Basic R Commands

/** Basic Arithmetic and Logical Operation **/

1. 1 + 1  #addition
2. 10 - 2  #subtraction
3. 2 * 2  #multiplication
4. 100 / 10  #division
5. 10 ^ 2  #exponentiation

/** Standard mathematical function **/

1. Write a command to get the absolute value of -10.
   Soln. abs(-10)
2. Write a command to get the ceil and floor value of 4.5.
   Soln. ceiling(4.2), floor(4.5).
3. Write a command to get the square root value of 100.
   Soln. sqrt(10)
4. Write a command to get the exponential value of 4.
   Soln. exp(2)
5. Write a command to get the pi value.
   Soln. pi
6. Write a command to get the logarithm value of 100.
   Soln. log(100)  #natural logs(i.e. base e)
   log(100, base=10)  #base 10 logs

# Use Parenthesis to clarify order of operator

1. (2+2)*2
2. 1+(2*2)

# Logical operation

1. Check whether 2 and 2 is equal or not
   Soln. 2 == 2
2. Check whether 5 is greater than or less than or equal to 6.
   Soln. 5 <= 6, 5 >= 6
3. Assume a data set contains data like 2, 5, 6, 7, 8. Check whether each data is greater than 5 or not.
   Soln. x <- c(2,5,7,8,9)
   x > 5
#Comments

Comments are any text on a line following #
e.g. mean(c(1,2,3,4))   # end of comment

# Multiple line commands

Command can generally span multiple lines as long R does not think the command has not finished

1. x<-c(“apple”,
“banana”)               # This works
2. Y<-10
   +10               #This does not work

# Multiple command on one line

x<-c(1,2,3);y<-rnorm(4,5)
x;y

#Understanding Directories

1. getwd()
2. setwd("")
3. from R studio you can change directory by following the path-session-set working directory-to project directory

#Workspaces and environments

1. Write a command to list environments
   Soln. search()
2. Write a command to create some objects in the global environments
   Soln. x<-1:10
        y<-1:5
3. Write a command to show objects in the global environment.
   Soln. ls( ).

#Removing objects

1. Write a command to remove the object what you have created just now.
   Soln. rm(x)
2. Write a command to remove all objects from global workspaces.
   Soln. rm(list==ls())

# Saving Objects

Save all objects in the workspace
save.image()
# Data types : logical, character, numeric

Basic data type :

Vector:

\[
x \leftarrow \text{c(TRUE, FALSE)} \quad \# \text{logical vector}
\]

\[
y \leftarrow \text{c(“a”, “b”, “cat”)} \quad \# \text{character vector}
\]

\[
z \leftarrow \text{c(1, 2, 3, 4)} \quad \# \text{numeric integer vector}
\]

\[
m \leftarrow \text{c(1, 2, 1.5, 1.7)} \quad \# \text{numeric real/double vector}
\]

\[
\text{class(x)}; \ \text{typeof(x)}; \ \text{mode(x)}
\]

\[
\text{class(y)}; \ \text{typeof(y)}; \ \text{mode(y)}
\]

#apropose(“^is”)

1. Write a command to check whether the data (“a”, “b”) is numeric or character
   Soln. \( \text{is.numeric(c(“a”, “b”))} \)

#Type conversion

Explicit Type conversion:

#apropose(“^as”)

1. Write a command to convert 1, 2, 3 to character type.
   Soln. \( \text{as.character(c(1, 2, 3))} \)

2. Write a command to convert (“cat”, “bat”) to numeric.
   Soln. \( \text{as.numeric(c(“cat”, “4”, “b”))} \)

implicit conversion

e.g \( \text{sum(c(FALSE, TRUE, TRUE))} \) \# convert logical to 0-1 numeric

#Creating Vector

1. Write a command to create a vector of 1 to 8.
   Soln. \( \text{c(1, 2, 3, 4, 5, 6, 7, 8)} \)
   1:10
   \( \text{seq(1, 8, by =2)} \)
   \( \text{rep(1, 5)} \)
   \( \text{rep(c(1, 2, 3), 5)} \)

2. Write a command to give name to vector what you have created just now.
   Soln. \( x \leftarrow \text{c(1, 2, 3, 4, 5)} \)
   \( \text{names(x)} \leftarrow \text{c(“a”, “b”, “c”, “d”, “e”)} \)
3. Write a command to extract vector by position, by name, by logic what you have created.
   Soln. 
   \[ x[c(1,2)] \] # by numeric position
   \[ x[x<3] \] # by logical vector
   \[ x[c("b","c") \] # by name

Matrices:

**all data must be of same type (numeric, character, logical)**

1. Write a command to create a matrix.
   Soln. 
   \[ y<-matrix(c(1,2,3,4,5,6), byrow=TRUE, ncol=2) \]

2. Check the class of y
   Soln. \[ class(y) \]

3. Write a command to get the no. of rows and cols
   Soln. 
   \[ dim(y) \] # no of rows and columns
   \[ nrow(y) \] # no of rows
   \[ ncol(y) \] # no of column

4. Write a command to give name to Rows and Columns
   Soln. 
   \[ rownames(y)<-(c("a","b","c") \]
   \[ colnames(y)<-(c("col1","col2") \]

5. Write a command to extract matrix data
   Soln. 
   \[ y["a",] \] # by row name
   \[ y[,"col1"] \] # by column name
   \[ y["a","col1"] \] # by both
   \[ y[c(1,2),] \] # by row position
   \[ y[.1] \] # by col position

Lists:

Lists store arbitrary structures of one or more named elements. Elements can be different length. Lists can contain lists can be nested to tree like structure. Lists are commonly used for representing result of data analysis.

1. Write a command to create a list.
   Soln. 
   \[ x = list(name="Arun Patel", nationality="Indian", height=5.5, grades=c(95,45,80)) \]

2. Write a command to access elements of list.
   Soln. 
   \[ x$name \] # using dollar notation
   \[ x$hei \] # abbreviation is ok
   \[ x[[1]] \] # by position

Data Frame:

Data frames are standard data structure used for storing data.
**columns can be of different data types (e.g. character, numeric, logical)**
1. Write a command to create data frame that stores numbers 1-9 and letters a-e.
   Soln.  
   ```r
   z<-data.frame(var1=1:9,var2=letters[1:9])
   ```

2. Write a command to filter the variable from data frame.
   Soln.  
   ```r
   View(z)
   ```

3. Write a command to load data from your computer.
   Soln.  
   ```r
   data=read.csv("file name",header=T,sep="",)
   ```
   #if it is in workspace directory

4. Write a command to show the no. of column and rows of your dataframe.
   Soln.  
   ```r
   nrow(data)
   ncol(data)
   ```

5. Write a command to see all the data
   Soln.  
   ```r
   data
   ```

6. Write a command to view the first few rows of the data
   Soln.  
   ```r
   head(data)
   ```

7. Write a command to view the last rows of the data
   Soln.  
   ```r
   tail(data)
   ```

8. Write a command to view the column names
   Soln.  
   ```r
   names(data)
   ```

### User-defined functions

# Hello world as a user-defined function
# Use the function keyword to assign code to a function

```r
hw.f1 <- function()
{
  hw <- "Hello World"
  hw
}
```

hw.f1()

### Packages:

R has many additional packages. To use a package you have to be installed. You only need to install a package once. To use a package you need to load the package each time

### Installation

Option1: use the install.packages function

```r
Install.packages("psych")
```

** Some packages rely on other packages
dependencies=TRUE ensures that dependencies are also installed.
```r
Install.packages("psych", dependencies=TRUE)
```

Option 2: use the package tab in R studio.
Click install and enter the package details.
# Loading an installed package

Option 1: use the library function
library(psych) #i.e. put this at the start of your script

Option 2:

Common errors:
Not having a package installed is a common error. If you try to load a package that is not installed
e.g.
library(foo)
You will get an error. Error in library (foo). There is no package called”foo”.
This means:
You mistyped the name of the package, or
You need to install the package
Install.packages(“foo”)

# Missing data

Missing is represented in R by NA

x<-c(1,2,NA,4)
x<-c(“a”,”b”,NA,”c”)
1. Write a command to see whether a value is missing or not.
   Soln. is.na(x)
   **If you have missing data, some functions will return NA by default rather than
   returning a value.
   e.g. mean(x)
   sd(x)
   2. Write a command to remove NA value .
   Soln. mean(x, na.rm=TRUE)
   Soln. sd(x,na.rm=TRUE)

   or you remove the missing data

   na.omit(x)

   3. Write a command to calculate mean after removing missing value.
   Soln. mean(na.omit(x))

#getting summary of data frames

library(MASS) #user survey data from MASS package
data(survey) #load an internal dataset
data() # list all datasets in MASS package
mydata<-survey

# Variable name
names(mydata)  #give the name of attributes of survey data.

# show structure
str(mydata)

# Summaries of numeric vector (or data frame variables)
x <- c(1,2,3,4,5)

# total
sum(x)  #sum of vector
prod(x)  #product of vector

#Basic Statistics

Central tendency :

mean(x)  #mean of vector
median(x)  #median of vector
length(x)  #length of vector

Spread :

sd(x)  #standard deviation
var(x)  #variance x
range(x)  #range x
min(x)  #min of vector
max(x)  #max of vector

vector operation:

dat<-data.frame(x=c(1,2,3,4,5),y=c(1,1,0,1,1))

vector operation typically operate on element wise.
dat$z<-dat$x + dat$y
dat$z<-dat$x+10