Democratic VS Republican – What can we know from president inauguration speeches?

Han Lin (hl3006)

9/17/2017

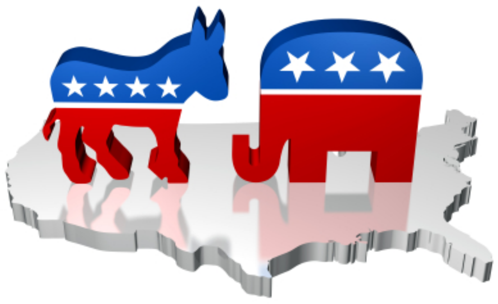


Figure 1:

As we all know, Democratic Party and Republican Party are two major contemporary political parties in the United States, the former is more progressive and the latter is more conservative. The two parties differ in their policies but also have similarities.

As American people vote to decide next president, they are basically choosing parties, in most cases, between Democratic and Republican. So one question we could ask is: what are the differences and similarities of the president inauguration speeches from different parties?

In this project, I will analyze inauguration speeches of different presidents from Democratic and Republican, trying to find differences and similarities. Here is some questions we want to find the answer:

- Will the presidents of the two parties differ in length of their speechs?
- Are there any differences of the length of sentence of speechs for two parties?
- What are the most frequent said words in the speech for two parties?
- How do typical words like 'freedom', 'people', 'government' show in the speeches?
- How does emotion change in the speeches and compared between two parties?

Step 0: Install and load libraries

```r
packages.used=c("rvest", "qdap",  "tidytext","plyr","syuzhet",
                "sentimentr", "gplots", "dplyr","tidyr","gridExtra",
                "tm","topicmodels","ggplot2","wordcloud","RColorBrewer")
# check packages that need to be installed.
packages.needed=setdiff(packages.used,
                        intersect(installed.packages()[,1],
                                  packages.used))
# install additional packages
```

```r
if(length(packages.needed)>0){
  install.packages(packages.needed, dependencies = TRUE)
}

lapply(packages.used, require, character.only = TRUE)

source("../lib/plotstacked.R")
source("../lib/speechFuncs.R")
```

This notebook was prepared with the following environmental settings.

```r
print(R.version)
```

```
##                 _
## platform       x86_64-apple-darwin13.4.0
## arch           x86_64
## os             darwin13.4.0
## system         x86_64, darwin13.4.0
## status
## major          3
## minor          3.3
## year           2017
## month          03
## day            06
## svn rev        72310
## language       R
## version.string R version 3.3.3 (2017-03-06)
## nickname       Another Canoe
```

Step 1: Import data

In this project, we need information about president inauguration speeches. We have basic information in 'InaugurationInfo.csv' and we scraped speech URLs from http://www.presidency.ucsb.edu/inaugurals.php.

```r
# basic information about inauguration speech
speeches <- read.csv("../data/InaugurationInfo.csv")

# inauguaral speeches
main.page <- read_html(x = "http://www.presidency.ucsb.edu/inaugurals.php")
# Get link URLs
# f.speechlinks is a function for extracting links from the list of speeches.
inaug=f.speechlinks(main.page)
#head(inaug)
#as.Date(inaug[,1], format="%B %e, %Y")
inaug=inaug[-nrow(inaug),] # remove the last line, irrelevant due to error.
speech.list=cbind(speeches, inaug)

# Add full text of speech to the dataframe
# Loop over each row in speech.list
speech.list$fulltext=NA
for(i in seq(nrow(speech.list))) {
  text <- read_html(speech.list$urls[i]) %>% # load the page
    html_nodes(".displaytext") %>% # isloate the text
    html_text() # get the text
  speech.list$fulltext[i]=text
  # Create the file name
  filename <- paste0("../data/InauguralSpeeches/",
                     speech.list$type[i],
                     speech.list$File[i], "-",
                     speech.list$Term[i], ".txt")
  sink(file = filename) %>% # open file to write
  cat(text)  # write the file
  sink() # close the file
}

# we only need data of democratic and republican
speech.list <- filter(speech.list,Party=='Democratic'|Party=='Republican')
speech.list$Party <- factor(speech.list$Party)
speech.list$File <- paste(speech.list$File,speech.list$Term)

# we also create a sorted speech list according to party
speech.list_sorted <- speech.list[order(speech.list$Party),]
rownames(speech.list_sorted) <- 1:nrow(speech.list_sorted)
speech.list$President <- as.character(speech.list$President)
```

Step 2: Data Processing — generate list of sentences and list of words

We will generate a list of sentences and a list of words for futher analysis, since sentence and word are unit for our analysis.

Generate list of sentences

We first generate a list of senteences, each row is a sentence of the speech for each president.

```
# We assign an sequential id to each sentence in a speech (`sent.id`) and also calculated the number of
sentence.list=NULL
for(i in 1:nrow(speech.list)){
  sentences=sent_detect(speech.list$fulltext[i],
                        endmarks = c("?", ".", "!", "|",";"))
  if(length(sentences)>0){
    emotions=get_nrc_sentiment(sentences)
    word.count=word_count(sentences)
    # colnames(emotions)=paste0("emo.", colnames(emotions))
    # in case the word counts are zeros?
    emotions=diag(1/(word.count+0.01))%*%as.matrix(emotions)
    sentence.list=rbind(sentence.list,
                        cbind(speech.list[i,-ncol(speech.list)],
                              sentences=as.character(sentences),
                              word.count,
                              emotions,
                              sent.id=1:length(sentences)
                              )
    )
  }
}

# Some non-sentences exist in raw data due to erroneous extra end-of sentence marks.
sentence.list=
  sentence.list%>%
  filter(!is.na(word.count))
```

Generate list of words

Now we have list of sentences with each row a sentence in the 'sentences' column. Now we want to convert it to one-token-per-document-per-row. So we need to break sentences into individual tokens, which could be done by *unnest_tokens* from tidytext. Below we then generate a list of words from sentence list above, each row is a word of a sentence for each president.

```
sentence.list$sentences <- as.character(sentence.list$sentences)
words.list <- sentence.list %>%
  unnest_tokens(word, sentences)
#data(stop_words)
words.list <- words.list %>%
  anti_join(stop_words)
```

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

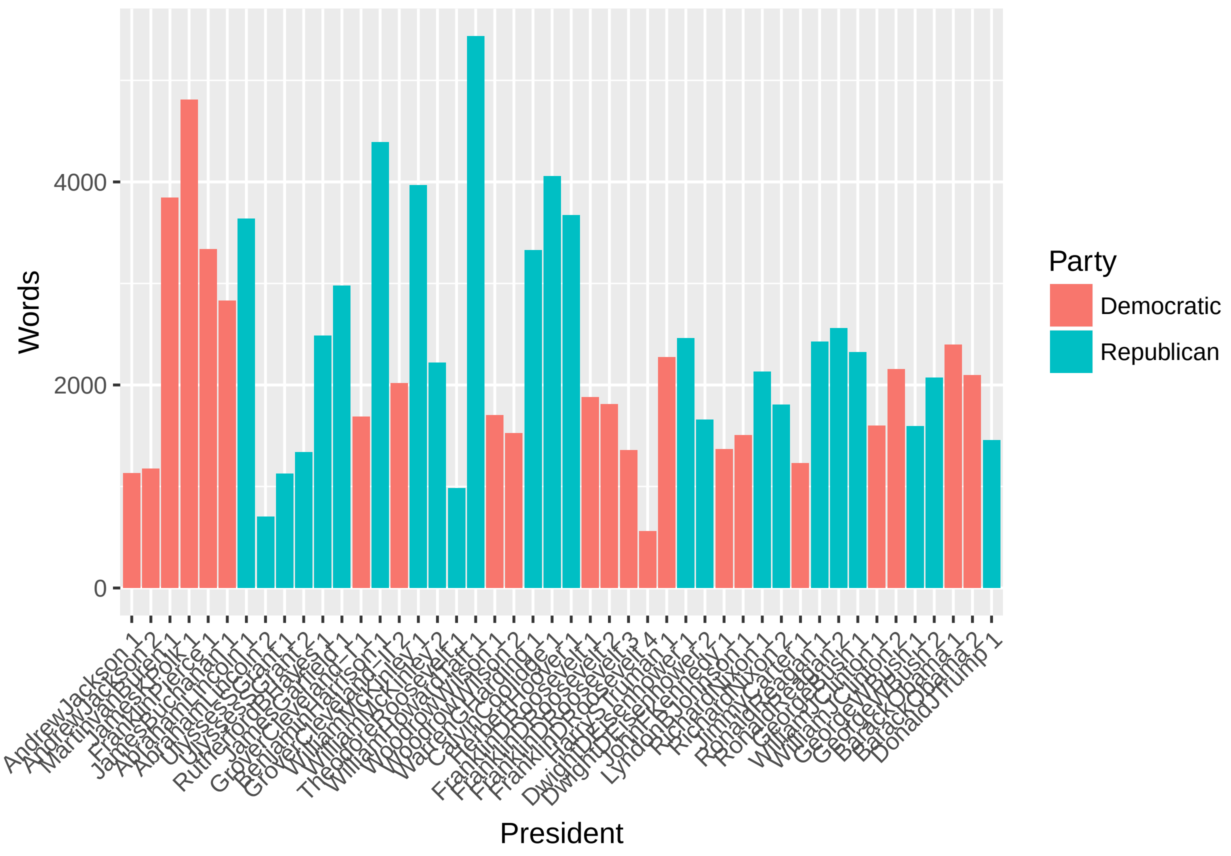
```
#words.list
words.list$File <- paste(words.list$File,words.list$Term,sep='')
words.list$links <- as.character(as.Date(words.list$links, format="%B %d, %Y"))
```

Step 3: Total words and length of sentences

Total words of speech

The first question we want to address is: Will the presidents of the two parties differ in length of their speechs? To answer to this question, we draw a bar plot showing the speech length for each president chronologically. And we use different colors to differ the two parties.

```
ggplot(speech.list,aes(x=reorder(File,1:nrow(speech.list)),y=Words,fill=Party))+
  geom_col()+
  theme(axis.text.x=element_text(angle=45, hjust=1))+
  xlab('President')
```



From this plot, we could see most speeches of republican are longer than those of democratic. We could also check this intuition by calculate mean total words for two parties.

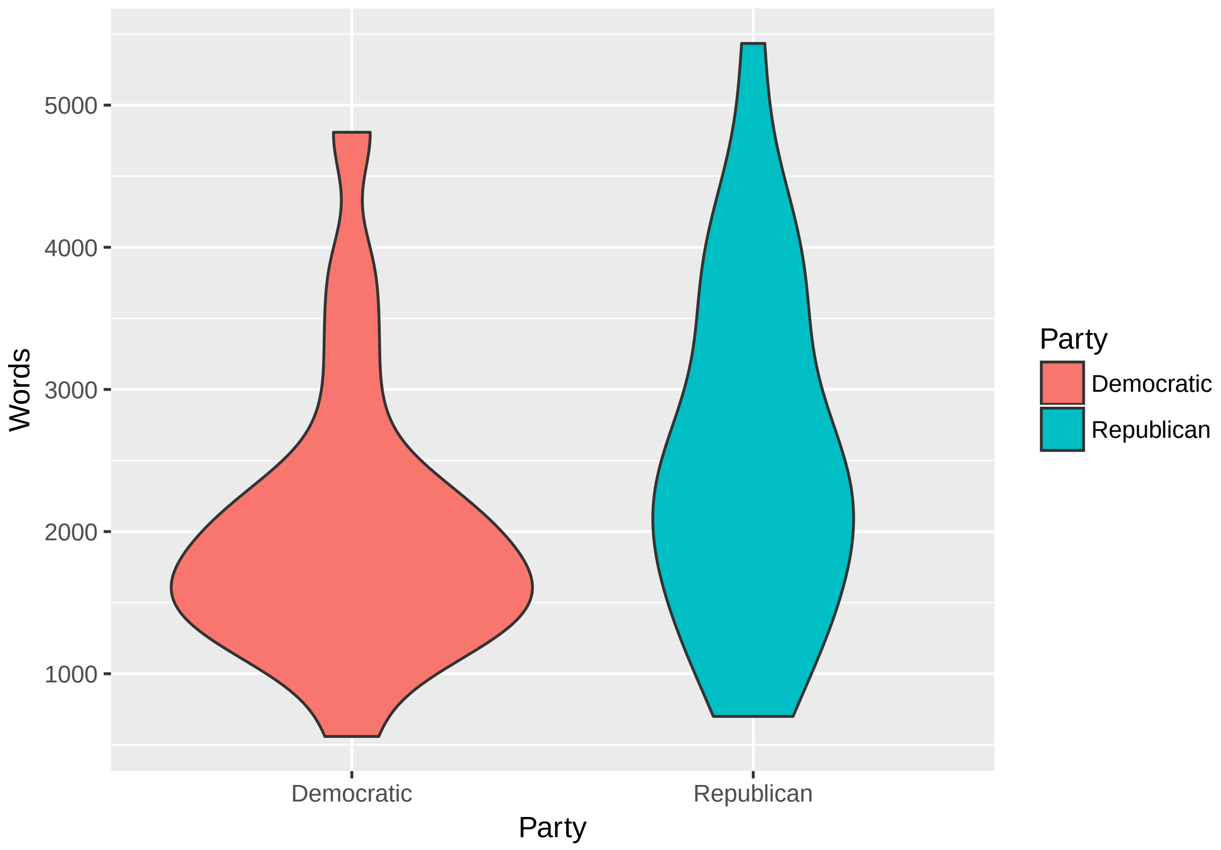
```r
tapply(speech.list$Words,speech.list$Party,mean)
```

```
## Democratic Republican
##   2012.682   2533.083
```

So mean total words of republican are 500 more than cemocratic.

We could also see the detailed distribution for each parties by violin plot.

```r
ggplot(speech.list, aes(factor(Party), Words,fill=Party)) +
  geom_violin()+
  xlab('Party')
```

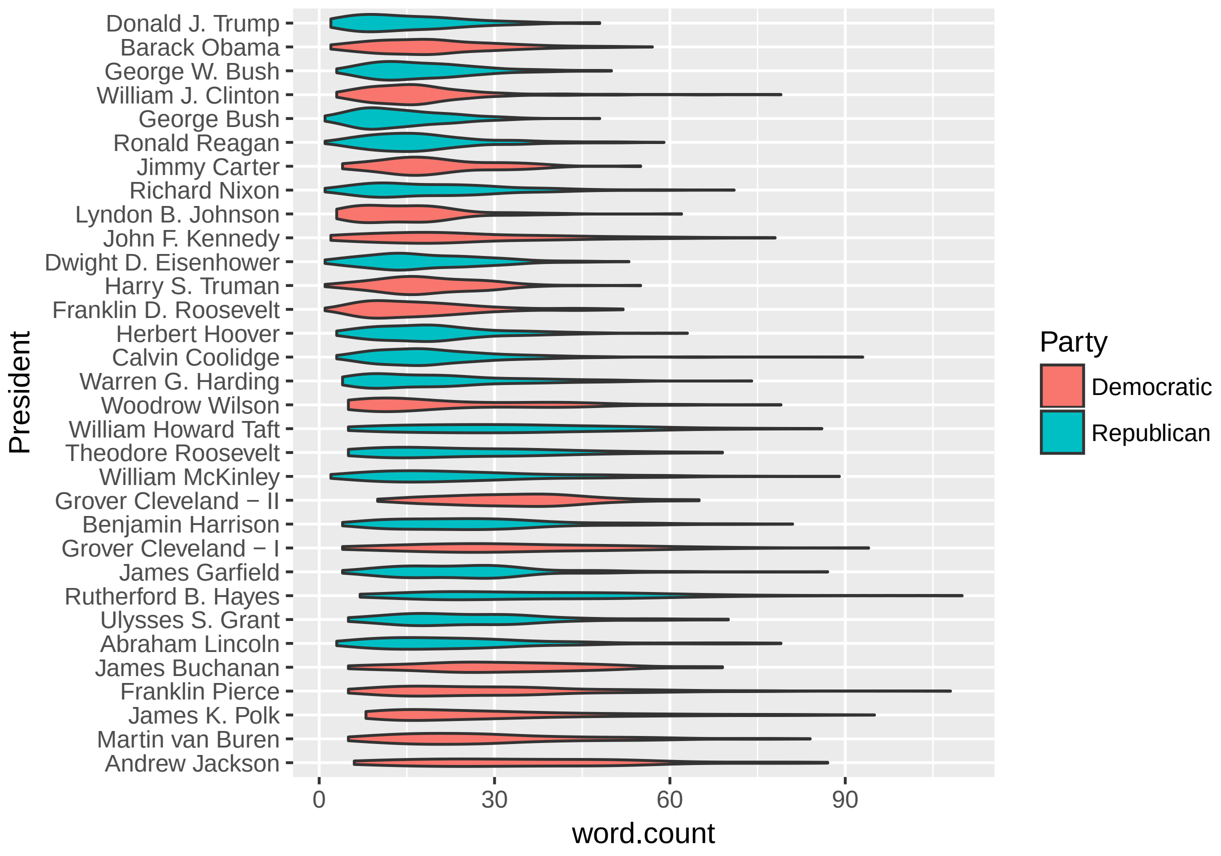


From the violin plot, we could see speech length of democratic is more concentrated, most around 1600 words. While speech length of republican is more dispersive, most ranging from 1000 to 3000 words.

Length of sentences

We also wonder if length of sentence differ between two parties. So we here draw a violin plot for each president, showing in the same plot. Again, we use different colors to differ two parties.

```
ggplot(sentence.list,aes(reorder(President,1:nrow(sentence.list)),word.count,fill=Party))+
  geom_violin()+
  xlab('President')+
  coord_flip()
```



We could also show the mean sentence length for each party.

```
tapply(sentence.list$word.count,sentence.list$Party,mean)
```

```
## Democratic Republican
##   23.11024   21.78651
```

We could see mean sentence length of republican is smaller than that of democratic.

Summarize from length of speech and length of sentence, president of democratic tend to use shorter speech but with longer sentences.

Step 4: Most frequent words

In this part, we will analyze most frequent words in the speeches for two parties. And we use word cloud to visualize the result.

Read in the speeches

Corpus is a collection of documents, there are many ways to create a corpus, here we use 'fulltext' column of the 'speech.list' dataframe.

```
corpus_demo <- Corpus(VectorSource(speech.list_sorted$fulltext[1:22]))
corpus_repub <- Corpus(VectorSource(speech.list_sorted$fulltext[23:46]))
```

Text processing

For the speeches, we remove extra white space, convert all letters to the lower case, remove stop words, removed empty words due to formatting errors, and remove punctuation.

After cleaning up corpus, we compute TDM. Documemnt Term Matrix (DTM) or Term Document Matrix (TDM) is a document that lists all occurrences of words of each document.

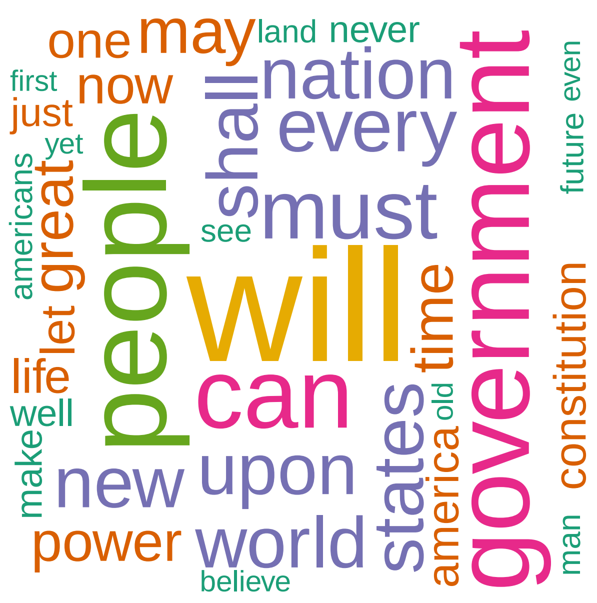
```r
# Corpus and TDM for democratic
corpus_demo<-tm_map(corpus_demo, stripWhitespace)
corpus_demo<-tm_map(corpus_demo, content_transformer(tolower))
corpus_demo<-tm_map(corpus_demo, removeWords, stopwords("english"))
corpus_demo<-tm_map(corpus_demo, removeWords, character(0))
corpus_demo<-tm_map(corpus_demo, removePunctuation)


tdm.all_demo<-TermDocumentMatrix(corpus_demo)
tdm.tidy_demo=tidy(tdm.all_demo)
tdm.overall_demo=summarise(group_by(tdm.tidy_demo, term), sum(count))


# Corpus and TDM for republican
corpus_repub<-tm_map(corpus_repub, stripWhitespace)
corpus_repub<-tm_map(corpus_repub, content_transformer(tolower))
corpus_repub<-tm_map(corpus_repub, removeWords, stopwords("english"))
corpus_repub<-tm_map(corpus_repub, removeWords, character(0))
corpus_repub<-tm_map(corpus_repub, removePunctuation)


tdm.all_repub<-TermDocumentMatrix(corpus_repub)
tdm.tidy_repub=tidy(tdm.all_repub)
tdm.overall_repub=summarise(group_by(tdm.tidy_repub, term), sum(count))
```

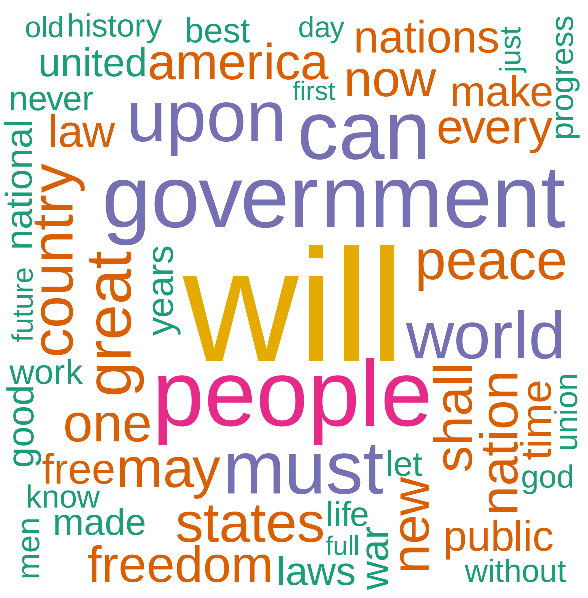
Word cloud

Next, we create word cloud to visualize the most frequent words for two parties.

```r
# set seed so that we get the same result every run
set.seed(42)

par(mfrow=c(1,2))

# word cloud for democratic
wordcloud(tdm.overall_demo$term, tdm.overall_demo$`sum(count)`,
          scale=c(5,0.5),
          max.words=100,
          min.freq=1,
          random.order=FALSE,
          rot.per=0.3,
          use.r.layout=T,
          random.color=FALSE,
          colors=brewer.pal(6,'Dark2'))

# word cloud for republican
```

```r
wordcloud(tdm.overall_repub$term, tdm.overall_repub$`sum(count)`,
          scale=c(5,0.5),
          max.words=100,
          min.freq=1,
          random.order=FALSE,
          rot.per=0.3,
          use.r.layout=T,
          random.color=FALSE,
          colors=brewer.pal(6,"Dark2"))
```



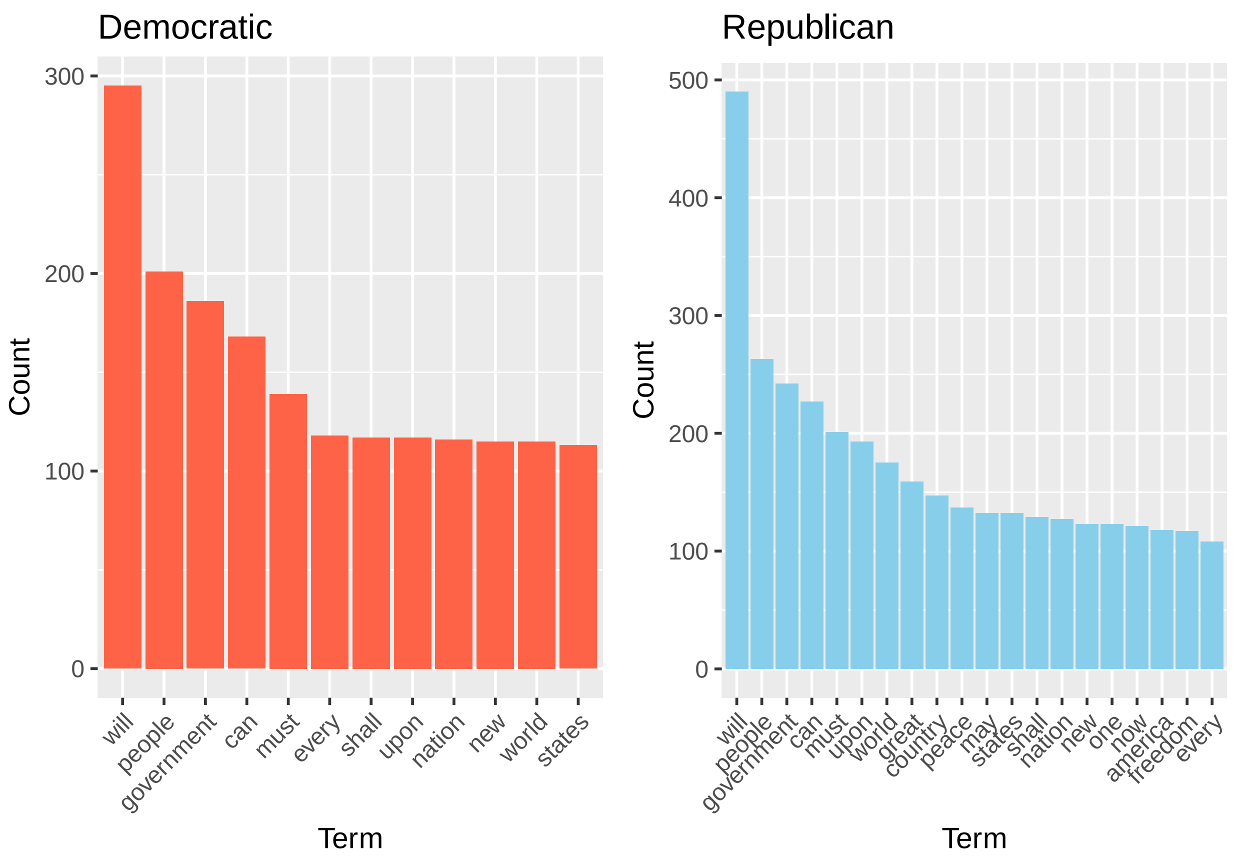


The left plot is word cloud for democratic, the right one is for republican.

It turns out that the most frequent words in the speeches for both parties are similar. For example, 'will', 'government', 'people' show most time, which is intuitively.

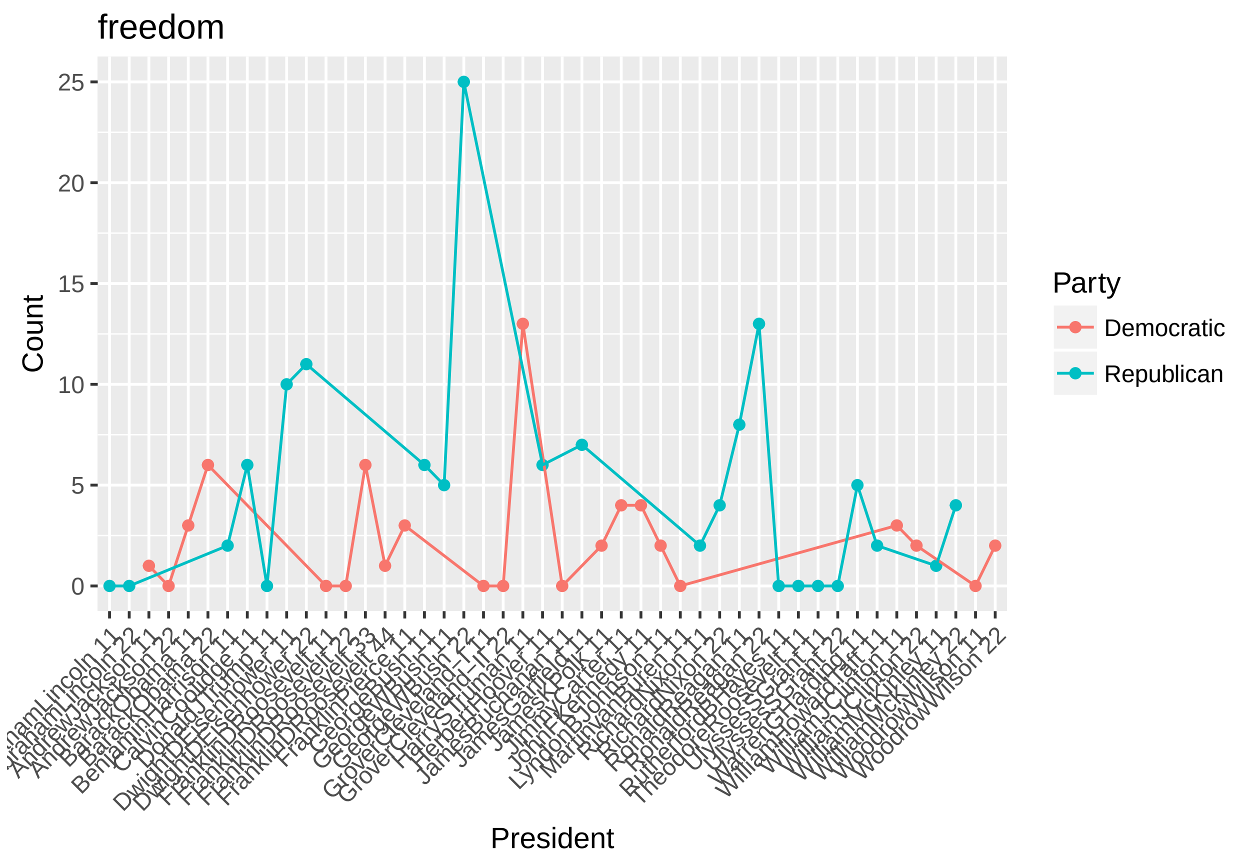
We could also show a barplot to show most frequent words.

```r
# bar plot for democratic
a <- ggplot(subset(tdm.overall_demo,`sum(count)`>100),aes(reorder(term,desc(`sum(count)`)),`sum(count)`
  geom_bar(stat='identity',fill='tomato')+
  theme(axis.text.x=element_text(angle=45, hjust=1))+
  labs(title='Democratic',x='Term',y='Count')

# bar plot for republican
b <- ggplot(subset(tdm.overall_repub,`sum(count)`>100),aes(reorder(term,desc(`sum(count)`)),`sum(count)
  geom_bar(stat='identity',fill='skyblue')+
  theme(axis.text.x=element_text(angle=45, hjust=1))+
  labs(title='Republican',x='Term',y='Count')

# make two plots side by side
grid.arrange(a, b, nrow=1, ncol=2)
```

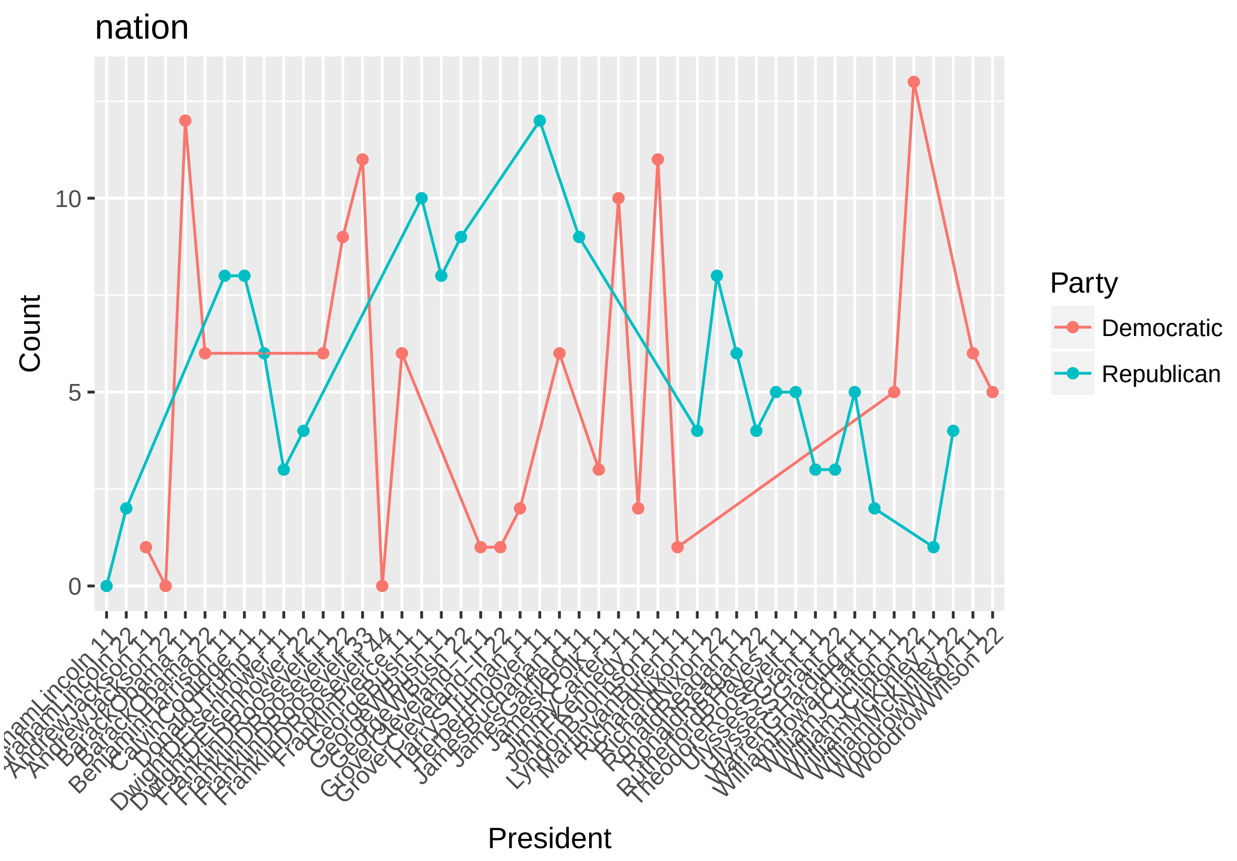


We also could compare appearance of some typical words for the two parties like 'freedom', 'nation', 'world', 'people', 'govenment'.

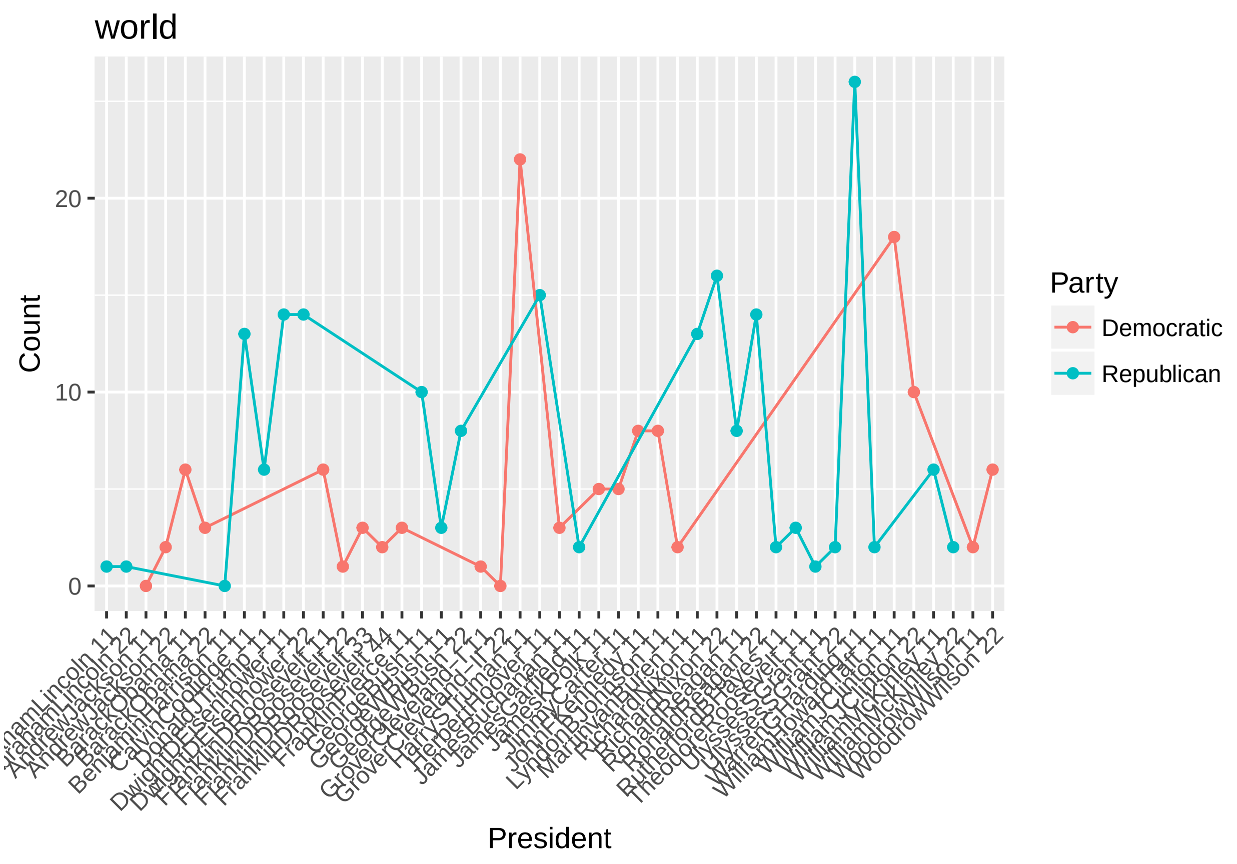
```r
word_trend <- function(word){
  temp <- ddply(words.list,.(File),function(x){
  return(c(unique(x[,'links']),sum(x[,'word']==word),unique(as.character(x[,'Party']))))}) %>%
  arrange(V1)
  colnames(temp) <- c('File','Date','Count','Party')
  temp$Count <- as.integer(temp$Count)

  ggplot(temp,aes(File,Count,group=Party,color=Party))+
    geom_point()+geom_line()+
    theme(axis.text.x=element_text(angle=45, hjust=1))+
    xlab('President')+
    ggtitle(word)

}


word_trend('freedom')
```



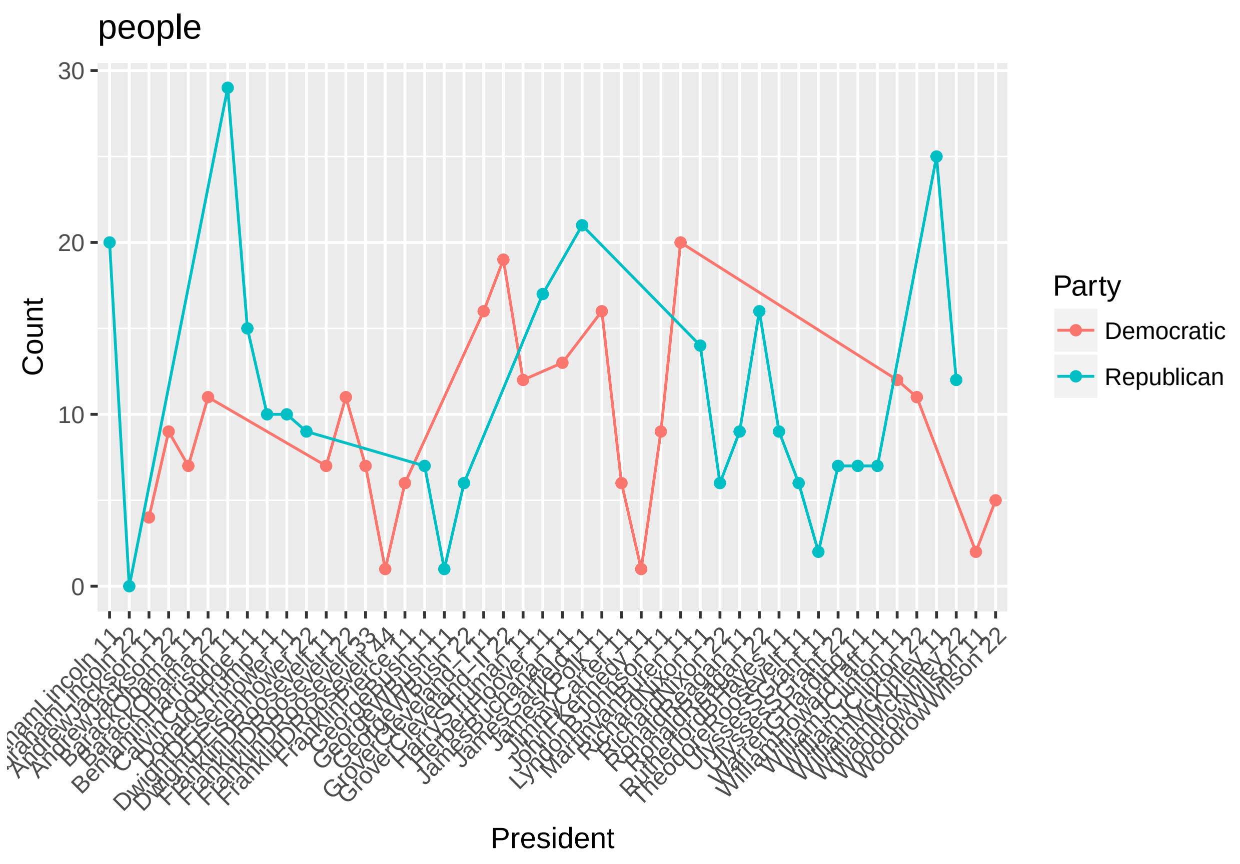
```
word_trend('nation')
```



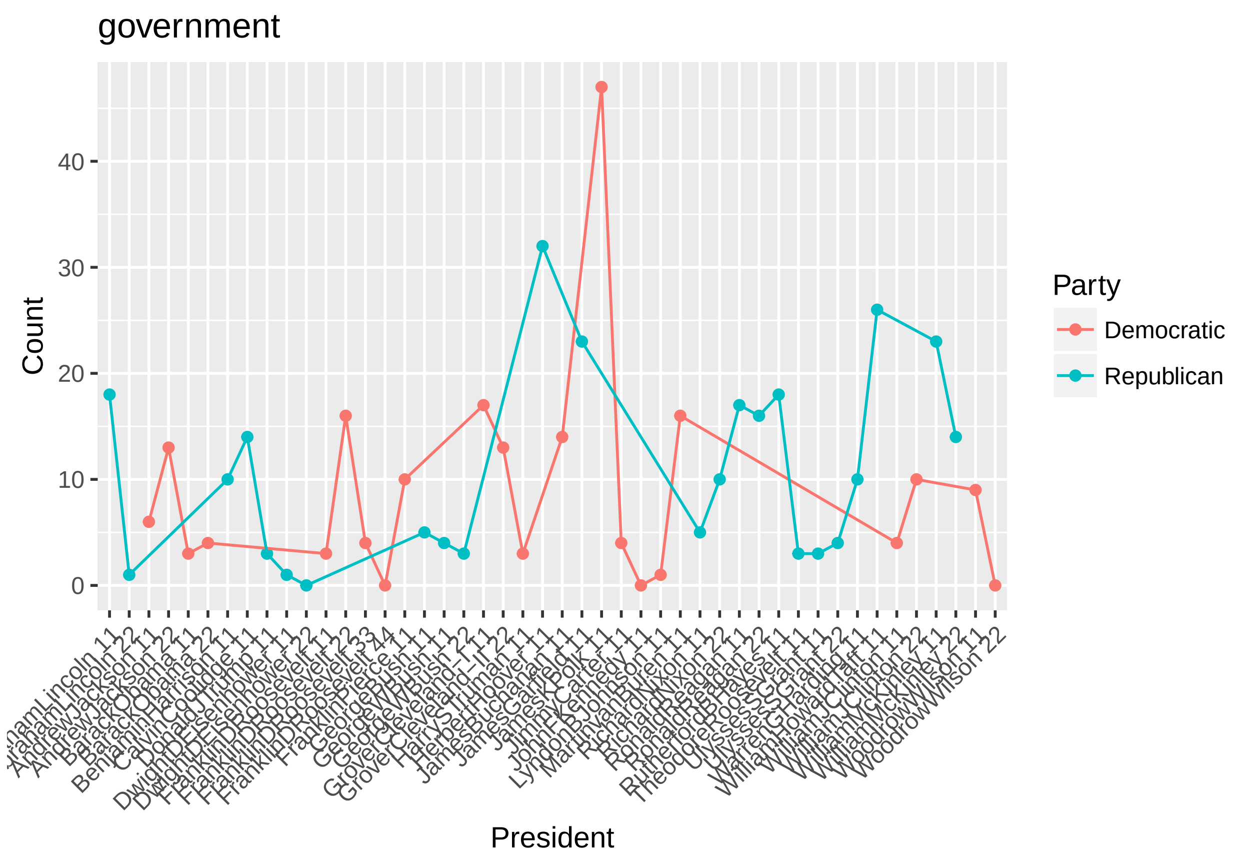
```
word_trend('world')
```



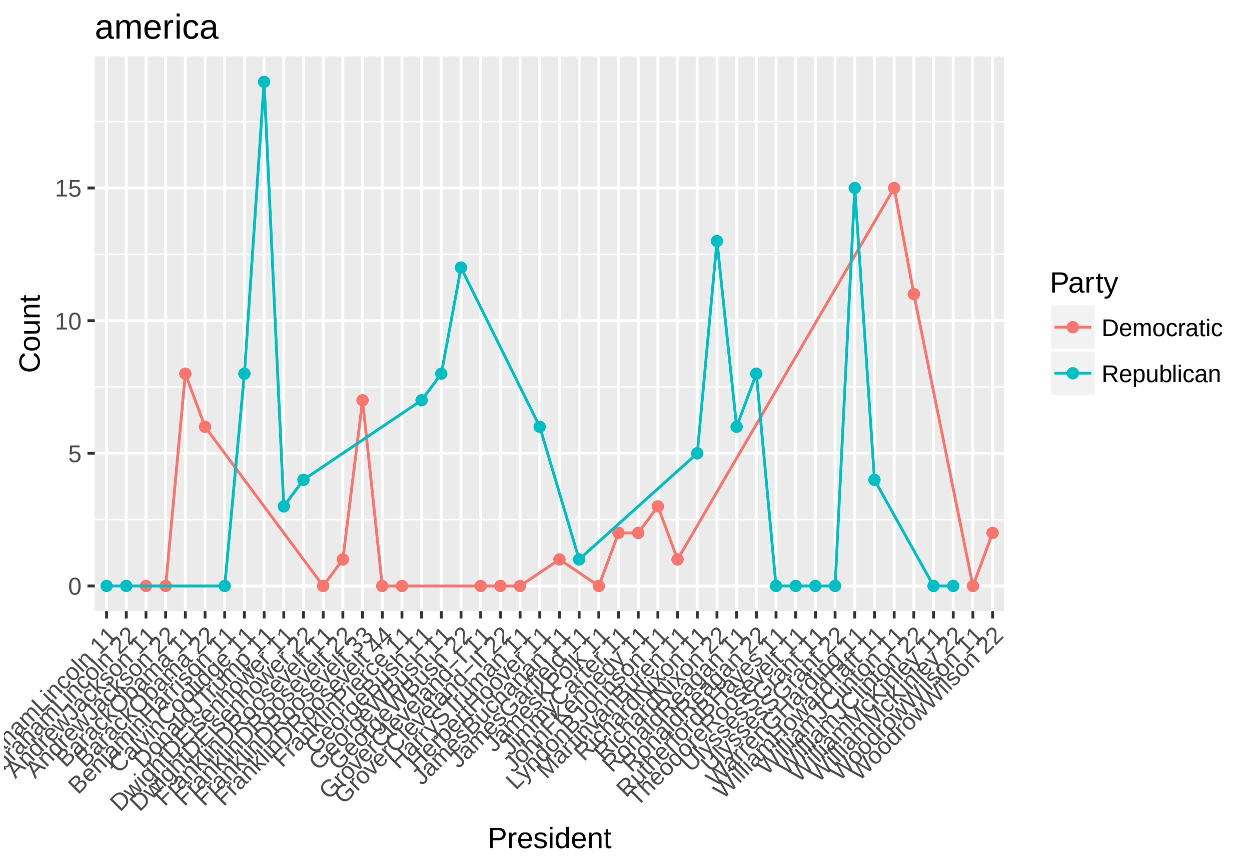
```
word_trend('people')
```



```r
word_trend('government')
```



```
word_trend('america')
```



We see from the plot that for most words, there is no much difference between two parties, but we could also find some interesting patterns like: Republican mentioned 'freedom' more than Democratic; Donald Trump mentioned 'America' more than any other president.

tf-idf

The idea of tf-idf is to find the important words for the content of each document by decreasing the weight for commonly used words and increasing the weight for words that are not used very much in a collection or corpus of documents.

We show below the top tf-idf score words in the speech.

```r
# tf-idf for democratic
dtm_demo <- DocumentTermMatrix(corpus_demo,
                          control = list(weighting = function(x)
                                            weightTfIdf(x,
                                                       normalize =FALSE),
                                    stopwords = TRUE))
tf_idf_demo <- tidy(dtm_demo)
print(head(arrange(tf_idf_demo,desc(count)),10))
```

```
## # A tibble: 10 x 3
##    document        term    count
##       <chr>       <chr>    <dbl>
## 1         4       union 41.26421
## 2         4       texas 34.59432
## 3         4     revenue 27.78755
## 4        20     century 25.79013
## 5        11        helped 24.21602
## 6        15       program 24.21602
## 7        16         sides 22.99575
## 8        15         aided 22.29716
## 9        17      covenant 20.75659
## 10        3 institutions 20.63211
```

```
# tf-idf for republican
dtm_repub <- DocumentTermMatrix(corpus_repub,
                          control = list(weighting = function(x)
                                                    weightTfIdf(x,
                                                                 normalize =FALSE),
                                                stopwords = TRUE))
tf_idf_repub <- tidy(dtm_repub)
print(head(arrange(tf_idf_repub,desc(count)),10))
```

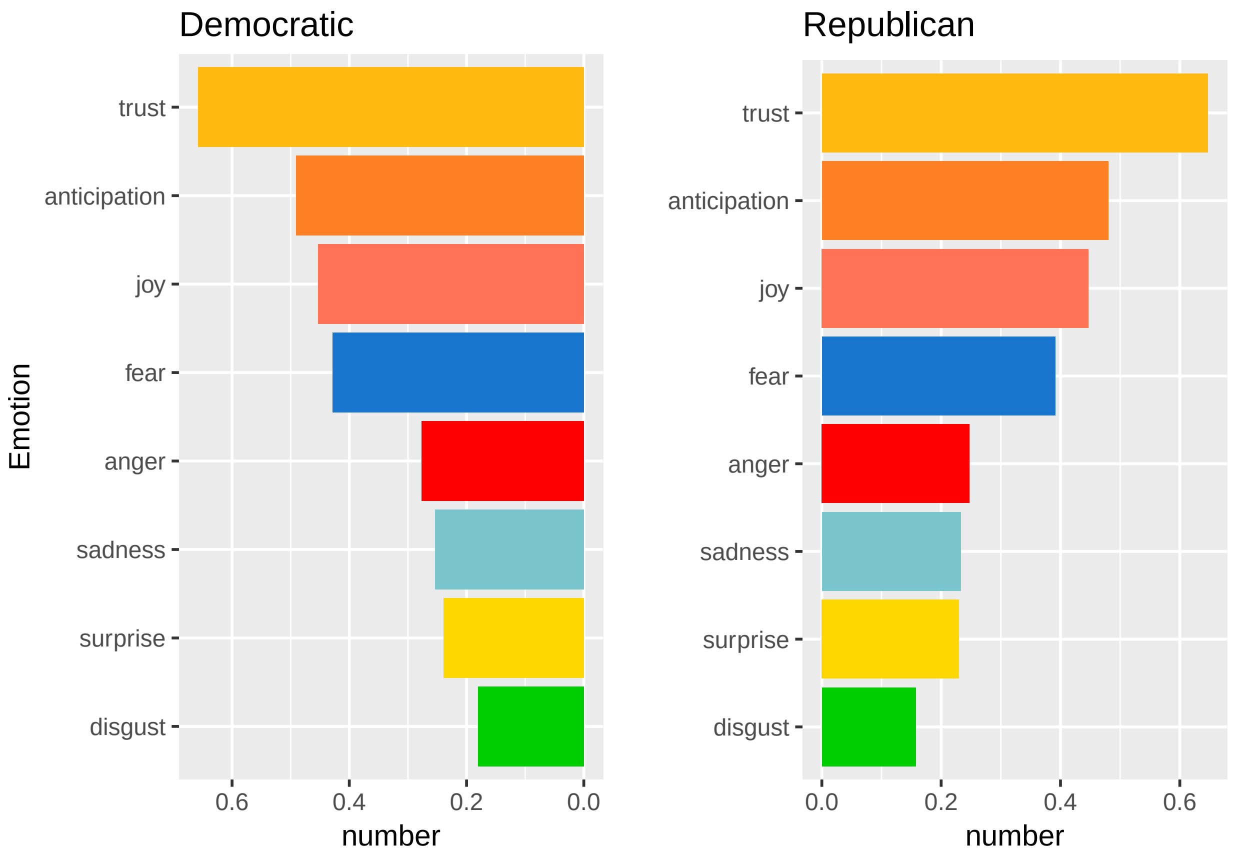
```
## # A tibble: 10 x 3
##    document       term    count
##       <chr>      <chr>    <dbl>
## 1        11 interstate 45.84963
## 2        11      negro 32.26466
## 3        11     tariff 29.41945
## 4        11       bill 28.67970
## 5        11   business 27.78676
## 6         8      loans 27.50978
## 7        11      coast 27.50978
## 8        11   suitable 27.50978
## 9        14       18th 27.50978
## 10        8    revenue 27.15641
```

Step 5: Sentiment Analysis

Emotion

There are a variety of methods and dictionaries that exist for evaluating the opinion or emotion in text. The tidytext package contains several sentiment lexicons in the sentiments dataset. There are *nrc*, *bing*, and *AFINN* lexicon, here we use NRC sentiment lexion., which categorizes words in a binary fashion ("yes"/"no") into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. More information could be found here.

```
# calculate means for each emotion
emo.means_demo=colMeans(select(filter(sentence.list,Party=='Democratic'), anger:trust)>0.01)
emo.means_repub=colMeans(select(filter(sentence.list,Party=='Republican'), anger:trust)>0.01)

# use barplot to show emotion for the two parties
color_use <- c('darkgoldenrod1','chocolate1','coral1','dodgerblue3','red','cadetblue3','gold','green3')

c <- ggplot(data.frame(emotion=names(emo.means_demo),number=emo.means_demo),aes(reorder(emotion,number)
  geom_bar(stat='identity',fill=rev(color_use))+
  xlab('Emotion')+
  ggtitle('Democratic')+
  scale_y_reverse()+
  coord_flip()+
  theme_grey()


d <- ggplot(data.frame(emotion=names(emo.means_repub),number=emo.means_repub),aes(reorder(emotion,numbe
  geom_bar(stat='identity',fill=rev(color_use))+
  xlab('')+
  ggtitle('Republican')+
  coord_flip()+
  theme_grey()



grid.arrange(c, d, nrow=1, ncol=2)
```

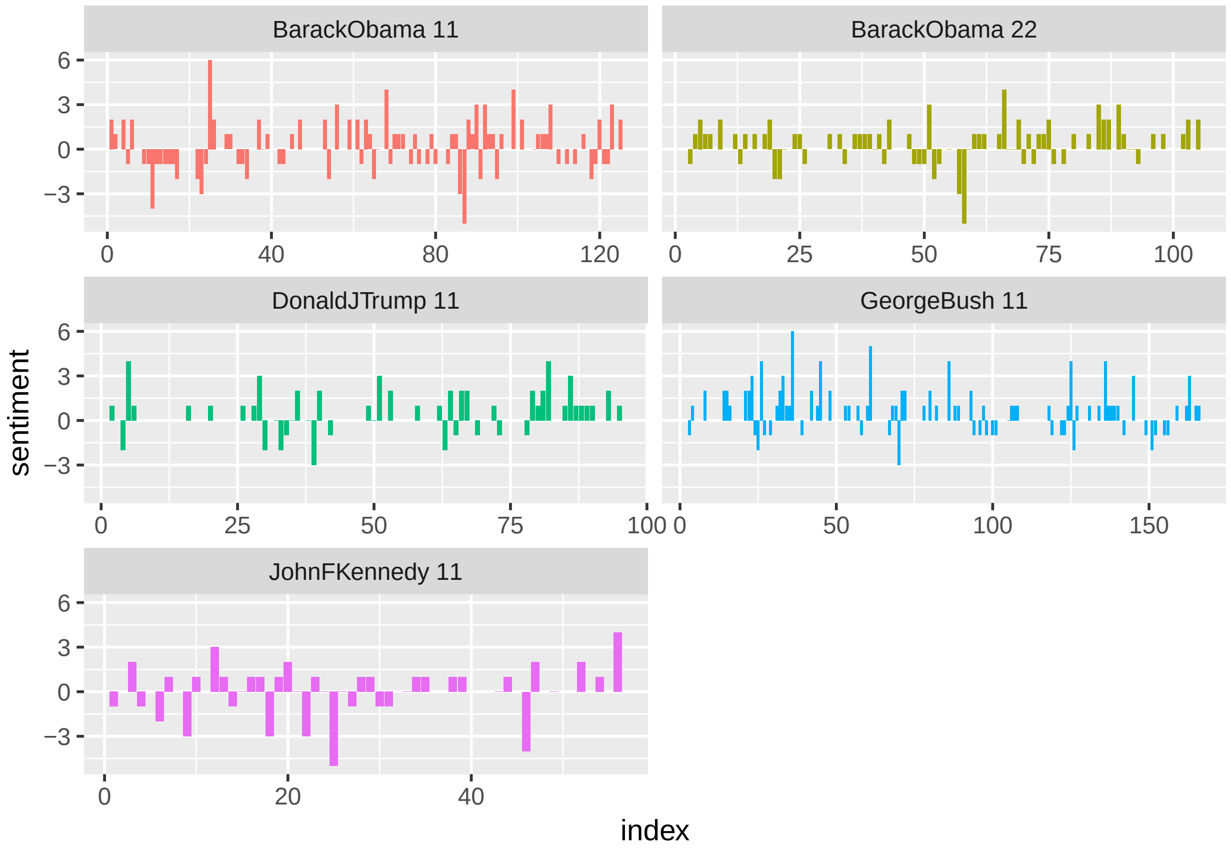


We could see similarities of the two parties. Top emotions are both ‘trust’, ‘anticipation’, and ‘joy’, which are all positive emotions. This is intuitive because inauguration speech is meant to inspire people.

Positive & Negative

We could also see the positive and negative sentiment change throughout speechs of each president.

```
# here we select four presidents to discuss
select_president <- c('John F. Kennedy', 'Barack Obama','George Bush', 'Donald J. Trump')
sentiment <- filter(words.list,President%in%select_president) %>%
  inner_join(get_sentiments("bing")) %>%
  count(File, index = sent.id, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)

ggplot(sentiment, aes(index, sentiment, fill = File)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~File, ncol = 2, scales = "free_x")+
  theme_gray()
```



Barack Obama and John F. Kennedy represent Democratic, while Donald J. Trump and George Bush represent Republican. We could see that Democratic seem to have a negative part after the beginning of the speech, while Republican seem to have positive and negative part throughout the speech.