STAT 231 Blog Project Wrangling

Ethan Van De Water, Maximo Gonzalez, Alex Nichols

2023-11-27

```
##############################.txt files############################################

#series of articles relating to finance and the israel gaza conflict. Interesting to see w

file_path <- "data/JamieDimonCNBCInterview.txt"

# Read the text file into a data frame with one column named "Paragraph"
JamieDimonCNBCInterview1 <- read_lines(file_path) %>%
  tibble(Paragraph = .)

# Separate each paragraph into words and unnest the data frame
JamieDimonCNBCInterview <- JamieDimonCNBCInterview1 %>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)

file_path2 <- "data/JpowellSemiAnnualAddress.txt"

JPowellSemiAnnualAddress1 <- read_lines(file_path2) %>%
  tibble(Paragraph = .)

JPowellSemiAnnualAddress <- JPowellSemiAnnualAddress1 %>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)

file_path3 <- "data/MichaelBarrTreasuryMarket.txt"

MichaelBarrTreasury1 <- read_lines(file_path3) %>%
  tibble(Paragraph = .)
```

```r
MichaelBarrTreasury <- MichaelBarrTreasury1%>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)

file_path4 <- "data/BloombergWarArticle.txt"

BloombergArticle1 <- read_lines(file_path4) %>%
  tibble(Paragraph = .)

BloombergArticle <- BloombergArticle1%>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)

file_path5 <- "data/BrookingsArticle.txt"

BrookingsArticle1 <- read_lines(file_path5) %>%
  tibble(Paragraph = .)

BrookingsArticle <- BrookingsArticle1%>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)

file_path6 <- "data/DODArticle.txt"

DODArticle1 <- read_lines(file_path6) %>%
  tibble(Paragraph = .)

DODArticle <- DODArticle1%>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)

file_path7 <- "data/CnnArticle.txt"

CnnArticle1 <- read_lines(file_path7) %>%
  tibble(Paragraph = .)

CnnArticle <- CnnArticle1%>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)

file_path8 <- "data/JPMarticle2.txt"

JPMarticle2 <- read_lines(file_path8) %>%
  tibble(Paragraph = .)

JPMarticle2 <- JPMarticle2%>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)

file_path9 <- "data/NYTArticle.txt"

NYTArticle1 <- read_lines(file_path9) %>%
  tibble(Paragraph = .)

NYTArticle <- NYTArticle1%>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)

file_path10 <- "data/GoldmanArticle.txt"

GoldmanArticle1 <- read_lines(file_path10) %>%
  tibble(Paragraph = .)

GoldmanArticle <- GoldmanArticle1%>%
  mutate(Word = str_split(Paragraph, "\\s+")) %>%
  unnest(Word) %>%
  select(Word)


##############################Webscrape########################################## really d

# GoldmanSachsArticle <- "https://www.cnbc.com/2023/11/03/goldman-says-israel-hamas-war-co
#
# robotstxt::paths_allowed(GoldmanSachsArticle)
#
```

```
# htmloutput <- httr::GET(GoldmanSachsArticle)#gets the html content
#
# html_content <- read_html(content(htmloutput, as = "text"))#returns the html content as
#
# article_text <- html_content %>%
#   html_nodes(".ArticleBody-articleBody p") %>%  # gets the paragraphs from inside the ar
#   html_text() %>%
#   str_trim() #gets rid of excess white space
#
# cat(article_text, sep = "\n")#prints the text (for testing purposes)
#
# GoldmanSachsArticle_df <- tibble(Text = article_text)
#
# GoldmanSachsArticle_word_df <- tibble(Word = str_split(article_text, "\\s+")|> unlist()


# BloombergArticle <- "https://www.bloomberg.com/news/features/2023-10-12/israel-hamas-war
#
# robotstxt::paths_allowed(BloombergArticle)
#
# Bloomberghtmloutput <- httr::GET(BloombergArticle)#gets the html content
#
# Bloomberghtml_content <- read_html(content(Bloomberghtmloutput, as = "text"))#returns th
#
# article_text <- Bloomberghtml_content %>%
#   html_nodes("p") %>%  # gets the paragraphs from inside the article
#   html_text() %>%
#   str_trim() #gets rid of excess white space
#
# cat(article_text, sep = "\n")#prints the text (for testing purposes)
#
# Bloomberg_df <- tibble(Text = article_text)
#
# Bloomberg_word_df <- tibble(Word = str_split(article_text, "\\s+") |> unlist()) #had iss

# BrookingsArticle <- "https://www.brookings.edu/articles/the-israel-and-gaza-war-economic
#
# robotstxt::paths_allowed(BrookingsArticle)
#
# response1 <- httr::GET(url)
#
```

```
# # Parse the HTML content
# html_content1 <- read_html(content(response1, as = "text"))
#
# # Extract the text from the article
# article_text1 <- html_content1 %>%
#   html_nodes("div.post-content p") %>%
#   html_text() %>%
#   str_trim()
#
# cat(article_text1, sep = "\n")#prints the text (for testing purposes)
#
# BrookingsArticle_df <- tibble(Text = article_text1)
#
# BrookingsArticle_word_df <- tibble(Word = str_split(article_text1, "\\s+") |> unlist())


 data(stop_words)

combined_df <- rbind(JamieDimonCNBCInterview, JPowellSemiAnnualAddress, MichaelBarrTreasur
                     BloombergArticle, BrookingsArticle, DODArticle, CnnArticle, JPMarticl
  mutate(Word = tolower(Word)) |>#makes everything lowercase
  rename(word = Word)|>
  mutate(word = str_replace_all(word, "[^a-z'-]", "")) #makes it so there's no punctuation

words_to_remove <- c("leslie", "picker", "kelly","jamie", "dimon", "portant", "dont", "lot

for (wordReplace in words_to_remove) {#uses loop to check if words in combined df are a ma
  combined_df <- combined_df %>%
    mutate(word = str_replace_all(word, wordReplace, ""))
}

combined_df<- combined_df |>
    filter(nchar(word)>0) |>#removes the blank strings
    anti_join(stop_words, by="word") #removes stop words


word_counts <- combined_df |> #creates the counts, orders by descending
  group_by(word) |>
  summarize(count = n()) |>
  group_by(count) |>
  arrange(desc(count))
```

Sources: (Add as needed)

https://www.cnbc.com/2023/11/03/goldman-says-israel-hamas-war-could-majorly-impact-europes-economy.html

https://www.cnbc.com/2023/08/02/cnbc-exclusive-cnbc-transcript-jpmorgan-chase-chairman-ceo-jamie-dimon-speaks-with-cnbcs-leslie-picker-on-power-lunch-today.html

https://www.federalreserve.gov/newsevents/testimony/powell20230621a.htm

https://www.federalreserve.gov/newsevents/speech/barr20231116a.htm

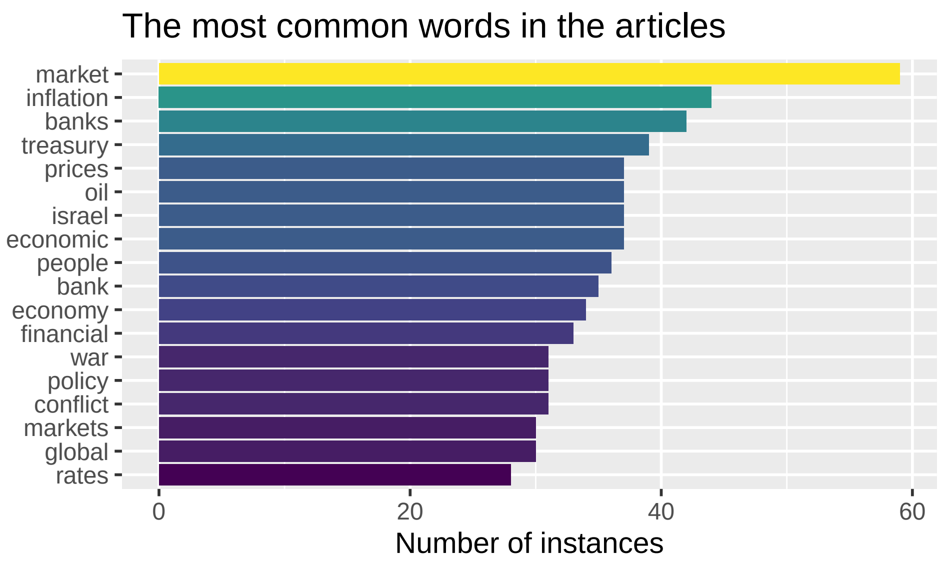
https://www.brookings.edu/articles/the-israel-and-gaza-war-economic-repercussions/

https://www.bloomberg.com/news/features/2023-11-28/evergrande-under-pressure-in-hong-kong-court-to-repay-creditors

https://www.defense.gov/News/News-Stories/Article/Article/3578196/defense-department-continues-to-stress-law-of-war-with-israel/

https://www.cnn.com/2023/10/15/economy/stocks-week-ahead-deglobalization/index.html

```r
##############################Visuals##########################################

#could make this one interactive so that you can select the number of words that get shown
Visual1Data <- word_counts |>
  filter(count>27)

ggplot(data = Visual1Data, aes(x = fct_reorder(word, count), y = count, fill = count)) +
  geom_col() +
  coord_flip() +
  scale_fill_viridis_c() +
  guides(fill = "none") +
  labs(
    x = NULL,
    y = "Number of instances",
    title = "The most common words in the articles")
```



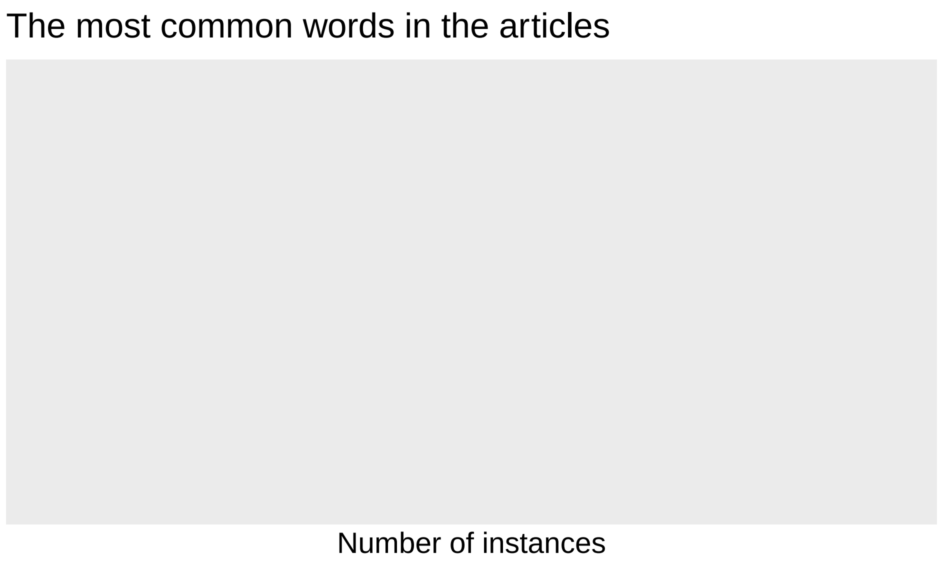
```r
api_key <- ""

api_url <- "https://api.openai.com/v1/chat/completions"

cnbc <- read_lines(file_path) |>
  paste(collapse = " ")
powell <- read_lines(file_path2) |>
  paste(collapse = " ")
barr <- read_lines(file_path3) |>
  paste(collapse = " ")
bloomberg <- read_lines(file_path4) |>
  paste(collapse = " ")
brooking <- read_lines(file_path5) |>
  paste(collapse = " ")
dod <- read_lines(file_path6) |>
  paste(collapse = " ")
cnn <- read_lines(file_path7) |>
  paste(collapse = " ")
jpm <- read_lines(file_path8) |>
  paste(collapse = " ")
nyt <- read_lines(file_path9) |>
  paste(collapse = " ")
goldman <- read_lines(file_path10) |>
  paste(collapse = " ")
```

```r
articles <- c(cnbc, powell, barr, bloomberg, brooking, dod, cnn, jpm, nyt, goldman)

postGPT <-" "

for (inst in articles){
  body <- list(
      model = "gpt-3.5-turbo-1106",
      messages = list(
        list(role = "user", content = paste(inst,
                                            " what are the three most important arguments
      max_tokens = 150,  # Adjust as needed
      temperature = 0.2  # Adjust for creativity (0.0-1.0)
  )

  response <- POST(
      url = api_url,
      add_headers(`Authorization` = paste("Bearer", api_key)),
      content_type_json(),
      body = toJSON(body, auto_unbox = TRUE)
  )
  if (http_status(response)$category == "Success") {
      result <- fromJSON(content(response, "text"))
      print(1)
      # Extract the content of the assistant's message
      if (length(result$choices) > 0 && length(result$choices$message) > 0) {
        postGPT <- paste(postGPT, result$choices$message$content, sep = "\n")
      } else {
        print("No response generated.")
      }
  } else {
      print("Error in API request:")
    }
}
```

```
[1] "Error in API request:"
[1] "Error in API request:"
[1] "Error in API request:"
[1] "Error in API request:"
[1] "Error in API request:"
[1] "Error in API request:"
[1] "Error in API request:"
```

```
[1] "Error in API request:"
[1] "Error in API request:"
[1] "Error in API request:"
```

```
data(stop_words)


postGPT <- postGPT |>
  tolower() |>
  removePunctuation() |>
  stripWhitespace()

postW <- str_split(postGPT, "\\s+")[[1]]


stops <- c("potential", "1", "2", "3", "monetary", "article", " ", "economy", "economic",
cleanW <- removeWords(postW, stops)
cleanW <- table(cleanW)
word_freq_df <- as.data.frame(cleanW, stringsAsFactors=FALSE)
names(word_freq_df) <- c("word", "freq")
word_freq_df <- word_freq_df[order(-word_freq_df$freq),]

word_freq_df<- word_freq_df |>
    anti_join(stop_words, by="word") #removes stop words


Visual2Data <- word_freq_df |>
  filter(freq>3) |>
  filter(freq<20)

ggplot(data = Visual2Data, aes(x = fct_reorder(word, freq), y = freq, fill = freq)) +
  geom_col() +
  coord_flip() +
  scale_fill_viridis_c() +
  guides(fill = "none") +
  labs(
    x = NULL,
    y = "Number of instances",
    title = "The most common words in the articles")
```



```r
#Wrangling the imports and exports data for the oil network.

#The units are B/D or barrels per day. 1 Barrel is about ~42 gallons

file_path_imports <- "data/CrudeOilImports.xlsx"

file_path_exports <- "data/CrudeOilExports.xlsx"

OilImports <- read_excel(file_path_imports, skip=2) |>
  mutate(across(c(2:44), as.character)) |>
  pivot_longer(!Country, names_to = "Year", values_to = "Imports")

OilExports <- read_excel(file_path_exports, skip=2)|>
  mutate(across(c(2:44), as.character)) |>
  pivot_longer(!Country, names_to = "Year", values_to = "Exports")

#ImportsAndExports <- left_join(OilExports, OilImports, by = 'Exports')

combined_df <- left_join(OilImports, OilExports, by = c("Country", "Year"))|>
  filter(Country!="Others") |>
   mutate(Exports = as.numeric(Exports),
         Imports = as.numeric(Imports),
         Country = ifelse(Country == "United States", "USA", Country),
         Country = ifelse(Country == "United Kingdom", "UK", Country))
```

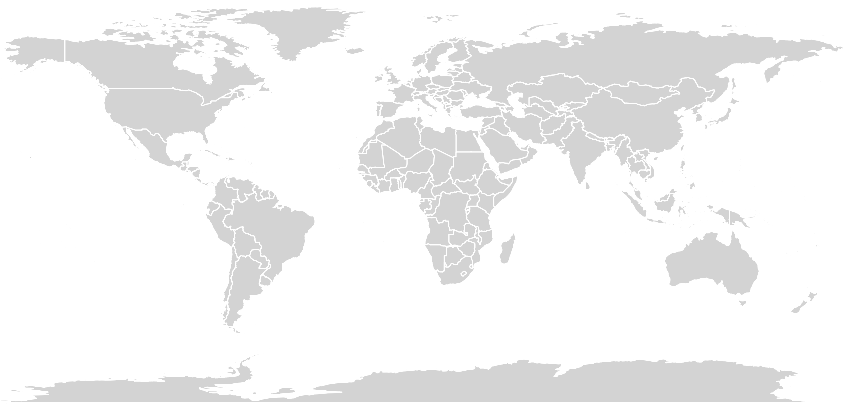
```
Warning in left_join(OilImports, OilExports, by = c("Country", "Year")): Detected an unexpec
i Row 173 of `x` matches multiple rows in `y`.
i Row 173 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `Imports = as.numeric(Imports)`.
Caused by warning:
! NAs introduced by coercion
```

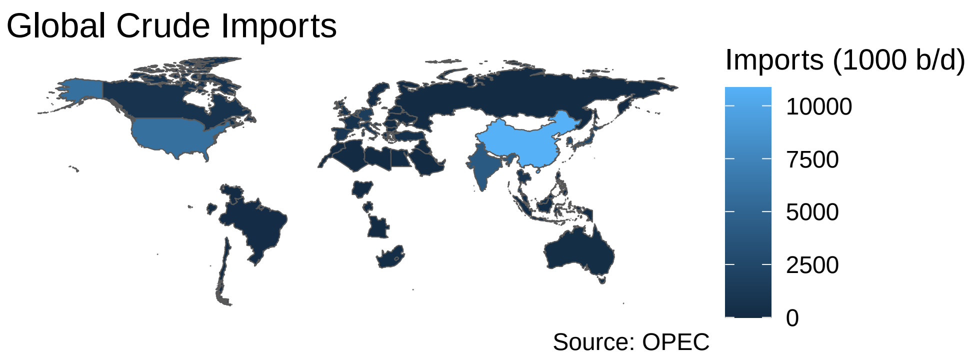
```r
monthlydata <- read.csv("data/MonthlyOil.csv") |>
#filter(FLOW %in% c("TOTEXPSB", "TOTIMPSB"))
filter(COUNTRY == "USA")


world_map2 <- maps::map("world", plot = FALSE, fill = TRUE) |>
  st_as_sf()
class(world_map2) #creates an sf object
```

```
[1] "sf"         "data.frame"
```

```r
ggplot(world_map2, aes(geometry=geom)) +
  geom_sf(fill = "lightgrey", color = "white") +
  theme_void()
```



```
world_import_export_map <- world_map2 %>%
  left_join(combined_df, by = c("ID" = "Country")) %>%
  mutate(Exports = ifelse(is.na(Exports), 0, Exports), Imports = ifelse(is.na(Imports), 0,
  filter(Year == "2020")

ggplot(world_import_export_map, aes(geometry=geom, fill = Imports)) +
  geom_sf() +
  theme_void() +
  labs(fill = "Imports (1000 b/d)"
       , title = "Global Crude Imports"
       , caption = "Source: OPEC")
```


Global Crude Imports
Imports (1000 b/d)
10000
7500
5000
2500
0
Source: OPEC