Data Handling

데이터의 정제와 가공: 전처리

with dplyr

데이터 전처리

- 주어진 데이터를 그대로 사용하는 경우보다 분석에 적합하게 데이터를 가공하여 사용하는 것이 유리
- 데이터 전처리 : Data Preprocessing
  - 데이터의 일부를 추출
  - 데이터를 종류별로 분류
  - 여러 데이터를 합치는 작업 등

데이터 전처리를 위한 준비
: dplyr 패키지

- dplyr 패키지
  - Hardley Wickham 교수가 만든 패키지로 data.frame 객체의 데이터를 쉽게 다룰 수 있도록 도와주는 패키지
  - 데이터 전처리 작업에 가장 많이 사용되는 패키지
  - Data Visualizaion을 위한 ggplot2 패키지와 궁합이 잘 맞고 다루기 쉬워 널리 애용하는 패키지

- dplyr 패키지의 설치와 사용

```
> install.packages("dplyr")
> library(dplyr)
```

데이터 전처리를 위한 준비
: dplyr 패키지

- dplyr의 주요 함수

함수	기능
filter()	행 추출
select()	열(변수) 추출
arrange()	정렬
mutate()	변수 추가
summarise()	통계치 산출
group_by()	그룹으로 나누기
left_join()	데이터 합치기(열)
bind_rows()	데이터 합치기(행)

데이터 전처리를 위한 준비
: 데이터의 준비

- 데이터의 준비

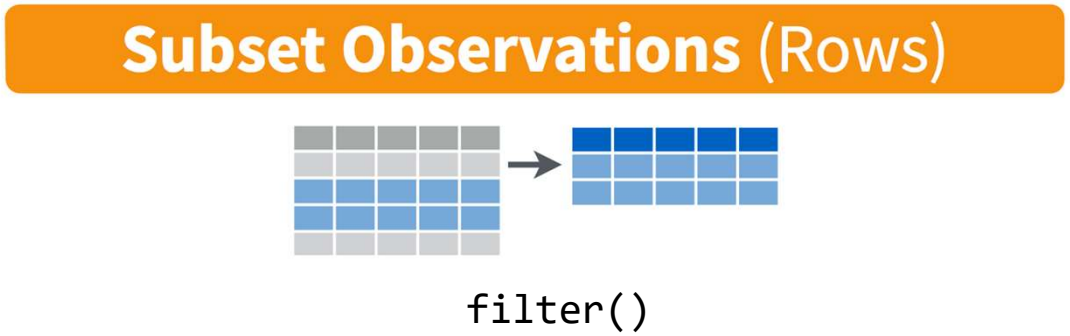
```
> scores <- read.csv("class_scores.csv")
> scores
```

- 데이터의 확인 : head, tail, str, summary 등으로 데이터를 확인해 봅시다

```
> summary(scores)
     Stu_ID          grade   class   gender       Math       
 Min.   :10101   Min.   :1   A:120   F:286   Min.   : 25.00  
 1st Qu.:10431   1st Qu.:1   B:120   M:314   1st Qu.: 45.75  
 Median :20321   Median :2   C:120           Median : 64.00  
 Mean   :20321   Mean   :2   D:120           Mean   : 63.66  
 3rd Qu.:30210   3rd Qu.:3   E:120           3rd Qu.: 83.00  
 Max.   :30540   Max.   :3                   Max.   :100.00  

...
```

데이터 전처리
: filter - 조건에 맞는 데이터 추출



- filter 함수의 사용

```
filter({데이터}, {필터링 조건})
```

```
> filter(scores, grade == 1) # scores 데이터프레임에서 grade가 1인 데이터만 추출
> filter(scores, gender == 'F') # scores 데이터프레임에서 gender가 F인 데이터만 추출
> head(filter(scores, Writing > 60)) # 타 함수와의 조합
```

- 여러 조건을 부여하고자 할 때는 논리 연산자를 이용하여 조합

```
> filter(scores, grade == 1 & class == 'B') # grade가 1이고 class가 B인 학급의 데이터
> filter(scores, English >= 80 | Writing >= 80) # 영어 점수가 80점 이상이거나 Writing
점수가 80점 이상인 데이터
```

데이터 전처리
: filter - 조건에 맞는 데이터 추출

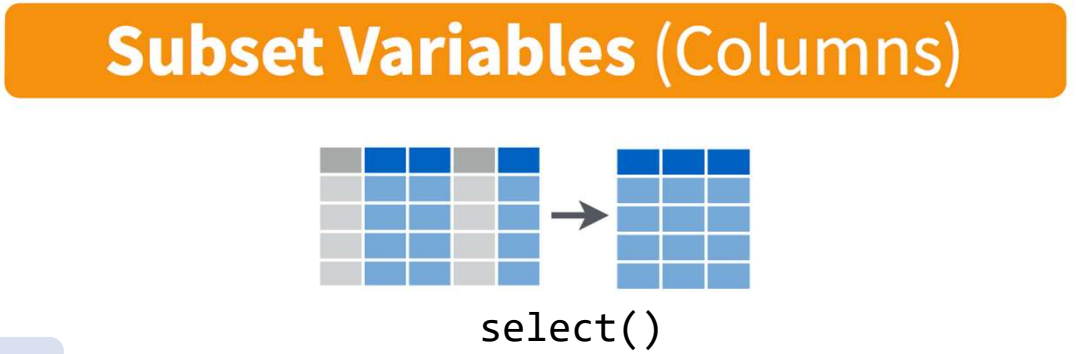
- %in% 연산자를 사용하면 조건 목록을 부여할 수 있음

```
> filter(scores, grade == 3 & class %in% c("A", "C", "E"))
```

- 정제된 데이터를 새 변수에 저장하기

```
> scores.grade1 <- filter(scores, grade == 1)
> scores.grade2 <- filter(scores, grade == 2)
> scores.grade3 <- filter(scores, grade == 3)
```

데이터 전처리
: select - 필요한 변수 추출



- select 함수의 사용

```
select({데이터}, {추출 변수 목록})
```

```
> select(scores, Math) # scores 데이터프레임에서 Math 변수 항목만 추출
```

- 여러 변수의 추출

```
> select(scores, Math, English, Writing) # 필요한 변수 항목만 추출
```

- 특정 변수의 배재 : -를 이용

```
> select(scores, -Stu_ID, -gender)
```

- :을 이용한 범위 변수 출력(인접 변수 선택)

```
> select(scores, Math:Writing)
```

데이터 전처리
: mutate - 파생변수의 추가

- mutate 함수의 사용

```
mutate({데이터}, 파생변수명 = {공식})
```

```
> mutate(scores, Total = Math + English + Science + Marketing + Writing)
```

- 여러 파생변수를 동시에 생성
  - 파생변수 생성 공식을 ,로 나열

```
> mutate(scores, Total = Math + English + Science + Marketing + Writing,
+     Avg = (Math + English + Science + Marketing + Writing) / 5)
```

데이터 전처리
: dplyr 함수의 조합

- [연습] 다음과 같이 처리해 봅시다
  - scores 데이터프레임에서 grade == 1, gender == F인 데이터를 추출하고
  - 합계(Total)와 평균(Avg) 파생변수를 만들고
  - Math, English, Science, Marketing, Writing 변수를 제외한 나머지 변수만 추출하여
  - scores.refined 변수에 담아봅시다

데이터 전처리
: dplyr 함수의 조합

- [Solution 1]

```
> temp.filtered <- filter(scores, gender == 'F' & grade == 1)
> temp.mutated <- mutate(temp.filtered,
+                        Total = Math + English + Science + Marketing + Writing,
+                        Avg = (Math + English + Science + Marketing + Writing) / 5)
> temp.selected <- select(temp.mutated,
+                         -Math, -English, -Science, -Marketing, -Writing)
> head(temp.selected)
...
```

데이터 전처리
: dplyr 함수의 조합

- [Solution 2]

```
> scores.refined <- select(
+   mutate(
+     filter(scores, gender == 'F' & grade == 1),
+     Total = Math + English + Science + Marketing + Writing,
+     Avg = (Math + English + Science + Marketing + Writing) / 5
+   ), -Math, -English, -Science, -Marketing, -Writing
+ )
```

- 뭔가 매우 불편
  - 생각의 순서와 함수의 작성 방향이 반대
  - 내포가 많아져 코드가 복잡해짐

데이터 전처리
: dplyr 함수의 조합 : with Chain Operator (%>%)

- Chain Operator : %>%
  - 각 단계별로 중간 결과를 보관하기 위한 임시 변수를 만들 필요가 없고
  - 앞 단계의 결과를 뒤 함수의 입력 값으로 활용하는 방식으로 논리의 흐름에 따른 코드 작성 가능
  - 간결하고 가독성 높은 코드를 작성할 수 있음

```
> scores.refined <- scores %>%
+   filter(gender == 'F' & grade == 1) %>%
+   mutate(Total = Math + English + Science + Marketing + Writing) %>%
+   mutate(Avg = Total / 5) %>%
+   select(-Math, -English, -Science, -Marketing, -Writing)
```

데이터 전처리
: mutate - 파생변수의 추가

- mutate에 ifelse() 를 적용하면 조건에 따라 다른 값을 갖는 파생변수를 만들 수 있음
  - 연습) scores.refined 데이터셋에 파생변수 Result를 다음 조건에 맞게 추가해 봅시다
    - 평균(Avg) >= 60 : PASS
    - 평균(Avg) < 60 : FAIL

```
> scores.result <- scores.refined %>%
+   mutate(Result = ifelse(Avg >= 60, "PASS", "FAIL"))
>
> head(scores.result)
 Stu_ID grade class gender Total  Avg Result
1  10103     1     A      F   317 63.4   PASS
2  10104     1     A      F   330 66.0   PASS
3  10105     1     A      F   311 62.2   PASS
4  10108     1     A      F   420 84.0   PASS
5  10114     1     A      F   221 44.2   FAIL
6  10115     1     A      F   305 61.0   PASS
```

데이터 전처리
: arrange - 데이터의 정렬

- arrange 함수의 사용

```
arrange({데이터}, 정렬기준변수)
```

- 오름차순 정렬

```
> arrange(scores.result, Avg)
> scores.result %>% arrange(Avg)
```

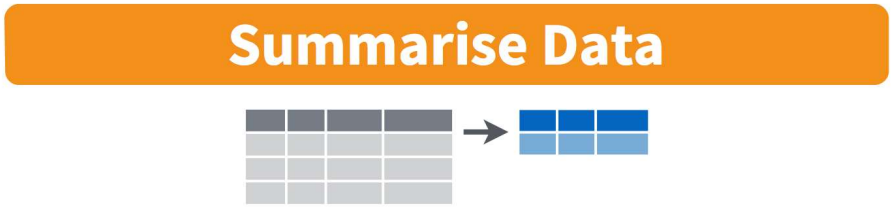
- 내림차순 정렬 : desc() 함수로 정렬기준변수를 감싸줌

```
> arrange(scores.result, desc(Avg))
> scores.result %>% arrange(desc(Avg))
```

- 정렬 기준이 여러 개일 때: 콤마(,)로 정렬 기준을 나열

```
> arrange(scores, desc(Writing), desc(English))
> scores %>% arrange(desc(Writing), desc(English))
```

데이터 전처리
: summary - 데이터의 요약



- summary 함수의 사용

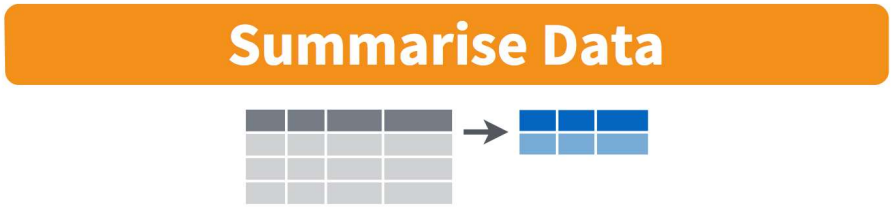
```
summary({데이터}, 요약함수)
```

- 기본 사용법

```
> summarise(scores, mean(Math)) # 수학 점수의 평균
  mean(Math)
1   63.66167

> summarise(scores, mean.math = mean(Math)) # 요약 내용에 변수명을 부여
  mean.math
1  63.66167
```

데이터 전처리
: summary - 데이터의 요약



- 여러 통계량 일괄 산출
  - 산출할 통계량 변수를 차례로 나열

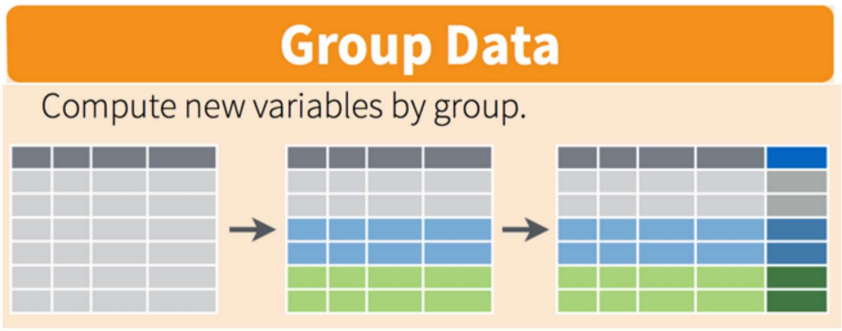
```
> summarise(scores, mean.math = mean(Math),
+                   mean.english = mean(English),
+                   mean.science = mean(Science))
  mean.math mean.english mean.science
1  63.66167     63.00667     63.42833
```

- summarise에 자주 사용하는 요약 통계량 함수

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

데이터 전처리

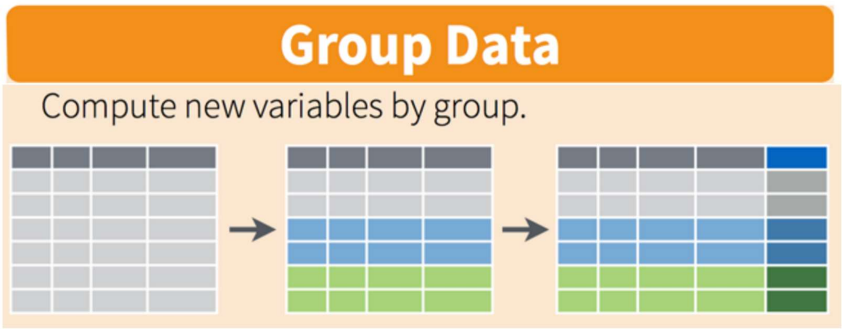
: group_by - 데이터를 집단별로 구분



- 단독으로 사용하기보다는 통계량 산출을 위해 데이터를 집단적으로 구분하는 기능 제공

```
> group_by(scores, grade)
# A tibble: 600 x 9
# Groups:   grade [3]
   Stu_ID grade class gender  Math English Science Marketing Writing
    <int> <int> <fct> <fct>  <int>   <int>   <int>     <int>   <int>
 1  10101     1 A     M         55      84      50        40      59
 2  10102     1 A     M         29      94      41        87      57
 3  10103     1 A     F         28      80      68        41     100
 4  10104     1 A     F         45      74      48        88      75
 5  10105     1 A     F         28      69      57        91      66
 6  10106     1 A     M         86      51      68        59      92
 7  10107     1 A     M         47      52      98        81      79
 8  10108     1 A     F         80      67      98        85      90
 9  10109     1 A     M         30      32      74        86      52
10  10110     1 A     M         94      39      90        67      28
# ... with 590 more rows
```

데이터 전처리
: group_by - 데이터를 집단별로 구분



- 집단별로 나눈 데이터를 대시 소집단으로 구분
  - scores 데이터 프레임을 grade로 1차 분류, class로 2차 문류하는 예

```
> group_by(scores, grade, class)
```

- 구분된 데이터의 통계량을 확인해 봅시다 (Math, English, Writing의 평균)

```
> summarise(
+   group_by(scores,
+            grade,
+            class),
+   mean.math = mean(Math),
+   mean.english = mean(English),
+   mean.writing = mean(Writing)
+ )
```

데이터 전처리
: dplyr 조합하기

- [예제] 코드를 확인하고 그 의미를 유추해 봅시다

```
scores %>%
  filter(grade == 1) %>%
  group_by(class) %>%
  mutate(Total = Math + English + Science + Marketing + Writing) %>%
  summarise(sum_tot = sum(Total), mean_tot = mean(Total)) %>%
  arrange(desc(mean_tot)) %>%
  head(3)
```

Data Handling

결측치와 이상치

결측치와 이상치

- 결측치(Missing Value) : 누락된 값, 비어 있는 값
- 이상치(Outlier) : 측정된 값은 있으나 정상 범주에서 벗어나 극단적으로 크거나 작은 경우
  - 이상치가 포함되어 있으면 분석 결과가 왜곡되므로 분석에 앞서 이상치를 제거하는 작업이 필요

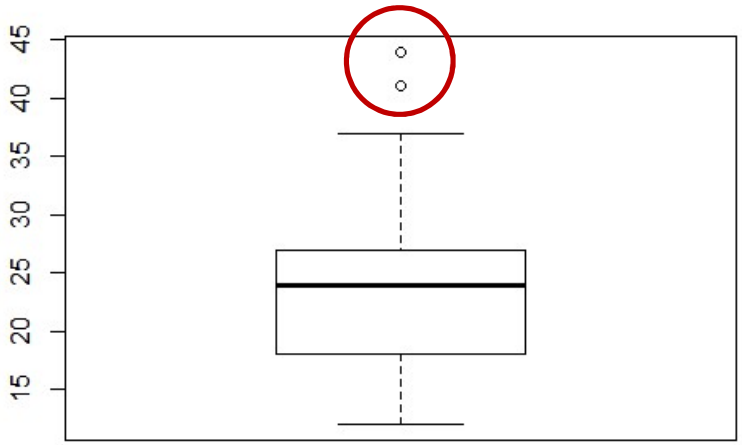
- R에서 결측치는 NA로 표기
  - 결측치를 판단하려면 is.na() 함수 이용

- 이상치를 제거하려면 정상 범위에 대한 명확한 규정이 필요
  - 논리적 판단
  - 통계적인 기준을 이용 : 상자 그림(Box Plot) 등을 이용

이상치의 제거

- ggplot의 데이터 mpg를 이용, 이상치를 제거해 봅시다

```
> library(ggplot2)
> bp <- boxplot(mpg$hwy)
> bp$stats
     [,1]
[1,]   12
[2,]   18
[3,]   24
[4,]   27
[5,]   37
attr(,"class")
        1
"integer"
```

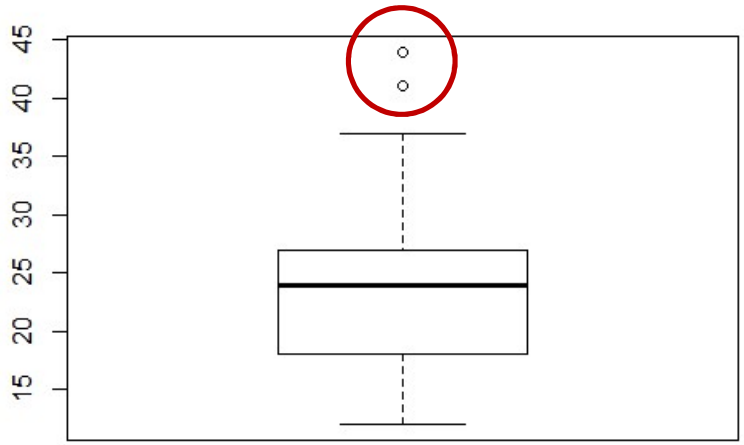


- Box Plot의 stats 값을 보고 이상치를 확인해 봅시다

이상치의 제거

- 이상치의 범위를 판단하고 해당 데이터의 값을 추출해 봅시다
  - 이 경우, 정상 범위는 12 ~ 37 사이임을 알 수 있습니다.

```
> mileage <- mpg[, c("cyl", "cty", "hwy")]
> library(dplyr)
> mileage %>%
+   filter(hwy < 12 | hwy > 37)
# A tibble: 3 x 3
    cyl   cty   hwy
  <int> <int> <int>
1     4    33    44
2     4    35    44
3     4    29    41
```



- 이상치 데이터를 모두 결측 처리

```
> mileage$hwy <- ifelse(mileage$hwy < 12 | mileage$hwy > 37, NA, mileage$hwy)
```

결측치의 제거

- 결측치의 확인, 결측 빈도 측정

```
> is.na(mileage)
> table(is.na(mileage$hwy))

FALSE  TRUE
  231     3
```

- 결측치(NA)가 포함된 데이터는 함수에 적용하면 정상 연산되지 않고 NA를 반환

```
> sum(mileage$hwy)
[1] NA
> mean(mileage$hwy)
[1] NA
```

- 결측치를 포함하지 않고 통계 함수를 실행하려면 rm.na 옵션을 TRUE로 부여

```
> sum(mileage$hwy, na.rm = TRUE)
[1] 5356
> mean(mileage$hwy, na.rm = TRUE)
[1] 23.18615
```

결측치의 대체

- 결측치가 많은 데이터의 경우, 결측치를 무작정 제거하면 너무 많은 데이터 손실로 분석 결과가 왜곡
  - 평균, 최빈값 등 대표값을 구해 결측치를 하나의 값으로 일괄 대체

- mileage$hwy의 데이터 결측치를 중앙값으로 대체해 봅니다

```
> median(mileage$hwy, na.rm = TRUE)
[1] 24
> mileage$hwy <- ifelse(is.na(mileage$hwy), 24, mileage$hwy)
> mean(mileage$hwy)
[1] 23.19658
```