############################TEXTMINING#####################################

#OBJECTIVE:

#1) Extract tweets for any user (try choosing a user who has more tweets)

#2) Perform sentimental analysis on the tweets extracted from the above

##################################################################################

#devtools::install\_github("jrowen/twitteR", ref = "oauth\_httr\_1\_0")

#devtools::install\_version("httr",version="0.6.0",repos="http://cran.us.r-project.org")

install.packages("twitteR")

library("twitteR")

install.packages("ROAuth")

library("ROAuth")

install.packages("base64enc")

library(base64enc)

install.packages("httpuv")

library(httpuv)

cred <- OAuthFactory$new(consumerKey='yPv2Z0W5ZTuiMVaZdAytqH8H0',

consumerSecret='E1UIuJczP1JRM0VFb36Z1BScMcJobPGtzqx9XUBU2nTkwJcIo7',

requestURL='https://api.twitter.com/oauth/request\_token',

accessURL='https://api.twitter.com/oauth/access\_token',

authURL='https://api.twitter.com/oauth/authorize')

#save(cred, file="twitter authentication.Rdata")

#load("twitter authentication.Rdata")

setup\_twitter\_oauth("yPv2Z0W5ZTuiMVaZdAytqH8H0",

"E1UIuJczP1JRM0VFb36Z1BScMcJobPGtzqx9XUBU2nTkwJcIo7",

"995572125254782976-VRxUhTLGjBeUn53q0CJTeIhPliDXjnS", # Access token

"TPX35FMKDruUOukEaLOZhVE8uxvpBcrZXMnWs8pUwHaW2") # Access token secret key

Tweets <- userTimeline('APJAbdulKalam', n = 1000)

TweetsDF <- twListToDF(Tweets)

write.csv(TweetsDF, "Tweets\_APJKalam.csv")

getwd()

##################################################################

fin\_txt<-sapply(Tweets, function(x) x$getText())

str(fin\_txt)

library(tm)

fin\_corpus<- Corpus(VectorSource(fin\_txt))

inspect(fin\_corpus[100])

fin\_clean<-tm\_map(fin\_corpus, removePunctuation)

fin\_clean<-tm\_map(fin\_clean, content\_transformer(tolower))

fin\_clean<-tm\_map(fin\_clean, removeWords, stopwords("english"))

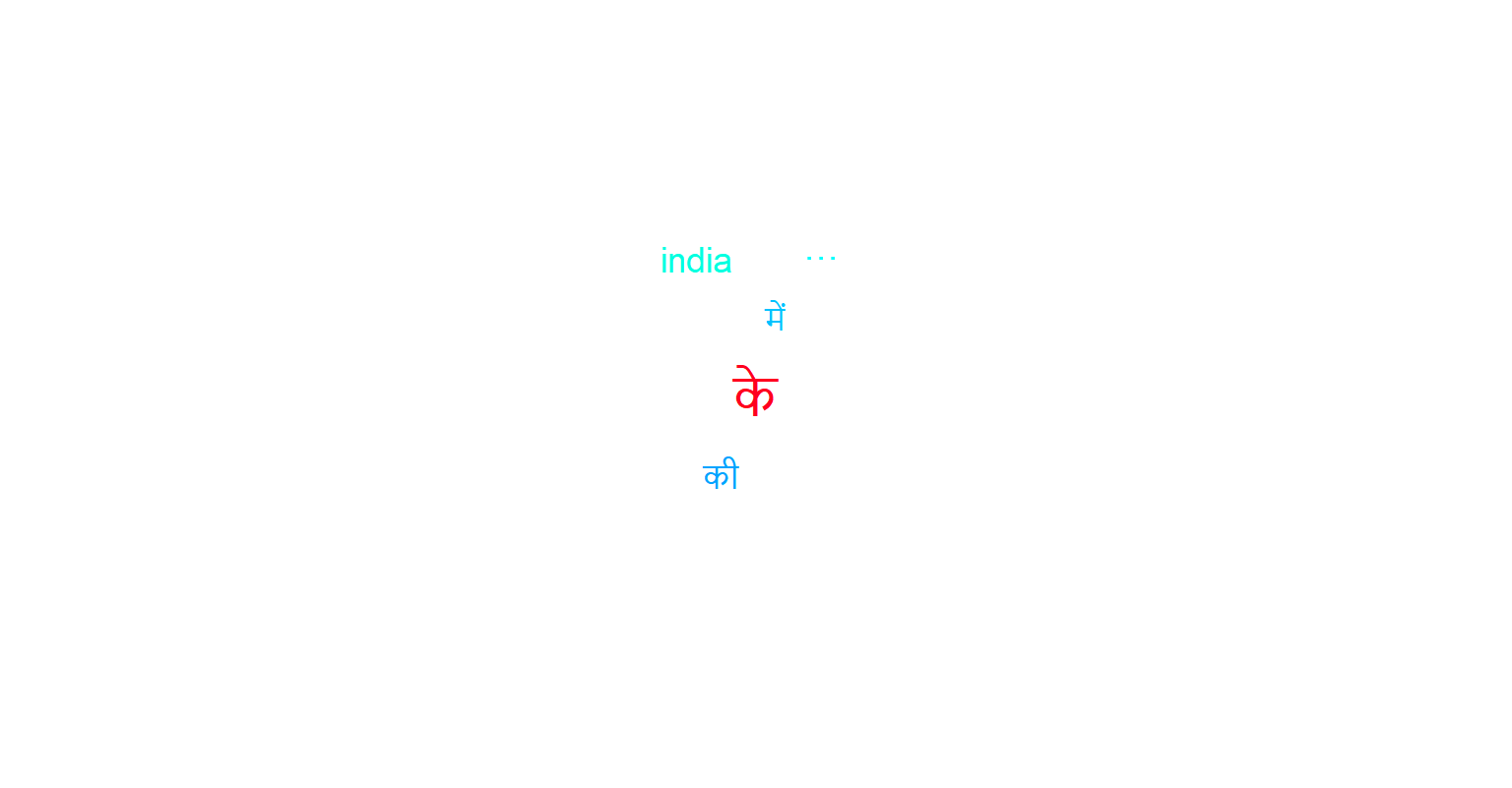
fin\_clean<-tm\_map(fin\_clean,removeNumbers)

fin\_clean<-tm\_map(fin\_clean, stripWhitespace)

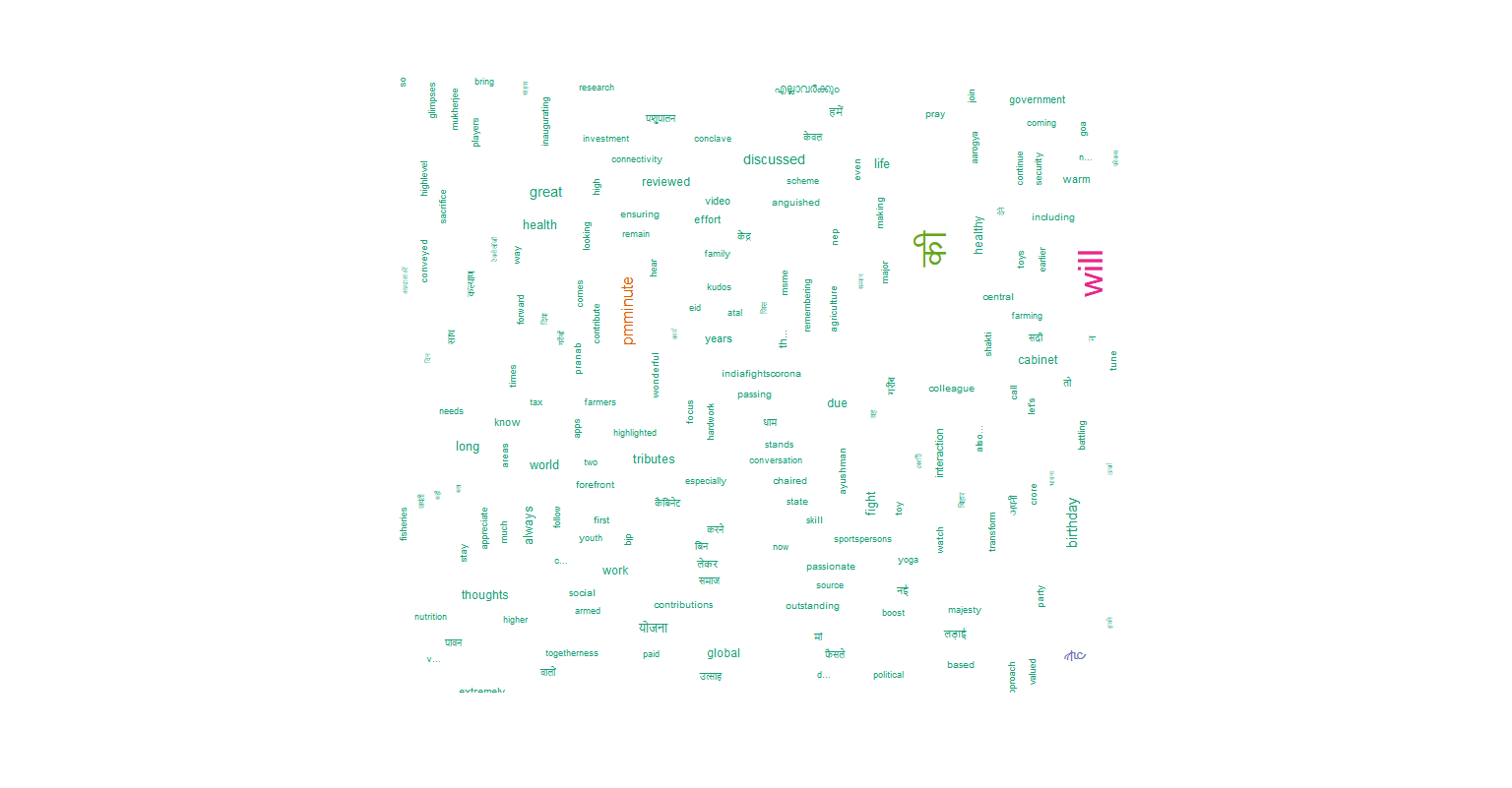
fin\_clean<-tm\_map(fin\_clean, removeWords, c("gameofthrones")) ## clean some words

library(wordcloud)

wordcloud(fin\_clean, random.order = F, max.words = 5,colors=rainbow(50))



.wordcloud(fin\_clean, rot.per=0.5, random.order=TRUE,colors=brewer.pal(8, "Dark2")



**# Term document matrix**

**# converting unstructured data to structured format using TDM**

tdm <- TermDocumentMatrix(fin\_clean)

dtm <- t(tdm)

tdm <- as.matrix(tdm)

# Bar plot

w <- rowSums(tdm)

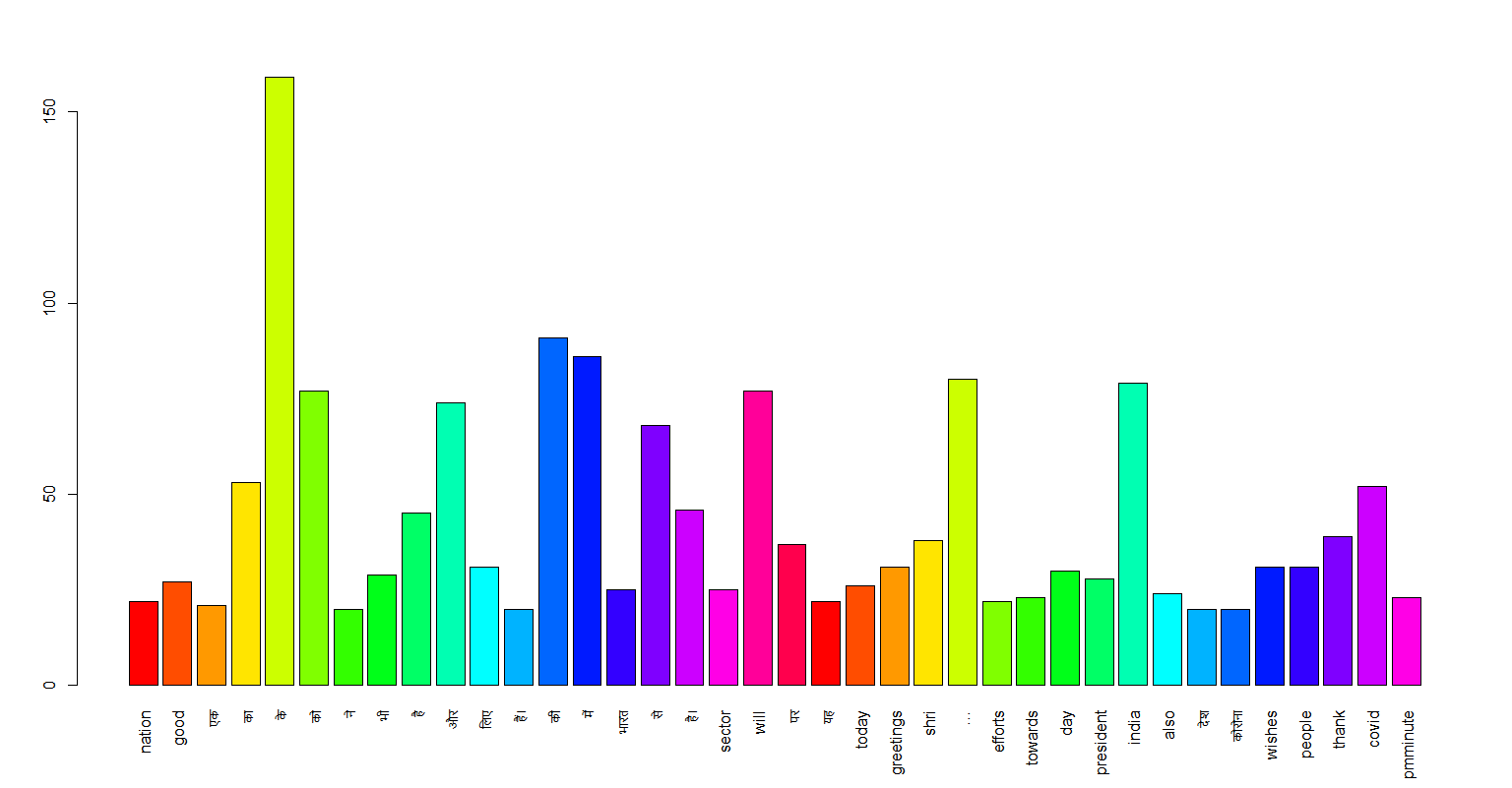
w

w\_sub <- subset(w, w >= 20)

w\_sub

#windows()

barplot(w\_sub, las=3, col = rainbow(20))

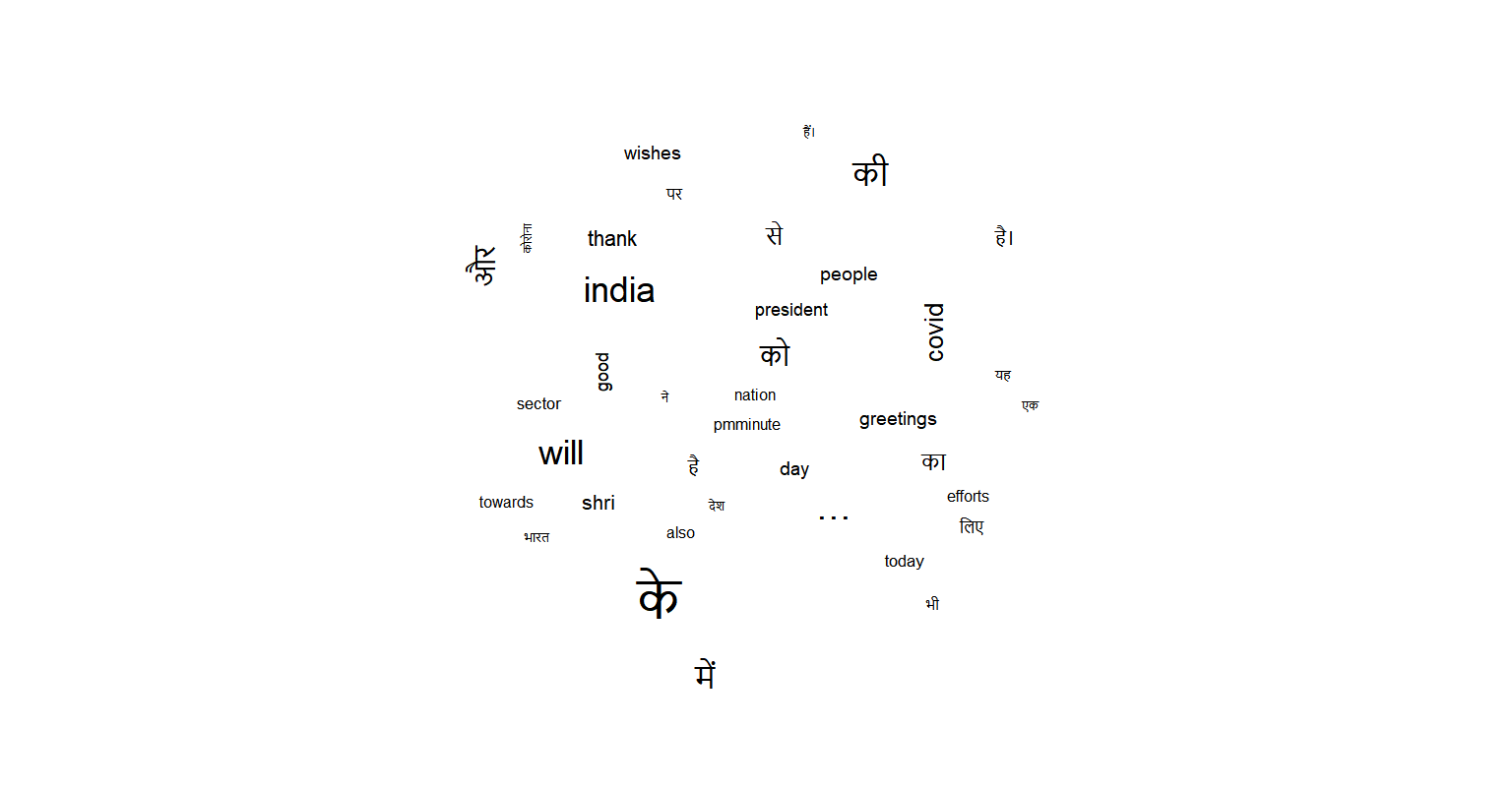


install.packages("wordcloud")

library(wordcloud)

windows()

wordcloud(words = names(w\_sub), freq = w\_sub) # wordcloud with only subset of words



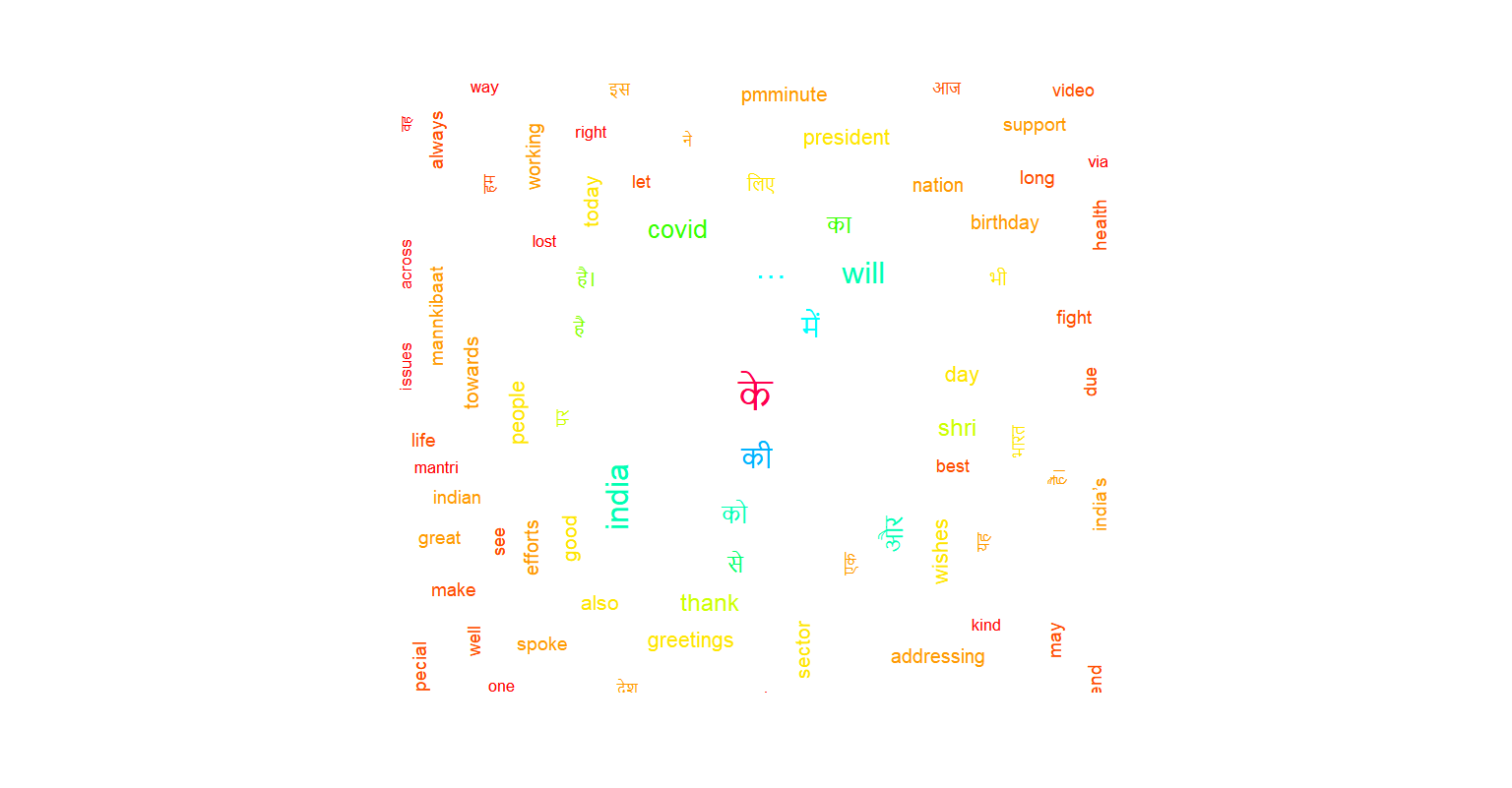
w\_sub1 <- sort(rowSums(tdm), decreasing = TRUE)

wordcloud(words = names(w\_sub1), freq = w\_sub1) # all words are considered



windows()

wordcloud(words = names(w\_sub1), freq = w\_sub1, random.order = F, colors = rainbow(20), scale=c(3,1), rot.per = 0.3)



**# lOADING +VE AND -VE dictonaries**

pos.words = scan(file.choose(), what="character", comment.char=";") # read-in positive-words.txt

neg.words = scan(file.choose(), what="character", comment.char=";") # read-in negative-words.txt

pos.words = c(pos.words,"wow", "kudos", "hurray") # including our own positive words to the existing list

**# Positive wordcloud**

pos.matches = match(names(w\_sub1), c(pos.words))

pos.matches = !is.na(pos.matches)

freq\_pos <- w\_sub1[pos.matches]

p\_names <- names(freq\_pos)

windows()

wordcloud(p\_names,freq\_pos,scale=c(4,1),colors = rainbow(20))



**# Negative wordcloud**

neg.matches = match(names(w\_sub1), c(neg.words))

neg.matches = !is.na(neg.matches)

freq\_neg <- w\_sub1[neg.matches]

n\_names <- names(freq\_neg)

windows()

wordcloud(n\_names,freq\_neg,scale=c(5,1),colors = brewer.pal(8,"Dark2"))



**################################## EMOTION MINING ########################**

install.packages("syuzhet")

library("syuzhet")

library(lubridate,ggplot2)

library(ggplot2)

library(scales)

library(dplyr)

library(reshape2)

x <- get\_nrc\_sentiment(fin\_txt)

head(x,n=5)

fin\_txt[4]

get\_nrc\_sentiment('happy')

get\_nrc\_sentiment('boring')

get\_sentiment('boring',method="afinn")

get\_sentiment('happy',method="afinn")

#each sentences by eight

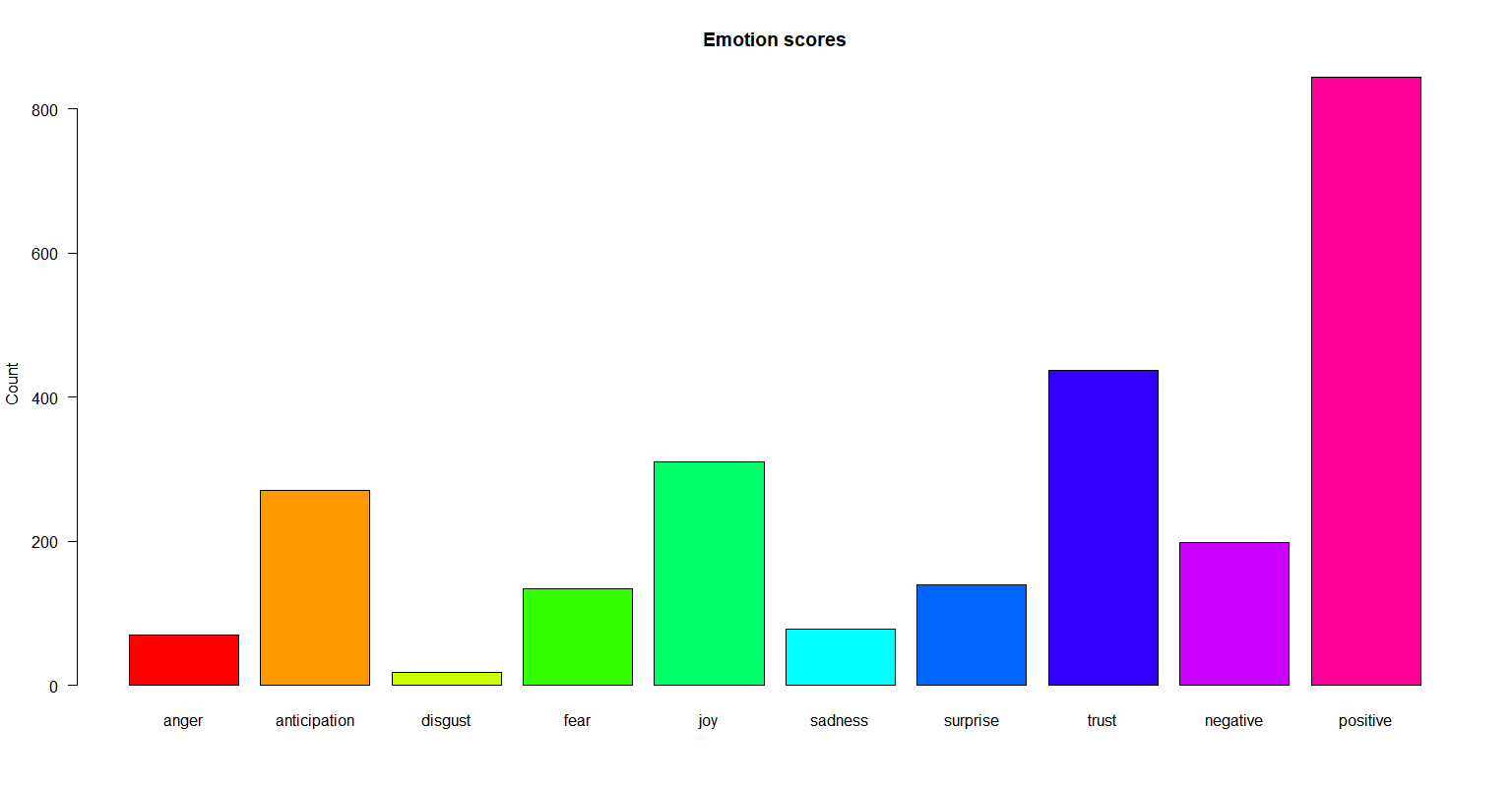
example<-get\_sentences(fin\_txt)

nrc\_data<-get\_nrc\_sentiment(example)

**# Bar plot for emotion mining**

windows()

barplot(colSums(nrc\_data), las = 1, col = rainbow(10), ylab = 'Count', main = 'Emotion scores')



sentiment\_vector<-get\_sentiment(example,method="bing")

sentiment\_afinn<-get\_sentiment(example,method="afinn")

sentiment\_nrc<-get\_sentiment(example,method="nrc")

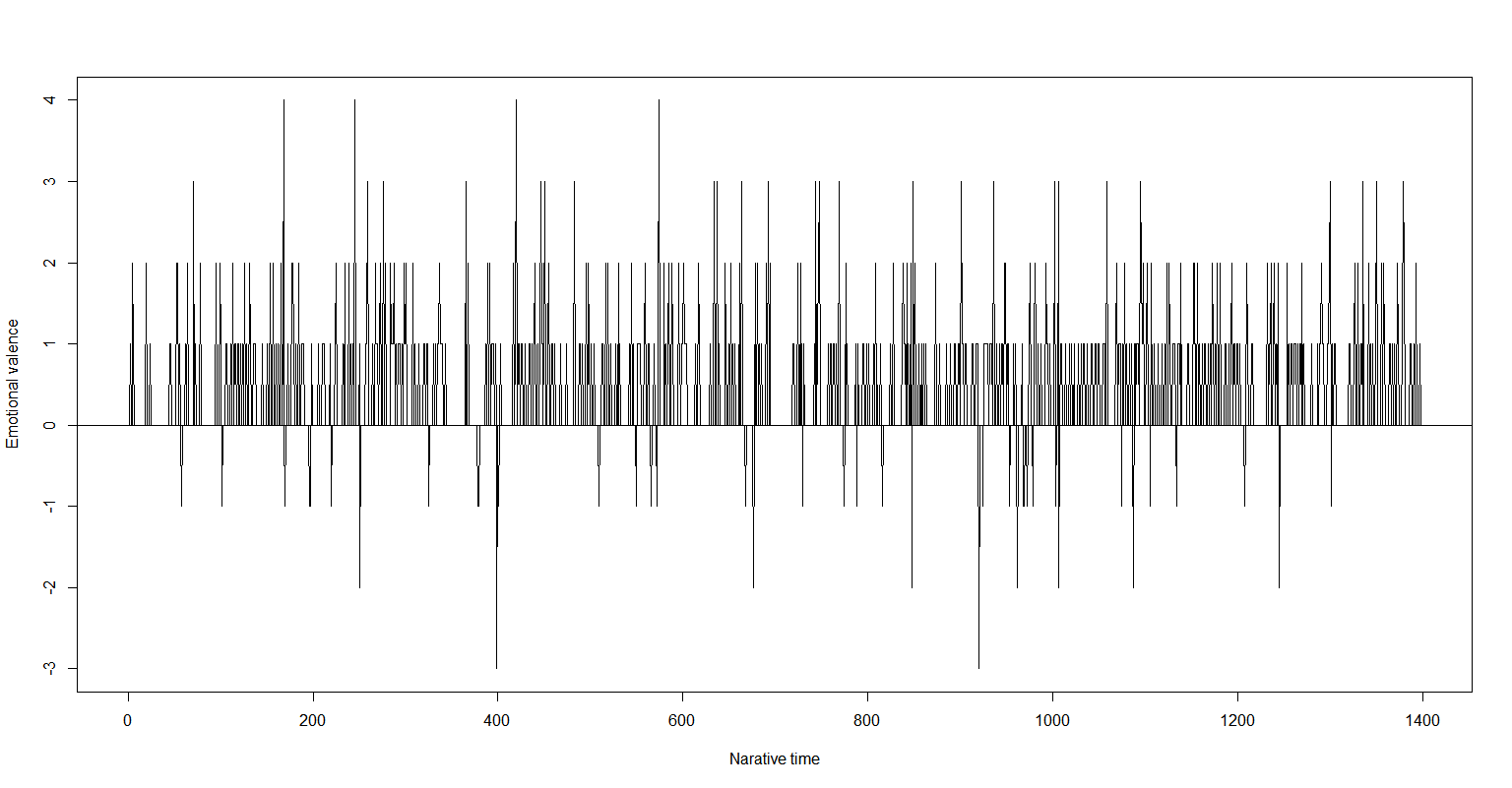
sum(sentiment\_afinn)

mean(sentiment\_afinn)

summary(sentiment\_afinn)

plot(sentiment\_vector,type='l',maim='Plot trajectory',xlab='Narative time',ylab='Emotional valence')

abline(h=0,color='red')



**##Shape smoothing and normalization using a Fourier based transformation and low pass filtering is achieved using the get\_transformed\_values function as shown below.**

ft\_values <- get\_transformed\_values(

sentiment\_vector,

low\_pass\_size = 3,

x\_reverse\_len = 100,

padding\_factor = 2,

scale\_vals = TRUE,

scale\_range = FALSE

)

plot(

ft\_values,

type ="l",

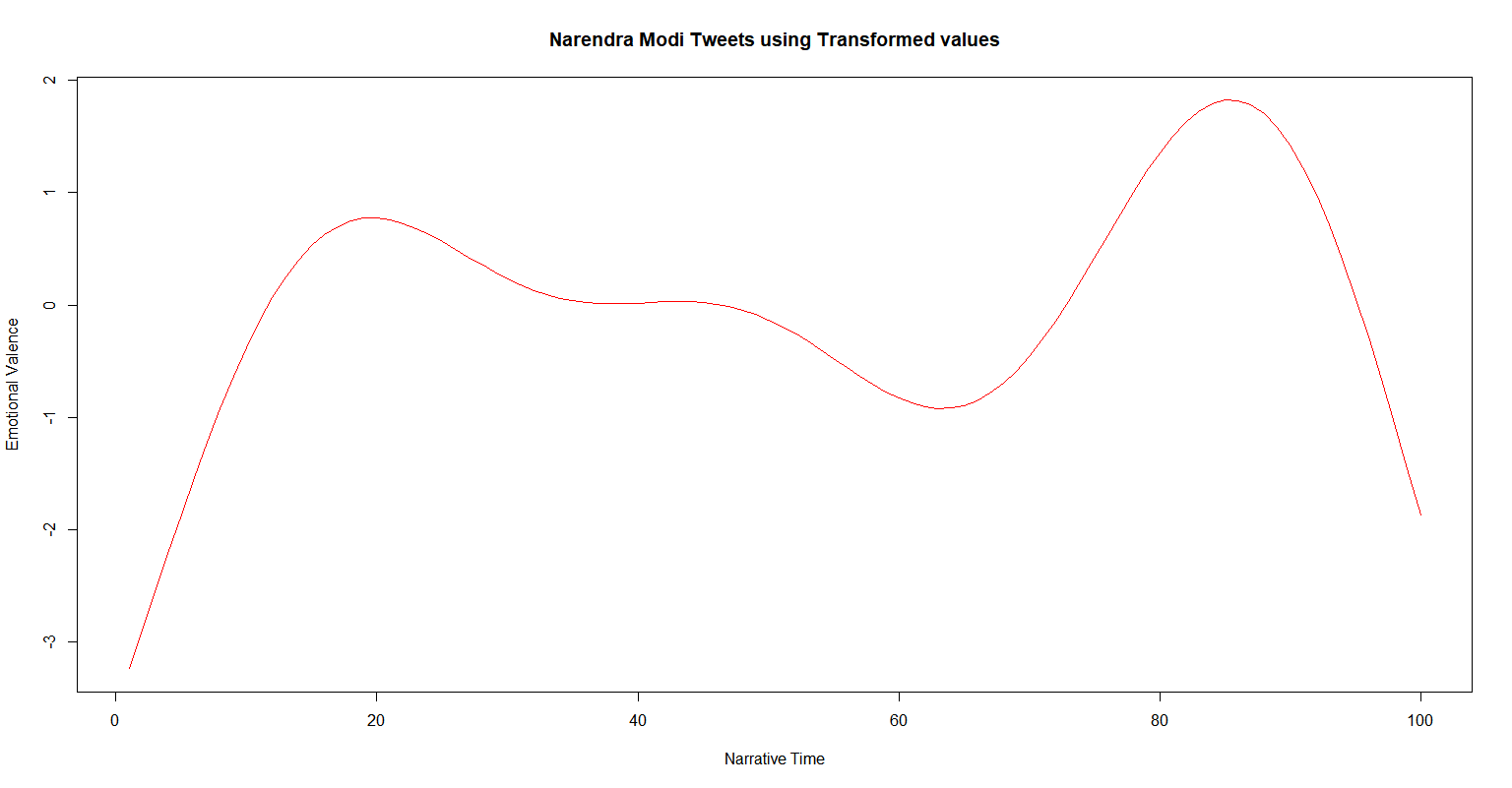
main ="Narendra Modi Tweets using Transformed values",

xlab = "Narrative Time",

ylab = "Emotional Valence",

col = "red"

)



#Most Negative and Positive reviews

negative<-example[which.min(sentiment\_vector)]

print(negative)

[1] "Her untimely and unfortunate demise left many saddened."

positive<-example[which.max(sentiment\_vector)]

print(positive)

[1] "#NationalSportsDay is a day to celebrate the remarkable achievements of all those exemplary sportspersons who have… https://t.co/v5ipU9ehfj"

99