R프로그램 기초: 설치 및 객체

Jinseog Kim
Dongguk University
jskim1986@gmail.com

2018-03-06

Contents

1 R소개 및 설치

1.1 R이란?

1. 데이터의 처리, 통계 계산 및 분석, 그래픽스를 위한 프로그래밍 언어
2. R의 모태 - S 언어 (AT&T Bell 랩의 John Chambers 등이 개발)
3. R의 개발: 뉴질랜드 Auckland 대학교 교수인 R. Ihaka 와 R. Gentleman
4. 현재, R Development Core Team이 개발 및 관리하고 있음
5. 인터프리터/스크립트 언어의 일종이며 무료

6. R은 기본(코어)패키지와 9천개 이상의 추가 패키지들로 구성됨

see https://www.r-project.org/about.html

1.2 R프로그램 작성 및 실행을 위한 개발 환경

- R-GUI: R의 기본 UI
- RStudio: R프로그램 작성 및 실행을 위한 통합 개발환경 프로그램(IDE)
  - rstudio.com에서 개발(무료)

1.3 RStudio Desktop & RStudio Server for Linux

- R프로그램 작성 및 실행을 위한 개발 환경프로그램(IDE)

1. RStudio: cross-platform open source IDE
2. RStudio-server: 리눅스 등 서버환경에서 다중 유저들에 대한 개발환경을 제공함

1.4 R의 설치

- https://www.r-project.org/
- https://www.rstudio.com/products/rstudio/download/

1.5 Rstudio의 설치

- 개발자를 위한 통합환경(IDE)

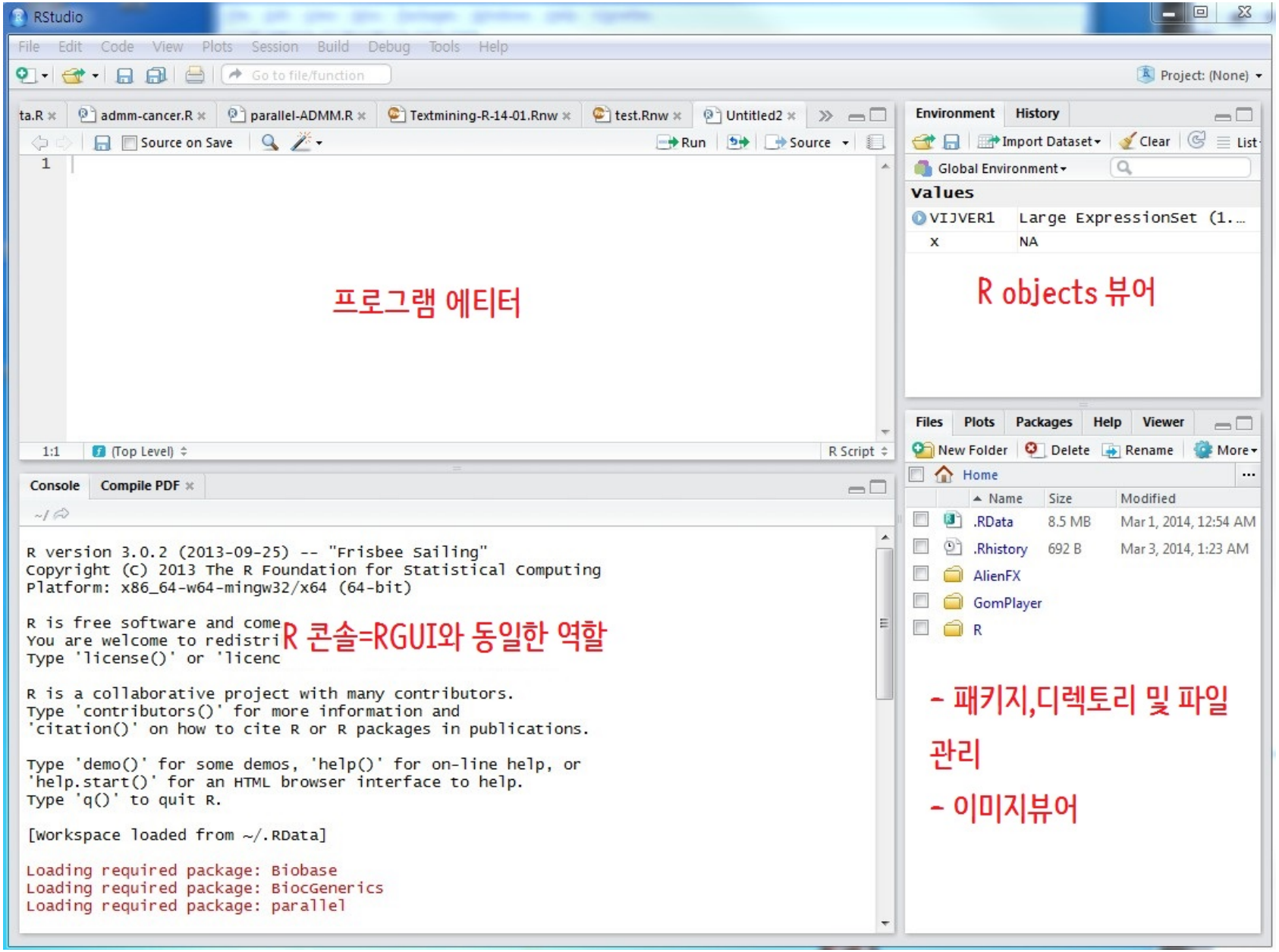

Figure 1:6RSTUDIO

2 R 시작

2.1 Starting R

- R consol with RGui or Rstudio

```
x <- 1
x
```

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

```
x <- "hello"
x
```

```
## [1] "hello"
```

2.2 R의 기초 용어 및 유틸리티

- object: R에서 자료, 함수, 연산자등 객체, 메모리에 저장
- ls() : 객체들의 리스트

```
x <- 1
y <- 1:10
ls()
```

```
## [1] "x" "y"
```

- rm(): R 객체를 삭제

```
rm(x,y)
ls()
```

```
## character(0)
```

- R 작업공간(R workspace): R을 이용하여 작업하는 동안 만들어지는 객체(object)들의 모임(collection)

- help(): R 객체들에 대한 도움말을 출력, 대신에 ?객체명을 사용할 수 있음

```
help(ls)
?ls
```

- 작업 디렉토리(working directory) 탐색 및 지정

```
  getwd()
```

```
## [1] "/var/www/html/lectures/lectures/2018-1/R"
```

```
  #setwd("D:/share/lectures/R-note")
```

- 작업공간(workspace) 저장, 작업 디렉토리에는 .RData라는 파일이 생성된다.

```
  save.image()
```

- R패키지: R의 확장기능 이용 ⇐ R패키지 추가 설치

- search()는 설치된 R패키지들을 확인하는 명령

```
  search()
```

```
##  [1] ".GlobalEnv"        "package:knitr"     "package:stats"
##  [4] "package:graphics"  "package:grDevices" "package:utils"
##  [7] "package:datasets"  "package:methods"   "Autoloads"
## [10] "package:base"
```

- library(): R에 설치된 모든 패키지 및 설명

```
library()
```

- library(package_name): 패키지를 현재 R세션으로 로딩

```
library(MASS) # MASS 패키지를 로드
```

- install.packages(): R에 새로운 패키지 설치

```
install.packages("stringr")
```

- help(), '?' : 함수 및 객체에 대한 도움말

```
help("ls")
?ls
```

2.3 기타

- 객체명 표시방법: 알파벳문자, '_','.', 숫자의 조합
- 라인 코멘트(주석문) : # comments

```
a <- 1
b_1 <- 10 # <-는 left assignment
```

3 R 객체(R objects)

 R 객체에는 아래와 같은 종류들이 있음

1. atomic(상수)
2. vector(벡터)
3. matrix(행렬)
4. list(리스트)
5. data.frame(데이터프레임)
6. function(함수)
7. operator(연산자) …

R 객체 중 데이터 객체 : atomic, vector, matrix, data.frame ⇒ 데이터 객체 (data object)

3.1 atomic(상수) 데이터 객체

 atomic(상수) 데이터 객체의 유형(type)

 atomic(상수) 데이터 객체

1. 정수형(integer)
2. 실수형(double)

3. 문자형(character)
4. 논리형(logical)
5. 복소수형(complex number)

- Special symbol/Values
  - NA : 결측치
  - Inf, -Inf : 무한대(소) ($1/0$)
  - NaN : Not a Number ($0/0$)

- atomic(상수) 데이터객체 예제
  - 실수형(double) / 정수형(integer)

```
typeof(10L)
```

```
## [1] "integer"
```

```
typeof(10)
```

```
## [1] "double"
```

  - 문자형(character)

```r
typeof("Hello World")
```

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

- 논리형(logical)

```r
typeof(2 < 4)
```

```
## [1] "logical"
```

3.2 벡터 (vector) r객체

- 벡터는 하나 이상의 원소로 이루어진 자료
- 벡터를 구성하는 각 원소는 그 유형(data type)이 동일해야 함 ⇒ (1,2,“a”,“b”)는 잘못된 벡터
- 벡터의 생성
  - c(,…,) : 벡터 또는 상수의 연결
  - ‘:’ - 연속된 정수벡터를 생성하는 연산자

```r
x1 <- c(1,2,3,4)
x3 <- c("Aaa", "Baa", "Kim") #character vector
x2 <- 10:15
y <- c(x1, 0, x2); y
```

```
##  [1]  1  2  3  4  0 10 11 12 13 14 15
```

벡터의 생성 함수

rep : 반복

```r
rep(2, 10)
```

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

```r
rep(c(1,2), each=5)
```

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

seq : 등차 수열 생성

```
  seq(0, 1, length=11)
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

1에서 9까지 2씩 증가하는 숫자로 이루어진 벡터를 만듦

```
  seq(1, 9, by = 2)
```

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

- numeric, double, integer, character: 속성이 numeric, double, integer, 혹은 character인 벡터를 괄호안의 수만큼 할당함

```
  integer(length = 10)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

- 벡터의 클래스
  - numeric: 연속형
  - factor: 범주형
  - ordered: 순서있는 범주형

Table 1: R객체의 저장모드 및 스토리지 클래스

R code	mode(x)	class(x)
x<-c(1:10)	“numeric”	“numeric”
x<-factor(1:10)	“numeric”	“factor”
x<-ordered(1:10)	“numeric”	“ordered” “factor”

- 벡터의 component를 접근
  - 인덱스 또는 component이름 이용

```
x <- c(1, 10, 7)
x[c(2:3)]
```

```
## [1] 10  7
```

```
y <- c(a=1, b=10, c=7)
y[c("a", "c")]
```

```
## a c
## 1 7
```

3.3 matrix(행렬)

- 2차원 자료의 저장: 행(row)과 열(column)으로 구성됨

$$A = \begin{pmatrix} 2 & 3 & 5 \\ 12 & 31 & 5 \\ 5 & 13 & 7 \\ 6 & 35 & 72 \end{pmatrix}$$

```
A <- matrix(c(2,12,5,6,3,31,13,35,5,5,7,72), ncol=3); A
```

```
##      [,1] [,2] [,3]
## [1,]    2    3    5
## [2,]   12   31    5
## [3,]    5   13    7
## [4,]    6   35   72
```

- matrix(행렬)의 생성: matrix()함수 이용

```
X1 <- matrix(1:20, nrow=2, ncol=5); X1
```

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

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

 matrix(행렬)의 생성: 대각행렬(diagonal matrix) 생성

```
X2 <- diag(1, 5); X2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

```
X2 <- diag(10)
X2 <- diag(1:10)
X2 <- diag(c(1,3,5,7,9))
```

 대각원소(diagonal elements)의 추출

```
diag(X2)
```

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

- 행렬/벡터의 결합
  - 열단위 결합

```
x <- c(1,2,3); y <- c(4,5,6)
cbind(x,y)
```

```
##      x y
## [1,] 1 4
## [2,] 2 5
## [3,] 3 6
```

  - 행단위 결합

```
rbind(x,y)
```

```
##   [,1] [,2] [,3]
## x    1    2    3
## y    4    5    6
```

 행렬 연산

  곱 (elementwise product)

```
x <- matrix(c(1:6), ncol=3); x
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
y <- matrix(c(1,-1,2,3,2,-1), ncol=3); y
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    2
## [2,]   -1    3   -1
```

```
x*y
```

```
##      [,1] [,2] [,3]
## [1,]    1    6   10
## [2,]   -2   12   -6
```

- 전치행렬 (transpose matrix)

```
t(x)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
```

- 행렬곱 (matrix product)

```
z <- t(x)%*%y; z
```

```
##      [,1] [,2] [,3]
## [1,]   -1    8    0
## [2,]   -1   18    2
## [3,]   -1   28    4
```

- 역행렬 (matrix inversion) A가 $n \times n$ 행렬일 때, 아래를 만족하는 $n \times n$ 행렬 B가 존재하면 B를 A의 역행렬이라고 하고 A^{-1}로 표시함

$$AB = BA = I_n(identity)$$

```
A <- matrix(c(1,2,3,3,0,1,5,4,2), ncol=3); A
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    0    4
## [3,]    3    1    2
```

```
solve(A)
```

```
##             [,1]        [,2] [,3]
## [1,] -0.13333333 -0.03333333  0.4
## [2,]  0.26666667 -0.43333333  0.2
## [3,]  0.06666667  0.26666667 -0.2
```

3.4 리스트(list)

- List는 서로 다른 R오브젝트들을 원소(component)로 가지는 오브젝트
- 리스트의 원소
  - 상수/벡터
  - 행렬/데이터프레임
  - 함수등 모든 R오브젝트
- 리스트의 생성
  - list()함수 이용

```
list(name_1=object_1, ..., name_m=object_m)
```

  - name_1 … name_m 은 콤포넌트의 이름
  - object_1 … 은 콤포넌트 값
  - 예:

```
Lst <- list(name="fred", wife="mary", child.ages=c(4,7,9))
Lst
```

```
## $name
## [1] "fred"
##
## $wife
## [1] "mary"
##
## $child.ages
## [1] 4 7 9
```

- 리스트 구성요소 접근
  - [[ ]]

```
Lst[[1]]
```

```
## [1] "fred"
```

  - 구성요소 이름이 있는 경우(named list)

```
  Lst[["name"]]; # or Lst$name
```

```
## [1] "fred"
```

  서브리스트(sub-list)

```
  Lst[2:3]
```

```
## $wife
## [1] "mary"
##
## $child.ages
## [1] 4 7 9
```

  콤포넌트의 개수: length()

```
  length(Lst)
```

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

 리스트의 결합

 c() : 벡터의 생성 또는 결합과 동일

```
list1 <- list(a1=1, b1=1:3)
list2 <- list(a2=c("Kim", "Park"))
c(list1, list2)
```

```
## $a1
## [1] 1
##
## $b1
## [1] 1 2 3
##
## $a2
## [1] "Kim"  "Park"
```

3.5 데이터프레임(data frame)

- 테이블 형태의 데이터 객체
- 컬럼은 벡터, 펙터(factor)등 서로 다른 속성을 가질 수 있음
- 변수(열)는 길이는 모두 동일
- 데이터프레임 생성

1. data.frame()함수

```
name   <- c("kim","lee","park","Oh")
sex    <- c('f','m','f','m')
income <- c(100,102,300,204)
d1     <- data.frame(name=name, gender=sex, incom=income)
d1
```

```
##   name gender incom
## 1  kim      f   100
## 2  lee      m   102
## 3 park      f   300
## 4   Oh      m   204
```

1. as.data.frame(): 리스트나 행렬을 데이터프레임으로 변환

 데이터프레임 관련 함수

  앞줄/끝줄 요약 보기

```
head(d1, 2) #tail(d1)
```

```
##   name gender incom
## 1  kim      f   100
## 2  lee      m   102
```

  변수명 출력/변수명 지정

```
names(d1)
```

```
## [1] "name"   "gender" "incom"
```

```
names(d1)[3] <- "income"
```

  데이터 차원 출력

```r
  nrow(d1) # number of rows
```

```
## [1] 4
```

```r
  ncol(d1) # number of columns
```

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

```r
  dim(d1)  # row and column dimension
```

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

 데이터프레임 예제(iris data)

```r
  head(iris, 3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

```r
  names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

```r
  dim(iris)
```

```
## [1] 150   5
```

```r
  nrow(iris); ncol(iris)
```

```
## [1] 150
```

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