Visualizing Text Data Using Word Clouds

Contents

Introduction **1**

Exercices **2**

Problem 1 : Preparing the Data 2

Problem 1.1 2

Problem 1.2 3

Problem 2 : Building a Word Cloud 3

Problem 2.1 3

Problem 2.2 3

Problem 2.3 4

Problem 2.4 4

Problem 3 : Size and Color 5

Problem 3.1 6

Problem 3.2 10

Problem 3.3 11

Problem 3.4 11

Problem 3.5 11

Problem 4 : Selecting a Color Palette 11

Problem 4.1 11

Problem 4.2 13

Problem 4.3 13

Introduction

Earlier in the course, we used text analytics as a predictive tool, using word frequencies as independent variables in our models. However, sometimes our goal is to understand commonly occurring topics in text data instead of to predict the value of some dependent variable. In such cases, word clouds can be a visually appealing way to display the most frequent words in a body of text.

A word cloud arranges the most common words in some text, using size to indicate the frequency of a word. For instance, this is a word cloud for the complete works of Shakespeare, removing English stopwords:

Shakespeare word cloud

While we could generate word clouds using free generators available on the Internet, we will have more flexibility and control over the process if we do so in R. We will visualize the text of tweets about Apple, a dataset we used earlier in the course. As a reminder, this dataset (which can be downloaded from tweets.csv has the following variables:

- **Tweet** : the text of the tweet
- **Avg** : the sentiment of the tweet, as assigned by users of Amazon Mechanical Turk. The score ranges on a scale from -2 to 2, where 2 means highly positive sentiment, -2 means highly negative sentiment, and 0 means neutral sentiment.

Exercices

Problem 1 : Preparing the Data

Problem 1.1 Download the dataset "tweets.csv", and load it into a data frame called "tweets" using the read.csv() function, remembering to use stringsAsFactors=FALSE when loading the data.

Next, perform the following pre-processing tasks (like we did in Unit 5), noting that we don't stem the words in the document or remove sparse terms:

1) Create a corpus using the Tweet variable
2) Convert the corpus to lowercase
3) Remove punctuation from the corpus
4) Remove all English-language stopwords
5) Build a document-term matrix out of the corpus
6) Convert the document-term matrix to a data frame called allTweets

```
## 'data.frame':    1181 obs. of  2 variables:
##  $ Tweet: chr  "I have to say, Apple has by far the best customer care service I have ever received!
##  $ Avg  : num  2 2 1.8 1.8 1.8 1.8 1.8 1.6 1.6 1.6 ...
```

```
## <<DocumentTermMatrix (documents: 1181, terms: 3780)>>
## Non-/sparse entries: 10273/4453907
## Sparsity           : 100%
## Maximal term length: 115
## Weighting          : term frequency (tf)
```

How many unique words are there across all the documents?

Answer : 3780

Explanation :
We can complete the pre-processing steps with the following commands: library(tm)

From the commands "frequencies", "str(allTweets)" or "ncol(allTweets)", we can read that there are 3780 unique words across all the tweets.

Problem 1.2 Although we typically stem words during the text preprocessing step, we did not do so here. **What is the most compelling rationale for skipping this step when visualizing text data?**

Answer :

1. It avoids the computational burden of stemming
2. **It will be easier to read and understand the word cloud if it includes full words instead of just the word stems**
3. We would not be able to create a word cloud if we stemmed the document

Explanation :
We want to create an interpretable display of a document's contents, and our results will be easier to read if they include full words instead of just the stems. Stemming has relatively minor computational burden, and we certainly could create a word cloud with a stemmed document.

Problem 2 : Building a Word Cloud

Problem 2.1 Install and load the "wordcloud" package, which is needed to build word clouds.

As we can read from ?wordcloud, we will need to provide the function with a vector of words and a vector of word frequencies. Which function can we apply to allTweets to get a vector of the words in our dataset, which we'll pass as the first argument to wordcloud()?

Answer :

1. str
2. rownames
3. **colnames**

Explanation:
Each tweet represents a row of allTweets, and each word represents a column. We need the names of all the columns of allTweets, which is returned by colnames(allTweets). While str(allTweets) displays the names of the variables along with other information, it doesn't return a vector that we can use as the first argument to wordcloud().

Problem 2.2 Which function should we apply to allTweets to obtain the frequency of each word across all tweets?

Answer:

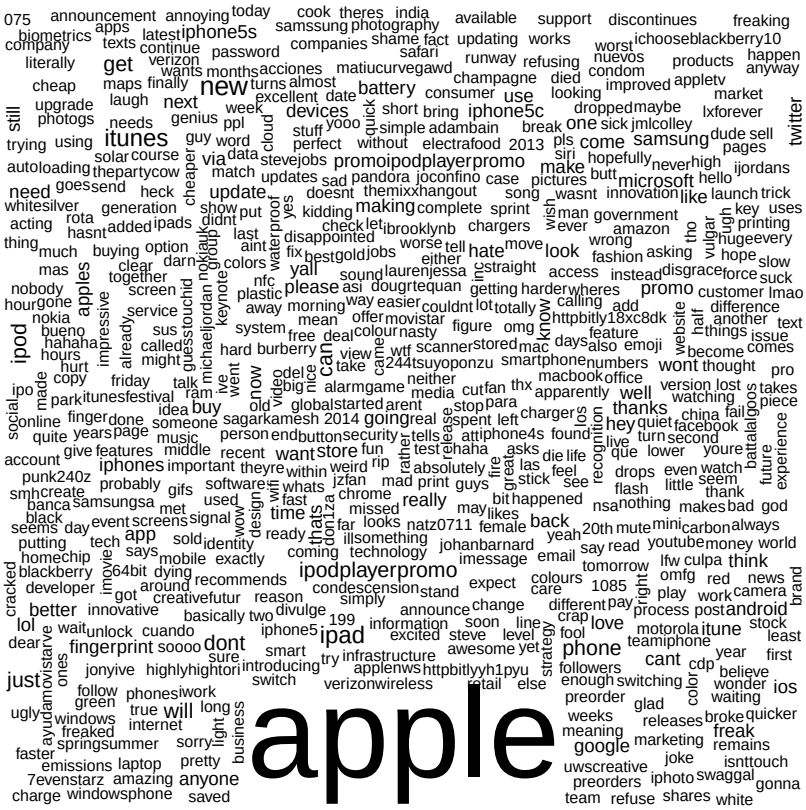
1. **colSums**
2. rowSums
3. sum

Explanation:
Each tweet represents a row in allTweets, and each word represents a column. Therefore, we need to access the sums of each column in allTweets, which is returned by colSums(allTweets).

Problem 2.3 Use allTweets to build a word cloud. Make sure to check out the help page for wordcloud if you are not sure how to do this.

Because we are plotting a large number of words, you might get warnings that some of the words could not be fit on the page and were therefore not plotted – this is especially likely if you are using a smaller screen. You can address these warnings by plotting the words smaller. From ?wordcloud, we can see that the "scale" parameter controls the sizes of the plotted words. By default, the sizes range from 4 for the most frequent words to 0.5 for the least frequent, as denoted by the parameter "scale=c(4, 0.5)". We could obtain a much smaller plot with, for instance, parameter "scale=c(2, 0.25)".

What is the most common word across all the tweets (it will be the largest in the outputted word cloud)? Please type the word exactly how you see it in the word cloud. The most frequent word might not be printed if you got a warning about words being cut off – if this happened, be sure to follow the instructions in the paragraph above.



Answer : Apple

Explanation:
We can output the word cloud with:

For smaller words, we could have used:

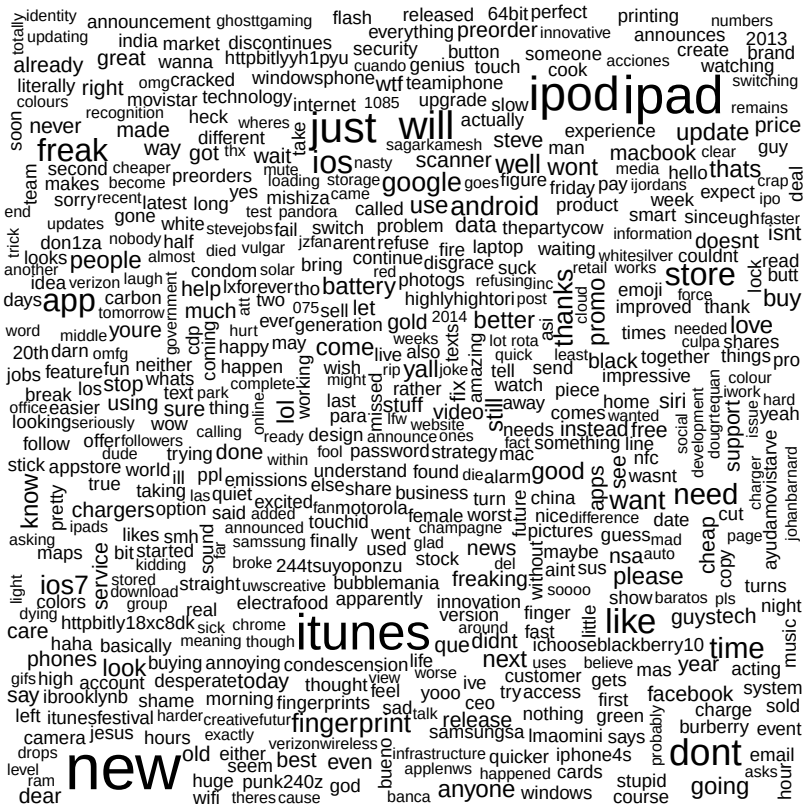
"apple" is by far the largest, and therefore most common, word.

Problem 2.4 In the previous subproblem, we noted that there is one word with a much higher frequency than the other words. Repeat the steps to load and pre-process the corpus, this time removing the most

frequent word in addition to all elements of stopwords("english") in the call to tm_map with removeWords. For a refresher on how to remove this additional word, see the Twitter text analytics lecture.

Replace allTweets with the document-term matrix of this new corpus – we will use this updated corpus for the remainder of the assignment.

Create a word cloud with the updated corpus. **What is the most common word in this new corpus (the largest word in the outputted word cloud)?** The most frequent word might not be printed if you got a warning about words being cut off – if this happened, be sure to follow the instructions in the previous problem.



Answer : iphone

Explanation:
We can do the specified update with the following commands:

For a much smaller plot, we could have used:

The most common (largest) word is now "iphone".

Problem 3 : Size and Color

So far, the word clouds we've built have not been too visually appealing – they are crowded by having too many words displayed, and they don't take advantage of color. One important step to building visually appealing visualizations is to experiment with the parameters available, which in this case can be viewed by

typing ?wordcloud in your R console. In this problem, you should look through the help page and experiment with different parameters to answer the questions.

Below are four word clouds, each of which uses different parameter settings in the call to the wordcloud() function:

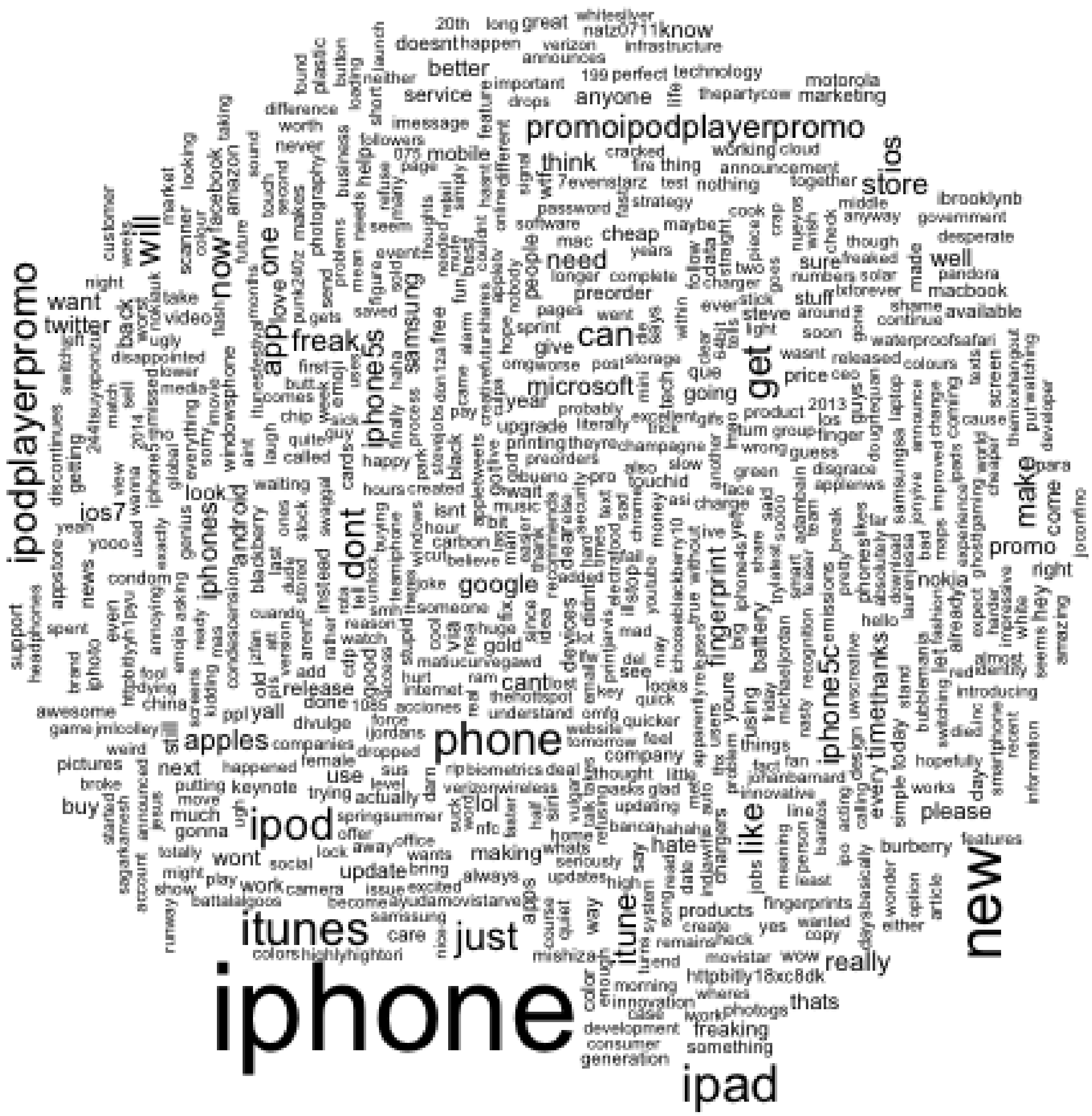


Figure 1: Word Cloud A

We will refer to these four word clouds in the next several problems.

Problem 3.1 Which word cloud is based only on the negative tweets (tweets with Avg value -1 or less)?

Answer:

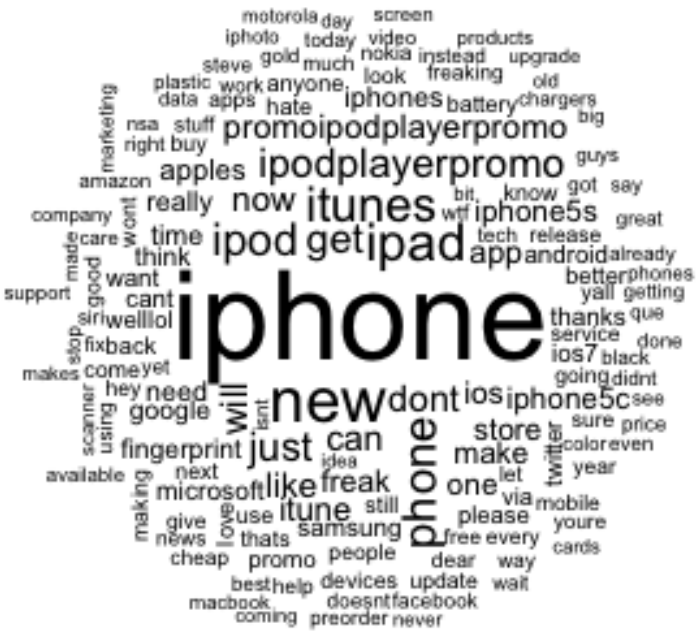
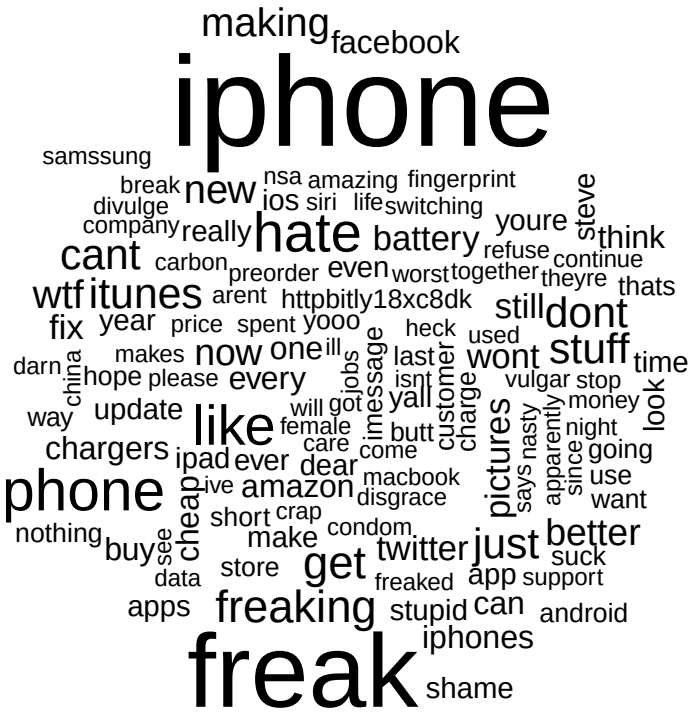


Figure 2: Word Cloud B

Figure 3: Word Cloud C

Figure 4: Word Cloud D

1. Word Cloud A
2. Word Cloud B
3. **Word Cloud C**
4. Word Cloud D



Explanation:
Word Cloud C is the only one with a different distribution of the most frequent words – negative words (or censored versions of negative words) are much more common in this cloud. It is quite simple to obtain a word cloud that is limited to a subset of the tweets using the subset function:

Problem 3.2 Only one word cloud was created without modifying parameters min.freq or max.words. **Which word cloud is this?**

Answer:

1. **Word Cloud A**
2. Word Cloud B
3. Word Cloud C
4. Word Cloud D

Explanation:
min.freq and max.words are parameters that can be used to remove the least frequent words, resulting is a less cluttered word cloud. Word Cloud A is much more cluttered than the others because it did not use either of these parameters, and therefore is displaying every word that appears more than 3 times.

Problem 3.3 Which word clouds were created with parameter random.order set to FALSE?

Answer:

1. Word Cloud A
2. **Word Cloud B**
3. Word Cloud C
4. **Word Cloud D**

Explanation:
If random.order is set to FALSE, then the most frequent (largest) words will be plotted first, resulting in them being displayed together in the center of the word cloud. This is the case in Word Cloud B and Word Cloud D.

Problem 3.4 Which word cloud was built with a non-default value for parameter rot.per?

Answer:

1. **Word Cloud A**
2. Word Cloud B
3. Word Cloud C
4. Word Cloud D

Explanation:
rot.per controls the proportion of words that are rotated to be vertical in the word cloud. By default 10% of words are rotated. However in Word Cloud A a much higher proportion (50%) are rotated, which was achieved by setting rot.per=0.5.

Problem 3.5 In Word Cloud C and Word Cloud D, we provided a color palette ranging from light purple to dark purple as the parameter colors (you will learn how to make such a color palette later in this assignment). **For which word cloud was the parameter random.color set to TRUE?**

Answer:

3. Word Cloud C
4. **Word Cloud D**

Explanation:
When random.color is set to TRUE, the words will be colored randomly. This is the case in Word Cloud D. Meanwhile, colors were assigned based on the number of appearances in Word Cloud C.

Problem 4 : Selecting a Color Palette

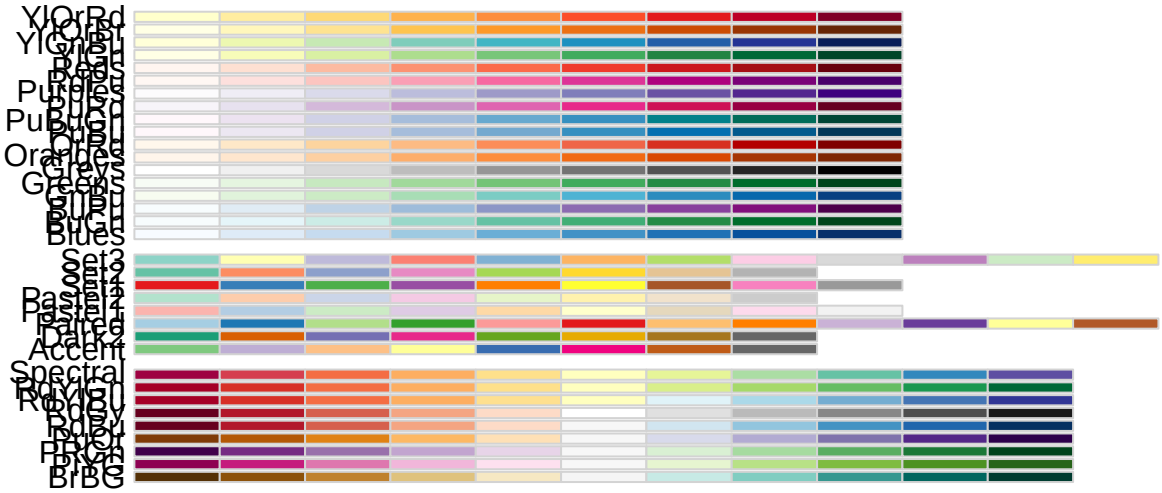
Problem 4.1 The use of a palette of colors can often improve the overall effect of a visualization. We can easily select our own colors when plotting; for instance, we could pass c("red", "green", "blue") as the colors parameter to wordcloud(). The RColorBrewer package, which is based on the ColorBrewer project (colorbrewer.org), provides pre-selected palettes that can lead to more visually appealing images. Though these palettes are designed specifically for coloring maps, we can also use them in our word clouds and other visualizations.

Begin by installing and loading the "RColorBrewer" package. This package may have already been installed and loaded when you installed and loaded the "wordcloud" package, in which case you don't need to go

through this additional installation step. If you obtain errors (for instance, "Error: lazy-load database 'P' is corrupt") after installing and loading the RColorBrewer package and running some of the commands, try closing and re-opening R.

The function brewer.pal() returns color palettes from the ColorBrewer project when provided with appropriate parameters, and the function display.brewer.all() displays the palettes we can choose from.

Which color palette would be most appropriate for use in a word cloud for which we want to use color to indicate word frequency?



Answer:

1. Accent
2. Set2
3. **YlOrRd**

Explanation:
From ?brewer.pal we read that Accent and Set2 are both "qualitative palettes," which means color changes don't imply a change in magnitude (we can also see this in the output of display.brewer.all). As a result, the colors selected would not visually identify the least and most frequent words. On the other hand, YlOrRd is a "sequential palette," with earlier colors begin lighter and later colors being darker. Therefore, it is a good palette choice for indicating low-frequency vs. high-frequency words.

Problem 4.2 Which RColorBrewer palette name would be most appropriate to use when preparing an image for a document that must be in grayscale?

Answer : Greys

Explanation:
As we can see from display.brewer.all(), palette "Greys" is the only one completely in grayscale.

Problem 4.3 In sequential palettes, sometimes there is an undesirably large contrast between the lightest and darkest colors. You can see this effect when plotting a word cloud for allTweets with parameter colors=brewer.pal(9, "Blues"), which returns a sequential blue palette with 9 colors.

Which of the following commands addresses this issue by removing the first 4 elements of the 9-color palette of blue colors? Select all that apply.

1. brewer.pal(9, “Blues”)[c(-5, -6, -7, -8, -9)]
2. **brewer.pal(9, “Blues”)[c(-1, -2, -3, -4)]**
3. brewer.pal(9, “Blues”)[c(1, 2, 3, 4)]
4. **brewer.pal(9, “Blues”)[c(5, 6, 7, 8, 9)]**

Explanation:
The fourth option limits to elements 5-9, which removes the first four. The second option uses negative indexes, which means remove elements 1-4. The first and third options actually keep colors 1-4, discarding the rest. A shorthand for this indexing is: