Unit-1 Examples and Exercises

Santosh Kumar Sah

2025-03-11

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

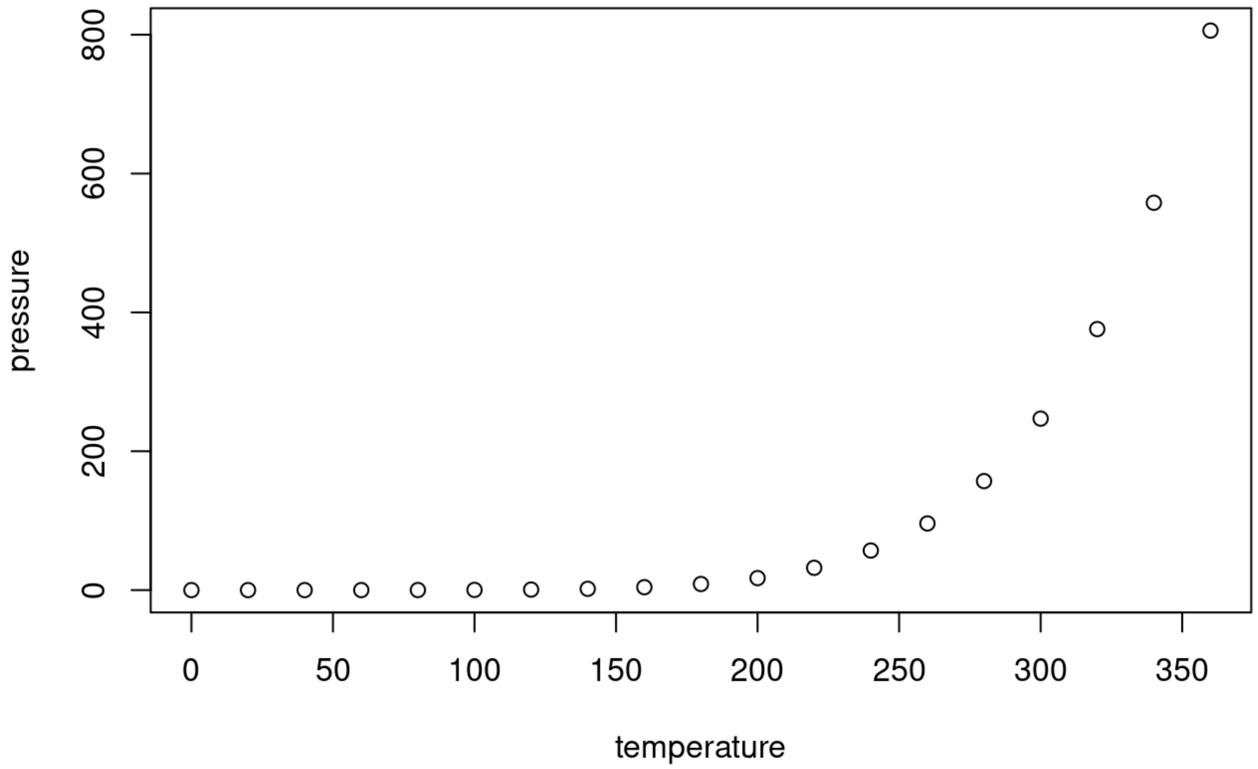
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Session-1 Code Examples and Exercises

Simple Data Entry in R.

```
data <- c(1,2,3,4,5,6,7,8)
data
```

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

```
text <-c("a","b","c",'d')
print(text)
```

```
## [1] "a" "b" "c" "d"
```

```
#text2 <- c(a,c,v,e,f)
#text2

data2<-cbind(data,text)

data2
```

```
##      data text
## [1,] "1"  "a"
## [2,] "2"  "b"
## [3,] "3"  "c"
## [4,] "4"  "d"
## [5,] "5"  "a"
## [6,] "6"  "b"
## [7,] "7"  "c"
## [8,] "8"  "d"
```

Array and Metrices in R.

```
M<-matrix(
c(1:9),
nrow=3,
ncol=3,
byrow=TRUE
)

print(M)
```

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

```
V<-c(1:12)
V
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

MultiDimensional Array

```
#MDA<-array(V,dim(2,3,))

#MDA<-array(V,dim(2,3,2))
#MDA<-array(V,dim=(2,3,2))
MDA<-array(V, dim = c(2,3,2))

print(MDA)
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]    7    9   11
## [2,]    8   10   12
```

Creating a Simple data.frame in R.

```
df<-data.frame(x=c(1,2,3),y=c(2,3,4),z=c(3,4,5))
df
```

```
##   x y z
## 1 1 2 3
## 2 2 3 4
## 3 3 4 5
```

```
class(df)
```

```
## [1] "data.frame"
```

A small but realistic dataframe and its use.

create dataframe

```
emp.data<-data.frame(
emp_id=c(1:5),
emp_name=c("Dyan","Mack","Ryan","Gary","Rick"),
salary=c(623.5,524.43,611.0,845.0,727.94),
start_date=as.Date(c("2012-01-01","2013-09-23","2014-11-25","2014-05-11","2015-0
3-27")),
stringAsFactors=FALSE
)


#### Print The data
print(emp.data)
```

```
##   emp_id emp_name salary start_date stringAsFactors
## 1      1     Dyan 623.50 2012-01-01           FALSE
## 2      2     Mack 524.43 2013-09-23           FALSE
## 3      3     Ryan 611.00 2014-11-25           FALSE
## 4      4     Gary 845.00 2014-05-11           FALSE
## 5      5     Rick 727.94 2015-03-27           FALSE
```

Structure and Summary of the sample dataframe in R.

```
print(str(emp.data)) # get the structure of the dataframe
```

```
## 'data.frame':    5 obs. of  5 variables:
##  $ emp_id         : int  1 2 3 4 5
##  $ emp_name       : chr  "Dyan" "Mack" "Ryan" "Gary" ...
##  $ salary         : num  624 524 611 845 728
##  $ start_date     : Date, format: "2012-01-01" "2013-09-23" ...
##  $ stringAsFactors: logi  FALSE FALSE FALSE FALSE FALSE
## NULL
```

```
#### Print the summary of the emp.data
print(summary(emp.data))
```

```
##      emp_id    emp_name             salary        start_date
##  Min.   :1   Length:5           Min.   :524.4   Min.   :2012-01-01
##  1st Qu.:2   Class :character   1st Qu.:611.0   1st Qu.:2013-09-23
##  Median :3   Mode  :character   Median :623.5   Median :2014-05-11
##  Mean   :3                      Mean   :666.4   Mean   :2014-01-16
##  3rd Qu.:4                      3rd Qu.:727.9   3rd Qu.:2014-11-25
##  Max.   :5                      Max.   :845.0   Max.   :2015-03-27
##  stringAsFactors
##  Mode :logical
##  FALSE:5
##
##
##
##
```

Extract part of data from dataframe in R,

```
result<-data.frame(emp.data$emp_name, emp.data$salary) # Extract specific columns
print(result)
```

```
##   emp.data.emp_name emp.data.salary
## 1              Dyan          623.50
## 2              Mack          524.43
## 3              Ryan          611.00
## 4              Gary          845.00
## 5              Rick          727.94
```

```
result<-emp.data[1:2,] # extract first two rows
print(result)
```

```
##   emp_id emp_name salary start_date stringAsFactors
## 1      1     Dyan 623.50 2012-01-01           FALSE
## 2      2     Mack 524.43 2013-09-23           FALSE
```

```
result<-emp.data[c(3,5), c(2,4)] # extract 3rd and 5th row with 2nd and 4th column
print(result)
```

```
##   emp_name start_date
## 3     Ryan 2014-11-25
## 5     Rick 2015-03-27
```

Add a new column in existing dataframe.

```
emp.data$dept<-c("IT","Operations","IT","HR","Finance") # Add the 'dept' column
v<-emp.data
print(v)
```

```
##   emp_id emp_name salary start_date stringAsFactors       dept
## 1      1     Dyan 623.50 2012-01-01           FALSE         IT
## 2      2     Mack 524.43 2013-09-23           FALSE Operations
## 3      3     Ryan 611.00 2014-11-25           FALSE         IT
## 4      4     Gary 845.00 2014-05-11           FALSE         HR
## 5      5     Rick 727.94 2015-03-27           FALSE    Finance
```

Expand dataframe in R. (Adding Cases)

```
emp.newdata<-data.frame(
emp_id=c(6:8),
emp_name=c("Rashmi","Pranab","Tushar"),
salary=c(623.5,524.43,611.0),
start_date=as.Date(c("2014-11-25","2014-05-11","2015-03-27")),
dept=c("IT","Operations","Finance"),
stringAsFactors=FALSE
)
emp.newdata
```

```
##   emp_id emp_name salary start_date       dept stringAsFactors
## 1      6   Rashmi 623.50 2014-11-25         IT           FALSE
## 2      7   Pranab 524.43 2014-05-11 Operations           FALSE
## 3      8   Tushar 611.00 2015-03-27    Finance           FALSE
```

Expand data frame in R (rbind is used)

```
emp.finaldata<-rbind(emp.data,emp.newdata)
emp.finaldata
```

```
##   emp_id emp_name salary start_date stringAsFactors       dept
## 1      1     Dyan 623.50 2012-01-01           FALSE         IT
## 2      2     Mack 524.43 2013-09-23           FALSE Operations
## 3      3     Ryan 611.00 2014-11-25           FALSE         IT
## 4      4     Gary 845.00 2014-05-11           FALSE         HR
## 5      5     Rick 727.94 2015-03-27           FALSE    Finance
## 6      6   Rashmi 623.50 2014-11-25           FALSE         IT
## 7      7   Pranab 524.43 2014-05-11           FALSE Operations
## 8      8   Tushar 611.00 2015-03-27           FALSE    Finance
```

Session-3 code examples and exercises

Basics of R

```
print(4*6+5)
```

```
## [1] 29
```

```
print((4*6)+5)
```

```
## [1] 29
```

```
print(4*(6+5))
```

```
## [1] 44
```

```
print((4+6)^2*5/10+9-1)
```

```
## [1] 58
```

Variables in R. Assigning and Removing

```
x<-2
x
```

```
## [1] 2
```

```
x=3
x
```

```
## [1] 3
```

```
4->x
x
```

```
## [1] 4
```

```
assign("x",5)
x
```

```
## [1] 5
```

Data Types

```
x<-c(1,2,3,4,5)
class(x)
```

```
## [1] "numeric"
```

```
y<-c(1:9)
class(y)
```

```
## [1] "integer"
```

```
y<-c(1L:9L)
class(y)
```

```
## [1] "integer"
```

```
y<-c(1L,2L,3L,4L,5L)
class(y)
```

```
## [1] "integer"
```

```
is.numeric(y)
```

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

character

```
x<-"data"
x
```

```
## [1] "data"
```

```
class(x)
```

```
## [1] "character"
```

```
nchar(x)
```

```
## [1] 4
```

Factor

```
y<-factor("data")
y
```

```
## [1] data
## Levels: data
```

```
class(y)
```

```
## [1] "factor"
```

Factoris used to create and store categorical variable in R.

```
gender<-factor(c("male","female","female","male"))
typeof(gender) # datatype
```

```
## [1] "integer"
```

```
attributes(gender)  # Levels and class
```

```
## $levels
## [1] "female" "male"
##
## $class
## [1] "factor"
```

```
unclass(gender) # check how it is stored in R.
```

```
## [1] 2 1 1 2
## attr(,"levels")
## [1] "female" "male"
```

Date

```
date1<-as.Date("2023-03-29")
date1
```

```
## [1] "2023-03-29"
```

```
class(date1)
```

```
## [1] "Date"
```

```
as.numeric(date1)
```

```
## [1] 19445
```

```
date2<-as.POSIXct("2023-03-29 6:30")
date2
```

```
## [1] "2023-03-29 06:30:00 +0545"
```

```
class(date2)
```

```
## [1] "POSIXct" "POSIXt"
```

```
as.numeric(date2)
```

```
## [1] 1680050700
```

Logical

```
k<-TRUE
class(k)
```

```
## [1] "logical"
```

```
is.logical(k)
```

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

```
TRUE*5
```

```
## [1] 5
```

```
2==3
```

```
## [1] FALSE
```

```
2!=3
```

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

```
2<3
```

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

```
"data"=="stats"
```

```
## [1] FALSE
```

```
"data"<"stats"
```

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

Vectors and its operations

```
x<-c(1,2,3,4,5)
x
```

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

```
x*3
```

```
## [1]  3  6  9 12 15
```

```
x+2
```

```
## [1] 3 4 5 6 7
```

```
x-3
```

```
## [1] -2 -1  0  1  2
```

```
x^2
```

```
## [1]  1  4  9 16 25
```

```
sqrt(x)
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

Two vectors of equal length

```
x<-1:10
y<--5:4
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
y
```

```
##  [1] -5 -4 -3 -2 -1  0  1  2  3  4
```

```
length(x)
```

```
## [1] 10
```

```
length(y)
```

```
## [1] 10
```

```
x+y
```

```
##  [1] -4 -2  0  2  4  6  8 10 12 14
```

```
x-y
```

```
##  [1] 6 6 6 6 6 6 6 6 6 6
```

```
length(x+y)
```

```
## [1] 10
```

```
x*y
```

```
##  [1] -5 -8 -9 -8 -5  0  7 16 27 40
```

```
x/y
```

```
##  [1] -0.2 -0.5 -1.0 -2.0 -5.0  Inf  7.0  4.0  3.0  2.5
```

```
x^y
```

```
##  [1] 1.000000e+00 6.250000e-02 3.703704e-02 6.250000e-02 2.000000e-01
##  [6] 1.000000e+00 7.000000e+00 6.400000e+01 7.290000e+02 1.000000e+04
```

Two vectors of unequal length

```
x<-1:10
y<-c(1,2)
x+y
```

```
##  [1]  2  4  4  6  6  8  8 10 10 12
```

```
z<-c(1,2,3)
x+z
```

```
## Warning in x + z: longer object length is not a multiple of shorter object
## length
```

```
##  [1]  2  4  6  5  7  9  8 10 12 11
```

Comparing vectors

```
x<-10:1
x
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```
y<--4:5
y
```

```
##  [1] -4 -3 -2 -1  0  1  2  3  4  5
```

```
x>y
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```
x<=5
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
x<y
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

```
any(x<y)
```

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

```
any(x>y)
```

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

```
all(x>y)
```

```
## [1] FALSE
```

```
nchar(y)
```

```
##  [1] 2 2 2 2 1 1 1 1 1 1
```

```
x
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```
x[1] # retrieve first element of x
```

```
## [1] 10
```

```
#x[1,2] # retrieves first and second element of x
x[c(1,4)]
```

```
## [1] 10  7
```

```
w<-1:3
names(w)<-c("a","b","c")
w
```

```
## a b c
## 1 2 3
```

NA types missing data in R

```
zchar<-c("Hockey",NA,"CRicket")
zchar
```

```
## [1] "Hockey"  NA        "CRicket"
```

```
nchar(z)
```

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

```
is.na(zchar)
```

```
## [1] FALSE  TRUE FALSE
```

```
z<-c(1,2,NA,4,5,NA)
mean(z)
```

```
## [1] NA
```

```
mean(z,na.rm=TRUE)
```

```
## [1] 3
```

```
var(z,na.rm = TRUE)
```

```
## [1] 3.333333
```

NULL type missing data in R.

```
z<-c(1,NULL,3)
z
```

```
## [1] 1 3
```

```
is.null(z)
```

```
## [1] FALSE
```

```
d<-NULL
is.null(d)
```

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

Pipes in R

```
library(magrittr)
x<-1:10
x%>%mean
```

```
## [1] 5.5
```

Chained Pipes in R

```
z<-c(1,2,NA,8,3,NA,3)
z%>%is.na%>%sum
```

```
## [1] 2
```

```
z%>%mean(na.rm = TRUE)
```

```
## [1] 3.4
```

Advanced Data Structures in R

```
x<-10:1
y< -4:5
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
q <-c("Hockey","Football","Baseball", "Kabaddi", "Rugby","Pingpong", "Basketbal
l","Tennis", "Cricket", "Volleyball")
theDF <-data.frame(x, y, q)
theDF
```

```
##     x  y          q
## 1  10 -4     Hockey
## 2   9 -3   Football
## 3   8 -2   Baseball
## 4   7 -1    Kabaddi
## 5   6  0      Rugby
## 6   5  1   Pingpong
## 7   4  2 Basketball
## 8   3  3     Tennis
## 9   2  4    Cricket
## 10  1  5 Volleyball
```

```
theDF <-data.frame(First=x,Second=y, Sport=q)
names(theDF)
```

```
## [1] "First"  "Second" "Sport"
```

```
names(theDF)[3]
```

```
## [1] "Sport"
```

```
rownames(theDF)
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

```
rownames(theDF) <- c("One","Two", "Three", "Four", "Five","Six", "Seven", "Eight",
"Nice","Ten")

rownames(theDF) <- NULL
rownames(theDF)
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

```
#Printing first few rows
head(theDF)
```

```
##   First Second    Sport
## 1    10     -4   Hockey
## 2     9     -3 Football
## 3     8     -2 Baseball
## 4     7     -1  Kabaddi
## 5     6      0    Rugby
## 6     5      1 Pingpong
```

```
#Printing first seven rows
head(theDF, n=7)
```

```
##   First Second      Sport
## 1    10     -4     Hockey
## 2     9     -3   Football
## 3     8     -2   Baseball
## 4     7     -1    Kabaddi
## 5     6      0      Rugby
## 6     5      1   Pingpong
## 7     4      2 Basketball
```

```
#Printing last few rows
tail(theDF)
```

```
##    First Second      Sport
## 5      6      0      Rugby
## 6      5      1   Pingpong
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5 Volleyball
```

```
class(theDF)
```

```
## [1] "data.frame"
```

```
#Structure of data frame by
#variables
str(theDF)
```

```
## 'data.frame':    10 obs. of  3 variables:
##  $ First : int  10 9 8 7 6 5 4 3 2 1
##  $ Second: int  -4 -3 -2 -1 0 1 2 3 4 5
##  $ Sport : chr  "Hockey" "Football" "Baseball" "Kabaddi" ...
```

```
theDF[3,2]; theDF[3, 2:3]
```

```
## [1] -2
```

```
##   Second    Sport
## 3     -2 Baseball
```

```
theDF[, 3]; theDF[3,]
```

```
##  [1] "Hockey"     "Football"   "Baseball"   "Kabaddi"    "Rugby"
##  [6] "Pingpong"   "Basketball" "Tennis"     "Cricket"    "Volleyball"
```

```
##   First Second    Sport
## 3     8     -2 Baseball
```

```
theDF[, c("First", "Sport")]
```

```
##    First      Sport
## 1     10     Hockey
## 2      9   Football
## 3      8   Baseball
## 4      7    Kabaddi
## 5      6      Rugby
## 6      5   Pingpong
## 7      4 Basketball
## 8      3     Tennis
## 9      2    Cricket
## 10     1 Volleyball
```

```
theDF[, "Sport", drop=FALSE]
```

```
##         Sport
## 1      Hockey
## 2    Football
## 3    Baseball
## 4     Kabaddi
## 5       Rugby
## 6    Pingpong
## 7  Basketball
## 8      Tennis
## 9     Cricket
## 10 Volleyball
```

Lists in R

```
#Three element list
list1 <- list(1,2,3)
#Single element list
list2 <- list(c(1,2,3))
#Two vector list
list3 <- list(c(1,2,3), 3:7)
#List with data.frame and vector
list4 <- list(theDF, 1:10)
#Three element list
list5 <- list(theDF, 1:10, list3)
#Names of the list
names(list5)
```

```
## NULL
```

```
names(list5) <-c("data.frame","vector", "list")
names(list5)
```

```
## [1] "data.frame" "vector"     "list"
```

```
list5
```

```
## $data.frame
##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1    Kabaddi
## 5      6      0      Rugby
## 6      5      1   Pingpong
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5 Volleyball
## 
## $vector
##  [1]  1  2  3  4  5  6  7  8  9 10
## 
## $list
## $list[[1]]
## [1] 1 2 3
## 
## $list[[2]]
## [1] 3 4 5 6 7
```

```
list6 <- list(TheDataFrame=theDF,
TheVector=1:10, TheList=list3)
names(list6)
```

```
## [1] "TheDataFrame" "TheVector"    "TheList"
```

Access Elements of Lists

```
#Use double square brackets

#Specify either the element number or name
list5[[1]]
```

```
##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1    Kabaddi
## 5      6      0      Rugby
## 6      5      1   Pingpong
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5 Volleyball
```

```
list5[["data.frame"]]
```

```
##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1    Kabaddi
## 5      6      0      Rugby
## 6      5      1   Pingpong
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5 Volleyball
```

```
# This allows access to only one element at a time

#Accessed element manipulation
list5[[1]]$Sport #Sport variable
```

```
##  [1] "Hockey"     "Football"   "Baseball"   "Kabaddi"    "Rugby"
##  [6] "Pingpong"   "Basketball" "Tennis"     "Cricket"    "Volleyball"
```

```
list5[[1]][, "Second"]
```

```
##  [1] -4 -3 -2 -1  0  1  2  3  4  5
```

```
list5[[1]][, "Second", drop=F]
```

```
##    Second
## 1      -4
## 2      -3
## 3      -2
## 4      -1
## 5       0
## 6       1
## 7       2
## 8       3
## 9       4
## 10      5
```

```
length(list5)
```

```
## [1] 3
```

```
#Adding new element
list5[[4]] <- 2
list5[["NewElement"]] <-3:6
```

Matrices in R

```
A <- matrix(1:10, nrow=5)
B <- matrix(21:30, nrow=5)
C <- matrix(21:40, nrow=2)

nrow(A)
```

```
## [1] 5
```

```
ncol(B)
```

```
## [1] 2
```

```
dim(C)
```

```
## [1]  2 10
```

```
A + B
```

```
##      [,1] [,2]
## [1,]   22   32
## [2,]   24   34
## [3,]   26   36
## [4,]   28   38
## [5,]   30   40
```

```
A * B
```

```
##      [,1] [,2]
## [1,]   21  156
## [2,]   44  189
## [3,]   69  224
## [4,]   96  261
## [5,]  125  300
```

```
A - B
```

```
##      [,1] [,2]
## [1,]  -20  -20
## [2,]  -20  -20
## [3,]  -20  -20
## [4,]  -20  -20
## [5,]  -20  -20
```

```
A = B
```

Matrix Multiplication and names in R.

```
# A %*% C will work
# A %*% B will not work
# Both A and B are 5 x 2 matrices
# so we will transpose B
A %*% t(B)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 1117 1164 1211 1258 1305
## [2,] 1164 1213 1262 1311 1360
## [3,] 1211 1262 1313 1364 1415
## [4,] 1258 1311 1364 1417 1470
## [5,] 1305 1360 1415 1470 1525
```

```
#Column/row names of matrix:
colnames(A)
```

```
## NULL
```

```
colnames(A) <- c("Left", "Right")
rownames(A) <- c("1st", "2nd",
"3rd", "4th", "5th")
t(A)
```

```
##       1st 2nd 3rd 4th 5th
## Left   21  22  23  24  25
## Right  26  27  28  29  30
```

```
colnames(B) <- c("First",
"Second")
rownames(B) <- c("One", "Two",
"Three", "Four", "Five")
```

Arrays in R.

```
theArray <- array(1:12,dim=c(2,3,2))
# 2 dimensional matrices both with 2 rows and 3 columns

theArray [1, , ] # 1st row of both
```

```
##      [,1] [,2]
## [1,]    1    7
## [2,]    3    9
## [3,]    5   11
```

```
theArray[1, ,1] #1st row of first
```

```
## [1] 1 3 5
```

```
theArray[,1,] # 1st column of both
```

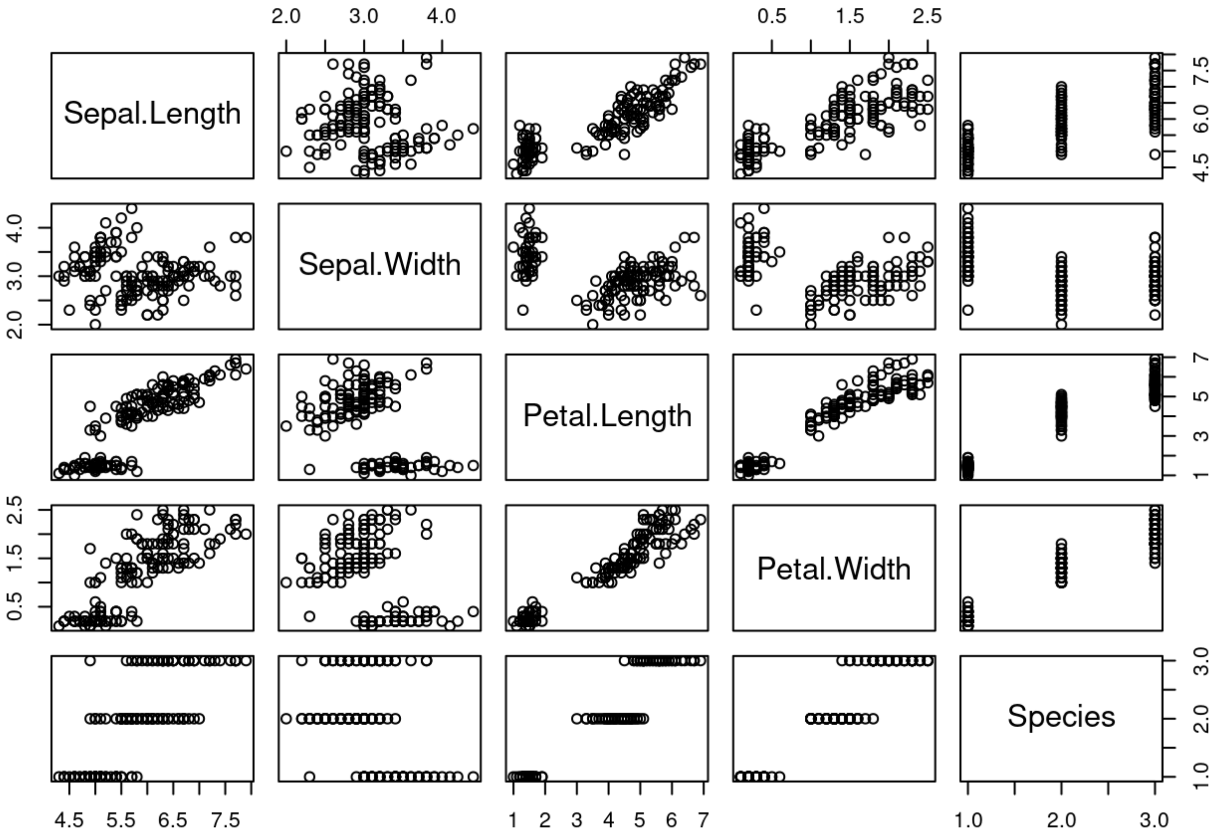
```
##      [,1] [,2]
## [1,]    1    7
## [2,]    2    8
```

Load the iris data from Internet.

```
iris <- read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"), header = FALSE)
head(iris)
```

```
##    V1  V2  V3  V4          V5
## 1 5.1 3.5 1.4 0.2 Iris-setosa
## 2 4.9 3.0 1.4 0.2 Iris-setosa
## 3 4.7 3.2 1.3 0.2 Iris-setosa
## 4 4.6 3.1 1.5 0.2 Iris-setosa
## 5 5.0 3.6 1.4 0.2 Iris-setosa
## 6 5.4 3.9 1.7 0.4 Iris-setosa
```

```
# Add column names for V1, V2, V3, V4 and V5 columns to the Iris data
names(iris) <- c("Sepal.Length", "Sepal.Width", "Petal.Length",
"Petal.Width", "Species")
write.csv(iris, "iris.csv")
library(magrittr) #for pipes
sink("session3.out")
plot(iris)
```



```
summary(iris)
iris$Sepal.Length.SQRT <- iris$Sepal.Length %>% sqrt()
#iris %>% cor(Sepal.Length, Sepal.Width)
sink()
detach("package:magrittr")
```

Session-5 code Examples and Exercises

```
# Built In Functions in R.
#round()
print(round(3.1415))
```

```
## [1] 3
```

```
round(3.1415, digits = 2)
```

```
## [1] 3.14
```

```
#factorial()
factorial(3)
```

```
## [1] 6
```

```
factorial(2*3)
```

```
## [1] 720
```

```
#mean()
mean(1:6)
```

```
## [1] 3.5
```

```
mean(c(1:30))
```

```
## [1] 15.5
```

“Sample” function: Random sampling without or with replacement in R

```
die <- 1:6
sample(x = die, size = 1)
```

```
## [1] 5
```

```
sample(x = die, size = 1)
```

```
## [1] 2
```

```
sample (x = die, size = 1, replace=TRUE)
```

```
## [1] 5
```

```
sample(x = die, size = 2)
```

```
## [1] 5 6
```

```
sample(x - die, size = 2)
```

```
## Warning in x - die: longer object length is not a multiple of shorter object
## length
```

```
## [1] -1  5
```

```
sample(x = die, size = 2, replace=TRUE)
```

```
## [1] 6 6
```

User Defined Functions in R

```r
roll <- function() {
die <- 1:6
dice <- sample(die, size = 2, replace = TRUE)
sum(dice)
}
# Function with arguments with default values
roll2 <- function(dice = 1:6) {

dice <- sample(dice, size = 2, replace = TRUE)
sum(dice)

}

roll()
```

```
## [1] 5
```

```r
roll()
```

```
## [1] 7
```

```r
roll()
```

```
## [1] 8
```

```r
roll2()
```

```
## [1] 10
```

```r
roll2()
```

```
## [1] 11
```

```r
roll2()
```

```
## [1] 10
```

Loops in R.

```
# for loop
#while loop
print_words <- function(sentence) {
for (word in sentence) {
print(word)
}
}

best_practice <- c("Let", "the", "computer", "do", "the", "work")
print_words(best_practice)
```

```
## [1] "Let"
## [1] "the"
## [1] "computer"
## [1] "do"
## [1] "the"
## [1] "work"
```

```
print_words(best_practice[-6])
```

```
## [1] "Let"
## [1] "the"
## [1] "computer"
## [1] "do"
## [1] "the"
```

Condition If and Else

#Checking values of y with x:

```
if (y < 20) {
x <- "Too low"
} else {
x <- "Too high"
}
```

```
## Warning in if (y < 20) {: the condition has length > 1 and only the first
## element will be used
```

```
#Can you get anything from this?

#Will this work?
check.y <- function(y) {
if (y < 20) {
print("Too Low") } else {
print("Two high")
}}

check.y(10)
```

```
## [1] "Too Low"
```

```
check.y(30)
```

```
## [1] "Two high"
```

Creating binary variables with “ifelse”

```
#Will this work?
y <- 1:40
ifelse(y<20, "Too low", "Too high")
```

```
##  [1] "Too low"  "Too low"  "Too low"  "Too low"  "Too low"  "Too low" 
##  [7] "Too low"  "Too low"  "Too low"  "Too low"  "Too low"  "Too low" 
## [13] "Too low"  "Too low"  "Too low"  "Too low"  "Too low"  "Too low" 
## [19] "Too low"  "Too high" "Too high" "Too high" "Too high" "Too high"
## [25] "Too high" "Too high" "Too high" "Too high" "Too high" "Too high"
## [31] "Too high" "Too high" "Too high" "Too high" "Too high" "Too high"
## [37] "Too high" "Too high" "Too high" "Too high"
```

```
#It’s a logical as:
ifelse(y<20, TRUE, FALSE)
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE
```

```
y <- 1:40
ifelse(y<20, 1, 0)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
## [39] 0 0
```

Multiple Conditions: combining “ifelse”

```r
# Will this work too?
x <- 1:99

# Binary numbers
x1 <- ifelse(x < 20, 1, 0)

# Binary text
x2.1 <- ifelse(x < 20, "<20", "20+")

# Define categorical variables for different ranges
x2.2 <- ifelse(x >= 20 & x < 40, "20-39", NA)
x2.3 <- ifelse(x >= 40 & x < 100, "40-99", NA)

# Now combine them in a single column with <20 =1, 20-39 = 2 and 40-99 = 3
# Create categorical variable of x
x3 <- ifelse(x < 20, 1, ifelse(x < 40, 2, 3))

# Output the categorical variable
x3
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2
## [39] 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3
## [77] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```r
# Display frequency count of categories
table(x3)
```

```
## x3
##  1  2  3
## 19 20 60
```