John Hopkins Data Science with R

Mandar Bhosale

7/15/2019

Introduction

This is our first project in R. The learning is acquired through **John Hopkins University Mooc** *Data Science with R.*

Contents

We have so far learned how to install R and Rstudio. We took a brief tour of the Rstudiio GUI. Other things covered in the Mooc are:
- Version Control using Git - Visit my GitHub Profile
- R Markdown
- Types of Analysis
- Experimental Design or DOE
- R Documentation

In order to get a brief understanding of the dataset use `summary(dataset name)`

```
print("Hello World")
```

```
## [1] "Hello World"
```

Descriptive Analysis

Summary of the Data Ex: Age distribution in US

Exloratory Analysis

Examine the Data and find relationships that weren't known before Correlation does not imply causation

Useful for discovering new conneciton Its just tells that a relation exists, but not the cause

Inferential Analysis

Small sample of data us used to say something about the population at large. Smaple data plays an important role. The data should be related. Ex: Sample data from US population can be used to infer the life expectancy of the pupoulation in US.

Predictive Analysis

Using current and historic data to predict about the future for the query in hand. You should be using the right variables and also understand that *correlation does not lead to causation.*

Causal Analysis

What happens to one variable when we change other.

Basic Setup

Run RStudio in your Windows/Mac System.

Type `getwd()` in thr R console to check your current working directory. The working directory is where all data files, R scipts are stored to be used later. Using "File" tab -> "..." -> choose folder to open -> More -> Set as working directory.
Or Got to the folder -> Copy the folder path -> in RStudio console -> type `setwd(pathname)` -> Run.

R Programming

Data Types :
- Interger
- Number
- Character
- Logical

Objects in R
- Vector
- Matrix
- Dataframe
- List

Assign variables Varibales store values. Eg: x <- 1, will store 1 in x. Values stored in a variables can be printed on the console using the variables by typing `x`. Other way is using `print(x)` function.

`class()` and `mode()` fucntion can help us to understand what type of data is stored in the variables.

```
a <- 1
class(a)
```

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

```
mode(a)
```

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

```
b <- "R programming"
class(b)
```

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

```
mode(b)
```

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

Function `c()` helps to concatenate values.

```
x = c(0.5, 0.6)
x = c(1,2,3)
x = c(T,F)
x = c("a","b","c")
```

Vector Funciton `vector()` creates empty vector with specified type.

```
z <- vector("numeric", length = 10)
y <- vector("character", length = 10)
```

Attributes

- Matrix has dimnames and names

```
test_mat <- matrix(1:12, nrow = 3, byrow = T)
rownames(test_mat) <- c("1","2","3")
colnames(test_mat) <- c("A","B","C","D")
test_mat
```

```
##   A  B  C  D
## 1 1  2  3  4
## 2 5  6  7  8
## 3 9 10 11 12
```

```
dimnames(test_mat)
```

```
## [[1]]
## [1] "1" "2" "3"
##
## [[2]]
## [1] "A" "B" "C" "D"
```

- Length of the vector

Explicit Coercion which also means: type conversion `as. *` is used to convert a object from one to other. Ex: `x <- 0.6 | as.numeric(x)`.

Lists

```
# Multi type vector list
xlist <- list(1,"a", TRUE, 1.1)
xlist
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
```

```
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1.1
```

```
# List can have other list, matrix and vector together
ylist <- list(xlist, test_mat)
ylist
```

```
## [[1]]
## [[1]][[1]]
## [1] 1
##
## [[1]][[2]]
## [1] "a"
##
## [[1]][[3]]
## [1] TRUE
##
## [[1]][[4]]
## [1] 1.1
##
##
## [[2]]
##   A  B  C  D
## 1 1  2  3  4
## 2 5  6  7  8
## 3 9 10 11 12
```

Matrices

`dim()` prints the dimensions in the matrix row | columns `attributes` prints the dimensions aswell the names of the rows and columns

We can create a matrix from a vector. consider vector `man`

```
#dim(man) <- c(2,5) # this converts vector 'man' to a matrix 'man', using the dimmensions provided.
#man
```

Other function in matrix are - cbind : Adds columns to a matrix or can be used to add vectors to make a matrix - rbind : Adds rows

`cb_mat <- cbind(x,y)` or `rb_mat <- rbind(x,y)`

Factors

Represents categorical data : Order and Unordered - Ordered : Male | Female
- Unordered : Low<Med<High
Factors have labels

`xfact <- factor(c("yes","no","no","yes","yes"))`

```r
temp <- c("L","M","H","L","L","M")  # Create a vector
temp_fact   <- factor(temp, levels = c("L","M","H"), ordered = T) # Factor function converts temp varia
temp_fact
```

```
## [1] L M H L L M
## Levels: L < M < H
```

Dealing with missing values

NA or NaN values NaN are for undefined mathematical operations

using `is.na()` or `is.nan()` helps to find the missing values in the vector

```r
xmiss <- c(1,2,NA,3,4,NaN)
is.na(xmiss)
```

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

```r
is.nan(xmiss)
```

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

The output is always logical and gives TRUE when there is a missing value at a location.

Data Frames

Used to store tabular data. Is like a list and all list have the same length. Column have varied data type. Attributes are `row.name()` We can import data from our system using `read.table()` or `read.csv()`.

To create dataframe : xframe <- data.frame(1:10)

We can create dataframe from vectors.

```r
# Create varied vectors
ID <- c(1:10)
Name <- c("a","b","c","d","e","f","g","h","i","j")
xdata <- data.frame(ID,Name)
xdata
```

```
##    ID Name
## 1   1    a
## 2   2    b
## 3   3    c
## 4   4    d
## 5   5    e
## 6   6    f
## 7   7    g
## 8   8    h
## 9   9    i
## 10 10    j
```

This creates a dataframe from the vectors and uses the vector name as column name. Other used functions : - head() : Gives top 6 rows from a dataframe
- tail() : GIves last 6 rows from the dataframe `xdata$ID` will print only the ID column from the dataframe