# M51-tidyr

LearningSpoonsR

2019-02-11

Part 0. Setup

Part I. **join**: 두 개의 데이터 프레임을 합하는 법 (a.k.a. **merge**)

Part II. Workding with “tidy” data

# Part 0. Setup

```r
library(tidyverse) # Wickham's
library(sqldf)
source("infile-tidyr.R")
```

1. `tidyverse`
   - Wickham이 만든 packages들을 다 모아놓은 패키지
   - **ggplot2**, **dplyr**, **tidyr**, **readr**, **purrr**, **tibble**, **stringr**, **forcats**
   - https://www.tidyverse.org/packages/
2. `sqldf`
   - R에서 SQL 명령어를 사용할 수 있게 해주는 패키지
   - SQL은 대용량의 복잡한 데이터를 다루는 데에 적합한 언어
   - 이런 Cross-Language 패키지들은 새로운 환경에서의 연착륙을 도와줌
3. `source("infile-tidyr.R")`
   - 해당 R 소스코드를 실행한 효과가 나옴
   - 긴 코드를 보이지 않게 숨기게 하는데에 유용함
   - 이번 노트에 사용되는 데이터프레임들을 정의하는 코드
   - 아래와 같은 코드가 들어있습니다.

```r
df1 <- data.frame(
  CustomerId = c(1:5),
  Product = c(rep("Toaster", 3), rep("Radio", 2)))
df2 <- data.frame(
  CustomerId = c(2, 4, 6),
  State = c(rep("Seoul", 2), rep("Busan", 1)))
```

# Part I. join: 두 개의 데이터 프레임을 합하는 법 (a.k.a. merge)

# 0. df1과 df2를 어떻게 합해야 할까요?

```
df1
```

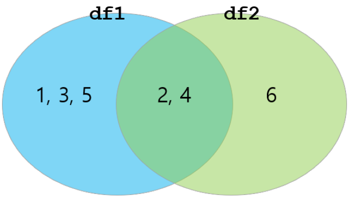
```
##   CustomerId Product
## 1          1 Toaster
## 2          2 Toaster
## 3          3 Toaster
## 4          4   Radio
## 5          5   Radio
```

```
df2
```

```
##   CustomerId State
## 1          2 Seoul
## 2          4 Seoul
## 3          6 Busan
```

- 어느쪽이 primary?
- 결측치가 생겨도 상관없음?

- join에는 4가지 방법이 있습니다.



1. inner
2. full (outer)
3. left
4. right

## 1. Inner Join

```
inner_join(df1, df2) # dplyr
merge(x = df1, y = df2, by = "CustomerId") # base
sqldf("SELECT CustomerId, Product, State
       FROM df1 JOIN df2 USING(CustomerID)") # sqldf
```

```
## Joining, by = "CustomerId"
##   CustomerId Product State
## 1          2 Toaster Seoul
## 2          4   Radio Seoul
```

## 2. Outer Join (full)

```
full_join(df1, df2) # dplyr
merge(x = df1, y = df2, by = "CustomerId", all = TRUE) # base
```

```
## Joining, by = "CustomerId"
##   CustomerId Product State
## 1          1 Toaster  <NA>
## 2          2 Toaster Seoul
## 3          3 Toaster  <NA>
## 4          4   Radio Seoul
## 5          5   Radio  <NA>
## 6          6    <NA> Busan
```

## 3. Left Join

```
left_join(df1, df2) # dplyr
merge(x = df1, y = df2, by = "CustomerId", all.x = TRUE) # base
sqldf("SELECT CustomerId, Product, State
       FROM df1 LEFT JOIN df2 USING(CustomerID)") # sqldf
```

```
## Joining, by = "CustomerId"
##   CustomerId Product State
## 1          1 Toaster  <NA>
## 2          2 Toaster Seoul
## 3          3 Toaster  <NA>
## 4          4   Radio Seoul
## 5          5   Radio  <NA>
```

## 4. Right Join

```
right_join(df1, df2) # dplyr
merge(x = df1, y = df2, by = "CustomerId", all.y = TRUE) # base
```

```
## Joining, by = "CustomerId"
##   CustomerId Product State
## 1          2 Toaster Seoul
## 2          4   Radio Seoul
## 3          6    <NA> Busan
```

## Summary

- Summary

```r
inner_join(df1, df2)
left_join(df1, df2)
full_join(df1, df2)
right_join(df1, df2)
```

- `join`할때 사용할 `key`변수를 구체화
  - `by =` argument로 key 변수를 아래처럼 지정합니다.
  - `by =`을 입력하지 않으면 같은 이름의 변수를 찾아서 key로 사용합니다.

```r
inner_join(df1, df2)
inner_join(x=df1, y=df2)
inner_join(x=df1, y=df2, by = "CustomerId")
inner_join(x=df1, y=df2, by = c("CustomerId"))
inner_join(x=df1, y=df2, by = c("CustomerId"="CustomerId"))
```

- `vlookup`이나 `index-match`함수를 이용해서 엑셀 파일 합해본 경험있으세요?
- R에서는 이게 정말 끝입니다.

## Discussion

- 여러분의 데이터가…
  - 각각의 관찰값에 1개 혹은 2개의 key 변수를 가지고 관리되고 있습니까?
  - → (아니라면 직관적인 처리를 위해서 만드는 것이 좋습니다.)
  - 여러 분의 key변수는 알기 쉽고, 중복되지 않고, 체계적인 규칙을 가지고 있습니까?
  - → (주민등록번로를 생각해보세요!)

# Appendix – NA의 처리

```
df3 <- full_join(df1, df2)
```

```
## Joining, by = "CustomerId"
```

```
df3$Population <- c(NA, 1000, NA, 1000, NA, 200)
df3
```

```
##   CustomerId Product State Population
## 1          1 Toaster  <NA>         NA
## 2          2 Toaster Seoul       1000
## 3          3 Toaster  <NA>         NA
## 4          4   Radio Seoul       1000
## 5          5   Radio  <NA>         NA
## 6          6    <NA> Busan        200
```

- 여러가지 이유로 위처럼 NA (결측치)가 생깁니다.
- 주로 3가지 방법으로 해결합니다.
  1. 결측치가 있는 관찰값을 제거하기
  2. 결측치를 다른 수치로 대체하기 (평균, 0 등의 값)
  3. 그대로 두고 함수를 적용할 때 주의해서 분석 (많은 함수들에서 na.rm=TRUE등의 옵션 사용이 가능합니다.)

1. 결측치가 있는 관찰값을 제거하기

```
is.na(df3$State)
```

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

```
is.na(df3$State) %>% which()
```

```
## [1] 1 3 5
```

```
df3[-(is.na(df3$State) %>% which()),]
```

```
##   CustomerId Product State Population
## 2          2 Toaster Seoul       1000
## 4          4   Radio Seoul       1000
## 6          6    <NA> Busan        200
```

- 아래 명령들도 같은 결과를 만들어 냅니다.

```
df3[!is.na(df3$State),]
df3[which(!is.na(df3$State)),]
```

2. 결측치를 다른 수치로 대체하기 (평균, 0등의 값)

```
is.na(df3$Population) %>% which()
```

```
## [1] 1 3 5
```

```
mean(df3$Population)
```

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

```
mean(df3$Population, na.rm = TRUE)
```

```
## [1] 733.3333
```

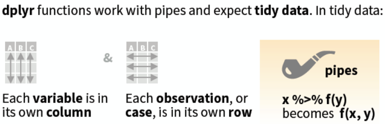
```
df3$Population[is.na(df3$Population) %>% which()] <- mean(df3$Population, na.rm = TRUE)
df3
```

```
##   CustomerId Product State Population
## 1          1 Toaster  <NA>   733.3333
## 2          2 Toaster Seoul  1000.0000
## 3          3 Toaster  <NA>   733.3333
## 4          4   Radio Seoul  1000.0000
## 5          5   Radio  <NA>   733.3333
## 6          6    <NA> Busan   200.0000
```

# Part II. Workding with “tidy” data

# 0. 단정한 데이터?

- M21 p.17
- tidy data.frame!
  1. 개체 타입은 data.frame
  2. 각각의 row는 관찰값을 의미
  3. 각각의 column은 변수를 의미



**dplyr** functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

x %>% f(y) becomes f(x, y)

Figure 1: from dplyr Cheatsheet

```
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
```

- table1과 같은 정보를 담고 있지만, tidy하게 되있지 않은 데이터 구조가 있습니다.
- 이들을 tidy하게 table1 모양으로 바꿉니다.
- Excel의 pivot_table 기능

# 0. 목 적

- 각 국가의 연도별 범죄 건수와 인구수입니다.
- table2, table3, table4로 부터 table1모양을 만들어야 했던 아픈 기억이 있으신가요?

- Before

```
table4a
```

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

```
table4b
```

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

- After

```
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
```

- Before

```
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
```

- After

```
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
```

- Before

```
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
```

- After

```
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
```

# 1. dplyr Review (mutate)

```
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
```

```
table1 %>% mutate(rate = cases / population * 100)
```

```
## # A tibble: 6 x 5
##   country      year  cases population    rate
##   <chr>       <int>  <int>      <int>   <dbl>
## 1 Afghanistan  1999    745   19987071 0.00373
## 2 Afghanistan  2000   2666   20595360 0.0129
## 3 Brazil       1999  37737  172006362 0.0219
## 4 Brazil       2000  80488  174504898 0.0461
## 5 China        1999 212258 1272915272 0.0167
## 6 China        2000 213766 1280428583 0.0167
```

# 1. dplyr Review (group_by & summarise)

```
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
```

```
table1 %>% group_by(year) %>% summarise(n = sum(cases))
table1 %>% count(year, wt = cases) # same as above
```

```
## # A tibble: 2 x 2
##    year      n
##   <int>  <int>
## 1  1999 250740
## 2  2000 296920
```

# 2. gather from table4a & table4b

```
table4a
```

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

```
tidy4a <- table4a %>%
  gather(colnames(table4a)[-1],
         key = "year",
         value = "cases")
tidy4a
```

```
## # 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
```

```
table4b
```

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

```
tidy4b <- table4b %>%
  gather(colnames(table4b)[-1],
         key = "year",
         value = "population")
tidy4b
```

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

- **gather()** 함수로 tidy 해집니다.
- 이제 **tidy4a**와 **tidy4b**로 어떻게 하면 **table1**이 만들어 질까요??

```r
inner_join(tidy4a, tidy4b)
inner_join(tidy4a, tidy4b, by = c("country", "year"))
inner_join(tidy4a, tidy4b, by = c("country"="country", "year"="year"))
```

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

## 3. spread from table2

```
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
```

# 4. separate from table3

```
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"), sep = "/")
```

```
## # 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
```

참고: Classical method

```
table3$cases <-
  sapply(strsplit(table3$rate, split = "/"), function(x) x[1])
table3$popul <-
  sapply(strsplit(table3$rate, split = "/"), function(x) x[2])
```