Problem Set 3

Natalia Sarabia Vasquez

September 29, 2021

1 Problem 1

Rather that writing modular functions, I made use of the functions lapply and map and write modular chunks that result in specific outputs.

I downloaded all the URLs of the page of interest. I realized that the dowloaded data contained URLs that do not belong to the table. After inspecting the website, I used regular expressions to filter the URLs I will use:

```
URL <- "https://www.presidency.ucsb.edu/documents/presidential-documents-archive-guidebook/annual-messag

# The URL's string is too big. I will paste it here as a comment. I cannot break the line.
# "https://www.presidency.ucsb.edu/documents/presidential-documents-archive-
# guidebook/annual-messages-congress-the-state-the-union#axzz265cEKp1a"

links <- read_html(URL) %>%
  html_elements("[href]") %>%
  html_attr('href') %>%
  as.data.frame() %>%
  rename("link" = ".") %>%
  filter(str_detect(link,
  "(^https://www.presidency.ucsb.edu){1}.*(ws{1}|address-before-joint-session-the-congress{1}|annual-mes

# The regular expression I used to clean the data is too long. As i cannot break
# the line, I will paste it here as a comment:
# "(^https://www.presidency.ucsb.edu){1}.*(ws{1}|address-before-joint-session-the
# -congress{1}|annual-message-{1})"
```

Then, I construct three main data sets to complete our tasks: speeches, words and sentences. I removed the text that was not said by the president using the function replace:

```
# Get the information of each speech: body, date and name of the president.
speeches <- links[1:257,] %>%
  map(read_html) %>%
  map(html_nodes,".field-docs-content, .date-display-single, .diet-title") %>%
  map(html_text)

# Get the sentences of each speech. As sentence delimiters, I used periods
# followed by spaces, semicolons, question and exclamation marks
# and quotation marks:
sentences <- speeches %>%
  lapply("[[",3) %>%
  str_replace_all("(\\[.*?\\])|(^ )","")  %>%
  str_trim() %>%
  str_split('(\\. )|(\\n)|(\\? )|(\\! )|(\\; )|(\\." )') %>%
```

```
  lapply(function(x) x[!x %in% ""])

# Get the words contained in each speech
# I removed the text not said by the president, and some punctuation (I  kept
# apostrophes (to deal with other problem later on), and dashes.)
words <- speeches %>%
  lapply("[[",3) %>%
  str_replace_all("(\\[.*?\\])|[^[:alnum:]\\-\\.\\-\\$\\%\\' ]","") %>%
  str_split("([. ])|(\\n)|(-)|(-)|(-)") %>%
  lapply(function(x) x[!x %in% ""])
```

```
# First sentences of the first speech in my set
head(sentences[[1]])
```

```
## [1] "The President"
## [2] "Thank you"
## [3] "Thank you"
## [4] "Thank you"
## [5] "Good to be back"
## [6] "As Mitch and Chuck will understand, it's good to be almost home, down the hall"
```

```
# First words of the first speech in my set
head(words[[1]])
```

```
## [1] "The"       "President" "Thank"     "you"       "Thank"     "you"
```

I will calculate, in modular form, the information that was asked for. My idea is to create character vectors which columns I will append at the very end to concentrate all the information in a data frame and be able to plot the results.

Here, I obtain the date, the name of each of the presidents who gave the speech, the number of words, characters, the average word length, the number of occurrences of '[[laughter]]' and '[[appaluse]]' (previously, I converted to lower case).

```
# Isolate the information of dates of each speech
dates <- speeches %>%
  lapply("[[",2) %>%
  unlist()

# Isolate the information of the names of the presidents of each speech
presidents <- speeches %>%
  lapply("[[",1) %>%
  unlist()

# Get the number of words of each speech
n_words <- words %>%
  lapply(function(x){length(unlist(x))}) %>%
  unlist()

# Get the average word length of each speech
n_chara <- words %>%
  lapply(function(x){nchar(x)}) %>%
  lapply(sum) %>%
  unlist()

# Get the average word length of each speech
```

```r
avg_word_length <- words %>%
  lapply(function(x){mean(nchar(x))}) %>%
  unlist()

# Isolate from the speeches the text, convert to lower case, and remove blanks
laughter_applause <- speeches %>%
  lapply("[[",3) %>%
  lapply(unlist) %>%
  lapply(str_to_lower) %>%
  str_replace_all(' ',"")

# Detect "[[laughter]]", then count how many occurrences it has
laughter <- laughter_applause %>%
  str_extract_all('(\\[laughter\\])') %>%
  lapply(function(x){length(unlist(x))}) %>%
  unlist()

# Detect "[[applause]]", then count how many occurrences it has
applause <- laughter_applause %>%
  str_extract_all('(\\[applause\\])') %>%
  lapply(function(x){length(unlist(x))}) %>%
  unlist()
```

```r
# First dates in my set
head(dates)
```

```
## [1] "April 28, 2021"    "February 28, 2017" "January 30, 2018"
## [4] "February 05, 2019" "February 04, 2020" "February 12, 2013"
```

```r
# First presidents' names in my set
head(presidents)
```

```
## [1] "Joseph R. Biden" "Donald J. Trump" "Donald J. Trump" "Donald J. Trump"
## [5] "Donald J. Trump" "Barack Obama"
```

```r
# First number of words in my set
head(n_words)
```

```
## [1] 8059 5028 5888 5649 6333 6847
```

```r
# First number of characters in my set
head(n_chara)
```

```
## [1] 36559 23359 27175 26395 30262 31789
```

```r
# First average word length in my set
head(avg_word_length)
```

```
## [1] 4.536419 4.645784 4.615319 4.672508 4.778462 4.642763
```

```r
# First number of laughters detected in my set
head(laughter)
```

```
## [1] 4 4 6 8 2 1
```

```r
# First number of applause detected in my set
head(applause)
```

```
## [1] 1 3 6 8 4 4
```

In order to achieve the counting of patterns, I will use lapply twice. The idea is to iterate first on the list of patterns and then on each speech. I will use the function str_count for each pattern and each speech. Finally, I convert the resulting list to a data frame:

```
# Patterns we are interested on
patterns <- c("I[( )$|'$]","[Ww]e[( )$|'$]","America[ |(n)|(')$]",
              "democra[(cy)$|(tic)$]","\\brepublic\\b","Democrat[ $|(ic)$]",
              "\\bRepublican\\b","free[ |(dom)$]","\\bwar\\b","\\bGod\\b",
              "God bless","(Jesus)|(Christ)|(Christian)","\\bMexico\\b")

# Counting the number of occurrences of each pattern in each speech
counts <- lapply(patterns, function(p) lapply(speeches,
                                              function(y) {str_count(string=y,
                                                                     pattern = p)}))

# The result of lapply is a list. I will convert it into a data frame in order
# to be able to append the columns with the rest of the information:
counting <- counts %>%
  map_df(as_tibble,.name_repair = "unique") %>%
  rownames_to_column( var = "row") %>%
  filter(row %in% c("3","6","9","12","15","18","21","24","27","30","33","36","39")) %>%
  select(-row) %>%
  t() %>%
  as.data.frame()
```

I aggregated all the requested measures in a data frame. I also filtered the data by year and generated some new columns to facilitate the data management:

```
# Concatenate the character vectors I have created, add names to simplify data management:
data <- cbind(presidents,dates,n_words,n_chara,avg_word_length,laughter,applause,
              counting)

# Rename each column
names(data) <- c("President","Date","n_words","n_chara","avg_word_length","n_laughter",
                 "n_applause","n_I","n_we","n_America","n_democra","n_republic",
                 "n_Democra","n_Republican","n_free","n_war","all_God",
                 "n_God_bless","n_Jesus","n_Mexico")

# I filtered the data for the graphs: restrict by year, create a variable 'party' depending
# on the party of each president, a variable to know the occurrences of  "God"
# as the difference of all the occurrences of "God" minus the occurrences of "God
# bless":
data_graphs <- data %>%
  mutate(n_God = all_God - n_God_bless,
         year = as.numeric(str_sub(Date,nchar(Date)-3,nchar(Date))),
         party = ifelse(President %in% c("Dwight D. Eisenhower","Richard Nixon",
                                         "Ronald Reagan","Gerald R. Ford",
                                         "George W. Bush","George Bush",
                                         "Donald J. Trump"),"Republican",
                        "Democrat")) %>%
  filter(year >= 1932,
         President != "Herbert Hoover") %>%
  select(-all_God)
```

```
# First observations of data set data_graphs
head(data_graphs)
```

```
##            President              Date n_words n_chara avg_word_length
## ...1 Joseph R. Biden    April 28, 2021    8059   36559        4.536419
## ...2 Donald J. Trump February 28, 2017    5028   23359        4.645784
## ...3 Donald J. Trump  January 30, 2018    5888   27175        4.615319
## ...4 Donald J. Trump February 05, 2019    5649   26395        4.672508
## ...5 Donald J. Trump February 04, 2020    6333   30262        4.778462
## ...6    Barack Obama February 12, 2013    6847   31789        4.642763
##      n_laughter n_applause n_I n_we n_America n_democra n_republic n_Democra
## ...1          4          1 128  193       112        17          0         5
## ...2          4          3  41  113        70         0          0         0
## ...3          6          6  39  137        76         0          0         0
## ...4          8          8  44  115        72         0          1         3
## ...5          2          4  60   97        89         1          1         0
## ...6          1          4  47  150        49         4          0         0
##      n_Republican n_free n_war n_God_bless n_Jesus n_Mexico n_God year
## ...1            4      3     6           1       0        0     2 2021
## ...2            2      6     4           2       1        0     1 2017
## ...3            0      4     2           1       1        2     3 2018
## ...4            1      5     4           2       0        1     3 2019
## ...5            2     10     1           2       1        3     8 2020
## ...6            0      9     3           2       0        1     1 2013
##           party
## ...1   Democrat
## ...2 Republican
## ...3 Republican
## ...4 Republican
## ...5 Republican
## ...6   Democrat
```

Finally, I plot some of the measures to show how the variables have changed by time and differentiating by the party of the president who gave the speech:

```
data_graphs %>%
ggplot(aes(x=year,y=n_words,colour = party)) +
  geom_point() +
  theme_classic() +
  ggtitle("Number of words in each speech \n since 1932") +
  xlab("Year") + ylab("Number of words") +
  theme(legend.position=c(0.7,0.9))
```

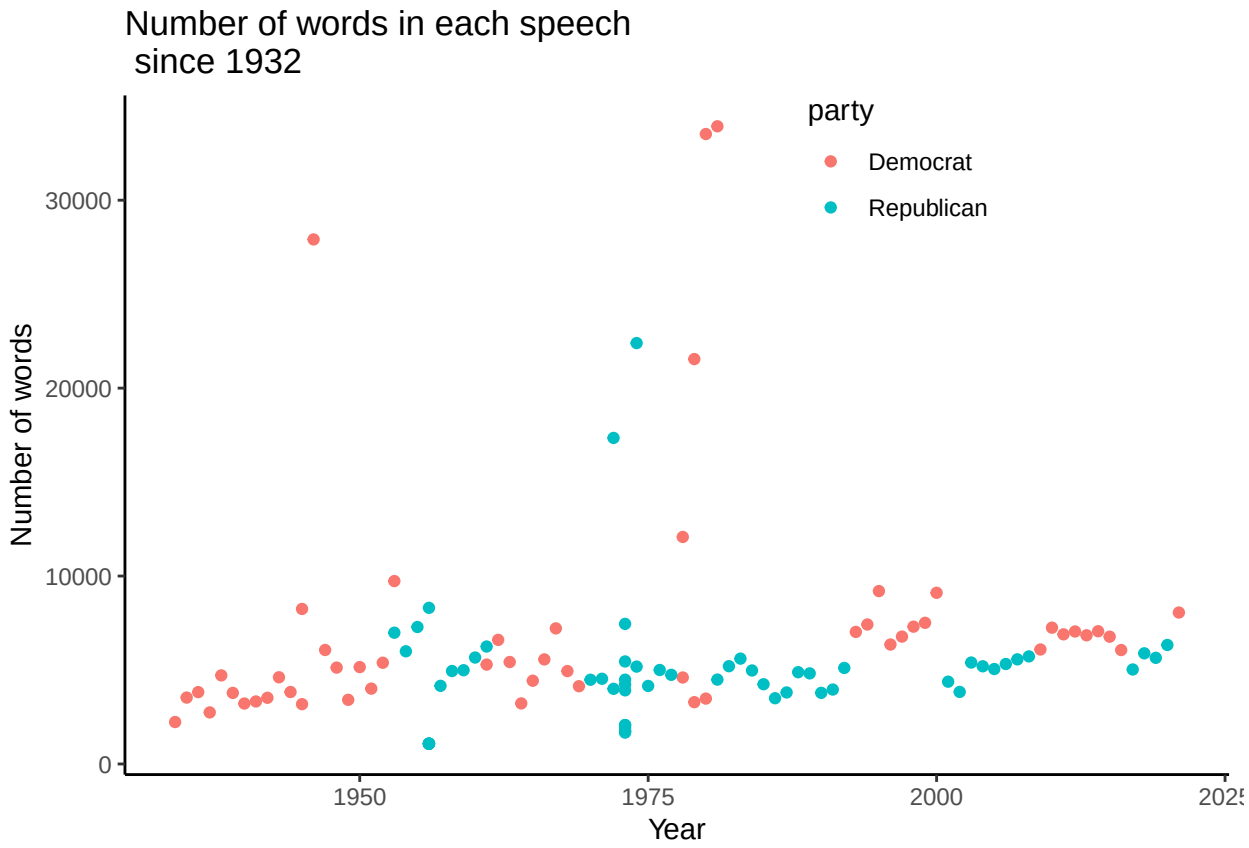


Figure 1: Number of words per speech

```
data_graphs %>%
  ggplot(aes(x=year,y=n_war,colour = party)) +
  geom_point() +
  theme_classic() +
  ggtitle("Number of times the word 'war' appears \n in each speech since 1932") +
  theme(legend.position=c(0.7,0.9)) +
  xlab("Year") + ylab("Number of words")
```

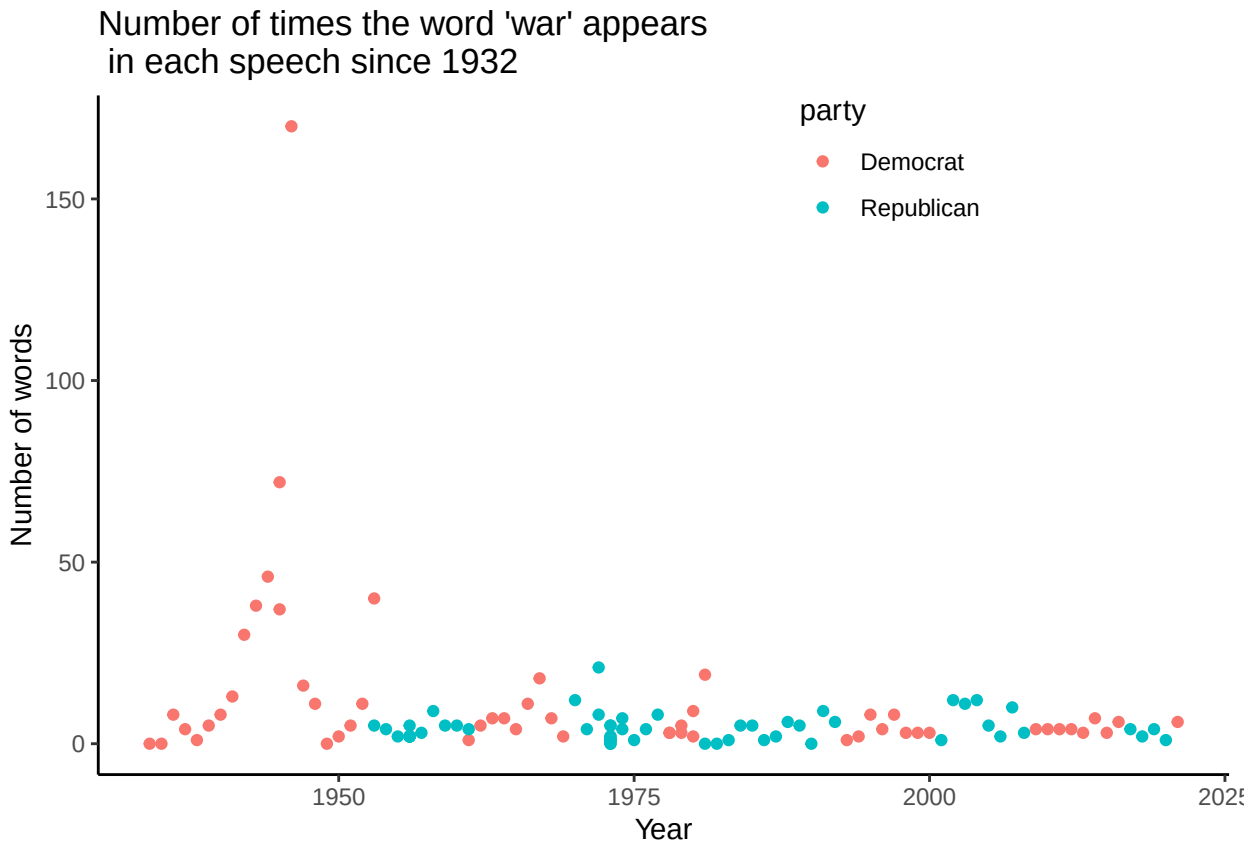


Figure 2: Number of occurences of word 'war'

```
data_graphs %>%
  ggplot(aes(x=year,y=n_God_bless,colour = party)) +
  geom_point() +
  theme_classic() +
  ggtitle("Number of times the words 'God bless' appears \n in each speach since 1932") +
  theme(legend.position=c(0.7,0.9)) +
  xlab("Year") + ylab("Number of occurences")
```

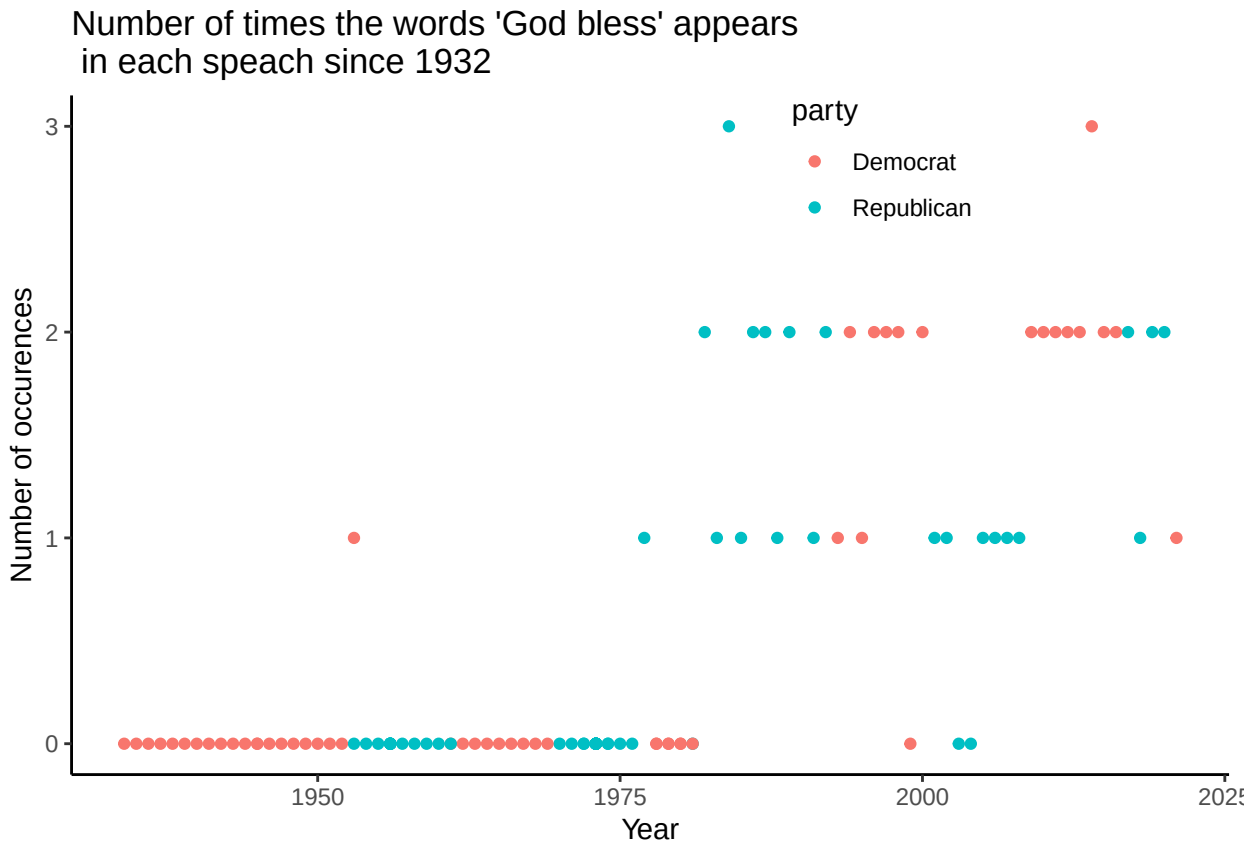


Figure 3: Number of occurrences of 'God bless'

```
data_graphs %>%
  ggplot(aes(x=year,y=n_applause,colour = party)) +
  geom_point() +
  theme_classic() +
  ggtitle("Number of times the president received an \n applause in each speech since 1932") +
  theme(legend.position=c(0.7,0.9)) +
  xlab("Year") + ylab("Counts of 'Applauses'")
```

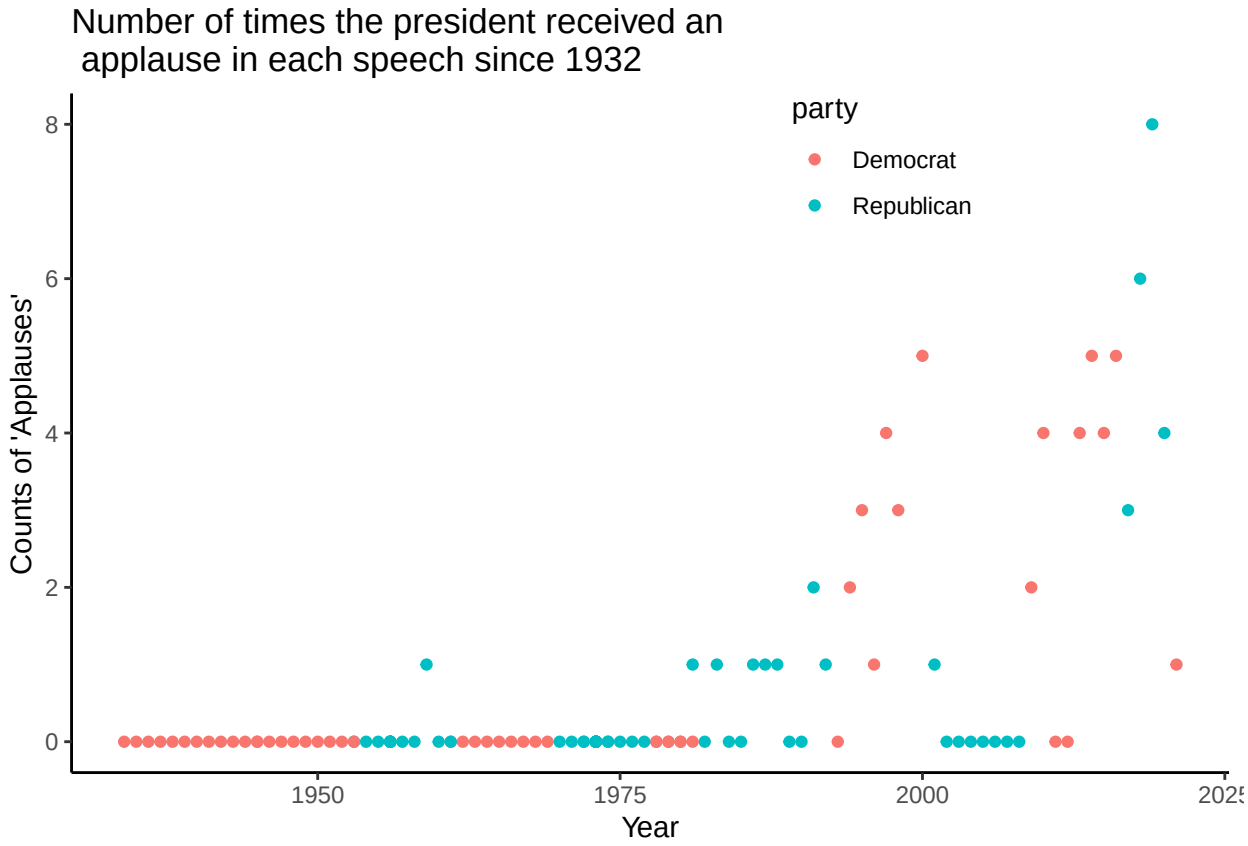


Figure 4: Number of times the string '[[Applause]]' appears in the speeches

```
data_graphs %>%
  ggplot(aes(x=year,y=n_Mexico,colour = party)) +
  geom_point() +
  theme_classic() +
  ggtitle("Number of times the word 'Mexico' appears in each speech since 1932") +
  theme(legend.position=c(0.7,0.9)) +
  xlab("Year") + ylab("Counts of occurrences")
```

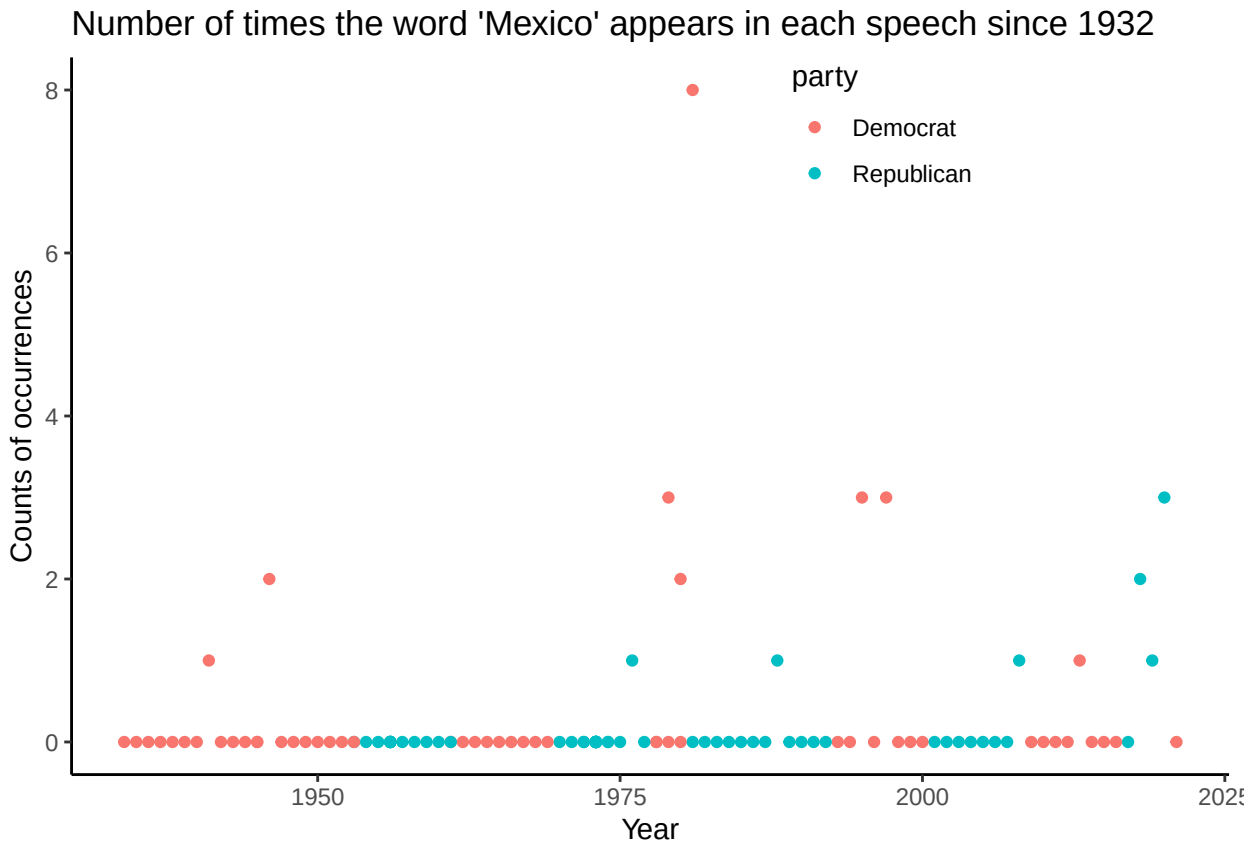


Figure 5: Number of times the words 'Mexico' appears in the speeches

For the extra credit, I made some research. In order to process the information contained in each text, I can use several approaches. The main area is Natural Language Processing and I will be working with a sub field, text analysis.

I can compute some plots as word clouds or heat maps. I also can compute sentiment analysis across the speeches. Using the bing scale (binary scale with -1 indicating negative and +1 indicating positive sentiment), I will perform sentiment analysis on each speech. Sentiment scores above 0 can be interpreted as the overall average sentiment across the all the responses is positive. I compute the sentiment analysis for each speech and then calculate the mean. I plot the mean sentiment across time for all the speeches:

```
bing_analysis <- lapply(words, get_sentiment, method = "bing") %>%
  map_dfr(summary)

bing_analysis <- cbind(data[,1:2],pull(bing_analysis[,4])) %>%
  as_data_frame() %>%
  rename("mean"="Freq") %>%
  select(-Var1)%>%
  mutate(year = as.numeric(str_sub(Date,nchar(Date)-3,nchar(Date))),
         party = ifelse(President %in% c("Dwight D. Eisenhower","Richard Nixon",
                                         "Ronald Reagan","Gerald R. Ford",
                                         "George W. Bush","George Bush",
                                         "Donald J. Trump"),"Republican",
                        "Democrat")) %>%
  filter(year >= 1932,
         President != "Herbert Hoover")
```

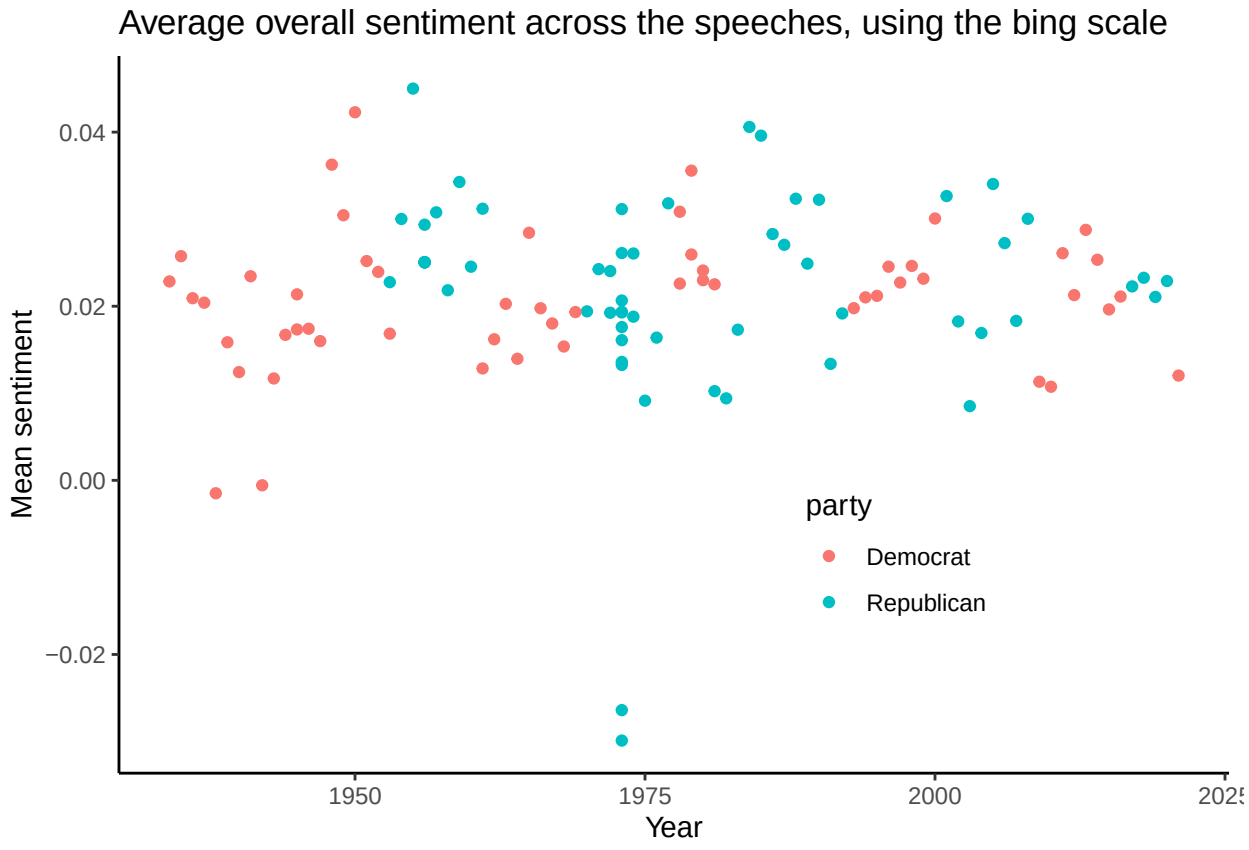
Here's how the data frame looks like:

```
head(bing_analysis )
```

```
## # A tibble: 6 x 5
##   President       Date                mean  year party
##   <chr>           <chr>              <dbl> <dbl> <chr>
## 1 Joseph R. Biden April 28, 2021    0.0120  2021 Democrat
## 2 Donald J. Trump February 28, 2017 0.0223  2017 Republican
## 3 Donald J. Trump January 30, 2018  0.0233  2018 Republican
## 4 Donald J. Trump February 05, 2019 0.0211  2019 Republican
## 5 Donald J. Trump February 04, 2020 0.0229  2020 Republican
## 6 Barack Obama    February 12, 2013 0.0288  2013 Democrat
```

A plot of the findings:

```
bing_analysis %>%
  ggplot(aes(x=year,y=mean,group = party,colour=party)) +
  geom_point() +
  theme_classic() +
  ggtitle("Average overall sentiment across the speeches, using the bing scale") +
  theme(legend.position=c(0.7,0.3)) +
  xlab("Year") + ylab("Mean sentiment")
```



2 Problem 2

I will have three classes: 'Speech', 'Plot_speech_info' and 'Download_data'. The main class will be 'speech'. This class will concentrate almost all the methods and attributes:

Class: Speech
Attributes:
Features that make an object of my class, identifiable:

- president_name: name of the president who delivered the speech
- date: date in which the speech was delivered
- body: the actual speech that was delivered

Methods:

- constructor: Method that takes as parameters the date, the body of the speech and the president's name of the speech and initializes an object of the class speech.
- speech_summary: Method that prints the summary (all info), of an object of the class speech. It takes as a parameter an object of the class speech.
- n_words: Method that calculates the number of words of each speech. It takes as a parameter an object of the class speech. First, it provides data cleaning to each speech (removes punctuation, blank spaces, etc.). Then it calculates the number of words per speech.
- n_chars: Method that calculates the number of characters per speech. It takes as a parameter an object of the class speech. First, it provides data cleaning to each speech (removes punctuation, blank spaces, etc.). Then it calculates the number of characters per speech.
- avg_word_length: Method that calculates the average word length of each speech. It takes as a parameter an object of the class speech. First, it provides data cleaning to each speech (removes punctuation, blank spaces, etc.). Then it calculates the average word length per speech.
- count_patterns: Method that receives as a parameter a list of patterns (regular expressions). Then, it searches for each pattern in every speech. It returns a data frame that has a many columns as regular expressions I provided as an input.
- rm_stop_words: Method that help us to remove all stop words the input (an object of the class speech). This method will be very important if we want to complete Natural Language Processing.
- metrics: Method that puts together the information of the speech: the date, body and president associated, the number of words, the number of characters, the number of senteces, the count of the patterns. It returns a data frame with the listed information.

The following class will help us to complete all the webscrapping in order to have the speech's body, the name of the president who delivered the speech and the date of the speech.

Class: Download_speech
Attributes:

- URL: URL of the speech we want to download.
- fields: vector with the fields we are interested in webscrapping. In my example: date, person who delivered it and body of the speech.

Methods:

- constructor: Method that takes as parameters the URL and the fields and creates an object of the class download_speech.
- download: Receives as a parameter the URL and the fields we are interested in each speech. It perfoms the webscrapping. It retuns a list with as many elements as the ones we are interested in webscrapping.

The follwoing class will be useful to plot the results of the analysis made in every speech.
Class: Plot_speech_info

Attributes:

- plot_type: There will be 5 plots given by default to the user.
- speeches: A list with all the speeches

Methods:

- constructor: Given a plot type and a speech, this method initializes a speech.

- join_info: Method to concatenate all the information of every speech. It will only concatenate the data frames that are result of the method metric of the class 'speech'.
- plot_speeches: Method that receives as a parameter an object of the class plot_speech_info and perform the specific plot. It has many features by default.

3 Problem 3

3.1 3a

To solve this problem, I will create a class called KillR in order to define a print method that allows me to just type 'k' and call the function quit. After this, I just have to create a new object and call it:

```
library(R6)

# Create the class KillR
killR <- R6Class("killR",
                 public = list(
                   initialize = function(){
                     library(fields)
                   },

                   print = function(){
                     q(save="yes")
                   }
                 )
                 )

# Create a new object of class KillR, I call it 'k'
k <- killR$new()

# If I execute this code, R will stop
k
```

3.2 3b

Yes, it will work. q() is a function (method) and my 'q' will be an instance of the class killR, that executes its own predefined print method. They belong to different classes. The class I
just created has a special print method to achieve the requested task in 3a). We could also have defined another function called q() that will coexist with the default q() given by R. I just have to invoke it as myclass::q(). This would be possible because they will be generic methods from different classes, that based on the type of object passed as argument, dispatch a class-specific function (method) that operates on the object.