Subject:	**R Programming Lab. (ITL804)**		
Class:	**BE IT / Semester – VIII (Rev-2016) / Academic year: 2020-21**		
Name of Student:	**Pranali Hanumant Kudtarkar**		
Roll No:	**26**	**Date of performance (DOP) :**	
Assignment/Experiment No:	**01**	**Date of checking (DOC) :**	
Title: Program to demonstrate basic functionality of R such as- helps, accessing packages, data types, number, complex, characters, basic operators			
Marks:		**Teacher's Signature:**	

1. Aim: To understand the basics functionality of R software.

2. Prerequisites:
1. Basics of programming disciplines.

3. Hardware Requirements:
1. PC with minimum 2GB RAM

4. Software Requirements:
1. Windows / Linux OS.
2. R version 3.6 or higher

5. Learning Objectives:
1. To understand R software as a software development platform.
2. To understand elementary building blocks of R software such as- data types, number, character, complex, vectors,, helps, packages.

6. Learning Objectives Applicable: LO 1

7. Program Outcomes Applicable: PO 1

8. Program Education Objectives Applicable: PEO 1

9. Theory:

HELP

Use the help() command

To find information for a particular function, such as the function print, type help('print') on the R command line and press enter (I recommend using quotes whenever you use this command, but there are some special cases when they are unnecessary). This will open up a window with information on how to use the required function.

Typing help() on the R command line and pressing enter will open a window telling you a bit on how to use the help()command

Alternatively, the same results can be achieved by typing a question mark followed by the name of the command to query. For instance, to bring up the help file for the function print, type ?print into the command line.

Use the help.search() command

If you don't know the name of the command you are looking for then this is the command for you. Used in the same way as the help() command, this will bring up a list of places in the help file where your word occurs. Then use the help() function to look up this references.

e.g. Typing help.search('affymetrix') brings up lots of topics relating to Affymetrix microarrays.

Use the help.start() command

This opens the HTML help browser, just as picking the option from the help menu would do.

Packages

Packages are collections of R functions, data, and compiled code in a well-defined format. The directory where packages are stored is called the library. R comes with a standard set of packages. Others are available for download and installation. Once installed, they have to be loaded into the session to be used.

```
.libPaths() # get library location
```

```
library()   # see all packages installed
```

```
search()    # see packages currently loaded
```

Data types

Generally, while doing programming in any programming language, you need to use various variables to store various information. Variables are nothing but reserved memory locations to store values. This means that, when you create a variable you reserve some space in memory.

You may like to store information of various data types like character, wide character, integer, floating point, double floating point, Boolean etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

In contrast to other programming languages like C and java in R, the variables are not declared as some data type. The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable. There are many types of R-objects. The frequently used ones are −

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

Number and complex number

- Basic: numeric, character or factor
- Logical: TRUE or FALSE
- Complex: A number with real and imaginary parts
- ou can make a complex number simply by appending an “imaginary” part to an actual number:
- > newvec <- c(1+1i, 2+3i)
- > newvec
- [1] 1+1i 2+3i
- So, R recognises 2+3i for example as a complex number with real part = 2, imaginary part = 3

Characters in R

In R, a piece of text is represented as a sequence of characters (letters, numbers, and symbols). The data type R provides for storing sequences of characters is *character*. Formally, the **mode** of an object that holds character strings in R is `"character"`.
You express character strings by surrounding text within double quotes:

```
"a character string using double quotes"
```

or you can also surround text within single quotes:

```
'a character string using single quotes'
```

The important thing is that you must match the type of quotes that your are using. A starting double quote must have an ending double quote. Likewise, a string with an opening single quote must be closed with a single quote.

Typing characters in R like in above examples is not very useful. Typically, you are going to create objects or variables containing some strings. For example, you can create a variable `string` that stores some string:

```
string <- 'do more with less'
string
#> [1] "do more with less"
```

Notice that when you print a character object, R displays it using double quotes (regardless of whether the string was created using single or double quotes). This allows you to quickly identify when an object contains character values.

When writing strings, you can insert single quotes in a string with double quotes, and vice versa:

```
# single quotes within double quotes
ex1 <- "The 'R' project for statistical computing"
# double quotes within single quotes
ex2 <- 'The "R" project for statistical computing'
```

However, you cannot directly insert single quotes in a string with single quotes, neither you can insert double quotes in a string with double quotes (Don't do this!):

```
ex3 <- "This "is" totally unacceptable"
ex4 <- 'This 'is' absolutely wrong'
```

In both cases R will give you an error due to the unexpected presence of either a double quote within double quotes, or a single quote within single quotes.

If you really want to include a double quote as part of the string, you need to *escape* the double quote using a backslash `\` before it:

```
"The \"R\" project for statistical computing"
```

We will talk more about escaping characters in the following chapters.

Operators

Arithmetic Operators

Following table shows the arithmetic operators supported by R language. The operators act on each element of the vector.

Operator	Description
+	Adds two vectors
−	Subtracts second vector from the first
*	Multiplies both vectors
/	Divide the first vector with the second

%%	Give the remainder of the first vector with the second
%/%	The result of division of first vector with second (quotient)
^	The first vector raised to the exponent of second vector

Relational Operators

Following table shows the relational operators supported by R language. Each element of the first vector is compared with the corresponding element of the second vector. The result of comparison is a Boolean value.

Operator	Description
>	Checks if each element of the first vector is greater than the corresponding element of the second vector.
<	Checks if each element of the first vector is less than the corresponding element of the second vector.
==	Checks if each element of the first vector is equal to the corresponding element of the second vector.
<=	Checks if each element of the first vector is less than or equal to the corresponding element of the second vector.
>=	Checks if each element of the first vector is greater than or equal to the corresponding element of the second vector.
!=	Checks if each element of the first vector is unequal to the corresponding element of the second vector.

Logical Operators

Following table shows the logical operators supported by R language. It is applicable only to vectors of type logical, numeric or complex. All numbers greater than 1 are considered as logical value TRUE.

Each element of the first vector is compared with the corresponding element of the second vector. The result of comparison is a Boolean value.

<table>
<tr><th>Operator</th><th>Description</th></tr>
<tr><td>&</td><td>It is called Element-wise Logical AND operator. It combines each element of the first vector with the corresponding element of the second vector and gives a output TRUE if both the elements are TRUE.</td></tr>
<tr><td>|</td><td>It is called Element-wise Logical OR operator. It combines each element of the first vector with the corresponding element of the second vector and gives a output TRUE if one the elements is TRUE.</td></tr>
<tr><td>!</td><td>It is called Logical NOT operator. Takes each element of the vector and gives the opposite logical value.</td></tr>
</table>

The logical operator && and || considers only the first element of the vectors and give a vector of single element as output.

<table>
<tr><th>Operator</th><th>Description</th></tr>
<tr><td>&&</td><td>Called Logical AND operator. Takes first element of both the vectors and gives the TRUE only if both are TRUE.</td></tr>
<tr><td>||</td><td>Called Logical OR operator. Takes first element of both the vectors and gives the TRUE if one of them is TRUE.</td></tr>
</table>

Assignment Operators

These operators are used to assign values to vectors.

Operator	Description

<− or = or <<−	Called Left Assignment
-> or ->>	Called Right Assignment

Miscellaneous Operators

These operators are used to for specific purpose and not general mathematical or logical computation.

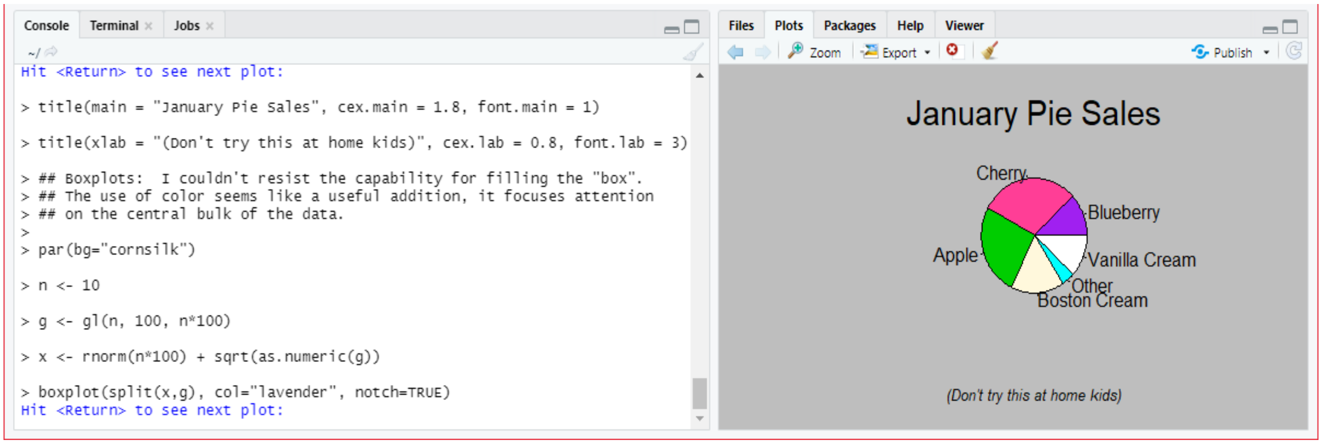
Operator	Description
:	Colon operator. It creates the series of numbers in sequence for a vector.
%in%	This operator is used to identify if an element belongs to a vector.
%*%	This operator is used to multiply a matrix with its transpose.

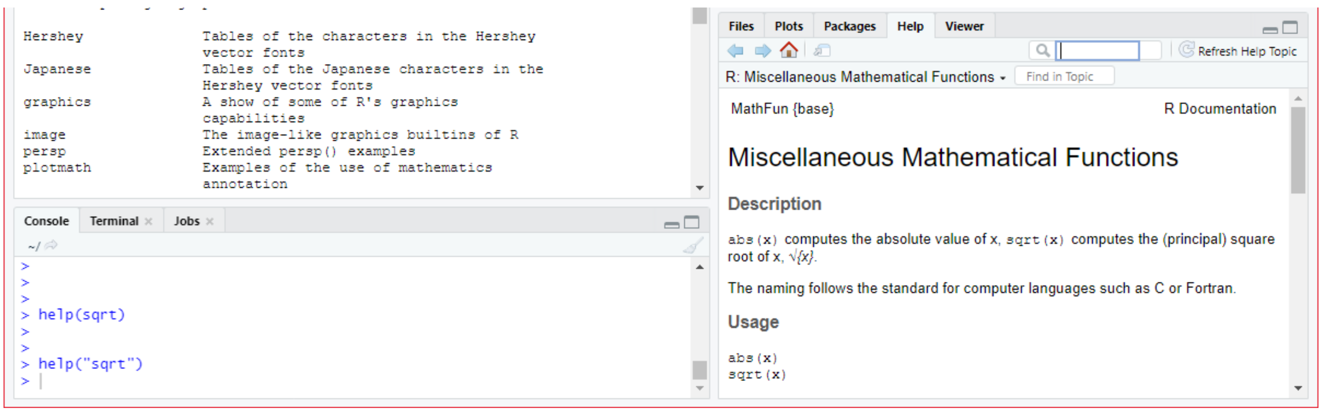
10. Results:

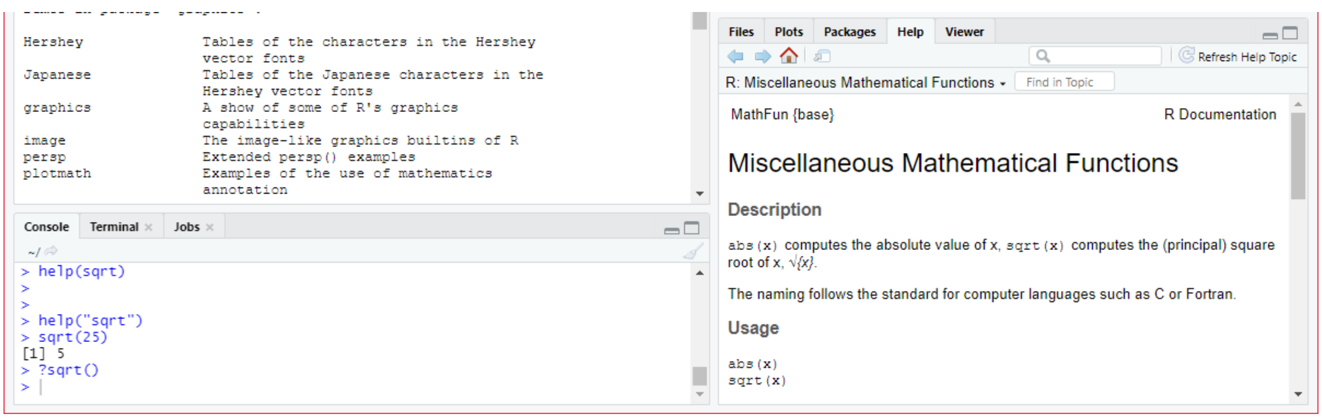
```
> x=5
> y=2
> z=x+y
> print(z)
[1] 7
> print(x-y)
[1] 3
> print("Lets do addition")
[1] "Lets do addition"
> license()

This software is distributed under the terms of the GNU General
Public License, either Version 2, June 1991 or Version 3, June 2007.
The terms of version 2 of the license are in a file called COPYING
which you should have received with
this software and which can be displayed by RShowDoc("COPYING").
Version 3 of the license can be displayed by RShowDoc("GPL-3").

Copies of both versions 2 and 3 of the license can be found
at https://www.R-project.org/Licenses/.

A small number of files (the API header files listed in
R_DOC_DIR/COPYRIGHTS) are distributed under the
LESSER GNU GENERAL PUBLIC LICENSE, version 2.1 or later.
This can be displayed by RShowDoc("LGPL-2.1"),
or obtained at the URI given.
Version 3 of the license can be displayed by RShowDoc("LGPL-3").

'Share and Enjoy.'
```

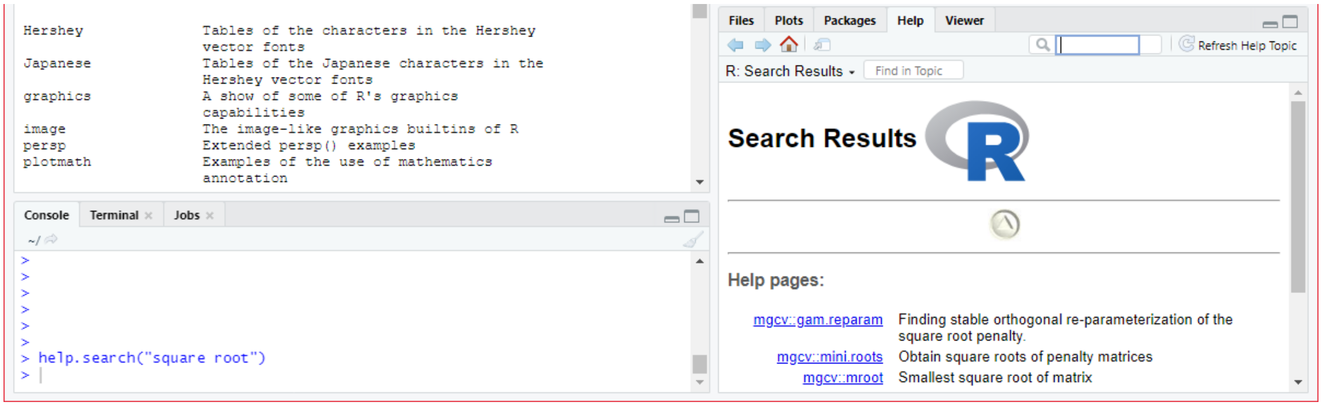
```
> citation()

To cite R in publications use:

  R Core Team (2021). R: A language and environment for
  statistical computing. R Foundation for Statistical Computing,
  Vienna, Austria. URL https://www.R-project.org/.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {R: A Language and Environment for Statistical Computing},
    author = {{R Core Team}},
    organization = {R Foundation for Statistical Computing},
    address = {Vienna, Austria},
    year = {2021},
    url = {https://www.R-project.org/},
  }

We have invested a lot of time and effort in creating R, please
cite it when using it for data analysis. See also
'citation("pkgname")' for citing R packages.

> 
```

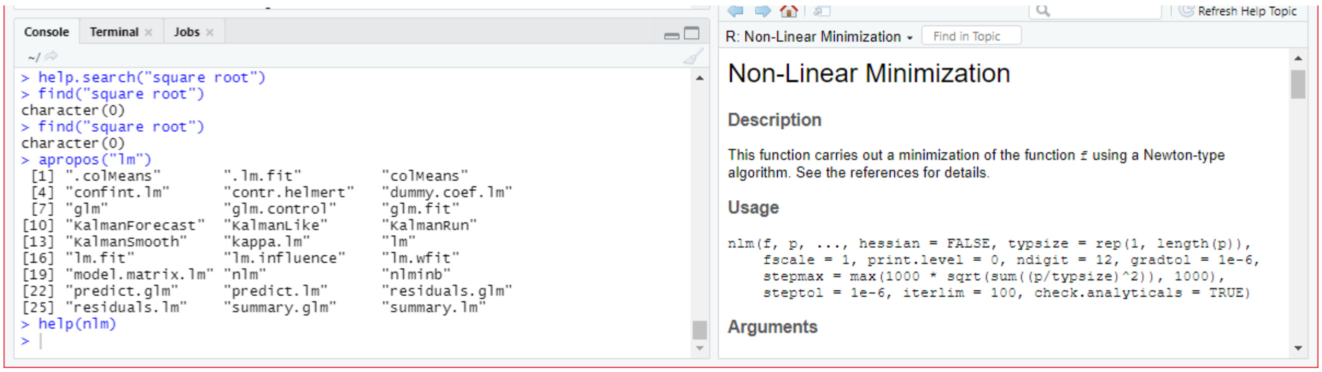
```
Console  Terminal  Jobs
~/
> demo()
> demo(graphics)


        demo(graphics)
        ---- ~~~~~~~~

Type  <Return>   to start :

> #  Copyright (C) 1997-2009 The R Core Team
>
> require(datasets)

> require(grDevices); require(graphics)

> ## Here is some code which illustrates some of the differences between
> ## R and S graphics capabilities.  Note that colors are generally specifie
d
> ## by a character string name (taken from the X11 rgb.txt file) and that l
ine
```

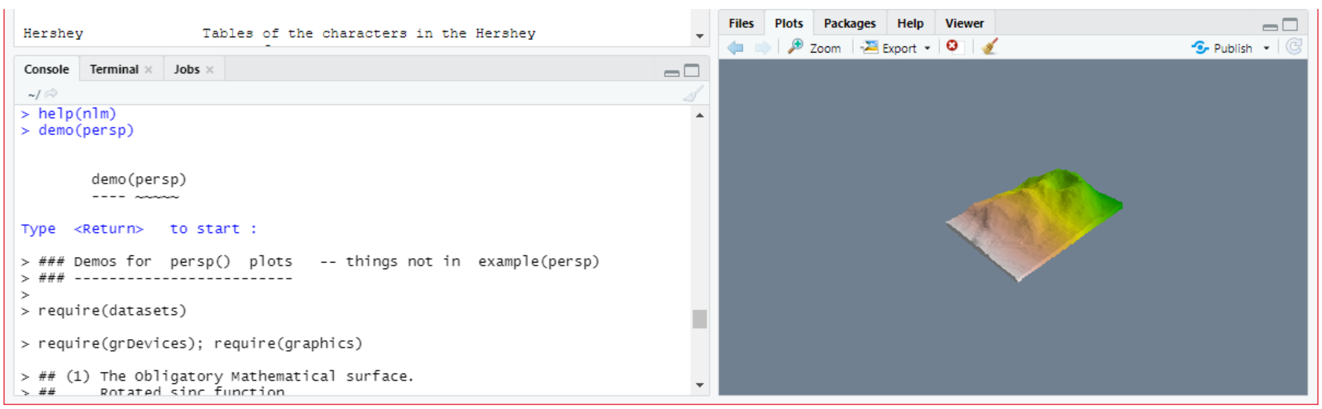

Console
Terminal
Jobs
Hit <Return> to see next plot:
> title(main = "January Pie Sales", cex.main = 1.8, font.main = 1)
> title(xlab = "(Don't try this at home kids)", cex.lab = 0.8, font.lab = 3)
> ## Boxplots: I couldn't resist the capability for filling the "box".
> ## The use of color seems like a useful addition, it focuses attention
> ## on the central bulk of the data.
>
> par(bg="cornsilk")
> n <- 10
> g <- gl(n, 100, n*100)
> x <- rnorm(n*100) + sqrt(as.numeric(g))
> boxplot(split(x,g), col="lavender", notch=TRUE)
Hit <Return> to see next plot:
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
January Pie Sales
Cherry
Blueberry
Apple
Vanilla Cream
Other
Boston Cream
(Don't try this at home kids)

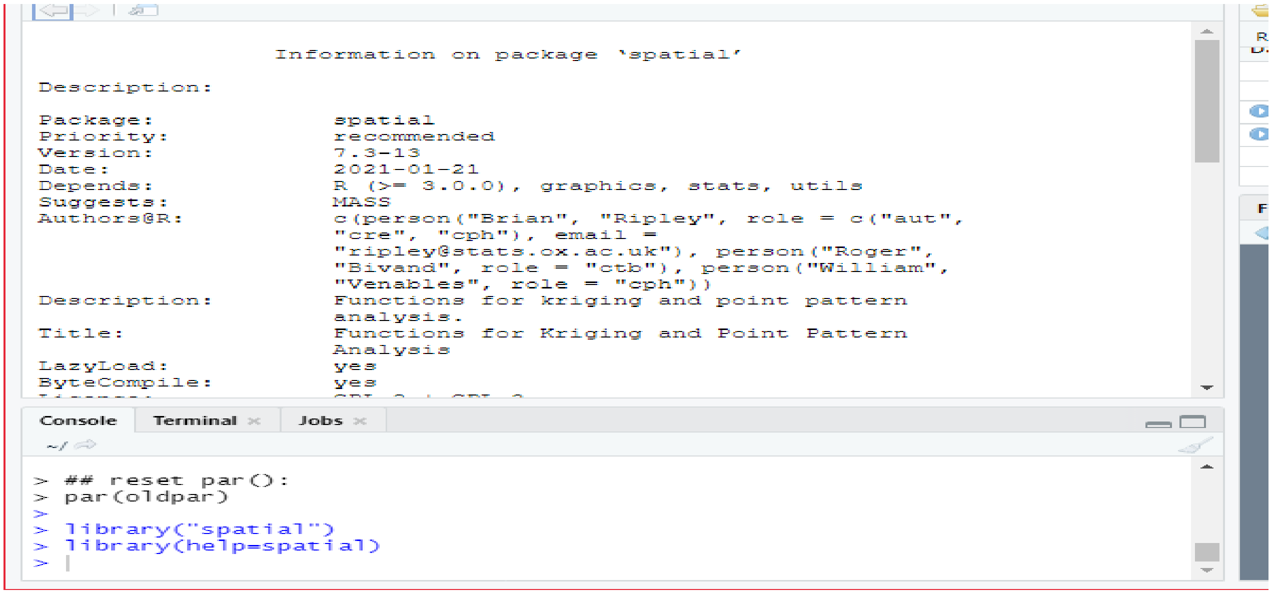

Hershey   Tables of the characters in the Hershey
                         vector fonts
Japanese   Tables of the Japanese characters in the
                         Hershey vector fonts
graphics   A show of some of R's graphics
                         capabilities
image   The image-like graphics builtins of R
persp   Extended persp() examples
plotmath   Examples of the use of mathematics
                         annotation
Console
Terminal
Jobs
>
>
>
> help(sqrt)
>
>
> help("sqrt")
>
Files
Plots
Packages
Help
Viewer
Refresh Help Topic
R: Miscellaneous Mathematical Functions
Find in Topic
MathFun {base}
R Documentation
Miscellaneous Mathematical Functions
Description
abs(x) computes the absolute value of x, sqrt(x) computes the (principal) square root of x, √{x}.
The naming follows the standard for computer languages such as C or Fortran.
Usage
abs(x)
sqrt(x)

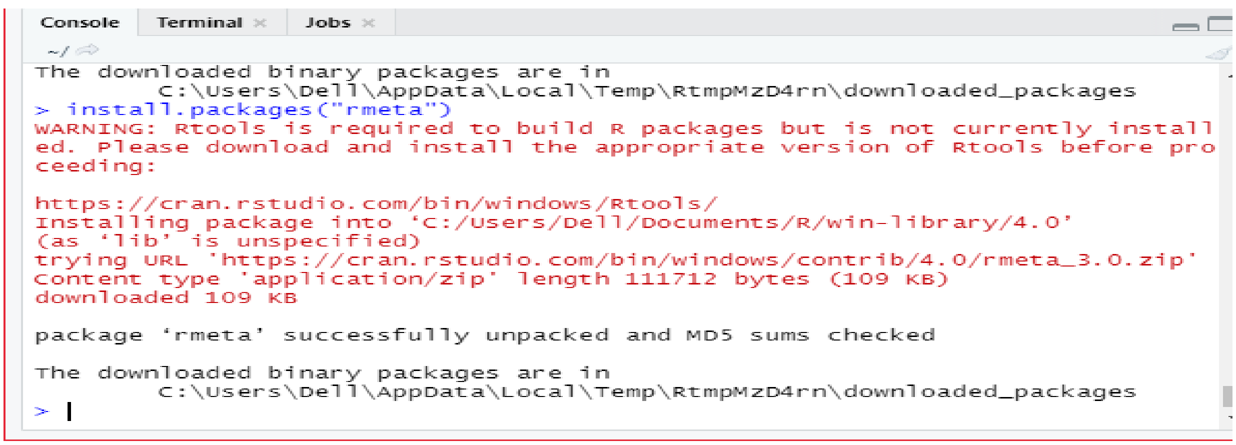

Hershey   Tables of the characters in the Hershey
                         vector fonts
Japanese   Tables of the Japanese characters in the
                         Hershey vector fonts
graphics   A show of some of R's graphics
                         capabilities
image   The image-like graphics builtins of R
persp   Extended persp() examples
plotmath   Examples of the use of mathematics
                         annotation
Console
Terminal
Jobs
> help(sqrt)
>
>
> help("sqrt")
> sqrt(25)
[1] 5
> ?sqrt()
>
Files
Plots
Packages
Help
Viewer
Refresh Help Topic
R: Miscellaneous Mathematical Functions
Find in Topic
MathFun {base}
R Documentation
Miscellaneous Mathematical Functions
Description
abs(x) computes the absolute value of x, sqrt(x) computes the (principal) square root of x, √{x}.
The naming follows the standard for computer languages such as C or Fortran.
Usage
abs(x)
sqrt(x)

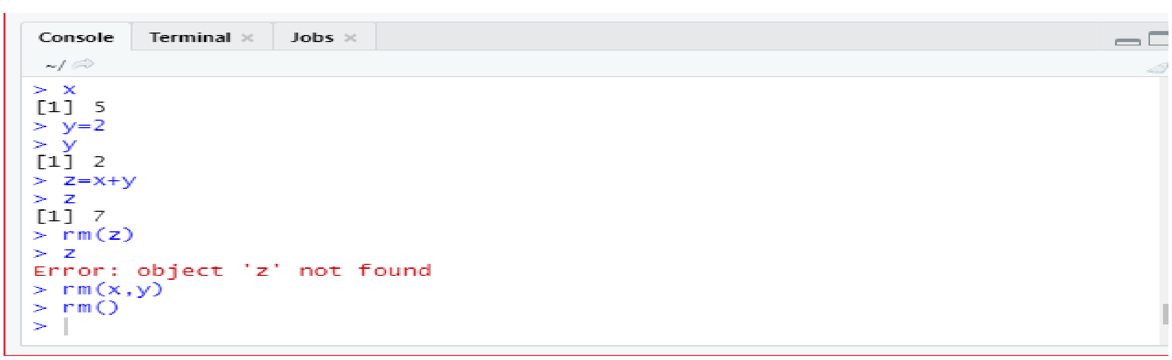

Hershey Tables of the characters in the Hershey vector fonts
Japanese Tables of the Japanese characters in the Hershey vector fonts
graphics A show of some of R's graphics capabilities
image The image-like graphics builtins of R
persp Extended persp() examples
plotmath Examples of the use of mathematics annotation
Console
Terminal
Jobs
> help.search("square root")
Files
Plots
Packages
Help
Viewer
Refresh Help Topic
R: Search Results
Find in Topic
Search Results
Help pages:
mgcv::gam.reparam Finding stable orthogonal re-parameterization of the square root penalty.
mgcv::mini.roots Obtain square roots of penalty matrices
mgcv::mroot Smallest square root of matrix


Console
Terminal
Jobs
> help.search("square root")
> find("square root")
character(0)
> find("square root")
character(0)
> apropos("lm")
[1] ".colMeans" ".lm.fit" "colMeans"
[4] "confint.lm" "contr.helmert" "dummy.coef.lm"
[7] "glm" "glm.control" "glm.fit"
[10] "KalmanForecast" "KalmanLike" "KalmanRun"
[13] "KalmanSmooth" "kappa.lm" "lm"
[16] "lm.fit" "lm.influence" "lm.wfit"
[19] "model.matrix.lm" "nlm" "nlminb"
[22] "predict.glm" "predict.lm" "residuals.glm"
[25] "residuals.lm" "summary.glm" "summary.lm"
> help(nlm)
Refresh Help Topic
R: Non-Linear Minimization
Find in Topic
Non-Linear Minimization
Description
This function carries out a minimization of the function f using a Newton-type algorithm. See the references for details.
Usage
nlm(f, p, ..., hessian = FALSE, typsize = rep(1, length(p)),
fscale = 1, print.level = 0, ndigit = 12, gradtol = 1e-6,
stepmax = max(1000 * sqrt(sum((p/typsize)^2)), 1000),
steptol = 1e-6, iterlim = 100, check.analyticals = TRUE)
Arguments


Hershey Tables of the characters in the Hershey
Console
Terminal
Jobs
> help(nlm)
> demo(persp)
demo(persp)
---- ~~~~~
Type <Return> to start :
> ### Demos for persp() plots -- things not in example(persp)
> ### -------------------------
>
> require(datasets)
> require(grDevices); require(graphics)
> ## (1) The Obligatory Mathematical surface.
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish


Information on package 'spatial'
Description:
Package: spatial
Priority: recommended
Version: 7.3-13
Date: 2021-01-21
Depends: R (>= 3.0.0), graphics, stats, utils
Suggests: MASS
Authors@R: c(person("Brian", "Ripley", role = c("aut", "cre", "cph"), email = "ripley@stats.ox.ac.uk"), person("Roger", "Bivand", role = "ctb"), person("William", "Venables", role = "cph"))
Description: Functions for kriging and point pattern analysis.
Title: Functions for Kriging and Point Pattern Analysis
LazyLoad: yes
ByteCompile: yes
Console
Terminal
Jobs
~/
> ## reset par():
> par(oldpar)
>
> library("spatial")
> library(help=spatial)
>


Console
Terminal
Jobs
~/
The downloaded binary packages are in
C:\Users\Dell\AppData\Local\Temp\RtmpMzD4rn\downloaded_packages
> install.packages("rmeta")
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/Dell/Documents/R/win-library/4.0'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/rmeta_3.0.zip'
Content type 'application/zip' length 111712 bytes (109 KB)
downloaded 109 KB
package 'rmeta' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\Dell\AppData\Local\Temp\RtmpMzD4rn\downloaded_packages
>


Console
Terminal
Jobs
~/
> x
[1] 5
> y=2
> y
[1] 2
> z=x+y
> z
[1] 7
> rm(z)
> z
Error: object 'z' not found
> rm(x,y)
> rm()
>

```
> a=30
> a
[1] 30
> A
Error: object 'A' not found
> a=c(5,2,3,7,6,8)
> a[5]
[1] 6
> a[0]
numeric(0)
> a[-1]
[1] 2 3 7 6 8
> length(a)
[1] 6
>
```

```
> 5%%2
[1] 1
> 11%4
Error: unexpected input in "11%4"
> 11%%4
[1] 3
> 11.9%%4.1
[1] 3.7
> 5.5%%2.1
[1] 1.3
> 5%/%2
[1] 2
> 5^2
[1] 25
> 5**2
[1] 25
> a=c(1,6,7,8,9)
> a^2
[1]  1 36 49 64 81
> b=a^3
> b
[1]   1 216 343 512 729
> c=a+b
> c
[1]   2 222 350 520 738
>
```

```
> x=5
> cat("x=",x)
x= 5
> cat("x^2=",x^2)
x^2= 25
> 4&3
[1] TRUE
> bitwAnd(7,11)
[1] 3
>
```

```
> c
[1] 5+3i
> b=8+5i
> a-b+c
[1] -2-2i  3-2i  4-2i  5-2i  6-2i
> flag=TRUE
> class(flag)
[1] "logical"
> mode(flag)
[1] "logical"
> TRUE
[1] TRUE
> FALSE
[1] FALSE
> T
[1] TRUE
> F
```

```
> x=6
> y=2
> cat("x>=4&&y<=3",x>=4&&y<=3)
x>=4&&y<=3 TRUE
> cat("x>=4||y<=3",x>=4||y<=3)
x>=4||y<=3 TRUE
> cat("x>=4||y<=3",x>=4||y<=3)
```

11. Learning Outcomes Achieved:

1. understood R software as a software development platform.
2. understood elementary building blocks of R software such as- data types, number, character, complex, vectors,, helps, packages.

12. Conclusion:

Demonstrated basic functionality of R such as- helps, accessing packages, data types, number, complex, characters, basic operators

13. Experiment/Assignment Evaluation

<table>
<tr><td colspan="5">Experiment/Assignment Evaluation:</td></tr>
<tr><td>Sr. No.</td><td colspan="2">Parameters</td><td>Marks obtained</td><td>Out of</td></tr>
<tr><td>1</td><td colspan="2">Technical Understanding (Assessment may be done based on Q & A <u>or</u> any other relevant method.) Teacher should mention the other method used -</td><td></td><td>6</td></tr>
<tr><td>2</td><td colspan="2">Neatness/presentation</td><td></td><td>2</td></tr>
<tr><td>3</td><td colspan="2">Punctuality</td><td></td><td>2</td></tr>
<tr><td>Date of performance (DOP)</td><td></td><td>Total marks obtained</td><td></td><td>10</td></tr>
<tr><td>Date of checking (DOC)</td><td></td><td colspan="3">Signature of teacher</td></tr>
</table>

References:

1. URL: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf (Online Resources)
2. R Cookbook Paperback – 2011 by Teetor Paul O Reilly Publications
3. Beginning R: The Statistical Programming Language by Dr. Mark Gardener, Wiley Publications
4. R Programming For Dummies by Joris Meys Andrie de Vries, Wiley Publications

Viva Questions

1. What is R ?
2. How is R different than Python?
3. What are different data-types in R?
4. How to define a string in R?
5. What is factor data class in R?
6. How to take help in R?
7. How to load packages and libraries in R?

Subject:	**R Programming Lab. (ITL804)**		
Class:	**BE IT / Semester – VIII (Rev-2016) / Academic year: 2020-21**		
Name of Student:	**Pranali Hanumant Kudtarkar**		
Roll No:	**29**	**Date of performance (DOP) :**	
Assignment/Experiment No:	**02**	**Date of checking (DOC) :**	
Title: Program to demonstrate data structures such as- vectors, matrix, list, data frames and factors.			
Marks:		**Teacher's Signature:**	

1. Aim: To understand the use of vectors, matrix, list and data frames in R.

2. Prerequisites:

1. Basics of R programming.

3. Hardware Requirements:

1. PC with minimum 2GB RAM

4. Software Requirements:

1. Windows / Linux OS.
2. R version 3.6 or higher

5. Learning Objectives:

1. To understand vectors, matrices and lists.
2. To understand *data frames* which are mainly required for data analysis in R.

6. Learning Objectives Applicable: LO 1, LO 2

7. Program Outcomes Applicable: PO 1

8. Program Education Objectives Applicable: PEO 1, PEO 2

9. Theory:

Vectors

The basic data structure in R is the vector. Vectors come in two flavours: atomic vectors and lists. They have three common properties:

- Type, typeof(), what it is.
- Length, length(), how many elements it contains.
- Attributes, attributes(), additional arbitrary metadata.

They differ in the types of their elements: all elements of an atomic vector must be the same type, whereas the elements of a list can have different types.

NB: is.vector() does not test if an object is a vector. Instead it returns TRUE only if the object is a vector with no attributes apart from names. Use is.atomic(x) || is.list(x) to test if an object is actually a vector.

Atomic vectors

There are four common types of atomic vectors that I'll discuss in detail: logical, integer, double (often called numeric), and character. There are two rare types that I will not discuss further: complex and raw.

Lists

Lists are different from atomic vectors because their elements can be of any type, including lists. You construct lists by using list() instead of c():

```
x <- list(1:3, "a", c(TRUE, FALSE, TRUE), c(2.3, 5.9))
str(x)
#> List of 4
#>  $ : int [1:3] 1 2 3
#>  $ : chr "a"
#>  $ : logi [1:3] TRUE FALSE TRUE
#>  $ : num [1:2] 2.3 5.9
```

Lists are sometimes called **recursive** vectors, because a list can contain other lists. This makes them fundamentally different from atomic vectors.

```
x <- list(list(list(list())))
str(x)
#> List of 1
#>  $ :List of 1
#>   ..$ :List of 1
#>   .. ..$ : list()
is.recursive(x)
```

```
#> [1] TRUE
```

Factors

One important use of attributes is to define factors. A factor is a vector that can contain only predefined values, and is used to store categorical data. Factors are built on top of integer vectors using two attributes: the class, “factor”, which makes them behave differently from regular integer vectors, and the levels, which defines the set of allowed values.

Matrices and arrays

Adding a dim attribute to an atomic vector allows it to behave like a multi-dimensional **array**. A special case of the array is the **matrix**, which has two dimensions. Matrices are used commonly as part of the mathematical machinery of statistics. Arrays are much rarer, but worth being aware of.

Matrices and arrays are created with matrix() and array(), or by using the assignment form of dim():

```
# Two scalar arguments to specify rows and columns
a <- matrix(1:6, ncol = 3, nrow = 2)
# One vector argument to describe all dimensions
b <- array(1:12, c(2, 3, 2))


# You can also modify an object in place by setting dim()
c <- 1:6
dim(c) <- c(3, 2)
c
#>      [,1] [,2]
#> [1,]    1    4
#> [2,]    2    5
#> [3,]    3    6
dim(c) <- c(2, 3)
c
#>      [,1] [,2] [,3]
#> [1,]    1    3    5
#> [2,]    2    4    6
```

length() and names() have high-dimensional generalisations:

- length() generalises to nrow() and ncol() for matrices, and dim() for arrays.
- names() generalises to rownames() and colnames() for matrices, and dimnames(), a list of character vectors, for arrays.

```
length(a)
```

```
#> [1] 6
nrow(a)
#> [1] 2
ncol(a)
#> [1] 3
rownames(a) <- c("A", "B")
colnames(a) <- c("a", "b", "c")
a
#>   a b c
#> A 1 3 5
#> B 2 4 6

length(b)
#> [1] 12
dim(b)
#> [1] 2 3 2
dimnames(b) <- list(c("one", "two"), c("a", "b", "c"), c("A", "B"))
b
#> , , A
#>
#>     a b c
#> one 1 3 5
#> two 2 4 6
#>
#> , , B
#>
#>     a  b  c
#> one 7  9 11
#> two 8 10 12
```

c() generalises to cbind() and rbind() for matrices, and to abind() (provided by the abind package) for arrays. You can transpose a matrix with t(); the generalised equivalent for arrays is aperm().

You can test if an object is a matrix or array using is.matrix() and is.array(), or by looking at the length of the dim(). as.matrix() and as.array() make it easy to turn an existing vector into a matrix or array.

Vectors are not the only 1-dimensional data structure. You can have matrices with a single row or single column, or arrays with a single dimension. They may print similarly, but will behave differently. The differences aren't too important, but it's useful to know they exist in case you get strange output from a function (tapply() is a frequent offender). As always, use str() to reveal the differences.

```
str(1:3)                  # 1d vector
#>  int [1:3] 1 2 3
str(matrix(1:3, ncol = 1)) # column vector
#>  int [1:3, 1] 1 2 3
str(matrix(1:3, nrow = 1)) # row vector
#>  int [1, 1:3] 1 2 3
str(array(1:3, 3))        # "array" vector
#>  int [1:3(1d)] 1 2 3
```

While atomic vectors are most commonly turned into matrices, the dimension attribute can also be set on lists to make list-matrices or list-arrays:

```
l <- list(1:3, "a", TRUE, 1.0)
dim(l) <- c(2, 2)
l
#>      [,1]      [,2]
#> [1,] Integer,3 TRUE
#> [2,] "a"       1
```

These are relatively esoteric data structures, but can be useful if you want to arrange objects into a grid-like structure. For example, if you're running models on a spatio-temporal grid, it might be natural to preserve the grid structure by storing the models in a 3d array.

Data frames

A data frame is the most common way of storing data in R, and if used systematically makes data analysis easier. Under the hood, a data frame is a list of equal-length vectors. This makes it a 2-dimensional structure, so it shares properties of both the matrix and the list. This means that a data frame has names(), colnames(), and rownames(), although names() and colnames() are the same thing. The length() of a data frame is the length of the underlying list and so is the same as ncol(); nrow() gives the number of rows.

As described in subsetting, you can subset a data frame like a 1d structure (where it behaves like a list), or a 2d structure (where it behaves like a matrix).

Creation

You create a data frame using data.frame(), which takes named vectors as input:

```
df <- data.frame(x = 1:3, y = c("a", "b", "c"))
str(df)
#> 'data.frame':    3 obs. of  2 variables:
#>  $ x: int  1 2 3
#>  $ y: Factor w/ 3 levels "a","b","c": 1 2 3
```

Beware data.frame()'s default behaviour which turns strings into factors. Use stringsAsFactors = FALSE to suppress this behaviour:

```
df <- data.frame(
  x = 1:3,
  y = c("a", "b", "c"),
  stringsAsFactors = FALSE)
str(df)
#> 'data.frame':    3 obs. of  2 variables:
#>  $ x: int  1 2 3
#>  $ y: chr  "a" "b" "c"
```

10. Results:

a=c(3,1,5,4,6,7)

b=c(3,2,6,5,7,8)

x=a+b

print(x)

a=c(3,1,5,4,6,7)

b=c(3,2)

x=a+b

print(x)

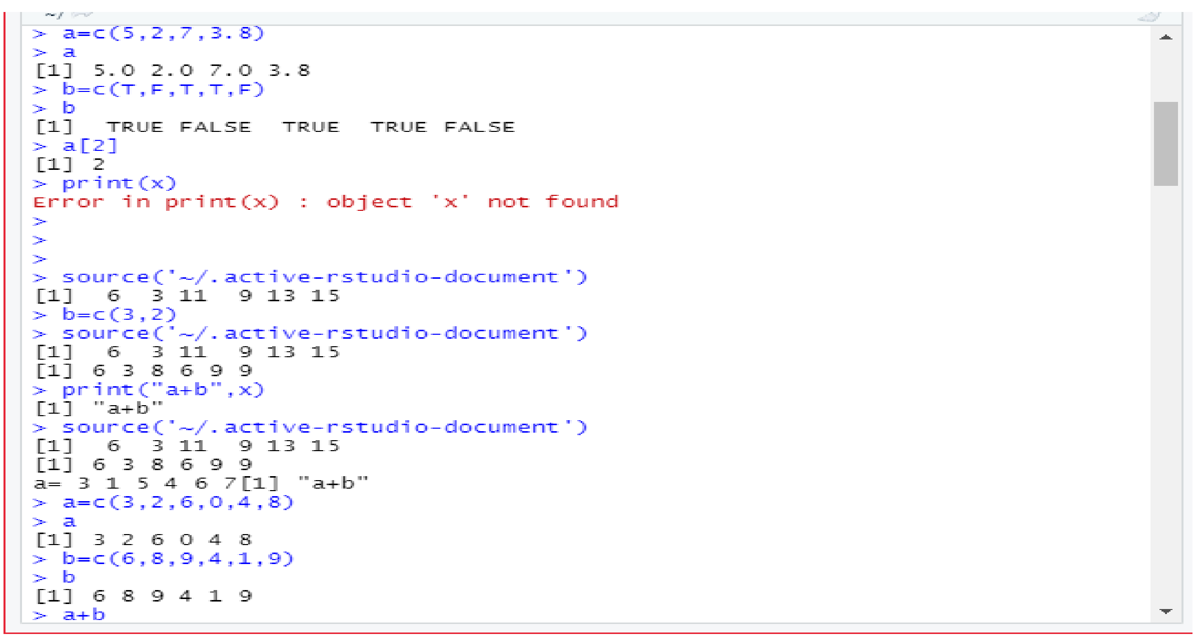
a=c(3,1,5,4,6,7)

cat("a=",a)

b=c(3,2)

x=a+b

print("a+b",x)



```
> a=c(5,2,7,3.8)
> a
[1] 5.0 2.0 7.0 3.8
> b=c(T,F,T,T,F)
> b
[1]  TRUE FALSE  TRUE  TRUE FALSE
> a[2]
[1] 2
> print(x)
Error in print(x) : object 'x' not found
>
>
>
> source('~/.active-rstudio-document')
[1]  6  3 11  9 13 15
> b=c(3,2)
> source('~/.active-rstudio-document')
[1]  6  3 11  9 13 15
[1] 6 3 8 6 9 9
> print("a+b",x)
[1] "a+b"
> source('~/.active-rstudio-document')
[1]  6  3 11  9 13 15
[1] 6 3 8 6 9 9
a= 3 1 5 4 6 7[1] "a+b"
> a=c(3,2,6,0,4,8)
> a
[1] 3 2 6 0 4 8
> b=c(6,8,9,4,1,9)
> b
[1] 6 8 9 4 1 9
> a+b
```

```
Console   Terminal ×   Jobs ×
~/
> a+b
[1]  9 10 15  4  5 17
> a%%b
[1] 3 2 6 0 0 8
> a*b
[1] 18 16 54  0  4 72
> a^b
[1]       729       256  10077696         0         4 134217728
> a^2
[1]  9  4 36  0 16 64
> a<b
[1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
> c=c(4,6,7,0)
> a+c
[1]  7  8 13  0  8 14
Warning message:
In a + c : longer object length is not a multiple of shorter object length
> max(a)
[1] 8
> round(1.5)
[1] 2
> sum(a)
[1] 23
> prod(a)
[1] 0
>
```

```
Console   Terminal ×   Jobs ×
~/
>
> A = matrix(data=c(4,7,2,9,0,4),nrow=2,ncol=3)
> A
     [,1] [,2] [,3]
[1,]    4    2    0
[2,]    7    9    4
> A = matrix(data=c(4,7,2,9,0,4),nrow=2,ncol=3,byrow=TRUE)
> A
     [,1] [,2] [,3]
[1,]    4    7    2
[2,]    9    0    4
> A = matrix(c(4,7,2,9,0,4),2,3,TRUE)
> A
     [,1] [,2] [,3]
[1,]    4    7    2
[2,]    9    0    4
> A = matrix(nrow=2,ncol=3,byrow=TRUE,data=c(4,7,2,9,0,4))
> A
     [,1] [,2] [,3]
[1,]    4    7    2
[2,]    9    0    4
> A[2,2]
[1] 0
> B=matrix(c(1,2,3,4,5,6),2,3)
> B
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
Console   Terminal ×   Jobs ×
~/
[2,]    2    4    6
> C=A+B
> C
     [,1] [,2] [,3]
[1,]    5   10    7
[2,]   11    4   10
> V=matrix(c(7,8),2,1)
> V
     [,1]
[1,]    7
[2,]    8
> A+V
Error in A + V : non-conformable arrays
> nrow(A)
[1] 2
> ncol(A)
[1] 3
> dim(A)
[1] 2 3
> is.matrix(A)
[1] TRUE
> x
[1] 6 3 8 6 9 9
> is.matrix(x)
[1] FALSE
```

```
Console   Terminal ×   Jobs ×
~/
[1] FALSE
> I=diag(3,nrow=5,ncol=5)
> I
     [,1] [,2] [,3] [,4] [,5]
[1,]    3    0    0    0    0
[2,]    0    3    0    0    0
[3,]    0    0    3    0    0
[4,]    0    0    0    3    0
[5,]    0    0    0    0    3
> D=matrix((data=10,nrow=4,ncol=5))
Error: unexpected ',' in "D=matrix((data=10,"
> D=matrix(data=10,nrow=4,ncol=5)
> D
     [,1] [,2] [,3] [,4] [,5]
[1,]   10   10   10   10   10
[2,]   10   10   10   10   10
[3,]   10   10   10   10   10
[4,]   10   10   10   10   10
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> 3:17
 [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
> x=matrix(1:15,5,3)
> x
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```

Console Terminal Jobs

~/

```
> x[2:4,1:2]
     [,1] [,2]
[1,]    2    7
[2,]    3    8
[3,]    4    9
> z=x[2:4,1:2]
> z
     [,1] [,2]
[1,]    2    7
[2,]    3    8
[3,]    4    9
> z=list(23,"FAMT",TRUE,c(2,3,6,7))
> z
[[1]]
[1] 23

[[2]]
[1] "FAMT"

[[3]]
[1] TRUE

[[4]]
[1] 2 3 6 7

> z[[1]]
[1] 23
> Y=list(name="rani",Age=23,Height=167)
> Y$name
[1] "rani"
> Y$Age
[1] 23
>
```

```
> A[2,3]="FAMT"
> A
     [,1] [,2] [,3]
[1,] "4"  "7"  "2"
[2,] "9"  "0"  "FAMT"
> x=list(5,3,c(5,3,4,7,6),matrix(1:10,5,2))
> x[[3]]
[1] 5 3 4 7 6
> x[[4]]
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> x=list(Rollno=22,name="mahi",Marks=88.97)
> x$name
[1] "mahi"
```

```
> f=data.frame(Roll=c(1,2,3), Name=c("A","B","C"), AGE=c(21,34,22))
> f
  Roll Name AGE
1    1    A  21
2    2    B  34
3    3    C  22
> f[2,3]
[1] 34
> f$Name
[1] "A" "B" "C"
> F$Roll
Error in F$Roll : $ operator is invalid for atomic vectors
> f$Roll
[1] 1 2 3
> f$name[2]
NULL
> f$AGE[2]
[1] 34
> f$name[2]
NULL
> f$name[1]
NULL
> f$Name[1]
[1] "A"
> f$Roll
[1] 1 2 3
> f$Name[1]
[1] "A"
> f$Name[1]
[1] "A"
> f$AGE[2]
[1] 34
> subset(f,AGE<=23)
```

```
> subset(f,AGE<=23)
  Roll Name AGE
1    1    A  21
3    3    C  22
> f1=subset(f,AGE<=23)
> f1
  Roll Name AGE
1    1    A  21
3    3    C  22
> y=c(6,2,3,8,7,6)
> y
[1] 6 2 3 8 7 6
> y=c(5,2,3,1,5)
> y
[1] 5 2 3 1 5
```

```
> z=factor(y,levels=c(1,2,3,4,5,6,7), labels=c("MOn","Tue","Wed","thu","fr
i","sat","sun"))
> z
[1] fri Tue Wed MOn fri
Levels: MOn Tue Wed thu fri sat sun
> mode(z)
[1] "numeric"
> class(x)
[1] "list"
> class(z)
[1] "factor"
>
```

11. Learning Outcomes Achieved:

1. understood vectors, matrices and lists.
2. understood *data frames* which are mainly required for data analysis in R.

12. Conclusion:

Demonstrated data structures such as- vectors, matrix, list, data frames and factors

13. Experiment/Assignment Evaluation

Experiment/Assignment Evaluation:					
Sr. No.	**Parameters**			**Marks obtained**	**Out of**
1	Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used -				6
2	Neatness/presentation				2
3	Punctuality				2
Date of performance (DOP)			**Total marks obtained**		**10**
Date of checking (DOC)			**Signature of teacher**		

References:

1. URL: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf (Online Resources)
2. R Cookbook Paperback – 2011 by Teetor Paul O Reilly Publications
3. Beginning R: The Statistical Programming Language by Dr. Mark Gardener, Wiley Publications
4. R Programming For Dummies by Joris Meys Andrie de Vries, Wiley Publications

Viva Questions

1. What is vector in R ?
2. How to create matrix in R ?
3. What is difference between vector and list?
4. How is the data-frame different than matrix?
5. What is importance of data-frames in R?

Subject:	**R Programming Lab. (ITL804)**		
Class:	**BE IT / Semester – VIII (Rev-2016) / Academic year: 2020-21**		
Name of Student:	**Pranali Hanumant Kudtarkar**		
Roll No:	**26**	**Date of performance (DOP) :**	
Assignment/Experiment No:	**03**	**Date of checking (DOC) :**	
Title: Program to demonstrate flow control instructions and functions			
Marks:		**Teacher's Signature:**	

1. Aim: To understand the use of various flow control instructions and functions in R.

2. Prerequisites:

1. Basics of R programming, various data structures used in R etc.

3. Hardware Requirements:

1. PC with minimum 2GB RAM

4. Software Requirements:

1. Windows / Linux OS.
2. R version 3.6 or higher

5. Learning Objectives:

1. To understand decision and loop control instructions.
2. To understand function definition and calling to it..

6. Learning Objectives Applicable: LO 1

7. Program Outcomes Applicable: PO 1, PO 2

8. Program Education Objectives Applicable: PEO 2

9. Theory:

Control Flow in R

R provides the following decision-making statements:

If Statement

It is one of the control statements in R programming that consists of a Boolean expression and a set of statements. If the Boolean expression evaluates to TRUE, the set of statements is executed. If the Boolean expression evaluates to FALSE, the statements after the end of the If statement are executed. The basic syntax for the If statement is given below:

```
if(Boolean_expression) {
This block of code will execute if the Boolean expression returns TRUE.
}
```

For example:

```
x <- “Intellipaat”
if(is.character(x)) {
print("X is a Character")
}
```

Output:[1] “X is a Character”

Else Statement

In the If -Else statement, an If statement is followed by an Else statement, which contains a block of code to be executed when the Boolean expression in the If the statement evaluates to FALSE. The basic syntax of it is given below:

```
if(Boolean_expression) {
This block of code executes if the Boolean expression returns TRUE.
} else {
This block of code executes if the Boolean expression returns FALSE.
}
```

For example:

```
x <- c("Intellipaat","R","Tutorial")
if("Intellipaat" %in% x) {
print("Intellipaat")
} else {
print("Not found")
}
```

Output: [1] “Intellipaat”

Else If Statement

An Else if statement is included between If and Else statements. Multiple Else-If statements can be included after an If statement. Once an If statement or an Else if statement evaluates to TRUE, none of the remaining Else if or Else statement will be evaluated.
The basic syntax of it is given below:

```
if(Boolean_expression1) {
This block of code executes if the Boolean expression 1 returns TRUE
 } else if(Boolean_expression2) {
This block of code executes if the Boolean expression 2 returns TRUE
 } else if(Boolean_expression3) {
This block of code executes if the Boolean expression returns TRUE
} else {
This block of code executes if none of the Boolean expression returns TRUE
}
```

For example:

```
x <- c("Intellipaat","R","Tutorial")
if("Intellipaat" %in% x) {
print("Intellipaat")
} else if ("Tutorial" %in% x)
print("Tutorial")
} else {
print("Not found")}
```

Output:[1] “Intellipaat”

Switch Statement

Switch statement is one of the control statements in R programming which is used to equate a variable against a set of values. Each value is called a case.
Basic syntax for a switch statement is as follows:

```
switch(expression, case1, case2, case3....)
```

For example:

```
x <- switch(
3,
"Intellipaat",
"R",
"Tutorial",
```

```
"Beginners"
)
print(x)
```

Output:[1] "Tutorial"

If the value passed as an expression is not a character string, then it is coerced to an integer and is compared with the indexes of cases provided in the switch statement.

```
y <-  "12"
x <- switch(
y,
"9"= "Good Morning",
"12"= "Good Afternoon",
"18"= "Good Evening",
"21"= "Good Night"
)
print(x)
```

Output:[1] "Good Afternoon"

If an expression evaluates to a character string, then it is matched (exactly) to the names of the cases mentioned in the switch statement.

- If there is more than one match, the first matching element is returned.
- No default argument is available.

Loops

The function of a looping statement is to execute a block of code, several times and to provide various control structures that allow for more complicated execution paths than a usual sequential execution. The types of loops in R are as follows:

Repeat Loop

A repeat loop is one of the control statements in R programming that executes a set of statements in a loop until the exit condition specified in the loop, evaluates to TRUE. Basic syntax for a repeat loop is given below:

```
repeat {
statements
if(exit_condition) {
break
```

```
}
}
```

For example:

```
v <- 9
repeat {
print(v)
v=v-1
if(v < 1) {
break
}
}
```

Output:

```
[1] 9
[1] 8
[1] 7
[1] 6
[1] 5
[1] 4
[1] 3
[1] 2
[1] 1
```

If we don't place a break condition in the repeat loop statement, the statements in the repeat block will get executed in an infinite loop.

While Loop

A while loop is one of the control statements in R programming which executes a set of statements in a loop until the condition (the Boolean expression) evaluates to TRUE. Basic syntax of a while loop is given below

```
while (Boolean_expression) {
statement
}
```

For example:

```
v <-9
while(v>5){
print(v)
```

```
v = v-1
}
```

Output:

```
[1] 9
[1] 8
[1] 7
[1] 6
```

For Loop

For loop is one of the control statements in R programming that executes a set of statements in a loop for a specific number of times, as per the vector provided to it. Basic syntax of a for loop is given below

```
for (value in vector) {
statements
}
```

For example:

```
v <- c(1:5)
for (i in v) {
print(i)
}
```

Output:

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

We can also use the break statement inside a for-loop to break it out abruptly. For example:

```
v <- c(1:5)
for (i in v) {
if(i == 3){
break
}
print(i)
}
```

Output:[1] 1
[1] 2

Loop-control Statements

Loop-control statements are part of control statements in R programming that are used to change the execution of a loop from its normal execution sequence.
There are two loop-control statements in R

Break Statement

A break statement is used for two purposes

-
    -
        - To terminate a loop immediately and resume at the next statement following the loop.
        - To terminate a case in a switch statement.

For example:

```
v <- c(0:6)
for (i in v) {
if(i == 3){
break
}
print(i)
}
```

Output:

```
[1] 0
[1] 1
[1] 2
```

Next Statement

A next statement is one of the control statements in R programming that is used to skip the current iteration of a loop without terminating the loop. Whenever a next statement is encountered, further evaluation of the code is skipped and the next iteration of the loop starts.
For example:

```
v <- c(0:6)
for (i in v) {
```

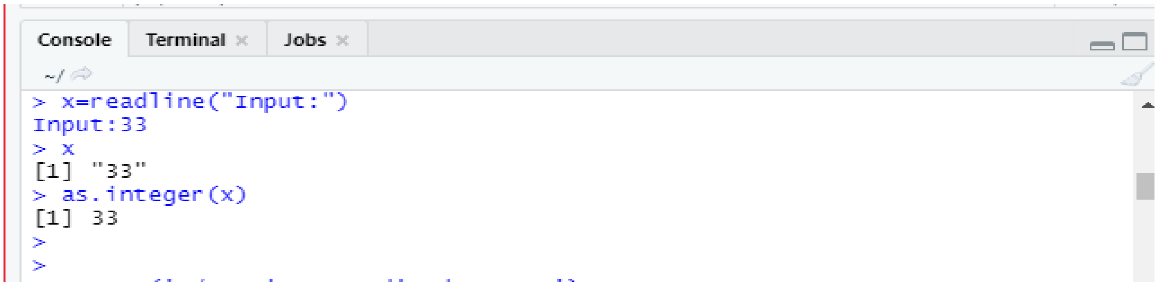
```
if(i == 3){
next
}
print(i)
}
```

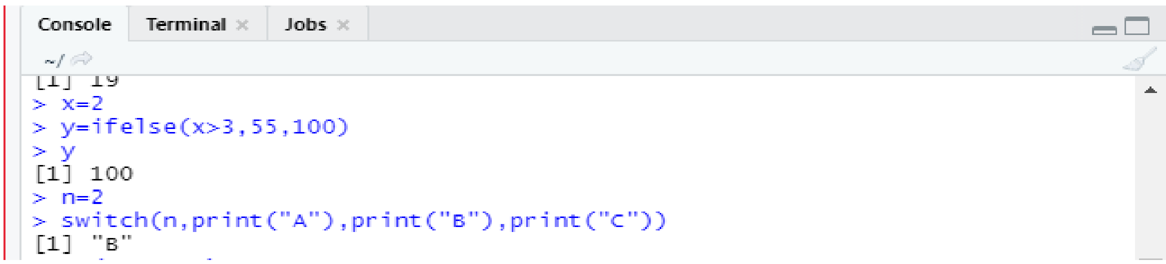
Output:

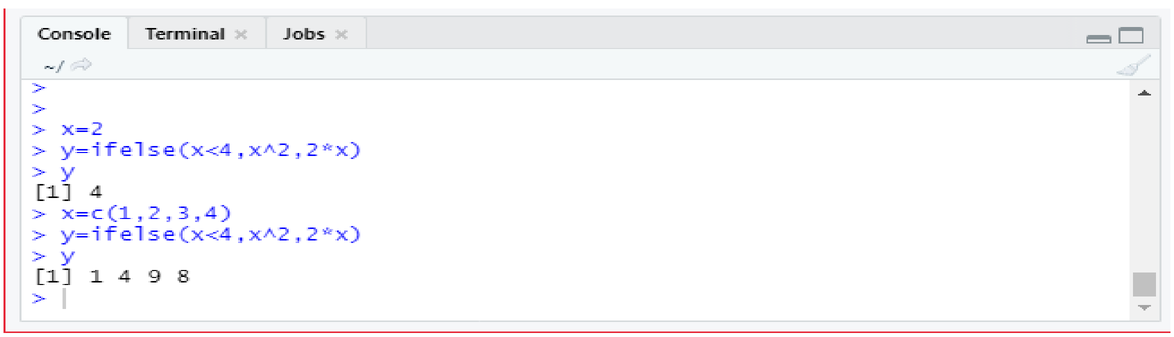
```
[1] 0
[1] 1
[1] 2
[1] 4
[1] 5
[1] 6
```

In this tutorial, we learned what control statements in R programming are, what decision making is, different decision-making statements in R, and how to use these statements to change the order of execution of a program. We also covered loops, different types of loops, and loop-control statements. In the next session, we are going to talk about the functions and types of functions in R

10. Results:


Console
Terminal
Jobs
~/
> x=readline("Input:")
Input:33
> x
[1] "33"
> as.integer(x)
[1] 33
>
>


Console
Terminal
Jobs
~/
> x=2
> y=ifelse(x>3,55,100)
> y
[1] 100
> n=2
> switch(n,print("A"),print("B"),print("C"))
[1] "B"


Console
Terminal
Jobs
~/
>
>
> x=2
> y=ifelse(x<4,x^2,2*x)
> y
[1] 4
> x=c(1,2,3,4)
> y=ifelse(x<4,x^2,2*x)
> y
[1] 1 4 9 8
>

Program

```
K=4
if(K>=4)
{
  print("hii")
}else
{
  print("hello")
}

print("")
x=c(1,2,3,4,5,6,7,8,9)
for(i in x)
{
  print(i)
}
print("")
for(i in 1:10)
{
  print(i)
}
print("")
for(i in seq(1,20,3))
{
  print(i)
}
print("")
i=1
while(i<6)
{
  print(i)
  i=i+1
}
```

```
[1] "hii"
[1] ""
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] ""
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] ""
[1] 1
[1] 4
[1] 7
[1] 10
[1] 13
[1] 16
[1] 19
[1] ""
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
>
```

11. Learning Outcomes Achieved:

1. Understood decision and loop control instructions.
2. Understood function definition and calling to it..

12. Conclusion:

Demonstrated flow control instructions and functions

13. Experiment/Assignment Evaluation

<table>
<tr><td colspan="5">Experiment/Assignment Evaluation:</td></tr>
<tr><td>Sr. No.</td><td colspan="2">Parameters</td><td>Marks obtained</td><td>Out of</td></tr>
<tr><td>1</td><td colspan="2">Technical Understanding (Assessment may be done based on Q & A <u>or</u> any other relevant method.) Teacher should mention the other method used -</td><td></td><td>6</td></tr>
<tr><td>2</td><td colspan="2">Neatness/presentation</td><td></td><td>2</td></tr>
<tr><td>3</td><td colspan="2">Punctuality</td><td></td><td>2</td></tr>
<tr><td>Date of performance (DOP)</td><td></td><td>Total marks obtained</td><td></td><td>10</td></tr>
<tr><td>Date of checking (DOC)</td><td></td><td colspan="3">Signature of teacher</td></tr>
</table>

References:

1. URL: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf (Online Resources)
2. R Cookbook Paperback – 2011 by Teetor Paul O Reilly Publications
3. Beginning R: The Statistical Programming Language by Dr. Mark Gardener, Wiley Publications
4. R Programming For Dummies by Joris Meys Andrie de Vries, Wiley Publications

Viva Questions

1. What are decision control instructions ?
2. What are loop control instructions ?
3. Compare flow control instructions in R with flow control instructions in Python ?
4. How to define function in R?
5. Can I shuffle arguments of the functions while calling it?

\

Finolex Academy of Management and Technology, Ratnagiri

Department of Information Technology

Subject:	**R Programming Lab. (ITL804)**		
Class:	**BE IT / Semester – VIII (Rev-2016) / Academic year: 2020-21**		
Name of Student:	**Pranali Hanumant Kudtarkar**		
Roll No:	**26**	**Date of performance (DOP) :**	
Assignment/Experiment No:	**04**	**Date of checking (DOC) :**	
Title: Exploratory data analysis such as- Range, summary, mean, variance, median, standard deviation, histogram, boxplot, scatterplot			
Marks:		**Teacher's Signature:**	

1. Aim: To understand the exploratory data analysis and the methods required to do it in R.

2. Prerequisites:

1. Basics of R programming, various data structures, functions etc.

3. Hardware Requirements:

1. PC with minimum 2GB RAM

4. Software Requirements:

1. Windows / Linux OS.
2. R version 3.6 or higher

5. Learning Objectives:

1. To understand exploratory data analysis.
2. To know library functions used for exploratory data analysis.

6. Learning Objectives Applicable: LO 3. LO 4

7. Program Outcomes Applicable: PO 2, PO 3

8. Program Education Objectives Applicable: PEO 2, PEO 3

9. Theory:

Minimum and maximum

Minimum and maximum can be found thanks to the

```
min()
```

and

```
max()
```

functions:

```
min(dat$Sepal.Length)
```

```
## [1] 4.3
```

```
max(dat$Sepal.Length)
```

```
## [1] 7.9
```

Alternatively the

```
range()
```

function:

```
rng <- range(dat$Sepal.Length)

rng
```

```
## [1] 4.3 7.9
```

gives you the minimum and maximum directly. Note that the output of the

```
range()
```

function is actually an object containing the minimum and maximum (in that order). This means you can actually access the minimum with:

```
rng[1] # rng = name of the object specified above
```

```
## [1] 4.3
```

and the maximum with:

```
rng[2]
```

```
## [1] 7.9
```

This reminds us that, in R, there are often several ways to arrive at the same result. The method that uses the shortest piece of code is usually preferred as a shorter piece of code is less prone to coding errors and more readable.

Range

The range can then be easily computed, as you have guessed, by substracting the minimum from the maximum:

```
max(dat$Sepal.Length) - min(dat$Sepal.Length)

## [1] 3.6
```

To my knowledge, there is no default function to compute the range. However, if you are familiar with writing functions in R , you can create your own function to compute the range:

```
range2 <- function(x) {

range <- max(x) - min(x)

return(range)

}

range2(dat$Sepal.Length)

## [1] 3.6
```

which is equivalent than $\max - \min$max−min presented above.

Mean

The mean can be computed with the

```
mean()
```

function:

```
mean(dat$Sepal.Length)

## [1] 5.843333
```

Tips:

- if there is at least one missing value in your dataset, use

  ```
  mean(dat$Sepal.Length, na.rm = TRUE)
  ```

  to compute the mean with the NA excluded. This argument can be used for most functions presented in this article, not only the mean
- for a truncated mean, use

  ```
  mean(dat$Sepal.Length, trim = 0.10)
  ```

and change the

```
trim
```

argument to your needs

Median

The median can be computed thanks to the

```
median()
```

function:

```
median(dat$Sepal.Length)
## [1] 5.8
```

or with the

```
quantile()
```

function:

```
quantile(dat$Sepal.Length, 0.5)
## 50%
## 5.8
```

since the quantile of order 0.5 ($q_{0.5}$q0.5) corresponds to the median.

First and third quartile

As the median, the first and third quartiles can be computed thanks to the

```
quantile()
```

function and by setting the second argument to 0.25 or 0.75:

```
quantile(dat$Sepal.Length, 0.25) # first quartile
## 25%
## 5.1
quantile(dat$Sepal.Length, 0.75) # third quartile
## 75%
## 6.4
```

You may have seen that the results above are slightly different than the results you would have found if you compute the first and third quartiles by hand. It is normal, there are many methods to compute them (R actually has 7 methods to compute the quantiles!). However, the methods presented here and in the article "descriptive statistics by hand" are the easiest and most "standard" ones. Furthermore, results do not dramatically change between the two methods.

Other quantiles

As you have guessed, any quantile can also be computed with the

```
quantile()
```

function. For instance, the 4th4th decile or the 98th98th percentile:

```
quantile(dat$Sepal.Length, 0.4) # 4th decile

## 40%

## 5.6

quantile(dat$Sepal.Length, 0.98) # 98th percentile

## 98%

## 7.7
```

Interquartile range

The interquartile range (i.e., the difference between the first and third quartile) can be computed with the

```
IQR()
```

function:

```
IQR(dat$Sepal.Length)

## [1] 1.3
```

or alternativaly with the

```
quantile()
```

function again:

```
quantile(dat$Sepal.Length, 0.75) - quantile(dat$Sepal.Length, 0.25)

## 75%

## 1.3
```

As mentioned earlier, when possible it is usually recommended to use the shortest piece of code to arrive at the result. For this reason, the

```
IQR()
```

function is preferred to compute the interquartile range.

Standard deviation and variance

The standard deviation and the variance is computed with the

```
sd()
```

and

```
var()
```

functions:

```
sd(dat$Sepal.Length) # standard deviation

## [1] 0.8280661

var(dat$Sepal.Length) # variance

## [1] 0.6856935
```

Remember from this article that the standard deviation and the variance are different whether we compute it for a sample or a population (see the difference between the two here). In R, the standard deviation and the variance are computed as if the data represent a sample (so the denominator is $n-1$n−1, where nn is the number of observations). To my knowledge, there is no function by default in R that computes the standard deviation or variance for a population.

Tip: to compute the standard deviation (or variance) of multiple variables at the same time, use

```
lapply()
```

with the appropriate statistics as second argument:

```
lapply(dat[, 1:4], sd)

## $Sepal.Length

## [1] 0.8280661

##

## $Sepal.Width

## [1] 0.4358663

##

## $Petal.Length
```

[1] 1.765298

##

$Petal.Width

[1] 0.7622377

The command

```
dat[, 1:4]
```

selects the variables 1 to 4 as the fifth variable is a qualitative variable and the standard deviation cannot be computed on such type of variable. See a recap of the different data types in R if needed.

Summary

You can compute the minimum, 1st1st quartile, median, mean, 3rd3rd quartile and the maximum for all numeric variables of a dataset at once using

```
summary()
```

:

```
summary(dat)

## Sepal.Length Sepal.Width Petal.Length Petal.Width

## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100

## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300

## Median :5.800 Median :3.000 Median :4.350 Median :1.300

## Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199

## 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800

## Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500

## Species

## setosa :50

## versicolor:50

## virginica :50

##

##
```

```
##
```

Tip: if you need these descriptive statistics by group use the

```
by()
```

function:

```
by(dat, dat$Species, summary)
## dat$Species: setosa
## Sepal.Length Sepal.Width Petal.Length Petal.Width
## Min. :4.300 Min. :2.300 Min. :1.000 Min. :0.100
## 1st Qu.:4.800 1st Qu.:3.200 1st Qu.:1.400 1st Qu.:0.200
## Median :5.000 Median :3.400 Median :1.500 Median :0.200
## Mean :5.006 Mean :3.428 Mean :1.462 Mean :0.246
## 3rd Qu.:5.200 3rd Qu.:3.675 3rd Qu.:1.575 3rd Qu.:0.300
## Max. :5.800 Max. :4.400 Max. :1.900 Max. :0.600
## Species
## setosa :50
## versicolor: 0
## virginica : 0
##
##
##
## ------------------------------------------------------------
## dat$Species: versicolor
## Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## Min. :4.900 Min. :2.000 Min. :3.00 Min. :1.000 setosa : 0
## 1st Qu.:5.600 1st Qu.:2.525 1st Qu.:4.00 1st Qu.:1.200 versicolor:50
## Median :5.900 Median :2.800 Median :4.35 Median :1.300 virginica : 0
```

```
## Mean :5.936 Mean :2.770 Mean :4.26 Mean :1.326

## 3rd Qu.:6.300 3rd Qu.:3.000 3rd Qu.:4.60 3rd Qu.:1.500

## Max. :7.000 Max. :3.400 Max. :5.10 Max. :1.800

## ------------------------------------------------------------

## dat$Species: virginica

## Sepal.Length Sepal.Width Petal.Length Petal.Width

## Min. :4.900 Min. :2.200 Min. :4.500 Min. :1.400

## 1st Qu.:6.225 1st Qu.:2.800 1st Qu.:5.100 1st Qu.:1.800

## Median :6.500 Median :3.000 Median :5.550 Median :2.000

## Mean :6.588 Mean :2.974 Mean :5.552 Mean :2.026

## 3rd Qu.:6.900 3rd Qu.:3.175 3rd Qu.:5.875 3rd Qu.:2.300

## Max. :7.900 Max. :3.800 Max. :6.900 Max. :2.500

## Species

## setosa : 0

## versicolor: 0

## virginica :50

##

##

##
```

where the arguments are the name of the dataset, the grouping variable and the summary function. Follow this order, or specify the name of the arguments if you do not follow this order.

If you need more descriptive statistics, use

```
stat.desc()
```

from the package

```
{pastecs}
```

:

```
library(pastecs)
```

```
stat.desc(dat)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## nbr.val 150.00000000 150.00000000 150.0000000 150.00000000 NA
## nbr.null 0.00000000 0.00000000 0.0000000 0.00000000 NA
## nbr.na 0.00000000 0.00000000 0.0000000 0.00000000 NA
## min 4.30000000 2.00000000 1.0000000 0.10000000 NA
## max 7.90000000 4.40000000 6.9000000 2.50000000 NA
## range 3.60000000 2.40000000 5.9000000 2.40000000 NA
## sum 876.50000000 458.60000000 563.7000000 179.90000000 NA
## median 5.80000000 3.00000000 4.3500000 1.30000000 NA
## mean 5.84333333 3.05733333 3.7580000 1.19933333 NA
## SE.mean 0.06761132 0.03558833 0.1441360 0.06223645 NA
## CI.mean.0.95 0.13360085 0.07032302 0.2848146 0.12298004 NA
## var 0.68569351 0.18997942 3.1162779 0.58100626 NA
## std.dev 0.82806613 0.43586628 1.7652982 0.76223767 NA
## coef.var 0.14171126 0.14256420 0.4697441 0.63555114 NA
```

You can have even more statistics (i.e., skewness, kurtosis and normality test) by adding the argument

```
norm = TRUE
```

in the previous function. Note that the variable

```
Species
```

is not numeric, so descriptive statistics cannot be computed for this variable and NA are displayed.

Coefficient of variation

The coefficient of variation can be found with

```
stat.desc()
```

(see the line

```
coef.var
```

in the table above) or by computing manually (remember that the coefficient of variation is the standard deviation divided by the mean):

```
sd(dat$Sepal.Length) / mean(dat$Sepal.Length)

## [1] 0.1417113
```

Mode

To my knowledge there is no function to find the mode of a variable. However, we can easily find it thanks to the functions

```
table()
```

and

```
sort()
```

:

```
tab <- table(dat$Sepal.Length) # number of occurences for each unique value

sort(tab, decreasing = TRUE) # sort highest to lowest

##

## 5 5.1 6.3 5.7 6.7 5.5 5.8 6.4 4.9 5.4 5.6 6 6.1 4.8 6.5 4.6 5.2 6.2 6.9 7.7

## 10 9 9 8 8 7 7 7 6 6 6 6 6 5 5 4 4 4 4 4

## 4.4 5.9 6.8 7.2 4.7 6.6 4.3 4.5 5.3 7 7.1 7.3 7.4 7.6 7.9

## 3 3 3 3 2 2 1 1 1 1 1 1 1 1 1
```

```
table()
```

gives the number of occurences for each unique value, then

```
sort()
```

with the argument

```
decreasing = TRUE
```

displays the number of occurences from highest to lowest. The mode of the variable

```
Sepal.Length
```

is thus 5. This code to find the mode can also be applied to qualitative variables such as

```
Species
```

:

```
sort(table(dat$Species), decreasing = TRUE)

##

## setosa versicolor virginica

## 50 50 50
```

or:

```
summary(dat$Species)

## setosa versicolor virginica

## 50 50 50
```

Contingency table

`table()`

introduced above can also be used on two qualitative variables to create a contingency table. The dataset

`iris`

has only one qualitative variable so we create a new qualitative variable just for this example. We create the variable

`size`

which corresponds to

`small`

if the length of the petal is smaller than the median of all flowers,

`big`

otherwise:

```
dat$size <- ifelse(dat$Sepal.Length < median(dat$Sepal.Length),

"small", "big"

)
```

Here is a recap of the occurences by size:

```
table(dat$size)

##

## big small
```

77 73

We now create a contingency table of the two variables

`Species`

and

`size`

with the

`table()`

function:

```
table(dat$Species, dat$size)
##
## big small
## setosa 1 49
## versicolor 29 21
## virginica 47 3
```

or with the

`xtabs()`

function:

```
xtabs(~ dat$Species + dat$size)
## dat$size
## dat$Species big small
## setosa 1 49
## versicolor 29 21
## virginica 47 3
```

The contingency table gives the number of cases in each subgroup. For instance, there is only one big setosa flower, while there are 49 small setosa flowers in the dataset. Note that

`Species`

are in rows and

```
size
```

in column because we specified

```
Species
```

and then

```
size
```

in

```
table()
```

. Change the order if you want to switch the two variables.

Instead of having the frequencies (i.e.. the number of cases) you can also have the relative frequencies in each subgroup by adding the

```
table()
```

function inside the

```
prop.table()
```

function:

```
prop.table(table(dat$Species, dat$size))

##

## big small

## setosa 0.006666667 0.326666667

## versicolor 0.193333333 0.140000000

## virginica 0.313333333 0.020000000
```

Note that you can also compute the percentages by row or by column by adding a second argument to the

```
prop.table()
```

function:

```
1
```

for row, or

```
2
```

for column:

```
# percentages by row:
```

```
round(prop.table(table(dat$Species, dat$size), 1), 2) # round to 2 digits with round()

##

## big small

## setosa 0.02 0.98

## versicolor 0.58 0.42

## virginica 0.94 0.06

# percentages by column:

round(prop.table(table(dat$Species, dat$size), 2), 2) # round to 2 digits with round()

##

## big small

## setosa 0.01 0.67

## versicolor 0.38 0.29

## virginica 0.61 0.04
```

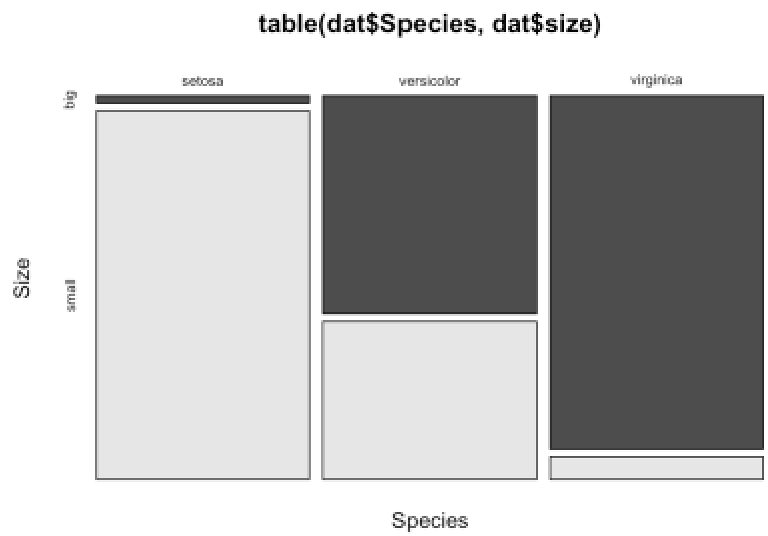
Barplot

Barplots can only be done on qualitative variables (see the difference with a quantative variable here). A barplot is a tool to visualize the distribution of a qualitative variable. We draw a barplot on the qualitative variable

```
size
```

:

```
barplot(table(dat$size)) # table() is mandatory
```

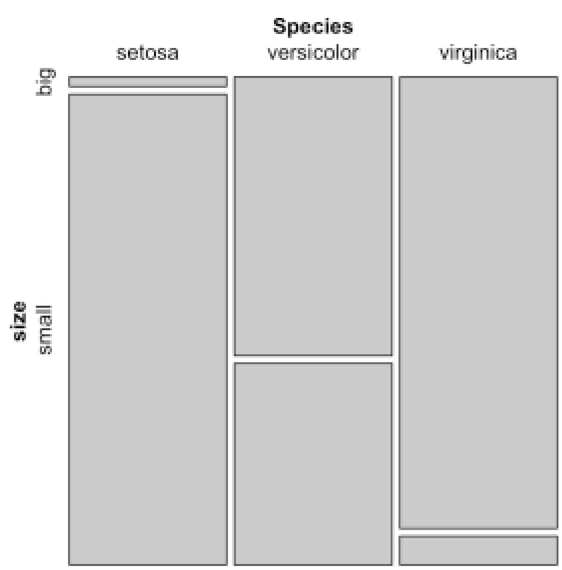


You can also draw a barplot of the relative frequencies instead of the frequencies by adding

```
prop.table()
```

as we did earlier:

barplot(prop.table(table(dat$size)))

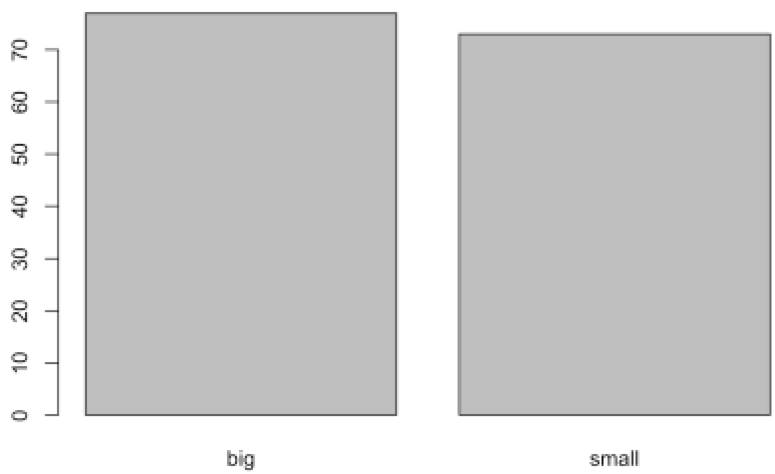


In

```
{ggplot2}
```

:

```
library(ggplot2) # needed each time you open RStudio

# The package ggplot2 must be installed first

ggplot(dat) +

aes(x = size) +

geom_bar()
```



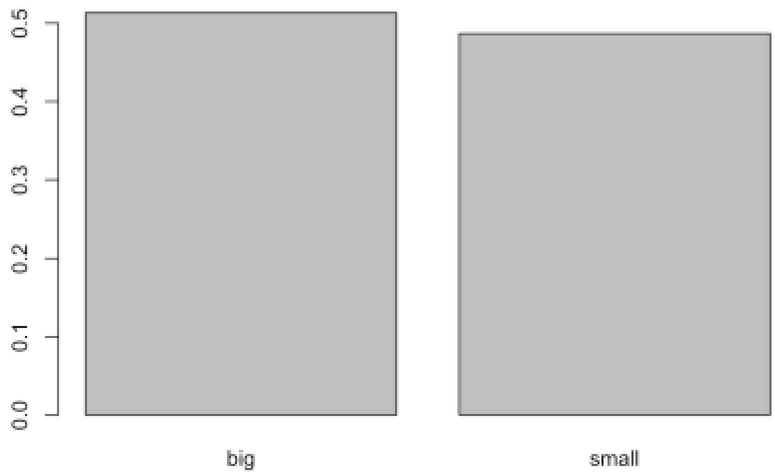
Histogram

A histogram gives an idea about the distribution of a quantitative variable. The idea is to break the range of values into intervals and count how many observations fall into each interval. Histograms are a bit similar to barplots, but histograms are used for quantitative variables whereas barplots are used for qualitative variables. To draw a histogram in R, use

```
hist()
```

:

```
hist(dat$Sepal.Length)
```



Add the arguments

```
breaks =
```

inside the

```
hist()
```

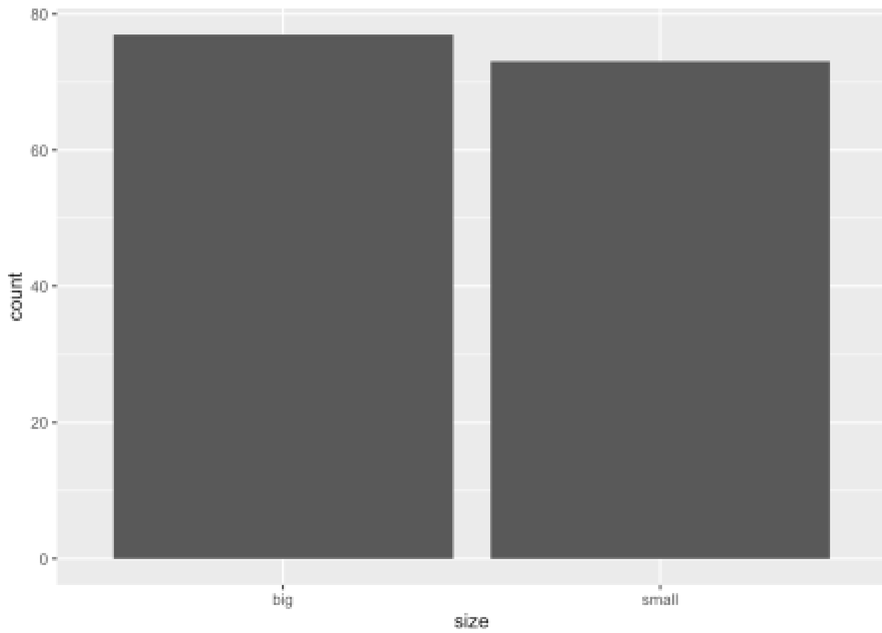
function if you want to change the number of bins. A rule of thumb (known as Sturges' law) is that the number of bins should be the rounded value of the square root of the number of observations. The dataset includes 150 observations so in this case the number of bins can be set to 12.

In

```
{ggplot2}
```

:

```
ggplot(dat) +

aes(x = Sepal.Length) +

geom_histogram()
```



By default, the number of bins is 30. You can change this value with

```
geom_histogram(bins = 12)
```

for instance.

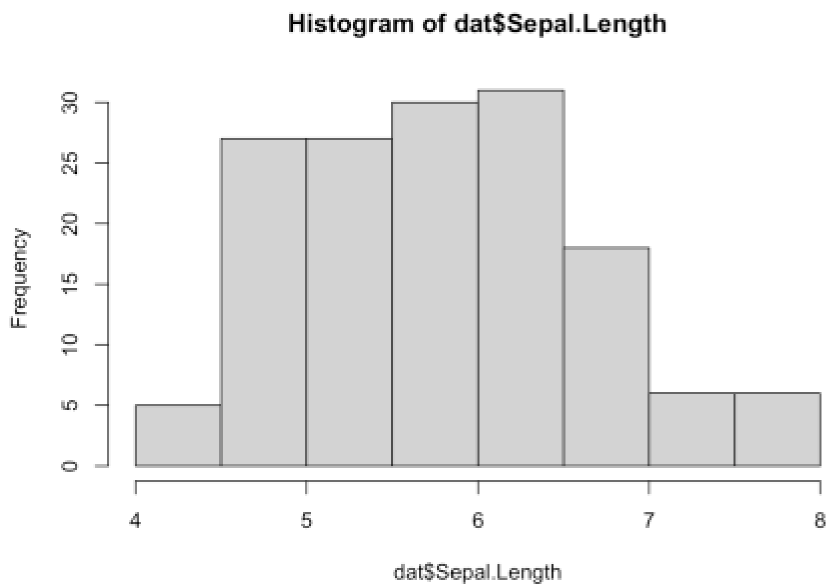
Boxplot

Boxplots are really useful in descriptive statistics and are often underused (mostly because it is not well understood by the public). A boxplot graphically represents the distribution of a quantitative variable by visually displaying five common location summary (minimum, median, first and third quartiles and maximum) and any observation that was classified as a suspected outlier using the interquartile range (IQR) criterion. The IQR criterion means that all observations above $q_{0.75}+1.5 \cdot IQR$ and below $q_{0.25}-1.5 \cdot IQR$ (where $q_{0.25}$ and $q_{0.75}$ correspond to first and third quartile respectively) are considered as potential outliers by R. The minimum and maximum in the boxplot are represented without these suspected outliers. Seeing all these information on the same plot help to have a good first overview of the dispersion and the location of the data. Before drawing a boxplot of our data, see below a graph explaining the information present on a boxplot:

Detailed boxplot. Source: LFSAB1105 at UCLouvain

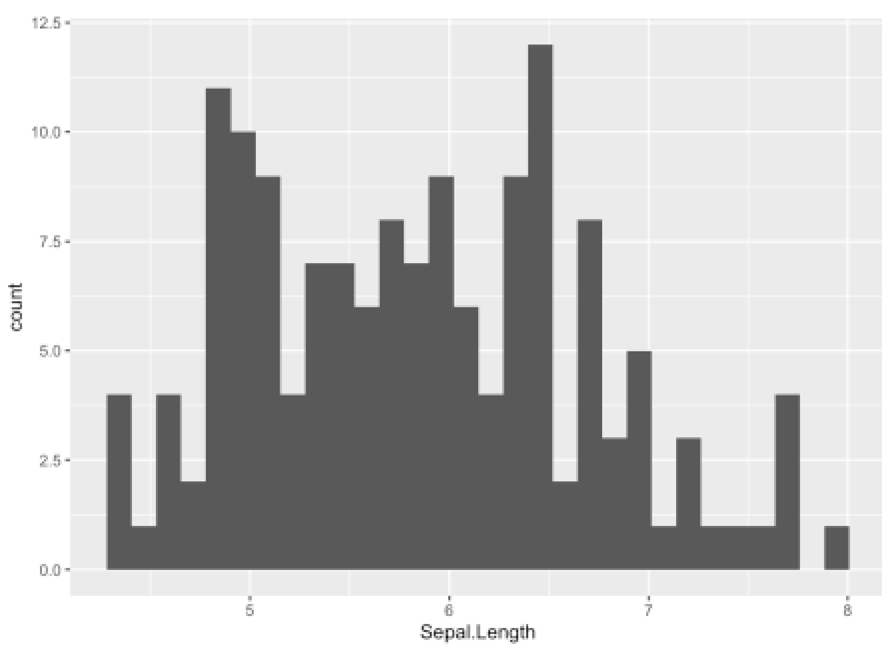
Now an example with our dataset:

```
boxplot(dat$Sepal.Length)
```



Boxplots are even more informative when presented side-by-side for comparing and contrasting distributions from two or more groups. For instance, we compare the length of the sepal across the different species:

```
boxplot(dat$Sepal.Length ~ dat$Species)
```



In

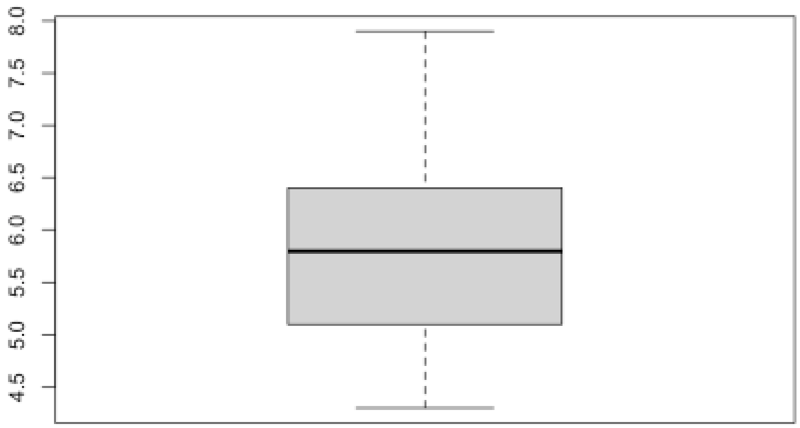
```
{ggplot2}
```

:

```
ggplot(dat) +
```

aes(x = Species, y = Sepal.Length) +

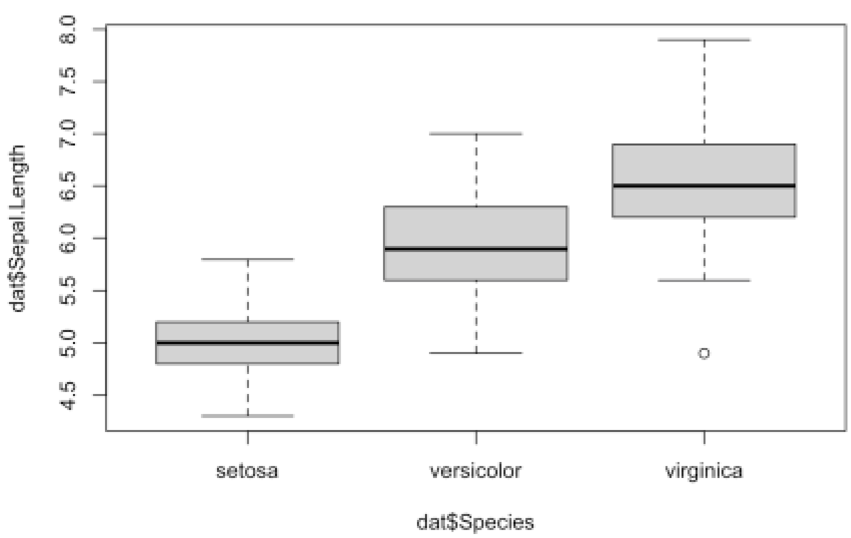
geom_boxplot()



Scatterplot

Scatterplots allow to check whether there is a potential link between two quantitative variables. For instance, when drawing a scatterplot of the length of the sepal and the length of the petal:

```
plot(dat$Sepal.Length, dat$Petal.Length)
```



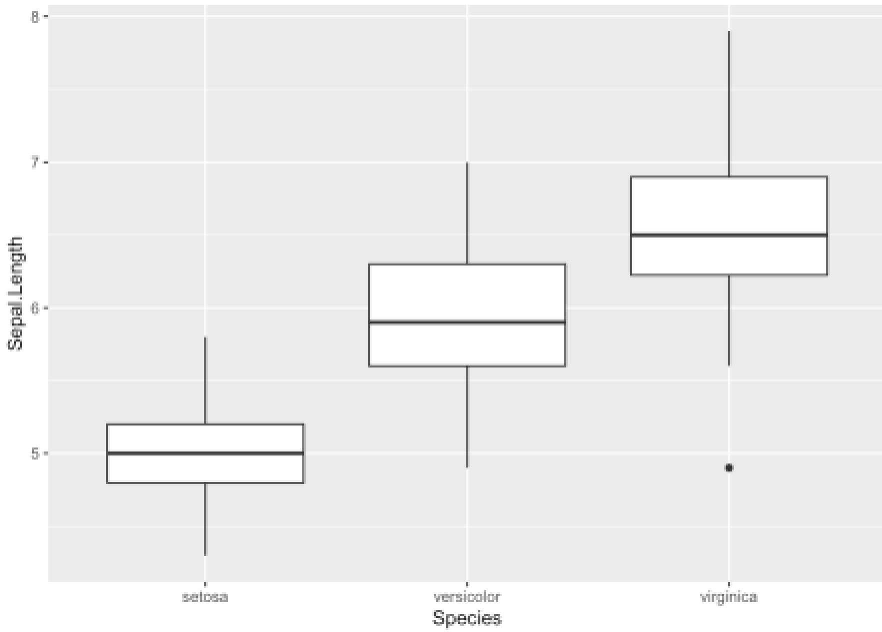
There seems to be a positive association between the two variables.

In

```
{ggplot2}
```

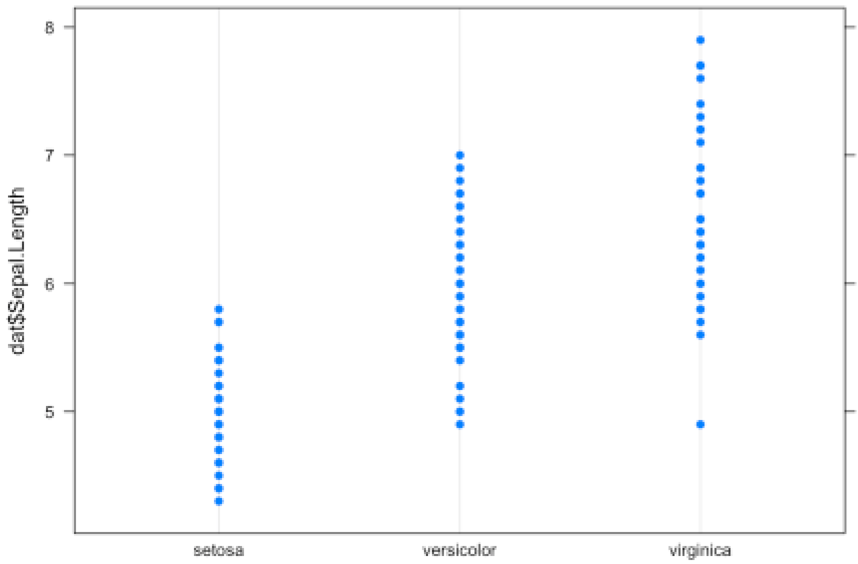
:

```
ggplot(dat) +

aes(x = Sepal.Length, y = Petal.Length) +

geom_point()
```



As boxplots, scatterplots are even more informative when differentiating the points according to a factor, in this case the species:

```
ggplot(dat) +

aes(x = Sepal.Length, y = Petal.Length, colour = Species) +

geom_point() +

scale_color_hue()
```

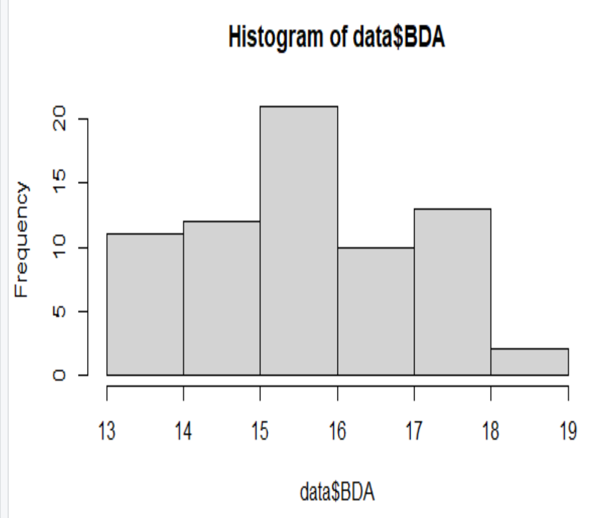


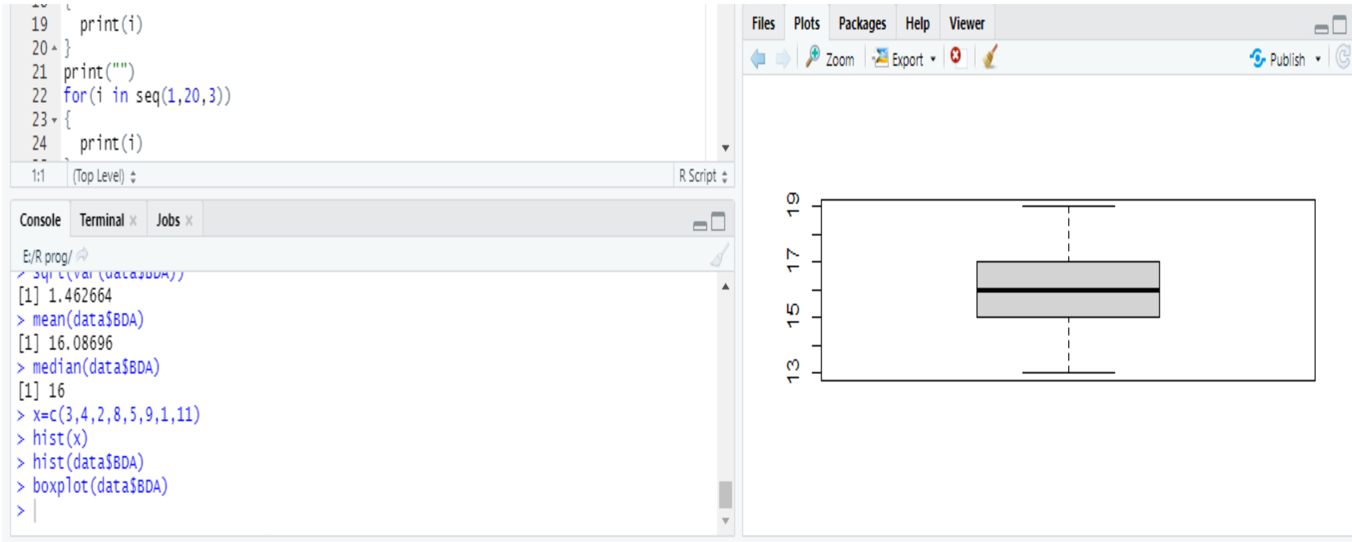
10. Results:

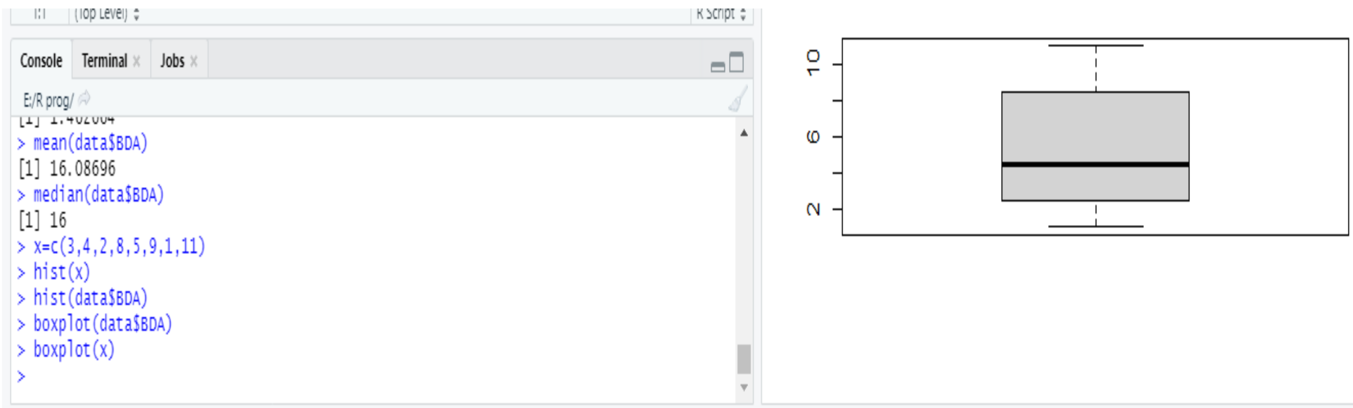
```
>
> getwd()
[1] "C:/Users/Dell/Documents/Downloads"
> setwd("E:/R prog")
> data=read.csv("testdata.csv")
> data
   BDA IOE ERP
1   16  17  20
2   15  16  18
3   14  14  16
4   16  18  18
5   18  17  20
6   14  14  16
7   15  17  18
8   17  17  18
9   15  16  18
10  16  18  20
11  16  17  16
12  16  17  18
13  15  16  16
14  15  17  14
15  17  16  18
16  16  16  18
17  18  16  17
18  15  14  18
19  15  16  20
20  16  18  18
21  15  16  14
22  16  18  20
23  17  17  20
24  18  18  20
25  13  18  15
26  17  17  18
27  18  16  20
28  18  16  19
29  19  15  19
30  18  17  20
31  14  14  16
```

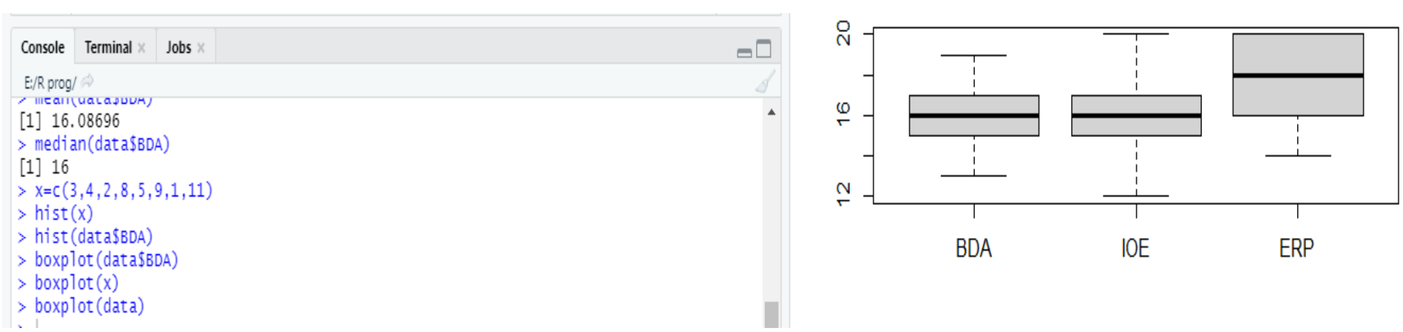
```
69  18  18  20
> class(data)
[1] "data.frame"
> data$BDA
 [1] 16 15 14 16 18 14 15 17 15 16 16 16 15 15 17 16 18 15 15 16 15 16 17 18 13 17
[27] 18 18 19 18 14 16 16 16 18 15 18 18 17 16 17 16 16 19 15 14 17 14 17 16 15 16
[53] 14 18 14 14 16 17 16 16 16 18 14 13 18 15 17 16 18
> data$IOE
 [1] 17 16 14 18 17 14 17 17 16 18 17 17 16 17 16 16 16 14 16 18 16 18 17 18 18 17
[27] 16 16 15 17 14 16 17 14 20 19 17 15 18 15 16 17 14 17 14 14 16 14 15 14 15 15
[53] 13 15 15 14 14 16 14 16 16 16 12 16 15 15 16 16 18
> data$ERP
 [1] 20 18 16 18 20 16 18 18 18 20 16 18 16 14 18 18 17 18 20 18 14 20 20 20 15 18
[27] 20 19 19 20 16 19 17 19 20 20 19 17 20 20 19 19 16 20 16 16 19 15 18 19 16 15
[53] 16 20 16 16 20 19 20 19 19 19 18 18 16 15 19 18 20
> range(data$BDA)
[1] 13 19
> range(data)
[1] 12 20
> range(data$IOE)
[1] 12 20
> range(data$ERP)
[1] 14 20
> summary(data$BDA)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  13.00   15.00   16.00   16.09   17.00   19.00
> summary(data$ERP)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  14.00   16.00   18.00   18.01   20.00   20.00
> summary(data$IOE)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.00   15.00   16.00   15.91   17.00   20.00
```

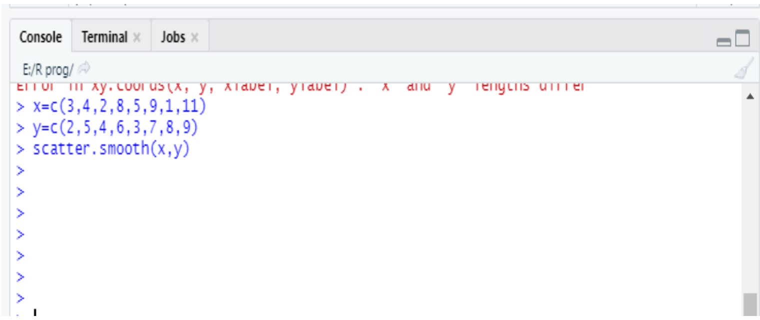
```
> summary(data)
      BDA             IOE             ERP
 Min.   :13.00   Min.   :12.00   Min.   :14.00
 1st Qu.:15.00   1st Qu.:15.00   1st Qu.:16.00
 Median :16.00   Median :16.00   Median :18.00
 Mean   :16.09   Mean   :15.91   Mean   :18.01
 3rd Qu.:17.00   3rd Qu.:17.00   3rd Qu.:20.00
 Max.   :19.00   Max.   :20.00   Max.   :20.00
> var(data)
          BDA       IOE      ERP
BDA 2.1393862 0.8164962 1.454604
IOE 0.8164962 2.3158568 1.015985
ERP 1.4546036 1.0159847 3.073316
> var(data$BDA)
[1] 2.139386
> sqrt(var(data$BDA))
[1] 1.462664
> mean(data$BDA)
[1] 16.08696
> median(data$BDA)
[1] 16
> x=c(3,4,2,8,5,9,1,11)
> hist(x)
> hist(data$BDA)
```

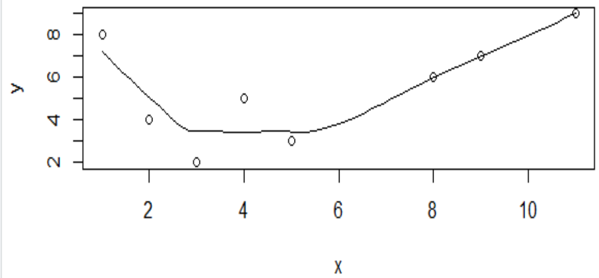


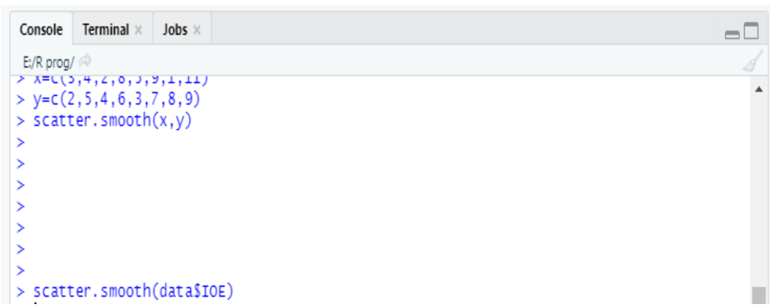

19 print(i)
20 }
21 print("")
22 for(i in seq(1,20,3))
23 {
24 print(i)
1:1 (Top Level) R Script
Console Terminal Jobs
E:/R prog/
[1] 1.462664
> mean(data$BDA)
[1] 16.08696
> median(data$BDA)
[1] 16
> x=c(3,4,2,8,5,9,1,11)
> hist(x)
> hist(data$BDA)
> boxplot(data$BDA)
>
Files Plots Packages Help Viewer
Zoom Export Publish
13 15 17 19

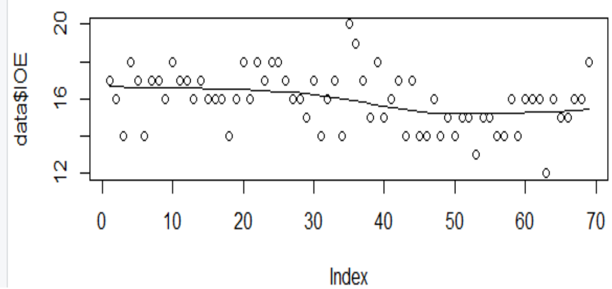

Console Terminal Jobs
E:/R prog/
> mean(data$BDA)
[1] 16.08696
> median(data$BDA)
[1] 16
> x=c(3,4,2,8,5,9,1,11)
> hist(x)
> hist(data$BDA)
> boxplot(data$BDA)
> boxplot(x)
>
2 6 10


Console Terminal Jobs
E:/R prog/
[1] 16.08696
> median(data$BDA)
[1] 16
> x=c(3,4,2,8,5,9,1,11)
> hist(x)
> hist(data$BDA)
> boxplot(data$BDA)
> boxplot(x)
> boxplot(data)
12 16 20
BDA IOE ERP










```
> quantile(data$BDA)
  0%  25%  50%  75% 100%
  13   15   16   17   19
>
```

11. Learning Outcomes Achieved:

1. Understood exploratory data analysis.
2. Know library functions used for exploratory data analysis.

12. Conclusion:

Data analysed such as- Range, summary, mean, variance, median, standard deviation, histogram, boxplot, scatterplot

13. Experiment/Assignment Evaluation

Experiment/Assignment Evaluation:					
Sr. No.	**Parameters**			**Marks obtained**	**Out of**
1	Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used -				6
2	Neatness/presentation				2
3	Punctuality				2
Date of performance (DOP)			**Total marks obtained**		**10**
Date of checking (DOC)			**Signature of teacher**		

References:

1. URL: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf (Online Resources)
2. R Cookbook Paperback – 2011 by Teetor Paul O Reilly Publications
3. Beginning R: The Statistical Programming Language by Dr. Mark Gardener, Wiley Publications
4. R Programming For Dummies by Joris Meys Andrie de Vries, Wiley Publications

Viva Questions

1. What is exploratory data analysis ?
2. What is the summary of the data ?
3. What is the importance of the median of the data collection ?
4. What is histogram? Why is it important in data?
5. What information does the box plot provide?
6. List various R library functions used in exploratory data analysis.

456

Finolex Academy of Management and Technology, Ratnagiri

Department of Information Technology

Subject:	**R Programming Lab. (ITL804)**		
Class:	**BE IT / Semester – VIII (Rev-2016) / Academic year: 2020-21**		
Name of Student:	**Pranali Hanumant Kudtarkar**		
Roll No:	**26**	**Date of performance (DOP) :**	
Assignment/Experiment No:	**05**	**Date of checking (DOC) :**	
Title: Working with graphics and tables			
Marks:		**Teacher's Signature:**	

1. Aim: To understand the exploratory data analysis and the methods required to do it in R.

2. Prerequisites:

1. Basics of R programming, various data structures for data sets.

3. Hardware Requirements:

1. PC with minimum 2GB RAM

4. Software Requirements:

1. Windows / Linux OS.
2. R version 3.6 or higher

5. Learning Objectives:

1. To understand various graphical visualization of data sets.
2. To understand the use of tables.

6. Learning Objectives Applicable: LO 5

7. Program Outcomes Applicable: PO 4, PO 5

8. Program Education Objectives Applicable: PEO 3, PEO 4

9. Theory:

In this section we present what you need to know if you want to customize your graphs in the default graph system.

- `plot()` is the main function for graphics. The arguments can be a single point such as 0 or c(.3,.7), a single vector, a pair of vectors or many other R objects.
- `par()` is another important function which defines the default settings for plots.
- There are many other plot functions which are specific to some tasks such as `hist()`, `boxplot()`, etc. Most of them take the same arguments as the `plot()` function.

```
> N <- 10^2
> x1 <- rnorm(N)
> x2 <- 1 + x1 + rnorm(N)
> plot(0)
> plot(0,1)
> plot(x1)
> plot(x1,x2) # scatter plot x1 on the horizontal axis and x2 on the vertical axis
> plot(x2 ~ x1) # the same but using a formula (x2 as a function of x1)
> methods(plot) # show all the available methods for plot (depending on the number
of loaded packages).
```

Titles, legends and annotations

Titles

`main` gives the main title, `sub` the subtitle. They can be passed as argument of the `plot()` function or using the `title()` function. `xlab` the name of the x axis and `ylab` the name of the y axis.

```
 plot(x1,x2, main = "Main title", sub = "sub title" , ylab = "Y axis", xlab = "X
axis")
 plot(x1,x2 ,  ylab = "Y axis", xlab = "X axis")
 title(main = "Main title", sub = "sub title" )
```

The size of the text can be modified using the parameters `cex.main`, `cex.lab`, `cex.sub`, `cex.axis`. Those parameters define a *scaling factor*, ie the value of the parameter multiply the size of the text. If you choose `cex.main=2` the main title will be twice as big as usual.

Legend

`legend()`. The position can be "bottomleft", "bottomright", "topleft", "topright" or exact coordinates.

```
plot(x1, type = "l", col = 1, lty = 1)
lines(x2, col = 2, lty = 2)
legend("bottomleft", legend = c("x1","x2"), col = 1:2, lty = 1:2)
```

Text in the margin

`mtext()` puts some texts in the margin. The margin can be at the bottom (1), the left (2), the top (3) or the right (4).

```
plot(x1, type = "l", col = 1, lty = 1) ; mtext("some text", side = 1) # the bottom
plot(x1, type = "l", col = 1, lty = 1) ; mtext("some text", side = 2) # the left
plot(x1, type = "l", col = 1, lty = 1) ; mtext("some text", side = 3) # the top
```

```
plot(x1, type = "l", col = 1, lty = 1) ; mtext("some text", side = 4) # the right
margin
```

Text in the graph]

`text()`

Mathematical annotations

We can add mathematical symbols using `expression()` and makes some substitution in a formula using `substitute()`.

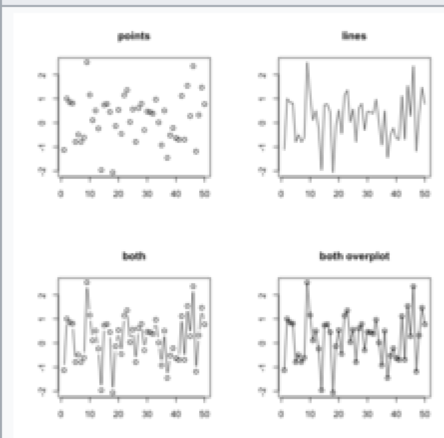
```
?plotmath # gives help for mathematical annotations
```

Types

The type of a plot can be :

- `n` for none (nothing is printed),
- `p` for points,
- `l` for lines,
- `b` for both,
- `o` for both overlayed,
- `h` for histogram-like
- and `s/S` for steps.

R code	Output

`x1 <- rnorm(50)`
`png("plottype.png")`
`par(mfrow = c(2,2))`
`plot(x1, type = "p", main = "points", ylab = "", xlab = "")`
`plot(x1, type = "l", main = "lines", ylab = "", xlab = "")`
`plot(x1, type = "b", main = "both", ylab = "", xlab = "")`
`plot(x1, type = "o", main = "both overplot", ylab = "", xlab = "")`
`dev.off()`
	 click on the graph to zoom

Axes

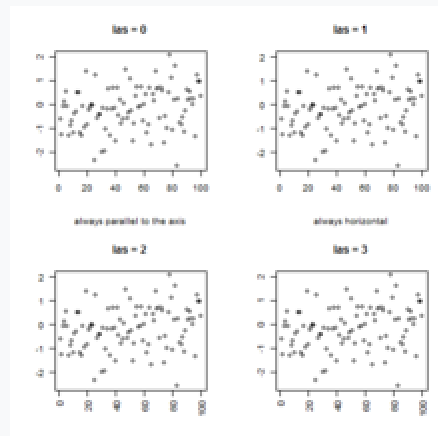
The default output print the axes. We can remove them with `axes=FALSE`. We can also change them using the `axis()` function.

```
> plot(x1,x2,axes=FALSE)
>
> plot(x1,x2,axes=FALSE)
> axis(1,col="red",col.axis="blue",font.axis=3)
> axis(2,col="red",col.axis="blue",font.axis=2,las=2)
```

`las` specifies the style of axis labels. It can be 0, 1, 2 or 3.

- 0 : always parallel to the axis [default],
- 1 : always horizontal,
- 2 : always perpendicular to the axis,
- 3 : always vertical.

R code	Output

`x1 <- rnorm(100)`
`par(mfrow = c(2,2))`
`plot(x1, las = 0, main = "las = 0", sub = "always parallel to the axis", xlab = "", ylab = "")`
`plot(x1, las = 1, main = "las = 1", sub = "always horizontal", xlab = "", ylab = "")`
`plot(x1, las = 2, main = "las = 2", sub = "always perpendicular to the axis", xlab = "", ylab = "")`
`plot(x1, las = 3, main = "las = 3", sub = "always vertical", xlab = "", ylab = "")`
	
	click on the graph

It is also possible to add another y axis on the right by adding `axis(4,)`.

Margins

Margins can be computed in inches or in lines. The default is `par(mar = c(5,4,4,2))` which means that there are 5 lines at the bottom, 4 lines on the left, 4 lines in the top and 2 lines on the right. This can be modified using the `par()` function. If you want to specify margins in inches, use `par(mai = c(bottom, left, top, right)`. If you want to modify margins in lines, use `par(mar = c(bottom, left, top, right)`. See `?par` to learn more about the topic.

Colors

The color of the points or lines can be changed using the `col` argument, `fg` for foreground colors (boxes and axes) and `bg` for background colors.

- `show.col(object=NULL)` (**Hmisc**) package plots the main R colors with their numeric code.
- The list of all colors in R (pdf)

```
colors() # list the r colors
```

```r
show.col(object=NULL) # graphs the main R colors
plot(x1, col = "blue")
plot(x1, col = "red")
plot(x1, col = "red", col.axis = "dodgerblue", col.lab = "firebrick", col.main =
"darkgreen", col.sub = "cyan4", main = "Testing colors", sub = "sub titles", ylab
= "y axis", xlab = "x axis")
```

- We can also generate new colors using the `rgb()` function. The first argument is the intensity of red, the second, the intensity of green and the third, the intensity of blue. They vary between 0 and 1 by default but this can be modified with the option `max = 255`. `col2rgb()` returns the RGB code of R colors. `col2hex()` (**gplots**) gives the hexadecimal code. `col2grey()` and `col2gray()` (**TeachingDemos**) converts colors to grey scale.

```r
> mycolor <- rgb(.2,.4,.6)
> plot(x1, col = mycolor)
> col2rgb("pink")
      [,1]
red    255
green  192
blue   203
> library("gplots")
> col2hex("pink")
[1] "#FFC0CB"
```

Points

For points the symbols can be changed using the `pch` option which takes integer values between 0 and 25 or a single character. `pch` can also takes a vector as argument. In that case the first points will use the first element of the vector as symbol, and so on.

```r
plot(x1, type = "p", pch = 0)
plot(x1, type = "p", pch = 10)
plot(x1, type = "p", pch = 25)
plot(x1, type = "p", pch = "a")
plot(x1, type = "p", pch = "*")
plot(x1[1:26], type = "p", pch = 0:25)
plot(x1[1:26], type = "p", pch = letters)
```

The following code displays all the symbols on the same plot :

```r
x <- rep(1,25)
plot(x, pch = 1:25, axes = F, xlab = "", ylab = "")
text(1:25,.95,labels = 1:25)
```

`points()` adds points to an existing plot.

```r
> plot(x1, pch = 0) # plot x1
> points(x2, pch = 1, col = "red") # add x2 to the existing plot
```

Lines

We can change the line type with `lty`. The argument is a string ("blank", "solid", "dashed", "dotted", "dotdash", "longdash", or "twodash") or an integer (0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash). The line width can be changed with `lwd`. The default is `lwd=1`. `lwd=2` means that the width is twice the normal width.

```
plot(x1, type = "l", lty = "blank")
plot(x1, type = "l", lty = "solid")
plot(x1, type = "l", lty = "dashed")
plot(x1, type = "l", lty = "dotted")
plot(x1, type = "l", lty = "dotdash")
plot(x1, type = "l", lty = "longdash")
plot(x1, type = "l", lty = "twodash")
```

`lines()` adds an additional lines on a graph.

```
plot(x1, type = "l", lty = "solid")
lines(x2, type = "l", lty = "dashed", col = "red")
```

`abline()` adds an horizontal line (`h=`), a vertical line (`v=`) or a linear function to the current plot (`a=` for the constant and `b=` for the slope). `abline()` can also plot the regression line.

```
> plot(x1, type = "l", lty = "solid")
> abline(h= -3, lty = "dashed", col = "gray")
> abline(v = 0, lty = "dashed", col = "gray")
> abline(a = -3 , b = .06, lty = "dotted", col = "red")
```

Boxes

Each graph is framed by a box. `bty` specifies the box type.

```
plot(x1, bty = "o") # the default
plot(x1, bty = "n") # no box
plot(x1, bty = "l")
plot(x1, bty = "7")
plot(x1, bty = "u")
plot(x1, bty = "c")
plot(x1, bty = "]")
```

See also `box()` to add a box to an existing plot.

Grid

`grid()` adds a grid to the current graph.

```
> plot(x1)
> grid()
```

Although grid has an optional argument nx for setting the number of grid lines, it is not possible to tell it explicitly where to place those lines (it will usually not place them at integer values). A more precise and manageable alternative is to use abline().

```
> abline(v=(seq(0,100,5)), col="lightgray", lty="dotted")
> abline(h=(seq(0,100,5)), col="lightgray", lty="dotted")
```

We can also add a circle to a plot with the `circle()` function in the **calibrate** package.

Background

You can choose the background of your plot. For instance, you can change the background color with `par(bg=)`.

```
par(bg="whitesmoke")
par(bg="transparent")
```

Overlaying plots

`matplot()` can plot several plots at the same time.

```
N <- 100
x1 <- rnorm(N)
x2 <- rnorm(N) + x1 + 1
y <- 1 + x1 + x2 + rnorm(N)
mydat <- data.frame(y,x1,x2)
matplot(mydat[,1],mydat[,2:3], pch = 1:2)
```

Multiple plot

With `par()` we can display multiple figures on the same plot. `mfrow = c(3,2)` prints 6 figures on the same plot with 3 rows and 2 columns. `mfcol = c(3,2)` does the same but the order is not the same.

```
par(mfrow = c(3,2))
plot(x1, type = "n")
plot(x1, type = "p")
plot(x1, type = "l")
plot(x1, type = "h")
plot(x1, type = "s")
plot(x1, type = "S")

par(mfcol = c(3,2))
plot(x1, type = "n")
plot(x1, type = "p")
plot(x1, type = "l")
plot(x1, type = "h")
plot(x1, type = "s")
plot(x1, type = "S")
```

Working with Tables in R

Intro

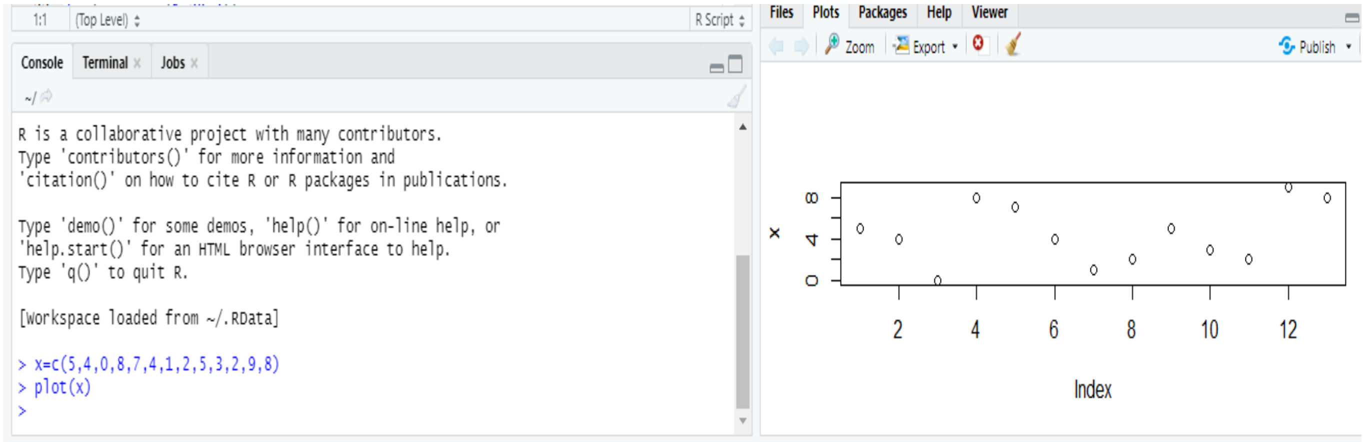
Tables are often essential for organzing and summarizing your data, especially with categorical variables. When creating a table in R, it considers your table as a specifc type of object (called "table") which is very similar to a data frame. Though this may seem strange since datasets are stored as data frames, this means working with tables will be very easy since we have covered data frames in detail over the previous tutorials. In this chapter, we will discuss how to create various types of tables, and how to use various statistical methods to analyze tabular data. Throughout the chapter, the AOSI dataset will be used.

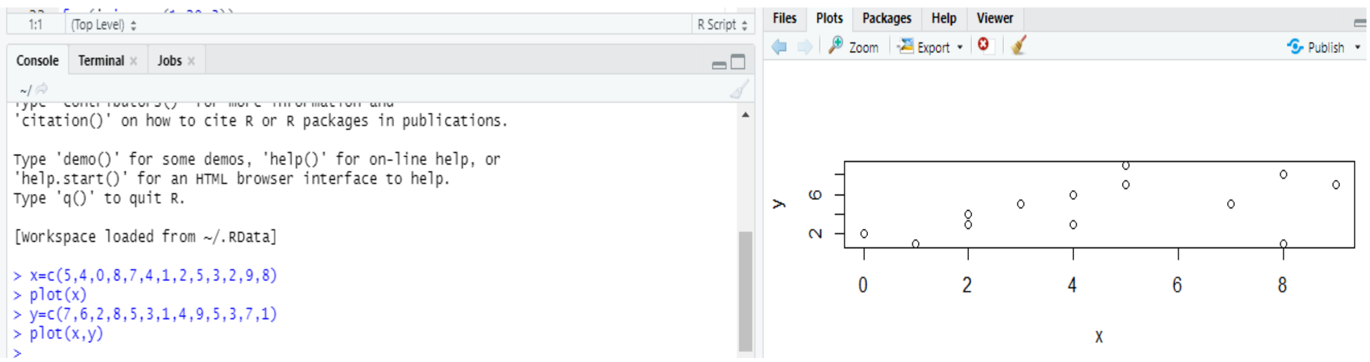
Creating Basic Tables: table() and xtabs()

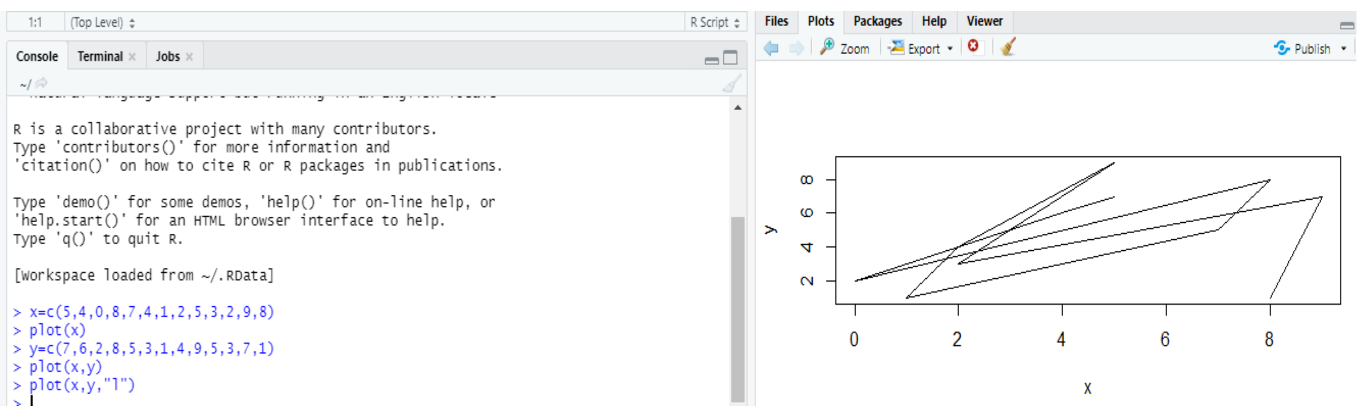
A contingency table is a tabulation of counts and/or percentages for one or more variables. In R, these tables can be created using **table()** along with some of its variations. To use table(), simply add in the variables you want to tabulate separated by a comma. Note that table() does not have a data= argument like many other functions do (e.g., ggplot2 functions), so you much reference the variable using dataset$variable. Some examples are shown below. By default, missing values are excluded from the counts; if you want a count for these missing values you must specify the argument useNA="ifany" or useNA="always". The below examples show how to use this function.

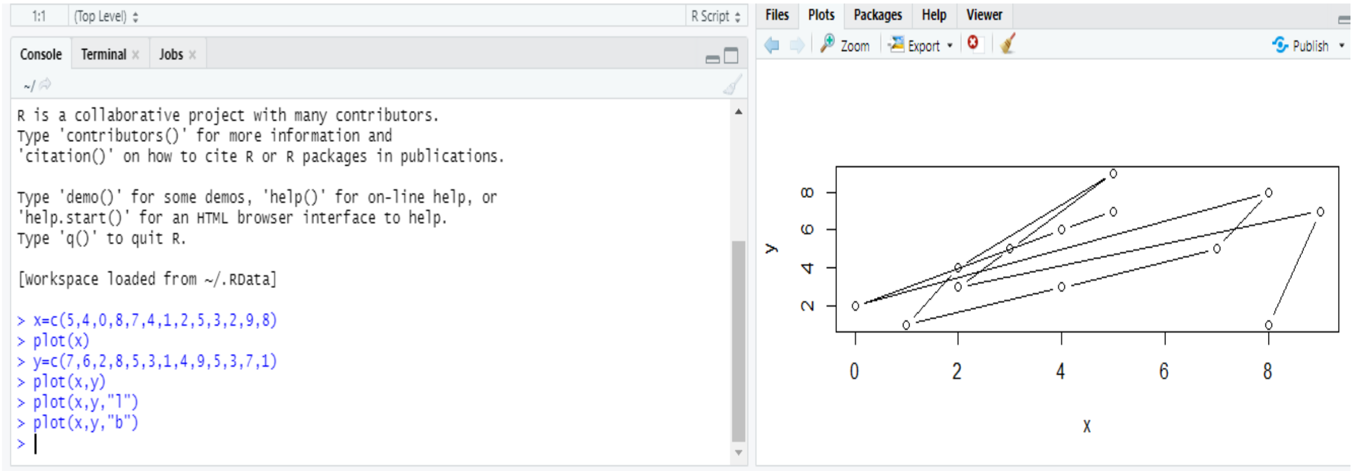
```
aosi_data <- read.csv("Data/cross-sec_aosi.csv", stringsAsFactors=FALSE, na.strings = ".")

# Table for gender
table(aosi_data$Gender)
##
## Female   Male
##    235    352
# Table for study site
table(aosi_data$Study_Site)
##
## PHI SEA STL UNC
## 149 152 145 141
```

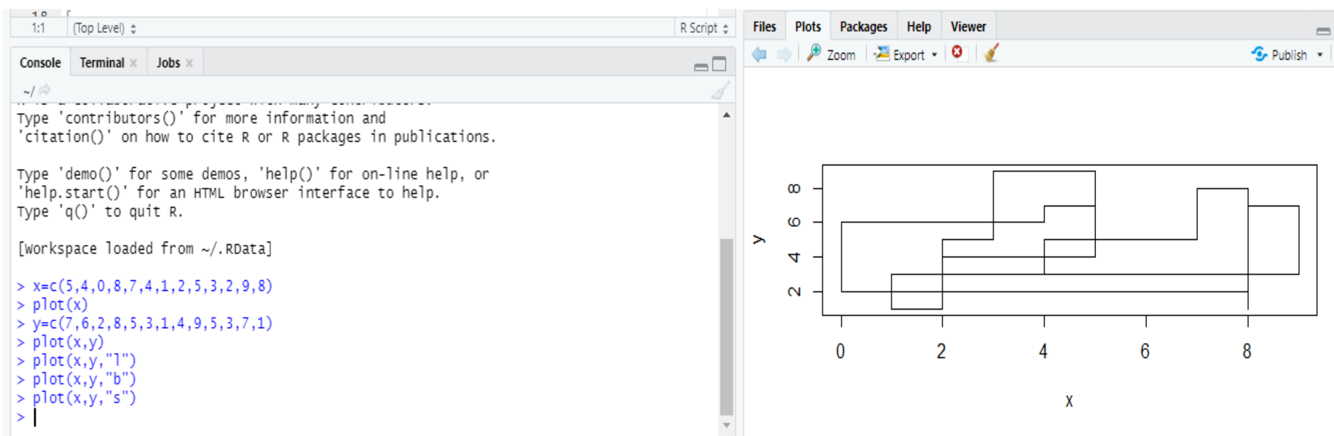
10. Results:

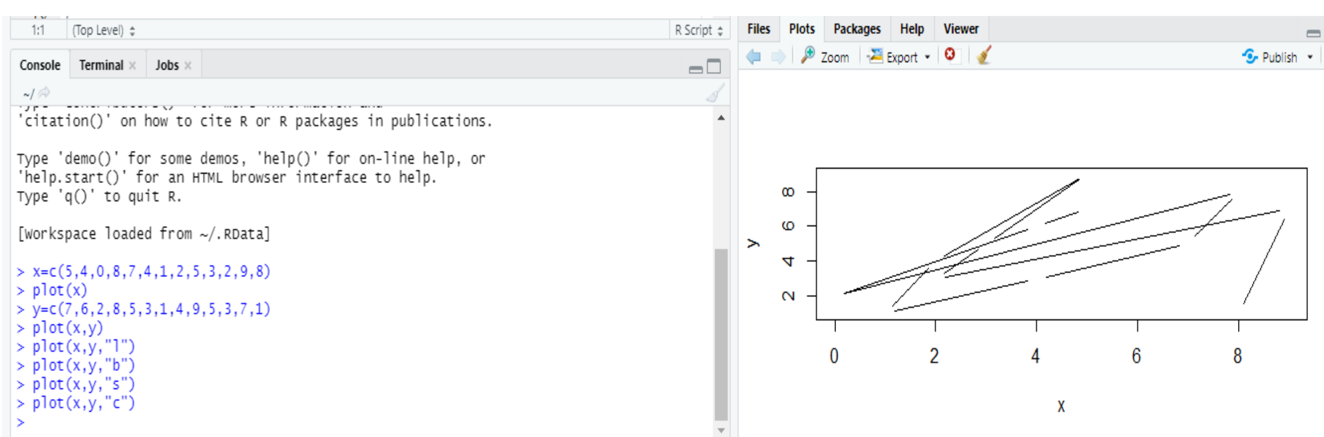

1:1
(Top Level)
R Script
Console
Terminal
Jobs
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
x
Index

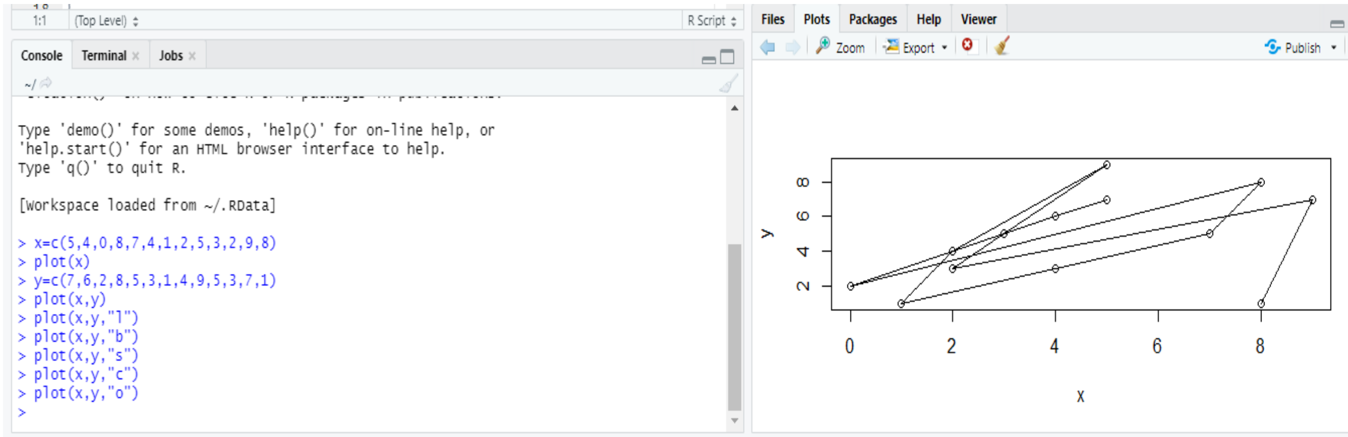

1:1
(Top Level)
R Script
Console
Terminal
Jobs
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
> y=c(7,6,2,8,5,3,1,4,9,5,3,7,1)
> plot(x,y)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
y
x

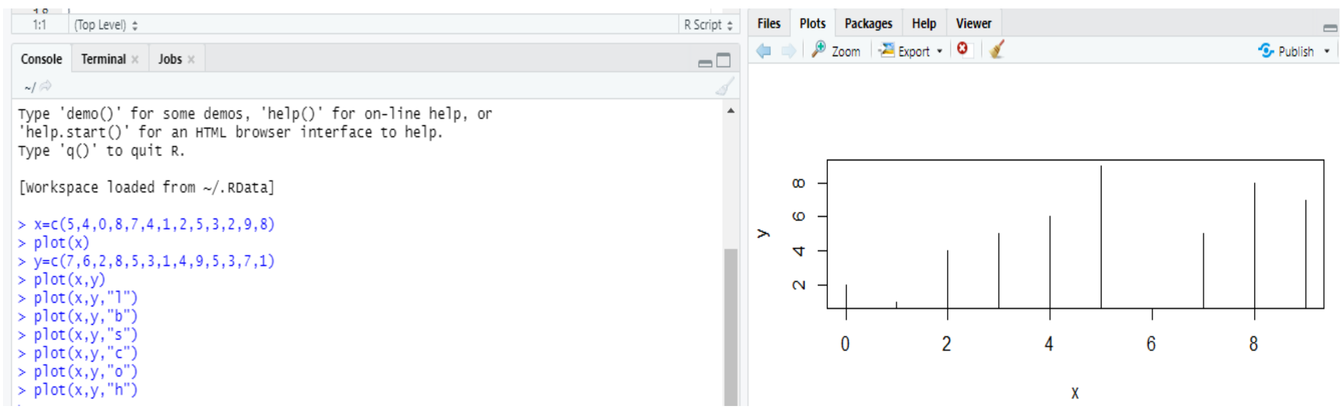

1:1
(Top Level)
R Script
Console
Terminal
Jobs
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
> y=c(7,6,2,8,5,3,1,4,9,5,3,7,1)
> plot(x,y)
> plot(x,y,"l")
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
y
x

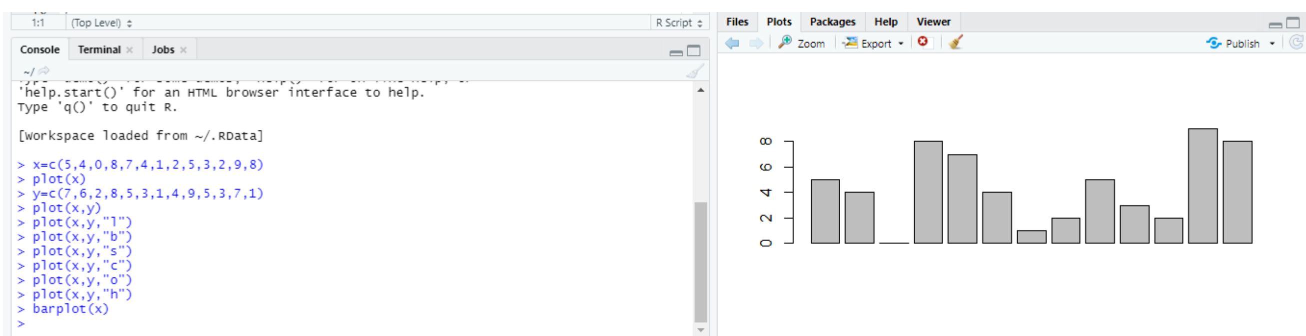

1:1
(Top Level)
R Script
Console
Terminal
Jobs
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
> y=c(7,6,2,8,5,3,1,4,9,5,3,7,1)
> plot(x,y)
> plot(x,y,"l")
> plot(x,y,"b")
>
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
y
2
4
6
8
0
2
4
6
8
x

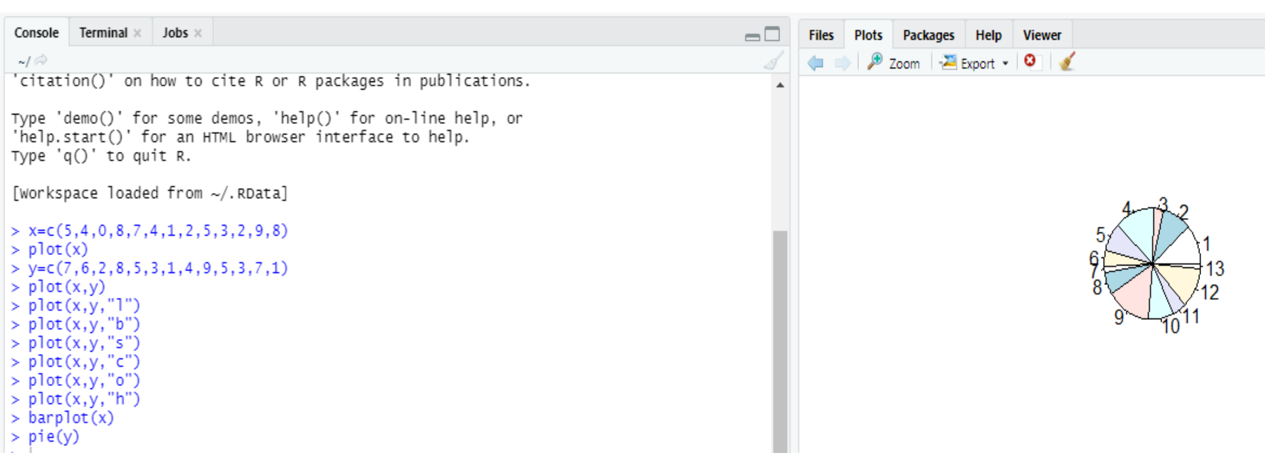

1:1
(Top Level)
R Script
Console
Terminal
Jobs
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
> y=c(7,6,2,8,5,3,1,4,9,5,3,7,1)
> plot(x,y)
> plot(x,y,"l")
> plot(x,y,"b")
> plot(x,y,"s")
>
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
y
2
4
6
8
0
2
4
6
8
x

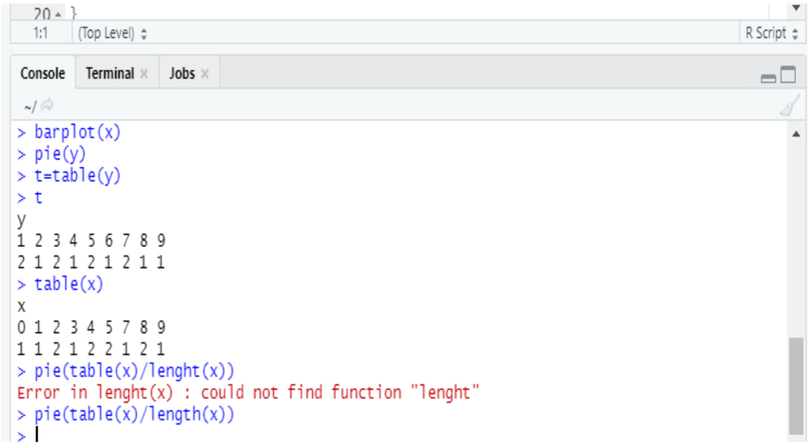

1:1
(Top Level)
R Script
Console
Terminal
Jobs
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
> y=c(7,6,2,8,5,3,1,4,9,5,3,7,1)
> plot(x,y)
> plot(x,y,"l")
> plot(x,y,"b")
> plot(x,y,"s")
> plot(x,y,"c")
>
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
y
2
4
6
8
0
2
4
6
8
x


Console
Terminal
Jobs
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
> y=c(7,6,2,8,5,3,1,4,9,5,3,7,1)
> plot(x,y)
> plot(x,y,"l")
> plot(x,y,"b")
> plot(x,y,"s")
> plot(x,y,"c")
> plot(x,y,"o")
>
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
y
x

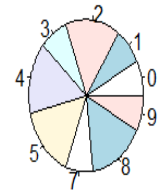

Console
Terminal
Jobs
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
> y=c(7,6,2,8,5,3,1,4,9,5,3,7,1)
> plot(x,y)
> plot(x,y,"l")
> plot(x,y,"b")
> plot(x,y,"s")
> plot(x,y,"c")
> plot(x,y,"o")
> plot(x,y,"h")
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
y
x


Console
Terminal
Jobs
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
> y=c(7,6,2,8,5,3,1,4,9,5,3,7,1)
> plot(x,y)
> plot(x,y,"l")
> plot(x,y,"b")
> plot(x,y,"s")
> plot(x,y,"c")
> plot(x,y,"o")
> plot(x,y,"h")
> barplot(x)
>
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish

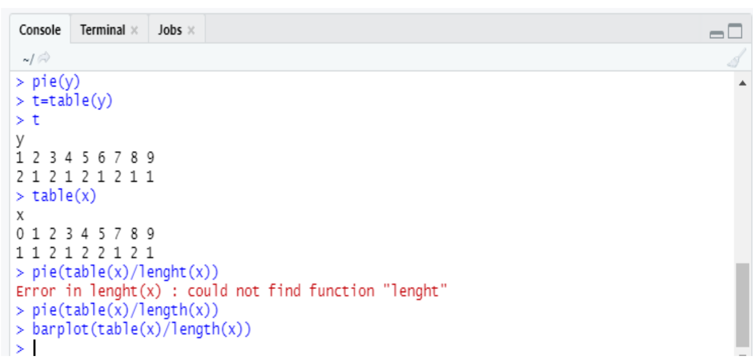

Console
Terminal
Jobs
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> x=c(5,4,0,8,7,4,1,2,5,3,2,9,8)
> plot(x)
> y=c(7,6,2,8,5,3,1,4,9,5,3,7,1)
> plot(x,y)
> plot(x,y,"l")
> plot(x,y,"b")
> plot(x,y,"s")
> plot(x,y,"c")
> plot(x,y,"o")
> plot(x,y,"h")
> barplot(x)
> pie(y)
Files
Plots
Packages
Help
Viewer
Zoom
Export



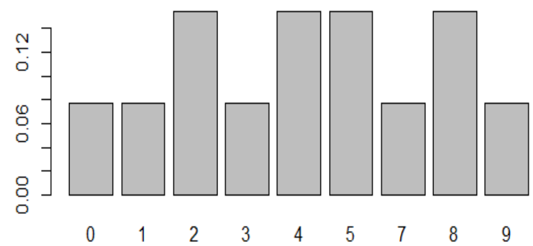
Console
Terminal
Jobs
> barplot(x)
> pie(y)
> t=table(y)
> t
y
1 2 3 4 5 6 7 8 9
2 1 2 1 2 1 2 1 1
> table(x)
x
0 1 2 3 4 5 7 8 9
1 1 2 1 2 2 1 2 1
> pie(table(x)/lenght(x))
Error in lenght(x) : could not find function "lenght"
> pie(table(x)/length(x))



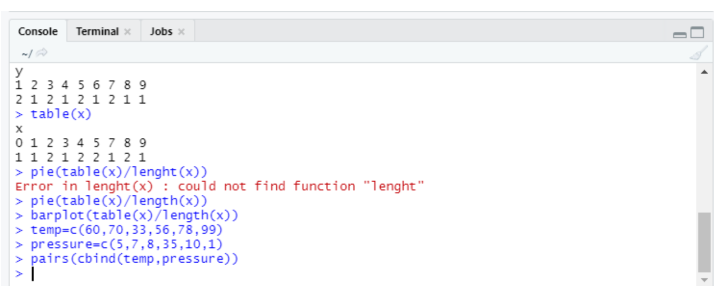
2
1
0
9
8
7
5
4
3



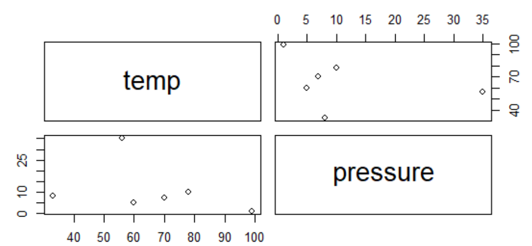
Console
Terminal
Jobs
> pie(y)
> t=table(y)
> t
y
1 2 3 4 5 6 7 8 9
2 1 2 1 2 1 2 1 1
> table(x)
x
0 1 2 3 4 5 7 8 9
1 1 2 1 2 2 1 2 1
> pie(table(x)/lenght(x))
Error in lenght(x) : could not find function "lenght"
> pie(table(x)/length(x))
> barplot(table(x)/length(x))



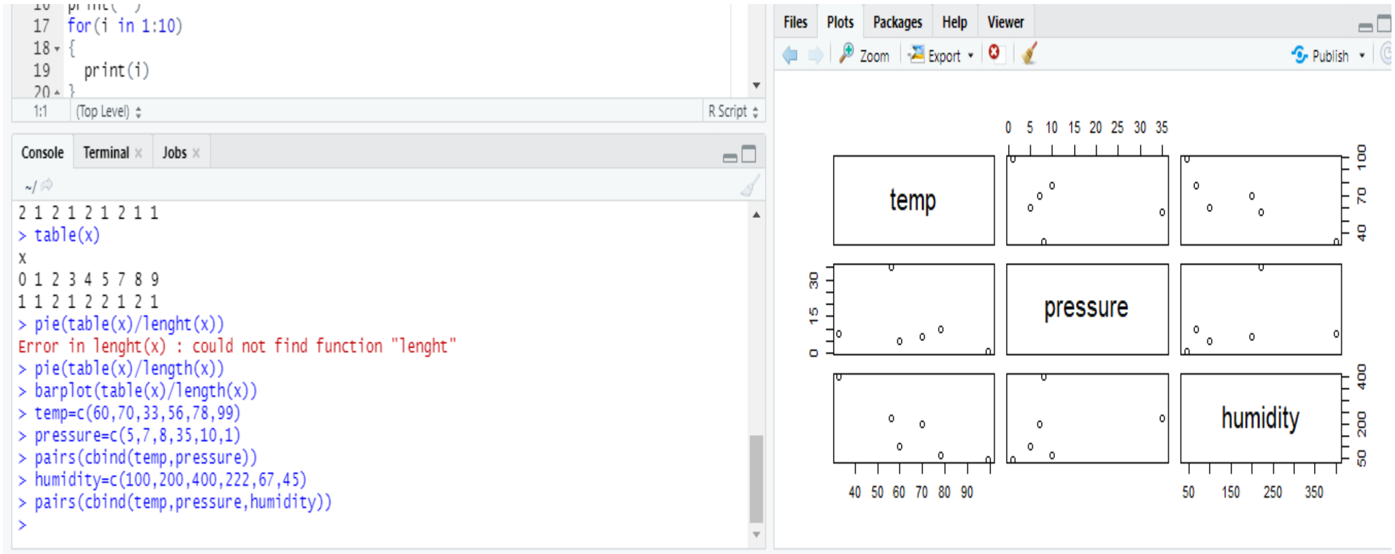
0.00
0.06
0.12
0
1
2
3
4
5
7
8
9



Console
Terminal
Jobs
y
1 2 3 4 5 6 7 8 9
2 1 2 1 2 1 2 1 1
> table(x)
x
0 1 2 3 4 5 7 8 9
1 1 2 1 2 2 1 2 1
> pie(table(x)/lenght(x))
Error in lenght(x) : could not find function "lenght"
> pie(table(x)/length(x))
> barplot(table(x)/length(x))
> temp=c(60,70,33,56,78,99)
> pressure=c(5,7,8,35,10,1)
> pairs(cbind(temp,pressure))



temp
pressure
0 5 10 15 20 25 30 35
40 70 100
0 10 25
40 50 60 70 80 90 100



11. Learning Outcomes Achieved:

1. Understood various graphical visualization of data sets.
2. Understood the use of tables.

12. Conclusion:

Understood the exploratory data analysis and the methods required to do it in R.

13. Experiment/Assignment Evaluation

<table>
<tr><td colspan="6">Experiment/Assignment Evaluation:</td></tr>
<tr><td>Sr. No.</td><td colspan="3">Parameters</td><td>Marks obtained</td><td>Out of</td></tr>
<tr><td>1</td><td colspan="3">Technical Understanding (Assessment may be done based on Q & A <u>or</u> any other relevant method.) Teacher should mention the other method used -</td><td></td><td>6</td></tr>
<tr><td>2</td><td colspan="3">Neatness/presentation</td><td></td><td>2</td></tr>
<tr><td>3</td><td colspan="3">Punctuality</td><td></td><td>2</td></tr>
<tr><td colspan="2">Date of performance (DOP)</td><td></td><td>Total marks obtained</td><td></td><td>10</td></tr>
<tr><td colspan="2">Date of checking (DOC)</td><td></td><td colspan="3">Signature of teacher</td></tr>
</table>

References:

1. URL: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf (Online Resources)
2. R Cookbook Paperback – 2011 by Teetor Paul O Reilly Publications
3. Beginning R: The Statistical Programming Language by Dr. Mark Gardener, Wiley Publications
4. R Programming For Dummies by Joris Meys Andrie de Vries, Wiley Publications

Viva Questions

1. What are different data visualization command and functions in R?
2. What is table?
3. How table is different than data frame?

Finolex Academy of Management and Technology, Ratnagiri

Department of Information Technology

Subject:	**R Programming Lab. (ITL804)**		
Class:	**BE IT / Semester – VIII (Rev-2016) / Academic year: 2020-21**		
Name of Student:	**Pranali Hanumant Kudtarkar**		
Roll No:	**26**	**Date of performance (DOP) :**	
Assignment/Experiment No:	**06**	**Date of checking (DOC) :**	
Title: Working with larger data-sets and introduction to ggplot2 graphics.			
Marks:		**Teacher's Signature:**	

1. Aim: To understand the exploratory data analysis and the methods required to do it in R.

2. Prerequisites:

1. Data-frames, tables, basic graphical functions.

3. Hardware Requirements:

1. PC with minimum 2GB RAM

4. Software Requirements:

1. Windows / Linux OS.
2. R version 3.6 or higher

5. Learning Objectives:

1. To understand the sources of larger data sets..
2. To understand how the larger data-sets are maintained and managed.
3. To understand the basic usages of ggplot2 graphics package.

6. Learning Objectives Applicable: LO 3, LO 5

7. Program Outcomes Applicable: PO 4, PO 5

8. Program Education Objectives Applicable: PEO 4, PEO 6

9. Theory:

Working with Large Datasets

The learning objectives of this section are to:

- Read and manipulate large datasets

R now offers now offers a variety of options for working with large datasets. We won't try to cover all these options in detail here, but rather give an overview of strategies to consider if you need to work with a large dataset, as well as point you to additional resources to learn more about working with large datasets in R.

A> While there are a variety of definitions of how large a dataset must be to qualify as "large," in this section we don't formally define a limit. Instead, this section is meant to give you some strategies anytime you work with a dataset large enough that you notice it's causing problems. For example, data large enough for R to be noticeably slow to read or manipulate the data, or large enough it's difficult to store the data locally on your computer.

Graphics with ggplot2

The ggplot2 package, created by Hadley Wickham, offers a powerful graphics language for creating elegant and complex plots. Its popularity in the R community has exploded in recent years. Origianlly based on Leland Wilkinson's The Grammar of Graphics, ggplot2 allows you to create graphs that represent both univariate and multivariate numerical and categorical data in a straightforward manner. Grouping can be represented by color, symbol, size, and transparency. The creation of trellis plots (i.e., conditioning) is relatively simple.

Mastering the **ggplot2** language can be challenging (see the **Going Further** section below for helpful resources). There is a helper function called **qplot()** (for quick plot) that can hide much of this complexity when creating standard graphs.

qplot()

The **qplot()** function can be used to create the most common graph types. While it does not expose **ggplot**'s full power, it can create a very wide range of useful plots. The format is:

```
qplot(x, y, data=, color=, shape=, size=, alpha=, geom=, method=, formula=, facets=, xlim=, ylim= xlab=, ylab=,
main=, sub=)
```

where the options are:

option	description

alpha	Alpha transparency for overlapping elements expressed as a fraction between 0 (complete transparency) and 1 (complete opacity)
color, shape, size, fill	Associates the levels of variable with symbol color, shape, or size. For line plots, color associates levels of a variable with line color. For density and box plots, fill associates fill colors with a variable. Legends are drawn automatically.
data	Specifies a data frame
facets	Creates a trellis graph by specifying conditioning variables. Its value is expressed as *rowvar ~ colvar*. To create trellis graphs based on a single conditioning variable, use *rowvar~*. or .*~colvar*)
geom	Specifies the geometric objects that define the graph type. The geom option is expressed as a character vector with one or more entries. geom values include "point", "smooth", "boxplot", "line", "histogram", "density", "bar", and "jitter".
main, sub	Character vectors specifying the title and subtitle
method, formula	If geom="smooth", a loess fit line and confidence limits are added by default. When the number of observations is greater than 1,000, a more efficient smoothing algorithm is employed. Methods include "lm" for regression, "gam" for generalized additive models, and "rlm" for robust regression. The formula parameter gives the form of the fit. For example, to add simple linear regression lines, you'd specify geom="smooth", method="lm", formula=y~x. Changing the formula to y~poly(x,2) would produce a quadratic fit. Note that the formula uses the letters x and y, not the names of the variables. For method="gam", be sure to load the mgcv package. For method="rml", load the MASS package.
x, *y*	Specifies the variables placed on the horizontal and vertical axis. For univariate plots (for example, histograms), omit *y*
xlab, ylab	Character vectors specifying horizontal and vertical axis labels
xlim,ylim	Two-element numeric vectors giving the minimum and maximum values for the horizontal and vertical axes, respectively

10. Results:

```
Console   Terminal ×   Jobs ×
E:/R prog/
>
> fr=read.csv("lendingdata.csv")
> fr
    X                       activity borrower_genders
1   0                Home Appliances            group
2   1                        Cereals           female
3   2                 Clothing Sales           female
4   3                 Clothing Sales           female
5   4                   Fish Selling           female
6   5        Personal Products Sales           female
7   6                 Transportation             male
8   7                Home Appliances            group
9   8                         Bakery             male
10  9                        Farming             male
11 10 Primary/secondary school costs           female
12 11           Construction Supplies           female
13 12                         Retail           female
14 13        Personal Products Sales           female
15 14                Cosmetics Sales           female
16 15          Higher education costs          female
17 16                  General Store             male
18 17                         Retail           female
19 18                        Farming            group
20 19                  Used Clothing           female
21 20       Personal Housing Expenses           group
22 21                        Farming            group
23 22                  Manufacturing           female
24 23                        Farming             male
25 24                  Cheese Making           female
```

Filter

	X	activity	borrower_genders	country	country_code	currency_pol
1	0	Home Appliances	group	Cambodia	KH	shared
2	1	Cereals	female	Philippines	PH	shared
3	2	Clothing Sales	female	Peru	PE	shared
4	3	Clothing Sales	female	Tajikistan	TJ	not sharec
5	4	Fish Selling	female	Uganda	UG	not sharec
6	5	Personal Products Sales	female	Jordan	JO	shared
7	6	Transportation	male	Tajikistan	TJ	not sharec
8	7	Home Appliances	group	Cambodia	KH	shared
9	8	Bakery	male	Nicaragua	NI	shared
10	9	Farming	male	Nigeria	NG	shared
11	10	Primary/secondary school costs	female	Colombia	CO	not sharec
12	11	Construction Supplies	female	Nicaragua	NI	shared
13	12	Retail	female	Colombia	CO	not sharec
14	13	Personal Products Sales	female	Philippines	PH	not sharec
15	14	Cosmetics Sales	female	Ecuador	EC	not sharec
16	15	Higher education costs	female	Colombia	CO	shared
17	16	General Store	male	Honduras	HN	shared
18	17	Retail	female	Benin	BJ	shared

Showing 1 to 18 of 27,518 entries, 15 total columns

```
Console   Terminal ×   Jobs ×
E:/R prog/
> View(fr)
>
```

Console   Terminal ×   Jobs ×

E:/R prog/

```
> head(fr)
  X             activity borrower_genders     country country_code
1 0         Home Appliances           group    Cambodia           KH
2 1                Cereals          female Philippines           PH
3 2          Clothing Sales          female        Peru           PE
4 3          Clothing Sales          female  Tajikistan           TJ
5 4            Fish Selling          female      Uganda           UG
6 5 Personal Products Sales          female      Jordan           JO
  currency_policy distribution_model lender_count loan_amount original_language
1          shared       field_partner            5         125           English
2          shared       field_partner            5         125           English
3          shared       field_partner           10         375           Spanish
4      not shared       field_partner           27         850           English
5      not shared       field_partner           17         550           English
6          shared       field_partner           22         575           English
  repayment_interval       sector status term_in_months  rMPI
1            monthly Personal Use funded              8 0.097
2          irregular         Food funded             14 0.055
3            monthly     Clothing funded              8 0.000
4            monthly     Clothing funded             14 0.021
5          irregular         Food funded              9 0.417
6            monthly       Retail funded             15 0.005
>
```

Console   Terminal ×   Jobs ×

E:/R prog/

```
> head(fr,3)
  X        activity borrower_genders     country country_code currency_policy
1 0 Home Appliances            group    Cambodia           KH          shared
2 1         Cereals           female Philippines           PH          shared
3 2  Clothing Sales           female        Peru           PE          shared
  distribution_model lender_count loan_amount original_language repayment_interval
1      field_partner            5         125           English            monthly
2      field_partner            5         125           English          irregular
3      field_partner           10         375           Spanish            monthly
       sector status term_in_months  rMPI
1 Personal Use funded              8 0.097
2        Food funded             14 0.055
3    Clothing funded              8 0.000
> |
```

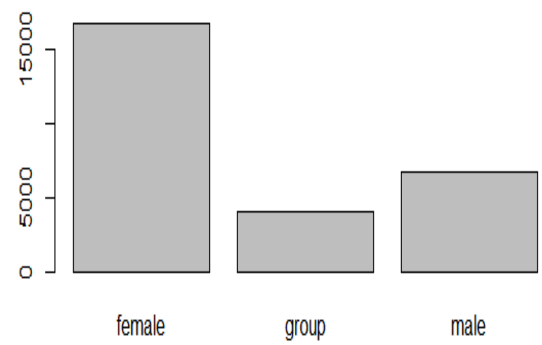
Console   Terminal ×   Jobs ×

E:/R prog/

```
3    Clothing funded              8 0.000
> tail(fr)
          X       activity borrower_genders     country country_code currency_policy
27513 27530 Transportation             male        Peru           PE          shared
27514 27531       Services           female      Zambia           ZM          shared
27515 27532           Pigs           female Philippines           PH      not shared
27516 27533  General Store           female Philippines           PH          shared
27517 27534    Food Market           female       Ghana           GH          shared
27518 27535        Farming             male  Tajikistan           TJ      not shared
      distribution_model lender_count loan_amount original_language
27513      field_partner           16        1025           Spanish
27514      field_partner           85        2300           English
27515      field_partner           17         475           English
27516      field_partner           38        1075           English
27517      field_partner           35        1075           English
27518      field_partner           55        2000           English
      repayment_interval         sector status term_in_months  rMPI
27513          irregular Transportation funded              6 0.062
27514             bullet       Services funded              9 0.158
27515          irregular    Agriculture funded              8 0.422
27516          irregular         Retail funded             11 0.055
27517          irregular           Food funded              6 0.122
```

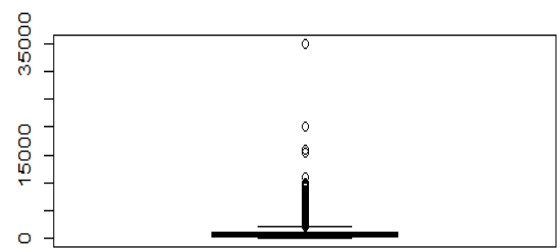
```
> tail(fr,2)
          X    activity borrower_genders    country country_code currency_policy
27517 27534 Food Market           female      Ghana           GH          shared
27518 27535     Farming             male Tajikistan           TJ      not shared
      distribution_model lender_count loan_amount original_language
27517      field_partner           35        1075           English
27518      field_partner           55        2000           English
      repayment_interval      sector status term_in_months  rMPI
27517          irregular        Food funded              6 0.122
27518            monthly Agriculture funded             14 0.033
>
```

```
27514            bullet    Services funded              9 0.158
27515         irregular Agriculture funded              8 0.422
27516         irregular      Retail funded             11 0.055
27517         irregular        Food funded              6 0.122
27518           monthly Agriculture funded             14 0.033
> tail(fr,2)
          X    activity borrower_genders    country country_code currency_policy
27517 27534 Food Market           female      Ghana           GH          shared
27518 27535     Farming             male Tajikistan           TJ      not shared
      distribution_model lender_count loan_amount original_language
27517      field_partner           35        1075           English
27518      field_partner           55        2000           English
      repayment_interval      sector status term_in_months  rMPI
27517          irregular        Food funded              6 0.122
27518            monthly Agriculture funded             14 0.033
> barplot(table(fr$borrower_genders))
```

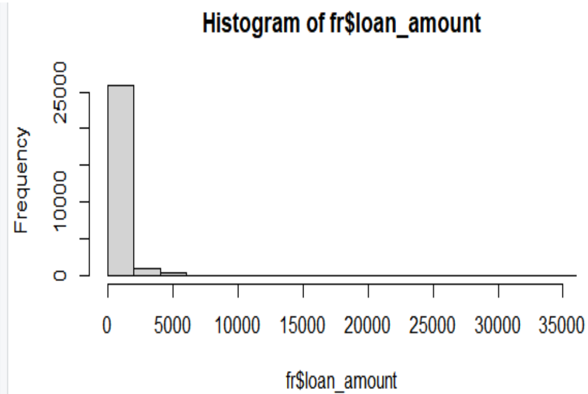



```
> summary(fr)
       X           activity          borrower_genders     country         
 Min.   :    0   Length:27518       Length:27518       Length:27518      
 1st Qu.: 6886   Class :character   Class :character   Class :character  
 Median :13768   Mode  :character   Mode  :character   Mode  :character  
 Mean   :13768                                                            
 3rd Qu.:20649                                                            
 Max.   :27535                                                            
 country_code       currency_policy    distribution_model  lender_count   
 Length:27518       Length:27518       Length:27518       Min.   :  0.00  
 Class :character   Class :character   Class :character   1st Qu.:  8.00  
 Mode  :character   Mode  :character   Mode  :character   Median : 14.00  
                                                          Mean   : 20.28  
                                                          3rd Qu.: 25.00  
                                                          Max.   :653.00  
  loan_amount      original_language  repayment_interval    sector         
 Min.   :   25.0   Length:27518       Length:27518       Length:27518      
 1st Qu.:  300.0   Class :character   Class :character   Class :character  
 Median :  550.0   Mode  :character   Mode  :character   Mode  :character  
 Mean   :  792.1                                                            
 3rd Qu.: 1000.0                                                            
 Max.   :35000.0                                                            
    status          term_in_months        rMPI       
 Length:27518       Min.   :  2.00   Min.   :0.0000  
 Class :character   1st Qu.:  8.00   1st Qu.:0.0390  
 Mode  :character   Median : 13.00   Median :0.0750  
                    Mean   : 13.29   Mean   :0.1296  
                    3rd Qu.: 14.00   3rd Qu.:0.1610  
                    Max.   :145.00   Max.   :0.6960  
> 
```

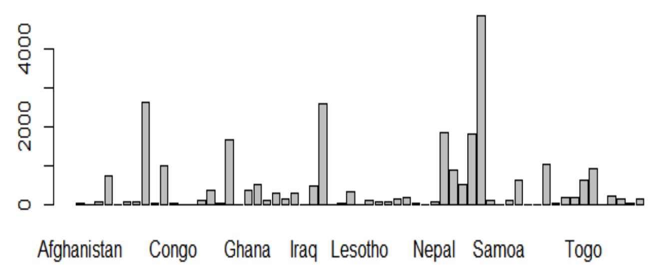
```
                                                              Max.   :653.00
 loan_amount      original_language  repayment_interval    sector
 Min.   :   25.0   Length:27518       Length:27518       Length:27518
 1st Qu.:  300.0   Class :character   Class :character   Class :character
 Median :  550.0   Mode  :character   Mode  :character   Mode  :character
 Mean   :  792.1
 3rd Qu.: 1000.0
 Max.   :35000.0
    status          term_in_months        rMPI
 Length:27518       Min.   :  2.00   Min.   :0.0000
 Class :character   1st Qu.:  8.00   1st Qu.:0.0390
 Mode  :character   Median : 13.00   Median :0.0750
                    Mean   : 13.29   Mean   :0.1296
                    3rd Qu.: 14.00   3rd Qu.:0.1610
                    Max.   :145.00   Max.   :0.6960
> boxplot(fr$loan_amount)
```



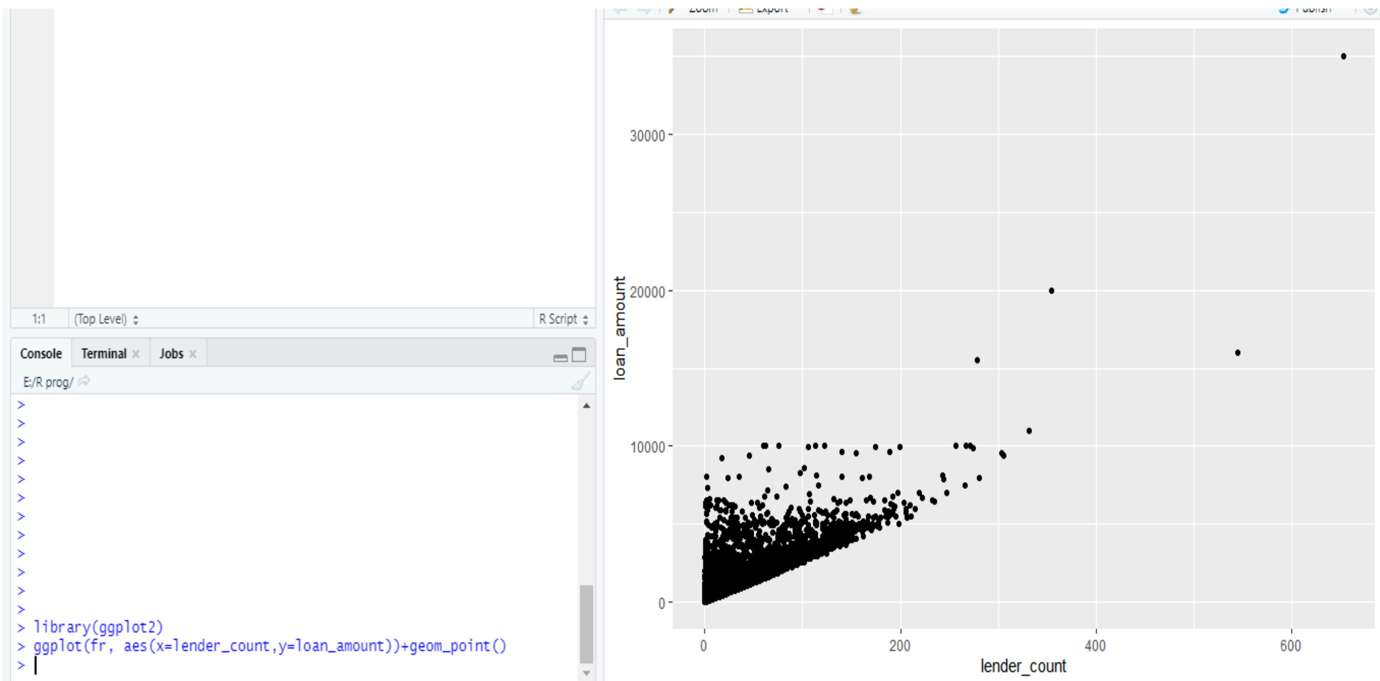

```
                                                              3rd Qu.: 25.00
                                                              Max.   :653.00
 loan_amount      original_language  repayment_interval    sector
 Min.   :   25.0   Length:27518       Length:27518       Length:27518
 1st Qu.:  300.0   Class :character   Class :character   Class :character
 Median :  550.0   Mode  :character   Mode  :character   Mode  :character
 Mean   :  792.1
 3rd Qu.: 1000.0
 Max.   :35000.0
    status          term_in_months        rMPI
 Length:27518       Min.   :  2.00   Min.   :0.0000
 Class :character   1st Qu.:  8.00   1st Qu.:0.0390
 Mode  :character   Median : 13.00   Median :0.0750
                    Mean   : 13.29   Mean   :0.1296
                    3rd Qu.: 14.00   3rd Qu.:0.1610
                    Max.   :145.00   Max.   :0.6960
> boxplot(fr$loan_amount)
> hist(fr$loan_amount)
>
```

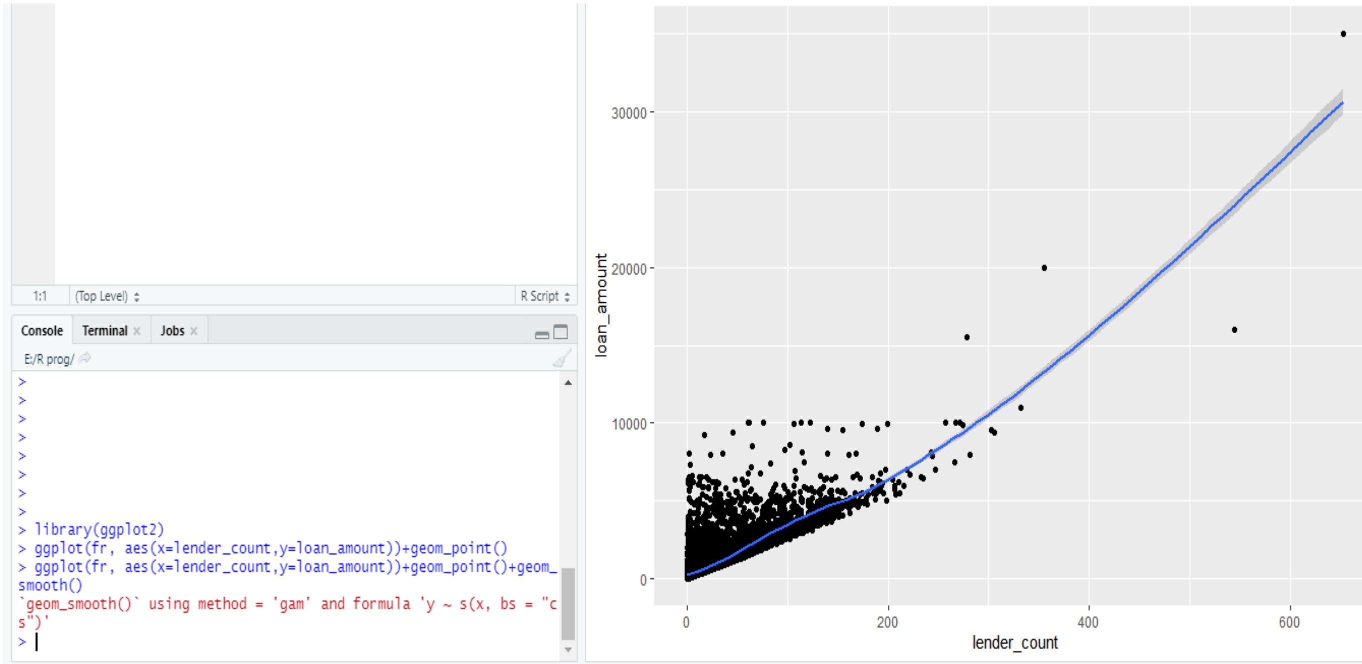


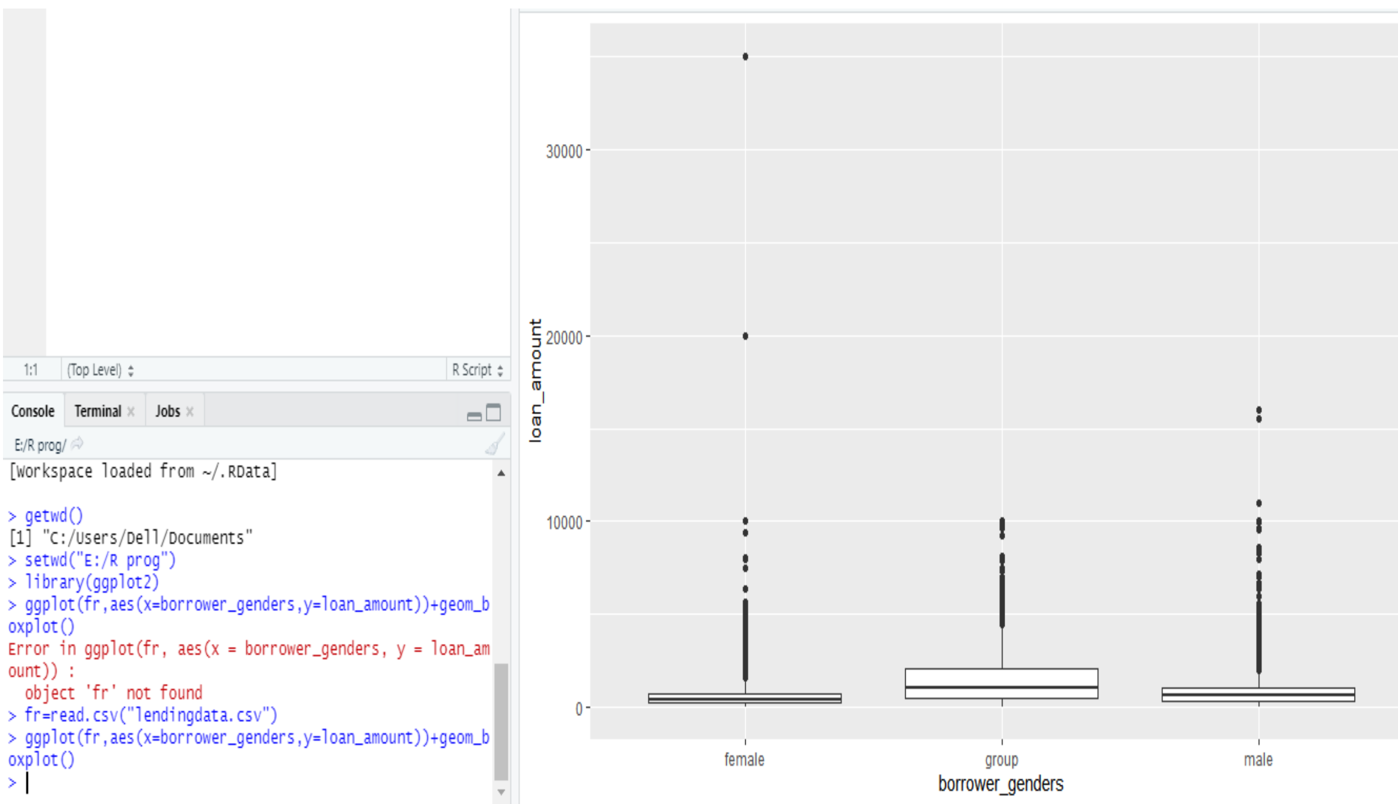

```
Console  Terminal  Jobs
E:/R prog/

 Max.   :35000.0

    status          term_in_months        rMPI
 Length:27518       Min.   :  2.00   Min.   :0.0000
 Class :character   1st Qu.:  8.00   1st Qu.:0.0390
 Mode  :character   Median : 13.00   Median :0.0750
                    Mean   : 13.29   Mean   :0.1296
                    3rd Qu.: 14.00   3rd Qu.:0.1610
                    Max.   :145.00   Max.   :0.6960
> boxplot(fr$loan_amount)
> hist(fr$loan_amount)
> barplot(table(fr$country))
>
```



```
> t=table(fr$country)
> t

                    Afghanistan
                             62
                         Belize
                              7
                          Benin
                            100
                        Bolivia
                            758
                         Brazil
                             17
                   Burkina Faso
                             99
                        Burundi
                             85
                       Cambodia
                           2629
                       Cameroon
                             36
                       Colombia
                           1005
                          Congo
                             34
                     Costa Rica
                              6
                  Cote D'Ivoire
                              8
```







11. Learning Outcomes Achieved:

1. Understood the sources of larger data sets..
2. Understood how the larger data-sets are maintained and managed.
3. Understood the basic usages of ggplot2 graphics package.

12. Conclusion:

Worked with larger data-sets and introduction to ggplot2 graphics.

13. Experiment/Assignment Evaluation

<table>
<tr><td colspan="5">Experiment/Assignment Evaluation:</td></tr>
<tr><td>Sr. No.</td><td colspan="2">Parameters</td><td>Marks obtained</td><td>Out of</td></tr>
<tr><td>1</td><td colspan="2">Technical Understanding (Assessment may be done based on Q & A <u>or</u> any other relevant method.) Teacher should mention the other method used -</td><td></td><td>6</td></tr>
<tr><td>2</td><td colspan="2">Neatness/presentation</td><td></td><td>2</td></tr>
<tr><td>3</td><td colspan="2">Punctuality</td><td></td><td>2</td></tr>
<tr><td>Date of performance (DOP)</td><td></td><td>Total marks obtained</td><td></td><td>10</td></tr>
<tr><td>Date of checking (DOC)</td><td></td><td colspan="3">Signature of teacher</td></tr>
</table>

References:

1. URL: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf (Online Resources)
2. R Cookbook Paperback – 2011 by Teetor Paul O Reilly Publications
3. Beginning R: The Statistical Programming Language by Dr. Mark Gardener, Wiley Publications
4. R Programming For Dummies by Joris Meys Andrie de Vries, Wiley Publications

Viva Questions

1. What are different ways to store larger data-set?
2. What are names of packages required to extract data from data-set stored in standard spreadsheet.
3. What are various plotting functions in ggplot2?

Finolex Academy of Management and Technology, Ratnagiri

Department of Information Technology

Subject:	**R Programming Lab. (ITL804)**		
Class:	**BE IT / Semester – VIII (Rev-2016) / Academic year: 2020-21**		
Name of Student:	**Pranali Hanumant Kudtarkar**		
Roll No:	**26**	**Date of performance (DOP) :**	
Assignment/Experiment No:	**07**	**Date of checking (DOC) :**	
Title: Program to demonstrate regression and correlation in tabular data including categorical data.			
Marks:		**Teacher's Signature:**	

1. Aim: To understand regression and correlation in tabular data including categorical data.

2. Prerequisites:

1. Working with larger data-sets.

3. Hardware Requirements:

1. PC with minimum 2GB RAM

4. Software Requirements:

1. Windows / Linux OS.
2. R version 3.6 or higher

5. Learning Objectives:

1. To understand the basic elements of larger data-sets.
2. To understand numerical and categorical variables in larger data-sets.
3. To understand how to apply regression to design decision model on the larger data-sets.

6. Learning Objectives Applicable: LO 5, LO 6

7. Program Outcomes Applicable: PO 4, PO 5

8. Program Education Objectives Applicable: PEO 4, PEO 6

9. Theory:

Correlation plots

Correlation plots help you to visualize the pairwise relationships between a set of quantitative variables by displaying their correlations using color or shading.

Consider the Saratoga Houses dataset, which contains the sale price and characteristics of Saratoga County, NY homes in 2006. In order to explore the relationships among the quantitative variables, we can calculate the Pearson Product-Moment correlation coefficients.

```
data(SaratogaHouses, package="mosaicData")

# select numeric variables
df <- dplyr::select_if(SaratogaHouses, is.numeric)

# calulate the correlations
r <- cor(df, use="complete.obs")
round(r,2)
##            price lotSize   age landValue livingArea pctCollege bedrooms
## price       1.00    0.16 -0.19      0.58       0.71       0.20     0.40
## lotSize     0.16    1.00 -0.02      0.06       0.16      -0.03     0.11
## age        -0.19   -0.02  1.00     -0.02      -0.17      -0.04     0.03
## landValue   0.58    0.06 -0.02      1.00       0.42       0.23     0.20
## livingArea  0.71    0.16 -0.17      0.42       1.00       0.21     0.66
## pctCollege  0.20   -0.03 -0.04      0.23       0.21       1.00     0.16
## bedrooms    0.40    0.11  0.03      0.20       0.66       0.16     1.00
## fireplaces  0.38    0.09 -0.17      0.21       0.47       0.25     0.28
## bathrooms   0.60    0.08 -0.36      0.30       0.72       0.18     0.46
## rooms       0.53    0.14 -0.08      0.30       0.73       0.16     0.67
##            fireplaces bathrooms rooms
## price            0.38      0.60  0.53
## lotSize          0.09      0.08  0.14
## age             -0.17     -0.36 -0.08
## landValue        0.21      0.30  0.30
## livingArea       0.47      0.72  0.73
## pctCollege       0.25      0.18  0.16
## bedrooms         0.28      0.46  0.67
## fireplaces       1.00      0.44  0.32
## bathrooms        0.44      1.00  0.52
## rooms            0.32      0.52  1.00
```

The `ggcorrplot` function in the `ggcorrplot` package can be used to visualize these correlations. By default, it creates a `ggplot2` graph were darker red indicates stronger positive correlations, darker blue indicates stronger negative correlations and white indicates no correlation.

```
library(ggplot2)
library(ggcorrplot)
ggcorrplot(r)
```

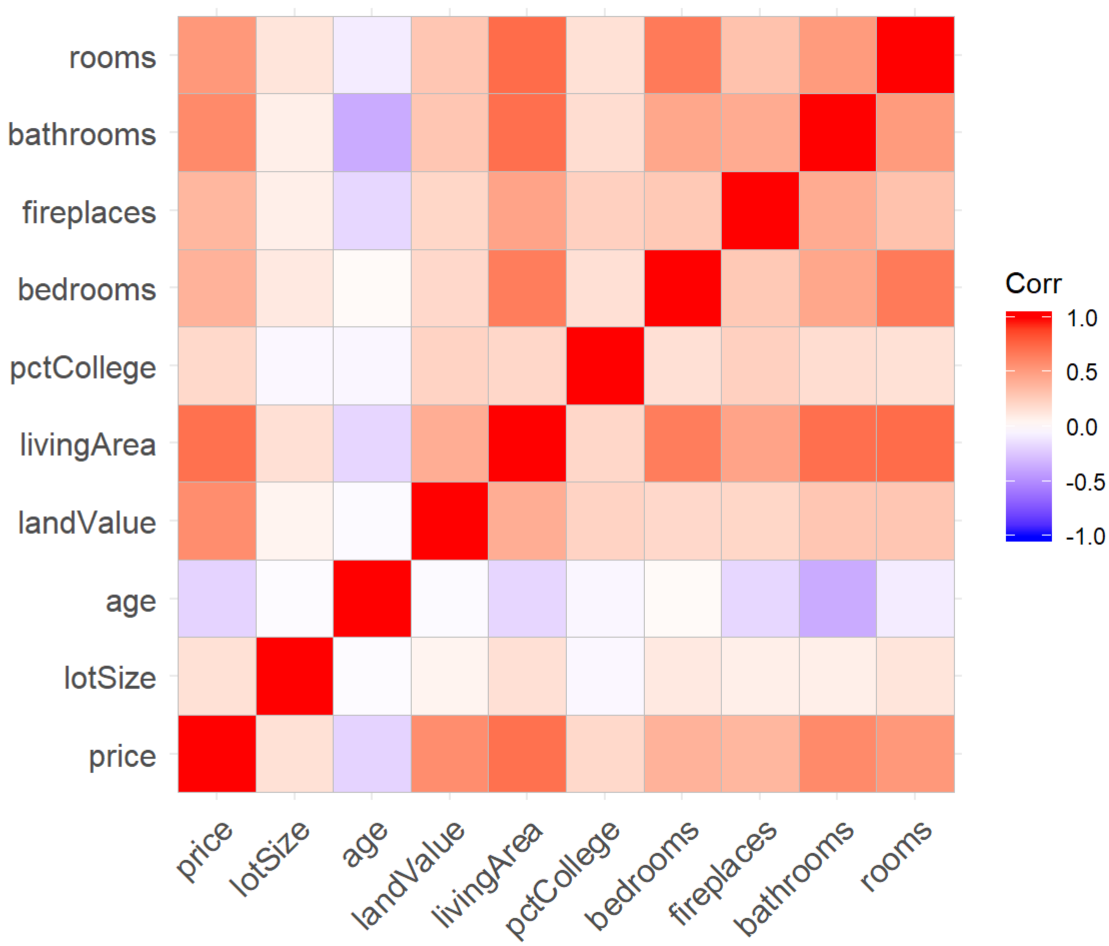


Figure 1: Correlation matrix

From the graph, an increase in number of bathrooms and living area are associated with increased price, while older homes tend to be less expensive. Older homes also tend to have fewer bathrooms.

The ggcorrplot function has a number of options for customizing the output. For example

- hc.order = TRUE reorders the variables, placing variables with similar correlation patterns together.
- type = "lower" plots the lower portion of the correlation matrix.
- lab = TRUE overlays the correlation coefficients (as text) on the plot.

Linear Regression

Linear regression allows us to explore the relationship between a quantitative response variable and an explanatory variable while other variables are held constant.

Consider the prediction of home prices in the Saratoga dataset from lot size (square feet), age (years), land value (1000s dollars), living area (square feet), number of bedrooms and bathrooms and whether the home is on the waterfront or not.

```r
data(SaratogaHouses, package="mosaicData")
houses_lm <- lm(price ~ lotSize + age + landValue +
                livingArea + bedrooms + bathrooms +
                waterfront,
              data = SaratogaHouses)
```

Table 8.1: Linear Regression results

term	estimate	std.error	statistic	p.value
(Intercept)	139878.80	16472.93	8.49	0.00
lotSize	7500.79	2075.14	3.61	0.00
age	-136.04	54.16	-2.51	0.01
landValue	0.91	0.05	19.84	0.00
livingArea	75.18	4.16	18.08	0.00
bedrooms	-5766.76	2388.43	-2.41	0.02
bathrooms	24547.11	3332.27	7.37	0.00
waterfrontNo	-120726.62	15600.83	-7.74	0.00

From the results, we can estimate that an increase of one square foot of living area is associated with a home price increase of $75, holding the other variables constant. Additionally, waterfront home cost approximately $120,726 more than non-waterfront home, again controlling for the other variables in the model.

The visreg package provides tools for visualizing these conditional relationships.

The `visreg` function takes (1) the model and (2) the variable of interest and plots the conditional relationship, controlling for the other variables. The option `gg = TRUE` is used to produce a `ggplot2` graph.

```r
# conditional plot of price vs. living area
library(ggplot2)
library(visreg)
visreg(houses_lm, "livingArea", gg = TRUE)
```

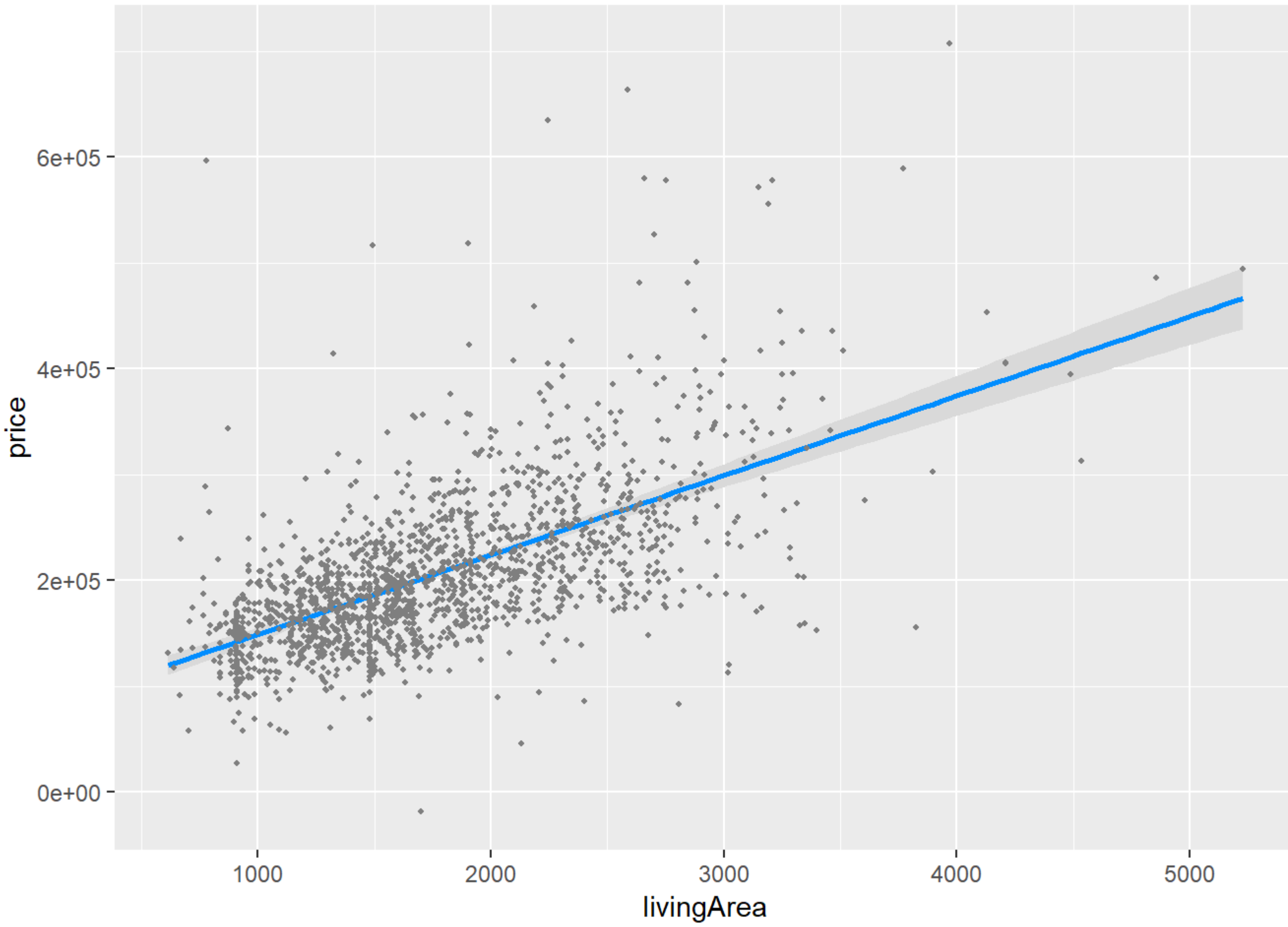
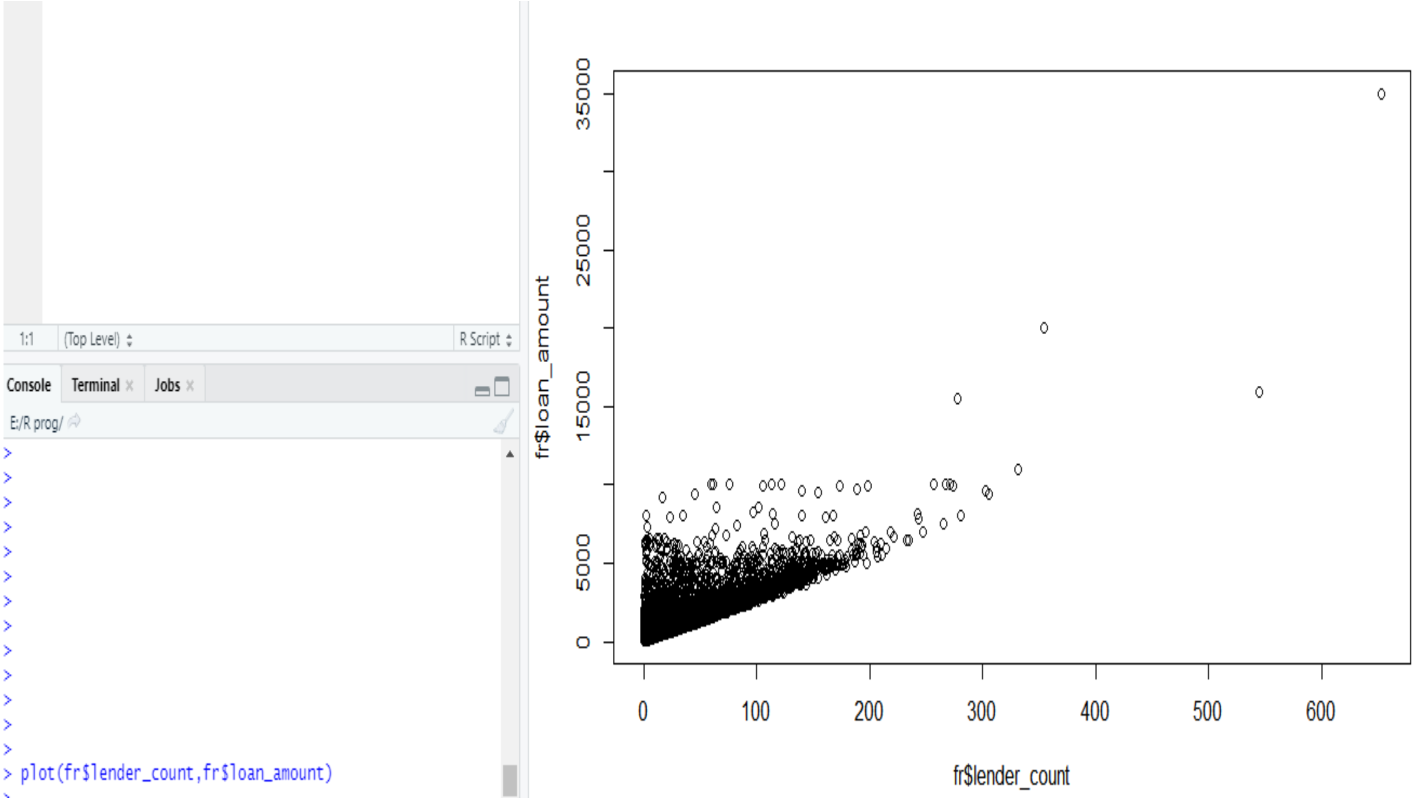


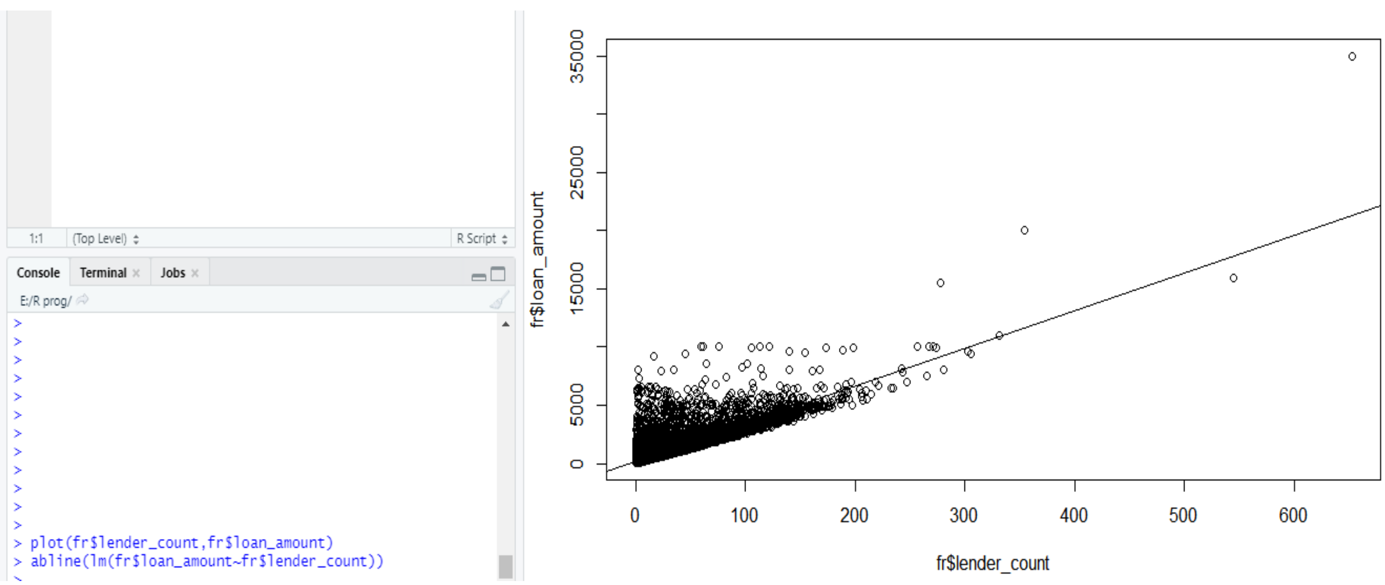
Figure 2: Conditional plot of living area and price

The graph suggests that, after controlling for lot size, age, living area, number of bedrooms and bathrooms, and waterfront location, sales price increases with living area in a linear fashion.

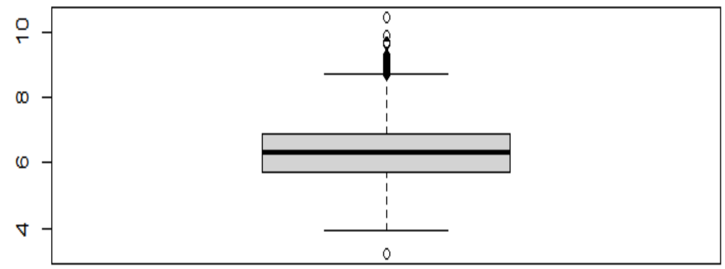
How does visreg work? The fitted model is used to predict values of the response variable, across the range of the chosen explanatory variable. The other variables are set to their median value (for numeric variables) or most frequent category (for categorical variables). The user can override these defaults and chose specific values for any variable in the model.

10. Results:


1:1
(Top Level)
R Script
Console
Terminal
Jobs
E:/R prog/
> plot(fr$lender_count,fr$loan_amount)
fr$loan_amount
0
5000
15000
25000
35000
0
100
200
300
400
500
600
fr$lender_count


1:1
(Top Level)
R Script
Console
Terminal
Jobs
E:/R prog/
> plot(fr$lender_count,fr$loan_amount)
> abline(lm(fr$loan_amount~fr$lender_count))
fr$loan_amount
0
5000
15000
25000
35000
0
100
200
300
400
500
600
fr$lender_count

```
>
>
>
> plot(fr$lender_count,fr$loan_amount)
> abline(lm(fr$loan_amount~fr$lender_count))
> x=c(1,2,3,4,5,6,7)
> y=c(1,2,3,4,5,6,7)
> cor(x,y)
[1] 1
> x=c(-1,2,-3,4)
> y=c(1,-2,3,-4)
> cor(x,y)
[1] -1
> z=c(-1,2,2,-9)
> cor(x,z)
[1] -0.7015157
> boxplot(log(fr$loan_amount))
>
```




```
> data=split(fr$loan_amount,fr$borrower_genders)
> data
$female
   [1]  125  375  850  550  575  325  400 1025  450 1650  225 1025
  [13]  325  225 1175 1450  525  200  100  625  700 1000  300  475
  [25]  750 1150  925  175  525  200  900 1150  125  700  600  775
  [37]  250  775  625  200  300  325  500  200  125 1200 1125  675
  [49]  325 1000  700  300  400  325  475  350 2475  400  450  600
  [61]  800  800  750 1500  100  400  400  250 1000 1000  425  950
  [73] 1725  175 1000  600  625  875  200  325   75 1300 1000  150
  [85]  300  200  300  275 1150 1475  250  125  500  150  175  725
  [97]  400  225  575  350 1000 2500  250  600  450  550  400  150
 [109]  175  525  325  800  550  200  500  325  300  250  375  175
 [121]  550  550  325  325  175  900  200  750  800  175  625 1375
 [133]  475 1200  500  400  850 1150  550  850 1875  275   75  200
 [145] 1250  325  525  500 2175 1000  225  525  150  250  125  875
 [157]  150  150  475  600  475  250  175  325  150  200  600  550
 [169]  875  200  350 1150  500  600  250 1000  175  325  575  625
 [181] 1400 1200  775 1325  200  300  275 1125 1000 1300  575  250
 [193]  125  125  225  725  900  500  375  300  950  600  200  550
 [205]  350  800 1500  225  225  350  300  975 1000  450 1050  400
 [217]  500  600  250  150  200   50  225  850 1050  500  475  125
 [229]  125  400  350  525  600  325  775  150  225  525   50  275
 [241]  300  275  500  750  250  400  425  875  350  450  150  325
 [253]  250  300  375  475  350  125  200  425  225  600  800  275
 [265]  950  400  175  150  225  625  275  475  225  675  100 1025
 [277]  375  175  125  900 1200  250 1250  600  550  850  325  600
 [289]  775  500  600  475  500 1050 1600  300  275  400  325  125
 [301]  125  325  775  675  275  750  300  125  200  250  475  825
 [313]  400 1175  275  725  500  275 1025  675 1000   75 1000  175
 [325] 1025  175  600  475  150  450  300  325  325  325  425  600
```

Console | Terminal | Jobs

E:/R prog/

```
> data$male
  [1] 1125 1400  600  775  600  775  600  600 1000 1000  800  500
 [13] 1000  200 1025  450  350  700  175  225  175  525 1400  875
 [25]  400 1125 1500  300  650  500  375 1125  300  500 1000  250
 [37]  500 1200  500 1000 1050  300  700  200  400  625  600  600
 [49]  325  200  950 1200 1000 1000  600  575 1000  200  600  125
 [61]  600 2100  250   50 1325  900 1125  225  250 1175  225 1200
 [73] 1450 1175  450   50 1000  350  600 2350  700  325 3200  300
 [85] 1725  175 1000 3975  850  225  400  300 1125  200  950  525
 [97] 1150 1000  400 2000  550   75  150 1200  950  700 1000  825
[109]  775  125  700 2750  800  100  100  625 1500 1025  175  950
[121] 1025  475 2000  400  400   75 2100  325 1125  600  700  700
[133]  925 1100  750  375  800  250 1175  950  925  400  475 1000
[145]  525 1550  375 1000  150  500 1225 1500  800 2000 1200  550
[157]  150  300 2500 1000  100  200  400  600  350  500  150  550
[169]  475  475  100 1625  375 1150 2000  500  800  975 3000  375
[181]  900 1000 1175  800  500 1525 1500 1500  350  250  275  400
[193]  300 1050  725  450  950 1050 1100  625  625  725 1550  800
[205]  900 1225  375  300  425  325  225  650  575 1575  925  475
[217]  925  400  900  950  600 1500 1050 3100 1050 1600  525  850
[229] 2125  450  100  125  125 1500  200 1400  700  500  750  925
[241]  700  125  300 1050  550 1200 1075  775  975  725  850  250
[253]  875  650  625  400  575  500  150  600 1000 1000  550  300
[265]  225 1000  225  300 1000  475 4150 1100  100 1175  300  275
[277] 2025  600 1500 1100  300  250  500  225  200 2000  200  100
[289] 1075 1000 1500  975  225  650  725  325  525  125 1100 1875
[301]  250  450  525  925 1075  325  300 1500 1300  800  525  225
[313]  800  625  600  500 1450 1500  325  800  725  850  500 2125
[325] 1350 1200 1000 3000 1100 1150 1050  450  600 1075  500  175
[337] 1550  200 1525 1000  725  700  900  500  525 1500  825  300
```

1:1 (Top Level) R Script

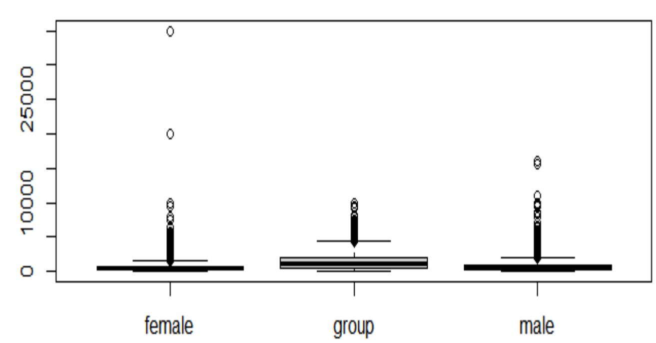
Console | Terminal | Jobs

E:/R prog/

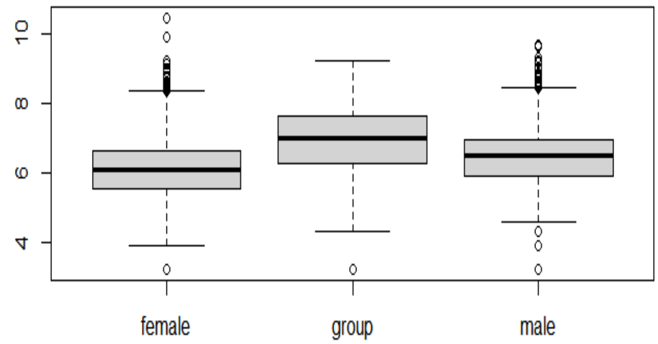
```
[841]  800  450  775  600 1050  650  200 2200 1150  200 6000  250
[853]  400 1300 4825 2050  125 1200  600 2525  200  900 1000 1225
[865]  300 1025   75  825  625  900 1075  500  375  550 1400  675
[877]  650  425 3475  825  400  700 1100 1075  950 2500  500  750
[889]  175  700  225  175  525  275  850  125  575  550  500  400
[901]  725 1000 1100  200  675 1175  825 1400  150 1775 1400  800
[913]  175  225  200 1325  625  450 1925 1550  300 1000 1000  600
[925] 1425  925 1000  700 1300  300  800  750  675  600  750 1575
[937]  925  300 1100  625 1000  425  425  200  800 2300  600 1300
[949]  375 1150  500  300  175 1450  750  700  250  275 1200  725
[961]  325 1475  100 1275  350 1650  800 1025 1025  600  650  725
[973] 1150  375 1000  400  250  400  900  475  450  200  725 1475
[985] 2000  625 1025  450  250  225   50  700  350  800  650  300
[997]  225  500  300  650
 [ reached getOption("max.print") -- omitted 5706 entries ]
> boxplot(data)
>
```

Files | Plots | Packages | Help | Viewer

Zoom | Export | Publish



```
Console   Terminal ×   Jobs ×
E:/R prog/
[865]  300 1025   75  825  625  900 1075  500  375  550 1400  675
[877]  650  425 3475  825  400  700 1100 1075  950 2500  500  750
[889]  175  700  225  175  525  275  850  125  575  550  500  400
[901]  725 1000 1100  200  675 1175  825 1400  150 1775 1400  800
[913]  175  225  200 1325  625  450 1925 1550  300 1000 1000  600
[925] 1425  925 1000  700 1300  300  800  750  675  600  750 1575
[937]  925  300 1100  625 1000  425  425  200  800 2300  600 1300
[949]  375 1150  500  300  175 1450  750  700  250  275 1200  725
[961]  325 1475  100 1275  350 1650  800 1025 1025  600  650  725
[973] 1150  375 1000  400  250  400  900  475  450  200  725 1475
[985] 2000  625 1025  450  250  225   50  700  350  800  650  300
[997]  225  500  300  650
 [ reached getOption("max.print") -- omitted 5706 entries ]
> boxplot(data)
> data=split(log(fr$loan_amount),fr$borrower_genders)
> boxplot(data)
```



11. Learning Outcomes Achieved:

1. Understood the basic elements of larger data-sets.
2. Understood numerical and categorical variables in larger data-sets.
3. Understood how to apply regression to design decision model on the larger data-sets.

12. Conclusion:

Demonstrated regression and correlation in tabular data including categorical data.

13. Experiment/Assignment Evaluation

<table>
<tr><td colspan="6">Experiment/Assignment Evaluation:</td></tr>
<tr><td>Sr. No.</td><td colspan="3">Parameters</td><td>Marks obtained</td><td>Out of</td></tr>
<tr><td>1</td><td colspan="3">Technical Understanding (Assessment may be done based on Q & A <u>or</u> any other relevant method.) Teacher should mention the other method used -</td><td></td><td>6</td></tr>
<tr><td>2</td><td colspan="3">Neatness/presentation</td><td></td><td>2</td></tr>
<tr><td>3</td><td colspan="3">Punctuality</td><td></td><td>2</td></tr>
<tr><td colspan="2">Date of performance (DOP)</td><td></td><td>Total marks obtained</td><td></td><td>10</td></tr>
<tr><td colspan="2">Date of checking (DOC)</td><td></td><td colspan="3">Signature of teacher</td></tr>
</table>

References:

1. URL: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf (Online Resources)
2. R Cookbook Paperback – 2011 by Teetor Paul O Reilly Publications
3. Beginning R: The Statistical Programming Language by Dr. Mark Gardener, Wiley Publications
4. R Programming For Dummies by Joris Meys Andrie de Vries, Wiley Publications

Viva Questions

1. What does it mean by categorical variables in data-sets?
2. What does it mean by regression?
3. What is correlation and how is it useful in data-science?