7. Getting Started with R and RStudio: Installation, Data Types, Structures, and Data Management

Dr. Eldho Varghese, Dr. J. Jayasankar and Thejus T. V.

Fishery Resources Assessment, Economics & Extension Division

ICAR-Central Marine Fisheries Research Institute, Kochi

1. Introduction

R is a powerful statistical computing environment offering data manipulation, computation, and visualization capabilities. It is a free, open-source implementation of the S language, originally developed by John Chambers at Bell Labs, with a commercial version, S+®, available from Insightful Corporation. Created by Ross Ihaka and Robert Gentleman, R provides efficient data handling, matrix operations, and extensive tools for data analysis and visualization.

RStudio, an integrated development environment (IDE) for R, features a console, syntax-highlighting editor, and tools for plotting, debugging, and workspace management. Available in open-source and commercial versions, it runs on Windows, macOS, Linux, and web-based RStudio Server editions.

2. Getting R and RStudio

The Comprehensive R Archive Network (CRAN) is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R.

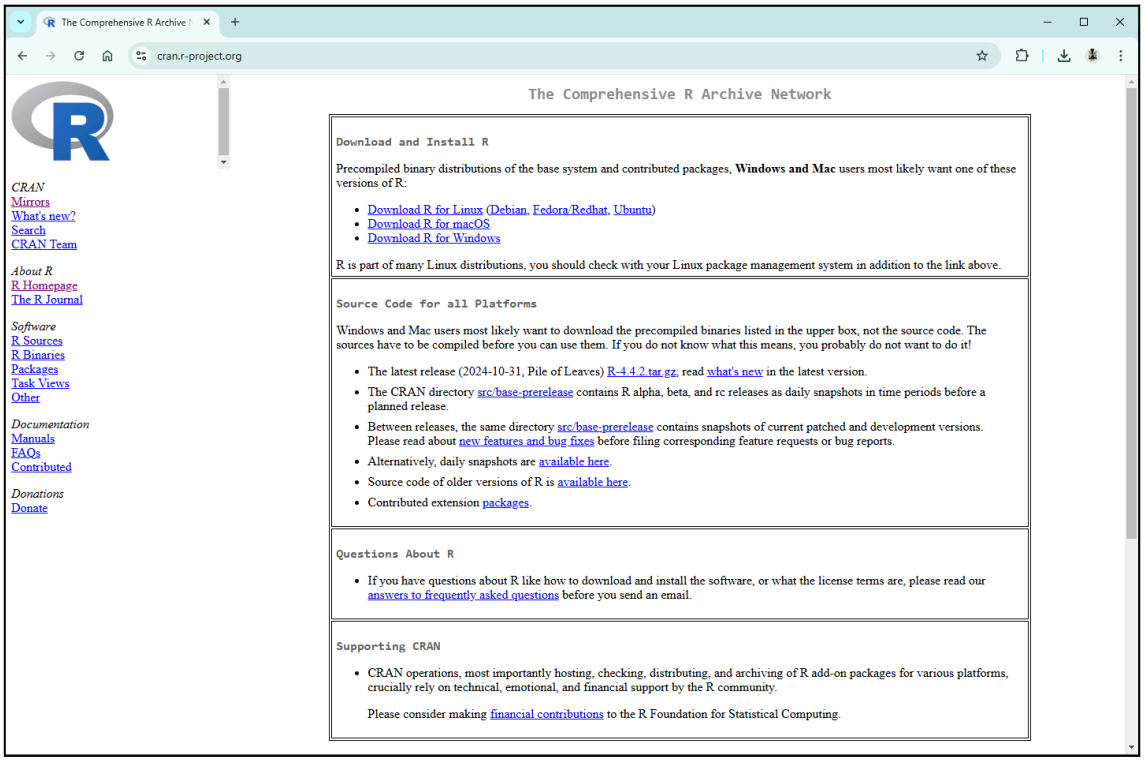


Fig.1: Home page: CRAN

R can be downloaded from https://cran.r-project.org/ . Various versions of R suited to Linux, Mac and Windows are available.

RStudio provides popular open source and enterprise-ready professional software for the R statistical computing environment. RStudio can be downloaded from https://www.rstudio.com/products/rstudio/download/

To install R/RStudio in a given machine, first double-click the downloaded file R.exe/RStudio.exe, then select language as 'English'. R setup wizard window will appear. Select on 'Next' and accept most of the default settings during the installation.

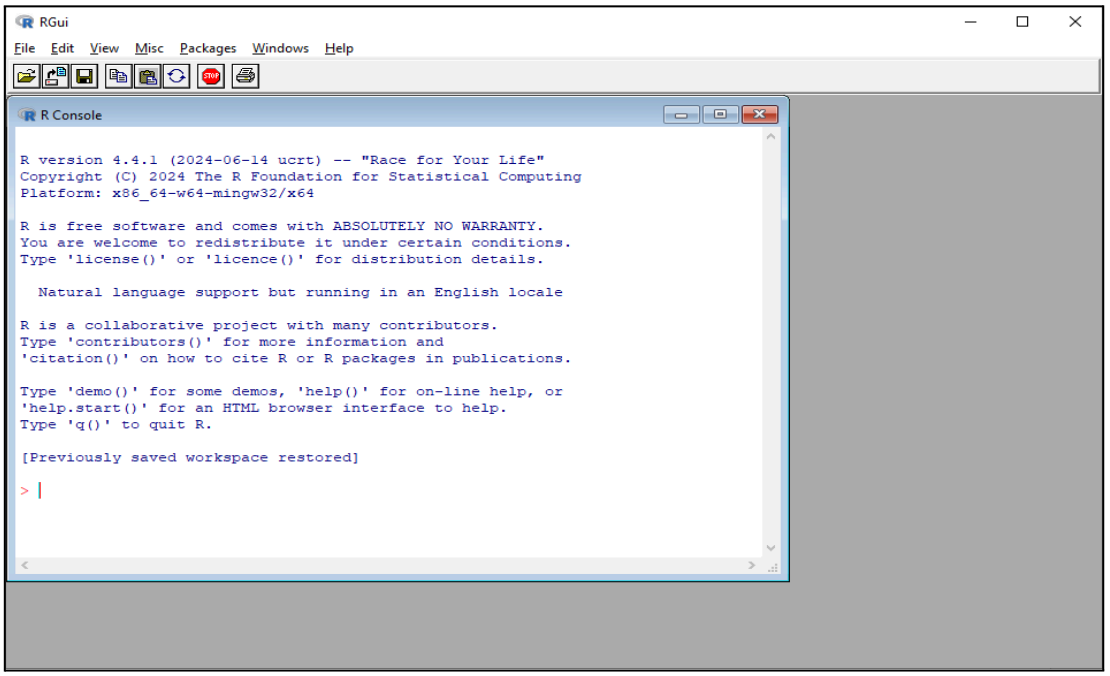


Fig 2.: R console

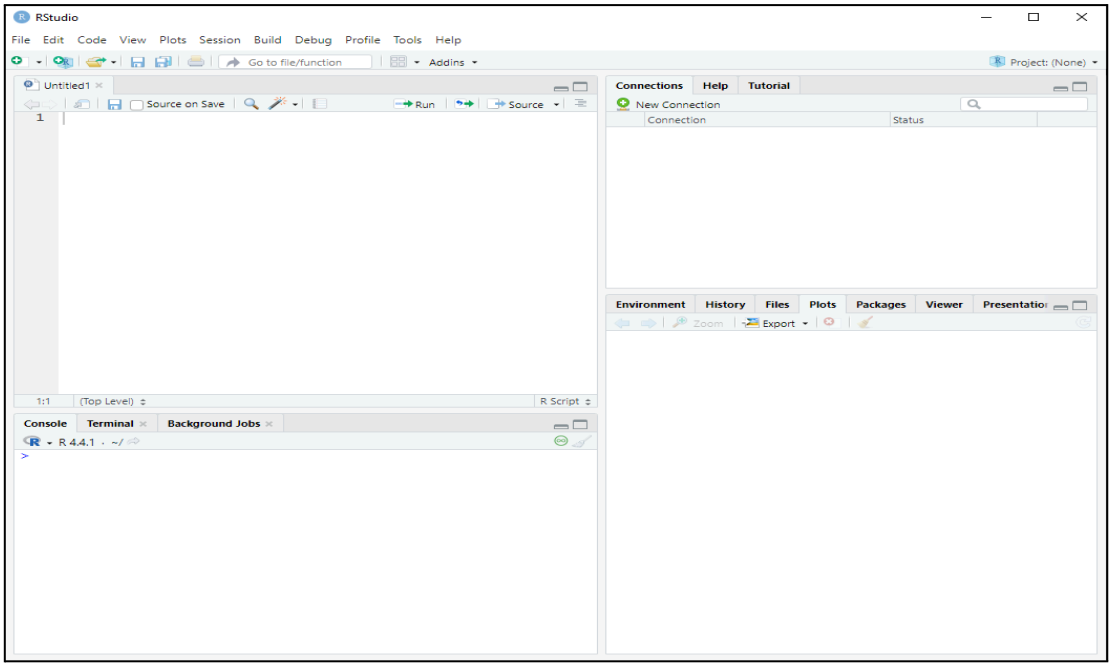


Fig.3: RStudio

RStudio provides a more user-friendly interface to use R. For using RStudio in a machine, base R must be installed in that machine. The interface of RStudio is given in Figure 3. The top left window in Figure 3 has the editor for writing R codes. Bottom left window is similar to the R console where R codes get executed. Top right window of RStudio has a few tabs: Connections, Help and Tutorial respectively. Help tab can be used to see the help on R functions.

3. R commands and Case-sensitivity

R is an expression-based language with a simple, case-sensitive syntax, where symbols like ABC, abc, and Abc represent distinct variables. Naming conventions depend on the system locale, typically allowing alphanumeric characters, accented letters (in some locales), and _ Names must begin with . or a letter, with restrictions if starting with . (the second character cannot be a digit). Names can be of unlimited length. Commands in R are either expressions or assignments. Expressions are evaluated, printed (unless hidden), and discarded, while assignments evaluate expressions and store the results in variables without automatic printing.

In RStudio, the Environment tab displays currently loaded objects, while the Files, Plots, Packages, History, and Viewer tabs offer additional functionality. The Files tab helps navigate files, the Plots tab displays generated plots, the Packages tab manages installed packages, the History tab logs executed commands, and the Viewer tab shows local web content.

4. R-help

To get more information on any specific named function, for example solve, the command is
> help(solve)
An alternative is
> ?solve
For a feature specified by special characters, the argument must be enclosed in double or single quotes, making it a “character string”: This is also necessary for a few words with syntactic meaning including if, for and function.
> help("[[")
Either form of quote mark may be used to escape the other, as in the string "It’s important". Our convention is to use double quote marks for preference. On most R installations help is available in HTML format by running
> help.start()

5. Setting a working directory

The working directory refers to the directory or folder where R is currently working. By default, the working directory is My documents or Documents. One can get the working directory by using code
> getwd()

[1] " "C:/Users/Eldho/OneDrive/Documents"

R can read and open files from working directory directly without specifying any path. Similarly, it can save files and write to files in the working directory. One can reset the working directory to a different folder using the code below.

> setwd("D:/CMFRI/Lectures delivered/Winter school_Dr. Rekha_2025")

```
> getwd()
[1] "C:/Users/Eldho/OneDrive/Documents"
> setwd("D:/CMFRI/Lectures delivered/Winter school_Dr. Rekha_2025")
> getwd()
[1] "D:/CMFRI/Lectures delivered/Winter school_Dr. Rekha_2025"
>
```

In the beginning of an R session, it is better to set the working directory to a folder where most of the data files and codes are located.

6. Data Types

R is an object-oriented language and therefore, all data types in R are some kind of object. Objects may be variables, vectors, matrices, arrays, character strings, functions, or more general structures built from such components. During an R session, objects are created and stored by name. One can use the command
> objects()
to display the names of the objects which are currently stored within R.

The collection of objects currently stored is called the workspace. One can remove objects using the function rm() . For example, the following code removes objects x and y form the workspace.
> rm(x, y)

```
> objects()
[1] "Fish_Landings"  "fishData"       "mydata"         "PellaTomlinson"
[5] "Scheafer"       "y"              "Year"
> rm(PellaTomlinson)
> objects()
[1] "Fish_Landings" "fishData"      "mydata"        "Scheafer"
[5] "y"             "Year"
>
```

An object created during an R session can be saved in a file for use in future R sessions. The entire workspace of an R session and the history of all the commands used during the session can also be saved. Some commonly encountered objects are discussed below.

(i) Vectors: Simplest object in R is a vector.
A vector is a collection of elements. For example,

Year<-c(2001,2002,2003,2004,2005,2006,2007,2008,2009,2010)
creates a vector of 10 numbers. Here the object Year contains those numbers and the function c() is used to assign those numbers to the object Year.

Vectors can be of three types a) numeric b) character and c) logical.
A numeric vector contains numbers, a character vector contains characters and a logical vector can contain values TRUE, FALSE or NA.

(ii) Matrices: A matrix object also is a collection of elements but it has two dimensions. They can also be numeric, character or logical in nature.
Following is an example of creating a matrix.
> fish<-matrix(c("Crab","Squid","Mackerel","shark"),nrow=2)
> fish

```
> fish<-matrix(c("Crab","Squid","Mackerel","shark"),nrow=2)
> fish
     [,1]    [,2]
[1,] "Crab"  "Mackerel"
[2,] "Squid" "shark"
>
```

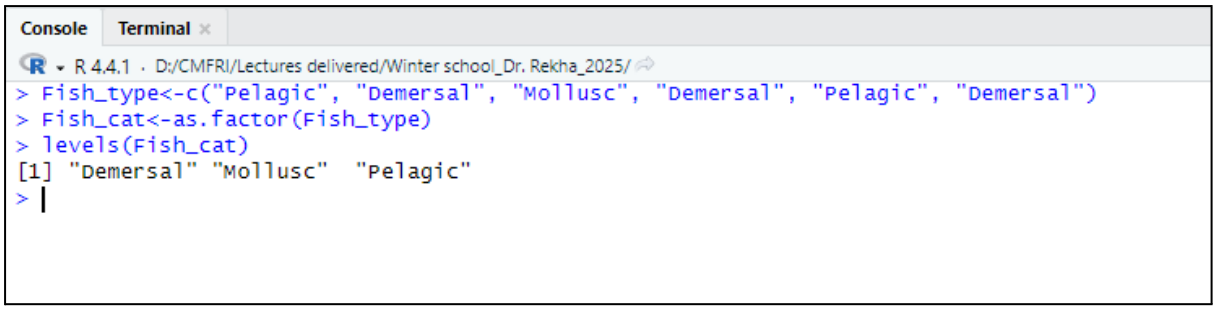
(iii) Arrays: Arrays are multi-dimensional generalization of vectors and matrices. A two-dimensional array is a matrix. Arrays can have more than two dimensions.

```
Catch_2011 <- c(20,30,45,15,26,30,23,29,30)
Catch_2012 <- c(30,40,50,70,41,45,12,17,60)
column.names <- c("Ribbonfish"," Prawns"," Tuna")
row.names <- c("Jan","Feb","Mar")
matrix.names <- c("2011","2012")
result <- array(c(Catch_2011, Catch_2012),dim = c(3,3,2),dimnames = list(row.names,column.names,  matrix.names))
print(result)
```

```
Console  Terminal
R 4.4.1 · D:/CMFRI/Lectures delivered/Winter school_Dr. Rekha_2025/
> Catch_2011 <- c(20,30,45,15,26,30,23,29,30)
> Catch_2012 <- c(30,40,50,70,41,45,12,17,60)
> column.names <- c("Ribbonfish"," Prawns"," Tuna")
> row.names <- c("Jan","Feb","Mar")
> matrix.names <- c("2011","2012")
> result <- array(c(Catch_2011, Catch_2012),dim = c(3,3,2),dimnames = list(row.names,column.names,  matrix.names))
> print(result)
, , 2011

    Ribbonfish  Prawns  Tuna
Jan         20      15    23
Feb         30      26    29
Mar         45      30    30

, , 2012

    Ribbonfish  Prawns  Tuna
Jan         30      70    12
Feb         40      41    17
Mar         50      45    60

>
```

(iv) Factors: Factor objects are used to specify categorical or classificatory or grouping variables. For example, males and females are two levels of a variable gender. Then gender can be thought of a factor object.

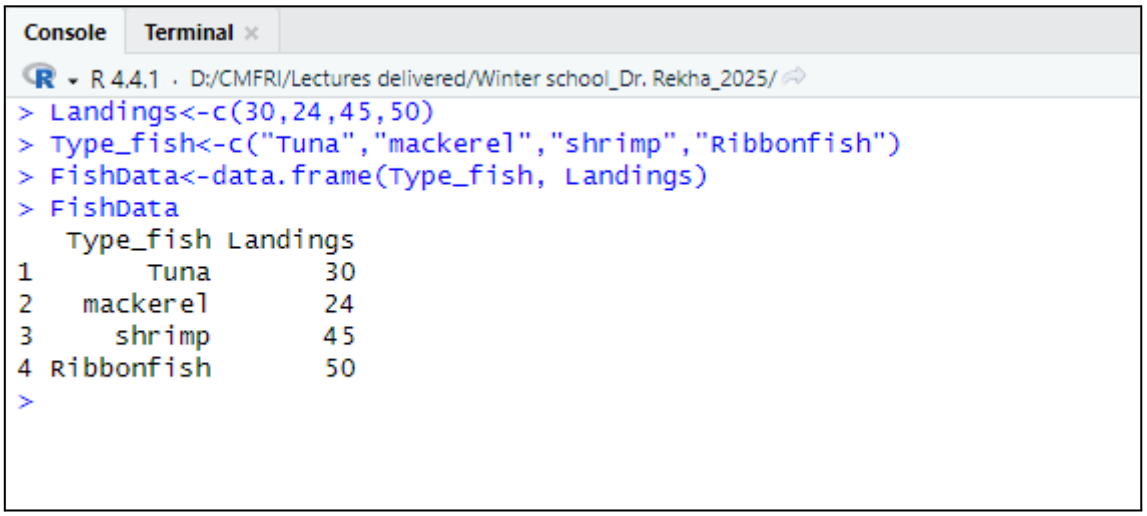
```
Fish_type<-c("Pelagic", "Demersal", "Mollusc", "Demersal", "Pelagic", "Demersal")
Fish_cat<-as.factor(Fish_type)
levels(Fish_cat)
```



Factor variables are particularly useful in analysis of variance and in linear model with grouping variables.

v) Data frames: A data frame is a two dimensional object. But unlike matrices, different columns of data frame can be different types, for example some columns can be numeric, some columns can be character, some columns can be factors. Here a column generally refers to a variable.

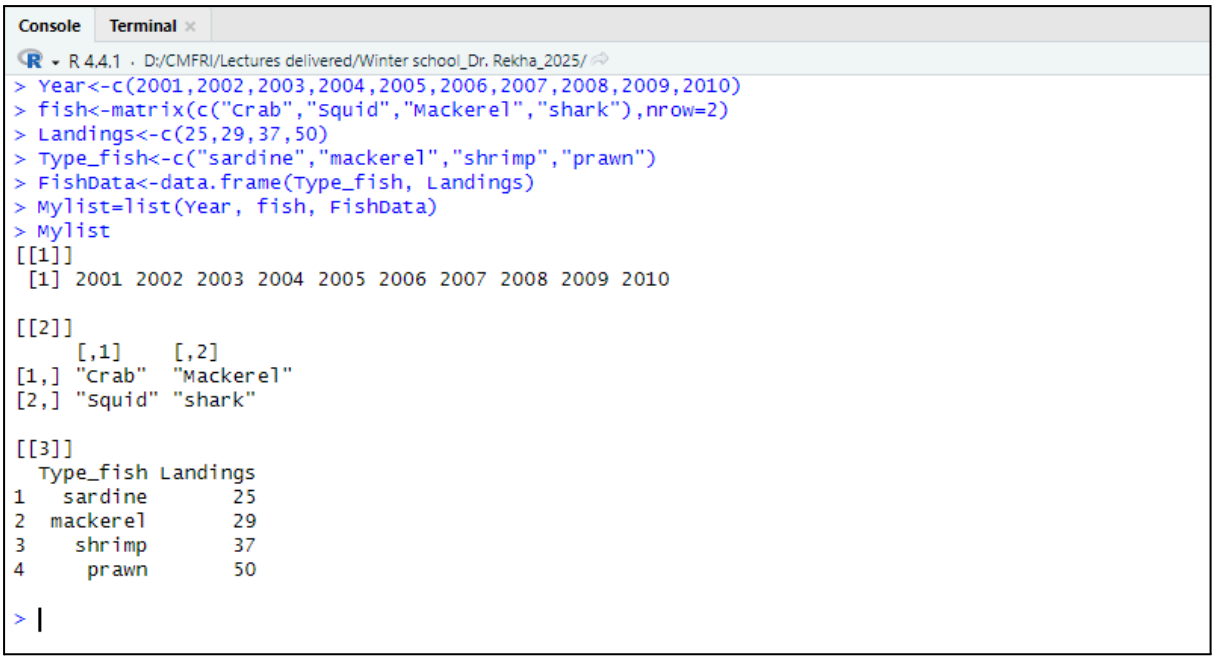
```
Landings<-c(25,29,37,50)
Type_fish<-c("sardine","mackerel","shrimp","prawn")
FishData<-data.frame(Type_fish, Landings)
FishData
```



The data.frame() function is used to create a data frame.

vi) Lists: A list is a collection of objects where each object can be of different type. For example, a list can have the first object as a vector, second object as a matrix and third object as a data frame.

```
Year<-c(1991,1992,1993,1994,1995,1996,1997,1998,1999,2000)
fish<-matrix(c("sardine","mackerel","tuna","shark"),nrow=2)
Landings<-c(25,29,37,50)
Type_fish<-c("sardine","mackerel","shrimp","prawn")
FishData<-data.frame(Type_fish, Landings)
Mylist=list(Year, fish, FishData)
Mylist
```

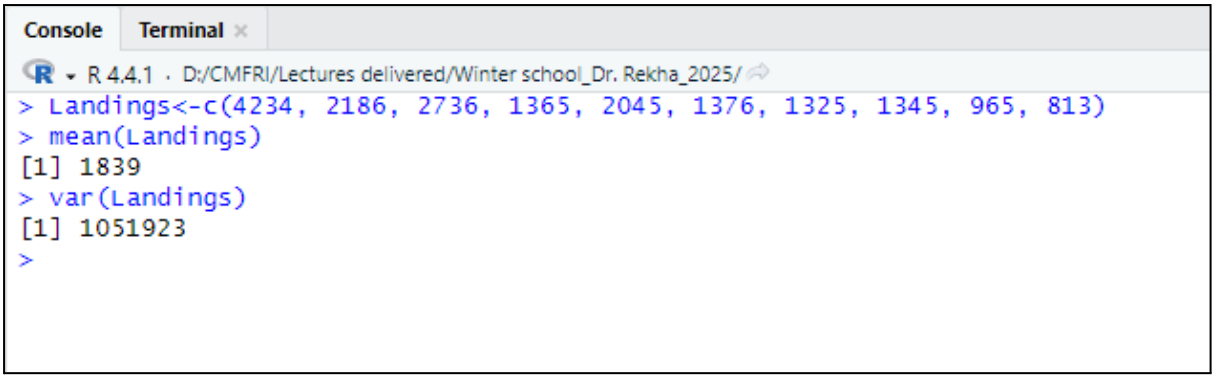


7. R Functions

Functions in R are a kind of object which takes one or more inputs and produces some result(s) as output. R has a number of in-built functions. R also provides facilities to create new functions by users. R has a huge number of in-built functions.

As an example, to obtain the mean and variance of landings of sharks during the period 2007-2016: 4234, 2186, 2736, 1365, 2045, 1376, 1325, 1345, 965, 813, the following code can be used.

```
Landings<-c(4234, 2186, 2736, 1365, 2045, 1376, 1325, 1345, 965, 813)
mean(Landings)
var(Landings)
```



Here, c() , mean() and var() are in-built functions of R. The function c() assigns those numbers to the object Landings. The commands mean(Landings) and var(Landings) computes the mean and variance of an object Landings. Here, Landings is the input, also called argument, to the function mean() and var() .

A complete list of in-built functions is available in the document R reference manual. The R reference manual opens by clicking on Help Manuals (in PDF) R reference. It opens the full reference manual. It contains a complete list of all the functions and objects in base R. Apart from in-built functions, a large number of external functions are available in contributed packages. Contributed packages are nothing but a collection of functions written by the authors of the packages to perform specific analysis. The manual of a package contains the details of the functions provided in that package. To know what are the argument(s) of a function and how to use it, you can use the code help(functionname) where functionname is the name of the function. This opens an html page in the browser containing the details of the function. For example, help(aov) gives the details of the usage of the function aov().

Some of the statistical functions are as follows:

- mean(x): Mean of the numbers in vector x
- median(x): Median of the numbers in vector x
- var(x): Estimated variance of the population from which the numbers in vector x are sampled
- sd(x): Estimated standard deviation of the population from which the numbers in vector x are sampled
- sort(x): The numbers in vector x in increasing order
- rank(x) : Ranks of the numbers (in increasing order) in vector x
- rank(-x): Ranks of the numbers (in decreasing order) in vector x
- t.test(x,mu=n, alternative = “two.sided”): Two-tailed t-test that the mean of the numbers in vector x is different from n.
- t.test(x,mu=n, alternative = “greater”): One-tailed t-test that the mean of the numbers in vector x is greater than n.
- t.test(x,mu=n, alternative = “less”): One-tailed t-test that the mean of the numbers in vector x is less than n.

- t.test(x,y,mu=0, var.equal = TRUE, alternative = “two.sided”): Two-tailed t-test that the mean of the numbers in vector x is different from the mean of the numbers in vector y. The variances in the two vectors are assumed to be equal.
- t.test(x,y,mu=0, alternative = “two.sided”, paired = TRUE) : Two-tailed t-test that the mean of the numbers in vector x is different from the mean of the numbers in vector y. The vectors represent matched samples.
- cor(x,y): Correlation coefficient between the numbers in vector x and the numbers in vector y
- cor.test(x,y): Correlation coefficient between the numbers in vector x and the numbers in vector y, along with a t-test of the significance of the correlation coefficient.
- lm(y~x, data = d): Linear regression analysis with the numbers in vector y as the dependent variable and the numbers in vector x as the independent variable. Data are in data frame d.
- coefficients(a) Slope and intercept of linear regression model a.
- confint(a): Confidence intervals of the slope and intercept of linear regression model a

8. R packages

All R functions and datasets are stored in packages. Only when a package is loaded are its contents available. This is done both for efficiency (the full list would take more memory and would take longer to search than a subset), and to aid package developers, who are protected from name clashes with other code.

To see which packages are installed at your site, issue the command
> library()
with no arguments.
To load a particular package named say, “boot”, use a command like
> library(boot)
Users connected to the Internet can use the install.packages() and update.packages() functions (available through the Packages menu in the Windows and MacOS GUIs, see Section “Installing packages” in R Installation and Administration) to install and update packages.

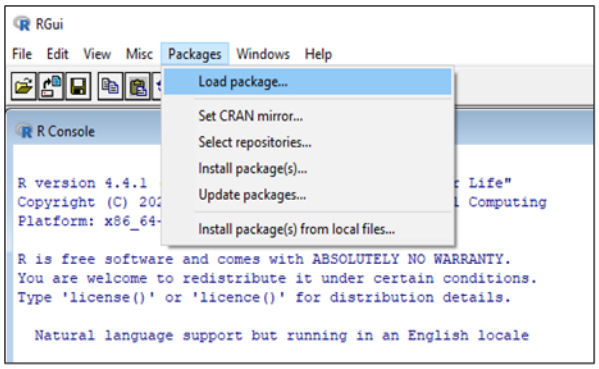


Fig.4: Package installation in R

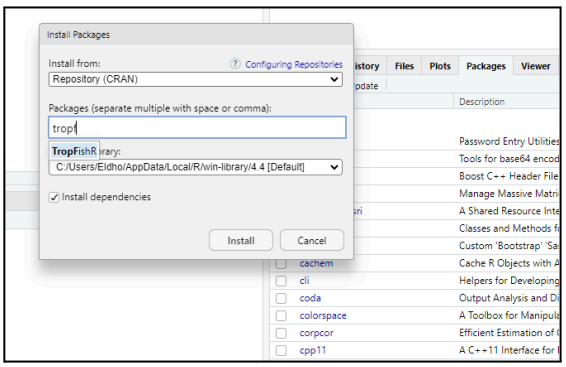


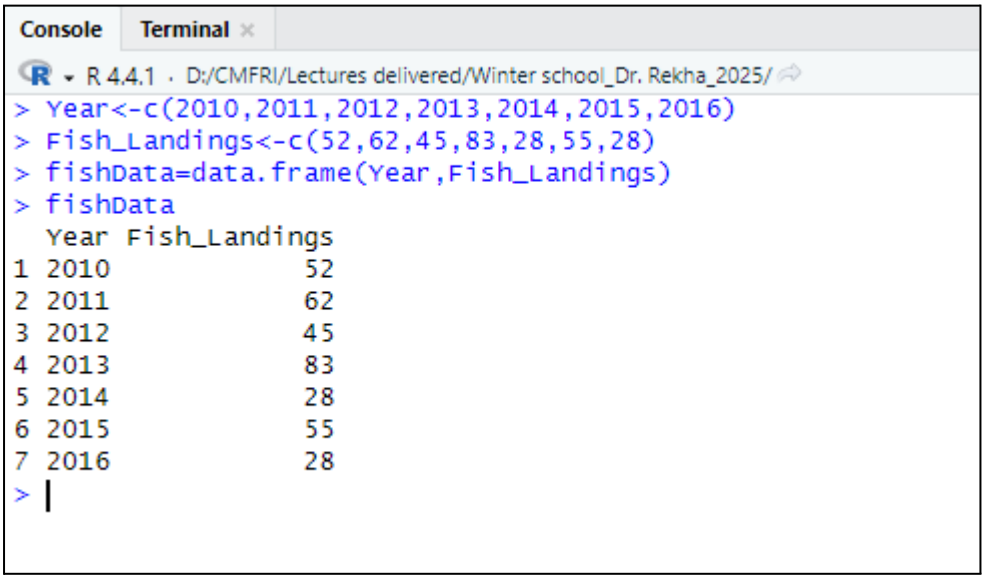
Fig.4: Package installation in RStudio

To see which packages are currently loaded, use
> search()

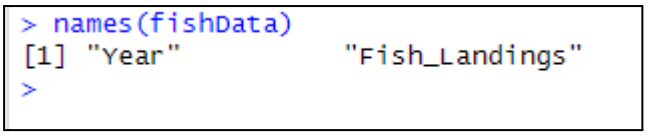
9. Reading Data File

(i) Entering data in R directly:
Suppose a data set contains few variables and few observations, then it can be read in R by typing in the Console R as shown in the following example.

> Year<-c(2010,2011,2012,2013,2014,2015,2016)
> Fish_Landings<-c(52,62,45,83,28,55,28)
> fishData=data.frame(Year,Fish_Landings)
> fishData

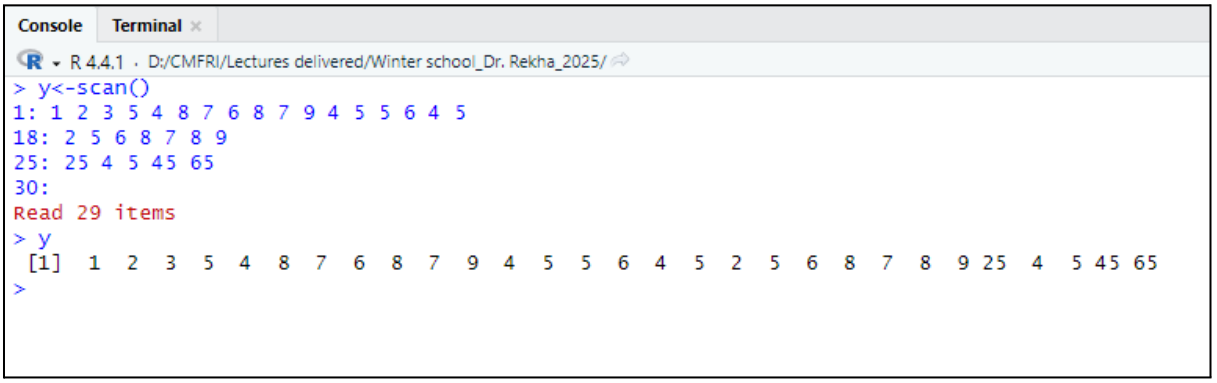


Note that data.frame() function combines the vectors month and rainfall into a data frame called fishData. Note that a dataset in R is always in the form of a two-dimensional array with columns representing variables and rows representing individual observations. Sometimes one may be interested to know the names of variables in a data set loaded in R. For example, to know the names of the variables in data set fishData, one can use following command:
> names(FishData)



The scan() function can also be used to read data directly typed in R console. For example,

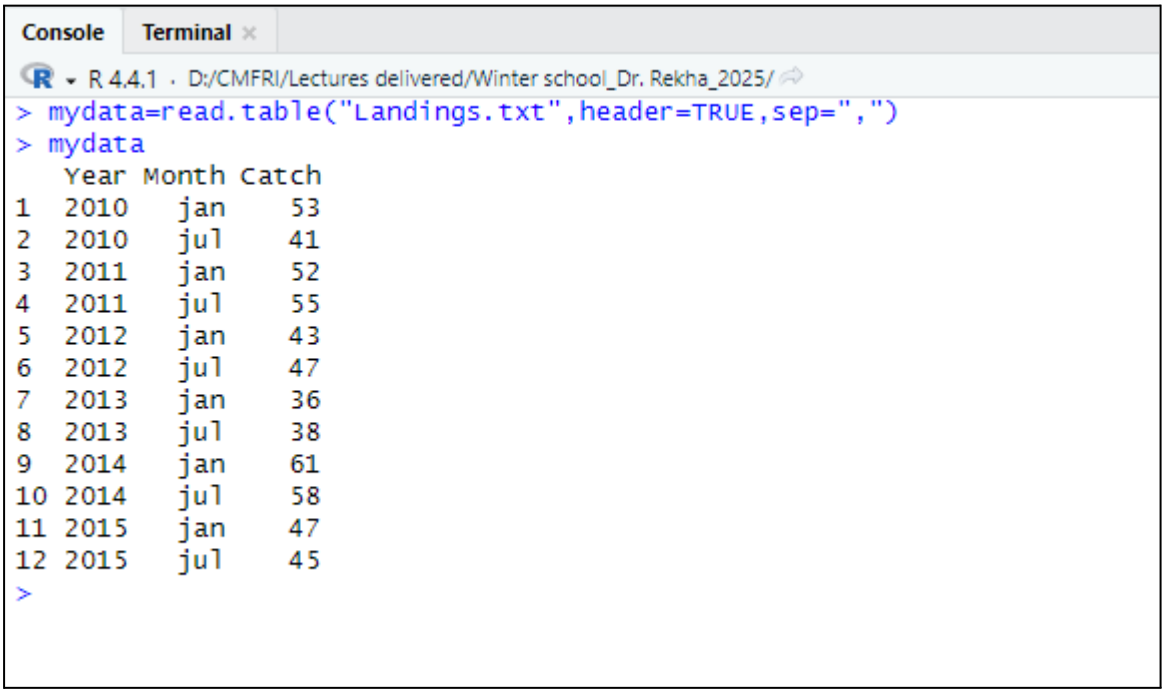
> y<-scan()



When entering data from the keyboard using scan() function, one has to hit the enter button twice to stop scanning. Then R stops scanning and loads the data into the object.
The function scan() is also able to read data from external file.

(ii) Loading data in R from an external file:
Most often the data may not be just a few observations. There may be quite many variables and observations. In that case, the data may be in a spreadsheet or some other external file, or from some other statistical software or from some web page. R provides facilities for loading data from each of them.

(a) Reading data from text file:
Data in text file should be kept such that the individual observations are separated with a delimiter. Some commonly used delimiters are ‘,’,‘:’,‘t’,‘ ’ i.e., blank space, ‘~’, ‘@’, ‘&’, ‘*’ etc. But be sure that none of the observations or variables in the data set have any of those characters, otherwise data will be loaded improperly and there may be error in loading of data.
Consider a text file contains landings of a particular species in the month of January and July during 2010-2015 with comma(‘,’) as a delimiter.
Year,Month,Catch
2010,jan,53
2010,jul,41
2011,jan,52
2011,jul,55
2012,jan,43
2012,jul,47
2013,jan,36
2013,jul,38
2014,jan,61
2014,jul,58
2015,jan,47
2015,jul,45
Let the file name is Landings.txt and is kept in the working directory. This data can be loaded in R by using the function read.table() as follows:
> mydata=read.table("Landings.txt",header=TRUE,sep=",")
> mydata



The first argument of read.table() refers to the external file. The second argument header=TRUE tells R that there is a header in the Landings.txt file, and those are used as variable names for the data. If there is no header in a text file, then header=FALSE should be used. Third argument sep="," tells R that observations are separated by a ‘,’. There are other arguments to read.table() function, but these three are essential. The details of the usage of the function read.table() is available with help(read.table) in the Console.

(b) Loading data from a spreadsheet:

There are some other functions to read files with specific delimiters. The function read.csv() function loads comma separated value (csv) files, i.e., files with comma delimited observations, read.csv2() function loads data from semicolon (‘;’) delimited files, read.delim() and read.delim2() functions load data from tab delimited files.

(c) Loading data from a spreadsheet:

To load data from an excel file to R, the relevant worksheet may be saved into a tab delimited text file or into a csv file and then the text file or .csv may be loaded using read.table() or read.csv() function. However, if you want to read the data from excel directly into R, a package called xlsx is needed. An example of loading data from excel is shown below.

```
> library(xlsx)
> read.xlsx("myfile.xlsx", sheetName = "Sheet1")
```

(d) Reading data from a webpage:

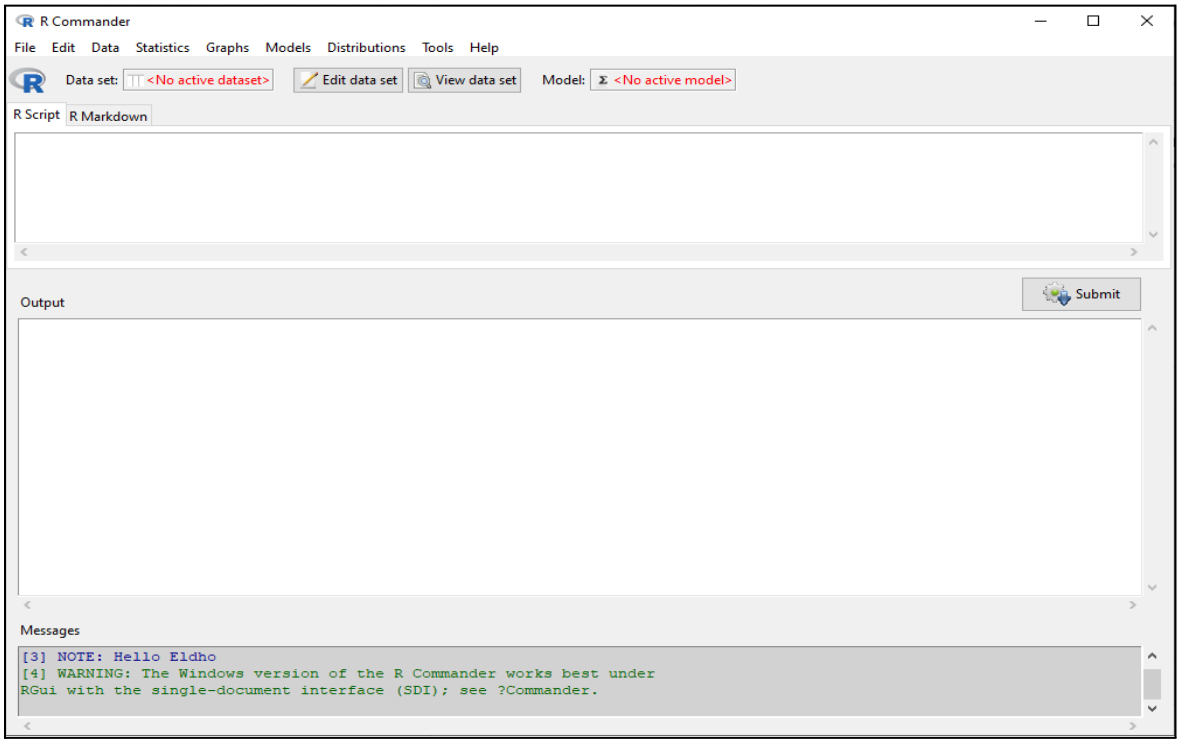
Suppose some data is available on a webpage. To read a dataset from a web page the function read.table() can be used with the complete address of the page. For example,

```
> webdata=read.table("http://cmfri.org.in/datasets/effort.dat")
```

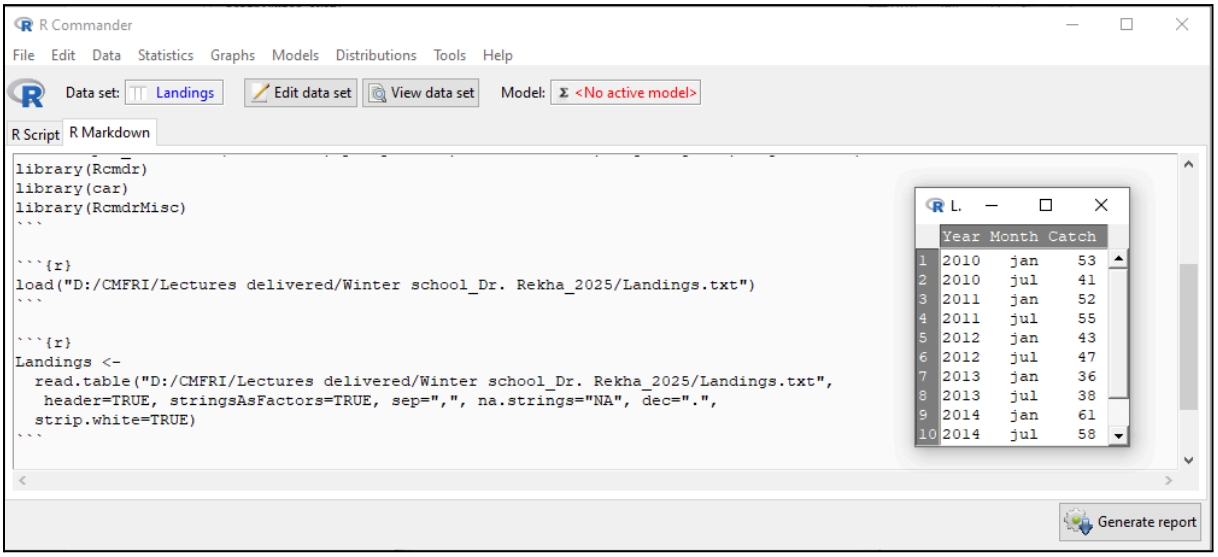
10. R Commander: A graphical interface for R

Rcmdr (R Commander) is a graphical user interface for R that simplifies data analysis by providing easy access to commonly used commands through an intuitive interface. It's particularly useful for users who are new to R, enabling them to perform analyses without needing extensive coding knowledge. Additionally, Rcmdr helps users learn R commands as they work, fostering the development of command-line skills necessary to fully utilize R's powerful features.

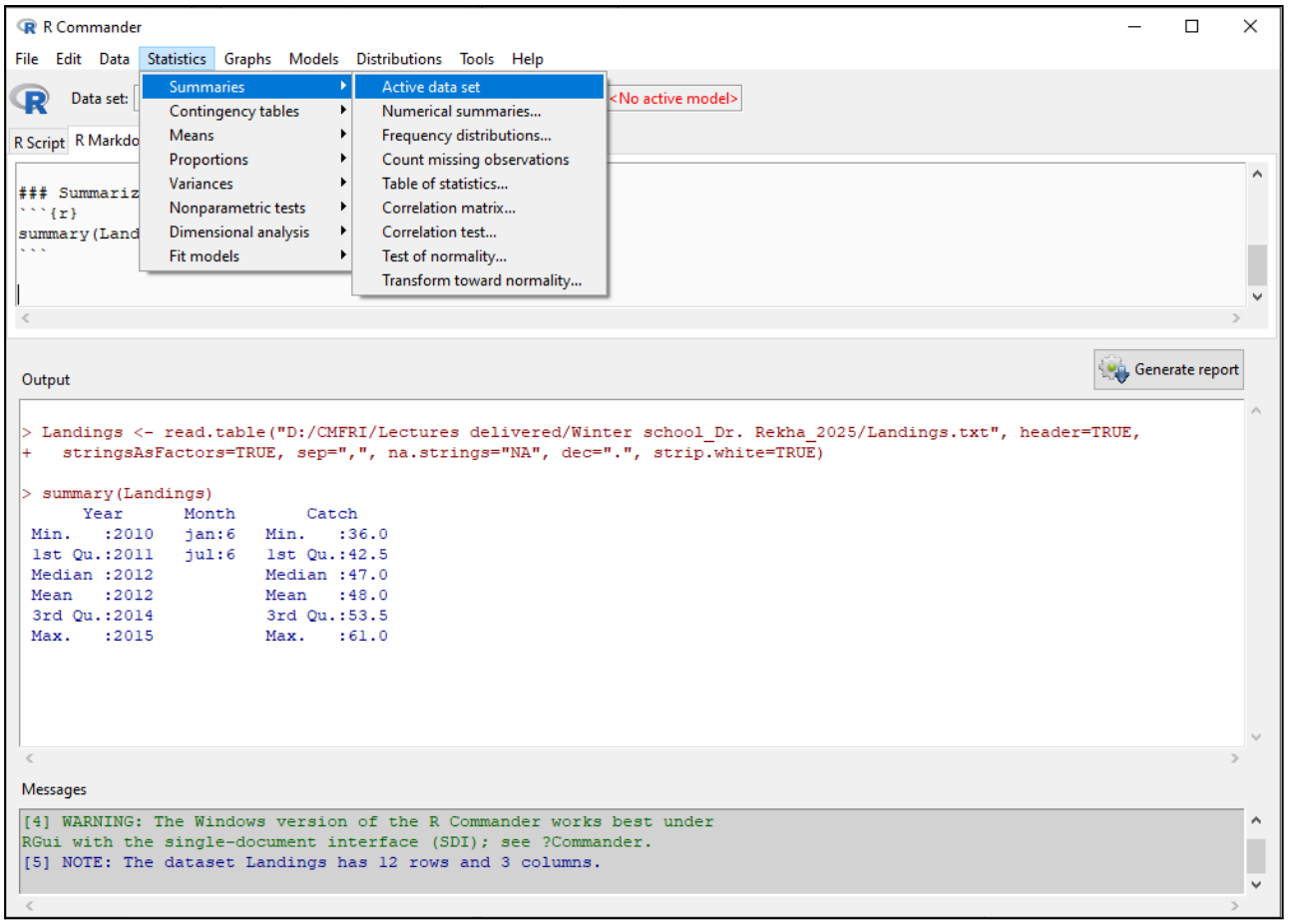
Loading R commander:
install.packages(Rcmdr)
library(Rcmdr)



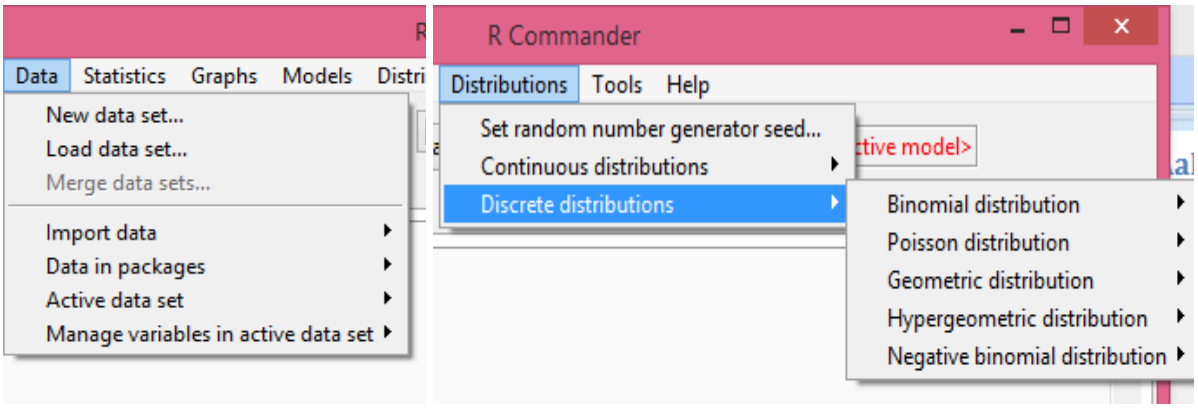
Active data set:

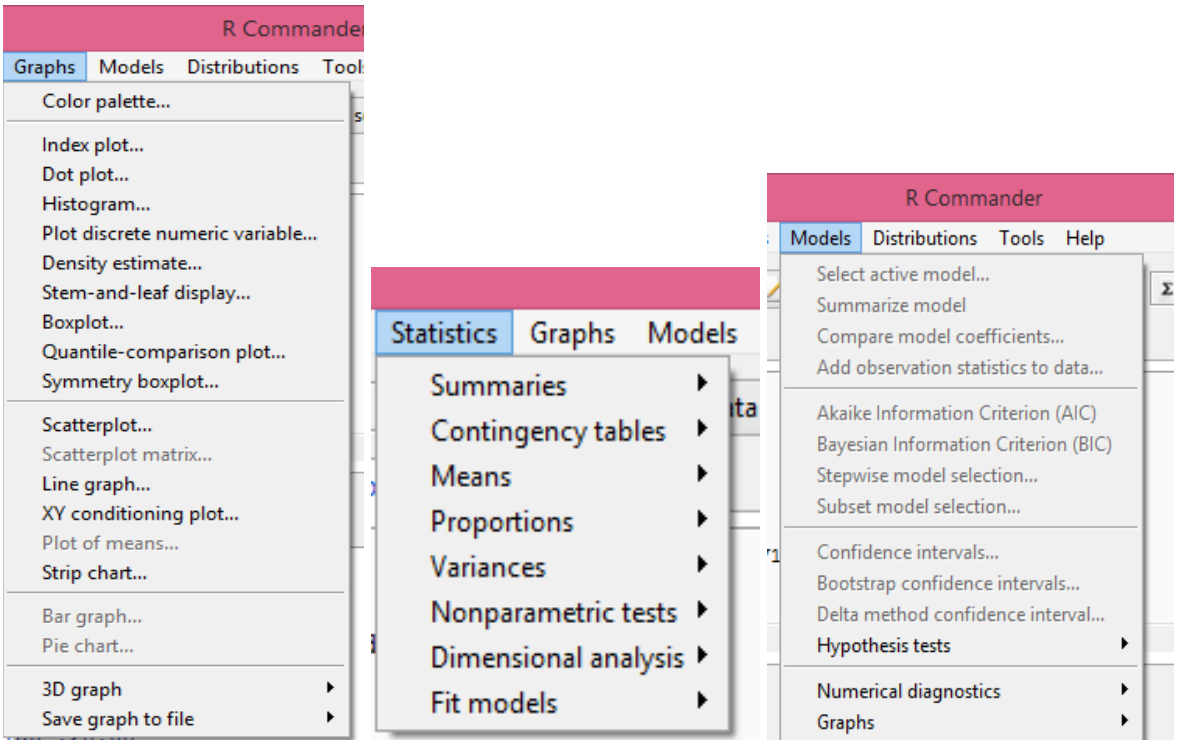


Summary of the active data set:



Menu options:





11. Some examples

The following data (Landings of Bombay duck, Crabs, Indian mackerel, Ribbon fishes and Sharks during 1997-2017 along A.P. coastline) is used for illustrations:

Year	Bombay duck	Crabs	Indian mackerel	Ribbon fishes	Sharks
1997	951	3491	15767	11273	3561
1998	732	3326	14879	8471	2956
1999	1032	3014	19681	20170	6871
2000	693	2791	9834	13842	4914
2001	2165	2929	9306	7278	3257
2002	1754	4955	14206	18372	1613
2003	701	5113	22572	15565	1921
2004	2986	6877	20209	9995	2250
2005	1318	5405	17665	6398	1855
2006	1821	6804	13004	16522	3806
2007	1832	5324	7903	11449	4434
2008	688	4678	18127	15960	2194

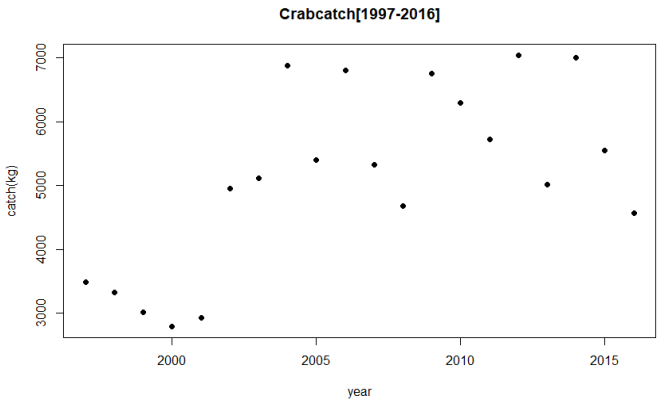
2009	556	6757	23078	11895	2737
2010	1053	6298	19564	9363	1263
2011	1262	5718	22410	15348	2068
2012	1057	7041	28407	21777	1385
2013	1335	5008	33716	18800	1319
2014	930	7007	55631	20269	1352
2015	443	5543	28236	8808	971
2016	466	4571	22849	14993	819

To read the file:
Landings=read.csv("LandingsData.csv",header=TRUE,sep=",")

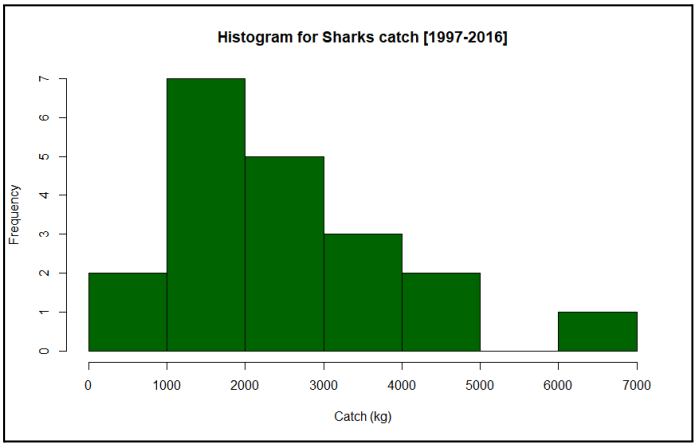
To get a summary:

```
Console   Terminal
R 4.4.1 · D:/CMFRI/Lectures delivered/Winter school_Dr. Rekha_2025/
> Landings=read.csv("LandingsData.csv",header=TRUE,sep=",")
> summary(Landings)
      Year        Bombay.duck       Crabs       Indian.mackerel Ribbon.fishes       Sharks
 Min.   :1997   Min.   : 443   Min.   :2791   Min.   : 7903   Min.   : 6398   Min.   : 819
 1st Qu.:2002   1st Qu.: 699   1st Qu.:4301   1st Qu.:14711   1st Qu.: 9837   1st Qu.:1377
 Median :2006   Median :1042   Median :5218   Median :19623   Median :14418   Median :2131
 Mean   :2006   Mean   :1189   Mean   :5132   Mean   :20852   Mean   :13827   Mean   :2577
 3rd Qu.:2011   3rd Qu.:1440   3rd Qu.:6413   3rd Qu.:22906   3rd Qu.:16985   3rd Qu.:3333
 Max.   :2016   Max.   :2986   Max.   :7041   Max.   :55631   Max.   :21777   Max.   :6871
>
```

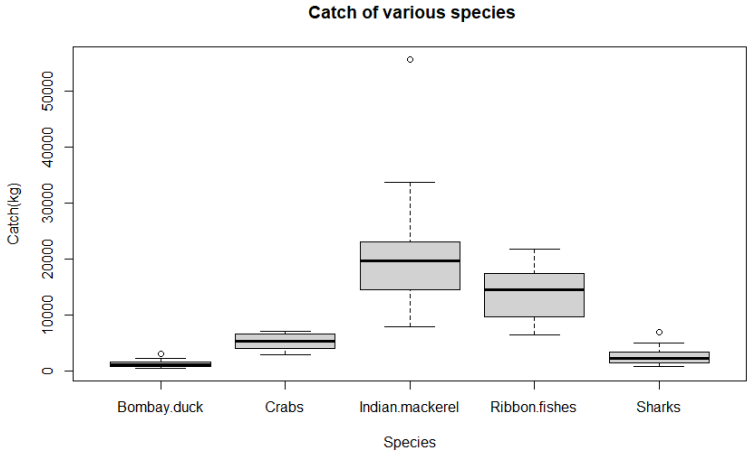
To make plot the landings:
plot(Landings$Year,Landings$Crabs,main="Crabcatch[1997-2016]",xlab="year",ylab= "catch(kg)")



To make a histogram:
hist(Landings$Sharks, main="Histogram for Sharks catch [1997-2016]", xlab="Catch (kg)", col="darkgreen", breaks = 5)



To make a Box plot:
boxplot(Landings[,-1], main=" Catch of various species", xlab="Species", ylab="Catch(kg)")



To make a Stem and leaf plot:

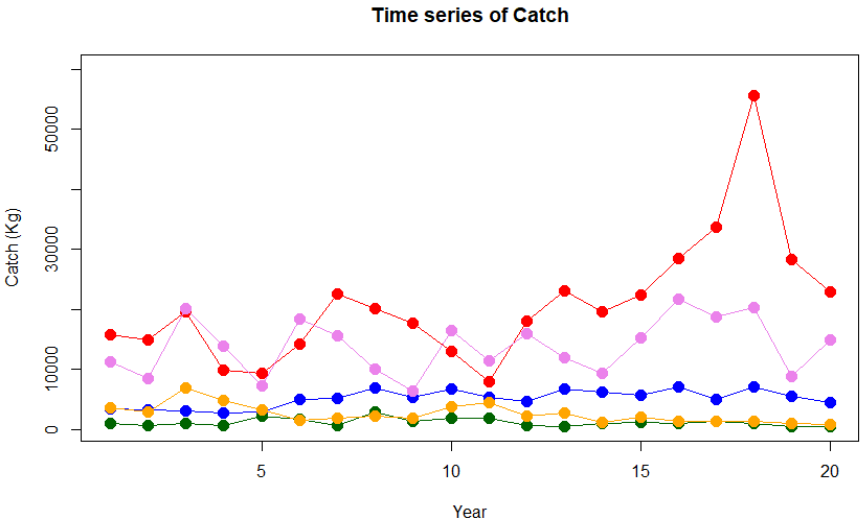
```
> stem(Landings$Shark)

  The decimal point is 3 digit(s) to the right of the |

  0 | 803344699
  2 | 12370368
  4 | 49
  6 | 9
```

```
> stem(Landings$Sharks,scale=2)

  The decimal point is 3 digit(s) to the right of the |

  0 | 8
  1 | 03344699
  2 | 1237
  3 | 0368
  4 | 49
  5 |
  6 | 9
```

```
> stem(Landings$Sharks,scale=3)

  The decimal point is 3 digit(s) to the right of the |

  0 | 8
  1 | 03344
  1 | 699
  2 | 123
  2 | 7
  3 | 03
  3 | 68
  4 | 4
  4 | 9
  5 |
  5 |
  6 |
  6 | 9
```

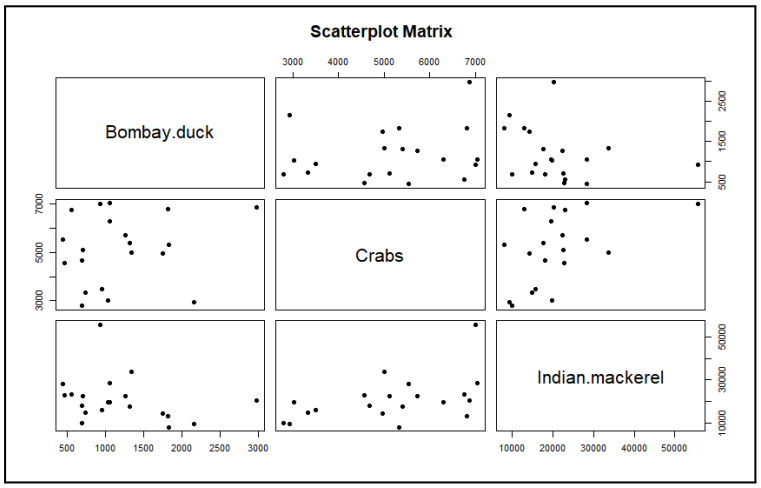
To make plot of all the species landings together:

```
plot(Landings$Indian.mackerel, type = "o",col = "red", xlab = "Year", ylab = "Catch (Kg)", main = "Time series of Catch", ylim=c(500, 60000), pch=19,cex=1.5)
# type ‘o’ means both point and lines
lines(Landings$Crabs, type = "o", col = "blue",pch=19,cex=1.5)
lines(Landings$Bombay.duck, type = "o", col = "darkgreen",pch=19,cex=1.5)
lines(Landings$Ribbon.fishes, type = "o", col = "violet",pch=19,cex=1.5)
lines(Landings$Sharks, type = "o", col = "orange",pch=19,cex=1.5)

# pch gives Solid bold circles, cex Increases the size of the points
```



To prepare a scatter plot matrix

pairs(~Bombay.duck+Crabs+Indian.mackerel,data=Landings, main=" Scatterplot Matrix")



To study length-weight relationship:

len<-c(90, 128, 112, 68, 56, 58, 111, 111, 115, 65)

wt<-c(9.3, 32.5, 19, 4.4, 2.1, 2.8, 16.1, 17.9, 22.7, 3.4)

logL<-log(len)

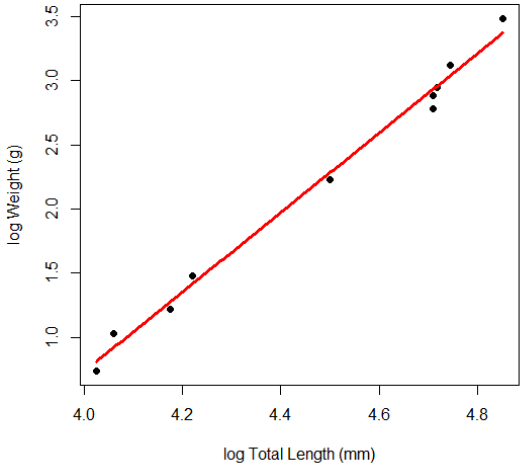
logW<-log(wt)

lm1<-lm(logW~logL)

summary(lm1)

```
Call:
lm(formula = logW ~ logL)

Residuals:
     Min       1Q   Median       3Q      Max
-0.14942 -0.04967 -0.02749  0.08066  0.11233

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -11.6354     0.4401  -26.44 4.50e-09 ***
logL          3.0924     0.0982   31.49 1.13e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.09391 on 8 degrees of freedom
Multiple R-squared:  0.992,     Adjusted R-squared:  0.991
F-statistic: 991.6 on 1 and 8 DF,  p-value: 1.125e-09
```

```
library(FSA)
fitPlot(lm1,xlab="log Total Length (mm)",ylab="log Weight (g)",main="")
```



To test H0 : β = 3 ⇒ H0 : "Isometric growth" HA : β ≠ 3 ⇒ HA : "Allometric growth":

```
> hoCoef(lm1,2,3)
 term Ho Value Estimate Std. Error         T df    p value
    2        3 3.092374 0.09820357 0.9406334  8 0.3744233
```

'2'means second coefficient and '3' means the value of H0

References

1. http://www.iasri.res.in/sscnars/R_manual/01%20R%20an%20Overview%20and%20Descriptive%20Statistics_dandapani.pdf
2. http://www.iasri.res.in/sscnars/sas_manual/19-An%20Introduction%20to%20R.pdf
3. https://sfg-ucsb.github.io/fishery-manageR/basic-fisheries-statistics.html
4. Mandal B.N. (2015). Introductory R for Beginners: A Beginner's Guide to Using R, Brillion Publishing
5. R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0,
6. URL http://www.R-project.org.
7. Varghese, Eldho and Jayasankar, J (2023) Introduction to R and R studio: Installation, data types, structures and data handling. In: Training Manual on Advanced Analytical Tools for Social Science Research Vol.1. CMFRI Training Manual Series No. 29/2023. ICAR- Central Marine Fisheries Research Institute, Kochi, pp. 27-60.
8. Venables, W. N., Smith, D. M. and The R Core Team (2018) An Introduction to R Notes on R: A Programming Environment for Data Analysis and Graphics (Version 3.5.1). Available at https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf