R Programming Notes for Data Science

dea

2021-12-30

Contents

Chapter 1

Acknowledgement

I have been compiling notes and tips on R programming from everywhere. Most of these notes coming from Data Science Specialization Course from Courseara.

Chapter 2

R Programming

R is a functional programming language. It is most popular among academia and Data Sciencetists.

2.1 General Information

Cleaning the environment

```
rm(list = ls())
```

Installing a package

```
install.packages("ggplot2")  # install
detach(ggplot2, unload = TRUE)  # removing the library
```

Browsing help on packages

```
browseVignettes("ggplot2")
```

```
wd <- getwd()
wd
```

```
## [1] "/Users/d842a922/Desktop/R/_my_R_book"
```

```
# listing environment objects
ls()
```

```
## [1] "wd"
```

```
# listing files in the working directory
files <- list.files()
head(files)
```

```
## [1] "_book"           "_bookdown_files" "_bookdown.yml"   "_build.sh"
## [5] "_deploy.sh"      "_my_R_book"
```

```
# listing files in the working directory
files2 <- dir()
head(files2)
```

```
## [1] "_book"           "_bookdown_files" "_bookdown.yml"   "_build.sh"
## [5] "_deploy.sh"      "_my_R_book"
```

```
dir( pattern = "^L", full.names = F, ignore.case = T )
```

```
old.dir <- getwd()

# creating a folder in the directory
dir.create("testdir")
```

```
setwd("testdir")
```

```
#create a file
file.create("testdir/mytest.R")
```

```
file.exists("testdir/mytest.R")
```

```
file.info("testdir/mytest.R")
```

```
# to list files in path
myfiles <- list.files(path="testdir", pattern = "[2]")
head(myfiles)
```

```
#rename filename from to
file.rename("testdir/mytest.R", "testdir/mytest4.R")

list.files(path="testdir", pattern = "[4]")
```

```
# interactive
 file1 <- file.choose()
```

```
# copy file from to
file.copy("testdir/mytest2.R", "testdir/mytest3.R")
```

```
myfiles
```

```
class(myfiles)   # character vector
```

```
myfiles[1]
```

```
setwd("testdir")

file.copy(myfiles[1], "deneme2.xlsx")
```

```
# assign a name to a file path (exist or not)
path1 <- file.path("mytest3.R")
```

```
path1
```

directory creation: testdir/deneme3

```
dir.create(file.path("testdir", "deneme3"), recursive = TRUE )
```

```
# assign a name to a folder path (olmak zorunda degiller)
abc <- file.path("testdir", "deneme")
abc
```

double colon

There may be multiple functions with the same name in multiple packages. The double colon operator allows you to specify the specific function you want:

```
dplyr::filter()
```

```
str(file.path)
args((file.path))
```

```
# then you can use variable names directly

attach(mtcars)
```

Chapter 3

Create sequence of numbers

```
a <- seq(from = 5, to = 14, by = 2)
a
```

```
## [1]  5  7  9 11 13
```

```
# generates integer sequence of length(along.with)
seq(along.with = 1:12)
```

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

```
seq_along(1:15)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```
seq(length.out = 4)
```

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

```
seq_len(10)
```

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

```
a = seq(10, 20)
b = seq(10, 30, by =2)
```

3.0.1 – %in%

This creates a logical vector, where testing each element in vector "a" if ever matches any element in vector "b"

```
c <- a %in% b
c
```

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

3.0.2 which()

which(x, arr.ind = FALSE, useNames = TRUE)

input a logical vector returns location index of true values

```
which(c)
```

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

```
d <- LETTERS[1:10]
d
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

```
e <- LETTERS[5:10]
e
```

```
## [1] "E" "F" "G" "H" "I" "J"
```

```
d %in% e
```

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

```
which(d %in% e)   ## location of TRUE values of vector d (matches vector e)
```

```
## [1]  5  6  7  8  9 10
```

```
g <- c("c", "d", "e", "k", "l", "m")
h <- c("a", "b", "c", "d", "e", "d")
```

```
i <- g %in% h
i
```

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

```
which(g == h)
```

```
## integer(0)
```

```
# subsetting property

g[g %in% h]
```

```
## [1] "c" "d" "e"
```

```
which( (1:12) %% 2 == 0, arr.ind = F)   ## location in the array (1:12)
```

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

3.0.3 Where is the min, max, first true/false?

which.min() which.max()

```
a = c(2, 4, 1, 7, 9, 1, 3, 5, 9, NA, "4")
a
```

```
##  [1] "2" "4" "1" "7" "9" "1" "3" "5" "9" NA  "4"
```

```
which.min(a > 4)
```

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

```
which.max(a)
```

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

```
a[which.max(a)]
```

```
## [1] "9"
```

match(a, b)

match: An integer vector giving the position in table of the first match if there is a match, otherwise nomatch.

min(which(x == a))

```
a = 1:15
b = seq(1, 20, by=3)

match(a, b)  ## returns location of true values of vector a
```

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

```
a %in% b
```

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

dataframe

```
df <- cars
head(df)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

```
# test if value 5 in speed column
5 %in% df$speed
```

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

```
# create a dataframe
df2 <- data.frame(Type = c("fruit", "fruit","fruit", "veggie","veggie"),
                  Name = c("red apple", "green apple", "red apple", "green apple" ,"red

df2
```

```
##     Type        Name  Color
## 1  fruit   red apple   <NA>
## 2  fruit green apple    red
## 3  fruit   red apple   blue
## 4 veggie green apple yellow
## 5 veggie   red apple    red
```

```
df2 <- within(df2,
              { newcol = "No"
              newcol[Type %in% c("fruit")] = "No"
              newcol[Name %in% c( "green apple")] = "Yes"
})

head(df2, 3)
```

```
##    Type        Name Color newcol
## 1 fruit   red apple  <NA>     No
## 2 fruit green apple   red    Yes
## 3 fruit   red apple  blue     No
```

subsetting

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df3 <- c("home", "veggie", "fruit")

df2 %>%
    filter(df2$Type %in% df3)
```

```
##     Type        Name  Color newcol
```

```
## 1  fruit   red apple   <NA>     No
## 2  fruit green apple    red    Yes
## 3  fruit   red apple   blue     No
## 4 veggie green apple yellow    Yes
## 5 veggie   red apple    red   <NA>
```

dropping columns

```
df2[, !(colnames(df2) %in% c("Name", "Color")) ]
```

```
##     Type newcol
## 1  fruit     No
## 2  fruit    Yes
## 3  fruit     No
## 4 veggie    Yes
## 5 veggie   <NA>
```

selecting columns

```
df2[, (colnames(df2) %in% c("Name", "Color")) ]
```

```
##          Name  Color
## 1   red apple   <NA>
## 2 green apple    red
## 3   red apple   blue
## 4 green apple yellow
## 5   red apple    red
```

creating custom operator

```
`%notin%` <- Negate(`%in%`)

numbs <- rep(seq(3), 4)
numbs
```

```
##  [1] 1 2 3 1 2 3 1 2 3 1 2 3
```

```
4 %notin% numbs
```

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

Chapter 4

Logic statements

TRUE vs FALSE

```
TRUE == TRUE
```

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

```
(FALSE == TRUE) == FALSE
```

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

```
6==7
```

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

```
6<=6
```

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

```
4 != 5
```

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

```
!(5 == 71)
```

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

```
TRUE & TRUE
```

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

```
FALSE & FALSE
```

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

```
TRUE & c(TRUE, FALSE, FALSE)
```

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

equivalent statement as

```
c(TRUE, TRUE, TRUE) & c(TRUE, FALSE, FALSE)
```

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

Chapter 5

be careful

```
TRUE && c(TRUE, FALSE, FALSE)
```

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

In this case, the left operand is only evaluated with the first member of the right operand (the vector). The rest of the elements in the vector aren't evaluated at all in this expression.

```
TRUE | FALSE
```

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

```
TRUE | c(TRUE, FALSE, FALSE)
```

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

```
TRUE || c(TRUE, FALSE, FALSE)
```

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

```
FALSE && 6 >= 6 || 7 >= 8 || 50 <= 49.5
```

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

```
!(8 > 4) ||  5 == 5.0 && 7.8 >= 7.79
```

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

```
TRUE && FALSE || 9 >= 4 && 3 < 6
```

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

```
99.99 > 100 || 45 < 7.3 || 4 != 4.0
```

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

```
isTRUE(6>4)
```

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

```
identical('twins', 'twins')
```

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

The xor() function stands for exclusive OR. If one argument evaluates to TRUE and one argument evaluates to FALSE, then this function will return TRUE, otherwise it will return FALSE.

```
xor(5 == 6, !FALSE)
```

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

```
xor(T, T)
```

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

```
xor(F, F)
```

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

```
xor(identical(xor, 'xor'), 7 == 7.0)
```

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

```
xor(4 >= 9, 8 != 8.0)
```

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

```
ints <- sample(10)
```

```
ints > 5
```

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

The which() function takes a logical vector as an argument and returns the indices of the vector that are TRUE.

```
which(c(TRUE, FALSE, TRUE))
```

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

```
x <- ints>7

which(x)
```

```
## [1]  5  6 10
```

The any() function will return TRUE if one or more of the elements in the logical vector is TRUE.

```
any(ints<0)
```

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

The all() function will return TRUE if every element in the logical vector is TRUE.

```
all(ints>0)
```

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

```
any(ints == 10)
```

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

```
all(c(TRUE, FALSE, TRUE))
```

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