

Count words and weight them with TF-IDF in R

BA3 Text as Data · Session 3 · BA2 review track (R)

Due Oct. 9 at 09:15 in your class/project repository: `demos/03-descriptive-patterns/`. I am asking you to create one R script, two word clouds (counts and TF-IDF), two bar charts of the top ten words, and a README with a short reflection. It is the same task as the Orange demonstration, done in R. Follow the steps in order and add each code block to your script.

Before you start

This week starts from the speeches you preprocessed in Session 2. You do not have to run Kiwi again. Pick one of the three routes in Step 4. Route A is the easiest if your Session 2 script works.

Step 1 · Create the script

In your course RStudio Project, choose File > New File > R Script. Save it as `demos/03-descriptive-patterns/workflow/s03-bag-of-words.R`

```
# BA3 Text as Data · Session 3 · Counts and TF-IDF for the presidential speeches
```

Step 2 · Install and load the packages

`tidytext` is new this week. The line below installs only the packages you do not have yet.

```
packages <- c("readr", "dplyr", "tidyr", "tidytext", "ggplot2", "ggwordcloud", "ragg")
install.packages(setdiff(packages, rownames(installed.packages())))

library(readr)
library(dplyr)
library(tidyr)
library(tidytext)
library(ggplot2)
library(ggwordcloud)
```

Step 3 · Set a seed

Same as last week. The seed makes your word clouds come out the same every time you run the script.

```
set.seed(2026)
```

Step 4 · Load your preprocessed speeches

Each route ends with the same table, words, with one row per word and the speech it came from.

Route A · Save your Session 2 result (recommended)

1. Open your Session 2 script, `demos/02-preprocessing/workflow/s02-preprocessing.R`.
2. At the end of its Step 6, the step that creates `clean` (the table of filtered nouns), add this line:

```
write_csv(clean, "data/speeches-clean.csv")
```

3. Run the Session 2 script once from top to bottom. This takes a few minutes, because Kiwi runs again. A new file, `speeches-clean.csv`, appears in your `data/` folder.

Then, in this week's script:

```
words <- read_csv(
  "data/speeches-clean.csv",
  col_types = cols(.default = col_character(), division_number = col_integer())
) |>
  select(division_number, word)
```

Hints:

- You only do this once. If you later change your Session 2 settings, run that script again to update the file.
- Adding this line after the Demo 2 deadline does not affect your Demo 2 mark.
- Commit data/speeches-clean.csv with Demo 3, so your script runs from the repository.
- If R says the file does not exist, check that RStudio shows your course Project in the top right. The paths start from the Project folder.
- Open the file from the Files pane to see what you saved. It has one row per word, with its speech, president, speech type, and POS tag.

Route B • Run the preprocessing again

Copy Session 2's Steps 4, 5, and 6 from your Session 2 script into this script, below Step 3, under a comment such as # Route B: Session 2 Steps 4 to 6. Add "reticulate" to the packages in Step 2 and library(reticulate) to the library lines. Then:

```
words <- clean |>
  select(division_number, word)
```

Route C • If Kiwi will not run

Download `presidential-speeches-100-processed.csv` into data/. Its processed_text column holds the nouns the course script kept, separated by spaces. You also need stopwords-ko.txt from Session 2.

```
words <- read_csv(
  "data/presidential-speeches-100-processed.csv",
  col_types = cols(.default = col_character(), division_number = col_integer())
) |>
  select(division_number, processed_text) |>
  separate_longer_delim(processed_text, delim = " ") |>
  rename(word = processed_text) |>
  filter(word != "") |>
  filter(!word %in% read_lines("data/stopwords-ko.txt"))
```

Check the result

Whichever route you used, run this and write the route in your README. Do not skip it, because Step 10 uses n_speeches.

```
n_speeches <- n_distinct(words$division_number)

n_speeches          # 100
nrow(words)         # about 28,840 words (28,821 with route C)
n_distinct(words$word) # about 4,370 different words (4,364 with route C)
```

The slides used the course file, as in route C. With routes A and B your numbers differ a little from the slides, because your R preprocessing and the course script differ slightly. That is expected.

Step 5 • The bag of words

Count each word in each speech. Word order is gone. What is left is which words each speech uses and how often. Look at the Moon Jae-in speech we used in class, his first-anniversary message. Its speech number is 1400600.

```
bow <- words |>
  count(division_number, word, name = "n")

bow |>
  filter(division_number == 1400600) |>
  arrange(desc(n))      # 국민 9, 대한민국 3, 세상 3, ...
```

Step 6 • The document-term matrix

The same counts as a table with one row per speech and one column per word. Most cells are zero.

```
dtm <- cast_sparse(bow, division_number, word, n)
dim(dtm)                # 100 speeches x about 4,370 words
mean(as.matrix(dtm) == 0) # about 0.96 of the cells are zero
```

Step 7 • Term frequency and document frequency

count is how often a word occurs in all speeches. df is the number of speeches it occurs in.

```
word_totals <- bow |>
  group_by(word) |>
  summarise(count = sum(n), df = n_distinct(division_number)) |>
  arrange(desc(count))

head(word_totals, 10) # 국민 443 in 83 speeches, then 경제, 세계, 정부, ...
```

Step 8 • TF-IDF

bind_tf_idf() adds the IDF, the natural log of 100 divided by the word's DF. We use count × IDF, as Orange's Bag of Words does with Count and IDF, and ignore the tf_idf column that bind_tf_idf() also adds.

```
tfidf <- bow |>
  bind_tf_idf(word, division_number, n) |>
  mutate(tfidf = n * idf)

tfidf |>
  filter(division_number == 1400600) |>
  arrange(desc(tfidf)) |>
  select(word, n, idf, tfidf) # 세상 and 정의 near the top, 국민 far down at 1.68
```

Compare this list with Step 5. 국민 is the most frequent word in the speech, but it is in 83 of the 100 speeches, so TF-IDF pushes it down.

Step 9 • Word cloud of the counts

```
korean_font <- switch(Sys.info()[["sysname"]],
  Windows = "Malgun Gothic", Darwin = "AppleGothic", "Noto Sans CJK KR")

dir.create("demos/03-descriptive-patterns/output", recursive = TRUE, showWarnings =
  ↪ FALSE)

top_counts <- word_totals |>
```

```

slice_head(n = 50)

cloud_counts <- ggplot(top_counts, aes(label = word, size = count)) +
  geom_text_wordcloud(family = korean_font, eccentricity = 1) +
  scale_size_area(max_size = 14) +
  labs(title = "Counts") +
  theme_minimal(base_family = korean_font)

ggsave("demos/03-descriptive-patterns/output/counts.png", cloud_counts,
  device = ragg::agg_png, width = 8, height = 5, dpi = 200, bg = "white")

```

Step 10 • Word cloud of TF-IDF

Each word needs one TF-IDF number for the whole corpus. As in Orange, we take the mean per speech: add up the word's TF-IDF over all speeches and divide by 100. Then keep the top 50 and draw the same kind of cloud.

```

tfidf_totals <- tfidf |>
  group_by(word) |>
  summarise(tfidf = sum(tfidf) / n_speeches) |> # mean per speech
  arrange(desc(tfidf))

head(tfidf_totals, 10) # 경찰 first (2.76 on the slides), then 양국, ...

top_tfidf <- tfidf_totals |>
  slice_head(n = 50)

cloud_tfidf <- ggplot(top_tfidf, aes(label = word, size = tfidf)) +
  geom_text_wordcloud(family = korean_font, eccentricity = 1) +
  scale_size_area(max_size = 14) +
  labs(title = "TF-IDF") +
  theme_minimal(base_family = korean_font)

ggsave("demos/03-descriptive-patterns/output/tfidf.png", cloud_tfidf,
  device = ragg::agg_png, width = 8, height = 5, dpi = 200, bg = "white")

```

Step 11 • Bar charts of the top ten words

A word cloud is hard to read precisely. A bar chart shows the order and the size of each value. Make one for the counts and one for TF-IDF.

```

bar_counts <- word_totals |>
  slice_head(n = 10) |>
  ggplot(aes(x = count, y = reorder(word, count))) +
  geom_col(fill = "#2C4A9E") +
  labs(title = "Top ten words by count", x = "Count", y = NULL) +
  theme_minimal(base_family = korean_font)

ggsave("demos/03-descriptive-patterns/output/counts-top10.png", bar_counts,
  device = ragg::agg_png, width = 7, height = 4.5, dpi = 200, bg = "white")

bar_tfidf <- tfidf_totals |>
  slice_head(n = 10) |>
  ggplot(aes(x = tfidf, y = reorder(word, tfidf))) +
  geom_col(fill = "#F46E32") +
  labs(title = "Top ten words by TF-IDF", x = "Mean TF-IDF per speech", y = NULL) +
  theme_minimal(base_family = korean_font)

ggsave("demos/03-descriptive-patterns/output/tfidf-top10.png", bar_tfidf,
  device = ragg::agg_png, width = 7, height = 4.5, dpi = 200, bg = "white")

```

`reorder()` sorts the bars so the largest value is at the top.

Step 12 • Which words rise and fall?

This table puts each word's rank by count next to its rank by TF-IDF.

```
rank_change <- full_join(
  word_totals |> mutate(count_rank = row_number()) |> select(word, count_rank),
  tfidf_totals |> mutate(tfidf_rank = row_number()) |> select(word, tfidf_rank),
  by = "word"
)

rank_change |> filter(count_rank <= 10) # 국민 falls from 1 to 72
rank_change |> filter(tfidf_rank <= 10) # 경찰 rises from about 27 to 1
```

Pick a word that rises and check which speeches put it there. For example:

```
bow |>
  filter(word == "경찰") |>
  arrange(desc(n))
```

Step 13 • Record your R setup

```
writelines(capture.output(sessionInfo()),
           "demos/03-descriptive-patterns/output/session-info.txt")
```

Save the script and run it again from top to bottom (Ctrl+Shift+Enter or Cmd+Shift+Enter) to see if it works.

Step 14 • README and reflection

In `demos/03-descriptive-patterns/README.md`, name the corpus, say which route you used in Step 4, and write down your settings. They are your Session 2 preprocessing settings (or the course file, for route C) and TF-IDF as $\text{count} \times \text{IDF}$, averaged per speech.

Then write a short reflection, one or two paragraphs. Use your two bar charts and Step 12.

- What do the counts tell you about the speeches?
- What does TF-IDF tell you that the counts do not?
- Name at least three words that rise or fall, and check in which speeches they occur.
- What did you learn about counting and weighting words?

Submit

The .R script, the four PNG files in `output/` (`counts.png`, `tfidf.png`, `counts-top10.png`, `tfidf-top10.png`), `output/session-info.txt`, and the README with your reflection. With route A, also commit `data/speeches-clean.csv`. Commit and push to the GitHub repository.

If Korean shows as boxes in the Plots pane, the saved PNG files are still correct. To fix the pane, choose Tools, Global Options, General, Graphics, Backend AGG.