웹 크롤링 & 텍스트마이닝

기생충, 숨바꼭질, 알라딘 영화리뷰 분석

CONTENTS

크롤링

| 네이버 영화

| 기생충 리뷰 크롤링

| 숨바꼭질 리뷰 크롤링

| 알라딘 리뷰 크롤링

빈도분석 & 워드크라우드

| 빈도분석

| 워드크라우드

감성분석

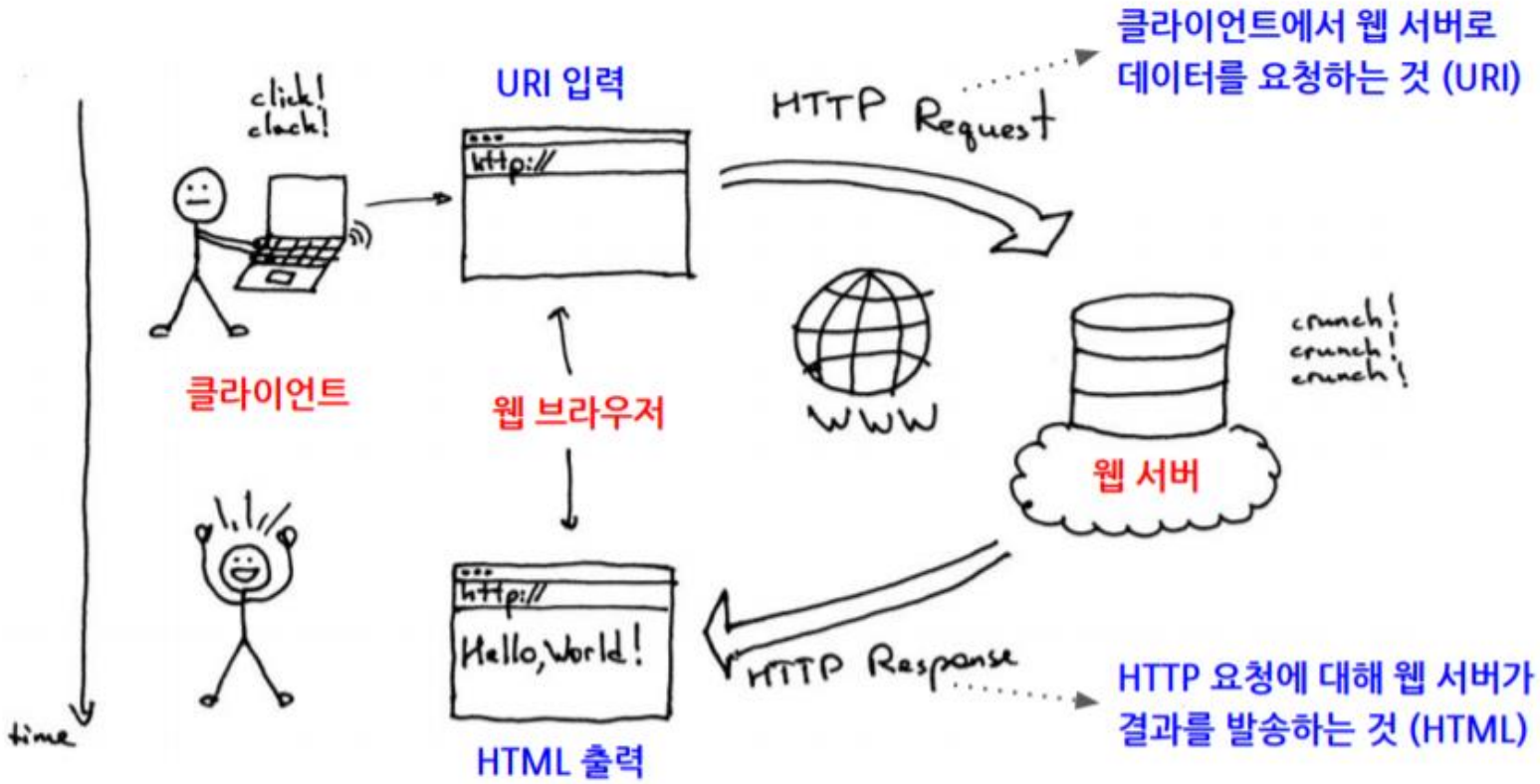
| 긍정 부정 점수 매기기

상관성분석

| 단어 간의 상관성 분석

네이버 웹 크롤링

우리가 인터넷에서 정보를 검색하는 방법



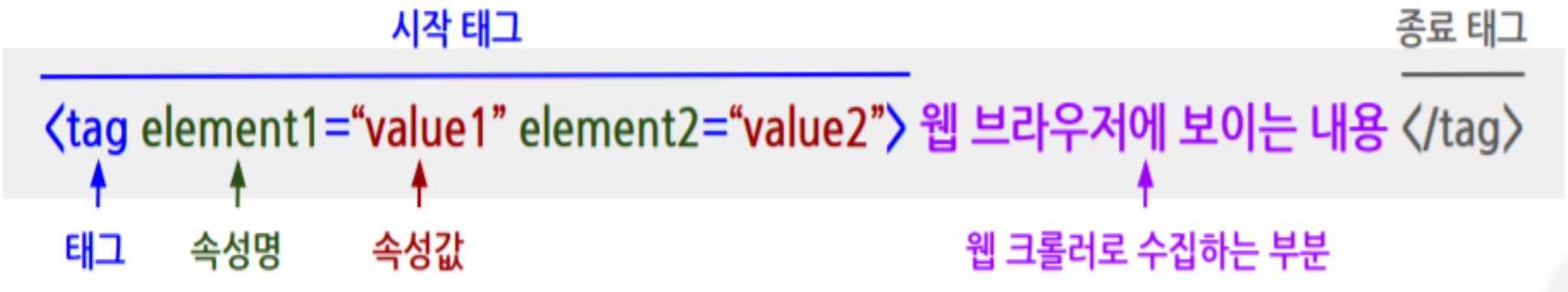
01

Crawling

웹 크롤링은 인터넷 검색과 유사

HTTP Request (요청)	- GET 방식과 POST 방식의 HTTP 통신 - JavaScript 및 RSelenium 이용	httr urltools RSelenium
HTTP Response (응답)	- 응답 결과 확인 (상태코드, 인코딩 방식 등) - 응답 받은 객체를 텍스트로 출력 - 응답 받은 객체에 찾는 HTML 포함 여부 확인	
HTML에서 데이터 추출	- 응답 받은 객체를 HTML으로 변환 - CSS 또는 XPath로 HTML 요소 찾기 - HTML 요소로부터 데이터 추출	rvest jsonlite
데이터 전처리 및 저장	- 텍스트 전처리 (결합, 분리, 추출, 대체) - 다양한 형태로 저장 (RDS, Rdata, xlsx, csv 등)	stringr dplyr

네이버 웹 크롤링



'rvest' package

: R의 웹 스크래핑을 위한 패키지로 Tag Selection, CSS Selecion 등 다양한 기능

1. Download package

```
install.package('rvest')
```

2. Load package

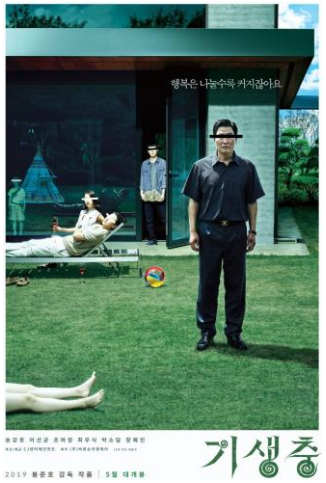
```
library(rvest)
```

01

Crawling

네이버 웹 크롤링

<기생충 & 숨바꼭질 & 알라딘>



01

https://movie.naver.com/movie/point/af/list.nhn?st=mcode&sword=161967

02

https://movie.naver.com/movie/point/af/list.nhn?st=mcode&sword=102824

03

https://movie.naver.com/movie/point/af/list.nhn?st=mcode&sword=163788

URL 저장

복사해뒀던 영화리뷰 url을 변수에 저장!

1. 기생충

   parasite_url <- ' 링크 주소 '

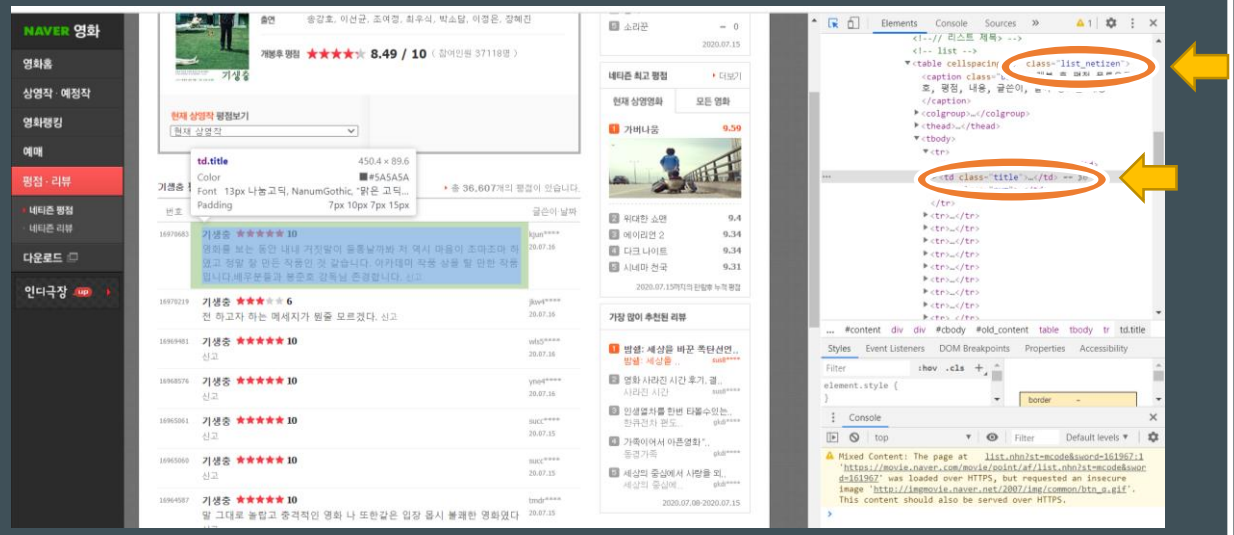
2. 숨바꼭질

   hideandseek_url <- '링크 주소'

3. 알라딘

   aladin_url <- '링크 주소'

리뷰 불러오기



(자세한 사항: https://mrchypark.github.io/getWebR/#26)

1. 리뷰 저장할 변수 설정하기

```
parasite.review <- c()
```

2. 리뷰 데이터 불러오기 (5페이지만)

```
for(page in 1: 5){
url <- paste0(parasite_url,page)
 htxt <- read_html(url, encoding="CP949")
 table <- html_nodes(htxt, '.list_netizen') %>%html_nodes('.title') %>%html_text()

 parasite.review <- c(parasite.review, table)
 print(page)
 }
```

나머지 영화도 동일!

데이터 전처리는 생략

=> 전처리된 명사,형용사 데이터 제공!

(자세한 사항: https://kutar37.tistory.com/entry/R%EC%9D%84%EC%9D%B4%EC%9A%A9%ED%95%9C%ED%85%8D%EC%8A%A4%ED%8A%B8%EB%A7%88%EC%9D%B4%EB%8B%9D%EC%9B%8C%EB%93%9C%ED%81%B4%EB%9D%BC%EC%9A%B0%EB%93%9C)

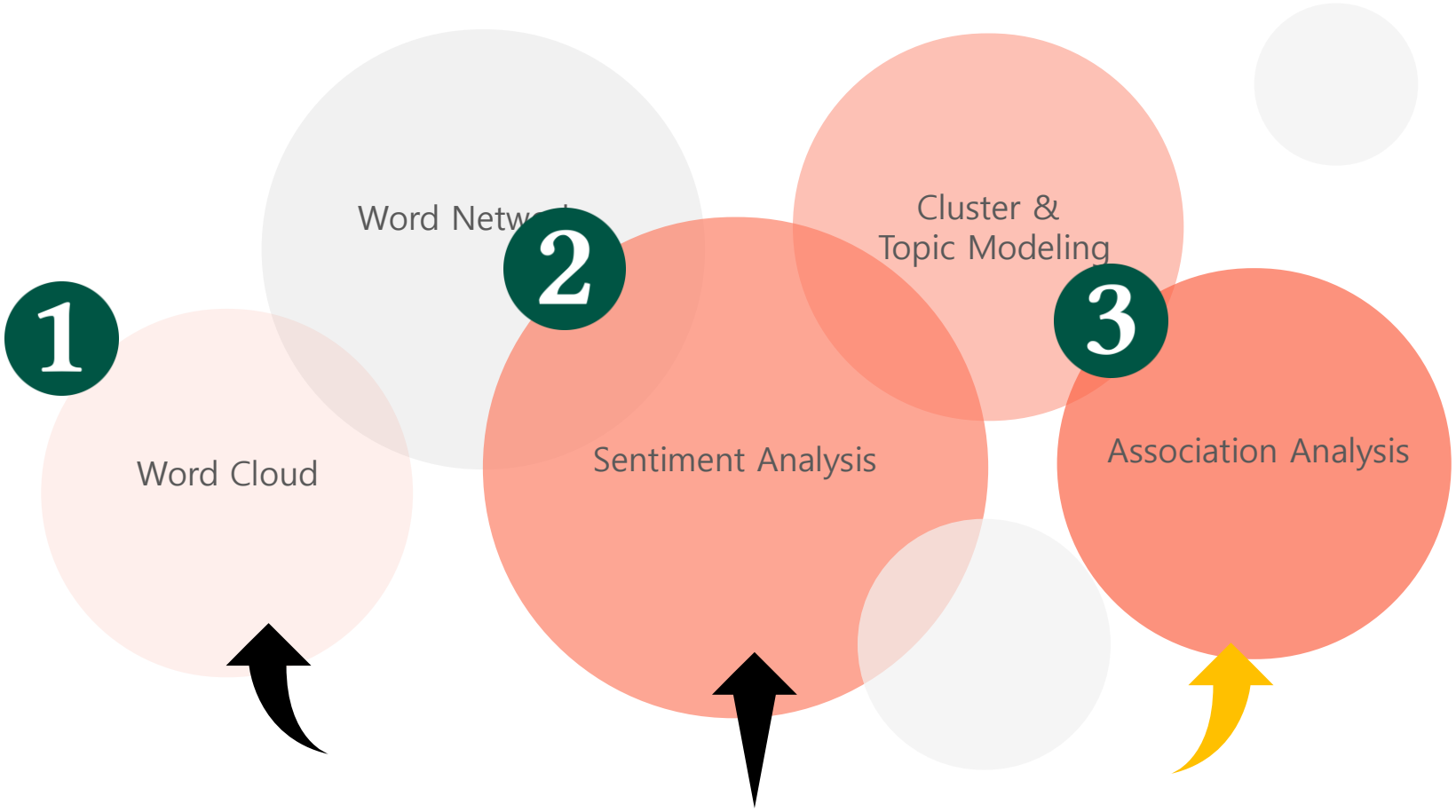
명사,형용사 파일 불러오기

Parasite hideandseek aladin

02

Text-mining

텍스트 마이닝



빈도분석 & 워드크라우드

텍스트마이닝

- 빈도분석 & 워드크라우드

추출된 파일

- **Table로 단어별 빈도수**

  Ex) parasite_wordcount <- table(parasite)

- **빈도수 정렬 -> 데이터프레임으로 저장**

  Ex)
  Parasite_wordcount_top <-
  Sort(parasite_wordcount , decreasing = T)

- **상위 10개 단어 시각화**

  Ggplot을 사용해 상위 10개 단어를 막대 그래프로 나타내기!!

텍스트마이닝

- 빈도분석 & 워드크라우드

추출된 파일

- **패키지 wordcloud, RColorBrewer 다운**

```
library(wordcloud)
library(RColorBrewer)
```

- **원하는 색깔 설정**

```
display.brewer.all()
```

- **워드크라우드 그려보기**

텍스트마이닝

워드크라우드

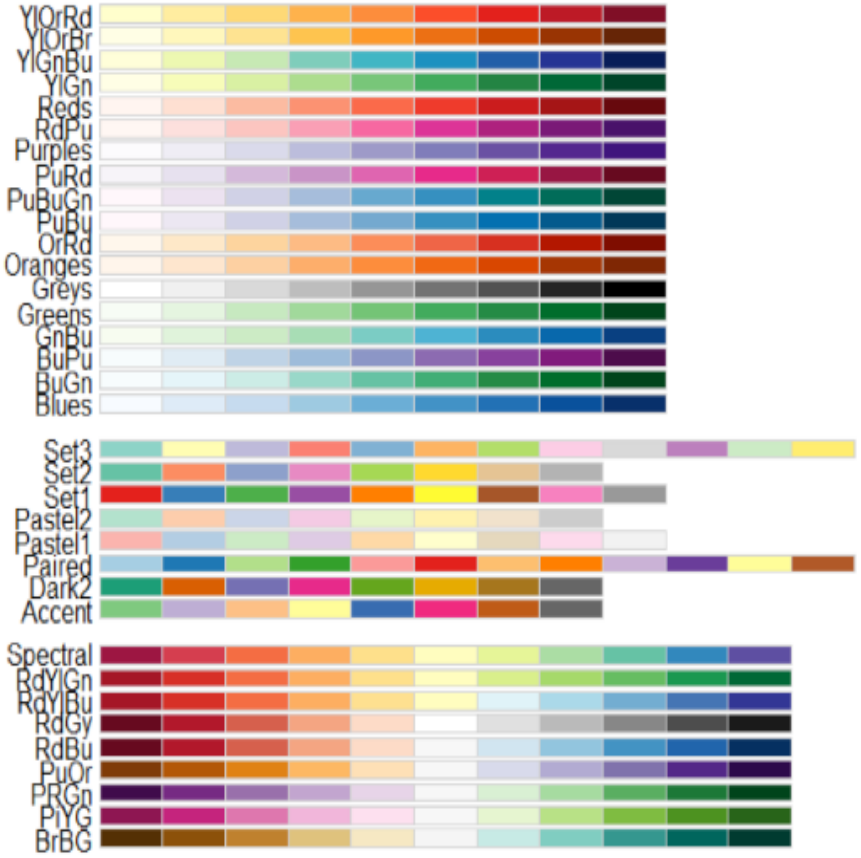
Wordcloud() 함수

wordcloud 옵션

Parameter	설명(explanation)
word	단어(Word)
freq	단어들의 빈도(frequency of words)
size	가장빈도가 큰 단어와 빈도가 가장 작은 단어 폰트 사이의 크기 차이(Size difference between the word with the largest frequency and the word font with the smallest frequency.)
min.freq	출력될 단어의 최소 빈도(Minimum frequency of words to be printed)
max.word	출력될 단어들의 최대개수(Maximum number of words to be printed)
random.order	TRUE 이면 램덤으로 단어출력, FALSE 이면 빈도수가 큰 단어일수록 중앙에 배치(Print words with TRUE-beneath rammed, and center for words with greater frequency behind FALSE)
random.color	TRUE 이면 단어색은 랜덤순으로 정해지고, FALSE이면 빈도순으로 정해짐(The color of words behind TRUE is set in random order, and if FALSE is, they are set in frequency order.)
rot.per	90도로 회전된 각도로 출력되는 단어의 비율(The percentage of words that are output at a rotated angle of 90 degrees.)
colors	가장 작은 빈도부터 큰 빈도까지의 단어색(Word color from smallest frequency to largest frequency)

library(RColorBrewer) 패키지

display.brewer.all(): 색깔 목록



감성 분석

Method 1

1) Left Join 함수를 사용

2) 긍정, 부정 단어 빈도분석

3) 긍정, 부정 전체 지수 시각화

Method 2

'plyr' package

:원본 데이터를 새로운 형태로 만들어주는 패키지

'stringr' package

: 문자열을 쉽게 처리하도록 도와주는 패키지

1. Download package
2. Load package

감정사전 불러오기

긍정, 부정에 해당하는 감정사전

```
positive <- readLines("positive.txt", encoding = "UTF-8")

negative <- readLines("negative.txt", encoding = "UTF-8")
```

텍스트마이닝

감성분석

```
sentimental = function(sentences, positive, negative){

  scores = laply(sentences, function(sentence, positive, negative) {

    sentence = gsub('[[:punct:]]', '', sentence) # 문장부호 제거
    sentence = gsub('[[:cntrl:]]', '', sentence) # 특수문자 제거
    sentence = gsub('\\d+', '', sentence)        # 숫자 제거

    word.list = str_split(sentence, '\\s+')      # 공백 기준으로 단어 생성 ->
    words = unlist(word.list)                    # unlist() : list를 vector

    pos.matches = match(words, positive)           # words의 단어를 positive
    neg.matches = match(words, negative)

    pos.matches = !is.na(pos.matches)            # NA 제거, 위치(숫자)만 추출
    neg.matches = !is.na(neg.matches)

    score = sum(pos.matches) - sum(neg.matches)  # 긍정 - 부정
    return(score)
  }, positive, negative)

  scores.df = data.frame(score=scores, text=sentences)
  return(scores.df)
}
```

3개 영화
모두에 적용!

텍스트마이닝

긍정, 부정 결과 계산하기

```
1 result$remark[result$score >=1] = "긍정"
2 result$remark[result$score ==0] = "중립"
3 result$remark[result$score < 0] = "부정"
```

이때 변수명을
Parasite_senti / hideandseek_senti / aladin_senti로 설정

텍스트마이닝

긍정, 부정 결과 계산하기

```
parasite_sentiment_result= table(parasite_senti$remark)
parasite_sentiment_result <- as.data.frame(parasite_sentiment_result)
```

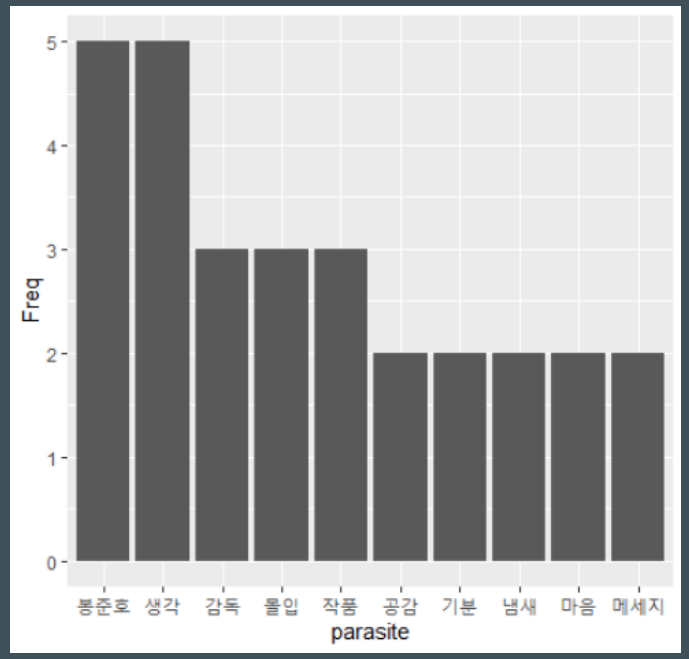
ggplot으로 Pie 차트 그려보기!!

```
ggplot(parasite_sentiment_result, aes(x = "",y = Freq, fill = Var1)) +
  geom_bar(stat = 'identity') + coord_polar("y")
```

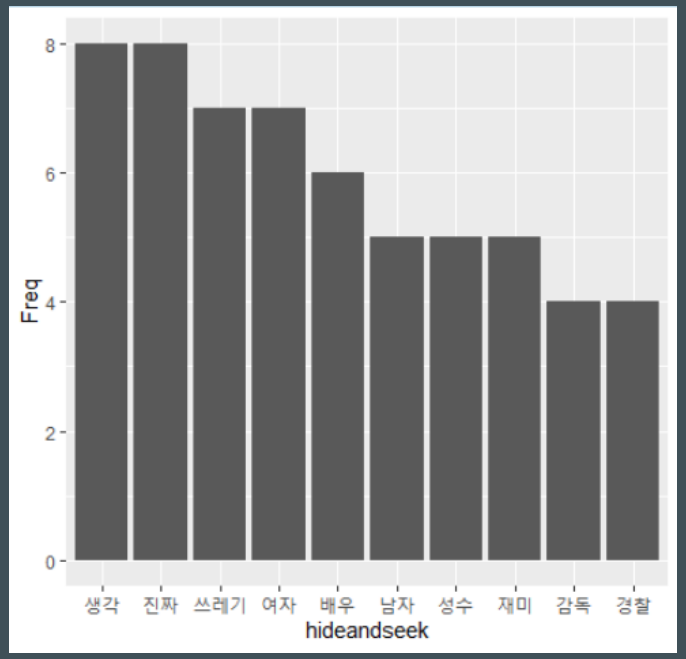
상관성 분석

빈도분석 비교

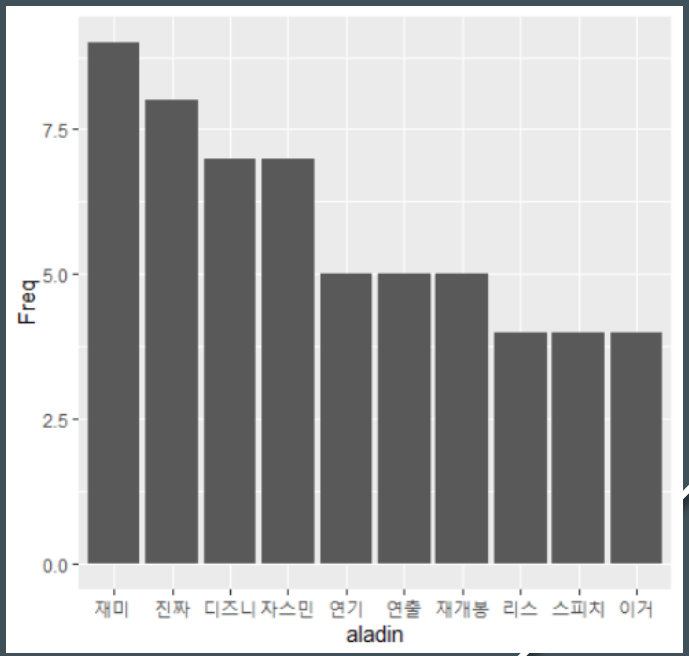
1) 기생충



2) 숨바꼭질



2) 알라딘

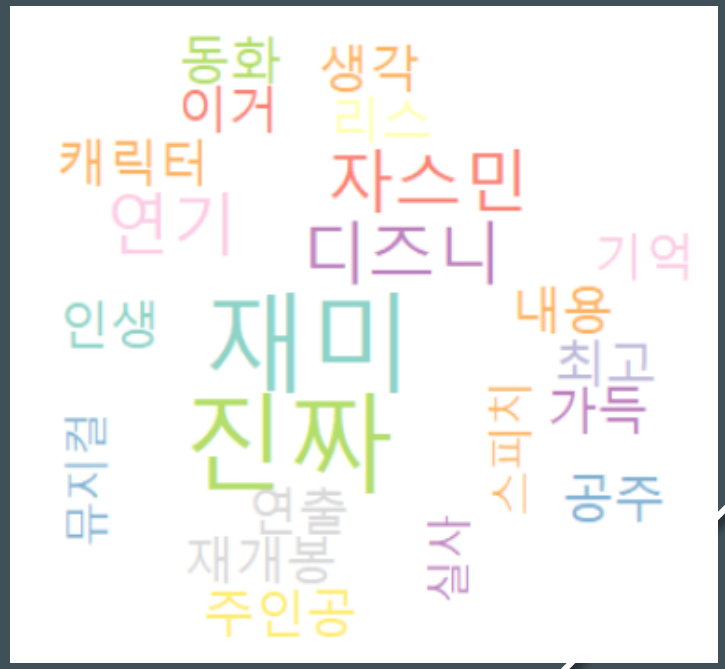


워드크라우드 비교

1) 기생충

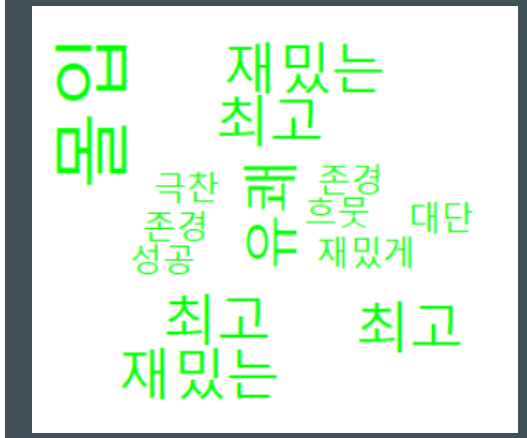
2) 숨바꼭질

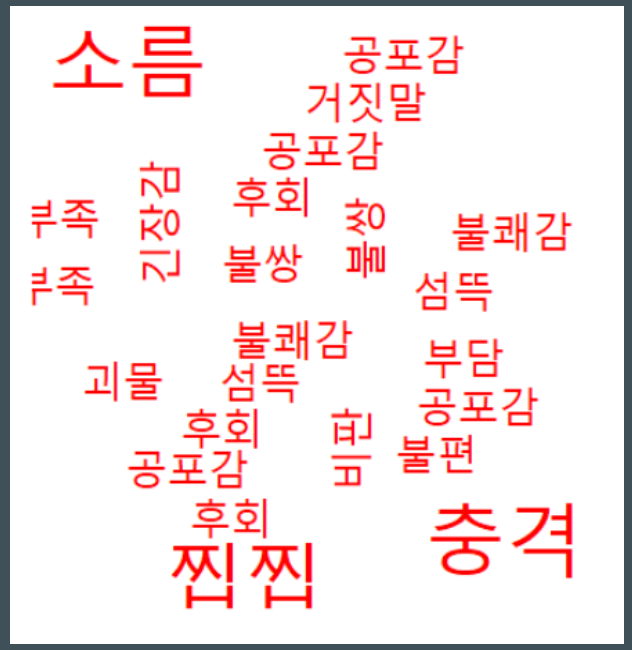
2) 알라딘

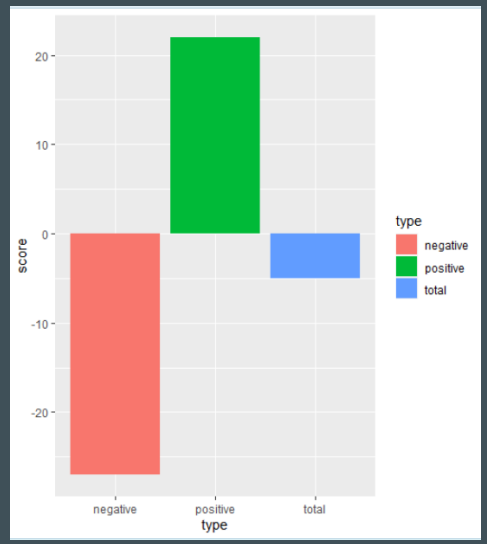


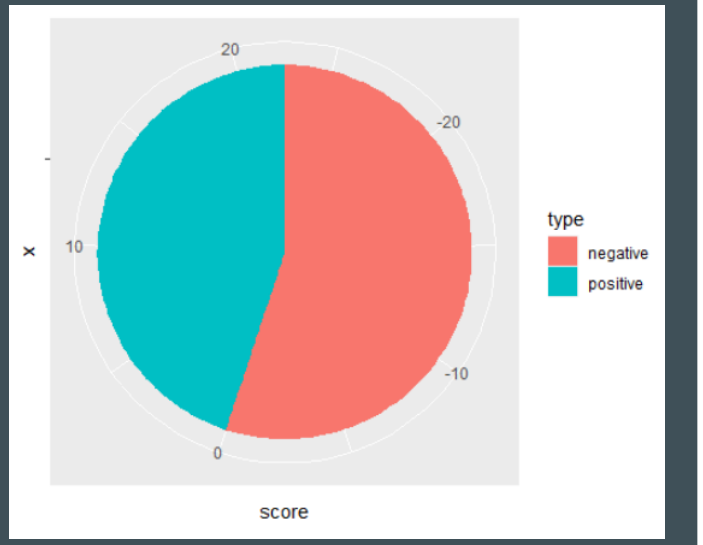
감성분석 비교

1) 기생충



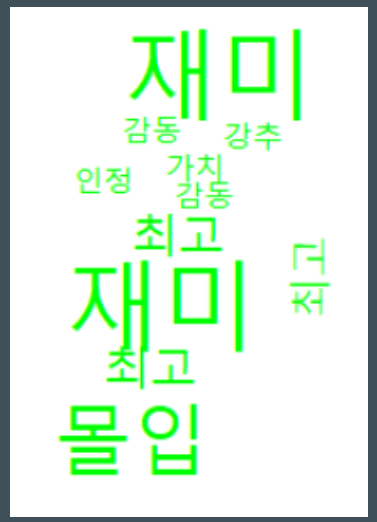


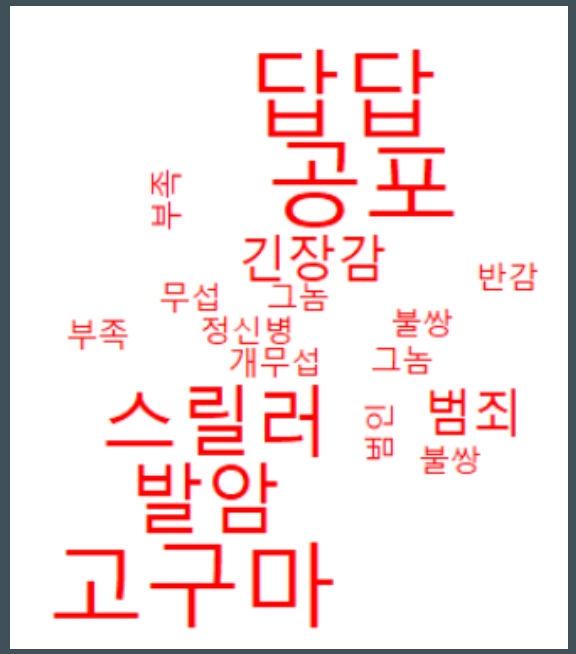


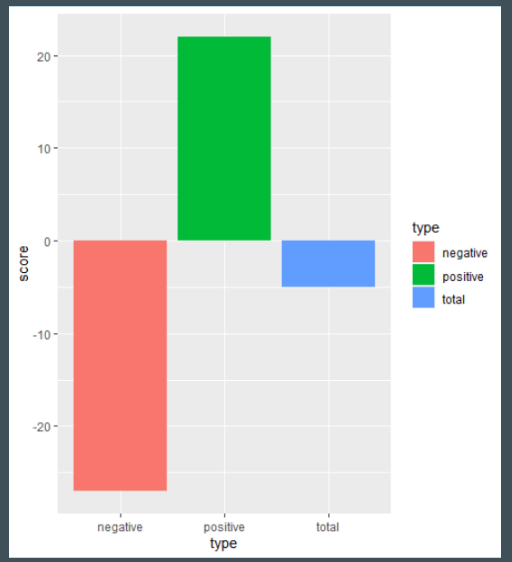


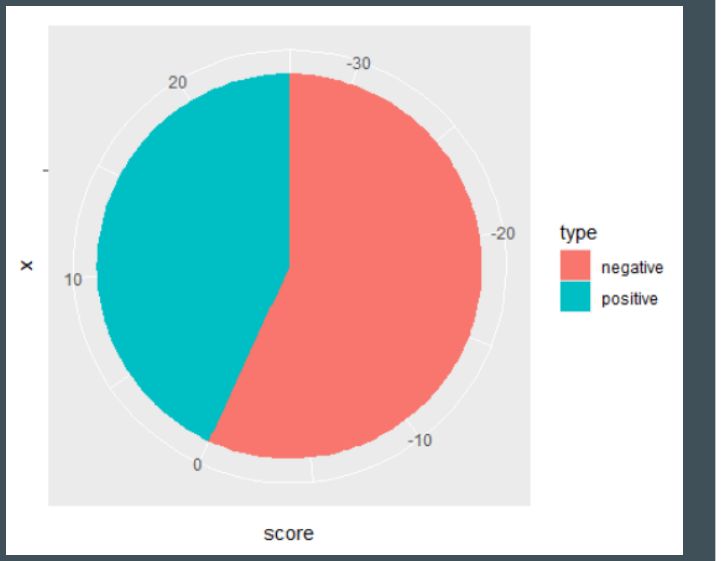
감성분석 비교

2) 숨바꼭질



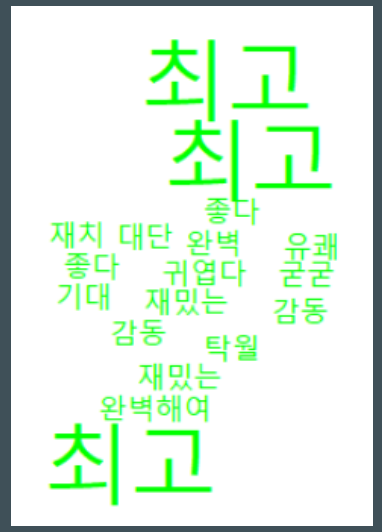


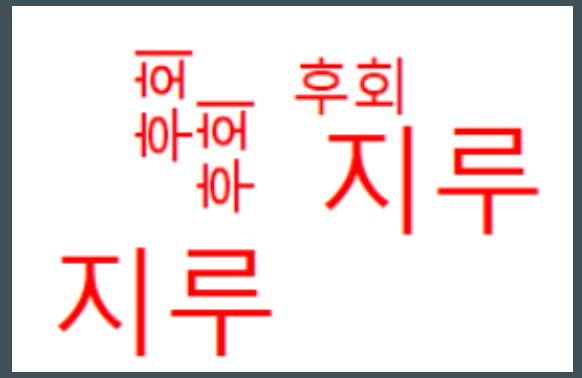


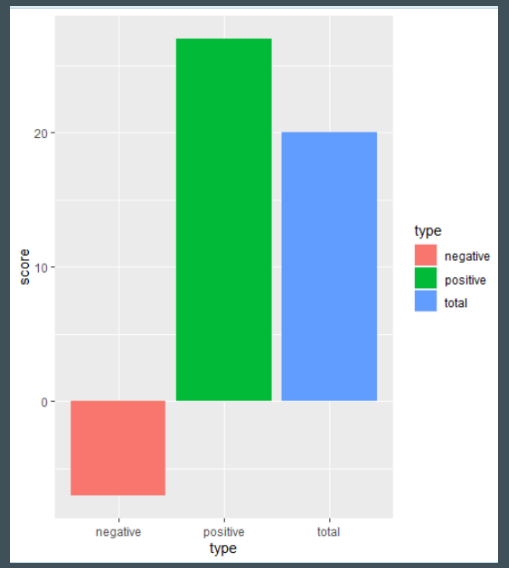


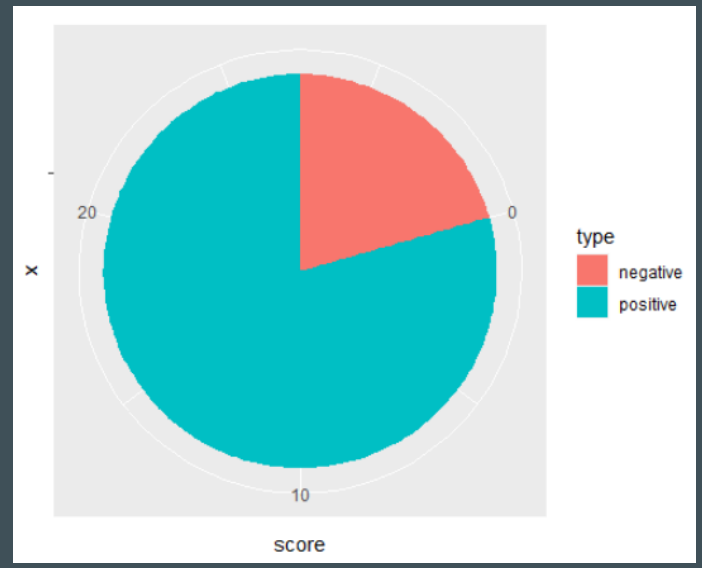
감성분석 비교

3) 알라딘



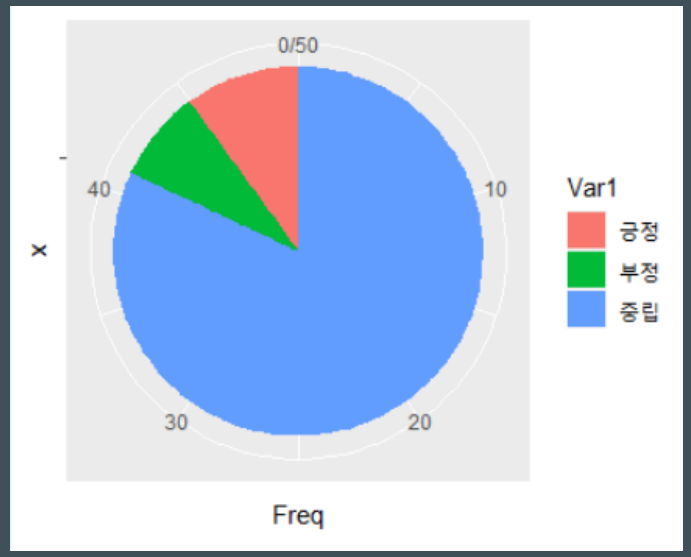




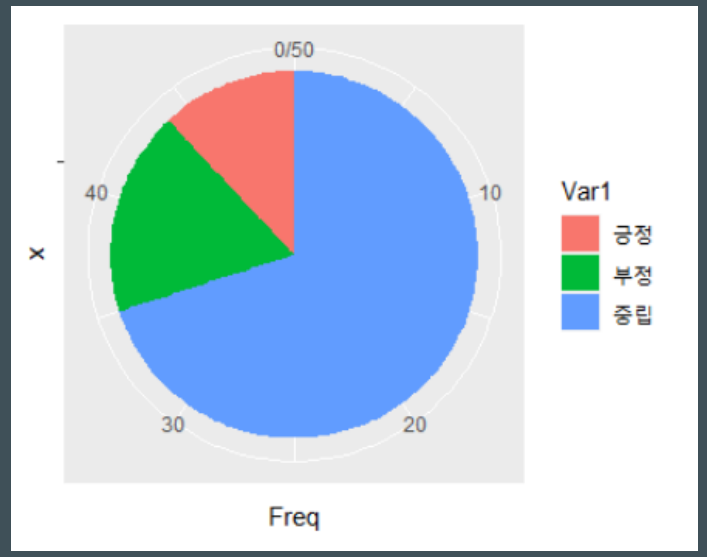


감성분석 비교

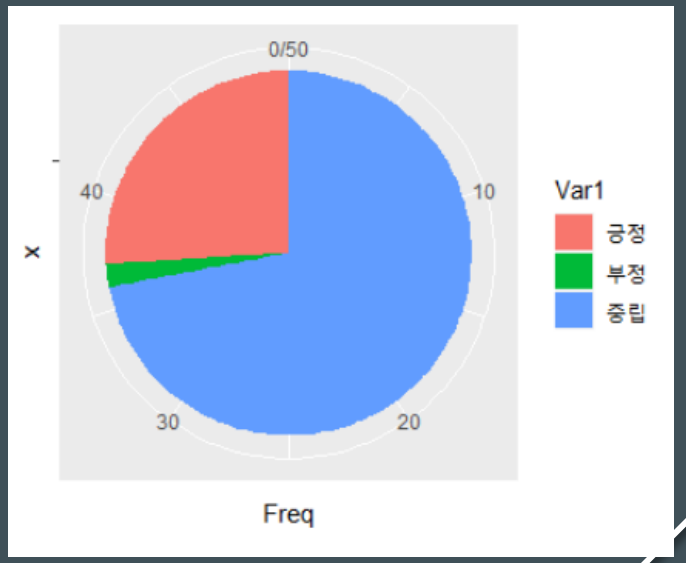
1) 기생충



2) 숨바꼭질



2) 알라딘



Thank you