GR5243 Project1

Qihang Li, ql2276

September 27, 2017

Step 0: Environmental settings and preparations

In this project, we are doing some researches about speeches of U.S. presidents and candidates according to their campaign slogans identified by analysis. First, we shall prepare this notebook with the following environmental settings.

```
# list all needed packeges
packages.need=c("rvest", "tibble", "qdap", "sentimentr", "gplots", "dplyr", "tm",
                "syuzhet", "factoextra", "beeswarm", "scales", "RColorBrewer",
                "RANN", "tm", "topicmodels", "tidytext", "wordcloud", "lubridate")

# load packages
for (i in 1:length(packages.need))
{
  if(sum(packages.need[i]==installed.packages()[, 1])<1)
    install.packages(packages.need[i], dependencies=T)
  library(packages.need[i], character.only=T)
}
rm(i)

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

print(R.version)
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          3
## minor          4.1
## year           2017
## month          06
## day            30
## svn rev        72865
## language       R
## version.string R version 3.4.1 (2017-06-30)
## nickname       Single Candle
```

Step 1: Read the speeches and campaign slogans

1.1. Data harvest: scrap speech URLs from http://www.presidency.ucsb.edu/.

Here, we will select all inaugural addresses of past presidents, nomination speeches of major party candidates and farewell addresses from The American Presidency Project. We don't use the farewell speeches since they are not highly related to the campaign slogans.

```
# Inauguaral speeches
main.page=read_html("http://www.presidency.ucsb.edu/inaugurals.php")
inaug=f.speechlinks(main.page)
inaug=inaug[-nrow(inaug), ] # remove the last line, irrelevant due to error.

# Nomination speeches
main.page=read_html("http://www.presidency.ucsb.edu/nomination.php")
nomin=f.speechlinks(main.page)
```

1.2. Read in the speech metadata from saved file

Here, we prepared CSV data sets for the speeches we will scrap.

```
inaug.list=read.csv("./data/inauglist.csv", stringsAsFactors=F)
inaug.list$Date=as.Date(ISOdate(inaug.list$Year,
                                inaug.list$Month,
                                inaug.list$Day))
inaug.list=inaug.list[, -(5:7)]

nomin.list=read.csv("./data/nominlist.csv", stringsAsFactors=F)
nomin.list$Date=as.Date(ISOdate(nomin.list$Year,
                                nomin.list$Month,
                                nomin.list$Day))
nomin.list=nomin.list[, -(5:7)]
```

We assemble all scrapped speeches into one list. Note here that we don't have the full text yet, only the links to full text transcripts.

1.3. scrap the texts of speeches from the speech URLs.

```
speech.list=rbind(inaug.list, nomin.list)
speech.list$type=c(rep("inaug", nrow(inaug.list)),
                   rep("nomin", nrow(nomin.list)))
speech.url=rbind(inaug, nomin)
speech.list=cbind(speech.list, speech.url)
```

Based on the list of speeches, we scrap the main text part of the transcript's html page. For simple html pages of this kind, Selectorgadget is very convenient for identifying the html node that `rvest` can use to scrap its content. For reproducibility, we also save our scrapped speeches into our local folder as individual speech files.

```
# 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$fulltext[i]=text
  # Create the file name
  filename=paste0("./output/", speech.list$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
}
# Order by time
speech.list=speech.list[order(speech.list$Date), ]
speech.list=cbind(Document=seq(nrow(speech.list)), speech.list)
rownames(speech.list)=seq(nrow(speech.list))
```

1.4. Read in the slogans from saved file

Here, we prepared CSV data sets for the speeches we will scrap. The data is from Wikipedia. Some of the slogans were not included, since their candidates' speeches are not found. Then we merge the data together. Note that presidents or candidates before William Henry Harrison will not be studied since they don't have campaign slogans.

```
slogan=read.csv("./data/slogan.csv", stringsAsFactors=F, header=T)
speech.list$slogan1=NA
speech.list$slogan2=NA
speech.list$slogan3=NA
speech.list$slogan4=NA
speech.list$slogan5=NA
speech.list$slogan6=NA
for(i in seq(nrow(speech.list)))
{
  for(j in seq(nrow(slogan)))
  {
    if(speech.list$President[i]==slogan$President[j]) # Match presidents
    {
      if(year(speech.list$Date[i])-slogan$Year[j]<=1
         &year(speech.list$Date[i])-slogan$Year[j]>=0) # Match terms
      {
        speech.list$slogan1[i]=slogan$Slogan1[j]
        speech.list$slogan2[i]=slogan$Slogan2[j]
        speech.list$slogan3[i]=slogan$Slogan3[j]
        speech.list$slogan4[i]=slogan$Slogan4[j]
        speech.list$slogan5[i]=slogan$Slogan5[j]
        speech.list$slogan6[i]=slogan$Slogan6[j]
      }
    }
  }
}
```

Step 2: Text processing

2.1. Speeches

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. Then we compute the Document-Term Matrix (DTM). Also, we make an overall wordcloud for intuition.

```
speech.all=Corpus(VectorSource(speech.list$fulltext))

speech.all=tm_map(speech.all, stripWhitespace)
speech.all=tm_map(speech.all, content_transformer(tolower))
speech.all=tm_map(speech.all, removeWords, stopwords("english"))
```

```
speech.all=tm_map(speech.all, removeWords, character(0))
speech.all=tm_map(speech.all, removePunctuation)

speech.tdm.all=TermDocumentMatrix(speech.all)
speech.tdm.tidy=tidy(speech.tdm.all)
speech.tdm.overall=summarise(group_by(speech.tdm.tidy, term), sum(count))

wordcloud(speech.tdm.overall$term, speech.tdm.overall$`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(9, "Greys"))
```

We can see that words “people”, “government”, “america” and etc. are the most frequent notional words, and the word “will” takes up the greatest percentage.

Now we compute TF-IDF weighted document-term matrices for individual speeches. As we would like to identify interesting words for each inaugural speech, we use TF-IDF to weigh each term within each speech.

It highlights terms that are more specific for a particular speech. We use Trump's inauguration as examples.

```
speech.dtm=DocumentTermMatrix(speech.all,
                              control=list(weighting=function(x)
                                                weightTfIdf(x, normalize=F),
                                                stopwords=T))
speech.dtm=tidy(speech.dtm)
```

2.2. Slogans

We do the same with slogans.

```
speech.list$slogan.all=NA
for(i in seq(nrow(speech.list)))
{
  n=sum(is.na(speech.list[i,13:18]))
  if(n<6)
    speech.list$slogan.all[i]=paste(speech.list[i, 13:(18-n)],collapse=" ")
}
slogan.all=Corpus(VectorSource(speech.list$slogan.all))

slogan.all=tm_map(slogan.all, stripWhitespace)
slogan.all=tm_map(slogan.all, content_transformer(tolower))
slogan.all=tm_map(slogan.all, removeWords, stopwords("english"))
slogan.all=tm_map(slogan.all, removeWords, character(0))
slogan.all=tm_map(slogan.all, removePunctuation)

slogan.tdm.all=TermDocumentMatrix(slogan.all)

slogan.tdm.tidy=tidy(slogan.tdm.all)
slogan.tdm.overall=summarise(group_by(slogan.tdm.tidy, term), sum(count))

wordcloud(slogan.tdm.overall$term, slogan.tdm.overall$`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(9, "Greys"))
```


```
slogan.dtm=DocumentTermMatrix(slogan.all,
                              control=list(weighting=function(x)
                                weightTfIdf(x, normalize=F),
                                stopwords=T))
slogan.dtm=tidy(slogan.dtm)
```

We can see that words "change", "peace", "people" and etc. are the most frequent notional words. This time, the word "america" takes up the greatest percentage.

Step 3: Overview of speech and slogan for a certian campaign

3.1 A function for showing relation with speech and slogan

We have already separate the data for every single speech. Here we will use a function to show the wordcloud for speeches and slogans according to the number of speech. Also, the ratio of slogan word to frequent speech words will be given.

```
Slogan.VS.Speech=function(Document, set.frequrncy=100, wordcloud=T, summary=T)
{
  # Get speech words
  speech.term=speech.dtm$term[speech.dtm$document==Document]
  speech.count=speech.dtm$count[speech.dtm$document==Document]
  # Check whether there is slogan
  if(!is.na(speech.list$slogan.all[Document]))
  {
    # Get slogan words
    slogan.term=slogan.dtm$term[slogan.dtm$document==Document]
```

```
  slogan.count=slogan.dtm$count[slogan.dtm$document==Document]
  # Make wordcloud
  if(wordcloud)
  {
    par(mfrow=c(1, 2))
    wordcloud(speech.term, speech.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(9, "Blues"))
   wordcloud(slogan.term, slogan.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(9, "Reds"))
  }
  # Generate frequent speech words
  frequent.term=speech.term[order(speech.count, decreasing=T)]
  frequent.term=frequent.term[1:set.frequrncy]
  # Compute ratio
  ratio=100*sum(is.element(el=slogan.term,set=frequent.term))/length(slogan.term)
  ratio=round(ratio,4)
  # Output
  if(summary)
    print(paste(ratio, "% of the slogan words appears in ",
                ifelse(speech.list$type[Document]=="inaug", "President", "Candidate"),
                " ", speech.list$President[Document], "\'s ",
                ifelse(speech.list$type[Document]=="inaug", "inauguration", "nomination"),
                " speech for the term ", speech.list$Term[Document], ".", sep=""))
  output=data.frame(President=speech.list$President[Document],
           Term=speech.list$Term[Document],
           Year=year(speech.list$Date[Document]),
           Type=ifelse(speech.list$type[Document]=="inaug", "inauguration", "nomination"),
           Ratio=ratio/100)
  return(output)
}
else
{
  if(wordcloud)
  {
    par(mfrow=c(1, 1))
    wordcloud(speech.term, speech.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(9, "Blues"))
  }
  if(summary)
    print(paste("There wasn\'t any slogan for ",
                ifelse(speech.list$type[Document]=="inaug", "President", "Candidate"),
                " ", speech.list$President[Document], "\'s ",
                ifelse(speech.list$type[Document]=="inaug", "inauguration", "nomination"),
                " speech for the term ", speech.list$Term[Document], ".", sep=""))
  output=data.frame(President=speech.list$President[Document],
           Term=speech.list$Term[Document],
           Year=year(speech.list$Date[Document]),
```

```
            Type=ifelse(speech.list$type[Document]=="inaug", "inauguration", "nomination"),
            Ratio=NA)
    return(output)
  }
  print(slogan.term)
}
```

3.2. Some examples and an overview

Some examples are taken: President Washington's inauguration of his 1st term; President Lincoln's inauguration of his 2nd term; candidate Franklin D. Roosevelt's nomination speech of his 4th term; President Trump's inauguration of his 2nd term. After that, we shall study all the ratios except for those who did not have a campaign slogan.

```
Slogan.VS.Speech(1)
```


```
## [1] "There wasn't any slogan for President George Washington's inauguration speech for the term 1."
```

```
##           President Term Year         Type Ratio
## 1 George Washington    1 1789 inauguration    NA
```

```
Slogan.VS.Speech(21)
```


```
## [1] "14.2857% of the slogan words appears in President Abraham Lincoln's inauguration speech for the
```

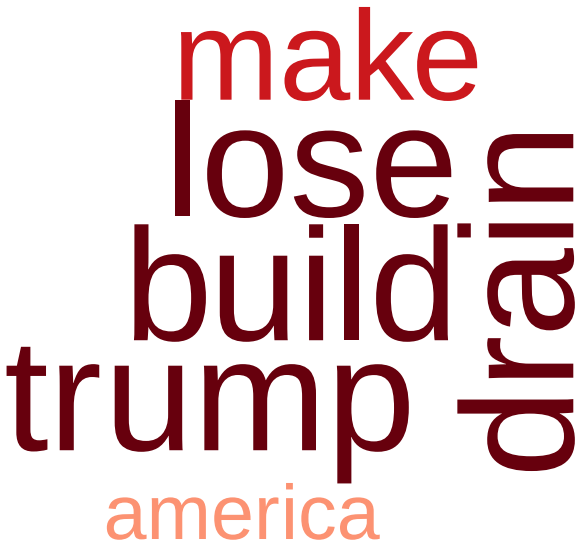
```
##         President Term Year         Type    Ratio
## 1 Abraham Lincoln    2 1865 inauguration 0.142857
```

```
Slogan.VS.Speech(58)
```


```
## [1] "0% of the slogan words appears in Candidate Franklin D. Roosevelt's nomination speech for the t
```

```
##               President Term Year       Type Ratio
## 1 Franklin D. Roosevelt    4 1944 nomination     0
```

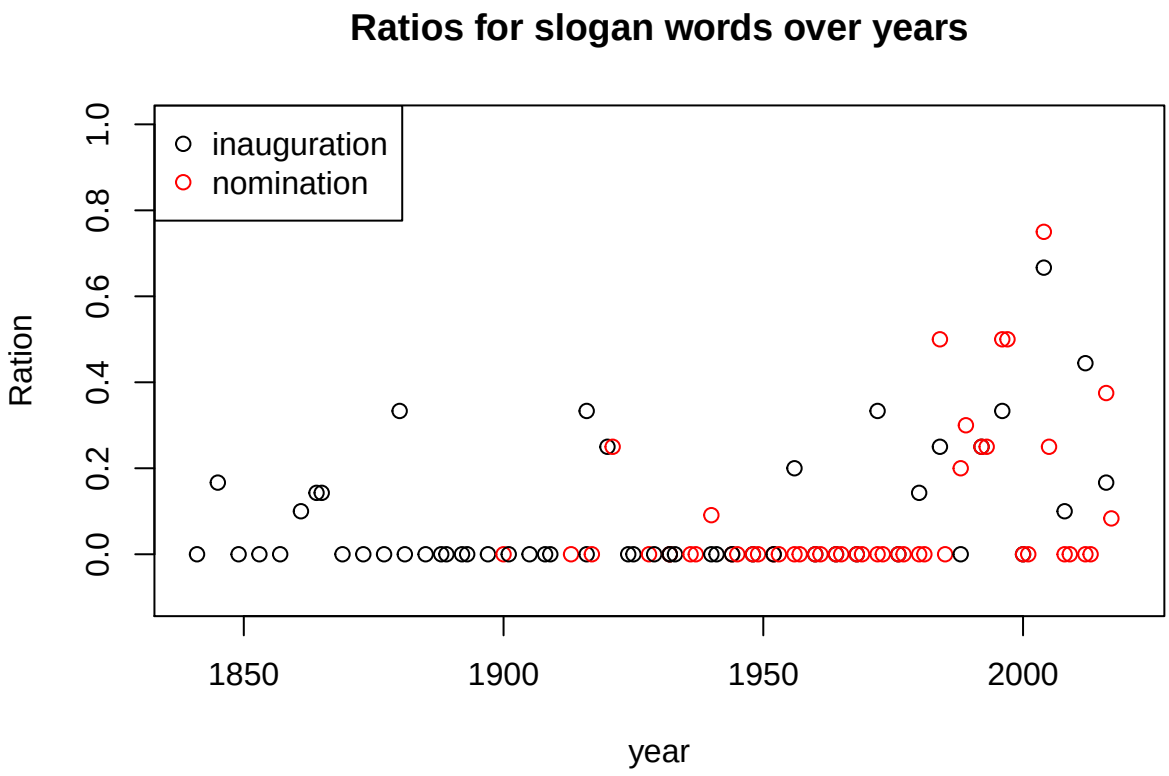
```
Slogan.VS.Speech(111)
```




```
## [1] "16.6667% of the slogan words appears in Candidate Donald J. Trump's nomination speech for the t
```

```
##         President Term Year       Type    Ratio
## 1 Donald J. Trump    1 2016 nomination 0.166667
```

```
Slogan.VS.Speech.out=NULL
for(i in seq(nrow(speech.list)))
  Slogan.VS.Speech.out=rbind(Slogan.VS.Speech.out,
                             Slogan.VS.Speech(i, wordcloud=F, summary=F))

par(mfrow=c(1, 1))
plot(Slogan.VS.Speech.out$Year[14:113],
     Slogan.VS.Speech.out$Ratio[14:113],
     xlab="year", ylab="Ration", xlim=c(1840,2020), ylim=c(-0.1,1.0),
     col=as.factor(Slogan.VS.Speech.out$Type),
     main="Ratios for slogan words over years")
legend(x="topleft", legend=c("inauguration", "nomination"),
       col=c(1,2), pch=1)
```



We can see that many of the ratios are 0, just like Roosevelt's nomination speech. Why? It's easy to realise that often, a slogan should be short, powerful, exciting and esay to remember. Sometimes politicians even use offensive or humorous slogans in order to get popular. This is different stody with official speeches, which should be long, clear and informative.

Note another thing that ratios to nomination speeches are relatively high. This is due to the fact that campaign slogans are used BEFORE a president is elected. We could imagine that candidates would focus more on election within the nomination speech, which is more related with their slogans; on the other hand, a president will address more about his plan and idea in the inauguration speech, which is less related to the campaign slogan.

Step 4: Data processing: generate list of sentences

We will use sentences as units of analysis for this project, as sentences are natural languge units for organizing thoughts and ideas. For each extracted sentence, we apply sentiment analysis using NRC sentiment lexion. "The NRC Emotion Lexicon is a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). The annotations were manually done by crowdsourcing."

We assign an sequential id to each sentence in a speech (`sent.id`) and also calculated the number of words in each sentence as *sentence length* (`word.count`). Some non-sentences exist in raw data due to erroneous extra end-of sentence marks.

```
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)
    emotions=diag(1/(word.count+0.01))%*%as.matrix(emotions)
    sentence.list=rbind(sentence.list,
                        cbind(speech.list[i, -(9:18)],
                              sentences=as.character(sentences),
                              word.count, emotions,
                              sent.id=1:length(sentences)))
  }
}
sentence.list=sentence.list%>%filter(!is.na(word.count))
```

Again, we do the same for slogans.

```
slogan.list=NULL
for(i in seq(nrow(speech.list)))
{
  n=sum(is.na(speech.list[i, 13:18]))
  if(n<6)
  {
    sentences=paste(speech.list[i, 13:(18-n)])
    emotions=get_nrc_sentiment(sentences)
    word.count=word_count(sentences)
    emotions=diag(1/(word.count+0.01),nrow=nrow(emotions))%*%as.matrix(emotions)
    slogan.list=rbind(slogan.list,
                      cbind(speech.list[i, -(9:18)],
                            sentences=as.character(sentences),
                            word.count, emotions,
                            sent.id=1:length(sentences)))
  }
  else
  {
    sentences=""
    emotions=get_nrc_sentiment(sentences)
    word.count=NA
    slogan.list=rbind(slogan.list,
                      cbind(speech.list[i, -(9:18)],
                            sentences=as.character(sentences),
                            word.count, emotions,
                            sent.id=1:length(sentences)))
  }
}
slogan.list=slogan.list%>%filter(!is.na(word.count))
rownames(slogan.list)=seq(nrow(slogan.list))
```

Step 5: Overview of sentence length distribution

First, we use a function to plot the length of sentences from speeches and slogans.

```
Sentence.Length=function(Document.No)
{
  sentence.list.sel=filter(sentence.list, Document==Document.No)
```

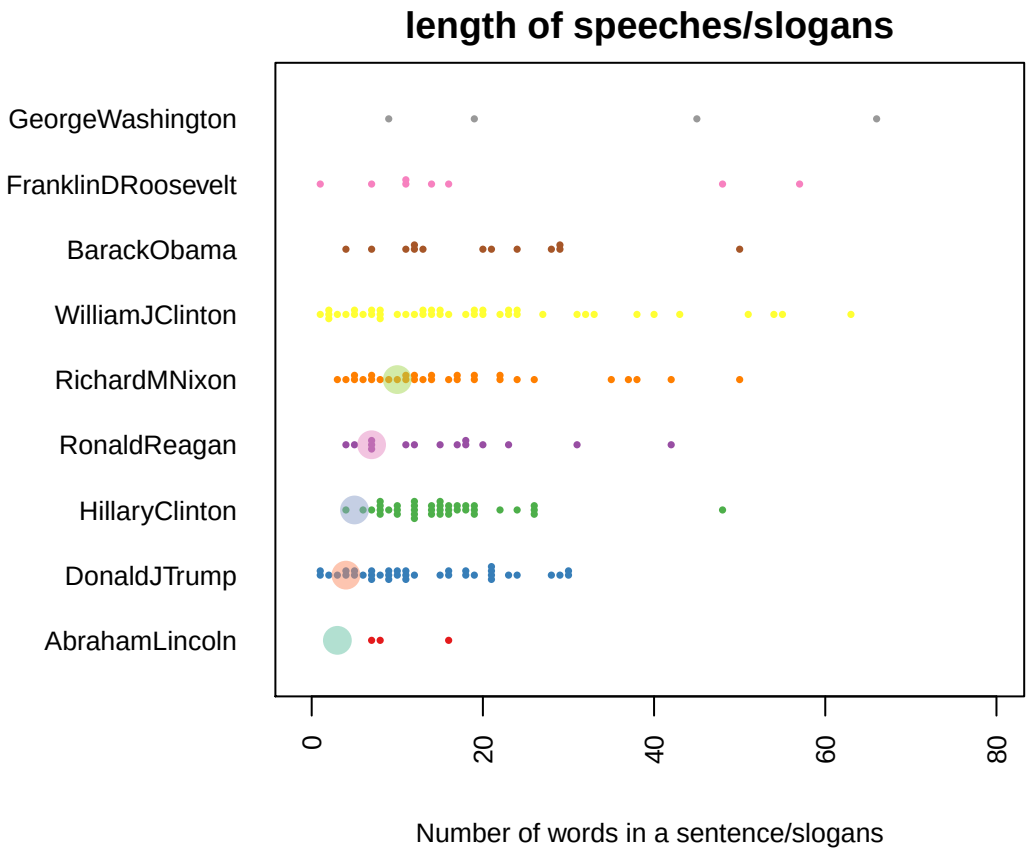
```
  sentence.list.sel$File=factor(sentence.list.sel$File)
  sentence.list.sel$FileOrdered=reorder(sentence.list.sel$File,
                                        sentence.list.sel$word.count,
                                        mean, order=T)
  slogan.list.sel=filter(slogan.list, Document==Document.No)
  slogan.list.sel$File=factor(slogan.list.sel$File)
  slogan.list.sel$FileOrdered=reorder(slogan.list.sel$File,
                                      slogan.list.sel$word.count,
                                      mean, order=T)
  # Make plot
  Title=paste("length of speeches/slogans")
  par(mar=c(4, 11, 2, 2))
  beeswarm(word.count~FileOrdered, data=sentence.list.sel,
         horizontal=T, pch=16,  xlim=c(-1,80),
         cex=0.5, cex.axis=0.8, cex.lab=0.8,
         col=alpha(brewer.pal(9, "Set1")), las=2,
         spacing=5/nlevels(sentence.list.sel$FileOrdered),
         xlab="Number of words in a sentence/slogans",
         ylab="", main=Title)
  if(length(slogan.list.sel$Document))
    beeswarm(word.count~FileOrdered, data=slogan.list.sel,
             horizontal=T, pch=16, cex=2, cex.axis=0.8, cex.lab=0.8,
             col=alpha(brewer.pal(9, "Set2"), 0.5), add=T,
             spacing=2/nlevels(sentence.list.sel$FileOrdered))
}
```

Once again, we use these examples: President Washington's inauguration of his 1st term; President Lincoln's inauguration of his 2nd term; candidate Franklin D. Roosevelt's nomination speech of his 4th term; President Trump's inauguration of his 2nd term. After that, we shall study all the ratios except for those who did not have a campaign slogan.

Since we have 9 colors, we shall add the below for reference:112 Hillary Clinton's nomination speech ,107 Barack Obama's inauguration speech of his 1st term; 97 William J. Clinton's nomination speech for his 2nd term; 86 Ronald Reagan's inauguration speech of his 1st term; 75 Richard M. Nixon's nomination speech of his 1st term.

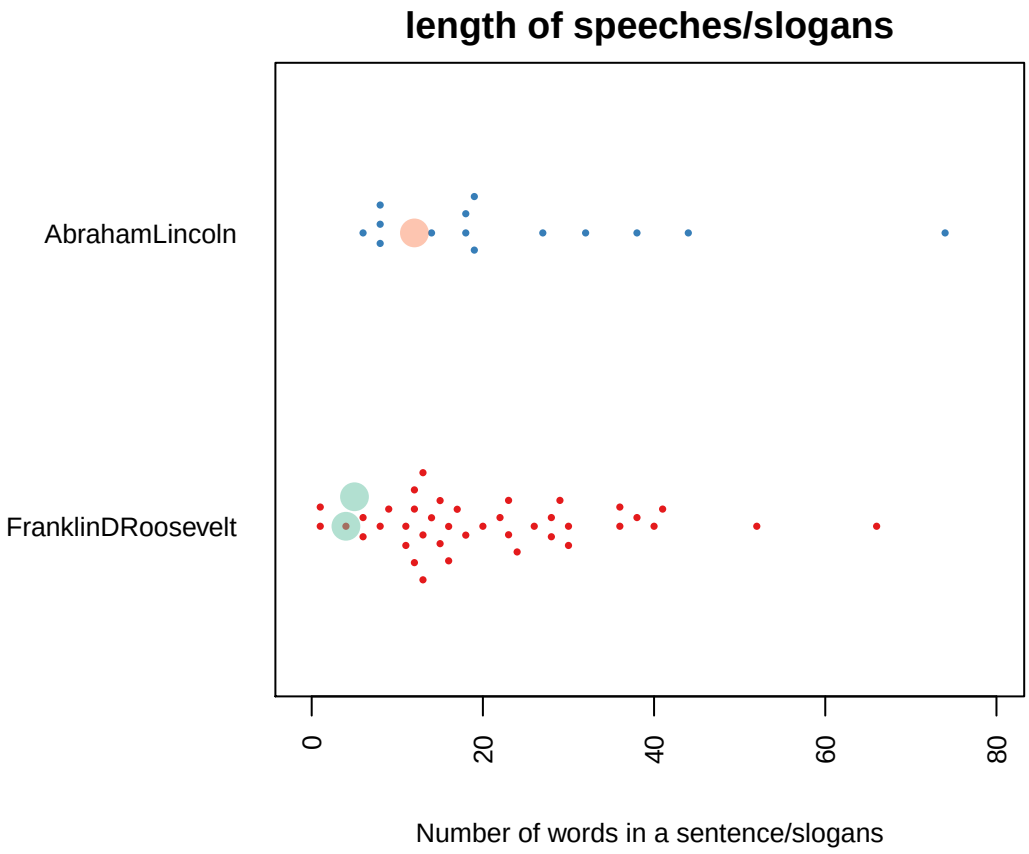
```
Sentence.Length(c(1, 21, 58, 75, 86, 97, 107, 111, 112))
```

```
## Warning in Document == Document.No: longer object length is not a multiple
## of shorter object length

## Warning in Document == Document.No: longer object length is not a multiple
## of shorter object length

## Warning in brewer.pal(9, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```



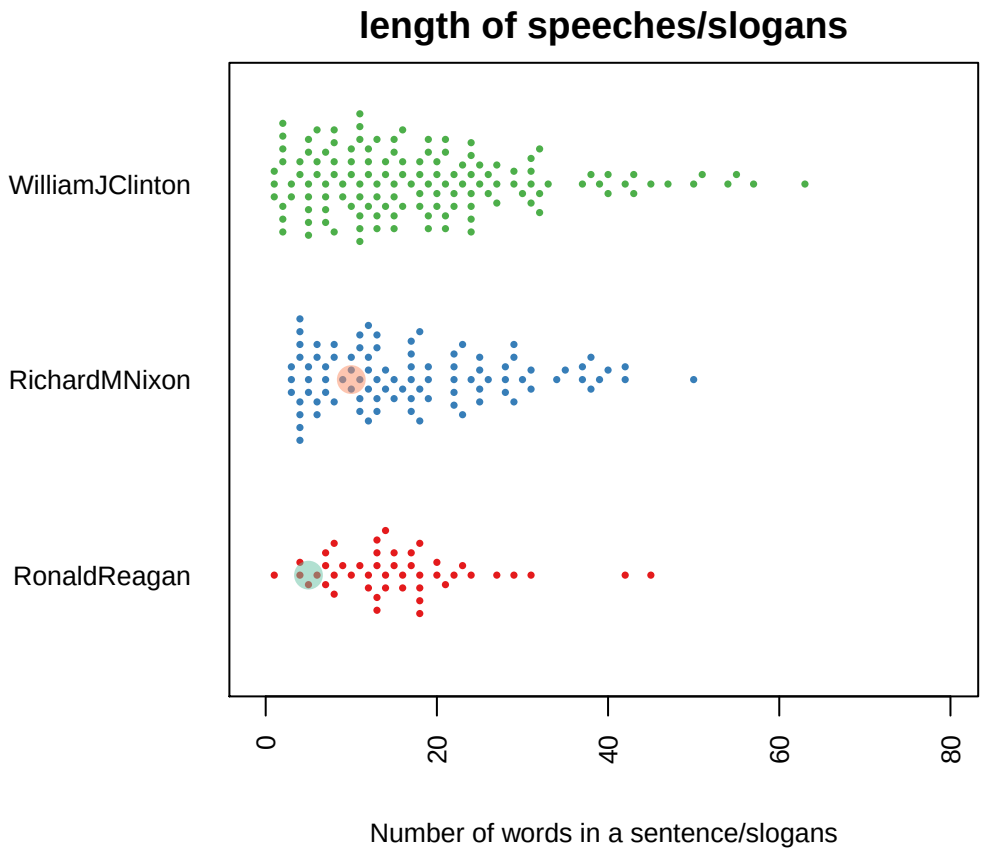

```
Sentence.Length(c(21, 58))
```

```
## Warning in Document == Document.No: longer object length is not a multiple of shorter object length

## Warning in Document == Document.No: n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```




```
Sentence.Length(c(75, 86, 97))
```

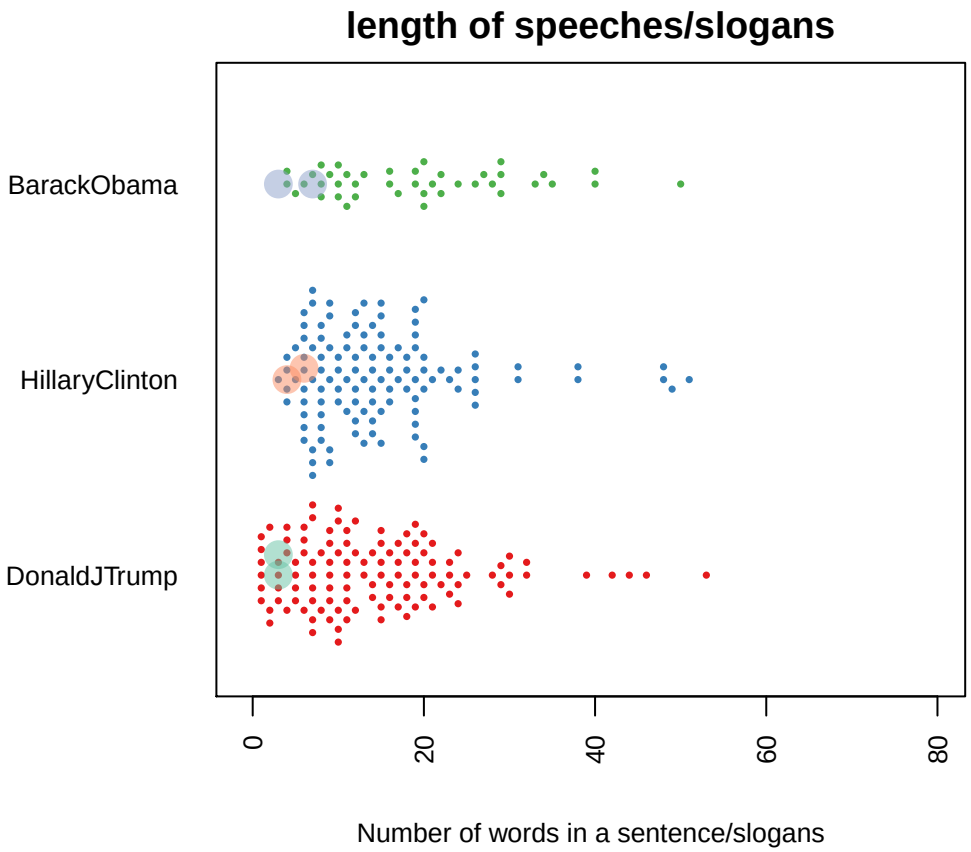
```
## Warning in Document == Document.No: longer object length is not a multiple of shorter object length

## Warning in Document == Document.No: n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```




```
Sentence.Length(c(107, 111, 112))
```

```
## Warning in Document == Document.No: longer object length is not a multiple of shorter object length
```

```
## Warning in Document == Document.No: n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```



The big dots are length of slogans and small ones speeches. We can see that on average, slogans are shorter that speech sentences. Some interesting facts: Lincoln was famous for his fewer but longer sentences, while Trump loves to speak many short sentences. As time moves on, politicians get more interesed in shorter sentences on average.