Data transformation

R for Data Science

index

- 5.1 Introduction
- 5.2 Filter rows with filter()
- 5.3 Arrange rows with arrange()
- 5.4 Select columns with select()
- 5.5 Add new variables with mutate()
- 5.6 Grouped summaries with
- 5.7 Grouped mutates (and filters)

5.1 Introduction

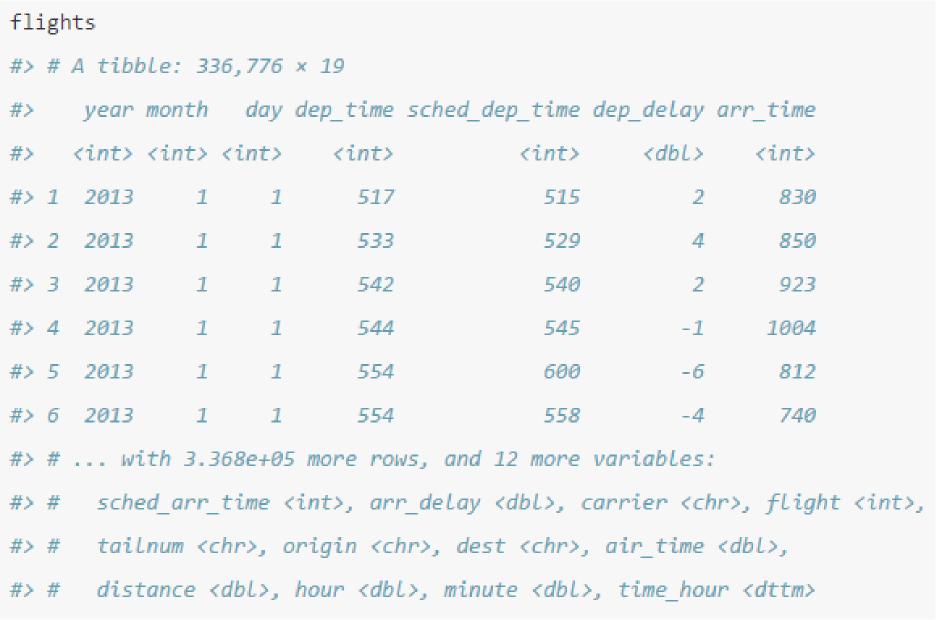
- Prerequisites

- library(nycflights13)
- library(tidyverse)

  -ggplot2, tibble, tidyr, readr, purrr, dplyr

5.1 Introduction
- nyflights13

- Nycflights13패키지의 flights데이터
- 2013년에 뉴욕에서 출발하는 336,776개의 항공편
- 전체 데이터 셋을 보기 위해서는 View(flights)입력



```
flights
#> # A tibble: 336,776 × 19
#>    year month   day dep_time sched_dep_time dep_delay arr_time
#>   <int> <int> <int>    <int>          <int>     <dbl>    <int>
#> 1  2013     1     1      517            515         2      830
#> 2  2013     1     1      533            529         4      850
#> 3  2013     1     1      542            540         2      923
#> 4  2013     1     1      544            545        -1     1004
#> 5  2013     1     1      554            600        -6      812
#> 6  2013     1     1      554            558        -4      740
#> # ... with 3.368e+05 more rows, and 12 more variables:
#> #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
#> #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#> #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

➢ 문자형 약어

- int - integers
- dbl - doubles, real numbers
- chr - character vectors, strings
- dttm – date+times
- lgl - logical
- fctr - factors
- date - 날짜

5.1 Introduction

- dplyr basics

➢ 데이터 전처리에 유용한 dplyr 명령어

- filter() - 값을 통한 관측치 선택
- arrage() - 행 재정렬
- select() - 이름을 통한 변수 선택
- mutate() - 기존 변수로 새 변수를 만듬
- summarise() - 여러 값 요약

+ group_by() – 그룹 단위로 묶어줌

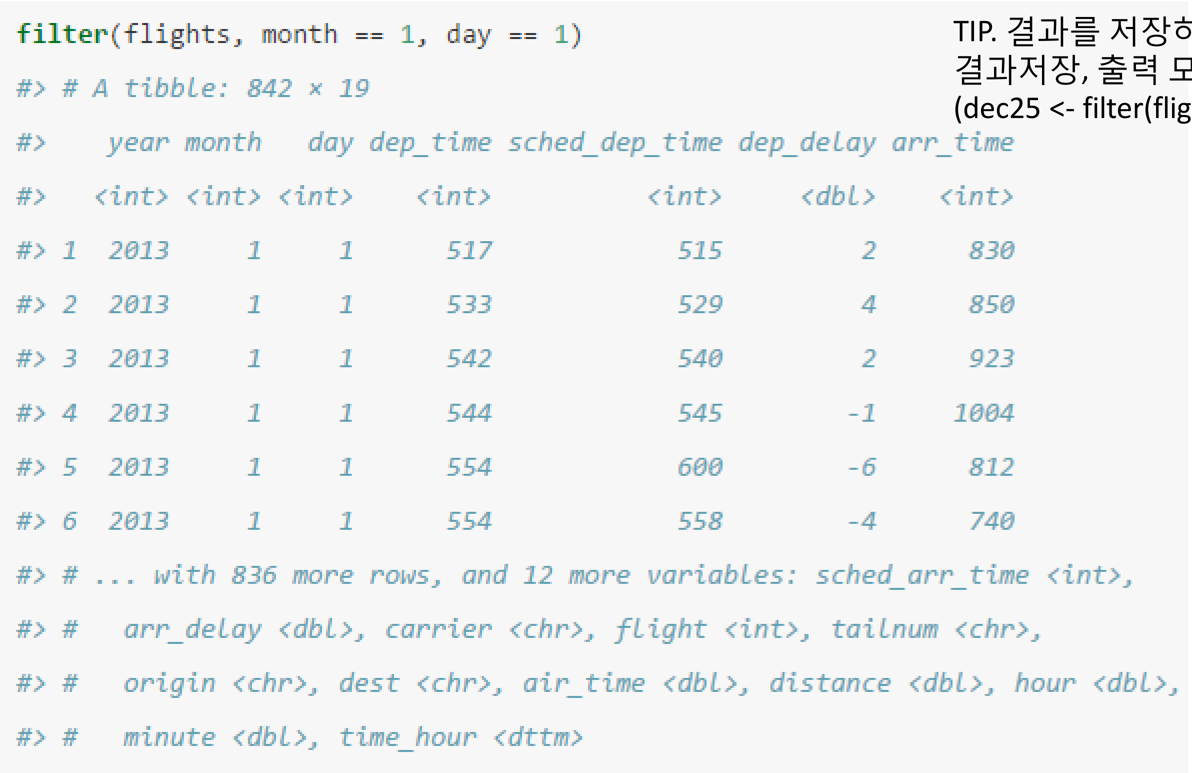
➢ 위 명령어들의 공통적인 특징

1. 첫 번째 인수는 데이터 프레임
2. 다음 인수는 따옴표를 제외한 변수 이름을 사용해서 데이터 프레임과 함께 무엇을 하는지 설명함
3. 그 결과는 새로운 데이터 프레임

5.2 Filter rows with filter()

- Filter()는 설정한 값을 기반해 관측치들의 부분집합을 만듬
- 첫 번째 인수는 데이터 프레임 명
- 그 다음 인수들은 필터링하는 표현식

< 1월 1일의 모든 항공편 출력 >



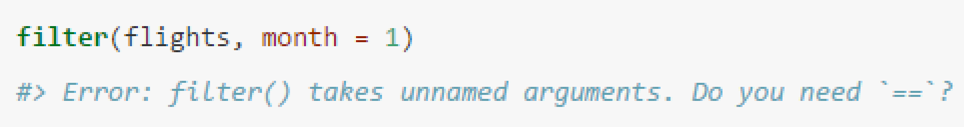
```
filter(flights, month == 1, day == 1)
#> # A tibble: 842 × 19
#>    year month   day dep_time sched_dep_time dep_delay arr_time
#>   <int> <int> <int>    <int>          <int>     <dbl>    <int>
#> 1  2013     1     1      517            515         2      830
#> 2  2013     1     1      533            529         4      850
#> 3  2013     1     1      542            540         2      923
#> 4  2013     1     1      544            545        -1     1004
#> 5  2013     1     1      554            600        -6      812
#> 6  2013     1     1      554            558        -4      740
#> # ... with 836 more rows, and 12 more variables: sched_arr_time <int>,
#> #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
#> #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#> #   minute <dbl>, time_hour <dttm>
```

TIP. 결과를 저장하고 싶을 경우 대입연산자 사용 (<-)
결과저장, 출력 모두 수행할 경우 괄호 사용
(dec25 <- filter(flights, month == 12, day == 25))

5.2 Filter rows with filter()

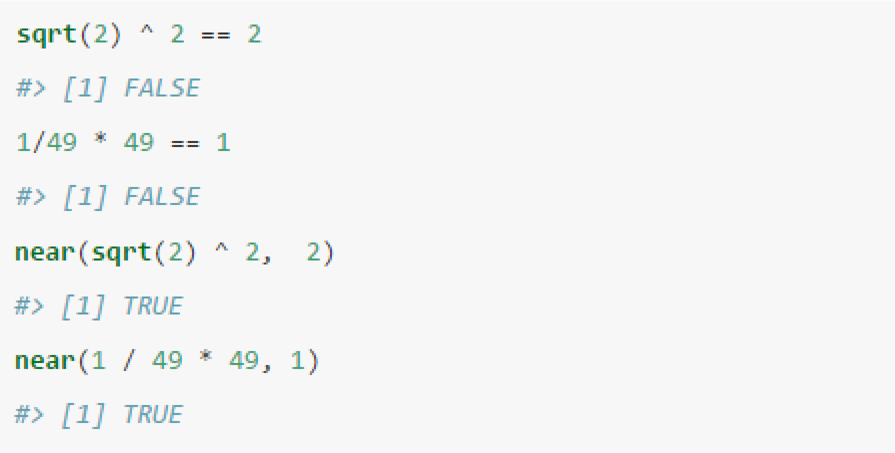
- Comparisons

- ➢ 비교 연산자
- >, >=, <, <=, !=, ==
- 흔히 하는 가장 쉬운 실수는==대신 =를 입력



```
filter(flights, month = 1)
#> Error: filter() takes unnamed arguments. Do you need `==`?
```

- ➢ ==(floating point numbers) 사용시 문제점
- 컴퓨터는 유한 정밀도 산술(finite precision arithmetic)을 사용
- 따라서 무한의 수를 저장할 수 없어 근사값을 사용함
- == 대신 near() 사용!



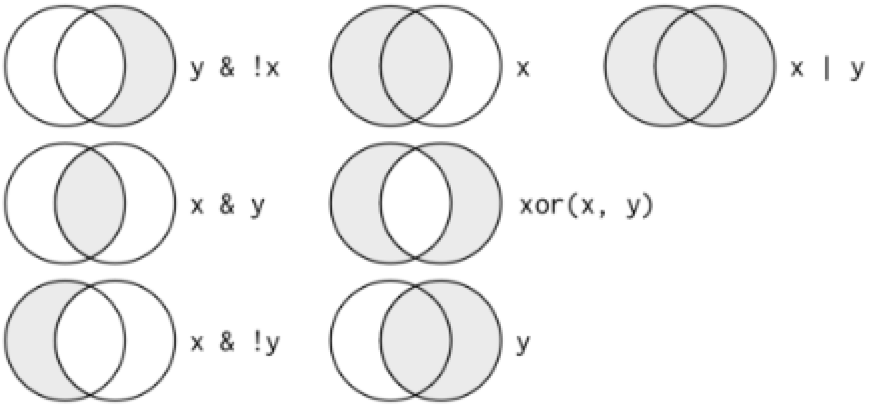
```
sqrt(2) ^ 2 == 2
#> [1] FALSE
1/49 * 49 == 1
#> [1] FALSE
near(sqrt(2) ^ 2,  2)
#> [1] TRUE
near(1 / 49 * 49, 1)
#> [1] TRUE
```

5.2 Filter rows with filter()

- Logical operators

Filter()에서는 다중 인수들이 “and”로 결합
-> 표현식들이 참이어야 함
“and”가 아닌 다른 유형의 조합일 경우 아래의 불연산자 사용

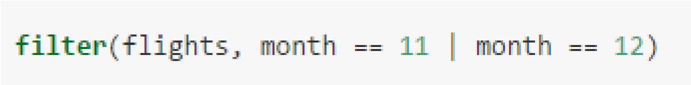
- ➢ Boolean operators
- & is “and”
- | is “or”
- ! is “not”



5.2 Filter rows with filter()

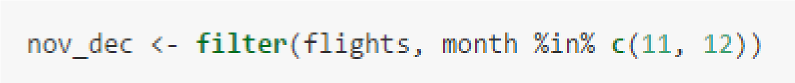
- Logical operators

<11월이나 12월에 출발한 항공편>



```
filter(flights, month == 11 | month == 12)
```

- x|y 로 쓸 경우 생기는 문제는 x %in% y로 해결가능
- x %in% y 는 y 값이 있는 x의 모든 행을 선택

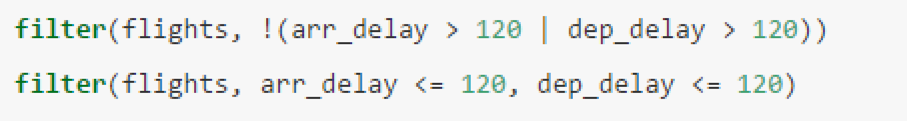


```
nov_dec <- filter(flights, month %in% c(11, 12))
```

➢ 드모르간 법칙을 이용해서 복잡한 부분집합을 단순화

- !(x & y) 는 !x | !y
- !(x | y) 는 !x & !y

<출발이나 도착이 2시간 이상 지연되지 않는 항공편>

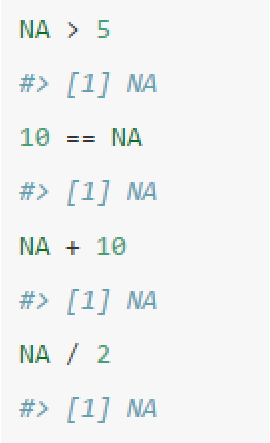


```
filter(flights, !(arr_delay > 120 | dep_delay > 120))
filter(flights, arr_delay <= 120, dep_delay <= 120)
```

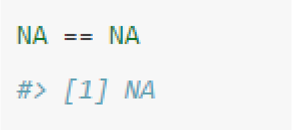
5.2 Filter rows with filter()

- Missing values

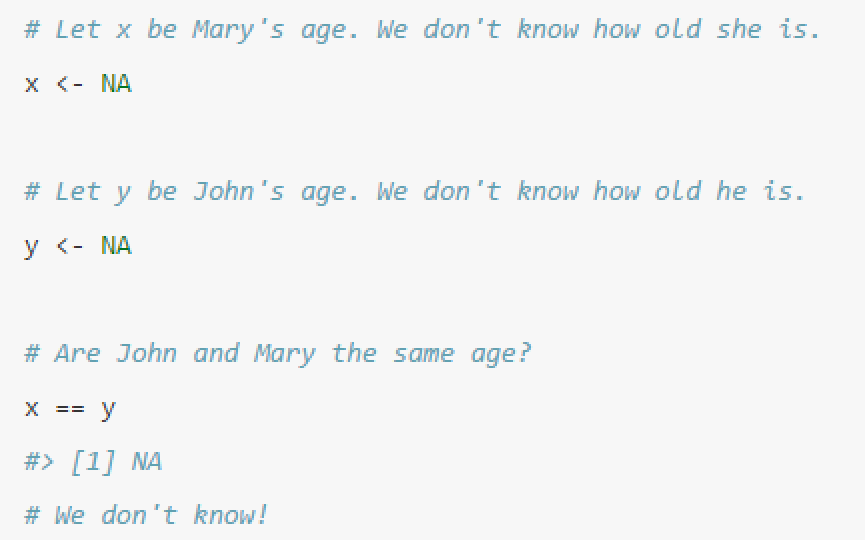
- NA represents an unknown value
- “contagious”
- almost any operation involving an unknown value will also be unknown



```
NA > 5
#> [1] NA
10 == NA
#> [1] NA
NA + 10
#> [1] NA
NA / 2
#> [1] NA
```



```
NA == NA
#> [1] NA
```

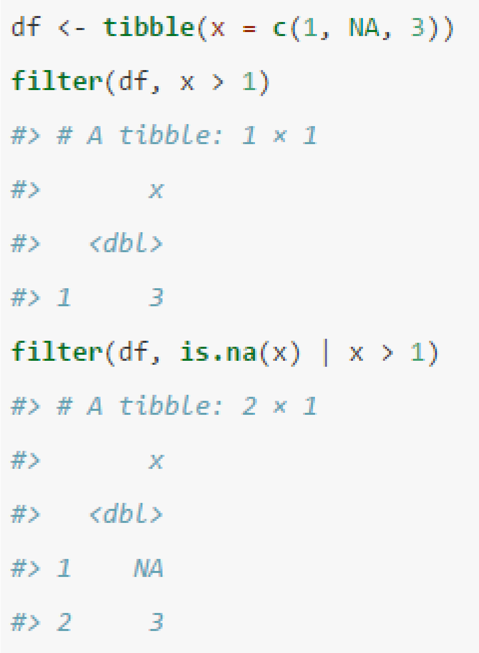


```
# Let x be Mary's age. We don't know how old she is.
x <- NA

# Let y be John's age. We don't know how old he is.
y <- NA

# Are John and Mary the same age?
x == y
#> [1] NA
# We don't know!
```

5.2 Filter rows with filter()

- Missing values

- Filter()는 조건이 True인 행만 포함, False와 NA값은 배제
- 결측값(missing values)를 포함하기 위해서는 명시적으로 표시
- Is. na() : 값이 누락 여부 판별



```
df <- tibble(x = c(1, NA, 3))
filter(df, x > 1)
#> # A tibble: 1 × 1
#>       x
#>   <dbl>
#> 1     3
filter(df, is.na(x) | x > 1)
#> # A tibble: 2 × 1
#>       x
#>   <dbl>
#> 1    NA
#> 2     3
```

5.2 Filter rows with filter()

- Exercises

1. Find all flights that
    1. Had an arrival delay of two or more hours
    2. Flew to Houston (IAH or HOU)
    3. Were operated by United, American, or Delta
    4. Departed in summer (July, August, and September)
    5. Arrived more than two hours late, but didn’t leave late
    6. Were delayed by at least an hour, but made up over 30 minutes in flight
    7. Departed between midnight and 6am (inclusive)

2. Another useful dplyr filtering helper is between(). What does it do? Can you use it to simplify the code needed to answer the previous challenges?

3. How many flights have a missing dep_time? What other variables are missing? What might these rows represent?

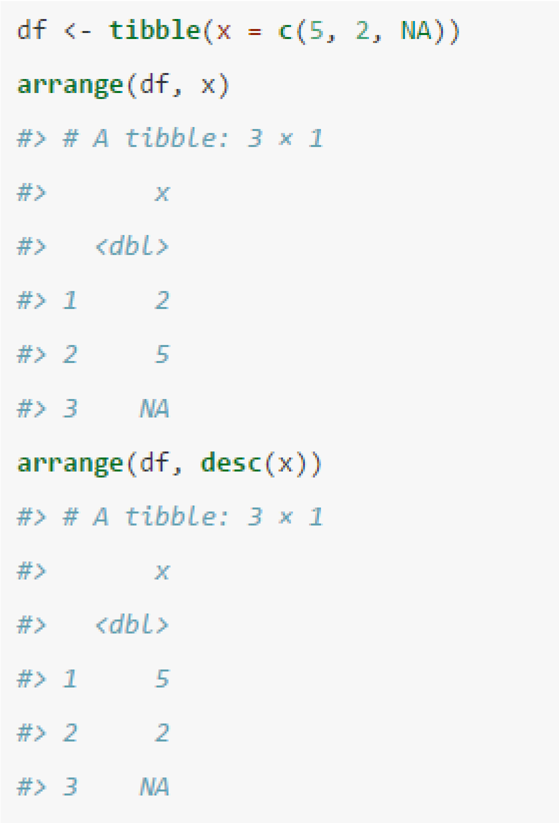
4. Why is NA ^ 0 not missing? Why is NA | TRUE not missing? Why is FALSE & NA not missing? Can you figure out the general rule? (NA * 0 is a tricky counterexample!)

5.3 Arrange rows with arrange()

- arrange()는 행을 선택하는 대신 순서가 변경된다는 것 외에는 filter()와 유사
- desc()를 사용해 내림차순으로 재정렬 가능

arrange(flights, desc(arr_delay))

- 결측치(missing value)는 항상 끝에 정렬됨



```
df <- tibble(x = c(5, 2, NA))
arrange(df, x)
#> # A tibble: 3 × 1
#>         x
#>     <dbl>
#> 1       2
#> 2       5
#> 3      NA
arrange(df, desc(x))
#> # A tibble: 3 × 1
#>         x
#>     <dbl>
#> 1       5
#> 2       2
#> 3      NA
```

5.3 Arrange rows with arrange()

- Exercises

1. How could you use arrange() to sort all missing values to the start? (Hint: use is.na()).

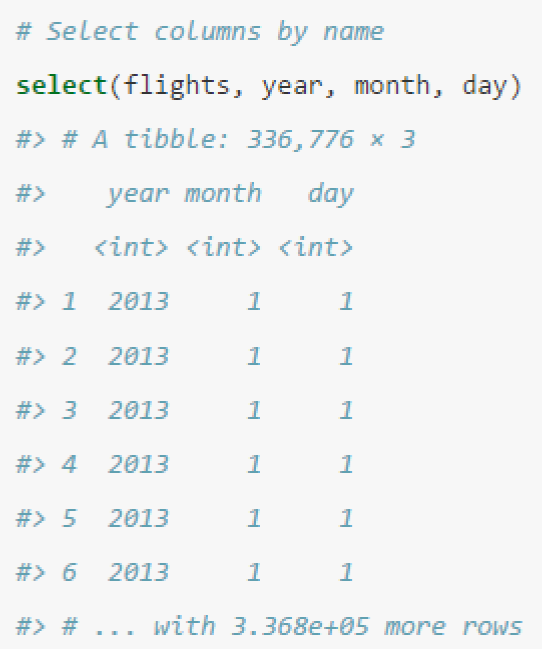
2. Sort flights to find the most delayed flights. Find the flights that left earliest.

3. Sort flights to find the fastest flights.

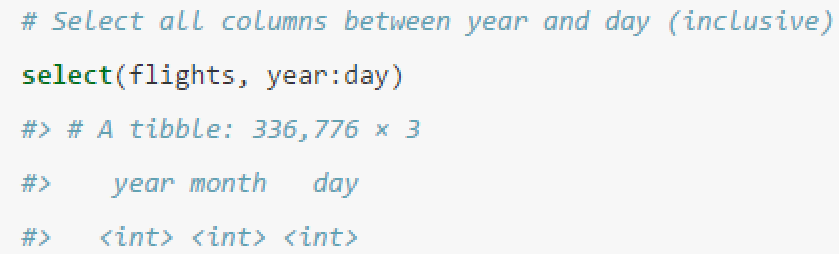
4. Which flights travelled the longest? Which travelled the shortest?

5.4 Select columns with select()

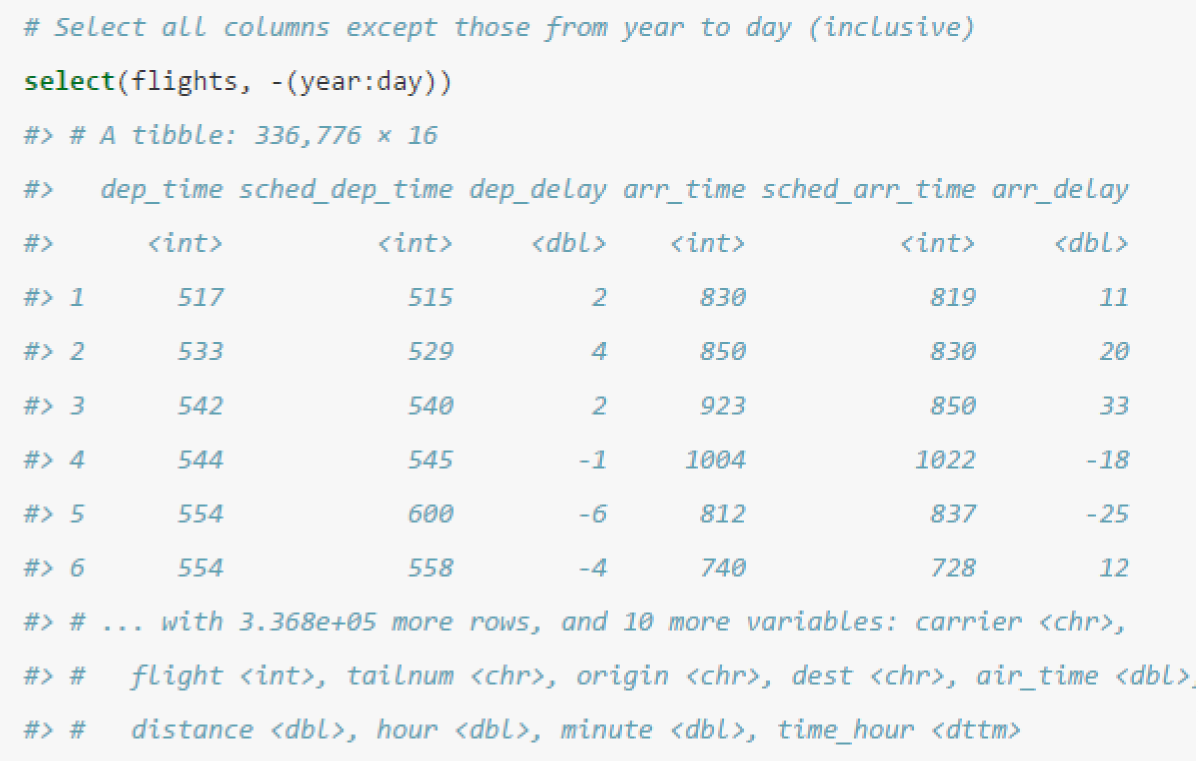
- Select()는 변수명 기반으로 원하는 변수 선택가능



```
# Select columns by name
select(flights, year, month, day)
#> # A tibble: 336,776 × 3
#>    year month   day
#>   <int> <int> <int>
#> 1  2013     1     1
#> 2  2013     1     1
#> 3  2013     1     1
#> 4  2013     1     1
#> 5  2013     1     1
#> 6  2013     1     1
#> # ... with 3.368e+05 more rows
```



```
# Select all columns between year and day (inclusive)
select(flights, year:day)
#> # A tibble: 336,776 × 3
#>    year month   day
#>   <int> <int> <int>
```



```
# Select all columns except those from year to day (inclusive)
select(flights, -(year:day))
#> # A tibble: 336,776 × 16
#>   dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
#>      <int>          <int>     <dbl>    <int>          <int>     <dbl>
#> 1      517            515         2      830            819        11
#> 2      533            529         4      850            830        20
#> 3      542            540         2      923            850        33
#> 4      544            545        -1     1004           1022       -18
#> 5      554            600        -6      812            837       -25
#> 6      554            558        -4      740            728        12
#> # ... with 3.368e+05 more rows, and 10 more variables: carrier <chr>,
#> #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#> #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

5.4 Select columns with select()

➢ select()에서 사용가능한 유용한 명령어

- stars_with("abc") : abc로 시작하는 이름을 매칭
- ends_with("xyz") : xyz로 끝나는 이름을 매칭
- contains("ijk") : ijk가 포함된 이름을 매칭
- match("(.)\\1") : 정규표현식과 일치하는 변수를 선택
- num_range("x", 1:3) : x1, x2, x3와 매칭

➢ select() : 변수이름을 변경가능하지만 명시하지 않는 변수들은 삭제됨

- rename() : 변수를 없애지 않으면서 변수이름을 변경

rename(flights, tail_num = tailnum)

- eveything() : 옵션으로 특정 변수들을 데이터 프레임의 시작부분으로 옮길 때 유용

select(flights, time_hour, air_time, **everything**())

5.4 Select columns with select()

- Exercises

1. Brainstorm as many ways as possible to select dep_time, dep_delay, arr_time, and arr_delay from flights.

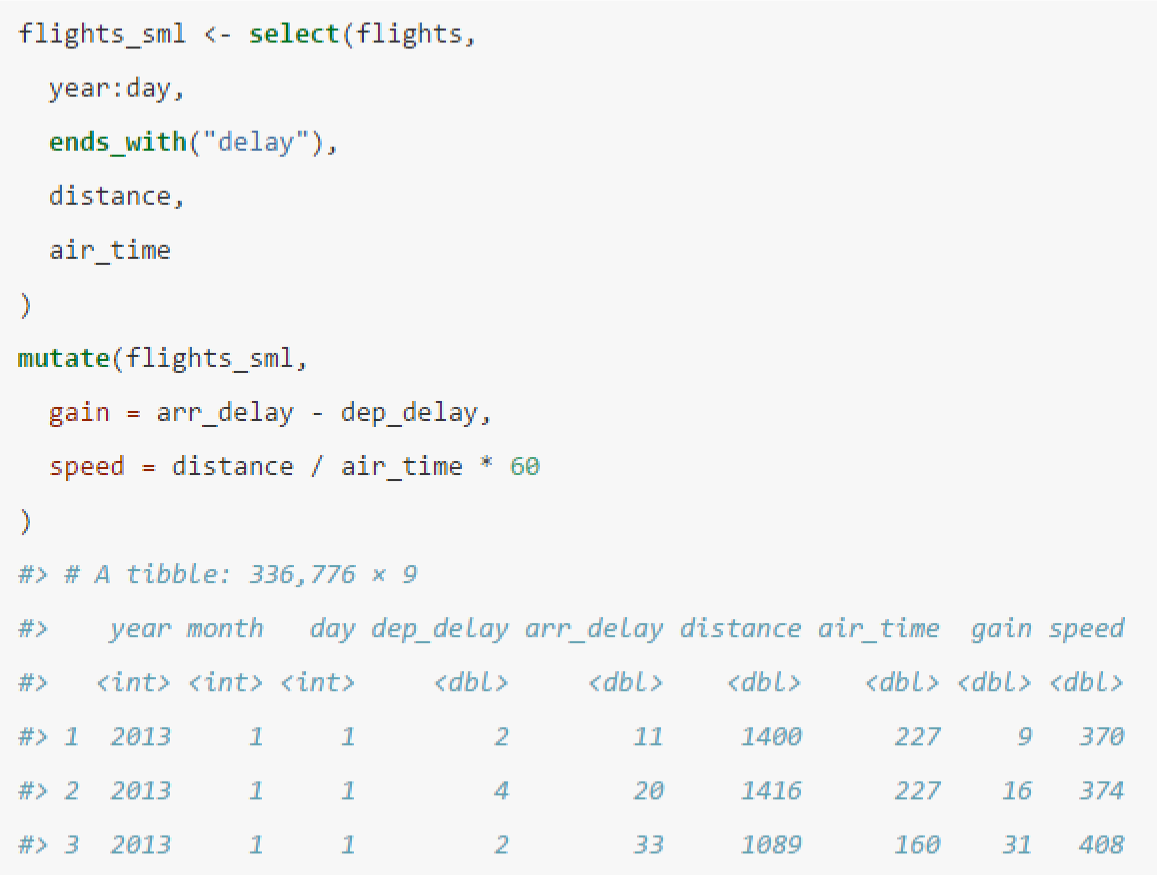
2. What happens if you include the name of a variable multiple times in a select() call?

3. What does the one_of() function do? Why might it be helpful in conjunction with this vector?
vars <- c("year", "month", "day", "dep_delay", "arr_delay")

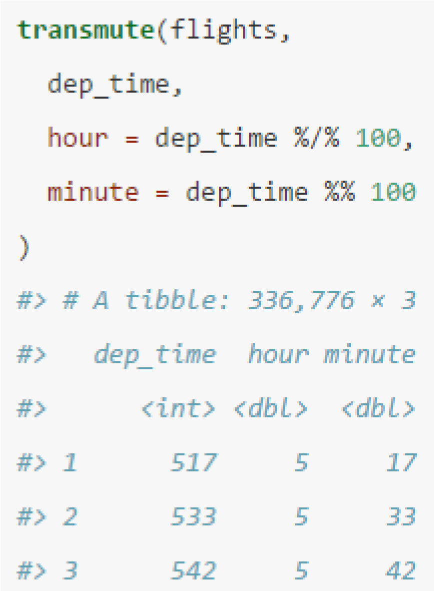
4. Does the result of running the following code surprise you? How do the select helpers deal with case by default? How can you change that default?
select(flights, contains("TIME"))

5.5 Add new variables with mutate()

- Mutate(): 기존 변수를 통해 새로운 변수를 생성
- 데이터 셋 끝에 새 열을 추가
- 금방 만들어낸 열을 활용가능
- Transmute() : 새 변수만 유지하고 싶을때 사용



```
flights_sml <- select(flights,
  year:day,
  ends_with("delay"),
  distance,
  air_time
)
mutate(flights_sml,
  gain = arr_delay - dep_delay,
  speed = distance / air_time * 60
)
#> # A tibble: 336,776 × 9
#>    year month   day dep_delay arr_delay distance air_time  gain speed
#>   <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl> <dbl>
#> 1  2013     1     1         2        11     1400      227     9   370
#> 2  2013     1     1         4        20     1416      227    16   374
#> 3  2013     1     1         2        33     1089      160    31   408
```



```
transmute(flights,
  dep_time,
  hour = dep_time %/% 100,
  minute = dep_time %% 100
)
#> # A tibble: 336,776 × 3
#>   dep_time  hour minute
#>      <int> <dbl>  <dbl>
#> 1      517     5     17
#> 2      533     5     33
#> 3      542     5     42
```

5.5 Add new variables with mutate()

- Useful creation functions

- Mutate()를 통해 새 변수를 만들기 위해 많은 기능들이 있음.
- 인풋, 아웃풋 모두 벡터 형태로 해야함

1. 산술 연산자 +, -, *, /, ^
2. 모듈러(modular) 산술

   %/%(정수 나누기), %%(나머지)

   x == y * (x %/% y) + (x %% Y)

   정수를 조각으로 나눌 수 있기 때문에 유용

3. 로그 : log(), log2(), log10()

   여러 차수의 데이터 범위를 다루는데 유용

   필자는 해석이 편한 log2() 추천

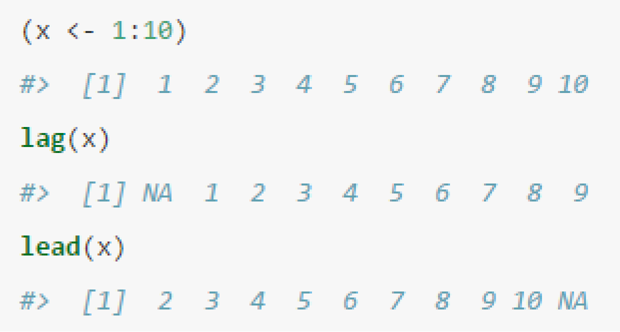
   log2() 눈금1차이는 원래의 두 배에 해당

4. 오프셋(offsets) : lead(), lag()

   시점을 하나 미루거나 당김

   x- lag(x) 처럼 전 값과의 차이 계산 가능

   group_by와 사용할 때 유용함



```
(x <- 1:10)
#>  [1]  1  2  3  4  5  6  7  8  9 10
lag(x)
#>  [1] NA  1  2  3  4  5  6  7  8  9
lead(x)
#>  [1]  2  3  4  5  6  7  8  9 10 NA
```

5.5 Add new variables with mutate()

- Useful creation functions

5. Cumulative and rolling aggregates

    cumsum(), cumprod(), cummin(), cummax(), cummean()

    합계, 곱, 최소, 최대, 평균 등 값이 누적됨

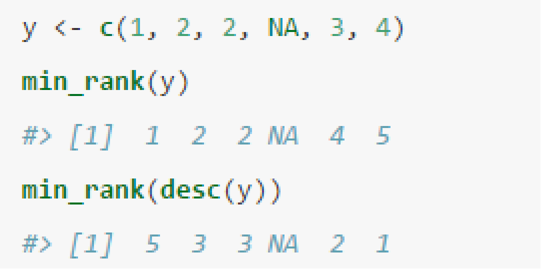
6. 논리비교 : < <= > >= !=

7. 순위 (ranking)

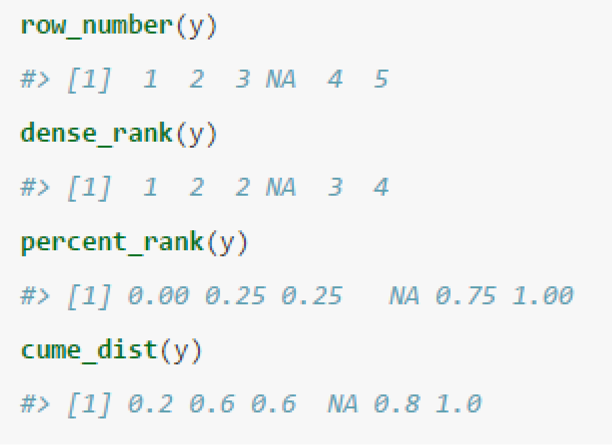
    min_rank() : 일반적인 순위 (1위, 2위…), 내림차순 가능

    row_number(), dense_rank()

    percent rank(), cume_dist(), ntile()



```
y <- c(1, 2, 2, NA, 3, 4)
min_rank(y)
#> [1]  1  2  2 NA  4  5
min_rank(desc(y))
#> [1]  5  3  3 NA  2  1
```



```
row_number(y)
#> [1]  1  2  3 NA  4  5
dense_rank(y)
#> [1]  1  2  2 NA  3  4
percent_rank(y)
#> [1] 0.00 0.25 0.25   NA 0.75 1.00
cume_dist(y)
#> [1] 0.2 0.6 0.6  NA 0.8 1.0
```

5.5 Add new variables with mutate()

- Exercises

1. Currently dep_time and sched_dep_time are convenient to look at, but hard to compute with because they’re not really continuous numbers. Convert them to a more convenient representation of number of minutes since midnight.

2. Compare air_time with arr_time - dep_time. What do you expect to see? What do you see? What do you need to do to fix it?

3. Compare dep_time, sched_dep_time, and dep_delay. How would you expect those three numbers to be related?

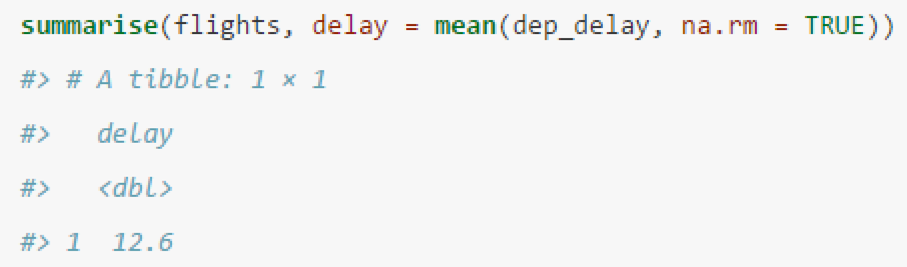
4. Find the 10 most delayed flights using a ranking function. How do you want to andle ties? Carefully read the documentation for min_rank().

5. What does 1:3 + 1:10 return? Why?

6. What trigonometric functions does R provide?

5.6 Grouped summaries with summarise()

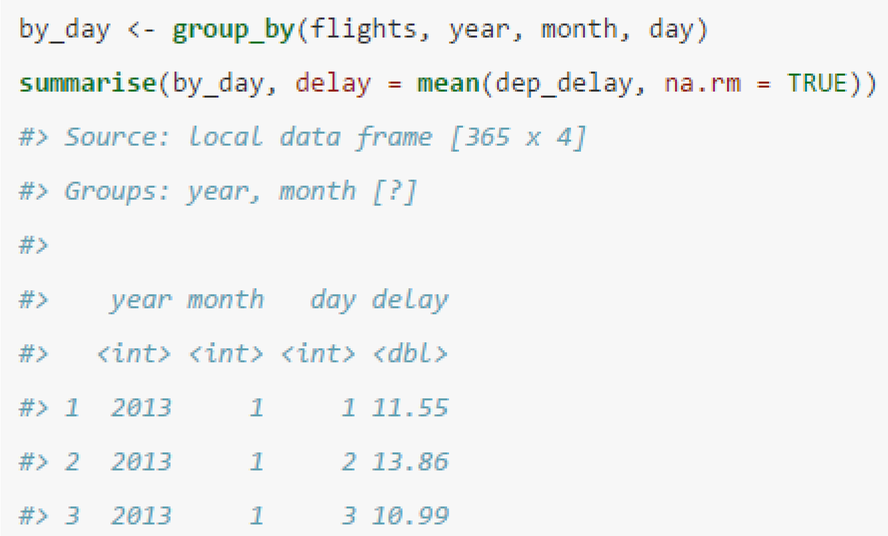
- Summarise() : 데이터프레임을 하나의 행으로 축소



```
summarise(flights, delay = mean(dep_delay, na.rm = TRUE))
#> # A tibble: 1 × 1
#>   delay
#>   <dbl>
#> 1  12.6
```

- Summarise 는 group_by()로 묶어주지 않으면 무쓸모
- Group_by()로 분석단위를 개별그룹으로 변경시킨 후, 그룹화된 데이터 프레임에서 dplyr 를 사용하면 그룹별로 적용이 됨

<날짜당 평균 출발 지연시간>

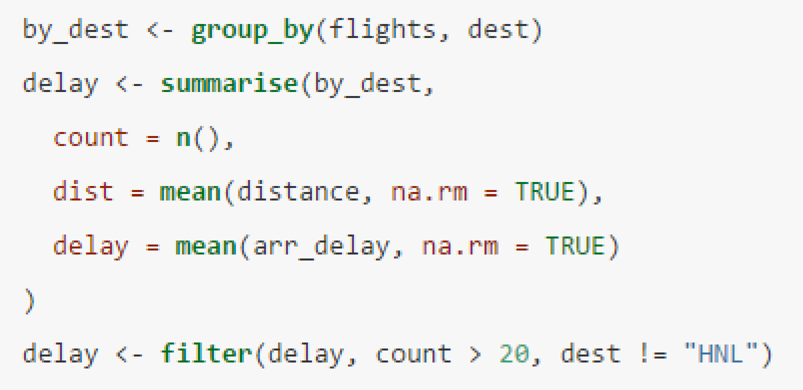


```
by_day <- group_by(flights, year, month, day)
summarise(by_day, delay = mean(dep_delay, na.rm = TRUE))
#> Source: local data frame [365 x 4]
#> Groups: year, month [?]
#>
#>    year month   day delay
#>   <int> <int> <int> <dbl>
#> 1  2013     1     1 11.55
#> 2  2013     1     2 13.86
#> 3  2013     1     3 10.99
```

5.6 Grouped summaries with summarise()

- Combining multiple operations with the pipe

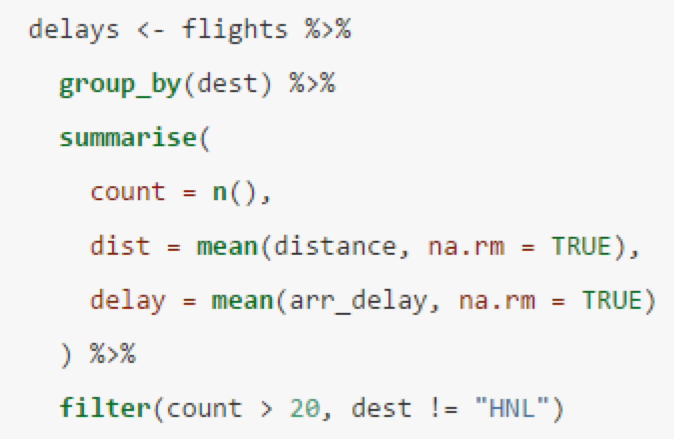
<각 위치에 대한 평균 도착 지연 시간과 거리 사이의 관계>



```r
by_dest <- group_by(flights, dest)
delay <- summarise(by_dest,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE)
)
delay <- filter(delay, count > 20, dest != "HNL")
```

1. 목적지별로 항공편을 그룹화
2. 거리, 평균 도착지연 시간, 항공편 수 요약
3. 잡음, 호놀룰루공항제거
   (다음 공항과 두 배로 멀리 있음)

=> 각 단계별로 중간 데이터 프레임 마다 쓸데없이 이름을 부여함.



```r
delays <- flights %>%
  group_by(dest) %>%
  summarise(
    count = n(),
    dist = mean(distance, na.rm = TRUE),
    delay = mean(arr_delay, na.rm = TRUE)
  ) %>%
  filter(count > 20, dest != "HNL")
```

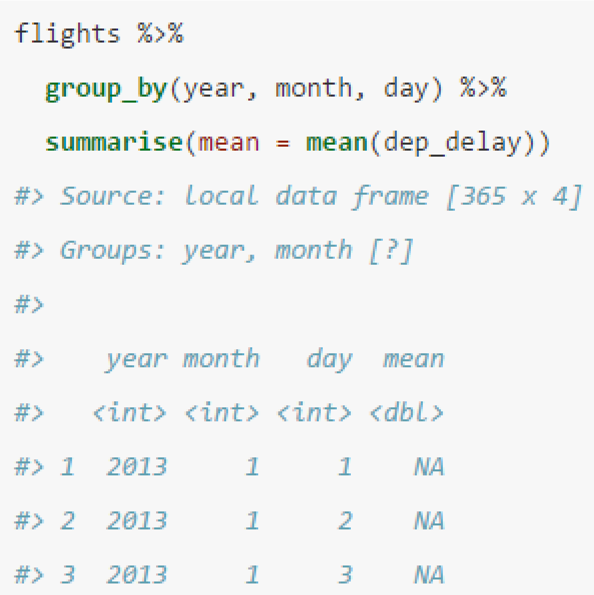
- %>% 파이프를 통해 위의 문제를 해결가능
- %>%를 사용하면 코드의 가독성을 높일 수 있음

- 내부적으로 x %>% f(y) 는 f(x, y)으로,
  x %>% f(y) %>% g(z) 는 g(f(x, y), z) 으로 변한다.

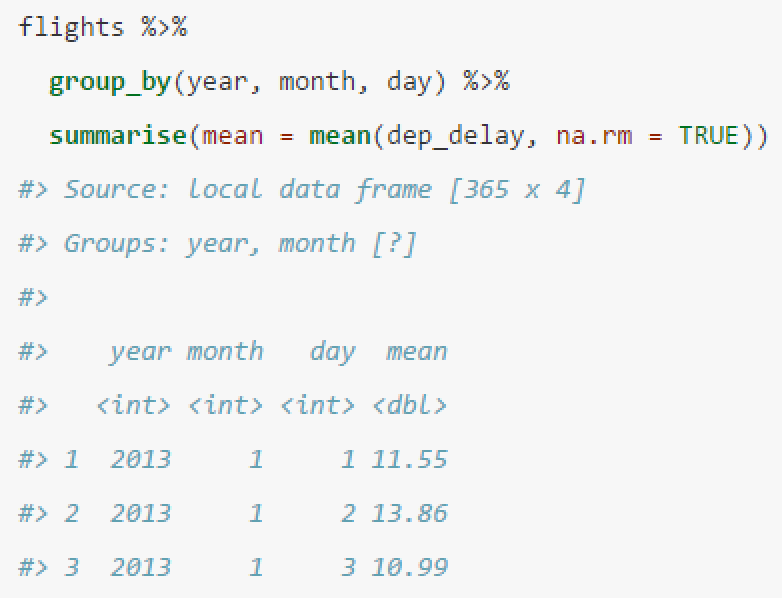
5.6 Grouped summaries with summarise()

- Missing values

- Na.rm를 설정하지 않을 경우 많은 결측값을 얻음
- 모든 집계함수에는 결측값을 제거하는 na.rm 인수가 있음



```
flights %>%
  group_by(year, month, day) %>%
  summarise(mean = mean(dep_delay))
#> Source: local data frame [365 x 4]
#> Groups: year, month [?]
#>
#>     year month   day  mean
#>    <int> <int> <int> <dbl>
#> 1   2013     1     1    NA
#> 2   2013     1     2    NA
#> 3   2013     1     3    NA
```



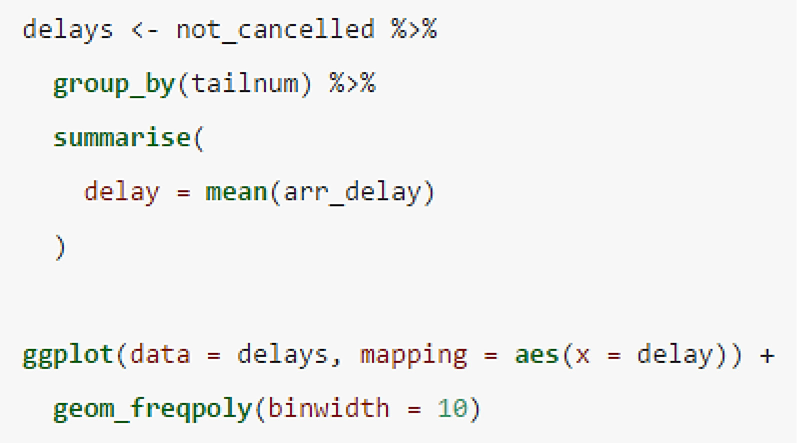
```
flights %>%
  group_by(year, month, day) %>%
  summarise(mean = mean(dep_delay, na.rm = TRUE))
#> Source: local data frame [365 x 4]
#> Groups: year, month [?]
#>
#>     year month   day  mean
#>    <int> <int> <int> <dbl>
#> 1   2013     1     1 11.55
#> 2   2013     1     2 13.86
#> 3   2013     1     3 10.99
```

5.6 Grouped summaries with summarise()

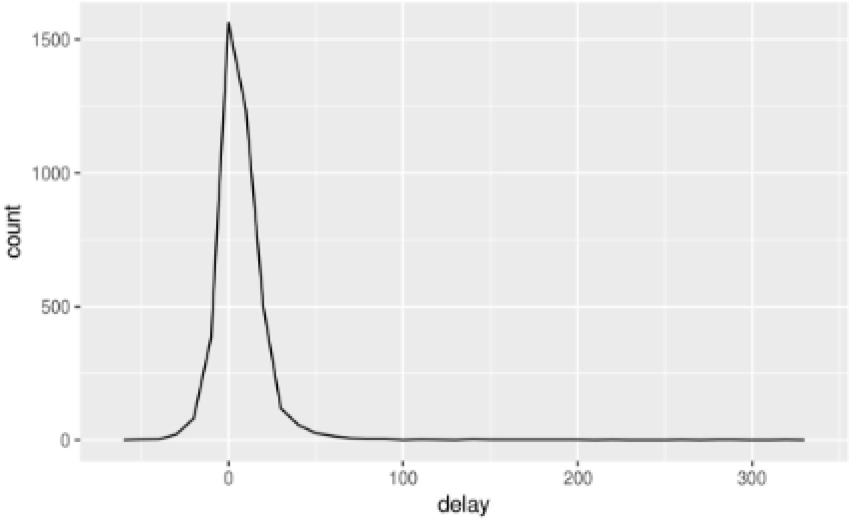
- Counts

TIP.
집계할 때마다 결측값을 뺀 count “sum(!is.na(x))”를 포함
이는 적은 양의 데이터를 기반으로 결론을 내리지 않았는지 확인할 수 있음

<평균 도착 지연이 큰 비행기(tail number로 식별가능)>



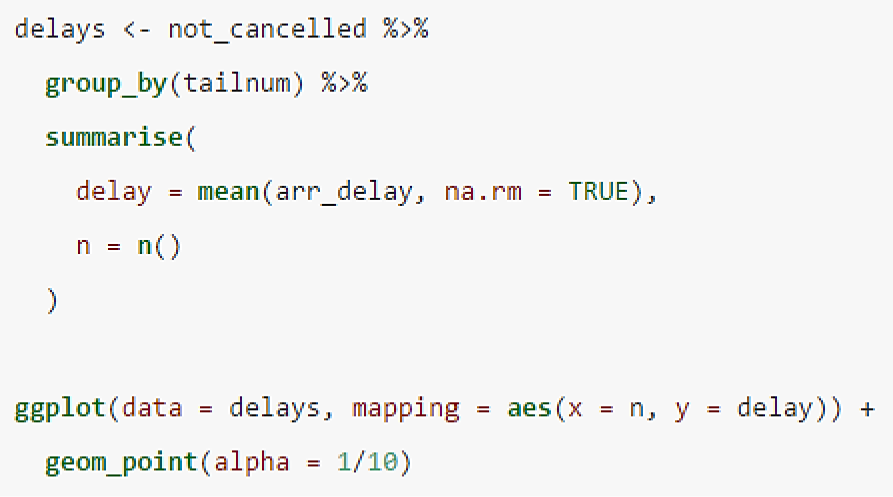
```
delays <- not_cancelled %>%
  group_by(tailnum) %>%
  summarise(
    delay = mean(arr_delay)
  )

ggplot(data = delays, mapping = aes(x = delay)) +
  geom_freqpoly(binwidth = 10)
```



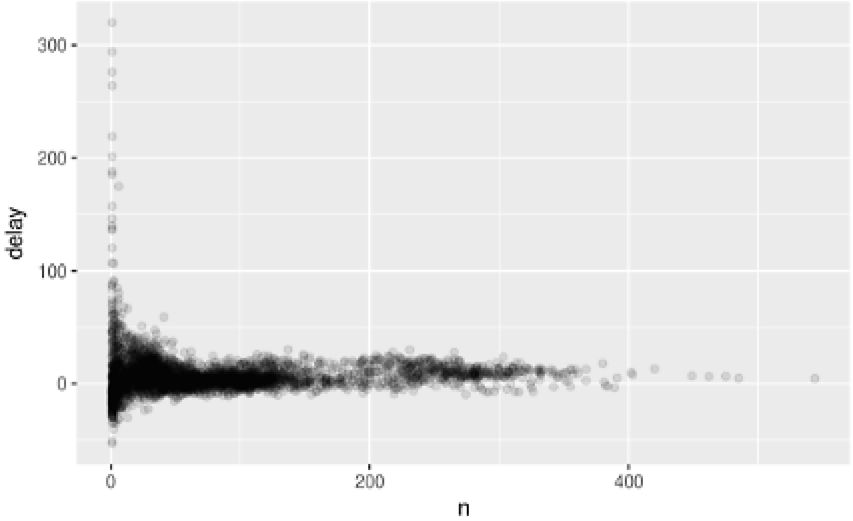
5.6 Grouped summaries with summarise()

- Counts

<비행 횟수 대 평균 지연이 큰 비행기(tail number로 식별가능)>



```
delays <- not_cancelled %>%
  group_by(tailnum) %>%
  summarise(
    delay = mean(arr_delay, na.rm = TRUE),
    n = n()
  )

ggplot(data = delays, mapping = aes(x = n, y = delay)) +
  geom_point(alpha = 1/10)
```



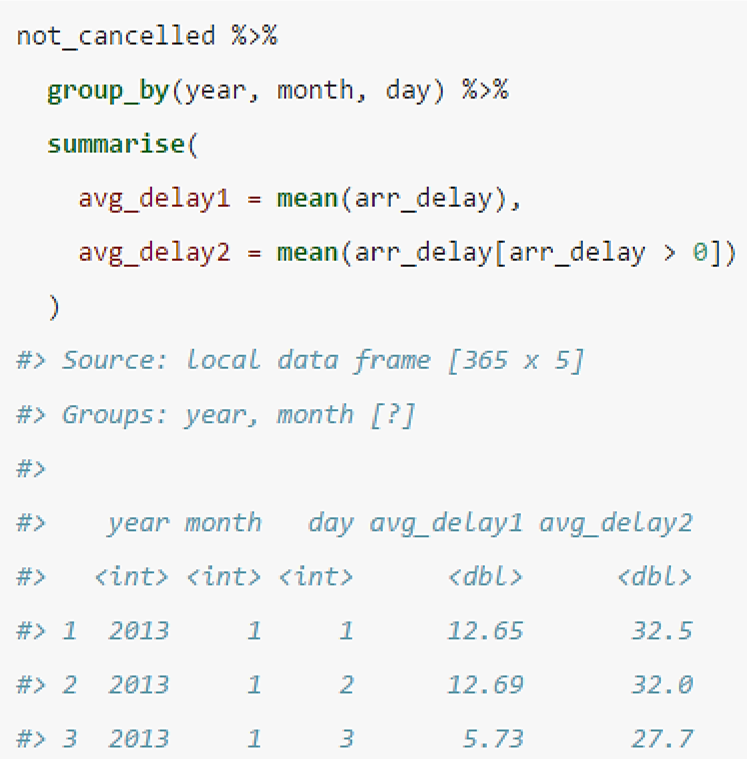
-> 항공편이 거의 없을 때 평균지연에 차이를 보임
-> 평균과 그룹크기를 산점도로 그려볼 때마다 표본크기에 따라 변동률이 감소함

5.6 Grouped summaries with summarise()

- Useful summary functions

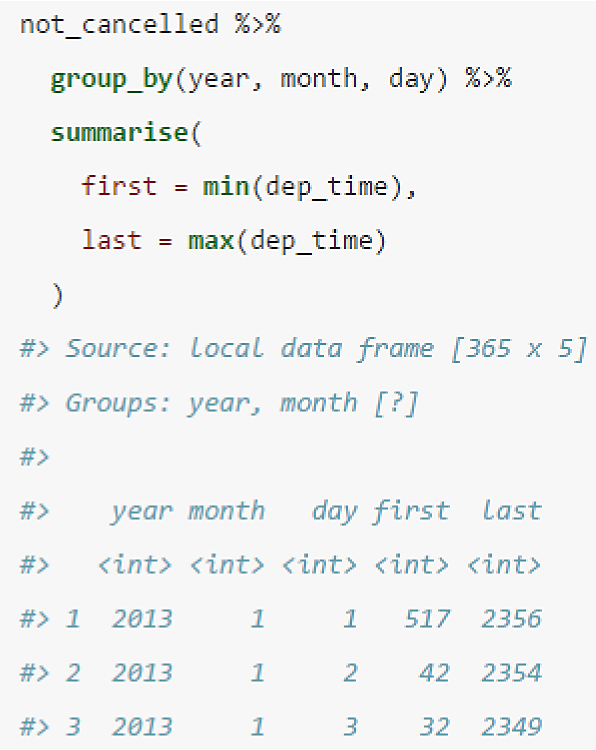
1. Measures of location : mean(), median()
2. Measures of spread : sd(x), IQR(x), mad(x)
3. Measures of rank : min(x), quantile(x, 0.25), max(x)

<날짜 별 평균 지연시간 >



```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(
    avg_delay1 = mean(arr_delay),
    avg_delay2 = mean(arr_delay[arr_delay > 0])
  )
#> Source: local data frame [365 x 5]
#> Groups: year, month [?]
#>
#>    year month   day avg_delay1 avg_delay2
#>   <int> <int> <int>      <dbl>      <dbl>
#> 1  2013     1     1      12.65       32.5
#> 2  2013     1     2      12.69       32.0
#> 3  2013     1     3       5.73       27.7
```

<날짜 별 첫 번째와 마지막 비행편의 시간>



```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(
    first = min(dep_time),
    last = max(dep_time)
  )
#> Source: local data frame [365 x 5]
#> Groups: year, month [?]
#>
#>    year month   day first  last
#>   <int> <int> <int> <int> <int>
#> 1  2013     1     1   517  2356
#> 2  2013     1     2    42  2354
#> 3  2013     1     3    32  2349
```

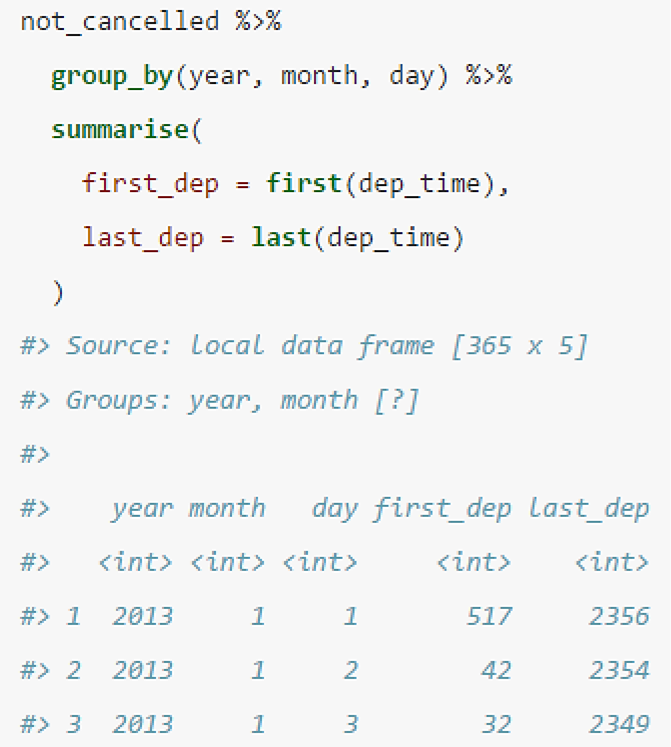
5.6 Grouped summaries with summarise()

- Useful summary functions

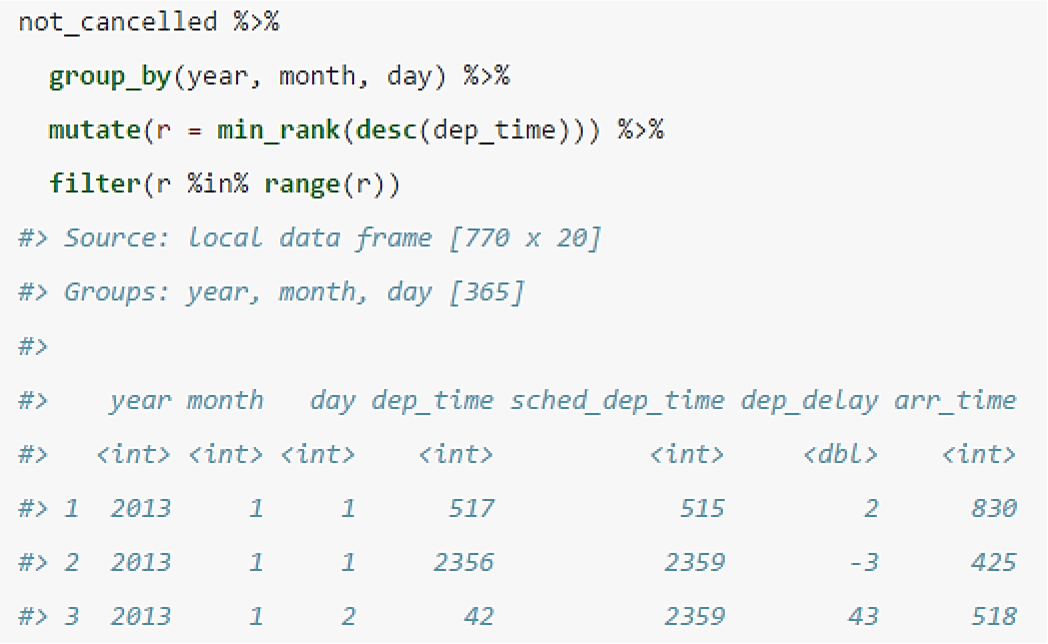
4. Measures of position : first(x), nth(x, 2), last(x)

x[1], x[2], x[length(x)]와 유사하지만 위치가 존재하지 않을 때 디폴트 값 설정가능

<날짜 별 첫 번째와 마지막 출발시각>



```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(
    first_dep = first(dep_time),
    last_dep = last(dep_time)
  )
#> Source: local data frame [365 x 5]
#> Groups: year, month [?]
#>
#>    year month   day first_dep last_dep
#>   <int> <int> <int>     <int>    <int>
#> 1  2013     1     1       517     2356
#> 2  2013     1     2        42     2354
#> 3  2013     1     3        32     2349
```



```
not_cancelled %>%
  group_by(year, month, day) %>%
  mutate(r = min_rank(desc(dep_time))) %>%
  filter(r %in% range(r))
#> Source: local data frame [770 x 20]
#> Groups: year, month, day [365]
#>
#>    year month   day dep_time sched_dep_time dep_delay arr_time
#>   <int> <int> <int>    <int>          <int>     <dbl>    <int>
#> 1  2013     1     1      517            515         2      830
#> 2  2013     1     1     2356           2359        -3      425
#> 3  2013     1     2       42           2359        43      518
```

5.6 Grouped summaries with summarise()

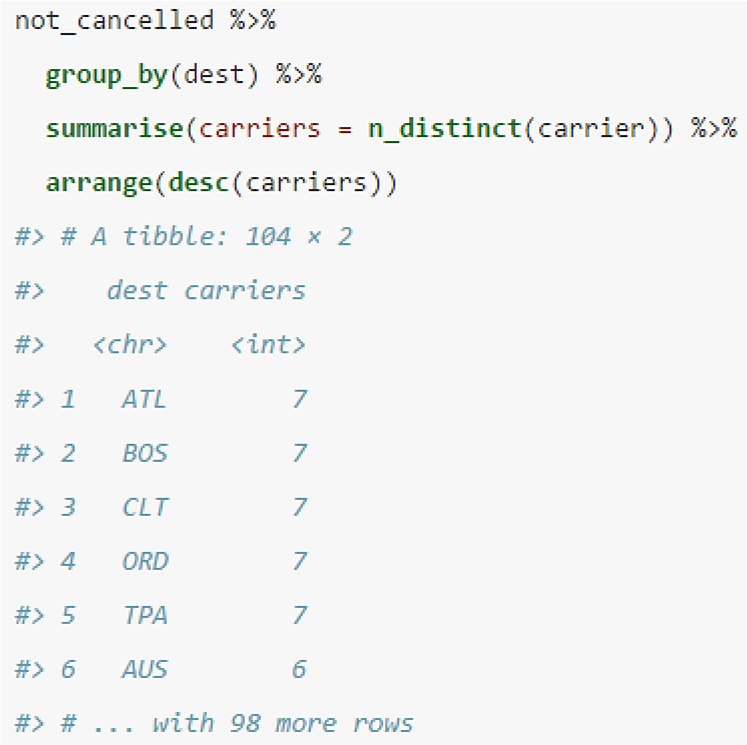
- Useful summary functions

5. Counts : n(), n_distinct()

n() 는 인자 없이도 해당 그룹의 크기를 반환

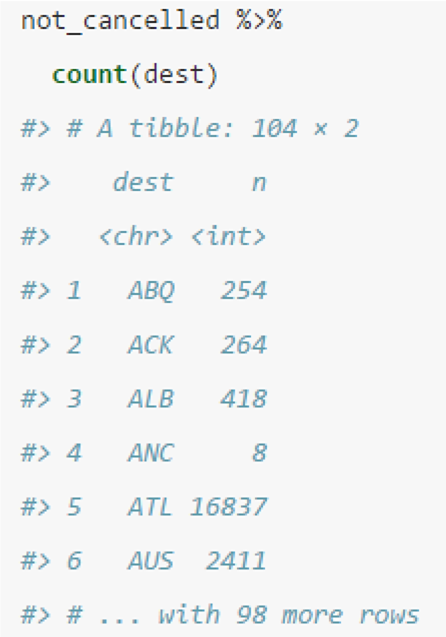
count()는 선택적으로 가중치를 줄 수 있음

<목적지별 항공사 개수>



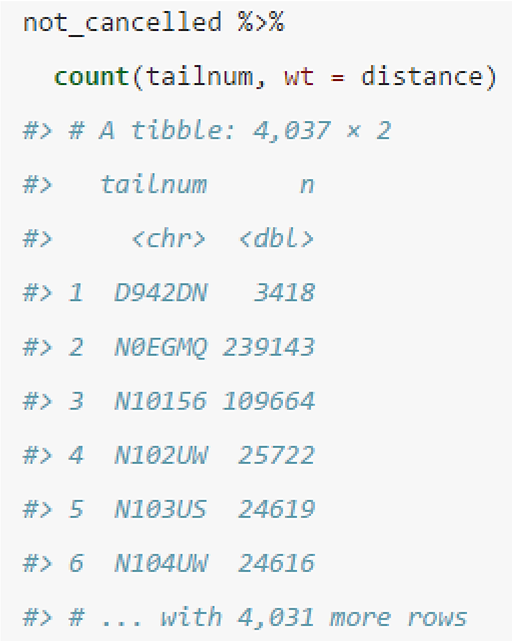
```
not_cancelled %>%
  group_by(dest) %>%
  summarise(carriers = n_distinct(carrier)) %>%
  arrange(desc(carriers))
#> # A tibble: 104 × 2
#>    dest carriers
#>    <chr>    <int>
#> 1   ATL         7
#> 2   BOS         7
#> 3   CLT         7
#> 4   ORD         7
#> 5   TPA         7
#> 6   AUS         6
#> # ... with 98 more rows
```

<목적지 개수>



```
not_cancelled %>%
  count(dest)
#> # A tibble: 104 × 2
#>     dest     n
#>    <chr> <int>
#> 1   ABQ    254
#> 2   ACK    264
#> 3   ALB    418
#> 4   ANC      8
#> 5   ATL  16837
#> 6   AUS   2411
#> # ... with 98 more rows
```

<거리에 가중치를 준 목적지 개수>



```
not_cancelled %>%
  count(tailnum, wt = distance)
#> # A tibble: 4,037 × 2
#>    tailnum      n
#>      <chr>  <dbl>
#> 1  D942DN    3418
#> 2  N0EGMQ  239143
#> 3  N10156  109664
#> 4  N102UW   25722
#> 5  N103US   24619
#> 6  N104UW   24616
#> # ... with 4,031 more rows
```

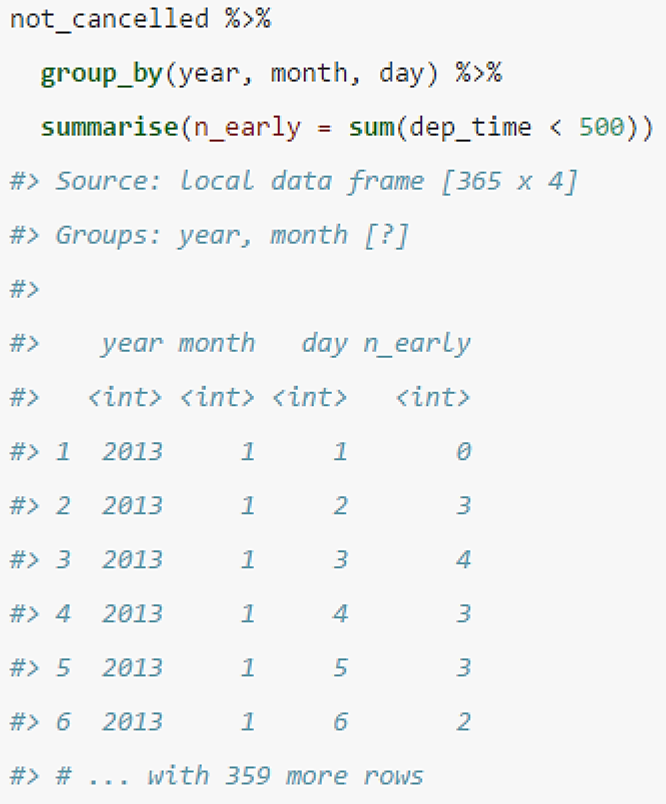
5.6 Grouped summaries with summarise()

- Useful summary functions

6. Counts and proportions of logical values: sum(x > 10), mean(y == 0)

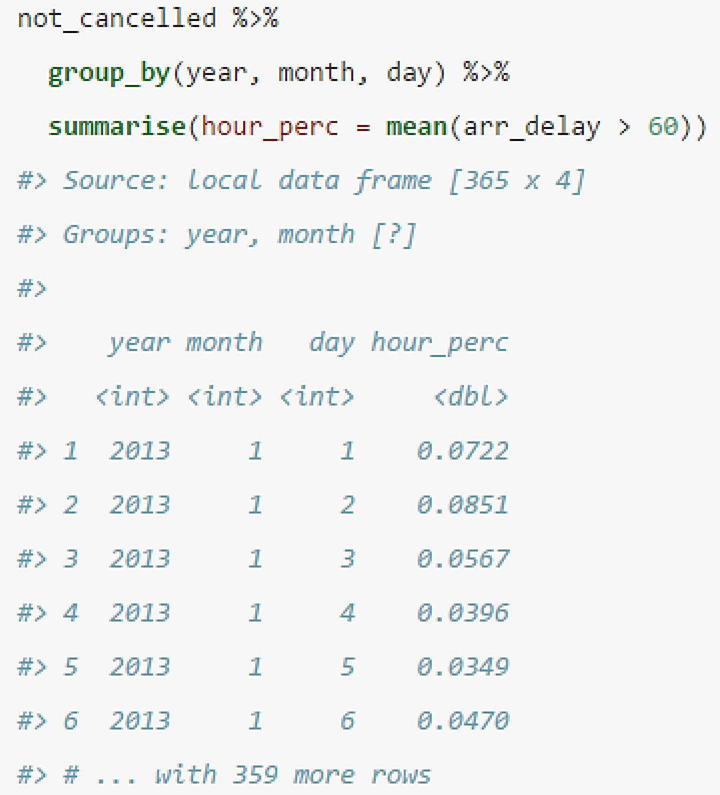
논리 값들은 수치함수와 사용될 때 True는 1, False는 0으로 변환
sum(x)는 x안의 TRUE 개수 반환, mean(x) 는 비율로 반환

<일별 5시 이전에 떠나는 항공편 개수>



```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(n_early = sum(dep_time < 500))
#> Source: local data frame [365 x 4]
#> Groups: year, month [?]
#>
#>     year month   day n_early
#>    <int> <int> <int>   <int>
#> 1   2013     1     1       0
#> 2   2013     1     2       3
#> 3   2013     1     3       4
#> 4   2013     1     4       3
#> 5   2013     1     5       3
#> 6   2013     1     6       2
#> # ... with 359 more rows
```

<일별 1시간 이상 지연되는 항공편 비율>

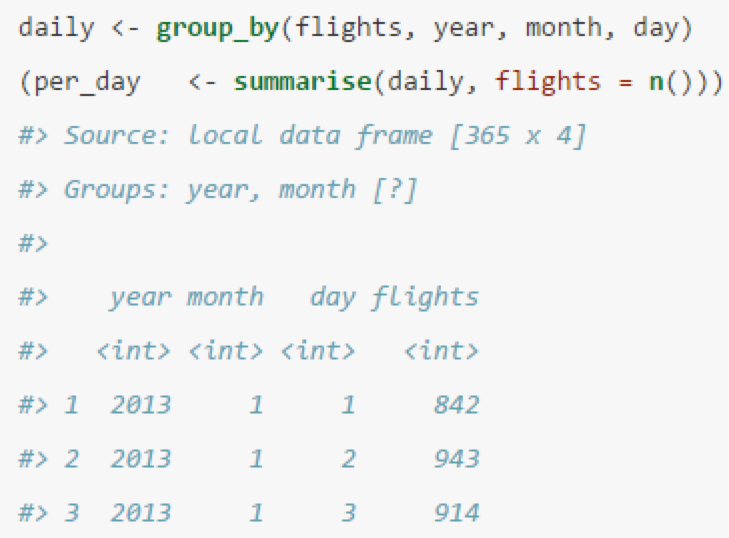


```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(hour_perc = mean(arr_delay > 60))
#> Source: local data frame [365 x 4]
#> Groups: year, month [?]
#>
#>     year month   day hour_perc
#>    <int> <int> <int>     <dbl>
#> 1   2013     1     1    0.0722
#> 2   2013     1     2    0.0851
#> 3   2013     1     3    0.0567
#> 4   2013     1     4    0.0396
#> 5   2013     1     5    0.0349
#> 6   2013     1     6    0.0470
#> # ... with 359 more rows
```

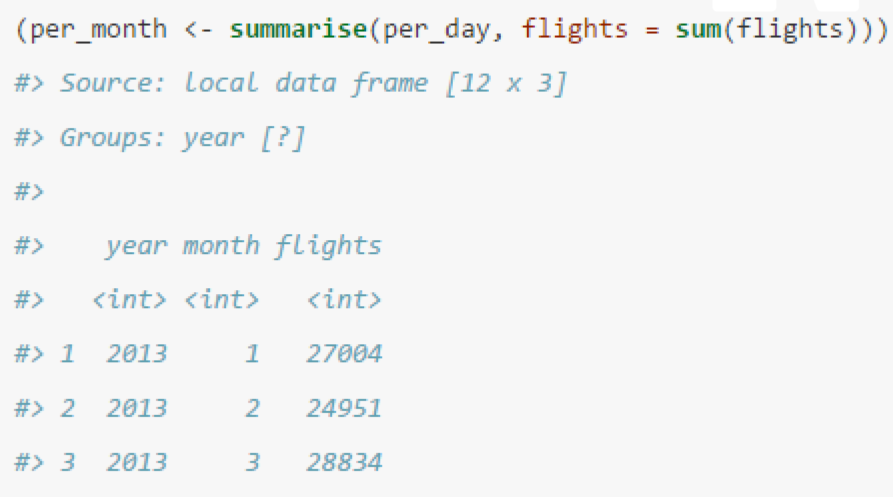
5.6 Grouped summaries with summarise()

- Grouping by multiple variables

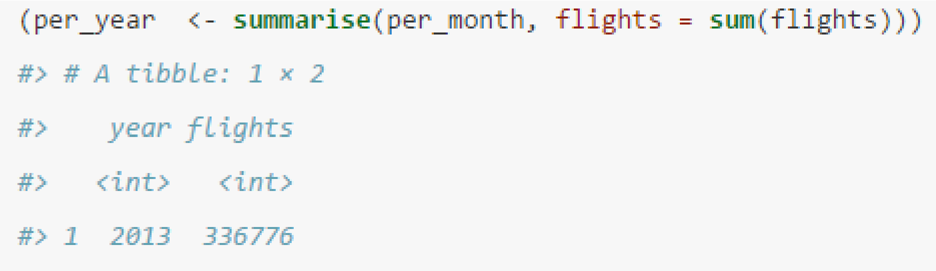
- ➢ 여러 변수를 사용해 그룹화할 때 각 요약은 그룹화의 한 수준을 벗겨냄
- ➢ 따라서 데이터 셋을 점진적으로 롤업하는 것을 쉽게 만듬



```
daily <- group_by(flights, year, month, day)
(per_day   <- summarise(daily, flights = n()))
#> Source: local data frame [365 x 4]
#> Groups: year, month [?]
#> 
#>     year month   day flights
#>    <int> <int> <int>   <int>
#> 1  2013     1     1     842
#> 2  2013     1     2     943
#> 3  2013     1     3     914
```



```
(per_month <- summarise(per_day, flights = sum(flights)))
#> Source: local data frame [12 x 3]
#> Groups: year [?]
#> 
#>     year month flights
#>    <int> <int>   <int>
#> 1  2013     1   27004
#> 2  2013     2   24951
#> 3  2013     3   28834
```



```
(per_year  <- summarise(per_month, flights = sum(flights)))
#> # A tibble: 1 × 2
#>     year flights
#>    <int>   <int>
#> 1  2013  336776
```

- ➢ 그룹화하며 점진적으로 요약할 경우 주의점

합계와 개수는 괜찮지만 가중평균과 분산에 대해 생각할 필요가 있음

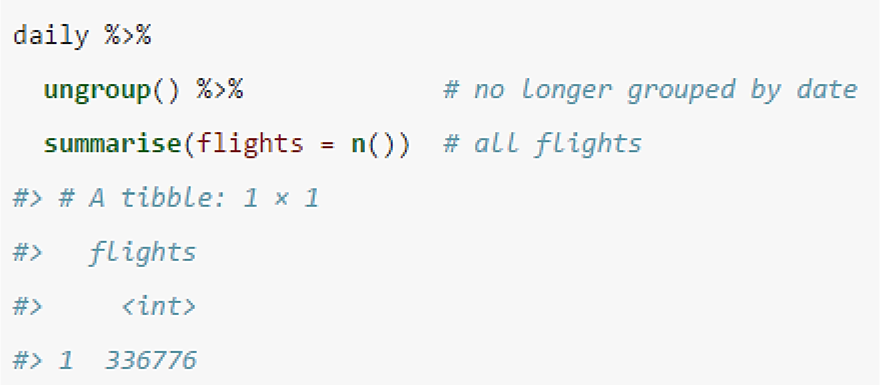
또한, 중앙값처럼 순위에 기반한 통계량은 정확하게 수행이 불가능

즉, 그룹 단위의 합계의 합은 전체의 합계지만

그룹단위의 중앙값의 중앙값은 전체의 중앙값이 아님

5.6 Grouped summaries with summarise()
- Ungrouping

- 그룹화를 제거해야 할 경우 ungroup() 사용



```
daily %>%
  ungroup() %>%                # no longer grouped by date
  summarise(flights = n())  # all flights
#> # A tibble: 1 × 1
#>   flights
#>     <int>
#> 1  336776
```

5.6 Grouped summaries with summarise()

- Exercises

1. Brainstorm at least 5 different ways to assess the typical delay characteristics of a group of flights.
Consider the following scenarios:
A flight is 15 minutes early 50% of the time, and 15 minutes late 50% of the time.
A flight is always 10 minutes late.
A flight is 30 minutes early 50% of the time, and 30 minutes late 50% of the time.
99% of the time a flight is on time. 1% of the time it's 2 hours late.
Which is more important: arrival delay or departure delay?

2. Come up with another approach that will give you the same output as not_cancelled %>% count(dest) and not_cancelled %>% count(tailnum, wt = distance) (without using count()).

3. Our definition of cancelled flights (is.na(dep_delay) | is.na(arr_delay)) is slightly suboptimal. Why? Which is the most important column?

4. Look at the number of cancelled flights per day. Is there a pattern? Is the proportion of cancelled flights related to the average delay?

5. Which carrier has the worst delays? Challenge: can you disentangle the effects of bad airports vs. bad carriers? Why/why not? (Hint: think about flights %>% roup_by (carrier, dest) %>% summarise(n()))

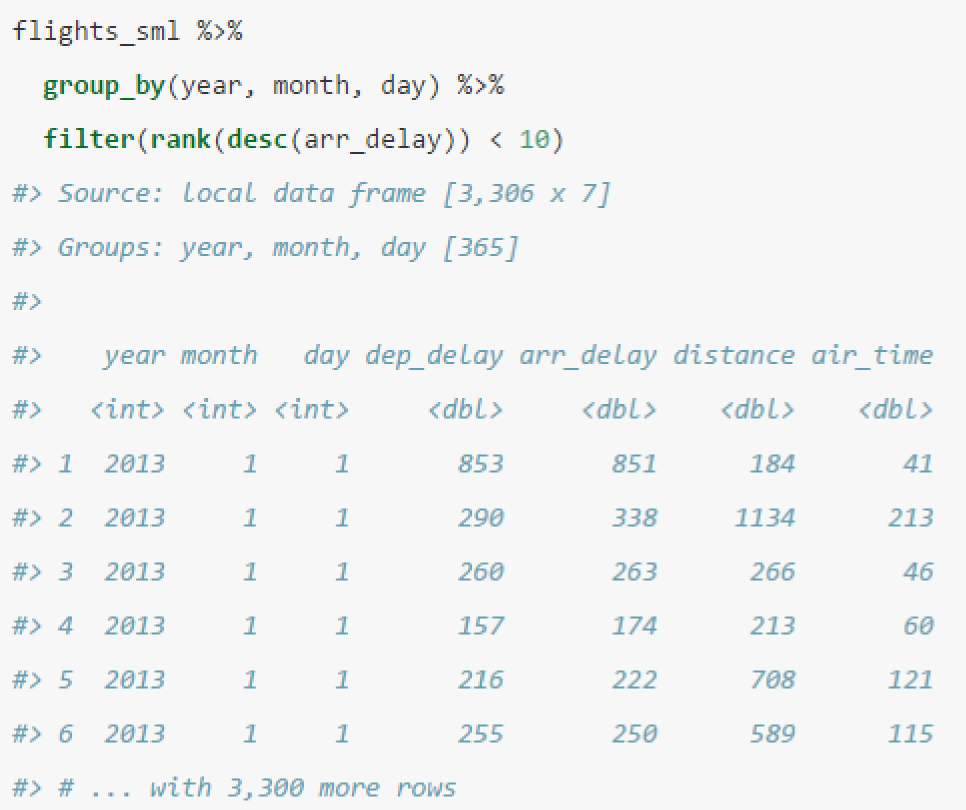
6. For each plane, count the number of flights before the first delay of greater than 1 hour.

7. What does the sort argument to count() do. When might you use it?

5.7 Grouped mutates (and filters)

✓ 그룹화는 summarise() 뿐 아니라 mutate(), filter()와 함께 쓸 때도 유용

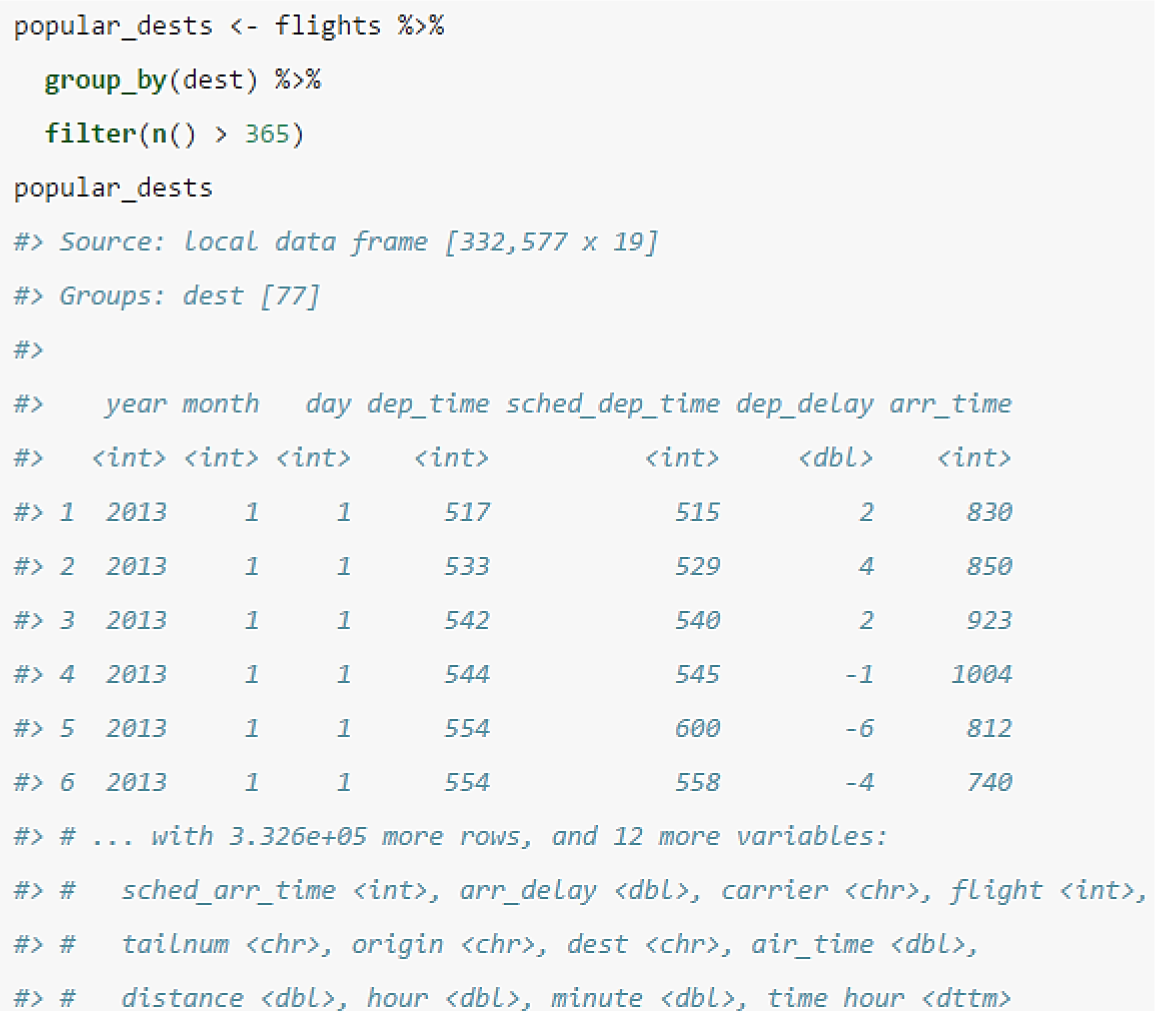
- Find the worst members of each group: 최악의 회원 찾기



```
flights_sml %>%
  group_by(year, month, day) %>%
  filter(rank(desc(arr_delay)) < 10)
#> Source: local data frame [3,306 x 7]
#> Groups: year, month, day [365]
#>
#>     year month   day dep_delay arr_delay distance air_time
#>    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl>
#> 1   2013     1     1       853       851      184       41
#> 2   2013     1     1       290       338     1134      213
#> 3   2013     1     1       260       263      266       46
#> 4   2013     1     1       157       174      213       60
#> 5   2013     1     1       216       222      708      121
#> 6   2013     1     1       255       250      589      115
#> # ... with 3,300 more rows
```

5.7 Grouped mutates (and filters)

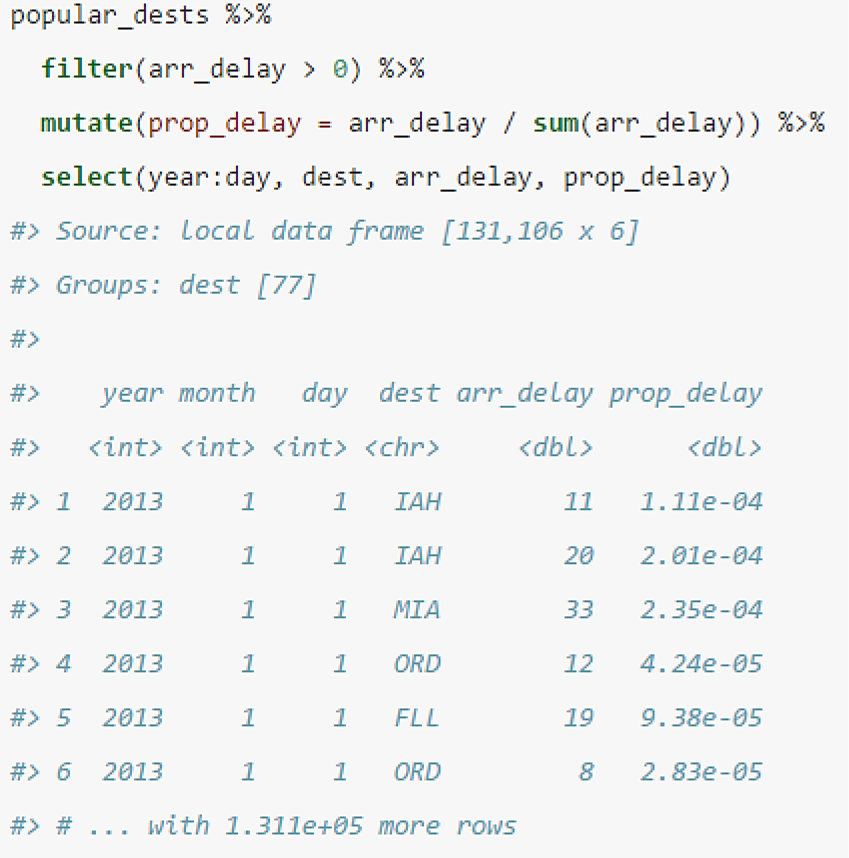
- Find all groups bigger than a threshold: 임계값보다 큰 그룹 찾기



```
popular_dests <- flights %>%
  group_by(dest) %>%
  filter(n() > 365)
popular_dests
#> Source: local data frame [332,577 x 19]
#> Groups: dest [77]
#>
#>    year month   day dep_time sched_dep_time dep_delay arr_time
#>   <int> <int> <int>    <int>          <int>     <dbl>    <int>
#> 1  2013     1     1      517            515         2      830
#> 2  2013     1     1      533            529         4      850
#> 3  2013     1     1      542            540         2      923
#> 4  2013     1     1      544            545        -1     1004
#> 5  2013     1     1      554            600        -6      812
#> 6  2013     1     1      554            558        -4      740
#> # ... with 3.326e+05 more rows, and 12 more variables:
#> #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
#> #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#> #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

5.7 Grouped mutates (and filters)

- Standardise to compute per group metrics: 그룹별로 표준화



```
popular_dests %>%
  filter(arr_delay > 0) %>%
  mutate(prop_delay = arr_delay / sum(arr_delay)) %>%
  select(year:day, dest, arr_delay, prop_delay)
#> Source: local data frame [131,106 x 6]
#> Groups: dest [77]
#>
#>     year month   day  dest arr_delay prop_delay
#>    <int> <int> <int> <chr>     <dbl>      <dbl>
#> 1  2013     1     1   IAH        11   1.11e-04
#> 2  2013     1     1   IAH        20   2.01e-04
#> 3  2013     1     1   MIA        33   2.35e-04
#> 4  2013     1     1   ORD        12   4.24e-05
#> 5  2013     1     1   FLL        19   9.38e-05
#> 6  2013     1     1   ORD         8   2.83e-05
#> # ... with 1.311e+05 more rows
```

5.7 Grouped mutates (and filters)

- Exercises

1. Refer back to the table of useful mutate and filtering functions. Describe how each operation changes when you combine it with grouping.

2. Which plane (tailnum) has the worst on-time record?

3. What time of day should you fly if you want to avoid delays as much as possible?

4. For each destination, compute the total minutes of delay. For each, flight, compute the proportion of the total delay for its destination.

5. Delays are typically temporally correlated: even once the problem that caused the initial delay has been resolved, later flights are delayed to allow earlier flights to leave. Using lag() explore how the delay of a flight is related to the delay of the immediately preceding flight.

6. Look at each destination. Can you find flights that are suspiciously fast? (i.e. flights that represent a potential data entry error). Compute the air time a flight relative to the shortest flight to that destination. Which flights were most delayed in the air?

7. Find all destinations that are flown by at least two carriers. Use that information to rank the carriers.

Thank you