

Cluster the speeches in R

BA3 Text as Data, Session 4, BA2 review track (R)

Due Oct. 16 at 09:15, in `demos/04-clustering/` in your class/project repository. I am asking you to create one R script, a dendrogram, two more figures or tables about the groups, and a README that describes and names the groups. 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 you preprocess the 300 speeches yourself, with your Session 2 script. Steps 3 and 4 repeat what you did in Sessions 2 and 3. From Step 5 on, you compare speeches with cosine similarity, group them with hierarchical clustering, draw the tree, and describe the groups.

Step 1. Create the script

In your course RStudio Project, choose File > New File > R Script. Save it as `demos/04-clustering/workflow/s04-clustering.R`

```
# BA3 Text as Data, Session 4, Hierarchical clustering of the presidential speeches
```

Step 2. Install and load the packages

No new packages this week. These are the Session 3 packages. The seed makes word clouds and the optional k-means come out the same every time.

```
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)

set.seed(2026)
```

Step 3. Preprocess the 300 speeches

Download `presidential-speeches-300.csv` into `data/`. It has the same columns as the 100-speech file.

1. Open your Session 2 script, `demos/02-preprocessing/workflow/s02-preprocessing.R`, and save a copy as `demos/04-clustering/workflow/s04-preprocessing.R`.
2. In the copy, change the file name `presidential-speeches-100.csv` to `presidential-speeches-300.csv`.
3. At the end of the step that creates `clean` (the table of filtered nouns), add this line, as in Session 3 route A.

```
write_csv(clean, "data/speeches-300-clean.csv")
```

4. Run the copy once from top to bottom. Kiwi now reads 300 speeches, so this takes several minutes.

Then, in this week's script:

```
speeches <- read_csv(
  "data/presidential-speeches-300.csv",
  col_types = cols(.default = col_character(), division_number = col_integer())
)

words <- read_csv(
  "data/speeches-300-clean.csv",
  col_types = cols(.default = col_character(), division_number = col_integer())
) |>
  select(division_number, word)

n_distinct(words$division_number) # 300
nrow(words) # about 86,000 words
```

Keep speeches. You need it to label and read the groups.

Step 4. TF-IDF as a matrix

As in Session 3, count each word in each speech and weight it with $\text{count} \times \text{IDF}$. Then cast the result into a document-term matrix, one row per speech and one column per word, and turn it into a regular R matrix.

```
tfidf <- words |>
  count(division_number, word, name = "n") |>
  bind_tf_idf(word, division_number, n) |>
  mutate(tfidf = n * idf)

m <- tfidf |>
  cast_sparse(division_number, word, tfidf) |>
  as.matrix()

dim(m) # 300 speeches x about 7,000 words
```

Step 5. Cosine distance

Cosine similarity does not depend on the length of the speeches, only on which words they use and in what proportions. 1 means the same words in the same proportions, 0 means no words in common. First divide each row by its length, so that every speech has length 1. In Orange, this is Regularization L2. Cosine similarity is then a matrix product, each row multiplied with each other row. Clustering works with distances, so convert similarity into distance by subtracting it from 1. `as.dist()` keeps one triangle of the table.

```
m_l2 <- m / sqrt(rowSums(m^2))
sim <- m_l2 %*% t(m_l2) # 300 x 300
d <- as.dist(1 - sim)
```

Check one pair. Lee Myung-bak's 2008 Police Day speech and Park Geun-hye's 2015 Police Day speech share many words.

```
sim["1330009", "1400368"] # about 0.77
```

Step 6. Hierarchical clustering

Cluster with Ward linkage, as in Orange. Orange's Ward is "ward.D2" in R. Then choose the number of groups, 3, 4, or 5, and write it in k.

```
hc <- hclust(d, method = "ward.D2")

k <- 5 # choose 3, 4, or 5
```

```
groups <- cutree(hc, k = k)
groups <- match(groups, unique(groups[hc$order]))
table(groups) # the slides' file gives, for k = 5, 12, 60, 35, 6, 187
```

cutree() gives each speech a group number. The match() line renumbers the groups in the order they appear in the tree, so group 1 in your tables is group 1 in the dendrogram. With the course file, k = 3 gives 12, 60, and 228, and k = 4 gives 12, 60, 35, and 193. Your sizes can differ a little, because your preprocessing differs slightly from the course script. That is expected.

Now put the group next to the information about each speech. info has the speeches in the same order as the rows of m.

```
info <- tibble(division_number = as.integer(rownames(m))) |>
  left_join(speeches, by = "division_number") |>
  mutate(group = groups)
```

Step 7. The dendrogram

A tree with 300 leaves cannot be read. In Orange, students prune the tree. In R, cut the tree so that each group becomes one leaf, and label each leaf with a column of your choice. Each label shows the group, its size, and the three most common values of that column. president, kind, and location work best. Title and date work too, but almost every speech has its own, so they say less about a group.

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

dir.create("demos/04-clustering/output", recursive = TRUE, showWarnings = FALSE)

label_column <- "president" # or "kind", "location", "title", "date"

group_labels <- sapply(1:k, function(g) {
  values <- head(sort(table(info[[label_column]][groups == g]), decreasing = TRUE), 3)
  paste0("Group ", g, " (", sum(groups == g), "): ",
    paste(names(values), values, collapse = ", "))
})
group_labels

cut_height <- mean(rev(hc$height)[c(k - 1, k)])
top <- as.hclust(cut(as.dendrogram(hc), h = cut_height)$upper)
top$labels <- group_labels

ragg::agg_png("demos/04-clustering/output/clusters.png",
  width = 10, height = 1.5 + 0.5 * k, units = "in", res = 200)
par(family = korean_font, mar = c(4, 1, 3, 26))
plot(as.dendrogram(top), horiz = TRUE, xlab = "Distance",
  main = paste("Ward clustering, cosine distance,", k, "groups"))
dev.off()
```

hc\$height holds the height of every merge. cut_height lies between the highest k - 1 merges and the rest, so cut() leaves exactly k branches. as.hclust() turns those branches into a small tree with k leaves. Group 1 is at the bottom. Open the PNG from the Files pane. If the labels are cut off on the right, make width larger.

Step 8. What is in each group

In the README you give each group a name. Base it on two things, the top words of the group and two or three speeches you read from it. For the top words, take the mean TF-IDF per speech as in Session 3, now within each group.

```

group_sizes <- info |>
  count(group, name = "size")

group_words <- tfidf |>
  left_join(select(info, division_number, group), by = "division_number") |>
  group_by(group, word) |>
  summarise(tfidf = sum(tfidf), .groups = "drop") |>
  left_join(group_sizes, by = "group") |>
  mutate(tfidf = tfidf / size) # mean per speech in the group

group_words |>
  group_by(group) |>
  slice_max(tfidf, n = 8, with_ties = FALSE) |>
  summarise(words = paste(word, collapse = " ")) # group 1: 여성 가정 지도자 ...

```

Then list the speeches in a group and read the start of one of them. Change the group number and the division_number to look at other groups and speeches.

```

info |>
  filter(group == 1) |>
  select(division_number, president, title, date)

speeches |>
  filter(division_number == 1400117) |>
  pull(speech_text) |>
  substr(1, 800) |>
  strwrap(width = 60) |>
  cat(sep = "\n")

```

Step 9. Further analysis

Choose at least two of the three below. Each one saves a file in output/ that you show in the README.

Option A. Groups by president

The same as Orange's Pivot Table. Count the speeches of each president in each group. For speech types, replace president with kind in both places.

```

groups_by_president <- info |>
  count(group, president) |>
  pivot_wider(names_from = president, values_from = n, values_fill = 0)

groups_by_president
write_csv(groups_by_president, "demos/04-clustering/output/groups-by-president.csv")

```

Option B. Group sizes by president

The same as Orange's Distributions. One bar per group, coloured by president. For speech types, replace president with kind.

```

sizes_plot <- ggplot(info, aes(y = factor(group), fill = president)) +
  geom_bar() +
  labs(title = "Speeches per group", x = "Speeches", y = "Group", fill = NULL) +
  theme_minimal(base_family = korean_font)

ggsave("demos/04-clustering/output/group-sizes.png", sizes_plot,
  device = ragg::agg_png, width = 7, height = 4, dpi = 200, bg = "white")

```

Option C. Top words of one group

The same as Orange's Word Cloud, drawn as a bar chart as in Session 3 Step 11. Change `group == 1` to the group you want, in the filter and in the file name. For a word cloud instead, use the code from Session 3 Step 10 on the top 50 words of the group.

```
words_plot <- group_words |>
  filter(group == 1) |>
  slice_max(tfidf, n = 10, with_ties = FALSE) |>
  ggplot(aes(x = tfidf, y = reorder(word, tfidf))) +
  geom_col(fill = "#F46E32") +
  labs(title = "Top ten words in group 1", x = "Mean TF-IDF per speech", y = NULL) +
  theme_minimal(base_family = korean_font)

ggsave("demos/04-clustering/output/group-1-words.png", words_plot,
  device = ragg::agg_png, width = 7, height = 4.5, dpi = 200, bg = "white")
```

Step 10. k-means (optional)

Not required. k-means asks for the number of groups first and then moves speeches between groups until each speech is closest to its group's centre. Use the L2-normalised rows from Step 5, or most speeches end up in one large group. k-means starts from random points, so set the seed right before it.

```
set.seed(2026)
km <- kmeans(m_l2, centers = k, nstart = 10)
km$size
table(hierarchical = groups, kmeans = km$cluster)
```

The table compares the two clusterings. Their group numbers do not match, so look for cells that hold most of a row. If you try k-means, add a short note to the README.

Step 11. Record your R setup

```
writelines(capture.output(sessionInfo()),
  "demos/04-clustering/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 12. README

In `demos/04-clustering/README.md`, write the following.

- The corpus, your preprocessing settings from Session 2, and your clustering settings (TF-IDF as count × IDF, cosine distance, Ward linkage, the number of groups, and the label column).
- `output/clusters.png` and your two other outputs, each with a sentence on what it shows. Show a PNG with `![Dendrogram](output/clusters.png)`. Show a table from Option A as a Markdown table, as below.
- A short name for each group, with one sentence on what it contains, based on its top words and on the two or three speeches you read from it.
- If you did Step 10, a short note on k-means.

Group	문재인	박근혜	이명박	
-----	-----	-----	-----	
1	1	10	1	

Before you submit

- The README names the settings (TF-IDF, cosine distance, Ward linkage, the number of groups, and the label).
- `output/clusters.png` shows the dendrogram with 3 to 5 groups.
- Two more figures or tables about the groups are in `output/` and shown in the README.
- The README says what each figure shows and names each group.
- The workflow or script is in `workflow/`.
- The final files are pushed to GitHub.

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.