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)
```

```
## Loading required package: NLP
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(RColorBrewer)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidytext)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```r
library(SnowballC)
library(qdap)
```

```
## Loading required package: qdapDictionaries
```

```
## Loading required package: qdapRegex
```

```
##
## Attaching package: 'qdapRegex'
```

```
## The following object is masked from 'package:ggplot2':
##
##     %+%
```

```
## The following objects are masked from 'package:dplyr':
##
##     escape, explain
```

```
## Loading required package: qdapTools
```

```
##
## Attaching package: 'qdapTools'
```

```
## The following object is masked from 'package:dplyr':
##
##     id
```

```
##
## Attaching package: 'qdap'
```

```
## The following object is masked from 'package:dplyr':
##
##     %>%
## The following objects are masked from 'package:tm':
##
##     as.DocumentTermMatrix, as.TermDocumentMatrix
```

```
## The following object is masked from 'package:NLP':
##
##     ngrams
```

```
## The following object is masked from 'package:base':
##
##     Filter
```

```r
library(sentimentr)
library(data.table)
```

```
## -------------------------------------------------------------------------
```

```
## data.table + dplyr code now lives in dtplyr.
## Please library(dtplyr)!
```

```
## -------------------------------------------------------------------------
```

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

```
## The following object is masked from 'package:qdapTools':
##
##     shift
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
library(scales)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

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-Pr
oj1-eeelaine")
folder.path<-paste(getwd(),"/data/InauguralSpeeches/",sep = "")
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
```

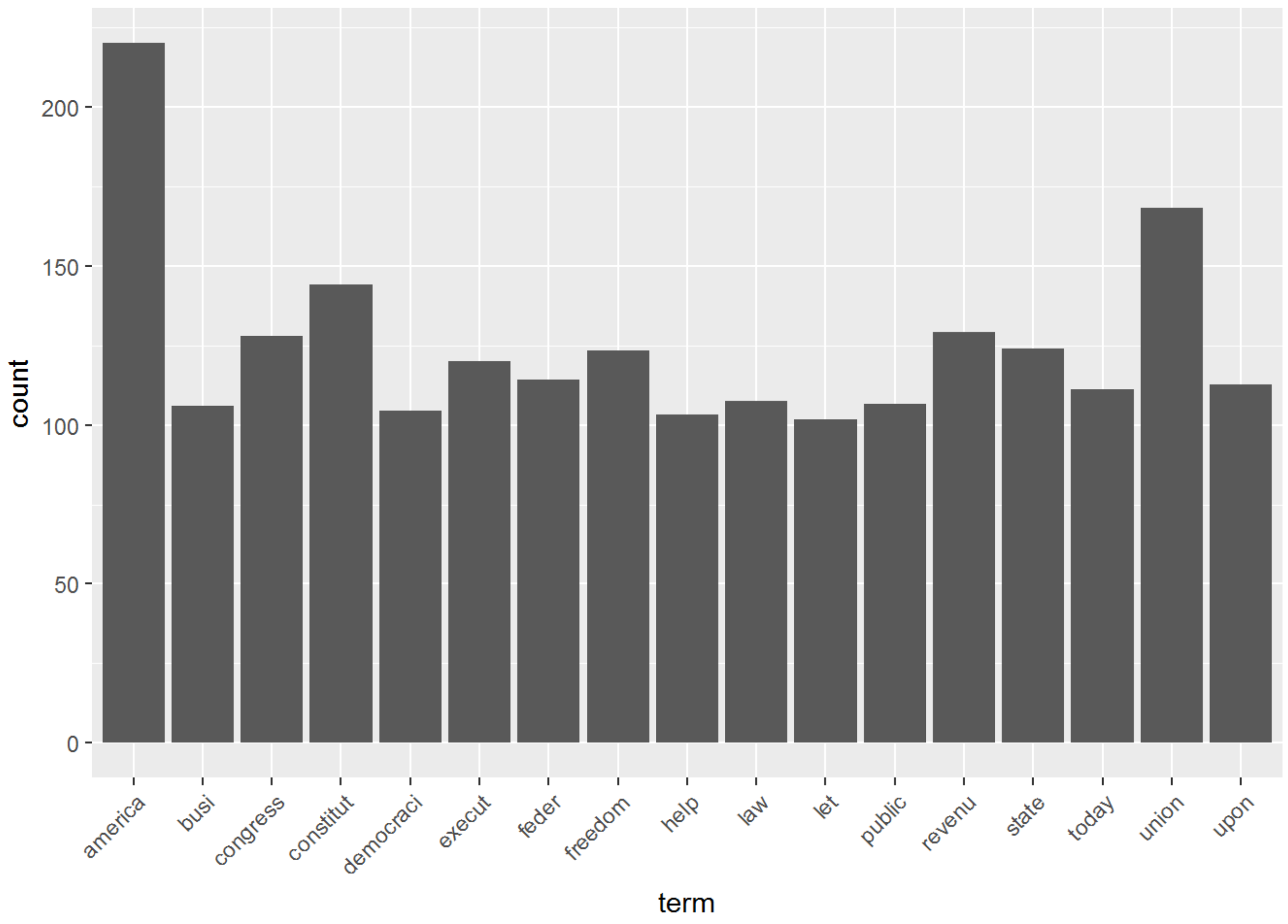
```
##15 most frequently used words
freq_all <- sort(colSums(as.matrix(dtm.all)), decreasing=TRUE)
head(freq_all, 15)
```

```
##   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
```



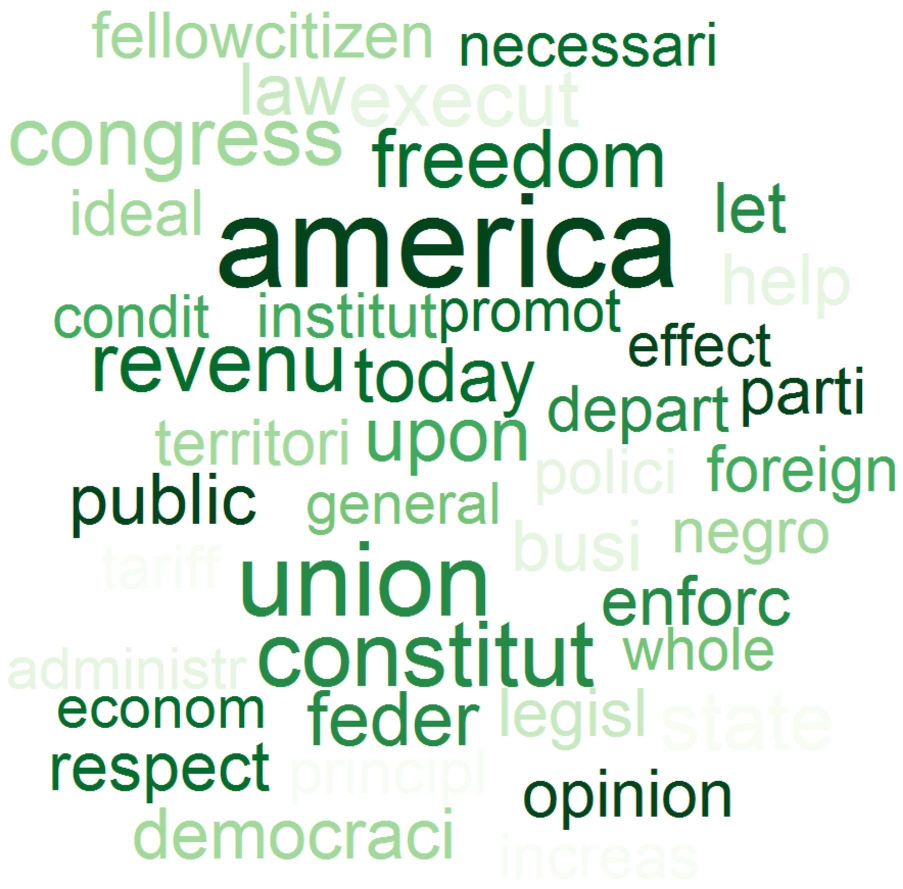
```
# ggsave(paste(getwd(),"/output/","AllSpeechesWordCloud,".png",sep = ""),plot=last_
plot())
```

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

```
## Warning in brewer.pal(10, "Greens"): n too large, allowed maximum for palette Gr
eens is 9
## Returning the palette you asked for with that many colors
```



```
# dev.off()

##trump's speech
Trump_Speech<-data.frame(term=ff.dtm$term[ff.dtm$document=="inaugDonaldJTrump-1.txt
"],count=ff.dtm$count[ff.dtm$document=="inaugDonaldJTrump-1.txt"])

Trump_Speech<-Trump_Speech[-3,]

##words at least mentioned 3 times
# png(paste(getwd(),"/output/WordCloud/", "TrumpWordCloud1",".png",sep = ""),
      # 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"))
```

```
## Warning in brewer.pal(10, "Greens"): n too large, allowed maximum for palette Gr
eens is 9
## Returning the palette you asked for with that many colors
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## disrepair could not be fit on page. It will not be plotted.
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## goodwil could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## infrastructur could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## landscap could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## nebraska could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## reinforc could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## solidar could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## sprawl could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## stolen could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## tombston could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## tunnel could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## unreal could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## unstopp could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## urban could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## windswept could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## capit could not be fit on page. It will not be plotted.
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## today could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## mountain could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## movement could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## righteous could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## total could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## wealth could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## million could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## forgotten could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## stop could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## airport could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## cash could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## crucial could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## detroit could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## empti could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## gang could not be fit on page. It will not be plotted.
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## mysteri could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## ravag could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## reap could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## red could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## student could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## unlock could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## wealthi could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## fight could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## gather could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## american could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## allow could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## bigger could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## carter could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## clinton could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## dissip could not be fit on page. It will not be plotted.
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## drug could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## erad could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## januari could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## millennium could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## pleasant could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## radic could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## railway could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## rediscov could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## redistribut could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## salut could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## shutter could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## spent could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## subsid could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## victori could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## women could not be fit on page. It will not be plotted.
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## black could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## compani could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## complain could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## highway could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## horizon could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## scatter could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## whether could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## togeth could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## washington could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## challeng could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## triumph could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## bridg could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## decay could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## decre could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## dollar could not be fit on page. It will not be plotted.
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## enrich could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## flourish could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## har could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## hardship could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## listen could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## match could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## neighborhood could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## potenti could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## space could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## stir could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## thrive could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## protect could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## bring could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## forward could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## depriv could not be fit on page. It will not be plotted.
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## hall could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## night could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## transit could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## white could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## longer could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## militari could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## belong could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## anyth could not be fit on page. It will not be plotted.
```

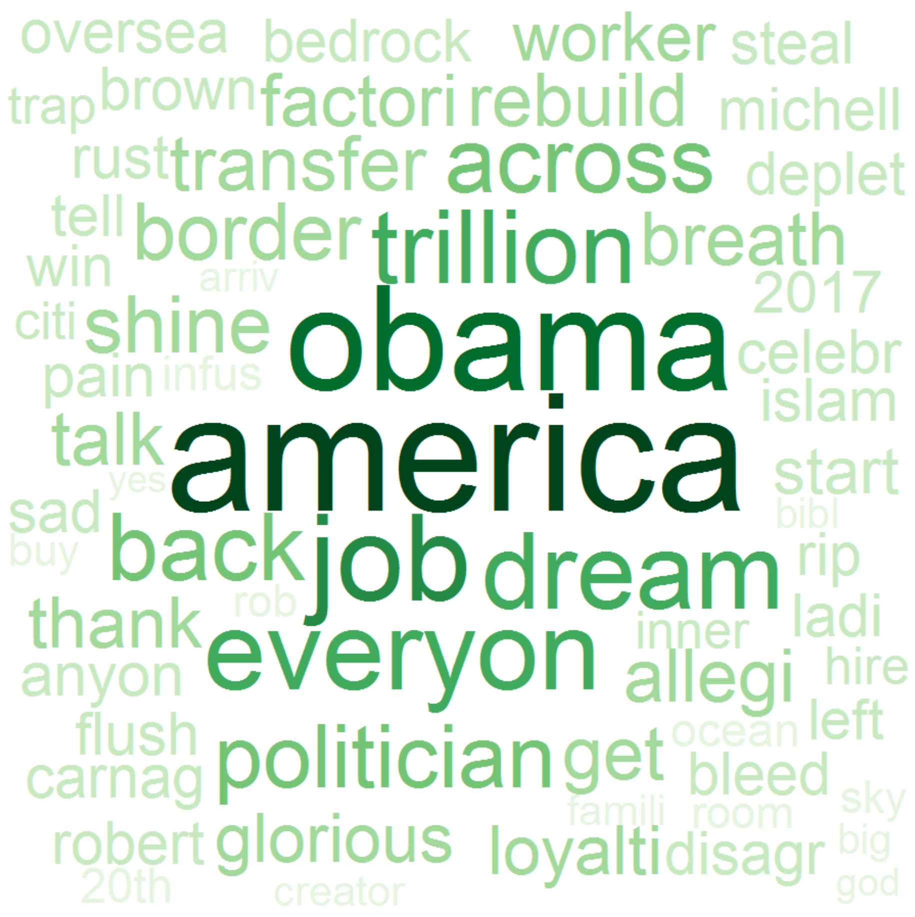
```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## bush could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## gracious could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## magnific could not be fit on page. It will not be plotted.
```

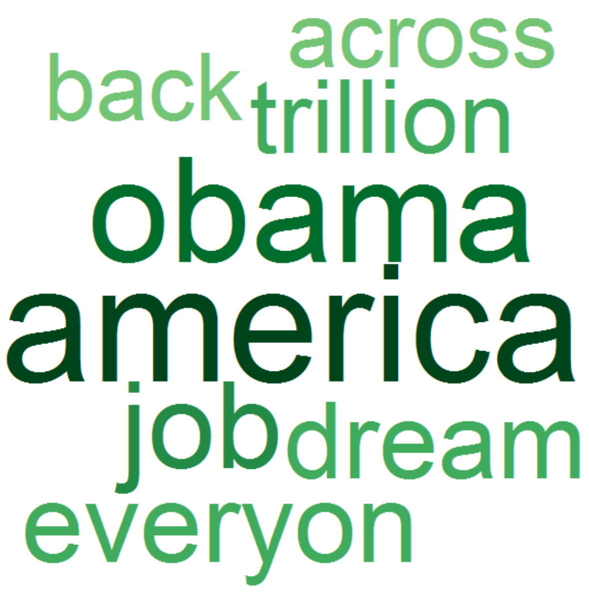
```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## miseri could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(Trump_Speech$term, Trump_Speech$count, min.freq = 3, :
## technolog could not be fit on page. It will not be plotted.
```



```
# dev.off()

##words at least mentioned 10 times
# png(paste(getwd(),"/output/WordCloud/", "TrumpWordCloud2",".png",sep = ""),
      # 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"))
```

```
## Warning in brewer.pal(10, "Greens"): n too large, allowed maximum for palette Gr
eens is 9
## Returning the palette you asked for with that many colors
```



```
# dev.off()
```

Step6: Split Sentences Function

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

```r
##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 f
rame
speechcombined<-sentCombine(speech.df) #combine
speechcombined.df<-data.table(speech=speechcombined$text.var,person=president.name)
#and then change to data frame
sentences<-data.table(sentSplit(speechcombined.df, "speech")) #split into sentencte
s, and keep in a list name "sentences",the list name for the splited sentences is "
speech", the last 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 syl
lables
sentences[, pct.complete := syllables.cumsum / sum(sentences$syllables)] #get the p
ercent of syllables
sentences[, pct.complete.100 := pct.complete * 100]#percentage

return(sentences)
}
```

Step6: 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)
}
```

Step7: 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

```r
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)
}
```

Step8: 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),
                                               "]"))]

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

  # write.csv(googlehits.sentences,file = paste(getwd(),"/output/Memorability/HitsS
entences/","HitsSentences_",president.name,".csv",sep = ""))

  #Plotting Google hits on a log scale reduces skew and allows us to work on a rati
o 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)
}
```

Step9: 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.

this part can not be knit to html since there is a loop in saving pictures to folders, so we do not run this while kniting, instead we run it alone to generate results.

```
# 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,commen
t="",header = FALSE,fill = TRUE)
#   sentences<-SentencesAnalysis(speech.raw,president.name)
#
#   ##sentiment Analysis
#   sentences<-SentimentAnalysis(sentences,president.name)
#   ggsave(paste(getwd(),"/output/SentimentPlots/","Sentiment_",president.name,".pn
g",sep = ""),plot=last_plot())
#
#   ##Readability Analysis
#   sentences<-ReadabilityAnalysis(sentences,president.name)
#   ggsave(paste(getwd(),"/output/ReadabilityPlots/","Readability_",president.name,
".png",sep = ""),plot=last_plot())
# }
```

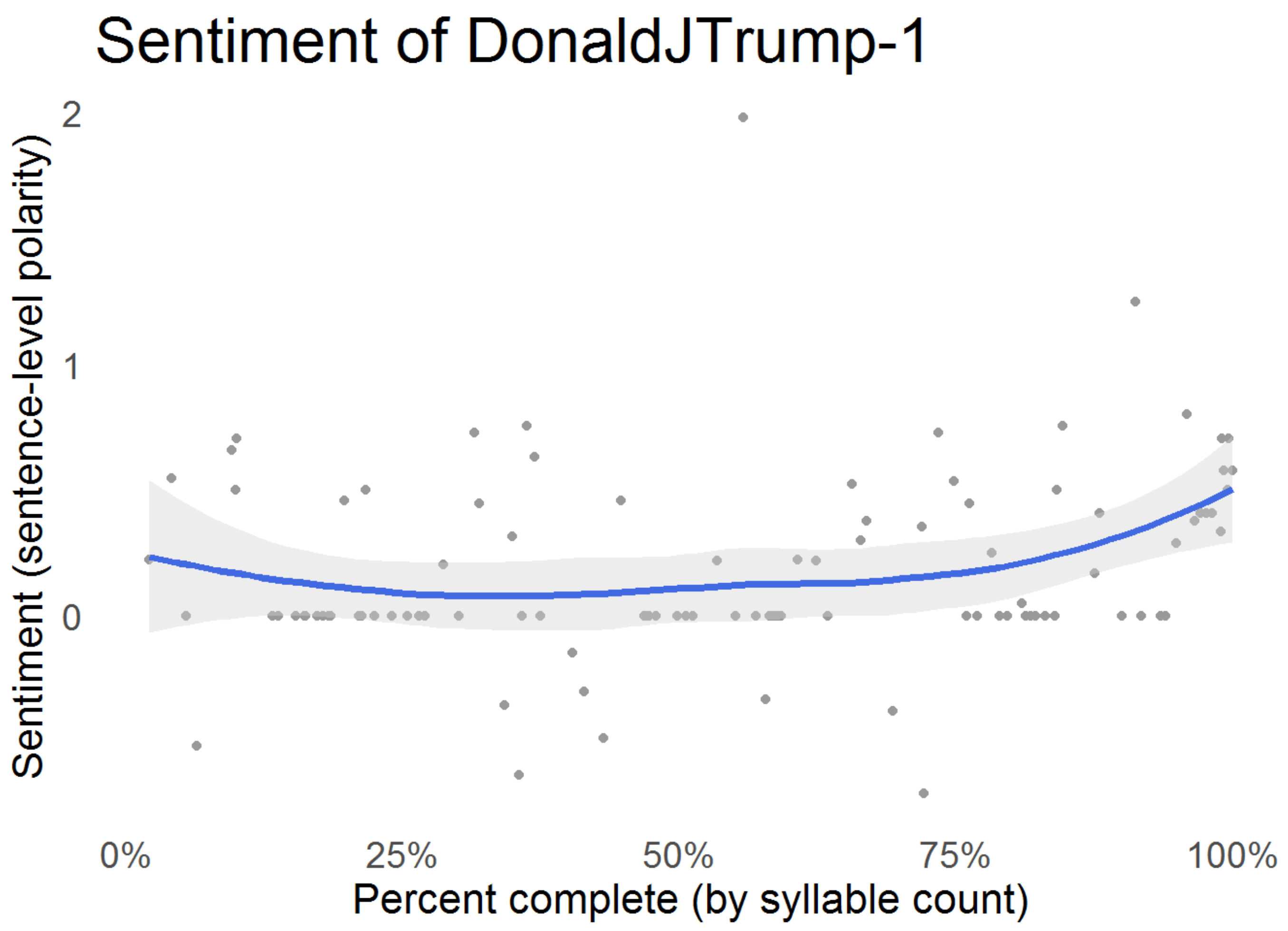
Results shows that for sentiment, most of the speech express more positive feelings at the end of the speech compmaring to the begining.

For readability, past presidents' speech require higher level of grade to understand their speech than recent presidents.

Step10: Sentiment,Readibility and Memorability Analysis of Donald Trump's Speech

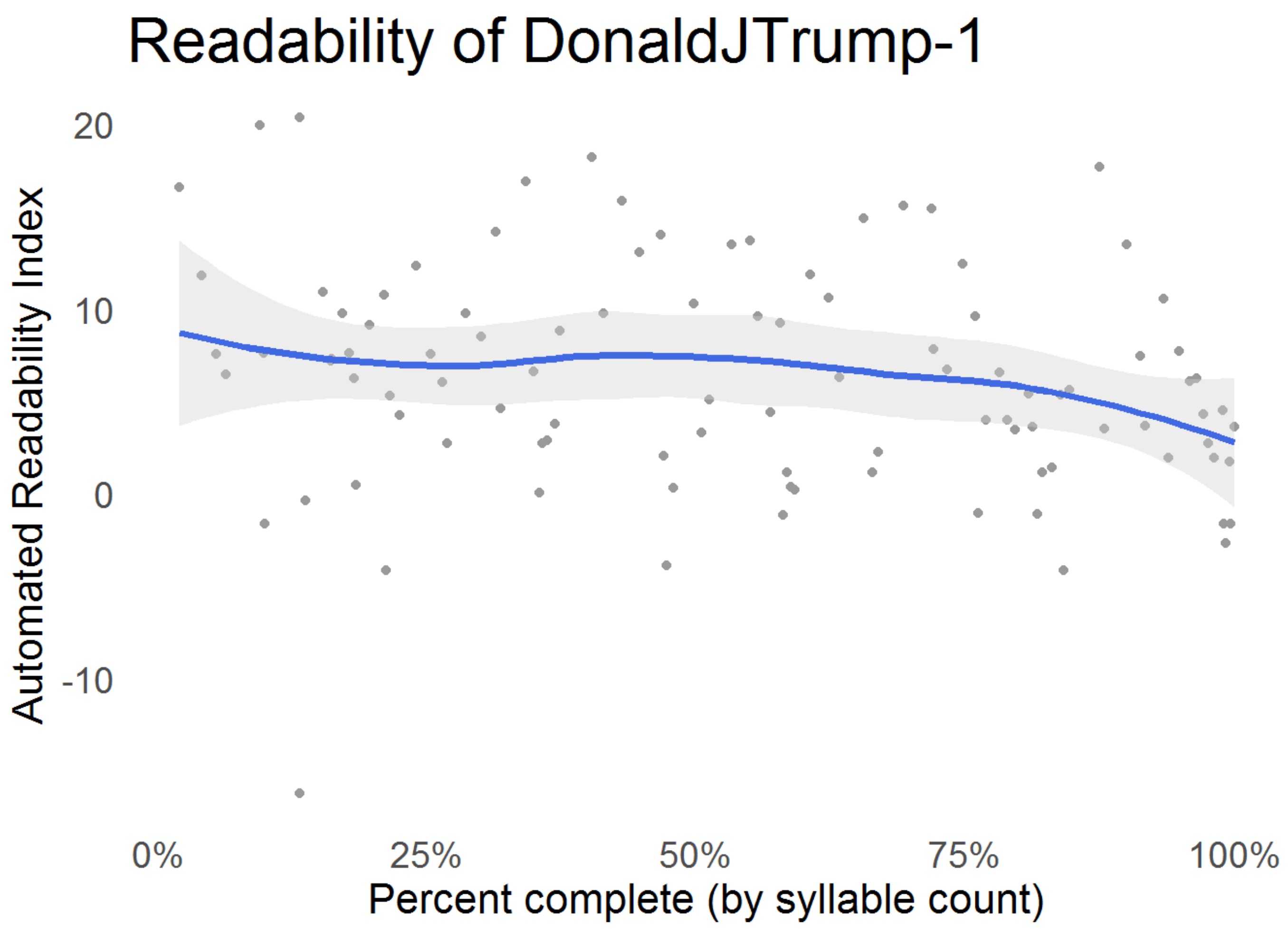
```
  i=9
  filename<-speeches[i]
  president.name<-substr(filename,6,nchar(filename)-4)
  speech.raw<-read.table(paste(folder.path,filename,sep = ""),quote = NULL,comment
="",header = FALSE,fill = TRUE)
  sentences<-SentencesAnalysis(speech.raw,president.name)

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

```
## `geom_smooth()` using method = 'loess'
```



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

```
## `geom_smooth()` using method = 'loess'
```



```
  ##Memorability Analysis,i.e Google Hits Analysis
  sentences<-GoogleHitsAnalysis(sentences,president.name)
```

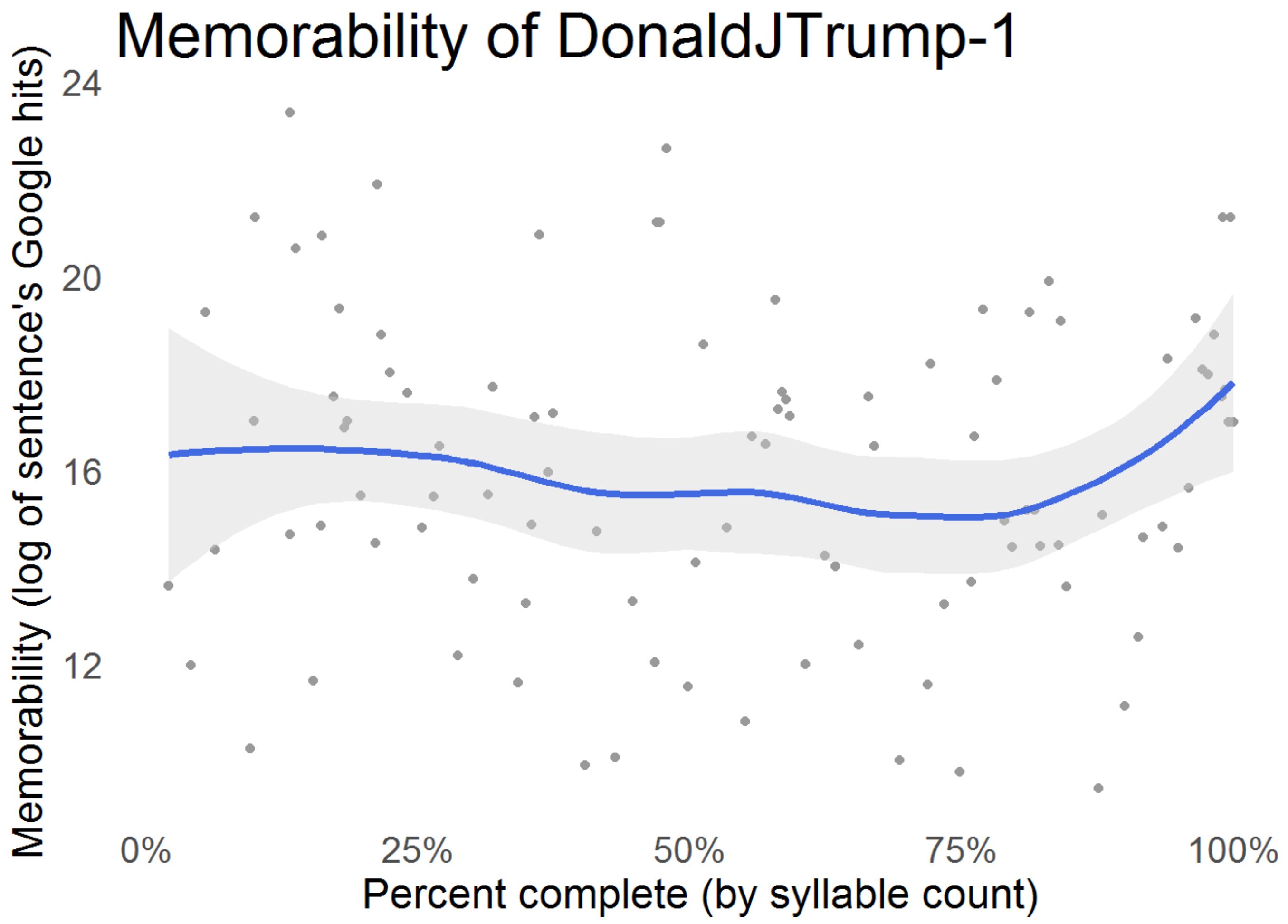
```
## Loading required package: XML
```

```
## Loading required package: RCurl
```

```
## Loading required package: bitops
```

```
  # ggsave(paste(getwd(),"/output/Memorability/Plots/","Memorability_",president.na
me,".png",sep = ""),plot=last_plot())
  last_plot()
```

```
## `geom_smooth()` using method = 'loess'
```



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

```
## [1] "C."
## [2] "And now, we are looking only to the future."
## [3] "This is your day."
## [4] "Thank you."
## [5] "Thank you."
## [6] "Thank you."
## [7] "across the world."
```

Step11: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.

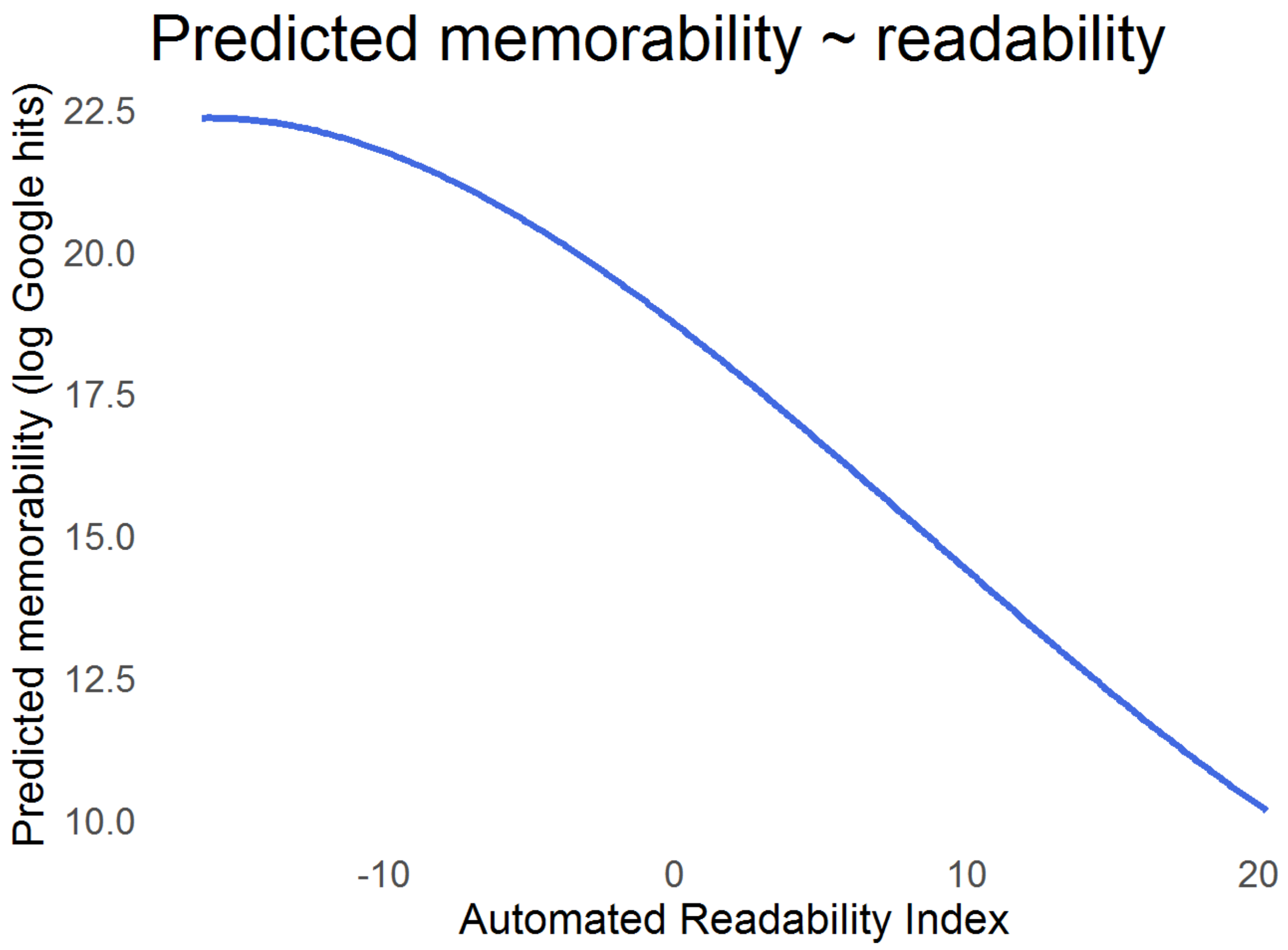
```
google.lm <- stepAIC(lm(log(google.hits) ~ poly(readability, 3) + pol + pct.comple
te.100, data=sentences))
```

```
## Start:  AIC=158.07
## log(google.hits) ~ poly(readability, 3) + pol + pct.complete.100
## 
##                        Df Sum of Sq     RSS    AIC
## - pol                   1      3.54  432.52 156.91
## <none>                               428.98 158.07
## - pct.complete.100      1     37.06  466.03 164.44
## - poly(readability, 3)  3    593.31 1022.28 239.78
## 
## Step:  AIC=156.91
## log(google.hits) ~ poly(readability, 3) + pct.complete.100
## 
##                        Df Sum of Sq     RSS    AIC
## <none>                               432.52 156.91
## - pct.complete.100      1     33.90  466.42 162.53
## - poly(readability, 3)  3    593.68 1026.20 238.17
```

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

```
## 
## Call:
## lm(formula = log(google.hits) ~ poly(readability, 3) + pct.complete.100, 
##     data = sentences)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -4.2085 -1.5960 -0.2101  1.6360  4.8571 
## 
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)    
## (Intercept)            17.08986    0.45735  37.367  < 2e-16 ***
## poly(readability, 3)1 -24.78301    2.16561 -11.444  < 2e-16 ***
## poly(readability, 3)2  -2.59942    2.16953  -1.198  0.23381    
## poly(readability, 3)3   1.75800    2.13875   0.822  0.41313    
## pct.complete.100       -0.02044    0.00745  -2.743  0.00726 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2.123 on 96 degrees of freedom
## Multiple R-squared:  0.5787, Adjusted R-squared:  0.5611 
## F-statistic: 32.96 on 4 and 96 DF,  p-value: < 2.2e-16
```

```
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(getwd(),"/output/Memorability/Plots/","Memorability_readability",pre
sident.name,".png",sep = ""),plot=last_plot())

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

```
##    user  system elapsed
##   13.53    1.50   15.58
```

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