

**Q1. Create a file “people.txt” with the following data:**

Age	Age Group	Height	Status	Years Married
21	adult	6.0	single	-1
2	child	3	married	0
18	adult	5.7	married	20
221	elderly	5	widowed	2
34	child	-7	married	3

i) Read the data from the file “people.txt”.

ii) Create a rule set E that contain rules to check for the following conditions :

- The age should be in the range 0-150.
- The age should be greater than years married.
- The status should be married or single or widowed.
- If age is less than 18 the age group should be child, if age is between 18 and 65 the age group should be adult, if age is more than 65 the age group should be elderly.

iii) Check whether rule set E is violated by the data in the file people.txt.

iv) Summarize the results obtained in part (iii).

v) Visualize the results obtained in part (iii).

people.text file:

people - Notepad

```
File Edit Format View Help
age      agegroup    height    status      yearsmarried
21       adult        6.0       single      -1
2        child        3         married     0
18       adult        5.7       married     20
221      elderly      5         widowed    2
34       child        -7        married    3
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Q1.r

```
1 install.packages("editrules")
2 library("editrules")
3
4 setwd("C:\\Users\\Lenovo\\Desktop\\DM_files")
5 getwd()
6
7 d <- read.table("people.txt", header = TRUE)
8 attach(d)
9
10 E <- editset(expression(
11   age >= 0,
12   age <= 150,
13   age > yearsmarried,
14   status %in% c('single', 'married', 'widowed'),
15   if (age <= 18) agegroup %in% c('child'),
16   if (age > 18 & age < 65) agegroup %in% c('adult'),
17   if (age >= 65) agegroup %in% c('elderly')
18 ))
19
20 sm <- violatedEdits(E, d)
21 summary(sm)
22 plot(sm)
```

Environment History Connections

Global Environment

height_norm	num [1:200]	160 156 166 162 163 ...
i	10L	
l	num [1:100]	95.5 95.5 95.5 95.5 95.5 ...
m	2L	
Marks	num [1:3]	10 20 30
modeAge	"20"	
n	2L	
Name	chr [1:3]	"ABC" "PQR" "XYZ"
num	int [1:50]	1 2 3 4 5 6 7 8 9 10 ...
p	2L	
q	2L	
re1	Named num [1:6]	0.333 0.833 1 0.333 0.667 ...
Roll	num [1:3]	1 2 3
s1	int [1:40]	6 4 5 4 3 6 6 4 6 6 ...
s2	int [1:70]	2 6 1 1 6 5 2 6 1 1 ...
s3	int [1:100]	5 5 2 2 3 3 2 3 6 2 ...
sm	'violatedEdits'	logi [1:5, 1:8] FALSE FALSE FALSE FALSE FALSE ...

Files Plots Packages Help Viewer

C:/Users/Lenovo/Desktop/DM\_files/

## Output:

The screenshot shows an RStudio interface with the following components:

- Console:** Displays R code and its output. The code reads a dataset from "people.txt", attaches it, and performs various edits on the "age" column. It then calculates edit violations and plots them.
- Environment:** Shows the global environment with objects like height\_norm, i, l, m, Marks, modeAge, n, Name, num, p, q, rel, Roll, s1, s2, s3, and sm.
- Plots:** Two plots are shown:
  - Edit violation frequency of top 8 edits:** A horizontal bar chart showing the frequency of edits. The x-axis ranges from 0.00 to 0.20. The bars represent edits num2, num3, mix4, and mix5, each with a frequency of approximately 0.20.
  - Edit violations per record:** A histogram showing the count of records with different numbers of violations. The x-axis is "Number of violations" (1.0 to 2.0) and the y-axis is "Count" (1.0). A single bar at 1.0 indicates 2 records with no violations.

**Q2. Perform the following preprocessing tasks on the dirty\_iris dataset.**

- Calculate the number and percentage of observations that are complete.
- Replace all the special values in data with NA.

iii) Define these rules in a separate text file and read them.

(Use editfile function in R (package editrules). Use similar function in Python).

Print the resulting constraint object.

- Species should be one of the following values: setosa, versicolor or virginica.
- All measured numerical properties of an iris should be positive.
- The petal length of an iris is atleast 2 times its petal width.
- The sepal length of an iris cannot exceed 30cm.
- The sepals of an iris are longer than its petals.

iv) Determine how often each rule is broken (violatedEdits). Also summarize and plot the result.

v) Find outliers in sepal length using boxplot and boxplot.stats

## **dirty\_iris.csv:**



The screenshot shows a Windows Notepad window titled "dirty\_iris - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main content area displays a CSV file with the following data:

```
"Sepal.Length","Sepal.Width","Petal.Length","Petal.Width","Species"
6.4,3.2,4.5,1.5,"versicolor"
6.3,3.3,6.2,5,"virginica"
6.2,$,5.4,2.3,"virginica"
5,3.4,1.6,0.4,"setosa"
5.7,2.6,3.5,1,"versicolor"
5.3,-1.5,_,0.2,"setosa"
6.4,2.7,5.3,$,"virginica"
5.9,3,5.1,1.8,"virginica"
5.8,2.7,4.1,1,"versicolor"
4.8,3.1,1.6,0.2,"setosa"
5,3.5,1.6,0.6,"setosa"
6,2.7,5.1,1.6,"versicolor"
6,3,4.8,$,"virginica"
6.8,2.8,4.8,1.4,"versicolor"
$,3.9,1.7,0.4,"setosa"
5,-3,3.5,1,"versicolor"
5.5,$,4,1.3,"versicolor"
4.7,3.2,1.3,0.2,"setosa"
$,4,$,0.2,"setosa"
5.6,$,4.2,1.3,"versicolor"
4.9,3.6,$,0.1,"setosa"
5.4,$,4.5,1.5,"versicolor"
6.2,2.8,$,1.8,"virginica"
6.7,3.3,5.7,2.5,"virginica"
$,3,5.9,2.1,"virginica"
4.6,3.2,1.4,0.2,"setosa"
4.9,3.1,1.5,0.1,"setosa"
73,29,63,$,"virginica"
6.5,3.2,5.1,2,"virginica"
< 2 0 0 92 1 2 "versicolor"
```

The status bar at the bottom indicates Ln 1, Col 1, 100%, Windows (CRLF), and UTF-8.

## **myfile.txt:**

myfile - Notepad

```
File Edit Format View Help
Species %in% c("Iris-setosa", "Iris-versicolor", "Iris-virginica")
Sepal.Length > 0
Sepal.Length < 30
Sepal.Width > 0
Petal.Length > 0
Petal.Width > 0
Petal.Length > 2*Petal.Width
Sepal.Length > Petal.Length
```

<

Ln 1, Col 1

100% Windows (CRLF)

UTF-8

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help, and a Project dropdown set to (None). The main area has two tabs: Q1.r (active) and Q2.r. The Q1.r tab contains the following R code:

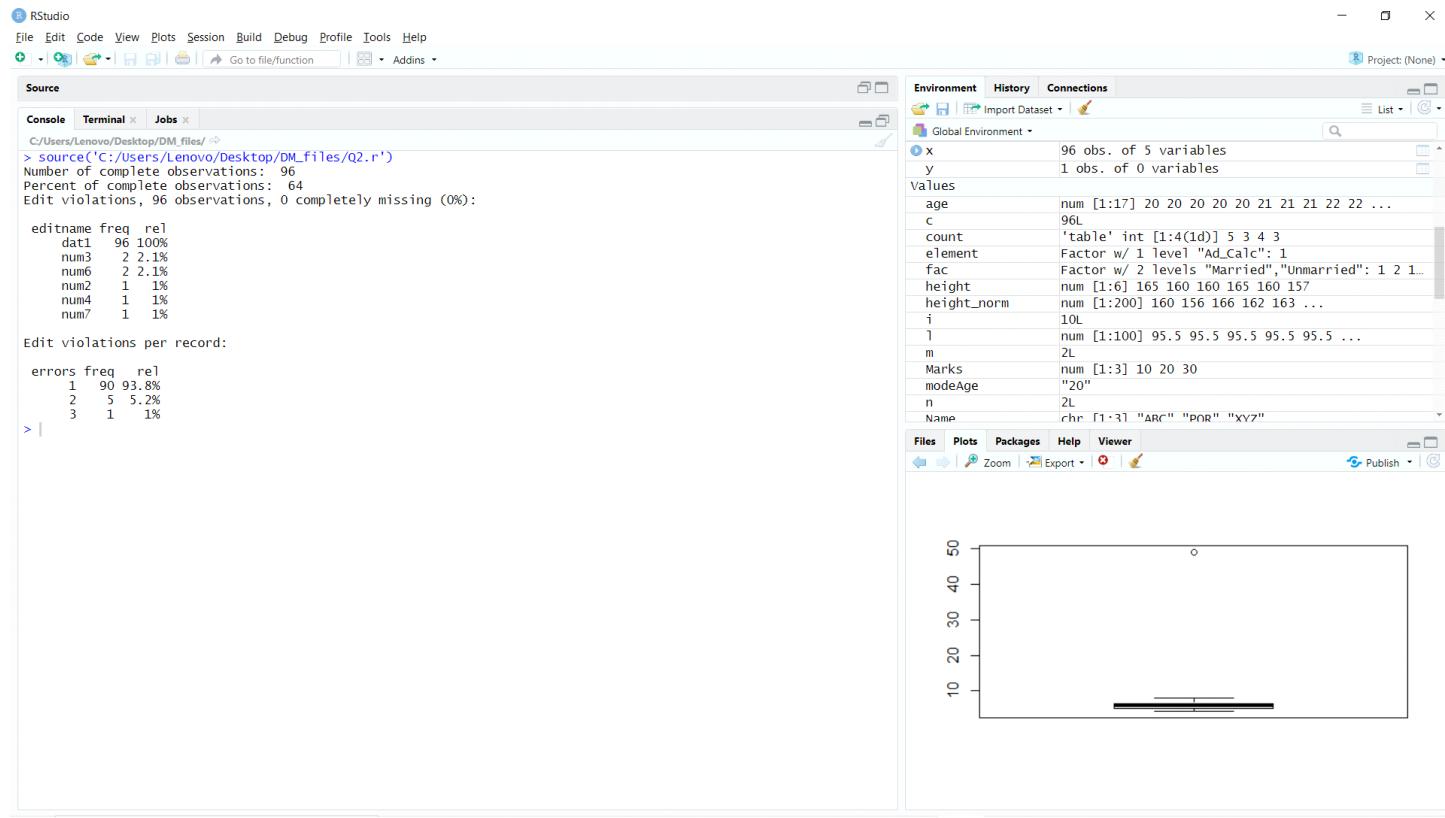
```
1 setwd("C:\\\\Users\\\\Lenovo\\\\Desktop\\\\DM_files")
2 library("editrules")
3 
4 x <- read.csv("dirty_iris.csv")
5 #replace speacial values with NA
6 x[,-5] <- lapply(x[,-5], function(y) as.numeric(as.character(y)))
7 
8 #total number of complete observations
9 c <- sum(complete.cases(x))
10 cat("Number of complete observations: ", c, "\\n")
11 
12 #percent of complete observations
13 cat("Percent of complete observations: ", c / (dim(x))[1] * 100, "\\n")
14 x = na.omit(x) #delete all records with NAs
15 
16 e <- editfile("myfile.txt")
17 
18 sm <- violatedEdits(e, x)
19 summary(sm)
20 plot(sm)
21 
22 boxplot(x[, "Sepal.Length"])
23 boxplot.stats(x[, "Sepal.Length"])
```

The Environment pane on the right lists variables and their types and values:

Variables	Type	Value
x	96 obs. of 5 variables	
y	1 obs. of 0 variables	
age	num [1:17]	20 20 20 20 20 21 21 21 22 22 ...
c	96L	
count	'table' int [1:4(1d)]	5 3 4 3
element	Factor w/ 1 level "Ad_Calc": 1	
fac	Factor w/ 2 levels "Married","Unmarried": 1 2 1...	
height	num [1:6]	165 160 160 165 160 157
height_norm	num [1:200]	160 156 166 162 163 ...
i	10L	
l	num [1:100]	95.5 95.5 95.5 95.5 95.5 ...
m	2L	
Marks	num [1:3]	10 20 30
modeAge	"20"	
n	2L	
Name	chr [1:3]	"ARC" "POR" "XYZ"

The bottom navigation bar includes tabs for Files, Plots, Packages, Help, and Viewer.

## Output:



**Q3. Load the data from wine dataset. Check whether all attributes are standardized or not (mean is 0 and standard deviation is 1). If not, standardize the attributes. Do the same with Iris dataset.**

**For wine dataset:**

The screenshot shows the RStudio interface. The left pane displays a script editor with the following R code:

```
1 setwd("C:\\\\Users\\\\Lenovo\\\\Desktop\\\\DM_files")
2 wine <- read.csv("wine.csv", header=FALSE)
3
4 flag <- 1
5 j <- 1
6 i <- 1
7
8 print("For wine dataset")
9 for(j in 1:length(wine)){
10  if(mean(wine[, j]) != 0 || sd(wine[, j]) != 1){
11    flag <- 0
12  }
13  }
14 j <- j+1
15
16 if(flag == 1){
17  print("Dataset is normalized")
18 } else{
19  print("Normalizing dataset...")
20 }
21
22 data_std <- function(x) (x - mean(x)) / sd(x)
23
24 wine <- data.frame(sapply(wine, data_std))
25
26 for(i in 1:length(wine)){
27  cat("The mean of ", names(wine[i]), "is ", as.integer(mean(wine[,i])), "and standard deviation: ",
28   as.integer(sd(wine[, i])))
29  cat("\n")
30  i <- i+1
31 }
```

The right pane shows the Environment view with the following variable details:

Variable	Type	Description
wine	178 obs. of 14 variables	Global Environment
x	96 obs. of 5 variables	
y	1 obs. of 0 variables	
Values		
age	num [1:17]	20 20 20 20 21 21 21 22 22 ...
c	96L	
count	'table' int [1:4(1d)]	5 3 4 3
element	Factor w/ 1 level "Ad_Calc":	1
fac	Factor w/ 2 levels "Married", "Unmarried":	1 2 1 ...
flag	0	
height	num [1:6]	165 160 160 165 160 157
height_norm	num [1:200]	160 156 166 162 163 ...
i	15	
j	15	
l	num [1:100]	95.5 95.5 95.5 95.5 95.5 ...
m	2L	
variable	num [1:21]	10 20 20

## Output:

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help, and a Project dropdown set to (None). The left sidebar has tabs for Source, Console, Terminal, and Jobs, with the Console tab selected. The main area displays the following R code and its output:

```
C:/Users/Lenovo/Desktop/DM_files/ >
> source('C:/Users/Lenovo/Desktop/DM_files/Q3.r')
[1] "For wine dataset"
[1] "Normalizing dataset..."
The mean of V1 is 0 and standard deviation: 1
The mean of V2 is 0 and standard deviation: 1
The mean of V3 is 0 and standard deviation: 0
The mean of V4 is 0 and standard deviation: 1
The mean of V5 is 0 and standard deviation: 1
The mean of V6 is 0 and standard deviation: 1
The mean of V7 is 0 and standard deviation: 1
The mean of V8 is 0 and standard deviation: 1
The mean of V9 is 0 and standard deviation: 1
The mean of V10 is 0 and standard deviation: 0
The mean of V11 is 0 and standard deviation: 0
The mean of V12 is 0 and standard deviation: 0
The mean of V13 is 0 and standard deviation: 1
The mean of V14 is 0 and standard deviation: 1
> |
```

The right pane, titled "Environment", lists global variables:

Variable	Type	Description
wine	data frame	178 obs. of 14 variables
x	matrix	96 obs. of 5 variables
y	vector	1 obs. of 0 variables
Values	list	age, c, count, element, fac, flag, height, height_norm, i, j, l, m, Marital
age	vector	num [1:17] 20 20 20 20 21 21 21 22 22 ...
c	vector	96L
count	vector	'table' int [1:4(1d)] 5 3 4 3
element	factor	Factor w/ 1 level "Ad_Calc": 1
fac	factor	Factor w/ 2 levels "Married", "Unmarried": 1 2 1 ...
flag	vector	0
height	vector	num [1:6] 165 160 160 165 160 157
height_norm	vector	num [1:200] 160 156 166 162 163 ...
i	vector	15
j	vector	15
l	vector	num [1:100] 95.5 95.5 95.5 95.5 95.5 ...
m	vector	2L
Marital	vector	num [1:21] 10 20 20

## For iris dataset:

The screenshot shows the RStudio interface. The left pane displays an R script named Q3.r with the following code:

```
1 setwd("C:\\\\Users\\\\Lenovo\\\\Desktop\\\\DM_files")
2
3 flag <- 1
4 j <- 1
5 i <- 1
6
7 my_iris <- iris[-c(5)]
8
9 print("For iris dataset")
10 for(j in 1:length(my_iris)){
11   if(mean(my_iris[, j]) != 0 || sd(my_iris[, j]) != 1){
12     flag <- 0
13   }
14   j <- j+1
15 }
16
17 if(flag == 1){
18   print("Dataset is normalized")
19 } else{
20   print("Normalizing dataset...")
21 }
22
23 data_std <- function(x) (x - mean(x)) / sd(x)
24
25 my_iris <- data.frame(sapply(my_iris, data_std))
26
27 for(i in 1:length(my_iris)){
28   cat("The mean of ", names(my_iris[i]), "is ", as.integer(mean(my_iris[,i])), "and standard deviation: ",
29       as.integer(sd(my_iris[, i])))
30   cat("\n")
31   i <- i+1
32 }
```

The right pane shows the Environment view with the following objects listed:

Object	Type	Description
my_iris	150 obs. of 4 variables	Global Environment
repo_content	List of 8	
repo_df	List of 8	
repos	List of 10	
s	num [1:2, 1:2] 5 7 6 8	
smaller	53814 obs. of 10 variables	
Student	6 obs. of 4 variables	
wine	178 obs. of 14 variables	
x	96 obs. of 5 variables	
y	1 obs. of 0 variables	
Values		
age	num [1:17] 20 20 20 20 21 21 21 22 22 ...	
c	96L	
count	'table' int [1:4(1d)] 5 3 4 3	
element	Factor w/ 1 level "Ad_Calc": 1	

## Output:

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help, and a Project dropdown set to (None). The left pane contains tabs for Source, Console, Terminal, and Jobs. The Console tab is active, displaying R code and its output:

```
C:/Users/Lenovo/Desktop/DM_files/03.r
> source('C:/Users/Lenovo/Desktop/DM_files/03.r')
[1] "For iris dataset"
[1] "Normalizing dataset..."
The mean of Sepal.Length is 0 and standard deviation: 1
The mean of Sepal.Width is 0 and standard deviation: 1
The mean of Petal.Length is 0 and standard deviation: 0
The mean of Petal.Width is 0 and standard deviation: 1
> |
```

The right pane shows the Environment view with the Global Environment listed:

Object	Type	Description
my_iris	150 obs. of 4 variables	Global Environment
repo_content	List of 8	
repo_df	List of 8	
repos	List of 10	
s	num [1:2, 1:2] 5 7 6 8	
smaller	53814 obs. of 10 variables	
Student	6 obs. of 4 variables	
wine	178 obs. of 14 variables	
x	96 obs. of 5 variables	
y	1 obs. of 0 variables	
Values		
age	num [1:17] 20 20 20 20 21 21 21 22 22 ...	
c	96L	
count	'table' int [1:4(1d)] 5 3 4 3	
element	Factor w/ 1 level "Ad_Calc": 1	

#### Q4. Run Apriori algorithm to find frequent itemsets and association rules

4.1 Use minimum support as 50% and minimum confidence as 75%

4.2 Use minimum support as 60% and minimum confidence as 60%

The screenshot shows the RStudio interface with the following components:

- Code Pane:** Displays the script `Q4.r` containing R code for performing association rule mining on the "Groceries" dataset using the `arules` package.
- Global Environment:** Shows the following objects and their details:
  - `f`: num [1:2, 1:2] 1 3 2 4
  - `Groceries`: Formal class transactions
  - `iris`: 150 obs. of 5 variables
  - `my_iris`: 150 obs. of 4 variables
  - `repo_content`: List of 8
  - `repo_df`: List of 8
  - `repos`: List of 10
  - `rules`: Formal class rules
  - `s`: num [1:2, 1:2] 5 7 6 8
  - `smaller`: 53814 obs. of 10 variables
  - `Student`: 6 obs. of 4 variables
  - `wine`: 178 obs. of 14 variables
  - `x`: 96 obs. of 5 variables
  - `y`: 1 obs. of 0 variables
- Console:** Shows the command `661 (Top Level) ::`
- Bottom Navigation:** Includes tabs for `Files`, `Plots`, `Packages`, `Help`, and `Viewer`.

## Output:

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Source

Console Terminal Jobs

```
> source('C:/Users/Lenovo/Desktop/DM_files/04.r')
Error in install.packages : Updating loaded packages
[1] "For minimum support = 50%, minimum confidence = 75%"'
Apriori

Parameter specification:
confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext
0.75 0.1 1 none FALSE TRUE 5 0.001 1 10 rules FALSE

Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.06s].
writing ... [777 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
> |
```

lhs	rhs	support	confidence	lift	count
[1] {rice,sugar}	=> {whole milk}	0.001220132	1	3.913649	12
[2] {canned fish,hygiene articles}	=> {whole milk}	0.001118454	1	3.913649	11
[3] {root vegetables,butter,rice}	=> {whole milk}	0.001016777	1	3.913649	10
[4] {root vegetables,whipped/sour cream,flour}	=> {whole milk}	0.001728521	1	3.913649	17
[5] {butter,soft cheese,domestic eggs}	=> {whole milk}	0.001016777	1	3.913649	10
[6] {citrus fruit,root vegetables,soft cheese}	=> {other vegetables}	0.001016777	1	5.168156	10

Environment History Connections

Import Dataset Global Environment

Data

- commits List of 10
- commits\_df 2 obs. of 9 variables
- df 3 obs. of 3 variables
- dfm 6 obs. of 5 variables
- f num [1:2, 1:2] 1 3 2 4
- Groceries Formal class transactions
- repo\_content List of 8
- repo\_df List of 8
- repos List of 10
- rules Formal class rules
- s num [1:2, 1:2] 5 7 6 8
- smaller 53814 obs. of 10 variables
- Student 6 obs. of 4 variables
- y 1 obs. of 0 variables

Files Plots Packages Help Viewer

New Folder Delete Rename More

Home

Name	Size	Modified
.RData	537.4 KB	Apr 27, 2020, 8:23 PM
.Rhistory	13.3 KB	Apr 27, 2020, 8:23 PM
admin_app		
Adobe		
algo		
ALGORITHMS PRACTICAL FILE_58038.docx	90.8 KB	May 3, 2019, 6:46 PM
Android-Keylogger-master		
App-Development-Agreement.docx	36.5 KB	Jun 8, 2019, 7:34 PM
arsd		
beach_water.blend	1.7 MB	Dec 10, 2018, 2:52 PM
beach_water.blend1	690.5 KB	Dec 9, 2018, 12:46 PM
cg_assign		
CG_Practical.docx	10.7 MB	May 16, 2020, 6:15 PM
chair.blend	508.5 KB	Oct 28, 2018, 6:23 PM

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ - Go to file/function | Addins -
Source
Console Terminal Jobs
~/r
> source('C:/Users/Lenovo/Desktop/DM_files/04.r')
Error in install.packages : Updating loaded packages
[1] "For minimum support = 60%, minimum confidence = 60%"
A priori

Parameter specification:
confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target ext
  0.6    0.1    1 none FALSE      TRUE     5   0.006    1    10  rules FALSE

Algorithmic control:
filter tree heap memopt load sort verbose
  0.1 TRUE TRUE FALSE TRUE    2   TRUE

Absolute minimum support count: 59

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.04s].
sorting and recoding items ... [109 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.01s].
writing ... [8 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
  lhs                      rhs          support  confidence lift  count
[1] {butter,whipped/sour cream} => {whole milk} 0.006710727 0.6600000 2.583008 66
[2] {butter,yogurt}             => {whole milk} 0.009354347 0.6388889 2.500387 92
[3] {root vegetables,butter}   => {whole milk} 0.008235892 0.6377953 2.496107 81
[4] {tropical fruit,curd}     => {whole milk} 0.006507372 0.6336634 2.479936 64
[5] {tropical fruit,butter}   => {whole milk} 0.006202339 0.6224490 2.436047 61
[6] {tropical fruit,other vegetables,yogurt} => {whole milk} 0.007625826 0.6198347 2.425816 75
>

```

**Q5. Use Naive bayes, K-nearest, and Decision tree classification algorithms and build classifiers. Divide the data set in to training and test set. Compare the accuracy of the different classifiers under the following situations:**

- 5.1 a) Training set = 75% Test set = 25%**
- 5.1 b) Training set = 66.6% (2/3rd of total), Test set = 33.3%**
- 5.2 Training set is chosen by i)hold out method ii)Random subsampling iii)Cross-validation. Compare the accuracy of the classifiers obtained.**
- 5.3 Data is scaled to standard format.**

## Naive Bayes algorithm:

The screenshot shows the RStudio interface with several windows open. The main window displays an R script named Q5.r containing code for a naive bayes classifier. The code includes library imports for klab, e1071, rpart, rpart.plot, caret, class, and plyr. It then defines two training sets: one with 75% of the data and another with 66.6% of the data. Both sets are used to train a naive Bayes model and predict on the remaining test set. The resulting confusion matrices are compared to determine which training set performs better. The final output shows a confusion matrix table with values for each species.

```

1 library("klab")
2 library("e1071")
3 library("rpart")
4 library("rpart.plot")
5 library("caret")
6 library("class")
7 library(plyr)
8
9 #naive bayes
10
11 ##training set 75%
12 s <- sample(150, 113)
13 iris_train <- iris[, -]
14 iris_test <- iris[-s, ]
15
16 model <- naiveBayes(Species~, data = iris_train)
17 pred <- predict(model, iris_test)
18 conf <- confusionMatrix(pred, iris_test$Species)$table
19 acc1 <- ((sum(diag(conf))) / sum(conf)) * 100
20
21 ##training set 66.6%
22
23 s <- sample(150, 100)
24 iris_train <- iris[, -]
25 iris_test <- iris[-s, ]
26
27 model <- naiveBayes(Species~, data = iris_train)
28 pred <- predict(model, iris_test)
29 conf <- confusionMatrix(pred, iris_test$Species)$table
30 acc2 <- ((sum(diag(conf))) / sum(conf)) * 100
31
32 if (acc1 > acc2){
33   cat("Training set of 75% records is better\n")
34 }else{
35   cat("Training set of 66.6% records is better\n")
36 }
37
38 #holdout method
39 s<-sample(150,75)
40 iris_train<-iris[, -]
41 iris_test<-iris[-s, ]
42
43 model<-naiveBayes(Species~,data=iris_train)
44 pred<-predict(model,iris_test)
45 conf<-confusionMatrix(pred,iris_test$Species)$table
46

```

Console

	train	x
Values	135 obs. of 6 variables	150 obs. of 4 variables
acc	num [1:26] 92 94.6 94.5 94.4 93 ...	
acc1	91.8918918918919	
acc2	98	
acc66	0	
acc75	94.5945945945946	
acc1d	92	
accrs	98	
accsd	95	
accu1	28.5714285714286	
accu2	16.66666666666667	
accu66	92	
accusd	0	
accv	94	
acc	0.913300867022	

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Script Editor (Q5.r):**

```

42 preu<-predict(model,iris_test)
43 conf<-confusionMatrix(pred,iris_test$Species)$table
44 acc1d<-(sum(diag(conf))/sum(conf))*100
45 cat("Accuracy of holdout method is: ", acc1d, "%\n")
46
47 #random subsampling
48 i <- 75
49 j <- 1
50 acc <- c()
51
52 for(i in 75: 100){
53   s <- sample(150, i)
54   iris_train <- iris[s, ]
55   iris_test <- iris[-s, ]
56
57   model<-naiveBayes(Species~.,data=iris_train)
58   pred<-predict(model,iris_test)
59   conf<-confusionMatrix(pred,iris_test$Species)$table
60   acc[j]<-c((sum(diag(conf))/sum(conf))*100,acc)
61   j <- j+1
62 }
63
64 accrs <- mean(acc)
65 cat("Accuracy of random subsampling is: ", accrs, "%\n")
66
67 #cross validation method
68
69 x <- iris[, -5]
70 y <- iris$Species
71
72 model = train(x, y, 'nb', trControl = trainControl(method = 'cv', number = 10))
73 cons <- table(predict(model$finalModel), x$class, y)
74 accv <- (sum(diag(cons)) / sum(cons)) * 100
75
76 greatest <- max(acc1d, accrs, accv)
77
78 if(greatest==acc1d){
79   cat("Holdout method does best classification\n")
80 }else if(greatest==accrs){
81   cat("Random subsampling method does best classification\n")
82 }else{
83   cat("cross validation method does best classification\n")
84 }
```
- Environment Viewer:**

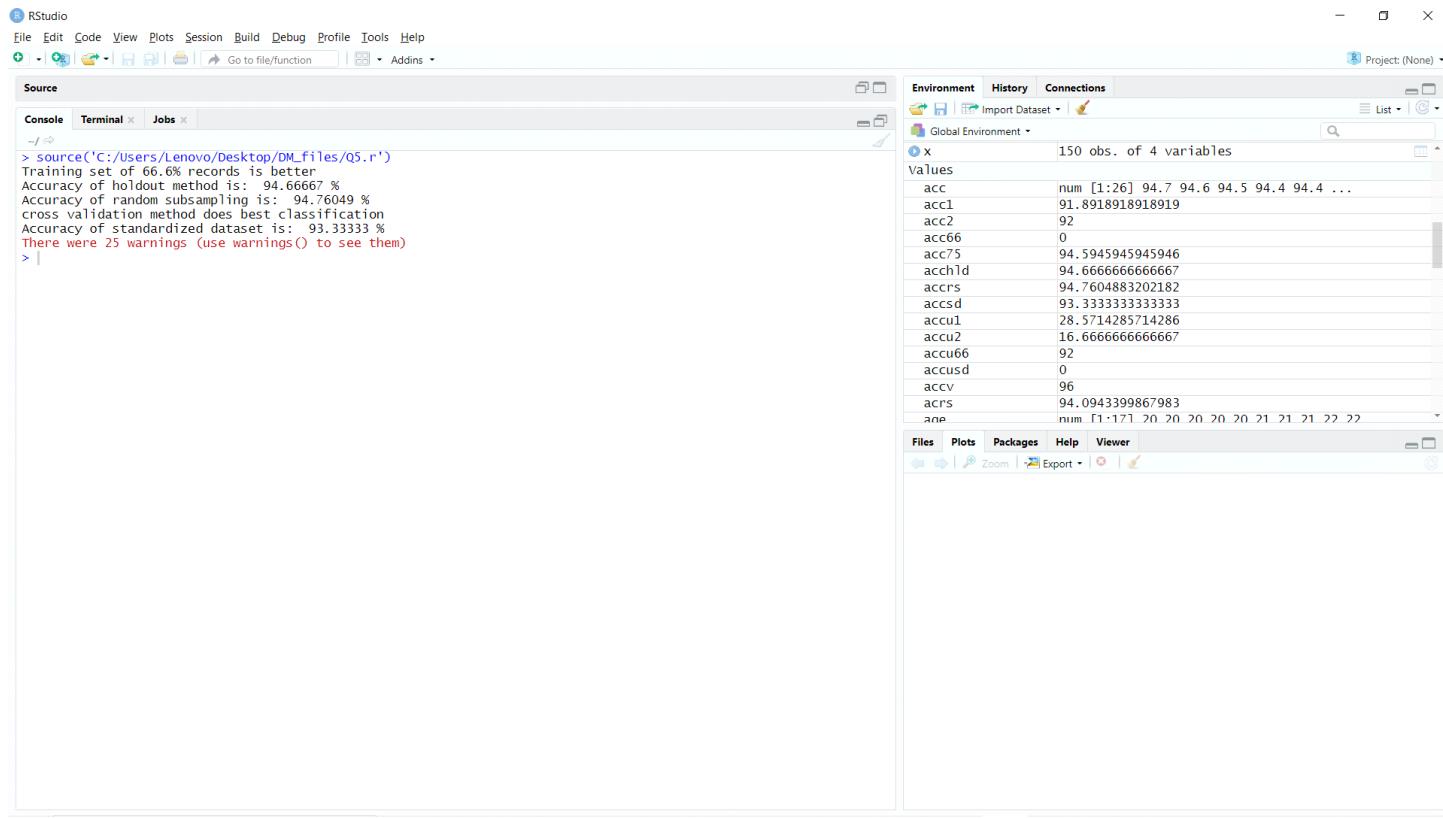
	train	x
Values	135 obs. of 6 variables 91.8918918918919	150 obs. of 4 variables 98
acc	num [1:26] 92 94.6 94.5 94.4 93 ...	
acc1	94.5945945945946	
acc2	98	
acc66	0	
acc75	94.5945945945946	
acc1d	92	
accrs	98	
accsd	95	
accu1	28.5714285714286	
accu2	16.66666666666667	
accu66	92	
accusd	0	
accv	94	
acc	0.001220086702	
- Console:** (Top Level) :
- R Script:** R Script

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Script Editor:** The script editor contains R code for a naive Bayes classifier. The code includes data loading, scaling, model training, and accuracy calculations for different subsampling methods. It also includes a comparison of Holdout, Random Subsampling, and Cross Validation methods.
- Environment Pane:** The environment pane shows the global environment with variables and their values. Key variables include:
  - train**: 135 obs. of 6 variables
  - x**: 150 obs. of 4 variables
  - Values** table:
 

acc	num [1:26] 92 94.6 94.5 94.4 93 ...
acc1	91.8918918918919
acc2	98
acc66	0
acc75	94.5945945945946
acc1d	92
accrs	98
accsd	95
accu1	28.5714285714286
accu2	16.66666666666667
accu66	92
accusd	0
accv	94
accs	94.5945945945946
- Console:** The console pane is visible at the bottom left.

## Output:



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Source

Console Terminal Jobs

```
> source('C:/Users/Lenovo/Desktop/ML_files/05.r')
Training set of 66.6% records is better
Accuracy of holdout method is: 94.66667 %
Accuracy of random subsampling is: 94.76049 %
cross validation method does best classification
Accuracy of standardized dataset is: 93.33333 %
There were 25 warnings (use warnings() to see them)
> |
```

Environment History Connections

Global Environment

x 150 obs. of 4 variables

Values	
acc	num [1:26] 94.7 94.6 94.5 94.4 94.4 ...
acc1	91.8918918918919
acc2	92
acc66	0
acc75	94.5945945945946
acc1ld	94.66666666666667
accrs	94.7604883202182
accsd	93.33333333333333
accul	28.5714285714286
accu2	16.66666666666667
accu66	92
accusd	0
accv	96
acrs	94.0943399867983
ano	num [1:171] 20 20 20 20 20 21 21 21 22 22 ...

Files Plots Packages Help Viewer

Import Dataset | Export |

## K-nearest algorithm:

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Q1.r Q2.r Q3.r Q4.r Q5.r

Go to file/function Addins

Source on Save Run Source

111 #K-nearest  
112 data\_std <- function(x) (x - mean(x)) / sd(x)  
113 iris\_norm <- sapply(iris[, -5], data\_std)  
114 ##training set 75%  
115  
116 iris\_train <- iris\_norm[1:113, ]  
117 iris\_test <- iris\_norm[114:150, ]  
118 pred <- knn(iris\_train, iris\_test, iris[1:113, 5], k = 13)  
119 con <- table(pred, iris[114:150, 5])  
120 acc75 <- ((sum(diag(con))) / sum(con)) \* 100  
121 cat("Accuracy with 75% training set is: ", acc75, "%\n")  
122 ##training set 66.6%  
123  
124 iris\_train <- iris\_norm[1:100, ]  
125 iris\_test <- iris\_norm[101:150, ]  
126 pred <- knn(iris\_train, iris\_test, iris[1:100, 5], k = 13)  
127 con <- table(pred, iris[101:150, 5])  
128 acc66 <- ((sum(diag(con))) / sum(con)) \* 100  
129 cat("Accuracy with 66.6% training set is: ", acc66, "%\n")  
130  
131 #holdout method  
132  
133 iris\_train <- iris\_norm[1:75, ]  
134 iris\_test <- iris\_norm[76:150, ]  
135 iris\_pred <- knn(iris\_train, iris\_test, iris[1:75, 5], k=13)  
136 con<-table(iris\_pred, iris[76:150, 5])  
137 acc1d <- ((sum(diag(con))) / sum(con)) \* 100  
138 cat("Accuracy with holdout is: ", acc1d, "%\n")  
139  
140 #random subsampling  
141  
142 #training sets are 80  
143  
144 iris\_train <- iris\_norm[1:80, ]  
145 iris\_test <- iris\_norm[81:150, ]  
146 iris\_pred <- knn(iris\_train, iris\_test, iris[1:80, 5], k=13)  
147 con <- table(iris\_pred, iris[81:150, 5])  
148 acc1 <- ((sum(diag(con))) / sum(con)) \* 100  
149  
150 #training sets are 90

155:22 (Top Level) :

Environment History Connections

Global Environment

x 150 obs. of 4 variables

Values	
acc	num [1:26] 94.7 94.6 94.5 94.4 94.4 ...
acc1	91.8918918918919
acc2	92
acc66	0
acc75	94.5945945945946
acc1d	94.6666666666667
accrs	94.7604883202182
accsd	93.3333333333333
accul	28.5714285714286
accu2	16.6666666666667
accu66	92
accusd	0
accv	96
accrs	94.0943399867983
ano	num [1:171] 20 20 20 20 20 21 21 21 22 22

Files Plots Packages Help Viewer

Zoom Export

R Script

Console

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Displays R script code for kNN classification, including data loading, model training, prediction, and accuracy calculation for different subsampling methods.
- Environment:** Shows the global environment with variables like `x`, `acc`, `acc1`, etc., and their values.
- Console:** Shows the command `source("Q5.r")` being run.
- Output:** Displays the output of the script, including accuracy percentages and a summary of the best method.

```

153
154
155 #training sets are 90
156
157 iris_train <- iris_norm[1:90, ]
158 iris_test <- iris_norm[91:150, ]
159 iris_pred <- knn(iris_train, iris_test, iris[1:90, 5], k=13)
160 con <- table(iris_pred, iris[91:150, 5])
161 accu2 <- ((sum(diag(con))) / sum(con)) * 100
162
163 accrs <- max(acc1, acc2)
164 cat("Accuracy with random subsampling is: ", accrs , "%\n")
165
166 #cross validation
167
168 x <- iris[, -5]
169 y <- iris$Species
170 res <- knn.cv(x, y, length(y)-1)
171
172 con <- table(res, y)
173 accv<- ((sum(diag(con))) / sum(con)) * 100
174 cat("Accuracy with cross validation is: ", accv , "%\n")
175
176 greatest<-max(acc1,accrs,accv)
177 if(greatest==acc1){
178   cat("Holdout method does best classification.\n")
179 } else if(greatest==accrs){
180   cat("Random subsampling method does best classification.\n")
181 } else{
182   cat("cross validation method does best classification.\n")
183 }
184
185
186 #data scaled to standard format
187
188 sapply(iris[, -5], data_std)
189
190 iris_train <- iris_norm[1:100, ]
191 iris_test <- iris_norm[101:150, ]
192 pred <- knn(iris_train, iris_test, iris[1:100, 5], k = 13)
193 con <- table(pred, iris[101:150, 5])
194 accusd <- ((sum(diag(con))) / sum(con)) * 100
195 cat("Accuracy of standardized dataset is: ", accusd, "%\n")
196
197
198

```

## Output:

The screenshot shows the RStudio interface with the following details:

- Console Tab:** Displays the output of an R script named "05.r". The output includes:
  - Accuracy with 75% training set is: 51.35135 %
  - Accuracy with 66.6% training set is: 0 %
  - Accuracy with holdout is: 33.33333 %
  - Accuracy with random subsampling is: 92 %
  - Accuracy with cross validation is: 0 %
  - Random subsampling method does best classification.
  - Accuracy of standardized dataset is: 0
- Environment Tab:** Shows a global environment with 150 observations and 4 variables. The variables are:

Value	Type	Content
acc	num	[1:26] 94.7 94.6 94.5 94.4 94.4 ...
acc1	num	91.8918918918919
acc2	num	92
acc66	num	0
acc75	num	51.3513513513513
acc1ld	num	33.3333333333333
accrs	num	92
accsd	num	93.3333333333333
accu1	num	28.5714285714286
accu2	num	16.6666666666667
accu66	num	92
accusd	num	0
accv	num	0
acrs	num	94.0943399867983
ane	num	[1:171] 20 20 20 20 20 21 21 21 22 22 ...

## Decision tree algorithm:

The screenshot shows the RStudio interface with an R script open in the left pane and various panes in the right side.

**Script Editor:**

```

199 #Decision tree
200 ##training set 75%
201 s <- sample(150, 113)
202 iris_train <- iris[s, ]
203 iris_test <- iris[-s, ]
204
205 dtm <- rpart(Species~, iris_train, method = "class")
206 rpart.plot(dtm)
207 pred <- predict(dtm, iris_test, type = "class")
208 con <- confusionMatrix(iris_test[, 5], pred$table)
209 acc75 <- (sum(diag(con)) / sum(con)) * 100
210 cat("The accuracy with 75% training data is ", acc75, "%\n")
211
212 ##training set 66.6%
213 s <- sample(150, 100)
214 iris_train <- iris[s, ]
215 iris_test <- iris[-s, ]
216 dtm <- rpart(Species~, iris_train, method="class")
217 rpart.plot(dtm)
218 pred <- predict(dtm, iris_test, type="class")
219 con <- confusionMatrix(iris_test[, 5], pred$table)
220 acc66 <- (sum(diag(con)) / sum(con)) * 100
221 cat("The accuracy with 66.6% training data is ", acc66, "%\n")
222
223 #holdout method
224 s <- sample(150, 75)
225 iris_train <- iris[s, ]
226 iris_test <- iris[-s, ]
227 dtm <- rpart(Species~, iris_train, method="class")
228 rpart.plot(dtm)
229 pred <- predict(dtm, iris_test, type="class")
230 cn <- confusionMatrix(iris_test[, 5], pred$table)
231 acchld <- (sum(diag(cn)) / sum(cn)) * 100
232 cat("The accuracy with holdout method is ", acchld, "%\n")
233
234 #random subsampling
235 i <- 75
236 j <- 1
237 acc <- c()
238 for(i in 75:100){
239   for(j in 1:100){
240     s <- sample(150, 75)
241     iris_train <- iris[s, ]
242     iris_test <- iris[-s, ]
243     dtm <- rpart(Species~, iris_train, method="class")
244     rpart.plot(dtm)
245     pred <- predict(dtm, iris_test, type="class")
246     cn <- confusionMatrix(iris_test[, 5], pred$table)
247     acc <- c(acc, (sum(diag(cn)) / sum(cn)) * 100)
248   }
249 }
250 mean(acc)

```

**Environment:**

Values	150 obs. of 4 variables
acc	num [1:26] 94.7 94.6 94.5 94.4 94.4 ...
acc1	91.8918918918919
acc2	92
acc66	0
acc75	51.3513513513513
acchld	33.3333333333333
accrs	92
accsd	93.3333333333333
accu1	28.5714285714286
accu2	16.6666666666667
accu66	92
accusd	0
accv	0
acrs	94.0943399867983
ane	num [1:171] 20.20.20.20.20.20.21.21.21.22.22 ...

**Console:**

R Script

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Q1.r Q2.r Q3.r Q4.r Q5.r

```

242 dcl <- c()
243 for(i in 75:100){
244   s <- sample(150, i)
245   iris_train <- iris[s, ]
246   iris_test <- iris[-s, ]
247   dtm <- rpart(Species~., iris_train, method="class")
248   rpart.plot(dtm)
249   pred <- predict(dtm, iris_test, type="class")
250   con <- confusionMatrix(iris_test[, 5], pred$table)
251   acc[i] <- c((sum(diag(con)) / sum(con)) * 100, acc)
252   j <- j+1
253 }
254 acrs <- mean(acc)
255 cat("The accuracy with random subsampling method is ", acrs, "%\n")
256
257 #cross validation
258
259 set.seed(123)
260 form <- "Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width"
261 folds <- split(iris, cut(sample(1:nrow(iris)), 10))
262 errs <- rep(NA, length(folds))
263 i <- 1
264
265 for (i in 1: length(folds)){
266   test <- lapply(folds[i], data.frame)
267   train <- lapply(folds[-i], data.frame)
268   tmp.model <- rpart(form, train, method = "class")
269   tmp.pred <- predict(tmp.model, newdata = test, type = "class")
270   conf.mat <- table(test$Species, tmp.pred)
271   errs[i] <- 1 - sum(diag(conf.mat)) / sum(conf.mat)
272 }
273
274 accv <- (1 - mean(errs)) * 100
275 cat("The average accuracy using k-fold cross validation is ", accv, "%\n")
276
277 #comparison of holdout,random subsampling and cross validation
278 greatest <- max(acch1d,acrs,accv)
279 if(greatest == acch1d){
280   cat("Holdout method does best classification\n")
281 }else if(greatest == acrs){
282   cat("Random subsampling method does best classification\n")
283 }else{
284   cat("cross validation method does best classification\n")
285 }
286
287 }

```

150 obs. of 4 variables

Values	
acc	num [1:26] 94.7 94.6 94.5 94.4 94.4 ...
acc1	91.8918918918919
acc2	92
acc66	0
acc75	51.3513513513513
acch1d	33.3333333333333
acrs	92
accsd	93.3333333333333
accu1	28.5714285714286
accu2	16.6666666666667
accu66	92
accusd	0
accv	0
acrs	94.0943399867983
ans	num [1:171] 20.20.20.20.20.20.21.21.21.22.22 ...

Files Plots Packages Help Viewer

Console

```

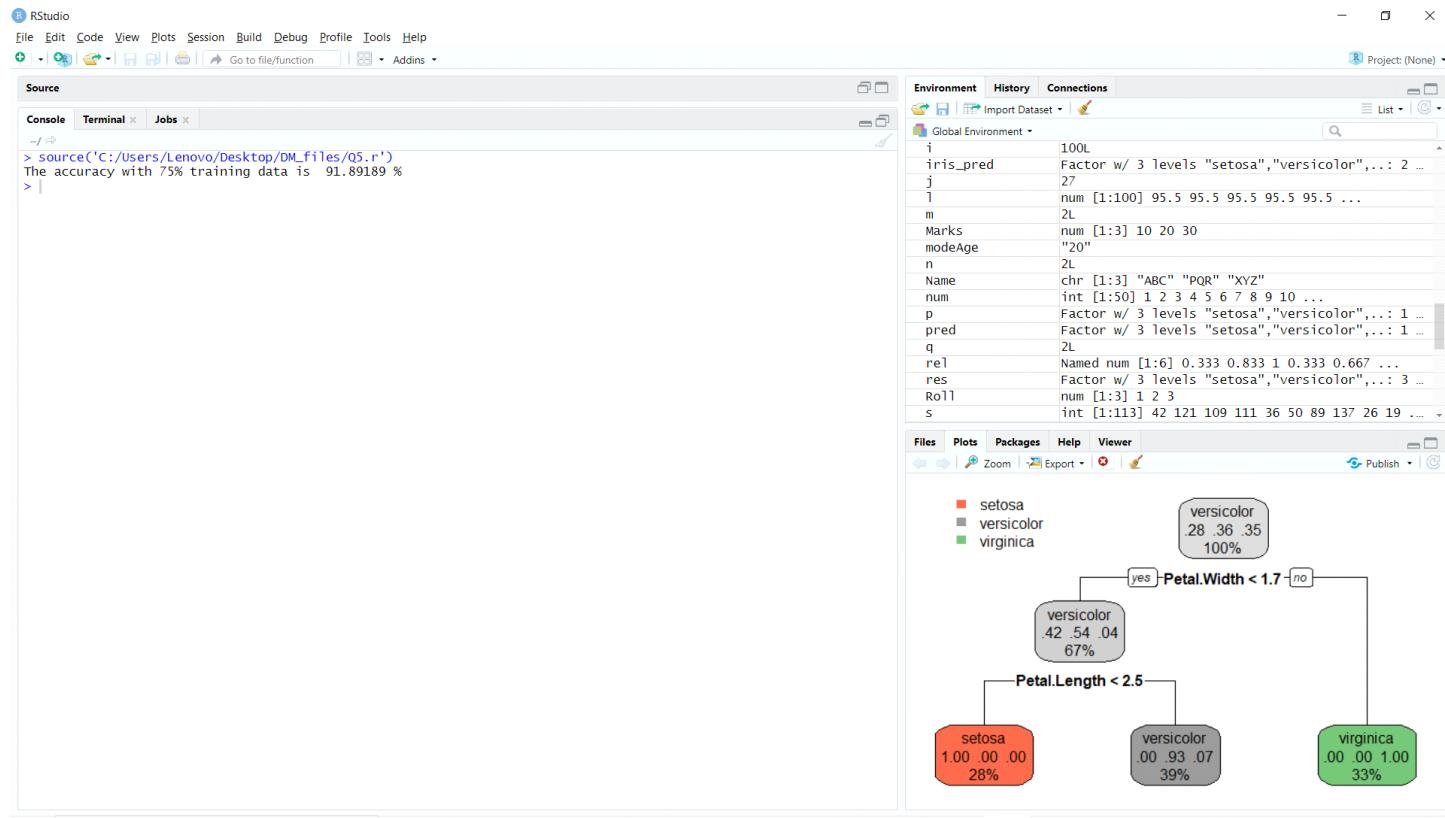
259 set.seed(123)
260 form <- "Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width"
261 folds <- split(iris, cut(sample(1:nrow(iris)), 10))
262 errs <- rep(NA, length(folds))
263 i <- 1
264
265 for (i in 1: length(folds)){
266   test <- lapply(folds[i], data.frame)
267   train <- lapply(folds[-i], data.frame)
268   tmp.model <- rpart(form, train, method = "class")
269   tmp.pred <- predict(tmp.model, newdata = test, type = "class")
270   conf.mat <- table(test$Species, tmp.pred)
271   errs[i] <- 1 - sum(diag(conf.mat)) / sum(conf.mat)
272 }
273
274 accv <- (1 - mean(errs)) * 100
275 cat("The average accuracy using k-fold cross validation is ", accv, "%\n")
276
277 #comparison of holdout, random subsampling and cross validation
278
279 greatest <- max(acc1,acrs,accv)
280 if(greatest == acc1){
281   cat("Holdout method does best classification\n")
282 }else if(greatest == acrs){
283   cat("Random subsampling method does best classification\n")
284 }else{
285   cat("cross validation method does best classification\n")
286 }
287
288 #standard normal form
289
290 data_std <- function(x) (x-mean(x))/sd(x)
291 sapply(iris[,-5],data_std)
292
293 s <- sample(150,90)
294 iris_train <- iris[s, ]
295 iris_test <- iris[-s, ]
296 dtm <- rpart(Species~., iris_train, method="class")
297 rpart.plot(dtm)
298 p <- predict(dtm, iris_test, type="class")
299 con <- confusionMatrix(iris_test[,5], p)$table
300 accsd <- (sum(diag(con)) / sum(con)) * 100
301 cat("The accuracy in standardised iris data is ", accsd, "%\n")
302

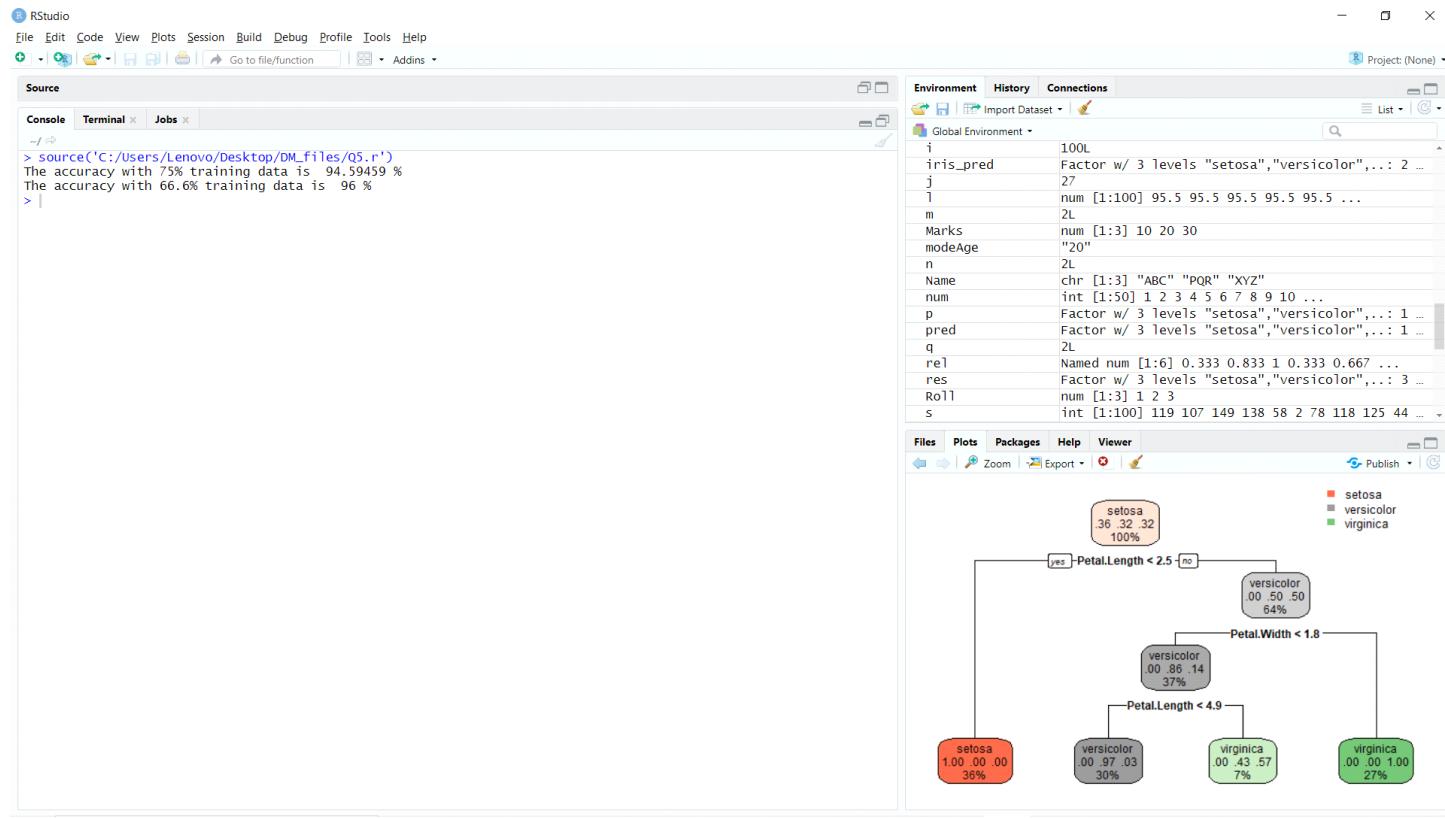
```

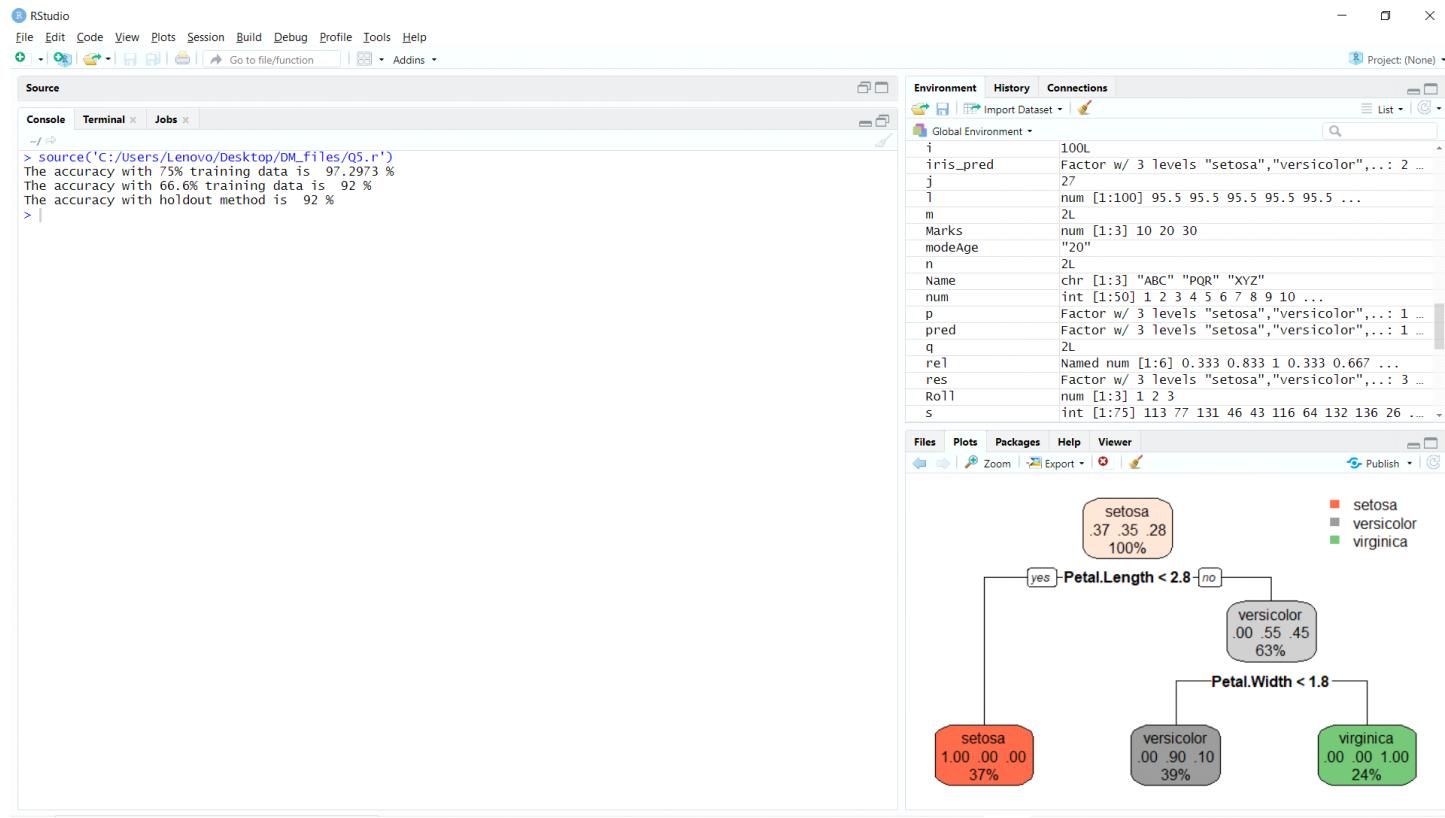
243:18 (Top Level) R Script

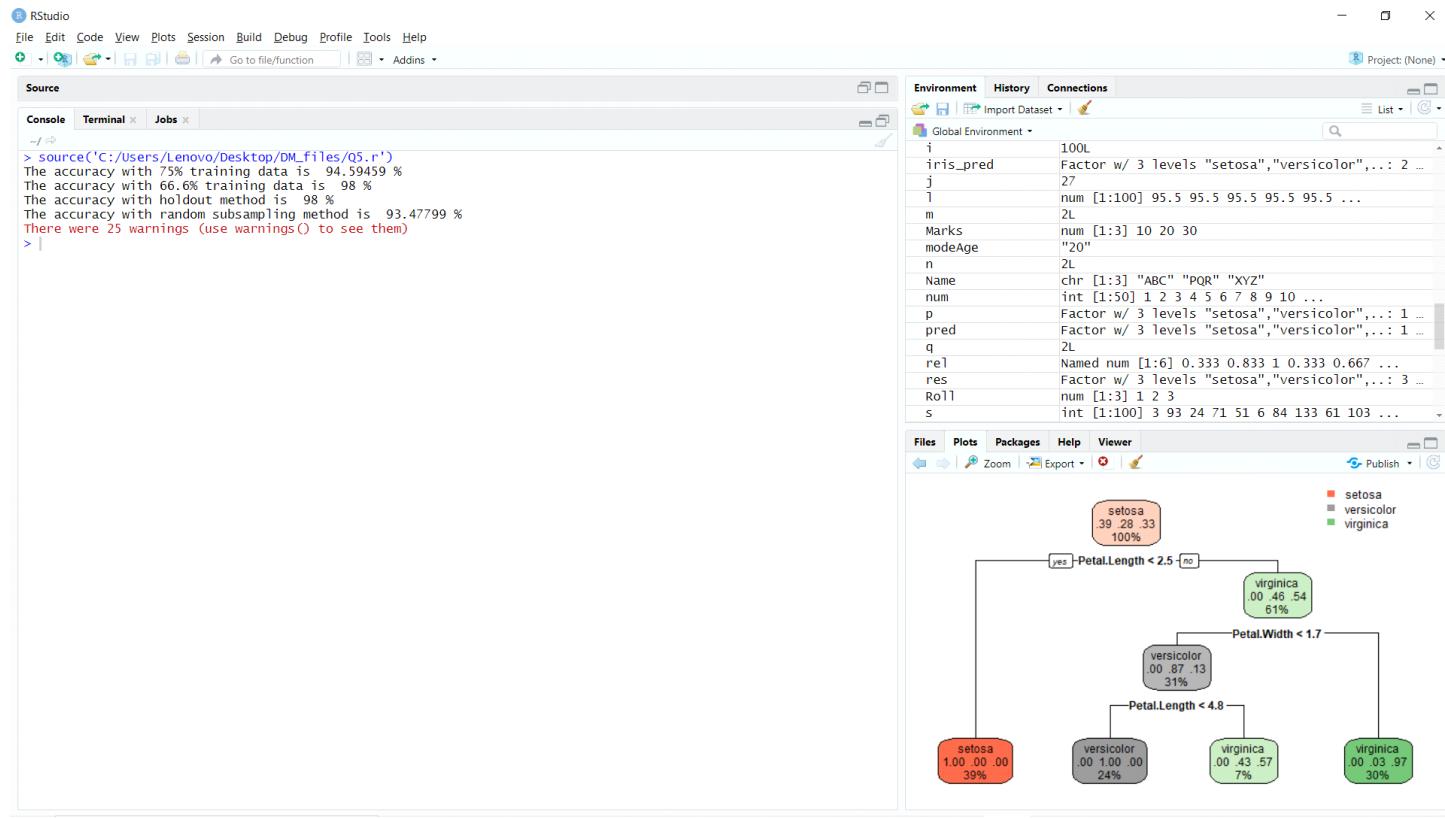
Console

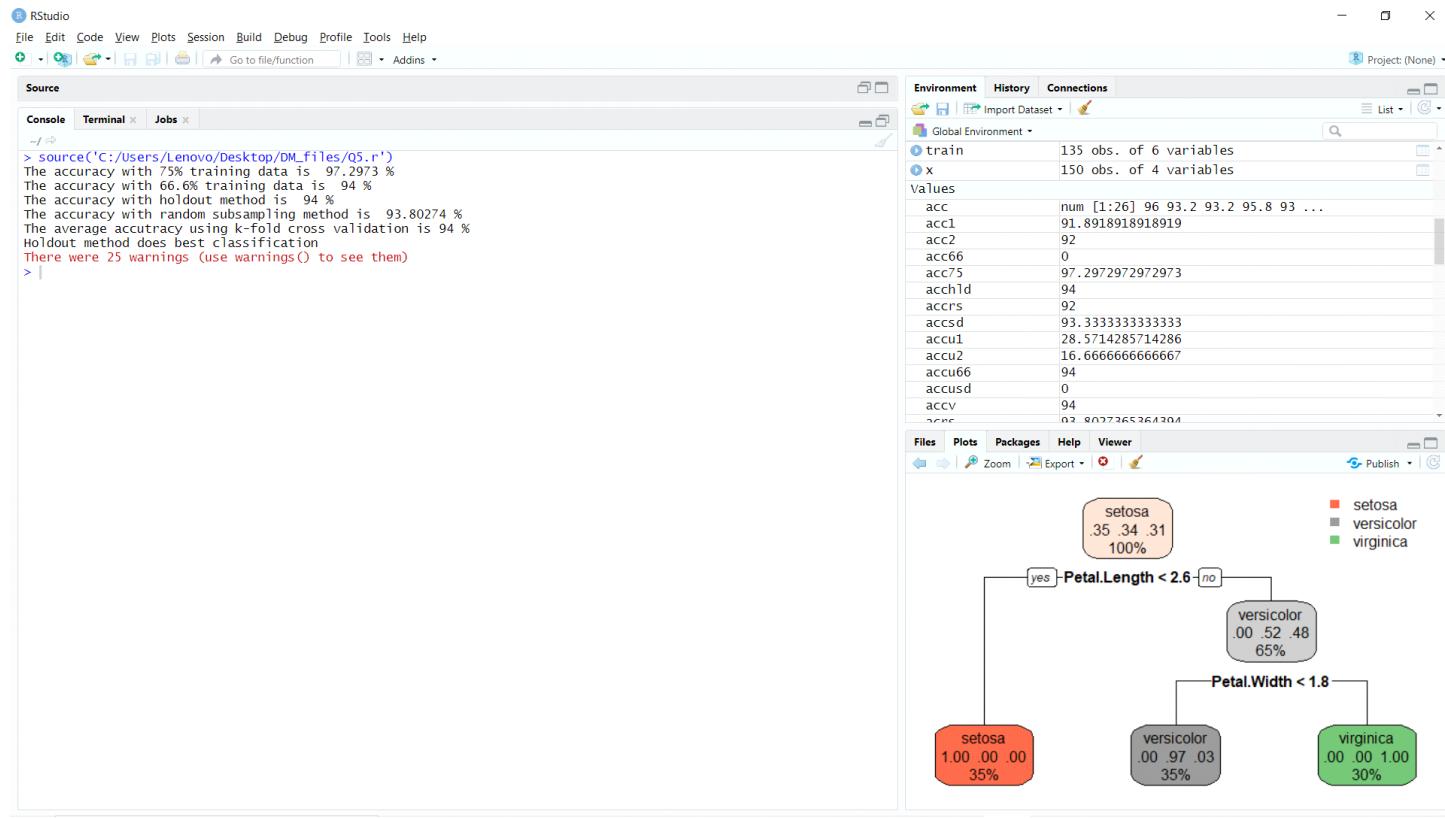
## Output:

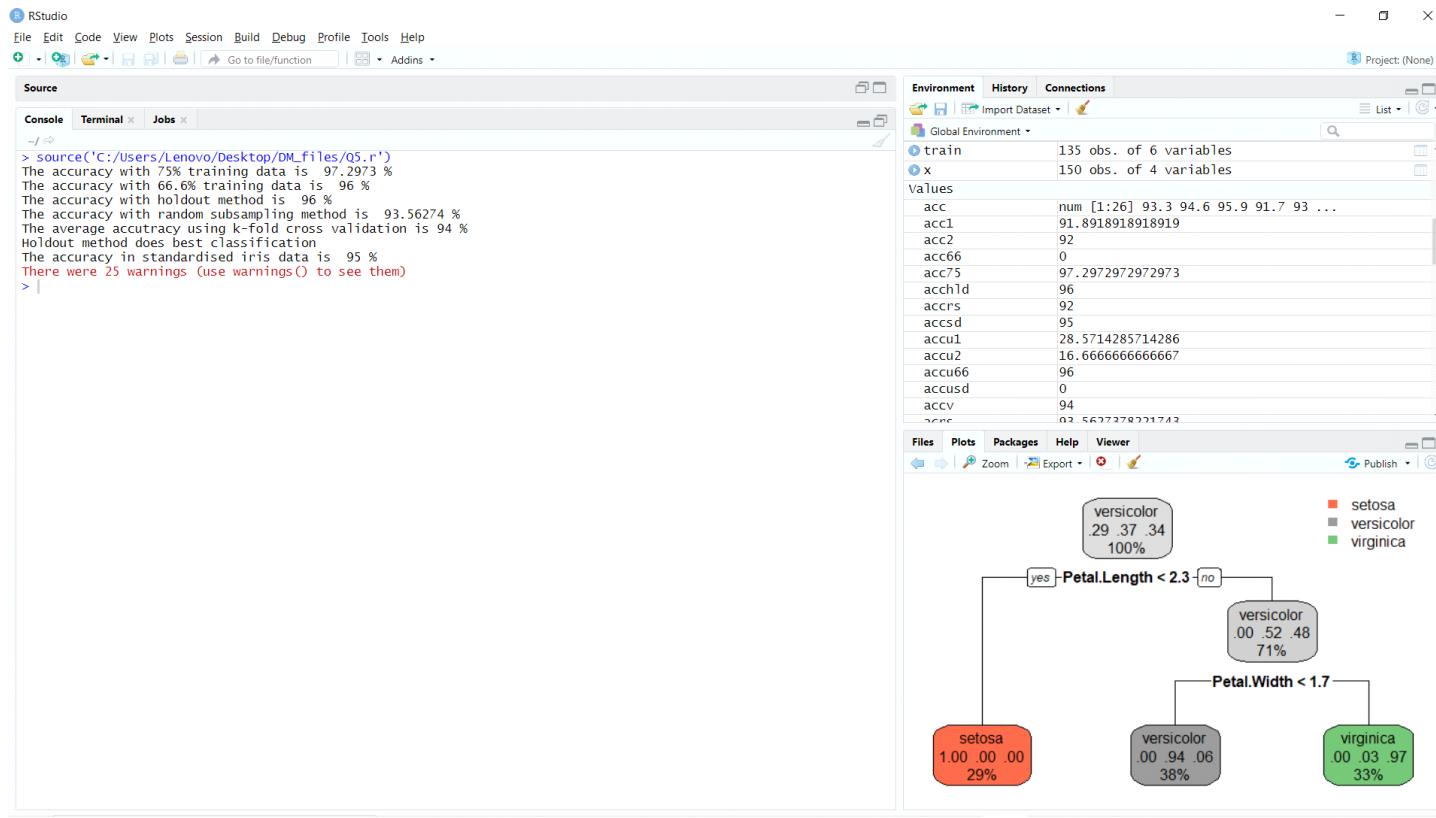












**Q6. Use Simple K-means, DBScan, and Hierarchical clustering algorithms for clustering. Compare the performance of clusters by changing the parameters involved in the algorithms.**

## K-means algorithm:

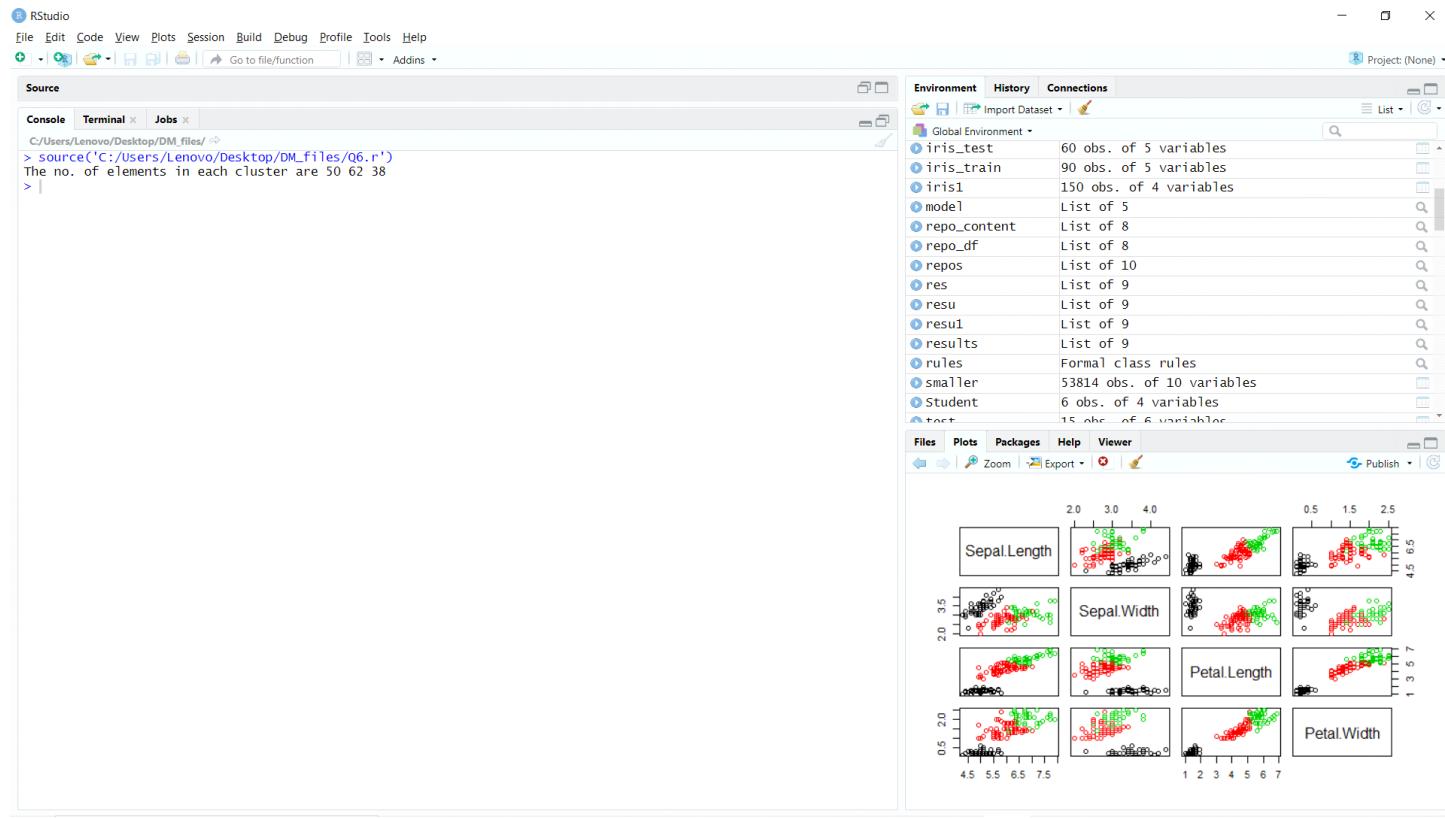
The screenshot shows the RStudio interface. The left pane displays an R script with code for data analysis, including library imports, dataset loading, and clustering. The right pane shows the Environment browser with various objects listed.

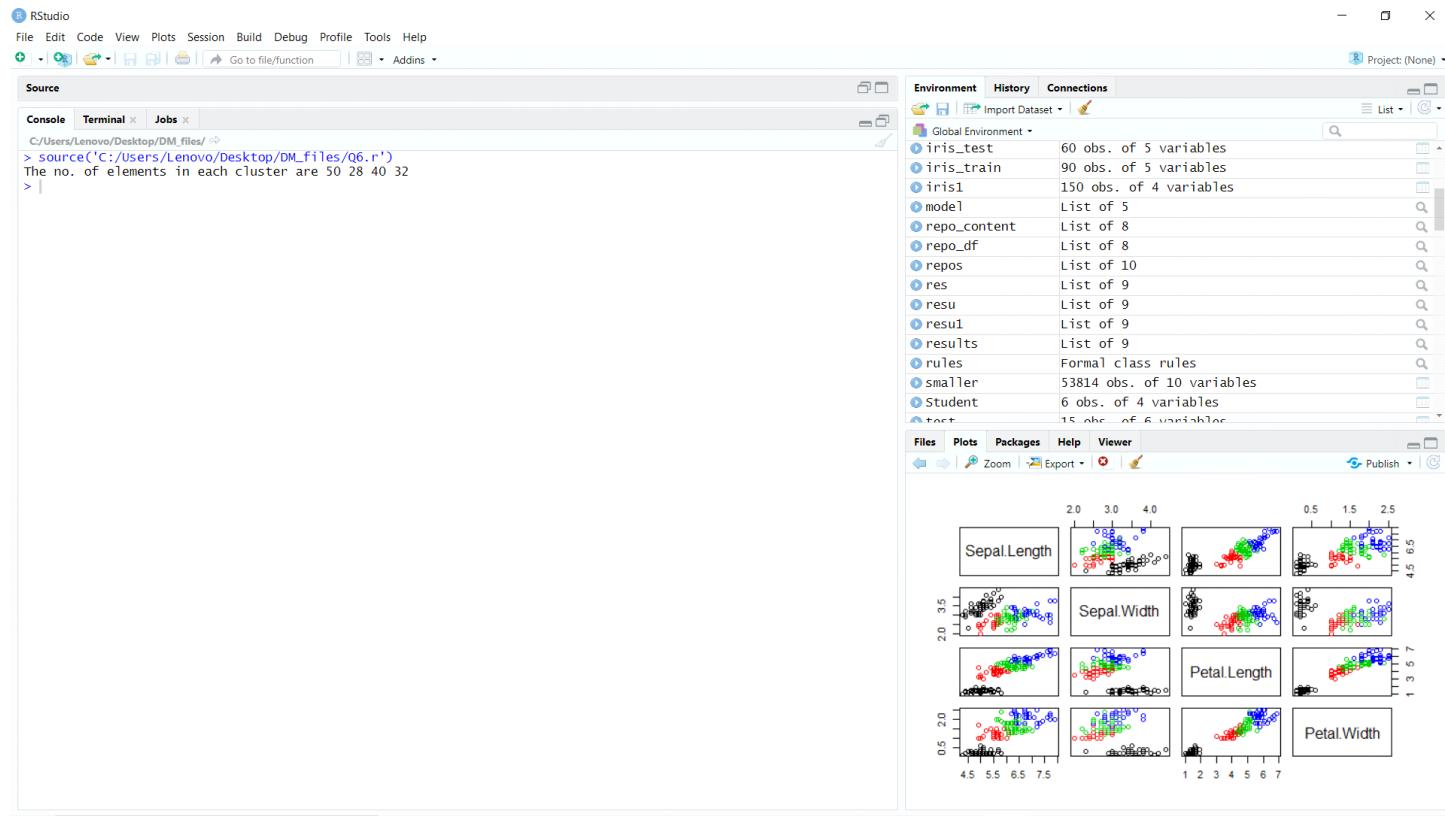
```
library(dbscan)
library(rpart)
library(rpart.plot)
library(caret)
library(e1071)
library(class)
library(cluster)
setwd("C:\\Users\\Lenovo\\Desktop\\DM_files")
#using iris
iris1 <- iris[, -5]
results <- kmeans(iris1, 3)
table(iris$Species, results$cluster)
cat("The no. of elements in each cluster are", table(results$cluster))
plot(iris[, -5], col = results$cluster)
res <- kmeans(iris1, 4)
table(iris$Species, res$cluster)
cat("The no. of elements in each cluster are", table(res$cluster))
plot(iris[, -5], col = res$cluster)
#using htru_2
htru <- read.csv("HTRU_2.csv", header = FALSE)
names(htru) <- c("Profile_mean", "Profile_stdev", "Profile_skewness", "DM_mean", "DM_stdev", "DM_kurtosis")
htru1 <- htru[, 1:length(htru)-1]
resu <- kmeans(htru1, 3)
table(htru$class, resu$cluster)
cat("The no. of elements in each cluster are", table(resu$cluster))
plot(htru[, -length(htru)], col = resu$cluster)
```

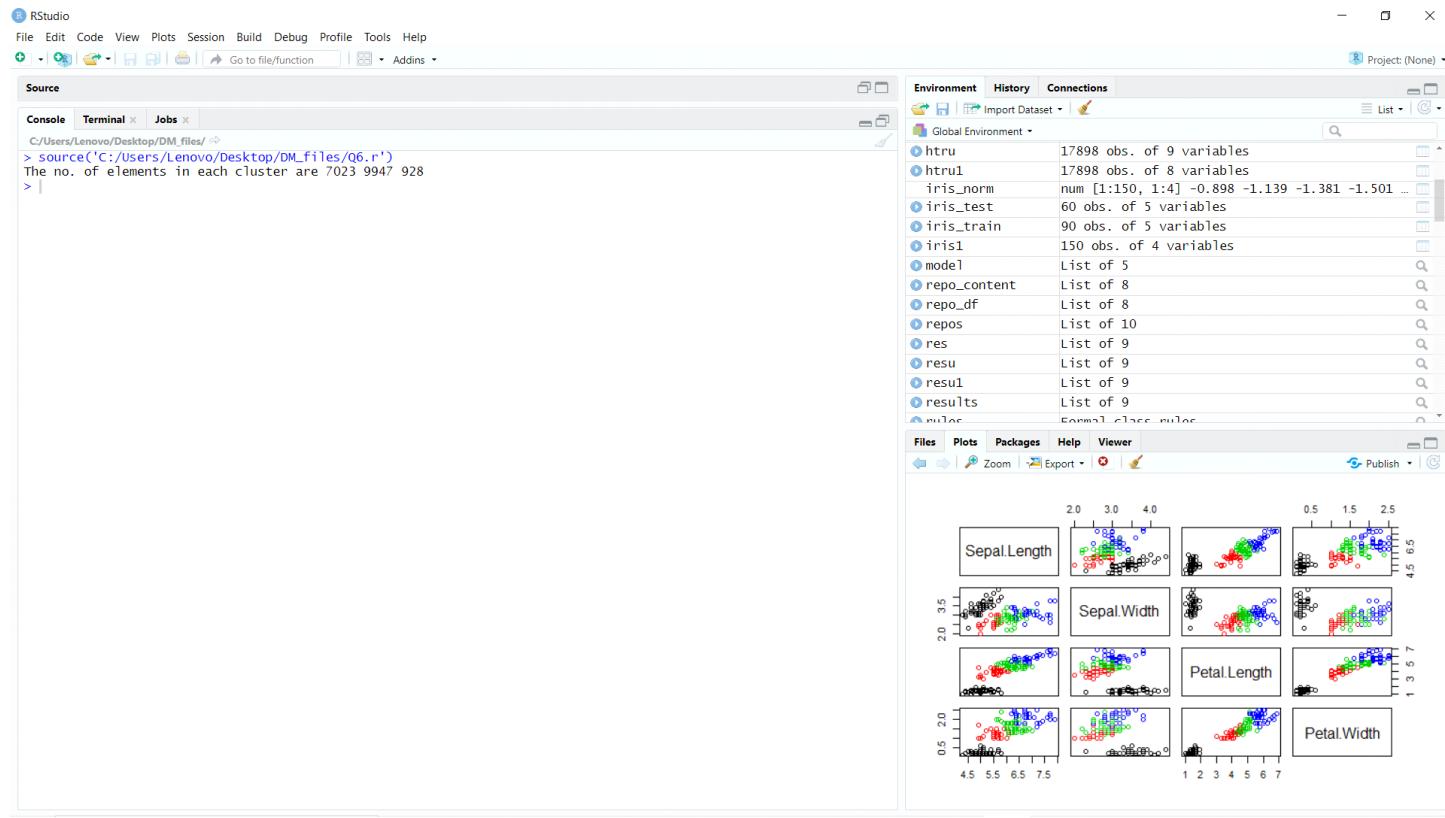
Environment Browser:

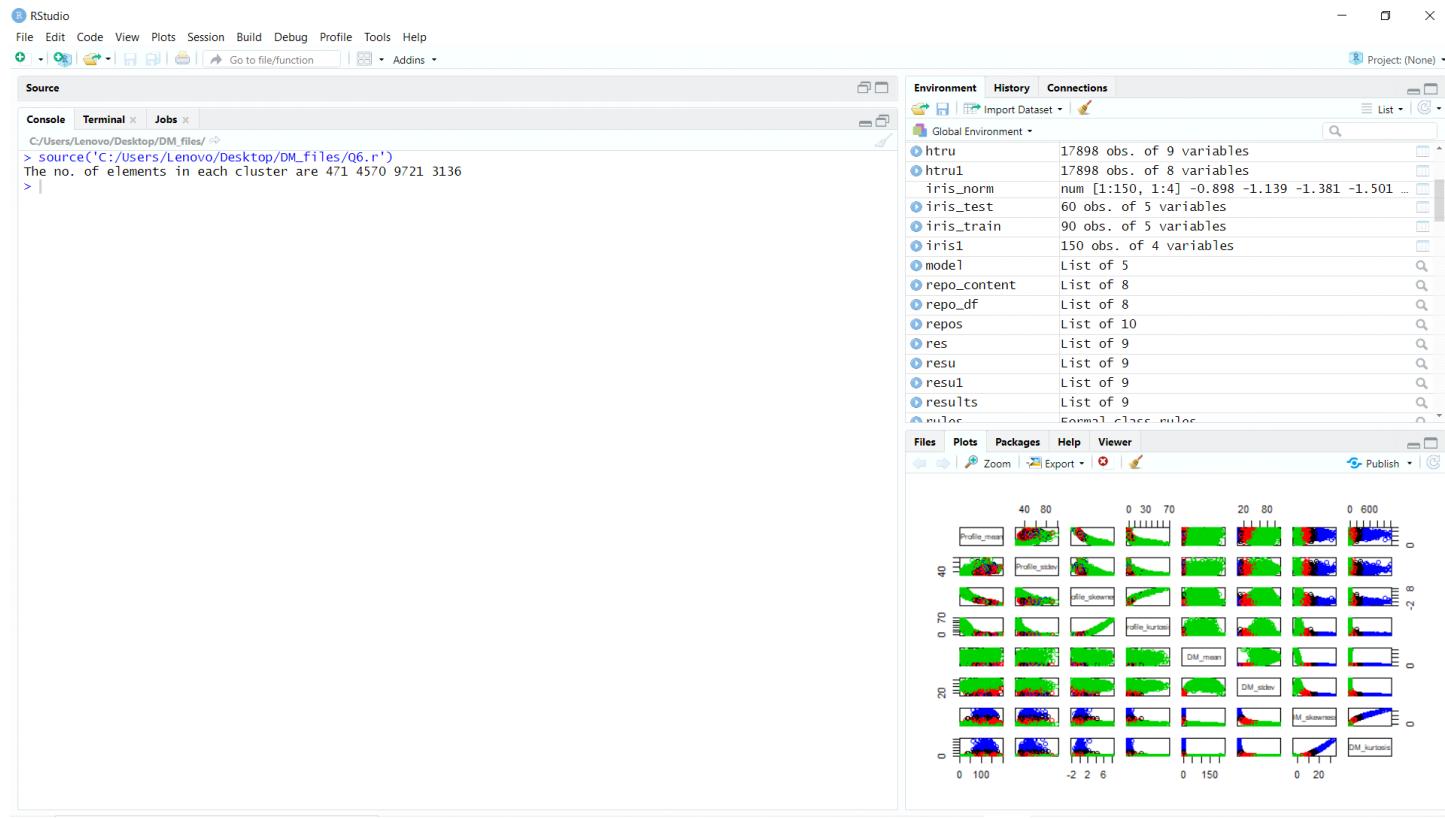
Object	Type	Description
iris_test	60 obs. of 5 variables	
iris_train	90 obs. of 5 variables	
iris1	150 obs. of 4 variables	
model	List of 5	
repo_content	List of 8	
repo_df	List of 8	
repos	List of 10	
res	List of 9	
resu	List of 9	
resu1	List of 9	
results	List of 9	
rules	Formal class rules	
smaller	53814 obs. of 10 variables	
Student	6 obs. of 4 variables	
test	15 obs. of 6 variables	

## Output:







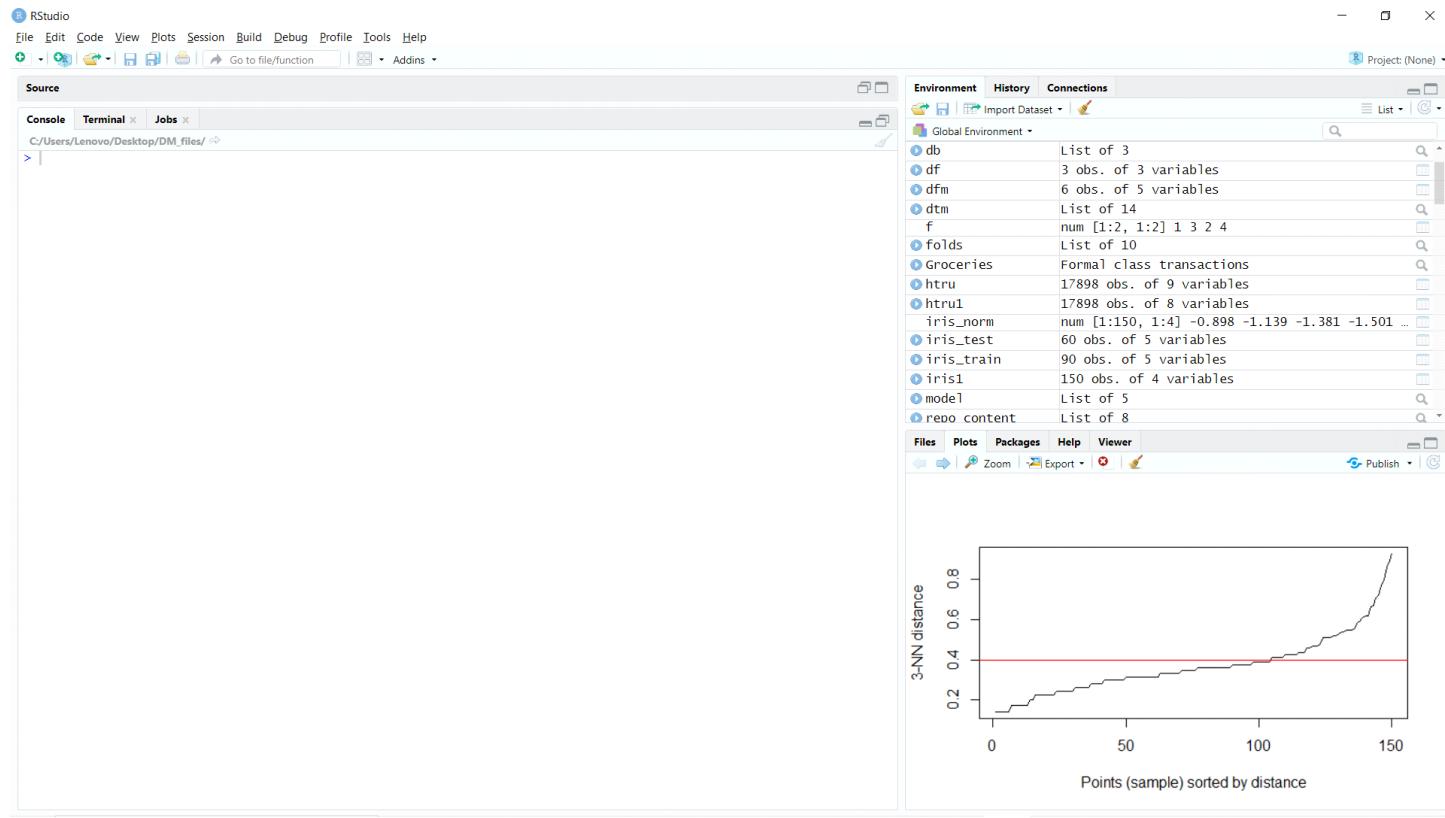


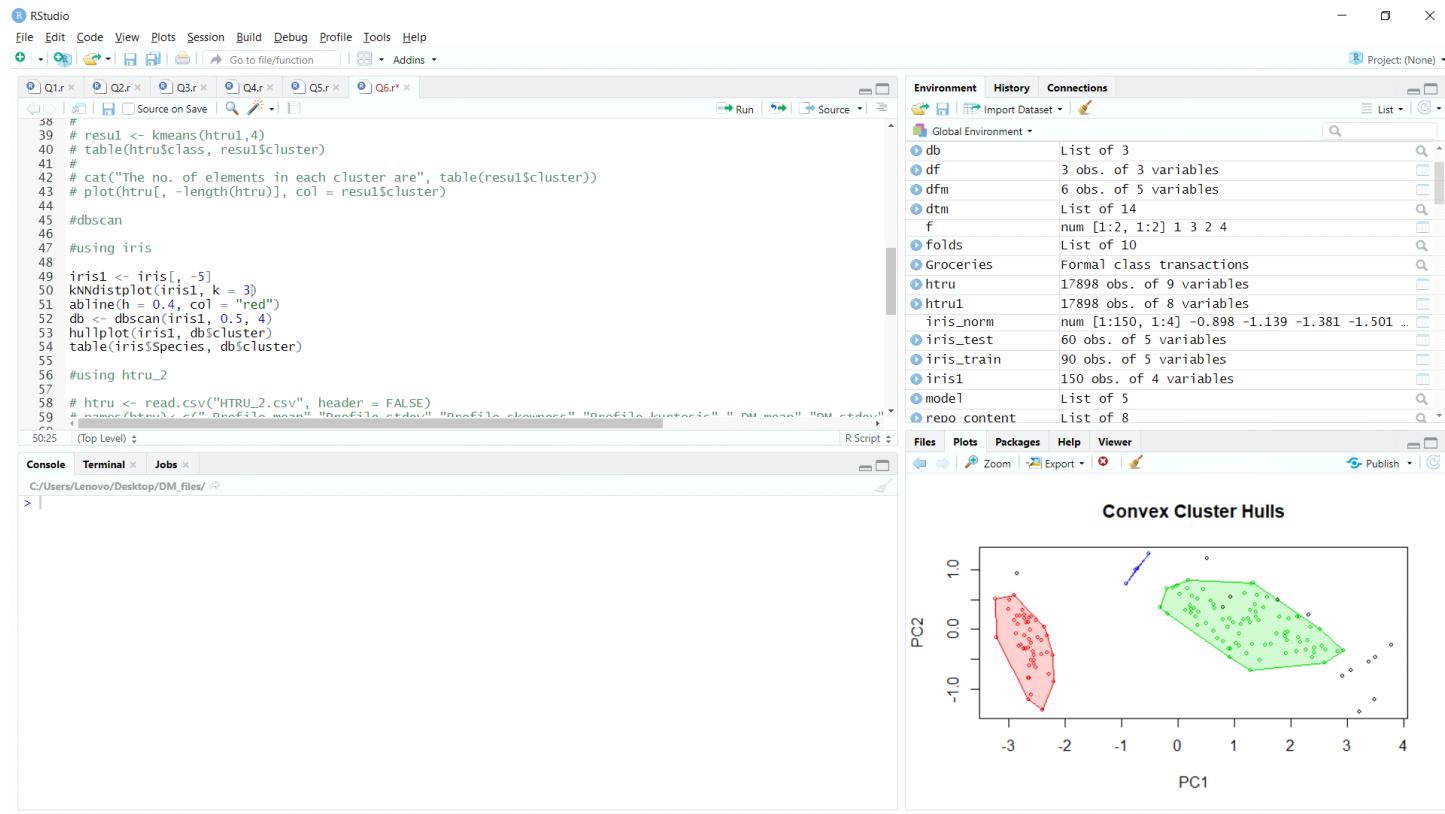
## DBScan algorithm:

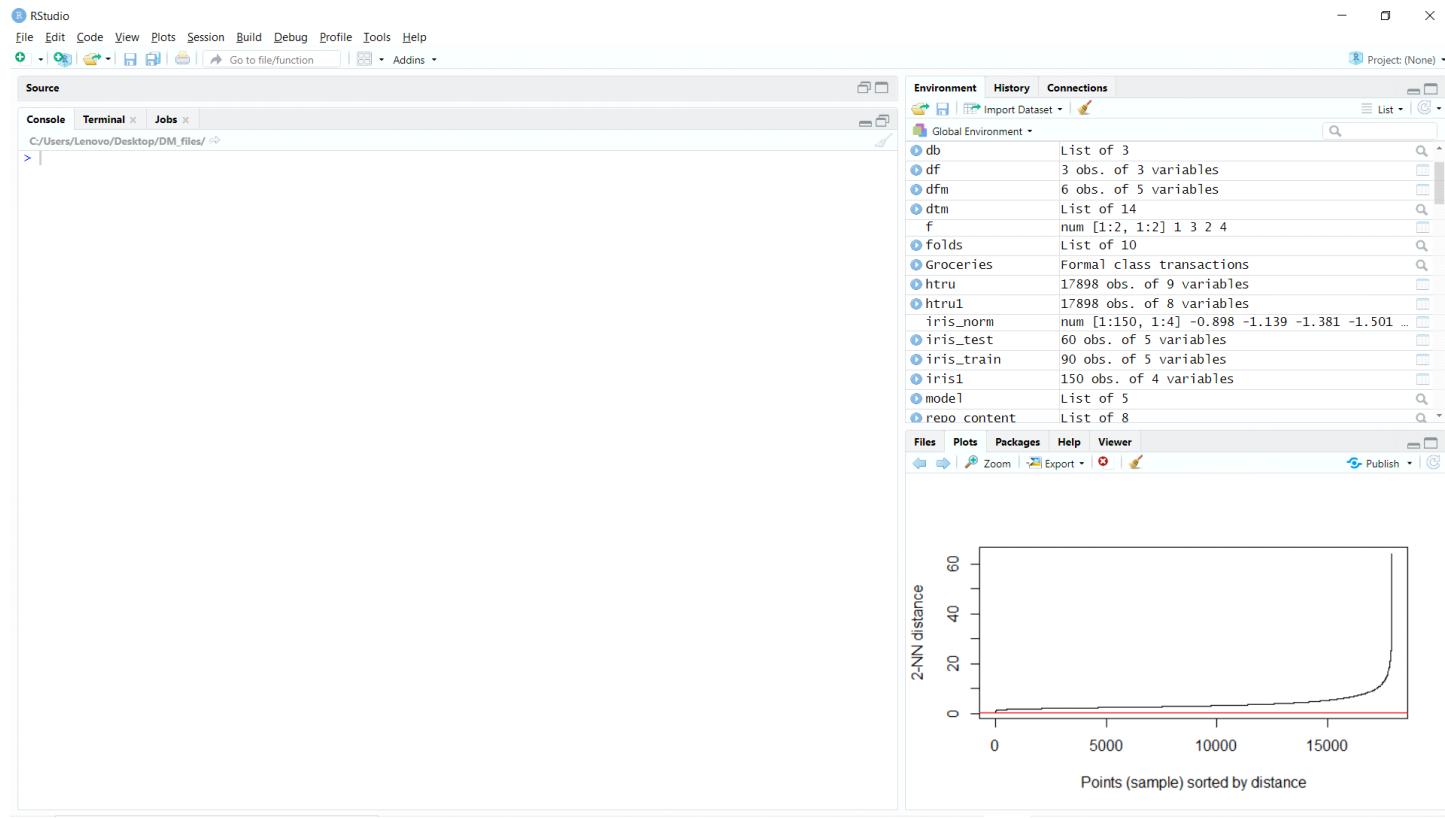
The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help, and a Project dropdown set to (None). The code editor on the left contains an R script with several lines of code related to clustering and visualization. The right panel displays the Global Environment browser, listing various objects such as db, df, ddm, dtm, folds, f, Groceries, htru, htru1, iris\_norm, iris\_test, iris\_train, iris1, model, and rebo, along with their types and dimensions. The bottom navigation bar includes Files, Plots, Packages, Help, and Viewer.

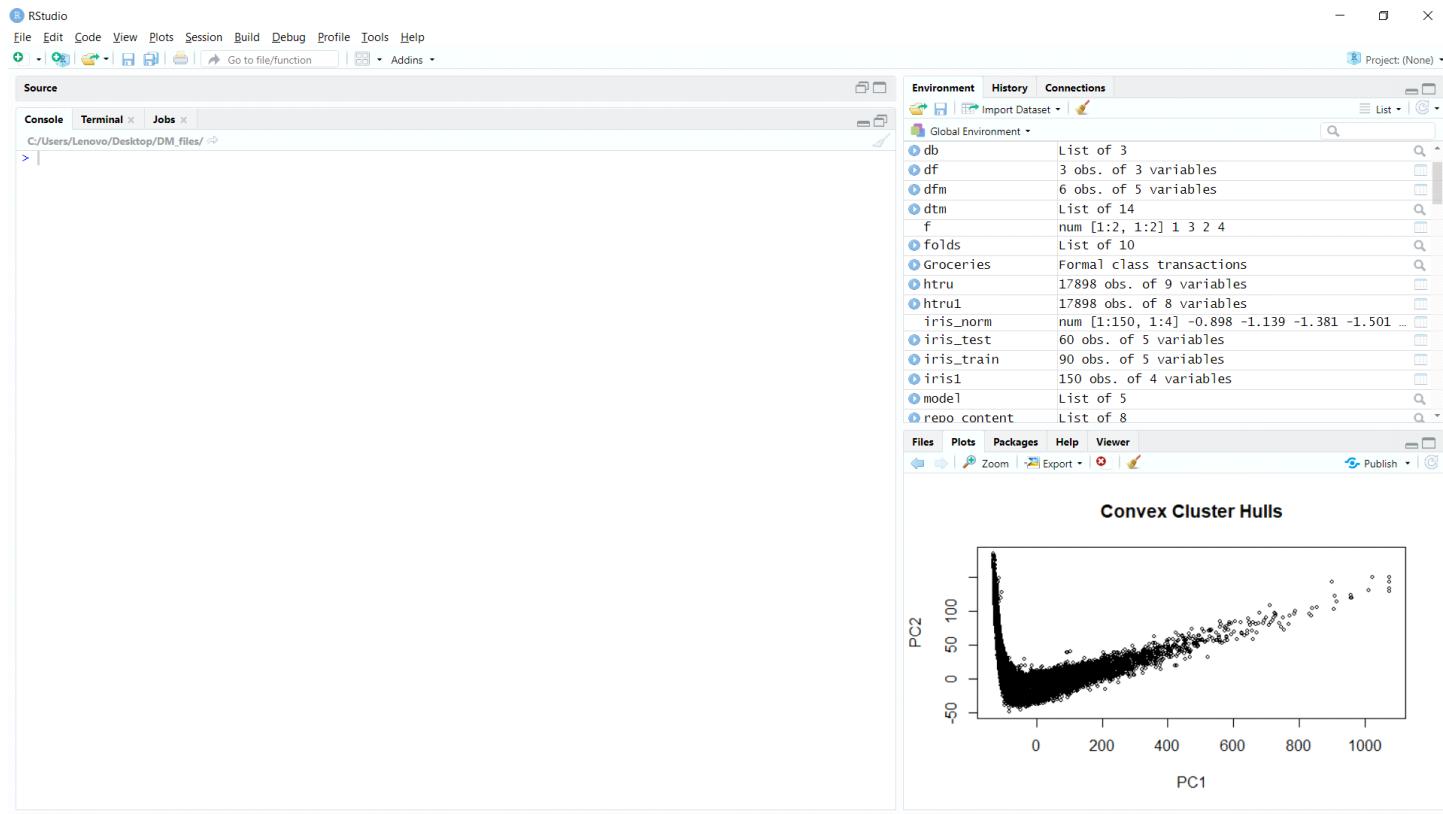
```
44
45 #dbscan
46
47 #using iris
48
49 iris1 <- iris[, -5]
50 knndistplot(iris1, k = 3)
51 abline(h = 0.4, col = "red")
52 db <- dbscan(iris1, 0.5, 4)
53 hullplot(iris1, db$cluster)
54 table(iris$Species, db$cluster)
55
56 #using htru_2
57
58 htru <- read.csv("HTRU_2.csv", header = FALSE)
59 names(htru)<-c("Profile_mean","Profile_stdev","Profile_skewness","Profile_kurtosis","DM_mean","DM_stdev",""
60 htru1 <- htru[ , 1:length(htru)-1]
61 knndistplot(htru1, k = 2)
62 abline(h = 0.4, col = "red")
63 db <- dbscan(htru1, 0.5, 2)
64 hullplot(htru1, db$cluster)
65 table(htru$class, db$cluster)
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
```

## Output:

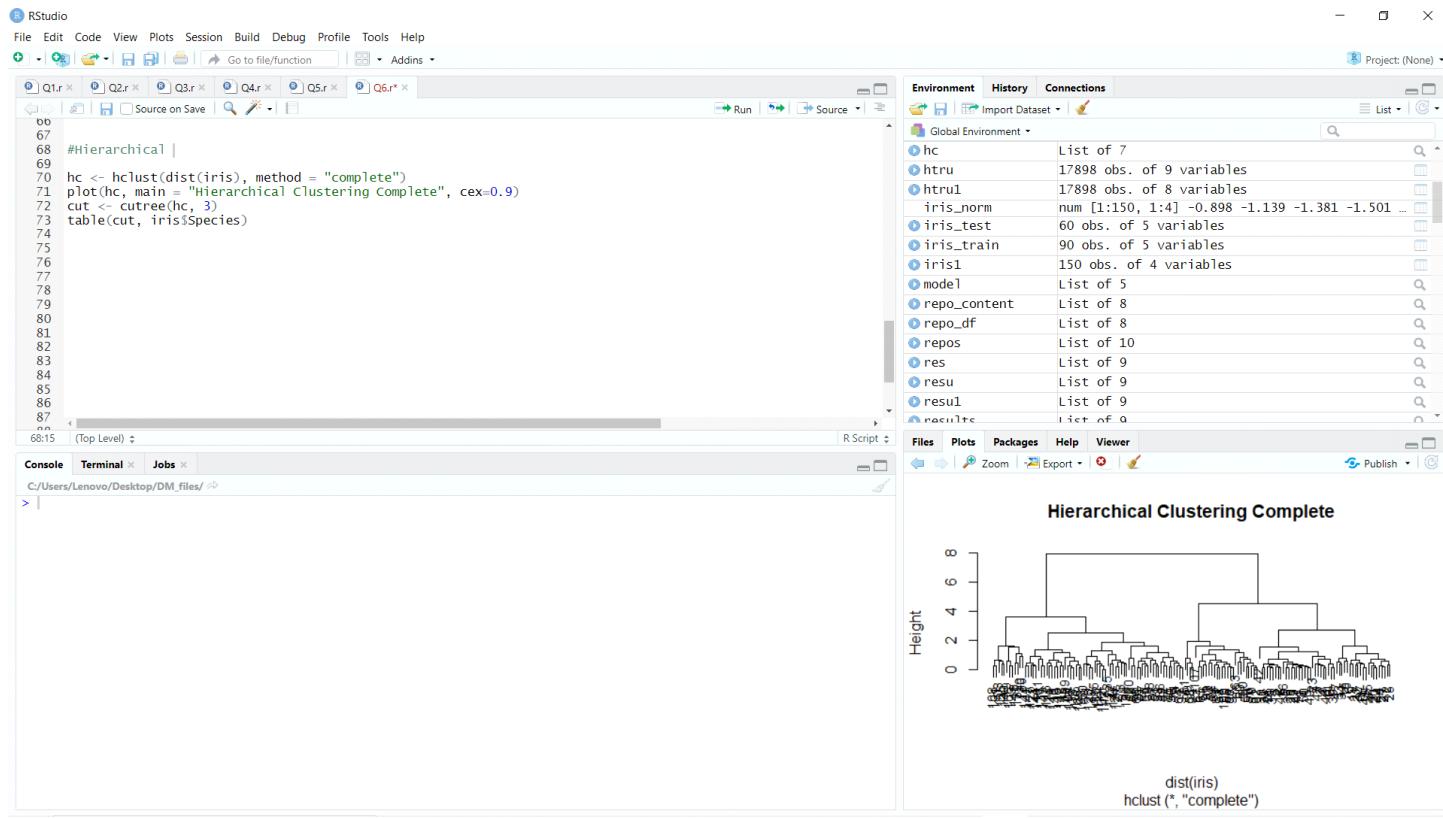








## Hierarchical algorithm:



## Output:

Plot Zoom

- X

Hierarchical Clustering Complete

