데이터 분석의 기본이 되는 데이터전처리

Step1. 데이터 전처리를 위한 기초 학습하기

https://mrchypark.github.io/dabrp_classnote3/class3

[pdf다운로드] [문의하기] [피드백하기]

박찬엽

2017년 9월 28일

목차

과제 질답

수업의 목표

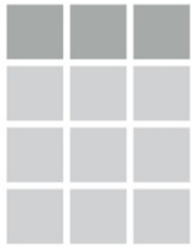
1. 데이터를 다루는 주요 7가지 동작을 설명할 수 있다.

2. tidy data, long form과 wide form의 장단점을 설명할 수 있다.

3. 데이터 베이스의 테이블을 R 객체에 연결하여 데이터를 다룰 수 있다.

데이터를 다루는 주요 7가지 동작

데이터 다루기

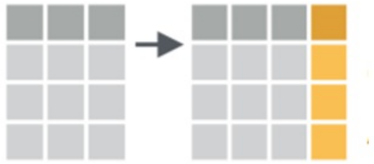


열 방향

선택

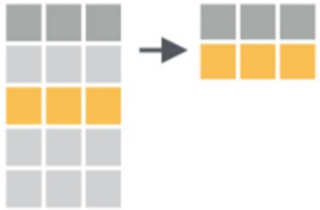
① 

계산

② 

행 방향

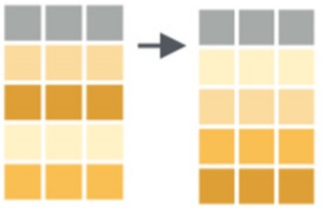
조건

③ 

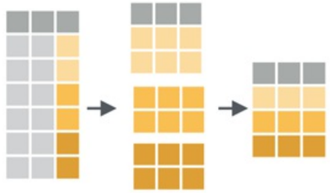
추가

④ 

정렬

⑤ 

그룹 계산

⑥ 

열 결합

⑦

x

A	B	C
a	t	1
b	u	2
c	v	3

+

y

A	B	D
a	t	3
b	u	2
d	w	1

=

A	B	C	D
a	t	1	3
b	u	2	2
c	v	3	NA

실습 데이터 소개

nycflights13는 2013년 미국의 비행기 운항기록에 관련된 airlines, airports, flights, planes, weather의 5개 데이터를 가지고 있는 데이터 패키지

```
if (!requireNamespace("nycflights13")) install.packages("nycflights13")
```

```
## package 'nycflights13' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##     C:\Users\mrchypark\AppData\Local\Temp\RtmpGkwJOm\downloaded_packages
```

```
library(nycflights13)
nycflights13::flights
```

```
## # 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
```

nycflights13 코드북

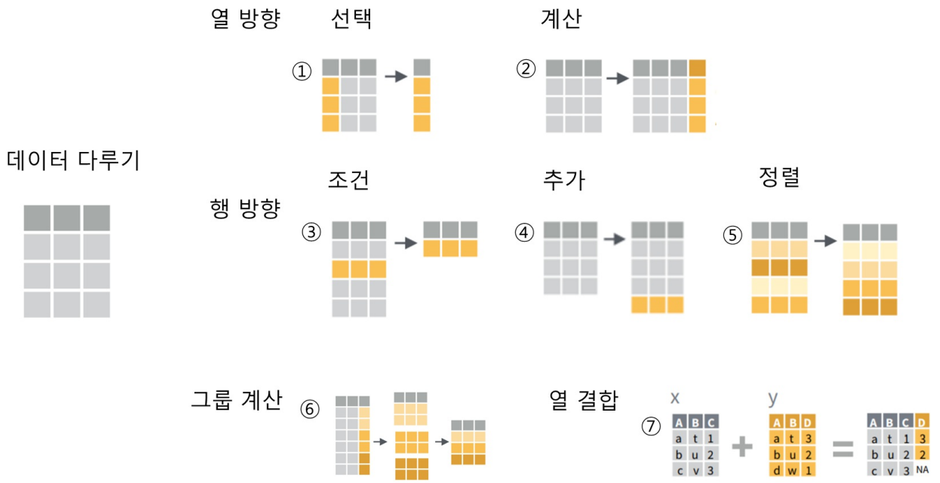
```
str(flights)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    336776 obs. of  19 variables:
##  $ year          : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int  517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int  515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : num  2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int  830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int  819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : num  11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr  "UA" "UA" "AA" "B6" ...
##  $ flight        : int  1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : chr  "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr  "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr  "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time      : num  227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : num  1400 1416 1089 1576 762 ...
##  $ hour          : num  5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : num  15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct, format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

데이터를 다루는 주요 7가지 동작

dplyr은 데이터를 다루는 주요 7가지 동작 자체를 함수로 가지고 추가적인 helper 함수를 함께 제공

1. 열 방향: 선택 - select()
2. 열 방향: 계산 - mutate()
3. 행 방향: 조건 - filter()
4. 행 방향: 추가 - bind_rows()
5. 행 방향: 정렬 - arrange()
6. 그룹 계산 - group_by() + summarise()
7. 열 결합 - left_join()



dplyr 준비

```
if (!requireNamespace("dplyr")) install.packages("dplyr")
```

```
## Loading required namespace: dplyr
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

열 방향: 선택 - select()

데이터에서 컬럼을 선택하여 사용함. select()는 선언된 순서대로 컬럼을 정렬함

```
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
```

열 방향: 선택 - select()

숫자에서만 제공하던 from:to 문법을 컬럼 순서를 기준으로 지원

```
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
```

열 방향: 선택 - select()

-(마이너스)는 지정한 컬럼을 제외하고 전부라는 의미

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

```
## # A tibble: 336,776 x 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
##  7      555            600        -5      913            854        19
##  8      557            600        -3      709            723       -14
##  9      557            600        -3      838            846        -8
## 10      558            600        -2      753            745         8
## # ... with 336,766 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>
```

열 방향: 선택 - select()

everything() 같은 helper 함수를 제공 everything()은 select()내에 선언된 컬럼을 제외한 나머지 전부라는 의미.

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

```
## # A tibble: 336,776 x 19
##                time_hour air_time  year month   day dep_time sched_dep_time
##                   <dttm>    <dbl> <int> <int> <int>    <int>          <int>
##  1 2013-01-01 05:00:00      227  2013     1     1      517            515
##  2 2013-01-01 05:00:00      227  2013     1     1      533            529
##  3 2013-01-01 05:00:00      160  2013     1     1      542            540
##  4 2013-01-01 05:00:00      183  2013     1     1      544            545
##  5 2013-01-01 06:00:00      116  2013     1     1      554            600
##  6 2013-01-01 05:00:00      150  2013     1     1      554            558
##  7 2013-01-01 06:00:00      158  2013     1     1      555            600
##  8 2013-01-01 06:00:00       53  2013     1     1      557            600
##  9 2013-01-01 06:00:00      140  2013     1     1      557            600
## 10 2013-01-01 06:00:00      138  2013     1     1      558            600
## # ... with 336,766 more rows, and 12 more variables: dep_delay <dbl>,
## #   arr_time <int>, sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, distance <dbl>,
## #   hour <dbl>, minute <dbl>
```

열 방향: 선택 - select()

ends_with()같이 글자의 일부에 해당하는 컬럼 전부를 가져오는 helper 함수도 있음. 정규표현식의 주요 기능을 함수로 제공. ? select로 확인

```
select(flights, year:day, ends_with("delay"), distance, air_time)
```

```
## # A tibble: 336,776 x 7
##     year month   day dep_delay arr_delay distance air_time
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl>
##  1  2013     1     1         2        11     1400      227
##  2  2013     1     1         4        20     1416      227
##  3  2013     1     1         2        33     1089      160
##  4  2013     1     1        -1       -18     1576      183
##  5  2013     1     1        -6       -25      762      116
##  6  2013     1     1        -4        12      719      150
##  7  2013     1     1        -5        19     1065      158
##  8  2013     1     1        -3       -14      229       53
##  9  2013     1     1        -3        -8      944      140
## 10  2013     1     1        -2         8      733      138
## # ... with 336,766 more rows
```

열 방향: 계산 - mutate()

출력 편의를 위해 일부 데이터만 사용

```
flights_sml <- select(flights, year:day, ends_with("delay"), distance, air_time)
flights_sml
```

```
## # A tibble: 336,776 x 7
##     year month   day dep_delay arr_delay distance air_time
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl>
##  1  2013     1     1         2        11     1400      227
##  2  2013     1     1         4        20     1416      227
##  3  2013     1     1         2        33     1089      160
##  4  2013     1     1        -1       -18     1576      183
##  5  2013     1     1        -6       -25      762      116
##  6  2013     1     1        -4        12      719      150
##  7  2013     1     1        -5        19     1065      158
##  8  2013     1     1        -3       -14      229       53
##  9  2013     1     1        -3        -8      944      140
## 10  2013     1     1        -2         8      733      138
## # ... with 336,766 more rows
```

열 방향: 계산 - mutate()

각 컬럼간의 계산으로 새로운 열을 만들 수 있음

```
mutate(flights_sml,
       gain = arr_delay - dep_delay,
       speed = distance / air_time * 60
)
```

```
## # A tibble: 336,776 x 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     9 370.0441
##  2  2013     1     1         4        20     1416      227    16 374.2731
##  3  2013     1     1         2        33     1089      160    31 408.3750
##  4  2013     1     1        -1       -18     1576      183   -17 516.7213
##  5  2013     1     1        -6       -25      762      116   -19 394.1379
##  6  2013     1     1        -4        12      719      150    16 287.6000
##  7  2013     1     1        -5        19     1065      158    24 404.4304
##  8  2013     1     1        -3       -14      229       53   -11 259.2453
##  9  2013     1     1        -3        -8      944      140    -5 404.5714
## 10  2013     1     1        -2         8      733      138    10 318.6957
## # ... with 336,766 more rows
```

열 방향: 계산 - mutate()

컬럼을 지우거나 기존의 컬럼을 변경하는 것도 가능

```
mutate(flights_sml,
       arr_delay = NULL,
       air_time = air_time / 60
)
```

```
## # A tibble: 336,776 x 6
##     year month   day dep_delay distance  air_time
##    <int> <int> <int>     <dbl>    <dbl>     <dbl>
##  1  2013     1     1         2     1400 3.7833333
##  2  2013     1     1         4     1416 3.7833333
##  3  2013     1     1         2     1089 2.6666667
##  4  2013     1     1        -1     1576 3.0500000
##  5  2013     1     1        -6      762 1.9333333
##  6  2013     1     1        -4      719 2.5000000
##  7  2013     1     1        -5     1065 2.6333333
##  8  2013     1     1        -3      229 0.8833333
##  9  2013     1     1        -3      944 2.3333333
## 10  2013     1     1        -2      733 2.3000000
## # ... with 336,766 more rows
```

열 방향: 계산 - mutate()

transmute()는 계산한 컬럼만 있는 테이블을 생성

```
transmute(flights,
          gain = arr_delay - dep_delay,
          hours = air_time / 60,
          gain_per_hour = gain / hours
)
```

```
## # A tibble: 336,776 x 3
##     gain     hours gain_per_hour
##    <dbl>     <dbl>          <dbl>
##  1     9 3.7833333       2.378855
##  2    16 3.7833333       4.229075
##  3    31 2.6666667      11.625000
##  4   -17 3.0500000      -5.573770
##  5   -19 1.9333333      -9.827586
##  6    16 2.5000000       6.400000
##  7    24 2.6333333       9.113924
##  8   -11 0.8833333     -12.452830
##  9    -5 2.3333333      -2.142857
## 10    10 2.3000000       4.347826
## # ... with 336,766 more rows
```

열 방향: 계산 - mutate()

group_by()와 함께 window function 들이 유용하게 사용됨

```
flights_smlg <- group_by(flights_sml, month)
mutate(flights_smlg, rank = row_number(desc(arr_delay)))
```

```
## # A tibble: 336,776 x 8
## # Groups:   month [12]
##     year month   day dep_delay arr_delay distance air_time  rank
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <int>
##  1  2013     1     1         2        11     1400      227  6970
##  2  2013     1     1         4        20     1416      227  5064
##  3  2013     1     1         2        33     1089      160  3458
##  4  2013     1     1        -1       -18     1576      183 21131
##  5  2013     1     1        -6       -25      762      116 23925
##  6  2013     1     1        -4        12      719      150  6699
##  7  2013     1     1        -5        19     1065      158  5226
##  8  2013     1     1        -3       -14      229       53 19019
##  9  2013     1     1        -3        -8      944      140 15534
## 10  2013     1     1        -2         8      733      138  7912
## # ... with 336,766 more rows
```

행 방향: 조건 - filter()

filter()는 데이터 중에 조건에 해당하는 일부 데이터만 필터해서 사용. 논리 연산자와 결합하여 많이 사용하며 이곳에서 추가적으로 내용을 확인할 수 있음

```
filter(flights, month == 1)
```

```
## # A tibble: 27,004 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 26,994 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>
```

행 방향: 조건 - filter()

& 는 **and** 라는 뜻이며 조건을 추가할 때 사용

```
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>
```

행 방향: 조건 - filter()

|는 **or** 라는 뜻

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

```
## # A tibble: 55,403 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    11     1        5           2359         6      352
##  2  2013    11     1       35           2250       105      123
##  3  2013    11     1      455            500        -5      641
##  4  2013    11     1      539            545        -6      856
##  5  2013    11     1      542            545        -3      831
##  6  2013    11     1      549            600       -11      912
##  7  2013    11     1      550            600       -10      705
##  8  2013    11     1      554            600        -6      659
##  9  2013    11     1      554            600        -6      826
## 10  2013    11     1      554            600        -6      749
## # ... with 55,393 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>
```

행 방향: 조건 - filter()

%in%는 유용하게 사용하는 논리 연산자로 왼쪽에 있는 벡터가 오른쪽 벡터의 데이터 중 어느 하나라도 맞으면 출력

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

```
## # A tibble: 55,403 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    11     1        5           2359         6      352
##  2  2013    11     1       35           2250       105      123
##  3  2013    11     1      455            500        -5      641
##  4  2013    11     1      539            545        -6      856
##  5  2013    11     1      542            545        -3      831
##  6  2013    11     1      549            600       -11      912
##  7  2013    11     1      550            600       -10      705
##  8  2013    11     1      554            600        -6      659
##  9  2013    11     1      554            600        -6      826
## 10  2013    11     1      554            600        -6      749
## # ... with 55,393 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>
```

행 방향: 조건 - filter()

! 는 local 데이터에서 결과를 반대로 뒤집는 역할을 하며 수학에서의 괄호와 같이 연산의 범위를 작성해 두는 것이 문제 발생 소지가 적어점

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

```
## # A tibble: 316,050 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 316,040 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>
```

행 방향: 추가 - bind_rows()

bind_rows()를 진행하기 위해서 데이터를 작성

```
feb<-filter(flights, month==2)
dec<-filter(flights, month==12)
dim(feb); dim(dec)
```

```
## [1] 24951    19
```

```
## [1] 28135    19
```

```
nrow(feb)+nrow(dec)
```

```
## [1] 53086
```

행 방향: 추가 - bind_rows()

bind_rows()는 컬럼 이름을 기준으로 같은 컬럼 밑에 데이터를 붙여서 **묶어줌**.

```
bind_rows(feb, dec)
```

```
## # A tibble: 53,086 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     2     1      456            500        -4      652
##  2  2013     2     1      520            525        -5      816
##  3  2013     2     1      527            530        -3      837
##  4  2013     2     1      532            540        -8     1007
##  5  2013     2     1      540            540         0      859
##  6  2013     2     1      552            600        -8      714
##  7  2013     2     1      552            600        -8      919
##  8  2013     2     1      552            600        -8      655
##  9  2013     2     1      553            600        -7      833
## 10  2013     2     1      553            600        -7      821
## # ... with 53,076 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>
```

행 방향: 추가 - bind_rows()

list()로 구분된 데이터도 **묶어줌**.

```
bind_rows(list(feb, dec))
```

```
## # A tibble: 53,086 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     2     1      456            500        -4      652
##  2  2013     2     1      520            525        -5      816
##  3  2013     2     1      527            530        -3      837
##  4  2013     2     1      532            540        -8     1007
##  5  2013     2     1      540            540         0      859
##  6  2013     2     1      552            600        -8      714
##  7  2013     2     1      552            600        -8      919
##  8  2013     2     1      552            600        -8      655
##  9  2013     2     1      553            600        -7      833
## 10  2013     2     1      553            600        -7      821
## # ... with 53,076 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>
```

행 방향: 추가 - bind_rows()

split()은 첫번째 인자로 받은 데이터를 컬럼을 기준으로 list()로 분리해 줌.

```
flights_mon<-split(flights, flights$month)
summary(flights_mon)
```

```
##    Length Class  Mode
## 1  19     tbl_df list
## 2  19     tbl_df list
## 3  19     tbl_df list
## 4  19     tbl_df list
## 5  19     tbl_df list
## 6  19     tbl_df list
## 7  19     tbl_df list
## 8  19     tbl_df list
## 9  19     tbl_df list
## 10 19     tbl_df list
## 11 19     tbl_df list
## 12 19     tbl_df list
```

행 방향: 추가 - bind_rows()

split()으로 분리된 12개의 list() 자료도 잘 합쳐줌

```
nrow(flights)
```

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

```
bind_rows(flights_mon)
```

```
## # 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>
```

행 방향: 추가 - bind_rows()

다른 종류의 데이터도 **묶어줌**. c()는 vector를 생성하고, data_frame은 data.frame을 생성함

```
bind_rows(
  c(a = 1, b = 2),
  data_frame(a = 3:4, b = 5:6),
  c(a = 7, b = 8)
)
```

```
## # A tibble: 4 x 2
##       a     b
##   <dbl> <dbl>
## 1     1     2
## 2     3     5
## 3     4     6
## 4     7     8
```

행 방향: 추가 - bind_rows()

데이터를 묶을 때 데이터를 구분하는 컬럼을 추가할 수 있음

```
bind_rows(list(feb, dec), .id = "id")
```

```
## # A tibble: 53,086 x 20
##        id  year month   day dep_time sched_dep_time dep_delay arr_time
##     <chr> <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1      1  2013     2     1      456            500        -4      652
##  2      1  2013     2     1      520            525        -5      816
##  3      1  2013     2     1      527            530        -3      837
##  4      1  2013     2     1      532            540        -8     1007
##  5      1  2013     2     1      540            540         0      859
##  6      1  2013     2     1      552            600        -8      714
##  7      1  2013     2     1      552            600        -8      919
##  8      1  2013     2     1      552            600        -8      655
##  9      1  2013     2     1      553            600        -7      833
## 10      1  2013     2     1      553            600        -7      821
## # ... with 53,076 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>
```

행 방향: 추가 - bind_rows()

데이터를 구분하는 컬럼에 대해 이름이 동작하는 방식

```
bind_rows(list(a = feb, b = dec), .id = "data")
```

```
## # A tibble: 53,086 x 20
##      data  year month   day dep_time sched_dep_time dep_delay arr_time
##     <chr> <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1      a  2013     2     1      456            500        -4      652
##  2      a  2013     2     1      520            525        -5      816
##  3      a  2013     2     1      527            530        -3      837
##  4      a  2013     2     1      532            540        -8     1007
##  5      a  2013     2     1      540            540         0      859
##  6      a  2013     2     1      552            600        -8      714
##  7      a  2013     2     1      552            600        -8      919
##  8      a  2013     2     1      552            600        -8      655
##  9      a  2013     2     1      553            600        -7      833
## 10      a  2013     2     1      553            600        -7      821
## # ... with 53,076 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>
```

행 방향: 추가 - bind_rows()

같은 이름의 컬럼이 없을 때는 NA로 채우면서 동작함

```
bind_rows(data.frame(x = 1:3), data.frame(y = 1:4))
```

```
##     x  y
## 1   1 NA
## 2   2 NA
## 3   3 NA
## 4  NA  1
## 5  NA  2
## 6  NA  3
## 7  NA  4
```

행 방향: 정렬 - arrange()

arrange()는 지정되는 컬럼 순으로 오름차순 정렬해주는 함수

```
arrange(flights, dep_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    12     7     2040           2123       -43       40
##  2  2013     2     3     2022           2055       -33     2240
##  3  2013    11    10     1408           1440       -32     1549
##  4  2013     1    11     1900           1930       -30     2233
##  5  2013     1    29     1703           1730       -27     1947
##  6  2013     8     9      729            755       -26     1002
##  7  2013    10    23     1907           1932       -25     2143
##  8  2013     3    30     2030           2055       -25     2213
##  9  2013     3     2     1431           1455       -24     1601
## 10  2013     5     5      934            958       -24     1225
## # ... 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>
```

행 방향: 정렬 - arrange()

desc()는 내림차순 정렬로 방향을 바꾸는 helper 함수

```
arrange(flights, desc(month), dep_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    12     7     2040           2123       -43       40
##  2  2013    12    25     2036           2059       -23     2313
##  3  2013    12     4     1910           1930       -20     2101
##  4  2013    12    11      710            730       -20     1039
##  5  2013    12    10     1841           1900       -19     2028
##  6  2013    12    14      921            940       -19     1256
##  7  2013    12     6      811            829       -18     1119
##  8  2013    12    30      657            715       -18      927
##  9  2013    12     7     1658           1715       -17     1956
## 10  2013    12     7     2043           2100       -17     2250
## # ... 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>
```

그룹 계산 - group_by() + summarise()

summarise()는 여러 데이터를 요약하여 특성을 파악하는 방식으로 동작하는 함수들을 적용할 때 사용.

```
summarise(flights, mean = mean(dep_delay, na.rm=T), n = n())
```

```
## # A tibble: 1 x 2
##       mean      n
##      <dbl>  <int>
## 1 12.63907 336776
```

그룹 계산 - group_by() + summarise()

group_by()는 데이터에 **지정한 컬럼별**이라는 추가 조건을 지정하는 기능을 수행

```
flights_g<-group_by(flights, month)
flights_g
```

```
## # A tibble: 336,776 x 19
## # Groups:   month [12]
##     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>
```

```
summarise(flights_g, mean = mean(dep_delay, na.rm=T), n = n())
```

```
## # A tibble: 12 x 3
```

그룹 계산 - group_by() + summarise()

group_by()에 의해 **지정한 컬럼별** summarise()연산을 수행함

```
summarise(flights_g, mean = mean(dep_delay, na.rm=T), n = n())
```

```
## # A tibble: 12 x 3
##    month      mean     n
##    <int>     <dbl> <int>
##  1     1 10.036665 27004
##  2     2 10.816843 24951
##  3     3 13.227076 28834
##  4     4 13.938038 28330
##  5     5 12.986859 28796
##  6     6 20.846332 28243
##  7     7 21.727787 29425
##  8     8 12.611040 29327
##  9     9  6.722476 27574
## 10    10  6.243988 28889
## 11    11  5.435362 27268
## 12    12 16.576688 28135
```

열 결합(Join) - left_join()

select()를 사용하여 데이터 준비

```
flights2 <- select(flights, year:day, hour, origin, dest, tailnum, carrier)
flights2
```

```
## # A tibble: 336,776 x 8
##     year month   day  hour origin  dest tailnum carrier
##    <int> <int> <int> <dbl>  <chr> <chr>   <chr>   <chr>
##  1  2013     1     1     5    EWR   IAH  N14228      UA
##  2  2013     1     1     5    LGA   IAH  N24211      UA
##  3  2013     1     1     5    JFK   MIA  N619AA      AA
##  4  2013     1     1     5    JFK   BQN  N804JB      B6
##  5  2013     1     1     6    LGA   ATL  N668DN      DL
##  6  2013     1     1     5    EWR   ORD  N39463      UA
##  7  2013     1     1     6    EWR   FLL  N516JB      B6
##  8  2013     1     1     6    LGA   IAD  N829AS      EV
##  9  2013     1     1     6    JFK   MCO  N593JB      B6
## 10  2013     1     1     6    LGA   ORD  N3ALAA      AA
## # ... with 336,766 more rows
```

열 결합(Join) - left_join()

left_join()은 왼쪽 데이터를 기준으로 하고, by로 지정된 컬럼이 같은 데이터임을 식별하는 key로 지정하여 오른쪽 데이터를 왼쪽 데이터에 결합하는 함수

```
left_join(flights2, airlines, by = "carrier")
```

```
## # A tibble: 336,776 x 9
##     year month   day  hour origin  dest tailnum carrier
##    <int> <int> <int> <dbl>  <chr> <chr>   <chr>   <chr>
##  1  2013     1     1     5    EWR   IAH  N14228      UA
##  2  2013     1     1     5    LGA   IAH  N24211      UA
##  3  2013     1     1     5    JFK   MIA  N619AA      AA
##  4  2013     1     1     5    JFK   BQN  N804JB      B6
##  5  2013     1     1     6    LGA   ATL  N668DN      DL
##  6  2013     1     1     5    EWR   ORD  N39463      UA
##  7  2013     1     1     6    EWR   FLL  N516JB      B6
##  8  2013     1     1     6    LGA   IAD  N829AS      EV
##  9  2013     1     1     6    JFK   MCO  N593JB      B6
## 10  2013     1     1     6    LGA   ORD  N3ALAA      AA
## # ... with 336,766 more rows, and 1 more variables: name <chr>
```

열 결합(Join) - left_join()

mutate(), match()등의 함수로 구현하려면 아래와 같음

```
mutate(flights2, name = airlines$name[match(carrier, airlines$carrier)])
```

```
## # A tibble: 336,776 x 9
##     year month   day  hour origin  dest tailnum carrier
##    <int> <int> <int> <dbl>  <chr> <chr>   <chr>   <chr>
##  1  2013     1     1     5    EWR   IAH  N14228      UA
##  2  2013     1     1     5    LGA   IAH  N24211      UA
##  3  2013     1     1     5    JFK   MIA  N619AA      AA
##  4  2013     1     1     5    JFK   BQN  N804JB      B6
##  5  2013     1     1     6    LGA   ATL  N668DN      DL
##  6  2013     1     1     5    EWR   ORD  N39463      UA
##  7  2013     1     1     6    EWR   FLL  N516JB      B6
##  8  2013     1     1     6    LGA   IAD  N829AS      EV
##  9  2013     1     1     6    JFK   MCO  N593JB      B6
## 10  2013     1     1     6    LGA   ORD  N3ALAA      AA
## # ... with 336,766 more rows, and 1 more variables: name <chr>
```

열 결합(Join) - left_join()

key 역할을 할 컬럼을 지정하지 않으면 양쪽 데이터에서 컬럼 이름이 같은 모든 컬럼을 key로 자동 지정

```
left_join(flights2, weather)
```

```
## Joining, by = c("year", "month", "day", "hour", "origin")
```

```
## # A tibble: 336,776 x 18
##     year month   day  hour origin  dest tailnum carrier  temp  dewp humid
##    <dbl> <dbl> <int> <dbl>  <chr> <chr>   <chr>   <chr> <dbl> <dbl> <dbl>
##  1  2013     1     1     5    EWR   IAH  N14228      UA    NA    NA    NA
##  2  2013     1     1     5    LGA   IAH  N24211      UA    NA    NA    NA
##  3  2013     1     1     5    JFK   MIA  N619AA      AA    NA    NA    NA
##  4  2013     1     1     5    JFK   BQN  N804JB      B6    NA    NA    NA
##  5  2013     1     1     6    LGA   ATL  N668DN      DL 39.92 26.06 57.33
##  6  2013     1     1     5    EWR   ORD  N39463      UA    NA    NA    NA
##  7  2013     1     1     6    EWR   FLL  N516JB      B6 39.02 26.06 59.37
##  8  2013     1     1     6    LGA   IAD  N829AS      EV 39.92 26.06 57.33
##  9  2013     1     1     6    JFK   MCO  N593JB      B6 39.02 26.06 59.37
## 10  2013     1     1     6    LGA   ORD  N3ALAA      AA 39.92 26.06 57.33
## # ... with 336,766 more rows, and 7 more variables: wind_dir <dbl>,
## #   wind_speed <dbl>, wind_gust <dbl>, precip <dbl>, pressure <dbl>,
## #   visib <dbl>, time_hour <dttm>
```

열 결합(Join) - left_join()

여러 컬럼이 key로써 가능할 때 명시적인 지정이 있으면 작성된 컬럼만 key로 동작

```
left_join(flights2, planes, by = "tailnum")
```

```
## # A tibble: 336,776 x 16
##    year.x month   day  hour origin  dest tailnum carrier year.y
##     <int> <int> <int> <dbl>  <chr> <chr>   <chr>   <chr>  <int>
##  1   2013     1     1     5    EWR   IAH  N14228      UA   1999
##  2   2013     1     1     5    LGA   IAH  N24211      UA   1998
##  3   2013     1     1     5    JFK   MIA  N619AA      AA   1990
##  4   2013     1     1     5    JFK   BQN  N804JB      B6   2012
##  5   2013     1     1     6    LGA   ATL  N668DN      DL   1991
##  6   2013     1     1     5    EWR   ORD  N39463      UA   2012
##  7   2013     1     1     6    EWR   FLL  N516JB      B6   2000
##  8   2013     1     1     6    LGA   IAD  N829AS      EV   1998
##  9   2013     1     1     6    JFK   MCO  N593JB      B6   2004
## 10   2013     1     1     6    LGA   ORD  N3ALAA      AA     NA
## # ... with 336,766 more rows, and 7 more variables: type <chr>,
## #   manufacturer <chr>, model <chr>, engines <int>, seats <int>,
## #   speed <int>, engine <chr>
```

열 결합(Join) - left_join()

여러 컬럼이 key로 동작했을 때 데이터가 잘못 되는 예

```
left_join(flights2, planes)
```

```
## Joining, by = c("year", "tailnum")

## # A tibble: 336,776 x 15
##     year month   day  hour origin  dest tailnum carrier  type manufacturer
##    <int> <int> <int> <dbl>  <chr> <chr>   <chr>   <chr> <chr>          <chr>
##  1  2013     1     1     5    EWR   IAH  N14228      UA  <NA>           <NA>
##  2  2013     1     1     5    LGA   IAH  N24211      UA  <NA>           <NA>
##  3  2013     1     1     5    JFK   MIA  N619AA      AA  <NA>           <NA>
##  4  2013     1     1     5    JFK   BQN  N804JB      B6  <NA>           <NA>
##  5  2013     1     1     6    LGA   ATL  N668DN      DL  <NA>           <NA>
##  6  2013     1     1     5    EWR   ORD  N39463      UA  <NA>           <NA>
##  7  2013     1     1     6    EWR   FLL  N516JB      B6  <NA>           <NA>
##  8  2013     1     1     6    LGA   IAD  N829AS      EV  <NA>           <NA>
##  9  2013     1     1     6    JFK   MCO  N593JB      B6  <NA>           <NA>
## 10  2013     1     1     6    LGA   ORD  N3ALAA      AA  <NA>           <NA>
## # ... with 336,766 more rows, and 5 more variables: model <chr>,
## #   engines <int>, seats <int>, speed <int>, engine <chr>
```

열 결합(Join) - left_join()

컬럼 이름이 다를 때는 아래와 같은 문법을 사용

```
left_join(flights2, airports, c("dest" = "faa"))
```

```
## # A tibble: 336,776 x 15
##     year month   day  hour origin  dest tailnum carrier
##    <int> <int> <int> <dbl>  <chr> <chr>   <chr>   <chr>
##  1  2013     1     1     5    EWR   IAH  N14228      UA
##  2  2013     1     1     5    LGA   IAH  N24211      UA
##  3  2013     1     1     5    JFK   MIA  N619AA      AA
##  4  2013     1     1     5    JFK   BQN  N804JB      B6
##  5  2013     1     1     6    LGA   ATL  N668DN      DL
##  6  2013     1     1     5    EWR   ORD  N39463      UA
##  7  2013     1     1     6    EWR   FLL  N516JB      B6
##  8  2013     1     1     6    LGA   IAD  N829AS      EV
##  9  2013     1     1     6    JFK   MCO  N593JB      B6
## 10  2013     1     1     6    LGA   ORD  N3ALAA      AA
## # ... with 336,766 more rows, and 7 more variables: name <chr>, lat <dbl>,
## #   lon <dbl>, alt <int>, tz <dbl>, dst <chr>, tzone <chr>
```

열 결합(Join) - left_join()

rename()을 이용해 맞추는 방법도 가능

```
left_join(flights2, rename(airports, dest=faa), by="dest")
```

```
## # A tibble: 336,776 x 15
##     year month   day  hour origin  dest tailnum carrier
##    <int> <int> <int> <dbl>  <chr> <chr>   <chr>   <chr>
##  1  2013     1     1     5    EWR   IAH  N14228      UA
##  2  2013     1     1     5    LGA   IAH  N24211      UA
##  3  2013     1     1     5    JFK   MIA  N619AA      AA
##  4  2013     1     1     5    JFK   BQN  N804JB      B6
##  5  2013     1     1     6    LGA   ATL  N668DN      DL
##  6  2013     1     1     5    EWR   ORD  N39463      UA
##  7  2013     1     1     6    EWR   FLL  N516JB      B6
##  8  2013     1     1     6    LGA   IAD  N829AS      EV
##  9  2013     1     1     6    JFK   MCO  N593JB      B6
## 10  2013     1     1     6    LGA   ORD  N3ALAA      AA
## # ... with 336,766 more rows, and 7 more variables: name <chr>, lat <dbl>,
## #   lon <dbl>, alt <int>, tz <dbl>, dst <chr>, tzone <chr>
```

tidy data, long form과 wide form

tidy data + universe

tidyverse 패키지는

1. RStudio가 개발, 관리하는 패키지
2. 공식 문서가 매우 잘 되어 있음
3. 사용자층이 두터워 영어로 검색하면 많은 질답을 찾을 수 있음
4. 커뮤니티 설명글도 매우 많음
5. 6개의 핵심 패키지 포함 23가지 패키지로 이루어진 메타 패키지
6. tidy data 라는 사상과 파이프 연산자로 대동단결
7. 사상에 영감을 받아 맞춰서 제작하는 개인 패키지가 많음 (ex> tidyquant, tidytext 등)

```
if (!requireNamespace("tidyverse")){
  install.packages("tidyverse")}
```

```
## Loading required namespace: tidyverse
```

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr

## Conflicts with tidy packages ----------------------

## filter(): dplyr, stats
## lag():    dplyr, stats
```

tidy data 란

1. Hadley Wickham 2. 고감자님의 블로그 3. 헬로우데이터과학

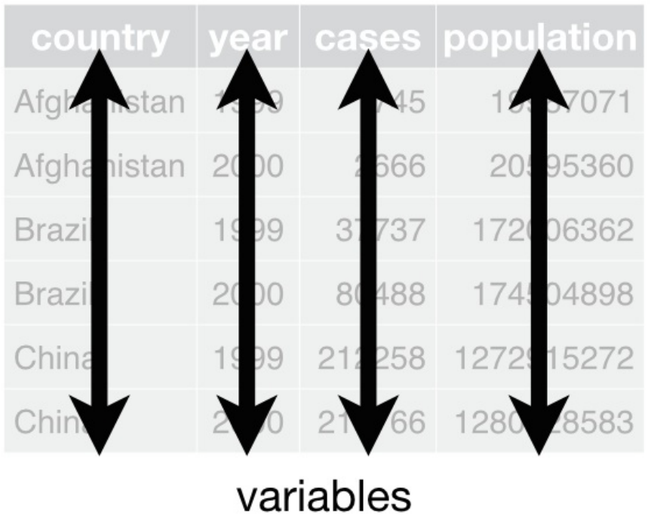
1.1 Each variable forms a column.
1.2 각 변수는 개별의 열(column)으로 존재한다.
1.3 각 열에는 개별 속성이 들어간다.

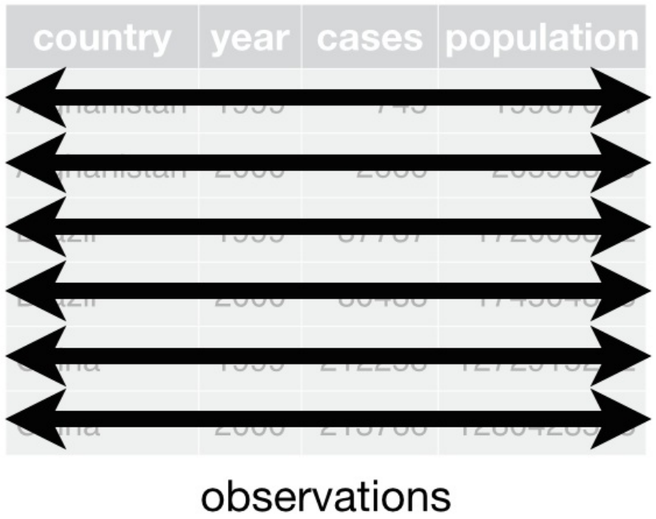
2.1 Each observation forms a row.
2.2 각 관측치는 행(row)를 구성한다.
2.3 각 행에는 개별 관찰 항목이 들어간다.

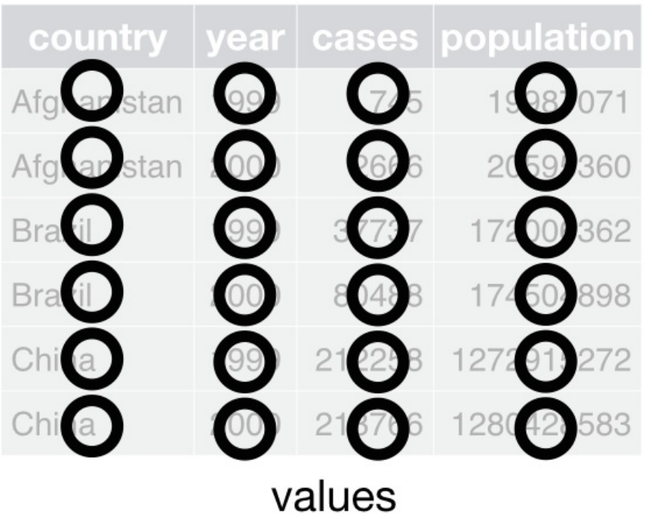
3.1 Each type of observational unit forms a table.
3.2 각 테이블은 단 하나의 관측기준에 의해서 조직된 데이터를 저장한다.
3.3 각 테이블에는 단일 유형의 데이터가 들어간다.

* 출처 : 금융데이터 분석을 위한 R 입문

tidy data 란







* 출처 : Garrett Grolemund의 Data Science with R 블로그

long form vs wide form

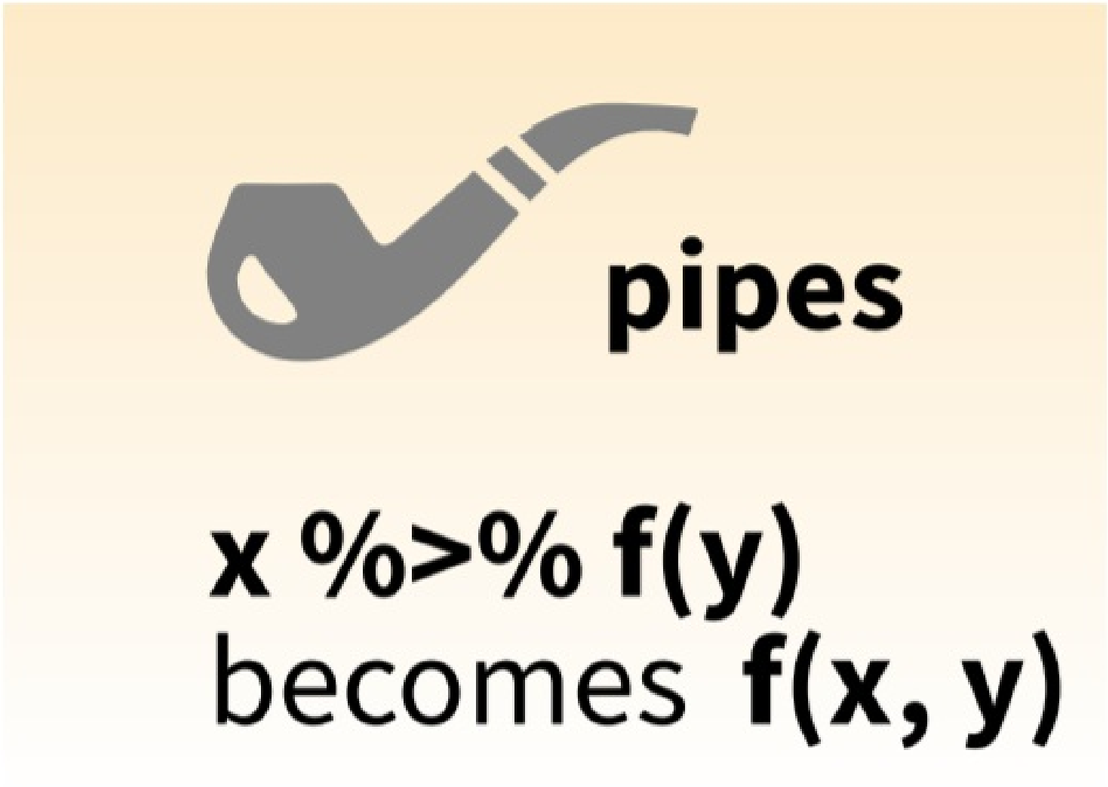
long form

1. 컴퓨터가 계산하기 좋은 모양
2. tidy data의 요건을 충족
3. tidyverse의 패키지 대부분의 입력 형태

wide form

1. 사람이 눈으로 보기 좋은 모양
2. 2개 변수에 대한 값만 확인 가능
3. dashboard 형이라고도 하며 조인 등 연산이 어려움

함수를 연결하는 파이프 연산자(%>%)



파이프 연산자(%>%)

함수를 중첩해서 사용할 일이 점점 빈번해 짐

```
plot(diff(log(sample(rnorm(10000,mean=10,sd=1),size=100,replace=FALSE))),col="red",type="l")
```

파이프 연산자(%>%)

함수를 중첩해서 사용할 일이 점점 빈번해 짐

```
plot(diff(log(sample(rnorm(10000,mean=10,sd=1),size=100,replace=FALSE))),col="red",type="l")
```

%>%를 사용하면

1. 생각의 순서대로 함수를 작성할 수 있음
2. 중간 변수 저장을 할 필요가 없음
3. 순서가 읽이 용이하여 기억하기 좋음

```
rnorm(10000,mean=10,sd=1) %>%
  sample(size=100,replace=FALSE) %>%
  log %>%
  diff %>%
  plot(col="red",type="l")
```

파이프 연산자(%>%)

flights 데이터에 파이프 연산자 사용예 1

```
flights %>%
  group_by(year,month,day) %>%
  summarise(delay=mean(dep_delay, na.rm = TRUE))
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [?]
##     year month   day     delay
##    <int> <int> <int>     <dbl>
##  1  2013     1     1 11.548926
##  2  2013     1     2 13.858824
##  3  2013     1     3 10.987832
##  4  2013     1     4  8.951595
##  5  2013     1     5  5.732218
##  6  2013     1     6  7.148014
##  7  2013     1     7  5.417204
##  8  2013     1     8  2.553073
##  9  2013     1     9  2.276477
## 10  2013     1    10  2.844995
## # ... with 355 more rows
```

파이프 연산자(%>%)

group_by()는 filter()와도 함께 사용할 수 있음

```
popular_dests <- flights %>%
  group_by(dest) %>%
  filter(n() > 365)
popular_dests
```

```
## # A tibble: 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
##  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 332,567 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>
```

파이프 연산자(%>%)

사용할 데이터부터 순서대로 함수를 작성할 수 있는 장점

```
popular_dests %>%
  filter(arr_delay > 0) %>%
  mutate(prop_delay = arr_delay / sum(arr_delay)) %>%
  select(year:day, dest, arr_delay, prop_delay)
```

```
## # A tibble: 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 1.106740e-04
##  2  2013     1     1   IAH        20 2.012255e-04
##  3  2013     1     1   MIA        33 2.350026e-04
##  4  2013     1     1   ORD        12 4.239594e-05
##  5  2013     1     1   FLL        19 9.377853e-05
##  6  2013     1     1   ORD         8 2.826396e-05
##  7  2013     1     1   LAX         7 3.444441e-05
##  8  2013     1     1   DFW        31 2.817951e-04
##  9  2013     1     1   ATL        12 3.996017e-05
## 10  2013     1     1   DTW        16 1.157257e-04
## # ... with 131,096 more rows
```


tidyr

tidyr이 데이터를 tidy하게 만드는 4개 함수

tidyr은 데이터를 tidy하게 만드는 4개 함수를 제공하고 추가적인 helper 함수를 함께 제공

1. gather() : wide form 데이터를 long form 으로 변환
2. spread() : long form 데이터를 wide form 으로 변환
3. separate() : 하나의 컬럼을 두 개로 나눔
4. unite() : 두 개의 컬럼을 하나로 합침

tidyr 준비

tidyr, dplyr은 tidyverse에 포함된 패키지이기 때문에 tidyverse를 설치하고 불러왔다면 생략가능

```
if (!requireNamespace("tidyr")) install.packages("tidyr")
library(tidyr)
```

데이터 소개

tidyr 패키지는 패키지의 동작을 설명하기 위해 내장 데이터를 준비하고 있음

```
table1
```

```
## # A tibble: 6 x 4
##       country  year  cases population
##         <chr> <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3      Brazil  1999  37737  172006362
## 4      Brazil  2000  80488  174504898
## 5       China  1999 212258 1272915272
## 6       China  2000 213766 1280428583
```

데이터 소개

long form 예시

```
table2
```

```
## # A tibble: 12 x 4
##         country  year       type       count
##           <chr> <int>      <chr>       <int>
##  1 Afghanistan  1999      cases         745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000      cases        2666
##  4 Afghanistan  2000 population   20595360
##  5       Brazil  1999      cases       37737
##  6       Brazil  1999 population  172006362
##  7       Brazil  2000      cases       80488
##  8       Brazil  2000 population  174504898
##  9        China  1999      cases      212258
## 10        China  1999 population 1272915272
## 11        China  2000      cases      213766
## 12        China  2000 population 1280428583
```

데이터 소개

한 컬럼에 두 개의 의미를 지닌 데이터가 들어 있는 경우

```
table3
```

```
## # A tibble: 6 x 3
##       country  year              rate
## *       <chr> <int>             <chr>
## 1 Afghanistan  1999      745/19987071
## 2 Afghanistan  2000     2666/20595360
## 3      Brazil  1999   37737/172006362
## 4      Brazil  2000   80488/174504898
## 5       China  1999 212258/1272915272
## 6       China  2000 213766/1280428583
```

데이터 소개

wide form 예시 1

```
table4a
```

```
## # A tibble: 3 x 3
##       country `1999` `2000`
## *       <chr>  <int>  <int>
## 1 Afghanistan    745   2666
## 2      Brazil  37737  80488
## 3       China 212258 213766
```

데이터 소개

wide form 예시 2

```
table4b
```

```
## # A tibble: 3 x 3
##       country     `1999`     `2000`
## *       <chr>      <int>      <int>
## 1 Afghanistan   19987071   20595360
## 2      Brazil  172006362  174504898
## 3       China 1272915272 1280428583
```

wide to long - gather()

gather()는 wide form의 데이터를 long form으로 바꾸는 역할을 수행. gather(data, key = "컬럼 이름이 데이터로 들어갈 그 컬럼의 이름", value = "매트릭스로 펼쳐져 있는 데이터가 모이는 컬럼의 이름", "데이터로 들어갈 컬럼들을 지정")의 형태로 작성. "데이터로 들어갈 컬럼들을 지정"은 위치에 자유로움.

값에 해당하는 데이터의 이동이 중요함. 메트릭스 모양이 한 줄의 컬럼으로 변경되는 것을 확인

```
table4a
```

```
## # A tibble: 3 x 3
##       country `1999` `2000`
## *       <chr>  <int>  <int>
## 1 Afghanistan    745   2666
## 2      Brazil  37737  80488
## 3       China 212258 213766
```

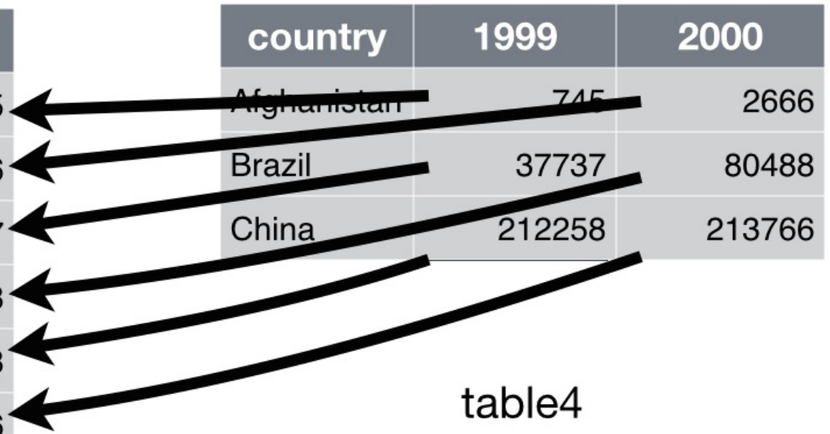
```
table4a %>%
  gather(`1999`, `2000`
         , key = "year"
         , value = "cases")
```

```
## # A tibble: 6 x 3
##       country  year  cases
##         <chr> <chr>  <int>
## 1 Afghanistan  1999    745
## 2      Brazil  1999  37737
## 3       China  1999 212258
## 4 Afghanistan  2000   2666
## 5      Brazil  2000  80488
## 6       China  2000 213766
```

gather()의 동작

값에 해당하는 데이터는 matrix -> column, 지정한 컬럼들은 key의 데이터로 변경

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766



country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

long to wide - spread()

spread()는 하나의 컬럼으로 되어 있는 데이터를 메트릭스의 형태로 **흩뿌리는** 동작을 수행. spread(data, key = "컬럼에 위치할 데이터가 있는 컬럼", value = "메트릭스 모양이로 펼쳐질 데이터가 있는 컬럼") 으로 작성

```
table2
```

```
## # A tibble: 12 x 4
##         country  year       type      count
##           <chr> <int>      <chr>      <int>
##  1 Afghanistan  1999      cases        745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000      cases       2666
##  4 Afghanistan  2000 population   20595360
##  5       Brazil  1999      cases      37737
##  6       Brazil  1999 population  172006362
##  7       Brazil  2000      cases      80488
##  8       Brazil  2000 population  174504898
##  9        China  1999      cases     212258
## 10        China  1999 population 1272915272
## 11        China  2000      cases     213766
## 12        China  2000 population 1280428583
```

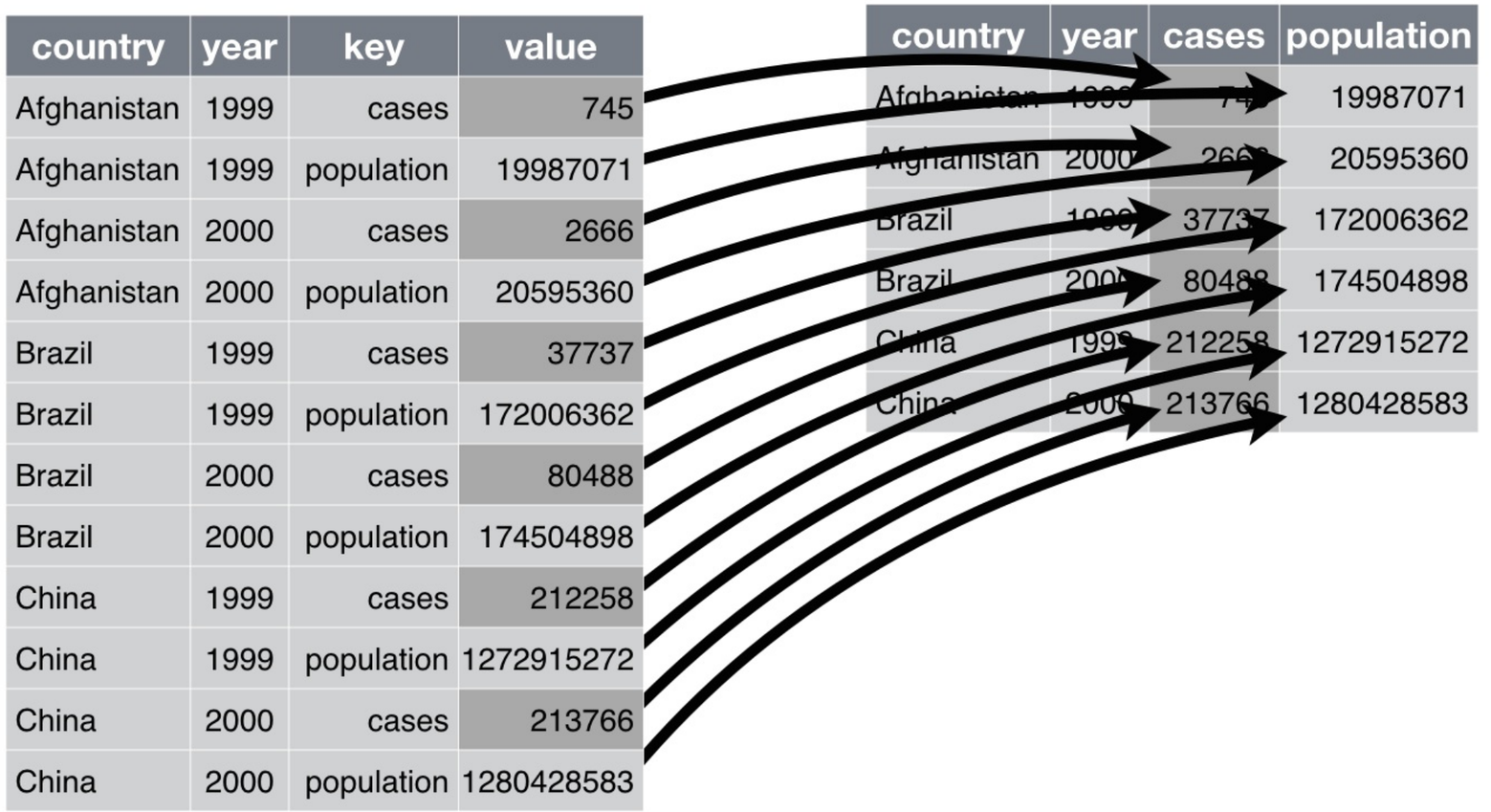
```
table2 %>%
  spread(key = type, value = count)
```

```
## # A tibble: 6 x 4
##       country  year  cases population
## *       <chr> <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3      Brazil  1999  37737  172006362
## 4      Brazil  2000  80488  174504898
## 5       China  1999 212258 1272915272
## 6       China  2000 213766 1280428583
```

spread()의 동작

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2



하나의 컬럼을 나누기 - separate()

아래와 같이 여러 부호로 그 의미가 나누어져있지만 한 컬럼에 데이터가 있는 경우 컬럼을 의미 단위로 분리하는 역할을 수행. into = c("나눠질 때 첫번째 컬럼 이름","나눠질 때 두번째 컬럼 이름")으로 새로 생성되는 컬럼의 이름을 지정할 수 있음

```
table3
```

```
## # A tibble: 6 x 3
##       country  year              rate
## *       <chr> <int>             <chr>
## 1 Afghanistan  1999      745/19987071
## 2 Afghanistan  2000     2666/20595360
## 3      Brazil  1999   37737/172006362
## 4      Brazil  2000   80488/174504898
## 5       China  1999 212258/1272915272
## 6       China  2000 213766/1280428583
```

```
table3 %>%
  separate(rate
           ,into = c("cases", "population"))
```

```
## # A tibble: 6 x 4
##       country  year  cases population
## *       <chr> <int>  <chr>      <chr>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3      Brazil  1999  37737  172006362
## 4      Brazil  2000  80488  174504898
## 5       China  1999 212258 1272915272
## 6       China  2000 213766 1280428583
```

간단한 형변환은 옵션으로 제공

```
table3 %>%
  separate(rate
   , into = c("cases"
         , "population")
         )
```

```
## # A tibble: 6 x 4
##       country  year  cases population
## *       <chr> <int>  <chr>      <chr>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3      Brazil  1999  37737  172006362
## 4      Brazil  2000  80488  174504898
## 5       China  1999 212258 1272915272
## 6       China  2000 213766 1280428583
```

```
table3 %>%
  separate(rate
   , into = c("cases"
         , "population")
         , convert = TRUE)
```

```
## # A tibble: 6 x 4
##       country  year  cases population
## *       <chr> <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3      Brazil  1999  37737  172006362
## 4      Brazil  2000  80488  174504898
## 5       China  1999 212258 1272915272
## 6       China  2000 213766 1280428583
```

두 컬럼을 합치기 - unite()

unite()는 두 컬럼을 paste0()와 비슷하게 합쳐주는 역할을 수행

```
table5
```

```
## # A tibble: 6 x 4
##       country century  year              rate
## *       <chr>   <chr> <chr>             <chr>
## 1 Afghanistan      19    99      745/19987071
## 2 Afghanistan      20    00     2666/20595360
## 3      Brazil      19    99   37737/172006362
## 4      Brazil      20    00   80488/174504898
## 5       China      19    99 212258/1272915272
## 6       China      20    00 213766/1280428583
```

```
table5 %>%
  unite(new, century, year)
```

```
## # A tibble: 6 x 3
##       country   new              rate
## *       <chr> <chr>             <chr>
## 1 Afghanistan 19_99      745/19987071
## 2 Afghanistan 20_00     2666/20595360
## 3      Brazil 19_99   37737/172006362
## 4      Brazil 20_00   80488/174504898
## 5       China 19_99 212258/1272915272
## 6       China 20_00 213766/1280428583
```

구분자 지정

sep 인자를 이용해 구분자로 사용할 문자열을 지정할 수 있음

```
table5 %>%
  unite(new, century, year)
```

```
## # A tibble: 6 x 3
##       country   new              rate
## *       <chr> <chr>             <chr>
## 1 Afghanistan 19_99       745/19987071
## 2 Afghanistan 20_00      2666/20595360
## 3      Brazil 19_99    37737/172006362
## 4      Brazil 20_00    80488/174504898
## 5       China 19_99 212258/1272915272
## 6       China 20_00 213766/1280428583
```

```
table5 %>%
  unite(new, century, year, sep = "")
```

```
## # A tibble: 6 x 3
##       country   new              rate
## *       <chr> <chr>             <chr>
## 1 Afghanistan  1999       745/19987071
## 2 Afghanistan  2000      2666/20595360
## 3      Brazil  1999    37737/172006362
## 4      Brazil  2000    80488/174504898
## 5       China  1999 212258/1272915272
## 6       China  2000 213766/1280428583
```

데이터 소스에 연결하기

데이터 소스로서 DBI

DBI 패키지가 연결하는 database의 연결정보를 바탕으로 dplyr 문법을 사용할 수 있습니다. 그렇게 하기 위해서는 dbplyr 패키지가 필요합니다.

```
if (!requireNamespace("dbplyr")) install.packages("dbplyr")
```

```
## Loading required namespace: dbplyr

## Installing package into 'C:/Users/mrchypark/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)

## package 'dbplyr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##     C:\Users\mrchypark\AppData\Local\Temp\RtmpGkwJOm\downloaded_packages
```

```
library(dbplyr)
```

```
##
## Attaching package: 'dbplyr'

## The following objects are masked from 'package:dplyr':
##
##     ident, sql
```

data.table

데이터 소스로서 data.table을 사용할 수 있습니다. data.table을 사용하기 위해서는 dtplyr 패키지를 설치해야 합니다.
data.table의 독립적인 동작은 cheat sheet을 확인해주세요.

```
if (!requireNamespace("dtplyr")) install.packages("dtplyr")
```

```
## Loading required namespace: dtplyr

## Installing package into 'C:/Users/mrchypark/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)

## package 'dtplyr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##     C:\Users\mrchypark\AppData\Local\Temp\RtmpGkwJOm\downloaded_packages
```

```
library(dtplyr)
```

R 데이터를 DB 테이블로 만들기 copy_to()

copy_to()는 DBI의 dbWriteTable()과 같은 기능을 수행. dplyr 패키지에 속한 copy_to()는 성능 개선을 통해 dbWriteTable() 보다 빠른 속도를 제공함

```
library(RSQLite)
library(nycflights13)

conn <- dbConnect(SQLite())

copy_to(conn
        , flights
        , temporary = FALSE
        , name = 'flights')
dbListTables(conn)
```

```
## [1] "flights"      "sqlite_stat1" "sqlite_stat4"
```

테이블의 연결정보를 R 객체에 저장 - tbl()

dbplyr과 DBI, dplyr로 데이터베이스의 테이블을 dplyr 문법으로 다루기 위해서는 DBI 패키지에서 conn 객체와 같이 테이블의 연결정보를 담고 있는 R 객체가 필요. tbl()는 DB내 테이블 연결정보를 R 객체로 만드는 함수

```
tb_flights <- tbl(conn, "flights")
```

속도를 빠르게 하는 indexes 옵션

copy_to()를 진행할 때 key 역할을 수행할 컬럼을 미리 지정해주면 관련 컬럼을 사용하는 연산(group_by에 key 컬럼 사용 등)에서 속도를 높일 있음

```
copy_to(conn,
        flights,
        name = 'flights_idx',
        temporary = FALSE,
        indexes = list("carrier"))

tb_flights <- tbl(conn, "flights")
tb_flights_idx <- tbl(conn, "flights_idx")
```

함수 속도를 비교 - microbenchmark()

함수의 속도와 결과를 비교해서 같은 결과에 빠른 속도의 함수를 사용하기 위해 비교 테스트를 진행

```
if (!requireNamespace("microbenchmark")) install.packages("microbenchmark")
```

```
## Loading required namespace: microbenchmark

## Installing package into 'C:/Users/mrchypark/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)

## package 'microbenchmark' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##     C:\Users\mrchypark\AppData\Local\Temp\RtmpGkwJOm\downloaded_packages
```

```
library(microbenchmark)
```

indexes 속도 비교

```
microbenchmark(tbl(conn, 'flights') %>%
                 group_by(carrier) %>%
                 summarise(count = n()) %>%
                 collect(),
               tbl(conn, 'flights_idx') %>%
                 group_by(carrier) %>%
                 summarise(count = n()) %>%
                 collect(),
               times = 10)
```

```
## Unit: milliseconds
##                                                                                                 expr
##      tbl(conn, "flights") %>% group_by(carrier) %>% summarise(count = n()) %>%      collect()
##  tbl(conn, "flights_idx") %>% group_by(carrier) %>% summarise(count = n()) %>%      collect()
##         min         lq       mean    median        uq       max neval
##  159.51046 177.37830 205.03707 194.09691 234.1822 290.87159    10
##   37.43287  37.84492  47.45835  44.17065  53.8540  68.35395    10
```

collect()

collect()는 DB에 전달하는 명령의 최종 결과를 R 객체로 가져오는 역할을 수행합니다.

```
tbl(conn, 'flights') %>%
                  group_by(carrier) %>%
                  summarise(count = n()) %>%
                  collect()
```

```
## # A tibble: 16 x 2
##    carrier count
##      <chr> <int>
##  1      9E 18460
##  2      AA 32729
##  3      AS   714
##  4      B6 54635
##  5      DL 48110
##  6      EV 54173
##  7      F9   685
##  8      FL  3260
##  9      HA   342
## 10      MQ 26397
## 11      OO    32
## 12      UA 58665
## 13      US 20536
## 14      VX  5162
## 15      WN 12275
## 16      YV   601
```

결과를 테이블로 저장 - compute()

compute()는 collect()와는 달리 연산된 결과를 R 객체로 저장하는 것이 아니라 새로 이름지은 테이블로 DB에 저장하는 동작을 수행

```
dbListTables(conn)
```

```
## [1] "flights"      "flights_idx"  "sqlite_stat1" "sqlite_stat4"
```

```
tbl(conn, 'flights') %>%
  group_by(tailnum) %>%
  summarise(count=n(),
            mean_distance = mean(distance),
            total_distance = sum(distance)) %>%
  filter(!is.na(tailnum)) %>%
  compute(name = 'planes_distance')
```

```
## # Source:   table<planes_distance> [?? x 4]
## # Database: sqlite 3.19.3 []
##    tailnum count mean_distance total_distance
##      <chr> <int>         <dbl>          <dbl>
##  1  D942DN     4      854.5000           3418
##  2  N0EGMQ   371      676.1887         250866
##  3  N10156   153      757.9477         115966
##  4  N102UW    48      535.8750          25722
##  5  N103US    46      535.1957          24619
##  6  N104UW    47      535.2553          25157
##  7  N10575   289      519.7024         150194
```

테이블 저장 결과 확인

```
dbListTables(conn)
```

```
## [1] "flights"         "flights_idx"     "planes_distance" "sqlite_stat1"
## [5] "sqlite_stat4"
```

```
dbReadTable(conn, "planes_distance")
```

```
##       tailnum count mean_distance total_distance
## 1      D942DN     4      854.5000           3418
## 2      N0EGMQ   371      676.1887         250866
## 3      N10156   153      757.9477         115966
## 4      N102UW    48      535.8750          25722
## 5      N103US    46      535.1957          24619
## 6      N104UW    47      535.2553          25157
## 7      N10575   289      519.7024         150194
## 8      N105UW    45      524.8444          23618
## 9      N107US    41      528.7073          21677
## 10     N108UW    60      534.5000          32070
## 11     N109UW    48      535.8750          25722
## 12     N110UW    40      535.3750          21415
## 13     N11106   129      771.4109          99512
## 14     N11107   148      705.8649         104468
## 15     N11109   148      714.0000         105672
## 16     N11113   138      719.7754          99329
## 17     N11119   148      723.3851         107061
## 18     N11121   154      719.3701         110783
## 19     N11127   124      748.1129          92766
## 20     N11137   112      726.5982          81379
```

show_query()

show_query()는 dplyr로 구성된 함수의 연결이 query문으로 어떻게 변환되는지를 보여줌

```
copy_to(conn, planes, name = 'planes', temporary = FALSE)
tbl(conn, 'planes_distance') %>%
  inner_join(tbl(conn, 'planes'), by='tailnum') %>%
  arrange(desc(total_distance)) %>%
  select(total_distance, manufacturer, model) %>%
  show_query()
```

```
## <SQL>
## SELECT `total_distance` AS `total_distance`, `manufacturer` AS `manufacturer`, `model` AS `model`
## FROM (SELECT *
## FROM (SELECT `TBL_LEFT`.`tailnum` AS `tailnum`, `TBL_LEFT`.`count` AS `count`, `TBL_LEFT`.`mean_dist
##   FROM `planes_distance` AS `TBL_LEFT`
##   INNER JOIN `planes` AS `TBL_RIGHT`
##   ON (`TBL_LEFT`.`tailnum` = `TBL_RIGHT`.`tailnum`)
## )
## ORDER BY `total_distance` DESC)
```

과제

- recomen 폴더에 있는 6개 데이터를 활용해서 다음장의 6개 질문에 답해주세요.
- 데이터가 5개이신 분은 아래 코드를 실행해서 다운로드해주세요. 1.4G라 시간이 좀 걸립니다.

```
chk<-file.info("./data/recomen/tran.csv")
if(is.na(chk$size)){
  recoment<-"http://rcoholic.duckdns.org/oc/index.php/s/jISrPutj4ocLci2/download"
  dir.create("./data", showWarnings = F)
  dir.create("./data/recomen", showWarnings = F)
  download.file(recoment,destfile="./data/recomen/tran.csv",mode='wb')
}
```

- 답을 구하기 위한 코드와 답을 class3assignment 폴더에 class3_[이름].R로 제출해주세요.(답은 주석으로 작성)
- sql, dplyr+tidyr, data.table 등 무엇이든 사용하시고, 외부서비스도 가능하시면 무엇이든 사용하세요. 몇 문제는 계산 시간이 오래걸릴 수 있습니다.

문제

1. receiptNum가 "6998419"인 구매기록의 가격(amout)의 합은 얼마인가요?
2. 가장 비싼 item은 무엇인가요?
3. 사용자들이 가장 많이 사용한 체널은 mobile/app과 onlinemall 중에 무엇입니까?
4. 월매출이 2015년 03월 가장 높은 매장의 storeCode는 무엇인가요?
5. 경쟁사의 이용기록이 가장 많은 사용자의 성별은 무엇입니까? (competitor 데이터에서 1row가 1건이라고 가정)
6. 한번에 3개 이상 구매한 경우에 가장 많이 구매에 포함된 제품 카테고리(cate_3)는 무엇입니까?