dplyr-data_preprocess

Kwon Namtaek

2021년 1월 18일

1 dplyr이란?

1.1 dplyr 소개

dplyr이란 r에서 배포된 데이터 전처리 패키지의 이름이다. 매우 유명한 패키지이며, R studio의 적극적인 지원을 받는 패키지이다. 여러 장점들이 있으며, 속도도 r 기존 함수들보다 훨씬 빠르다. 무엇보다 매우 직관적이고, 플로우를 갖는 전처리를 가능하게 한다. dplyr 패키지의 기능들에 대해 소개하기 전에 일단 이런 dplyr 계열 패키지들에 대해 알아보자.

```
#install.packages(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
```

1.2 tidyverse

R은 tidyverse와 함께 성장해왔다. tidyverse도 패키지 이름인데, 패키지들에 대한 패키지이다. 시각화 패키지 ggplot2, 데이터전처리 패키지 dplyr, 문자열 전처리 패키지 stringr, 데이터프레임 패키지 tibble, 데이터로딩 패키지 readr, 범주형데이터 처리 패키지 forcats 가 기본으로 딸려오고, 기본으로 딸려오지 않는 날짜 데이터 처리 패키지 lubridate 등등… 정말 많은 패키지들이 존재한다. 이런 다양한 패키지를 불러와서 %>% 파이프 연산자를 통해 하나의 과정으로 문제를 다룰 수 있다는 것이 tidyverse 패키지가 가지는 매우 큰 장점이고, R를 사랑하는 이유다. 파이썬을 공부해야하는데 R에 한번 빠지면 이 편리함을 벗어날 수 없다…

이 tidyverse의 철학은 'tidy data'를 만드는 것에 있다. tidy data라 함은 하나의 철학인데, 듣고나면 너무나도 당연한 것이다.

1) 각 변수는 컬럼에 위치한다.
2) 각 관측치는 행에 위치한다.
3) 하나의 셀에는 하나의 값이 존재한다.

그러나 현실에서 마주하는 데이터들은 이런 tidy data가 아니기 때문에, 이렇게 어떤 데이터가 좋은(tidy) 데이터인지 기준을 잡는 것이 필요했고, 해들리 위컴(Hadley Wickham, tidyverse 프로젝트의 총 책임자?)의 논문을 통해 처음 이런 논의가 시작됐다. 아무튼 우리한테 중요한 결론은, dplyr 패키지를 통해 데이터 전처리를 깔끔하게 할 수 있다는 것!

1.3 데이터불러오기

R의 내장함수에는 `read.csv`로 데이터를 불러올 수 있고, tidyverse쪽 내장함수로는 `read_csv`로 데이터를 불러올 수 있다. 하지만 데이터 불러오기에 한에서는 `data.table` 패키지에 존재하는 `fread` (fast read) 함수를 이용하자. data.table은 tidyverse보다 빠르며, 파이썬을 상회하는 속도의 데이터핸들링를 지원하지만, 문법이 tidyverse에 비해 직관적이지 않다. 따라서 데이터로딩만 `fread`를 사용하자.

```
# install.packages(data.table)
presidential = data.table::fread('presidential.csv', data.table = F)
```

1.4 파이프연산자

R에서 기본적인 함수 연산은 다음과 같이 `()`를 통해 이루어진다.

```
summary(head(presidential))
```

```
##      name               start                 end
##  Length:6           Min.   :1953-01-20   Min.   :1961-01-20
##  Class :character   1st Qu.:1961-10-06   1st Qu.:1965-03-07
##  Mode  :character   Median :1966-06-22   Median :1971-10-30
##                     Mean   :1966-06-14   Mean   :1971-02-13
##                     3rd Qu.:1973-03-20   3rd Qu.:1976-06-10
##                     Max.   :1977-01-20   Max.   :1981-01-20
##     party
##  Length:6
##  Class :character
##  Mode  :character
##
##
##
```

위의 함수는 데이터의 상위 6개 행을 뽑고, 각 열 별로 통계량을 요약하라는 명령어다. 마치 수학에서 $f(g(h(x)))$를 하는 것과 같은 입력방식이다. 이는 우리가 보기에 불편하다. 그래서 tidyverse는 %>%라는 파이프 연산자를 도입한다. 파이프 연산자는 `'ctrl + shift + m'`을 통해 단축키로 입력이 가능하다. 파이프 연산자를 활용하면,

```
presidential %>% head %>% summary
```

```
##      name               start                 end
##  Length:6           Min.   :1953-01-20   Min.   :1961-01-20
##  Class :character   1st Qu.:1961-10-06   1st Qu.:1965-03-07
##  Mode  :character   Median :1966-06-22   Median :1971-10-30
##                     Mean   :1966-06-14   Mean   :1971-02-13
##                     3rd Qu.:1973-03-20   3rd Qu.:1976-06-10
##                     Max.   :1977-01-20   Max.   :1981-01-20
##     party
```

```
##  Length:6
##  Class :character
##  Mode  :character
##
##
##
```

위와 같이 조금더 우리의 의도를 직관적으로 이해할 수 있다. 왼쪽에서부터 쭉 읽으면 된다. 개꿀!

더불어서 데이터를 확인할 때, `glimpse` 함수는 매우 유용하다.

```
presidential %>% glimpse
```

```
## Rows: 11
## Columns: 4
## $ name  <chr> "Eisenhower", "Kennedy", "Johnson", "Nixon", "Ford", "Carter"...
## $ start <date> 1953-01-20, 1961-01-20, 1963-11-22, 1969-01-20, 1974-08-09, ...
## $ end   <date> 1961-01-20, 1963-11-22, 1969-01-20, 1974-08-09, 1977-01-20, ...
## $ party <chr> "Republican", "Democratic", "Democratic", "Republican", "Repu...
```

2 함수 소개

우리가 다룰 함수는 다음과 같다.

- filter() : 조건에 맞는 row만 필터링한다.
- arrange() : 입력한 변수의 오름차순으로 데이터프레임의 행을 정렬한다.
- select() : 입력된 행만을 선택한다. 나머지는 제외한다.
- mutate() : 데이터프레임에 새 변수를 추가한다.
- group_by() : 입력한 변수의 범주별로 데이터의 행을 그룹화한다.
- summarise() : 데이터프레임의 요약 변수를 생성해 tibble(데이터프레임) 형태로 출력한다. group_by와 같이 많이 사용한다.

다른 함수들 `left_join`, `gather`, `spread` 등등 유용한 함수들이 많지만, 위의 것들만 다루자.

2.1 filter

filter는 조건을 걸고, 조건에 해당하는 행만 출력한다.

```
filter(presidential, party == 'Republican')
```

```
##         name      start        end      party
## 1 Eisenhower 1953-01-20 1961-01-20 Republican
## 2      Nixon 1969-01-20 1974-08-09 Republican
## 3       Ford 1974-08-09 1977-01-20 Republican
## 4     Reagan 1981-01-20 1989-01-20 Republican
## 5       Bush 1989-01-20 1993-01-20 Republican
## 6       Bush 2001-01-20 2009-01-20 Republican
```

```
presidential %>% filter(party == 'Republican')
```

```
##         name      start        end      party
## 1 Eisenhower 1953-01-20 1961-01-20 Republican
## 2      Nixon 1969-01-20 1974-08-09 Republican
## 3       Ford 1974-08-09 1977-01-20 Republican
## 4     Reagan 1981-01-20 1989-01-20 Republican
## 5       Bush 1989-01-20 1993-01-20 Republican
## 6       Bush 2001-01-20 2009-01-20 Republican
```

다음과 같이 파이프 연산자로 더 편한 함수를 만들어 낼 수 있다. 더 다양한 조건을 걸어보자.

```
presidential %>% filter(party == 'Republican' & start > 1973)
```

```
##     name      start        end      party
## 1 Reagan 1981-01-20 1989-01-20 Republican
## 2   Bush 1989-01-20 1993-01-20 Republican
## 3   Bush 2001-01-20 2009-01-20 Republican
```

2.2 select

select는 원하는 열을 선택한다. filter 뒤에 걸어보자.

```
presidential %>%
  filter(party == 'Republican' & start > 1973) %>%
  select(name, party)
```

```
##     name      party
## 1 Reagan Republican
## 2   Bush Republican
## 3   Bush Republican
```

2.3 mutate

mutate는 새로운 변수를 만들어낸다. 집권일 'term_length'변수를 임기 종료일에서 임기 시작일을 빼서 만들어내자.

```
mypresidents = presidential %>%
  mutate(term_length = end - start)
```

2.4 arrange

만들어놓은 'mypresident' 데이터프레임에 대해 집권일이 높은 순으로 정렬해보자.

```
mypresidents %>% arrange(term_length)
```

```
##          name      start        end      party term_length
## 1        Ford 1974-08-09 1977-01-20 Republican         895
## 2     Kennedy 1961-01-20 1963-11-22 Democratic        1036
## 3      Carter 1977-01-20 1981-01-20 Democratic        1461
## 4        Bush 1989-01-20 1993-01-20 Republican        1461
## 5     Johnson 1963-11-22 1969-01-20 Democratic        1886
## 6       Nixon 1969-01-20 1974-08-09 Republican        2027
## 7  Eisenhower 1953-01-20 1961-01-20 Republican        2922
## 8      Reagan 1981-01-20 1989-01-20 Republican        2922
## 9     Clinton 1993-01-20 2001-01-20 Democratic        2922
## 10       Bush 2001-01-20 2009-01-20 Republican        2922
## 11      Obama 2009-01-20 2017-01-20 Democratic        2922
```

```
mypresidents %>% arrange(desc(term_length))
```

```
##          name      start        end      party term_length
## 1  Eisenhower 1953-01-20 1961-01-20 Republican        2922
## 2      Reagan 1981-01-20 1989-01-20 Republican        2922
## 3     Clinton 1993-01-20 2001-01-20 Democratic        2922
## 4        Bush 2001-01-20 2009-01-20 Republican        2922
## 5       Obama 2009-01-20 2017-01-20 Democratic        2922
## 6       Nixon 1969-01-20 1974-08-09 Republican        2027
## 7     Johnson 1963-11-22 1969-01-20 Democratic        1886
## 8      Carter 1977-01-20 1981-01-20 Democratic        1461
## 9        Bush 1989-01-20 1993-01-20 Republican        1461
## 10    Kennedy 1961-01-20 1963-11-22 Democratic        1036
## 11       Ford 1974-08-09 1977-01-20 Republican         895
```

추가적으로 정렬의 기준을 정할 수도 있다.

```
mypresidents %>% arrange(desc(term_length), party)
```

```
##          name      start        end      party term_length
## 1     Clinton 1993-01-20 2001-01-20 Democratic        2922
## 2       Obama 2009-01-20 2017-01-20 Democratic        2922
## 3  Eisenhower 1953-01-20 1961-01-20 Republican        2922
## 4      Reagan 1981-01-20 1989-01-20 Republican        2922
## 5        Bush 2001-01-20 2009-01-20 Republican        2922
## 6       Nixon 1969-01-20 1974-08-09 Republican        2027
## 7     Johnson 1963-11-22 1969-01-20 Democratic        1886
## 8      Carter 1977-01-20 1981-01-20 Democratic        1461
## 9        Bush 1989-01-20 1993-01-20 Republican        1461
## 10    Kennedy 1961-01-20 1963-11-22 Democratic        1036
## 11       Ford 1974-08-09 1977-01-20 Republican         895
```

3 group_by & summarise

summarise 함수만 사용하게 되면 전체 데이터에 대한 요약이 가능하다.

```
mypresidents %>%
  summarize(N = n(),
            first_year = min(start),
            last_year = max(end),
            num_dems = sum(party == "Democratic"),
            days = sum(term_length) / 365.25,
            avg_term_length = mean(term_length))
```

```
##    N first_year  last_year num_dems days avg_term_length
## 1 11 1953-01-20 2017-01-20        5   64        2125.091
```

하지만 group_by 함수와 함께 사용하면 더 많은 정보를 줄 수 있다.

```
mypresidents %>%
  group_by(party) %>%
  summarize(N = n(),
            avg_term_length = mean(term_length))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 3
##   party          N avg_term_length
##   <chr>      <int>           <dbl>
## 1 Democratic     5           2045.
## 2 Republican     6           2192.
```