Text Mining Project_Tibble

Oluwatobi Ogunronbi

2024-04-23

Step 1: Pre-work

Installing Packages

```
#install.packages("stringi")
#install.packages("stringr")
#install.packages("qdap")
#install.packages("rJava")
#install.packages("ggthemes")hf
#install.packages("tm")
#install.packages("tidyr")
#install.packages("tidytext")
#install.packages("tidyverse")
#install.packages("ggplot2")
#install.packages("scales")
#install.packages("tokenizers")
#install.packages("reshape2")
#install.packages("tools")
#install.packages("textstem")
```

Loading installed packages

```
library(stringi)
library(stringr)
library(qdap)
```

```
## Loading required package: qdapDictionaries

## Loading required package: qdapRegex

## Loading required package: qdapTools

## Loading required package: RColorBrewer

##
## Attaching package: 'qdap'
```

```
## The following objects are masked from 'package:base':
##
##     Filter, proportions
```

```
library(rJava)
library(ggthemes)
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:qdap':
##
##     ngrams
```

```
##
## Attaching package: 'tm'
```

```
## The following objects are masked from 'package:qdap':
##
##     as.DocumentTermMatrix, as.TermDocumentMatrix
```

```
library(tidyr)
library(tidytext)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.0.2
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x ggplot2::%+%()      masks qdapRegex::%+%()
## x ggplot2::annotate() masks NLP::annotate()
## x dplyr::explain()    masks qdapRegex::explain()
## x dplyr::filter()     masks stats::filter()
## x dplyr::id()         masks qdapTools::id()
## x dplyr::lag()        masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
```

```
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(pdftools)
```

```
## Using poppler version 23.04.0
```

```
library(tools)
library(stopwords)
```

```
##
## Attaching package: 'stopwords'
##
## The following object is masked from 'package:tm':
##
##     stopwords
```

```
library(readtext)
library(dplyr)
library(tokenizers)
library(SnowballC)
library(textstem)
```

```
## Loading required package: koRpus.lang.en
## Loading required package: koRpus
## Loading required package: sylly
## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()
##
##
## Attaching package: 'koRpus'
##
## The following object is masked from 'package:readr':
##
##     tokenize
##
## The following object is masked from 'package:tm':
##
##     readTagged
##
## The following object is masked from 'package:qdap':
##
##     SMOG
##
##
```

```
## Attaching package: 'textstem'
##
## The following object is masked from 'package:qdap':
##
##     stem_words
```

```
library(broom)
```

Step 2: Import the Annual Reports

We'll write a script to recursively list all PDF files within the "DJ-AR" folder, read their contents, and organize them in a tibble with columns for the company name, year, and the text content of each report. Given the folder structure and file naming convention you described, we can extract the year and company abbreviation directly from the filenames.

```
# Path to the DJ-AR folder
pdf_path <- "/Users/bigtobi/Desktop/ITEC_724/final_project/AR_data/DJ-AR"

# Recursively list all PDF files
pdf_files <- list.files(path = pdf_path, pattern = "\\.pdf$", full.names = TRUE, recursive = TRUE)
pdf_filenames <- basename(pdf_files)

print(pdf_filenames)
```

```
##  [1] "2018_10K_UNH.pdf"  "2019_10K_UNH.pdf"  "2020_10K_UNH.pdf"
##  [4] "2021_10K_UNH.pdf"  "2022_10K_UNH.pdf"  "2019_10K_V.pdf"
##  [7] "2020_10K_V.pdf"    "2021_10K_V.pdf"    "2022_10K_V.pdf"
## [10] "2023_10K_V.pdf"    "2020_10K_JPM.pdf"  "2021_10K_JPM.pdf"
## [13] "2022_10K_JPM.pdf"  "2023_10K_JPM.pdf"  "2024_10K_JPM.pdf"
## [16] "2020_10K_AXP.pdf"  "2021_10K_AXP.pdf"  "2022_10K_AXP.pdf"
## [19] "2023_10K_AXP.pdf"  "2024_10K_AXP.pdf"  "2019_10K_HD.pdf"
## [22] "2020_10K_HD.pdf"   "2021_10K_HD.pdf"   "2022_10K_HD.pdf"
## [25] "2023_10K_HD.pdf"   "2020_10K_WMT.pdf"  "2021_10K_WMT.pdf"
## [28] "2022_10K_WMT.pdf"  "2023_10K_WMT.pdf"  "2024_10K_WMT.pdf"
## [31] "2019_10K_PG.pdf"   "2020_10K_PG.pdf"   "2021_10K_PG.pdf"
## [34] "2022_10K_PG.pdf"   "2023_10K_PG.pdf"   "2019_10K_NKE.pdf"
## [37] "2020_10K_NKE.pdf"  "2021_10K_NKE.pdf"  "2022_10K_NKE.pdf"
## [40] "2023_10K_NKE.pdf"  "2020_10K_JNJ.pdf"  "2021_10K_JNJ.pdf"
## [43] "2022_10K_JNJ.pdf"  "2023_10K_JNJ.pdf"  "2024_10K_JNJ.pdf"
## [46] "2020_10K_MRK.pdf"  "2021_10K_MRK.pdf"  "2022_10K_MRK.pdf"
## [49] "2023_10K_MRK.pdf"  "2024_10K_MRK.pdf"  "2019_10K_AMGN.pdf"
## [52] "2020_10K_AMGN.pdf" "2021_10K_AMGN.pdf" "2022_10K_AMGN.pdf"
## [55] "2023_10K_AMGN.pdf" "2019_10K_MSFT.pdf" "2020_10K_MSFT.pdf"
## [58] "2021_10K_MSFT.pdf" "2022_10K_MSFT.pdf" "2023_10K_MSFT.pdf"
## [61] "2019_10K_AAPL.pdf" "2020_10K_AAPL.pdf" "2021_10K_AAPL.pdf"
## [64] "2022_10K_AAPL.pdf" "2023_10K_AAPL.pdf" "2020_10K_CRM.pdf"
## [67] "2021_10K_CRM.pdf"  "2022_10K_CRM.pdf"  "2023_10K_CRM.pdf"
## [70] "2024_10K_CRM.pdf"  "2020_10K_INTC.pdf" "2021_10K_INTC.pdf"
## [73] "2022_10K_INTC.pdf" "2023_10K_INTC.pdf" "2024_10K_INTC.pdf"
## [76] "2019_10K_GS.pdf"   "2020_10K_GS.pdf"   "2021_10K_GS.pdf"
## [79] "2022_10K_GS.pdf"   "2023_10K_GS.pdf"
```

```r
class(pdf_filenames)
```

```
## [1] "character"
```

```r
# Function to parse filename for year and company abbreviation
parse_filename <- function(filename) {
  parts <- str_match(basename(filename), "^(\\d{4})_10K_([A-Za-z]+)")
  if(is.null(parts) || nrow(parts) == 0) {
    year <- NA
    company_abbreviation <- NA
  } else {
    year <- parts[, 2]
    company_abbreviation <- parts[, 3]
  }
  return(tibble(year = year, company_abbreviation = company_abbreviation))
}

# Example file paths for testing
test_filenames <- c("/Users/bigtobi/Desktop/ITEC_724/Project/AR_data/DJ-AR/4_HomeDepot/2022_10K_HD.pdf"
class(test_filenames)
```

```
## [1] "character"
```

```r
# Test the parsing function
parsed_test <- map_df(test_filenames, parse_filename)
print(parsed_test)
```

```
## # A tibble: 4 x 2
##   year  company_abbreviation
##   <chr> <chr>
## 1 2022  HD
## 2 2021  MSFT
## 3 2022  V
## 4 <NA>  <NA>
```

```r
# Apply parsing and text extraction using both filenames and full paths
annual_reports <- map_df(1:length(pdf_files), function(idx) {
  # Use pdf_filenames for parsing to get year and company abbreviation
  parsed_info <- parse_filename(pdf_filenames[idx])

  # Use pdf_files to extract text content from the PDF
  text_content <- paste(pdf_text(pdf_files[idx]), collapse = " ")

  # Combine parsed info with text content into a single tibble row
  parsed_info <- mutate(parsed_info, text_content = text_content)

  return(parsed_info)
})
```

View the tibble

```r
# View the first few rows of the tibble
head(annual_reports)
```

```
## # A tibble: 6 x 3
##   year  company_abbreviation text_content
##   <chr> <chr>                <chr>
## 1 2018  UNH                  "                                        UNITED STATE~
## 2 2019  UNH                  "                                      UNITED STATES\n  ~
## 3 2020  UNH                  "                                     UNITED STATES\n   ~
## 4 2021  UNH                  "                                        UNITED STATE~
## 5 2022  UNH                  "                                      UNITED STATES\n ~
## 6 2019  V                    "Table of Contents\n\n                                  ~
```

```r
#tail(annual_reports)
#class(annual_reports)
```

Include doc_id in annual reports

```r
annual_reports <- mutate(annual_reports,
         doc_id = paste(company_abbreviation, year, sep = "_"))

head(annual_reports)
```

```
## # A tibble: 6 x 4
##   year  company_abbreviation text_content                                  doc_id
##   <chr> <chr>                <chr>                                         <chr>
## 1 2018  UNH                  "                                       UNITE~ UNH_2~
## 2 2019  UNH                  "                                    UNITED STA~ UNH_2~
## 3 2020  UNH                  "                                   UNITED STAT~ UNH_2~
## 4 2021  UNH                  "                                       UNITE~ UNH_2~
## 5 2022  UNH                  "                                     UNITED ST~ UNH_2~
## 6 2019  V                    "Table of Contents\n\n                        ~ V_2019
```

Create a mapping of company abbreviations and names to their respective industries

```r
# Create a mapping of company abbreviations and names to their respective industries
industry_mapping <- data.frame(
  company_abbreviation = c("UNH", "JNJ", "MRK", "AMGN", "MSFT", "AAPL", "CRM", "INTC",
                           "GS", "V", "JPM", "AXP", "HD", "WMT", "PG", "NKE"),
  company_name = c("UnitedHealth Group", "Johnson & Johnson", "Merck & Co.", "Amgen",
                   "Microsoft", "Apple", "Salesforce", "Intel Corporation",
                   "Goldman Sachs", "Visa", "JPMorgan Chase", "American Express",
                   "Home Depot", "Walmart", "Procter & Gamble", "Nike"),
  industry = c("Healthcare", "Healthcare", "Healthcare", "Healthcare",
               "Technology", "Technology", "Technology", "Technology",
               "Financial Services", "Financial Services", "Financial Services", "Financial Services",
```

```
               "FMCG", "FMCG", "FMCG", "FMCG")
)

# Join the industry mapping to the 'annual_reports' dataframe
annual_reports <- annual_reports %>%
  left_join(industry_mapping, by = "company_abbreviation")

# Check the first few rows to confirm the join
head(annual_reports)
```

```
## # A tibble: 6 x 6
##   year  company_abbreviation text_content           doc_id company_name industry
##   <chr> <chr>                <chr>                  <chr>  <chr>        <chr>
## 1 2018  UNH                  "                    ~ UNH_2~ UnitedHealt~ Healthc~
## 2 2019  UNH                  "                    ~ UNH_2~ UnitedHealt~ Healthc~
## 3 2020  UNH                  "                    ~ UNH_2~ UnitedHealt~ Healthc~
## 4 2021  UNH                  "                    ~ UNH_2~ UnitedHealt~ Healthc~
## 5 2022  UNH                  "                    ~ UNH_2~ UnitedHealt~ Healthc~
## 6 2019  V                    "Table of Contents\n\~ V_2019 Visa         Financi~
```

Step 3: Basic Text Preprocessing

Cleaning up

```
annual_reports <- annual_reports %>%
  mutate(
    text_content = tolower(text_content),  # Convert text to lowercase
    text_content = str_remove_all(text_content, "[^\\w\\s]"),  # Remove punctuation
    text_content = str_remove_all(text_content, "\\d+")  # Remove numbers
  )
```

Step 4: Tokenization & TF Analysis

```
# Tokenization and calculating term frequency
tokenized_terms <- annual_reports %>%
  unnest_tokens(word, text_content) %>%
  count(doc_id, industry, company_abbreviation, company_name, year, word, sort = TRUE)  # Count words,

# View the top terms per company per year
head(tokenized_terms)
```

```
## # A tibble: 6 x 7
##   doc_id  industry           company_abbreviation company_name year  word      n
##   <chr>   <chr>              <chr>                <chr>        <chr> <chr> <int>
## 1 GS_2019 Financial Services GS                   Goldman Sac~ 2019  the   28841
## 2 GS_2020 Financial Services GS                   Goldman Sac~ 2020  the   21190
## 3 GS_2023 Financial Services GS                   Goldman Sac~ 2023  the   20537
```

```
## 4 GS_2019 Financial Services GS                    Goldman Sac~ 2019  of    20506
## 5 GS_2022 Financial Services GS                    Goldman Sac~ 2022  the   20036
## 6 GS_2021 Financial Services GS                    Goldman Sac~ 2021  the   19712
```

Step 5:Developing Custom Stopwords Library

```
custom_stopwords <- c(
  "financial", "fiscal", "firm", "statement", "report", "year",
  "company", "business", "date", "aa", "income", "stock", "shares", "stocks",
  "management", "operations", "performance", "results", "objective",
  "strategy", "risk", "opportunity", "outlook", "significantly",
  "approximately", "primarily", "including", "regarding", "concerning",
  "due to", "pursuant", "accordance", "thereof", "therein", "hereby",
  "hereto", "hereunder", "usd", "ebitda", "gaap", "qoq", "yoy", "fy",
  "qtr", "one", "two", "first", "second", "third", "quarter", "annual",
  "monthly", "weekly", "day", "month", "year", "law", "regulation",
  "section", "act", "legal", "compliance", "regulatory", "filings",
  "securities", "exchange", "commission", "corporation", "incorporated",
  "plc", "llc", "ltd", "group", "holdings", "united states", "us", "usa",
  "america", "north america", "international", "global", "worldwide", "%",
  "return", "returns", "eps", "earnings per share", "roe", "return on equity",
  "taxes", "ratio", "cash", "cash flow", "dividend", "revenue", "earnings",
  "expense", "expenses", "asset", "liability", "leverage",
  "patient", "customer", "marketing", "sales",
  "sox", "sec", "sec filing", "ifrs", "audit",

  # Company names
  "unitedhealth group", "unitedhealth", "uhg",
  "johnson & johnson", "johnson and johnson", "johnson", "j&j",
  "merck", "merck & co", "mrk",
  "amgen", "amgn",
  "microsoft", "msft",
  "apple", "aapl",
  "salesforce", "crm",
  "intel", "intel corporation", "intc",
  "goldman sachs", "goldman", "gs", "sachs",
  "visa", "v",
  "jpmorgan chase", "jpmorgan", "jpm", "chase",
  "american express", "amex", "axp",
  "home depot", "hd",
  "walmart", "wmt",
  "procter & gamble", "pg", "p&g", "procter", "gamble",
  "nike", "nke"
)

# Create a tibble for custom stopwords to match tidytext format
custom_stopwords_df <- tibble(word = custom_stopwords, lexicon = "custom")

# Combine custom stopwords with standard English stopwords
all_stopwords <- bind_rows(stop_words, custom_stopwords_df)
```

#Remove Stopwords #### From tokenized

```r
# Removing stopwords
clean_terms <- tokenized_terms %>%
  anti_join(all_stopwords, by = "word")

head(clean_terms)
```

```
## # A tibble: 6 x 7
##   doc_id  industry           company_abbreviation company_name year  word      n
##   <chr>   <chr>              <chr>                <chr>        <chr> <chr> <int>
## 1 GS_2019 Financial Services GS                   Goldman Sac~ 2019  seri~  3176
## 2 GS_2019 Financial Services GS                   Goldman Sac~ 2019  pref~  2009
## 3 GS_2020 Financial Services GS                   Goldman Sac~ 2020  award  1368
## 4 GS_2019 Financial Services GS                   Goldman Sac~ 2019  award  1358
## 5 GS_2023 Financial Services GS                   Goldman Sac~ 2023  award  1345
## 6 GS_2020 Financial Services GS                   Goldman Sac~ 2020  pref~  1333
```

```r
# Let's first tokenize the text content

# Tokenize the text content
full_tokens <- annual_reports %>%
  unnest_tokens(word, text_content)

# Ensure all tokens are lowercase (assuming your stopwords are all lowercase)
#tokens$word <- tolower(tokens$word)

# Load standard English stopwords from the tidytext package
#data("stop_words")

# Combine custom stopwords with tidytext's stopwords if you have custom ones
# custom_stopwords <- tibble(word = c("your", "custom", "stopwords", "..."))
# all_stopwords <- bind_rows(stop_words, custom_stopwords)

# Now, filter out the stopwords from the tokens
clean_full_tokens <- full_tokens %>%
  anti_join(all_stopwords, by = "word")

# If you need to reassemble the cleaned text for each document:
clean_annual_reports <- clean_full_tokens %>%
  group_by(doc_id, industry, year, company_abbreviation, company_name) %>%
  summarize(text_content = paste(word, collapse = " ")) %>%
  ungroup()
```

From untokenized

```
## `summarise()` has grouped output by 'doc_id', 'industry', 'year',
## 'company_abbreviation'. You can override using the `.groups` argument.
```

```r
head(clean_annual_reports)
```

```
## # A tibble: 6 x 6
```

```
##   doc_id    industry   year  company_abbreviation company_name text_content
##   <chr>     <chr>      <chr> <chr>                <chr>        <chr>
## 1 AAPL_2019 Technology 2019  AAPL                 Apple        united washingto~
## 2 AAPL_2020 Technology 2020  AAPL                 Apple        united washingto~
## 3 AAPL_2021 Technology 2021  AAPL                 Apple        united washingto~
## 4 AAPL_2022 Technology 2022  AAPL                 Apple        united washingto~
## 5 AAPL_2023 Technology 2023  AAPL                 Apple        united washingto~
## 6 AMGN_2019 Healthcare 2019  AMGN                 Amgen        united washingto~
```

```
# clean_annual_reports now contains the cleaned text_content without stopwords
```

Research Question 1

Term Frequency-Inverse Document Frequency (TF-IDF) for the words across the annual reports

```
# Document Frequency (DF)
# Count how many documents each word appears in
document_freq <- clean_terms %>%
  group_by(word) %>%
  summarise(n_docs = n_distinct(year))

# 3. Inverse Document Frequency (IDF)
# Calculate IDF
total_documents <- n_distinct(clean_terms$year)
idf <- mutate(document_freq, idf = log(total_documents / n_docs))

# 4. TF-IDF Calculation
# Join TF and IDF values and calculate TF-IDF
tf_idf <- clean_terms %>%
  inner_join(idf, by = "word") %>%
  mutate(tf_idf = n * idf)

# Optional step: Remove extremely common words if necessary
tf_idf <- tf_idf %>%
  filter(!word %in% c("the", "of", "and", "to", "in", "for", "on", "with", "as", "by"))

# Sort by TF-IDF to find the most important words
tf_idf <- tf_idf %>%
  arrange(desc(tf_idf))

# View the results
head(tf_idf)
```

```
## # A tibble: 6 x 10
##   doc_id    industry  company_abbreviation company_name year  word      n n_docs
##   <chr>     <chr>     <chr>                <chr>        <chr> <chr> <int>  <int>
## 1 AMGN_2023 Healthca~ AMGN                 Amgen        2023  lice~   280      1
## 2 GS_2019   Financia~ GS                   Goldman Sac~ 2019  seri~  3176      6
## 3 AMGN_2023 Healthca~ AMGN                 Amgen        2023  comm~   238      1
## 4 V_2019    Financia~ V                    Visa         2019  swing   265      2
```

```
## 5 GS_2023   Financia~ GS                   Goldman Sac~ 2023  mrt      96      1
## 6 AMGN_2023 Healthca~ AMGN                 Amgen        2023  cell~   550      5
## # i 2 more variables: idf <dbl>, tf_idf <dbl>
```

```
tf_idf
```

```
## # A tibble: 345,476 x 10
##    doc_id    industry company_abbreviation company_name year  word      n n_docs
##    <chr>     <chr>    <chr>                <chr>        <chr> <chr> <int>  <int>
##  1 AMGN_2023 Healthc~ AMGN                 Amgen        2023  lice~   280      1
##  2 GS_2019   Financi~ GS                   Goldman Sac~ 2019  seri~  3176      6
##  3 AMGN_2023 Healthc~ AMGN                 Amgen        2023  comm~   238      1
##  4 V_2019    Financi~ V                    Visa         2019  swing   265      2
##  5 GS_2023   Financi~ GS                   Goldman Sac~ 2023  mrt      96      1
##  6 AMGN_2023 Healthc~ AMGN                 Amgen        2023  cell~   550      5
##  7 GS_2020   Financi~ GS                   Goldman Sac~ 2020  seri~  1192      6
##  8 JPM_2021  Financi~ JPM                  JPMorgan Ch~ 2021  loans  1150      6
##  9 JPM_2020  Financi~ JPM                  JPMorgan Ch~ 2020  loans  1108      6
## 10 AMGN_2019 Healthc~ AMGN                 Amgen        2019  beig~   473      5
## # i 345,466 more rows
## # i 2 more variables: idf <dbl>, tf_idf <dbl>
```

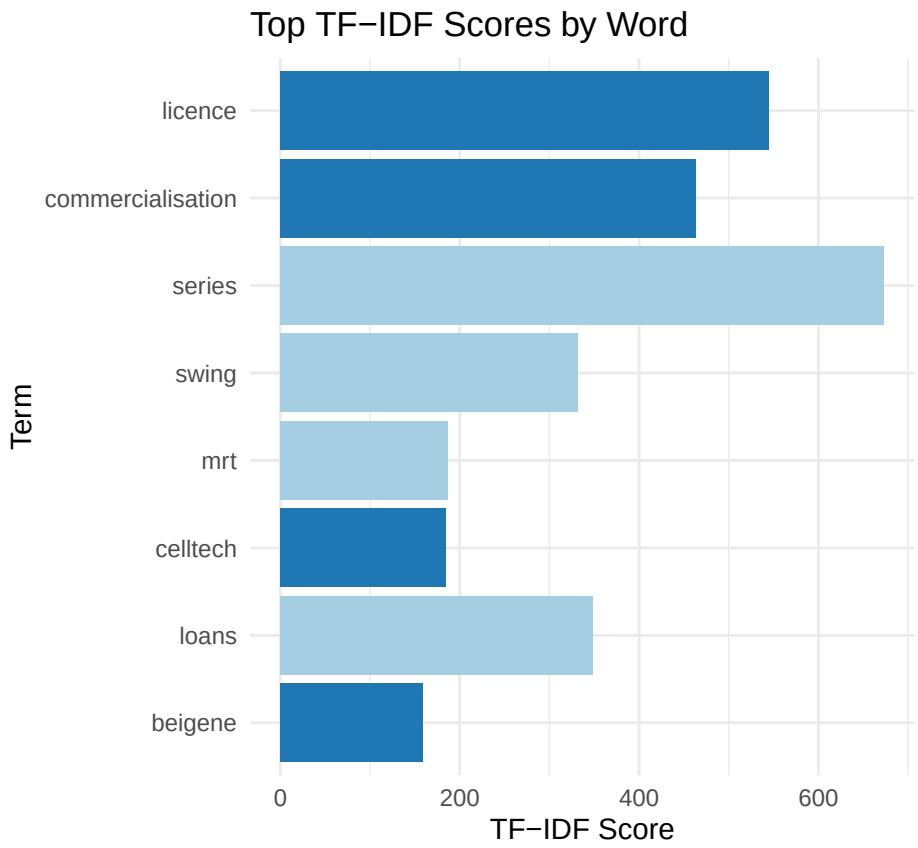
The head of the tf_idf tibble points to terms with high TF-IDF scores, highlighting words like "licence" and "commercialisation" for AMGN in 2023, and "series" and "swing" for companies like GS and V in various years. High scores signify these terms' uniqueness or emphasis within particular documents, indicating a specific focus for those companies in those years. For example, the unique terms for AMGN suggest a focus on licensing and commercialization in 2023. The term "series" for GS, despite appearing in several documents, still garners a significant score, suggesting a potential emphasis on financial instruments or classifications like series bonds or stock series.

Conversely, the tail of the tf_idf tibble showcases terms with lower TF-IDF scores, which include "commodities," "financings," and "arbitration." These terms are more common across multiple documents and do not feature prominently within any single report. Their presence across various documents reduces their TF-IDF score, implying they are part of the standard lexicon in annual reports and are not indicative of particular corporate priorities or trends.

In synthesis, the high TF-IDF terms are potentially more informative about a company's strategic focus in a given year, hinting at specific initiatives or concerns that are top of mind. They warrant further investigation into the context of their usage within the reports. Low TF-IDF terms, while relevant to the industry and necessary in reporting, offer less in the way of distinguishing between the strategic directions of different companies or identifying sector-wide trends. They serve as background industry language that, while important, is not as indicative of individual company focus or unique industry developments.

```
# Let's take the top 10 words for simplicity
top_tfidf <- tf_idf %>%
  arrange(desc(tf_idf)) %>%
  top_n(10, tf_idf)

# Create a bar chart
ggplot(top_tfidf, aes(x = reorder(word, tf_idf), y = tf_idf, fill = industry)) +
  geom_bar(stat = "identity") +
```

```
  coord_flip() +  # Flip coordinates to make it easier to read long words
  labs(title = "Top TF-IDF Scores by Word",
       x = "Term",
       y = "TF-IDF Score") +
  theme_minimal() +
  scale_fill_brewer(palette = "Paired")  # Optional: use a color palette for visual distinction
```

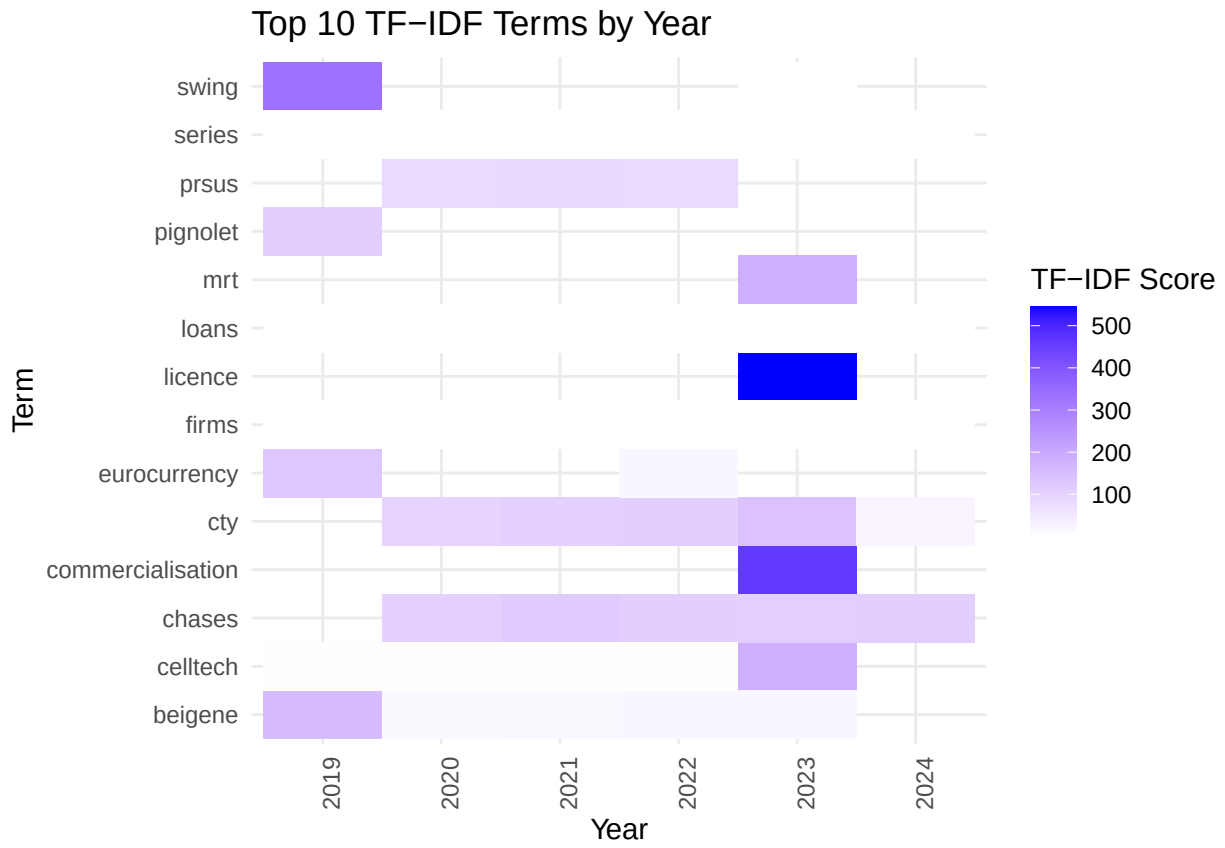


Bar Chart for Top TF-IDF Scores

Heatmap of TF-IDF by Company and Year

```
# Assuming `tf_idf` is your dataframe with the TF-IDF results
# Select the top 10 words by TF-IDF score
top_terms <- tf_idf %>%
  arrange(desc(tf_idf)) %>%
  slice(1:30) %>%
  .$word  # Extract just the words

# Filter the original tf_idf data to only include the top terms
top_tfidf <- tf_idf %>%
  filter(word %in% top_terms)

# Create the heatmap
ggplot(top_tfidf, aes(x = year, y = word, fill = tf_idf)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Top 10 TF-IDF Terms by Year",
       x = "Year",
```

```
       y = "Term",
       fill = "TF-IDF Score") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Rotate x labels for better readability
```



When taken together, these visualizations reveal both the overall importance of certain terms within the corpus (bar chart) and their specific relevance across different temporal segments (heatmap). For example, while "licence" has a significant TF-IDF score overall, it is particularly dominant in one year, suggesting a pivotal event or initiative related to licensing in that year.

The use of terms such as "commercialisation," "series," and "swing" across several years points to ongoing strategic themes within their respective industries. Conversely, some terms appear to be more transient, emerging strongly in one year and fading in others, potentially reflecting temporary strategic shifts, market conditions, or regulatory changes.

N-Grams Analysis

```
# Generate bigrams
bigrams <- annual_reports %>%
  unnest_tokens(bigram, text_content, token = "ngrams", n = 2) %>%
  group_by(doc_id, industry, company_name, year) %>%
  count(bigram, sort = TRUE) %>%
  ungroup() %>%
  separate(bigram, into = c("word1", "word2"), sep = " ") %>%
```

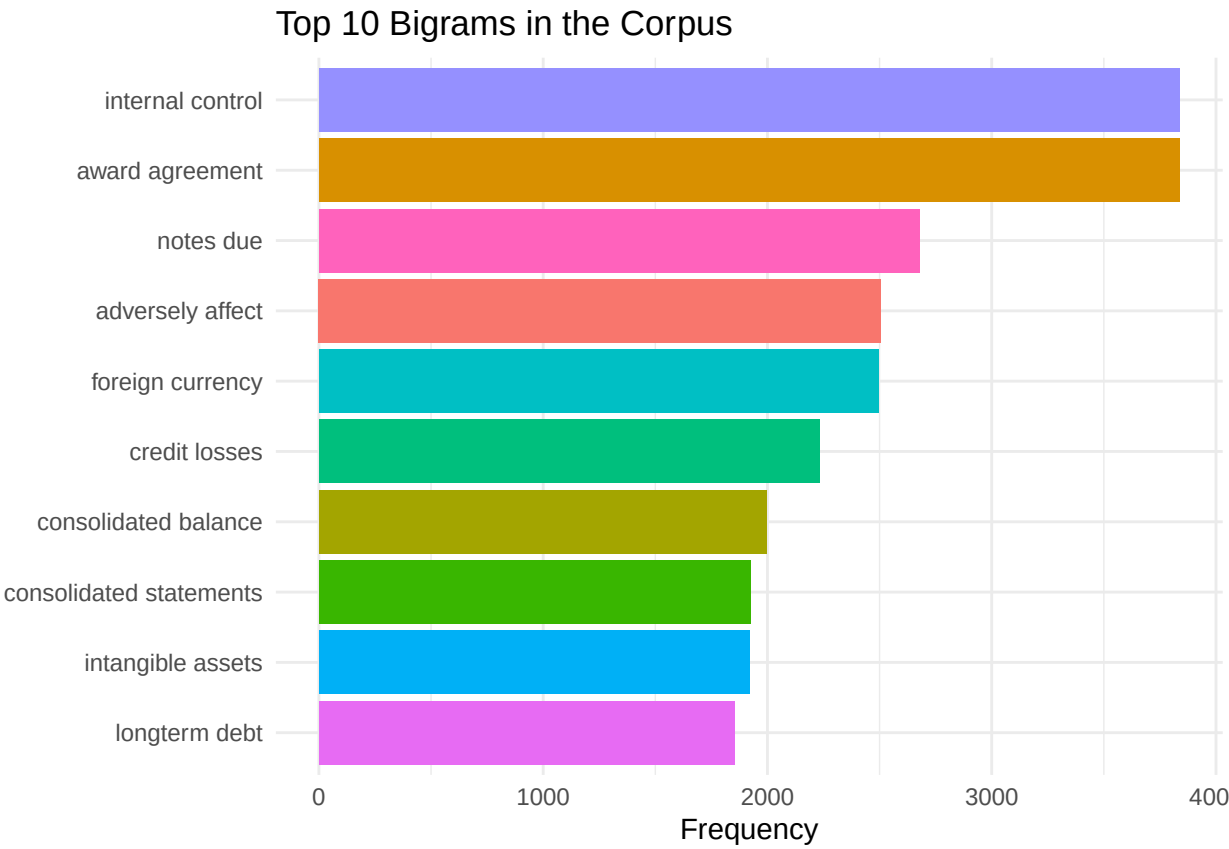
```
  mutate(bigram = paste(word1, word2, sep = " ")) %>%  # Add this line to reassemble bigrams
  #select(-word1, -word2) %>%  # Remove individual word columns
  anti_join(all_stopwords, by = c("word1" = "word")) %>%
  anti_join(all_stopwords, by = c("word2" = "word"))


# View the most common bigrams (adjust as necessary for specific analyses)
head(bigrams)
```

Bigrams analysis

```
## # A tibble: 6 x 8
##   doc_id    industry           company_name  year  word1      word2     n bigram
##   <chr>     <chr>              <chr>         <chr> <chr>      <chr> <int> <chr>
## 1 GS_2020   Financial Services Goldman Sachs 2020  award      agre~   560 award~
## 2 GS_2023   Financial Services Goldman Sachs 2023  award      agre~   558 award~
## 3 GS_2019   Financial Services Goldman Sachs 2019  award      agre~   556 award~
## 4 V_2019    Financial Services Visa          2019  administr~ agent   534 admin~
## 5 GS_2022   Financial Services Goldman Sachs 2022  award      agre~   515 award~
## 6 AMGN_2022 Healthcare         Amgen         2022  administr~ agent   447 admin~
```

```
#class(bigrams)
#bigrams[bigrams$word1 == "goldman"]
```

```
# Aggregate bigram counts across all industries and years
aggregated_bigrams <- bigrams %>%
  group_by(word1, word2) %>%
  summarize(n = sum(n), .groups = 'drop') %>%
  arrange(desc(n))

# Take the top 10 bigrams for visualization
top_bigrams <- head(aggregated_bigrams, 10)

# Combine word1 and word2 into a bigram column for the top bigrams
top_bigrams <- top_bigrams %>%
  unite("bigram", word1, word2, sep = " ")

# Plot the top bigrams
ggplot(top_bigrams, aes(x = reorder(bigram, n), y = n, fill = bigram)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = NULL, y = "Frequency", title = "Top 10 Bigrams in the Corpus") +
  theme_minimal() +
  theme(legend.position = "none")
```



Bigrams Plot

```
# Generate trigrams
trigrams <- annual_reports %>%
  unnest_tokens(trigram, text_content, token = "ngrams", n = 3) %>%
  select(doc_id, industry, company_name, year, trigram) %>%
  count(doc_id, industry, company_name, year, trigram, sort = TRUE)

# Separate the trigram into three words
trigram_separated <- trigrams %>%
  separate(trigram, into = c("word1", "word2", "word3"), sep = " ")

# Filter out trigrams that have stopwords as any word if necessary
trigram_separated <- trigram_separated %>%
  anti_join(all_stopwords, by = c("word1" = "word")) %>%
  anti_join(all_stopwords, by = c("word2" = "word")) %>%
  anti_join(all_stopwords, by = c("word3" = "word"))

# View the most common trigrams
print(trigram_separated)
```

Trigrams analysis

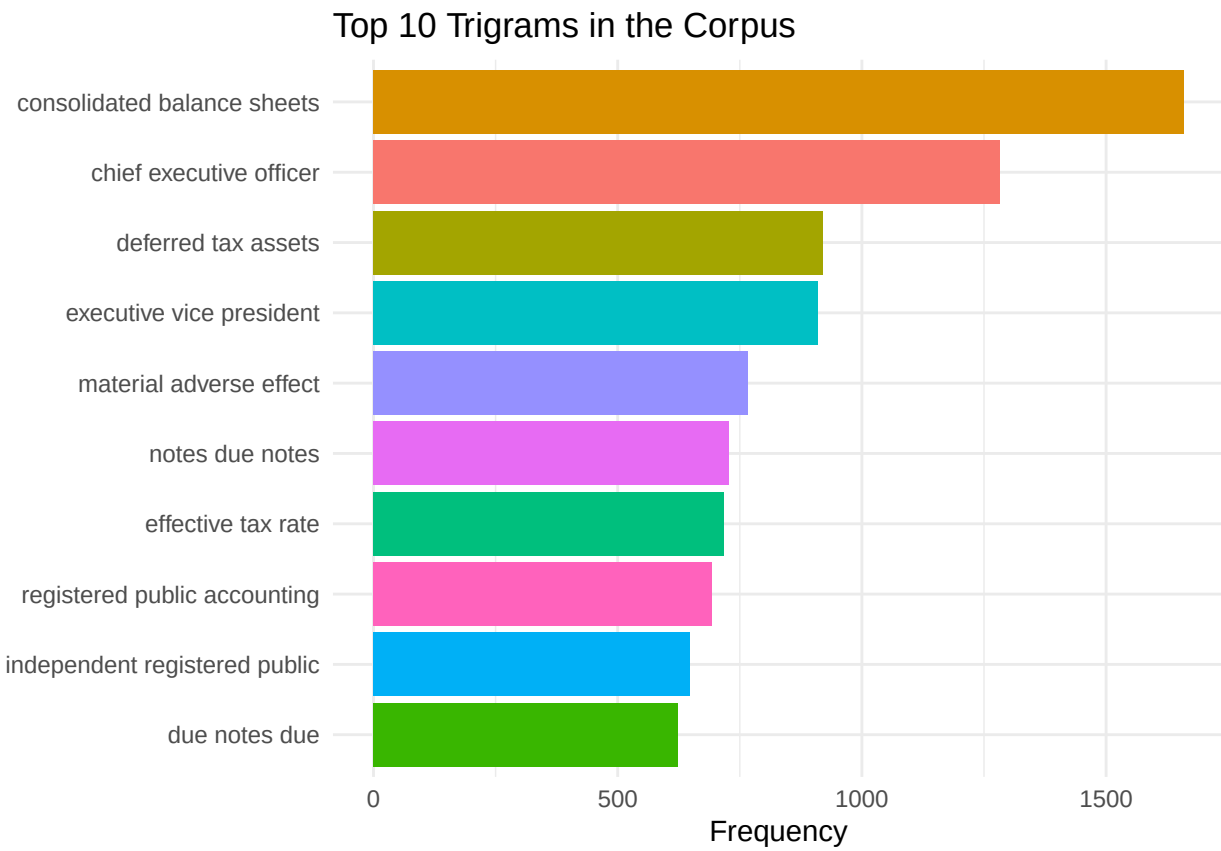
```
## # A tibble: 447,665 x 8
##    doc_id    industry          company_name     year  word1   word2 word3     n
```

```
##    <chr>     <chr>              <chr>           <chr> <chr>   <chr> <chr> <int>
##  1 GS_2019   Financial Services Goldman Sachs   2019  rights  pref~ priv~   137
##  2 GS_2019   Financial Services Goldman Sachs   2019  duly    auth~ comm~   134
##  3 WMT_2023  FMCG               Walmart         2023  purdue  phar~ lp      127
##  4 WMT_2021  FMCG               Walmart         2021  purdue  phar~ lp      123
##  5 WMT_2020  FMCG               Walmart         2020  purdue  phar~ lp      119
##  6 WMT_2022  FMCG               Walmart         2022  purdue  phar~ lp      118
##  7 AMGN_2023 Healthcare         Amgen           2023  territ~ comm~ lead    108
##  8 PG_2019   FMCG               Procter & Gamble 2019 princi~ dome~ manu~    95
##  9 AMGN_2023 Healthcare         Amgen           2023  licens~ anti~ prod~    92
## 10 WMT_2021  FMCG               Walmart         2021  va      cir   ct       83
## # i 447,655 more rows
```

```
print(head(trigrams))
```

```
## # A tibble: 6 x 6
##   doc_id  industry           company_name  year  trigram               n
##   <chr>   <chr>              <chr>         <chr> <chr>             <int>
## 1 GS_2019 Financial Services Goldman Sachs 2019  of the corporation   706
## 2 GS_2019 Financial Services Goldman Sachs 2019  the holders of       644
## 3 GS_2023 Financial Services Goldman Sachs 2023  as of december       516
## 4 GS_2019 Financial Services Goldman Sachs 2019  board of directors   514
## 5 GS_2022 Financial Services Goldman Sachs 2022  as of december       509
## 6 GS_2021 Financial Services Goldman Sachs 2021  as of december       498
```

```
# Aggregate trigram counts across all industries and years
aggregated_trigrams <- trigram_separated %>%
  group_by(word1, word2, word3) %>%
  summarize(n = sum(n), .groups = 'drop') %>%
  arrange(desc(n))

# Take the top trigrams for visualization
top_trigrams <- head(aggregated_trigrams, 10)

# Combine the separated words back into a trigram column for the top trigrams
top_trigrams <- top_trigrams %>%
  unite("trigram", word1, word2, word3, sep = " ")

# Plot the top trigrams
ggplot(top_trigrams, aes(x = reorder(trigram, n), y = n, fill = trigram)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = NULL, y = "Frequency", title = "Top 10 Trigrams in the Corpus") +
  theme_minimal() +
  theme(legend.position = "none")
```



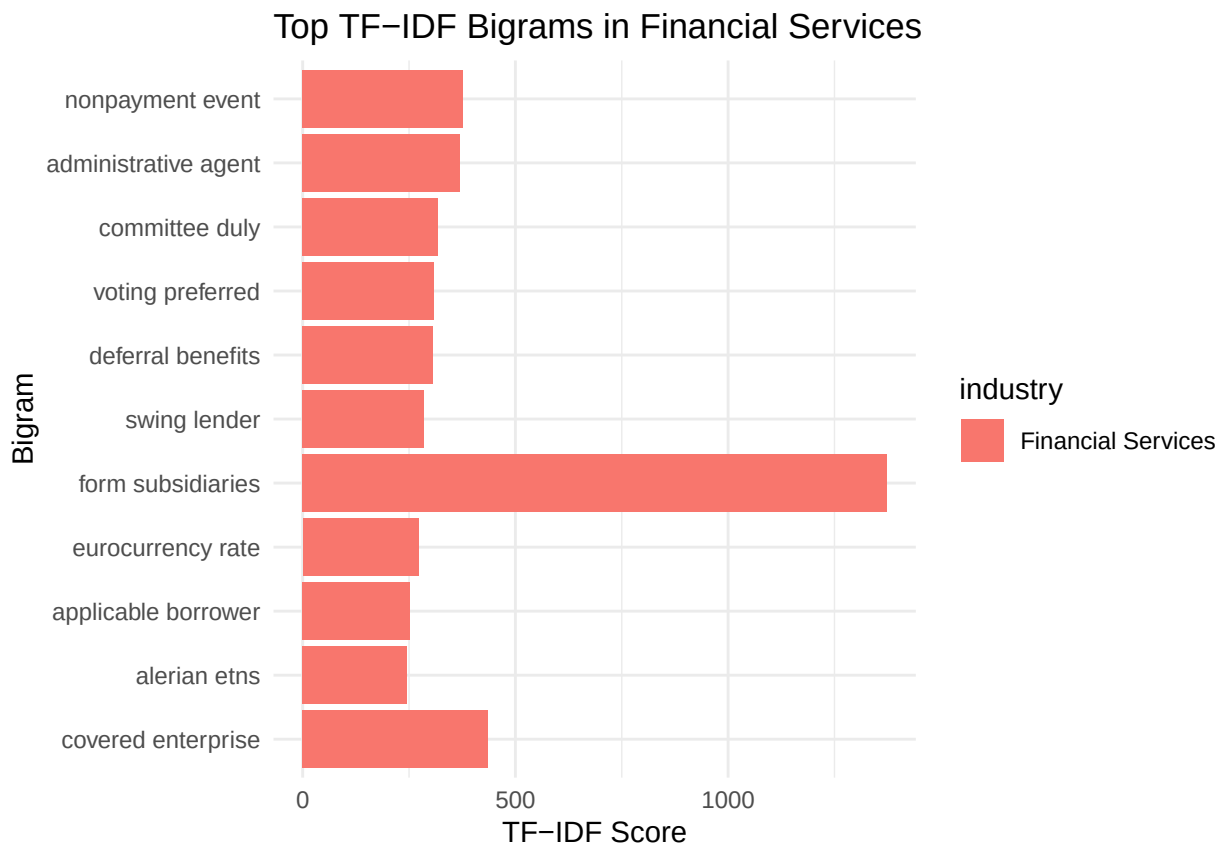
Trigrams Plot

Research Question 3 #### Bigrams For one industry

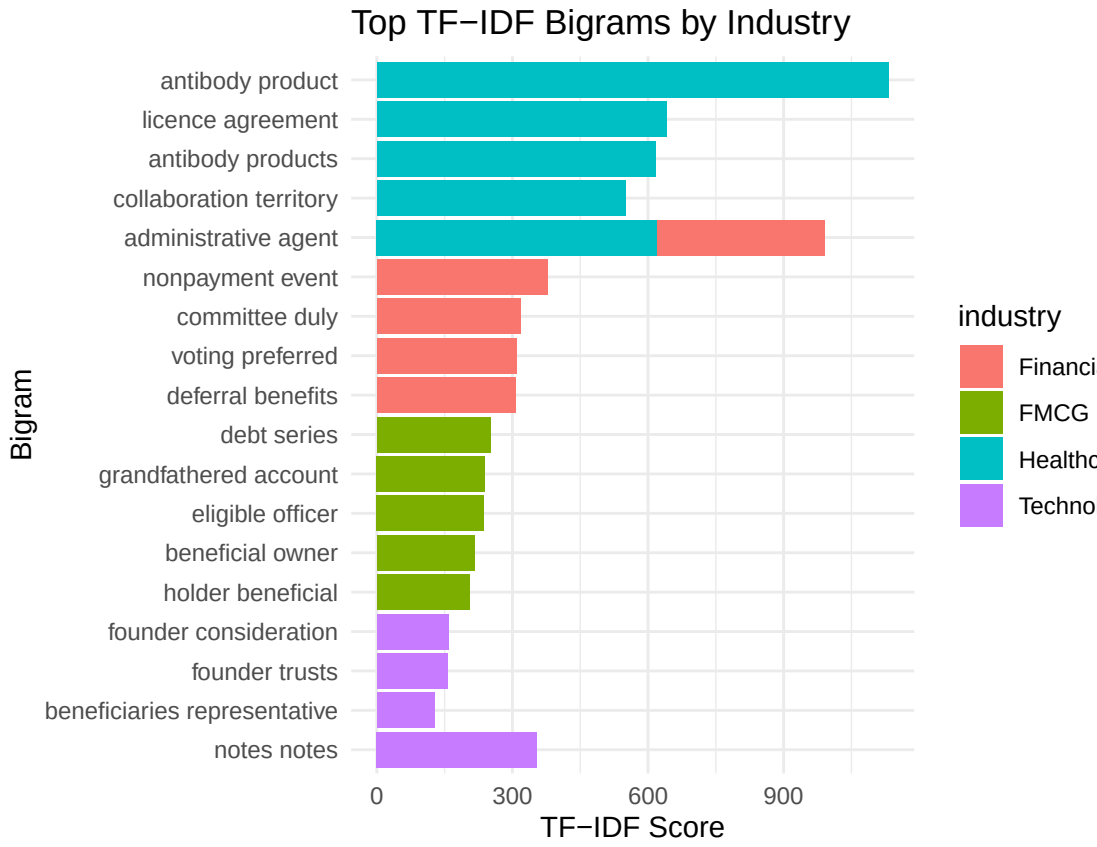
```
# Step 1: Generate bigrams
bigrams3 <- clean_annual_reports %>%
  unnest_tokens(bigram, text_content, token = "ngrams", n = 2) %>%
  group_by(doc_id, industry, year) %>%
  ungroup()

# Step 2: Count the frequency of bigrams within each industry
bigram_freq <- bigrams3 %>%
  count(doc_id, industry, company_abbreviation, bigram) %>%
  anti_join(all_stopwords, by = c("bigram" = "word"))


# Steps 3-4: Calculate the number of documents per industry that each bigram appears in
bigram_document_freq <- bigram_freq %>%
  group_by(bigram, industry) %>%
  summarize(n_docs = n_distinct(doc_id)) %>%
  ungroup() %>%
  left_join(bigram_freq %>%
              group_by(industry) %>%
              summarize(total_docs = n_distinct(doc_id)) %>%
              ungroup(),
            by = "industry") %>%
  mutate(idf = log(total_docs / n_docs),
         total_docs = NULL)
```

```
## `summarise()` has grouped output by 'bigram'. You can override using the
## `.groups` argument.
```

```r
# Steps 5-6: Calculate TF-IDF
tf_idf_by_industry <- bigram_freq %>%
  left_join(bigram_document_freq, by = c("bigram", "industry")) %>%
  mutate(tf_idf = n * idf)

# Steps 7-8: Filter and view the top bigrams for a single industry
top_tfidf_financial <- tf_idf_by_industry %>%
  arrange(desc(tf_idf)) %>%
  group_by(industry) %>%
  top_n(15, tf_idf) %>%
  ungroup() %>%
  filter(industry == "Financial Services")

# Step 9: Visualize the top TF-IDF bigrams for the financial industry
ggplot(top_tfidf_financial, aes(x = reorder(bigram, tf_idf), y = tf_idf, fill = industry)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Bigram", y = "TF-IDF Score", title = "Top TF-IDF Bigrams in Financial Services") +
  theme_minimal()
```



```r
# Steps 7-8: Filter and view the top bigrams for all industries
top_tfidf_all <- tf_idf_by_industry %>%
  arrange(industry, desc(tf_idf)) %>%
  #arrange(industry) %>%
  group_by(industry) %>%
  top_n(5, tf_idf) %>%
  ungroup()

# Filter s
# Step 9: Visualize the top TF-IDF bigrams for all industries
ggplot(top_tfidf_all, aes(x = reorder(bigram, tf_idf), y = tf_idf, fill = industry)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Bigram", y = "TF-IDF Score", title = "Top TF-IDF Bigrams by Industry") +
  theme_minimal()
```



Bigrams for all industry

```r
# Assuming 'bigrams' contains your bigram data with 'company_abbreviation' and 'industry' columns.
# First, count the bigrams for each company's documents.
bigram_counts <- bigrams %>%
  count(company_name, bigram, sort = TRUE)
```

```r
# Calculate the document frequency for each bigram across all companies.
document_freq <- bigram_counts %>%
  group_by(bigram) %>%
  summarize(n_docs = n_distinct(company_name)) %>%
  ungroup()

# Calculate IDF for each bigram.
idf <- document_freq %>%
  mutate(idf = log(n_distinct(bigrams$company_name) / n_docs))

# Join with the bigram counts to calculate the TF-IDF score for each bigram in each company's documents
tf_idf <- bigram_counts %>%
  left_join(idf, by = "bigram") %>%
  mutate(tf_idf = n * idf)

# Now, select the top bigrams for each company based on the TF-IDF score.
top_tfidf_by_company <- tf_idf %>%
  group_by(company_name) %>%
  top_n(3, tf_idf) %>%
  ungroup()

# Optionally, you can also join back with the industry information if needed.
top_tfidf_by_company <- top_tfidf_by_company %>%
  left_join(bigrams %>% select(company_name, industry) %>% distinct(), by = "company_name")

# Now you can view or save the result
print(head(top_tfidf_by_company))
```

Bigrams for companies

```
## # A tibble: 6 x 7
##   company_name     bigram                 n n_docs   idf tf_idf industry
##   <chr>            <chr>              <int>  <int> <dbl>  <dbl> <chr>
## 1 American Express aaa certificates       5      1  2.77   13.9 Financial Serv~
## 2 American Express aaa notes              5      1  2.77   13.9 Financial Serv~
## 3 American Express aac statutory          5      1  2.77   13.9 Financial Serv~
## 4 American Express abandoned property     5      1  2.77   13.9 Financial Serv~
## 5 American Express abusive debt           5      1  2.77   13.9 Financial Serv~
## 6 American Express accelerate payments    5      1  2.77   13.9 Financial Serv~
```

```r
print(tail(top_tfidf_by_company))
```

```
## # A tibble: 6 x 7
##   company_name bigram                 n n_docs   idf tf_idf industry
##   <chr>        <chr>              <int>  <int> <dbl>  <dbl> <chr>
## 1 Walmart      winter storms          5      1  2.77   13.9 FMCG
## 2 Walmart      withstand challenge    5      1  2.77   13.9 FMCG
## 3 Walmart      worker safety          5      1  2.77   13.9 FMCG
## 4 Walmart      world save             5      1  2.77   13.9 FMCG
## 5 Walmart      worms bot              5      1  2.77   13.9 FMCG
## 6 Walmart      yearend balances       5      1  2.77   13.9 FMCG
```

```r
# Create a summary table for the top bigrams by company
summary_table <- top_tfidf_by_company %>%
  arrange(industry, company_name, desc(tf_idf)) %>%
  group_by(industry, company_name) %>%
  slice_max(order_by = tf_idf, n = 3) %>%
  ungroup() %>%
  select(company_name, bigram, tf_idf)

# Print the table to the R console
#print(summary_table)

# If the table is too large, you can use the `head` function to display the first few rows
print(head(summary_table))
```

```
## # A tibble: 6 x 3
##   company_name bigram                tf_idf
##   <chr>        <chr>                  <dbl>
## 1 Home Depot   accept demand           13.9
## 2 Home Depot   access ratings          13.9
## 3 Home Depot   accounting charges      13.9
## 4 Home Depot   accrued salaries        13.9
## 5 Home Depot   accrued selfinsurance   13.9
## 6 Home Depot   achieved recipients     13.9
```

```r
print(tail(summary_table))
```

```
## # A tibble: 6 x 3
##   company_name bigram                   tf_idf
##   <chr>        <chr>                     <dbl>
## 1 Salesforce   xbrl included              13.9
## 2 Salesforce   xlri jamshedpur            13.9
## 3 Salesforce   yearoveryear compounding   13.9
## 4 Salesforce   yearoveryear total         13.9
## 5 Salesforce   yen canadian               13.9
## 6 Salesforce   zealand sfdc               13.9
```

```r
# Or write it out to a CSV file for opening in Excel or similar
#write.csv(summary_table, "top_bigrams_by_company.csv", row.names = FALSE)
```

```r
# View the top trigrams for each company
top_trigrams_by_company <- trigram_separated %>%
  group_by(company_name) %>%
  top_n(1, n) %>%
  ungroup()

# Print the top trigrams for each company
print(top_trigrams_by_company)
```

Trigrams

```
## # A tibble: 18 x 8
##    doc_id    industry           company_name       year  word1 word2 word3     n
##    <chr>     <chr>              <chr>              <chr> <chr> <chr> <chr> <int>
##  1 GS_2019   Financial Services Goldman Sachs      2019  righ~ pref~ priv~   137
##  2 WMT_2023  FMCG               Walmart            2023  purd~ phar~ lp      127
##  3 AMGN_2023 Healthcare         Amgen              2023  terr~ comm~ lead    108
##  4 PG_2019   FMCG               Procter & Gamble   2019  prin~ dome~ manu~    95
##  5 V_2019    Financial Services Visa               2019  revo~ cred~ agre~    77
##  6 JPM_2021  Financial Services JPMorgan Chase     2021  na    na    na       76
##  7 JPM_2024  Financial Services JPMorgan Chase     2024  cons~ bala~ shee~    76
##  8 MRK_2023  Healthcare         Merck & Co.        2023  chief exec~ offi~    63
##  9 CRM_2022  Technology         Salesforce         2022  equi~ ince~ plan     61
## 10 UNH_2022  Healthcare         UnitedHealth Group 2022  mill~ notes due      60
## 11 AAPL_2023 Technology         Apple              2023  due   notes due      55
## 12 AAPL_2023 Technology         Apple              2023  notes due   notes    55
## 13 HD_2023   FMCG               Home Depot         2023  seni~ notes due      46
## 14 NKE_2020  FMCG               Nike               2020  cons~ bala~ shee~    40
## 15 AXP_2024  Financial Services American Express   2024  net   writ~ rate     39
## 16 INTC_2021 Technology         Intel Corporation  2021  dec   dec   dec      38
## 17 JNJ_2022  Healthcare         Johnson & Johnson  2022  effe~ tax   rate     32
## 18 MSFT_2019 Technology         Microsoft          2019  york  mell~ trust    30
```

Research Question 2

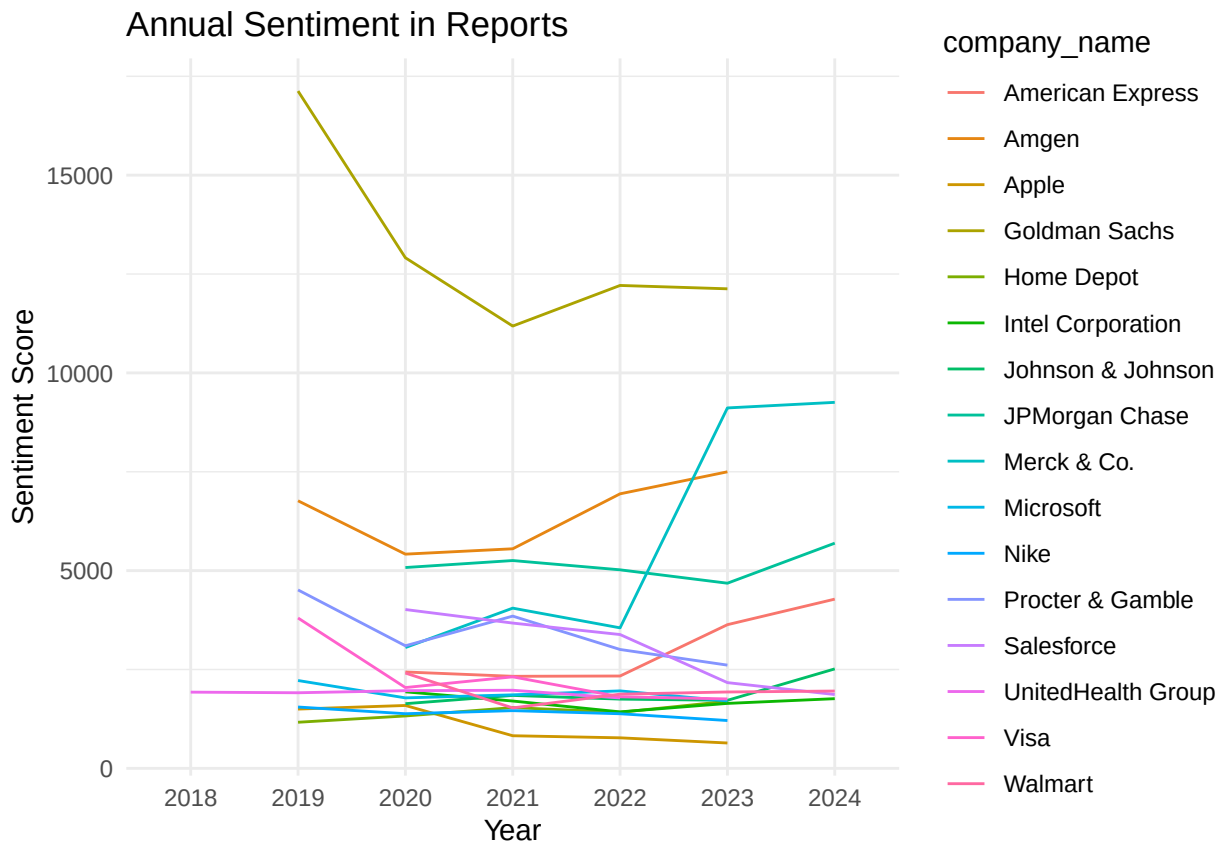
```
# Assuming 'annual_reports' has columns 'text_content', 'company_abbreviation', and 'year'

# Step 1: Unnest text content into sentences or words
words <- annual_reports %>%
  unnest_tokens(word, text_content)

# Step 2: Get sentiment scores using a sentiment lexicon
sentiment_lexicon <- get_sentiments("afinn")  # bing or AFINN
sentiment_scores <- inner_join(words, sentiment_lexicon, by = "word")

sentiment_scores
```

Sentiment Analysis per Company

```
## # A tibble: 447,154 x 7
##    year  company_abbreviation doc_id   company_name       industry   word  value
##    <chr> <chr>                <chr>    <chr>              <chr>      <chr> <dbl>
##  1 2018  UNH                  UNH_2018 UnitedHealth Group Healthcare unit~     1
##  2 2018  UNH                  UNH_2018 UnitedHealth Group Healthcare no       -1
##  3 2018  UNH                  UNH_2018 UnitedHealth Group Healthcare yes       1
##  4 2018  UNH                  UNH_2018 UnitedHealth Group Healthcare no       -1
##  5 2018  UNH                  UNH_2018 UnitedHealth Group Healthcare yes       1
##  6 2018  UNH                  UNH_2018 UnitedHealth Group Healthcare no       -1
##  7 2018  UNH                  UNH_2018 UnitedHealth Group Healthcare yes       1
##  8 2018  UNH                  UNH_2018 UnitedHealth Group Healthcare no       -1
##  9 2018  UNH                  UNH_2018 UnitedHealth Group Healthcare yes       1
```

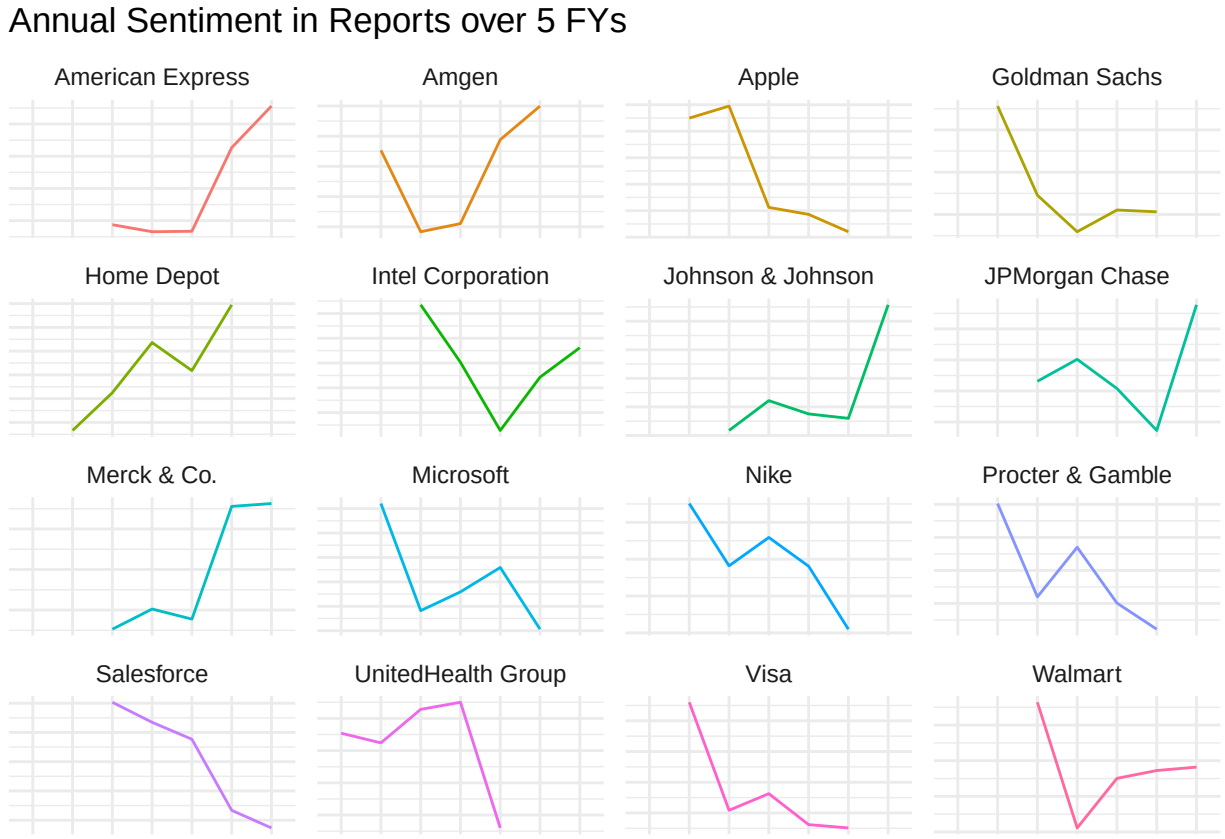
```
## 10 2018  UNH                 UNH_2018 UnitedHealth Group Healthcare no       -1
## # i 447,144 more rows
```

```r
# Step 3: Aggregate sentiment scores by year (and company if needed)
annual_sentiment <- sentiment_scores %>%
  group_by(year, company_name, company_abbreviation) %>%
  summarize(sentiment_score = sum(value), .groups = 'drop')

# Step 4: Create a time-series plot for sentiment over time
ggplot(annual_sentiment, aes(x = year, y = sentiment_score, group = company_name, color = company_name)) +
  geom_line() +
  theme_minimal() +
  labs(title = "Annual Sentiment in Reports", x = "Year", y = "Sentiment Score")
```



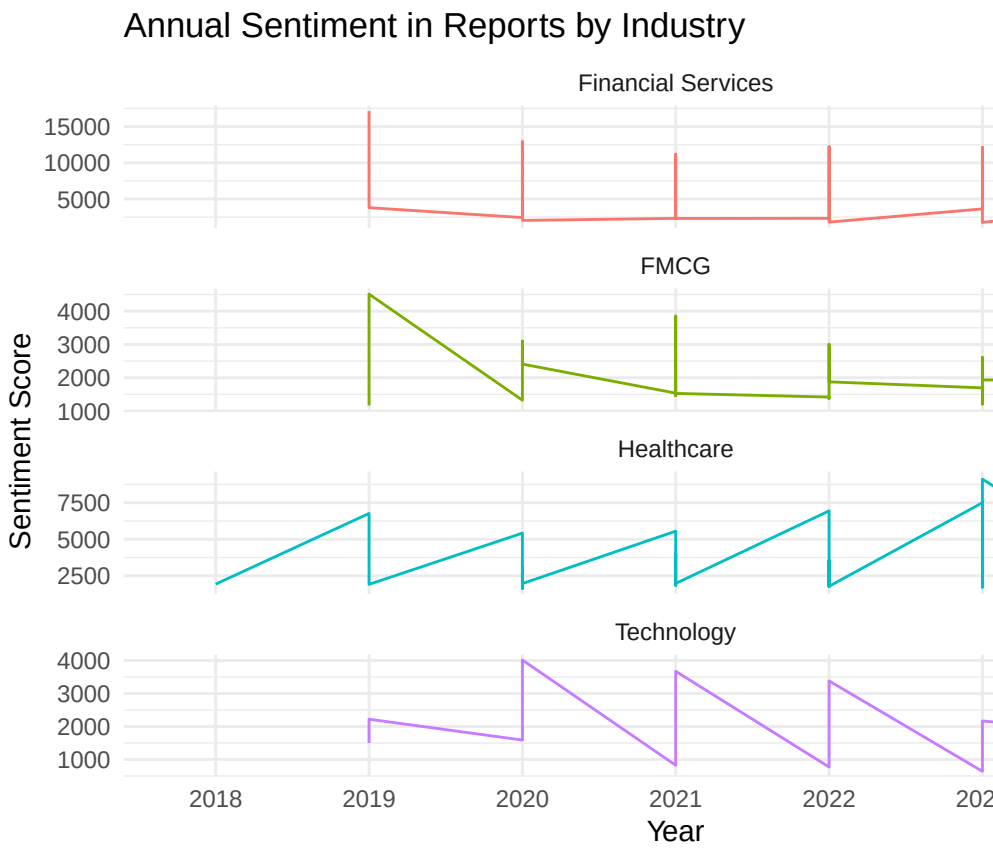

```r
# Alternative plot
ggplot(annual_sentiment, aes(x = year, y = sentiment_score, group = company_name, color = company_name)) +
  geom_line() +
  facet_wrap(~company_name, scales = "free_y", ncol = 4) +  # Adjust the number of columns as needed
  theme_minimal() +
  labs(title = "Annual Sentiment in Reports over 5 FYs", x = "Year", y = "Sentiment Score") +
  theme(#strip.text = element_blank(),  # Remove facet labels
        axis.text = element_blank(),   # Remove axis text
        axis.title = element_blank()
        ) +  # Remove axis titles
  guides(color = FALSE)
```

```
## Warning: The ‘<scale>‘ argument of ‘guides()‘ cannot be ‘FALSE‘. Use "none" instead as
## of ggplot2 3.3.4.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```




```
# This will plot sentiment trends for each company. To see industry-wide trends, group by industry.
```

```
# Aggregate sentiment scores by year and industry
annual_sentiment_by_industry <- sentiment_scores %>%
  group_by(year, industry, company_name, company_abbreviation) %>%
  summarize(sentiment_score = sum(value), .groups = 'drop')

# Create a time-series plot for sentiment over time by industry
ggplot(annual_sentiment_by_industry, aes(x = year, y = sentiment_score, group = industry, color = indus
  geom_line() +
  theme_minimal() +
  labs(title = "Annual Sentiment in Reports by Industry", x = "Year", y = "Sentiment Score") +
  facet_wrap(~ industry, scales = "free_y", ncol = 1) + # Adjust the number of columns as needed
  guides(color = FALSE)
```

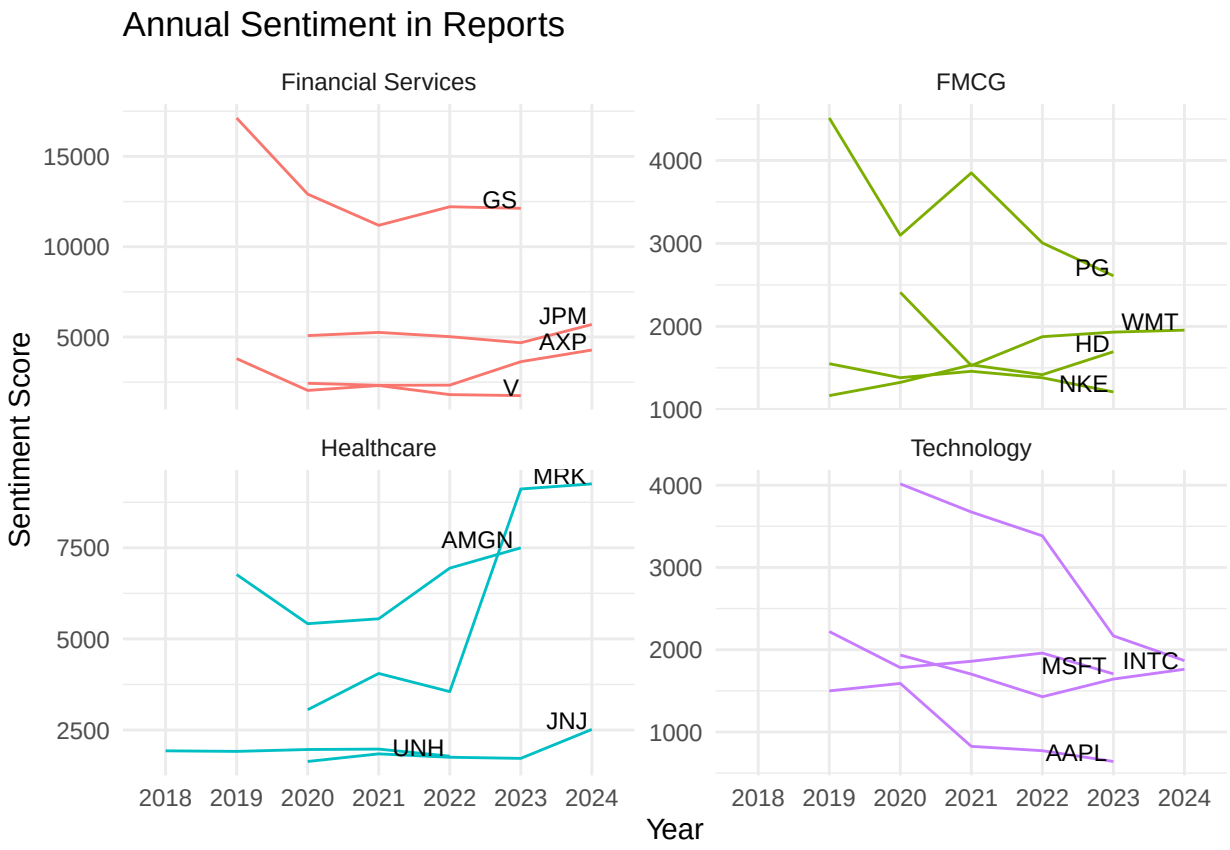


Sentiment Analysis by Industry

Research Question 4:

```
# Assuming you have a sentiment score calculated and 'industry' variable in your dataset

# Step 1: Aggregate sentiment scores by company within each industry
sentiment_by_company_industry <- sentiment_scores %>%
  group_by(industry, company_name) %>%
  summarize(mean_sentiment = mean(value), .groups = 'drop')

# Step 2: Perform sub-group comparisons within each industry
# Note: This can be a visualization or a statistical test depending on the depth of analysis required

# Alternative Plot
ggplot(annual_sentiment_by_industry, aes(x = year, y = sentiment_score, group = company_name, color = in
  geom_line() +
  geom_text(data = annual_sentiment_by_industry %>% group_by(company_name) %>% filter(year == max(year))
            aes(label = company_abbreviation),
            hjust = 1.1, vjust = 0,
            size = 3,
            color = "black",
            check_overlap = TRUE,
            angle = 0) +  # adjust the angle if needed
```

```r
  theme_minimal() +
  labs(title = "Annual Sentiment in Reports", x = "Year", y = "Sentiment Score") +
  facet_wrap(~ industry, scales = "free_y") +
  theme(legend.position = "none")  # This will remove the legend
```



Analysis

```r
# For Statistical Testing (e.g., ANOVA):
# Assuming sentiment_scores also has 'value' as the sentiment score column
anova_results <- sentiment_by_company_industry %>%
  group_by(industry) %>%
  do(tidy(aov(mean_sentiment ~ company_name, data = .)))

anova_results
```

```
## # A tibble: 4 x 5
## # Groups:   industry [4]
##   industry           term            df  sumsq meansq
##   <chr>              <chr>        <dbl>  <dbl>  <dbl>
## 1 FMCG               company_name     3 0.0598 0.0199
## 2 Financial Services company_name     3 0.0333 0.0111
## 3 Healthcare         company_name     3 0.0635 0.0212
## 4 Technology         company_name     3 0.0353 0.0118
```

```r
# Step 3: Interpret the results
# Depending on the outcome of the visualization or statistical test, draw conclusions about
# the different strategic postures or conditions of companies within the same industry.
```