

4장 데이터 조작 I

iris 데이터

```
> head(iris)
  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
4          4.6          3.1          1.5          0.2  setosa
5          5.0          3.6          1.4          0.2  setosa
6          5.4          3.9          1.7          0.4  setosa

> str(iris)
'data.frame': 150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

iris 데이터

- Species:
 - 붓꽃의 종. setosa, versicolor, virginica의 세가지 값중 하나를 저장한 범주형 변수.
- Sepal.Width
 - 꽃받침의 너비. Number 변수.
- Sepal.Length
 - 꽃받침의 길이. Number 변수.
- Petal.Width
 - 꽃잎의 너비. Number 변수.
- Petal.Length
 - 꽃잎의 길이. Number 변수.
- 그 이외의 dataset을 보려면
 - `library(help=datasets)`

파일 입출력

■ CSV파일

— read.csv(파일명, header=TRUE) #header가 있는 경우(default)

— eg) a.csv

	id	name	score
	1	"Mr. Foo"	95
	2	"Ms. Bar"	97
	3	"Mr. Baz"	92

```
> x <- read.csv("a.csv")
> x
  id  name score
1  1 Mr. Foo   95
2  2 Ms. Bar   97
3  3 Mr. Baz   92
> str(x)
'data.frame': 3 obs. of  3 variables:
 $ id    : int  1 2 3
 $ name  : Factor w/ 3 levels "Mr. Baz","Mr. Foo",...: 2 3 1
 $ score : int  95 97 92
```

파일 입출력

- csv 파일에 헤더행이 없다면 다음과 같이 header=FALSE를 지정

```
> x <- read.csv("b.csv")
> x
  X1 Mr..Foo X95
1  2 Ms. Bar  97
2  3 Mr. Baz  92
> names(x) <- c("id", "name", "score")
> x
  id    name score
1  2 Ms. Bar   97
2  3 Mr. Baz   92
> str(x)
'data.frame': 2 obs. of  3 variables:
 $ id    : int  2 3
 $ name  : Factor w/ 2 levels "Mr. Baz","Ms. Bar": 2 1
 $ score : int  97 92
```

파일 입출력

- 기본적으로 문자열은 모두 Factor형태로 변환
- 문자열로 사용하려면 재변화해야함

```
> x$name = as.character(x$name)
> str(x)
'data.frame': 3 obs. of 3 variables:
 $ id      : int  1 2 3
 $ name    : chr  "Mr. Foo" "Ms. Bar" "Mr. Baz"
 $ score   : int  95 97 92
```

- 입력단계에서 문자열로 입력 받으려면

```
> x <- read.csv("a.csv", stringsAsFactors=FALSE)
> str(x)
'data.frame': 3 obs. of 3 variables:
 $ id      : int  1 2 3
 $ name    : chr  "Mr. Foo" "Ms. Bar" "Mr. Baz"
 $ score   : int  95 97 92
```

파일 입출력

- NA 지정 문자열이 있을 경우

```
id,name,score
1,"Mr. Foo",95
2,"Ms. Bar",NIL
3,"Mr. Baz",92
```

```
> x <- read.csv("c.csv")
```

```
> x
```

```
  id  name score
1  1 Mr. Foo   95
2  2 Ms. Bar  NIL
3  3 Mr. Baz   92
```

```
> str(x)
```

```
'data.frame': 3 obs. of 3 variables:
```

```
$ id : int 1 2 3
```

```
$ name : Factor w/ 3 levels "Mr. Baz","Mr. Foo",...: 2 3 1
```

```
$ score: Factor w/ 3 levels " 92"," 95"," NIL": 2 3 1
```

자료형이 제대로 설정안됨

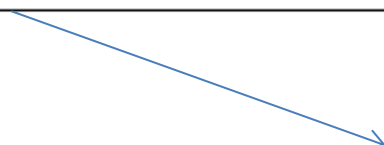


파일 입출력

```
> x <- read.csv("c.csv", na.strings=c("NIL"))
> str(x)
'data.frame': 3 obs. of 3 variables:
 $ id    : int  1 2 3
 $ name  : Factor w/ 3 levels "Mr. Baz","Mr. Foo",...: 2 3 1
 $ score: int  95 NA 92
> is.na(x$score)
[1] FALSE  TRUE FALSE
```


csv 파일로의 저장


```
> write.csv(x, "b.csv", row.names=F)
```



```
"id","name","score"  
1,"Mr. Foo",95  
2,"Ms. Bar",97  
3,"Mr. Baz",92
```

- row.names = T로 지정하면

```
","id","name","score"  
"1",1,"Mr. Foo",95  
"2",2,"Ms. Bar",97  
"3",3,"Mr. Baz",92
```



행번호가 출력됨

객체의 파일 입출력

- Binary 형태로 객체를 입출력

```
> x <- 1:5  
> y <- 6:10  
> save(x, y, file="xy.RData")
```

```
> rm(list=ls())  
> x  
Error: object 'x' not found  
> y  
Error: object 'y' not found  
> load("xy.RData")  
> x  
[1] 1 2 3 4 5  
> y  
[1] 6 7 8 9 10
```

데이터 프레임의 행과 열 병합

- `rbind()`, `cbind()`
 - 행렬이나 데이터 프레임의 데이터 병합


```
> rbind(c(1, 2, 3), c(4, 5, 6))  
      [,1] [,2] [,3]  
[1,]    1    2    3  
[2,]    4    5    6
```

```
> cbind(c(1, 2, 3), c(4, 5, 6))  
      [,1] [,2]  
[1,]    1    4  
[2,]    2    5  
[3,]    3    6
```

데이터 프레임의 행과 열 병합

```
> x <- data.frame(id=c(1, 2), name=c("a", "b"), stringsAsFactors=F)
> x
  id name
1  1   a
2  2   b
> str(x)
'data.frame': 2 obs. of  2 variables:
 $ id   : num  1 2
 $ name: chr  "a" "b"
> y <- rbind(x, c(3, "c"))
> y
  id name
1  1   a
2  2   b
3  3   c
```

지정 안하면 name 컬럼은
Factor형이 되므로 a와 b이외에는
추가 못함



데이터 프레임의 행과 열 병합

```
> y <- cbind(x, greek=c('alpha', 'beta'))
> y
  id name greek
1  1    a alpha
2  2    b  beta
> str(y)
'data.frame': 2 obs. of  3 variables:
 $ id    : num  1 2
 $ name  : chr  "a" "b"
 $ greek: Factor w/ 2 levels "alpha","beta": 1 2
> y <- cbind(x, greek=c('alpha', 'beta'), stringsAsFactors=F)
> str(y)
'data.frame': 2 obs. of  3 variables:
 $ id    : num  1 2
 $ name  : chr  "a" "b"
 $ greek: chr  "alpha" "beta"
```

데이터 프레임의 행과 열 병합

- 데이터 프레임에 새로운 열을 추가할 때는 `cbind()` 를 사용하지 않고 '변수명\$컬럼명 <- 데이터' 형태로도 열을 추가할 수 있음

Apply 계열 함수들

함수	설명	다른 함수와 비교했을 때의 특징
apply()	배열 또는 행렬에 주어진 함수를 적용한 뒤 그 결과를 벡터, 배열 또는 리스트로 반환	배열 또는 행렬에 적용
lapply()	벡터, 리스트 또는 표현식에 함수를 적용하여 그 결과를 리스트로 반환	결과가 리스트
sapply()	lapply와 유사하지만 결과를 벡터, 행렬 또는 배열로 반환	결과가 벡터, 행렬 또는 배열
tapply()	벡터에 있는 데이터를 특정 기준에 따라 그룹으로 묶은 뒤 각 그룹마다 주어진 함수를 적용하고 그 결과를 반환	데이터를 그룹으로 묶은 뒤 함수를 적용
mapply()	sapply의 확장된 버전으로, 여러 개의 벡터 또는 리스트를 인자로 받아 함수에 각 데이터의 첫째 요소들을 적용한 결과, 둘째 요소들을 적용한 결과, 셋째 요소들을 적용한 결과 등을 반환	여러 데이터를 함수의 인자로 적용

apply()

- `apply(행렬, 방향, 함수)` **#벡터에는 적용 안됨**
 - ‘방향’은 1이 주어진다면 행, 2가 주어진다면 열

```
> sum(1:10)
[1] 55
```

```
> d <- matrix(1:9, ncol=3)
> d
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

```
> apply(d, 1, sum)
[1] 12 15 18
```

```
> apply(d, 2, sum)
[1]  6 15 24
```


apply()

```
> head(iris)
  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
...
> apply(iris[, 1:4], 2, sum)
Sepal.Length Sepal.Width Petal.Length Petal.Width
      876.5      458.6      563.7      179.9
```

- 자주 사용하므로 rowSums(), colSums() 함수가 정의되어 있음

```
> colSums(iris[, 1:4])
Sepal.Length Sepal.Width Petal.Length Petal.Width
      876.5      458.6      563.7      179.9
```

- rowMeans() 또는 colMeans()도 사용 가능

apply()

■ 주의

- `apply (iris [, 1] , 2, sum)`나 `colSums (iris [, 1])` 같이 1 column은 사용 못함
 - 최소 2columns 이상만 적용 가능
- 벡터 형태이므로 이 경우는 `sum(iris [, 1])`과 같이 `sum`을 이용해야 함

```
> apply ( iris [, 1] , 2, sum )
Error in apply(iris[, 1], 2, sum) : dim(X) must have a positive length
> sum( iris [, 1] )
[1] 876.5
```

lapply()

- lapply(X, 함수)
 - 'X'는 벡터 또는 리스트, 데이터 프레임 등 가능
 - 결과는 list 형태
 - 예) 입력이 벡터 형태

```
> result <- lapply(1:3, function(x) { x*2 })
```

```
> result
```

```
[[1]]
```

```
[1] 2
```

```
[[2]]
```

```
[1] 4
```

```
[[3]]
```

```
[1] 6
```

```
> result[[1]]
```

```
[1] 2
```

```
> unlist(result)
```

```
[1] 2 4 6
```

리스트를 벡터로 변환

lapply()

- 입력이 리스트 형태

```
> x <- list(a=1:3, b=4:6)
> x
$a
[1] 1 2 3

$b
[1] 4 5 6

> lapply(x, mean)
$a
[1] 2

$b
[1] 5
```

lapply()

- 입력이 프레임 형태

```
> lapply(iris[, 1:4], mean)
$Sepal.Length
[1] 5.843333

$Sepal.Width
[1] 3.057333

$Petal.Length
[1] 3.758

$Petal.Width
[1] 1.199333
```

- colMeans로도 계산 가능

```
> colMeans(iris[, 1:4])
Sepal.Length Sepal.Width Petal.Length Petal.Width
    5.843333    3.057333    3.758000    1.199333
```

lapply()

- lapply()로 데이터 프레임을 처리한 결과는 리스트임
 - 리스트를 데이터 프레임으로 변환 방법
 - unlist()로 벡터로 변환
 - matrix()로 1 X 4 matrix로 변환 (생략하면 1 column으로 변환됨)
 - 벡터를 직접 데이터 프레임으로 바꾸면 1column 프레임으로 변환됨
 - as.data.frame()으로 데이터 프레임으로 변환

```
> d <- as.data.frame(matrix(unlist(lapply(iris[, 1:4], mean)),
+                             ncol=4, byrow=TRUE))
> names(d) <- names(iris[, 1:4])
> d
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1      5.843333      3.057333         3.758      1.199333
```

lapply()

- 데이터 프레임으로 바꾸는 다른 방법
 - do.call()을 이용
 - do.call(함수, 인자) #함수에 인자를 적용하여 결과 반환

```
> data.frame(do.call(cbind, lapply(iris[, 1:4], mean)))  
  Sepal.Length Sepal.Width Petal.Length Petal.Width  
1      5.843333      3.057333        3.758      1.199333
```

- lapply의 결과인 리스트를 cbind의 인자로 넣어 결과를 얻음
 - 결과는 list 각 요소가 column으로 결합된 matrix 형태

apply()

- lapply()와 유사하지만 리스트대신 행렬, 벡터 등으로 결과를 반환
- 입력: 벡터, 리스트, 데이터 프레임 등

```
> lapply(iris[, 1:4], mean)
$Sepal.Length
[1] 5.843333


$Sepal.Width
[1] 3.057333

$Petal.Length
[1] 3.758

$Petal.Width
[1] 1.199333

> sapply(iris[, 1:4], mean)
Sepal.Length Sepal.Width Petal.Length Petal.Width
      5.843333      3.057333      3.758000      1.199333
> class(sapply(iris[, 1:4], mean))
[1] "numeric"
```

결과는 벡터



apply()

■ 데이터 프레임으로 변환

- `t(x)`를 사용해 벡터의 행과 열을 바꿔주지 않으면 기대한 것과 다른 모양의 데이터 프레임을 얻게 됨
- 아래의 예에서 `x`는 vector임.
 - `t(x)`를 실행하면 1 X 4의 matrix가 생성됨

```
> x <- sapply(iris[, 1:4], mean)
> as.data.frame(x)
              x
Sepal.Length 5.843333
Sepal.Width  3.057333
Petal.Length 3.758000
Petal.Width  1.199333
> as.data.frame(t(x))
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1    5.843333    3.057333     3.758    1.199333
>
```

sapply()

- 결과가 matrix인 sapply의 예

```
> y <- sapply(iris[, 1:4], function(x) { x > 3 })
> class(y)
[1] "matrix"
> head(y)
      Sepal.Length Sepal.Width Petal.Length Petal.Width
[1,]          TRUE          TRUE          FALSE          FALSE
[2,]          TRUE          FALSE          FALSE          FALSE
[3,]          TRUE          TRUE          FALSE          FALSE
[4,]          TRUE          TRUE          FALSE          FALSE
[5,]          TRUE          TRUE          FALSE          FALSE
[6,]          TRUE          TRUE          FALSE          FALSE
...
```

tapply

- 그룹별 처리를 위한 apply 함수
- tapply(데이터, 색인, 함수)
 - 예) 1부터 10까지의 숫자가있고 이들이 모두 한 그룹에 속해있을때 각 그룹에 속한 데이터의 합

```
> tapply(1:10, rep(1, 10), sum)
1
55
```

- 예) 짝수, 홀수별 합

```
> tapply(1:10, 1:10 %% 2 == 1, sum)
FALSE TRUE
30    25
```


- iris데이터에서 Species별 Sepal.Length의 평균

```
> tapply(iris$Sepal.Length, iris$Species, mean)
setosa versicolor virginica
5.006    5.936    6.588
```

tapply

```
> m <- matrix(1:8,  
+           ncol=2,  
+           dimnames=list(c("spring", "summer", "fall", "winter"),  
+                         c("male", "female")))  
> m
```

	male	female
spring	1	5
summer	2	6
fall	3	7
winter	4	8



■ 반기별 남성 셀의 합과 여성 셀의 합

```
> tapply(m, list(c(1, 1, 2, 2, 1, 1, 2, 2),  
+               c(1, 1, 1, 1, 2, 2, 2, 2)), sum)
```

	1	2
1	3	11
2	7	15

summary

```
> summary(iris)

  Sepal.Length   Sepal.Width   Petal.Length   Petal.Width
Min.      :4.300   Min.      :2.000   Min.      :1.000   Min.      :0.100
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
Median :5.800   Median :3.000   Median :4.350   Median :1.300
Mean    :5.843   Mean    :3.057   Mean    :3.758   Mean    :1.199
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
Max.    :7.900   Max.    :4.400   Max.    :6.900   Max.    :2.500

      Species
setosa      :50
versicolor:50
virginica   :50
```

order

- 데이터를 정렬하기 위한 순서를 반환

```
> order(iris$Sepal.Width)
[1] 61 63 69 120 42 ...
...
```

- 정렬

```
> iris[order(iris$Sepal.Width),]
   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
61           5.0         2.0         3.5         1.0 versicolor
63           6.0         2.2         4.0         1.0 versicolor
...
```

- 여러 개의 정렬기준
 - iris [order (iris \$ Sepal.Width, iris \$ Sepal.Length) ,]

데이터 분리(split, subset)

- split(데이터, 분리조건)
 - 결과는 분리된 데이터가 저장된 리스트

```
> split(iris, iris$Species)
$setosa
  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
...
$versicolor
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
51           7.0          3.2          4.7          1.4 versicolor
52           6.4          3.2          4.5          1.5 versicolor
...
$virginica
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
101           6.3          3.3          6.0          2.5 virginica
102           5.8          2.7          5.1          1.9 virginica
...
```

데이터 분리(split, subset)

- 위 결과에 lapply()를 적용해서 iris의 종별 Sepal.Length의 평균을 구할 수 있음

```
> lapply(split(iris$Sepal.Length, iris$Species), mean)
$setosa
[1] 5.006

$versicolor
[1] 5.936

$virginica
[1] 6.588
```


데이터 분리(split, subset)

■ subset

- 조건에 맞는 특정 부분만 찾음

```
> subset(iris, Species == "setosa")
  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
...
```

벡터간 연산이므로 &&가 아님

```
> subset(iris, Species == "setosa" & Sepal.Length > 5.0)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1          3.5          1.4          0.2  setosa
6          5.4          3.9          1.7          0.4  setosa
...
```

데이터 분리(split, subset)

- 특정 열을 선택: select 인자로 지정

```
> subset(iris, select=c(Sepal.Length, Species))
```

	Sepal.Length	Species
1	5.1	setosa
2	4.9	setosa
3	4.7	setosa
4	4.6	setosa
5	5.0	setosa
...		

데이터 병합(merge)

- Database의 조인과 같음

```
> x <- data.frame(name=c("a", "b", "c"), math=c(1, 2, 3))
> y <- data.frame(name=c("c", "b", "a"), english=c(4, 5, 6))
> merge(x, y)
```

	name	math	english
1	a	1	6
2	b	2	5
3	c	3	4

```
> x <- data.frame(name=c("a", "b", "c"), math=c(1, 2, 3))
> y <- data.frame(name=c("c", "b", "a"), english=c(4, 5, 6))
> cbind(x, y)
```

	name	math	name	english
1	a	1	c	4
2	b	2	b	5
3	c	3	a	6

cbind는 단순 결합

정렬(sort, order)

■ sort()

- 벡터를 정렬

```
> x <- c(20, 11, 33, 50, 47)
> sort(x)
[1] 11 20 33 47 50
> sort(x, decreasing=TRUE)
[1] 50 47 33 20 11
> x
[1] 20 11 33 50 47
```

■ order()

- 정렬하기 위한 색인(순서)을 반환

```
> x
[1] 20 11 33 50 47
> order(x)
[1] 2 1 3 5 4
```

큰 수부터 정렬하려면 x대신에 -x를 사용하여 음수로 만들

with, within

- with(data, expression)

```
> print(mean(iris$Sepal.Length))  
[1] 5.843333  
> print(mean(iris$Sepal.Width))  
[1] 3.057333
```

```
> with(iris, {  
+   print(mean(Sepal.Length))  
+   print(mean(Sepal.Width))  
+ })  
[1] 5.843333  
[1] 3.057333
```

with, within

- within: 데이터 수정에 사용
- 예는 다음 페이지

```
> x <- data.frame(val=c(1, 2, 3, 4, NA, 5, NA))
```

```
> x
```

```
  val
```

```
1    1
```

```
2    2
```

```
3    3
```

```
4    4
```

```
5   NA
```

```
6    5
```

```
7   NA
```

```
> x <- within(x, {
```

```
+   val <- ifelse(is.na(val), median(val, na.rm=TRUE), val)
```

```
+ })
```

```
> x
```

```
  val
```

```
1    1
```

```
2    2
```

```
3    3
```

```
4    4
```

```
5    3
```

```
6    5
```

```
7    3
```

생략하면 결과가 NA로 나옴



```
> x$val[is.na(x$val)] <- median(x$val, na.rm=TRUE)
```

동일한 기능

attach, detach

- attach는 인자로 주어진 데이터 프레임이나 리스트를 곧바로 접근. 해제는 detach

```
> Sepal.Width
Error: object 'Sepal.Width' not found
> attach(iris)
> head(Sepal.Width)
[1] 3.5 3.0 3.2 3.1 3.6 3.9
> ?detach
> detach(iris)
> Sepal.Width
Error: object 'Sepal.Width' not found
```


attach, detach

- 주의: attach()한 변수값은 detach()시 원래의 데이터 프레임에는 반영되지 않음

```
> data(iris)
> head(iris)
  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
...
> attach(iris)
> Sepal.Width[1] = -1
> Sepal.Width
[1] -1.0  3.0  3.2  3.1  3.6  3.9  3.4  3.4  2.9  3.1  3.7  3.4  3.0
     3.0  4.0  4.4  3.9  3.5  3.8
...
> detach(iris)
> head(iris)
  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
...
```

조건에 맞는 데이터 색인 찾기

- 조건에 맞는 값 자체를 찾을 때는 subset이나 조건문 지정

```
> subset(iris, Species == 'setosa')
  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
4          4.6         3.1          1.5          0.2  setosa
5          5.0         3.6          1.4          0.2  setosa
6          5.4         3.9          1.7          0.4  setosa
7          4.6         3.4          1.4          0.3  setosa
...
```

```
> iris[iris$Species == "setosa",]
  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
4          4.6         3.1          1.5          0.2  setosa
5          5.0         3.6          1.4          0.2  setosa
6          5.4         3.9          1.7          0.4  setosa
7          4.6         3.4          1.4          0.3  setosa
8          5.0         3.4          1.5          0.2  setosa
...
```

조건에 맞는 데이터 색인 찾기

- which: 조건에 맞는 색인(위치) 찾기

```
> x <- c(2, 4, 6, 7, 10)
> x %% 2
[1] 0 0 0 1 0
> which(x %% 2 == 0)
[1] 1 2 3 5
> x[which(x %% 2 == 0)]
[1] 2 4 6 10
```

```
> x <- c(2, 4, 6, 7, 10)
> which.min(x)
[1] 1
> x[which.min(x)]
[1] 2
> which.max(x)
[1] 5
> x[which.max(x)]
[1] 10
```

```
> which(iris$Species == "setosa")
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
[21] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
[41] 41 42 43 44 45 46 47 48 49 50
>
```

- 공통된 이름이 없는 경우에는 한쪽에 데이터가 없게 되는데 이 경우 값을 NA로 채우면서 전체 데이터를 모두 합치려면 다음과 같이 all 인자에 TRUE를 지정

```
> merge(x, y, all=TRUE)
```

	name	math	english
1	a	1	4
2	b	2	5
3	c	3	NA
4	d	NA	6