28-12-17-R-Basics-Intro

December 29, 2017

0.1 Softwares

- Base R GUI
- R-Studio

0.2 Installation

0.2.1 Base R GUI

https://cran.r-project.org/

0.2.2 R-Studio

https://www.rstudio.com/

0.3 Syntax

- R programming is case sensitive, which means for example a letter 'm' is not equal to 'M'
- Comments can be written by using # at the beginning of the statement
- We can Run single/multiple lines of code. This is much more exible because of it's interpreted nature
- Script les are saved with .R extension, workspace saved with .RData, code history saved with .Rhistory
- R has two assignment operators which can assign values into objects. less than symbol followed by the hyphen (<-) or Equal operator (=)

0.4 Basic Functions

In [ ]:
```
# sqrt() function computes the square root of a numeric vector.

    # sqrt(x)

    # x: numeric or complex vector, array
    sqrt(9)
    sqrt(-1)
    sqrt(3+5i)
    sqrt(c(4,9,16))
```

In [ ]:
```r
# rep() function replicates the values in x.

# rep(x, ...)
# rep.int(x, times)

# x: numeric vector
# ...: arguments including times (default = 1), length.out, each (each elements how ma

x <- rep(1:5)

# Repeat 1 -5 two times:
x <- rep(1:5,2)
```

In [ ]:
```r
#seq

# seq() function generates a sequence of numbers.

# seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
#     length.out = NULL, along.with = NULL, ...)

#  from, to: begin and end number of the sequence
#  by: step, increment (Default is 1)
#  length.out: length of the sequence
# ...

# Generate a sequence from -6 to 7:
x <- seq(-6,7)



# From -6 till 7, step=2:
x <- seq(-6,7,by=2)
x


# Let's try smaller step:
x <- seq(-2,2,by=0.3)
x
```

In [ ]:
```r
# abs() function computes the absolute value of numeric data.
abs(x)
# x: Numeric value, array or vector
abs(-1)
abs(20)
```

In [ ]:
```r
factorial() function computes the factorial of a number.

factorial(x)
```

```
        # x: numeric vector


        factorial(2)  #2 ⅇ 1


        factorial(1)   #1 ⅇ 1


         factorial(3)  #3 ⅇ 2 ⅇ 1


         factorial(4)  #4 ⅇ 3 ⅇ 2 ⅇ 1


         factorial(c(4,3,2))
```

In [ ]:
```
        # log10() function computes base 10 logarithm.

        log10(x)

        # x: numeric vector


        log10(100)


        x <- c(100,1000, 10000)
         log10(x)
```

In [ ]:
```
        # toupper() function converts a string to its upper case.

        toupper(x)

        # x: character vector


        x <- c("Green", "Red", "Black")
        toupper(x)

        # tolower() function converter string to its lower case.

        tolower(x)

        # x: character vector
```

```
tolower("EndMemo R Tutorial")
```

```
x <- c("Green", "Red", "Black")
tolower(x)
```

In [ ]:
```
# strsplit() function splits the elements of a character vector x into substrings acco

# strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
y <- strsplit(x,"t")
y

unlist(y)
```

In [ ]:
```
# Paste

# Concatenate vectors after converting to character.


# Usage:

# paste(..., sep = " ", collapse = NULL)

# Arguments:

# ...:        one or more R objects, to be converted to character vectors.
# sep:        a character string to separate the terms.
# collapse:       an optional character string to separate the results.
```

In [ ]:
```
# substr() function extract or replace substrings in a character vector.

# substr(x, start, stop)
# substring(text, first, last = 1000000L)
# substr(x, start, stop) <- value
# substring(text, first, last = 1000000L) <- value

# x,text: character vector
# start, first: integer, the first element to be replaced
# stop, last: integer, the last element to be replaced
# value: character vector, recycled if necessary


substr("tutorial",2,3)


x <- c("green","red","blue")
substr(x,2,3)
```

0.5 Help and Documentation

0.5.1 help()

```
In [6]: # To access documentation for the standard lm (linear model) function
        help(lm)
        #or
        help("lm")
        help(package="MASS")
```

0.5.2 ?

```
In [7]: ?lm
        ?"lm"
```

```
In [ ]: # To access help for a function in a package thats not currently loaded
        help(package="MASS")
        # to get some examples from a function
        example(lm)
```

0.5.3 apropos

```
In [9]: apropos("^glm")
```

1. 'glm' 2. 'glm.control' 3. 'glm.fit'

0.5.4 help.search() and ??

```
In [10]: help.search("^glm")
```

0.5.5 help.start()

```
In [11]: # Manuals and Materials for learning
         help.start()
```

```
If nothing happens, you should open
'http://127.0.0.1:18431/doc/html/index.html' yourself
```

0.5.6 Vignettes and Code Demonstrations

- browseVignettes()
- vignette()
- demo()

```
In [14]: # browseVignettes(package=package-name)
         browseVignettes(package="survival")
         # vignette("vignette-name")
         vignette("timedep")
         vignette("timedep", package="survival")
         # The command demo() lists all demos for all packages
         demo(package="stats")
```

```
In [ ]: # Browse for Vignettes
        browseVignettes()
```