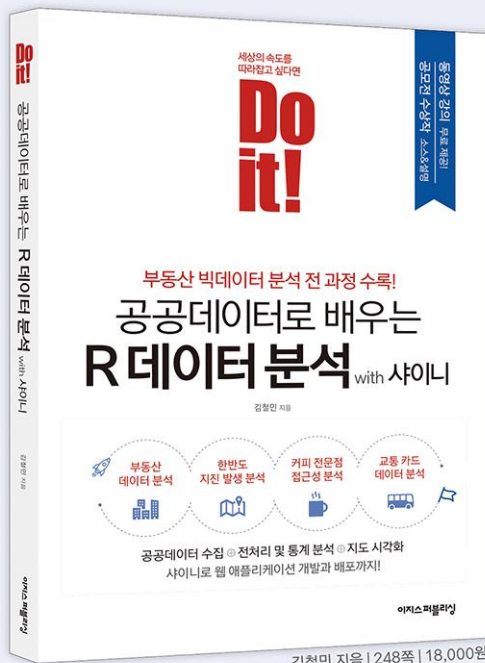


R로 공공데이터를 분석하는 전 과정 실습!
공모전 수상작으로 배우는 R 데이터 분석



공모전
수상작
소스 & 설명

동영상
강의
무료 제공

김철민 지음 | 248쪽 | 18,000원

이 장에서는 공공데이터포털에서 제공하는 API로 자료를 수집할 때 먼저 무엇을 준비해야 하는지 알아봅니다. 또한 분석에 필요한 자료를 요청하고 응답 결과를 자동으로 정리하는 크롤러를 만들어 실제로 자료를 수집해 봅니다.

03

자료 수집: API 크롤러 만들기

- 03-1 크롤링 준비: 무엇을 준비할까?
- 03-2 요청 목록 생성: 자료를 어떻게 요청할까?
- 03-3 크롤러 제작: 자동으로 자료 수집하기
- 03-4 자료 정리: 자료 통합하기

03-1 크롤링 준비: 무엇을 준비할까?

1단계 작업 폴더 설정하기


Do it! 작업 폴더 설정

03_자료수집.R

```
08: install.packages("rstudioapi") # rstudioapi 설치
09: setwd(dirname(rstudioapi::getSourceEditorContext()$path)) # 작업 폴더 설정
10: getwd() # 작업 폴더 확인
```

 실행 결과

```
[1] "C:/Users/user/Documents"
```

 rstudioapi라는 라이브러리를 이용하면 스크립트가 저장된 위치를 작업 폴더로 쉽게 설정할 수 있습니다.

03-1 크롤링 준비: 무엇을 준비할까?

2단계 수집 대상 지역 설정하기

Do it! 수집 대상 지역 설정

03_자료수집.R

```
14: loc <- read.csv("./01_code/sigun_code/sigun_code.csv") # 지역 코드
15: loc$code <- as.character(loc$code) # 행정구역명 문자 변환
16: head(loc, 2) # 확인
```

☞ 실행 결과

	code	sido	sigungu	addr_1	addr_2
1	11110	서울특별시	종로구	서울_종로	서울특별시 종로구
2	11140	서울특별시	중구	서울_중구	서울특별시 중구

03-1 크롤링 준비: 무엇을 준비할까?

2단계 수집 대상 지역 설정하기



지역 코드가 뭔가요?

지역 코드는 기초 자치 단체인 시·군·구에 할당한 코드로서 광역시·도(2자리) + 기초시·군·구(3자리)로 이루어집니다. 예를 들어 서울특별시 종로구의 지역 코드는 11110인데, 이는 서울특별시 11과 종로구 110의 조합입니다.

표 3-1 지역 코드 예시

code	sido	sigungu	addr_1	addr_2
11110	서울특별시	종로구	서울_종로	서울특별시 종로구
11140	서울특별시	중구	서울_중구	서울특별시 중구
11170	서울특별시	용산구	서울_용산	서울특별시 용산구
11200	서울특별시	성동구	서울_성동	서울특별시 성동구

03-1 크롤링 준비: 무엇을 준비할까?

3단계 수집 기간 설정하기

Do it! 수집 기간 설정

03_자료수집.R

```
20: datelist <- seq(from = as.Date('2021-01-01'), # 시작
21:                  to   = as.Date('2021-12-31'), # 종료
22:                  by    = '1 month')             # 단위
23: datelist <- format(datelist, format = '%Y%m')   # 형식 변환(YYYY-MM-DD => YYYYMM)
24: datelist[1:3] # 확인
```

seq(from = A, to = B, by = C)

실행 결과

"202101" "202102" "202103"

03-1 크롤링 준비: 무엇을 준비할까?

4단계 인증키 입력하기

Do it! 인증키 입력

03_자료수집.R

```
28: service_key <- "인증키" # 인증키 입력
```

개발계정 상세보기

일반 인증키 (Encoding)	R5rAjPY3n8V%2FBZ7jg0nF5v61b6vEiXN7k9KAW1ornFa
----------------------	---

03-2 요청 목록 생성: 자료를 어떻게 요청할까?

1단계 요청 목록 만들기

Do it! 요청 목록 만들기

03_자료수집.R

```
37: url_list <- list() # 빈 리스트 만들기  
38: cnt <- 0          # 반복문의 제어 변수 초깃값 설정
```

03-2 요청 목록 생성: 자료를 어떻게 요청할까?

2단계 요청 목록 채우기

요청 목록(url_list)은 ‘프로토콜 + 주소 + 포트 번호 + 리소스 경로 + 요청 내역’ 등 5가지 정보로 구성됩니다. 대부분은 고정된 내용이지만 요청 내역은 대상 지역과 기간이라는 2가지 조건에 따라 변합니다. 이러한 조건을 고려하려면 반복문 안에 또 다른 반복문이 동작하는 중첩이 필요합니다.

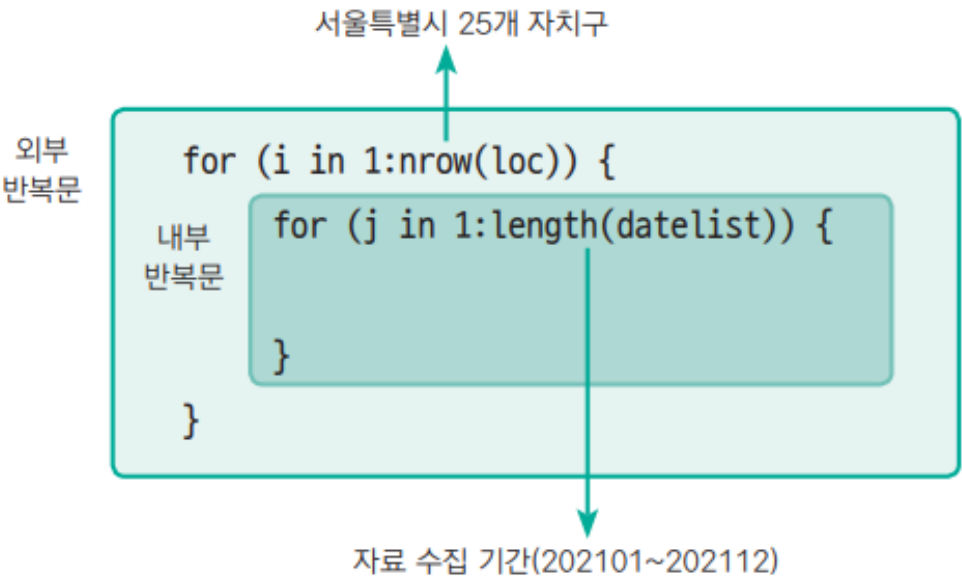


그림 3-1 중첩 반복문 구조

03-2 요청 목록 생성: 자료를 어떻게 요청할까?

2단계 요청 목록 채우기

Do it! 요청 목록 채우기

03_자료수집.R

```
42: for (i in 1:nrow(loc)) {           # 외부 반복: 25개 자치구
43:   for (j in 1:length(datelist)) {   # 내부 반복: 12개월
44:     cnt <- cnt + 1                  # 반복 누적 세기
45:     #---# 요청 목록 채우기 (25 X 12= 300)
46:     url_list[cnt] <- paste0("http://openapi.molit.go.kr:8081/...DataSvcAptTrade?",
47:                             "LAWD_CD=", loc[i,1],           # 지역 코드
48:                             "&DEAL_YMD=", datelist[j],       # 수집 월
49:                             "&numOfRows=", 100,              # 가져올 최대 자료 수
50:                             "&serviceKey=", service_key)     # 인증키
51:   }
52:   Sys.sleep(0.1) # 0.1초간 멈춤
53:   msg <- paste0("[", i, "/", nrow(loc), "] ", loc[i,3], "
                  의 크롤링 목록이 생성됨 => 총 [", cnt, "] 건") # 알림 메시지
54:   cat(msg, "\n\n")
55: }
```

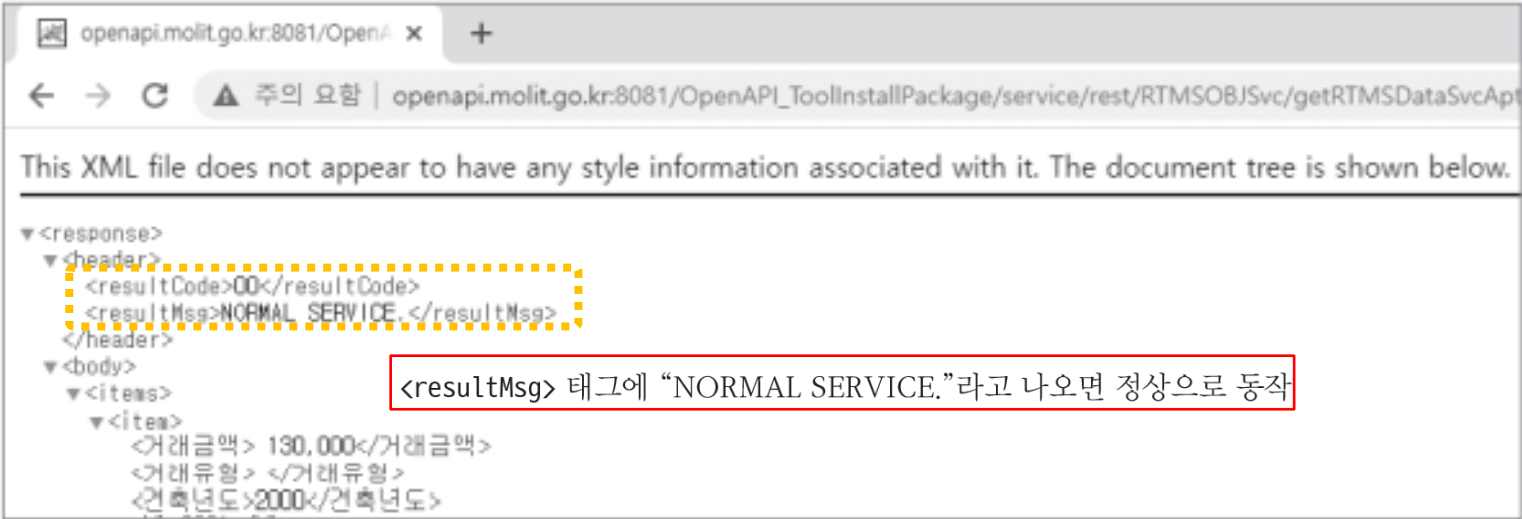
03-2 요청 목록 생성: 자료를 어떻게 요청할까?

3단계 요청 목록 확인하기

Do it! 요청 목록 동작 확인

03_자료수집.R

```
59: length(url_list)           # 요청 목록 개수 확인
60: browseURL(paste0(url_list[1])) # 정상 동작 확인(웹 브라우저 실행)
```



<resultMsg> 태그에 “NORMAL SERVICE.”라고 나오면 정상으로 동작

그림 3-2 요청 URL에 따른 정상 응답 메시지

03-3 크롤러 제작: 자동으로 자료 수집하기

1단계 임시 저장 리스트 만들기

응답 결과인 XML 파일을 저장할 리스트(`raw_data`)와 XML에서 개별 거래 내역만 추출하여 저장할 리스트(`root_Node`), 개별 거래 내역을 순서대로 정리할 리스트(`total`)를 만듭니다.

Do it! 임시 저장 리스트 생성

03_자료수집.R

```
69: library(XML)           # install.packages("XML")
70: library(data.table)     # install.packages("data.table")
71: library(stringr)        # install.packages("stringr")
72:
73: raw_data <- list()        # XML 임시 저장소
74: root_Node <- list()      # 거래 내역 추출 임시 저장소
75: total <- list()          # 거래 내역 정리 임시 저장소
76: dir.create("02_raw_data") # 새로운 폴더 만들기
```

03-3 크롤러 제작: 자동으로 자료 수집하기

2단계 자료 요청하고 응답받기

Do it! URL 요청 - XML 응답

03_자료수집.R

```
80: for(i in 1:length(url_list)) { # 요청 목록(url_list) 반복
81:   raw_data[[i]] <- xmlTreeParse(url_list[i], useInternalNodes = TRUE,
                                encoding = "utf-8") # 결과 저장
82:   root_Node[[i]] <- xmlRoot(raw_data[[i]]) # xmlRoot로 루트 노드 이하 추출
```

03-3 크롤러 제작: 자동으로 자료 수집하기

3단계 전체 거래 건수 확인하기

Do it! 전체 거래 건수 확인

03_자료수집.R

86: items <- root_Node[[i]][[2]][['items']] # 전체 거래 내역(items) 추출
87: size <- xmlSize(items) # 전체 거래 건수 확인

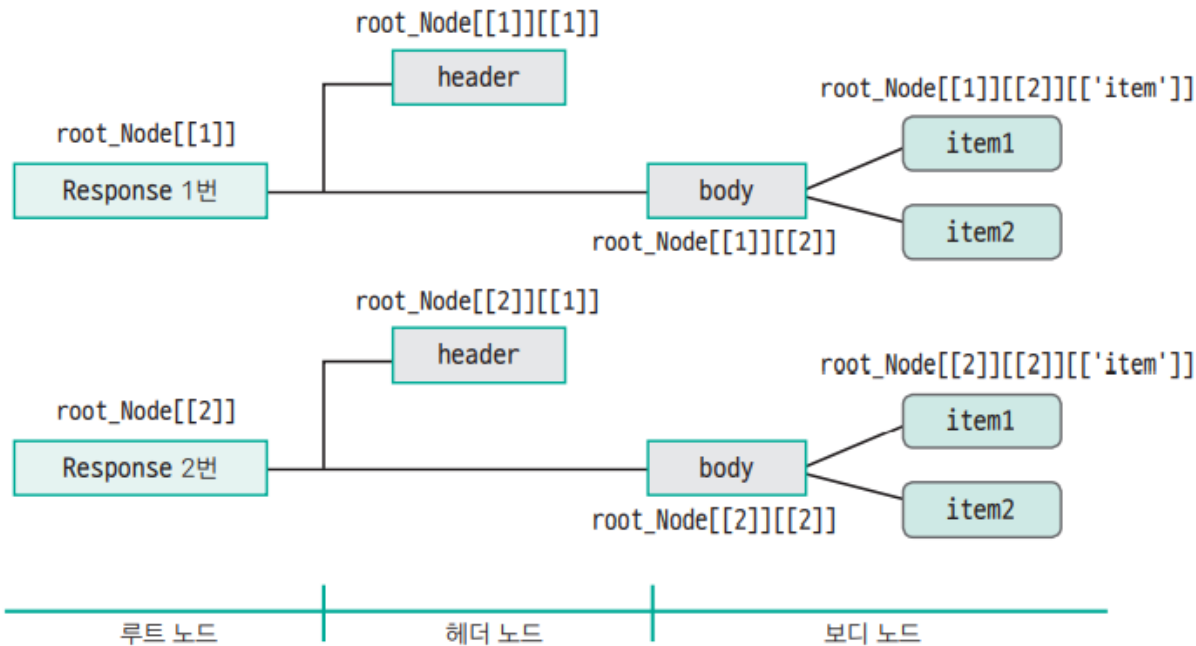


그림 3-3 응답 XML 구조

03-3 크롤러 제작: 자동으로 자료 수집하기

4단계 개별 거래 내역 추출하기



알아 두면
좋아요!

리스트형 자료를 하나로 통합하는 방법

크롤러를 활용하여 데이터를 수집할 때 리스트형 자료를 사용하는 경우가 많습니다. 그러나 데이터를 저장하거나 분석하려면 리스트형보다 데이터프레임형으로 변환하는 것이 편리합니다. `rbindlist()`나 `ldply()`를 사용하면 리스트 안에 포함된 작은 데이터프레임 여러 개를 하나로 결합할 수 있습니다.

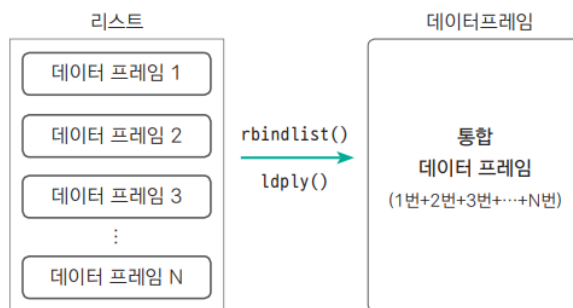


그림 3-4 리스트를 데이터프레임으로 변환하기

Do it! 거래 내역 추출

03_자료수집.R

```
91: item <- list() # 전체 거래 내역(items) 저장 임시 리스트 생성
92: item_temp_dt <- data.table() # 세부 거래 내역(item) 저장 임시 테이블 생성
93: Sys.sleep(.1) # 0.1초 멈춤
94: for(m in 1:size) { # 전체 거래 건수(size)만큼 반복
95:   #---# 세부 거래 내역 분리
96:   item_temp <- xmlSApply(items[[m]],xmlValue)
97:   item_temp_dt <- data.table(year = item_temp[4], # 거래 연도
98:                             month = item_temp[7], # 거래 월
99:                             day = item_temp[8], # 거래 일
100:                             price = item_temp[1], # 거래 금액
101:                             code = item_temp[12], # 지역 코드
102:                             dong_nm = item_temp[5], # 법정동
103:                             jibun = item_temp[11], # 지번
104:                             con_year = item_temp[3], # 건축 연도
105:                             apt_nm = item_temp[6], # 아파트 이름
106:                             area = item_temp[9], # 전용면적
107:                             floor = item_temp[13]) # 층수
108:   item[[m]] <- item_temp_dt } # 분리된 거래 내역 순서대로 저장
109: apt_bind <- rbindlist(item) # 통합 저장
```

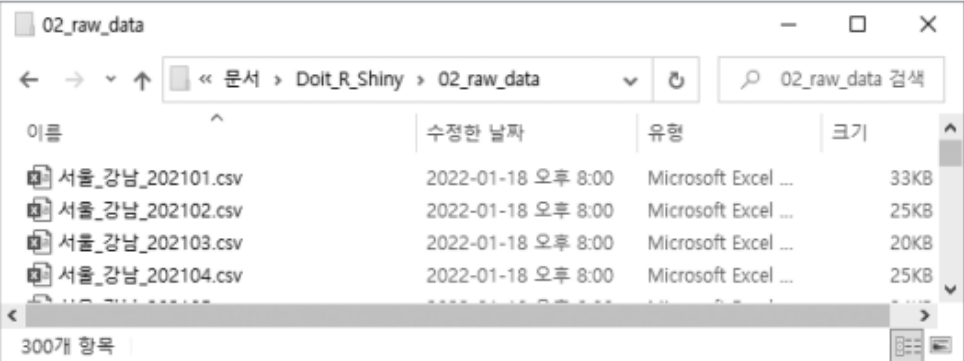
03-3 크롤러 제작: 자동으로 자료 수집하기

5단계 응답 내역 저장하기

Do it! 응답 내역 저장

03_자료수집.R

```
113: region_nm <- subset(loc, code== str_sub(url_list[i],115, 119))$addr_1 # 지역명
114: month <- str_sub(url_list[i],130, 135) # 연월(YYYYMM)
115: path <- as.character(paste0("./02_raw_data/", region_nm, "_", month, ".csv"))
116: write.csv(apt_bind, path) # CSV 저장
117: msg <- paste0("[", i, "/", length(url_list),
                  "] 수집한 데이터를 [", path, "]에 저장 합니다.") # 알림 메시지
118: cat(msg, "\n\n")
119: }
```



	A	B	C	D	E	F	G	H	I	J	K	L
1		year	month	day	price	code	dong_nm	jibun	con_year	apt_nm	area	floor
2	1	2021	1	5	31,000	11680	역삼동	720-25	2002	대우디오빌	30.03	4
3	2	2021	1	6	61,000	11680	역삼동	766-8	2002	트레벨	33.48	3
4	3	2021	1	7	198,000	11680	역삼동	757	2005	역삼래미안	59.73	14
5	4	2021	1	7	193,800	11680	역삼동	757	2005	역삼래미안	59.4	16
6	5	2021	1	9	91,000	11680	역삼동	796-29	2012	강남서해더블루	66.04	10
7	6	2021	1	11	80,000	11680	역삼동	783-3	2002	갤러리하우스	84.95	1
8	7	2021	1	11	90,000	11680	역삼동	832-5	2006	역삼디오슈페리움	46.9	16
9	8	2021	1	11	255,000	11680	역삼동	754-1	2006	역삼푸르지오	84.9097	7
10	9	2021	1	12	85,800	11680	역삼동	713-11	2006	역삼TPARK	28.246	10
11	10	2021	1	13	99,000	11680	역삼동	796-29	2012	강남서해더블루	80.2	7
12	11	2021	1	14	198,000	11680	역삼동	757	2005	역삼래미안	59.53	8

그림 3-5 응답 내역을 저장한 CSV 파일

03-4 자료 정리: 자료 통합하기

1단계 CSV 파일 통합하기

Do it! CSV 파일 통합

03_자료수집.R

```
128: setwd(dirname(rstudioapi::getSourceEditorContext()$path)) # 작업 폴더 설정
129: files <- dir("./02_raw_data") # 폴더 내 모든 파일명 읽기
130: library(plyr) # install.packages("plyr")
131: apt_price <- ldply(as.list(paste0("./02_raw_data/", files)), read.csv) # 결합
132: tail(apt_price, 2) # 확인
```

실행 결과

	X	year	month	day	price	code	(... 생략 ...)
159307	215	2020	12	30	68,000	11260	(... 생략 ...)
159308	216	2020	12	30	55,000	11260	(... 생략 ...)

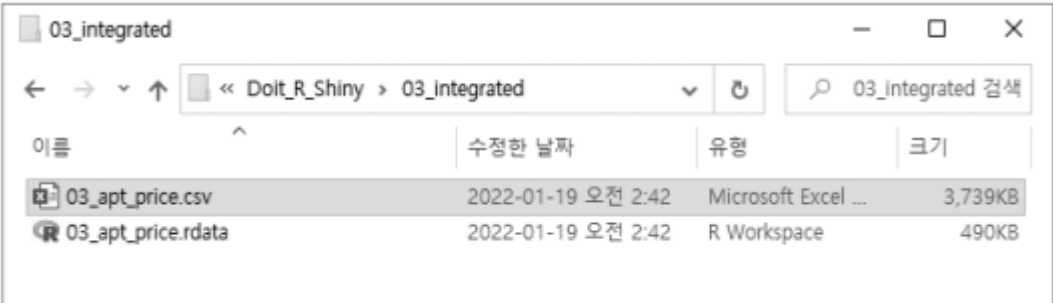
03-4 자료 정리: 자료 통합하기

2단계 통합 데이터 저장하기

Do it! RDATA와 CSV 형식으로 저장

03_자료수집.R

```
136: dir.create("./03_integrated") # 새로운 폴더 생성
137: save(apt_price, file = "./03_integrated/03_apt_price.rdata") # 저장
138: write.csv(apt_price, "./03_integrated/03_apt_price.csv")
```



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		X	year	month	day	price	code	dong_nm	jibun	con_year	apt_nm	area	floor
2	1	1	2021	1	5	31,000	11680	역삼동	720-25	2002	대우디오빌	30.03	4
3	2	2	2021	1	6	61,000	11680	역삼동	766-8	2002	트레벨	33.48	3
4	3	3	2021	1	7	198,000	11680	역삼동	757	2005	역삼래미안	59.73	14
5	4	4	2021	1	7	193,800	11680	역삼동	757	2005	역삼래미안	59.4	16
6	5	5	2021	1	9	91,000	11680	역삼동	796-29	2012	강남서해더블루	66.04	10
7	6	6	2021	1	11	80,000	11680	역삼동	783-3	2002	갤러리하우스	84.95	1
8	7	7	2021	1	11	90,000	11680	역삼동	832-5	2006	역삼디오슈퍼리움	46.9	16
9	8	8	2021	1	11	255,000	11680	역삼동	754-1	2006	역삼푸르지오	84.9097	7
10	9	9	2021	1	12	85,800	11680	역삼동	713-11	2006	역삼IPARK	28.246	10
11	10	10	2021	1	13	99,000	11680	역삼동	796-29	2012	강남서해더블루	80.2	7
12	11	11	2021	1	14	198,000	11680	역삼동	757	2005	역삼래미안	59.53	8
13	12	12	2021	1	15	24,000	11680	역삼동	606-18	1999	현대휴먼터치빌	25.92	6
14	13	13	2021	1	16	291,000	11680	역삼동	711-3	2016	역삼자이	114.019	12
15	14	14	2021	1	16	38,500	11680	역삼동	720-25	2002	대우디오빌	39.13	9
16	15	15	2021	1	16	80,000	11680	역삼동	720-25	2002	대우디오빌	59.505	19
17	16	16	2021	1	18	77,500	11680	역삼동	794-19	2006	강남한솔	74.61	2

그림 3-6 한 파일로 통합한 데이터



단골 코드 정리하기

• 날짜로 연속형 변수 만들기

```
seq(from = as.Date('1990-01-01'), # 시작 시점  
    to   = as.Date('2020-12-31'), # 종료 시점  
    by   = '1 year')              # 단위
```

• 중첩 반복문 만들기

```
for (i in 1:3) { # 외부 반복문  
  for (j in 1:3) { # 내부 반복문  
    Sys.sleep(0.2) # 0.1초 멈춤  
    print(paste(i,j,sep=","))  
  }  
}
```



• XML 자료 저장하기

```
#---# 주소 가져오기
URL <- "https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Frestaurants.xml"
#---# https://~를 http://~로 변경하고 저장
file <- xmlTreeParse(sub("s", "", URL), useInternal = TRUE)
#---# 저장된 XML을 데이터프레임으로 변환
file <- xmlToDataFrame(file)
#---# 행렬 바꾸기(matrix transpose)
file <- as.data.frame(t(file))
```