


## Chapter9. 워드클라우드



---



## 목차

---



주제!

**I. 워드클라우드&패키지설치**

**II. 고객만족도 데이터 워드클라우드 시각화**

**III. 대국민담화문 데이터 워드클라우드 시각화**

**IV. 인터넷 검색어 분석**

**V. 워드클라우드응용**

## 소스1. readLines() & extractNoun()

readLines() 함수를 이용해 문장을 불러오고 extractNoun() 함수를 이용하여 명사 추출

```
# 한글이 불러오고 깨지기 때문에 encoding 옵션에 UTF-8 지정  
txt <- readLines("ch8.txt", encoding = "UTF-8")
```

ch8.txt - Windows 메모장  
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)  
닭이 너무 맛있어요 최고!! 육질이 살아있음  
배송도 빠르고 상품도 좋습니다. ^^

txt

"닭이 너무 맛있어요 최고!! 육질이 살아있음"

```
# txt에서 명사만 뽑아서 n에 저장  
n <- extractNoun(txt)
```

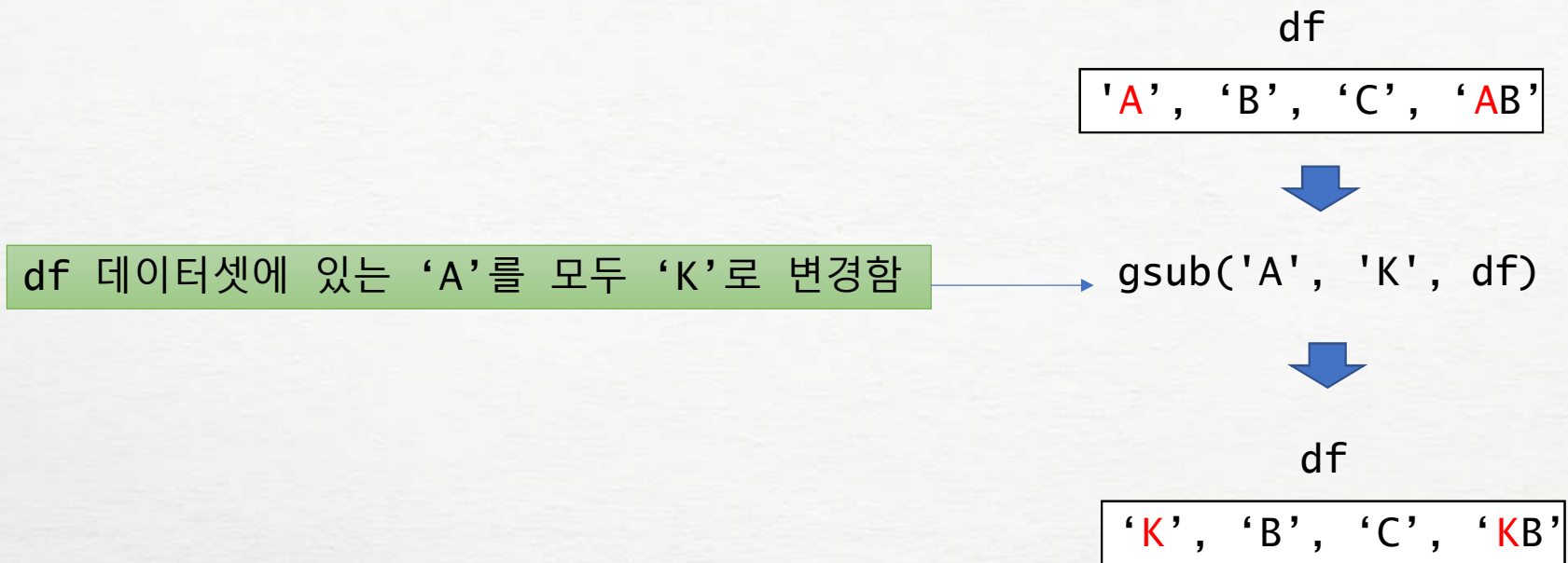
n-명사

"닭" "최고" "육" "질" "살아있" "음"

## 소스2. gsub() 함수

### ▣ 텍스트 수정

gsub() 함수를 이용해 텍스트 수정



### 소스3. unlist()

벡터, 행렬, 데이터프레임을 하나씩 정의하여 list를 벡터로 만들기

```
v=c(1,2,3)
m=matrix(1:4,2)
df=data.frame(a=c(10,20,30),b=c(40,50,60))
```

```
> m1=list(v,m,df)
> m1
[[1]]          #벡터 (v)
[1] 1 2 3
[[2]]          #매트릭스 (m)
[,1] [,2]
[1,] 1 3
[2,] 2 4
[[3]]          #데이터프레임 (df)
  a b
1 10 40
2 20 50
3 30 60
```

```
➤ unlist(m1) # 리스트에 unlist 함수를 적용
      a1 a2 a3 b1 b2 b3
```

```
1 2 3 1 2 3 4 10 20 30 40 50 60
```

일차원  
벡터

5/2  
4



## 소스4. Filter() 함수

글자수가 n개 이상인 글자만 선택하는 함수를 만들어 필터링

Filter(f, x) > x 데이터를 f의 함수에 적용하여 참 또는 거짓의 결과를 가져옴

```
> c2<- c( "최고" , "육질" , "살아있" , "배송" , "상품" , "기가막히게" , "사장님" , "감사" , "부" , "부족")
```



```
c3 <- Filter( function(x) {nchar(x) >2}, c2)
```



3글자  
이상

```
> c3  
[1] "살아있"      "기가막히게" "사장님"
```

※ 각 단어를 x에 대입하여 길이가 2보다 크면 true

```
function(x){  
  nchar(x) >2  
  return(True)  
}
```

#nchar 함수(count the number of Characters)  
->문자열의 길이, 개수를 반환하는 함수

```
>nchar("기가막히게")
```

## 소스5. sort() 함수

정렬

```
> c2  
[1] "살아있" "기가막히게" "사장님" "기가막히게"
```

```
wordcnt <- table(c2)
```

c2를 table 함수를 이용해 단어별 빈도수가 나오게 만들고, wordcnt에 저장

```
> wordcnt  
c2  
기가막히게  사장님  살아있  
          2      1      1
```

T 또는  
TRUE

오름차순  
:F 또는  
FALSE

```
sort(wordcnt, decreasing = T)
```

# 내림차순으로 정렬해서 어떤 단어가 많이 나왔는지 확인

```
> sort(wordcnt, decreasing = TRUE)  
c2  
기가막히게  사장님  살아있  
          2      1      1
```

## 소스6. brewer.pal( )

### 색지정

```
# RColorBrewer : 다양한 색상을 적용하기 위한 패키지 설치
install.packages("RColorBrewer")
library(RColorBrewer)

# 팔레트 지정
> Dark2 <- brewer.pal(8, "Dark2")
```

"Dark2" 팔레트  
에서 8가지 색  
지정

```
> Dark2
[1] "#1B9E77" "#D95F02" "#7570B3" "#E7298A" "#66A61E" "#E6AB02" "#A6761D" "#666666"
```

"Dark2"  
대신  
사용할 수  
있는 색상

‘Set1’ : 밝은 색조와 대조를 갖는 9개의 색상으로 이루어진 팔레트  
‘Set2’ : 보다 매끄러운 톤의 8개 색상으로 이루어진 팔레트  
‘Set3’ : 더 많은 색상을 포함하며 다양한 톤의 12개 색상으로 이루어진 팔레트  
‘Paired’ : 연관된 데이터를 나타내는 12개의 색상으로 이루어진 팔레트  
‘Accent’ : 강조할 데이터를 나타내는 8개의 색상으로 이루어진 팔레트



## 소스7. wordcloud() 함수

```
wordcloud(names(wordcount), # 단어들
          freq=wordcount,   # 단어들의 빈도
          scale=c(6,0.7),   # 단어의 폰트 크기
          min.freq=3,        # 단어의 최소 빈도
          random.order=F,    # 단어의 출력 위치
          rot.per= .1,       # 90도 회전 단어 비율
          colors=pal2)       # 단어의 색
```

- **names(wordcount)**

워드클라우드 상에 표시할 단어를 지정한다.

- **freq=wordcount**

워드클라우드 상에 표시할 단어의 빈도수를 지정한다.

- **scale=c(6,0.7)**

표시할 단어의 폰트 크기를 지정한다. 여기서 6은 폰트의 최대 크기, 0.7은 폰트의 최소 크기를 의미한다.

- **min.freq=3**

빈도수가 3 이상인 단어들만 표시한다.

- **random.order=F**

단어가 표시될 위치를 지정한다. T는 단어의 표시 위치를 무작위로 지정할 수 있고, F는 빈도수가 높은 단어일수록 중앙쪽에 배치된다.

- **rot.per=.1**

단어를 표시할 때 세로 방향으로 표시할 단어의 비율을 지정한다. 여기서 .1은 10%를 의미한다.

- **colors=pal2**

빈도수에 따라 pal2에 있는 색으로 단어의 색을 지정한다.

## 소스7. wordcloud() 함수

```
wordcloud(names(wordcount), # 단어들
          freq=wordcount,   # 단어들의 빈도
          scale=c(4,0.5),   # 단어의 폰트 크기(최대,최소)
          min.freq=3,       # 단어의 최소 빈도
          random.order=F,   # 단어의 출력 위치
          rot.per=.1,       # 90도 회전 단어 비율
          colors=pal2)      # 단어의 색
```

```
> wordcloud(names(wordcnt), freq=wordcnt, scale=c(4, 0.5), rot.per=1, min.freq=1, random.order=F, random.color=T, colors=Dark2)
```

글자방향:  
세로 100%

가장 큰 글씨 폰트  
4, 가장 작은 글씨 0.5

```
> wordcloud(names(wordcnt), freq=wordcnt, scale=c(4, 0.5), rot.per=0.25, min.freq=1, random.order=F, random.color=T, colors=Dark2)
```

글자방  
향:세로  
가 25%

빈도수 큰  
글자가  
나타나는  
위치:가  
운데

```
> wordcloud(names(wordcnt), freq=wordcnt, scale=c(4, 0.5), rot.per=0.25, min.freq=1, random.order=T, random.color=T, colors=Dark2)
```

빈도수  
큰글자가  
나타나는  
위치 랜덤

---

## 0. 소스 분석 응용

문제1~문제7까지 문제를 보고 답을 쓰시오.

---

## 소스1(문제1). readLines() & extractNoun()

readLines() 함수를 이용해 [약자.txt]의 문장을 불러오고 extractNoun() 함수를 이용하여 명사 추출하도록 (가)~(다)를 채우시오.

```
# 한글이 불러오고 깨지기 때문에 encoding 옵션에 UTF-8 지정  
약자txt <- readLines( (가) )
```

ksj.txt - Windows 메모장  
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)  
닭이 너무 맛있어요 최고!! 육질이 살아있음  
배송도 빠르고 상품도 좋습니다. ^^

txt

"배송도 빠르고 상품도 좋습니다. ^^"

```
# 약자txt에서 명사만 뽑아서 n에 저장  
n <- extractNoun( (나) )
```

명사

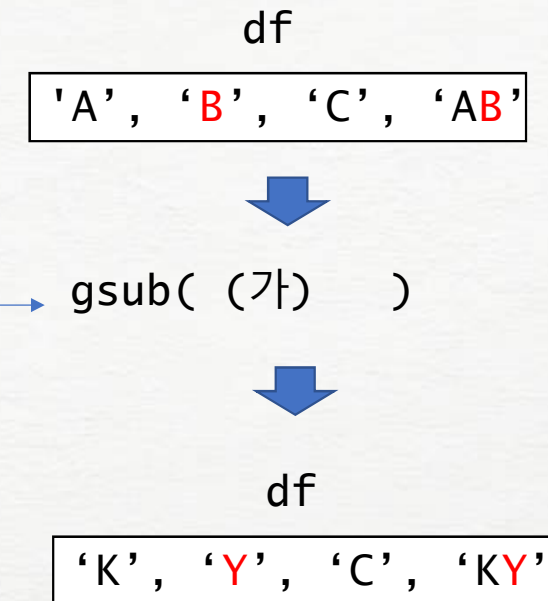
(다)

## 소스2(문제2). gsub() 함수

### ■ 텍스트 수정

gsub() 함수를 이용해 아래와 같이 텍스트를 수정하도록 (가)를 채우시오.

df 데이터셋에 있는 'B'를 모두 'Y'로 변경함





## 소스3(문제3). unlist()

벡터, 행렬, 데이터프레임을 하나씩 정의하여 list를 벡터로 만든 결과 (가)를 쓰시오.

```
v=c(4,5,6)
m=matrix(2:5,2)
df=data.frame(a=c(10,20,30),b=c(40,50,60))
```

```
> m3=list(v,m,df)
> m3
[[1]]          #벡터
[1] 4 5 6
[[2]]          #매트릭스
[,1] [,2]
[1,] 2 4
[2,] 3 5
[[3]]          #데이터프레임
  a b
1 10 40
2 20 50
3 30 60
```

```
➤ unlist(m3) # 리스트에 unlist 함수를 적용
a1 a2 a3 b1 b2 b3
```

(가)

벡터

## 소스4(문제4). Filter() 함수

글자수가 4개 이상인 데이터만 선택하도록 (가)를 쓰시오.

```
> c2<- c( "최고" , "육질" , "살아있" , "배송" , "상품" , "기가막히게" , "사장님" , "감사" , "부" , "부족")
```



```
c3 <- Filter((가), c2)
```



```
> c3  
[1] "기가막히게"
```

※ R에서 사용자 정의 함수를 만드는 방법

```
function(인자1, 인자2, ...){
```

함수내용

```
return(반환값) # 없을 시 생략
```

```
}
```

#nchar 함수(count the number of Characters)

->문자열의 길이, 개수를 반환하는 함수

```
>nchar("기가막히게")
```

## 소스5(문제5). sort() 함수

정렬된 결과가 나오도록 (가), (나), (다)를 채우시오.

```
> c2  
[1] "살아있" "기가막히게" "사장님" "기가막히게"
```

```
wordcnt <- (가)(c2)
```

단어별 빈도수가 나오게 만들고, wordcnt에 저장

```
> wordcnt  
c2  
기가막히게  사장님  살아있  
          2      1      1
```

```
(나)(wordcnt, (다) )
```

# 오름차순으로 정렬해서 어떤 단어가 많이 나왔는지 확인

```
c2  
사장님  살아있  기가막히게  
      1      1      2
```

## 소스6(문제6). brewer.pal( )

6가지의 색을 지정하도록 (가)를 채우시오.

```
# RColorBrewer :다양한 색상을 적용하기 위한 패키지 설치
install.packages("RColorBrewer")
library(RColorBrewer)

# 팔레트 지정
> Dark2 <- (가)
```

```
> Dark2
[1] "#1B9E77" "#D95F02" "#7570B3" "#E7298A" "#66A61E"
```

## 소스7(문제7). wordcloud() 함수

```
wordcloud(names(wordcount), # 단어들
          freq=wordcount,   # 단어들의 빈도
          scale=c(6,0.7),   # 단어의 폰트 크기
          min.freq=3,       # 단어의 최소 빈도
          random.order=F,   # 단어의 출력 위치
          rot.per=.1,       # 90도 회전 단어 비율
          colors=pal2)      # 단어의 색
```

```
wordcloud(names(wordcnt), freq=wordcnt, scale=c(1, 0.1),
          rot.per=0.25, min.freq=1, random.order=F, random.color=T,
          colors=Dark2)
```



글자의 가로 세로 방향이  
반반씩 나오게 하고, 글자의  
크기가 가장 큰 값은 3, 가장  
작은 값은 0.2가 되도록  
수정하시오.

```
wordcloud(names(wordcnt), freq=wordcnt,
          scale=c(1, 0.1), rot.per=0.25, min.freq=1,
          random.order=F, random.color=T, colors=Dark2)
```



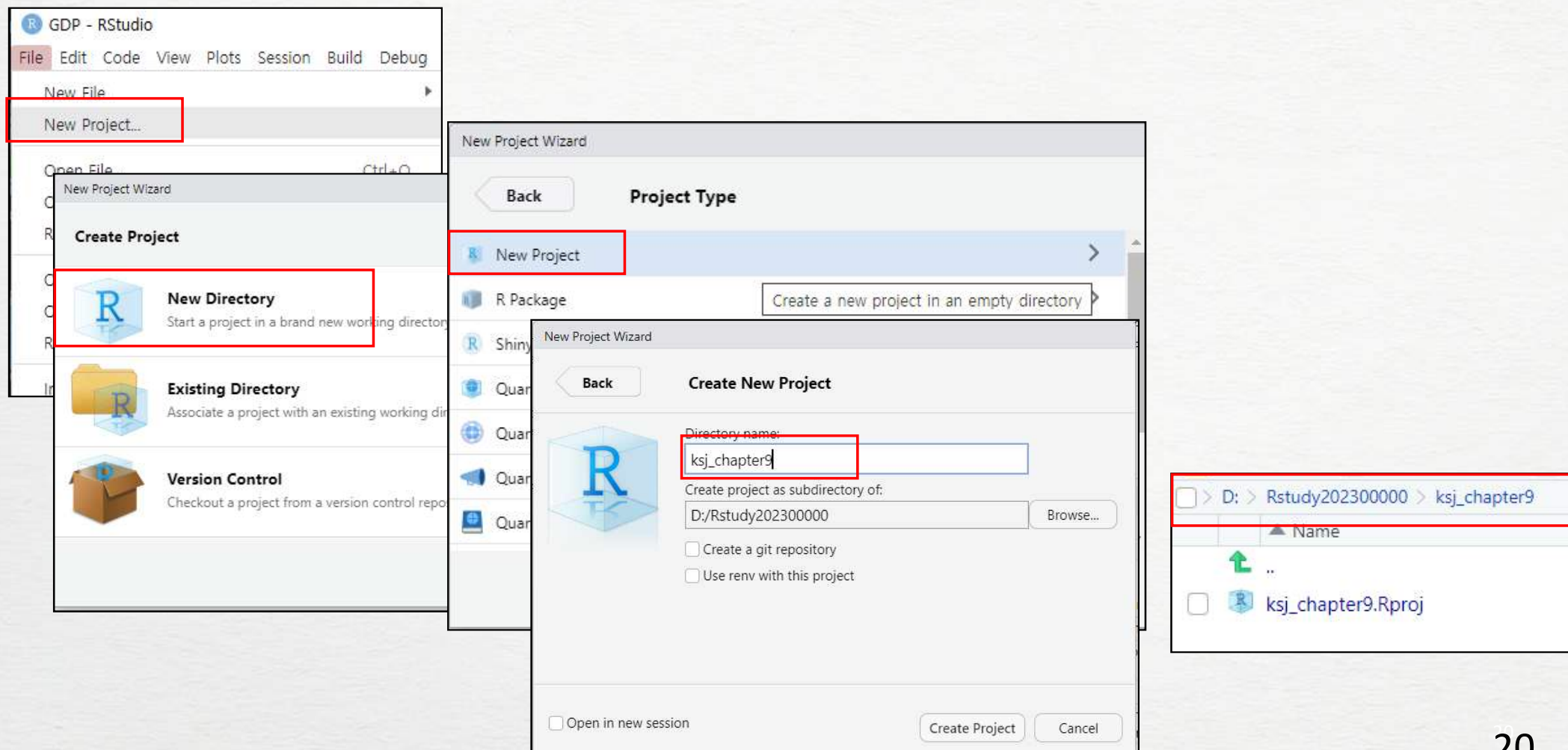


---

# **I. 워드클라우드&패키지설치**

---

## \* 프로젝트 시작



# 1. 워드클라우드란?

- 워드클라우드(Word Cloud)는 말 그대로 단어를 구름처럼 표현하는 방법으로 많은 키워드(텍스트) 중에서 가장 빈도수가 높은 단어를 크기와 색상으로 강조해 시각화 시킨 것



<워드클라우드 예시, wikipedia>

## 2. 워드클라우드의 목적

- 지금까지 숫자 형태의 데이터를 다루는 방법에 관하여 학습
- 분석 대상 데이터 중에는 숫자가 아닌 문자나 문장 형태의 데이터도 있음 -ex) 이메일 내용이나 SNS 메시지, 댓글
- 워드클라우드(word cloud)는 문자형 데이터를 분석하는 대표적인 방법으로, 대상 데이터에서 단어(주로 명사)를 추출하고 단어들의 출현 빈도수를 계산하여 시각화하는 기능
- 출현 빈도수가 높은 단어는 그만큼 중요하거나 관심도가 높다는 것을 의미



워드클라우드의 예



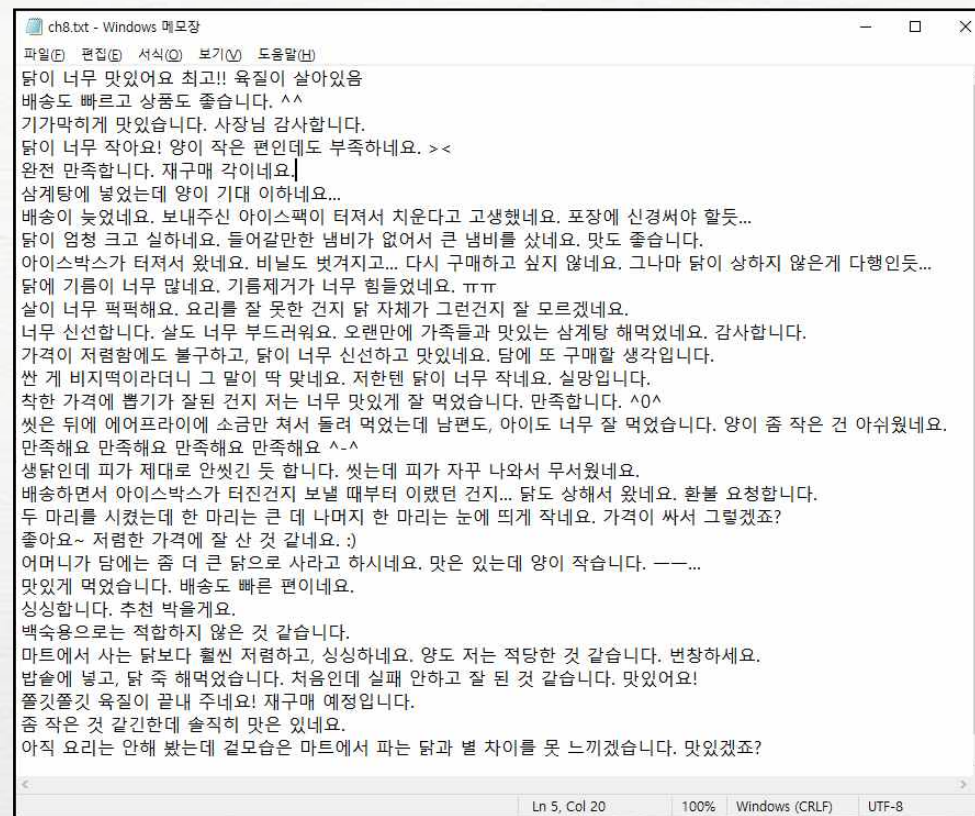
## 3. 텍스트 데이터

### 3.1 Raw 데이터 확인

- 분석 목적 : 쇼핑몰의 리뷰를 통해 고객 만족 여부 조사

ch8.txt 데이터 셋

총 30개의 고객리뷰를 담은 한글 텍스트 파일 데이터



```
ch8.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
닭이 너무 맛있어요 최고!! 육질이 살아있음
배송도 빠르고 상품도 좋습니다. ^^
기가막히게 맛있습니다. 사장님 감사합니다.
닭이 너무 작아요! 양이 작은 편인데도 부족하네요. ><
완전 만족합니다. 재구매 각이네요!
삼계탕에 넣었는데 양이 기대 이하네요...
배송이 늦었네요. 보내주신 아이스팩이 터져서 치운다고 고생했네요. 포장에 신경써야 할듯...
닭이 엄청 크고 실하네요. 들어갈만한 냄비가 없어서 큰 냄비를 샀네요. 맛도 좋습니다.
아이스박스가 터져서 왔네요. 비닐도 벗겨지고... 다시 구매하고 싶지 않네요. 그나마 닭이 상하지 않은게 다행인듯...
닭에 기름이 너무 많네요. 기름제거가 너무 힘들었네요. πππ
살이 너무 찝찝해요. 요리를 잘 못한 건지 닭 자체가 그런건지 잘 모르겠네요.
너무 신선합니다. 살도 너무 부드러워요. 오랜만에 가족들과 맛있는 삼계탕 해먹었네요. 감사합니다.
가격이 저렴한데도 불구하고, 닭이 너무 신선하고 맛있네요. 닭에 또 구매할 생각입니다.
싼 게 비지떡이라더니 그 말이 딱 맞네요. 저한테 닭이 너무 작네요. 실망입니다.
착한 가격에 뽕기가 잘된 건지 저는 너무 맛있게 잘 먹었습니다. 만족합니다. ^0^
씻은 뒤에 에어프라이어에 소금만 쳐서 돌려 먹었는데 남편도, 아이도 너무 잘 먹었습니다. 양이 좀 작은 건 아쉬웠네요.
만족해요 만족해요 만족해요 만족해요 ^^
생닭인데 피가 제대로 안씻긴 듯 합니다. 씻는데 피가 자꾸 나와서 무서웠네요.
배송하면서 아이스박스가 터진건지 보낼 때부터 이랬던 건지... 닭도 상해서 왔네요. 환불 요청합니다.
두 마리를 시켰는데 한 마리는 큰 데 나머지 한 마리는 눈에 띄게 작네요. 가격이 싸서 그럴까요?
좋아요~ 저렴한 가격에 잘 산 것 같네요. :)
어머니가 담에는 좀 더 큰 닭으로 사라고 하시네요. 맛은 있는데 양이 작습니다. ---...
맛있게 먹었습니다. 배송도 빠른 편이네요.
싱싱합니다. 추천 박을게요.
백숙용으로는 적합하지 않은 것 같습니다.
마트에서 사는 닭보다 훨씬 저렴하고, 싱싱하네요. 양도 저는 적당한 것 같습니다. 변창하세요.
밥술에 넣고, 닭 죽 해먹었습니다. 처음인데 실패 안하고 잘 된 것 같습니다. 맛있어요!
쫄깃쫄깃 육질이 끝내 주네요! 재구매 예정입니다.
좀 작은 것 같긴한데 솔직히 맛은 있네요.
아직 요리는 안해 봤는데 겉모습은 마트에서 파는 닭과 별 차이를 못 느끼겠습니다. 맛있겠죠?
```

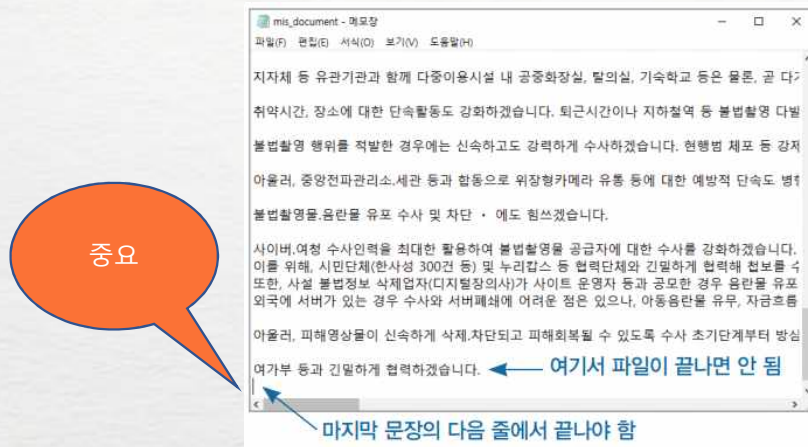


### 3. 텍스트 데이터

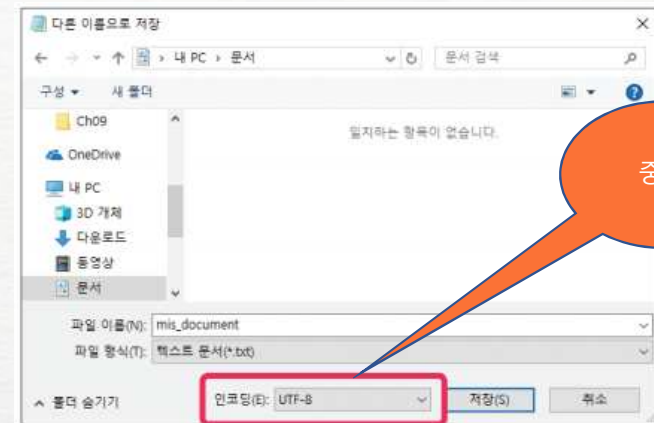
#### 3.2 워드클라우드 문서 파일 준비

응용에서  
작성할  
예정

- 워드클라우드를 작성할 대상 문서는 일반적으로 텍스트 파일 형태로 준비
- 파일의 끝부분 처리를 아래 왼쪽 그림과 같이 마지막 문장이 끝나면 반드시 줄 바꿈을 한 후 저장
- 파일을 저장할 때, [다른 이름으로 저장]을 선택하고 아래 오른쪽 그림과 같이 인코딩을 'UTF-8'로 선택을 하여 저장
- 파일 이름이나 파일이 저장된 폴더 경로에 한글이 포함되어 있으면 파일을 읽을 때 에러가 발생하는 경우가 있으므로 파일을 저장할 때는 파일 이름을 영어로 설정



텍스트 파일 끝부분에서의 줄 바꿈

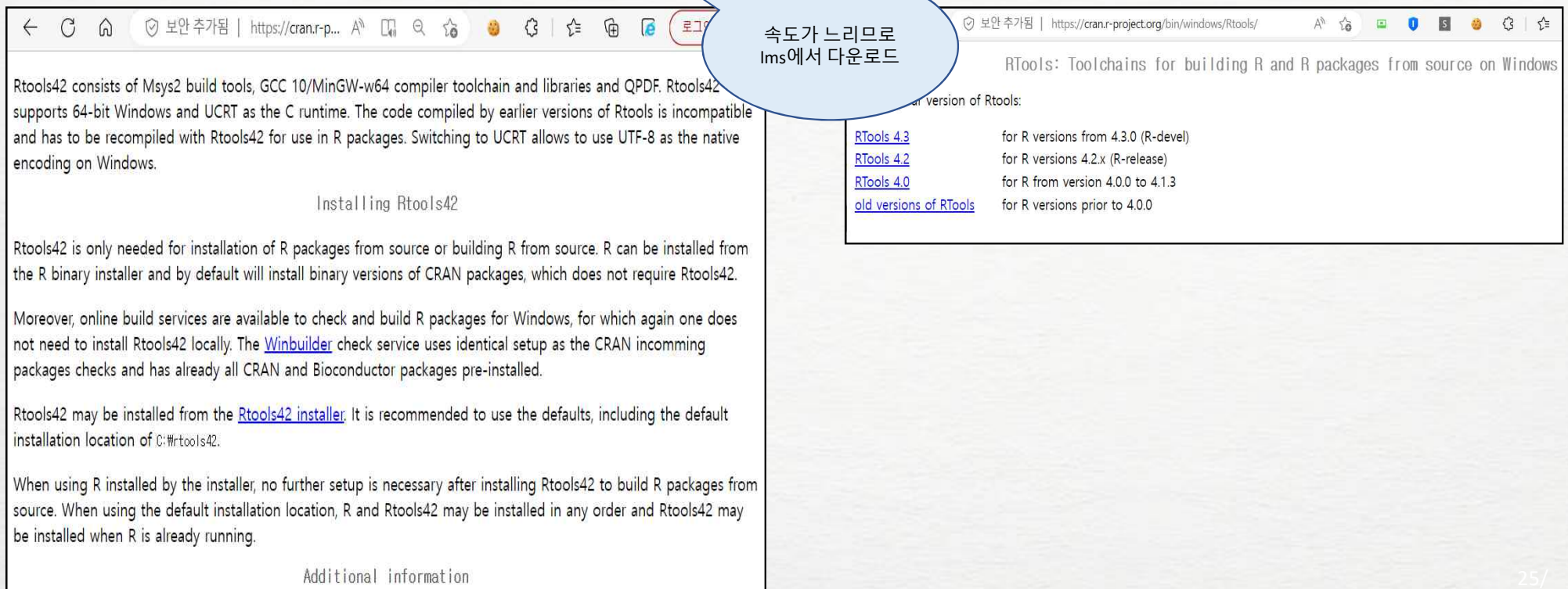


UTF-8로 텍스트 파일 저장

## 4. Rtools 설치

### 4.1 Rtools 다운로드

- 워드클라우드 및 감성 분석에서 필요한 패키지들이 Rtools를 요구하기 때문에 사전에 프로그램 설치
- <https://cran.r-project.org/bin/windows/Rtools/>



A screenshot of the Rtools download page on the CRAN website. The page title is "RTools: Toolchains for building R and R packages from source on Windows". The main content area is titled "Installing Rtools42" and contains several paragraphs of text explaining the toolchain and its installation. A callout bubble points to the URL in the browser's address bar, containing the text "속도가 느리므로 lms에서 다운로드" (Download from lms as the speed is slow). The right sidebar lists different versions of RTools for various R versions.

속도가 느리므로 lms에서 다운로드

RTools42 consists of Msys2 build tools, GCC 10/MinGW-w64 compiler toolchain and libraries and QPDF. Rtools42 supports 64-bit Windows and UCRT as the C runtime. The code compiled by earlier versions of Rtools is incompatible and has to be recompiled with Rtools42 for use in R packages. Switching to UCRT allows to use UTF-8 as the native encoding on Windows.

### Installing Rtools42

Rtools42 is only needed for installation of R packages from source or building R from source. R can be installed from the R binary installer and by default will install binary versions of CRAN packages, which does not require Rtools42.

Moreover, online build services are available to check and build R packages for Windows, for which again one does not need to install Rtools42 locally. The [Winbuilder](#) check service uses identical setup as the CRAN incoming packages checks and has already all CRAN and Bioconductor packages pre-installed.

Rtools42 may be installed from the [Rtools42 installer](#). It is recommended to use the defaults, including the default installation location of C:\Rtools42.

When using R installed by the installer, no further setup is necessary after installing Rtools42 to build R packages from source. When using the default installation location, R and Rtools42 may be installed in any order and Rtools42 may be installed when R is already running.

### Additional information

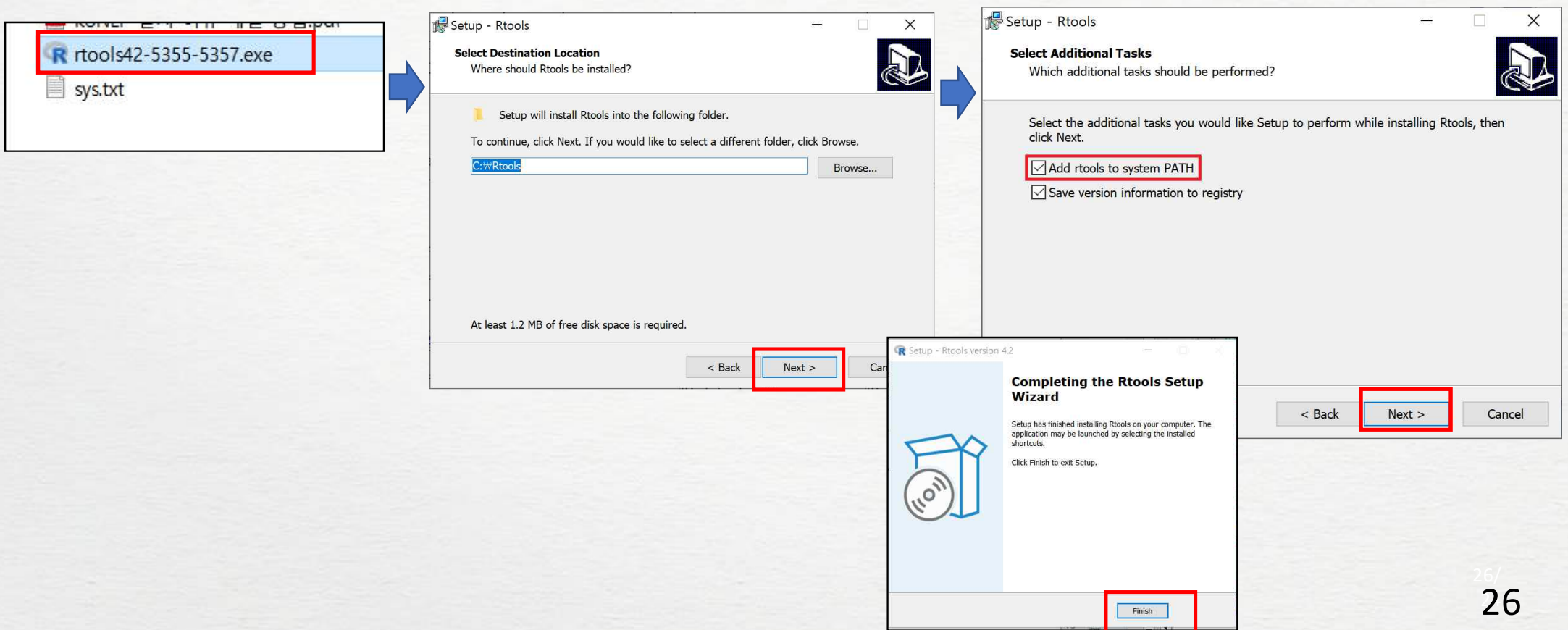
Current version of Rtools:

<a href="#">RTools 4.3</a>	for R versions from 4.3.0 (R-devel)
<a href="#">RTools 4.2</a>	for R versions 4.2.x (R-release)
<a href="#">RTools 4.0</a>	for R from version 4.0.0 to 4.1.3
<a href="#">old versions of RTools</a>	for R versions prior to 4.0.0

## 4. Rtools 설치

### 4.2 Rtools 설치하기

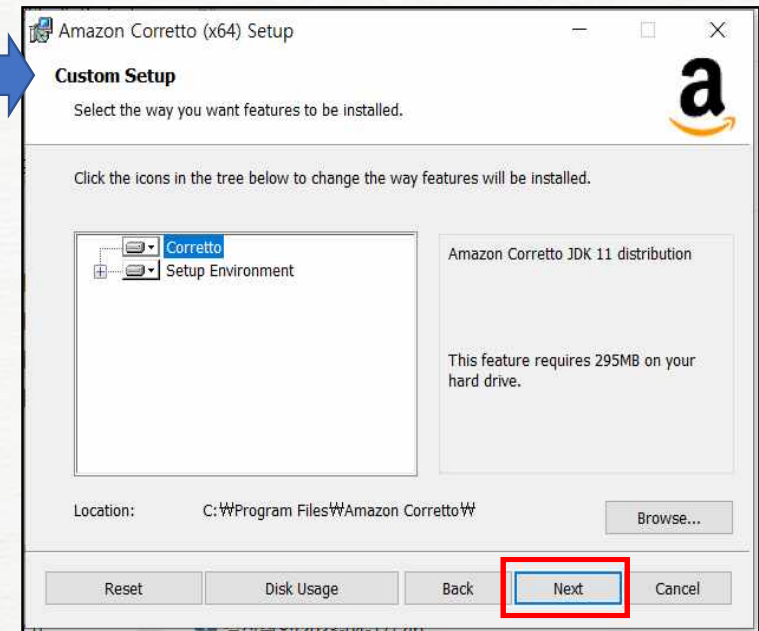
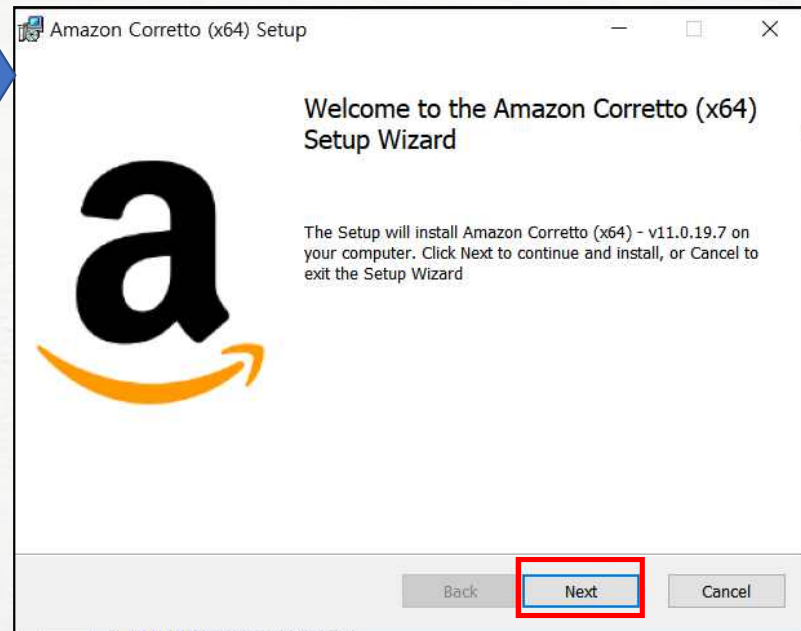
- Next 버튼을 눌러 설치하되 Add rtools to system PATH 체크 후 설치 마무리



## 5. Java설치

아래 사이트에서 다운로드

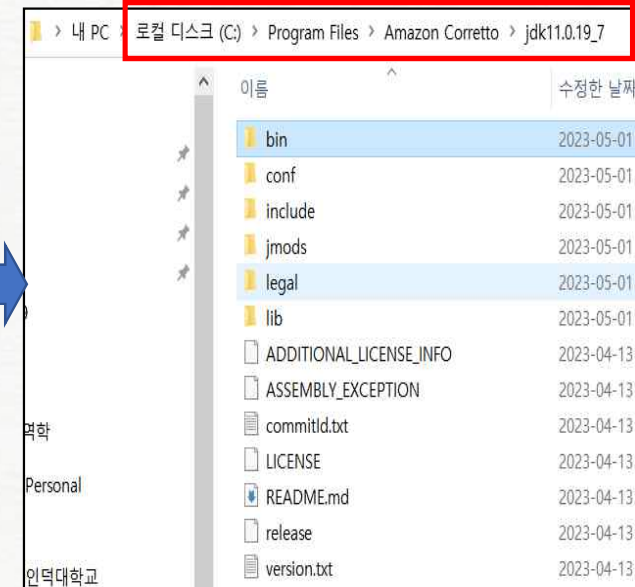
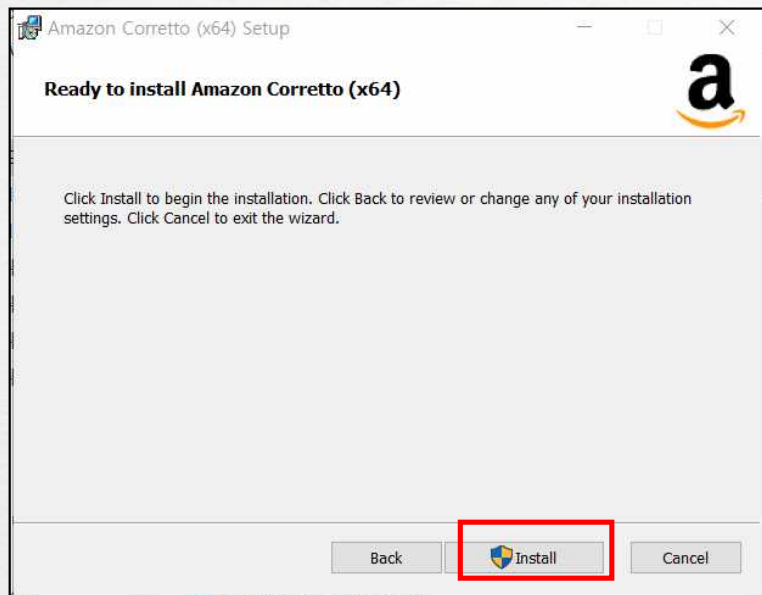
<https://corretto.aws/downloads/latest/amazon-corretto-11-x64-windows-jdk.msi>





## 5. Java설치

설치  
확인





## 6. 패키지 설치

### 6.1 의존성 패키지 설치하기

- KoNLP 패키지 설치를 위한 여러 패키지 설치

#### CODE

```
# 한글 자연어 분석 패키지 KoNLP(Korea Natural Language Processing)
# KoNLP 패키지가 더 이상 업데이트 되지 않아 기본 명령어로 설치되지 않음
# 따라서 의존성 패키지를 먼저 설치하고, github에 올려진 패키지를 수동 설치
> install.packages("hash")
> install.packages("rJava")
> install.packages("tau")
> install.packages("Sejong")
> install.packages("RSQLite")
> install.packages("devtools")

# Java 실행 문제 제거를 위해 설치
> install.packages("multilinguer")
> library(multilinguer)
```

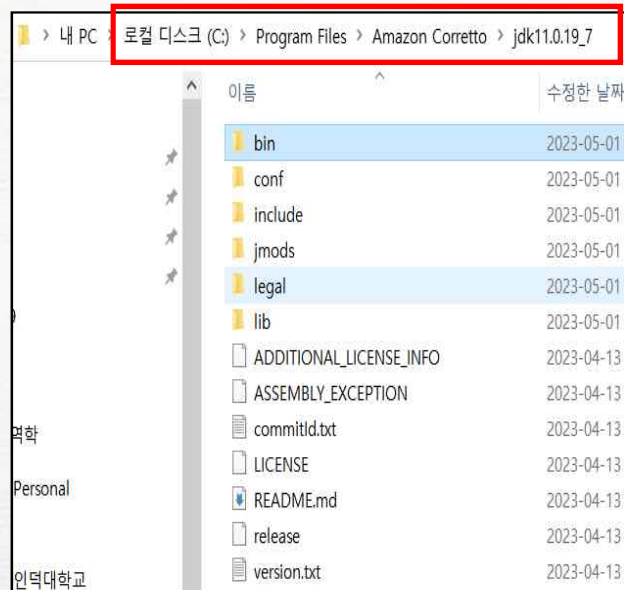
## 6. 패키지 설치

### 6.2 Java설치 경로

- KoNLP 패키지 설치를 하기 위해서는 JAVA가 필요하므로 JAVA가 설치된 폴더의 경로를 등록함

#### CODE

```
> Sys.setenv(JAVA_HOME="C:/Program Files/Amazon Corretto/jdk11.0.19_7") # JAVA가 설치된 경로
```



## 6. 패키지 설치

### 6.3 KoNLP 패키지 설치

- KoNLP 패키지가 필요하나 내부적인 문제로 업데이트되지 않아 CRAN 저장소에서 제거되어 원격으로 GitHub를 통해 설치

#### CODE

```
> install.packages("remotes")  
> library(remotes)  
> install_github('haven-jeon/KoNLP', upgrade = "never", INSTALL_opts=c("--no-multiarch"))
```

```
Fail to locate 'scala-library-2.11.8.jar'. Recommend to locate 'scala-library-2.11.8.jar' manually on C:/Users/sujko/AppData/Local/R/win-library/4.2/KoNLP/java  
** testing if installed package keeps a record of temporary installation path  
* DONE (KoNLP)
```

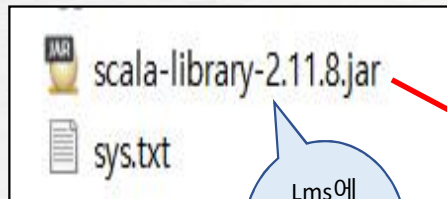
최종  
메세지

설치후  
파일이  
존재하지  
않는다는  
메시지를  
띄움

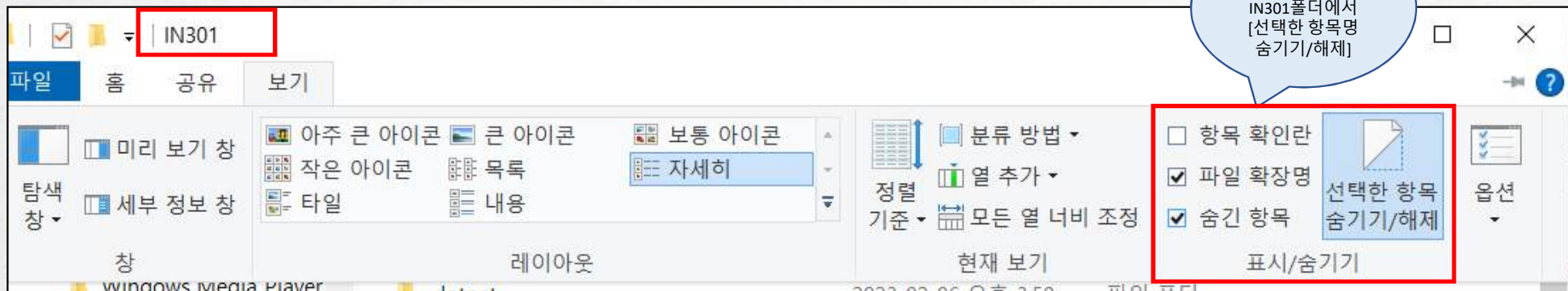
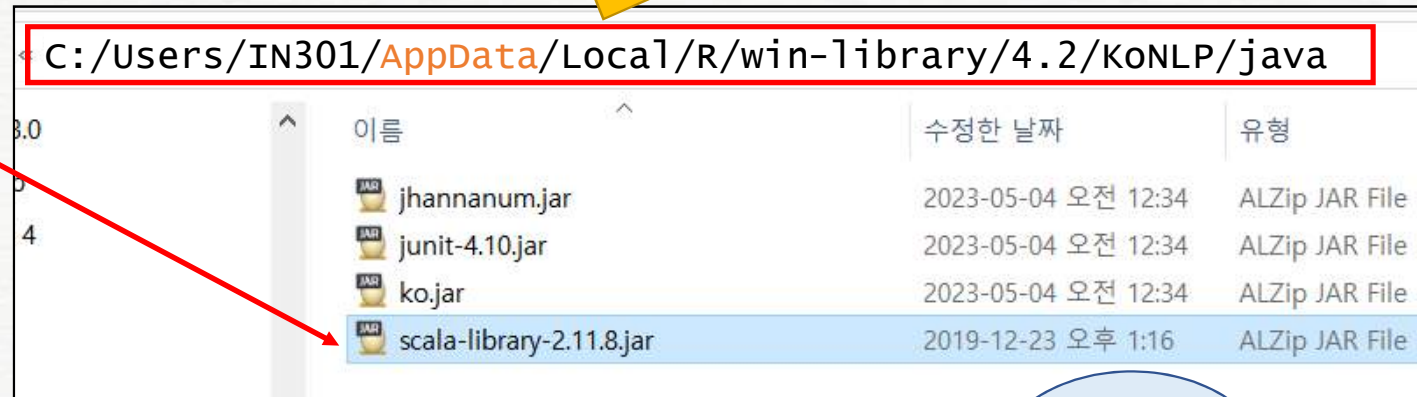
## 6. 패키지 설치

### 6.4 scala-library-2.11.8.jar

폴더확인  
->안할 경우  
실행안됨



Lms에  
서  
다운로  
드



## 6. 패키지 설치

### 6.5 KoNLP 라이브러리 셋팅

- KoNLP 시작

CODE

```
> library(KoNLP)  
#Checking user defined dictionary!
```

Checking user defined  
dictionary!메세지가 나오면  
성공



## 7. KoNLP를 이용한 한국어 텍스트 마이닝

- KoNLP는 한국어를 전용으로 처리하는 텍스트 마이닝 라이브러리
  - SejongDic, NIADic이라는 세 종류의 사전을 지원함

### CODE

```
> useSejongDic()  
> useNIADic()
```

```
> library(KoNLP)  
> useSejongDic()  
Backup was just finished!  
  
370957 words dictionary was built.  
  
> useNIADic()  
Backup was just finished!  
  
983012 words dictionary was built.
```

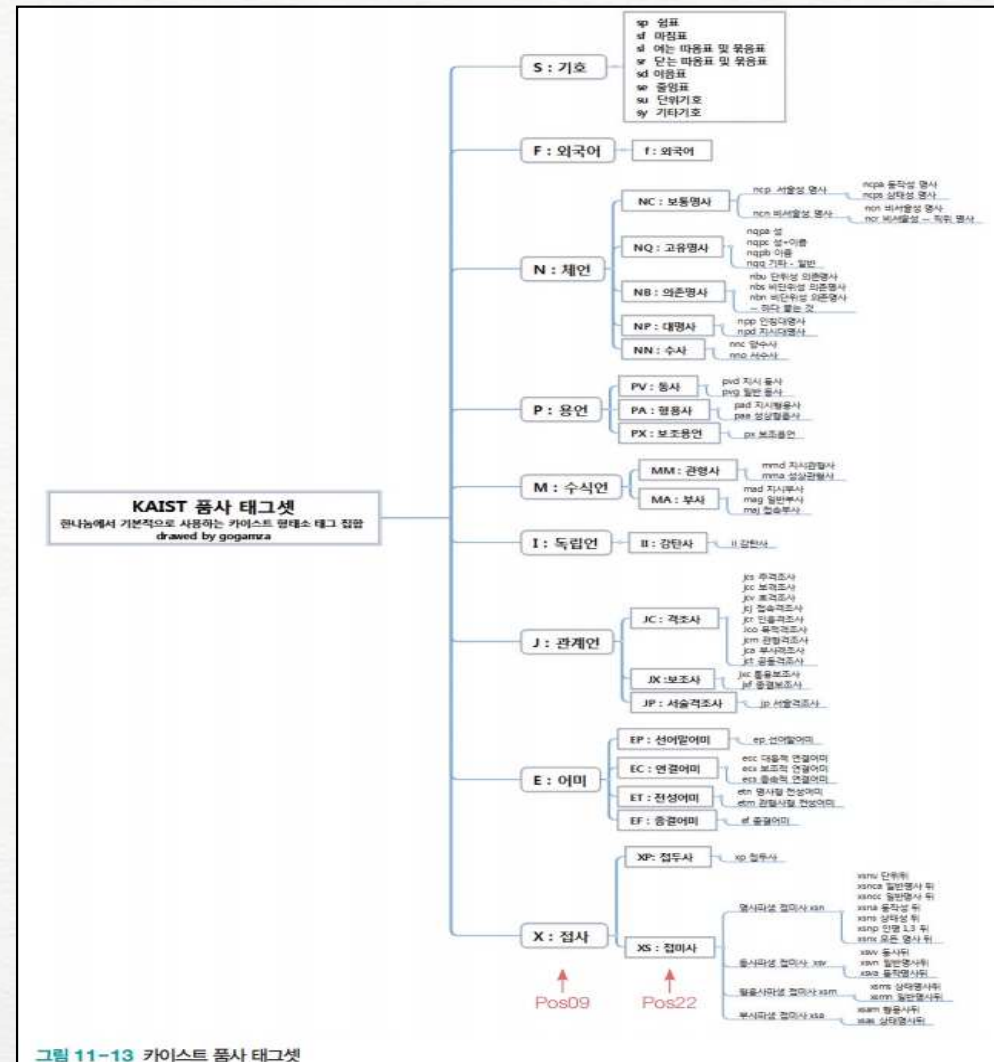
사전 크기

## 8. KoNLP를 이용한 형태소 분석

- 한국어 품사 태그셋
  - Pos09는 9 종류로 품사 구분
  - Pos22는 22 종류로 품사 구분

**NOTE** KoNLP와 카이스트 품사 태그셋에 관한 자세한 내용은 다음 문서를 참조하라.

<https://cran.r-project.org/web/packages/KoNLP/vignettes/KoNLP-API.html>



## 8. KoNLP를 이용한 형태소 분석

- 한글 형태소 분석 예제
  - 형태소<sub>morpheme</sub>란 언어를 구성하는 가장 작은 문법 요소
  - 형태소 분석이란 문장을 형태소 단위로 분할하는 작업

extractNoun 함수는 명사를 추출

```
> useSejongDic()
> s='너에게 묻는다. 연탄재 함부로 발로 차지 마라 너는 누구에게 한번이라도 뜨거운 사람이었느냐'
> extractNoun(s)
[1] "너"      "연탄재"  "발"      "차"      "너"      "누구"    "한"      "번"
[9] "사람이었" "나"
```

```
s='너에게 묻는다. 연탄재 함부로 발로 차지 마라 너는 누구에게 한번이라도 뜨거운 사람이었느냐'
extractNoun(s)
```

---

## II. 고객만족도 데이터 워드클라우드 시각화

---

# 1. 사전 불러오기

## 세종사전 불러오기

- KoNLP 패키지의 useSejongDic() 함수를 통해 세종사전 다운로드 및 불러오기

### CODE

```
# 한글처리에 필요한 세종 사전 수행, 최초 실행 시 1을 입력해 설치 실시  
> useSejongDic()
```

```
> # 한글처리에 필요한 세종 사전 수행  
> useSejongDic()  
Backup was just finished!  
Downloading package from url: https://github.com/haven-jeon/NIADic/releases/  
download/0.0.1/NIADic_0.0.1.tar.gz  
These packages have more recent versions available.  
It is recommended to update all of them.  
which would you like to update?  
  
1: All  
2: CRAN packages only  
3: None  
4: vctrs (0.3.5 -> 0.3.6)  
5: fansi (0.4.1 -> 0.4.2)  
6: crayon  
7: cli  
8: tibble  
9: diff  
10: withr  
11: waldo  
12: bro  
13: xfun  
14: fastmap (1.0.1 -> 1.0.2)  
15: tinytex (0.28 -> 0.29)  
16: htmltools (0.5.0 -> 0.5.1.1) [CRAN]  
17: knitr (1.30 -> 1.31) [CRAN]  
18: Rcpp (1.0.5 -> 1.0.6) [CRAN]  
19: DBI (1.1.0 -> 1.1.1) [CRAN]  
20: ggplot2 (3.3.2 -> 3.3.3) [CRAN]  
21: data.table (1.13.4 -> 1.13.6) [CRAN]  
  
Enter one or more numbers, or an empty line to skip updates:  
Building the package will delete...  
'C:/AppData/Local/Temp/Rtmp8ggZnR/remotes6887eba7df9/NIADic/in  
st/doc'  
Are you sure?  
  
1: Yes  
2: No  
  
선택: 1]
```

안 나올 경우 다음 장으로 넘어감

<useSejongDic() 함수 최초 실행 시 발생할 수 있는 메시지>

```
Console Terminal Jobs  
C:/r-study/ ↗  
  
선택: 1  
✓ checking for file 'C:/Users/.../AppData/Local/Temp/Rtmp6Ngab2/remotes  
4634e055fa3/NIADic/DESCRIPTION' (339ms)  
- preparing 'NIADic':  
✓ checking DESCRIPTION meta-information  
- installing the package to build vignettes  
✓ creating vignettes (11.7s)  
- checking for LF line-endings in source  
- checking for empty or unneeded directories  
- building 'NIADic_0.0.1.tar.gz'  
  
* installing *source* package 'NIADic' ...  
** using staged installation  
** R  
** inst  
** byte-compile and prepare package for lazy loading  
** help  
*** installing help indices  
converting help for package 'NIADic'  
finding HTML links ... 완료  
get_dic  
html  
*** building package indices  
*** installing vignettes  
** testing if installed package can be loaded from temporary location  
** testing if installed package can be loaded from final location  
** testing if installed package keeps a record of temporary installation path  
h  
* DONE (NIADic)  
370957 words dictionary was built.  
> |
```

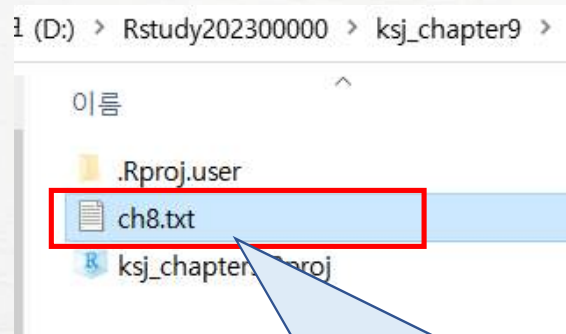
안 나올 경우 다음 장으로 넘어감

<useSejongDic() 함수 실행이 정상적으로 완료된 메시지>



## 2 데이터 불러오기

### 2.1 데이터 셋팅



Lms에서 다운 받은 ch8.txt 파일을  
[약자\_chapter9]폴더로 이동

## 2. 데이터 불러오기

### 2.2 데이터 불러오기

- readLines() 함수를 이용해 ch8.txt 파일을 불러오되 R에서 한글이 깨지기 때문에 encoding 옵션에 UTF-8 지정

#### CODE

```
> txt <- readLines("ch8.txt", encoding = "UTF-8")
> head(txt)
[1] "닭이 너무 맛있어요 최고!! 육질이 살아있음"
[2] "배송도 빠르고 상품도 좋습니다. ^^"
[3] "기가막히게 맛있습니다. 사장님 감사합니다."
[4] "닭이 너무 작아요! 양이 작은 편인데도 부족하네요. ><"
[5] "완전 만족합니다. 재구매 각이네요."
[6] "삼계탕에 넣었는데 양이 기대 이하네요..."
```

## 3. 데이터 가공하기

### 3.1 텍스트 데이터 가공하기

- KoNLP 라이브러리의 `extractNoun()` 함수를 이용해 명사만 추출

#### CODE

```
# txt에서 명사만 뽑아서 n에 저장
> n <- extractNoun(txt)
> head(n) # 데이터 확인
[[1]]
[1] "닭"    "최고"  "육"    "질"    "살아있" "음"

[[2]]
[1] "배송"  "상품"  "좋습니"

[[3]]
[1] "기가막히게" "사장님"  "감사"

[[4]]
[1] "닭"  "양"  "편"  "부족"

[[5]]
[1] "완전" "만족" "재구" "매"  "각이"

[[6]]
[1] "삼계탕" "양"  "기대" "이하"
```

ch8.txt 파일의 각 라인에서 명사만을  
추출함

## 3. 데이터 가공하기

### 3.2 gsub() 함수를 이용해 텍스트 수정

#### CODE

```
# 텍스트 수정을 위해 n의 내용을 unlist해서 c에 저장함
> c <- unlist(n)

# gsub 함수를 통해 텍스트 수정 실시
> c2 <- gsub("육","육질", c) # "육"은 "육질"로 변경
> c2 <- gsub("재구","재구매", c2) # "재구"는 "재구매"로 변경
> c2 <- gsub("에서","", c2) # "에서"는 제거
> head(c2,30)
[1] "닭"      "최고"    "육질"    "질"      "살아있"
[6] "음"      "배송"    "상품"    "좋습니"  "기가막히게"
[11] "사장님"  "감사"    "닭"      "양"      "편"
[16] "부족"    "완전"    "만족"    "재구매"  "매"
[21] "각이"    "삼계탕"  "양"      "기대"    "이하"
[26] "배송"    "아이스"  "팩"      "고생"    "포장"
```

## 3. 데이터 가공하기

### 3.3 글자수가 2개 이상인 단어만 선택

- Filter() 함수를 이용해 글자수가 2개 이상인 데이터만 선택하는 함수를 만들어 필터링

#### CODE

```
# c2에 저장된 명사 중 두 글자 이상이 되는 것만 필터링
> c3 <- Filter(function(x) {nchar(x) >=2}, c2)
> head(c3,30)
[1] "최고"      "육질"      "살아있"    "배송"      "상품"
[6] "줄습니"    "기가막히게" "사장님"    "감사"      "부족"
[11] "완전"      "만족"      "재구매"    "각이"      "삼계탕"
[16] "기대"      "이하"      "배송"      "아이스"    "고생"
[21] "포장"      "넙비"      "넙비"      "아이스"    "박스"
[26] "비닐"      "벗겨지고.." "구매"      "다행"      "기름제거"
```

※ R에서 사용자 정의 함수를 만드는 방법

```
function(인자1, 인자2, ...){
  함수내용
  return(반환값) # 없을 시 생략
}
```



## 3. 데이터 가공하기

### 3.4 단어별 빈도수

- table() 함수를 이용해 단어별로 빈도수 확인 후 빈도수별 내림차순 정렬

#### CODE

```
# c3를 table 함수를 이용해 단어별 빈도수가 나오게 만들고, wordcnt에 저장
> wordcnt <- table(c3)
# 내림차순으로 정렬해서 어떤 단어가 많이 나왔는지 확인
> sort(wordcnt, decreasing = TRUE)
c3
```

만족	가격	마리	배송	아이스
6	4	3	3	3
감사	구매	냄비	마트	박스
2	2	2	2	2
삼계탕	신선	싱싱	요리	육질
2	2	2	2	2
재구매	저렴	가족	각이	갈긴한데
2	2	1	1	1
겉모습	고생	기가막히게	기대	기름제거
1	1	1	1	1
나머지	남편	다행	박을게요	밥솥
1	1	1	1	1

이하 생략

## 4. 워드클라우드 그리기

- RColorBrewer 라이브러리를 이용해 워드클라우드에 적용한 색상 선택 (Dark2 선택)
- wordcloud 패키지 설치 및 wordcloud() 함수 이용 워드클라우드 그리기

### CODE

```
# 다양한 색상을 적용하기 위해 RColorBrewer 패키지 설치
> library(RColorBrewer)

# 팔레트 지정
> Dark2 <- brewer.pal(8, "Dark2")

# 워드클라우드 패키지 설치 및 라이브러리 불러오기
> install.packages("wordcloud")
> library(wordcloud)

# 워드클라우드로 표현
> wordcloud(names(wordcnt), freq=wordcnt, scale=c(4, 0.5), rot.per=0.25,
min.freq=1, random.order=F, random.color=T, colors=Dark2)
```



∴ 고객리뷰 확인 결과 만족, 재구매와 같은 긍정적인 단어가 크고, 많이 보임

## 5. 워드클라우드 색상 변경

```
CODE # 다양한 색상을 적용하기 위해 RColorBrewer 패키지 설치
> library(RColorBrewer)

# 팔레트 지정
> Dark2 <- brewer.pal(8, "Dark2")

> wordcloud(names(wordcnt), freq=wordcnt, scale=c(4, 0.5), rot.per=0.25, min.freq=1,
random.order=F, random.color=T, colors=Dark2)
```

- Set1 : 밝은 색조와 대조를 갖는 9개의 색상으로 이루어진 팔레트
- Set2 : 보다 매끄러운 톤의 8개 색상으로 이루어진 팔레트
- Set3 : 더 많은 색상을 포함하며 다양한 톤의 12개 색상으로 이루어진 팔레트
- Paired : 연관된 데이터를 나타내는 12개의 색상으로 이루어진 팔레트
- Accent : 강조할 데이터를 나타내는 8개의 색상으로 이루어진 팔레트

“Dark2”  
대신 색을  
변경하여  
실습합니다.



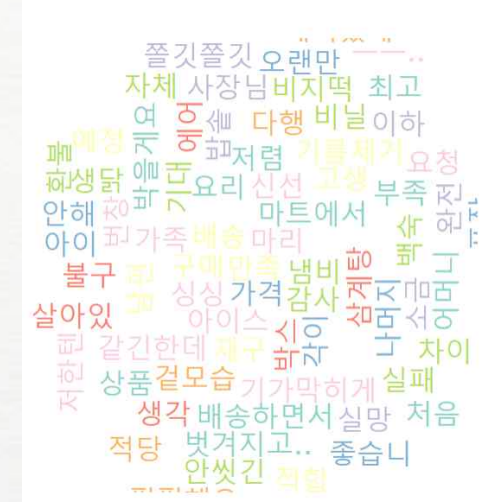
5가지 옵션에 따라 색이 변경됨을 확인

## 6. 워드클라우드 글자 크기 변경

CODE # 다양한 글자크기를 적용

```
> wordcloud(names(wordcnt), freq=wordcnt, scale=c(6, 0.1), rot.per=0.25, min.freq=1, random.order=F, random.color=T, colors=Dark2)
```

최대:6, 4, 2  
최소:0.1, 1, 2



글자크기 변경



## 7. 워드클라우드 글자 위치 변경

CODE # 다양한 글자위치를 적용

```
> wordcloud(names(wordcnt), freq=wordcnt, scale=c(6, 0.1), rot.per=0.25, min.freq=1, random.order=T, random.color=T, colors=Dark2)
```

F를 T로  
변경함



빈도수가 가장  
큰 단어가 정  
중앙에  
나타나지  
않음(랜덤)



글자위치 변경



## 8. 워드클라우드 글자 방향 변경

CODE # 다양한 글자위치를 적용

```
> wordcloud(names(wordcnt), freq=wordcnt, scale=c(6, 0.1), rot.per=1, min.freq=1, random.order=T, random.color=T, colors=Dark2)
```

rot.per=1로  
변경,  
다음으로  
rot.per=0으로  
변경



rot.per=1  
모두 세로로  
출력됨(단어)



글자방향 변경



rot.per=0  
모두 가로로  
출력됨(단어)

---

### III. 대국민담화문 데이터 워드클라우드 시각화

---

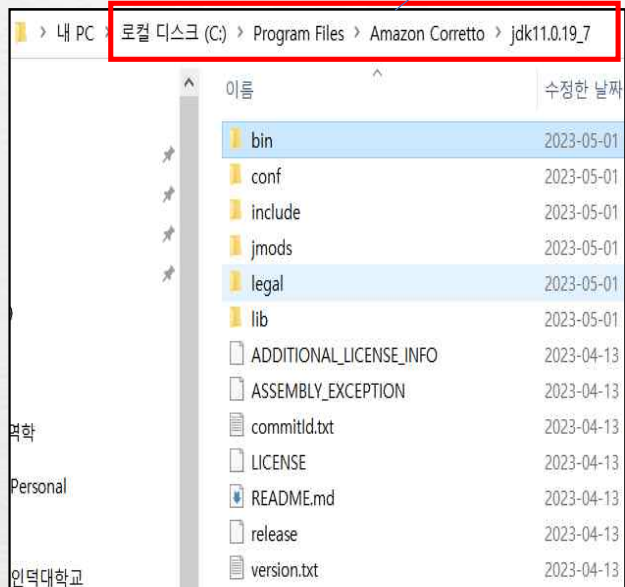
# 1. 패키지 확인

## Java설치 경로 확인

- KoNLP 패키지 설치를 하기 위해서는 JAVA가 필요하므로 JAVA가 설치된 폴더의 경로를 등록함

### CODE

```
>Sys.setenv(JAVA_HOME="C:/Program Files/Amazon Corretto/jdk11.0.19_7") # JAVA가 설치된 경로  
>library(wordcloud) # 워드클라우드  
>library(KoNLP) # 한국어 처리  
>library(RColorBrewer) # 색상 선택
```



## 2. 사전생성

한글사전 로딩 및 팔레트 생성

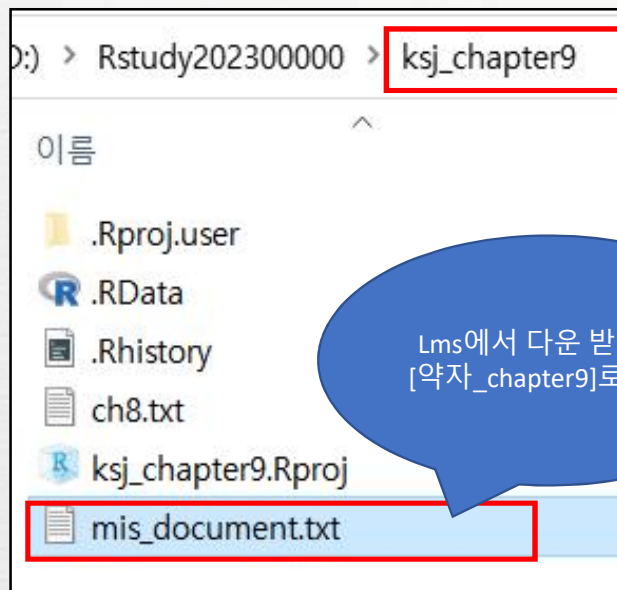
CODE

```
buildDictionary(ext_dic = "woorimalsam") # '우리말씀' 한글사전 로딩  
pal2 <- brewer.pal(8, "Accent")         # 팔레트 생성
```

### 3. 데이터불러오기와 명사추출

[mis\_document.txt]파일을 [약자\_chapter9]폴더에 드래그 한 후 명사 추출

```
CODE text <- readLines("mis_document.txt", encoding = "UTF-8" ) # 파일 읽기
noun <- extractNoun(text) # 명사 추출
noun # 추출된 명사 출력
```



```
> text <- readLines("mis_document.txt", encoding = "UTF-8" ) # 파일 읽기
>
> noun <- extractNoun(text)
>
> noun
[[1]]
 [1] "1"      "여성"   "가족"   "부"     "불법"   "촬영"   "등"     "디지털" "성범죄" "안전"
[11] "한"     "사회"   "국민"   "들"     "말"

[[2]]
[1] ""
```



## 4. 빈도수 높은 단어를 막대그래프로 작성하기

CODE

```
noun2 <- unlist(noun)           # 추출된 명사 통합
wordcount <- table(noun2)       # 단어 빈도수 계산
temp <- sort(wordcount, decreasing=T)[1:10] # 빈도수 높은 단어 10개 추출
temp
temp <- temp[-1]               # 공백 단어 제거
barplot(temp,                  # 막대그래프 작성
  names.arg = names(temp),    # 막대 이름을 단어로 표시
  col = "lightblue",          # 막대의 색상 지정
  main = "빈도수 높은 단어", ylab = "단어 빈도수")
```

```
> noun2 <- unlist(noun)           # 추출된 명사 통합
> wordcount <- table(noun2)       # 단어 빈도수 계산
> temp <- sort(wordcount, decreasing=T)[1:10] # 빈도수 높은 단어 10개 추출
> temp
noun2
  42  등  불법  여성  한  촬영  수사  할  경찰  성범죄
    27   27   24   23   21   18   12   11   11
> temp <- temp[-1]               # 공백 단어 제거
```

42의 빈도수를 가진 항목에 제목이 없음

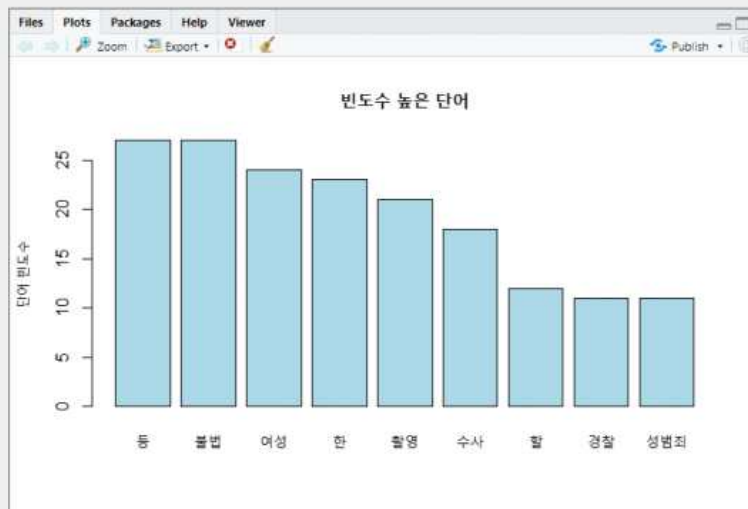
공백단어 제거

```
noun2
  등  불법  여성  한  촬영  수사  할  경찰  부
  27   27   24   23   21   18   12   11   11
```

## 4. 빈도수 높은 단어를 막대그래프로 작성하기

```
> barplot(temp,
+         names.arg = names(temp),
+         col = "lightblue",
+         main = "빈도수 높은 단어", ylab = "단어 빈도수")
>
```

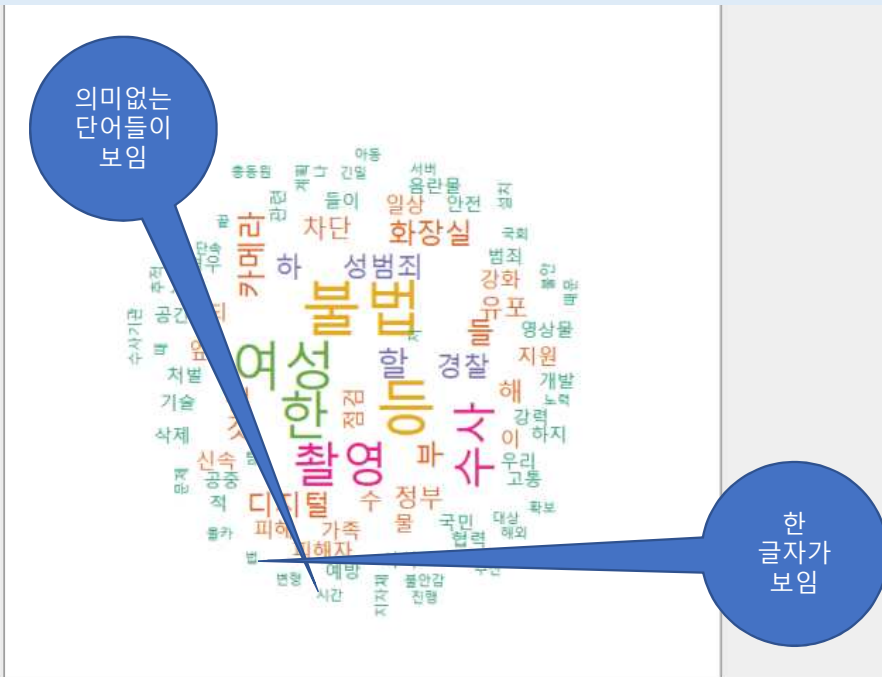
# 막대그래프 작성  
# 막대 이름을 단어로 표시  
# 막대의 색상 지정



## 5. 워드클라우드

### CODE

```
wordcloud(names(wordcount), # 단어들
           freq=wordcount,  # 단어들의 빈도
           scale=c(6,0.7),  # 단어의 폰트 크기
           min.freq=3,      # 단어의 최소 빈도
           random.order=F,  # 단어의 출력 위치
           rot.per=.1,      # 90도 회전 단어 비율
           colors=pal2)     # 단어의 색
```



- **names(wordcount)**

워드클라우드 상에 표시할 단어를 지정한다.

- **freq=wordcount**

워드클라우드 상에 표시할 단어의 빈도수를 지정한다.

- **scale=c(6,0.7)**

표시할 단어의 폰트 크기를 지정한다. 여기서 6은 폰트의 최대 크기, 0.7은 폰트의 최소 크기를 의미한다.

- **min.freq=3**

빈도수가 3 이상인 단어들만 표시한다.

- **random.order=F**

단어가 표시될 위치를 지정한다. T는 단어의 표시 위치를 무작위로 지정할 수 있고, F는 빈도수가 높은 단어일수록 중앙쪽에 배치된다.

- **rot.per=.1**

단어를 표시할 때 세로 방향으로 표시할 단어의 비율을 지정한다. 여기서 .1은 10%를 의미한다.

- **colors=pal2**

빈도수에 따라 pal2에 있는 색으로 단어의 색을 지정한다.

## 6. 데이터 가공하기

\* `gsub()` 함수를 이용해 텍스트 수정

### CODE

```
> noun2 <- unlist(noun)
# gsub 함수를 통해 텍스트 수정 실시
> noun2_2 <- gsub("하게", "", noun2)
> noun2_2 <- gsub("정현백입니다 ", " ", noun2_2)
> noun2_2 <- gsub("성평등한", "성평등", noun2_2)
> head(noun2_2, 30)
```

의미없는  
단어들이  
보임

```
> noun2
[1] "1"
[4] "부"
[7] "등"
[10] "안전"
[13] "국민"
[16] ""
[19] "여러분"
[22] "부"
[25] "우리사회"
[28] "운동"
[31] "화"
[34] "맞선"
[37] "연대"
[40] "성평등한"
[43] "사회"
[46] "하게"
"여성"
"불법"
"디지털"
"한"
"들"
"존경"
"여성"
"장관"
"들불"
"계기"
"폭력"
"여성"
"이"
"가족"
"촬영"
"성범죄"
"사회"
"말"
"국민"
"가족"
"정현백입니다"
"미투"
"일상"
"차별"
"들"
"이"
"우리"
"강력"
"일상"
```

한  
글자가  
보임

```
> head(noun2_2, 30)
[1] "1" "여성" "가족" "부" "불법" "촬영" "등" "디지털"
[9] "성범죄" "안전" "한" "사회" "국민" "들" "말" ""
[17] "존경" "국민" "여러분" "여성" "가족" "부" "장관" ""
[25] "우리사회" "들불" "미투" "운동" "계기" "일상"
```



## 6. 데이터 가공하기

### \* 글자 수 2개 이상인 단어만 선택

- Filter() 함수를 이용해 글자수가 2개 이상인 데이터만 선택하는 함수를 만들어 필터링

#### CODE

```
# 저장된 명사 중 두 글자 이상이 되는 것만 필터링
> noun2_3 <- Filter(function(x) {nchar(x) >=2}, noun2_2)
> head(noun2_3,30)
> wordcount <- table(noun2_2)
```

한  
글자는  
제외됨

```
> noun2_3 <- Filter(function(x) {nchar(x) >=2}, noun2_2)
> head(noun2_3,30)
[1] "여성" "가족" "불법" "촬영" "디지털" "성범죄" "안전" "사회"
[9] "국민" "존경" "국민" "여러분" "여성" "가족" "장관" "우리사회"
[17] "들불" "미투" "운동" "계기" "일상" "폭력" "차별" "맞선"
[25] "여성" "연대" "성평등" "세상" "우리" "사회"
```



## 7. 워드클라우드 수정

### CODE

```
par(mar=c(3,3,3,3))  
#워드클라우드가 잘 보이도록 여백 조정  
wordcloud(names(wordcount), # 단어들  
           freq=wordcount,  # 단어들의 빈도  
           scale=c(6,0.7),  # 단어의 폰트 크기  
           min.freq=3,      # 단어의 최소 빈도  
           random.order=F,  # 단어의 출력 위치  
           rot.per=.1,      # 90도 회전 단어 비율  
           colors=pal2)     # 단어의 색
```



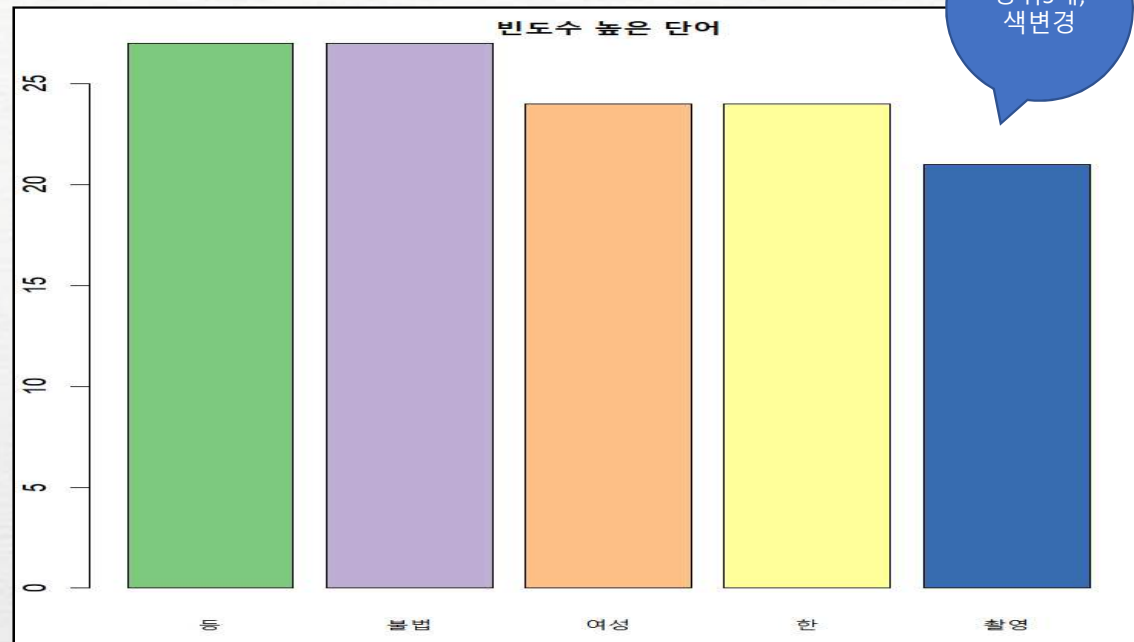
- **names(wordcount)**  
워드클라우드 상에 표시할 단어를 지정한다.
- **freq=wordcount**  
워드클라우드 상에 표시할 단어의 빈도수를 지정한다.
- **scale=c(6,0.7)**  
표시할 단어의 폰트 크기를 지정한다. 여기서 6은 폰트의 최대 크기, 0.7은 폰트의 최소 크기를 의미한다.
- **min.freq=3**  
빈도수가 3 이상인 단어들만 표시한다.
- **random.order=F**  
단어가 표시될 위치를 지정한다. T는 단어의 표시 위치를 무작위로 지정할 수 있고, F는 빈도수가 높은 단어일수록 중앙쪽에 배치된다.
- **rot.per=.1**  
단어를 표시할 때 세로 방향으로 표시할 단어의 비율을 지정한다. 여기서 .1은 10%를 의미한다.
- **colors=pal2**  
빈도수에 따라 pal2에 있는 색으로 단어의 색을 지정한다.

## 8. 그래프 수정하기

CODE

```
par(mar=c(3,3,3,3))  
pal2 <- brewer.pal(8, "Accent")  
temp <- sort(wordcount, decreasing=T)[1:6]  
temp  
temp <- temp[-1]  
barplot(temp,  
  names.arg = names(temp),  
  col = pal2,  
  main = "빈도수 높은 단어", ylab = "단어 빈도수")
```

#그래프가 잘 보이도록 여백 조정  
#막대색 변경  
# 빈도수 높은 단어 5개 추출  
  
# 공백 단어 제거  
# 막대그래프 작성  
# 막대 이름을 단어로 표시  
# 막대의 색상 지정



---

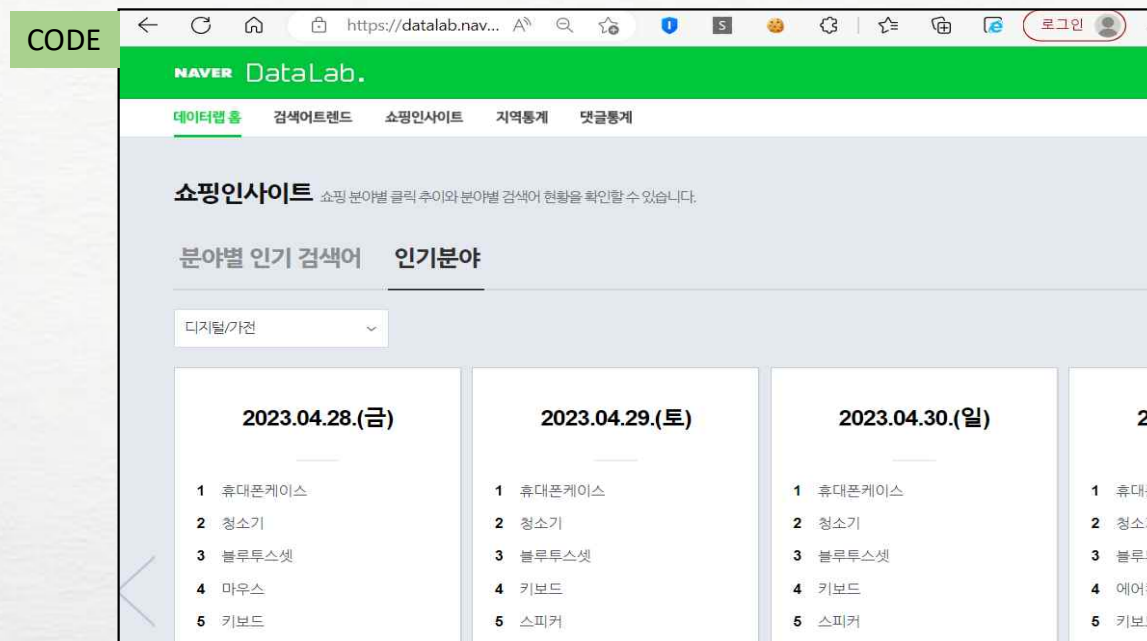
## IV. 인터넷 검색어 분석

---

# 1. 네이버 데이터랩

인터넷 검색어를 중심으로 사용자들의 관심사를 분석할 수 있도록 지원하는 많은 사이트들이 있음

- 네이버 데이터랩과 구글 트렌드가 대표적
- 네이버 데이터랩에서는 주로 국내의 관심사를 알아볼 수 있고, 구글 트렌드에서는 전 세계적인 관심사를 확인



네이버 데이터랩 초기화면(<http://datalab.naver.com/>)

## 2. 관심 키워드로 트렌드 분석

\* 관심있는 주제의 키워드를 입력

**CODE** **검색어트렌드** 네이버통합검색에서 특정 검색어가 얼마나 많이 검색되었는지 확인해보세요. 검색어를 기간별/연령별/성별/지역별/기기별로 분석할 수 있습니다.

궁금한 주제어를 설정하고, 하위 주제어에 해당하는 검색어를 콤마(,)로 구분입력해 주세요. 입력한 단어의 추이를 하나씩 클릭하여 상세히 살펴보실 수 있습니다. 예) 주제어 캠핑 : 캠핑, Camping, 캠핑용품, 겨울캠핑, 캠핑장, 글램핑, 오토캠핑, 캠핑카, 캠핑장

주제어1  주제어 1에 해당하는 모든 검색어를 콤마(,)로 구분하여 최대 20개까지 입력

주제어2  주제어 2에 해당하는 모든 검색어를 콤마(,)로 구분하여 최대 20개까지 입력

기간

2023년 04월 04일 ~ 2023년 05월 04일  
\* 2016년 1월 이후 조회할 수 있습니다.

범위 ☐ 전체 ☐ 모바일 ☐ PC

성별 ☐ 전체 ☐ 여성 ☐ 남성

연령선택 ☐ 전체

☐ -12 ☐ 13-18 ☐ 19-24 ☐ 25-29 ☐ 30-34 ☐ 35-39 ☐ 40-44 ☐ 45-49 ☐ 50-54 ☐ 55-59 ☐ 60-

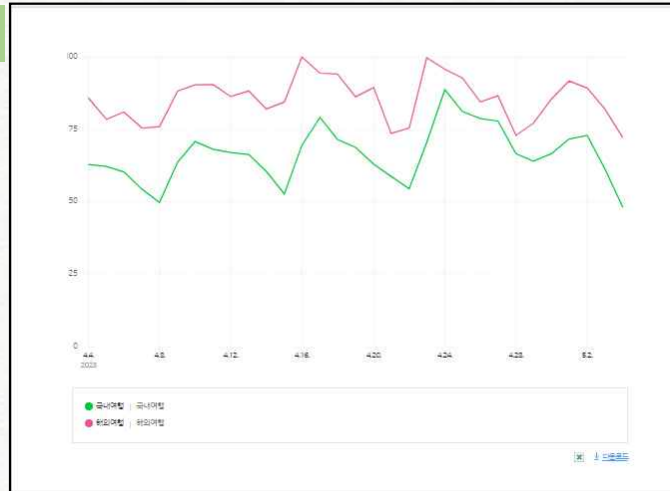
[네이버 검색 데이터 조회](#)

키워드를 통한 검색어 트렌드 조회



## 2. 관심 키워드로 트렌드 분석

CODE



‘국내여행’과 ‘해외여행’에 대한 검색어 트렌드

- 전반적으로 해외여행 보다는 국내여행에 대한 검색 비중이 높음
- 특히, 여름휴가가 다가오는 4월 24일에 검색 횟수가 급증

### 3. 주제어와 기간으로 분석

- 주제어와 기간, 범위, 성별 등을 입력하고 데이터를 조회

CODE

← → ↺ 🏠

https://datalab.nav... A 🔍 ☆

S 🍪 ⚙️ | ☆ 📁 🌐

로그인

궁금한 주제어를 설정하고, 하위 주제어에 해당하는 검색어를 콤마(,)로 구분입력해 주세요. 입력한 단어의 추이를 하나로 합산하여 해당 주제가 네이버에서 얼마나 검색되는지 조회할 수 있습니다. 예) 주제어 캠핑 : 캠핑, Camping, 캠핑용품, 겨울캠핑, 캠핑장, 글램핑, 오토캠핑, 캠핑카, 텐트, 캠핑요리

주제어1	빅데이터	빅데이터
주제어2	웹	웹
주제어3	앱	앱
주제어4	주제어 4 입력	주제어 4에 해당하는 모든 검색어를 콤마(,)로 구분하여 최대 20개까지 입력
주제어5	주제어 5 입력	주제어 5에 해당하는 모든 검색어를 콤마(,)로 구분하여 최대 20개까지 입력

기간

전체

1개월

3개월

1년

직접입력

일간

2022

05

04

-

2023

05

04

· 2016년 1월 이후 조회할 수 있습니다.

범위

☒ 전체
 ☒ 모바일
 ☒ PC

성별

☒ 전체
 ☒ 여성
 ☒ 남성

연령선택

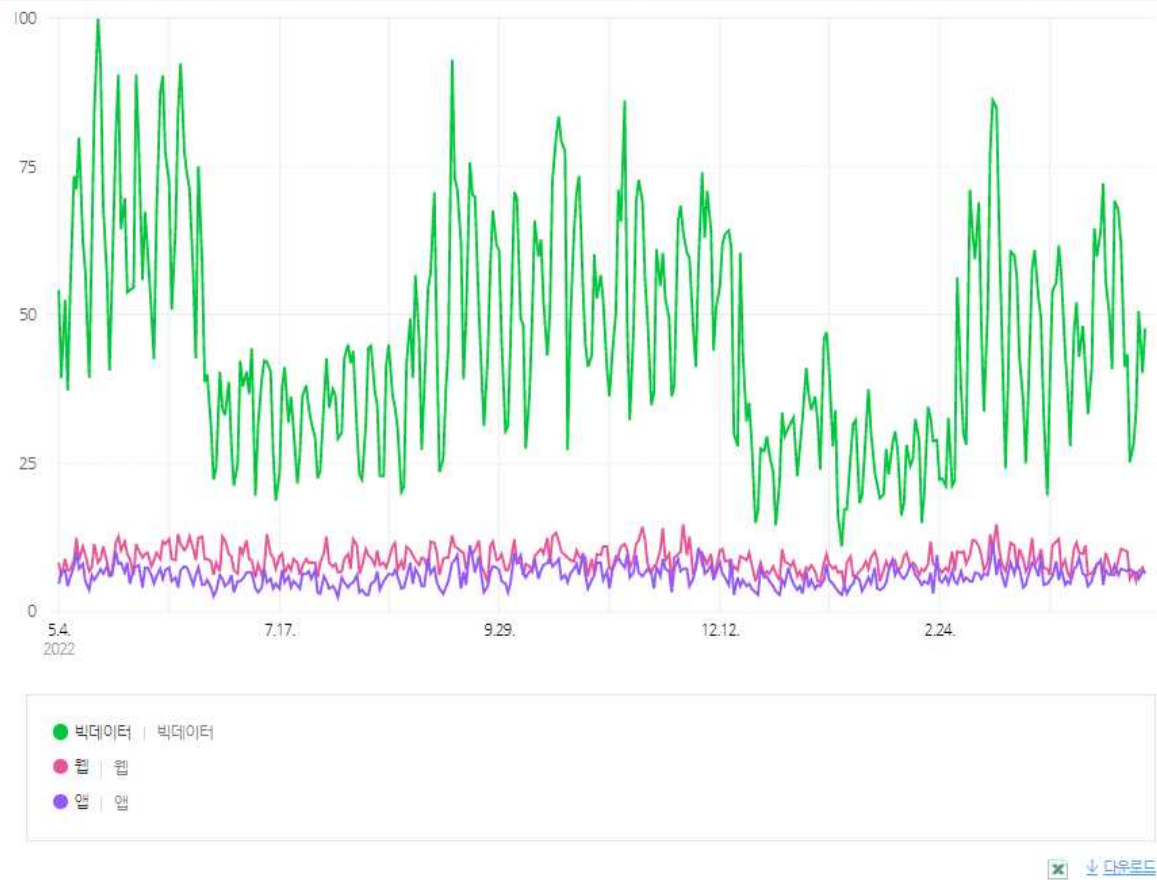
☐ 전체
 ☐ ~12
 ☐ 13~18
 ☒ 19~24
 ☒ 25~29
 ☐ 30~34
 ☐ 35~39
 ☐ 40~44
 ☐ 45~49
 ☐ 50~54
 ☐ 55~59
 ☐ 60~

네이버 검색 데이터 조회

### 3. 주제어와 기간으로 분석

\* 그래프로 분석 결과: 빅데이터에 대한 관심이 매우 높아지고 있음을 알 수 있음

CODE



---

## V. 워드클라우드 응용

---

# 1. 주제어 선택

실습1\_주제어를 선택한 화면을 캡처

네이버 데이터랩 (<http://datalab.naver.com/>)

=> 2022년 핫했던 주제어 하나를 선택

The screenshot shows the Naver DataLab interface. At the top, there's a green header with 'NAVER DataLab.' and a navigation bar with links: '데이터랩 홈', '검색어트렌드', '쇼핑인사이드', '지역통계', and '댓글통계'. The main section is titled '쇼핑인사이드' with a subtitle '쇼핑 분야별 클릭 추이와 분야별 검색어 현황을 확인할 수 있습니다.' Below this, there are tabs for '분야별 인기 검색어' and '인기분야'. A dropdown menu shows '여가/생활편의' and a '월간' button. The content is organized into four columns for the months of 2023: February, March, April, and May. Each column lists the top 10 trending search terms.

2023.02. 2023.02.01. ~ 2023.02.28.	2023.03. 2023.03.01. ~ 2023.03.31.	2023.04. 2023.04.01. ~ 2023.04.30.	2023.05. 2023.05.01. ~ 2023.05.01.
1 화환	1 화환	1 문화상품권	1 문화상품권
2 근조화환	2 근조화환	2 화환	2 카네이션화분
3 꽃배달	3 개업화분	3 일본유심	3 카네이션
4 개업화분	4 일본유심	4 티니핑뮤지컬	4 카네이션생화
5 축하화환	5 부산요트투어	5 부산요트투어	5 일본유심
6 비발디파크리프트권	6 문화상품권	6 신세계상품권	6 컬처랜드문화상품권
7 결혼식화환	7 베트남유심	7 근조화환	7 컬처랜드
8 꽃바구니	8 태국유심	8 피카소와20세기저장들	8 부산요트투어
9 개업화환	9 에어컨청소	9 컬처랜드	9 카네이션꽃다발
10 일본유심	10 꽃바구니	10 롯데월드자유이용권	10 신세계상품권

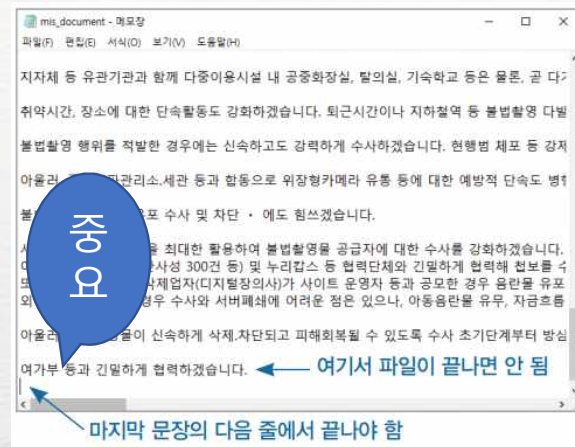


## 2. 워드클라우드 문서 파일 준비

실습2\_메모장 화면 캡처

선택한 주제어를 사용하여 관련된 내용을 인터넷에서 찾아서 메모장에 복사하여 붙임

- 파일의 끝부분 처리를 아래 왼쪽 그림과 같이 마지막 문장이 끝나면 반드시 줄 바꿈을 한 후 저장
- 파일을 저장할 때, [다른 이름으로 저장]을 선택하고 아래 오른쪽 그림과 같이 인코딩을 'UTF-8'로 선택을 하여 저장
- 파일 이름이나 파일이 저장된 폴더 경로에 한글이 포함되어 있으면 파일을 읽을 때 에러가 발생하는 경우가 있으므로 파일을 저장할 때는 파일 이름을 영어로 설정



텍스트 파일 끝부분에서의 줄 바꿈



UTF-8로 텍스트 파일 저장

## 2. 워드클라우드 문서 파일 준비

빅데이터의 불편한 진실  
전 세계가 빅데이터에 열광하는 이유는 이 때문이다. 강화, 최적의 정책 결정으로

로봇이 기사를 쓴다고?  
2012년 4월 하순 미국 미디어 업계에 깜짝 놀라움을 선사한 기사 20여 명을 해고하고 저널리스트(Jo

저널리스트의 고객은 시카고 트리뷴뿐만이 아니다. 포브스(Forbes)도 마찬가지이다. 저널리스트는 내러티브 사언론의 기술을 사용하는데 내러티브 사언

인터넷을 통해 다양한 통계 수치를 얻을 수 있는 분야라서일까, 저널리스트는 현재 스포츠와 비즈니스 분야의 기사만 다룬다. 그러나 IT전문지인 《와이어드》에

구글, 애플, 마이크로소프트의 음성인식과 외국어 번역 기술이 접목돼 발전하면 더는 외국어 통·번역자가 필요 없듯이 인간의 고유 영역으로 생각되어 왔

시카고 트리뷴은 지역 신문 위기에 5년간 온라인과 지면 미디어에 적지 않은 투자를 해왔다. 그러나 자사의 지역 신문사인 트리브로컬(TRIBlocal)의 기자,

당시 시카고 트리뷴의 편집장은 “이것이 지역 소식을 더욱 많이 제공할 수 있는 효율적인 방법이 될 것으로 믿는다”고 밝혔다. 기가움은 “로봇과 알고리즘

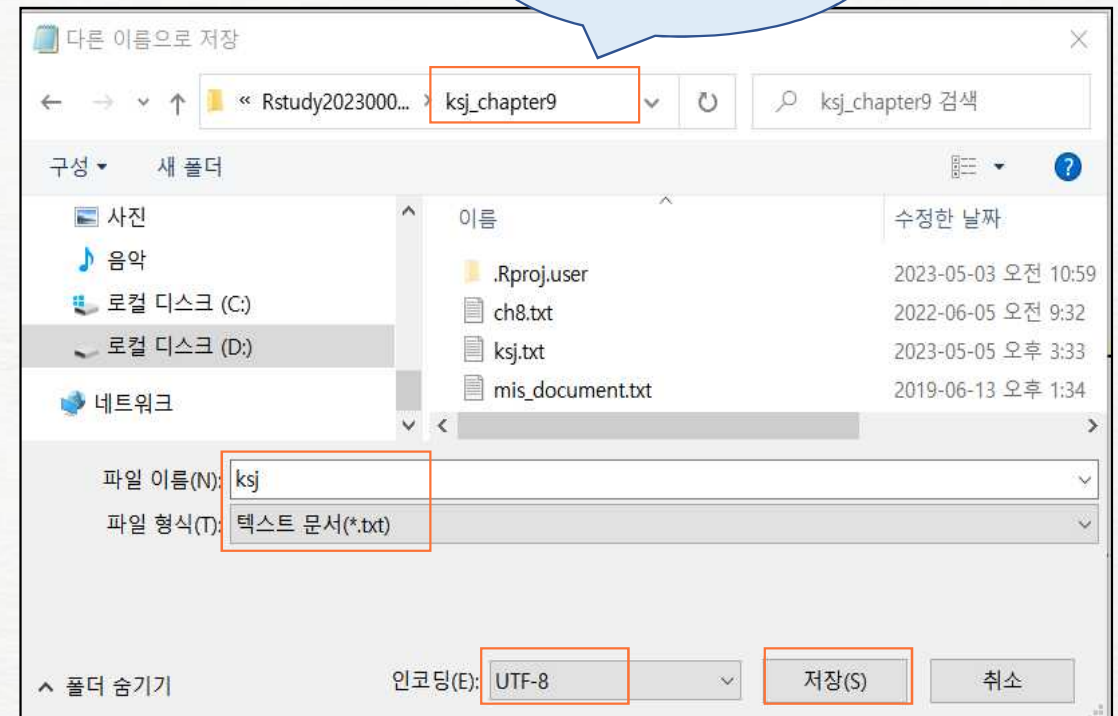
물론 저널리스트가 완전히 시스템에 의해서만 운영되지는 않는다. 저널리스트는 기본적으로 프리랜서를 채용해 시스템에서 부족한 부분을 채우고 있는데 저널리스트의

빅데이터 분석, BI와 IT 전문가 의존도를 낮춘다

내러티브 사언론의 기술 활용사례는 최근 미국 보스턴에서 개최된 ‘엔터프라이즈 2.0 보스턴 2012’ 컨퍼런스에서도 인용되었다. MIT의 수석 리서치 사

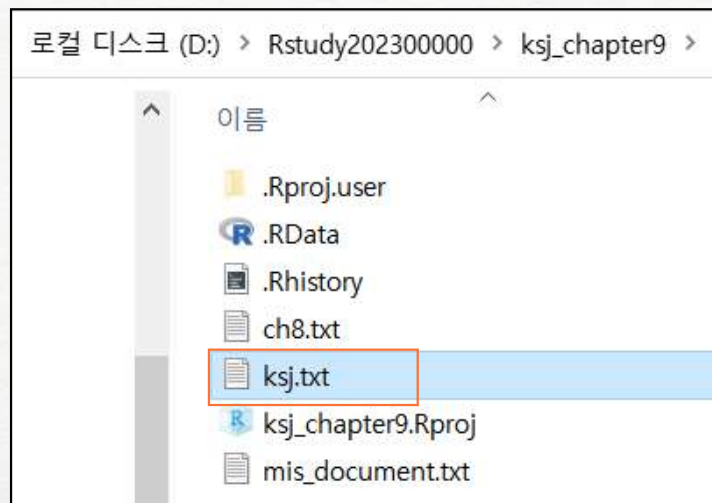
인포메이션위크 산하 브레인야드(<http://www.informationweek.com/social.asp>)는 앤드류 매카피 박사의 강연 내용을 소개했는데, 앤드류 매카피 박사는

인터넷에서  
주제어에 관련된  
자료를 찾아서  
메모장에 복사하여  
붙여넣기



## 2. 워드클라우드 문서 파일 준비

- [약자\_chapter9]폴더에 저장된 약자.txt의 파일 확인



### 3. 워드클라우드

실습3\_(가)~(바) 명령어 쓰기

(가) [약자.txt]파일을 [약자\_txt]의 데이터로 셋팅

```
약자_txt <- readLines(      )
```

(나) 약자\_txt의 내용 보기

```
_____ (약자_txt)
```

(다) 약자\_txt의 명사를 추출

```
n약자 <- _____ (약자_txt)
```

(라) [n약자] 확인

```
____ (n약자)
```

(마) [n약자]를 일차원 벡터로 만들기

```
c약자 <- _____
```

(바) gsub()를 이용하여 의미없는 단어 삭제(gsub()를 3개이상 사용하기)

예)c약자2 <- gsub( )

```
      c약자2 <- gsub(      )
```

```
      c약자2 <- gsub(      )
```

### 3. 워드클라우드

실습4\_(사)~(타) 명령어 쓰기

(사) 2글자 이상의 단어만 필터링

```
c약자3 <- Filter( , c약자2)
```

(아) [c약자3]의 빈도 구하기

```
wordcnt약자 <-
```

(자) [wordcnt약자]를 내림차순으로 정렬

```
sort( )
```

(차) 팔레트 지정

```
library(RColorBrewer)
```

```
Set2 <-
```

# “Dark2” 외 다른 파스텔톤으로 변경

(카) 여백 지정

```
Library(wordcloud)
```

```
par( =c(0,0,0,0))
```

#워드클라우드가 잘 보이도록 조정함

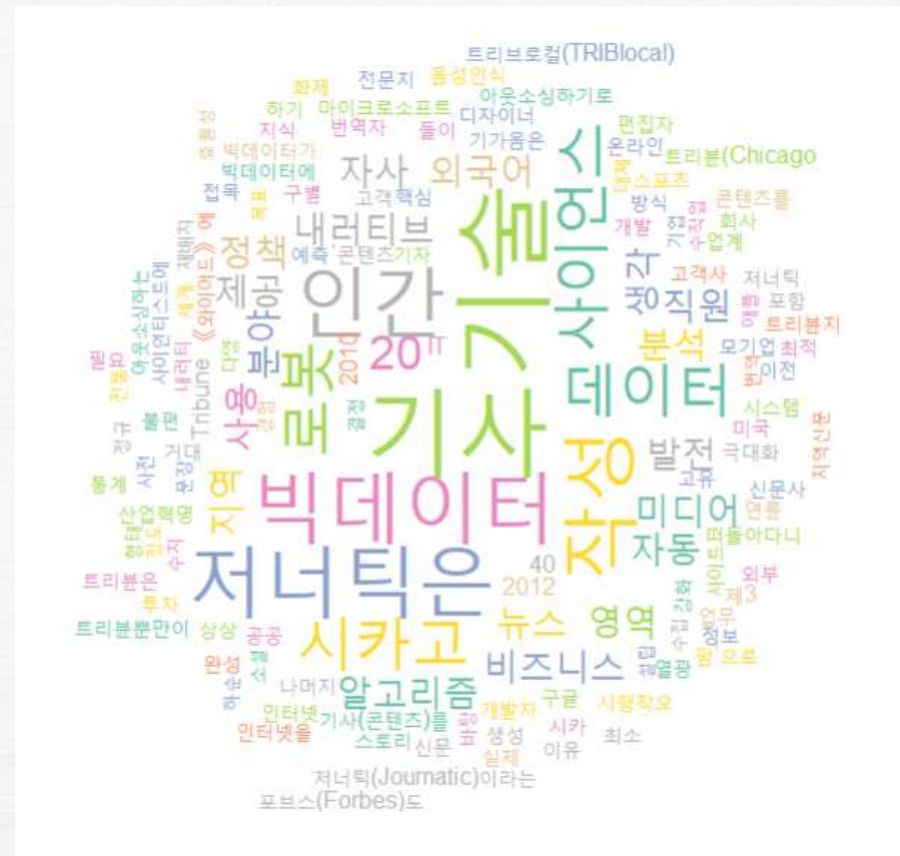
(타) 워드클라우드(글자크기 3에서 시작해서 0.3까지, 글자방향 가로 25%)

```
wordcloud(names(wordcnt약자), _____, _____ ,  
_____, min.freq=1, random.order=F, random.color=T, _____ )
```



### 3. 워드클라우드

실습5\_워드클라우드 화면캡처



## 4. 그래프 그리기

실습6\_(가)~(다)명령어를 쓰고, 그래프 화면캡처

(가) [wordcnt약자]를 내림차순으로 정렬한 후 상위 10개의 단어를 temp약자에 저장

temp약자 <- sort(

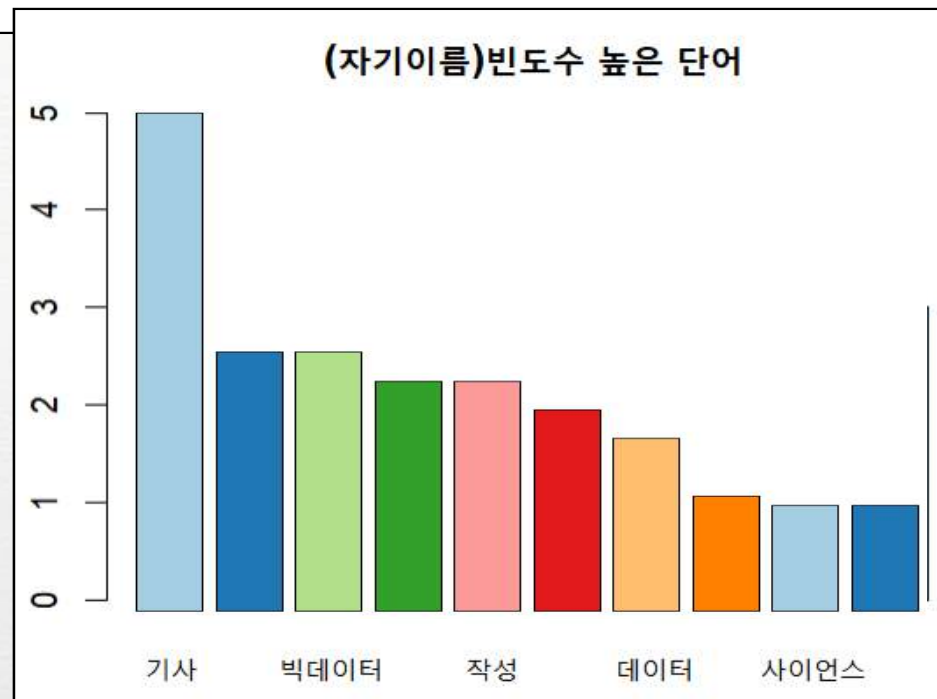
(나) 여백지정

par(mar=c(3,3,3,3)) #그래프가 잘 보이도록 조정함

(다) 그래프 그리기

pal2 <- brewer.pal( \_\_\_\_\_ ) #막대색변경

barplot(temp약자, names.arg = names(temp약자), \_\_\_\_\_, main = "(자기이름)빈도수 높은 단어", ylab = "단어 빈도수")



오늘도 잘했어요 🍷