R Programming

Day 1

To run a program, you need to select the lines to run

Intro

Path

To get the current working directory of the environment.

```
getwd()
```

```
list.files() #total number of files in the present working directory
length(list.files()) #display the filenames
```

Sets path (for Mac)

```
setwd("/Users/mac/Desktop/College/R")
```

Sets path (for windows)

```
setwd("D:\\College\\R course")
```

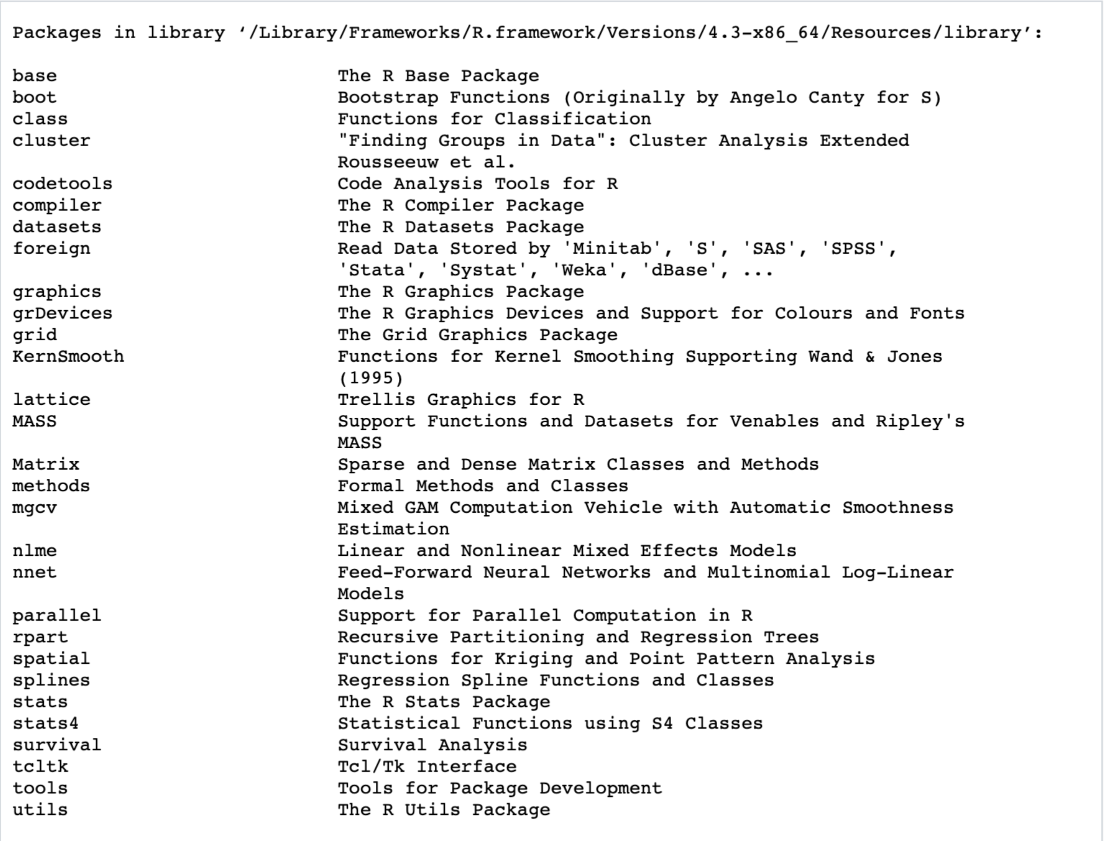
or

```
setwd("D:/College/R course")
```

double \ or single /

Built-in packages

```
library()
```



```
Packages in library ‘/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/library’:

base                    The R Base Package
boot                    Bootstrap Functions (Originally by Angelo Canty for S)
class                   Functions for Classification
cluster                 "Finding Groups in Data": Cluster Analysis Extended
                        Rousseeuw et al.
codetools               Code Analysis Tools for R
compiler                The R Compiler Package
datasets                The R Datasets Package
foreign                 Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS',
                        'Stata', 'Systat', 'Weka', 'dBase', ...
graphics                The R Graphics Package
grDevices               The R Graphics Devices and Support for Colours and Fonts
grid                    The Grid Graphics Package
KernSmooth              Functions for Kernel Smoothing Supporting Wand & Jones
                        (1995)
lattice                 Trellis Graphics for R
MASS                    Support Functions and Datasets for Venables and Ripley's
                        MASS
Matrix                  Sparse and Dense Matrix Classes and Methods
methods                 Formal Methods and Classes
mgcv                    Mixed GAM Computation Vehicle with Automatic Smoothness
                        Estimation
nlme                    Linear and Nonlinear Mixed Effects Models
nnet                    Feed-Forward Neural Networks and Multinomial Log-Linear
                        Models
parallel                Support for Parallel Computation in R
rpart                   Recursive Partitioning and Regression Trees
spatial                 Functions for Kriging and Point Pattern Analysis
splines                 Regression Spline Functions and Classes
stats                   The R Stats Package
stats4                  Statistical Functions using S4 Classes
survival                Survival Analysis
tcltk                   Tcl/Tk Interface
tools                   Tools for Package Development
utils                   The R Utils Package
```

To install packages. Eg: XML

Installs from CRAN (Comprehensive R Archive Network)

```
install.packages("XML")
```

Shows built-in datasets

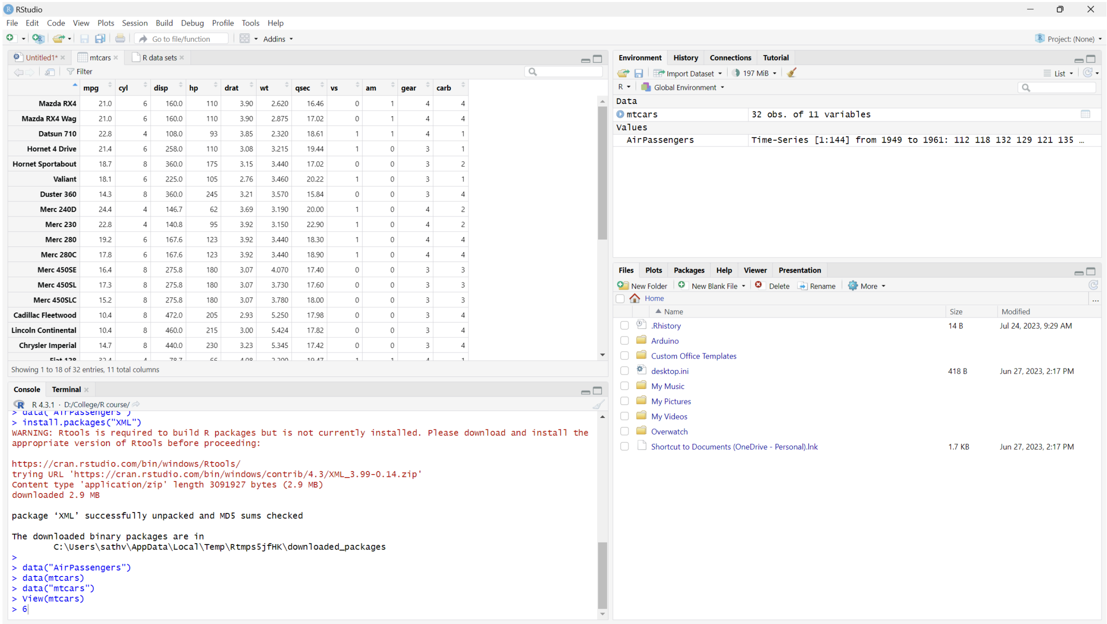
```
data()
```

If we want to use a particular data set .Eg:

```
data("AirPassengers")
```

We have 2 types of datasets

1. Tabular - on clicking, data shown in tabular format. eg:mtcars
2. Non-tabular - displayed with time series. eg: AirPassengers



	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4

Showing 1 to 18 of 32 entries, 11 total columns

```
> data( AirPassengers )
> install.packages("XML")
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/
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/XML_3.99-0.14.zip'
Content type 'application/zip' length 3091927 bytes (2.9 MB)
downloaded 2.9 MB

package ‘XML’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\sathv\AppData\Local\Temp\Rtmps5jfHK\downloaded_packages
>
> data("AirPassengers")
> data(mtcars)
> data("mtcars")
> View(mtcars)
> 6
```

```
Data
mtcars             32 obs. of 11 variables
Values
  AirPassengers    Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 …
```

Name	Size	Modified
.Rhistory	14 B	Jul 24, 2023, 9:29 AM
Arduino		
Custom Office Templates		
desktop.ini	418 B	Jun 27, 2023, 2:17 PM
My Music		
My Pictures		
My Videos		
Overwatch		
Shortcut to Documents (OneDrive - Personal).lnk	1.7 KB	Jun 27, 2023, 2:17 PM

- mtcars has a proper data set and format and hence it is shown in tabular form when clicked on
- AirPassesngers doesn’t open when clicked on

▼ Assignment operators

- operators used for assigning the values

left assign operator←

```
a<-10
```

right assign operator→

```
a→10
```

equal to operator= gives error

- variable naming is the same as c language

▼ Print statement

- for printing a line (string)

```
print("R programming")
```

- for printing values

```
a=20
b=10
c=a-b
print(c)
#print("c=", c) #string with variable name does not work, print displays
```

- for printing values along with string

```
a=10
b=10
c=a+b
paste("c=",c)
```

- it can all be done in one line

```
a=10
b=10
c=a+b
print(paste("c=",c))
```

- Removes the space

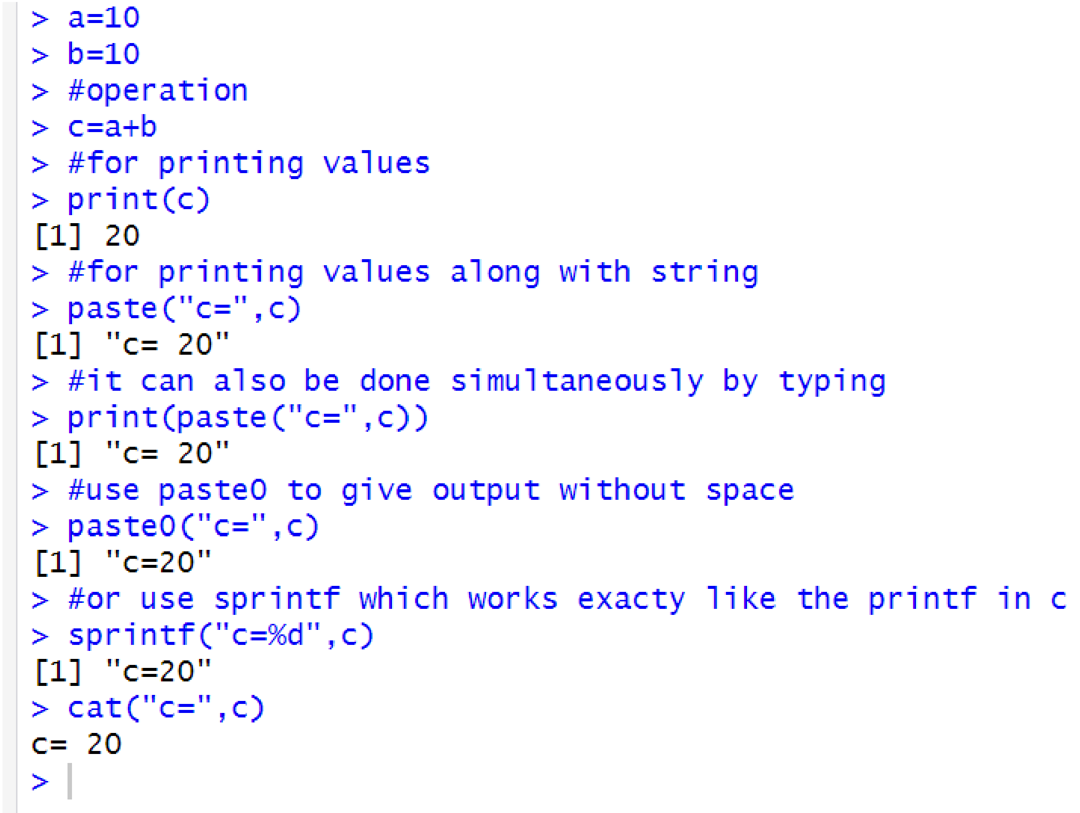
```
paste0("c=", c)
```

- printf from c
  prints with %d, %f, etc

```
sprintf("c=%d",c)
```

- Concatenation of string and value

```
cat("c=",c)
```



```
> a=10
> b=10
> #operation
> c=a+b
> #for printing values
> print(c)
[1] 20
> #for printing values along with string
> paste("c=",c)
[1] "c= 20"
> #it can also be done simultaneously by typing
> print(paste("c=",c))
[1] "c= 20"
> #use paste0 to give output without space
> paste0("c=",c)
[1] "c=20"
> #or use sprintf which works exacty like the printf in c
> sprintf("c=%d",c)
[1] "c=20"
> cat("c=",c)
c= 20
> 
```

Data types

class(x) -displays the data type of the value of x

1. Numeric

```
x ← 10.5 #decimal value is numeric
class(x)
```

2. Integer

```
x ← 10L #L represents integer value
class(x)
```

3. Logical / Boolean

```
x←TRUE
class(x)
```

4. RAW Data Type

Specifies values as raw bytes

A given value can be converted into a raw data type

- give a string and convert to raw data type

```
r=charToRaw("Hello")
class(r)
```

- to convert raw back to character

```
r=charToRaw("Hello")
#now the char is raw
r=rawToChar(r)
#now it is char
class(r)
```

Converting to Integer

```
as.integer("4") #4
as.integer("1.6") #1
as.integer("-3.2") #-3
as.integer("0x400") #1024
```

Converting to Character

```
as.character(1)
as.character(2 + 3)
as.character(1.5)
```

Reading input

Default: String

- For integer, you have to convert after

```
readline("Enter your name:")
```

- After execution, user can give input
  Without variable name, its not getting stored

```
a = readline("Enter your name:")
```

- With variable, its stored in environment

For getting multiple inputs use curly braces{}

```
{
name=readline("Enter your name:")
SRN=readline("Enter your SRN:")
}
```

Input from text file

- create a text file in the same directory and then use the following code to read from the txt file

```
readLines("textfile.txt")
#don't forget to add file extension
#L capital
```

- create a text file in a different directory and then use the following code to read from the txt file

```
readLines("path")
```

- can be assigned to a variable

```
readtext=readLines("textfile.txt")
```

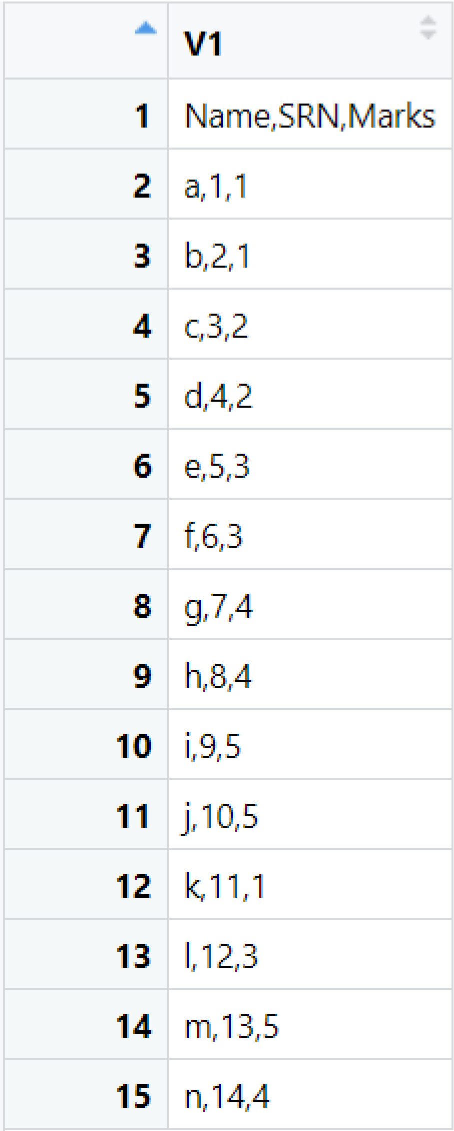
- Copies text from one file and stores it in another
- Also creates a new text file in the same directory

```
file=readLines("textfile.txt") #stored in variable file
writeLines(file,"textfile2.txt") #Creates a new file Demo2 with text of De
```

Reading from a CSV file

- To read a csv file

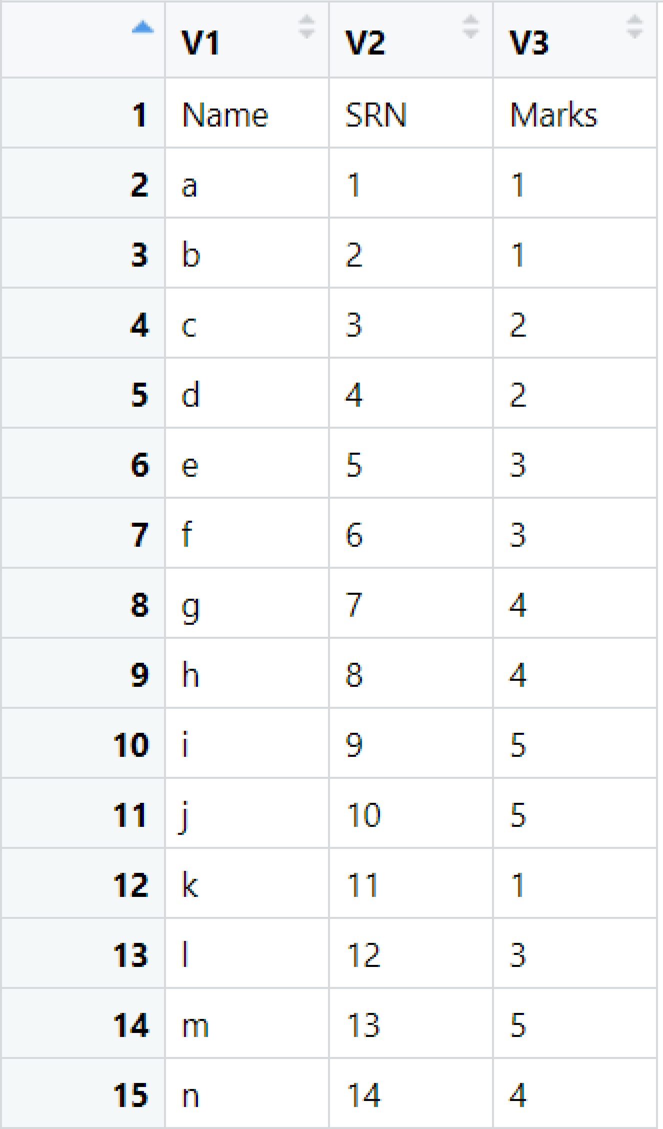
```
z=read.table("csv.csv")
```



	V1
1	Name,SRN,Marks
2	a,1,1
3	b,2,1
4	c,3,2
5	d,4,2
6	e,5,3
7	f,6,3
8	g,7,4
9	h,8,4
10	i,9,5
11	j,10,5
12	k,11,1
13	l,12,3
14	m,13,5
15	n,14,4

- to read and differentiate between the rows and columns we add ***separator***

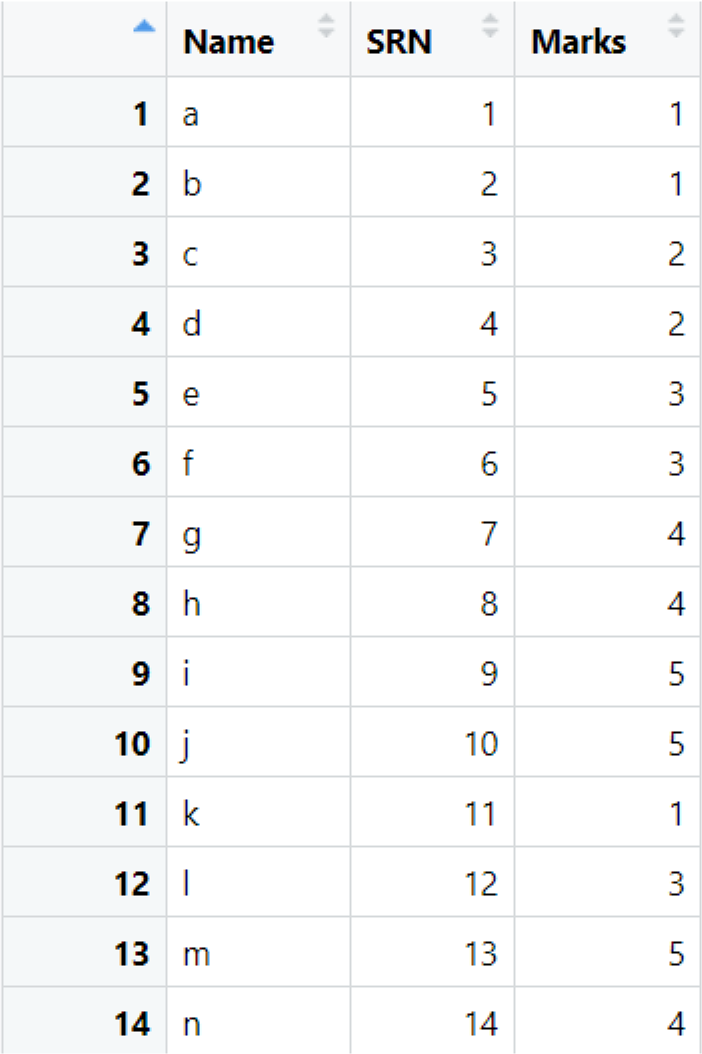
```
z=read.table("csv.csv", sep=",")
```



	V1	V2	V3
1	Name	SRN	Marks
2	a	1	1
3	b	2	1
4	c	3	2
5	d	4	2
6	e	5	3
7	f	6	3
8	g	7	4
9	h	8	4
10	i	9	5
11	j	10	5
12	k	11	1
13	l	12	3
14	m	13	5
15	n	14	4

- To make the first row a table header

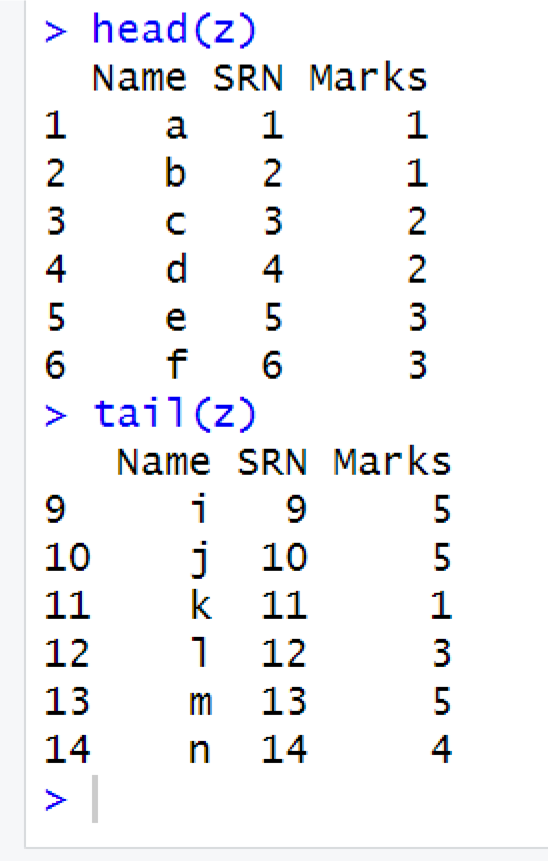
```
z=read.table("CSV1.csv", sep = ",", header = TRUE )
```



	Name	SRN	Marks
1	a	1	1
2	b	2	1
3	c	3	2
4	d	4	2
5	e	5	3
6	f	6	3
7	g	7	4
8	h	8	4
9	i	9	5
10	j	10	5
11	k	11	1
12	l	12	3
13	m	13	5
14	n	14	4

- head(z)-first n rows of the dataset is printed
- tail(z)- last n rows of the data set is printed

```
z=read.table("csv.csv", sep=",", header = TRUE)
head(z)
tail(z)
```



```
> head(z)
  Name SRN Marks
1    a   1     1
2    b   2     1
3    c   3     2
4    d   4     2
5    e   5     3
6    f   6     3
> tail(z)
   Name SRN Marks
9     i   9     5
10    j  10     5
11    k  11     1
12    l  12     3
13    m  13     5
14    n  14     4
> 
```

Built-in functions

Math Functions

```
max(5,10,15)
min(5, 10, 15)
sqrt(16)
abs(-4.7) #4.7
ceiling (1.4)
floor(1.4)
```

Letters

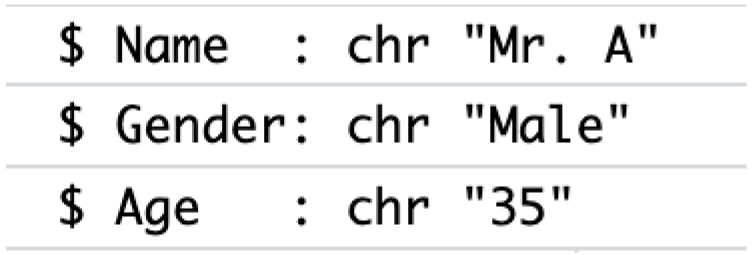
```
LETTERS[10] #Only gives 10th letter
LETTERS[1:10] #Prints the letters in range
LETTERS[1:30]
```

Month

```
month.name #Gives full name
month.abb #Gives abbreviation
```

dput() and dget()

```
x ← data.frame(Name="Mr. A", Gender = "Male", Age = "35")
dput(x)
```



```
$ Name  : chr "Mr. A"
$ Gender: chr "Male"
$ Age   : chr "35"
```

```
x ← data.frame(Name="Mr. A", Gender = "Male", Age = "35")
dput(x)
dput(x,file = "/Users/mac/Desktop/College/R/dput.r")
y ←dget("/Users/mac/Desktop/College/R/dput.r")
```

- dput() - Stores the dataframe values in a structure/object
  dput(path) - Creates the file and stores it
- dget() - Converts a dput file back

dump()

- dump() - Store data of any data type, in whatever way you've given it as

```
x ← 100
d ← data.frame(Name="Mr. A", Gender = "Male", Age = "35")
dump(c("x","d"),file="/Users/mac/Desktop/College/R/dump.r")
```

source()

include external code into your current R session.

```
source("/Users/mac/Desktop/College/R/dump.r")
x
d
str(d)
```

loads an external R script "dump.r," accesses the variables "x" and "d" defined in that script, and then prints information about the structure of the "d" variable.

Assignment 1

1. Create a dataset. Fetch and display the data.
2. Tell user to give, name, SRN, 3 subject Marks
3. Display and save total marks into a different file

```
#Part 1: Create a dataset. Fetch and display the data.
forestfire=read.table("forestfires.csv")
forestfire=read.table("forestfires.csv", sep = ",", header = TRUE )

#Part 2: Tell user to give, name, SRN, 3 subject Marks. Display and save to
{
name=readline("Enter your name:")
SRN=readline("Enter your SRN:")
Sub1=readline("Enter Subject 1 marks:")
Sub2=readline("Enter Subject 2 marks:")
Sub3=readline("Enter Subject 3 marks:")}

total = as.integer(Sub1)+as.integer(Sub2)+as.integer(Sub3)
l=as.character(total)
writeLines(l,"sum.txt")

x <- data.frame(Name = name, SRN = SRN, Subject1 = Sub1, Subject2 = Su
dput(x,file = "/Users/mac/Desktop/College/R/details.r")
y ←dget("/Users/mac/Desktop/College/R/details.r")
y
```

▼ Day 2

▼ Bind

Row Bind

```
df=data.frame(name="PES", age="50")
df
df=data.frame(name=c("a","b","c"), age=c(11,12,13))
df1=data.frame(name=c("d","e"), age=c(11,12))
df3=rbind(df,df1) #Row Bind, common column names(wont be repeated)
df4=cbind(df,df1) #Error, column bind requires same number of rows
```

Column Bind

```
df=data.frame(name="PES", age="50")
df
df=data.frame(name=c("a","b","c"), age=c(11,12,13))
df2=data.frame(name=c("f","g","h"), age=c(10,11,12))
df4=cbind(df,df2) #Column Bind
```

Length

Gives no of columns

```
length(df)
```

▼ Operators

Arithmetic Operators

```
a <-2
b <-3

print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

```
a <-2
b <-3
print(a%%b) #Modulus
print(a%/%b) #Integer division, without decimal value
print(a^b)
```

Miscellaneous Operators

```
a1<-8
a2<-12
t ← 1:10 #Colon opperator, gives range
print(a1%in%t) #Checks if its in the list/vector
print(a2%in%t)
```

Data Structures

Vectors

c used to create vector

Print or Remove

```
a=c(2,3,4,5,6) #c used to create vector
b=c("a","b","c","d")

a[1]
a[c(1,3)] #1st and 3rd value printed
```

```
a[c(1:3)] #1st to 3rd value printed

a[c(-1)] #1st element removed
#a[c(-n)] nth element removed
```

Replace

```
b=c("a","b","c","d")
b[1]← "pear"
b
```

Built-in sorting

```
a=c(2,3,4,5,6)
sort(a)
```

Print pattern

```
a=c(1,2,3)

#1 1 1 2 2 2 3 3 3
r=rep(a)
r=rep(c(1,2,3),each=4)
r

#1 2 3 1 2 3 1 2 3
r=rep(a,3)

#1 1 1 1 2 3 3
r=rep(c(1,2,3), times=c(4,1,2))
r

#1234234234
```

```
r=rep(c(2,3,4),3)
append(r,1,after=0)
```

Append(list,value,poition)

```
append(r,1,after=0)
```

▼ List

```
d ← list("apple","banana","cherry")
d

append(d,"orange") #Appends to the end, does not store it in environm
append(d,"orange", after = 2) #Added at 3rd value
```

▼ Arrays

Creating Array

```
a ← c(1:5)
a
```

Pattern Printing

To create two dimensional array,
td=array(data,dim,dimname)

[1] 1 2

```
a ← c(1:5)
td=array(a, dim = 2)
td
```

[,1] [,2]
[1,] 1 4

```
[2,] 2 5
[3,] 3 1
```

```
a ← c(1:5)
td=array(a, dim = c(3,2))
td
```

Matrices

Creating Matrix

1. Column-wise (default)

   ```
   [,1] [,2] [,3]
   [1,] 1 4 1
   [2,] 2 5 2
   [3,] 3 6 3
   ```

```
m=matrix(c(1,2,3,4,5,6),nrow = 3, ncol = 3)
m
```

2. Row-wise

   ```
   [,1] [,2] [,3]
   [1,] 1 2 3
   [2,] 4 5 6
   [3,] 1 2 3
   ```

```
m=matrix(c(1,2,3,4,5,6),nrow = 3, ncol = 3, byrow = TRUE)
m
```

Display

```
m=matrix(c(1,2,3,4,5,6),nrow=3,ncol=3,byrow = TRUE)

#to display a particular row
```

```
m[2,]
#to display a particular column
m[,2]
#to display a particular number
m[1,3]
```

byrow can be used only in matrices

▼ Conditional Statements

▼ If

```
a ← 33
b ← 200
if(b>a){
   print("b is greater than a")
}
```

If else

```
a ← 330
b ← 200
if(b>a){
   print("b is greater than a")
}else{
   print("a is greater than b")
}
```

else if & else

```
a<-33
b<-200
if(b>a){
  print("b is greater than a")
}else if(a>b){
  print("a is greater than b")
}else{
  print("a is equal to b")
}
```

Example: Largest of 3 numbers

```
#greatest of three numbers with user input
{
a=readline("enter 1st number:")
b=readline("enter 2nd number:")
c=readline("enter 3rd nummber:")
if((a>b)&&(b>c)){
  print("b is greatest number")
}else if((a<b)&&(c<b)){
  print("b is greatest number")
}else{
  print("a is greatest number ")
}}
```

Switch

Within Switch case

Parameter 1 = condition, 2nd parameter = cases

```
a= "12"
#Parameter 1 = condition, 2nd parameter = cases
y=switch(a,
         "9" = "hello a",
         "12" = "hello b",
         "18" = "hello c",
         "21" = "hello d"
)
y
```

Example 3

```
{
x=readline("1 or 2?:")
y=readline("1 or 2?:")
xy=switch(
  paste(x,y,sep=""),
  "11"="hello xy=11",
  "12"="hello xy=12",
  "21"="hello xy=21",
  "22"="hello xy=22"
)
xy
}
```

Assignment 2

Design simple calculator using switch case.
Number, operator, operand should be from user
Print full answer

```
{
a=readline("enter a number:")
b=readline("enter another number:")
d=readline("enter an operation:")
calculation=switch(paste(d),
                   "+"="a+b",
                   "-"="a-b",
                   "*"="a*b",
                   "/"="a/b"
)
calculation
if(a+b){
  d=a+b
  print(d)
}else if(a-b){
  d=a-b
  print(d)
}else if(a*b){
  d=a*b
  print(d)
}else{
  d=a/b
  print(d)
}
}
```

```
{
A=readline("enter a number:")
a=as.integer(A)
B=readline("enter another number:")
b=as.integer(B)
d=readline("enter an operation:")
calculation=switch(paste(d),
                   "+"="addition of a and b",
                   "-"="subtraction of a and b",
                   "*"="multiplication of a and b",
                   "/"="division of a and b"
)
calculation
if(calculation=+){
  e=a+b
  sprintf("%s=%d",calculation,e)
}else if(calculation=-){
  e=a-b
  sprintf("%s=%d",calculation,e)
}else if(calculation=*){
  e=a*b
  sprintf("%s=%d",calculation,e)
}else if(calcultaion=/){
  e=a/b
  sprintf("%s=%d",calculation,e)
}
}
```

Day 3

Data Visualisation

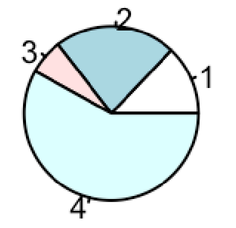
Pie

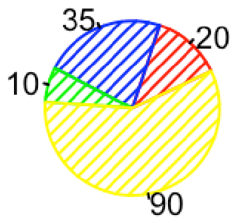
ESA Question: Make a pie chart with minimum 6 attributes

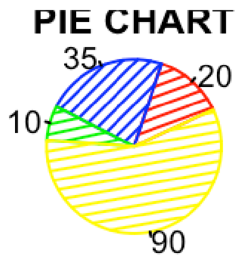
```
a=c(20,35,10,90)

pie(a)
pie(a, labels=a)
pie(a, labels=c("A","B","C","D"))
pie(a, labels=a, col=c("red","blue","green","yellow"))
pie(a, labels=a, col=c("red","blue","green","yellow"), init.angle = 25, der
pie(a, labels=a, col=c("red","blue","green","yellow"), init.angle = 25, der
#init angle is to change the initial angle
#angle is to change the angles of the shading lines(Density)
pie(a, labels=a, col=c("red","blue","green","yellow"), init.angle = 25, der
#changed the border color
pie(a, labels=a, col=c("red","blue","green","yellow"), init.angle = 25, der

png(filename="hi1.png")#to create a png file in the current working dire
pie(a, labels=a, col=c("red","blue","green","yellow"), init.angle = 25, der
#main gives a title to the plot
#lty changes line type
```

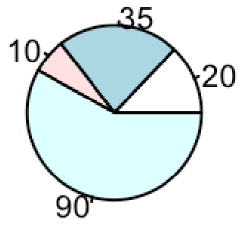
Attribute/command	Notes
pie (a)	Creates pie chart
labels = a	Uses values from a as labels
col=c("red","blue","green","yellow")	Assigns each colour to a value
init.angle = 25	To change the initial angle

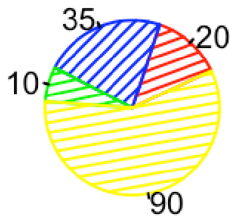
density = 29	The density of the shading lines
angle=c(20,45,12,10)	To change the angles of the shading lines(Density)
border="red"	Changes border colour
lty=2	Line type
main="PIE CHART”	Title of the plot
png(filename="hi1.png")	Creates a png file in the current working directory to store the pie chart. Needs to be before pie function
dev.off()	To let the system know that we are done with the plotting and now we can save it in the png file we created

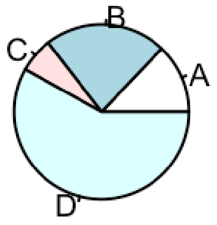


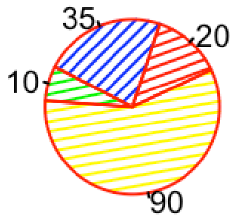


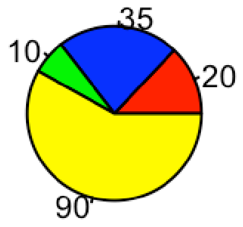


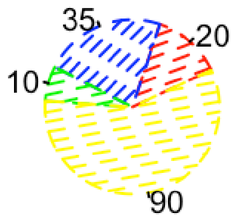






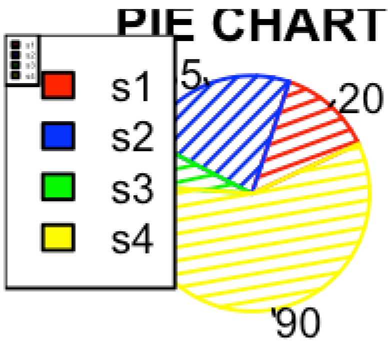






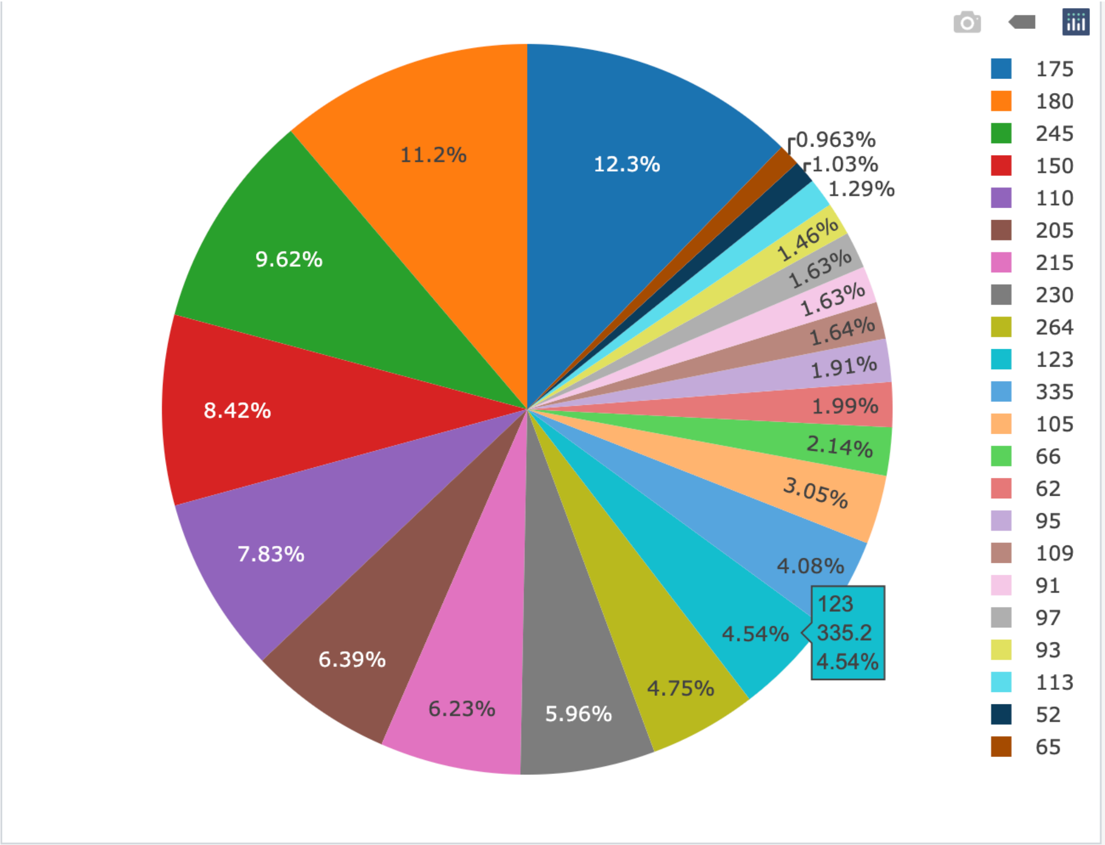
```
b=c("s1","s2","s3","s4")
legend("topleft", b, fill=c("red","blue","green","yellow"))#b should alway
legend("topleft", b, fill=c("red","blue","green","yellow"), cex=0.2)#cex is
```

legend()	Creates a legend
legend(”topleft”)	Location of legend
fill=c("red","blue","green","yellow")	Colours of the boxes of the legend
cex=0.2	To change the size



Using plotly

```
fig <- plot_ly  (mtcars, labels= ~hp, values = ~disp, type = "pie")
fig
```

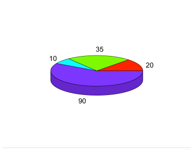


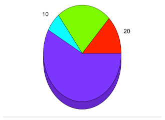
▼ 3D Pie

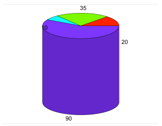
- Library to be installed: plotrix

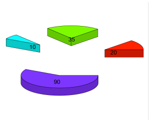
```
install.packages("plotrix")
library(plotrix)

a=c(20,35,10,90)
pie3D(a, labels=a)
#initial angle for pie3d doesn't work
pie3D(a, labels=a, theta=2)#changes the orientation
pie3D(a, labels=a, radius=1, height=1)
pie3D(a, labels=a, explode=1)#seperates into segments
pie3D(a, labels=a, height=0.5, radius =1, explode=1, theta=0.5)
```

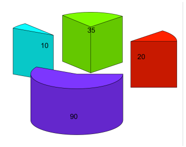
Attribute	**Note**
pie3D(a)	
labels=a	Uses values from a as labels
theta=2	Changes the orientation. Initial angle for pie3d doesn't work
radius=1, height=1	
explode=1	Seperates into segments





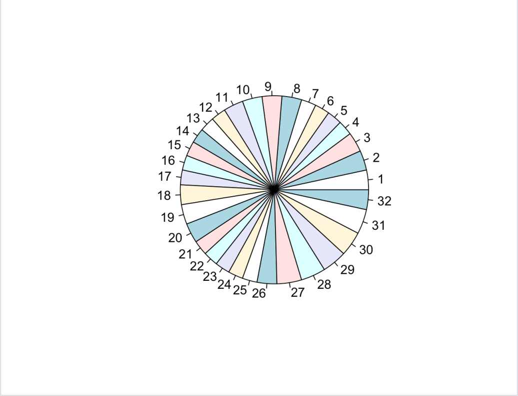


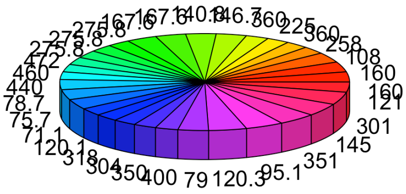




Plotting a dataset

```
pie(mtcars$gear)
pie3D(mtcars$gear, labels=mtcars$disp)
```



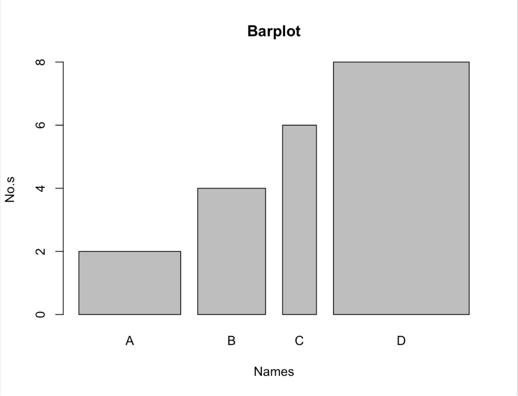


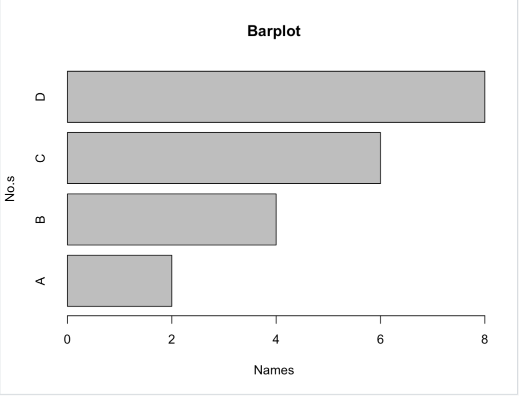
▼ Bar Plot

```
#BARPLOT

x←c(2,4,6,8)
y←c("A","B","C","D")

barplot(x)
barplot(x, names.arg=y, main="Barplot", xlab="Names", ylab="No.s", w
#names.arg gives the names to the bars
#main gives the title
#xlab displays the name of the x parameter and y lab of y
#width sets the width of the bars
barplot(x, names.arg=y, main="Barplot", xlab="Names", ylab="No.s", h
#horiz is to change the bar plot into a horizontal one
```

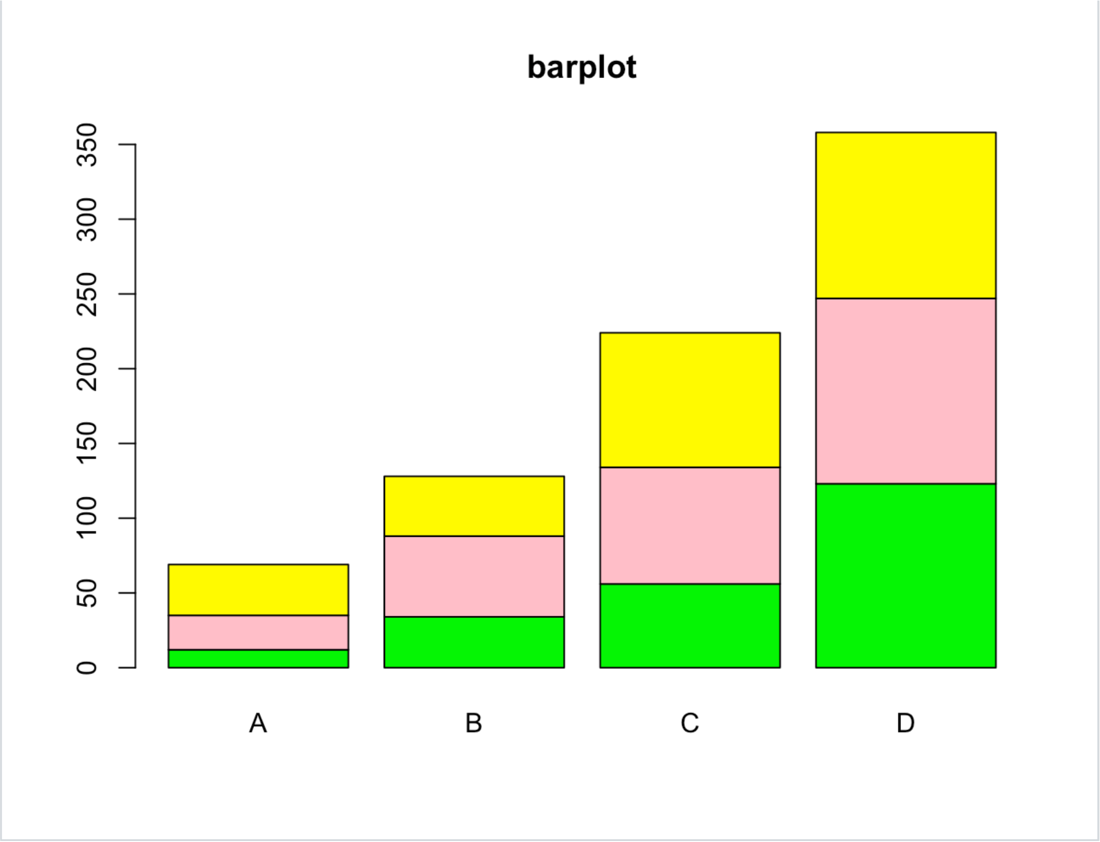
Attributes	
barplot (x)	
names.arg=y	gives the names to the bars
main="Barplot"	gives the title
col=c("green","pink","yellow")	

xlab="Names", ylab="No.s"	xlab displays the name of the x parameter and y lab of y
width=c(3,2,1,4)	sets the width of the bars
horiz=TRUE	To change the bar plot into a horizontal one
as.matrix(y1)	to plot a stacked bar plot we need to enter the values in matrix format





```
#to plot a stacked bar plot we need to enter the values in matrix format
y1=matrix(c(12,23,34,34,54,40,56,78,90,123,124,111), ncol=4,nrow=3)

barplot(as.matrix(y1),col=c("green","pink","yellow"), names.arg=y,main
```



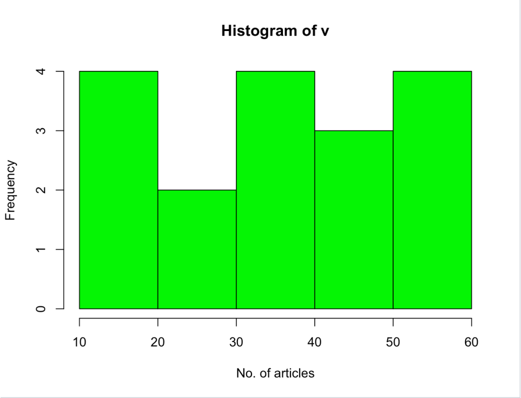
▼ Histogram

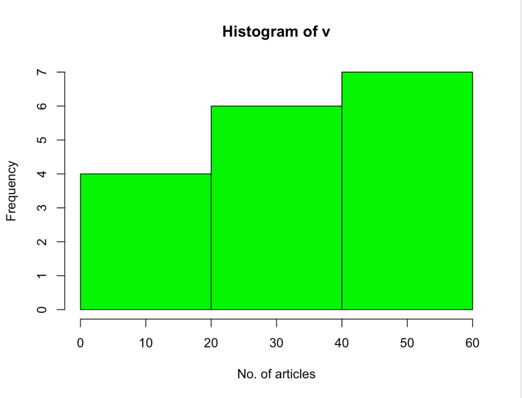
```
#HISTOGRAM

v←c(19,34,12,56,19,21,32,23,34,56,56,34,58,18,45,45,46)
hist(v,xlab="No. of articles", col="green", border="black")
hist(v,xlab="No. of articles", col="green", border="black", breaks=2)
#breaks the number of times you've given
```

Attributes	Notes

xlab="No. of articles"
ylab=“”
	x and y axis labels

col="green"	
border="black"	
breaks=2	Breaks the number of times you've given





Line Graph

```
#LINE GRAPH

v←c(13,22,28,7,31)
w←c(11,13,32,6,35)
x←c(12,22,15,34,35)

plot(v)#gives a scatter plot
plot(v, type="l")#gives the line graph
plot(v,type="o")#combination of scatter and line plots
plot(v,type="b",lty=2)#run to see how it looks
plot(v,type="o", lwd=2)#lwd sets the width of the line
plot(v,type="p")#same as default plot v
plot(v,type="s")#stair steps like (vertical lines are the ranges. eg, 13-22
plot(v,type="S")#stair steps like (horizontal lines are the ranges. eg, 13-
plot(v,type="h")#histogram type
plot(v,type="n")#blank
plot(v,type="s",lty=2,lwd=2)

plot(c(v,w,x),type="l")#it will plot all three in the same line
{
plot(v,type="l")
lines(w,type="s",lty=2)
lines(x,type="p")
}#to plot the three values in three diff lines of diff types in the same gra

plot(v,type="l",xlab="a",ylab="b",ylim=c(1,30),xlim=c(1,30))
#ylim and xlim sets the limit range of y and x axis in the graph
```

Attribute	Note
plot(v)	Scatter plot
type="l"	line plot
type="o"	Combination of line & scatter
type="b"	Combination of line & scatter with breaks
type="p"	scatter plot, same as plot v
type="s"	stairs type, with ranges as vertical lines
type="S"	stairs type, with ranges as horizontal lines
type="h"	histogram
type="n"	blank
lty=2	line type
lwd=2	line width
xlab & ylab	x and y axis labels

xlim=c(1,30),
ylim=c(1,30)
	limit range for x and y axis in the graph

Loops

for loop

```
week←c('Mon','Tue','Wed','Thurs','Fri','Sat','Sun')
for(val in week)
{
  print(val)
}
```

For lists

- keyword: seq_along
- for accessing the list, double box brackets [[i]]

```
# Create a list of numbers
my_list ← list(1, 2, 3, 4, 5)

# Loop through the list and print each element
for (i in seq_along(my_list)) {
  current_element ← my_list[[i]]
  print(paste("The current element is:", current_element))
}
```

For matrix

- Keyword: seq_len

```
# Create a 3×3 matrix of integers
my_matrix ← matrix(1:9, nrow = 3)

# Loop through the matrix and print each element
for (i in seq_len(nrow(my_matrix))) {
  for (j in seq_len(ncol(my_matrix))) {
    current_element ← my_matrix[i, j]
    print(paste("The current element is:", current_element))
  }
}
```

▼ while loop

```
i ← 1while (i < 6) {
    print(i)
    i ← i + 1
}
```

▼ Functions

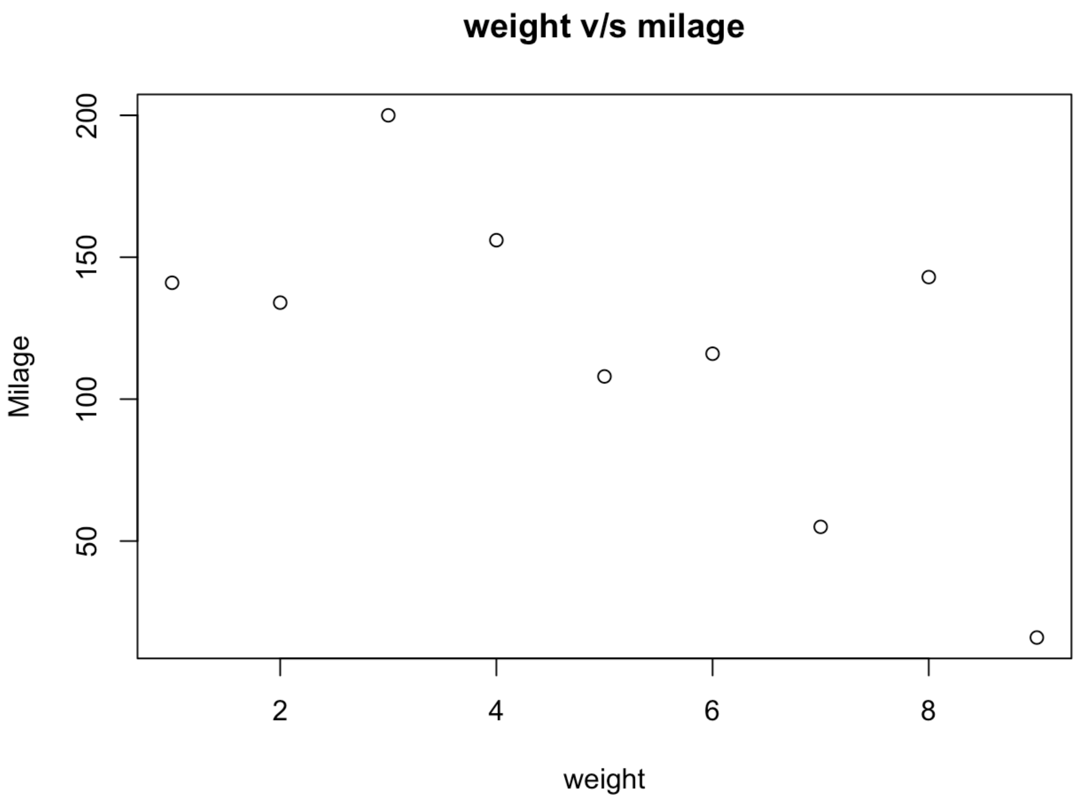
```
fname←function(){
  print("hello")
}
fname()
```

```
myfunc←function(f){
  paste(f,"college")
}
myfunc("PES")
```

Day 4

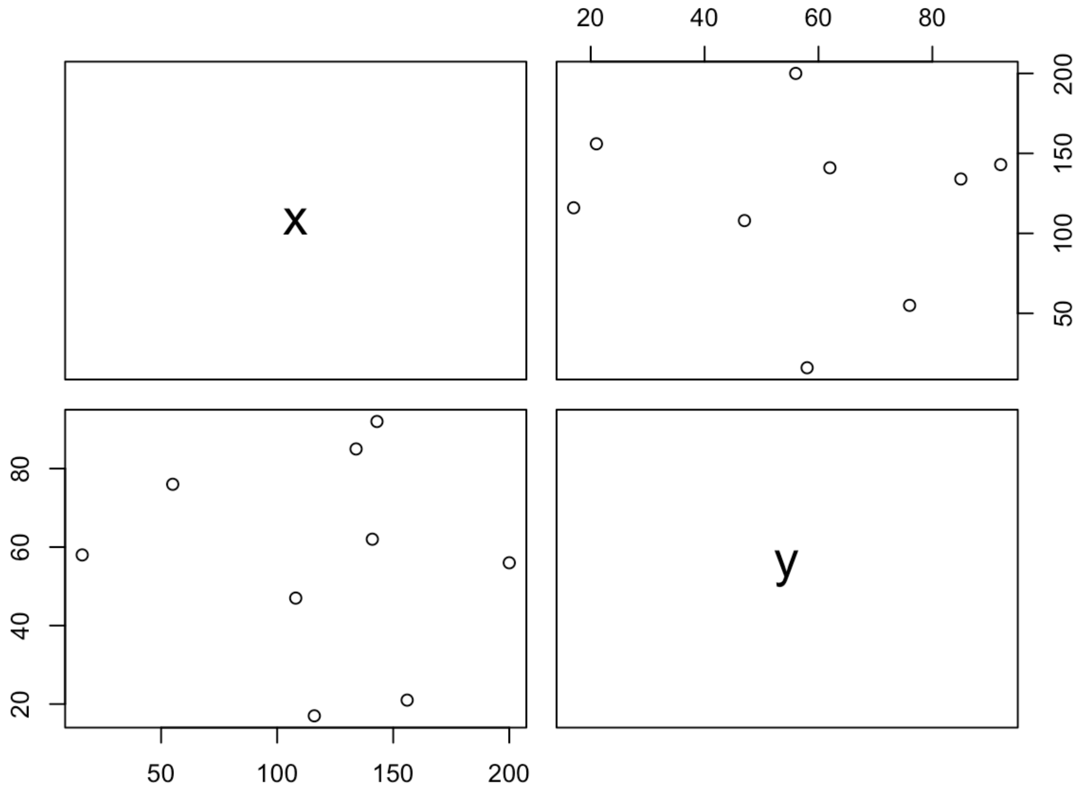
Scatter Plots

```
#Scatter Plots
x ← c(141, 134, 200, 156, 108, 116, 55, 143, 16)
y ← c(62,85,56, 21, 47, 17,76, 92, 58)

plot(x = x, xlab="weight", ylab = "Milage", main = "weight v/s milage")
```



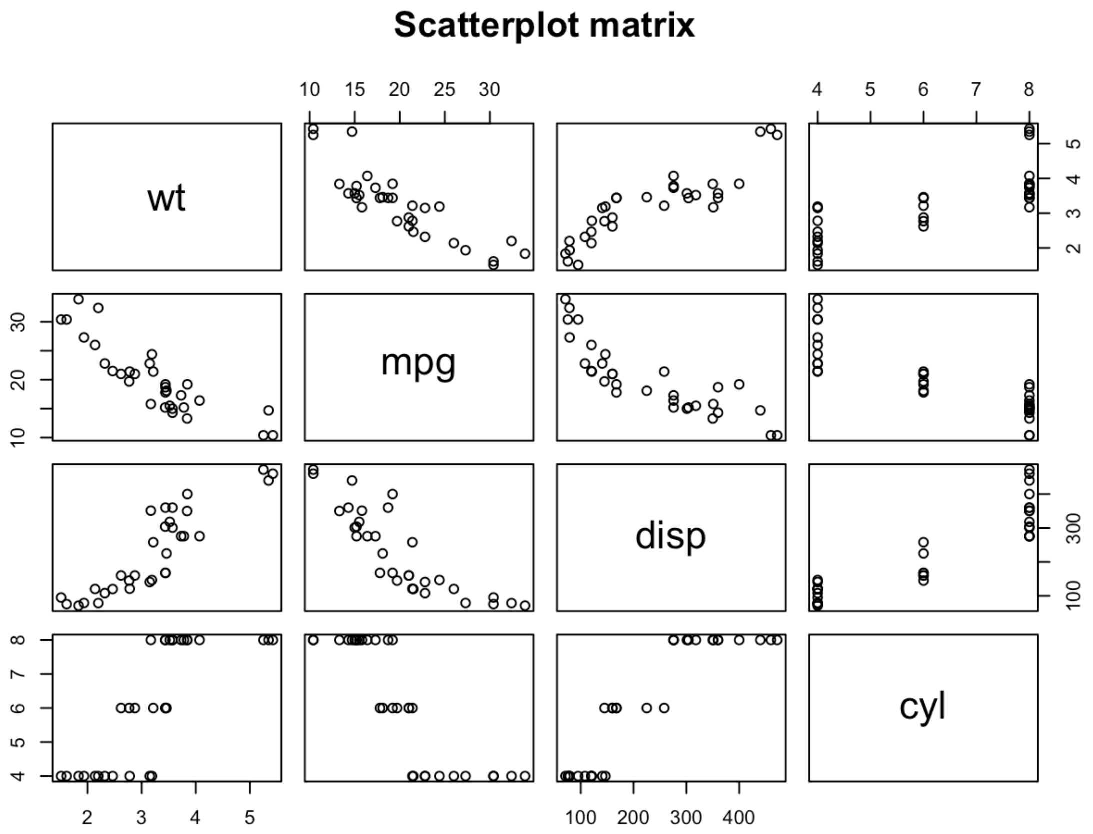
Scatterplot Matrix

```
#Scatter Plots
x ← c(141, 134, 200, 156, 108, 116, 55, 143, 16)
y ← c(62, 85, 56, 21, 47, 17, 76, 92, 58)

plot(x = x, xlab="weight", ylab = "Milage", main = "weight v/s milage")
pairs(~x+y)
#No of values (length) of x and y datasets should be equal
```



Each column gives x-axis and every row gives y-axis

```
#Scatter plots with pairs
pairs(~x+y)
pairs(~wt +mpg +disp +cyl, data=mtcars, main = "Scatterplot matrix")
```



▼ Graphical Plot

▼ ggplot

- Library to be installed: ggplot2

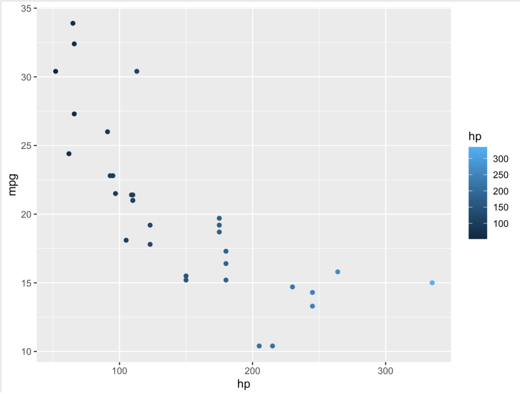
```
install.packages("ggplot2")
library(ggplot2)
ggplot()
ggplot(data = mtcars, aes(x = hp, y=mpg, col= hp))+geom_point() #col
labs(title = "Miles per Gallon vs Horsepower", x= "Horsepower", y="Gal



ggplot(data = mtcars, aes(x = hp, y=mpg))+geom_area()
#aes = aesthetic
```

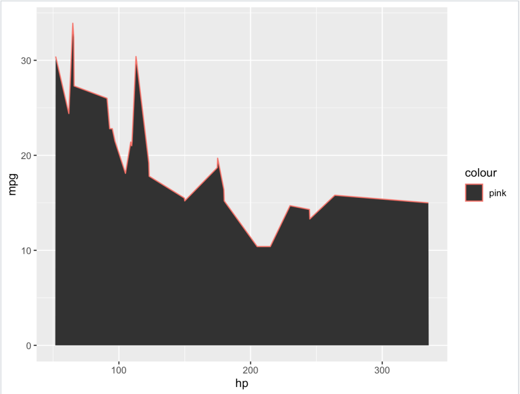
Attributes	
(data = mtcars)	

(aes(x = hp, y=mpg, col= hp))	aes= aesthetic
(+geom_point())	
labs()	

Note: For ggplot, for colour its col (within aes)

in plotly, its color



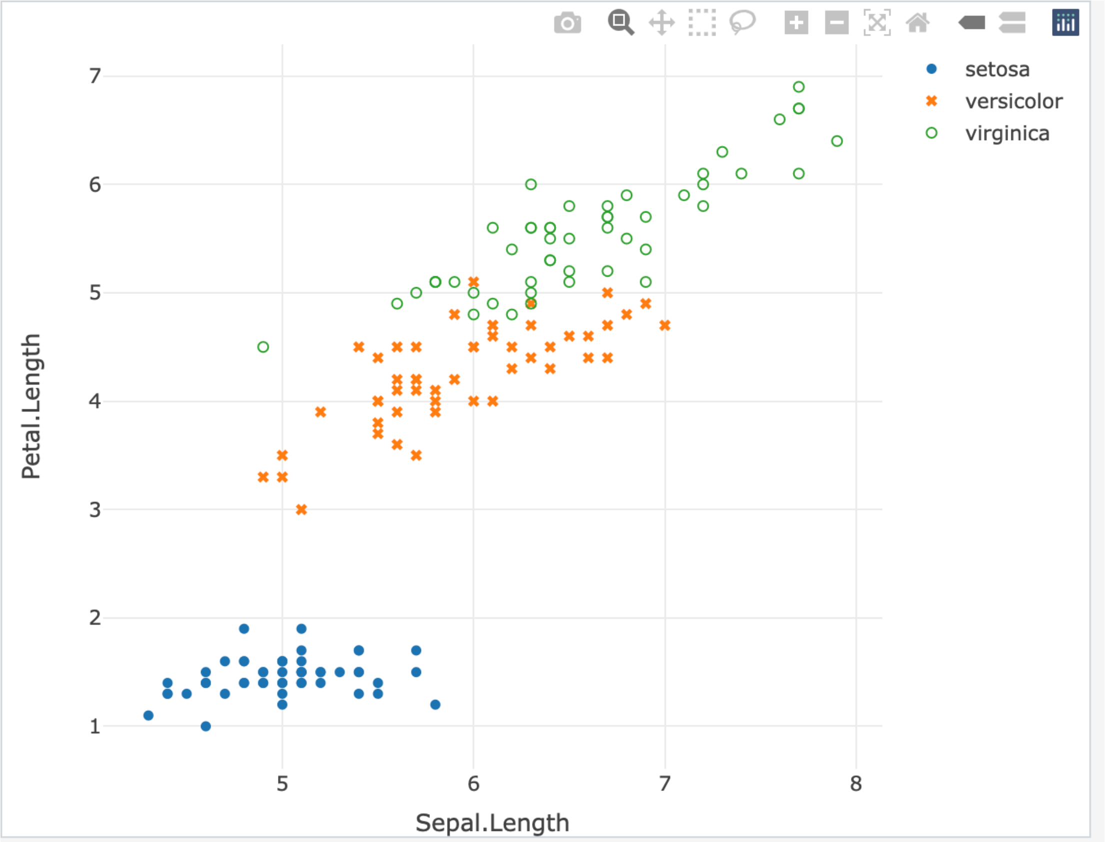


▼ plotly

- Library to be installed: plotly

```
install.packages("plotly")
library(plotly)
fig ← plot_ly (data = mtcars, x = ~disp, y = ~hp)
#For column name always give ~ before
fig
```

```
data("iris")
fig ← plot_ly(data = iris, x= ~Sepal.Length, y = ~Petal.Length, symbol=
fig
#symbol selects the column for which we're working on
#symbols is used for assigning symbols (circle, cross, hollow circle)
```

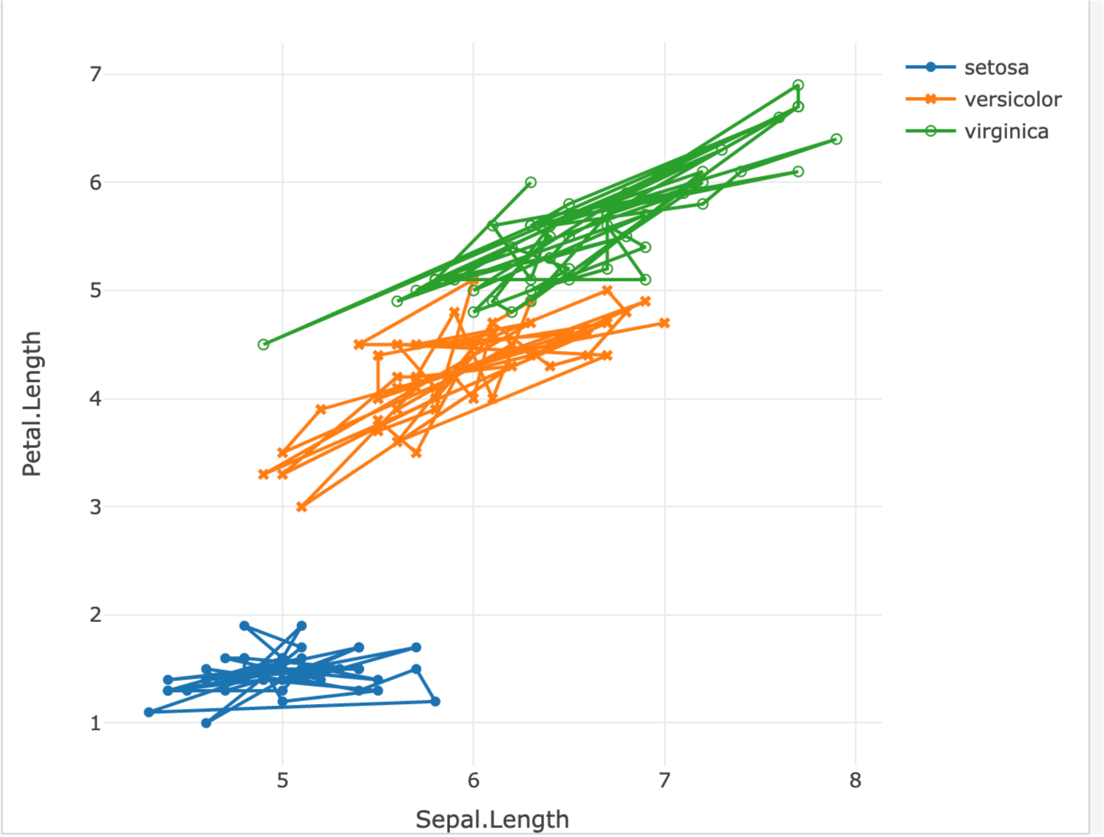


Attribute	
data = mtcars	
x=~hp, y=~disp	
symbol = ~	

- Trace - Helps us in plotting properly

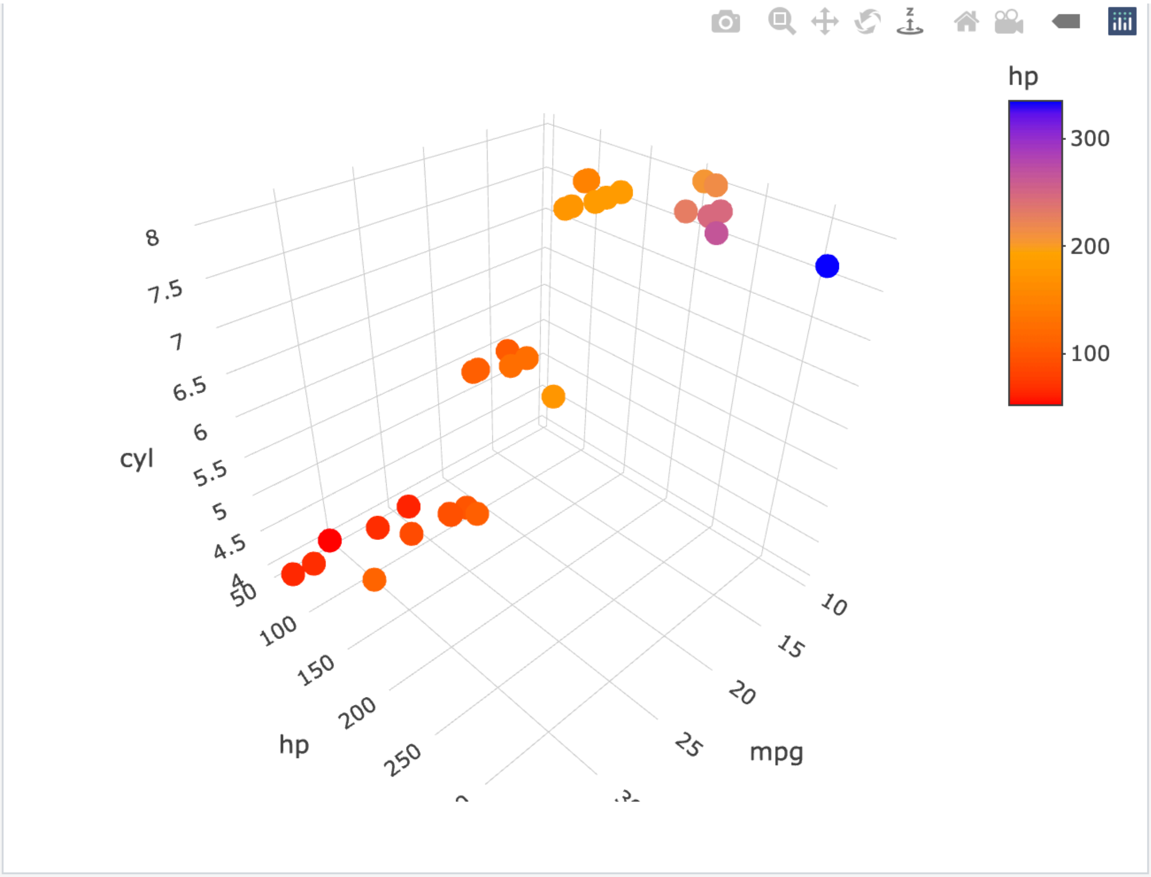
```
add_trace(fig, type="scatter", mode="markers+lines")
```

Attributes	Notes
add_trace(x)	
type= "scatter"	
mode = "markers+lines"	



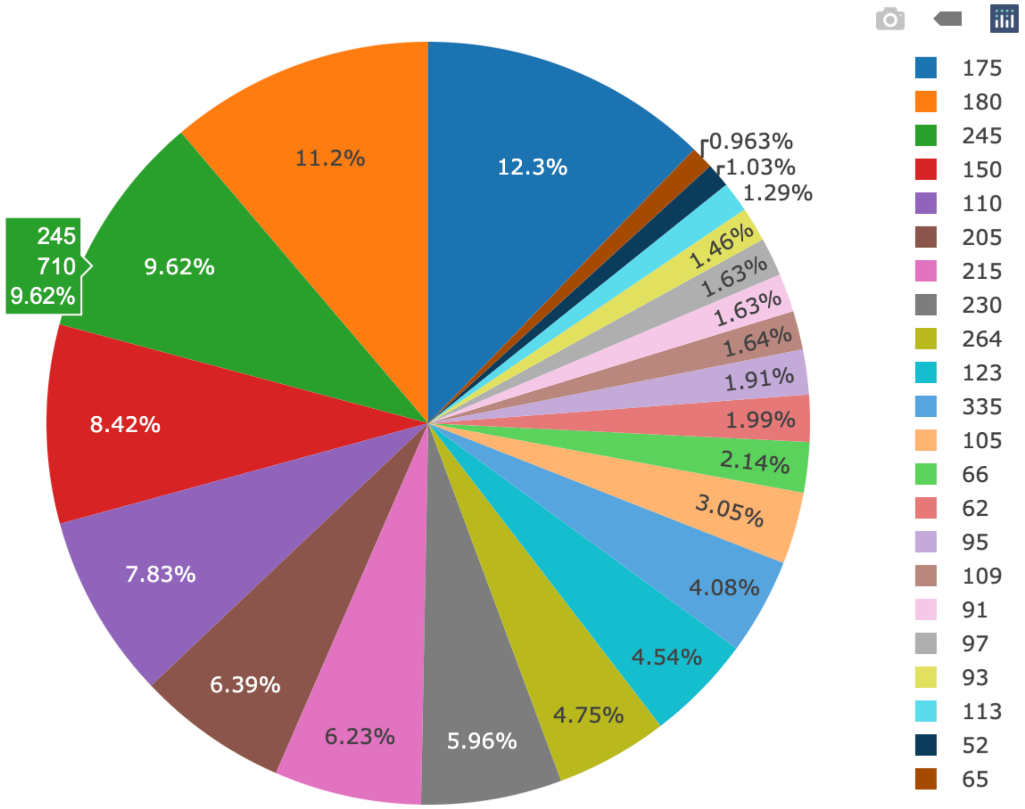
▼ 3D Scatter Plot

```
attach(mtcars)
fig ← plot_ly (data = mtcars, x = ~mpg, y = ~hp, z=~cyl, color=~hp,
                    colors = c("red","orange","blue"))
#For column name always give ~ before
#If you dont give hp, it wont give multiple colours. hp gives shade bar
fig
```



Plotly for pie chart

```
fig ← plot_ly  (mtcars, labels= ~hp, values = ~disp, type = "pie")
fig
```



Plotly for barplot

```
max.temp = c(22,27,33,26,24,18,20)
barplot(max.temp, main = "Max temps in a week", )
```

▼ Dataset

```
?mtcars #Shows all deatils, for detailed study of the dataset
summary(mtcars)
```

```
names(mtcars)[1:10]
names(mtcars[1:10])

rownames(mtcars)
rownames(mtcars)[which.max(mtcars$hp)]
rownames(mtcars)[which.min(mtcars$hp)]
```

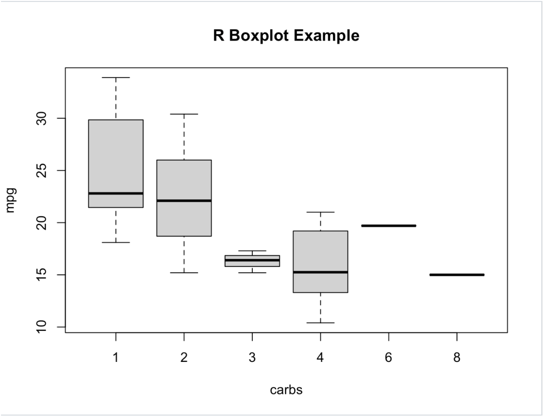
```
Data_Cars ← mtcars

dim(Data_Cars) #Dimension of dataset

mean(Data_Cars$wt)
median(Data_Cars$wt)

sort(table(mtcars$wt))
#with minus the order is reversed
sort(-table(mtcars$wt)) #Sorts according to frequency
names(sort(-table(mtcars$wt))) #Makes it into names (strings)
```

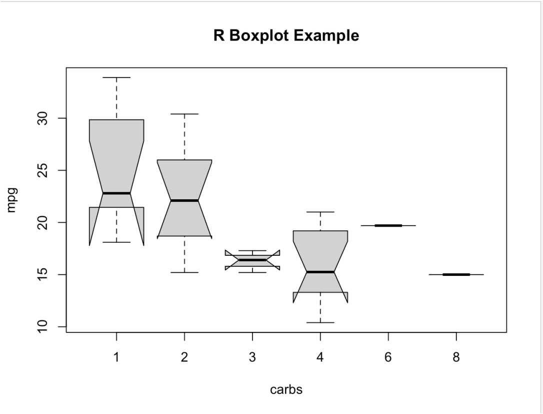
Day 5

Box Plot

```
boxplot(mpg ~ carb, data = mtcars, xlab = "carbs", ylab = "mpg",
            main = "R Boxplot Example")
```

```
boxplot(mpg ~ carb, data = mtcars, xlab = "carbs", ylab = "mpg",
            notch = TRUE, #Narrowing of box at median
            main = "R Boxplot Example")
```

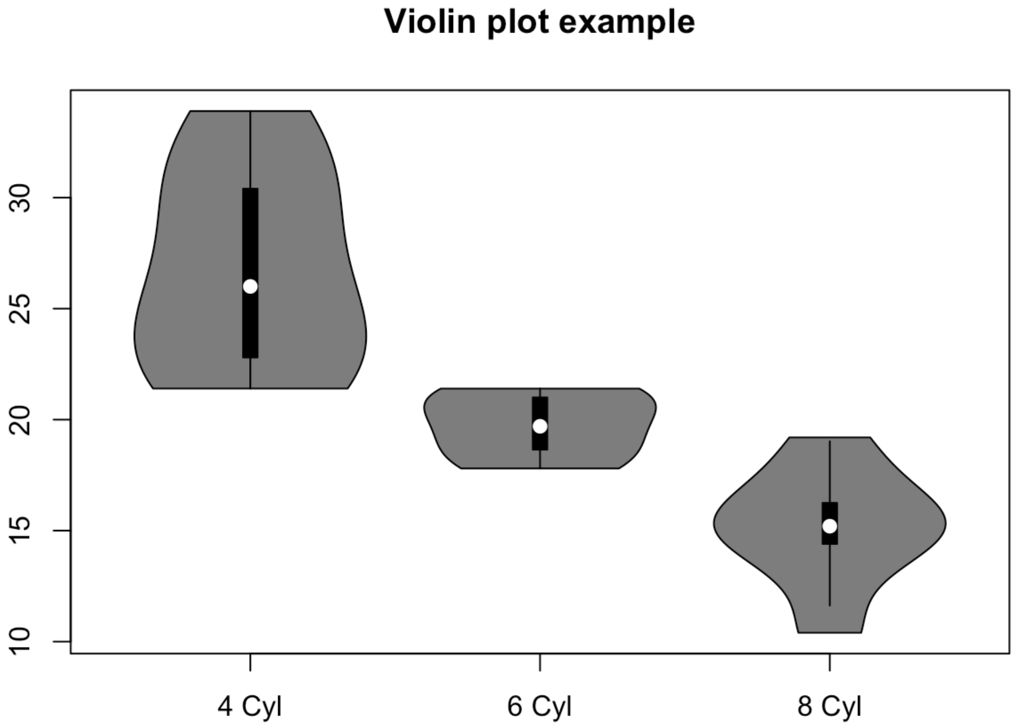




Attribute	Notes
boxplot(mpg ~ carb)	The 2 variables at x and y axis that are used for plotting
data = mtcars	
xlab, ylab	labels at x and y axis
notch = TRUE	Narrowing of box at median
main	title of boxplot

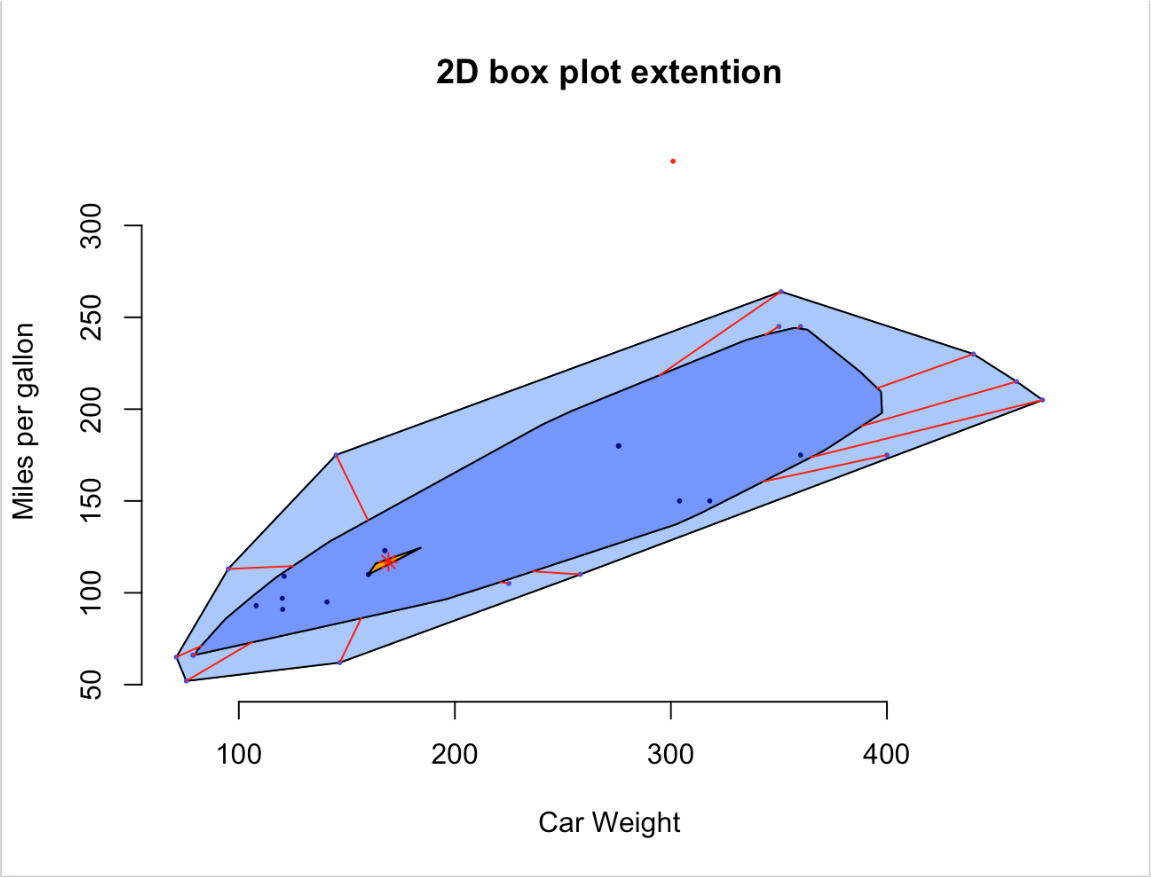
Violin Plot

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

x1 = mtcars$mpg[mtcars$cyl==4]
x2 = mtcars$mpg[mtcars$cyl==6]
x3 = mtcars$mpg[mtcars$cyl==8]
vioplot(x1,x2,x3, names = c("4 Cyl","6 Cyl","8 Cyl"))
title("Violin plot example")
```



▼ Bag plot

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

attach(mtcars)
bagplot(disp,hp, xlab = "Car Weight", ylab = "Miles per gallon",
            main = "2D box plot extention")
```


2D box plot extention
Miles per gallon
50
100
150
200
250
300
Car Weight
100
200
300
400