Chapter 2: Getting Started with Statistical Programming

2.1 Data Types

Below are the data types that can be written in R:

Table 2.1: Data Types in R

Data Type	Example
Numeric	4.3, -2, 50
Integer	30L, 50L
Complex	2+3i
Logical	TRUE, FALSE, NA
Character	“New world”, “Male”

2.2 Data Structure

There are six data structures in R:

1. Vector
2. Data frame
3. Matrix
4. Array
5. List
6. Factor

2.3 Vector

Vector is a set of elements that share same data type. The element can be many types of data. Vector is used store the information of single variable. Below is how to assign a vector.

Example 1

```
> age<-c(40,23,24)
> age
[1] 40 23 24
```

Example 1 is a numeric vector type because all elements are numbers. Vector also can be written in a form of character data type as in Example 2.

Example 2

```
> color<-c("blue", "red", "green")
> color
[1] "blue" "red" "green"
```

Vector in logical data type can be written as in Example 3.

Example 3

```
> exercise<-c(TRUE, TRUE, FALSE, NA)
> exercise
[1] TRUE TRUE FALSE NA
```

In vector, elements with different data type cannot be mixed. If there are numeric, logical and character elements in the same vector, the numeric and logical elements will be turn into character vector.

Example 4

```
> cats<-c(1, "one", TRUE)
> cats
[1] "1", "one", "TRUE"
```

If numeric vector mix with logical vector, TRUE will be assigned as 1 and FALSE as 0.

Example 5

```
> a<-c(80, TRUE)
> a
[1] 80 1
```

Some of the vectors can be generated easily by using some operators as follows:

Sequence

Example 6

Generate a sequence from 1 to 10

```
> sequence<-1:10
> sequence
[1] 1 2 3 4 5 6 7 8 9 10
```

Example 7

Generate a sequence from 1 to 10 with increment of 2

```
> sequence<-seq(1, 10, 2)
> sequence
[1] 1 3 5 7 9
```

Replicate

Example 8

Generate word female 5 times

```
> female<-rep("Female", 5)
> female
[1] "Female" "Female" "Female" "Female" "Female"
```

2.3.1 Extracting the Elements from Vector

The element in a vector can be extracted by using the following operator:

Example 9

Consider the following vector named as 'data'.

```
data<-c(12,10,15,18,9,21)
```

Select the third element of the vector (no 15).

```
> data[3]
[1] 15
```

Example 10

From Example 9, select the first three element (no 12, 10, 15).

```
> data[1:3]
[1] 12 10 15
```

Example 11

From Example 9, select elements at position 1,4 and 6 (no 12, 18, 21).

```
> data[c(1,4,6)]
[1] 12 18 21
```

2.3.2 Removing the Elements from Vector

The elements in a vector can be removed by using the following operator: From Example 9, remove the second element (remove no 10).

```
> data[-2]
[1] 12 15 18 9 21
```

Example 12

From Example 9, remove the last three element (remove no 18,9,21).

```
> data[-(4:6)]
[1] 12 10 15
```

Example 13

From Example 9, remove elements at position 1,3 and 6 (remove no 12,15,21).

```
> data[-c(1,3,6)]
[1] 10 18 9
```

Example 14

To determine the number of elements in a vector use length().

```
> length(data)
[1] 6
```

2.3.3 Operation in a Vector

We can also use logical operator to extract or remove elements from vector.

Table 2.2: Logical Operators

Operator	Description	Example	Output
`x < y`	x less than y	`3 < 4`	`TRUE`
`x > y`	x greater than y	`3 > 4`	`FALSE`
`x <= y`	x less than or equal to y	`3 <= 4`	`TRUE`
`x >= y`	x greater than or equal to y	`3 >= 4`	`FALSE`
`x == y`	x equal to y	`3 == 4`	`FALSE`
`x != y`	x not equal to y	`3 != 4`	`TRUE`
`!x`	not x	`!(3 > 4)`	`TRUE`
`X | y`	x or y	`(3 > 4) | TRUE`	`TRUE`
`X & y`	x and y	`(3 < 4) & (4 > 0)`	`TRUE`

Refer to **Example 9** to run the following operators.

Example 15

```
> data>10
[1] TRUE FALSE TRUE TRUE FALSE TRUE
```

Example 16

```
> data<10
[1] FALSE FALSE FALSE FALSE TRUE FALSE
```

Example 17

```
> data==10
[1] FALSE TRUE FALSE FALSE FALSE FALSE
```

Example 18

```
> data!=10
[1] TRUE FALSE TRUE TRUE TRUE TRUE
```

Example 19

```
> data>10|data<18
[1] TRUE TRUE TRUE TRUE TRUE TRUE
```

Example 20

```
> data>10&data<18
[1] TRUE FALSE TRUE FALSE FALSE FALSE
```

Example 21

```
> data[data>15]
[1] 18 21
```

Example 22

```
> data[data!=10]
[1] 12 15 18 9 21
```

Example 23

```
> sum(data<15)
[1] 3
```

Example 24

```
> which(data>15)
[1] 4 6
```

Example 25

```
> data[which(data>15)]
[1] 18 21
```

Example 26

```
> min(data)
[1] 9
```

Example 27

```
> which(data==min(data))
[1] 5
```

Example 28

```
> which.max(data)
[1] 6
```

Example 29

```
> sort(data)
[1] 9 10 12 15 18 21
```

Consider the following vectors:

```
x<-c(2,5,7,11,14)
y<-c(1,4,5,7,6)
```

Example 30

```
> x+2
[1] 4 7 9 13 16
```

Example 31

```
> 3*x-1
[1] 5 14 20 32 41
```

Example 32

```
> x+y
[1] 3 9 12 18 20
```

2.4 Data frame

The function `data.frame()` creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software. Data frame is a set of data that combine vectors with the same length.

Example 33

Let say we have these 2 following vectors and want to combine them.

```
Height<-c(156, 160, 154, 148, 170, 178, 188, 163, 165, 157)
Weight<-c(48, 56, 45, 50, 68, 75, 81, 65, 70, 49)
```

Use the following command.

```
> Mydata<-data.frame(Height,Weight)
> Mydata
   Height     Weight
1     156         48
2     160         56
3     154         45
4     148         50
5     170         68
6     178         75
7     188         81
8     163         65
9     165         70
10    157         49
```

Example 34

We can rename the observations, or each line as follows:

```
> Namesresp<-c("Susan", "Mary" ,"Lily" ,"Hillary ","Mike", "John", "William",
"Cathy", "Jeff", "Julie")
> Variables<-c("Height", "Weight")
> dimnames(Mydata)<-list(Namesresp, Variables)
```

Hence, the output is:

```
> Mydata
        Height  Weight
Susan      156      48
Mary       160      56
Lily       154      45
Hillary    148      50
Mike       170      68
John       178      75
William    188      81
Cathy      163      65
Jeff       165      70
Julie      157      49
```

Example 35

If we want to add more variables, let say 'Gender' and 'Age',

```
Gender<-c("Female", "Female", "Female", "Female", "Male", "Male", "Male",
"Female", "Male", "Female")
Age<-c(30, 36, 26, 47, 31, 49, 28, 50, 42, 47)
```

Use either `data.frame()`:

```
> Mydata<-data.frame(Mydata, Gender)
> Mydata
        Height  Weight    Gender
Susan      156      48    Female
Mary       160      56    Female
Lily       154      45    Female
Hillary    148      50    Female
Mike       170      68      Male
John       178      75      Male
William    188      81      Male
Cathy      163      65    Female
Jeff       165      70      Male
Julie      157      49    Female
```

or `cbind()`:

```
> Mydata<-cbind(Mydata, Age)
> Mydata
        Height  Weight    Gender     Age
Susan      156      48    Female      30
Mary       160      56    Female      36
Lily       154      45    Female      26
Hillary    148      50    Female      47
Mike       170      68      Male      31
John       178      75      Male      49
William    188      81      Male      28
Cathy      163      65    Female      50
Jeff       165      70      Male      42
Julie      157      49    Female      47
```

Example 36

To add more observations, use `rbind()`:

```
a<-c(177,60,"Female",33)
> Mydata<-rbind(Mydata,a)
> Mydata
      Height  Weight    Gender     Age
Susan    156      48    Female      30
Mary     160      56    Female      36
Lily     154      45    Female      26
Hillary  148      50    Female      47
Mike     170      68      Male      31
John     178      75      Male      49
William  188      81      Male      28
Cathy    163      65    Female      50
Jeff     165      70      Male      42
Julie    157      49    Female      47
11       177      60    Female      33
```

2.4.1 Extract elements from Data Frame

We can selects specific row, column or cell from data frame using [,].

Example 37

If we want to select on first column from Mydata in Example 35.

```
> Mydata[,1]
[1] 156 160 154 148 170 178 188 163 165 157
```

You can also use the dollar sign operator, $ to select column:

```
> Mydata$Height
[1] 156 160 154 148 170 178 188 163 165 157
```

Example 38

If we want to select column 2 and 3 from Mydata in Example 35.

```
> Mydata[,2:3]
         Weight   Gender
Susan        48   Female
Mary         56   Female
Lily         45   Female
Hillary      50   Female
Mike         68     Male
John         75     Male
William      81     Male
Cathy        65   Female
Jeff         70     Male
Julie        49   Female
```

Example 39

If we want to select column 1 and 3 from Mydata in Example 35.

```
> Mydata[,c(1,3)]
      Height Gender
Susan    156 Female
Mary     160 Female
Lily     154 Female
Hillary  148 Female
Mike     170   Male
John     178   Male
William  188   Male
```

```
Cathy    163 Female
Jeff     165   Male
Julie    157 Female
```

Example 40

If we want to select observation in row 4 only from Mydata in Example 35.

```
> Mydata[4,]
         Height Weight Gender Age
Hillary     148     50 Female  47
```

Example 41

If we want to select observations in row 3 to 6 from Mydata in Example 35.

```
> Mydata[3:6,]
         Height Weight  Gender Age
Lily        154     45  Female  26
Hillary     148     50  Female  47
Mike        170     68    Male  31
John        178     75    Male  49
```

Example 42

If we want to select observations from specific rows, row 2, 5 and 7 from Mydata in Example 35.

```
> Mydata[c(2,5,7),]
         Height Weight  Gender Age
Mary        160     56  Female  36
Mike        170     68    Male  31
William     188     81    Male  28
```

2.4.2 Remove elements from Data frame

We can remove row, column and cell by using [,] and add '-' at front.

Example 43

If we want to remove first column from Mydata in Example 35.

```
> Mydata[,-1]
      Weight Gender Age
Susan     48 Female  30
Mary      56 Female  36
Lily      45 Female  26
Hillary   50 Female  47
Mike      68   Male  31
John      75   Male  49
William   81   Male  28
Cathy     65 Female  50
Jeff      70   Male  42
Julie     49 Female  47
```

Example 44

If we want to remove column 3 and 4 from Mydata in Example 35.

```
> Mydata[,-(3:4)]
         Height Weight
Susan       156     48
Mary        160     56
Lily        154     45
Hillary     148     50
Mike        170     68
John        178     75
```

```
William     188     81
Cathy       163     65
Jeff        165     70
Julie       157     49
```

Example 45

If we want to remove column 2 and 4 from Mydata in Example 35.

```
> Mydata[,-c(2,4)]
         Height   Gender
Susan       156   Female
Mary        160   Female
Lily        154   Female
Hillary     148   Female
Mike        170     Male
John        178     Male
William     188     Male
Cathy       163   Female
Jeff        165     Male
Julie       157   Female
```

Example 46

If we want to remove row 3 from Mydata in Example 35.

```
> Mydata[-3,]
         Height Weight  Gender Age
Susan       156     48  Female  30
Mary        160     56  Female  36
Hillary     148     50  Female  47
Mike        170     68    Male  31
John        178     75    Male  49
```

```
William     188     81    Male  28
Cathy       163     65  Female  50
Jeff        165     70    Male  42
Julie       157     49  Female  47
```

Example 47

If we want to remove row 1 to 6 from Mydata in Example 35.

```
> Mydata[-(1:6),]
         Height Weight Gender Age
William     188     81   Male  28
Cathy       163     65 Female  50
Jeff        165     70   Male  42
Julie       157     49 Female  47
```

Example 48

If we want to remove specific rows, row 3, 7 and 10 from Mydata in Example 35.

```
> Mydata[-c(3,7,10),]
         Height Weight  Gender      Age
Susan       156     48  Female      30
Mary        160     56  Female      36
Hillary     148     50  Female      47
Mike        170     68    Male      31
John        178     75    Male      49
Cathy       163     65  Female      50
Jeff        165     70    Male      42
```

If you want to arrange data according to names in alphabetical order:

```
    Mydata<-Mydata[with(Mydata, order(Namesresp)), ]
```

Change column name:

```
colnames(Mydata)[which(names(Mydata) == "Gender")] = "Sex"
```

Data frame can be saved in R and reload by using the following command:

```
save(Mydata.file="Mydata.rda")
load("Mydata.rda")
```

2.5 Matrices

Elements can be stored in matrices form. Data in matrices can be used for computation coefficients in regression analysis.

Example 49

Let 'x' represents data in a form of vector.

```
> x<-c(20, 30, 27, 25, 28, 21, 23, 24, 29, 21, 33, 19, 22, 24, 22, 20)
```

To convert data from vector data structure to matrices:

```
> m<-matrix(x,nrow=4,ncol = 4)
> m
     [,1]  [,2]  [,3]  [,4]
[1,]   20    28    29    22
[2,]   30    21    21    24
[3,]   27    23    33    22
[4,]   25    24    19    20
```

The matrix by default arrange the data column by column. If we want arrange the matrix row by row, use the following command:

```
> m<-matrix(x,nrow=4,ncol = 4,byrow = T)
> m
     [,1]  [,2]  [,3]  [,4]
```

```
[1,]    20    30    27    25
[2,]    28    21    23    24
[3,]    29    21    33    19
[4,]    22    24    22    20
```

The names of columns and rows can be assigned as follows:

column name:

```
group<-c("A","B","C","D")
```

row name:

```
obs<-c("sample 1", "sample 2", "sample 3", "sample 4")
> dimnames(m)<-list(obs,group)
> m
           A  B  C  D
sample 1  20 28 29 22
sample 2  30 21 21 24
sample 3  27 23 33 22
sample 4  25 24 19 20
```

You can use [ ] to select or remove elements in a matrix

Example 50

Select second row only from the matrix in Example 49.

```
> m[2,]
 A    B     C     D
30   21    21    24
```

Example 51

Select column 2 and 3 from the matrix in Example 49.

```
> m[2:3,]
          A  B  C  D
sample 2 30 21 21 24
sample 3 27 23 33 22
```

Example 52

Remove third column from the matrix in Example 49.

```
> m[,-3]
             A     B     D
sample 1    20    28    22
sample 2    30    21    24
sample 3    27    23    22
sample 4    25    24    20
```

Example 53

Remove column 2 and 4 from the matrix in Example 49.

```
> m[,c(2,4)]
             B     D
sample 1    28    22
sample 2    21    24
sample 3    23    22
sample 4    24    20
```

You can also use cbind( ) and rbind( ) to add column and row

Example 54

Add the following vector to a row matrix in Example 49.

```
> sample_5<-c(3,7,14,18)
> rbind(m,sample_5)
```

```
          A  B  C  D
sample 1 20 28 29 22
sample 2 30 21 21 24
sample 3 27 23 33 22
sample 4 25 24 19 20
sample_5  3  7 14 18
```

Example 55

Add the following vector to a column matrix in Example 49.

```
> E<-c(13,17,20,21)
> cbind(m,E)
          A  B  C  D  E
sample 1 20 28 29 22 13
sample 2 30 21 21 24 17
sample 3 27 23 33 22 20
sample 4 25 24 19 20 21
```

You can also create a matrix with the value of same elements, for example all elements are value 1

```
> z<-matrix(1,2,3)
> z
     [,1]  [,2]  [,3]
[1,]    1     1     1
[2,]    1     1     1
```

2.6 List

A list is a one-dimensional heterogeneous data structure. So it is indexed like a vector with a single integer value, but each element can contain an element of any type.

Example 56

```
> a<-c(T,F,F,T,T)
> b<-matrix(c(1,3,5,8),ncol=2)
> c<-data.frame(id=101:103,age=c(40,38,33))
> abc<-list(a,b,c)
> abc
[[1]]
[1] TRUE FALSE FALSE TRUE TRUE

[[2]]
     [,1]  [,2]
[1,]    1     5
[2,]    3     8

[[3]]
   id age
1 101  40
2 102  38
3 103  33
```

Lists can be subset using square brackets `[]`. The `[]` syntax returns a list, while the `[[]]` returns an element of a list.

Using `[]` to select object in list:

Example 57

```
> abc[1]
[[1]]
[1]   TRUE FALSE FALSE  TRUE  TRUE
```

```
> abc[c(1,3)]
[[1]]
TRUE FALSE FALSE  TRUE  TRUE

[[2]]
   id age
1 101  40
2 102  38
3 103  33
```

2.7 Array

Arrays are the R data objects which can store matrices in more than one dimensions. For example if we create an array of dimension (2, 3, 4) then it creates 4 rectangular matrices each with 2 rows and 3 columns. Arrays can store only numeric data type. An array is created using the array() function. It takes vectors as input and uses the values in the dim parameter to create an array.

Example 58

The following example creates an array of two 3x3 matrices each with 3 rows and 3 columns.

```
> a <- c(5,9,3)
> b <- c(10,11,12,13,14,15)
> ab <- array(c(a,b),dim = c(3,3,2))
> ab

, , 1
      [,1]  [,2]  [,3]
[1,]    5    10    13
[2,]    9    11    14
[3,]    3    12    15
```

```
, , 2

      [,1]  [,2]  [,3]
[1,]     5    10    13
[2,]     9    11    14
[3,]     3    12    15
```

We can give names to the rows, columns and matrices in the array by using the dimnames parameter.

Example 59

Refer to Example 58.

```
> column.names <- c("COL1","COL2","COL3")
> row.names <- c("ROW1","ROW2","ROW3")
> matrix.names <- c("Matrix1","Matrix2")
> ab <- array(c(a,b),dim = c(3,3,2),dimnames = list(row.names,column.names,
+     matrix.names))
> ab
, , Matrix1

      COL1  COL2  COL3
ROW1     5    10    13
ROW2     9    11    14
ROW3     3    12    15

, , Matrix2

      COL1 COL2 COL3
ROW1     5   10   13
ROW2     9   11   14
ROW3     3   12   15
```

Extract the element from array:

Example 60

Refer to Example 58.

```
> ab[3,,2]
COL1 COL2 COL3
   3   12   15
> ab[3,,2]
COL1 COL2 COL3
   3   12   15
> ab[1,3,1] [1] 13
> ab[,,2]
      COL1  COL2  COL3
ROW1     5    10    13
ROW2     9    11    14
ROW3     3    12    15
```