Week 2

C. Kalinowski

3/1/2021

Task 2 - Exploratory Data Analysis

Exploratory Analysis and Word Frequencies

Based on tokenization, we can explore the words in the corpus. A possible function would be to create a clean tokenized list of vectors, unlist the contents, and return a frequency table as a data frame.

```
tokenFreq<-function(x){
      ## creates frequency table data frame of tokens from input text line x
      ## runs input x through cleanToken to get token list
      ## unlists token list to get single list
      ## creates table count of single list
      ## returns table as data frame

   words<-cleanToken(x)
   wordslist<-unlist(words)
   as.data.frame(table(wordslist))
}

sample1<-sampleReader("blogs",5)
sample1[4]
```

```
[1] "so anyways, i am going to share some home decor inspiration that i have been storing in my folder o
```

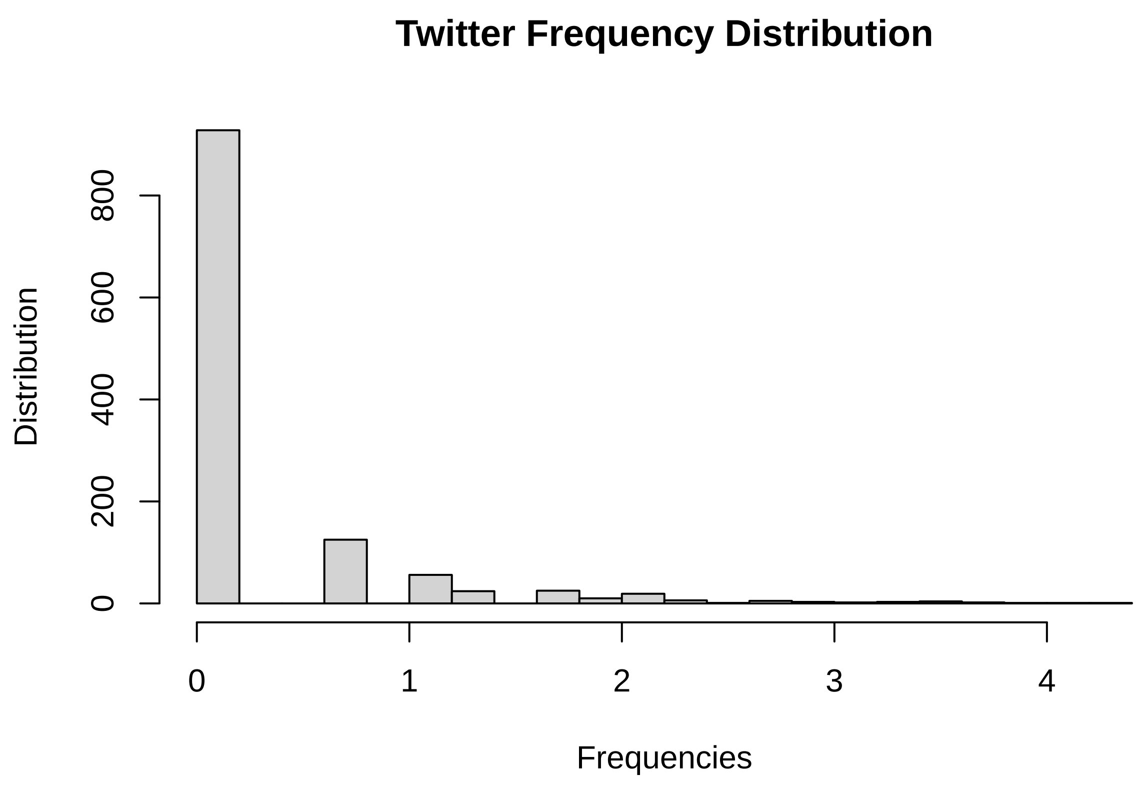
```
tokens1<-tokenFreq(sample1)
head(tokens1)
```

```
  wordslist Freq
1         a    3
2     after    2
3       all    3
4    almost    1
5      also    1
6        am    1
```

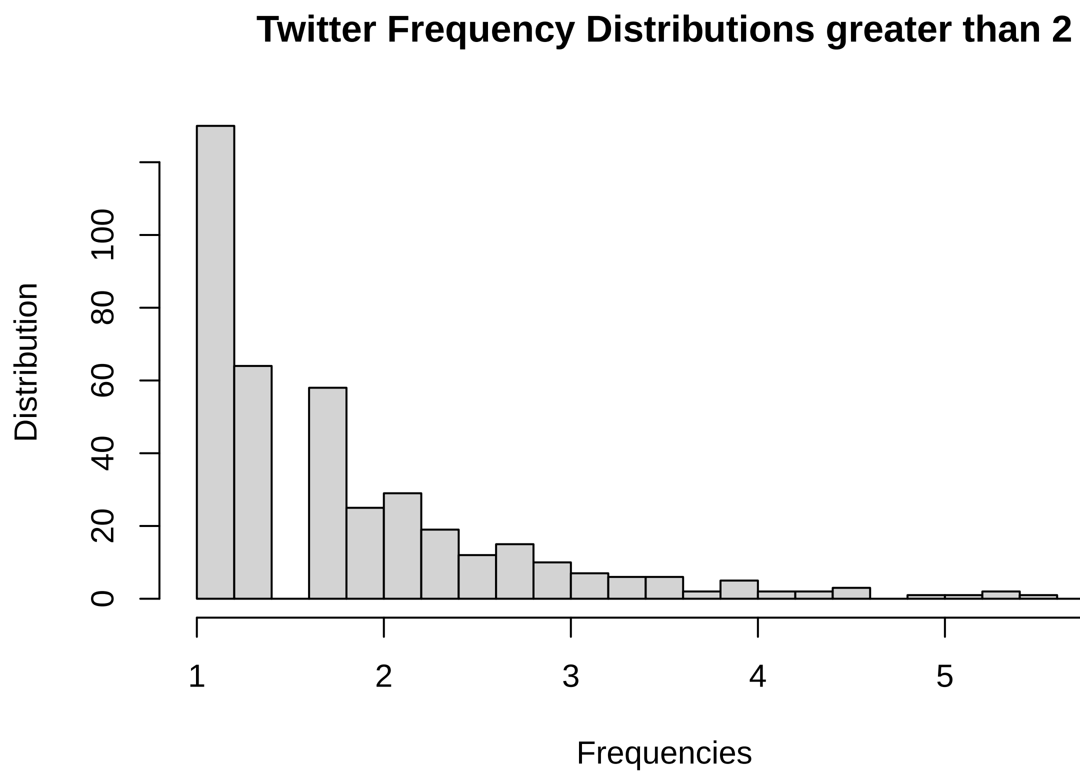
In each data set, a random sample of 200 lines has the following frequency distribution:

```
set.seed=322021
twitter<-sampleReader("twitter")
sampleTwitter<-sample(twitter,size=200,replace=F)
rm(twitter)
blog<-sampleReader("blogs")
sampleBlog<-sample(blog,size=200,replace=F)
rm(blog)
news<-sampleReader("news")
```

```r
sampleNews<-sample(news,size=200,replace=F)
rm(news)
twitterToken<-tokenFreq(sampleTwitter)
blogToken<-tokenFreq(sampleBlog)
newsToken<-tokenFreq(sampleNews)
hist(log(twitterToken$Freq),main="Twitter Frequency Distribution",xlab="Frequencies",
     ylab="Distribution",breaks=20)
```



```r
hist(log(subset(blogToken,Freq>2)$Freq),main="Twitter Frequency Distributions greater than 2",
     xlab="Frequencies",ylab="Distribution",breaks=20)
```



The words with the highest counts in each of the 3 English corpora are:

```
# BLOGS:
maxBlog<-subset(blogToken,Freq %in% head(sort(blogToken$Freq,decreasing=TRUE),10))
maxBlog[order(maxBlog$Freq,decreasing=TRUE),]
```

```
     wordslist Freq
2441       the  337
92         and  228
2490        to  209
1            a  189
1700        of  185
1178         I  161
1205        in  127
1250        is  114
2439      that   96
1255        it   94
```

```
# NEWS:
maxNews<-subset(newsToken,Freq %in% head(sort(newsToken$Freq,decreasing=TRUE),10))
maxNews[order(maxNews$Freq,decreasing=TRUE),]
```

```
     wordslist Freq
2349       the  307
1            a  162
2391        to  156
1612        of  147
101        and  139
```

```
1166        in  123
2030         s   81
914        for   73
2347      that   61
1222        is   57
```

```
# TWITTER:
maxTwitter<-subset(twitterToken,Freq %in% head(sort(twitterToken$Freq,decreasing=TRUE),10))
maxTwitter[order(maxTwitter$Freq,decreasing=TRUE),]
```

```
     wordslist Freq
1014       the   71
505          I   67
1052        to   64
40         and   48
1            a   47
1206       you   40
376        for   30
515         in   30
529         is   29
723         of   29
735         on   29
```

N-Gram Frequency

An easy way to create N-grams is to paste together token vectors.

```
ngrammer<-function(x,y){
      ## Creates list of y-grams from input tokenized list x
      ## Checks whether y is greater than length of x, else returns NULL
      ## Creates y vectors with stepped start and end points
      ## Vector 1 starts at 1 and ends at length of x - y
      ## Vector y starts at y and ends at end of x
      ## Binds vectors into data frame
      ## Pastes rows of data frame with space separator
      ## Returns list of pasted rows

   ngramMatrix<-NULL
   size<-length(x)
   if(size<=y){
      return()
   }

   ngramMatrix<-matrix(nrow=(size-y+1),ncol=0)
   for (i in (1:y)){
      tokenlist<-x[i:(size-y+i)]
      ngramMatrix<-cbind(ngramMatrix,tokenlist)
   }

   df_args <- c(as.data.frame(ngramMatrix), sep=" ")
   do.call(paste, df_args)
}


sample1[4]
```

```
[1] "so anyways, i am going to share some home decor inspiration that i have been storing in my folder
```

```
head(ngrammer(cleanToken(sample1[4]),4),10)
```

```
 [1] "so anyways i am"             "anyways i am going"
 [3] "i am going to"               "am going to share"
 [5] "going to share some"         "to share some home"
 [7] "share some home decor"       "some home decor inspiration"
 [9] "home decor inspiration that" "decor inspiration that i"
```

Task 3 - Modeling

The first task for modeling is creating ngram frequencies from the test sets and assigning them probabilities. Each of the 3 English corpora were split 60-20-20 into training, validation, and test sets.

```
ngramtable<-function(x,y,z=TRUE){
      ## Creates ngram frequency table of n-1 as rows and last word as column
      ## x is tokenized list, y is ngram number
      ## If y>2, send x to ngrammer function with y-1 for row generator
      ## If y=2, create vector using x from 1 to end-1 as first word generator
      ## Create vector using x from y to end as last word generator
      ## Create data frame with ngrammer or first word as first column
      ## and last word as second column
      ## If z, create and return frequency table of ngrammer by last word
      ## If not z, return data frame

   if(y>2){
      a<-ngrammer(x,(y-1))
   }
   else{
      a<-x[1:(length(x)-1)]
   }
   b<-x[y:(length(x))]
   df<-cbind(a[1:length(b)],b)
   if(z){
   table(df[,1],df[,2])
   }
   else{df}
}
```

For multi-line corpora, using lapply gets the correct output

```
token<-lapply(trainTwitter[1:2],cleanToken)

grams<-lapply(token,ngrammer,y=4)

gramDF<-lapply(token[1:2],ngramtable,y=4,z=FALSE)

gramList<-do.call("rbind",gramDF)

gramsFreqSmooth<-table(gramList[,1],gramList[,2])+1

gramsFreqSmooth[1:6,1:6]
```

```
                   A D don fun Going I
```

```
  A LOT D          1 1   1   1     1 1
  D Played Lazer   1 1   1   1     1 1
  D Ughh Going     1 1   1   1     1 1
  decided its more 1 1   1   2     1 1
  fun if I         1 1   2   1     1 1
  Going To Sleep   1 1   1   1     1 1
```

Frequencies

Unigrams

```
# Clean and tokenize the twitter data
twitterClean<-lapply(trainTwitter,cleanToken)
# unlist the token list to create a single list
twitter1gramList<-unlist(twitterClean)
# create a frequency table
twitter1gramFreq<-table(twitter1gramList)
# create a probability table
twitter1gramProb<-round((twitter1gramFreq/length(twitter1gramList)*100),6)
```

Bigrams

```
library(dplyr)
# create 2gram dfs
twitter2grams<-lapply(twitterClean,ngramtable,y=2,z=FALSE)
# combine 2gram dfs
twitter2gramList<-do.call("rbind",twitter2grams)
twitter2gramList<-as.data.frame(twitter2gramList)
# create a frequency chart
twitter2gramFreq<-table(twitter2gramList[,1],twitter2gramList[,2])
```

******************normalize frequencies by unigrams*********** frequency of "a b" / frequency of a