

빅데이터 수집 part 2

# R 크롤링 따라하기



한국기술교육대학교  
온라인평생교육원





## 학습목표



- 중고차 사이트 크롤링하기
- 영화 감상평 크롤링하기



- R을 이용한 크롤링을 통해 중고차 사이트의 데이터를 수집할 수 있다.
- R을 이용한 크롤링을 통해 영화 감상평 데이터를 수집할 수 있다.



# 중고차 사이트 크롤링하기

## 1 실습개요

### 실습 사이트

- 'B' 중고차 사이트  
: <http://www.bobaedream.co.kr/cyber/CyberCar.php?gubun=K&page=1>

### 파싱 방법

- 태그와 css를 이용한 파싱



## 중고차 사이트 크롤링하기

### 2 실습절차



Rvest 패키지 설치하기



HTML태그 가져오기



관심 항목에 대한 HTML태그 가져오기

가져올 HTML태그	■ 자동차 종(type)	■ 연료종류(fuel)
	■ 변속기(transmission)	■ 거리(distance)



데이터를 하나의 자료구조로 바꾸기 : data.frame 형태로 바꾸기



data.frame에 담긴 최종 정보 확인하기



# 중고차 사이트 크롤링하기

## 3 실습보기



### 1 Html 태그 검색하기

- 보배드림 홈페이지 → [사이버 매장] → [국산차 매장]
- → [F12] 키를 눌러 Element 창 열기 → 태그를 선택하여 필요한 태그 찾기



### 2 Rvest 패키지 설치

```
> install.packages('rvest')
```

- R 실행 → install.packages('rvest') : Rvest 패키지 설치



### 3 Library 설치

- library(rvest) → library(stringr)



### 4 Title 내용 가져오기

- 2페이지 선택 후 1페이지로 되돌아가 PageNum이 있는 URL 가져오기
- → URL 입력 → read\_html(url) : 태그 가져오기 → usedCar : 가져온 태그들 확인

```
> url <-  
  "http://www.bobaedream.co.kr/cyber/CyberCar.php?gubun=K&page=1&order=S11"  
> usedCar <- read_html(url)  
> usedCar
```

- 사이트에서 태그 확인 : 띄어쓰기는 점(.)으로 대체
- html\_nodes(usedCar, css = '.mode-cell.title') : 가져온 태그 입력

```
> carInfo <- html_nodes(usedCar, css = '.mode-cell.title')  
> carInfo
```

- 'mode-cell.title' 안의 'tit ellipsis'의 내용을 가져와야 함
- titles <- carinfo %>% html\_nodes('.tit.ellipsis') %>% html\_text()
- → titles → title 내용만 가져온 것을 확인

```
> titles <- carinfo %>% html_nodes('.tit.ellipsis') %>% html_text()
```



## 중고차 사이트 크롤링하기

### 3 실습보기

5

#### 불필요한 문자 제거하기

```
> gsub("제거할 문자", "", title)
```

- `gsub("제거할 문자", "", title) → titles` → 불필요한 문자 제거된 화면 출력

6

#### 기타 항목 가져오기

```
> carDetailInfo <- carinfo %>% html_nodes('.data.clearfix') %>%
  html_text()
> carDetailInfo
```

- 변속기, 연료, 거리만 필요하므로 [4], [5], [6]열만 가져오는 작업이 필요함
- `gsub(" ", "", str_split(carDetailInfo[1], '\\n')) : '\\n'을 기준으로 문자분리`
- `splitFunction`을 이용하여 [4], [5], [6]열의 데이터만 분리

```
> splitFunction <- function(row){
+ a <- gsub(" ", "", str_split(row, '\\n') [[1]])}
+ return (list(c(a[4],a[5],a[6]))) }
>
> carDetailInfo <- lapply(carinfo %>%
  html_nodes('.data.clearfix') %>% html_text, splitFunction)
> carDetailInfo
```

7

#### Matrix로 표현하기

```
> carDetailInfo <- matrix(unlist(carDetailInfo), ncol=3, byrow=T)
```



## 중고차 사이트 크롤링하기

### 3 실습보기



#### 8 data.frame 형태로 표현

```
> transmission <- carDetailInfo[,1]
> fuel <- carDetailInfo[,2]
> distance <- as.numeric(str_replace_all(carDetailInfo[,3], '[,a-z]', ''))
> data <- data.frame(titles, transmission, fuel, distance)
>
> data
```

- Title과 Matrix 정보 종합하기
- [1]열 : 변속기/[2]열 : 연료/[3]열 : 거리(숫자로 표현하기 위해 영문 제거)
- 변속기, 연료, 거리의 내용을 정리된 결과 확인



# 영화 감상평 크롤링하기

## 1 실습개요

### 실습 사이트

- 'D' 영화 - 영화 '어거스트 러쉬' 감상평  
: <http://movie.daum.net/moviedb/main?movieId=43104>

### 가져올 HTML태그

- 리뷰

➤ 출처 : 안재준 교수, 인공지능망이란 무엇인가?, Creative & Smart, 2017, <http://blog.lgcns.com/1359>





## 영화 감상평 크롤링하기

---

### 2 실습절차

- 1 패키지 설치하기
- 2 [개발자 도구창]에서 리뷰가 적힌 태그 찾기
- 3 HTML가져오기
- 4 데이터를 하나의 자료구조로 바꾸기 : TermDocumentMatrix 형태로 바꾸기
- 5 최종 정보 확인하기
- 6 결과 시각화하기 : qqgraph 명령어로 시각화 하기



# 영화 감상평 크롤링하기

## 3 실습보기



### 1 Html 태그 검색하기

- Daum 홈페이지 → [영화] → '어거스트 러쉬' 검색
- → [평점] 선택 → [F12] 키를 눌러 Element 창 열기 → 태그를 선택하여 필요한 태그 찾기
- 리뷰 태그  
: <p class = "desc\_review">...</p>/감상평이 없는 곳은 빈칸으로 표시됨



### 2 Library 설치

- library()로 필요한 Library 설치/ 필요 시 install package()로 필요한 패키지 설치

```
> library(rvest)
> library(httr)
> library(KoNLP)
> library(stringr)
> library(tm)
> library(qgraph)
```



### 3 URL 입력하기

- 2페이지 선택 후 1페이지로 되돌아가 PageNum 전까지의 주소 가져오기
- 1~500 페이지를 가져올 예정이므로 페이지 숫자 부분은 비워둠

```
> url_base <-
  'http://movie.daum.net/moviedb/grade?movieId=43104&type=netizen&page='
```



## 3 실습보기



### 4 리뷰 가져오기

```
> all.reviews <- c()

> for(page in 1:500)
+ url <- paste(url_base, page, sep="")
+ htxt <- read_html(url)
+ comments <- html_nodes(htxt, 'div') %>% html_nodes('p')
+ reviews <- html_text(comments)
+ reviews <- repair_encoding(reviews, from = 'utf-8')
+ reviews
+ if( length(reviews) == 0 ){
+ break
+ }
+ reviews <- str_trim(reviews)
+ all.reviews <- c(all.reviews, reviews)}
```

- for(page in 1:500) : 페이지를 늘리거나 줄이려면 숫자 수정
- html\_nodes('p') : p태그를 불러오기
- all.reviews에 가져오기 명령어 모두 포함
- all.reviews 실행 시 감상평이 없는 평점도 모두 수집됨



### 5 감상평 없는 평점 제거하기

```
> all.reviews <- all.reviews[!str_detect(all.reviews, "평점")]
> all.reviews
```

- all.reviews <- all.reviews[!str\_detect(all.reviews, "평점")] : 불필요한 단어가 포함된 내용 제거
- ! : 반대로 하라는 의미('평점'을 검색하여 없는 것을 all.reviews에 삽입하라는 의미)
- all.reviews 실행 시 '평점'이 포함된 글들 모두 제거됨



### 3 실습보기



#### 6 명사/형용사 추출하기

```
> ko.words <- function(doc){
+ d <- as.character(doc)
+ pos <- paste(SimplePos09(d))
+ extracted <- str_match(pos, '([가-힣]+)/[NP]')
+ keyword <- extracted[,2]
+ keyword[!is.na(keyword)]}
```

- 문장에서 말뭉치들을 분리하기
- SimplePos09 : 문장의 각 형태 분리/str\_match : '가-힣' 까지의 단어와 함께 N(명사), P(형용사)만을 가져오도록 함

```
> options(mc.cores=1)
> cps <- VCorpus(VectorSource(all.reviews))
> tdm <- TermDocumentMatrix(cps,
+ control=list(tokenize=ko.words,
+ removePunctuation=T,
+ removeNumbers=T,
+ wordLengths=c(2, 6),
+ weighting=weightBin))
>
> tdm.matrix <- as.matrix(tdm)
> rownames(tdm.matrix)[1:100]
```

- VCorpus : 말뭉치들을 Matrix 형태로 생성함
- tdm(TermDocumentMatrix) : Document별 단어 빈도수를 Matrix 형태로 보여줌



## 3 실습보기

7

### 결과 시각화(Matrix)

```
> word.count <- rowSums(tdm.matrix)
> word.order <- order(word.count, decreasing=T)
> freq.words <- tdm.matrix[word.order[1:20], ]
> co.matrix <- freq.words %*% t(freq.words)
> co.matrix
```

- `word.count <- rowSums(tdmMatrix)` : 각 단어별 합계 구하기
- `word.order <- order(word.count, decreasing=T)` : 단어들을 쓰인 횟수에 따라 내림차순으로 정렬
- 단어별 노출 빈도수가 많은 20개의 단어를 추려내고, 행렬 곱을 하여 빈도수가 높은 단어를 수치로 나타냄
- 대각선의 숫자 : '영화'라는 단어의 노출 빈도수/하나의 감상평에서 단어별로 연관성을 빈도수로 표현

8

### 결과 시각화(Graph)

```
> qqgraph(co.matrix, labels=rownames(co.matrix),
+ diag=F,
+ layout='spring',
+ edge.color='blue',
+ vaixe=log(diag(co.matrix))*2)
```

- `qqgraph` 명령어 → [Enter]
- 연관성이 높은 단어별로 크기와 선 굵기로 나타남



### 중고차 사이트 크롤링하기

#### + 실습개요

- 실습 사이트  
: 'B' 중고차 사이트  
(<http://www.bobaedream.co.kr/cyber/CyberCar.php?gubun=K&page=1>)
- 파싱 방법 : 태그와 css를 이용한 파싱

#### + 실습절차

- ① Rvest 패키지 설치하기 : `install.packages('rvest')`
- ② HTML태그 가져오기
- ③ 관심 항목에 대한 HTML태그 가져오기
  - [F12] 키 → [개발자 도구창] 확인 → 웹 페이지에 대응하는 HTML태그 확인  
→ 해당 태그 안의 모든 정보 가져오기
- ④ 묶여있는 정보 분리하기 : 'stringr' 사용
- ⑤ 데이터를 하나의 자료구조로 바꾸기 : list 정보를 matrix로 바꾸기



### 영화 감상평 크롤링하기

#### + 실습개요

- 실습 사이트  
: 'D' 영화 - 영화 '어거스트 러쉬' 감상평  
(<http://movie.daum.net/moviedb/main?movieId=43104>)
- 가져올 HTML태그 : 리뷰

#### + 실습절차

- ① 패키지 설치하기
- ② [개발자 도구창]에서 리뷰가 적힌 태그 찾기
- ③ HTML가져오기
  - read\_html : 해당 URL의 html을 header와 body로 가져옴
  - html\_nodes : 태그를 찾아주는 함수
  - str\_detect : 수집에 불필요한 단어 제거
- ④ 관심 내용 추출하기
  - SimplePos09 : 문장의 각 형태 분리
  - str\_match : '가-힐' 까지의 단어와 함께 N(명사), P(형용사)만을 가져오도록 함
  - Corpus : 일종의 말뭉치, 데이터 분석을 위한 텍스트 집합.
  - wordLengths : 최대/최소 단어 길이를 지정
  - weightBin : 한 문장에서 동일하게 반복되는 말은 중복 카운트 하지 않음
- ⑤ 데이터를 하나의 자료구조로 바꾸기
  - TermDocumentMatrix : Document별 단어 빈도수를 Maxtrix 형태로 보여줌
- ⑥ 결과 시각화 하기 : qqgraph 명령어로 시각화 하기



## 요점정리



## 용어사전

용어	내용
크롤링(Crawling)	<ul style="list-style-type: none"><li>• 무수히 많은 컴퓨터에 분산 저장되어 있는 문서를 수집하여 검색 대상의 색인으로 포함시키는 기술</li></ul>