dan","Michelle","Ryan","Gary"), salary = c(623.3,515.2,611.0,729.0,843.25),

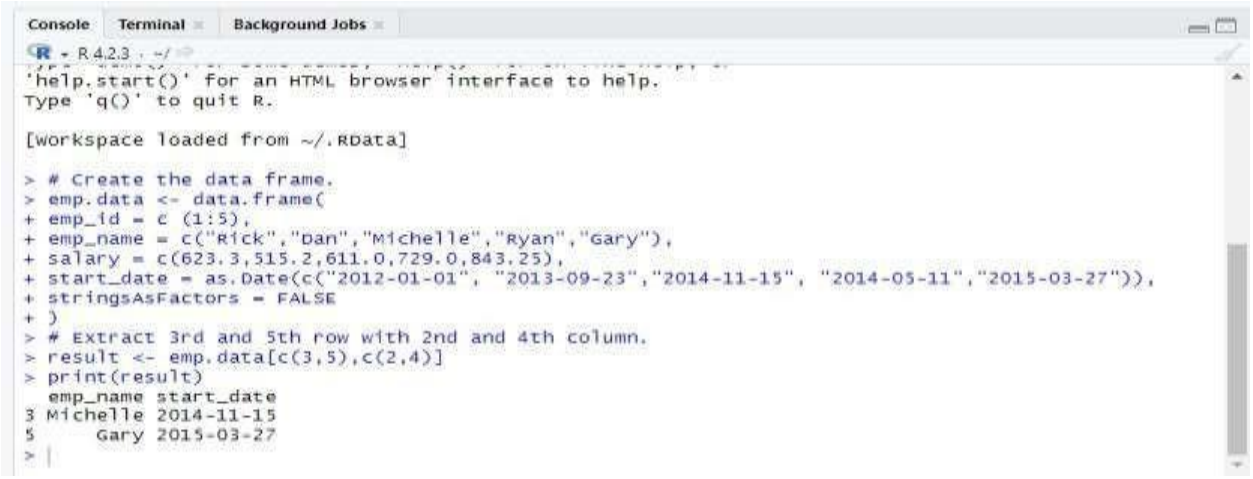
start_date = as.Date(c("2012-01-01", "2013-09-23","2014-11-15", "2014-05-11", "2015-03-27")),

stringsAsFactors = FALSE

)

Extract 3rd and 5th row with 2nd and 4th column. result <- emp.data[c(3,5),c(2,4)] print(result)

Output:-



Aim:- Expand Data Frame A) Add Column

Code:-

Create the data frame.

emp.data <- data.frame(emp_id = c (1:5),

emp_name = c("Rick","Dan","Michelle","Ryan","Gary"), salary = c(623.3,515.2,611.0,729.0,843.25),

start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11", "2015-03-27")),

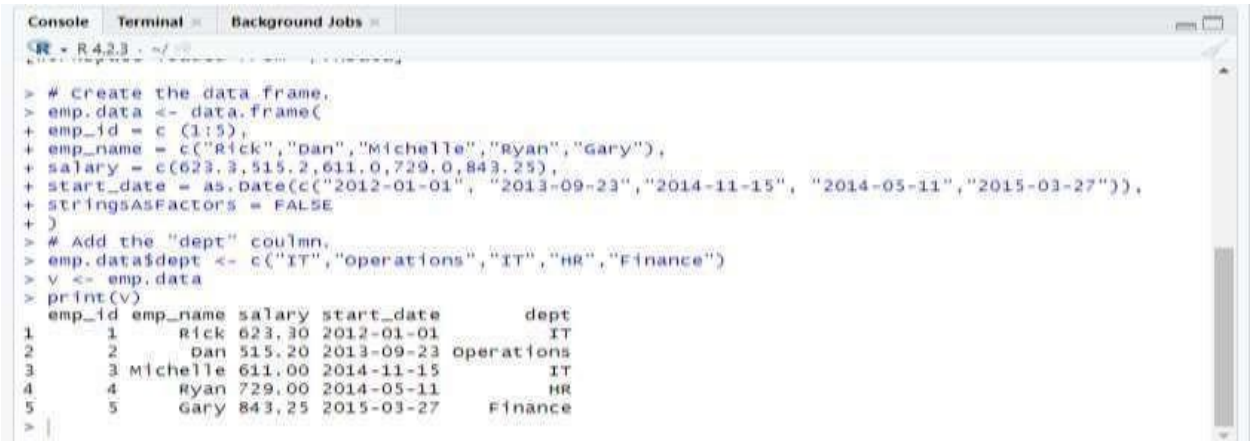
stringsAsFactors = FALSE

)

Add the "dept" coulmn.

emp.data$dept <-c("IT","Operations","IT","HR","Finance") v <- emp.data print(v)

Output:-

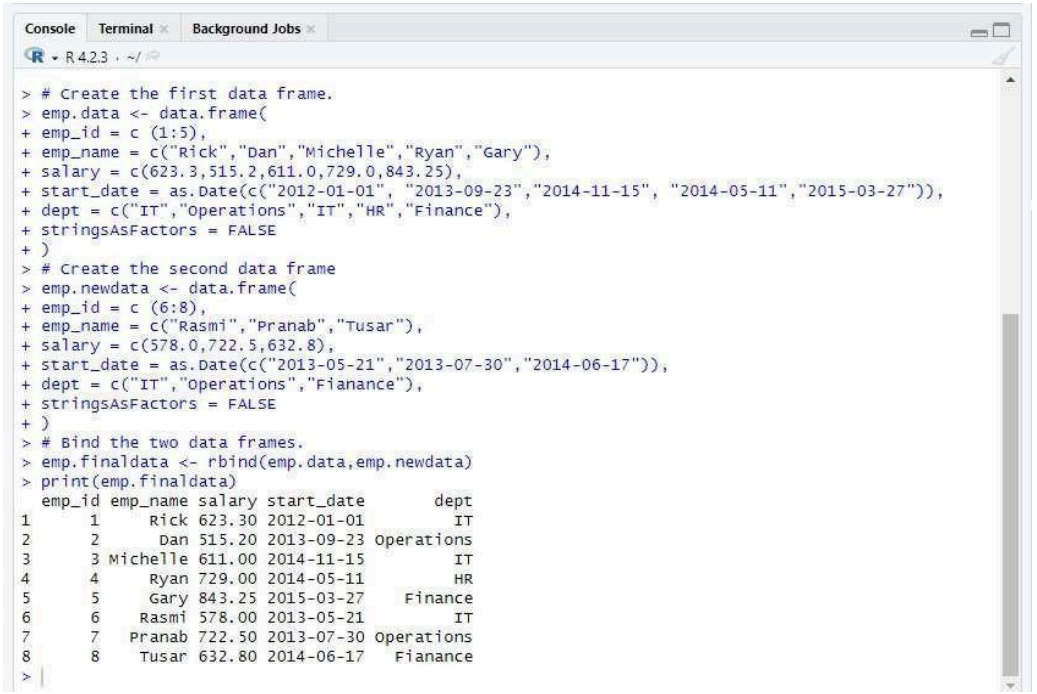


Aim:- Expand Data Frame B) Add Row

Code:-

```
# Create the first data frame.

 emp.data <- data.frame( emp_id = c (1:5),

emp_name = c("Rick","Dan","Michelle","Ryan","Gary"), salary = c(623.3,515.2,611.0,729.0,843.25),

 start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11", "2015-03-27")),

dept = c("IT","Operations","IT","HR","Finance"), stringsAsFactors = FALSE )

# Create the second data frame

emp.newdata <- data.frame( emp_id = c (6:8), emp_name = c("Rasmi","Pranab","Tusar"),

salary = c(578.0,722.5,632.8),

start_date = as.Date(c("2013-05-21","2013-07-30","2014-06-17")),

 dept = c("IT","Operations","Fianance"), stringsAsFactors = FALSE

)

# Bind the two data frames.

emp.finaldata <- rbind(emp.data,emp.newdata)

print(emp.finaldata)
```

Output:-



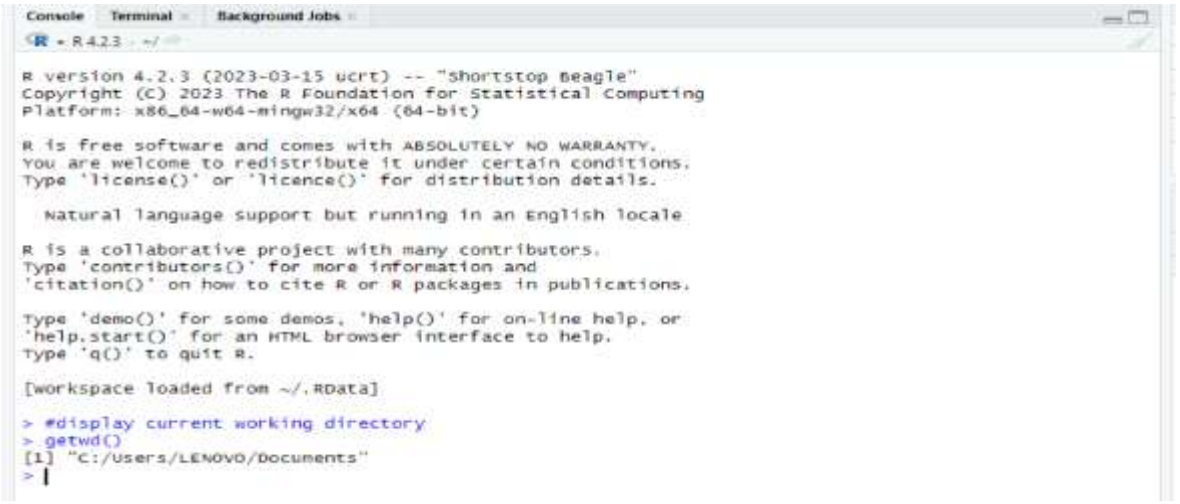
```
> # Create the first data frame.
> emp.data <- data.frame(
+ emp_id = c (1:5),
+ emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
+ salary = c(623.3,515.2,611.0,729.0,843.25),
+ start_date = as.Date(c("2012-01-01", "2013-09-23","2014-11-15", "2014-05-11","2015-03-27")),
+ dept = c("IT","Operations","IT","HR","Finance"),
+ stringsAsFactors = FALSE
+ )
> # Create the second data frame
> emp.newdata <- data.frame(
+ emp_id = c (6:8),
+ emp_name = c("Rasmi","Pranab","Tusar"),
+ salary = c(578.0,722.5,632.8),
+ start_date = as.Date(c("2013-05-21","2013-07-30","2014-06-17")),
+ dept = c("IT","Operations","Fianance"),
+ stringsAsFactors = FALSE
+ )
> # Bind the two data frames.
> emp.finaldata <- rbind(emp.data,emp.newdata)
> print(emp.finaldata)
  emp_id emp_name salary start_date       dept
1      1     Rick 623.30 2012-01-01         IT
2      2      Dan 515.20 2013-09-23 Operations
3      3 Michelle 611.00 2014-11-15         IT
4      4     Ryan 729.00 2014-05-11         HR
5      5     Gary 843.25 2015-03-27    Finance
6      6    Rasmi 578.00 2013-05-21         IT
7      7   Pranab 722.50 2013-07-30 Operations
8      8    Tusar 632.80 2014-06-17   Fianance
>
```

Aim:- Reading and Writing data setwd (), getwd (), data (), rm ()

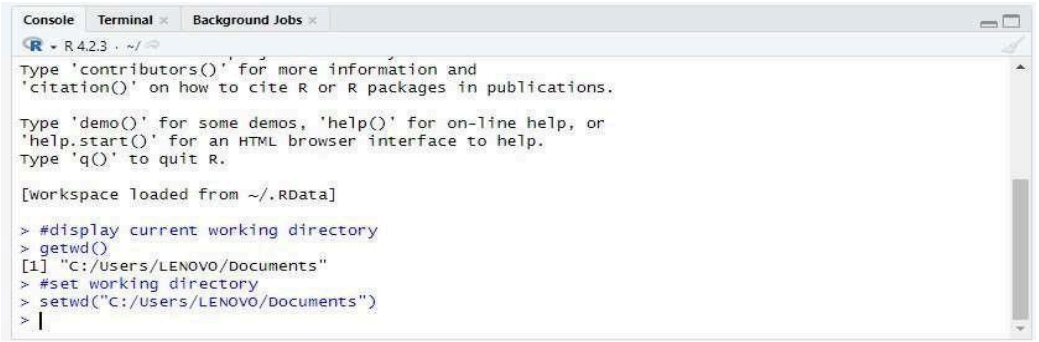
Code:-

#display current working directory getwd()

Output:-



Aim:- Set Working Directory #set working **Code:-** directory setwd("C:/Users/LENOVO/Documents")
output:-



Aim:- Data()

Code:-

Load a specific dataset (e.g., 'iris') data(iris)

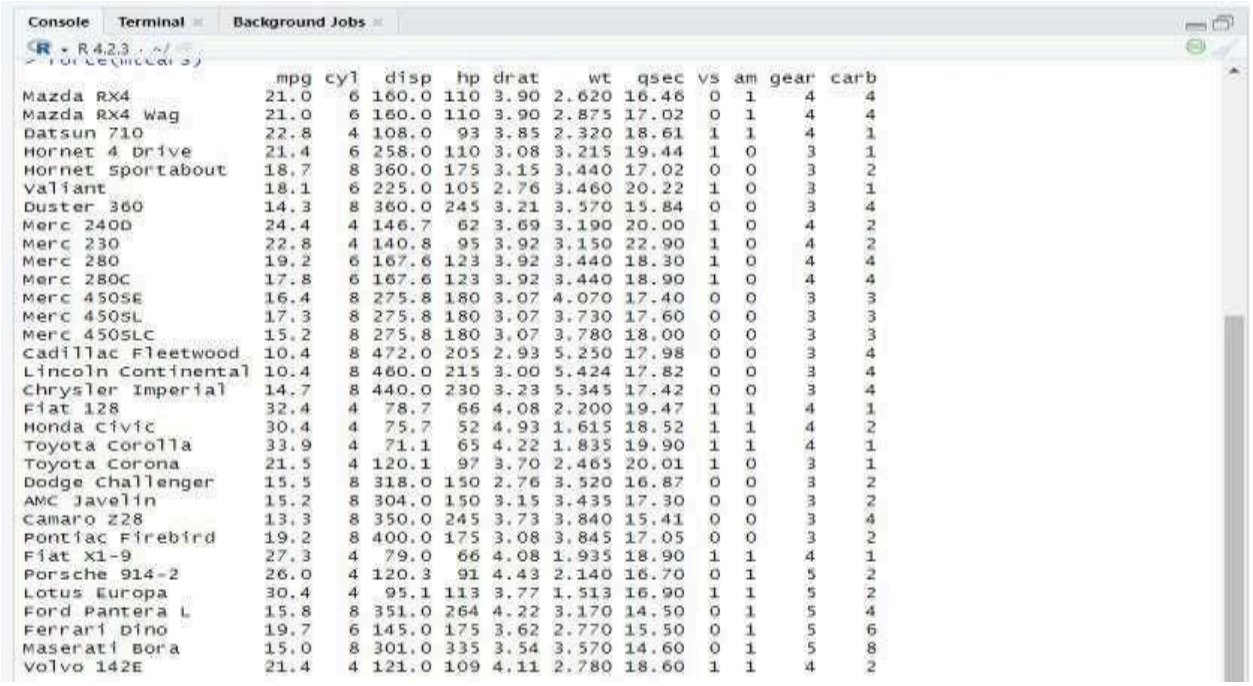
Display the first few rows of the dataset head(iris)

Output:-

```
Console  Terminal  Background Jobs
R 4.2.3 · ~/

[Workspace loaded from ~/.RData]

> # Load a specific dataset (e.g., 'iris')
> data(iris)
> # Display the first few rows of the dataset
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
```

data(mtcars) force(mtcars)

```
Console  Terminal  Background Jobs
R 4.2.3 · ~/
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> data(mtcars)
> force(mtcars)
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
```



```
Console   Terminal   Background Jobs
R 4.2.3 · ~/
> force(mtcars)
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood 10.4  8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128          32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic       30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla    33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona     21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin       15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28        13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird  19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9         27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2     26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa      30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L    15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora     15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E        21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Aim:- Attaching and Detaching data (CSV, Excel)

Get and print current working directory. print(getwd())

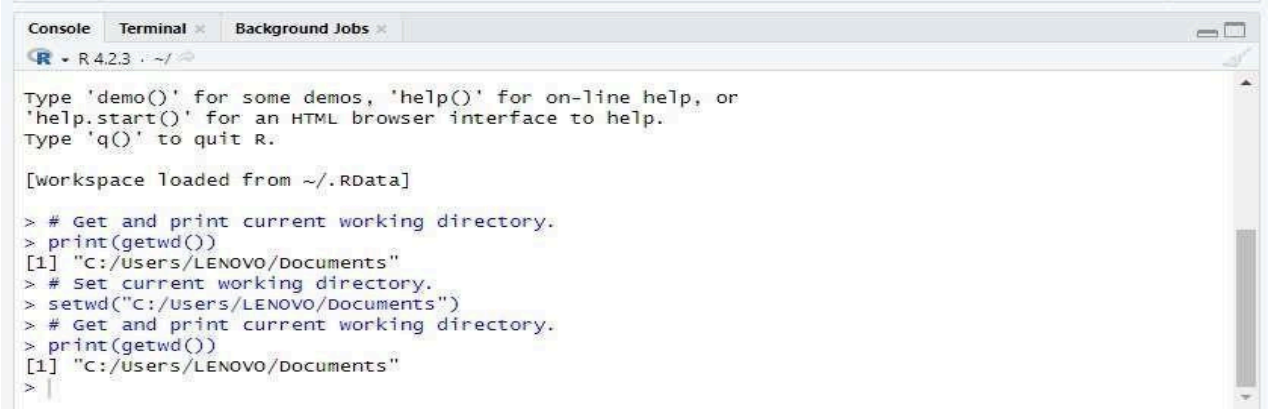
Set current working directory.

setwd("/web/com")

Get and print current working directory.

print(getwd())

Output:-



Aim:- Input as CSV File Code:- id,name,salary,start_date,dept 1,Rick,623.3,2012-01-01,IT

2,Dan,515.2,2013-09,23,Operations

3,Michelle,611,2014-11-15,IT

4,Ryan,729,2014-05-11,HR

5,Gary,843.25,2015-03-27,Finance

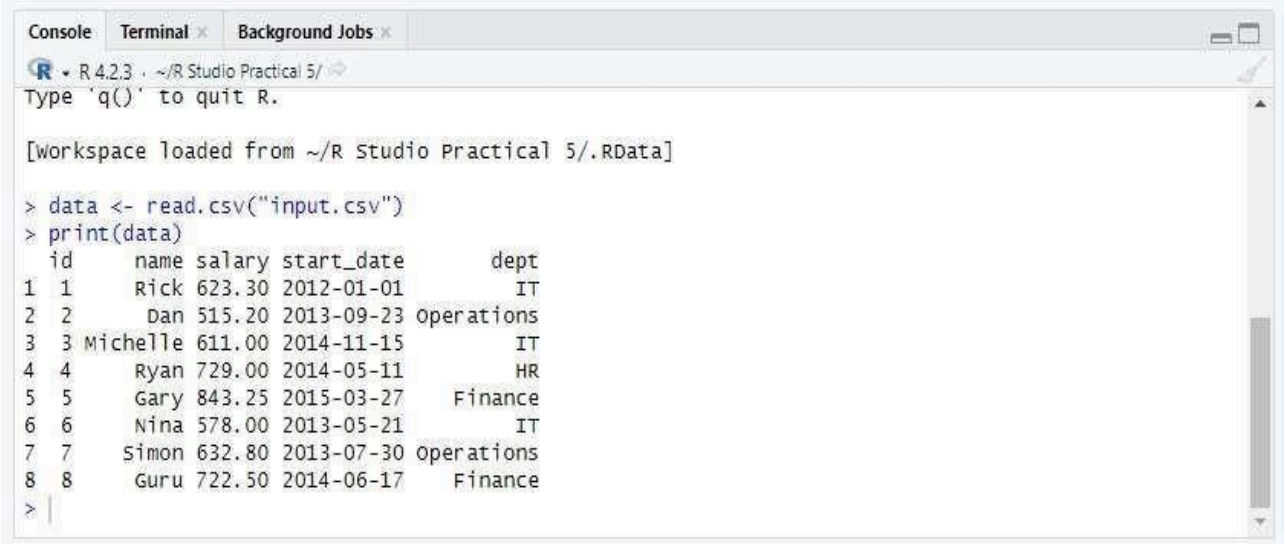
6,Nina,578,2013-05-21,IT

7,Simon,632.8,2013-07-30,Operations

8,Guru,722.5,2014-06-17,Finance

Aim:- Reading a CSV File

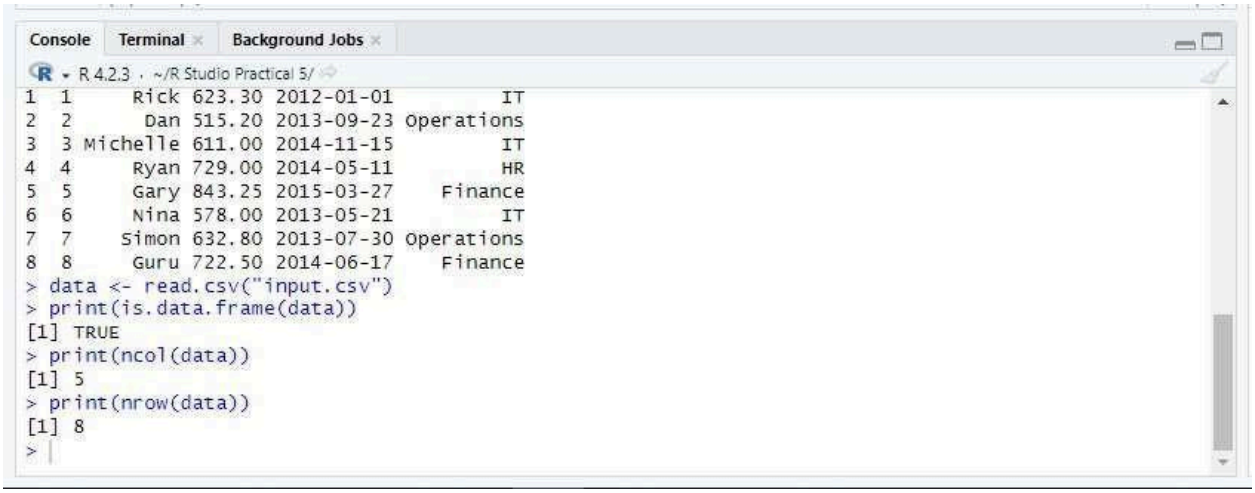
Code:- data <- read.csv("input.csv") print(data)



Aim:- Analyzing the CSV File

Code:- data <- read.csv("input.csv") print(is.data.frame(data)) print(ncol(data)) print(nrow(data))

Output-



Aim:- Get the maximum salary

Code:-

Create a data frame.

data <- read.csv("input.csv")

Get the max salary from the data frame. sal <- max(data$salary) print(sal)

Output:-

Aim:- Get the details of the person with max salary

Code:-

Create a data frame.

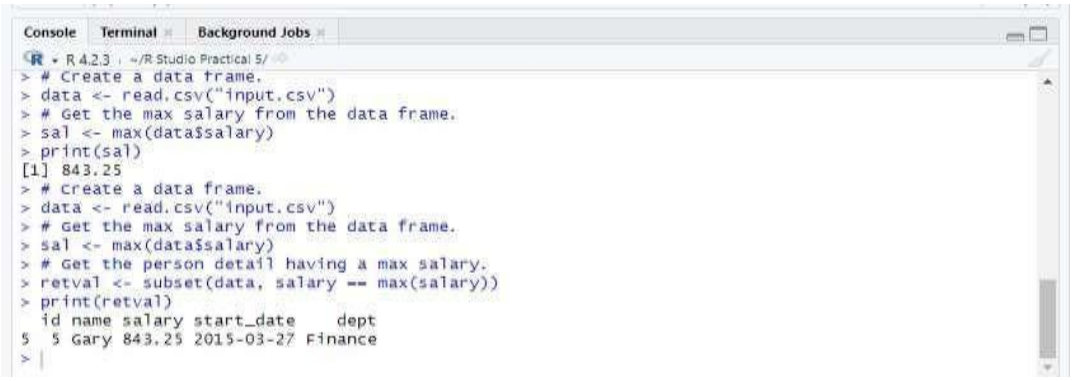
data <- read.csv("input.csv")

Get the max salary from the data frame.

sal <- max(data$salary)

Get the person detail having a max salary. retval <- subset(data, salary == max(salary)) print(retval)

Output:-



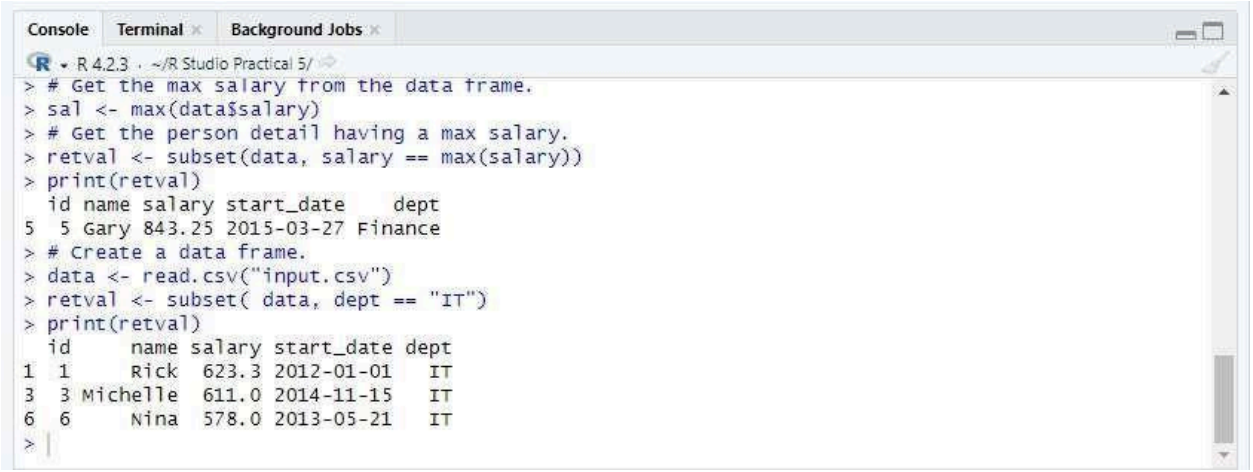
Aim:- Get all the people working in IT department

Code:-

Create a data frame.

data <- read.csv("input.csv") retval <- subset(data, dept == "IT") print(retval)

Output:-



6. Implementation of Data preprocessing techniques like, Naming and Renaming variables, adding a new variable. Dealing with missing data. Dealing with categorical data. Data reduction using subsetting

Data Preprocessing refers to the set of techniques used to prepare raw data for analysis or modeling by transforming it into a clean and organized format. It involves a series of steps aimed at improving the quality of data, ensuring it is suitable for further analysis, statistical modeling, or machine learning applications. Data preprocessing is crucial because real-world data is often incomplete, inconsistent, or noisy, and raw data cannot be directly fed into machine learning models without significant modification.

Naming and Renaming Variables refers to the process of assigning or changing the names of variables (or columns) in a dataset or a programming environment. In the context of data analysis and programming, variables are used to store data values, and the name of a variable is used to refer to that stored value.

Adding a New Variable refers to the process of introducing a new column or feature into a dataset or a new variable into a program, model, or analysis. This new variable can be derived from existing variables, based on specific logic, or can represent a completely new attribute.

Dealing with Missing Data refers to the various techniques used to handle incomplete or missing entries in a dataset. Missing data is a common challenge in data analysis, machine learning, and statistical modeling. If not addressed properly, missing data can lead to biased results, reduced accuracy, and unreliable conclusions.

Dealing with Categorical Data refers to the process of handling variables that represent categories or groups, rather than numerical values. Categorical data is common in many datasets and can include things like gender, product types, country names, or any other feature where the values are labels or classes.

Data Reduction Using Subsetting refers to the process of reducing the size of a dataset by selecting only a subset of the data that is relevant to the analysis, model, or task at hand. This is an essential data preprocessing technique used to simplify datasets, improve computational efficiency, and focus on the most important aspects of the data.

1. **Naming and Renaming variables, adding a new variable.**
2. **Dealing with missing data.**
3. **Dealing with categorical data.**
4. **Data reduction using subsetting.**

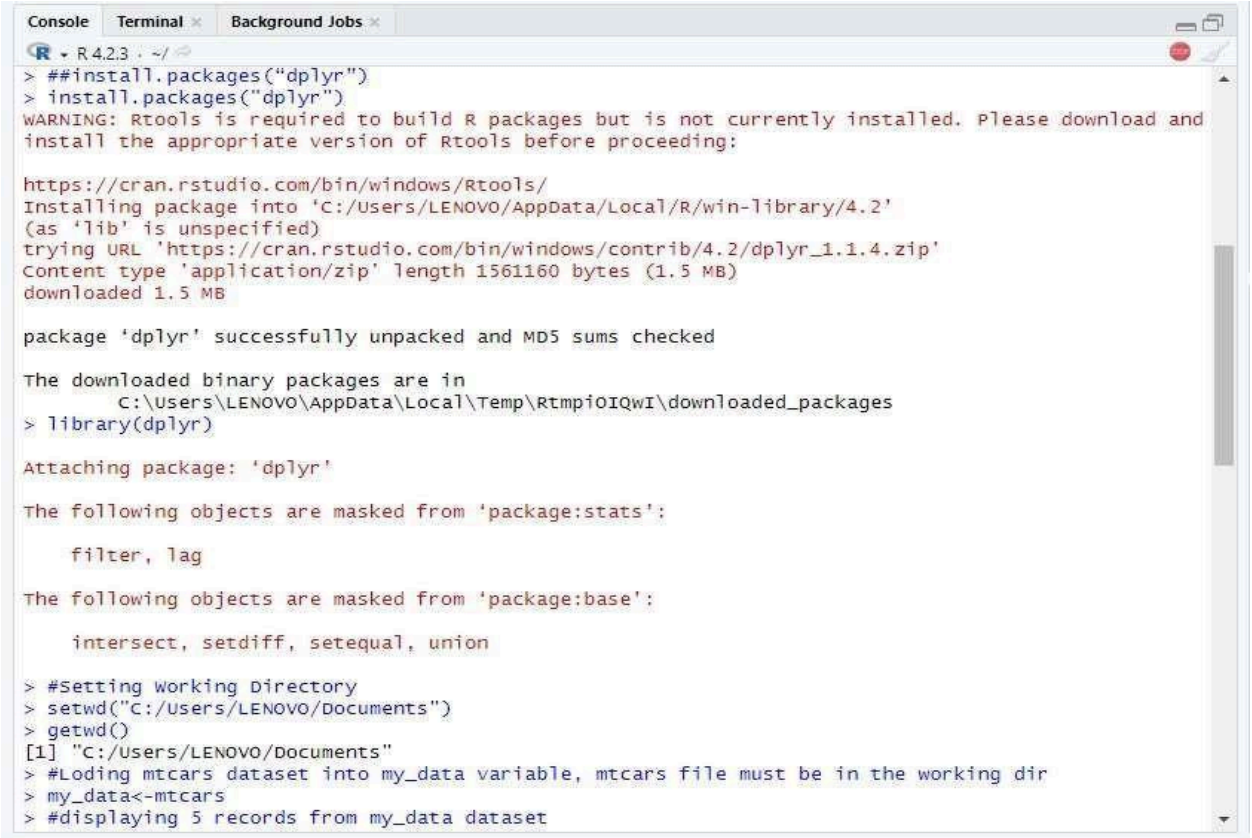
Aim:- PlayingWithVariable

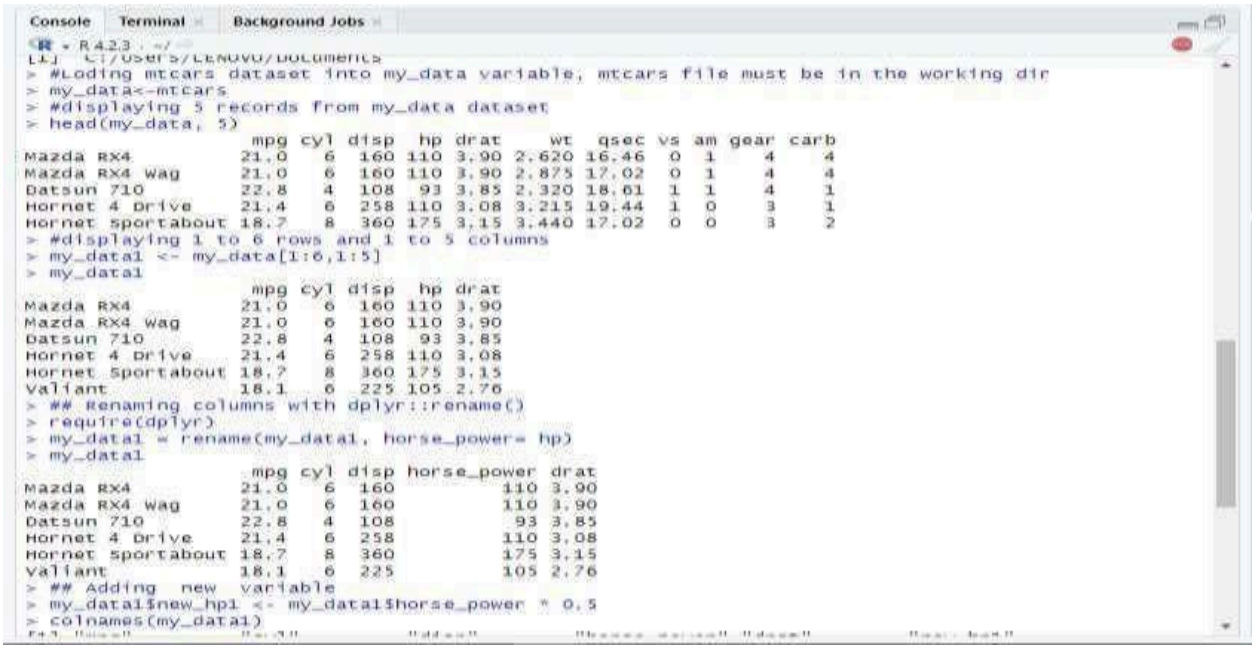
Code:-

```
##install.packages(“dplyr”) install.packages("dplyr") library(dplyr)

#Setting Working Directory setwd("C:/Users/LENOVO/Documents")

 getwd()

#Loding mtcars dataset into my_data variable, mtcars file must be in the working dir
```

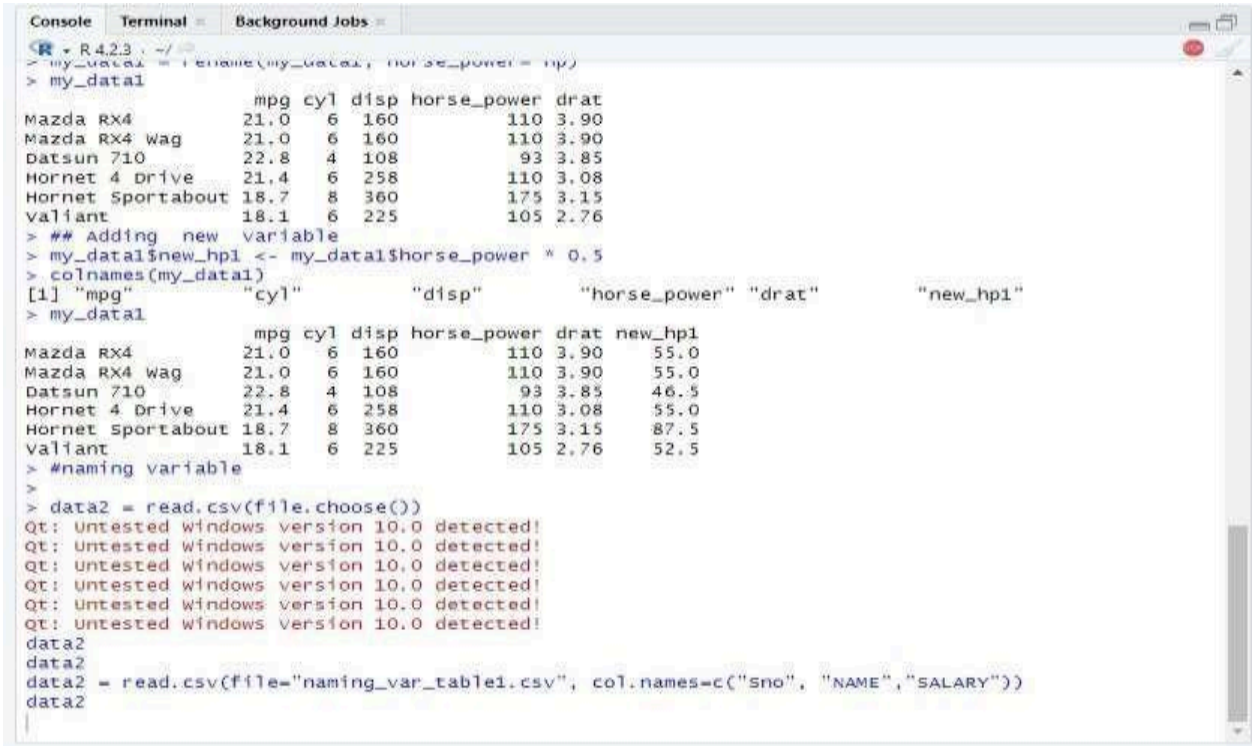
```
my_data<-mtcars

#displaying 5 records from my_data dataset head

(my_data, 5)

#displaying 1 to 6 rows and 1 to 5 columns my_data1 <- my_data[1:6,1:5]  my_data1

## Renaming columns with

dplyr::rename() require(dplyr) my_data1 = rename(my_data1, horse_power= hp) my_data1



## Adding  new  variable

my_data1$new_hp1 <- my_data1$horse_power * 0.5 colnames(my_data1) my_data1

#naming variable



data2 = read.csv(file.choose()) data2 data2 = read.csv(file="naming_var_table1.csv", col.names=c("Sno",
"NAME","SALARY")) data2
```

Output:-


Console
Terminal
Background Jobs
R 4.2.3 · ~/
> ##install.packages("dplyr")
> install.packages("dplyr")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and
install the appropriate version of Rtools before proceeding:
https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/LENOVO/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/dplyr_1.1.4.zip'
Content type 'application/zip' length 1561160 bytes (1.5 MB)
downloaded 1.5 MB
package 'dplyr' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\LENOVO\AppData\Local\Temp\RtmpiOIQwI\downloaded_packages
> 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
> #Setting Working Directory
> setwd("C:/Users/LENOVO/Documents")
> getwd()
[1] "C:/Users/LENOVO/Documents"
> #Loding mtcars dataset into my_data variable, mtcars file must be in the working dir
> my_data<-mtcars
> #displaying 5 records from my_data dataset



```
Console  Terminal  Background Jobs
R 4.2.3 · ~/
[1] "C:/Users/LENOVO/Documents"
> #Loding mtcars dataset into my_data variable, mtcars file must be in the working dir
> my_data<-mtcars
> #displaying 5 records from my_data dataset
> head(my_data, 5)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
> #displaying 1 to 6 rows and 1 to 5 columns
> my_data1 <- my_data[1:6,1:5]
> my_data1
                   mpg cyl disp  hp drat
Mazda RX4         21.0   6  160 110 3.90
Mazda RX4 Wag     21.0   6  160 110 3.90
Datsun 710        22.8   4  108  93 3.85
Hornet 4 Drive    21.4   6  258 110 3.08
Hornet Sportabout 18.7   8  360 175 3.15
Valiant           18.1   6  225 105 2.76
> ## Renaming columns with dplyr::rename()
> require(dplyr)
> my_data1 = rename(my_data1, horse_power= hp)
> my_data1
                   mpg cyl disp horse_power drat
Mazda RX4         21.0   6  160         110 3.90
Mazda RX4 Wag     21.0   6  160         110 3.90
Datsun 710        22.8   4  108          93 3.85
Hornet 4 Drive    21.4   6  258         110 3.08
Hornet Sportabout 18.7   8  360         175 3.15
Valiant           18.1   6  225         105 2.76
> ## Adding  new  variable
> my_data1$new_hp1 <- my_data1$horse_power * 0.5
> colnames(my_data1)
```



```
Console  Terminal  Background Jobs
R 4.2.3 · ~/
> my_data1
                   mpg cyl disp horse_power drat
Mazda RX4         21.0   6  160         110 3.90
Mazda RX4 Wag     21.0   6  160         110 3.90
Datsun 710        22.8   4  108          93 3.85
Hornet 4 Drive    21.4   6  258         110 3.08
Hornet Sportabout 18.7   8  360         175 3.15
Valiant           18.1   6  225         105 2.76
> ## Adding  new  variable
> my_data1$new_hp1 <- my_data1$horse_power * 0.5
> colnames(my_data1)
[1] "mpg"         "cyl"         "disp"        "horse_power" "drat"        "new_hp1"
> my_data1
                   mpg cyl disp horse_power drat new_hp1
Mazda RX4         21.0   6  160         110 3.90    55.0
Mazda RX4 Wag     21.0   6  160         110 3.90    55.0
Datsun 710        22.8   4  108          93 3.85    46.5
Hornet 4 Drive    21.4   6  258         110 3.08    55.0
Hornet Sportabout 18.7   8  360         175 3.15    87.5
Valiant           18.1   6  225         105 2.76    52.5
> #naming variable
>
> data2 = read.csv(file.choose())
Qt: Untested Windows version 10.0 detected!
Qt: Untested Windows version 10.0 detected!
Qt: Untested Windows version 10.0 detected!
Qt: Untested Windows version 10.0 detected!
Qt: Untested Windows version 10.0 detected!
Qt: Untested Windows version 10.0 detected!
data2
data2
data2 = read.csv(file="naming_var_table1.csv", col.names=c("Sno", "NAME","SALARY"))
data2
```

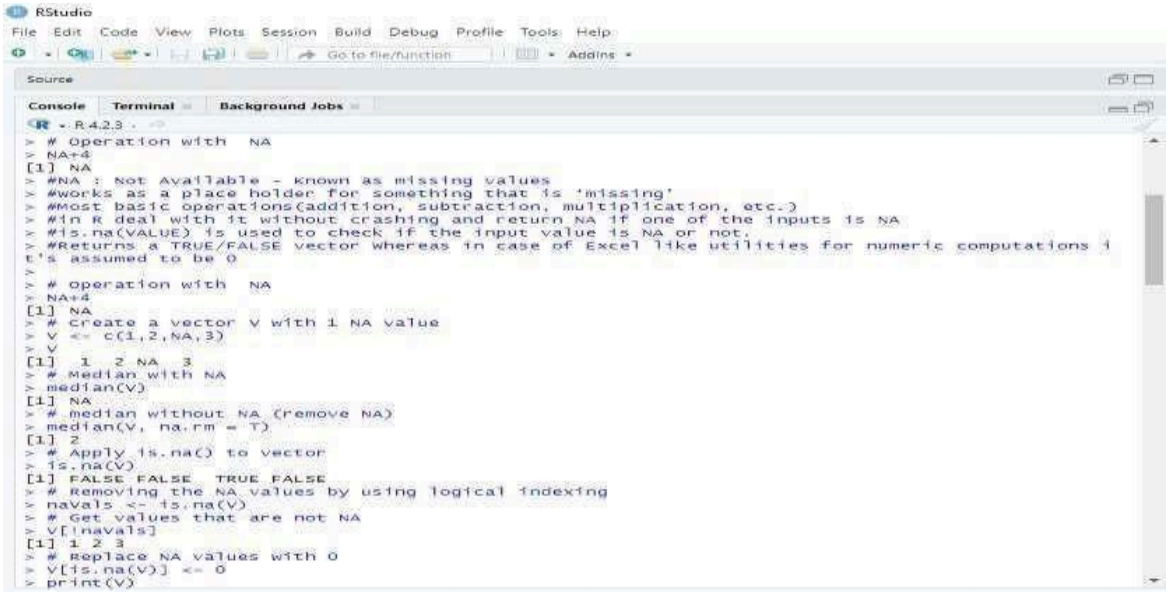
Aim:- MissingValues

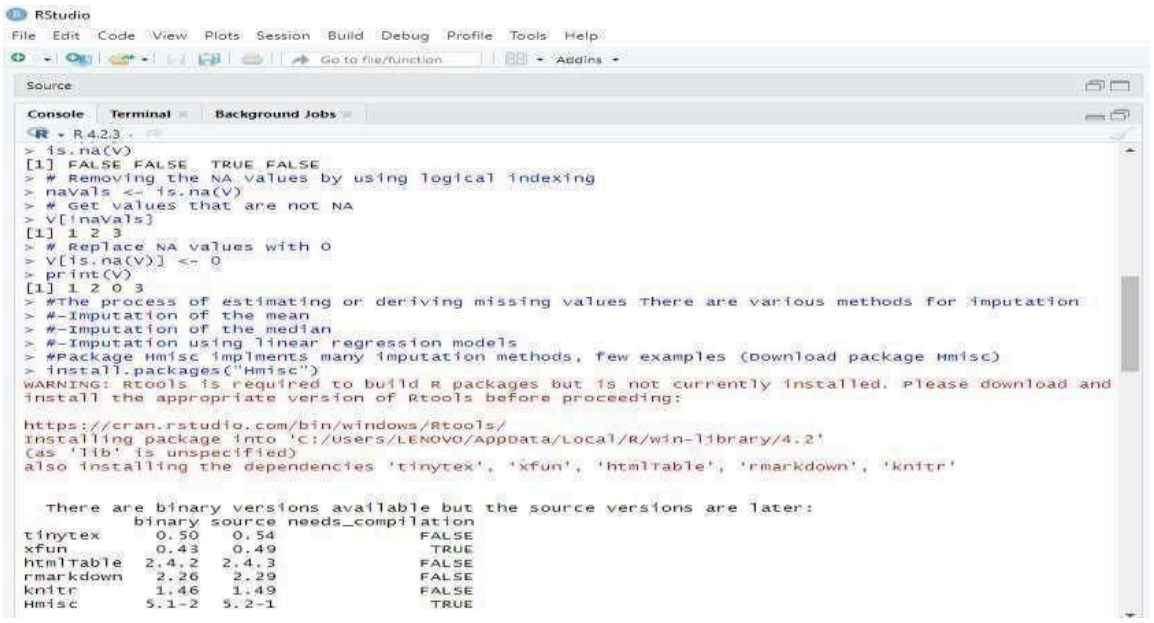
Code:-

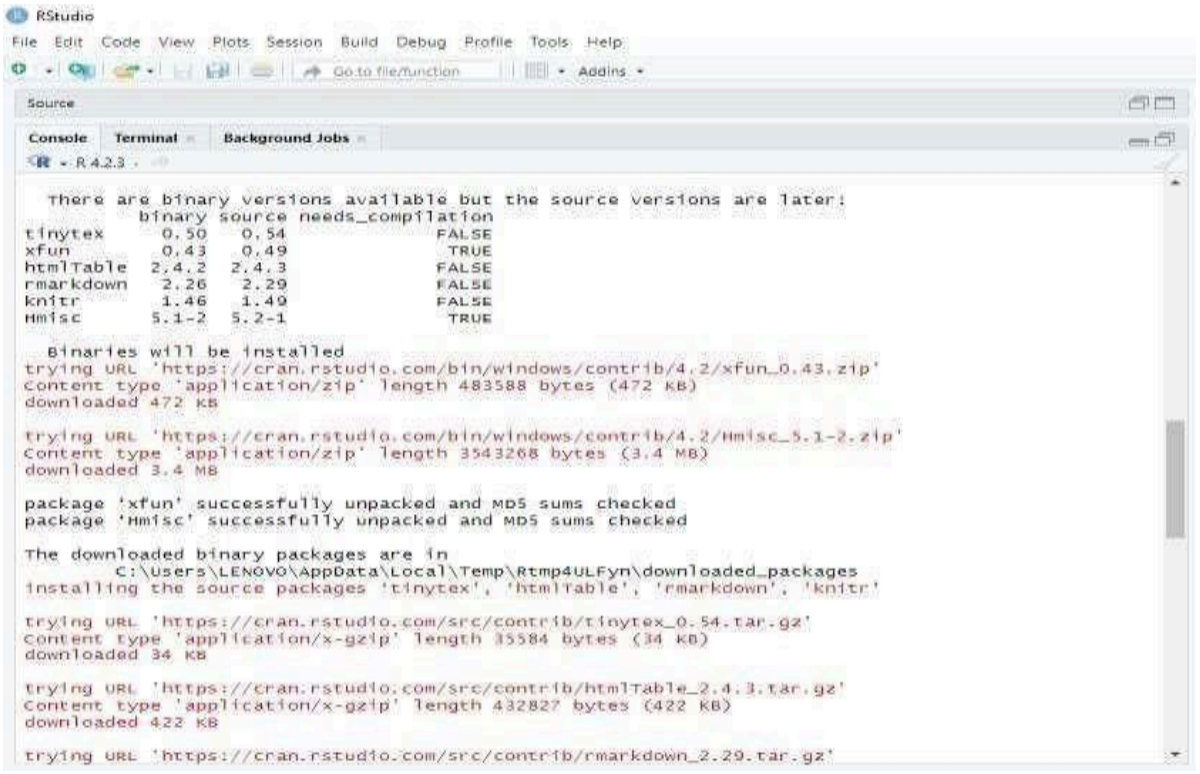
```
#NA : Not Available - Known as missing values

#Works as a place holder for something that is 'missing'

#Most basic operations(addition, subtraction, multiplication, etc.)  #in R deal with it without crashing and return NA if one of the inputs is NA #is.na(VALUE) is used to check if the input value is NA or not.

#Returns a TRUE/FALSE vector Whereas in case of Excel like utilities for numeric computations it's assumed to be 0

# Operation with  NA

NA+4

# Create a vector V with 1 NA value

V <- c(1,2,NA,3)

V

# Median with NA median(V)

# median without NA (remove NA) median(V, na.rm = T)  # Apply is.na() to vector is.na(V)

# Removing the NA values by using logical indexing naVals <- is.na(V)

# Get values that are not NA

V[!naVals]

# Replace NA values with 0

V[is.na(V)] <- 0 print(V)

#The process of estimating or deriving missing values There are various methods for imputation

#–Imputation of the mean

#–Imputation of the median

#–Imputation using linear regression models

#Package Hmisc implments many imputation methods, few examples (Download package Hmisc)
install.packages("Hmisc")   library(Hmisc) ##  create  a  vector x = c(1,2,3,NA,4,4,NA) x
```

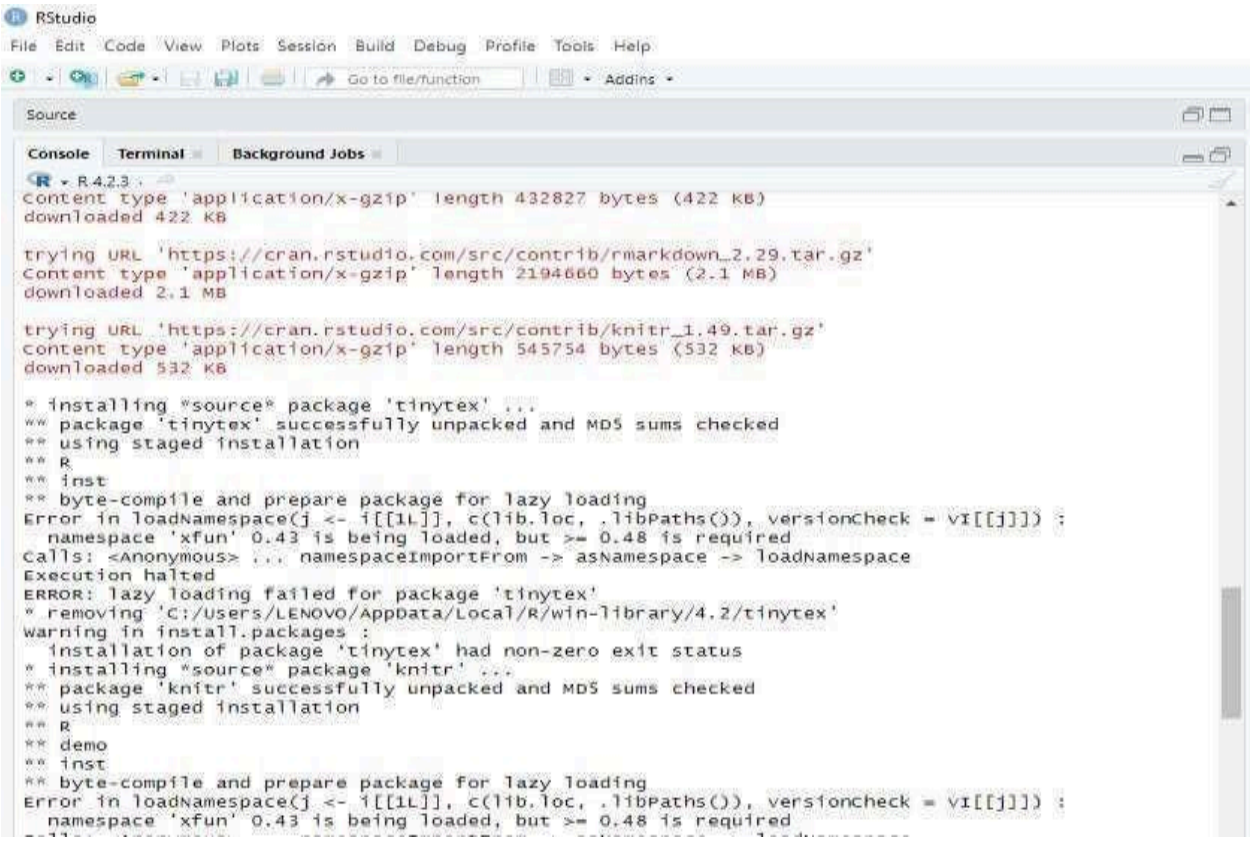
mean imputation - from package, mention name of function to be used x <- impute(x, fun = mean) x

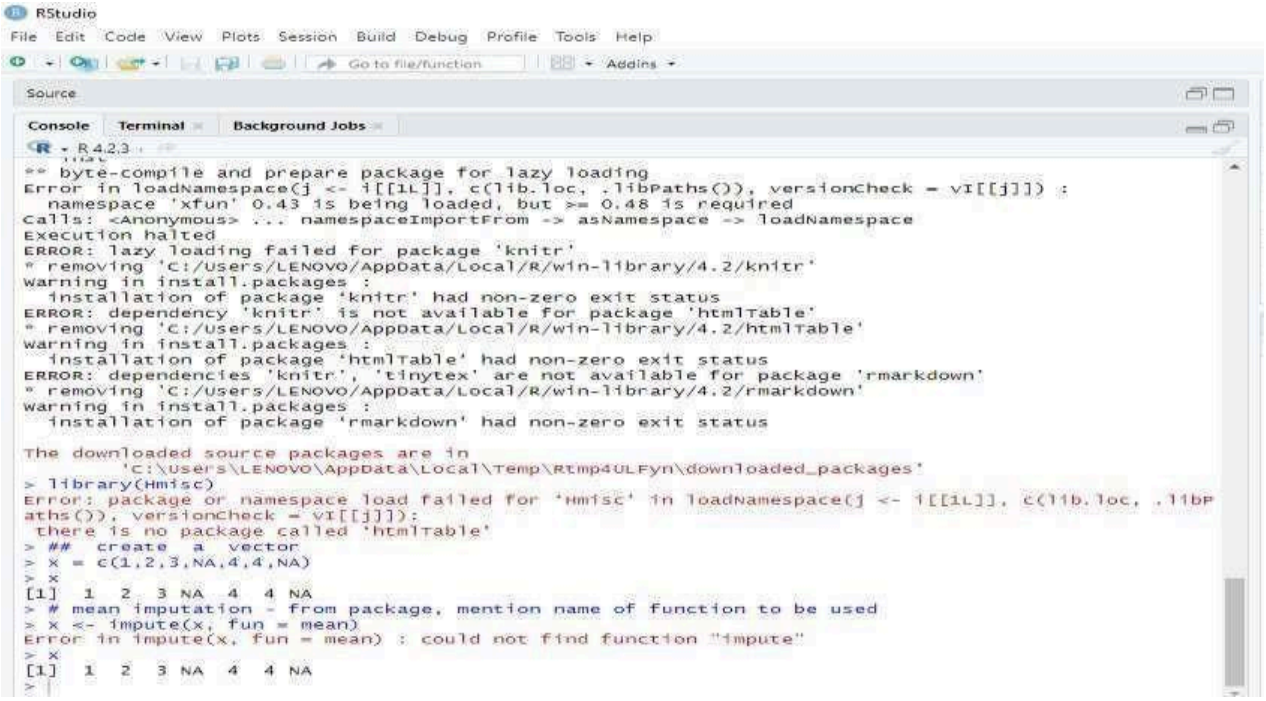
Output:-






RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Terminal Background Jobs
R 4.2.3 ·
There are binary versions available but the source versions are later:
binary source needs_compilation
tinytex 0.50 0.54 FALSE
xfun 0.43 0.49 TRUE
htmlTable 2.4.2 2.4.3 FALSE
rmarkdown 2.26 2.29 FALSE
knitr 1.46 1.49 FALSE
Hmisc 5.1-2 5.2-1 TRUE
Binaries will be installed
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/xfun_0.43.zip'
Content type 'application/zip' length 483588 bytes (472 KB)
downloaded 472 KB
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/Hmisc_5.1-2.zip'
Content type 'application/zip' length 3543268 bytes (3.4 MB)
downloaded 3.4 MB
package 'xfun' successfully unpacked and MD5 sums checked
package 'Hmisc' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\LENOVO\AppData\Local\Temp\Rtmp4ULFyn\downloaded_packages
installing the source packages 'tinytex', 'htmlTable', 'rmarkdown', 'knitr'
trying URL 'https://cran.rstudio.com/src/contrib/tinytex_0.54.tar.gz'
Content type 'application/x-gzip' length 35584 bytes (34 KB)
downloaded 34 KB
trying URL 'https://cran.rstudio.com/src/contrib/htmlTable_2.4.3.tar.gz'
Content type 'application/x-gzip' length 432827 bytes (422 KB)
downloaded 422 KB
trying URL 'https://cran.rstudio.com/src/contrib/rmarkdown_2.29.tar.gz'


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Terminal Background Jobs
R 4.2.3
Content type 'application/x-gzip' length 432827 bytes (422 KB)
downloaded 422 KB
trying URL 'https://cran.rstudio.com/src/contrib/rmarkdown_2.29.tar.gz'
Content type 'application/x-gzip' length 2194660 bytes (2.1 MB)
downloaded 2.1 MB
trying URL 'https://cran.rstudio.com/src/contrib/knitr_1.49.tar.gz'
Content type 'application/x-gzip' length 545754 bytes (532 KB)
downloaded 532 KB
* installing *source* package 'tinytex' ...
** package 'tinytex' successfully unpacked and MD5 sums checked
** using staged installation
** R
** inst
** byte-compile and prepare package for lazy loading
Error in loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), versionCheck = vI[[j]]) :
namespace 'xfun' 0.43 is being loaded, but >= 0.48 is required
Calls: <Anonymous> ... namespaceImportFrom -> asNamespace -> loadNamespace
Execution halted
ERROR: lazy loading failed for package 'tinytex'
* removing 'C:/Users/LENOVO/AppData/Local/R/win-library/4.2/tinytex'
Warning in install.packages :
installation of package 'tinytex' had non-zero exit status
* installing *source* package 'knitr' ...
** package 'knitr' successfully unpacked and MD5 sums checked
** using staged installation
** R
** demo
** inst
** byte-compile and prepare package for lazy loading
Error in loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), versionCheck = vI[[j]]) :
namespace 'xfun' 0.43 is being loaded, but >= 0.48 is required

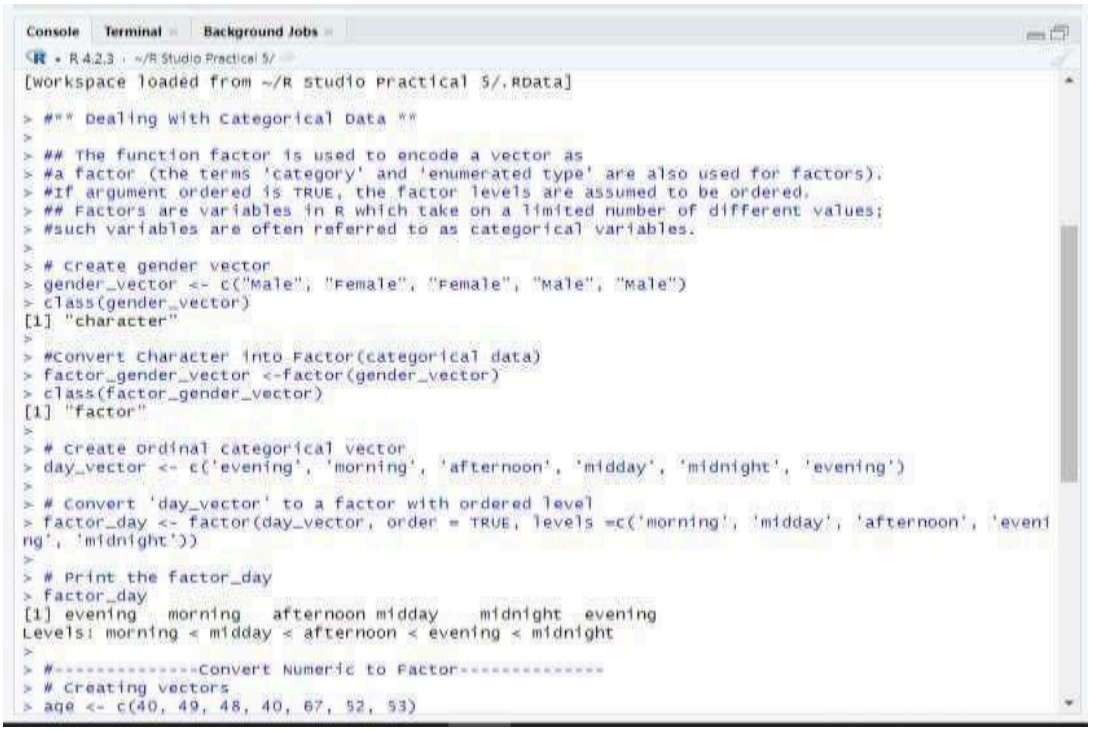

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Terminal Background Jobs
R 4.2.3
** byte-compile and prepare package for lazy loading
Error in loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), versionCheck = vI[[j]]) :
namespace 'xfun' 0.43 is being loaded, but >= 0.48 is required
Calls: <Anonymous> ... namespaceImportFrom -> asNamespace -> loadNamespace
Execution halted
ERROR: lazy loading failed for package 'knitr'
* removing 'C:/Users/LENOVO/AppData/Local/R/win-library/4.2/knitr'
Warning in install.packages :
installation of package 'knitr' had non-zero exit status
ERROR: dependency 'knitr' is not available for package 'htmlTable'
* removing 'C:/Users/LENOVO/AppData/Local/R/win-library/4.2/htmlTable'
Warning in install.packages :
installation of package 'htmlTable' had non-zero exit status
ERROR: dependencies 'knitr', 'tinytex' are not available for package 'rmarkdown'
* removing 'C:/Users/LENOVO/AppData/Local/R/win-library/4.2/rmarkdown'
Warning in install.packages :
installation of package 'rmarkdown' had non-zero exit status
The downloaded source packages are in
'C:\Users\LENOVO\AppData\Local\Temp\Rtmp4ULFyn\downloaded_packages'
> library(Hmisc)
Error: package or namespace load failed for 'Hmisc' in loadNamespace(j <- i[[1L]], c(lib.loc, .libP
aths()), versionCheck = vI[[j]]):
there is no package called 'htmlTable'
> ## create a vector
> x = c(1,2,3,NA,4,4,NA)
> x
[1] 1 2 3 NA 4 4 NA
> # mean imputation - from package, mention name of function to be used
> x <- impute(x, fun = mean)
Error in impute(x, fun = mean) : could not find function "impute"
> x
[1] 1 2 3 NA 4 4 NA

Aim:- CategoricalData

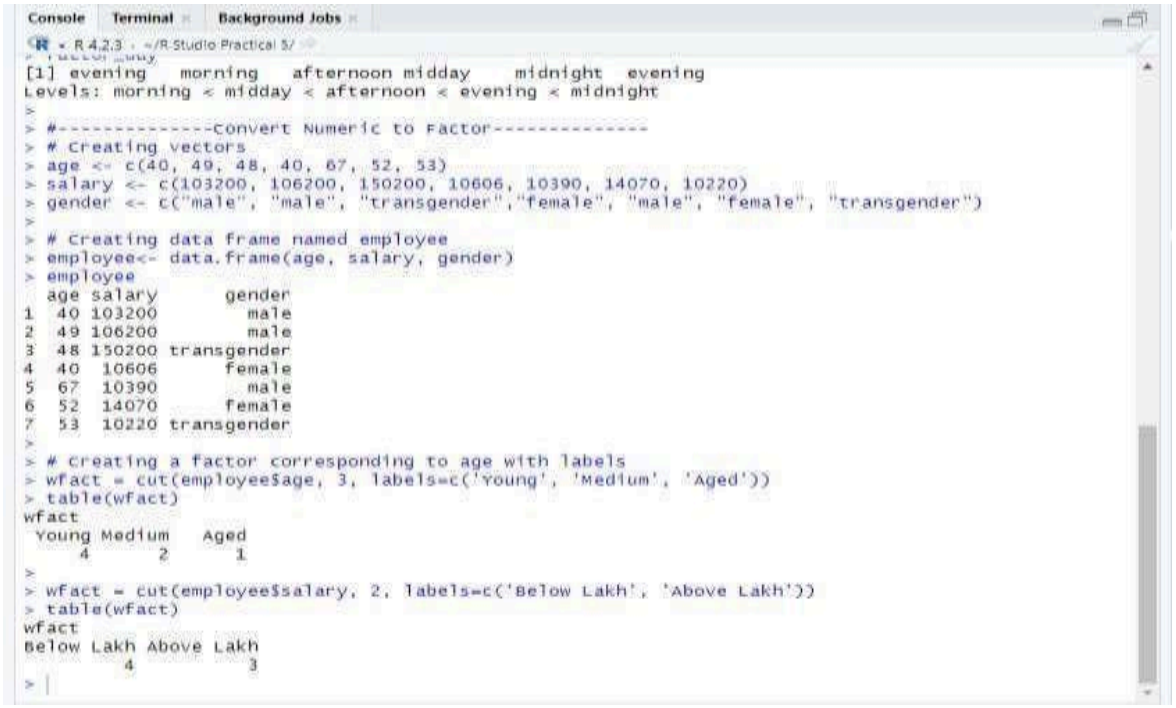
Code:-

```
#** Dealing With Categorical Data **

## The function factor is used to encode a vector as

#a factor (the terms ‘category’ and ‘enumerated type’ are also used for factors).

#If argument ordered is TRUE, the factor levels are assumed to be ordered. ## Factors are variables in R which take on a limited number of different values;

#such variables are often referred to as categorical variables.

# Create gender vector gender_vector <- c("Male", "Female", "Female", "Male", "Male") class(gender_vector)

#Convert Character into Factor(categorical data)  factor_gender_vector <-factor(gender_vector) class(factor_gender_vector) # Create Ordinal categorical vector day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')

# Convert 'day_vector' to a factor with ordered level

factor_day <- factor(day_vector, order = TRUE, levels =c('morning', 'midday', 'afternoon', 'evening', 'midnight')) # Print the factor_day factor_day

#--------------Convert Numeric to Factor---------------

# Creating vectors    age <- c(40, 49, 48, 40, 67, 52, 53) salary <- c(103200, 106200, 150200, 10606, 10390, 14070, 10220) gender <- c("male", "male", "transgender","female", "male", "female", "transgender")

# Creating data frame named employee  employee<- data.frame(age, salary, gender) employee

# Creating a factor corresponding to age with labels wfact = cut(employee$age, 3, labels=c('Young', 'Medium', 'Aged')) table(wfact) wfact = cut(employee$salary, 2, labels=c('Below Lakh', 'Above Lakh')) table(wfact)
```

Output:-



```
Console  Terminal  Background Jobs
R · R 4.2.3 · ~/R Studio Practical 5/
[Workspace loaded from ~/R Studio Practical 5/.RData]

> #"" Dealing With Categorical Data ""
>
> ## The function factor is used to encode a vector as
> #a factor (the terms 'category' and 'enumerated type' are also used for factors).
> #If argument ordered is TRUE, the factor levels are assumed to be ordered.
> ## Factors are variables in R which take on a limited number of different values;
> #such variables are often referred to as categorical variables.
>
> # Create gender vector
> gender_vector <- c("Male", "Female", "Female", "Male", "Male")
> class(gender_vector)
[1] "character"
>
> #Convert Character into Factor(categorical data)
> factor_gender_vector <-factor(gender_vector)
> class(factor_gender_vector)
[1] "factor"
>
> # Create Ordinal categorical vector
> day_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')
>
> # Convert 'day_vector' to a factor with ordered level
> factor_day <- factor(day_vector, order = TRUE, levels =c('morning', 'midday', 'afternoon', 'eveni
ng', 'midnight'))
>
> # Print the factor_day
> factor_day
[1] evening   morning   afternoon midday    midnight  evening
Levels: morning < midday < afternoon < evening < midnight
>
> #--------------Convert Numeric to Factor--------------
> # Creating vectors
> age <- c(40, 49, 48, 40, 67, 52, 53)
```

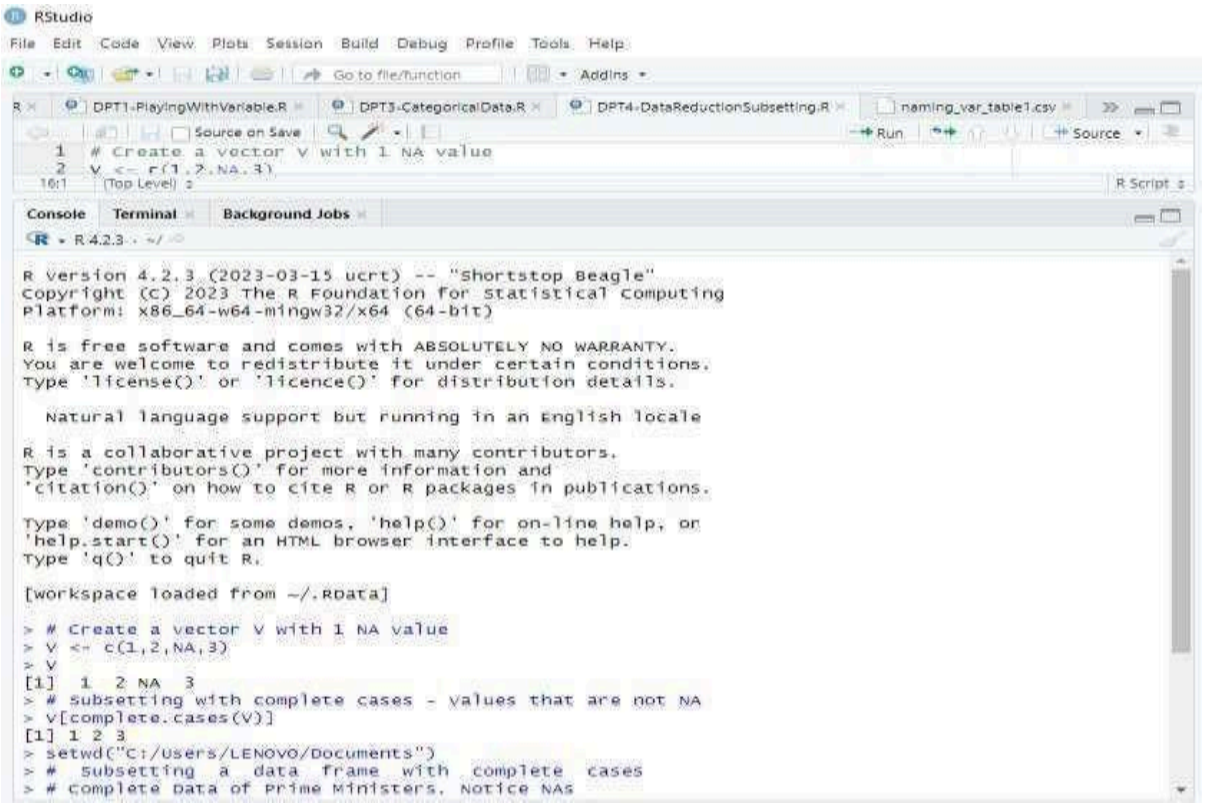


```
Console  Terminal  Background Jobs
R · R 4.2.3 · ~/R Studio Practical 5/
[1] evening   morning   afternoon midday    midnight  evening
Levels: morning < midday < afternoon < evening < midnight
>
> #--------------Convert Numeric to Factor--------------
> # Creating vectors
> age <- c(40, 49, 48, 40, 67, 52, 53)
> salary <- c(103200, 106200, 150200, 10606, 10390, 14070, 10220)
> gender <- c("male", "male", "transgender","female", "male", "female", "transgender")
>
> # Creating data frame named employee
> employee<- data.frame(age, salary, gender)
> employee
  age salary      gender
1  40 103200        male
2  49 106200        male
3  48 150200 transgender
4  40  10606      female
5  67  10390        male
6  52  14070      female
7  53  10220 transgender
>
> # Creating a factor corresponding to age with labels
> wfact = cut(employee$age, 3, labels=c('Young', 'Medium', 'Aged'))
> table(wfact)
wfact
 Young Medium   Aged
     4      2      1
>
> wfact = cut(employee$salary, 2, labels=c('Below Lakh', 'Above Lakh'))
> table(wfact)
wfact
Below Lakh Above Lakh
         4          3
>
```

Aim:- Data ReductionSubsetting Code:-

```
# Create a vector V with 1 NA value

V <- c(1,2,NA,3)

V

# Subsetting with complete cases - values that are not NA V[complete.cases(V)]
setwd("C:/Users/LENOVO/Documents")

# Subsetting a data frame with complete cases # Complete Data of Prime Ministers. Notice NAs dataC <-
read.csv(file ="naming_var_table1.csv") dataC dataCompleteCases <- dataC[complete.cases(dataC),]
dataCompleteCases
```

Output:-



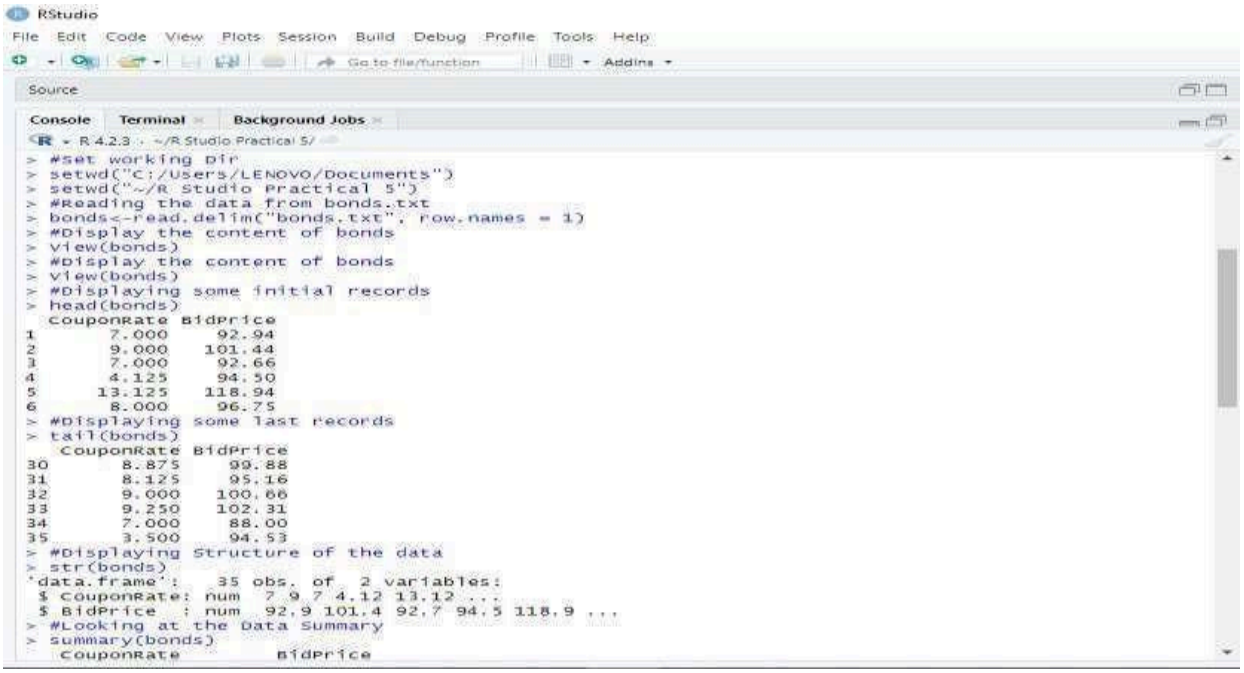
7.Implementation and analysis of Linear regression.

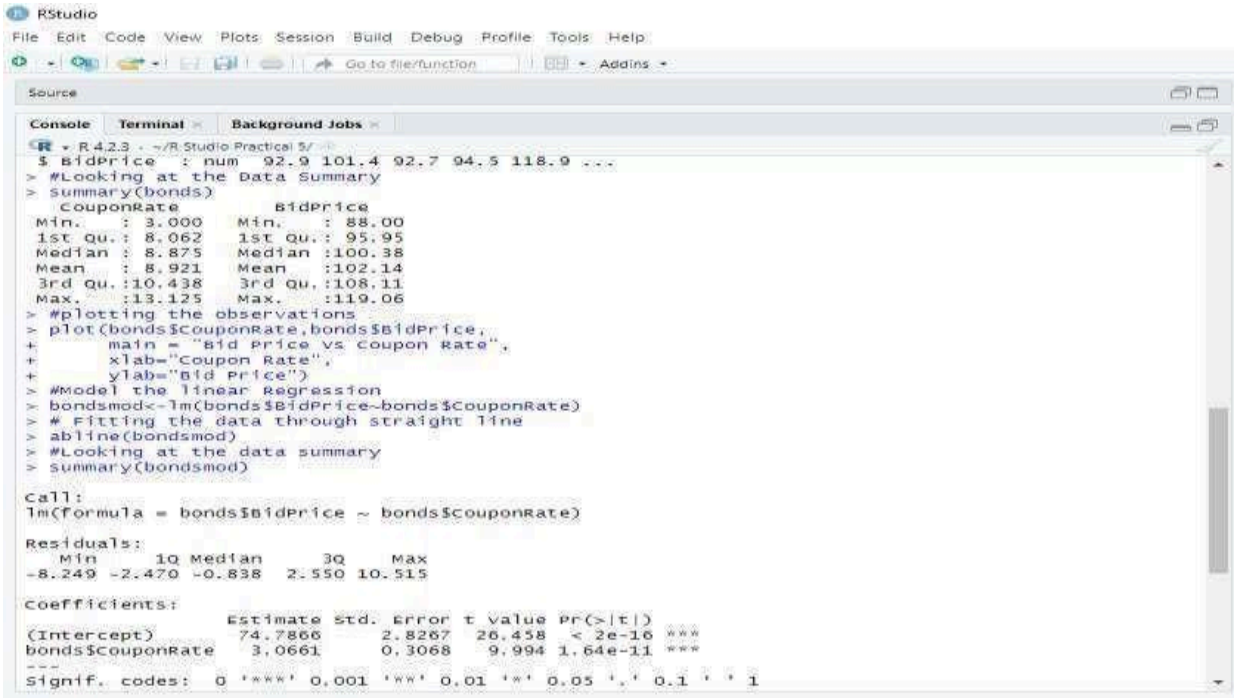
Linear Regression is a statistical method used to model the relationship between a dependent variable (also known as the target or response variable) and one or more independent variables (also known as predictors or features). The goal of linear regression is to fit a linear equation to observed data, so that the dependent variable can be predicted based on the values of the independent variables.

Code:-

```
#Set working Dir setwd("C:/Users/LENOVO/Documents") #Reading the data from bonds.txt
bonds<-read.delim("bonds.txt", row.names = 1)

#Display the content of bonds

View(bonds)

#Displaying some initial records  head(bonds)

#Displaying some last records tail(bonds)

#Displaying Structure of the data str(bonds)

#Looking at the Data Summary summary(bonds)

#plotting the observations plot(bonds$CouponRate,bonds$BidPrice,      main = "Bid Price Vs Coupon Rate",
xlab="Coupon Rate",      ylab="Bid Price") #Model the linear Regression
bondsmod<-lm(bonds$BidPrice~bonds$CouponRate) # Fitting the data through straight line abline(bondsmod)

#Looking at the data summary summary(bondsmod) alpha=0.05 n=35 p=1 qt(p=1-(alpha/2),df=n-p-1)

#Calculating F-Statistics

SSE<-sum((bonds$BidPrice-bondsmod$fitted.values)^2)

SSE

SSR<-sum((bondsmod$fitted.values-mean(bonds$BidPrice))^2) SSR n=35

(SSR/SSE)*(n-2)
```

Output:-


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.2.3 · ~/R Studio Practical 5/
> #set working Dir
> setwd("C:/Users/LENOVO/Documents")
> setwd("~/R Studio Practical 5")
> #Reading the data from bonds.txt
> bonds<-read.delim("bonds.txt", row.names = 1)
> #Display the content of bonds
> view(bonds)
> #Display the content of bonds
> view(bonds)
> #Displaying some initial records
> head(bonds)
couponRate BidPrice
1 7.000 92.94
2 9.000 101.44
3 7.000 92.66
4 4.125 94.50
5 13.125 118.94
6 8.000 96.75
> #Displaying some last records
> tail(bonds)
couponRate BidPrice
30 8.875 99.88
31 8.125 95.16
32 9.000 100.66
33 9.250 102.31
34 7.000 88.00
35 3.500 94.53
> #Displaying Structure of the data
> str(bonds)
'data.frame': 35 obs. of 2 variables:
$ couponRate: num 7 9 7 4.12 13.12 ...
$ BidPrice : num 92.9 101.4 92.7 94.5 118.9 ...
> #Looking at the Data Summary
> summary(bonds)
couponRate BidPrice


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.2.3 · ~/R Studio Practical 5/
$ BidPrice : num 92.9 101.4 92.7 94.5 118.9 ...
> #Looking at the Data Summary
> summary(bonds)
couponRate BidPrice
Min. : 3.000 Min. : 88.00
1st Qu.: 8.062 1st Qu.: 95.95
Median : 8.875 Median :100.38
Mean : 8.921 Mean :102.14
3rd Qu.:10.438 3rd Qu.:108.11
Max. :13.125 Max. :119.06
> #plotting the observations
> plot(bonds$couponRate,bonds$BidPrice,
+ main = "Bid Price vs coupon Rate",
+ xlab="Coupon Rate",
+ ylab="Bid Price")
> #Model the linear Regression
> bondsmod<-lm(bonds$BidPrice~bonds$couponRate)
> # Fitting the data through straight line
> abline(bondsmod)
> #Looking at the data summary
> summary(bondsmod)
Call:
lm(formula = bonds$BidPrice ~ bonds$couponRate)
Residuals:
Min 1Q Median 3Q Max
-8.249 -2.470 -0.838 2.550 10.515
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 74.7866 2.8267 26.458 < 2e-16 ***
bonds$couponRate 3.0661 0.3068 9.994 1.64e-11 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

RStudio

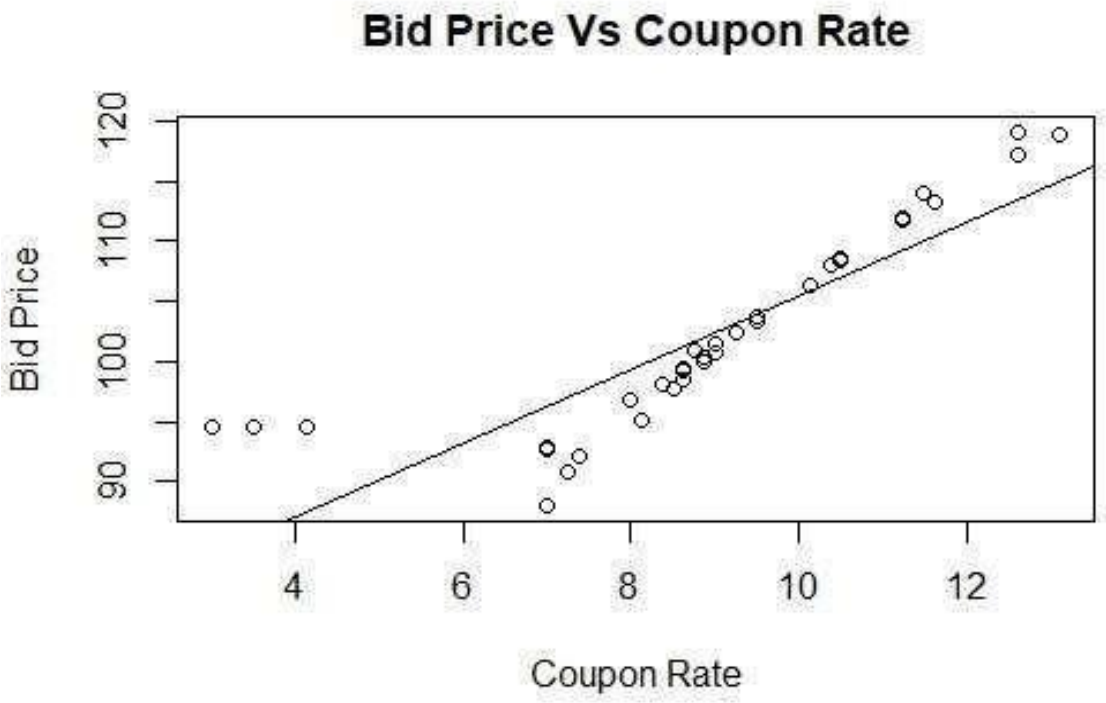
File Edit Code View Plots Session Build Debug Profile Tools Help

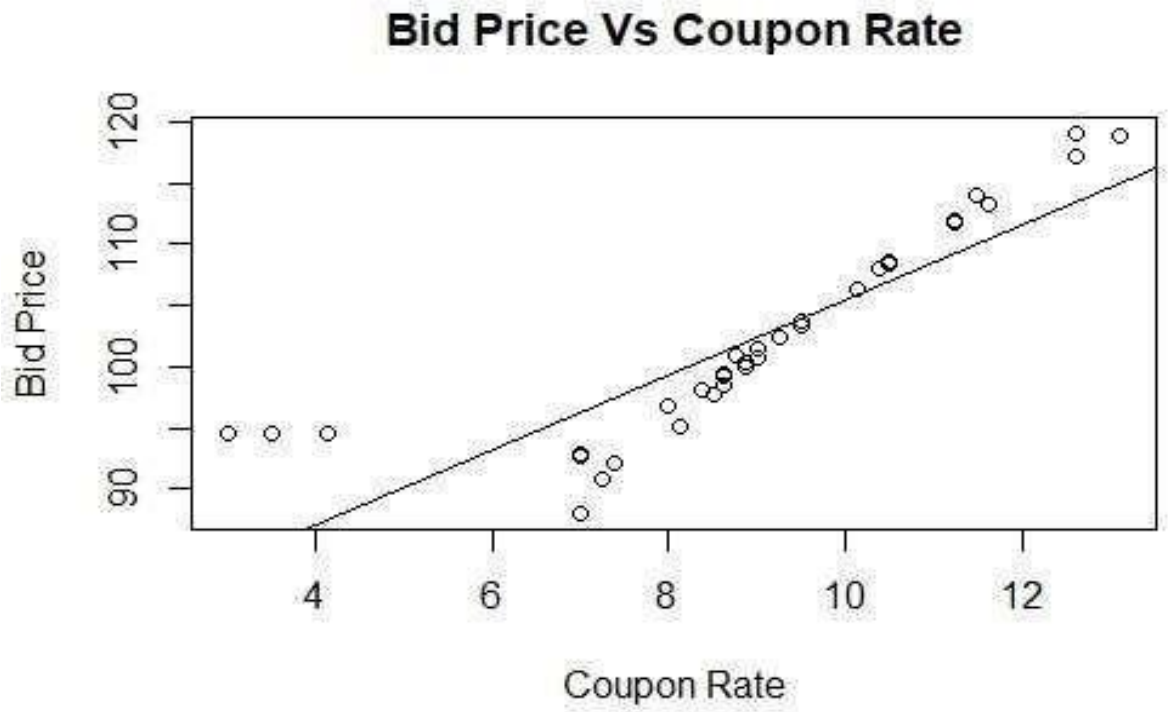
Source

Console Terminal Background Jobs

R 4.2.3 · ~/R Studio Practical 5/

```
Call:
lm(formula = bonds$BidPrice ~ bonds$CouponRate)

Residuals:
    Min      1Q  Median      3Q     Max
-8.249 -2.470 -0.838   2.550  10.515

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      74.7866     2.8267  26.458  < 2e-16 ***
bonds$CouponRate  3.0661     0.3068   9.994 1.64e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.175 on 33 degrees of freedom
Multiple R-squared:  0.7516,    Adjusted R-squared:  0.7441
F-statistic: 99.87 on 1 and 33 DF,  p-value: 1.645e-11

> alpha=0.05
> n=35
> p=1
> qt(p=1-(alpha/2),df=n-p-1)
[1] 2.034515
> #Calculating F-Statistics
> SSE<-sum((bonds$BidPrice-bondsmod$fitted.values)^2)
> SSE
[1] 575.3418
> SSR<-sum((bondsmod$fitted.values-mean(bonds$BidPrice))^2)
> SSR
[1] 1741.263
> n=35
> (SSR/SSE)*(n-2)
[1] 99.87401
>
```


Bid Price Vs Coupon Rate
Bid Price
90
100
110
120
4
6
8
10
12
Coupon Rate


Bid Price Vs Coupon Rate
Bid Price
90
100
110
120
4
6
8
10
12
Coupon Rate

8 Implementation and analysis of Classification algorithms - Naive Bayesian

Naive Bayes is a family of probabilistic classifiers based on **Bayes' Theorem**. It is called "naive" because it makes the **naive assumption** that all features (or variables) are independent of each other, given the class label. This assumption simplifies the computation of the likelihood of a class given the data, even though in real-world applications, features may often be correlated.

Code:- library(e1071) # Sample data data <- data.frame(age = c(25, 30, 45, 35, 50, 23, 37, 61, 22, 42), income = c('high', 'high', 'medium', 'low', 'low', 'medium', 'medium', 'high', 'low',

'medium'), student = c('no', 'no', 'no', 'yes', 'no', 'yes', 'yes', 'no', 'yes', 'yes'), credit_rating = c('fair', 'excellent', 'fair', 'fair', 'excellent', 'excellent', 'fair',

'fair', 'excellent', 'fair'), buys_computer = c('no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'yes', 'yes')

)

head(data) str(data)

Convert categorical variables to factors data$income <- as.factor(data$income) data$student <- as.factor(data$student) data$credit_rating <- as.factor(data$credit_rating) data$buys_computer <- as.factor(data$buys_computer) str(data)

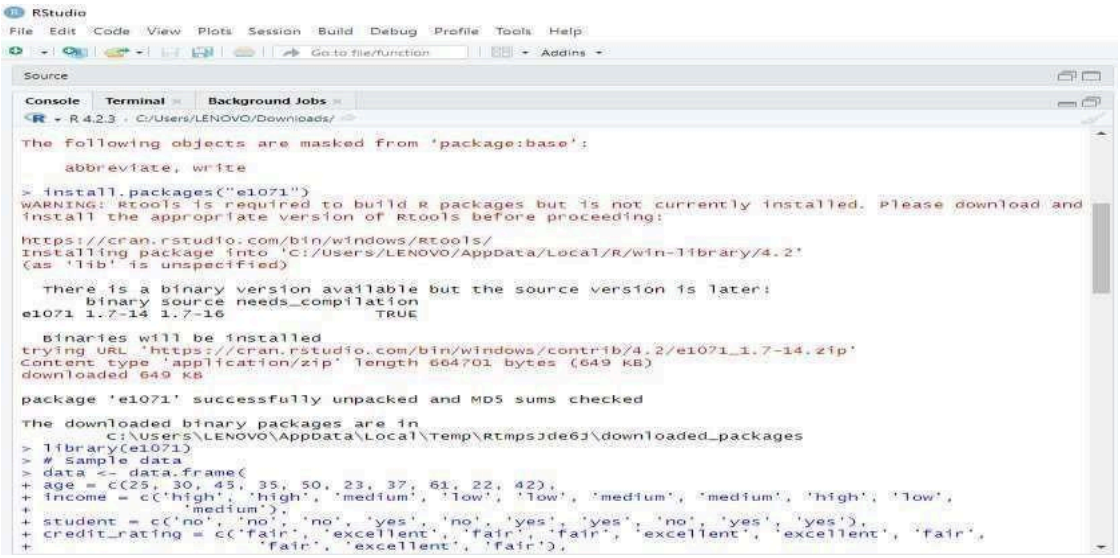
#Split the Data into Training and Testing Sets set.seed(123) # For reproducibility train_index <- sample(1:nrow(data), 0.7 * nrow(data)) train_data <- data[train_index,] test_data <- data[-train_index,] test_data

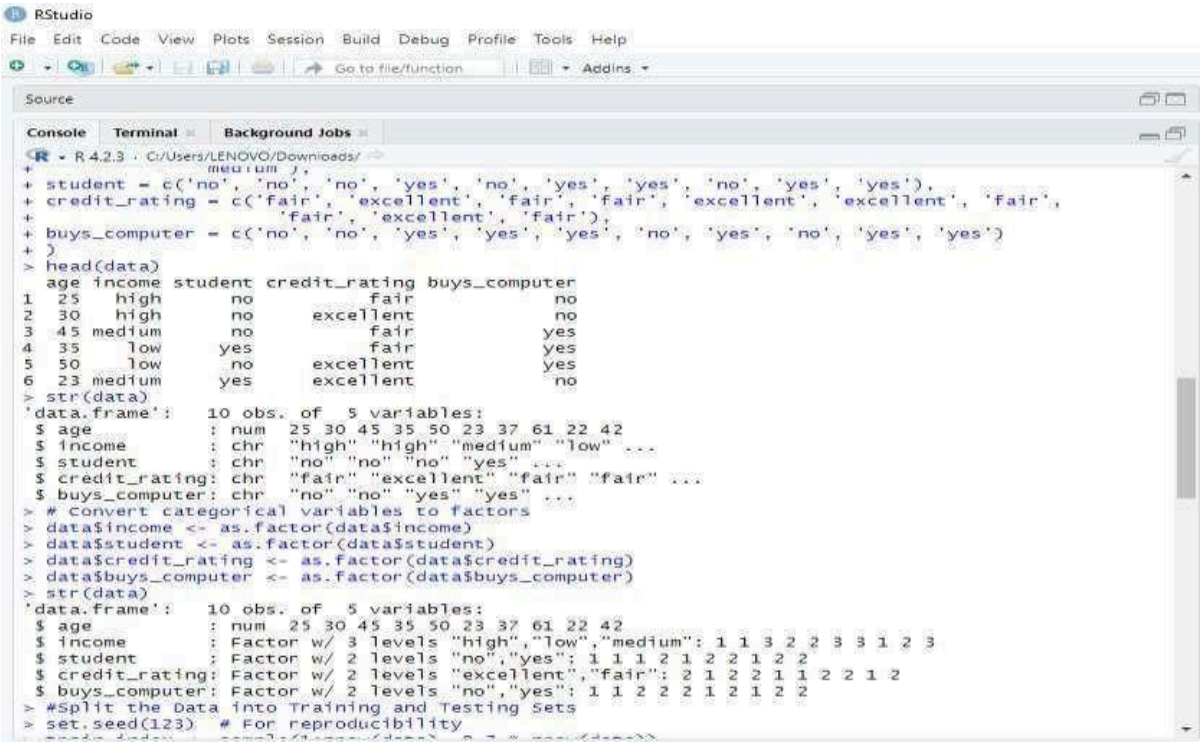
#Train the Naive Bayes Model model <- naiveBayes(buys_computer ~ ., data = train_data) print(model)

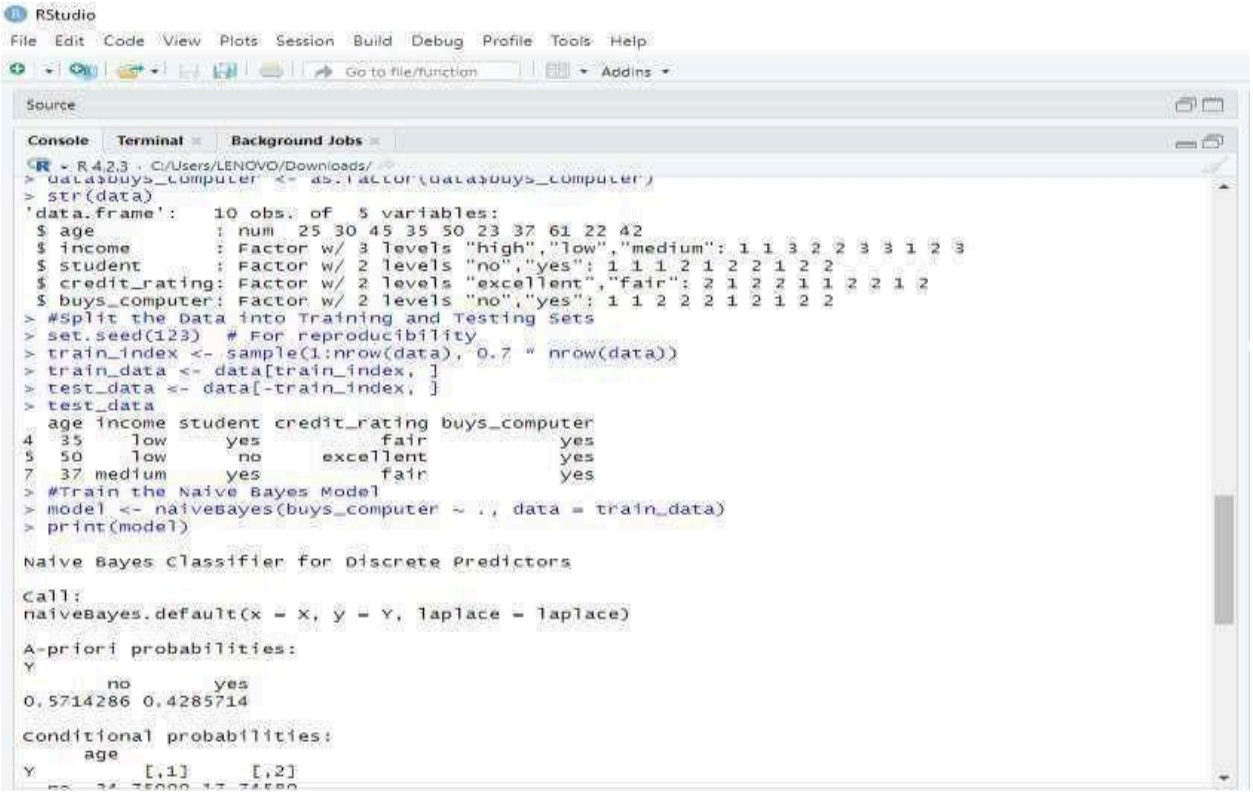
#Make Predictions

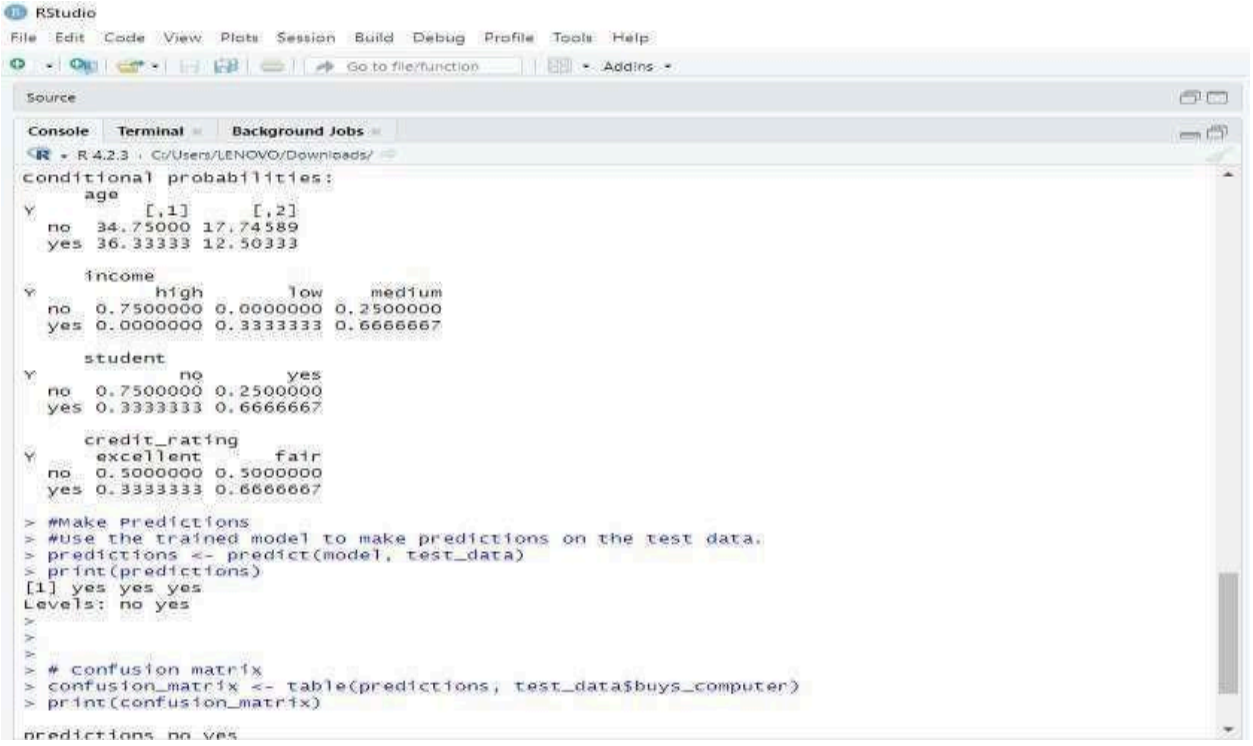
#Use the trained model to make predictions on the test data. predictions <- predict(model, test_data) print(predictions) # Confusion matrix confusion_matrix <- table(predictions, test_data$buys_computer) print(confusion_matrix) # Calculate accuracy accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix) print(paste("Accuracy:", round(accuracy, 2)))

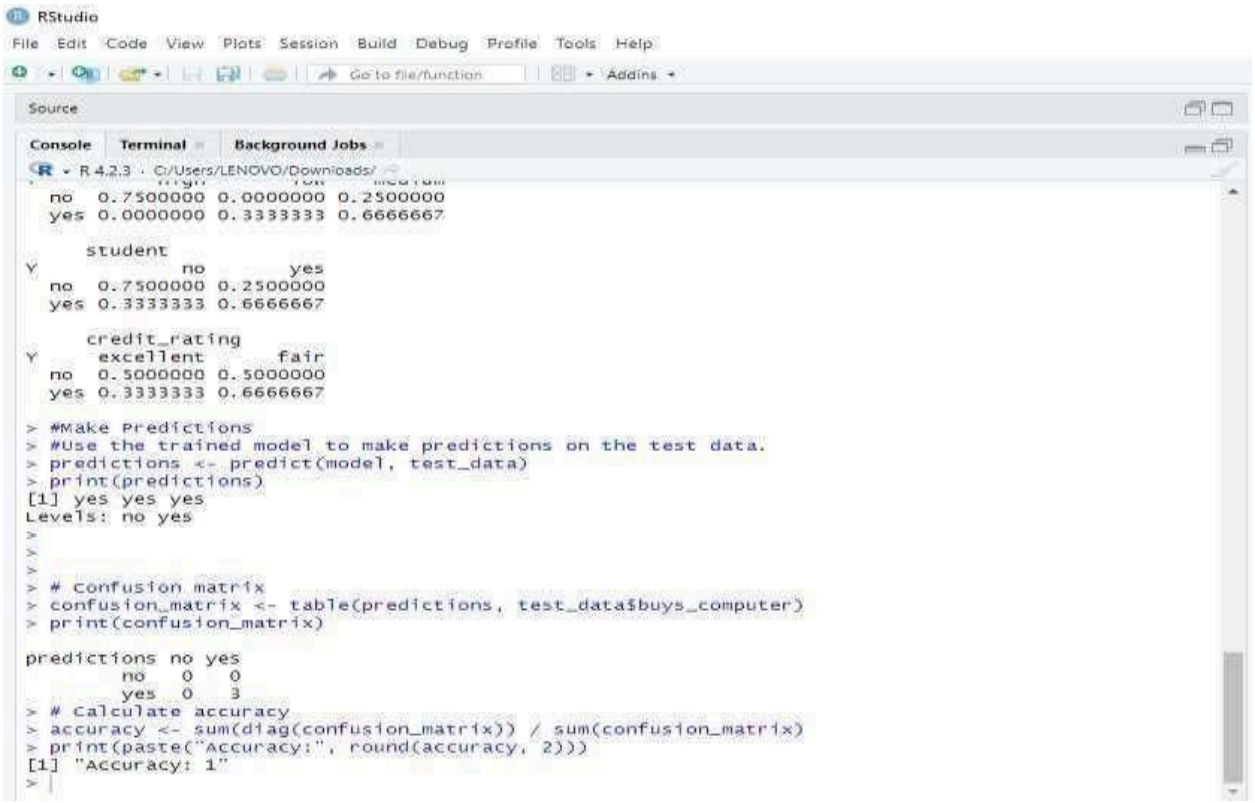
Output:-


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
The following objects are masked from 'package:base':
abbreviate, write
> install.packages("e1071")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:
https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/LENOVO/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)
There is a binary version available but the source version is later:
binary source needs_compilation
e1071 1.7-14 1.7-16 TRUE
Binaries will be installed
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/e1071_1.7-14.zip'
Content type 'application/zip' length 664701 bytes (649 KB)
downloaded 649 KB
package 'e1071' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\LENOVO\AppData\Local\Temp\RtmpsJde6J\downloaded_packages
> library(e1071)
> # Sample data
> data <- data.frame(
+ age = c(25, 30, 45, 35, 50, 23, 37, 61, 22, 42),
+ income = c('high', 'high', 'medium', 'low', 'low', 'medium', 'medium', 'high', 'low',
+ 'medium'),
+ student = c('no', 'no', 'no', 'yes', 'no', 'yes', 'yes', 'no', 'yes', 'yes'),
+ credit_rating = c('fair', 'excellent', 'fair', 'fair', 'excellent', 'excellent', 'fair',
+ 'fair', 'excellent', 'fair'),


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
+ medium'),
+ student = c('no', 'no', 'no', 'yes', 'no', 'yes', 'yes', 'no', 'yes', 'yes'),
+ credit_rating = c('fair', 'excellent', 'fair', 'fair', 'excellent', 'excellent', 'fair',
+ 'fair', 'excellent', 'fair'),
+ buys_computer = c('no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'yes', 'yes')
+)
> head(data)
age income student credit_rating buys_computer
1 25 high no fair no
2 30 high no excellent no
3 45 medium no fair yes
4 35 low yes fair yes
5 50 low no excellent yes
6 23 medium yes excellent no
> str(data)
'data.frame': 10 obs. of 5 variables:
$ age : num 25 30 45 35 50 23 37 61 22 42
$ income : chr "high" "high" "medium" "low" ...
$ student : chr "no" "no" "no" "yes" ...
$ credit_rating: chr "fair" "excellent" "fair" "fair" ...
$ buys_computer: chr "no" "no" "yes" "yes" ...
> # Convert categorical variables to factors
> data$income <- as.factor(data$income)
> data$student <- as.factor(data$student)
> data$credit_rating <- as.factor(data$credit_rating)
> data$buys_computer <- as.factor(data$buys_computer)
> str(data)
'data.frame': 10 obs. of 5 variables:
$ age : num 25 30 45 35 50 23 37 61 22 42
$ income : Factor w/ 3 levels "high","low","medium": 1 1 3 2 2 3 3 1 2 3
$ student : Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 2 1 2 2
$ credit_rating: Factor w/ 2 levels "excellent","fair": 2 1 2 2 1 1 2 2 1 2
$ buys_computer: Factor w/ 2 levels "no","yes": 1 1 2 2 2 1 2 1 2 2
> #Split the Data into Training and Testing Sets
> set.seed(123) # For reproducibility


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
> data$buys_computer <- as.factor(data$buys_computer)
> str(data)
'data.frame': 10 obs. of 5 variables:
$ age : num 25 30 45 35 50 23 37 61 22 42
$ income : Factor w/ 3 levels "high","low","medium": 1 1 3 2 2 3 3 1 2 3
$ student : Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 2 1 2 2
$ credit_rating: Factor w/ 2 levels "excellent","fair": 2 1 2 2 1 1 2 2 1 2
$ buys_computer: Factor w/ 2 levels "no","yes": 1 1 2 2 2 1 2 1 2 2
> #Split the Data into Training and Testing Sets
> set.seed(123) # For reproducibility
> train_index <- sample(1:nrow(data), 0.7 * nrow(data))
> train_data <- data[train_index,]
> test_data <- data[-train_index,]
> test_data
age income student credit_rating buys_computer
4 35 low yes fair yes
5 50 low no excellent yes
7 37 medium yes fair yes
> #Train the Naive Bayes Model
> model <- naiveBayes(buys_computer ~ ., data = train_data)
> print(model)
Naive Bayes Classifier for Discrete Predictors
Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)
A-priori probabilities:
Y
no yes
0.5714286 0.4285714
Conditional probabilities:
age
Y [,1] [,2]


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
Conditional probabilities:
age
Y [,1] [,2]
no 34.75000 17.74589
yes 36.33333 12.50333
income
Y high low medium
no 0.7500000 0.0000000 0.2500000
yes 0.0000000 0.3333333 0.6666667
student
Y no yes
no 0.7500000 0.2500000
yes 0.3333333 0.6666667
credit_rating
Y excellent fair
no 0.5000000 0.5000000
yes 0.3333333 0.6666667
> #Make Predictions
> #Use the trained model to make predictions on the test data.
> predictions <- predict(model, test_data)
> print(predictions)
[1] yes yes yes
Levels: no yes
> # Confusion matrix
> confusion_matrix <- table(predictions, test_data$buys_computer)
> print(confusion_matrix)
predictions no yes



9 Aim: Implementation and analysis of Classification algorithms - K-Nearest Neighbor

K-Nearest Neighbors (K-NN) is a **non-parametric**, **instance-based** machine learning algorithm used for **classification** and **regression** tasks. It is one of the simplest and most intuitive algorithms for supervised learning. The core idea behind K-NN is that a data point's classification (or value, in the case of regression) can be determined by looking at the **k** closest training examples (neighbors) in the feature space.

Code:- library(xlsx) # SedanCar. xlsx df=read.xlsx(file.choose(),1,header = T) View(df) df=df[,!apply(is.na(df),2,all)] View(df)

str(df)

df$Ownership<-as.factor(df$Ownership)

str(df) head(df) plot(df$Annual.Income,df$Household.Area, las=1, xlab="Annual Income (a.'lakhs)", ylab = "Household Area 00s ft2", xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])

i=as.numeric(df$Ownership) legend("bottomright", inset = 0.005, c("non-owner","owner"), pch = c(21,19), cex=0.7, x.intersp=0.5, y.intersp=0.5)

#New Observation

#Annual_Income=6 lpa, Household Area=20 points(6,20, type="p",pch=4)

#Normalization dfb=df

str(dfb) df[,1:2]=scale(df[,1:2],center = T,scale = T) head(df)

#Partitioning- training(15) testing(5) partidx=sample(1:nrow(df),15, replace = F) partidx[1:15] #displaying the observations that has been selected in sampling process dftrain=df[partidx,] dftest=df[-partidx,]

View(dftest) #view test data sample

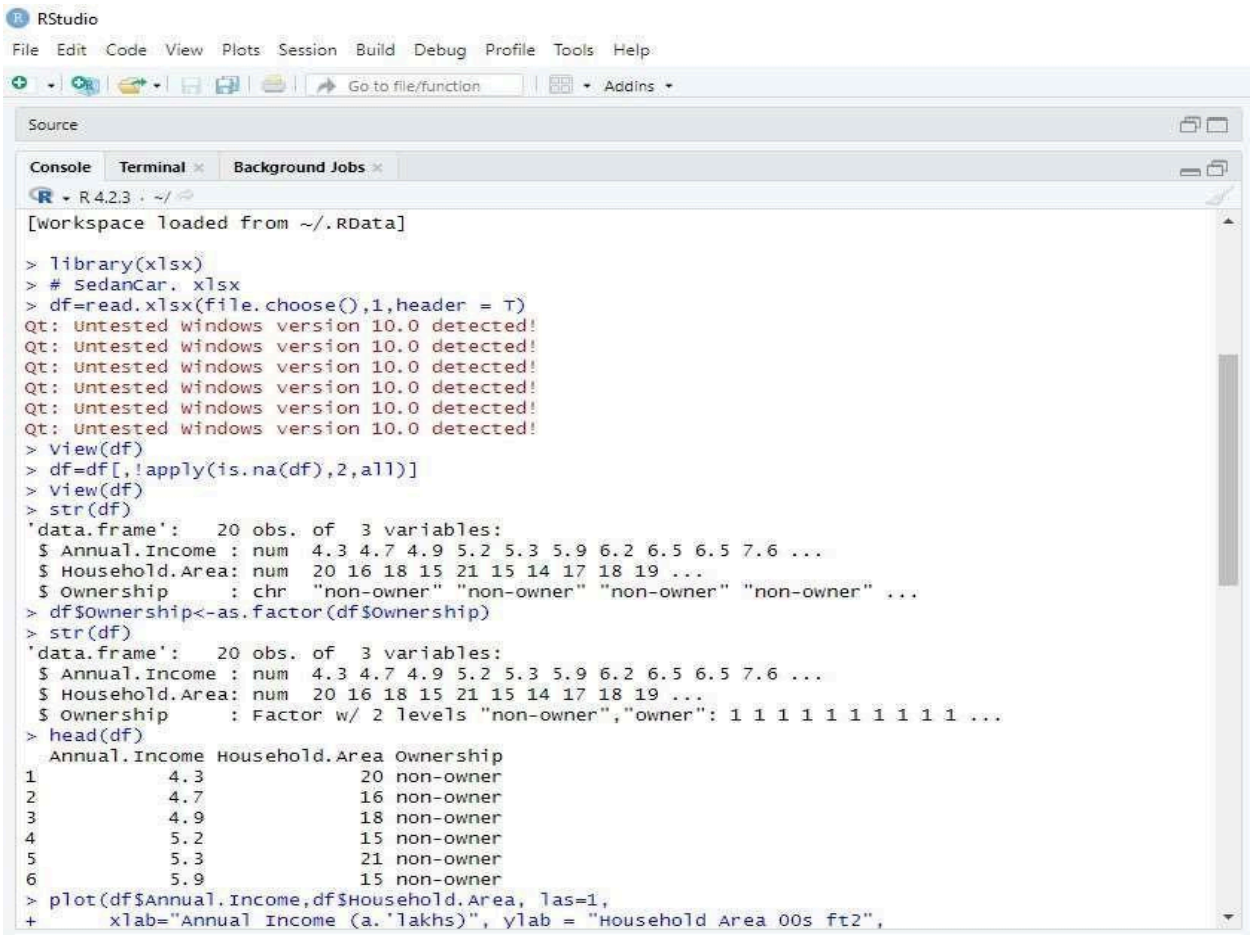
View(dftrain) #view training data sample

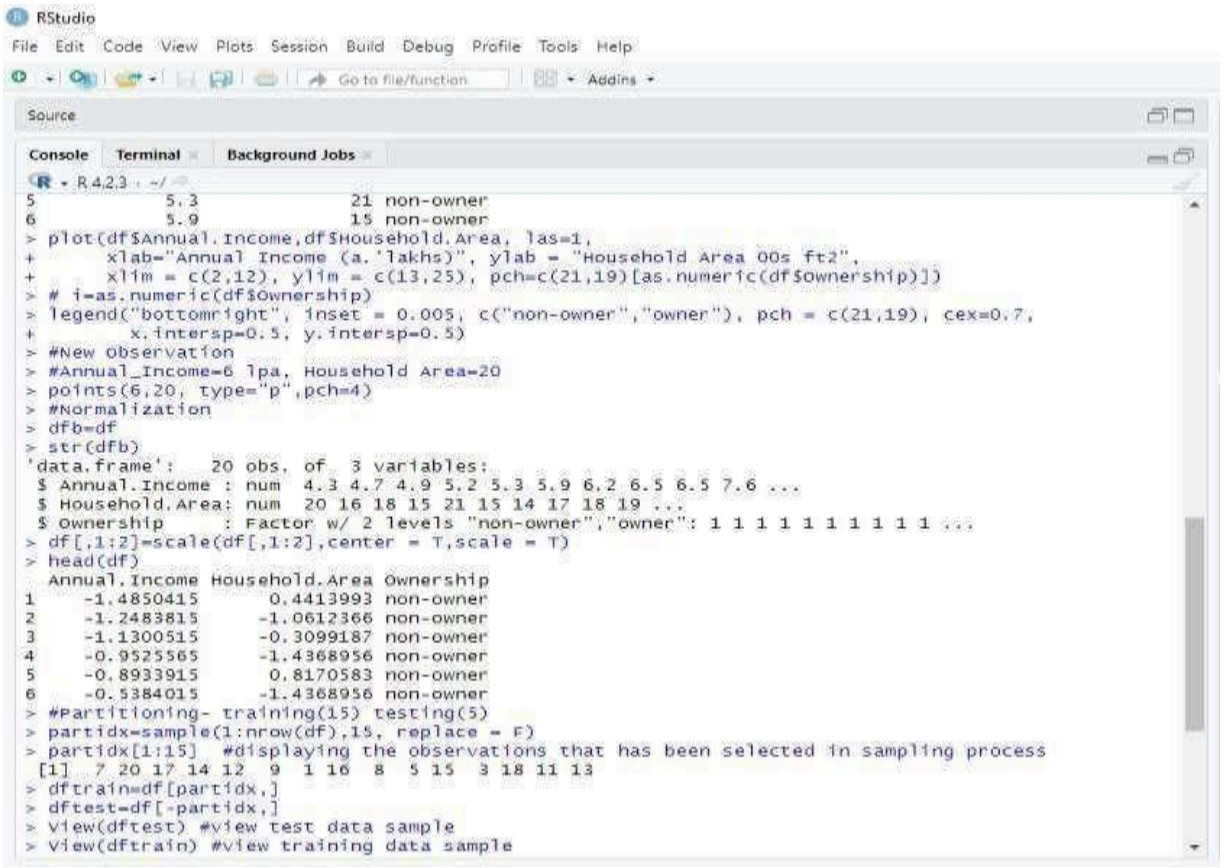
#Modeling

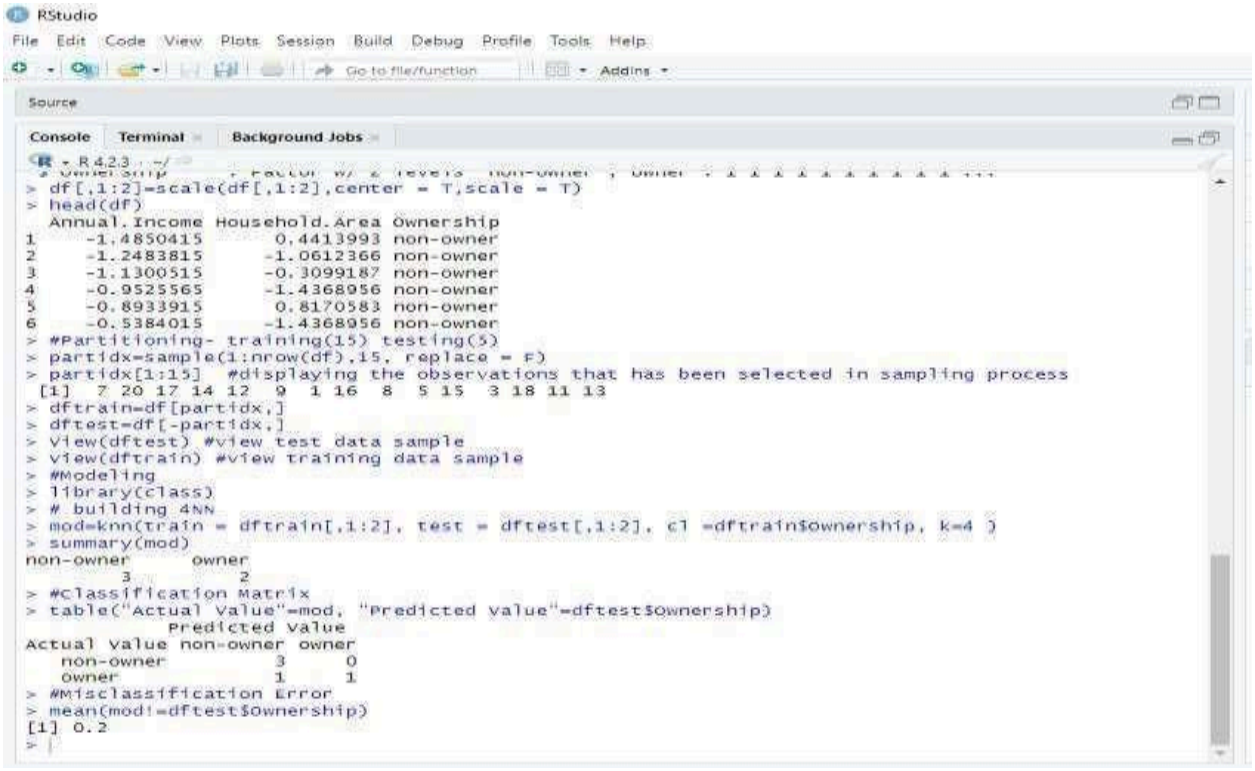
library(class) # building 4NN mod=knn(train = dftrain[,1:2], test = dftest[,1:2], cl =dftrain$Ownership, k=4) summary(mod)

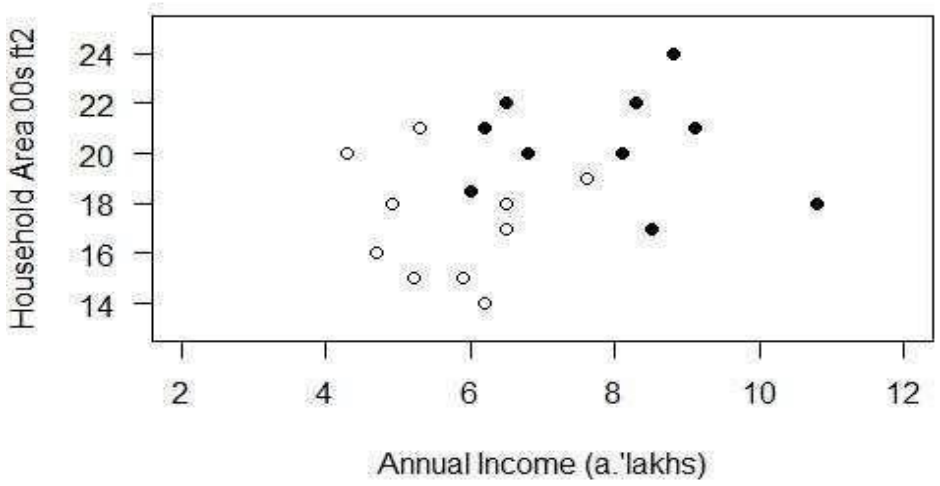
#Classification Matrix table("Actual Value"=mod, "Predicted Value"=dftest$Ownership)

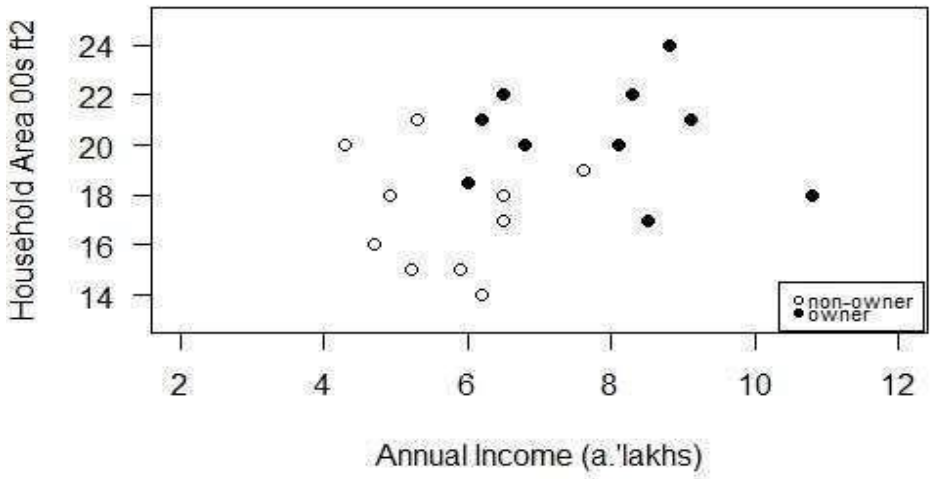
#Misclassification Error mean(mod!=dftest$Ownership) **Output:-**

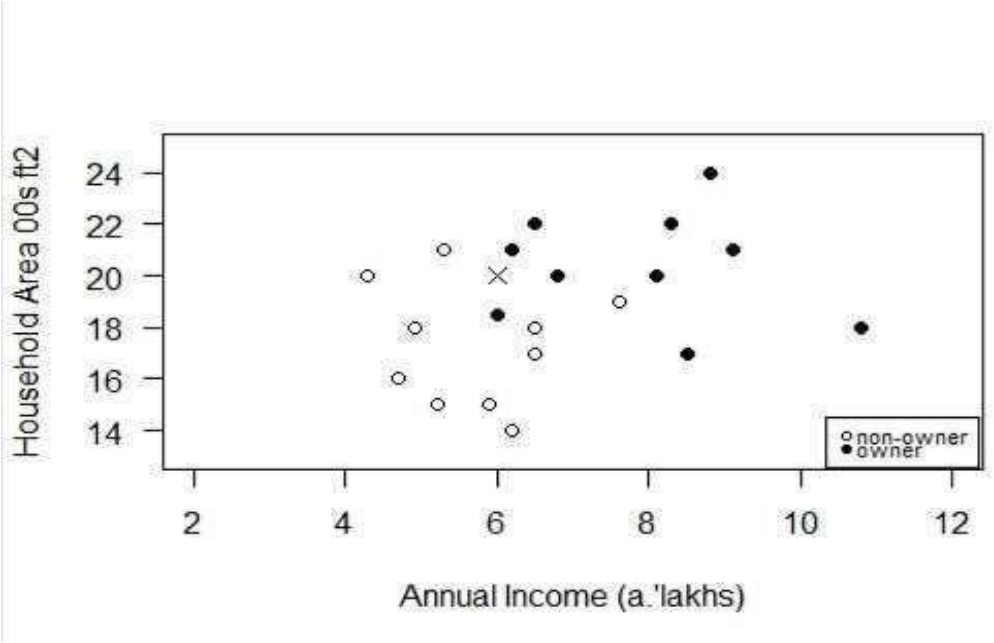



RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
5 5.3 21 non-owner
6 5.9 15 non-owner
> plot(df$Annual.Income,df$Household.Area, las=1,
+ xlab="Annual Income (a.'lakhs)", ylab = "Household Area 00s ft2",
+ xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
> # i=as.numeric(df$Ownership)
> legend("bottomright", inset = 0.005, c("non-owner","owner"), pch = c(21,19), cex=0.7,
+ x.intersp=0.5, y.intersp=0.5)
> #New Observation
> #Annual_Income=6 lpa, Household Area=20
> points(6,20, type="p",pch=4)
> #Normalization
> dfb=df
> str(dfb)
'data.frame': 20 obs. of 3 variables:
$ Annual.Income : num 4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
$ Household.Area: num 20 16 18 15 21 15 14 17 18 19 ...
$ Ownership : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> df[,1:2]=scale(df[,1:2],center = T,scale = T)
> head(df)
Annual.Income Household.Area Ownership
1 -1.4850415 0.4413993 non-owner
2 -1.2483815 -1.0612366 non-owner
3 -1.1300515 -0.3099187 non-owner
4 -0.9525565 -1.4368956 non-owner
5 -0.8933915 0.8170583 non-owner
6 -0.5384015 -1.4368956 non-owner
> #Partitioning- training(15) testing(5)
> partidx=sample(1:nrow(df),15, replace = F)
> partidx[1:15] #displaying the observations that has been selected in sampling process
[1] 7 20 17 14 12 9 1 16 8 5 15 3 18 11 13
> dftrain=df[partidx,]
> dftest=df[-partidx,]
> View(dftest) #view test data sample
> View(dftrain) #view training data sample


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
> df[,1:2]=scale(df[,1:2],center = T,scale = T)
> head(df)
Annual.Income Household.Area Ownership
1 -1.4850415 0.4413993 non-owner
2 -1.2483815 -1.0612366 non-owner
3 -1.1300515 -0.3099187 non-owner
4 -0.9525565 -1.4368956 non-owner
5 -0.8933915 0.8170583 non-owner
6 -0.5384015 -1.4368956 non-owner
> #Partitioning- training(15) testing(5)
> partidx=sample(1:nrow(df),15, replace = F)
> partidx[1:15] #displaying the observations that has been selected in sampling process
[1] 7 20 17 14 12 9 1 16 8 5 15 3 18 11 13
> dftrain=df[partidx,]
> dftest=df[-partidx,]
> View(dftest) #view test data sample
> View(dftrain) #view training data sample
> #Modeling
> library(class)
> # building KNN
> mod=knn(train = dftrain[,1:2], test = dftest[,1:2], cl =dftrain$Ownership, k=4)
> summary(mod)
non-owner owner
3 2
> #Classification Matrix
> table("Actual Value"=mod, "Predicted Value"=dftest$Ownership)
Predicted Value
Actual Value non-owner owner
non-owner 3 0
owner 1 1
> #Misclassification Error
> mean(mod!=dftest$Ownership)
[1] 0.2


Household Area 00s ft2
24
22
20
18
16
14
2
4
6
8
10
12
Annual Income (a.'lakhs)


Household Area 00s ft2
24
22
20
18
16
14
non-owner
owner
2
4
6
8
10
12
Annual Income (a.'lakhs)



10 Implementation and analysis of Apriori Algorithm using Market Basket Analysis.

Market Basket Analysis (MBA) is a data mining technique used to analyze co-occurrence behavior, specifically identifying patterns in the purchase behavior of customers. It seeks to uncover associations between items that are frequently bought together. The goal is to understand customer purchasing patterns and help businesses make decisions regarding product placement, promotions, inventory management, and targeted marketing strategies.

Code:- #Association

#Example: purchase of Mobile Phone Covers

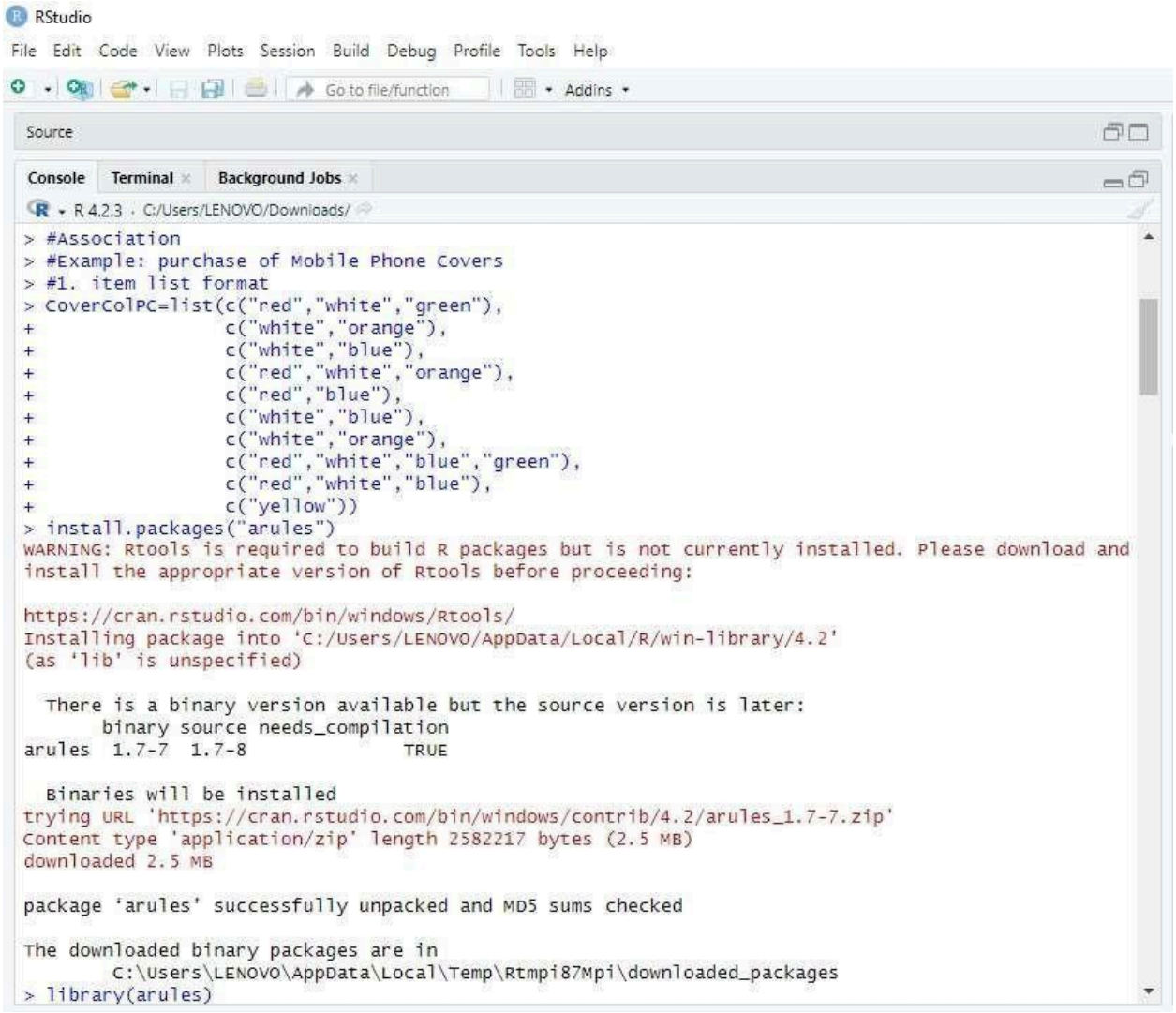
#1. item list format

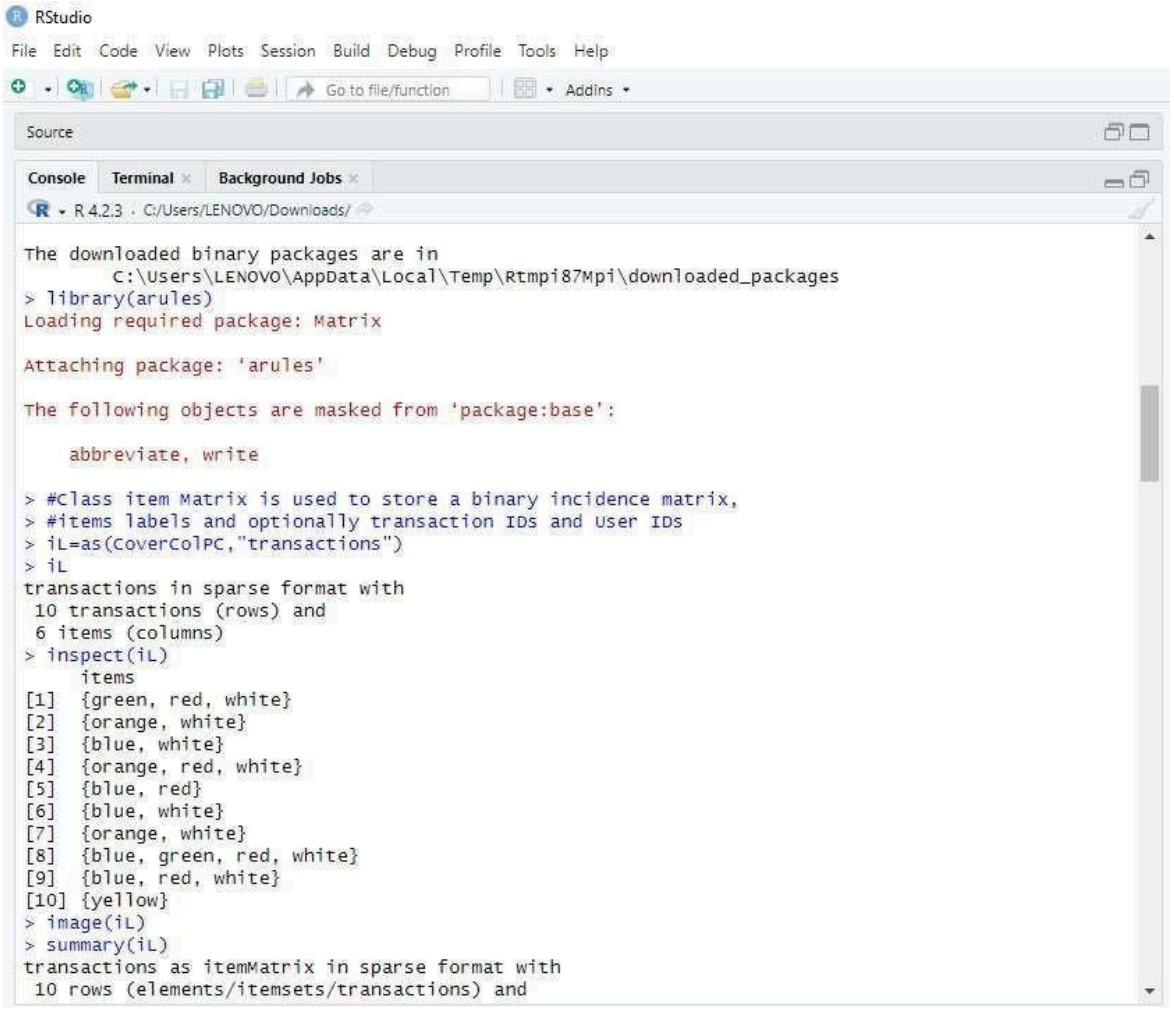
CoverColPC=list(c("red","white","green"),

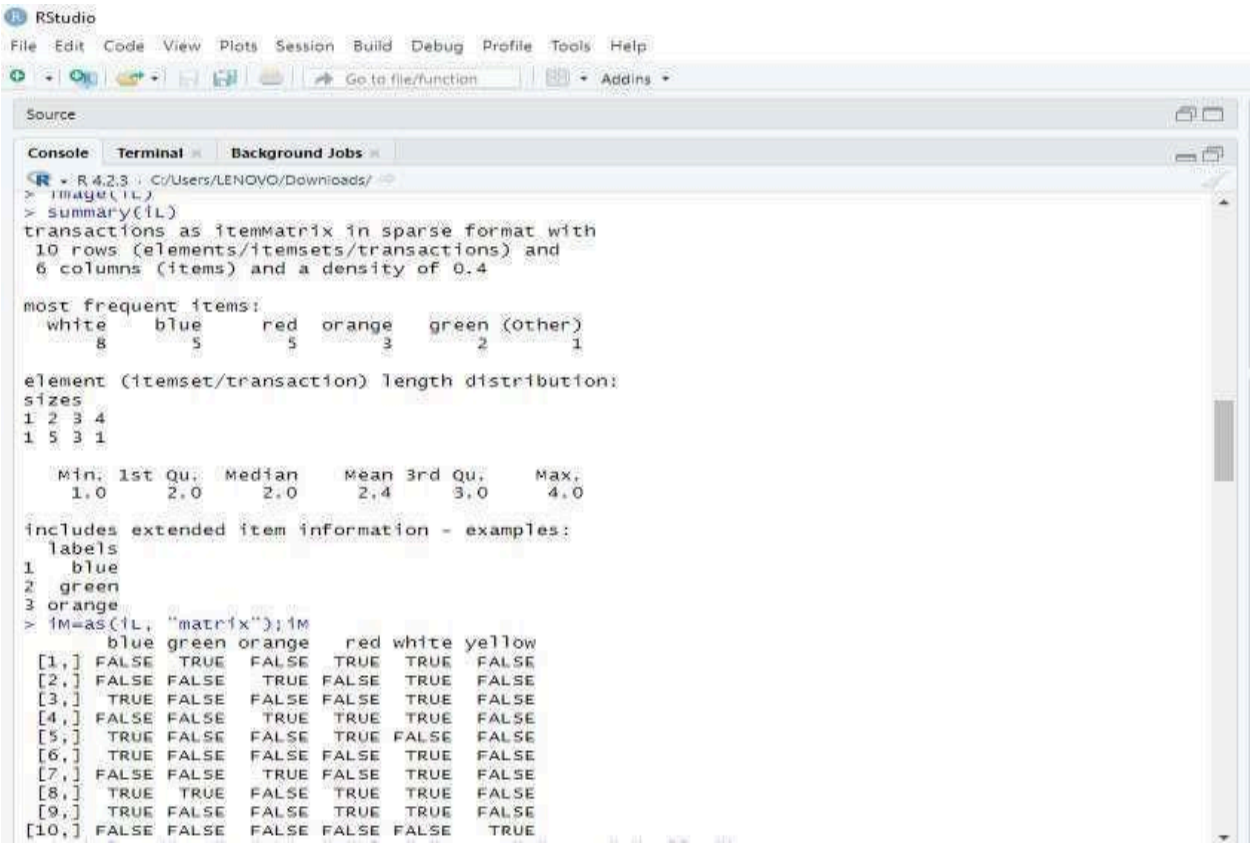
c("white","orange"), c("white","blue"), c("red","white","orange"), c("red","blue"), c("white","blue"), c("white","orange"), c("red","white","blue","green"), c("red","white","blue"), c("yellow")) install.packages("arules") library(arules)

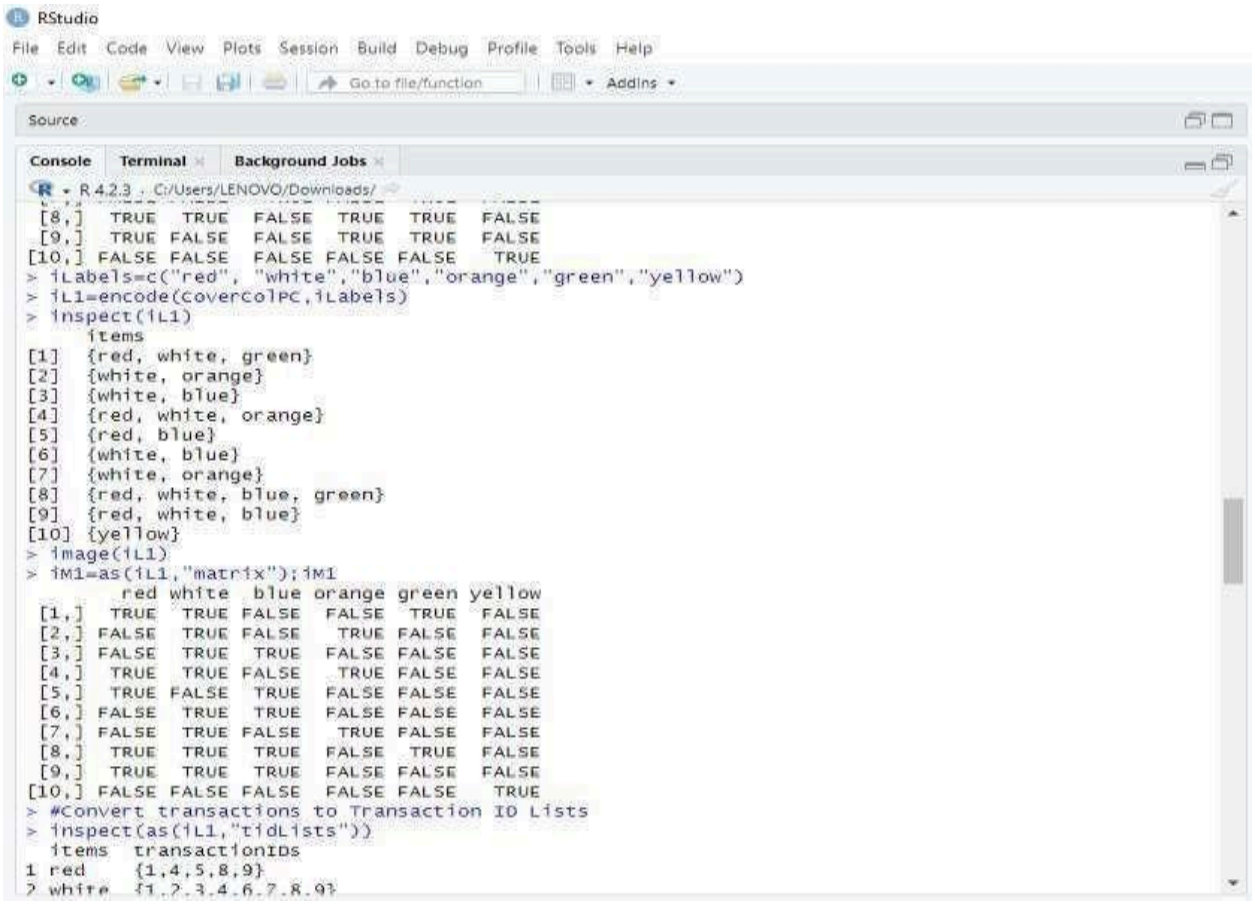
```
#Class item Matrix is used to store a binary incidence matrix, #items labels and optionally transaction IDs and User IDs iL=as(CoverColPC,"transactions") iL inspect(iL)

image(iL) summary(iL) iM=as(iL, "matrix");iM iLabels=c("red", "white","blue","orange","green","yellow") iL1=encode(CoverColPC,iLabels) inspect(iL1) image(iL1) iM1=as(iL1,"matrix");iM1

#Convert transactions to Transaction ID Lists inspect(as(iL1,"tidLists")) #2 Binary Matrix format iMn = apply(iM1,c(1,2),as.numeric) dimnames(iMn)=list(c(1:nrow(iM1)),itemLabels(iL1)); iMn

#Itemset with support (count)>=2 isets= eclat(iL1, parameter = list(support=0.2),control = list(verbose=F)) isets1=inspect(isets) isets1= isets1[order(size(isets)),];isets1

#minimum support=20% & minimum confidence=70% rules1=apriori(iL1,parameter = list(support=0.2,confidence=0.7)) rules1.sorted = sort(rules1, by="lift") inspect(rules1.sorted) round(quality(rules1.sorted),digits = 2) summary(rules1.sorted)
```

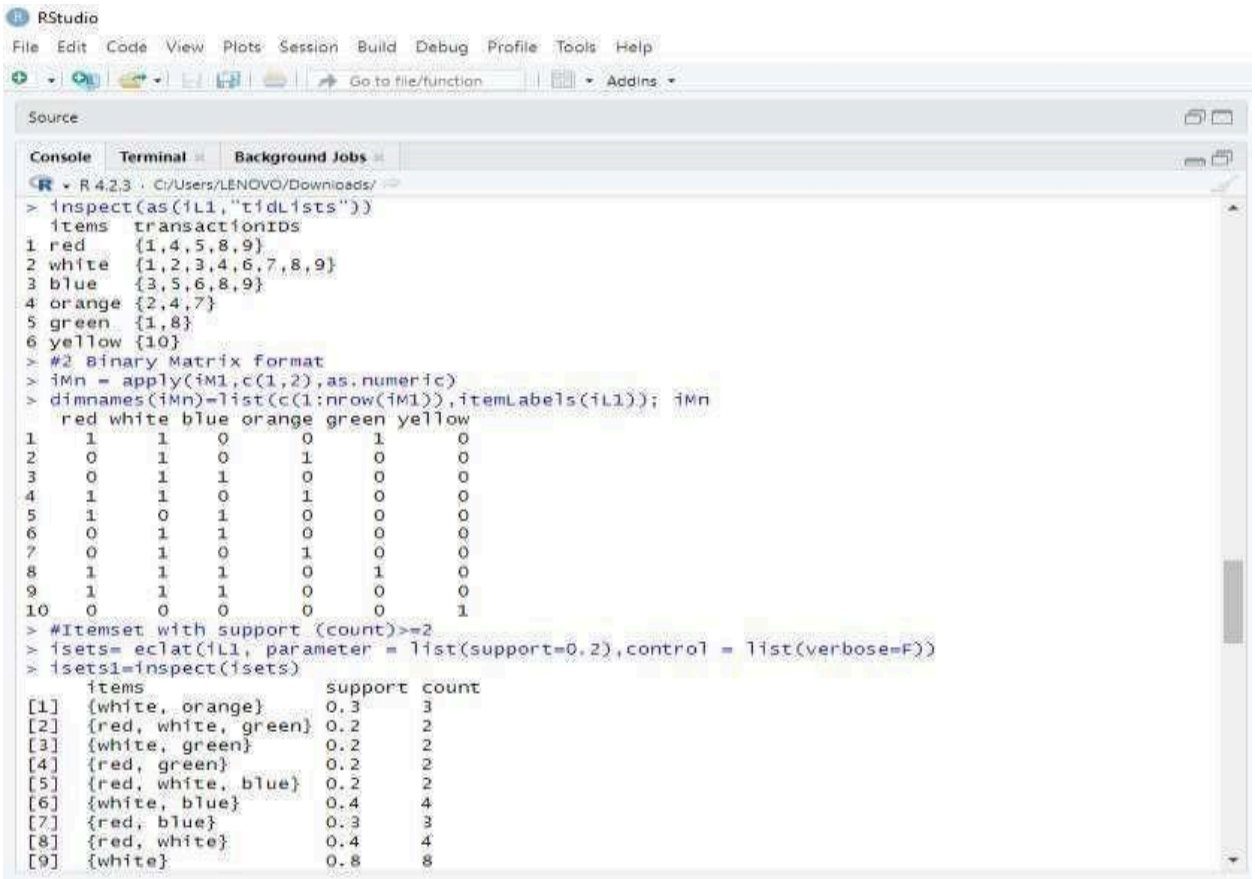
Output:-

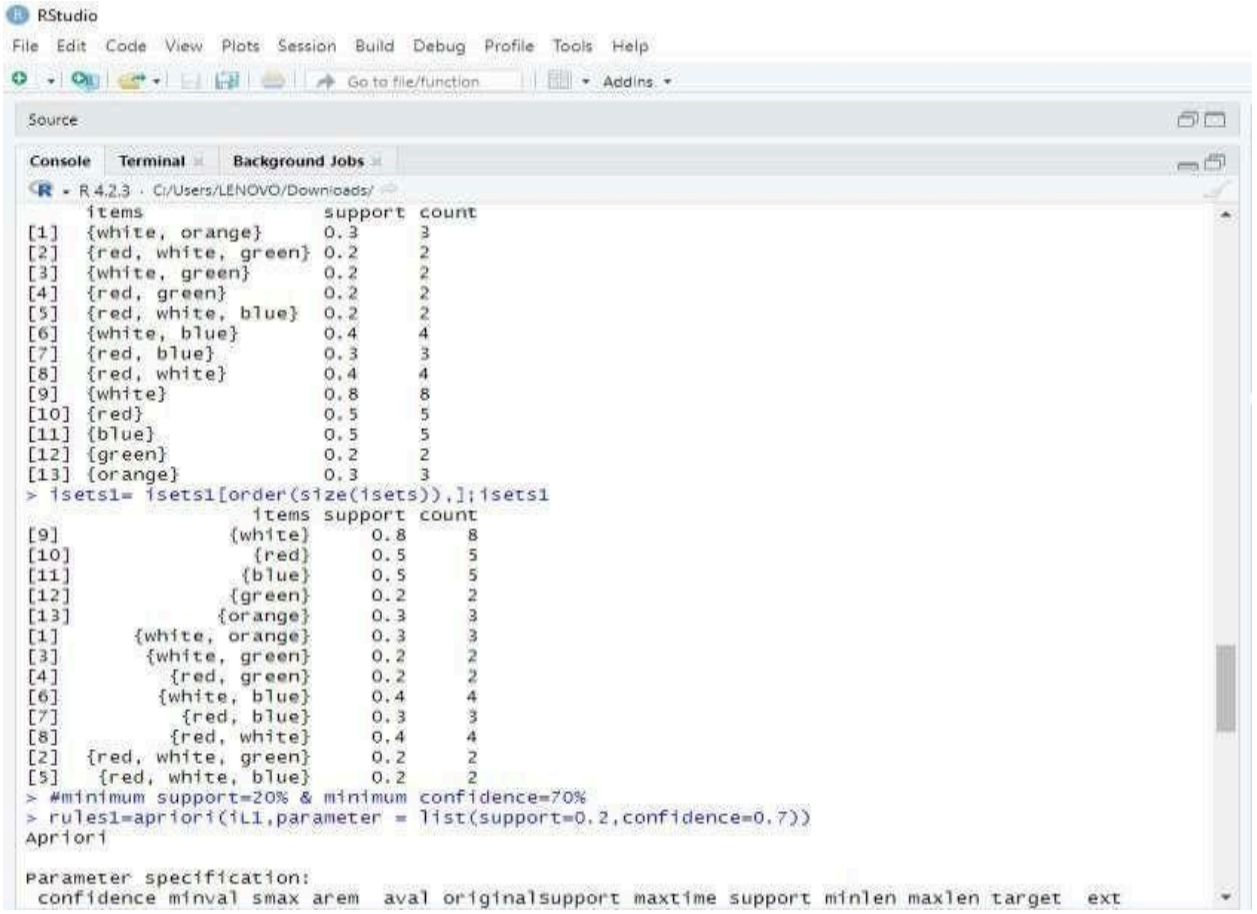

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
> #Association
> #Example: purchase of Mobile Phone Covers
> #1. item list format
> CoverColPC=list(c("red","white","green"),
+ c("white","orange"),
+ c("white","blue"),
+ c("red","white","orange"),
+ c("red","blue"),
+ c("white","blue"),
+ c("white","orange"),
+ c("red","white","blue","green"),
+ c("red","white","blue"),
+ c("yellow"))
> install.packages("arules")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and
install the appropriate version of Rtools before proceeding:
https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/LENOVO/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)
There is a binary version available but the source version is later:
binary source needs_compilation
arules 1.7-7 1.7-8 TRUE
Binaries will be installed
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/arules_1.7-7.zip'
Content type 'application/zip' length 2582217 bytes (2.5 MB)
downloaded 2.5 MB
package 'arules' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\LENOVO\AppData\Local\Temp\Rtmpi87Mpi\downloaded_packages
> library(arules)

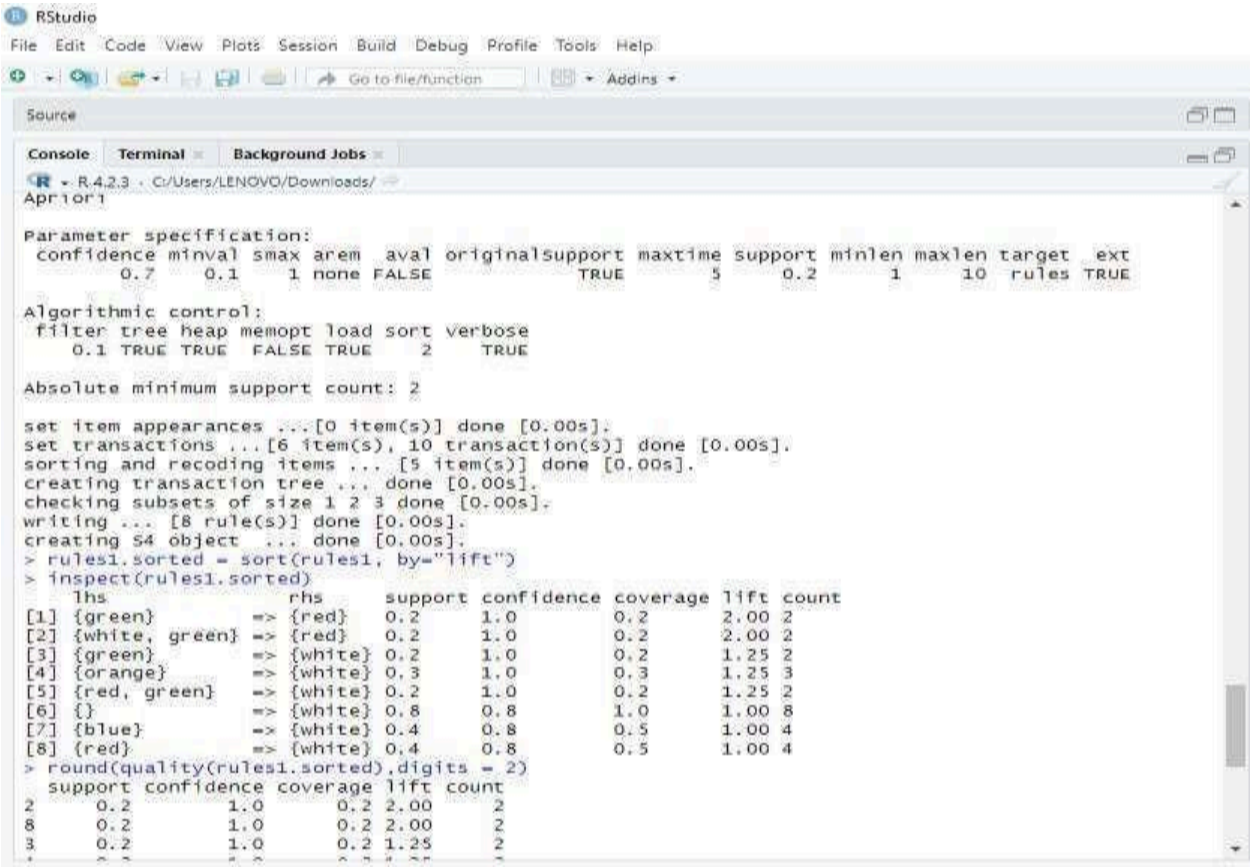

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
The downloaded binary packages are in
C:\Users\LENOVO\AppData\Local\Temp\Rtmpi87Mpi\downloaded_packages
> library(arules)
Loading required package: Matrix
Attaching package: 'arules'
The following objects are masked from 'package:base':
abbreviate, write
> #Class item Matrix is used to store a binary incidence matrix,
> #items labels and optionally transaction IDs and User IDs
> iL=as(CoverColPC,"transactions")
> iL
transactions in sparse format with
10 transactions (rows) and
6 items (columns)
> inspect(iL)
items
[1] {green, red, white}
[2] {orange, white}
[3] {blue, white}
[4] {orange, red, white}
[5] {blue, red}
[6] {blue, white}
[7] {orange, white}
[8] {blue, green, red, white}
[9] {blue, red, white}
[10] {yellow}
> image(iL)
> summary(iL)
transactions as itemMatrix in sparse format with
10 rows (elements/itemsets/transactions) and

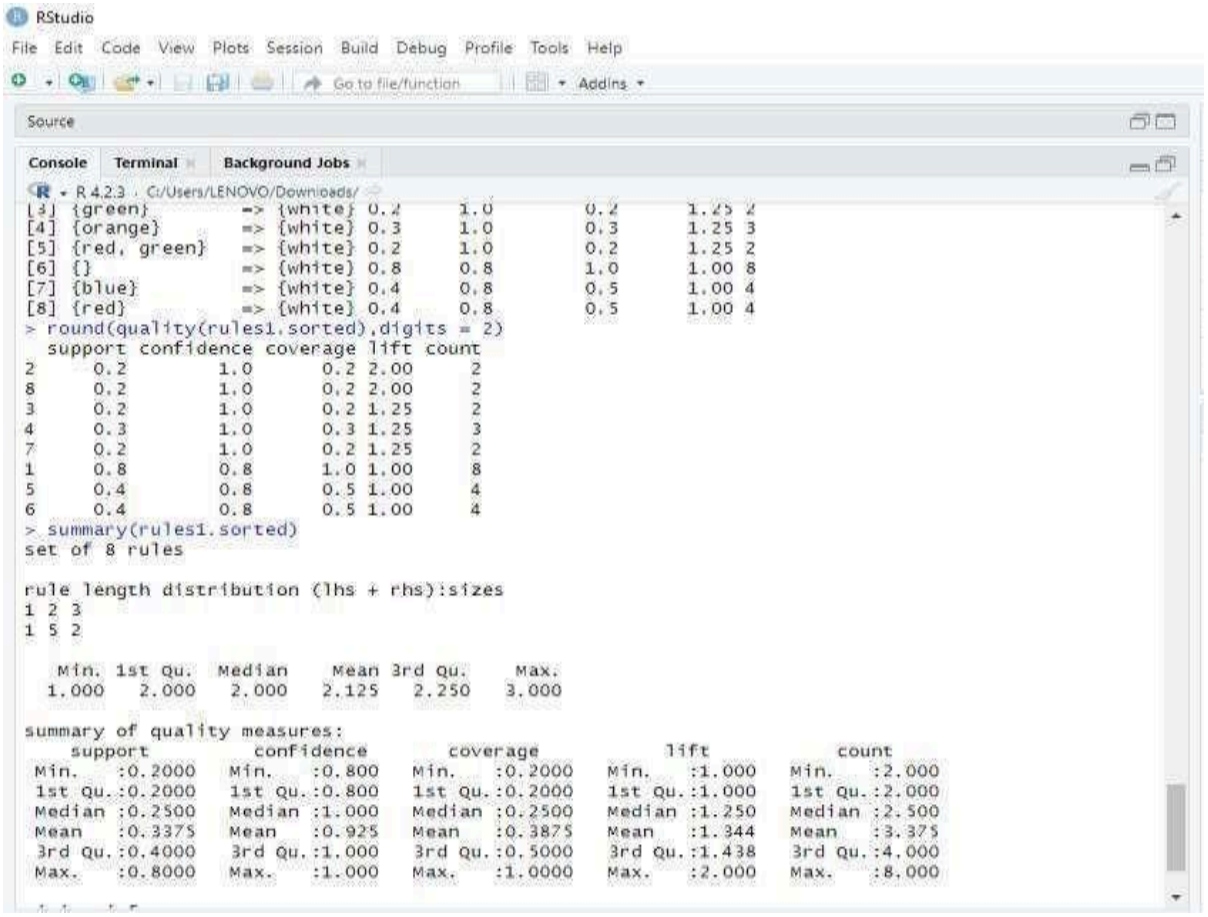

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Terminal Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
> image(iL)
> summary(iL)
transactions as itemMatrix in sparse format with
10 rows (elements/itemsets/transactions) and
6 columns (items) and a density of 0.4
most frequent items:
white blue red orange green (Other)
8 5 5 3 2 1
element (itemset/transaction) length distribution:
sizes
1 2 3 4
1 5 3 1
Min. 1st Qu. Median Mean 3rd Qu. Max.
1.0 2.0 2.0 2.4 3.0 4.0
includes extended item information - examples:
labels
1 blue
2 green
3 orange
> iM=as(iL, "matrix");iM
blue green orange red white yellow
[1,] FALSE TRUE FALSE TRUE TRUE FALSE
[2,] FALSE FALSE TRUE FALSE TRUE FALSE
[3,] TRUE FALSE FALSE FALSE TRUE FALSE
[4,] FALSE FALSE TRUE TRUE TRUE FALSE
[5,] TRUE FALSE FALSE TRUE FALSE FALSE
[6,] TRUE FALSE FALSE FALSE TRUE FALSE
[7,] FALSE FALSE TRUE FALSE TRUE FALSE
[8,] TRUE TRUE FALSE TRUE TRUE FALSE
[9,] TRUE FALSE FALSE TRUE TRUE FALSE
[10,] FALSE FALSE FALSE FALSE FALSE TRUE

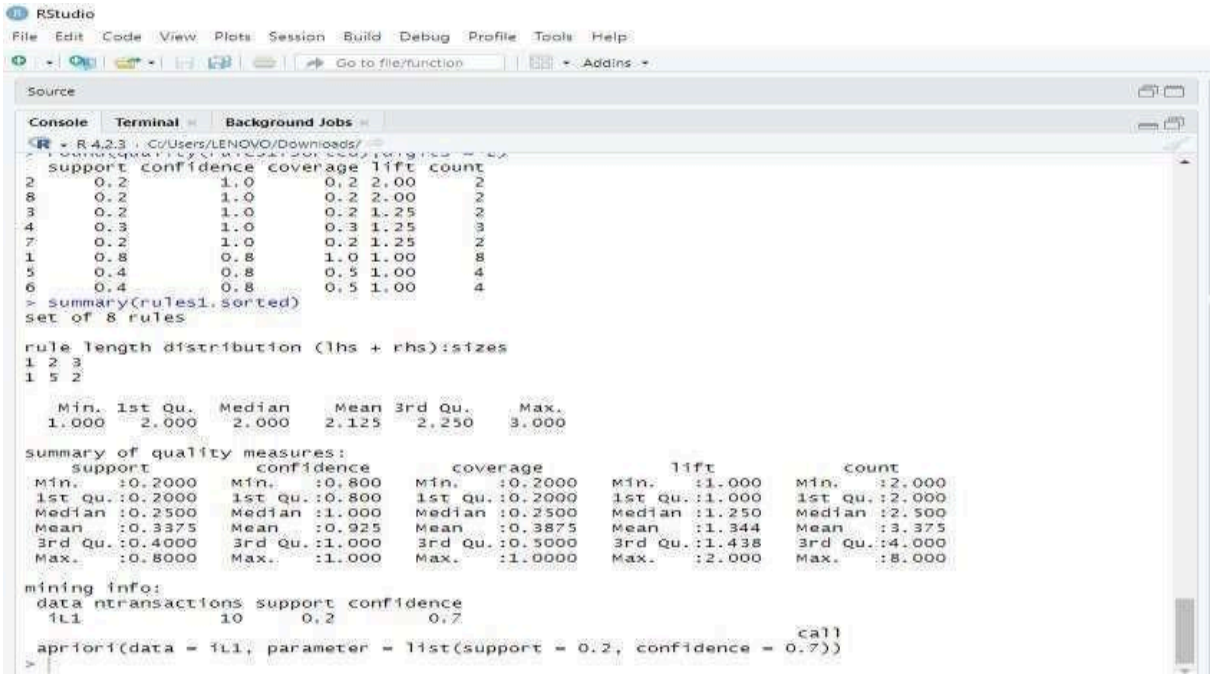

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Terminal Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
[8,] TRUE TRUE FALSE TRUE TRUE FALSE
[9,] TRUE FALSE FALSE TRUE TRUE FALSE
[10,] FALSE FALSE FALSE FALSE FALSE TRUE
> iLabels=c("red", "white","blue","orange","green","yellow")
> iL1=encode(covercolPC,iLabels)
> inspect(iL1)
items
[1] {red, white, green}
[2] {white, orange}
[3] {white, blue}
[4] {red, white, orange}
[5] {red, blue}
[6] {white, blue}
[7] {white, orange}
[8] {red, white, blue, green}
[9] {red, white, blue}
[10] {yellow}
> image(iL1)
> iM1=as(iL1,"matrix");iM1
red white blue orange green yellow
[1,] TRUE TRUE FALSE FALSE TRUE FALSE
[2,] FALSE TRUE FALSE TRUE FALSE FALSE
[3,] FALSE TRUE TRUE FALSE FALSE FALSE
[4,] TRUE TRUE FALSE TRUE FALSE FALSE
[5,] TRUE FALSE TRUE FALSE FALSE FALSE
[6,] FALSE TRUE TRUE FALSE FALSE FALSE
[7,] FALSE TRUE FALSE TRUE FALSE FALSE
[8,] TRUE TRUE TRUE FALSE TRUE FALSE
[9,] TRUE TRUE TRUE FALSE FALSE FALSE
[10,] FALSE FALSE FALSE FALSE FALSE TRUE
> #Convert transactions to Transaction ID Lists
> inspect(as(iL1,"tidLists"))
items transactionIDs
1 red {1,4,5,8,9}
2 white {1,2,3,4,6,7,8,9}

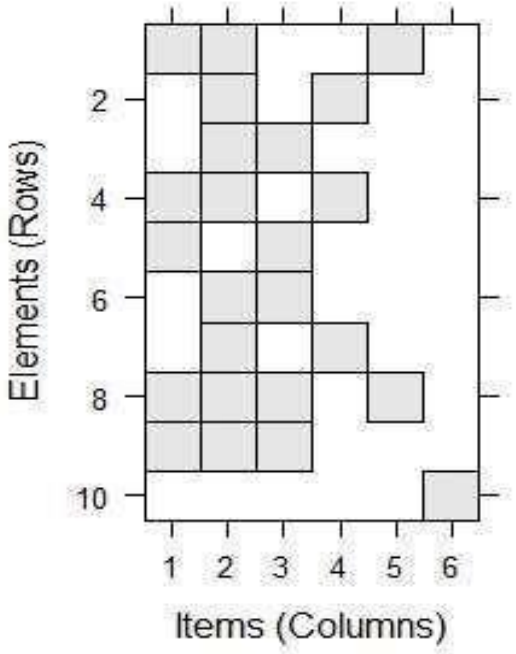

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Terminal Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
> inspect(as(iL1,"tidLists"))
items transactionIDs
1 red {1,4,5,8,9}
2 white {1,2,3,4,6,7,8,9}
3 blue {3,5,6,8,9}
4 orange {2,4,7}
5 green {1,8}
6 yellow {10}
> #2 Binary Matrix format
> iMn = apply(iM1,c(1,2),as.numeric)
> dimnames(iMn)=list(c(1:nrow(iM1)),itemLabels(iL1)); iMn
red white blue orange green yellow
1 1 1 0 0 1 0
2 0 1 0 1 0 0
3 0 1 1 0 0 0
4 1 1 0 1 0 0
5 1 0 1 0 0 0
6 0 1 1 0 0 0
7 0 1 0 1 0 0
8 1 1 1 0 1 0
9 1 1 1 0 0 0
10 0 0 0 0 0 1
> #Itemset with support (count)>=2
> isets= eclat(iL1, parameter = list(support=0.2),control = list(verbose=F))
> isets1=inspect(isets)
items support count
[1] {white, orange} 0.3 3
[2] {red, white, green} 0.2 2
[3] {white, green} 0.2 2
[4] {red, green} 0.2 2
[5] {red, white, blue} 0.2 2
[6] {white, blue} 0.4 4
[7] {red, blue} 0.3 3
[8] {red, white} 0.4 4
[9] {white} 0.8 8


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
items support count
[1] {white, orange} 0.3 3
[2] {red, white, green} 0.2 2
[3] {white, green} 0.2 2
[4] {red, green} 0.2 2
[5] {red, white, blue} 0.2 2
[6] {white, blue} 0.4 4
[7] {red, blue} 0.3 3
[8] {red, white} 0.4 4
[9] {white} 0.8 8
[10] {red} 0.5 5
[11] {blue} 0.5 5
[12] {green} 0.2 2
[13] {orange} 0.3 3
> isets1= isets1[order(size(isets)),];isets1
items support count
[9] {white} 0.8 8
[10] {red} 0.5 5
[11] {blue} 0.5 5
[12] {green} 0.2 2
[13] {orange} 0.3 3
[1] {white, orange} 0.3 3
[3] {white, green} 0.2 2
[4] {red, green} 0.2 2
[6] {white, blue} 0.4 4
[7] {red, blue} 0.3 3
[8] {red, white} 0.4 4
[2] {red, white, green} 0.2 2
[5] {red, white, blue} 0.2 2
> #minimum support=20% & minimum confidence=70%
> rules1=apriori(iL1,parameter = list(support=0.2,confidence=0.7))
Apriori
Parameter specification:
confidence minval smax arem aval originalsupport maxtime support minlen maxlen target ext


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Terminal Background Jobs
R 4.4.2.3 · C:/Users/LENOVO/Downloads/
Apriori
Parameter specification:
confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext
0.7 0.1 1 none FALSE TRUE 5 0.2 1 10 rules TRUE
Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE FALSE TRUE 2 TRUE
Absolute minimum support count: 2
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[6 item(s), 10 transaction(s)] done [0.00s].
sorting and recoding items ... [5 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [8 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
> rules1.sorted = sort(rules1, by="lift")
> inspect(rules1.sorted)
lhs rhs support confidence coverage lift count
[1] {green} => {red} 0.2 1.0 0.2 2.00 2
[2] {white, green} => {red} 0.2 1.0 0.2 2.00 2
[3] {green} => {white} 0.2 1.0 0.2 1.25 2
[4] {orange} => {white} 0.3 1.0 0.3 1.25 3
[5] {red, green} => {white} 0.2 1.0 0.2 1.25 2
[6] {} => {white} 0.8 0.8 1.0 1.00 8
[7] {blue} => {white} 0.4 0.8 0.5 1.00 4
[8] {red} => {white} 0.4 0.8 0.5 1.00 4
> round(quality(rules1.sorted),digits = 2)
support confidence coverage lift count
2 0.2 1.0 0.2 2.00 2
8 0.2 1.0 0.2 2.00 2
3 0.2 1.0 0.2 1.25 2


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
[3] {green} => {white} 0.2 1.0 0.2 1.25 2
[4] {orange} => {white} 0.3 1.0 0.3 1.25 3
[5] {red, green} => {white} 0.2 1.0 0.2 1.25 2
[6] {} => {white} 0.8 0.8 1.0 1.00 8
[7] {blue} => {white} 0.4 0.8 0.5 1.00 4
[8] {red} => {white} 0.4 0.8 0.5 1.00 4
> round(quality(rules1.sorted),digits = 2)
support confidence coverage lift count
2 0.2 1.0 0.2 2.00 2
8 0.2 1.0 0.2 2.00 2
3 0.2 1.0 0.2 1.25 2
4 0.3 1.0 0.3 1.25 3
7 0.2 1.0 0.2 1.25 2
1 0.8 0.8 1.0 1.00 8
5 0.4 0.8 0.5 1.00 4
6 0.4 0.8 0.5 1.00 4
> summary(rules1.sorted)
set of 8 rules
rule length distribution (lhs + rhs):sizes
1 2 3
1 5 2
Min. 1st Qu. Median Mean 3rd Qu. Max.
1.000 2.000 2.000 2.125 2.250 3.000
summary of quality measures:
support confidence coverage lift count
Min. :0.2000 Min. :0.800 Min. :0.2000 Min. :1.000 Min. :2.000
1st Qu.:0.2000 1st Qu.:0.800 1st Qu.:0.2000 1st Qu.:1.000 1st Qu.:2.000
Median :0.2500 Median :1.000 Median :0.2500 Median :1.250 Median :2.500
Mean :0.3375 Mean :0.925 Mean :0.3875 Mean :1.344 Mean :3.375
3rd Qu.:0.4000 3rd Qu.:1.000 3rd Qu.:0.5000 3rd Qu.:1.438 3rd Qu.:4.000
Max. :0.8000 Max. :1.000 Max. :1.0000 Max. :2.000 Max. :8.000


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.2.3 · C:/Users/LENOVO/Downloads/
support confidence coverage lift count
2 0.2 1.0 0.2 2.00 2
8 0.2 1.0 0.2 2.00 2
3 0.2 1.0 0.2 1.25 2
4 0.3 1.0 0.3 1.25 3
7 0.2 1.0 0.2 1.25 2
1 0.8 0.8 1.0 1.00 8
5 0.4 0.8 0.5 1.00 4
6 0.4 0.8 0.5 1.00 4
> summary(rules1.sorted)
set of 8 rules
rule length distribution (lhs + rhs):sizes
1 2 3
1 5 2
Min. 1st Qu. Median Mean 3rd Qu. Max.
1.000 2.000 2.000 2.125 2.250 3.000
summary of quality measures:
support confidence coverage lift count
Min. :0.2000 Min. :0.800 Min. :0.2000 Min. :1.000 Min. :2.000
1st Qu.:0.2000 1st Qu.:0.800 1st Qu.:0.2000 1st Qu.:1.000 1st Qu.:2.000
Median :0.2500 Median :1.000 Median :0.2500 Median :1.250 Median :2.500
Mean :0.3375 Mean :0.925 Mean :0.3875 Mean :1.344 Mean :3.375
3rd Qu.:0.4000 3rd Qu.:1.000 3rd Qu.:0.5000 3rd Qu.:1.438 3rd Qu.:4.000
Max. :0.8000 Max. :1.000 Max. :1.0000 Max. :2.000 Max. :8.000
mining info:
data ntransactions support confidence
iL1 10 0.2 0.7
call
apriori(data = iL1, parameter = list(support = 0.2, confidence = 0.7))

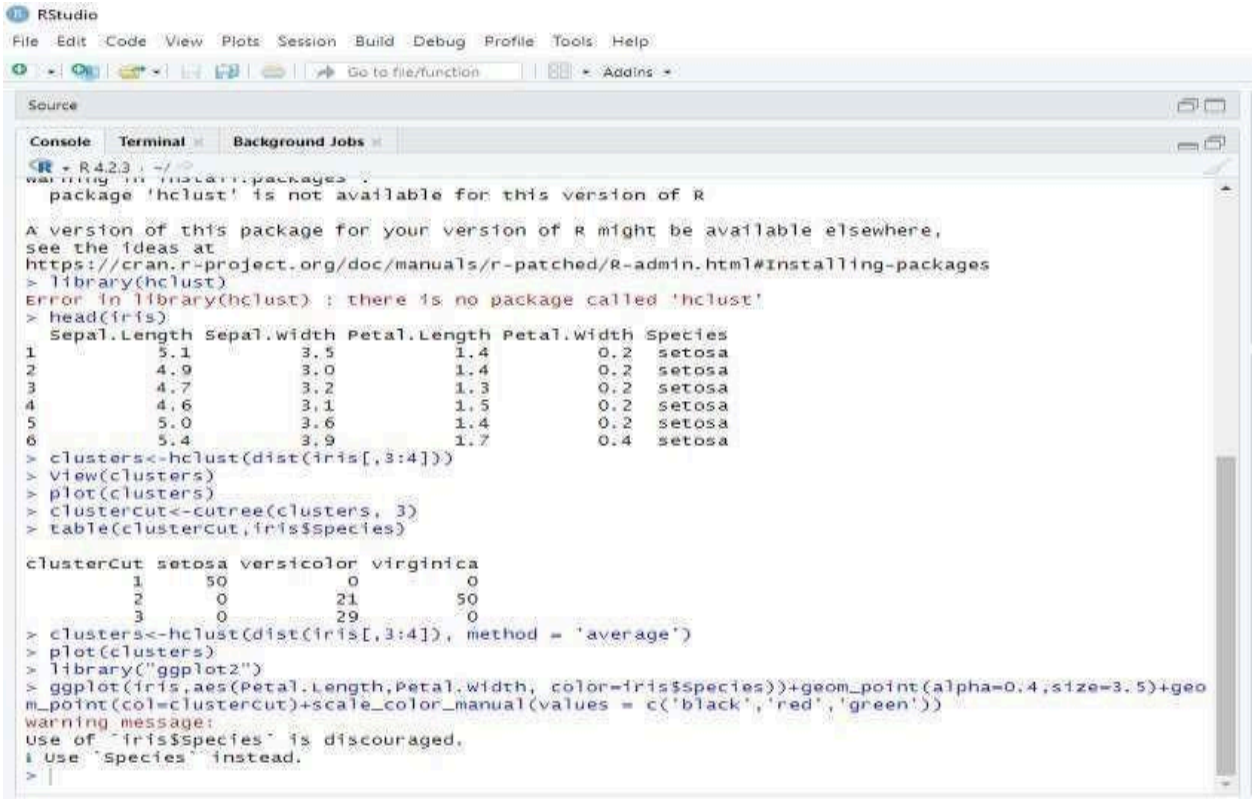


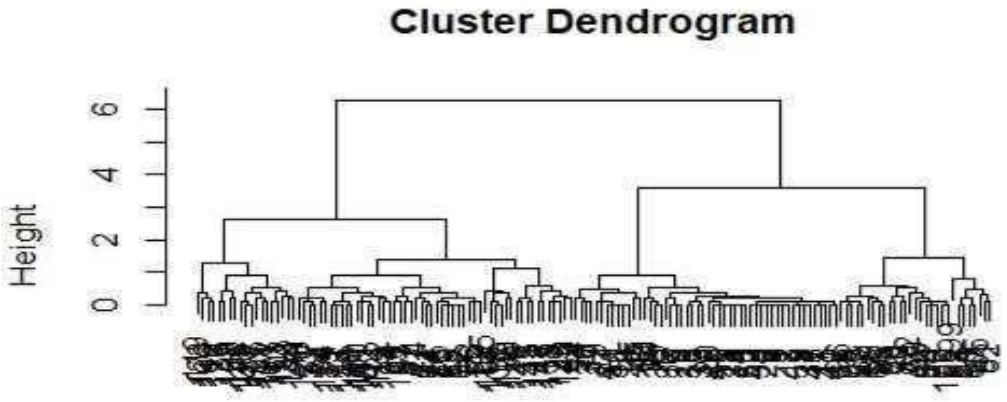
11 Implementation and analysis of clustering algorithms - Agglomerative

Agglomerative refers to a **bottom-up** approach used in certain types of algorithms, particularly in **hierarchical clustering**. In this context, agglomerative methods are used to build a hierarchy of clusters by iteratively merging smaller clusters into larger ones. The goal of agglomerative clustering is to create a tree-like structure called a **dendrogram**, which shows how clusters are merged at different levels of similarity or distance. **Code:-** install.packages("hclust") library(hclust) head(iris) clusters<-hclust(dist(iris[,3:4])) View(clusters) plot(clusters) clusterCut<-cutree(clusters, 3) table(clusterCut,iris$Species) clusters<-hclust(dist(iris[,3:4]), method = 'average') plot(clusters) library("ggplot2") ggplot(iris,aes(Petal.Length,Petal.Width,

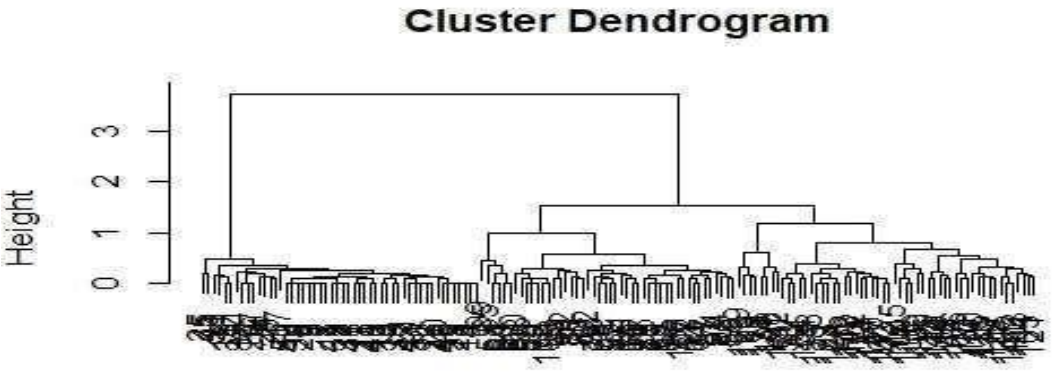
color=iris$Species))+geom_point(alpha=0.4,size=3.5)+geom_point(col=clusterCut)+scale_color_manual(values = c('black','red','green'))

Output:-

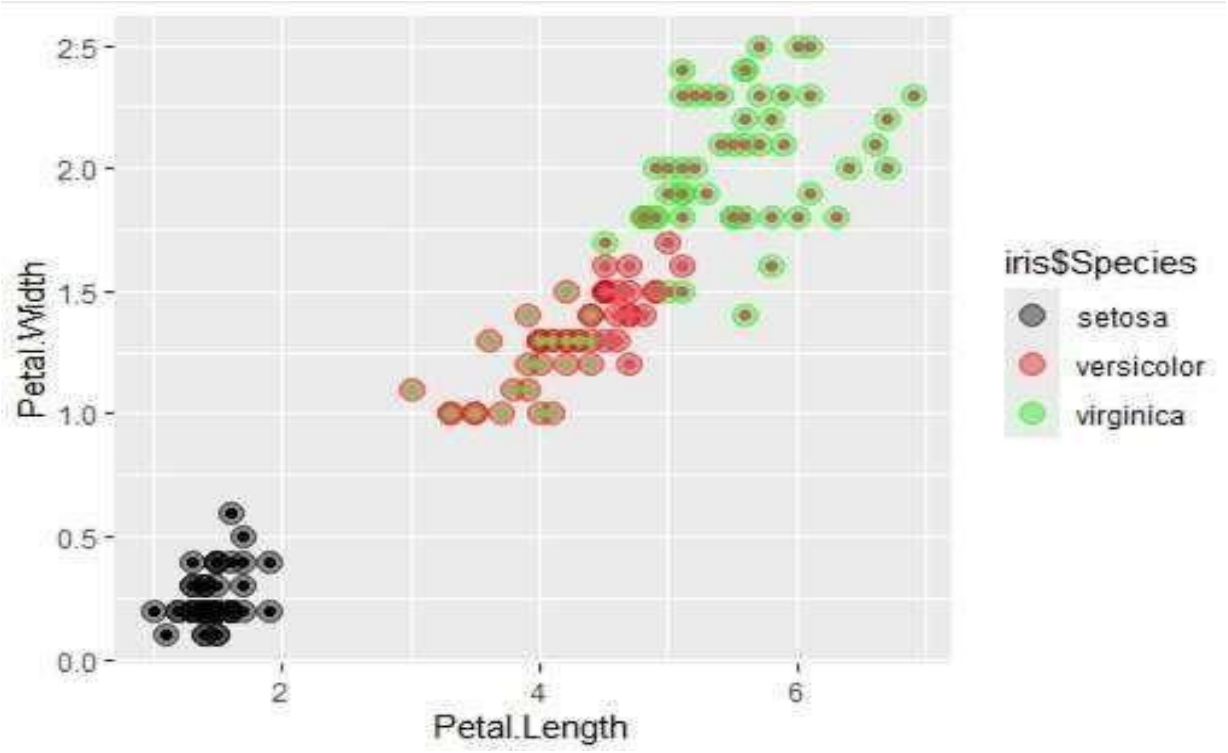




dist(iris[, 3:4])
hclust (*, "complete")



dist(iris[, 3:4])
hclust (*, "average")



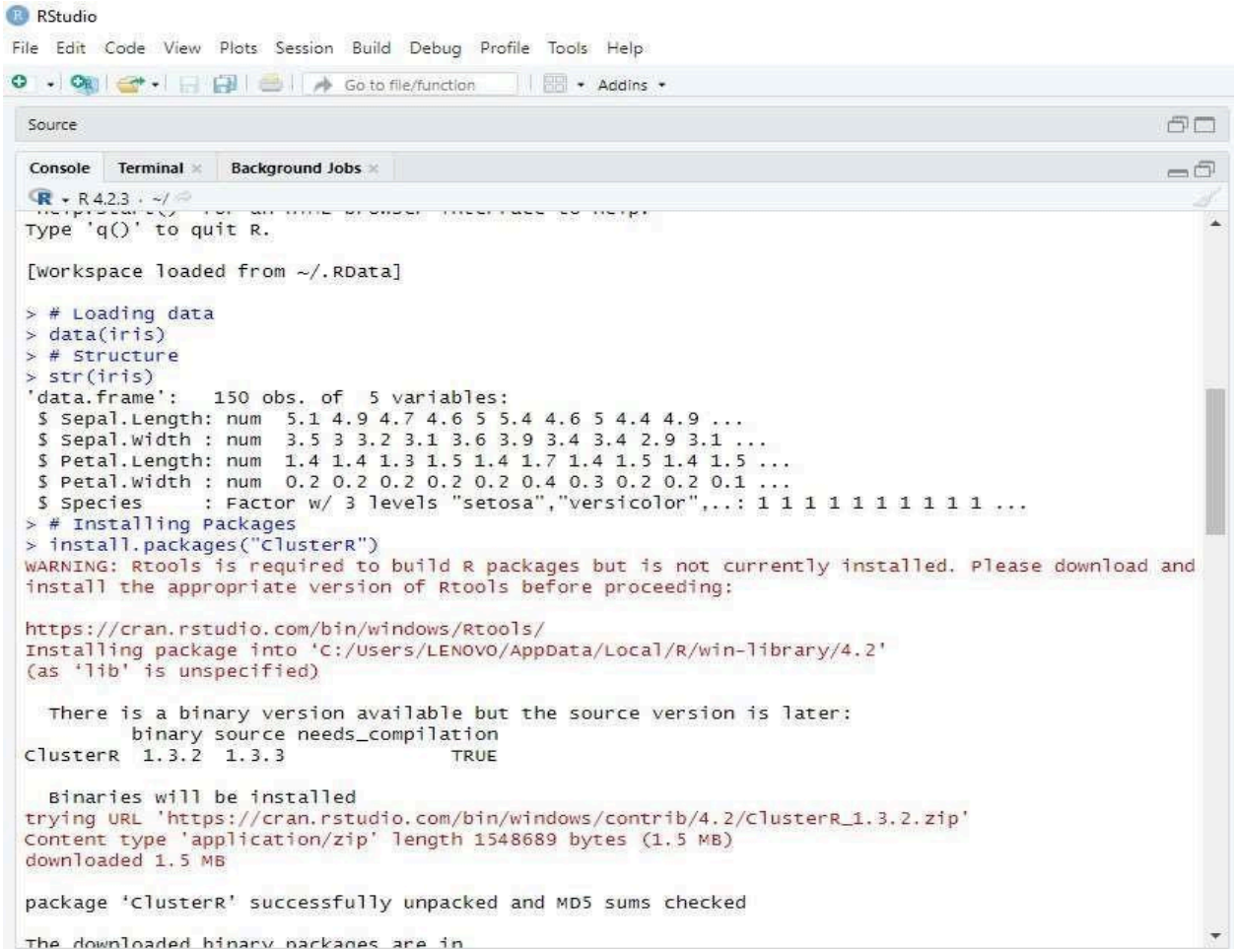
12 Implementation and analysis of clustering algorithms - K-Means

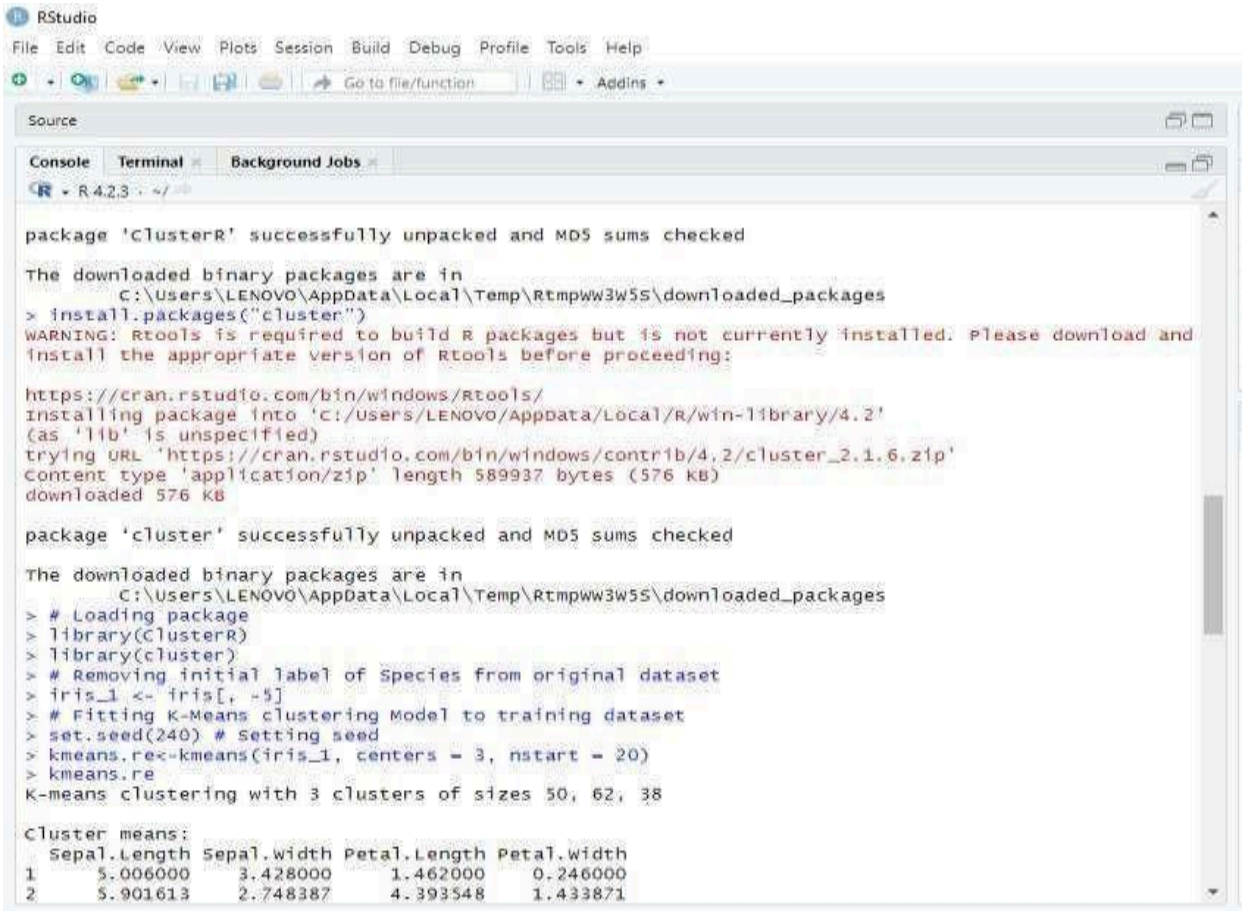
K-Means is a **popular unsupervised machine learning algorithm** used for **clustering** tasks, where the goal is to partition a set of data points into **K distinct, non-overlapping clusters**. Each cluster contains data points that are more similar to each other than to those in other clusters. K-Means is widely used in applications such as customer segmentation, image compression, anomaly detection, and pattern recognition.

Code:-

```
# Loading data data(iris) # Structure  str(iris)

# Installing Packages install.packages("ClusterR") install.packages("cluster") # Loading package
library(ClusterR) library(cluster)

# Removing initial label of Species from original dataset iris_1 <- iris[, -5]

# Fitting K-Means clustering Model to training dataset set.seed(240) # Setting seed kmeans.re<-kmeans(iris_1,
centers = 3, nstart = 20) kmeans.re

# Cluster identification for each observation kmeans.re$cluster # Confusion Matrix cm<-table(iris$Species,
kmeans.re$cluster) cm

# Model Evaluation and visualization plot(iris_1[c("Sepal.Length","Sepal.Width")],       col = kmeans.re$cluster,
main = "K-means with 3 clusters") ## Plotiing cluster centers kmeans.re$centers kmeans.re$centers[,
c("Sepal.Length","Sepal.Width")]



# cex is font size, pch is symbol points(kmeans.re$centers[, c("Sepal.Length","Sepal.Width")],         col = 1:3,
pch = 8, cex = 3)  ## Visualizing clusters y_kmeans <- kmeans.re$cluster clusplot(iris_1[,
c("Sepal.Length","Sepal.Width")],         y_kmeans,         lines = 0,         shade = TRUE,         color = TRUE,
labels = 2,         plotchar = FALSE,         span = TRUE,         main = paste("Cluster iris"),         xlab =
'Sepal.Length',         ylab = 'Sepal.Width')
```

Output:-


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Background Jobs
R 4.2.3 · ~/
Type 'q()' to quit R.
[workspace loaded from ~/.RData]
> # Loading data
> data(iris)
> # Structure
> str(iris)
'data.frame': 150 obs. of 5 variables:
$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
$ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
$ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
$ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> # Installing Packages
> install.packages("ClusterR")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and
install the appropriate version of Rtools before proceeding:
https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/LENOVO/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)
There is a binary version available but the source version is later:
binary source needs_compilation
ClusterR 1.3.2 1.3.3 TRUE
Binaries will be installed
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/ClusterR_1.3.2.zip'
Content type 'application/zip' length 1548689 bytes (1.5 MB)
downloaded 1.5 MB
package 'ClusterR' successfully unpacked and MD5 sums checked
The downloaded binary packages are in


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Background Jobs
R 4.2.3 · ~/
package 'ClusterR' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\LENOVO\AppData\Local\Temp\RtmpWW3W5S\downloaded_packages
> install.packages("cluster")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:
https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/LENOVO/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/cluster_2.1.6.zip'
Content type 'application/zip' length 589937 bytes (576 KB)
downloaded 576 KB
package 'cluster' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\LENOVO\AppData\Local\Temp\RtmpWW3W5S\downloaded_packages
> # Loading package
> library(ClusterR)
> library(cluster)
> # Removing initial label of Species from original dataset
> iris_1 <- iris[, -5]
> # Fitting K-Means clustering Model to training dataset
> set.seed(240) # Setting seed
> kmeans.re<-kmeans(iris_1, centers = 3, nstart = 20)
> kmeans.re
K-means clustering with 3 clusters of sizes 50, 62, 38
Cluster means:
Sepal.Length Sepal.Width Petal.Length Petal.Width
1 5.006000 3.428000 1.462000 0.246000
2 5.901613 2.748387 4.393548 1.433871

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

Source

Console Terminal Background Jobs

R 4.2.3 · ~/

```
Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [46] 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [91] 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3 2 2 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 2 3
[136] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 15.15100 39.82097 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> # Cluster identification for each observation
> kmeans.re$cluster
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [46] 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [91] 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3 2 2 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 2 3
[136] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 2
> # Confusion Matrix
> cm<-table(iris$Species, kmeans.re$cluster)
> cm

            1  2  3
  setosa     50  0  0
  versicolor  0 48  2
  virginica   0 14 36
> # Model Evaluation and visualization
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

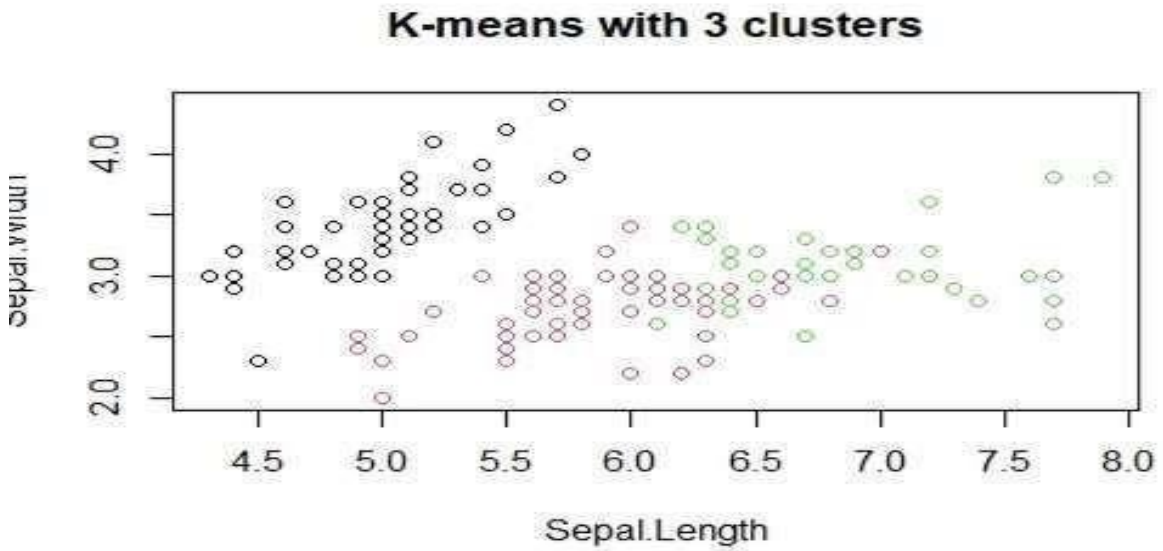
Go to file/function | Addins

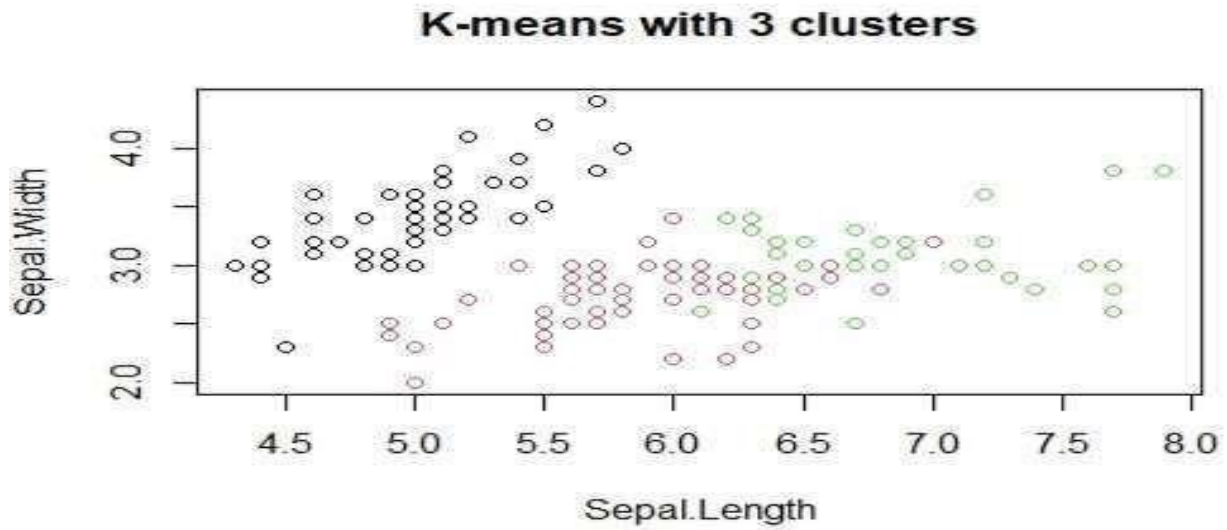
Source

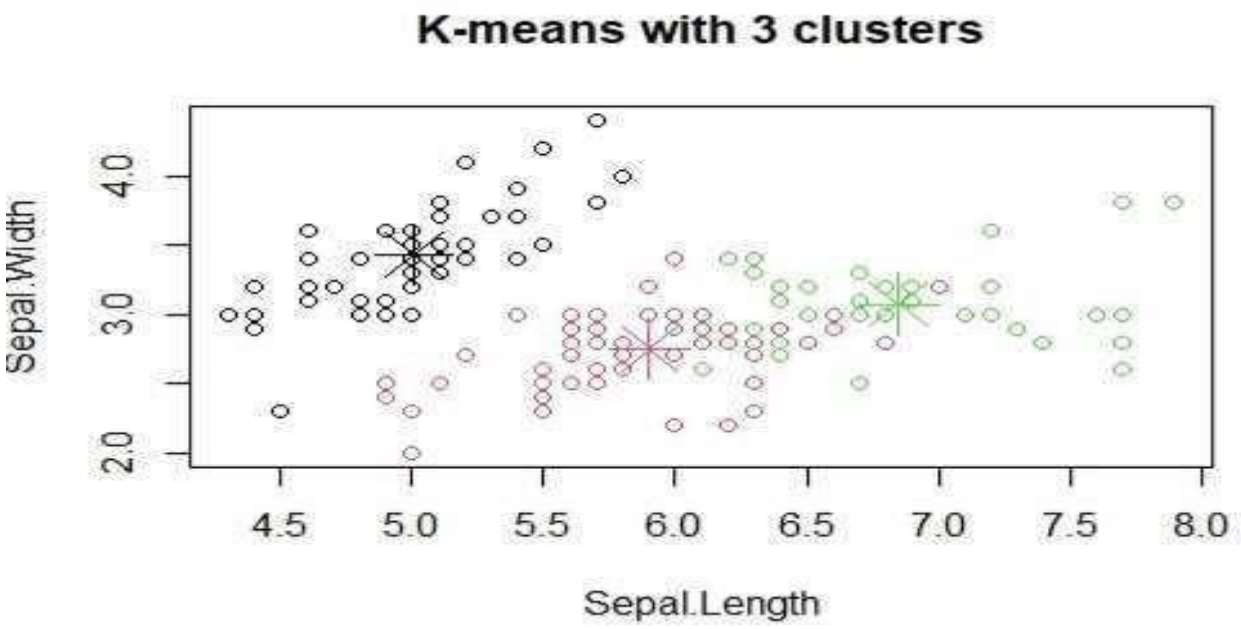
Console Terminal Background Jobs

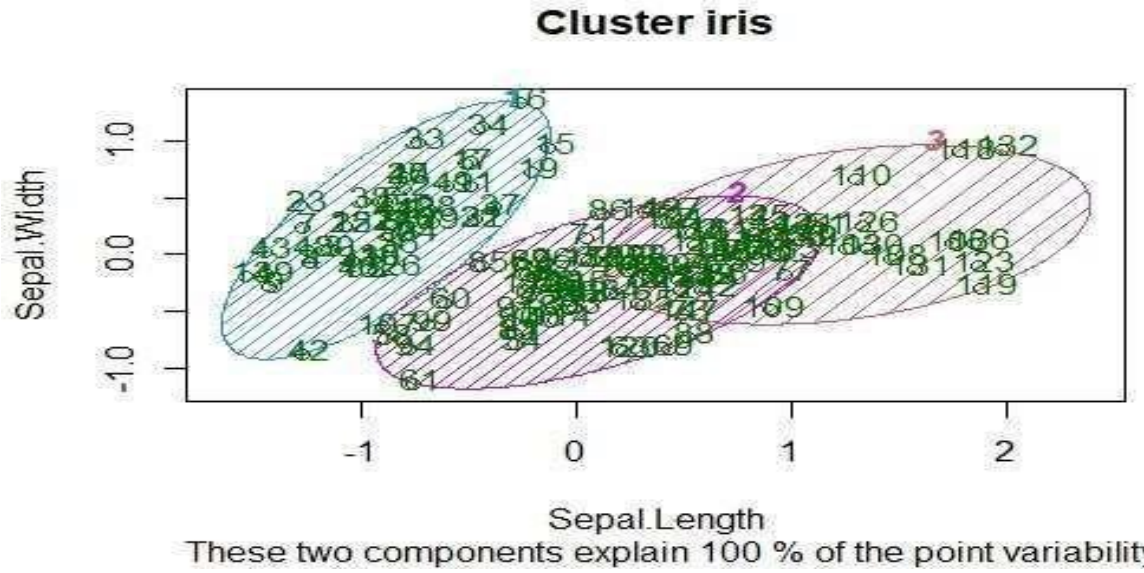
R 4.2.3 · ~/

```
  versicolor  0 48  2
  virginica   0 14 36
> # Model Evaluation and visualization
> plot(iris_1[c("Sepal.Length","Sepal.Width")],
+      col = kmeans.re$cluster,
+      main = "K-means with 3 clusters")
> ## Plotting cluster centers
> kmeans.re$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053
> kmeans.re$centers[, c("Sepal.Length","Sepal.Width")]
  Sepal.Length Sepal.Width
1     5.006000    3.428000
2     5.901613    2.748387
3     6.850000    3.073684
> # cex is font size, pch is symbol
> points(kmeans.re$centers[, c("Sepal.Length","Sepal.Width")],
+        col = 1:3, pch = 8, cex = 3)
> ## visualizing clusters
> y_kmeans <- kmeans.re$cluster
> clusplot(iris_1[, c("Sepal.Length","Sepal.Width")],
+          y_kmeans,
+          lines = 0,
+          shade = TRUE,
+          color = TRUE,
+          labels = 2,
+          plotchar = FALSE,
+          span = TRUE,
+          main = paste("Cluster iris"),
+          xlab = 'Sepal.Length',
+          ylab = 'Sepal.Width')
>
```


K-means with 3 clusters
Sepal.Width
2.0
3.0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal.Length


K-means with 3 clusters
Sepal.Width
2.0
3.0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal.Length





These two components explain 100 % of the point variability.