Text Analysis with R: Steps.

By Anahita Sanandaji (please do not share without permission)

In this document we emphasize on those lines of codes that are necessary to be used for the Text Analysis in R.

Step 0: Install and load the required packages; Setting up working directory and Read data from website or file (e.g. a .txt file)

```
install.packages("tm")
install.packages("SnowballC")
install.packages("wordcloud")

library(tm)
library(SnowballC)
library(wordcloud)
```

If you want to read data from website:

```
dataFromWeb <- readLines("http://cob.ohio.edu/anahitas/MIS4580/dataAnalysis.txt", warn= FALSE)
```

If you want to read data from a .txt file saved in your working directory.

```
dataFromFile <- readLines("HarryPotter.txt", warn= FALSE)
```

Step 1: Do some text pre-processing and data cleaning:

- Remove HTML codes
- Remove all punctuation marks
- Remove all extra white space characters
- Convert all data to lowercase, so that words like “read” and “Read” are considered the same word for analysis

```
#=========================================================================
# Step 1: Do some text pre-processing and data cleaning:
#=========================================================================

myCleanedData <- dataFromFile

#Remove HTML codes
myCleanedData <- gsub("<.*?>", "", myCleanedData)

#Remove all punctuation marks
myCleanedData <- gsub("[[:punct:]]","", myCleanedData)

#Remove all extra white space characters
myCleanedData <- gsub("\\s+", " ", myCleanedData)
# NOTe: or use myCleanedData <- gsub("[[:space:]]"," ", myCleanedData)

#Convert all data to lowercase
myCleanedData <- tolower(myCleanedData)
```

Step 2: Create a Corpus and Use VectorSource Function:

- To prepare your data to be converted into a corpus.
- Then use the VCorpus function to take that data and turn it into a required corpus data structure to create a TermDocumentMatrix.

```
#=============================================================================================
# Step 2: Use a VectorSource function to prepare your data to be converted into a corpus:
#=============================================================================================
#install.packages("tm") #may needed in VDI each time
#library(tm)

### Load the data as a corpus
myCorpusData <- VCorpus(VectorSource((myCleanedData)))
myCorpusData[[1]]$content
```

Step 3: Use tm_map() function to clean and prepare the text:

- **Remove stop words:** stop words are common words, that does not add much value to the text, and are preferred to be filtered out in a text so statistics concerning them will not be calculated. This allows for a better inspection of the more of the interesting words. (Examples, I, we, what, on, in, …)
- **Stem the data:** Technique to reduce the inflected words to their basic forms. So, the word like 'play', 'plays', 'playing', 'played' will all become 'play' after the 'stemming.
    - **Note:** I may ask you not to stem the document. So be careful!

```
#=============================================================================================
#Step 3: Use tm_map function to clean and prepare the text:
#=============================================================================================

## Remove stop words:
myMainCleanData <- myCorpusData
myMainCleanData <- tm_map(myMainCleanData,removeWords, stopwords("english"))

### stemming
##install.packages("SnowballC") #may needed
##library(SnowballC)
#stem the data then check it out
myMainCleanData <- tm_map(myMainCleanData,stemDocument)
```

Step 4: Create "term document matrix":

- Use TermDocumentMatrix, as a function that will take the Corpus and create your matrix as a list object where the document names are the column names, and the terms are the row names.

```
#=============================================================================================
#Step 4: Create "term document matrix", use TermDocumentMatrix:
#=============================================================================================
myTDMData <- TermDocumentMatrix(myMainCleanData)
```

Step 5: Do Analysis

Example:

- Find Terms that occurs at least "???" times
- Find words associated with "???" based on a given correlation of ??%

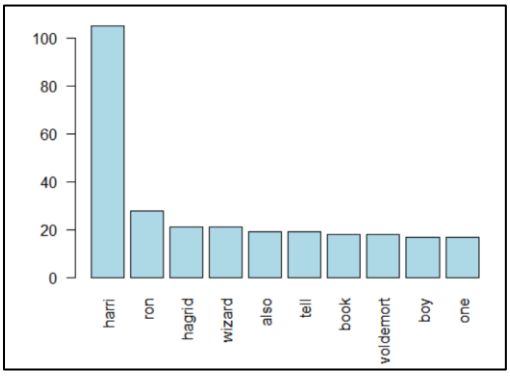
- Calculate the frequency of words and create word cloud based on those frequencies.

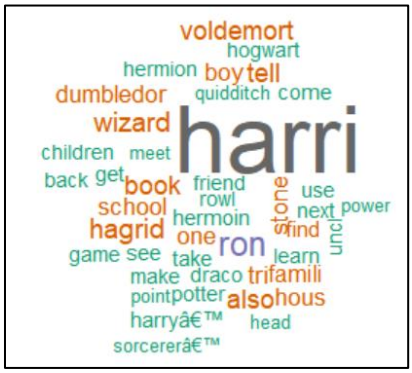
Sample analysis

```
#================================================================================
#Step 5: Do Actual Text Analysis:
#================================================================================

## To find terms that occur at least 4 times ( the lower frequency bound),
# then we can use the findFreqTerms().
findFreqTerms(myTDMData,lowfreq = 4, highfreq = Inf)
findFreqTerms(myTDMData, 4, Inf) #same as above

## we can use the FindAssocs function to find words which associate together
findAssocs(x=myTDMData, term = "wizard",corlimit = 0.5)
```

Visualization

```
#===Visualization================================================================
#my need to install.packages("wordcloud")
library(wordcloud)

# calculate the frequency of words
freqOfTerms <- sort(rowSums(as.matrix(myTDMData)),decreasing = TRUE)

#basic barplot example
barplot(freqOfTerms[1:10],col ="lightblue",las = 2)

# get list of words from heading
namesFromData <- names(freqOfTerms)

#Wordcloud: show the importance of words based on frequency with a word cloud
wordcloud(namesFromData, freqOfTerms, min.freq = 10,
          max.words = 200,colors=brewer.pal(8, "Dark2")) #by A.Sanandaji
```





How to create a colorful word cloud?

You can use this line of code to produce a colorful wordcloud if you wish:

wordcloud(namesFromData, freqOfTerms, min.freq = ??, max.words = ??, ***colors = brewer.pal(8, "Dark2"))***

See next page ➔

Side note (Using tm_map() to combine steps 1,2,3)— This is Optional:

You can use tm_map() to do some data cleaning (Remove all punctuation marks, Remove all extra white space characters, Convert all data to lowercase) that you did in step 1 (instead of using gsub() or tolower()) and combine it directly with step 2 and step 3. If you use this approach you do not need to conduct the previous steps 1-3.

Here is the alternative code that combines steps 1,2, and 3:

```
#================================================================================
#Step 1 and 2 and 3 combined (Alternate way to clean data using tm_map()
#================================================================================
library(tm)
library(SnowballC)

myCleanedData=dataFromFile

#Remove HTML codes
myCleanedData = gsub("<.*?>", "", myCleanedData)

#Load the data as a corpus
myCorpusData=VCorpus(VectorSource((myCleanedData)))
myMainCleanData=myCorpusData

#Convert all data to lowercase
myMainCleanData=tm_map(myMainCleanData,content_transformer(tolower))

#Remove all punctuation marks
myMainCleanData=tm_map(myMainCleanData,removePunctuation)

#Remove all extra white space characters
myMainCleanData=tm_map(myMainCleanData,stripWhitespace)

# remove stopwords
myMainCleanData=tm_map(myMainCleanData,removeWords, stopwords("English"))

#stem data
myMainCleanData=tm_map(myMainCleanData,stemDocument)

## end of *** Step 1,2 and 3 Combined
#================================================================================
```