R을 이용한 머신러닝

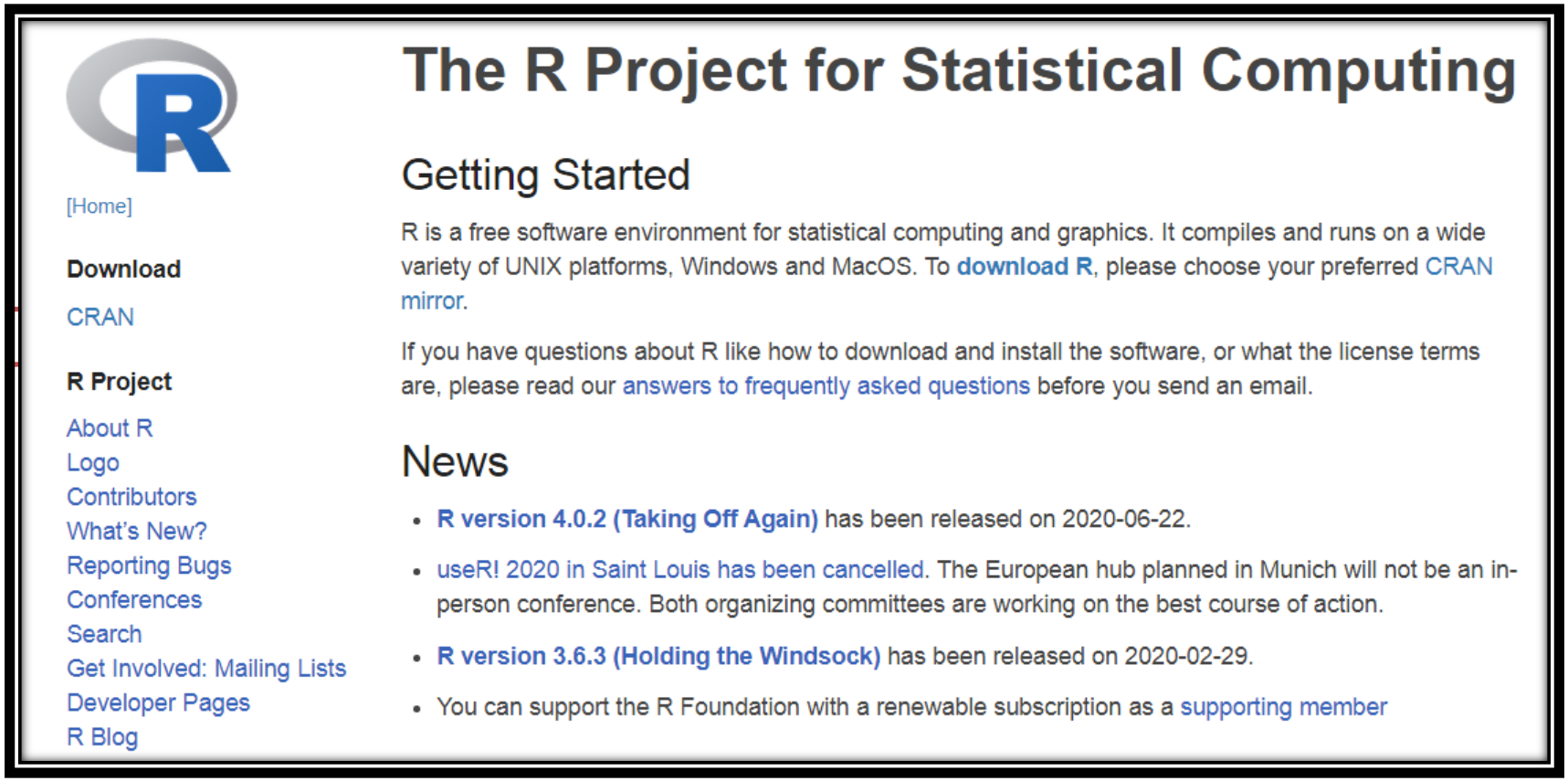
R 사용법

RStudio 설치

R 설치 및 사용법

R console 다운로드

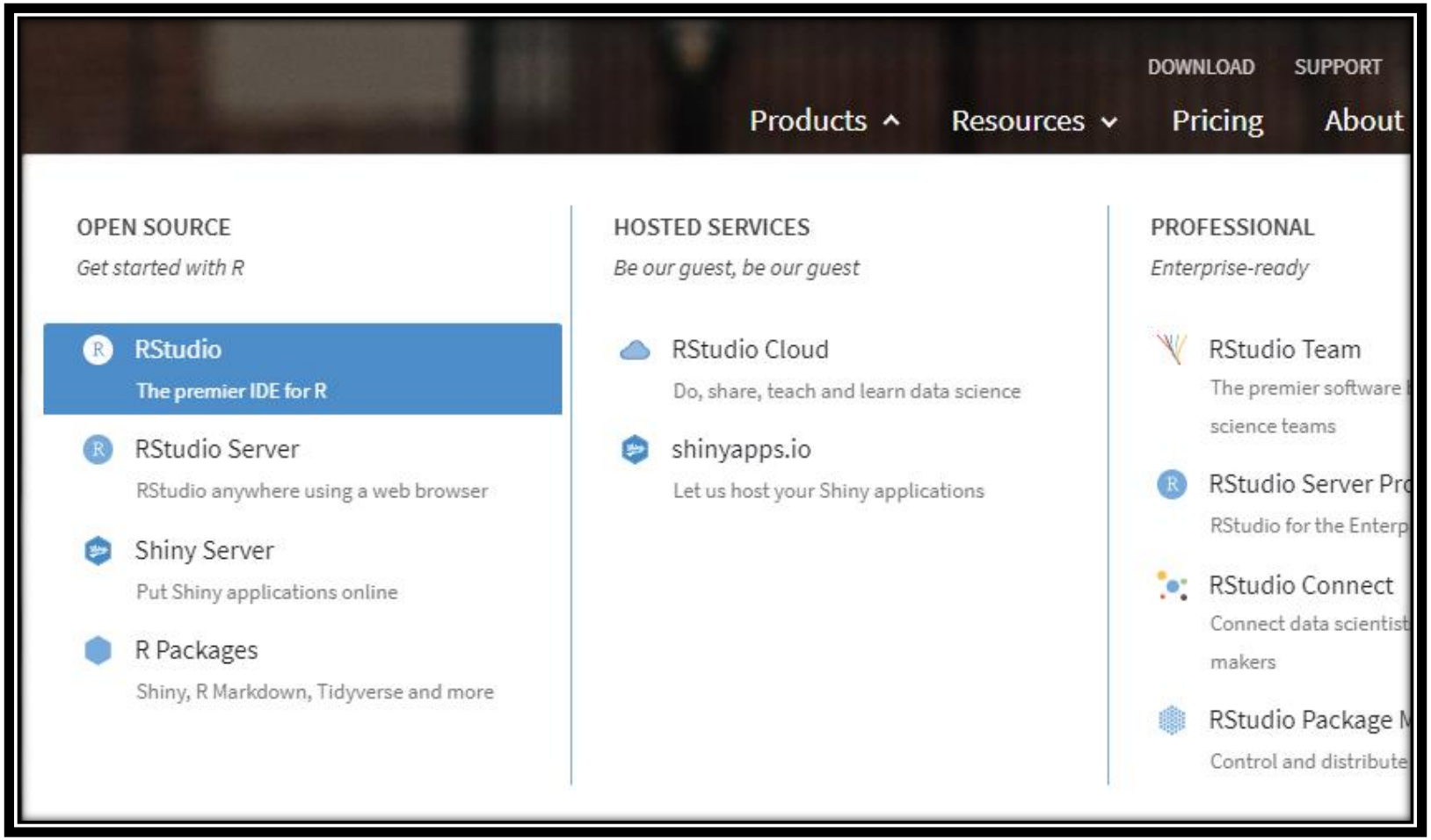
- http://www.r-project.org에서 다운로드 (ver 3.x -> ver 4.x 로 업데이트)

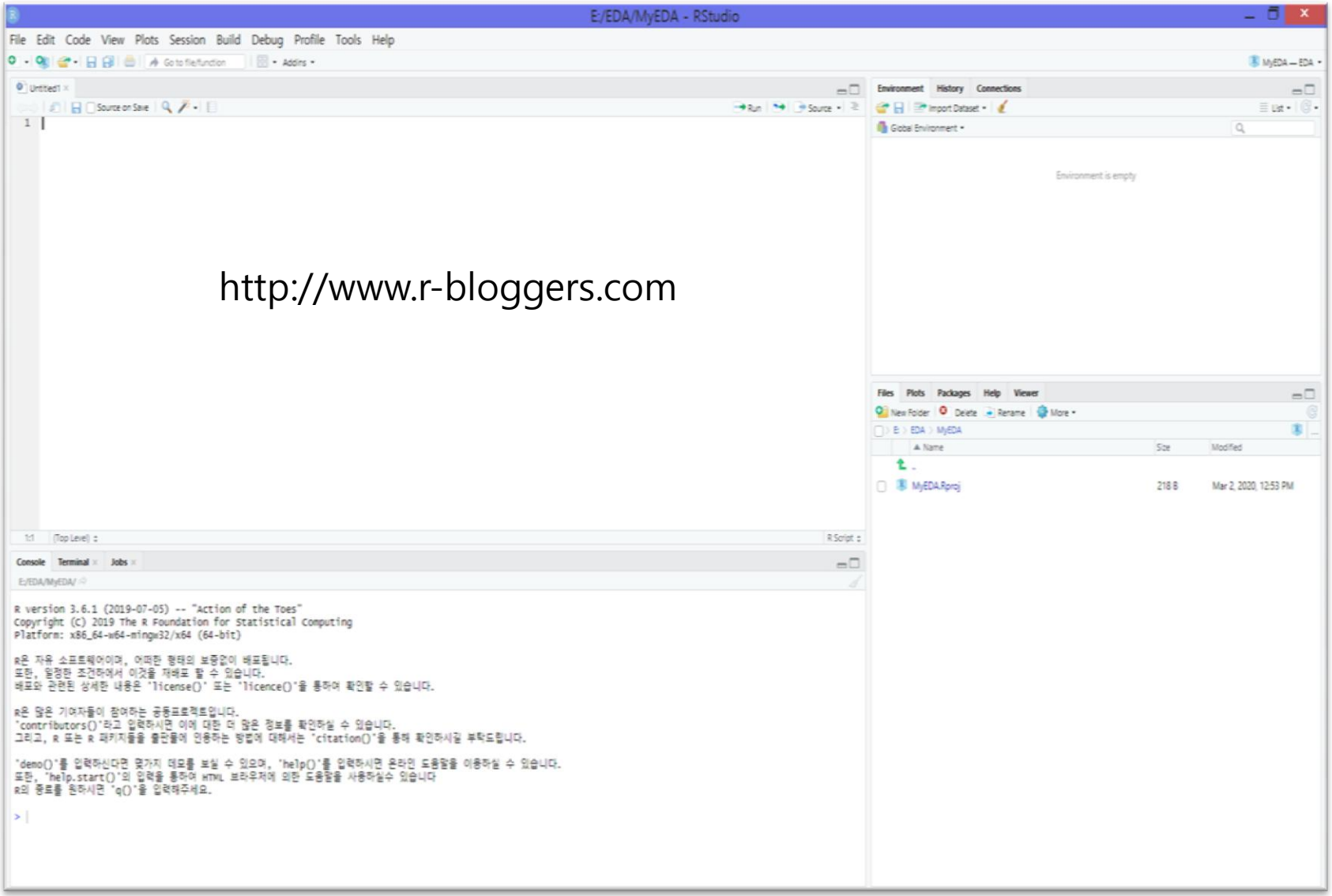


- 관리자 권한으로 프로그램 실행

Rstudio 다운로드

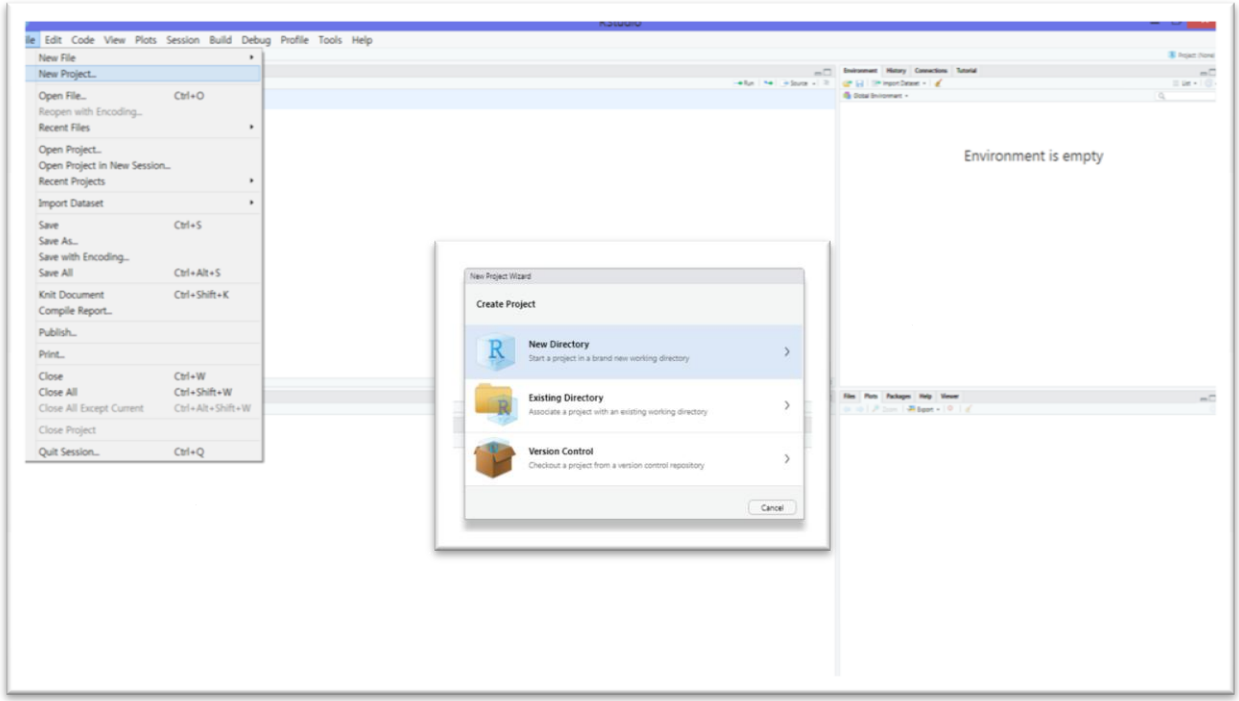
- http://www.rstudio.com 에서 Rstudio 다운로드 및 설치



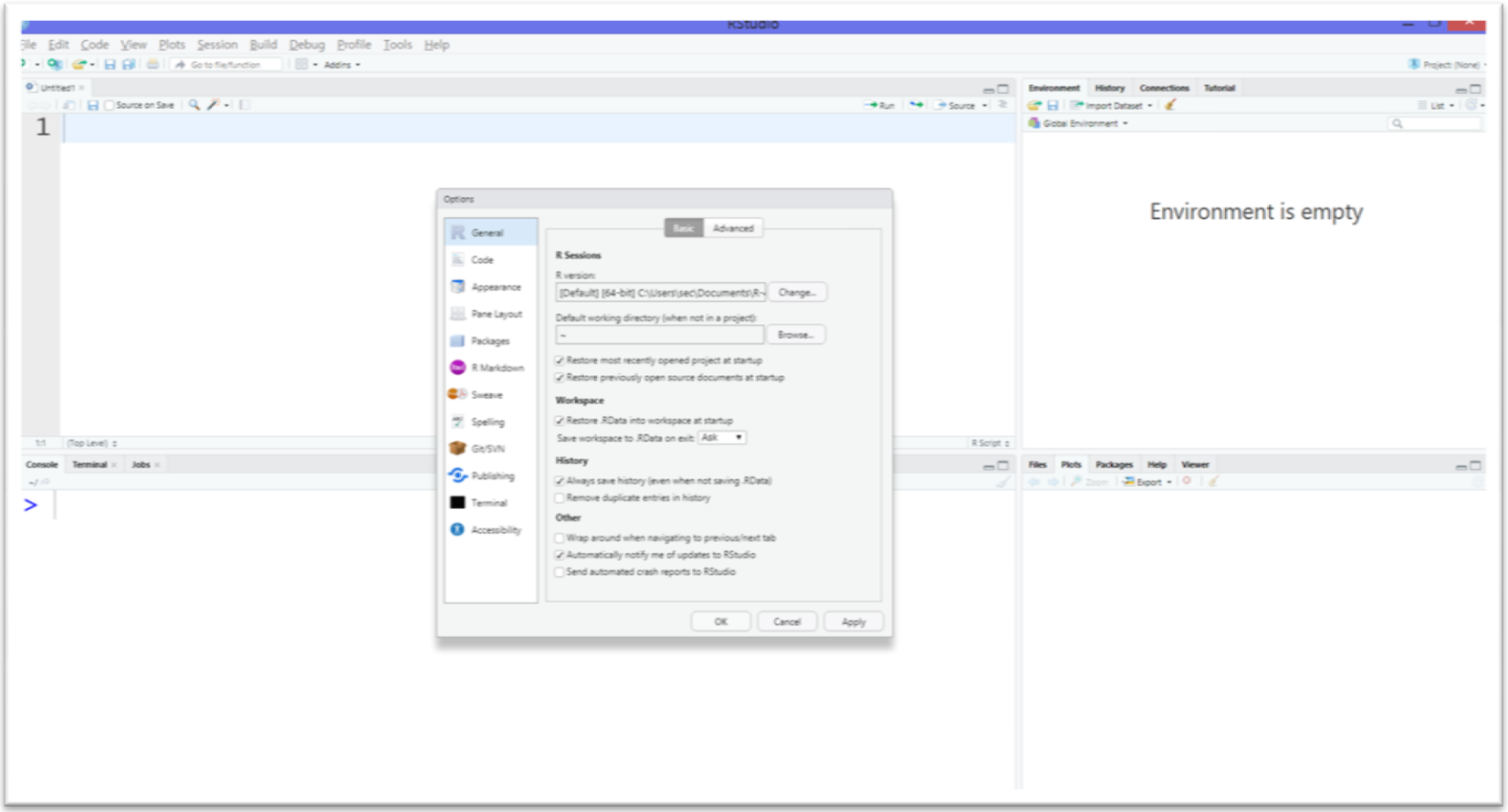

E:/EDA/MyEDA - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1
Source on Save
Run
Source
Environment History Connections
Import Dataset
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder
Delete
Rename
More
E EDA MyEDA
Name
Size
Modified
MyEDA.Rproj
218 B
Mar 2, 2020, 12:53 PM
Console Terminal Jobs
E:/EDA/MyEDA/
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R은 자유 소프트웨어이며, 어떠한 형태의 보증없이 배포됩니다.
또한, 일정한 조건하에서 이것을 재배포 할 수 있습니다.
배포와 관련된 상세한 내용은 'license()' 또는 'licence()'을 통하여 확인할 수 있습니다.
R은 많은 기여자들이 참여하는 공동프로젝트입니다.
'contributors()'라고 입력하시면 이에 대한 더 많은 정보를 확인하실 수 있습니다.
그리고, R 또는 R 패키지들을 출판물에 인용하는 방법에 대해서는 'citation()'을 통해 확인하시길 부탁드립니다.
'demo()'를 입력하신다면 몇가지 데모를 보실 수 있으며, 'help()'를 입력하시면 온라인 도움말을 이용하실 수 있습니다.
또한, 'help.start()'의 입력을 통하여 HTML 브라우저에 의한 도움말을 사용하실수 있습니다
R의 종료를 원하시면 'q()'을 입력해주세요.
>
http://www.r-bloggers.com

Rstudio 프로젝트

- Rstudio 의 프로젝트 기능은 Rstudio 핵심 기능 중 하나.
- 프로젝트는 연관된 데이터, 결과물, 그래프 파일 등으로 구성
- 각각의 프로젝트는 각자 고유의 working디렉토리를 가진다.
- 프로젝트를 사용하면 작업을 잘 정돈할 수 있다.



Rstudio 옵션


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Environment History Connections Tutorial
Environment is empty
Options
General
Code
Appearance
Pane Layout
Packages
R Markdown
Sweave
Spelling
Git/SVN
Publishing
Terminal
Accessibility
Basic
Advanced
R Sessions
Workspace
History
Other
OK
Cancel
Apply
Console
Terminal
Jobs
Files
Plots
Packages
Help
Viewer

프로그래밍의 6원칙

- 변수명
- 데이터형(선언이 필요없다-분석용) + 자료구조
  - 데이터 형 : 문자형, 숫자형, 논리형, Null
  - 구조 : 벡터, 행렬, 배열, 데이터프레임, 리스트
- 연산자
  - 산술, 논리. 비교, 연결 등
- 제어문
  - If/else
- 반복문
  - for, while
- 함수
  - Packages 함수 꾸러미

R 패키지

- Package 인스톨
  - chooseCRANmirror() : 선택 : # 입력
  - Install.packages("ggplot2")
- Package 제거
  - Remove.package()
- Package 로딩
  - Library(ggplot2)
- 업데이트
  - Update.packages (checkBuilt = TRUE, ask=false)
- package 언로딩
  - Detach("package:tidyverse")

변수명

- 변수선언
  - 변수명(객체이름) <- 값 : 새 객체가 생성
  - 변수명 = 값

- 변수명 규칙
  1. <- 할당 연산자. (alt + -)로 자동생성.
  2. 객체이름은 문자로 시작.(문자, 숫자,_,.만 포함)
  3. 따옴표와 괄호는 항상 짝을 이룸.
  4. 대/소문자 구분
  5. # 주석처리
  6. 세미콜론(;) : 여러 명령어를 한 줄에 입력
  7. rm() : 불필요한 객체(값)을 제거

R DataType

데이터 유형

```
#숫자
x <- 2
is.numeric(x)
class(x) #데이터 유형
```

```
#문자 : character
x <- "data"
is.character(x)

#문자 : factor
y <- factor("data")
is.factor(y)
is.character(y)

#문자의 길이
nchar(x)
nchar(y)
nchar("hello”)
```

```
#날짜 및 시간

Sys.Date() #현재날짜

Sys.time() #현재날짜 및 시간

date1 <- as.Date("2020-09-08")

class(date1)

as.numeric(date1)

#1970년 1월1일을 기준으로 날짜와 초를 계산

date2 <- as.POSIXct("2020-09-08 16:30")

class(date2)

as.numeric(date2)
```

#논리형 : true, false

2==3 #2와 3은 같은가?

2!=3 #2와 3은 다른가?

2<3 #2가 3보다 작은가?

2<=3 #2가 3보다 작거나 같은가?

2>3 #2가 3보다 큰가?

2>=3 #2가 3보다 크거나 같은가?

"data" == "stats" #"data"가 "stats"와 같은가?

"data" < "stats" #"data"가 "stats"보다 작은가?

```
#결측값 ; . 또는 99(다른 통계P/G) , NA(R)
z <- c(1,2,NA,8,3,NA,3)
is.na(z) #결측값인지 아닌지 확인

zChar <- c("Hockey",NA,"Lacrosse")
is.na(zChar)
mean(z) # 요소 값이 NA인 벡터에 대해 mean을 적용하면?
mean(z, na.rm = TRUE) # 결측치 제거

#NULL ; 아무것도 없음을 의미
#단독으로 존재하고 벡터 안에 있을 수 없다.
#만약 벡터 안에 사용하면?
z <- c(1,NULL,3)
length(z)
```

R 연산자 및 함수

연산자 및 함수

R연산자	설 명	R연산자	설 명
<-	할당 연산자	>	크다
+, -, *, /	산술 연산자	>=	크거나 같다
^	제곱 연산자	<	작다
~	모형 표현에 사용	<=	작거나 같다
:	수열(모형에서는 교호작용)	||, &&	객체의 첫 번째 요소만 비교
::	팩키지에서 함수 참조	%/%	나눗셈 몫
!	NOT	%%	나눗셈 나머지
&	AND	%*%	행렬곱
|	OR	%in%	매칭 확인
==	Equal to	%any%	연산자생성
!=	Not equal	%>%	Pipe operator

파이프 연산자

- 함수를 호출하는 새로운 패러다임
- magrittr 패키지가 제공
- 기능
  - 파이프를 중심으로 왼쪽에 있는 값이나 객체가 파이프의 오른쪽에 오는 함수의 첫 번째 인자로 삽입하는 방식으로 사용
  - Ctrl + Shift + M
- 예제

```
library(magrittr)
mean(x)
```

→

```
x %>% mean
```

- 문제

```
z <- c(1,2,NA,8,3,NA,3)
sum(is.na(z))
mean(z, na.rm = TRUE)
```

→ ?

통계함수	
mean	산술평균
median	중간값
sd	표준편차
var	분산
max	최대값
min	최소값
IQR	사분위수 범위
range	범위
quantile	사분위수
sum	합계
prod	모든 원소의 곱
length	관측치 개수

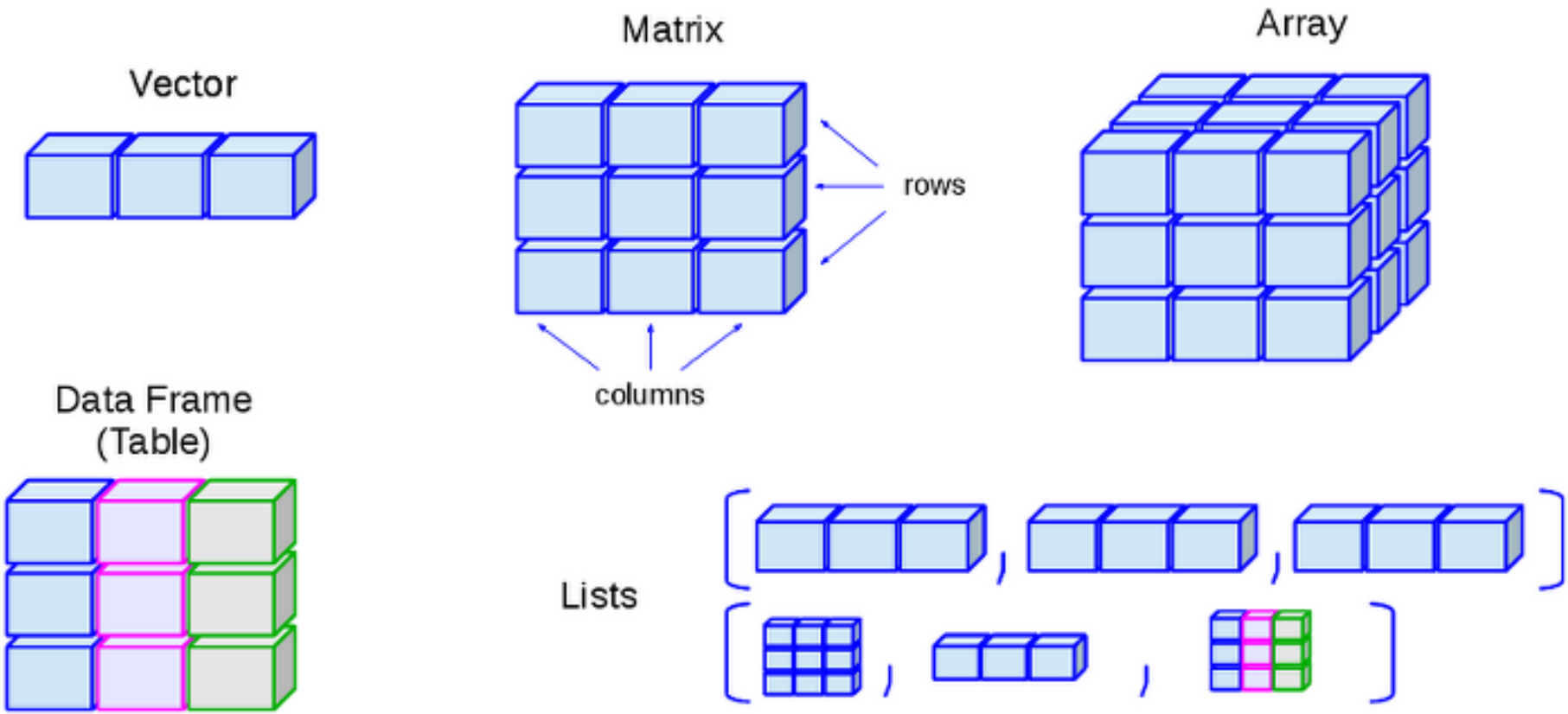
데이터 전처리

- dplyr 패키지

함수	기능
filter()	행 추출
select()	열 추출
group_by()	집단별로 나누기
summarise()	통계치 요약
arrange()	정렬
mutate()	변수추가
left_join()	데이터 합치기(열)
bind_rows()	데이터 합치기(행)

R data structure

데이터 구조


Vector
Matrix
rows
columns
Array
Data Frame
(Table)
Lists

벡터

- R의 기본구조
- 동일한 데이터 형식
- 문자형, 수치형, 복소수형, 논리형
- NULL(값이 존재하지 않음, 길이=0), NA(결측치, 길이=1)

팩터

- 오직 범주형 변수나 서열 변수에 적용
- 정보 저장에 필요한 메모리 크기가 줄어듬.
- 문자형, 수치형 중 한가지만 사용

리스트

- 서로 다른 데이터 타입을 가짐

데이터프레임

- 데이터의 행과 열을 가지고 있는 구조 (2차원 구조)
- 동일한 개수의 값을 갖는 벡터나 팩터의 리스트
- 문자형, 수치형, 복소수형, 논리형

행렬

- 데이터의 행과 열을 갖는 2차원 표를 나타내는 데이터 구조
- 동일한 데이터 형식

R 데이터 구조

- 벡터

```
#vector 입력
subject_name <- c("John Doe","Jane Doe","Steve Graves")
temperature <- c(98.1, 98.6, 101.4)
flu_status <- c(FALSE, FALSE, TRUE)

#index 지정
temperature[2]
temperature[2:3]

#data 제외
temperature[-2]

#logical vector
temperature[c(TRUE, TRUE, FALSE)]
```

R 데이터 구조

– 팩터

```
#factor
gender <- factor(c("MALE","FEMALE","MALE"))
gender #Levels는 factor가 가질 수 있는 가능한 범주의 집합

#raw data에 없는 데이터 level 추가
blood <- factor(c("O","AB","A"),
                levels = c("A","B","AB","O"))
blood

# 명목척도->서열척도 # MILD 경증, MODERATE 중등증, SEVERE 중증
symptoms <- factor(c("SEVERE","MILD","MODERATE"),
                   levels = c("MILD","MODERATE","SEVERE"),
                   ordered = TRUE)
symptoms
symptoms > "MODERATE"
symptoms >= "MODERATE"
```

R 데이터 구조

- 리스트

```
#list
subject_name <- c("John Doe","Jane Doe","Steve Graves")
temperature <- c(98.1, 98.6, 101.4)
flu_status <- c(FALSE, FALSE, TRUE)
gender <- factor(c("MALE","FEMALE","MALE"))
blood <- factor(c("O","AB","A"),
                levels = c("A","B","AB","O"))
symptoms <- factor(c("SEVERE","MILD","MODERATE"),
                   levels = c("MILD","MODERATE","SEVERE"),
                   ordered = TRUE)

#첫 번째 환자의 전체 데이터
subject_name[1]
temperature[1]
flu_status[1]
gender[1]
blood[1]
symptoms[1]
```

R 데이터 구조

- 리스트

```
#모든 값을 하나의 객체로 그룹화해 반복적으로 사용
subject1 <- list(fullname=subject_name[1],
                 temperature=temperature[1],
                 flu_status=flu_status[1],
                 gender=gender[1],
                 blood=blood[1],
                 symptoms=symptoms[1])
subject1

#list내 구성요소 추출
subject1[2]

#구성요소의 값(벡터)을 추출
subject1[[2]]
subject1$temperature # $리스트 구성요소 이름

#여러 구성요소 추출
subject1[c("temperature", "flu_status")]
```

R 데이터 구조

- 데이터 프레임

```
#dataframe
pt_data <- data.frame(subject_name, temperature, flu_status,
                      gender, blood, symptoms,
                      stringsAsFactors = FALSE)
pt_data
pt_data$subject_name
pt_data[c("temperature","flu_status")]
pt_data[1,2] #환자 데이터 프레임의 첫번째 행, 두번째 열의 값 추출
pt_data[c(1,3), c(2,4)] #1,3번째 행과 2,4번째 열
pt_data[,1] #1번째 열의 모든 행 추출
pt_data[1,] #1번째 행의 모든 열 추출
pt_data[,] #모든 데이터 추출
pt_data[c(1,3),c("temperature","gender")] # 숫자 혹은 열이름으로 추출
pt_data[-2, c(-1,-3,-5,-6)] #(-)부호로 행 또는 열 제외
pt_data$temp_c <- (pt_data$temperature-32)*(5/9)
pt_data
pt_data[c("temperature","temp_c")]
```

R 데이터 구조

- 행렬

```
#행렬과 배열
m <- matrix(c(1,2,3,4), nrow = 2) # 열 우선 방식
m
m <- matrix(c(1,2,3,4), ncol = 2)
m
m <- matrix(c(1,2,3,4,5,6), nrow = 2)
m
m <- matrix(c(1,2,3,4,5,6), ncol = 2)
m
m[1,]
m[,1]
```

문제[1]

	x	y	q
1	10	-4	Hotkey
2	9	-3	Football
3	8	-2	Baseball
4	7	-1	Curling
5	6	0	Rugby
6	5	1	Lacrosse
7	4	2	Basketball
8	3	3	Tennis
9	2	4	Cricket
10	1	5	Soccer

- 함수 data.frame을 이용하여 위의 데이터 프레임(theDF)을 만들어라.
- 세 개의 벡터 x, y, q를 이용하여라.
- 열 이름을 다음과 같이 변경하여라.
  - First=x, Second=y, Sport=q
- theDF의 3,5행과 2,3열의 데이터를 출력하여라.