Project1: How the public take the inaugural speeches.

Yi Xiang[yx2365@columbia.edu]

January 30, 2017

Step0: Install and load libraries

```
ptm <- proc.time()
packages.used=c("tm", "wordcloud", "RColorBrewer",
                "dplyr", "tidytext","ggplot2","SnowballC","qdap"
                ,"data.table","scales","MASS")


# check packages that need to be installed.
packages.needed=setdiff(packages.used,
                        intersect(installed.packages()[,1],
                                  packages.used))
# install additional packages
if(length(packages.needed)>0){
  install.packages(packages.needed, dependencies = TRUE,
                   repos='http://cran.us.r-project.org')
}

library(tm)
library(wordcloud)
library(RColorBrewer)
library(dplyr)
library(tidytext)
library(ggplot2)
library(SnowballC)
library(qdap)
library(sentimentr)
library(data.table)
library(scales)
library(MASS)
```

Step1: Read Speeachs and save in Corpus

please change the working directory before runing the code, current working directory is under my local drive.

```
# setwd("C:/Users/Elaine/Documents/Columbia/Spring_2017/AppliedDataScience/Spr2017-Proj1-e
eelaine")
folder.path<- "../data/InauguralSpeeches/"
speeches<-list.files(path=folder.path,pattern = "*.txt")
prex.out<-substr(speeches,6,stop = nchar(speeches)-4)

ff.all<-Corpus(DirSource(folder.path))
```

Step2: Clean the data using text mining tools

```
CleanCorpus<-function(mycorpus){
mycorpus<-tm_map(mycorpus, stripWhitespace)#strip unnecessary white space
mycorpus<-tm_map(mycorpus, content_transformer(tolower))#convert to lowercases
mycorpus<-tm_map(mycorpus,removeWords,stopwords("en"))#remove english stopwords
mycorpus<-tm_map(mycorpus, removeWords, character(0))
mycorpus<-tm_map(mycorpus,removePunctuation)#remove punctuations
mycorpus<-tm_map(mycorpus,stemDocument)#remove common word endings
}

ff.all<-CleanCorpus(ff.all)
```

Step3: Calculate the TF-IDF(Term Frequency-Inverse Document frequency) weighted matrix for speeches

TF-IDF weights the terms while eliminating the influence of the most commonly used words in all the documents. In other words, term frequency is adjusted by the inverse document frequency, which take the common word in all the documents into account. The more common a word is in all documents, the smaller idf is, and tf-idf=tf*idf, is smaller.

```
dtm.all<-DocumentTermMatrix(ff.all,control=list(weighting =
                                              function(x)
                                              weightTfIdf(x, normalize =
                                                          FALSE),
                                              stopwords = TRUE))


ff.dtm<-tidy(dtm.all)
ff.matrix<-as.matrix(dtm.all)
```

Step4: Find the most frequently used words

```
## words after sparse unfamiliar words
dtms <- removeSparseTerms(dtm.all, 0.15)##remove sparse terms
freq_removedsparse <- colSums(as.matrix(dtms))
freq_removedsparse
```

```
##      can  countri    everi     good   govern    great     hope     just
## 35.93538 27.20055 55.48114 33.83172 67.32005 28.50312 31.91557 25.97779
##      may     must   nation      now      one    peopl    power    right
## 43.57028 67.72781 16.93641 35.60433 55.03007 31.13503 58.29027 28.45380
##    shall     time    world
## 66.37869 34.20592 43.44022
```

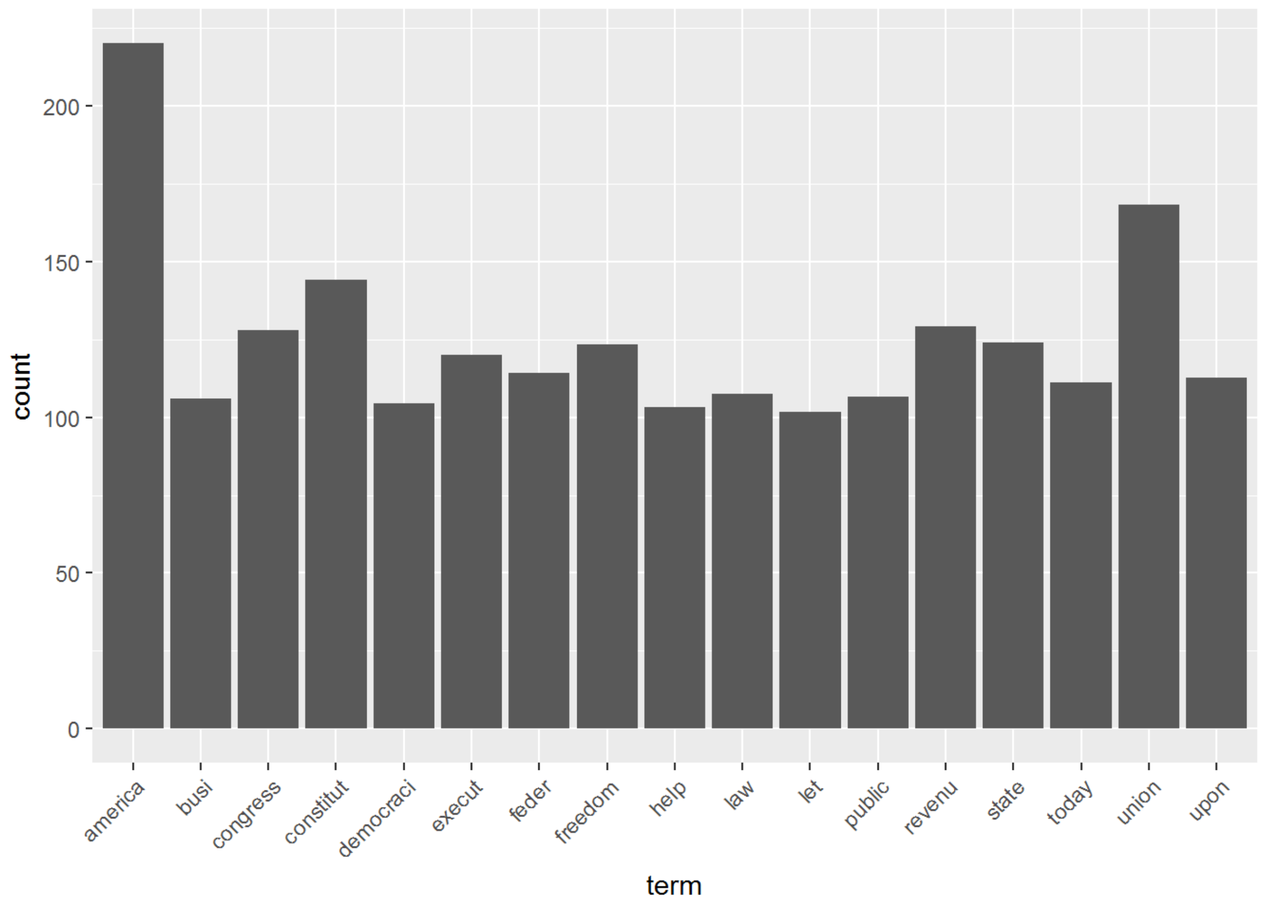
```
##15 most frequently used words
freq_all <- sort(colSums(as.matrix(dtm.all)), decreasing=TRUE)
head(freq_all, 15)
```

```
##   america      union constitut    revenu  congress     state   freedom
##  220.0000   168.1040  144.1235  129.2478  128.0000  123.9504  123.2202
##    execut      feder      upon     today       law    public      busi
##  120.1173   114.3218  112.5585  111.1240  107.6083  106.6370  105.9877
## democraci
##  104.4606
```

```
##words that has been used over 100 times
freq_overhundred<-findFreqTerms(dtm.all,lowfreq = 100)
freq_overhundred
```

```
##  [1] "america"   "busi"      "congress"  "constitut" "democraci"
##  [6] "execut"    "feder"     "freedom"   "help"      "law"
## [11] "let"       "public"    "revenu"    "state"     "today"
## [16] "union"     "upon"
```

```
##plot the terms that has frequency over 100 times
ff.dataframe<-data.frame(term=names(freq_all),count=freq_all)
p <- ggplot(subset(ff.dataframe, count>100), aes(term, count))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
p
```



```
png(filename = "../output/AllSpeechesFrequentTerm.png",width = 300, height = 300)
p <- ggplot(subset(ff.dataframe, count>100), aes(term, count))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
p
dev.off()
```

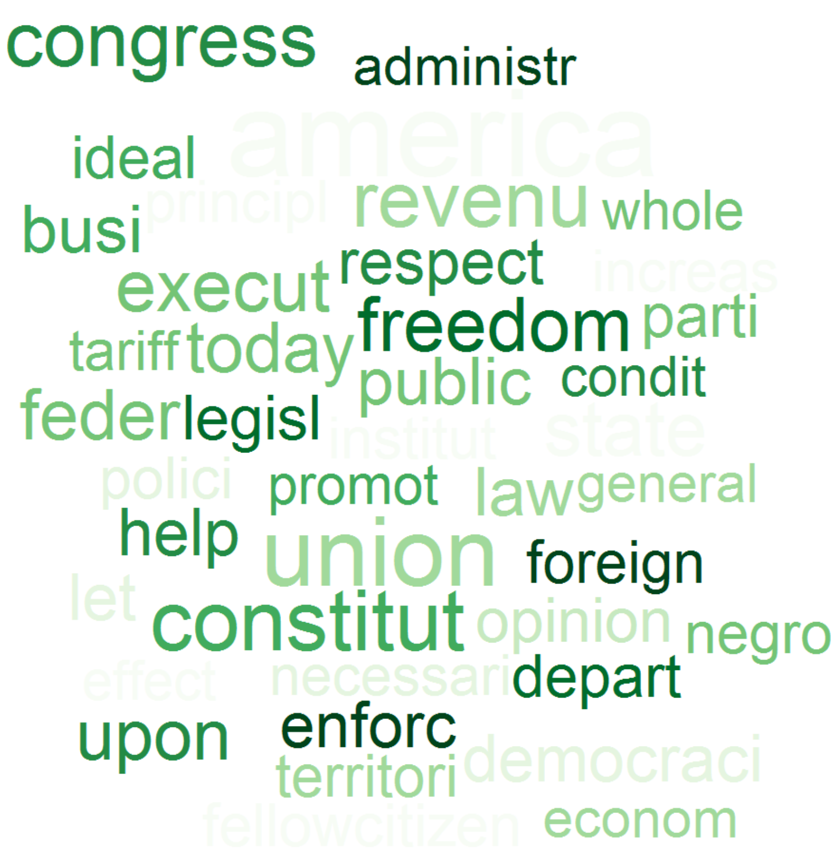
```
## png
##   2
```

Step5:Draw the wordcloud of all the speeches and also Donald trump’s speech

results shows that the most frequently used words in all speeches include “freedom”.“constitute”,“ideal”,“public”,“law”,“tariff”.

And in Donald Trump’s speech it includes “job”,“back”,“dream”,“everyone”. The results shows the main policy Mr.Trump is holding.

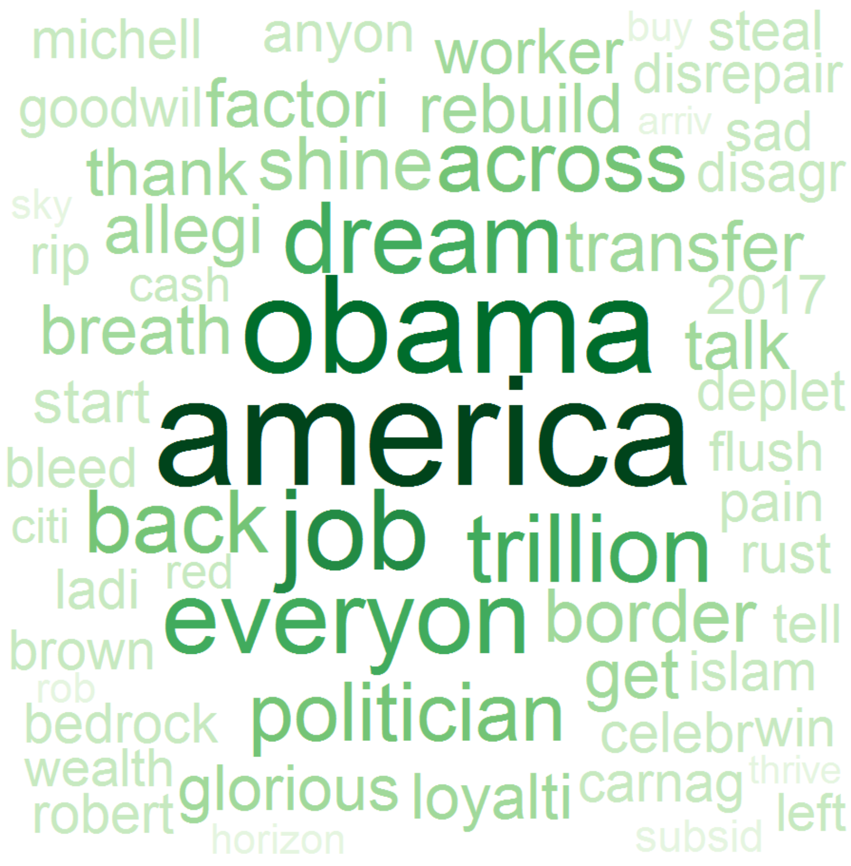
```
par(mfrow=c(1,1))
##wordcloud of all the speeches
wordcloud(names(freq_all), freq_all, min.freq=80,
              rot.per=0,
              random.color=TRUE,
              colors=brewer.pal(10,"Greens"))
```



```
png("../output/WordCloud/WordCloudAll.png",width=300, height=300)
wordcloud(names(freq_all), freq_all, min.freq=80,
              rot.per=0,
              random.color=TRUE,
              colors=brewer.pal(10,"Greens"))
dev.off()
```

```
## png
##   2
```

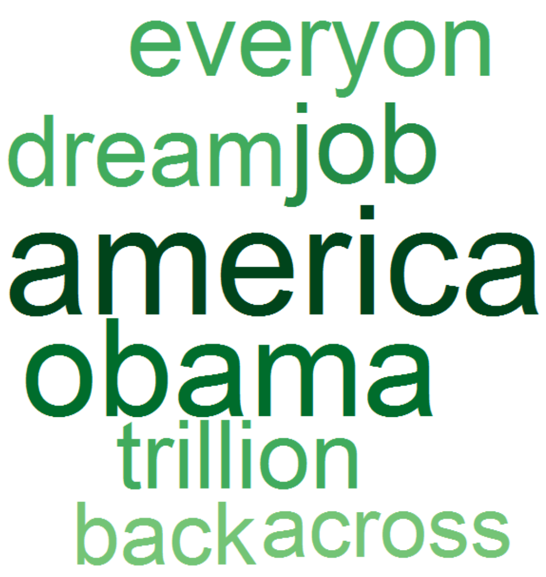
```
##trump's speech
Trump_Speech<-data.frame(term=ff.dtm$term[ff.dtm$document=="inaugDonaldJTrump-1.txt"],coun
t=ff.dtm$count[ff.dtm$document=="inaugDonaldJTrump-1.txt"])

Trump_Speech<-Trump_Speech[-3,]

##words at least mentioned 3 times
wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq=3,
              scale=c(5,0.5),
              max.words=200,
              random.order=FALSE,
              rot.per=0,
              use.r.layout=FALSE,
              random.color=FALSE,
              colors=brewer.pal(10,"Greens"))
```



```
png("../output/WordCloud/TrumpWordCloud1.png",width=300, height=300)
wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq=3,
              scale=c(5,0.5),
              max.words=200,
              random.order=FALSE,
              rot.per=0,
              use.r.layout=FALSE,
              random.color=FALSE,
              colors=brewer.pal(10,"Greens"))

dev.off()
```

```
## png
##   2
```

```r
##words at least mentioned 10 times
wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq=10,
              scale=c(5,0.5),
              max.words=200,
              random.order=FALSE,
              rot.per=0,
              use.r.layout=FALSE,
              random.color=FALSE,
              colors=brewer.pal(10,"Greens"))
```



```r
png("../output/WordCloud/TrumpWordCloud2.png",width=300, height=300)
wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq=10,
              scale=c(5,0.5),
              max.words=200,
              random.order=FALSE,
              rot.per=0,
              use.r.layout=FALSE,
              random.color=FALSE,
              colors=brewer.pal(10,"Greens"))
dev.off()
```

```
## png
##   2
```

Step6: Split Sentences Function

Split the speech into sentences which are used for further analysis.

```
##following uncommented code help analyze the frequent word in each speech:
# speech.corpus<-Corpus(VectorSource(speech.raw))
# speech.corpus<-CleanCorpus(speech.corpus)
# speech.tidy<-tidy(speech.corpus)
# dtm.speech<-DocumentTermMatrix(speech.corpus)
# freq_terms<-findFreqTerms(dtm.speech,lowfreq = 5)

SentencesAnalysis<-function(speech.raw,president.name){

# Split the speech into sentences
# qdap's sentSplit is modeled after dialogue data, so person field is needed
speech.df <- data.table(speech=speech.raw, person=president.name) #change to data frame
speechcombined<-sentCombine(speech.df) #combine
speechcombined.df<-data.table(speech=speechcombined$text.var,person=president.name) #and t
hen change to data frame
sentences<-data.table(sentSplit(speechcombined.df, "speech")) #split into sentenctes, and
keep in a list name "sentences",the list name for the splited sentences is "speech", the l
ast variable here

# Add a sentence counter and remove unnecessary variables
sentences[, sentence.num := seq(nrow(sentences))]
sentences[, person := NULL]
sentences[, tot := NULL]
setcolorder(sentences, c("sentence.num", "speech"))

# Syllables per sentence
sentences[, syllables := syllable_sum(speech)]
sentences<-na.omit(sentences)

# Add cumulative syllable count and percent complete as proxy for progression
sentences[, syllables.cumsum := cumsum(syllables)] ##get the cumulative sums of syllables
sentences[, pct.complete := syllables.cumsum / sum(sentences$syllables)] #get the percent
of syllables
sentences[, pct.complete.100 := pct.complete * 100]#percentage

return(sentences)
}
```

Step7: Sentiment Analysis Function: record the sentiment change throughout every speech

qdap’s sentiment analysis is based on a sentence-level formula classifying each word as either positive, negative, neutral, negator or amplifier, per Hu & Liu’s sentiment lexicon. The function also provides a word count.

```r
my.theme <-
  theme(plot.background = element_blank(), # Remove background
        panel.grid.major = element_blank(), # Remove gridlines
        panel.grid.minor = element_blank(), # Remove more gridlines
        panel.border = element_blank(), # Remove border
        panel.background = element_blank(), # Remove more background
        axis.ticks = element_blank(), # Remove axis ticks
        axis.text=element_text(size=14), # Enlarge axis text font
        axis.title=element_text(size=16), # Enlarge axis title font
        plot.title=element_text(size=24, hjust=0)) # Enlarge, left-align title

CustomScatterPlot <- function(gg)
  return(gg + geom_point(color="grey60") + # Lighten dots
           stat_smooth(color="royalblue", fill="lightgray", size=1.4) +
           xlab("Percent complete (by syllable count)") +
           scale_x_continuous(labels = percent) + my.theme)

SentimentAnalysis<-function(sentences,president.name){
  pol.df <- polarity(sentences$speech)$all#get the datafram of polarity analysis
  sentences[, words := pol.df$wc]   #wordcount
  sentences[, pol := pol.df$polarity]   #polarity score of words

  ##plot and save in output folder, sentimentplots subfolder


CustomScatterPlot(ggplot(sentences, aes(pct.complete, pol)) +
                    ylab("Sentiment (sentence-level polarity)") +
                    ggtitle(paste("Sentiment of",president.name,sep = " ")))

return(sentences)
}
```

Step8: Readability Tests Function

Readability Tests typically based on syllables, words, and sentences in order to approximate the grade level required to comprehend a text.

The higher the grade level stands for higher educated level.

Here the method used is automated readability index, which has the following score levels:

1 5-6 Kindergarten

2 6-7 First Grade

3 7-8 Second Grade

4 8-9 Third Grade

5 9-10 Fourth Grade

6 10-11 Fifth Grade

7 11-12 Sixth Grade

8 12-13 Seventh Grade

9 13-14 Eighth Grade

10 14-15 Ninth Grade

11 15-16 Tenth Grade

12 16-17 Eleventh grade

13 17-18 Twelfth grade

14 18-22 College

automated readability index=4.71*(characters/words)+0.5*(words/sentences)-21.43

```
ReadabilityAnalysis<-function(sentences,president.name){
  sentences[, readability := automated_readability_index(speech, sentence.num)
            $Readability$Automated_Readability_Index]

  CustomScatterPlot(ggplot(sentences, aes(pct.complete, readability)) +
                    ylab("Automated Readability Index") +
                    ggtitle(paste("Readability of",president.name,sep = " ")))

  return(sentences)
}
```

Step9: Memorability Analysis

Using google search hits to indicate the public opinion about sentences in each speech, the most popular senence would naturally have highest google hits. Here we plot the memorability of sentences throughout each speech, and also record 7 sentences with highest google hits in a csv file. one drawback of this method is that google will block our program after 300 times search.

```
GoogleHits <- function(query){
  require(XML)
  require(RCurl)

  url <- paste0("https://www.google.com/search?q=", gsub(" ", "+", query))

  CAINFO = paste0(system.file(package="RCurl"), "/CurlSSL/ca-bundle.crt")
  script <- getURL(url, followlocation=T, cainfo=CAINFO)
  doc <- htmlParse(script)
  res <- xpathSApply(doc, '//*/div[@id="resultStats"]', xmlValue)
  return(as.numeric(gsub("[^0-9]", "", res)))
}

GoogleHitsAnalysis<-function(sentences,president.name){

  sentences[, google.hits := GoogleHits(paste0("[", gsub("[,;!.]", "", speech),
                                              "]",president.name,"inaugural","speech"))]

  googlehits.sentences<-head(sentences[order(-google.hits)]$speech, 7)

  write.csv(googlehits.sentences,file = paste("../output/Memorability/HitsSentences/","Hit
sSentences_",president.name,".csv",sep = ""))

  #Plotting Google hits on a log scale reduces skew and allows us to work on a ratio scale
.
  sentences[, log.google.hits := log(google.hits)]

  CustomScatterPlot(ggplot(sentences, aes(pct.complete, log.google.hits)) +
                    ylab("Memorability (log of sentence's Google hits)") +
                    ggtitle(paste("Memorability of",president.name,sep = " ")))

  return(sentences)
}
```

Step10: Loop through all the speeches and perform Sentiment Analysis and Readability Analysis. Since google literally blocked me after trying to search 300th times using code, I choose to use DonaldTrump’s speech as an example of Memorability Analysis in step11.

Sentiment Analysis Results graphs are saved in ../output/SentimentPlots Folder

Readability Analysis Results graphs are saved in ../output/ReadabilityPlots Folder

```
for ( i in 1:length(speeches)){
  filename<-speeches[i]
  president.name<-substr(filename,6,nchar(filename)-4)
  speech.raw<-read.table(paste(folder.path,filename,sep = ""),quote = NULL,comment="",hea
der = FALSE,fill = TRUE)
  sentences<-SentencesAnalysis(speech.raw,president.name)

  ##sentiment Analysis
  sentences<-SentimentAnalysis(sentences,president.name)
  ggsave(paste("../output/SentimentPlots/","Sentiment_",president.name,".png",sep = ""),p
lot=last_plot())

  ##Readability Analysis
  sentences<-ReadabilityAnalysis(sentences,president.name)
  ggsave(paste("../output/ReadabilityPlots/","Readability_",president.name,".png",sep = ""
),plot=last_plot())

}
```

Results shows that:

For sentiment:

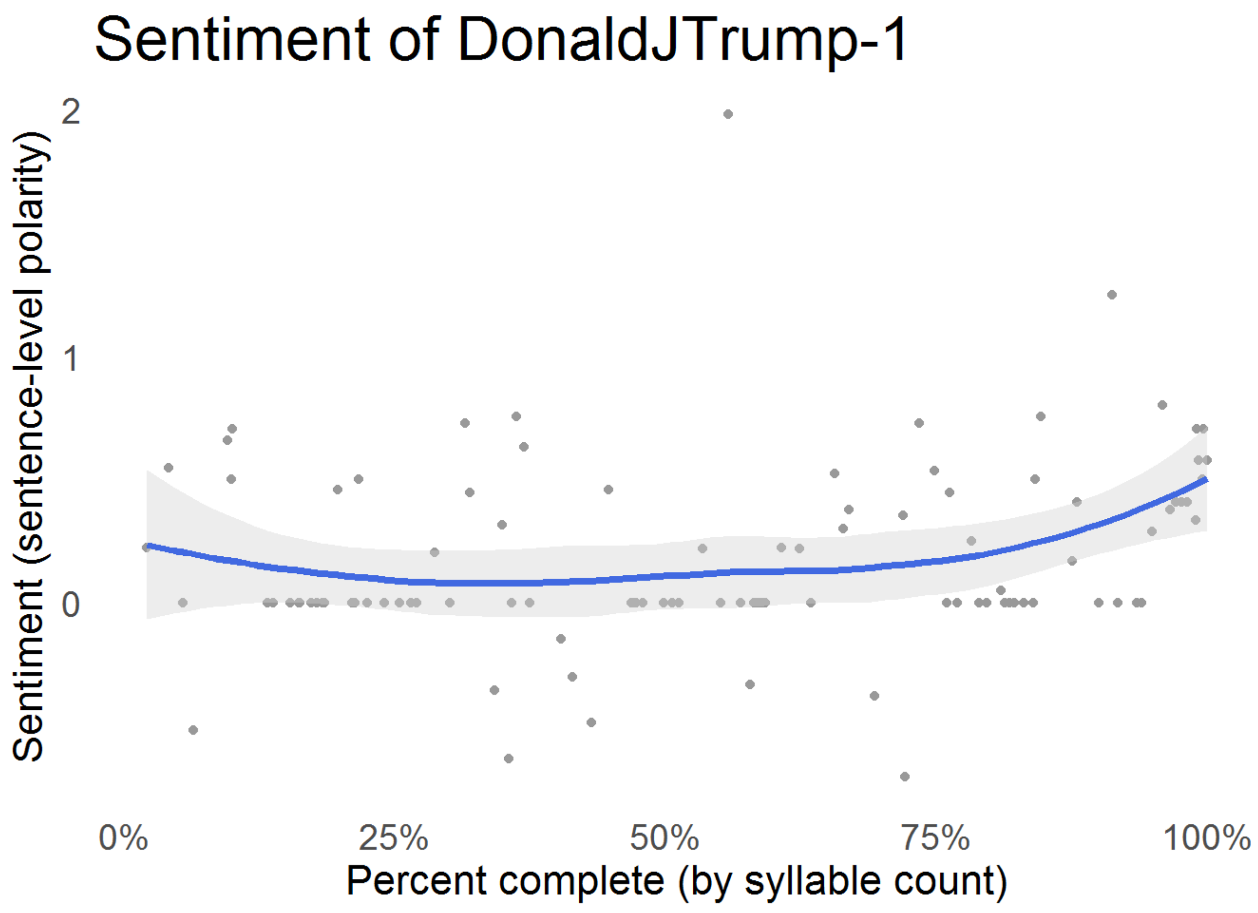
most of the speech express more positive feelings at the end of the speech compmaring to the begining.

For readability:

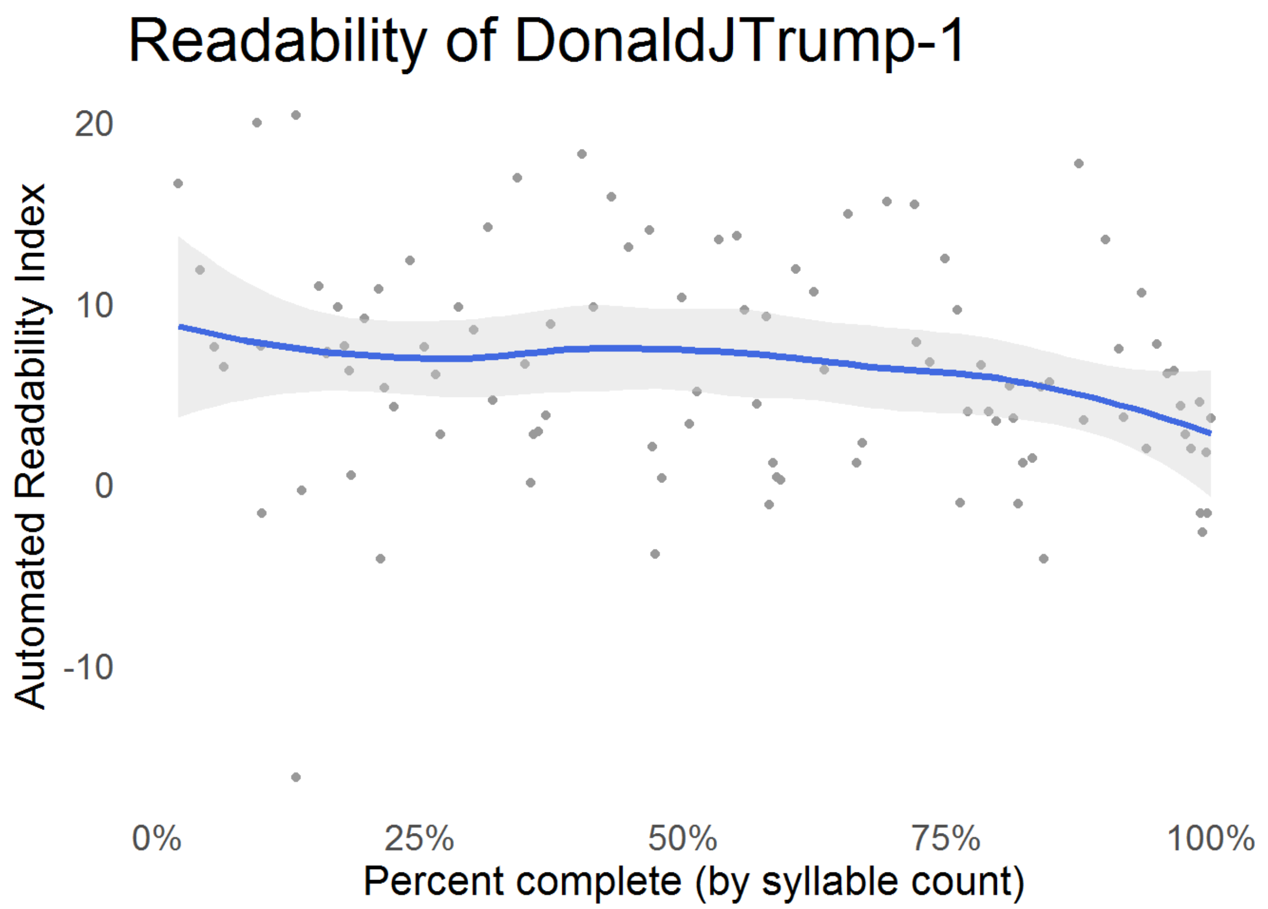
past presidents’ speech require higher level of grade to understand their speech than recent presidents.

Step11: Sentiment,Readibility and Memorability Analysis of Donald Trump’s Speech

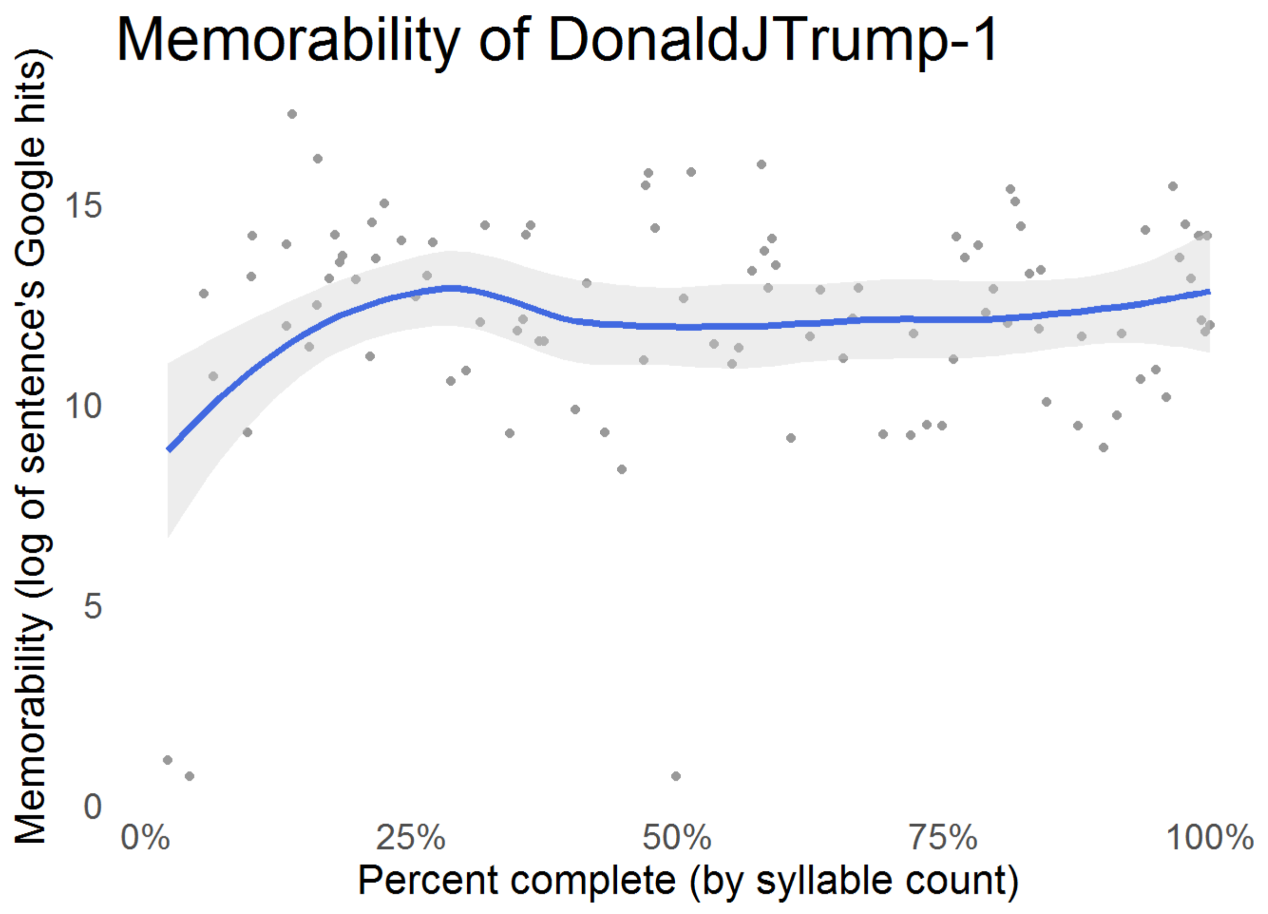
```
  i=9
  filename<-speeches[i]
  president.name<-substr(filename,6,nchar(filename)-4)
  speech.raw<-read.table(paste(folder.path,filename,sep = ""),quote = NULL,comment="",hea
der = FALSE,fill = TRUE)
  sentences<-SentencesAnalysis(speech.raw,president.name)

  ##sentiment Analysis
  sentences<-SentimentAnalysis(sentences,president.name)
  last_plot()
```



```
  ##Readability Analysis
  sentences<-ReadabilityAnalysis(sentences,president.name)
  last_plot()
```



```
  ##Memorability Analysis,i.e Google Hits Analysis
  sentences<-GoogleHitsAnalysis(sentences,president.name)
  ggsave(paste("../output/Memorability/Plots/","Memorability_",president.name,".png",sep =
"") ,plot=last_plot())
  last_plot()
```



```r
##the most heat 7 sentences in Trump's speech
head(sentences[order(-google.hits)]$speech, 7)
```

```
## [1] "and giving it back to you, the people."
## [2] "closed."
## [3] "America will start winning again, winning like never before."
## [4] "be only America first, America first."
## [5] "But that is the past."
## [6] "across the world."
## [7] "Together, we will make America strong again."
```

Analysis of Mr.Trump’s speech:

Sentiment:

the sentiment has a trend of going negative in the first half of the speech, and then going positive in the second part. This meet the pattern of almost all other speeches, that is expressing concerns first and then the promises toward future. Negative sentiments centered in 25% part of Mr.Trump’s speech.

Readability:

the readability of Mr.Trump’s speech is starting around 10 and decreading to around 5, this is average the lowest even in recent presidents, comparing to Obama’s and Bush’s around 10. Not to mention presidents in the last century’s readability usually pass 20, even reach 50 sometimes, like ThomasJefferson.

Memorability:

The google search hits indicates that the 25% part of Mr.Trump’s speech is the most memorable.

Observation:

25% is the place with most centered negative sentiment and also with hightest memorability, which indicates that in Mr.Trump’s speech, negative sentiment is the one public pays most attention to.

Step12:further explore the determinants of memorability of sentences

Results shows that the higher level of reading grade, the less memorability it is for the sentences.

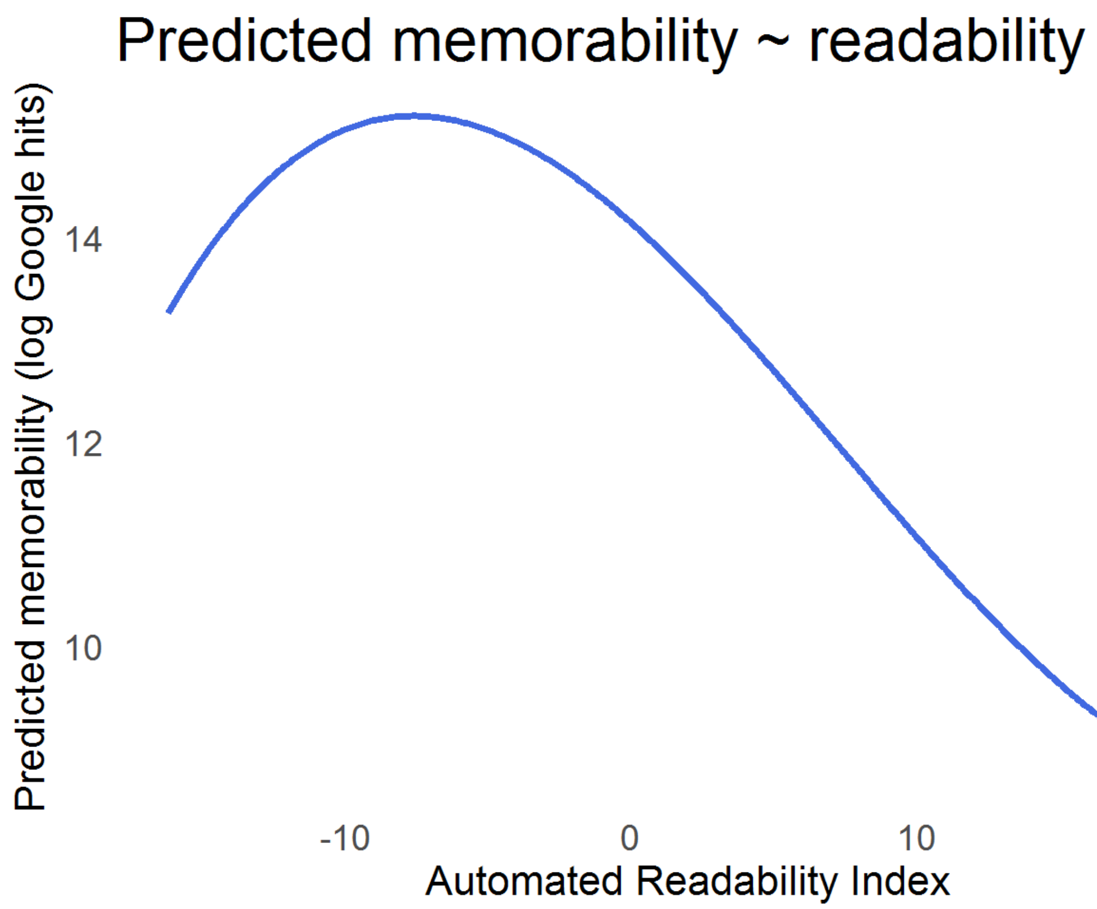
```r
google.lm <- stepAIC(lm(log(google.hits) ~ poly(readability, 3) + pct.complete.100, data=
sentences))
```

```
## Start:  AIC=168.98
## log(google.hits) ~ poly(readability, 3) + pct.complete.100
##
##                        Df Sum of Sq    RSS    AIC
## - pct.complete.100      1     5.332 492.78 168.08
## <none>                              487.45 168.98
## - poly(readability, 3)  3   271.902 759.35 207.75
##
## Step:  AIC=168.08
## log(google.hits) ~ poly(readability, 3)
##
##                        Df Sum of Sq    RSS    AIC
## <none>                              492.78 168.08
## - poly(readability, 3)  3    270.62 763.40 206.29
```

```
summary(google.lm)
```

```
##
## Call:
## lm(formula = log(google.hits) ~ poly(readability, 3), data = sentences)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.2878  -0.7159   0.1121   1.0263   4.6474
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            12.2041     0.2243  54.416   <2e-16 ***
## poly(readability, 3)1 -15.5852     2.2539  -6.915    5e-10 ***
## poly(readability, 3)2  -3.9690     2.2539  -1.761   0.0814 .
## poly(readability, 3)3   3.4597     2.2539   1.535   0.1281
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.254 on 97 degrees of freedom
## Multiple R-squared:  0.3545, Adjusted R-squared:  0.3345
## F-statistic: 17.76 on 3 and 97 DF,  p-value: 2.894e-09
```

```
new.data <- data.frame(readability=seq(min(sentences$readability),
                                       max(sentences$readability), by=0.1),
                       pct.complete.100=mean(sentences$pct.complete.100))

new.data$pred.hits <- predict(google.lm, newdata=new.data)

ggplot(new.data, aes(readability, pred.hits)) +
  geom_line(color="royalblue", size=1.4) +
  xlab("Automated Readability Index") +
  ylab("Predicted memorability (log Google hits)") +
  ggtitle("Predicted memorability ~ readability") +
  my.theme
```



```
ggsave(paste("../output/Memorability/Plots/","Memorability_readability",president.name,".p
ng",sep = ""),plot=last_plot())

usedtime<-proc.time() - ptm
##time used in runing the program
usedtime
```

```
##    user  system elapsed 
##  129.20    7.30  143.64 
```

this program can be reproduced within 3 minutes

Loading [Contrib]/a11y/accessibility-menu.js