4장. dplyr에 의한 데이터 다듬기

데이터 프레임 다루기

1

dplyr

- 기본 dplyr 함수
  - 행을 작업 대상으로 하는 함수
  - 열을 작업 대상으로 하는 함수
  - 요약 통계량 계산 함수
- 그룹 데이터 프레임
  - 그룹 데이터 프레임 생성
  - 그룹 데이터 프레임에서 기본 dplyr 함수의 작동 방식
- 여러 열을 대상으로 동일한 작업 수행
- 행 단위 작업 수행

2

2

기본 dplyr 함수

- 행을 작업 대상으로 하는 함수
  - `filter()`, `slice()`, `arrange()`, `distinct()`
- 열의 작업 대상으로 하는 함수
  - `select()`, `rename()`, `rename_with()`, `mutate()`, `transmute()`, `relocate()`
- 요약 통계량 계산 함수
  - `summarise()`
- 기본 dplyr 함수들의 공통점
  - 첫 번째 입력 요소는 데이터 프레임(또는 tibble)
  - 각 함수 내에서는 인덱싱 없이 데이터 프레임의 변수 사용 가능
  - 각 함수의 결과물은 데이터 프레임(또는 tibble)

pipe 기능으로 더 효율적인 프로그램 작성 가능

3

3

- Pipe 기능
  - 명령문을 서로 연결하여 한 명령문의 결과물을 바로 다음 명령문의 입력 요소로 직접 사용할 수 있도록 하는 기능
  - pipe 연산자: %>% (RStudio 단축키: Shift+Ctrl+M)
  - 기본적인 형태:
    - `lhs %>% rhs`
      - `lhs`: 데이터 객체 또는 데이터 객체를 생성하는 함수
      - `rhs`: `lhs`를 입력 요소로 하는 함수
  - 예:
    - `x %>% f() -> f(x)`
    - `x %>% f(y) -> f(x,y)` :첫번째 요소
    - `x %>% f(y,.) -> f(y,x)` :첫번째 요소가 아닌 경우 해당 위치에 “.” 표시
  - 기타 pipe연산자: `%<>%`, `%$%`, `%T>%`

4

4

- 예제

```
> library(dplyr)
> x <- 1:5
> y <- log(x)
```

```
> mean(x)                          #  f(x)
[1] 3
> x%>%mean()                       #  x %>% f()
[1] 3
```

```
> plot(x,y)                        # f(x,y)
> x%>%plot(y)                      # x %>% f(y)
> y%>%plot(x,.)                    # y %>% f(x,.)
```

```
> mean(log(x))
[1] 0.9574983
> x%>%log()%>%mean()
[1] 0.9574983
>
> sqrt(mean(log(x)))
[1] 0.9785184
> x%>%log()%>%mean()%>%sqrt()
[1] 0.9785184
```

5

5

- Pipe, 할당문

```
> x=1:5
> u=x %>% mean                     #  = 오른쪽 명령문(들) 결과 할당
> u
[1] 3
> v<-x %>% mean
> v
[1] 3
> x %>% mean->w                    #  ->
> w
[1] 3
```

```
> s=as_tibble(airquality) %>% print(n = 3) # print():대상객체 일부 인쇄
# A tibble: 153 x 6                        #                 전부 생성
  Ozone Solar.R  Wind  Temp Month   Day
  <int>   <int> <dbl> <int> <int> <int>
1    41     190   7.4    67     5     1
2    36     118   8      72     5     2
3    12     149  12.6    74     5     3
# ... with 150 more rows
> dim(s)
[1] 153   6
```

6

6

4.1 행을 작업 대상으로 하는 함수

- 조건에 의한 행 선택: `filter()`
  - 조건을 모두 만족하는 행 선택
  - 조건 설정에 사용되는 연산자
    - 비교 연산자(>, >=, <, <=, ==, !=)
    - 논리 연산자(&, |, !)
    - %in% 연산자

7

7

- 예제: mtcars
  - 변수 mpg의 값이 30 이상인 자동차 선택

```
> library(tidyverse)
> mtcars_t <- as_tibble(mtcars)        #mtcars:32대 차량, 11개 변수
> mtcars_t %>% filter(mpg >= 30)
# A tibble: 4 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  32.4     4  78.7    66  4.08  2.2   19.5     1     1     4     1
2  30.4     4  75.7    52  4.93  1.62  18.5     1     1     4     2
3  33.9     4  71.1    65  4.22  1.84  19.9     1     1     4     1
4  30.4     4  95.1   113  3.77  1.51  16.9     1     1     5     2
```

- mpg의 값이 30 이상이고 wt의 값이 1.8 미만

```
> mtcars_t %>% filter(mpg >= 30, wt < 1.8)
# A tibble: 2 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  30.4     4  75.7    52  4.93  1.62  18.5     1     1     4     2
2  30.4     4  95.1   113  3.77  1.51  16.9     1     1     5     2
```

논리 연산자 '&' 대신 콤마(,) 사용 가능

8

8

- 변수 mpg가 30 이하, 변수 cyl이 6 또는 8, 변수 am이 1(manual)인 자동차 선택

```
> mtcars_t %>% filter(mpg <= 30, cyl %in% c(6,8), am == 1)
# A tibble: 5 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21       6   160   110  3.9   2.62  16.5     0     1     4     4
2  21       6   160   110  3.9   2.88  17.0     0     1     4     4
3  15.8     8   351   264  4.22  3.17  14.5     0     1     5     4
4  19.7     6   145   175  3.62  2.77  15.5     0     1     5     6
5  15       8   301   335  3.54  3.57  14.6     0     1     5     8
```

cyl == 6|cyl == 8 보다 cyl %in% c(6,8)이 더 좋은 선택

9

9

- 변수 mpg의 값이 mpg의 중앙값과 Q_3 사이에 있는 자동차 선택

```
> mtcars_t %>%
    filter(
      mpg >= median(mpg), mpg <= quantile(mpg, probs = 0.75)
    ) %>%
    print(n=3)
# A tibble: 10 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21       6   160   110  3.9   2.62  16.5     0     1     4     4
2  21       6   160   110  3.9   2.88  17.0     0     1     4     4
3  22.8     4   108    93  3.85  2.32  18.6     1     1     4     1
# ... with 7 more rows
```

벡터 x가 특정 두 숫자(left, right) 사이에 있는지 확인
1) x >= left & x <= right
2) between(x, left, right)

```
> mtcars_t %>%
    filter(
      between(mpg, median(mpg), quantile(mpg, probs = 0.75))
      )
```

10

10

- 예제: airquality :뉴욕 공기상태, 1973, 5월-9월

```
> airs <- as_tibble(airquality) %>%
    print(n = 3)
# A tibble: 153 x 6
  Ozone Solar.R  Wind  Temp Month   Day
  <int>   <int> <dbl> <int> <int> <int>
1    41     190   7.4    67     5     1
2    36     118   8      72     5     2
3    12     149  12.6    74     5     3
# ... with 150 more rows
```

  - 변수 Ozone 또는 Solar.R이 결측값인 관찰값 선택

```
> airs %>%
    filter(is.na(Ozone) | is.na(Solar.R)) %>%
    print(n=3)
# A tibble: 42 x 6
  Ozone Solar.R  Wind  Temp Month   Day
  <int>   <int> <dbl> <int> <int> <int>
1    NA      NA  14.3    56     5     5
2    28      NA  14.9    66     5     6
3    NA     194   8.6    69     5    10
# ... with 39 more rows
```

11

11

- 위치에 의한 행 선택: `slice()` 및 그와 관련된 함수
  - 함수 `slice()`에 의한 행 선택
    - 행 번호를 직접 입력하여 특정 행의 선택 또는 제거
    - 선택: 양의 정수
    - 제거: 음의 정수
  - 예제: `iris` :붓꽃, 꽃받침(길이,너비), 꽃잎(길이,너비), 종류, 변수,n=150,
    - iris를 tibble로 전환하고 5~10번째 행 선택

```
> iris_t <- as_tibble(iris)
> iris_t %>% slice(5:10)                        #<fct>:factor
# A tibble: 6 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          5           3.6          1.4         0.2 setosa
2          5.4         3.9          1.7         0.4 setosa
3          4.6         3.4          1.4         0.3 setosa
4          5           3.4          1.5         0.2 setosa
5          4.4         2.9          1.4         0.2 setosa
6          4.9         3.1          1.5         0.1 setosa
```

12

12

- 5~10번째 행 제거

```
> iris_t %>% slice(-(5:10)) %>%
    print(n=3)
# A tibble: 144 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3            1.4         0.2 setosa
3          4.7         3.2          1.3         0.2 setosa
# ... with 141 more rows
```

- 마지막 행 선택

```
> iris_t %>% slice(n())
# A tibble: 1 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          5.9           3          5.1         1.8 virginica
```

함수 n()
- 데이터 프레임의 행 개수를 세는 함수
- 단독으로는 사용할 수 없음

13

13

- 함수 slice_head()와 slice_tail()에 의한 행 선택
  - 처음 몇 개 행 또는 마지막 몇 개 행 선택
  - 행의 개수(n) 또는 비율(prop) 지정

- iris의 처음 3개 행 선택

```
> iris_t %>% slice_head(n = 3)
# A tibble: 3 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3            1.4         0.2 setosa
3          4.7         3.2          1.3         0.2 setosa
```

- iris의 마지막 3개 행 선택

```
> iris_t %>% slice_tail(n = 3)
# A tibble: 3 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          6.5         3            5.2         2   virginica
2          6.2         3.4          5.4         2.3 virginica
3          5.9         3            5.1         1.8 virginica
```

14

14

- 함수 slice_sample()에 의한 행 선택
  - 단순임의추출에 의한 행 선택
  - 행의 개수(n) 또는 비율(prop) 지정
  - 비복원추출이 디폴트. replace = TRUE 지정으로 복원추출 가능

- iris에서 3개 행 임의 추출

```
> iris_t %>% slice_sample(n = 3)
# A tibble: 3 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          6           2.2          5           1.5 virginica
2          5.5         2.6          4.4         1.2 versicolor
3          5.4         3.7          1.5         0.2 setosa
```

- 전체 행 중 2% 행 복원 추출

```
> iris_t %>% slice_sample(prop = 0.02, replace = TRUE)
# A tibble: 3 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          7.3         2.9          6.3         1.8 virginica
2          5           2.3          3.3         1   versicolor
3          6.7         3            5           1.7 versicolor
```

15

15

- 함수 slice_max()와 slice_min()에 의한 행 선택
  - 특정 변수가 가장 큰 값 또는 가장 작은 값을 갖는 행 선택
  - 기준 변수와 선택하고자 하는 행의 개수(n) 또는 비율(prop) 지정

- iris에서 Sepal.Width의 값이 가장 큰 2개 행 선택

```
> iris_t %>% slice_max(Sepal.Width, n = 2)
# A tibble: 2 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          5.7         4.4          1.5         0.4 setosa
2          5.5         4.2          1.4         0.2 setosa
```

- Petal.Length의 값이 가장 작은 2개 행 선택

```
> iris_t %>% slice_min(Petal.Length, n = 2)
# A tibble: 2 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          4.6         3.6          1           0.2 setosa
2          4.3         3            1.1         0.1 setosa
```

16

16

- 행의 정렬: `arrange()`

  - 특정 변수를 기준으로 데이터 프레임의 행 재배열

  - 기본적인 사용법: `arrange(df, var_1, var_2, …)`
    - `df`: 데이터 프레임
    - `var_1`: 제1 정렬 기준 변수
    - `var_2`: 제2 정렬 기준 변수
    - 2개 이상의 정렬 기준 변수 나열: 추가된 변수는 앞선 변수가 같은 값을 갖는 행의 정렬 기준
    - 오름차순 정렬이 디폴트
    - 내림차순 정렬: 기준 변수를 함수 `desc()`에 입력 (descending 의미)

17

17

- 예제: mtcars

  - 변수 mpg의 값이 가장 좋지 않은 자동차부터 재배열

```
> mtcars_t %>% arrange(mpg) %>% print(n = 3)
# A tibble: 32 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  10.4     8   472   205  2.93  5.25  18.0     0     0     3     4
2  10.4     8   460   215  3     5.42  17.8     0     0     3     4
3  13.3     8   350   245  3.73  3.84  15.4     0     0     3     4
# ... with 29 more rows
```

  - mpg가 가장 좋지 않은 자동차부터 배열하되, mpg 값이 같은 자동차는 wt의 값이 높은 자동차부터 배열

```
> mtcars_t %>% arrange(mpg, desc(wt)) %>% print(n = 3)
# A tibble: 32 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  10.4     8   460   215  3     5.42  17.8     0     0     3     4
2  10.4     8   472   205  2.93  5.25  18.0     0     0     3     4
3  13.3     8   350   245  3.73  3.84  15.4     0     0     3     4
# ... with 29 more rows
```

18

18

- 예제: airquality
  - 5월 1일부터 5월 10일까지의 자료만을 대상으로 변수 Ozone의 값이 가장 낮았던 날부터 재배열

```
> airs_1 <- as_tibble(airquality) %>%
    filter(Month == 5, Day <= 10)
> airs_1 %>% arrange(Ozone)
# A tibble: 10 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1     8      19  20.1    61     5     9
 2    12     149  12.6    74     5     3
 3    18     313  11.5    62     5     4
 4    19      99  13.8    59     5     8
 5    23     299   8.6    65     5     7
 6    28      NA  14.9    66     5     6
 7    36     118   8      72     5     2
 8    41     190   7.4    67     5     1
 9    NA      NA  14.3    56     5     5
10    NA     194   8.6    69     5    10
```

19

19

- airs_1을 변수 Ozone이 결측값인 케이스를 가장 앞으로 배열 (나머지는 원래순서)

```
> airs_1 %>% arrange(!is.na(Ozone))
# A tibble: 10 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1    NA      NA  14.3    56     5     5
 2    NA     194   8.6    69     5    10
 3    41     190   7.4    67     5     1
 4    36     118   8      72     5     2
 5    12     149  12.6    74     5     3
 6    18     313  11.5    62     5     4
 7    28      NA  14.9    66     5     6
 8    23     299   8.6    65     5     7
 9    19      99  13.8    59     5     8
10     8      19  20.1    61     5     9
```

  - 배열 기준으로 논리형 벡터 사용
  - TRUE, FALSE 배열에서 우선 순위는 FALSE
  - !is.na(Ozone): 변수 Ozone이 결측값인 케이스가 우선 순위

20

20

- airs_1을 변수 Ozone이 가장 높은 날부터 배열하되 결측값이 있는 케이스를 가장 앞으로 배치

```
> airs_1 %>% arrange(!is.na(Ozone), desc(Ozone))
# A tibble: 10 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1    NA      NA  14.3    56     5     5
 2    NA     194   8.6    69     5    10
 3    41     190   7.4    67     5     1
 4    36     118   8      72     5     2
 5    28      NA  14.9    66     5     6
 6    23     299   8.6    65     5     7
 7    19      99  13.8    59     5     8
 8    18     313  11.5    62     5     4
 9    12     149  12.6    74     5     3
10     8      19  20.1    61     5     9
```

- NA 그룹내 : 원래 순서대로
- NA가 아닌 그룹내: Ozone 내림차순

21

21

- 중복된 행 제거: `distinct()`
  - 중복 여부를 결정할 변수 지정: 없는 경우에는 모든 변수 대상
  - 옵션 .keep_all = TRUE: 모든 변수 유지. 중복된 행 중 첫 번째 행만 유지
    = FALSE: 해당 변수만.
  - 예제

```
> df1 <- tibble(id = rep(1:3, times = 2:4), x1 = c(1:2,1:3,1:4))
> df1
# A tibble: 9 x 2
     id    x1
  <int> <int>
1     1     1
2     1     2
3     2     1
4     2     2
5     2     3
6     3     1
7     3     2
8     3     3
9     3     4
```

22

22

- 변수 id가 중복되지 않는 행 선택

```
> df1 %>% distinct(id, .keep_all = TRUE)
# A tibble: 3 x 2
     id    x1
  <int> <int>
1     1     1
2     2     1
3     3     1
```

```
> df1
     id    x1
1     1     1
2     1     2
3     2     1
4     2     2
5     2     3
6     3     1
7     3     2
8     3     3
9     3     4
```

- 변수 id가 중복된 행 중 x1의 값이 가장 큰 행 선택

```
> df1 %>% arrange(id, desc(x1)) %>%
    distinct(id, .keep_all = TRUE)
# A tibble: 3 x 2
     id    x1
  <int> <int>
1     1     2
2     2     3
3     3     4
```

```
> df1 %>% arrange(id,
desc(x1))
     id    x1
1     1     2
2     1     1
3     2     3
4     2     2
5     2     1
6     3     4
7     3     3
8     3     2
9     3     1
```

23

23

- 예제: 모든 변수의 값이 중복된 행 제거

```
> df2 <- tibble(id = rep(1:3, each = 2), x1 = c(2,2,3,1,4,4))
> df2
# A tibble: 6 x 2
     id    x1
  <int> <dbl>
1     1     2
2     1     2
3     2     3
4     2     1
5     3     4
6     3     4
```

```
> df2 %>% distinct(.keep_all = TRUE)
# A tibble: 4 x 2
     id    x1
  <int> <dbl>
1     1     2
2     2     3
3     2     1
4     3     4
```

24

24

4.2 열을 작업 대상으로 하는 함수

- 열의 선택: `select()`
  - 데이터 세트의 크기 증가: 지나치게 많은 변수를 줄이는 작업이 필요
  - 분석에 필요한 변수 선택 필요
  - 변수 선택 방법: 패키지 `tidyselect`의 방식 적용
  - <tidy-select> 방식
    1) 열 번호(또는 열 이름)에 의한 선택
    2) 변수의 유형에 의한 선택
    3) 변수 선택과 관련된 몇몇 함수에 의한 선택

1) 열 번호(또는 열 이름)에 의한 선택

- 열 이름은 열 번호와 같은 취급
- 열 번호를 콤마로 구분하여 나열
- 연속된 열은 콜론(:) 연산자를 이용하여 나열
- 나열된 열들은 차례로 합집합 구성

- 예제: mtcars
  - 행 이름을 변수 row.name으로 추가하고 tibble 전환
  - 첫 번째에서 세 번째, 그리고 일곱 번째 변수 선택

```
> mtcars_t <- mtcars %>%
    rownames_to_column(var = "row.name") %>%
    as_tibble()
```

```
> mtcars_t %>% select(1:3, 7)
# A tibble: 32 x 4
  row.name         mpg   cyl    wt
  <chr>          <dbl> <dbl> <dbl>
1 Mazda RX4       21       6  2.62
2 Mazda RX4 Wag   21       6  2.88
3 Datsun 710      22.8     4  2.32
# ... with 29 more rows
```

```
> mtcars_t %>%
  select(row.name:cyl, wt)
```

```
mtcars_t VARs:row.name mpg cyl disp hp drat wt qsec vs am gear carb
```

- 열 제거
  - 논리 부정 연산자(!): 여집합 구성

```
> mtcars_t %>% select(!c(1:3, 7))
# A tibble: 32 x 8
   disp    hp  drat  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1   160   110  3.9   16.5     0     1     4     4
2   160   110  3.9   17.0     0     1     4     4
3   108    93  3.85  18.6     1     1     4     1
# ... with 29 more rows
```

  - 마이너스 연산자(-): 차집합 구성

```
> mtcars_t %>% select(1:3, -1)
# A tibble: 32 x 2
    mpg   cyl
  <dbl> <dbl>
1    21     6
2    21     6
# ... with 30 more rows
```

- select(1:3, -1): (1, 2, 3)에서 1 제외 -> (1, 2)
- select(1:3, !1): (1, 2, 3)과 (1을 제외한 나머지)의 합집합 -> 전체

mtcars_t VARs:row.name mpg cyl disp hp drat wt qsec vs am gear carb

27

27

- 마이너스 연산자를 첫 번째로 입력: 논리 부정 연산자와 동일

```
> mtcars_t %>% select(-c(1:3, 7))
# A tibble: 32 x 8
   disp    hp  drat  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1   160   110  3.9   16.5     0     1     4     4
2   160   110  3.9   17.0     0     1     4     4
3   108    93  3.85  18.6     1     1     4     1
# ... with 29 more rows
```

mtcars_t VARs:row.name mpg cyl disp hp drat wt qsec vs am gear carb

28

28

2) 변수 유형에 의한 선택

- 함수 `where()`에 의한 선택
  - 사용법: `where(fn)`

    `fn`: 결과가 TRUE 또는 FALSE인 predicate 함수

    예) `is.numeric`, `is.character`
  - `fn`의 결과가 TRUE인 변수 선택

29

29

- 예제: `ggplot2::mpg`

```
> mpg %>% print(n=3)
# A tibble: 234 x 11
  manufacturer model displ  year   cyl trans drv     cty   hwy fl
  <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr>
1 audi         a4      1.8  1999     4 auto~ f        18    29 p
2 audi         a4      1.8  1999     4 manu~ f        21    29 p
3 audi         a4      2    2008     4 manu~ f        20    31 p
# ... with 231 more rows, and 1 more variable: class <chr>
```

- 숫자형 변수 선택

```
> mpg %>% select(where(is.numeric))
# A tibble: 234 x 5
  displ  year   cyl   cty   hwy
  <dbl> <int> <int> <int> <int>
1   1.8  1999     4    18    29
2   1.8  1999     4    21    29
3   2    2008     4    20    31
# ... with 231 more rows
```

몇 가지 유형 함께 선택 가능

```
select(where(is.numeric)|
       where(is.character))
```

30

30

3) 변수 선택과 관련된 함수

- everything(): 모든 변수 선택
- last_col(): 마지막 변수 선택

- starts_with("x"): 이름이 "x"로 시작하는 변수 선택
- ends_with("x"): 이름이 "x"로 끝나는 변수 선택
- contains("x"): 이름에 "x"가 있는 변수 선택
- num_range("x", 1:10): x1, x2, ..., x10과 동일

- all_of(vec): 벡터 vec에 이름 있는 변수 선택. 단, 데이터 프레임에 없는 변수 이름이 vec에 있으면 오류 발생
- any_of(vec): all_of(vec)과 동일. 단, 데이터 프레임에 없는 이름이 vec에 있어도 오류 발생하지 않음

31

31

- mtcars_t에서 첫 번째와 마지막 변수 선택

```
> mtcars_t %>% select(1, last_col())
# A tibble: 32 x 2
  row.name       carb
  <chr>         <dbl>
1 Mazda RX4         4
2 Mazda RX4 Wag     4
3 Datsun 710        1
# ... with 29 more rows
```

- mtcars_t에서 이름이 "m"으로 시작하는 변수 선택

```
> mtcars_t %>%
select(starts_with("m"))
# A tibble: 32 x 1
    mpg
  <dbl>
1  21
2  21
3  22.8
# ... with 29 more rows
```

```
mtcars_t VARs:row.name mpg cyl disp hp drat wt qsec vs am gear carb
```

32

32

- 이름이 "p"로 끝나는 변수

```
> mtcars_t %>% select(ends_with("p"))
# A tibble: 32 x 2
   disp    hp
  <dbl> <dbl>
1   160   110
2   160   110
3   108    93
# ... with 29 more rows
```

- 이름에 "A"가 있는 변수 선택

```
> mtcars_t %>% select(contains("A"))
# A tibble: 32 x 5
  row.name       drat    am  gear  carb
  <chr>         <dbl> <dbl> <dbl> <dbl>
1 Mazda RX4      3.9      1     4     4
2 Mazda RX4 Wag  3.9      1     4     4
3 Datsun 710     3.85     1     4     1
# ... with 29 more rows
```

- 소문자와 대문자 구분 없음
- 옵션 ignore.case = TRUE가 디폴트

```
> mtcars_t %>% select(contains("A", ignore.case=FALSE))
# A tibble: 32 x 0
```

```
mtcars_t VARs:row.name mpg cyl disp hp drat wt qsec vs am gear carb
```

33

33

- 벡터에 이름이 있는 변수 선택

```
> vars <- c("model", "mpg", "wt")
> mtcars_t %>% select(any_of(vars))
# A tibble: 32 x 2
    mpg    wt
  <dbl> <dbl>
1  21    2.62
2  21    2.88
3  22.8  2.32
# ... with 29 more rows
```

```
> mtcars_t %>% select(all_of(vars))
에러: Can't subset columns that don't exist.
x Column `model` doesn't exist.
```

```
mtcars_t VARs:row.name mpg cyl disp hp drat wt qsec vs am gear carb
```

34

34

- 숫자형 변수 중 이름에 "c"가 있는 변수 선택

```
> mtcars_t %>% select(where(is.numeric) & contains("c"))
# A tibble: 32 x 3
    cyl  qsec  carb
  <dbl> <dbl> <dbl>
1     6  16.5     4
2     6  17.0     4
3     4  18.6     1
# ... with 29 more rows
```

```
mtcars_t VARs:row.name mpg cyl disp hp drat wt qsec vs am gear carb
```

35

35

- 열 이름 변경: `select()`, `rename()`, `rename_with()`
  - `select()`에 의한 열 이름 변경
    - new_name = old_name 방식
    - 변경되지 않은 열 삭제 (selcect는 기본적으로 변수 선택 함수)

```
> mtcars_t <- mtcars %>%
    rownames_to_column(var = "row.name") %>%
    as_tibble()
```

- 변수 row.name을 model로 이름 변경

```
> mtcars_t %>% select(model = row.name)
# A tibble: 32 x 1
  model
  <chr>
1 Mazda RX4
2 Mazda RX4 Wag
3 Datsun 710
# ... with 29 more rows
```

- 모든 열 유지

```
> mtcars_t %>% select(model = row.name, everything())
```

36

36

- rename()에 의한 변경
  - new_name = old_name 방식
  - 모든 열 유지

```
> mtcars_t %>% rename(model = row.name)
# A tibble: 32 x 12
  model      mpg   cyl  disp    hp  drat    wt  qsec    vs    am
  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Mazda R~  21       6   160   110  3.9   2.62  16.5     0     1
2 Mazda R~  21       6   160   110  3.9   2.88  17.0     0     1
3 Datsun ~  22.8     4   108    93  3.85  2.32  18.6     1     1
# ... with 29 more rows, and 2 more variables: gear <dbl>,
#   carb <dbl>
```

```
mtcars_t VARs:row.name mpg cyl disp hp drat wt qsec vs am gear carb
```

37

37

- rename_with()에 의한 변경
  - 많은 변수 이름을 공통된 양식으로 모두 변경해야 하는 경우

    예) 대문자 이름을 모두 소문자로 변경
  - 사용법: rename_with(fn)

    fn: 변수 이름을 변경하는 함수

- mtcars_t의 모든 변수 이름 대문자로 변경

```
> mtcars_t %>% rename_with(toupper)
# A tibble: 32 x 12
  ROW.NAME   MPG   CYL  DISP    HP  DRAT    WT  QSEC    VS    AM
  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Mazda R~  21       6   160   110  3.9   2.62  16.5     0     1
2 Mazda R~  21       6   160   110  3.9   2.88  17.0     0     1
3 Datsun ~  22.8     4   108    93  3.85  2.32  18.6     1     1
# ... with 29 more rows, and 2 more variables: GEAR <dbl>,
#   CARB <dbl>
```

모든 변수가 변경 대상이 되는 것이 디폴트

38

38

- 이름에 "a"가 포함된 변수 이름 대문자로 변경
- rename_with()의 대상은 모든 변수가 디폴트
- 함수의 입력요소로 변경대상

```
> mtcars_t %>% rename_with(toupper, contains("a"))
# A tibble: 32 x 12
  ROW.NAME   mpg   cyl  disp    hp  DRAT    wt  qsec    vs    AM
  <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Mazda R~  21       6   160   110  3.9   2.62  16.5     0     1
2 Mazda R~  21       6   160   110  3.9   2.88  17.0     0     1
3 Datsun ~  22.8     4   108    93  3.85  2.32  18.6     1     1
# ... with 29 more rows, and 2 more variables: GEAR <dbl>,
#   CARB <dbl>
```

39

39

- 열의 위치 변경: `relocate()`
  - 여러 개의 열 위치 변경 가능
  - <tidy-select> 방식으로 열 선택
  - 이동 위치: 제일 앞(디폴트), 옵션 .after 혹은 .before에서 지정 가능

- iris의 Species를 첫 번째 변수로 이동

```
> iris_t <- as_tibble(iris)
> iris_t %>% relocate(Species)
# A tibble: 150 x 5
  Species Sepal.Length Sepal.Width Petal.Length Petal.Width
  <fct>          <dbl>       <dbl>        <dbl>       <dbl>
1 setosa           5.1         3.5          1.4         0.2
2 setosa           4.9         3            1.4         0.2
3 setosa           4.7         3.2          1.3         0.2
# ... with 147 more rows
```

```
> iris_t %>% relocate(ends_with("th"), .after = Species)
```

```
> iris_t %>% relocate(ends_with("th"), .after = last_col())
```

```
iris_t VARs:Sepal.Length Sepal.Width Petal.Length Petal.Width Species
```

40

40

- 새로운 열의 추가: `mutate()`와 `transmute()`
  - 기존의 변수를 이용하여 새로운 변수 생성
  - `mutate()` : 대상 데이터프레임 입력후 새로운 변수 표현식들 입력
    생성된 변수는 데이터 프레임에 마지막 변수로 추가
  - `transmute()` : 생성된 변수만 유지

- 예제: mtcars
  - 다음 조건으로 변수 `kml`과 `gp_kml`을 만들고, 데이터 프레임의 첫 번째와 두 번째 변수로 추가
    1) `kml`: 1 `mpg`는 0.43 `kml`
       `gp_kml`: `kml`이 10 이상이면 'good', 10 미만이면 'bad'
    2) `kml`: 1 `mpg`는 0.43 `kml`
       `gp_kml`: `kml`이 11 이상이면 'excellent', 11 미만 8 이상이면 'good', 8 미만이면 'bad'

41

41

- 연속형 변수를 기반으로 두 개의 범주를 갖는 범주형 변수 생성

  `if_else(condition, true, false)`

  `condition`이 만족되면 `true`의 값, 아니면 `false`의 값

```
> as_tibble(mtcars) %>%
    mutate(kml = 0.43*mpg,
           gp_kml = if_else(kml >= 10, "good", "bad")
           ) %>%
    relocate(kml, gp_kml)
# A tibble: 32 x 13
    kml gp_kml   mpg   cyl  disp    hp  drat    wt  qsec    vs
  <dbl> <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  9.03 bad     21       6   160   110  3.9   2.62  16.5     0
2  9.03 bad     21       6   160   110  3.9   2.88  17.0     0
3  9.80 bad     22.8     4   108    93  3.85  2.32  18.6     1
# ... with 29 more rows, and 3 more variables: am <dbl>,
#   gear <dbl>, carb <dbl>
```

42

42

- 예제: mtcars

2) gp_kml: kml이 11 이상이면 'excellent',
11 미만 8 이상이면 'good',
8 미만이면 'bad'

- 연속형 변수를 기반으로 3개 이상의 범주를 갖는 범주형 변수 생성

```
case_when(
       condition_1 ~ value_1,
       condition_2 ~ value_2,
        TRUE       ~ value_3
         )
```

condition_1이 TRUE이면 value_1,
condition_1이 FALSE이고 condition_2가 TRUE이면 value_2,
두 조건 모두 FALSE이면 value_3

- condition은 순서대로 평가되므로 범위를 넓혀가는 조건이 잇따라 제시되어야 함
- 제시되는 조건의 개수에는 제한이 없음
- value_1, value_2, ... 의 값은 동일한 유형

43

43

```
> as_tibble(mtcars) %>%
    mutate(kml = 0.43*mpg,
           gp_kml = case_when(
             kml < 8   ~ "bad",
             kml < 11  ~ "good",
             TRUE      ~ "excellent"
             )
           ) %>%
    relocate(kml, gp_kml)
# A tibble: 32 x 13
    kml gp_kml   mpg   cyl  disp    hp  drat    wt  qsec    vs
  <dbl> <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  9.03 good    21       6   160   110  3.9   2.62  16.5     0
2  9.03 good    21       6   160   110  3.9   2.88  17.0     0
3  9.80 good    22.8     4   108    93  3.85  2.32  18.6     1
# ... with 29 more rows, and 3 more variables: am <dbl>,
#   gear <dbl>, carb <dbl>
```

```
gp_kml = case_when(
             kml >= 11 ~ "excellent",
             kml >= 8  ~ "good",
             TRUE      ~ "bad"
             )
```

동일한 결과

44

44

4.3 여러 행 자료의 요약: summarise()

- 변수들의 요약 통계량 계산할 때 유용하게 사용되는 함수
- name = fun의 형태로 열 이름과 요약 통계량을 계산하는 함수 연결

- 예제: mpg
  - 변수 hwy의 전체 케이스의 개수, 서로 다른 값을 갖고 있는 케이스의 개수, 평균값 계산

```
> mpg %>%
    summarise(n = n(), n_hwy = n_distinct(hwy),
              avg_hwy = mean(hwy))
# A tibble: 1 x 3
      n n_hwy avg_hwy
  <int> <int>   <dbl>
1   234    27    23.4
```

  - n(): 전체 행의 개수
  - n_distinct(): 서로 다른 값을 갖고 있는 행의 개수

45

45

- 길이가 2 이상이 되는 벡터를 결과로 출력하는 요약 통계량 함수의 사용

```
> mpg %>%
    summarise(avg_hwy = mean(hwy), rng_hwy = range(hwy))
# A tibble: 2 x 2
  avg_hwy rng_hwy
    <dbl>   <int>
1    23.4      12
2    23.4      44
```

길이가 1인 요약 통계량의 경우에만, 결과를 반복시켜 길이를 맞춰서 데이터 프레임 구성

- 다음 시도에 어떤 문제가 있는가?

```
> mpg %>%
    summarise(avg_hwy = mean(hwy), rng_hwy = range(hwy),
              q_hwy = quantile(hwy, probs = c(0.25,0.5,0.75)))
에러: Problem with `summarise()` input `q_hwy`.
x Input `q_hwy` must be size 2 or 1, not 3.
i Input `q_hwy` is `quantile(hwy, probs = c(0.25, 0.5, 0.75))`.
i An earlier column had size 2.
```

46

46

4.4 그룹 데이터 프레임

- 그룹 데이터 프레임의 생성: `group_by()`
  - 한 개 이상의 변수를 이용하여 전체 데이터 프레임을 그룹으로 구분
  - 실행 결과는 tibble
  - 출력된 형태에는 큰 차이가 없으나, grouped_df라는 속성 추가

```
> by_cyl <- mpg %>% group_by(cyl)
> by_cyl
# A tibble: 234 x 11
# Groups:   cyl [4]
  manufacturer model displ  year   cyl trans drv     cty   hwy
  <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int>
1 audi         a4      1.8  1999     4 auto~ f        18    29
2 audi         a4      1.8  1999     4 manu~ f        21    29
3 audi         a4      2    2008     4 manu~ f        20    31
# ... with 231 more rows, and 2 more variables: fl <chr>,
#   class <chr>
```

'Groups: cyl [4]' : cyl에 의하여 4개 그룹이 구성된 것을 의미

47

47

- 각 그룹에 속한 자료 개수 확인

그룹 데이터 프레임 대상

```
> by_cyl %>% tally()
# A tibble: 4 x 2
    cyl     n
  <int> <int>
1     4    81
2     5     4
3     6    79
4     8    70
```

일반 데이터 프레임 대상

```
> mpg %>% count(cyl)
# A tibble: 4 x 2
    cyl     n
  <int> <int>
1     4    81
2     5     4
3     6    79
4     8    70
```

- 그룹 변수 추가

```
> by_cyl %>%  group_by(drv, .add = TRUE)
```

cyl과 drv로 그룹 구성

- 그룹 변수 변경

```
> by_cyl %>%  group_by(drv)
```

drv로 그룹 구성

- 그룹 해제

```
> by_cyl %>% ungroup()
```

48

48

```
> by_cyl <- mpg %>% group_by(cyl)

> by_cyl %>% tally()
# A tibble: 4 x 2
    cyl     n
  <int> <int>
1     4    81
2     5     4
3     6    79
4     8    70
```

```
> by_cyl %>%
group_by(drv, .add = TRUE)
%>% tally()
# A tibble: 9 x 3
# Groups:   cyl [4]
    cyl drv       n
  <int> <chr> <int>
1     4 4        23
2     4 f        58
3     5 f         4
4     6 4        32
5     6 f        43
6     6 r         4
7     8 4        48
8     8 f         1
9     8 r        21
```

```
> by_cyl %>% group_by(drv)%>% tally()
# A tibble: 3 x 2
  drv       n
  <chr> <int>
1 4       103
2 f       106
3 r        25
```

```
> by_cyl %>%
ungroup() %>% tally()
# A tibble: 1 x 1
      n
  <int>
1   234
```


- 그룹 데이터 프레임에서 기본 dplyr 함수들의 작동 방식

  - 함수 summarise()
    - 각 그룹별 요약 통계량 계산

  - 예제 :
    1) 월별 변수 Ozone의 평균값
    2) 월별 Ozone의 결측값 날수와 관측된 날수
    3) 월별 첫날과 마지막 날의 Ozone 값
    4) 월별 Ozone의 가장 작은 값과 가장 큰 값
    5) 월별 Ozone의 개별 값이 전체 기간 동안의 평균값보다 작은 날 수

  다섯 문제에 공통적으로 사용될 그룹 데이터 프레임 생성

```
> airs_M <- airquality %>% group_by(Month)
```

1) 월별 변수 Ozone의 평균값

```
> airs_M %>%
    summarise(avg_Oz = mean(Ozone, na.rm = TRUE))
# A tibble: 5 x 2
  Month avg_Oz
  <int>  <dbl>
1     5   23.6
2     6   29.4
3     7   59.1
4     8   60.0
5     9   31.4
```

51

51

2) 월별 Ozone의 결측값 날수와 관측된 날수

```
> airs_M %>%
    summarise(n_miss = sum(is.na(Ozone)),
              n_obs = sum(!is.na(Ozone)))
# A tibble: 5 x 3
  Month n_miss n_obs
  <int>  <int> <int>
1     5      5    26
2     6     21     9
3     7      5    26
4     8      5    26
5     9      1    29
```

52

52

3) 월별 첫날과 마지막 날의 Ozone 값

벡터 인덱싱 함수: `first()`, `last()`, `nth()`

- `first(x)`: `x[1]`과 동일
- `last(x)`: `x[length(x)]`와 동일
- `nth(x, 2)`: `x[2]`와 동일
- `nth(x, -2)`: `x[length(x)-1]`과 동일

```
> airs_M %>%
    summarise(first_Oz = first(Ozone),
              last_Oz = last(Ozone))
# A tibble: 5 x 3
  Month first_Oz last_Oz
  <int>    <int>   <int>
1     5       41      37
2     6       NA      NA
3     7      135      59
4     8       39      85
5     9       96      20
```

53

53

4) 월별 Ozone의 가장 작은 값과 가장 큰 값

```
> airs_M %>%
    summarise(max_Oz = max(Ozone, na.rm = TRUE),
              min_Oz = min(Ozone, na.rm = TRUE))
# A tibble: 5 x 3
  Month max_Oz min_Oz
  <int>  <int>  <int>
1     5    115      1
2     6     71     12
3     7    135      7
4     8    168      9
5     9     96      7
```

54

54

5) 월별 Ozone의 개별 값이 전체 기간 동안의 평균값보다 작은 날 수

```
> m_Oz <- mean(airquality$Ozone, na.rm = TRUE)
> airs_M %>%
    summarise(low_Oz = sum(Ozone < m_Oz, na.rm = TRUE))
# A tibble: 5 x 2
  Month low_Oz
  <int>  <int>
1     5     24
2     6      8
3     7      8
4     8     10
5     9     22
```

55

55

- 함수 select()
  - 그룹을 구성하는 변수는 선택 대상이 아니어도 항상 포함

```
> mpg %>% group_by(cyl) %>%
    select(hwy)
# A tibble: 234 x 2
# Groups:   cyl [4]
    cyl   hwy
  <int> <int>
1     4    29
2     4    29
3     4    31
# ... with 231 more rows
```

56

56

- 함수 arrange()
  - 옵션 .by_group=TRUE가 추가되면 그룹 변수가 첫 번째 정렬변수

```
> mpg %>% group_by(cyl) %>%
    arrange(hwy, .by_group = TRUE)
# A tibble: 234 x 11
# Groups:   cyl [4]
   manufacturer model displ  year   cyl trans drv     cty   hwy
   <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int>
 1 toyota       4run~   2.7  1999     4 manu~ 4        15    20
 2 toyota       4run~   2.7  1999     4 auto~ 4        16    20
 3 toyota       toyo~   2.7  1999     4 manu~ 4        15    20
 4 toyota       toyo~   2.7  1999     4 auto~ 4        16    20
 5 toyota       toyo~   2.7  2008     4 manu~ 4        17    22
 6 subaru       fore~   2.5  2008     4 auto~ 4        18    23
 7 dodge        cara~   2.4  1999     4 auto~ f        18    24
 8 subaru       fore~   2.5  1999     4 auto~ 4        18    24
 9 audi         a4 q~   1.8  1999     4 auto~ 4        16    25
10 subaru       fore~   2.5  1999     4 manu~ 4        18    25
# ... with 224 more rows, and 2 more variables: fl <chr>,
#   class <chr>
```

57

57

- 함수 mutate()와 transmute()
  - 요약 통계량이나 순위 계산 함수 사용하여 새 변수 생성 시 다른 결과

```
> mpg %>% select(cyl, hwy) %>%
    mutate(std_hwy = hwy - mean(hwy), rank = min_rank(hwy))
# A tibble: 234 x 4
    cyl   hwy std_hwy  rank
  <int> <int>   <dbl> <int>
1     4    29    5.56   187
2     4    29    5.56   187
# ... with 232 more rows
```

```
> mpg %>% select(cyl, hwy) %>%
    group_by(cyl) %>%
    mutate(std_hwy = hwy - mean(hwy), rank = min_rank(hwy))
# A tibble: 234 x 4
# Groups:   cyl [4]
    cyl   hwy std_hwy  rank
  <int> <int>   <dbl> <int>
1     4    29   0.198    38
2     4    29   0.198    38
# ... with 232 more rows
```

min_rank( ): 입력된 벡터의 (오름차순) 순위 계산. 크기가 같은 자료에는 해당되는 순위 중 가장 작은 순위를 모두에게 부여

58

58

- 함수 `filter()`
  - 요약 통계량을 사용한 조건 설정: 그룹별로 다른 조건에 의한 행 선택

mpg의 cyl 그룹별 hwy가 가장 큰 값을 갖는 행 선택

```
> mpg %>% group_by(cyl) %>%
    select(1:2, hwy) %>%
    filter(hwy == max(hwy)) %>%
    arrange(hwy, .by_group = TRUE)
# A tibble: 8 x 4
# Groups:   cyl [4]
    cyl manufacturer model        hwy
  <int> <chr>        <chr>      <int>
1     4 volkswagen   jetta         44
2     4 volkswagen   new beetle    44
3     5 volkswagen   jetta         29
4     5 volkswagen   jetta         29
5     5 volkswagen   new beetle    29
6     6 chevrolet    malibu        29
7     8 chevrolet    corvette      26
8     8 chevrolet    corvette      26
```

59

59

- 함수 `slice()`
  - 그룹별로 각각의 행 선택 가능

airquality에서 Ozone의 월별 첫날과 마지막 날, 가장 큰 값과 가장 작은 값을 갖는 행 선택

```
> airs_M <- airquality %>%
    group_by(Month) %>%
    select(1, 5, 6)
```

매달 첫날의 Ozone 값

```
> airs_M %>% slice_head(n = 1)
# A tibble: 5 x 3
# Groups:   Month [5]
  Ozone Month   Day
  <int> <int> <int>
1    41     5     1
2    NA     6     1
3   135     7     1
4    39     8     1
5    96     9     1
```

매달 마지막 날 Ozone 값

```
> airs_M %>% slice_tail(n = 1)
# A tibble: 5 x 3
# Groups:   Month [5]
  Ozone Month   Day
  <int> <int> <int>
1    37     5    31
2    NA     6    30
3    59     7    31
4    85     8    31
5    20     9    30
```

60

60

월별 가장 작은 Ozone 값

```
> airs_M %>% slice_min(Ozone, n = 1)
# A tibble: 6 x 3
# Groups:   Month [5]
  Ozone Month   Day
  <int> <int> <int>
1     1     5    21
2    12     6    19
3     7     7    15
4     9     8     2
5     9     8    22
6     7     9    24
```

월별 가장 큰 Ozone 값

```
> airs_M %>% slice_max(Ozone, n = 1)
# A tibble: 5 x 3
# Groups:   Month [5]
  Ozone Month   Day
  <int> <int> <int>
1   115     5    30
2    71     6     9
3   135     7     1
4   168     8    25
5    96     9     1
```

61

61

- 함수 `n()`과 `cur_data()`
  - 단독으로는 사용할 수 없고 `summarise()`나 `mutate()`와 함께 사용
  - `n()`: 각 그룹에 속한 행의 개수 계산
  - `cur_data()`: 각 그룹별 데이터를 따로 불러옴

- 예제 (skip)

- mpg에서 cyl 그룹별로 hwy를 반응변수, displ을 설명변수로 하는 회귀모형 적합하고 결정계수 계산
- 그룹별 케이스가 5를 초과하지 않는 그룹 자료는 제외

- 회귀모형 적합

  `lm(y ~ x , data=df)` 데이터 프레임df의 변수 y와 x의 회귀모형 적합
- 결정계수 계산

  `summary(lm 객체)$r.squared`

62

62

```
> mpg %>%
    group_by(cyl) %>%
    filter(n() > 5) %>%
    summarise(r2 = summary(lm(hwy ~ displ,
                              data = cur_data()))$r.squared)
# A tibble: 3 x 2
    cyl     r2
  <int>  <dbl>
1     4 0.335
2     6 0.259
3     8 0.0548
```

63

63

4.5 다수의 열을 대상으로 하는 작업: across()

- 여러 열을 대상으로 동일한 작업의 반복: 모든 숫자형 변수의 평균값 계산
  - 기존의 방법: 숫자형 변수 하나하나에 mean() 적용

    매우 번거로운 작업 방식

```
> mtcars %>%
    summarise(mpg=mean(mpg), cyl=mean(cyl),
              disp=mean(disp), hp=mean(hp))
       mpg    cyl     disp       hp
1 20.09062 6.1875 230.7219 146.6875
```

  - 새로운 방법: 함수 across()의 활용

    dplyr 기본 함수 안에서 사용

```
> mtcars %>%
    summarise(across(mpg:hp, mean))
       mpg    cyl     disp       hp
1 20.09062 6.1875 230.7219 146.6875
```

64

64

- 함수 `across()` 사용법

  `across(.cols=everything(), .fns=NULL, .names=NULL)`

  `.cols`: <tidy-select> 방식으로 작업 대상 열 선택

  `.fns`: 선택된 각각의 열에 적용되는 함수 지정

  1) 하나의 함수: 예) mean

  2) `purrr` 방식: 예) ~ mean(.x, na.rm=TRUE)

  3) 여러 함수의 리스트: 예) list(m=mean, miss=~sum(is.na(.x)))

  `.names`: 결과물로 생성되는 열 이름 작성.

  - 디폴트(.fns에 하나의 함수 지정): "{col}"
  - 디폴트(.fns에 여러 함수의 리스트 지정): "{col}_{fn}"

65

65

- 함수 `summarise()`와 함께 사용

  - iris에서 모든 숫자형 변수의 평균값 계산

```
> iris_t <- as_tibble(iris)
> iris_t %>%
    summarise(across(where(is.numeric), mean))
# A tibble: 1 x 4
  Sepal.Length Sepal.Width Petal.Length Petal.Width
         <dbl>       <dbl>        <dbl>       <dbl>
1         5.84        3.06         3.76        1.20
```

  - iris에서 숫자형 변수는 평균, 요인은 수준의 개수

```
> iris_t %>%
    summarise(across(where(is.numeric), mean),
              across(where(is.factor), nlevels)
    )
# A tibble: 1 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl>   <int>
1         5.84        3.06         3.76        1.20       3
```

66

66

- 전체 행의 개수와 숫자형 변수의 표준편차

```
> iris_t %>%
    summarise(n=n(), across(where(is.numeric), sd))
# A tibble: 1 x 5
      n Sepal.Length Sepal.Width Petal.Length Petal.Width
  <dbl>        <dbl>       <dbl>        <dbl>       <dbl>
1    NA        0.828       0.436         1.77       0.762
```

- 열 n이 150이 아닌 NA가 나온 이유:
  n이 숫자이어서 across( )에 의해 sd 계산에 적용됨

대안: n의 계산을 across( ) 다음에 배치

```
> iris_t %>%
    summarise(across(where(is.numeric), sd), n=n())
# A tibble: 1 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width     n
         <dbl>       <dbl>        <dbl>       <dbl> <int>
1        0.828       0.436         1.77       0.762   150
```

67

67

- iris에서 "Se"로 시작하는 변수의 평균과 표준편차

```
> iris_t %>%
    summarise(across(starts_with("Se"), list(M = mean, SD = sd)))
# A tibble: 1 x 4
  Sepal.Length_M Sepal.Length_SD Sepal.Width_M Sepal.Width_SD
           <dbl>           <dbl>         <dbl>          <dbl>
1           5.84           0.828          3.06          0.436
```

열 이름 형식 변경

```
> iris_t %>%
    summarise(across(starts_with("Se"), list(M = mean, SD = sd),
                     .names="{fn}_{col}"))
# A tibble: 1 x 4
  M_Sepal.Length SD_Sepal.Length M_Sepal.Width SD_Sepal.Width
           <dbl>           <dbl>         <dbl>          <dbl>
1           5.84           0.828          3.06          0.436
```

68

68

- iris에서 각 숫자형 변수의 결측값 개수

```
> iris_t %>%
    summarise(across(where(is.numeric), ~ sum(is.na(.x))))
# A tibble: 1 x 4
  Sepal.Length Sepal.Width Petal.Length Petal.Width
         <int>       <int>        <int>       <int>
1            0           0            0           0
```

- Species별로 각 숫자형 변수에서 중복되지 않는 자료 개수

```
> iris %>%
    group_by(Species) %>%
    summarise(across(where(is.numeric), ~ length(unique(.x))))
# A tibble: 3 x 5
  Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
  <fct>             <int>       <int>        <int>       <int>
1 setosa               15          16            9           6
2 versicolor           21          14           19           9
3 virginica            21          13           20          12
```

```
> aa=c(1,1,2,2,2,3)
> unique(aa)
[1] 1 2 3
```

69

69

- airquality에서 Ozone과 Solar.R의 평균과 표준편차

```
> mean_sd <- list(
    mean = ~ mean(.x, na.rm=TRUE),
    sd = ~ sd(.x, na.rm=TRUE)
  )

> as_tibble(airquality) %>%
    summarise(across(c(Ozone, Solar.R), mean_sd))
# A tibble: 1 x 4
  Ozone_mean Ozone_sd Solar.R_mean Solar.R_sd
       <dbl>    <dbl>        <dbl>      <dbl>
1       42.1     33.0         186.       90.1
```

70

70

- 다른 기본 함수와 사용: `mutate()`
  - iris에서 요인을 문자형 변수로 전환

```
> iris_t <- as_tibble(iris)
> iris_t %>%
    mutate(across(where(is.factor), ~ as.character(.x)))
# A tibble: 150 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <chr>
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3            1.4         0.2 setosa
3          4.7         3.2          1.3         0.2 setosa
# ... with 147 more rows
```

  - 센티미터 단위로 측정된 자료를 인치 단위로 변환

```
> iris_t %>%
    mutate(across(where(is.numeric), ~ .x/2.54))
# A tibble: 150 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1         2.01        1.38        0.551      0.0787 setosa
2         1.93        1.18        0.551      0.0787 setosa
# ... with 148 more rows
```

71

71

- 다른 기본 함수와 사용: `filter()`
  - iris에서 이름에 'Len'이 있는 변수의 자료가 모두 6.5 이상이 되는 행 선택

```
> iris_t %>%
    filter(across(contains("Len"), ~ .x >= 6.5))
# A tibble: 4 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          7.6         3            6.6         2.1 virginica
2          7.7         3.8          6.7         2.2 virginica
# ... with 2 more rows
```

  - airquality에서 적어도 하나의 결측값이 있는 행 제거

```
> airquality %>% as_tibble() %>%
    filter(across(.fns = ~ !is.na(.x)))
# A tibble: 111 x 6
  Ozone Solar.R  Wind  Temp Month   Day
  <int>   <int> <dbl> <int> <int> <int>
1    41     190   7.4    67     5     1
2    36     118   8      72     5     2
3    12     149  12.6    74     5     3
# ... with 108 more rows
```

- 첫 번째 요소(.cols) 생략
- 두 번째 요소 .fns를 반드시 표시해야 함

함수 na.omit( )으로도 동일한 결과

72

72

- 다른 기본 함수와 사용: `distinct()`, `count()`
  - mpg에서 처음 두 변수 manufacturer와 model의 자료가 중복되지 않는 행 선택

```
> mpg %>% distinct(across(1:2), .keep_all=TRUE)
# A tibble: 38 x 11
  manufacturer model displ  year   cyl trans drv     cty   hwy
  <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int>
1 audi         a4      1.8  1999     4 auto~ f        18    29
2 audi         a4 q~   1.8  1999     4 manu~ 4        18    26
3 audi         a6 q~   2.8  1999     6 auto~ 4        15    24
# ... with 35 more rows, and 2 more variables: fl <chr>,
#   class <chr>
```

  - manufacturer와 model으로 구성되는 그룹에 속한 행 개수 계산하여 내림차순으로 정렬

```
> mpg %>% count(across(1:2), sort=TRUE)
# A tibble: 38 x 3
  manufacturer model                  n
  <chr>        <chr>              <int>
1 dodge        caravan 2wd           11
2 dodge        ram 1500 pickup 4wd   10
3 dodge        dakota pickup 4wd      9
# ... with 35 more rows
```

73

73

4.6 행 단위 작업: `rowwise()`

- dplyr은 열 단위 작업에 특화되어 있음
  - 주된 분석 대상은 열(변수)
- 행 단위 작업: 데이터 프레임의 각 행 자료를 대상으로 이루어지는 작업
  - 루프 연산으로 실행 가능하지만 매우 비효율적
  - 함수 `rowwise()`: 행 단위로 그룹 구성

```
> df1 <- tibble(x = 1:2, y = 3:4, z = 5:6)
> df1 %>% rowwise()
# A tibble: 2 x 3
# Rowwise:
      x     y     z
  <int> <int> <int>
1     1     3     5
2     2     4     6
```

  - rowwise_df라는 속성이 추가됨
  - 각 행 단위로 그룹이 구성된 것임
  - `ungroup()`: 구성된 그룹 해체

74

74

- 행 단위로 df1의 세 변수 x, y, z의 합 계산

```
> df1 %>%
    rowwise() %>%
    mutate(total = sum(c(x, y, z)))
# A tibble: 2 x 4
# Rowwise:
      x     y     z total
  <int> <int> <int> <int>
1     1     3     5     9
2     2     4     6    12
```

rowwise()의 실행이 없으면 어떤 결과가 나오는가?

75

75

- 함수 c_across()
  - 행 단위 연산에서도 많은 열을 대상으로 작업이 이루어짐
  - rowwise()로 생성된 데이터 프레임을 대상으로 <tidy-select> 방식으로 열을 선택하기 위한 함수
  - 사용법: c_across(cols=everything())
    - cols에 <tidy-select> 방식으로 열 지정

```
> df2 <- tibble(id = 1:3, w = 10:12, x = 20:22,
                y = 30:32, z = 40:42)
> df2 %>%
    rowwise() %>%
    summarise(total = sum(c_across(where(is.numeric))))
# A tibble: 3 x 1
  total
  <int>
1   101
2   106
3   111
```

76

76

- 함수 rowwise() 안에 변수 지정
  - <tidy-select> 방식으로 변수 지정 가능
  - 지정된 변수는 각 행의 ID 역할 수행: 연산 과정에서 제외됨

```
> df2 %>%
    rowwise(id) %>%
    mutate(total = sum(c_across(where(is.numeric))))
# A tibble: 3 x 6
# Rowwise:  id
     id     w     x     y     z total
  <int> <int> <int> <int> <int> <int>
1     1    10    20    30    40   100
2     2    11    21    31    41   104
3     3    12    22    32    42   108
```

77

77

- 데이터 프레임 df2의 각 열 자료를 행 단위 합 total에 대한 비율로 변환

```
> df2 %>%
    rowwise(id) %>%
    mutate(total = sum(c_across(where(is.numeric)))) %>%
    ungroup() %>%
    mutate(across(w:z, ~ .x/total))
# A tibble: 3 x 6
     id     w     x     y     z total
  <int> <dbl> <dbl> <dbl> <dbl> <int>
1     1 0.1   0.2   0.3   0.4     100
2     2 0.106 0.202 0.298 0.394   104
3     3 0.111 0.204 0.296 0.389   108
```

ungroup()을 제외하고 실행해도 동일한 결과를 얻을 수 있으나
rowwise_df 속성이 그대로 유지됨

78

78