Collecting, classifying, and analyzing textual data using R

Wednesday, June 5th, 2024
3rd Summer Training on Qualitative and Quantitative Methods
Universitas Islam Internasional Indonesia

Aichiro Suryo Prabowo
aichiro@cornell.edu

Agenda (09:30 – 16:15)

09.30-10.45	**: Session 1, introduction**
10.45-11.00	: Break (15 minutes)
11.00-12.15	**: Session 2, importing text data**
12.15-13.15	: Break (60 minutes)
13.15-14.45	**: Session 3, analyzing text data**
14.45-15.00	: Break (15 minutes)
15.00-15.30	**: Session 4, wrap-up**
15.30-16.15	**: Session 4 (optional) individual discussions**

Session 1

Introduction and setup

Why text-as-data?

- Information-rich, but underexplored
- Textual analysis is not entirely new
- Going through texts is time-consuming. That's where the computer comes in handy.
- R is a computer language. The application is called RStudio, which is free and easy to use. Note that there are other languages and applications. Click here to read the instructions for downloading RStudio.

Questions that can be answered

- What terms (or n-grams) are mentioned most frequently?
- What are the key themes within a collection of documents?
- What is the emotional tone (i.e., sentiment) behind a body of text?

Question that can *not* be answered

- Is there correlation?
- Is there causality?

- You might want to combine textual analysis with other methods, such as categorical data analysis, panel data analysis, survey and sampling methods, etc.
- Please take a look at other sessions in STQ2M.

Caveats

- You'll learn some useful techniques, but they're not exhaustive.

- Computational text analysis is a growing method. It is not perfect. Some scholars remain skeptical about its reliability.

- The goal of today's session is *not* to train you to become an expert. Instead, it is to draw you into this area, hoping that you'll then become part of the community that will apply, refine, and extend the methodology.

RStudio basics

- We'll be coding on an "R Script"
- Before we're able to run a code, install the R package

  install.packages("tidyverse")
- Once installed, load the R package to have it activated

  library(tidyverse)
- For help

  ?tidyverse

A few more details

Clearing R environment to start

```
rm(list = ls())                    #rm stands for remove
```

Specifying a working directory on your computer

```
setwd("Documents/…")               #change to your folder
```

Assigning left-arrow operator

`X` <- `Y` means assign the value (or operation) on the right-hand side (Y) to the object on the left-hand side (X)

library(tidyverse) includes:

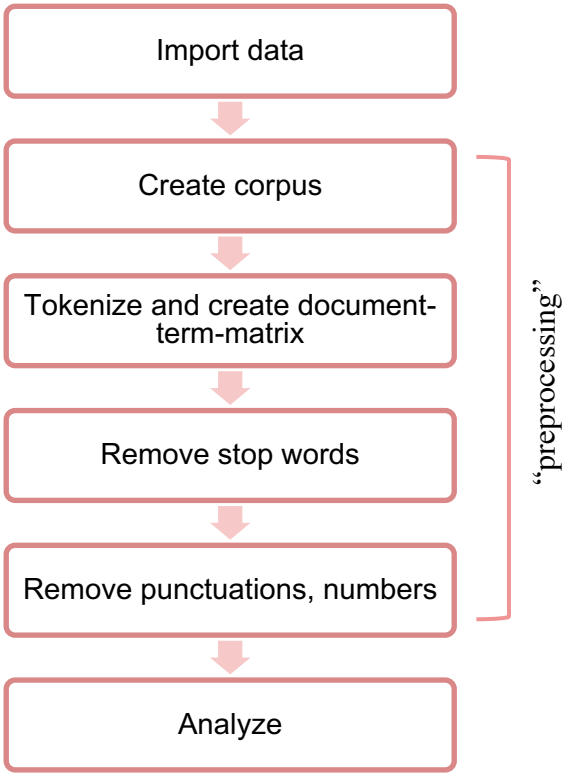
readr : to import CSV files
haven : to import STATA files
dplyr : to manipulate data
tidyr : to tidy data
ggplot2 : to visualize data

There are more packages in R.

Session 2

Importing text data

Steps in computational text analysis



What's the objective of this exercise?

1. Importing raw text data to R
2. Transforming raw text data into a corpus
3. Inspecting basic statistics of the corpus

Where could we find *text* data?

Anywhere, including:

- Files (PDF, CSV, etc.)
- Website
- Social media: twitter, etc.
- Archival documents, usually these are manually-scanned

Importing text data from CSV files

```
install.packages("tidyverse")   #if you haven't, install the package
library(tidyverse)              #load the package
read.csv("pidato.csv")          #read the csv file
```

Importing text data from a website

```
install.packages("tidyverse")                                  #if you haven't, install the package
install.packages("rvest")                                      #if you haven't, install the package
library(tidyverse)  #load the package                          #load the package
library(rvest)                                                 #load the package

url2023 <- "https://bit.ly/3RdKpxk"                            #enter the target webpage as data source
webpage2023 <- read_html(url2023)                              #read the HTML content

nodes2023 <- html_nodes(webpage2023, xpath = "//p") #select nodes (e.g., all paragraphs <p> nodes)
p_nodes2023 <- html_text(nodes2023)                            #extract text content from selected nodes

combined_2023 <- paste(p_nodes2023, collapse = " ")  #combine all paragraphs into one document

pidato2023 <- data.frame(year = c(2023),
                         president = c("joko widodo"))         #create a new dataframe
pidato2023$speechtext <- c(combined_2023)                      #add a new variable for the extracted text
view(pidato2023)                                               #review results
```

Your turn!

Extract the text of the presidential speech from 2022 on this webpage:

https://bit.ly/4aKkhkt

Then, combine the 2023 and 2022 data frames into one data frame. There are multiple ways to do this, and one of them is:

`rbind(dataframe1, dataframe2)`

Your turn!

```
url2022 <- "https://bit.ly/4aKkhkt"                    #enter the target webpage as data source
webpage2022 <- read_html(url2022)                      #read the HTML content

nodes2022 <- html_nodes(webpage2022, xpath = "//p") #select nodes (e.g., all paragraphs <p> nodes)
p_nodes2022 <- html_text(nodes2022)                    #extract text content from selected nodes

combined_2022 <- paste(p_nodes2022, collapse = " ")  #combine all paragraphs into one document

pidato2022 <- data.frame(year = c(2022),
                         president = c("joko widodo"))  #create a new dataframe
pidato2022$speechtext <- c(combined_2022)              #add a new variable for the extracted text
view(pidato2022)                                       #review results

pidato2 <- rbind(pidato2022, pidato2023)               #combine the two dataframes
view(pidato2)                                          #review results
```

Alternative functions for web scraping

#Option #1, which was done previously

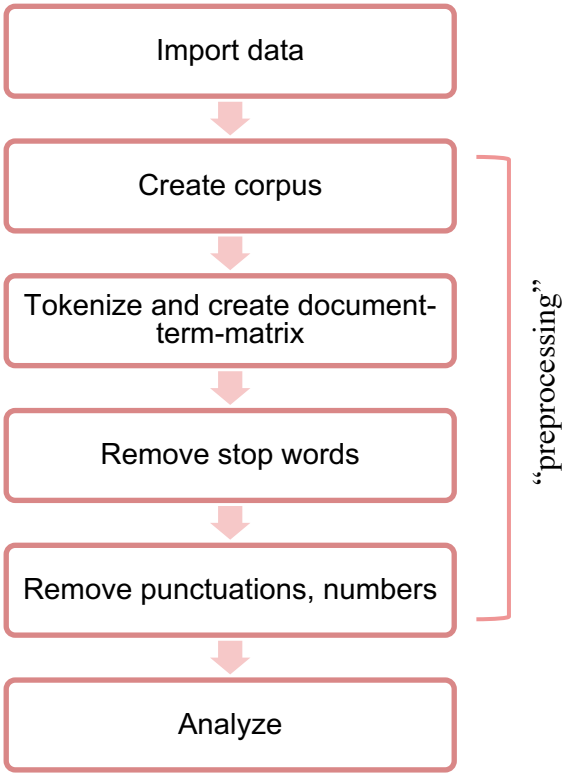
```
url2023 <- "https://bit.ly/3RdKpxk"                    #enter the target webpage as data source
webpage2023 <- read_html(url2023)                      #read the HTML content

nodes2023 <- html_nodes(webpage2023, xpath = "//p") #select nodes (e.g., all paragraphs <p> nodes)
p_nodes2023 <- html_text(nodes2023)                    #extract text content from selected nodes
```

#Option #2, incorporating R pipes

```
url2023 <- "https://bit.ly/3RdKpxk"                    #enter the target webpage as data source
p_nodes2023 <- webpage2023 %>%
  html_nodes(xpath = "//p") %>%
  html_text()                                          #combine all the remaining steps
```

#Options #1 and #2 should generate *identical* results!

Steps in computational text analysis



Creating corpus based on imported data

#from this point onward, we'll be working with "pidato.csv"

```
speechanalysis <- read.csv("pidato.csv")          #import the csv file
view(speechanalysis)                              #inspect the imported data

install.packages("quanteda")                      #if you haven't, install the package
library(quanteda)                                 #load the package

pidato_corpus <- corpus(speechanalysis, text_field = 'speechtext')  #create the corpus
pidato_statistics <- summary(pidato_corpus)       #get the summary statistics
view(pidato_statistics)                           #view results
write.csv(pidato_statistics,"pidato_statistics.csv")   #save results as a csv file
```

#How many types (unique terms), tokens (terms), and sentences do you see in each presidential speech?

Session 3

Analyzing text data

What's the objective of this exercise?

1. Constructing document-term-matrix to identify the most frequent terms in a given document
2. Analyzing the context of a keyword based on surrounding words

Tokenization & document-term matrix

Tokenizing text data breaks down large bodies of text into individual terms.

DTM represent the frequency that a term appears in a set of documents. It is structured as follows:

Rows = documents

Columns = terms

Cells = ?

Tokenization & document-term matrix

```
pidato_token <- tokens(pidato_corpus)          #tokenize words in the corpus
pidato_token
pidato_dtm <- dfm(pidato_token)                #construct DTM

topfeatures(pidato_dtm, 5)                     #identify the 5 most frequent terms
```

Your turn!

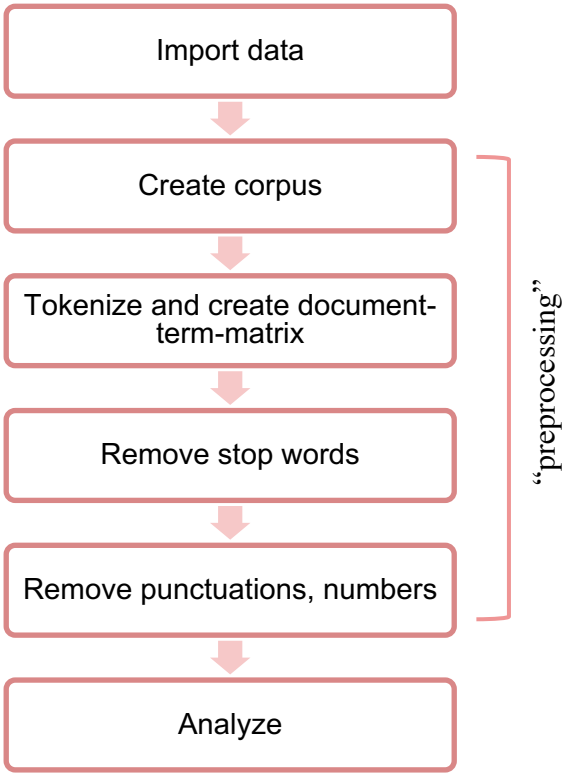
Using the same tokens and DTM, identify the 5 most frequent terms each year. Then expand your result to identify the 20 most frequent terms. Discuss what you've found.

```
pidato_token <- tokens(pidato_corpus)        #tokenize words in the corpus
pidato_token
pidato_dtm <- dfm(pidato_token)              #construct DTM

topfeatures(pidato_dtm, 5)                   #identify the 5 most frequent terms
topfeatures(pidato_dtm, 5, groups=year)      #identify the 5 most frequent terms each year

topfeatures(pidato_dtm, 20)                  #identify the 20 most frequent terms
topfeatures(pidato_dtm, 20, groups=year)     #identify the 20 most frequent terms each year
```

Removing stopwords, punctuations, numbers

```
library(stopwords)                                    #load the package
head(stopwords::stopwords("id", source = "stopwords-iso"), 20)     #show the list of stopwords
pidato_token2 <- tokens_remove(tokens(pidato_corpus),
stopwords("id", source = "stopwords-iso"))            #remove stopwords (indonesian)

pidato_dtm2 <- dfm(pidato_token2)                     #create dtm after stopwords removed
topfeatures(pidato_dtm2, 5)                           #identify the 5 most frequent terms
topfeatures(pidato_dtm2, 5, groups=year)              #identify the 5 most frequent terms each year
#what looks different compared to the previous result?

pidato_token2 <- tokens(pidato_token2, what = "word",
 remove_punct = TRUE, remove_numbers = TRUE,)         #remove punctuations and numbers

pidato_dtm2 <- dfm(pidato_token2)                     #create dtm after punctuations removed
topfeatures(pidato_dtm2, 5)                           #identify the 5 most frequent terms
topfeatures(pidato_dtm2, 5, groups=year)              #identify the 5 most frequent terms each year
#what looks different compared to the previous result?
```

Steps in computational text analysis



Wordclouds

```
library(quanteda.textplots)
textplot_wordcloud(pidato_dtm2, max_words = 20)     #top 20 (most frequent) words
textplot_wordcloud(pidato_dtm2, max_words = 20,
color = c('brown','red'))                           #change colors
```

Your turn!

Create a word cloud based on the following text data:

Presidential speech on 16 August 2009
Presidential speech on 16 August 2010
Presidential speech on 16 August 2011

Compare with the speeches we've previously analyzed.

Keyword-in-context (KWIC)

#A keyword-in-context shows words in the vicinity of a given keyword, allowing us to infer the context. This is a helpful feature that complements word cloud analysis.

```
#kwic = economy
pidato_context_econ <- kwic(tokens(pidato_corpus), 'ekonomi', window = 5)
pidato_context_econ

#kwic = environment
pidato_context_envi <- kwic(tokens(pidato_corpus), 'hutan', window = 5)
pidato_context_envi
```

Your turn!

Perform KWIC analysis based on 3 keywords of your choosing.

Session 4

Wrap-up