Rayanna Harduarsingh

November 4th, 2020

IST 687

Module 11 Notes

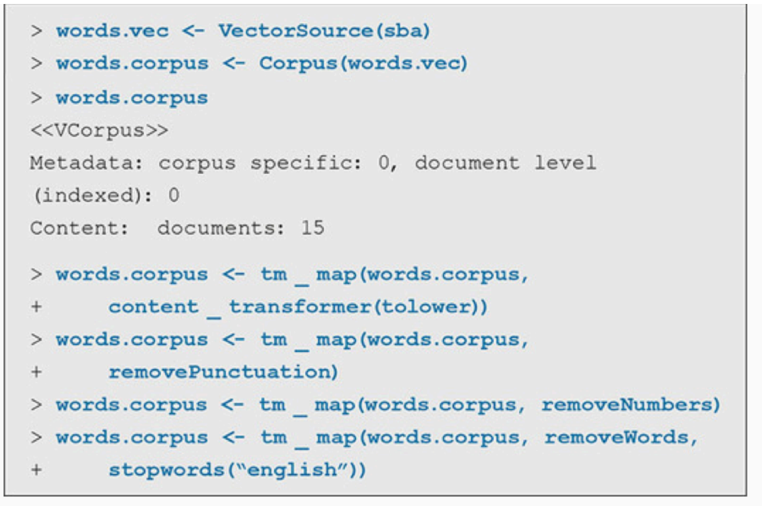
Text Mining

- Looking for unexpected patterns in large data sets
- Focuses on statistical approaches and strategies such as counting word frequencies
- Natural Language Processing (NLP)
  - Understand how to program machines to make sense of human language
  - Use linguistics to break text into its grammatical pieces such as nouns and verbs

- Word Cloud generated by R
- Extract text from a speech or any other text
- Build a term-ducment matrix
- Find frequent words and associations form the matrix
- Presents frequently occurring words

- **Text Mining packages in R:**
    - TM- Provides functions for text mining
    - Wordcloud- visualizes results
  - **Build a corpus**
    - A corpus is a ‘bag of words’.
    - Coerce text file vector insta a custom “Class” provided by the tm package called a “Corpus”.

        - The Corpus Class defines the most fundamental object that text miners care about, a corpus containing a collection of documents
    - Text Transformations (Four Transformations)
        - Making all of the letters lowercase
        - Removing the punctuation
        - Removing the numbers
        - Taking out the “stop” words
            - Ex: the, a, at
- Text Transformations in R



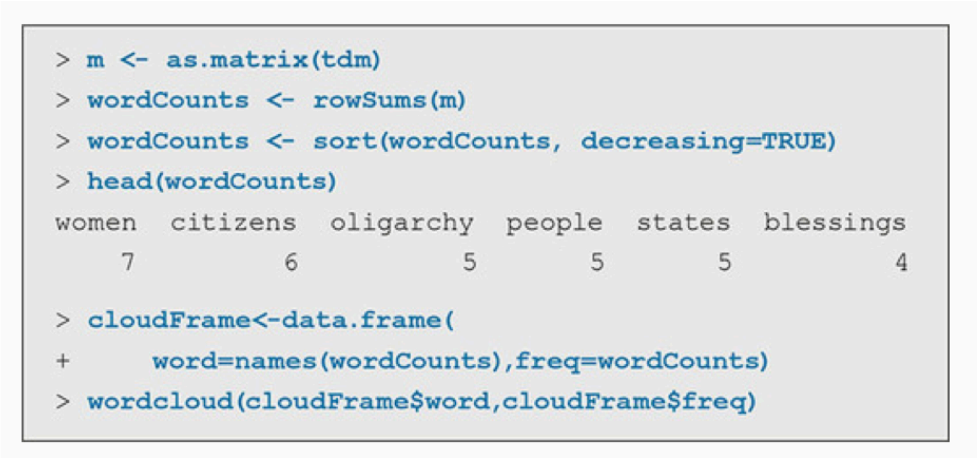
```
> words.vec <- VectorSource(sba)
> words.corpus <- Corpus(words.vec)
> words.corpus
<<VCorpus>>
Metadata: corpus specific: 0, document level
(indexed): 0
Content:  documents: 15

> words.corpus <- tm _ map(words.corpus,
+     content _ transformer(tolower))
> words.corpus <- tm _ map(words.corpus,
+     removePunctuation)
> words.corpus <- tm _ map(words.corpus, removeNumbers)
> words.corpus <- tm _ map(words.corpus, removeWords,
+     stopwords(“english”))
```

- A Term-Document Matrix:
    - Rectangular data structure with terms (words) as the rows and documents as the columns
    - A term may be a single word, like biology, or it could be a compound word, like data analysis
    - Creating a Term Document Matrix in R:

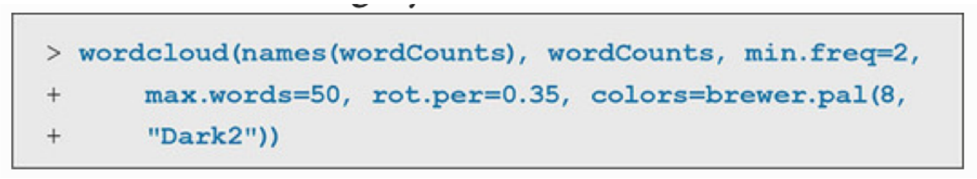
- How useful are word clouds? When are they appropriate to use?
    - Words clouds are mainly useful when you are dealing with huge amounts of documents that contain a lot of text. They are useful to use when you want to get the

big idea of those documents, what is being said, and what is being repeated, instead of single-handily going through each of them. It can list with topics/words are listed the most. For example, if you have a survey of about 1000 people pertaining to the most common snacks among teenagers and are analyzing those results, a word cloud might be useful to compress those results. You can visualize which snack was said the most, for example like Oreos. That word can pop up bigger and we can see it is most likely the most popular snack among teenagers.

Creating Word Clouds in R:



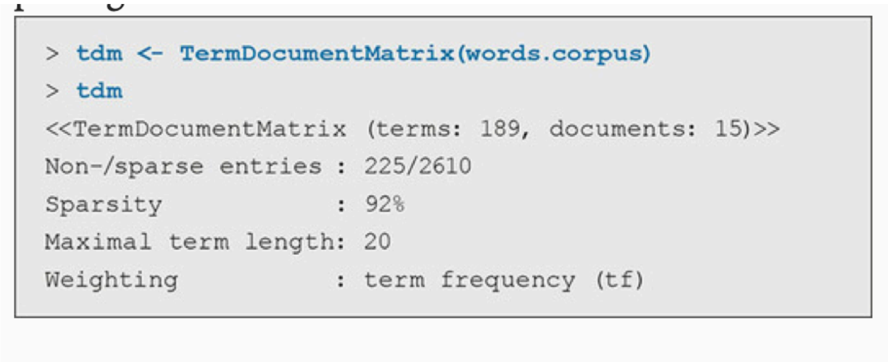
```
> m <- as.matrix(tdm)
> wordCounts <- rowSums(m)
> wordCounts <- sort(wordCounts, decreasing=TRUE)
> head(wordCounts)
women  citizens  oligarchy  people  states  blessings
    7         6          5       5       5          4

> cloudFrame<-data.frame(
+     word=names(wordCounts),freq=wordCounts)
> wordcloud(cloudFrame$word,cloudFrame$freq)
```



```
> wordcloud(names(wordCounts), wordCounts, min.freq=2,
+     max.words=50, rot.per=0.35, colors=brewer.pal(8,
+     "Dark2"))
```

- Sentiment Analysis
    - Conceptual Methodology
        - Load positive and negative words lists
            - Count positive words and negative words
            - Computer the ratio of positive to negative words
- Question: Does this truly measure sentiment? Where could it go wrong?

- One issue where it could be inaccurate is similar to the saying, wrong place wrong timing or slang. There could be negative words that are meant to be positive and it depends on the context. For example, in the discussion, ridiculous good is positive but ridiculous is a negative word. So sometimes, it could be inaccurate as negative words could actually be positive.



```
> tdm <- TermDocumentMatrix(words.corpus)
> tdm
<<TermDocumentMatrix (terms: 189, documents: 15)>>
Non-/sparse entries : 225/2610
Sparsity            : 92%
Maximal term length: 20
Weighting           : term frequency (tf)
```