2부 Tidyverse 활용하기

Tidyverse는 data science를 위해 개발된 패키지들의 묶음이라고 할 수 있다. 여기에 속한 패키지들은 모두 공통된 분석 방식을 공유하고 있으며, 자료 분석에 필요한 모든 과정을 망라하고 있다. 자료 분석 내용을 과정별로 나누어 보면, 먼저 분석 대상이 되는 외부 자료를 R로 불러와야 한다. 불러온 자료가 항상 분석 가능한 형태라는 보장이 없기 때문에, 분석이 가능한 혹은 분석하기 편리한 형태로 자료를 다듬어야 한다. 이어서 분석에 필요한 자료를 생성하는 등의 변형과정을 거쳐서 자료의 시각화와 모형화 과정을 통해 자료를 분석하게 된다.

패키지 `tidyverse`에 속한 많은 패키지들은 개별적으로 하나하나 설치할 필요 없이 `install.packages("tidyverse")`를 실행하면 모든 패키지들이 설치된다. 하지만 이 패키지들을 사용하기 위하여 `library(tidyverse)`를 실행하면 설치 때와는 다르게 몇몇 패키지들만 R 세션에 올라오는데, 이 몇몇 패키지들을 core tidyverse라고 부른다. 여기에 속하지 않는 다른 많은 패키지들은 모두 개별적으로 함수 `library()`로 불러와야 한다. Tidyverse 버전 1.2.1의 상황에서 core tidyverse에 속하는 패키지는 다음과 같다.

- `readr`: 자료 불러오기
- `tibble`: 개선된 형태의 데이터 프레임
- `tidyr`: 분석이 편리한 형태인 tidy 자료 생성
- `dplyr`: 데이터 프레임 다루기
- `stringr`: 문자열 다루기
- `forcats`: 요인 다루기
- `ggplot2`: 자료의 시각화
- `purr`: 함수형 프로그래밍

2부에서는 core tidyverse에 속한 패키지들의 사용법을 살펴보고자 한다.

6. Tibble: 개선된 형태의 데이터 프레임

6.1 Tibble의 생성

Core tidyverse에 속한 패키지 tibble을 사용하기 위해 library(tidyverse)를 실행하면, 다음과 같이 몇몇 패키지들이 세션에 함께 올라오는 것을 볼 수 있다.

```
> library(tidyverse)
-- Attaching packages --------------------------------- tidyverse 1.2.1 --
√ ggplot2 3.0.0     √ purrr   0.2.5
√ tibble  1.4.2     √ dplyr   0.7.6
√ tidyr   0.8.1     √ stringr 1.3.1
√ readr   1.1.1     √ forcats 0.3.0
-- Conflicts ------------------------------------tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

패키지 ggplot2에 있는 데이터 프레임 mpg는 1999년부터 2008년까지 매년 새롭게 출시된 38종류의 인기차종 234대를 대상으로 조사된 연비를 비롯한 11개 변수에 대한 자료이다. 데이터 프레임 mpg의 내용을 확인해 보자.

```
> mpg
# A tibble: 234 x 11
   manufacturer model   displ  year   cyl  trans   drv     cty   hwy    fl
   <chr>        <chr>   <dbl> <int> <int> <chr>   <chr> <int> <int> <chr>
 1 audi         a4        1.80  1999     4  auto(l~  f        18    29  p
 2 audi         a4        1.80  1999     4  manual~  f        21    29  p
 3 audi         a4        2.00  2008     4  manual~  f        20    31  p
 4 audi         a4        2.00  2008     4  auto(a~  f        21    30  p
 5 audi         a4        2.80  1999     6  auto(l~  f        16    26  p
 6 audi         a4        2.80  1999     6  manual~  f        18    26  p
 7 audi         a4        3.10  2008     6  auto(a~  f        18    27  p
 8 audi         a4 qua~   1.80  1999     4  manual~  4        18    26  p
 9 audi         a4 qua~   1.80  1999     4  auto(l~  4        16    25  p
10 audi         a4 qua~   2.00  2008     4  manual~  4        20    28  p
# ... with 224 more rows, and 1 more variable: class <chr>
```

1부에서 살펴봤던 전통적인 데이터 프레임의 출력 형태와 많이 다른 모습을 볼 수 있는데, 이것이 바로 tibble의 출력 형태이다. Tibble은 tidyverse에 속한 패키지들이 공통적으로 사용하는 개선된 형태의 데이터 프레임이다. 기존의 전통적인 데이터 프레임과의 차이점 비교는 6.2절에서 살펴볼 수 있다.

기존의 전통적인 데이터 프레임을 tibble로 전환하기 위해서는 함수 as_tibble()을 사용하면 된다.

```
> as_tibble(cars)
# A tibble: 50 x 2
   speed  dist
   <dbl> <dbl>
 1  4.00  2.00
 2  4.00 10.0
 3  7.00  4.00
 4  7.00 22.0
 5  8.00 16.0
 6  9.00 10.0
 7 10.0  18.0
 8 10.0  26.0
 9 10.0  34.0
10 11.0  17.0
# ... with 40 more rows
```

개별 벡터를 이용한 tibble의 생성은 함수 tibble()로 할 수 있다. 길이가 1인 스칼라는 순환법칙이 적용되며, 함께 입력된 변수를 이용하여 다른 변수를 만들 수 있다.

```
> tibble(x=1:3,y=x+1,z=1)
# A tibble: 3 x 3
      x     y     z
  <int> <dbl> <dbl>
1     1  2.00  1.00
2     2  3.00  1.00
3     3  4.00  1.00
```

함께 입력되는 변수를 이용하여 다른 변수를 만들 수 있는 기능은 매우 유용하게 사용될 수 있는 것인데, 함수 data.frame()에서는 가능하지 않던 것이다.

```
> data.frame(x=1:3,y=x+1)
Error in data.frame(x = 1:3, y = x + 1) : object 'x' not found
```

또한 문자형 벡터를 요인으로 자동 전환시켰던 함수 data.frame()과는 다르게 tibble()은 입력 자료의 유형을 바꾸지 않는다.

```
> tibble(x=1:3,y=letters[1:3])
# A tibble: 3 x 2
      x   y
  <int> <chr>
1     1   a
2     2   b
3     3   c

> str(data.frame(x=1:3,y=letters[1:3]))
'data.frame': 3 obs. of  2 variables:
 $ x: int  1 2 3
 $ y: Factor w/ 3 levels "a","b","c": 1 2 3
```

함수 tibble()에서는 열(변수) 단위로 자료를 입력해야 했는데, 경우에 따라서 행 단위로 입력하는 것이 더 편리할 때가 있다. 이런 경우에는 함수 tribble()을 사용해야 한다. 첫 줄의 변수 이름은 기호 `~`으로 시작되는 공식으로 정의되며, 각 자료는 콤마로 구분된다.

```
> tribble(
    ~x, ~y,
     1, "a",
     2, "b",
     3, "c"
  )
# A tibble: 3 x 2
      x   y
  <dbl> <chr>
1  1.00  a
2  2.00  b
3  3.00  c
```

6.2 Tibble과 전통적 데이터 프레임의 비교

Tibble과 전통적인 데이터 프레임은 출력 방식과 자료의 부분을 선택하는 인덱싱 방식에서 차이가 있으며 row names를 다루는 방식에서도 차이가 있다.

- 출력 방식의 차이

먼저 출력 방식에서 보면 전통적인 데이터 프레임은 대규모 데이터를 대상으로 하는 경우 조금 불편한 측면이 있다. 즉, 전통적 데이터 프레임의 이름을 실행시키면 가능한 모든 자료를 화면에 출력하게 되어 있는데, 이런 방식은 대규모 자료의 경우 내용을 확인하기 매우 어려운 측면이 있다. 이에 반하여 tibble은 처음 10개 케이스만을 출력하며, 변수도 화면의 크기에 따라 출력되는 개수를 조절하여, 한 화면에서 자료의 특성을 비교적 편하게 확인할 수 있게 한다.

```
> as_tibble(MASS::Cars93)
# A tibble: 93 x 27
   Manufacturer Model     Type    Min.Price  Price  Max.Price  MPG.city
 * <fct>        <fct>     <fct>       <dbl>  <dbl>      <dbl>     <int>
 1 Acura        Integra   Small        12.9   15.9       18.8        25
 2 Acura        Legend    Midsize      29.2   33.9       38.7        18
 3 Audi         90        Compact      25.9   29.1       32.3        20
 4 Audi         100       Midsize      30.8   37.7       44.6        19
 5 BMW          535i      Midsize      23.7   30.0       36.2        22
 6 Buick        Century   Midsize      14.2   15.7       17.3        22
 7 Buick        LeSabre   Large        19.9   20.8       21.7        19
 8 Buick        Roadmaster Large       22.6   23.7       24.9        16
 9 Buick        Riviera   Midsize      26.3   26.3       26.3        19
10 Cadillac     DeVille   Large        33.0   34.7       36.3        16
# ... with 83 more rows, and 20 more variables: MPG.highway <int>,
#   AirBags <fct>, DriveTrain <fct>, Cylinders <fct>, EngineSize <dbl>,
#   Horsepower <int>, RPM <int>, Rev.per.mile <int>,
#   Man.trans.avail <fct>, Fuel.tank.capacity <dbl>, Passengers <int>,
#   Length <int>, Wheelbase <int>, Width <int>, Turn.circle <int>,
#   Rear.seat.room <dbl>, Luggage.room <int>, Weight <int>, Origin <fct>,
```

```
#   Make <fct>
```

변수 이름과 더불어 변수의 유형을 바로 밑 줄에 표시하고 있으며, 화면 크기 때문에 자료가 출력되지 않은 변수는 이름과 유형을 가장 밑에 나열해 두었다. 만일 조금 더 많은 자료를 확인하고자 한다면 함수 print()를 옵션과 함께 사용하면 된다. 예를 들어 처음 20개 케이스와 모든 변수를 다 출력하고자 한다면 다음과 같이 옵션 'n=20'과 'width=Inf'를 사용한다.

```
> print(as_tibble(MASS::Cars93),n=20,width=Inf)
```

- Row names 처리 방식의 차이

자료가 출력될 때 전통적인 데이터 프레임의 경우에는 row name이 함께 출력되지만, tibble에서는 생략이 된다. 데이터 프레임 mtcars를 출력해 보자.

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

> mtcars_t <- as_tibble(mtcars)
> print(mtcars_t,n=6)
# A tibble: 32 x 11
    mpg   cyl   disp    hp    drat    wt  qsec    vs    am    gear  carb
  * <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21.0  6.00   160    110    3.90  2.62  16.5  0     1.00  4.00  4.00
2  21.0  6.00   160    110    3.90  2.88  17.0  0     1.00  4.00  4.00
3  22.8  4.00   108    93.0   3.85  2.32  18.6  1.00  1.00  4.00  1.00
4  21.4  6.00   258    110    3.08  3.22  19.4  1.00  0     3.00  1.00
5  18.7  8.00   360    175    3.15  3.44  17.0  0     0     3.00  2.00
6  18.1  6.00   225    105    2.76  3.46  20.2  1.00  0     3.00  1.00
# ... with 26 more rows
```

생략된 row name은 함수 rownames_to_column()을 사용하면 변수로 전환할 수 있다.

```
> mtcars_t <- rownames_to_column(mtcars_t,var="rowname")
> mtcars_t
# A tibble: 32 x 12
   rowname     mpg   cyl   disp    hp   drat    wt   qsec    vs    am    gear
   <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Mazda RX4   21.0  6.00   160   110    3.90  2.62  16.5   0     1.00  4.00
 2 Mazda RX4~  21.0  6.00   160   110    3.90  2.88  17.0   0     1.00  4.00
 3 Datsun 710  22.8  4.00   108   93.0   3.85  2.32  18.6  1.00  1.00  4.00
 4 Hornet 4 ~  21.4  6.00   258   110    3.08  3.22  19.4  1.00  0     3.00
 5 Hornet Sp~  18.7  8.00   360   175    3.15  3.44  17.0  0     0     3.00
```

```
 6 Valiant      18.1  6.00   225   105    2.76  3.46  20.2  1.00  0     3.00
 7 Duster 360   14.3  8.00   360   245    3.21  3.57  15.8   0    0     3.00
 8 Merc 240D    24.4  4.00   147   62.0   3.69  3.19  20.0   1.00  0     4.00
 9 Merc 230     22.8  4.00   141   95.0   3.92  3.15  22.9   1.00  0     4.00
10 Merc 280     19.2  6.00   168  123     3.92  3.44  18.3   1.00  0     4.00
# ... with 22 more rows, and 1 more variable: carb <db1>
```

- 인덱싱 방식의 차이

데이터 프레임의 일부분을 선택하는 인덱싱 기법은 tibble에서도 동일하게 적용된다. 한 가지 차이점은 기호 '$'을 이용하는 경우인데, 데이터 프레임에서는 변수 이름이 부분 매칭만 되어도 선택되지만, tibble에서는 엄격한 기준이 적용되어 변수의 전체 이름을 다 사용해야 선택된다.

```
> df1 <- data.frame(xyz=1:3,abc=letters[1:3])
> df1$x
[1] 1 2 3

> tb1 <- as_tibble(df1)
> tb1$x
NULL
Warning message:
Unknown or uninitialised column: 'x'.
> tb1$xyz
[1] 1 2 3
```

흔한 상황은 아니지만, 만일 tibble을 입력했을 때 처리를 못하는 함수가 있다면, tibble을 전통적인 데이터 프레임으로 전환해야 한다. 데이터 프레임으로의 전환은 함수 `as.data.frame()`을 사용하면 된다. 이러한 전환이 필요한 상황 중 하나는 대괄호를 이용하여 행렬 방식으로 인덱싱을 실시하는 경우이다. 전통적 데이터 프레임의 경우 행렬 방식으로 자료를 부분 선택할 때 선택된 변수가 두 개 이상이면 데이터 프레임이 유지되지만, 선택된 변수가 한 개라면 벡터가 된다. 즉, `mtcars[,1:2]`는 데이터 프레임이지만 `mtcars[,1]`은 벡터가 된다는 것이다. 반면에 tibble의 경우에는 선택되는 변수의 개수와 관계 없이 그대로 tibble 속성이 유지가 된다. 따라서 `mtcars_t[,1:2]`와 `mtcars_t[,1]` 모두 tibble이며, `mtcars_t[1,1]`도 tibble 속성을 그대로 유지하고 있다. 따라서 데이터 프레임에서 행렬 방식으로 하나의 변수를 선택했을 때 벡터가 되어야 작동이 되는 경우라면 tibble을 사용해서는 안 되는 상황이 되는 것이다.

6.3 6장을 마치며

Tidyverse로 인하여 R은 완전히 새로 태어났다고 할 수 있다. 사용 방식에 일관성이 확립되었고 어렵지 않게 확장 할 수 있는 체계가 구성되었다. 따라서 데이터를 다듬고 그래프를 작성하는 작업은 이제는 base 패키지를 이용하는 것보다 tidyverse의 일원인 `dplyr`이나 `ggplot2`를 이용하

는 것이 훨씬 더 바람직하다고 하겠다. 다음 장에서는 패키지 dplyr에 의한 데이터 다듬기에 대하여 살펴보겠다.