데이터마이닝 실습 – Practice5 (Team 1)

강하영, 박성찬, 김효진, 조선주

필요한 package 불러오기

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
library(dplyr)
library(stringr)
library(tidyr)
library(ggplot2)
library(lubridate)
library(tidytext)
library(scales)
library(textdata)
library(wordcloud)
library(reshape2)
library(topicmodels)
library(wordcloud2)
library(KoNLP)
library(memoise)
library(rJava)
library(tm)
library(tidyquant)
library(stopwords)
theme_set(theme_grey(base_family='NanumGothic'))
options("scipen" = 1000)
```

데이터 불러오기

```{r}
Everytime_df <- readLines('eta_secret_board.txt', encoding = 'UTF-8')
Everytime_df <- data.frame(data.frame(values = Everytime_df)[-1, ])
head(Everytime_df)
```

가장 먼저는 에브리타임을 크롤링한 데이터를 불러오도록 한다. readLines 함수를 통해서 엔터를 기준으로 분리하고, 열의 이름을 담고 있는 가장 첫 번째 행은 제거하도록 한다. 이를 통해 데이터를 확인해보면, 총 21,500개의 글들이 불러와진 것을 확인할 수 있다.

```{r}
Everytime_df <- Everytime_df %>%
  separate(colnames(Everytime_df), c('type', 'datetime'), '"', "'", extra = 'merge') %>%
  separate(datetime, c('datetime', 'content'), '"', "'", extra = 'merge')

Everytime_df <- Everytime_df %>%
  mutate(type = gsub('"', '', type),
         content = gsub('"', '', content))

dim(Everytime_df)
head(Everytime_df, 10)
```

```
> head(Everytime_df, 10)
       type    datetime
1      post      34분 전
2   comment       8분 전
3      post      49분 전
4   comment      45분 전
5   comment      44분 전
6   comment      43분 전
7      post 10/04 02:07
8   comment 10/04 02:10
9   comment 10/04 02:29
10  comment 10/04 07:29
                                                                                content
1   아만다 가입심사 받는사람중에우리학교사람 봤는데 참 민망하군마주치지 맙시다ㅋㅋㅋㅋㅋ
2                                                                  뭐...심심해서 할수도 있죠...
3                                엄마랑 싸웠는데 내가 사과해야되는건지 좀 판단해줄사람있오..?
4                                                              어떻게 싸웠는데?? ㅜ
5                                                                        쪽지함까?
6                                                                          ㅇㅇㅇ
7                                                                  돈 진짜 개빡친다
8                                                                        왜왜
9                                                                  주식으로 날렸나?
10                                                                       영화?
> |
```

그 후에는 불러온 데이터를 열로 구분하기 위하여 separate 함수를 통해 ", "를 기준으로 type, datetime, content로 나눠주도록 한다. 또한 각 데이터에서 제거되지 않았던 큰따옴표(")를 gsub 함수를 통해 제거해준다. 그 후 결과를 확인해보면, 각 데이터들이 정상적으로 잘 불러와진 것을 확인할 수 있다.

1. Tokenization

```{r}
# 띄어쓰기가 안된 데이터들을 KoSpacing 패키지를 적용하기

# install.packages('installr')
# library(installr)
# install.Rtools()

# if (!require('devtools')) install.packages('devtools')
# devtools::install_github('talgalili/installr')
# library(installr)
# install.conda()

# if (!require('remotes')) install.packages('remotes')
# remotes::install_github('haven-jeon/KoSpacing')
# set_env()
# library(KoSpacing)
```

```{r}
# 띄어쓰기가 안되어 있는 데이터를 수정
# spaced_word <- read.csv('spaced_word.csv', header = T, fileEncoding = "CP949", encoding =
"UTF-8")[, c('word', 'spaced_word')]

# spaced_word <- spaced_word %>%
#   distinct(word, spaced_word)

# for (i in 1:dim(Everytime_df)[1]){
#   print(i)
#   target <- Everytime_df$content[i]
#   for (j in 1:dim(spaced_word)[1]){
#     if (str_detect(target, spaced_word$word[j])){
#       target <- gsub(spaced_word$word[j], spaced_word$spaced_word[j], target)
#     }
#   }
#   Everytime_df$new_content[i] <- target
# }

df <- read.csv('spaced_eta_data1.csv')
```

```
> head(Everytime_df)
     type datetime
1    post  34분 전
2 comment   8분 전
3    post  49분 전
4 comment  45분 전
5 comment  44분 전
6 comment  43분 전
                                                                                content
1 아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다ㅋㅋㅋㅋㅋ
2                                                                  뭐...심심해서 할수도 있죠...
3                                엄마랑 싸웠는 데 내가 사과해야되는건지 좀 판단해줄 사람 있오..?
4                                                                  어떻게 싸웠는데?? ㅜ
5                                                                        쪽지할까?
6                                                                          ㅇㅇㅇ
```

Tokenization을 수행하기 위해 가장 먼저 확인해야 할 것은 text 데이터이다. 이를 위해 확인해보면, 안타깝게도 중간중간마다 띄어쓰기가 명확하게 이루어지지 않은 데이터들이 있다. 예를 들면, 첫 문장 같은 경우에 '받는사람중에우리학교사람' 이라고 표현되어 있는데, 이를 Tokenization을 수행하면 기본적으로 띄어쓰기를 기반으로 하기 때문에 값이 정확하게 나오지 않을 수 있다.

따라서 한국어 띄어쓰기를 해줄 수 있는 딥러닝 기반의 오픈소스인 KoSpacing이라는 패키지를 활용하도록 한다. installr과 KoSpacing 패키지를 다운 받은 후에, spacing이라는 함수를 활용하고자 한다. 다만, 실행해보니 많은 양의 단어들을 돌리는데 꽤 많은 시간들이 소요가 되는 한계점이 있었다. 따라서 띄어쓰기가 되어 있지 않은 데이터만 추려서 해당 데이터를 바꿔주는 작업으로

수행하도록 했다. 또한 실행 시간이 오래 걸려서 수정이 완료된 데이터를 저장하여 불러오는 방식을 택했다. 최종적으로 값을 수정한 df를 출력해보면, 앞서 언급했던 띄어쓰기들이 그나마 잘 수정된 것을 확인할 수 있다.

<https://hwangknock.tistory.com/8>

```
```{r}
# 필요 없는 데이터 삭제 및 수정
Everytime_df <- Everytime_df[-grep('삭제된 댓글입니다.', Everytime_df$content), ]
Everytime_df$datetime <- ifelse(str_detect(Everytime_df$datetime, '분 전'), '10/05 12:00',
Everytime_df$datetime)

head(Everytime_df)
```
```

다음 작업을 수행하기 전에 간단한 몇 가지 전처리를 수행해주도록 한다. 가장 먼저는 에브리타임에서 댓글 중 사용자가 삭제한 댓글은 '삭제된 댓글입니다'로 나온다. 따라서 이 데이터는 분석에 있어서 의미가 없으므로 제거해주도록 한다. 또한 향후 datetime 변수를 date type으로 형 변환하기 위해서 ~분 전으로 표현된 것들을 편의상 10월 5일 12시로 수정해주도록 한다.

```
```{r}
str(Everytime_df)
```
```

```
'data.frame':   21249 obs. of  3 variables:
 $ type    : chr  "post" "comment" "post" "comment" ...
 $ datetime: chr  "10/05 12:00" "10/05 12:00" "10/05 12:00" "10/05 12:00" ...
 $ content : chr  "아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다ㅋㅋㅋㅋㅋ"
"뭐...심심해서 할수도 있죠..." "엄마랑 싸웠는 데 내가 사과해야되는건지 좀 판단해줄 사람 있오..?" "어떻게 싸웠는
데?? ㅜ" ...
```

그리고 각 변수 별로의 type을 str 함수를 통해서 확인하면, 다음과 같음을 알 수 있다. 모든 변수들이 chr로 불러졌는데, 우선 datetime을 시간 형태의 변수로 수정해주도록 한다.

```
```{r}
# datetime을 date type으로 바꿔주기 위해 데이터 수정
Everytime_df$datetime <- ifelse(str_length(Everytime_df$datetime) == 11,
                                paste0('20/', Everytime_df$datetime),
                                Everytime_df$datetime)

Everytime_df$datetime <- as.POSIXct(Everytime_df$datetime, format = '%y/%m/%d %H:%M')
str(Everytime_df)
```
```

```
'data.frame':   21249 obs. of  3 variables:
 $ type    : chr  "post" "comment" "post" "comment" ...
 $ datetime: POSIXct, format: "2020-10-05 12:00:00" "2020-10-05 12:00:00" "2020-10-05 12:00:00"
"2020-10-05 12:00:00" ...
 $ content : chr  "아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다ㅋㅋㅋㅋㅋ"
"뭐...심심해서 할수도 있죠..." "엄마랑 싸웠는 데 내가 사과해야되는건지 좀 판단해줄 사람 있오..?" "어떻게 싸웠는
데?? ㅜ" ...
```

datetime의 형태를 바꾸기 위해 가장 먼저 해당 변수의 타입을 맞춰주도록 한다. 데이터를 확인해보니, 일부는 'MM/DD HH:MM'으로 표현되어 있고, 일부는 'YY/MM/DD HH:MM'으로 표현되어 있다. 전자의 경우에는 이 데이터를 크롤링 할 당시인 2020년이 빠진 것으로 추정할 수 있다. 따라서 str_length 함수를 통해서 길이가 11인 데이터는 앞에 연도를 의미하는 '20/'를 붙여주도록 한다. 그리고 데이터 타입을 확인하면, date type으로 잘 변화된 것을 확인할 수 있다.

```{r}
# post와 comment들을 매칭시킬 수 있도록 post_num 변수 추가
i <- 1
Everytime_df$post_num <- NA
for (row in 1:dim(Everytime_df)[1]){
  if (Everytime_df$type[row] == 'post'){
    Everytime_df$post_num[row] <- i
    i <- i + 1
  }
}

Everytime_df$post_num <- na.locf(Everytime_df$post_num)
ori_Everytime_df <- Everytime_df
head(ori_Everytime_df)
```

가장 마지막으로는 향후 각 글(post)과 댓글(comment)을 구분할 수 있도록 post_num 변수를 만들어준다. 이를 위한 알고리즘은 type 변수에서 post에만 1씩 증가하면서 새로운 숫자를 만들어준다. 그리고 zoo 패키지에 있는 na.locf 함수를 통해서 NA를 전에 있는 값으로 대체해주도록 한다.

```{r}
# 특수 문자와 자음과 모음으로만 이루어진 문자 제거
Everytime_df <- Everytime_df %>%
  dplyr::mutate(content = gsub("[^가-힣]+", " ", content),      # 한글이 아닌 것들 제거
                target_content = gsub("  ", " ", content))

head(Everytime_df)
```

```
> head(Everytime_df)
     type            datetime
1    post 2020-10-05 12:00:00
2 comment 2020-10-05 12:00:00
3    post 2020-10-05 12:00:00
4 comment 2020-10-05 12:00:00
5 comment 2020-10-05 12:00:00
6 comment 2020-10-05 12:00:00
                                                                  content post_num
1 아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다          1
2                                                    뭐 심심해서 할수도 있죠          1
3                      엄마랑 싸웠는 데 내가 사과해야되는건지 좀 판단해줄 사람 있오          2
4                                                  어떻게 싸웠는데          2
5                                                         쪽지함까          2
6                                                                           2
                                                           target_content
1 아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다
2                                                    뭐 심심해서 할수도 있죠
3                      엄마랑 싸웠는 데 내가 사과해야되는건지 좀 판단해줄 사람 있오
4                                                  어떻게 싸웠는데
5                                                         쪽지함까
6
```

그렇다면, Text mining을 하기 위해서 가장 먼저 Tokenization을 실시하도록 한다. 가장 먼저는 .,!?~과 같은 특수 문자를 제거하고, ㅋㅋ나 ㅜㅜ와 같이 자음 또는 모음으로만 표현된 단어를 제거한다. 또한 영어나 이모티콘과 같이 한글이 아닌 문자와 함께 숫자도 제거해주도록 한다. 이렇게 정리된 데이터를 target_content 라는 새로운 변수로 만들도록 한다.

```{r}
# KoNLP 패키지의 SimplePos09로 형태소를 분리하기
Everytime_df <- Everytime_df %>%
  unnest_tokens(word_type, target_content, SimplePos09)

Everytime_df %>%
  filter(content == Everytime_df$content[1])
```

```
         word_type
1          아만다/n
1.1          가입/n
1.2          심사/n
1.3       받/p+는/e
1.4          사람/n
1.5       중/n+에/j
1.6          우리/n
1.7          학교/n
1.8          사람/n
1.9   보/p+아ㄴ데/e
1.10           참/i
1.11     민망하군/n
1.12 마주치/p+지/e
1.13         맙시/n
1.14           다/m
```

그리고 나서는 한글 자연어 분석 패키지인 KoNLP 패키지를 사용하여 분석을 수행하도록 한다. KoNLP 패키지에는 SimplePos09 라는 함수가 있는데, 이는 9개의 품사 태그를 달아주는 함수이다. 9개 품사는 기호(S), 외국어(F), 체언(N), 용언(P), 수식언(M), 독립언(I), 관계언(J), 어미(E), 접사(X)로 나뉜다. 따라서 위에서 특수 문자와 자음 또는 모음으로만 표현된 단어를 제거한 target_content 변수에 대해 결과를 확인한다. 그 결과 가장 첫번째 content의 글에 대해 단어와 그 품사가 달려진 결과를 확인할 수 있다.

```r
```{r}
# 체언과 용언만 남기기
noun <- Everytime_df %>%
  filter(str_detect(word_type, '/n')) %>%
  mutate(word = str_remove(word_type, '/.*$'))

verb <- Everytime_df %>%
  filter(str_detect(word_type, '/p')) %>%
  mutate(word = str_replace_all(word_type, '/.*$', '다'))

bind_rows(head(noun, 5), head(verb, 5))
```
```

```
> bind_rows(head(noun, 5), head(verb, 5))
        type            datetime
1       post 2020-10-05 12:00:00
1.1     post 2020-10-05 12:00:00
1.2     post 2020-10-05 12:00:00
1.4     post 2020-10-05 12:00:00
1.5     post 2020-10-05 12:00:00
1.3     post 2020-10-05 12:00:00
1.9     post 2020-10-05 12:00:00
1.12    post 2020-10-05 12:00:00
2.2  comment 2020-10-05 12:00:00
3.1     post 2020-10-05 12:00:00
                                                                                content post_num
1    아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다          1
1.1  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다          1
1.2  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다          1
1.4  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다          1
1.5  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다          1
1.3  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다          1
1.9  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다          1
1.12 아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다          1
2.2                                                          뭐 심심해서 할수도 있죠          1
3.1                          엄마랑 싸웠는 데 내가 사과해야되는건지 좀 판단해줄 사람 있오          2
             word_type     word
1             아만다/n   아만다
1.1             가입/n     가입
1.2             심사/n     심사
1.4             사람/n     사람
1.5           중/n+에/j       중
1.3           받/p+는/e     받다
1.9       보/p+아ㄴ데/e     보다
1.12      마주치/p+지/e 마주치다
2.2  하/p+ㄹ/e+수/n+도/j     하다
3.1       싸우/p+엇는/e   싸우다
```

품사를 추출한 후에는 모든 단어들이 필요하지는 않으므로, 체언(N)과 용언(P)만 추출하여 확인해보고자 한다. 체언은 문장에서 주체의 구실을 하는 단어로 일반적으로 명사, 대명사가 이에 속한다고 한다. 또한 용언은 독립된 뜻을 가지고, 서술어의 기능 즉, 동사의 기능을 하는 단어라고 할 수 있다. 따라서 문장에서 중요하다고 할 수 있는 명사와 동사를 추출하여 텍스트를 분석해보고자 한다. 체언과 용언을 추출하여 결과를 확인해보면, 아만다, 가입, 심사 등과 같은 명사와 받다, 보다, 마주치다 등의 동사가 나오는 것을 볼 수 있다.

```r
```{r}
# 불용어 제거하기
Korean_stop_words <- readLines('https://raw.githubusercontent.com/stopwords-iso/stopwords-ko/master
/stopwords-ko.txt')
Korean_stop_words <- data.frame(data.frame(Korean_stop_words))

Korean_stop_words2 <- as.data.frame(stopwords::stopwords('ko', source = 'marimo'))
colnames(Korean_stop_words2) <- colnames(Korean_stop_words)[1]
Korean_stop_words <- rbind(Korean_stop_words, Korean_stop_words2)

Everytime_df <- bind_rows(noun, verb) %>%
  arrange(desc(type)) %>%
  arrange(desc(datetime)) %>%
  filter(nchar(word) > 1) %>%
  select(-word_type) %>%
  anti_join(Korean_stop_words, by = c('word' = 'Korean_stop_words'))

head(Everytime_df, 7)
```
```

```
> head(Everytime_df, 7)
     type            datetime
1    post 2020-10-05 12:00:00
1.1  post 2020-10-05 12:00:00
1.2  post 2020-10-05 12:00:00
1.4  post 2020-10-05 12:00:00
1.7  post 2020-10-05 12:00:00
1.8  post 2020-10-05 12:00:00
1.11 post 2020-10-05 12:00:00
                                                                                content post_num
1    아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다         1
1.1  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다         1
1.2  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다         1
1.4  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다         1
1.7  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다         1
1.8  아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다         1
1.11 아만다 가입 심사 받는 사람 중에 우리 학교 사람 봤는데 참 민망하군 마주치지 맙시다         1
        word
1       아만다
1.1       가입
1.2       심사
1.4       사람
1.7       학교
1.8       사람
1.11 민망하군
```

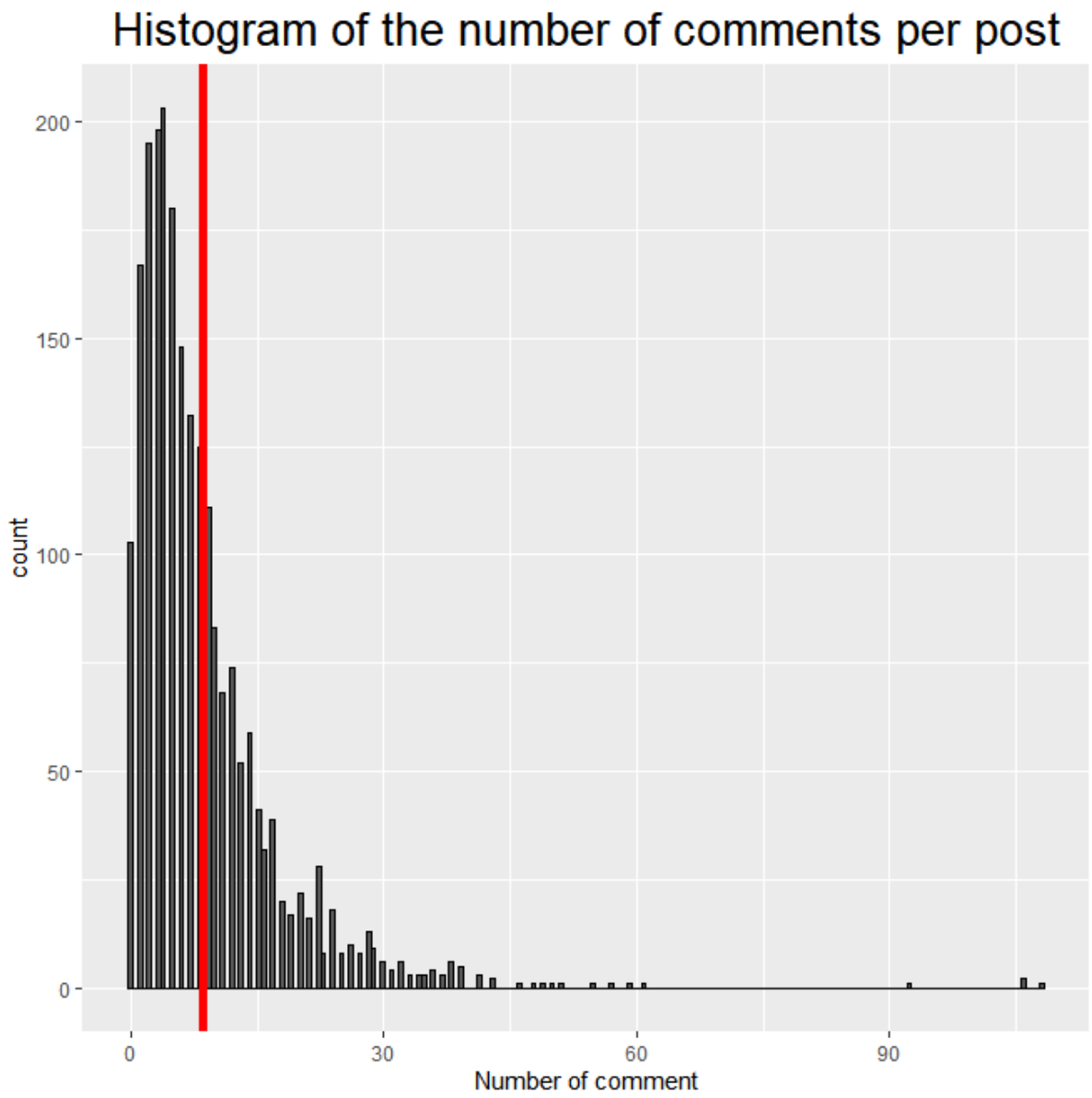
최종적으로는 한국어의 불용어를 제거해주도록 한다. 영어와는 다르게 사전에 정리된 패키지 등이 제한적이므로, 구글링을 통해서 다른 사람들이 정리해놓은 데이터를 불러와서 불용어 사전을 만든다. 그리고 앞서 정리해둔 데이터에 대해 불용어를 anti_join 하여 의미 없는 정보들은 최대한 제거하도록 한다.

2. 데이터 탐색 및 결과 설명

```r
```{r}
# 하나의 글에 댓글이 평균적으로 몇 개 정도 달리는지?
comment_mean <- ori_Everytime_df %>%
  group_by(post_num) %>%
  dplyr::mutate(post_num_sum = sum(n()-1)) %>%
  group_by(post_num) %>%
  dplyr::summarise(post_num = unique(post_num_sum)) %>%
  dplyr::summarise(mean = mean(post_num)) %>%
  unlist() %>%
  unname()

comment_mean

ori_Everytime_df %>%
  group_by(post_num) %>%
  dplyr::mutate(post_num_sum = sum(n()-1)) %>%
  group_by(post_num) %>%
  dplyr::summarise(post_num = unique(post_num_sum)) %>%
  ggplot(aes(x = post_num)) +
  geom_histogram(bins = 200, color = 'black') +
  geom_vline(xintercept = comment_mean, color="red",  size = 2)+
  labs(title="Histogram of the number of comments per post",
       x= "Number of comment") +
  theme(plot.title = element_text(hjust=0.5, size = 20))
```
```



가장 먼저는 문제에서 주어진 것처럼 하나의 글에 대해 댓글이 평균적으로 얼마나 달리는지 확인해보았다. 위에서 만들었던 post_num 변수를 기준으로 묶고, n()을 하게 되면 개수를 확인할 수 있는데 각 데이터에는 post의 값이 1개씩은 들어가 있으므로 1을 빼주도록 한다. 이렇게 되면 순수하게 댓글의 개수만을 추릴 수 있다. 그리고 이에 대한 평균값을 계산해보면, 최종적으로 약 8.45 정도의 값을 얻을 수 있다. 이를 조금 더 직관적으로 확인하기 위해 히스토그램을 표현해보았는데, 전반적으로 30개 미만의 댓글이 달리고 있는 것을 확인할 수 있다.

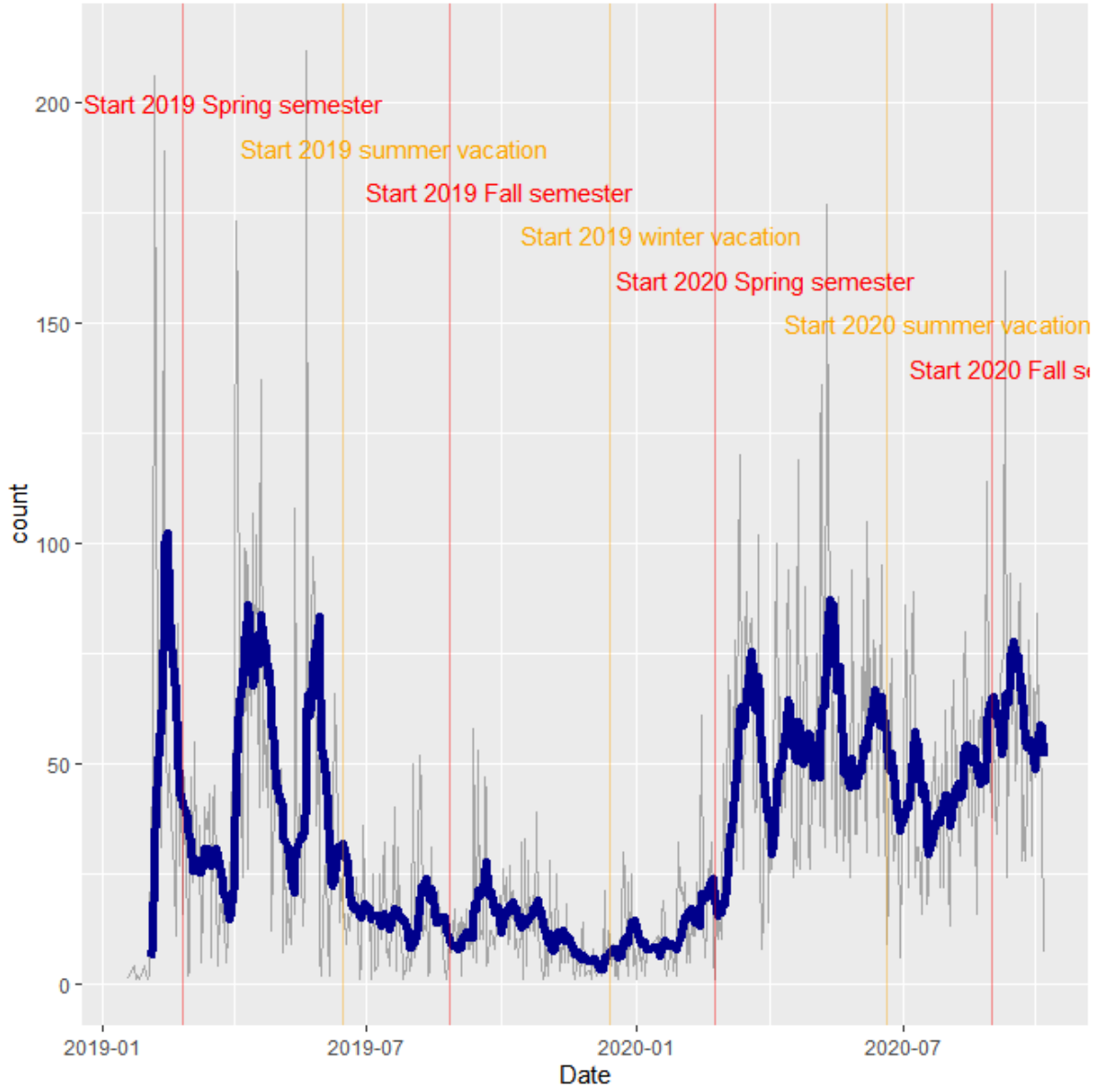
```r
# 학기 중과 방학 중에 글의 수
ori_Everytime_df %>%
  group_by(year(datetime)) %>%
  dplyr::summarise(sum = n())
```

```
> ori_Everytime_df %>%
+   group_by(year(datetime)) %>%
+   dplyr::summarise(sum = n())
# A tibble: 6 x 2
  `year(datetime)`   sum
             <dbl> <int>
1             2015    15
2             2016    21
3             2017    15
4             2018    67
5             2019  8755
6             2020 12376
```

다음으로는 학기 중과 방학 중에 생기는 글의 개수를 비교해보도록 한다. 이를 위해서 year 별로 얼마나 데이터가 분포되어 있는지를 확인해보았다. 결과를 보니, 2015년부터 2018년에는 충분한 데이터가 없었기 때문에 2019년과 2020년에 대해서 결과를 확인해보도록 한다.

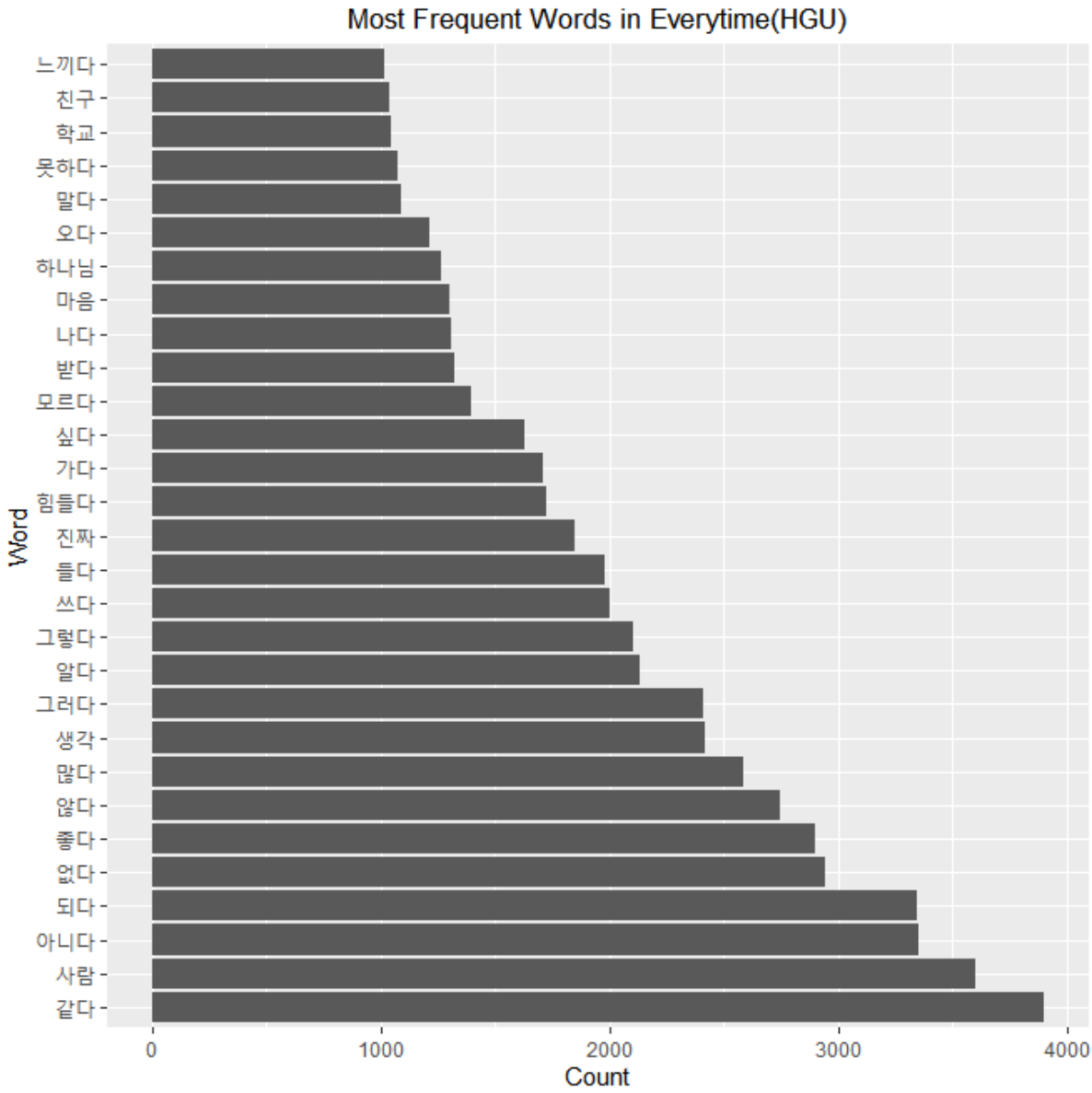
```r
```{r}
# 학기 중과 방학 중에 글의 수
ori_Everytime_df %>%
  filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', 'vacation',
                        ifelse(date(datetime) < '2019-06-15', 'semester',
                          ifelse(date(datetime) < '2019-08-26', 'vacation',
                            ifelse(date(datetime) < '2019-12-14', 'semester',
                              ifelse(date(datetime) < '2020-02-24', 'vacation',
                                ifelse(date(datetime) < '2020-06-20', 'semester',
                                  ifelse(date(datetime) < '2020-08-31', 'vacation',
'semester')))))))) %>%
  group_by(date(datetime)) %>%
  dplyr::summarise(count = n()) %>%
  dplyr::rename(Date = 'date(datetime)') %>%
  ggplot(aes(x = Date, y = count)) + geom_line(alpha = 0.3) +
  geom_ma(ma_fun = SMA, n = 10, size = 2, linetype = 1) +
  geom_vline(xintercept = as.numeric(ymd('2019-02-25')), color = 'red', size = 0.5, alpha = 0.4) +
  geom_text(aes(x = as.Date('2019-02-25'), y = 200, label = 'Start 2019 Spring semester'),
            size = 4, color = 'red', nudge_x = 35, check_overlap = TRUE) +
  geom_vline(xintercept = as.numeric(ymd('2019-06-15')), color = 'orange', size = 0.5, alpha = 0.4)
+
  geom_text(aes(x = as.Date('2019-06-15'), y = 190, label = 'Start 2019 summer vacation'),
            size = 4, color = 'orange', nudge_x = 35, check_overlap = TRUE) +
  geom_vline(xintercept = as.numeric(ymd('2019-08-26')), color = 'red', size = 0.5, alpha = 0.4) +
  geom_text(aes(x = as.Date('2019-08-26'), y = 180, label = 'Start 2019 Fall semester'),
            size = 4, color = 'red', nudge_x = 35, check_overlap = TRUE) +
  geom_vline(xintercept = as.numeric(ymd('2019-12-14')), color = 'orange', size = 0.5, alpha = 0.4)
+
  geom_text(aes(x = as.Date('2019-12-14'), y = 170, label = 'Start 2019 winter vacation'),
            size = 4, color = 'orange', nudge_x = 35, check_overlap = TRUE) +
  geom_vline(xintercept = as.numeric(ymd('2020-02-24')), color = 'red', size = 0.5, alpha = 0.4) +
  geom_text(aes(x = as.Date('2020-02-24'), y = 160, label = 'Start 2020 Spring semester'),
            size = 4, color = 'red', nudge_x = 35, check_overlap = TRUE) +
  geom_vline(xintercept = as.numeric(ymd('2020-06-20')), color = 'orange', size = 0.5, alpha = 0.4)
+
  geom_text(aes(x = as.Date('2020-06-20'), y = 150, label = 'Start 2020 summer vacation'),
            size = 4, color = 'orange', nudge_x = 35, check_overlap = TRUE) +
  geom_vline(xintercept = as.numeric(ymd('2020-08-31')), color = 'red', size = 0.5, alpha = 0.4) +
  geom_text(aes(x = as.Date('2020-08-31'), y = 140, label = 'Start 2020 Fall semester'),
            size = 4, color = 'red', nudge_x = 35, check_overlap = TRUE)

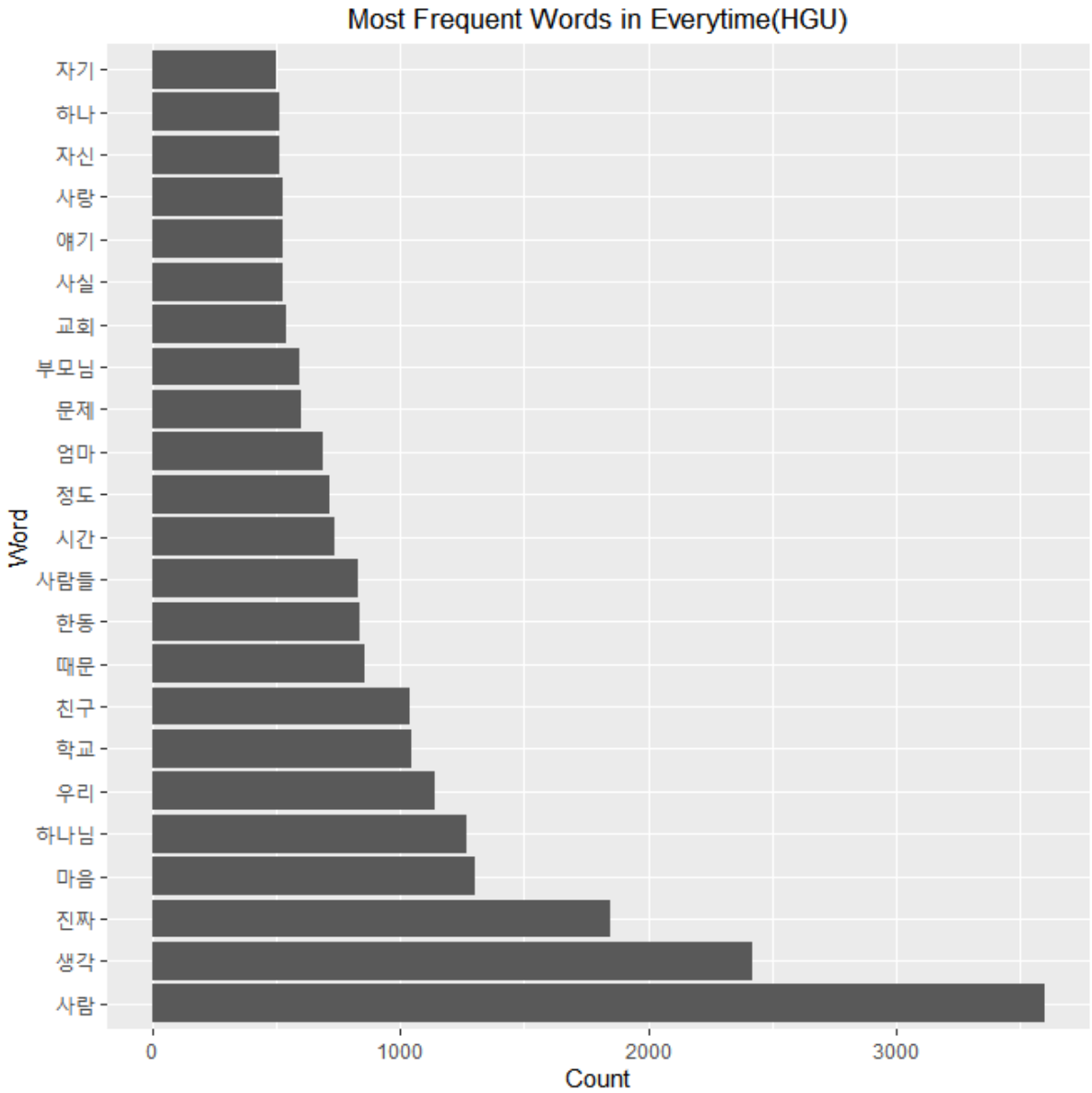
```
```



2019년과 2020년에 대하여 히즈넷에서 학사정보를 확인해서 각각 학기와 방학을 의미하는 label 변수를 만들어준다. 그리고 각 날짜별로 글과 댓글이 생긴 갯수를 count 하여 시계열의 형태로 그래프를 그려주도록 한다. 결과를 확인해보니, 생각보다 변동성이 심해서 alpha를 주고, 10일 이동평균선으로 값을 확인하도록 한다. 수직선으로 학기와 방학의 시작을 표시했다. 전반적으로 확인할 수 있는 것은 방학보다 학기 중에 더 많은 글들이 생긴 것을 볼 수 있다. 또한 학기가 시작하기 직전과 학기 중반에 급격하게 글이 많아지는 것을 확인할 수 있다.

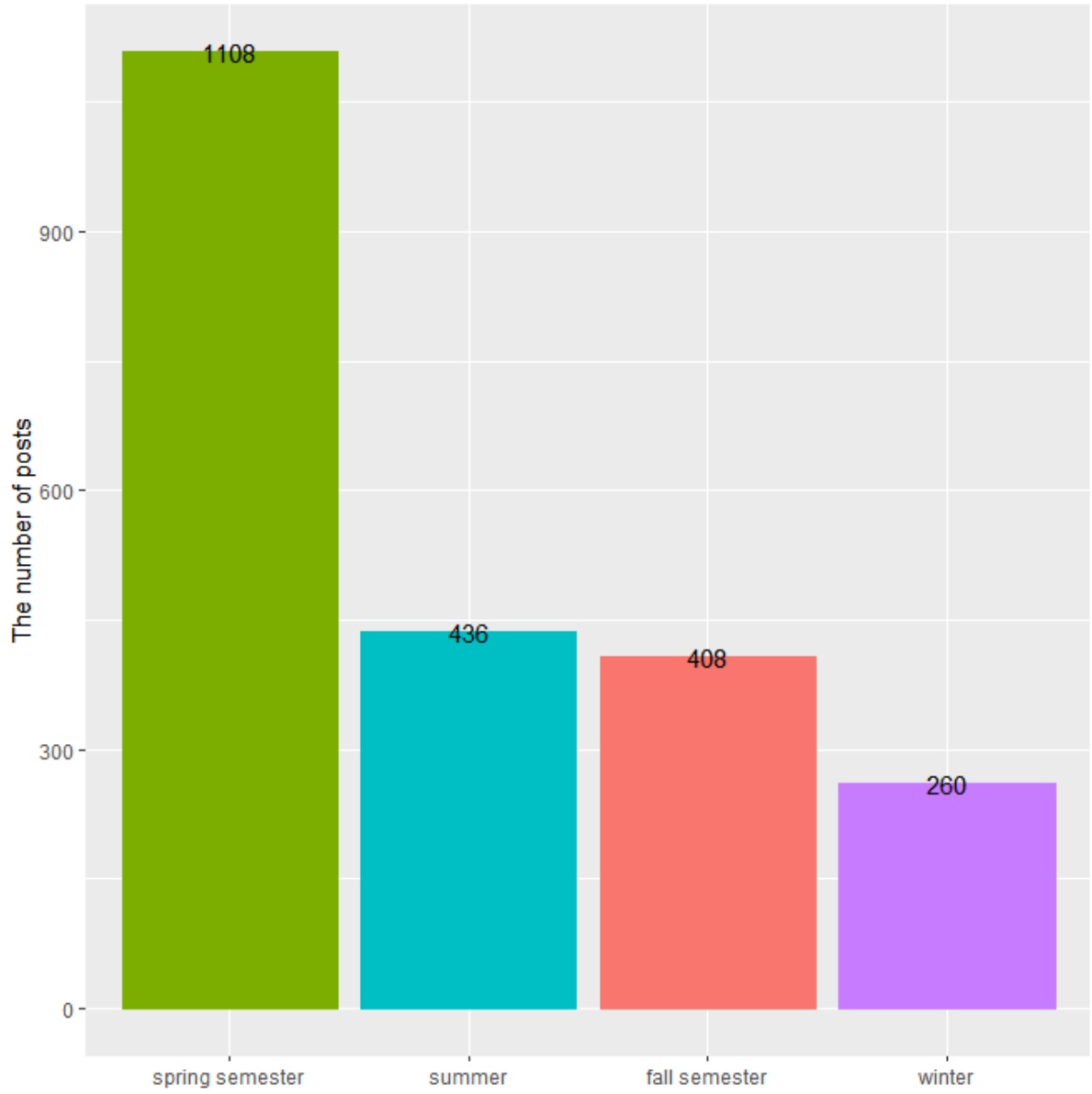
```{r}
# post와 comment에서 많이 사용된 단어
Everytime_df %>%
  dplyr::count(word, sort = T) %>%
  dplyr::filter(str_length(word) >= 2 & n > 1000) %>%
  ggplot(aes(x = reorder(word, -n), y = n)) +
  geom_col() + xlab("Word") + ylab("Count") +
  ggtitle("Most Frequent Words in Everytime(HGU)")+
  coord_flip() +
  theme(plot.title = element_text(hjust=0.5))

noun %>%
  dplyr::count(word, sort = T) %>%
  dplyr::filter(str_length(word) >= 2 & n > 500) %>%
  ggplot(aes(x = reorder(word, -n), y = n)) +
  geom_col() + xlab("Word") + ylab("Count") +
  ggtitle("Most Frequent Words in Everytime(HGU)")+
  coord_flip() +
  theme(plot.title = element_text(hjust=0.5))
```





이번에는 한동의 학생들이 post와 comment 모두에서 가장 많이 사용하는 단어는 무엇인지 확인해보고자 한다. 확인해보니, 용언은 단순히 동사로써 그 의미를 이해하기 조금 힘들어서 체언인 명사로 다시 결과를 확인하도록 한다. 또한 단어가 하나인 단어는 제외하고, 500번 이상 나온 단어를 바탕으로 시각화를 수행해보았다. 그 결과를 보면, 사람, 친구, 자신, 엄마, 부모님 등 사람에 관련된 이야기를 비롯하여 한동, 학교와 같이 우리 대학을 지칭하는 단어들도 많이 나왔다. 뿐만 아니라, 생각, 얘기, 시간, 문제 등의 단어로 보아 자신들의 고민들을 이야기하는 빈도도 상당히 높은 것으로 추정된다.

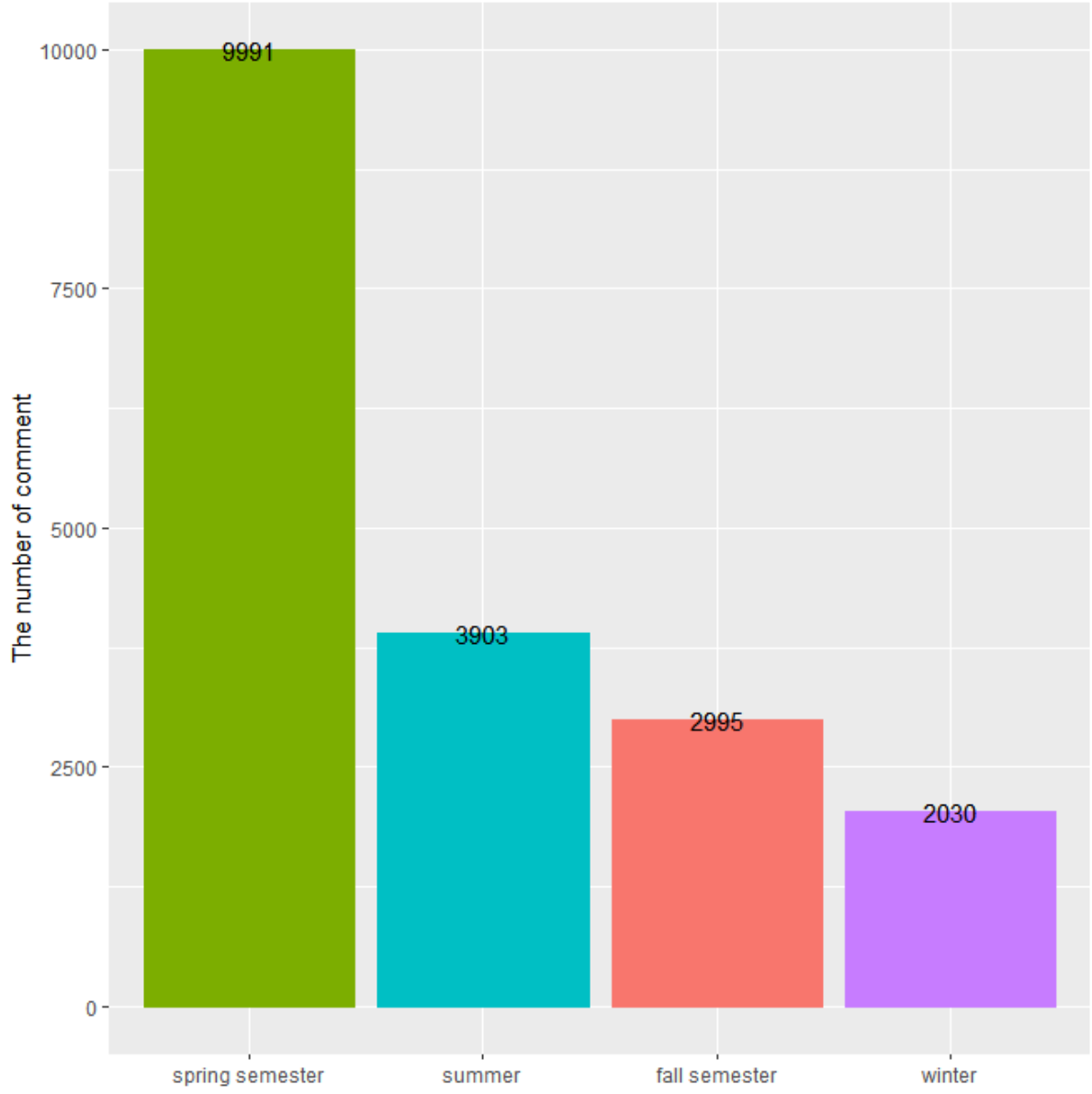
```r
# 학기 중과 방학 중의 게시글(POST) 개수의 차이
ori_Everytime_df %>%
  filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', 'winter',
                        ifelse(date(datetime) < '2019-06-15', 'spring semester',
                          ifelse(date(datetime) < '2019-08-26', 'summer',
                            ifelse(date(datetime) < '2019-12-14', 'fall semester',
                              ifelse(date(datetime) < '2020-02-24', 'winter',
                                ifelse(date(datetime) < '2020-06-20', 'spring semester',
                                  ifelse(date(datetime) < '2020-08-31', 'summer', 'fall semester')))))))) %>%
  filter(type == 'post') %>%
  group_by(label) %>%
  count() %>%
  ggplot(aes(reorder(label, -n), n, col = label, fill = label)) +
  geom_col()+
  theme(legend.position = 'none')+
  labs(x = NULL, y = 'The number of posts')+
  geom_text(aes(label= n), col = "black")
```



이번에는 2019년과 2020년의 봄과 가을 학기 그리고 여름과 겨울 방학으로 나누어서 게시글(POST)의 차이를 확인한다. 전반적으로 봄 학기에 글들이 많이 만들어진 것을 볼 수 있고, 그 다음으로는 여름방학, 가을학기, 겨울방학의 순이다. 상대적으로 봄 학기의 게시글들이 다른 시기보

다 많았던 것을 확인할 수 있다.

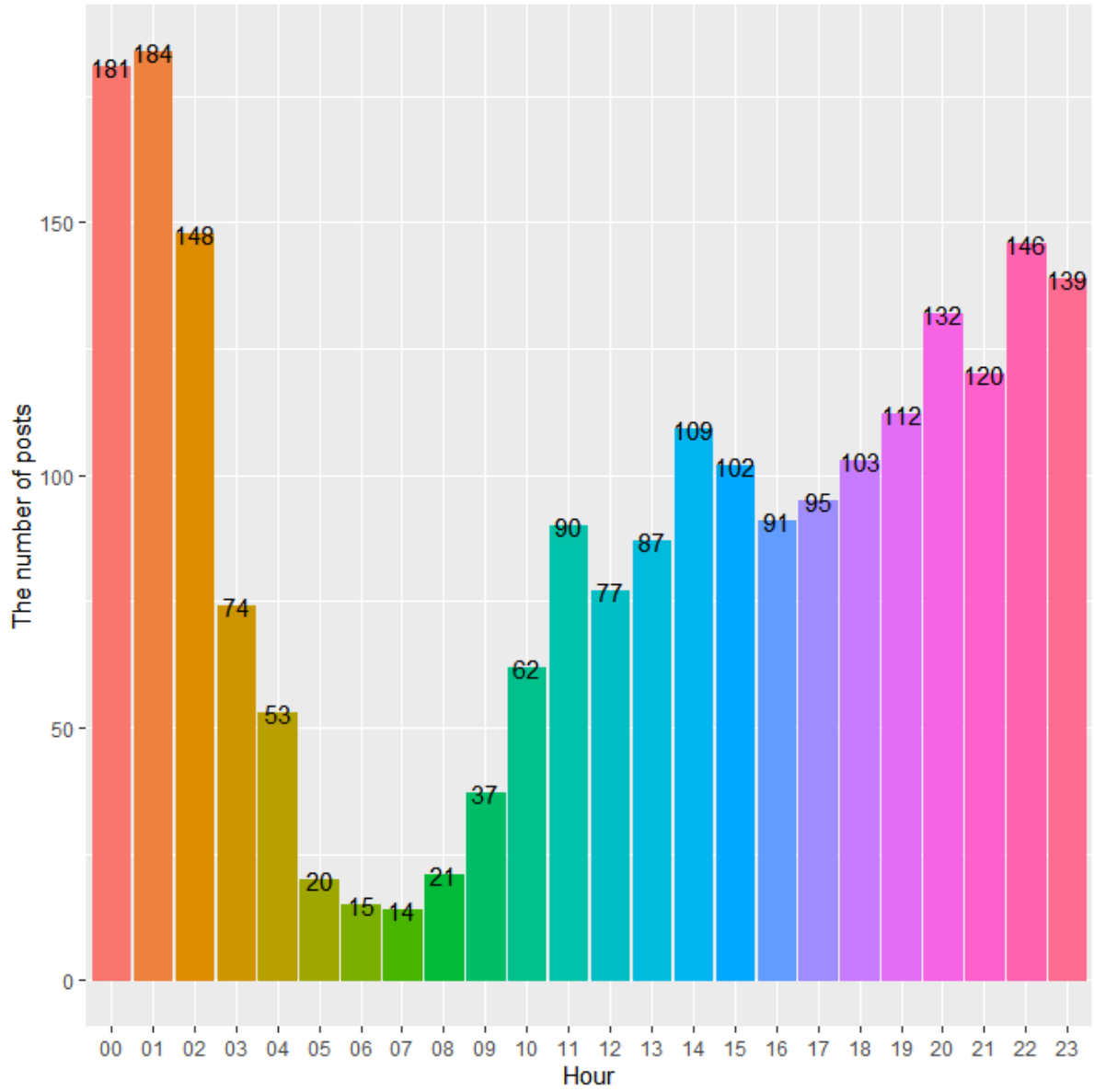
```r
# 학기 중과 방학 중의 게시글(POST) 개수의 차이
ori_Everytime_df %>%
  filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', 'winter',
                          ifelse(date(datetime) < '2019-06-15', 'spring semester',
                            ifelse(date(datetime) < '2019-08-26', 'summer',
                              ifelse(date(datetime) < '2019-12-14', 'fall semester',
                                ifelse(date(datetime) < '2020-02-24', 'winter',
                                  ifelse(date(datetime) < '2020-06-20', 'spring semester',
                                    ifelse(date(datetime) < '2020-08-31', 'summer', 'fall semester')))))))) %>%
  dplyr::mutate(hour = format(datetime, '%H')) %>%
  filter(type == 'comment') %>%
  group_by(label) %>%
  count() %>%
  ggplot(aes(reorder(label, -n), n, col = label, fill = label)) +
  geom_col()+
  theme(legend.position = 'none')+
  labs(x = NULL, y = 'The number of comment')+
  geom_text(aes(label= n), col = "black")
```



그리고 댓글에 대한 것도 시기 별로 개수를 확인해보았다. 댓글 개수의 추이는 게시글들이 많을

수록 그 댓글 수도 증가한 것을 볼 수 있다. 결과적으로 게시글과 댓글의 개수가 봄학기 - 여름방학 - 가을학기 - 겨울방학 순으로 많은 것을 확인할 수 있다.

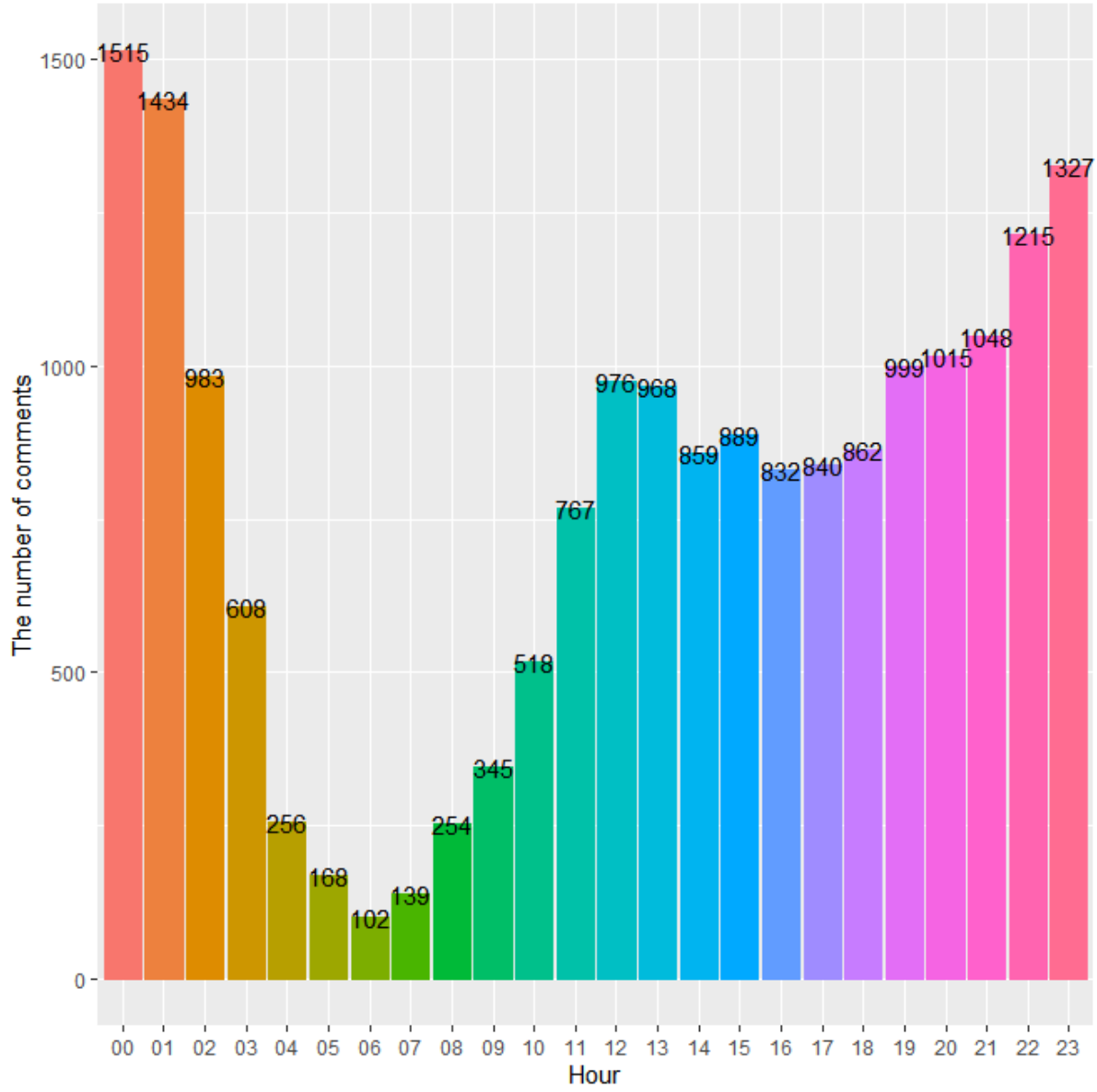
```r
# 00시부터 24시까지 시간에 따라 게시글의 개수 비교
ori_Everytime_df %>%
  filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(hour = format(datetime, '%H')) %>%
  filter(type == 'post') %>%
  group_by(hour) %>%
  count() %>%
  ggplot(aes(hour, n, col = hour, fill = hour)) +
  geom_col()+
  theme(legend.position = 'none')+
  labs(x = 'Hour', y = 'The number of posts')+
  geom_text(aes(label= n), col = "black")
```



그리고 00시부터 24시까지 학생들이 시간에 따라 얼마나 게시글을 작성하는지 그 분포를 확인해보고자 한다. 눈에 띄게 확인할 수 있는 것은 일반적으로 취침시간이라고 할 수 있는 4시부터 9시까지는 게시글의 수가 확연히 감소하는 것을 볼 수 있다. 또한 저녁 8시부터 취침 전 약 새벽

3시까지 게시글의 개수가 가장 많은 것을 확인할 수 있다.

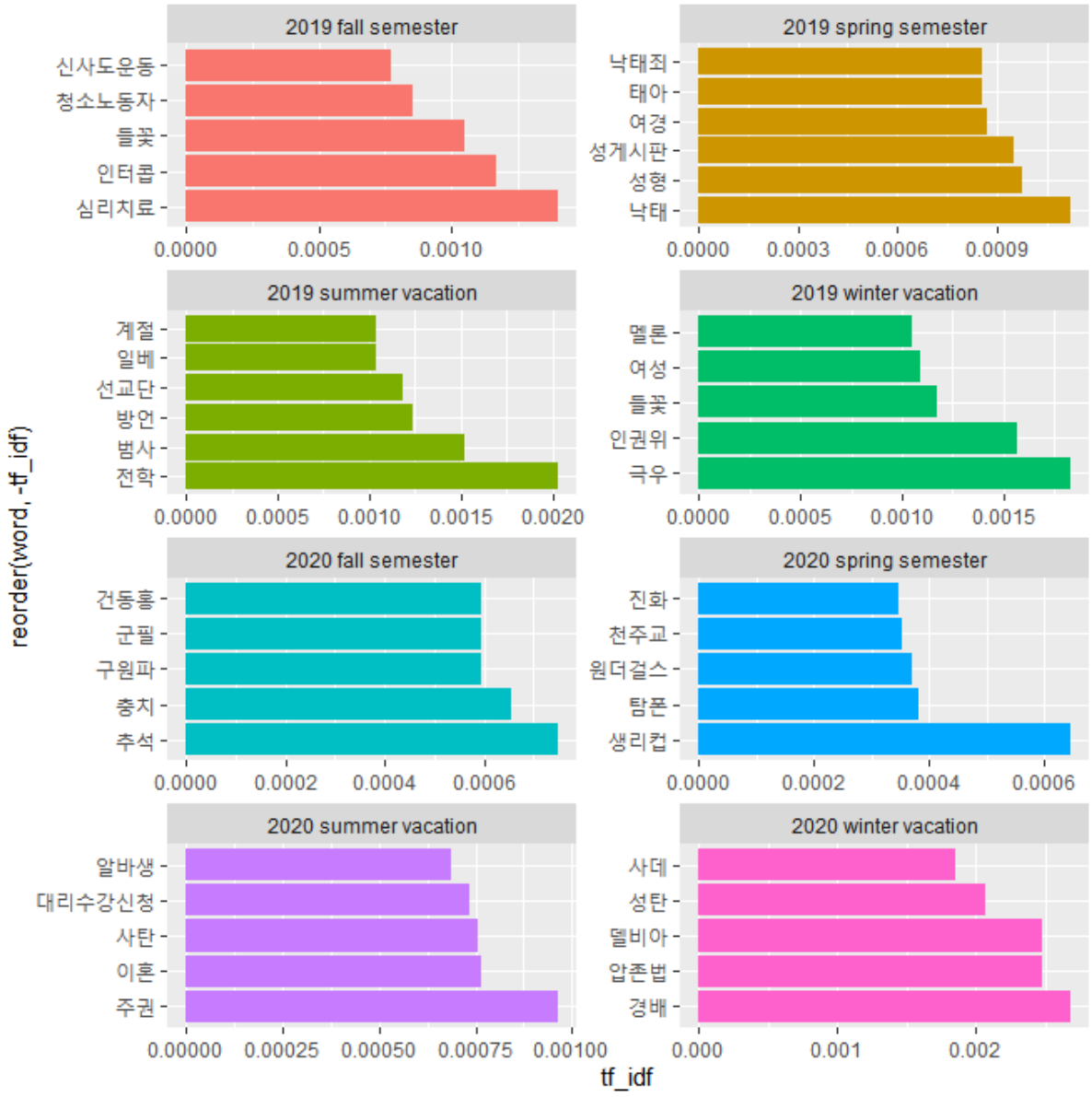
```{r}
# 00시부터 24시까지 시간에 따라 댓글의 개수 비교
ori_Everytime_df %>%
  filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(hour = format(datetime, '%H')) %>%
  filter(type == 'comment') %>%
  group_by(hour) %>%
  count() %>%
  ggplot(aes(hour, n, col = hour, fill = hour)) +
  geom_col()+
  theme(legend.position = 'none')+
  labs(x = 'Hour', y = 'The number of comments')+
  geom_text(aes(label= n), col = "black")
```



시간대별 댓글의 개수 역시 위에서 봤던 것처럼 게시글과 거의 선형적으로 움직이는 것을 확인할 수 있다.

3. 전체 구간을 학기와 방학으로 나누어 TF-IDF 계산

```r
df_tf_idf <- Everytime_df %>%
  dplyr::filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', '2019 winter vacation',
                        ifelse(date(datetime) < '2019-06-15', '2019 spring semester',
                          ifelse(date(datetime) < '2019-08-26', '2019 summer vacation',
                            ifelse(date(datetime) < '2019-12-14', '2019 fall semester',
                              ifelse(date(datetime) < '2020-02-24', '2020 winter vacation',
                                ifelse(date(datetime) < '2020-06-20', '2020 spring semester',
                                  ifelse(date(datetime) < '2020-08-31', '2020 summer vacation', '2020 fall semester'))))))))
%>%
  group_by(label) %>%
  dplyr::count(word, sort = T) %>%
  dplyr::filter(str_length(word) >= 2) %>%
  group_by(label) %>%
  dplyr::mutate(total = sum(n)) %>%
  bind_tf_idf(word, label, n)%>%
  dplyr::arrange(desc(tf_idf))

df_tf_idf %>%
  dplyr::mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(label) %>% top_n(5) %>% ungroup() %>%
  ggplot(aes(x = reorder(word, -tf_idf), y = tf_idf, fill = label)) +
  geom_col(show.legend = F) + labs(X = NULL, y = 'tf_idf') +
  facet_wrap(~label, ncol = 2, scales = 'free') + coord_flip()
```



이번에는 전체 구간을 학기와 방학으로 나누어서 TF-IDF를 계산해보도록 한다. 앞선 문제와 동일하게 2019년 전에는 데이터가 충분하지 않으므로, 2019년과 2020년 데이터만 활용하도록 한다. 또한 히즈넷의 학사정보에서 각 학기 별로 학기의 시작과 방학이 언제였는지를 label 변수로 만들어주도록 한다. 이를 바탕으로 각 학기 및 방학 별로 등장하는 단어들의 개수를 count 하여 tf-idf 값을 계산하도록 한다. tf-idf는 Term Frequency 즉, 어떤 단어가 특정 문서에서 얼마나 많이 쓰였는지와 Inverse Document Frequency 즉, 특정 문서에서 집중적으로 나오는 단어는 무엇인지 의미하는 두 변수를 곱해서 값을 계산한다. 그리고 이 값이 클수록 다른 문서에서는 적고, 해당 문서에서 자주 등장하는 것을 의미한다. 이 문제의 경우에는 다른 학기 및 방학보다 해당 시점에서 더 많이 등장하는 단어들이 무엇인지를 확인해보는 것이다. 결과를 확인해보면, 다음과 같다.

- 2019 winter vacation: 들꽃, 인권위 - 들꽃 단체의 논란이 된 행사에 대한 관심도 증가

- 2019 spring semester: 낙태, 태아, 성형, 여경 - 낙태죄가 7년만에 합헌에서 위헌이 됨

- 2019 summer vacation: 전학, 범사, 방언, 선교단 - 방학 때 수련회 등을 참석한 것(?)

- 2019 fall semester: 들꽃, 청소노동자, 인터콥, 심리치료 - 인터콥 단체의 이단 이슈 논란, 들꽃 단체의 활동에 대한 관심 증가

- 2020 winter vacation: 경배, 성탄 - 크리스마스를 맞이한 학생들의 관심도 증가

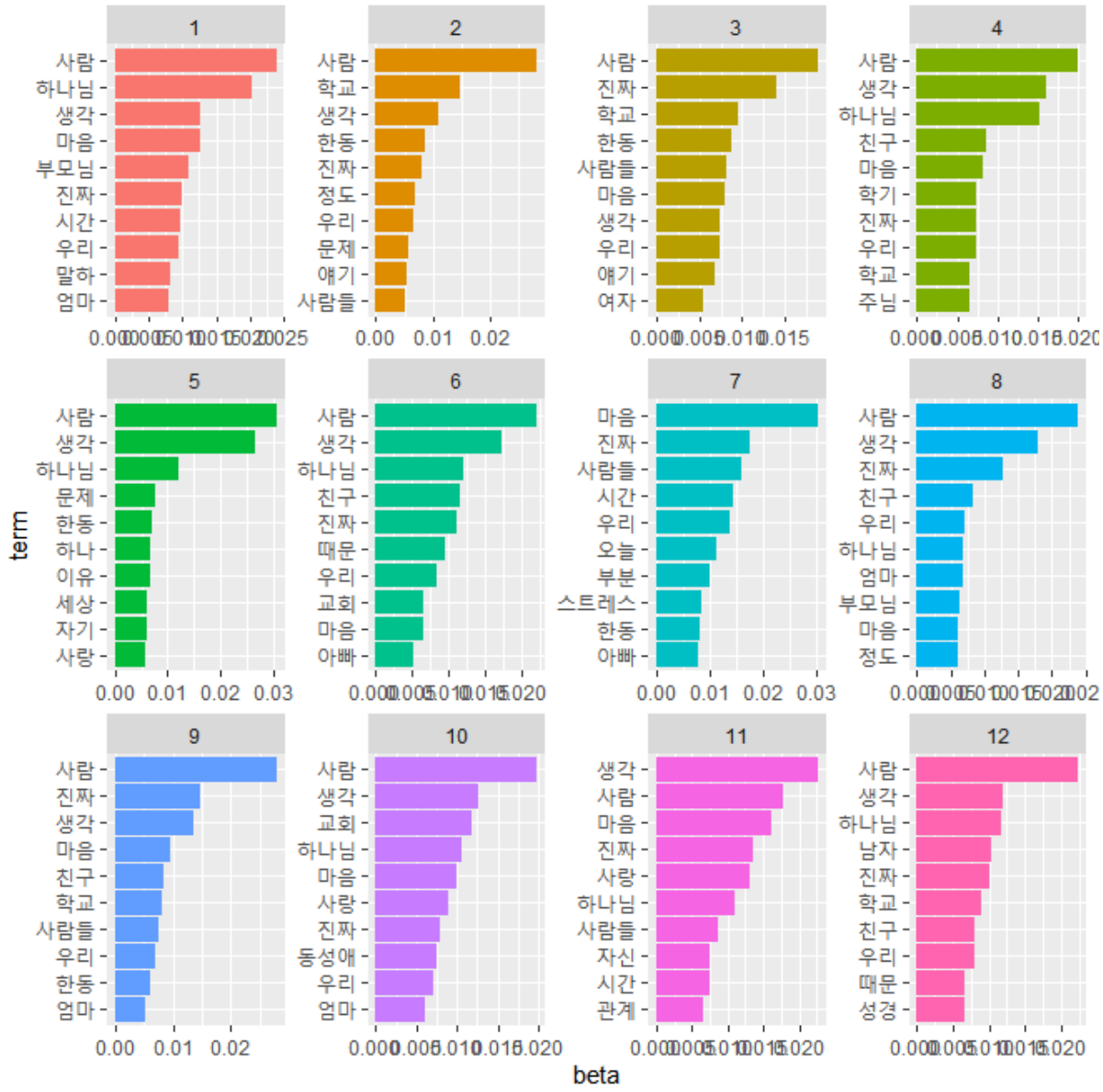
- 2020 spring semester: 천주교, 생리컵, 진화 - 특별한 연관도 모르겠음

- 2020 summer vacation: 주권, 이혼, 사탄, 대리수강신청 - 학기 시작 전 수강신청에 대한 관심도 증가

- 2020 fall semester: 구원파, 추석, 충치 - 특별한 연관도 모르겠음

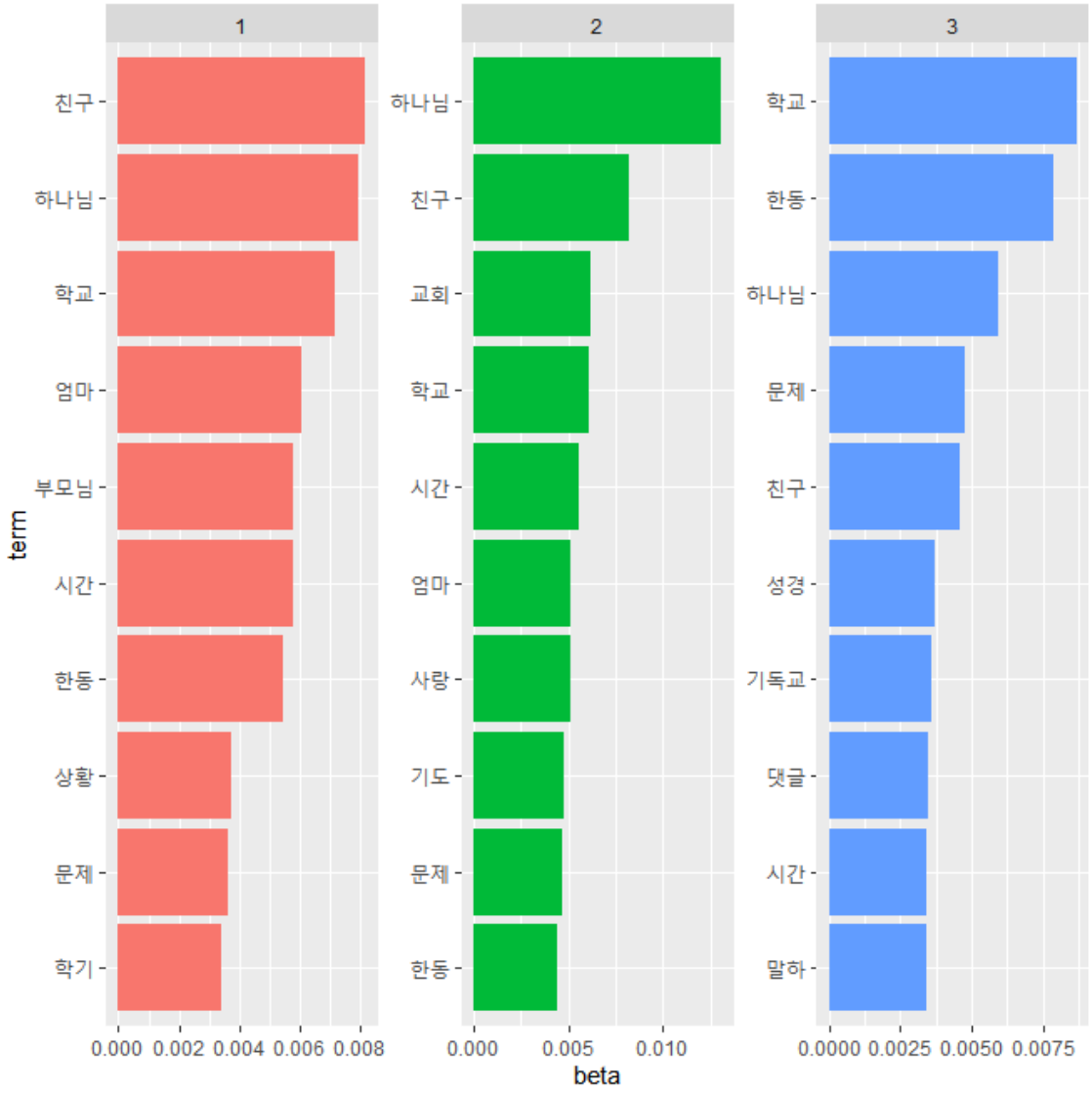
4. Topic modeling 수행 - 주로 어떠한 주제들이 언급되는지 유추

```r
label_dtm <- noun %>%
  dplyr::filter(str_length(word) >= 2) %>%
  dplyr::filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', '2019 winter vacation',
                        ifelse(date(datetime) < '2019-06-15', '2019 spring semester',
                          ifelse(date(datetime) < '2019-08-26', '2019 summer vacation',
                            ifelse(date(datetime) < '2019-12-14', '2019 fall semester',
                              ifelse(date(datetime) < '2020-02-24', '2020 winter vacation',
                                ifelse(date(datetime) < '2020-06-20', '2020 spring semester',
                                  ifelse(date(datetime) < '2020-08-31', '2020 summer vacation', '2020 fall semester')))))))) %>%
  group_by(label) %>%
  dplyr::count(word, sort = T) %>%
  cast_dtm(label, word, n)

label_lda <- LDA(label_dtm, k = 12, control = list(seed = 1234))
label_topics <- tidy(label_lda, matrix = 'beta')

label_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = F) +
  facet_wrap(~topic, scales = 'free') +
  coord_flip() + scale_x_reordered()
```



이번에는 LDA 방식을 통한 Topic modeling을 통해서 어떤 단어들이 함께 묶여서 어떤 주제를 만들고 있는지 확인해본다. 가장 먼저는 12개 topic으로 나누어 결과를 확인해보았는데, 전반적으로 단어의 빈도수가 높았던 '생각, 사람, 진짜' 등의 단어가 거의 대부분의 토픽에 들어간 것을 확인할 수 있다. 따라서 너무 자주 나오는 단어와 의미성이 떨어지는 단어를 제거한 후에 다시 LDA를 수행해보도록 한다.

```r
except_word <- c('사람', '생각', '진짜', '근데', '우리', '얘기', '때문', '정도', '여자', '남자', '그것', '누구',
                 '감사합니', '자신', '사실', '자기', '경우', '마음', '사람들', '이유')

label_dtm <- noun %>%
  dplyr::filter(str_length(word) >= 2) %>%
  dplyr::filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::filter(!word %in% except_word) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', '2019 winter vacation',
                        ifelse(date(datetime) < '2019-06-15', '2019 spring semester',
                          ifelse(date(datetime) < '2019-08-26', '2019 summer vacation',
                            ifelse(date(datetime) < '2019-12-14', '2019 fall semester',
                              ifelse(date(datetime) < '2020-02-24', '2020 winter vacation',
                                ifelse(date(datetime) < '2020-06-20', '2020 spring semester',
                                  ifelse(date(datetime) < '2020-08-31', '2020 summer vacation', '2020 fall semester'))))))) %>%
  group_by(label) %>%
  dplyr::count(word, sort = T) %>%
  cast_dtm(label, word, n)

label_lda <- LDA(label_dtm, k = 3, control = list(seed = 1234))
label_topics <- tidy(label_lda, matrix = 'beta')

label_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  dplyr::arrange(topic, -beta) %>%
  dplyr::mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = F) +
  facet_wrap(~topic, scales = 'free') +
  coord_flip() + scale_x_reordered()
```



위에서 이야기했던 것처럼 단어의 빈도수가 높거나, 의미성이 떨어지는 단어를 담은 except_word 벡터를 정의하여, 해당 단어는 제외하고 다시 3개 토픽을 LDA로 만들어보았다. 안타깝게도 이 결과 역시, 각 토픽들에서 겹치는 단어들이 꽤 있어서 눈에 띄게 어떤 주제와 관련되었다고 유추하기가 힘들었다.

- 1번 토픽: 친구, 엄마, 부모님, 학교, 한동, 사랑 등의 단어로 보아 학교 생활에 있어서 자주 등장하는 사람이나 자신의 고민에 대한 내용인 것 같다.

- 2번 토픽: 사랑, 학교, 기도, 한동, 말씀, 교회, 감사 등의 단어로 보아 기독교적인 단어들의 표현으로 교회와 관련된 주제일 것 같다.

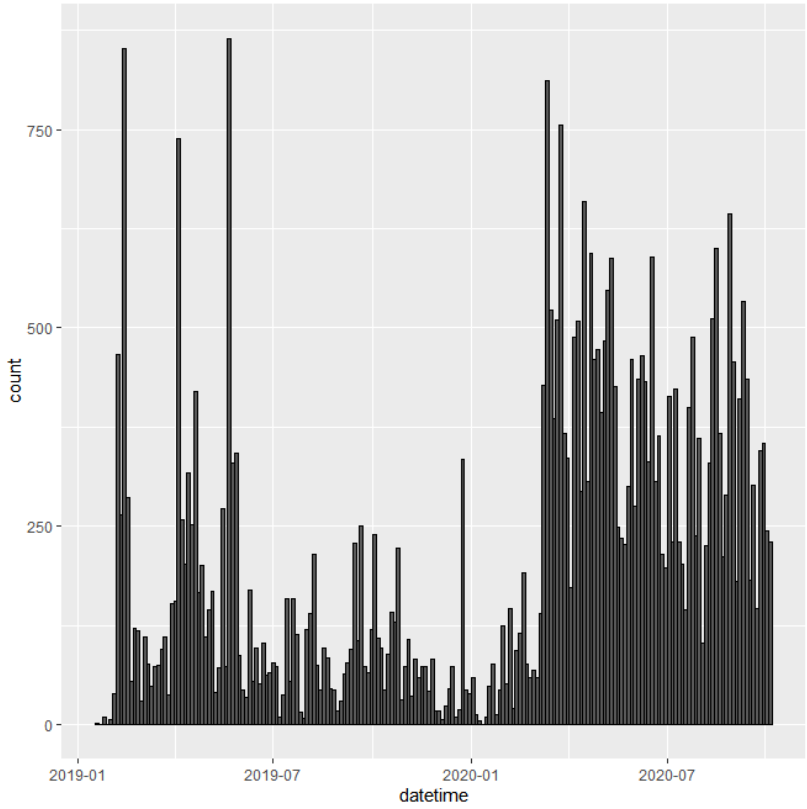
- 3번 토픽: 학교, 한동, 성경, 기독교, 수업, 이해 등의 단어로 보아 2번과 유사하지만, 기독교나 수업, 교수님 등의 단어로 보아 수업과 관련된 주제일 것 같다.

5. 시기별(분기, 월, 주차)로 감성분석을 수행하고, 결과 분석

```{r}
sentiment_words <- read.delim('https://raw.githubusercontent.com/park1200656/KnuSentiLex/master/SentiWord_Dict.txt')
sentiment_words$X1 <- ifelse(is.na(sentiment_words$X1), -1, sentiment_words$X1)
sentiment_words$X... <- ifelse(sentiment_words$X... == '갈등 -1', '갈등', sentiment_words$X...)
colnames(sentiment_words) <- c('word', 'score')

sentiment_df <- Everytime_df %>%
  left_join(sentiment_words, by = 'word')

sentiment_df

sentiment_df %>%
  dplyr::filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', '2019 winter vacation',
                        ifelse(date(datetime) < '2019-06-15', '2019 spring semester',
                          ifelse(date(datetime) < '2019-08-26', '2019 summer vacation',
                            ifelse(date(datetime) < '2019-12-14', '2019 fall semester',
                              ifelse(date(datetime) < '2020-02-24', '2020 winter vacation',
                                ifelse(date(datetime) < '2020-06-20', '2020 spring semester',
                                  ifelse(date(datetime) < '2020-08-31', '2020 summer vacation', '2020 fall semester'))))))) %>%
  dplyr::summarise(na_sum = sum(!is.na(score)))
```

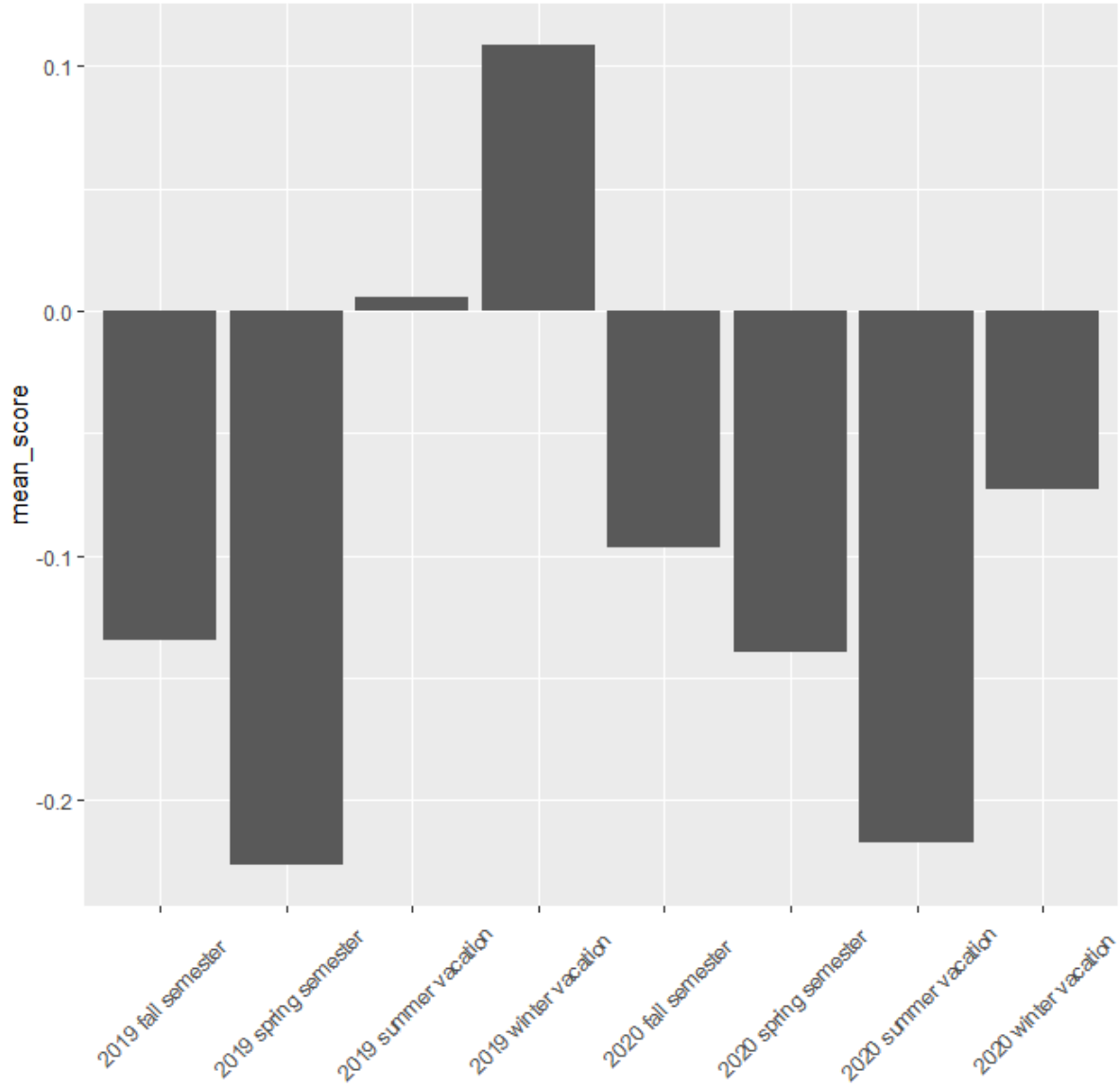
군산대학교 한국어 감성 사전으로부터 데이터를 불러오도록 한다. 확인해보니, 긍정과 부정을 따로 나눠놓은 데이터도 있었고, 각 단어에 대해서 -2부터 2까지 score를 배분한 데이터도 있었다. 이번 분석에서는 현재 가지고 있는 텍스트에 대하여 감성을 점수로 부여하여 결과를 확인해보고자 한다. 따라서 기존에 만들어 놓았던 Everytime_df 데이터에 감성 사전을 left_join 하여 각 단어와 그에 따른 감성 점수를 부여하도록 한다. 그 전에 NA 개수를 확인해보니, 꽤 많은 데이터들에 대해 일치하는 단어들이 없어서 감성 점수가 들어가지 못했던 것도 확인할 수 있었다.

```{r}
# NA를 제거하여 감성 점수가 반영된 데이터의 시기 별 분포
sentiment_df[!is.na(sentiment_df$score), ] %>%
  filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  ggplot(aes(x = datetime)) +
  geom_histogram(bins = 200, color = 'black')
```



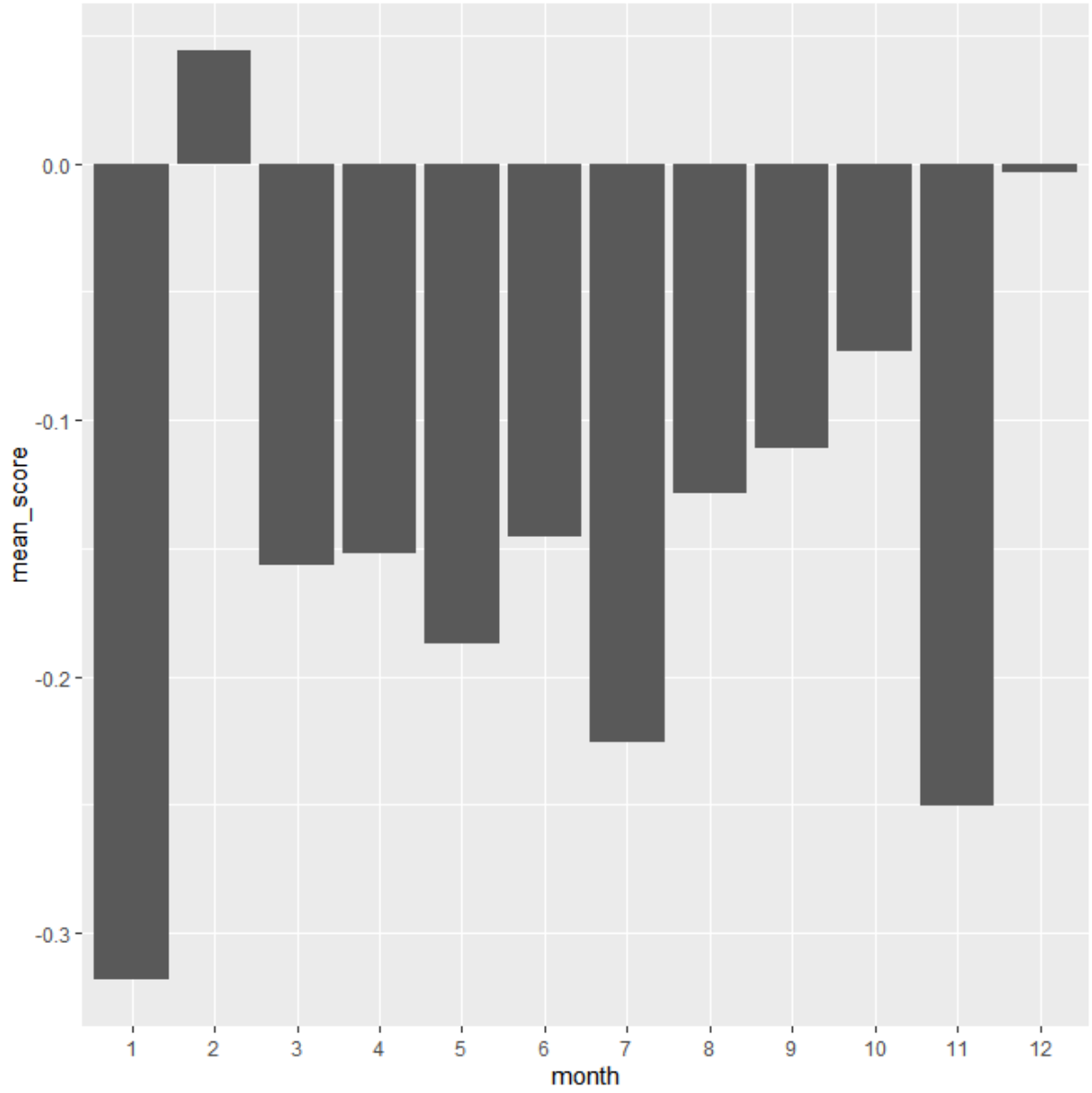
데이터가 충분히 있는 2019년과 2020년에 대해서 결과를 확인해보고자 한다. 감성사전과 우리 단어에 대해 join을 하려고 하다 보니, 단어가 완전히 일치하지 않으면 잘 묶이지 않는다는 한계가 있어서 NA의 개수가 많이 나온다. 그럼에도 불구하고, 현재 있는 약 4만개 데이터들의 분포를 확인해보면, 2019년에는 듬성듬성 데이터가 분포하는 것을 볼 수 있고, 2020년에는 전반적으로 감성점수가 매치된 단어들의 비율이 조금 더 높은 것을 확인할 수 있다.

```r
# 학기 별 감성분석
sentiment_df %>%
  dplyr::filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', '2019 winter vacation',
                        ifelse(date(datetime) < '2019-06-15', '2019 spring semester',
                          ifelse(date(datetime) < '2019-08-26', '2019 summer vacation',
                            ifelse(date(datetime) < '2019-12-14', '2019 fall semester',
                              ifelse(date(datetime) < '2020-02-24', '2020 winter vacation',
                                ifelse(date(datetime) < '2020-06-20', '2020 spring semester',
                                  ifelse(date(datetime) < '2020-08-31', '2020 summer vacation', '2020 fall semester')))))))) %>%
  group_by(label) %>%
  dplyr::summarise(mean_score = mean(score, na.rm = T)) %>%
  dplyr::arrange(desc(mean_score)) %>%
  ggplot(aes(x = factor(label), y = mean_score)) +
  geom_bar(stat = 'identity') +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5)) + labs(x = "")
```



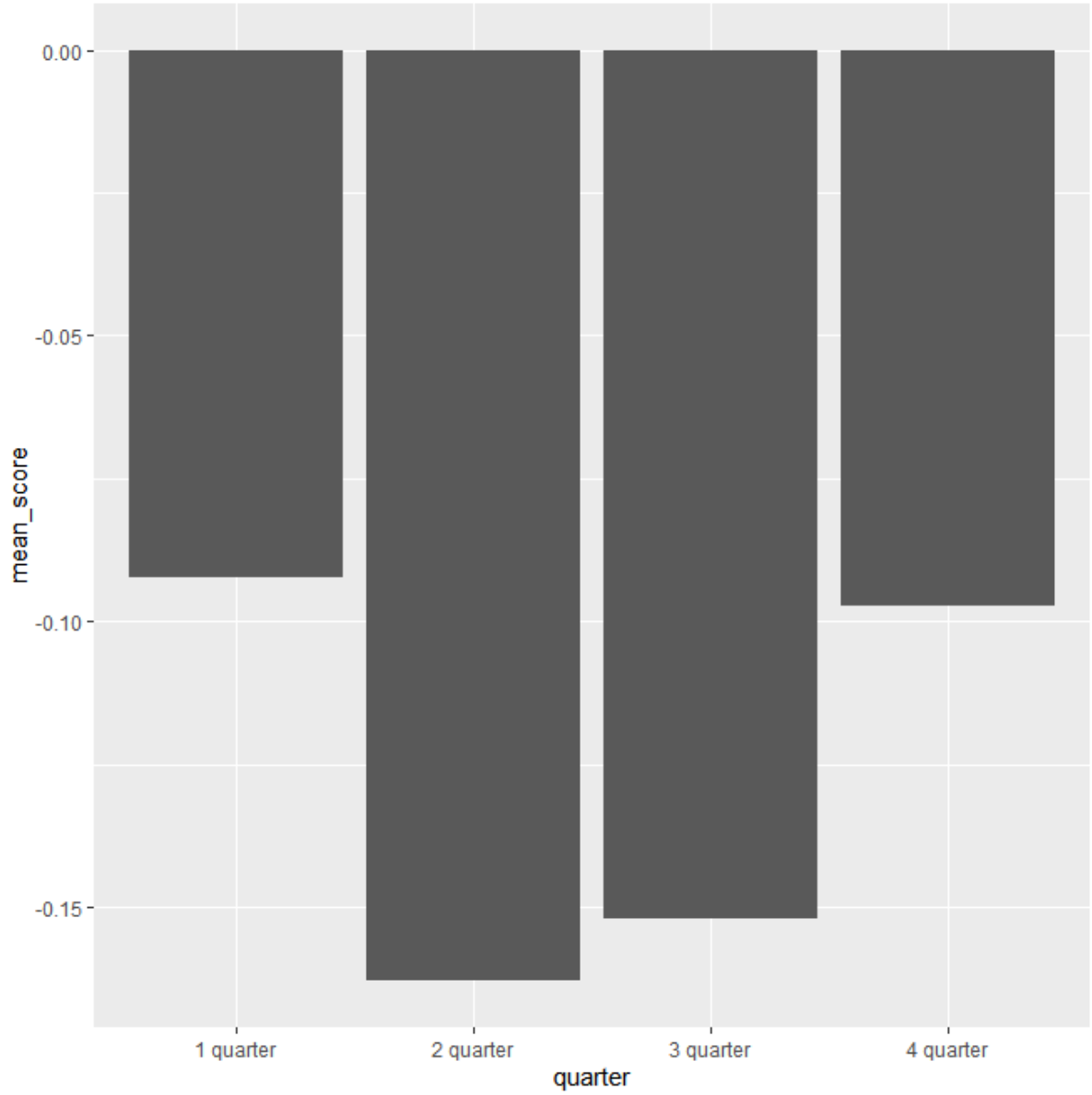
그리고 나서 가장 기초적으로는 학기 별로 감성 점수들의 평균을 계산해보고자 한다. 2019년과 2020년의 봄과 가을학기 그리고 여름과 겨울방학을 나누어 감성 점수의 평균을 시각화 해보았다. 전반적으로 2019년 봄과 가을학기에는 부정적인 감성이 높았던 것을 볼 수 있고, 여름과 겨울 방학은 감성점수가 소폭 증가하는 것을 확인할 수 있다. 반면, 2020년에는 학기와 방학 모두 낮은 감성점수를 확인할 수 있다.

```r
# 월 별 감성분석
sentiment_df %>%
  group_by(month(datetime)) %>%
  dplyr::summarise(mean_score = mean(score, na.rm = T)) %>%
  ggplot(aes(x = factor(`month(datetime)`), y = mean_score)) +
  geom_bar(stat = "identity") + labs(x = "month")
```



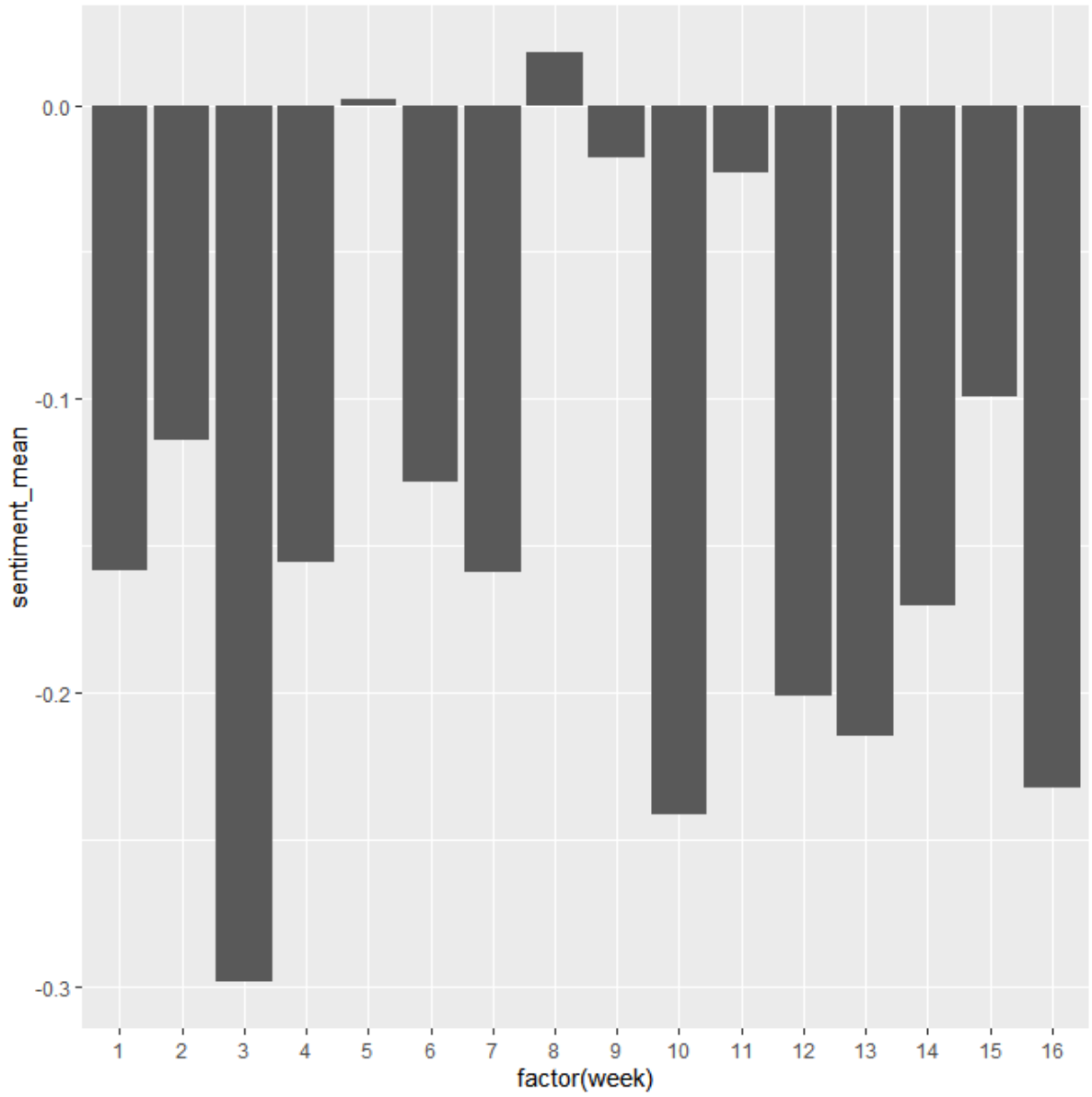
다음은 월 별로 감성분석을 수행해보도록 한다. 전반적으로 모든 월에 대해 부정적인 감성 점수를 보이고 있으며, 2월 달에는 새로운 학기가 시작하는 이유인지 약간 긍정적인 감성 점수가 나오는 것을 확인할 수 있다.

```{r}
# 분기 별 감성분석
sentiment_df %>%
  mutate(month = month(datetime),
         month_label = ifelse(month %in% c(1, 2, 3), '1 quarter',
                              ifelse(month %in% c(4, 5, 6), '2 quarter',
                                     ifelse(month %in% c(7, 8, 9), '3 quarter', '4 quarter')))) %>%
  group_by(month_label) %>%
  dplyr::summarise(mean_score = mean(score, na.rm = T)) %>%
  ggplot(aes(x = factor(month_label), y = mean_score)) +
  geom_bar(stat = "identity") + labs(x = "quarter")
```



또한 1, 2, 3월 & 4, 5, 6월 & 7, 8, 9월 & 10, 11, 12월로 1, 2, 3, 4분기로 나누어서 감성 점수를 확인해본다. 전반적으로 모든 분기 안에서 부정적인 감성이 대부분이라는 것을 확인해볼 수 있다.

```r
```{r}
# 2020년 1학기 주차별 평균 감성분석
sentiment_df %>%
  dplyr::filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', '2019 winter vacation',
                               ifelse(date(datetime) < '2019-06-15', '2019 spring semester',
                                      ifelse(date(datetime) < '2019-08-26', '2019 summer vacation',
                                             ifelse(date(datetime) < '2019-12-14', '2019 fall semester',
                                                    ifelse(date(datetime) < '2020-02-24', '2020 winter vacation',
                                                           ifelse(date(datetime) < '2020-06-20', '2020 spring semester',
                                                                  ifelse(date(datetime) < '2020-08-31', '2020 summer vacation', '2020
fall semester'))))))))) %>%
  filter(label == '2020 spring semester') %>%
  mutate(diff = difftime(datetime, as.Date('2020-03-01'), units = 'days'),
         week = ceiling(diff / 7)) %>%
  filter(week >= 1) %>%
  group_by(week) %>%
  summarise(sentiment_mean = mean(score, na.rm = T)) %>%
  ggplot(aes(x = factor(week), y = sentiment_mean))+
  geom_col()
```
```



2020년 1학기에 대해서 1주차부터 16주차까지 주차를 나누어 감성분석을 시행해보도록 한다. 직관적으로 알 수 있듯, 대부분의 주차에서 부정적인 감성 단어들이 많이 나오는 것을 확인할 수 있다. 다만, 중간고사가 있는 8주차에서 의외로 감성점수가 높게 나오는 것을 확인했는데, 이를 다시 보기 위해 추가적으로 분석을 수행해보도록 한다.

```r
# 8주차에서 긍정이 많이 나온 이유를 워드클라우드로 확인
df <- sentiment_df %>%
  dplyr::filter((year(datetime) == 2019) | (year(datetime) == 2020)) %>%
  dplyr::mutate(label = ifelse(date(datetime) < '2019-02-25', '2019 winter vacation',
                               ifelse(date(datetime) < '2019-06-15', '2019 spring semester',
                                      ifelse(date(datetime) < '2019-08-26', '2019 summer vacation',
                                             ifelse(date(datetime) < '2019-12-14', '2019 fall semester',
                                                    ifelse(date(datetime) < '2020-02-24', '2020 winter vacation',
                                                           ifelse(date(datetime) < '2020-06-20', '2020 spring semester',
                                                                  ifelse(date(datetime) < '2020-08-31', '2020 summer vacation', '2020
fall semester')))))))) %>%
  filter(label == '2020 spring semester') %>%
  mutate(diff = difftime(datetime, as.Date('2020-03-01'), units = 'days'),
         week = ceiling(diff / 7),
         label = ifelse(score %in% c(-2, -1), 'negative', 'positive')) %>%
  filter(week >= 1) %>%
  filter(week == 8)

df %>%
  group_by(word) %>%
  count() %>%
  left_join(df[, c('word', 'label')], by = 'word') %>%
  acast(word ~ label, value.var = 'n', fill = 0) %>%
  comparison.cloud(colors = c('darkred', 'darkcyan'), max.words = 300)
```

8주차에서 긍정적인 감성 점수가 더 높았던 이유를 확인해보기 위해 긍정과 부정에 대한 각 감성 단어들을 워드클라우드로 만들어보았다. 시험기간이여서 힘들 것 같았던 8주차에 긍정적인 감성이 높은 이유를 확인해보고자 했는데, 아니다, 없다, 힘들다 등과 같은 부정적인 단어들도 생각보다 많이 나오고 있음을 직관적으로 확인할 수 있었다. 해당 결과로는 정확한 이유를 유추하는데 약간의 한계가 있는 것 같았다.