PR12 - Data Wrangling

강현승

2022-11-21

1. Data Wrangling with tidyverse

Data Wrangling이란, 분석을 진행하기 위해 날것(raw)의 데이터를 분석에 적합한 형태로 정형화시키는 작업입니다.

R에서는 tidyverse 라는 패키지 생태계를 구성하고 있어서, 일관성 있고 쉬운 작업을 가능하게 합니다.

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

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ─
─
## ✔ ggplot2 3.4.0      ✔ purrr   0.3.5
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.3      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ─
─
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

2. `tidyr`

tidyr 은 Hadley Wickham이 만든 데이터의 포맷을 변경하기 위한 패키지

`tidyr` 의 주요함수

함수 설명

- `gather()` 데이터를 wide에서 long 포맷으로 변경
- `spread()` 데이터를 long에서 wide 포맷으로 변경
- `separate()` 단일 열(column)을 복수 열들로 분리
- `unite()` 복수 열(column)들을 단일 열로 결합

`tidyr` 실습 데이터: `cases` in `EDAWR`

Dataset to support the Expert Data Analysis with R: `EDAWR`

```
library(devtools)
```

```
## Loading required package: usethis
```

```
# devtools::install_github("rstudio/EDAWR", force = TRUE)
library(EDAWR)
```

```
##
## Attaching package: 'EDAWR'
```

```
## The following object is masked from 'package:dplyr':
##
##     storms
```

```
## The following objects are masked from 'package:tidyr':
##
##     population, who
```

```
head(cases)
```

```
##   country  2011  2012  2013
## 1      FR  7000  6900  7000
## 2      DE  5800  6000  6200
## 3      US 15000 14000 13000
```

```
head(pollution)
```

```
##       city  size amount
## 1 New York large     23
## 2 New York small     14
## 3   London large     22
## 4   London small     16
## 5  Beijing large    121
## 6  Beijing small     56
```

```
head(storms)
```

```
##     storm wind pressure       date
## 1 Alberto  110     1007 2000-08-03
## 2    Alex   45     1009 1998-07-27
## 3 Allison   65     1005 1995-06-03
## 4     Ana   40     1013 1997-06-30
## 5  Arlene   50     1010 1999-06-11
## 6  Arthur   45     1010 1996-06-17
```

2.1. gather() 함수

- wide 포맷의 데이터를 원하는 조건에 맞게 long 포맷으로 변환하는 함수
- `gather ( 데이터 , 키(Key), 값(Value), ... )`

- 키(Key) **새로운데이터**에 변수로 표시될 열이름
- 값(Value) **새로운데이터**에 변수의 값이 표시 될 열이름
- ... **원데이터**로 부터 모으기(gather)가 진행될 열들의 범위

```
gather(cases, Year, n, 2:4)
```

```
##   country Year     n
## 1      FR 2011  7000
## 2      DE 2011  5800
## 3      US 2011 15000
## 4      FR 2012  6900
## 5      DE 2012  6000
## 6      US 2012 14000
## 7      FR 2013  7000
## 8      DE 2013  6200
## 9      US 2013 13000
```

2.2. spread() 함수

- long 포맷의 데이터를 원하는 조건에 맞게 long 포맷으로 변환하는 함수
- separate(데이터, 키(Key), 값(Value), ~)
  - 키(Key) 복수개의 열로 spread될 기존 long 포맷의 열이름
  - 값(Value) 복수개의 열로 spread 되어 값이 될 기존 long 포맷의 열이름

```
spread(pollution, size, amount)
```

```
##       city large small
## 1  Beijing   121    56
## 2   London    22    16
## 3 New York    23    14
```

2.3. separate() 함수

- 하나의 열을 특정 조건에 따라 여러개의 열로 나누어 주는 함수입니다.
- separte(data, col, into, sep, ~)
  - col 조건에 따른 분할을 진행할 열이름
  - into 분할된 결과가 저장될 각 열들의 이름
  - sep 분할 조건

```
storms2 = separate(storms, date, c("year" , "month" , "day"), sep = "-")
storms2
```

```
## # A tibble: 6 × 6
##   storm    wind pressure year  month day
##   <chr>   <int>    <int> <chr> <chr> <chr>
## 1 Alberto   110     1007 2000  08    03
## 2 Alex       45     1009 1998  07    27
## 3 Allison    65     1005 1995  06    03
## 4 Ana        40     1013 1997  06    30
## 5 Arlene     50     1010 1999  06    11
## 6 Arthur     45     1010 1996  06    17
```

2.4. unite() 함수

- 여러개로 나누어진 열을 특정 조건에 따라 결합해주는 함수입니다.
- unite(data, col, , sep)
  - col 조건에 따라 결합된 결과가 저장될 열이름
  - ... 합쳐질 열이름들
  - sep 결합시 구분자

```
unite(storms2, "date" , year, month, day, sep = "-")
```

```
## # A tibble: 6 × 4
##   storm    wind pressure date
##   <chr>   <int>    <int> <chr>
## 1 Alberto   110     1007 2000-08-03
## 2 Alex       45     1009 1998-07-27
## 3 Allison    65     1005 1995-06-03
## 4 Ana        40     1013 1997-06-30
## 5 Arlene     50     1010 1999-06-11
## 6 Arthur     45     1010 1996-06-17
```

3. dplyr

- dplyr 은 Hadley Wickham이 만든 데이터 핸들링을 위한 패키지
- dplyr 은 C++로 작성되어 기존 데이터핸들링 패키지보다 빠른 데이터조작이 가능
- 각종 데이터베이스 지원(**MySQL**, **PostgreSQL**, **SQLite**, **BigQuery**)
- R의 기본문법과 프로그래밍능력만으로도 데이터의 조작이 가능하지만, dplyr 패키지를 활용하면 통일된 문법양식으로 데이터조작이 가능함
- 체인연산자을 지원함으로(%>%`) 앞부분의 연산결과를 뒤에 오는 함수의 입력값으로 사용할 수 있음

dplyr 의 주요함수

함수	설명	기존 함수
filter()	지정한 조건식에 맞는 데이터 추출	subset()
arrange()	정렬	order() , sort()
select()	열의 추출	data[, c("Year", , "Month")]

mutate()	열 추가	transform()
summarise()	집계	aggregate()

dplyr 실습데이터 nycflights13

미국 휴스턴에서 출발하는 모든 비행기의 이착륙기록

```
# install.packages("nycflights13") #해당 패키지에 데이터가 있음
library(nycflights13)
library(dplyr)
head(flights)
```

```
## # A tibble: 6 × 19
##    year month   day dep_time sched_dep…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrie
r
##   <int> <int> <int>    <int>       <int>   <dbl>   <int>   <int>   <dbl> <chr>
## 1  2013     1     1      517         515       2     830     819      11 UA
## 2  2013     1     1      533         529       4     850     830      20 UA
## 3  2013     1     1      542         540       2     923     850      33 AA
## 4  2013     1     1      544         545      -1    1004    1022     -18 B6
## 5  2013     1     1      554         600      -6     812     837     -25 DL
## 6  2013     1     1      554         558      -4     740     728      12 UA
## # … with 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>, and abbreviated variable names ¹sched_dep_time,
## #   ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

3.1 filter() 함수

- 데이터에서 원하는 조건에 따라 행을 추출하는 함수
- filter(데이터, 조건1 : 조건2) : 조건1 또는 조건2 둘중 한가지를 충족하는 데이터 추출
- filter(데이터, 조건1 & 조건2) : 조건1과 조건2 모두 충족하는 데이터 추출
- 조건을 작성할때쉼표 는 AND, | 는 OR와 같음

```
filter(flights, month == 1 | day == 1) #37198row
```

```
## # A tibble: 37,198 × 19
##     year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrie
r
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # … with 37,188 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

```
filter(flights, month == 1, day == 1) #842row
```

```
## # A tibble: 842 × 19
##     year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrie
r
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # … with 832 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

```
filter(flights,
       month == 1,
       day == 1,
       year == 2013) #832row
```

```
## # A tibble: 842 × 19
##     year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrie
r
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # … with 832 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

3.2 arrange() 함수

- 데이터를 원하는 조건에 따라 정렬해주는 함수
- arrange(데이터, 정렬기준컬럼1, 정렬기준컬럼2, 정렬기준컬럼3)
- 내림차순으로 정렬시 desc함수 사용 arrange(데이터, desc(정렬기준컬럼1))

```
arrange(flights, year, month, day) #ArrDelay, Month, Year 순으로 정렬
```

```
## # A tibble: 336,776 × 19
##     year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrie
r
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # … with 336,766 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

```
arrange(flights, desc(month)) #Month컬럼기준으로 내림차순으로 정렬
```

```
## # A tibble: 336,776 × 19
##     year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrie
r
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013    12     1        13       2359      14     446     445       1 B6
##  2  2013    12     1        17       2359      18     443     437       6 B6
##  3  2013    12     1       453        500      -7     636     651     -15 US
##  4  2013    12     1       520        515       5     749     808     -19 UA
##  5  2013    12     1       536        540      -4     845     850      -5 AA
##  6  2013    12     1       540        550     -10    1005    1027     -22 B6
##  7  2013    12     1       541        545      -4     734     755     -21 EV
##  8  2013    12     1       546        545       1     826     835      -9 UA
##  9  2013    12     1       549        600     -11     648     659     -11 US
## 10  2013    12     1       550        600     -10     825     854     -29 B6
## # … with 336,766 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

3.3 select() 함수

- select함수는 원하는 열(column)을 추출
- select(데이터, 컬럼1, 컬럼2, 컬럼3)
- select(데이터, 컬럼1: 컬럼3)
- 컬럼명을 변경할수 있음

```
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
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # … with 336,766 more rows
```

```
select(flights, year: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
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # … with 336,766 more rows
```

```
select(flights, -(year:day))
```

```
## # A tibble: 336,776 × 16
##    dep_t…¹ sched…² dep_d…³ arr_t…⁴ sched…⁵ arr_d…⁶ carrier flight tailnum origi
n
##      <int>   <int>   <dbl>   <int>   <int>   <dbl> <chr>    <int> <chr>   <chr>
##  1     517     515       2     830     819      11 UA        1545 N14228  EWR
##  2     533     529       4     850     830      20 UA        1714 N24211  LGA
##  3     542     540       2     923     850      33 AA        1141 N619AA  JFK
##  4     544     545      -1    1004    1022     -18 B6         725 N804JB  JFK
##  5     554     600      -6     812     837     -25 DL         461 N668DN  LGA
##  6     554     558      -4     740     728      12 UA        1696 N39463  EWR
##  7     555     600      -5     913     854      19 B6         507 N516JB  EWR
##  8     557     600      -3     709     723     -14 EV        5708 N829AS  LGA
##  9     557     600      -3     838     846      -8 B6          79 N593JB  JFK
## 10     558     600      -2     753     745       8 AA         301 N3ALAA  LGA
## # … with 336,766 more rows, 6 more variables: dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>, and abbreviated
## #   variable names ¹dep_time, ²sched_dep_time, ³dep_delay, ⁴arr_time,
## #   ⁵sched_arr_time, ⁶arr_delay
```

3.4 distinct() 함수

- 중복항목을 제외한 데이터를 확인 할 수 있음(unique함수와 동일)
- distinct(데이터, 컬럼명)

```
distinct(select(flights, tailnum))
```

```
## # A tibble: 4,044 × 1
##    tailnum
##    <chr>
##  1 N14228
##  2 N24211
##  3 N619AA
##  4 N804JB
##  5 N668DN
##  6 N39463
##  7 N516JB
##  8 N829AS
##  9 N593JB
## 10 N3ALAA
## # … with 4,034 more rows
```

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

```
## # A tibble: 224 × 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
```

3.5 mutate() 함수

- 기존 데이터 프레임에 새로운 열을 추가해줌
- 데이터프레임 내의 변수들을 활용해 새로운 변수를 만들때 효과적임
- 새로 생성한 변수를 해당 함수내에서 바로 활용이 가능

```
#arr_delay - dep_delay값으로 gain컬럼 추가
mutate(flights, gain = arr_delay - dep_delay)
```

```
## # A tibble: 336,776 × 20
##     year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrie
r
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # … with 336,766 more rows, 10 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, gain <dbl>, and abbreviated variable names
## #   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

```
#gain컬럼을 만드는 동시에 gain컬럼을 이용해 다른 변수를 생성가능
mutate(flights,
       gain = arr_delay - dep_delay,
       gain_per_hour = gain / (air_time / 60))
```

```
## # A tibble: 336,776 × 21
##     year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrie
r
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # … with 336,766 more rows, 11 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, gain <dbl>, gain_per_hour <dbl>, and
## #   abbreviated variable names ¹sched_dep_time, ²dep_delay, ³arr_time,
## #   ⁴sched_arr_time, ⁵arr_delay
```

3.6 summarise() 함수

- mean(), sd(), var() , median()함수를 활용해 기술통계량을 확인
- 결과를 데이터프레임으로 반환함

```
summarise(flights, delay = mean(dep_delay, na.rm = TRUE))
```

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

3.7 group_by() 함수

- 변수의 레벨에 따라 자료를 그룹화해줌
- 그룹에 따른 수치자료를 산출하고 싶을때 편리함
- summarize함수와 함께 사용시 aggregate함수와 같은 기능
- ex)직급에 따른 평균 연봉과 사용가능한 연차일수(휴가)를 구하고 싶을 때

```
#비행기별로 그룹만들기
by_tailnum = group_by(flights, tailnum) #비행기별로 그룹만들기

#비행기별 비행회수, 비행거리평균, 연착시간평균 산출
delay = summarise(
  by_tailnum,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE)
)

#회수가 20회이상 , 거리가 2000이하인 비행기만 추출
delay = filter(delay, count > 20, dist < 2000)
```

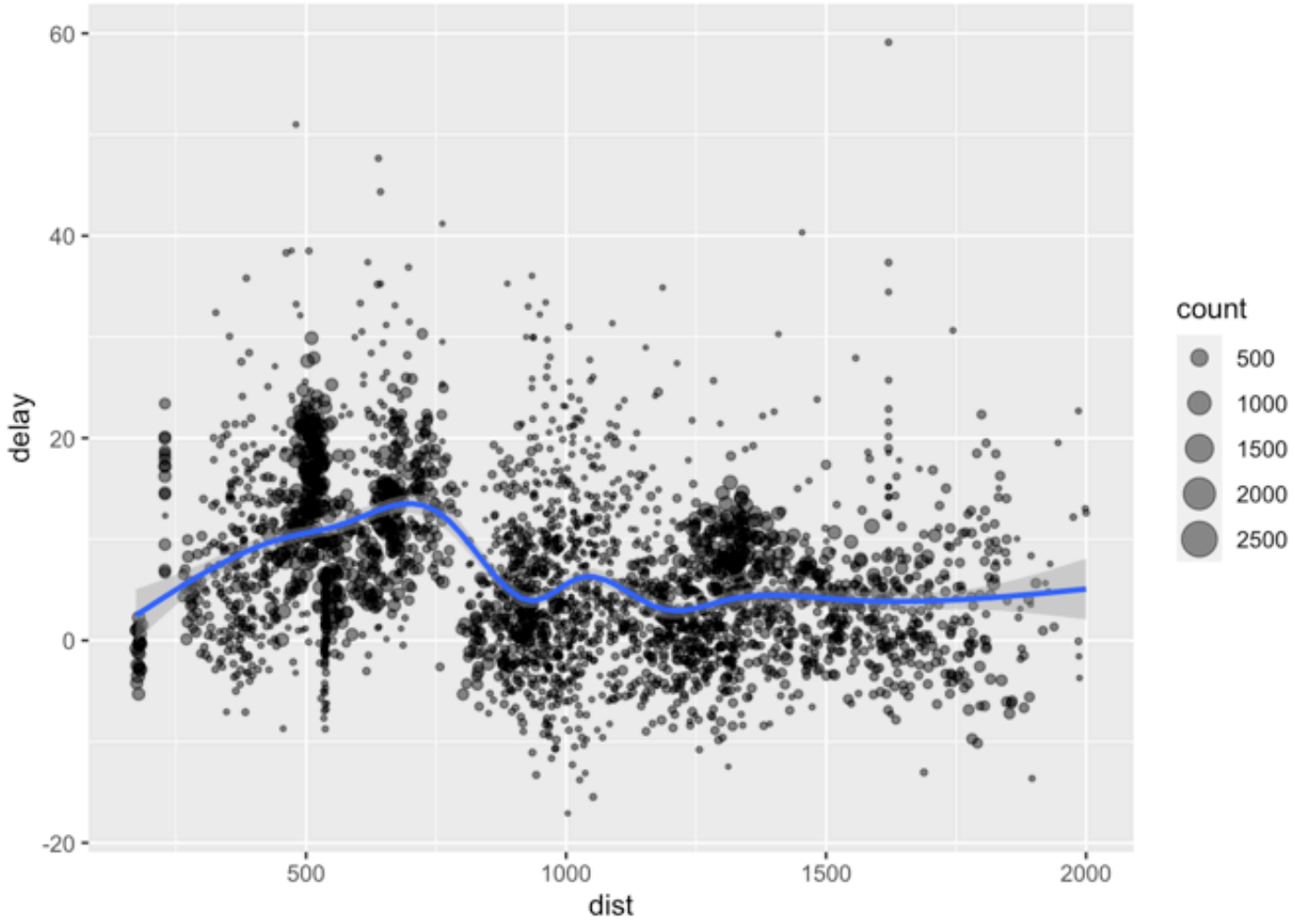
- 위에서 만든 delay데이터로 시각화

```
library(ggplot2)
ggplot(delay, aes(dist, delay)) +
  geom_point(aes(size = count), alpha = 1 / 2) +
  geom_smooth() +
  scale_size_area()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



3.8. join() 함수

- join(x, y) 또는 join(x, y, by="기준열") 형태
- 조인의 기준이 되는 단일 컬럼이 존재하는 경우 별도 by인수를 지정하지 않아도됨
- 단일 칼럼이 존재하지 않는 경우 by=c(기준열1 = 기준열2)와 같이 설정을 해주어야 함.
- 조인의 기준이 되는 컬럼이 여러개이거나, 여러가지 컬럼을 동시에 활용해야하는 경우 by인수를 사용

```
#join 실습 데이터 생성
superheroes = "
name, alignment, gender, publisher
Magneto, bad, male, Marvel
Storm, good, female, Marvel
Mystique, bad, female, Marvel
Batman, good, male, DC
Joker, bad , male, DC
Catwoman, bad, female, DC
Hellboy, good, male, Dark Horse Comics
"

publishers = "
publisher, yr_founded
DC, 1934
Marvel, 1939
Image, 1992
"
superheroes = read_csv(superheroes, trim_ws = TRUE, skip = 1)
```

```
## Rows: 7 Columns: 4
## ── Column specification ────────────────────────────────────────────────────
─
## Delimiter: ","
## chr (4): name, alignment, gender, publisher
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this messag
e.
```

```
publishers = read_csv(publishers, trim_ws = TRUE, skip = 1)
```

```
## Rows: 3 Columns: 2
## ── Column specification ────────────────────────────────────────────────────
─
## Delimiter: ","
## chr (1): publisher
## dbl (1): yr_founded
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this messag
e.
```

- inner_join, left_join, full_join, anti_join, semi_join 각각의 출력값확인하기

```
inner_join(superheroes, publishers) #X, Y의 교집합
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 6 × 5
##   name     alignment gender publisher yr_founded
##   <chr>    <chr>     <chr>  <chr>           <dbl>
## 1 Magneto  bad       male   Marvel           1939
## 2 Storm    good      female Marvel           1939
## 3 Mystique bad       female Marvel           1939
## 4 Batman   good      male   DC               1934
## 5 Joker    bad       male   DC               1934
## 6 Catwoman bad       female DC               1934
```

```
left_join(superheroes, publishers) #x기준 (왼쪽)으로 머징
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 7 × 5
##   name     alignment gender publisher         yr_founded
##   <chr>    <chr>     <chr>  <chr>                  <dbl>
## 1 Magneto  bad       male   Marvel                  1939
## 2 Storm    good      female Marvel                  1939
## 3 Mystique bad       female Marvel                  1939
## 4 Batman   good      male   DC                      1934
## 5 Joker    bad       male   DC                      1934
## 6 Catwoman bad       female DC                      1934
## 7 Hellboy  good      male   Dark Horse Comics         NA
```

```
full_join(superheroes, publishers) #x, y의 합집합
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 8 × 5
##   name     alignment gender publisher         yr_founded
##   <chr>    <chr>     <chr>  <chr>                  <dbl>
## 1 Magneto  bad       male   Marvel                  1939
## 2 Storm    good      female Marvel                  1939
## 3 Mystique bad       female Marvel                  1939
## 4 Batman   good      male   DC                      1934
## 5 Joker    bad       male   DC                      1934
## 6 Catwoman bad       female DC                      1934
## 7 Hellboy  good      male   Dark Horse Comics         NA
## 8 <NA>     <NA>      <NA>   Image                   1992
```

```
anti_join(superheroes, publishers) #x의 컬럼만 유지하여 머징
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 1 × 4
##   name    alignment gender publisher
##   <chr>   <chr>     <chr>  <chr>
## 1 Hellboy good      male   Dark Horse Comics
```

```
semi_join(superheroes, publishers) #Y의 여집합
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 6 × 4
##   name     alignment gender publisher
##   <chr>    <chr>     <chr>  <chr>
## 1 Magneto  bad       male   Marvel
## 2 Storm    good      female Marvel
## 3 Mystique bad       female Marvel
## 4 Batman   good      male   DC
## 5 Joker    bad       male   DC
## 6 Catwoman bad       female DC
```

4. magrittr

- magrittr 패키지는 연산자(operator)들의 집합들을 제공합니다.
- 데이터 연산을 왼쪽에서 오른쪽 순서로 구조화,
- nested 함수 호출을 피함,
- 지역 변수 및 함수의 정의의 필요성을 최소화,
- 연산 순서 내에서 어디서나 추가 step을 만들 수 있음
- f(x)를 X %>% f()로 대체할 수 있음
- 이 연산자가 main operator(chaining)인데 해당 기능이 의미 없이 보이시겠지만 여러가지 기능을 결합할 때 그 이점이 더욱 명확해집니다.
- dplyr을 불러오면 자동으로 불러와지게 됩니다.

4.1 main operator (Chaining; %>%`)

- 여러단계의 함수나 연산을 연결하여 한번에 수행할 때 사용
- 앞의 함수의 결과는 바로 뒤에오는 함수의 입력값이 됨
- 데이터를 여러객체에 할당하지 않아도 되기때문에 메모리 관리에 유리함

체인연산 사용하지 않을때

```
a1 = group_by(flights, year, month, day)
a2 = select(a1, year:day, arr_delay)
a3 = summarise(a2, arr = mean(arr_delay, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

```
a4 = filter(a3, arr > 30)
a4
```

```
## # A tibble: 42 × 4
## # Groups:   year, month [11]
##     year month   day   arr
##    <int> <int> <int> <dbl>
##  1  2013     1    16  34.2
##  2  2013     1    31  32.6
##  3  2013     2    11  36.3
##  4  2013     2    27  31.3
##  5  2013     3     8  85.9
##  6  2013     3    18  41.3
##  7  2013     4    10  38.4
##  8  2013     4    12  36.0
##  9  2013     4    18  36.0
## 10  2013     4    19  47.9
## # … with 32 more rows
```

체인연산 사용했을때

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

```
## Adding missing grouping variables: `year`, `month`, `day`
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 42 × 4
## # Groups:   year, month [11]
##     year month   day   arr
##    <int> <int> <int> <dbl>
##  1  2013     1    16  34.2
##  2  2013     1    31  32.6
##  3  2013     2    11  36.3
##  4  2013     2    27  31.3
##  5  2013     3     8  85.9
##  6  2013     3    18  41.3
##  7  2013     4    10  38.4
##  8  2013     4    12  36.0
##  9  2013     4    18  36.0
## 10  2013     4    19  47.9
## # … with 32 more rows
```

4.2. . 의 역할

- `.` 의 역할에 대해서 알아봅시다.
- 일반적으로 %>% 연산자만 사용하시게 되면 제일 첫 인수에 자동으로 배정이 됩니다.

```
head(iris, 3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

```
iris %>% head(3) # = head(., 3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- 데이터를 넘겨줘야 할 인수의 위치가 첫번째가 아닐 경우 다음과 같은 에러를 확인할 수 있음
- `gsub()` 는 찾아 바꾸는 함수로써, 사용방법은 `gsub(찾을문자나 숫자, 바꿀 문자나 숫자, 데이터)`

```
a = c("bannananana", "an apple")
gsub("n", "l", a)
```

```
## [1] "ballalalala" "al apple"
```

```
a %>% gsub("n", "l")
```

```
## Warning in gsub(., "n", "l"): argument 'pattern' has length > 1 and only the
## first element will be used
```

```
## [1] "l"
```

- 이러한 상황에서, `.` 을 원하는 위치에 넣어주시면 해당 위치에 데이터가 넘어가게 됨
- `!!` 은 `magrittr` 나 `dplyr` 에만 속해 있는 것이 아니라 R의 base에 정해진 규칙으로 `.~cyl` 의 사용법과 같습니다.

```
gsub("n", "l", a)
```

```
## [1] "ballalalala" "al apple"
```

```
a %>% gsub("n", "l", .)
```

```
## [1] "ballalalala" "al apple"
```

4.3. Chaining 예제

4.3.1. mtcars aggregate

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
car_data =
  mtcars %>% #1
  subset(hp > 100) %>% #2
  aggregate(. ~ cyl, data = ., FUN = . %>% mean %>% round(2)) %>% #3
  transform(kpl = mpg %>% multiply_by(0.4251)) %>% #4
  print #5
```

```
##   cyl   mpg   disp     hp drat   wt  qsec   vs   am gear carb       kpl
## 1   4 25.90 108.05 111.00 3.94 2.15 17.75 1.00 1.00 4.50 2.00 11.010090
## 2   6 19.74 183.31 122.29 3.59 3.12 17.98 0.57 0.43 3.86 3.43  8.391474
## 3   8 15.10 353.10 209.21 3.23 4.00 16.77 0.00 0.14 3.29 3.50  6.419010
```

- 예제 해석
    - mtcars 데이터셋을(#1)
    - hp를 기준으로 100보다 큰 데이터만 추출한 후(#2)
    - cyl를 기준으로 각 변수들의 평균을 구한 다음에 소수점 둘째 자리까지 반올림을 한 후(#3)
    - kpl(kilometer per liter) 열을 만들어 mpg*0.4251을 수행하고(#4)
    - 만들어진 데이터를 출력(#5)과 동시에 car_data에 할당하는 과정입니다.
- 체인연산없이 실행

```
car_data =
  transform(aggregate(
    . ~ cyl,
    data = subset(mtcars, hp > 100),
    FUN = function(x)
      round(mean(x), 2)
  ),
  kpl = mpg * 0.4251)
car_data
```

```
##   cyl   mpg   disp     hp drat   wt  qsec   vs   am gear carb       kpl
## 1   4 25.90 108.05 111.00 3.94 2.15 17.75 1.00 1.00 4.50 2.00 11.010090
## 2   6 19.74 183.31 122.29 3.59 3.12 17.98 0.57 0.43 3.86 3.43  8.391474
## 3   8 15.10 353.10 209.21 3.23 4.00 16.77 0.00 0.14 3.29 3.50  6.419010
```

4.3.2. 예제 변환

- **2.1.** 예제

tidyr의 함수들도 chaining 연산과 함께 사용하면 직관적으로 사용할 수 있습니다.

```
cases %>% gather(Year, n, 2:4)
```

```
##   country Year     n
## 1      FR 2011  7000
## 2      DE 2011  5800
## 3      US 2011 15000
## 4      FR 2012  6900
## 5      DE 2012  6000
## 6      US 2012 14000
## 7      FR 2013  7000
## 8      DE 2013  6200
## 9      US 2013 13000
```

- **3.7.** 예제

dplyr에서도 함께 쓰여 데이터를 그룹화하고 수치를 요약하는 등의 작업에 특화되어 있습니다.

```
#비행기별 비행회수, 비행거리평균, 연착시간평균 산출
flights %>%
  group_by(tailnum) %>%
  summarise(
    count = n(),
    dist = mean(distance, na.rm = TRUE),
    delay = mean(arr_delay, na.rm = TRUE)
  )
```

```
## # A tibble: 4,044 × 4
##    tailnum count  dist  delay
##    <chr>   <int> <dbl>  <dbl>
##  1 D942DN      4  854. 31.5
##  2 N0EGMQ    371  676.  9.98
##  3 N10156    153  758. 12.7
##  4 N102UW     48  536.  2.94
##  5 N103US     46  535. -6.93
##  6 N104UW     47  535.  1.80
##  7 N10575    289  520. 20.7
##  8 N105UW     45  525. -0.267
##  9 N107US     41  529. -5.73
## 10 N108UW     60  534. -1.25
## # … with 4,034 more rows
```

5. tibble

- `tibble` 은 tidyverse 생태계에서 데이터 프레임을 대신하여 편리한 기능들 및 동작을 포함한 자료형입니다.
  - factor 자동 변환
  - 일부값만 출력
  - 출력시 자료형 명시
- 데이터 프레임과 비교 | 작업유형 | 데이터프레임 명령어 | 티블 명령어 | | ---- | ----- | ---- | ----- | | 생성 | `data.frame()`, `data_frame()` | `tibble()`, `tribble()` | 강제변환 (Coercion) | `as.data.frame()` | `as_tibble()` | 데이터 불러오기 | `read.*()` | `read_delim()`, `read_csv()`, `read_csv2()`, `read_tsv()` |

5.1. `tibble` 생성

`tibble()`

```
tibble(x = 1:5,
       y = 1,
       Z = x ^ 2 + y)
```

```
## # A tibble: 5 × 3
##       x     y     Z
##   <int> <dbl> <dbl>
## 1     1     1     2
## 2     2     1     5
## 3     3     1    10
## 4     4     1    17
## 5     5     1    26
```

`tribble()`

코드 단계에서 데이터를 입력받도록 하기 위해 존재하는 함수입니다.

```
tribble(~ x, ~ y, ~ z,
        #--/----/----
        "a", 2, 3.6,
        "b", 1, 8.5)
```

```
## # A tibble: 2 × 3
##   x         y     z
##   <chr> <dbl> <dbl>
## 1 a         2   3.6
## 2 b         1   8.5
```

`as_tibble()`

기존의 데이터 프레임을 tibble 형으로 전환 합니다.

```
iris_tibble = as_tibble(iris) # 기존의 데이터 프레임을 tibble로
print(class(iris)) # 기존 데이터 프레임 클래스
```

```
## [1] "data.frame"
```

```
print(class(iris_tibble)) # 새롭게 정의된 tibble 클래스 (데이터 프레임도)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
head(iris_tibble)
```

```
## # A tibble: 6 × 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
## 4          4.6         3.1          1.5         0.2 setosa
## 5          5           3.6          1.4         0.2 setosa
## 6          5.4         3.9          1.7         0.4 setosa
```

5.2. 데이터 불러오기

- 데이터를 읽어올 때, dataframe이 아닌 tibble로 읽어오기 위해서, 동일한 `tidyverse` 생태계에 속한 `readr` 패키지의 함수들을 필요로 합니다.
- 이미 `tidyverse` 를 library하였으므로 바로 이용 가능합니다.

read_csv(file)

- 기존의 데이터 불러오기와 동일하게 파일명을 지정하여 해당 파일을 tibble로 읽어올 수 있습니다.

```
read_csv("traffic.csv")
```

```
## New names:
## Rows: 500 Columns: 12
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: "," chr
## (4): rpt.id, rpt.contents, info.tp, info.tit dbl (5): ...1, start.pos.x,
## start.pos.y, end.pos.x, end.pos.y date (2): occ.dtime, end.dtime time (1):
## reg.dtime
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## • `` -> `...1`
```

```
## # A tibble: 500 × 12
##     ...1 rpt.id   rpt.co…¹ info.tp info.…² occ.dtime  reg.d…³ end.dtime  start…
⁴
##    <dbl> <chr>    <chr>    <chr>   <chr>   <date>     <time>  <date>       <dbl
>
##  1     1 01149328 서부간…  A4      단순정… 2014-06-15 17:17   2014-06-15    127.
##  2     2 01149327 동부간…  A4      단순정… 2014-06-15 17:16   2014-06-15    127.
##  3     3 01149326 북부간…  A4      단순정… 2014-06-15 17:16   2014-06-15    127.
##  4     4 01149325 올림픽…  A4      단순정… 2014-06-15 17:15   2014-06-15    127.
##  5     5 01149324 강변북…  A1      단순사… 2014-06-15 17:15   2014-06-15    127.
##  6     6 01149323 내부순…  A4      단순정… 2014-06-15 17:14   2014-06-15    127.
##  7     7 01149322 평택-시… A4      단순정… 2014-06-15 17:14   2014-06-15      0
##  8     8 01149321 서울-춘… A4      단순정… 2014-06-15 17:13   2014-06-15    127.
##  9     9 01149320 천안-논… A4      단순정… 2014-06-15 17:13   2014-06-15    127.
## 10    10 01149319 영동고…  A4      단순정… 2014-06-15 17:12   2014-06-15    127.
## # … with 490 more rows, 3 more variables: start.pos.y <dbl>, end.pos.x <dbl>,
## #   end.pos.y <dbl>, and abbreviated variable names ¹rpt.contents, ²info.tit,
## #   ³reg.dtime, ⁴start.pos.x
```

read_csv(csv_url)

- 외부에서 공개된 CSV 파일도 바로 읽어올 수 있습니다.
    - github, gist, google drive

```
file_url = "https://gist.githubusercontent.com/theoroe3/8bc989b644adc24117bc66f50c
292fc8/raw/f677a2ad811a9854c9d174178b0585a87569af60/tibbles_data.csv"
read_csv(file_url)
```

```
## Rows: 4 Columns: 4
## ── Column specification ────────────────────────────────────────────────────
─
## Delimiter: ","
## chr (1): name
## dbl (3): <-, 8, %
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this messag
e.
```

```
## # A tibble: 4 × 4
##    `<-`   `8`   `%` name
##   <dbl> <dbl> <dbl> <chr>
## 1     1     2  0.25 t
## 2     2     4  0.25 h
## 3     3     6  0.25 e
## 4     4     8  0.25 o
```

locale 설정

- 한글이 포함된 데이터를 읽어올 때, `read.csv` 에서 `fileEncdoing` 으로 조정을 하였습니다.

- read_csv 에서는 주로 `locale` 인자를 설정해 주어야 하는데, 통상적으로 `locale("ko",encoding="euc-kr")` 와 같이 설정해줍니다.

예제는 아래의 연습문제에서 데이터를 불러오는 것으로 알아보겠습니다.

5.3. 결측값 처리

결측값을 처리하는 방법으로 결측값이 있는 행을 삭제하거나, 다른 값으로 치환하는 방법이 있습니다.

- `drop_na()` 는 결측값이 있는 행을 삭제하는 함수입니다.
- `fill()` 은 인접한 값들을 이용해서 결측값을 치환하는 방법입니다.
- `replace_na()` 는 특정한 값을 이용해서 결측값을 치환하는 방법입니다.

drop_na()

```
library(dplyr)
df = tibble(x = c(1, 2, NA) , y = c("a", NA, "b"))
df %>% drop_na()
```

```
## # A tibble: 1 × 2
##       x y
##   <dbl> <chr>
## 1     1 a
```

```
df %>% drop_na(x)
```

```
## # A tibble: 2 × 2
##       x y
##   <dbl> <chr>
## 1     1 a
## 2     2 <NA>
```

```
vars = "y"
df %>% drop_na(x, any_of(vars))
```

```
## # A tibble: 1 × 2
##       x y
##   <dbl> <chr>
## 1     1 a
```

fill()

```
sales = tibble::tribble(
  ~quarter, ~year, ~sales,
  "Q1", 2000, 66013,
  "Q2", NA, 69182,
  "Q3", NA, 53175,
  "Q4", NA, 21001,
  "Q1", 2001, 46036,
  "Q2", NA, 58842,
  "Q3", NA, 44568,
  "Q4", NA, 50197,
  "Q1", 2002, 39113,
  "Q2", NA, 41668,
  "Q3", NA, 30144,
  "Q4", NA, 52897,
  "Q1", 2004, 32129,
  "Q2", NA, 67686,
  "Q3", NA, 31768,
  "Q4", NA, 49094
)
sales %>% fill(year)
```

```
## # A tibble: 16 × 3
##    quarter  year sales
##    <chr>   <dbl> <dbl>
##  1 Q1       2000 66013
##  2 Q2       2000 69182
##  3 Q3       2000 53175
##  4 Q4       2000 21001
##  5 Q1       2001 46036
##  6 Q2       2001 58842
##  7 Q3       2001 44568
##  8 Q4       2001 50197
##  9 Q1       2002 39113
## 10 Q2       2002 41668
## 11 Q3       2002 30144
## 12 Q4       2002 52897
## 13 Q1       2004 32129
## 14 Q2       2004 67686
## 15 Q3       2004 31768
## 16 Q4       2004 49094
```

fill(.direction="up")

```
tidy_pets = tibble::tribble(
  ~rank, ~pet_type, ~breed,
  1L, NA, "Boston Terrier",
  2L, NA, "Retrievers (Labrador)",
  3L, NA, "Retrievers (Golden)",
  4L, NA, "French Bulldogs",
  5L, NA, "Bulldogs",
  6L, "Dog", "Beagles",
  1L, NA, "Persian",
  2L, NA, "Maine Coon",
  3L, NA, "Ragdoll",
  4L, NA, "Exotic",
  5L, NA, "Siamese",
  6L, "Cat", "American Short"
)
tidy_pets %>%
  fill(pet_type, .direction = "up")
```

```
## # A tibble: 12 × 3
##     rank pet_type breed
##    <int> <chr>    <chr>
##  1     1 Dog      Boston Terrier
##  2     2 Dog      Retrievers (Labrador)
##  3     3 Dog      Retrievers (Golden)
##  4     4 Dog      French Bulldogs
##  5     5 Dog      Bulldogs
##  6     6 Dog      Beagles
##  7     1 Cat      Persian
##  8     2 Cat      Maine Coon
##  9     3 Cat      Ragdoll
## 10     4 Cat      Exotic
## 11     5 Cat      Siamese
## 12     6 Cat      American Short
```

replace_na()

```
df = tibble(x = c(1, 2, NA) , y = c("a", NA, "b"))
df %>% replace_na(list(x = 0, y = "unknown"))
```

```
## # A tibble: 3 × 2
##       x y
##   <dbl> <chr>
## 1     1 a
## 2     2 unknown
## 3     0 b
```

```
df %>% dplyr::mutate(x = replace_na(x, 0))
```

```
## # A tibble: 3 × 2
##       x y
##   <dbl> <chr>
## 1     1 a
## 2     2 <NA>
## 3     0 b
```

PR12 연습문제

```
data1 = read.csv("data1.csv", fileEncoding = "EUC-KR")
data2 = read.csv("data2.csv", fileEncoding = "EUC-KR")
```

문제 1

- data1.csv에는 지역별 온실가스 배출량 정보가 있으며, data2.csv에는 지역 별 기업 수가 있다.
- data1.csv는 data1이라는 변수에 저장하고 data2.csv는 data2이라는 변수에 저장하고, 두 변수를 시도 를 기준으로 하나의 데이터프레임으로 만드시오.
- 조건1. wide 형태를 long으로 바꾸어야 함.
- 조건2. join을 진행하여야 함.
- 조건3. head(, 10)을 통해 상위 10개만 출력하시오.

```
data2_long = gather(data2, 광역시도명, n, 2:18)
fetched_data = inner_join(data1, data2_long)
```

```
## Joining, by = "광역시도명"
```

```
head(fetched_data, 10)
```

```
##     년도 광역시도명 전체온실가스배출량 X1인당인구배출량
## 1  2019 서울특별시            11366609             1.168
## 2  2019 서울특별시            11366609             1.168
## 3  2019 서울특별시            11366609             1.168
## 4  2019 서울특별시            11366609             1.168
## 5  2019 부산광역시             6747556             1.977
## 6  2019 부산광역시             6747556             1.977
## 7  2019 부산광역시             6747556             1.977
## 8  2019 부산광역시             6747556             1.977
## 9  2019 대구광역시             4003434             1.642
## 10 2019 대구광역시             4003434             1.642
##     관리업체1개당.온실가스배출량 사업장1개당온실가스.배출량       등록현황      n
## 1                       35744.05                  3305.208   규모(소기업)     19
## 2                       35744.05                  3305.208   규모(중기업)    363
## 3                       35744.05                  3305.208   규모(대기업) 11206
## 4                       35744.05                  3305.208 규모(중견기업)      9
## 5                       47518.00                  9032.873   규모(소기업)     33
## 6                       47518.00                  9032.873   규모(중기업)    527
## 7                       47518.00                  9032.873   규모(대기업) 10202
## 8                       47518.00                  9032.873 규모(중견기업)      1
## 9                       33642.30                  8087.745   규모(소기업)     30
## 10                      33642.30                  8087.745   규모(중기업)    438
```

문제2.

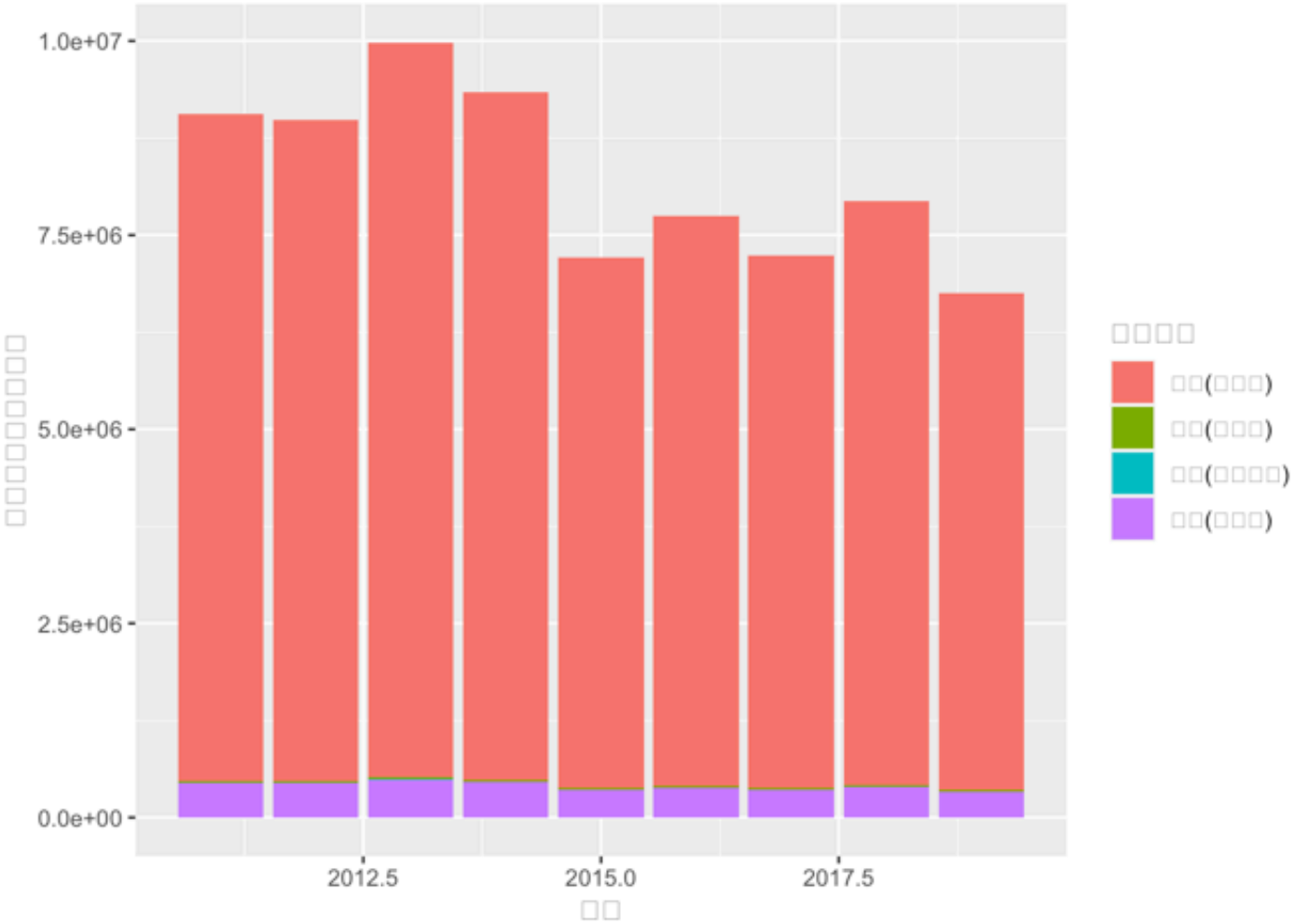
group_by, filter 함수를 사용하여 문제 1의 데이터를 필요에 따라 변형/집계/추출한 다음 시각화를 진행하시오.

부산 시의 전체온실가스배출량의 연도 별 추이를, 또 연도 별 사업장 등록형태 별 비중도 보고 싶어요.

```
entire_pusan_corporative = sum((data2_long %>% filter(광역시도명 == '부산광역시'))$n)
co2_pusan_by_year = fetched_data %>%
  filter(광역시도명 == '부산광역시') %>%
  group_by(년도) %>%
  select(전체온실가스배출량, 등록현황, n) %>%
  mutate(
    전체온실가스배출량 = n / entire_pusan_corporative * 전체온실가스배출량
  )
```

```
## Adding missing grouping variables: `년도`
```

```
ggplot(co2_pusan_by_year, aes(fill = 등록현황, y = 전체온실가스배출량, x = 년도)) +
  geom_bar(position = "stack", stat = "identity")
```



PR12 도전문제

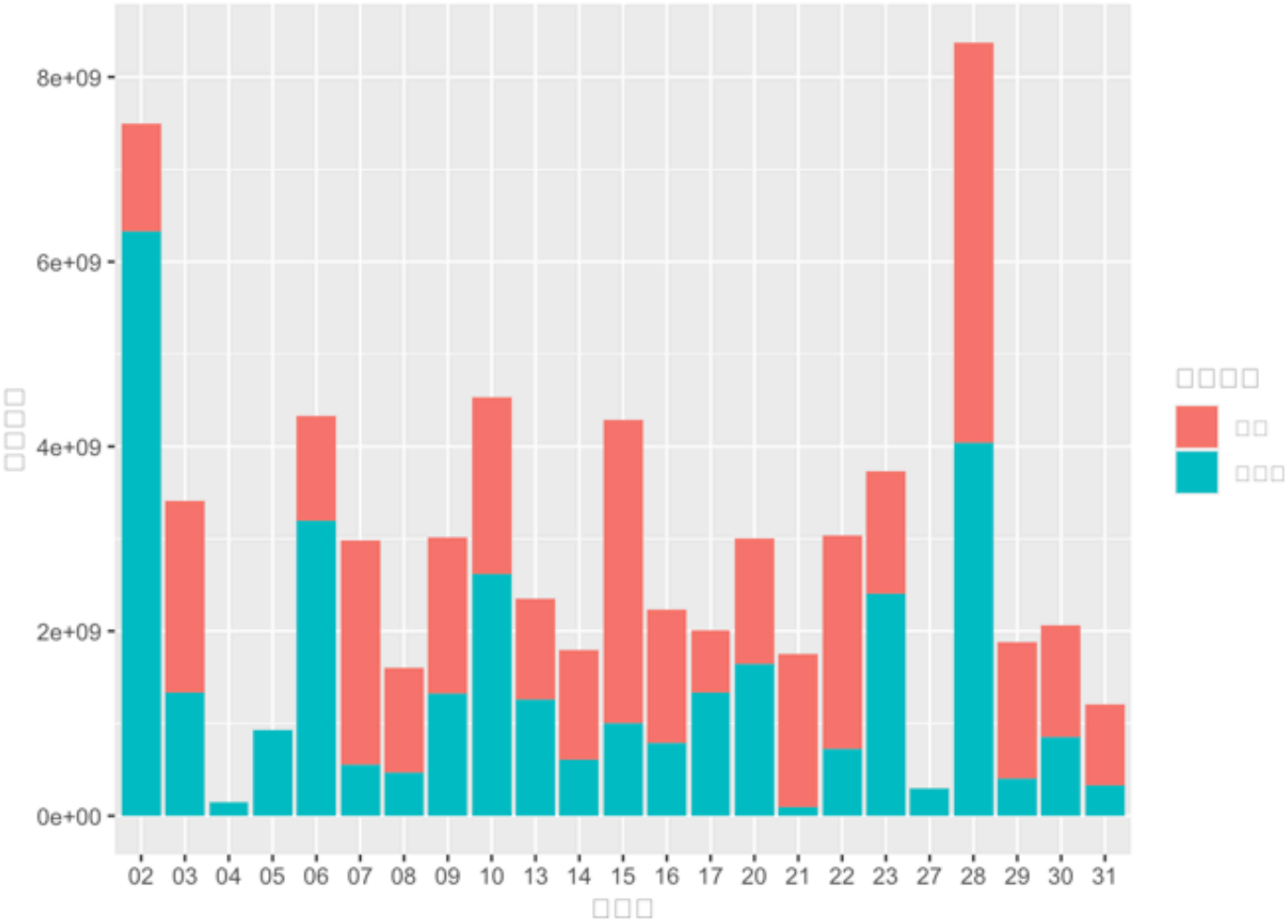
제공되는 데이터를 이용하여 수업시간에 배운 함수들을 최대한 활용해 주세요.

- 조건1. 최소 10줄 이상의 코드를 작성하세요.
- 조건2. tidyr, dplyr, magrittr의 함수를 최소 1개 이상씩 사용하세요.
- 조건3. 주석을 최대한 상세하게 추가해 주세요.

1. 2020년 1월 한 달 동안 일 별 대출 금액 추이와 일 별 대출 금액 중 내방 고객 대 온라인 고객의 차지하는 비율을 시각적으로 확인하고 싶어요.

```
# 경기도 지원 대출금 파일을 분석하기 위해 read.csv 함수를 사용하여 메모리에 저장
supportive_g_money_raw = read.csv('G-money지원현황.csv', fileEncoding = 'euc-kr')
# 구문을 간결하게 하기 위해 margrittr의 pipeline을 이용
supportive_g_money_data = supportive_g_money_raw %>%
  # supportive_g_money_raw의 신청일자 열을 년월일 각자의 열로 분리하는데, 하이픈을 기준으로 구분
  # 열을 특정 기준으로 여러 열로 쪼개기 위해 tidyr::separate를 이용
  separate(col = 신청일자, sep = '-', into = c('신청년', '신청월', '신청일')) %>%
  # 신청년과 신청월을 필터링하기 위해 dplyr::filter 이용
  filter(신청년 == '2020', 신청월 == '01')

# 시각화를 위해 ggplot을 이용, 접수구분 열로 색 구분을, 대출금액의 합계를 y축, 신청일을 x축으로 하고자 함
ggplot(supportive_g_money_data, aes(fill = 접수구분, y = 대출금액, x = 신청일)) +
  # 갯수를 세는 게 아닌, 이미 값이 구해져 있기 때문에 stat 파라미터를 identity로 설정하여야
  geom_bar(position = "stack", stat = "identity")
```



2. 금리가 4퍼센트가 넘는 지원대출의 신청년월에 따른 대출금액 추이를 보고 싶어요. 또 신청년월 별로 자금대분류 별 비중도 보고 싶어요.

```
# supportive_g_money_raw를 pipeline을 통해 전달
supportive_g_money_data_1 = supportive_g_money_raw %>%
  # 금리가 4퍼센트 이상만 필터링
  filter(대출금리 > 4) %>%
  # 신청일자 열을 년월일 각자의 열로 분리하는데, 하이픈을 기준으로 구분
  # 하나의 열을 특정 규칙으로 여러 개의 열로 쪼개기 위해 tidyr::separate 이용
  separate(col = 신청일자, sep = '-', into = c('신청년', '신청월', '신청일')) %>%
  # 신청년 열과 신청월 열을 하나의 열로 관리
  # 두 개의 열을 하나의 열로 묶기 위해 다시 tidyr::unite로 묶는 과정
  # 각 행을 신청년월로 분류하고자 이와같은 과정을 실시함
  unite(col = 신청년월, 신청년, 신청월, sep = '')

# 시각화를 위해 ggplot을 이용, 자금대분류 열로 색 구분을, 대출금액의 합계를 y축, 신청일을 x축으로 하고자 함
ggplot(supportive_g_money_data_1, aes(fill = 자금대분류, y = 대출금액, x = 신청년월)) +
  # 갯수를 세는 게 아닌, 이미 값이 구해져 있기 때문에 stat 파라미터를 identity로 설정하여야
  geom_bar(position = "stack", stat = "identity") +
  theme(axis.text.x=element_text(angle=90, hjust=1))
```