오픈API 연동 (1) - JSON 데이터의 처리

JSON은 경량의 데이터 표현 형식으로 최근에는 웹, 모바일 등을 중심으로 **서로 다른 플랫폼간의 데이터 교환**을 위하여 활용되고 있는 사실상의 산업 표준 입니다. 특히 OpenAPI등을 통해 다른 곳에서 제공하는 데이터를 수집할 경우 JSON 형식으로 되어 있는 경우가 많기 때문에 R에서도 HTTP 통신을 기반으로 OpenAPI로부터 JSON 형식의 데이터를 수집하고 이를 데이터 프레임으로 변환하는 기능을 제공하고 있습니다.

#01. JSON 구조의 이해

이름(key)와 값(value)의 쌍을 이루는 구조

```
{ "이름": "값", "이름": "값" ... }
```

> 값에는 숫자, 문자열, 논리값(true/false) 모두 가능하며 값이 문자열인 경우는 쌍따옴표나 홑따옴표로 감싸서 표현한다.

값을 배열 형식으로 구성가능

하나의 이름에 여러 개의 값을 포함시키고, 0부터 시작되는 인덱스 번호를 통해 값에 접근한다.

```
{ "이름": [ "값0", "값1", "값2" ] }
```

계층화 된 데이터의 표현

```
{
    "이름": { "이름": "값", "이름": "값" },
    "이름": { "이름": "값", "이름": "값" }
};
```

목록형 데이터의 표현

가장 일반적인 형식.

하나의 key에 대응되는 값이 배열 형태이고 배열의 각 원소가 동일한 구조를 갖는 JSON들로 구성된다.

> article[0].subject --> 글 제목

```
{ "article" : [
    {"subject": "글 제목", "content": "글 내용", "writer": "작성자", "date": "작성일"},
    {"subject": "글 제목", "content": "글 내용", "writer": "작성자", "date": "작성일"},
    ...
    {"subject": "글 제목", "content": "글 내용", "writer": "작성자", "date": "작성일"}
]}
```

#02. 필요한 패키지 로드하기

```
REPO_URL <- "https://cran.seoul.go.kr/"
if (!require(httr))   install.packages("httr", repos=REPO_URL)
if (!require(rjson))  install.packages("rjson", repos=REPO_URL)
if (!require(dplyr))  install.packages("dplyr", repos=REPO_URL)

library(httr)       # 온라인상의 데이터를 가져오기 위한 통신 기능 패키지
library(rjson)      # JSON 처리 패키지
library(dplyr)      # 데이터 전처리 패키지
```

#03. key, value의 쌍으로 구성된 단순 구조의 경우

1) 통신으로 온라인 상의 컨텐츠 가져오기

```
# 가져올 데이터 URL
simple_json_url <- "http://www.itpaper.co.kr/demo/r/simple.json"

# HTTR 패키지로 데이터 가져오기
simple_resp <- GET(simple_json_url)
simple_resp
```

▶ 출력결과

```
Response [http://www.itpaper.co.kr/demo/r/simple.json]
  Date: 2020-01-13 08:16
  Status: 200
  Content-Type: application/json
  Size: 113 B
{
    "name": "갤럭시 S6",
    "type": "삼성",
    "img": "http://itpaper.co.kr/demo/app/img/GalaxyS6.png"
}
```

2) 가져온 데이터를 R에서 활용하기 위해 변환하기

텍스트 형식으로 추출

모든 json은 인코딩이 `UTF-8` 이므로 encoding옵션은 항상 `utf-8` 이라고 지정한다.

```
simple_text <- content(simple_resp, as="text", encoding="utf-8")
simple_text
```

▶ 출력결과

```
'{\n    "name": "갤럭시 S6",\n    "type": "삼성",\n    "img": "http://itpaper.co.kr/demo/app/img/GalaxyS6.png'
```

R의 list 타입으로 변환

`as` 파라미터에 "parse"라고 지정한다.

```
simple_list <- content(simple_resp, as="parse", encoding="utf-8")
typeof(simple_list)
simple_list
```

▶ 출력결과

```
'list'

$name   '갤럭시 S6'
$type   '삼성'
$img    'http://itpaper.co.kr/demo/app/img/GalaxyS6.png'
```

3) 변환된 데이터의 하위 요소에 접근하기

json 데이터의 계층은 `$` 를 사용해서 구분한다.

```
simple_list$name
```

▶ 출력결과

```
'갤럭시 S6'
```

```
simple_list$type
```

▶ 출력결과

```
'삼성'
```

```
simple_list$img
```

▶ 출력결과

```
'http://itpaper.co.kr/demo/app/img/GalaxyS6.png'
```

#04. 계층화된 구조에 접근하기

1) 통신으로 온라인 상의 컨텐츠 가져오기

```
# 가져올 데이터 URL
phone_json_url <- "http://www.itpaper.co.kr/demo/r/phone.json"

# HTTR 패키지로 데이터 가져오기
phone_resp <- GET(phone_json_url)
phone_resp
```

▶ 출력결과

```
Response [http://www.itpaper.co.kr/demo/r/phone.json]
  Date: 2020-01-13 08:16
  Status: 200
  Content-Type: application/json
  Size: 260 B
{
        "rt": "OK",
        "rtmsg": "SUCCESS",
        "item": {
        "name": "갤럭시 S6",
        "type": "삼성",
        "img": "http://itpaper.co.kr/demo/app/img/GalaxyS6.png",
        "price": {
                "fixed": 1000000,
                "sale": 850000
...
```

2) 가져온 데이터를 R에서 활용하기 위해 변환하기

```
phone_list <- content(phone_resp, as="parse", encoding="utf-8")
phone_list
```

▶ 출력결과

```
$rt     'OK'
$rtmsg  'SUCCESS'
$item   $name   '갤럭시 S6'
        $type   '삼성'
        $img    'http://itpaper.co.kr/demo/app/img/GalaxyS6.png'
        $price      $fixed      1000000
                    $sale       850000
```

3) 변환된 데이터의 하위 요소에 접근하기

json 데이터의 계층은 `$` 를 사용해서 구분한다.

```
phone_list$rt
phone_list$rtmsg
phone_list$item$name
phone_list$item$type
phone_list$item$img
phone_list$item$price$fixed
phone_list$item$price$sale
```

▶ 출력결과

```
'OK'
'SUCCESS'
'갤럭시 S6'
'삼성'
'http://itpaper.co.kr/demo/app/img/GalaxyS6.png'
1000000
850000
```

#05. 목록 형식의 구조에 접근하기

1) 통신으로 온라인 상의 컨텐츠 가져오기

```
# JSON List URL
list_json_url <- "http://www.itpaper.co.kr/demo/r/student.json"

# HTTR 패키지로 데이터 가져오기
list_resp <- GET(list_json_url)
list_resp
```

▶ 출력결과

```
Response [http://www.itpaper.co.kr/demo/r/student.json]
  Date: 2020-01-13 08:16
  Status: 200
  Content-Type: application/json
  Size: 171 B
{
        "student": [
                        {"name":"철수","math":85,"kor":80,"eng":70},
                        {"name":"영이","math":92,"kor":70,"eng":85},
                        {"name":"순이","math":61,"kor":100,"eng":72}
        ]
}
```

2) 가져온 데이터를 R에서 활용하기 위해 변환하기

```
student_list <- content(list_resp, as="parse", encoding="utf-8")
student_list
```

▶ 출력결과

```
$student    =

1.$name     '철수'
  $math     85
  $kor      80
  $eng      70

2.$name     '영이'
  $math     92
  $kor      70
  $eng      85

3.$name     '순이'
  $math     61
  $kor      100
  $eng      72
```

3) DataFrame으로 변환

목록형 JSON의 경우 배열에 해당하는 부분을 DataFrame으로 변환할 수 있다.

```
stud_df <- bind_rows(student_list$student)
stud_df
```

▶ 출력결과

A tibble: 3 × 4

name	math	kor	eng
<chr>	<int>	<int>	<int>
철수	85	80	70
영이	92	70	85
순이	61	100	72

4) 데이터 전처리

컬럼이름 변경

```
이름변경 <- rename(stud_df, '이름'=name, '수학'=math, '국어'=kor, '영어'=eng)
이름변경
```

▶ 출력결과

A tibble: 3 × 4

이름	수학	국어	영어
<chr>	<int>	<int>	<int>
철수	85	80	70
영이	92	70	85
순이	61	100	72

총점,평균

```
df <- 이름변경 %>%
        mutate(총점=수학+국어+영어, 평균=총점/3)
df
```

▶ 출력결과

A tibble: 3 × 6

이름	수학	국어	영어	총점	평균
<chr>	<int>	<int>	<int>	<int>	<dbl>
철수	85	80	70	235	78.33333
영이	92	70	85	247	82.33333
순이	61	100	72	233	77.66667