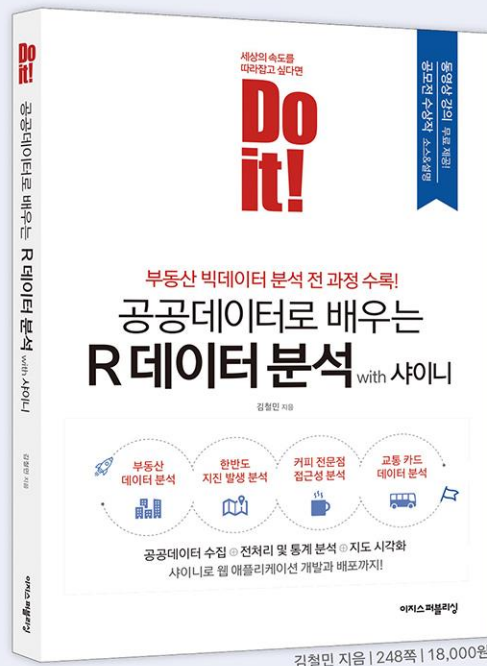


R로 공공데이터를 분석하는 전 과정 실습!
공모전 수상작으로 배우는 R 데이터 분석



김철민 지음 | 248쪽 | 18,000원

공모전
수상작
소스&설명

동영상
강의
무료 제공

04

전처리: 데이터를 알맞게 다듬기

04-1 불필요한 정보 지우기

04-2 항목별 데이터 다듬기

04-3 전처리 데이터 저장하기

데이터를 분석할 때 수집한 자료를 별도의 수정이나 보완 없이 그대로 사용하는 경우는 거의 없습니다. 중복되거나 불필요한 정보는 지우고 형태를 바꾸거나 잘못된 내용을 치환하는 등 전처리 과정을 거쳐야 합니다. 이 장에서는 앞에서 수집한 자료를 분석하기 좋게 다듬는 전처리 과정을 알아보겠습니다.

04-1 불필요한 정보 지우기

1단계 수집한 데이터 불러오기

중요도가 떨어지는 경고^{warning} 메시지를 무시

Do it! 아파트 실거래 자료 불러오기

04_전처리.R

```
08: setwd(dirname(rstudioapi::getSourceEditorContext())$path))
09: options(warn=-1)
10:
11: load("./03_integrated/03_apt_price.rdata") # 실거래 자료 불러오기
12: head(apt_price, 2) # 자료 확인
```

 실행 결과

	X	year	month	day	price	code	dong_nm	jibun	con_year	apt_nm	area	floor
1	1	2021	1	5	31,000	11680	역삼동	720-25	2002	대우디오빌	30.03	4
2	2	2021	1	6	61,000	11680	역삼동	766-8	2002	트레벨	33.48	3

04-1 불필요한 정보 지우기

2단계 결측값과 공백 제거하기

NA^{not available}

Do it! 결측값 확인

04_전처리.R

```
16: table(is.na(apt_price)) # 결측값 확인
```

 실행 결과

```
FALSE TRUE
```

```
520056 12
```

table() 함수를 함께 사용하면 NA가 몇 개 포함되었는지 알 수 있습니다

Do it! 결측값 제거 후 확인

04_전처리.R

```
17: apt_price <- na.omit(apt_price) # 결측값 제거
```

```
18: table(is.na(apt_price)) # 결측값 확인
```

 실행 결과

```
FALSE
```

```
519924
```

04-1 불필요한 정보 지우기

2단계 결측값과 공백 제거하기

Do it! 공백 확인

04_전처리.R

```
20: head(apt_price$price, 2) # 매매가 확인
```

실행 결과

```
[1] "    31,000" "    61,000"
```

공백^{white space}

Do it! 공백 제거와 확인

04_전처리.R

```
22: library(stringr) # install.packages("stringr") # 문자열 처리 패키지 실행
```

```
23: apt_price <- as.data.frame(apply(apt_price, 2, str_trim)) # 공백 제거
```

```
24: head(apt_price$price, 2) # 매매가 확인
```

실행 결과

```
[1] "31,000" "61,000"
```

공백을 제거하려면 stringr 패키지에 들어 있는 str_trim() 함수를 사용

04-2 항목별 데이터 다듬기

현재 아파트 실거래 자료 안에는 문자^{character}와 숫자^{numeric}가 섞여 있습니다. 이처럼 데이터의 형태가 일관되지 않으면 분석할 때 제약이 있습니다. 이번 절에서는 데이터의 속성을 바꿔 분석하기에 알맞게 다듬어 보겠습니다.

표 4-1 항목별 데이터 속성 변경

항목	속성(이전)	속성(이후)
연도(year)	문자형	날짜형
월(month)	문자형	
일(day)	문자형	
가격(price)	문자형	숫자형
지역 코드(code)	문자형	주소(문자형)
법정동(dong_nm)	문자형	
지번(jibun)	문자형	
아파트 이름(apt_nm)	문자형	
건축 연도(con_year)	문자형	숫자형
면적(area)	문자형	숫자형
층수(floor)	문자형	숫자형

04-2 항목별 데이터 다듬기

1단계 매매 연월일 만들기

Do it! 매매 연월일, 연월 데이터 만들기

04_전처리.R

```
32: library(lubridate)
33: library(dplyr)
34: apt_price <- apt_price %>% mutate(ymd=make_date(year, month, day)) # 연월일
35: apt_price$ym <- floor_date(apt_price$ymd, "month") # 연월
36: head(apt_price, 2) # 자료 확인
```

👉 실행 결과

	X	year	month	day	price	...	area	floor	ymd	ym
1	1	2021	1	5	31,000	...	30.0300	4	2021-01-05	2021-01-01
2	2	2021	1	6	61,000	...	33.4800	3	2021-01-06	2021-01-01

1단계 매매 연월일 만들기



알아 두면
좋아요!

파이프라인으로 계산식을 간단하게 만들기

dplyr 패키지가 제공하는 파이프라인 연산자(`%>%`)는 복잡한 계산을 간단히 처리해 줍니다. 예를 들어 $Y=h(g(f(x)))$ 라는 중첩 함수식을 계산하려면 안쪽부터 바깥쪽까지 연이어 결과를 구하고 대입하는 과정을 반복해야 합니다. 이렇게 하면 코드가 복잡하고 길어질 수밖에 없습니다. 그런데 파이프라인(`%>%`)을 이용하면 $Y = x \%>\% f(\dots) \%>\% g(\dots) \%>\% h(\dots)$ 같이 직관적으로 표현할 수 있습니다.

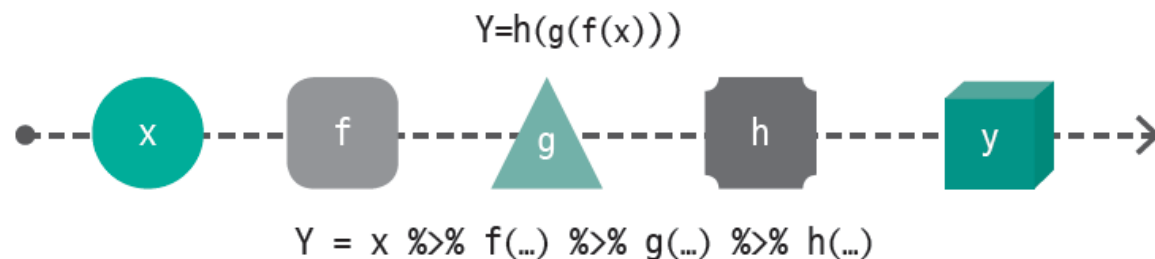


그림 4-2 파이프라인 연산자

04-2 항목별 데이터 다듬기

2단계 매매가 변환하기

Do it! 매매가 확인

04_전처리.R

```
40: head(apt_price$price, 3) # 매매가 확인
```

 실행 결과

```
[1] "31,000" "61,000" "198,000"
```

Do it! 매매가 변환(문자 → 숫자)

04_전처리.R

```
42: apt_price$price <- apt_price$price %>% sub(",", "", .) %>% as.numeric() # 쉼표 제거
```

```
43: head(apt_price$price, 3) # 매매가 확인
```

 실행 결과

```
[1] 31000 61000 198000
```


04-2 항목별 데이터 다듬기

3단계 주소 조합하기

Do it! 아파트 이름 현황

04_전처리.R

```
47: head(apt_price$apt_nm, 30)
```

 실행 결과

```
[1] (... 생략 ...)
```

```
[25] "역삼동하나빌" "역삼푸르지오" "래미안그레이트(진달래2차)" "쌍용플레티넘밸류"
```


```
[29] "이스턴오피스텔" "e-편한세상"
```

Do it! 괄호 이후 삭제

04_전처리.R

```
49: apt_price$apt_nm <- gsub("\\(.+", "", apt_price$apt_nm) # 괄호 이후 삭제
```

```
50: head(apt_price$apt_nm, 30) # 아파트 이름 확인
```

 실행 결과

```
[1] (... 생략 ...)
```

```
[25] "역삼동하나빌" "역삼푸르지오" "래미안그레이트" "쌍용플레티넘밸류"
```

```
[29] "이스턴오피스텔" "e-편한세상"
```

04-2 항목별 데이터 다듬기

3단계 주소 조합하기

Do it! 주소 조합

04_전처리.R

```
52: loc <- read.csv("./01_code/sigun_code/sigun_code.csv") # 지역 코드 불러오기
53: apt_price <- merge(apt_price, loc, by = 'code')          # 지역명 결합하기
54: apt_price$juso_jibun <- paste0(apt_price$addr_2, apt_price$dong, " ",
55:                                apt_price$jibun, " ", apt_price$apt_nm) # 주소 조합
56: head(apt_price, 2)    # 자료 확인
```

🔍 실행 결과

```
code  X year month ... juso_jibun
1 11110 1 2021   1   ... 서울특별시 종로구청운동 56-45 청운현대
2 11110 2 2021   1   ... 서울특별시 종로구사직동 9-1 광화문스페이스본
```

04-2 항목별 데이터 다듬기

4단계 건축 연도, 면적 변환하기

Do it! 건축 연도 현황

04_전처리.R

```
60: head(apt_price$con_year, 3) # 건축 연도 확인
```

👉 실행 결과

```
[1] "2000" "2008" "2004"
```

Do it! 건축 연도 변환(문자 → 숫자)

04_전처리.R

```
62: apt_price$con_year <- apt_price$con_year %>% as.numeric() # 건축 연도 숫자 변환
```

```
63: head(apt_price$con_year, 3) # 건축 연도 확인
```

👉 실행 결과

```
[1] 2000 2008 2004
```

04-2 항목별 데이터 다듬기

5단계 전용 면적 변환하기

Do it! 전용 면적 현황

04_전처리.R

```
67: head(apt_price$area, 3) # 전용 면적 확인
```

🔍 실행 결과

```
[1] "129.7600" "144.5200" "174.5500"
```

Do it! 전용 면적 변환 (문자 → 숫자)

04_전처리.R

```
69: apt_price$area <- apt_price$area %>% as.numeric() %>% round(0) # 전용 면적 변환
```

```
70: head(apt_price$area, 3) # 전용 면적 확인
```

🔍 실행 결과

```
[1] 130 145 175
```

04-2 항목별 데이터 다듬기

5단계 전용 면적 변환하기

Do it! 평당 매매가 만들기

04_전처리.R

```
72: apt_price$py <- round(((apt_price$price/apt_price$area) * 3.3), 0) # 평당 가격
73: head(apt_price$py, 3)      # 평당 매매가 확인
```

 실행 결과

```
[1] 3300 3414 3300
```

04-2 항목별 데이터 다듬기

6단계 층수 변환하기

Do it! 층수 현황

04_전처리.R

```
78: min(apt_price$floor) # 층수 확인
```

실행 결과

```
[1] "-1"
```

year	price	Dong_nm	jibun	floor
2020	19,500	쌍문동	302	10
2020	15,200	미아동	48-12	-4
2019	78,500	후암동	205-44	14
2019	132,000	아현동	288	-1

as.numeric()



abs()

year	price	Dong_nm	jibun	floor
2020	19,500	쌍문동	302	10
2020	15,200	미아동	48-12	4
2019	78,500	후암동	205-44	14
2019	132,000	아현동	288	1

04-2 항목별 데이터 다듬기

6단계 층수 변환하기

Do it! 층수 변환(문자 → 숫자)

04_전처리.R

```
80: apt_price$floor %>% as.numeric() %>% abs() # 층수 변환  
81: min(apt_price$floor) # 층수 확인
```

 실행 결과

```
[1] 1
```

6단계 총수 변환하기

Do it! 카운트 변수 추가

04_전처리.R

```
83: apt_price$cnt <- 1 # 모든 데이터에 숫자 1 할당
84: head(apt_price, 2) # 자료 확인
```

👉 실행 결과

	ymd	ym	year	...	area	floor	py	cnt
1	2021-01-14	2021-01-01	2021	...	130	2	3300	1
2	2021-01-07	2021-01-01	2021	...	145	6	3414	1

04-3 전처리 데이터 저장하기

1단계 필요한 칼럼만 추출하기

Do it! 칼럼 추출

04_전처리.R

```
93: apt_price <- apt_price %>% select(ymd, ym, year, code, addr_1, apt_nm,  
94:                                juso_jibun, price, con_year, area, floor, py, cnt) # 칼럼 추출  
95: head(apt_price, 2) # 자료 확인
```

 실행 결과

	ymd	ym	year	...	area	floor	py	cnt
1	2021-01-14	2021-01-01	2021	...	130	2	3300	1
2	2021-01-07	2021-01-01	2021	...	145	6	3414	1

04-3 전처리 데이터 저장하기

2단계 전처리 데이터 저장하기

Do it! 칼럼 추출 및 저장

04_전처리.R

```
99 : setwd(dirname(rstudioapi::getSourceEditorContext())$path))
100: dir.create("./04_pre_process")    # 새로운 폴더 생성
101: save(apt_price, file = "./04_pre_process/04_pre_process.rdata")    # 저장
102: write.csv(apt_price, "./04_pre_process/04_pre_process.csv")
```



• 날짜 만들기

```
year <- 2011:2020    # 연도 만들기
month <- 2:11        # 월 만들기
day <- 3: 12         # 일 만들기
date <- data.frame(cbind(year, month, day))                # 연, 월, 일 결합하기
date <- date %>% mutate(ymd=make_date(year, month, day))    # 날짜 만들기
head(date, 2)
```



단골 코드 정리하기

• 여러 단계를 거쳐서 다항식 계산하기

```
a <- filter(mtcars, carb > 1)      # 계산 1: 필터링
b <- group_by(a, cyl)              # 계산 2: 그룹화
c <- summarise(b, Avg_mpg = mean(mpg)) # 계산 3: 요약 변수 생성
d <- arrange(c, desc(Avg_mpg))     # 계산 4: 높은 순서 정렬
print(d)
```

• 파이프라인 연산자를 활용하여 다항식 연산을 한번에 계산하기

```
mtcars %>%
  filter(carb > 1) %>%             # 계산 1: 필터링
  group_by(cyl) %>%                #           그룹화
  summarise(Avg_mpg = mean(mpg)) %>% #           요약 변수 생성
  arrange(desc(Avg_mpg))           #           높은 순서 정렬
```



- 특수문자 제거: 공백과 쉼표 제거

```
library(stringr)
tmp <- " 324,135"      # 공백과 쉼표가 포함된 문자열
tmp %>% str_trim() %>% sub(",", "", .) %>% as.numeric()  # 공백, 쉼표 제거
```