Tidy

박찬영

2024-08-23

tidyverse library를 사용합니다.

티블이 뭐야?

데이터프레임을 확장한 것, 스타크래프트 브루드 워 같은 거라고 할 수 있다. data.frame() 에 비해 tibble() 은 동작의 규모가 훨씬 작다는 것에 주의해야 한다. 즉, 입력의 유형을 절대로 변경하지 않고 (예를 들어, 문자열을 팩터형으로 변환하지 않는다!), 변수의 이름을 바꾸거나, 행이름을 생성하지 않는다.

```
ir=as_tibble(iris) #티블로 변환

t=tibble(x=1:5, y=1, z=x^2+y) #이렇게도 만들 수 있음

tribble(
    ~x, ~y, ~z,
    #--/--/----
    "a", 1, 3.6,
    "b", 65, 7.7
) #이렇게도 만들 수 있음2 *tribble 임
```

```
## # A tibble: 2 x 3
##   x         y     z
##   <chr> <dbl> <dbl>
## 1 a         1   3.6
## 2 b        65   7.7
```

```
t[1] #하나만 넣으면 열번호
```

```
## # A tibble: 5 x 1
##       x
##   <int>
## 1     1
## 2     2
```

```
## 3     3
## 4     4
## 5     5
```

```
t[3,1] #두개 넣으면 행 열
```

```
## # A tibble: 1 x 1
##       x
##   <int>
## 1     3
```

```
t[[1]]
```

```
## [1] 1 2 3 4 5
```

```
t[[3,1]]
```

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

```
#대괄호 하나는 데이터프레임의 일부
#대괄호 두개는 벡터로의 추출
t$x #이것도 벡터로의 추출
```

```
## [1] 1 2 3 4 5
```

```
t %>% .$x #파이프 사용하려면 .붙이기
```

```
## [1] 1 2 3 4 5
```

```
t %>% .[[1]]
```

```
## [1] 1 2 3 4 5
```

벡터 파싱!

벡터를 parser 해봅시다

```
parse_logical(c("TRUE","FALSE","NA"))
```

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

```
parse_integer(c("1","2","3"))
```

```
## [1] 1 2 3
```

```
parse_date(c("2010-01-01", "1979-10-14"))
```

```
## [1] "2010-01-01" "1979-10-14"
```

```
#parse_*() 함수는 문자열데이터를 다른 자료형으로 바꾼다. json 데이터에 유용

parse_integer(c("1", "231", ".", "456"), na = ".") #na= 인수는 결측치로 표시할 거 정하는 것
```

```
## [1]   1 231  NA 456
```

```
#문자열을 수치형으로 변환하는건 어려움. 왜? 천단위구분기호나 %같은 단위때문

parse_double("1.23")
```

```
## [1] 1.23
```

```
parse_double("1,23" ,locale=locale(decimal_mark=",")) #locale 설정으로 소수점 구분기호 바꾸기
```

```
## [1] 1.23
```

```
#parse_double은 엄격하다
#parse_number는 약하다. 문자열안에 포함된 수를 그냥 파싱한다.
parse_number("It cost $123.45")
```

```
## [1] 123.45
```

```
#천단위 구분기호를 locale을 이용해 바꿔보자

# 미국 방식
parse_number("$123,456,789")
```

```
## [1] 123456789
```

```
# 유럽의 많은 국가 방식
parse_number("123.456.789", locale = locale(grouping_mark = "."))
```

```
## [1] 123456789
```

```
# 스위스 방식
parse_number("123'456'789", locale = locale(grouping_mark = "'"))
```

```
## [1] 123456789
```

```
#문자열의 파싱에는 복잡성이 있다. 그것은 바로 인코딩
#R은 기본적으로 UTF-8 인코딩을 사용한다.
#언어 서버가 에러가 생겨 vs-code말고 Rstudio에서 테스트 해보시길

#팩터형 파싱을 해보자
fruit <- c("apple", "banana","melon")
parse_factor(c("apple", "banana"), levels = fruit) #팩터생성과 유사함
```

```
## [1] apple  banana
## Levels: apple banana melon
```

```
#날짜 데이터 파싱
#원하는 것이 date (1970-01-01 이후의 일 수)
#date-time (1970-01-01 자정 이후의 초 수)
#time (자정 이후의 초 수)인지에 따라 세 가지 파서 중에서 선택하면 된다.

parse_datetime("20240823") #이런 느낌
```

```
## [1] "2024-08-23 UTC"
```

도전 문제풀이!

```
data=read_csv("F:\\data\\challenge.csv", guess_max = 1000)
```

```
## Rows: 2000 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): x
## date (1): y
##
## 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.
```

```
problems(data)
```

```
## # A tibble: 0 x 5
## # i 5 variables: row <int>, col <int>, expected <chr>, actual <chr>, file <chr>
```

```
#요즘은 잘 되는듯
#기본적으로 problems와 read_csv인수
#col_type= cols(col_*()~~) 을 이용해서 열의 자료형을 잘 바꿔주는것이다.

#파일저장
write_csv(data, "test.csv")
```

tidy한 데이터

열은 변수 행은 한 관측 각 값은 자신의 셀이 있어야함

```
table1 #tidy 데이터
```

```
## # A tibble: 6 x 4
##   country      year  cases population
```

```
##   <chr>       <dbl>  <dbl>      <dbl>
## 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
```

```
table2 #변수명이 셀에있어서 not tidy
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <dbl> <chr>           <dbl>
##  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
```

```
table3 #rate열이 tidy하지 않음
```

```
## # A tibble: 6 x 3
##   country      year rate
##   <chr>       <dbl> <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
```

```
table4a #한 행에 두 관측이 들어있음, 열에 값이 있음
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
```

```
##   <chr>        <dbl>  <dbl>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

```
table4b # 위와 동일
```

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

```
# table2와 table4a, b에서 rate를 계산하자

cases=table2 %>% filter(type=="cases") %>% .$count
population=table2 %>% filter(type=="population") %>% .$count
table2 %>% mutate(rate=rep(cases/population * 10000, each=2))
```

```
## # A tibble: 12 x 5
##    country      year type            count  rate
##    <chr>       <dbl> <chr>           <dbl> <dbl>
##  1 Afghanistan  1999 cases             745 0.373
##  2 Afghanistan  1999 population   19987071 0.373
##  3 Afghanistan  2000 cases            2666 1.29
##  4 Afghanistan  2000 population   20595360 1.29
##  5 Brazil       1999 cases           37737 2.19
##  6 Brazil       1999 population  172006362 2.19
##  7 Brazil       2000 cases           80488 4.61
##  8 Brazil       2000 population  174504898 4.61
##  9 China        1999 cases          212258 1.67
## 10 China        1999 population 1272915272 1.67
## 11 China        2000 cases          213766 1.67
## 12 China        2000 population 1280428583 1.67
```

```
#꽤 어렵다
```

데이터를 tidy하게 만들기 위해 Longer를 배워보자 (gather) longer는 열에 값이 있을 때 쓴다.

```
table4a
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
```

```
##   <chr>        <dbl>  <dbl>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

```
# 얘는 열에 값이 있다. 그러면 열을 값으로 만들어 새로운 열로 저장해줘야 한다
# 각 값들도 열들로 만들어 줘야 한다
table4a %>%
    pivot_longer(c(`1999`, `2000`), names_to = "year", values_to = "cases")
```

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

```
#사용법: 1.정리할 열 2.열 이름을 어디로? 3.값들은 어디로?

# 1999 2000은 열이름일 수 없어서 ``로 묶어주기...
#근데 year열을 봐라 int가 아니라 chr이다....
#파싱해야겠지?
tidy4a= table4a %>%
    pivot_longer(c(`1999`, `2000`), names_to = "year", values_to = "cases") %>%
    mutate(year=parse_integer(year))
#깔쌈하죠잉, 파이프 연산자가 짱이야

tidy4b= table4b %>%
    pivot_longer(c(`1999`, `2000`), names_to = "year", values_to = "population") %>%
    mutate(year=parse_integer(year))
#table4b도 끝!
#이제 두 티블을 합쳐볼까?

left_join(tidy4a, tidy4b) #table1과 같다 tidy!
```

```
## Joining with `by = join_by(country, year)`
```

```
## # A tibble: 6 x 4
##   country      year  cases population
```

```
##   <chr>       <int>  <dbl>      <dbl>
## 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
```

다음은 wider (spread) 이다. wider는 한 관측이 여러행에 퍼져있을 때 사용한다.

```
table2 #얘가 한 관측이 여러행으로 퍼져있고, 셀에 변수가 있다.
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <dbl> <chr>           <dbl>
##  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
```

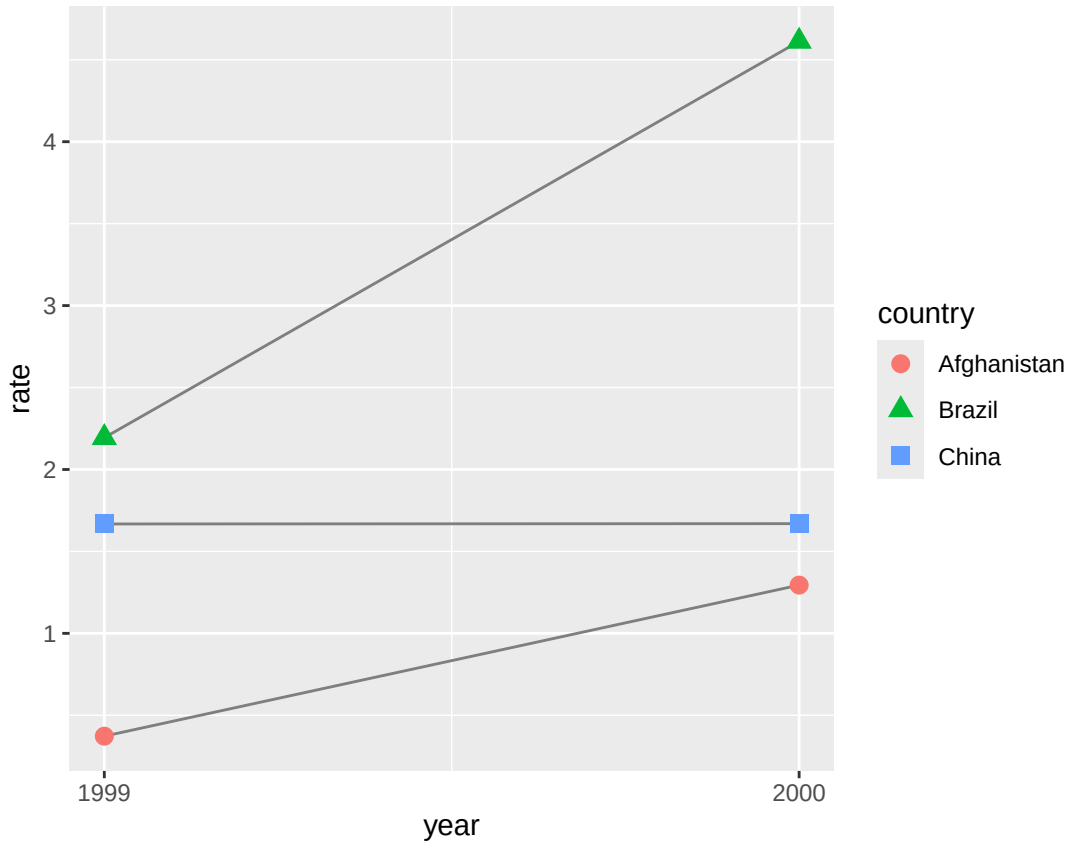
```
#임마는 cases와 population을 열로 보내야한다, 값은 count

table2 %>% pivot_wider(names_from = type, values_from = count)
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <dbl>  <dbl>      <dbl>
## 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
```

```r
#원래 셀에 있던 놈들이 변수이름이라 열이름 지을필요 없고
#처음에는 열이름이 오는 열, 두번째는 값이 오는 열

#longer는 아래로 길어지고 wider는 옆으로 넓어지는 느낌...

table2 %>%
    pivot_wider(names_from = type, values_from = count) %>%
    mutate(rate=cases/population * 10000)
```

```
## # A tibble: 6 x 5
##   country      year  cases population  rate
##   <chr>       <dbl>  <dbl>      <dbl> <dbl>
## 1 Afghanistan  1999    745   19987071 0.373
## 2 Afghanistan  2000   2666   20595360 1.29
## 3 Brazil       1999  37737  172006362 2.19
## 4 Brazil       2000  80488  174504898 4.61
## 5 China        1999 212258 1272915272 1.67
## 6 China        2000 213766 1280428583 1.67
```

```r
#아까 했던 작업이 진짜 간단해진다

table2 %>%
    pivot_wider(names_from = type, values_from = count) %>%
    mutate(rate=cases/population * 10000) %>%
    ggplot(aes(year, rate)) +
    geom_line(aes(group=country), color="gray50") +
    geom_point(aes(colour = country, shape = country), size=3) +
    scale_x_continuous(breaks=c(1999,2000)) + #이산적으로 만들기
    theme(aspect.ratio = 1/1)
```




```
# 더 나아가 case population rate를 연도별로 만들어 6개의 열로 넓혀보자

table2 %>%
    pivot_wider(names_from = type, values_from = count) %>%
    mutate(rate=cases/population * 10000) %>%
    pivot_wider(names_from=year, values_from = c(cases, population, rate))
```

```
## # A tibble: 3 x 7
##   country     cases_1999 cases_2000 population_1999 population_2000 rate_1999
##   <chr>            <dbl>      <dbl>           <dbl>           <dbl>     <dbl>
## 1 Afghanistan        745       2666        19987071        20595360     0.373
## 2 Brazil           37737      80488       172006362       174504898     2.19
## 3 China           212258     213766      1272915272      1280428583     1.67
## # i 1 more variable: rate_2000 <dbl>
```

```
#생각보다 쉽다 values_from을 벡터로 주면 알아서 잘해줌...
#이제 열들을 정리하자
table2 %>%
    pivot_wider(names_from = type, values_from = count) %>%
    mutate(rate=cases/population * 10000) %>%
```

```
    pivot_wider(names_from=year, values_from = c(cases, population, rate)) %>%
    relocate(country, contains("1999"))
```

```
## # A tibble: 3 x 7
##   country     cases_1999 population_1999 rate_1999 cases_2000 population_2000
##   <chr>            <dbl>           <dbl>     <dbl>      <dbl>           <dbl>
## 1 Afghanistan        745        19987071     0.373       2666        20595360
## 2 Brazil           37737       172006362     2.19       80488       174504898
## 3 China           212258      1272915272     1.67      213766      1280428583
## # i 1 more variable: rate_2000 <dbl>
```

```
#연습 문제
preg <- tribble(
  ~pregnant, ~male, ~female,
  "yes",     NA,    10,
  "no",      20,    12
)

preg %>%
    select(-(male)) %>%
    pivot_wider(names_from=pregnant, values_from=female)
```

```
## # A tibble: 1 x 2
##     yes    no
##   <dbl> <dbl>
## 1    10    12
```

이제 table3 도 tidy하게 해보자

```
table3 #rate를 쪼개야 한다.
```

```
## # A tibble: 6 x 3
##   country      year rate
##   <chr>       <dbl> <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
```

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

```
## # A tibble: 6 x 4
##   country      year cases  population
##   <chr>       <dbl> <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
```

```r
#쉽죠잉, 근데 자료형에 예민해야한다

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

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <dbl>  <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
```

```r
# 파싱할거없이 함수에 내장되어있음

table5=table3 %>% separate(year, into=c("century","year"),sep=2)
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
```

```r
#위치 고정으로 쪼개기 가능
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
```