



Web Crawling & Word Cloud

네이버 140자 평론 댓글을 모아서 Word Cloud 만든 후 에 분석하기

한국교통대학교 소프트웨어전공 18학번 이철현

INDEX

1

도대체 어떻게 할 생각인데?

2

준비하기

3

PyCharm 으로 데이터 모으기

4

R Studio 로 시각화 하기

1

도대체 어떻게 할 생각인데?

1. 도대체 어떻게 할 생각인데?

데이터를 모으고자 하는 사이트를 선택한다.

PyCharm 으로 사이트속 데이터를 모은다.

R Studio 로 데이터를 시각화 한다.

1. 도대체 어떻게 할 생각인데?

선택하기 (Chose)

모으기 (Crawling)

시각화하기 (Visualization)

2

준비하기

2. 준비하기

Python

IDE : PyCharm

R x64,32

IDE : R Studio

+ 각종 패키지 와 라이브러리

3

PyCharm 으로 데이터 모으기

3. PyCharm 으로 데이터 모으기 (Crawling)

우리가 사용 할 파이썬 라이브러리는 총 2가지

requests

HTML 소스코드
불러오기
라이브러리

Beautiful Soup 4

불러온
HTML 소스코드
중에서
필요한 데이터만
추출해내기

3. PyCharm 으로 데이터 모으기 (Crawling)

라이브러리 설치과정

1. 먼저 파이썬과 파이참을 설치한다. (파이썬 -> 파이참)
2. 파이참을 실행한뒤
3. File -> Setting(Ctrl + Alt + s) -> Project : 프로젝트명(version control 밑에) -> Project Interpreter
4. 오른쪽 화면을 보면 + 모양이 있다. 그것을 누른다.
5. 위에 검색창에 다음 2가지를 입력한 뒤에 왼쪽 아래에 Install Package 를 누른다.
 1. request
 2. BeautifulSoup 4
6. 설치가 완료되면 그걸로 라이브러리 설치는 마무리 한다.

3. PyCharm 으로 데이터 모으기 (Crawling)

requests

Python 에서 HTTP 요청을 보내는 라이브러리

[HTTP는 (Hyper Text Transfer Protocol) WWW 상에서 정보를 주고받을 수 있는 프로토콜이다.]

Beautiful Soup 4

request 라이브러리 를 통해서 Crawling 해온 소스코드 중
“내가 원하는 데이터만 뽑아내는 것(Parsing)”을
도와주는 소스코드 파싱 라이브러리 이다

3. PyCharm 으로 데이터 모으기 (Crawling)

requests

Python 에서 HTTP 요청을 보내는 라이브러리

[HTTP는 (Hyper Text Transfer Protocol) WWW 상에서 정보를 주고받을 수 있는 프로토콜이다.]

3. PyCharm 으로 데이터 모으기 (Crawling)

1.요청을 저장할 변수 = request.get(URL)

```
import requests

connect = requests.get('http://www.google.com')
print(connect)
```

- 해당 URL의 서버와의 HTTP요청 허가를 가져온다
- 출력하면 <Response [200]>을 출력한다.
- 위의 소스코드를 보면 connect라는 변수에다가 요청허가를 저장해 두었다.
- 이후 "connect.~~~" 를 해석할 때는 "~(을)를 HTTP에 요청합니다" 로 해석하면 이해하기 쉽다.

3. PyCharm 으로 데이터 모으기 (Crawling)

2. 소스코드를 저장할 변수 = connect.text

```
html = connect.text  
print(html)
```

- 받아온 URL의 HTML소스코드를 HTTP에게 요청하는 함수이다.
- 정상적으로 작동한다면 크롬에서 "소스보기"를 했을 때와 같은 결과 값이 나온다.
- 위 소스코드를 보면 html이라는 변수에다가 받아온 URL의 HTML소스코드 를 txt형태로 저장했다.
- 이때 type(html)을 해보면 string 타입으로 저장 됨을 알 수 있다.
- "HTML로 된 소스코드 text타입 데이터를 HTTP에 요청합니다"

3. PyCharm 으로 데이터 모으기

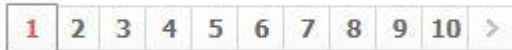
중간 점검

```
1 import requests
2
3 connect = requests.get('http://www.google.com')
4 print(connect)
5
6 html = connect.text
7 print(html)
```

3. PyCharm 으로 데이터 모으기 (Crawling)

3. 내가 Crawling 할 사이트 정하기 (1)

1. 먼저 네이버 영화페이지 에 들어가서 자신이 원하는 영화를 검색한다.
2. 검색한 영화페이지 에서 평점을 클릭한다
3. 쪽 내려서 1~10번까지 넘길 수 있는 배너를 오른쪽 클릭하고 검사(Ctrl + shift + i)를 누른후에
4. 나오는 여러가지 페이지 중에서 하나를 클릭한다.
5. 그다음 첫번째 페이지로 이동한다.



<< 바로 이 배너 오른쪽 클릭하고 검사

3. PyCharm 으로 데이터 모으기 (Crawling)

3. 내가 Crawling 할 사이트 정하기 (2)

네이버 영화 평론 페이지는 다음과 같은 URL 구성으로 되어있다!

```
https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?  
code=163608&type=after&onlyActualPointYn=Y&page=1
```

이때 우리가 주목 해야할 것은 다음과 같다.

```
?code=163608  
  
&page=1
```

code 부분을 보면 영화는 영화마다 code를 갖는다 따라서 내가 선택한 영화(돈, 아이언맨3)는 다음의 코드르 가져 졌다

- 아이언맨3 = 70254
- 돈 = 163608

그리고 우리가 무조건 봐야하는!!! 가장 중요한 것은 바로 저 **&page=1**부분 이다.

저부분은 평점 페이지가다음으로 넘어갈 때마다 1씩 증가한다. 따라서 우리가 페이지 1의 평론부터

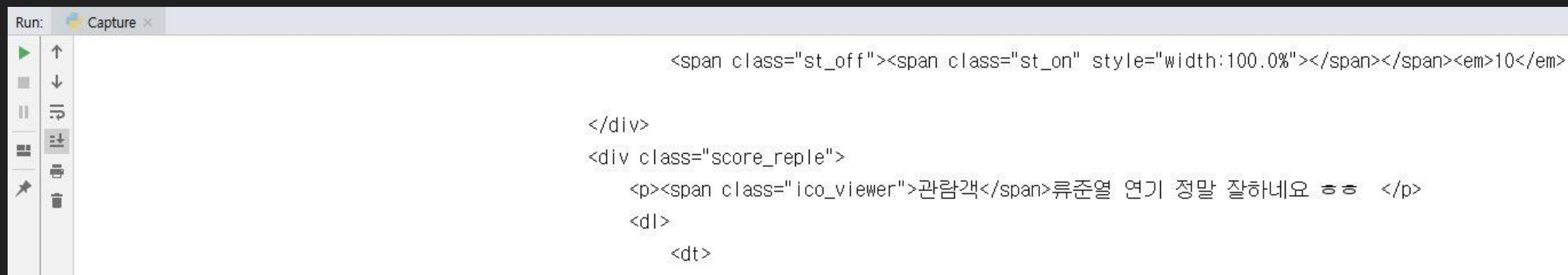
가장 마지막 페이지의 평론까지 모두 크롤링 하려면 이부분을 for문을 통해서 1부터 끝까지 반복해주면 된다.

3. PyCharm 으로 데이터 모으기

Request 후

```
1 import requests
2
3 URL = 'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=163608&type=after&onlyActualPointYn=Y&page=1'
4
5 connect = requests.get(URL)
6
7 html = connect.text
8 print(html)
```

결과 중... 중간



```
Run: Capture x
<span class="st_off"><span class="st_on" style="width:100.0%"></span></span><em>10</em>
</div>
<div class="score_reple">
  <p><span class="ico_viewer">관람객</span>류준열 연기 정말 잘하네요 ㅎㅎ </p>
  <dl>
    <dt>
```

3. PyCharm 으로 데이터 모으기 (Crawling)

Beautiful Soup 4

request 라이브러리 를 통해서 Crawling 해온 소스코드 중
“내가 원하는 데이터만 뽑아내는 것(Parsing)”을
도와주는 소스코드 파싱 라이브러리 이다

3. PyCharm 으로 데이터 모으기 (Crawling)

1. 객체용 변수 = BeautifulSoup(소스코드가 저장된 변수 , 'html.parser')

```
from bs4 import BeautifulSoup  
  
soup = BeautifulSoup(html, 'html.parser')
```

- 위 작업은 "소스코드가 저장된 변수" 로부터 소스코드를 받아서 BeautifulSoup의 함수들을 사용하여 편집할 수 있는 하나의 객체로 전환 시켜주는 작업이다.
- 또한 html.parser이란 python 내부에 있는 html을 건드릴 수 있는 함수로 이부분을 뒤로 내가 선언한 변수를 통해서 html을 터치할 수 있다.

Q) 왜 import 가 아니라 From ~ Import 인가요!

A) From 라이브러리 import 기능

3. PyCharm 으로 데이터 모으기 (Crawling)

2. 선택된 것들을 모을 변수 = 객체용 변수.select(' CSS selector ')

```
soup = BeautifulSoup(html, 'html.parser')

sentence_array = soup.select('')

print(sentence_array)
```

- 위 작업은 앞서 선언해준 객체용 변수"소스코드가 저장된 변수"에서!! select라는 기능을 사용할 수 있도록 허가를 하는 것이다!
- select기능 : 뒤에 있는 CSS selector을 갖는 html 소스코드를 추출해준다.
- 만약 예를 들어 body > div > p 를 적으면 소스코드 전체중에 저 스타일을 갖는 소스코드를 <p>~</p>까지 추출해준다

3. PyCharm 으로 데이터 모으기 (Crawling)

중간 점검

```
soup = BeautifulSoup(html, 'html.parser')

sentence_array = soup.select('')

print(sentence_array)
```

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 # request 부분
5 URL = 'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=163608&type=after&onlyActualPointYn=Y&page=1'
6
7 connect = requests.get(URL)
8
9 html = connect.text
10
11 # BeautifulSoup 부분
12 soup = BeautifulSoup(html, 'html.parser')
13
14 sentence_array = soup.select('')
15
16 print(sentence_array)
```


3. PyCharm 으로 데이터 모으기 (Crawling)

3. CSS Selector 뽑아내기

CSS selector 뽑아내는 방법

1. 크롬을 켜다
2. 원하는 페이지를 오른쪽 클릭해서 검사(Ctrl + shift + i)를 누른다
3. 원하는 부분을 오른쪽 클릭해서 Copy -> Copy Selector
4. 네이버 로고의 경우 다음의 CSS selector을 갖는다.
 1. #PM_ID_ct > div.header > div.special_bg > div > div.area_logo > h1 > a > span
5. 이때 . 뒤에 나온 모든것들은 지우고 기본적인 태그형태만 남긴다.
 1. #PM_ID_ct > div > div > div > div > h1 > a > span

영화 평론의 경우 : ' body > div > div > div > ul > li > div > p '

3. PyCharm 으로 데이터 모으기 (Crawling)

3. CSS Selector 까지 대입 한 후의 소스코드

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 # request 부분
5 URL = 'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=163608&type=after&onlyActualPointYn=Y&page=1'
6
7 connect = requests.get(URL)
8
9 html = connect.text
10
11 # BeautifulSoup 부분
12 soup = BeautifulSoup(html, 'html.parser')
13
14 sentence_array = soup.select('body > div > div > div > ul > li > div > p')
15
16 print(sentence_array)
```

결과

```
▶ ↑ C:\Users\ChoRong\PycharmProjects\Study\venv\Scripts\python.exe C:/Users/ChoRong/PycharmProjects/Study/Capture.py
■ ↓ [<p><span class="ico_viewer">관람객</span>재미, 속도감, 서스펜스 삼박자 골고루 갖춘 잘만든 오락 영화. 돈에 대해서 많
|| ↳
☐ ↳
➤ ↳
↳ Process finished with exit code 0
```


3. PyCharm 으로 데이터 모으기 (Crawling)

4. 원하는 문장만 추출

ABCDEF안녕하세요HIJK

```
def get_sentence(data):  
    text = str(data)  
  
    start = text.find("ABCDEF") + len("ABCDEF")  
    end = text.find("HIJK")  
  
    sentence = text[start:end]  
    return sentence  
  
print(get_sentence("ABCDEF안녕하세요HIJK"))
```

<p>관람객꿀잼ㅠㅠㅠ미쳤어요ㅠㅠㅠ류준열짱!! </p>

```
def get_sentence(data):  
    text = str(data)  
  
    start = text.find("</span>") + len("</span>")  
    end = text.find("</p>")  
  
    sentence = text[start:end]  
    return sentence  
  
print(get_sentence(sentence_array[0]))
```

3. PyCharm 으로 데이터 모으기 (Crawling)

4. Sentence_array 에서 10개의 데이터를 모두 원하는 문장으로 바꿔주기

```
# 결과물
data_array = []

...

for i in range(len(sentence_array)):
    data_array.append(get_sentence(sentence_array[i]))

print(data_array)
```

결과물 배열을 하나 만들어 준 다음 간단한 for 반복문을 통해서 10개의 배열 요소에 저장된 파싱 되지않은 문장들을 각각 파싱한 후에 만들어 둔 결과물 배열에 차곡차곡 쌓아준다.

3. PyCharm 으로 데이터 모으기 (Crawling)

5. 1페이지에 있는 10개의 평점 글 데이터 모아서 하나의 문장으로 만들기

```
text = ""  
  
for i in range (len(data_array)):  
    text = text + " " + data_array[i]  
  
print(text)
```

3. PyCharm 으로 데이터 모으기 (Crawling)

중간점검

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 def get_sentence(data):
5     text = str(data)
6
7     start = text.find("</span>") + len("</span>")
8     end = text.find("</p>")
9
10    sentence = text[start:end]
11    return sentence
```

```
13 # 결과물
14 data_array = []
15
16 # request 부분
17 URL = 'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=163608&type=after&onlyActualPointYn=Y&page=1'
18
19 connect = requests.get(URL)
20
21 html = connect.text
22
23 # BeautifulSoup 부분
24 soup = BeautifulSoup(html, 'html.parser')
25
26 sentence_array = soup.select('body > div > div > div > ul > li > div > p')
27
28 # 하나의 결과물 배열로 모으기
29 for i in range(len(sentence_array)):
30     data_array.append(get_sentence(sentence_array[i]))
31
32 print(data_array)
33
34 # 하나의 문장으로 모으기
35 text = ""
36
37 for i in range(len(data_array)):
38     text = text + " " + data_array[i]
39
40 print(text)
```

3. PyCharm 으로 데이터 모으기 (Crawling)

6. 9개의 페이지 크롤링 하기 (for 단순 반복문)

```
for page in range(1, 10):  
    # request 부분  
    URL = 'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=163608&type=after&onlyActualPointYn=Y&page='+str(page)
```



3. PyCharm 으로 데이터 모으기 (Crawling)

최종 소스코드 및 결과 처리법

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 def get_sentence(data):
5     text = str(data)
6
7     start = text.find("</span>") + len("</span>")
8     end = text.find("</p>")
9
10    sentence = text[start:end]
11    return sentence
```

최종 결과물은 원하는
폴더에 메모장 txt파일을
만들어 복사 붙여넣기 한다.

```
13 # 결과물
14 data_array = []
15
16 for page in range(1, 10):
17     # request 부분
18     URL = 'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=163608&type=after&onlyActualPointYn=Y&page='+str(page)
19
20     connect = requests.get(URL)
21
22     html = connect.text
23
24     # BeautifulSoup 부분
25     soup = BeautifulSoup(html, 'html.parser')
26
27     sentence_array = soup.select('body > div > div > div > ul > li > div > p')
28
29     # 하나의 결과물 배열로 모으기
30     for i in range(len(sentence_array)):
31         data_array.append(get_sentence(sentence_array[i]))
32
33
34     # 하나의 문장으로 모으기
35     text = ""
36
37     for i in range(len(data_array)):
38         text = text + " " + data_array[i]
39
40     print(text)
```

4

R Studio 로 시각화 하기

4. R Studio 로 시각화 하기 (Visualizations)

1. 1. R studio 패키지 설치 및 사용준비

```
# wordcloud를 위한 패키지들 import
install.packages("stringr")
install.packages("ggplot2")
install.packages("SnowballC")
install.packages("wordcloud")
install.packages("RColorBrewer")
install.packages("plyr")
```

1. 3. 패키지 사용

```
# import 한 패키지들 사용
library("stringr")
library("ggplot2")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
library("plyr")
```

1. 2. KoNLP 설치 및 사전 생성

KoNLP란?

-> 한글 자연어 분석 패키지

```
# 가장 중요한 한글 체킹 패키지
install.packages("KoNLP")
install.packages("rJava")
Sys.setenv(JAVA_HOME='C:\\Program Files\\Java\\jre1.8.0_201')
library("KoNLP")
useSejongDic()
```

```
# 사용자미름 에 해당하는 data.frame의 사전을 하나 생성한다.
mergeUserDic(data.frame("사전미름", "ncn"))
```


4. R Studio 로 시각화 하기 (Visualizations)

2. Working Directory 를 설정하고 앞서 저장해둔 txt파일 불러오기

```
setwd("C:\\Users\\ChoRong\\Desktop\\Base\\04. 개인 연구\\[4]web_crawling")  
# Textfile 가져오기  
text <- readLines("review.txt")
```

4. R Studio 로 시각화 하기 (Visualizations)

3. Txt 파일안에서 단어 쪼개고 쓸모없는 단어 삭제하기

```
# 명사 단위로 쪼개기
text <- sapply(text, extractNoun, USE.NAMES = F)

# 리스트 해제 하고 2개의 길이만 허용하는 필터 = [1차원 형태로 만들기]
text <- unlist(text)
text <- Filter(function(x) {nchar(x) >= 2}, text)

# 필요없는 단어 삭제하기
text <- str_replace_all(text, "[^[:alpha:]]", "") # 특수문자
text <- str_replace_all(text, "[A-Za-z0-9]", "") # 숫자
text <- gsub("을", "", text)
text <- gsub("를", "", text)
text <- gsub("ㅋ", "", text)
text <- gsub("ㅠ", "", text)
text <- gsub("ㅇ", "", text)
text <- gsub("영화", "", text)
```

4. R Studio 로 시각화 하기 (Visualizations)

4. 공백 과같은 것들을 없애주기 위해서 임시파일로 만들어내고, 테이블 형태로 만들어서 최대 빈도로 나온 값을 정렬 해주는 것

```
# 임시적인 파일을 만들어서 테이블 형태로 불러오기.  
write(unlist(text) , "kr_text.txt")  
text_table <- read.table("kr_text.txt")
```

```
# 테이블 형태로 데이터 만들기  
word_count <- table(text_table)
```

```
# dataframe 만들기  
terms <- data.frame(word_count)
```

```
# 열 이름 바꿔주기  
names(terms) <- c("word", "freq")
```

```
# 정렬 해 주기  
terms <- arrange(terms, desc(freq))
```

```
terms
```

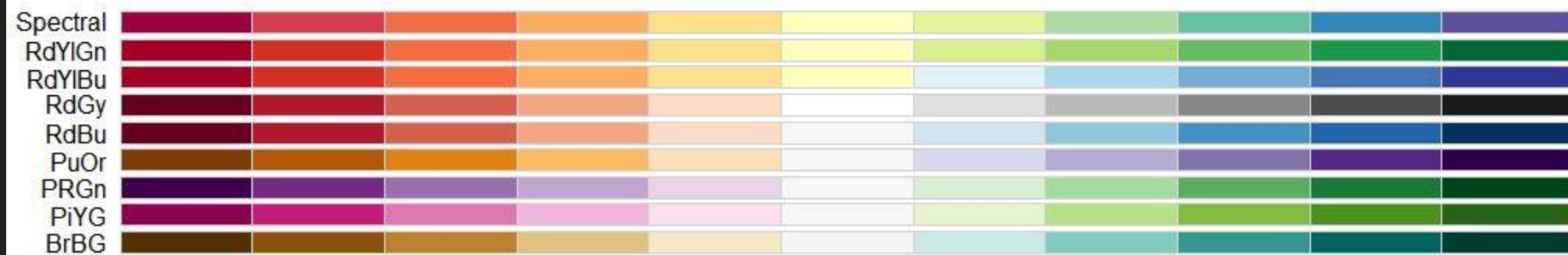
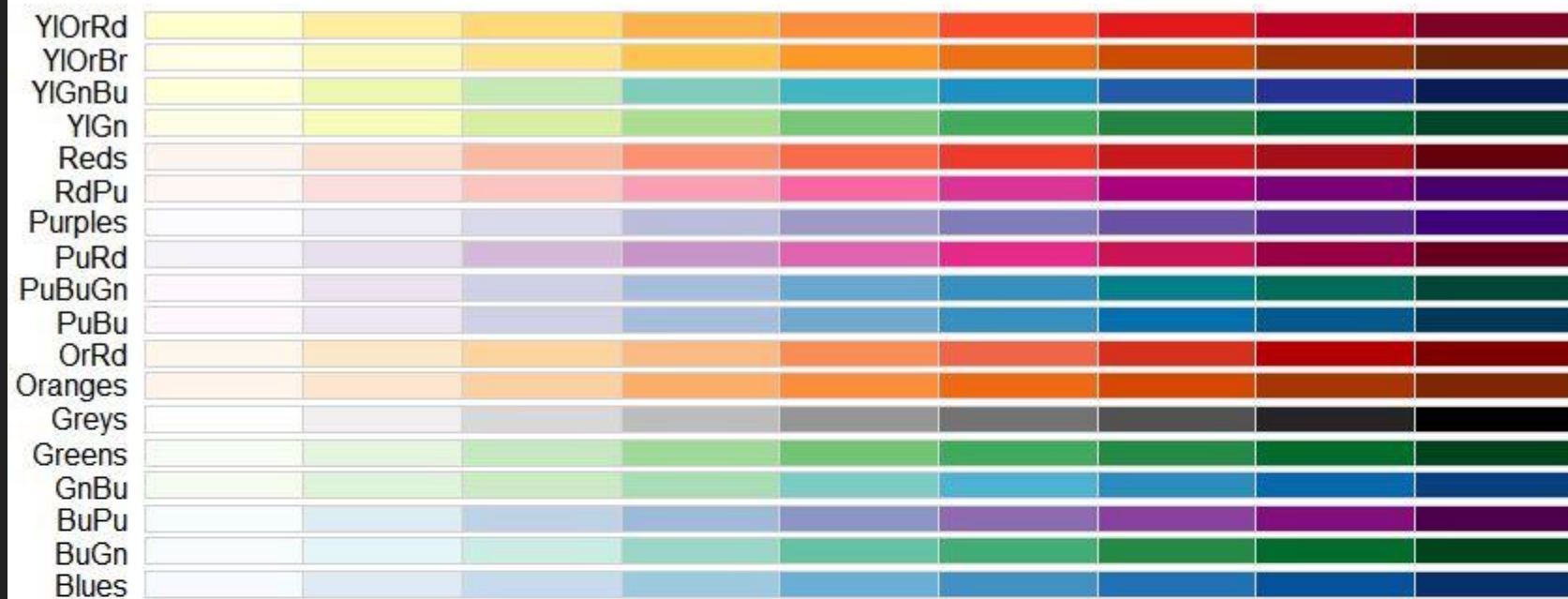
4. R Studio 로 시각화 하기 (Visualizations)

5. WordCloud 만들기

```
wordcloud(words=terms$word, freq=terms$freq, random.order = F, colors=brewer.pal(12,"Paired"))
```

wordcloud(단어선택, 기준, 랜덤하게?, 워드클라우드 색상)

(기준 번호, 색상코드)



4. R Studio 로 시각화 하기 (Visualizations)



THANK YOU!!!