

정오표 및 활용 안내

(빅데이터 분석의 첫걸음, R로 배우는 코딩)

2018. 11. 5.

■ p.18. 아래에서 2행

2016년의 → 2017년의

■ p.43. 말풍선 2행

c(170, 155, 165) → x <- c(170, 155, 165)

■ p.82. R (리스트 만들기) 2행

X → x

■ p.88. 잠깐! 2행

x[3] <- lapply(x[3], as.character) → x[,3] <- lapply(x[,3], as.character)

■ p.108(4장), p.178, p.182, p.183, p.185(7장)

geocode() 함수의 문제:

(1) geocode()는 geocode(..., source="google")이 디폴트로서, 비즈니스 목적이 아닌 경우의 트래픽 제한을 위해 지오코드 변환 값이 한 개 또는 두 개 정도가 반환되며, 나머지는 NA로 출력되기도 함

(2) 영문 지명인 경우, geocode(..., source="dsk")를 사용하면 정상적으로 출력되지만, 한글의 경우에는 '시' 단위까지 가능하나 일부 시는 서비스 안됨(서울은 '구' 단위까지 가능)

[사용 예]

```
addr <- c("서울", "대구", "부산", "광주", "제주", "춘천")
gc <- geocode(enc2utf8(addr), source="dsk")
gc
```

☞ 현재, 구글이 지도 등의 서비스 이용에 대한 유료화로 전환되고 있는 과정으로, 기존 서비스에 대한 무료 사용의 제약이 발생하고 있음

(3) 네이버 API를 이용한 지오코드 확인 방법


① API 발급 (Client ID, Client_Secret) (10장 뉴스/블로그 검색 p.265 참조)

<https://developers.naver.com/docs/common/openapiguide/apilist.md>

비로그인 방식 오픈 API

비로그인 방식 오픈 API는 HTTP 헤더에 클라이언트 아이디와 클라이언트 시크릿 값만 전송해 사용할 수 있는 오픈 API입니다. 네이버 아이디로 로그인 인증을 통한 접근 토큰을 획득할 필요가 없습니다.

다음과 같은 네이버 오픈API가 비로그인 방식 오픈 API입니다.

- **검색**: 네이버 검색 결과를 뉴스, 백과사전, 블로그, 쇼핑, 영화, 웹 문서, 전문정보, 지식iN, 책, 카페글 등 분야별로 볼 수 있는 API입니다. 그 외에 지역 검색 결과와 성인 검색어 판별 기능, 오타 변환 기능을 제공합니다.
- **공유하기**: 콘텐츠를 네이버 블로그, 네이버 카페, PHOLAR에 공유할 수 있게 하는 API입니다.
- **단축URL**:  원본 URL을 `http://me2.do/example` 과 같은 형태의 짧은 URL로 반환받을 수 있는 API입니다.
- **지도**: 앱 서비스나 애플리케이션에 네이버 지도를 활용할 수 있게 하는 API입니다.
- **캡차(이미지)**: 네이버 서비스에서 사용하는 이미지 캡차 기능을 외부 서비스에 사용할 수 있게 하는 API입니다.
- **Clova Face Recognition**: 입력된 사진 이미지 속의 얼굴을 인식하거나 얼굴 감지를 이용한 애플리케이션을 만들 수 있게 하는 API입니다.
- **Clova Speech Recognition**: 음성 데이터(한국어, 영어, 일어, 중국어(간체))를 인식해 텍스트를 반환하는 API입니다.
- **Clova Speech Synthesis**: 입력된 텍스트를 성우의 낭독 음성(한국어, 영어, 일본어, 중국어(간체))으로 변환해 반환하는 API입니다.
- **Papago NMT 번역**: 인공 신경망 기술 기반의 기계 번역 결과를 반환하는 API입니다(지원 언어: 영어, 중국어(간체)).
- **Papago SMT 번역**: 통계 기반의 기계 번역 결과를 반환하는 API입니다(지원 언어: 영어, 일본어, 중국어(간체, 번체)).

지도 > JavaScript v3

네이버 지도 API v3는 JavaScript 형태로 제공되는 NAVER 지도 플랫폼으로써, 웹 서비스 또는 애플리케이션에 NAVER 지도 기능을 구현할 수 있도록 다양한 클래스와 메서드를 제공합니다. 특히, 높은 성능을 위해 데스크톱과 모바일 환경에 최적화된 코드를 제공하고 있으며, 주요 웹 브라우저를 완벽하게 지원할 수 있도록 새롭게 설계되었습니다.

네이버 지도 API v3는 별도의 개발 가이드 사이트를 제공합니다.

[네이버 지도 API v3 사이트 바로가기 >](#)

Hello, NAVER Maps API

네이버 지도 API는 무료로 제공됩니다.
클라이언트 ID를 등록하고, 지금 바로 사용해 보세요.

[오픈 API 이용 신청 >](#)

[등록 과정: pp.266-267 참조]

NAVER Developers

애플리케이션 등록 (API 이용신청)

애플리케이션의 기본 정보를 등록하면, 좌측 **내 애플리케이션** 메뉴의 서브 메뉴에 등록하신 애플리케이션 이름으로 서브 메뉴가 만들어 집니다.

애플리케이션 이름	<div>애플리케이션 이름 ⓘ</div> <ul style="list-style-type: none">• 네이버 아이디로 로그인할 때 사용자에게 표시되는 이름이므로 가급적 10자 이내의 간결한 이름을 사용해주세요.• 40자 이내의 영문, 한글, 숫자, 공백문자, "-", "_" 만 입력 가능합니다.
사용 API	<div>선택하세요. ▼</div> <div><div>지도 (모바일) ×</div><div>지도 (웹) ×</div></div>
비로그인 오픈 API 서비스 환경	<div>환경 추가 ⓘ</div>

② 지오코드 확인 R 소스 (10장 pp.271-272 참조)

```
install.packages("RCurl")
install.packages("XML")
library(RCurl)
library(XML)

# 주소 -> 좌표 변환 API
searchUrl <- "https://openapi.naver.com/v1/map/geocode.xml?encoding=utf-8&coordType=latlng&query="

# 발급받은 키 설정
Client_ID <- "JIAd.....Awst"
Client_Secret <- "huy.....NRP"

# 주소: 서울역
query <- URLencode(iconv("서울특별시 용산구 한강대로 405", "euc-kr", "UTF-8"))
url <- paste(searchUrl, query, sep="")

# 지오코드 요청
doc <- getURL(url,
               httpheader = c('X-Naver-Client-Id' = Client_ID,
                              'X-Naver-Client-Secret' = Client_Secret))

# XML 문서 파싱
doc2 <- htmlParse(doc, encoding="UTF-8")
# 경도
long <- xpathSApply(doc2, "//items//x", xmlValue)
long
# 위도
lat <- xpathSApply(doc2, "//items//y", xmlValue)
lat
```

☞ 네이버 개발자 센터(API 공통 가이드 참조)

<https://developers.naver.com/docs/common/openapiguide/apilist.md#비로그인-방식-오픈-api>

■ p.117. 오른쪽 그래프 Y축 라벨

발생 건수 → 지진 강도

■ p.165. R (1초 간격으로 그래프 그리기) 4행

y <- runif(5, 0, 1), → y <- runif(5, 0, 1)

■ p.223. R (출력결과의 이미지 저장) 3행

(F) → wordcloud(names(word_count), freq=word_count, scale=c(6,0.3), min.freq=3, random.order=F, rot.per=.1, colors=pal2)

■ p.226. R (연설문의 단어에 대한 워드 클라우드 만들기) 5행

wordcloud(names (wordcount),) → wordcloud(names(wordcount),)

■ p.252. R 3행

df_busRoute <- subest(df, busRouteNm==busRtNm) →

df_busRoute <- subset(df, busRouteNm==busRtNm)

■ p.252. R 5행

df_busRoute\$busRouteID →

df_busRoute\$busRouteId

■ p.280. R 1~2행 (XMP과 Rcurl이 설치된 경우, 실행 불필요)

install.packages("XML") →

install.packages("RCurl")

install.packages("XML")

■ p.291. 플로차트

Url <- c"http://....." → url <- "http://....."

■ p.292. R (웹 스크래핑) 2행

url <- http://..... → url <- "http://....."

■ p.292. R (웹 스크래핑) 3행 (웹페이지 크기가 클 때)

```
url <- "http://...."  
doc <- htmlParse(url, encoding="UTF-8") →  
url <- "http://...."  
web_page <- readLines(url)  
doc <- htmlParse(web_page, encoding="UTF-8")
```

■ p.301. R (웹 스크래핑) 6행

```
doc <- htmlParse(url2, encoding="UTF-8") →  
web_page <- readLines(url2)  
doc <- htmlParse(web_page, encoding="UTF-8")
```

■ p.301. R (웹 스크래핑) 7행

```
prod_name <- xpathSApply(doc, "//ul[@id='productList']//dd[@class  
='name']", xmlValue) →  
prod_name <- xpathSApply(doc, "//ul[@id='productList']//div[@class  
='name']", xmlValue)
```

■ p.302. R 1행

```
price <- xpathSApply(doc, "//ul[@id='productList']//strong[@ 'pricevalue']",  
xmlValue) →  
price <- xpathSApply(doc, "//ul[@id='productList']//strong[@class=  
'pricevalue']", xmlValue)
```

■ p.340. R 1행

```
average.path.length(g_star) → average.path.length(g_ring)
```

■ p.364. 제목

④ 연결 정도 분포 그리기 → [4] 연결 정도 분포 그리기

■ p.365. 제목

⑤ 가장 많이 협업한 연구자 10명의 리스트 출력 →

[5] 가장 많이 협업한 연구자 10명의 리스트 출력

■ p.410. 6단계, 예측 1행, 3행

122~127 → 122~126

■ p.411. R (그래픽 출력) 1행

71:100 → 82:121

71:105 → 82:126

■ p.411. R (그래픽 출력) 3행

101:105 → 122:126

■ p.411. R (그래픽 출력) 4행

v=100 → v=121

- 끝 -