I. R 데이터 불러오기 & 저장하기

(dat, csv, txt, Rdata)

데이터의 저장 및 불러오기

➢ 저장하기

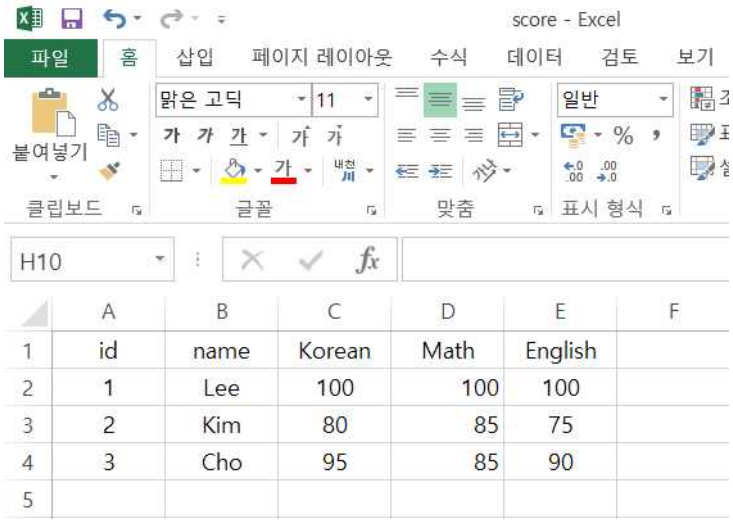
```
> no = c(1,2,3,4)
> name = c("Apple", "Banana","Peach","Berry")
> price = c(500,200,300,400)
> qty=c(5,2,7,9)
> fruit = data.frame(No=no,Name=name,Price=price,Quantity=qty)
>
> fruit
  No    Name Price Quantity
1  1   Apple   500        5
2  2  Banana   200        2
3  3   Peach   300        7
4  4   Berry   400        9
>
> getwd()
[1] "C:/Users/KSL/Documents/R/prog/chap1"
> save(fruit,file="test.dat")
```

➢ 불러오기

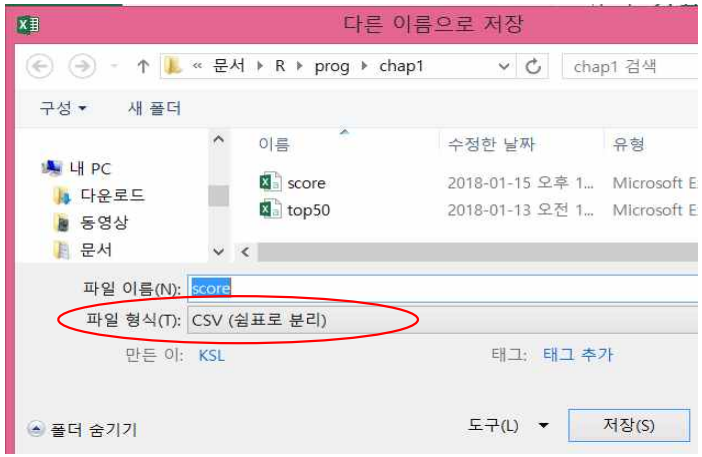
```
>
> fruit
Error: object 'fruit' not found
>
> load("test.dat")
> fruit
  No    Name Price Quantity
1  1   Apple   500        5
2  2  Banana   200        2
3  3   Peach   300        7
4  4   Berry   400        9
```

엑셀데이터의 저장 및 불러오기

➢ 엑셀파일 저장하기



	A	B	C	D	E	F
1	id	name	Korean	Math	English	
2	1	Lee	100	100	100	
3	2	Kim	80	85	75	
4	3	Cho	95	85	90	
5						



➢ 불러오기

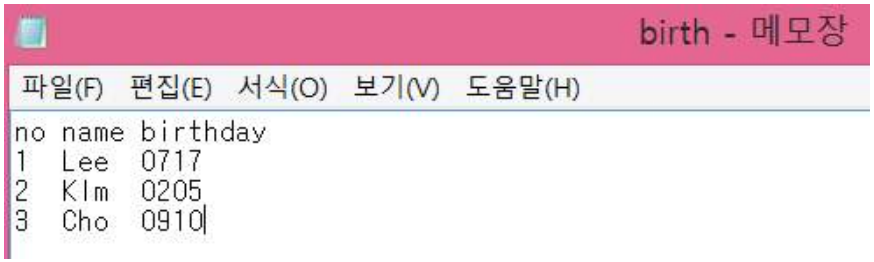
```
>
> getwd()
[1] "C:/Users/KSL/Documents/R/prog/chap1"
> score=read.csv("score.csv")
> score
  id name Korean Math English
1  1  Lee    100  100     100
2  2  Kim     80   85      75
3  3  Cho     95   85      90
>
```

➢ 저장하기

```
>
> no = c(1,2,3,4)
> name=c("apple","pear","banana","peach")
> price=c(100,200,300,400)
>
> fruit=data.frame(No=no,Name=name,PRICE=price)
>
> fruit
  No   Name PRICE
1  1  apple   100
2  2   pear   200
3  3 banana   300
4  4  peach   400
>
> write.csv(fruit,"fruit.csv")
>
```

일반 데이터의 저장 및 불러오기

- 텍스트파일 저장하기



- 불러오기

```
> getwd()
[1] "C:/Users/KSL/Documents/R/prog/chap1"
> b = scan("birth.txt",what="")
Read 12 items
> b
 [1] "no"       "name"     "birthday" "1"        "Lee"      "0717"
 [7] "2"        "KIm"      "0205"     "3"        "Cho"      "0910"
>
> c=read.table("birth.txt",header=T)
> c
  no name birthday
1  1  Lee     2001
2  2  KIm     2003
3  3  Cho     2007
```

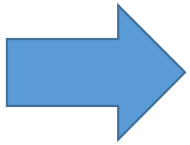
- 저장하기

```
>
> vec1=c(1,2,3)
> vec2=c(4,5,6)
> mat=rbind(vec1,vec2)
> mat
     [,1] [,2] [,3]
vec1    1    2    3
vec2    4    5    6
>
> save(mat,file="testmat.txt")
>
> dfile=load("testmat.txt")
> dfile
[1] "mat"
> mat
     [,1] [,2] [,3]
vec1    1    2    3
vec2    4    5    6
>
```

엑셀 입출력: 유의점

- a.csv

id	name	score
1	Lee	95
2	Kim	97
3	Park	92



```
> (x=read.csv("a.csv"))
  id name score
1  1  Lee    95
2  2  Kim    97
3  3 Park    92
> str(x)
'data.frame':   3 obs. of  3 variables:
 $ id   : int  1 2 3
 $ name : Factor w/ 3 levels "Kim","Lee","Park": 2 1 3
 $ score: int  95 97 92
```

- Data frame으로 반환됨
- Name이 factor로 변환됨. => 문자열로 변환해야 함

```
x$name = as.character(x$name) 혹은
x = read.csv("a.csv", stringsAsFactors=FALSE)
```

엑셀 입출력: 유의점

- b.csv: header 가 없는 경우

1	Lee	95
2	Kim	97
3	Park	92

```
> x=read.csv("b.csv",header=FALSE)
> x
  V1   V2 V3
1  1  Lee 95
2  2  Kim 97
3  3 Park 92
> names(x)=c("id","name","score")
> x
  id name score
1  1  Lee    95
2  2  Kim    97
3  3 Park    92
```

엑셀 입출력: 유의점

- c.csv: NIL이 저장된 경우

id	name	score
1	Lee	95
2	Kim	nil
3	Park	92

```
> x=read.csv("c.csv")
> x
  id name score
1  1  Lee    95
2  2  Kim   nil
3  3 Park    92
> str(x)
'data.frame':   3 obs. of  3 variables:
 $ id   : int  1 2 3
 $ name : Factor w/ 3 levels "Kim","Lee","Park": 2 1 3
 $ score: Factor w/ 3 levels "92","95","nil": 2 3 1
```

```
> x = read.csv("c.csv",na.strings=c("nil"))
> x
  id name score
1  1  Lee    95
2  2  Kim    NA
3  3 Park    92
> str(x)
'data.frame':   3 obs. of  3 variables:
 $ id   : int  1 2 3
 $ name : Factor w/ 3 levels "Kim","Lee","Park": 2 1 3
 $ score: int  95 NA 92
```

nil => na

read.csv("c.csv", na.strings=c("nil"))

객체의 파일 입출력

- save(), load()
  - 데이터를 저장하여 나중에 불러올 수 있음

```
> 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
```

II. R 데이터 조작, 처리, 가공
: 기본 함수

[1] apply 계열 함수

- 벡터, 행렬, 데이터 프레임에 임의의 함수를 적용한 결과를 얻기 위한 함수
- 데이터 전체에 대해 함수를 한번에 적용하는 연산 수행을 통해 데이터 조작, 처리

함수	설명
apply()	배열 또는 행렬에 주어진 함수를 적용한 뒤 그 결과를 벡터, 배열 또는 리스트로 반환
lapply()	벡터, 리스트 또는 표현식에 함수를 적용하여 그 결과를 리스트로 반환
sapply	lapply 와 유사하지만, 결과를 벡터, 행렬, 또는 배열로 반환
tapply	벡터에 있는 데이터를 특정 기준에 따라 그룹으로 묶은 뒤 각 그룹마다 주어진 함수를 적용하고 그 결과를 반환
mapply	sapply의 확장된 버전으로, 여러 개의 벡터 또는 리스트를 인자로 받아 함수에 각 데이터들을 적용한 결과 등을 반환

apply

```
apply(X, MARGIN, FUN)
```

X: 배열

MARGIN: 함수를 적용하는 방향(1: 행, 2: 열, c(1,2): 행,열 모두)

FUN: 적용할 함수

Q: iris 데이터들에 대해, 꽃받침 길이 및 넓이, 꽃잎 길이 및 넓이 각각의 합계를 구하라.

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

lapply, sapply

`lapply(X,FUN, 추가인자)`

X: 배열,리스트,표현식

FUN: 적용할 함수

결과를 리스트로 반환

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

$c
[1] 4 5 6

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

$c
[1] 5
```

`sapply(X,FUN, 추가인자)`

X: 배열,리스트,표현식

FUN: 적용할 함수

결과를 행렬, 벡터 등의 데이터 타입으로 반환

```
> 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"
> 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
```

tapply

```
tapply(X, INDEX, FUN)
```

X: 배열

INDEX: 데이터를 그룹으로 묶을 색인, factor를 지정해야 하며, factor가 아닐 때는 자동 변환

FUN: 적용할 함수

Q: iris 데이터들에 대해, Species별 Sepal.Length의 평균 구하기

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

mapply

`mapply(FUN, 적용할 인자)`

FUN: 적용할 함수

Q: rnorm()을 다음 세가지 조합에 대해 호출할 때
* rnorm(n(난수 개수), mean, sd)

n	mean	sd
1	0	1
2	10	1
3	100	1

```
> mapply(rnorm,
+ c(1,2,3),          #n
+ c(0,10,100),      #mean
+ c(1,1,1))         #sd
[[1]]
[1] -0.1447985

[[2]]
[1] 10.80663 10.88324

[[3]]
[1] 100.7147 100.4280 100.4653
```

[2] 데이터를 그룹으로 묶은 후 함수 호출

- **doBy** package

함수	특징
summaryBy	데이터프레임을 컬럼 값에 따라 그룹으로 묶은 후 요약 값 계산

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa     :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 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
>
```

```
> summaryBy(Sepal.Width+Sepal.Length~Species, iris)
     Species Sepal.Width.mean Sepal.Length.mean
1     setosa             3.428              5.006
2 versicolor             2.770              5.936
3  virginica             2.974              6.588
```

[2] 데이터를 그룹으로 묶은 후 함수 호출

- doBy package

함수	특성
orderBy	지정된 컬럼값에 따라 데이터 프레임을 정렬

```
> order(iris$Sepal.Width)
  [1]  61  63  69 120  42  54  88  94  58  81  82  70  73  90  99 107 109 114 147  80
 [21]  91  93 119 135  60  68  83  84  95 102 112 124 143  55  56  72  74  77 100 115
 [41] 122 123 127 129 131 133 134   9  59  64  65  75  79  97  98 104 108   2  13  14
 [61]  26  39  46  62  67  76  78  85  89  92  96 103 105 106 113 117 128 130 136 139
 [81] 146 148 150   4  10  31  35  53  66  87 138 140 141 142   3  30  36  43  48  51
[101]  52  71 111 116 121 126 144  24  50  57 101 125 145   7   8  12  21  25  27  29
[121]  32  40  86 137 149   1  18  28  37  41  44   5  23  38 110  11  22  49  19  20
[141]  45  47 118 132   6  17  15  33  34  16
> orderBy(~Sepal.Width, iris)
    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
69           6.2         2.2          4.5         1.5 versicolor
120          6.0         2.2          5.0         1.5  virginica
42           4.5         2.3          1.3         0.3     setosa
54           5.5         2.3          4.0         1.3 versicolor
88           6.3         2.3          4.4         1.3 versicolor
94           5.0         2.3          3.3         1.0 versicolor
:
```

[2] 데이터를 그룹으로 묶은 후 함수 호출

➢ doBy package

함수	특징
sampleBy	데이터프레임을 컬럼 값에 따라 그룹으로 묶은 후 sample 추출

```
> sampleBy(~Species, frac=0.1, data=iris)
             Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
setosa.7              4.6         3.4          1.4         0.3     setosa
setosa.19             5.7         3.8          1.7         0.3     setosa
setosa.24             5.1         3.3          1.7         0.5     setosa
setosa.25             4.8         3.4          1.9         0.2     setosa
setosa.39             4.4         3.0          1.3         0.2     setosa
versicolor.53         6.9         3.1          4.9         1.5 versicolor
versicolor.84         6.0         2.7          5.1         1.6 versicolor
versicolor.88         6.3         2.3          4.4         1.3 versicolor
versicolor.94         5.0         2.3          3.3         1.0 versicolor
versicolor.97         5.7         2.9          4.2         1.3 versicolor
virginica.105         6.5         3.0          5.8         2.2  virginica
virginica.108         7.3         2.9          6.3         1.8  virginica
virginica.120         6.0         2.2          5.0         1.5  virginica
virginica.128         6.1         3.0          4.9         1.8  virginica
virginica.143         5.8         2.7          5.1         1.9  virginica
```

: iris데이터에서 각 Species별로 10%의 데이터(5개씩)를 추출

[3] 데이터 분리 및 병합

함수	특징
split()	주어진 조건에 따라 데이터를 분리한다.

```
> 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
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

:
$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
53           6.9         3.1          4.9         1.5 versicolor
54           5.5         2.3          4.0         1.3 versicolor
55           6.5         2.8          4.6         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
103          7.1         3.0          5.9         2.1 virginica
104          6.3         2.9          5.6         1.8 virginica
```

Iris데이터를 iris$Species에 따라 분리하고 결과를 리스트에 저장

[3] 데이터 분리 및 병합

함수	특징
subset()	주어진 조건을 만족하는 데이터를 선택한다.

```
> 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
:
```

Iris데이터 중 조건을 만족하는 특정 부분만 취하여 반환: setosa종만 추출

```
> subset(iris, Sepal.Length > 5.0 & Sepal.Width > 4.0)
```

[3] 데이터 분리 및 병합

함수	특징
merge()	데이터를 공통된 값에 기준해 병합한다.

```
> 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
```

[4] 데이터 프레임 컬럼 접근

함수	특성
with()	코드 불록 안에서 필드 이름만으로 데이터를 곧바로 접근할 수 있도록 함
within()	with()와 동일한 기능을 제공하지만 데이터에 저장된 값을 손쉽게 변경하는 기능 제공

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

Iris$Sepal.Length와 같이 쓰지 않아도 각 컬럼을 곧바로 접근 가능

```
> 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
```

X$val로 표시하지 않더라도, NA값을 평균으로 교체

[4] 데이터 프레임 컬럼 접근

함수	특성
attach()	Attach()이후 코드에서는 필드 이름만으로 데이터를 곧바로 접근할 수 있도록 함
detach()	Attach()의 반대 역할도 detach() 이후 코드에서 더 이상 필드 이름으로 데이터를 곧바로 접근할 수 없도록 함

```
> 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(iris)
> Sepal.Width
Error: object 'Sepal.Width' not found
```

II. R 데이터 조작, 처리, 가공

: dplyr 패키지

[5] dplyr 패키지

➢데이터 전처리: 분석에 적합하게 데이터를 가공하는 작업

- 일부 추출, 종류별로 나누기, 여러 데이터 합치기 등의 작업 수행

➢dplyr: 데이터 전처리 작업에 많이 사용되는 패키지

dplyr 함수	기능
filter()	행추출
select()	열(변수) 추출
arrange()	정렬
mutate()	변수 추가
summarize()	통계치 산출
group_by	집단별로 나누기
left_join()	데이터 합치기(열)
bind_rows()	데이터 합치기(행)

❖ Pipe operator 제공: %>%

```
B = f1(A, ...)
C = f2(B, ...)
D = f3(C, ...)
:
:
```

```
A%>% f1() %>% f2() %>% f3()
```

데이터 가공: [1] 추출

1. 조건에 맞는 데이터만 추출: object_name %>% **filter(조건식)**

Class	English	Science
2	98	50
1	97	60
2	86	78
1	98	58
1	80	65
2	89	98

```
exam%>%filter(class==1)
```

Class	English	Science
1	97	60
1	98	58
1	80	65

데이터 전처리: 추출 실습

➢object_name%>%filter

	id	class	math	english	science
1	1	1	50	98	50
2	2	1	60	97	60
3	3	1	45	86	78
4	4	1	30	98	58
5	5	2	25	80	65
6	6	2	50	89	98
7	7	2	80	90	45
8	8	2	90	78	25
9	9	3	20	98	15
10	10	3	50	98	45
11	11	3	65	65	65
12	12	3	45	85	32
13	13	4	46	98	65
14	14	4	48	87	12
15	15	4	75	56	78
16	16	4	58	98	65
17	17	5	65	68	98
18	18	5	80	78	90
19	19	5	89	68	87
20	20	5	78	83	58

```
exam%>%filter(class==1)
```

```
exam%>%filter(class!=1)
```

```
exam%>%filter(math > 50)
```

```
exam%>%filter(class==1 & math > 50)
```

```
exam%>%filter(math>90 | English>90)
```

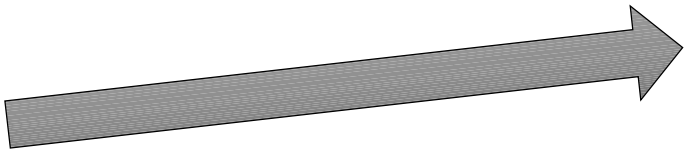
```
exam%>%filter(class %in% c(1,3,5))
```

연산자	기능
<	작다
<=	작거나같다
>	크다
>=	크거나같다
==	같다
!=	같지 않다
|	또는
&	그리고
%in%	매칭 확인
+	더하기
-	빼기
*	곱하기
/	나누기
^,**	제곱
%/%	나눗셈의 몫
%%	나눗셈의 나머지

데이터 가공: [2] 변수 추출

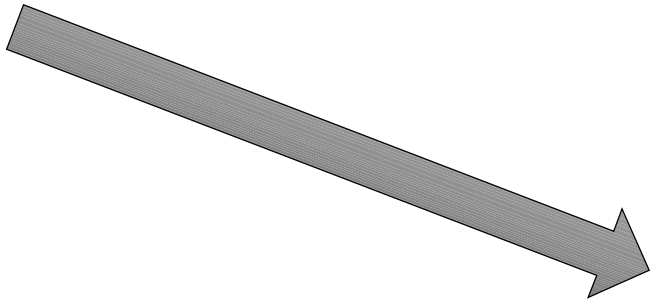
2. 조건에 맞는 변수만 추출: object_name%>%**select**

Class	English	Science
2	98	50
1	97	60
2	86	78
1	98	58
1	80	65
2	89	98



```
exam%>%select(class,English)
```

Class	English
2	98
1	97
2	86
1	98
1	80
2	89



```
exam%>%select(-English)
```

Class	Science
2	50
1	60
2	78
1	58
1	65
2	98

데이터 가공: 실습

1반 학생의 영어 점수 추출: filter()와 select() 연결

Class	English	Science
2	98	50
1	97	60
2	86	78
1	98	58
1	80	65
2	89	98

```
exam%>%filter(class==1) %>% select(english)
```

데이터 가공: [3] 정렬

3. 정렬하기: arrange()

	id	class	math	english	science
1	1	1	50	98	50
2	2	1	60	97	60
3	3	1	45	86	78
4	4	1	30	98	58
5	5	2	25	80	65
6	6	2	50	89	98
7	7	2	80	90	45
8	8	2	90	78	25
9	9	3	20	98	15
10	10	3	50	98	45
11	11	3	65	65	65
12	12	3	45	85	32
13	13	4	46	98	65
14	14	4	48	87	12
15	15	4	75	56	78
16	16	4	58	98	65
17	17	5	65	68	98
18	18	5	80	78	90
19	19	5	89	68	87
20	20	5	78	83	58

```
exam %>% arrange(math)
```

	id	class	math	english	science
1	9	3	20	98	15
2	5	2	25	80	65
3	4	1	30	98	58
4	3	1	45	86	78
5	12	3	45	85	32
6	13	4	46	98	65
7	14	4	48	87	12
8	1	1	50	98	50
9	6	2	50	89	98
10	10	3	50	98	45
11	16	4	58	98	65
12	2	1	60	97	60
13	11	3	65	65	65
14	17	5	65	68	98
15	15	4	75	56	78
16	20	5	78	83	58
17	7	2	80	90	45
18	18	5	80	78	90
19	19	5	89	68	87
20	8	2	90	78	25

데이터 가공: 정렬

3. 정렬하기: arrange()

	id	class	math	english	science
1	1	1	50	98	50
2	2	1	60	97	60
3	3	1	45	86	78
4	4	1	30	98	58
5	5	2	25	80	65
6	6	2	50	89	98
7	7	2	80	90	45
8	8	2	90	78	25
9	9	3	20	98	15
10	10	3	50	98	45
11	11	3	65	65	65
12	12	3	45	85	32
13	13	4	46	98	65
14	14	4	48	87	12
15	15	4	75	56	78
16	16	4	58	98	65
17	17	5	65	68	98
18	18	5	80	78	90
19	19	5	89	68	87
20	20	5	78	83	58

```
exam %>% arrange(class,
desc(math))
```

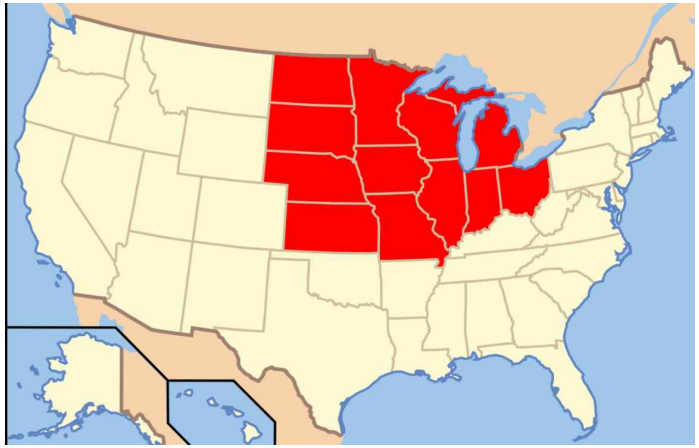
```
> exam %>% arrange(class,desc(math))
   id class math english science
1   2     1   60      97      60
2   1     1   50      98      50
3   3     1   45      86      78
4   4     1   30      98      58
5   8     2   90      78      25
6   7     2   80      90      45
7   6     2   50      89      98
8   5     2   25      80      65
9  11     3   65      65      65
10 10     3   50      98      45
11 12     3   45      85      32
12  9     3   20      98      15
13 15     4   75      56      78
14 16     4   58      98      65
15 14     4   48      87      12
16 13     4   46      98      65
17 19     5   89      68      87
18 18     5   80      78      90
19 20     5   78      83      58
20 17     5   65      68      98
```

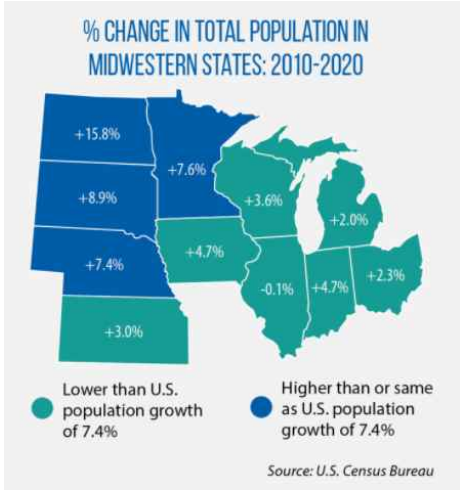
DPLYR 실습:
Midwest demographics

① 각 county별로 전체 인구 대비 아시아 인구 백분율을 구하고 그 값이 Midwest지역의 평균을 초과할 경우는 "large", 그 외에는 "small"로 분류하여 "large"와 "small"에 해당하는 지역이 각각 얼마나 되는지 확인 하여라

② 아시아 인구 백분율이 가장 높은 상위 10개 county(지역)의 아시아 인구 백분율을 출력하시오.

③ 아시아 인구 백분율이 가장 높은 순으로 정렬하여 county와 아시아 인구 백분율을 asiapop.csv파일로 생성하시오.





실습: R데이터셋 분석(Midwest demographics)

midwest {ggplot2}

Midwest demographics

Description

Demographic information of midwest counties

Usage

```
midwest
```

Format

A data frame with 437 rows and 28 variables

	PID	county	state	area	poptotal	popdensity	popwhite	popblack	popamerindian	popasian	popother
	<int>	<chr>	<chr>	<dbl>	<int>	<dbl>	<int>	<int>	<int>	<int>	<int>
1	561	ADAMS	IL	0.0520	66090	1271	63917	1702	98	249	124
2	562	ALEXANDER	IL	0.0140	10626	759	7054	3496	19	48	9
3	563	BOND	IL	0.0220	14991	681	14477	429	35	16	34
4	564	BOONE	IL	0.0170	30806	1812	29344	127	46	150	1139
5	565	BROWN	IL	0.0180	5836	324	5264	547	14	5	6
6	566	BUREAU	IL	0.0500	35688	714	35157	50	65	195	221

```
# ... with 17 more variables: percwhite <dbl>, percblack <dbl>, percamerindan <dbl>, percasian <dbl>,
#   percother <dbl>, popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
#   poppovertyknown <int>, percpovertyknown <dbl>, percbelowpoverty <dbl>,
#   percchildbelowpovert <dbl>, percadultpoverty <dbl>, percelderlypoverty <dbl>, inmetro <int>,
#   category <chr>
```

실습: Midwest demographics 분석

midwest {ggplot2}

Midwest demographics

Description

Demographic information of midwest counties

Usage

```
midwest
```

Format

A data frame with 437 rows and 28 variables

```
library(ggplot2)
library(dplyr)

dim(midwest)
str(midwest)
head(midwest)

#Q1
midwest$ratio=midwest$popasian/midwest$poptotal
x=mean(midwest$ratio)
midwest$grade=ifelse(midwest$ratio>=x,"large","small")
table(midwest$grade)
qplot(midwest$grade)

#Q2
midwest_new = midwest %>%
  arrange(desc(midwest$ratio)) %>%
  select(county, ratio) %>% head(10)

#Q3
write.csv(midwest_new, "asian_midwest.csv")
```

데이터 가공: [4] 추가

4. 파생변수 추가: mutate

실습: exam데이터

① 총합 변수 추가

② 평균 추가

③ Pass/Fail, 추가(과학점수 60점 기준)

```
> exam %>% mutate(
+ total=math+english+science,
+ mean=(math+english+science)/3,
+ test = ifelse(science >= 60, "pass","fail")
+ )
```

	id	class	math	english	science
1	1	1	50	98	50
2	2	1	60	97	60
3	3	1	45	86	78
4	4	1	30	98	58
5	5	2	25	80	65
6	6	2	50	89	98
7	7	2	80	90	45
8	8	2	90	78	25
9	9	3	20	98	15
10	10	3	50	98	45
11	11	3	65	65	65
12	12	3	45	85	32
13	13	4	46	98	65
14	14	4	48	87	12
15	15	4	75	56	78
16	16	4	58	98	65
17	17	5	65	68	98
18	18	5	80	78	90
19	19	5	89	68	87
20	20	5	78	83	58

	id	class	math	english	science	total	mean	test
1	1	1	50	98	50	198	66.00000	fail
2	2	1	60	97	60	217	72.33333	pass
3	3	1	45	86	78	209	69.66667	pass
4	4	1	30	98	58	186	62.00000	fail
5	5	2	25	80	65	170	56.66667	pass
6	6	2	50	89	98	237	79.00000	pass
7	7	2	80	90	45	215	71.66667	fail
8	8	2	90	78	25	193	64.33333	fail
9	9	3	20	98	15	133	44.33333	fail
10	10	3	50	98	45	193	64.33333	fail
11	11	3	65	65	65	195	65.00000	pass
12	12	3	45	85	32	162	54.00000	fail
13	13	4	46	98	65	209	69.66667	pass
14	14	4	48	87	12	147	49.00000	fail
15	15	4	75	56	78	209	69.66667	pass
16	16	4	58	98	65	221	73.66667	pass
17	17	5	65	68	98	231	77.00000	pass
18	18	5	80	78	90	248	82.66667	pass
19	19	5	89	68	87	244	81.33333	pass
20	20	5	78	83	58	219	73.00000	fail

데이터 가공 : 실습

➢exam.csv에서

① 총합(total) 변수 추가

② 평균(mean) 추가

③ Pass/Fail, 추가(과학점수 60점 기준)

④ total에 따라 정렬

```
> exam %>% mutate(
+ total=math+english+science,
+ mean=(math+english+science)/3,
+ test = ifelse(science >= 60, "pass","fail")
+ )
```

%>%arrange(total)

	id	class	math	english	science	total	mean	test
1	1	1	50	98	50	198	66.00000	fail
2	2	1	60	97	60	217	72.33333	pass
3	3	1	45	86	78	209	69.66667	pass
4	4	1	30	98	58	186	62.00000	fail
5	5	2	25	80	65	170	56.66667	pass
6	6	2	50	89	98	237	79.00000	pass
7	7	2	80	90	45	215	71.66667	fail
8	8	2	90	78	25	193	64.33333	fail
9	9	3	20	98	15	133	44.33333	fail
10	10	3	50	98	45	193	64.33333	fail
11	11	3	65	65	65	195	65.00000	pass
12	12	3	45	85	32	162	54.00000	fail
13	13	4	46	98	65	209	69.66667	pass
14	14	4	48	87	12	147	49.00000	fail
15	15	4	75	56	78	209	69.66667	pass
16	16	4	58	98	65	221	73.66667	pass
17	17	5	65	68	98	231	77.00000	pass
18	18	5	80	78	90	248	82.66667	pass
19	19	5	89	68	87	244	81.33333	pass
20	20	5	78	83	58	219	73.00000	fail

데이터 가공: [5] 집단별 요약

5. 집단별로 요약하기: group_by(), summarise()

class	english	science
2	98	50
1	97	60
2	86	78
1	98	58
1	80	65
2	89	98

class	english	science
1	97	60
1	98	58
1	80	65

class	english	science
2	98	50
2	86	78
2	89	98

Mean(science)	
class1	61.0

Mean(science)	
class2	75.3

Mean(science)	
class1	61.0
class2	75.3

```
%>% group_by(class) %>%summarise(mean = mean(science))
```

데이터 가공 실습

➢반별로 수학 성적 요약하기(평균, 합계, 중앙값)

```
> exam%>%
+ group_by(class) %>%
+ summarise(mean_math=mean(math),
+ sum_math=sum(math),
+ median_math=median(math),
+ n=n())
# A tibble: 5 x 5
  class mean_math sum_math median_math     n
  <int>     <dbl>    <int>       <dbl> <int>
1     1      46.2      185        47.5     4
2     2      61.2      245        65.0     4
3     3      45.0      180        47.5     4
4     4      56.8      227        53.0     4
5     5      78.0      312        79.0     4
```

함수	의미
mean()	평균
sd	표준편차
sum()	합계
median()	중앙값
min()	최솟값
max()	최댓값
n()	빈도

데이터 가공 실습

➢자동차 집단별로 요약하기

MPG 데이터상에서 각 회사별(manufacturer)로 구동방식(drv: 4, f, r) 에 따라 도시 연비가 어떻게 다른지 분석하라

```
> mpg%>%group_by(manufacturer,drv) %>% summarise(mean_cty=mean(cty)) %>% head(10)
```

	manufacturer	drv	mean_cty
	<chr>	<chr>	<dbl>
1	audi	4	16.8
2	audi	f	18.9
3	chevrolet	4	12.5
4	chevrolet	f	18.8
5	chevrolet	r	14.1
6	dodge	4	12.0
7	dodge	f	15.8
8	ford	4	13.3
9	ford	r	14.8
10	honda	f	24.4

데이터 가공: [6] 합치기

6. 데이터 합치기

- 가로로 합치기

id	midterm
1	60
2	80
3	70

+

id	final
1	70
2	83
3	65

=

id	midterm	final
1	60	70
2	80	83
3	70	65

- 세로로 합치기

id	test
1	60
2	80
3	70

+

id	test
4	70
5	83
6	65

=

id	test
1	70
2	83
3	65
4	70
5	83
6	65

데이터 가공: 합치기

➢가로로 합치기: left_join(data1,data2,by="기준변수명")

```
> test1= data.frame(id=c(1,2,3,4,5),midterm=c(60,80,70,90,85))
> test2=data.frame(id=c(1,2,3,4,5), final=c(70,83,65,95,80))

> total=left_join(test1,test2,by="id")
> total
  id midterm final
1  1      60    70
2  2      80    83
3  3      70    65
4  4      90    95
5  5      85    80
```

데이터 가공: 실습

- 가로로 합치기 : exam + 담임선생님
  - Class 별 선생님 이름 추가

Class	teacher
1	Kim
2	Lee
3	Park
4	Choi
5	Jung

+

class	english	science
2	98	50
1	97	60
2	86	78
1	98	58
1	80	65
2	89	98
:	:	:
5	78	83

=

class	english	science	teacher
2	98	50	Lee
1	97	60	Kim
2	86	78	Lee
1	98	58	Kim
1	80	65	Kim
2	89	98	Lee
:	:	:	
5	78	83	Jung

데이터 가공: 실습

- 가로로 합치기 : exam + 담임선생님
  - Class별 담임선생님 table 생성

```
> name=data.frame(class=c(1,2,3,4,5), teacher=c("kim","lee","park","choi","jung"))
> name
  class teacher
1     1     kim
2     2     lee
3     3    park
4     4    choi
5     5    jung
```

  - class를 기준으로 exam 데이터와 합치기

```
> exam_new=left_join(exam, name, by="class")
> exam_new
   id class math english science teacher
1   1     1   50      98      50     kim
2   2     1   60      97      60     kim
3   3     1   45      86      78     kim
4   4     1   30      98      58     kim
5   5     2   25      80      65     lee
6   6     2   50      89      98     lee
7   7     2   80      90      45     lee
8   8     2   90      78      25     lee
9   9     3   20      98      15    park
10 10     3   50      98      45    park
11 11     3   65      65      65    park
12 12     3   45      85      32    park
13 13     4   46      98      65    choi
14 14     4   48      87      12    choi
15 15     4   75      56      78    choi
16 16     4   58      98      65    choi
17 17     5   65      68      98    jung
18 18     5   80      78      90    jung
19 19     5   89      68      87    jung
20 20     5   78      83      58    jung
```

데이터 가공

➢세로로 합치기: bind_rows(data1, data2)

```
> group_a=data.frame(id=c(1,2,3,4,5), test=c(60,80,70,90,85))
> group_b=data.frame(id=c(6,7,8,9,10), test=c(70,83,65,95,80))
> group_all = bind_rows(group_a, group_b)
> group_all
   id test
1   1   60
2   2   80
3   3   70
4   4   90
5   5   85
6   6   70
7   7   83
8   8   65
9   9   95
10 10   80
```

※ 변수명이 다를때는 rename을 이용하여 변수명을 동일하게 맞춘 후 합칠 것

실습: MPG 데이터 분석

mpg {ggplot2} R Documentation

Fuel economy data from 1999 and 2008 for 38 popular models of car

Description

This dataset contains a subset of the fuel economy data that the EPA makes available on http://fueleconomy.gov. It contains only models which had a new release every year between 1999 and 2008 - this was used as a proxy for the popularity of the car.

Usage

```
mpg
```

Format

A data frame with 234 rows and 11 variables

```
> head(mpg)
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 audi         a4      1.80  1999     4 auto(l5)   f        18    29 p     compact
2 audi         a4      1.80  1999     4 manual(m5) f        21    29 p     compact
3 audi         a4      2.00  2008     4 manual(m6) f        20    31 p     compact
4 audi         a4      2.00  2008     4 auto(av)   f        21    30 p     compact
5 audi         a4      2.80  1999     6 auto(l5)   f        16    26 p     compact
6 audi         a4      2.80  1999     6 manual(m5) f        18    26 p     compact
```

Q1: class가 "suv"인 자동차와 "compact"인 자동차 중 어떤 자동차의 도시연비(cty)가 더 높은지 알아보시오.

Q2: "audi"에서 생산한 자동차 중에서 고속도로연비(hwy)가 1~5위에 해당하는 자동차의 데이터를 출력하시오.

Q3: 어떤 회사에서 "compact"(경차) 차종을 가장 많이 생산하는지 분석하시오

실습: MPG 데이터 분석

Q1: class가 "suv"인 자동차와 "compact"인 자동차 중 어떤 자동차의 도시연비(cty)가 더 높은지 알아보시오.

```
> mpg=as.data.frame(ggplot2::mpg)
> df=mpg%>%select(class,cty)
> df_suv=df%>%filter(class=="suv")
> df_compact=df%>%filter(class=="compact")
> mean(df_suv$cty)
[1] 13.5
> mean(df_compact$cty)
[1] 20.12766
```

Q2: "audi"에서 생산한 자동차 중에서 고속도로연비(hwy)가 1~5위에 해당하는 자동차의 데이터를 출력하시오.

```
mpg%>%filter(manufacturer=="audi")
  %>% arrange(desc(hwy))
  %>% head(5)
```

실습: MPG 데이터 분석

Q3: 어떤 회사에서 "compact"(경차) 차종을 가장 많이 생산하는지 분석하시오

```
> head(mpg)
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 audi         a4      1.80  1999     4 auto(l5)   f        18    29 p     compact
2 audi         a4      1.80  1999     4 manual(m5) f        21    29 p     compact
3 audi         a4      2.00  2008     4 manual(m6) f        20    31 p     compact
4 audi         a4      2.00  2008     4 auto(av)   f        21    30 p     compact
5 audi         a4      2.80  1999     6 auto(l5)   f        16    26 p     compact
6 audi         a4      2.80  1999     6 manual(m5) f        18    26 p     compact
```

1. Class로 분류 → 2. manufacturer로 분류 → 3. 빈도를 summarise → 4. 빈도에 따라 정렬

```
> mpg%>%filter(class=="compact")%>%
+ group_by(manufacturer)%>%
+ summarise(num=n())%>%
+ arrange(desc(num))
```

III R 데이터 정제

: 결측치 및 이상치 제거

데이터 정제: (1) 결측치 정제

- 결측치: 누락된 값, 비어있는 값
- R에서는 NA로 표시됨
- R의 결측치 확인 함수: is.na()

```
> df=data.frame(gender=c("M","F", NA, "M","F"), score=c(5,4,3,4,NA))
> df
  gender score
1      M     5
2      F     4
3   <NA>     3
4      M     4
5      F    NA
> is.na(df)
     gender score
[1,]  FALSE FALSE
[2,]  FALSE FALSE
[3,]   TRUE FALSE
[4,]  FALSE FALSE
[5,]  FALSE  TRUE
> table(is.na(df))

FALSE  TRUE
    8     2
```

데이터 정제: (1) 결측치 정제 –제거

① 결측치 찾기

② 결측치 제거
- 결측치가 있는 행만 추출하여 제거: filter
- 여러변수에 동시에 결측치가 없는 데이터만 추출(=결측치가 하나라도 있으면 제거): na.omit()

```
> table(is.na(df$gender))

FALSE  TRUE
    4     1
> table(is.na(df$score))

FALSE  TRUE
    4     1
> mean(df$score)
[1] NA
```

```
> df
  gender score
1      M     5
2      F     4
3   <NA>     3
4      M     4
5      F    NA
```

```
> df_nomiss=df %>% filter(!is.na(score))
> df_nomiss
  gender score
1      M     5
2      F     4
3   <NA>     3
4      M     4
> mean(df_nomiss$score)
[1] 4
>
> df_nomiss= df %>% filter(!is.na(score)&!is.na(gender))
> df_nomiss
  gender score
1      M     5
2      F     4
3      M     4
> mean(df_nomiss$score)
[1] 4.333333
> df_nomiss = na.omit(df)
> mean(df_nomiss$score)
[1] 4.333333
```

데이터 정제: (1) 결측치 정제 - 제외

➢함수의 결측치 제외 기능 이용하기: na.rm=T

```
> mean(df$score, na.rm=T) #결측치 제외하고 평균 산출
[1] 4
>
> sum(df$score, na.rm=T)  #결측치 제외하고 합계 산출
[1] 16
>
> df %>% summarise(mean_score=mean(score, na.rm=T))
  mean_score
1          4
```

데이터 정제: (1) 결측치 정제 - 대체

- 데이터가 작고 결측치가 많은 경우 사용
- 결측치를 제거하는 대신 다른 값을 채워 넣는 방법
  - 평균, 최빈값으로 일괄 대체
  - 통계 분석 기법으로 각 결측치의 예측값을 추정해 대체

```
> df$score
[1]  5  4  3  4 NA
> # 결측치를 평균값으로 대체
> df$score = ifelse(is.na(df$score), mean(df$score,na.rm=T), df$score)
> df$score
[1] 5 4 3 4 4
```

데이터 정제: (2) 이상치 제거

- 이상치(Outlier): 정상 범주에서 크게 벗어난 값
- 제거 순서
  - ① 결측치(na)로 변환
  - ② 분석에서 제외

```
> outlier=data.frame(gender=c(1,2,1,3,2,1),score=c(5,4,3,4,2,600))
> outlier
  gender score
1      1     5
2      2     4
3      1     3
4      3     4
5      2     2
6      1   600
> table(outlier$gender)

1 2 3
3 2 1
> table(outlier$score)

  2   3   4   5 600
  1   1   2   1   1
```

```
> outlier$gender=ifelse(outlier$gender==3, NA, outlier$gender)
> outlier$score=ifelse(outlier$score > 5, NA, outlier$score)
>
> mean(outlier$gender)
[1] NA
> mean(outlier$score)
[1] NA
>
> outlier %>% filter(!is.na(gender)&!is.na(score)) %>% group_by(gender) %>%
+   summarise(mean_score = mean(score))
# A tibble: 2 x 2
  gender mean_score
   <dbl>      <dbl>
1      1          4
2      2          3
```

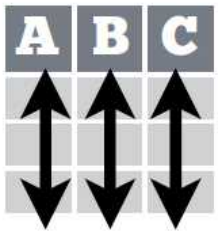
III. R 데이터 정제

: tidyr 패키지

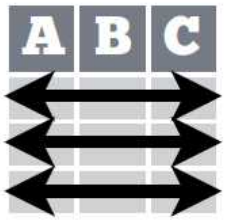
Tidy Data

- Tidy data: a form of tabular data.

A table is tidy if:



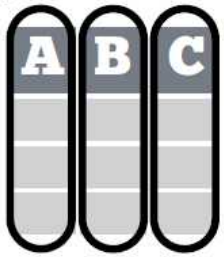
&



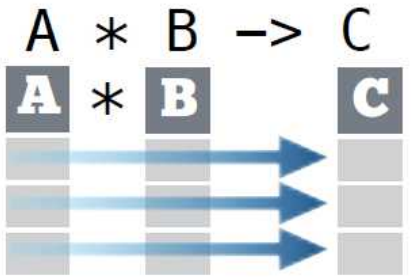
① Each **variable** is in its own **column**

② Each **observation**, or **case**, is in its own **row**

Tidy data:



Makes variables easy to access as vectors



Preserves cases during vectorized operations

③ Each cell contains a single value

Tidy Data

➢ tidyr 패키지: tidy data로 데이터를 정제하기 위한 다양한 함수 제공

기능	함수
Reshape data	gather(), spread()
Handle missing values	drop_na(), fill()
Expand tables	complete(), expand()
Split cells	Separate(), separate_rows(), unite()

➢ 실습 데이터셋

- TB(Tuberculosis) cases in Afghanistan, Brazil, and China between 1999 and 2000
- A subset of the data contained in the World Health Organization Global Tuberculosis Report
  https://www.who.int/tb/country/data/download/en/
- Datasets: table1, table2, table3, table4a, table4b, table5
  - Four variables: country, year, cases and population
  - Each table organizes the values in a different layout

```
> table1
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```

```
> table3
# A tibble: 6 x 3
  country      year  rate
* <chr>       <int> <chr>
1 Afghanistan  1999 745/19987071
2 Afghanistan  2000 2666/20595360
3 Brazil       1999 37737/172006362
4 Brazil       2000 80488/174504898
5 China        1999 212258/1272915272
6 China        2000 213766/1280428583
```

```
> table2
# A tibble: 12 x 4
   country      year type            count
   <chr>       <int> <chr>           <int>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       1999 cases           37737
 6 Brazil       1999 population  172006362
 7 Brazil       2000 cases           80488
 8 Brazil       2000 population  174504898
 9 China        1999 cases          212258
10 China        1999 population 1272915272
11 China        2000 cases          213766
12 China        2000 population 1280428583
```

```
> table4a
# A tibble: 3 x 3
  country     `1999` `2000`
* <chr>        <int>  <int>
1 Afghanistan    745   2666
2 Brazil       37737  80488
3 China       212258 213766
```

```
> table4b
# A tibble: 3 x 3
  country         `1999`     `2000`
* <chr>            <int>      <int>
1 Afghanistan   19987071   20595360
2 Brazil       172006362  174504898
3 China       1272915272 1280428583
```

```
> table5
# A tibble: 6 x 4
  country     century year  rate
* <chr>       <chr>   <chr> <chr>
1 Afghanistan 19      99    745/19987071
2 Afghanistan 20      00    2666/20595360
3 Brazil      19      99    37737/172006362
4 Brazil      20      00    80488/174504898
5 China       19      99    212258/1272915272
6 China       20      00    213766/1280428583
```

Reshaping data in tidyr

➢ change the layout of values in a table

gather(data, key, value, ..., na.rm = FALSE, convert = FALSE, factor_key = FALSE**)**

gather() moves column names into a **key** column, gathering the column values into a single **value** column.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

key value

gather(table4a, `1999`, `2000`, key = "year", value = "cases")

spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL**)**

spread() moves the unique values of a **key** column into the column names, spreading the values of a **value** column across the new columns.

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

key value

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

spread(table2, type, count)

Handling missing values in tidyr

drop_na(data, ...**)**

Drop rows containing NA's in … columns.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
D	3

drop_na(x, x2)

fill(data, ..., .direction = c("down", "up")**)**

Fill in NA's in … columns with most recent non-NA values.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	1
C	1
D	3
E	3

fill(x, x2)

replace_na(data, replace = list(), ...**)**

Replace NA's by column.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	2
C	2
D	3
E	2

replace_na(x, list(x2 = 2))

Expanding tables in tidyr

complete(data, ..., fill = list())

Adds to the data missing combinations of the values of the variables listed in …

(ex) *complete(mtcars, cyl, gear, carb)*

expand(data, ...)

Create new tibble with all possible combinations of the values of the variables listed in …

(ex) *expand(mtcars, cyl, gear, carb)*

Example:

```
df=tibble(
          year=c(2010,2010,2010,2010,2012,2012,2012),
          qtr=c(1,2,3,4,1,2,3),
          revenue=c(10,20,30,40,NA,60,70)
      )
df
df %>% complete(year=full_seq(year,1), qtr)
```

Year	Qtr	Return
2010	1	10
2010	2	20
2010	3	30
2010	4	40
2011	1	NA
2011	2	NA
2011	3	NA
2011	4	NA
2012	1	NA
2012	2	60
2012	3	70
2012	4	NA

Splitting cells in tidyr

separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...**)**

Separate each cell in a column to make several columns.

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174
C	1999	212K	1T
C	2000	213K	1T

separate(table3, rate, sep = "/", into = c("cases", "pop"))

separate_rows(data, ..., sep = "[^[:alnum:]]+", convert = FALSE**)**

Separate each cell in a column to make several rows.

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	rate
A	1999	0.7K
A	1999	19M
A	2000	2K
A	2000	20M
B	1999	37K
B	1999	172M
B	2000	80K
B	2000	174M
C	1999	212K
C	1999	1T
C	2000	213K
C	2000	1T

separate_rows(table3, rate, sep = "/")

Uniting cells in tidyr

unite(data, col, ..., sep = "_", remove = TRUE)

Collapse cells across several columns to make a single column.

table5

country	century	year
Afghan	19	99
Afghan	20	00
Brazil	19	99
Brazil	20	00
China	19	99
China	20	00

→

country	year
Afghan	1999
Afghan	2000
Brazil	1999
Brazil	2000
China	1999
China	2000

unite(table5, century, year, col = "year", sep = "")