Experiment No. 1

Aim : Installation of R and RStudio

Theory :

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

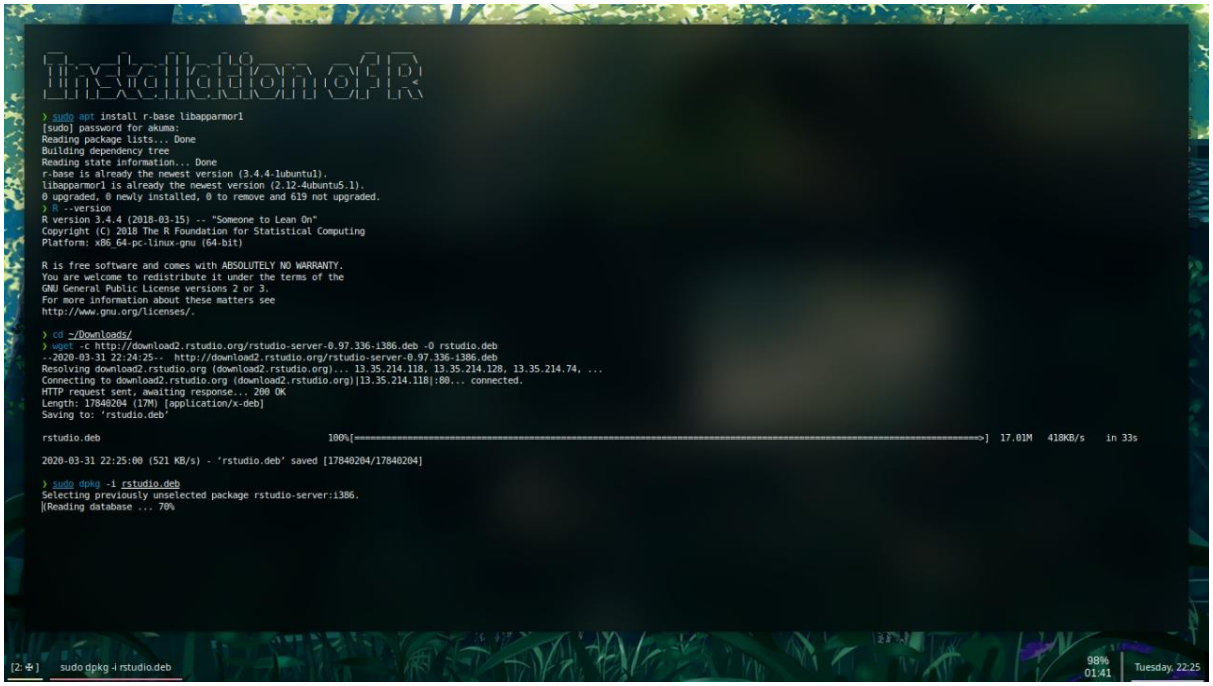
Steps to install R :

1. Open an internet browser and go to www.r-project.org.
2. Click the "download R" link in the middle of the page under "Getting Started."
3. Select a CRAN location (a mirror site) and click the corresponding link.
4. Click on the "Download R for Windows" link at the top of the page.
5. Click on the "install R for the first time" link at the top of the page.
6. Click "Download R for Windows" and save the executable file somewhere on your computer. Run the .exe file and follow the installation instructions.
7. Now that R is installed, you need to download and install RStudio.

Steps to install RStudio :

1. Go to www.rstudio.com and click on the "Download RStudio" button.
2. Click on "Download RStudio Desktop."
3. Click on the version recommended for your system, or the latest Windows version, and save the executable file. Run the .exe file and follow the installation instructions.

Output :



Conclusion : Thus we have installed R and Rstudio.

Experiment No. 2

Aim : Basic Functionality of R - variables, basic arithmetic and help commands.

Theory :

A variable provides us with named storage that our programs can manipulate. A variable in R can store an atomic vector, group of atomic vectors or a combination of many R objects. A valid variable name consists of letters, numbers and the dot or underline characters. The variable name starts with a letter or the dot not followed by a number.

We can name variables, vectors, matrices, functions, and subroutines. You are not restricted by the names you use, as long as they are not reserved for existing functions (like the function *sum()*) in the R package you are using. R is case sensitive, so the variable x is different from the variable X.

The help() function and ? help operator in R provides access to the documentation pages for R functions, data sets, and other objects, both for packages in the standard R distribution and for contributed packages. To access documentation for the standard lm (linear model) function, for example, enter the command help(lm) or help("lm"), or ?lm or ?"lm" (i.e., the quotes are optional).

To access help for a function in a package that's *not* currently loaded, specify in addition the name of the package: For example, to obtain documentation for the rlm() (robust linear model) function in the MASS package, help(rlm, package="MASS").

Output :

Code :

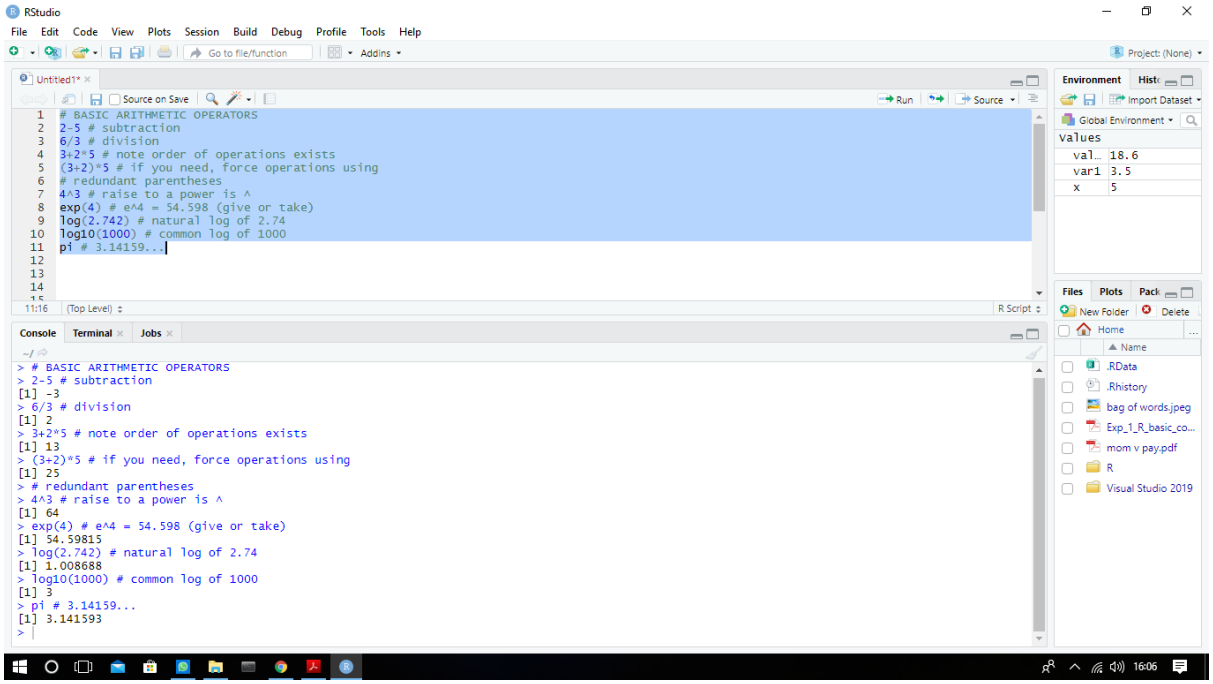
```
# BASIC ARITHMETIC OPERATORS
2-5 # subtraction
6/3 # division
3+2*5 # note order of operations exists
(3+2)*5 # if you need, force operations using
# redundant parentheses
4^3 # raise to a power is ^
exp(4) # e^4 = 54.598 (give or take)
log(2.742) # natural log of 2.74
log10(1000) # common log of 1000
pi # 3.14159...



# VARIABLE NAMES
x <- 5
x # 1 way to print out contents of a variable
```

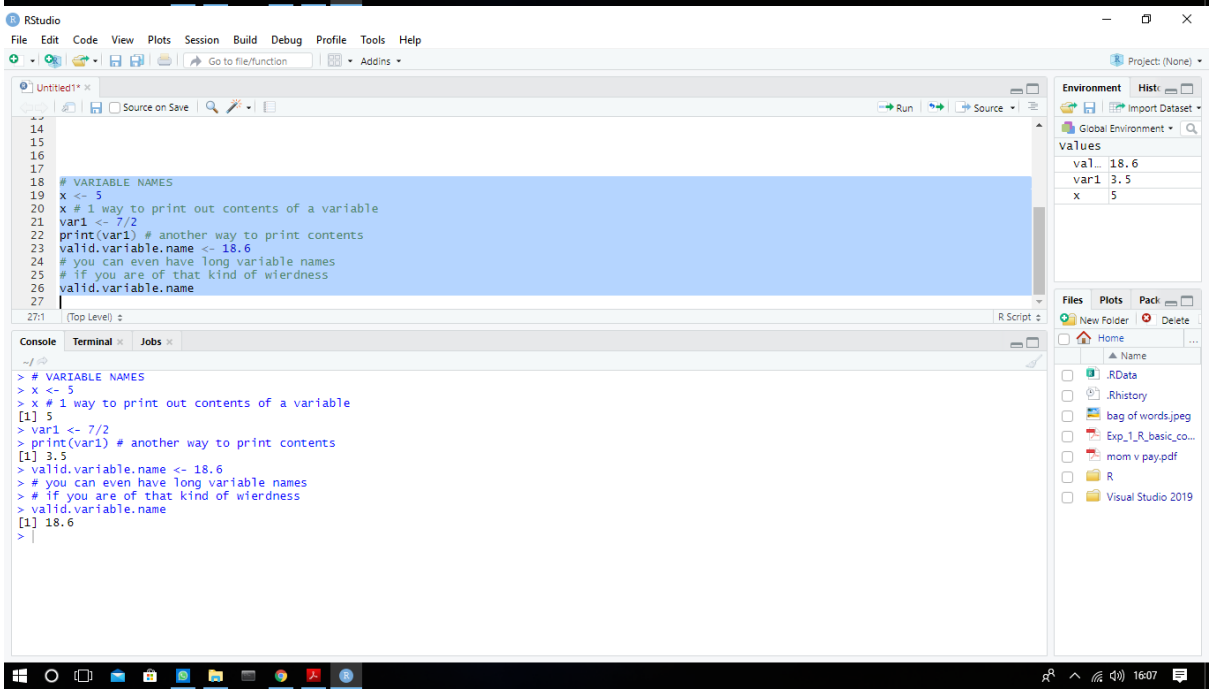
```
var1 <- 7/2
print(var1) # another way to print contents
valid.variable.name <- 18.6
# you can even have long variable names
# if you are of that kind of weirdness
valid.variable.name



a <- 3 # This stores the value 3 into variable a
a
sqrt(a) # square root of a (which is 3 at this moment)
a^4 # a raised to the fourth power
log(a) # natural log of a
log10(a) # common log of a
exp(a) # e to the power a
tan(a) # tangent of a
b <- pi # pie, the number approx 3.14159
b


# VECTORS
x <- c(2,3,5,1,4,4) # create a row vector with those 6 numbers
# and then call the vector x
x
sum(x) # sums the elements of vector x
mean(x) # finds the mean of vector x
sd(x) # finds standard deviation of vector x
median(x) # finds the median of x
sqrt(x) # finds square root of every element of x
x^2 # finds the square of every element of x
seq(1,10) # makes a vector 10 long, from 1 to 10
seq(1,10, 2) # makes a vector from 1 to 10, starting with 1, then skipping 1 ending up with
only odds from 1 to 10
seq(1:10) # same as seq(1,10)
seq(1,10,by=2) # same as seq(1,10,2)
y <- c(1:7) # create a row vector with elements 1 through 6
y
z <- 1:7 # same as y--use when you increment by 1
z
w <- c(1:12,0,-6) # use the c( when you increment by 1 followed by more numbers not in
sequence
w


# SOME OPERATIONS WITH NUMERICAL VECTORS AND LOGICAL VECTORS
x <- c(-5,0,7,-6,14,27) # data vector x
x[1] # prints first element of x
x[length(x)] # prints last element of x
i<-3
```

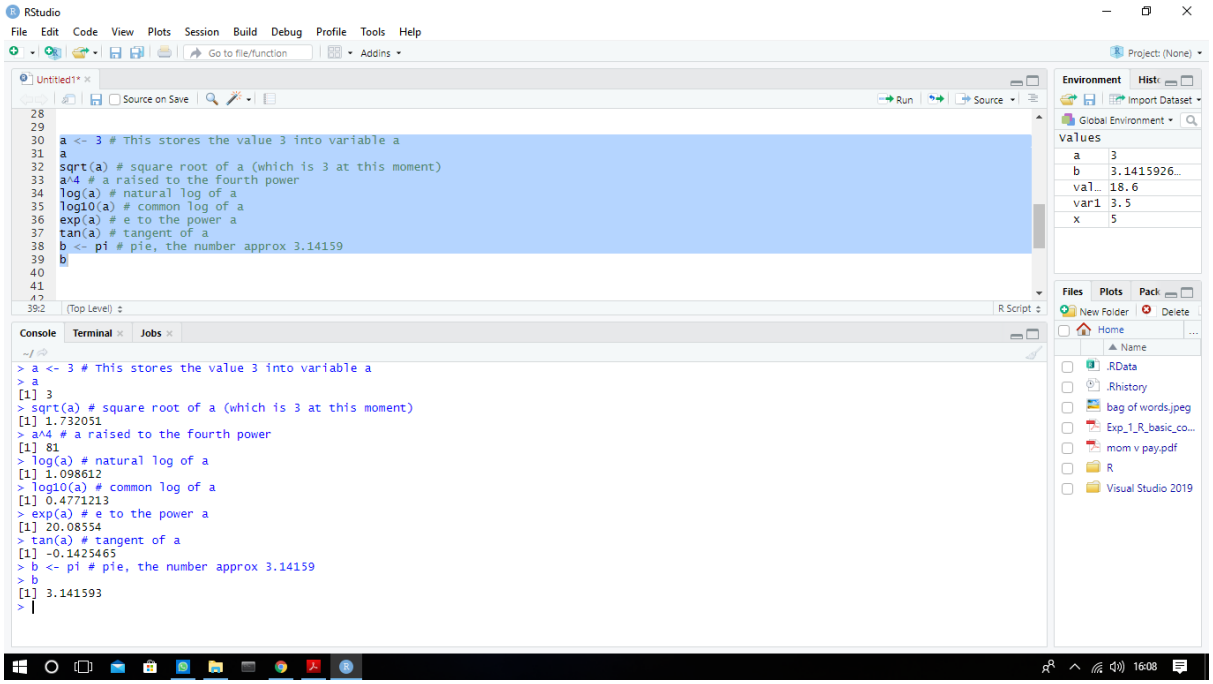
```
x[i] # the ith entry if 1<=i<=n, NA if i>n, all but the ith if -n<=i<=-1, an error if i < -n, and an empty vector if i=0
x[c(2,3)] # the second and 3rd entries
x[-c(2,3)] # all but the second and third entries
x[i] <- 5 # assign a value of 5 to the first entry; also can use x[i] = 5
x[c(1,4)] <- c(2,3) # assign values to the first and fourth entries
y
x[indices] <- y # assign to those indices indicated by the values
# of INDICES: if y is not long enough, values
# are recycled; if y is too long, just its initial
# values are used and a warning is issued
x
x < 3 # vector with length n of TRUE or FALSE # depending if x[i] < 3 which(x<3) which indices correspond to the TRUE values of x<3
x[x<3] # the x values when x<3 is TRUE -- same as x[which(x<3)]



# INTRO TO GRAPHING
# input a vector x with data
x <- c(2,4,4,6,6,5,5,7,3,7,3,8,9,7,9,6,4,3,4,4,6,2,2,1,2,4,6,6,8)
# input vector y of consecutive numbers
# 1 through 29
y <- c(1:29)
# make a histogram of x
hist(x)
# make a scatter plot of y vs x
plot(x,y)
# make a time plot of x, connecting dots
plot(x, type="b")
# make a histogram of a sampling of 20 from
# the binomial distribution B(12,.4)
y <- rbinom(20, 12, .4)
hist(y)
# make another binomial histogram, but using
a
# sample of 200
y <- rbinom(200, 12, .4)
hist(y)
# make a boxplot of x data
boxplot(x)
# make side by side boxplots of x and y data
boxplot(x,y)



#LOGICAL OPERATORS
x <- 1:5 # x is a row vector = [1,2,3,4,5]
test <- (x <4) # test is a row vector (logical <)
# = [TRUE TRUE TRUE FALSE FALSE]
```

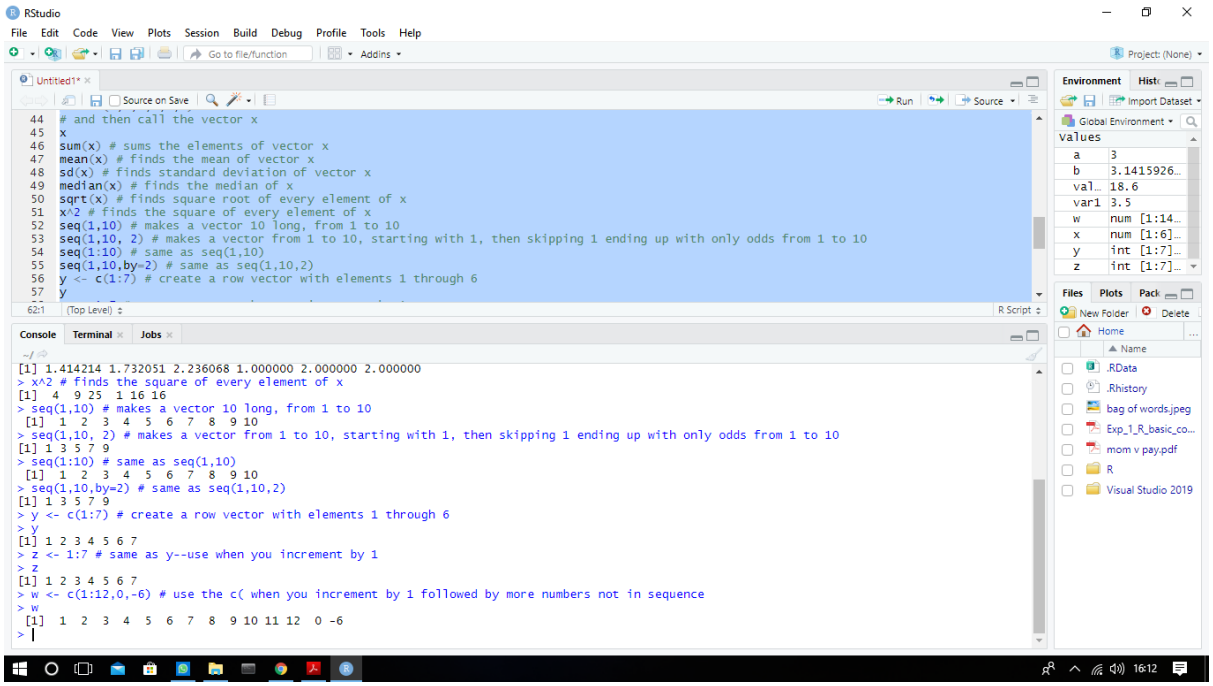
```
test <- (x==3) # test is a row vector (logical =)
# = [FALSE FALSE TRUE FALSE FALSE]
test <- (x>1 & x<4) # test is a row vector (logical AND)
# = [FALSE TRUE TRUE FALSE FALSE]
test <- (x>4 | x==2) # test is a row vector (logical OR)
# = [FALSE TRUE FALSE FALSE TRUE]
test<- (x != 4) # test is a row vector (logical NOT =)
# = [TRUE TRUE TRUE FALSE TRUE]
test <- (x %in% c(2,4)) # test is a row vector
# testing whether the entry is
# a 2 or a 4
# = [FALSE TRUE FALSE TRUE FALSE]



# THE NORMAL DISTRIBUTION
pnorm(1.43) # finds P(Z < 1.43)
pnorm(129,100,16) # finds P(X < 129) where x comes from
# normal with mean 100 and std dev 16
qnorm(.994) # finds the .994 quantile of the standard normal
qnorm(.95,100,16) # finds the .95 quantile of N(100,16) distribution


# THINGS TO KNOW
x <- runif(30,0,1) # creates a vector 30 long
x # of random values from U(0,1)
y <- c() # before you can use a vector in aloop
for (i in 1:10) { # you must "save space" for it by
  y[i] <- rbinom(1,3,.2) # typing the y <- c() command
}
y
s <- c(1:5) # for R you can use <- or =
s # for a variable assignment
t = c(2:6)
t
help(mean)
```

Screenshots :






RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
a <- 3 # This stores the value 3 into variable a
a
sqrt(a) # square root of a (which is 3 at this moment)
a^4 # a raised to the fourth power
log(a) # natural log of a
log10(a) # common log of a
exp(a) # e to the power a
tan(a) # tangent of a
b <- pi # pie, the number approx 3.14159
b
Console Terminal Jobs
> a <- 3 # This stores the value 3 into variable a
> a
[1] 3
> sqrt(a) # square root of a (which is 3 at this moment)
[1] 1.732051
> a^4 # a raised to the fourth power
[1] 81
> log(a) # natural log of a
[1] 1.098612
> log10(a) # common log of a
[1] 0.4771213
> exp(a) # e to the power a
[1] 20.08554
> tan(a) # tangent of a
[1] -0.1425465
> b <- pi # pie, the number approx 3.14159
> b
[1] 3.141593
Environment
Import Dataset
Global Environment
values
a 3
b 3.1415926...
val... 18.6
var1 3.5
x 5
Files Plots
New Folder Delete
Home
Name
.RData
.Rhistory
bag of words.jpeg
Exp_1_R_basic_co...
mom v pay.pdf
R
Visual Studio 2019
16:08


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
and then call the vector x
x
sum(x) # sums the elements of vector x
mean(x) # finds the mean of vector x
sd(x) # finds standard deviation of vector x
median(x) # finds the median of x
sqrt(x) # finds square root of every element of x
x^2 # finds the square of every element of x
seq(1,10) # makes a vector 10 long, from 1 to 10
seq(1,10, 2) # makes a vector from 1 to 10, starting with 1, then skipping 1 ending up with only odds from 1 to 10
seq(1:10) # same as seq(1,10)
seq(1,10,by=2) # same as seq(1,10,2)
y <- c(1:7) # create a row vector with elements 1 through 6
y
Console Terminal Jobs
[1] 1.414214 1.732051 2.236068 1.000000 2.000000 2.000000
> x^2 # finds the square of every element of x
[1] 4 9 25 1 16 16
> seq(1,10) # makes a vector 10 long, from 1 to 10
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1,10, 2) # makes a vector from 1 to 10, starting with 1, then skipping 1 ending up with only odds from 1 to 10
[1] 1 3 5 7 9
> seq(1:10) # same as seq(1,10)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1,10,by=2) # same as seq(1,10,2)
[1] 1 3 5 7 9
> y <- c(1:7) # create a row vector with elements 1 through 6
> y
[1] 1 2 3 4 5 6 7
> z <- 1:7 # same as y--use when you increment by 1
> z
[1] 1 2 3 4 5 6 7
> w <- c(1:12,0,-6) # use the c(when you increment by 1 followed by more numbers not in sequence
> w
[1] 1 2 3 4 5 6 7 8 9 10 11 12 0 -6
Environment
Import Dataset
Global Environment
values
a 3
b 3.1415926...
val... 18.6
var1 3.5
w num [1:14...
x num [1:6]...
y int [1:7]...
z int [1:7]...
Files Plots
New Folder Delete
Home
Name
.RData
.Rhistory
bag of words.jpeg
Exp_1_R_basic_co...
mom v pay.pdf
R
Visual Studio 2019
16:12

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
x[i] <- 5 # assign a value of 5 to the first entry; also can use x[i] = 5
x[c(1,4)] <- c(2,3) # assign values to the first and fourthentries
y
x[indices] <- y # assign to those indices indicated by the values
# of INDICES: if y is not long enough, values
# are recycled; if y is too long, just its initial
# values are used and a warning is issued
x
x < 3 # vector with length n of TRUE or FALSE # depending if x[i] < 3 which(x<3) which indices correspond to the TRUE values of x<3
x[x<3] # the x values when x<3 is TRUE -- same as x[which(x<3)]
```

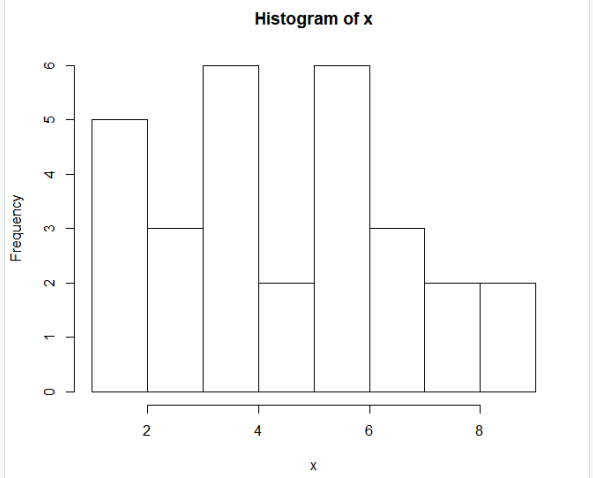
```
> 1<-3
> x[i] # the ith entry if 1<=i<=n, NA if i>n, all but the ith if -n<=i<=-1, an error if i < -n, and an empty vector if i=0
[1] 7
> x[c(2,3)] # the second and 3rd entries
[1] 0 7
> x[-c(2,3)] # all but the seccond and third entries
[1] -5 -6 14 27
> x[i] <- 5 # assign a value of 5 to the first entry; also can use x[i] = 5
> x[c(1,4)] <- c(2,3) # assign values to the first and fourthentries
> y
[1] 1 2 3 4 5 6 7
> x[indices] <- y # assign to those indices indicated by the values
warning message:
In x[indices] <- y :
  number of items to replace is not a multiple of replacement length
> # of INDICES: if y is not long enough, values
> # are recycled; if y is too long, just its initial
> # values are used and a warning is issued
> x
[1]  1  0  5  3 14 27  2
> x < 3 # vector with length n of TRUE or FALSE # depending if x[i] < 3 which(x<3) which indices correspond to the TRUE values of x<3
[1]  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE
```

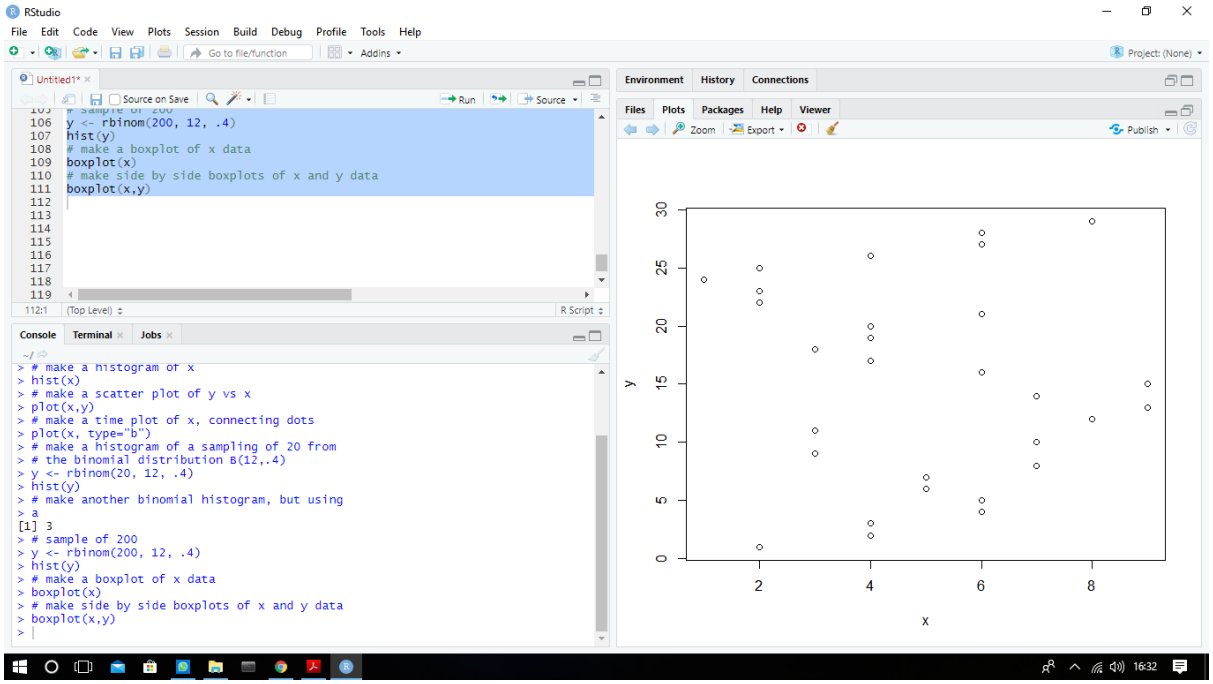

RStudio

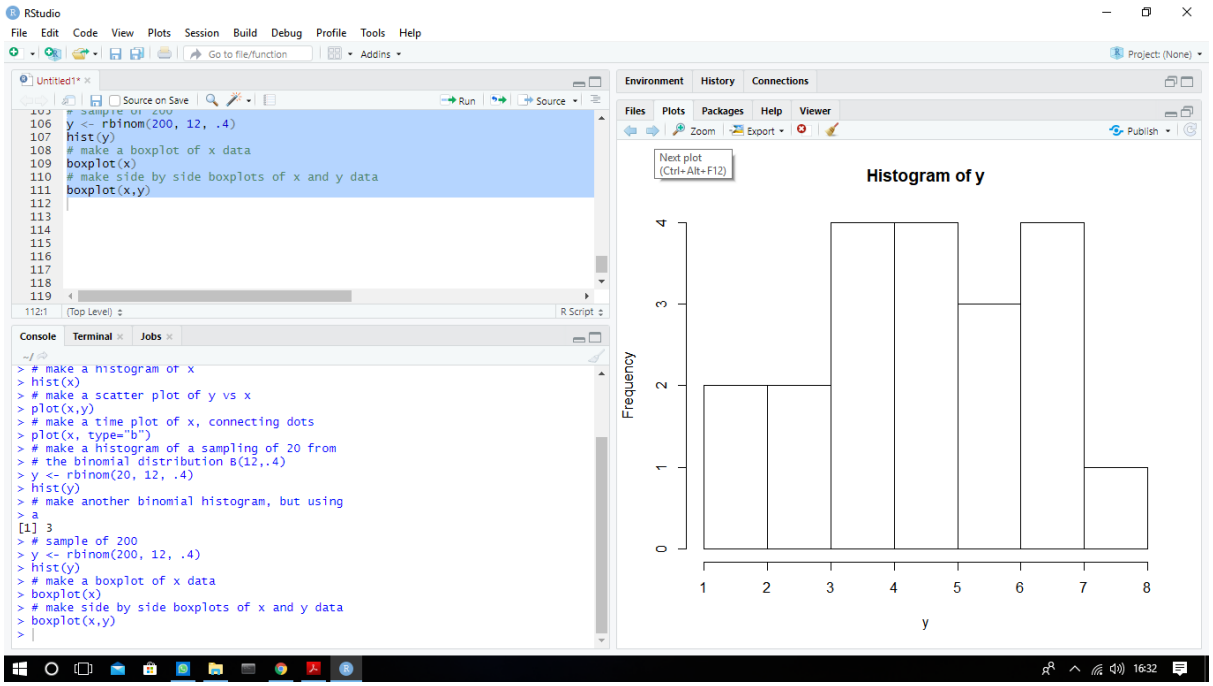
File Edit Code View Plots Session Build Debug Profile Tools Help

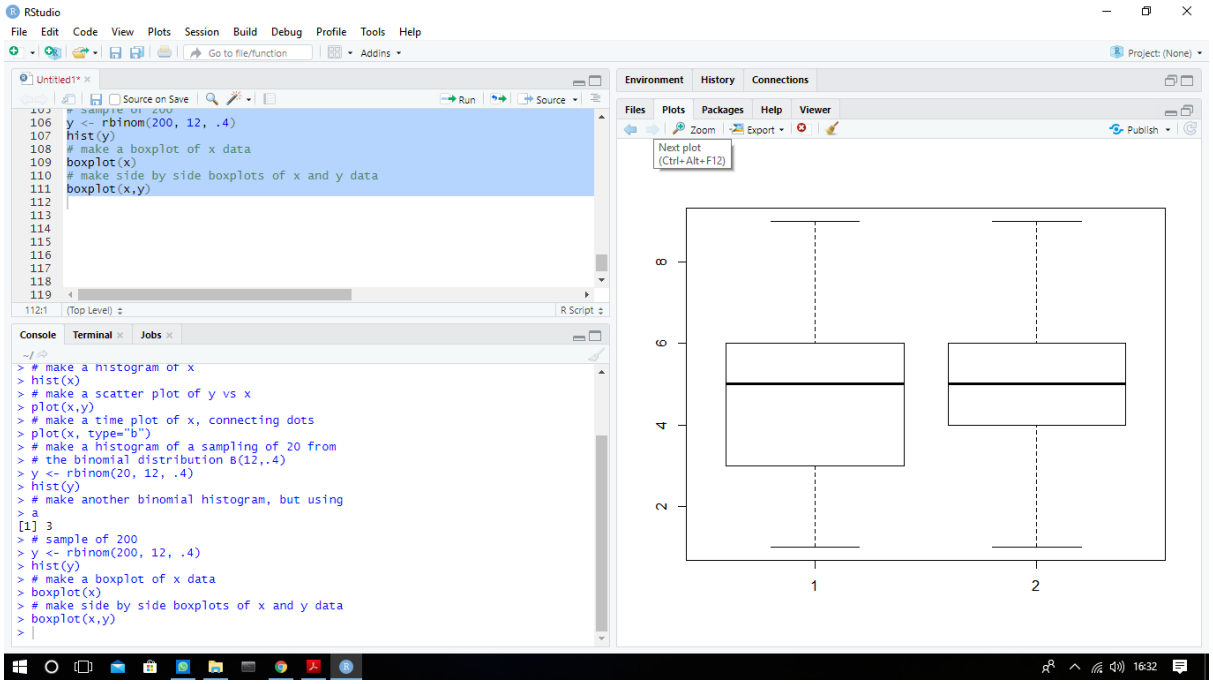
```
y <- rbinom(200, 12, .4)
hist(y)
# make a boxplot of x data
boxplot(x)
# make side by side boxplots of x and y data
boxplot(x,y)
```

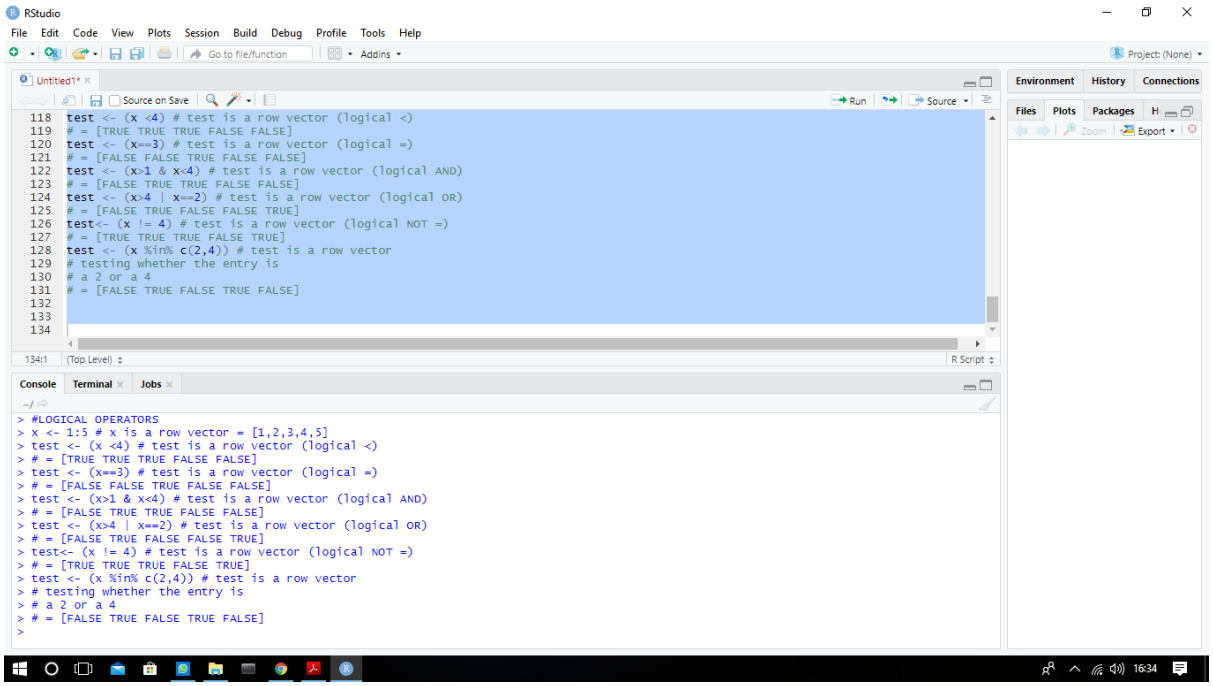
```
> # make a histogram of x
> hist(x)
> # make a scatter plot of y vs x
> plot(x,y)
> # make a time plot of x, connecting dots
> plot(x, type="b")
> # make a histogram of a sampling of 20 from
> # the binomial distribution B(12,.4)
> y <- rbinom(20, 12, .4)
> hist(y)
> # make another binomial histogram, but using
> a
[1] 3
> # sample of 200
> y <- rbinom(200, 12, .4)
> hist(y)
> # make a boxplot of x data
> boxplot(x)
> # make side by side boxplots of x and y data
> boxplot(x,y)
>
```

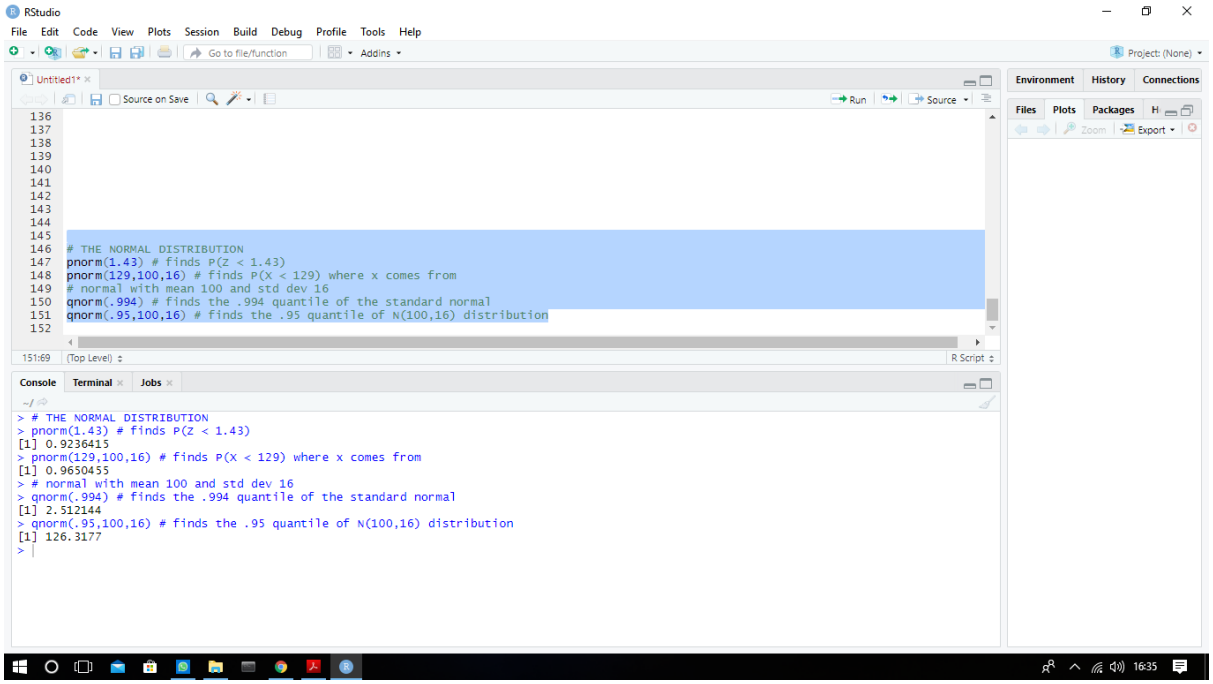


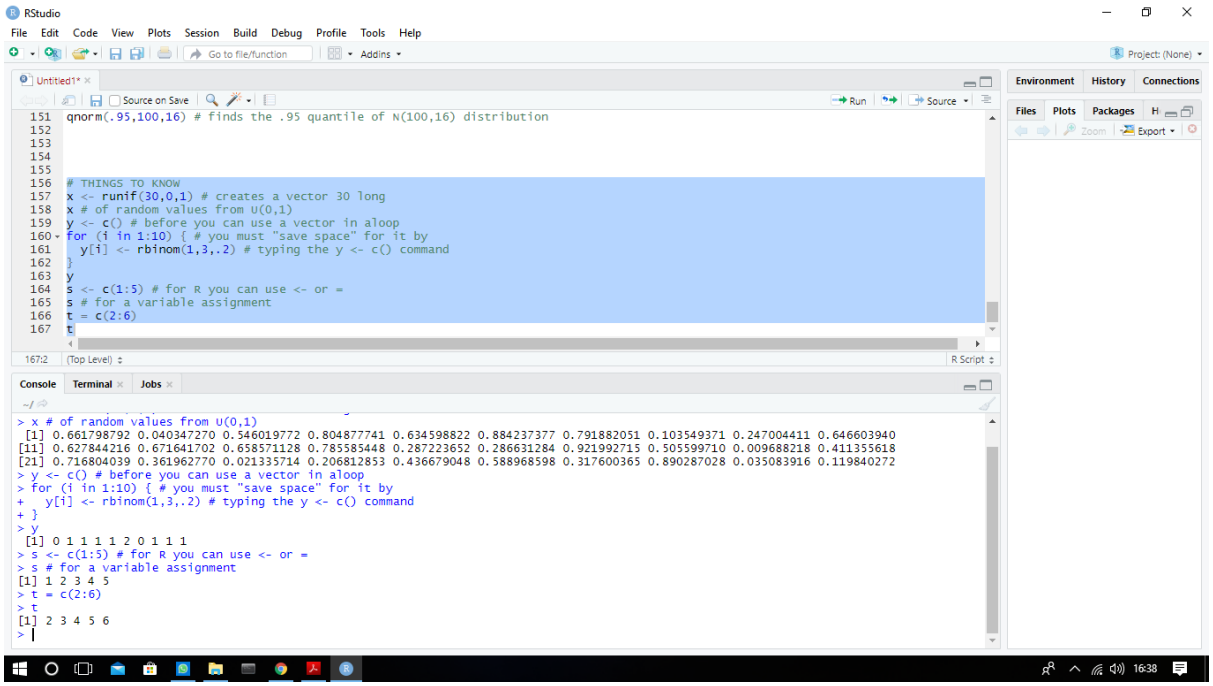

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
106 y <- rbinom(200, 12, .4)
107 hist(y)
108 # make a boxplot of x data
109 boxplot(x)
110 # make side by side boxplots of x and y data
111 boxplot(x,y)
Console Terminal Jobs
> # make a histogram of x
> hist(x)
> # make a scatter plot of y vs x
> plot(x,y)
> # make a time plot of x, connecting dots
> plot(x, type="b")
> # make a histogram of a sampling of 20 from
> # the binomial distribution B(12,.4)
> y <- rbinom(20, 12, .4)
> hist(y)
> # make another binomial histogram, but using
> a
[1] 3
> # sample of 200
> y <- rbinom(200, 12, .4)
> hist(y)
> # make a boxplot of x data
> boxplot(x)
> # make side by side boxplots of x and y data
> boxplot(x,y)
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
y
x

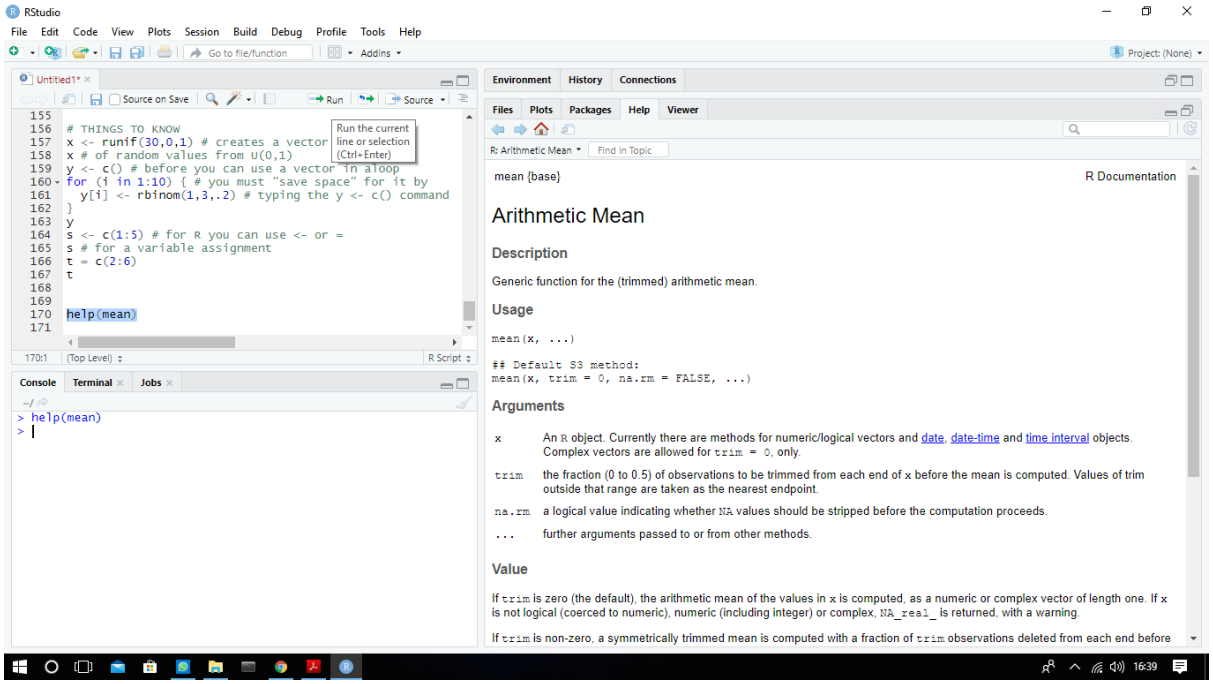

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
106 y <- rbinom(200, 12, .4)
107 hist(y)
108 # make a boxplot of x data
109 boxplot(x)
110 # make side by side boxplots of x and y data
111 boxplot(x,y)
Console Terminal Jobs
> # make a histogram of x
> hist(x)
> # make a scatter plot of y vs x
> plot(x,y)
> # make a time plot of x, connecting dots
> plot(x, type="b")
> # make a histogram of a sampling of 20 from
> # the binomial distribution B(12,.4)
> y <- rbinom(20, 12, .4)
> hist(y)
> # make another binomial histogram, but using
> a
[1] 3
> # sample of 200
> y <- rbinom(200, 12, .4)
> hist(y)
> # make a boxplot of x data
> boxplot(x)
> # make side by side boxplots of x and y data
> boxplot(x,y)
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
Next plot (Ctrl+Alt+F12)
Histogram of y
Frequency
y


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
y <- rbinom(200, 12, .4)
hist(y)
# make a boxplot of x data
boxplot(x)
# make side by side boxplots of x and y data
boxplot(x,y)
Console
Terminal
Jobs
> # make a histogram of x
> hist(x)
> # make a scatter plot of y vs x
> plot(x,y)
> # make a time plot of x, connecting dots
> plot(x, type="b")
> # make a histogram of a sampling of 20 from
> # the binomial distribution B(12,.4)
> y <- rbinom(20, 12, .4)
> hist(y)
> # make another binomial histogram, but using
> a
[1] 3
> # sample of 200
> y <- rbinom(200, 12, .4)
> hist(y)
> # make a boxplot of x data
> boxplot(x)
> # make side by side boxplots of x and y data
> boxplot(x,y)
Environment
History
Connections
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Next plot
(Ctrl+Alt+F12)
8
6
4
2
1
2


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
test <- (x <4) # test is a row vector (logical <)
# = [TRUE TRUE TRUE FALSE FALSE]
test <- (x==3) # test is a row vector (logical =)
# = [FALSE FALSE TRUE FALSE FALSE]
test <- (x>1 & x<4) # test is a row vector (logical AND)
# = [FALSE TRUE TRUE FALSE FALSE]
test <- (x>4 | x==2) # test is a row vector (logical OR)
# = [FALSE TRUE FALSE FALSE TRUE]
test<- (x != 4) # test is a row vector (logical NOT =)
# = [TRUE TRUE TRUE FALSE TRUE]
test <- (x %in% c(2,4)) # test is a row vector
# testing whether the entry is
# a 2 or a 4
# = [FALSE TRUE FALSE TRUE FALSE]
Console
Terminal
Jobs
> #LOGICAL OPERATORS
> x <- 1:5 # x is a row vector = [1,2,3,4,5]
> test <- (x <4) # test is a row vector (logical <)
> # = [TRUE TRUE TRUE FALSE FALSE]
> test <- (x==3) # test is a row vector (logical =)
> # = [FALSE FALSE TRUE FALSE FALSE]
> test <- (x>1 & x<4) # test is a row vector (logical AND)
> # = [FALSE TRUE TRUE FALSE FALSE]
> test <- (x>4 | x==2) # test is a row vector (logical OR)
> # = [FALSE TRUE FALSE FALSE TRUE]
> test<- (x != 4) # test is a row vector (logical NOT =)
> # = [TRUE TRUE TRUE FALSE TRUE]
> test <- (x %in% c(2,4)) # test is a row vector
> # testing whether the entry is
> # a 2 or a 4
> # = [FALSE TRUE FALSE TRUE FALSE]
Environment
History
Connections
Files
Plots
Packages
Export


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
Environment History Connections
Files Plots Packages H
Export
# THE NORMAL DISTRIBUTION
pnorm(1.43) # finds P(Z < 1.43)
pnorm(129,100,16) # finds P(X < 129) where x comes from
normal with mean 100 and std dev 16
qnorm(.994) # finds the .994 quantile of the standard normal
qnorm(.95,100,16) # finds the .95 quantile of N(100,16) distribution
151:69 (Top Level) R Script
Console Terminal Jobs
> # THE NORMAL DISTRIBUTION
> pnorm(1.43) # finds P(Z < 1.43)
[1] 0.9236415
> pnorm(129,100,16) # finds P(X < 129) where x comes from
[1] 0.9650455
> # normal with mean 100 and std dev 16
> qnorm(.994) # finds the .994 quantile of the standard normal
[1] 2.512144
> qnorm(.95,100,16) # finds the .95 quantile of N(100,16) distribution
[1] 126.3177
16:35


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
Environment History Connections
Files Plots Packages H
Export
qnorm(.95,100,16) # finds the .95 quantile of N(100,16) distribution
# THINGS TO KNOW
x <- runif(30,0,1) # creates a vector 30 long
x # of random values from U(0,1)
y <- c() # before you can use a vector in aloop
for (i in 1:10) { # you must "save space" for it by
y[i] <- rbinom(1,3,.2) # typing the y <- c() command
}
y
s <- c(1:5) # for R you can use <- or =
s # for a variable assignment
t = c(2:6)
t
167:2 (Top Level) R Script
Console Terminal Jobs
> x # of random values from U(0,1)
[1] 0.661798792 0.040347270 0.546019772 0.804877741 0.634598822 0.884237377 0.791882051 0.103549371 0.247004411 0.646603940
[11] 0.627844216 0.671641702 0.658571128 0.785585448 0.287223652 0.286631284 0.921992715 0.505599710 0.009688218 0.411355618
[21] 0.716804039 0.361962770 0.021335714 0.206812853 0.436679048 0.588968598 0.317600365 0.890287028 0.035083916 0.119840272
> y <- c() # before you can use a vector in aloop
> for (i in 1:10) { # you must "save space" for it by
+ y[i] <- rbinom(1,3,.2) # typing the y <- c() command
+ }
> y
[1] 0 1 1 1 1 2 0 1 1 1
> s <- c(1:5) # for R you can use <- or =
> s # for a variable assignment
[1] 1 2 3 4 5
> t = c(2:6)
> t
[1] 2 3 4 5 6
16:38



Conclusion : Thus we have studied the basic functionality of R.

Experiment No. 3

Aim : Basic Data types and data structures in R - vector ,matrices,list and dataframes.

Theory :

Vectors are the most basic R data objects and there are six types of atomic vectors. They are logical, integer, double, complex, character and raw. Elements of a Vector are accessed using indexing. The [ ] brackets are used for indexing. Indexing starts with position 1. Giving a negative value in the index drops that element from the result. TRUE, FALSE or 0 and 1 can also be used for indexing.

Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout. They contain elements of the same atomic types. Though we can create a matrix containing only characters or only logical values, they are not of much use. We use matrices containing numeric elements to be used in mathematical calculations.

A Matrix is created using the matrix() function. The basic syntax for creating a matrix in R is −

matrix(data, nrow, ncol, byrow, dimnames)

Following is the description of the parameters used −

- data is the input vector which becomes the data elements of the matrix.
- nrow is the number of rows to be created.
- ncol is the number of columns to be created.
- byrow is a logical clue. If TRUE then the input vector elements are arranged by row.
- dimname is the names assigned to the rows and columns.

Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it. A list can also contain a matrix or a function as its elements. List is created using list() function.

R list is the object which contains elements of different types – like *strings, numbers, vectors and another list* inside it. R list can also contain a matrix or a function as its elements. The list is created using the *list()* function in R. In other words, a list is a generic vector containing other objects.

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column. Following are the characteristics of a data frame.

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factor or character type.

- Each column should contain the same number of data items.

Output :

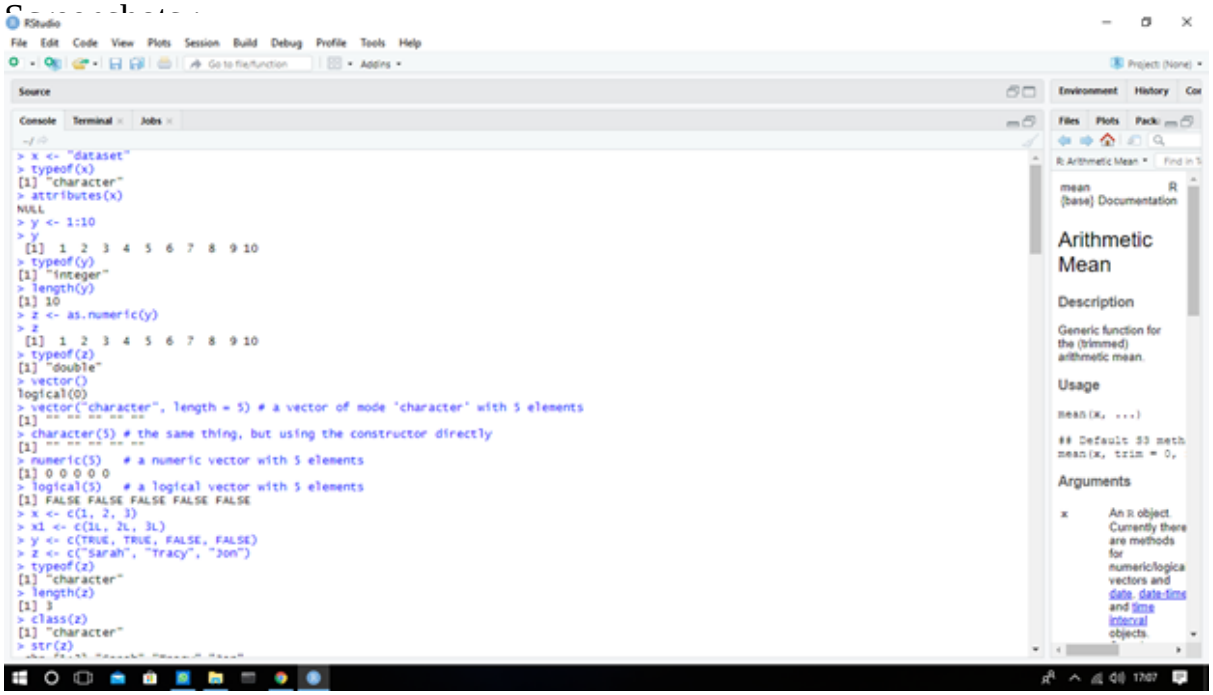
Code :

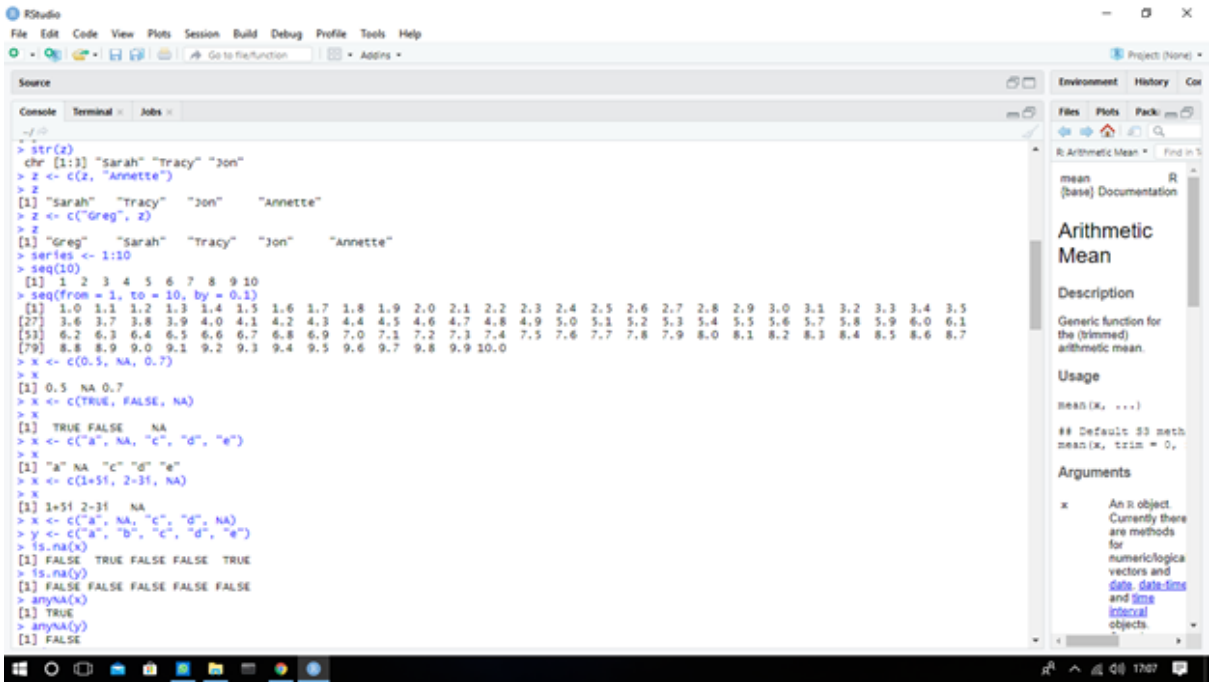
```
x <- "dataset"
typeof(x)
attributes(x)
y <- 1:10
y
typeof(y)
length(y)
z <- as.numeric(y)
z
typeof(z)
vector()
vector("character", length = 5) # a vector of mode 'character' with 5 elements
character(5) # the same thing, but using the constructor directly
numeric(5)   # a numeric vector with 5 elements
logical(5)   # a logical vector with 5 elements
x <- c(1, 2, 3)
x1 <- c(1L, 2L, 3L)
y <- c(TRUE, TRUE, FALSE, FALSE)
z <- c("Sarah", "Tracy", "Jon")
typeof(z)
length(z)
class(z)
str(z)
z <- c(z, "Annette")
z
z <- c("Greg", z)
z
series <- 1:10
seq(10)
seq(from = 1, to = 10, by = 0.1)
x <- c(0.5, NA, 0.7)
x
x <- c(TRUE, FALSE, NA)
x
x <- c("a", NA, "c", "d", "e")
x
x <- c(1+5i, 2-3i, NA)
x
```

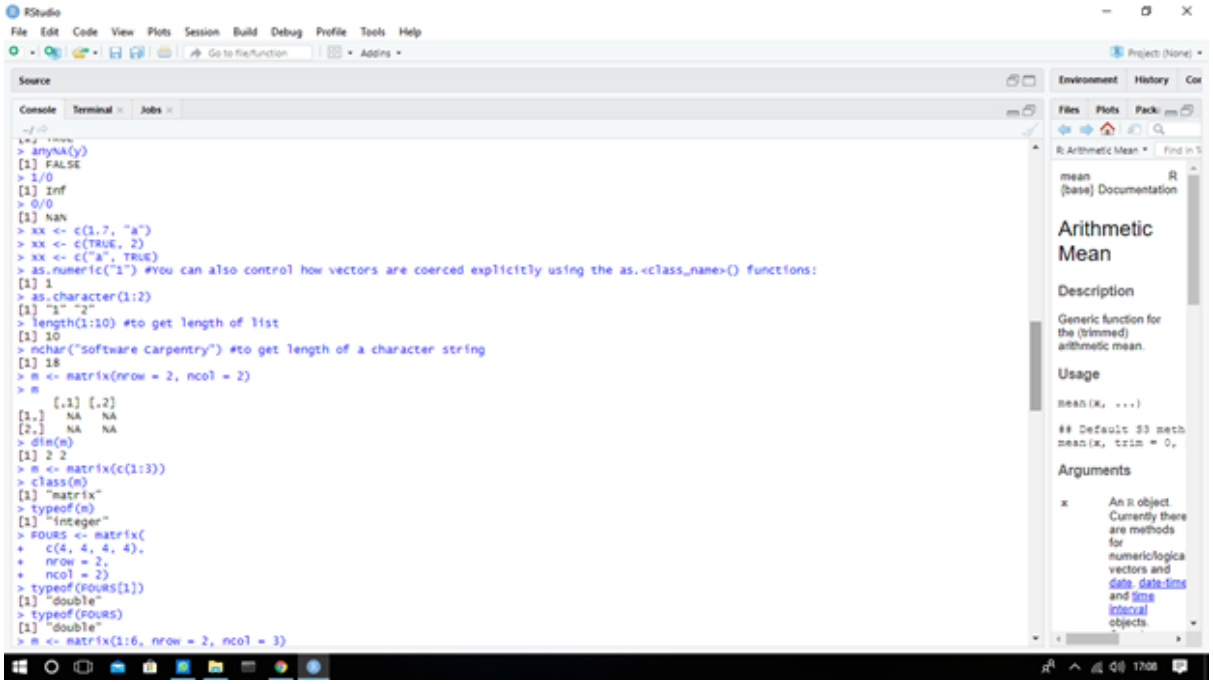
```
x <- c("a", NA, "c", "d", NA)
y <- c("a", "b", "c", "d", "e")
is.na(x)
is.na(y)
anyNA(x)
anyNA(y)
1/0
0/0
xx <- c(1.7, "a")
xx <- c(TRUE, 2)
xx <- c("a", TRUE)
as.numeric("1") #You can also control how vectors are coerced explicitly using the as.<class_name>() functions:
as.character(1:2)
length(1:10) #to get length of list
nchar("Software Carpentry") #to get length of a character string
m <- matrix(nrow = 2, ncol = 2)
m
dim(m)
m <- matrix(c(1:3))
class(m)
typeof(m)
FOURS <- matrix(
  c(4, 4, 4, 4),
  nrow = 2,
  ncol = 2)
typeof(FOURS[1])
typeof(FOURS)
m <- matrix(1:6, nrow = 2, ncol = 3)
m      <- 1:10 #another way to make a matrix
dim(m) <- c(2, 5)
x <- 1:3
y <- 10:12
cbind(x, y)
rbind(x, y)
mdat <- matrix(c(1, 2, 3, 11, 12, 13),
               nrow = 2,
               ncol = 3,
               byrow = TRUE)
mdat
mdat[2, 3]
x <- list(1, "a", TRUE, 1+4i)
x
x <- vector("list", length = 5) # empty list
```

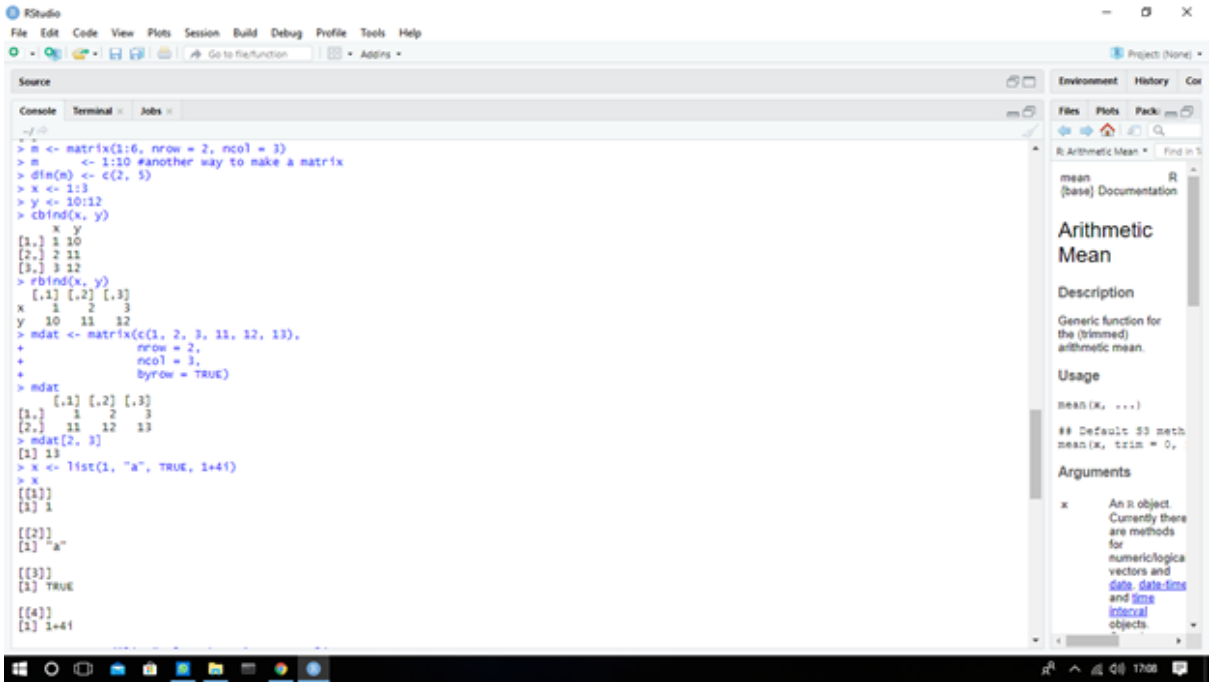
```
length(x)
x[[1]]
#Vectors can be coerced to lists as follows:

x <- 1:10
x <- as.list(x)
length(x)
xlist <- list(a = "Karthik Ram", b = 1:10, data = head(iris)) #Elements of a list can be named (i.e. lists can have the names attribute)
xlist
names(xlist)

dat <- data.frame(id = letters[1:10], x = 1:10, y = 11:20) #To create data frames by hand
dat
is.list(dat) #data frame is a special list
class(dat)
dat[1, 3]
#As data frames are also lists, it is possible to refer to columns (which are elements of such list) using the list notation, i.e. either double square brackets or a $
dat[["y"]]
dat$y
```

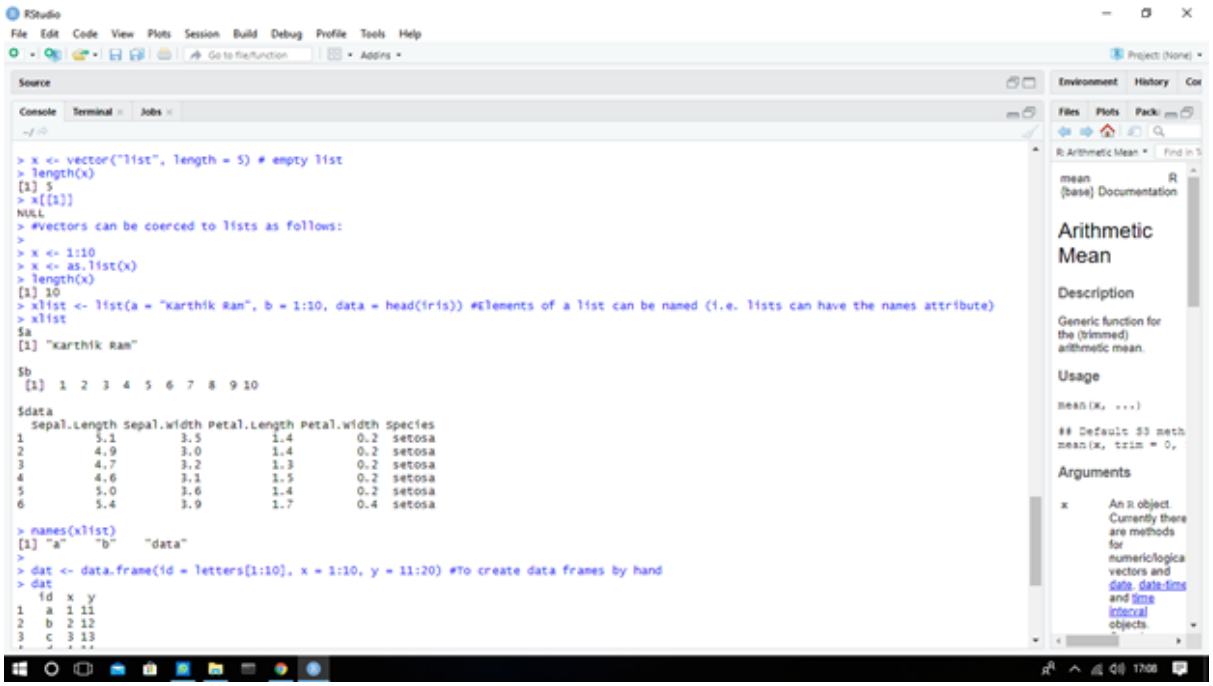
Screenshots:

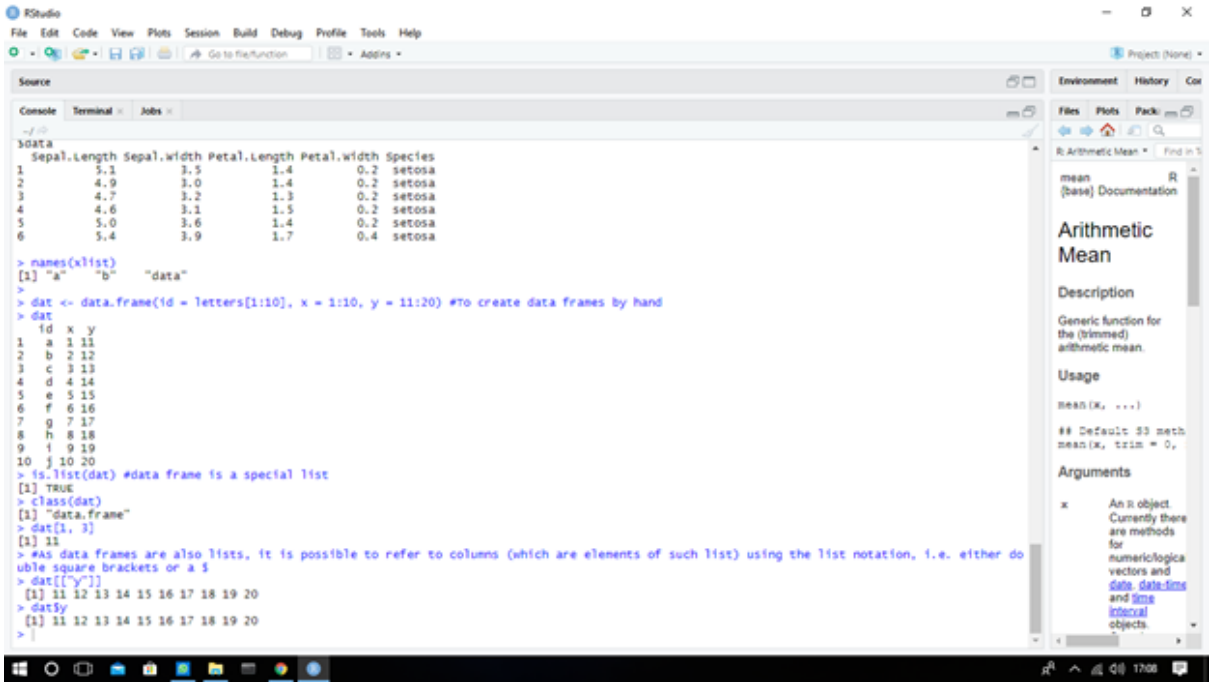





RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
> anyNA(y)
[1] FALSE
> 1/0
[1] Inf
> 0/0
[1] NaN
> xx <- c(1.7, "a")
> xx <- c(TRUE, 2)
> xx <- c("a", TRUE)
> as.numeric("1") #You can also control how vectors are coerced explicitly using the as.<class_name>() functions:
[1] 1
> as.character(1:2)
[1] "1" "2"
> length(1:10) #to get length of list
[1] 10
> nchar("Software Carpentry") #to get length of a character string
[1] 18
> m <- matrix(nrow = 2, ncol = 2)
> m
[,1] [,2]
[1,] NA NA
[2,] NA NA
> dim(m)
[1] 2 2
> m <- matrix(c(1:3))
> class(m)
[1] "matrix"
> typeof(m)
[1] "integer"
> FOURS <- matrix(
+ c(4, 4, 4, 4),
+ nrow = 2,
+ ncol = 2)
> typeof(FOURS[1])
[1] "double"
> typeof(FOURS)
[1] "double"
> m <- matrix(1:6, nrow = 2, ncol = 3)
Environment History
Files Plots Packages
R: Arithmetic Mean
mean R
{base} Documentation
Arithmetic Mean
Description
Generic function for the (trimmed) arithmetic mean.
Usage
mean(x, ...)
Default S3 meth
mean(x, trim = 0,
Arguments
x An R object. Currently there are methods for numeric/logica vectors and date, date-time and time interval objects.


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m <- 1:10 #another way to make a matrix
> dim(m) <- c(2, 5)
> x <- 1:3
> y <- 10:12
> cbind(x, y)
x y
[1,] 1 10
[2,] 2 11
[3,] 3 12
> rbind(x, y)
[,1] [,2] [,3]
x 1 2 3
y 10 11 12
> mdat <- matrix(c(1, 2, 3, 11, 12, 13),
+ nrow = 2,
+ ncol = 3,
+ byrow = TRUE)
> mdat
[,1] [,2] [,3]
[1,] 1 2 3
[2,] 11 12 13
> mdat[2, 3]
[1] 13
> x <- list(1, "a", TRUE, 1+4i)
> x
[[1]]
[1] 1
[[2]]
[1] "a"
[[3]]
[1] TRUE
[[4]]
[1] 1+4i
Environment History
Files Plots Packages
R: Arithmetic Mean
mean R
{base} Documentation
Arithmetic Mean
Description
Generic function for the (trimmed) arithmetic mean.
Usage
mean(x, ...)
Default S3 meth
mean(x, trim = 0,
Arguments
x An R object. Currently there are methods for numeric/logica vectors and date, date-time and time interval objects.





Conclusion : Thus, we have studied basic Data types and data structures in R - vector ,matrices,list and dataframes.

Experiment No. 4

Aim : Programming constructs in R - loops ,conditional executions

Theory :

A loop statement allows us to execute a statement or group of statements multiple times and the following is the general form of a loop statement in most of the programming languages .

1. Repeat Loop : The Repeat loop executes the same code again and again until a stop condition is met.
2. While Loop : The Repeat loop executes the same code again and again until a stop condition is met.
3. For Loop : A For loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

A conditional expression or a conditional statement is a programming construct where a decision is made to execute some code based on a boolean (true or false) condition. A more commonly used term for conditional expression in programming is an 'if-else' condition. In plain English, this is stated as 'if this test is true then do this operation; otherwise do this different operation'.

Output :

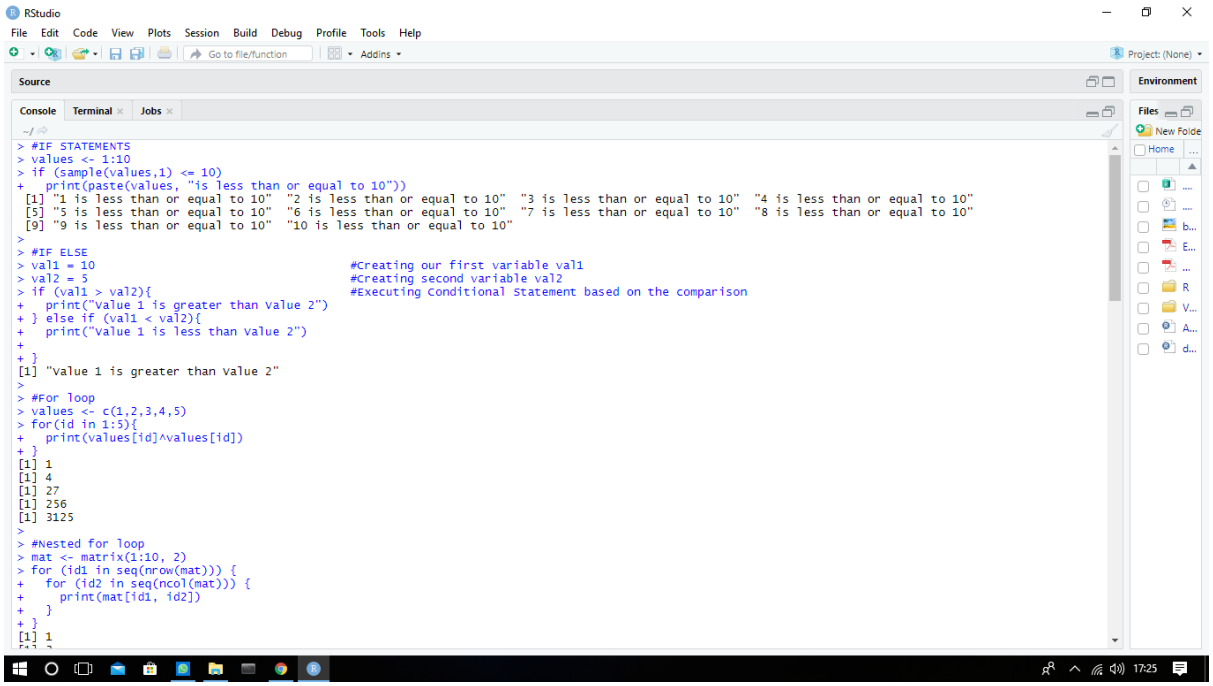
Code :

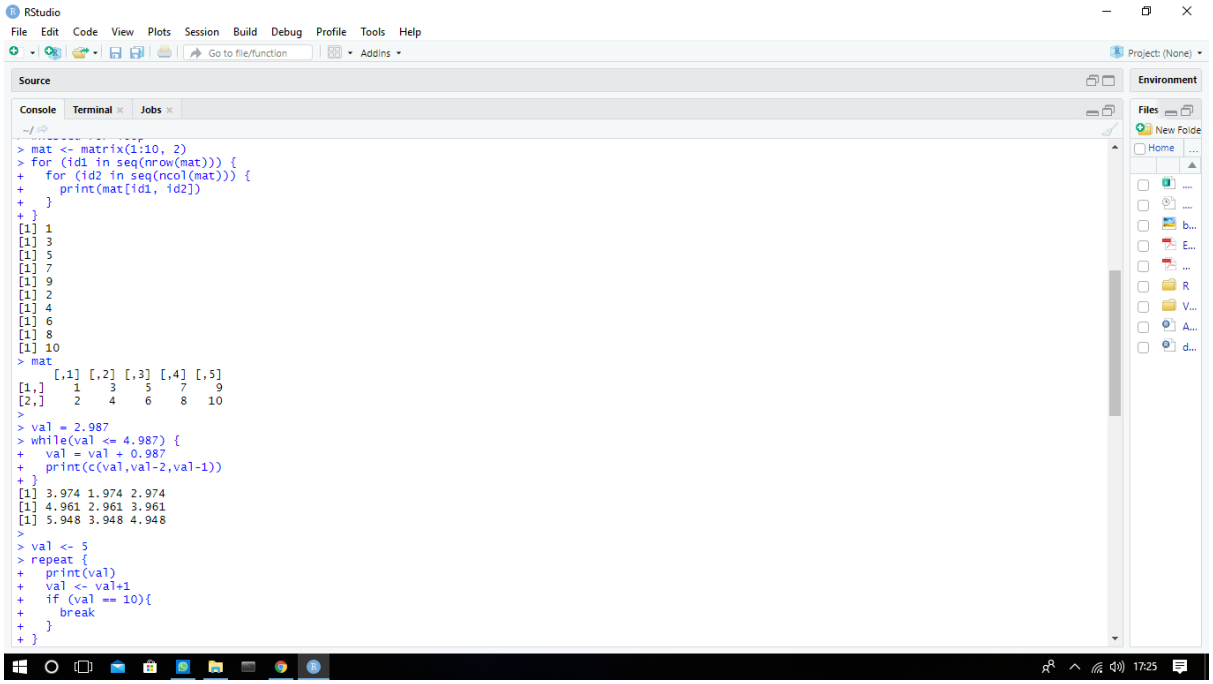
```
#IF STATEMENTS
values <- 1:10
if (sample(values,1) <= 10)
  print(paste(values, "is less than or equal to 10"))

#if else
val1 = 10                          #Creating our first variable val1
val2 = 5                           #Creating second variable val2
if (val1 > val2){                    #Executing Conditional Statement based on the comparison
  print("Value 1 is greater than Value 2")
} else if (val1 < val2){
  print("Value 1 is less than Value 2")

}

#For loop
values <- c(1,2,3,4,5)
for(id in 1:5){
```

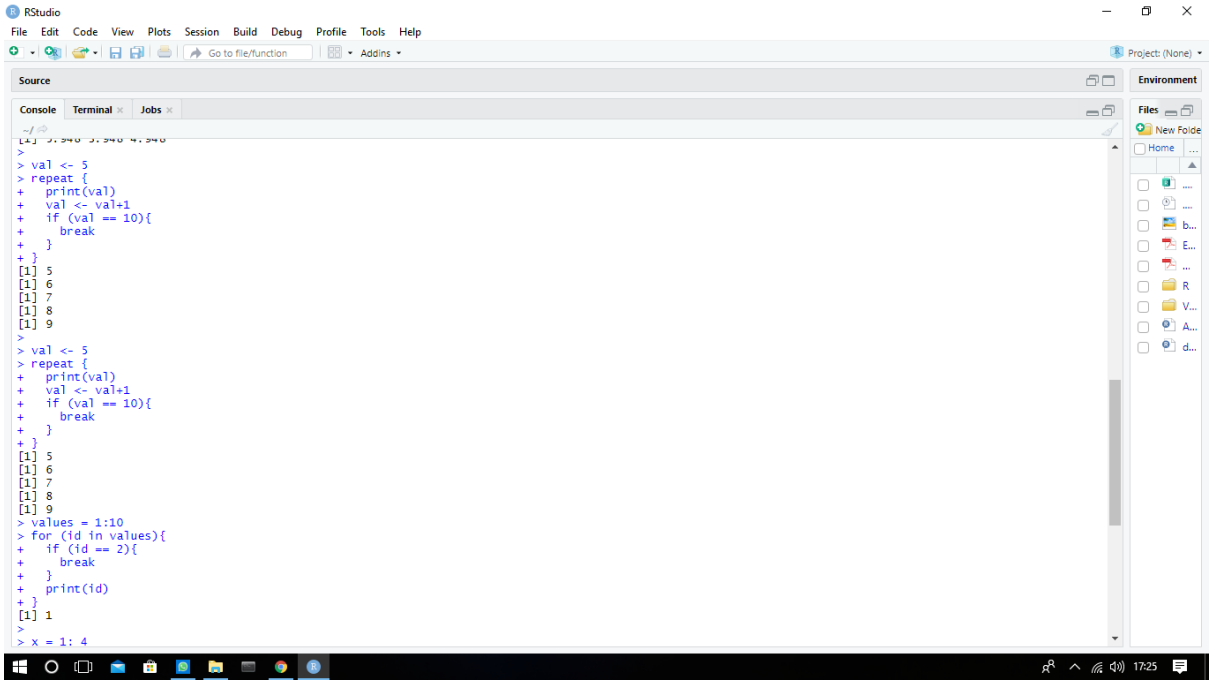
```
  print(values[id]^values[id])
}

#Nested for loop
mat <- matrix(1:10, 2)
for (id1 in seq(nrow(mat))) {
  for (id2 in seq(ncol(mat))) {
    print(mat[id1, id2])
  }
}
mat

val = 2.987
while(val <= 4.987) {
  val = val + 0.987
  print(c(val,val-2,val-1))
}

val <- 5
repeat {
  print(val)
  val <- val+1
  if (val == 10){
    break
  }
}

val <- 5
repeat {
  print(val)
  val <- val+1
  if (val == 10){
    break
  }
}
values = 1:10
for (id in values){
  if (id == 2){
    break
  }
  print(id)
}
x = 1: 4
for (i in x) {
```

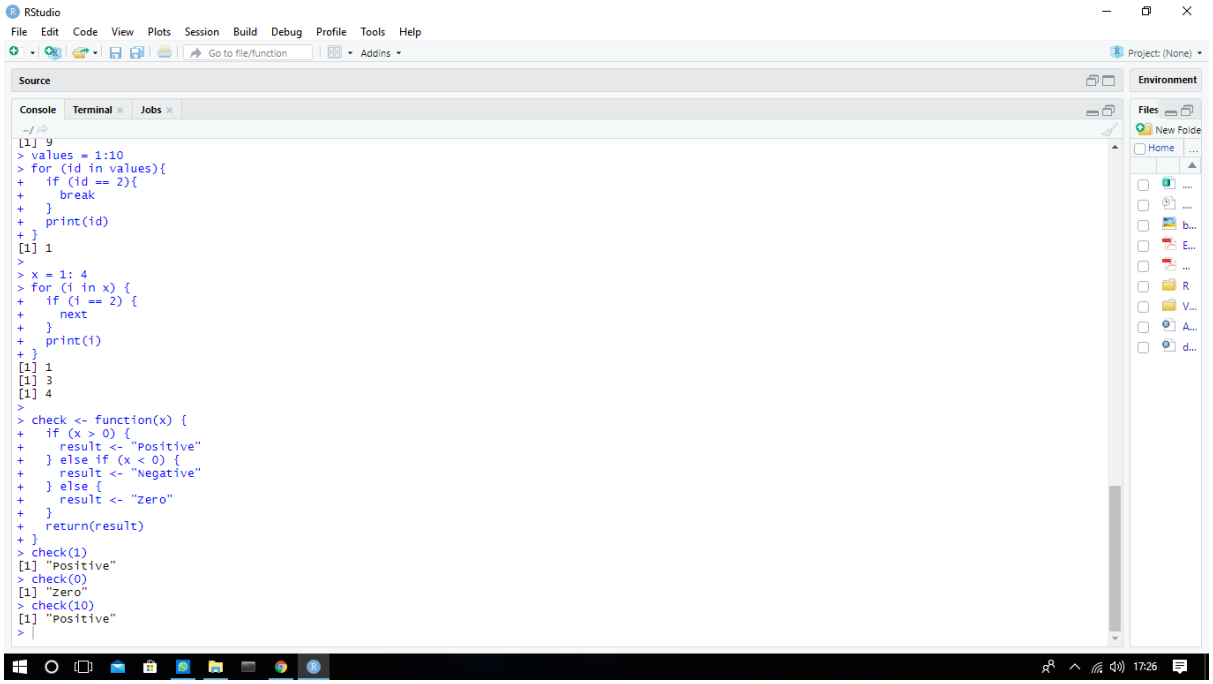
```
  if (i == 2) {
    next
  }
  print(i)
}

check <- function(x) {
  if (x > 0) {
    result <- "Positive"
  } else if (x < 0) {
    result <- "Negative"
  } else {
    result <- "Zero"
  }
  return(result)
}
check(1)
check(0)
check(10)
```

Screenshots :




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
Environment
Files
> mat <- matrix(1:10, 2)
> for (id1 in seq(nrow(mat))) {
+ for (id2 in seq(ncol(mat))) {
+ print(mat[id1, id2])
+ }
+ }
[1] 1
[1] 3
[1] 5
[1] 7
[1] 9
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
> mat
[,1] [,2] [,3] [,4] [,5]
[1,] 1 3 5 7 9
[2,] 2 4 6 8 10
>
> val = 2.987
> while(val <= 4.987) {
+ val = val + 0.987
+ print(c(val,val-2,val-1))
+ }
[1] 3.974 1.974 2.974
[1] 4.961 2.961 3.961
[1] 5.948 3.948 4.948
>
> val <- 5
> repeat {
+ print(val)
+ val <- val+1
+ if (val == 10){
+ break
+ }
+ }


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
Environment
Files
>
> val <- 5
> repeat {
+ print(val)
+ val <- val+1
+ if (val == 10){
+ break
+ }
+ }
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
>
> val <- 5
> repeat {
+ print(val)
+ val <- val+1
+ if (val == 10){
+ break
+ }
+ }
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
> values = 1:10
> for (id in values){
+ if (id == 2){
+ break
+ }
+ print(id)
+ }
[1] 1
>
> x = 1; 4



Conclusion : Thus we have studied the programming constructs in R.

Experiment No. 5

Aim : Tables with labels in R - multiple response variables excel function translations

Theory:

expss computes and displays tables with support for 'SPSS'-style labels, multiple / nested banners, weights, multiple-response variables and significance testing. There are facilities for nice output of tables in knitr', R notebooks, 'Shiny' and 'Jupyter' notebooks. Proper methods for labelled variables add value labels support to base R functions and to some functions from other packages. Additionally, the package offers useful functions for data processing in marketing research / social surveys - popular data transformation functions from 'SPSS' Statistics ('RECODE', 'COUNT', 'COMPUTE', 'DO IF', etc.) and 'Excel' ('COUNTIF', 'VLOOKUP', etc.). Package is intended to help people to move data processing from 'Excel'/'SPSS' to R. See examples below. You can get help about any function by typing ?function_name in the R console.

Installation:

expss is on CRAN, so for installation, you can print in the console install.packages("expss").

Output:

Code:

```
install.packages("expss")
library(expss)
data(product_test)

w = product_test # shorter name to save some keystrokes

# here we recode variables from first/second tested product to separate variables for each product according to their cells
# 'h' variables - VSX123 sample, 'p' variables - 'SDF456' sample
# also we recode preferences from first/second product to true names
# for first cell there are no changes, for the second cell we should exchange 1 and 2.
w = w %>%
  do_if(cell == 1, {
    recode(a1_1 %to% a1_6, other ~ copy) %into% (h1_1 %to% h1_6)
    recode(b1_1 %to% b1_6, other ~ copy) %into% (p1_1 %to% p1_6)
    recode(a22, other ~ copy) %into% h22
    recode(b22, other ~ copy) %into% p22
    c1r = c1
  }) %>%
  do_if(cell == 2, {
    recode(a1_1 %to% a1_6, other ~ copy) %into% (p1_1 %to% p1_6)
```

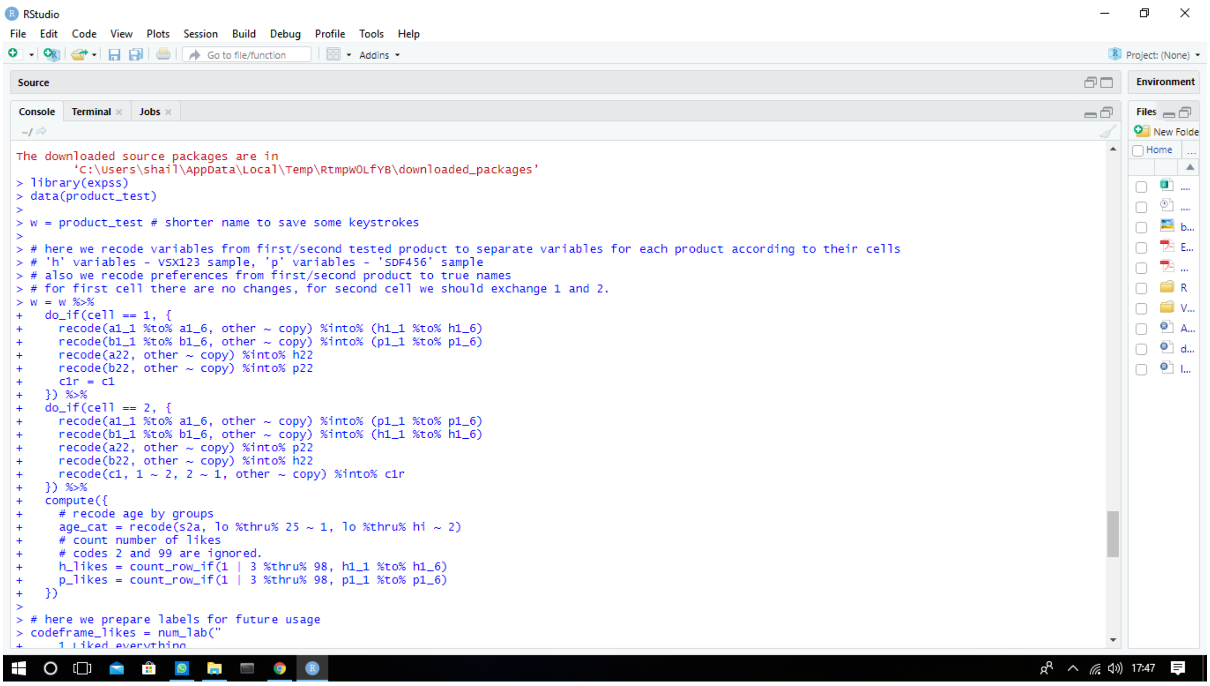
```
    recode(b1_1 %to% b1_6, other ~ copy) %into% (h1_1 %to% h1_6)
    recode(a22, other ~ copy) %into% p22
    recode(b22, other ~ copy) %into% h22
    recode(c1, 1 ~ 2, 2 ~ 1, other ~ copy) %into% c1r
  }) %>%
  compute({
    # recode age by groups
    age_cat = recode(s2a, lo %thru% 25 ~ 1, lo %thru% hi ~ 2)
    # count number of likes
    # codes 2 and 99 are ignored.
    h_likes = count_row_if(1 | 3 %thru% 98, h1_1 %to% h1_6)
    p_likes = count_row_if(1 | 3 %thru% 98, p1_1 %to% p1_6)
  })

# here we prepare labels for future usage
codeframe_likes = num_lab("
    1 Liked everything
    2 Disliked everything
    3 Chocolate
    4 Appearance
    5 Taste
    6 Stuffing
    7 Nuts
    8 Consistency
    98 Other
    99 Hard to answer
")

overall_liking_scale = num_lab("
    1 Extremely poor
    2 Very poor
    3 Quite poor
    4 Neither good, nor poor
    5 Quite good
    6 Very good
    7 Excellent
")

w = apply_labels(w,
                 c1r = "Preferences",
                 c1r = num_lab("
        1 VSX123
        2 SDF456
        3 Hard to say
    "),

                 age_cat = "Age",
                 age_cat = c("18 - 25" = 1, "26 - 35" = 2),

                 h1_1 = "Likes. VSX123",
```

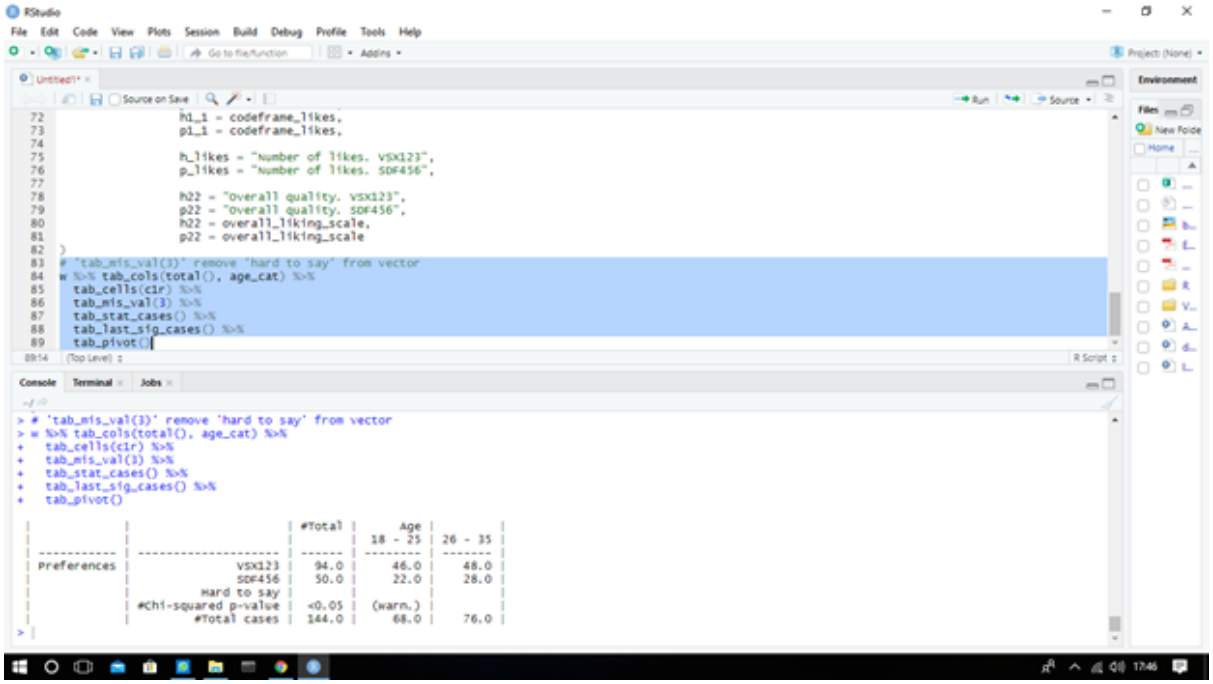
```
            p1_1 = "Likes. SDF456",
            h1_1 = codeframe_likes,
            p1_1 = codeframe_likes,

            h_likes = "Number of likes. VSX123",
            p_likes = "Number of likes. SDF456",

            h22 = "Overall quality. VSX123",
            p22 = "Overall quality. SDF456",
            h22 = overall_liking_scale,
            p22 = overall_liking_scale
)
# 'tab_mis_val(3)' remove 'hard to say' from vector
w %>% tab_cols(total(), age_cat) %>%
  tab_cells(c1r) %>%
  tab_mis_val(3) %>%
  tab_stat_cases() %>%
  tab_last_sig_cases() %>%
  tab_pivot()

# lets specify repeated parts of table creation chains
banner = w %>% tab_cols(total(), age_cat, c1r)
# column percent with significance
tab_cpct_sig = . %>% tab_stat_cpct() %>%
  tab_last_sig_cpct(sig_labels = paste0("<b>",LETTERS, "</b>"))

# means with significance
tab_means_sig = . %>% tab_stat_mean_sd_n(labels = c("<b><u>Mean</u></b>", "sd", "N"))
%>%
  tab_last_sig_means(
    sig_labels = paste0("<b>",LETTERS, "</b>"),
    keep = "means")

# Preferences
banner %>%
  tab_cells(c1r) %>%
  tab_cpct_sig() %>%
  tab_pivot()

# Overall liking
banner %>%
  tab_cells(h22) %>%
  tab_means_sig() %>%
  tab_cpct_sig() %>%
  tab_cells(p22) %>%
  tab_means_sig() %>%
  tab_cpct_sig() %>%
  tab_pivot()

# Likes
```

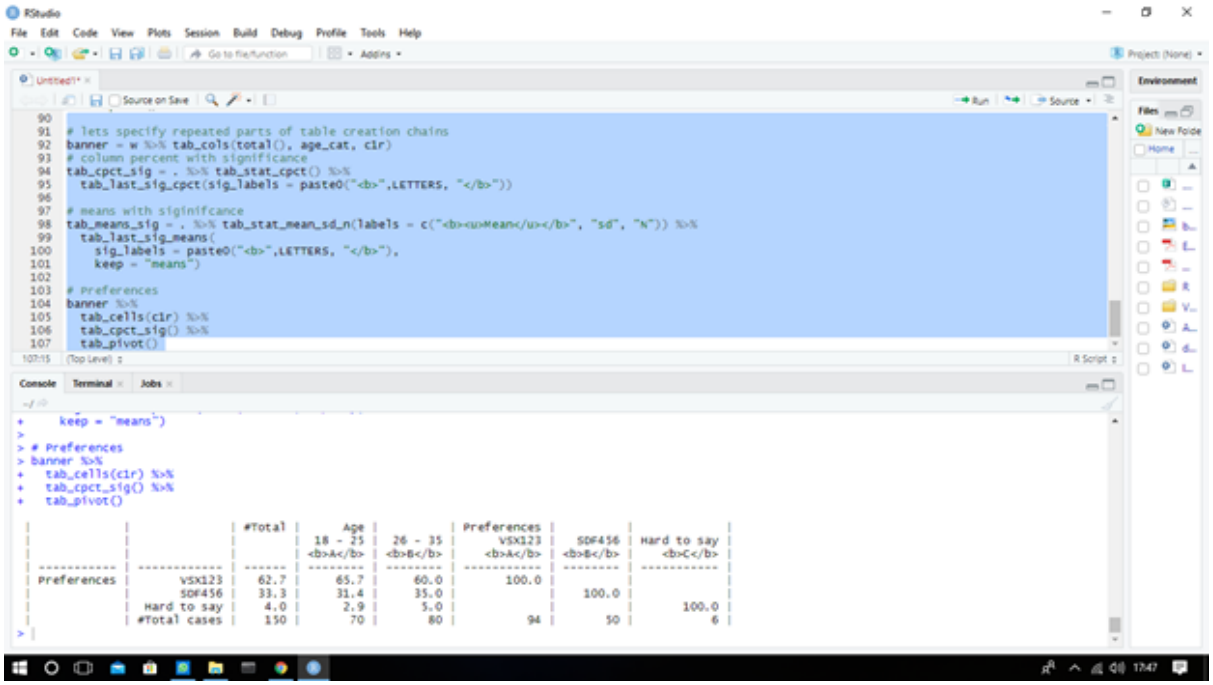
```
banner %>%
  tab_cells(h_likes) %>%
  tab_means_sig() %>%
  tab_cells(mrset(h1_1 %to% h1_6)) %>%
  tab_cpct_sig() %>%
  tab_cells(p_likes) %>%
  tab_means_sig() %>%
  tab_cells(mrset(p1_1 %to% p1_6)) %>%
  tab_cpct_sig() %>%
  tab_pivot()

# below more complicated table where we compare likes side by side
# Likes - side by side comparison
w %>%
  tab_cols(total(label = "#Total| |"), c1r) %>%
  tab_cells(list(unvr(mrset(h1_1 %to% h1_6)))) %>%
  tab_stat_cpct(label = var_lab(h1_1)) %>%
  tab_cells(list(unvr(mrset(p1_1 %to% p1_6)))) %>%
  tab_stat_cpct(label = var_lab(p1_1)) %>%
  tab_pivot(stat_position = "inside_columns")
write_labelled_csv(w, file = "product_test.csv")

library(expss)
w = text_to_columns("
       a  b  c
       2 15 50
       1 70 80
       3 30 40
       2 30 40
")
w$d = ifelse(w$b>60, 1, 0)
w = compute(w, {
  d = ifelse(b>60, 1, 0)
  e = 42
  abc_sum = sum_row(a, b, c)
  abc_mean = mean_row(a, b, c)
})
count_if(1, w)
calculate(w, count_if(1, a, b, c))

w$d = count_row_if(gt(1), w)
w = compute(w, {
  d = count_row_if(gt(1), a, b, c)
})
count_col_if(le(1), w$a)
sum(w, na.rm = TRUE)
w$d = mean_row(w)
#or
w = compute(w, {
  d = mean_row(a, b, c)
```

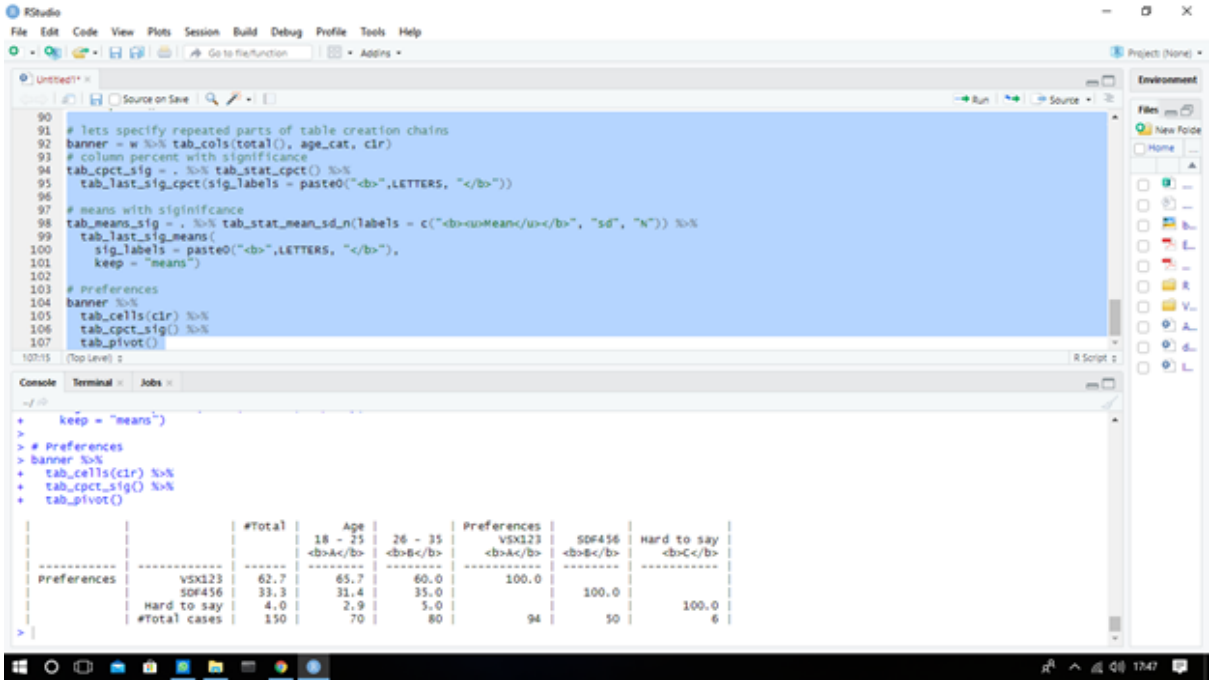
```
})
sum_col(w$a)
sum_if(gt(40), w)
#or
calculate(w, sum_if(gt(40), a, b, c))
w$d = sum_row_if(lt(40), w)
#or
w = compute(w, {
  d = sum_row_if(lt(40), a, b, c)
})
mean_col_if(lt(3), w$a, data = w$b)
#or
calculate(w, mean_col_if(lt(3), a, data = sheet(b, c)))

dict = text_to_columns("
    x  y
    1  apples
    2  oranges
    3  peaches
")
w$d = vlookup(w$a, dict, 2)
#or
w$d = vlookup(w$a, dict, "y")
```

Screenshots:




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
h1_1 = codeframe_likes,
p1_1 = codeframe_likes,
h_likes = "Number of likes. VSX123",
p_likes = "Number of likes. SDF456",
h22 = "Overall quality. VSX123",
p22 = "Overall quality. SDF456",
h22 = overall_liking_scale,
p22 = overall_liking_scale
)
'tab_mis_val(3)' remove 'Hard to say' from vector
w %>% tab_cols(total(), age_cat) %>%
tab_cells(c1r) %>%
tab_mis_val(3) %>%
tab_stat_cases() %>%
tab_last_sig_cases() %>%
tab_pivot()
Console Terminal Jobs
> # 'tab_mis_val(3)' remove 'Hard to say' from vector
> w %>% tab_cols(total(), age_cat) %>%
+ tab_cells(c1r) %>%
+ tab_mis_val(3) %>%
+ tab_stat_cases() %>%
+ tab_last_sig_cases() %>%
+ tab_pivot()
		#Total	Age	
			18 - 25	26 - 35
Preferences	VSX123	94.0	46.0	48.0
	SDF456	50.0	22.0	28.0
#Chi-squared p-value	<0.05	(warn.)		
#Total cases	144.0	68.0	76.0

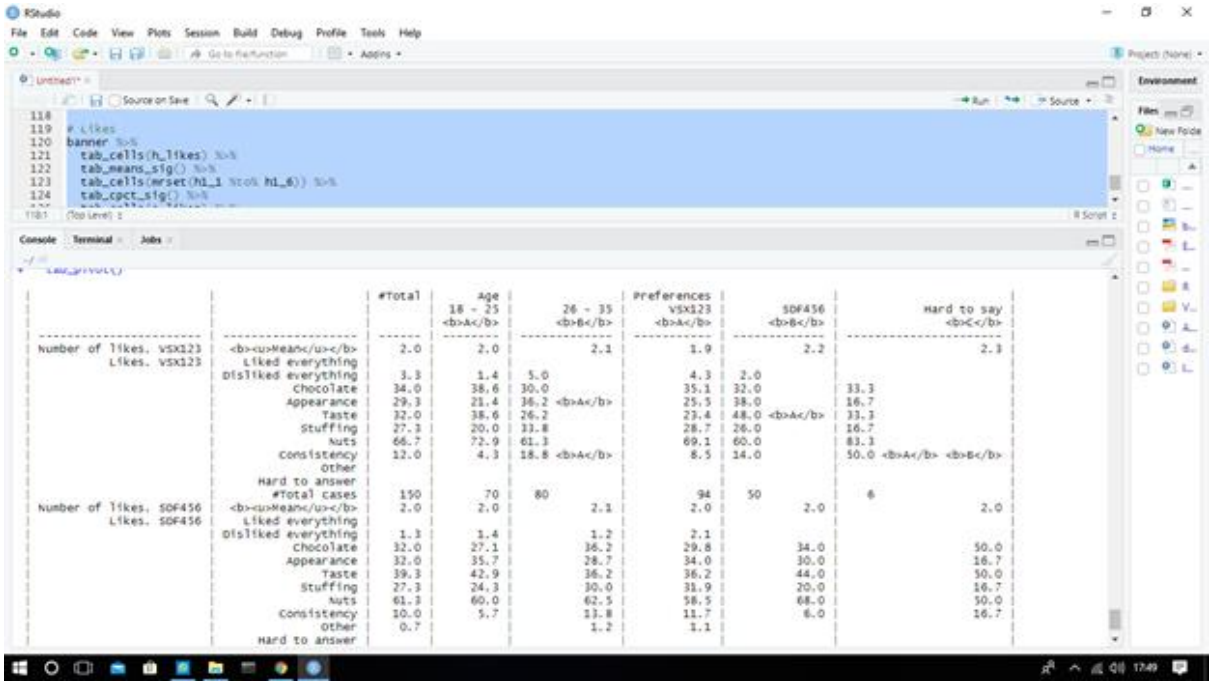

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
# lets specify repeated parts of table creation chains
banner = w %>% tab_cols(total(), age_cat, c1r)
# column percent with significance
tab_cpct_sig = . %>% tab_stat_cpct() %>%
tab_last_sig_cpct(sig_labels = paste0("",LETTERS, ""))
# means with siginifcance
tab_means_sig = . %>% tab_stat_mean_sd_n(labels = c("<u>Mean</u>", "sd", "N")) %>%
tab_last_sig_means(
sig_labels = paste0("",LETTERS, ""),
keep = "means")
# Preferences
banner %>%
tab_cells(c1r) %>%
tab_cpct_sig() %>%
tab_pivot()
Console Terminal Jobs
+ keep = "means")
>
> # Preferences
> banner %>%
+ tab_cells(c1r) %>%
+ tab_cpct_sig() %>%
+ tab_pivot()
		#Total	Age		Preferences		
			18 - 25	26 - 35	VSX123	SDF456	Hard to say
			A	B	A	B	C
Preferences	VSX123	62.7	65.7	60.0	100.0		
	SDF456	33.3	31.4	35.0		100.0	
	Hard to say	4.0	2.9	5.0			100.0
#Total cases	150	70	80	94	50	6



```
# lets specify repeated parts of table creation chains
banner = w %>% tab_cols(total(), age_cat, c1r)
# column percent with significance
tab_cpct_sig = . %>% tab_stat_cpct() %>%
  tab_last_sig_cpct(sig_labels = paste0("<b>",LETTERS, "</b>"))

# means with siginificance
tab_means_sig = . %>% tab_stat_mean_sd_n(labels = c("<b><u>Mean</u></b>", "sd", "N")) %>%
  tab_last_sig_means(
    sig_labels = paste0("<b>",LETTERS, "</b>"),
    keep = "means")

# Preferences
banner %>%
  tab_cells(c1r) %>%
  tab_cpct_sig() %>%
  tab_pivot()
```

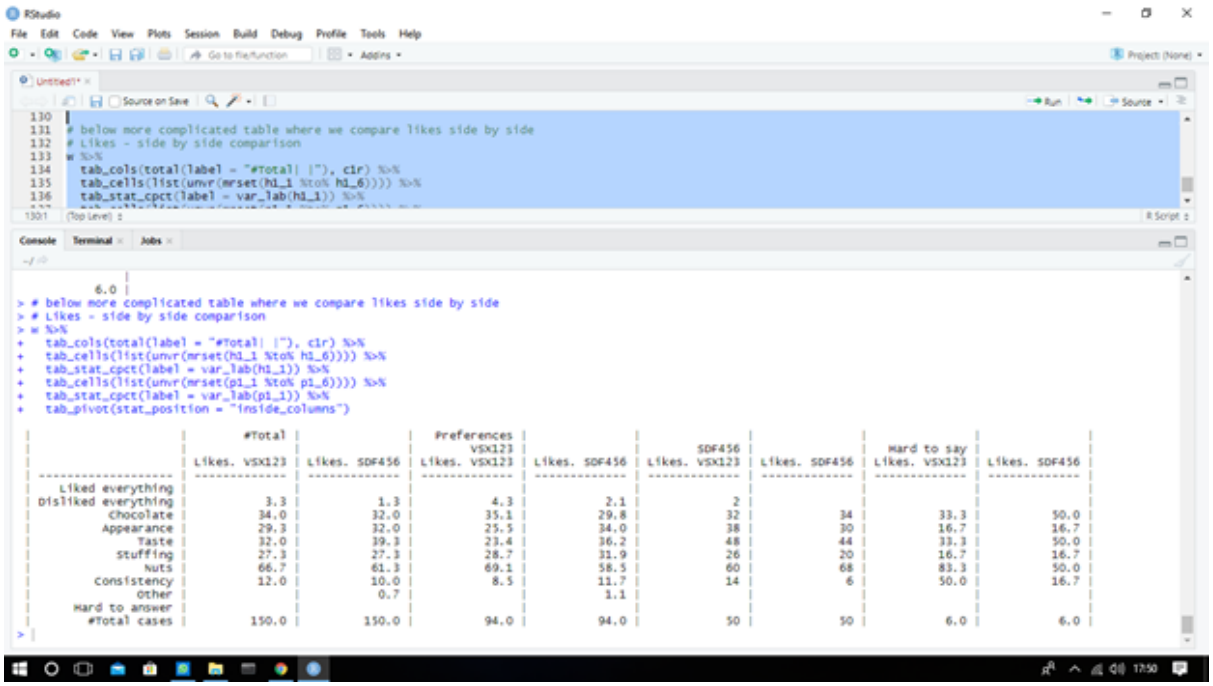
```
+     keep = "means")
>
> # Preferences
> banner %>%
+   tab_cells(c1r) %>%
+   tab_cpct_sig() %>%
+   tab_pivot()
```

		#Total	Age		Preferences		
			18 - 25	26 - 35	VSX123	SDF456	Hard to say
			A	B	A	B	C
Preferences	VSX123	62.7	65.7	60.0	100.0		
	SDF456	33.3	31.4	35.0		100.0	
	Hard to say	4.0	2.9	5.0			100.0
	#Total cases	150	70	80	94	50	6



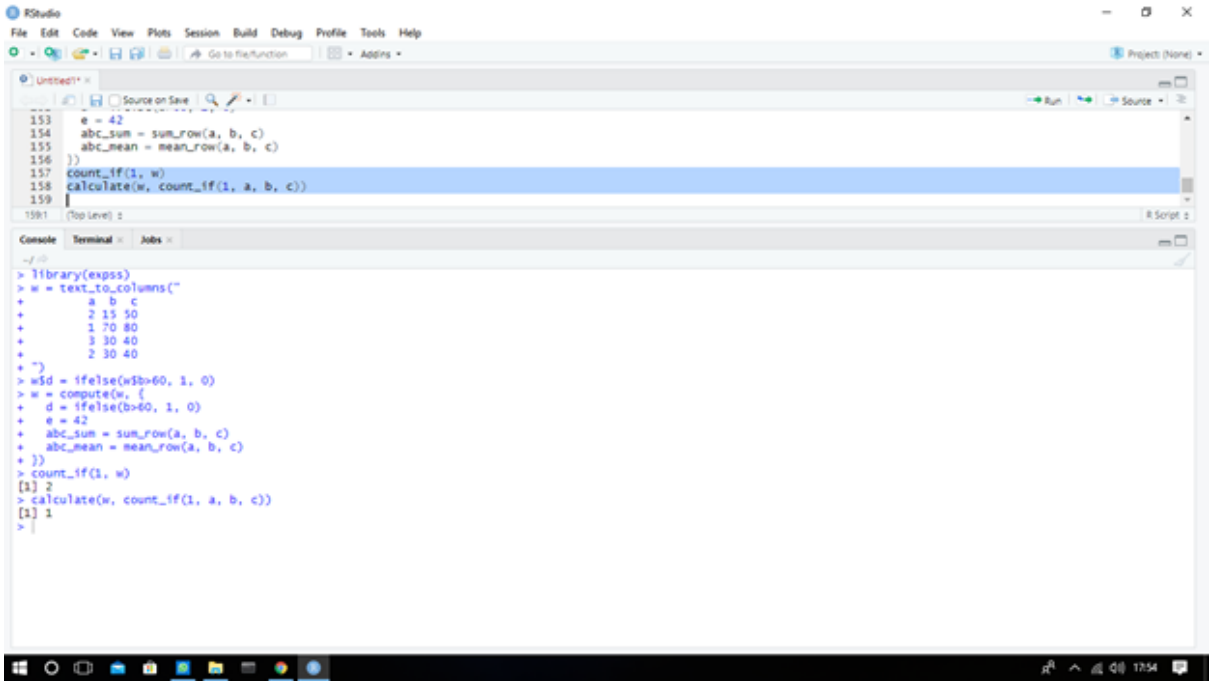
```
# Likes
banner %>%
  tab_cells(h_likes) %>%
  tab_means_sig() %>%
  tab_cells(mrset(h1_1 %to% h1_6)) %>%
  tab_cpct_sig() %>%
```

		#Total	Age		Preferences		
			18 - 25	26 - 35	VSX123	SDF456	Hard to say
			A	B	A	B	C
Number of likes. VSX123	<u>Mean</u>	2.0	2.0	2.1	1.9	2.2	2.3
Likes. VSX123	Liked everything						
	Disliked everything	3.3	1.4	5.0	4.3	2.0	
	Chocolate	34.0	38.6	30.0	35.1	32.0	33.3
	Appearance	29.3	21.4	36.2 A	25.5	38.0	16.7
	Taste	32.0	38.6	26.2	23.4	48.0 A	33.3
	Stuffing	27.3	20.0	33.8	28.7	26.0	16.7
	Nuts	66.7	72.9	61.3	69.1	60.0	83.3
	Consistency	12.0	4.3	18.8 A	8.5	14.0	50.0 A B
	Other						
	Hard to answer						
	#Total cases	150	70	80	94	50	6
Number of likes. SDF456	<u>Mean</u>	2.0	2.0	2.1	2.0	2.0	2.0
Likes. SDF456	Liked everything						
	Disliked everything	1.3	1.4	1.2	2.1		
	Chocolate	32.0	27.1	36.2	29.8	34.0	50.0
	Appearance	32.0	35.7	28.7	34.0	30.0	16.7
	Taste	39.3	42.9	36.2	36.2	44.0	50.0
	Stuffing	27.3	24.3	30.0	31.9	20.0	16.7
	Nuts	61.3	60.0	62.5	58.5	68.0	50.0
	Consistency	10.0	5.7	13.8	11.7	6.0	16.7
	Other	0.7		1.2	1.1		
	Hard to answer						



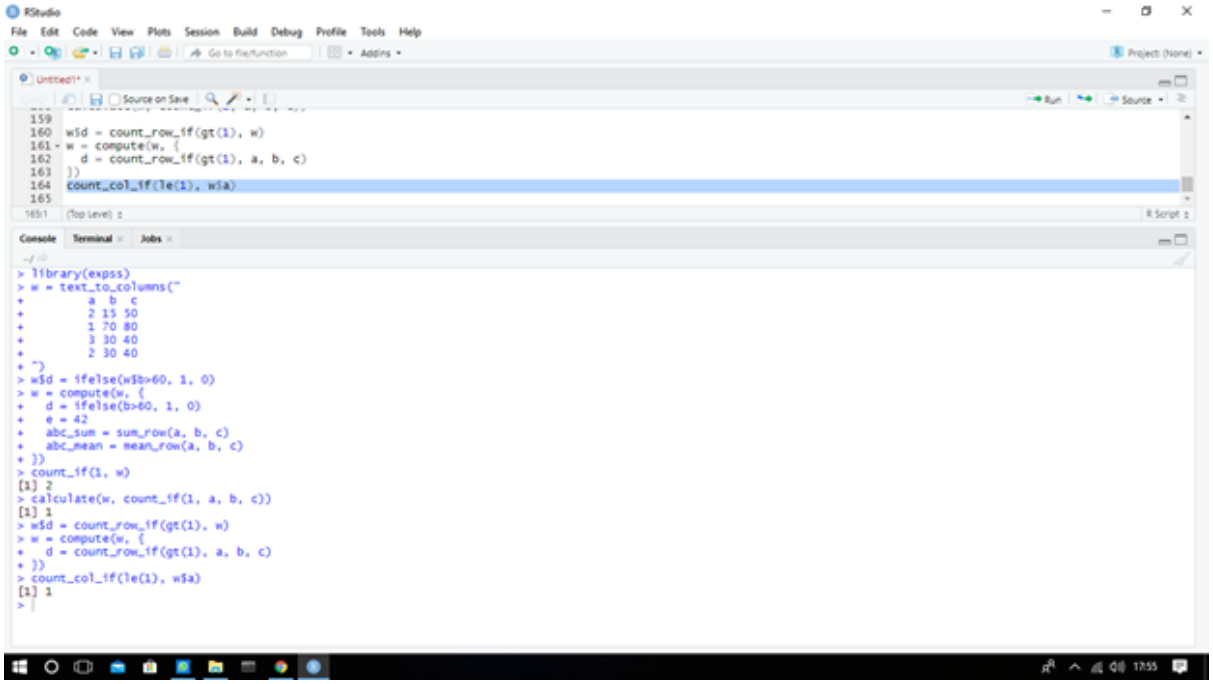
```
# below more complicated table where we compare likes side by side
# Likes - side by side comparison
w %>%
  tab_cols(total(label = "#Total| |"), cir) %>%
  tab_cells(list(unvr(mrset(hl_1 %to% hl_6)))) %>%
  tab_stat_cpct(label = var_lab(hl_1)) %>%
```

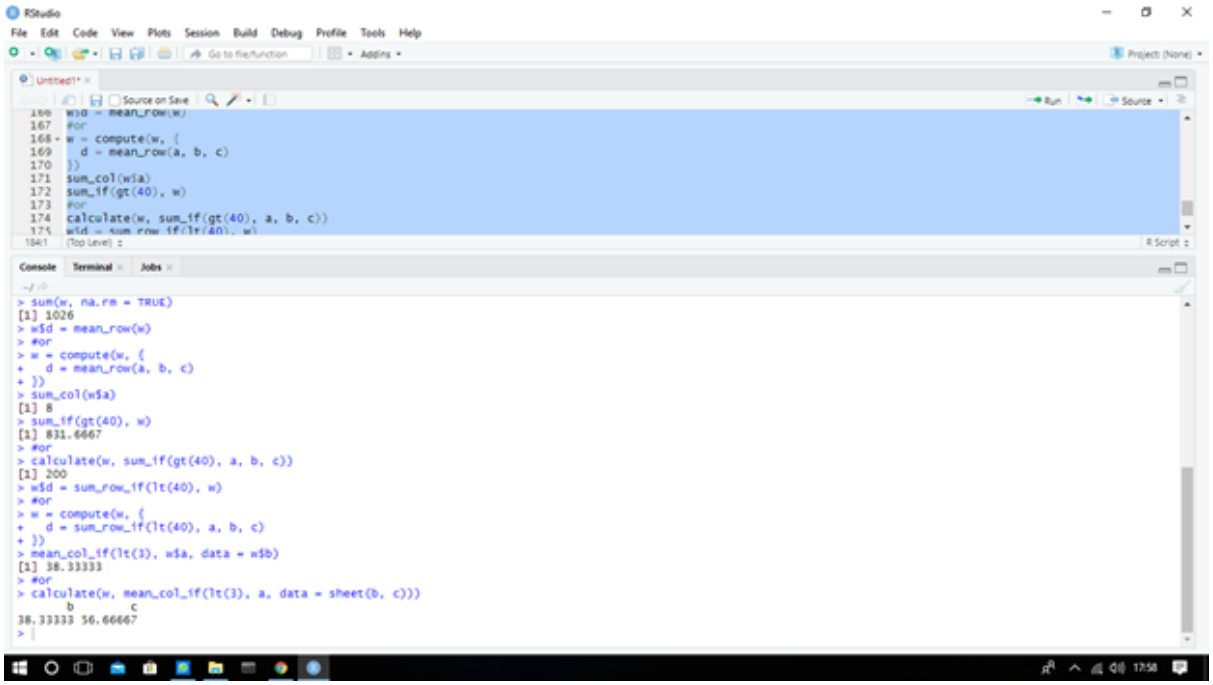
```
          6.0 |
> # below more complicated table where we compare likes side by side
> # Likes - side by side comparison
> w %>%
+   tab_cols(total(label = "#Total| |"), cir) %>%
+   tab_cells(list(unvr(mrset(hl_1 %to% hl_6)))) %>%
+   tab_stat_cpct(label = var_lab(hl_1)) %>%
+   tab_cells(list(unvr(mrset(p1_1 %to% p1_6)))) %>%
+   tab_stat_cpct(label = var_lab(p1_1)) %>%
+   tab_pivot(stat_position = "inside_columns")

|                    |       #Total |              |  Preferences |              |          |          |  Hard to say |              |
|                    |              |              |        VSX123 |              |       SDF456 |          |              |              |
|                    | Likes. VSX123 | Likes. SDF456 | Likes. VSX123 | Likes. SDF456 | Likes. VSX123 | Likes. SDF456 | Likes. VSX123 | Likes. SDF456 |
| ------------------ | ------------ | ------------ | ------------ | ------------ | ------------ | ------------ | ------------ | ------------ |
|    Liked everything |              |              |              |              |              |              |              |              |
| Disliked everything |          3.3 |          1.3 |          4.3 |          2.1 |            2 |              |              |              |
|          Chocolate |         34.0 |         32.0 |         35.1 |         29.8 |           32 |           34 |         33.3 |         50.0 |
|         Appearance |         29.3 |         32.0 |         25.5 |         34.0 |           38 |           30 |         16.7 |         16.7 |
|              Taste |         32.0 |         39.3 |         23.4 |         36.2 |           48 |           44 |         33.3 |         50.0 |
|           Stuffing |         27.3 |         27.3 |         28.7 |         31.9 |           26 |           20 |         16.7 |         16.7 |
|               Nuts |         66.7 |         61.3 |         69.1 |         58.5 |           60 |           68 |         83.3 |         50.0 |
|        Consistency |         12.0 |         10.0 |          8.5 |         11.7 |           14 |            6 |         50.0 |         16.7 |
|              Other |              |          0.7 |              |          1.1 |              |              |              |              |
|     Hard to answer |              |              |              |              |              |              |              |              |
|      #Total cases |        150.0 |        150.0 |         94.0 |         94.0 |           50 |           50 |          6.0 |          6.0 |
> |
```

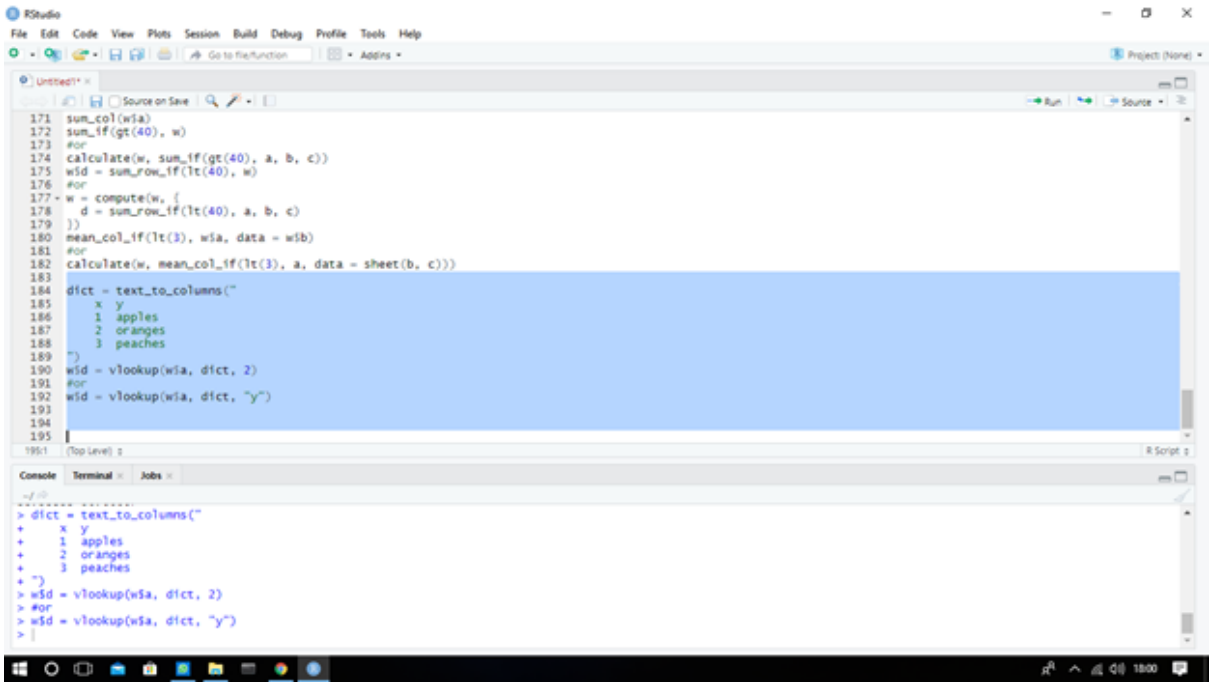


```
  e = 42
  abc_sum = sum_row(a, b, c)
  abc_mean = mean_row(a, b, c)
})
count_if(1, w)
calculate(w, count_if(1, a, b, c))
```

```
> library(expss)
> w = text_to_columns("
+         a  b  c
+         2 15 50
+         1 70 80
+         3 30 40
+         2 30 40
+ ")
> w$d = ifelse(w$b>60, 1, 0)
> w = compute(w, {
+   d = ifelse(b>60, 1, 0)
+   e = 42
+   abc_sum = sum_row(a, b, c)
+   abc_mean = mean_row(a, b, c)
+ })
> count_if(1, w)
[1] 2
> calculate(w, count_if(1, a, b, c))
[1] 1
> |
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
159
160 w$d = count_row_if(gt(1), w)
161 w = compute(w, {
162 d = count_row_if(gt(1), a, b, c)
163 })
164 count_col_if(le(1), w$a)
165
165:1 (Top Level)
R Script
Console Terminal Jobs
> library(expss)
> w = text_to_columns("
+ a b c
+ 2 15 50
+ 1 70 80
+ 3 30 40
+ 2 30 40
+ ")
> w$d = ifelse(w$b>60, 1, 0)
> w = compute(w, {
+ d = ifelse(b>60, 1, 0)
+ e = 42
+ abc_sum = sum_row(a, b, c)
+ abc_mean = mean_row(a, b, c)
+ })
> count_if(1, w)
[1] 2
> calculate(w, count_if(1, a, b, c))
[1] 1
> w$d = count_row_if(gt(1), w)
> w = compute(w, {
+ d = count_row_if(gt(1), a, b, c)
+ })
> count_col_if(le(1), w$a)
[1] 1
>
17:55


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
167 #or
168 w = compute(w, {
169 d = mean_row(a, b, c)
170 })
171 sum_col(w$a)
172 sum_if(gt(40), w)
173 #or
174 calculate(w, sum_if(gt(40), a, b, c))
184:1 (Top Level)
R Script
Console Terminal Jobs
> sum(w, na.rm = TRUE)
[1] 1026
> w$d = mean_row(w)
> #or
> w = compute(w, {
+ d = mean_row(a, b, c)
+ })
> sum_col(w$a)
[1] 8
> sum_if(gt(40), w)
[1] 831.6667
> #or
> calculate(w, sum_if(gt(40), a, b, c))
[1] 200
> w$d = sum_row_if(lt(40), w)
> #or
> w = compute(w, {
+ d = sum_row_if(lt(40), a, b, c)
+ })
> mean_col_if(lt(3), w$a, data = w$b)
[1] 38.33333
> #or
> calculate(w, mean_col_if(lt(3), a, data = sheet(b, c)))
b c
38.33333 56.66667
>
17:58



Conclusion: Thus, we have studied tables with labels in R.

Experiment No. 6

Aim : Working with large datasets with dplyr and data.table

Theory :

Data manipulation operations such as subset, group, update, join etc., are all inherently related. Keeping these related operations together allows for:

- concise and consistent syntax irrespective of the set of operations you would like to perform to achieve your end goal.
- performing analysis fluidly without the cognitive burden of having to map each operation to a particular function from a potentially huge set of functions available before performing the analysis.
- automatically optimising operations internally, and very effectively, by knowing precisely the data required for each operation, leading to very fast and memory efficient code.

data.table is an R package that provides an enhanced version of data.frames, which are the standard data structure for storing data in base R.

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

- mutate() adds new variables that are functions of existing variables
- select() picks variables based on their names.
- filter() picks cases based on their values.
- summarise() reduces multiple values down to a single summary.
- arrange() changes the ordering of the rows.

These all combine naturally with group_by() which allows you to perform any operation “by group”. dplyr is designed to abstract over how the data is stored. That means as well as working with local data frames, you can also work with remote database tables, using exactly the same R code. Install the dplyr package then read vignette("databases", package = "dbplyr").

Output :

Code :

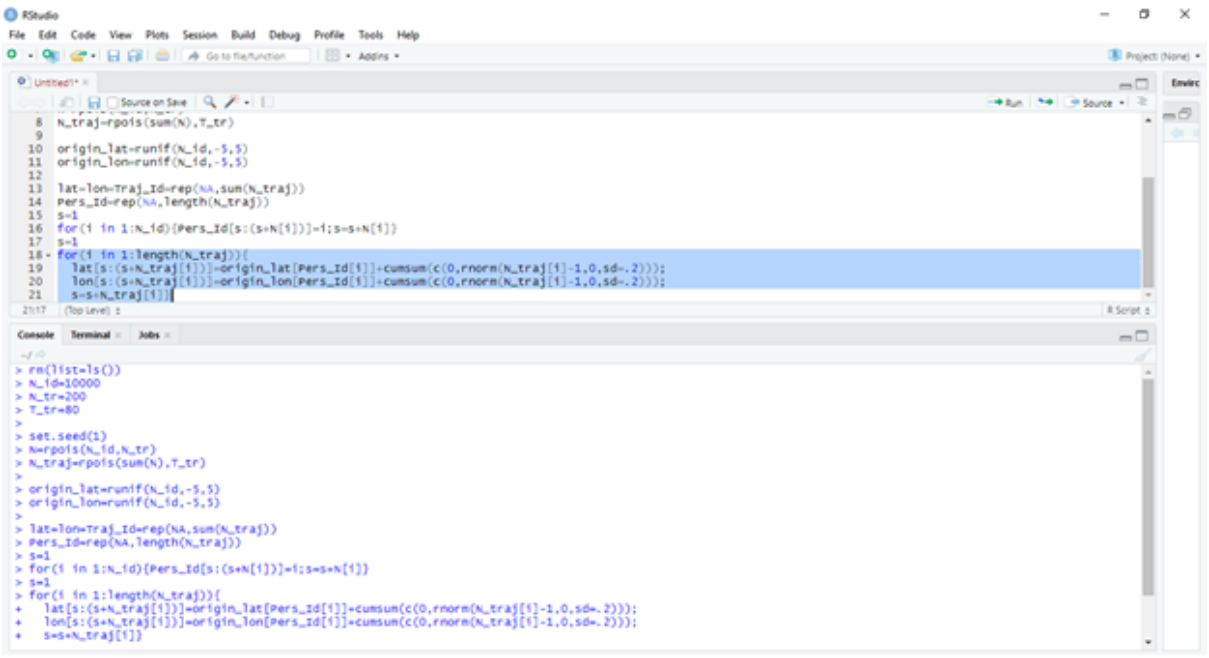
```
rm(list=ls())
N_id=10000
N_tr=200
T_tr=80
set.seed(1)
N=rpois(N_id,N_tr)
N_traj=rpois(sum(N),T_tr)
```

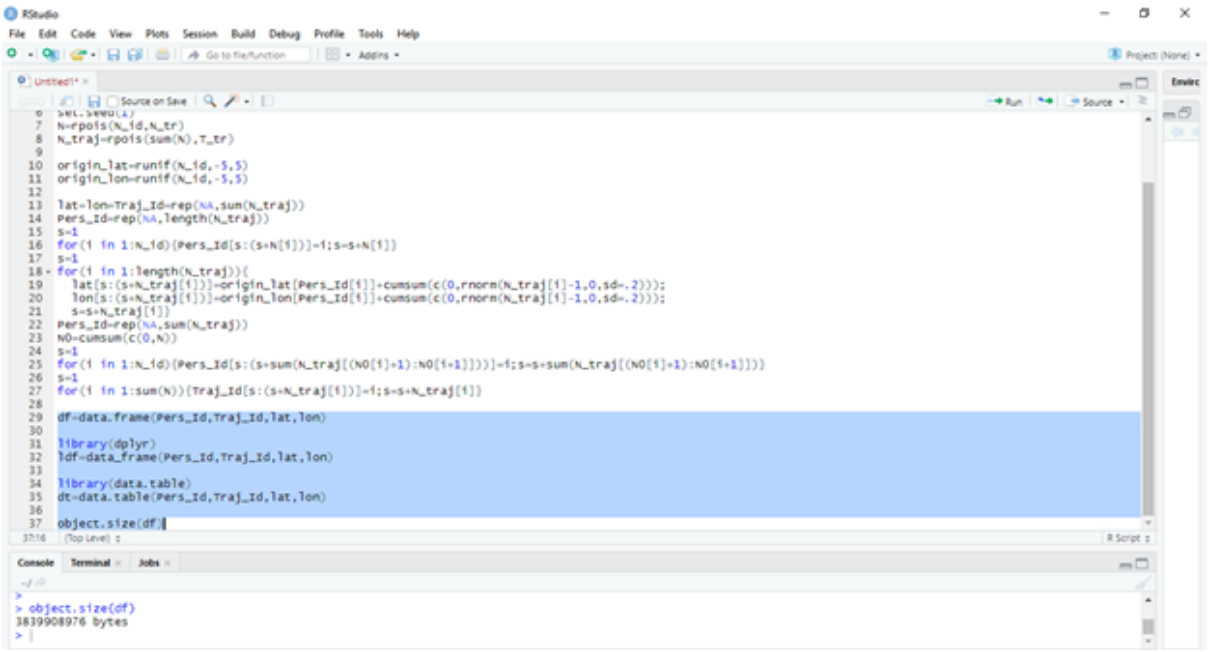
```
origin_lat=runif(N_id,-5,5)
origin_lon=runif(N_id,-5,5)
lat=lon=Traj_Id=rep(NA,sum(N_traj))
Pers_Id=rep(NA,length(N_traj))
s=1
for(i in 1:N_id){Pers_Id[s:(s+N[i])]=i;s=s+N[i]}
s=1
for(i in 1:length(N_traj))
  {
  lat[s:(s+N_traj[i])]=origin_lat[Pers_Id[i]]+cumsum(c(0,rnorm(N_traj[i]-1,0,sd=.2)));
  lon[s:(s+N_traj[i])]=origin_lon[Pers_Id[i]]+cumsum(c(0,rnorm(N_traj[i]-1,0,sd=.2)));
  s=s+N_traj[i]
  }

Pers_Id=rep(NA,sum(N_traj))
N0=cumsum(c(0,N))
s=1
for(i in
1:N_id){Pers_Id[s:(s+sum(N_traj[(N0[i]+1):N0[i+1]]))]=i;s=s+sum(N_traj[(N0[i]+1):N0[i+1
]])}
s=1
for(i in 1:sum(N)){Traj_Id[s:(s+N_traj[i])]=i;s=s+N_traj[i]}

df=data.frame(Pers_Id,Traj_Id,lat,lon)
library(dplyr)
ldf=data_frame(Pers_Id,Traj_Id,lat,lon)

library(data.table)
dt=data.table(Pers_Id,Traj_Id,lat,lon)

object.size(df)

rm(list=ls())
library(data.table)
system.time( load("dt_json_2.RData") )

dropbox_ldf <- "https://dl.dropboxusercontent.com/s/l7rolinwojn37e2/ldf_json_2.RData"
dropbox_df  <- "https://dl.dropboxusercontent.com/s/wzr19v9pyl7ah0j/df_json_2.RData"
dropbox_dt  <- "https://dl.dropboxusercontent.com/s/nlwi1mmsxr5f23k/dt_json_2.RData"

source_https <- function(loc,...)
  {
  curl=getCurlHandle()
  curlSetOpt(cookiejar="cookies.txt", useragent="Mozilla/5.0", followlocation=TRUE)
  tmp <- getURLContent(loc, .opts=list(ssl.verifypeer=FALSE),curl=curl)
  fch <- source(tmp)
  fch
}

system.time(load("df_json_2.RData"))
```

```
system.time(n <- nrow(df))
system.time(df$first<-c(1,df$Traj_Id[2:n]!=df$Traj_Id[1:(n-1)]))
system.time(df$last<-c(df$Traj_Id[2:n]!=df$Traj_Id[1:(n-1)],1))
object.size(df)

lat_0=0
lon_0=0
system.time(df$test <-(lat_0-df$lat)^2+(lon_0-df$lon)^2) <=1)&(df$last==1)

f$first <- NULL
df$last <- NULL
object.size(df)
system.time(df$test<-(lat_0-df$lat)^2+(lon_0-df$lon)^2) <=1&(c(df$Traj_Id[2:n]!=
df$Traj_Id[1:(n-1)],1)==1)
system.time(list_Traj <- unique(df$Traj_Id[df$test==TRUE]))
system.time(base <- df[(df$Traj_Id %in% list_Traj)& (c(1,df$Traj_Id[2:n]!=df$Traj_Id[1:(n-
1)])==1),c("lat","lon")])
head(base)

X <- base[,c("lon","lat")]
library(KernSmooth)
kde2d <- bkde2D(X, bandwidth=c(bw.ucv(X[,1]),bw.ucv(X[,2])),gridsize = c(251L, 251L))
image(x=kde2d$x1, y=kde2d$x2,z=kde2d$fhat,col=rev(heat.colors(100)))
contour(x=kde2d$x1, y=kde2d$x2,z=kde2d$fhat, add=TRUE)

rm(list=ls())
library(data.table)
system.time( load("dt_json_2.RData") )

system.time( setkey(dt,Traj_Id) )
system.time(depart <-dt[!duplicated(Traj_Id)])
system.time( depart <- dt[J(unique(Traj_Id)), mult = "first"])
system.time( arrivee <- dt[J(unique(Traj_Id)), mult = "last"] )
lat_0=0
lon_0=0
system.time( arrivee[,dist:=(lat-lat_0)^2+(lon-lon_0)^2] )

system.time( fin <- subset(arrivee,dist <= 1) )
system.time( fin[,Pers_Id:=NULL] )
system.time( fin[,lat:=NULL] )
system.time( fin[,lon:=NULL] )
system.time( setkey(fin, Traj_Id) )
system.time( base <<- merge(fin,depart,all.x=TRUE) )
system.time( base <<- merge(fin,depart,all.x=TRUE) )
head(base)
rm(list=ls())
library(dplyr)
library(data.table)
system.time( load("ldf_json_2.RData") )
system.time( ldepart <<- ldf %>% group_by(Traj_Id) %>%
```

```
            + summarise(first_lat=head(lat,1),
                        first_lon=head(lon,1)) )
system.time( larrive <<- ldf %>% group_by(Traj_Id) %>%
            + summarise(last_lat=tail(lat,1),last_lon=tail(lon,1)) )
lat_0=0
lon_0=0
system.time( system.time( larrive <<- mutate(larrive,dist=(last_lat-lat_0)^2+(last_lon-
lon_0)^2) ))
system.time( lfin <<- filter(larrive,dist<=1) )
system.time( lbase <- left_join(lfin,ldepart) )
head(lbase)
```

Screenshots :





Conclusion : Thus, we have studied working with large datasets with dplyr and data.table.

Experiment No : 7

Aim : EDA with R - Range, Summary, Mean, Variance, Median, Standard Deviation, Histogram, Box plot, Scatter Graph.

Theory:

Exploratory data analysis or "EDA" is a critical first step in analyzing the data from an experiment. Here are the main reasons we use EDA:

- detection of mistakes
- checking of assumptions
- preliminary selection of appropriate models
- determining relationships among the explanatory variables, and
- assessing the direction and rough size of relationships between explanatory and outcome variables.

Loosely speaking, any method of looking at data that does not include formal statistical modeling and inference falls under the term exploratory data analysis.

Types of EDA:

Exploratory data analysis is generally cross-classified in two ways. First, each method is either non-graphical or graphical. And second, each method is either univariate or multivariate (usually just bivariate).
Non-graphical methods generally involve calculation of summary statistics, while graphical methods obviously summarize the data in a diagrammatic or pictorial way. Univariate methods look at one variable (data column) at a time, while multivariate methods look at two or more variables at a time to explore relationships. Usually our multivariate EDA will be bivariate (looking at exactly two variables), but occasionally it will involve three or more variables.

Univariate non-graphical EDA:

The data that come from making a particular measurement on all of the subjects in a sample represent our observations for a single characteristic such as age, gender, speed at a task, or response to a stimulus. We should think of these measurements as representing a "sample distribution" of the variable, which in turn more or less represents the "population distribution" of the variable. The usual goal of univariate non-graphical EDA is to better appreciate the "sample distribution" and also to make some tentative conclusions about what population distribution(s) is/are compatible with the sample distribution. Outlier detection is also a part of this analysis.

1. Categorical data
2. Quantitative data

Several statistics are commonly used as a measure of the spread of a distribution, including variance, standard deviation, and interquartile range. Spread is an indicator of how far away from the center we are still likely to find data values. The variance is a standard measure of spread. It is calculated for a list of numbers, e.g., the n observations of a particular measurement labeled x1 through xn, based on the n sample deviations (or just "deviations").A third measure of spread is the interquartile range. To define IQR, we first need to define the concepts of quartiles. The quartiles of a population or a sample are the three values which divide the distribution or observed data into even fourths.

Skewness is a measure of asymmetry. Kurtosis is a more subtle measure of peakedness compared to a Gaussian distribution.

Univariate graphical EDA:
If we are focusing on data from observation of a single variable on n subjects, i.e., a sample of size n, then in addition to looking at the various sample statistics discussed in the previous section, we also need to look graphically at the distribution of the sample. Non-graphical and graphical methods complement each other.

1. Histogram
2. Stem and leaf graph - A stem and leaf plot shows all data values and the shape of the distribution.
3. Boxplots - Boxplots show robust measures of location and spread as well as providing information about symmetry and outliers.
4. Quantile-normal plots - Quantile-Normal plots allow detection of non-normality and diagnosis of skewness and kurtosis.

Multivariate non-graphical EDA:

Multivariate non-graphical EDA techniques generally show the relationship between two or more variables in the form of either cross-tabulation or statistics.

Multivariate graphical EDA:

There are few useful techniques for graphical EDA of two categorical random variables. The only one used commonly is a grouped barplot with each group representing one level of one of the variables and each bar within a group representing the levels of the other variable.

Output:

Code:

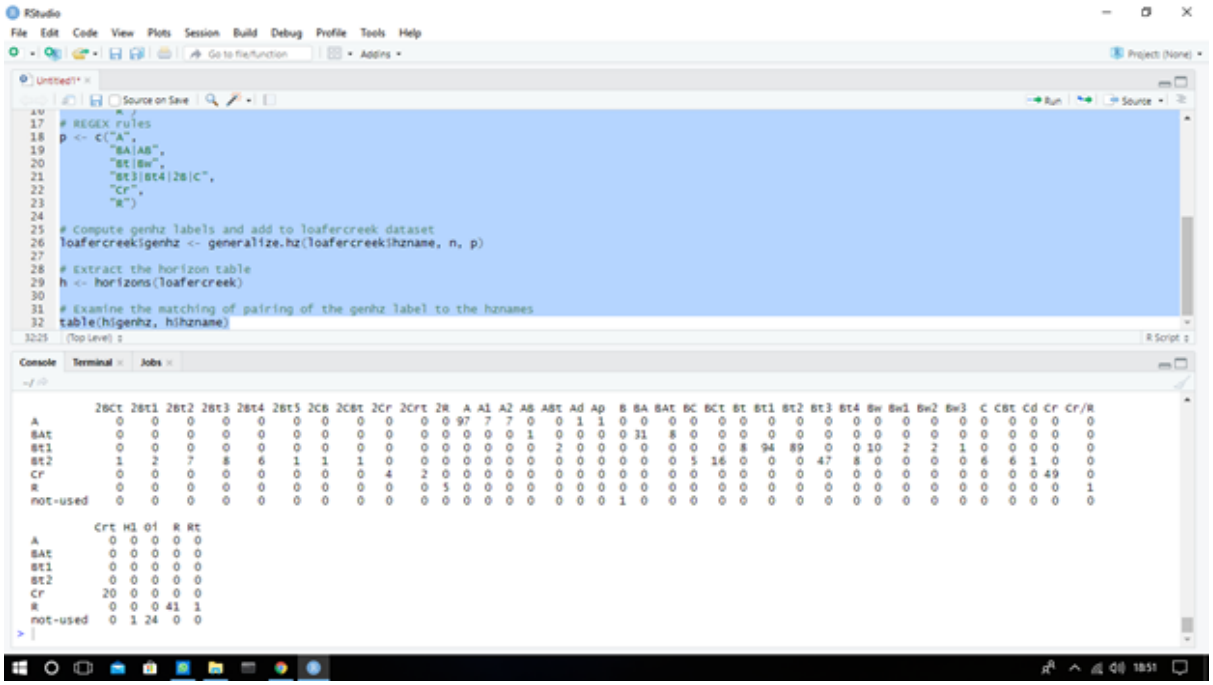
```
install.packages('aqp', dep=TRUE)
install.packages("soilDB")
```

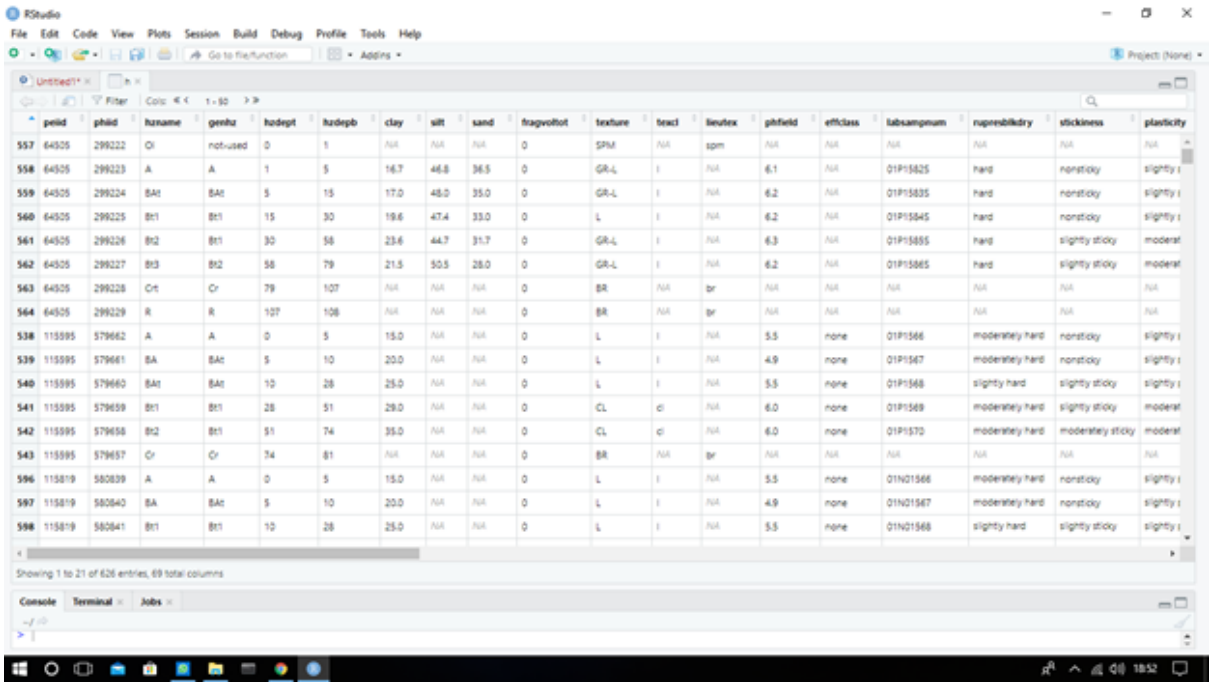
```
library(aqp)
library(soilDB)

# Load from the the loakercreek dataset
data("loafercreek")

# Construct generalized horizon designations
n <- c("A",
       "BAt",
       "Bt1",
       "Bt2",
       "Cr",
       "R")
# REGEX rules
p <- c("A",
       "BA|AB",
       "Bt|Bw",
       "Bt3|Bt4|2B|C",
       "Cr",
       "R")

# Compute genhz labels and add to loafercreek dataset
loafercreek$genhz <- generalize.hz(loafercreek$hzname, n, p)

# Extract the horizon table
h <- horizons(loafercreek)

# Examine the matching of pairing of the genhz label to the hznames
table(h$genhz, h$hzname)
View(h)
vars <- c("genhz", "clay", "total_frags_pct", "phfield", "effclass")
summary(h[, vars])

# just for factors
levels(h$genhz)
# for characters and factors
sort(unique(h$hzname))

h$hzname <- ifelse(h$hzname == "BT", "Bt", h$hzname)

# or

h$hzname[h$hzname == "BT"] <- "Bt"

# or as a last resort we could manually edit the spreadsheet in R

edit(h)

# first remove missing values and create a new vector
```

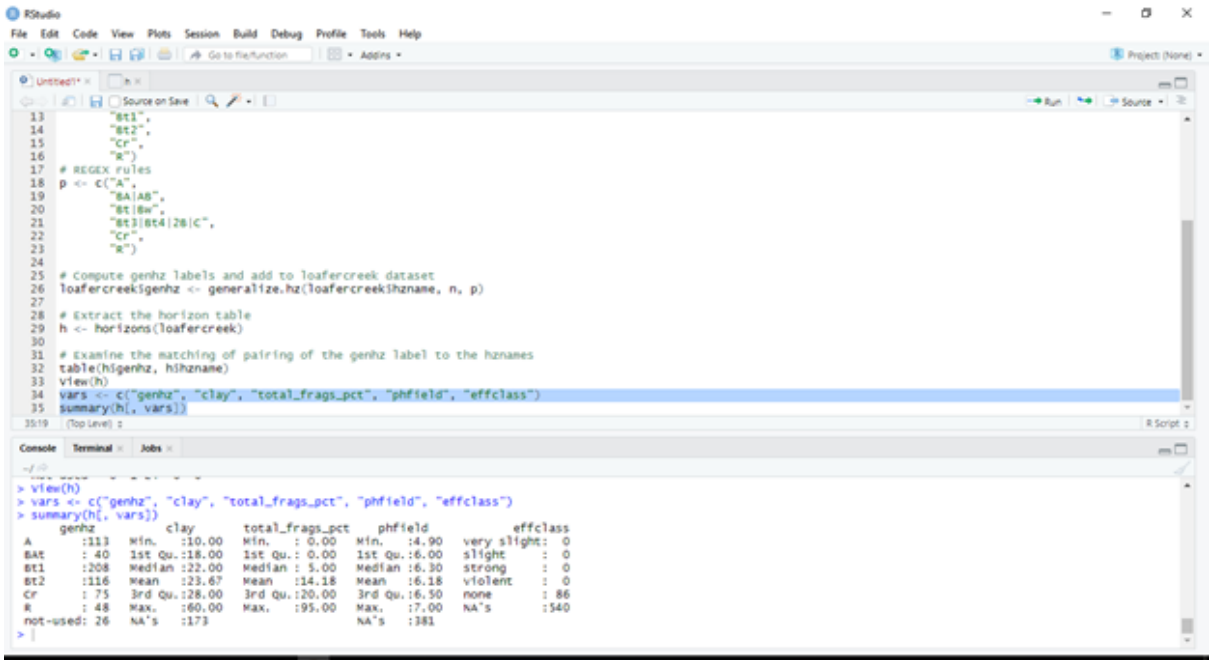
```
clay <- na.exclude(h$clay)
mean(clay)

mean(h$clay, na.rm = TRUE)
median(clay)
sort(table(round(h$clay)), decreasing = TRUE)[1] # sort and select the 1st value, which will be the mode
table(h$genhz)
table(h$genhz, h$texcl)


# append the table with row and column sums
addmargins(table(h$genhz, h$texcl))

# calculate the proportions relative to the rows, margin = 1 calculates for rows, margin = 2 calculates for columns, margin = NULL calculates for total observations
round(prop.table(table(h$genhz, h$texture_class), margin = 1) * 100)
knitr::kable(addmargins(table(h$genhz, h$texcl)))

aggregate(clay ~ genhz, data = h, mean)
aggregate(clay ~ genhz, data = h, median)
# or we could use the summary() function to get both the mean and median
aggregate(clay ~ genhz, data = h, summary)

var(h$clay, na.rm=TRUE)
sd(h$clay, na.rm = TRUE)
cv <- sd(clay) / mean(clay) * 100
cv

quantile(clay)
range(clay)
diff(range(clay))
IQR(clay)

install.packages("ggplot2")
library(ggplot2)

# bar plot
ggplot(h, aes(x = texcl)) +
  geom_bar()
ggplot(h, aes(x = clay)) +
  geom_histogram(bins = nclass.Sturges(h$clay))
ggplot(h, (aes(x = genhz, y = clay))) +
  geom_boxplot()

h$hzdepm <- (h$hzdepb + h$hzdept) / 2 # Compute the middle horizon depth

vars <- c("hzdepm", "clay", "sand", "total_frags_pct", "phfield")

round(cor(h[, vars], use = "complete.obs"), 2)
```

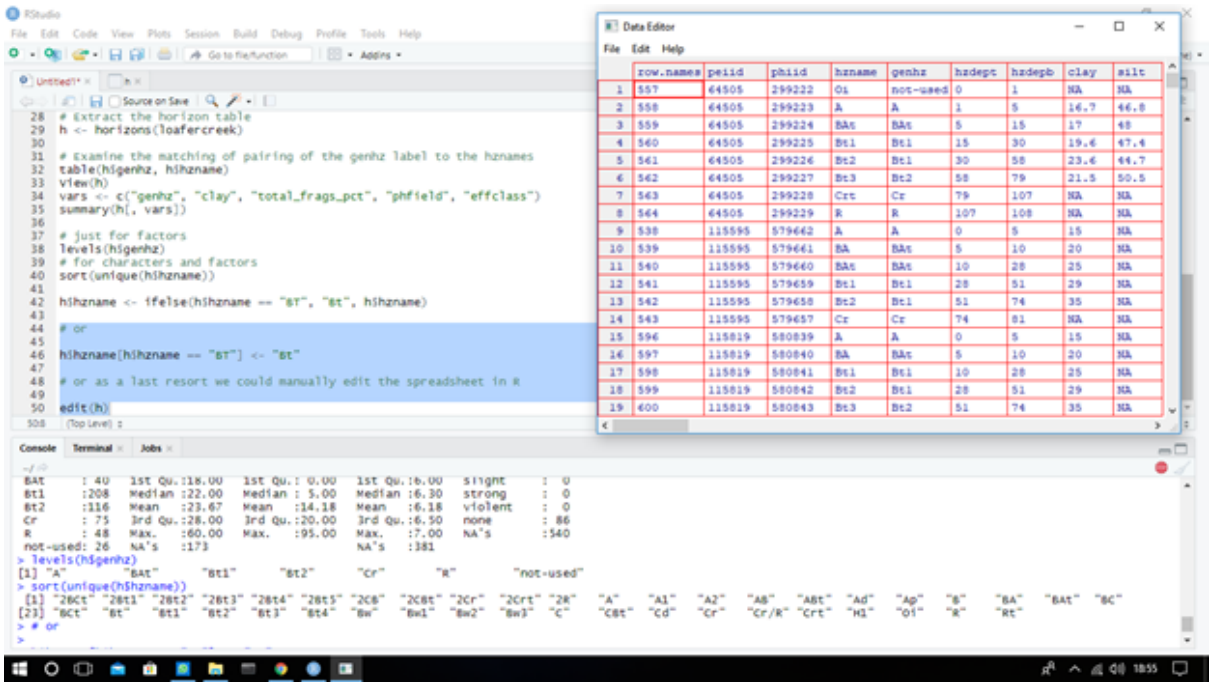
```
# scatter plot
ggplot(h, aes(x = clay, y = hzdepm)) +
  geom_point() +
  ylim(100, 0)
```

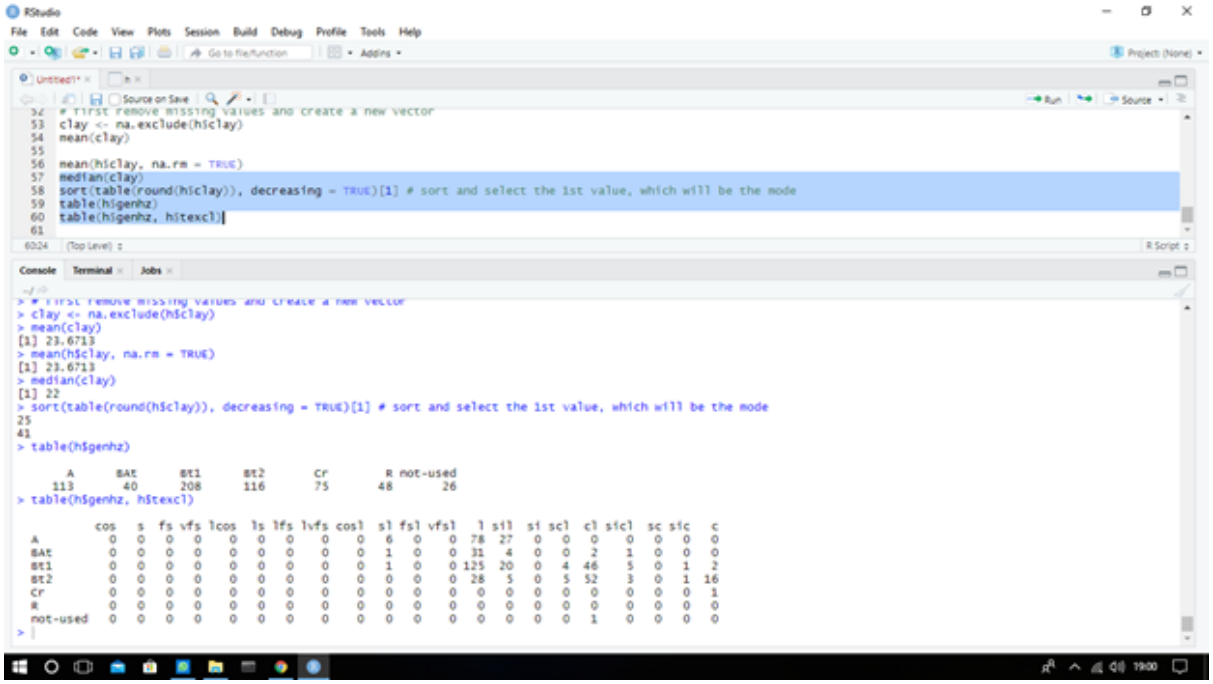
Screenshots:





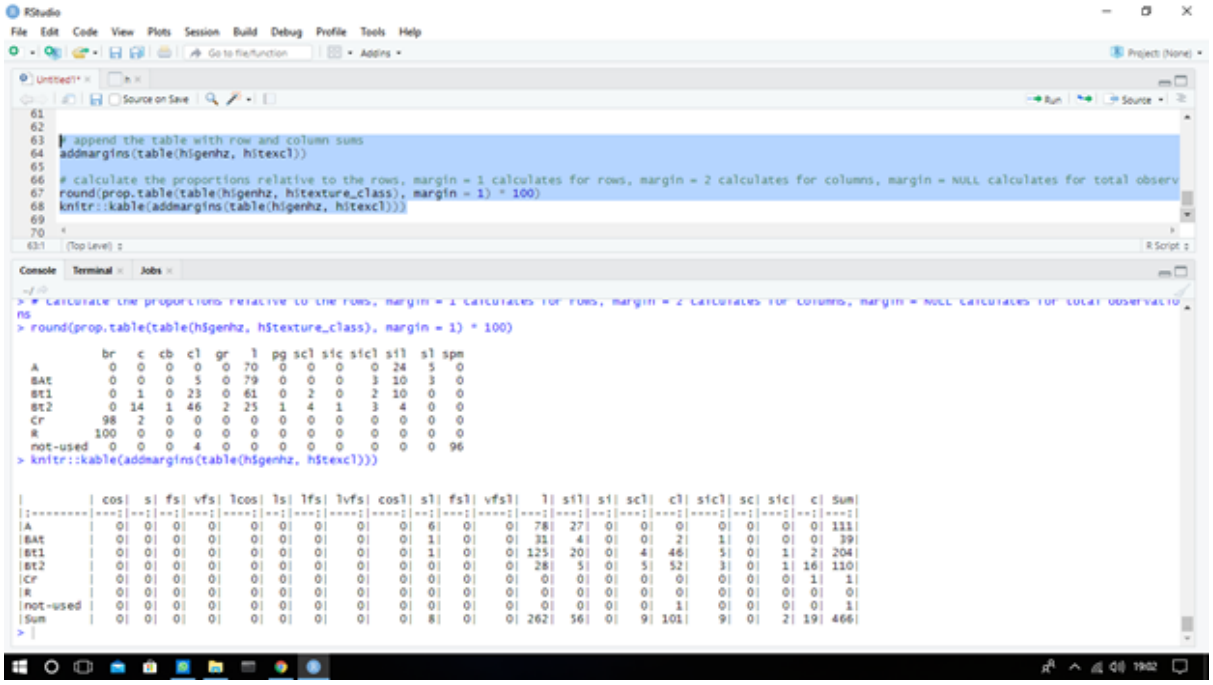

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
"Bt1",
"Bt2",
"Cr",
"R")
# REGEX rules
p <- c("A",
"BA|AB",
"Bt|Bw",
"Bt3|Bt4|2B|C",
"Cr",
"R")
Compute genhz labels and add to loafercreek dataset
loafercreek$genhz <- generalize.hz(loafercreek$hzname, n, p)
# Extract the horizon table
h <- horizons(loafercreek)
# Examine the matching of pairing of the genhz label to the hznames
table(h$genhz, h$hzname)
view(h)
vars <- c("genhz", "clay", "total_frags_pct", "phfield", "effclass")
summary(h[, vars])
Console Terminal Jobs
> view(h)
> vars <- c("genhz", "clay", "total_frags_pct", "phfield", "effclass")
> summary(h[, vars])
genhz clay total_frags_pct phfield effclass
A :113 Min. :10.00 Min. : 0.00 Min. :4.90 very slight: 0
BAt : 40 1st Qu.:18.00 1st Qu.: 0.00 1st Qu.:6.00 slight : 0
Bt1 :208 Median :22.00 Median : 5.00 Median :6.30 strong : 0
Bt2 :116 Mean :23.67 Mean :14.18 Mean :6.18 violent : 0
Cr : 75 3rd Qu.:28.00 3rd Qu.:20.00 3rd Qu.:6.50 none : 86
R : 48 Max. :60.00 Max. :95.00 Max. :7.00 NA's :540
not-used: 26 NA's :173 NA's :381


RStudio
# Extract the horizon table
h <- horizons(loafercreek)
# Examine the matching of pairing of the genhz label to the hznames
table(h$genhz, h$hzname)
view(h)
vars <- c("genhz", "clay", "total_frags_pct", "phfield", "effclass")
summary(h[, vars])
# just for factors
levels(h$genhz)
# for characters and factors
sort(unique(h$hzname))
h$hzname <- ifelse(h$hzname == "BT", "Bt", h$hzname)
or
h$hzname[h$hzname == "BT"] <- "Bt"
# or as a last resort we could manually edit the spreadsheet in R
edit(h)
Data Editor
File Edit Help
row.names peiid phiid hzname genhz hzdept hzdepb clay silt
1 557 64505 299222 Oi not-used 0 1 NA NA
2 558 64505 299223 A A 1 5 16.7 46.8
3 559 64505 299224 BAt BAt 5 15 17 48
4 560 64505 299225 Bt1 Bt1 15 30 19.4 47.4
5 561 64505 299226 Bt2 Bt1 30 58 23.4 44.7
6 562 64505 299227 Bt3 Bt2 58 79 21.5 50.5
7 563 64505 299228 Crt Cr 79 107 NA NA
8 564 64505 299229 R R 107 108 NA NA
9 538 115595 579662 A A 0 5 15 NA
10 539 115595 579661 BA BAt 5 10 20 NA
11 540 115595 579660 BAt BAt 10 28 25 NA
12 541 115595 579659 Bt1 Bt1 28 51 29 NA
13 542 115595 579658 Bt2 Bt1 51 74 35 NA
14 543 115595 579657 Cr Cr 74 81 NA NA
15 596 115819 580839 A A 0 5 15 NA
16 597 115819 580840 BA BAt 5 10 20 NA
17 598 115819 580841 Bt1 Bt1 10 28 25 NA
18 599 115819 580842 Bt2 Bt1 28 51 29 NA
19 600 115819 580843 Bt3 Bt2 51 74 35 NA
Console Terminal Jobs
BAt : 40 1st Qu.:18.00 1st Qu.: 0.00 1st Qu.:6.00 slight : 0
Bt1 :208 Median :22.00 Median : 5.00 Median :6.30 strong : 0
Bt2 :116 Mean :23.67 Mean :14.18 Mean :6.18 violent : 0
Cr : 75 3rd Qu.:28.00 3rd Qu.:20.00 3rd Qu.:6.50 none : 86
R : 48 Max. :60.00 Max. :95.00 Max. :7.00 NA's :540
not-used: 26 NA's :173 NA's :381
> levels(h$genhz)
[1] "A" "BAt" "Bt1" "Bt2" "Cr" "R" "not-used"
> sort(unique(h$hzname))
[1] "2BCt" "2Bt1" "2Bt2" "2Bt3" "2Bt4" "2Bt5" "2CB" "2CBt" "2Cr" "2Crt" "2R" "A" "A1" "A2" "AB" "ABt" "Ad" "Ap" "B" "BA" "BAt" "BC"
[23] "BCt" "Bt" "Bt1" "Bt2" "Bt3" "Bt4" "Bw" "Bw1" "Bw2" "Bw3" "C" "CBt" "Cd" "Cr" "Cr/R" "Crt" "H1" "Oi" "R" "Rt"
> # or



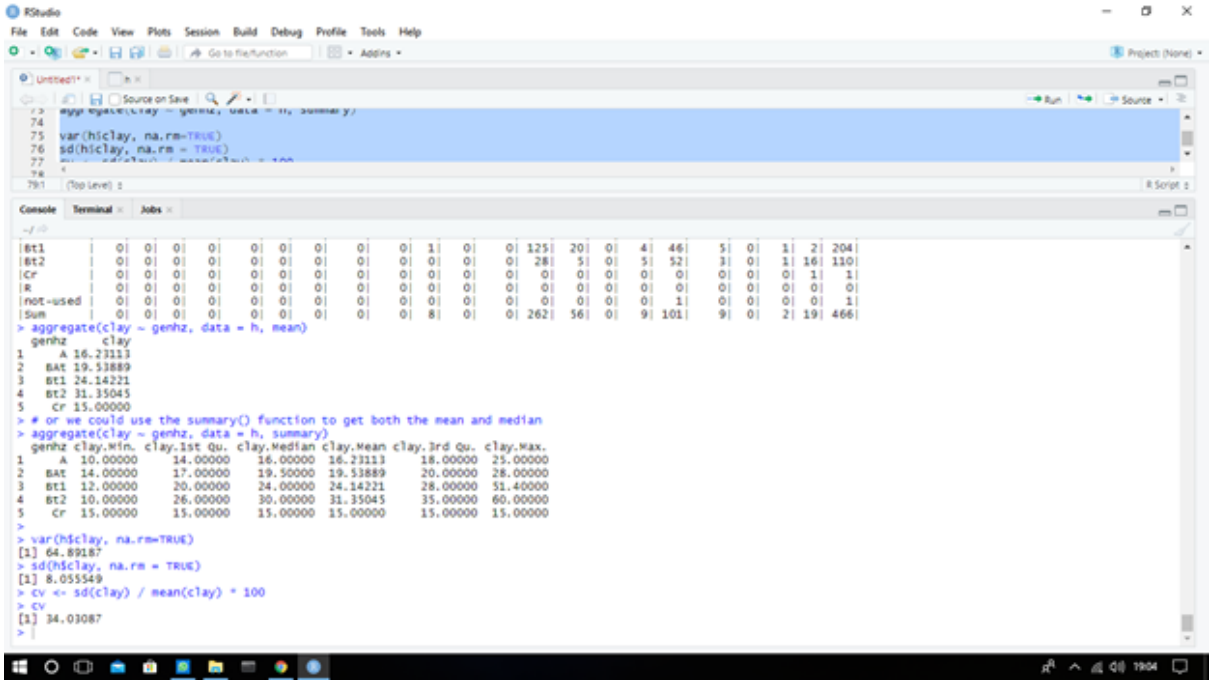
```
# first remove missing values and create a new vector
clay <- na.exclude(h$clay)
mean(clay)

mean(h$clay, na.rm = TRUE)
median(clay)
sort(table(round(h$clay)), decreasing = TRUE)[1] # sort and select the 1st value, which will be the mode
table(h$genhz)
table(h$genhz, h$texcl)
```

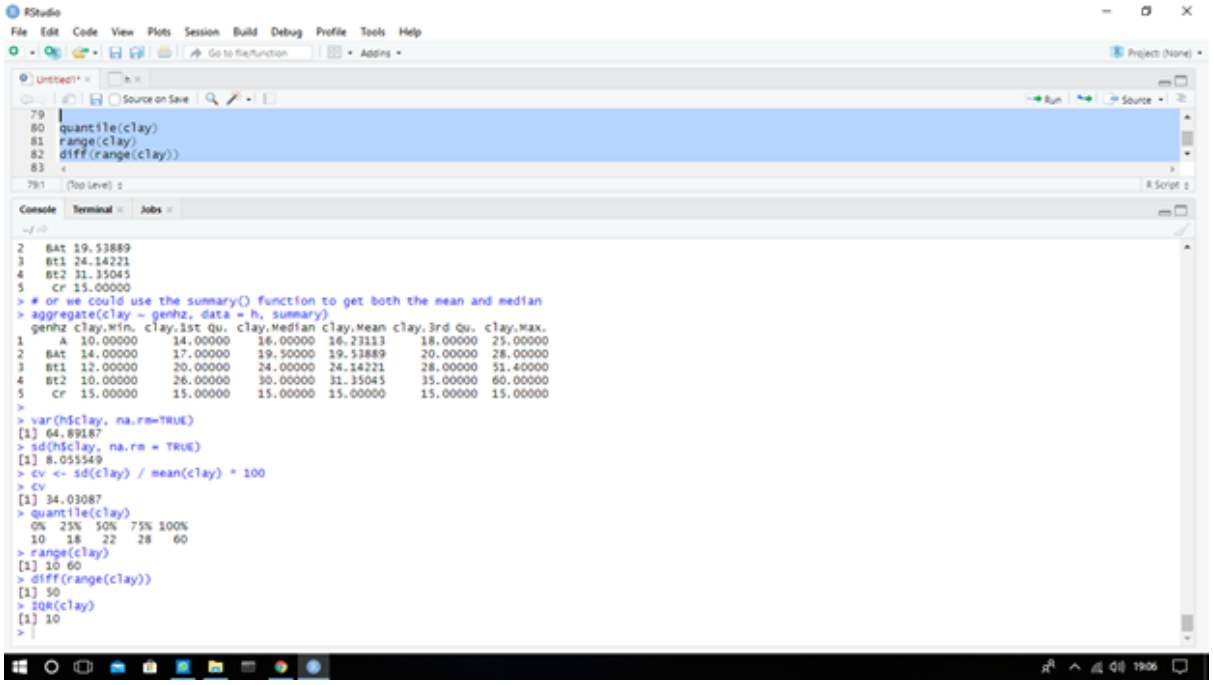
```
> # first remove missing values and create a new vector
> clay <- na.exclude(h$clay)
> mean(clay)
[1] 23.6713
> mean(h$clay, na.rm = TRUE)
[1] 23.6713
> median(clay)
[1] 22
> sort(table(round(h$clay)), decreasing = TRUE)[1] # sort and select the 1st value, which will be the mode
25
41
> table(h$genhz)

       A      BAt      Bt1      Bt2       Cr        R not-used
     113       40      208      116       75       48       26
> table(h$genhz, h$texcl)

          cos   s  fs vfs lcos  ls lfs lvfs cosl  sl fsl vfsl   l sil  si scl  cl sicl  sc sic   c
  A         0   0   0   0    0   0   0    0    0   6   0    0  78  27   0   0   0    0   0   0   0
  BAt       0   0   0   0    0   0   0    0    0   1   0    0  31   4   0   0   2    1   0   0   0
  Bt1       0   0   0   0    0   0   0    0    0   1   0    0 125  20   0   4  46    5   0   1   2
  Bt2       0   0   0   0    0   0   0    0    0   0   0    0  28   5   0   5  52    3   0   1  16
  Cr        0   0   0   0    0   0   0    0    0   0   0    0   0   0   0   0   0    0   0   0   1
  R         0   0   0   0    0   0   0    0    0   0   0    0   0   0   0   0   0    0   0   0   0
  not-used  0   0   0   0    0   0   0    0    0   0   0    0   0   0   0   0   1    0   0   0   0
>
```

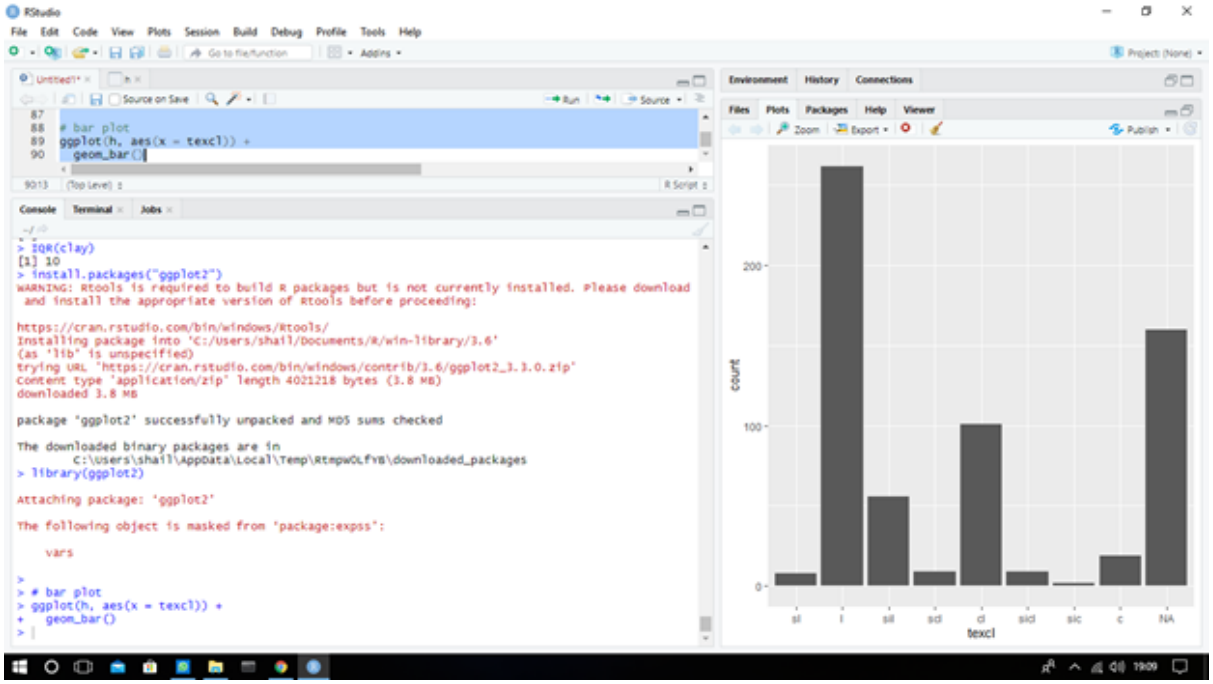


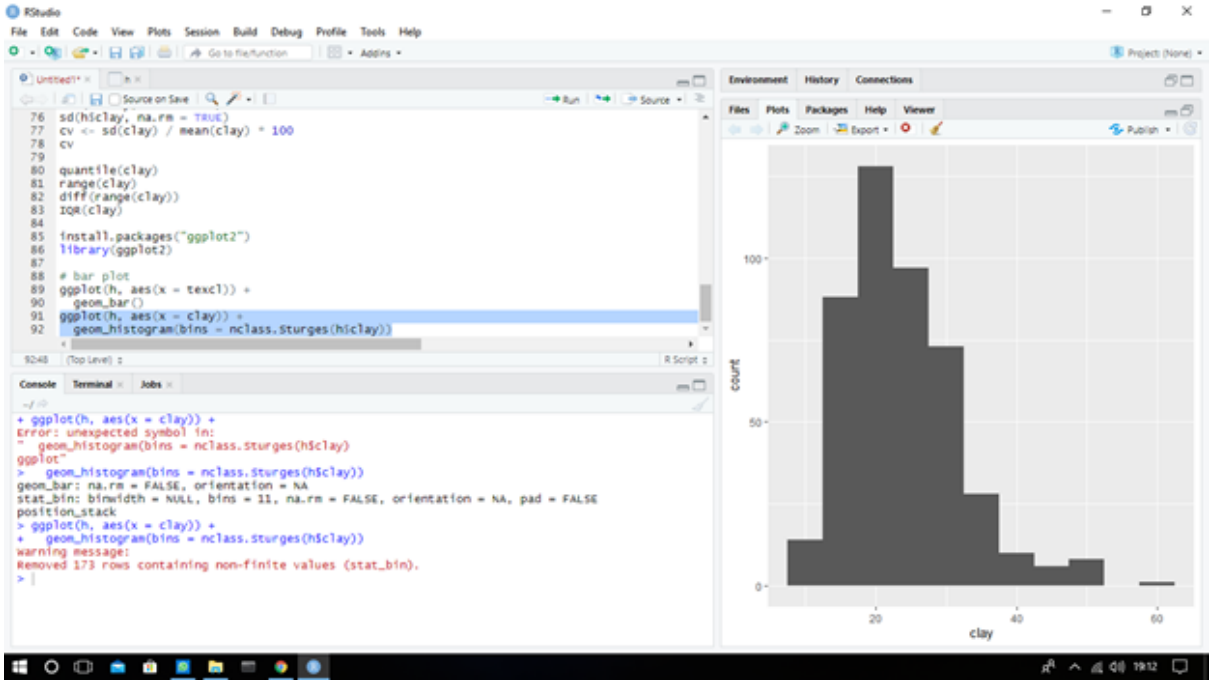
```
# append the table with row and column sums
addmargins(table(h$genhz, h$texcl))

# calculate the proportions relative to the rows, margin = 1 calculates for rows, margin = 2 calculates for columns, margin = NULL calculates for total observ
round(prop.table(table(h$genhz, h$texture_class), margin = 1) * 100)
knitr::kable(addmargins(table(h$genhz, h$texcl)))
```

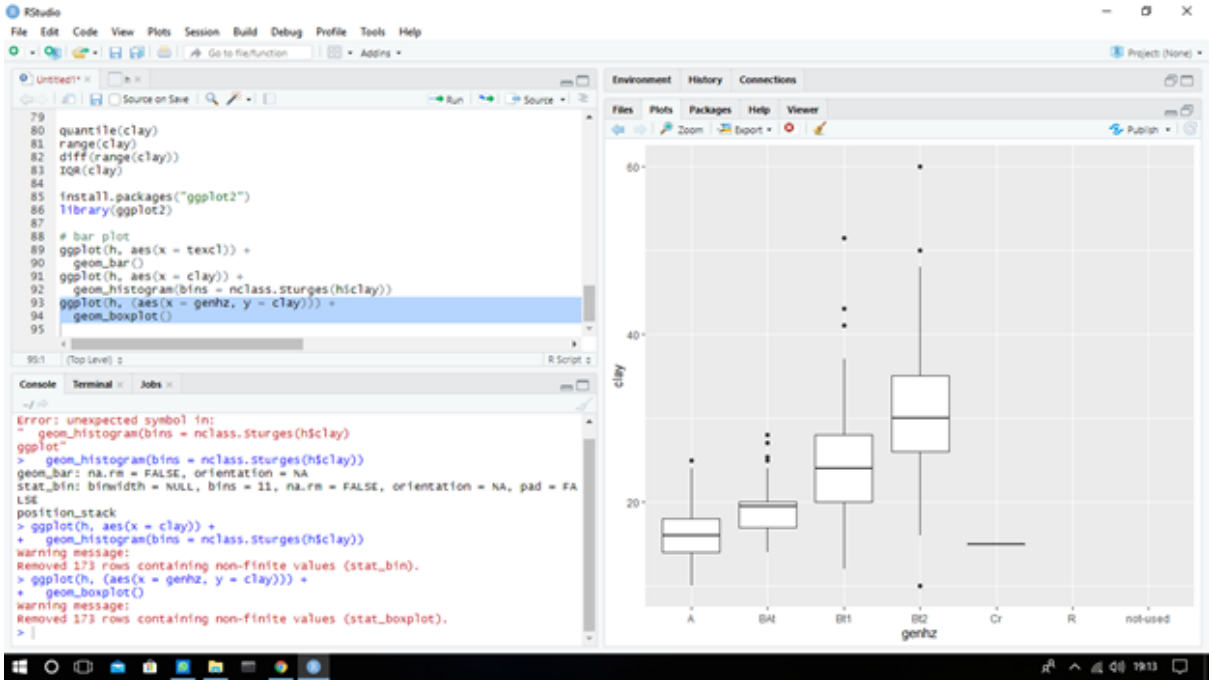
```
> # calculate the proportions relative to the rows, margin = 1 calculates for rows, margin = 2 calculates for columns, margin = NULL calculates for total observatio
ns
> round(prop.table(table(h$genhz, h$texture_class), margin = 1) * 100)

          br   c  cb  cl  gr   l  pg scl sic sicl sil  sl spm
  A        0   0   0   0   0  70   0   0   0    0  24   5   0
  BAt      0   0   0   5   0  79   0   0   0    3  10   3   0
  Bt1      0   1   0  23   0  61   0   2   0    2  10   0   0
  Bt2      0  14   1  46   2  25   1   4   1    3   4   0   0
  Cr      98   2   0   0   0   0   0   0   0    0   0   0   0
  R      100   0   0   0   0   0   0   0   0    0   0   0   0
  not-used  0   0   0   4   0   0   0   0   0    0   0   0  96
> knitr::kable(addmargins(table(h$genhz, h$texcl)))


|          | cos|  s| fs| vfs| lcos| ls| lfs| lvfs| cosl| sl| fsl| vfsl|   l| sil| si| scl|  cl| sicl| sc| sic|  c| Sum|
|:---------|---:|--:|--:|---:|----:|--:|---:|----:|----:|--:|---:|----:|---:|---:|--:|---:|---:|----:|--:|---:|--:|---:|
|A         |   0|  0|  0|   0|    0|  0|   0|    0|    0|  6|   0|    0|  78|  27|  0|   0|   0|    0|  0|   0|  0| 111|
|BAt       |   0|  0|  0|   0|    0|  0|   0|    0|    0|  1|   0|    0|  31|   4|  0|   0|   2|    1|  0|   0|  0|  39|
|Bt1       |   0|  0|  0|   0|    0|  0|   0|    0|    0|  1|   0|    0| 125|  20|  0|   4|  46|    5|  0|   1|  2| 204|
|Bt2       |   0|  0|  0|   0|    0|  0|   0|    0|    0|  0|   0|    0|  28|   5|  0|   5|  52|    3|  0|   1| 16| 110|
|Cr        |   0|  0|  0|   0|    0|  0|   0|    0|    0|  0|   0|    0|   0|   0|  0|   0|   0|    0|  0|   0|  1|   1|
|R         |   0|  0|  0|   0|    0|  0|   0|    0|    0|  0|   0|    0|   0|   0|  0|   0|   0|    0|  0|   0|  0|   0|
|not-used  |   0|  0|  0|   0|    0|  0|   0|    0|    0|  0|   0|    0|   0|   0|  0|   0|   1|    0|  0|   0|  0|   1|
|Sum       |   0|  0|  0|   0|    0|  0|   0|    0|    0|  8|   0|    0| 262|  56|  0|   9| 101|    9|  0|   2| 19| 466|
>
```

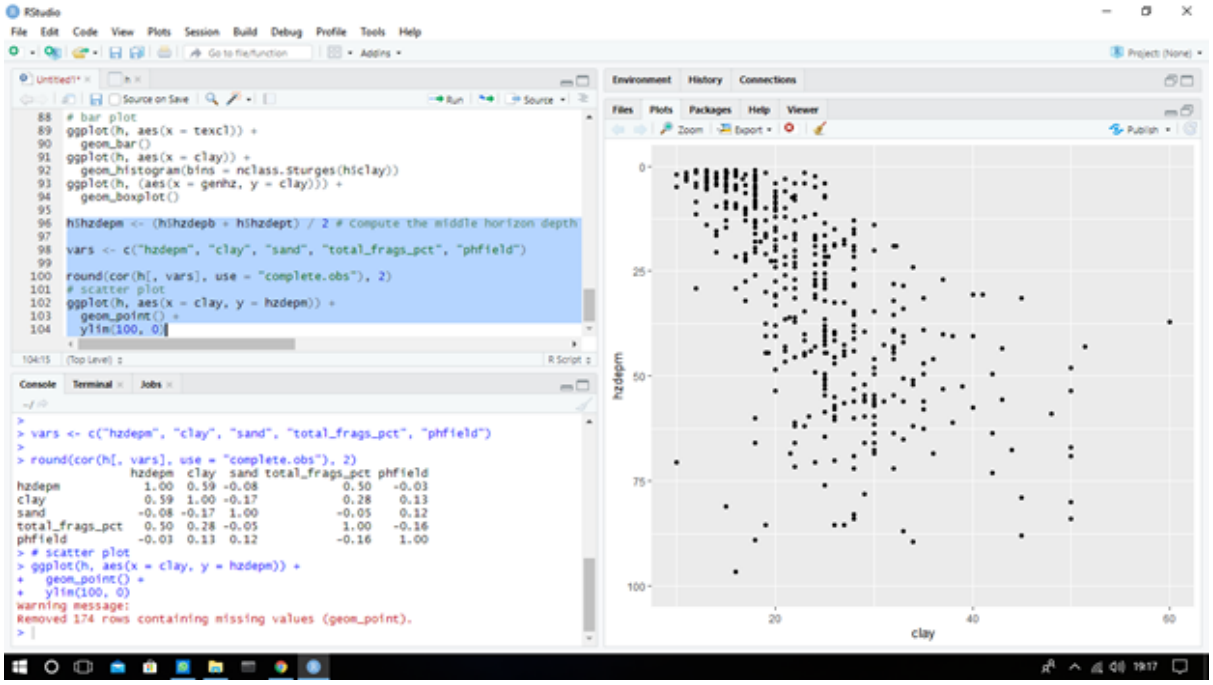

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
var(h$clay, na.rm=TRUE)
sd(h$clay, na.rm = TRUE)
Bt1	0	0	0	0	0	0	0	0	0	1	0	0	125	20	0	4	46	5	0	1	2	204
Bt2	0	0	0	0	0	0	0	0	0	0	0	0	28	5	0	5	52	3	0	1	16	110
Cr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
not-used	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
Sum	0	0	0	0	0	0	0	0	8	0	0	262	56	0	9	101	9	0	2	19	466
> aggregate(clay ~ genhz, data = h, mean)
genhz clay
1 A 16.23113
2 BAt 19.53889
3 Bt1 24.14221
4 Bt2 31.35045
5 Cr 15.00000
> # or we could use the summary() function to get both the mean and median
> aggregate(clay ~ genhz, data = h, summary)
genhz clay.Min. clay.1st Qu. clay.Median clay.Mean clay.3rd Qu. clay.Max.
1 A 10.00000 14.00000 16.00000 16.23113 18.00000 25.00000
2 BAt 14.00000 17.00000 19.50000 19.53889 20.00000 28.00000
3 Bt1 12.00000 20.00000 24.00000 24.14221 28.00000 51.40000
4 Bt2 10.00000 26.00000 30.00000 31.35045 35.00000 60.00000
5 Cr 15.00000 15.00000 15.00000 15.00000 15.00000 15.00000
> var(h$clay, na.rm=TRUE)
[1] 64.89187
> sd(h$clay, na.rm = TRUE)
[1] 8.055549
> cv <- sd(clay) / mean(clay) * 100
> cv
[1] 34.03087


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
quantile(clay)
range(clay)
diff(range(clay))
2 BAt 19.53889
3 Bt1 24.14221
4 Bt2 31.35045
5 Cr 15.00000
> # or we could use the summary() function to get both the mean and median
> aggregate(clay ~ genhz, data = h, summary)
genhz clay.Min. clay.1st Qu. clay.Median clay.Mean clay.3rd Qu. clay.Max.
1 A 10.00000 14.00000 16.00000 16.23113 18.00000 25.00000
2 BAt 14.00000 17.00000 19.50000 19.53889 20.00000 28.00000
3 Bt1 12.00000 20.00000 24.00000 24.14221 28.00000 51.40000
4 Bt2 10.00000 26.00000 30.00000 31.35045 35.00000 60.00000
5 Cr 15.00000 15.00000 15.00000 15.00000 15.00000 15.00000
> var(h$clay, na.rm=TRUE)
[1] 64.89187
> sd(h$clay, na.rm = TRUE)
[1] 8.055549
> cv <- sd(clay) / mean(clay) * 100
> cv
[1] 34.03087
> quantile(clay)
0% 25% 50% 75% 100%
10 18 22 28 60
> range(clay)
[1] 10 60
> diff(range(clay))
[1] 50
> IQR(clay)
[1] 10


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
87
88 # bar plot
89 ggplot(h, aes(x = texcl)) +
90 geom_bar()
Console Terminal Jobs
> IQR(clay)
[1] 10
> install.packages("ggplot2")
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/shail/Documents/R/win-library/3.6'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/ggplot2_3.3.0.zip'
Content type 'application/zip' length 4021218 bytes (3.8 MB)
downloaded 3.8 MB
package 'ggplot2' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\shail\AppData\Local\Temp\RtmpWOLfYB\downloaded_packages
> library(ggplot2)
Attaching package: 'ggplot2'
The following object is masked from 'package:expss':
vars
> # bar plot
> ggplot(h, aes(x = texcl)) +
+ geom_bar()
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
count
200
100
0
sl l sil scl cl sicl sic c NA
texcl


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
76 sd(h$clay, na.rm = TRUE)
77 cv <- sd(clay) / mean(clay) * 100
78 cv
79
80 quantile(clay)
81 range(clay)
82 diff(range(clay))
83 IQR(clay)
84
85 install.packages("ggplot2")
86 library(ggplot2)
87
88 # bar plot
89 ggplot(h, aes(x = texcl)) +
90 geom_bar()
91 ggplot(h, aes(x = clay)) +
92 geom_histogram(bins = nclass.Sturges(h$clay))
Console Terminal Jobs
+ ggplot(h, aes(x = clay)) +
Error: unexpected symbol in:
" geom_histogram(bins = nclass.Sturges(h$clay)
ggplot"
> geom_histogram(bins = nclass.Sturges(h$clay))
geom_bar: na.rm = FALSE, orientation = NA
stat_bin: binwidth = NULL, bins = 11, na.rm = FALSE, orientation = NA, pad = FALSE
position_stack
> ggplot(h, aes(x = clay)) +
+ geom_histogram(bins = nclass.Sturges(h$clay))
Warning message:
Removed 173 rows containing non-finite values (stat_bin).
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
count
100
50
0
20 40 60
clay





Conclusion: Thus, we have studied EDA in R.

Experiment No. 8

Aim : Basic and Advanced graphics in R

Theory :

One of the main reasons data analysts turn to R is for its strong graphic capabilities. These include density plots (histograms and kernel density plots), dot plots, bar charts (simple, stacked, grouped), line charts, pie charts (simple, annotated, 3D), box plots (simple, notched, violin plots, bagplots) and Scatter Plots (simple, with fit lines, scatterplot matrices, high density plots, and 3D plots).

The R base function plot() can be used to create graphs. Some of them are :

- A scatter plot can be created using the function plot(x, y). The function lm() will be used to fit linear models between y and x. A regression line will be added on the plot using the function abline(), which takes the output of lm() as an argument. You can also add a smoothing line using the function loess().
- Boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them. Boxplots are created in R by using the boxplot() function.
- A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to a bar chart but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range. R creates a histogram using hist() function. This function takes a vector as an input and uses some more parameters to plot histograms.
- A line chart is a graph that connects a series of points by drawing line segments between them. These points are ordered in one of their coordinate (usually the x-coordinate) values. Line charts are usually used in identifying the trends in data.The plot() function in R is used to create the line graph.
- A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function barplot() to create bar charts. R can draw both vertical and Horizontal bars in the bar chart. In the bar chart each of the bars can be given different colors.
- A pie-chart is a representation of values as slices of a circle with different colors. The slices are labeled and the numbers corresponding to each slice is also represented in the chart. In R the pie chart is created using the pie() function which takes positive numbers as a vector input.

Output :

Code :

```
City = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/City77.csv")
# Copy city names to row names, i.e. observation labels
row.names(City) = City$City
# Remove city name column from the data frame
City = City[,-1]
names(City)

hist(City$medinc,prob=T,nclass=20,col="pink",main="Histogram of Median
Income",xlab="Median Income")
lines(density(City$medinc),lty=2,col="red",lwd=2)

require(ggplot2)

a = ggplot(City,aes(medinc))
a + geom_histogram(binwidth=1000,aes(y=..density..),fill="pink",color="black") +
  geom_density(linetype=2) +
  xlab("Median Income") +
  ggtitle("Median Income for Top 77 Cities")

plot(City$poverty,City$infmort,xlab="Percent Below Poverty Level",ylab="Infant Mortality
Rate")
lines(lowess(City$poverty,City$infmort),lty=2,col="pink",lwd=2)
abline(lm(infmort~poverty,data=City),lwd=2)
# identify(City$poverty,City$infmort,labels=row.names(City))
title(main="Infant Morality vs. Poverty Level")

a = ggplot(City,aes(poverty,infmort))
a + geom_point() + geom_smooth() +
  geom_text(aes(label=row.names(City)),size=2) +
  xlab("Poverty") + ylab("Infant Mortality") +
  ggtitle("Infant Mortality vs. Poverty Rate")

Olives = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/Olives.txt")
names(Olives)

par(mfrow=c(3,2)) # sets a plotting region for with 3 rows and 2 columns
boxplot(split(Olives$Oleic,Olives$Area.Name),xlab="Area",ylab="Oleic Acid")
boxplot(split(Olives$Linoleic,Olives$Area.Name),xlab="Area",ylab="Linoleic Acid")
boxplot(split(Olives$Strearic,Olives$Area.Name),xlab="Area",ylab="Strearic Acid")
boxplot(split(Olives$Linolenic,Olives$Area.Name),xlab="Area",ylab="Linolenic Acid")
boxplot(split(Olives$Eicosanoic,Olives$Area.Name),xlab="Area",ylab="Eicosanoic Acid")
boxplot(split(Olives$Eicosenoic,Olives$Area.Name),xlab="Area",ylab="Eicosenoic Acid")

require(s20x)

boxqq(Oleic~Region.Name,data=Olives)

a = ggplot(Olives,aes(Area.Name,Oleic))
a + geom_boxplot(fill="lightblue") + xlab("Area Name") +
  ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
```

```
require(violinmplot)

violinmplot(Area.Name~Oleic,xlab="Oleic Acid",ylab="Area
Name",data=Olives,horizontal=T)

a = ggplot(Olives,aes(Area.Name,Oleic))
a + geom_violin(fill="lightblue") + xlab("Area Name") +
  ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")

data("UCBAdmissions")
UCBAdmissions

temp = data.frame(UCBAdmissions)
temp


DeptCount = margin.table(UCBAdmissions,3)
DeptCount

par(mfrow=c(1,2))
pie(DeptCount)
barplot(DeptCount,xlab="Department",ylab="Frequency")

par(mfrow=c(1,2))
DeptAdmit = margin.table(UCBAdmissions,c(1,3))
DeptAdmit

barplot(DeptAdmit,xlab="Department",ylab="Frequency")
barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)

GenderAdmit = margin.table(UCBAdmissions,c(1,2))
GenderAdmit

pie(GenderAdmit[,1],main="Males")
pie(GenderAdmit[,2],main="Females")

barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)

par(mfrow=c(1,2))
mosaicplot(~Gender+Admit,data=UCBAdmissions,color=T)
mosaicplot(~Admit+Gender,data=UCBAdmissions,color=T)

par(mfrow=c(1,2))
mosaicplot(~Dept+Admit,data=UCBAdmissions,color=T)
mosaicplot(~Admit+Dept,data=UCBAdmissions,color=T)


mosaicplot(~Dept+Gender+Admit,data=UCBAdmissions,color=T)
```

```
mosaicplot(~Dept+Admit+Gender,data=UCBAdmissions,color=T)

mosaicplot(~Gender+Dept+Admit,data=UCBAdmissions,color=T)

mosaicplot(~Admit+Dept+Gender,data=UCBAdmissions,color=T)

names(Olives)

olive.mat = Olives[,c(7,6,5,3)]
pairs(olive.mat)

panel.cor = function (x, y, digits = 2, prefix = "", cex.cor)
{
  usr <- par("usr")
  on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste(prefix, txt, sep = "")
  if (missing(cex.cor))
    cex <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex * r)
}

panel.trendsd = function (x, y, f = 0.5)
{
  par(err = -1)
  xs <- sort(x, index = T)
  x <- xs$x
  ix <- xs$ix
  y <- y[ix]
  trend <- lowess(x, y, f)
  e2 <- (y - trend$y)^2
  scatter <- lowess(x, e2)
  uplim <- trend$y + sqrt(abs(scatter$y))
  lowlim <- trend$y - sqrt(abs(scatter$y))
  points(x, y, pch = 1)
  lines(trend, col = "Blue")
  lines(scatter$x, uplim, lty = 2, col = "Red")
  lines(scatter$x, lowlim, lty = 2, col = "Red")
  invisible()
}

panel.hist = function (x, ...)
{
  usr <- par("usr")
  on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5))
  h <- hist(x, plot = FALSE)
```

```
  breaks <- h$breaks
  nB <- length(breaks)
  y <- h$counts
  y <- y/max(y)
  rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
}

pairs.trendsd = function(data,...) {
  pairs(data,lower.panel=panel.cor,upper.panel=panel.trendsd,
        diag.panel=panel.hist,...)}


require(lattice)
pairs.trendsd(olive.mat)

Boston = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/Boston.txt")
names(Boston)

require(corrgram)

corrgram(Boston[,-c(1:4)],lower.panel=panel.pts,upper.panel=panel.pie)

corrgram(Boston[,-c(1:4)],lower.panel=panel.shade,upper.panel=panel.pie)

require(corrplot)

Boston.corr = cor(Boston[,-c(1:4)])   # find pairwise correlations between all variables.
options(digits=2)  # reduce the number of significant digits shown.
Boston.corr

corrplot(Boston.corr)

corrplot(Boston.corr,method="ellipse")

corrplot(Boston.corr,order="FPC")

corrplot(Boston.corr,order="hclust")

data(iris)
names(iris)

library(lattice)
xyplot(Sepal.Length~Petal.Length|Species,data=iris)

xyplot(Sepal.Length~Petal.Length|Species,layout=c(2,2),data=iris)

SepWid = equal.count(iris$Sepal.Width)
plot(SepWid)

print(SepWid)
```

```
xyplot(Sepal.Length~Petal.Length|SepWid,data=iris)

xyplot(Sepal.Length~Petal.Length|SepWid,groups=Species,layout=c(3,2),
      auto.key=list(columns=3),main="Sepal Length * Petal Length | Sepal
 Width",data=iris)


splom(~iris[,1:4],groups=Species,auto.key=list(columns=3),data=iris)

bwplot(Species~Sepal.Width,data=iris,xlab="Sepal Width (mm)",main="Boxplots of Iris
Sepal Width")

stripplot(Species ~ jitter(Sepal.Width), data = iris, xlab="Sepal Width (mm)",main="Sepal
Width Across Species")

stripplot(Species ~ jitter(Sepal.Width), data = iris, aspect = 1,
        jitter=T,xlab="Sepal Width (mm)",main="Sepal Width Across Species")

coplot(Sepal.Width~Sepal.Length|Species,data=iris)

coplot(Sepal.Width~Sepal.Length|Petal.Width,number=4,overlap=.2,data=iris)

coplot(Sepal.Width~Sepal.Length|Petal.Width*Petal.Length,number=c(3,3),
      overlap=.5,col=as.numeric(iris$Species),pch=as.numeric(iris$Species)+1,data=iris)

Ozdata = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/Ozdata.csv")
names(Ozdata)

coplot(upoz~safb|inbh*v500,number=c(4,4),panel=panel.smooth,data=Ozdata)

coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=function(x,y,...)
panel.smooth(x,y,span=.6,...),data=Ozdata)


library(ash)
Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/Swiss.csv")
names(Swiss)

d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of Bill Diagonal (mm)")
lines(d1)

diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab="",cex=.5,theta=-
45,phi=30,shade=1,col="pink")


image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
```

```
contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add=T)
points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),cex=.4)

Genuine <- Swiss$genu
Genuine[Genuine==0] <- "Forged"
Genuine[Genuine==1] <- "Real"
xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)

splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))

pairs.image <- function(x) {
  pairs(x,panel=function(x,y) {
    foo <- bin2(cbind(x,y),nbin=c(75,75))
    foo <- ash2(foo,m=c(6,6))
    image(foo,add=T,xlab="",ylab="",col=topo.colors(1000))
    points(x,y,pch=".")
  })
}
pairs.image(Swiss[,1:6])

pairs.persp <- function(x) {
  par(bg="sky blue")
  pairs(x,panel=function(x,y) {
    foo <- bin2(cbind(x,y),nbin=c(75,75))
    foo <- ash2(foo,m=c(8,8))
    par(new=T)
    persp(foo,xlab="",ylab="",theta=-45,phi=35,col="red",
          shade=.75,box=F,scale=F,border=NA,expand=.9)
  })
  par(new=F,bg="white")
}
pairs.persp(Swiss[,1:3])

NHL = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/NHL.csv")
names(NHL)

plot(NHL$GF,NHL$GA,xlab="Goals For",ylab="Goals Against",main="Plot of GA vs. GF
with Symbols = Wins")
symbols(NHL$GF,NHL$GA,circles=NHL$W,add=T,inches=.25)  # circles are too big if
inches is not specified!
symbols(NHL$GF,NHL$GA,circles=NHL$W,add=T,bg="lightblue",inches=.25)  # bg
option colors circles light blue
text(NHL$GF,NHL$GA,labels=NHL$TEAM,cex=.6)

kodiak.crab = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/kodiak.csv")
survey.crab = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/survey.csv")
attach(kodiak.crab)  # This file contains latitude and longitude of the island coastline.
attach(survey.crab)  # This file contains crab counts by type, year, and location (lat, long)

names(survey.crab)
```

```
table(year)

crab.plot <- function(lat,long,y,year,yname=deparse(substitute(y))){

symbols(long[year==year],lat[year==year],circles=y[year==year],inches=.1,bg="blue",xlab=
"Longitude",ylab="Latitude",
        main=paste("Circles Proportional to",yname))
  title(sub= paste("Year =",year))
  points(kodiak.crab$longit,kodiak.crab$latit,pch=".")
}
nr.pp = nrec/npots
crab.plot(lat,long,y=nr.pp,year=83,yname="Number of Recruits per Pot")

detach(survey.crab)
detach(kodiak.crab)  # if you attach, be sure to detach when finished!

attach(City)        # This command readies the data frame City for analysis.

names(City)          # The names command lists the names of the variables in a data set.

View(City)

city.mat <-cbind(popdens,pctblack,pcthisp,medinc,pct.pa,poverty,infmort,unempl)
stars(city.mat, key.loc = c(15,1.25),main = "Starplots for U.S.
Cities",label=row.names(City),cex=.7)

palette(rainbow(12, s = 0.6, v = 0.75))
stars(city.mat,len=.80,key.loc = c(15,1.25),main = "Starplots for U.S. Cities",draw.segments
= TRUE,label=row.names(City),cex=.7)

palette("default")

library(MASS)

parcoord(city.mat)
names(Olives)
olive.mat = Olives[,3:9]
palette(rainbow(12,s=0.6,v=0.75))
parcoord(olive.mat,col=as.numeric(Olives$Area.Name))
```
Screenshots :

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

graphicsnr.R

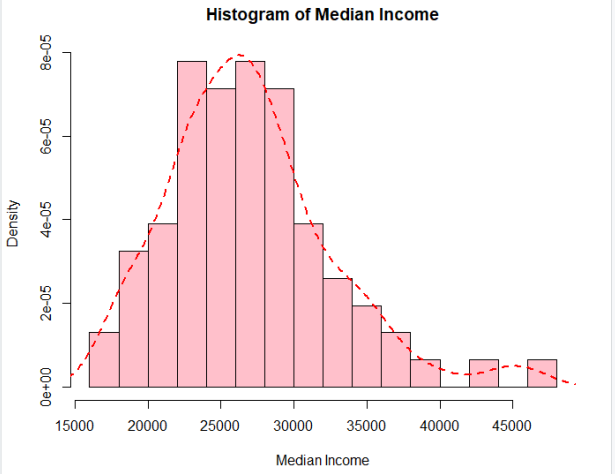
```
City = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/City7
# Copy city names to row names, i.e. observation labels
row.names(City) = City$City
# Remove city name column from the data frame
City = City[,-1]
names(City)

hist(City$medinc,prob=T,nclass=20,col="pink",main="Histogram of Median
lines(density(City$medinc),lty=2,col="red",lwd=2)

require(ggplot2)

a = ggplot(City,aes(medinc))
a + geom_histogram(binwidth=1000,aes(y=..density..),fill="pink",color="
  geom_density(linetype=2) +
  xlab("Median Income") +
  ggtitle("Median Income for Top 77 Cities")

plot(City$poverty,City$infmort,xlab="Percent Below Poverty Level",ylab=
lines(lowess(City$poverty,City$infmort),lty=2,col="pink",lwd=2)
abline(lm(infmort~poverty,data=City),lwd=2)
# identify(City$poverty,City$infmort,labels=row.names(City))
title(main="Infant Morality vs. Poverty Level")

a = ggplot(City,aes(poverty,infmort))
a + geom_point() + geom_smooth() +
  geom_text(aes(label=row.names(City)),size=2) +
  xlab("Poverty") + ylab("Infant Mortality") +
  ggtitle("Infant Mortality vs. Poverty Rate")
```

Console Terminal Jobs

```
> hist(City$medinc,prob=T,nclass=20,col="pink",main="Histogram of Median Inco
me",xlab="Median Income")
> lines(density(City$medinc),lty=2,col="red",lwd=2)
>
```

Environment History Connections

Files Plots Packages Help Viewer



14:39

RStudio

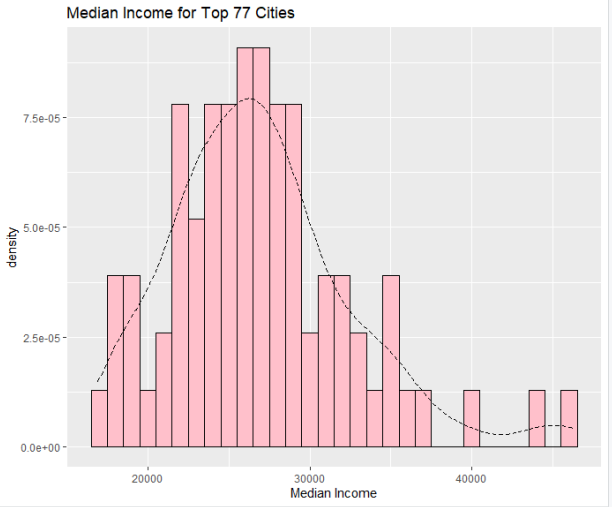
File Edit Code View Plots Session Build Debug Profile Tools Help

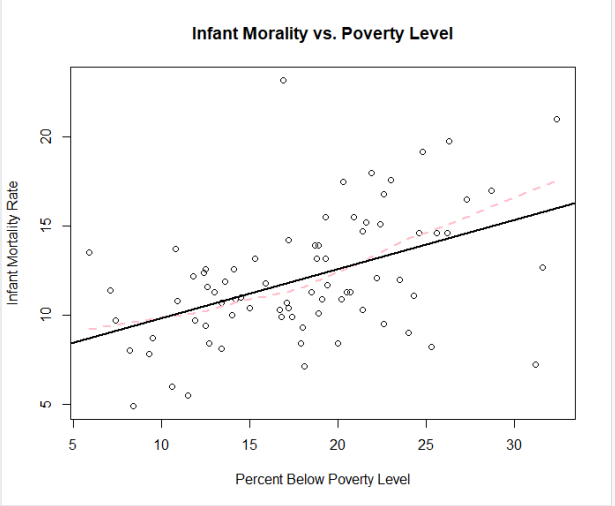
graphicsinr.R

```
City = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/City7
# Copy city names to row names, i.e. observation labels
row.names(City) = City$City
# Remove city name column from the data frame
City = City[,-1]
names(City)

hist(City$medinc,prob=T,nclass=20,col="pink",main="Histogram of Median
lines(density(City$medinc),lty=2,col="red",lwd=2)

require(ggplot2)

a = ggplot(City,aes(medinc))
a + geom_histogram(binwidth=1000,aes(y=..density..),fill="pink",color="
  geom_density(linetype=2) +
  xlab("Median Income") +
  ggtitle("Median Income for Top 77 Cities")

plot(City$poverty,City$infmort,xlab="Percent Below Poverty Level",ylab=
lines(lowess(City$poverty,City$infmort),lty=2,col="pink",lwd=2)
abline(lm(infmort~poverty,data=City),lwd=2)
# identify(City$poverty,City$infmort,labels=row.names(City))
title(main="Infant Morality vs. Poverty Level")

a = ggplot(City,aes(poverty,infmort))
a + geom_point() + geom_smooth() +
  geom_text(aes(label=row.names(City)),size=2) +
  xlab("Poverty") + ylab("Infant Mortality") +
  ggtitle("Infant Mortality vs. Poverty Rate")
```

Console Terminal Jobs

```
> abline(lm(infmort~poverty,data=City),lwd=2)
> # identify(City$poverty,City$infmort,labels=row.names(City))
> title(main="Infant Morality vs. Poverty Level")
>
```





RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

graphicsinr.R

```
23 title(main="Infant Morality vs. Poverty Level")
24
25 a = ggplot(City,aes(poverty,infmort))
26 a + geom_point() + geom_smooth() +
27   geom_text(aes(label=row.names(City)),size=2) +
28   xlab("Poverty") + ylab("Infant Mortality") +
29   ggtitle("Infant Mortality vs. Poverty Rate")
30
31 olives = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/Oli
32 names(olives)
33
34 par(mfrow=c(3,2)) # sets a plotting region for with 3 rows and 2 column
35 boxplot(split(olives$Oleic,olives$Area.Name),xlab="Area",ylab="Oleic Ac
36 boxplot(split(olives$Linoleic,olives$Area.Name),xlab="Area",ylab="Linol
37 boxplot(split(olives$Strearic,olives$Area.Name),xlab="Area",ylab="Strea
38 boxplot(split(olives$Linolenic,olives$Area.Name),xlab="Area",ylab="Lino
39 boxplot(split(olives$Eicosanoic,olives$Area.Name),xlab="Area",ylab="Eic
40 boxplot(split(olives$Eicosenoic,olives$Area.Name),xlab="Area",ylab="Eic
41
42 require(s20x)
43
44 boxqq(Oleic~Region.Name,data=Olives)
45
46 a = ggplot(Olives,aes(Area.Name,Oleic))
47 a + geom_boxplot(fill="lightblue") + xlab("Area Name") +
48   ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
49
50 require(violinmplot)
51
52 violinmplot(Area.Name~Oleic,xlab="Oleic Acid",ylab="Area Name",data=Oli
53
```

41:1 (Top Level) R Script

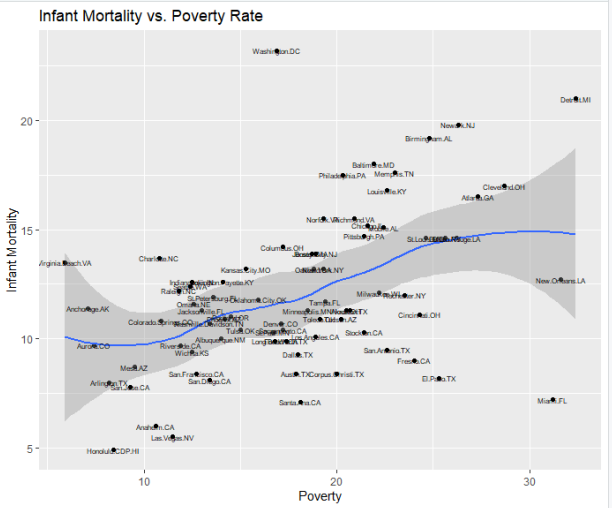
Console Terminal Jobs

```
oic Acid")
> boxplot(split(Olives$Eicosenoic,Olives$Area.Name),xlab="Area",ylab="Eicosen
oic Acid")
>
```

Environment History Connections

Files Plots Packages Help Viewer

Zoom Export Publish



14:42

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

graphicsinr.R

```
23 title(main="Infant Morality vs. Poverty Level")
24
25 a = ggplot(City,aes(poverty,infmort))
26 a + geom_point() + geom_smooth() +
27   geom_text(aes(label=row.names(City)),size=2) +
28   xlab("Poverty") + ylab("Infant Mortality") +
29   ggtitle("Infant Mortality vs. Poverty Rate")
30
31 olives = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/Oli
32 names(olives)
33
34 par(mfrow=c(3,2)) # sets a plotting region for with 3 rows and 2 column
35 boxplot(split(olives$Oleic,olives$Area.Name),xlab="Area",ylab="Oleic Ac
36 boxplot(split(olives$Linoleic,olives$Area.Name),xlab="Area",ylab="Linol
37 boxplot(split(olives$Strearic,olives$Area.Name),xlab="Area",ylab="Strea
38 boxplot(split(olives$Linolenic,olives$Area.Name),xlab="Area",ylab="Lino
39 boxplot(split(olives$Eicosanoic,olives$Area.Name),xlab="Area",ylab="Eic
40 boxplot(split(olives$Eicosenoic,olives$Area.Name),xlab="Area",ylab="Eic
41
42 require(s20x)
43
44 boxqq(Oleic~Region.Name,data=Olives)
45
46 a = ggplot(Olives,aes(Area.Name,Oleic))
47 a + geom_boxplot(fill="lightblue") + xlab("Area Name") +
48   ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
49
50 require(violinmplot)
51
52 violinmplot(Area.Name~Oleic,xlab="Oleic Acid",ylab="Area Name",data=Oli
53
```

41:1 (Top Level) R Script

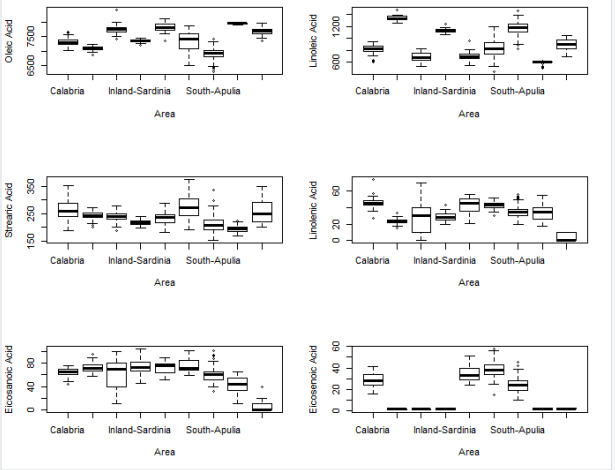
Console Terminal Jobs

```
oic Acid")
> boxplot(split(Olives$Eicosenoic,Olives$Area.Name),xlab="Area",ylab="Eicosen
oic Acid")
>
```

Environment History Connections

Files Plots Packages Help Viewer

Zoom Export Publish



14:43

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

graphicsnr.R

```
23  title(main="Infant Morality vs. Poverty Level")
24
25  a = ggplot(City,aes(poverty,infmort))
26  a + geom_point() + geom_smooth() +
27    geom_text(aes(label=row.names(City)),size=2) +
28    xlab("Poverty") + ylab("Infant Mortality") +
29    ggtitle("Infant Mortality vs. Poverty Rate")
30
31  olives = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/Oli
32  names(olives)
33
34  par(mfrow=c(3,2)) # sets a plotting region for with 3 rows and 2 column
35  boxplot(split(olives$Oleic,olives$Area.Name),xlab="Area",ylab="Oleic Ac
36  boxplot(split(olives$Linoleic,olives$Area.Name),xlab="Area",ylab="Linol
37  boxplot(split(olives$Strearic,olives$Area.Name),xlab="Area",ylab="Strea
38  boxplot(split(olives$Linolenic,olives$Area.Name),xlab="Area",ylab="Lino
39  boxplot(split(olives$Eicosanoic,olives$Area.Name),xlab="Area",ylab="Eic
40  boxplot(split(olives$Eicosenoic,olives$Area.Name),xlab="Area",ylab="Eic
41
42  require(s20x)
43
44  boxqq(Oleic~Region.Name,data=olives)
45
46  a = ggplot(olives,aes(Area.Name,Oleic))
47  a + geom_boxplot(fill="lightblue") + xlab("Area Name") +
48    ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
49
50  require(violinmplot)
51
52  violinmplot(Area.Name~Oleic,xlab="Oleic Acid",ylab="Area Name",data=Oli
53
```

48:62 (Top Level) R Script

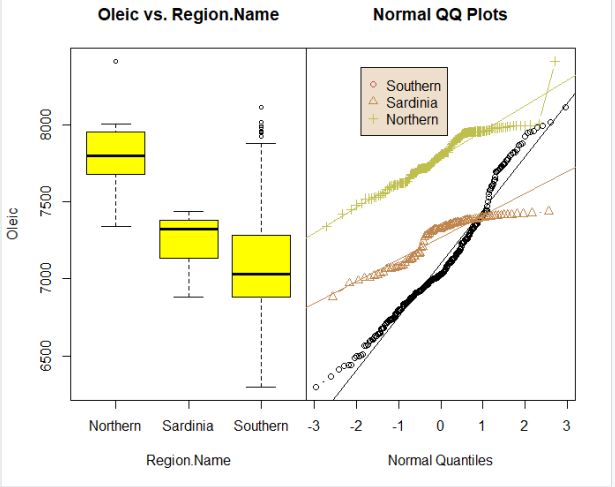
Console Terminal Jobs

```
> a = ggplot(olives,aes(Area.Name,Oleic))
> a + geom_boxplot(fill="lightblue") + xlab("Area Name") +
+   ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
>
```

Environment History Connections

Files Plots Packages Help Viewer

Zoom Export Publish



14:43

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

graphicsnr.R

```
23  title(main="Infant Morality vs. Poverty Level")
24
25  a = ggplot(City,aes(poverty,infmort))
26  a + geom_point() + geom_smooth() +
27    geom_text(aes(label=row.names(City)),size=2) +
28    xlab("Poverty") + ylab("Infant Mortality") +
29    ggtitle("Infant Mortality vs. Poverty Rate")
30
31  olives = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Data/Oli
32  names(olives)
33
34  par(mfrow=c(3,2)) # sets a plotting region for with 3 rows and 2 column
35  boxplot(split(olives$Oleic,olives$Area.Name),xlab="Area",ylab="Oleic Ac
36  boxplot(split(olives$Linoleic,olives$Area.Name),xlab="Area",ylab="Linol
37  boxplot(split(olives$Strearic,olives$Area.Name),xlab="Area",ylab="Strea
38  boxplot(split(olives$Linolenic,olives$Area.Name),xlab="Area",ylab="Lino
39  boxplot(split(olives$Eicosanoic,olives$Area.Name),xlab="Area",ylab="Eic
40  boxplot(split(olives$Eicosenoic,olives$Area.Name),xlab="Area",ylab="Eic
41
42  require(s20x)
43
44  boxqq(Oleic~Region.Name,data=olives)
45
46  a = ggplot(olives,aes(Area.Name,Oleic))
47  a + geom_boxplot(fill="lightblue") + xlab("Area Name") +
48    ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
49
50  require(violinmplot)
51
52  violinmplot(Area.Name~Oleic,xlab="Oleic Acid",ylab="Area Name",data=Oli
53
```

48:62 (Top Level) R Script

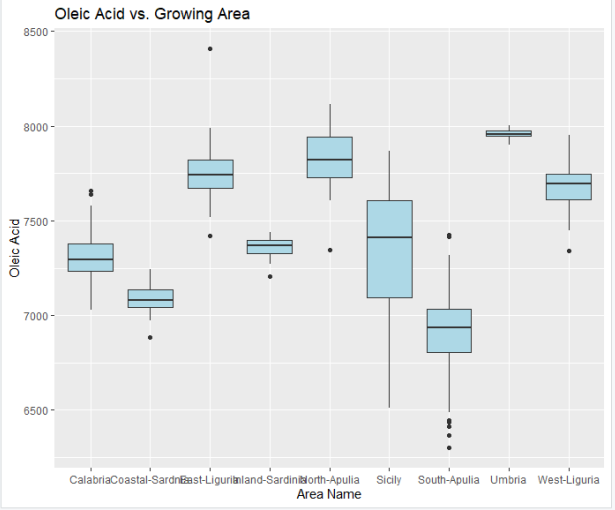
Console Terminal Jobs

```
> a = ggplot(olives,aes(Area.Name,Oleic))
> a + geom_boxplot(fill="lightblue") + xlab("Area Name") +
+   ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
>
```

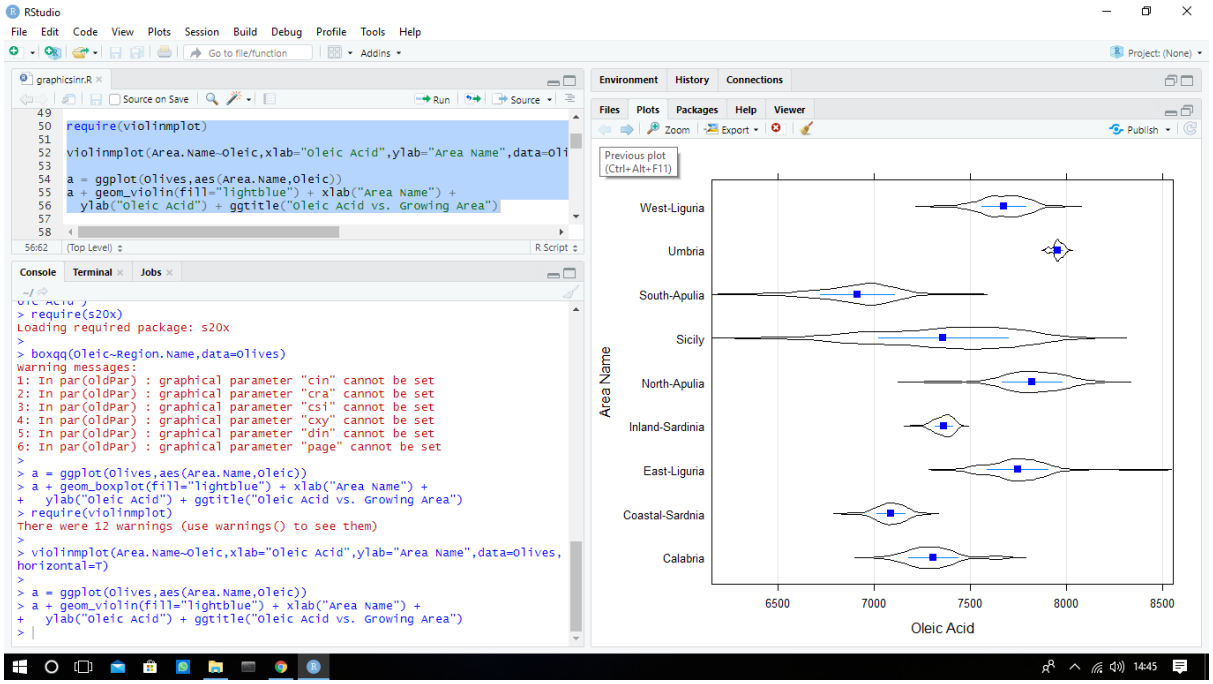
Environment History Connections

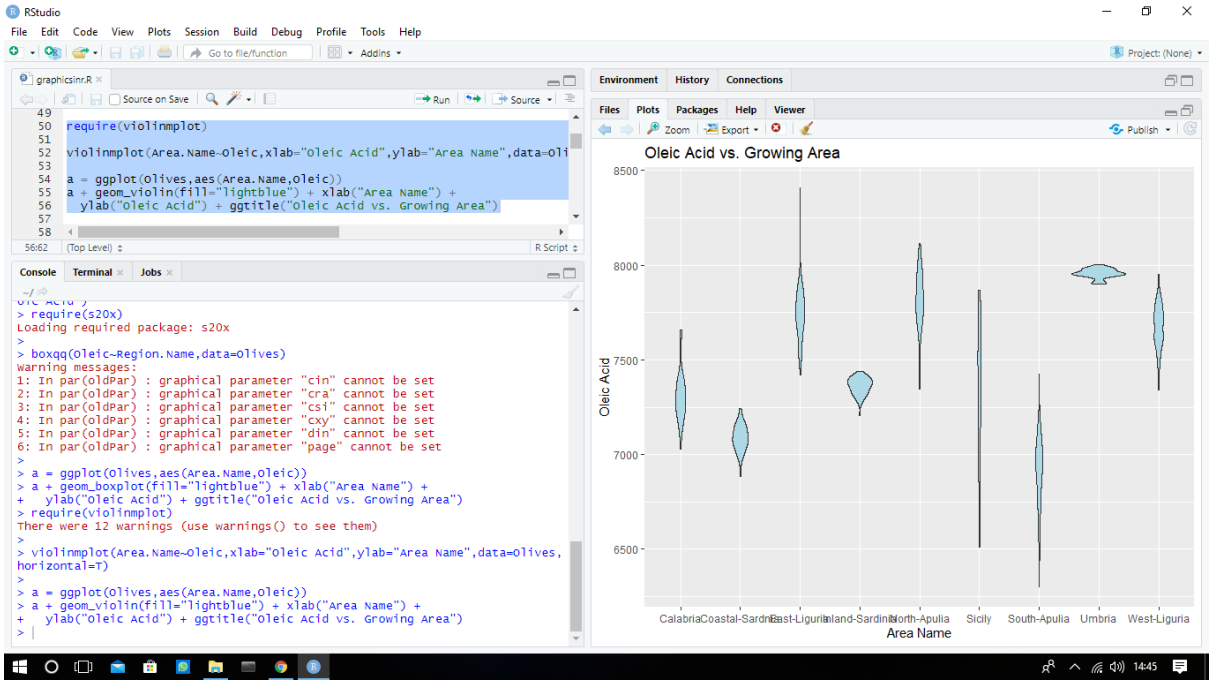
Files Plots Packages Help Viewer

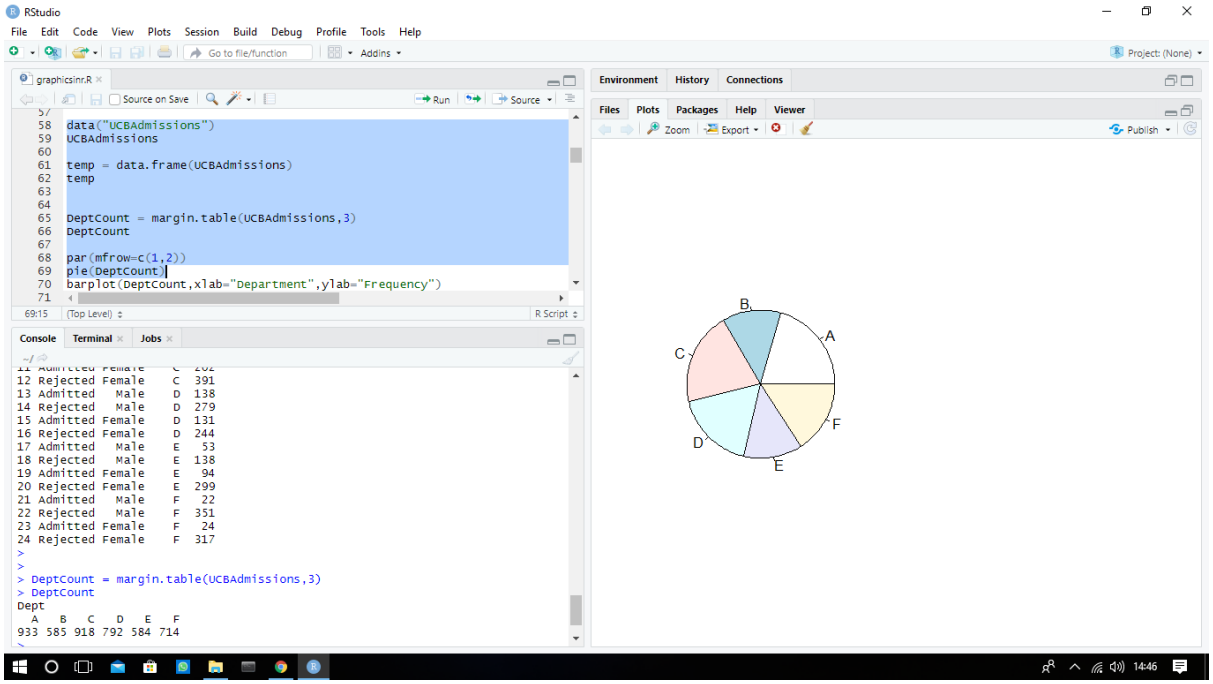
Zoom Export Publish

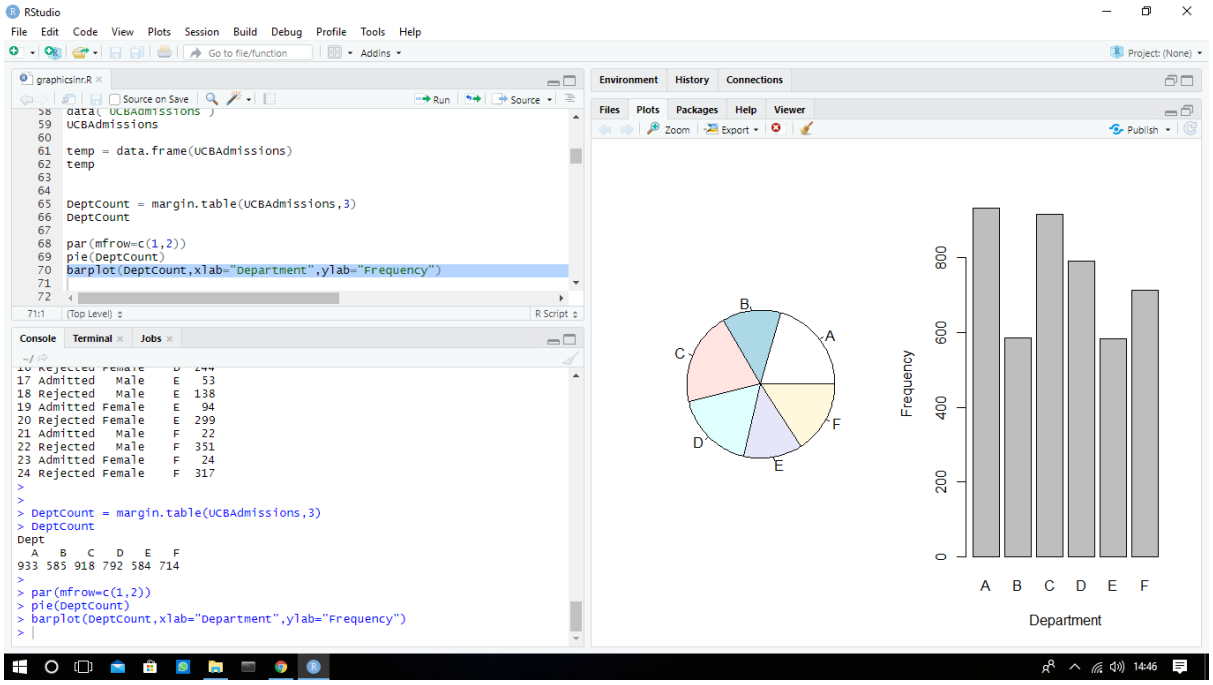


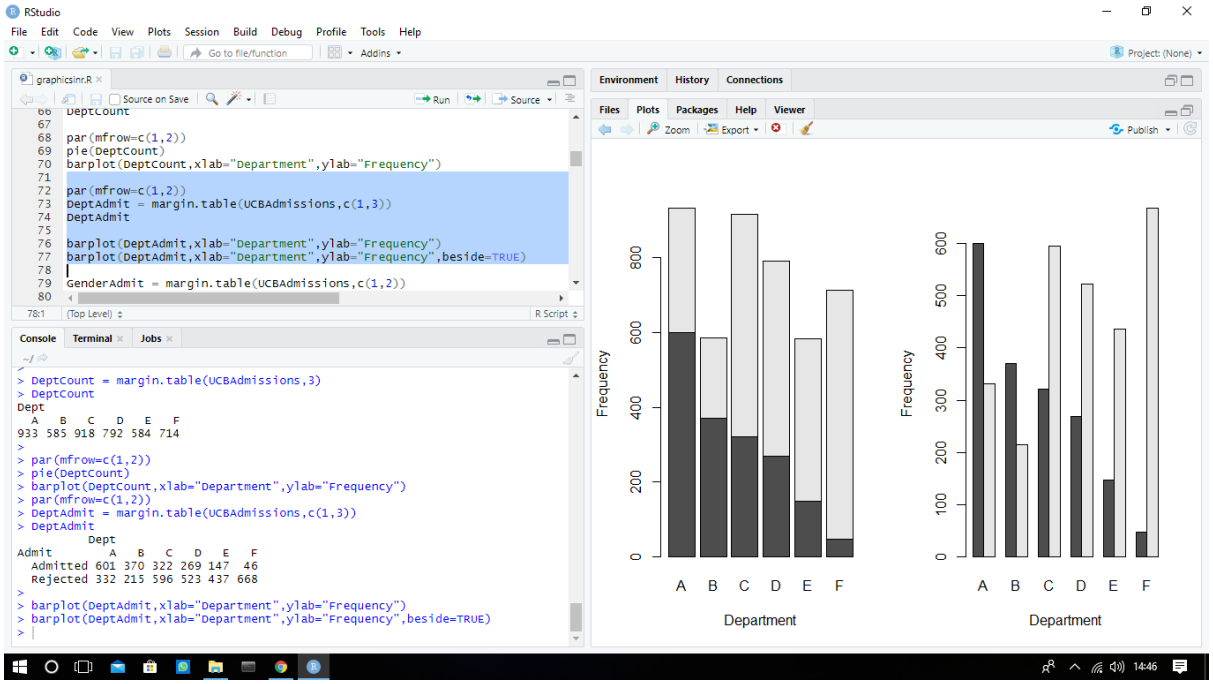
14:43

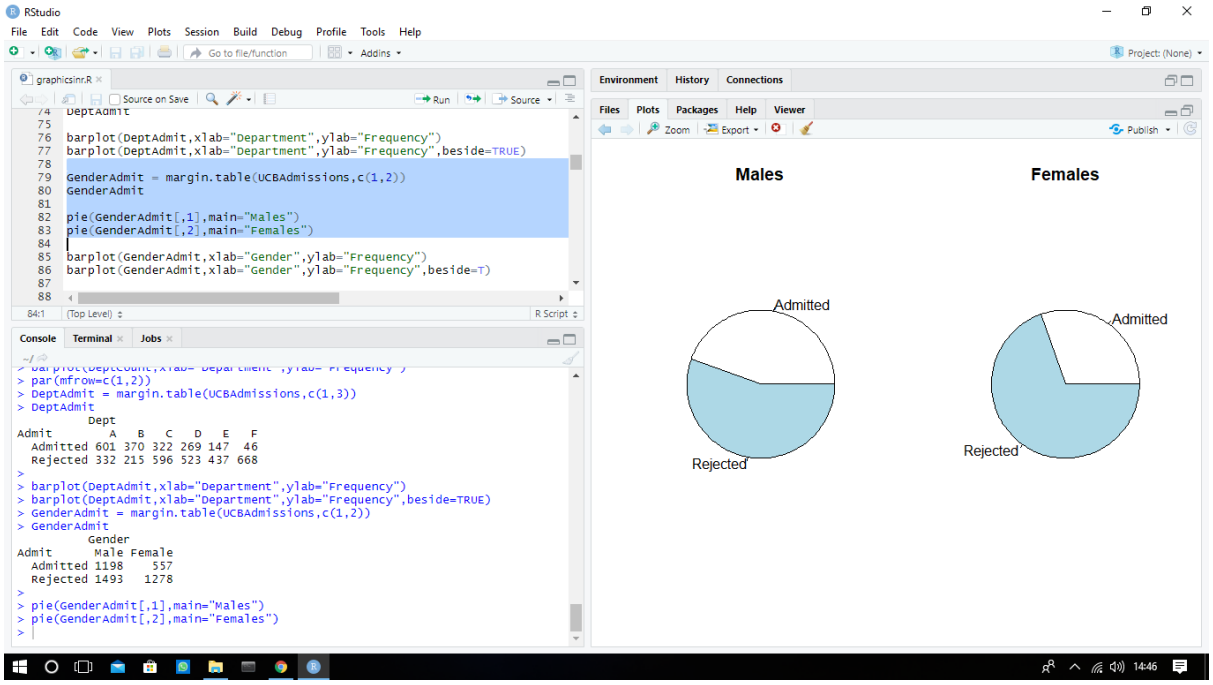

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
require(violinmplot)
violinmplot(Area.Name~Oleic,xlab="Oleic Acid",ylab="Area Name",data=Oli
a = ggplot(Olives,aes(Area.Name,Oleic))
a + geom_violin(fill="lightblue") + xlab("Area Name") +
ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
Console Terminal Jobs
> require(s20x)
Loading required package: s20x
> boxqq(Oleic~Region.Name,data=Olives)
Warning messages:
1: In par(oldPar) : graphical parameter "cin" cannot be set
2: In par(oldPar) : graphical parameter "cra" cannot be set
3: In par(oldPar) : graphical parameter "csi" cannot be set
4: In par(oldPar) : graphical parameter "cxy" cannot be set
5: In par(oldPar) : graphical parameter "din" cannot be set
6: In par(oldPar) : graphical parameter "page" cannot be set
> a = ggplot(Olives,aes(Area.Name,Oleic))
> a + geom_boxplot(fill="lightblue") + xlab("Area Name") +
+ ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
> require(violinmplot)
There were 12 warnings (use warnings() to see them)
> violinmplot(Area.Name~Oleic,xlab="Oleic Acid",ylab="Area Name",data=Olives,
horizontal=T)
> a = ggplot(Olives,aes(Area.Name,Oleic))
> a + geom_violin(fill="lightblue") + xlab("Area Name") +
+ ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
Environment History Connections
Files Plots Packages Help Viewer
Previous plot (Ctrl+Alt+F11)
West-Liguria
Umbria
South-Apulia
Sicily
North-Apulia
Inland-Sardinia
East-Liguria
Coastal-Sardnia
Calabria
Area Name
6500 7000 7500 8000 8500
Oleic Acid


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
require(violinmplot)
violinmplot(Area.Name~Oleic,xlab="Oleic Acid",ylab="Area Name",data=Oli
a = ggplot(Olives,aes(Area.Name,Oleic))
a + geom_violin(fill="lightblue") + xlab("Area Name") +
ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
Console Terminal Jobs
> require(s20x)
Loading required package: s20x
> boxqq(Oleic~Region.Name,data=Olives)
Warning messages:
1: In par(oldPar) : graphical parameter "cin" cannot be set
2: In par(oldPar) : graphical parameter "cra" cannot be set
3: In par(oldPar) : graphical parameter "csi" cannot be set
4: In par(oldPar) : graphical parameter "cxy" cannot be set
5: In par(oldPar) : graphical parameter "din" cannot be set
6: In par(oldPar) : graphical parameter "page" cannot be set
> a = ggplot(Olives,aes(Area.Name,Oleic))
> a + geom_boxplot(fill="lightblue") + xlab("Area Name") +
+ ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
> require(violinmplot)
There were 12 warnings (use warnings() to see them)
> violinmplot(Area.Name~Oleic,xlab="Oleic Acid",ylab="Area Name",data=Olives,
horizontal=T)
> a = ggplot(Olives,aes(Area.Name,Oleic))
> a + geom_violin(fill="lightblue") + xlab("Area Name") +
+ ylab("Oleic Acid") + ggtitle("Oleic Acid vs. Growing Area")
Environment History Connections
Files Plots Packages Help Viewer
Oleic Acid vs. Growing Area
8500
8000
7500
7000
6500
Oleic Acid
Calabria Coastal-Sardnia East-Liguria Inland-Sardinia North-Apulia Sicily South-Apulia Umbria West-Liguria
Area Name


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
data("UCBAdmissions")
UCBAdmissions
temp = data.frame(UCBAdmissions)
temp
DeptCount = margin.table(UCBAdmissions,3)
DeptCount
par(mfrow=c(1,2))
pie(DeptCount)
barplot(DeptCount,xlab="Department",ylab="Frequency")
Console Terminal Jobs
12 Rejected Female C 391
13 Admitted Male D 138
14 Rejected Male D 279
15 Admitted Female D 131
16 Rejected Female D 244
17 Admitted Male E 53
18 Rejected Male E 138
19 Admitted Female E 94
20 Rejected Female E 299
21 Admitted Male F 22
22 Rejected Male F 351
23 Admitted Female F 24
24 Rejected Female F 317
> DeptCount = margin.table(UCBAdmissions,3)
> DeptCount
Dept
A B C D E F
933 585 918 792 584 714
Environment History Connections
Files Plots Packages Help Viewer
A
B
C
D
E
F
14:46


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
data("UCBAdmissions")
UCBAdmissions
temp = data.frame(UCBAdmissions)
temp
DeptCount = margin.table(UCBAdmissions,3)
DeptCount
par(mfrow=c(1,2))
pie(DeptCount)
barplot(DeptCount,xlab="Department",ylab="Frequency")
Console Terminal Jobs
17 Admitted Male E 53
18 Rejected Male E 138
19 Admitted Female E 94
20 Rejected Female E 299
21 Admitted Male F 22
22 Rejected Male F 351
23 Admitted Female F 24
24 Rejected Female F 317
> DeptCount = margin.table(UCBAdmissions,3)
> DeptCount
Dept
A B C D E F
933 585 918 792 584 714
> par(mfrow=c(1,2))
> pie(DeptCount)
> barplot(DeptCount,xlab="Department",ylab="Frequency")
Environment History Connections
Files Plots Packages Help Viewer
A
B
C
D
E
F
Frequency
0
200
400
600
800
A B C D E F
Department
14:46


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
DeptCount
par(mfrow=c(1,2))
pie(DeptCount)
barplot(DeptCount,xlab="Department",ylab="Frequency")
par(mfrow=c(1,2))
DeptAdmit = margin.table(UCBAdmissions,c(1,3))
DeptAdmit
barplot(DeptAdmit,xlab="Department",ylab="Frequency")
barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)
GenderAdmit = margin.table(UCBAdmissions,c(1,2))
Console Terminal Jobs
> DeptCount = margin.table(UCBAdmissions,3)
> DeptCount
Dept
A B C D E F
933 585 918 792 584 714
> par(mfrow=c(1,2))
> pie(DeptCount)
> barplot(DeptCount,xlab="Department",ylab="Frequency")
> par(mfrow=c(1,2))
> DeptAdmit = margin.table(UCBAdmissions,c(1,3))
> DeptAdmit
Dept
Admit A B C D E F
Admitted 601 370 322 269 147 46
Rejected 332 215 596 523 437 668
> barplot(DeptAdmit,xlab="Department",ylab="Frequency")
> barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)
Environment History Connections
Files Plots Packages Help Viewer
Frequency
Department
A B C D E F


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
barplot(DeptAdmit,xlab="Department",ylab="Frequency")
barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)
GenderAdmit = margin.table(UCBAdmissions,c(1,2))
GenderAdmit
pie(GenderAdmit[,1],main="Males")
pie(GenderAdmit[,2],main="Females")
barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)
Console Terminal Jobs
> par(mfrow=c(1,2))
> DeptAdmit = margin.table(UCBAdmissions,c(1,3))
> DeptAdmit
Dept
Admit A B C D E F
Admitted 601 370 322 269 147 46
Rejected 332 215 596 523 437 668
> barplot(DeptAdmit,xlab="Department",ylab="Frequency")
> barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)
> GenderAdmit = margin.table(UCBAdmissions,c(1,2))
> GenderAdmit
Gender
Admit Male Female
Admitted 1198 557
Rejected 1493 1278
> pie(GenderAdmit[,1],main="Males")
> pie(GenderAdmit[,2],main="Females")
Environment History Connections
Files Plots Packages Help Viewer
Males
Admitted
Rejected
Females
Admitted
Rejected

RStudio

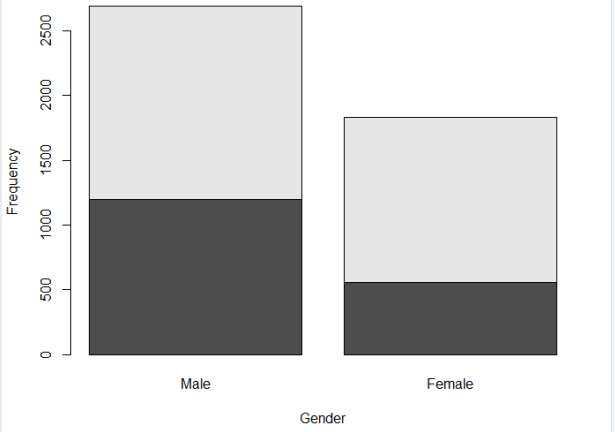
File Edit Code View Plots Session Build Debug Profile Tools Help

graphicsnr.R

```
barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)

GenderAdmit = margin.table(UCBAdmissions,c(1,2))
GenderAdmit

pie(GenderAdmit[,1],main="Males")
pie(GenderAdmit[,2],main="Females")

barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)

par(mfrow=c(1,2))
mosaicplot(~Gender+Admit,data=UCBAdmissions,color=T)
mosaicplot(~Admit+Gender,data=UCBAdmissions,color=T)
```

Console Terminal Jobs

```
> DeptAdmit
          Dept
Admit        A   B   C   D   E   F
  Admitted 601 370 322 269 147  46
  Rejected 332 215 596 523 437 668
>
> barplot(DeptAdmit,xlab="Department",ylab="Frequency")
> barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)
> GenderAdmit = margin.table(UCBAdmissions,c(1,2))
> GenderAdmit
          Gender
Admit      Male Female
  Admitted 1198    557
  Rejected 1493   1278
>
> pie(GenderAdmit[,1],main="Males")
> pie(GenderAdmit[,2],main="Females")
> barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
> barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)
>
```



RStudio

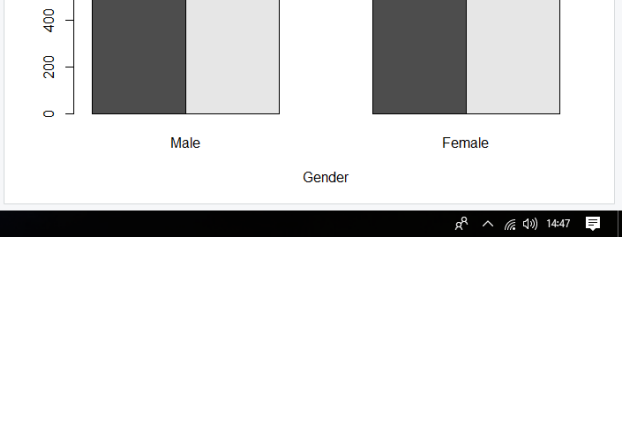
File Edit Code View Plots Session Build Debug Profile Tools Help

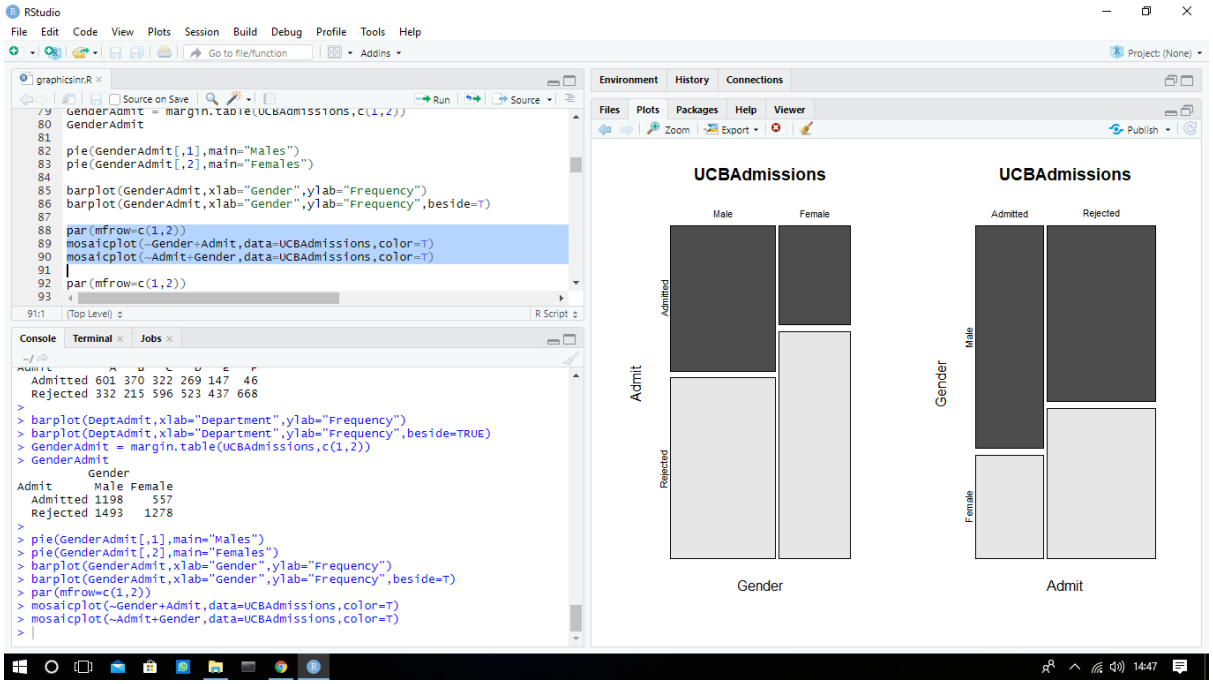
graphicsnr.R

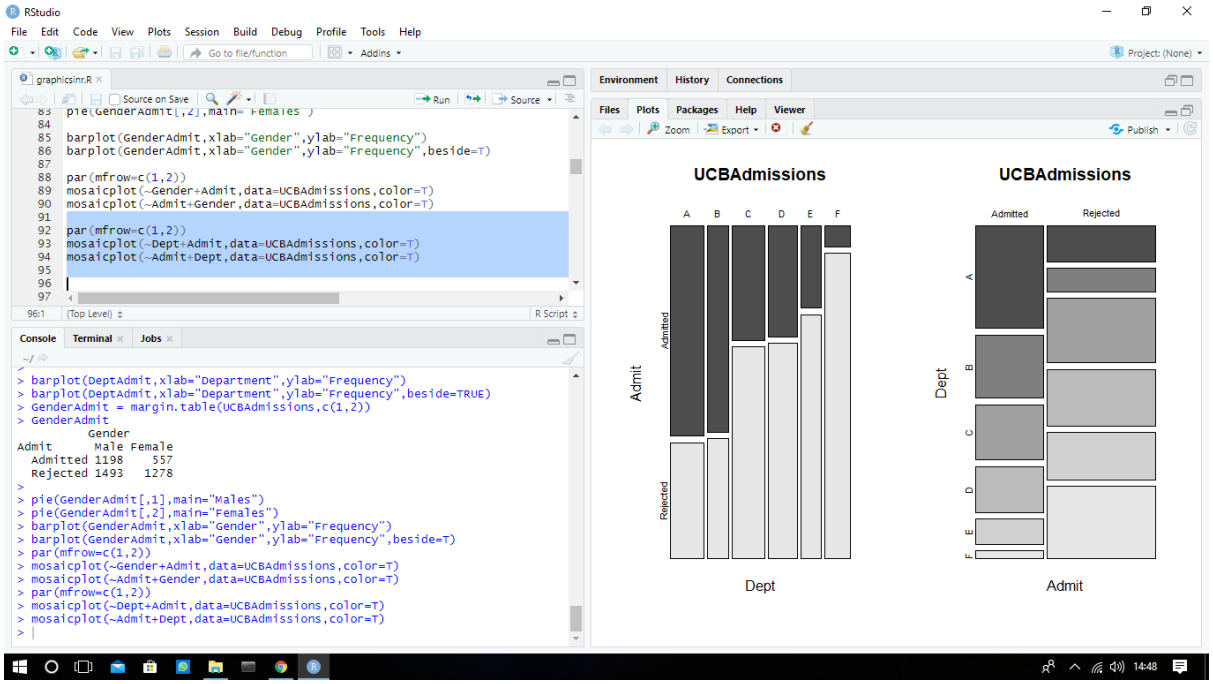
```
barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)

GenderAdmit = margin.table(UCBAdmissions,c(1,2))
GenderAdmit

pie(GenderAdmit[,1],main="Males")
pie(GenderAdmit[,2],main="Females")

barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)

par(mfrow=c(1,2))
mosaicplot(~Gender+Admit,data=UCBAdmissions,color=T)
mosaicplot(~Admit+Gender,data=UCBAdmissions,color=T)
```

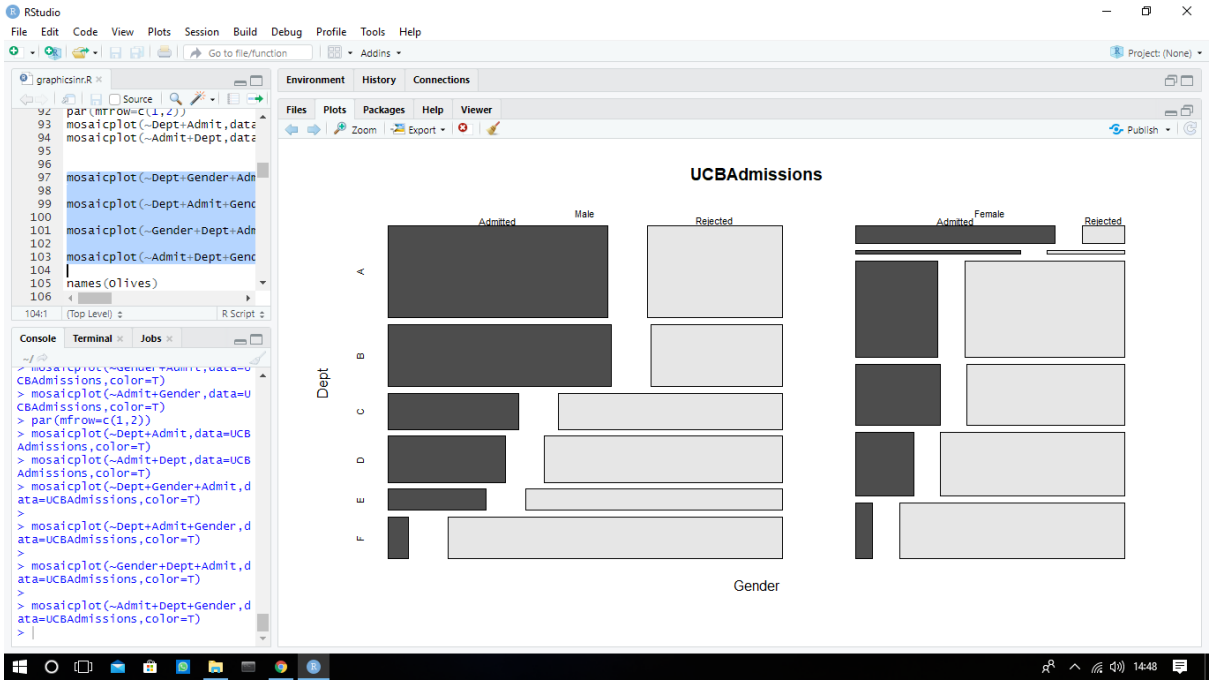
Console Terminal Jobs

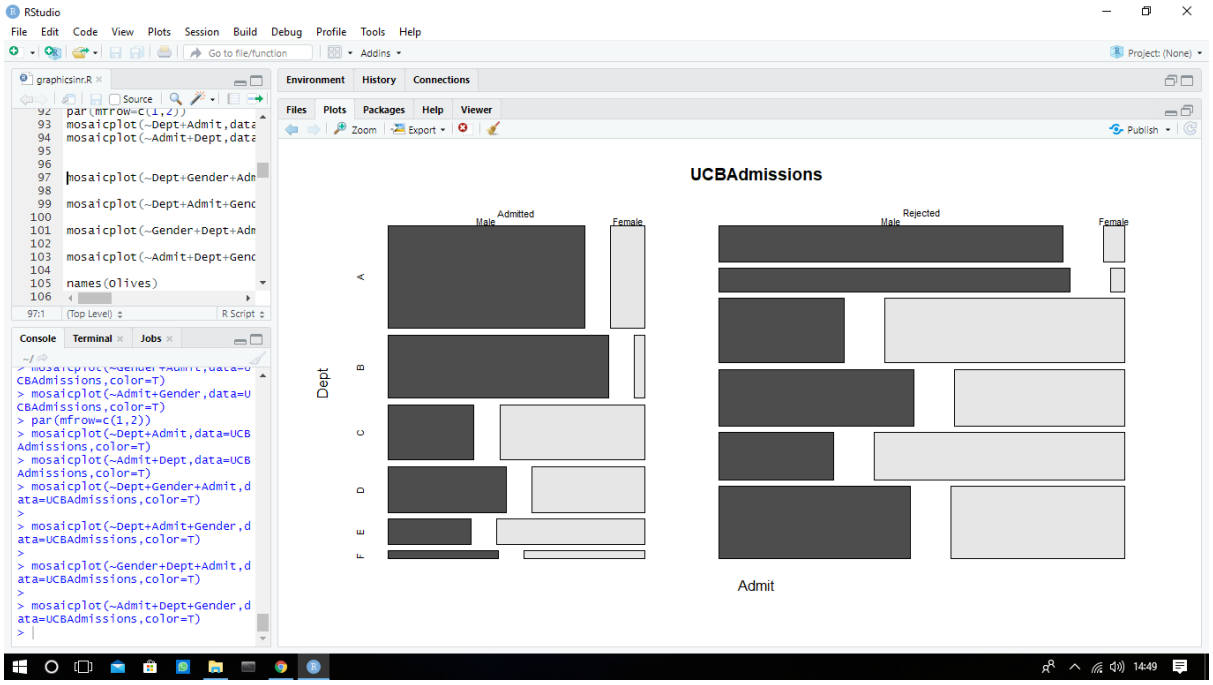
```
> DeptAdmit
          Dept
Admit        A   B   C   D   E   F
  Admitted 601 370 322 269 147  46
  Rejected 332 215 596 523 437 668
>
> barplot(DeptAdmit,xlab="Department",ylab="Frequency")
> barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)
> GenderAdmit = margin.table(UCBAdmissions,c(1,2))
> GenderAdmit
          Gender
Admit      Male Female
  Admitted 1198    557
  Rejected 1493   1278
>
> pie(GenderAdmit[,1],main="Males")
> pie(GenderAdmit[,2],main="Females")
> barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
> barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)
>
```




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsnr.R
Source on Save
Run
Source
GenderAdmit = margin.table(UCBAdmissions,c(1,2))
GenderAdmit
pie(GenderAdmit[,1],main="Males")
pie(GenderAdmit[,2],main="Females")
barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)
par(mfrow=c(1,2))
mosaicplot(~Gender+Admit,data=UCBAdmissions,color=T)
mosaicplot(~Admit+Gender,data=UCBAdmissions,color=T)
par(mfrow=c(1,2))
91:1 (Top Level) R Script
Console Terminal Jobs
Admitted 601 370 322 269 147 46
Rejected 332 215 596 523 437 668
> barplot(DeptAdmit,xlab="Department",ylab="Frequency")
> barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)
> GenderAdmit = margin.table(UCBAdmissions,c(1,2))
> GenderAdmit
Gender
Admit Male Female
Admitted 1198 557
Rejected 1493 1278
> pie(GenderAdmit[,1],main="Males")
> pie(GenderAdmit[,2],main="Females")
> barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
> barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)
> par(mfrow=c(1,2))
> mosaicplot(~Gender+Admit,data=UCBAdmissions,color=T)
> mosaicplot(~Admit+Gender,data=UCBAdmissions,color=T)
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
UCBAdmissions
Male Female
Admitted Rejected
Admit
Gender
UCBAdmissions
Admitted Rejected
Male Female
Gender
Admit
14:47


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsnr.R
Source on Save
Run
Source
barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)
par(mfrow=c(1,2))
mosaicplot(~Gender+Admit,data=UCBAdmissions,color=T)
mosaicplot(~Admit+Gender,data=UCBAdmissions,color=T)
par(mfrow=c(1,2))
mosaicplot(~Dept+Admit,data=UCBAdmissions,color=T)
mosaicplot(~Admit+Dept,data=UCBAdmissions,color=T)
96:1 (Top Level) R Script
Console Terminal Jobs
> barplot(DeptAdmit,xlab="Department",ylab="Frequency")
> barplot(DeptAdmit,xlab="Department",ylab="Frequency",beside=TRUE)
> GenderAdmit = margin.table(UCBAdmissions,c(1,2))
> GenderAdmit
Gender
Admit Male Female
Admitted 1198 557
Rejected 1493 1278
> pie(GenderAdmit[,1],main="Males")
> pie(GenderAdmit[,2],main="Females")
> barplot(GenderAdmit,xlab="Gender",ylab="Frequency")
> barplot(GenderAdmit,xlab="Gender",ylab="Frequency",beside=T)
> par(mfrow=c(1,2))
> mosaicplot(~Gender+Admit,data=UCBAdmissions,color=T)
> mosaicplot(~Admit+Gender,data=UCBAdmissions,color=T)
> par(mfrow=c(1,2))
> mosaicplot(~Dept+Admit,data=UCBAdmissions,color=T)
> mosaicplot(~Admit+Dept,data=UCBAdmissions,color=T)
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
UCBAdmissions
A B C D E F
Admitted Rejected
Admit
Dept
UCBAdmissions
Admitted Rejected
A B C D E F
Dept
Admit
14:48


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
Environment History Connections
Files Plots Packages Help Viewer
Console Terminal Jobs
UCBAdmissions
Male
Female
Admitted
Rejected
Dept
A
B
C
D
E
F
Gender

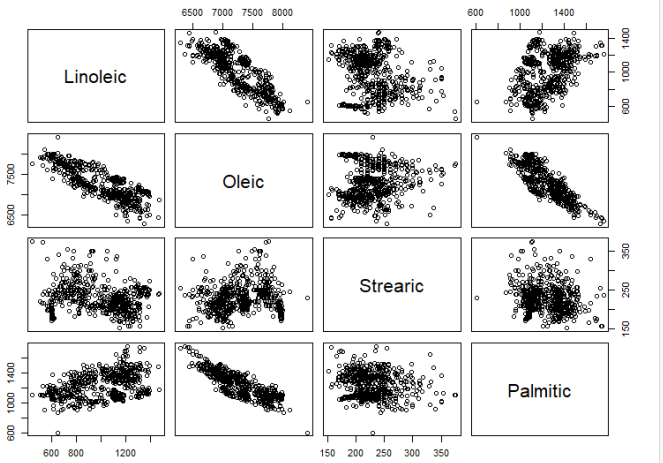

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
Environment History Connections
Files Plots Packages Help Viewer
Console Terminal Jobs
UCBAdmissions
Admitted
Rejected
Male
Female
Dept
A
B
C
D
E
F
Admit

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
require(corrplot)

Boston.corr = cor(Boston[,-c(1:4)])   # find pairwise correla
options(digits=2)  # reduce the number of significant digits
Boston.corr

corrplot(Boston.corr)

corrplot(Boston.corr,method="ellipse")

corrplot(Boston.corr,order="FPC")

corrplot(Boston.corr,order="hclust")
```

```
          TAX PTRATIO      B  LSTAT
CMEDV  -0.472   -0.51  0.335 -0.741
CRIM    0.583    0.29 -0.385  0.456
ZN     -0.315   -0.39  0.176 -0.413
INDUS   0.721    0.38 -0.357  0.604
CHAS   -0.036   -0.12  0.049 -0.054
NOX     0.668    0.19 -0.380  0.591
RM     -0.292   -0.36  0.128 -0.614
AGE     0.506    0.26 -0.274  0.602
DIS    -0.534   -0.23  0.292 -0.497
RAD     0.910    0.46 -0.444  0.489
TAX     1.000    0.46 -0.442  0.544
PTRATIO 0.461    1.00 -0.177  0.374
B      -0.442   -0.18  1.000 -0.366
LSTAT   0.544    0.37 -0.366  1.000
> corrplot(Boston.corr)
> corrplot(Boston.corr,method="ellipse")
```



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
require(corrplot)

Boston.corr = cor(Boston[,-c(1:4)])   # find pairwise correla
options(digits=2)  # reduce the number of significant digits
Boston.corr

corrplot(Boston.corr)

corrplot(Boston.corr,method="ellipse")

corrplot(Boston.corr,order="FPC")

corrplot(Boston.corr,order="hclust")
```

```
          TAX PTRATIO      B  LSTAT
CMEDV  -0.472   -0.51  0.335 -0.741
CRIM    0.583    0.29 -0.385  0.456
ZN     -0.315   -0.39  0.176 -0.413
INDUS   0.721    0.38 -0.357  0.604
CHAS   -0.036   -0.12  0.049 -0.054
NOX     0.668    0.19 -0.380  0.591
RM     -0.292   -0.36  0.128 -0.614
AGE     0.506    0.26 -0.274  0.602
DIS    -0.534   -0.23  0.292 -0.497
RAD     0.910    0.46 -0.444  0.489
TAX     1.000    0.46 -0.442  0.544
PTRATIO 0.461    1.00 -0.177  0.374
B      -0.442   -0.18  1.000 -0.366
LSTAT   0.544    0.37 -0.366  1.000
> corrplot(Boston.corr)
> corrplot(Boston.corr,method="ellipse")
```

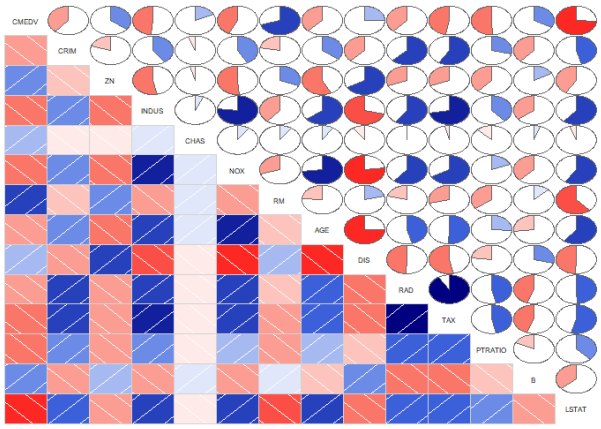
RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
172 require(corrplot)
173
174 Boston.corr = cor(Boston[,-c(1:4)])   # find pairwise correla
175 options(digits=2)  # reduce the number of significant digits
176 Boston.corr
177
178 corrplot(Boston.corr)
179
180 corrplot(Boston.corr,method="ellipse")
181
182 corrplot(Boston.corr,order="FPC")
183
184 corrplot(Boston.corr,order="hclust")
185
186
```

181:1 (Top Level) R Script

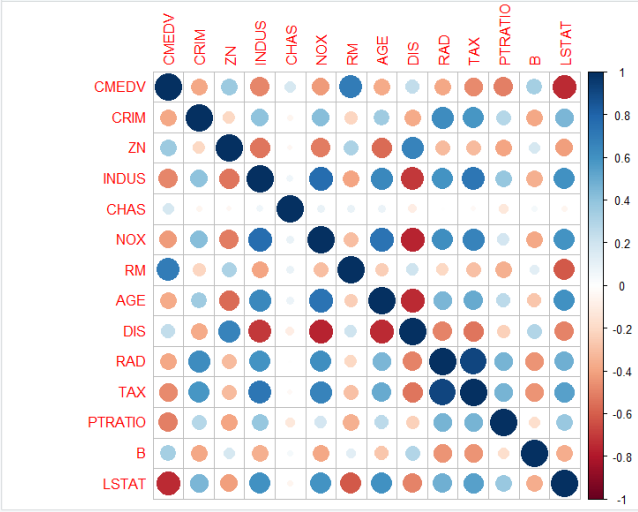
```
          TAX PTRATIO      B  LSTAT
CMEDV  -0.472   -0.51  0.335 -0.741
CRIM    0.583    0.29 -0.385  0.456
ZN     -0.315   -0.39  0.176 -0.413
INDUS   0.721    0.38 -0.357  0.604
CHAS   -0.036   -0.12  0.049 -0.054
NOX     0.668    0.19 -0.380  0.591
RM     -0.292   -0.36  0.128 -0.614
AGE     0.506    0.26 -0.274  0.602
DIS    -0.534   -0.23  0.292 -0.497
RAD     0.910    0.46 -0.444  0.489
TAX     1.000    0.46 -0.442  0.544
PTRATIO 0.461    1.00 -0.177  0.374
B      -0.442   -0.18  1.000 -0.366
LSTAT   0.544    0.37 -0.366  1.000
>
> corrplot(Boston.corr)
>
> corrplot(Boston.corr,method="ellipse")
>
```



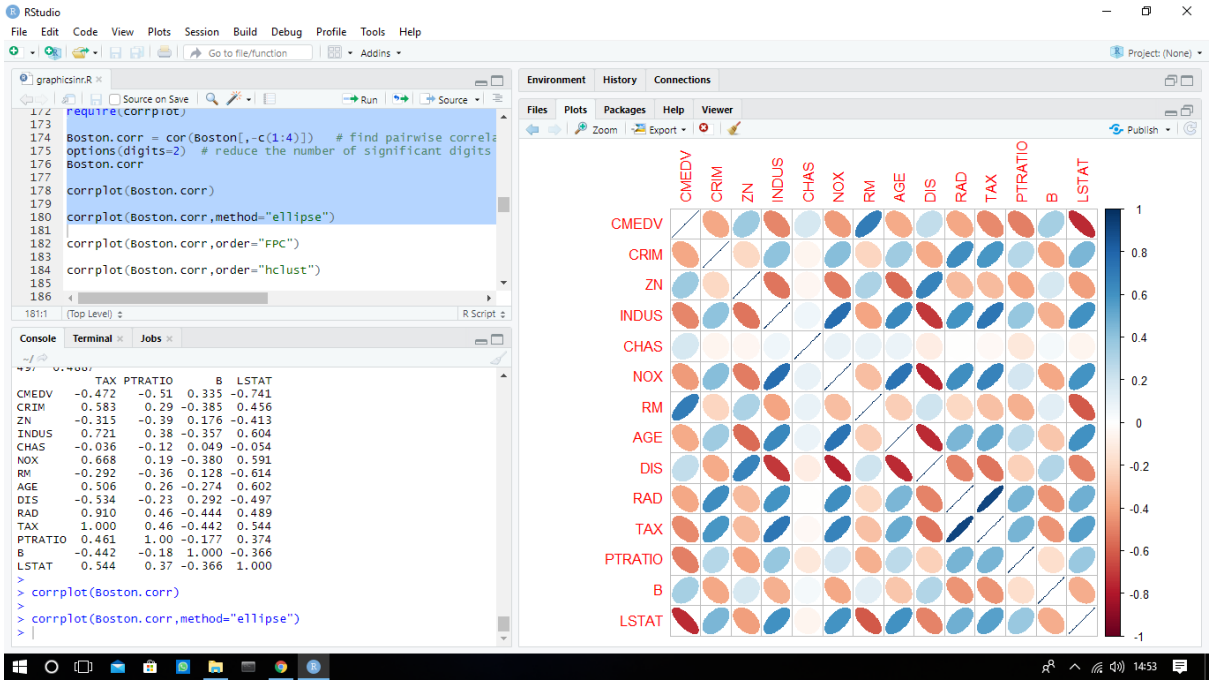
14:52

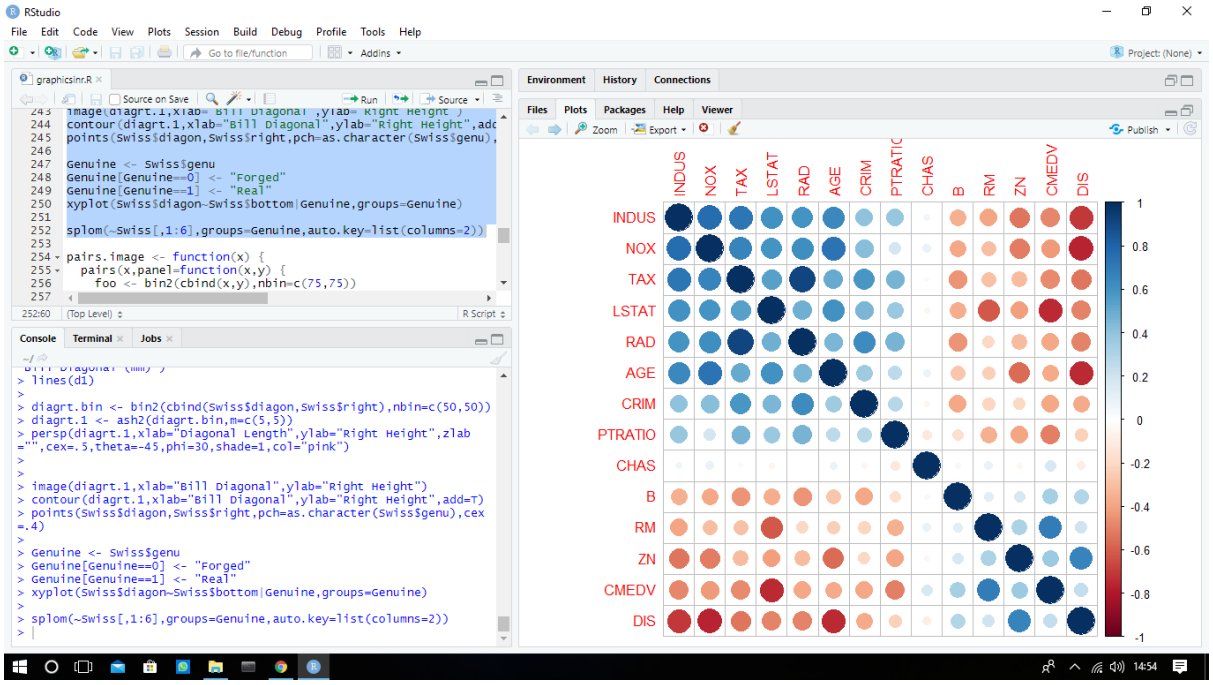
RStudio

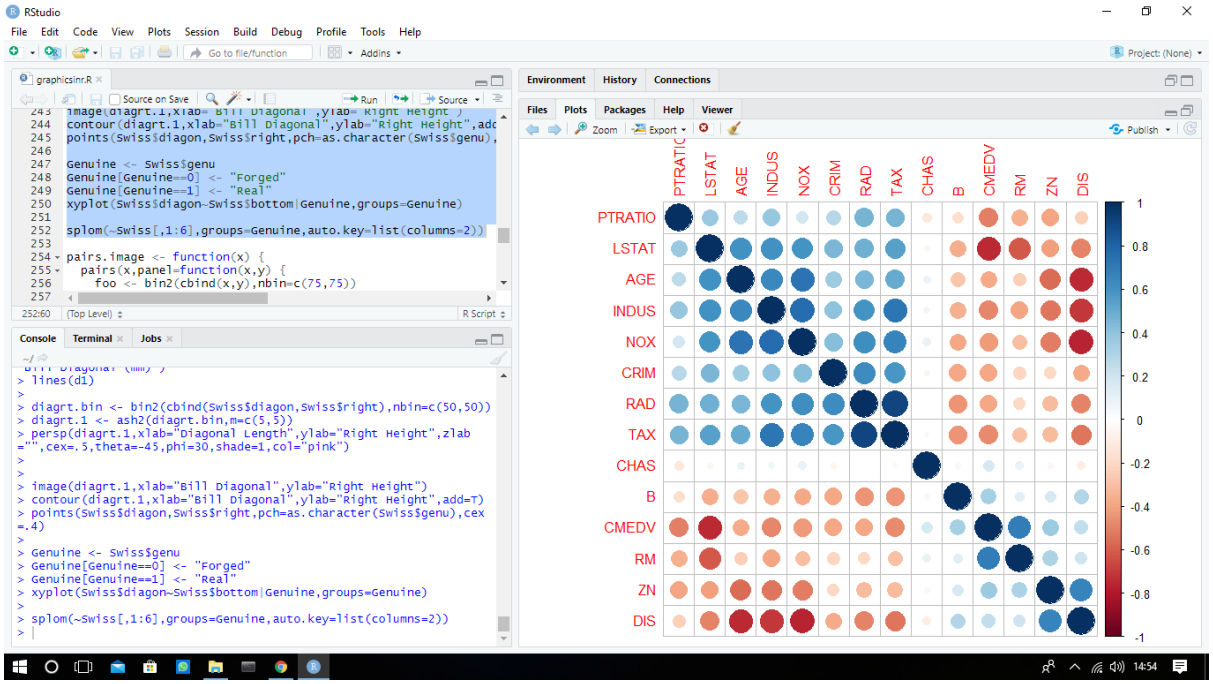
File Edit Code View Plots Session Build Debug Profile Tools Help

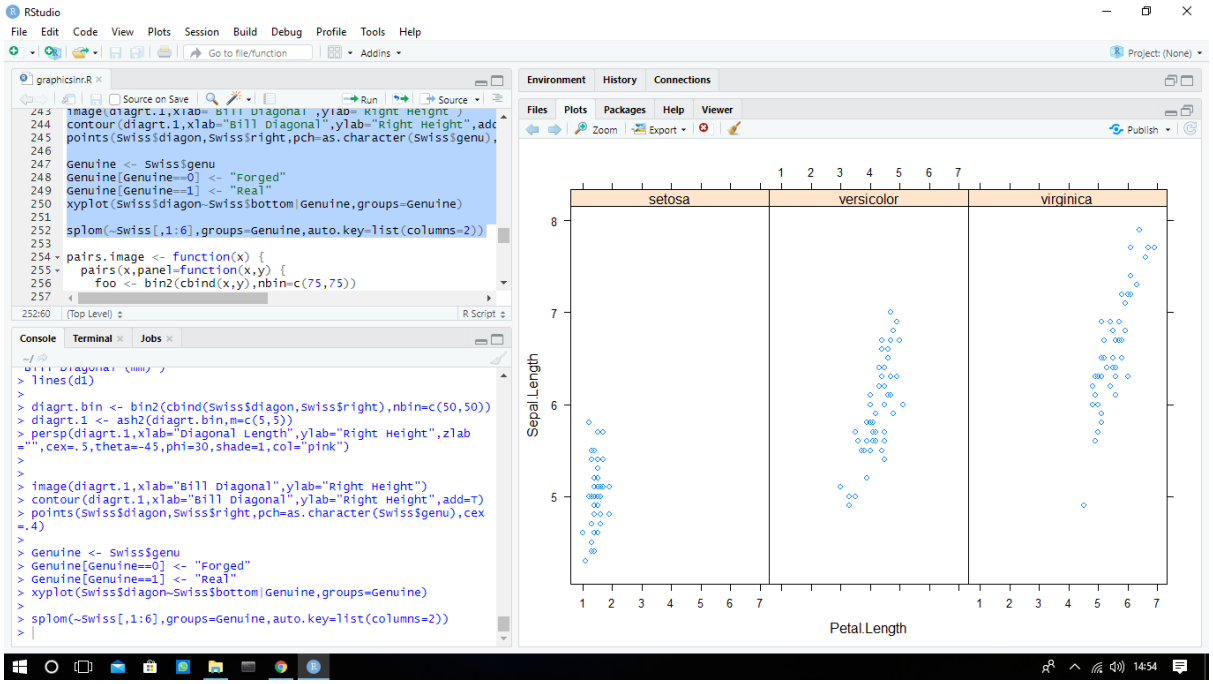


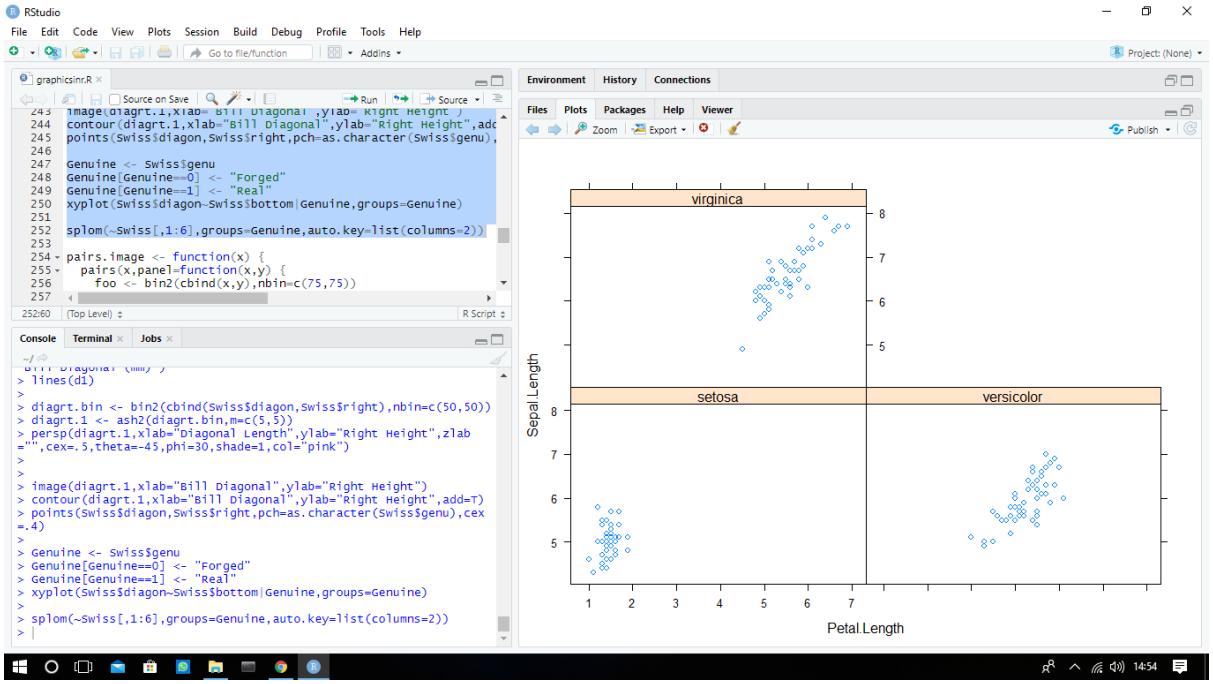
14:53


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
require(corrplot)
Boston.corr = cor(Boston[,-c(1:4)]) # find pairwise correla
options(digits=2) # reduce the number of significant digits
Boston.corr
corrplot(Boston.corr)
corrplot(Boston.corr,method="ellipse")
corrplot(Boston.corr,order="FPC")
corrplot(Boston.corr,order="hclust")
Console Terminal Jobs
TAX PTRATIO B LSTAT
CMEDV -0.472 -0.51 0.335 -0.741
CRIM 0.583 0.29 -0.385 0.456
ZN -0.315 -0.39 0.176 -0.413
INDUS 0.721 0.38 -0.357 0.604
CHAS -0.036 -0.12 0.049 -0.054
NOX 0.668 0.19 -0.380 0.591
RM -0.292 -0.36 0.128 -0.614
AGE 0.506 0.26 -0.274 0.602
DIS -0.534 -0.23 0.292 -0.497
RAD 0.910 0.46 -0.444 0.489
TAX 1.000 0.46 -0.442 0.544
PTRATIO 0.461 1.00 -0.177 0.374
B -0.442 -0.18 1.000 -0.366
LSTAT 0.544 0.37 -0.366 1.000
> corrplot(Boston.corr)
> corrplot(Boston.corr,method="ellipse")
Environment History Connections
Files Plots Packages Help Viewer
CMEDV CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX PTRATIO B LSTAT
1 0.8 0.6 0.4 0.2 0 -0.2 -0.4 -0.6 -0.8 -1
14:53


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
Genuine <- Swiss$genu
Genuine[Genuine==0] <- "Forged"
Genuine[Genuine==1] <- "Real"
xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
pairs.image <- function(x) {
pairs(x,panel=function(x,y) {
foo <- bin2(cbind(x,y),nbin=c(75,75))
Console Terminal Jobs
> lines(d1)
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
> image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
> contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add=T)
> points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),cex
=.4)
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
Environment History Connections
Files Plots Packages Help Viewer
INDUS NOX TAX LSTAT RAD AGE CRIM PTRATIO CHAS B RM ZN CMEDV DIS
1 0.8 0.6 0.4 0.2 0 -0.2 -0.4 -0.6 -0.8 -1
14:54


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
Source on Save
Run
Source
243 image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252 splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
253
254 pairs.image <- function(x) {
255 pairs(x,panel=function(x,y) {
256 foo <- bin2(cbind(x,y),nbin=c(75,75))
257
252:60 (Top Level) R Script
Console Terminal Jobs
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
>
> image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
> contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add=T)
> points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),cex
=.4)
>
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
>
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
>
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
PTRATIO LSTAT AGE INDUS NOX CRIM RAD TAX CHAS B CMEDV RM ZN DIS
1 0.8 0.6 0.4 0.2 0 -0.2 -0.4 -0.6 -0.8 -1
14:54


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
Source on Save
Run
Source
243 image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252 splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
253
254 pairs.image <- function(x) {
255 pairs(x,panel=function(x,y) {
256 foo <- bin2(cbind(x,y),nbin=c(75,75))
257
252:60 (Top Level) R Script
Console Terminal Jobs
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
>
> image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
> contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add=T)
> points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),cex
=.4)
>
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
>
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
>
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
setosa versicolor virginica
Sepal.Length
Petal.Length
14:54

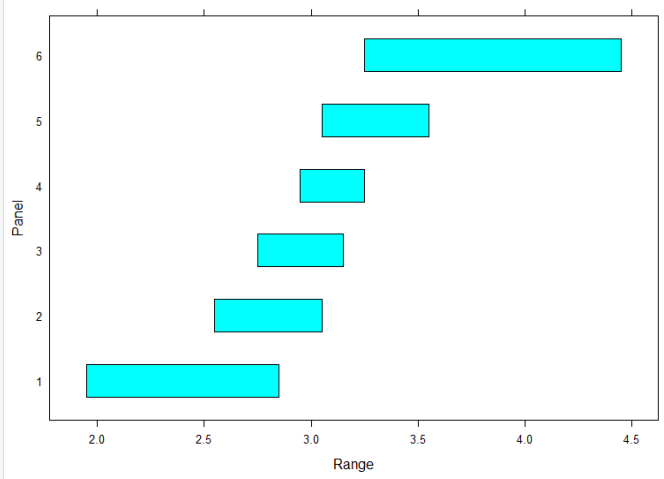

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsnr.R
Source on Save
Run
Source
image(diagrt.1,xlab= Bill Diagonal ",ylab= Right Height ")
contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
Genuine <- Swiss$genu
Genuine[Genuine==0] <- "Forged"
Genuine[Genuine==1] <- "Real"
xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
pairs.image <- function(x) {
pairs(x,panel=function(x,y) {
foo <- bin2(cbind(x,y),nbin=c(75,75))
252:60
(Top Level)
R Script
Console
Terminal
Jobs
> lines(d1)
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
> image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
> contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add=T)
> points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),cex
=.4)
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
Environment
History
Connections
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
virginica
setosa
versicolor
Sepal.Length
Petal.Length
14:54

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252 splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
253
254 pairs.image <- function(x) {
255   pairs(x,panel=function(x,y) {
256     foo <- bin2(cbind(x,y),nbin=c(75,75))
257
```

```
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
>
> image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
> contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add=T)
> points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),cex
=.4)
>
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
>
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
>
```

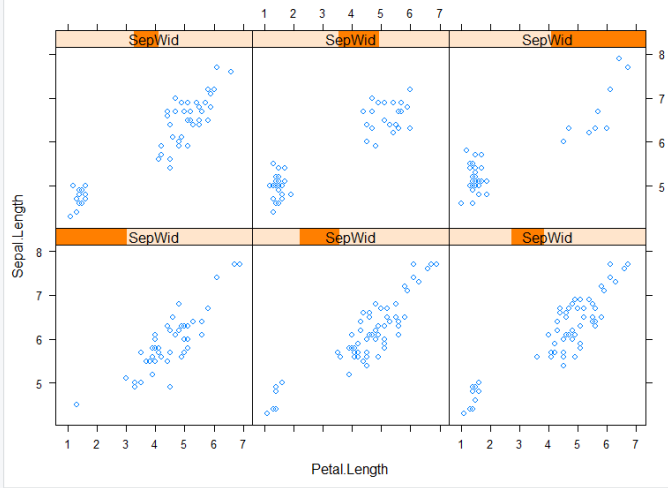


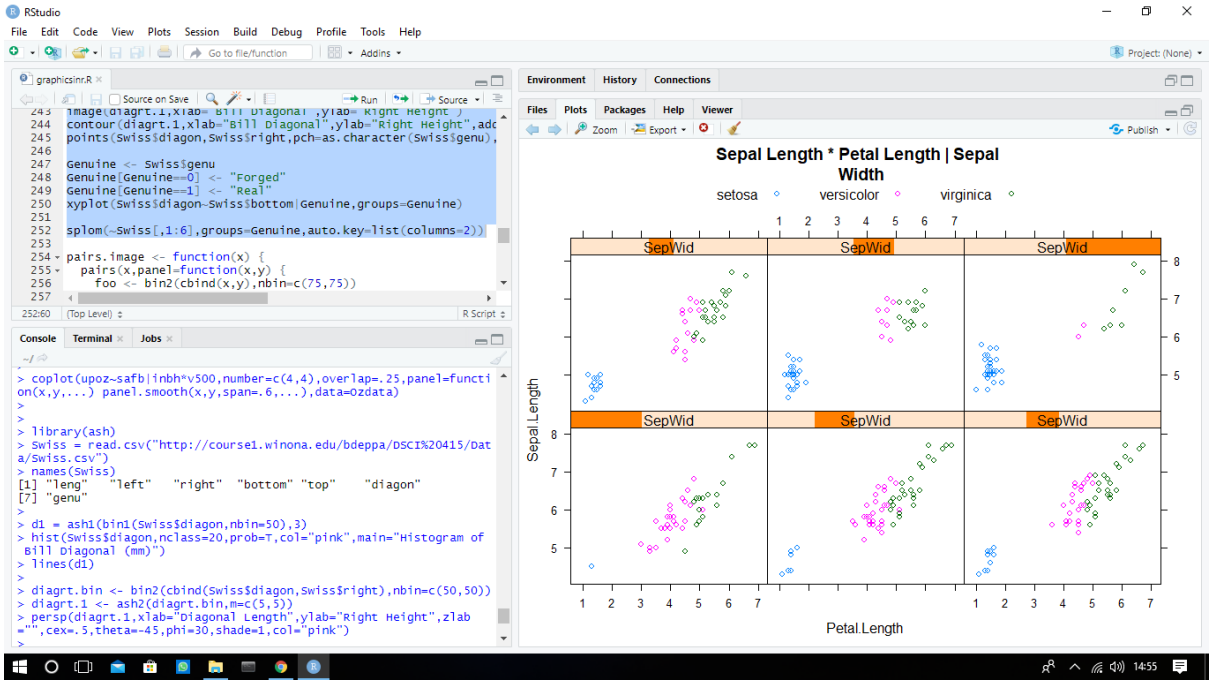
RStudio

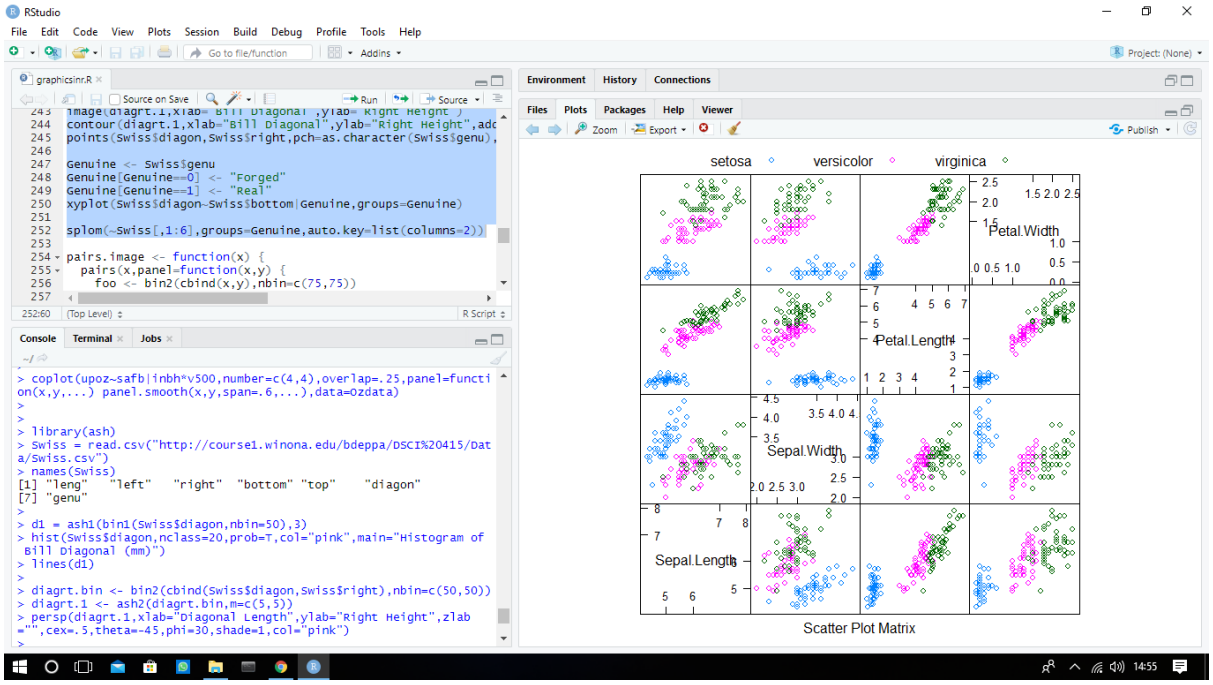
File Edit Code View Plots Session Build Debug Profile Tools Help

```
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252 splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
253
254 pairs.image <- function(x) {
255   pairs(x,panel=function(x,y) {
256     foo <- bin2(cbind(x,y),nbin=c(75,75))
257
```

```
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
>
>
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng"   "left"   "right"  "bottom" "top"    "diagon"
[7] "genu"
>
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
 Bill Diagonal (mm)")
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
```




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsnr.R
Source on Save
Run
Source
243 image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252 splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
253
254 pairs.image <- function(x) {
255 pairs(x,panel=function(x,y) {
256 foo <- bin2(cbind(x,y),nbin=c(75,75))
257
252:60 (Top Level) R Script
Console Terminal Jobs
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
>
>
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng" "left" "right" "bottom" "top" "diagon"
[7] "genu"
>
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
Bill Diagonal (mm)")
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
Sepal Length * Petal Length | Sepal Width
setosa versicolor virginica
SepWid
Sepal.Length
Petal.Length
14:55

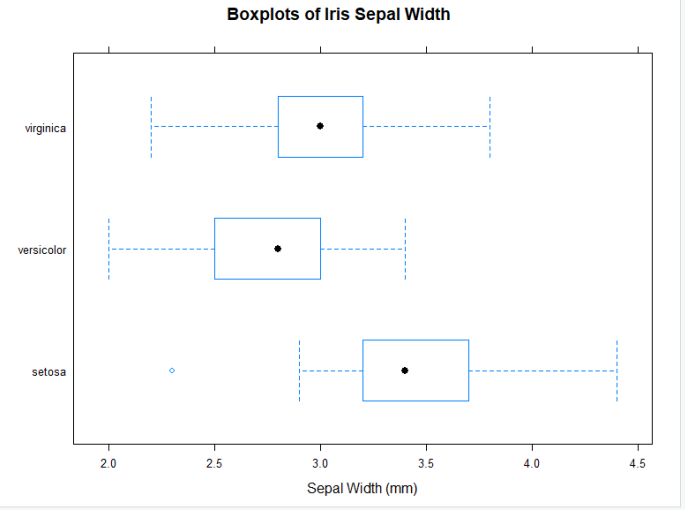

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsnr.R
Source on Save
Run
Source
243 image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252 splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
253
254 pairs.image <- function(x) {
255 pairs(x,panel=function(x,y) {
256 foo <- bin2(cbind(x,y),nbin=c(75,75))
257
252:60 (Top Level) R Script
Console Terminal Jobs
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
>
>
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng" "left" "right" "bottom" "top" "diagon"
[7] "genu"
>
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
Bill Diagonal (mm)")
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
setosa versicolor virginica
Petal.Width
Petal.Length
Sepal.Width
Sepal.Length
Scatter Plot Matrix
14:55

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
243 image(diagrt.1,xlab= Bill Diagonal ,ylab= Right Height )
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252 splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
253
254 pairs.image <- function(x) {
255   pairs(x,panel=function(x,y) {
256     foo <- bin2(cbind(x,y),nbin=c(75,75))
257
```

```
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
>
>
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng"   "left"   "right"  "bottom" "top"    "diagon"
[7] "genu"
>
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
 Bill Diagonal (mm)")
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
```

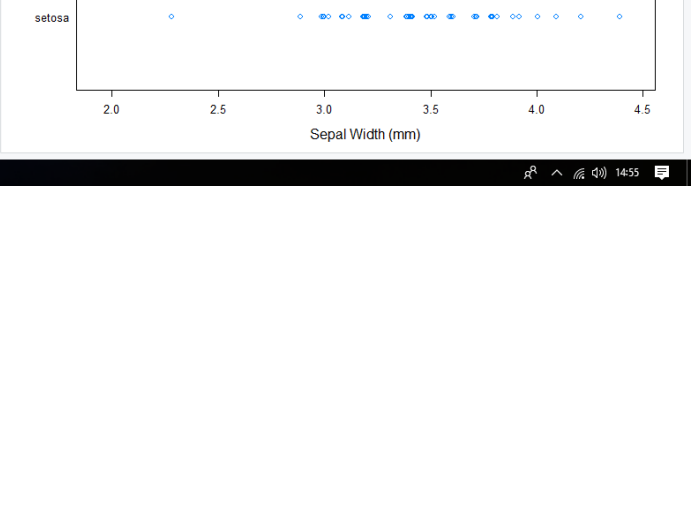


RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
243 image(diagrt.1,xlab= Bill Diagonal ,ylab= Right Height )
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252 splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
253
254 pairs.image <- function(x) {
255   pairs(x,panel=function(x,y) {
256     foo <- bin2(cbind(x,y),nbin=c(75,75))
257
```

```
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
>
>
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng"   "left"   "right"  "bottom" "top"    "diagon"
[7] "genu"
>
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
 Bill Diagonal (mm)")
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
```



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

graphicsinr.R

```
239 diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
240 persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zla
241
242
243 image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252
```

252:60 (Top Level) R Script

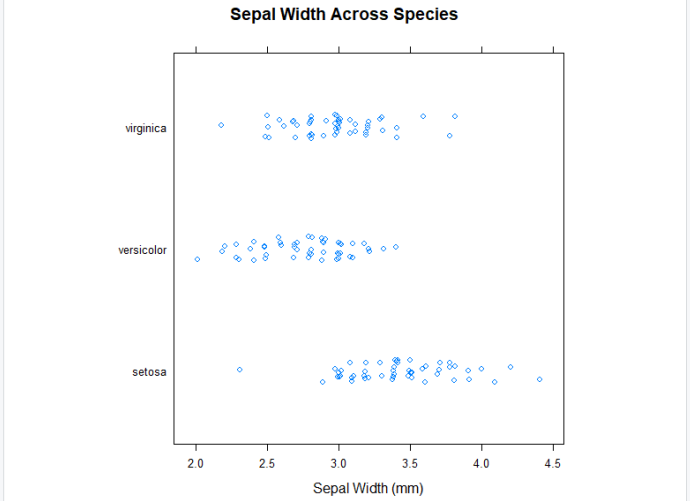
Console Terminal Jobs

```
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
>
>
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng"   "left"   "right"  "bottom" "top"    "diagon"
[7] "genu"
>
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
 Bill Diagonal (mm)")
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
```

Environment History Connections

Files Plots Packages Help Viewer

Zoom Export Publish



14:56

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

graphicsinr.R

```
239 diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
240 persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zla
241
242
243 image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252
```

252:60 (Top Level) R Script

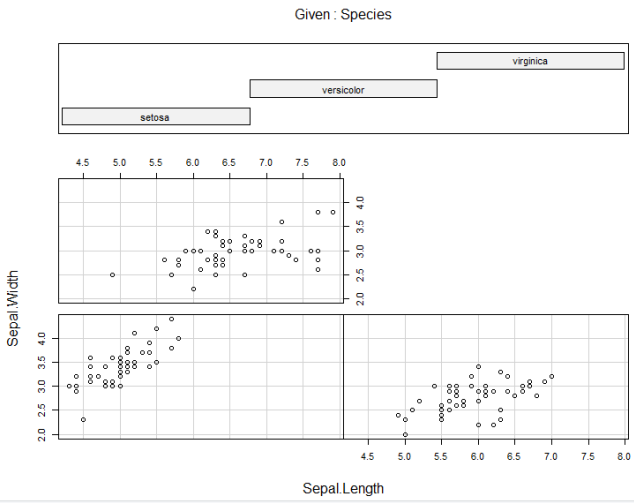
Console Terminal Jobs

```
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
>
>
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng"   "left"   "right"  "bottom" "top"    "diagon"
[7] "genu"
>
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
 Bill Diagonal (mm)")
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
```

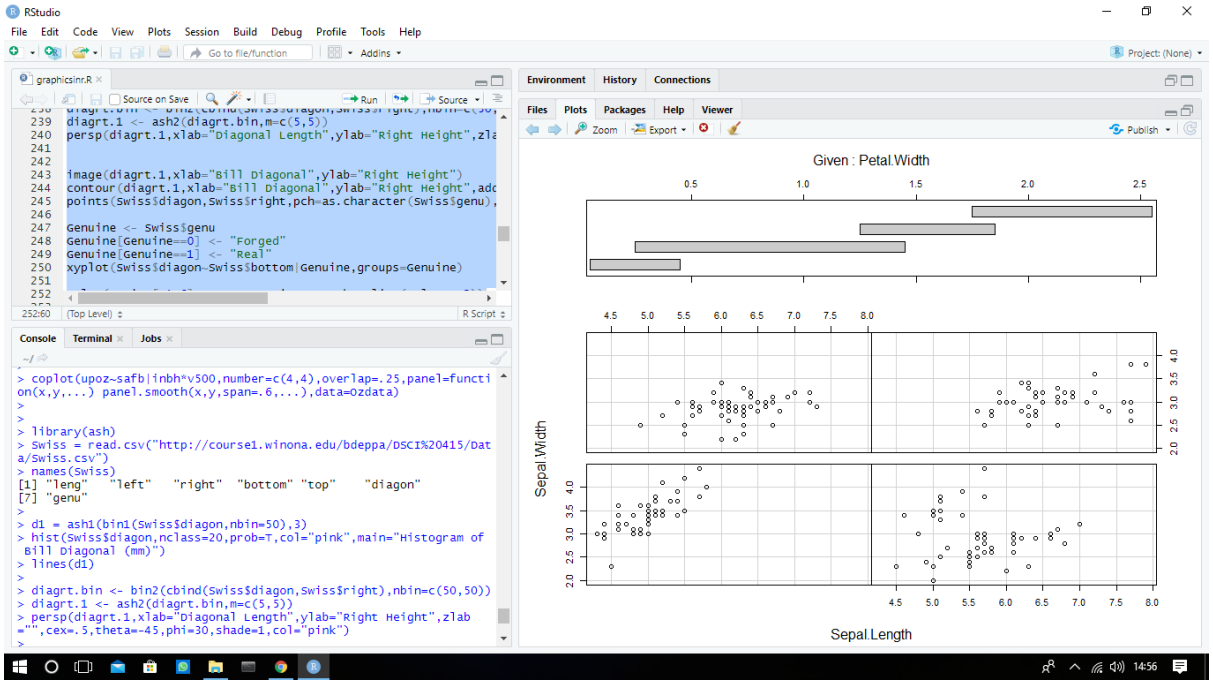
Environment History Connections

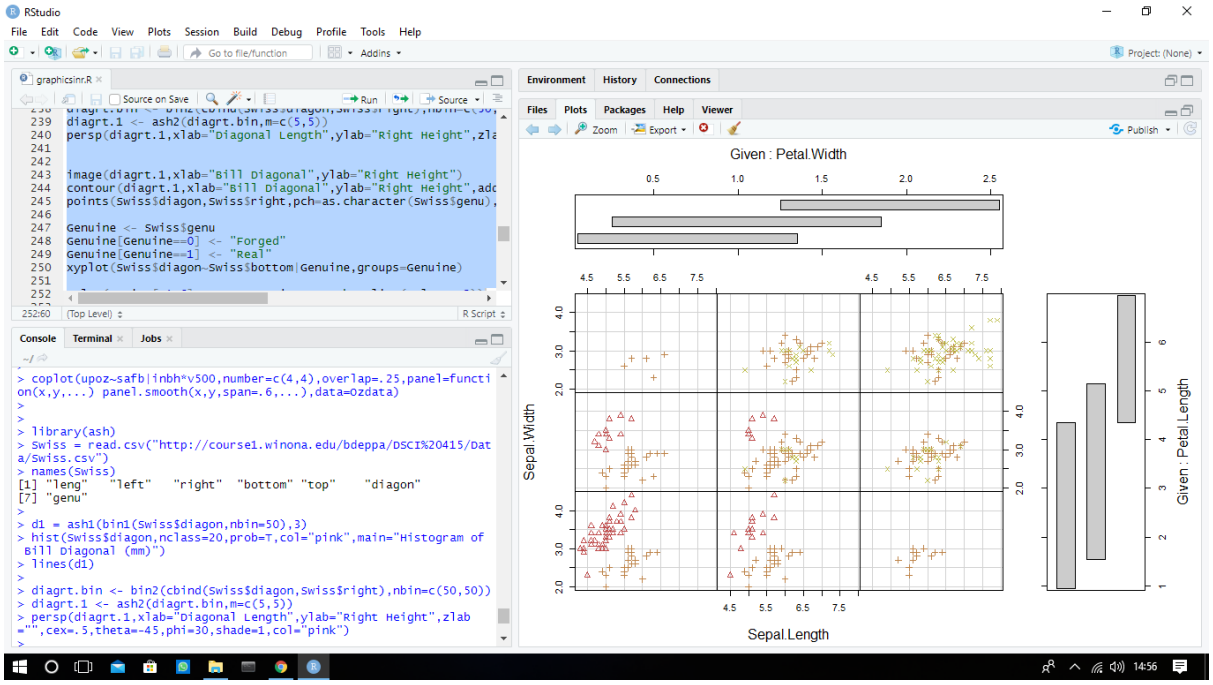
Files Plots Packages Help Viewer

Zoom Export Publish



14:56


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zla
image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
Genuine <- Swiss$genu
Genuine[Genuine==0] <- "Forged"
Genuine[Genuine==1] <- "Real"
xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
Console Terminal Jobs
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng" "left" "right" "bottom" "top" "diagon"
[7] "genu"
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
Bill Diagonal (mm)")
> lines(d1)
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
Environment History Connections
Files Plots Packages Help Viewer
Given : Petal.Width
Sepal.Width
Sepal.Length

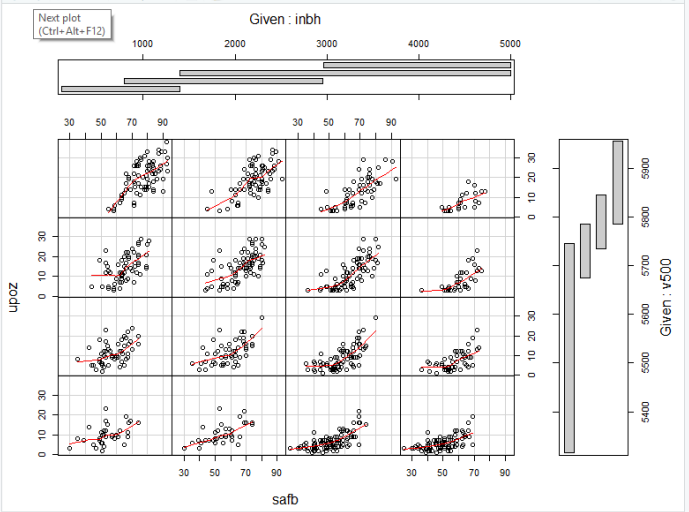

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R
diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zla
image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
Genuine <- Swiss$genu
Genuine[Genuine==0] <- "Forged"
Genuine[Genuine==1] <- "Real"
xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
Console Terminal Jobs
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng" "left" "right" "bottom" "top" "diagon"
[7] "genu"
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
Bill Diagonal (mm)")
> lines(d1)
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
Environment History Connections
Files Plots Packages Help Viewer
Given : Petal.Width
Sepal.Width
Given : Petal.Length
Sepal.Length

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

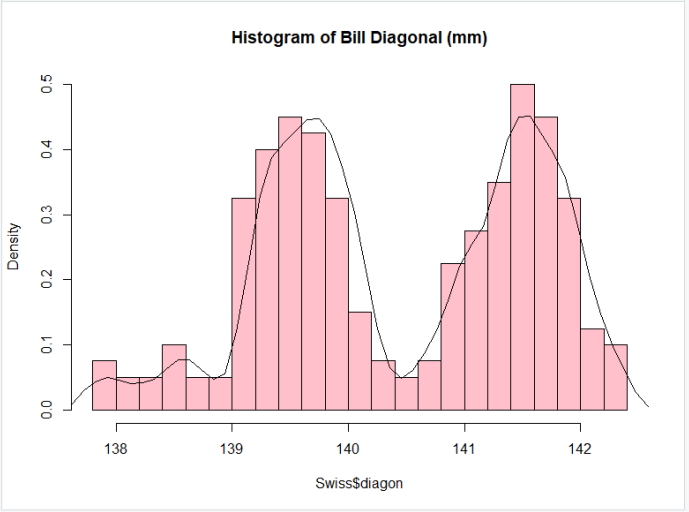
```
239 diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
240 persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zla
241
242
243 image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
244 contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add
245 points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),
246
247 Genuine <- Swiss$genu
248 Genuine[Genuine==0] <- "Forged"
249 Genuine[Genuine==1] <- "Real"
250 xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
251
252
```

```
> coplot(upoz~safb|inbh*v500,number=c(4,4),overlap=.25,panel=functi
on(x,y,...) panel.smooth(x,y,span=.6,...),data=Ozdata)
>
>
> library(ash)
> Swiss = read.csv("http://course1.winona.edu/bdeppa/DSCI%20415/Dat
a/Swiss.csv")
> names(Swiss)
[1] "leng"   "left"   "right"  "bottom" "top"    "diagon"
[7] "genu"
>
> d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
> hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram of
 Bill Diagonal (mm)")
> lines(d1)
>
> diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,50))
> diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="pink")
>
```



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

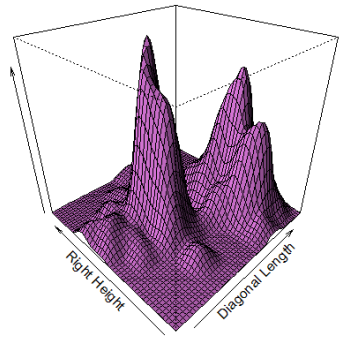


RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
234  d1 = ash1(bin1(Swiss$diagon,nbin=50),3)
235  hist(Swiss$diagon,nclass=20,prob=T,col="pink",main="Histogram
236  lines(d1)
237
238  diagrt.bin <- bin2(cbind(Swiss$diagon,Swiss$right),nbin=c(50,
239  diagrt.1 <- ash2(diagrt.bin,m=c(5,5))
240  persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zla
241
242
243  image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
244
```

```
> image(diagrt.1,xlab="Bill Diagonal",ylab="Right Height")
> contour(diagrt.1,xlab="Bill Diagonal",ylab="Right Height",add=T)
> points(Swiss$diagon,Swiss$right,pch=as.character(Swiss$genu),cex
=.4)
>
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
>
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="purple")
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=1,col="violet")
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=2,col="violet")
> persp(diagrt.1,xlab="Diagonal Length",ylab="Right Height",zlab
="",cex=.5,theta=-45,phi=30,shade=0.5,col="violet")
>
```

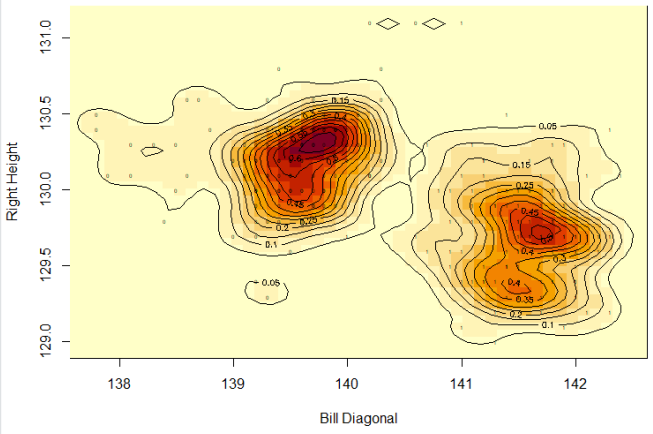


RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
298    points(kodiak.crab$longit,kodiak.crab$latit,pch=".")
299  }
300  nr.pp = nrec/npots
301  crab.plot(lat,long,y=nr.pp,year=83,yname="Number of Recruits
302
303  detach(survey.crab)
304  detach(kodiak.crab)  # if you attach, be sure to detach when
305
306  attach(City)         # This command readies the data frame cit
307
308
```

```
>
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
>
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
>
> pairs.image <- function(x) {
+   pairs(x,panel=function(x,y) {
+     foo <- bin2(cbind(x,y),nbin=c(75,75))
+     foo <- ash2(foo,m=c(6,6))
+     image(foo,add=T,xlab="",ylab="",col=topo.colors(1000))
+     points(x,y,pch=".")
+   })
+ }
> pairs.image(Swiss[,1:6])
>
> pairs.persp <- function(x) {
+   par(bg="sky blue")
+   pairs(x,panel=function(x,y) {
```

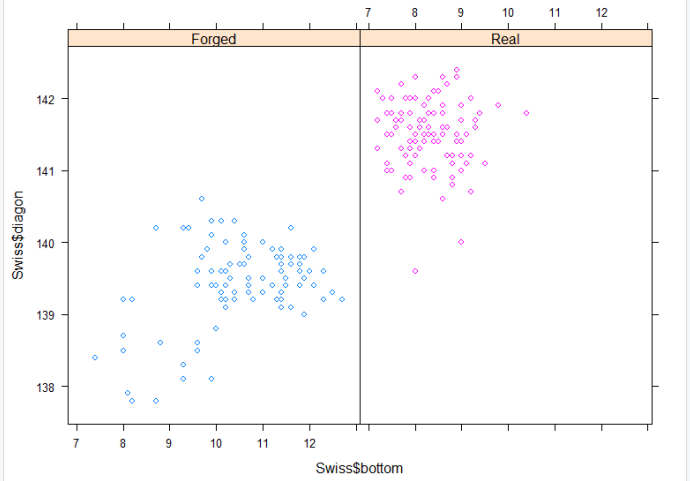


RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
  points(kodiak.crab$longit,kodiak.crab$latit,pch=".")
}
nr.pp = nrec/npots
crab.plot(lat,long,y=nr.pp,year=83,yname="Number of Recruits

detach(survey.crab)
detach(kodiak.crab)  # if you attach, be sure to detach when

attach(City)         # This command readies the data frame cit
```

```
> 
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
> 
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
> 
> pairs.image <- function(x) {
+   pairs(x,panel=function(x,y) {
+     foo <- bin2(cbind(x,y),nbin=c(75,75))
+     foo <- ash2(foo,m=c(6,6))
+     image(foo,add=T,xlab="",ylab="",col=topo.colors(1000))
+     points(x,y,pch=".")
+   })
+ }
> pairs.image(Swiss[,1:6])
> 
> pairs.persp <- function(x) {
+   par(bg="sky blue")
+   pairs(x,panel=function(x,y) {
```

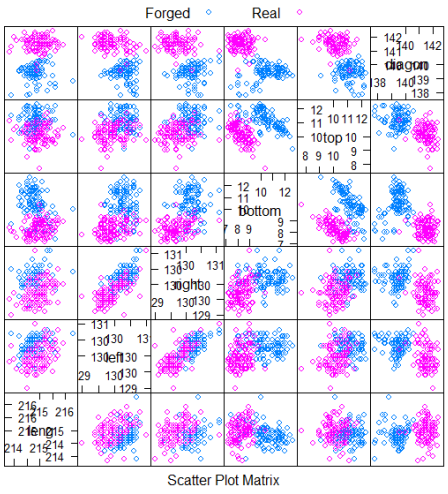


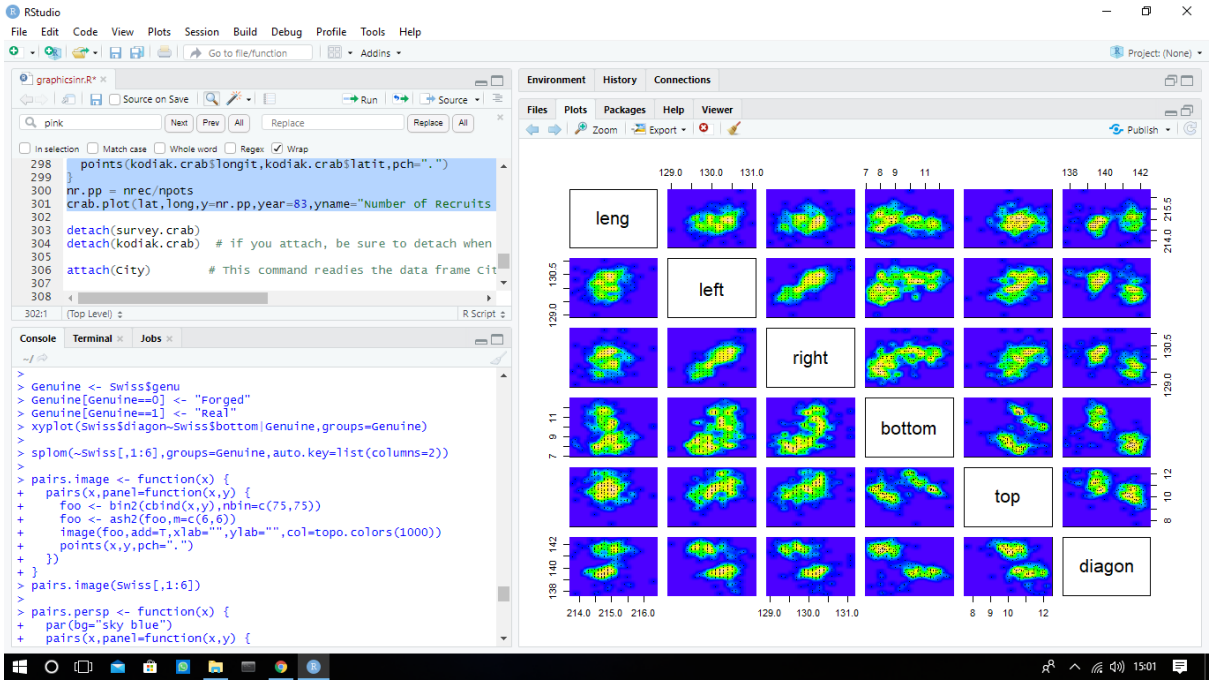
RStudio

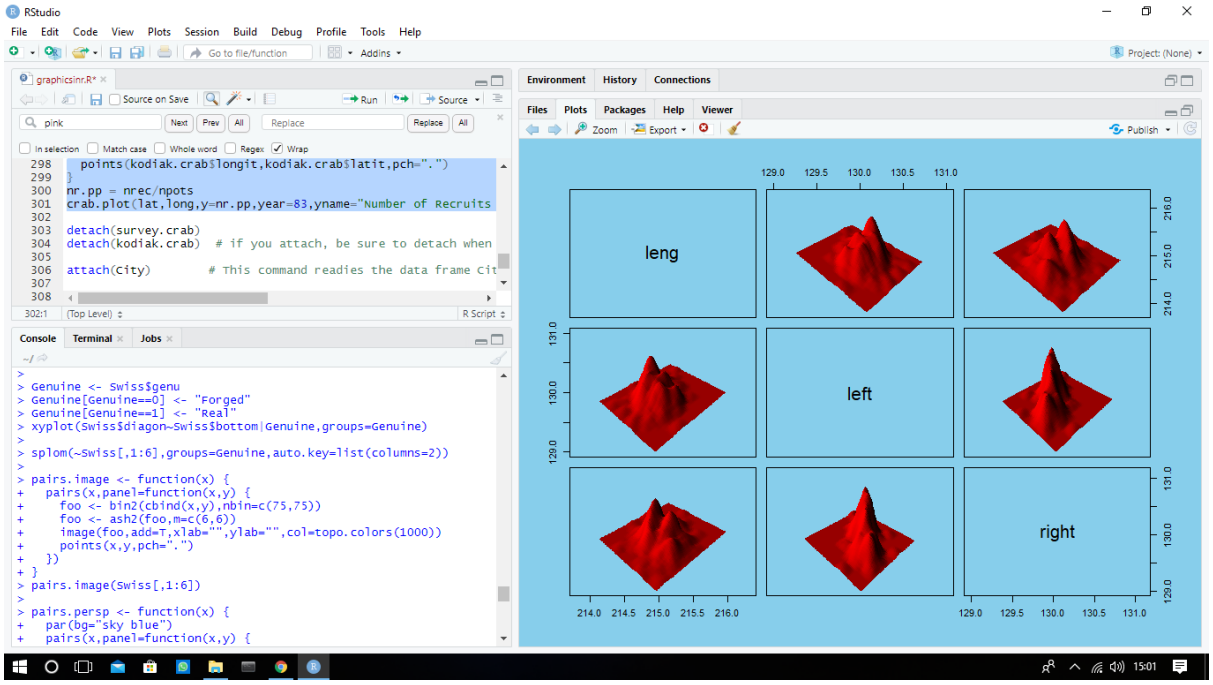
File Edit Code View Plots Session Build Debug Profile Tools Help

```
  points(kodiak.crab$longit,kodiak.crab$latit,pch=".")
}
nr.pp = nrec/npots
crab.plot(lat,long,y=nr.pp,year=83,yname="Number of Recruits

detach(survey.crab)
detach(kodiak.crab)  # if you attach, be sure to detach when

attach(City)         # This command readies the data frame cit
```

```
> 
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
> 
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
> 
> pairs.image <- function(x) {
+   pairs(x,panel=function(x,y) {
+     foo <- bin2(cbind(x,y),nbin=c(75,75))
+     foo <- ash2(foo,m=c(6,6))
+     image(foo,add=T,xlab="",ylab="",col=topo.colors(1000))
+     points(x,y,pch=".")
+   })
+ }
> pairs.image(Swiss[,1:6])
> 
> pairs.persp <- function(x) {
+   par(bg="sky blue")
+   pairs(x,panel=function(x,y) {
```




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R*
Source on Save
Run
Source
pink
Next Prev All
Replace
Replace All
In selection Match case Whole word Regex Wrap
298 points(kodiak.crab$longit,kodiak.crab$latit,pch=".")
299 }
300 nr.pp = nrec/npots
301 crab.plot(lat,long,y=nr.pp,year=83,yname="Number of Recruits
302
303 detach(survey.crab)
304 detach(kodiak.crab) # if you attach, be sure to detach when
305
306 attach(City) # This command readies the data frame cit
307
308
302:1 (Top Level) R Script
Console Terminal Jobs
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
> pairs.image <- function(x) {
+ pairs(x,panel=function(x,y) {
+ foo <- bin2(cbind(x,y),nbin=c(75,75))
+ foo <- ash2(foo,m=c(6,6))
+ image(foo,add=T,xlab="",ylab="",col=topo.colors(1000))
+ points(x,y,pch=".")
+ })
+ }
> pairs.image(Swiss[,1:6])
> pairs.persp <- function(x) {
+ par(bg="sky blue")
+ pairs(x,panel=function(x,y) {
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
leng
left
right
bottom
top
diagon
15:01

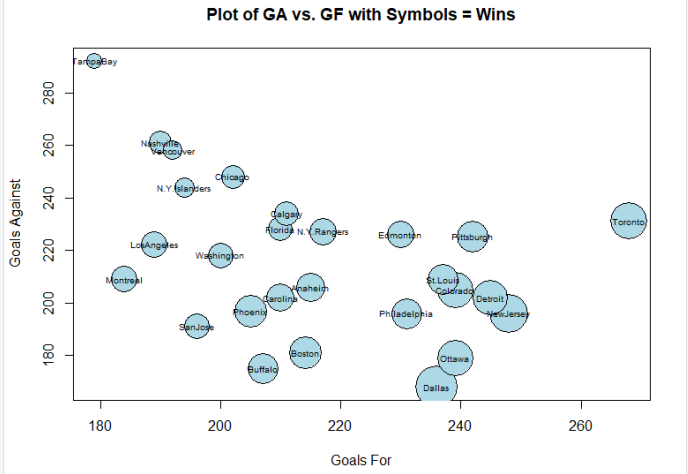

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsinr.R*
Source on Save
Run
Source
pink
Next Prev All
Replace
Replace All
In selection Match case Whole word Regex Wrap
298 points(kodiak.crab$longit,kodiak.crab$latit,pch=".")
299 }
300 nr.pp = nrec/npots
301 crab.plot(lat,long,y=nr.pp,year=83,yname="Number of Recruits
302
303 detach(survey.crab)
304 detach(kodiak.crab) # if you attach, be sure to detach when
305
306 attach(City) # This command readies the data frame cit
307
308
302:1 (Top Level) R Script
Console Terminal Jobs
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
> pairs.image <- function(x) {
+ pairs(x,panel=function(x,y) {
+ foo <- bin2(cbind(x,y),nbin=c(75,75))
+ foo <- ash2(foo,m=c(6,6))
+ image(foo,add=T,xlab="",ylab="",col=topo.colors(1000))
+ points(x,y,pch=".")
+ })
+ }
> pairs.image(Swiss[,1:6])
> pairs.persp <- function(x) {
+ par(bg="sky blue")
+ pairs(x,panel=function(x,y) {
Environment History Connections
Files Plots Packages Help Viewer
Zoom Export Publish
leng
left
right
15:01

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
298   points(kodiak.crab$longit,kodiak.crab$latit,pch=".")
299   }
300   nr.pp = nrec/npots
301   crab.plot(lat,long,y=nr.pp,year=83,yname="Number of Recruits
302
303   detach(survey.crab)
304   detach(kodiak.crab)  # if you attach, be sure to detach when
305
306   attach(City)         # This command readies the data frame cit
307
308
```

```
> Genuine <- Swiss$genu
> Genuine[Genuine==0] <- "Forged"
> Genuine[Genuine==1] <- "Real"
> xyplot(Swiss$diagon~Swiss$bottom|Genuine,groups=Genuine)
>
> splom(~Swiss[,1:6],groups=Genuine,auto.key=list(columns=2))
>
> pairs.image <- function(x) {
+   pairs(x,panel=function(x,y) {
+     foo <- bin2(cbind(x,y),nbin=c(75,75))
+     foo <- ash2(foo,m=c(6,6))
+     image(foo,add=T,xlab="",ylab="",col=topo.colors(1000))
+     points(x,y,pch=".")
+   })
+ }
> pairs.image(Swiss[,1:6])
>
> pairs.persp <- function(x) {
+   par(bg="sky blue")
+   pairs(x,panel=function(x,y) {
```

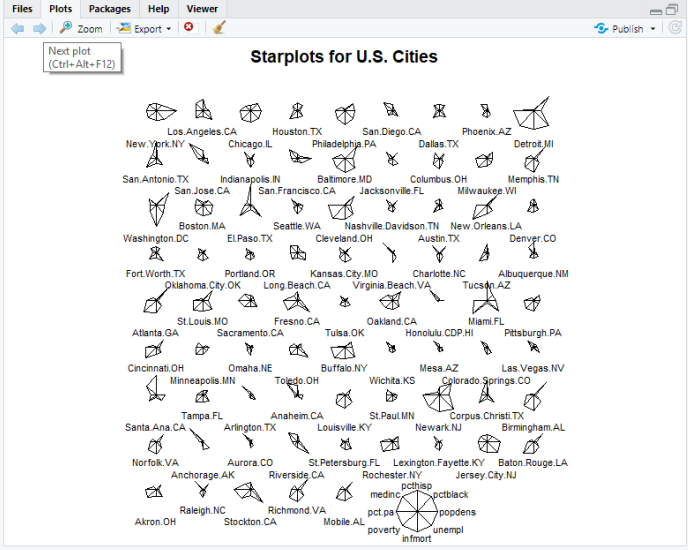


RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

	pop90	pop80	growth	area	popdens	blackpop	hispc
New.York.NY	7311966	7071639	3.4	309	23671	2102512	1
Los.Angeles.CA	3489779	2968528	17.6	469	7436	487674	1
Chicago.IL	2768483	3005072	-7.9	227	12185	1087711	5
Houston.TX	1690180	1595138	6.0	540	3131	457990	4
Philadelphia.PA	1552572	1688210	-8.0	135	11492	631936	8
San.Diego.CA	1148851	875538	31.2	324	3546	104261	2
Dallas.TX	1022497	904599	13.0	342	2986	296994	2
Phoenix.AZ	1012230	789704	28.2	420	2411	51053	1
Detroit.MI	1012110	1203368	-15.9	139	7297	777916	2
San.Antonio.TX	966437	785940	23.0	333	2902	65884	5
San.Jose.CA	801331	629400	27.3	171	4678	36790	2
Indianapolis.IN	746538	700807	6.5	362	2064	165570	7
San.Francisco.CA	728921	678974	7.4	47	15609	79039	1
Baltimore.MD	726096	786741	-7.7	81	8986	435768	7
Jacksonville.FL	661177	540920	22.2	759	871	160283	1
Columbus.OH	642987	565021	13.8	191	3368	142748	6
Milwaukee.WI	617043	636297	-3.0	96	6421	191255	3
Memphis.TN	610275	646174	-5.6	256	2384	334737	4
Washington.DC	585221	638432	-8.3	61	9531	399604	3

Showing 1 to 23 of 77 entries, 72 total columns

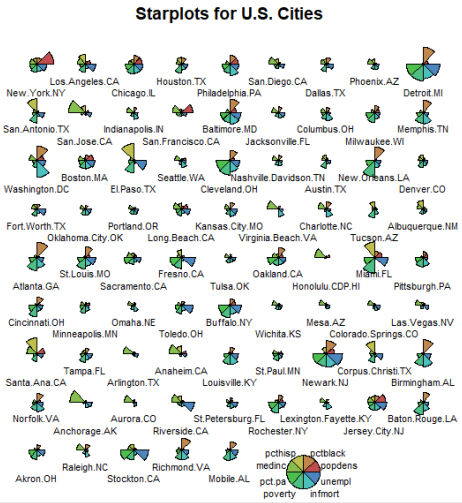


RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
318 palette("default")
319
320 library(MASS)
321
322 parcoord(city.mat)
323 names(olives)
324 olive.mat = olives[,3:9]
325 palette(rainbow(12,s=0.6,v=0.75))
326 parcoord(olive.mat,col=as.numeric(olives$Area.Name))
327
328
329
330
```

```
[66] "ctrend"    "zeroday"   "freeze"    "hot"       "ddays"
[71] "high.jul"  "low.jan"
>
> view(City)
>
> city.mat <-cbind(popdens,pctblack,pcthisp,medinc,pct.pa,poverty,i
nfmort,unempl)
> stars(city.mat, key.loc = c(15,1.25),main = "Starplots for U.S. C
ities",label=row.names(City),cex=.7)
>
> palette(rainbow(12, s = 0.6, v = 0.75))
> stars(city.mat,len=.80,key.loc = c(15,1.25),main = "Starplots for
 U.S. Cities",draw.segments = TRUE,label=row.names(City),cex=.7)
>
> palette("default")
>
> library(MASS)
>
> parcoord(city.mat)
> names(olives)
 [1] "Region.Name" "Area.Name"   "Palmitic"    "Palmitoleic"
```

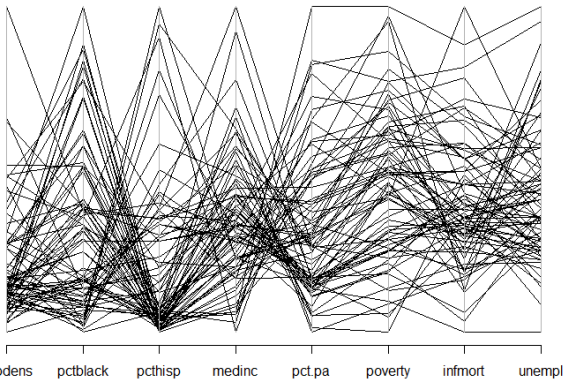


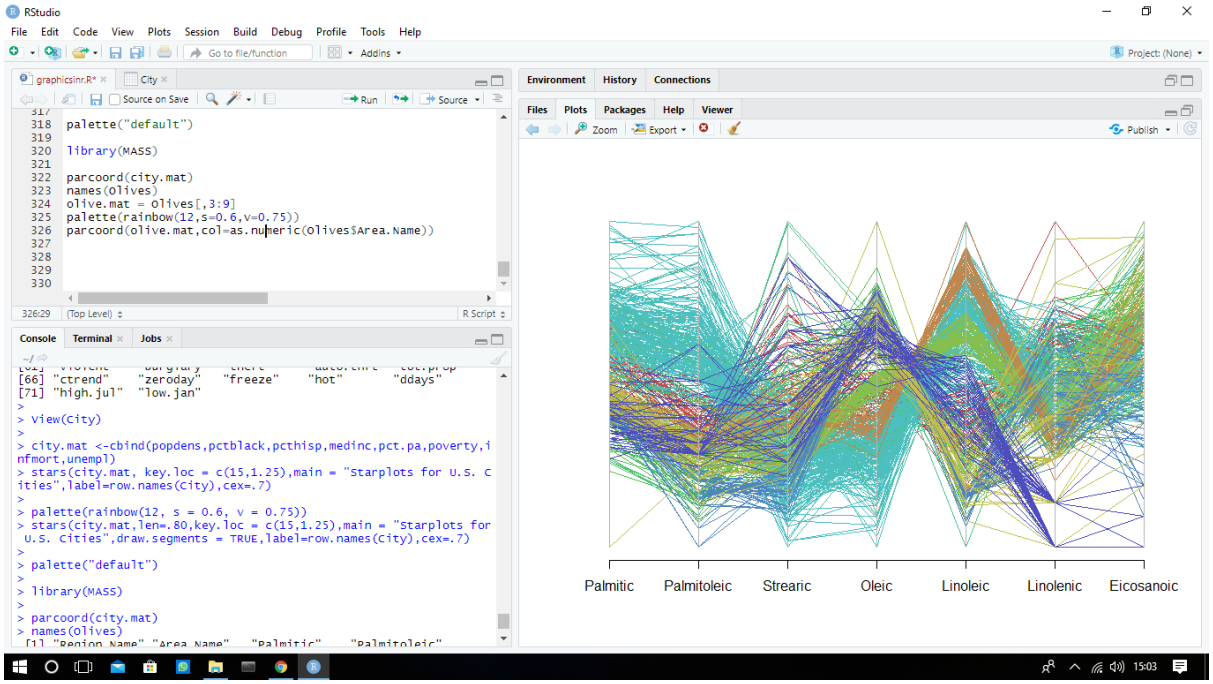
RStudio

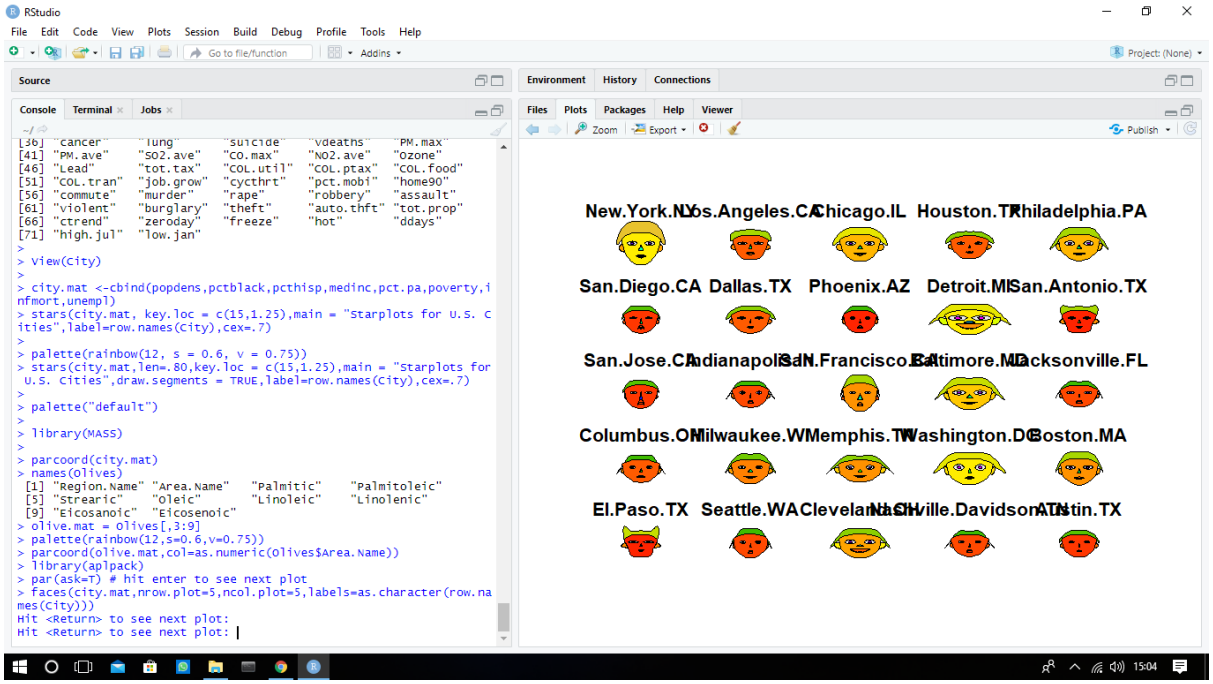
File Edit Code View Plots Session Build Debug Profile Tools Help

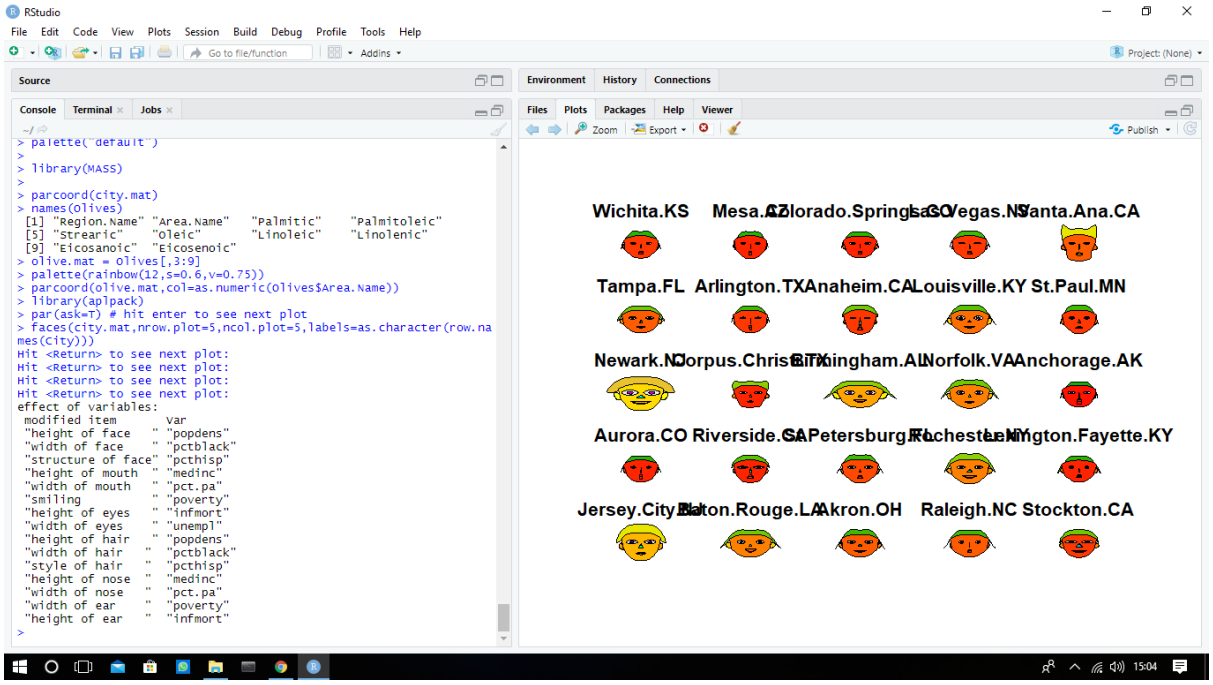
```
318 palette("default")
319
320 library(MASS)
321
322 parcoord(city.mat)
323 names(olives)
324 olive.mat = olives[,3:9]
325 palette(rainbow(12,s=0.6,v=0.75))
326 parcoord(olive.mat,col=as.numeric(olives$Area.Name))
327
328
329
330
```

```
[66] "ctrend"    "zeroday"   "freeze"    "hot"       "ddays"
[71] "high.jul"  "low.jan"
>
> view(City)
>
> city.mat <-cbind(popdens,pctblack,pcthisp,medinc,pct.pa,poverty,i
nfmort,unempl)
> stars(city.mat, key.loc = c(15,1.25),main = "Starplots for U.S. C
ities",label=row.names(City),cex=.7)
>
> palette(rainbow(12, s = 0.6, v = 0.75))
> stars(city.mat,len=.80,key.loc = c(15,1.25),main = "Starplots for
 U.S. Cities",draw.segments = TRUE,label=row.names(City),cex=.7)
>
> palette("default")
>
> library(MASS)
>
> parcoord(city.mat)
> names(olives)
 [1] "Region.Name" "Area.Name"   "Palmitic"    "Palmitoleic"
```




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
graphicsInr.R*
City
Source on Save
Run
Source
palette("default")
library(MASS)
parcoord(city.mat)
names(olives)
olive.mat = olives[,3:9]
palette(rainbow(12,s=0.6,v=0.75))
parcoord(olive.mat,col=as.numeric(olives$Area.Name))
Console
Terminal
Jobs
[66] "ctrend" "zeroday" "freeze" "hot" "ddays"
[71] "high.jul" "low.jan"
> view(City)
> city.mat <-cbind(popdens,pctblack,pcthisp,medinc,pct.pa,poverty,infmort,unempl)
> stars(city.mat, key.loc = c(15,1.25),main = "Starplots for U.S. Cities",label=row.names(City),cex=.7)
> palette(rainbow(12, s = 0.6, v = 0.75))
> stars(city.mat,len=.80,key.loc = c(15,1.25),main = "Starplots for U.S. Cities",draw.segments = TRUE,label=row.names(City),cex=.7)
> palette("default")
> library(MASS)
> parcoord(city.mat)
> names(olives)
Environment
History
Connections
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Palmitic
Palmitoleic
Strearic
Oleic
Linoleic
Linolenic
Eicosanoic





Conclusion : Thus we have studied basic and advanced graphics in R.

Experiment No : 9

Aim: Basic of correlation, simple and multiple regression in R

Theory:

Correlation and linear regression each explore the relationship between two quantitative variables. Both are very common analyses.

Correlation determines if one variable varies systematically as another variable changes. It does not specify that one variable is the dependent variable and the other is the independent variable. Often, it is useful to look at which variables are correlated to others in a data set, and it is especially useful to see which variables correlate to a particular variable of interest.

The three forms of correlation presented here are Pearson, Kendall, and Spearman. The test determining the *p*-value for Pearson correlation is a parametric test that assumes that data are bivariate normal. Kendall and Spearman correlation use nonparametric tests.

In contrast, linear regression specifies one variable as the independent variable and another as the dependent variable. The resultant model relates the variables with a linear relationship.

The tests associated with linear regression are parametric and assume normality, homoscedasticity, and independence of residuals, as well as a linear relationship between the two variables.

Appropriate data

- For Pearson correlation, two interval/ratio variables. Together the data in the variables are bivariate normal. The relationship between the two variables is linear. Outliers can detrimentally affect results.

- For Kendall correlation, two variables of interval/ratio or ordinal type.

- For Spearman correlation, two variables of interval/ratio or ordinal type.

- For linear regression, two interval/ratio variables. The relationship between the two variables is linear. Residuals are normal, independent, and homoscedastic. Outliers can affect the results unless robust methods are used.

Hypotheses

- For correlation, null hypothesis: The correlation coefficient (*r*, *tau*, or *rho*) for the sampled population is zero. Or, there is no correlation between the two variables.

- For linear regression, null hypothesis: The slope of the fit line for the sampled population is zero. Or, there is no linear relationship between the two variables.

Interpretation

- For correlation, reporting significant results as "Variable A was significantly correlated to Variable B" is acceptable. Alternatively, "A significant correlation between Variable A and Variable B was found." Or, "Variables A, B, and C were significantly correlated."

- For linear regression, reporting significant results as “Variable A was significantly linearly related to Variable B” is acceptable. Alternatively, “A significant linear regression between Variable A and Variable B was found.”

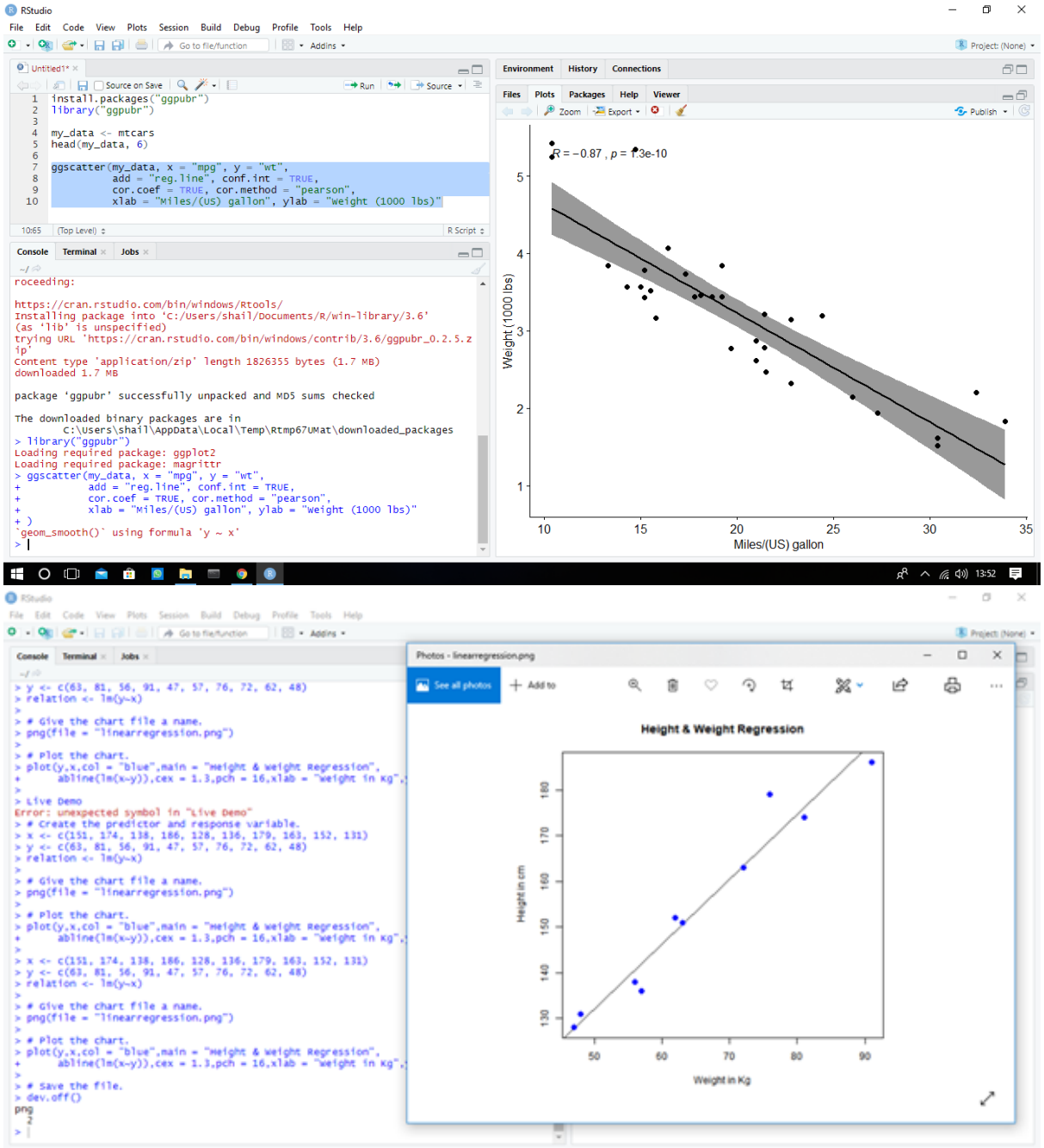
Output:

Code:

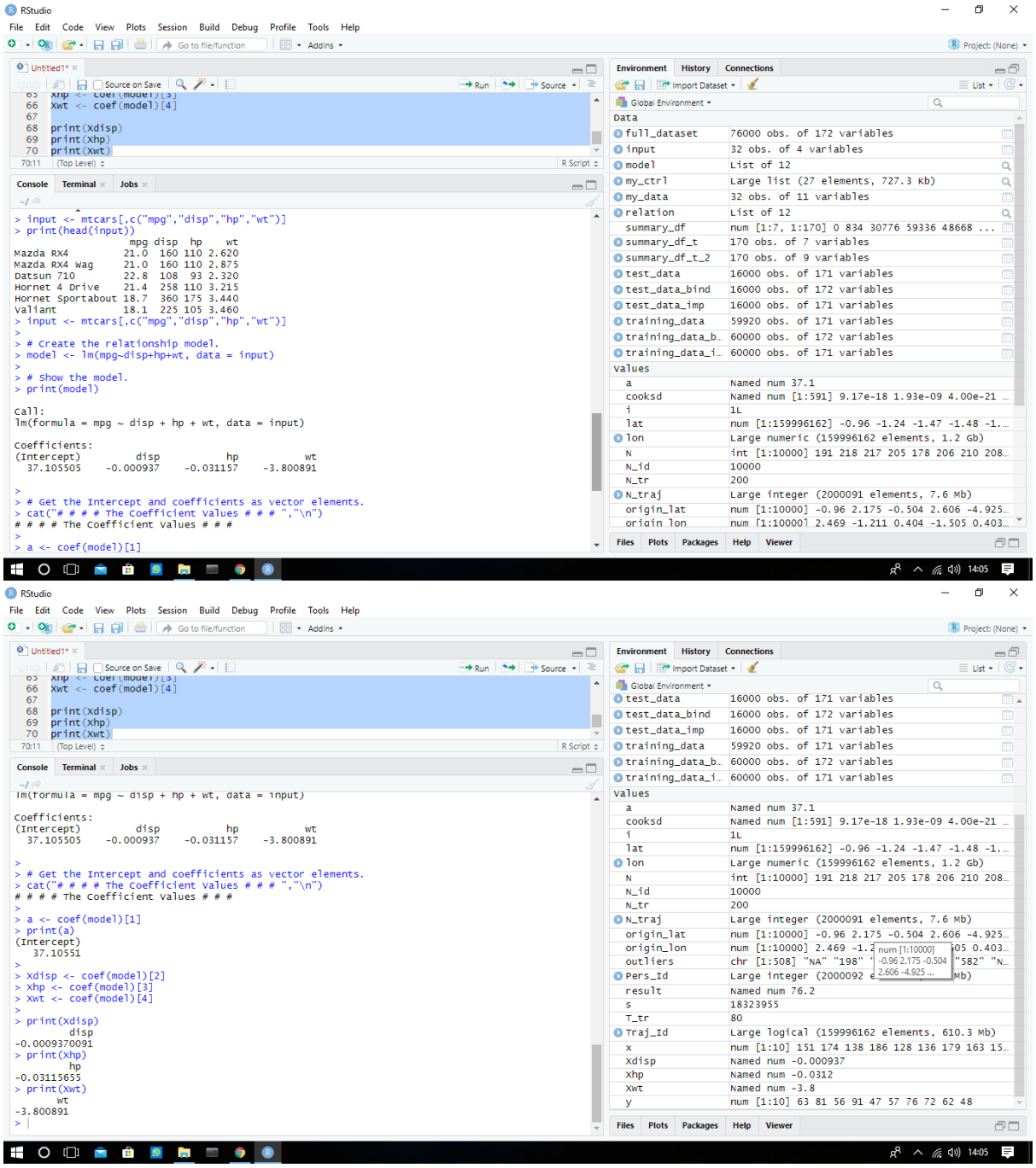
```
install.packages("ggpubr")
library("ggpubr")

my_data <- mtcars
head(my_data, 6)

ggscatter(my_data, x = "mpg", y = "wt",
        add = "reg.line", conf.int = TRUE,
        cor.coef = TRUE, cor.method = "pearson",
        xlab = "Miles/(US) gallon", ylab = "Weight (1000 lbs)")

x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

print(relation)

x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

print(summary(relation))


# The predictor vector.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

# The resposne vector.
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

# Find the weight of a person with height 170.
a <- data.frame(x = 170)
result <-  predict(relation,a)
print(result)
```

```
# Create the predictor and response variable.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
relation <- lm(y~x)

# Give the chart file a name.
png(file = "linearregression.png")

# Plot the chart.
plot(y,x,col = "blue",main = "Height & Weight Regression",
    abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")

# Save the file.
dev.off()

input <- mtcars[,c("mpg","disp","hp","wt")]
print(head(input))
input <- mtcars[,c("mpg","disp","hp","wt")]

# Create the relationship model.
model <- lm(mpg~disp+hp+wt, data = input)

# Show the model.
print(model)

# Get the Intercept and coefficients as vector elements.
cat("# # # # The Coefficient Values # # # ","\n")

a <- coef(model)[1]
print(a)

Xdisp <- coef(model)[2]
Xhp <- coef(model)[3]
Xwt <- coef(model)[4]

print(Xdisp)
print(Xhp)
print(Xwt)
```

Screenshot:


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
1 install.packages("ggpubr")
2 library("ggpubr")
4 my_data <- mtcars
5 head(my_data, 6)
7 ggscatter(my_data, x = "mpg", y = "wt",
8 add = "reg.line", conf.int = TRUE,
9 cor.coef = TRUE, cor.method = "pearson",
10 xlab = "Miles/(US) gallon", ylab = "Weight (1000 lbs)"
Console Terminal Jobs
roceeding:
https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/shail/Documents/R/win-library/3.6'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/ggpubr_0.2.5.zip'
Content type 'application/zip' length 1826355 bytes (1.7 MB)
downloaded 1.7 MB
package 'ggpubr' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\shail\AppData\Local\Temp\Rtmp67UMat\downloaded_packages
> library("ggpubr")
Loading required package: ggplot2
Loading required package: magrittr
> ggscatter(my_data, x = "mpg", y = "wt",
+ add = "reg.line", conf.int = TRUE,
+ cor.coef = TRUE, cor.method = "pearson",
+ xlab = "Miles/(US) gallon", ylab = "Weight (1000 lbs)"
+)
`geom_smooth()` using formula 'y ~ x'
Environment History Connections
Files Plots Packages Help Viewer
R = −0.87 , p = 1.3e-10
Weight (1000 lbs)
Miles/(US) gallon
13:52

> y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
> relation <- lm(y~x)
> # Give the chart file a name.
> png(file = "linearregression.png")
> # Plot the chart.
> plot(y,x,col = "blue",main = "Height & Weight Regression",
+ abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",
> Live Demo
Error: unexpected symbol in "Live Demo"
> # Create the predictor and response variable.
> x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
> y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
> relation <- lm(y~x)
> # Give the chart file a name.
> png(file = "linearregression.png")
> # Plot the chart.
> plot(y,x,col = "blue",main = "Height & Weight Regression",
+ abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",
> # Save the file.
> dev.off()
Photos - linearregression.png
See all photos
Add to
Height & Weight Regression
Height in cm
Weight in Kg



Conclusion: Thus, we have studied the basics of correlation, simple and multiple regression in R.

Experiment No: 10

Aim: Clustering in R - k-means

Theory:

K Means Clustering is an unsupervised learning algorithm that tries to cluster data based on their similarity. Unsupervised learning means that there is no outcome to be predicted, and the algorithm just tries to find patterns in the data. In k means clustering, we have to specify the number of clusters we want the data to be grouped into. The algorithm randomly assigns each observation to a cluster and finds the centroid of each cluster. Then, the algorithm iterates through two steps:

- Reassign data points to the cluster whose centroid is closest.
- Calculate the new centroid of each cluster.

These two steps are repeated until the within-cluster variation cannot be reduced any further. The within-cluster variation is calculated as the sum of the euclidean distance between the data points and their respective cluster centroids.

Clustering Types:

Clustering can be broadly divided into two subgroups:

- Hard clustering: in hard clustering, each data object or point either belongs to a cluster completely or not. For example in the Uber dataset, each location belongs to either one borough or the other.
- Soft clustering: in soft clustering, a data point can belong to more than one cluster with some probability or likelihood value. For example, you could identify some locations as the border points belonging to two or more boroughs.

The clustering algorithm that we are going to use is the K-means algorithm, which we can find in the package stats. The K-means algorithm accepts two parameters as input:

- The data;
- A **K** value, which is the number of groups that we want to create.

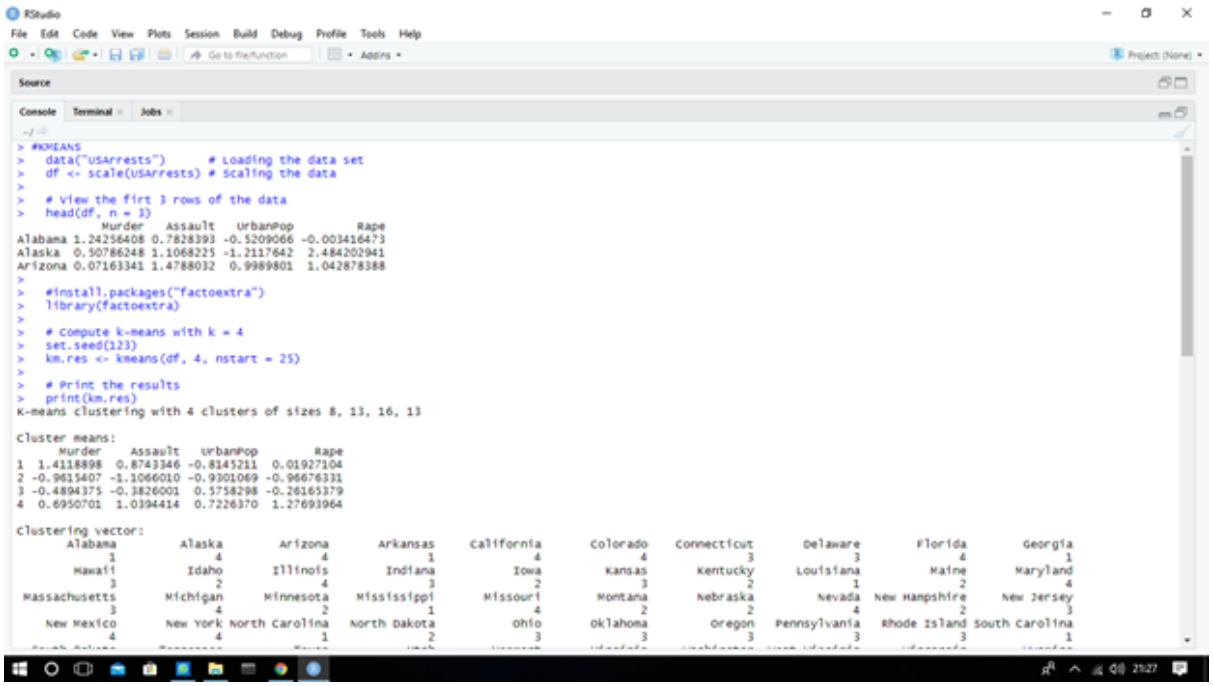
Conceptually, the K-means behaves as follows:

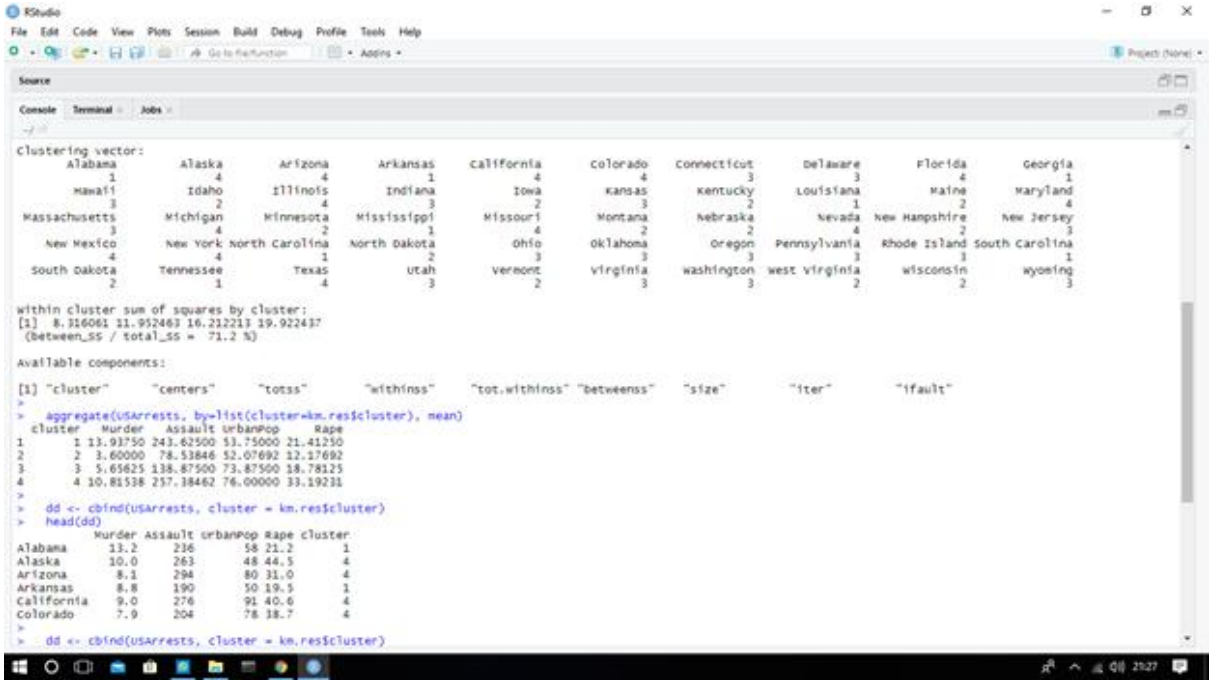
1. It chooses K centroids randomly;
2. Matches each point in the data (in our case, each mammal) with the closest centroid in an n-dimensional space where n is the number of features used in the clustering (in our example, 5 features — water, protein, fat, lactose, ash). After this step, each point belongs to a group.
3. Now, it recalculates the centroids as being the mean point (vector) of all other points in the group.
4. It keeps repeating the steps 2 and 3 until either when the groups are stabilized, that is, when no points are reallocated to another centroid or when it reaches the maximum number of iterations (the stats library uses 10 as default).

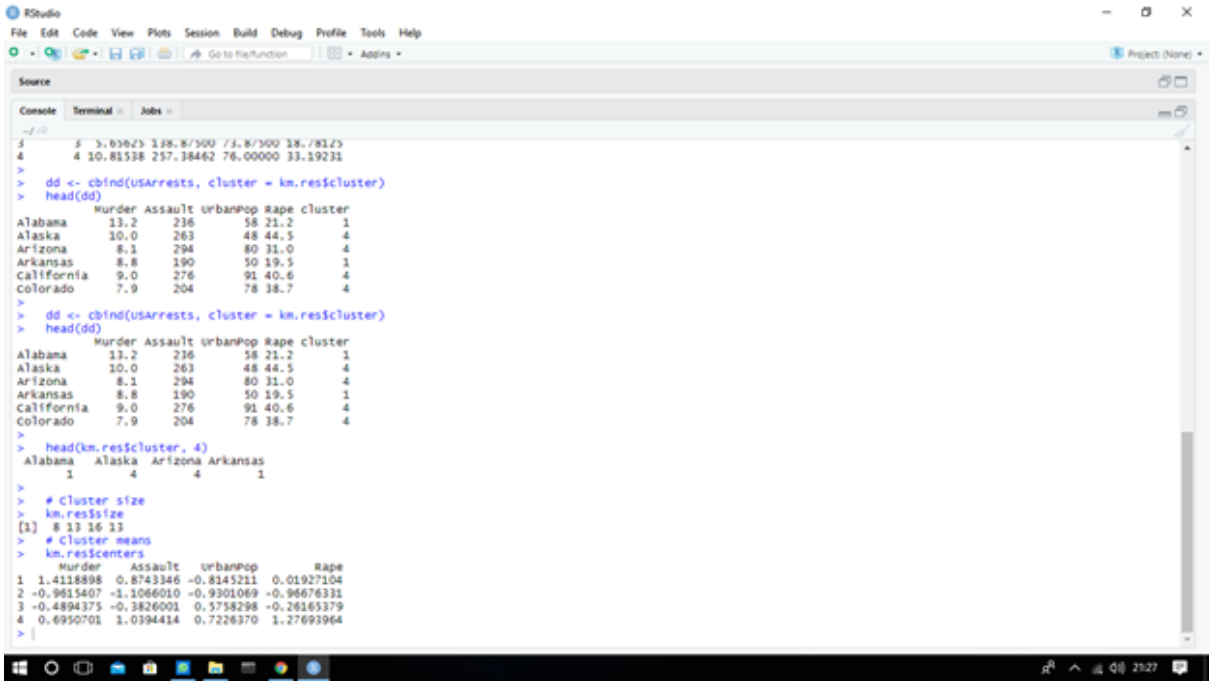
Output :

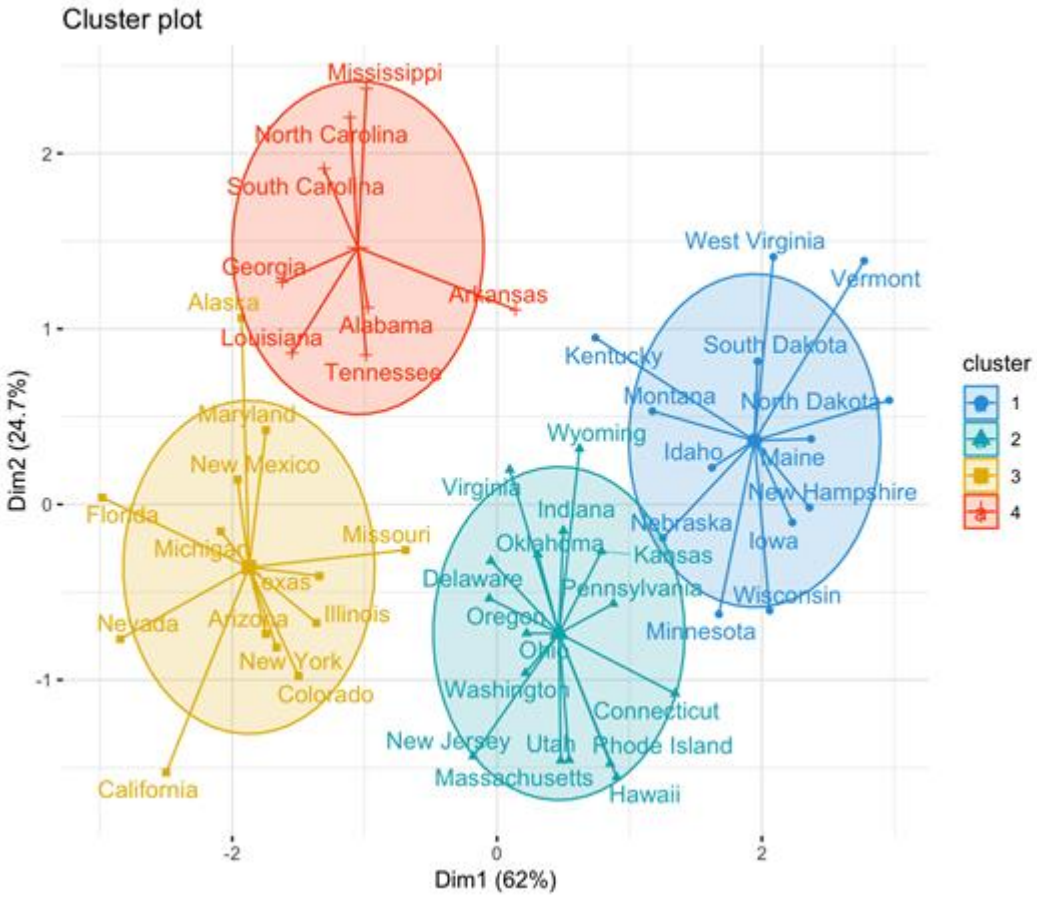
Code :

Screenshots :




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
Clustering vector:
Alabama Alaska Arizona Arkansas California Colorado Connecticut Delaware Florida Georgia
1 4 4 1 4 4 3 3 4 1
Hawaii Idaho Illinois Indiana Iowa Kansas Kentucky Louisiana Maine Maryland
3 2 4 3 2 3 2 1 2 4
Massachusetts Michigan Minnesota Mississippi Missouri Montana Nebraska Nevada New Hampshire New Jersey
3 4 2 1 4 2 2 4 2 3
New Mexico New York North Carolina North Dakota Ohio Oklahoma Oregon Pennsylvania Rhode Island South Carolina
4 4 1 2 3 3 3 3 1 1
South Dakota Tennessee Texas Utah Vermont Virginia Washington West Virginia Wisconsin Wyoming
2 1 4 3 2 3 3 2 2 3
Within cluster sum of squares by cluster:
[1] 8.316061 11.952463 16.212213 19.922437
(between_SS / total_SS = 71.2 %)
Available components:
[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size" "iter" "ifault"
> aggregate(USArrests, by=list(cluster=km.res$cluster), mean)
cluster Murder Assault UrbanPop Rape
1 1 13.93750 243.62500 53.75000 21.41250
2 2 3.60000 78.53846 52.07692 12.17692
3 3 5.65625 138.87500 73.87500 18.78125
4 4 10.81538 257.38462 76.00000 33.19231
> dd <- cbind(USArrests, cluster = km.res$cluster)
> head(dd)
Murder Assault UrbanPop Rape cluster
Alabama 13.2 236 58 21.2 1
Alaska 10.0 263 48 44.5 4
Arizona 8.1 294 80 31.0 4
Arkansas 8.8 190 50 19.5 1
California 9.0 276 91 40.6 4
Colorado 7.9 204 78 38.7 4
> dd <- cbind(USArrests, cluster = km.res$cluster)


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
4 4 10.81538 257.38462 76.00000 33.19231
> dd <- cbind(USArrests, cluster = km.res$cluster)
> head(dd)
Murder Assault UrbanPop Rape cluster
Alabama 13.2 236 58 21.2 1
Alaska 10.0 263 48 44.5 4
Arizona 8.1 294 80 31.0 4
Arkansas 8.8 190 50 19.5 1
California 9.0 276 91 40.6 4
Colorado 7.9 204 78 38.7 4
> dd <- cbind(USArrests, cluster = km.res$cluster)
> head(dd)
Murder Assault UrbanPop Rape cluster
Alabama 13.2 236 58 21.2 1
Alaska 10.0 263 48 44.5 4
Arizona 8.1 294 80 31.0 4
Arkansas 8.8 190 50 19.5 1
California 9.0 276 91 40.6 4
Colorado 7.9 204 78 38.7 4
> head(km.res$cluster, 4)
Alabama Alaska Arizona Arkansas
1 4 4 1
> # Cluster size
> km.res$size
[1] 8 13 16 13
> # Cluster means
> km.res$centers
Murder Assault UrbanPop Rape
1 1.4118898 0.8743346 -0.8145211 0.01927104
2 -0.9615407 -1.1066010 -0.9301069 -0.96676331
3 -0.4894375 -0.3826001 0.5758298 -0.26165379
4 0.6950701 1.0394414 0.7226370 1.27693964



Conclusion : This, we have studied clustering in R - k-means

Experiment No. 11

Aim : Classification on large and noisy dataset with R - logistic regression and naive bayes.

Theory :

In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

R makes it very easy to fit a logistic regression model. The function to be called is glm() and the fitting process is not so different from the one used in linear regression. In this post I am going to fit a binary logistic regression model and explain each step.

Naïve Bayes classification is a kind of simple probabilistic classification method based on Bayes' theorem with the assumption of independence between features. The model is trained on a training dataset to make predictions by predict() function. This article introduces two functions naiveBayes() and train() for the performance of Naïve Bayes classification.

Output:

Code:

```
training_data <- read.csv("aps_failure_training_set.csv", skip = 20,na.strings = "na")
test_data <- read.csv("aps_failure_test_set.csv",skip = 20, na.strings = "na")

dim(training_data)
dim(test_data)

library(dplyr)
library(caret)
library(caretEnsemble)
library(mice)
library(doParallel)
library(car)

install.packages("car")

glimpse(training_data)
```

```
glimpse(test_data)

summary(training_data$class)
summary(test_data$class)

options(digits = 2)
prop.table(table(training_data$class))
prop.table(table(test_data$class))

options(scipen = 999)
summary_df <- do.call(cbind, lapply(training_data[,
                                    2:ncol(training_data)], summary))
summary_df_t <- as.data.frame(round(t(summary_df),0))
names(summary_df_t)[7] <- paste("Missing_values")
summary_df_t_2 <- summary_df_t %>%
  mutate(obs = nrow(training_data),
         Missing_prop = Missing_values / obs)
print(summary_df_t_2)

summary_df_t_2 %>% summarise(Min = mean(Min.),
                        first_Q = mean(`1st Qu.`),
                        Median = median(Median),
                        Mean = mean(Mean),
                        third_Q = mean(`3rd Qu.`),
                        Max = max(Max.),
                        mean_MV = mean(Missing_values),
                        obs = mean(obs),
                        mean_MV_perc = mean_MV / obs)

#replicate our sets
training_data_bind <- training_data
test_data_bind <- test_data
#create a new column "set" to label the observations
training_data_bind$set <- "TRAIN"
test_data_bind$set <- "TEST"
#merge them into 1 single set
full_dataset <- rbind(training_data_bind, test_data_bind)
dim(full_dataset)

set.seed(123)
imputed_full <- mice(full_dataset,
                     m=1,
                     maxit = 5,
                     method = "pmm",
                     seed = 500)

full_imputed <- complete(imputed_full, 1)
dim(full_imputed)

#subset the full_imputed_filtered dataset
```

```
training_data_imp <- subset(training_data)
test_data_imp <- subset(test_data)
#drop the "set" column, we don't need it anymore
training_data_imp$set <- NULL
test_data_imp$set <- NULL
#check dimensions
dim(training_data_imp)
dim(test_data_imp)

install.packages("DMwR")
library(DMwR)
knnImputation(training_data, k = 10, scale = T, meth = "weighAvg",distData = NULL)

cooksd <- cooks.distance(glm(class ~ .,
                        family = "binomial",
                        data = training_data_imp))

plot(cooksd,
    pch="*",
    cex=2,
    main="Influential Obs by Cooks distance")
abline(h = 4*mean(cooksd, na.rm=T), col="red")

outliers <- rownames(training_data_imp[cooksd > 4*mean(cooksd, na.rm=T), ])
print(outliers)

sum((correlation > 0.5 | correlation < -0.5) & correlation < 1) / (162*162)
sum((correlation > 0.7 | correlation < -0.7) & correlation < 1) / (162*162)
sum((correlation > 0.9 | correlation < -0.9) & correlation < 1) / (162*162)

registerDoParallel(3)
getDoParWorkers()
set.seed(123)
my_ctrl <- trainControl(method = "cv",
                    number = 5,
                    classProbs = TRUE,
                    savePredictions = "final",
                    na.action = na.pass,
                    index = createResample(training_data_imp$class, 3),
                    sampling = "up",
                    allowParallel = TRUE)

model_list <- caretList(class ~ .,
                    data = training_data_imp,
                    methodList = c("glm", "nb"),
                    metric = "Kappa",
                    tuneList = NULL,
                    na.action = na.pass,
                    continue_on_fail = FALSE,
                    preProcess = c("center", "scale"),
```

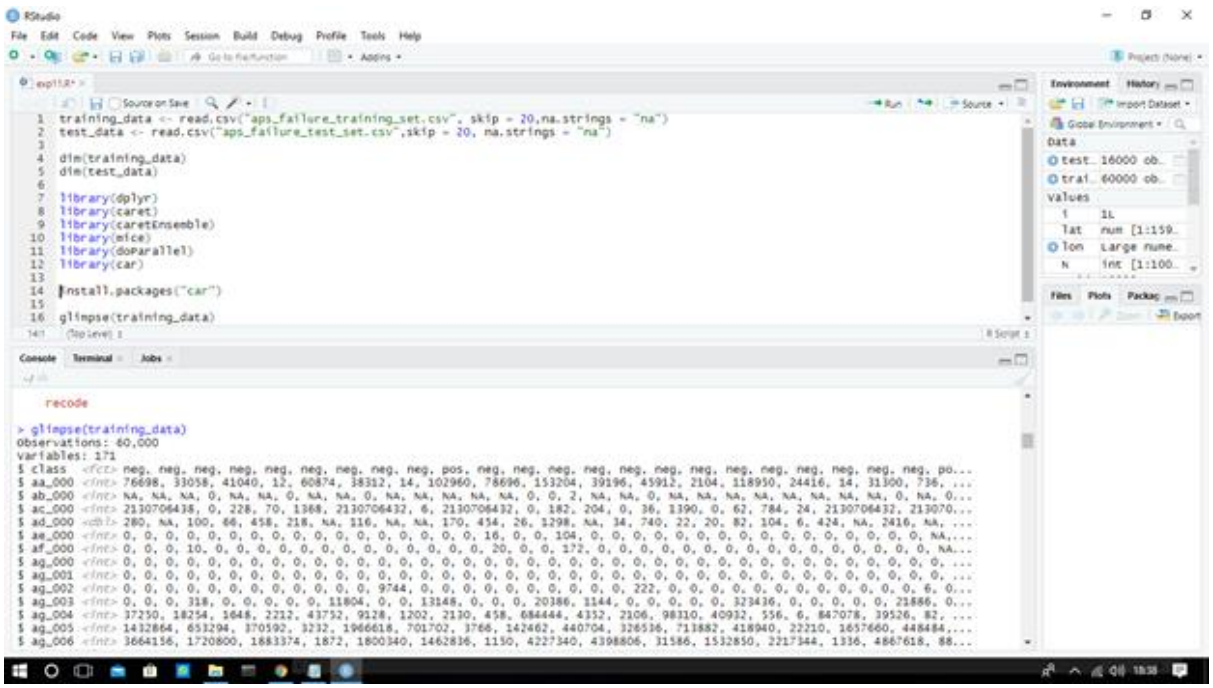
trControl = my_ctrl)

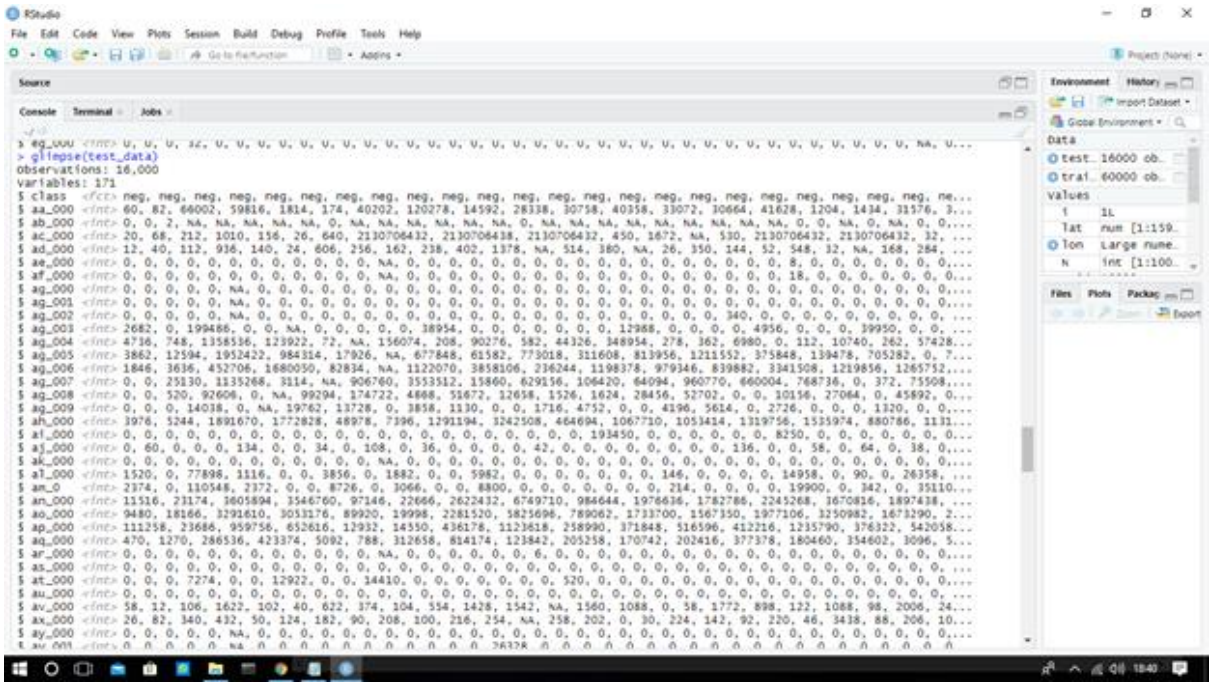
model_list$glm
model_list$nb

Screenshots:

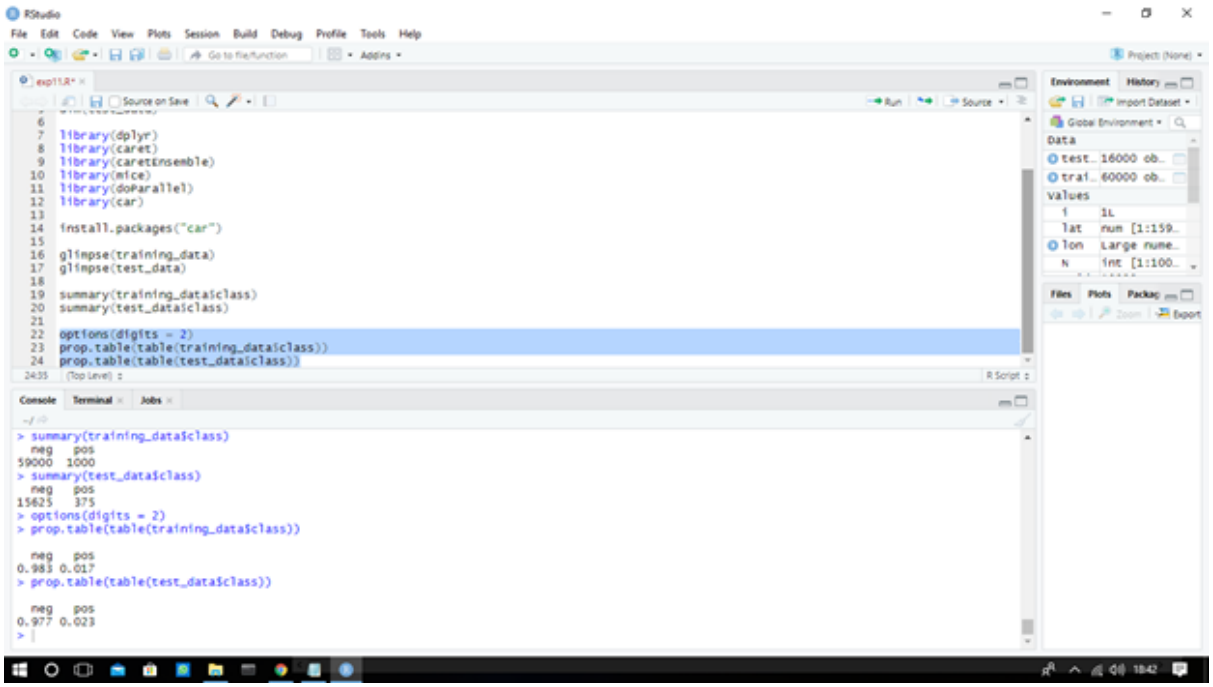
STEPS:

1. Download the dataset from https://archive.ics.uci.edu/ml/machine-learning-databases/00421/
2. Load the data set and View Data (str or dplyr's glimpse)

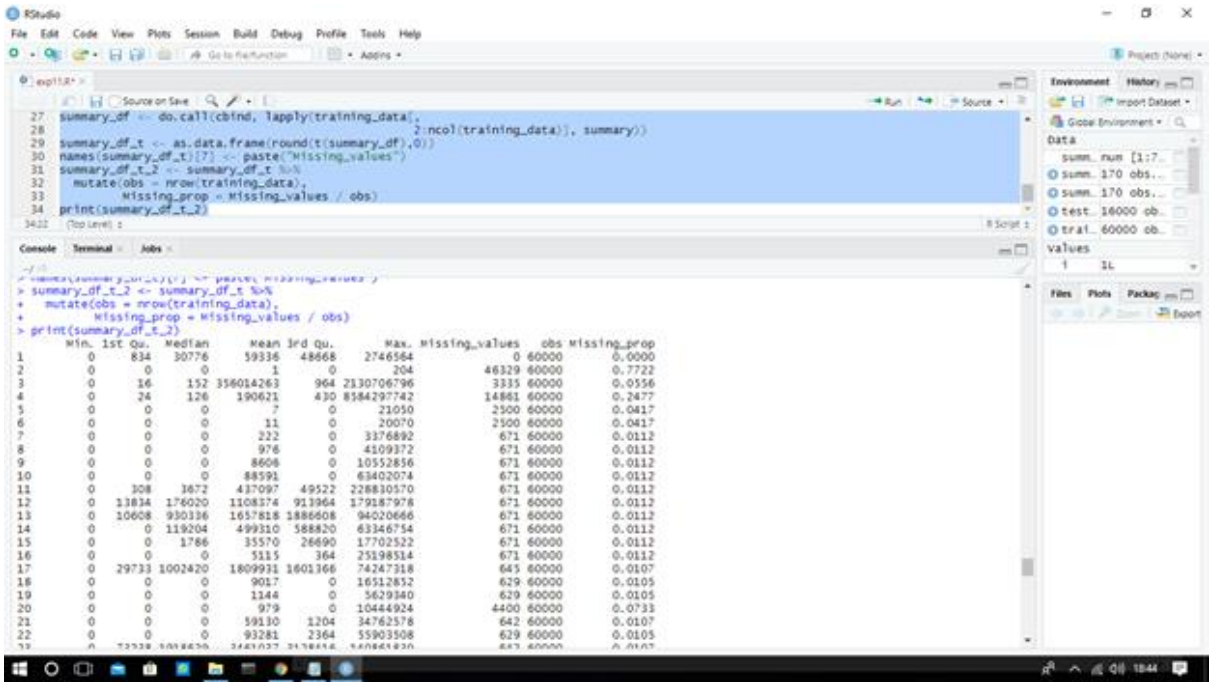


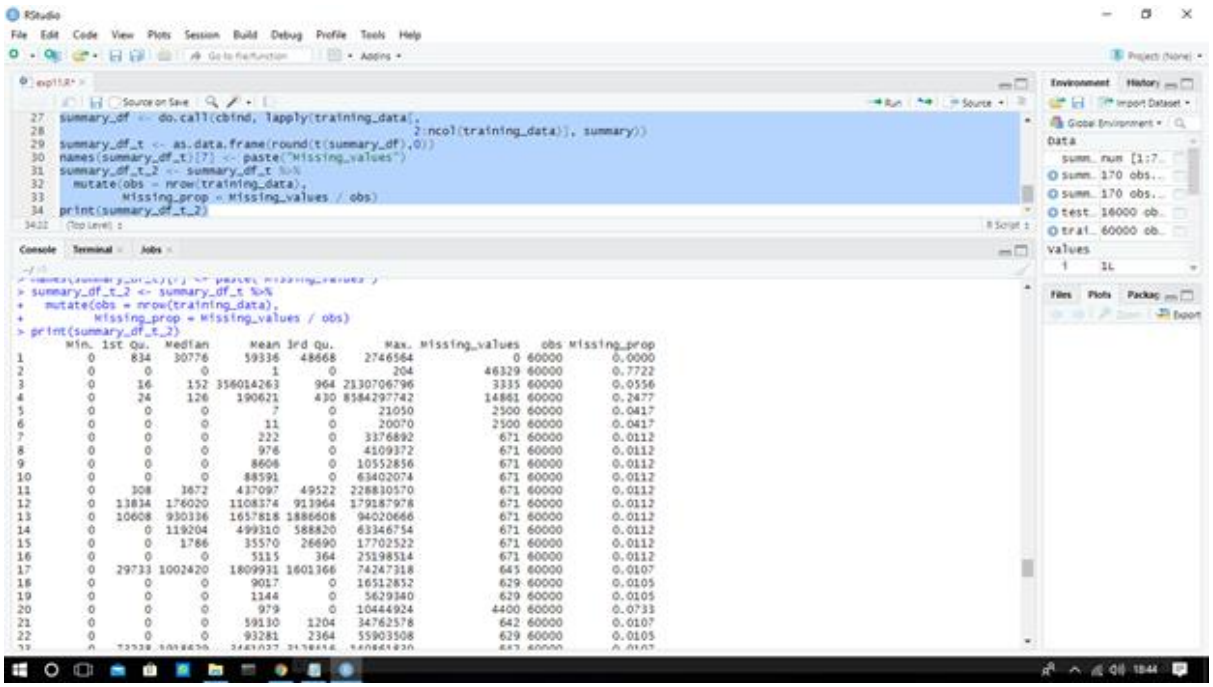


3. Check How many neg and pos do we have in each set and check proportions

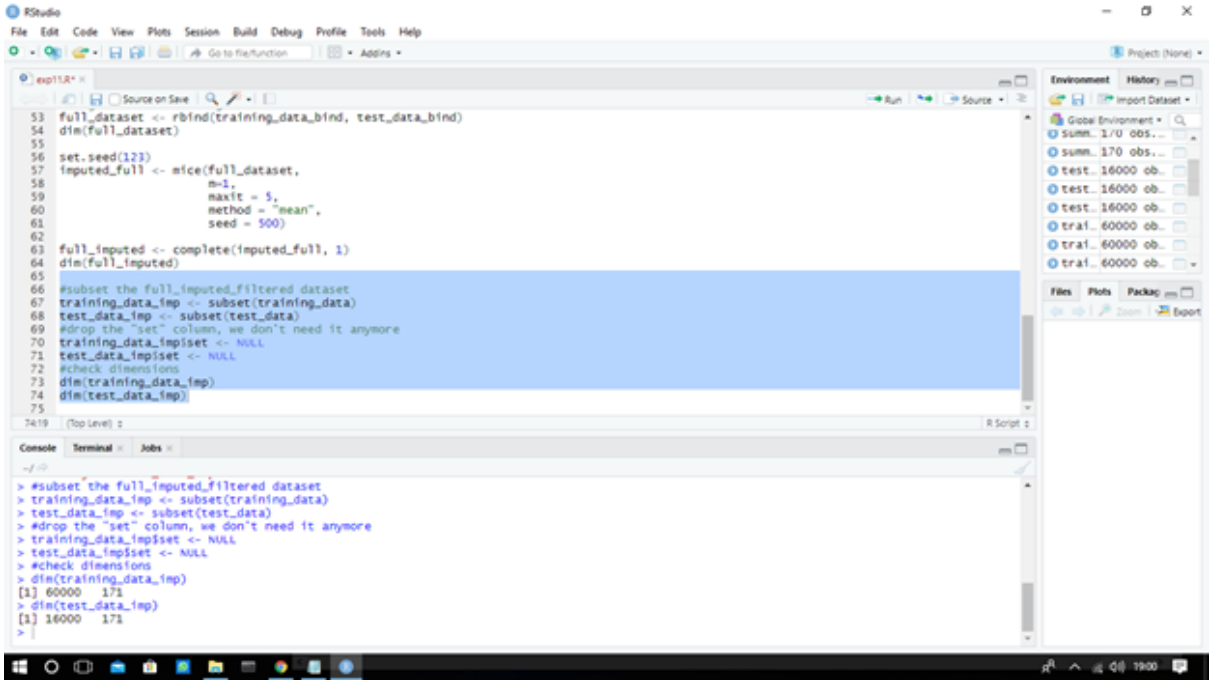


4.Rest of the features. Summary statistics. Understand your data

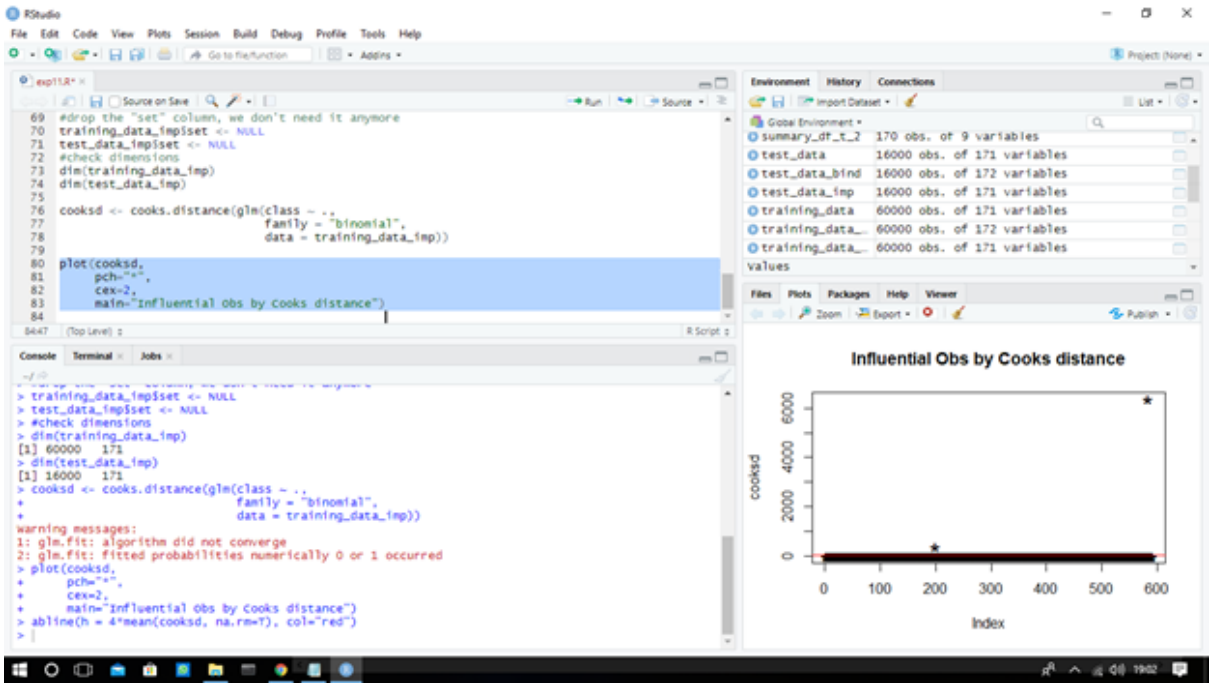


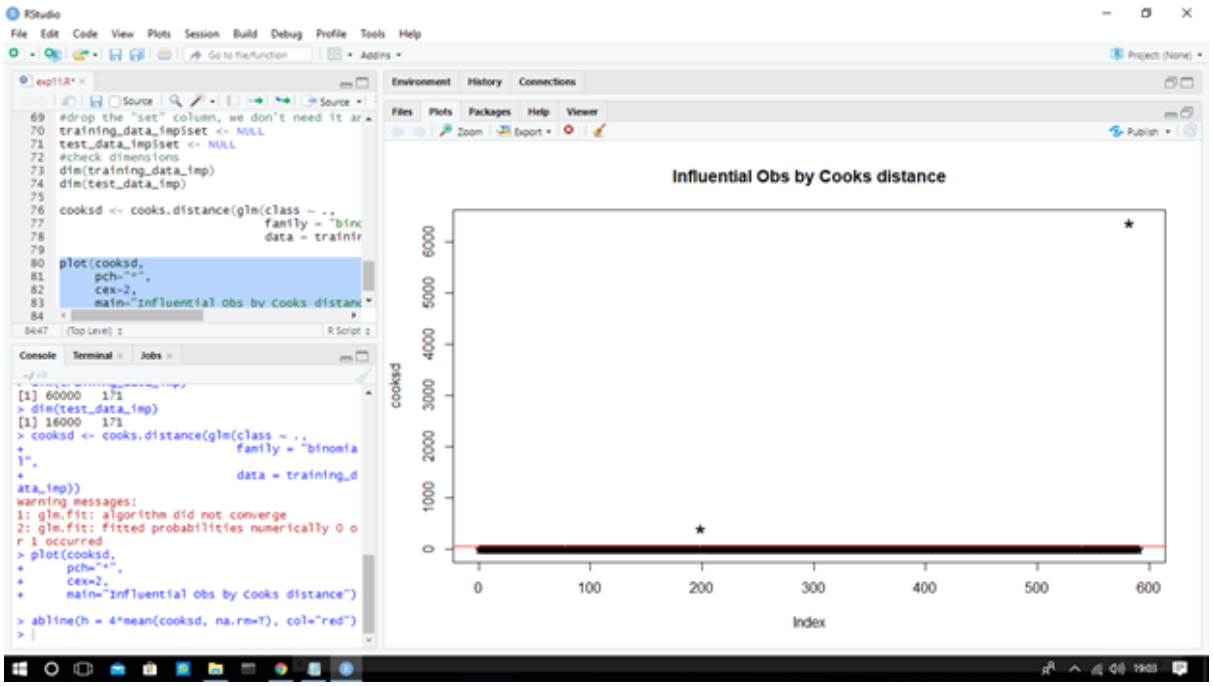


5. Our features present more than 8% missing values on average. Because it's a lot of information we don't want to lose, our approach is going to be to impute them. Impute the full dataset using mean imputation

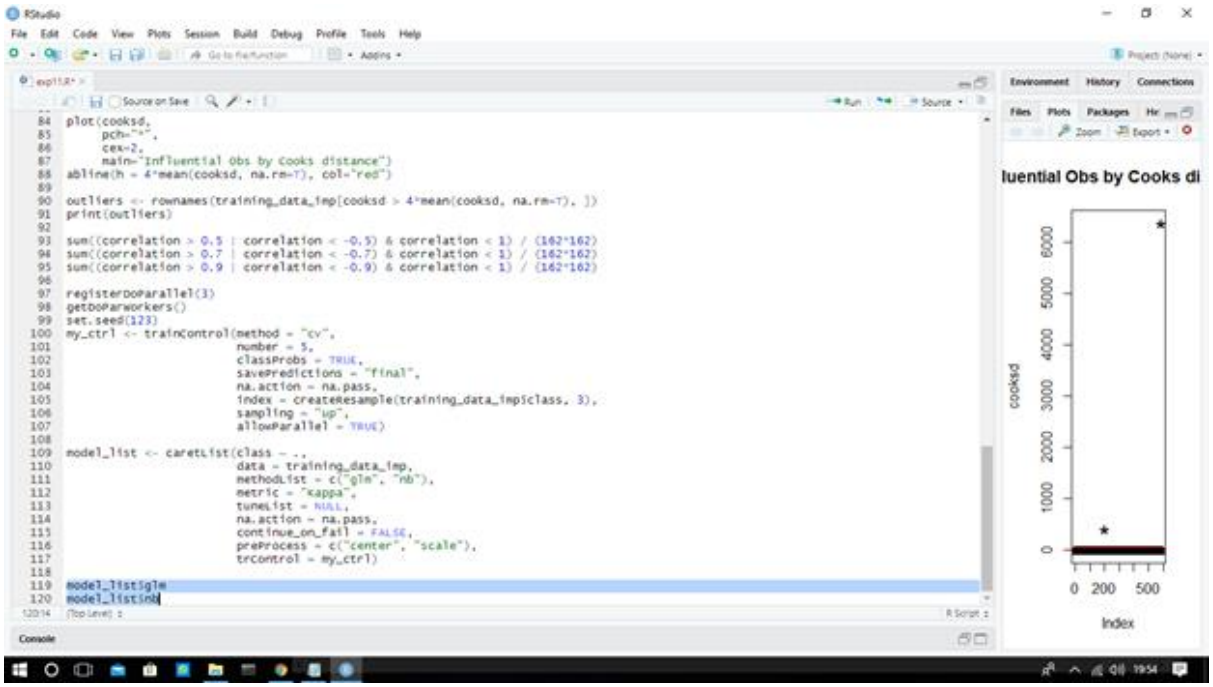


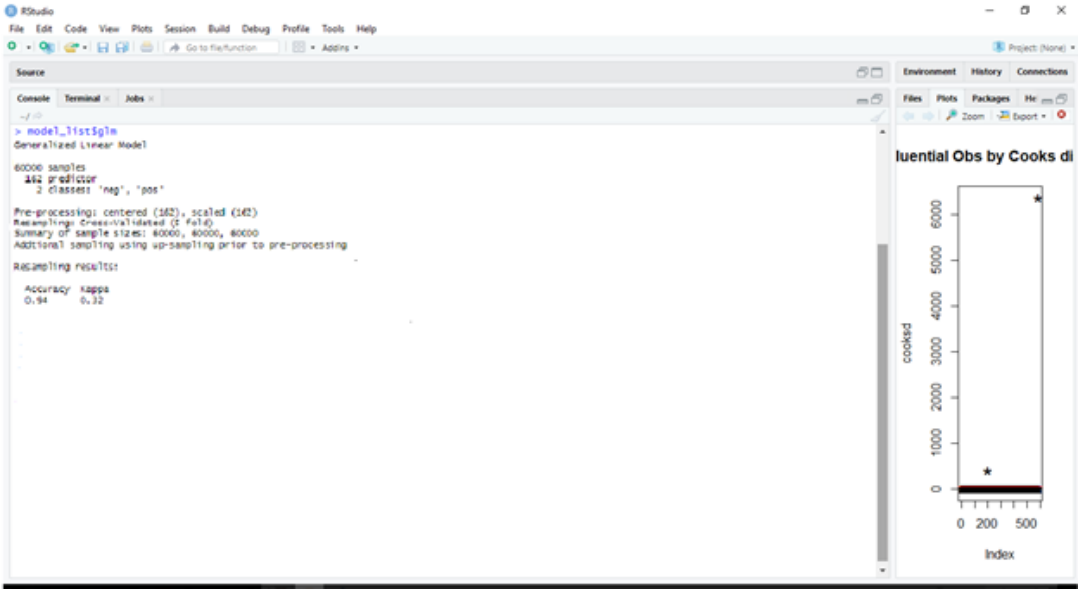
6. Check for outliers and other inconsistent data points. Box-plots. Cooks' distance. DBSCAN

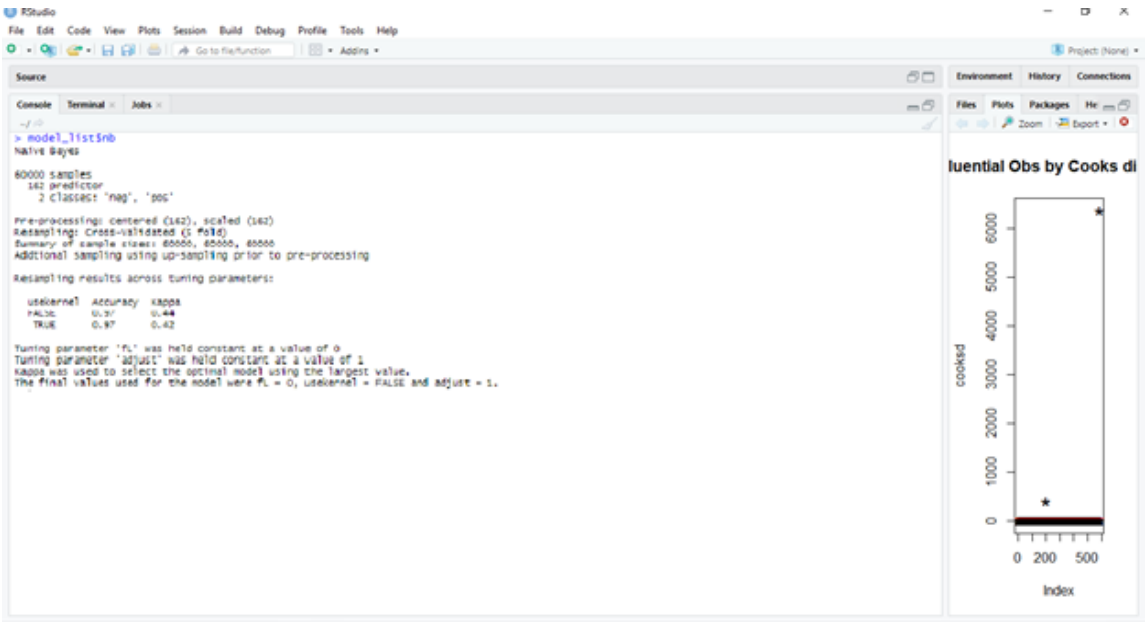




7. We'll use caretEnsemble's caretList()to train both at the same time and with the same resampling .







Conclusion: Thus, we have studied classification on large and noisy dataset with R - logistic regression and naive bayes.