Ch6. 데이터 조작

grade_csv.csv 준비

1. dplyr 패키지 활용
dplyr 패키지는 데이터프레임 자료구조를 갖는 데이터를 처리하는데 적합한 패키지

[표 6.1] dplyr 패키지의 주요 함수
tbl_df()
filter()
arrange()
select()
mutate()
summarise()
group_by()
join()
bind()
renames()

1.1 파이프 연산자(%>%)를 이용한 함수 적용

파이프 연산자(%>%): 데이터프레임을 조작하는데 필요한 함수를 순차적으로 적용할 경우 사용할 수 있는 연산자

형식: dataframe %>% 함수1() %>% 함수2()

%>% 연산자는 인수를 함수에 편하게 적용할 수 있다.
%>% 연산자의 >(라이트 앵글 브래킷) 기호는 방향의 의미로 왼쪽에 있는 인자를 오른쪽에 있는 함수에 집어넣는 것이 파이프라인의 기능이다.
%>% 연산자를 사용하면 여러 가지 함수를 한 번에 사용할 수 있다.
%>% 함수의 큰 장점은 한 번에 한 줄로 코드를 사용할 수 있으므로 함수를 사용할 때마다 따로 저장하는 과정이 없이 여러 함수를 실행할 수 있는 편리성에 있다.
%>% 연산자에서 왼쪽 %(퍼센트)를 LHS(Left Hand Side) 변수라고 하며 오른쪽 %(퍼센트)를

RHS(Right Hand Side) 변수라고 한다.

%>% 함수를 생성하여 반환한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% head( ) %>% summary( )
```

실습 (iris 데이터셋을 대상으로 '%>%'연산자를 이용하여 함수 적용
dplyr패키지 설치

```
install.packages("dplyr")
library(dplyr)
iris %>% head()

iris %>% head() %>% subset(Sepal.Length >= 5.0)
```

%>% 함수의 도트(.) 연산자

도트(.) 연산자를 사용하여 LHS가 들어갈 위치를 정할 수 있다.
도트(.) 연산자는 저장한 인자의 이름을 다시 쓰지 않고 데이터를 사용할 수 있다.

Ex.
```
library(dplyr)
# 1:5의 인자를 다시 쓰지 않고 사용할 수 있으며 sum 함수로 호출한다.
1:5 %>% sum(.)

# length 함수에 의해서 길이인 5까지 더해져서 20을 반환한다.
1:5 %>% sum(length(.))
```

sum(length(1:5))와 같이 수열을 지정하여 반환하므로 수열의 길이인 5까지 더해서 20을 반환한다.

```
5 %>% sum(1:.)

# { }(브레이스)를 적용하면 함수를 한 번씩만 실행한다.
5 %>% {sum(1:.)}
csvgrade <- read.csv("grade_csv.csv")

# 1부터 행의 수만큼 수열로 출력하고 나눈 나머지가 0인 행의 부분집합을 추출한다.
csvgrade %>% subset(1:nrow(.) %% 2 == 0)
```

1.2 콘솔 창의 크기에 맞게 데이터 추출

데이터 셋을 대상으로 콘솔 창의 크기에 맞게 데이터를 추출하고, 나머지는 축약형으로 제공

실습 (dply 패키지와 hflights 데이터 셋 설치)

```
install.packages(c("dplyr", "hflights"))
library(dplyr)
library(hflights)

str(hflights)
```

dplyr 로딩
hflights 데이터 로딩

hflights 데이터 셋 설명

실습(hflights 데이터 셋 구조 보기)

실습(tbl_df()함수 사용하기)

```
hflights_df <- tbl_df(hflights)
hflights_df
```

tbl_df()함수: 현재 R의 console창 크기에서 볼수 있는 만큼 결과를 나타내고, 나머지는 아래에 생략된 행 수와 컬럼명 표시

hflights <- hflights %>% tbl_df() 형태로 tbl_df()함수 적용

1.3 조건에 맞는 데이터 필터링

dplyr 패키지의 filter 함수를 사용하여 조건에 맞는 데이터만 추출한다.
filter 함수는 조건에 해당하는 데이터의 모든 컬럼을 추출한다.

형식:
filter(dataframe, 조건1, 조건2)

==(더블 이퀄) 연산자

== 연산자를 적용하여 객체의 특정값을 호출한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(class == 1)
```

!=(엑스클러메이션 포인트 이퀄) 연산자

!= 연산자를 적용하여 객체의 특정값인 아닌 나머지 값들을 호출한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(class != 1)
```

>(라이트 앵글 브래킷) 연산자

> 연산자를 적용하여 객체의 특정값 초과인 값들을 호출한다

Ex.

```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(math > 50)
```

<(레프트 앵글 브래킷) 연산자

< 연산자를 적용하여 객체의 특정값 미만인 값들을 호출한다.

Ex.

```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(math < 50)
```

>=(라이트 앵글 브래킷 이퀄) 연산자

>= 연산자를 적용하여 객체의 특정값 이상인 값들을 호출한다.

Ex.

```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(english >= 80)
```

<=(레프트 앵글 브래킷 이퀄) 연산자

<= 연산자를 적용하여 객체의 특정값 이하인 값들을 호출한다.

Ex.

```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(english <= 80)
```

&(앰퍼샌드) 연산자

& 연산자는 객체를 평가하고 객체 모두 TRUE일 때 TRUE로 평가하고 그렇지 않으면 FALSE로 평가한다.

& 연산자를 적용하여 여러 조건을 충족하는 행을 추출한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(class == 1 & math >= 50)
```

|(버티컬바) 연산자
| 연산자는 객체를 평가하고 객체가 모두 FALSE일 때 FALSE로 평가하고 그렇지 않으면 TRUE로 평가한다.

| 연산자를 적용하여 여러 조건 중 하나 이상 충족하는 행을 추출한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
# 영어 점수가 90점 미만인 학생이거나 과학 점수가 50점 미만인 학생의 데이터를 호출한다.
csvgrade %>% filter(english < 90 | science < 50)
```

| 연산자를 적용하여 목록에 해당하는 행을 추출한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(class == 1 | class == 3 | class == 5)
```

%in% 연산자

%in% 연산자는 값의 포함 여부를 확인하고 반환한다.
%in% 연산자는 여러 객체에서 여러 조건을 줄 때는 사용할 수 없다.

포함된 데이터를 추출하여 반환한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(class %in% c(1, 3, 5))
```

객체 생성
필터링한 데이터의 객체를 생성한다.

데이터의 객체를 생성하여 반환한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
# 객체 생성
class1 <- csvgrade %>% filter(class == 1)
# class1 객체의 math 컬럼에 접근하여 평균 산출
mean(class1$math)
# class1 객체의 english 컬럼에 접근하여 평균 산출
mean(class1$english)
# class1 객체의 science 컬럼에 접근하여 평균 산출
mean(class1$science)
```

실습 (hflights_df를 대상으로 특정일의 데이터 추출하기)

```
filter(hflights_df, Month == 1 & DayofMonth == 2)  # 1월 2일 데이터 추출
```

파이프연산자(%>%)사용하여 hflights_df데이터 셋에 filter()함수 적용:
hflights_df %>% filter(Month ==1 & DayofMonth ==1) 형식

실습 (hflight_df 대상으로 지정된 월의 데이터 추출)

filter(hflights_df, Month == 1 | Month == 2) # 1월 또는 2월 데이터 추출

1.4 컬럼으로 데이터 정렬
데이터 셋의 특정 컬럼을 기준으로 오름차순 또는 내림차순으로 정렬하는 arrange()함수 사용

형식: arrange(dataframe, 컬럼1, 컬럼2, ...) #default는 오름차순
arrange(dataframe, desc(컬럼1, ...)) #desc()함수로 지정하는 경우 내림차순

단일 객체의 오름차순 정렬

오름차순으로 데이터를 정렬하여 추출

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% arrange(math)
```

단일 객체의 내림차순 정렬
arrange 함수를 이용하여 데이터를 내림차순으로 정렬하여 추출하려면 기준 객체에 desc 함수를 적용한다.

내림차순으로 데이터를 정렬하여 추출한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% arrange(desc(math))
```

다중 객체의 오름차순 정렬
arrange 함수를 이용하여 다중 객체의 데이터를 ,(콤마)로 기준으로 오름차순으로 정렬하며 추출한다.

여러 객체의 데이터를 오름차순으로 정렬하여 추출한다.

Ex.

```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% arrange(class, math)
```

실습 (hglights_df를 대상으로 데이터 정렬)

```
arrange(hflights_df, Year, Month, DepTime, ArrTime)
```

오름차순 정렬 시 우선순위는 가장 왼쪽의 컬럼부터 시작하여 오른쪽 컬럼 순으로 정렬

파이프 연산자를 사용하여 hflights_df 데이터 셋에 arrange()함수 적용:
hflights_df %>% arrange(yes, Month, DepTime, AirTime) 형식

1.5 컬럼으로 데이터 검색
데이터 셋의 특정 컬럼을 기준으로 데이터 검색 시 select()함수 사용

형식: select(dataframe, 컬럼1, 컬럼2, ...)

select 함수는 추출하고자 하는 객체를 할당하면 해당 객체만 추출한다.

단일 객체

단일 객체를 추출하여 반환한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% select(math)
```

다중 객체
select 함수의 인자에 ,(콤마)를 활용하여 여러 객체를 추출한다.

여러 객체를 추출하여 반환한다.

```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% select(class, math)
```

객체 제외
select 함수의 인자에 -(마이너스) 연산자를 활용하여 객체 제외하고 추출한다.

객체 제외하고 추출하여 반환한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
```

csvgrade %>% select(-math)

%>% 함수 적용

%>% 함수를 적용하여 함수를 조합하여 구현한다.
%>% 함수를 적용하면 코드의 길이가 줄어들어 이해하기도 쉽고 실행하는데 불필요한 부분이 줄어들어 시간이 단축된다.

특정 객체의 값을 추출한다.

Ex.
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% filter(class == 1) %>% select(english)

특정 객체의 값 일부를 추출한다.

Ex.
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% select(id, math) %>% head(3)

실습 (hflights_df를 대상으로 지정된 컬럼 데이터 검색)

select(hflights_df, Year, Month, DepTime, ArrTime)

hflights_df %>% select(hflights_df, Year, Month, DepTime, AirTime)형식

실습 (hflights_df 대상 컬럼의 범위로 검색)

select(hflights_df, Year:ArrTime)

select()함수 사용시 특정 컬럼만이 아닌 컬럼의 범위 설정 가능
검색조건으로 시작걸럼:종료컬럼 형식으로 컬럼 범위의 시작과 끝을 지정

특정 컬럼 또는 컬럼의 범위를 검색에서 제외하려는 경우 제외하려는 컬럼 이름 또는 범위 앞에 "-"속성을 지정
예) Year부터 DepTime컬럼까지 제외한 나머지 컬럼만 선택하여 검색할 때
select(hflights_df, -(Year:DepTime)) 형식

1.6 데이터셋에 컬럼 추가

데이터 셋에 특정 컬럼을 추가하는 mutate()함수

형식: mutate(dataframe, 컬럼명1=수식1, 컬럼명2=수식2, ...)

실습 (hflights_df에서 출발 지연시간과 도착 지연시간의 차이를 계산한 컬럼 추가)

```
mutate(hflights_df, gain = ArrTime - DepTime,
        gain_per_hour = gain / (AirTime / 60))
```

hflights_df %>% mutate(gain = ArrDelay – DepDelay, gain_per_hour = gain / (AirTime / 60)) 형식

실습 (mutate()함수에 의해 추가된 컬럼 보기)

```
select(mutate(hflights_df, gain = ArrDelay - DepDelay,
                gain_per_hour = gain / (AirTime / 60)),
        Year, Month, ArrDelay, DepDelay, gain, gain_per_hour)
```

console참 크기 이외의 컬럼명 확인이 어려운 경우 select()함수 안에 mutate()함수 사용

1.7 요약통계 구하기
데이터 셋에 특정 컬럼을 대상으로 기술통계량을 계산하는 summarise() 함수

형식: summarise(dataframe, 추가할 컬럼명 = 함수(컬럼명), ...)

컬럼의 평균과 같이 컬럼을 요약한 통계량을 구할 때는 summarise 함수와 group_by 함수를 사용한다.
구한 결과를 요약표로 만들면 컬럼의 집단 간에 어떤 차이가 있고 어떤 분포를 갖는지 쉽게 파악할 수 있다.
그룹으로 나눈 데이터는 다른 분석에도 사용할 수 있으므로 활용도가 매우 높다.

전체의 평균, 표준편차, 사분위수 등 전체적인 값들에 대한 요약 통계량을 산출할 때는 summary 함수를 사용하고, 개별 컬럼의 데이터에 대한 요약 통계량을 구할 때는 summarise 함수를 사용한다.

컬럼 데이터의 요약 통계량을 반환한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% summarise(mean_math = mean(math))
```

실습 (hflights_df에서 비행시간의 평균 구하기)

```
summarise(hflights_df, avgAirTime = mean(AirTime, na.rm = TRUE))
# hflights_df %>% summarise(avgAirTime = mean(AirTime, na.rm = TRUE))
```

mean()함수를 사용하여 평균을 계산하여 avgAirTime변수에 저장
hflights_df %>% summarise(avgAirTime = mean(AirTime, na.rm=TRUE)) 형식 사용

실습 (hflights_df 의 관측치 길이 구하기)

```
summarise(hflights_df, cnt = n(),
```

```
            delay = mean(AirTime, na.rm = TRUE))
```

n()함수: 데이터 셋의 관측치 길이를 구하는 함수

실습 (도착시간(AirTime)의 표준편차와 분산 계산)

```
summarise(hflights_df, arrTimeSd = sd(ArrTime, na.rm = TRUE),
            arrTimeVar = var(ArrTime, na.rm = T))
```

1.8 집단변수 대상 그룹화

데이터 셋 내 범주형 컬럼을 대상으로 그룹화하는 group_by()함수 사용

형식: group_by(dataframe, 집단변수)

group_by 함수에 객체를 지정하면 컬럼별로 분리한다.

컬럼의 집단별 평균 요약 통계량

컬럼의 집단별 평균에 대한 요약 통계량을 반환한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% group_by(class) %>% summarise(mean_math = mean(math))
```

group_by 함수는 인자로 범주형 변수가 포함된 컬럼명을 인자로 받아 요약 통계량을 계산

컬럼의 집단별 다중 요약 통계량

컬럼의 집단별 다중 요약 통계량을 반환한다.

Ex.
```
library(dplyr)
csvgrade <- read.csv("grade_csv.csv")
csvgrade %>% group_by(class) %>% summarise(mean_math = mean(math), sum_math
= sum(math), median_math = median(math))
```

실습 (집단변수를 이용하여 그룹화)

```
species <- group_by(iris, Species)
str(species)
species
```

species <- iris %>% group_by(species) 형식 이용

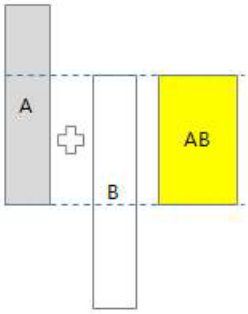
1.9 데이터프레임 병합

서로 다른 데이터프레임을 대상으로 공통 컬럼을 이용하여 하나의 데이터프레임을 병합하는 join() 함수

[표 6.2] join 관련 함수
inner_join(df1, df2, x)
left_join(df1, df2, x)
right_join(df1, df2, x)
full_join(df1, df2, x)

inner_join 함수

inner_join 함수는 키를 기준으로 열에서 일치하는 열만 결합한다.



형식: inner_join(dataframe1, dataframe2, 공통변수)
inner_join(A, B, by = key)
where
A 매개변수: 객체를 설정한다.
B 매개변수: 객체를 설정한다.
by 옵션: 결합할 기준이 되는 칼럼을 설정한다.

Ex.

```
library(dplyr)
a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
b <- data.frame(id = c(3, 4 , 5, 6, 7), weight = c(80, 90, 85, 60, 85))
# id 열을 일치하는 열만 결합한다.
merge(a, b, by="id")
inner_join(a, b, by = "id")
```

P180 실습 (공통변수를 이용하여 내부조인(inner_join)하기
1단계: join 실습용 데이터프레임 생성

```
df1 <- data.frame(x = 1:5, y = rnorm(5))
df2 <- data.frame(x = 2:6, z = rnorm(5))
```

df1

df2

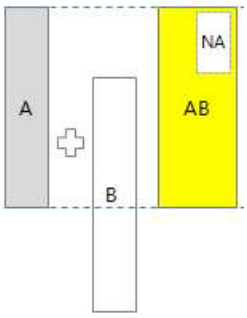
2단계: inner_join하기

inner_join(df1, df2, by = 'x')

inner_join : 내부조인

left_join 함수

left_join 함수는 키를 기준으로 왼쪽 열을 결합한다.



형식: left_join(dataframe1, dataframe2, 공통변수)
left_join(A, B, by = key)
where
A 매개변수: 객체를 설정한다.
B 매개변수: 객체를 설정한다.
by 옵션: 결합할 기준이 되는 칼럼을 설정한다.

Ex.

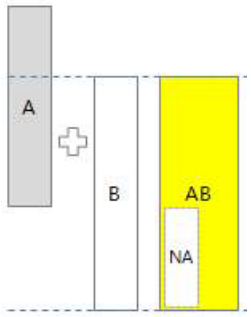
```
library(dplyr)
a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
b <- data.frame(id = c(3, 4, 5, 6, 7), weight = c(80, 90, 85, 60, 85))
# id 열의 왼쪽 열 기준으로 결합한다.
merge(a, b, by="id", all.x = T)
left_join(a, b, by = "id")
```

실습 (공통변수를 이용하여 왼쪽 조인(left_join)하기

```
left_join(df1, df2, by = 'x')
```

right_join 함수

right_join 함수는 키를 기준으로 오른쪽 열을 결합한다.



형식: right_join(dataframe1, dataframe2, 공통변수)
right_join(A, B, by = key)
where
A 매개변수: 객체를 설정한다.
B 매개변수: 객체를 설정한다.
by 옵션: 결합할 기준이 되는 칼럼을 설정한다.

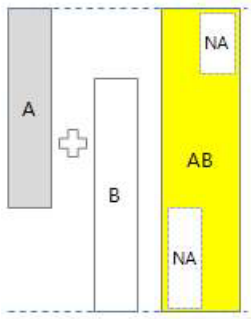
Ex.
```
library(dplyr)
a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
b <- data.frame(id = c(3, 4, 5, 6, 7), weight = c(80, 90, 85, 60, 85))
# id 열의 오른쪽 열 기준으로 결합한다.
merge(a, b, by = "id", all.y = T)
right_join(a, b, by = "id")
```

실습 (공통변수를 이용하여 오른쪽 조인(right_join)하기)

```
right_join(df1, df2, by = 'x')
```

full_join 함수

full_join 함수는 키를 기준으로 모든 열을 결합한다.



형식: full_join(dataframe1, dataframe2, 공통변수)
full_join(A, B, by = key)
where
A 매개변수: 객체를 설정한다.
B 매개변수: 객체를 설정한다.
by 옵션: 결합할 기준이 되는 칼럼을 설정한다.

Ex.

```
library(dplyr)
a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
b <- data.frame(id = c(3, 4, 5, 6, 7), weight = c(80, 90, 85, 60, 85))
# id 열에 모든 열을 결합한다.
merge(a, b, by="id", all = T)
full_join(a, b, by = "id")
```

실습 (공통변수를 이용하여 전체 조인(full_join)하기)

full_join(df1, df2, by = 'x')

* 다른 한쪽의 데이터프레임에서 x의 값이 없는 경우 결측치(NA)로 나타난다.

1.10 데이터프레임 합치기

서로 다른 데이터프레임을 대상으로 행 단위 또는 열 단위로 합치는 함수

[표 6.3] bind관련 함수
bind_rows(df1, df2)
bind_cols(df1, df2)

세로 결합

dplyr 패키지의 bind_rows 함수로 나뉘어져 있는 데이터를 세로로 결합할 수 있다.

형식: bind_rows(dataframe1, dataframe2)

Ex.
```
library(dplyr)
a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
b <- data.frame(id = c(3, 4, 5, 6, 7), weight = c(80, 90, 85, 60, 85))
# 세로로 결합한다.
bind_rows(a, b)
```

실습 (두 개의 데이터프레임을 행 단위로 합치기)

```
df1 <- data.frame(x = 1:5, y = rnorm(5))
df2 <- data.frame(x = 6:10, y = rnorm(5))

df1
df2

df_rows <- bind_rows(df1, df2)
df_rows
```

rbind 함수로 행을 결합하여 반환한다.

rbind 함수로 행을 결합하기 위해서는 Data Frame의 열 개수, 칼럼 이름이 같아야 한다

형식: rbind(a, b)

Ex.

```
a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
a
b <- data.frame(id = c(6, 7 , 8), score = c(80, 90, 85))
b
rbind(a, b)
```

가로 결합

dplyr 패키지의 bind_cols 함수로 나뉘어져 있는 데이터를 가로로 결합할 수 있다.

형식: bind_cols(dataframe1, dataframe2)

실습 (두 개의 데이터프레임을 열 단위로 합치기)

```
df_cols <- bind_cols(df1, df2)
df_cols
```

cbind 함수로 열을 결합하여 반환한다.
cbind 함수로 열을 결합하기 위해서는 Data Frame의 행 개수가 서로 같아야 한다.

형식: cbind(a,b)

Ex.

```
a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
a
b <- data.frame(age = c(20, 19 , 20, 19, 21), weight = c(80, 90, 85, 60, 85))
b
```

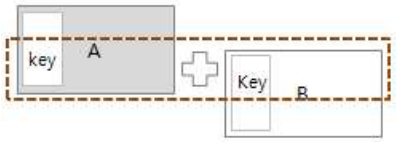
cbind(a, b)

열의 잘못된 결합으로 반환한다.

Ex.
# 관측치가 서로 같은 3, 4, 5가 있고 서로 다른 1, 2, 6, 7이 있다.
a <- data.frame(id = c(1, 2, 3, 4, 5), score = c(60, 80, 70, 90, 85))
a
b <- data.frame(id = c(3, 4 , 5, 6, 7), weight = c(80, 90, 85, 60, 85))
b
cbind(a, b)

다른 관측치 열의 결합
열을 결합하기 위해서는 Data Frame의 행 개수가 서로 같지 않아도 된다.
merge 함수로 열을 결합하여 반환한다.

merge 함수
merge 함수는 키를 기준으로 열을 결합하여 반환한다.



형식: merge(A, B, by = key, all = FALSE, all.x = all, all.y = all) 함수
where
A 매개변수: 객체를 설정한다.
B 매개변수: 객체를 설정한다.
by 옵션: 결합할 기준이 되는 칼럼을 설정한다.
all 옵션: 모든 열을 결합하며 기본값은 FALSE다.
all.x 옵션: A 객체를 기준으로 열을 결합한다.
all.y 옵션: B 객체를 기준으로 열을 결합한다.

1.11 컬럼명 수정하기

데이터프레임을 구성하는 컬럼명을 수정하는 rename()함수 사용

형식: rename(데이터프레임, 변경후컬럼명 = 변경전컬럼명)

객체명을 변경하여 반환한다.

Ex.

```
library(dplyr)
df <- data.frame(one = c(4, 3, 8))
df

df <- rename(df, "원" = one)
df
```

실습 (데이터프레임의 컬럼명 수정)

```
df_rename <-rename(df_cols, x2 = x1)
df_rename <- rename(df_rename, y2 = y1)
df_rename
```

2. reshape2 패키지 활용

reshape패키지의 기본 골격만을 대상으로 개발된 패키지
melt()함수와 dcast/acast()함수를 적용하여 집단변수를 통해서 데이터의구조를 유연하게 변경해주는 기능을 제공

2.1 긴 형식을 넓은 형식으로 변경
dcast()함수: 긴 형식(Long format)의 데이터를 넓은 형식(Wide format)으로 변경

실습 (reshape2 패키지 설치와 데이터 가져오기)

```
install.packages("reshape2")
data <- read.csv("C:/Rwork/Part-II/data.csv")
data
library(reshape2)
```

reshape2 패키지 설치
data.csv파일 준비

data.csv 데이터 셋 설명

형식 dcast(데이터셋, 앞변수 ~뒤변수, 적용함수)

실습 (넓은 형식(wide format)으로 변경)

```
wide <- dcast(data, Customer_ID ~ Date, sum)
wide
```

앞변수(Customer_ID)는 행으로 구성
뒷변수(Date)는 열로 구성
Buy변수는 측정변수로 사용

실습 (파일 저장 및 읽기)
row.names=FALSE 속성을 이용하여 행 번호 없이 저장

```
setwd("C:/Rwork/")
write.csv(wide, "wide.csv", row.names = FALSE)

wide <- read.csv("wide.csv")
colnames(wide) <- c('Customer_ID', 'day1', 'day2', 'day3',
                    'day4', 'day5', 'day6', 'day7')
wide
```

2.2 넓은 형식을 긴 형식으로 변경
melt()함수: 넓은 형식(wide format)을 긴 형식(long format)으로 변경

형식: melt(데이터셋, id="컬럼명")

실습 (넓은 형식의 데이터를 긴 형식으로 변경)
1단계: 데이터를 긴 형식으로 변경

```
long <- melt(wide, id = "Customer_ID")
long
```

2단계: 컬럼명 변경

```
name <- c("Customer_ID", "Date","Buy")
colnames(long) <- name
head(long)
```

실습 (smiths 데이터 셋 확인하기)
1단계: smiths 데이터 셋 가져오기

```
data("smiths")
smiths
```

2단계: 넓은 형식의 smiths 데이터 셋을 긴 형식으로 변경

```
long <- melt(id = 1:2, smiths)
long
```

3단계: 긴 형식을 넓은 형식으로 변경하기
subject와 time컬럼 기준: subject+time ~

```
dcast(long, subject + time ~ ...)
```

2.3 3차원 배열 형식으로 변경

dcast()함수: 데이터프레임 형식으로 구조를 변경
acast()함수: 3차원 구조를 갖는 배열형태로 변경

실습(airquality 데이터 셋의 구조 변경)

1단계: airquality 데이터셋 가져오기

```
data('airquality')
str(airquality)
airquality
```

airquality 데이터 셋 설명

2단계: 컬럼제목을 대문자로 일괄 변경

```
names(airquality) <- toupper(names(airquality))
head(airquality)
```

toupper()함수

3단계: melt()함수를 이용하여 넓은 형식을 긴 형식으로 변경

```
air_melt <- melt(airquality, id = c("MONTH", "DAY"), na.rm = TRUE)
head(air_melt)
```

month(월)과 day(일) 기준 긴 형식

4단계: acast()함수를 이용하여 3차원으로 구조 변경

```
names(air_melt) <- tolower(names(air_melt))
```

```
acast <- acast(air_melt, day ~ month ~ variable)
acast
class(acast)
```

day~month~variable 의 3차원 배열

```
# 집합함수 적용하기
acast(air_melt, month ~ variable, sum, margins = TRUE)
```

acast()함수: month 컬럼을 기준으로 측정변수의 합계 계산
margin=TRUE 속성: 각 행과 열의 합계(all) 컬럼 추가
3차 구조인 air_melt데이터셋에 dcast()함수 적용 시 데이터프레임 형태로 구조가 변경

연습문제 풀기