통계 교육과 빅데이터 실습

박선철 충북대학교 정보통계학과 pscstat@chungbuk.ac.kr

8/4/2021

Contents

1 통계 패키지 R

- R 웹사이트
- R Studio 웹사이트
- R Shiny 웹사이트

2 R의 기본 함수

- R은 기본적으로 작업 수행을 위해 함수를 거치게 된다. 함수는 보통 `함수명(투입값1, 투입값2, ...)` 등으로 이루어진다.
- 우리가 가장 쉽게 보는 `c()` 또한 함수로, concentrate의 의미를 갖고 있으며 투입값들을 하나의 벡터로 묶어준다.

```
x <- c(1,3,2.5)
x
```

```
## [1] 1.0 3.0 2.5
```

```
y <- c(1,4,3)
y
```

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

- R에서의 덧셈: 원소별로 덧셈해준다.

```
x+y
```

```
## [1] 2.0 7.0 5.5
```

- R에서의 곱셈: 역시 원소별로 곱셈해준다.

```
x*y
```

```
## [1]  1.0 12.0  7.5
```

- R에서의 내적: `%*%`

```
x%*%y
```

```
##      [,1]
## [1,] 20.5
```

- R에서의 외적: `outer()` 함수

```
f <- outer(x,y, function(x,y) cos(y)/(1+x^2))
f
```

```
##            [,1]        [,2]        [,3]
## [1,] 0.27015115 -0.32682181 -0.49499625
## [2,] 0.05403023 -0.06536436 -0.09899925
## [3,] 0.07452446 -0.09015774 -0.13655069
```

- `length()`: 자료의 길이를 체크하는 함수

```
length(x)
```

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

```
length(y)
```

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

- `seq()`: 수열 함수

```
z <- seq(1,10) #1부터 10까지
z
```

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

```
z <- 1:10 #위와 동일
z
```

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

```
z <- seq(-pi, pi, length=50)
z
```

```
##  [1] -3.14159265 -3.01336438 -2.88513611 -2.75690784 -2.62867957 -2.50045130
##  [7] -2.37222302 -2.24399475 -2.11576648 -1.98753821 -1.85930994 -1.73108167
```

```
## [13] -1.60285339 -1.47462512 -1.34639685 -1.21816858 -1.08994031 -0.96171204
## [19] -0.83348377 -0.70525549 -0.57702722 -0.44879895 -0.32057068 -0.19234241
## [25] -0.06411414  0.06411414  0.19234241  0.32057068  0.44879895  0.57702722
## [31]  0.70525549  0.83348377  0.96171204  1.08994031  1.21816858  1.34639685
## [37]  1.47462512  1.60285339  1.73108167  1.85930994  1.98753821  2.11576648
## [43]  2.24399475  2.37222302  2.50045130  2.62867957  2.75690784  2.88513611
## [49]  3.01336438  3.14159265
```

```
z <- seq(1,100, by=3)
z
```

```
##  [1]   1   4   7  10  13  16  19  22  25  28  31  34  37  40  43  46  49  52  55
## [20]  58  61  64  67  70  73  76  79  82  85  88  91  94  97 100
```

- ls(): 모든 오브젝트 (데이터, 함수 등)의 이름을 보여준다.
- rm(): 오브젝트를 지울 때 쓰는 함수

```
ls()
```

```
## [1] "f"  "OS" "x"  "y"  "z"
```

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

```
## [1] "f"  "OS" "z"
```

- 도움말: 우리가 쓰고자 하는 함수 앞에 ?을 붙인다.

```
#?ls
```

3 R의 행렬 작성

- 행렬 matrix() 함수: matrix(data, row갯수, col갯수)로 선언

```
x <- matrix(data=c(1,2,3,4), nrow=2, ncol=2)
x
```

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

- R에서는 1행부터 차례대로 원소가 들어간다. 이것을 1열부터 차례대로 원소가 들어가게 하려면 byrow=TRUE로 쓰면 된다.

```
matrix(c(1,2,3,4),2,2,byrow=TRUE)
```

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

- 제곱근: sqrt() 함수, 제곱: ^2 함수

```
sqrt(x)
```

```
##          [,1]     [,2]
## [1,] 1.000000 1.732051
## [2,] 1.414214 2.000000
```

```
x^2
```

```
##      [,1] [,2]
## [1,]    1    9
## [2,]    4   16
```

4 무작위 자료의 생성

- 정규분포를 따르는 자료를 생성할 때에는 rnorm() 함수를 쓴다.

```
x <- rnorm(50) #자료의 갯수가 50개
y <- x + rnorm(50, mean = 50, sd = .1) #평균이 50이고 표준편차가 1인 정규분포를 따르는 무작위자료 50개 + x
```

- 공분산 함수: cov()

```
cov(x,x)
```

```
## [1] 1.189356
```

- 상관관계 함수: cor()

```
cor(x,y)
```

```
## [1] 0.9958072
```

- 시뮬레이션 데이터 고정: 무작위 추출시 매 실행때마다 결과값이 달라지므로 같은 결과를 얻기 위해서는 먼저 set.seed() 함수를 통해 시드를 고정해야 한다.

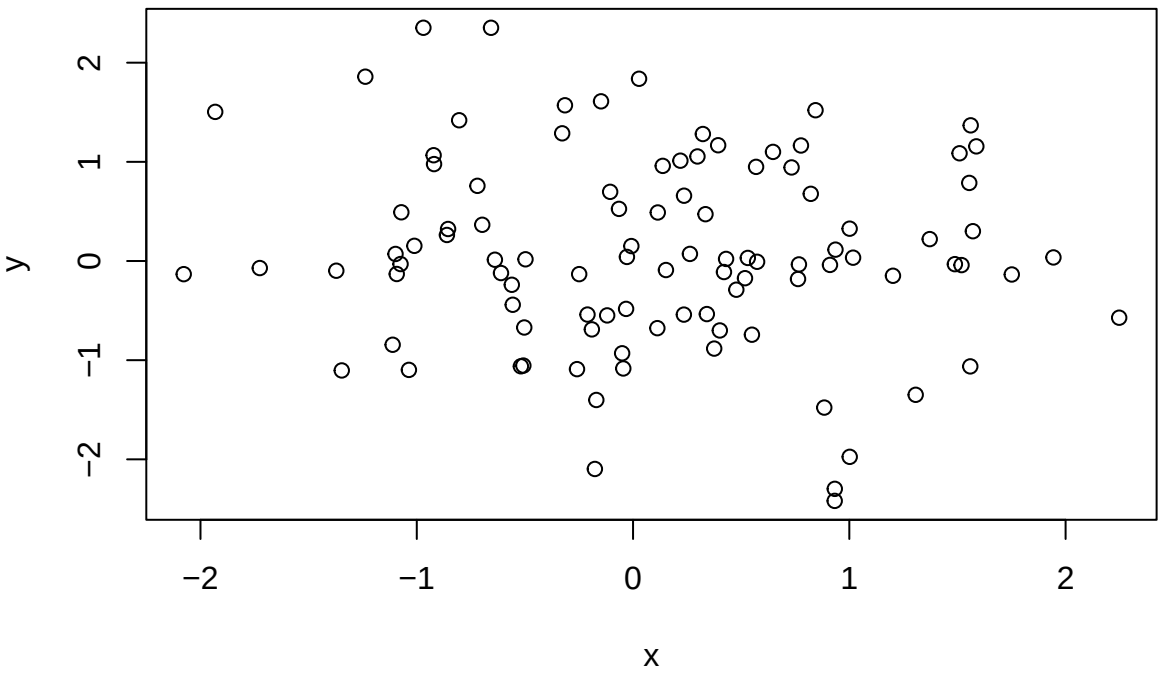
```
set.seed(1303)
rnorm(50)
```

```
##  [1] -1.1439763145  1.3421293656  2.1853904757  0.5363925179  0.0631929665
##  [6]  0.5022344825 -0.0004167247  0.5658198405 -0.5725226890 -1.1102250073
## [11] -0.0486871234 -0.6956562176  0.8289174803  0.2066528551 -0.2356745091
## [16] -0.5563104914 -0.3647543571  0.8623550343 -0.6307715354  0.3136021252
## [21] -0.9314953177  0.8238676185  0.5233707021  0.7069214120  0.4202043256
## [26] -0.2690521547 -1.5103172999 -0.6902124766 -0.1434719524 -1.0135274099
## [31]  1.5732737361  0.0127465055  0.8726470499  0.4220661905 -0.0188157917
## [36]  2.6157489689 -0.6931401748 -0.2663217810 -0.7206364412  1.3677342065
## [41]  0.2640073322  0.6321868074 -1.3306509858  0.0268888182  1.0406363208
## [46]  1.3120237985 -0.0300020767 -0.2500257125  0.0234144857  1.6598706557
```

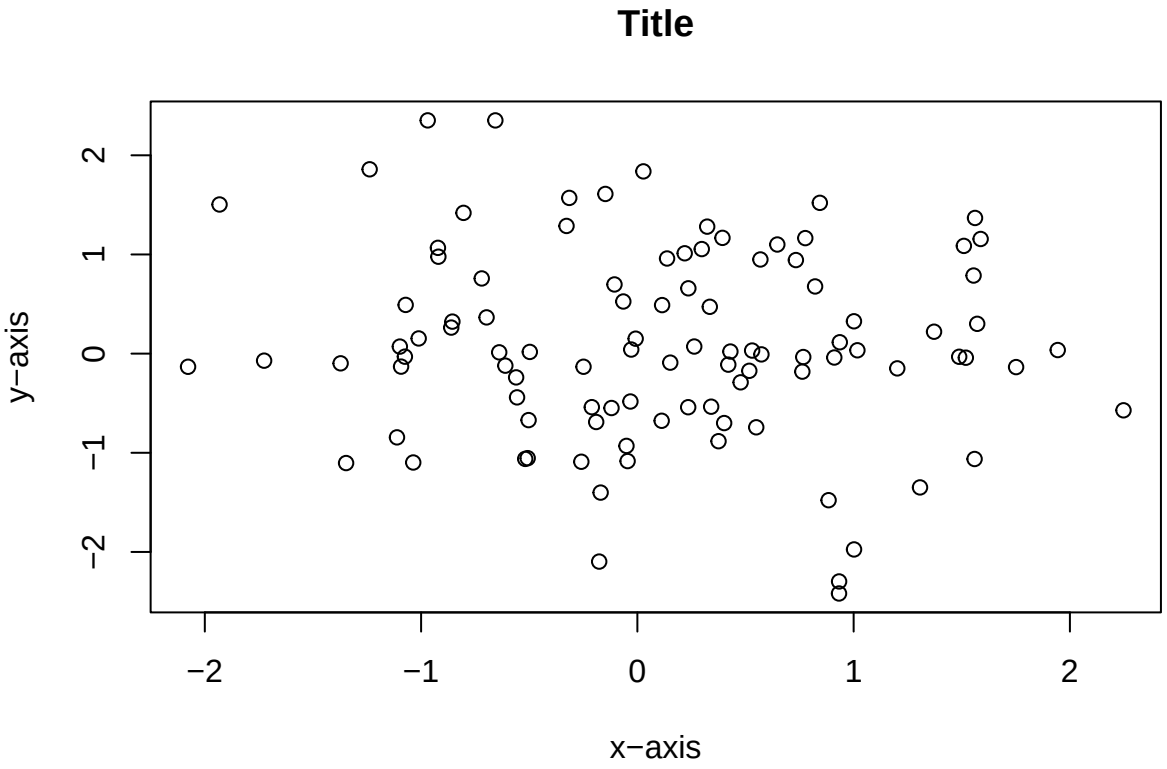
5 R 그래픽스

- plot(): 산점도 등을 그릴 때 쓰는 가장 대표적인 함수

```
x <- rnorm(100)
y <- rnorm(100)
plot(x,y)
```

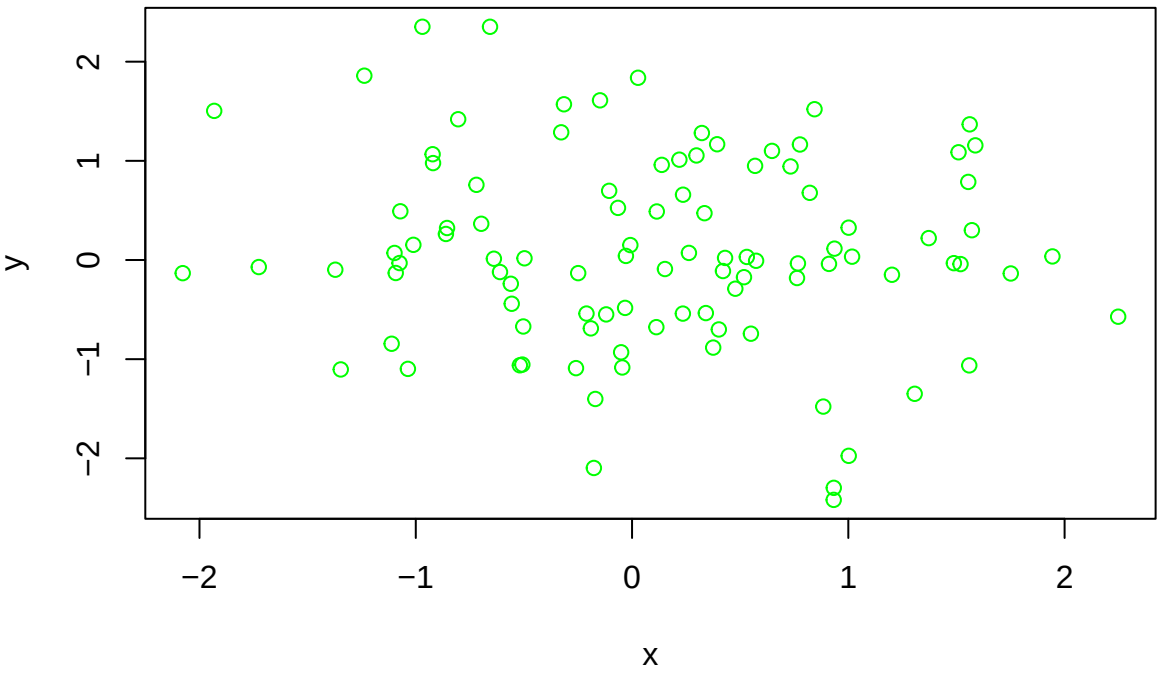


```
plot(x,y, xlab="x-axis", ylab="y-axis", main="Title")
```



- pdf(), jpeg(), png(): 자료를 그림파일로 저장

```
#pdf("Figure_example.pdf")
plot(x,y, col="green")
```

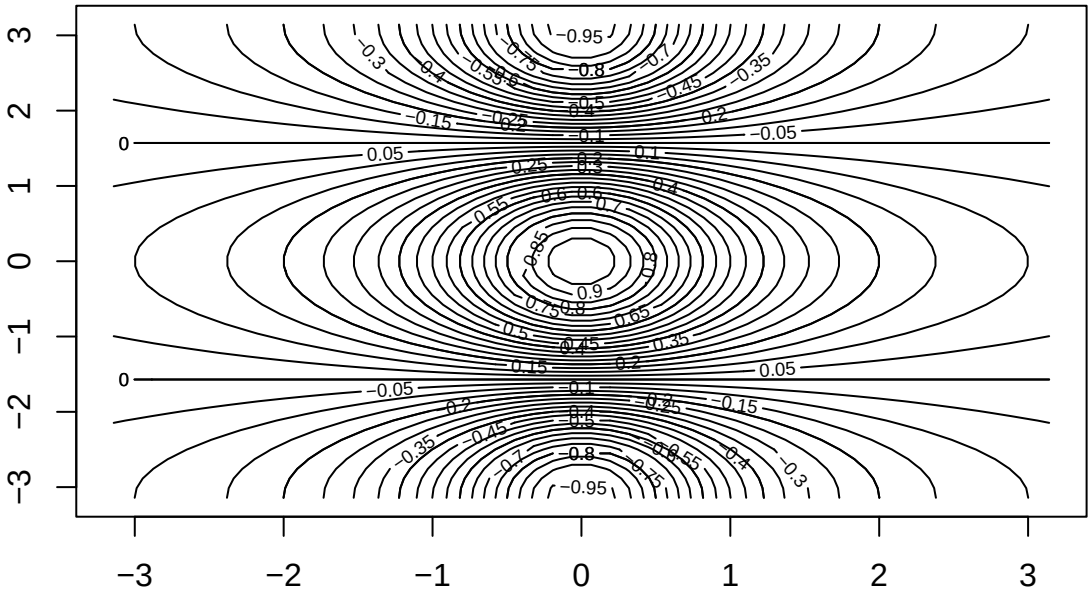



```
#dev.off() #꼭 넣어야 함
```

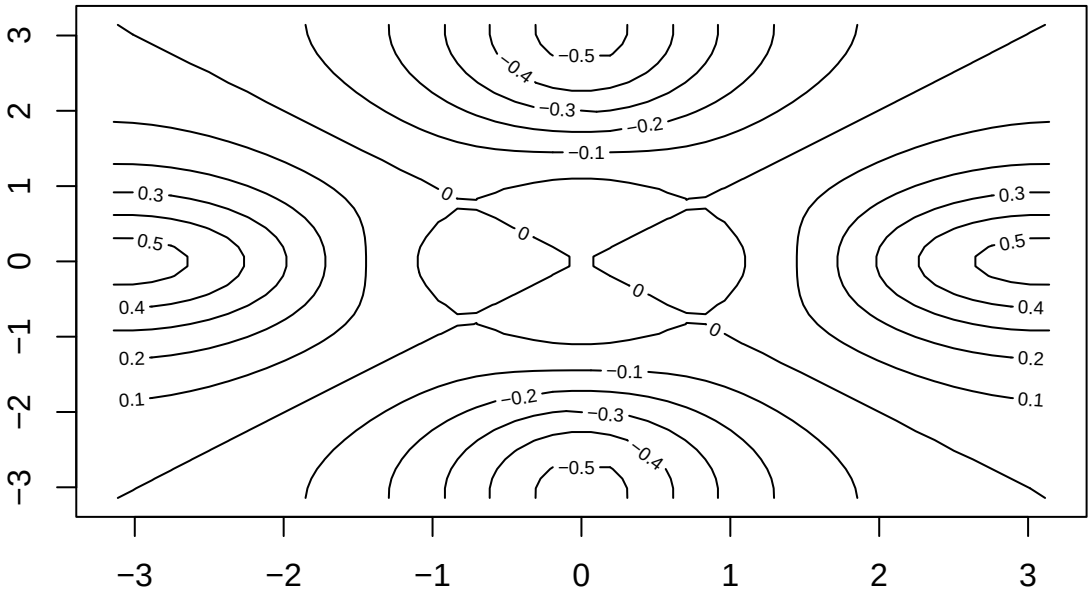
- 등고선 그림: contour()

```
x <- seq(-pi,pi,length=50)
y <- x
f <- outer(x, y, function(x, y) cos(y)/(1+x^2))
contour(x,y,f)
```

```
contour(x,y,f,nlevels=45,add=T) #nlevels: 등고선 레벨 지정
```

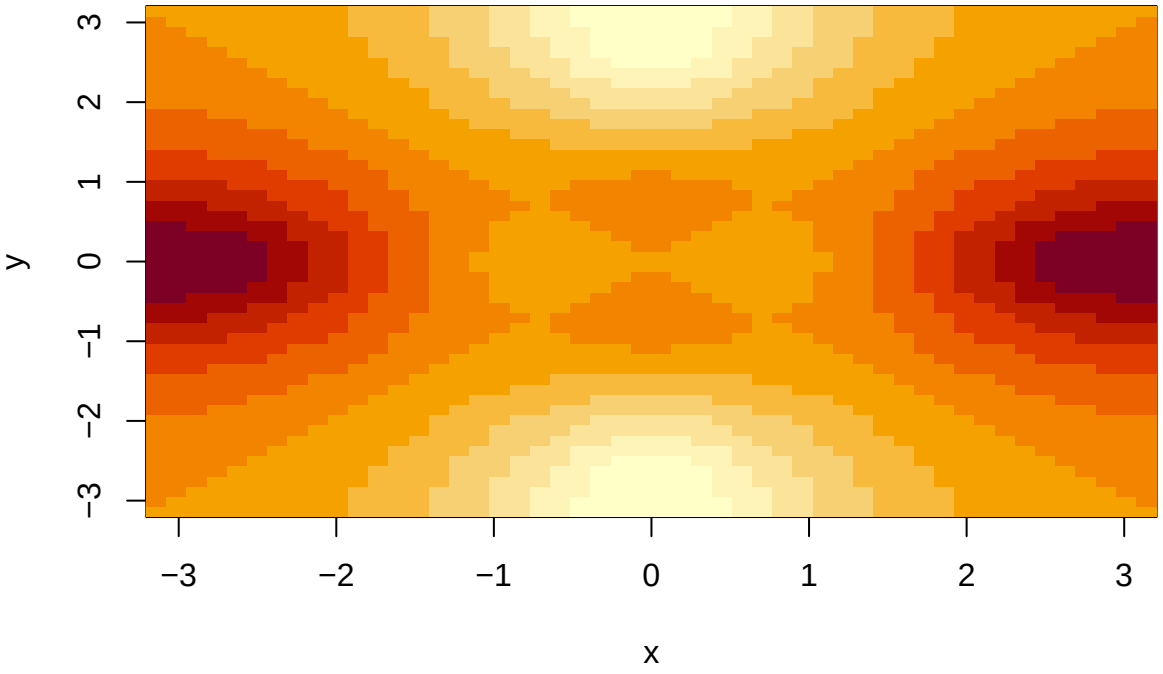


```
fa=(f-t(f))/2
contour(x,y,fa,nlevels=15)
```

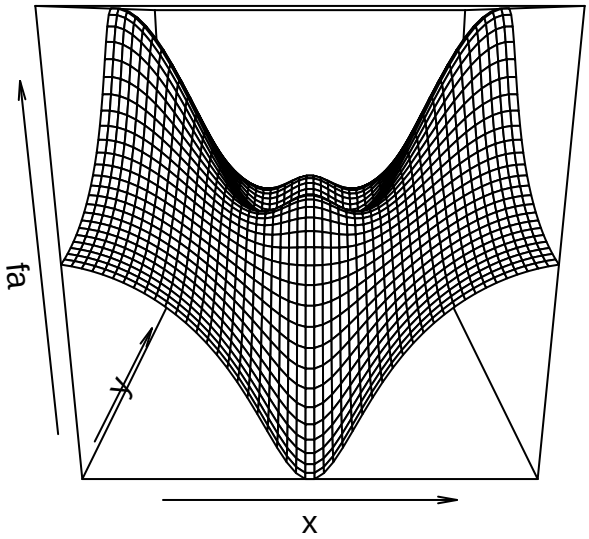


- 이미지 함수: `image()`, 참고로 행렬 등도 `image()` 함수로 표현 가능

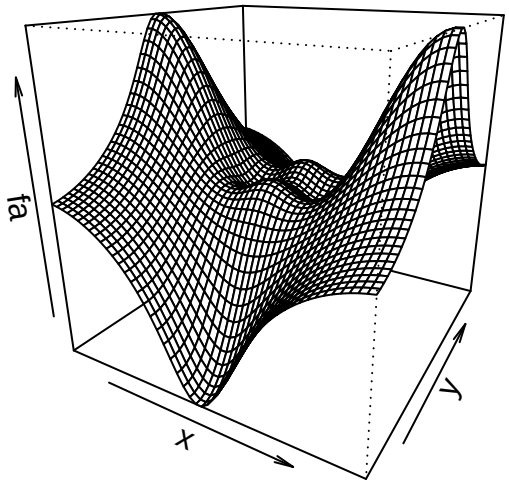
```
image(x,y,fa)
```




```
persp(x,y,fa) #3차원 버전
```




```
persp(x,y,fa,theta=30,phi=20)
```



6 자료의 인덱싱

- 파이썬과는 달리, R에서는 1,2,3,… 순서대로 벡터 및 행렬의 위치를 인덱싱한다.

```
A <- matrix(1:16, 4,4)
A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```
A[2,3] #10
```

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

- 범위지정으로 불러낼 수도 있다.

```
A[c(1,3),c(2,4)]
```

```
##      [,1] [,2]
## [1,]    5   13
## [2,]    7   15
```

```
A[1:3,2:4]
```

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

```
A [1:2 ,]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
```

```
A [ ,1:2]
```

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

```
A[1,]
```

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

```
A[-c(1,3),]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    6   10   14
## [2,]    4    8   12   16
```

```
dim(A)
```

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

7 자료 불러오기

- `setwd()`: 디렉토리 설정

```
OS <- .Platform$OS.type

if(OS == "unix"){
```

```
  #Mac
  setwd("~/Dropbox/Data/Books/An Introduction to Statistical Learning")
}else if(OS == "windows"){
  #Windows
  setwd("D:\\Dropbox\\Data\\Books\\An Introduction to Statistical Learning")
}
```

- `read.table()`: 가장 기본적인 데이터 불러오는 함수

```
Auto <- read.table("Auto.data")
head(Auto)
```

V1	V2	V3	V4	V5	V6	V7	V8	V9
mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
18.0	8	307.0	130.0	3504.	12.0	70	1	chevrolet chevelle malibu
15.0	8	350.0	165.0	3693.	11.5	70	1	buick skylark 320
18.0	8	318.0	150.0	3436.	11.0	70	1	plymouth satellite
16.0	8	304.0	150.0	3433.	12.0	70	1	amc rebel sst
17.0	8	302.0	140.0	3449.	10.5	70	1	ford torino

```
#fix(Auto)
dim(Auto)
```

```
## [1] 398   9
```

```
Auto[1:4,]
```

V1	V2	V3	V4	V5	V6	V7	V8	V9
mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
18.0	8	307.0	130.0	3504.	12.0	70	1	chevrolet chevelle malibu
15.0	8	350.0	165.0	3693.	11.5	70	1	buick skylark 320
18.0	8	318.0	150.0	3436.	11.0	70	1	plymouth satellite

위와 같이 부르면 첫 번째 열이 이상해진다. 이럴때에는 `header=T`라는 옵션을 쓴다.

```
Auto <- read.table("Auto.data", header=T, na.strings="?")
head(Auto)
```

mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
18	8	307	130	3504	12.0	70	1	chevrolet chevelle malibu
15	8	350	165	3693	11.5	70	1	buick skylark 320
18	8	318	150	3436	11.0	70	1	plymouth satellite
16	8	304	150	3433	12.0	70	1	amc rebel sst
17	8	302	140	3449	10.5	70	1	ford torino
15	8	429	198	4341	10.0	70	1	ford galaxie 500

```
#fix(Auto)
dim(Auto)
```

```
## [1] 397   9
```

```
Auto[1:4,]
```

mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
18	8	307	130	3504	12.0	70	1	chevrolet chevelle malibu
15	8	350	165	3693	11.5	70	1	buick skylark 320
18	8	318	150	3436	11.0	70	1	plymouth satellite
16	8	304	150	3433	12.0	70	1	amc rebel sst

- `na.omit()`: 결측치가 있을 때 그 부분 (또는 그 행)을 뺀다.

```
Auto <- na.omit(Auto)
dim(Auto)
```

```
## [1] 392   9
```

```
names(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

7.1 Auto 자료를 이용한 그래프 및 자료 요약

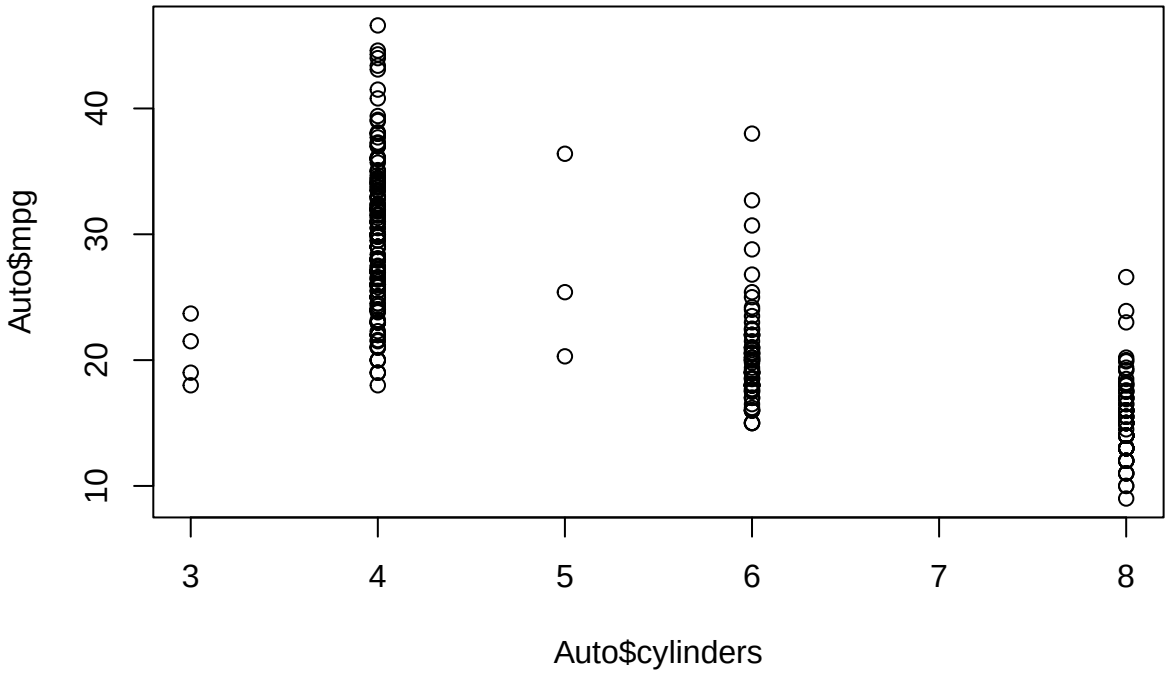
- 간단한 plot
- 우리는 Auto 안에 있는 cylinders라는 것에 대해 그림을 그리고 싶다. 그러나 다음과 같이 하면 오류가 난다.

```
plot(cylinders, mpg)
```

```
## Error in plot(cylinders, mpg): object 'cylinders' not found
```

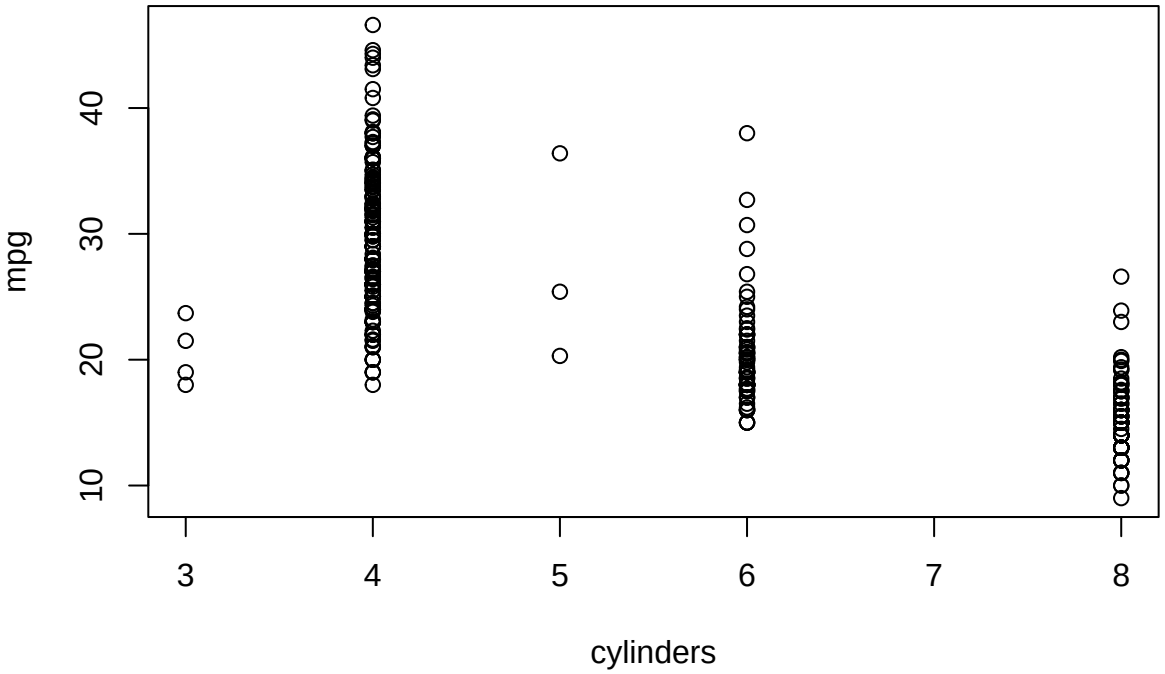
- 방법 1: 변수 이름을 부르기 위해 $를 쓴다.

```
plot(Auto$cylinders , Auto$mpg )
```



- 방법 2: attach 함수를 써서 cylinders변수를 만들게 한다.

```
attach(Auto)
plot(cylinders , mpg)
```




```
class(cylinders)
```

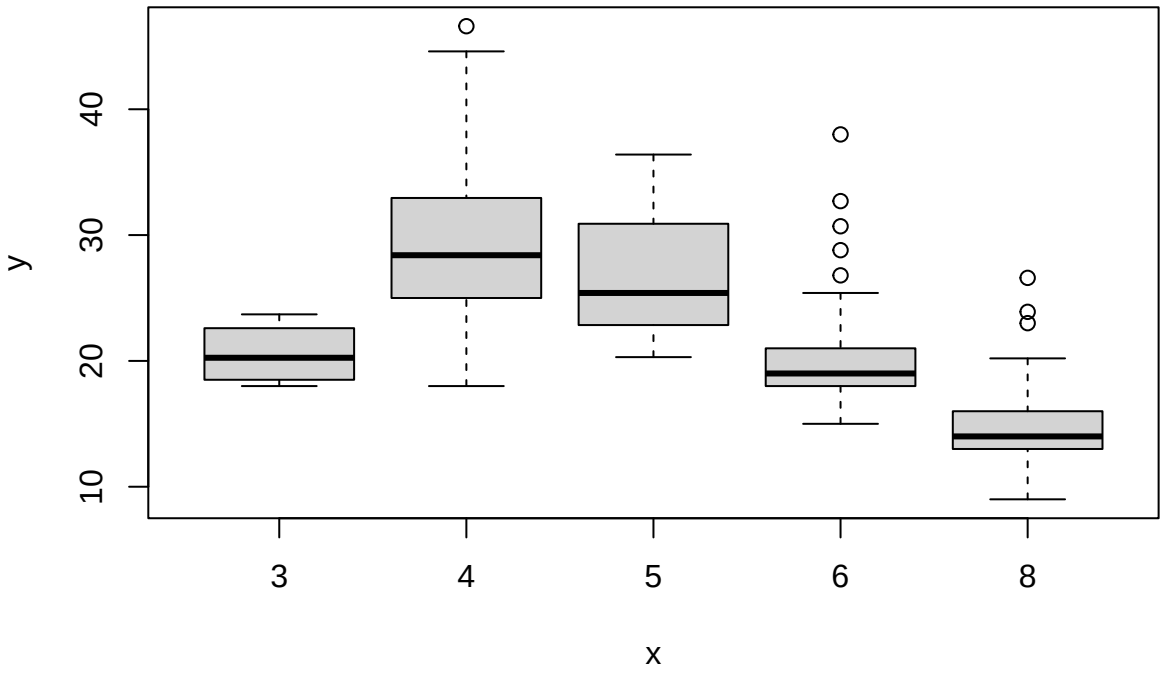
```
## [1] "integer"
```

- 지금 `cylinders`는 `numeric`으로 되어 있다. 그러나 `cylinders`가 갖을 수 있는 값이 몇 개 없으므로 이것을 `as.factor()` 함수를 통해 범주형 변수로 다룰 수 있다.

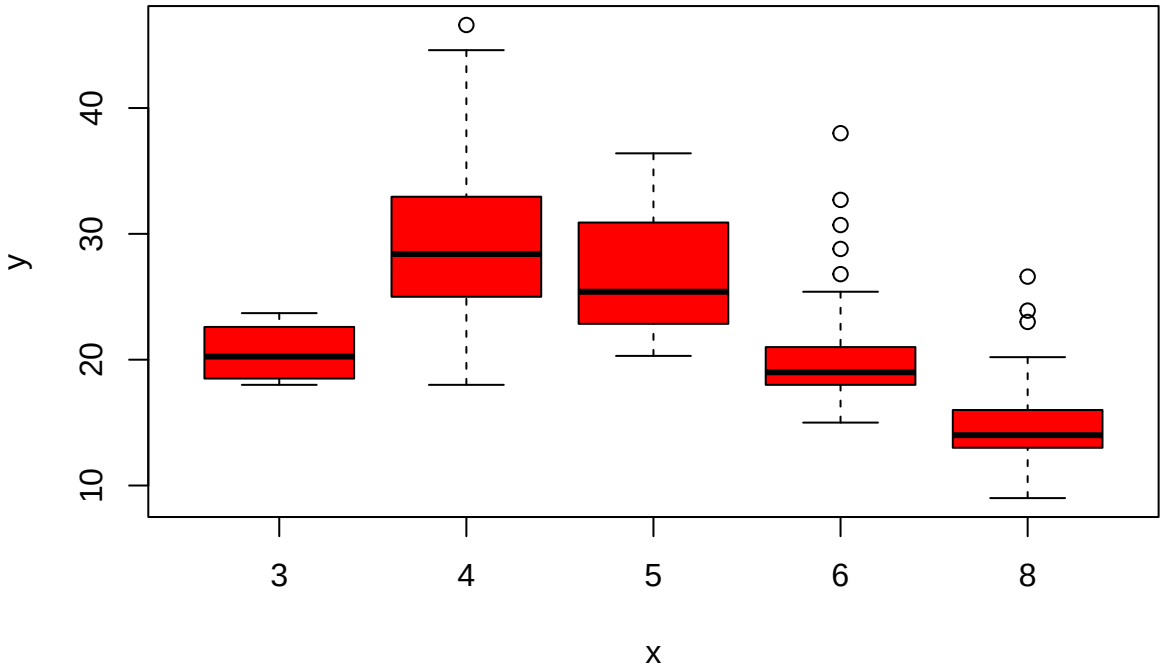
```
cylinders=as.factor(cylinders)
```

- 범주형 자료의 `plot()`은 자동으로 `boxplot`으로 바뀐다.

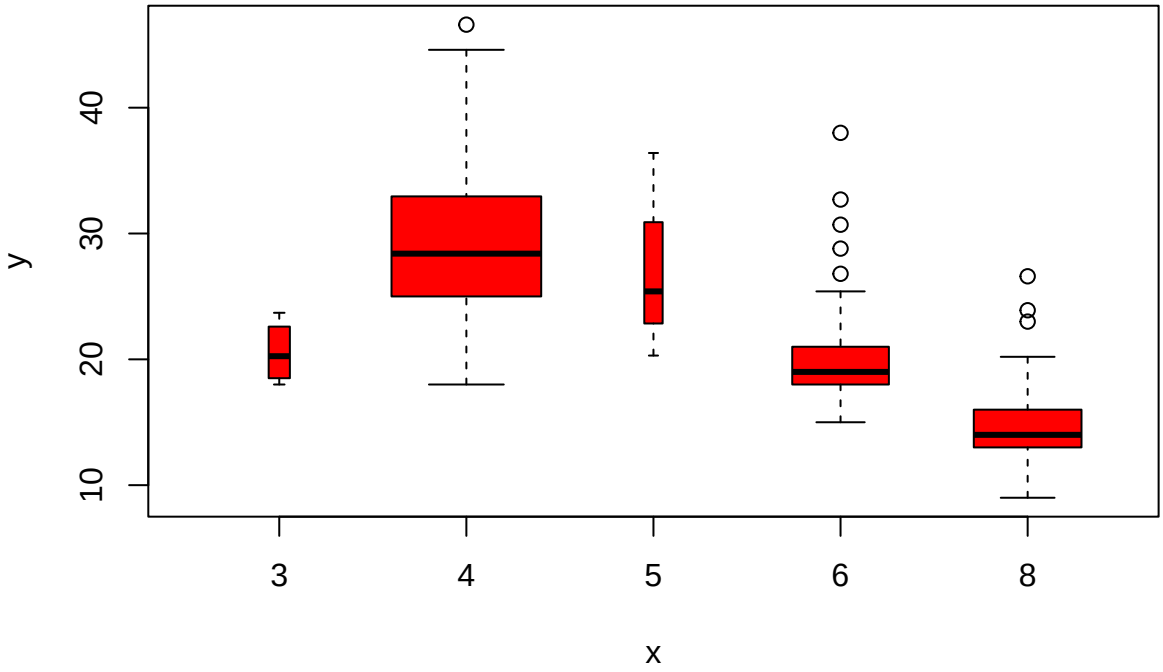
```
plot(cylinders, mpg)
```



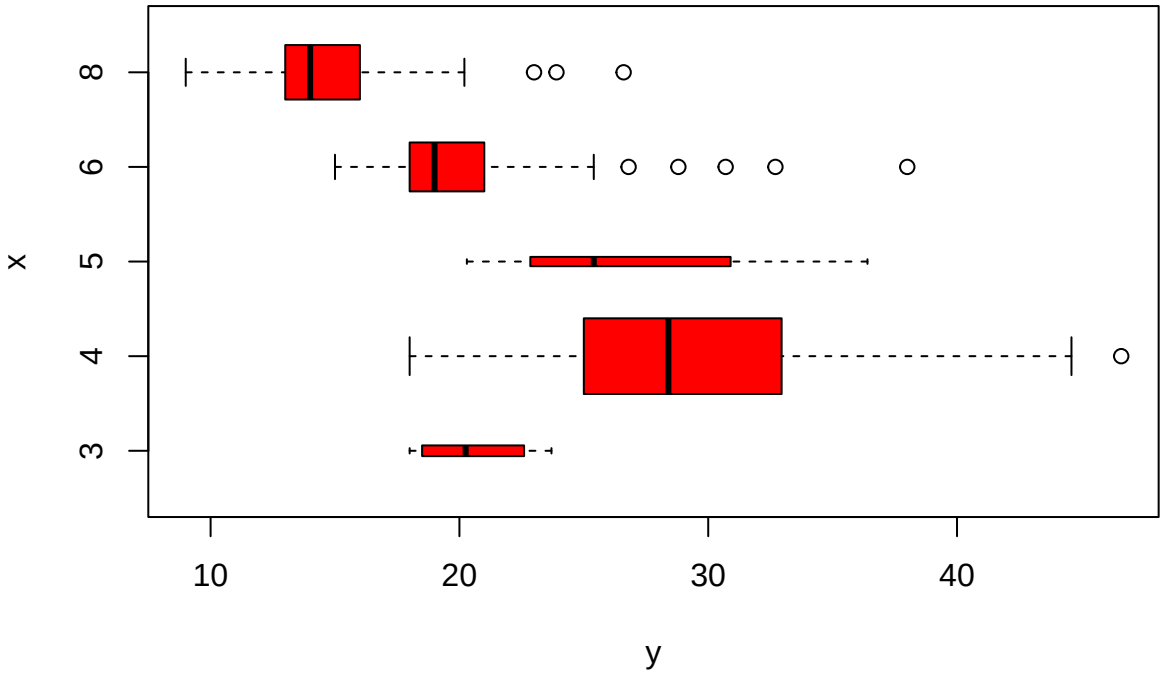

```
plot(cylinders, mpg, col="red")
```



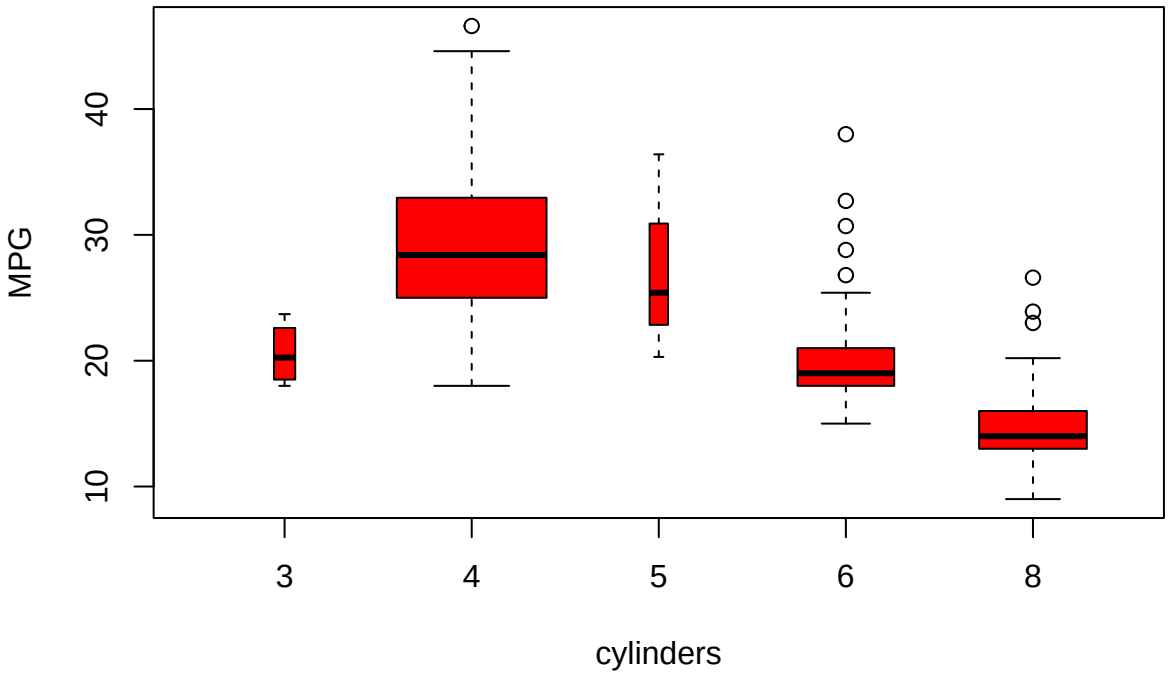

```
plot(cylinders, mpg, col="red", varwidth=T)
```




```
plot(cylinders, mpg, col="red", varwidth=T, horizontal=T)
```

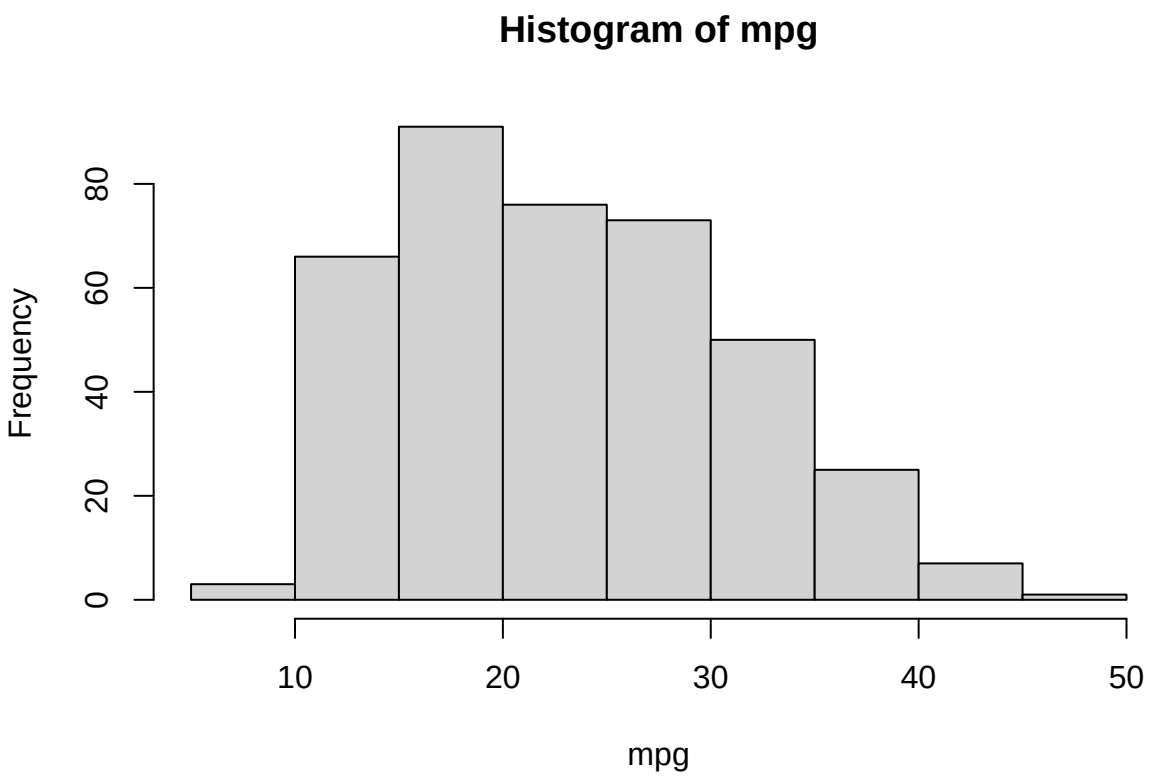



```
plot(cylinders, mpg, col="red", varwidth=T, xlab="cylinders", ylab="MPG")
```

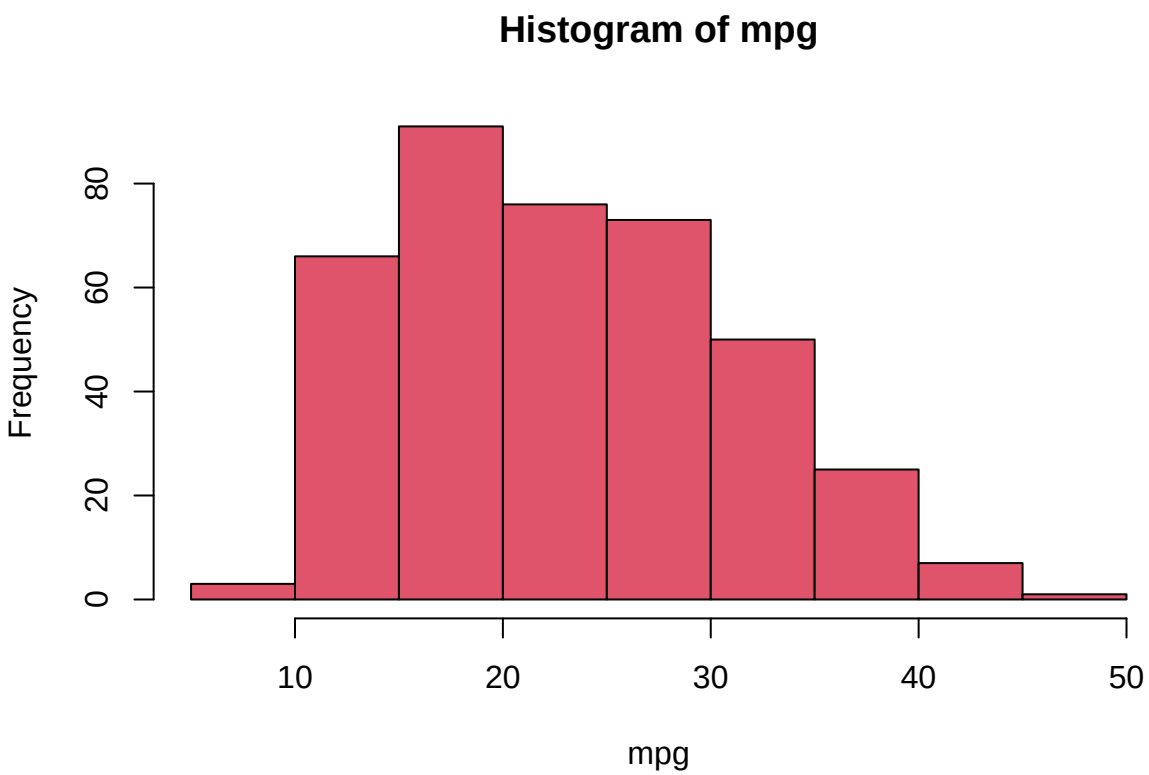


- `hist()`: 히스토그램 함수

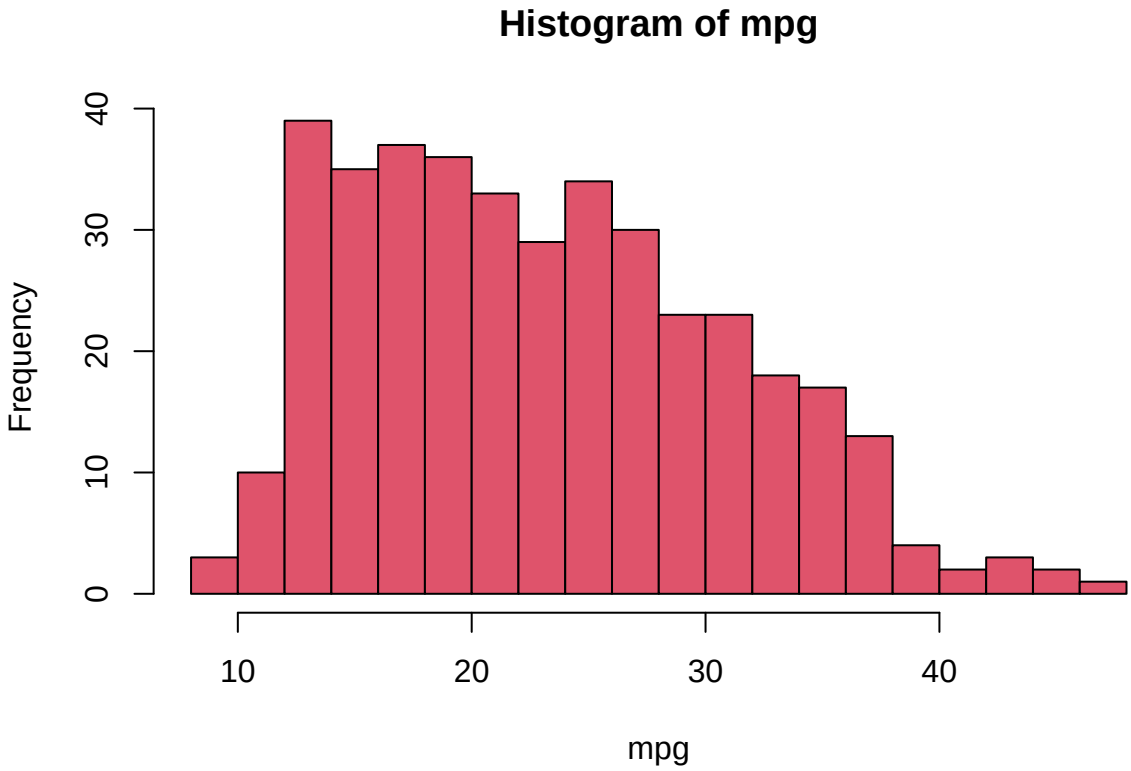
```
hist(mpg)
```




```
hist(mpg, col=2)
```

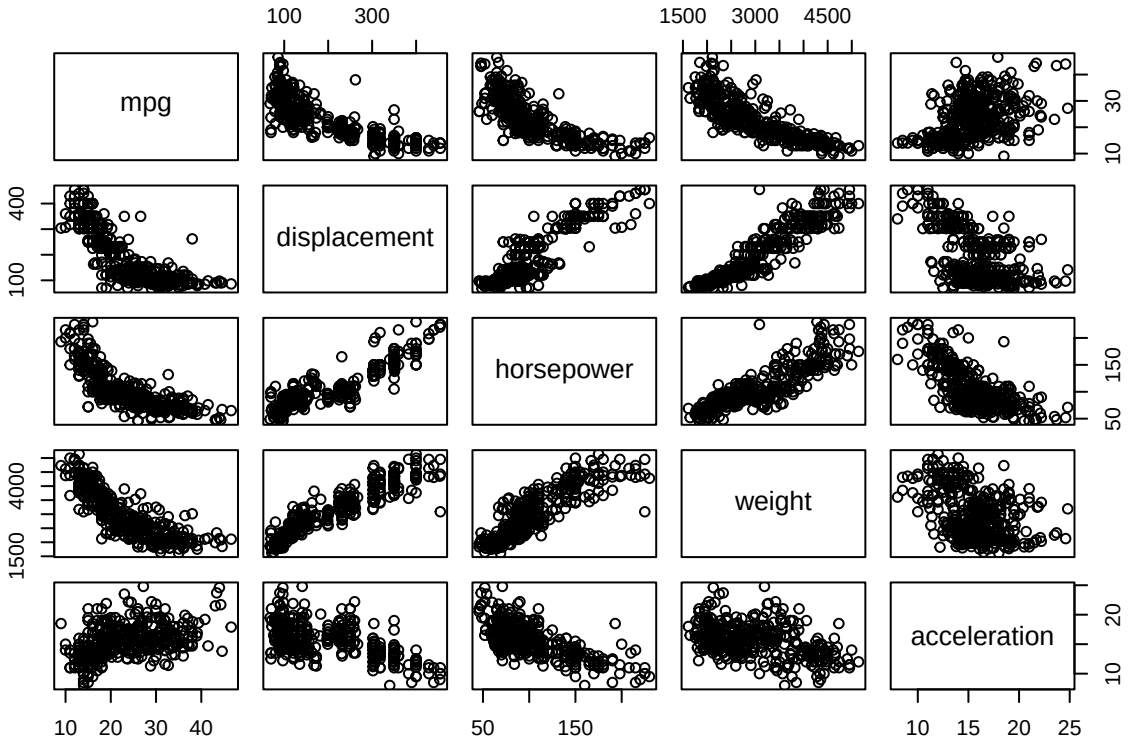



```
hist(mpg, col=2, breaks=15)
```



- `pairs()`: 주어진 자료에 있는 변수들의 모든 `scatterplot()`을 그려줌

```
pairs(~ mpg + displacement + horsepower + weight + acceleration, Auto)
```



- `summary()`: 자료 요약 함수

```
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
```

```
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##   acceleration        year           origin          name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   Length:392
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   Class :character
##  Median :15.50   Median :76.00   Median :1.000   Mode  :character
##  Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :24.80   Max.   :82.00   Max.   :3.000
```

```
summary(mpg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   17.00   22.75   23.45   29.00   46.60
```

8 초보자가 주의해야 할 점들

- R은 자료의 대소문자를 구분한다. 아래 코드는 x와 X가 다르기 때문에 오류가 발생한다.

```
x <- rnorm(100)
summary(X)
```

```
## Error in summary(X): object 'X' not found
```

```
summary(x)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.83937 -0.73626 -0.04952  0.01001  0.79687  3.28639
```

- 작업 파일의 위치를 제대로 지정하였는지 항상 확인하여야 한다.

```
getwd()
```

```
## [1] "/Users/seoncheolpark/Dropbox/Data/Books/An Introduction to Statistical Learning"
```

- 어떤 함수들은 package를 꼭 설치해야 한다.

```
a <- mpfr(1/3, 100)
```

```
## Error in mpfr(1/3, 100): could not find function "mpfr"
```

```
#install.packages(Rmpfr)
#library(Rmpfr)
```

- 괄호의 사용
  - "()": 함수의 인수부분 (투입값)을 지칭할 때
  - "[]": 벡터나 행렬 등에서 특정 요소를 지칭할 때
  - "{}": for나 while문에서 2개 이상의 R 문을 블록으로 묶을 때

```
log(exp(1))
```

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

```
log[exp(1)] #오류
```

```
## Error in log[exp(1)]: object of type 'special' is not subsettable
```

- 곱셈기호 *의 누락: 필기로 계산할 때에는 곱셈기호를 생략할 때가 많지만 R에서는 곱셈기호를 항상 사용해줘야 한다.

```
n <- 10
n2 <- 2n #오류
n2 <- 2*n
```

```
## Error: <text>:2:8: unexpected symbol
## 1: n <- 10
## 2: n2 <- 2n
##           ^
```

Advanced R

9 R의 데이터 구조

	동질적	비동질적
1d	벡터	리스트
2d	행렬	데이터 프레임
nd	Array	

9.1 벡터

- `c()`: atomic vector 생성

9.2 타입 체크

1. `typeof()` 함수로 체크
2. 특정 타입의 이름 앞에 `is.`를 붙인다: `is.character()`, `is.double()`, `is.integer()`, `is.logical()` 등

```
int_var <- c(1L, 6L, 10L)
typeof(int_var)
```

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

```
is.integer(int_var)
```

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

```
dbl_var <- c(1, 2.5, 4.5)
typeof(dbl_var)
```

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

```
is.double(dbl_var)
```

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

9.3 여러 데이터 타입이 섞이거나 또는 변환될 경우

- `character`와 `integer`가 섞이면 `character`가 된다.

```
str(c("a", 1))
```

```
##  chr [1:2] "a" "1"
```

- `logical` vector가 `integer` 또는 `double`로 변환하면, `TRUE`는 1, `FALSE`는 0이 된다. 결국 `numeric`처럼 쓸 수 있는 것이다.

```
x <- c(FALSE, FALSE, TRUE)
as.numeric(x)
```

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

```
sum(x)
```

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

```
mean(x)
```

```
## [1] 0.3333333
```

9.4 리스트

- (개인적인 의견) R을 다른 프로그램과 가르는 핵심적인 요소가 `list`이다. 물론 다른 프로그래밍 언어에도 `list`와 비슷한 역할을 하는 요소가 많이 있지만 R의 리스트는 편리하기도 하고 다양한 기능이 있다.

```
x <- list(1:3, "a", c(TRUE, FALSE, TRUE), c(2.3, 5.9))
str(x)
```

```
## List of 4
##  $ : int [1:3] 1 2 3
##  $ : chr "a"
##  $ : logi [1:3] TRUE FALSE TRUE
##  $ : num [1:2] 2.3 5.9
```

```
#리스트 원소를 부를땐 [[]]를 쓴다
x[[1]]
```

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

- 리스트 안에 또 리스트를 넣을 수 있다.

```
y <- list(list(list(list())))
str(y)
```

```
## List of 1
##  $ :List of 1
##   ..$ :List of 1
##   .. ..$ : list()
```

- c() 함수는 리스트 또한 결합할 수 있다.

```
x <- list(list(1,2), c(3,4))
y <- c(list(1,2), c(3,4))
str(x)
```

```
## List of 2
##  $ :List of 2
##   ..$ : num 1
##   ..$ : num 2
##  $ : num [1:2] 3 4
```

```
str(y)
```

```
## List of 4
##  $ : num 1
##  $ : num 2
##  $ : num 3
##  $ : num 4
```

- 우리가 잘 아는 `lm` 오브젝트 또한 `list`이다.

```
is.list(mtcars)
```

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

```
mod <- lm(mpg ~ wt, data = mtcars)
```

9.5 할당 (attributes)

모든 오브젝트들은 특정할 일을 하도록 할당이라는 것을 할 수 있다. 이것 또한 특별한 이름을 가진 `list`라고 볼 수 있다고 한다.

```
y <- 1:10
attr(y, "my_attribute") <- "This is a vector"
attr(y, "my_attribute")
```

```
## [1] "This is a vector"
```

```
str(attributes(y))
```

```
## List of 1
##  $ my_attribute: chr "This is a vector"
```

9.6 데이터 프레임

- data.frame() 함수로 작성

```
df <- data.frame(x = 1:3, y = c("a", "b", "c"))
str(df)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ x: int  1 2 3
##  $ y: chr  "a" "b" "c"
```

- 데이터 프레임에 리스트를 넣을 때

```
df <- data.frame(x = 1:3)
df$y <- list(1:2, 1:3, 1:4)
df
```

x	y
1	1, 2
2	1, 2, 3
3	1, 2, 3, 4

```
data.frame(x = 1:3, y = list(1:2, 1:3, 1:4))
```

```
## Error in (function (..., row.names = NULL, check.rows = FALSE, check.names = TRUE, : arguments imply differi
```

10 Subsetting

10.1 논리연산과 관련된 operator들

- &: and
- |: or
- all: 모든 값들이 참인가?
- any: 적어도 한 개가 참인가?

```
#집에서 계산해보세요.
TRUE & TRUE
TRUE & FALSE
FALSE & FALSE
TRUE | TRUE
TRUE | FALSE
FALSE | FALSE
all(c(TRUE, TRUE))
all(c(FALSE, TRUE))
all(c(FALSE, FALSE))
any(c(TRUE, TRUE))
any(c(FALSE, TRUE))
any(c(FALSE, FALSE))
```

- !: 결과를 바꿀 때 쓴다.

```
#집에서 계산해보세요.
X <- TRUE
Y <- FALSE
!(X & Y)
!(X | Y)
```

10.2 which()

- 특정 조건을 만족하는 원소를 찾을 때 쓴다. 결과는 특정 조건을 만족하는 값의 위치를 반환한다.

```
x <- c(1:10)
which(x < 5)
```

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

- Variation으로 `which.min()`, `which.max()`가 있다. 참고로 `which`류들은 값을 반환하는 것이 아닌 벡터에서 특정 조건을 만족하는 값의 위치를 반환한다는 것에 주의하자.

```
x <- c(1:10, 5:15, -3:-1)
which.min(x)
```

```
## [1] 22
```

```
which.max(x)
```

```
## [1] 21
```

10.3 rev(), sort(), order() 등

- `rev()`: 벡터의 순서를 뒤집을 때 쓴다.

```
rev(c(1:10))
```

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

- `sort()`: 정렬할 때 쓴다.

```
sort(x)
```

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

```
sort(x, decreasing = TRUE)
```

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

- `order()`: 가장 작은 수부터 위치를 말해주는 함수

```
z <- c(3,4,1,5,9,10,2,7,6,8)
order(z)
```

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

- `union()`: 합집합

```
z1 <- c(3,4,1,5,9)
z2 <- c(10,2,7,6,8)
union(z1, z2)
```

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

- `intersect()`: 교집합

```
z1 <- c(3,4,1,5,9,11)
z2 <- c(10,2,7,6,8,11)
intersect(z1, z2)
```

```
## [1] 11
```

- `setdiff()`: 차집합

```
setdiff(z1, z2)
```

```
## [1] 3 4 1 5 9
```

10.4 글자와 관련된 matching들

- `%in%` 함수: 특정 문자열이 있는지 체크해주고 `TRUE`, `FALSE`를 반환

```
sstr <- c("c","ab","B","bba","c",NA,"@","bla","a","Ba","%")
sstr[sstr %in% c(letters, LETTERS)]
```

```
## [1] "c" "B" "c" "a"
```

- `subset`: `character`의 부분집합을 잡아준다.

```
substr("abcdef", 2, 4)
```

```
## [1] "bcd"
```

- strsplit: 자료를 처리할 때, 때때로 특정 문자를 기준으로 character를 갈라놓아야 할 때가 있는데 그럴 때 쓴다.

```
name_vec <- c("Cheongju-si", "Seowon-gu", "Gaesin-dong")
strsplit(name_vec, "-")
```

```
## [[1]]
## [1] "Cheongju" "si"
##
## [[2]]
## [1] "Seowon" "gu"
##
## [[3]]
## [1] "Gaesin" "dong"
```

```
unlist(strsplit(name_vec, "-")) #unlist: 리스트 풀때
```

```
## [1] "Cheongju" "si"       "Seowon"   "gu"       "Gaesin"   "dong"
```

- subset: 말 그대로 부분집합이며 특정 조건을 만족하는 집합을 찾는 것이다.

```
head(subset(airquality, Temp > 80, select = c(Ozone, Temp)))
```

	Ozone	Temp
29	45	81
35	NA	84
36	NA	85
38	29	82
39	NA	87
40	71	90

11 조건문과 apply류 함수들

11.1 for문

- for문은 반복연산시 사용한다.

```
y <- 0
for(i in 1:10){
  y <- y+i
  cat("summation from 0 to ", i, " is ", y, "\n", sep="")
}
```

```
## summation from 0 to 1 is 1
## summation from 0 to 2 is 3
## summation from 0 to 3 is 6
## summation from 0 to 4 is 10
## summation from 0 to 5 is 15
## summation from 0 to 6 is 21
## summation from 0 to 7 is 28
## summation from 0 to 8 is 36
## summation from 0 to 9 is 45
## summation from 0 to 10 is 55
```

11.2 if문

- if문은 else if, else 등과 같이 써서 보통 분기 조건을 나타낼 때 쓴다.

```
x <- 0
if (x < 0) {
  print("Negative number")
```

```
} else if (x > 0) {
  print("Positive number")
} else {
  print("Zero")
}
```

```
## [1] "Zero"
```

11.3 while문

- while문은 해당 조건을 만족할 때까지 계속 계산을 하기 때문에 만약 만족 불가능 (아니면 거의 힘든) 조건을 넣었을 경우 무한히 계산만 하게 된다. 그럴 경우를 방지하기 위해 중간에 break를 넣어주는 게 좋다.

```
j <- 1
while(TRUE){
  j <- j+3
  if(j > 5){
    break
  }
}
j
```

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

11.4 apply 함수와 그 친구들

- 교재에 나와있듯이, R에서는 일반적으로 for 문을 쓰는 것보다 apply 류의 함수를 쓰는 것이 계산 속도가 조금은 빠름이 알려져 있다.
- apply: array나 matrix와 같이 쓴다.

```
A <- matrix(c(1:16), nrow=4, ncol=4)
apply(A, 1, sum)
```

```
## [1] 28 32 36 40
```

```
apply(A, 2, sum)
```

```
## [1] 10 26 42 58
```

- lapply: 리스트에 쓰는 apply 함수라고 생각하면 편하다. 반환도 리스트로 한다.

```
x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))
lapply(x, mean)
```

```
## $a
## [1] 5.5
##
## $beta
## [1] 4.535125
##
## $logic
## [1] 0.5
```

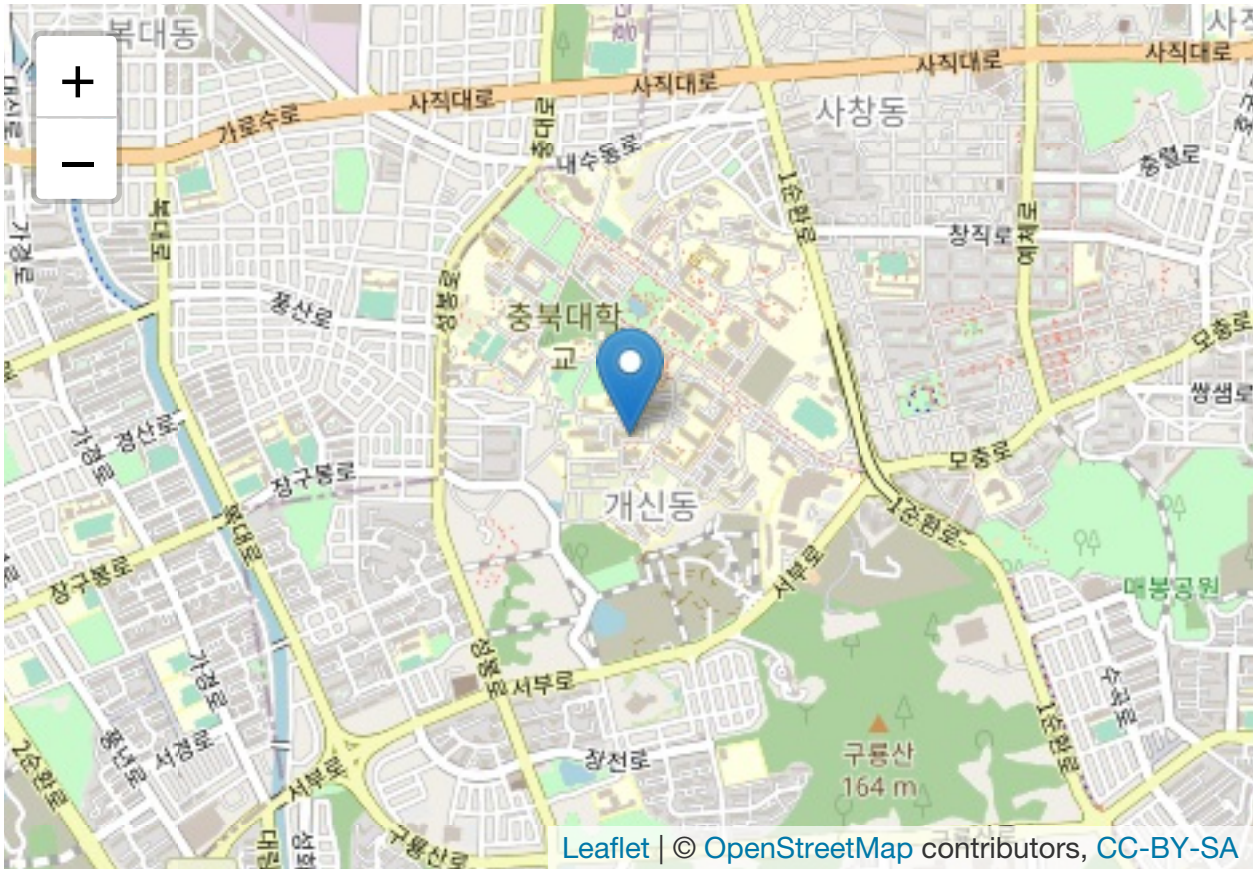
- sapply: lapply와 거의 같은 역할을 하나 결과가 벡터나 matrix 꼴로 나온다는 점에서 사용자가 사용하기 편하다.

```
region_vec <- c("Chungcheongbuk-do", "Chungcheongnam-do", "Daejeon-si")
sapply(region_vec, function(x) substr(x, start=1, stop=11)=="Chungcheong")
```

```
## Chungcheongbuk-do Chungcheongnam-do        Daejeon-si
##              TRUE              TRUE             FALSE
```

12 R 패키지 leaflet

```
library(leaflet)

m <- leaflet()
m <- addTiles(m)
m <- addMarkers(m, lng=127.45587, lat=36.62558, popup="CBNU Dept. of Information Statistics")
m
```



12.1 leaflet과 point data

12.1.1 청주시 미세먼지 측정소 위치 출력해보기

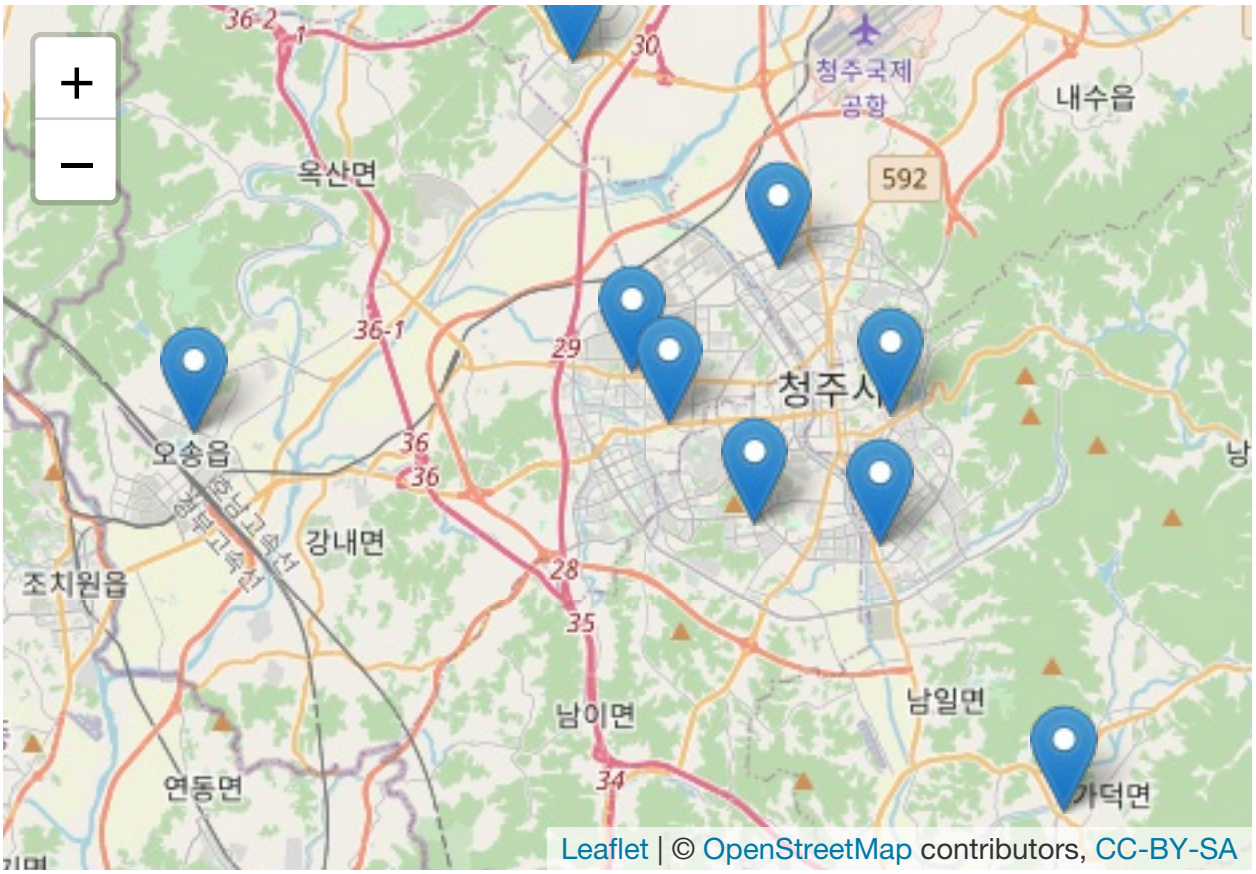
에어코리아로부터 다음과 같은 측정소 정보를 얻었다. 미세먼지 농도는 2021년 3월 19일 오전 2시 기준 미세먼지 농도이다.

측정소명	영문측정소명	타입	주소	경도	위도	미세먼지
오창읍	Ochang-eup	도시대기	충북 청주시 청원구 오창읍 각리 637-4 각리중학교 4층 (오창중앙로 48)	127.4219	36.7088	105
사천동	Sacheon-dong	도시대기	충북 청주시 청원구 사천동 233-223번지 (사뜸로 61번길 88-14), 청주청원도서관 옥상(2층)	127.4749	36.6662	79
용담동	Yongdam-dong	도시대기	충북 청주시 상당구 교동로 139번길 20	127.5040	36.6360	58

측정소명	영문측정소명	타입	주소	경도	위도	미세먼지
용암동	Yongam-dong	도시대기	충북 청주시 상당구 용암동 1590(중흥로 29), 용암1동 주민센터 3층 옥상	127.5013	36.6088	53
가덕면	Gadeok-myeon	도시대기	충청북도 청주시 상당구 가덕면 보청대로 4650	127.5488	36.5536	53
산남동	Sannam-dong	도시대기	충청북도 청주시 서원구 원흥로 81	127.4688	36.6132	48
송정동(봉명동)	Bongmyeong-dong	도시대기	충북 청주시 흥덕구 직지대로 393(송정동)	127.4371	36.6446	104
복대동	Bokdae-dong	도로변대기	충북 청주시 흥덕구 복대동 111, 산단6거리 교차점 광장	127.4471	36.6344	56
오송읍	Osong-eup	도시대기	충청북도 청주시 흥덕구 오송읍 오송생명로 150	127.3250	36.6319	119

이번엔 이 자료를 `leaflet` 패키지로 표현해 본다.

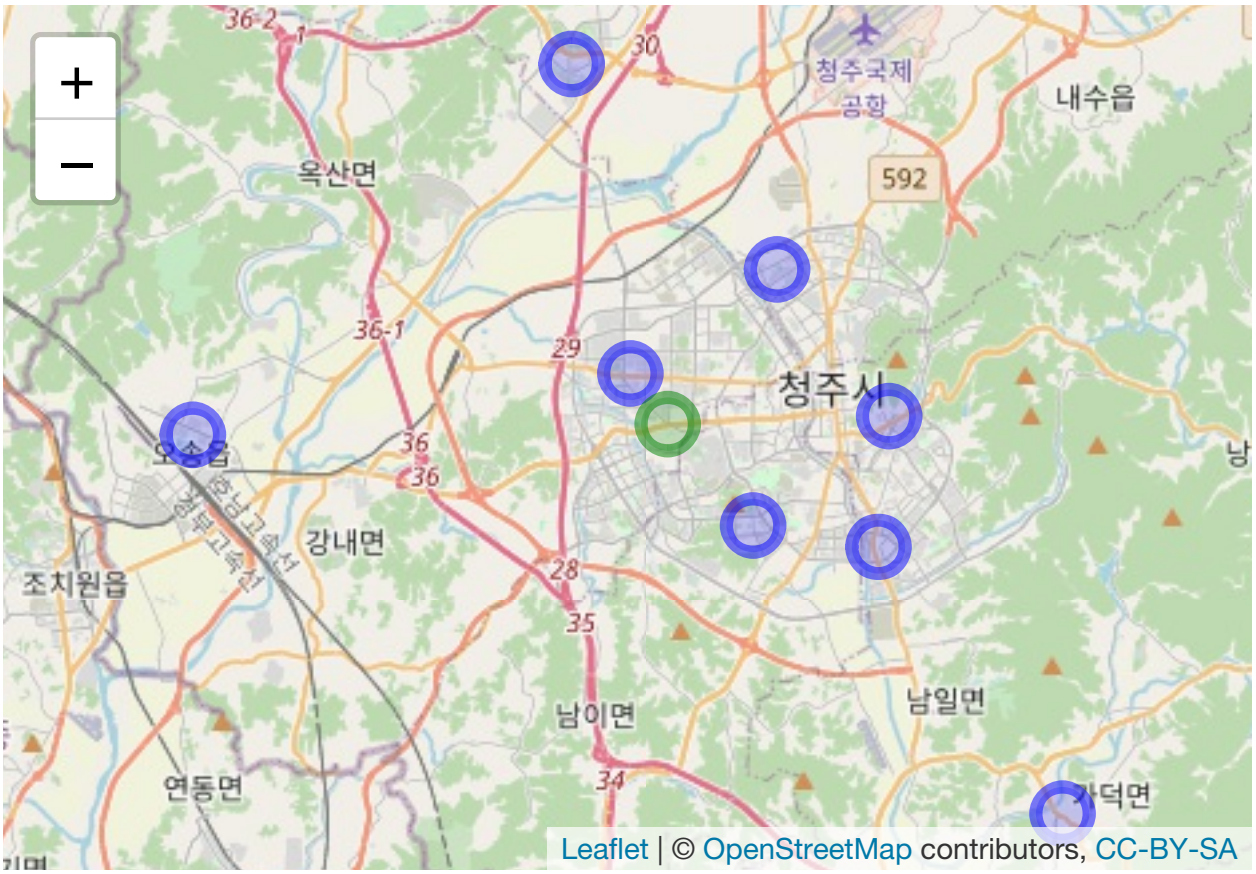
```
location_Cheongju <- data.frame(
  name = c("Ochang-eup", "Sacheon-dong", "Yongdam-dong", "Yongam-dong", "Gadeok-myeon",
           "Sannam-dong", "Bongmyeong-dong", "Bokdae-dong", "Osong-eup"),
  type = c(rep("City", 7), "Road", "City"),
  long = c(127.4219, 127.4749, 127.5040, 127.5013, 127.5488, 127.4688, 127.4371, 127.4471, 127.3250),
  lat = c(36.7088, 36.6662, 36.6360, 36.6088, 36.5536, 36.6132, 36.6446, 36.6344, 36.6319),
  PM10 = c(105, 79, 58, 53, 53, 48, 104, 56, 119)
)

m <- leaflet()
m <- addTiles(m)
m <- addMarkers(m, lng=location_Cheongju$long, lat=location_Cheongju$lat, popup=location_Cheongju$name)
m
```



12.1.2 마커의 색상 바꾸기

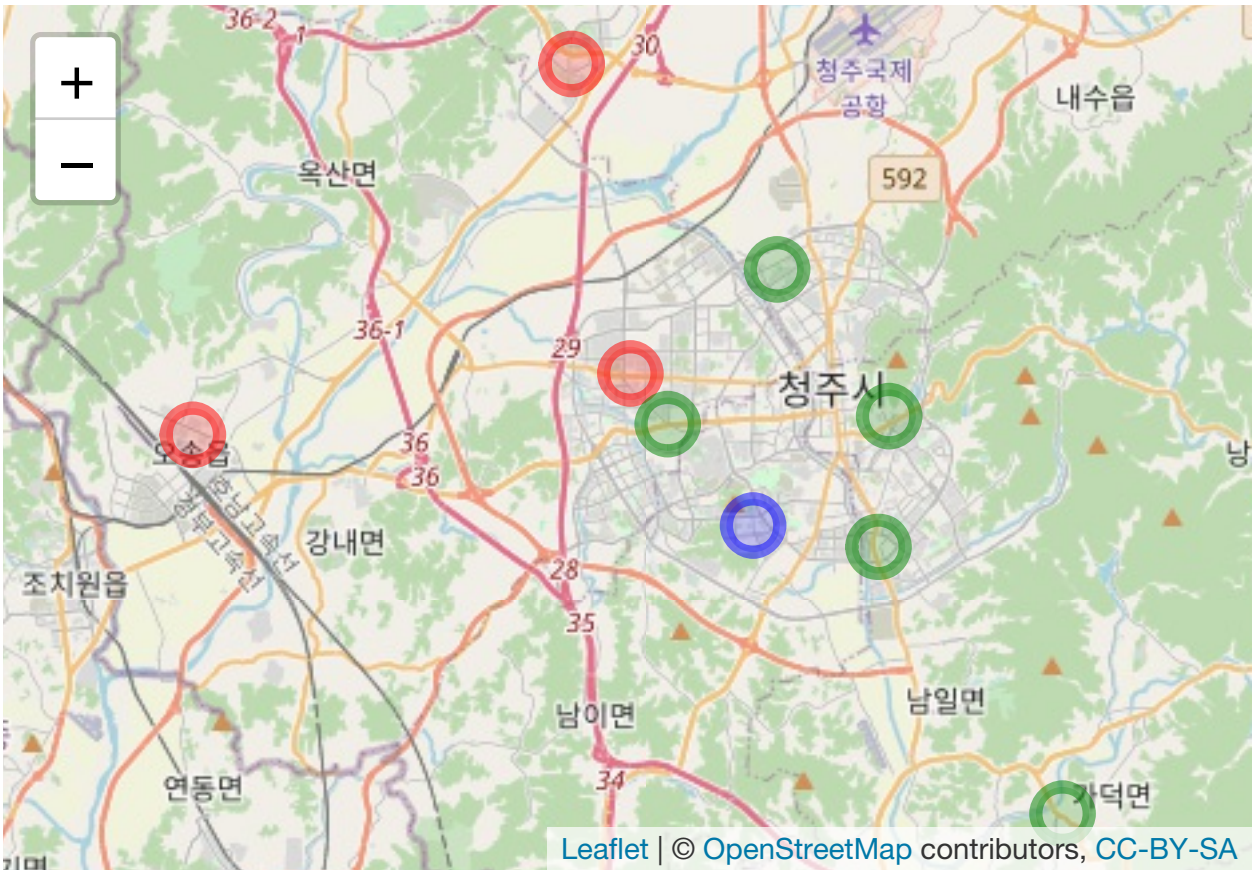
그런데 우리의 자료 중 복대동 측정소는 도로변 대기 측정소이다. 이를 구분하기 위해 마커의 색상을 바꿔보자.

```
colors <- ifelse(location_Cheongju$type=="City", "blue", "green")

m <- leaflet()
m <- addTiles(m)
m <- addCircleMarkers(m, lng=location_Cheongju$long, lat=location_Cheongju$lat, popup=location_Cheongju$name, c
m
```



이번엔 미세먼지의 농도에 따라 색깔을 바꿔자. 미세먼지 농도가 50 이하이면 파란색, 50~100이면 녹색, 100 이상이면 빨간색을 쓰기로 한다.

```
#  PM10 = c(105, 79, 58, 53, 53, 48, 104, 56, 119)
colors <- c("red", "green", "green", "green", "green", "blue", "red", "green", "red")

m <- leaflet()
m <- addTiles(m)
m <- addCircleMarkers(m, lng=location_Cheongju$long, lat=location_Cheongju$lat, popup=location_Cheongju$name, c
m
```



12.2 leaflet과 polygon

R 공간통계에서 polygon이라 함은 보통 면적이 있는 닫힌 폐곡선/폐곡선들, 즉 도형들을 의미한다. 사실 R 에서의 이런 polygon들은 점의 집합이라고 할 수 있다. 이것의 대표적인 예는 행정구역 경계다. 대한민국 지도를 그릴 때에도 본토와 제주도를 비롯한 섬들을 각각의 polygon으로 보고 출력해 내는 것이다. 다만 대한민국은 섬이 너무 많아 모든 섬을 다 출력하려면 시간이 너무 오래 걸릴 것이다.

여기서는 청주시 행정구역 경계를 한번 출력해 보기로 한다. 이러한 자료는 S4 타입이기 때문에 $가 아닌 @를 이용해 구성요소를 살펴봐야 한다. 절대 plot(shape)를 쓰기 말기를 바란다. 시간이 엄청 오래 걸릴 것이다.

```
library(rgdal)
```

```
## Loading required package: sp
```

```
## rgdal: version: 1.5-23, (SVN revision 1121)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.2.1, released 2020/12/29
## Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/4.1/Resources/library/rgdal/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 7.2.1, January 1st, 2021, [PJ_VERSION: 721]
## Path to PROJ shared files: /Library/Frameworks/R.framework/Versions/4.1/Resources/library/rgdal/proj
## PROJ CDN enabled: FALSE
## Linking to sp version:1.4-5
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading rgdal.
## Overwritten PROJ_LIB was /Library/Frameworks/R.framework/Versions/4.1/Resources/library/rgdal/proj
```

```
shape <- readOGR("~/Dropbox/Maps/KOR_SGG(WGS)/TL_SCCO_SIG.shp")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/Users/seoncheolpark/Dropbox/Maps/KOR_SGG(WGS)/TL_SCCO_SIG.shp", layer: "TL_SCCO_SIG"
## with 252 features
## It has 3 fields
```

```
#grep: pattern matching
grep("Cheongju-si", shape@data$SIG_ENG_NM)
```

```
## [1] 239 240 241 242
```

```
shape.Cheongju <- shape[grep("Cheongju-si", shape@data$SIG_ENG_NM),]
```

```
m <- leaflet()
m <- addTiles(m)
m <- addPolygons(m,data = shape.Cheongju, fill = TRUE)
m
```