R tidyverse 마스터 클래스

2강 - `dplyr` 패키지 중급 기술

슬기로운통계생활

Issac Lee

dplyr의 고급 동사들

2개의 테이블을 연결하는 방법

전설의 VLOOKUP을 아시나요?

tab_a

id	score
1	35
3	70
4	63
5	80

tab_b

id	name
1	issac
2	jelly
3	soony
4	bomi

테이블 합치기

`id` 정보를 기준으로 테이블 2개 합치기

- `left_join(x, y)`: 테이블을 **왼쪽으로** 합쳐줘. $x \leftarrow y$

```
tab_a %>% left_join(tab_b) %>% p
```

```
## Joining, by = "id"
```

옵션들

- `by`: 합칠때의 기준열
  - `by = c("col1" = "col2")`

```
## # A tibble: 4 x 3
##      id score name
##   <int> <int> <chr>
##       1    35 issac
##       3    70 soony
##       4    63 bomi
##       5    80 <NA>
```

테이블 합치는 방법 종류

합치는 방법도 가지가지

Mutating joins

을 가지고 의 테이블을 만들때

- `left_join()`: 가장 중요!
- `right_join()`
- `inner_join()`: 유용함!
- `full_join()`: 유용함!

Filtering joins

을 가지고 을 필터링을 할 때

- `semi_join()`: 조건 선택
- `anti_join()`: 조건 제거

Mutating join

두 테이블을 하나의 테이블로 합체

left_join()

왼쪽으로 합쳐서 테이블 생성

tab_a

id	score
1	35
3	70
4	63
5	80

tab_b

id	name
1	issac
2	jelly
3	soony
4	bomi

```
left_join(tab_a, tab_b) %>% prir
```

```
## Joining, by = "id"
```

```
## # A tibble: 4 x 3
##      id score name
##   <int> <int> <chr>
##       1    35 issac
##       3    70 soony
##       4    63 bomi
##       5    80 <NA>
```

right_join()

오른쪽으로 합쳐서 테이블 생성

tab_a

id	score
1	35
3	70
4	63
5	80

tab_b

id	name
1	issac
2	jelly
3	soony
4	bomi

```
right_join(tab_a, tab_b) %>%
  print()
```

```
## Joining, by = "id"
```

```
## # A tibble: 4 x 3
##      id score name
##   <int> <int> <chr>
##       1    35 issac
##       3    70 soony
##       4    63 bomi
##       2    NA jelly
```

inner_join()

교집합 만 빼서 테이블 생성

tab_a

id	score
1	35
3	70
4	63
5	80

tab_b

id	name
1	issac
2	jelly
3	soony
4	bomi

```
inner_join(tab_a, tab_b) %>%
  print()
```

```
## Joining, by = "id"
```

```
## # A tibble: 3 x 3
##      id score name
##   <int> <int> <chr>
##       1    35 issac
##       3    70 soony
##       4    63 bomi
```

full_join()

모든 데이터 보존 테이블 생성

tab_a

id	score
1	35
3	70
4	63
5	80

tab_b

id	name
1	issac
2	jelly
3	soony
4	bomi

```
full_join(tab_a, tab_b) %>%
  print()
```

```
## Joining, by = "id"
```

```
## # A tibble: 5 x 3
##      id score name
##   <int> <int> <chr>
##       1    35 issac
##       3    70 soony
##       4    63 bomi
##       5    80 <NA>
##       2    NA jelly
```

Filtering join

하나의 테이블을 다른 테이블로 필터

semi_join()

매칭 데이터 보존 테이블 생성

tab_a

id	score
1	35
3	70
4	63
5	80

tab_b

id	name
1	issac
2	jelly
3	soony
4	bomi

```
semi_join(tab_a, tab_b) %>%
  print()
```

```
## Joining, by = "id"
```

```
## # A tibble: 3 x 2
##      id score
##   <int> <int>
##       1    35
##       3    70
##       4    63
```

anti_join()

매칭 데이터 제거 테이블 생성

tab_a

id	score
1	35
3	70
4	63
5	80

tab_b

id	name
1	issac
2	jelly
3	soony
4	bomi

```
anti_join(tab_a, tab_b) %>%
  print()
```

```
## Joining, by = "id"
```

```
## # A tibble: 1 x 2
##      id score
##   <int> <int>
##       5    80
```

이동 데이터로 실습하기

시도 정보를 `moving_data`에 합쳐보세요!

- 출발 시군구 코드

```
library(tidyverse)
library(magrittr)

# 데이터 로드
moving_data <- read_csv("./data/seoul_moving_202107_09_hr.csv")
reference_data <- readxl::read_excel("./data/reference.xlsx")

# 변수 이름 수정
moving_data <- janitor::clean_names(moving_data)
reference_data <- janitor::clean_names(reference_data)
```

문제. 서울시에서 출발한 데이터는 몇개?

힌트

- 두 테이블에는 공통으로 들어있는 정보가 있습니다~! 이 정보를 이용해서 시도 코드를 `moving_data`에 붙여보세요!

`mutate()` 심화학습

궁합좋은 함수들, 꿀팁들

만들 위치 지정

- 만들 위치 지정하기
  - `.after`, `.before` 옵션

```
tab_a %<>%
  mutate(ranking = rank(score),
         .after = "id")
tab_a
```

```
## # A tibble: 4 x 3
##      id ranking score
##   <int>   <dbl> <int>
## 1     1       1    35
## 2     3       3    70
## 3     4       2    63
## 4     5       4    80
```

빈칸 (NA) 과 관련한 함수들

- NA와 관련한 함수들
  - na_if():
  - coalesce():
  - replace_na(): tidyr 패키지 함수

```
y <- c("a", "b", "", "c")
y_na <- na_if(y, "")
y_na
coalesce(y_na, "hi")
replace_na(y_na, "hello")
```

if_else()

특정 조건으로 값 만들기

```
tab_a %<>%
  mutate(
    status = if_else(
      ranking < 3,
      "high",
      "low"),
    .after = "ranking"
  )
tab_a
```

```
## # A tibble: 4 x 4
##      id ranking status score
##   <int>   <dbl> <chr>  <int>
## 1     1       1 high      35
## 2     3       3 low       70
## 3     4       2 high      63
## 4     5       4 low       80
```

- `NA`를 입력하고 싶을땐?

case_when()

조건 여러개로 값 만들기

- 주의사항: 에서 점점 넓혀가야 함

```
tab_a %>%
  mutate(
    status = case_when(
      ranking == 1 ~ "leader",
      ranking < 3  ~ "followers"
      ranking == 3 ~ "third",
      TRUE ~ "else"),
    .after = "ranking"
  )
```

```
## # A tibble: 4 x 4
##       id ranking status    scc
##    <int>   <dbl> <chr>     <ir
## 1      1       1 leader
## 2      3       3 third
## 3      4       2 followers
## 4      5       4 else
```

이동시간을 이용한 trip 분류

- 이동시간을 시간단위로 바꿔 새로운 변수를 만들어보세요.
  - `trip_time_hr` 이름 사용
- `trip_time_hr`을 이용해서 `trip_time_class`를

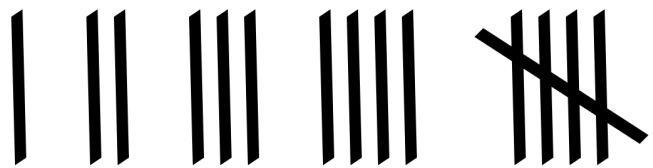
```
moving_data %<>%
  mutate(trip_time_hr = pyeong_gyun_idong_sigan_bun / 60) %>%
  mutate(trip_time =
           case_when(
               between(trip_time_hr, 0, 0.5) ~ "short trip",
               between(trip_time_hr, 0.5, 1) ~ "middle trip",
               trip_time_hr >= 1 ~ "longer trip",
               TRUE ~ as.character(trip_time_hr)
           ), .before = 1
  )
```

그룹 데이터 다루기

`group_by()`를 이용한 그룹별 내용 요약

`group_by`를 이용한 그룹 데이터 만들기

- grouping이 된 tibble은 프린트를 하면 표시가 됨.
  - 이러한 그룹 변수를 사용해서 궁합이 좋은 함수들이 있음
  - `arrange()`, `count()`, `tally()`



```
tab_a %<>%
  group_by(status)
tab_a
```

```
## # A tibble: 4 x 4
## # Groups:   status [2]
##      id ranking status score
##   <int>   <dbl> <chr>  <int>
## 1     1       1 high      35
## 2     3       3 low       70
## 3     4       2 high      63
## 4     5       4 low       80
```

`count()`와 `tally()`

- `tally()`는 그룹 데이터를 그룹 변수별 count를 해줌.
- `count()`는 와 를 한방 해결해 줌.

```
tab_a %>% tally()
```

```
## # A tibble: 2 x 2
##   status     n
##   <chr>  <int>
## 1 high       2
## 2 low        2
```

`arrange()` 함수와의 궁합

```
tab_a %>%
  arrange(id) %>%
  print()
```

```
## # A tibble: 4 x 4
##      id ranking status score
##   <int>   <dbl> <chr>  <int>
##       1       1 high      35
##       3       3 low       70
##       4       2 high      63
##       5       4 low       80
```

```
tab_a %>%
  arrange(id, .by_group = TRUE)
  print()
```

```
## # A tibble: 4 x 4
##      id ranking status score
##   <int>   <dbl> <chr>  <int>
##       1       1 high      35
##       4       2 high      63
##       3       3 low       70
##       5       4 low       80
```

그룹 키와 그룹 해제

- `group_keys()`: 현재 묶인 그룹의 unique 값을 보여줌.
- `ungroup()`을 사용해서 그룹을 해제해 줄 수 있음.

```
tab_a %>% group_keys()
```

```
## # A tibble: 2 x 1
##   status
##   <chr>
## 1 high
## 2 low
```

```
tab_a %<>% ungroup()
tab_a
```

```
## # A tibble: 4 x 4
##      id ranking status score
##   <int>   <dbl> <chr>  <int>
```

실습해보기

- 서울시안에서 short trip이 가장 많이 일어나는 곳은?

다음시간

`dplyr` 고급 동사들

Column-wise operations

Row-wise operations

같이 보면 좋은 책 추천

[1] R for Data Science

- 웹 상에 무료 공개된 책입니다.
- 위 교재의 한글 번역본 R 을 활용한 데이터과학도 있습니다.
- 도서 제목 클릭하셔서 구매하시면 저의 사리사욕을 충당하는데 도움이 됩니다.