Table of Contents

Text Analytics Individual Assignment

Student ID: 2095515

R Packages

```
knitr::opts_chunk$set(echo = TRUE)
rm(list=ls())
library(knitr)
library(dplyr)
library(stringr)
library(rbenchmark)
library(tidyverse)
# Download Forms 10-K/10 Q from SEC
library(edgar)
# For sentiment datasets
library(data.table)
library(parallel)
library(tidytext)
library(textstem)
library(rvest)
library(ggplot2)
library(tidyr)
# part B
library(tm)
library(SentimentAnalysis)
# part C
library(koRpus)
library(stm)
```

Part A

Loading the dataset

We chose 30 companies which have similar GICS sub industry classifications this would make for easier comparisons between companies.

We previously considered choosing top 30 companies by their market capitalisation. We assumed that these companies would have more public scrutiny. Therefore, they are more conscious over management discussion. However, they had more missing 10-K documents and yield to such a really low TF-IDF values. Thus, we decided to proceed with sub-industries with the most companies in to make an easier comparison. Here we have also identified a lot less missing 10-K reports to support our analysis. (Initially we had 112 missing 10-K, here we only missed 60 10-K reports)

```r
# importing the dataset

# since we are provided with 71 companies to choose from the appendix
# we prepared the data in Excel and sorted them according to their GICS sub i
ndustry
cik_table <- readxl::read_xlsx("companies31.xlsx")

# Name the columns
colnames(cik_table) <- c("Symbol", "Security", "GICS_Sector", "GICS_Sub_Indus
try", "CIK")

# Format column types
cik_table$Symbol <- as.character(cik_table$Symbol)
cik_table$Security <- as.character(cik_table$Security)
cik_table$GICS_Sector <- as.character(cik_table$GICS_Sector)
cik_table$GICS_Sub_Industry <- as.character(cik_table$GICS_Sub_Industry)
cik_table$CIK <- as.integer(cik_table$CIK)

# Check and remove the duplicated CIK as it belong to the same company
# (cross-checked from Master Indexes)
cik_table %>% group_by(CIK) %>% summarise(total=n()) %>% arrange(desc(total))
```

```
## # A tibble: 30 x 2
##       CIK total
##     <int> <int>
##  1   2488     1
##  2   4127     1
##  3   6281     1
##  4   6951     1
##  5  50863     1
##  6  97476     1
##  7 319201     1
##  8 707549     1
##  9 723125     1
## 10 743316     1
## # ... with 20 more rows
```

```r
# assumed that companies with same cik
# are treated as one single unit
# as edgar filings are made per CIK
# delete duplicates
cik_table <- cik_table[!duplicated(cik_table$CIK),]

# extract all cik and length of it
#cik <- cik_table$CIK
```

Get MDA reports

This analysis is conducted only on 10-K reports. In a case where there are missing 10-K reports, we will just keep them as the way they are and not replace them with any

corresponding 10-Q forms or any other supporting materials. Making our analysis limited to in-depth analysis of the available 10-K reports.

```
# to get parts of 10-K reports for the 30 companies above
f <- edgar::getMgmtDisc(cik.no = cik_table$CIK, filing.year = c(2010:2020))
```

Loading the master key index

```
master_index <- data.frame()
for (i in 2010:2020){
  file_name <- paste0("Master Indexes/", i, "master.Rda")
  load(file_name)
  index_h <- year.master %>%
    filter(cik %in% c(cik_table$CIK), form.type %in% c("10-K", "10-Q"))

  master_index <- rbind(master_index, index_h)
}

# Format column type
master_index$cik <- as.character(master_index$cik) %>% as.integer()
master_index$company.name <- as.character(master_index$company.name)
master_index$form.type <- as.character(master_index$form.type)
master_index$date.filed <- as.Date(master_index$date.filed)
master_index$edgar.link <- as.character(master_index$edgar.link)
master_index <- master_index %>%
  mutate(year = format(as.Date(master_index$date.filed, "%Y, %m, %d"), "%Y"))
master_index$year <- as.integer(master_index$year)

saveRDS(master_index, "master_index.rds")

# Create a master index data frame for 10-K (annual) form type
master_index_10K <- master_index %>%
  filter(master_index$form.type == "10-K")

# Create a master index data frame for 10-Q (annual) form type
master_index_10Q <- master_index %>%
  filter(master_index$form.type == "10-Q")

# Check if there any duplicated form grouped by CIK and Filing Year for 10-K
report
master_index_10K %>% group_by(cik, company.name, year) %>% summarise(total =
n()) %>% arrange(desc(total))

#searching for missing years in 10-K
master_index_10K_NA<- master_index_10K %>%
  select(cik, company.name, year) %>%
  group_by(cik) %>%
  complete(year = 2010:2020)%>%
  filter(is.na(company.name)) #filter for NA in 10-k missing years
```

```
master_index_10K_NA
#here we have 60 missing 10-K reports throughout the period

# loading the master index
master_index <- readRDS("master_index.rds")
rm(master_index_10Q, master_index_10K, master_index_10K_NA)

# to get the full 10-Q report from the 30 companies
g <- edgar::getFilingsHTML(cik.no = cik_table$CIK, form.type = c("10-K", "10-
Q"), filing.year = c(2010:2020))
```

Casting MDA files into a dataframe

```
# Combining management discussions into a dataframe
mgmt_disc_df <- data.frame()

file_list <- list.files("MD&A section Text")
file_path <- paste0("MD&A section Text/",
                    file_list)

for (i in 1:length(file_path)) {
  file <- readLines(file_path[i])

  mgmt_disc <- tibble(cik = as.integer(gsub("CIK:","",file[1])),
                      company_name = tolower(gsub("Company Name:","", file[2]
)),
                      date_filed = gsub("Filing Date:","",file[4]),
                      accession_no = gsub("Accession Number:","", file[5]),
                      #cleaning the textual (remove non-Latin)
                      #words, digits and punctuation
                      mgmt_disc = gsub(" s ", " ", tolower(file[8])) %>%
                        tm::removePunctuation() %>%
                        tm::removeNumbers()%>%
                        tm::stripWhitespace())
  mgmt_disc_df <- bind_rows(mgmt_disc_df, mgmt_disc)
}
# Rename columns for Management Discussions Table
colnames(mgmt_disc_df) <- c("CIK", "Company_Name", "Date_Filed", "Accession_N
umber", "Text")

# Format column type
mgmt_disc_df$CIK <- as.character(mgmt_disc_df$CIK) %>% as.integer()
mgmt_disc_df$Company_Name <- as.character(mgmt_disc_df$Company_Name)
mgmt_disc_df$Date_Filed <- as.Date(mgmt_disc_df$Date_Filed)

# Add in 'Year' column
mgmt_disc_df <- mgmt_disc_df %>% mutate(Year = format(as.Date(mgmt_disc_df$Da
te_Filed, "%Y, %m, %d"), "%Y"))
mgmt_disc_df$Year <- as.integer(mgmt_disc_df$Year)
```

```
# Add in Quarter column
mgmt_disc_df <- mgmt_disc_df %>% mutate(Quarter = substr(quarters(as.Date(Dat
e_Filed)), 2, 2))
mgmt_disc_df$Quarter <- as.integer(mgmt_disc_df$Quarter)

# Saving dataframe for efficiency
saveRDS(mgmt_disc_df, "mgmt_disc_df.rds")

# Merge with Master Index Table
merged_mda <- mgmt_disc_df %>%
  unique() %>%
  inner_join(cik_table, by=c("CIK"="CIK"))

merged_mda <- merged_mda %>%
  select(CIK, Symbol, Security, Company_Name, GICS_Sub_Industry, Date_Filed,
Quarter, Year, Accession_Number, Text)


saveRDS(merged_mda, "merged_mda.rds")

# Loading the RDS file
mgmt_disc_df <- readRDS("mgmt_disc_df.rds")
merged_mda <- readRDS("merged_mda.rds")
```

Text cleaning and tokenization

Once the reports are merged into a dataframe and saved in rds format, we are now ready to do some necessary text cleaning to ensure the text are ready for analysis.

```
# Importing stopwords from LoughranMcDonald's finance-specific dictionary
library(readr)

##
## Attaching package: 'readr'

## The following object is masked from 'package:rvest':
##
##     guess_encoding

## The following object is masked from 'package:koRpus':
##
##     tokenize

stop_words_LM <- c()
stop_words_dict <- list.files('stop_words_LM')

for (i in 1: length(stop_words_dict)){
  file_path_stop_words_LM <- paste('stop_words_LM', stop_words_dict[i], sep="
/")
  local_list <-read_lines(file_path_stop_words_LM)
  #  remove any symbols in text (/n,/r) and set it to lowercase
```

```
  local_list <- iconv(local_list, "ASCII", "UTF-8", sub="") %>% tolower()
  stop_words_LM <- c(stop_words_LM, local_list)
}

# Add customised stopwords
stopw_custom <- c("financial", "statement", "exhibit", "report", "figure", "f
ig", "table", "company", "footnote", "page")

# Finalising stopwords
stopw_final <- c(stop_words_LM, stopw_custom)
#rm(stop_words_LM, stopw_custom, stop_words_dict, file_path_stop_words_LM, lo
cal_list)

# loading the stop words data
library(lexicon)
library(httr)
library(textclean)
library(tidytext)

data(stop_words) # multilingual stop words list
data("sw_fry_1000") #Fry's 1000 most commonly used English words

# Create a function to automatically remove digit,
# remove punctuations and transform all letters to lower
stopword_funct <- function(x){
  x <- gsub('[[:digit:]]+',' ', x)
  x <- gsub('[[:punct:]]+',' ', x)
  x <- tolower(x)
  return(x)
}

# Symbol as stopwords
merged_mda$Symbol <- stopword_funct(merged_mda$Symbol)

Symbol <- merged_mda %>%
  select(Symbol) %>%
  unique(.)

symbol_stopwords <- merged_mda$Symbol


# Company name as stopwords
merged_mda$Company_Name <- stopword_funct(merged_mda$Company_Name)

company_name <- merged_mda %>%
  select(Company_Name) %>%
  unnest_tokens(word, Company_Name) %>%
  select(word) %>%
  unique(.)
```

```r
company_name_stopwords <- company_name$word

# GICS Sub Industry as stopwords
merged_mda$GICS_Sub_Industry <- stopword_funct(merged_mda$GICS_Sub_Industry)

subindustry <- merged_mda %>%
  select(GICS_Sub_Industry) %>%
  unnest_tokens(word, GICS_Sub_Industry) %>%
  select(word) %>%
  unique(.)

subindustry_stopwords <- subindustry$word

custom_stopwords <- tibble(word = unique(c(company_name_stopwords, subindustr
y_stopwords, symbol_stopwords)), lexicon = "custom")

# Create custom stop words list
custom_stopwords <- rbind(stop_words, custom_stopwords)

#tokenisation, remove custom stopwords and LM's dictionary list
tokens_lists <- merged_mda %>%
  unnest_tokens(word, Text) %>%
  anti_join(custom_stopwords) %>%
  filter(!(word %in% sw_fry_1000)) %>%
  filter(!(word %in% stopw_final))

## Joining, by = "word"

format(length(tokens_lists$Accession_Number), big.mark = ",")

## [1] "802,324"

head(tokens_lists,10)

##        CIK Symbol Security Company_Name    GICS_Sub_Industry
## 1  1013462   anss    ANSYS    ansys inc application software
## 2  1013462   anss    ANSYS    ansys inc application software
## 3  1013462   anss    ANSYS    ansys inc application software
## 4  1013462   anss    ANSYS    ansys inc application software
## 5  1013462   anss    ANSYS    ansys inc application software
## 6  1013462   anss    ANSYS    ansys inc application software
## 7  1013462   anss    ANSYS    ansys inc application software
## 8  1013462   anss    ANSYS    ansys inc application software
## 9  1013462   anss    ANSYS    ansys inc application software
## 10 1013462   anss    ANSYS    ansys inc application software
##                GICS_Sector Date_Filed Quarter Year      Accession_Number
## 1  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 2  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 3  Information Technology 2010-02-25       1 2010  0001193125-10-040617
```

```
## 4  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 5  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 6  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 7  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 8  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 9  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 10 Information Technology 2010-02-25       1 2010  0001193125-10-040617
##          word
## 1        item
## 2  management
## 3  discussion
## 4    analysis
## 5     results
## 6  operations
## 7    overview
## 8     results
## 9    december
## 10    reflect
```

```
saveRDS(tokens_lists,"tokens_lists.rds")
#802324

tokens_lists <- readRDS("tokens_lists.rds")
#rm(stopw_custom, stop_words_dict, stop_words_LM, stopword_funct, Symbol, com
pany_name, company_name_stopwords, subindustry, subindustry_stopwords)
```

Stemming and Lemmatisation

Download language model and define cores for parallel processing

```
# Download English model from udpipe
langmodel_download <- udpipe::udpipe_download_model("english")

# Load model
langmodel <- udpipe::udpipe_load_model(langmodel_download$file_model)

# Specify the number of cores
no_cores <- parallel::detectCores() - 1

# Run udpipe on tokenised management discussions
postagged_mda <- udpipe:: udpipe(object = langmodel, x = tokens_lists$word, p
arallel.cores = no_cores,
                    parallel.chunks = 100,
                    trace = T)

postagged_mda_df <- as.data.frame(postagged_mda)

# Initiate cluster
cl <- parallel::makeCluster(no_cores)
```

```
# Stop cluster
parallel::stopCluster(cl)
saveRDS(postagged_mda_df, "postagged_tokens_mda.rds")
```

Joining the lemmatized udpipe output of management discussions with the other variables in the data frame

```
tokens_lists <- tokens_lists %>%
  mutate(doc_id = row_number())

length(unique(tokens_lists$doc_id))

## [1] 802324

length(unique(postagged_mda_df$doc_id))

## [1] 802324

postagged_mda_df_joined <- postagged_mda_df  %>%
  mutate(doc_id = as.numeric(doc_id)) %>%
  left_join(tokens_lists)

## Joining, by = "doc_id"
```

Removing short and long words from listings output of udpipe

```
postagged_mda_df_joined$word_length <- nchar(postagged_mda_df_joined$lemma)

# defining cut-off values for filtration
quantile(postagged_mda_df_joined$word_length, c(0.025, 0.975))

##  2.5% 97.5%
##     3    12

# 2.5% 97.5%
#    3    12

# filtering out words that have less than 3 characters or words that have gre
ater than 12 characters
postagged_mda_df_joined <- postagged_mda_df_joined %>%
  arrange(word_length) %>%
  filter(between(word_length, 3, 12))

length(postagged_mda_df_joined$lemma)

## [1] 778955
```

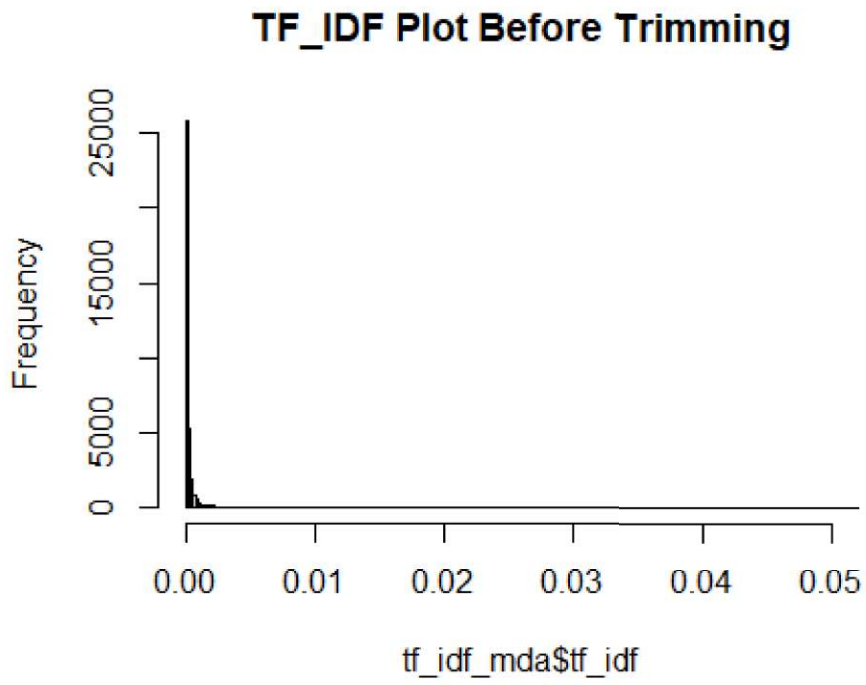
Removing Stopwords again after lemmatisation on management discussion documents

```
#removing stopwords again after lemmatisation
postagged_mda_df_joined_short <- postagged_mda_df_joined %>%
```

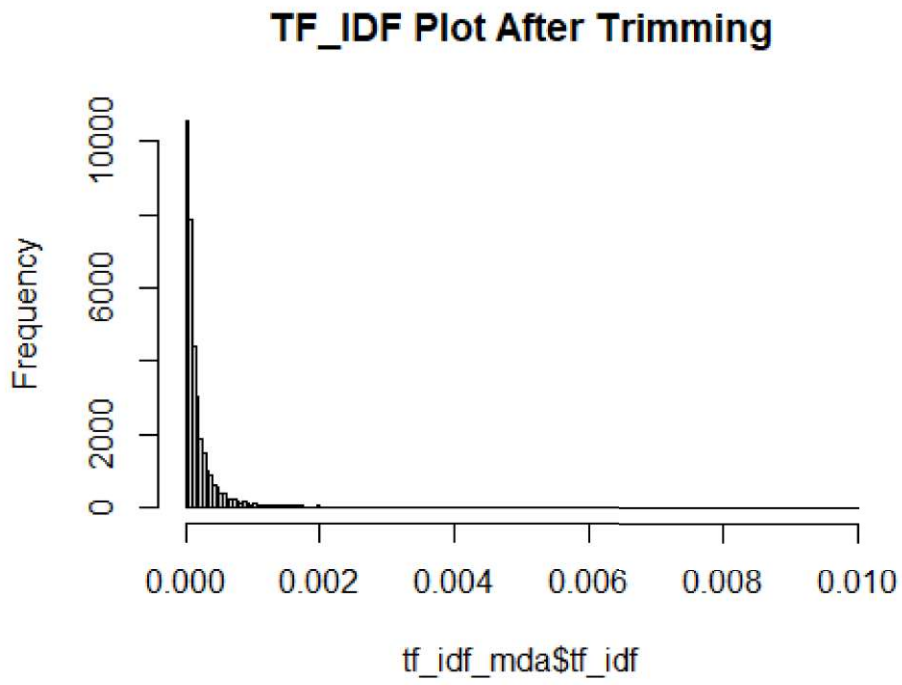
```
  mutate(word = lemma) %>%
  anti_join(custom_stopwords) %>%
  filter(!(word %in% stopw_final))

## Joining, by = "word"

length(postagged_mda_df_joined_short$lemma)

## [1] 765671

#calculate difference
format((length(postagged_mda_df_joined$lemma) -length(postagged_mda_df_joined
_short$lemma)), big.mark = ",")

## [1] "13,284"

saveRDS(postagged_mda_df_joined_short, "postagged_mda_df_joined_short.rds")

clean_mda <- readRDS("postagged_mda_df_joined_short.rds")
#rm(postagged_mda, postagged_mda_df)
```

Performing TF-IDF

```
#calculate word counts
tokens_count_all <- clean_mda %>%
  count(lemma, sort = TRUE)

# TF-IDF on management discussions
# arranging post-tagged words in descending order by tf-idf
tf_idf_mda <- clean_mda  %>%
  group_by(CIK) %>%
  count(lemma, sort = TRUE) %>%
  bind_tf_idf(lemma, CIK, n) %>%
  arrange(desc(tf_idf))

# Visualizing in a histogram
hist(tf_idf_mda$tf_idf, breaks = 200, main = "TF_IDF Plot Before Trimming")
```




```r
# From the plot we could see that the right cut-off value is at 0.01
tf_idf_mda <- tf_idf_mda %>%
              filter(tf_idf<0.01)

# Visualizing in a histogram
hist(tf_idf_mda$tf_idf, breaks = 200, main = "TF_IDF Plot After Trimming")
```




```
# Determining cut-off values for TF-IDF
# checking for the upper & lower quantiles of tf-idf
quantile(tf_idf_mda$tf_idf, na.rm = T, c(0.025, 0.975))
```

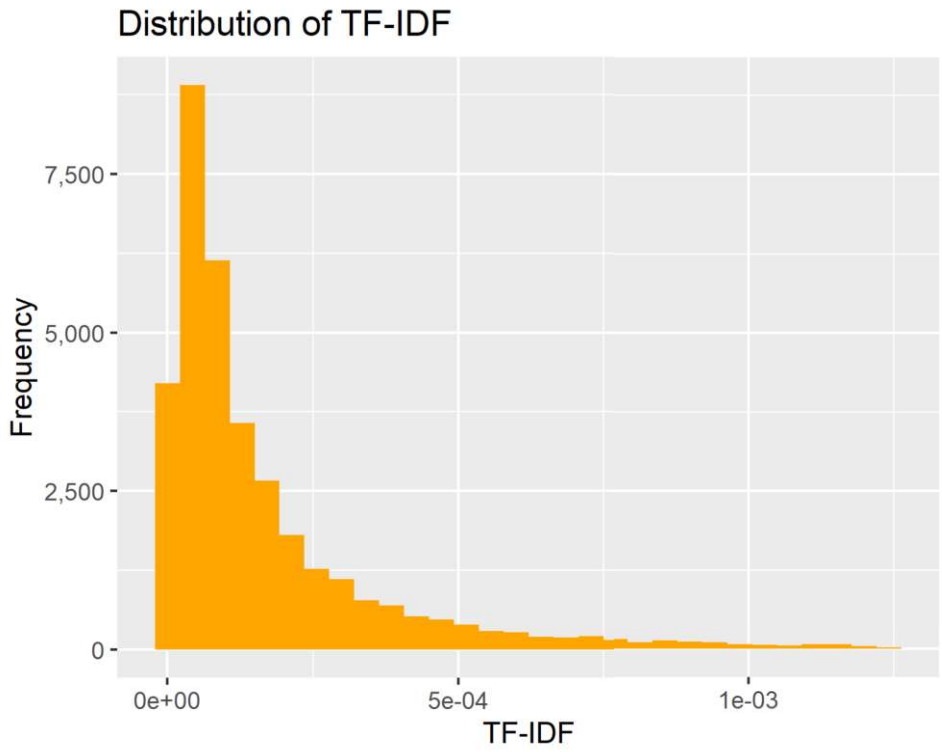
```
##        2.5%       97.5%
## 0.000000000 0.001242894
```

```
#2.5%       97.5%
#0.000000000 0.001242894
```

```
# Imposing further TF-IDF's upper and lower quantiles and plot them in a hist
ogram
tf_idf_mda <- tf_idf_mda %>%
              filter(between(tf_idf, 0.000000000, 0.001242894)) %>%
              arrange(desc(tf_idf))


tf_idf_mda_plot <- ggplot(tf_idf_mda, aes(x=tf_idf, fill="blue")) +
              geom_histogram() +
              labs(title="Distribution of TF-IDF", x="TF-IDF", y="Frequency")
+
              scale_y_continuous(labels=scales::comma)+ scale_fill_manual(val
ues = c("orange")) +
              theme(legend.position="none")


ggsave(tf_idf_mda_plot, filename = "tf_idf_mda_plot.png")
```

```
## Saving 5 x 4 in image

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

rm(tf_idf_mda_plot)
```

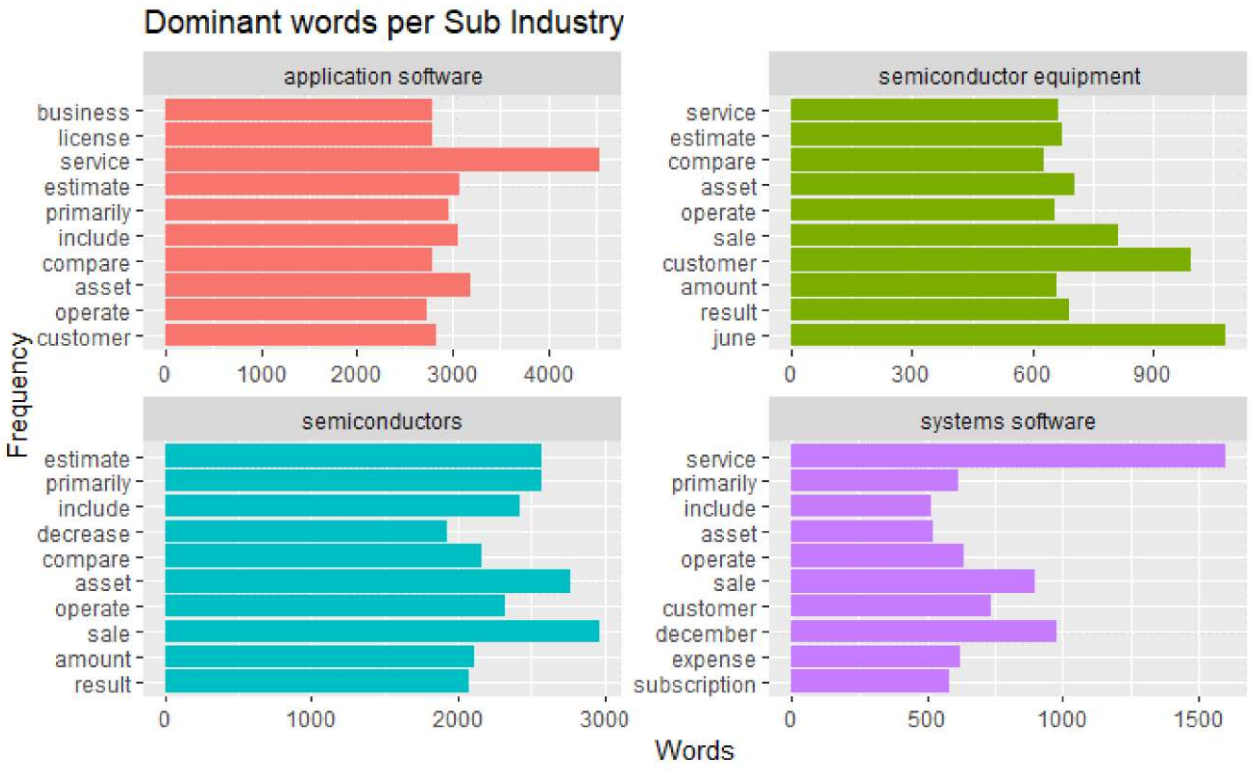
TF-IDF of the words follow a Zipf's law distribution and here we used a set of cut-off values using quantiles of 2.5% and 97.5% for the lower and upper cut-offs respectively. Here we could see after imposing the cut-off values, we have:



Dominant words by sub-industry

```
library(ggplot2)
library(forcats)

# adding custom stop words
# filtering out words that will not help in the analysis
custom_stopwords1 <- c("tax","revenue","increase","cash","income","net","rela
te","due","expensis")

# cleaning tokens further by removing additional stop words
tf_idf_mda <- tf_idf_mda %>%
              filter(!(lemma %in% custom_stopwords1))


gcis_sub <- tf_idf_mda %>%
  group_by(GICS_Sub_Industry) %>%
```

```
  count(lemma, GICS_Sub_Industry) %>%
   arrange(desc(n)) %>%
  top_n(10,n) %>% ungroup() %>%
  mutate(word2=fct_reorder(lemma, n))

ggplot(gcis_sub, aes(reorder(word2,n), n, fill=GICS_Sub_Industry)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~GICS_Sub_Industry, scales="free") +
  coord_flip() + labs(title="Dominant words per Sub Industry", x="Frequency",
y="Words")
```



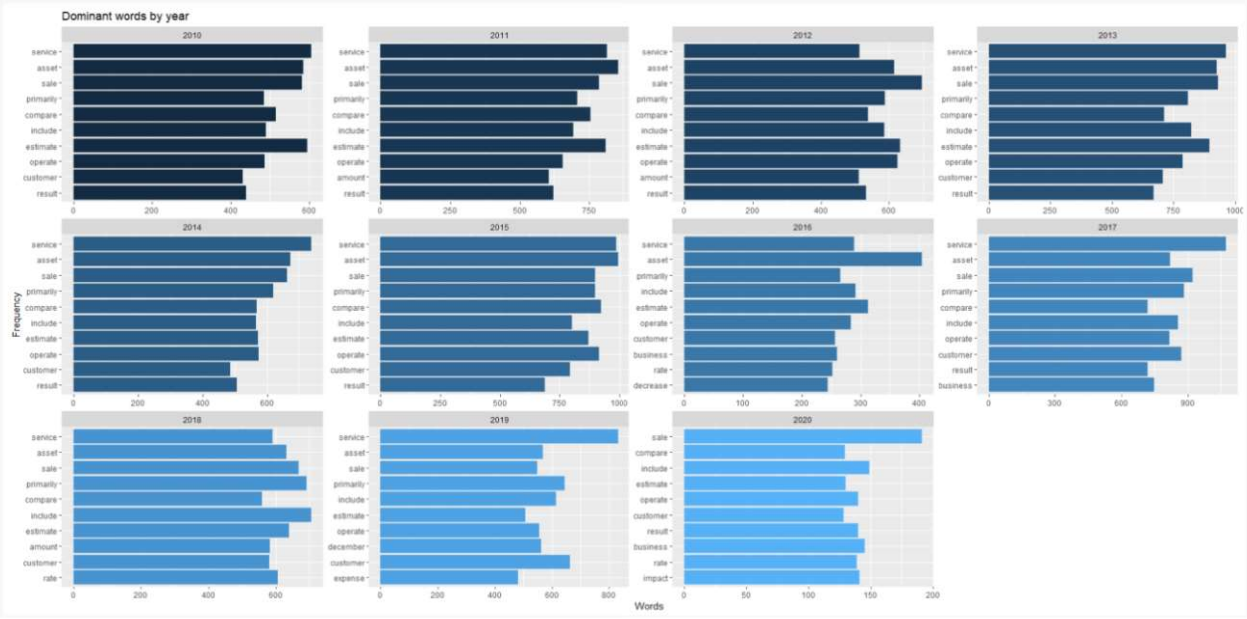
On sub-industry level we could see that TF-IDF could capture the most dominant words in each corresponding sub-industry. Take for an example in both system software and application software industry the word 'service' dominates both sub-industries. The reason being the nature of their products which solely based on service.

Whereas in both semiconductors and semiconductor equipment sub-industries, they have words associated with selling consumer goods, namely 'sale' and 'customer'.

Dominant words by year

```
library(ggplot2)
library(forcats)

# adding custom stop words
```

```r
# filtering out words that will not help in the analysis
custom_stopwords1 <- c("fiscal","tax","revenue","increase","cash","income","n
et","relate","due","expensis")

# cleaning tokens further by removing additional stop words
tf_idf_mda <- tf_idf_mda %>%
              filter(!(lemma %in% custom_stopwords1))


gcis_sub <- tf_idf_mda %>%
  group_by(Year) %>%
  count(lemma, Year) %>%
   arrange(desc(n)) %>%
  top_n(10,n) %>% ungroup() %>%
  mutate(word2=fct_reorder(lemma, n))

ggplot(gcis_sub, aes(reorder(word2,n), n, fill=Year)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~Year, scales="free") +
  coord_flip() + labs(title="Dominant words by year", x="Frequency", y="Words
")
```



Whilst by plotting TF-IDF analysis across the span of years of analysis (2010-2020), they are quite consistent throughout the years, which mainly revolves services & consumer goods. This could be because of the nature of sub-industries chosen in the scope of this analysis. In the future, analysts could also consider several different sub-industries to check the validity of the analysis.

Conclusions

Part A has guided this study to build a corpus of Management Discussion & Analysis 10-K reports. Insightful information relating to sub-industry level trends had been reflected by using TF-IDF analysis that gives greater weightage to more important words.

Part B

This section aims to quantify the textual information in a report. Sentiment analysis can draw out the subjective information in a text, making it a good measure to quantify financial reports. The underlying assumption is that the financial market will react to how good or bad the financial report describes the state of a company. In a perfect world, it would imply that a bad sentiment would reflect to a drop in stock prices and vice versa.

Analyse sentiment with multiple dictionaries

```
library(readxl)
# Importing Loughran % McDonald's Sentiment
LM_dictionary <- read_xlsx("LoughranMcDonald_SentimentWordLists_2018 copy.xls
x")
# Name the columns
colnames(LM_dictionary) <- c("word", "sentiment")
LM_dictionary$word <-  tolower(LM_dictionary$word)
LM_dictionary$sentiment <- tolower(LM_dictionary$sentiment)

#Creating dummy dictionaries
dummy_LM <- tibble(accession_number = 'dummy', positive=0, negative=0, litigi
ous=0, uncertainty=0, constraining=0 )
dummy_nrc <- tibble(accession_number = 'dummy', positive=0, negative=0, anger
=0, fear=0, trust=0, sadness=0, suprise=0, disgust=0, joy=0, anticipation=0 )
dummy_bing <- tibble(accession_number = 'dummy', positive=0, negative=0 )
```

Calculating sentiment measures and casting them to sentiment table

Sentiments are calculated using different dictionaries including Syuzhet, Vader, Loughran&McDonald, NRC, Sentimentr, Bing and Afinn.

```
library(textfeatures)

merged_mda <- readRDS("merged_mda.rds")

# Sentiment Syuzhet, Vader and n_words
#sentiment_syuzhet_vader <- textfeatures(merged_mda$Text, normalize = FALSE,
word_dims = FALSE, sentiment = TRUE) %>%
#  select(sent_syuzhet, sent_vader) %>%
#  mutate(Accession_Number = merged_mda$Accession_Number)

# Text complexity
tokenised <- merged_mda %>%
```

```
  unnest_tokens(word, Text)

#n_words <- tokenised %>%
#  group_by(Accession_Number) %>%
#  count(Accession_Number)
#n_complex <- tokenised %>%
#  group_by(word, Accession_Number) %>%
#  mutate(complexity = nchar(gsub("[^X]", "", gsub("[aeiouy]+", "x", tolower(
word))))) #%>%
  #filter(complexity >= 3) %>%
  #group_by(Accession_Number) %>%
  #count(Accession_Number)

#complexity <- tibble(Accession_Number = n_words$Accession_Number, complexity
= n_complex$n / n_words$n)

# We decided to skip Syuzhet and Vader dictionaries because their complexity
yields to 0 for this specific dataframe.

# Sentiment Loughran&McDonald
tokens_LM <- tokenised %>%
  inner_join(LM_dictionary, by=c("word"="word"))

head(tokens_LM, 10)
```

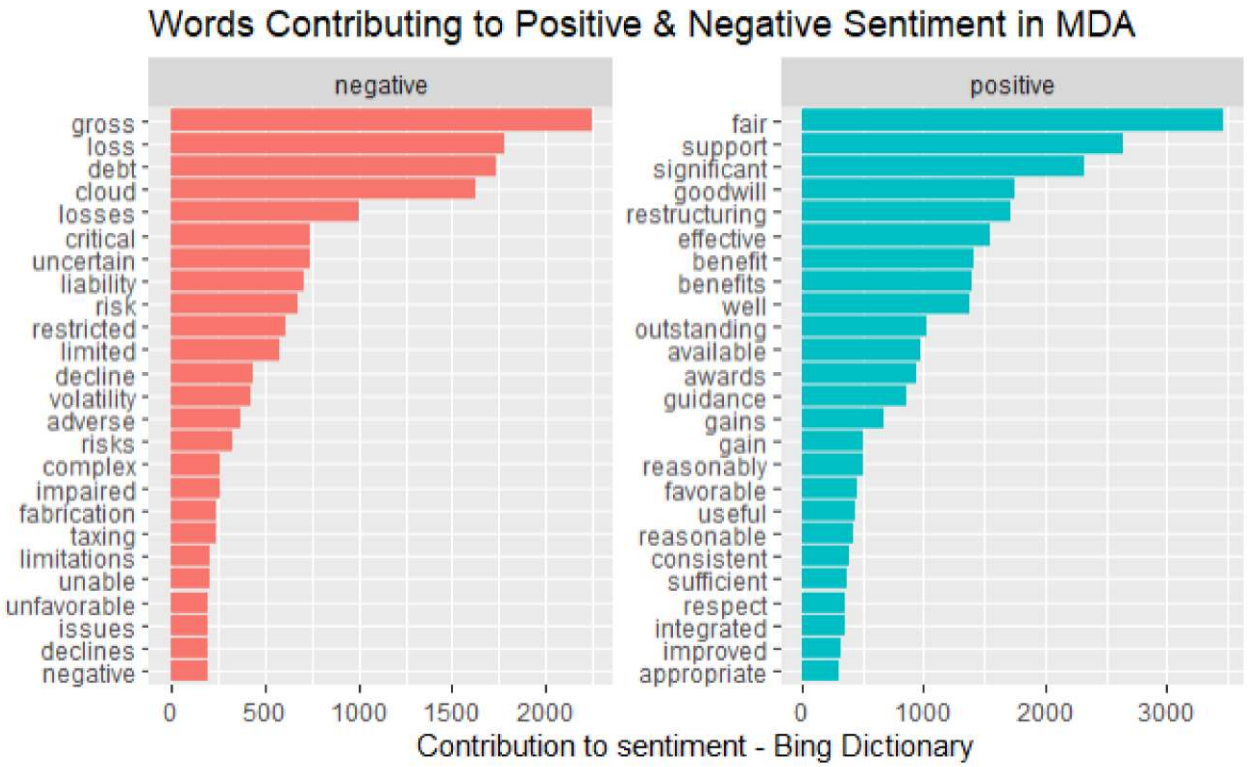
```
##        CIK Symbol Security Company_Name    GICS_Sub_Industry
## 1  1013462   ANSS    ANSYS    ansys inc Application Software
## 2  1013462   ANSS    ANSYS    ansys inc Application Software
## 3  1013462   ANSS    ANSYS    ansys inc Application Software
## 4  1013462   ANSS    ANSYS    ansys inc Application Software
## 5  1013462   ANSS    ANSYS    ansys inc Application Software
## 6  1013462   ANSS    ANSYS    ansys inc Application Software
## 7  1013462   ANSS    ANSYS    ansys inc Application Software
## 8  1013462   ANSS    ANSYS    ansys inc Application Software
## 9  1013462   ANSS    ANSYS    ansys inc Application Software
## 10 1013462   ANSS    ANSYS    ansys inc Application Software
##                GICS_Sector Date_Filed Quarter Year      Accession_Number
## 1  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 2  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 3  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 4  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 5  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 6  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 7  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 8  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 9  Information Technology 2010-02-25       1 2010  0001193125-10-040617
## 10 Information Technology 2010-02-25       1 2010  0001193125-10-040617
##                word   sentiment
## 1         hereafter   litigious
## 2           decline    negative
```

```
## 3           adverse    negative
## 4      fluctuations uncertainty
## 5        volatility    negative
## 6        volatility uncertainty
## 7       unfavorable    negative
## 8         adversely    negative
## 9  underperformance    negative
## 10          against    negative
```

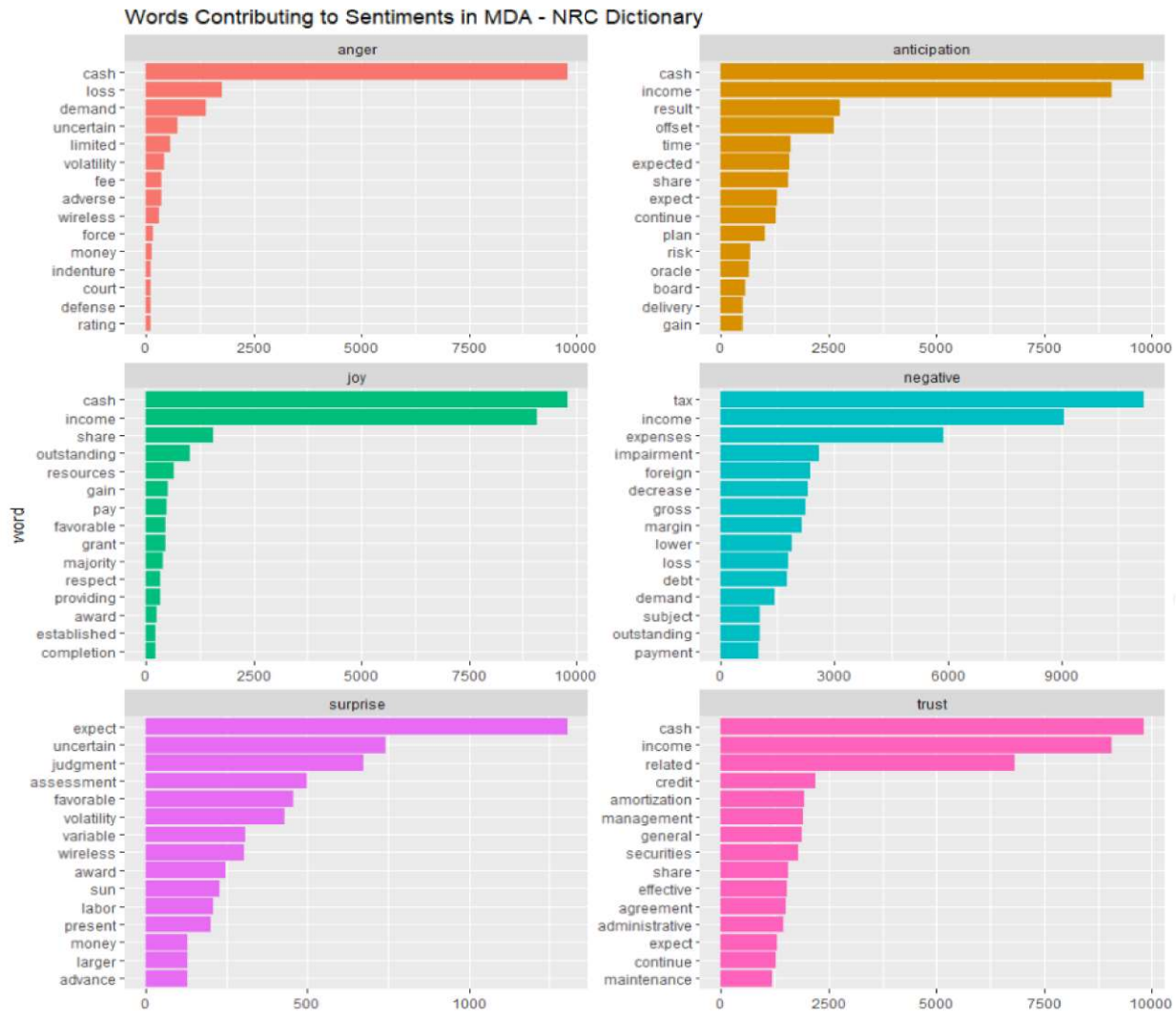
```
word_count_LM <- tokens_LM %>%
  group_by(Accession_Number) %>%
  summarise(LM_total_words=n())

sentiment_LM <- tokens_LM %>%
  group_by(Accession_Number, sentiment) %>%
  summarise(total_sentiment = n()) %>%
  spread(sentiment, total_sentiment, fill=0) %>%
  bind_rows(dummy_LM) %>%
  left_join(word_count_LM) %>%
  mutate(LM_sent = positive-negative,
         LM_positive = positive/LM_total_words,
         LM_negative = negative/LM_total_words,
         LM_litigious = litigious/LM_total_words,
         LM_uncertainty = uncertainty/LM_total_words,
         LM_constraining = constraining/LM_total_words ) %>%
  select(-c(positive, negative, litigious, uncertainty, constraining)) %>%
  filter(Accession_Number !="dummy")

## `summarise()` has grouped output by 'Accession_Number'. You can override u
sing the `.groups` argument.

## Joining, by = "Accession_Number"

# Sentiment Afinn
tokens_afinn <- tokenised %>%
  inner_join(get_sentiments("afinn"))

## Joining, by = "word"

sentiment_afinn <- tokenised %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(Accession_Number) %>%
  summarise(afinn_sent =sum(value))

## Joining, by = "word"

rm(tokens_afinn)

# Sentiment Bing
tokens_bing <- tokenised %>%
  inner_join((get_sentiments("bing")))
```

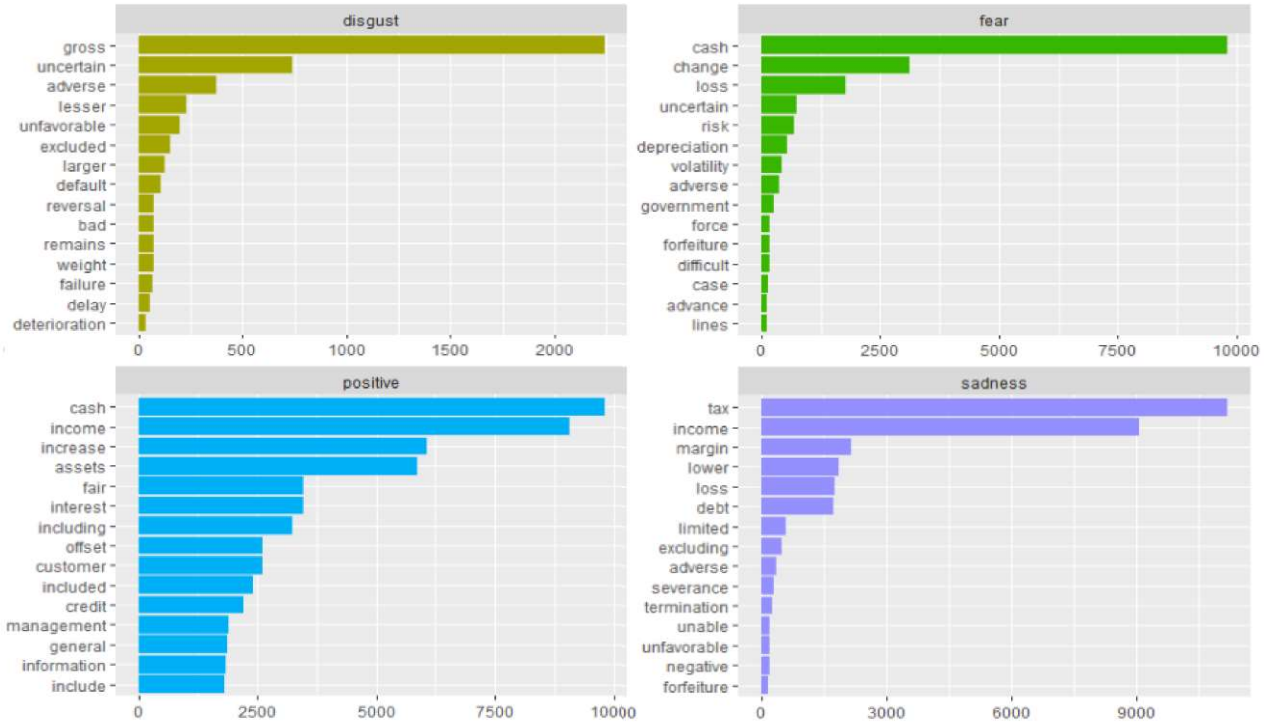
```
## Joining, by = "word"

word_count_bing <- tokens_bing %>%
  group_by(Accession_Number) %>%
  summarise(bing_total_words = n())

sentiment_bing <- tokens_bing %>%
  group_by(Accession_Number, sentiment) %>%
  summarise(total_sentiment=n()) %>%
  spread(sentiment, total_sentiment, fill=0) %>%
  bind_rows(dummy_bing) %>%
  left_join(word_count_bing) %>%
  mutate(bing_sent=positive-negative,
  bing_positive = positive/bing_total_words,
  bing_negative = negative/bing_total_words) %>%
  select(-c(positive, negative)) %>%
  filter(Accession_Number !='dummy')

## `summarise()` has grouped output by 'Accession_Number'. You can override u
sing the `.groups` argument.
## Joining, by = "Accession_Number"

rm(tokens_bing, word_count_bing)

# Sentiment NRC
tokens_nrc <- tokenised %>%
  inner_join((get_sentiments("nrc")))

## Joining, by = "word"

word_count_nrc <- tokens_nrc %>%
  group_by(Accession_Number) %>%
  summarise(nrc_total_words = n())

sentiment_nrc <- tokens_nrc %>%
  group_by(Accession_Number, sentiment) %>%
  summarise(total_sentiment=n()) %>%
  spread(sentiment, total_sentiment, fill=0) %>%
  bind_rows(dummy_nrc) %>%
  left_join(word_count_nrc) %>%
  mutate(nrc_sent=positive-negative,
         nrc_positive = positive/nrc_total_words,
         nrc_negative = negative/nrc_total_words,
         nrc_anger    = anger/nrc_total_words,
         nrc_fear     = fear/nrc_total_words,
         nrc_trust    = trust/nrc_total_words,
         nrc_sadness  = sadness/nrc_total_words,
         nrc_surprise = surprise/nrc_total_words,
         nrc_disgust  = disgust/nrc_total_words,
         nrc_joy      = joy/nrc_total_words,
         nrc_anticipation = anticipation/nrc_total_words) %>%
```

```
        select(-c(positive, negative, anger, trust, sadness, surprise, disgus
t, joy, anticipation, fear)) %>%
  filter(Accession_Number !='dummy')

## `summarise()` has grouped output by 'Accession_Number'. You can override u
sing the `.groups` argument.
## Joining, by = "Accession_Number"

rm(tokens_nrc, word_count_nrc)

# Merging sentiment features
sentiment_df <- sentiment_LM %>%
#left_join(complexity) %>%
  left_join(sentiment_afinn) %>%
  left_join(sentiment_bing) %>%
  left_join(sentiment_nrc)

## Joining, by = "Accession_Number"

## Joining, by = c("Accession_Number", "accession_number")
## Joining, by = c("Accession_Number", "accession_number")

saveRDS(sentiment_df, "sentiment_df.rds")
#rm(sentiment_LM, sentiment_afinn, sentiment_bing, sentiment_nrc,  tokenised)

# Reading sentiment rds
sentiment_df <- readRDS("sentiment_df.rds")



# Plot common positive and negative words using Bing dictionary
tokens_bing %>% ungroup()  %>%
  count(word, sentiment, sort = TRUE) %>%
  mutate(word = reorder(word, n)) %>%
  group_by(sentiment) %>%
  top_n(25) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free") +
  labs(y = "Contribution to sentiment - Bing Dictionary", x = NULL) +
  coord_flip() +
  ggtitle('Words Contributing to Positive & Negative Sentiment in MDA')
```




```r
# Plot common sentiment words using NRC dictionary
tokens_nrc %>% ungroup()  %>%
  count(word, sentiment, sort = TRUE) %>%
  mutate(word = reorder(word, n)) %>%
  group_by(sentiment) %>%
  top_n(15) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free") +
 # labs(y = "Contribution to sentiment - NRC Dictionary", x = NULL) +
  coord_flip() +
  ggtitle('Words Contributing to Sentiments in MDA - NRC Dictionary')
```


Words Contributing to Sentiments in MDA - NRC Dictionary
anger
cash
loss
demand
uncertain
limited
volatility
fee
adverse
wireless
force
money
indenture
court
defense
rating
0
2500
5000
7500
10000
anticipation
cash
income
result
offset
time
expected
share
expect
continue
plan
risk
oracle
board
delivery
gain
0
2500
5000
7500
10000
word
joy
cash
income
share
outstanding
resources
gain
pay
favorable
grant
majority
respect
providing
award
established
completion
0
2500
5000
7500
10000
negative
tax
income
expenses
impairment
foreign
decrease
gross
margin
lower
loss
debt
demand
subject
outstanding
payment
0
3000
6000
9000
surprise
expect
uncertain
judgment
assessment
favorable
volatility
variable
wireless
award
sun
labor
present
money
larger
advance
0
500
1000
trust
cash
income
related
credit
amortization
management
general
securities
share
effective
agreement
administrative
expect
continue
maintenance
0
2500
5000
7500
10000



Downloading stock price

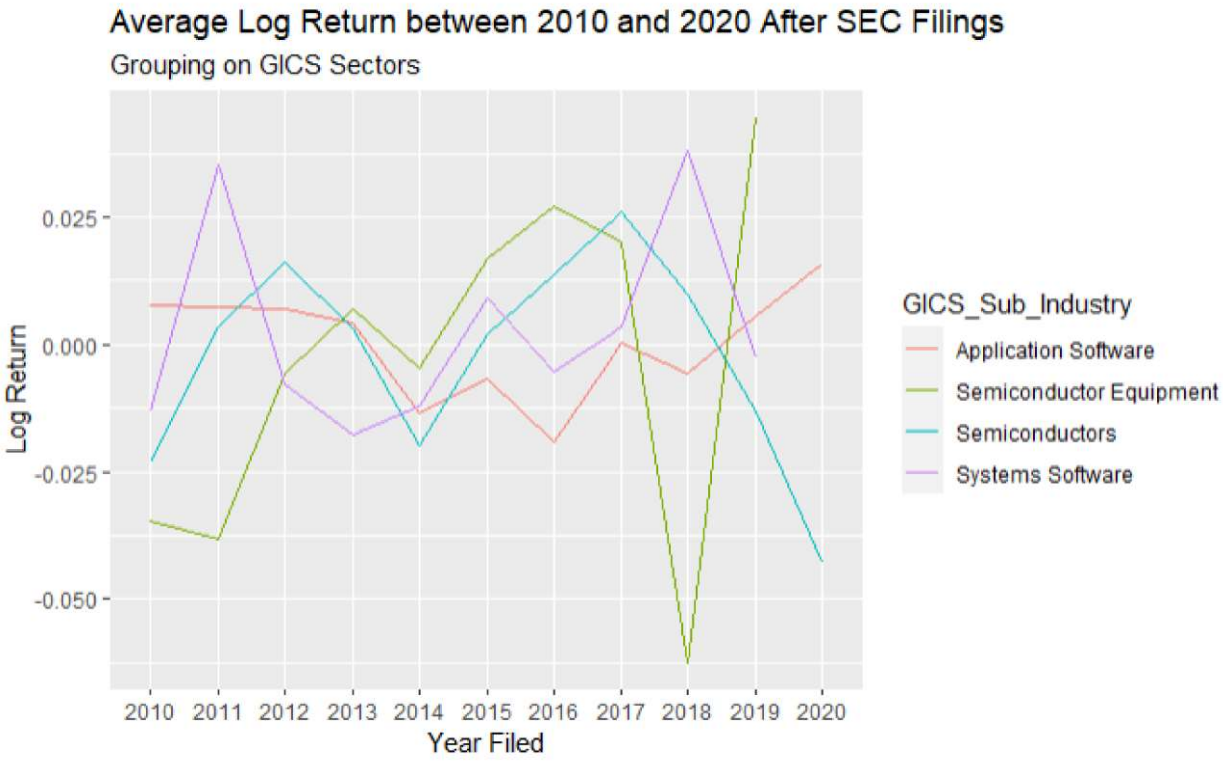
Here we are assuming that the effect of report filing can be captured by comparing prices in Day-7 and Day+7 of the filing date.

In this section, we consider two measures of stock price change: the log return adjusted value, where the log return difference between 2 dates are taken, as well as the adjusted price change which takes the ratio between the new adjusted stock price (price after posting a dividend) is divided by the old adjusted price of 100%. A value of 0% implies no price change and 10% implies a 10% increase in the stock price.

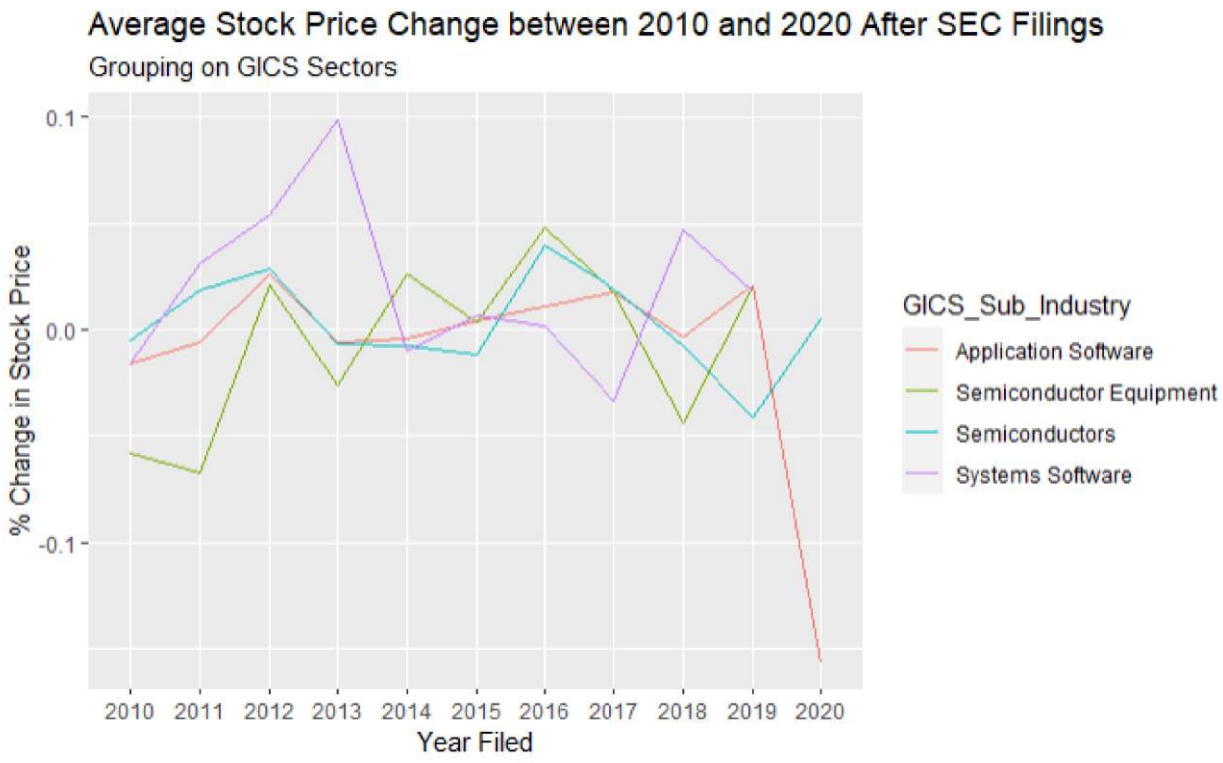
```
# Merge stock information with Master Index
#stock_info_all <- merged_mda %>% inner_join(stock_info, by=c("Symbol"="ticke
r"))

# Downloading stock prices from BatchGetSymbols
library(chron)
library(BatchGetSymbols)
library(data.table)
library(lubridate)

#Read RDS
merged_mda <- readRDS("merged_mda.rds")
# Get Average Log Return for each company
stockreturn_adj_price <-data.frame()

for (r in 1: nrow(merged_mda)){
```

```
  tryCatch({
    mda_report <- merged_mda [r, ]

    # Extract stock information
    stock_data <- BatchGetSymbols(tickers = mda_report$Symbol,
                                  first.date = mda_report$Date_Filed -7,
                                  last.date = mda_report$Date_Filed +3,
                                  type.return="log")

    # FIler the second day and last day
    stock_data_filtered <- stock_data [[2]] %>%
      filter(ref.date == max(ref.date) | row_number()==2) %>%
      arrange(desc(ref.date))

    # Calculate stock price change on log scale
    return_adj_price <- mda_report %>% select(Accession_Number) %>%
      mutate(return_adj_price=stock_data_filtered$ret.closing.prices[1] - sto
ck_data_filtered$ret.closing.prices[2]) #return difference

    price_adj_ratio <- mda_report %>% select(Accession_Number) %>%
      mutate(price_adj_ratio = (stock_data_filtered$price.adjusted[1]/ stock_
data_filtered$price.adjusted[2]) - 1) # stock price ratio

    accession_number <- mda_report$Accession_Number[1]
  })
  stockreturn_adj_price <- rbind(stockreturn_adj_price, return_adj_price)
}


# Merge stock adjustment price with Master Index
stockreturn_adj_pricev <- merged_mda %>% inner_join(stockreturn_adj_price, by
=c("Accession_Number"="Accession_Number"))

saveRDS(stockreturn_adj_pricev, "stockreturn_adj_pricev.rds")

stockreturn_adj_pricev <-readRDS("stockreturn_adj_pricev.rds")

merged_mda <- readRDS("merged_mda.rds")
# Get Price Adjustment Ratio for each company
stockprice_adj_ratio <-data.frame()

for (r in 1: nrow(merged_mda)){
  tryCatch({
    mda_report <- merged_mda [r, ]

    # Extract stock information
    # we will calculate 2 new fields: first.date and last.date
    # this is done in case the report is filed on Friday
    # so we allow for a period of two calendar weeks to be
```

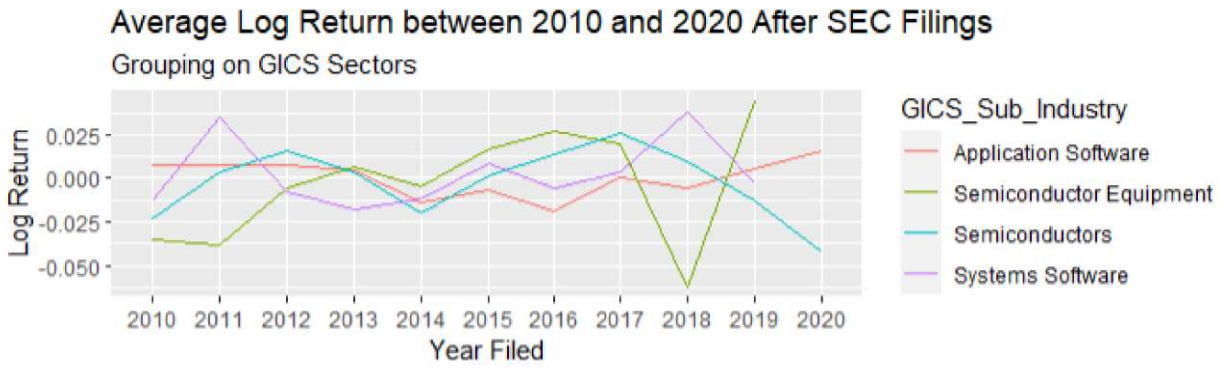
```
    # to be captured within the report release
    stock_data <- BatchGetSymbols(tickers = mda_report$Symbol,
                                  first.date = mda_report$Date_Filed -7,
                                  last.date = mda_report$Date_Filed +3,
                                  type.return="log")

    # FIlter the second day and last day
    stock_data_filtered <- stock_data [[2]] %>%
      filter(ref.date == max(ref.date) | row_number()==2) %>%
      arrange(desc(ref.date))

    # Calculate stock price change on log scale
    return_adj_price <- mda_report %>% select(Accession_Number) %>%
      mutate(return_adj_price=stock_data_filtered$ret.closing.prices[1] - sto
ck_data_filtered$ret.closing.prices[2]) #return difference

    price_adj_ratio <- mda_report %>% select(Accession_Number) %>%
      mutate(price_adj_ratio = (stock_data_filtered$price.adjusted[1]/ stock_
data_filtered$price.adjusted[2]) - 1) # stock price ratio

    accession_number <- mda_report$Accession_Number[1]
  })
  stockprice_adj_ratio <- rbind(stockprice_adj_ratio, price_adj_ratio)
}


# Merge stock adjustment price with Master Index
price_adj_ratiov <- merged_mda %>% inner_join(stockprice_adj_ratio, by=c("Acc
ession_Number"="Accession_Number"))

saveRDS(price_adj_ratiov, "price_adj_ratiov.rds")

price_adj_ratiov <-readRDS("price_adj_ratiov.rds")

# Plot Average Return
stockreturn_adj_pricev$Year <- as.factor(stockreturn_adj_pricev$Year)
return_avg <- stockreturn_adj_pricev %>%
              group_by(GICS_Sub_Industry, Year) %>%
              summarise(return_adj_price = mean(return_adj_price))

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

return_avg_plot <- return_avg %>%
  ggplot(aes(x = Year, y = return_adj_price, color = GICS_Sub_Industry)) +
  geom_line(aes(group=GICS_Sub_Industry)) +
  labs(title = "Average Log Return between 2010 and 2020 After SEC Filings",
subtitle = "Grouping on GICS Sectors", x="Year Filed", y = "Log Return", lege
nd = "GICS Sector")
return_avg_plot
```

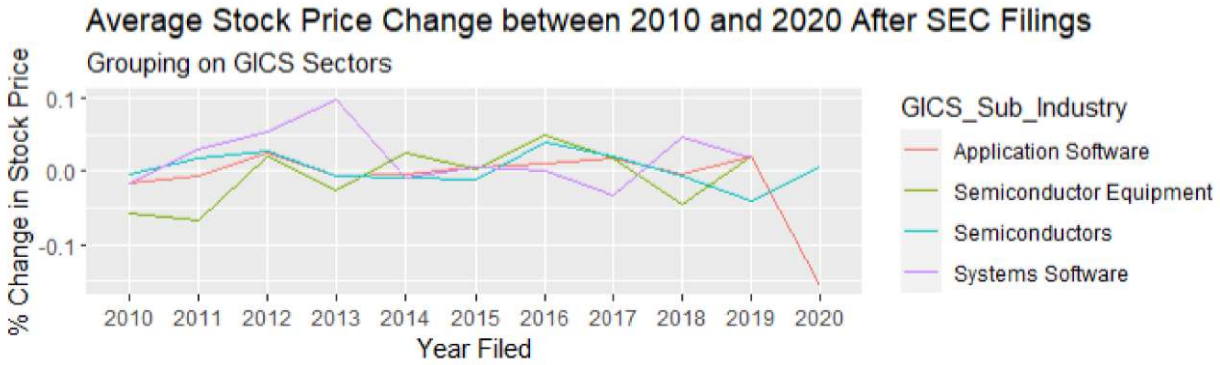



```
# Plot Average Stock Price Change
price_adj_ratiov$Year <- as.factor(price_adj_ratiov$Year)
price_avg <- price_adj_ratiov %>%
              group_by(GICS_Sub_Industry, Year) %>%
              summarise(price_adj_ratio = mean(price_adj_ratio))

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

price_avg_plot <- price_avg %>%
  ggplot(aes(x = Year, y = price_adj_ratio, color = GICS_Sub_Industry)) +
  geom_line(aes(group=GICS_Sub_Industry)) +
  labs(title = "Average Stock Price Change between 2010 and 2020 After SEC Fi
lings", subtitle = "Grouping on GICS Sectors", x="Year Filed", y = "% Change
in Stock Price", legend = "GICS Sector")
price_avg_plot
```




```
library(ggpubr)

arrange <- ggarrange(return_avg_plot, price_avg_plot, ncol = 1, nrow = 2)
arrange
```





The average stock price change where the % change is used appears to have less variability than the log return. Moreover, by having the y-scale in % change makes it relatively easier to interpret and has been normalised as it is already in a percentage format. Thus, we decided to progress with our analysis using the stock price % of change instead.

Sentiment Analysis Effect on Stock Price Change

Evaluate the predictability of stock price at the time of the report against the variables obtained from sentiment analysis

```
# Joining sentiment features with price
sentiment_data <- sentiment_df %>%
  left_join(price_adj_ratiov) %>%
  left_join(stockreturn_adj_pricev)

## Joining, by = "Accession_Number"

## Joining, by = c("Accession_Number", "CIK", "Symbol", "Security", "Company_
Name", "GICS_Sub_Industry", "GICS_Sector", "Date_Filed", "Quarter", "Year", "
Text")

# Running the regression models
sentiment_regression <- sentiment_data %>%
  select(-c(Year, CIK, Company_Name, GICS_Sector, GICS_Sub_Industry, Symbol,
Security, Date_Filed, Quarter, return_adj_price))

  str(sentiment_regression)
```

```
## grouped_df [176 x 31] (S3: grouped_df/tbl_df/tbl/data.frame)
##  $ Accession_Number: chr [1:176] " 0000002488-18-000042" " 0000002488-19-0
00011" " 0000004127-11-000015" " 0000004127-12-000042" ...
##  $ strong_modal    : num [1:176] 29 21 15 6 5 5 5 4 4 25 ...
##  $ weak_modal      : num [1:176] 34 19 22 12 10 11 11 11 9 50 ...
##  $ accession_number: chr [1:176] NA NA NA NA ...
##  $ LM_total_words  : int [1:176] 588 249 410 259 246 237 237 231 211 562 .
..
##  $ LM_sent         : num [1:176] -101 -10 -47 -42 -39 -31 -34 -37 -37 -50
...
##  $ LM_positive     : num [1:176] 0.0986 0.1606 0.0829 0.0618 0.0935 ...
##  $ LM_negative     : num [1:176] 0.27 0.201 0.198 0.224 0.252 ...
##  $ LM_litigious    : num [1:176] 0.2109 0.0884 0.0951 0.1313 0.1463 ...
##  $ LM_uncertainty  : num [1:176] 0.133 0.161 0.385 0.34 0.297 ...
##  $ LM_constraining : num [1:176] 0.18 0.229 0.149 0.174 0.15 ...
##  $ afinn_sent      : num [1:176] 416 195 217 142 149 180 176 136 101 338 .
..
##  $ bing_total_words: int [1:176] 350 182 248 171 173 174 165 143 138 282 .
..
##  $ bing_sent       : num [1:176] 136 60 72 41 41 36 23 21 22 84 ...
##  $ bing_positive   : num [1:176] 0.694 0.665 0.645 0.62 0.618 ...
##  $ bing_negative   : num [1:176] 0.306 0.335 0.355 0.38 0.382 ...
##  $ suprise         : num [1:176] NA NA NA NA NA NA NA NA NA NA ...
##  $ nrc_total_words : int [1:176] 2782 1695 1658 1349 1318 1408 1420 1396 1
262 2296 ...
##  $ nrc_sent        : num [1:176] 457 194 232 207 181 208 185 169 145 310 .
..
##  $ nrc_positive    : num [1:176] 0.312 0.277 0.276 0.277 0.266 ...
##  $ nrc_negative    : num [1:176] 0.148 0.163 0.136 0.124 0.128 ...
##  $ nrc_anger       : num [1:176] 0.0474 0.0531 0.0597 0.0615 0.0577 ...
##  $ nrc_fear        : num [1:176] 0.0381 0.0466 0.076 0.0764 0.0744 ...
##  $ nrc_trust       : num [1:176] 0.187 0.165 0.16 0.17 0.167 ...
##  $ nrc_sadness     : num [1:176] 0.0816 0.0956 0.0808 0.0697 0.0713 ...
##  $ nrc_surprise    : num [1:176] 0.01042 0.00649 0.01327 0.01408 0.01517 .
..
##  $ nrc_disgust     : num [1:176] 0.00827 0.01121 0.00965 0.01631 0.01821 .
..
##  $ nrc_joy         : num [1:176] 0.069 0.082 0.073 0.0778 0.0789 ...
##  $ nrc_anticipation: num [1:176] 0.0988 0.0997 0.1152 0.1127 0.1237 ...
##  $ Text            : chr [1:176] "item management discussion and analysis
of financial condition and results of operations the following discussi"| __t
runcated__ "item management discussion and analysis of financial condition an
d results of operations the following discussi"| __truncated__ "item manageme
nt discussion and analysis of financial condition and results of operation th
e following discussio"| __truncated__ "item management discussion and analysi
s of financial condition and results of operations the following discussi"| _
_truncated__ ...
##  $ price_adj_ratio : num [1:176] 0.00768 -0.04476 0.01635 0.07506 0.03951
...
##  - attr(*, "groups")= tibble [176 x 2] (S3: tbl_df/tbl/data.frame)
```

```
##   ..$ Accession_Number: chr [1:176] " 0000002488-18-000042" " 0000002488-1
9-000011" " 0000004127-11-000015" " 0000004127-12-000042" ...
##   ..$ .rows           : list<int> [1:176]
##   .. ..$ : int 1
##   .. ..$ : int 2
##   .. ..$ : int 3
##   .. ..$ : int 4
##   .. .. ..

          $ : int 98
##   .. ..$ : int 99
##   .. .. [list output truncated]
##   .. ..@ ptype: int(0)
##   ..- attr(*, ".drop")= logi TRUE
```

```
reg_LM <- lm(price_adj_ratio ~ strong_modal + weak_modal + LM_total_words + L
M_sent + LM_positive + LM_negative + LM_litigious + LM_uncertainty + LM_const
raining, data=sentiment_regression)

reg_bing <- lm(price_adj_ratio ~ bing_total_words+ bing_sent+ bing_positive+
bing_negative, data=sentiment_regression)

reg_afinn <- lm(price_adj_ratio ~ afinn_sent, data=sentiment_regression)

reg_nrc <- lm(price_adj_ratio ~ nrc_total_words + nrc_sent + nrc_positive + n
rc_negative + nrc_anger + nrc_fear + nrc_trust + nrc_sadness + nrc_surprise +
nrc_disgust + nrc_joy + nrc_anticipation, data=sentiment_regression)

lm_final <- stargazer::stargazer(reg_LM, reg_bing, reg_afinn, reg_nrc, type='
text')
```

	Dependent variable:			
	price_adj_ratio			
	(1)	(2)	(3)	(4)
strong_modal	0.002**			
	(0.001)			
weak_modal	0.001*			
	(0.001)			
LM_total_words	-0.0001			
	(0.0001)			
LM_sent	-0.0001			
	(0.0001)			
LM_positive	0.646*			
	(0.380)			
LM_negative	0.441			
	(0.276)			
LM_litigious	0.487			
	(0.326)			
LM_uncertainty	0.467			
	(0.306)			
LM_constraining	0.528			
	(0.340)			
bing_total_words		0.00004		
		(0.00003)		
bing_sent		-0.0001		
		(0.0001)		
bing_positive		0.069		
		(0.085)		
bing_negative				
afinn_sent			0.00002	
			(0.00001)	

```
nrc_total_words                                                                    -0.00000
                                                                                  (0.00001)

nrc_sent                                                                           0.00004
                                                                                   (0.0001)

nrc_positive                                                                       -0.133
                                                                                   (0.453)

nrc_negative                                                                       -0.217
                                                                                   (0.538)

nrc_anger                                                                          -0.451
                                                                                   (0.578)

nrc_fear                                                                           -0.317
                                                                                   (0.621)

nrc_trust                                                                          -0.290
                                                                                   (0.421)

nrc_sadness                                                                        -0.922*
                                                                                   (0.532)

nrc_surprise                                                                       -0.312
                                                                                   (1.005)

nrc_disgust                                                                         1.251
                                                                                   (0.845)

nrc_joy                                                                            0.785
                                                                                   (0.618)

nrc_anticipation

Constant                  -0.440              -0.046              -0.004              0.161
                          (0.275)             (0.053)             (0.007)             (0.336)

-------------------------------------------------------------------------------------------------
Observations                176                 176                 176                 176
R2                         0.035               0.008               0.006               0.087
Adjusted R2               -0.017              -0.009              0.0004               0.026
Residual Std. Error  0.047 (df = 166)    0.047 (df = 172)    0.046 (df = 174)    0.046 (df = 164)
F Statistic         0.677 (df = 9; 166) 0.480 (df = 3; 172) 1.067 (df = 1; 174) 1.422 (df = 11; 164)
=================================================================================================
Note:                                                                 *p<0.1; **p<0.05; ***p<0.01
```

Out of all dictionaries tested, NRC has the highest fit for coefficient of determination with R-Sq = 0.087, followed by Loughran&McDonald's of R-Sq = 0.035 and Afinn is at the bottom list of R-Sq = 0.006.

However, none of the p-values from the dictionaries are significant thus sentiment does not seem to play a role in predicting stock price.

Sentiment Features for Predicting Stock Price Change After 10-K Filings

```r
# Model sentiment every GICS Sub Industry
model_output_sub_industry <- data.frame()

for (s in 1:length(sub_industry)) {
  model_data <- sentiment_data %>%
```

```
    filter(GICS_Sub_Industry == sub_industry[s]) %>%
  select(-c(Accession_Number, Year, CIK, Company_Name, return_adj_price, GICS
_Sector))

# Modelling
  model <- lm(price_adj_ratio ~., data = model_data)

# Prep data for plotting
  local_df <- head(as.data.frame(summary(model)$coefficients) %>%
                     tibble::rownames_to_column() %>%
                     mutate(absolute_t_value = abs('t value')) %>%
                     arrange(desc(absolute_t_value)), 10) %>%
                     mutate(rowname = factor(rowname, levels = rowname)) %>%
    mutate(significance = case_when('Pr(>|t|' <= 0.001 ~ 'significant***', 'P
r(>|t|' <= 0.01 ~ 'significant**', 'Pr(>|t|' <= 0.05 ~ 'significant*', TRUE ~
'not significant')) %>%
    mutate(GICS_Sub_Industry = sub_industry[s]) %>%
    mutate(r_squared = paste('Multiple R-Squared:', as.character(round(summar
y(model)$r.squared,3))))

  model_output_sub_industry <- bind_rows(model_output_sub_industry, local_df)
}

ggplot(model_output_sub_industry, aes(x = reorder(rowname, absolute_t_value),
y = absolute_t_value, fill = significance)) +geom_bar(stat = "identity") +
  labs(title = "Top 10 Sentiment Features for Predicting Stock Price Change A
fter 10-K Filings", subtitle = "Grouping on GICS Sub Industry", y= "Absolute
t-value", x="Features") +
  ylim(0,4) + coord_flip() +
  facet_wrap(~GICS_Sub_Industry+r_squared, ncol = 4, scales = "free")

rm(model_output_sub_industry)
```

Conclusions

This section has shown out of all dictionaries tested, NRC has the highest fit for coefficient of determination with R-Sq = 0.087, followed by Loughran&McDonald's of R-Sq = 0.035 and Afinn is at the bottom list of R-Sq = 0.006.

However, none of the p-values from the dictionaries are significant thus sentiment does not seem to play a role in predicting stock price.

Part C

Data Preparation

This analysis is conducted only on 10-K reports. In a case where there are missing 10-K reports, we will just keep them as the way they are and not replace them with any corresponding 10-Q forms or any other supporting materials. Making our analysis limited to in-depth analysis of the available 10-K reports.

```
library(stm)

# Staging the dataset
# From our result in part B, we decided to use % change in stock price instea
d of the log price scale
# Thus we would like to merge the downloaded price data with our tokenised wo
rds
# Loading the necessary files:
price_adj_ratiov <- readRDS("price_adj_ratiov.rds")
# We also had completed post-tagging in Part A, thus we just have to load the
dataframe
tokens <- readRDS("postagged_mda_df_joined_short.rds")

# Merging the two dataset together we have
tokens <- price_adj_ratiov %>%
          select(Accession_Number, price_adj_ratio) %>%
          inner_join(tokens, by=(c("Accession_Number" = "Accession_Number")))
tokens <- as.data.frame(tokens)

data_for_stm <- tokens %>%
              group_by(Accession_Number) %>%

              summarise(annotated_text = paste(lemma, collapse = " "))

subset_data <- tokens %>%
                select(Accession_Number, price_adj_ratio)

# Prepare metadata for the STM model
data_for_stm <- data_for_stm %>%
                left_join(subset_data) %>%
                distinct(.) %>%
                na.omit()

## Joining, by = "Accession_Number"

# gsub for expensis
# according to http://www.definition.com.co/expensis.html
# we can gsub them with expense
data_for_stm$annotated_text <- gsub("expensis", "expense", data_for_stm$annot
ated_text, ignore.case = T)
```

```
processed <- textProcessor(data_for_stm$annotated_text,
metadata = data_for_stm,
customstopwords = c("tax","revenue","increase","cash","income","net","relate"
,"due", "fiscal"),
stem = F)

## Building corpus...
## Converting to Lower Case...
## Removing punctuation...
## Removing stopwords...
## Remove Custom Stopwords...
## Removing numbers...
## Creating Output...

# Keep only those words that appear
# on 1% of the corpus
# considering the size of the corpus
threshold <- round(1/100 * length(processed$documents),0)

out <- prepDocuments(processed$documents,
                     processed$vocab,
                     processed$meta,
                     lower.thresh = threshold)

## Removing 2321 of 6122 terms (3076 of 132104 tokens) due to frequency
## Your corpus now has 176 documents, 3801 terms and 129028 tokens.

saveRDS(out, "out.rds")
saveRDS(data_for_stm, "data_for_stm.rds")
out <- readRDS("out.rds")
# look at out$vocab to inspect the vocab
# spotted a few spelling errors which we corrected
```

Initial exploration (Unsupervised Approach)

The aim of topic modelling is to discover topics assumed to represent a corpus of documents. Robert, et. al. (2016) explains that Structural Topic Modelling allows analysts to estimate the relationships between topics and their corresponding documents' meta-data and we will be showing this accordingly. Intial explorations will be conducted until it converges into meaningful topic members.

```
library(Rtsne)
library(rsvd)
library(geometry)

# Running the STM algorithm with K set to 0
# to get the right number of topics
stmfit_0 <- stm(documents = out$documents,
                vocab = out$vocab,
```

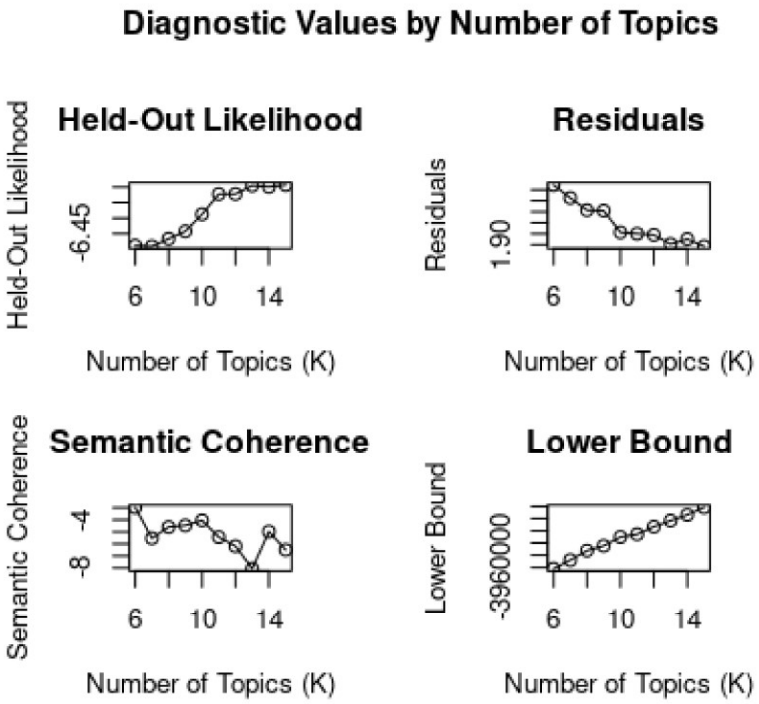
```
                      K = 0,
                      prevalence =~price_adj_ratio,
                      max.em.its = 40,
                      data = out$meta,
                      reportevery= 5,
                      # gamma.prior = "L1",
                      sigma.prior = 0.7,
                      seed = 123,
                      init.type = "Spectral")
# yields to 66 topics, it terminated before convergence reached
```

Supervised Approach

Ideally, we would want to have the best K that has the highest Held-Out Ratio, Semantic Coherence and simultaneously has the lowest Residuals. From the function above, when we set K=0, the function yield to 66 topics as the 'best' outcome. However, it performed poorly by only having 3.8% of variance explained and there were a lot of cross-loading issues of the top loadings observed. Thus, we decided to opt for a heuristic approach to trial out different values of K that will give the 'best' outcome.

Finally, we came up with a range of numbers for 'searchK' function to run. We decided to do a local search from K=6 to K=15 with a step of 1. The reason is because the difference of Semantic Coherence is relatively higher than the Held-Out Likelihood (from -7 to -3 =|4| whilst difference in likelihood is only 0.15). In addition to that, a fewer number of topics would make a relatively easier interpretation.

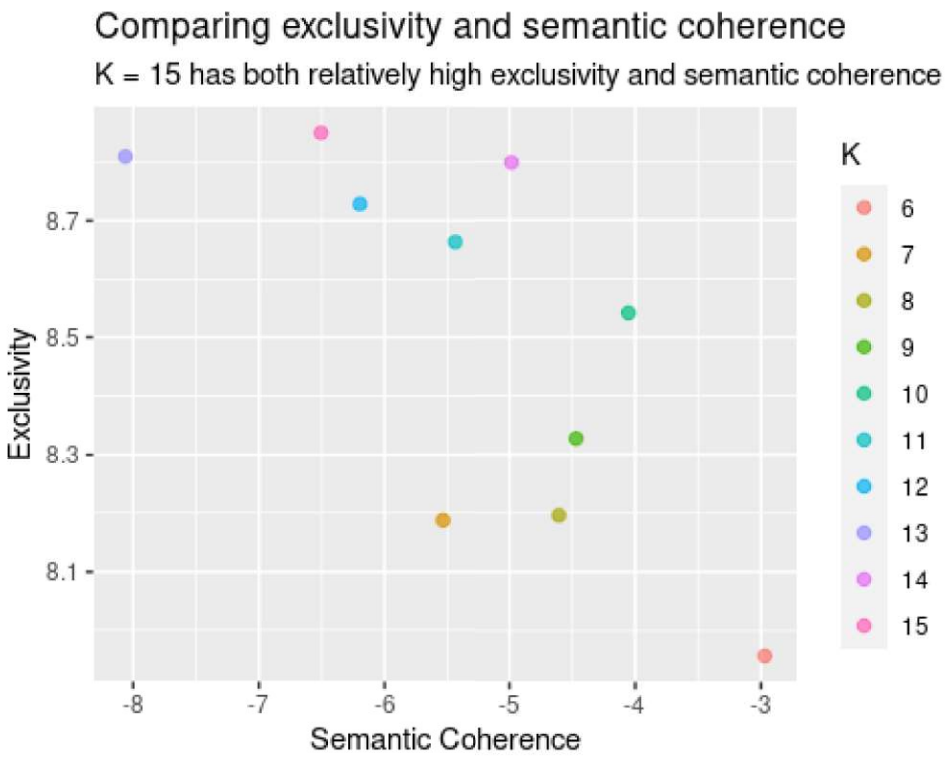
```
# By setting K = 0, STM returned us with 134 topics as the optimum topic numb
er
#searching for optimal k
numtopics <- searchK(out$documents, out$vocab, K = seq(from=6, to=15,by=1), p
revalence=~price_adj_ratio, data = out$meta)

plot(numtopics)
```



Evaluating model performance

From the result above, we could see that K=9 returns a model with the highest Semantic Coherence, relatively high Held-Out Likelihood, low Residuals and K = 11 seem to also to be a good candidate for K. Please refer to Appendix for other supporting results (trial with several different values of K).

```
#let's look at exclusivity vs semantic coherence
numtopics$results %>%
  select(K, exclus, semcoh) %>%
  filter(K %in% seq(from=6, to=15,by=1)) %>%
  unnest() %>%
  mutate(K = as.factor(K)) %>%
  ggplot(aes(semcoh, exclus, color = K)) +
  geom_point(size = 2, alpha = 0.7) +
  labs(x = "Semantic Coherence",
       y = "Exclusivity",
       title = "Comparing exclusivity and semantic coherence",
       subtitle = "K = 15 has both relatively high exclusivity and semantic c
oherence")

## Warning: `cols` is now required when using unnest().
## Please use `cols = c(K, exclus, semcoh)`
```



From the graph above, ideally we would want to have K that would give the highest values both in Exclusivity and Semantic Coherence. Both K = 9 and K= 11 can potentially be good candidates for K. There seem to have been a trade-off between Semantic Coherence and Exclusivity in our case here. Thus we would carry on with both K values and decide which K gives the best result & interpretation.

After further analysis, we decided to carry on with K=9 because it provides a relatively higher variance explained (28.1%) as opposed to those from K=11 (22.8% of variance explained).

```
# after running multiple models with various values of K
# we determined to go with K = 15 as it has the highest variance explained
# and topics are distinct enough to label

stmfit_1 <- stm(documents = out$documents,
                vocab = out$vocab,
                K = 9,
                prevalence =~price_adj_ratio,
                max.em.its = 75,
                data = out$meta,
                reportevery=3,
                # gamma.prior = "L1",
                sigma.prior = 0.7,
                seed = 123,
                init.type = "Spectral")
```

Extracting the theta matrix

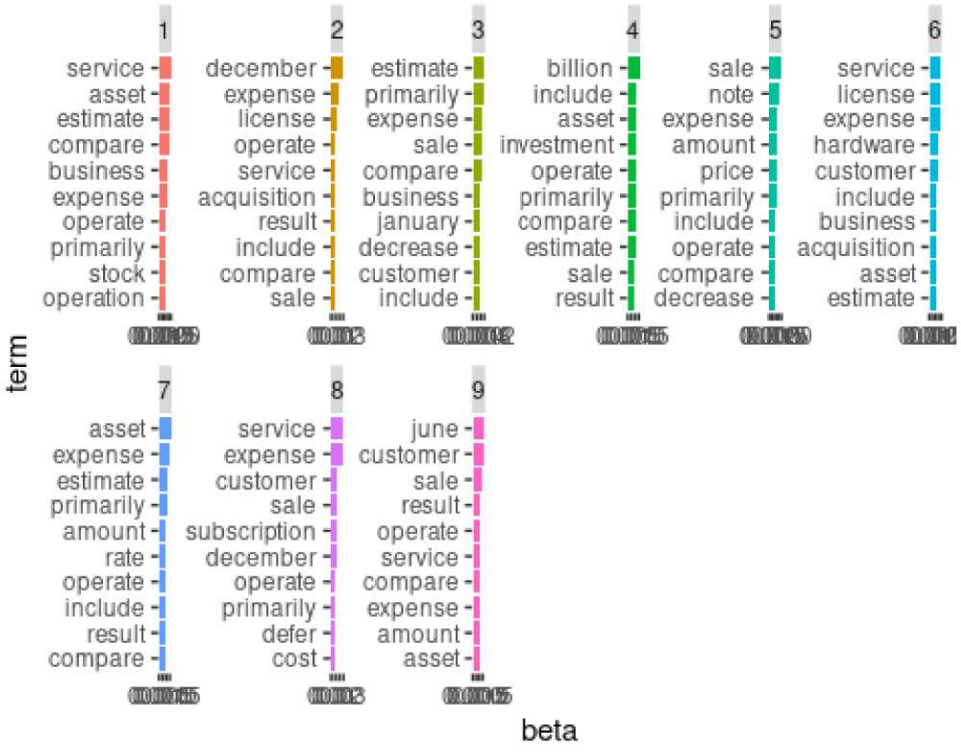
```
#extract the theta matrix
convergence_theta <- as.data.frame(stmfit_1$theta)
colnames(convergence_theta)<-paste0("topic_", 1:9)

#assigning topic summary, proportions and labels to variables
topic_summary <- summary(stmfit_1)

## A topic model with 9 topics, 176 documents and a 3801 word dictionary.

topic_proportions <- colMeans(stmfit_1$theta)
topic_labels <- paste0("topic_",1:9)

#wordcloud for topics
stm::cloud(stmfit_1)
```

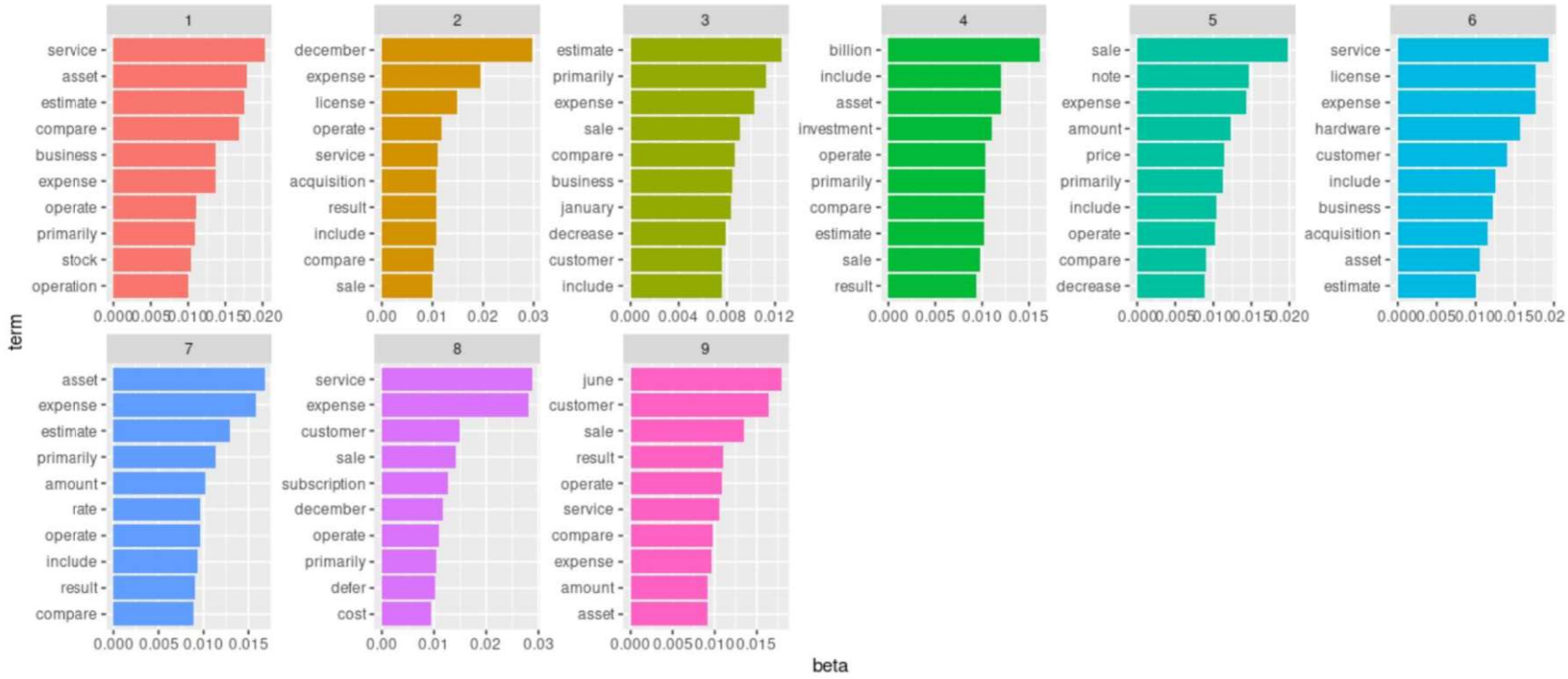
Here is a word cloud of the most common words across the topic model.

Finding the most common words in within each topics:

```
td_beta <- tidy(stmfit_1)
td_beta

top_terms <-
  td_beta %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
```

```
  arrange(topic, desc(beta))

plot <- top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, ncol = 6, scales = "free") +
  scale_y_reordered()
plot

td_gamma <- tidy(stmfit_1, matrix = "gamma",
                 document_names = rownames(out$documents))
top_terms <-
  td_beta %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, desc(beta))

plot <- top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, ncol = 6, scales = "free") +
  scale_y_reordered()
plot
```

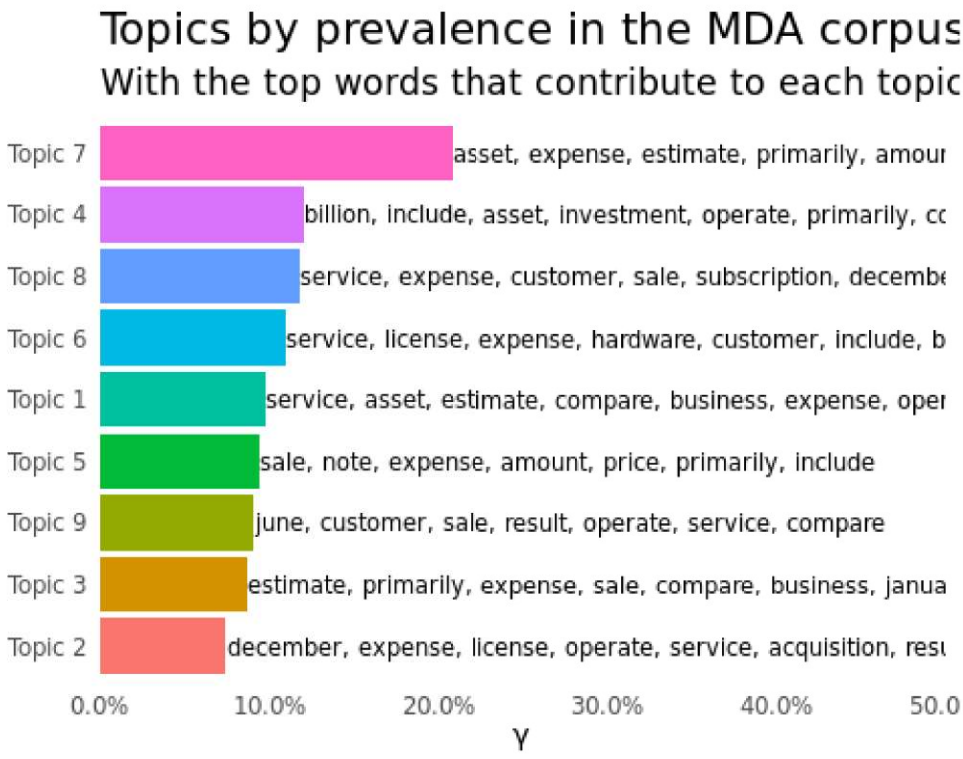



1
service
asset
estimate
compare
business
expense
operate
primarily
stock
operation
0.000 0.005 0.010 0.015 0.020
2
december
expense
license
operate
service
acquisition
result
include
compare
sale
0.00 0.01 0.02 0.03
3
estimate
primarily
expense
sale
compare
business
january
decrease
customer
include
0.000 0.004 0.008 0.012
4
billion
include
asset
investment
operate
primarily
compare
estimate
sale
result
0.000 0.005 0.010 0.015
5
sale
note
expense
amount
price
primarily
include
operate
compare
decrease
0.000 0.005 0.010 0.015 0.020
6
service
license
expense
hardware
customer
include
business
acquisition
asset
estimate
0.000 0.005 0.010 0.015 0.02
7
asset
expense
estimate
primarily
amount
rate
operate
include
result
compare
0.000 0.005 0.010 0.015
8
service
expense
customer
sale
subscription
december
operate
primarily
defer
cost
0.00 0.01 0.02 0.03
9
june
customer
sale
result
operate
service
compare
expense
amount
asset
0.000 0.005 0.010 0.015
term
beta

```
library(ggthemes)
library(scales)
library(purrr)

##
## Attaching package: 'purrr'

## The following object is masked from 'package:scales':
##
##     discard

## The following object is masked from 'package:data.table':
##
##     transpose

top_terms <- td_beta %>%
  arrange(beta) %>%
  group_by(topic) %>%
  top_n(7, beta) %>%
  arrange(-beta) %>%
  select(topic, term) %>%
  summarise(terms = list(term)) %>%
  mutate(terms = map(terms, paste, collapse = ", ")) %>%
  unnest()

## Warning: `cols` is now required when using unnest().
## Please use `cols = c(terms)`

gamma_terms <- td_gamma %>%
  group_by(topic) %>%
  summarise(gamma = mean(gamma)) %>%
  arrange(desc(gamma)) %>%
  left_join(top_terms, by = "topic") %>%
  mutate(topic = paste0("Topic ", topic),
         topic = reorder(topic, gamma))

gamma_terms %>%
  top_n(20, gamma) %>%
  ggplot(aes(topic, gamma, label = terms, fill = topic)) +
  geom_col(show.legend = FALSE) +
  geom_text(hjust = 0, nudge_y = 0.0005, size = 3,
            family = "IBMPlexSans") +
  coord_flip() +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0, 0.50),
                     labels = percent_format()) +
  theme_tufte(base_family = "IBMPlexSans", ticks = FALSE) +
  theme(plot.title = element_text(size = 16,
                                  family="IBMPlexSans-Bold"),
        plot.subtitle = element_text(size = 13)) +
  labs(x = NULL, y = expression(gamma),
```

```
      title = "Topics by prevalence in the MDA corpus",
      subtitle = "With the top words that contribute to each topic")
```

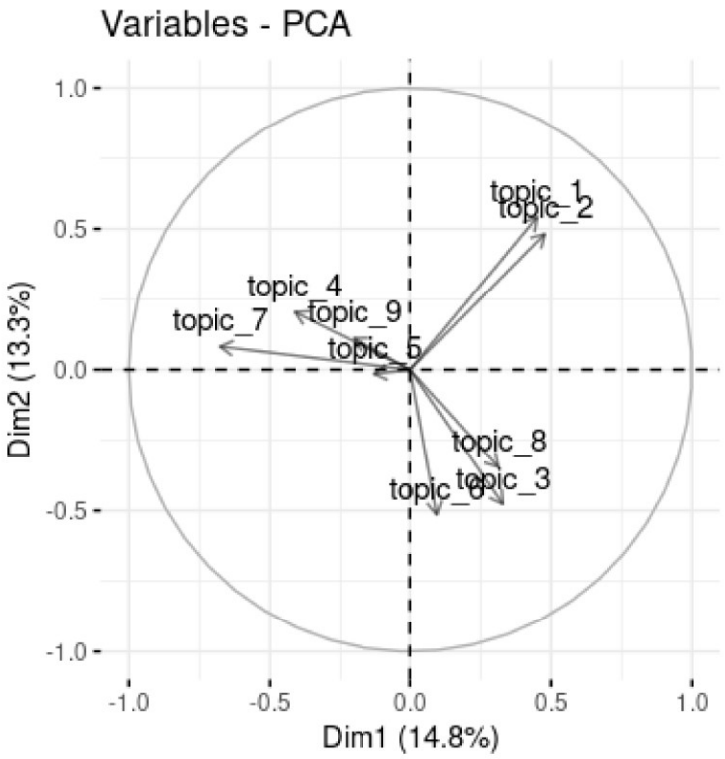
Here is a list of the most dominating topics in the Management Discussion & Analysis corpus:



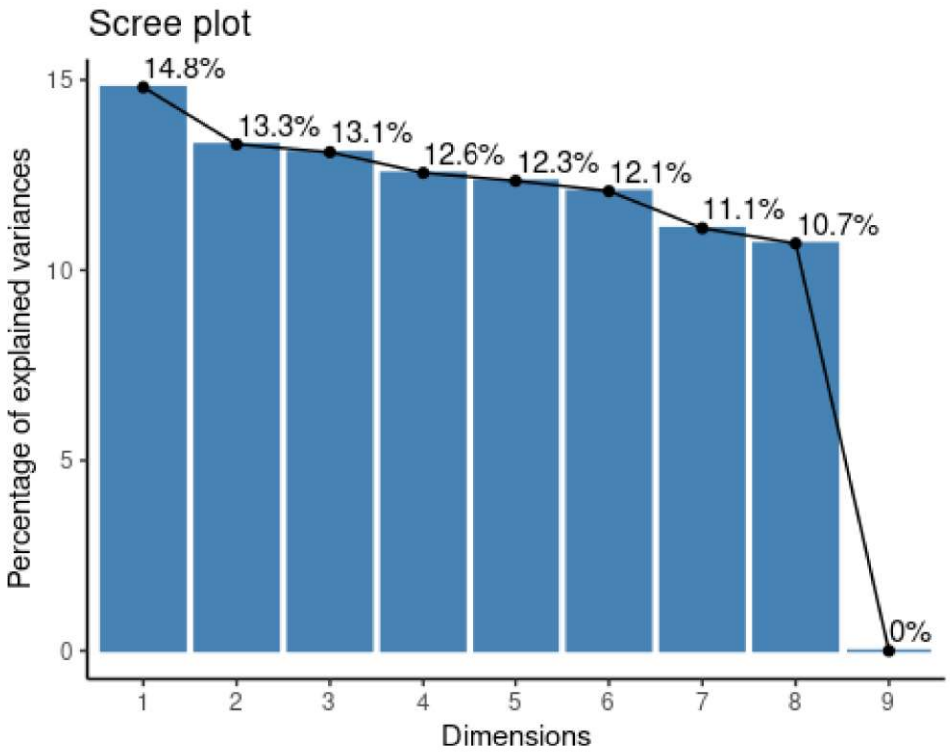

```
gamma_terms %>%
  select(topic, gamma, terms) %>%
  knitr::kable(digits = 3,
        col.names = c("Topic", "Expected topic proportion", "Top 7 terms"))
```

Topic	Expected topic proportion	Top 7 terms in each topics
Topic 7	0.209	asset, expense, estimate, primarily, amount, rate, operate
Topic 4	0.120	billion, include, asset, investment, operate, primarily, compare
Topic 8	0.118	service, expense, customer, sale, subscription, december, operate
Topic 6	0.109	service, license, expense, hardware, customer, include, business
Topic 1	0.097	service, asset, estimate, compare, business, expense, operate
Topic 5	0.094	sale, note, expense, amount, price, primarily, include
Topic 9	0.091	june, customer, sale, result, operate, service, compare
Topic 3	0.087	estimate, primarily, expense, sale, compare, business, january
Topic 2	0.074	december, expense, license, operate, service, acquisition, result

```
# Tidying it up
gamma_topics <- td_gamma %>%
pivot_wider(names_from = topic, values_from = gamma)

# Adding column name
colnames(gamma_topics) <- c("document",topic_labels)

#remove the document from the column
gamma_topics <- as.data.frame(gamma_topics)
rownames(gamma_topics) <- gamma_topics$document
gamma_topics$document <- NULL

#perform PCA
pcav <- FactoMineR::PCA(gamma_topics, graph = FALSE)
factoextra::fviz_pca_var(pcav, alpha.var = .5)
```

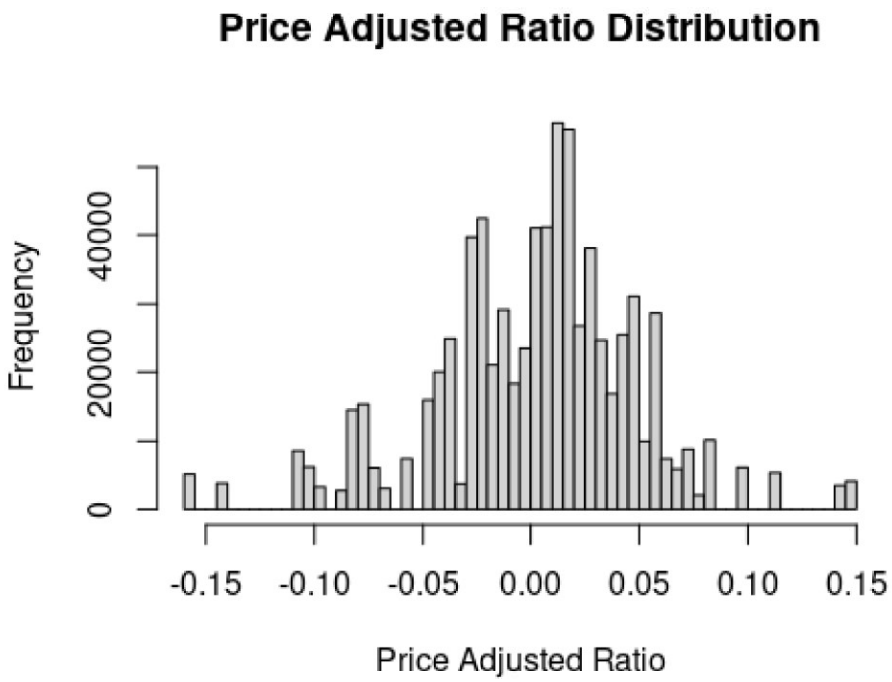



```
factoextra::fviz_screeplot(pcav, addlabels = TRUE) + theme_classic()
```



From the PCA plot above, we could see that a model with 9 topics yield to the best variance explained of 28.1%. This can also be reflected in the scree plot above.

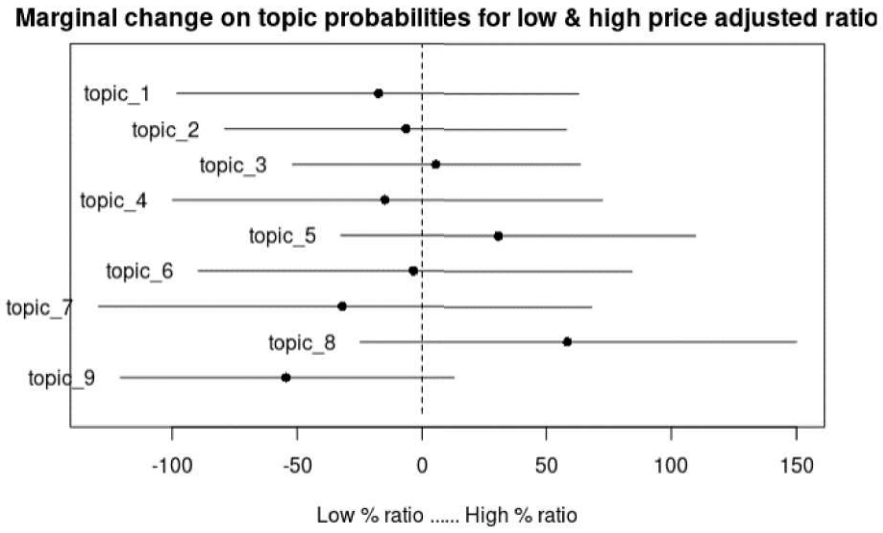
```
# checking the price distribution
hist(tokens$price_adj_ratio, xlab="Price Adjusted Ratio", main="Price Adjuste
d Ratio Distribution", breaks = 50)
```



STM Effect Estimation

```
# Estimate the effects of price and review score
# on topic probability
effects <- estimateEffect(~ as.numeric(price_adj_ratio),
                          stmobj = stmfit_1,
                          metadata = out$meta,
                          uncertainty = c("None"))

# Effect of review score on topic
# probability
plot(effects, covariate = "price_adj_ratio",
     topics = c(1:9),
     model = stmfit_1,
     method = "difference",
     cov.value1 = "100",
     cov.value2 = "0",
     xlab = "Low % ratio ..... High % ratio",
     #xlim = c(-0.6,0.6),
     main = "Marginal change on topic probabilities for low & high price adju
```

```
sted ratio",
     custom.labels = topic_labels[c(1:9)],
     labeltype = "custom")
```

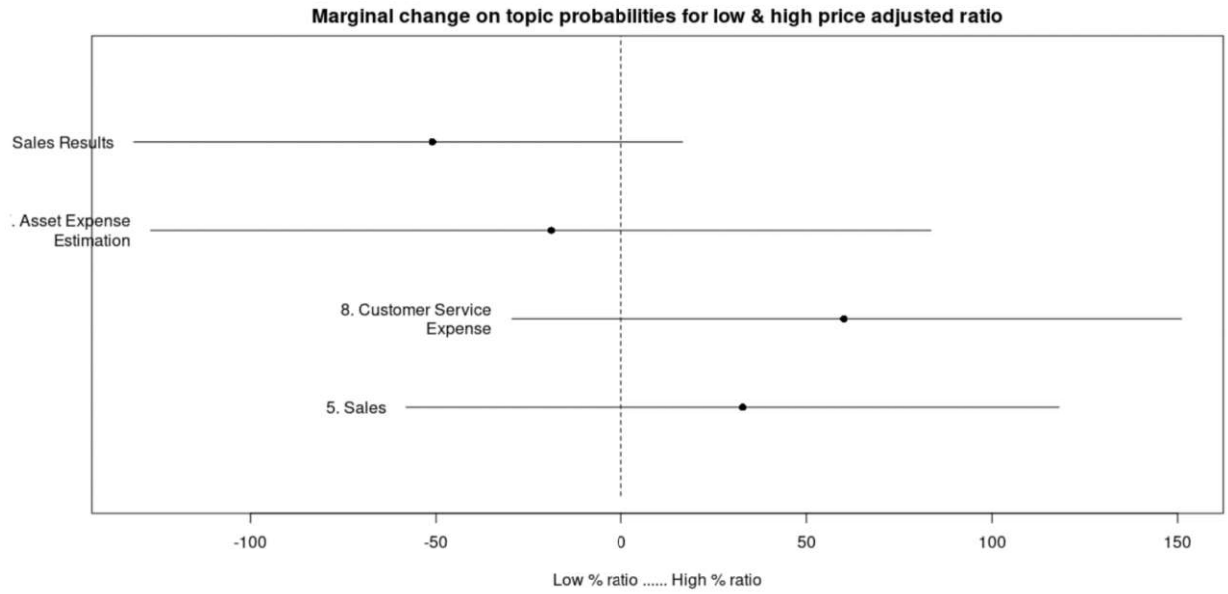



```
# let's add labels to each topic
# and dig into specific topics
# that lead to relatively lower and
# relatively higher % ratio

custom_labels <- c("1. Service Estimation ", "2. EoY License Expense", "3. Sa
les Estimation Comparisons", "4. Asset Investment", "5. Sales", "6. Business
Licensing Expense", "7. Asset Expense Estimation", "8. Customer Service Expen
se", "9. Customer Sales Results")

# summarizing topic labels below:
label_review <- knitr::kable(custom_labels)
label_review
```

TOPIC LABELS

1. Service Estimation
2. EoY License Expense
3. Sales Estimation Comparisons
4. Asset Investment
5. Sales
6. Business Licensing Expense
7. Asset Expense Estimation
8. Customer Service Expense
9. Customer Sales Results

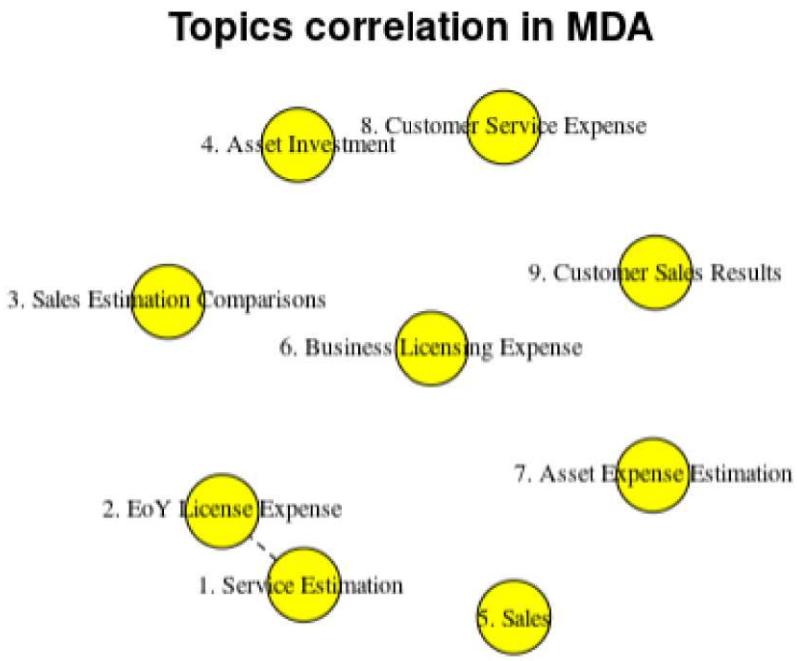
```
# The code below picks three topics and
# plots them according to their association with the
# low/high rating variable.


plot(effects, covariate = "price_adj_ratio",
     topics = c(9, 7, 8 ,5),
     model = stmfit_1,
     method = "difference",
     cov.value1 = "100",
     cov.value2 = "0",
     xlab = "Low % ratio ..... High % ratio",
     main = "Marginal change on topic probabilities for low & high price adju
sted ratio",
     custom.labels = custom_labels[c(9, 7, 8 ,5)],
     labeltype = "custom")
```



Topic 8 (Customer Service Expense) yields to the highest percentage (%) change in stock price. If we look deeper into topic members, they reflect on how well they treat their customers through operations such as end of year (Christmas) sales & subscription. Whilst, from topic 9 (Sales Results) we could see that customer sales in June (summertime) contributes the least into percentage (%) change in stock price.

```
# checking correlation between topics
plot(topicCorr(stmfit_1),
      topics = c(1:9),
      layout = NULL,
      vertex.color = "yellow",
```

```
      vertex.size = 30,
      vlabels = custom_labels,
      main = "Topics correlation in MDA")
```



The additive predictability that some topics add on estimating the stock price

Similar to what we did in sentiment analysis, we ran a regression of model is fit to all topics

```
to_regress_topic <- cbind(out$meta, convergence_theta)

model_topic_price <- lm(price_adj_ratio~ topic_1+topic_2+topic_3+topic_4+topi
c_5+topic_6+topic_7+topic_8+topic_9,
                        data = to_regress_topic)

stargazer::stargazer(model_topic_price,
          title="Regression Results",
          align=TRUE,
          type="text",
          covariate.labels = c("Service Estimation", "EoY License Expense", "
Sales Estimation Comparisons", "Asset Investment", "Sales", "Business Licensi
ng Expense", "Asset Expense Estimation","Customer Service Expense", "Customer
Sales Results"),
          omit.stat=c("LL","ser","f"))

##
```

```
## Regression Results
## ========================================================
##                                    Dependent variable:
##                                ----------------------------
##                                       price_adj_ratio
## --------------------------------------------------------
## Service Estimation                          0.009
##                                            (0.019)
##
## EoY License Expense                         0.013
##                                            (0.020)
##
## Sales Estimation Comparisons                0.017
##                                            (0.019)
##
## Asset Investment                            0.012
##                                            (0.018)
##
## Sales                                       0.022
##                                            (0.018)
##
## Business Licensing Expense                  0.012
##                                            (0.018)
##
## Asset Expense Estimation                    0.010
##                                            (0.017)
##
## Customer Service Expense                    0.026
##                                            (0.017)
##
## Customer Sales Results
##
##
## Constant                                   -0.012
##                                            (0.013)
##
## --------------------------------------------------------
## Observations                                 176
## R2                                          0.019
## Adjusted R2                                -0.028
## ========================================================
## Note:                          *p<0.1; **p<0.05; ***p<0.01
```

It appears that none of the topics affecting the change in stock price (p-values >.05).

Checking for LDA

```
# Filtering to retain only verbs, nouns and adjectives
mda_filtered <- tokens%>% filter(upos %in% c("VERB", "NOUN", "ADJ"))
mda_filtered <- mda_filtered %>% select(Accession_Number, price_adj_ratio, le
mma, price_adj_ratio)

# Casting dtm
mda_dtm <- mda_filtered %>% count(Accession_Number, lemma) %>% cast_dtm(Acces
sion_Number, lemma,n)

mda_sel_idx <- slam::row_sums(mda_dtm) > 0
mda_dtm <- mda_dtm[mda_sel_idx, ]

# Computing LDA model
LDA_mda <- topicmodels::LDA(mda_dtm, k=10, method="Gibbs", control = list(see
d= 1234))

# Obtaining the per-word-per-topic probability
topics_mda <- tidy(LDA_mda,matrix="beta")
topics_mda
```

```
## # A tibble: 52,460 x 3
##    topic term          beta
##    <int> <chr>        <dbl>
##  1     1 ability 0.000692
##  2     2 ability 0.00248
##  3     3 ability 0.00185
##  4     4 ability 0.000671
##  5     5 ability 0.00128
##  6     6 ability 0.000572
##  7     7 ability 0.000605
##  8     8 ability 0.00150
##  9     9 ability 0.000741
## 10    10 ability 0.00000194
## # … with 52,450 more rows
```

```
saveRDS(topics_mda, "topics_mda.rds")



topics_mda <- readRDS("topics_mda.rds")
```

Conclusions

From topic modelling we have provided an analysis of topics that dominates the corpus using both unsupervised and supervised approaches. We have also discussed an optimum number of topics (Kappa, K) that best represent the topic model. Lastly, we also showed that none of the topics & topics additive are significant to predict stock price.

References

Robert et. al., (2016). A Model of Text for Experimentation in the Social Sciences https://www.tandfonline.com/doi/abs/10.1080/01621459.2016.1141684

Hadley Wickham (2021). rvest: Easily Harvest (Scrape) Web Pages. https://rvest.tidyverse.org/, https://github.com/tidyverse/rvest.

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL https://www.jstatsoft.org/v40/i03/.

Silge J, Robinson D (2016). "tidytext: Text Mining and Analysis Using Tidy Data Principles in R." _JOSS_, *1*(3). doi: 10.21105/joss.00037 (URL:https://doi.org/10.21105/joss.00037), <URL:http://dx.doi.org/10.21105/joss.00037>.

Silge, J., & Robinson, D. (2017). Text mining with R: A tidy approach. " O'Reilly Media, Inc.".

Jeroen Ooms (2021). cld3: Google's Compact Language Detector 3. https://docs.ropensci.org/cld3/, https://github.com/ropensci/cld3 (devel) https://github.com/google/cld3 (upstream).

Kurt Hornik (2020). NLP: Natural Language Processing Infrastructure. R package version 0.2-1.

Ingo Feinerer and Kurt Hornik (2020). tm: Text Mining Package. R package version 0.7-8. http://tm.r-forge.r-project.org/

Ingo Feinerer, Kurt Hornik, and David Meyer (2008). Text Mining Infrastructure in R. Journal of Statistical Software 25(5): 1-54. URL: https://www.jstatsoft.org/v25/i05/.

Rinker, T. W. (2020). qdap: Quantitative Discourse Analysis Package. 2.4.2. Buffalo, New York. https://github.com/trinker/qdap

Hadley Wickham, Jim Hester and Winston Chang (2021). devtools: Tools to Make Developing R Packages Easier. https://devtools.r-lib.org/, https://github.com/r-lib/devtools.

Rinker, T. W. (2018). textclean: Text Cleaning Tools version 0.9.3. Buffalo, New York. https://github.com/trinker/textclean

Rinker, T. W. (2018). textstem: Tools for stemming and lemmatizing text version 0.1.4. Buffalo, New York. http://github.com/trinker/textstem

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Jeroen Ooms (2020). hunspell: High-Performance Stemmer, Tokenizer, and Spell Checker. https://docs.ropensci.org/hunspell/ (docs), https://github.com/ropensci/hunspell (devel), https://hunspell.github.io (upstream).

Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3.

David Robinson (2020). widyr: Widen, Process, then Re-Tidy Data. R package version 0.1.3. http://github.com/dgrtwo/widyr

Jan Wijffels (2020). udpipe: Tokenization, Part of Speech Tagging, Lemmatization and Dependency Parsing with the 'UDPipe' 'NLP' Toolkit. https://bnosac.github.io/udpipe/en/index.html, https://github.com/bnosac/udpipe.

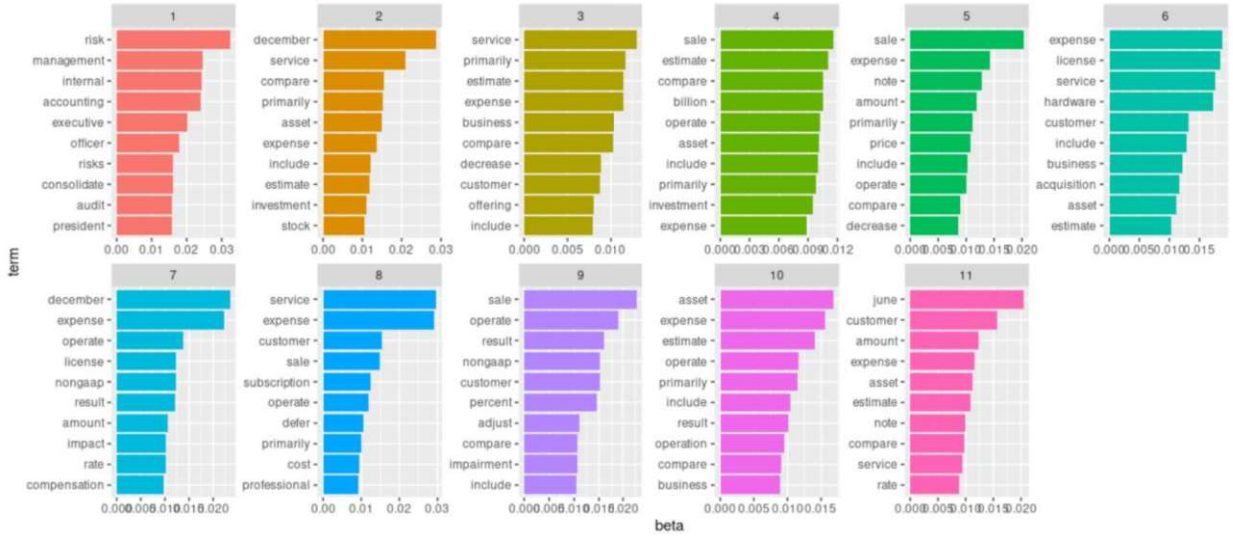
Emil Hvitfeldt (2020). textdata: Download and Load Various Text Datasets. R package version 0.4.1. https://github.com/EmilHvitfeldt/textdata

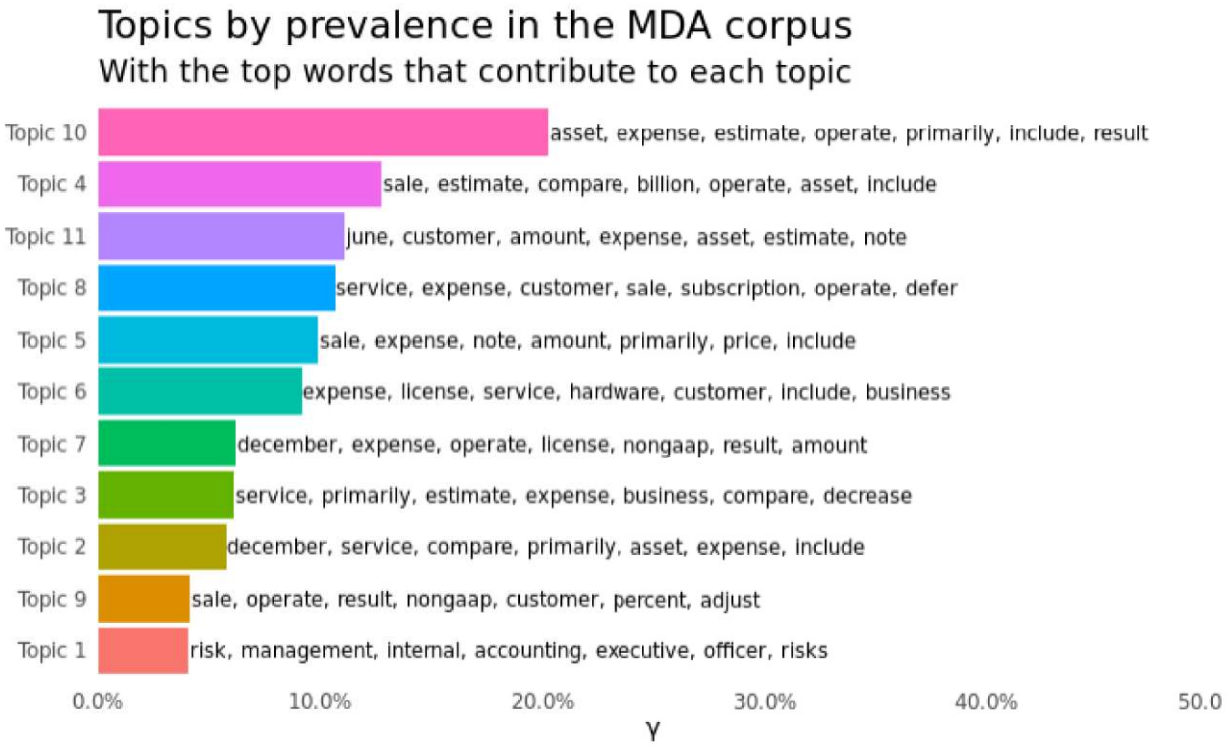
Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.

Appendix

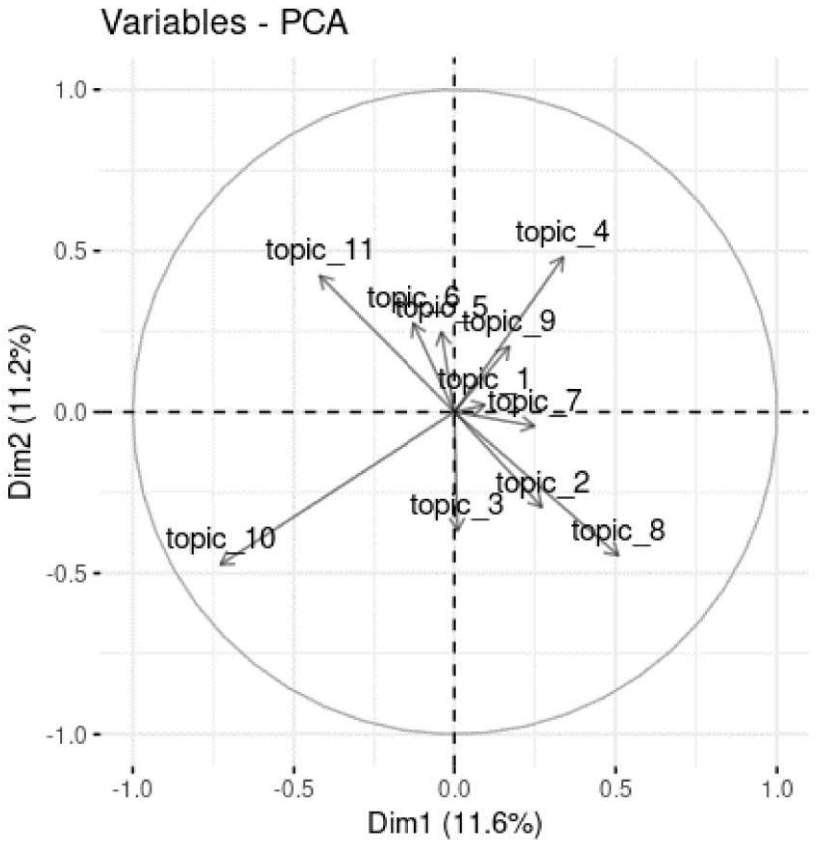
Part C

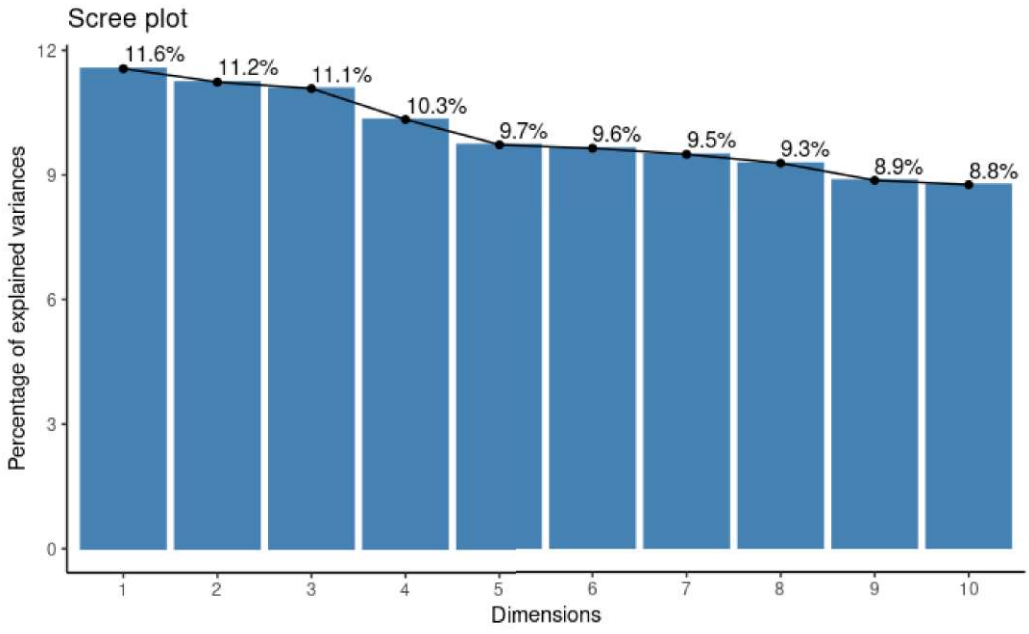
K = 11

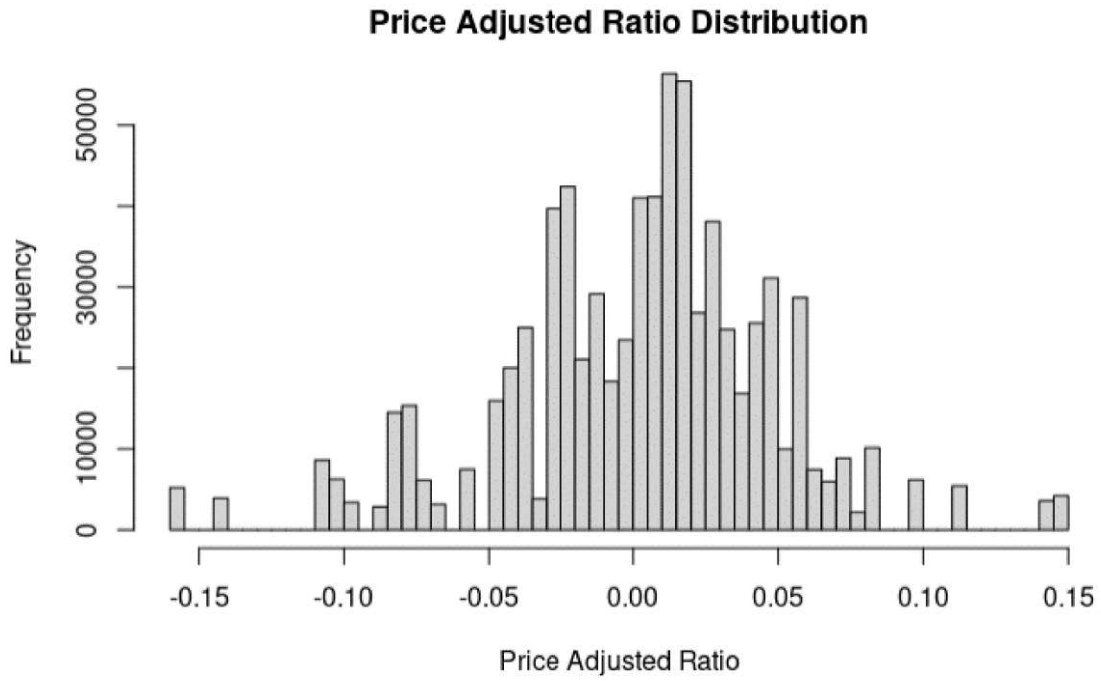


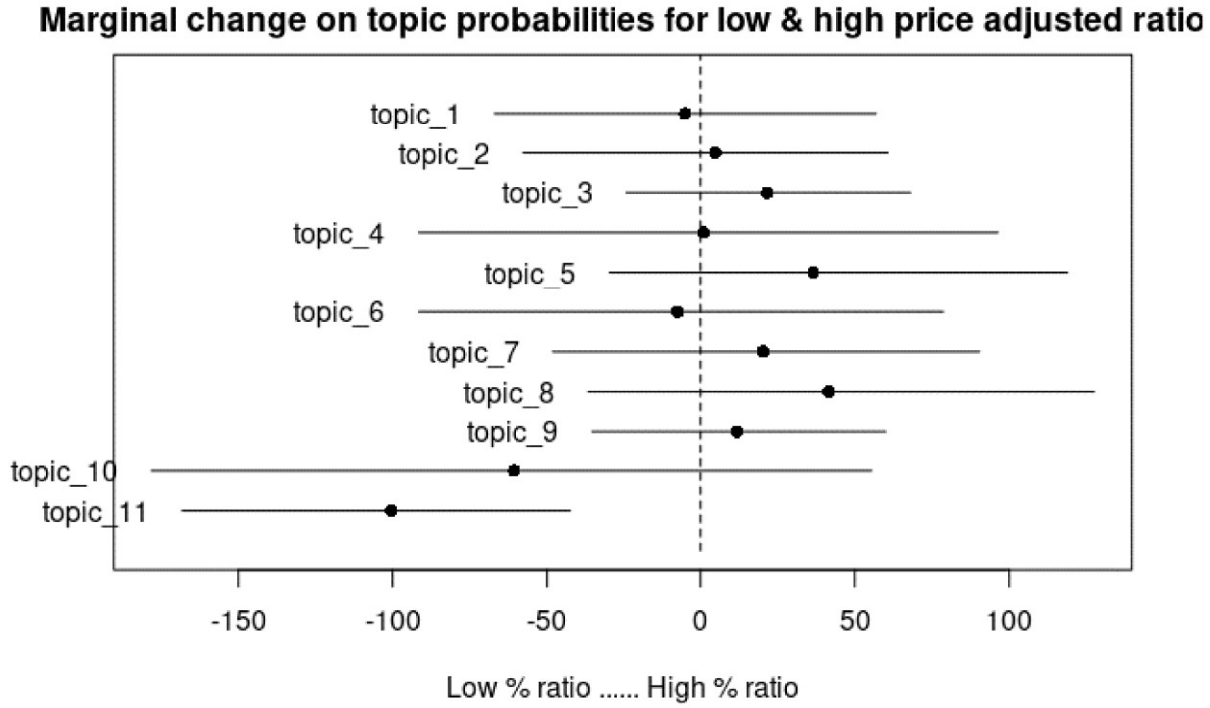


Topic	Expected topic proportion	Top 7 terms
Topic 10	0.203	asset, expense, estimate, operate, primarily, include, result
Topic 4	0.128	sale, estimate, compare, billion, operate, asset, include
Topic 11	0.111	june, customer, amount, expense, asset, estimate, note
Topic 8	0.106	service, expense, customer, sale, subscription, operate, defer
Topic 5	0.099	sale, expense, note, amount, primarily, price, include
Topic 6	0.091	expense, license, service, hardware, customer, include, business
Topic 7	0.062	december, expense, operate, license, nongaap, result, amount
Topic 3	0.061	service, primarily, estimate, expense, business, compare, decrease
Topic 2	0.058	december, service, compare, primarily, asset, expense, include
Topic 9	0.041	sale, operate, result, nongaap, customer, percent, adjust
Topic 1	0.041	risk, management, internal, accounting, executive, officer, risks


Variables - PCA
topic_11
topic_4
topic_6
topic_5
topic_9
topic_1
topic_7
topic_2
topic_3
topic_8
topic_10
Dim2 (11.2%)
Dim1 (11.6%)
1.0
0.5
0.0
-0.5
-1.0


Scree plot
11.6%
11.2%
11.1%
10.3%
9.7%
9.6%
9.5%
9.3%
8.9%
8.8%
Percentage of explained variances
Dimensions
12
9
6
3
0
1
2
3
4
5
6
7
8
9
10


Price Adjusted Ratio Distribution
Frequency
0
10000
30000
50000
-0.15
-0.10
-0.05
0.00
0.05
0.10
0.15
Price Adjusted Ratio


Marginal change on topic probabilities for low & high price adjusted ratio
topic_1
topic_2
topic_3
topic_4
topic_5
topic_6
topic_7
topic_8
topic_9
topic_10
topic_11
-150
-100
-50
0
50
100
Low % ratio ..... High % ratio