Module 5 Text Mining Practice

ALY 6040

Jeff Hackmeister

2024-05-05

Introduction

To demonstrate the ability within in R to extract meaningful data from text resources, we'll be using the famous *I Have a Dream* speech, delivered by Martin Luther King, Jr. on August 28th, 1963 in Washington, DC.

To conduct the analysis, we will utilize several packages from R.

```
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
```

The tm package allows for text mining through the use of memory objects called corpora. The SnowballC package allows for word stemming, which collapses words to their root word for better data analysis and comparison. Finally wordcloud and RColorBrewer are visualization packages that will help create visual representations of the analysis conducted.

Once our packages are installed and loaded, we can read in the text of the speech.

```
filePath <- "http://www.sthda.com/sthda/RDoc/example-files/martin-luther-king-i-have-a-dream-
text <- readLines(filePath)
```

Next, we'll convert the text into a corpus for use in analysis.

```
docs <- Corpus(VectorSource(text))

inspect(docs)
```

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 46

 [1]
 [2] And so even though we face the difficulties of today and tomorrow, I still have a dream
 [3]
 [4] I have a dream that one day this nation will rise up and live out the true meaning of it
 [5]
 [6] We hold these truths to be self-evident, that all men are created equal.
 [7]
 [8] I have a dream that one day on the red hills of Georgia, the sons of former slaves and t
 [9]
```

[10] I have a dream that one day even the state of Mississippi, a state sweltering with the h
[11]
[12] I have a dream that my four little children will one day live in a nation where they wil
[13]
[14] I have a dream today!
[15]
[16] I have a dream that one day, down in Alabama, with its vicious racists, with its governo
[17]
[18] I have a dream today!
[19]
[20] I have a dream that one day every valley shall be exalted, and every hill and mountain s
[21]
[22] This is our hope, and this is the faith that I go back to the South with.
[23]
[24] With this faith, we will be able to hew out of the mountain of despair a stone of hope.
[25]
[26] And this will be the day, this will be the day when all of God s children will be able t
[27]
[28] My country tis of thee, sweet land of liberty, of thee I sing.
[29] Land where my fathers died, land of the Pilgrim s pride,
[30] From every mountainside, let freedom ring!
[31] And if America is to be a great nation, this must become true.
[32] And so let freedom ring from the prodigious hilltops of New Hampshire.
[33] Let freedom ring from the mighty mountains of New York.
[34] Let freedom ring from the heightening Alleghenies of Pennsylvania.
[35] Let freedom ring from the snow-capped Rockies of Colorado.
[36] Let freedom ring from the curvaceous slopes of California.
[37]
[38] But not only that:
[39] Let freedom ring from Stone Mountain of Georgia.
[40] Let freedom ring from Lookout Mountain of Tennessee.
[41] Let freedom ring from every hill and molehill of Mississippi.
[42] From every mountainside, let freedom ring.
[43] And when this happens, when we allow freedom ring, when we let it ring from every villag
[44] Free at last! Free at last!
[45]
[46] Thank God Almighty, we are free at last!

Data Preparation

We now have the entirety of the speech available for analysis. As seen in the print out above, our corpus contains every word of the speech, and while preserving the text of a historically significant speech is important, many alterations can be made to prepare the test for analysis.

As with any data analysis, the next steps are to prepare and clean the data. We'll utilize the content_transformer function from tm to remove any special characters from the text data and replace them with spaces. This will make further analysis easier and more effective.

```
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "/")
docs <- tm_map(docs, toSpace, "@")
docs <- tm_map(docs, toSpace, "\\|")
```

Next, we will convert all text to lower case and remove any numbers from the text using the tm_map function.

```
# Convert the text to lower case
docs <- tm_map(docs, content_transformer(tolower))
# Remove numbers
docs <- tm_map(docs, removeNumbers)
```

Next, we'll use the tm package to remove what are referred to as stopwords, or filler words in English that have high usage but don't provide much analytic value.

```
docs <- tm_map(docs, removeWords, stopwords("english"))
```

We continue the cleaning process by removing punctuation and white spaces in the text.

```
# Remove punctuations
docs <- tm_map(docs, removePunctuation)
# Eliminate extra white spaces
docs <- tm_map(docs, stripWhitespace)
```

And finally, we'll reduce the remaining words in the text to their root by using the stemDocument call.

```
docs <- tm_map(docs, stemDocument)
```

Analysis

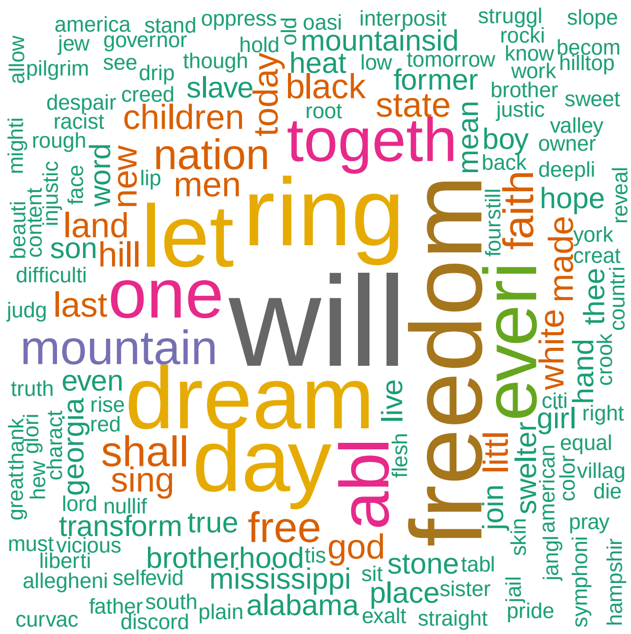
With our text cleaned and prepped, we can run a meaningful analysis. We will create a term document matrix which will show the frequency of each word in our text and we'll sort it in descending order so the most frequently used words will be at the top. This illustrates the importance of the previous cleaning techniques as words like "a" and "the" have been removed and by stemming the text, various expressions of a root word will be grouped together.

```
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 10)
```

```
           word freq
will       will   17
freedom freedom   13
ring       ring   12
dream     dream   11
day         day   11
let         let   11
everi     everi    9
one         one    8
abl         abl    8
togeth   togeth    7
```

Now that we have our matrix, we can generate a world cloud. We'll use setseed to ensure our work can be reproduced and then use the wordcloud function to create a word cloud where the most commonly used words are centered and larger that others. We'll also limit the cloud to the top 200 words and use rot.per at 0.35 to indicate that 35% of the words will be vertical, this keeps the cloud more concentrated. Finally, we'll use the RColorBrewer Dark2 color package to add colors for the most frequently appearing words in the speech.

```
set.seed(1234)
wordcloud(words = d$word, freq = d$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```



We can also use the findFreqTerms function to produce a list of words that appear at least a stated number of times. For example, we'll look at all words appearing at least 4 times.

```
findFreqTerms(dtm, lowfreq = 4)
```

```
 [1] "dream"    "day"      "nation"   "one"      "will"     "abl"
 [7] "togeth"   "freedom"  "everi"    "mountain" "shall"    "faith"
[13] "free"     "let"      "ring"
```

We can also look at corelations between words by using the findAssocs function. Here we will limit the results to works with a correlation of at least 0.3.

```
findAssocs(dtm, terms = "freedom", corlimit = 0.3)
```

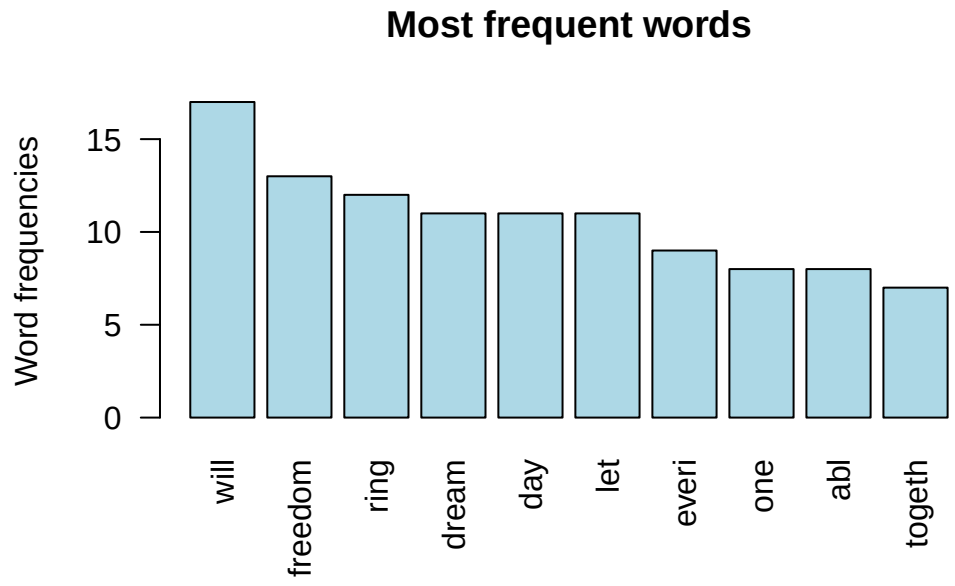
```
$freedom
        let         ring     mountain mississippi    transform        stone
       0.89         0.86         0.40         0.34         0.34         0.34
mountainsid        state        everi
       0.34         0.32         0.32
```

Finally, we return to our matrix of most frequently used words and create a bar chart of the top 10.

```r
head(d, 10)
```

```
           word freq
will       will   17
freedom freedom   13
ring       ring   12
dream     dream   11
day         day   11
let         let   11
everi     everi    9
one         one    8
abl         abl    8
togeth   togeth    7
```

```r
barplot(d[1:10,]$freq, las = 2, names.arg = d[1:10,]$word,
        col ="lightblue", main ="Most frequent words",
        ylab = "Word frequencies")
```



Conclusion

Text mining reveals the hidden patterns within narrative data that quantitative analysis alone cannot capture. In this work we've demonstrated how unstructured textual information can be transformed into actionable insights through systematic processing and visualization.

Utilizing both the tm and SnowballC packages for preprocessing and cleaning, we were able to find patterns in word choice in the *I Have a Dream* speech. While the wordcloud visualizations provided an intuitive representation of concept prominence.

Moving forward, these techniques could be further expanded by implementing more advanced natural language processing methods such as topic modeling or sentiment analysis to extract even deeper insights from narrative data. Text mining thus serves not just as a supplementary tool, but as an essential component in comprehensive data analysis frameworks where human expression adds critical context to quantitative findings.

References

[1] *RColorBrewer package - RDocumentation*. (2022). Rdocumentation.org.
https://www.rdocumentation.org/packages/RColorBrewer/versions/1.1-3

[2] *SnowballC package - RDocumentation*. (2023). Rdocumentation.org.
https://www.rdocumentation.org/packages/SnowballC/versions/0.7.1

[3] *tm package - RDocumentation*. (2025). Rdocumentation.org.
https://www.rdocumentation.org/packages/tm/versions/0.7-16

[4] *wordcloud package - RDocumentation*. (2018). Rdocumentation.org.
https://www.rdocumentation.org/packages/wordcloud/versions/2.6