

Draft_markdown_v1

River Kim

2024-04-07

Set up

Data importing (Following Course Book “Assessment Data”)

```
library(ggplot2)
library(plyr)
library(gdata)

## 
## Attaching package: 'gdata'

## The following object is masked from 'package:stats':
## 
##     nobs

## The following object is masked from 'package:utils':
## 
##     object.size

## The following object is masked from 'package:base':
## 
##    startsWith

library(stringr)
library(data.table)

## 
## Attaching package: 'data.table'

## The following objects are masked from 'package:gdata':
## 
##     first, last

## Prep Osnabrugge et al.
data = fread("/Users/garamkim/Downloads/dataverse_files/uk_data.csv", encoding="UTF-8")
data$date = as.Date(data$date)

#Create time variable
```

```

data$time= NA
data$time[data$date>=as.Date("2001-01-01") & data$date<=as.Date("2001-06-30")] = "01/1"
data$time[data$date>=as.Date("2001-07-01") & data$date<=as.Date("2001-12-31")] = "01/2"
data$time[data$date>=as.Date("2002-01-01") & data$date<=as.Date("2002-06-30")] = "02/1"
data$time[data$date>=as.Date("2002-07-01") & data$date<=as.Date("2002-12-31")] = "02/2"
data$time[data$date>=as.Date("2003-01-01") & data$date<=as.Date("2003-06-30")] = "03/1"
data$time[data$date>=as.Date("2003-07-01") & data$date<=as.Date("2003-12-31")] = "03/2"
data$time[data$date>=as.Date("2004-01-01") & data$date<=as.Date("2004-06-30")] = "04/1"
data$time[data$date>=as.Date("2004-07-01") & data$date<=as.Date("2004-12-31")] = "04/2"
data$time[data$date>=as.Date("2005-01-01") & data$date<=as.Date("2005-06-30")] = "05/1"
data$time[data$date>=as.Date("2005-07-01") & data$date<=as.Date("2005-12-31")] = "05/2"
data$time[data$date>=as.Date("2006-01-01") & data$date<=as.Date("2006-06-30")] = "06/1"
data$time[data$date>=as.Date("2006-07-01") & data$date<=as.Date("2006-12-31")] = "06/2"
data$time[data$date>=as.Date("2007-01-01") & data$date<=as.Date("2007-06-30")] = "07/1"
data$time[data$date>=as.Date("2007-07-01") & data$date<=as.Date("2007-12-31")] = "07/2"
data$time[data$date>=as.Date("2008-01-01") & data$date<=as.Date("2008-06-30")] = "08/1"
data$time[data$date>=as.Date("2008-07-01") & data$date<=as.Date("2008-12-31")] = "08/2"
data$time[data$date>=as.Date("2009-01-01") & data$date<=as.Date("2009-06-30")] = "09/1"
data$time[data$date>=as.Date("2009-07-01") & data$date<=as.Date("2009-12-31")] = "09/2"
data$time[data$date>=as.Date("2010-01-01") & data$date<=as.Date("2010-06-30")] = "10/1"
data$time[data$date>=as.Date("2010-07-01") & data$date<=as.Date("2010-12-31")] = "10/2"
data$time[data$date>=as.Date("2011-01-01") & data$date<=as.Date("2011-06-30")] = "11/1"
data$time[data$date>=as.Date("2011-07-01") & data$date<=as.Date("2011-12-31")] = "11/2"
data$time[data$date>=as.Date("2012-01-01") & data$date<=as.Date("2012-06-30")] = "12/1"
data$time[data$date>=as.Date("2012-07-01") & data$date<=as.Date("2012-12-31")] = "12/2"
data$time[data$date>=as.Date("2013-01-01") & data$date<=as.Date("2013-06-30")] = "13/1"
data$time[data$date>=as.Date("2013-07-01") & data$date<=as.Date("2013-12-31")] = "13/2"
data$time[data$date>=as.Date("2014-01-01") & data$date<=as.Date("2014-06-30")] = "14/1"
data$time[data$date>=as.Date("2014-07-01") & data$date<=as.Date("2014-12-31")] = "14/2"
data$time[data$date>=as.Date("2015-01-01") & data$date<=as.Date("2015-06-30")] = "15/1"
data$time[data$date>=as.Date("2015-07-01") & data$date<=as.Date("2015-12-31")] = "15/2"
data$time[data$date>=as.Date("2016-01-01") & data$date<=as.Date("2016-06-30")] = "16/1"
data$time[data$date>=as.Date("2016-07-01") & data$date<=as.Date("2016-12-31")] = "16/2"
data$time[data$date>=as.Date("2017-01-01") & data$date<=as.Date("2017-06-30")] = "17/1"
data$time[data$date>=as.Date("2017-07-01") & data$date<=as.Date("2017-12-31")] = "17/2"
data$time[data$date>=as.Date("2018-01-01") & data$date<=as.Date("2018-06-30")] = "18/1"
data$time[data$date>=as.Date("2018-07-01") & data$date<=as.Date("2018-12-31")] = "18/2"
data$time[data$date>=as.Date("2019-01-01") & data$date<=as.Date("2019-06-30")] = "19/1"
data$time[data$date>=as.Date("2019-07-01") & data$date<=as.Date("2019-12-31")] = "19/2"

data$time2 = data$time
data$time2 = str_replace(data$time2, "/", "_")

data$stage = 0
data$stage[data$m_questions==1] = 1
data$stage[data$u_questions==1] = 2
data$stage[data$queen_debate_others==1] = 3
data$stage[data$queen_debate_day1==1] = 4
data$stage[data$pm_questions==1] = 5

```

Inspecting data and selecting some parts of data for the research

```

# Packages
library(tidyverse)

## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## vforcats   1.0.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr    1.3.0
## v purrr    1.0.2
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::arrange()     masks plyr::arrange()
## x dplyr::between()    masks data.table::between()
## x dplyr::combine()    masks gdata::combine()
## x purrr::compact()    masks plyr::compact()
## x dplyr::count()      masks plyr::count()
## x dplyr::desc()       masks plyr::desc()
## x dplyr::failwith()   masks plyr::failwith()
## x dplyr::filter()     masks stats::filter()
## x dplyr::first()      masks data.table::first(), gdata::first()
## x lubridate::hour()   masks data.table::hour()
## x dplyr::id()         masks plyr::id()
## x lubridate::isoweek() masks data.table::isoweek()
## x purrr::keep()       masks gdata::keep()
## x dplyr::lag()        masks stats::lag()
## x dplyr::last()       masks data.table::last(), gdata::last()
## x lubridate::mday()   masks data.table::mday()
## x lubridate::minute() masks data.table::minute()
## x lubridate::month()  masks data.table::month()
## x dplyr::mutate()    masks plyr::mutate()
## x lubridate::quarter() masks data.table::quarter()
## x dplyr::rename()    masks plyr::rename()
## x lubridate::second() masks data.table::second()
## x dplyr::starts_with() masks tidyr::starts_with(), gdata::starts_with()
## x dplyr::summarise()  masks plyr::summarise()
## x dplyr::summarize()  masks plyr::summarize()
## x purrr::transpose()  masks data.table::transpose()
## x lubridate::wday()   masks data.table::wday()
## x lubridate::week()   masks data.table::week()
## x lubridate::yday()   masks data.table::yday()
## x lubridate::year()   masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(readr)
library(tidytext)
library(quanteda)

## Package version: 3.3.1
## Unicode version: 14.0
## ICU version: 71.1
## Parallel computing: 8 of 8 threads used.
## See https://quanteda.io for tutorials and examples.

```

```

library(textdata)

#Filtering data columns
colnames(data)

## [1] "id_speech"                  "id_mp"
## [3] "period"                     "last_name"
## [5] "first_name"                 "date"
## [7] "pm_questions"                "queen_debate_day1"
## [9] "queen_debate_others"         "m_questions"
## [11] "u_questions"                 "other_debate"
## [13] "leader"                      "prime_minister"
## [15] "senior_minister"             "shadow"
## [17] "cabinet"                     "chair"
## [19] "government"                 "female"
## [21] "age"                         "electoral_cycle"
## [23] "party"                       "linear_trend"
## [25] "words"                        "text"
## [27] "emotive_count"               "neutral_count"
## [29] "emotive_rhetoric"            "emotive_rhetoric_log"
## [31] "emotive_words"                "top_topic"
## [33] "anew_rescaled"                "emotive_rhetoric_liwc"
## [35] "positive_count"               "negative_count"
## [37] "emotive_positive"              "emotive_negative"
## [39] "emotive_count_250_8"           "neutral_count_250_8"
## [41] "emotive_rhetoric_250_8"        "emotive_count_300_10"
## [43] "neutral_count_300_10"          "emotive_rhetoric_300_10"
## [45] "emotive_count_a1"              "neutral_count_a1"
## [47] "emotive_rhetoric_a1"           "emotive_count_a2"
## [49] "neutral_count_a2"              "emotive_rhetoric_a2"
## [51] "time"                          "time2"
## [53] "stage"

data <- data %>%
  select(last_name, first_name, date, female, age, party, text)
head(data)

##   last_name first_name      date female   age      party
##   <char>     <char>    <Date>  <int> <int>    <char>
## 1: dalyell      tam 2001-06-13     0    69       Labour
## 2: young       george 2001-06-13     0    60 Conservative
## 3: cook        robin 2001-06-13     0    55       Labour
## 4: hague      william 2001-06-13     0    40 Conservative
## 5: kennedy     charles 2001-06-13     0    42 Liberal Democrats
## 6: trimble     david 2001-06-13     0    57 Ulster Unionist Party
##
##
## 1:
## 2:
## 3:
## 4: On behalf of all my right hon and hon Friends on the Opposition Benches I offer our sincerest con
## 5:
## 6:

```

Defining research data by filtering research words

```
# Filtering data containing immigration related words
# Define the keywords to search for
immig_words <- c('immigration', 'immigrant', 'asylum')
visa_words <- "\b(UK)?visas?\b"
all_words <- paste0(c(paste0(immig_words, collapse = "|"), visa_words), collapse = "|")

# lower case text with keywords
tidy_data_notoken <- data %>%
  mutate(desc = tolower(text)) %>%
  filter(grepl(all_words, desc))

# arrange the data by party and count the number of speech
tidy_data_notoken %>%
  group_by(party) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

## # A tibble: 9 x 2
##   party           count
##   <chr>          <int>
## 1 Conservative    7230
## 2 Labour          6251
## 3 Scottish National Party 1104
## 4 Liberal Democrats  954
## 5 Others           105
## 6 Democratic Unionist Party  67
## 7 Plaid Cymru      63
## 8 Green             21
## 9 Ulster Unionist Party  14
```

1. Word frequency of data

Making 'Year' column

```
# Make 'Year' using 'date'
tidy_data_notoken$Year <- format(as.Date(tidy_data_notoken$date), "%Y")
data$Year <- format(as.Date(data$date), "%Y")

# Count the number of text by year and party
text_count <- tidy_data_notoken %>%
  group_by(Year, party) %>%
  summarise(n = n(), .groups = 'drop')

text_count$Year <- as.numeric(as.character(text_count$Year)) # Make 'Year' Column of numeric character.
```

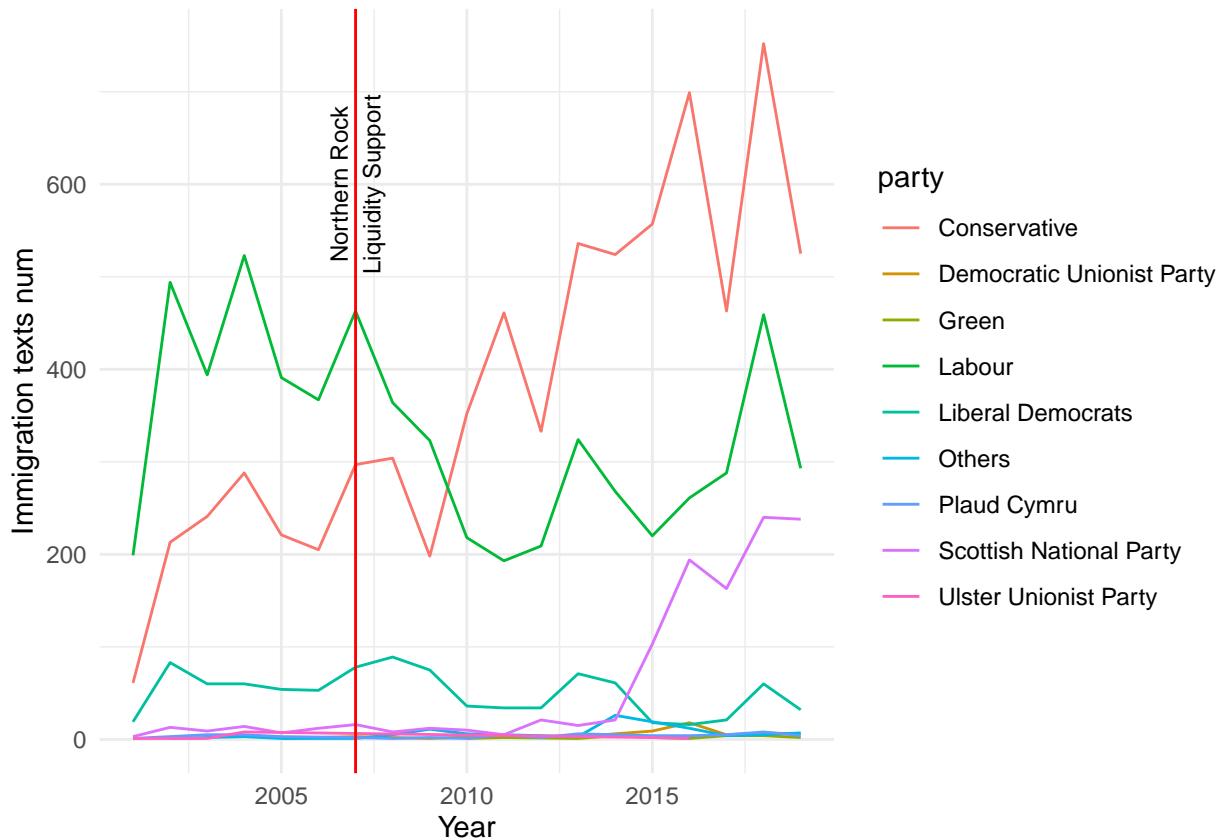
A Simple descriptive plot - The number of immigration related texts by party

```
# Plotting counts of immigration texts by party
ggplot(text_count, aes(x = Year, y = n, group = party, color = party)) +
```

```

geom_line() +
  labs(y = "Immigration texts num", x = "Year") +
  theme_minimal() +
  geom_vline(xintercept = as.numeric(format(as.Date("2007-09-14"), "%Y")), col="red") +
  annotate("text", x = as.numeric(format(as.Date("2007-09-14"), "%Y"))), y = 600, label="Northern Rock\\nLiquidity Support"

```



Dividing the number of immigration texts by the total number of texts

```

# Count the total number of texts in data by year and party
data_count <- data %>%
  group_by(Year, party) %>%
  summarise(total_n = n(), .groups = 'drop')

# Merge the counts and calculate the ratio
ratio_count <- merge(text_count, data_count, by = c("Year", "party"))
ratio_count$ratio <- with(ratio_count, n / total_n)

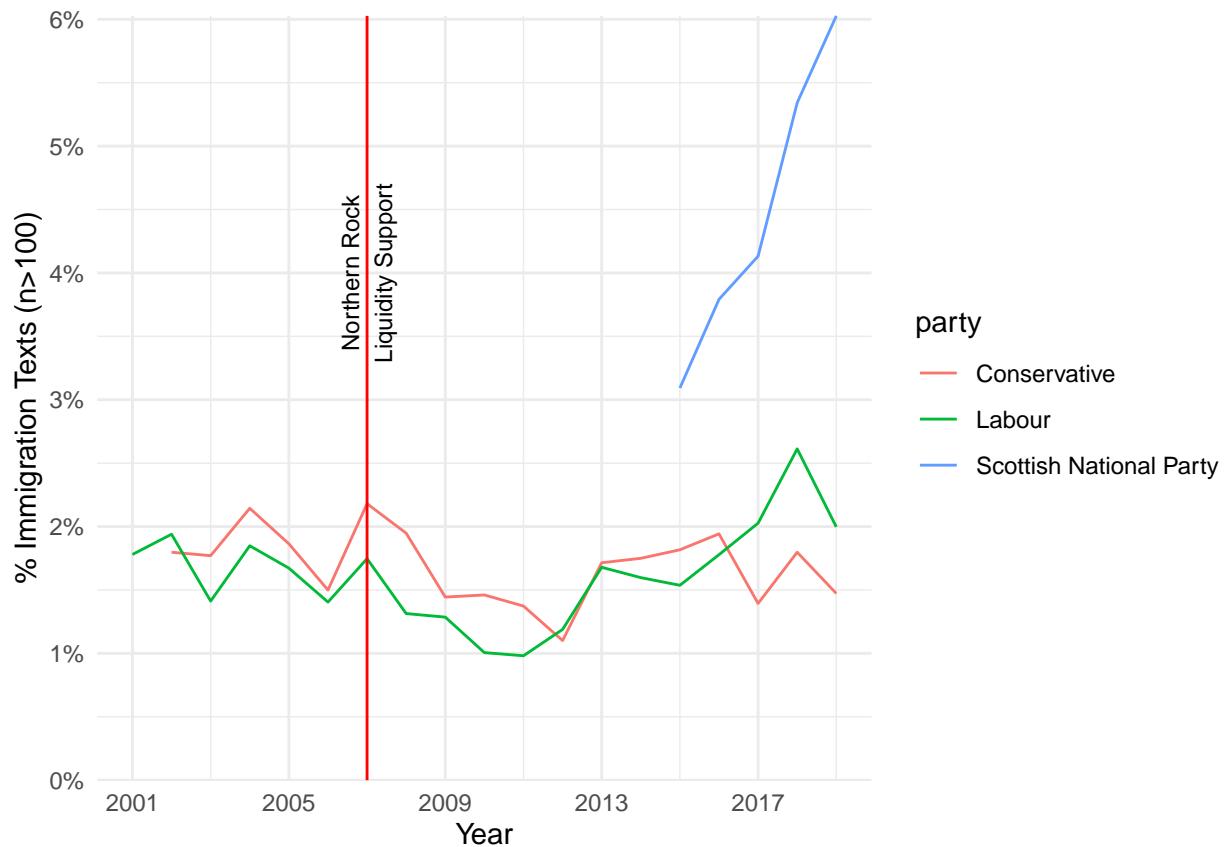
# Only counts more than 100 texts
filtered_ratio_count <- ratio_count %>%
  filter(n > 100)

filtered_ratio_count$Year <- as.numeric(as.character(filtered_ratio_count$Year))

```

A ratio plot of a number of immigration related texts by total texts

```
#Plotting the ratios of immigration related texts
ggplot(filtered_ratio_count, aes(x = Year, y = ratio, group = party, color = party)) +
  geom_line() +
  labs(y = "% Immigration Texts (n>100)", x = "Year") +
  scale_y_continuous(labels = scales::percent_format(), expand = c(0, 0), limits = c(0, NA)) +
  scale_x_continuous(breaks = seq(min(filtered_ratio_count$Year), max(filtered_ratio_count$Year), by = 1))
  theme_minimal() +
  geom_vline(xintercept = as.numeric(format(as.Date("2007-09-14"), "%Y")), col="red") +
  annotate("text", x = as.numeric(format(as.Date("2007-09-14"), "%Y")), y = 0.04, label="Northern Rock\\Liquidity Support")
```



Tokenisation and processing stop words

```
# Tokenisation & removing stop words
tidy_data <- tidy_data_notoken %>%
  unnest_tokens(word, desc) %>%
  filter(str_detect(word, "[a-z]")) %>%
  filter(!word %in% stop_words$word)

tidy_data <- tidy_data %>%
  arrange(date)
tidy_data$order <- 1:nrow(tidy_data) # Make orders of each word
```

```
# Common tokens
word_count <- tidy_data %>%
  count(word, sort = T)
```

```

show(word_count)

##          word      n
##          <char> <int>
## 1:    people 27644
## 2:      hon 26615
## 3: government 26551
## 4: immigration 19105
## 5: minister 14229
##   ---
## 39501: énarques     1
## 39502: ørsted      1
## 39503: þæt         1
## 39504: šefcovic    1
## 39505: štefan       1

# Common tokens by year
word_count_year <- tidy_data %>%
  group_by(Year) %>%
  count(word, sort = T)

show(word_count_year)

## # A tibble: 226,043 x 3
## # Groups: Year [19]
##   Year   word      n
##   <chr> <chr> <int>
## 1 2018 government 2122
## 2 2018 people     2060
## 3 2015 people     2058
## 4 2018 immigration 2046
## 5 2004 government 2007
## 6 2002 people     1883
## 7 2018 hon        1851
## 8 2004 hon        1826
## 9 2002 hon        1792
## 10 2007 hon       1792
## # i 226,033 more rows

# Comon tokens by party
word_count_party <- tidy_data %>%
  group_by(party) %>%
  count(word, sort = T)

show(word_count_party)

## # A tibble: 98,986 x 3
## # Groups: party [9]
##   party   word      n
##   <chr> <chr> <int>
## 1 Labour people 12163

```

```

## 2 Conservative government 12117
## 3 Labour hon 12010
## 4 Conservative hon 11848
## 5 Conservative people 11229
## 6 Labour government 9743
## 7 Conservative immigration 9001
## 8 Labour immigration 7336
## 9 Conservative minister 6290
## 10 Conservative country 6258
## # i 98,976 more rows

```

2. Sentiment Analysis Using ‘NRC’ Sentiment Dictionary

1) Total sentiment of immigration related texts

Getting NRC sentiment dictionary and tidying data by calculating sentiment ratio

```
get_sentiments("nrc")
```

```

## # A tibble: 13,872 x 2
##   word      sentiment
##   <chr>     <chr>
## 1 abacus    trust
## 2 abandon   fear
## 3 abandon   negative
## 4 abandon   sadness
## 5 abandoned anger
## 6 abandoned fear
## 7 abandoned negative
## 8 abandoned sadness
## 9 abandonment anger
## 10 abandonment fear
## # i 13,862 more rows

```

```

# Make nrc sentiment tables of data
nrc_data <- tidy_data %>%
  inner_join(get_sentiments("nrc"), by = "word") %>%
  count(date, sentiment) %>%
  spread(key = sentiment, value = n, fill = 0) %>%
  mutate(ratio = negative / (positive+1)) # Calculating negative/positive ratio. Adding 1 to avoid the

```

```

## Warning in inner_join(., get_sentiments("nrc"), by = "word"): Detected an unexpected many-to-many relationship between 'date' and 'word'.
## i Row 6 of 'x' matches multiple rows in 'y'.
## i Row 5657 of 'y' matches multiple rows in 'x'.
## i If a many-to-many relationship is expected, set 'relationship =
##   "many-to-many"' to silence this warning.

```

```
# Check sentiments
colnames(nrc_data)
```

```

## [1] "date"           "anger"          "anticipation"   "disgust"        "fear"
## [6] "joy"            "negative"       "positive"       "sadness"       "surprise"
## [11] "trust"          "ratio"

```

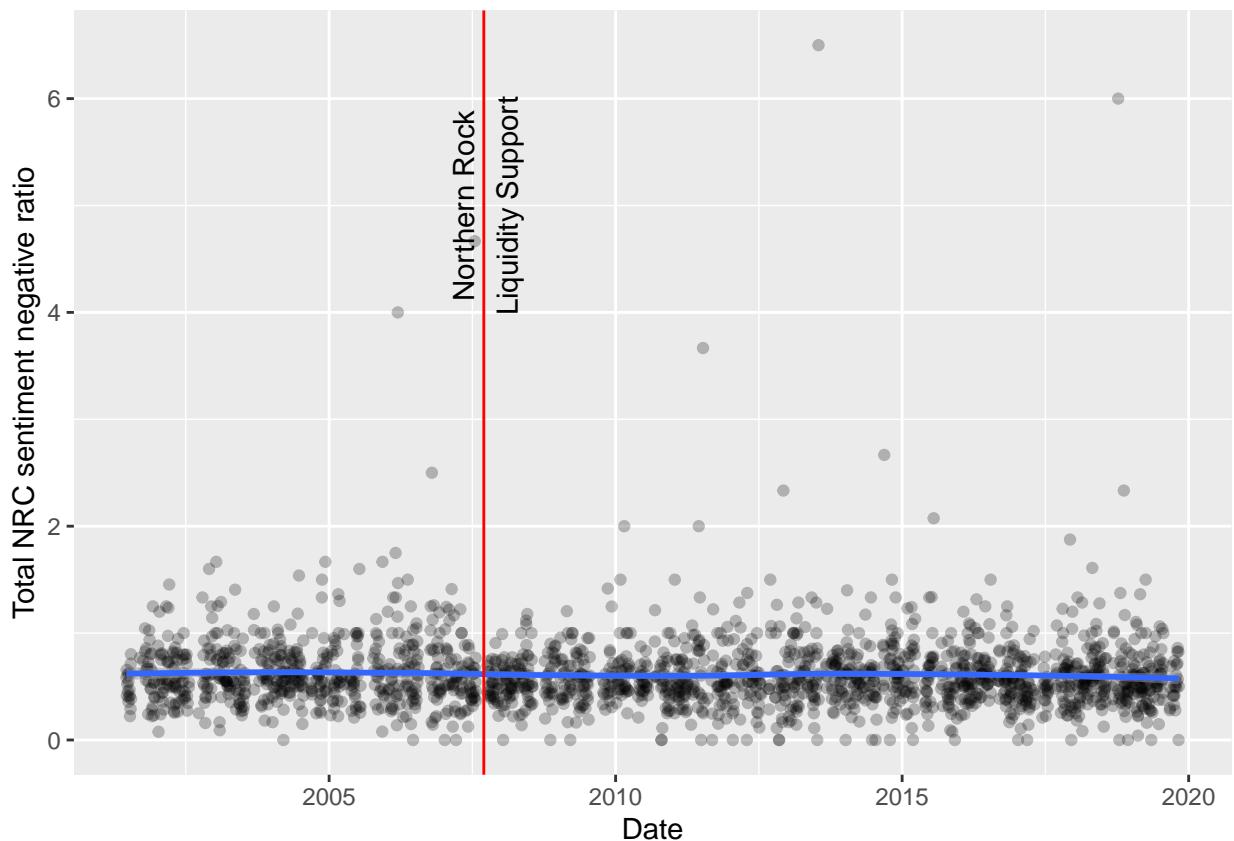
Plotting the total negative ratio

```

# Total sentiment of text including immigration related words plot (Using negative ratio)
nrc_data %>%
  ggplot(aes(date, ratio)) +
  geom_point(alpha=0.25) +
  geom_smooth(method="loess", alpha=0.5) +
  labs(y = "Total NRC sentiment negative ratio", x = "Date") +
  geom_vline(xintercept = as.numeric(as.Date("2007-09-14")), col="red") +
  annotate("text", x = as.Date("2007-09-24"), y = 5, label="Northern Rock\nLiquidity Support", angle=90)

## `geom_smooth()` using formula = 'y ~ x'

```



There is no time when particularly negative emotions are revealed. Rather, negative feelings toward immigrants are less visible after the financial crisis.

2) Negative ratio plots by party

Calculating NRC sentiment negative ratios of texts which contain immigration related words *by party*

```

# Arranging data by party and calculate the ratio of negative sentiment.
nrc_data_party <- tidy_data %>%
  inner_join(get_sentiments("nrc"), by = "word") %>%
  group_by(party, date) %>%
  count(sentiment) %>%
  spread(key = sentiment, value = n, fill = 0) %>%
  mutate(ratio = negative/(positive+1)) %>%
  ungroup()

## Warning in inner_join(., get_sentiments("nrc"), by = "word"): Detected an unexpected many-to-many re...
## i Row 6 of 'x' matches multiple rows in 'y'.
## i Row 5657 of 'y' matches multiple rows in 'x'.
## i If a many-to-many relationship is expected, set 'relationship' =
##   "many-to-many" to silence this warning.

filtered_nrc_data_party <-nrc_data_party %>%
  group_by(party) %>%
  filter(n() > 100) %>% # party with more than 100 related words
  ungroup()

```

Plotting a negative ratio plot of party. Points indicate the actual scores while smooth lines show the trends.

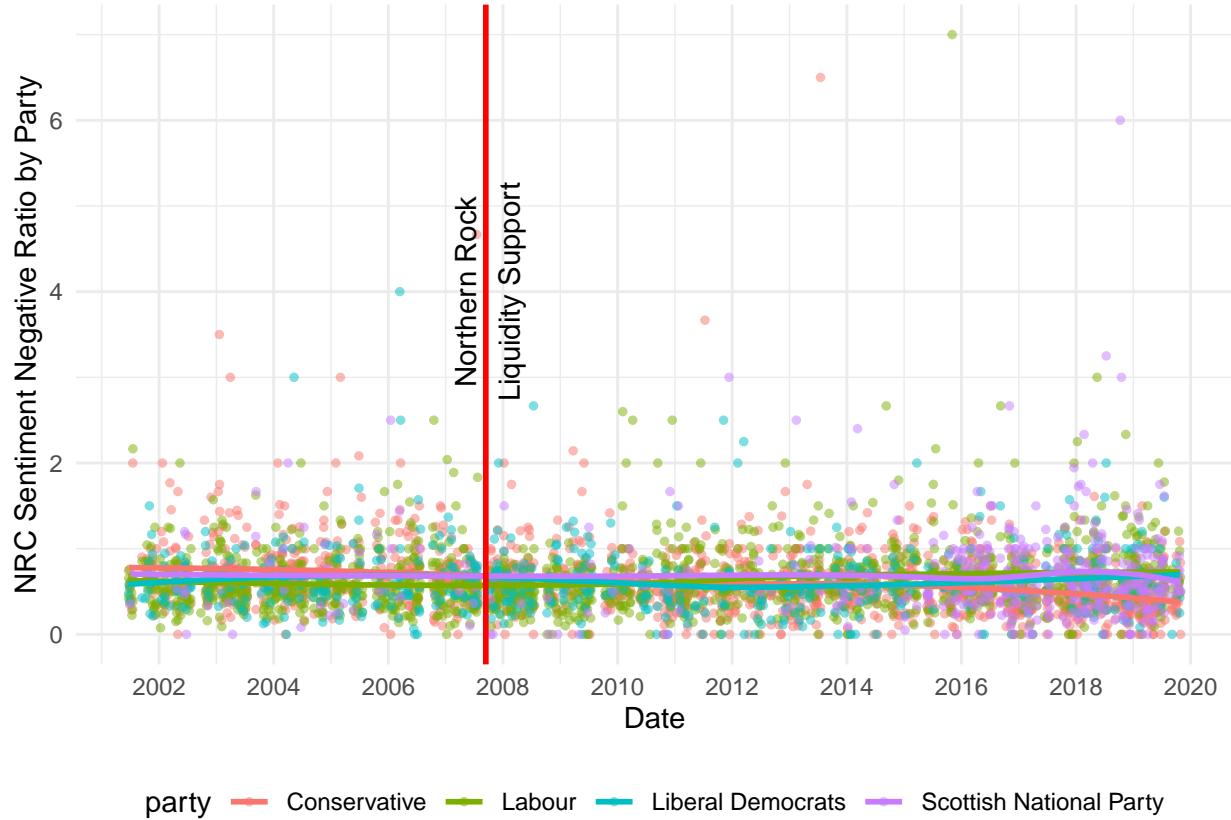
```

# A negative sentiment ratio plot of four parties
filtered_nrc_data_party %>%
  ggplot(aes(date, ratio, color = party)) +
  geom_point(alpha=0.5, size = 1) +
  geom_smooth(method="loess", se = F, alpha=0.7, size = 1) +
  scale_x_date(date_breaks = "2 years", date_labels = "%Y") +
  labs(y = "NRC Sentiment Negative Ratio by Party", x = "Date") +
  geom_vline(xintercept = as.numeric(as.Date("2007-09-14")), col="red", size = 1) +
  annotate("text", x = as.Date("2007-09-24"), y = 4, label="Northern Rock\nLiquidity Support", angle=90
  theme_minimal() +
  theme(legend.position = "bottom")

## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.

## 'geom_smooth()' using formula = 'y ~ x'

```

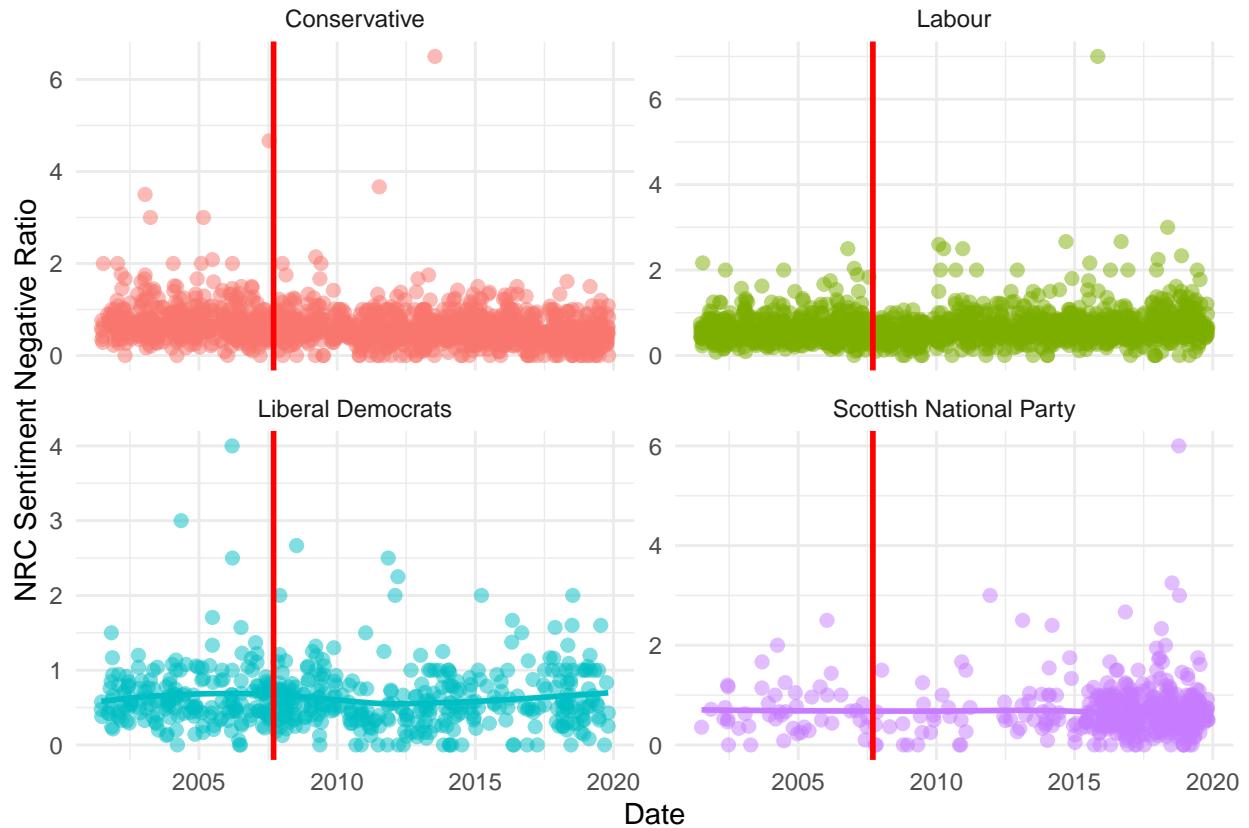


Similar to the overall sentiment plot, there is no period when the proportion of negative words appears prominently.

Make the plot above readable, dividing plots by party

```
# Faceting by party
filtered_nrc_data_party %>%
  ggplot(aes(x = date, y = ratio, color = party)) +
  geom_point(alpha = 0.5, size = 2) +
  geom_smooth(method = "loess", se = FALSE, alpha = 0.7, size = 1) +
  labs(y = "NRC Sentiment Negative Ratio", x = "Date") +
  facet_wrap(~party, scales = "free_y") +
  geom_vline(xintercept = as.numeric(as.Date("2007-09-14")), col = "red", size = 1) +
  theme_minimal() +
  theme(legend.position = "none")

## `geom_smooth()` using formula = 'y ~ x'
```



Similar to the overall sentiment plot, there is no period when the proportion of negative words appears prominently.

3) Sentiments counts by party

What is the most common sentiment? Summing up scores of sentiments by party to compare.

```
# Total sentiment word counts by party
dominant_senti_nrc_party <- filtered_nrc_data_party %>%
  group_by(party) %>%
  summarise(across(c(anger, anticipation, disgust, fear, joy, sadness, surprise, trust, negative, positive), sum))

## Warning: There was 1 warning in `summarise()` .
## i In argument `across(...)` .
## i In group 1: `party = "Conservative"` .
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
## # Previously
##   across(a:b, mean, na.rm = TRUE)
##
## # Now
##   across(a:b, \((x) mean(x, na.rm = TRUE)))
```

```

show(dominant_senti_nrc_party)

## # A tibble: 4 x 11
##   party    anger anticipation disgust   fear    joy sadness surprise trust negative
##   <chr>    <dbl>        <dbl>    <dbl> <dbl>    <dbl>    <dbl> <dbl>    <dbl>
## 1 Conser~ 33439       52127    17861  56500  36474    32322   17703  99258   84545
## 2 Labour   35105       53846    18420  57405  38456    34318   18408  97785   85101
## 3 Libera~  5625        9303     2963   9620   5875     5778    3149   15899   14675
## 4 Scotti~  4637        7398     2523   8180   4986     5388    2638   12649   12877
## # i 1 more variable: positive <dbl>

```

The large number of sentiment counts is generated by the Conservatives and Labor parties.

4) Sentiment changes by party

Beyond negative/positive ratio, focusing on specific sentiments trends by party.

```

# Combining all sentiments into one column
long_nrc_data <- filtered_nrc_data_party %>%
  pivot_longer(cols = c(anger, fear, trust, sadness, disgust, anticipation, surprise, joy), names_to =

```

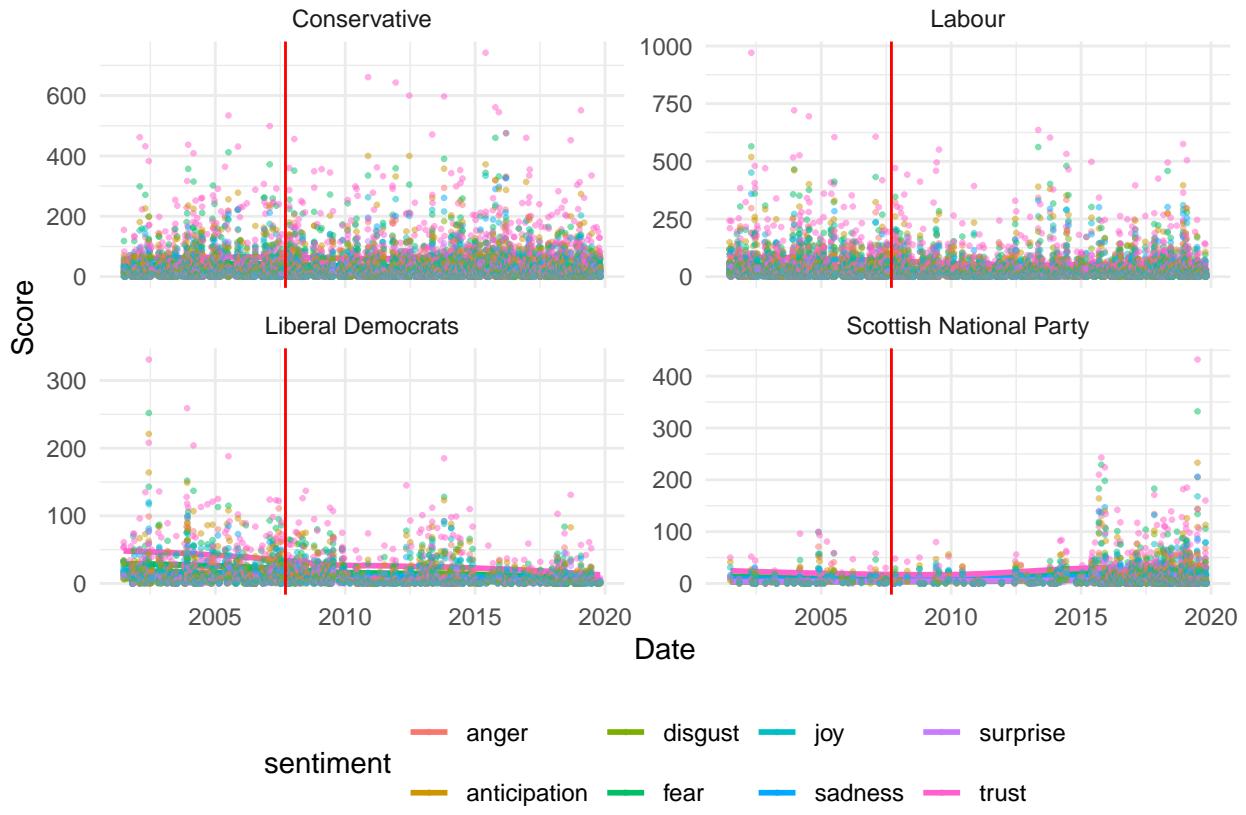
Faceting plots to compare the sentiment changes of each party

```

# Plotting specific sentiment changes by party
long_nrc_data %>%
  ggplot(aes(x = date, y = score, color = sentiment)) +
  geom_smooth(method = "loess", se = F, alpha = 1, size = 1) +
  geom_point(alpha = 0.5, size = 0.5) +
  facet_wrap(~ party, scales = "free_y") +
  labs(x = "Date", y = "Score") +
  theme_minimal() +
  geom_vline(xintercept = as.numeric(as.Date("2007-09-14")), col = "red") +
  theme_minimal() +
  theme(legend.position = "bottom")

## `geom_smooth()` using formula = 'y ~ x'

```



The smooth line in each plot shows a relatively steady and unchanged shape.

To make sentiment points recognizable, adjusting the shapes of points by sentiment and plotting

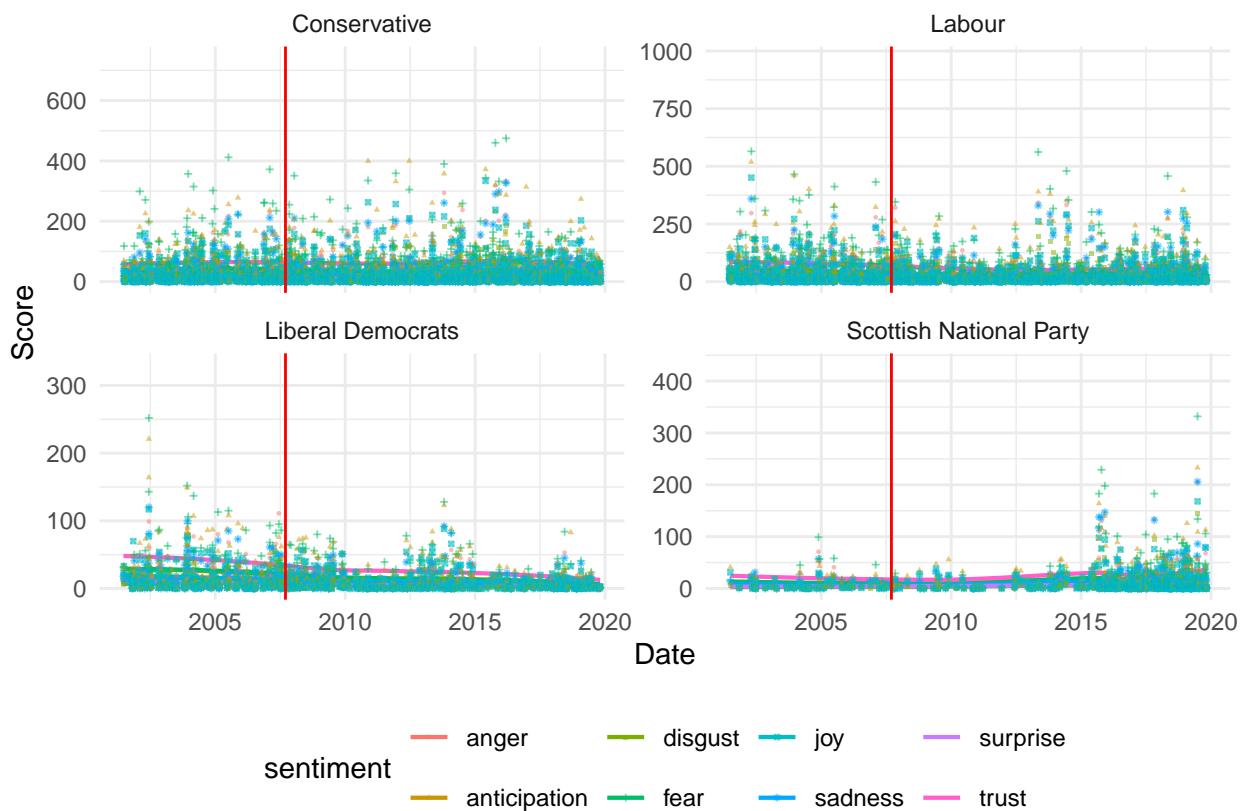
```
# Shaping sentiment points differently
shape_mapping <- c("anger" = 17, "disgust" = 18, "fear" = 19, "sadness" = 8, "trust" = 21, "anticipation" = 20)

long_nrc_data %>%
  ggplot(aes(x = date, y = score, color = sentiment)) +
  geom_smooth(method = "loess", se = F, alpha = 0.75, size = 0.75) +
  geom_point(aes(shape = sentiment), alpha = 0.5, size = 0.5) +
  facet_wrap(~ party, scales = "free_y", nrow = 2) +
  labs(x = "Date", y = "Score") +
  geom_vline(xintercept = as.numeric(as.Date("2007-09-14")), col = "red") +
  theme_minimal() +
  theme(legend.position = "bottom")

## `geom_smooth()` using formula = 'y ~ x'

## Warning: The shape palette can deal with a maximum of 6 discrete values because more
## than 6 becomes difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need
## that many have them.

## Warning: Removed 8132 rows containing missing values or values outside the scale range
## ('geom_point()').
```

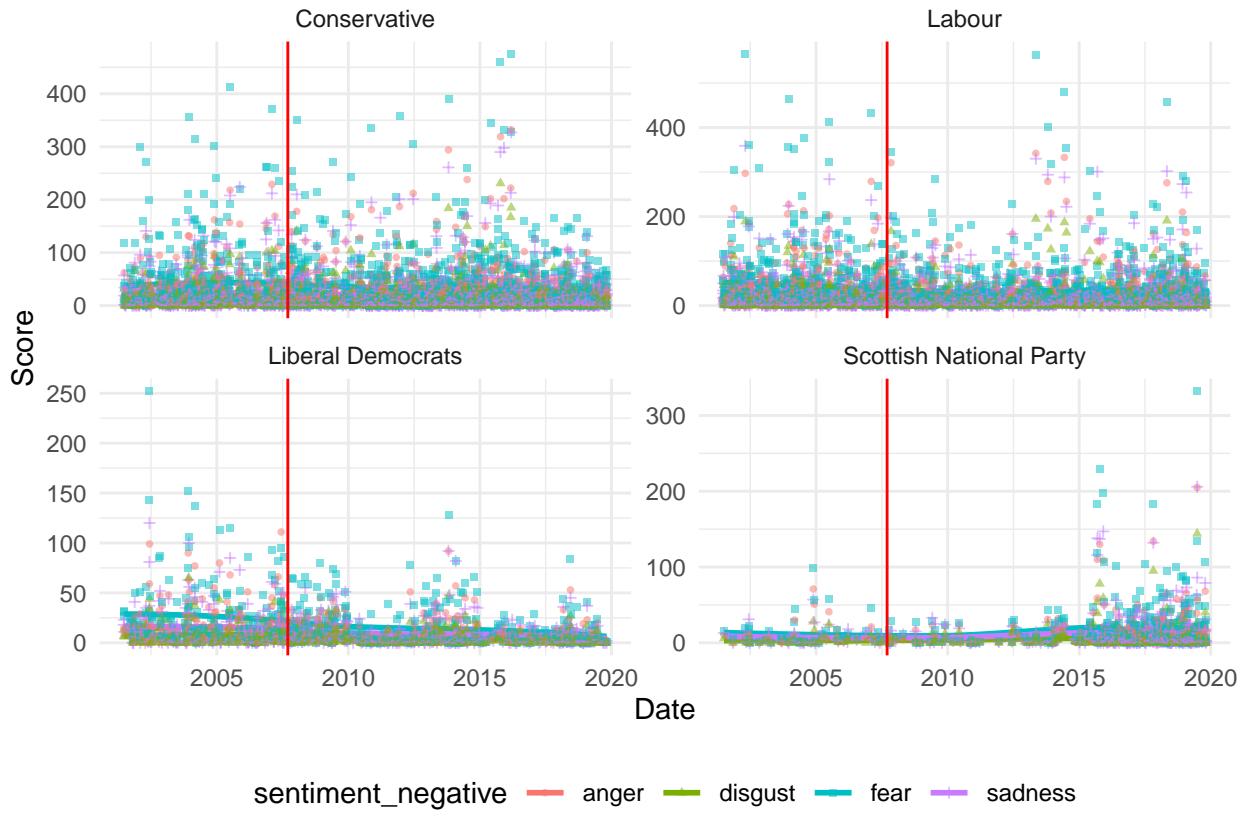


Selecting negative sentiments to confirm sentiment changes

```
# Specifying negative feelings only
long_nrc_data_neg <- filtered_nrc_data_party %>%
  pivot_longer(cols = c(anger, fear, sadness, disgust), names_to = "sentiment_negative", values_to = "score")

# Plots of specific negative sentiments by party
long_nrc_data_neg %>%
  ggplot(aes(x = date, y = score, color = sentiment_negative)) +
  geom_smooth(method = "loess", se = F, alpha = 0.75, size = 1) +
  geom_point(aes(shape = sentiment_negative), alpha = 0.5, size = 1) +
  facet_wrap(~ party, scales = "free_y", nrow = 2) +
  labs(x = "Date", y = "Score") +
  geom_vline(xintercept = as.numeric(as.Date("2007-09-14")), col = "red") +
  theme_minimal() +
  theme(legend.position = "bottom")

## `geom_smooth()` using formula = 'y ~ x'
```



Regarding negative sentiments, the Conservative party presents more diverse and fluctuated points than other parties. However, the general smooth lines stay steady. ## 5) Random sample sentiment

To compare with sentiment trends of immigration related text, make a random sample of 10000 and do sentiment analysis

```
# Random sampling
data_sample <- data %>%
  sample_n(10000)

# tidy sample
tidy_samp <- data_sample %>%
  mutate(desc = tolower(text)) %>%
  unnest_tokens(word, desc) %>%
  filter(str_detect(word, "[a-z]")) %>%
  filter(!word %in% stop_words$word) %>%
  arrange(date)

tidy_samp$order <- 1:nrow(tidy_samp)

# Applying NRC dictionary and calculating total negative ratio
samp_nrc_data <- tidy_samp %>%
  inner_join(get_sentiments("nrc"), by = "word") %>%
  count(date, sentiment) %>%
  spread(key = sentiment, value = n, fill = 0) %>%
  mutate(ratio = negative / (positive+1))
```

```

## Warning in inner_join(., get_sentiments("nrc"), by = "word"): Detected an unexpected many-to-many re...
## i Row 1 of 'x' matches multiple rows in 'y'.
## i Row 12673 of 'y' matches multiple rows in 'x'.
## i If a many-to-many relationship is expected, set 'relationship =
##   "many-to-many"' to silence this warning.

```

#Applying NRC dictionary and calculating negative ratio by party

```

samp_nrc_party <- tidy_samp %>%
  inner_join(get_sentiments("nrc"), by = "word") %>%
  group_by(party, date) %>%
  count(sentiment) %>%
  spread(key = sentiment, value = n, fill = 0) %>%
  mutate(ratio = negative / (positive+1)) %>%
  ungroup()

```

```

## Warning in inner_join(., get_sentiments("nrc"), by = "word"): Detected an unexpected many-to-many re...
## i Row 1 of 'x' matches multiple rows in 'y'.
## i Row 12673 of 'y' matches multiple rows in 'x'.
## i If a many-to-many relationship is expected, set 'relationship =
##   "many-to-many"' to silence this warning.

```

```

filtered_samp_nrc_party <- samp_nrc_party %>%
  group_by(party) %>%
  filter(n() > 100) %>%
  ungroup()

```

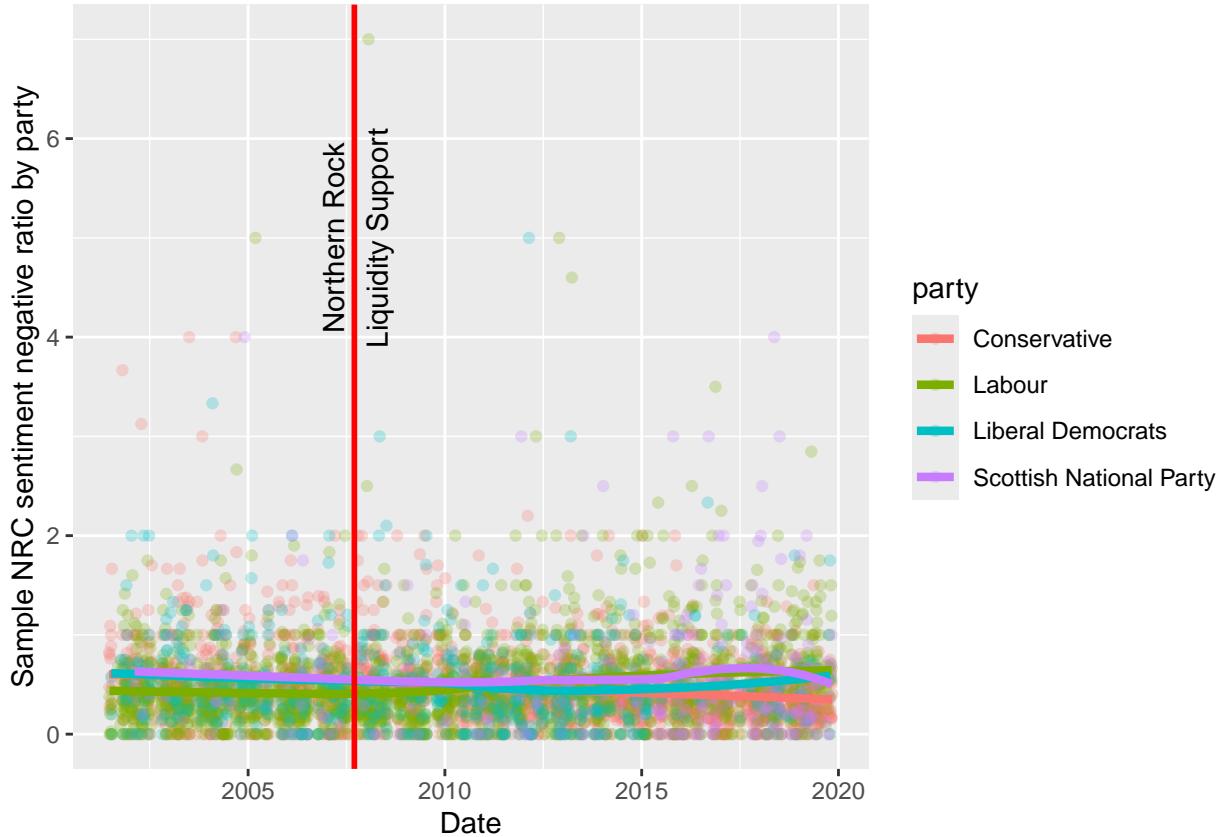
A plot of sample text negative sentiment ratio by party

```

filtered_samp_nrc_party %>%
  ggplot(aes(date, ratio, color = party)) +
  geom_point(alpha=0.25) +
  geom_smooth(method="loess", se = F, alpha=0.25, size = 1.5) +
  labs(y = "Sample NRC sentiment negative ratio by party", x = "Date") +
  geom_vline(xintercept = as.numeric(as.Date("2007-09-14")), col = "red", size = 1) +
  annotate("text", x = as.Date("2007-09-24"), y = 5, label="Northern Rock\nLiquidity Support", angle=90)

## 'geom_smooth()' using formula = 'y ~ x'

```



```
## List of 136
## $ line                               :List of 6
##   ..$ colour      : chr "black"
##   ..$ linewidth   : num 0.5
##   ..$ linetype    : num 1
##   ..$ lineend     : chr "butt"
##   ..$ arrow       : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ...- attr(*, "class")= chr [1:2] "element_line" "element"
## $ rect                                :List of 5
##   ..$ fill       : chr "white"
##   ..$ colour     : chr "black"
##   ..$ linewidth  : num 0.5
##   ..$ linetype   : num 1
##   ..$ inherit.blank: logi TRUE
##   ...- attr(*, "class")= chr [1:2] "element_rect" "element"
## $ text                                 :List of 11
##   ..$ family     : chr ""
##   ..$ face       : chr "plain"
##   ..$ colour     : chr "black"
##   ..$ size        : num 11
##   ..$ hjust      : num 0.5
```

```

## ..$ vjust      : num 0.5
## ..$ angle      : num 0
## ..$ lineheight : num 0.9
## ..$ margin      : 'margin' num [1:4] 0points 0points 0points 0points
## ... - attr(*, "unit")= int 8
## ..$ debug      : logi FALSE
## ..$ inherit.blank: logi TRUE
## ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ title          : NULL
## $ aspect.ratio   : NULL
## $ axis.title     : NULL
## $ axis.title.x    :List of 11
##   ..$ family      : NULL
##   ..$ face        : NULL
##   ..$ colour      : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
## ... - attr(*, "unit")= int 8
## ..$ debug      : NULL
## ..$ inherit.blank: logi TRUE
## ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ axis.title.x.top   :List of 11
##   ..$ family      : NULL
##   ..$ face        : NULL
##   ..$ colour      : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
## ... - attr(*, "unit")= int 8
## ..$ debug      : NULL
## ..$ inherit.blank: logi TRUE
## ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ axis.title.x.bottom : NULL
## $ axis.title.y    :List of 11
##   ..$ family      : NULL
##   ..$ face        : NULL
##   ..$ colour      : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
## ... - attr(*, "unit")= int 8
## ..$ debug      : NULL
## ..$ inherit.blank: logi TRUE
## ..- attr(*, "class")= chr [1:2] "element_text" "element"

```

```

## $ axis.title.y.left : NULL
## $ axis.title.y.right :List of 11
## ..$ family : NULL
## ..$ face : NULL
## ..$ colour : NULL
## ..$ size : NULL
## ..$ hjust : NULL
## ..$ vjust : num 1
## ..$ angle : num -90
## ..$ lineheight : NULL
## ..$ margin : 'margin' num [1:4] 0points 0points 0points 2.75points
## ... - attr(*, "unit")= int 8
## ..$ debug : NULL
## ..$ inherit.blank: logi TRUE
## ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ axis.text :List of 11
## ..$ family : NULL
## ..$ face : NULL
## ..$ colour : chr "grey30"
## ..$ size : 'rel' num 0.8
## ..$ hjust : NULL
## ..$ vjust : NULL
## ..$ angle : NULL
## ..$ lineheight : NULL
## ..$ margin : NULL
## ..$ debug : NULL
## ..$ inherit.blank: logi TRUE
## ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ axis.text.x :List of 11
## ..$ family : NULL
## ..$ face : NULL
## ..$ colour : NULL
## ..$ size : NULL
## ..$ hjust : NULL
## ..$ vjust : num 1
## ..$ angle : NULL
## ..$ lineheight : NULL
## ..$ margin : 'margin' num [1:4] 2.2points 0points 0points 0points
## ... - attr(*, "unit")= int 8
## ..$ debug : NULL
## ..$ inherit.blank: logi TRUE
## ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ axis.text.x.top :List of 11
## ..$ family : NULL
## ..$ face : NULL
## ..$ colour : NULL
## ..$ size : NULL
## ..$ hjust : NULL
## ..$ vjust : num 0
## ..$ angle : NULL
## ..$ lineheight : NULL
## ..$ margin : 'margin' num [1:4] 0points 0points 2.2points 0points
## ... - attr(*, "unit")= int 8
## ..$ debug : NULL

```

```

## ..$ inherit.blank: logi TRUE
## ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ axis.text.x.bottom : NULL
## $ axis.text.y :List of 11
##   ..$ family : NULL
##   ..$ face : NULL
##   ..$ colour : NULL
##   ..$ size : NULL
##   ..$ hjust : num 1
##   ..$ vjust : NULL
##   ..$ angle : NULL
##   ..$ lineheight : NULL
##   ..$ margin : 'margin' num [1:4] 0points 2.2points 0points 0points
##   ... ..- attr(*, "unit")= int 8
##   ..$ debug : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ axis.text.y.left : NULL
## $ axis.text.y.right :List of 11
##   ..$ family : NULL
##   ..$ face : NULL
##   ..$ colour : NULL
##   ..$ size : NULL
##   ..$ hjust : num 0
##   ..$ vjust : NULL
##   ..$ angle : NULL
##   ..$ lineheight : NULL
##   ..$ margin : 'margin' num [1:4] 0points 0points 0points 2.2points
##   ... ..- attr(*, "unit")= int 8
##   ..$ debug : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ axis.text.theta : NULL
## $ axis.text.r :List of 11
##   ..$ family : NULL
##   ..$ face : NULL
##   ..$ colour : NULL
##   ..$ size : NULL
##   ..$ hjust : num 0.5
##   ..$ vjust : NULL
##   ..$ angle : NULL
##   ..$ lineheight : NULL
##   ..$ margin : 'margin' num [1:4] 0points 2.2points 0points 2.2points
##   ... ..- attr(*, "unit")= int 8
##   ..$ debug : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ axis.ticks : list()
## ..- attr(*, "class")= chr [1:2] "element_blank" "element"
## $ axis.ticks.x : NULL
## $ axis.ticks.x.top : NULL
## $ axis.ticks.x.bottom : NULL
## $ axis.ticks.y : NULL
## $ axis.ticks.y.left : NULL

```

```

## $ axis.ticks.y.right : NULL
## $ axis.ticks.theta : NULL
## $ axis.ticks.r : NULL
## $ axis.minor.ticks.x.top : NULL
## $ axis.minor.ticks.x.bottom : NULL
## $ axis.minor.ticks.y.left : NULL
## $ axis.minor.ticks.y.right : NULL
## $ axis.minor.ticks.theta : NULL
## $ axis.minor.ticks.r : NULL
## $ axis.ticks.length : 'simpleUnit' num 2.75points
## ..- attr(*, "unit")= int 8
## $ axis.ticks.length.x : NULL
## $ axis.ticks.length.x.top : NULL
## $ axis.ticks.length.x.bottom : NULL
## $ axis.ticks.length.y : NULL
## $ axis.ticks.length.y.left : NULL
## $ axis.ticks.length.y.right : NULL
## $ axis.ticks.length.theta : NULL
## $ axis.ticks.length.r : NULL
## $ axis.minor.ticks.length : 'rel' num 0.75
## $ axis.minor.ticks.length.x : NULL
## $ axis.minor.ticks.length.x.top : NULL
## $ axis.minor.ticks.length.x.bottom: NULL
## $ axis.minor.ticks.length.y : NULL
## $ axis.minor.ticks.length.y.left : NULL
## $ axis.minor.ticks.length.y.right : NULL
## $ axis.minor.ticks.length.theta : NULL
## $ axis.minor.ticks.length.r : NULL
## $ axis.line : list()
## ..- attr(*, "class")= chr [1:2] "element_blank" "element"
## $ axis.line.x : NULL
## $ axis.line.x.top : NULL
## $ axis.line.x.bottom : NULL
## $ axis.line.y : NULL
## $ axis.line.y.left : NULL
## $ axis.line.y.right : NULL
## $ axis.line.theta : NULL
## $ axis.line.r : NULL
## $ legend.background : list()
## ..- attr(*, "class")= chr [1:2] "element_blank" "element"
## $ legend.margin : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
## ..- attr(*, "unit")= int 8
## $ legend.spacing : 'simpleUnit' num 11points
## ..- attr(*, "unit")= int 8
## $ legend.spacing.x : NULL
## $ legend.spacing.y : NULL
## $ legend.key : list()
## ..- attr(*, "class")= chr [1:2] "element_blank" "element"
## $ legend.key.size : 'simpleUnit' num 1.2lines
## ..- attr(*, "unit")= int 3
## $ legend.key.height : NULL
## $ legend.key.width : NULL
## $ legend.key.spacing : 'simpleUnit' num 5.5points
## ..- attr(*, "unit")= int 8

```

```

## $ legend.key.spacing.x : NULL
## $ legend.key.spacing.y : NULL
## $ legend.frame : NULL
## $ legend.ticks : NULL
## $ legend.ticks.length : 'rel' num 0.2
## $ legend.axis.line : NULL
## $ legend.text :List of 11
##   ..$ family : NULL
##   ..$ face : NULL
##   ..$ colour : NULL
##   ..$ size : 'rel' num 0.8
##   ..$ hjust : NULL
##   ..$ vjust : NULL
##   ..$ angle : NULL
##   ..$ lineheight : NULL
##   ..$ margin : NULL
##   ..$ debug : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ legend.text.position : NULL
## $ legend.title :List of 11
##   ..$ family : NULL
##   ..$ face : NULL
##   ..$ colour : NULL
##   ..$ size : NULL
##   ..$ hjust : num 0
##   ..$ vjust : NULL
##   ..$ angle : NULL
##   ..$ lineheight : NULL
##   ..$ margin : NULL
##   ..$ debug : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
## $ legend.title.position : NULL
## $ legend.position : chr "bottom"
## $ legend.position.inside : NULL
## $ legend.direction : NULL
## $ legend.byrow : NULL
## $ legend.justification : chr "center"
## $ legend.justification.top : NULL
## $ legend.justification.bottom : NULL
## $ legend.justification.left : NULL
## $ legend.justification.right : NULL
## $ legend.justification.inside : NULL
## $ legend.location : NULL
## $ legend.box : NULL
## $ legend.box.just : NULL
## $ legend.box.margin : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
## $ legend.box.background : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
## $ legend.box.spacing : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
## [list output truncated]

```

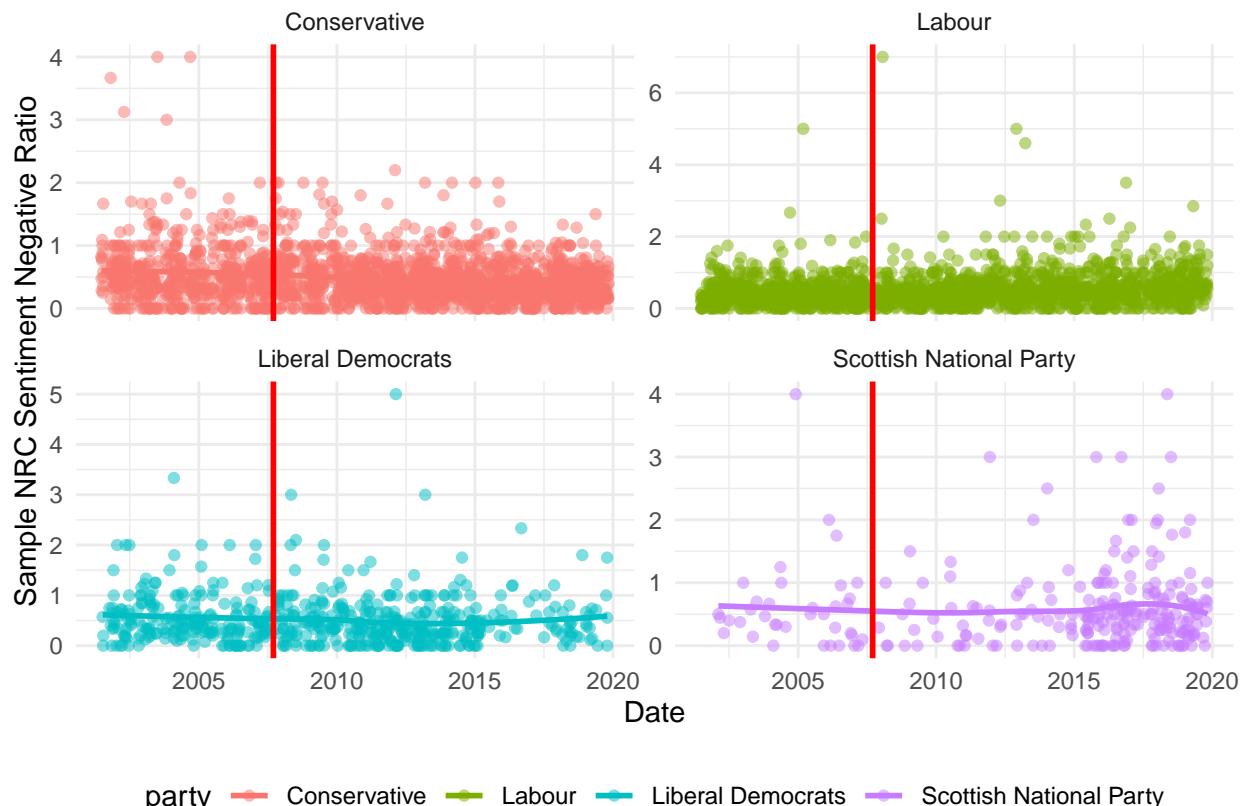
```
## - attr(*, "class")= chr [1:2] "theme" "gg"
## - attr(*, "complete")= logi TRUE
## - attr(*, "validate")= logi TRUE
```

Negative sentiments of sample texts also show steady lines.

Dividing the sample sentiment plot by party

```
# Faceting plots by party
filtered_samp_nrc_party %>%
  ggplot(aes(date, ratio, color = party)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "loess", se = F, alpha = 0.7, size = 1) +
  labs(y = "Sample NRC Sentiment Negative Ratio", x = "Date") +
  facet_wrap(~party, scales = "free_y") +
  geom_vline(xintercept = as.numeric(as.Date("2007-09-14")), col = "red", size = 1) +
  theme_minimal() +
  theme(legend.position = "bottom")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Compared to the immigration related text sentiment plots, sample sentiment plots show no significant difference.

3. Word Embedding GloVe

```
# Set up
library(text2vec)
library(stringr)
library(umap)
library(ggrepel)
```

1) Training total immigration related texts

(1) The GloVe model for total parliament speech

Training process

```
# choice parameters
WINDOW_SIZE <- 6 # context up to 6 words
DIM <- 300 # the length of the word vector
ITERS <- 100 # the maximum number of iterations
COUNT_MIN <- 10 # minimum count of words

# shuffle text
set.seed(42L) # for reproducibility
glove_text <- sample(tidy_data_notoken$desc)

# create vocab
tokens <- space_tokenizer(glove_text) # tokenizing
it <- itoken(tokens, progressbar = FALSE) # create the vocabulary object
vocab <- create_vocabulary(it) # create a vocabulary
vocab_pruned <- prune_vocabulary(vocab, term_count_min = COUNT_MIN) # keep only words that meet count

#vectorize and create term co-occurrence matrix
vectorizer <- vocab_vectorizer(vocab_pruned)
tcm <- create_tcm(it, vectorizer, skip_grams_window = WINDOW_SIZE, skip_grams_window_context = "symmetric",
                    weights = rep(1, WINDOW_SIZE))
```

Setting model parameters

```
#set model parameters
glove <- GlobalVectors$new(rank = DIM, x_max = 100, learning_rate = 0.05)

# fit model
word_vectors_main <- glove$fit_transform(tcm, n_iter = ITERS, convergence_tol = 0.001, n_threads = RcppThreads::RThreads())

# get output
word_vectors_context <- glove$components
glove_embedding <- word_vectors_main + t(word_vectors_context) # word vectors. combine main and context

#save
saveRDS(glove_embedding, file = "local_glove.rds")
```

Modelling takes time. So I will use the prepared model made by same process for knitting.

```
url <- "https://github.com/RiverKim-garam/CTA24-Final-assessment/blob/main/local_glove.rds?raw=true"
glove_embedding <- readRDS(url, method = "libcurl")
```

Visualization of GloVe word embedding model by using umap.

```
# GloVe dimension reduction (two dimension)
glove_umap <- umap(glove_embedding, n_components = 2, metric = "cosine", n_neighbors = 25, min_dist = 0)

# Put results in a dataframe for ggplot
df_glove_umap <- as.data.frame(glove_umap[["layout"]])

# Add the labels of the words to the dataframe
df_glove_umap$word <- rownames(df_glove_umap)
colnames(df_glove_umap) <- c("UMAP1", "UMAP2", "word")
```

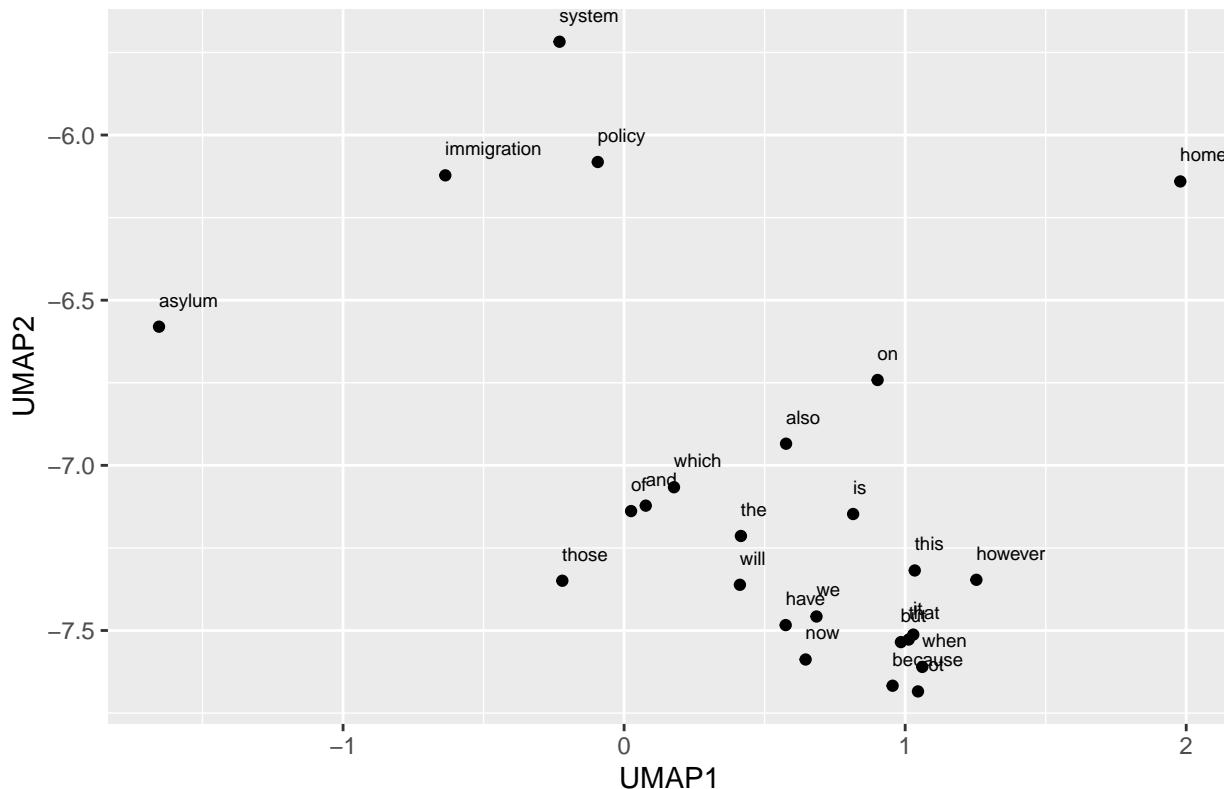
Plot the total word embedding of ‘immigration’ with the GloVe model generated above.

```
word <- glove_embedding[["immigration"],, drop = FALSE]
cos_sim = sim2(x = glove_embedding, y = word, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
selected_words <- df_glove_umap %>%
  inner_join(y=select, by= "word")
```

The ggplot visual for GloVe regarding total immigration related texts. Total parliament speech word embedding of words related to *immigration*

```
ggplot(selected_words, aes(x = UMAP1, y = UMAP2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(UMAP1, UMAP2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0) +
  labs(title = "GloVe word embedding of words related to 'immigration'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

GloVe word embedding of words related to 'immigration'



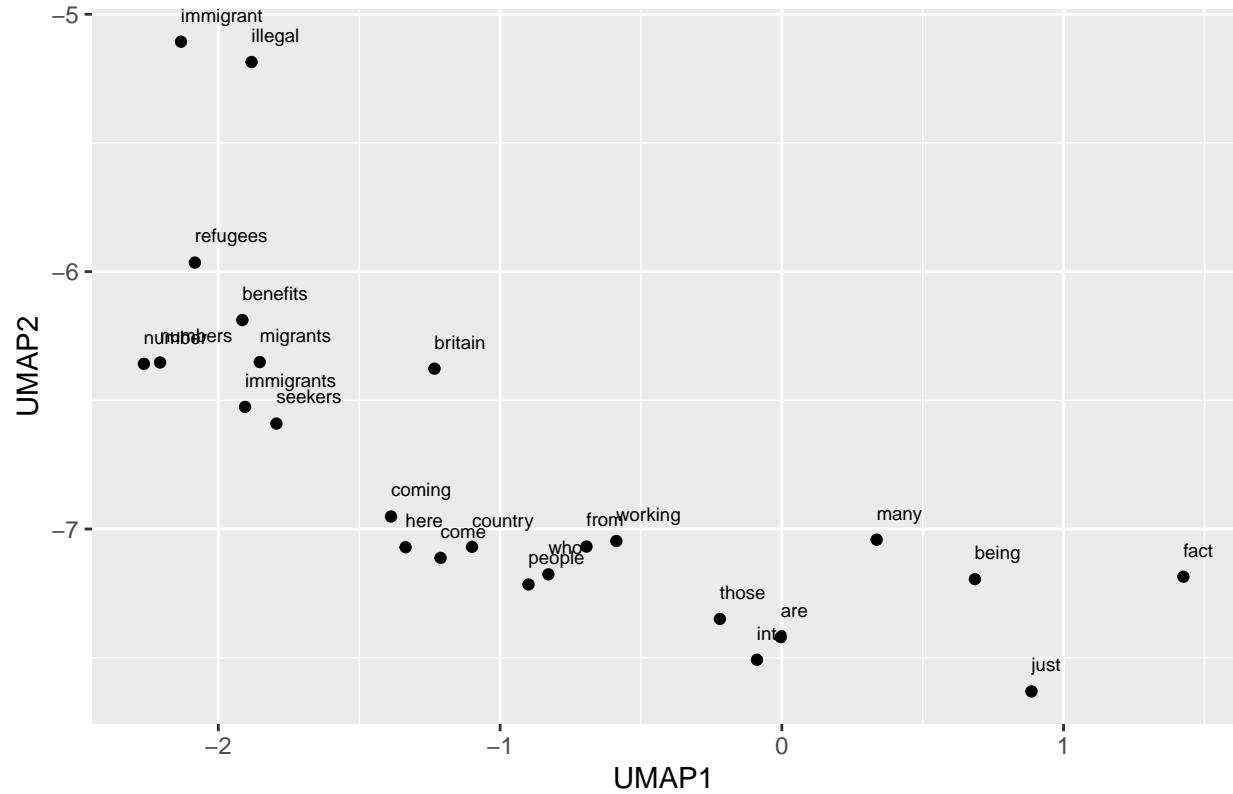
I did it for confirming the modeling process, but we might need to use this map as a comparison plot.

Total parliament speech word embedding of words related to *immigrants*

```
# Plot the word embedding of words that are related for the GloVe model (Case2: immigrants)
word_2 <- glove_embedding[["immigrants"], , drop = FALSE]
cos_sim = sim2(x = glove_embedding, y = word_2, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
selected_words_2 <- df_glove_umap %>%
  inner_join(y=select, by= "word")
```

```
#The ggplot visual for GloVe
ggplot(selected_words_2, aes(x = UMAP1, y = UMAP2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(UMAP1, UMAP2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0) +
  labs(title = "Total parliament immigration related speech word embedding of 'immigrants'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

Total parliament immigration related speech word embedding of 'immigrant'



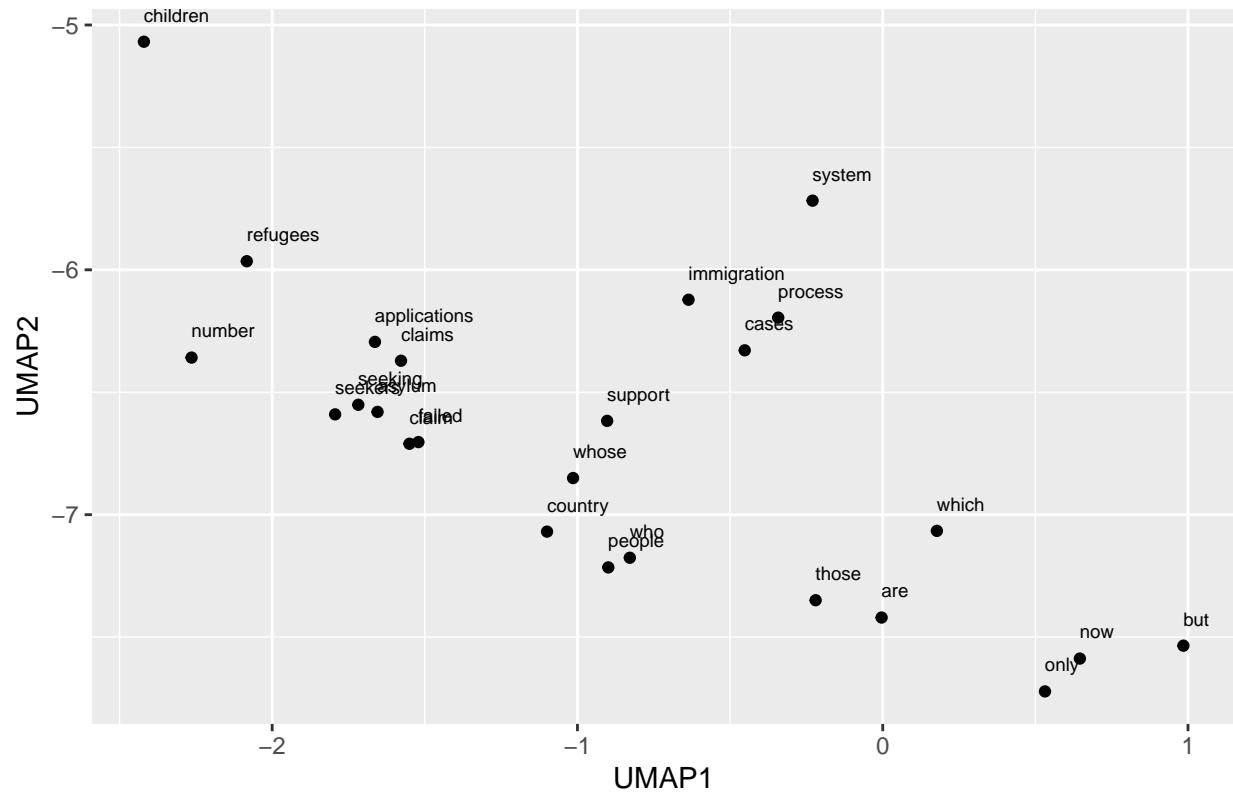
This can be also used for comparison. Also, word embedding maps tell us the general relationships between words, especially those we are interested in.

For the last pilot modelling map, there is a total parliament immigration related speech word embedding of words related to *asylum*

```
# Plot the word embedding of words that are related for the GloVe model (asylum)
word_4 <- glove_embedding["asylum"], drop = FALSE)
cos_sim = sim2(x = glove_embedding, y = word_4, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_4 <- df_glove_umap %>%
  inner_join(y=select, by= "word")

#The ggplot visual for Glove
ggplot(selected_words_4, aes(x = UMAP1, y = UMAP2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(UMAP1, UMAP2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0) +
  labs(title = "Total parliament immigration related speech word embedding of 'asylum'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

Total parliament immigration related speech word embedding of 'asylum'



(2) The Glove model for total Parliament speech before the 2008 economic crisis

```
# filter data by date. Now have pre/post economic crisis text data related to immigration.

data_pre <- filter(tidy_data_notoken, date < as.Date("2007-09-24"))
data_post <- filter(tidy_data_notoken, date >= as.Date("2007-09-24"))

# repeat the same modeling process
set.seed(42L)
glove_text_pre <- sample(data_pre$desc)

tokens_pre <- space_tokenizer(glove_text_pre)
it_pre <- itoken(tokens_pre, progressbar = FALSE)
vocab_pre <- create_vocabulary(it_pre)
vocab_pruned_pre <- prune_vocabulary(vocab_pre, term_count_min = COUNT_MIN)

vectorizer_pre <- vocab_vectorizer(vocab_pruned_pre)
tcm_pre <- create_tcm(it_pre, vectorizer_pre, skip_grams_window = WINDOW_SIZE, skip_grams_window_context = CONTEXT_SIZE)

glove_pre <- GlobalVectors$new(rank = DIM, x_max = 100, learning_rate = 0.05)
word_vectors_main_pre <- glove_pre$fit_transform(tcm_pre, n_iter = ITERS, convergence_tol = 0.001, n_threads = 4)

word_vectors_context_pre <- glove_pre$components
glove_embedding_pre <- word_vectors_main_pre + t(word_vectors_context_pre)
```

```
saveRDS(glove_embedding_pre, file = "local_glove_pre.rds")
```

Modelling takes time. So I will use the prepared model made by same process for knitting.

```
url_pre <- "https://github.com/RiverKim-garam/CTA24-Final-assessment/blob/main/local_glove_pre.rds?raw=true"
glove_embedding_pre <- readRDS(url(url_pre, method = "libcurl"))
```

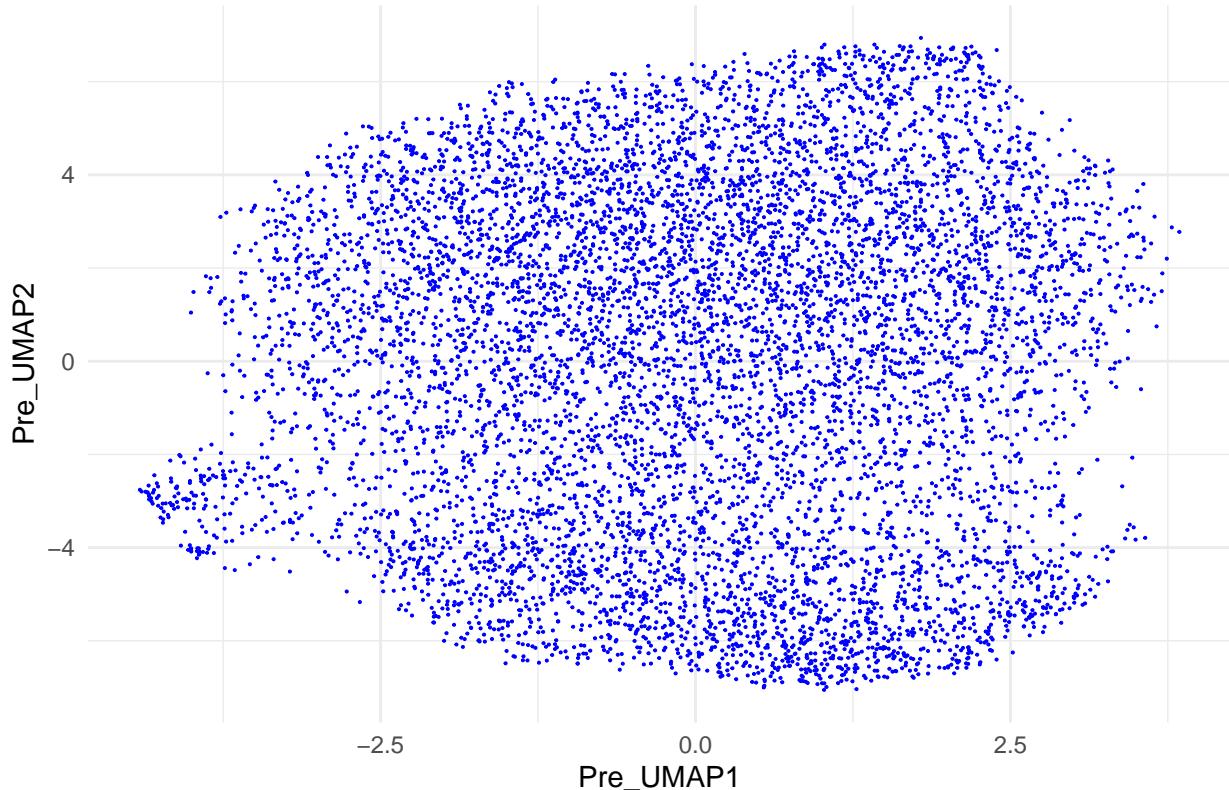
Pre-crisis GloVe word embedding model of total parliament texts by two dimensional umap

```
# Plotting the whole word embedding of pre-crisis immigration related text
umap_pre <- umap(glove_embedding_pre, n_components = 2, metric = "cosine", n_neighbors = 25, min_dist = 0.1)
df_umap_pre <- as.data.frame(umap_pre[["layout"]])
df_umap_pre$word <- rownames(df_umap_pre)
colnames(df_umap_pre) <- c("Pre_UMAP1", "Pre_UMAP2", "word")
```

Let's see the whole word embedding map of two dimensions

```
ggplot(df_umap_pre) +
  geom_point(aes(x = Pre_UMAP1, y = Pre_UMAP2), color = 'blue', size = 0.05) +
  labs(title = "(Pre-crisis) GloVe word embedding of words related to 'immigration'", subtitle = "") +
  theme_minimal()
```

(Pre-crisis) GloVe word embedding of words related to 'immigration'

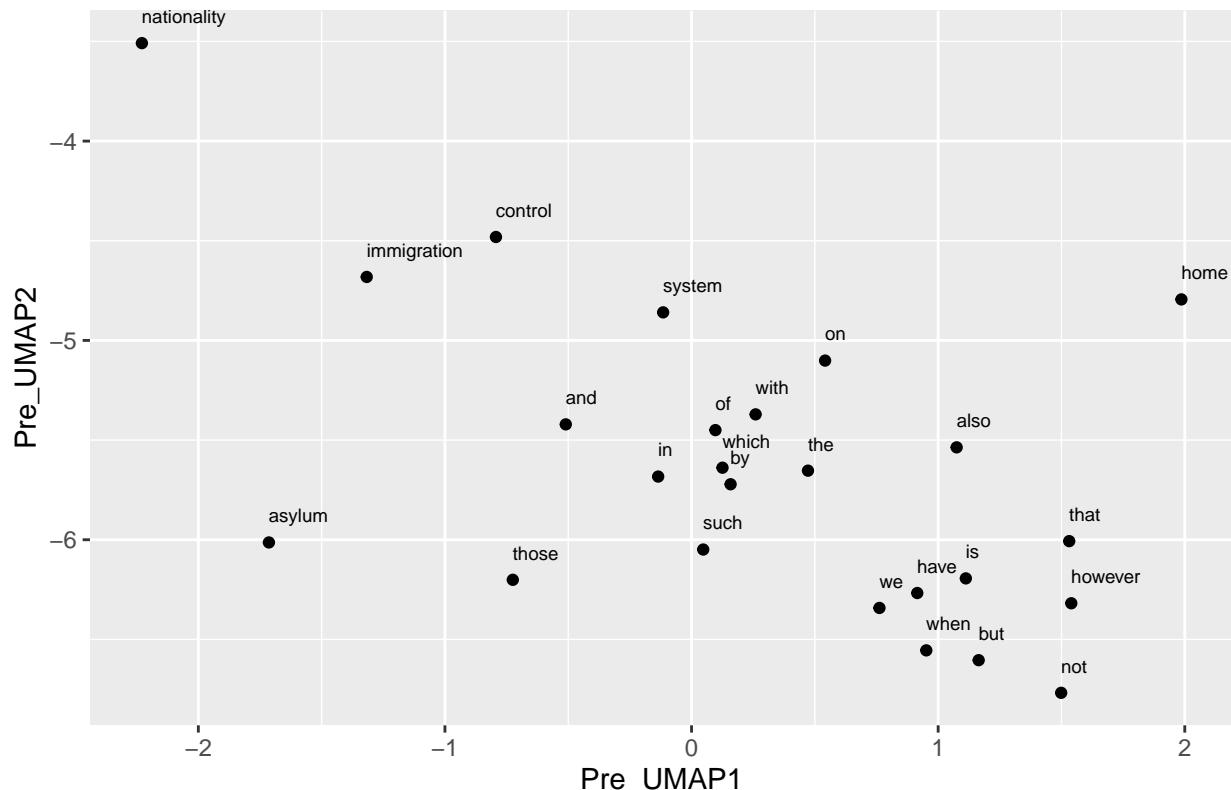


Specify the map. Which words are close to the word 'immigration' in the total parliament texts? Pre-crisis total parliament speech word embedding of words related to *immigration*

```
# Plot the word embedding of words that are related for the GloVe model (Case1: immigration)
word_pre_1 <- glove_embedding_pre[["immigration"], , drop = FALSE]
cos_sim = sim2(x = glove_embedding_pre, y = word_pre_1, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
selected_words_pre_1 <- df_umap_pre %>%
  inner_join(v=select, by= "word")
```

```
#The ggplot visual for GloVe
ggplot(selected_words_pre_1, aes(x = Pre_UMAP1, y = Pre_UMAP2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Pre_UMAP1, Pre_UMAP2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0)
  labs(title = "(Pre-Crisis) total parliament speech word embedding of 'immigration'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

(Pre-Crisis) total parliament speech word embedding of 'immigration'

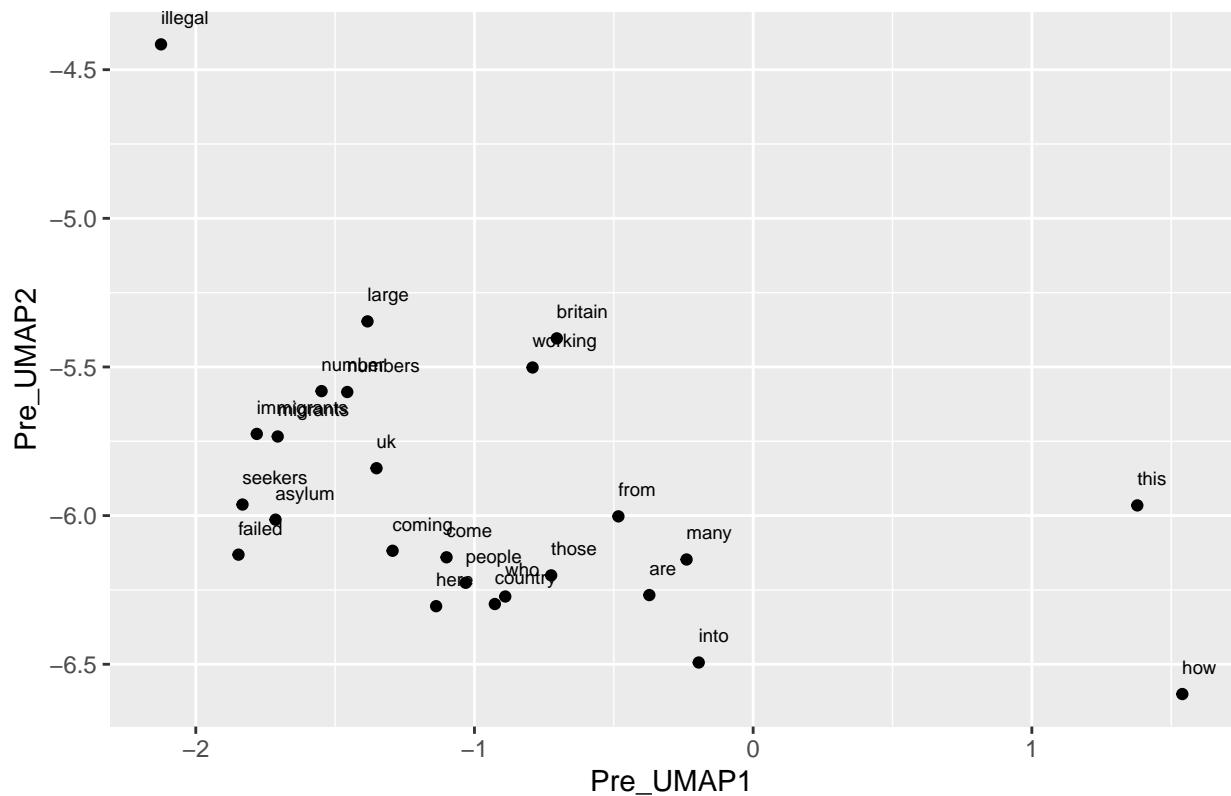


Pre-crisis total parliament speech word embedding of words related to *immigrants*

```
# Plot the word embedding of words that are related for the GloVe model (Case2: immigrants)
word_pre_2 <- glove_embedding_pre[["immigrants"], , drop = FALSE]
cos_sim = sim2(x = glove_embedding_pre, y = word_pre_2, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
selected_words_pre_2 <- df_umap_pre %>%
  inner_join(y=select, by= "word")
```

```
#The ggplot visual for GloVe
ggplot(selected_words_pre_2, aes(x = Pre_UMAP1, y = Pre_UMAP2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Pre_UMAP1, Pre_UMAP2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0) +
  labs(title = "(Pre-Crisis) total parliament speech word embedding of 'immigrants'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

(Pre-Crisis) total parliament speech word embedding of 'immigrants'

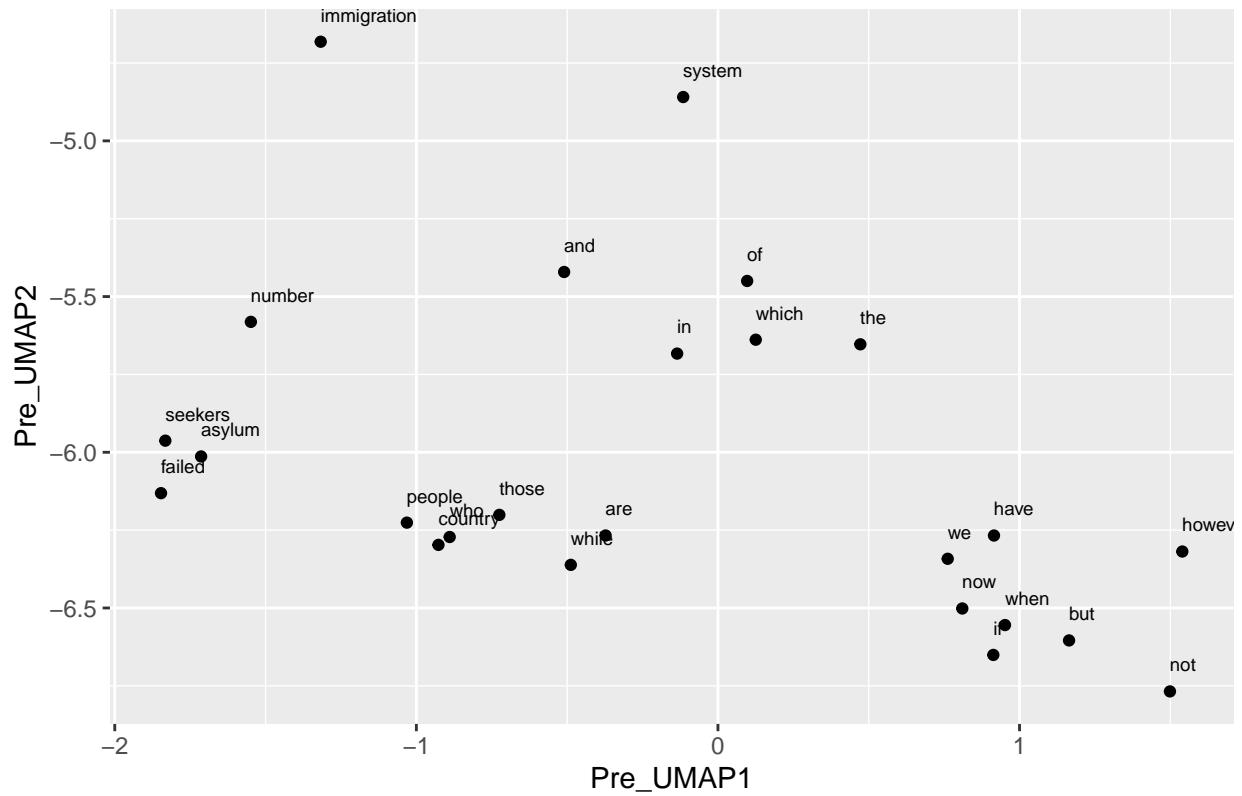


Pre-crisis total parliament speech word embedding of words related to *asylum*

```
# Plot the word embedding of words that are related for the GloVe model (asylum)
word_pre_3 <- glove_embedding_pre["asylum",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_pre, y = word_pre_3, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_pre_3 <- df_umap_pre %>%
  inner_join(y=select, by= "word")
```

```
#The ggplot visual for GloVe
ggplot(selected_words_pre_3, aes(x = Pre_UMAP1, y = Pre_UMAP2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Pre_UMAP1, Pre_UMAP2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0) +
  labs(title = "(Pre-Crisis) total parliament speech word embedding of 'asylum'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

(Pre-Crisis) total parliament speech word embedding of 'asylum'



Now we can check the words relationships before the 2008 financial crisis.

(3) The Glove model for total Parliament speech after the 2008 financial crisis

Again, same modeling process

```
set.seed(42L)
glove_text_post <- sample(data_post$desc)

tokens_post <- space_tokenizer(glove_text_post)
it_post <- itoken(tokens_post, progressbar = FALSE)
vocab_post <- create_vocabulary(it_post)
vocab_pruned_post <- prune_vocabulary(vocab_post, term_count_min = COUNT_MIN)

vectorizer_post <- vocab_vectorizer(vocab_pruned_post)
tcm_post <- create_tcm(it_post, vectorizer_post, skip_grams_window = WINDOW_SIZE, skip_grams_window_con

glove_post <- GlobalVectors$new(rank = DIM, x_max = 100, learning_rate = 0.05)
word_vectors_main_post <- glove_post$fit_transform(tcm_post, n_iter = ITERS, convergence_tol = 0.001, n

word_vectors_context_post <- glove_post$components
glove_embedding_post <- word_vectors_main_post + t(word_vectors_context_post)

saveRDS(glove_embedding_post, file = "local_glove_post.rds")
```

Modelling takes time. So I will use the prepared model made by same process for knitting.

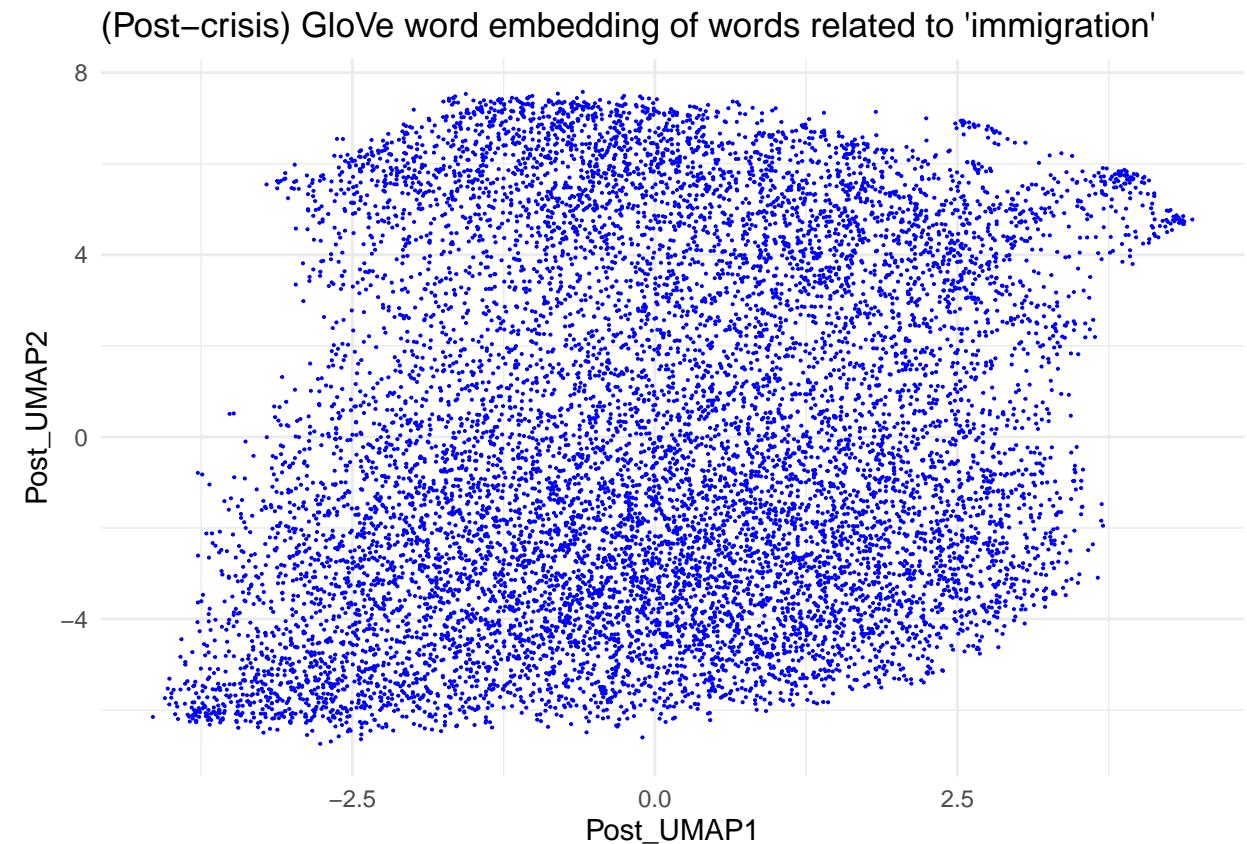
```
url_post <- "https://github.com/RiverKim-garam/CTA24-Final-assessment/blob/main/local_glove_post.rds?raw=true"
glove_embedding_post <- readRDS(url(url_post, method = "libcurl"))
```

Pre-crisis total GloVe word embedding two dimensional umap

```
# Plotting the whole word embedding of pre-crisis immigration related text
umap_post <- umap(glove_embedding_post, n_components = 2, metric = "cosine", n_neighbors = 25, min_dist = 0.1)

df_umap_post <- as.data.frame(umap_post[["layout"]])
df_umap_post$word <- rownames(df_umap_post)
colnames(df_umap_post) <- c("Post_UMAP1", "Post_UMAP2", "word")

ggplot(df_umap_post) +
  geom_point(aes(x = Post_UMAP1, y = Post_UMAP2), color = 'blue', size = 0.05) +
  labs(title = "(Post-crisis) GloVe word embedding of words related to 'immigration'"") +
  theme_minimal()
```

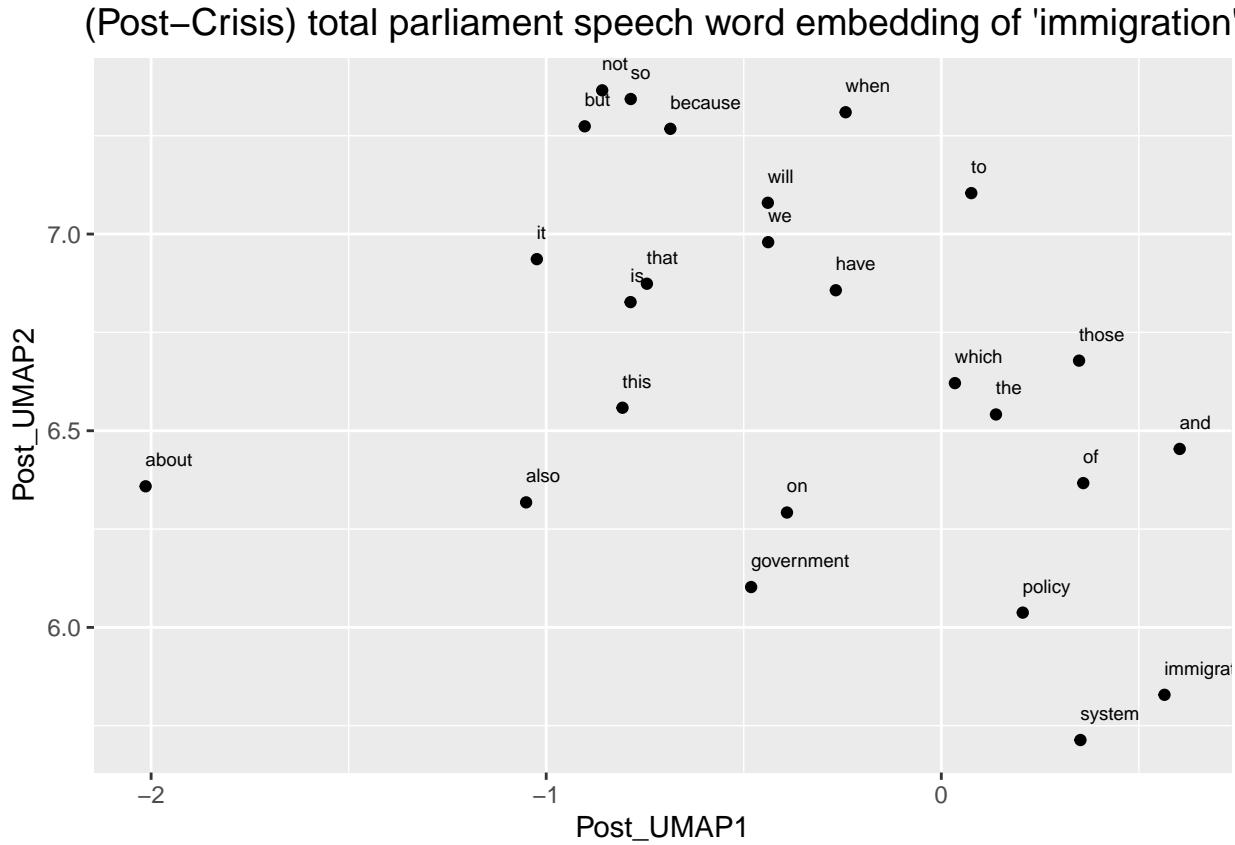


Pre-crisis total parliament speech word embedding of words related to *immigration*

```
# Plot the word embedding of words that are related for the GloVe model (Case1: immigration)
word_post_1 <- glove_embedding_post["immigration", , drop = FALSE]
cos_sim = sim2(x = glove_embedding_post, y = word_post_1, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
```

```
selected_words_post_1 <- df_umap_post %>%
  inner_join(y=select, by= "word")

#The ggplot visual for GloVe
ggplot(selected_words_post_1, aes(x = Post_UMAP1, y = Post_UMAP2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Post_UMAP1, Post_UMAP2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0) +
  labs(title = "(Post-Crisis) total parliament speech word embedding of 'immigration'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

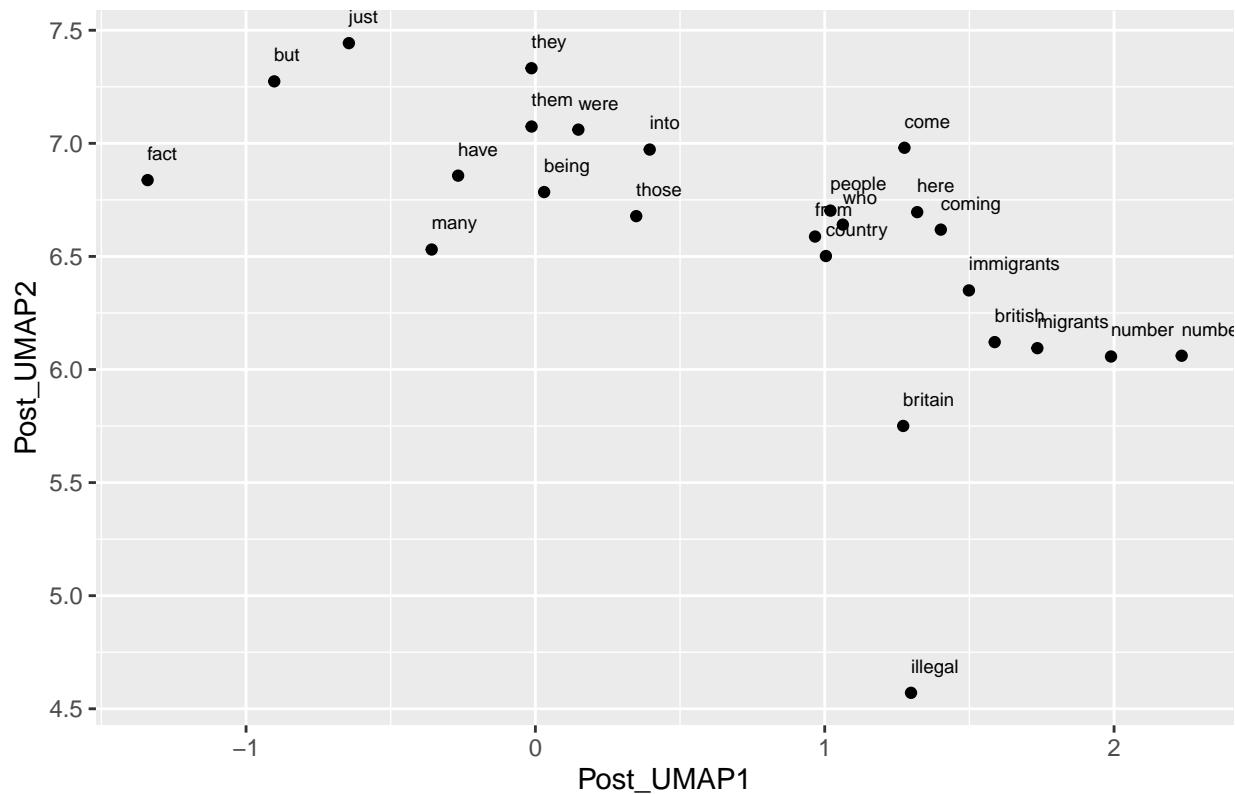


Pre-crisis total parliament speech word embedding of words related to *immigrants*

```
# Plot the word embedding of words that are related for the GloVe model (Case2: immigrants)
word_post_2 <- glove_embedding_post[["immigrants"], drop = FALSE]
cos_sim = sim2(x = glove_embedding_post, y = word_post_2, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_post_2 <- df_umap_post %>%
  inner_join(y=select, by= "word")
```

```
#The ggplot visual for GloVe
ggplot(selected_words_post_2, aes(x = Post_UMAP1, y = Post_UMAP2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Post_UMAP1, Post_UMAP2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0.5) +
  labs(title = "(Post-Crisis) total parliament speech word embedding of 'immigrants'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

(Post–Crisis) total parliament speech word embedding of 'immigrants'

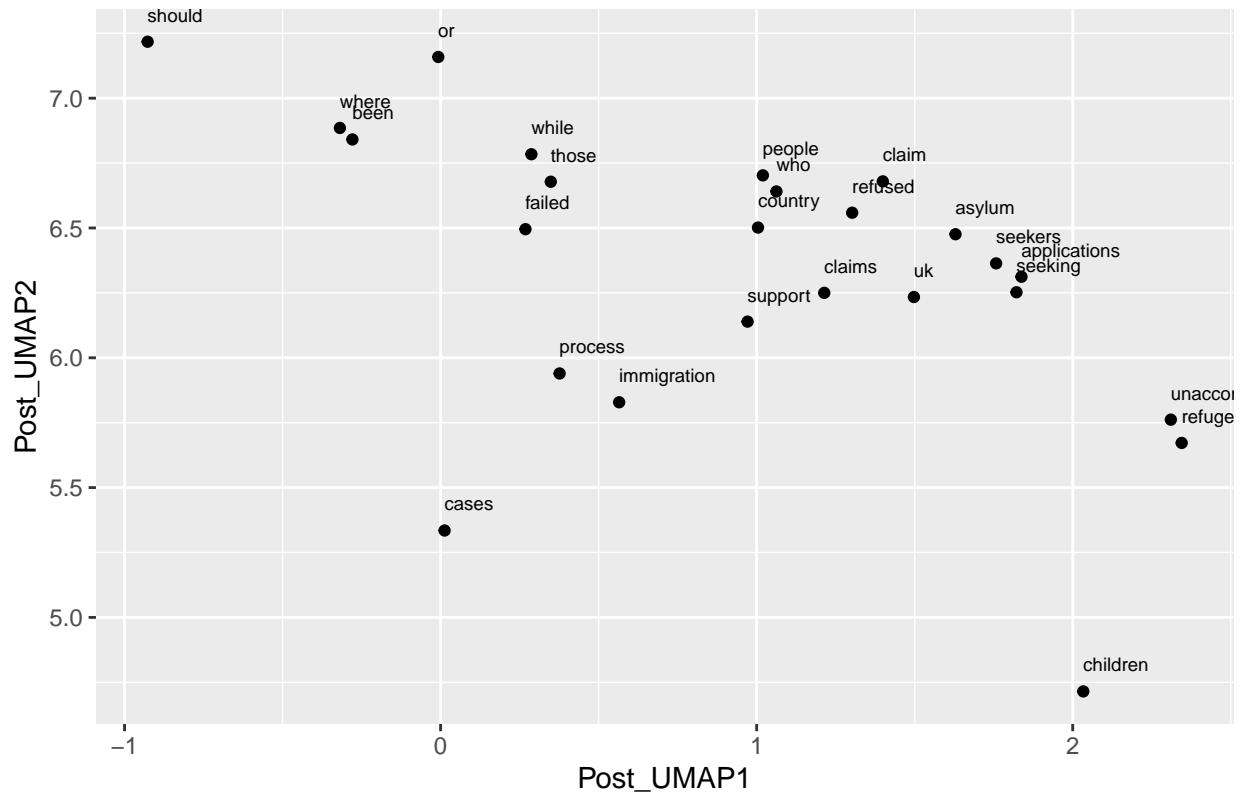


Post-crisis total parliament speech word embedding of words related to *asylum*

```
# Plot the word embedding of words that are related for the GloVe model (asylum)
word_post_4 <- glove_embedding_post["asylum",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_post, y = word_post_4, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_post_4 <- df_umap_post %>%
  inner_join(y=select, by= "word")

#The ggplot visual for GloVe
ggplot(selected_words_post_4, aes(x = Post_UMAP1, y = Post_UMAP2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Post_UMAP1, Post_UMAP2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=.5)
```

(Post–Crisis) total parliament speech word embedding of 'asylum'



We can now compare the two periods with three different embedding maps.

2) Training by party and pre/post economic crisis

To make four different GloVe embedding models, dividing data by party and date

```
library(RcppParallel)

# filter data by party and date. Now have pre/post economic crisis text data of Conservative and Labour
data_conser_pre <- filter(tidy_data_notoken, party == "Conservative" & date < as.Date("2007-09-24"))
data_labour_pre <- filter(tidy_data_notoken, party == "Labour" & date < as.Date("2007-09-24"))

data_conser_post <- filter(tidy_data_notoken, party == "Conservative" & date >= as.Date("2007-09-24"))
data_labour_post <- filter(tidy_data_notoken, party == "Labour" & date >= as.Date("2007-09-24"))
```

(1) The GloVe model for Conservative party before the 2008 economic crisis

Training process

```
set.seed(42L)
glove_text_conser_pre <- sample(data_conser_pre$desc)

tokens_conser_pre <- space_tokenizer(glove_text_conser_pre)
it_conser_pre <- itoken(tokens_conser_pre, progressbar = FALSE)
```

```

vocab_conser_pre <- create_vocabulary(it_conser_pre)
vocab_pruned_conser_pre <- prune_vocabulary(vocab_conser_pre, term_count_min = COUNT_MIN)

vectorizer_conser_pre <- vocab_vectorizer(vocab_pruned_conser_pre)
tcm_conser_pre <- create_tcm(it_conser_pre, vectorizer_conser_pre, skip_grams_window = WINDOW_SIZE, skip_grams_min = COUNT_MIN)

glove_conser_pre <- GlobalVectors$new(rank = DIM, x_max = 100, learning_rate = 0.05)
word_vectors_main_conser_pre <- glove_conser_pre$fit_transform(tcm_conser_pre, n_iter = ITERS, convergence_threshold = 0.001)

word_vectors_context_conser_pre <- glove_conser_pre$components
glove_embedding_conser_pre <- word_vectors_main_conser_pre + t(word_vectors_context_conser_pre)

saveRDS(glove_embedding_conser_pre, file = "local_glove_conser_pre.rds")

```

Modelling takes time. So I will use the prepared model made by same process for knitting.

```

url_conser_pre <- "https://github.com/RiverKim-garam/CTA24-Final-assessment/blob/main/local_glove_conser_pre.rds"
glove_embedding_conser_pre <- readRDS(url(url_conser_pre, method = "libcurl"))

```

Total GloVe word embedding two dimensional umap

```

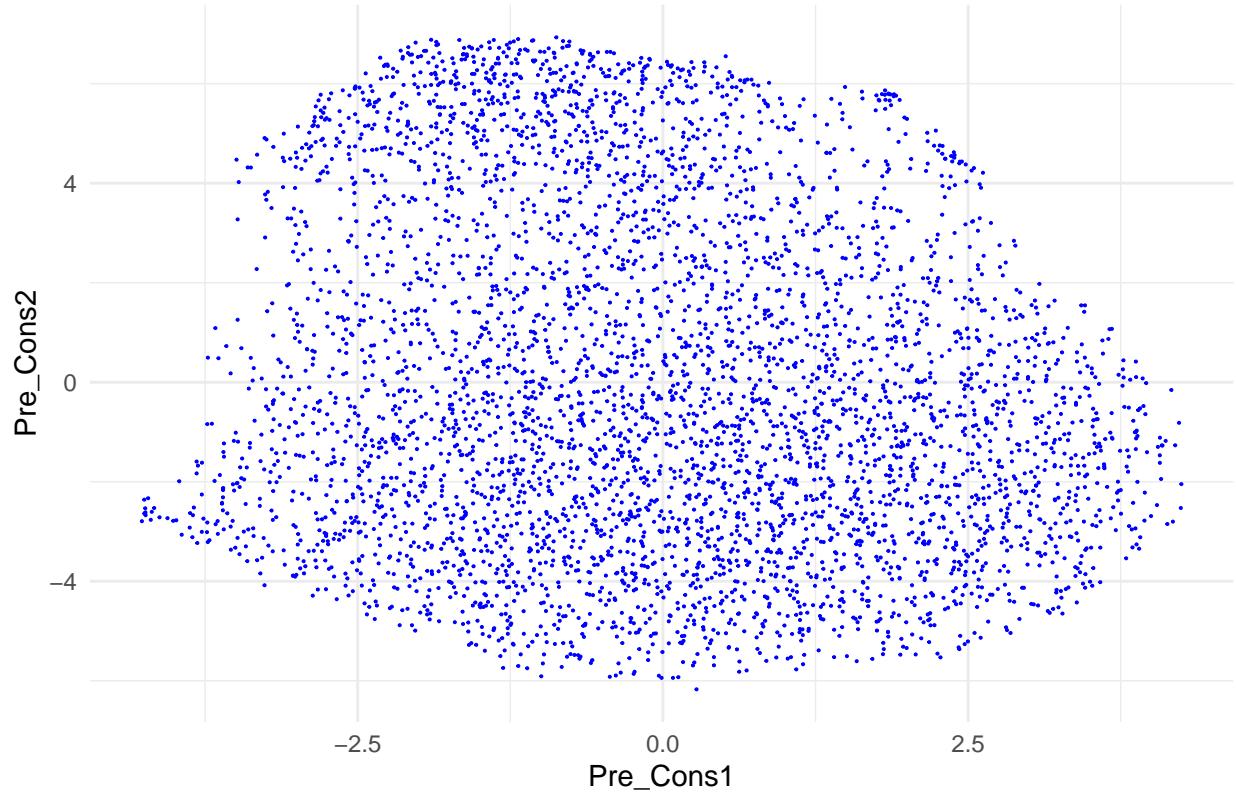
# Plotting the whole word embeddings of pre-crisis Conservative party immigration related text
umap_conser_pre <- umap(glove_embedding_conser_pre, n_components = 2, metric = "cosine", n_neighbors = 10, min_dist = 0.1, random_state = 123456789)

df_umap_conser_pre <- as.data.frame(umap_conser_pre[["layout"]])
df_umap_conser_pre$word <- rownames(df_umap_conser_pre)
colnames(df_umap_conser_pre) <- c("Pre_Cons1", "Pre_Cons2", "word")

ggplot(df_umap_conser_pre) +
  geom_point(aes(x = Pre_Cons1, y = Pre_Cons2), color = 'blue', size = 0.05) +
  labs(title = "Conservative Party (Pre-Crisis): Word Embeddings of GloVe and UMAP") +
  theme_minimal()

```

Conservative Party (Pre–Crisis): Word Embeddings of GloVe and UMAP

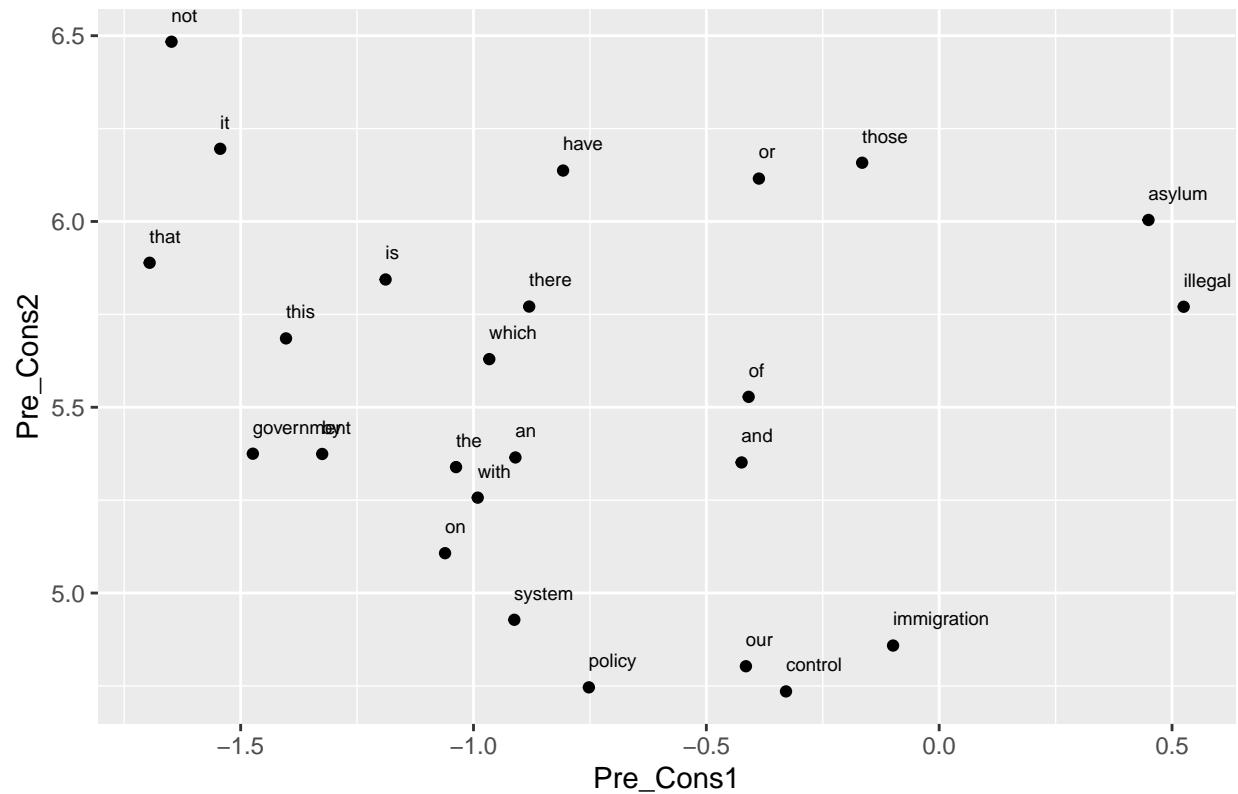


Pre-crisis Conservative party word embedding of words related to *immigration*

```
# Plot the word embedding of words that are related for the GloVe model (Case1: immigration)
word_conser_pre_1 <- glove_embedding_conser_pre["immigration",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_conser_pre, y = word_conser_pre_1, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_conser_pre_1 <- df_umap_conser_pre %>%
  inner_join(y=select, by= "word")

#The ggplot visual for GloVe
ggplot(selected_words_conser_pre_1, aes(x = Pre_Cons1, y = Pre_Cons2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Pre_Cons1, Pre_Cons2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0)
  labs(title = "Conservative party (Pre-Crisis) word embedding of 'immigration'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

Conservative party (Pre–Crisis) word embedding of 'immigration'

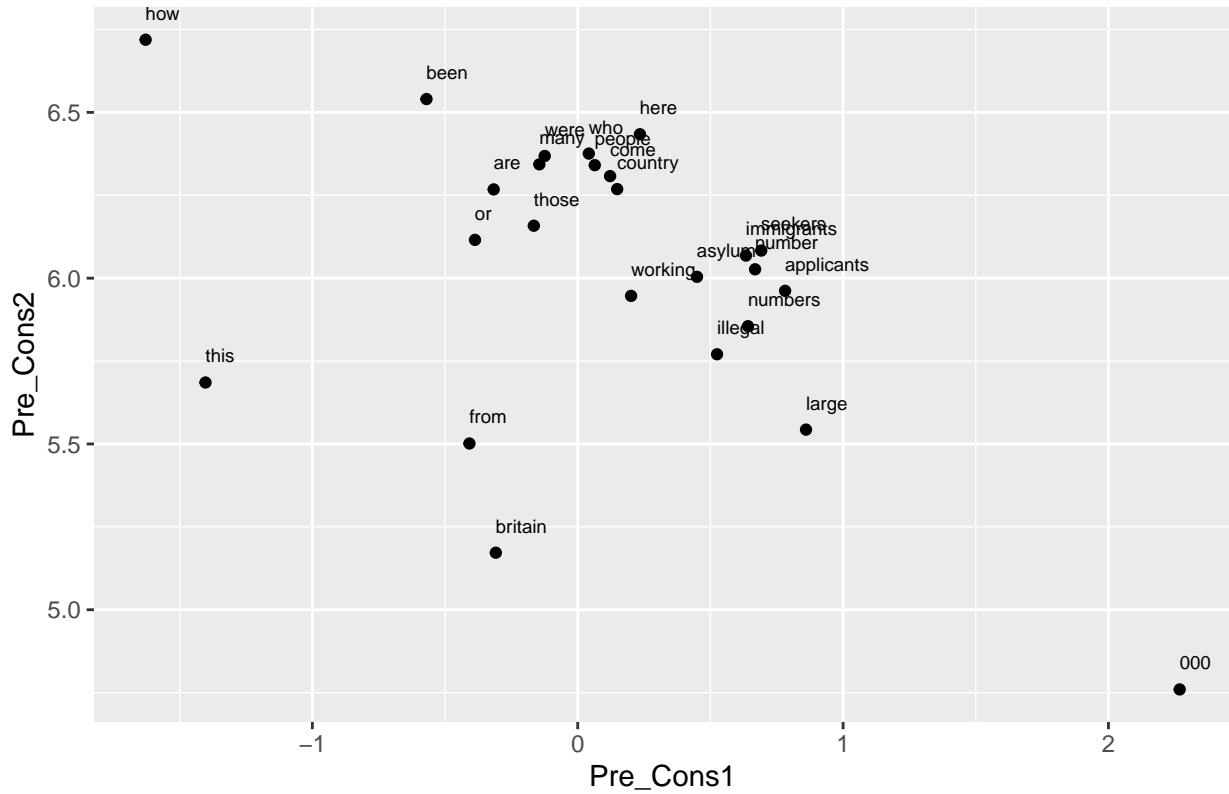


Pre-crisis Conservative party word embedding of words related to *immigrants*

```
# Plot the word embedding of words that are related for the GloVe model (Case2: immigrants)
word_conser_pre_2 <- glove_embedding_conser_pre["immigrants",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_conser_pre, y = word_conser_pre_2, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_conser_pre_2 <- df_umap_conser_pre %>%
  inner_join(y=select, by= "word")

#The ggplot visual for GloVe
ggplot(selected_words_conser_pre_2, aes(x = Pre_Cons1, y = Pre_Cons2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Pre_Cons1, Pre_Cons2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0)
  labs(title = "Conservative party (Pre–Crisis) word embedding of 'immigrants'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

Conservative party (Pre-Crisis) word embedding of 'immigrants'

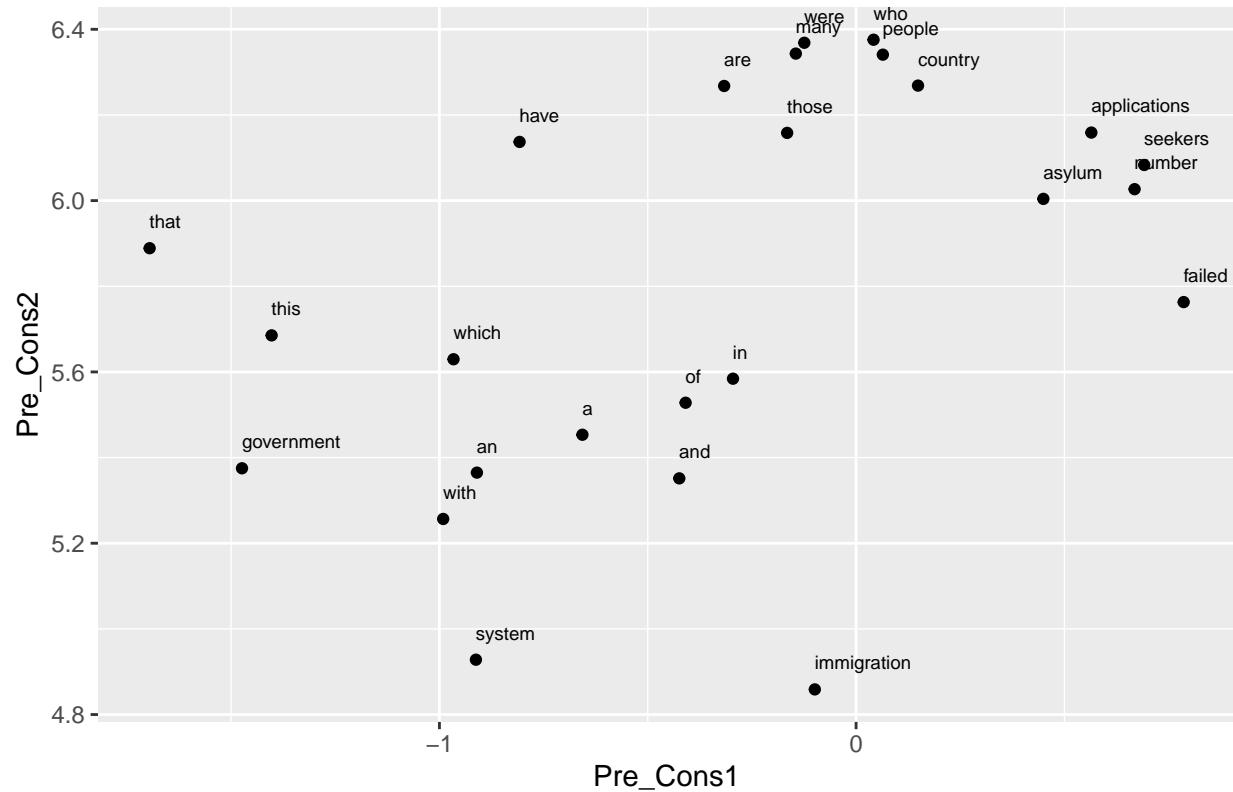


Pre-crisis Conservative party word embedding of words related to *asylum*

```
# Plot the word embedding of words that are related for the GloVe model (case3: asylum)
word_conser_pre_3 <- glove_embedding_conser_pre["asylum",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_conser_pre, y = word_conser_pre_3, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_conser_pre_3 <- df_umap_conser_pre %>%
  inner_join(y=select, by= "word")
```

```
#The ggplot visual for GloVe
ggplot(selected_words_conser_pre_3, aes(x = Pre_Cons1, y = Pre_Cons2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Pre_Cons1, Pre_Cons2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0)
  labs(title = "Conservative party (Pre-Crisis) word embedding of 'asylum'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

Conservative party (Pre–Crisis) word embedding of 'asylum'



(2) The GloVe model for Conservative party after the 2008 economic crisis

Training process

```
set.seed(42L)
glove_text_conser_post <- sample(data_conser_post$desc)

tokens_conser_post <- space_tokenizer(glove_text_conser_post)
it_conser_post <- itoken(tokens_conser_post, progressbar = FALSE)
vocab_conser_post <- create_vocabulary(it_conser_post)
vocab_pruned_conser_post <- prune_vocabulary(vocab_conser_post, term_count_min = COUNT_MIN)

vectorizer_conser_post <- vocab_vectorizer(vocab_pruned_conser_post)
tcm_conser_post <- create_tcm(it_conser_post, vectorizer_conser_post, skip_grams_window = WINDOW_SIZE, n_gram_size = 2)

glove_conser_post <- GlobalVectors$new(rank = DIM, x_max = 100, learning_rate = 0.05)
word_vectors_main_conser_post <- glove_conser_post$fit_transform(tcm_conser_post, n_iter = ITERS, convergence_threshold = 0.001)

word_vectors_context_conser_post <- glove_conser_post$components
glove_embedding_conser_post <- word_vectors_main_conser_post + t(word_vectors_context_conser_post)

saveRDS(glove_embedding_conser_post, file = "local_glove_conser_post.rds")
```

Modelling takes time. So I will use the prepared model made by same process for knitting.

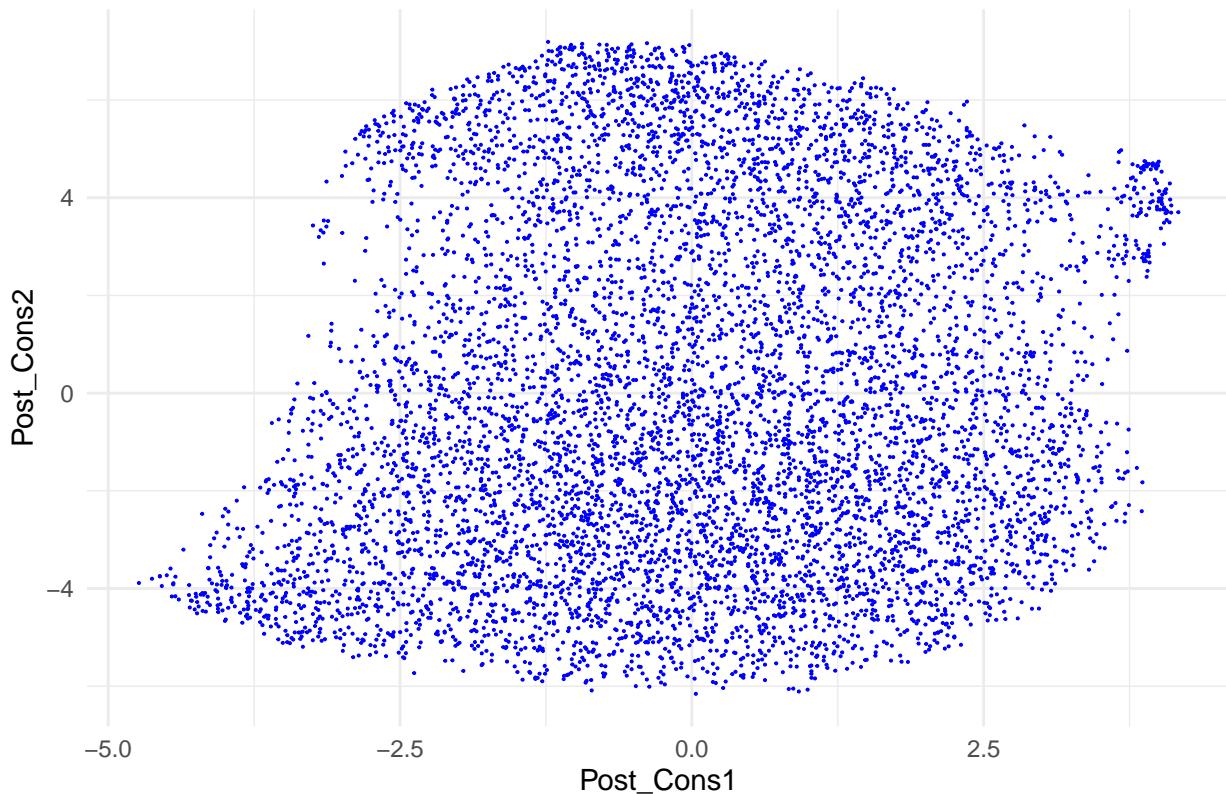
```
url_conser_post <- "https://github.com/RiverKim-garam/CTA24-Final-assessment/blob/main/local_glove_cons
glove_embedding_conser_post <- readRDS(url(url_conser_post, method = "libcurl"))
```

Total GloVe word embedding two dimensional umap

```
# Plotting the whole word embeddings of post-crisis Conservative party immigration related text
umap_conser_post <- umap(glove_embedding_conser_post, n_components = 2, metric = "cosine", n_neighbors =
df_umap_conser_post <- as.data.frame(umap_conser_post[["layout"]])
df_umap_conser_post$word <- rownames(df_umap_conser_post)
colnames(df_umap_conser_post) <- c("Post_Cons1", "Post_Cons2", "word")

ggplot(df_umap_conser_post) +
  geom_point(aes(x = Post_Cons1, y = Post_Cons2), color = 'blue', size = 0.05) +
  labs(title = "Conservative Party (Post-Crisis): Word Embeddings of GloVe and UMAP") +
  theme_minimal()
```

Conservative Party (Post–Crisis): Word Embeddings of GloVe and UMAP



Post-crisis Conservative party word embedding of words related to *immigration*

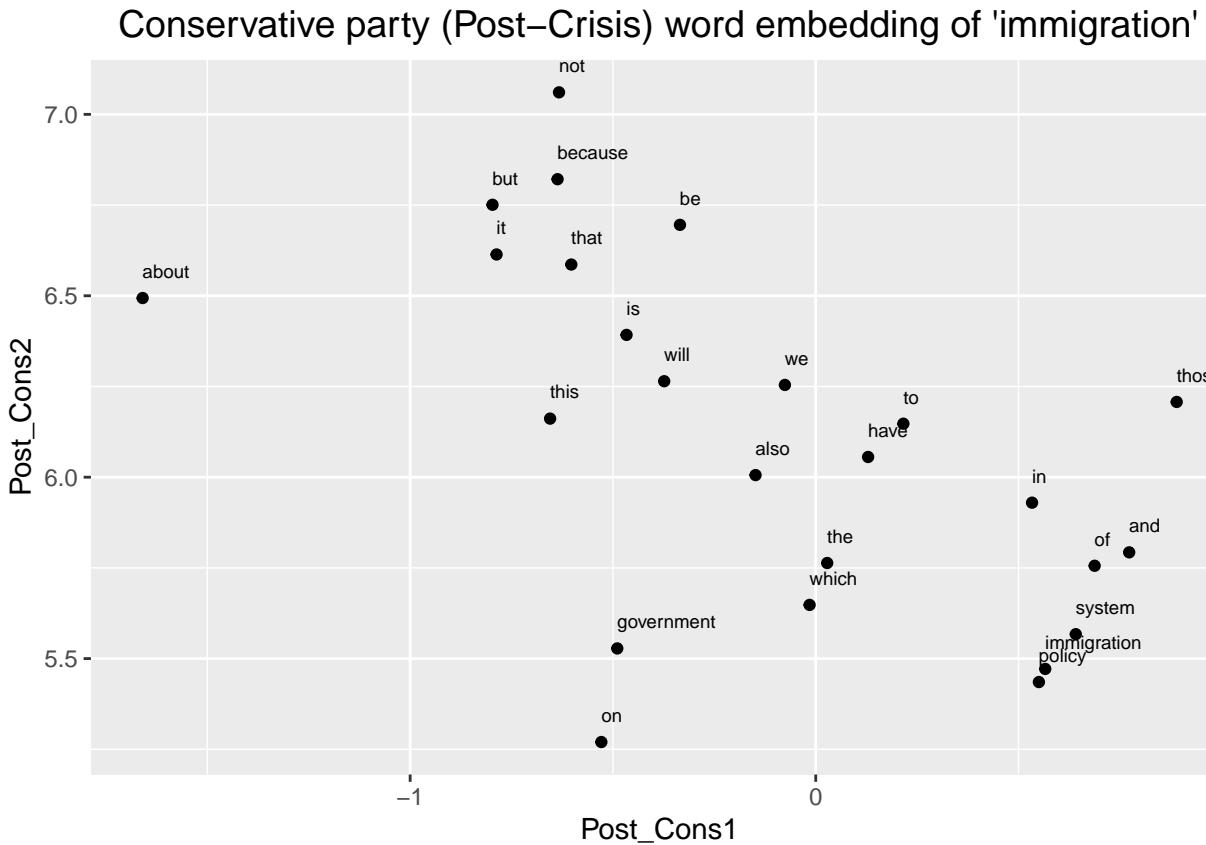
```
# Plot the word embedding of words that are related for the GloVe model (Case1: immigration)
word_conser_post_1 <- glove_embedding_conser_post[["immigration",], drop = FALSE]
cos_sim = sim2(x = glove_embedding_conser_post, y = word_conser_post_1, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
```

```

selected_words_conser_post_1 <- df_umap_conser_post %>%
  inner_join(y=select, by= "word")

#The ggplot visual for GloVe
ggplot(selected_words_conser_post_1, aes(x = Post_Cons1, y = Post_Cons2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Post_Cons1, Post_Cons2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=.5) +
  labs(title = "Conservative party (Post-Crisis) word embedding of 'immigration'") +
  theme(plot.title = element_text(hjust = .5, size = 14))

```



Post-crisis Conservative party word embedding of words related to *immigrants*

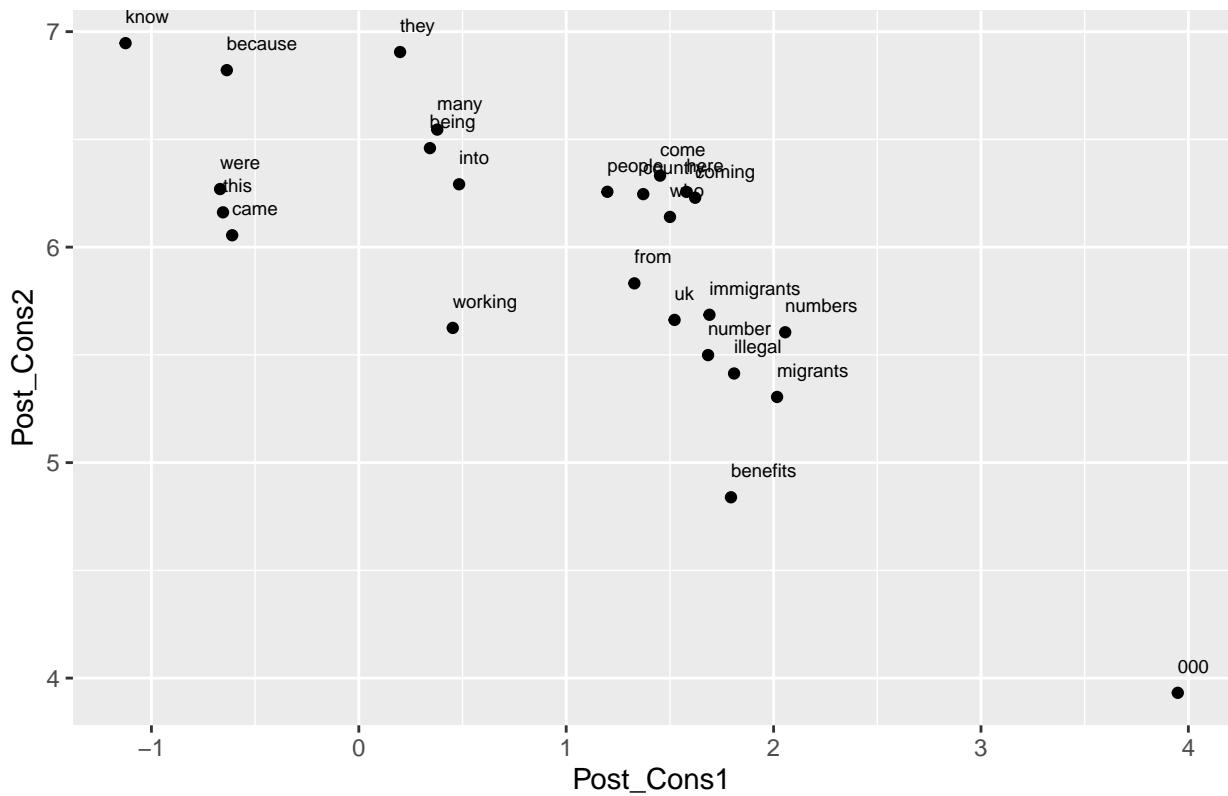
```

# Plot the word embedding of words that are related for the GloVe model (case2: immigrants)
word_conser_post_2 <- glove_embedding_conser_post["immigrants",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_conser_post, y = word_conser_post_2, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
selected_words_conser_post_2 <- df_umap_conser_post %>%
  inner_join(y=select, by= "word")

#The ggplot visual for GloVe
ggplot(selected_words_conser_post_2, aes(x = Post_Cons1, y = Post_Cons2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Post_Cons1, Post_Cons2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=.5) +
  labs(title = "Conservative party (Post-Crisis) word embedding of 'immigrants'") +
  theme(plot.title = element_text(hjust = .5, size = 14))

```

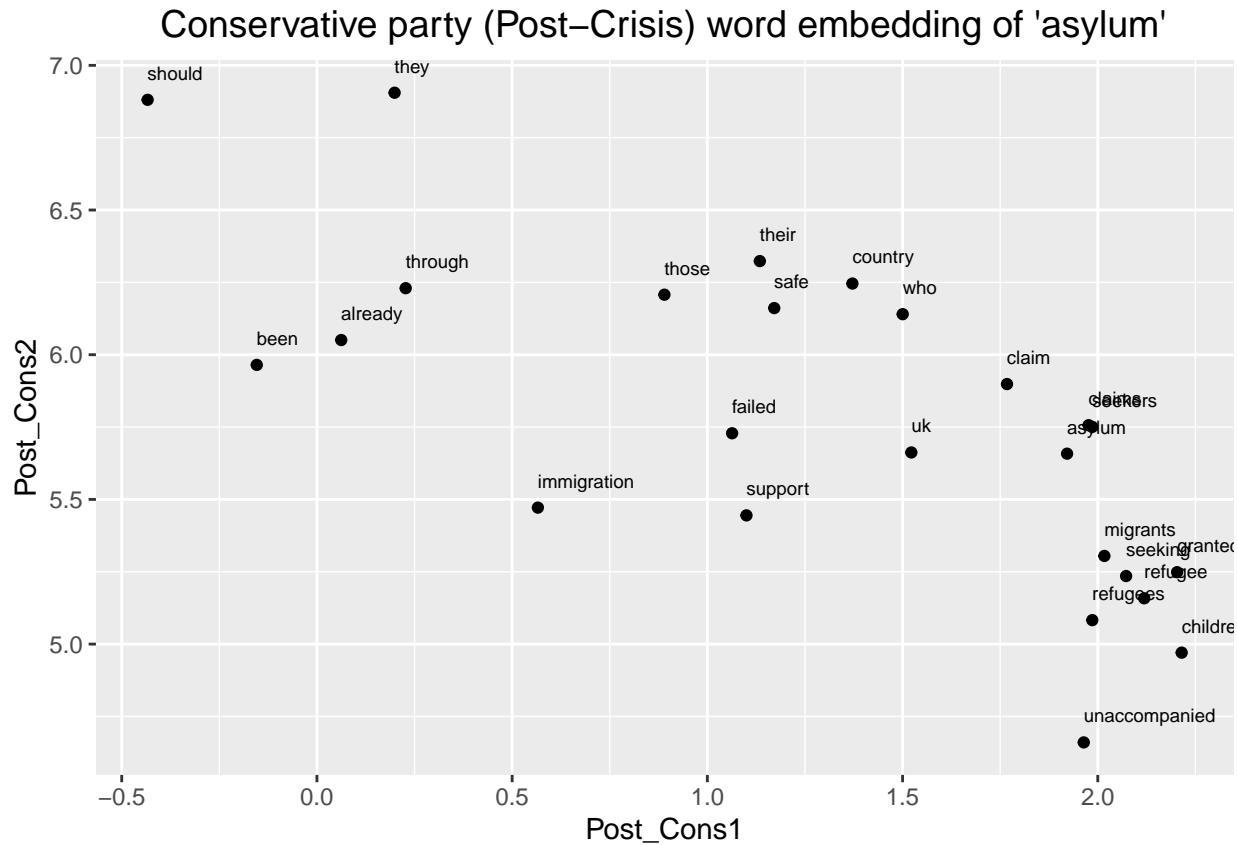
Conservative party (Post–Crisis) word embedding of 'immigrants'



Post-crisis Conservative party word embedding of words related to *asylum*

```
# Plot the word embedding of words that are related for the GloVe model (Case3: asylum)
word_conser_post_4 <- glove_embedding_conser_post["asylum",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_conser_post, y = word_conser_post_4, method = "cosine", norm = "12")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_conser_post_4 <- df_umap_conser_post %>%
  inner_join(y=select, by= "word")
```

```
#The ggplot visual for GloVe
ggplot(selected_words_conser_post_4, aes(x = Post_Cons1, y = Post_Cons2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Post_Cons1, Post_Cons2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=.5) +
  labs(title = "Conservative party (Post-Crisis) word embedding of 'asylum'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```



(3) The GloVe model for Labour party before the 2008 economic crisis

Training process

```
set.seed(42L)
glove_text_labour_pre <- sample(data_labour_pre$desc)

tokens_labour_pre <- space_tokenizer(glove_text_labour_pre)
it_labour_pre <- itoken(tokens_labour_pre, progressbar = FALSE)
vocab_labour_pre <- create_vocabulary(it_labour_pre)
vocab_pruned_labour_pre <- prune_vocabulary(vocab_labour_pre, term_count_min = COUNT_MIN)

vectorizer_labour_pre <- vocab_vectorizer(vocab_pruned_labour_pre)
tcm_labour_pre <- create_tcm(it_labour_pre, vectorizer_labour_pre, skip_grams_window = WINDOW_SIZE, skip_n_grams = 1)

glove_labour_pre <- GlobalVectors$new(rank = DIM, x_max = 100, learning_rate = 0.05)
word_vectors_main_labour_pre <- glove_labour_pre$fit_transform(tcm_labour_pre, n_iter = ITERS, convergence_threshold = 0.001)

word_vectors_context_labour_pre <- glove_labour_pre$components
glove_embedding_labour_pre <- word_vectors_main_labour_pre + t(word_vectors_context_labour_pre)

saveRDS(glove_embedding_labour_pre, file = "local_glove_labour_pre.rds")
```

Modelling takes time. So I will use the prepared model made by same process for knitting.

```
url_labour_pre <- "https://github.com/RiverKim-garam/CTA24-Final-assessment/blob/main/local_glove_labour.R"
glove_embedding_labour_pre <- readRDS(url(url_labour_pre, method = "libcurl"))
```

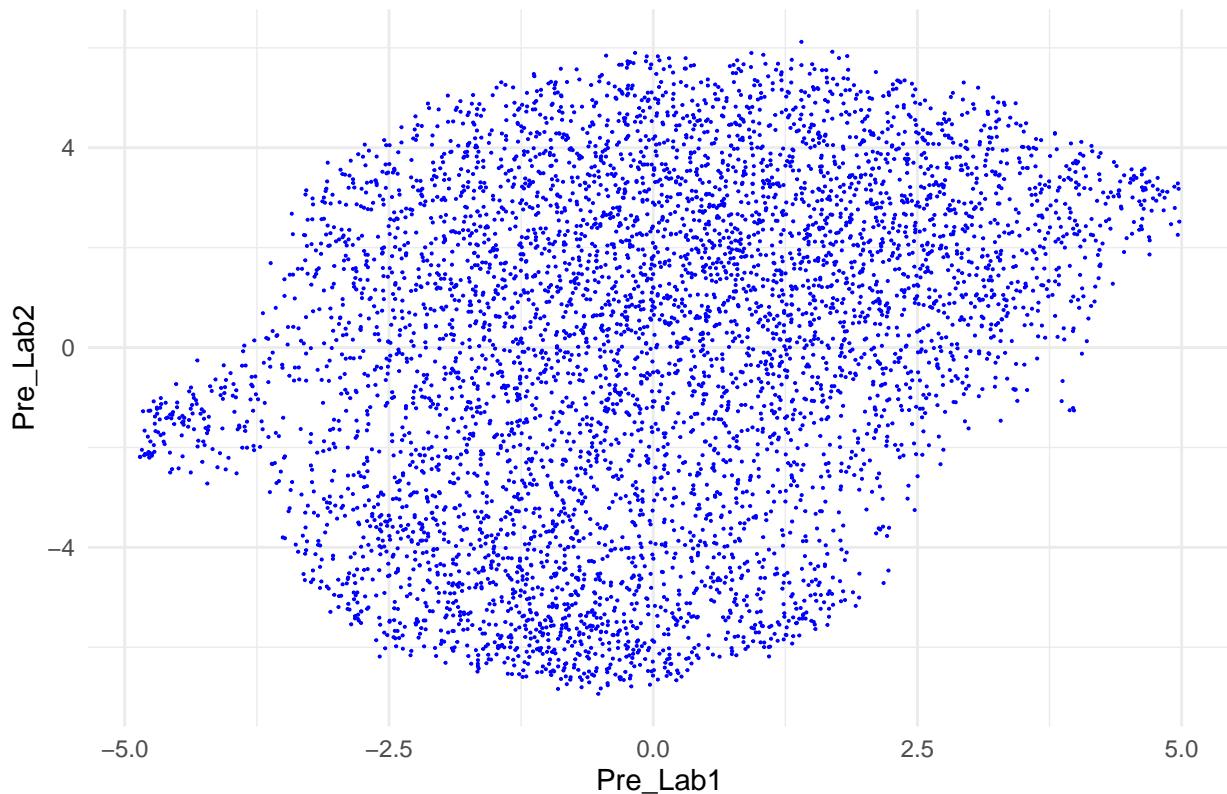
Total GloVe word embedding two dimensional umap

```
# Plotting the whole word embeddings of pre-crisis Labour party immigration related text
umap_labour_pre <- umap(glove_embedding_labour_pre, n_components = 2, metric = "cosine", n_neighbors = 10)

df_umap_labour_pre <- as.data.frame(umap_labour_pre[["layout"]])
df_umap_labour_pre$word <- rownames(df_umap_labour_pre)
colnames(df_umap_labour_pre) <- c("Pre_Lab1", "Pre_Lab2", "word")

ggplot(df_umap_labour_pre) +
  geom_point(aes(x = Pre_Lab1, y = Pre_Lab2), color = 'blue', size = 0.05) +
  labs(title = "Labour Party (Pre-Crisis): Word Embeddings of GloVe and UMAP") +
  theme_minimal()
```

Labour Party (Pre–Crisis): Word Embeddings of GloVe and UMAP



Pre-crisis Labour party word embedding of words related to *immigration*

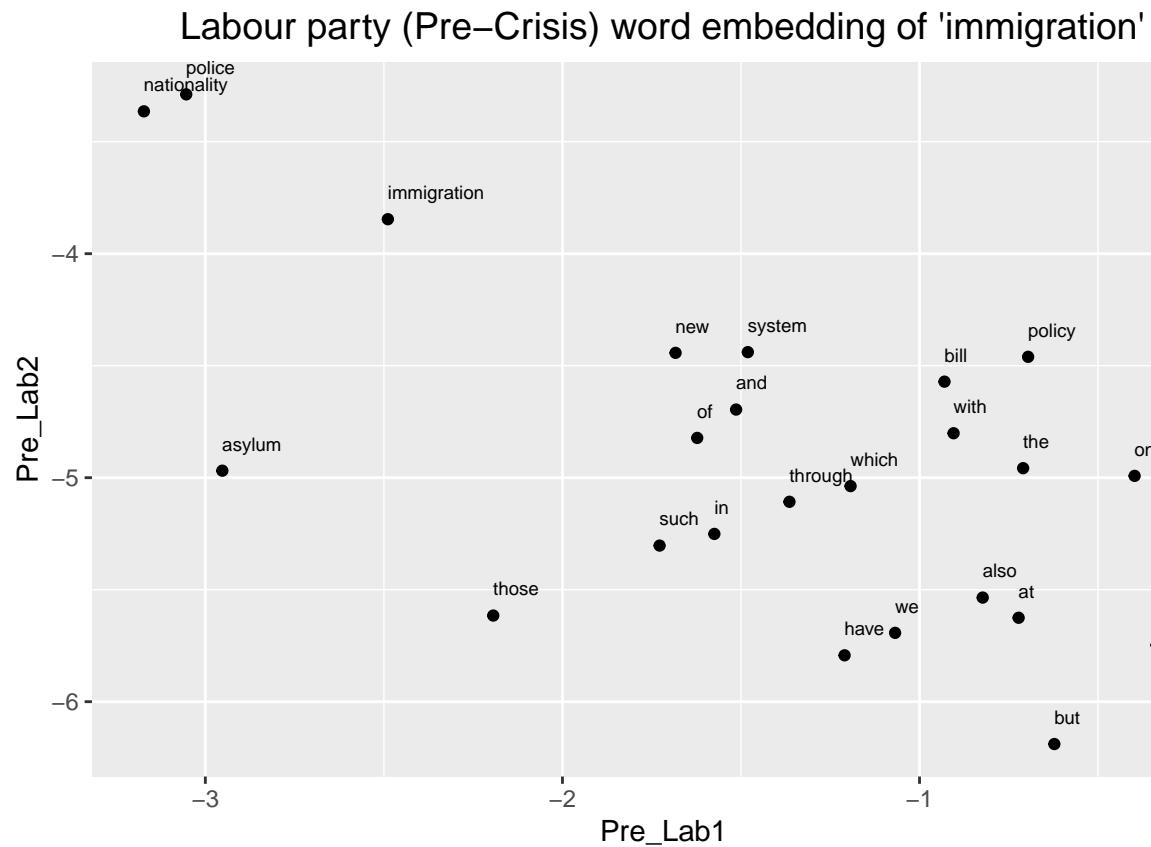
```
# Plot the word embedding of words that are related for the GloVe model (case1: immigration)
word_labour_pre_1 <- glove_embedding_labour_pre[["immigration"],, drop = FALSE]
cos_sim = sim2(x = glove_embedding_labour_pre, y = word_labour_pre_1, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
```

```

selected_words_labour_pre_1 <- df_umap_labour_pre %>%
  inner_join(y=select, by= "word")

#The ggplot visual for GloVe
ggplot(selected_words_labour_pre_1, aes(x = Pre_Lab1, y = Pre_Lab2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Pre_Lab1, Pre_Lab2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0)
  labs(title = "Labour party (Pre-Crisis) word embedding of 'immigration'" +
    theme(plot.title = element_text(hjust = .5, size = 14))

```



Pre-crisis Labour party word embedding of words related to *immigrants*

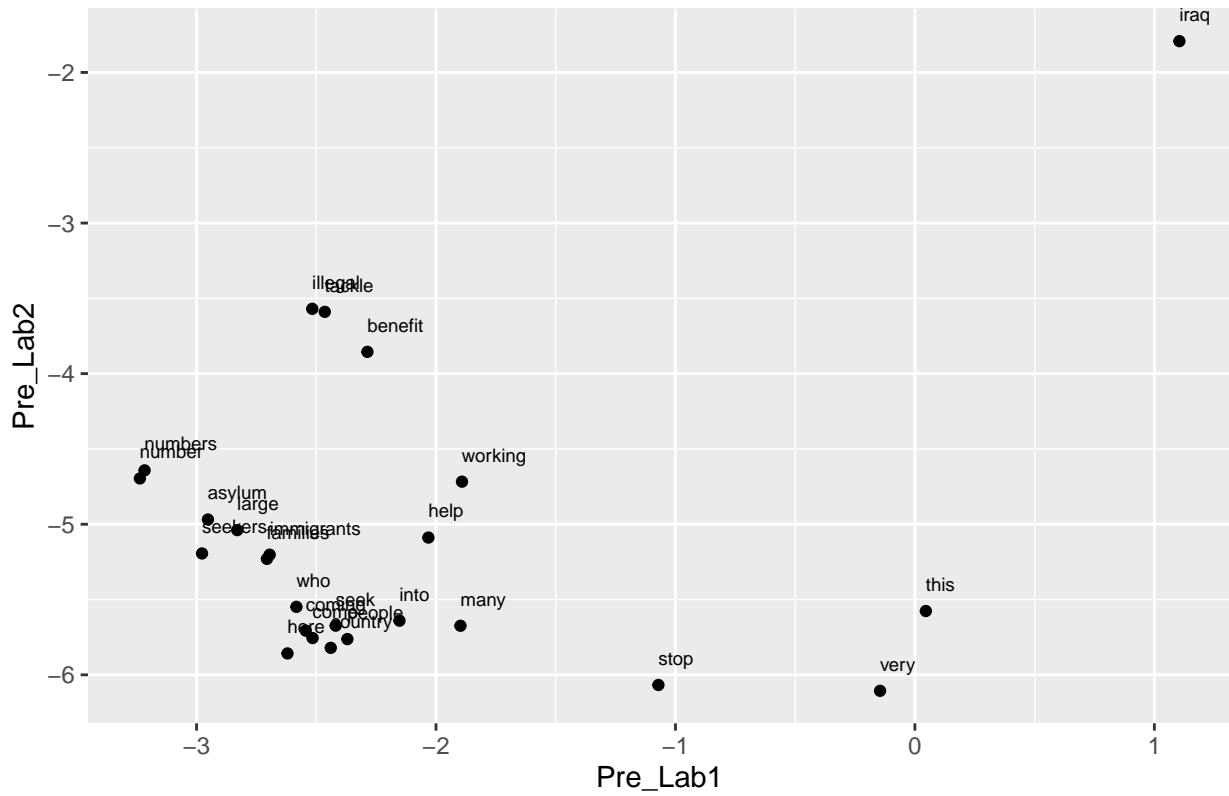
```

# Plot the word embedding of words that are related for the GloVe model (Case2: immigrants)
word_labour_pre_2 <- glove_embedding_labour_pre["immigrants",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_labour_pre, y = word_labour_pre_2, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
selected_words_labour_pre_2 <- df_umap_labour_pre %>%
  inner_join(y=select, by= "word")

#The ggplot visual for GloVe
ggplot(selected_words_labour_pre_2, aes(x = Pre_Lab1, y = Pre_Lab2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Pre_Lab1, Pre_Lab2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0)
  labs(title = "Labour party (Pre-Crisis) word embedding of 'immigrants'" +
    theme(plot.title = element_text(hjust = .5, size = 14))

```

Labour party (Pre–Crisis) word embedding of 'immigrants'

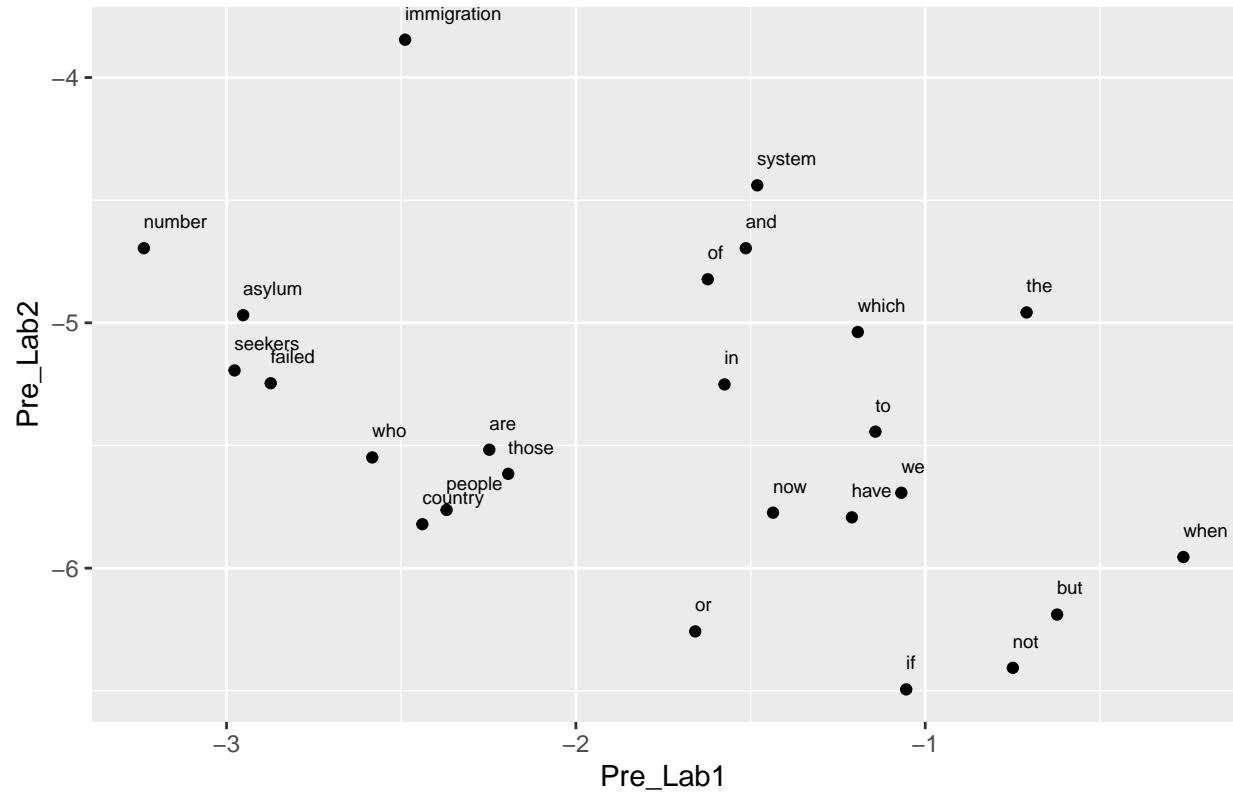


Pre-crisis Labour party word embedding of words related to *asylum*

```
# Plot the word embedding of words that are related for the GloVe model (case3: asylum)
word_labour_pre_3 <- glove_embedding_labour_pre[\"asylum\",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_labour_pre, y = word_labour_pre_3, method = "cosine", norm = "12")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_labour_pre_3 <- df_umap_labour_pre %>%
  inner_join(y=select, by= "word")
```

```
#The ggplot visual for GloVe
ggplot(selected_words_labour_pre_3, aes(x = Pre_Lab1, y = Pre_Lab2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Pre_Lab1, Pre_Lab2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0)
  labs(title = "Labour party (Pre-Crisis) word embedding of 'asylum'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

Labour party (Pre–Crisis) word embedding of 'asylum'



(4) The GloVe model for Labour party after the 2008 economic crisis

Training process

```

set.seed(42L)
glove_text_labour_post <- sample(data_labour_post$desc)

tokens_labour_post <- space_tokenizer(glove_text_labour_post)
it_labour_post <- itoken(tokens_labour_post, progressbar = FALSE)
vocab_labour_post <- create_vocabulary(it_labour_post)
vocab_pruned_labour_post <- prune_vocabulary(vocab_labour_post, term_count_min = COUNT_MIN)

vectorizer_labour_post <- vocab_vectorizer(vocab_pruned_labour_post)
tcm_labour_post <- create_tcm(it_labour_post, vectorizer_labour_post, skip_grams_window = WINDOW_SIZE, 

glove_labour_post <- GlobalVectors$new(rank = DIM, x_max = 100, learning_rate = 0.05)
word_vectors_main_labour_post <- glove_labour_post$fit_transform(tcm_labour_post, n_iter = ITERS, converge = 0.001)

word_vectors_context_labour_post <- glove_labour_post$components
glove_embedding_labour_post <- word_vectors_main_labour_post + t(word_vectors_context_labour_post)

saveRDS(glove_embedding_labour_post, file = "local_glove_labour_post.rds")

```

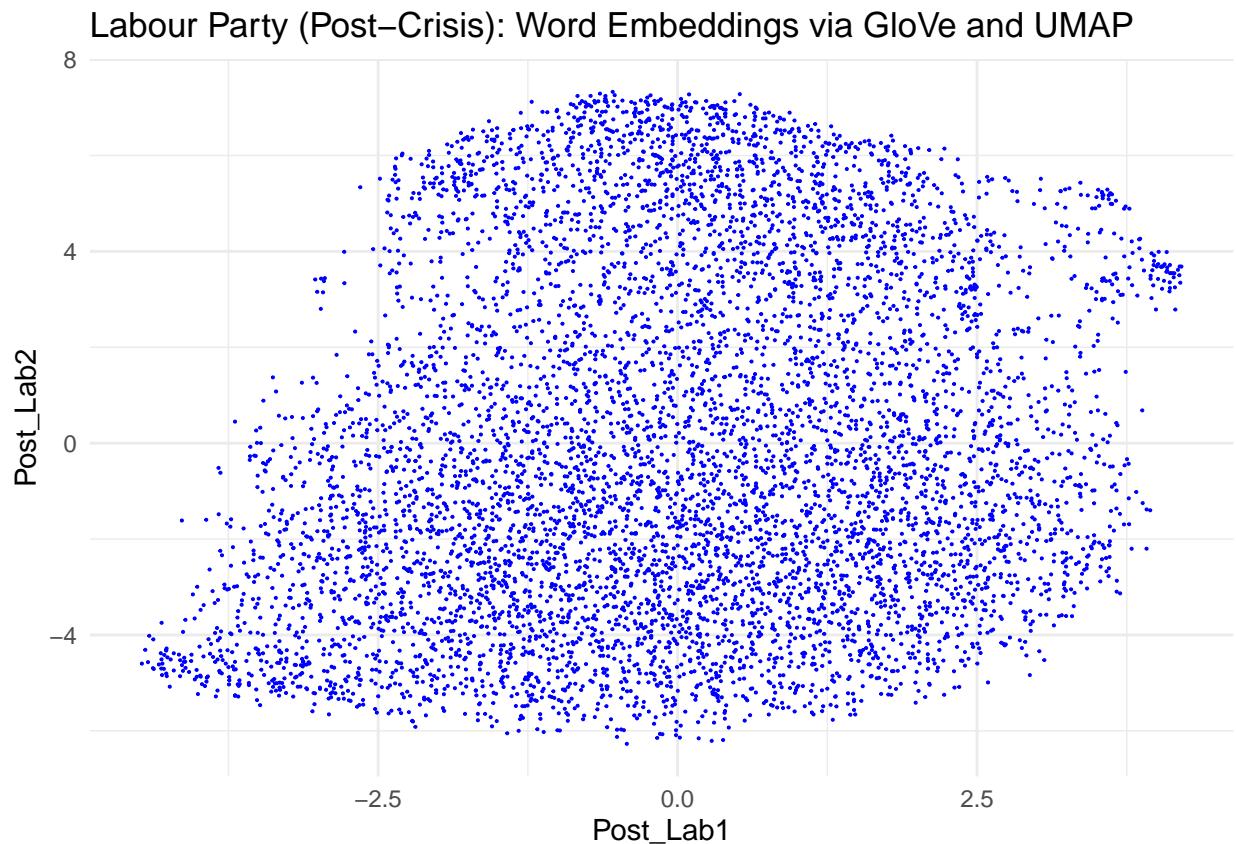
Modelling takes time. So I will use the prepared model made by same process for knitting.

```
url_labour_post <- "https://github.com/RiverKim-garam/CTA24-Final-assessment/blob/main/local_glove_labour.R"
glove_embedding_labour_post <- readRDS(url(url_labour_post, method = "libcurl"))
```

Total GloVe word embedding two dimensional umap

```
# Plotting the whole word embeddings of post-crisis Labour party immigration related text
umap_labour_post <- umap(glove_embedding_labour_post, n_components = 2, metric = "cosine", n_neighbors = 10)
df_umap_labour_post <- as.data.frame(umap_labour_post[["layout"]])
df_umap_labour_post$word <- rownames(df_umap_labour_post)
colnames(df_umap_labour_post) <- c("Post_Lab1", "Post_Lab2", "word")

ggplot(df_umap_labour_post) +
  geom_point(aes(x = Post_Lab1, y = Post_Lab2), color = 'blue', size = 0.05) +
  labs(title = "Labour Party (Post-Crisis): Word Embeddings via GloVe and UMAP") +
  theme_minimal()
```



Post-crisis Labour party word embedding of words related to *immigration*

```
# Plot the word embedding of words that are related for the GloVe model (case1: immigration)
word_labour_post_1 <- glove_embedding_labour_post[["immigration",], drop = FALSE]
cos_sim = sim2(x = glove_embedding_labour_post, y = word_labour_post_1, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
```

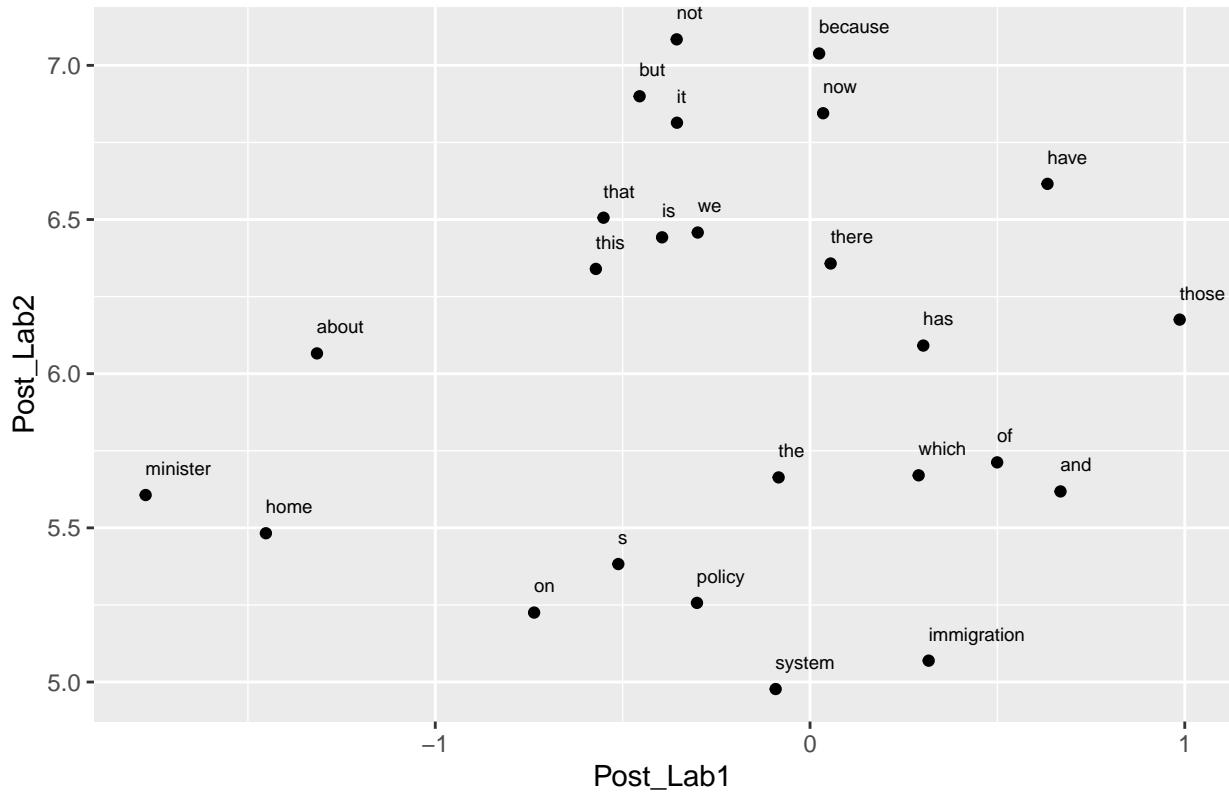
```

selected_words_labour_post_1 <- df_umap_labour_post %>%
  inner_join(y=select, by= "word")

#The ggplot visual for GloVe
ggplot(selected_words_labour_post_1, aes(x = Post_Lab1, y = Post_Lab2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Post_Lab1, Post_Lab2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0) +
  labs(title = "Labour party (Post-Crisis) word embedding of 'immigration'") +
  theme(plot.title = element_text(hjust = .5, size = 14))

```

Labour party (Post–Crisis) word embedding of 'immigration'



Pre-crisis Labour party word embedding of words related to *immigrants*

```

# Plot the word embedding of words that are related for the GloVe model (case2: immigrants)
word_labour_post_2 <- glove_embedding_labour_post["immigrants",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_labour_post, y = word_labour_post_2, method = "cosine", norm = "l2")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25))))
colnames(select) <- "word"
selected_words_labour_post_2 <- df_umap_labour_post %>%
  inner_join(y=select, by= "word")

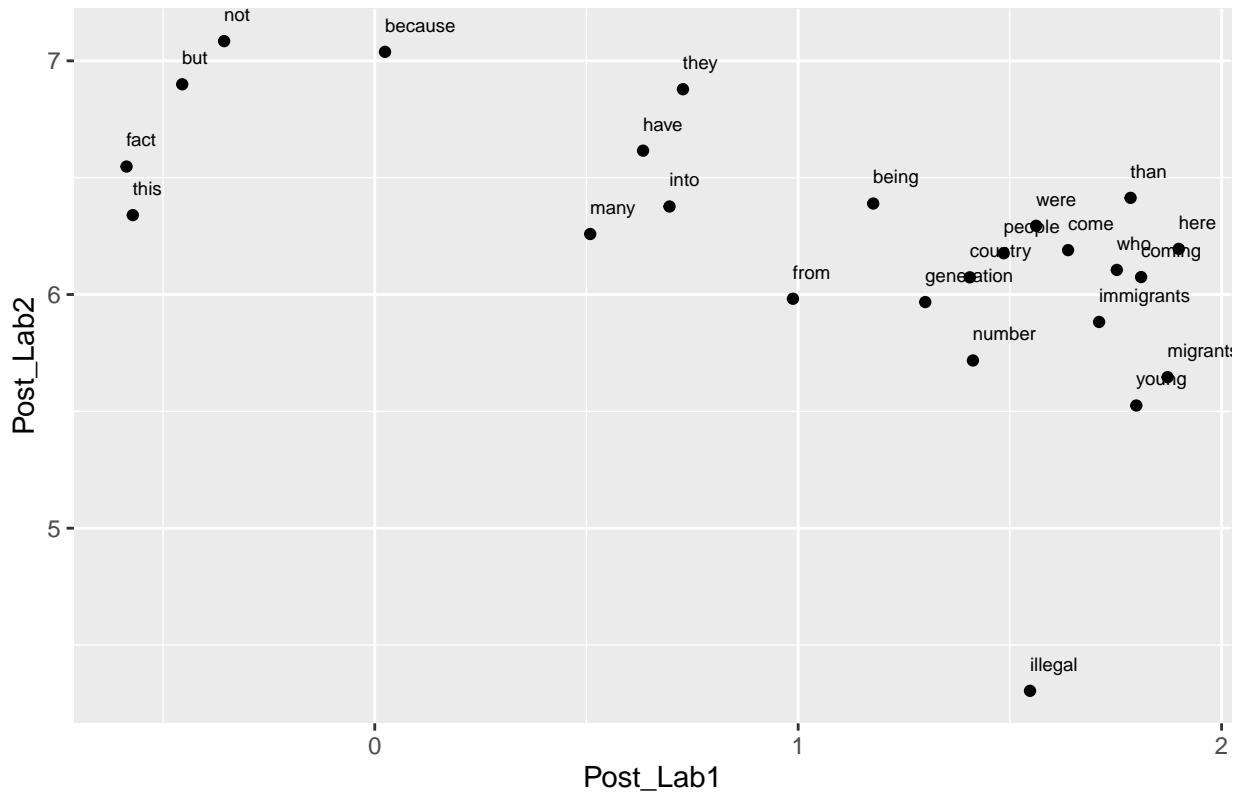
```

```

#The ggplot visual for GloVe
ggplot(selected_words_labour_post_2, aes(x = Post_Lab1, y = Post_Lab2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Post_Lab1, Post_Lab2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0) +
  labs(title = "Labour party (Post-Crisis) word embedding of 'immigrants'") +
  theme(plot.title = element_text(hjust = .5, size = 14))

```

Labour party (Post–Crisis) word embedding of 'immigrants'

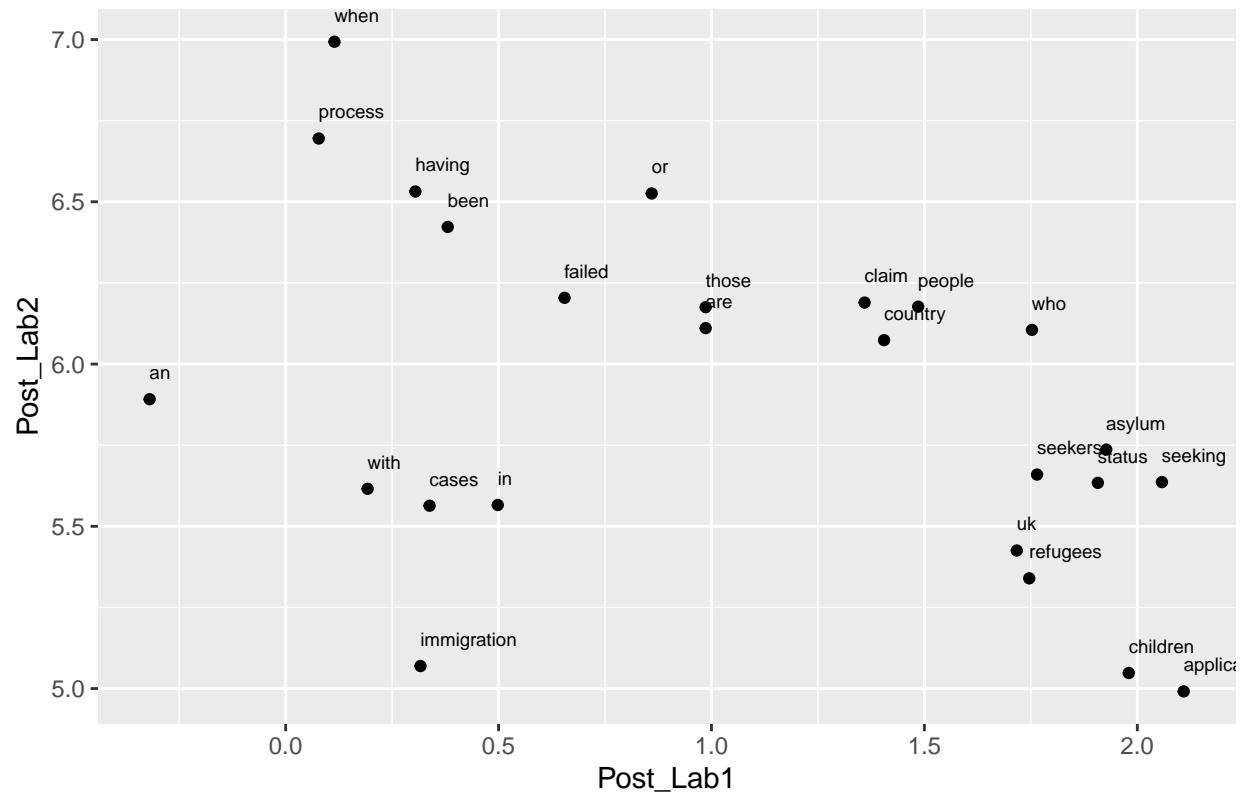


Pre-crisis Labour party word embedding of words related to *asylum*

```
# Plot the word embedding of words that are related for the GloVe model (case3: asylum)
word_labour_post_3 <- glove_embedding_labour_post[\"asylum\",, drop = FALSE]
cos_sim = sim2(x = glove_embedding_labour_post, y = word_labour_post_3, method = "cosine", norm = "12")
select <- data.frame(rownames(as.data.frame(head(sort(cos_sim[,1], decreasing = TRUE), 25)))
colnames(select) <- "word"
selected_words_labour_post_3 <- df_umap_labour_post %>%
  inner_join(y=select, by= "word")
```

```
#The ggplot visual for GloVe
ggplot(selected_words_labour_post_3, aes(x = Post_Lab1, y = Post_Lab2)) +
  geom_point(show.legend = FALSE) +
  geom_text(aes(Post_Lab1, Post_Lab2, label = word), show.legend = FALSE, size = 2.5, vjust=-1.5, hjust=0)
  labs(title = "Labour party (Post-Crisis) word embedding of 'asylum'") +
  theme(plot.title = element_text(hjust = .5, size = 14))
```

Labour party (Post–Crisis) word embedding of 'asylum'



Now we made three different specific words (immigration, immigrants, asylum) GloVe embedding umap by party and the period. By comparing the connections of words, we can estimate the contexts how the words were used.

```
# Instead of LaTeX, use TinyTeX to knit the mark down result into pdf
library(tinytex)
```