

텍스트 분석을 위한 R

LG 인화원

강사: 박찬엽

2019년 04월 24일

목차

1 텍스트 관련 R 패키지 설치 가이드	1
1.1 windows 사용자	1
1.1.1 jdk 8 설치	1
1.1.2 KoNLP 패키지 설치 및 테스트	3
1.1.3 RcppMeCab 패키지 설치 및 테스트	4
1.2 macOS 사용자	5
1.2.1 터미널에서 설치해야 하는 도구들	5
1.2.2 KoNLP 패키지 설치 및 테스트	7
1.2.3 RcppMeCab 패키지 설치 및 테스트	7
1.3 공통 패키지 설치	8
2 사전 지식	9
2.1 tidyverse	9
2.1.1 파이프 연산자(%>%)	10
2.1.2 단정한 데이터(tidy data)	12
2.1.3 long form과 wide form	14
2.2 데이터 패키지(presidentSpeech) 소개	14
2.2.1 조건 확인 함수	15
2.2.2 연설문 텍스트 가져오기	16
2.2.3 연습문제	17
3 단정하게 텍스트를 다루는 tidytext	19
3.1 tidy text data	19
3.2 tidytext 패키지	19
3.2.1 token 단위 처리 unnest_tokens()	20
3.2.2 연습문제	21

4.1 한글의 특징 형태소23
4.2 형태소 분석기25
4.2.1 R의 대표적인 형태소 분석기25
4.2.2 생각해보기27
4.2.3 형태소로 token화27
4.2.4 불용어 제거31
4.2.5 정규 표현식33
4.2.6 연습문제34
4.2.7 RcppMeCab 실습35
5 텍스트 마이닝 지표39
5.1 단어 출현 빈도39
5.1.1 단어 출현 빈도 계산39
5.1.2 사용예 : 워드클라우드40
5.1.3 연습문제42
5.2 동시 출현 빈도43
5.2.1 동시 출현 빈도 계산43
5.2.2 기준 단어로 데이터 탐색44
5.2.3 연습문제45
5.2.4 사용예 : 네트워크 시각화46
5.3 tf-idf48
5.3.1 tf-idf 계산48
5.3.2 연습문제49
5.4 감성 분석50
5.4.1 사전 소개51
5.4.2 감성 분석 점수52

1 텍스트 관련 R 패키지 설치 가이드

1.1 windows 사용자

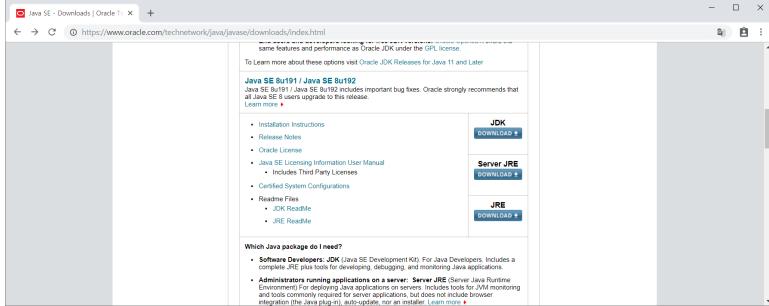
windows 10 64bit, windows 7 32bit에서 테스트하였습니다.

1.1.1 jdk 8 설치

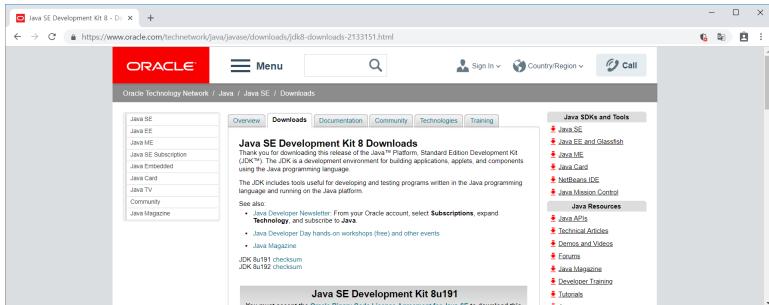
oracle 홈페이지 (<https://www.oracle.com/technetwork/java/javase/downloads/index.html>) 방문



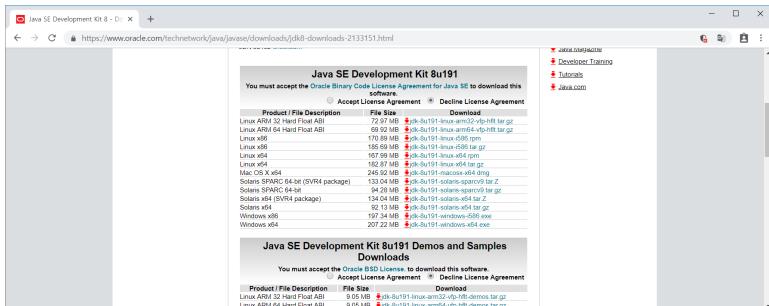
아래로 스크롤 하여 Java SE 8uXXX 에서 JDK의 다운로드 클릭



jdk 8 다운로드를 위한 화면



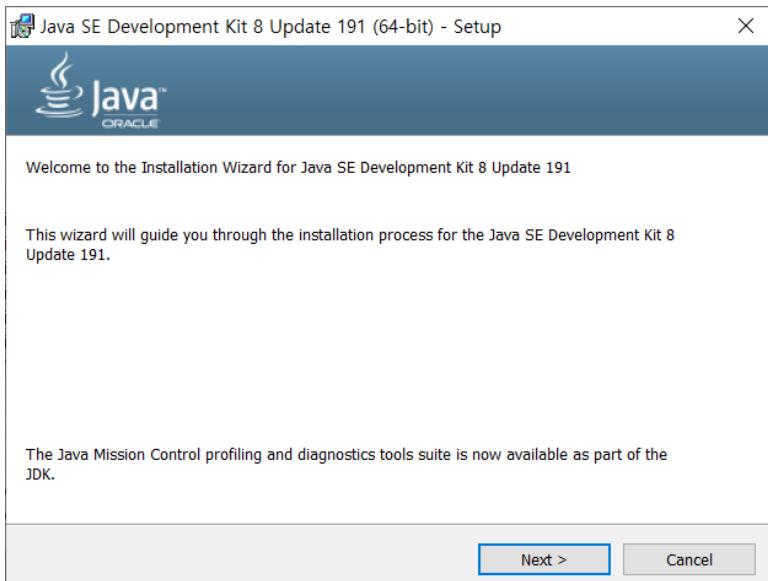
Java SE Development Kit 8uXXX 위치에 Accept License Agreement 클릭



x64가 64bit, x86이 32bit를 의미하므로 컴퓨터의 bit에 맞는 버전을 다운로드



설치 파일을 실행 후 다른 설정없이 Next를 계속 클릭하여 설치를 진행



1.1.2 KoNLP 패키지 설치 및 테스트

1.1.2 KoNLP 패키지 설치 및 테스트

KoNLP는 가장 유명한 형태소 분석기입니다.

```
if (!requireNamespace("KoNLP")) {
  install.packages("KoNLP", repos = "https://cloud.r-project.org")
}
library(KoNLP)

test <- "한글 테스트 입니다."
# 아래 결과가 나와야 합니다.
extractNoun(test)
#> [1] "한글"   "테스트"
```

1.1.3 RcppMeCab 패키지 설치 및 테스트

RcppMeCab은 빠른 형태소 분석기입니다.

```
if (!requireNamespace("RcppMeCab")) {
  install.packages("RcppMeCab", repos =
"https://cloud.r-project.org")
}
if (!requireNamespace("RmecabKo")) {
  install.packages("RmecabKo", repos =
"https://cloud.r-project.org")
}
#> Loading required namespace: RmecabKo

# c에 권한이 없다면 "d:/mecab"으로 설정
RmecabKo::install_mecab("c:/mecab")

library(RcppMeCab)
test <- "한글 테스트 입니다."
```

```
#> $`<c7><U+0471><db> <c5><U+05FD><U+00BA><U+01AE> <c0>
<U+0534><U+03F4><d9>.
#> [1] "<c7><d1>/SL" "<U+00B1><db> /SY"
#> [3] "<c5><d7>/SL" "<U+00BD><U+00BA>/SY"
#> [5] "<U+01AE>/SL" "<c0><U+0534>/SY"
#> [7] "<U+03F4>/SL" "<d9>./SY"
```

만약에 글자가 깨진다면 `iconv()` 함수를 사용해 보세요.

```
library(RcppMeCab)
test <- "한글 테스트 입니다."
# iconv 함수는 인코딩을 변경하는 함수입니다.
test <- iconv(test, to = "UTF-8")
pos(test)
#> $`한글 테스트 입니다.
#> [1] "한글/NNG"           "E//스트/NNG"
#> [3] "입니다/VCP+EF" ".SF"
```

1.2 macOS 사용자

1.2.1 터미널에서 설치해야 하는 도구들

1.2.1.1 JDK8

동영상 가이드를 참고하여 설치해주세요.

<https://www.youtube.com/watch?v=v8xZWbIASc0>

```
java -version
```

버전이 1.8.XXXXXX로 표시되면 잘 설치된 것입니다.

1.2.1.2 mecab-ko

형태소 분석기를 설치합니다.

공식 문서를 참고하세요.

<https://bitbucket.org/eunjeon/mecab-ko-dic>

맥의 터미널에서 아래 명령어를 수행합니다.

```
wget https://bitbucket.org/eunjeon/mecab-
ko/downloads/mecab-0.996-ko-0.9.2.tar.gz
tar zxvf mecab-0.996-ko-0.9.2.tar.gz
cd mecab-0.996-ko-0.9.2
./configure
make
make check
# 아래 실행 후 컴퓨터 비밀번호 입력
sudo make install
```

1.2.1.3 3. mecab-ko-dic

형태소 분석기의 사전을 설치합니다.

공식 문서를 참고하세요.

<https://bitbucket.org/eunjeon/mecab-ko-dic>

맥의 터미널에서 아래 명령어를 수행합니다.

```
wget https://bitbucket.org/eunjeon/mecab-ko-
dic/downloads/mecab-ko-dic-2.1.1-20180720.tar.gz
tar zxvf mecab-ko-dic-2.1.1-20180720.tar.gz
cd mecab-ko-dic-2.1.1-20180720
./configure
make
make check
```

```
# 아래 실행 후 컴퓨터 비밀번호 입력  
sudo make install
```

1.2.2 KoNLP 패키지 설치 및 테스트

KoNLP는 가장 유명한 형태소 분석기입니다.

```
if (!requireNamespace("KoNLP")) {  
  install.packages("KoNLP", repos = "https://cloud.r-project.org")  
}  
library(KoNLP)  
  
test <- "한글 테스트 입니다."  
extractNoun(test)  
#> [1] "한글" "테스트"
```

1.2.3 RcppMeCab 패키지 설치 및 테스트

RcppMeCab은 빠른 형태소 분석기입니다.

```
if (!requireNamespace("remotes")) {  
  install.packages("remotes", repos =  
    "https://cloud.r-project.org")  
}  
if (!requireNamespace("RcppMeCab")) {  
  remotes::install_github("junhewk/RcppMeCab")  
}  
library(RcppMeCab)  
  
test <- "한글 테스트 입니다."  
pos(test)  
#> $<c7><U+0471><db> <c5><U+05FD><U+00BA><U+01AE> <c0>  
#> <U+0534><U+03F4><d9>.  
#> [1] "<c7><d1>/SL" "<U+00B1><db> /SY"  
#> [3] "<c5><d7>/SL" "<U+00BD><U+00BA>/SY"
```

만약에 글자가 깨진다면 `iconv()` 함수를 사용해 보세요.

```
# iconv 함수는 인코딩을 변경하는 함수입니다.
test <- iconv(test, to = "UTF-8")
pos(test)
#> $`한글 테스트 입니다.`
#> [1] "한글/NNG"      "Ell스트/NNG"
#> [3] "입니다/VCP+EF" ".SF"
```

1.3 공통 패키지 설치

`remotes` 패키지는 패키지 인스톨을 위한 함수를 제공합니다.
`tidytext` 패키지는 텍스트를 tidy하게 다룰 수 있게 해줍니다.
`presidentSpeech` 패키지는 역대 대통령 연설문 텍스트를 사용할 수 있게 해주는 패키지입니다.

```
if (!requireNamespace("remotes")) {
  install.packages("remotes", repos =
  "https://cloud.r-project.org")
}
if (!requireNamespace("tidytext")) {
  install.packages("tidytext", repos =
  "https://cloud.r-project.org")
}
if (!requireNamespace("presidentSpeech")) {
  remotes::install_github("forkonlp/presidentSpeech")
}

library(tidytext)
library(presidentSpeech)
```

위 패키지를 불러오는 것이 동작하는 것을 확인해주세요.

2 사전 지식

2.1 tidyverse



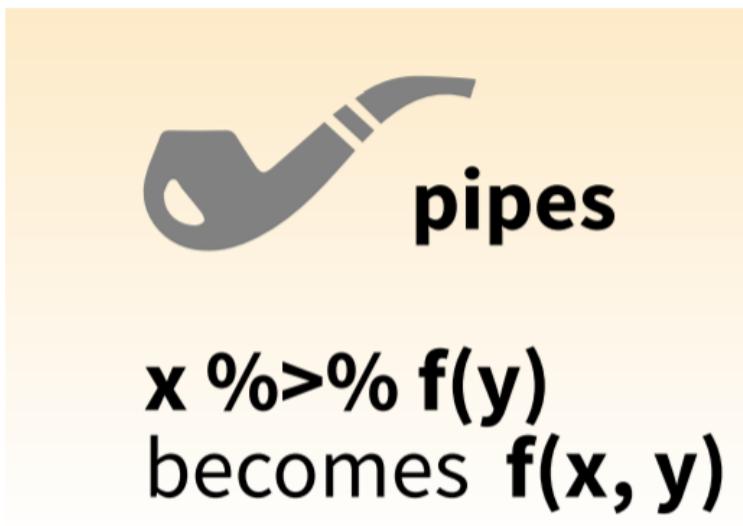
tidyverse 패키지는

1. RStudio가 개발, 관리하는 패키지
2. 공식 문서가 매우 잘 되어 있음
3. 사용자층이 두터워 영어로 검색하면 많은 질답을 찾을 수 있음
4. 커뮤니티 설명글도 매우 많음
5. 6개의 핵심 패키지 포함 23가지 패키지로 이루어진 메타 패키지
6. tidy data 라는 사상과 파이프 연산자로 대동단결
7. 사상에 영감을 받아 맞춰서 제작하는 개인 패키지가 많음(ex> [tidyquant \(<https://github.com/business-tidyquant/tidyquant>\)](https://github.com/business-tidyquant/tidyquant)

, [tidytext](https://github.com/juliasilge/tidytext) (<https://github.com/juliasilge/tidytext>)
등)

```
if (!requireNamespace("tidyverse")) {  
  install.packages("tidyverse")}  
library(tidyverse)
```

2.1.1 파이프 연산자(%>%)



함수를 중첩해서 사용할 일이 점점 빈번해짐

```
plot(diff(log(sample(rnorm(10000, mean = 10, sd = 1),  
size = 100, replace = FALSE))), col = "red", type =  
"l")
```

%>%를 사용하면

1. 생각의 순서대로 함수를 작성할 수 있음
2. 중간 변수 저장을 할 필요가 없음
3. 순서가 읽이 용이하여 기억하기 좋음

```
rnorm(10000,mean=10,sd=1) %>%  
  sample(size=100,replace=FALSE) %>%  
  log %>%  
  diff %>%  
  plot(col="red",type="l")
```

flights 데이터에 파이프 연산자 사용 예 1

```
flights %>%  
  group_by(year,month,day) %>%  
  summarise(delay = mean(dep_delay, na.rm = TRUE))  
#> # A tibble: 365 x 4  
#>   year month day delay  
#>   <int> <int> <int> <dbl>  
#> 1 2013     1     1  11.5  
#> 2 2013     1     2  13.9  
#> 3 2013     1     3  11.0  
#> 4 2013     1     4   8.95  
#> 5 2013     1     5   5.73  
#> 6 2013     1     6   7.15  
#> # ... with 359 more rows
```

group_by()는 filter()와도 함께 사용할 수 있음

```
flights %>%  
  group_by(dest) %>%  
  filter(n() > 365) ->  
  popular_dests  
popular_dests  
#> # A tibble: 332,577 x 19
```

```
#>   <int> <int> <int>   <int>
#> 1 2013    1    1    517
#> 2 2013    1    1    533
#> 3 2013    1    1    542
#> 4 2013    1    1    544
#> 5 2013    1    1    554
#> 6 2013    1    1    554
#> # ... with 3.326e+05 more rows, and 15
#> #   more variables:
#> #     sched_dep_time <int>,
#> #     dep_delay <dbl>, arr_time <int>,
#> #     sched_arr_time <int>,
#> #     arr_delay <dbl>, carrier <chr>,
#> #     flight <int>, tailnum <chr>,
#> #     origin <chr>, dest <chr>,
#> #     air_time <dbl>, distance <dbl>,
#> #     hour <dbl>, minute <dbl>,
#> #     time_hour <dttm>
```

사용할 데이터부터 순서대로 함수를 작성할 수 있는 장점

```
popular_dests %>%
  filter(arr_delay > 0) %>%
  mutate(prop_delay = arr_delay / sum(arr_delay)) %>%
  select(year:day, dest, arr_delay, prop_delay)
#> # A tibble: 131,106 x 6
#> # Groups:   dest [77]
#>   year month day dest arr_delay
#>   <int> <int> <int> <chr>     <dbl>
#> 1 2013    1    1 IAH      11
#> 2 2013    1    1 IAH      20
#> 3 2013    1    1 MIA      33
#> 4 2013    1    1 ORD      12
#> 5 2013    1    1 FLL      19
#> 6 2013    1    1 ORD       8
#> # ... with 1.311e+05 more rows, and 1
#> #   more variable: prop_delay <dbl>
```

2.1.2 단정한 데이터(tidy data)

2.1.2 단정한 데이터(tidy data)

1. [Hadley Wickham \(<https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html>\)](https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html).
2. [고감자님의 블로그 \(<http://freesearch.pe.kr/archives/3942>\)](http://freesearch.pe.kr/archives/3942).
3. [헬로우데이터과학 \(<http://www.hellobadascience.com/?p=287>\)](http://www.hellobadascience.com/?p=287).

1.1 Each variable forms a column.

1.2 각 변수는 개별의 열(column)으로 존재한다.

1.3 각 열에는 개별 속성이 들어간다.

2.1 Each observation forms a row.

2.2 각 관측치는 행(row)를 구성한다.

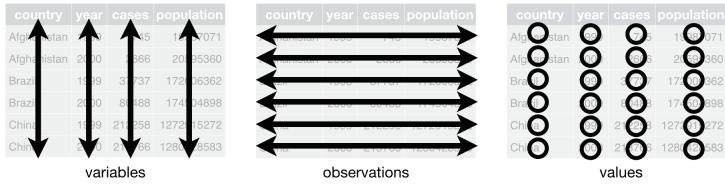
2.3 각 행에는 개별 관찰 항목이 들어간다.

3.1 Each type of observational unit forms a table.

3.2 각 테이블은 단 하나의 관측기준에 의해서 조직된 데이터를 저장한다.

3.3 각 테이블에는 단일 유형의 데이터가 들어간다.

* 출처 : [금융데이터 분석을 위한 R 입문 \(\[https://mrchypark.github.io/kisa_finR\]\(https://mrchypark.github.io/kisa_finR\)\)](https://mrchypark.github.io/kisa_finR)



* 출처 : [Garrett Grolemund의 Data Science with R 블로그](http://garrettgman.github.io/tidying/) (<http://garrettgman.github.io/tidying/>).

2.1.3 long form과 wide form

long form

1. 컴퓨터가 계산하기 좋은 모양
2. tidy data의 요건을 충족
3. tidyverse의 패키지 대부분의 입력 형태

wide form

1. 사람이 눈으로 보기 좋은 모양
2. 2개 변수에 대한 값만 확인 가능
3. dashboard 형이라고도 하며 조인 등 연산이 어려움

2.2 데이터 패키지 (presidentSpeech) 소개

대통령 기록 연구실의 대통령 연설문을 제공(패키지 설명 (<https://forkonlp.github.io/presidentSpeechKr/>))

```
if (!requireNamespace("presidentSpeech")) {  
  remotes::install_github("forkonlp/presidentSpeech")  
}  
library(presidentSpeech)
```

2.2.1 조건 확인 함수

대통령 조건 확인

```
get_president()  
#> [1] "이승만" "윤보선" "박정희" "최규하"  
#> [5] "전두환" "노태우" "김영삼" "김대중"  
#> [9] "노무현" "이명박"
```

연설 분야 조건 확인

```
get_field()  
#> [1] "국정전반"      "정치/사회"  
#> [3] "산업/경제"     "외교/통상"  
#> [5] "국방"           "과학기술정보"  
#> [7] "교육"           "문화/체육/관광"  
#> [9] "환경"           "기타"
```

연설 유형 확인

```
get_event()  
#> [1] "취임사"        "신년사"  
#> [3] "국회연설"      "기념사"  
#> [5] "만찬사"        "환영사"  
#> [7] "치사"          "성명/담화문"  
#> [9] "라디오연설"    "기타"
```

연설 리스트 데이터

```
library(dplyr)
data(spidx)
glimpse(spidx)
#> Observations: 6,681
#> Variables: 6
#> $ president <chr> "이승만", "이승만", "이승...
#> $ field      <chr> "기타", "국정전반", "정치...
#> $ event       <chr> "성명/담화문", "취임사", ...
#> $ title       <chr> "학생제군에게", "대통령 취임...
#> $ date        <chr> "1948", "1948.07...."
#> $ link         <chr> "http://www.pa.go...
```

데이터를 사용한 필터링 예시

```
spidx %>%
  filter(president == "윤보선")
#> # A tibble: 3 x 6
#>   president field event title date
#>   <chr>     <chr> <chr> <chr> <chr>
#> 1 윤보선    국정전반~ 취임사~ 제2대 ~ 1960~
#> 2 윤보선    기타   기타   "윤보선~ 1960~
#> 3 윤보선    기타   기타   "윤보선~ 1960~
#> # ... with 1 more variable: link <chr>
```

2.2.2 연설문 텍스트 가져오기

```
spidx %>%
  filter(president == "윤보선") %>%
  top_n(1, wt = desc(date)) %>%
  pull(link) -> tar
get_speech(tar)
#> # A tibble: 1 x 9
#>   title date president place field
#>   <chr> <chr> <chr> <chr> <chr>
#> 1 제2대 ~ 1960~ 윤보선 국내 국정전반~
#> # ... with 4 more variables:
```

2.2.3 연습문제

1. presidentSpeech 패키지에서 검색할 수 있는 대통령은 총 몇명인가요?
2. **윤보선** 대통령과 **박정희** 대통령은 각각 몇 개의 연설문이 있나요?
3. nchar() 함수는 글자수를 세주는 함수입니다. **최규하** 대통령의 취임사는 총 몇 글자 인가요?

3 단정하게 텍스트를 다루는 tidytext

3.1 tidy text data

단정한 데이터 원칙을 아래 문장과 함께 적용한다.

- a table with one-token-per-row
- 한 행(row)에 한 토큰(token)으로 테이블을 구성해야 한다.

Token 이란?

글자 중 의미를 가진 단위를 총칭.

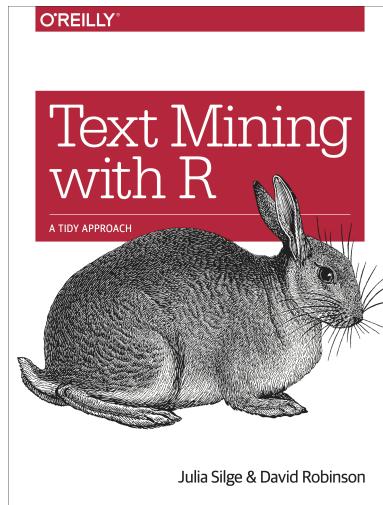
tokenization은 가지고 있는 텍스트 자원을 token 단위로 나누는 것을 뜻함.

ex> 자소(자음, 모음), 음소(글자), 형태소, 단어, n-gram 등

3.2 tidytext 패키지

- [tidytext \(<https://juliasilge.github.io/tidytext/>\)](https://juliasilge.github.io/tidytext/)
- 한 행(row)에 한 토큰(token)으로 테이블을 구성하기 위한 패키지
- 파이프 연산자를 지원
- 여러 가지 token과 tm 패키지와의 호환 기능을 제공

- 자세히 소개하는 온라인 사이트(영문)(<https://www.tidytextmining.com/>).



설치하기

```
if (!requireNamespace("tidytext")) {  
  install.packages("tidytext")  
}  
library(tidytext)
```

3.2.1 token 단위 처리 `unnest_tokens()`

기본값인 단어 단위(특수문자 제거, 띄어쓰기 기준) `token`으로 동작.

함수 설명

```
unnest_tokens(  
  # 다른고자 하는 텍스트 데이터 객체  
  tbl = 텍스트 데이터,  
  # token화의 결과가 작성될 열의 이름  
  output = 결과열의 이름,  
  # 텍스트 데이터 객체 내의 텍스트 열  
  input = 목표 텍스트 열,  
  # 기본값(words 단위 = 띄어쓰기 단위)이 있어 생략 가능  
  token = "words",  
  # 기타 옵션들  
  ...  
)
```

예시 코드

```
tar  
#> [1]  
"http://www.pa.go.kr/research/contents/speech/index.jsp?  
spMode=view&cat id=c_pa02062&art id=1310437"  
# 연설문 중 1개를 가져와서  
get_speech(tar) %>%  
  # 대통령 컬럼과 연설문 컬럼만 선택한 후  
  select(president, content) %>%  
  # 연설문 컬럼을 띄어쓰기 단위로 쪼갠 결과물을 word라  
  는 컬럼으로 출력  
  unnest_tokens(  
    input = content,  
    output = word  
  )  
#> # A tibble: 483 x 2  
#>   president word  
#>   <chr>     <chr>  
#> 1 윤보선     제  
#> 2 윤보선     2  
#> 3 윤보선     공화국의  
#> 4 윤보선     초대대통령으로  
#> 5 윤보선     영예의  
#> 6 윤보선     당선을  
#> # ... with 477 more rows
```

3.2.2 연습문제

1. 김영삼 대통령의 첫 국무회의 연설문을 띄어쓰기 단위로 자르면 총 몇 단어인가요?

- 띄어쓰기 단위로 나눴을 때 문제점

하다가 몇 가지 단어가 되는지

<https://namu.wiki/w/파일:M4nNWBR.png>

Korean verb '하다' Conjugated

regular verb

Form	Conjugation
base	하 ha
base2	하 ha
base3	하 ha
declarative present informal low	해 hae
declarative present informal high	해요 hae-yo
declarative present formal low	한다 han-da
declarative present formal high	합니다 hab-ni-da
past base	했 haess
declarative past informal low	했어 haess-eo
declarative past informal high	했어요 haess-eo-yo
declarative past formal low	했다 haess-da
declarative past formal high	했습니다 haess-seub-ni-da
future base	할 hal
declarative future informal low	할 거야 hal geo-ya
declarative future informal high	할 거예요 hal geo-ye-yo
declarative future formal low	할 거다 hal geo-da
declarative future formal high	할 겁니다 hal geob-ni-da
declarative future conditional informal low	하겠어 ha-gess-eo
declarative future conditional informal high	하겠어요 ha-gess-eo-yo
declarative future conditional formal low	하겠다 ha-gess-da
declarative future conditional formal high	하겠습니까 ha-gess-seub-ni-da
inquisitive present informal low	해? hae?
inquisitive present informal high	해요? hae-yo?
inquisitive present formal low	하니? ha-ni?

4 형태소 분석

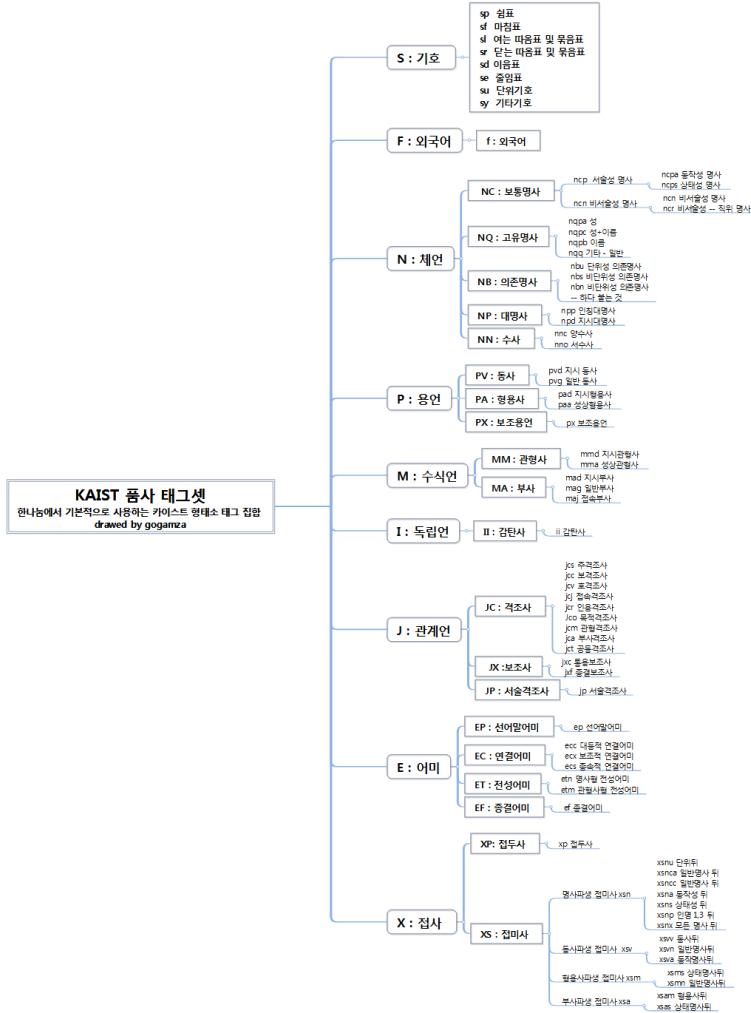
4.1 한글의 특징 형태소

형태소란 의미를 가지는 최소 단위

철수가 밥을 먹었다.

```
#> $철수가
#> [1] "철수/ncpat+가/jcc"
#> [2] "철수/ncpat+가/jcs"
#> [3] "철수/ncpat+가/ncn"
#> [4] "철수/ncpat+01/jpt+가/ecc"
#> [5] "철수/ncn+가/jcc"
#> [6] "철수/ncn+가/jcs"
#> [7] "철수/ncn+가/ncn"
#> [8] "철수/ncn+01/jpt+가/ecc"
#> [9] "철/xp+수가/ncn"
#> [10] "철/xp+수/ncn+가/jcc"
#> [11] "철/xp+수/ncn+가/jcs"
#> [12] "철/xp+수/ncn+01/jpt+가/ecc"
#>
#> $밥을
#> [1] "밥/ncn+을/jco" "밥/ncn+을/jcs"
#> [3] "밥/ncpat+을/jco" "밥/ncpat+을/jcs"
#> [5] "밥/ncpst+을/jco" "밥/ncpst+을/jcs"
#>
#> $먹었다
#> [1] "먹/pvg+었/ep+d/ef"
#>
#> $.
#> [1] "./sf" "./sy"
```

형태소 분석기의 형태소 품사



[크게 보기 \(\[https://github.com/haven-jeon/KoNLP/blob/master/etc/figures/konlp_tags.png?raw=true\]\(https://github.com/haven-jeon/KoNLP/blob/master/etc/figures/konlp_tags.png?raw=true\)\)](https://github.com/haven-jeon/KoNLP/blob/master/etc/figures/konlp_tags.png?raw=true)

[여러 체계의 형태소 품사 \(\[https://docs.google.com/spreadsheets/d/1OGAjUvalBuX-oZvZ_-9tEfYD2gQe7hTGsgUpiBSXI8/edit#gid=0\]\(https://docs.google.com/spreadsheets/d/1OGAjUvalBuX-oZvZ_-9tEfYD2gQe7hTGsgUpiBSXI8/edit#gid=0\)\)](https://docs.google.com/spreadsheets/d/1OGAjUvalBuX-oZvZ_-9tEfYD2gQe7hTGsgUpiBSXI8/edit#gid=0)

4.2 형태소 분석기

4.2.1 R의 대표적인 형태소 분석기

RcppMeCab

1. 일본어 형태소 분석기인 mecab 기반
2. C++로 작성하여 속도가 매우 빠름
3. 일본어, 중국어 등도 사용 가능
4. 형태소 분석 함수를 제공
5. 띄어쓰기에 덜 민감함

KoNLP

1. 가장 유명한 형태소 분석기
2. java로 작성된 한나눔 분석기 기반
3. 우리샘, NiaDIC 등 자체 사전
4. 텍스트 분석을 위한 기능들을 제공
5. 친절한 설명서 (<https://github.com/haven-jeon/KoNLP/blob/master/etc/KoNLP-API.md>).

RcppMeCab 설치 확인

```
library(RcppMeCab)
pos(iconv("롯데마트가 판매하고 있는 흑마늘 양념 치킨이
논란이 되고 있다.", to = "utf8"))
#> $'롯데마트가 판매하고 있는 흑마늘 양념 치킨이 논란이
#> 되고 있다.'
#> [1] "롯데마트/NNP"  "가/JKS"
#> [3] "판매/VNG"      "힣/XSV"
```

```
#> [9] "마늘/NNG"      "양념/NNG"
#> [11] "치킨/NNG"     "O1/JKS"
#> [13] "논란/NNG"     "O1/JKS"
#> [15] "되/VV"          "고/EC"
#> [17] "있/VX"          "다/EF"
#> [19] "./SF"
```

KoNLP 설치 확인

```
library(KoNLP)
SimplePos09("롯데마트가 판매하고 있는 흑마늘 양념 치킨
이 논란이 되고 있다.")
#> $롯데마트가
#> [1] "롯데마트/N+가/J"
#>
#> $판매하고
#> [1] "판매/N+하고/J"
#>
#> $있는
#> [1] "있/P+는/E"
#>
#> $흑마늘
#> [1] "흑마늘/N"
#>
#> $양념
#> [1] "양념/N"
#>
#> $치킨/OI
#> [1] "치킨/N+OI/J"
#>
#> $논란/OI
#> [1] "논란/N+OI/J"
#>
#> $되고
#> [1] "되/P+고/E"
#>
#> $있다
#> [1] "있/P+다/E"
#>
#> $.
#> [1] "./S"
```

4.2.2 생각해보기

1. 노태우 대통령의 취임사를 RcppMeCab 패키지의 pos() 함수로 형태소 분석한 결과를 출력하세요.
2. 김대중 대통령의 취임사를 KoNLP 패키지의 SimplePos09() 함수로 형태소 분석한 결과를 출력하세요.

4.2.3 형태소로 token화

4.2.3.1 데이터 준비

```
library(tidytext)
library(dplyr)
library(presidentSpeech)

spidx %>%
  filter(president == "이명박") %>%
  filter(grepl("취임사", title)) %>%
  pull(link) ->
tar
```

연설문 가져와서 word(띄어쓰기) 단위로 나누기

```
# 연설문 중 1개를 가져와서
get_speech(tar, paragraph = T) %>%
  # 문단 컬럼과 연설문 컬럼만 선택한 후
  select(paragraph, content) %>%
  # 연설문 컬럼을 word 단위로 쪼갠 결과물을 word라는 컬
  # 럼으로 출력
  unnest_tokens(
    input = content,
```

```

)
#> # A tibble: 2,019 x 2
#>   paragraph word
#>   <int> <chr>
#> 1       1 존경하는
#> 2       1 국민
#> 3       1 여러분
#> 4       2 700
#> 5       2 만
#> 6       2 해외동포
#> # ... with 2,013 more rows

```

형태소 분석기와 함께 사용하기

```

unnest_tokens(
  tbl = 텍스트 데이터,
  input = 목표 텍스트 열,
  output = 결과열의 이름,
  token = "words",           ← 여기에 형태소 분석 함수를
  적용
  ...
)

```

KoNLP의 `SimplePos09()` 함수를 활용해서 형태소 단위로 쪼갠 데이터를 만듭니다.

```

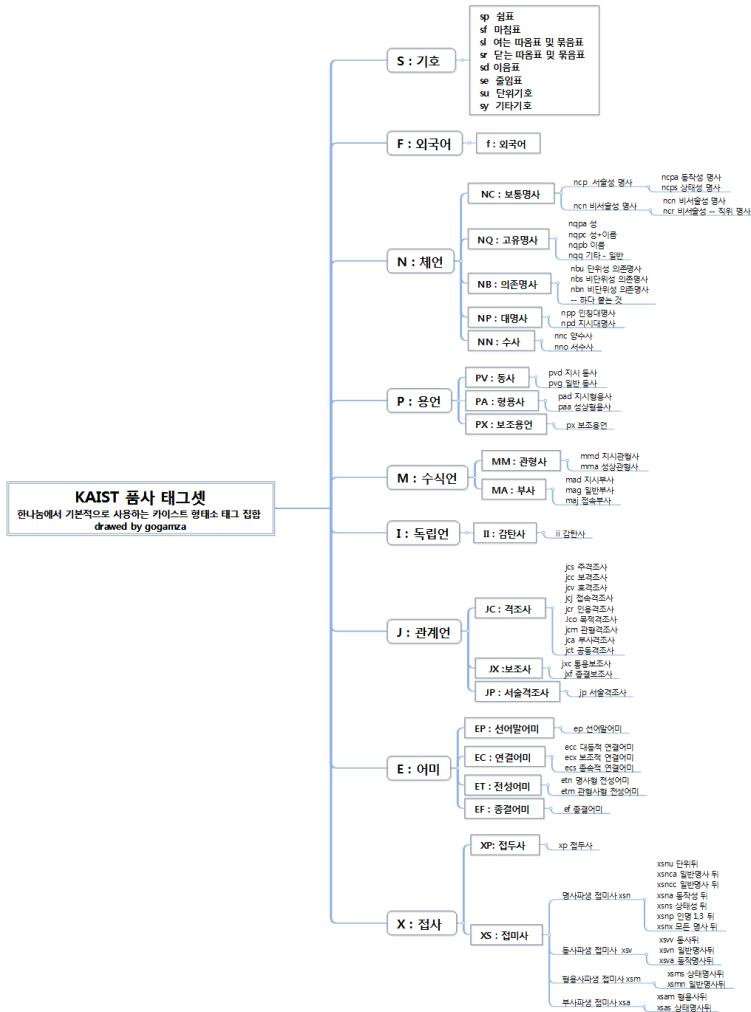
library(KoNLP)
# 연설문 중 1개를 가져와서
get_speech(tar, paragraph = T) %>%
  # 문단 컬럼과 연설문 컬럼만 선택한 후
  select(paragraph, content) %>%
  # 연설문 컬럼을 형태소 단위로 쪼개
  # pos라는 컬럼으로 출력
  unnest_tokens(pos, content, token = SimplePos09) %>%
  # pos 결과물의 순서 보장을 위해 순서 값 추가
  mutate(pos_order = 1:n()) ->
  pos_res
pos_res

```

```
#>      <int> <chr>      <int>
#> 1  1 존경하/n+는/j  1
#> 2  1 국민/n        2
#> 3  1 여려분/n      3
#> 4  1 |/s            4
#> 5  2 700/n+만/j    5
#> 6  2 해외동포/n    6
#> # ... with 2,210 more rows
```

4.2.3.2 형태소 분석기가 사용하는 태그

4.2.3.2 형태소 분석기가 사용하는 태그



* 크게 보기 (https://github.com/haven-jeon/KoNLP/blob/master/etc/figures/konlp_tags.png?raw=true)

4.2.4 불용어 제거

4.2.4.1 필요한 형태소 정보만 선택

신뢰할 수 있는 stop word 사전 등이 없기 때문에, 형태소 분석 후 필요한 형태소만 활용.

```
pos_res %>%
  # 우선 `filter()` 와 `grep1()` 함수를 활용하여 명사 (n)만 추출
  filter(grep1("/n", pos)) %>%
  # 형태소 정보를 제거
  mutate(pos_done = gsub("/.*$","", pos)) ->
  n_done

n_done
#> # A tibble: 1,315 x 4
#>   paragraph pos      pos_order pos_done
#>   <int>     <chr>    <int>     <chr>
#> 1       1 존경하/n+는/~      1 존경하
#> 2       1 국민/n            2 국민
#> 3       1 여러분/n          3 여러분
#> 4       2 700/n+만/~        5 700
#> 5       2 해외동포/n~      6 해외동포
#> 6       2 여러분/n          7 여러분
#> # ... with 1,309 more rows
```

4.2.4.2 명사, 형용사, 동사 가져오기

명사는 n, 동사/형용사는 p로 표시. 형태소 분석 후 한 글자는 전후 맥락 없이 의미를 파악하기 어렵기 때문에 제거

```
pos_res %>%
  filter(grep1("/p", pos)) %>%
```

```

p_done

bind_rows(n_done, p_done) %>%
  arrange(pos_order) %>%
  filter(nchar(pos_done) > 1) %>%
  select(paragraph, pos_done) ->
  pos_done

pos_done
#> # A tibble: 1,633 x 2
#>   paragraph pos_done
#>   <int> <chr>
#> 1      1 존경하
#> 2      1 국민
#> 3      1 여러분
#> 4      2 700
#> 5      2 해외동포
#> 6      2 여러분
#> # ... with 1,627 more rows

```

4.2.4.3 함께 사용한 함수 설명

`grep()`, `grep1()`

`grep()` 함수는 글자 데이터내에 찾고자 하는 글자가 있는 위치를 인덱스(숫자)로 알려줌

`grep1()` 함수는 결과를 T/F로 알려줌

```

grep(
  pattern = 찾고자 하는 글자,
  x = 글자 데이터,
  fixed = 정규식을 사용 여부, # T/F로 되어 있으며
  FALSE가 기본값
  ...
)

```

`gsub()`

`gsub()` 함수는 찾고자 하는 글자를 원하는 글자로 바꿔줌

```
gsub(
  pattern = 찾고자 하는 글자,
  replacement = 찾은 글자가 바뀌게 될 글자,
  x = 글자 데이터,
  fixed = 정규식을 사용 여부, # T/F로 되어 있으며
  FALSE가 기본값
  ...
)
```

`nchar()`

`nchar()`는 글자 데이터를 받아서 글자수를 알려줌

```
nchar(
  x = 세고자 하는 글자,
  ...
)
```

4.2.5 정규 표현식

글자를 다루는데 유용한 기능을 제공

- ^ : 이걸로 시작함
- \$: 이걸로 끝남
- . : 임의의 글자 하나
- ?: 앞에 있는 문자가 없거나 하나
- + : 앞에 있는 문자가 하나 이상
- * : 앞에 있는 문자가 없거나 하나 이상

참고 :

https://mrchypark.github.io/dabrp_classnote3/class4

4.2.6 연습문제

1. 아래 코드로 이명박 대통령 연설문 중 10개를 가져오세요.
2. grep() 함수를 사용해서 제목(title 컬럼)에 나눔이 있는 연설문을 찾으세요.
3. gsub() 함수를 사용해서 제목(title 컬럼)에 인터넷 연설 글자를 없애보세요.
4. mutate() 함수를 사용해서 연설문id를 1~10까지 id 컬럼으로 추가하세요.
5. id와 content 컬럼만 선택하세요.
6. 형태소 분석을 하여 결과를 pos 컬럼으로 추가하세요.
7. grep() 함수를 사용해서 명사만 남겨보세요.
8. gsub() 함수를 사용해서 POS 정보를 지우고 한글만 남기세요.

```
library(presidentSpeech)
library(magrittr)
library(tidyverse)
try_speech <- insistently(get_speech)
spidx %>%
  filter(president == "이명박") %>%
  arrange(date) %>%
  top_n(10) %$%
  map_dfr(link, try_speech) ->
  tar
#> Selecting by link
```

```
#>   title date president place field
#>   <chr> <chr> <chr>     <chr> <chr>
#> 1 제98차~ ""    이명박    국내  외교/통~
#> 2 제99차~ ""    이명박    국내  기타
#> 3 제100~ ""    이명박    국내  국정전반~
#> 4 제101~ ""    이명박    국내  외교/통~
#> 5 제102~ ""    이명박    국내  과학기술~
#> 6 제103~ ""    이명박    국내  외교/통~
#> # ... with 4 more rows, and 4 more
#> #   variables: event <chr>,
#> #   source <chr>, paragraph <int>,
#> #   content <chr>
```

4.2.7 RcppMeCab 실습

4.2.7.1 RcppMeCab의 pos() 함수 사용

```
library(RcppMeCab)
spidx %>%
  filter(president == "이명박") %>%
  filter(grepl("취임사", title)) %>%
  pull(link) ->
  tar

tar %>%
  get_speech(paragraph = T) %>%
  select(paragraph, content) %>%
  # token에 사용할 함수 이름 pos를 입력
  unnest_tokens(pos, content, token = pos) %>%
  mutate(pos_order = 1:n()) ->
  pos_res
pos_res
#> # A tibble: 4,180 x 3
#>   paragraph pos      pos_order
#>   <int> <chr>     <int>
#> 1          1 존경/nng      1
#> 2          1 𠮩/xsv       2
#> 3          1 는/etm       3
#> 4          1 국민/nng      4
#> 5          1 여려분/np      5
```

4.2.7.2 앞선 코드의 문제점

새로운다 같은 글자가 발생

```
pos_res %>%
  filter(grep1("/va", pos)) %>%
  mutate(pos_done = gsub("/.*$","다", pos))
#> # A tibble: 49 x 4
#>   paragraph pos      pos_order pos_done
#>   <int> <chr>     <int> <chr>
#> 1       9 눈물겹/va    453 눈물겹다
#> 2       10 수많/va    502 수 많다
#> 3       14 새로운/vate~ 623 새로운다
#> 4       16 아름답/va   798 아름답다
#> 5       16 없/va     825 없다
#> 6       16 없/va     835 없다
#> # ... with 43 more rows
```

4.2.7.3 정규화 예시

정규화란 표현 방법이 다른 단어들을 통합시켜서 같은 단어로 만들어주는 것. 보통 US, USA 등 같은 뜻이지만 다른 모양의 단어를 찾아 통일시키는 과정을 뜻함. 현재는 /vv+etm 중 마지막에 ㄴ 받침으로 끝나는 형태가 ㄴ 만 떼면 원형이 되는 경우를 정리하여 정규화하려고 함.

```
library(tidyr)
library(purrr)
library(KoNLP)

pos_res %>%
  filter(grep1("/n", pos)) %>%
  mutate(pos_done = gsub("/.*$","", pos)) ->
```

```
pos_res %>%
  filter(grep1("/v(a|v)$", pos)) %>%
  mutate(pos_done = gsub("/.*$", "\uacfc", pos)) ->
  v_done

pos_res %>%
  filter(grep1("/sn", pos)) %>%
  mutate(pos_done = gsub("/.*$", "", pos)) ->
  sn_done

jamos <-
  function(text) {
    paste0(
      convertHangulStringToJamos(text)
      , collapse = " ")
  }

pos_res %>%
  filter(grep1("/v(a|v)WW+etm", pos)) %>%
  mutate(po = gsub("/.*$", "", pos)) %>%
  mutate(post = po %>%
    map(jamos)) %>%
  unnest %>%
  mutate(n = nchar(post)) %>%
  mutate(posts = substr(post, 1, n - 1)) %>%
  mutate(done = posts %>%
    map_chr(HangulAutomata)) %>%
  mutate(pos_done = paste0(done, "\uacfc")) ->
  etm_done

bind_rows(sn_done,
          n_done,
          v_done,
          etm_done) %>%
  filter(nchar(pos_done) > 1) %>%
  arrange(pos_order) %>%
  select(paragraph, pos_done) ->
  pos_done

pos_done
#> # A tibble: 1,543 x 2
#>   paragraph pos_done
#>   <int>     <chr>
#> 1           1 존경
```

```
#> 4      2 700  
#> 5      2 해외  
#> 6      2 동포  
#> # ... with 1,537 more rows
```

5 텍스트 마이닝 지표

1. 단어 출현 빈도 : 단순히 단어가 나타난 횟수를 세서 확인
2. 동시 출현 빈도 : 기준 단어와 함께 나타난 단어들과 그 횟수를 세서 확인
3. tf-idf : 전체 문서에서 나타난 횟수와 개별 문서에서 나타난 횟수로 만든 지표

$$w_{x,y} = \text{tf}_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

TF-IDF
Term x within document y

$\text{tf}_{x,y}$ = frequency of x in y
 df_x = number of documents containing x
 N = total number of documents

4. 감성 분석 : 단어를 점수화한 감성사전을 사용하여 점수를 합산하여 만든 지표

5.1 단어 출현 빈도

5.1.1 단어 출현 빈도 계산

`count()` 함수는 데이터에서 총 몇 번 나왔는지 세어주는 집계 함수. `group_by()`와 함께 사용하여 각 연설문별 출현 횟수 등을 구할 수 있음.

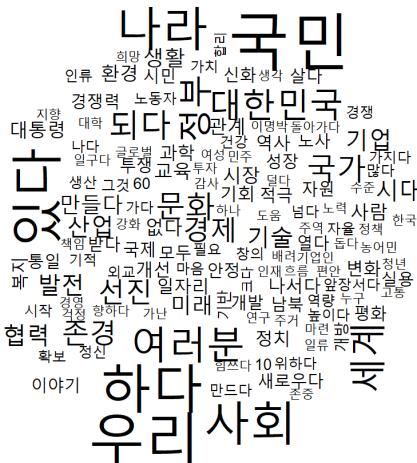
```
library(dplyr)
pos_done %>%
  count(pos_done, sort = T) ->
  wn
#> # A tibble: 746 x 2
```

```
#> 1 국민      30
#> 2 우리      27
#> 3 하다      25
#> 4 있다      24
#> 5 나라      22
#> 6 사회      22
#> # ... with 740 more rows
```

5.1.2 사용예 : 워드클라우드

`count()` 함수로 단어와 그 빈도 테이블을 만들었다면, `{wordcloud}` 패키지를 사용해서 워드클라우드를 만들 수 있음. `{showtext}` 패키지를 출력 결과물의 폰트를 설정하기 위한 패키지로 [Google Fonts \(<https://fonts.google.com/>\)](https://fonts.google.com/)에서 폰트 데이터를 받아와서 출력물에 사용할 수 있음.

```
library(wordcloud)
library(showtext)
font_add_google("Noto Sans", "notosans")
showtext_auto()
wn %>%
  with(wordcloud(pos_done, n, family = "notosans"))
```



빈도에 따른 색 입히기

<https://github.com/EmilHvitfeldt/r-color-palettes>

에 R에서 사용할 수 있는 색 테마 패키지들을 소개하고 있음.

```
# install.packages("Redmonder")
library(Redmonder)
pal = redmonder.pal(6, "sPBIRdPu")

wn %>%
  with(wordcloud(pos_done,
                 n,
                 family = "notosans",
                 colors = pal))
```



5.1.3 연습문제

```
library(presidentSpeech)
library(magrittr)
library(tidyverse)
try_speech <- insistently(get_speech)
spidx %>%
  filter(president == "이명박") %>%
  arrange(date) %>%
  top_n(10) %>%
  map_dfr(link, try_speech) ->
  tar
#> Selecting by link
tar
#> # A tibble: 10 x 9
#>   title date  president place field
#>   <chr> <chr> <chr>    <chr> <chr>
#> 1 제98차~   ""    이명박    국내 외교/통~
#> 2 제99차~   ""    이명박    국내 기타
#> 3 제100~    ""    이명박    국내 국정전략~
```

```
#> 6 제103~    이명박    국내 외교/통~
#> # ... with 4 more rows, and 4 more
#> # variables: event <chr>,
#> #   source <chr>, paragraph <int>,
#> #   content <chr>
```

1. tar의 content 컬럼을 pos() 함수로 형태소 분석을 진행 해주세요.
2. 그 중 명사만 남기고, 형태소 정보는 지워주세요. 한글 자 명사도 지워주세요.
3. count() 함수를 이용해서 단어 출현 빈도를 계산해 주세요.
4. wordcloud를 만들어 주세요.
5. 다른 색 조합으로 시도해 주세요.
6. group_by()를 활용하여 각 연설문 별로 단어 출현 빈도를 계산해주세요.
7. 각 연설문에서 “우리”가 몇 번 사용되었는지 확인해 주세요.

5.2 동시 출현 빈도

5.2.1 동시 출현 빈도 계산

pairwise_count() 함수는 그룹 단위 내에서 단어가 동시에 출현한 횟수를 세어주는 함수. 보통 문장 단위를 그룹으로 처리

```
#> # A tibble: 22,303 x 3
#>   item1  item2      n
#>   <chr>  <chr>  <dbl>
```

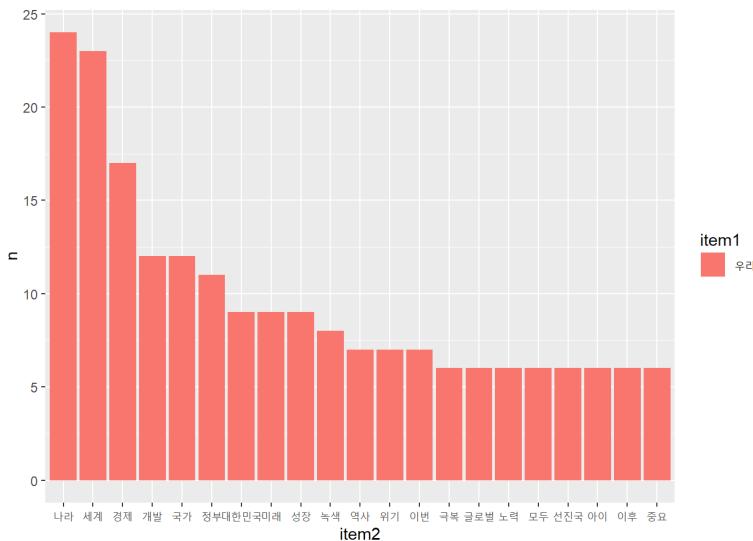
```
#> 3 우리    세계      23
#> 4 국민    사랑      22
#> 5 여러분 사랑      22
#> 6 시장    전통      22
#> # ... with 2.23e+04 more rows
```

5.2.2 기준 단어로 데이터 탐색

`filter()` 함수로 기준 단어를 조회하면 함께 자주 나오는 단어와 그 빈도를 확인할 수 있음

```
pw %>%
  filter(item1 == "우리")
#> # A tibble: 711 x 3
#>   item1 item2     n
#>   <chr>  <chr> <dbl>
#> 1 우리    나라     24
#> 2 우리    세계     23
#> 3 우리    경제     17
#> 4 우리    국가     12
#> 5 우리    개발     12
#> 6 우리    정부     11
#> # ... with 705 more rows

library(forcats)
library(ggplot2)
# bar plot
pw %>%
  filter(item1 %in% c("우리")) %>%
  top_n(15) %>%
  mutate(item2 = fct_reorder(item2, n, .desc = TRUE))
%>%
  ggplot(aes(x = item2, y = n, fill = item1)) +
  geom_bar(stat = "identity")
#> Selecting by n
```



5.2.3 연습문제

```

library(presidentSpeech)
library(magrittr)
library(tidyverse)
try_speech <- insistently(get_speech)
spidx %>%
  filter(president == "이명박") %>%
  arrange(date) %>%
  top_n(10) %$%
  map_dfr(link, try_speech) ->
  tar
#> Selecting by link
tar
#> # A tibble: 10 x 9
#>   title date  president place field
#>   <chr> <chr> <chr>    <chr> <chr>
#> 1 제98차~ ""    이명박    국내  외교/통~
#> 2 제99차~ ""    이명박    국내  기타
#> 3 제100~  ""    이명박    국내  국정전반~

```

```
#> 6 제103~ ""      이명박      국내 외교/통~
#> # ... with 4 more rows, and 4 more
#> #   variables: event <chr>,
#> #   source <chr>, paragraph <int>,
#> #   content <chr>
```

1. tar의 content 컬럼을 문장 단위로 나누어 주세요.
2. 새롭게 문장별 id를 id 컬럼으로 추가해주세요.
3. 문장별 id를 유지한 채로 pos() 함수를 사용하여 형태소 분석을 진행해 주세요.
4. 명사(/n), 동사(/vv), 형용사(/va)인 형태소만 가져와 주세요.
5. 형태소 정보는 제거하지 말고 그대로 두세요.
6. 동시 출현 빈도 테이블을 만들어 주세요. (컬럼이 item1, item2, n으로 구성됩니다.)
7. 우리/np와 함께 출현한 단어들과 그 빈도를 확인하세요.
8. 명사는 형태소 정보를 제거하고, 형용사와 동사는 형태소 정보를 제거한후 뒤에 다를 붙여주세요.
9. 한 글자는 제거해 주세요.
10. 동시 출현 빈도 테이블을 만들어 주세요. (컬럼이 item1, item2, n으로 구성됩니다.)
11. 사랑과 함께 출현한 단어들과 그 빈도를 확인하세요.

5.2.4 사용예 : 네트워크 시각화

```
library(igraph)
pw %>%
  filter(n > 5) %>%
  graph_from_data_frame() ->
```

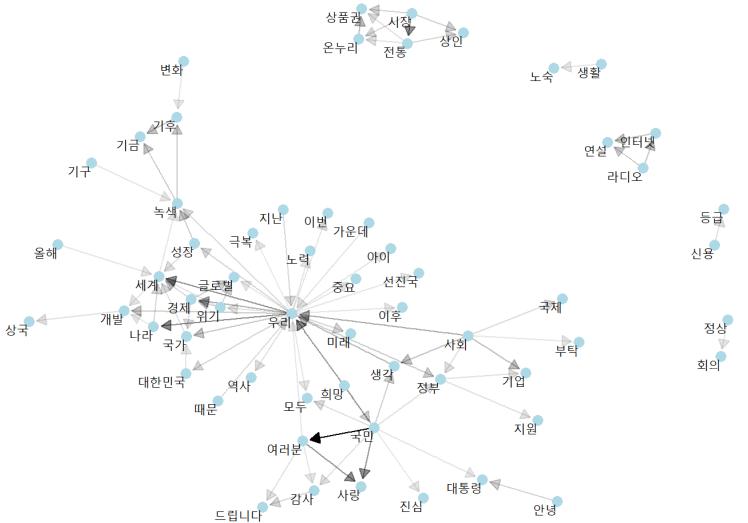
```
#> /IGRAPH 7cb7bb8 DN-- 60 85 --
#> + attr: name (v/c), n (e/n)
#> + edges from 7cb7bb8 (vertex names):
#> [1] 국민 ->여러분 우리 ->나라
#> [3] 우리 ->세계 국민 ->사랑
#> [5] 여러분->사랑 시장 ->전통
#> [7] 국민 ->우리 사회 ->우리
#> [9] 우리 ->경제 생각 ->우리
#> [11] 온누리->상품권 사회 ->기업
#> [13] 사회 ->생각 기후 ->기금
#> [15] 라디오->인터넷 라디오->연설
#> + ... omitted several edges
```

네트워크 데이터는 node, edge로 구성됨

```
library(ggraph)
set.seed(2018)

a <- grid::arrow(type = "closed", length = unit(.1,
"inches"))

ggraph(pw_graph) +
  geom_edge_link(aes(edge_alpha = n), show.legend =
  FALSE,
  arrow = a, end_cap = circle(.07,
  'inches')) +
  geom_node_point(color = "lightblue", size = 3) +
  geom_node_text(aes(label = name), vjust = 1, hjust =
  1) +
  theme_void()
#> Using `nicely` as default layout
```



5.3 tf-idf

- tf : 전체 문서내의 단어 빈도
 - idf : 단어를 가지는 문서 비율의 역수

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
 df_x = number of documents containing x
 N = total number of documents

5.3.1 tf-idf 계산

`bind_tf_idf()` 함수가 `tf`, `idf`, `tf-idf` 점수 모두를 제공하며 문서 단위의 정의가 매우 중요함. 보통 각 연설문, 개별 뉴스 본문 등을 하나의 문서로 정의함. `tf-idf` 가 높을 수록 각 문서에서 특별한 의미를 지니는 것으로 판단할 수 있음.

```

tar %>%
  mutate(id = as.numeric(1:n())) %>%
  unnest_tokens(pos, content, token = pos) %>%
  select(id, pos) %>%
  filter(grepl("/n|/v\\|a", pos)) %>%
  mutate(pos = gsub("/.*$","",pos)) %>%
  filter(nchar(pos) > 1) %>%
  group_by(id) %%%
  count(pos) ->
  tfidf_tar

tfidf_tar %>%
  bind_tf_idf(pos, id, n) %>%
  arrange(desc(tf_idf))
#> # A tibble: 3,429 x 6
#> # Groups:   id [10]
#>      id pos     n    tf    idf tf_idf
#>   <dbl> <chr> <int> <dbl> <dbl> <dbl>
#> 1     2 주석      7 0.0330  1.61 0.0531
#> 2     8 선거      6 0.0208  2.30 0.0478
#> 3     1 그린란드~    9 0.0178  2.30 0.0409
#> 4     6 아랍에미리~    9 0.0174  2.30 0.0401
#> 5     4 기금     12 0.0232  1.61 0.0373
#> 6     6 원전      8 0.0155  2.30 0.0356
#> # ... with 3,423 more rows

```

5.3.2 연습문제

```

library(presidentSpeech)
library(magrittr)
library(tidyverse)
try_speech <- insistently(get_speech)
spidx %>%
  filter(president == "이명박") %>%
  arrange(date) %>%
  top_n(10) %$%
  map_dfr(link, try_speech) ->
  tar
#> Selecting by link
tar

```

```
#> <chr> <chr> <chr> <chr> <chr>
#> 1 제98차~ "" 이명박 국내 외교/통~
#> 2 제99차~ "" 이명박 국내 기타
#> 3 제100~ "" 이명박 국내 국정전반~
#> 4 제101~ "" 이명박 국내 외교/통~
#> 5 제102~ "" 이명박 국내 과학기술~
#> 6 제103~ "" 이명박 국내 외교/통~
#> # ... with 4 more rows, and 4 more
#> # variables: event <chr>,
#> # source <chr>, paragraph <int>,
#> # content <chr>
```

1. 새롭게 연설문별 id를 id 컬럼으로 추가해주세요.
2. 문장별 id를 유지한 채로 pos() 함수를 사용하여 형태소 분석을 진행해 주세요.
3. 명사(/n), 동사(/vv), 형용사(/va)인 형태소만 가져와 주세요.
4. 명사는 형태소 정보를 제거하고, 형용사와 동사는 형태소 정보를 제거한후 뒤에 다를 붙여주세요.
5. 한 글자는 제거해 주세요.
6. 연설문 별로 형태소 단위 빈도를 계산해 주세요.
7. bind_tf_idf() 함수를 사용해서 tf, idf, tf-idf 를 계산 해주세요.
8. 각 연설문 별로 tf-idf 점수가 가장 높은 단어를 확인 하세요.
9. 각 연설문 별로 tf-idf 점수가 가장 높은 3개 단어씩을 확인하세요.

5.4 감성 분석

- 한글의 특성상, 형태소이며 ngram에 점수를 부여하는 것이 가장 효과적일 것
- 단순한 형태로는 unigram의 형태소에 점수나 종류를 부여하는 것
- 개별 단어의 점수를 부여한 뒤 문장 단위로 합산하여 계산
- 합산으로 0에 가까운 값이 나올 수도 있으므로 점수를 부여받은 단어의 갯수등도 고려 필요
- 안정적으로 기구축된 한글 사전을 찾기 어려움

5.4.1 사전 소개

[KnuSentiLex](https://github.com/park1200656/KnuSentiLex) (<https://github.com/park1200656/KnuSentiLex>)는 군산대 [Data Intelligence Lab](http://dilab.kunsan.ac.kr/) (<http://dilab.kunsan.ac.kr/>)에서 기존 사전들을 참조, 활용하여 18년 구축한 감성 사전. 구조가 단순하고 이모티콘 등을 추가한 점이 장점인 반면, 형태소 형식이 아니라 점수의 신뢰도가 낮은 편임.

[KOSAC](http://word.snu.ac.kr/kosac/) (<http://word.snu.ac.kr/kosac/>)은 서울대에서 13년에 구축한 감성사전으로 구조가 복잡하고 점수를 내기 어렵지만 12년에 구축한 감성 스키마를 따르고 있어 다양한 감성 정보를 얻을 수 있음.

본 예시에는 구조가 단순한 KnuSentiLex을 사용

```
# remotes::install_github("mrcchypark/KnuSentiLexR")
library(KnuSentiLexR)
tar %>%
```

```
filter(nchar(sent) < 20) %>%
select(sent) ->
senti_tar
```

5.4.2 감성 분석 점수

- senti_score() 함수는 문장을 unigram 부터 3-gram 까지 작성한 후, 감성 사전에 점수를 합산하여 문장 점수를 계산
- senti_magnitude() 함수는 몇개의 ngram이 점수화되었는지를 계산
- dic 객체가 word, polarity 컬럼을 가지고 있는 감성 사전임

```
senti_tar %>%
  mutate(score = senti_score(sent),
         magni = senti_magnitude(sent)) %>%
  filter(score != 0)
#> # A tibble: 38 x 3
#>   sent                      score magni
#>   <chr>                    <dbl>  <dbl>
#> 1 먼저 함께 보시죠.          1      1
#> 2 자랑 좀 해 보세요.        -2     1
#> 3 사회자 눈물이 그렁그렁하네요. ~    -1      1
#> 4 이상하게.                -1      1
#> 5 이분은 ‘한번 해 보자.’     -2      1
#> 6 대통령 그렇게 해 주세요. -2      1
#> # ... with 32 more rows
```

