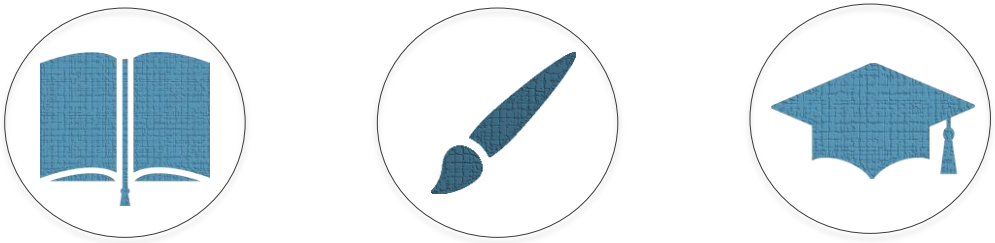


Data Science Using R

Lesson02–Fundamentals of R

Objective

After completing this lesson you will be able to:

- Import data files into an R system
- Perform basic data manipulation

Reading Data into R

Data can be imported to R in multiple ways. Two commonly used ways are:

From a csv file

```
myData <-
read.table(file.choose(),hea
der=TRUE, sep=",")
# first row of the csv being
#read, contains variable
#names, comma is separator.
```

From an excel file

```
library(xlsx)
myData <-
read.xlsx(file.choose(),head
er = TRUE, sheetIndex = 1)
# first row of the excel
#being read contains
#variable names.
```

The best way to read an Excel file is to export it to a comma delimited file and import it using read.table() or read.csv(). gdata/readxl another library to import more than 10 Mb file.

Creating and Renaming Variables

Creating and renaming variables are two important aspects in R.

Creating New Variables

- New variables are created by using the assignment operator '<-'.

```
mydata <- list(x1= 2, x2=5)
mydata$sum <- mydata$x1 +
mydata$x2
mydata$mean <- (mydata$x1 +
mydata$x2)/2

attach(mydata)
mydata$sum <- x1 + x2
mydata$mean <- (x1 + x2)/2
detach(mydata)
```

Renaming variables

- Variables can be renamed programmatically or interactively

```
# rename interactively
t <- list(name= "roy",
age=30, gender="M")
fix(t) # results are saved on
close

# rename programmatically
library(reshape)
mydata <- rename(mydata,
c(oldname="newname"))
```

Understanding Data Types—Vectors and Matrix

There is a fine distinction between vectors and matrix.

- Vectors: All elements must be of the same type. **Example:**

```
name <- c("Mike", "Lucy", "John") #vector of string
or character
age <- c(20, 25, 30) #vector of integers
```

- Matrix: A special kind of vector with two additional attributes i.e. the number of rows and the number of columns. **Example**:

```
x <- matrix(c(1,2,3,4), nrow=2, ncol=2)
```

Understanding Data Types—List

List is an ordered collection of objects which allows to gather a variety of (possibly unrelated) objects under one name.

- **Example** of a list with 4 components: A string, a numeric vector, a matrix and a scaler.

```
w <- list(name="Fred", mynumbers=a, mymatrix=y,
age=5.3)
```

- **Example** of a list containing four vectors:

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc", "dd", "ee")
b = c(TRUE, FALSE, TRUE, FALSE, FALSE)
x = list(n, s, b, 3)   # x is a list which contains
copies of n, s, b
```

Understanding Data Types—Factors

A 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) and 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 1s and 30 2s and
associates. 1=female, 2=male.

#R now treats gender as a nominal variable
summary(gender)
```

- The order of the levels can be set using the levels argument to factor(). This can be important in linear modelling.

Understanding Data Types—DataFrames

Data frames are used to store tabular information.

- They are represented as a special type of list, where every element of the list has to have the same length
- Each element of the list can be thought of as a column and the length of each element of the list is the number of rows
- Unlike matrices, data frames can store different classes of objects in each column (just like lists); matrices must have every element of the same class
- Data frames also have a special attribute called row.names
- Data frames are usually created by calling read.table() or read.csv()
- Can be converted to a matrix by calling data.matrix()

Data Sub setting in R

R has robust subsetting feature which can be used for selecting or excluding variables from a dataset.

- Below are the examples for selecting or excluding variables

Examples for selection:

```
# select variables mpg, cyl,
disp from mtcars dataset
myvars <- c("mpg", "cyl",
"disp")
newdata <- mtcars[myvars]


# select 1st and 7th through
11th variables
newdata <- mtcars[c(1,7:11)]
```

Examples for exclusion:

```
# exclude variables mpg, cyl,
disp from mtcars dataset
myvars <- names(mtcars) %in%
c("mpg", "cyl", "disp")
newdata <- mtcars[!myvars]

# exclude 1st and 7th
variables
newdata <- mtcars[c(-1,-7)]
```

Data Sub setting in R

Subsetting feature in R can be used for selecting or excluding observations as well.

- Below are the examples for selecting observations from a dataset:

Examples for selection:

```
# first 5 observations of
#mtcars dataset across all
#variables
newdata <- mtcars[1:5,]

# select observations of
#mtcars dataset based on
#condition
newdata <-
mtcars[which(mtcars$hp >100 &
mtcars$cyl > 4),]
```

Examples for selection using subset:

```
#select hp and cyl from
#mtcars
newdata <- subset(mtcars, hp
> 100 | cyl < 10,
select=c(hp, cyl))]

# select 1st through 6th
#variable from mtcars dataset
newdata <- subset(mtcars, hp
> 100 | cyl < 10,
select=c(1:6))
```

Data Manipulation—Sort

To sort a dataframe in R, use the order( ) function. By default, sorting is ASCENDING. Prepend the sorting variable by a minus sign to indicate the DESCENDING order.

***Example**: Sorting examples using the mtcars dataset*

```
data(mtcars)
# sort by mpg
newdata = mtcars[order(mtcars$mpg),]
#sort by mpg and cyl
newdata <- mtcars[order(mtcars$mpg, mtcars$cyl),]
#sort by mpg (ascending) and cyl (descending)
newdata <- mtcars[order(mtcars$mpg, -mtcars$cyl),]
```

Data Manipulation—Merge

To merge two DataFrames horizontally, use the merge function. Typically, DataFrames are joined by one or more common key variables. Merge two data frames by ID

```
total <- merge(dataframeA,dataframeB,by="ID")
```

Delete the extra variables in dataframeA or

```
total <- merge(dataframeA,dataframeB,by=c("ID","Country"))
```

Create the additional variables in dataframeB and set them to NA (missing) before joining

```
total <- rbind(dataframeA, dataframeB)
```

For merging vertically, two dataframes must have same variables. If not then:

- Merge two data frames by ID and Country
- To join two dataframes (datasets) vertically, use the rbind function.

Data Manipulation—Aggregate

Aggregate function is used to summarize the data by a variable

```
# aggregate dataframe iris by species and return
means for numeric variables
attach(iris)
aggdata <-aggregate(iris, by=list(iris$Species),
FUN=mean, na.rm=TRUE)
print(aggdata)
```

When using the aggregate() function, the **'by'** variables must be in a list (even if there is only one). The function can be built-in or user provided.

Data Manipulation—Aggregate

Other ways to aggregate data is by using summarize() function available in the Hmisc package.

- Example 1:

```
summarize(iris$Sepal.Length,iris$Species,FUN=mean)
```

- Example 2:

```
set.seed(1)
temperature <- rnorm(300, 70, 10)
month <- sample(1:12, 300, TRUE)
year  <- sample(2000:2001, 300, TRUE)
g <-
function(x)c(Mean=mean(x,na.rm=TRUE),Median=median(x,na.rm=
TRUE))
summarize(temperature, month, g)
```

Data Manipulation—Data Conversion

- Data conversion in R can be done using the pre-defined functions.
- For example, adding a character string to a numeric vector converts all the elements in the vector to character.

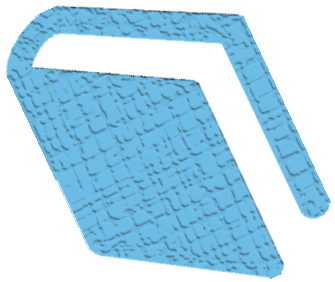
Some useful functions for type conversion:

is.numeric(), is.character(), is.vector(), is.matrix(), is.data.frame(), as.numeric(), as.character(), as.vector(), as.matrix(), as.data.frame()

This demo will show the use of data operations in R using Rstudio.

Summary

Summary of the topics covered in this lesson:



- Vectors, Matrix, List, Factors and Dataframes are different data types which can be used to store datasets.
- Read, Sort, Merge, Aggregate are some of the basic data manipulation techniques which can be helpful in data analysis.
- R has robust subsetting feature which can be used for selecting or excluding variables or observations from a dataset.

QUIZ TIME

Quiz Question 1

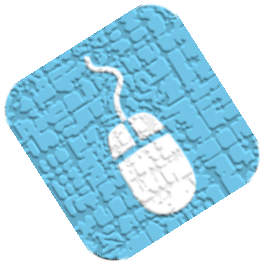
Quiz 1	What will be the data type of age in the following code: age <- c(20, 25, 30). *Select all that apply.*

a. *Vector of integers*

b. *List*

c. *DataFrame*

d. *Matrix*



Quiz Question 1

Quiz 1	What will be the data type of age in the following code: age <- c(20, 25, 30). *Select all that apply.*

a. *Vector of integers*

b. *List*

c. *DataFrame*

d. *Matrix*

Correct answer is: age is a vector of integers.

a

Quiz Question 2

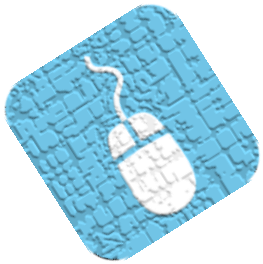
Quiz 2	Which command when typed in console will show the inbuilt dataset in R?

a. *dataset()*

b. *dataframe()*

c. *data()*

d. *showData()*



Quiz Question 2

Quiz 2 Which command when typed in console will show the inbuilt dataset in R?

a. *dataset()*

b. *dataframe()*

c. *data()*

d. *showData()*

Correct answer is: *c*

data() shows the inbuilt dataset in R when typed in console; others are not commands to show built in dataset in R.

Quiz Question 3

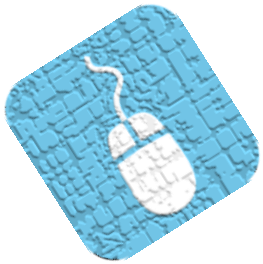
Quiz 3 What will the following code give as output: *newdata <- iris[c(-1,-2)]*

a. *Will create a variable newdata with the first two columns included.*

b. *Will create a variable newdata with the first two columns excluded.*

c. *Will create a variable iris with the first two columns excluded.*

d. *Will create a variable iris with the first two columns included.*



Quiz Question 3

Quiz 3 What will the following code give as output: *newdata <- iris[c(-1,-2)]*

a. *Will create a variable newdata with the first two columns included.*

b. *Will create a variable newdata with the first two columns excluded.*

c. *Will create a variable iris with the first two columns excluded.*

d. *Will create a variable iris with the first two columns included.*

Correct answer is: ***b***

iris dataset is inbuilt in R and the above command will exclude the first two columns.

End of Lesson02–Fundamentals of R

R is an official part of the Free Software Foundation's GNU project.
RStudio and Shiny are affiliated projects of the Foundation for Open Access Statistics.