Mining company reviews on Glassdoor

Liudmyla Kotusenko

March 2020

1. Loading necessary packages

```
library(tidyverse)
library(dplyr)
library(scales)
library(knitr)
library(qdap) # for TM
library(tm) # for TM
library(RWeka)
library("purrr")
library("rJava")
library(ggplot2) # to make neat plots
library("FactoMineR") # to run correspondence analysis (CA)
library("factoextra") # to plot results from CA
library("wordcloud") # to build word coulds
library("viridisLite") # to change color in word clouds
library("plotrix") # to make piramid plots
library(corrplot) # to visualize chi-square residuals matrix
library(topicmodels) # to run LDA (Latent Dirichlet Allocation)
library(tidytext)


# to resolve issues with qdap and RWeka related to Java:
# 1. install Java 64-bit (the same as R version)
# 2. Run this line (ensure the path is correct):
# Sys.setenv(JAVA_HOME="C:/Program Files/Java/jre1.8.0_241")
```

2. Exploring the data set

```
data <- read.csv("~/HEC/!Text and network mining/Team project/Glassdoor/data.csv",
                 stringsAsFactors=FALSE)
```

Let's briefly explore our data…

```
names(data)
```

```
## [1] "Company" "recom"   "pros"    "cons"    "sector"
```

Let's explore how many reviews per company (sector) we have and share of those who recommend a given company (sector).

```
options(digits=4)
tab = data %>%
  group_by(Company) %>%
  summarize(n=n(), share_recom = mean(recom, na.rm = TRUE)*100)
tab
```

```
## # A tibble: 14 x 3
##    Company          n share_recom
##    <chr>        <int>       <dbl>
##  1 Bell           500        67.4
##  2 BMO            500        68.9
```

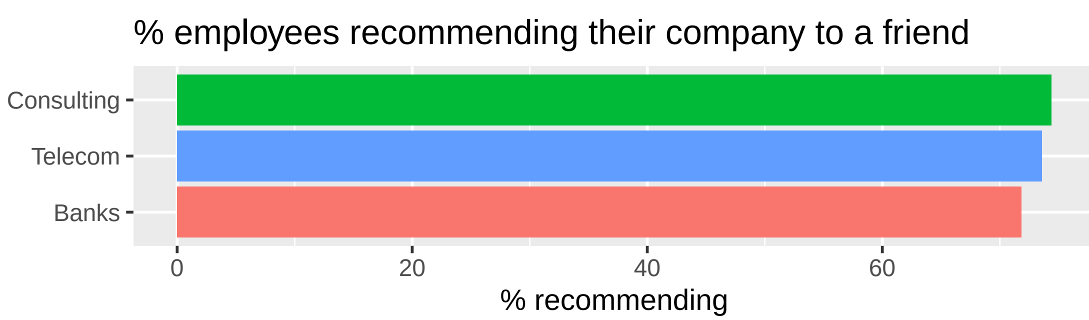
```
##   3 CIBC           500         67.8
##   4 Deloitte       500         73.3
##   5 EY             374         78.2
##   6 KPMG           438         73.4
##   7 NationalBank   500         69.1
##   8 PwC            500         73.5
##   9 RBC            500         78.5
##  10 Rogers         500         75.3
##  11 Scotiabank     500         65.2
##  12 TD             500         79.8
##  13 Telus          500         78.4
##  14 Videotron      119         71.7
```

```
tab2 = data %>%
  group_by(sector) %>%
  summarize(n=n(), share_recom = mean(recom, na.rm = TRUE)*100)
tab2
```

```
## # A tibble: 3 x 3
##   sector          n share_recom
##   <chr>       <int>       <dbl>
## 1 Banks        3000        71.9
## 2 Consulting   1812        74.4
## 3 Telecom      1619        73.6
```

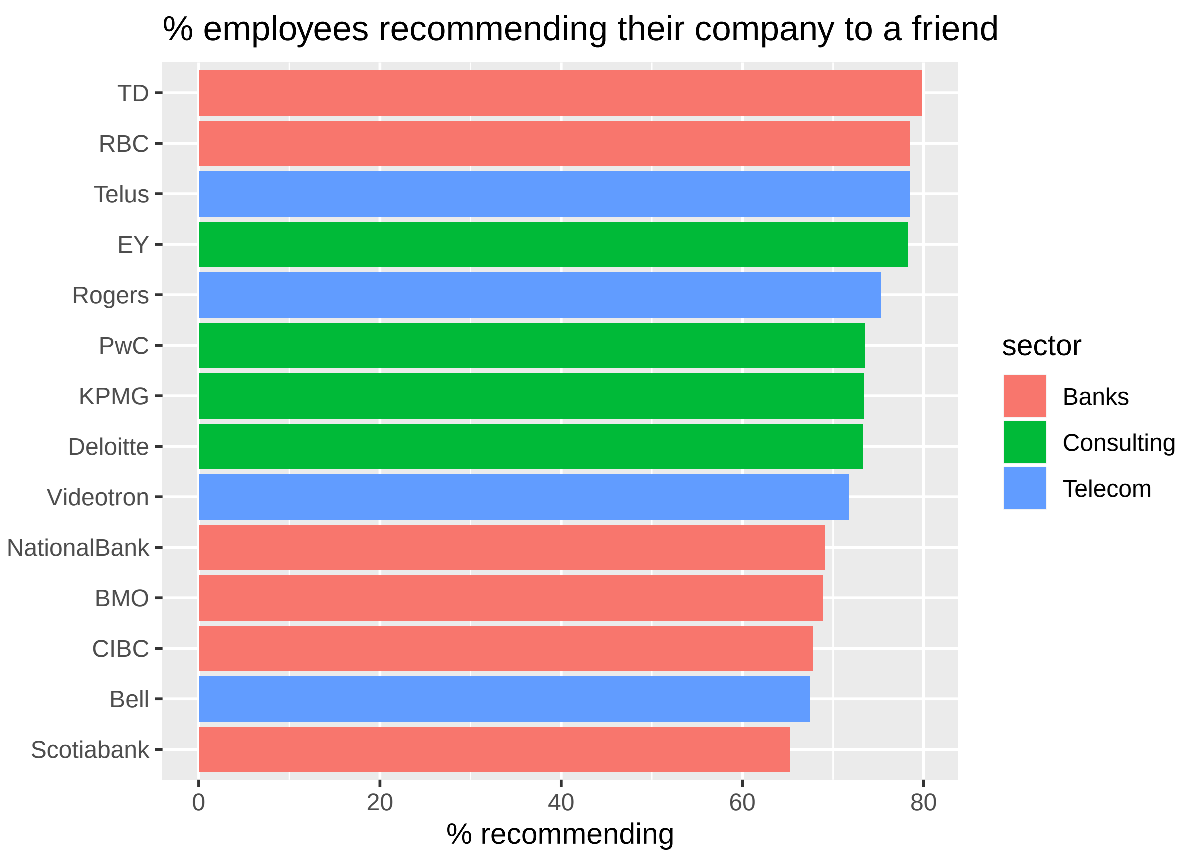
Plotting % people recommending their sectors

```
data %>%
  group_by(sector) %>%
  summarize(n=n(), share_recom = mean(recom, na.rm = TRUE)*100) %>%
  arrange(desc(share_recom)) %>%
  ggplot(aes(x=reorder(sector, share_recom), y=share_recom, fill=sector)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  ggtitle("% employees recommending their company to a friend") +
  xlab("") + ylab("% recommending")
```



Plotting % people recommending their companies

```
data %>%
  group_by(Company,sector) %>%
  summarize(n=n(), share_recom = mean(recom, na.rm = TRUE)*100) %>%
  arrange(desc(share_recom)) %>%
  ggplot(aes(x=reorder(Company, share_recom), y=share_recom, fill=sector)) +
  geom_col() +
  # facet_wrap(sector ~.) +
  coord_flip() +
  ggtitle("% employees recommending their company to a friend") +
  xlab("") + ylab("% recommending")
```



3. Creating the stopwords list

There are several lists of stopwords in R. Let's look at them.

```
sort(Top200Words) # explore Top200Words from qdapDictionaries package
```

```
##   [1] "a"         "about"     "after"     "again"     "air"       "all"
##   [7] "also"      "America"   "an"        "and"       "animal"    "another"
##  [13] "answer"    "any"       "are"       "around"    "as"        "ask"
##  [19] "at"        "away"      "back"      "be"        "because"   "been"
##  [25] "before"    "big"       "boy"       "but"       "by"        "call"
##  [31] "came"      "can"       "change"    "come"      "could"     "day"
##  [37] "did"       "different" "do"        "does"      "down"      "each"
##  [43] "end"       "even"      "find"      "first"     "follow"    "for"
##  [49] "form"      "found"     "from"      "get"       "give"      "go"
##  [55] "good"      "great"     "had"       "hand"      "has"       "have"
##  [61] "he"        "help"      "her"       "here"      "him"       "his"
##  [67] "home"      "house"     "how"       "I"         "if"        "in"
##  [73] "into"      "is"        "it"        "its"       "just"      "kind"
##  [79] "know"      "land"      "large"     "learn"     "letter"    "like"
##  [85] "line"      "little"    "live"      "long"      "look"      "made"
##  [91] "make"      "man"       "many"      "may"       "me"        "mean"
##  [97] "men"       "more"      "most"      "mother"    "move"      "much"
## [103] "must"      "my"        "name"      "need"      "new"       "no"
## [109] "not"       "now"       "number"    "of"        "off"       "oil"
## [115] "old"       "on"        "one"       "only"      "or"        "other"
## [121] "our"       "out"       "over"      "page"      "part"      "people"
## [127] "picture"   "place"     "play"      "point"     "put"       "read"
## [133] "right"     "said"      "same"      "say"       "see"       "sentence"
## [139] "set"       "she"       "should"    "show"      "small"     "so"
```

```
## [145] "some"      "sound"     "spell"     "still"     "study"     "such"
## [151] "take"      "tell"      "than"      "that"      "the"       "their"
## [157] "them"      "then"      "there"     "these"     "they"      "thing"
## [163] "think"     "this"      "three"     "through"   "time"      "to"
## [169] "too"       "try"       "turn"      "two"       "up"        "us"
## [175] "use"       "very"      "want"      "was"       "water"     "way"
## [181] "we"        "well"      "went"      "were"      "what"      "when"
## [187] "where"     "which"     "who"       "why"       "will"      "with"
## [193] "word"      "work"      "world"     "would"     "write"     "year"
## [199] "you"       "your"
```

```r
sort(tm::stopwords("english"))
```

```
##   [1] "a"          "about"      "above"      "after"      "again"
##   [6] "against"    "all"        "am"         "an"         "and"
##  [11] "any"        "are"        "aren't"     "as"         "at"
##  [16] "be"         "because"    "been"       "before"     "being"
##  [21] "below"      "between"    "both"       "but"        "by"
##  [26] "can't"      "cannot"     "could"      "couldn't"   "did"
##  [31] "didn't"     "do"         "does"       "doesn't"    "doing"
##  [36] "don't"      "down"       "during"     "each"       "few"
##  [41] "for"        "from"       "further"    "had"        "hadn't"
##  [46] "has"        "hasn't"     "have"       "haven't"    "having"
##  [51] "he"         "he'd"       "he'll"      "he's"       "her"
##  [56] "here"       "here's"     "hers"       "herself"    "him"
##  [61] "himself"    "his"        "how"        "how's"      "i"
##  [66] "i'd"        "i'll"       "i'm"        "i've"       "if"
##  [71] "in"         "into"       "is"         "isn't"      "it"
##  [76] "it's"       "its"        "itself"     "let's"      "me"
##  [81] "more"       "most"       "mustn't"    "my"         "myself"
##  [86] "no"         "nor"        "not"        "of"         "off"
##  [91] "on"         "once"       "only"       "or"         "other"
##  [96] "ought"      "our"        "ours"       "ourselves"  "out"
## [101] "over"       "own"        "same"       "shan't"     "she"
## [106] "she'd"      "she'll"     "she's"      "should"     "shouldn't"
## [111] "so"         "some"       "such"       "than"       "that"
## [116] "that's"     "the"        "their"      "theirs"     "them"
## [121] "themselves" "then"       "there"      "there's"    "these"
## [126] "they"       "they'd"     "they'll"    "they're"    "they've"
## [131] "this"       "those"      "through"    "to"         "too"
## [136] "under"      "until"      "up"         "very"       "was"
## [141] "wasn't"     "we"         "we'd"       "we'll"      "we're"
## [146] "we've"      "were"       "weren't"    "what"       "what's"
## [151] "when"       "when's"     "where"      "where's"    "which"
## [156] "while"      "who"        "who's"      "whom"       "why"
## [161] "why's"      "with"       "won't"      "would"      "wouldn't"
## [166] "you"        "you'd"      "you'll"     "you're"     "you've"
## [171] "your"       "yours"      "yourself"   "yourselves"
```

The first list seems a bit too extensive while the latter does not include some annoying words. Let's add a few words from the first to the second list.

```r
# looking at stopwords in tm::stopwords but not in Top200Words
diff1 = anti_join(as.tibble(tm::stopwords("english")),
                  as.tibble(Top200Words))$value
```

```
## Warning: `as.tibble()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
```

```
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```r
sort(diff1)
```

```
##  [1] "above"      "against"    "am"         "aren't"     "being"
##  [6] "below"      "between"    "both"       "can't"      "cannot"
## [11] "couldn't"   "didn't"     "doesn't"    "doing"      "don't"
## [16] "during"     "few"        "further"    "hadn't"     "hasn't"
## [21] "haven't"    "having"     "he'd"       "he'll"      "he's"
## [26] "here's"     "hers"       "herself"    "himself"    "how's"
## [31] "i"          "i'd"        "i'll"       "i'm"        "i've"
## [36] "isn't"      "it's"       "itself"     "let's"      "mustn't"
## [41] "myself"     "nor"        "once"       "ought"      "ours"
## [46] "ourselves"  "own"        "shan't"     "she'd"      "she'll"
## [51] "she's"      "shouldn't"  "that's"     "theirs"     "themselves"
## [56] "there's"    "they'd"     "they'll"    "they're"    "they've"
## [61] "those"      "under"      "until"      "wasn't"     "we'd"
## [66] "we'll"      "we're"      "we've"      "weren't"    "what's"
## [71] "when's"     "where's"    "while"      "who's"      "whom"
## [76] "why's"      "won't"      "wouldn't"   "you'd"      "you'll"
## [81] "you're"     "you've"     "yours"      "yourself"   "yourselves"
```

```r
# looking at stopwords in Top200Words but not in tm::stopwords
diff2 = anti_join(as.tibble(Top200Words),
                  as.tibble(tm::stopwords("english")))$value
sort(diff2)
```

```
##   [1] "air"       "also"      "America"   "animal"    "another"   "answer"
##   [7] "around"    "ask"       "away"      "back"      "big"       "boy"
##  [13] "call"      "came"      "can"       "change"    "come"      "day"
##  [19] "different" "end"       "even"      "find"      "first"     "follow"
##  [25] "form"      "found"     "get"       "give"      "go"        "good"
##  [31] "great"     "hand"      "help"      "home"      "house"     "I"
##  [37] "just"      "kind"      "know"      "land"      "large"     "learn"
##  [43] "letter"    "like"      "line"      "little"    "live"      "long"
##  [49] "look"      "made"      "make"      "man"       "many"      "may"
##  [55] "mean"      "men"       "mother"    "move"      "much"      "must"
##  [61] "name"      "need"      "new"       "now"       "number"    "oil"
##  [67] "old"       "one"       "page"      "part"      "people"    "picture"
##  [73] "place"     "play"      "point"     "put"       "read"      "right"
##  [79] "said"      "say"       "see"       "sentence"  "set"       "show"
##  [85] "small"     "sound"     "spell"     "still"     "study"     "take"
##  [91] "tell"      "thing"     "think"     "three"     "time"      "try"
##  [97] "turn"      "two"       "us"        "use"       "want"      "water"
## [103] "way"       "well"      "went"      "will"      "word"      "work"
## [109] "world"     "write"     "year"
```

```r
# list of stopwords from Top200Words to add to tm list
(add = sort(diff2)[c(2,5,7,9,10,15,21,27,30,31,36:38,44,50:51,54:55,59,62,68,88,104,106)])
```

```
##  [1] "also"    "another" "around"  "away"    "back"    "can"     "even"
##  [8] "get"     "good"    "great"   "I"       "just"    "kind"    "like"
## [15] "made"    "make"    "may"     "mean"    "much"    "need"    "one"
## [22] "still"   "well"    "will"
```

```r
# creating our own list of stopwords to exclude
stop.words = c(c(tm::stopwords("english"), add, "job", "work", "really", "always", "sometimes",
                 "deloitte", "bmo", "bell", "kpmg", "rbc", "rogers", "telus", "videotron",
                 "td", "national", "bank", "cibc")) #
```

```r
sort(stop.words) # explore the full list of stopwords
```

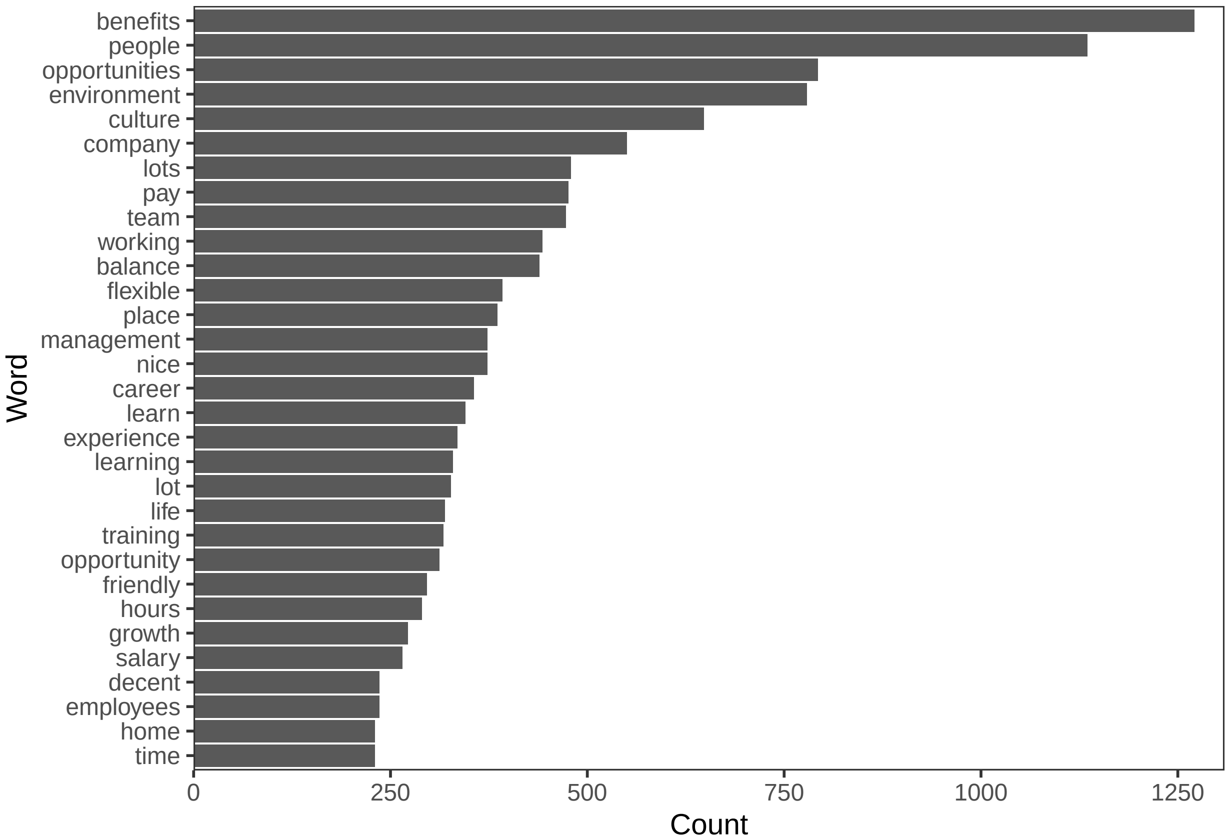
```
##   [1] "a"           "about"       "above"       "after"       "again"
##   [6] "against"     "all"         "also"        "always"      "am"
##  [11] "an"          "and"         "another"     "any"         "are"
##  [16] "aren't"      "around"      "as"          "at"          "away"
##  [21] "back"        "bank"        "be"          "because"     "been"
##  [26] "before"      "being"       "bell"        "below"       "between"
##  [31] "bmo"         "both"        "but"         "by"          "can"
##  [36] "can't"       "cannot"      "cibc"        "could"       "couldn't"
##  [41] "deloitte"    "did"         "didn't"      "do"          "does"
##  [46] "doesn't"     "doing"       "don't"       "down"        "during"
##  [51] "each"        "even"        "few"         "for"         "from"
##  [56] "further"     "get"         "good"        "great"       "had"
##  [61] "hadn't"      "has"         "hasn't"      "have"        "haven't"
##  [66] "having"      "he"          "he'd"        "he'll"       "he's"
##  [71] "her"         "here"        "here's"      "hers"        "herself"
##  [76] "him"         "himself"     "his"         "how"         "how's"
##  [81] "i"           "I"           "i'd"         "i'll"        "i'm"
##  [86] "i've"        "if"          "in"          "into"        "is"
##  [91] "isn't"       "it"          "it's"        "its"         "itself"
##  [96] "job"         "just"        "kind"        "kpmg"        "let's"
## [101] "like"        "made"        "make"        "may"         "me"
## [106] "mean"        "more"        "most"        "much"        "mustn't"
## [111] "my"          "myself"      "national"    "need"        "no"
## [116] "nor"         "not"         "of"          "off"         "on"
## [121] "once"        "one"         "only"        "or"          "other"
## [126] "ought"       "our"         "ours"        "ourselves"   "out"
## [131] "over"        "own"         "rbc"         "really"      "rogers"
## [136] "same"        "shan't"      "she"         "she'd"       "she'll"
## [141] "she's"       "should"      "shouldn't"   "so"          "some"
## [146] "sometimes"   "still"       "such"        "td"          "telus"
## [151] "than"        "that"        "that's"      "the"         "their"
## [156] "theirs"      "them"        "themselves"  "then"        "there"
## [161] "there's"     "these"       "they"        "they'd"      "they'll"
## [166] "they're"     "they've"     "this"        "those"       "through"
## [171] "to"          "too"         "under"       "until"       "up"
## [176] "very"        "videotron"   "was"         "wasn't"      "we"
## [181] "we'd"        "we'll"       "we're"       "we've"       "well"
## [186] "were"        "weren't"     "what"        "what's"      "when"
## [191] "when's"      "where"       "where's"     "which"       "while"
## [196] "who"         "who's"       "whom"        "why"         "why's"
## [201] "will"        "with"        "won't"       "work"        "would"
## [206] "wouldn't"    "you"         "you'd"       "you'll"      "you're"
## [211] "you've"      "your"        "yours"       "yourself"    "yourselves"
```

```r
# # We could also remove these words but let's keep them for now.
# stop.words = c(stop.words, "lot", "lots", "like", "always", "sometimes",
#                "years", "times") #  "high", "low", "bad",

# create right away stopwords list for bi-gram treatment - we'll keep "work":
stop.words2 = stop.words[-200]
```

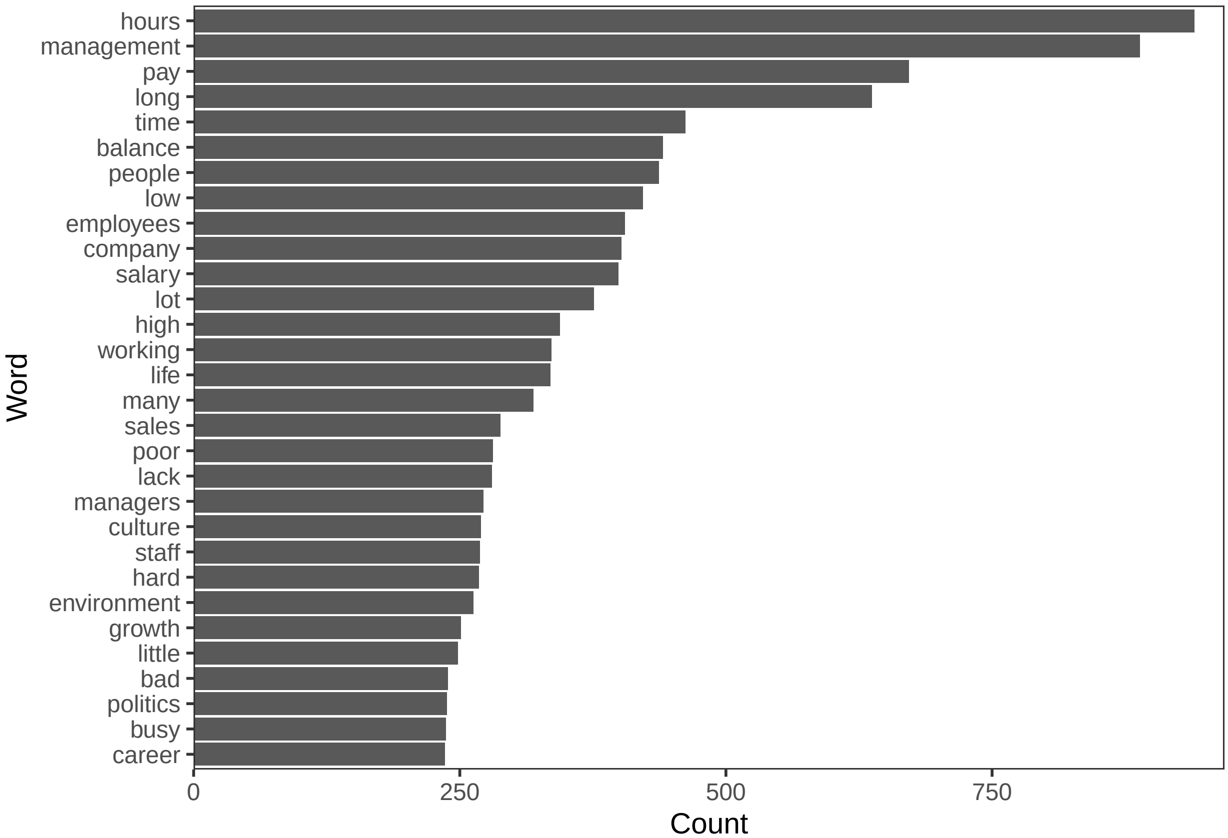
4. Getting the first quick-and-dirty plots

Using the function `freq_terms` from the `qdap` package, we can make quick-and-drity plots with frequent pros and cons. We remove the stopwords and keep words with 3+ letters, and the function will remove punctuation. Plots show counts of pros and cons overall, across all companies/sectors.

```
plot(freq_terms(data$pros, top=30, at.least=3, stopwords=stop.words)) # plot top-30 pros
```




```
plot(freq_terms(data$cons, top=30, at.least=3, stopwords=stop.words)) # plot top-30 cons
```



5. Correspondence analysis on most frequent terms

Let's see what we can get with stopword filtration and picking top-30 most frequent words across all companies, using the same `freq_terms` function. Correspondence analysis, widely used in market research, can help us get plots mapping companies along with the comments most associated to them. It takes as an input a 2-way table with terms in rows, companies in columns (in our case, but vise versa would also work) and raw frequencies (counts) of words per company in cells.

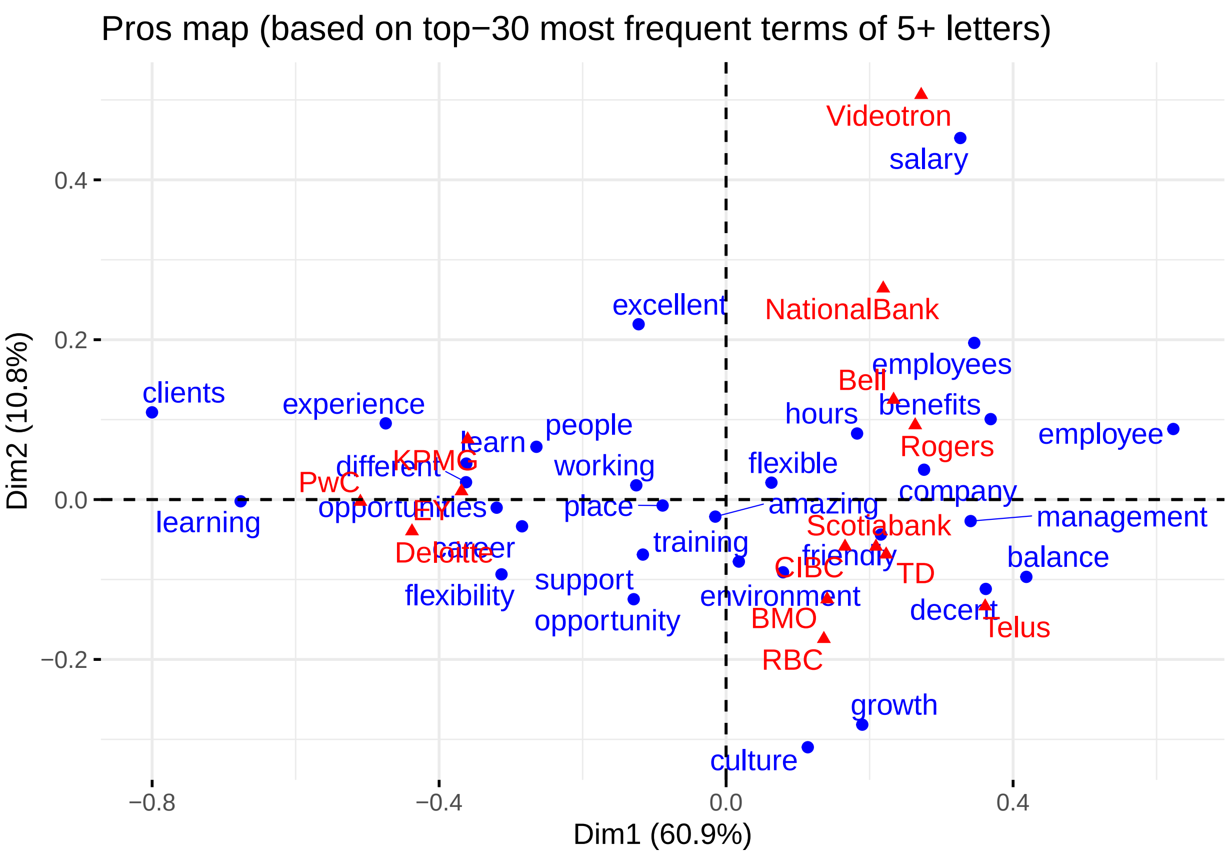
To make plots, we create a function that will take as an input several columns from a dataframe, number of terms to show, min.number of letters in a term and a stopwords list and produce a correspondence analysis plot.

```
# Create a function to produce CA plots
# input to the function: dataframe with columns named "pros", "cons", "Company"
CA_plot = function(terms.col=data$pros, # column with comments (pros or cons)
                   comp.col=data$Company, # column with companies (or sectors)
                   topNterms=20, # number of top-N terms to show, 20 by default
                   minLettersInTerm=5, # min number of letters in a term to keep
                   stopwords=stop.words) {

  company = unique(comp.col)

  # create df of most frequent terms from the whole corpus
  pros = as.data.frame(freq_terms(terms.col, top=topNterms,
                                  at.least=minLettersInTerm,
                                  stopwords=stopwords))
  pros.l = list()

  # create a list containing a dataframe with most frequent terms per each company
  # we take top-100 terms per company to ensure most frequent terms IN TOTAL are there
  for (i in company) {
```

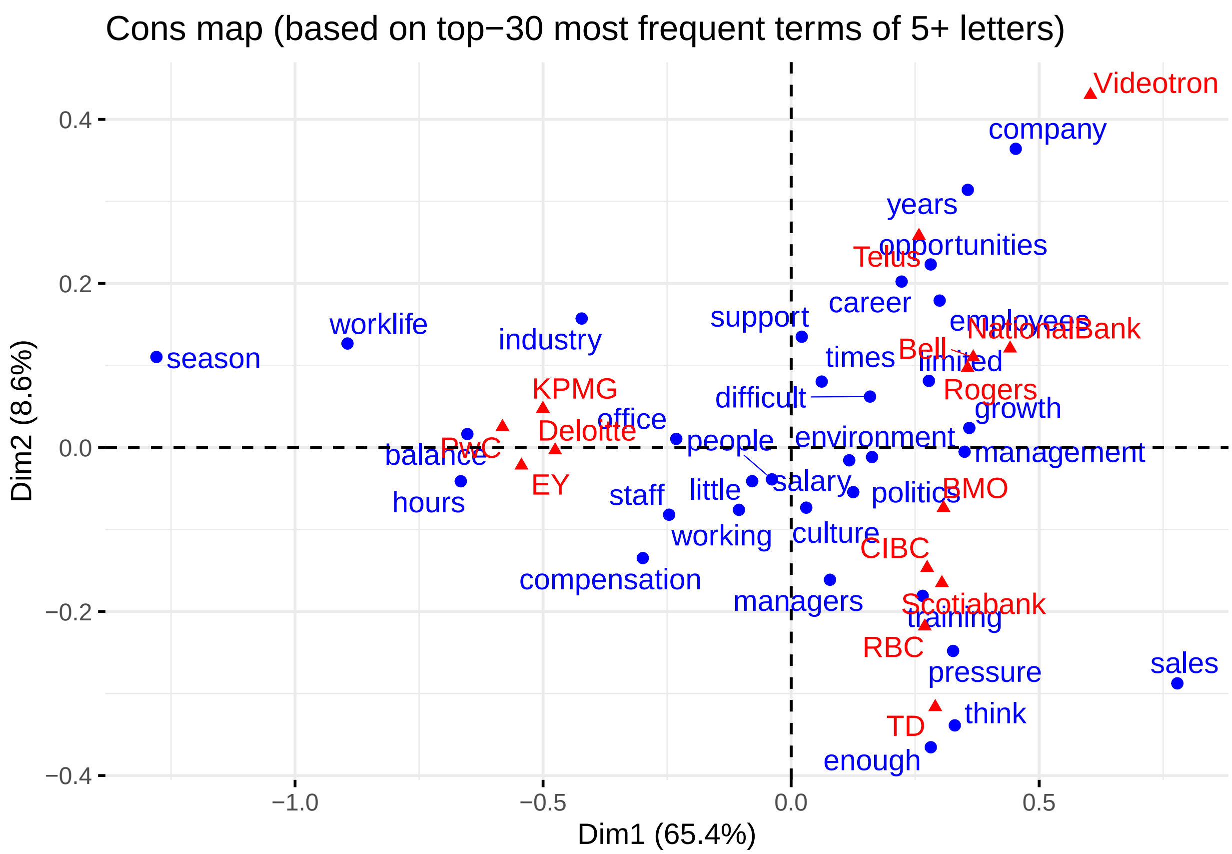
```
    pros.l[[i]] = as.data.frame(freq_terms(terms.col[which(comp.col==i)],
                                            top=100, at.least=minLettersInTerm,
                                            stopwords=stopwords))
  }

  # flatten our list to a dataframe
  prosTop100 = list_df2df(pros.l, col1 = "Company")

  prosTopX = prosTop100 %>%
    filter(WORD %in% pros$WORD)

  # Using tidyr::pivot_wider to convert long data to wide
  # to have words in rows, companies in cols and term frequencies (raw) in cells
  prosTopXw = pivot_wider(prosTopX, names_from = Company,
                          values_from = FREQ)

  # to use CA() function, we need terms in rownames and not in a separate column
  prosTopXw = as.data.frame(prosTopXw) # convert tibble to dataframe
  prosTopXw[is.na(prosTopXw)]=0 # replace NA with 0
  rownames(prosTopXw) = prosTopXw$WORD # set terms to rownames
  prosTopXw = prosTopXw[,-1] # remove the first column with terms
  # print(prosTopXw) # print a two-way table of raw frequencies
  pros.ca <- CA(prosTopXw, graph = FALSE) # run correspondence analysis

  # make a CA plot
  # repel= TRUE to avoid text overlapping (slow if many points)
  fviz_ca_biplot(pros.ca, repel = TRUE)
}
```

Using the function just created, we can make CA plots, easily changing the number of top-most frequent terms and the term length accepted. We can play around with it, but the solution below is not bad. It should be noted that projection of multi-dimentional space on 2d plot may not be ideal, and sometimes words and companies may be situated close to each other while there is no relationship between them in reality (if measured by chi-square residuals). However, in most cases, such plots show decent results and can provide intuitive maps.

```
set.seed(1234)
CA_plot(terms.col=data$pros, comp.col=data$Company,
                  topNterms=30, minLettersInTerm=5) +
  ggtitle("Pros map (based on top-30 most frequent terms of 5+ letters)")
```




```r
CA_plot(terms.col=data$cons, comp.col=data$Company,
                         topNterms=30, minLettersInTerm=5) +
  ggtitle("Cons map (based on top-30 most frequent terms of 5+ letters)")
```



6. Corpus creation and cleaning

Let's now delve into text mining and create our corpus and clean it. Accroding to the DataCamp, "if your text data is in a data frame you can use DataframeSource() for your analysis. The data frame passed to DataframeSource() must have a specific structure:

- Column one must be called doc_id and contain a unique string for each row.
- Column two must be called text with "UTF-8" encoding (pretty standard).
- Any other columns, 3+ are considered metadata and will be retained as such."

```
names(data)
```

```
## [1] "Company" "recom"   "pros"    "cons"    "sector"
```

```
# Create dataframes suitable for corpus creation
pros.df = data %>%
  mutate(doc_id = c(1:n())) %>%
  rename(text = pros) %>%
  select(doc_id, text, Company, sector)

cons.df = data %>%
  mutate(doc_id = c(1:n())) %>%
  rename(text = cons) %>%
  select(doc_id, text, Company, sector)

# create volatile corpus for pros and cons
pros_corpus = VCorpus(DataframeSource(pros.df))
cons_corpus = VCorpus(DataframeSource(cons.df))

# explore corpus
pros_corpus[[1]][1] # to look at text
```

```
## $content
## [1] "Diverse and inclusive work culture ,  It's the people that makes this company Awesome!! Love workin
```

```
# meta(pros_corpus)[1] # to look at metadata (prints too much)
```

We can make a custom function for corpus cleaning that will remove extra white spaces, make everything lower case, filter stopwords, remove punctuation and stem the words. We would also like to complete stems to full words, however, **`stemCompletion()` function works with the term-document matrix (and not a corpus) so we will use it later in our analysis**.

Below, we create two functions for corpus cleaning: `clean_corpus()`, that includes stemming, and `clean_corpus2()` that does not include it. Most of the time in the analysis below, we will be using `clean_corpus()`.

```
# cleaning corpus based on tm package functions
clean_corpus <- function(corpus){
  corpus <- tm_map(corpus, stripWhitespace) # remove extra white spaces
  corpus <- tm_map(corpus, content_transformer(tolower)) # everything to lower case
  corpus <- tm_map(corpus, removeWords, stop.words) # filter stopwords
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, removeWords, stop.words) # some SW are very stubborn...
  corpus <- tm_map(corpus, stripWhitespace) # after stopword removal, new spaces are added
  corpus <- tm_map(corpus, stemDocument) # stem the document.
  return(corpus)
}

# the same function but without stemming
clean_corpus2 <- function(corpus){
  corpus <- tm_map(corpus, stripWhitespace) # remove extra white spaces
  corpus <- tm_map(corpus, content_transformer(tolower)) # everything to lower case
  corpus <- tm_map(corpus, removeWords, stop.words) # filter stopwords
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, removeWords, stop.words) # some SW are very stubborn...
  corpus <- tm_map(corpus, stripWhitespace) # after stopword removal, there are new spaces
  return(corpus)
}
```

Let's clean our corpora of pros and cons and compare one review before and after. For this comparison, we use a function that will stem our terms.

```
# with stemming
pros_corpus.cl = clean_corpus(pros_corpus)
cons_corpus.cl = clean_corpus(cons_corpus)

# without stemming
pros_corpus.cl2 = clean_corpus2(pros_corpus)
cons_corpus.cl2 = clean_corpus2(cons_corpus)

pros_corpus[[5]][1]
```

```
## $content
## [1] "Friendly co-workers and good mentor-ship dependent on your 'coach'."
```

```
pros_corpus.cl[[5]][1]
```

```
## $content
## [1] "friend cowork mentorship depend coach"
```

With the clean corpus, we can make our first word clouds of pros and cons across all companies. Let's not use stemming this time to make plots more readable.

```r
color_pal <- viridis(n = 7)
# without stemming
wordcloud(pros_corpus.cl2, max.words = 70, colors = color_pal) # pros
```

```
## Warning in wordcloud(pros_corpus.cl2, max.words = 70, colors = color_pal):
## people could not be fit on page. It will not be plotted.
```


```r
wordcloud(cons_corpus.cl2, max.words = 70, colors = color_pal) # cons
```

```
## Warning in wordcloud(cons_corpus.cl2, max.words = 70, colors = color_pal):
## management could not be fit on page. It will not be plotted.

## Warning in wordcloud(cons_corpus.cl2, max.words = 70, colors = color_pal): long
## could not be fit on page. It will not be plotted.
```

Next, we can create term-document and document-term matrices (TDM and DTM). We print TDM to explore the number of terms and the matrix sparcity. It's 100% for both matrices - our comments are quite short and each document covers just a tiny number of all terms in a corpus.

```
(pros_tdm <- TermDocumentMatrix(pros_corpus.cl))
```

```
## <<TermDocumentMatrix (terms: 2998, documents: 6431)>>
## Non-/sparse entries: 42894/19237244
## Sparsity           : 100%
## Maximal term length: 30
## Weighting          : term frequency (tf)
```

```
(cons_tdm <- TermDocumentMatrix(cons_corpus.cl))
```

```
## <<TermDocumentMatrix (terms: 4928, documents: 6431)>>
## Non-/sparse entries: 57618/31634350
## Sparsity           : 100%
## Maximal term length: 27
## Weighting          : term frequency (tf)
```

```
pros_dtm <- DocumentTermMatrix(pros_corpus.cl)
cons_dtm <- DocumentTermMatrix(cons_corpus.cl)
```

Using TDM, we can find correlations between some terms of interest. Here, I'm using stemmed terms, so it required some expreimentation/exploration before I found out what stems to use. Of course, if we don't stem, we will have to write "balance" instead of "balanc". For some terms, I set higher threshold for correlations because too much noise is returned.

```
# Some words from pros and their correlations with other words
findAssocs(pros_tdm, "balanc", 0.1)
```

```
## $balanc
## life
## 0.92
```

```r
findAssocs(pros_tdm, "life", 0.2)
```

```
## $life
## balanc
##   0.92
```

```r
findAssocs(pros_tdm, "decent", 0.1)
```

```
## $decent
##        pay    benefit    acclaim     expect findnegoti     salari
##       0.18       0.15       0.13       0.13       0.13       0.12
```

```r
findAssocs(pros_tdm, "pay", 0.1)
```

```
## $pay
##  decent benefit  averag   bonus
##    0.18    0.12    0.11    0.10
```

```r
# Some words from cons and their correlations with other words
findAssocs(cons_tdm, "balanc", 0.1)
```

```
## $balanc
##     life  worklif maintain sacrific   former     poor
##     0.87     0.19     0.12     0.12     0.10     0.10
```

```r
findAssocs(cons_tdm, "hour", 0.2)
```

```
## $hour
##   long season   busi
##   0.63   0.28   0.20
```

```r
findAssocs(cons_tdm, "long", 0.1)
```

```
## $long
##    hour  season    busi   login    vaca partner    term
##    0.63    0.24    0.18    0.12    0.12    0.10    0.10
```

```r
findAssocs(cons_tdm, "low", 0.2)
```

```
## $low
##    pay salari  moral
##   0.29   0.23   0.22
```

```r
findAssocs(cons_tdm, "pay", 0.25)
```

```
## $pay
##  low
## 0.29
```

```r
findAssocs(cons_tdm, "busi", 0.2)
```

```
## $busi
## season   hour
##    0.6    0.2
```

```r
findAssocs(cons_tdm, "season", 0.2)
```

```
## $season
## busi hour long
## 0.60 0.28 0.24
```

7. Factor analysis

We can see what people tend to talk about by applying the factor analysis. For it, we use the DTM (documents in rows, terms in columns) after having stemmed our terms. Before proceeding, we remove the most sparse terms and turn our DTM to a dataframe. Deciding how many terms to keep is rather art than science, and after some experimenting, the below specification of `sparse` parameter seems satisfactory.

```
pros_dtm1 <- removeSparseTerms(pros_dtm, sparse=0.98)
pros_dtm1.df = as.data.frame(as.matrix(pros_dtm1))

cons_dtm1 <- removeSparseTerms(cons_dtm, sparse=0.97)
cons_dtm1.df = as.data.frame(as.matrix(cons_dtm1))
```

We can run factor analysis now. After many options tried, these ones seem the best, though for cons we get a few factors that are difficult to interpret (they include too many keywords and are difficult to interpret). Notice that even though we concentrate on the most frequent terms, the factor analysis below explains only 15% and 22% of variance of terms in pros and cons, respectively, so inevitably we lose some less frequent terms. Also, in both cases, we fail to confirm the hypothesis that the number of factors we specified is sufficient (see low p-values in the outputs). Though this is not a good practice, we will keep these solutions for their interpretability.

```
# sink("factor analysis.out") # to save FA output
fa.pros = factanal(pros_dtm1.df,10,scores="regression")
print(fa.pros, sort=TRUE)
```

```
##
## Call:
## factanal(x = pros_dtm1.df, factors = 10, scores = "regression")
##
## Uniquenesses:
##     advanc       amaz     balanc    benefit        big      bonus      brand
##      0.950      0.968      0.005      0.788      0.967      0.940      0.977
##       busi     career     client   colleagu    compani     cowork     cultur
##      0.925      0.005      0.810      0.977      0.904      0.953      0.950
##     decent    develop     differ     divers    employe    environ      excel
##      0.879      0.901      0.829      0.986      0.944      0.905      0.978
##     experi    exposur       firm    flexibl     friend       grow     growth
##      0.944      0.787      0.944      0.294      0.913      0.948      0.905
##       help       home       hour   industri   interest      learn       life
##      0.911      0.930      0.825      0.849      0.927      0.851      0.153
##      locat        lot      manag       mani       move    network        new
##      0.986      0.861      0.840      0.937      0.924      0.945      0.968
##       nice      offic   opportun        pay      peopl      place profession
##      0.936      0.963      0.293      0.922      0.786      0.942      0.975
##    project     salari      smart      staff    support       team       time
##      0.940      0.934      0.869      0.976      0.829      0.938      0.954
##      train       work
##      0.972      0.925
##
## Loadings:
##            Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7 Factor8
## balanc      0.994
## life        0.917
## career              0.984
## flexibl                     0.836
## opportun            0.195           0.134   0.665  -0.114
## advanc              0.195
## amaz                                                0.151
## benefit                                                     0.449
## big
## bonus                                                       0.234
## brand
## busi                                0.215
```

```
## client                              0.421
## colleagu
## compani                                             0.144
## cowork
## cultur
## decent                                                      0.344
## develop             0.164                           0.165
## differ                              0.329   0.174   0.113
## divers
## employe                                             0.173   0.122
## environ                                             0.102
## excel
## experi                              0.215
## exposur                             0.450
## firm                                0.169                           0.115
## friend
## grow                                        0.218
## growth              0.145                   0.116
## help                                                0.238
## home                        0.237
## hour                        0.402
## industri                            0.377
## interest                                                            0.257
## learn                               0.169   0.305          -0.133
## locat
## lot                                         0.331                   0.106
## manag                                               0.382
## mani                                0.154   0.105                   0.121
## move                                        0.191
## network                             0.147   0.168
## new                                                 0.105           0.112
## nice                                                                0.118
## offic                                                               0.169
## pay                                                         0.266
## peopl                                                               0.438
## place                                                      -0.114
## profession
## project                                                             0.209
## salari                                                      0.251
## smart                                                               0.348
## staff
## support                                             0.373
## team                                                0.206
## time                        0.140                   0.110
## train                                               0.119
## work                        0.150                   0.126           0.105
##            Factor9 Factor10
## balanc
## life
## career
## flexibl
## opportun    0.436
## advanc
## amaz
## benefit
## big                -0.118
## bonus
## brand
## busi
## client
## colleagu            0.142
## compani            -0.237
```

```
## cowork              0.194
## cultur      0.115  -0.107
## decent
## develop     0.176
## differ
## divers      0.104
## employe
## environ             0.275
## excel
## experi
## exposur
## firm
## friend              0.276
## grow
## growth      0.229
## help       -0.107
## home
## hour
## industri
## interest
## learn
## locat
## lot
## manag
## mani
## move               -0.151
## network
## new
## nice       -0.155   0.138
## offic
## pay
## peopl
## place      -0.180
## profession  0.111
## project
## salari
## smart
## staff               0.101
## support     0.157
## team
## time
## train
## work                0.132
##
##                 Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7 Factor8
## SS loadings       1.844   1.120   0.984   0.945   0.892   0.673   0.662   0.590
## Proportion Var    0.032   0.019   0.017   0.016   0.015   0.012   0.011   0.010
## Cumulative Var    0.032   0.051   0.068   0.084   0.100   0.111   0.123   0.133
##                 Factor9 Factor10
## SS loadings       0.504    0.451
## Proportion Var    0.009    0.008
## Cumulative Var    0.142    0.149
##
## Test of the hypothesis that 10 factors are sufficient.
## The chi square statistic is 4056 on 1118 degrees of freedom.
## The p-value is 0
```

```r
fa.cons = factanal(cons_dtm1.df,9,scores="regression")
print(fa.cons, sort=TRUE)
```

```
##
## Call:
## factanal(x = cons_dtm1.df, factors = 9, scores = "regression")
```

```
##
## Uniquenesses:
##       bad    balanc       big      busi    career     chang   compani   compens
##     0.959     0.096     0.965     0.929     0.849     0.937     0.918     0.990
##    cultur difficult   employe   environ    expect    growth      hard      high
##     0.933     0.991     0.805     0.850     0.776     0.773     0.966     0.853
##      hour      lack      life     littl      long       lot       low     manag
##     0.045     0.968     0.153     0.974     0.581     0.939     0.005     0.538
##      mani      move  opportun       pay     peopl     polit      poor   pressur
##     0.873     0.964     0.840     0.866     0.798     0.975     0.909     0.005
##    promot    salari      sale      slow     staff    stress      take      team
##     0.900     0.935     0.915     0.976     0.733     0.844     0.766     0.870
##      time      work      year
##     0.811     0.875     0.716
##
## Loadings:
##           Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7 Factor8
## balanc     0.938                                           0.119
## life       0.905                                           0.127
## hour               0.961                   0.105
## long               0.636
## low                        0.992
## pressur                                            0.992
## manag                              0.210   0.375           0.511
## bad                                0.133                   0.141
## big                                0.171
## busi               0.205           0.102   0.102
## career                                                     0.103   0.361
## chang                              0.235
## compani                            0.247   0.118
## compens
## cultur                             0.206                   0.144
## difficult
## employe                            0.231   0.287           0.203
## environ                                                    0.116
## expect             0.121                   0.419
## growth                                                             0.468
## hard                               0.157
## high                                       0.125   0.129
## lack                                                       0.148
## littl                                                      0.107   0.103
## lot                                0.212
## mani                               0.239   0.243
## move                               0.127                           0.117
## opportun                                                           0.377
## pay                        0.298           0.149
## peopl                              0.394                   0.185
## polit                              0.133
## poor                                                       0.286
## promot                             0.235   0.148                   0.117
## salari                     0.241
## sale                                               0.228
## slow                                                               0.128
## staff                              0.126   0.464           0.155
## stress
## take                               0.231   0.412
## team                               0.255                   0.210
## time                               0.178   0.366
## work               0.181           0.252
## year                               0.468   0.200
##           Factor9
## balanc
```

```
## life
## hour
## long
## low
## pressur
## manag
## bad
## big
## busi
## career
## chang
## compani
## compens
## cultur
## difficult
## employe
## environ    0.357
## expect     0.162
## growth
## hard
## high       0.326
## lack
## littl
## lot
## mani
## move
## opportun
## pay
## peopl
## polit
## poor
## promot
## salari
## sale       0.123
## slow
## staff
## stress     0.381
## take
## team       0.126
## time       0.118
## work       0.134
## year
##
##                Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7 Factor8
## SS loadings      1.725   1.478   1.184   1.180   1.177   1.079   0.667   0.623
## Proportion Var   0.040   0.034   0.028   0.027   0.027   0.025   0.016   0.014
## Cumulative Var   0.040   0.075   0.102   0.129   0.157   0.182   0.197   0.212
##                Factor9
## SS loadings      0.521
## Proportion Var   0.012
## Cumulative Var   0.224
##
## Test of the hypothesis that 9 factors are sufficient.
## The chi square statistic is 2072 on 552 degrees of freedom.
## The p-value is 2.48e-174
```

```r
# sink() # close the output file
```

We create dataframes for pros and cons with company names, sectors and newly created factors. Based on the words in factors, we also make consise names for them. For cons, it was hard to come up with a good name in one case (since the factor mixes up a lot of things), so we call it "Miscellaneous."

```r
names(pros.df)
```

```
## [1] "doc_id"  "text"    "Company" "sector"
```

```r
# create dataframes with company names and factors
pros.df.fa = cbind(pros.df[,c(1,3:4)],fa.pros$scores)
cons.df.fa = cbind(cons.df[,c(1,3:4)],fa.cons$scores)

names(pros.df.fa)
```

```
##  [1] "doc_id"   "Company"  "sector"   "Factor1"  "Factor2"  "Factor3"
##  [7] "Factor4"  "Factor5"  "Factor6"  "Factor7"  "Factor8"  "Factor9"
## [13] "Factor10"
```

```r
# after reviewing words in factors, create and add factor names

colnames(pros.df.fa) = c("doc_id","Company","sector",
                      "Work-life balance",
"Career advancement opportunities",
"Flexible working hours, WFH",
"Exposure to different clients and industries",
"Learning and growth opportunities",
"Management support and teamwork",
"Decent salary/benefits",
"Interesting projects and smart people",
"Learning and growth opportunities",
"Friendly envorinment"
)

colnames(cons.df.fa) = c("doc_id","Company","sector",
                        "Work-life balance",
"Long hours",
"Low pay",
"Miscellaneous",
"High expectations",
"High pressure to meet sale targets",
"Poor management",
"Limited career growth opportunities",
"High-stress environment"
)
```

If we count the share of time each factor got value above 2 for each company, we can make correspondence analysis maps, similarly to what we did above, but this time with factors instead of words. We won't explore those shares as they are diffuclult to interpret due to all the data manipulations that we've undertaken to make our factor analysis. Still, they might be still good to highlight differences between companies on a plot and derive some high-level conclusions.

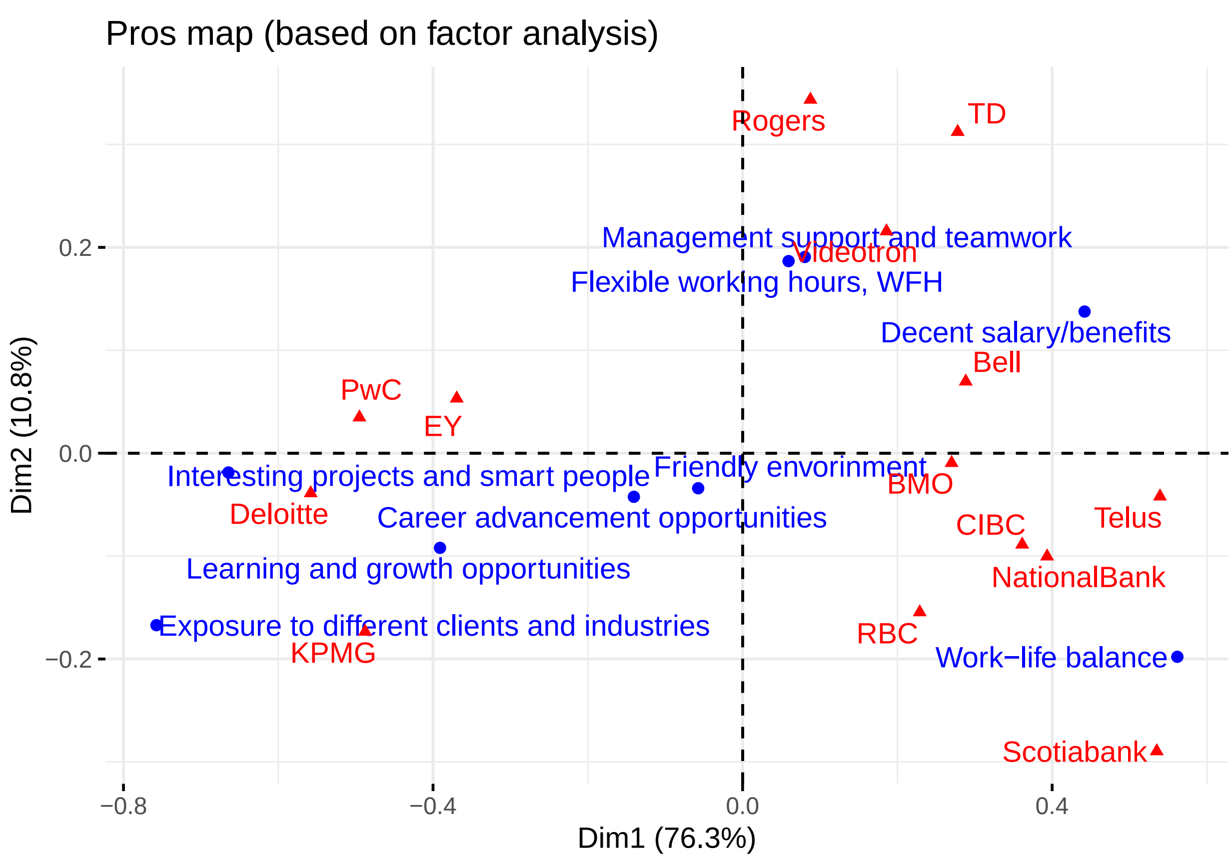
```r
# create dataframes with companies in cols, factors in rows
# in cells: % of factor values above 2.
prosFact = pros.df.fa %>%
  pivot_longer(cols = -c("doc_id", "Company", "sector"),
               names_to = "factor",values_to = "correl") %>%
  group_by(Company,factor) %>%
  summarize(above2=sum(correl>=2)/n()) %>%
  pivot_wider(names_from = Company,values_from = above2)
```

```
## Warning: Duplicate column names detected, adding .copy variable
```

```
consFact = cons.df.fa %>%
  pivot_longer(cols = -c("doc_id", "Company", "sector"),
               names_to = "factor",values_to = "correl") %>%
  group_by(Company,factor) %>%
  summarize(above2=sum(correl>=2)/n()) %>%
  pivot_wider(names_from = Company,values_from = above2)
```

These maps highlight pretty much the same as the ones above. People appreciate work in consulting because of learning and growth opportunities and smart coworkers, while in the other two sectors - mostly for flexible work hours, decent pay and work-life balance.

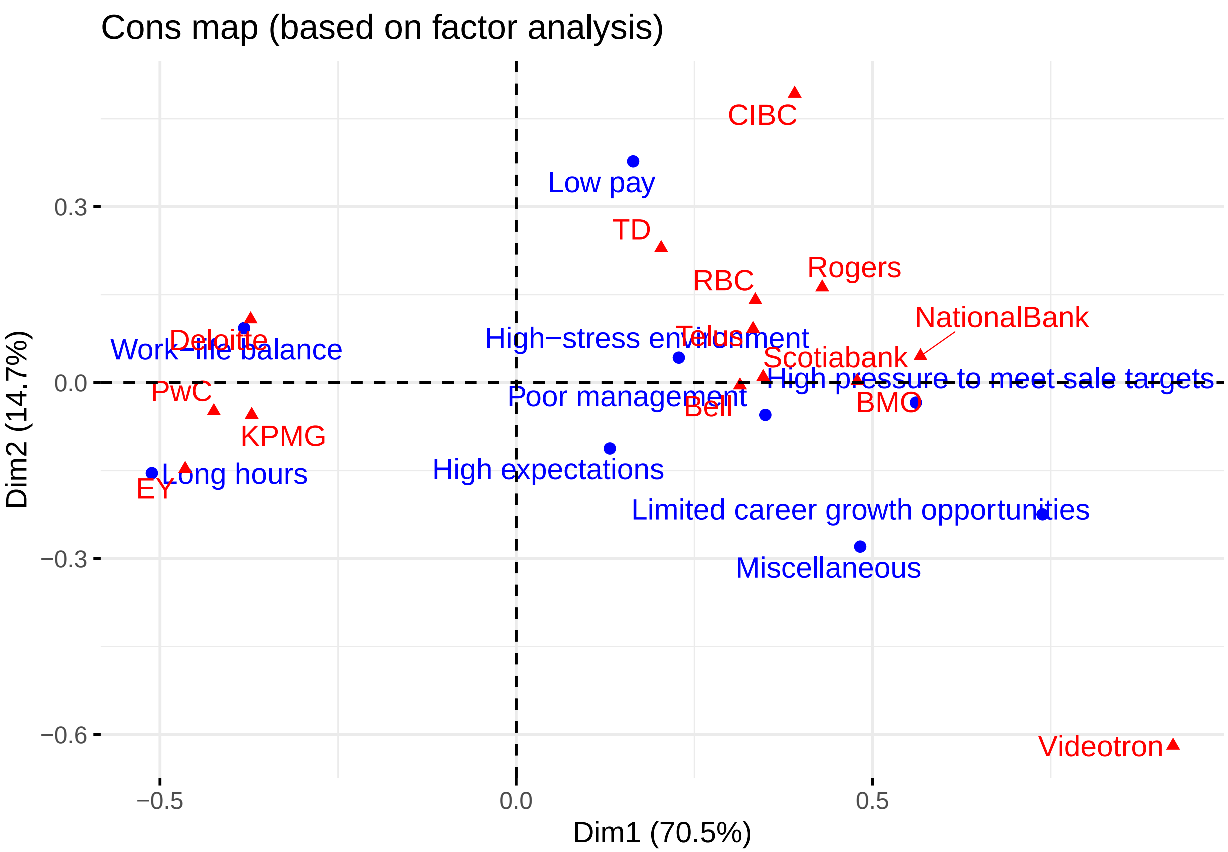
```
# PROS
prosFact.df = as.data.frame(prosFact) # convert tibble to dataframe
prosFact.df[is.na(prosFact.df)]=0 # replace NA with 0
rownames(prosFact.df) = prosFact.df$factor # set company to rownames
prosFact.df = prosFact.df[,-1] # remove the first column with terms
pros.fa.ca <- CA(prosFact.df, graph = FALSE)
set.seed(1)
CA_FA_pros = fviz_ca_biplot(pros.fa.ca, repel = TRUE) +
  ggtitle("Pros map (based on factor analysis)")
CA_FA_pros
```



On the downside, people working in consulting face poor work-life balance while those in banks/telecom might face low career growth opportunities, poor management and high pressure to meet sales targets.

```
# CONS
consFact.df = as.data.frame(consFact) # convert tibble to dataframe
consFact.df[is.na(consFact.df)]=0 # replace NA with 0
rownames(consFact.df) = consFact.df$factor # set company to rownames
consFact.df = consFact.df[,-1] # remove the first column with terms
```

```
cons.fa.ca <- CA(consFact.df, graph = FALSE)
set.seed(1)
CA_FA_cons = fviz_ca_biplot(cons.fa.ca, repel = TRUE) +
  ggtitle("Cons map (based on factor analysis)")
CA_FA_cons
```



8. Exploring words in common

8.1. Common words in pros and cons

This analysis was inspired by one of DataCamp courses on text mining. We can explore how many times different keywords are mentioned in pros and cons and plot words with the highest differences in mentions. For this, we create a function that will take a list of dataframes as input (one company/industry = one dataframe) and output a plot. We will be able to specify chart title, select top-N words to show along with the gap between left and right parts of the plot (to ensure words are shown in full).

In this and several following chapters, we use stemming and then `stemCompletion()` function to complete stems with most frequent words from the dictionary.

```
# Augment our stopwords list
stop.words = c(stop.words, "lot", "lots", "always", "sometimes",
               "little", "nice")

sort(stop.words)
```

```
##   [1] "a"          "about"      "above"      "after"      "again"
##   [6] "against"    "all"        "also"       "always"     "always"
##  [11] "am"         "an"         "and"        "another"    "any"
##  [16] "are"        "aren't"     "around"     "as"         "at"
##  [21] "away"       "back"       "bank"       "be"         "because"
```

```
##  [26] "been"        "before"      "being"       "bell"        "below"
##  [31] "between"     "bmo"         "both"        "but"         "by"
##  [36] "can"         "can't"       "cannot"      "cibc"        "could"
##  [41] "couldn't"    "deloitte"    "did"         "didn't"      "do"
##  [46] "does"        "doesn't"     "doing"       "don't"       "down"
##  [51] "during"      "each"        "even"        "few"         "for"
##  [56] "from"        "further"     "get"         "good"        "great"
##  [61] "had"         "hadn't"      "has"         "hasn't"      "have"
##  [66] "haven't"     "having"      "he"          "he'd"        "he'll"
##  [71] "he's"        "her"         "here"        "here's"      "hers"
##  [76] "herself"     "him"         "himself"     "his"         "how"
##  [81] "how's"       "i"           "I"           "i'd"         "i'll"
##  [86] "i'm"         "i've"        "if"          "in"          "into"
##  [91] "is"          "isn't"       "it"          "it's"        "its"
##  [96] "itself"      "job"         "just"        "kind"        "kpmg"
## [101] "let's"       "like"        "little"      "lot"         "lots"
## [106] "made"        "make"        "may"         "me"          "mean"
## [111] "more"        "most"        "much"        "mustn't"     "my"
## [116] "myself"      "national"    "need"        "nice"        "no"
## [121] "nor"         "not"         "of"          "off"         "on"
## [126] "once"        "one"         "only"        "or"          "other"
## [131] "ought"       "our"         "ours"        "ourselves"   "out"
## [136] "over"        "own"         "rbc"         "really"      "rogers"
## [141] "same"        "shan't"      "she"         "she'd"       "she'll"
## [146] "she's"       "should"      "shouldn't"   "so"          "some"
## [151] "sometimes"   "sometimes"   "still"       "such"        "td"
## [156] "telus"       "than"        "that"        "that's"      "the"
## [161] "their"       "theirs"      "them"        "themselves"  "then"
## [166] "there"       "there's"     "these"       "they"        "they'd"
## [171] "they'll"     "they're"     "they've"     "this"        "those"
## [176] "through"     "to"          "too"         "under"       "until"
## [181] "up"          "very"        "videotron"   "was"         "wasn't"
## [186] "we"          "we'd"        "we'll"       "we're"       "we've"
## [191] "well"        "were"        "weren't"     "what"        "what's"
## [196] "when"        "when's"      "where"       "where's"     "which"
## [201] "while"       "who"         "who's"       "whom"        "why"
## [206] "why's"       "will"        "with"        "won't"       "work"
## [211] "would"       "wouldn't"    "you"         "you'd"       "you'll"
## [216] "you're"      "you've"      "your"        "yours"       "yourself"
## [221] "yourselves"
```

Let's create a **dictionary for stem completion**. To do this, we just pick 500 most frequent words from our corpus of all pros and cons. If there are several words with the same stem in this dictionary, e.g. learn and learning, while applying stemCompletion()‘ function, we will pick up the first (most frequent) term from the list to complete our stems.

```
# create a dictionary
top500terms = freq_terms(c(data$pros,data$cons), top=500, at.least=3, stopwords=stop.words)
# top500terms$WORD
```

Here, we prepare data for several following functions: create lists for pros/cons & sectors/companies. One element in a list = one data frame = one company (or one sector). Separate lists created for pros and cons.

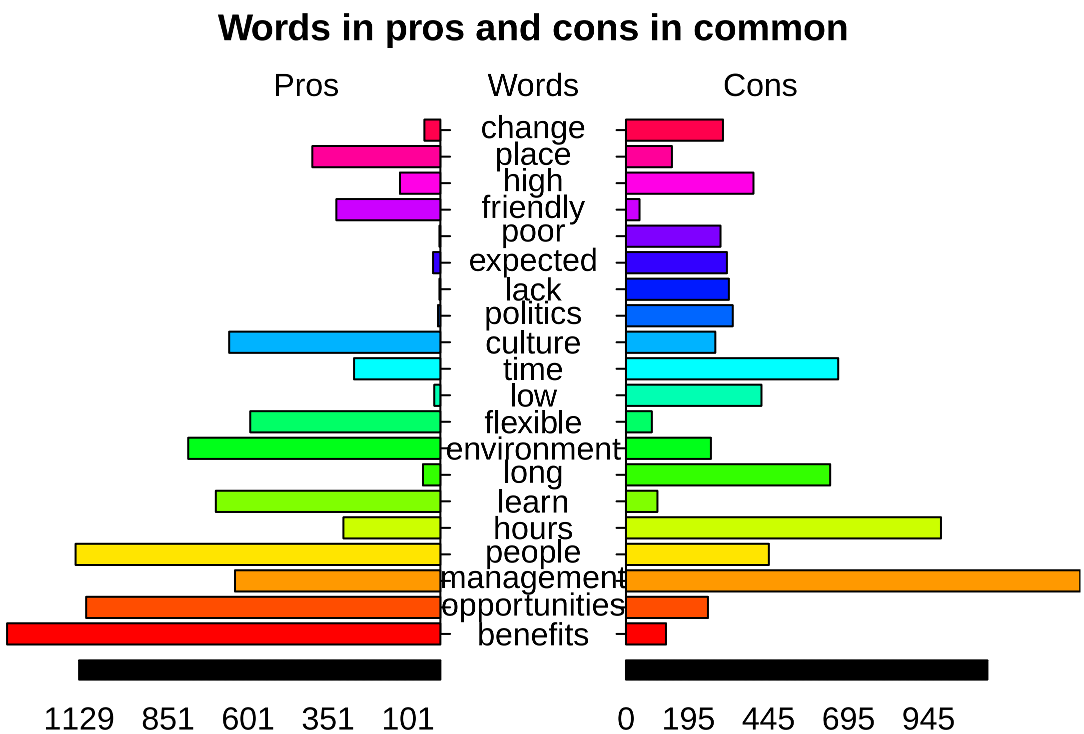
```
# 2 lists with pros:
pros.sector.list = split(pros.df,f=pros.df$sector)
pros.comp.list = split(pros.df,f=pros.df$Company)
# 2 lists with cons:
cons.sector.list = split(cons.df,f=cons.df$sector)
cons.comp.list = split(cons.df,f=cons.df$Company)
```

```r
# create a function to make pyramid plots
# gap means the width between left and right part of the plot
# we can change whether to use stemming or not within a function
pyramid_plot = function(data, title =  "Words in Common",
                        topNwords = 25, gap = 100) {
    # LOOP OVER PAIRS OF SECTORS OR COMPANIES
    for (i in 1:(length(data)-1)) {
      for (j in (i+1):length(data)) {
        vec = NULL
        # print(names(data[c(i,j)]))
        text_i = paste(data[[i]]$text, collapse = " ")
        text_j = paste(data[[j]]$text, collapse = " ")
        vec = c(vec, text_i, text_j)
        # Create a corpus and clean it
        corpus <- VCorpus(VectorSource(vec))

        corpus_cl = clean_corpus(corpus) # using a function with stemming

        # Create TDM with two sectors/companies at a time
        tdm <- TermDocumentMatrix(corpus_cl)
        # Give the columns distinct names
        colnames(tdm) <- names(data[c(i,j)])
        m <- as.matrix(tdm)
        df = as.data.frame(m)

        comm_words <- df %>%
          rownames_to_column(var = "word") %>%
          # Keep rows where word appears everywhere
        filter_all(all_vars(. > 0))

        comm_words$difference = abs(comm_words[2] - comm_words[3])

        top25_df = comm_words %>%
        top_n(n=topNwords, wt = difference)
        top25_df = top25_df[order(top25_df$difference,decreasing = TRUE),]

        # stem completion
        top25_df$word = stemCompletion(top25_df$word, dictionary = top500terms$WORD, type = "first")

        pyramid.plot(
          top25_df[,2],
          top25_df[,3],
          labels = top25_df$word,
          # top.labels = c(names(data[i]), "Words", names(data[j])),
          top.labels = c("Pros", "Words", "Cons"),
          main = title,
          unit = NULL, gap = gap)
        }
    }
}
```

As we can see below, across all companies, *benefits*, *opportunities* and *peope* have the highest positive balance of mentions, while *management*, *hours* and *long* - the highest negative balance (the lower the term on the plot, the higher the absolute difference between its mentions in pros and cons).

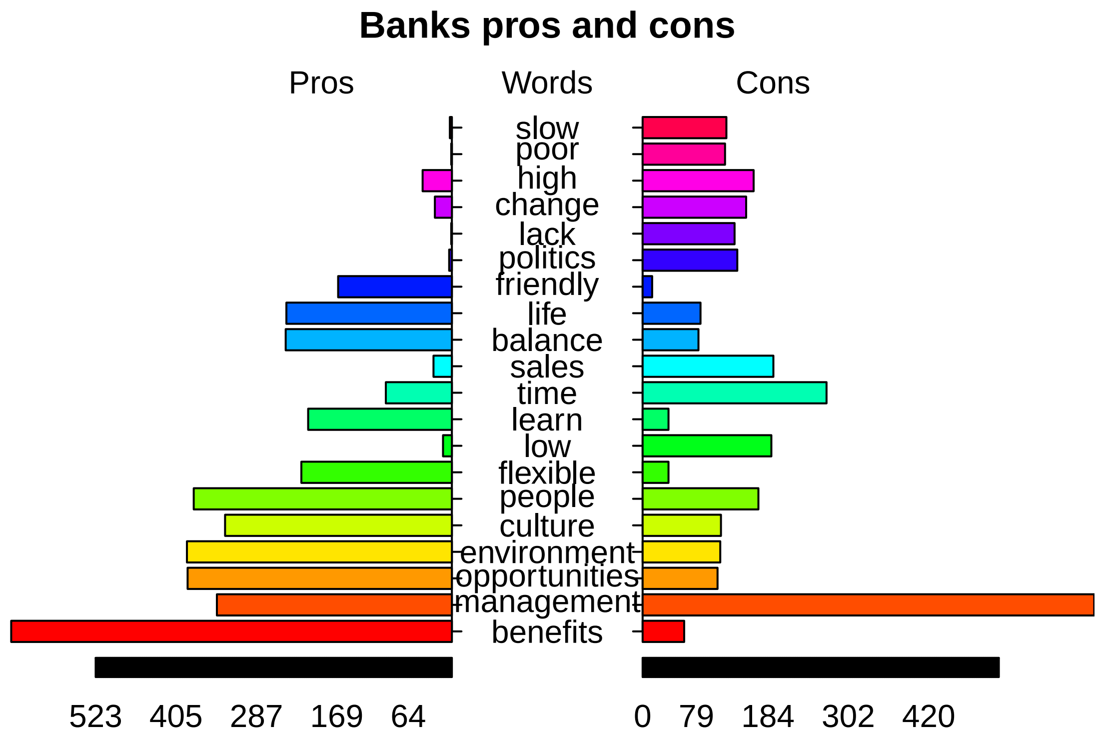
```r
pros_cons_list = list(pros.df, cons.df)
names(pros_cons_list) = c("Pros", "Cons")
```

```
pyramid_plot(pros_cons_list, title = "Words in pros and cons in common", topNwords = 20, gap = 290)
```

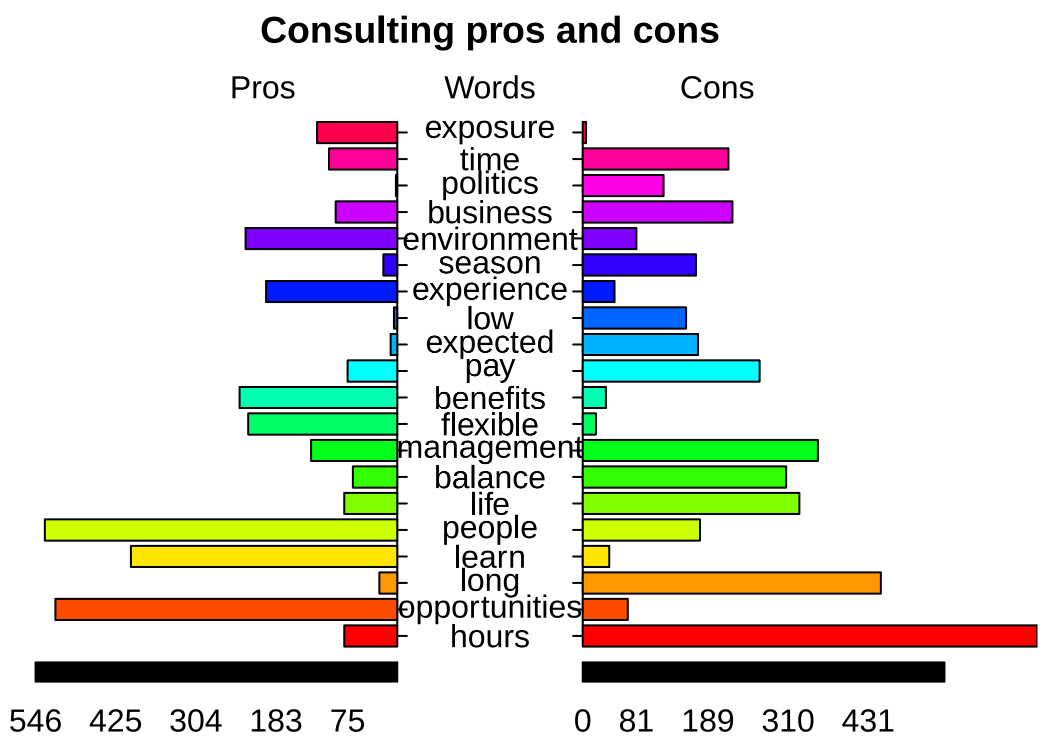


We can do the same comparisons within sectors. We see quite a bunch of terms with positive balance in banks, such as *benefits*, *opportunities*, *environment* and *culture*, while *management* is the term with the highest negative balance. In consulting, *opportunities*, *people* and *learning* stand out as terms with positive balance, while the major drawback is *long hours* and *work-life balance*. Telecom is attractive by its *benefits* but the *management* gathers a lot of complaints.

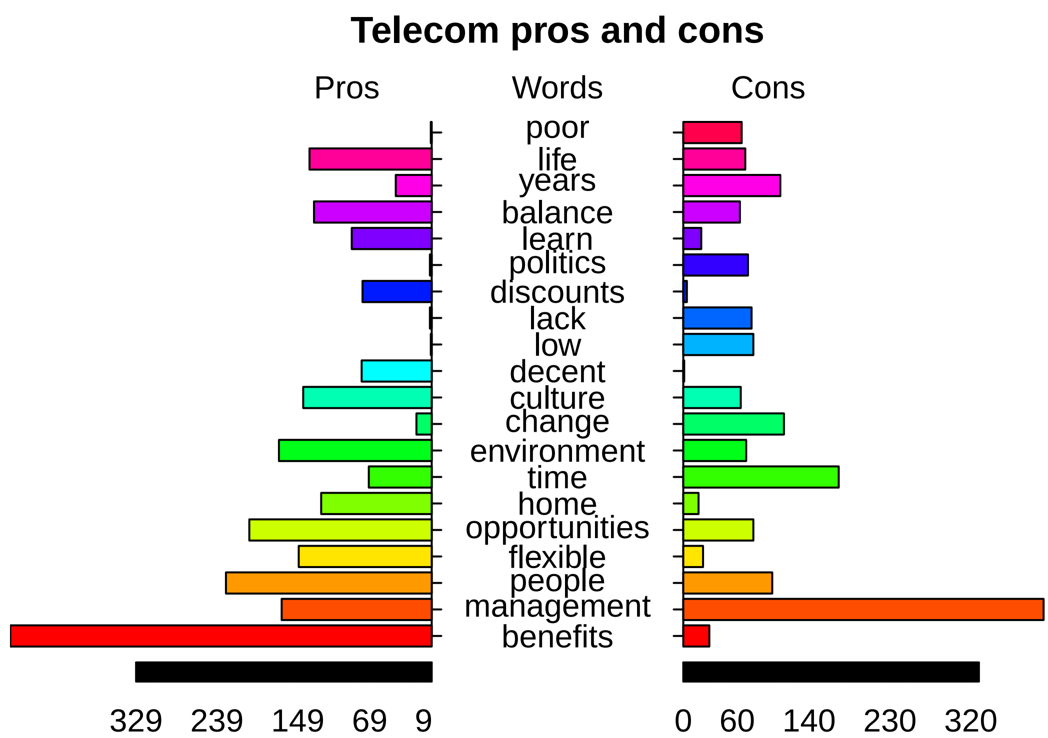
```
pyramid_plot(list(pros.sector.list[[1]],cons.sector.list[[1]]), topNwords = 20, gap = 140,
             title = paste(names(pros.sector.list)[1], "pros and cons"))
```




```
pyramid_plot(list(pros.sector.list[[2]],cons.sector.list[[2]]), topNwords = 20, gap = 140,
             title = paste(names(pros.sector.list)[2], "pros and cons"))
```




```
pyramid_plot(list(pros.sector.list[[3]],cons.sector.list[[3]]), topNwords = 20, gap = 140,
             title = paste(names(pros.sector.list)[3], "pros and cons"))
```



8.2. Common words across sectors and companies

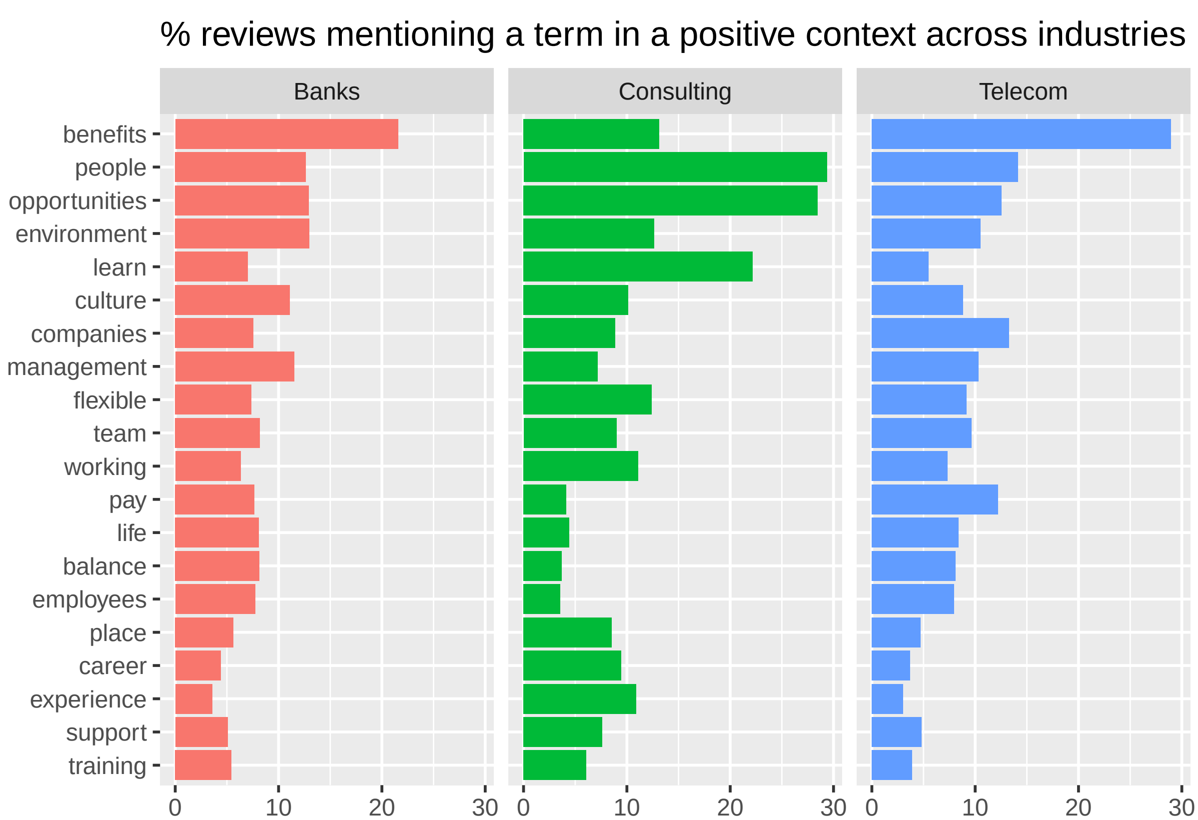
It could be nice to compare industries or companies side-by-side. A function to prepare data for such plots is provided below. As an input, it will take lists with pros/cons for *companies* and aggregate companies to sectors.

```
# data = list with dataframes as above
data_prep = function(data, max.words=25) {
  vec = NULL
    for (i in 1:length(data)) {
    # print(names(data[i]))
    text_i = paste(data[[i]]$text, collapse = " ")
    vec = c(vec, text_i)
  }
  # Create corpus and clean it: choose whether to use stemming or not.
  corpus <- VCorpus(VectorSource(vec))
  corpus_cl = clean_corpus(corpus) # stemming
  # corpus_cl = clean_corpus2(corpus) # no stemming

  # Create TDM with all sectors/companies to appear on the plot
  tdm <- TermDocumentMatrix(corpus_cl) # one column per sector or company
  # Give the columns distinct names
  colnames(tdm) <- names(data)
  # Create matrix
  m <- as.matrix(tdm)

  x = m %>%
  as_tibble(rownames = "Term") %>%
  mutate(term_freq = rowSums(.[2:length(data)])) %>%
  mutate(Banks = BMO+CIBC+NationalBank+RBC+Scotiabank+TD,
```

```
        Consulting = Deloitte + EY + KPMG + PwC,
        Telecom = Bell+Rogers+Telus+Videotron) %>%
  arrange(desc(term_freq)) %>%
  head(max.words)

  # stem completion
  x$Term <- stemCompletion(x$Term, dictionary=top500terms$WORD,type="first")

  return(x)
}

pros_top = data_prep(pros.comp.list,max.words = 20)
cons_top = data_prep(cons.comp.list,max.words = 20)
```

Let's compare most frequent terms mentioned in pros and cons across sectors. In all these plots, we divide term frequency by total number of reviews per company/industry to get % of reviews mentioning a given term. Again, consulting stands out among the three sectors with its specific pros.

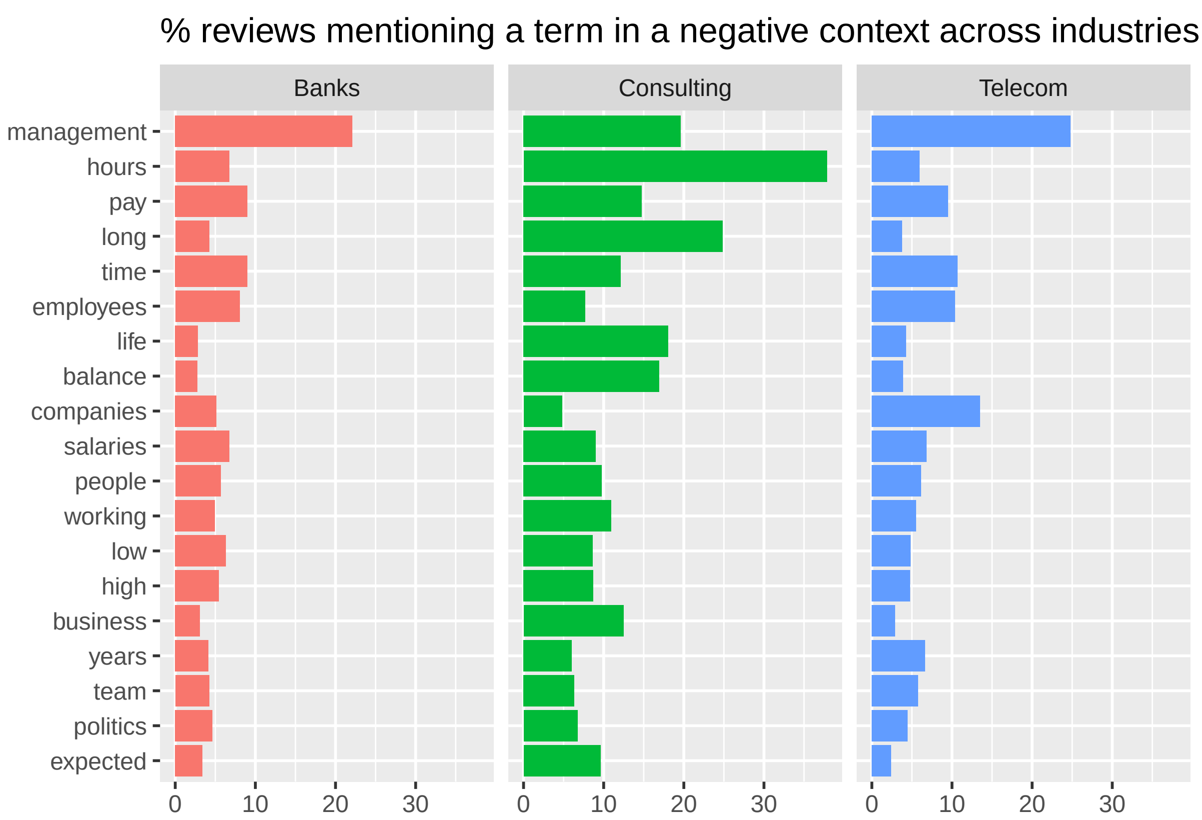
```
pros_top %>%
  select(Term,Banks,Consulting,Telecom) %>%
  mutate(Banks=Banks/3000*100, Consulting=Consulting/1812*100, Telecom = Telecom/1619*100) %>%
  pivot_longer(cols = -Term, names_to = "sector", values_to = "Count") %>%
  group_by(Term,sector) %>%
  ggplot(aes(x=reorder(Term, Count), y=Count, fill=sector)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(sector ~.) +
  coord_flip() +
  ggtitle("% reviews mentioning a term in a positive context across industries") +
  xlab("") + ylab("")
```



Consulting, again, stands out in cons with its work-life balance and long hours. Almost 40% of cons mention *hours*!

Stem completion is not capable to complete the word mani (many) and produces NA. Since this word is not informative, we remove it from all plots with cons (here and later on).

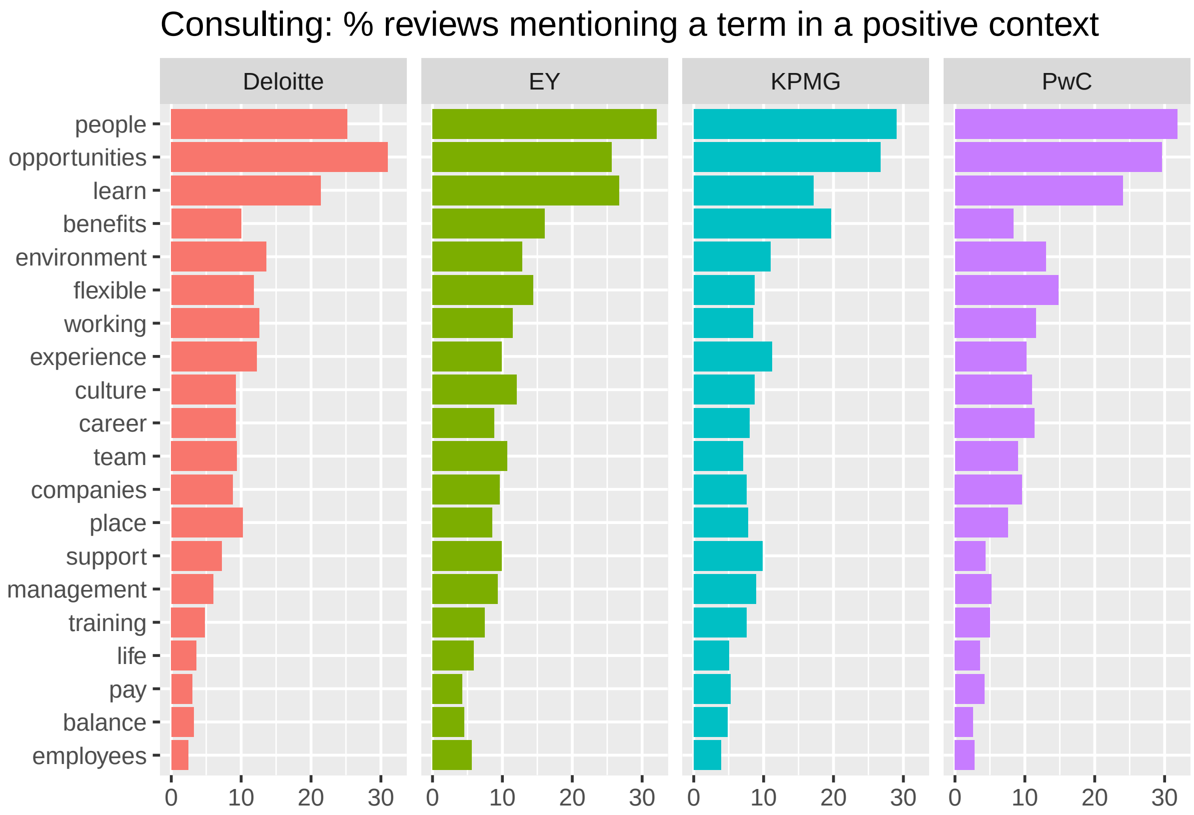
```
cons_top %>%
  filter(!is.na(Term)) %>% # filter one line with NA
  select(Term,Banks,Consulting,Telecom) %>%
  # normalize by the number of reviews per sector (company)
  mutate(Banks=Banks/3000*100, Consulting=Consulting/1812*100, Telecom = Telecom/1619*100) %>%
  pivot_longer(cols = -Term, names_to = "sector", values_to = "Count") %>%
  group_by(Term,sector) %>%
  ggplot(aes(x=reorder(Term, Count), y=Count, fill=sector)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(sector ~.) +
  coord_flip() +
  ggtitle("% reviews mentioning a term in a negative context across industries") +
  xlab("") + ylab("")
```



We can also dive deeper and explore how the companies within the same industry fare. There are some small differences in pros for consulting, but overall they are quite similar.

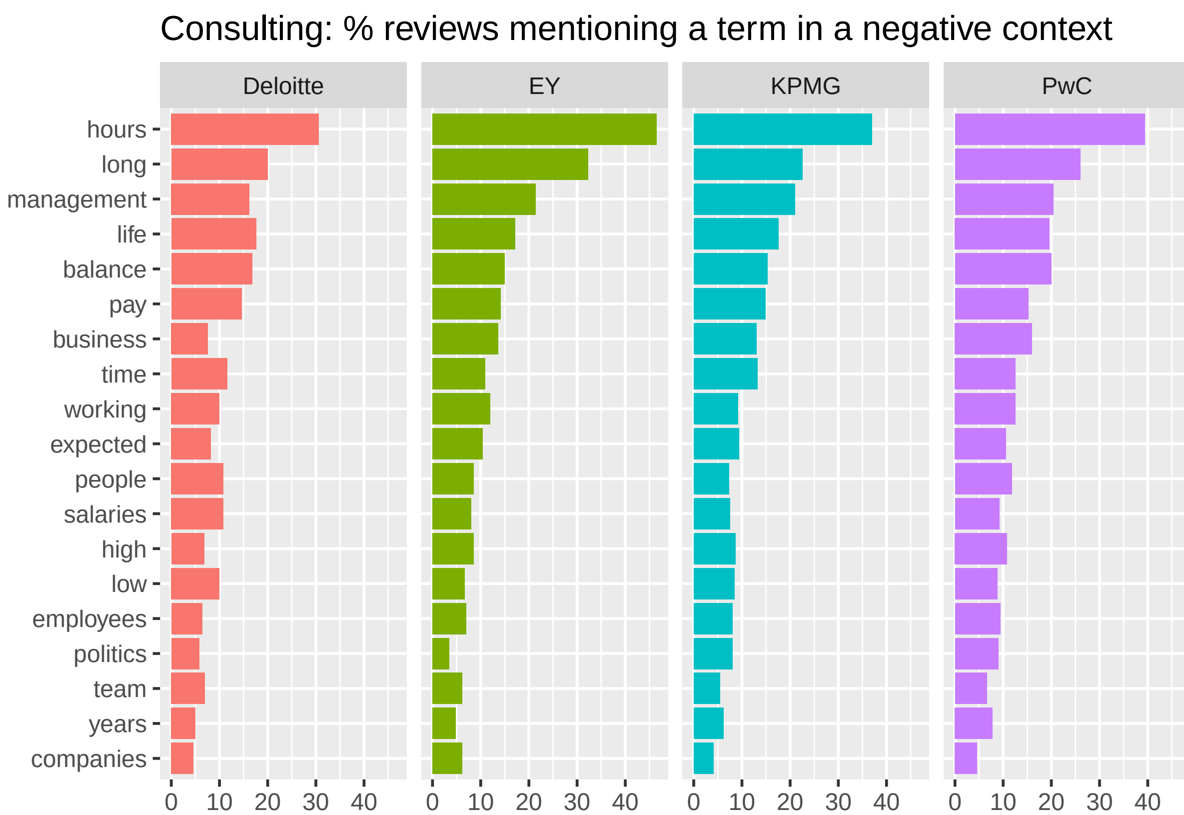
```
pros_top %>%
  select(Term,Deloitte, EY, KPMG, PwC) %>%
  mutate(Deloitte = Deloitte/500*100, EY = EY/374*100, KPMG = KPMG/438*100, PwC = PwC/500*100) %>%
  pivot_longer(cols = -Term, names_to = "Company", values_to = "Count") %>%
  group_by(Term,Company) %>%
  ggplot(aes(x=reorder(Term, Count), y=Count, fill=Company)) +
  geom_col(show.legend = FALSE) +
  facet_grid(. ~ Company) +
  coord_flip() +
```

```
  ggtitle("Consulting: % reviews mentioning a term in a positive context") +
  xlab("") + ylab("")
```



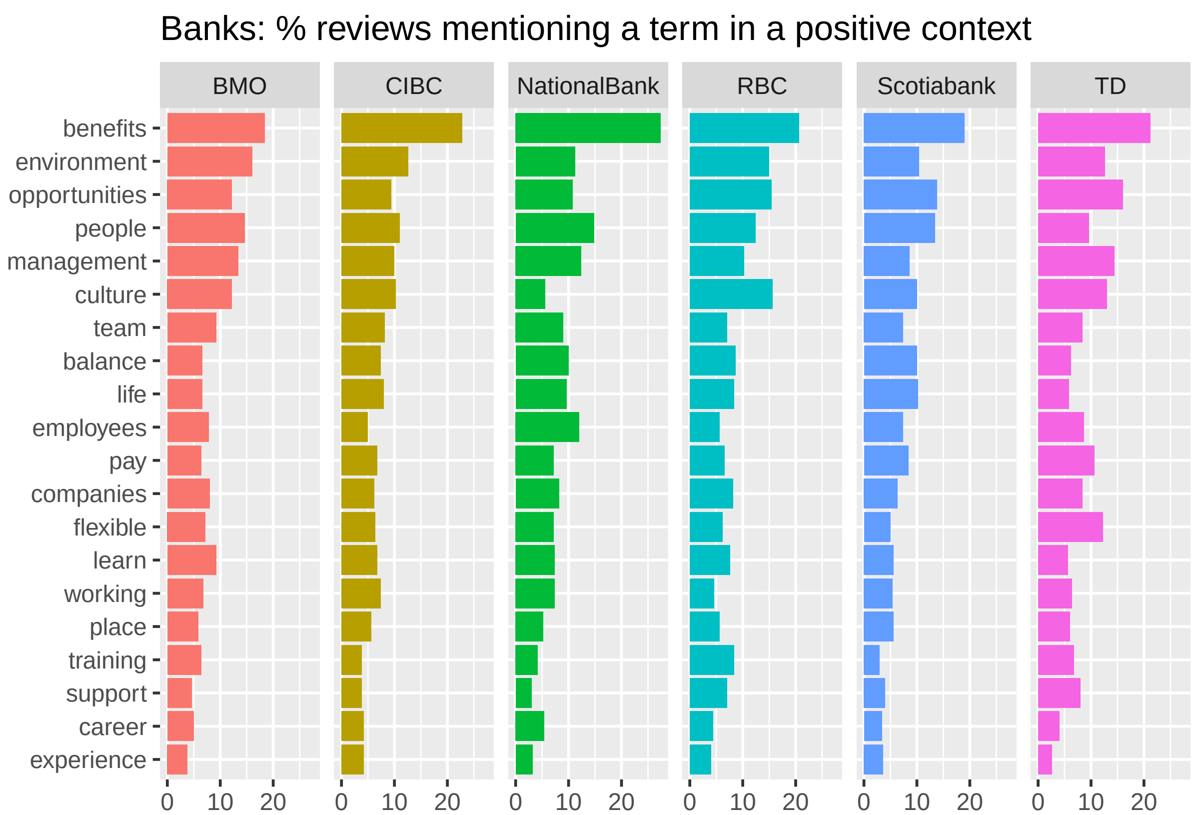
With regard to *long hours*, Deloitte tends to fare the best while EY the worst.

```
cons_top %>%
  filter(!is.na(Term)) %>% # filter one line with NA
  select(Term,Deloitte, EY, KPMG, PwC) %>%
  mutate(Deloitte = Deloitte/500*100, EY = EY/374*100, KPMG = KPMG/438*100, PwC = PwC/500*100) %>%
  pivot_longer(cols = -Term, names_to = "Company", values_to = "Count") %>%
  group_by(Term,Company) %>%
  ggplot(aes(x=reorder(Term, Count), y=Count, fill=Company)) +
  geom_col(show.legend = FALSE) +
  facet_grid(. ~ Company) +
  coord_flip() +
  ggtitle("Consulting: % reviews mentioning a term in a negative context") +
  xlab("") + ylab("")
```

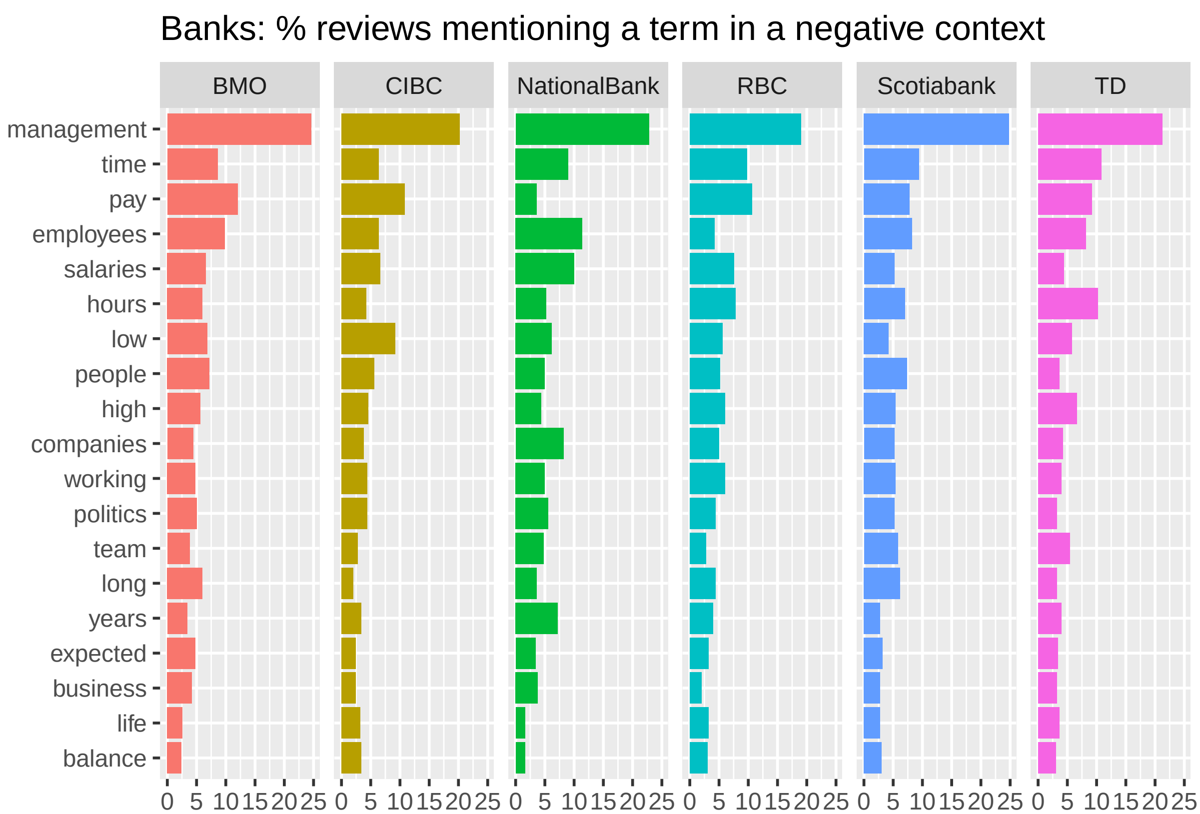


Among banks, National Bank tends to have the highest number of mentions of *benefits*, and the lowest of *culture* (where RBC stands out), but is in line or even outperforming others by *people* and *employees*.

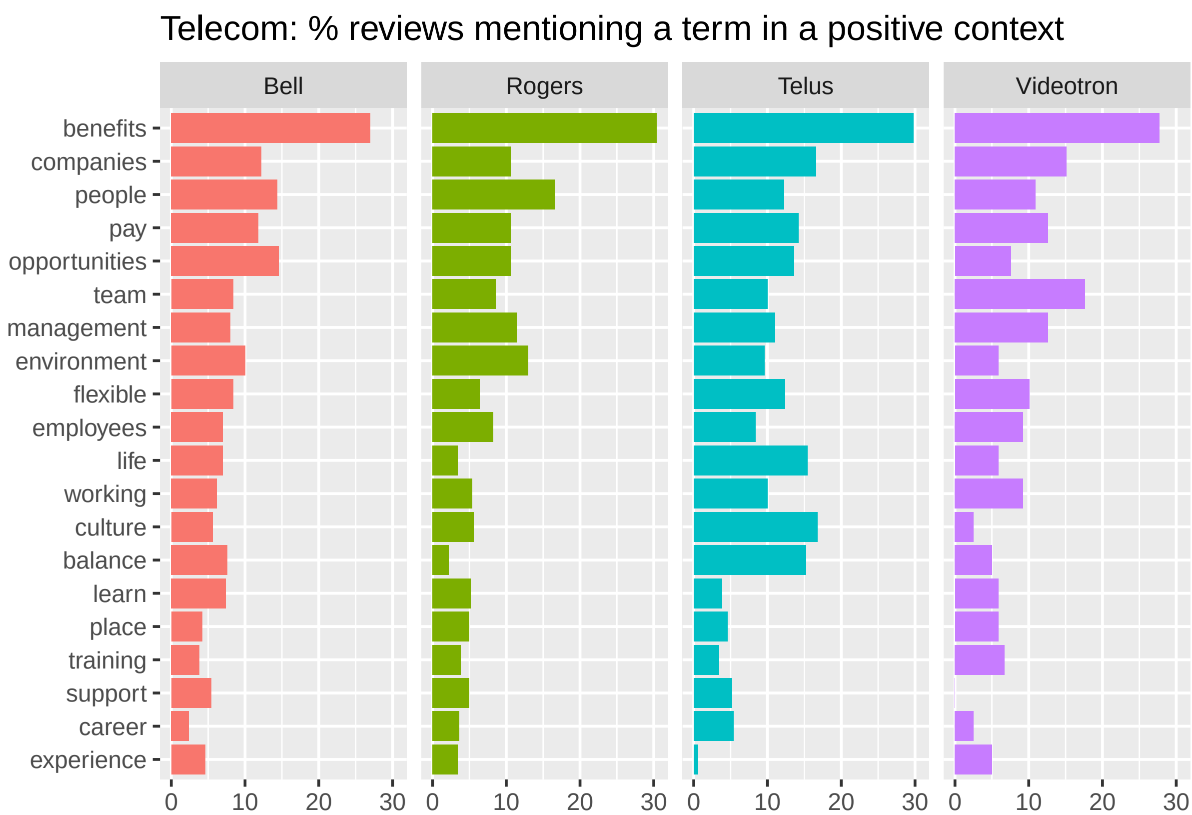
```
pros_top %>%
  select(Term, BMO, CIBC, NationalBank, RBC, Scotiabank, TD) %>%
  mutate(BMO = BMO/500*100, CIBC = CIBC/500*100, NationalBank = NationalBank/500*100,
         RBC = RBC/500*100, Scotiabank = Scotiabank/500*100, TD = TD/500*100) %>%
  pivot_longer(cols = -Term, names_to = "Company", values_to = "Count") %>%
  group_by(Term,Company) %>%
  ggplot(aes(x=reorder(Term, Count), y=Count, fill=Company)) +
  geom_col(show.legend = FALSE) +
  facet_grid(. ~ Company) +
  coord_flip() +
  ggtitle("Banks: % reviews mentioning a term in a positive context") +
  xlab("") + ylab("")
```



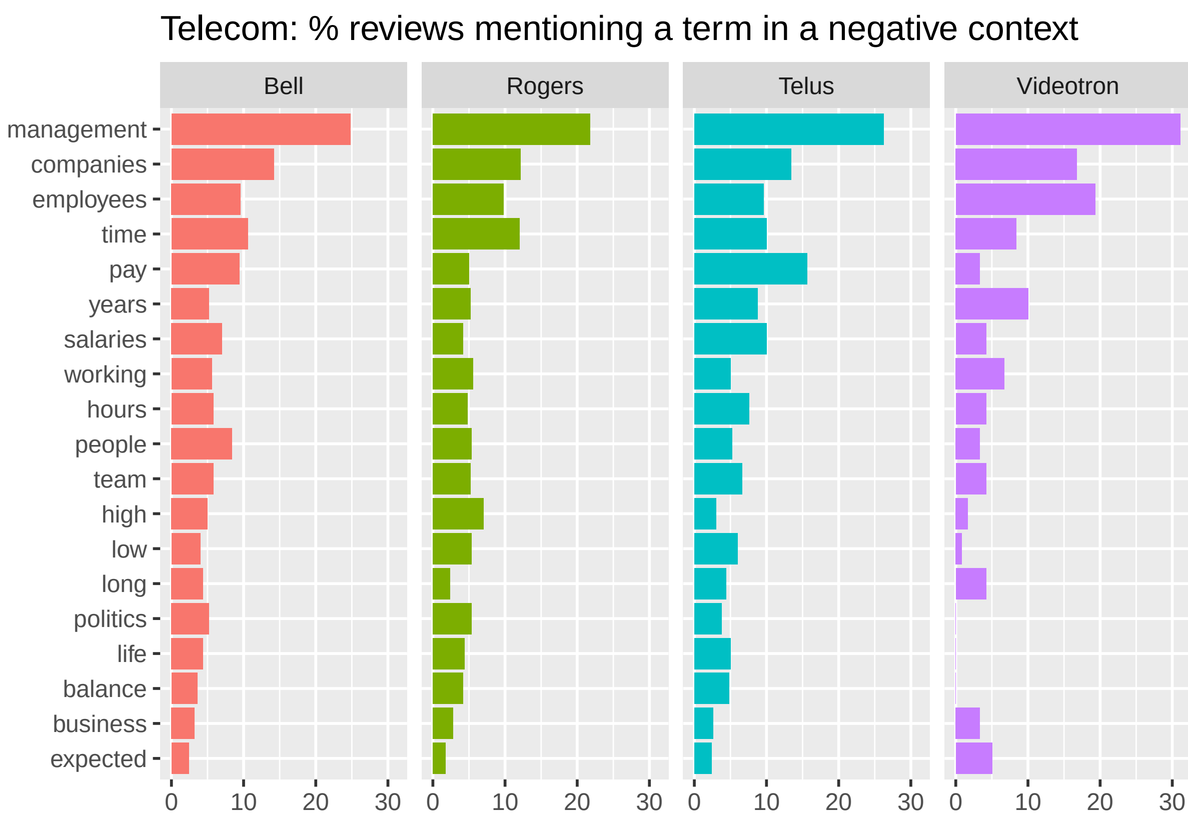

```
cons_top %>%
  filter(!is.na(Term)) %>% # filter one line with NA
  select(Term, BMO, CIBC, NationalBank, RBC, Scotiabank, TD) %>%
  mutate(BMO = BMO/500*100, CIBC = CIBC/500*100, NationalBank = NationalBank/500*100,
         RBC = RBC/500*100, Scotiabank = Scotiabank/500*100, TD = TD/500*100) %>%
  pivot_longer(cols = -Term, names_to = "Company", values_to = "Count") %>%
  group_by(Term,Company) %>%
  ggplot(aes(x=reorder(Term, Count), y=Count, fill=Company)) +
  geom_col(show.legend = FALSE) +
  facet_grid(. ~ Company) +
  coord_flip() +
  ggtitle("Banks: % reviews mentioning a term in a negative context") +
  xlab("") + ylab("")
```




```
pros_top %>%
  select(Term, Bell, Rogers, Telus, Videotron) %>%
  mutate(Bell = Bell/500*100, Rogers = Rogers/500*100,
         Telus = Telus/500*100, Videotron = Videotron/119*100) %>%
  pivot_longer(cols = -Term, names_to = "Company", values_to = "Count") %>%
  group_by(Term,Company) %>%
  ggplot(aes(x=reorder(Term, Count), y=Count, fill=Company)) +
  geom_col(show.legend = FALSE) +
  facet_grid(. ~ Company) +
  coord_flip() +
  ggtitle("Telecom: % reviews mentioning a term in a positive context") +
  xlab("") + ylab("")
```




```
cons_top %>%
  filter(!is.na(Term)) %>% # filter one line with NA
  select(Term, Bell, Rogers, Telus, Videotron) %>%
  mutate(Bell = Bell/500*100, Rogers = Rogers/500*100,
         Telus = Telus/500*100, Videotron = Videotron/119*100) %>%
  pivot_longer(cols = -Term, names_to = "Company", values_to = "Count") %>%
  group_by(Term,Company) %>%
  ggplot(aes(x=reorder(Term, Count), y=Count, fill=Company)) +
  geom_col(show.legend = FALSE) +
  facet_grid(. ~ Company) +
  coord_flip() +
  ggtitle("Telecom: % reviews mentioning a term in a negative context") +
  xlab("") + ylab("")
```



9. Chi-square residuals analysis

```
pros_top
```

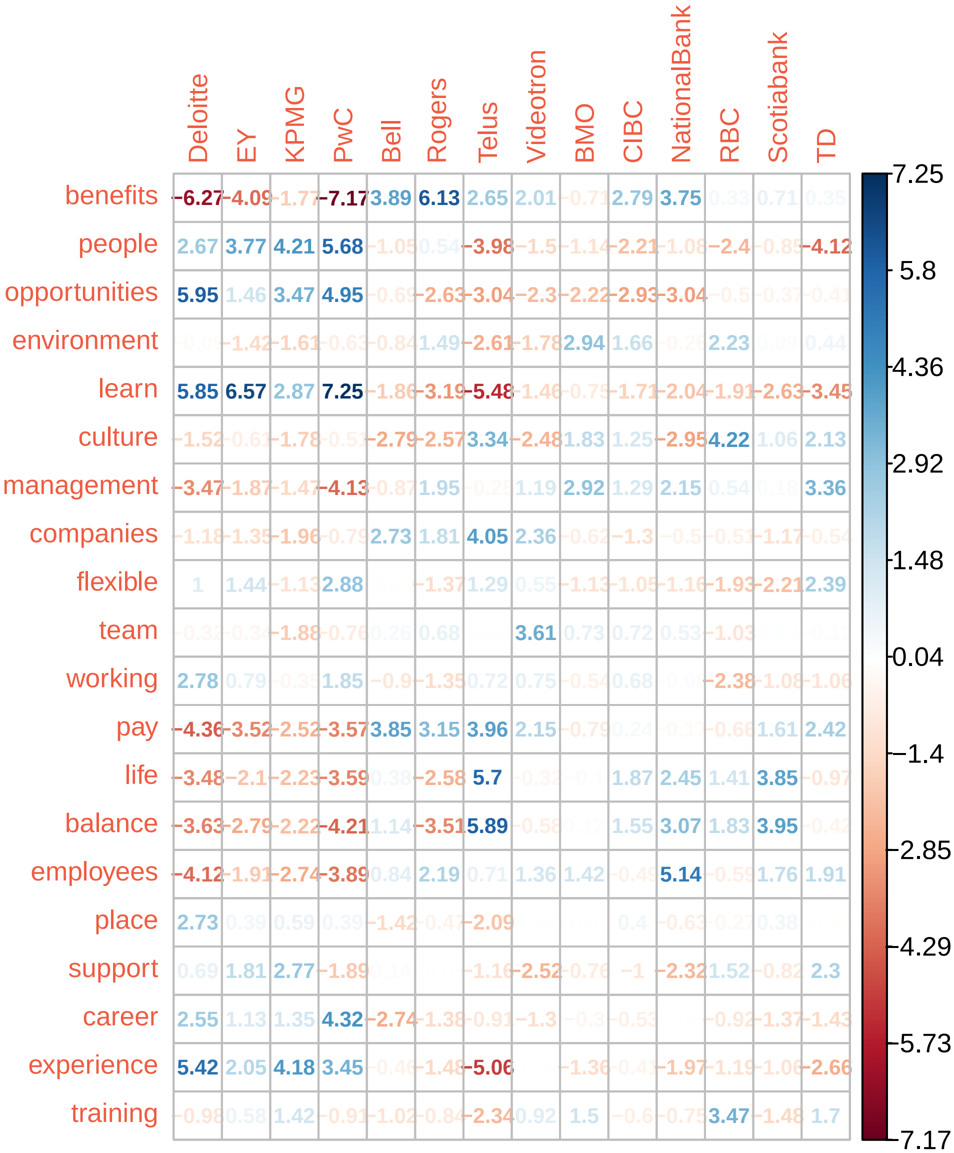
```
## # A tibble: 20 x 19
##    Term   Bell   BMO  CIBC Deloitte    EY  KPMG NationalBank   PwC   RBC Rogers
##    <chr> <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>        <dbl> <dbl> <dbl>  <dbl>
##  1 bene~   135    92   114       50    60    86          137    42   103    152
##  2 peop~    72    73    55      126   120   127           74   159    62     83
##  3 oppo~    73    61    47      155    96   117           54   148    77     53
##  4 envi~    50    80    63       68    48    48           56    65    75     65
##  5 learn    37    46    34      107   100    75           37   120    38     26
##  6 cult~    28    61    51       46    45    38           28    55    78     28
##  7 mana~    40    67    50       30    35    39           62    26    51     57
##  8 comp~    61    40    31       44    36    33           41    48    41     53
##  9 flex~    42    36    32       59    54    38           36    74    31     32
## 10 team     42    46    41       47    40    31           45    45    35     43
## 11 work~    31    34    37       63    43    37           37    58    23     27
## 12 pay      59    32    34       15    16    23           36    21    33     53
## 13 life     35    33    40       18    22    22           48    18    42     17
## 14 bala~    38    33    37       16    17    21           50    13    43     11
## 15 empl~    35    39    25       12    21    17           60    14    28     41
## 16 place    21    29    28       51    32    34           26    38    28     25
## 17 supp~    27    23    19       36    37    43           15    22    35     25
## 18 care~    12    25    21       46    33    35           27    57    22     18
## 19 expe~    23    19    21       61    37    49           16    51    20     17
## 20 trai~    19    32    19       24    28    33           21    25    42     19
## # ... with 8 more variables: Scotiabank <dbl>, TD <dbl>, Telus <dbl>,
## #   Videotron <dbl>, term_freq <dbl>, Banks <dbl>, Consulting <dbl>,
```

```
## #   Telecom <dbl>
```

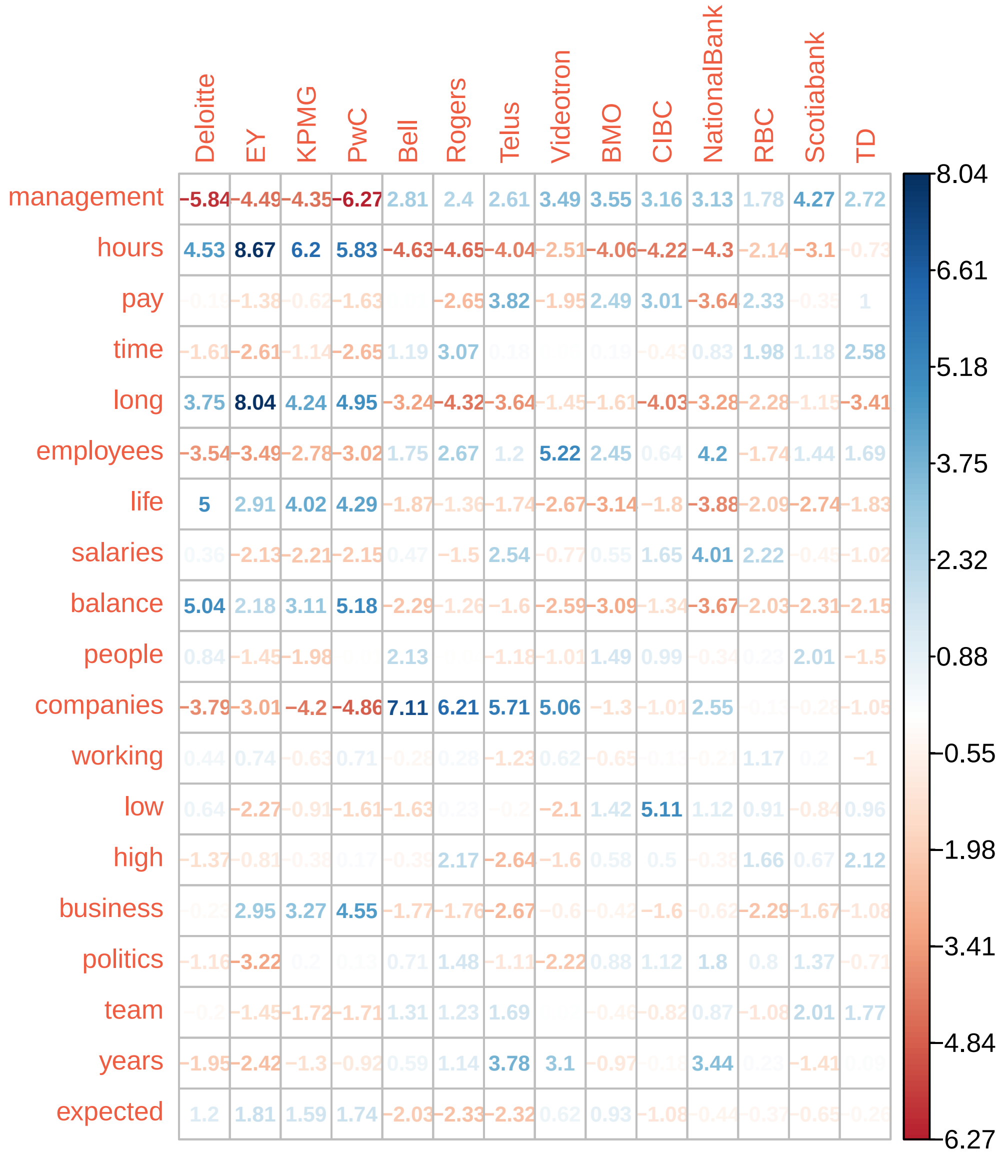
Similarly to Correspondence analysis, we can use term frequencies per company to calculate chi-square residuals for term/company ompbinations. The higher/lower the residual in ABSOLUTE value, the more/less the connection between a term and a company is outstanding. As a rule of thumb, residuals below 2 in absolute value point to the average connection between a term and a company. Absolute values of residuals above 2 point to a connection between a term and a company: high positive (negative) residuals mean that a characteristic is over-represented (under-represented) for a given company.

Here, again, we use stemming and then `stemCompletion()` to complete stems with most frequent words from the dictionary. Below are chi-square test residuals for pros (first plot) and cons (second plot) for all companies.

```
pros_sec = pros_top %>%
  # select(-Banks,-Consulting,-Telecom)
  select(Term, Deloitte, EY, KPMG, PwC, Bell, Rogers, Telus, Videotron,
         BMO, CIBC, NationalBank, RBC, Scotiabank, TD)

pros_sec_m = as.matrix(pros_sec[,2:15])
rownames(pros_sec_m) = pros_sec$Term
chisq_all_pros = chisq.test(pros_sec_m)

corrplot(chisq_all_pros$residuals, method = "number", is.corr = FALSE, tl.col = "tomato2", tl.cex = 0.8,
         number.cex = 0.7)
```




```r
cons_sec = cons_top %>%
  filter(!is.na(Term)) %>% # filter one line with NA
  # select(-Banks,-Consulting,-Telecom)
  select(Term, Deloitte, EY, KPMG, PwC, Bell, Rogers, Telus, Videotron,
         BMO, CIBC, NationalBank, RBC, Scotiabank, TD)

cons_sec_m = as.matrix(cons_sec[,2:15])
rownames(cons_sec_m) = cons_sec$Term
chisq_all_cons = chisq.test(cons_sec_m)

# chisq_all_cons$residuals[sort(chisq_all_cons$residuals[,1],decreasing=T),]
corrplot(chisq_all_cons$residuals, method = "number", is.corr = FALSE, tl.col = "tomato2", tl.cex = 0.8,
         number.cex = 0.7)
```



10. Creating bi-grams

To see what combinations of two terms exist, we remove word "work" from the stoplist to keep it in our corpus but we want to keep some words like nice etc. that we removed later in uni-gram analysis. That's why I created `stop.words2` list at the beginning of this document.

Once we create bi-grams, the number of terms in our TDM explodes: from about 6,000 for uni-grams (across all pros and cons) to about 70,000 for bi-grams. At the same time, the frequency per term declines substantially.

Next, we manually grouped several most often synonymes in the form of bi-grams, e.g., "lifework balanc" and "worklif balanc". This process takes **a lot of time**, so we cleaned only 11 top-most frequent bi-gram combinations for both pros and cons as shown below:

- "worklife balance",
- "remote work",
- "flexible hours",
- "long hours",
- "learning opportunities",
- "growth opportunities",
- "senior management",
- "low pay",

- "company culture",
- "busy season",
- "high turnover".

It's not easy to complete stems for bi-grams, so here we leave stems uncompleted.

```
# data = list with dataframes as above
data_prep_bigram = function(data, max.words=25) {
  vec = NULL
    for (i in 1:length(data)) {
    # print(names(data[i]))
    text_i = paste(data[[i]]$text, collapse = " ")
    vec = c(vec, text_i)
  }
  # Create corpus and clean it
  corpus <- VCorpus(VectorSource(vec))

  # clean corpus
  corpus_cl <- tm_map(corpus, stripWhitespace)
  corpus_cl <- tm_map(corpus_cl, content_transformer(tolower))
  corpus_cl <- tm_map(corpus_cl, removeWords, stop.words2)
  corpus_cl <- tm_map(corpus_cl, removePunctuation)
  corpus_cl  <- tm_map(corpus_cl, stemDocument)
  corpus_cl <- tm_map(corpus_cl, removeWords, stop.words2)
  corpus_cl <- tm_map(corpus_cl, stripWhitespace)

workLifeBalance = c("work life balanc",
  "life unbalanc",
"lifework balanc",
"worklif balanc",
"worklifestyl balanc",
"worklik balanc",
"work life balanc",
"work life unbalanc",
"work lifework balanc",
"work time balanc",
"work balanc",
"hectic balanc",
"word life balanc",
"work famili balanc")

for (i in 1:length(data)) {
  for (j in workLifeBalance) {
  corpus_cl[[i]][1] = gsub(j, "worklife balance", corpus_cl[[i]][1])
  }
}

work_from_home = c("flexibl work home",
                   "remot work",
                   "remot workstyl",
                   "remot home",
                   "remot flexibl place",
                   "wfh",
                   "work home"
                   )

for (i in 1:length(data)) {
  for (j in work_from_home) {
```

```r
  corpus_cl[[i]][1] = gsub(j, "remote work", corpus_cl[[i]][1])
  }
}

flexible_hours = c("flexibl work",
                   "flexibl workstyl",
                   "flexibl work style",
                   "flexibl schedul",
                   "flexibl hour",
                   "flexibl shift",
                   "flexibl time",
                   "flexibl offic hour",
                   "flexible hourslife",
                   "flexibl work time"
                   )

for (i in 1:length(data)) {
  for (j in flexible_hours) {
  corpus_cl[[i]][1] = gsub(j, "flexible hours", corpus_cl[[i]][1])
  }
}

long_hours = c("long hour",
               "long hourslow",
               "long hoursw",
               "long work hour",
               "long work",
               "high work hour",
               "crazi work hour",
               "work long"
)

for (i in 1:length(data)) {
  for (j in long_hours) {
  corpus_cl[[i]][1] = gsub(j, "long hours", corpus_cl[[i]][1])
  }
}

learn_opport = c("learn lot",
                 "opportun learn",
                 "learn opportun",
                 "learn oppurtun",
                 "learn new",
                 "learn veri fast")

for (i in 1:length(data)) {
  for (j in learn_opport) {
  corpus_cl[[i]][1] = gsub(j, "learning opportunities", corpus_cl[[i]][1])
  }
}

growth_opport = c("growth opportun",
                  "opportun growth",
                  "opportun grow",
                  "opportun growdevelop",
                  "opportun grown",
```

```
                  "career growth",
                  "career grow",
                  "career growrg",
                  "career growthdevelop",
                  "career growthprogress",
                  "lot opportun")

for (i in 1:length(data)) {
  for (j in growth_opport) {
  corpus_cl[[i]][1] = gsub(j, "growth opportunities", corpus_cl[[i]][1])
  }
}

senior_manag = c("senior manag",
                 "upper manag")

for (i in 1:length(data)) {
  for (j in senior_manag) {
  corpus_cl[[i]][1] = gsub(j, "senior management", corpus_cl[[i]][1])
  }
}

low_pay = c("low pay",
            "low salari",
            "low base",
            "low compar",
            "low compens",
            "low wage",
            "lower pay",
            "lower averag",
            "lower avg",
            "lower base",
            "lowest salari",
            "lowest pay",
            "lowish pay",
            "lowish salari",
            "lowno bonus")

for (i in 1:length(data)) {
  for (j in low_pay) {
  corpus_cl[[i]][1] = gsub(j, "low pay", corpus_cl[[i]][1])
  }
}

work_culture = c("work cultur",
                 "compani cultur",
                 "team cultur",
                 "corp cultur",
                 "corpor cultur",
                 "corprat cultur",
                 "corrupt cultur",
                 "cowork cultur",
                 "crappi cultur",
                 "creat cultur",
                 "cultur cultur",
                 "cultur multicultur",
```

```
                 "custom cultur",
                  "decent cultur",
                 "amaz cultur",
                 "opportun cultur",
                 "opportunities cultur",
                 "nice cultur",
                 "offic cultur",
                 "leadership cultur",
                 "learn cultur",
                 "leader cultur",
                 "inclus cultur",
                 "grow cultur",
                 "growth cultur",
                 "firm cultur",
                 "excel cultur"
                 )

for (i in 1:length(data)) {
  for (j in work_culture) {
  corpus_cl[[i]][1] = gsub(j, "company culture", corpus_cl[[i]][1])
  }
}

busy_season = c("busi season",
                 "peak season",
                "tax season"
)

for (i in 1:length(data)) {
  for (j in busy_season) {
  corpus_cl[[i]][1] = gsub(j, "busy season", corpus_cl[[i]][1])
  }
}

high_turnov = c("high turnov","staff turnov","employe turnov","lot turnov", "manag turnov")

for (i in 1:length(data)) {
  for (j in high_turnov) {
  corpus_cl[[i]][1] = gsub(j, "high turnover", corpus_cl[[i]][1])
  }
}

#creating bigrams/trigrams
tokenizer <- function(x)
  NGramTokenizer(x, Weka_control(min=2, max=2))

all_tdm_bi <- TermDocumentMatrix(corpus_cl, control = list(tokenize=tokenizer))

colnames(all_tdm_bi) <- names(data)
all_m_bi <- as.matrix(all_tdm_bi)

df.bi <- as.data.frame(all_m_bi)
df.bi <- rownames_to_column(df.bi, "Term")

x = df.bi %>%
  mutate(term_freq = rowSums(.[2:length(data)])) %>%
```

```r
  mutate(Banks = BMO+CIBC+NationalBank+RBC+Scotiabank+TD,
         Consulting = Deloitte + EY + KPMG + PwC,
         Telecom = Bell+Rogers+Telus+Videotron) %>%
  arrange(desc(term_freq)) %>%
  head(max.words)

  return(x)
}

pros_top_bi = data_prep_bigram(pros.comp.list,max.words = 20)
cons_top_bi = data_prep_bigram(cons.comp.list,max.words = 20)
```

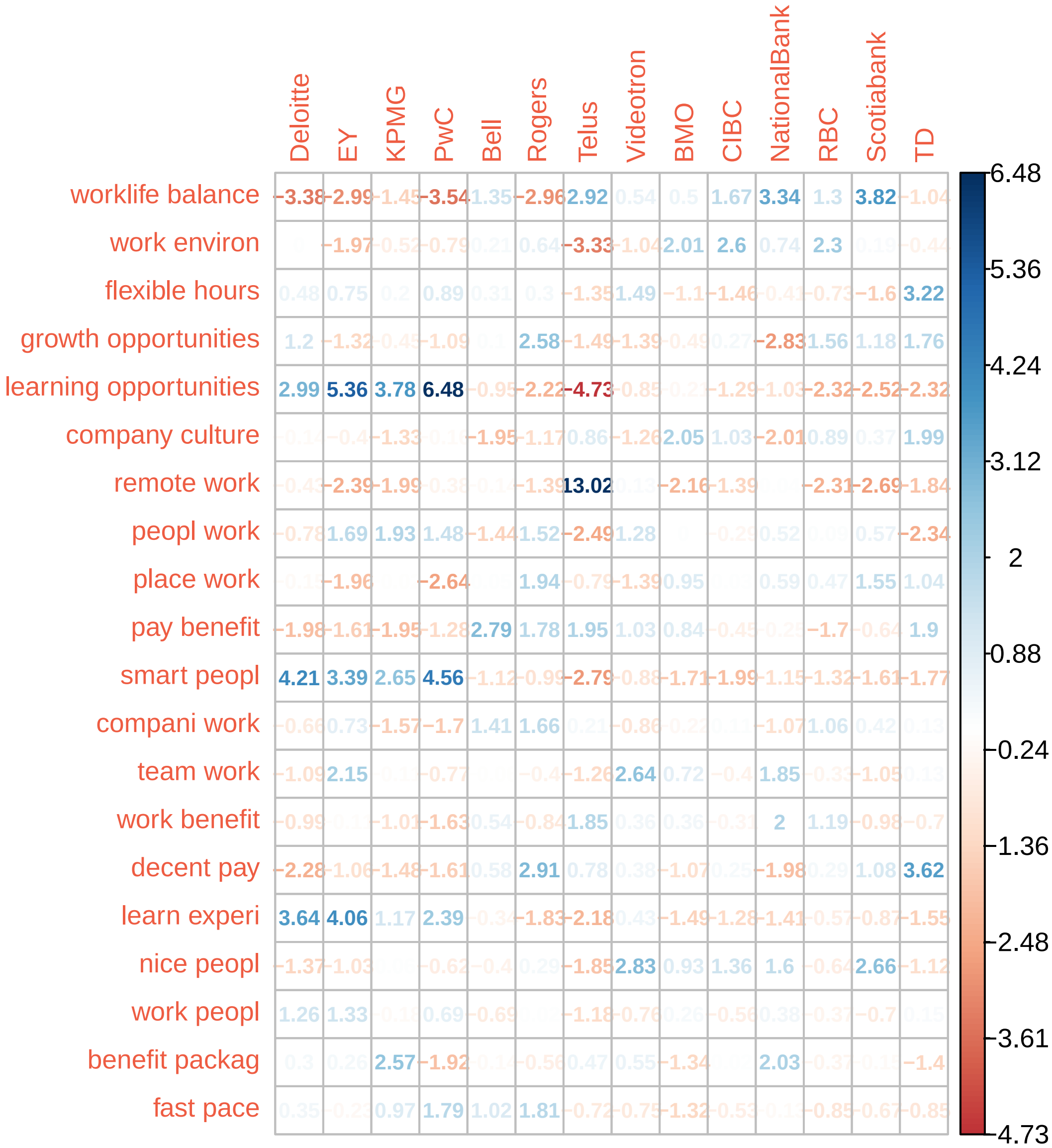
11. Chi-square residulas analysis on bi-grams

Among our bi-grams, we have a mix of those that underwent manual cleaning and those that did not. **The frequencies of the manually treated terms will be higher (and more accurate) than for those not treated, other things being equal** (since we combine several terms into one), so we should not compare frequencies across bi-grams.

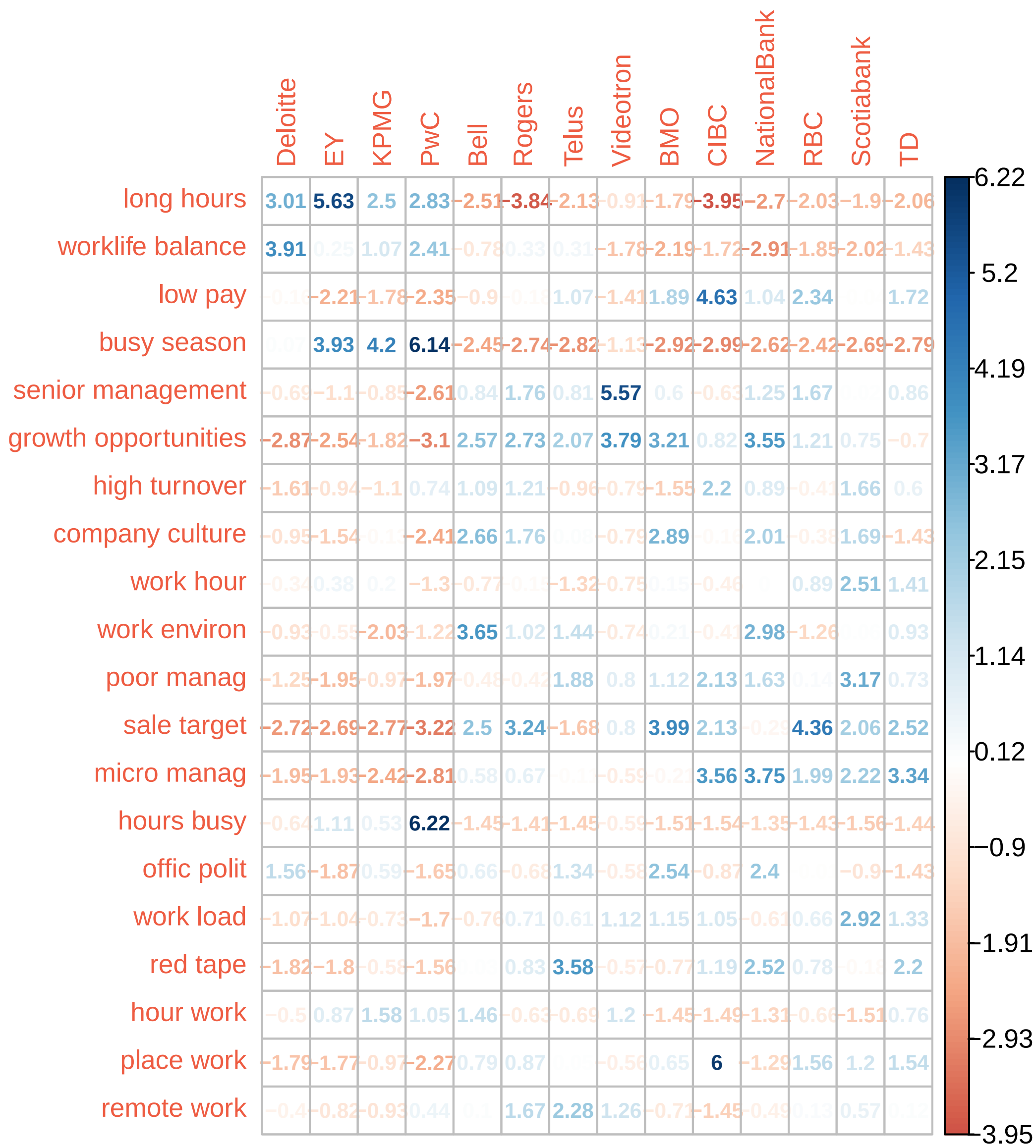
However, **the relative frequency of terms across companies is still meaningful** and shows how companies compare to each other over a given characteristic (term). In this regard, analysis of chi-square residuals shows which terms stand out for which companies. It allows comparing terms across the line (across companies) but it disregards how widespread (frequent) a given bi-gram across our corpus is.

Many findings from the uni-gram analysis are reconfirmed by bi-grams. However, bi-grams allow us to obtain more granular information at the company level. For example, we can learn that several banks have nice work enironment, but there is some micro-management and office polictics, and so on.

```r
pros_sec = pros_top_bi %>%
  # select(-Banks,-Consulting,-Telecom)
  select(Term, Deloitte, EY, KPMG, PwC, Bell, Rogers, Telus, Videotron,
         BMO, CIBC, NationalBank, RBC, Scotiabank, TD)

pros_sec_m = as.matrix(pros_sec[,2:15])
rownames(pros_sec_m) = pros_sec$Term
chisq_all_pros = chisq.test(pros_sec_m)

corrplot(chisq_all_pros$residuals, method = "number", is.corr = FALSE, tl.col = "tomato2", tl.cex = 0.8,
         number.cex = 0.7)
```




```r
cons_sec = cons_top_bi %>%
  # select(-Banks,-Consulting,-Telecom)
  select(Term, Deloitte, EY, KPMG, PwC, Bell, Rogers, Telus, Videotron,
         BMO, CIBC, NationalBank, RBC, Scotiabank, TD)

cons_sec_m = as.matrix(cons_sec[,2:15])
rownames(cons_sec_m) = cons_sec$Term
chisq_all_cons = chisq.test(cons_sec_m)

# chisq_all_cons$residuals[sort(chisq_all_cons$residuals[,1],decreasing=T),]
corrplot(chisq_all_cons$residuals,method = "number", is.corr = FALSE, tl.col = "tomato2", tl.cex = 0.8,
         number.cex = 0.7)
```




```
# # to save the code as .R file
# knitr::purl(input = "TextMining.v3.Rmd", documentation = 2)
```