R 기초

D:plus

목차

1.개요

1.1 R과 R studio 설치하기

(1) R 설치

① CRAN(The Comprehensive R Archive Network, http://www.cran.r-project.org/)에 접속하여

② OS에 맞게 R 다운로드

③ Subdirectories에서 ‘base’ 선택

④ R버전과 OS 확인하고 다운로드 하여 설치

(설치 시 꼭 C 드라이브에)

1.1 R과 R studio 설치하기

(2) R studio 설치

① RStudio 사이트 접속 (http://www.rstudio.com/)

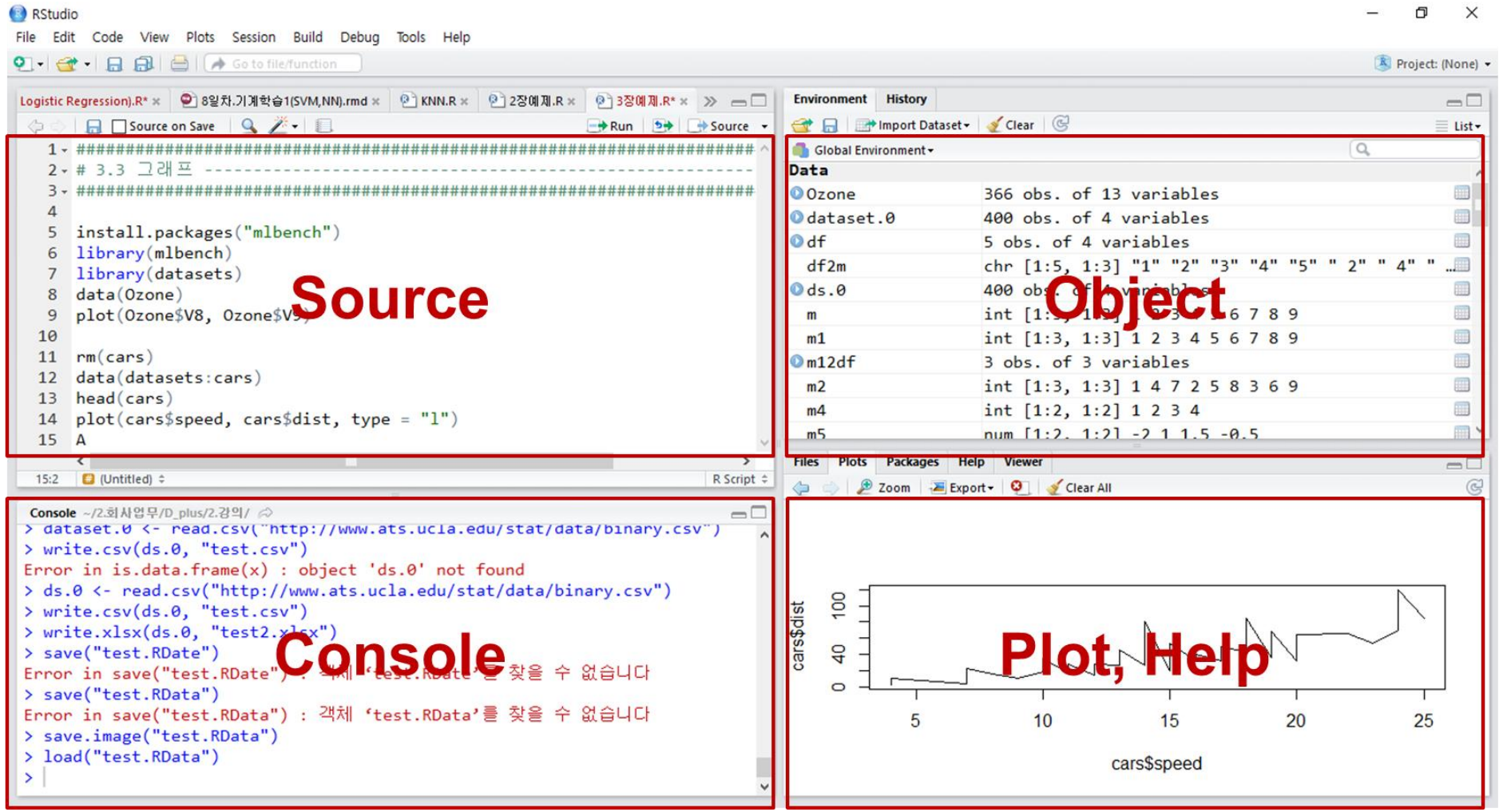
② Download Rstudio 클릭

③ 'Desktop'과 'Server' 중에서 'Desktop' 선택

④ 'Open Source Edition'과 'Commercial License' 중에서 선택

⑤ OS에 맞게 RStudio 다운로드하고 설치
(설치 시 꼭 C 드라이브에)

1.2 R Studio 기능 살펴보기




```
1  ##########################################################################
2  # 3.3 그래프 ------------------------------------------------------------
3  ##########################################################################
4
5  install.packages("mlbench")
6  library(mlbench)
7  library(datasets)
8  data(Ozone)
9  plot(Ozone$V8, Ozone$V9)
10
11 rm(cars)
12 data(datasets:cars)
13 head(cars)
14 plot(cars$speed, cars$dist, type = "l")
15 A
```

```
> dataset.0 <- read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
> write.csv(ds.0, "test.csv")
Error in is.data.frame(x) : object 'ds.0' not found
> ds.0 <- read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
> write.csv(ds.0, "test.csv")
> write.xlsx(ds.0, "test2.xlsx")
> save("test.RDate")
Error in save("test.RDate") : 객체 'test.RDate'를 찾을 수 없습니다
> save("test.RData")
Error in save("test.RData") : 객체 'test.RData'를 찾을 수 없습니다
> save.image("test.RData")
> load("test.RData")
>
```

1.3 Library(Package)

라이브러리 활용

- 사용 가능 패키지 : 11300개 이상.
- 설치
  - install.packages(“패키지이름“)
  - 설치 위치 : .libPaths()
  - 설치여부 확인 :
- Loading / Unloading
  - library(패키지이름) 혹은 require(패키지이름)
  - detach("package:패키지이름", unload=TRUE)

R code

```
.libPaths()

install.packages(“ggplot2”)

library(ggplot2)

Detach(“package:ggplot2”, unload = T)
```

D:plus

1.4 도움말

Help

- 함수, 변수가 궁금하다면 : ?<함수/변수명>
- ?? : 함수이름을 정확히 모를 때
- 함수 예제 살펴보기 : example(함수명)
- 함수 이름 찾기

R code

```
?seq

??seq

example(seq)
```

D:plus

2.R 데이터 자료형

2.1 변수 - 개요

- 변수 이름
  - 알파벳, 한글, 숫자, '_', '.' 로 구성.
  - 첫 글자는 문자나, '.' 로 시작
- 변수에 값 할당하기 : <- , =

R code

```
a <- 3
b = 4.5
c <- a + b
```

D:plus

2.2 변수 - 스칼라

- 숫자 : int, num
- 문자열
  - 한 개 문자가 아닌 문자열로 처리
  - “”, ‘’ 둘 다 사용 가능

R code

```
d <- "Hello"
e <- 'R World!'
f <- d + e
f <- paste(d, e, sep = ' ')
```

D:plus

2.2 변수 - 스칼라

- 논리값
  - TRUE, T
  - FALSE, F
  - 논리연산자 : & (and), | (or), ! (not)
  - &&, ||

R code

```
TRUE & FALSE
TRUE | FALSE
c(TRUE, T) & c(FALSE, F)
!T
```

2.2 변수 - 스칼라

- Factor(요인, 범주)
  - 범주형 변수를 저장하는 형식
  - 특정 범주로 정의된 값만 저장 가능
  - 실제는 숫자가 저장됨

R code

```
gender <- factor('male'
                 , c('male', 'female'))

str(gender)
```

D:plus

2.2 변수 - 벡터

- 배열의 개념
- 벡터 안에서는 모두 동일한 데이터형식
- 벡터 내 데이터 접근 : 벡터명[index]
- 몇몇 함수들 : seq(), rep(), length()
- 벡터의 연산 +,-,*,/
- 벡터 안에 벡터를 포함

R code

```
x <- c(1,2,3,4,5)
y <- c('a','b','c','d',2)
x[2]
y[2:3]
x1 <- 1:20
seq(1, 10, 0.2)
rep(5:10,2)
length(x1)
x + x1
x2 <- c(x,6,7,8)
```

2.4 리스트

- 딕셔너리 같은 개념
- 다양한 데이터 형식들의 집합
- 키, 값들로 구성
- 리스트 내 데이터 접근
  - list명[index] : 서브 데이터 반환
  - list명[[key_index]][value_index] : 값 반환
- 언제 사용하지?

R code

```
x <- list(name="hanky“
       , sight = c(0.8, 0.5))
class(x)
x[1]
x[[2]][1]
x$sight
x$sight[1]
```

2.5 행렬

- m X n
- 행렬 내 데이터 형식 동일
- 행렬 내 데이터 접근
  –행렬명[행, 열]

R code

```
matrix(1:9, nrow = 3)
matrix(1:9, nrow = 3, byrow = T)
matrix(1:9, nrow = 3
       , dimnames = list(c("r1","r2","r3")
                         ,c("c1","c2","c3")))
m <- matrix(1:9, nrow = 3)
m[2,3]
m[2,]
```

D:plus

2.5 행렬

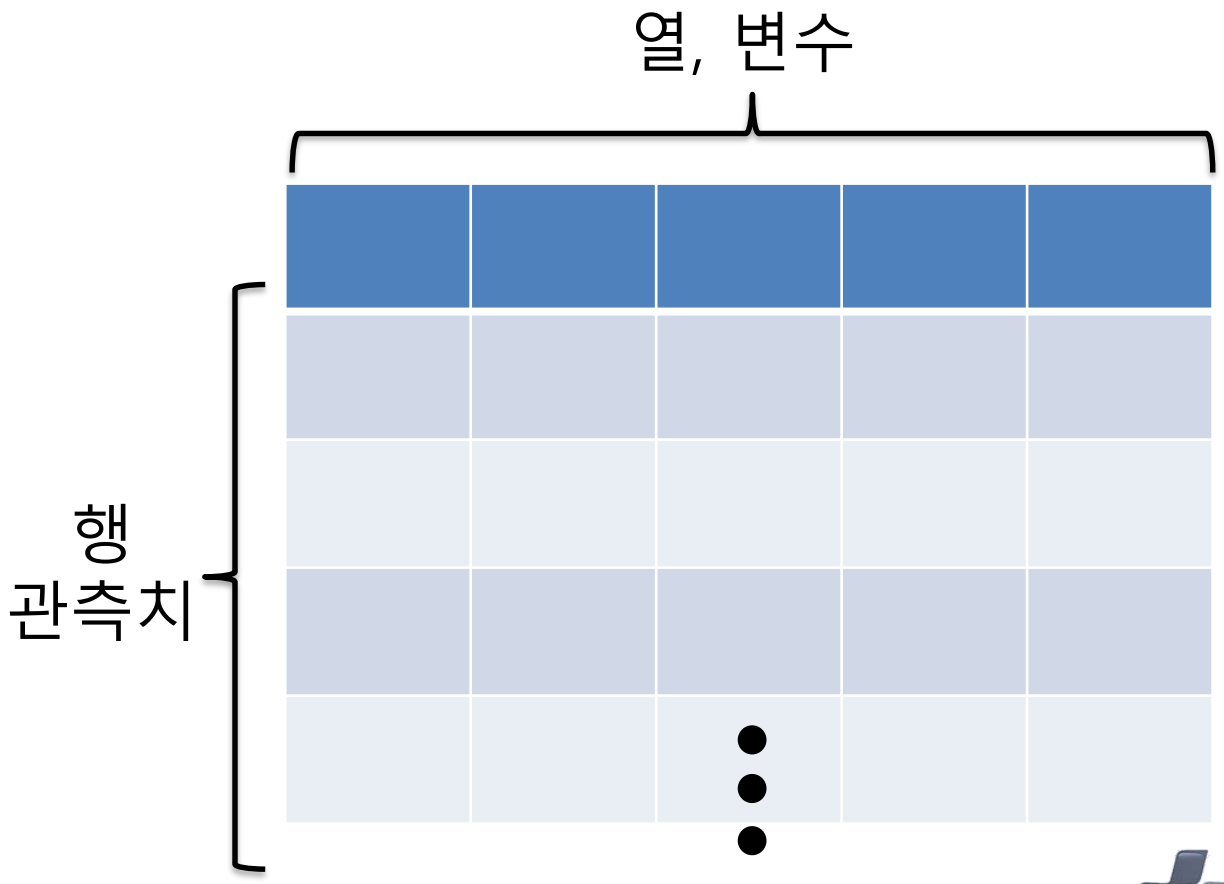
- 행렬의 연산
  - 상수와 연산
  - 행렬의 합,차 : +, -
  - 행렬의 곱 : %*%
  - 역행렬(Inverse) : solve(행렬)
  - 전치행렬(transpose) : t(행렬)

R code

```
m2 <- matrix(1:9, nrow = 3)
m3 <- matrix(1:9, nrow = 3, byrow = T)

m2 * 2
m2 + m3
m2 * m3
m2 %*% m3

m4 <- matrix(1:4, nrow = 2)
m5 <- solve(m4)
m4 %*% m5

t(m4)
```

2.6 데이터프레임

- R에서 가장 중요한 자료형
- RDBMS에서의 테이블 형태
- 변수(벡터)들의 집합
  → 각 열을 변수라고 부른다!
- 행렬과 다른 점은?



2.6 데이터 프레임

- 데이터 프레임 만들기
- 데이터 프레임 데이터 접근
  - –df[**row_index**, **column_index**]
  - –column_index 대신 변수이름을 적어도 됨.
- 상위/하위 몇 개 행만 가져오기
  - –head(df)
  - –tail(df)

R code

```
df <- data.frame(x=1:5, y = seq(2,10,2))
df$z <- c("kim","lee","park","han","kang")

df[,"z"]
df$z
df[2:3,c("x","z")]
df[2:3,c(1,3)]

head(cars)
tail(cars)
```

2.6 데이터 프레임

- 구조 살펴보기 : str()
- 복사하기 : NewDF <- DF
- 열 이름 확인 : names
- 열 이름 변경

R code

```
str(cars)


cars2 <- cars
names(cars2)
a <- names(cars2)
a[1] <- "speed_mph"
names(dia) <- a
names(dia)
```

D:plus

2.7 데이터 형식 변경하기

- 데이터 형식 확인하기
  - –class( )
- 형식 변환하기
  - –as.integer( )
  - –as.character( )
  - –as.vector( )
  - –as.factor( )
  - –as.matrix( )
  - –as.data.frame( )

R code

```
class(m1)
class(df)

df2m <- as.matrix(df)
m12df <- as.data.frame(m1)

class(df2m)
class(m12df)
```

D:plus

2.8 특이한 값

(1) NA

- NA가 뭐지?
  - –Not Available
  - –Missing Value
  - –누락된 값
  - –측정되지 않은 값
  - –사용할 수 없는 값
- NA 찾기
- NA에 값 부여하기

R code

```
df$City <- c("Seoul","Busan",NA,NA,"Incheon")

is.na(df)
colSums(is.na(df))

df$City[is.na(df$City)] <- "Daejeon"
df$City
```

D:plus

2.8 특이한 값

(2) NULL

- 값이 지정되지 않음.
- 값 초기화

R code

```
i <- NULL

is.null(i)
```

실습

011 R기초_자료형_데이터다루기_연습문제.R

3. 데이터 다루기

D:plus

3.1 파일 가져오기 & 파일로 저장하기

- 작업디렉토리 설정하기
  - –getwd( ) : 현재 디렉토리 가져오기
  - –setwd( ) : ‘//’나 ‘\’로 하위 디렉토리(폴더) 표시
- 파일 가져오기
  - –read.csv, write.csv
  - –read.xlsx
  - –Web에 있는 파일

R code

```
getwd()
setwd("C:/Users/…/")

read.csv("testdata.csv")

ds.0 <-
read.csv("https://archive.ics.uci.edu/ml/
machine-learning-databases/car/car.data")
```

3.1 파일 가져오기 & 파일로 저장하기

- 특정 오브젝트 파일로 저장하기
  - –write.csv
  - –write.xlsx

R code

```
write.csv(ds.0, "test.csv")
write.xlsx(ds.0, "test2.xlsx")
```

D:plus

3.2 작업환경 저장하기 & 불러오기

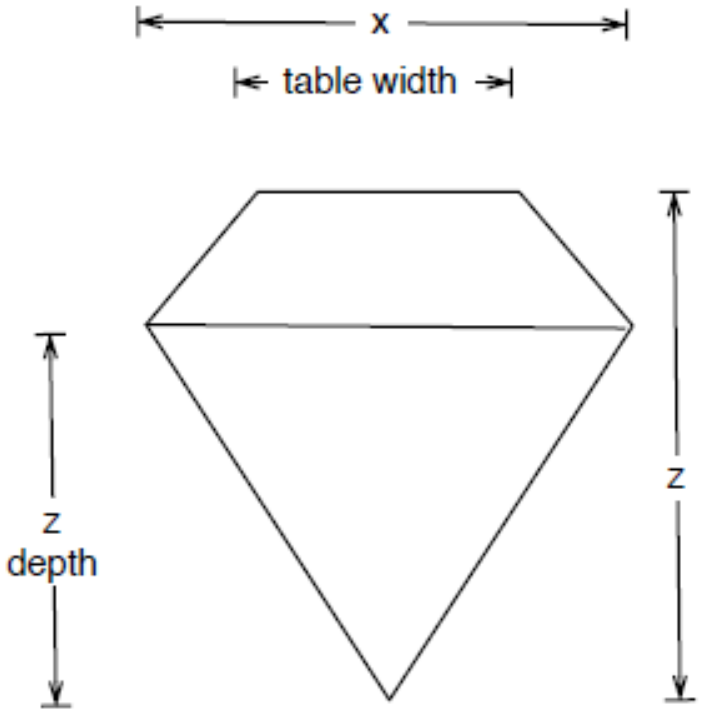
- 오브젝트 전체 한꺼번에 저장하기
  - –save.image( )
- 로딩하기
  - –load( )
- 오브젝트 제거하기
  - –rm( )

R code

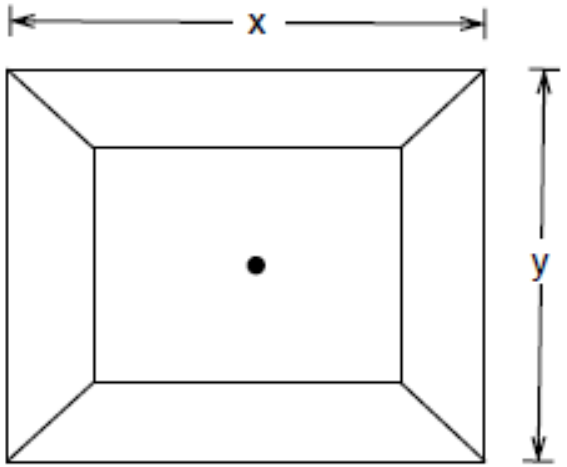
```
save.image("test.RData")
load("test.RData")

rm(ds.0)
rm(list = ls())
```

D:plus

Diamonds 데이터셋



※ 1 carat = 0.2g



depth = z depth / z * 100
table = table width / x * 100

R code

```
library(ggplot2)

data(package = "ggplot2")

diamonds
?diamonds
```

D:plus

3.3 데이터프레임 열 추가, 변경, 삭제

- 열 추가
  - dataframe$NewColumn <- NA
  - dataframe$NewColumn <- 계산식
- 열 변경
  - dataframe$column <- 변경할 값
- 열 삭제
  - dataframe$column <- NULL

R code

```
str(diamonds)

dia <- data.frame(diamonds)

dia$volume <- dia$x * dia$y * dia$z
dia$volume <- round(dia$volume,1)
dia$volume2 <- NA
str(dia)
dia$volume2 <- NULL
```

D:plus

3.4 원하는 데이터 조회하기

- 원하는 열, 행(조건) 가져오기
  - subset()
  - which() : 해당 조건에 맞는 인덱스값 반환
- attach , detach
  - 데이터프레임 이름을 생략가능

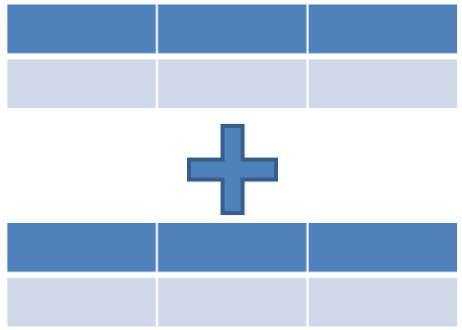
R code

```
subset(dia, select = c("carat", "color", "price")
       , subset = (color == "E" & price > 1000))

which(dia$color == "E" & dia$price > 1000)

dia[which(dia$color == "E" & dia$price > 1000)
        , c("carat", "cut", "color", "price")]

attach(dia)
dia[which(color == "E" & price > 1000)
        , c("carat", "cut", "color", "price")]
detach(dia)
```

3.5 데이터 프레임 합치기

- cbind

- rbind



R code

```
cbind(df1, df2)

rbind(df1, df2)
```

D:plus

3.6 dplyr 패키지

dplyer 함수	설명	유사함수
filter()	조건에 따른 데이터추출	subset()
select()	열 선택	subset()
mutate()	열 추가	transform()
arrange()	정렬	order(), sort()
group_by()	집계를 위한 그룹핑	
summarise()	집계	aggregate()

※ 함수 이름 : 모두 소문자!

D:plus

3.6 dplyr 패키지

select(dataset, variable1, variable2, …)

- SQL의 select 절

R code

```
select(diamonds, carat, cut, color)
```

3.6 dplyr 패키지

filter(dataset, variable1 <조건식>)

- SQL의 where 절

R code

```
filter(diamonds, carat >= 1, cut == 'Premium')
filter(diamonds, carat >= 1 & cut == 'Premium')
filter(diamonds, carat >= 1 | cut == 'Premium')
```

- 같은 값 비교 시, '=='

3.6 dplyr 패키지

mutate(dataset, new_variable = <계산식>)

- SQL의 select 절에서 새로운 계산열 만들기

R code

```
mutate(diamonds, volume = round(x*y*z, digits = 2))
```

3.6 dplyr 패키지

arrange(dataset, variable1, variable2)

- SQL의 Order by 절

R code

```
arrange(diamonds, price)
arrange(diamonds, desc(price))
```

D:plus

3.6 dplyr 패키지

group_by(dataset, variable1, variable2)
Summarize(dataset, new_var = <집계함수>)

- SQL의 group by 절

R code

```
groupby.df <- group_by(diamonds, color)
summarise(groupby.df, AvgPrice = mean(price))
```

3.6 dplyr 패키지

여러 문장을 한꺼번에 연결하기(chain) : %>%

- dataset %>% 함수1() %>% 함수2()

R code

```
group_by(diamonds, color) %>%
    summarise(AvgPrice = mean(price))

group_by(diamonds, color) %>% summarise(AvgPrice = mean(price))

group_by(diamonds, color)
%>% summarise(AvgPrice = mean(price))
```

3.7 apply 함수들

apply

R code

```
x<-matrix(rnorm(30), nrow=5,
ncol=6)
apply(x, 2 ,sum)
```

-1.7189391	-1.0863995	1.0996117	-0.55559727	-0.1792310	-0.8088577
-2.2542126	-1.3201873	-2.0533779	1.29055209	0.3264156	0.5412132
1.9874737	0.6265486	-0.3684977	1.40028967	-0.7574303	-2.3241569
0.2140376	0.8850445	1.4782993	-1.28177703	-0.5015628	1.1537703
0.9637687	1.3191502	0.8000988	0.09345943	1.4535431	1.0935720

Result

sum

-0.8078717	0.4241565	**0.9561342**	0.9469269	0.3417346	-0.3444591

D:plus

3.7 apply 함수들

lapply, sapply

- lapply() : 결과를 리스트로
- sapply() : 결과를 벡터/행렬형태로

R code

```
lapply(cars, sum)
sapply(cars, sum)
```

```
> apply(cars, 2 ,max)
speed  dist
   25   120
> lapply(cars, sum)
$speed
[1] 770

$dist
[1] 2149

> sapply(cars, sum)
speed  dist
  770  2149
```

D:plus

3.7 apply 함수들

tapply

- SQL의 group by로 집계하는 구문과 유사

R code

```
tapply(mpg$cty, mpg$manufacturer, mean)
```

D:plus

실습

012 R기초_데이터다루기_연습문제.R