PROGRAMMING WITH R

R Programming – A beginning

- R is an open source programming language and software environment for statistical computing and graphics.

- The R language is widely used among statisticians and data miners for developing statistical software and data analytics tools

History of R

- R project was created by Robert Gentleman and Ross Ihaka, Department of Statistics, University of Auckland (1995).
- Modelled after S & S-plus, developed at AT&T labs in late 1980s.
- R ranks 11th in the TIOBE index, a measure of programming language popularity.

History of R

- The official R software environment is a free software environment within the GNU package, available under the GNU General Public License.
- It is written primarily in C, Fortran, and R itself (partially self-hosting).
- Multiple third-party graphical user interfaces are also available, such as RStudio, an integrated development environment, and Jupyter, a notebook interface.

STATISTICAL SOFTWARE'S

ADaMSoft	JMP	OpenEpi	Salstat	Statgraphics
Analyse-it	LIMDEP	Orange	SAS	Statistica
ASReml	Maple	Origin	SciPy	StatPlus
BMDP	MATLAB	OxMetrics	SHAZAM	Statsmodels
DataMelt	Mathematica	Primer	SigmaXL	SYSTAT
Dataplot	MaxStat Pro	Python	Skytree Infinity	TSP
ELKI	MedCalc	R	SOCR	UNISTAT
Epi Info	Minitab	R Studio	SOFA Statistics	Unscrambler
EViews	NCSS	RATS	SPlus	Winpepi
GAUSS	NLOGIT	RKWard	SPSS	WPS
GraphPad Prism	NMath Stats	ROOT	Stata	WINKS
GenStat	NumXL	SageMath	StatCrunch	XploRe
Excel				

Why R is different from others?

SPSS	SAS
Minitab	MATLAB
Statistica	R
.....	.....
Non-Programming	Programming

DATA SCIENTIST REQUIREMENTS

- R Programming
- Python Coding
- Hadoop Platform
- Apache Spark

R VERSIONS

- R-4.1.2 (Bird Hippie)
  - Released on 2021-11-01.
  - 86 megabytes, 32/64 bit

- R-4.0.5 (Shake and throw)
  - released on 2021-03-31.

Use Current Version

Download and Install R

Installing R on windows PC:

- Click on download option (R 4.1.2 for windows).
  - *Cran.R on Google*
- Save this to the folder C:\R on your PC.
- When downloading is complete, close or minimize the Internet browser.
- Double click on .exe in C:\R to install.

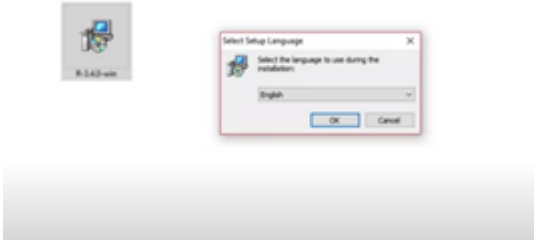
Installing R on Linux:

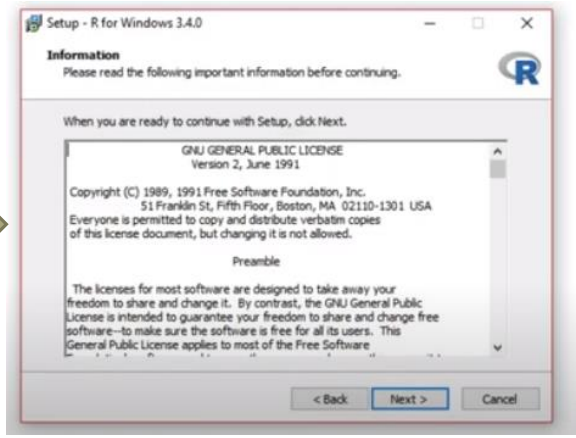
- sudo apt-get install r-base-core

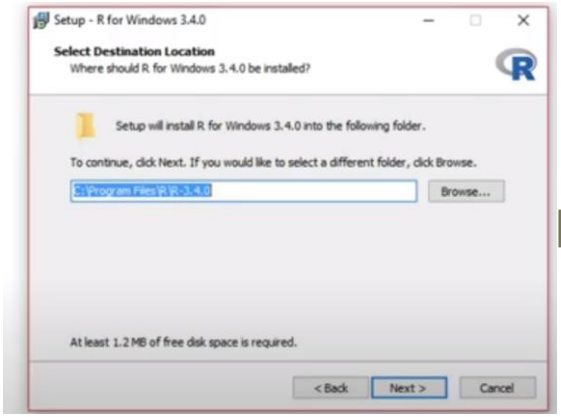
Download R:

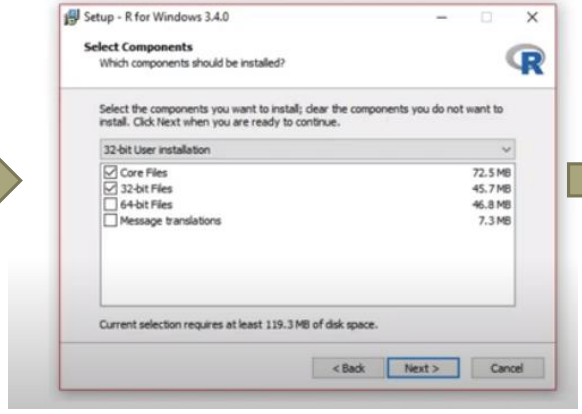
https://cran.r-project.org/bin/windows/base/

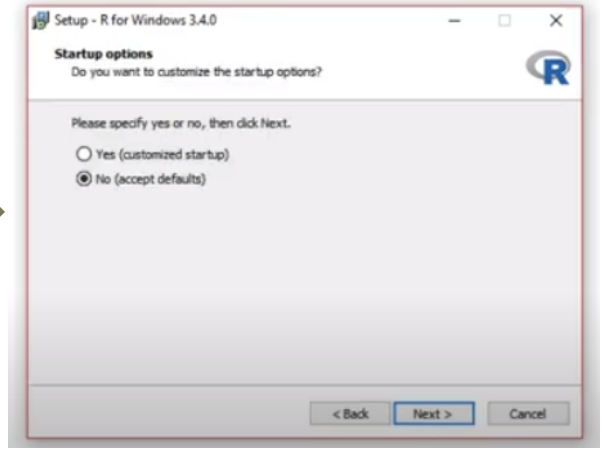
INSTALLATION STEPS



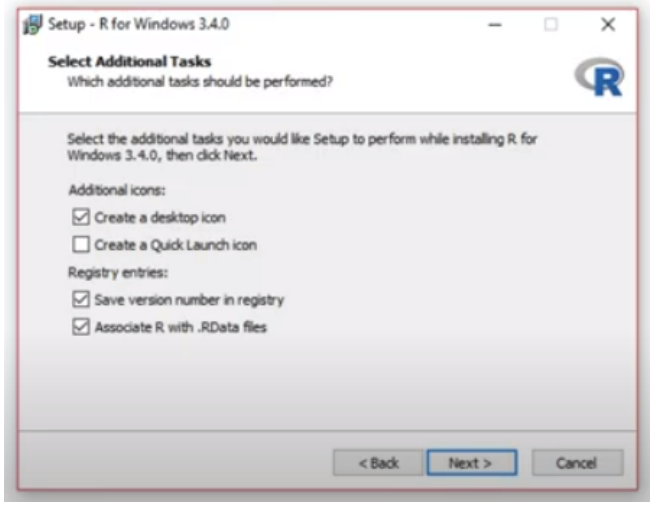


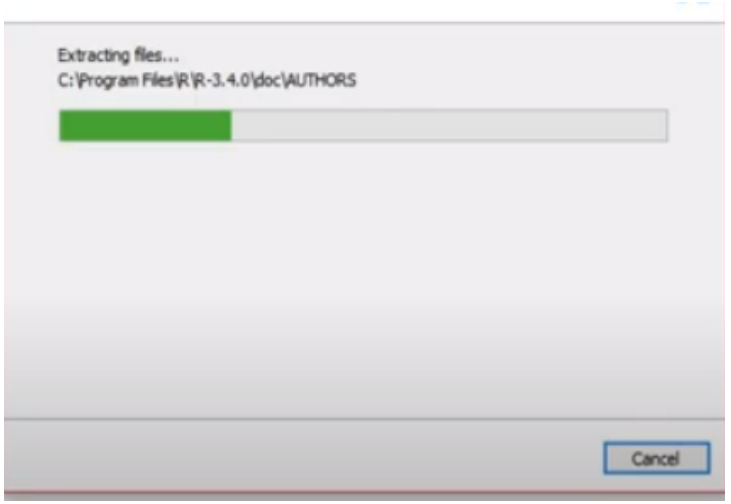






INSTALLATION STEPS CONTD....





RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

INSTALLING R STUDIO:

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

Download RStudio:

http://www.rstudio.com/ide/download/desktop

VERSION

Get R version

R. Version()

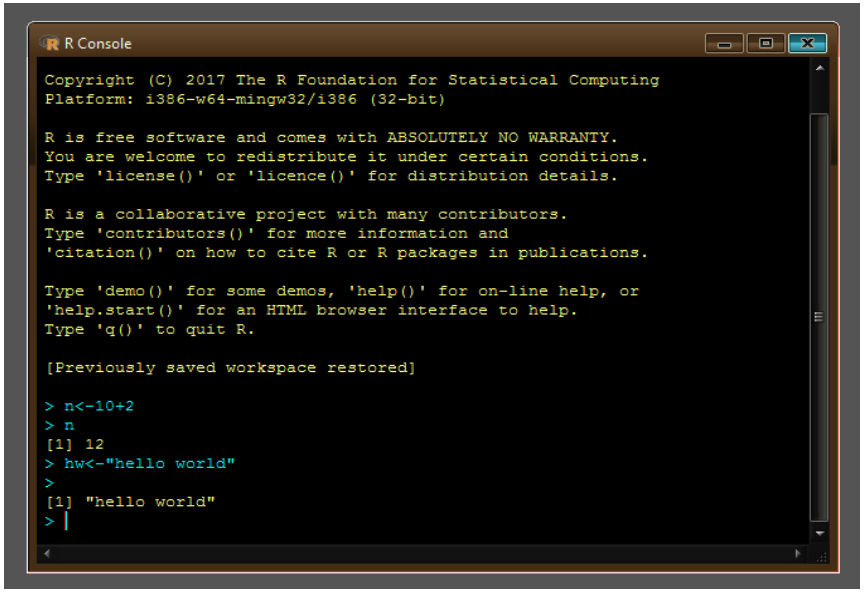
Get R Studio version

R Studio: Toolbar at top > Help > About RStudio

A TEST RUN WITH R IN WINDOWS

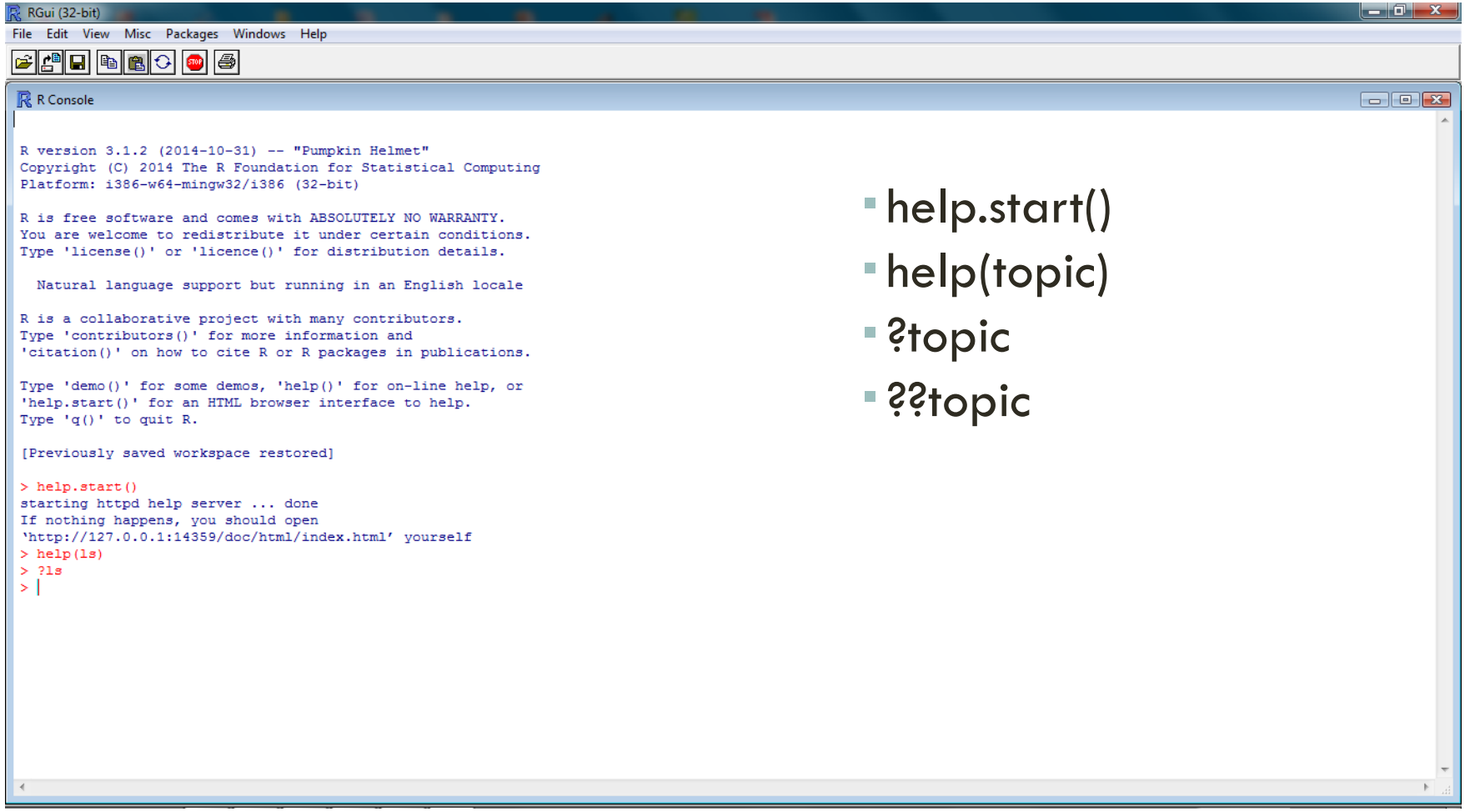
Double click the R icon on the Desktop and the R Console will open.

Wait while the program loads. You observe something like this!!!!!!!



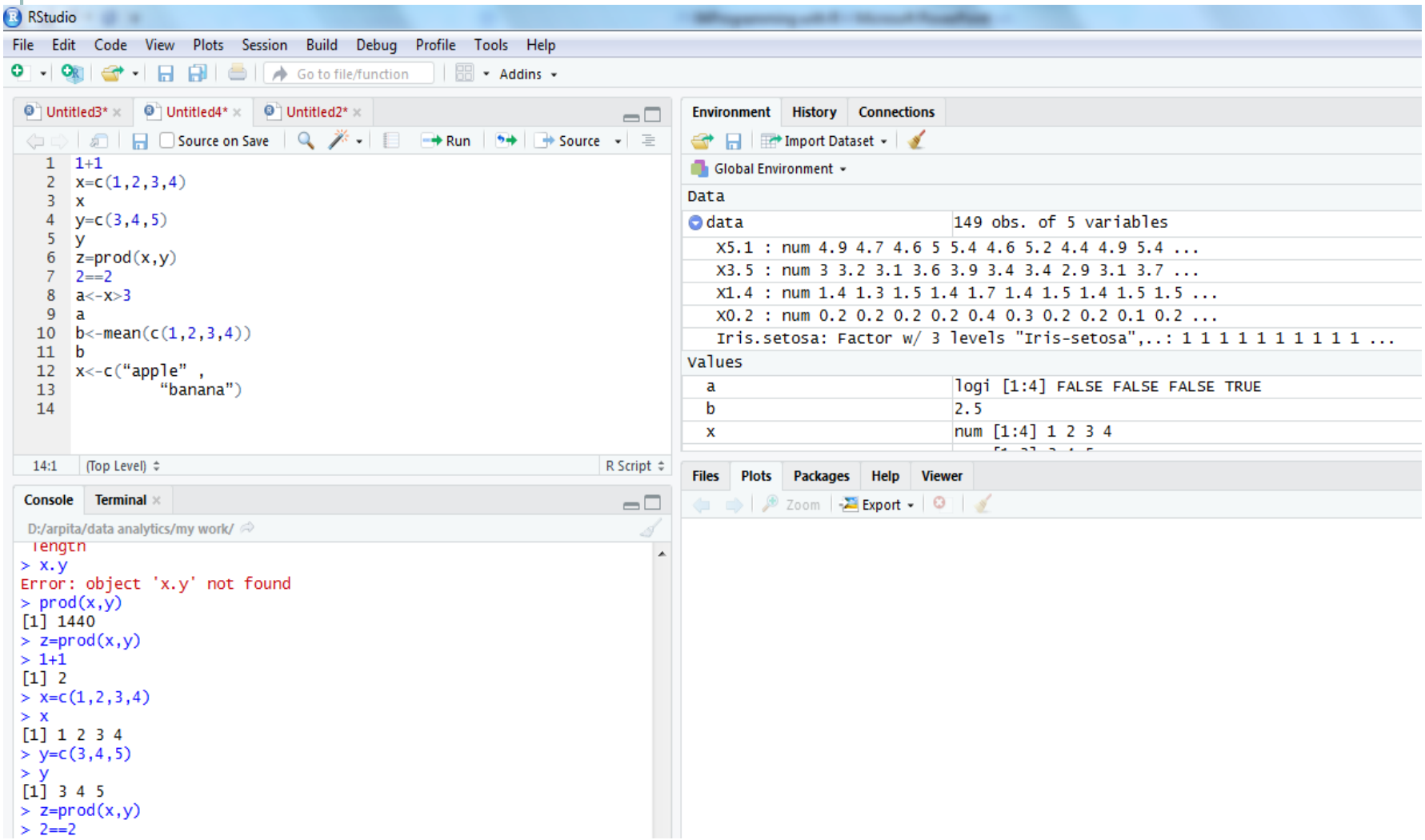
- You can type your own program at the prompt line >.

GETTING HELP FROM R CONSOLE



- help.start()
- help(topic)
- ?topic
- ??topic

R COMMAND IN INTEGRATED ENVIRONMENT



THINGS TO REMEMBER.....

- Everything in R is case sensitive
- Comments #
- The data types available in R are known as *modes*
  - numeric (integers and real)
  - factor (characters)
  - logical (True or False)

DATA INPUT

- Numeric Variables
- Character Variables
- From Spread Sheet
- Existing Data
- Cloud Space

R AS A CALCULATOR

```
> 3+5          # Addition
> 2-1          # Subtraction
> 2*3          # Multiplication
> 4/2          # Division
> 2^3          # Power
```

Storing of values to Variables

```
> x=1
> x<-12
> 12->x
> 14->x
```

```
>y=1
> 1=y
```

Error in 1 = y : invalid (do set) left-hand side to assignment

HOW TO USE R FOR SIMPLE MATHS

```
> 3+5
> 12 + 3 / 4 – 5 + 3*8
> (12 + 3 / 4 – 5) + 3*8
> pi * 2^3 – sqrt(4)
>factorial(4)
>log(2,10)
>log(2, base=10)
>log10(2)
>log(2)
```

Note

R ignores spaces

HOW TO STORE RESULTS OF CALCULATIONS FOR FUTURE USE

```
>  x = 3+5

> x

>  y = 12 + 3 / 4 – 5 + 3*8

> y

> z =  (12 + 3 / 4 – 5) + 3*8

> z

> A <- 6 + 8      ## no space should be between < & -

> a               ## Note: R is case sensitive

>A
```

FAMILIARIZE SOME FUNCTIONS IN R

seq

rep

letters/LETTERS

scan

c

C FUNCTION AND SEQUENCE FUNCTION

1 c function

c(1,2,4)

x= c(1,2,4)

x

2 Sequence function

seq(from = 1, to = 10, by =2)

S = seq(from = 1, to = 10, by =2)

s

seq(1,10,1) .....> 1:10

USING C COMMAND

```
> data1 = c(3, 6, 9, 12, 78, 34, 5, 7, 7)   ## numerical data
> data1.text = c('Mon', 'Tue', "Wed")  ## Text data
                                   ## Single or double quote both ok
                          ##copy/paste into R console may not work
> data1.text = c(data1.text, 'Thu', 'Fri')
```

SCAN COMMAND FOR MAKING DATA

```
> data = scan()    ## data separated by Space / Press
                   ## Press Enter key twice to exit
1: 4 5 7 8
5: 2 9 4
8: 3
9:
```

Console

```
> data                          ## Read 8 items
[1] 4 5 7 8 2 9 4 3
```

SCAN COMMAND FOR MAKING DATA

> d3 = scan(what = 'character')

1: mon

2: tue

3: wed thu

5:

> d3

[1] "mon" "tue" "wed" "thu"

> d3[2]

[1] "tue"

> d3[2]='mon'

> d3

[1] "mon" "mon" "wed" "thu"

- > d3[6]='sat'
-
- > d3
- [1] "mon" "mon" "wed" "thu" NA "sat"
-
- > d3[2]='tue'
-
- > d3[5] = 'fri'
-
- > d3
- [1] "mon" "tue" "wed" "thu" "fri" "sat"

IDENTIFIERS NAMING

Don't use underscores (_) or hyphens (-) in identifiers.

The preferred form for variable names is all lower case letters and words separated with dots (variable.name) but variableName is also accepted.

Examples:

avg.clicks	GOOD
avgClicks	OK
avg_Clicks	BAD

Function names have initial capital letters and no dots (e.g., FunctionName).

CONCEPT OF WORKING DIRECTORY

```
>getwd()
[1] "C:\Users\DSamanta\R\Database"

> setwd('D:\Data Analytics\Project\Database)

> dir()                    ## working directory listing

>ls()                     ## Workspace listing of objects

>rm('object')             ## Remove an element "object", if exist

> rm(list = ls())         ## Cleaning
```

READING DATA FROM A DATA FILE

```
> setwd("D:/arpita/data analytics/my work") #Set the working directory to file location

> getwd()

[1] "D:/arpita/data analytics/my work“

> dir()

 [1] "Arv.txt"           "DiningAtSFO"       "LatentView-DPL"     "TC-10-Rec.csv"      "TC.csv"

rm(list=ls(all=TRUE))  # Refresh session

> data=read.csv('iris.csv', header = T, sep=",")

(data = read.table('iris.csv', header = T, sep = ','))

> ls()

[1] "data"

> str(data)

'data.frame':          149 obs. of  5 variables:

 $ X5.1       : num  4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 5.4 ...

 $ X3.5       : num  3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 ...

 $ X1.4       : num  1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 ...

 $ X0.2       : num  0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 0.2 ...

 $ Iris.setosa: Factor w/ 3 levels "Iris-setosa",..: 1 1 1 1 1 1 1 1 1 1 ...
```

ACCESSING ELEMENTS FROM A FILE

```
> data$X5.1
 [1] 4.9  4.7  4.6  5.0  5.4  4.6  5.0  4.4  4.9  5.4  4.8  4.8  4.3  5.8  5.7
> data$X5.1[7]=5.2
> data$X5.1
  [1] 4.9  4.7  4.6  5.0  5.4  4.6  5.2  4.4  4.9  5.4  4.8  4.8  4.3  5.8  5.7
```

#Note: This change has happened in workspace only not in the file.

How to make it permanent?

write.csv / write.table

>write.table(data, file ='iris_mod.csv', row.names = FALSE, sep = ',')

If row.names is TRUE, R adds one ID column in the beginning of file.

So its suggested to use row.names = FALSE option

>write.csv(data, file =='iris_mod.csv', row.names = TRUE) ## to test

DIFFERENT DATA ITEMS IN R

Vector

Matrix

Data Frame

List

VECTORS IN R

```
>x=c(1,2,3,4,56)
>x
> x[2]
> x  =  c(3, 4, NA, 5)
>mean(x)
[1] NA
>mean(x, rm.NA=T)
[1] 4
> x  =  c(3, 4, NULL, 5)
>mean(x)
[1] 4
```

MORE ON VECTORS IN R

>y = c(x,c(-1,5),x)

>length(x)

>length(y)

There are useful methods to create long vectors whose elements are in arithmetic progression:

> x=1:20

> x

If the common difference is not 1 or -1 then we can use the seq function

> y=seq(2,5,0.3)

> y

[1] 2.0 2.3 2.6 2.9 3.2 3.5 3.8 4.1 4.4 4.7 5.0

> length(y)

[1] 11

MORE ON VECTORS IN R

```
> x=1:5
> mean(x)
[1] 3
> x
[1] 1 2 3 4 5
> x^2
[1]  1  4  9 16 25
> x+1
[1] 2 3 4 5 6
> 2*x
[1]  2  4  6  8 10
> exp(sqrt(x))
[1] 2.718282 4.113250 5.652234 7.389056
9.356469
```

- It is very easy to add/subtract/multiply/divide two vectors entry by entry.
- > y=c(0,3,4,0)
- > x+y
- [1] 1 5 7 4 5
- > y=c(0,3,4,0,9)
- > x+y
- [1] 1 5 7 4 14
- Warning message:
- In x + y : longer object length is not a multiple of shorter object length
- > x=1:6
- > y=c(9,8)
- > x+y
- [1] 10 10 12 12 14 14

MATRICES IN R

Same data type/mode – number , character, logical

a.matrix <- matrix(vector, nrow = r, ncol = c, byrow = FALSE, dimnames = list(char-vector-rownames, char-vector-col-names))

dimnames is optional argument, provides labels for rows & columns.

```
> y <- matrix(1:20, nrow = 4, ncol = 5)

>A = matrix(c(1,2,3,4),nrow=2,byrow=T)

>A

>A = matrix(c(1,2,3,4),ncol=2)

>B = matrix(2:7,nrow=2)

>C = matrix(5:2,ncol=2)

>mr <- matrix(1:20, nrow = 5, ncol = 4, byrow = T)

>mc <- matrix(1:20, nrow = 5, ncol = 4)

>mr

>mc
```

MORE ON MATRICES IN R

```
>dim(B)          #Dimension
>nrow(B)
>ncol(B)
>A+C
>A-C
>A%*%C          #Matrix multiplication. Where will be the result?
>A*C             #Entry-wise multiplication
>t(A)            #Transpose
>A[1,2]
>A[1,]
>B[1,c(2,3)]
>B[,-1]
```

LISTS IN R

Vectors and matrices in R are two ways to work with a collection of objects.

Lists provide a third method. Unlike a vector or a matrix a list can hold different kinds of objects.

One entry in a list may be a number, while the next is a matrix, while a third is a character string (like "Hello R!").

Statistical functions of R usually return the result in the form of lists. So we must know how to unpack a list using the $ symbol.

EXAMPLES OF LISTS IN R

```
>x = list(name="Arun Patel", nationality="Indian", height=5.5,
marks=c(95,45,80))

>names(x)

>x$name

>x$hei                    #abbreviations are OK

>x$marks

>x$m[2]
```

DATA FRAME IN R

A data frame is more general than a matrix, in that different columns can have different modes (numeric, character, factor, etc.).

```
>d <- c(1,2,3,4)
>e <- c("red", "white", "red", NA)
>f <- c(TRUE,TRUE,TRUE,FALSE)
>myframe <- data.frame(d,e,f)
>names(myframe) <- c("ID","Color","Passed") # Variable names
>myframe
>myframe[1:3,]        # Rows 1 , 2, 3 of data frame
>myframe[,1:2]        # Col 1, 2 of data frame
>myframe[c("ID","Color")] #Columns ID and color from data frame
>myframe$ID       # Variable ID in the data frame
```

FACTORS IN R

In R we can make a variable is nominal by making it a factor.

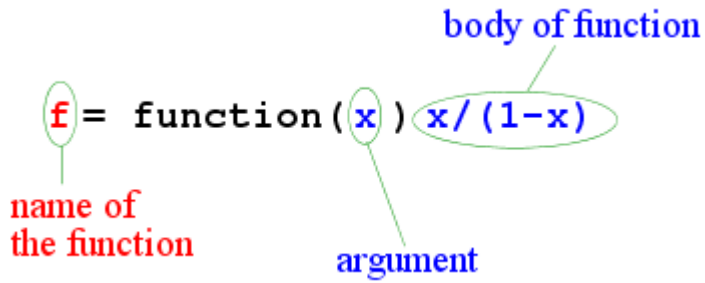
The factor stores the nominal values as a vector of integers in the range [1... k] (where k is the number of unique values in the nominal variable).

An internal vector of character strings (the original values) mapped to these integers.

```
# Example: variable gender with 20 "male" entries and
# 30 "female" entries
>gender <- c(rep("male",20), rep("female", 30))
>gender <- factor(gender)
# Stores gender as 20 1's and 30 2's

# 1=male, 2=female internally (alphabetically)
# R now treats gender as a nominal variable
>summary(gender)
```

FUNCTIONS IN R




```
>g = function(x,y) (x+2*y)/3
>g(1,2)
>g(2,1)
```