R프로그램 - 데이터 핸들링

Jinseog Kim
Dongguk University
jskim1986@gmail.com

2018-03-13

Contents

1 데이터 핸들링 - 기본

1.1 벡터의 핸들링

- subscripting, indexing

```
x <- sample(1:10, 15, rep=T)
x
```

```
##  [1]  8  6  6  1  1  9  1  3  3  3  4  4  4 10  8
```

```
others <- (x > 1)
others
```

```
##  [1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [12]  TRUE  TRUE  TRUE  TRUE
```

```
x[others]
```

```
##  [1]  8  6  6  9  3  3  3  4  4  4 10  8
```

```r
ind <- which(x > 1)
ind
```

```
##  [1]  1  2  3  6  8  9 10 11 12 13 14 15
```

```r
x[ind]
```

```
##  [1]  8  6  6  9  3  3  3  4  4  4 10  8
```

```r
x[!others]
```

```
## [1] 1 1 1
```

```r
x[-ind]
```

```
## [1] 1 1 1
```

1.2 데이터프레임 (data.frame)의 핸들링

- 예제 데이터(USArrests)
  - This data set contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.
  - 변수 설명
    1. Murder: numeric Murder arrests (per 100,000)
    2. Assault: numeric Assault arrests (per 100,000)
    3. UrbanPop: numeric Percent urban population
    4. Rape: numeric Rape arrests (per 100,000)

```
head(USArrests,3)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
```

1. subscripting, indexing
   - Numeric subscripts

```r
# Top 5 states with high murder rate
nidx <- order(USArrests$Murder, decreasing=T)[1:5]
nidx
```

```
## [1] 10 24  9 18 40
```

```r
USArrests[nidx,]
```

```
##                Murder Assault UrbanPop Rape
## Georgia          17.4     211       60 25.8
## Mississippi      16.1     259       44 17.1
## Florida          15.4     335       80 31.9
## Louisiana        15.4     249       66 22.2
## South Carolina   14.4     279       48 22.5
```

- Logical subscripts

```r
lidx <- (USArrests$Murder
         < quantile(USArrests$Murder, 0.1))
head(lidx, 10)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
USArrests[lidx,]
```

```
##               Murder Assault UrbanPop Rape
## Iowa             2.2      56       57 11.3
## Maine            2.1      83       51  7.8
## New Hampshire    2.1      57       56  9.5
## North Dakota     0.8      45       44  7.3
## Vermont          2.2      48       32 11.2
```

2. 행/열 추출 : subset

```
subset(USArrests, UrbanPop > 85)
```

```
##              Murder Assault UrbanPop Rape
## California      9.0     276       91 40.6
## New Jersey      7.4     159       89 18.8
## New York       11.1     254       86 26.1
## Rhode Island    3.4     174       87  8.3
```

```r
subset(USArrests, UrbanPop < 40 &  Murder < 10,
     select = c(Assault, Rape))
```

```
##                Assault Rape
## Vermont             48 11.2
## West Virginia       81  9.3
```

3. 데이터 결합 : merge

```r
authors <- data.frame(
    surname = c("Tukey", "Venables", "Tierney", "Ripley", "McNeil"),
    nationality = c("US", "Australia", "US", "UK", "Australia")   )
books <- data.frame(
    name = c("Tukey", "Venables", "Tierney",
             "Ripley", "Ripley", "McNeil", "R Core"),
    title = c("Exploratory Data Analysis",
              "Modern Applied Statistics ...",
              "LISP-STAT",
              "Spatial Statistics", "Stochastic Simulation",
              "Interactive Data Analysis",
              "An Introduction to R"))
authors
```

```
##    surname nationality
## 1    Tukey          US
## 2 Venables   Australia
## 3  Tierney          US
## 4   Ripley          UK
## 5   McNeil   Australia
```

```
books
```

```
##       name                           title
## 1    Tukey     Exploratory Data Analysis
## 2 Venables Modern Applied Statistics ...
## 3  Tierney                     LISP-STAT
## 4   Ripley            Spatial Statistics
## 5   Ripley         Stochastic Simulation
## 6   McNeil     Interactive Data Analysis
## 7   R Core           An Introduction to R
```

 authors의 “surname”과 authors, books의 “name”을 키로 결합

```r
m1 <- merge(authors, books, by.x = "surname", by.y = "name")
m1
```

```
##    surname nationality                         title
## 1   McNeil   Australia     Interactive Data Analysis
## 2   Ripley          UK             Spatial Statistics
## 3   Ripley          UK         Stochastic Simulation
## 4  Tierney          US                      LISP-STAT
## 5    Tukey          US     Exploratory Data Analysis
## 6 Venables   Australia Modern Applied Statistics ...
```

4. 데이터 요약: aggregate

    Splits the data into subsets, computes summary statistics for each

```r
aggregate(요약변수, list(그룹화변수), 요약함수)
aggregate(x = testDF, by = list(fby1, fby2), FUN = "mean")
```

```r
aggregate(Sepal.Length~Species, data=iris, FUN=mean)
```

```
##      Species Sepal.Length
## 1     setosa        5.006
## 2 versicolor        5.936
## 3  virginica        6.588
```

5. 벡터화에 의한 데이터 요약 :

```
apply(array, MARGIN, FUN, ...)
lapply(vector or list, FUN, ...)
```

1. apply: Array with Margins

```
apply(iris[, 1:4], 2, mean)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```
apply(iris[, 1:4], 1, sum)[1:10]
```

```
##  [1] 10.2  9.5  9.4  9.4 10.2 11.4  9.7 10.1  8.9  9.6
```

2. lapply(): List or Vector

```r
lapply(iris[1:4], mean) # lapply(iris[,1:4], mean)
```

```
## $Sepal.Length
## [1] 5.843333
##
## $Sepal.Width
## [1] 3.057333
##
## $Petal.Length
## [1] 3.758
##
## $Petal.Width
## [1] 1.199333
```

```r
lapply(1:4, function(i) mean(iris[,i]))
```

```
## [[1]]
## [1] 5.843333
##
## [[2]]
## [1] 3.057333
##
```

```
## [[3]]
## [1] 3.758
##
## [[4]]
## [1] 1.199333
```

2 데이터 처리를 위한 R 패키지

- 데이터 처리 과정
  - 데이터의 전처리
  - 변환
  - 필터링
- 데이터 전처리를 위한 R 패키지
  - data.table : 대용량 자료의 빠른 처리
  - dplyr: Hadley Wickham가 작성, C++로 작성
  - plyr: R로 작성, 처리 속도가 느림
  - reshape2
  - sqldf: SQL구문을 이용한 데이터 조작

3 data.table 패키지

- 기본 객체인 데이터프레임의 연산에 비해 50~100배 속도 향상
- fread: fast and friendly file reader
- setkey(DT,col1,col2): a fast primary ordered index

3.1 data.table 객체생성

```
data.table(...)
```

```
library(data.table)
data.table(x=c("b","b","a","a"),v=rnorm(4))
```

```
##    x          v
## 1: b -0.6037000
## 2: b -1.6370087
## 3: a -0.7218597
## 4: a  0.5528330
```

```
system.time(DF <- fread("apt.csv"))
```

```
##    user  system elapsed
##   0.008   0.000   0.010
```

```
system.time(df <- read.csv("apt.csv"))
```

```
##    user  system elapsed
##   0.076   0.000   0.075
```

3.2 data.frame으로부터 data.table 객체생성

```
CARS <- data.table(cars)
head(CARS)
```

```
##    speed dist
## 1:     4    2
## 2:     4   10
## 3:     7    4
```

```
## 4:     7   22
## 5:     8   16
## 6:     9   10
```

3.3 table 목록

```
tables()
```

```
##      NAME  NROW NCOL MB
## [1,] CARS    50    2  1
## [2,] DF   5,222   12  1
##      COLS
## [1,] speed,dist
## [2,] 시군구,본번,부번,단지명,전월세구분,전용면적(㎡),계약일,보증금(만원),월세(만원),층,건축년도
##      KEY
## [1,]
## [2,]
## Total: 2MB
```

3.4 select columns, rows

```
DF[,c(1,3, 5), with=F] # 1,3,5번째 컬럼
```

```
##                         시군구 부번 전월세구분
##    1: 서울특별시 강남구 개포동    0       월세
##    2: 서울특별시 강남구 개포동    0       전세
##    3: 서울특별시 강남구 개포동    0       전세
##    4: 서울특별시 강남구 개포동    0       전세
##    5: 서울특별시 강남구 개포동    0       전세
##   ---
## 5218: 서울특별시 중랑구 중화동    0       전세
## 5219: 서울특별시 중랑구 중화동    0       전세
## 5220: 서울특별시 중랑구 중화동    0       전세
## 5221: 서울특별시 중랑구 중화동    0       전세
## 5222: 서울특별시 중랑구 중화동    0       전세
```

```
DF[1:5] # 첫 5행
```

```
##                         시군구 본번 부번                 단지명 전월세구분
```

```
## 1: 서울특별시 강남구 개포동  649    0                    경남1       월세
## 2: 서울특별시 강남구 개포동  649    0                    경남1       전세
## 3: 서울특별시 강남구 개포동   12    0 대청3단지(6개동 일반분양)       전세
## 4: 서울특별시 강남구 개포동   12    0 대청3단지(6개동 일반분양)       전세
## 5: 서울특별시 강남구 개포동   12    0 대청3단지(6개동 일반분양)       전세
##    전용면적(㎡) 계약일 보증금(만원) 월세(만원) 층 건축년도      도로명
## 1:       166.48   1~10       30,000        230 12     1984      언주로
## 2:        96.98   1~10       65,000          0  7     1984      언주로
## 3:        51.12   1~10       38,000          0 10     1992 개포로109길
## 4:        39.53  11~20       30,000          0  5     1992 개포로109길
## 5:        39.53   1~10       30,000          0  1     1992 개포로109길
```

3.5 setkey 키변수 지정

```
setkey(DF, 시군구)
DF
```

```
##                       시군구 본번 부번                  단지명
##    1: 서울특별시 강남구 개포동  649    0                   경남1
##    2: 서울특별시 강남구 개포동  649    0                   경남1
```

```
##    3: 서울특별시 강남구 개포동   12    0 대청3단지(6개동 일반분양)
##    4: 서울특별시 강남구 개포동   12    0 대청3단지(6개동 일반분양)
##    5: 서울특별시 강남구 개포동   12    0 대청3단지(6개동 일반분양)
##   ---
## 5218: 서울특별시 중랑구 중화동  450    0                 한신1차
## 5219: 서울특별시 중랑구 중화동  450    0                 한신1차
## 5220: 서울특별시 중랑구 중화동  450    0                 한신1차
## 5221: 서울특별시 중랑구 중화동  450    0                 한신1차
## 5222: 서울특별시 중랑구 중화동  450    0                 한신1차
##       전월세구분 전용면적(㎡) 계약일 보증금(만원) 월세(만원) 층 건축년도
##    1:       월세       166.48  1~10       30,000        230 12     1984
##    2:       전세        96.98  1~10       65,000          0  7     1984
##    3:       전세        51.12  1~10       38,000          0 10     1992
##    4:       전세        39.53 11~20       30,000          0  5     1992
##    5:       전세        39.53  1~10       30,000          0  1     1992
##   ---
## 5218:       전세        84.87 11~20       32,000          0  4     1997
## 5219:       전세        50.37 11~20       23,000          0 13     1997
## 5220:       전세        50.37 11~20       24,000          0 21     1997
## 5221:       전세        59.76  1~10       28,000          0 19     1997
## 5222:       전세        59.76 11~20       28,500          0 24     1997
##          도로명
```

```
##    1:      언주로
##    2:      언주로
##    3: 개포로109길
##    4: 개포로109길
##    5: 개포로109길
##   ---
## 5218:      동일로
## 5219:      동일로
## 5220:      동일로
## 5221:      동일로
## 5222:      동일로
```

3.6 Group summary

- dt[,연산식, by=그룹 variable]

```
Iris <- data.table(iris)
names(Iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

```r
# Species에 따라 Petal.Width의 평균계산
Iris[, mean(Petal.Width), by=Species]
```

```
##       Species    V1
## 1:     setosa 0.246
## 2: versicolor 1.326
## 3:  virginica 2.026
```

```r
# Species에 따라 다른 모든 변수들의 평균계산
Iris[, lapply(.SD, mean), by=Species] # .SD는 모든 컬럼
```

```
##       Species Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1:     setosa        5.006       3.428        1.462       0.246
## 2: versicolor        5.936       2.770        4.260       1.326
## 3:  virginica        6.588       2.974        5.552       2.026
```

4 dplyr 패키지

- dplyr 패키지는 데이터 프레임을 핸들링하는 함수군으로 구성
- 다음 형식의 데이터 객체에도 이용 가능
  - data.table : data.table 패키지와 함께 사용
  - 각종 데이터베이스 : 현재 MySQL, PostgreSQL, SQLite, BigQuery를 지원

4.1 패키지의 설치 및 로드

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

4.2 dplyr 패키지의 함수

Table 1: dplyr 함수

함수명	내용	base패키지 함수
filter()	행 추출	subset()

함수명	내용	base패키지 함수
select()	열 추출	data[, c("V1", "V2")]
mutate()	열 추가	transform()
arrange()	정렬	order(), sort()
distinct()	unique 행 추출	unique()
summarise()	집계	aggregate()
sample_n()	랜덤샘플링	sample()
sample_frac()		
group_by()	그룹별 집계	
tbl_df()	데이터프레임의 변환	
as_data_frame()	데이터프레임의 변환	as.data.frame()

4.3 예제 데이터 : New York flights data

```
# install.packages("nycflights13", repos="http://cran.nexr.com/");
# install.packages("sqldf", repos="http://cran.nexr.com/");
library(nycflights13) # 실습용 데이터 로드
```

- 2013년 미국 New York에서 출발하는 항공기의 이착륙 기록 data.frame
- 레코드 수 : 336776 / 변수: 19개 변수

Table 2: Variables

변수	설명
year,month,day	date of departure
dep_time,arr_time	Actual departure and arrival times, local tz
sched_dep_time,sched_arr_time	Scheduled departure and arrival times, local tz
dep_delay,arr_delay	Departure and arrival delays, in minutes
hour,minute	Time of scheduled departure broken into hour and minutes.
carrier	Two letter carrier abbreviation. See airlines to get name
tailnum	Plane tail number
flight	Flight number
origin,dest	Origin and destination
air_time	Amount of time spent in the air
distance	Distance flown
time_hour	Scheduled date and hour of the flight as a POSIXct date

- Basic information & preview

```
library(nycflights13)
dim(flights)
```

```
## [1] 336776     19
```

```
head(flights)
```

```
## # A tibble: 6 x 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 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>
```

```
flights <- tbl_df(flights)
flights[,1:8]
```

```
## # A tibble: 336,776 x 8
```

```
##     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
##  7  2013     1     1      555            600       -5.      913
##  8  2013     1     1      557            600       -3.      709
##  9  2013     1     1      557            600       -3.      838
## 10  2013     1     1      558            600       -2.      753
## # ... with 336,766 more rows, and 1 more variable: sched_arr_time <int>
```

4.4 dplyr::filter() row 추출

- filter(): 데이터 프레임에서 조건에 따른 레코드(row) 추출
  - AND: ‘,’ 또는 ‘&’
  - OR: ‘|’

```r
# 1월 1일 데이터 추출
filter(flights, month == 1, day == 1)
```

```
## # A tibble: 842 x 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
##  7  2013     1     1      555            600       -5.      913
##  8  2013     1     1      557            600       -3.      709
##  9  2013     1     1      557            600       -3.      838
## 10  2013     1     1      558            600       -2.      753
## # ... with 832 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>
```

```r
# equivalent to base operation:
#   flights[flights$month == 1 & flights$day == 1, ]
# 1월 혹은 2월 데이터 추출
filter(flights, month == 1 | month == 2)
```

```
## # A tibble: 51,955 x 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
##  7  2013     1     1      555            600       -5.      913
##  8  2013     1     1      557            600       -3.      709
##  9  2013     1     1      557            600       -3.      838
## 10  2013     1     1      558            600       -2.      753
## # ... with 51,945 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>
```

```r
# 행 index를 이용한 추출
slice(flights, 1:10)
```

```
## # A tibble: 10 x 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
##  7  2013     1     1      555            600       -5.      913
##  8  2013     1     1      557            600       -3.      709
##  9  2013     1     1      557            600       -3.      838
## 10  2013     1     1      558            600       -2.      753
## # ... with 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>
```

4.5 Extract distinct (unique) rows

```
distinct(flights, origin, dest)
```

```
## # A tibble: 224 x 2
##    origin dest
##    <chr>  <chr>
##  1 EWR    IAH
##  2 LGA    IAH
##  3 JFK    MIA
##  4 JFK    BQN
##  5 LGA    ATL
##  6 EWR    ORD
##  7 EWR    FLL
##  8 LGA    IAD
##  9 JFK    MCO
## 10 LGA    ORD
## # ... with 214 more rows
```

4.6 dlpyr::arrange()를 이용한 정렬

- 지정한 열을 기준으로 데이터를 정렬
  - desc(변수명) : 역순정렬

```
# year, month, day에 대한 정렬
arrange(flights, year, month, day)
```

```
## # A tibble: 336,776 x 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
##  7  2013     1     1      555            600       -5.      913
##  8  2013     1     1      557            600       -3.      709
##  9  2013     1     1      557            600       -3.      838
## 10  2013     1     1      558            600       -2.      753
```

```
## # ... with 336,766 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>
```

```
# arr_delay에 대한 역순정렬
arrange(flights, desc(arr_delay))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     9      641            900     1301.     1242
##  2  2013     6    15     1432           1935     1137.     1607
##  3  2013     1    10     1121           1635     1126.     1239
##  4  2013     9    20     1139           1845     1014.     1457
##  5  2013     7    22      845           1600     1005.     1044
##  6  2013     4    10     1100           1900      960.     1342
##  7  2013     3    17     2321            810      911.      135
##  8  2013     7    22     2257            759      898.      121
##  9  2013    12     5      756           1700      896.     1058
## 10  2013     5     3     1133           2055      878.     1250
## # ... with 336,766 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>
```

4.7 dplyr::select()를 이용한 열의 조작

1. ‘,’, ‘:’: column 추출

```
# year, month, day 열을 추출
select(flights, year, month, day)
```

```
## # A tibble: 336,776 x 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
##  7  2013     1     1
```

```
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # ... with 336,766 more rows
```

```
# year부터 day까지 추출
select(flights, year:day)
```

```
## # A tibble: 336,776 x 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
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # ... with 336,766 more rows
```

2. '-': column 제외

```r
# year 부터 flight까지 열 제외
select(flights, -(year:flight))
```

```
## # A tibble: 336,776 x 8
##    tailnum origin dest  air_time distance  hour minute time_hour
##    <chr>   <chr>  <chr>    <dbl>    <dbl> <dbl>  <dbl> <dttm>
##  1 N14228  EWR    IAH       227.    1400.    5.    15. 2013-01-01 05:00:00
##  2 N24211  LGA    IAH       227.    1416.    5.    29. 2013-01-01 05:00:00
##  3 N619AA  JFK    MIA       160.    1089.    5.    40. 2013-01-01 05:00:00
##  4 N804JB  JFK    BQN       183.    1576.    5.    45. 2013-01-01 05:00:00
##  5 N668DN  LGA    ATL       116.     762.    6.     0. 2013-01-01 06:00:00
##  6 N39463  EWR    ORD       150.     719.    5.    58. 2013-01-01 05:00:00
##  7 N516JB  EWR    FLL       158.    1065.    6.     0. 2013-01-01 06:00:00
##  8 N829AS  LGA    IAD        53.     229.    6.     0. 2013-01-01 06:00:00
##  9 N593JB  JFK    MCO       140.     944.    6.     0. 2013-01-01 06:00:00
## 10 N3ALAA  LGA    ORD       138.     733.    6.     0. 2013-01-01 06:00:00
## # ... with 336,766 more rows
```

4.8 dplyr::mutate()를 이용한 열 추가

```
transmute(flights, gain = arr_delay - dep_delay, speed = distance / air_time * 60)
```

```
## # A tibble: 336,776 x 2
##     gain speed
##    <dbl> <dbl>
##  1    9.  370.
##  2   16.  374.
##  3   31.  408.
##  4  -17.  517.
##  5  -19.  394.
##  6   16.  288.
##  7   24.  404.
##  8  -11.  259.
##  9   -5.  405.
## 10   10.  319.
## # ... with 336,766 more rows
```

4.9 dplyr::transmute(): 추가된 열만 저장

```
transmute(flights, gain = arr_delay - dep_delay, speed = distance / air_time * 60)
```

```
## # A tibble: 336,776 x 2
##     gain speed
##    <dbl> <dbl>
##  1    9.  370.
##  2   16.  374.
##  3   31.  408.
##  4  -17.  517.
##  5  -19.  394.
##  6   16.  288.
##  7   24.  404.
##  8  -11.  259.
##  9   -5.  405.
## 10   10.  319.
## # ... with 336,766 more rows
```

4.10 dplyr::rename(): 컬럼명 재지정

 year를 Year로 변경

```
rename(flights, Year = year)
```

```
## # A tibble: 336,776 x 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
##  7  2013     1     1      555            600       -5.      913
##  8  2013     1     1      557            600       -3.      709
##  9  2013     1     1      557            600       -3.      838
## 10  2013     1     1      558            600       -2.      753
## # ... with 336,766 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>
```

4.11 dplyr::summarise()를 이용한 집계

- summarise()
  - mean(), sd(), var(), median() 등을 이용 기초 통계량
  - 결과는 데이터프레임

```
# 평균 출발지연시간 계산
summarise(flights,  delay = mean(dep_delay, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   delay
##   <dbl>
## 1  12.6
```

4.12 Randomly sample rows with sample_n() and sample_frac()

- sample_n(): fixed number
- sample_frac(): fraction

```r
sample_n(flights, 10)
```

```
## # A tibble: 10 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     4    16     1420           1430      -10.     1617
##  2  2013     8     6     1706           1710       -4.     2000
##  3  2013    11    16      821            815        6.     1114
##  4  2013    12     2     1843           1845       -2.     2127
##  5  2013    11     3     1932           1932        0.     2204
##  6  2013     9    11       NA           1820       NA        NA
##  7  2013    11     1      604            610       -6.      855
##  8  2013    10    19     1244           1247       -3.     1542
##  9  2013    11     2      841            845       -4.      930
## 10  2013     2     4     1156           1150        6.       NA
## # ... with 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>
```

4.13 함수 group_by()를 이용한 그룹화

 group_by(): 열의 수준(level)별 그룹화 - 그룹별 인덱스화

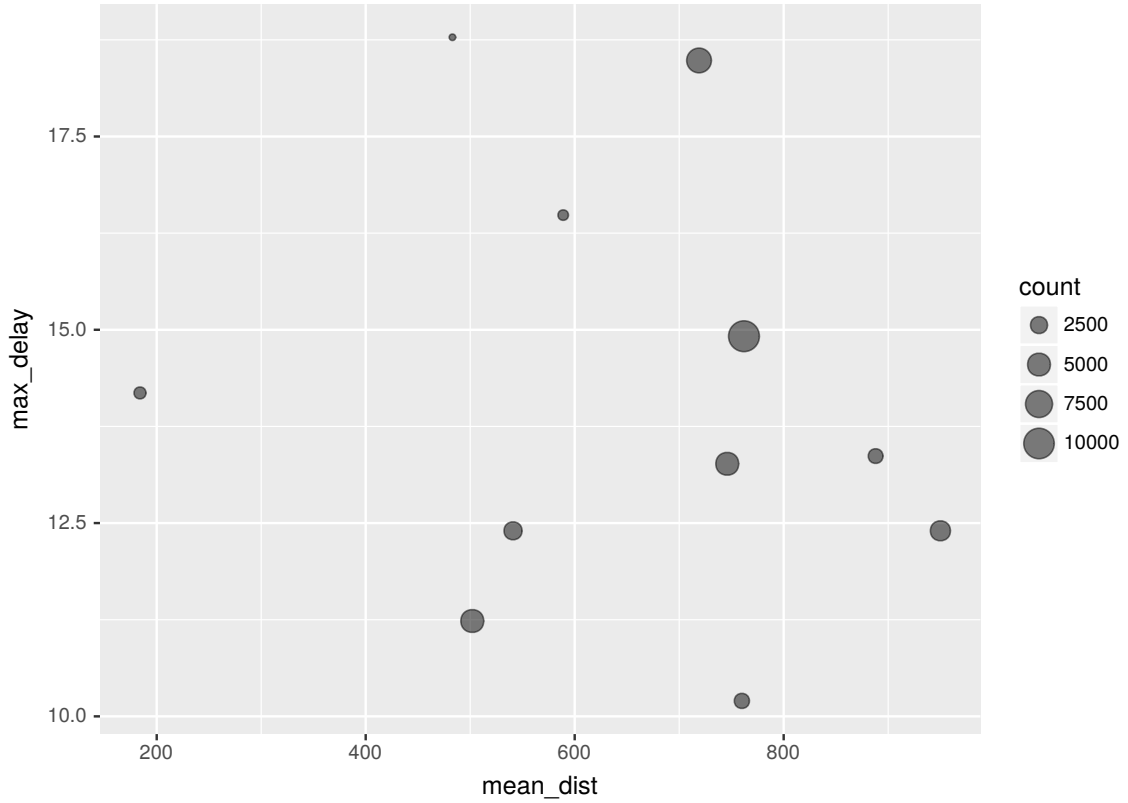
 비행편수 200편 이상, 평균거리 1,000마일 이하, 최대 연착시간이 10분이상이 되는 OD구간을 구하시오.

```
# OD그룹화
od <- group_by(flights, origin, dest)
# OD그룹별 비행편수, 평균거리, 최대연착시간
delay <- summarise(od,
  count = n(),
  mean_dist = mean(distance, na.rm = TRUE),
  max_delay = max(arr_delay, na.rm = TRUE)/60)
# 비행편수 200 이상, 거리 1,000마일 이하, 최대 연착시간이 10분이상이 되는 OD구간
(delay2 <- filter(delay, count > 200, mean_dist <= 1000, max_delay >= 10))
```

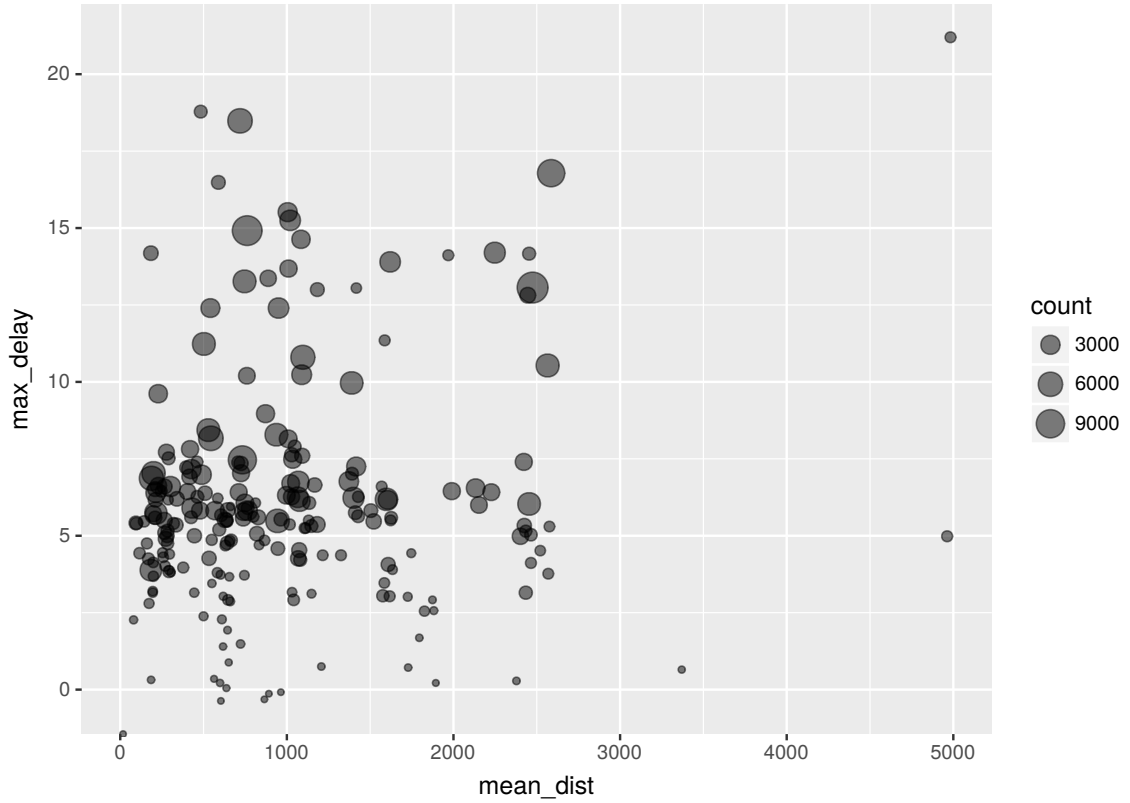
```
## # A tibble: 11 x 5
## # Groups:   origin [3]
##    origin dest  count mean_dist max_delay
##    <chr>  <chr> <int>     <dbl>     <dbl>
##  1 EWR    ATL    5022      746.      13.3
##  2 EWR    ORD    6100      719.      18.5
```

```
##  3 JFK    ATL    1930      760.      10.2
##  4 JFK    BWI    1221      184.      14.2
##  5 JFK    CLT    2870      541.      12.4
##  6 JFK    CMH     742      483.      18.8
##  7 JFK    CVG     997      589.      16.5
##  8 LGA    ATL   10263      762.      14.9
##  9 LGA    DTW    5040      502.      11.2
## 10 LGA    MCO    3677      950.      12.4
## 11 LGA    STL    1822      888.      13.4
```

```
library(ggplot2)
ggplot(delay2, aes(mean_dist, max_delay)) + geom_point(aes(size = count), alpha = 1/2)
```




```
ggplot(delay, aes(mean_dist, max_delay)) + geom_point(aes(size = count), alpha = 1/2)
```



4.14 단계별 데이터 조작

1. `flights` → `a1` → `a2` → `a3` → `a4` : 각 단계별 절차

```r
# year, month, day의 수준별 그룹화
a1 <- group_by(flights, year, month, day)
# year:day, arr_delay, dep_delay 열 선택
a2 <- select(a1, year:day, arr_delay, dep_delay)
# 평균 도착지연, 출발지연시간 계산
a3 <- summarise(a2, arr = mean(arr_delay, na.rm = TRUE),
                dep = mean(dep_delay, na.rm = TRUE))
# 평균 도착지연, 출발지연시간이 30분 이상 추출
a4 <- filter(a3, arr > 30 | dep > 30)
```

2. dplyr::%>% : Piping data

 %>% : 각 단계별 절차를 연결하는 오퍼레이터

```r
flights %>% group_by(year, month, day) %>% summarise(arr = mean(arr_delay,
     na.rm = TRUE), dep = mean(dep_delay, na.rm = TRUE)) %>%
  filter(arr > 30 | dep > 30)
```

```
## # A tibble: 49 x 5
## # Groups:   year, month [11]
##     year month   day   arr   dep
##    <int> <int> <int> <dbl> <dbl>
```

```
##  1  2013     1    16  34.2  24.6
##  2  2013     1    31  32.6  28.7
##  3  2013     2    11  36.3  39.1
##  4  2013     2    27  31.3  37.8
##  5  2013     3     8  85.9  83.5
##  6  2013     3    18  41.3  30.1
##  7  2013     4    10  38.4  33.0
##  8  2013     4    12  36.0  34.8
##  9  2013     4    18  36.0  34.9
## 10  2013     4    19  47.9  46.1
## # ... with 39 more rows
```

5 reshape2를 이용한 데이터 변환

5.1 wide/long format

1. wide format: 데이터의 컬럼이 변수이고 컬럼의 값들이 해당 변수의 값을 나타내는 형식

```
library(reshape2)
head(mtcars) # wide format
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

2. long format: 데이터가 적은 수의 컬럼(보통 2~3)으로 이루어지며 각 컬럼은 ID, 변수 또는 해당 변수의 값을 나타내는 형식

```
##                 car variable value
```

```
## 1          Mazda RX4      mpg  21.0
## 2      Mazda RX4 Wag      mpg  21.0
## 3         Datsun 710      mpg  22.8
## 4     Hornet 4 Drive      mpg  21.4
## 5 Hornet Sportabout      mpg  18.7
## 6            Valiant      mpg  18.1
```

5.2 wide/long format의 변환

1. reshape2::melt(): wide format의 데이터를 long format으로 변환
2. reshape2::cast(): long format의 데이터를 wide format으로 변환

 melt()

```
mtcarsMelt <- melt(mtcars, id.vars = 'car')
head(mtcarsMelt)
```

```
##                 car variable value
## 1          Mazda RX4      mpg  21.0
## 2      Mazda RX4 Wag      mpg  21.0
## 3         Datsun 710      mpg  22.8
```

```
## 4    Hornet 4 Drive      mpg  21.4
## 5 Hornet Sportabout      mpg  18.7
## 6           Valiant      mpg  18.1
```

- dcast(long_format_data, id_var ~ 변수명_변수_in_long_format)

```
mtcarsCast <- dcast(mtcarsMelt, car ~ variable)
head(mtcarsCast)
```

```
##                  car  mpg cyl disp  hp drat    wt  qsec vs am gear carb
## 1        AMC Javelin 15.2   8  304 150 3.15 3.435 17.30  0  0    3    2
## 2 Cadillac Fleetwood 10.4   8  472 205 2.93 5.250 17.98  0  0    3    4
## 3         Camaro Z28 13.3   8  350 245 3.73 3.840 15.41  0  0    3    4
## 4  Chrysler Imperial 14.7   8  440 230 3.23 5.345 17.42  0  0    3    4
## 5         Datsun 710 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## 6   Dodge Challenger 15.5   8  318 150 2.76 3.520 16.87  0  0    3    2
```

6 sqldf를 이용한 데이터프레임 핸들링

 sqldf : R의 데이터프레임 객체를 SQL을 이용하여 조작하도록 지원하는 패키지

```
library(sqldf)## Load the package
sqldf('SELECT * FROM iris WHERE Species="setosa" AND "Sepal.Length" > 5.5')
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.8         4.0          1.2         0.2  setosa
## 2          5.7         4.4          1.5         0.4  setosa
## 3          5.7         3.8          1.7         0.3  setosa
```

```
sqldf('SELECT Species, COUNT(*) FROM iris GROUP BY Species')
```

```
##      Species COUNT(*)
## 1     setosa       50
## 2 versicolor       50
## 3  virginica       50
```

```
sqldf('SELECT Species, COUNT(*) AS N, AVG("Sepal.Width") AS SL
      FROM iris GROUP BY Species')
```

```
##      Species  N    SL
## 1     setosa 50 3.428
## 2 versicolor 50 2.770
## 3  virginica 50 2.974
```