Visualizing Text Data Using Word Clouds

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.2

## -- Attaching packages ----------------------------------------------------------------------------------

## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.0     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0

## Warning: package 'ggplot2' was built under R version 3.4.4

## Warning: package 'tibble' was built under R version 3.4.4

## Warning: package 'tidyr' was built under R version 3.4.3

## Warning: package 'purrr' was built under R version 3.4.4

## Warning: package 'dplyr' was built under R version 3.4.4

## Warning: package 'forcats' was built under R version 3.4.3

## -- Conflicts -------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.4.3
```

```r
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.4.3

## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
library(wordcloud)
```

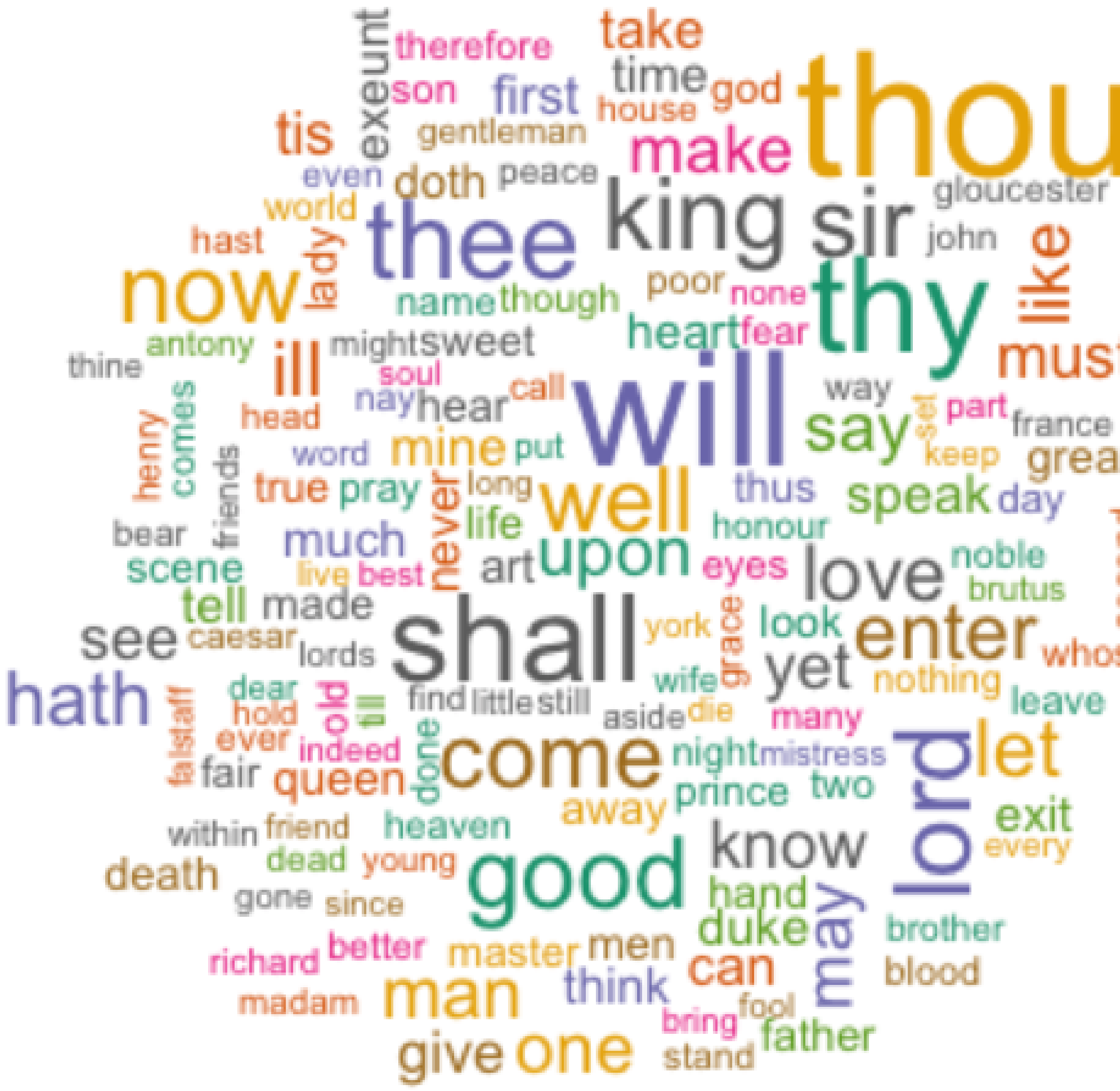
```
## Warning: package 'wordcloud' was built under R version 3.4.4

## Loading required package: RColorBrewer
```

```r
library(RColorBrewer)
```

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:

```r
include_graphics('wc.png')
```



1.1) Preparing the Data

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.

```r
tweets = read.csv('tweets.csv')
```

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

```r
corpus = VCorpus(VectorSource(tweets$Tweet))
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time(), tz = "GMT"): unknown timezone
## 'zone/tz/2018i.1.0/zoneinfo/America/Chicago'
```

2) Convert the corpus to lowercase

```r
corpus = tm_map(corpus, content_transformer(tolower))
```

3) Remove punctuation from the corpus

```r
corpus = tm_map(corpus, removePunctuation)
```

4) Remove all English-language stopwords

```r
corpus = tm_map(corpus, removeWords, stopwords("english"))
```

5) Build a document-term matrix out of the corpus

```r
dtm = DocumentTermMatrix(corpus)

dtm
```

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

6) Convert the document-term matrix to a data frame called allTweets

```r
all_tweets = as.data.frame(as.matrix(dtm))
```

How many unique words are there across all the documents?

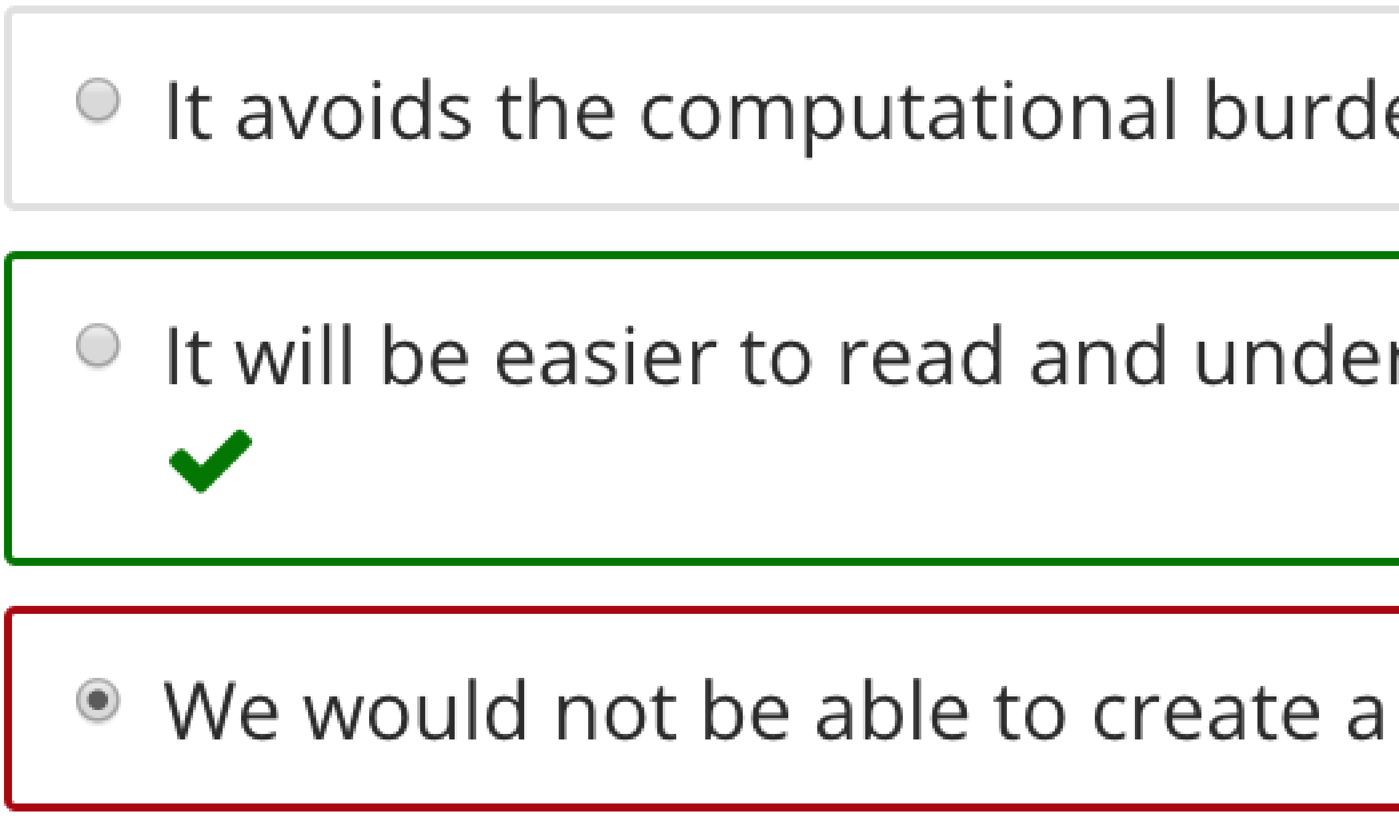
```r
ncol(all_tweets)
```

```
## [1] 3780
```

1.2) Preparing the Data

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?**

```r
include_graphics('1.2.png')
```



- It avoids the computational burd
- It will be easier to read and under ✓
- [✓ We would not be able to create a]

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.

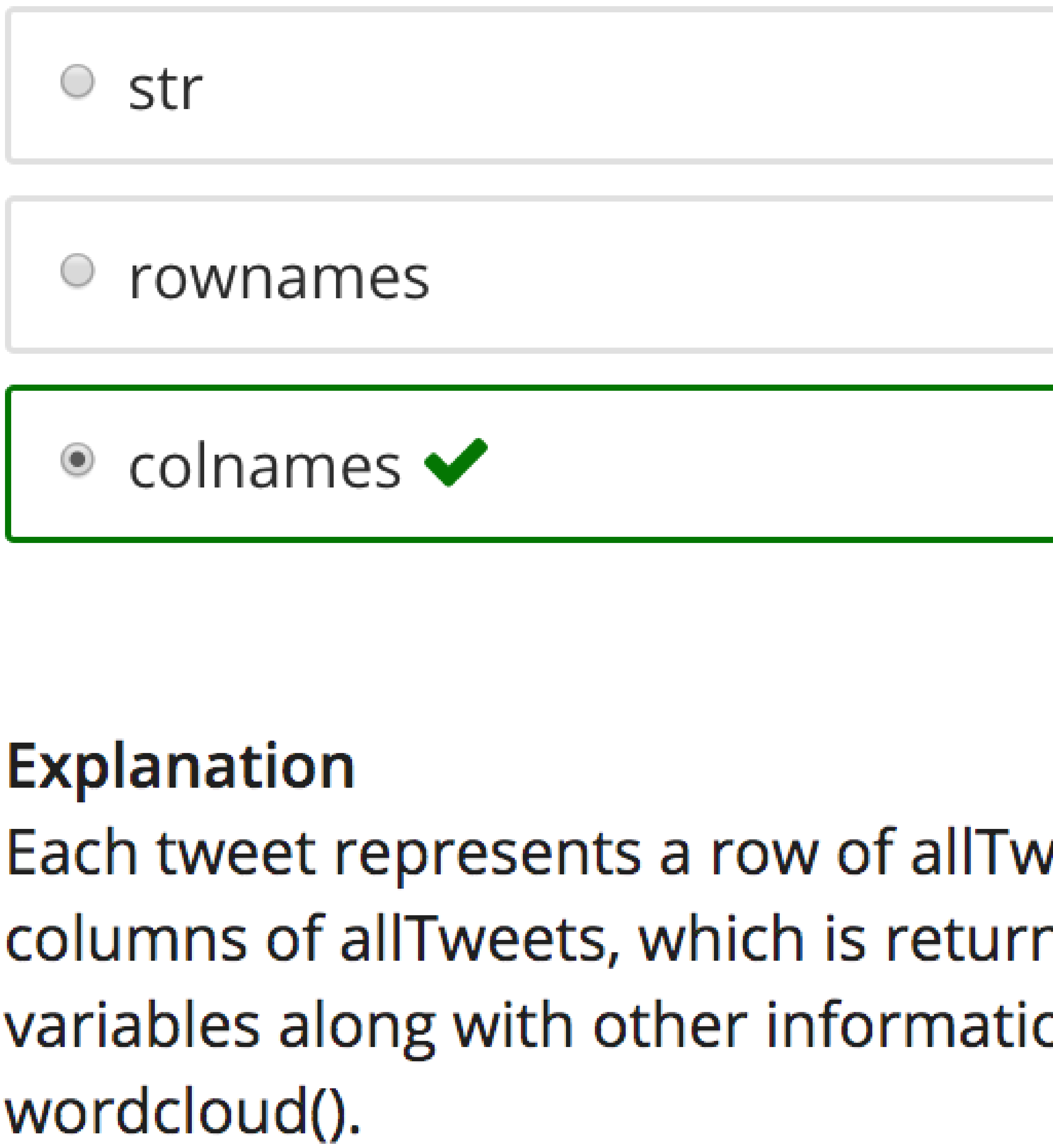
Stemming has relatively minor computational burden, and we certainly could create a word cloud with a stemmed document.

2.1) Building a Word Cloud

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()?

```
include_graphics('2.1.png')
```



- ○ str
- ○ rownames
- [✓ colnames] ✓

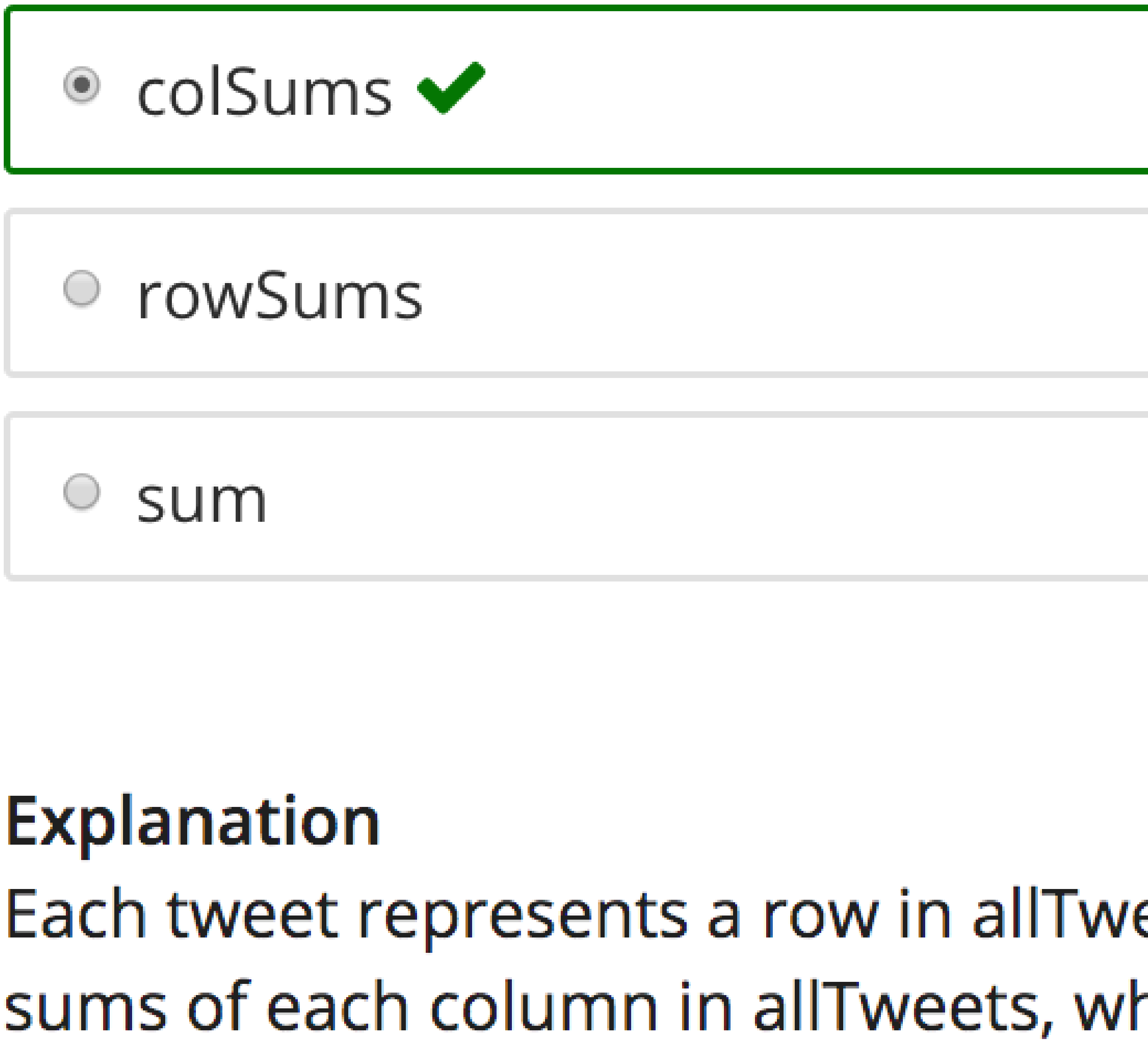
Explanation

Each tweet represents a row of allTw
columns of allTweets, which is return
variables along with other informatio
wordcloud().

2.2) Building a Word Cloud

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

```r
include_graphics('2.2.png')
```



[✓ colSums] ✓

rowSums

sum

Explanation

Each tweet represents a row in allTwe

sums of each column in allTweets, wh

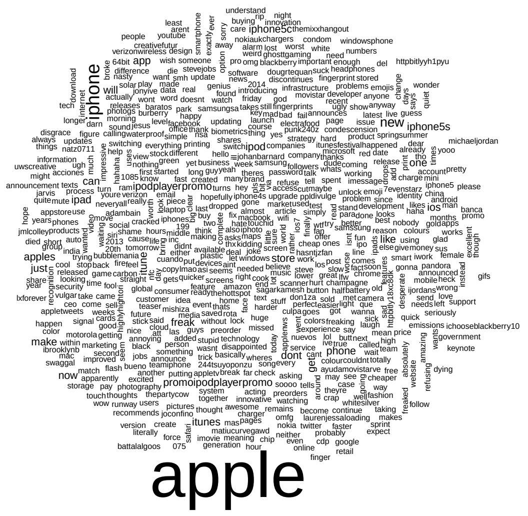
2.3) Building a Word Cloud

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.

```
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25))
```



2.4) Building a Word Cloud

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.

1) Create a corpus using the Tweet variable

```
corpus = VCorpus(VectorSource(tweets$Tweet))
```

2) Convert the corpus to lowercase

```
corpus = tm_map(corpus, content_transformer(tolower))
```

3) Remove punctuation from the corpus

```
corpus = tm_map(corpus, removePunctuation)
```

4) Remove all English-language stopwords

```r
corpus = tm_map(corpus, removeWords, stopwords("english"))
corpus = tm_map(corpus, removeWords, 'apple')
```

5) Build a document-term matrix out of the corpus

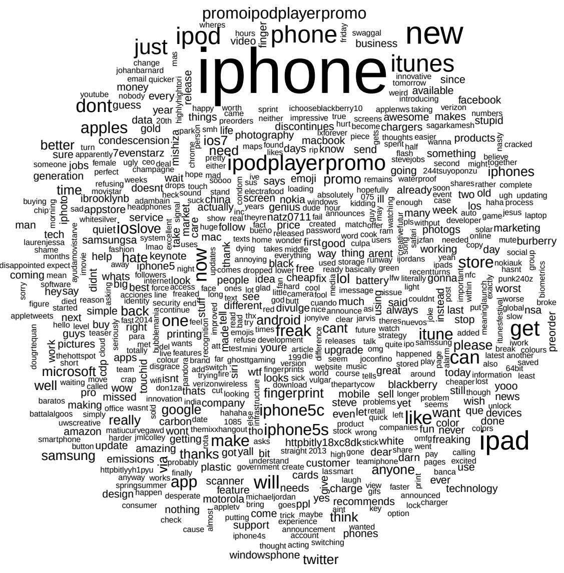
```r
dtm = DocumentTermMatrix(corpus)

dtm
```

```
## <<DocumentTermMatrix (documents: 1181, terms: 3779)>>
## Non-/sparse entries: 9121/4453878
## Sparsity           : 100%
## Maximal term length: 115
## Weighting          : term frequency (tf)
```

6) Convert the document-term matrix to a data frame called allTweets

```r
all_tweets = as.data.frame(as.matrix(dtm))
```

```r
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25))
```



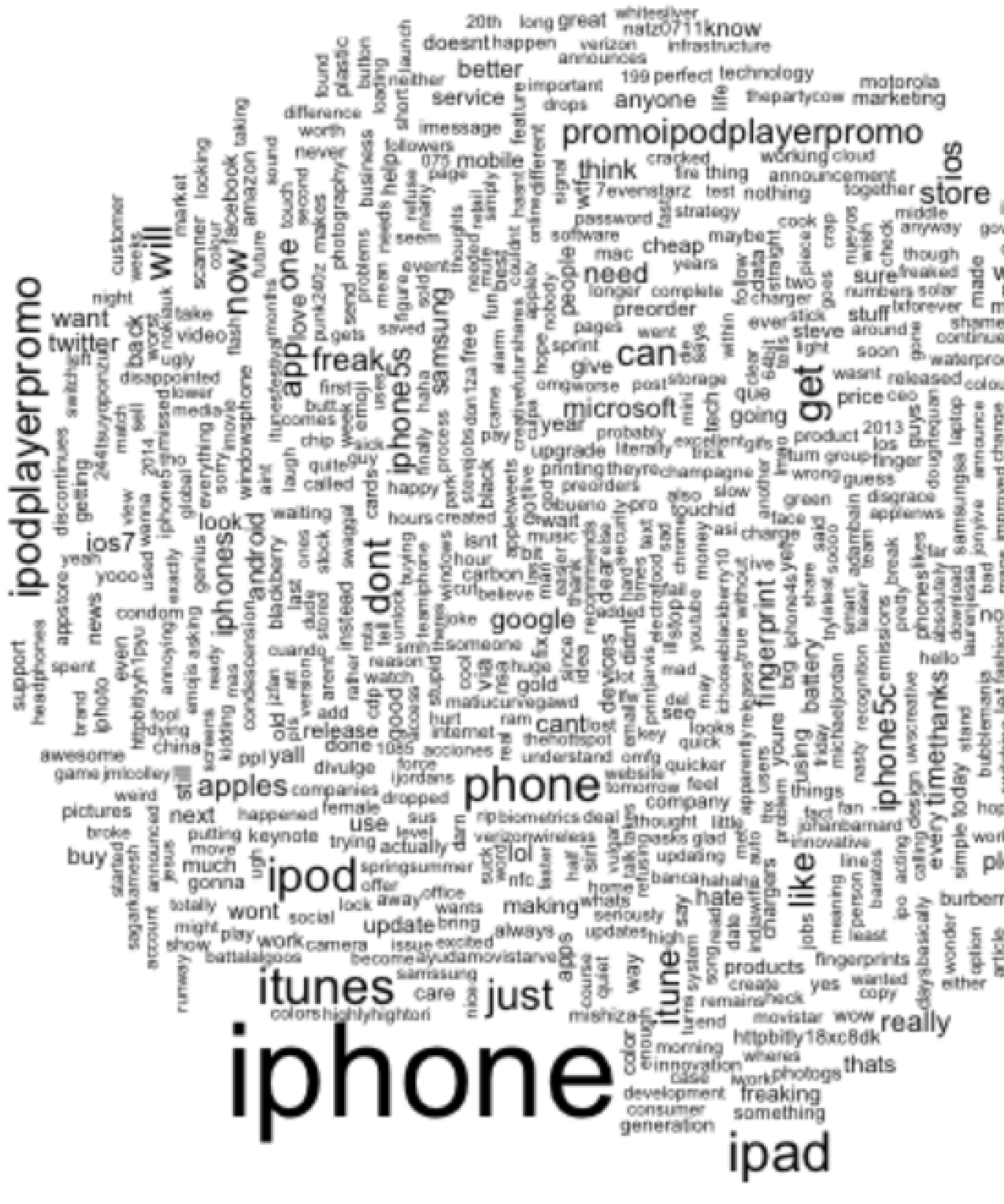
3) Size and Color

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:

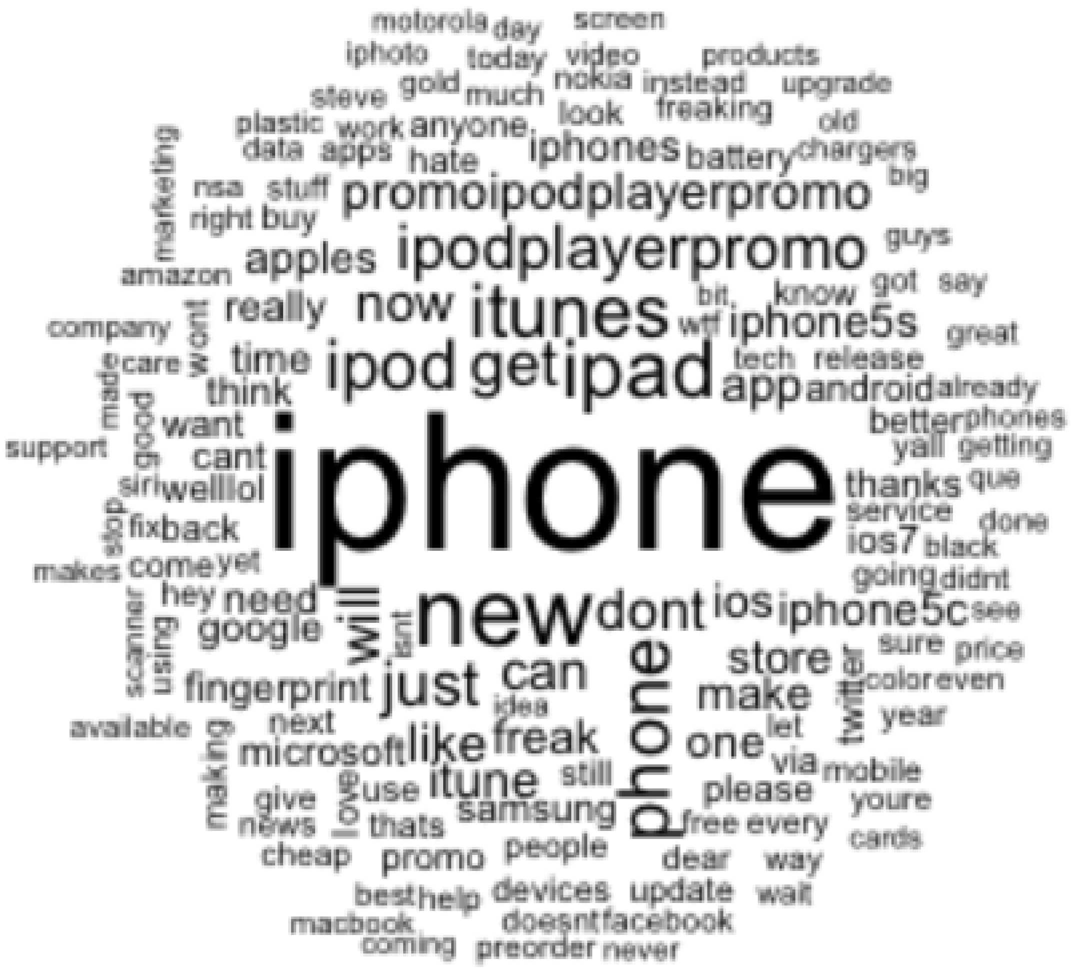
Word Cloud A:

```r
include_graphics('wcA.png')
```



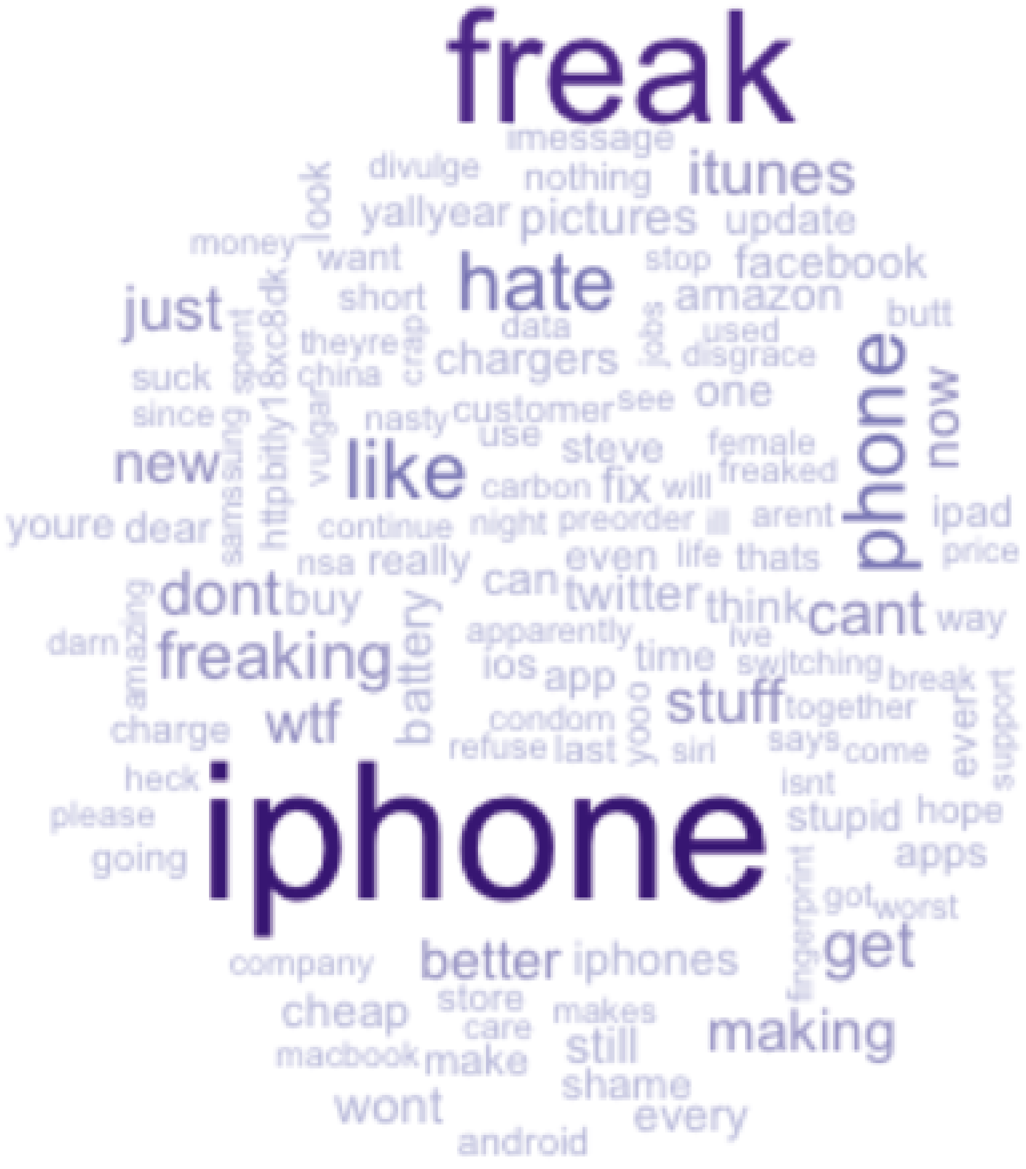
Word Cloud B

```r
include_graphics('wcB.png')
```



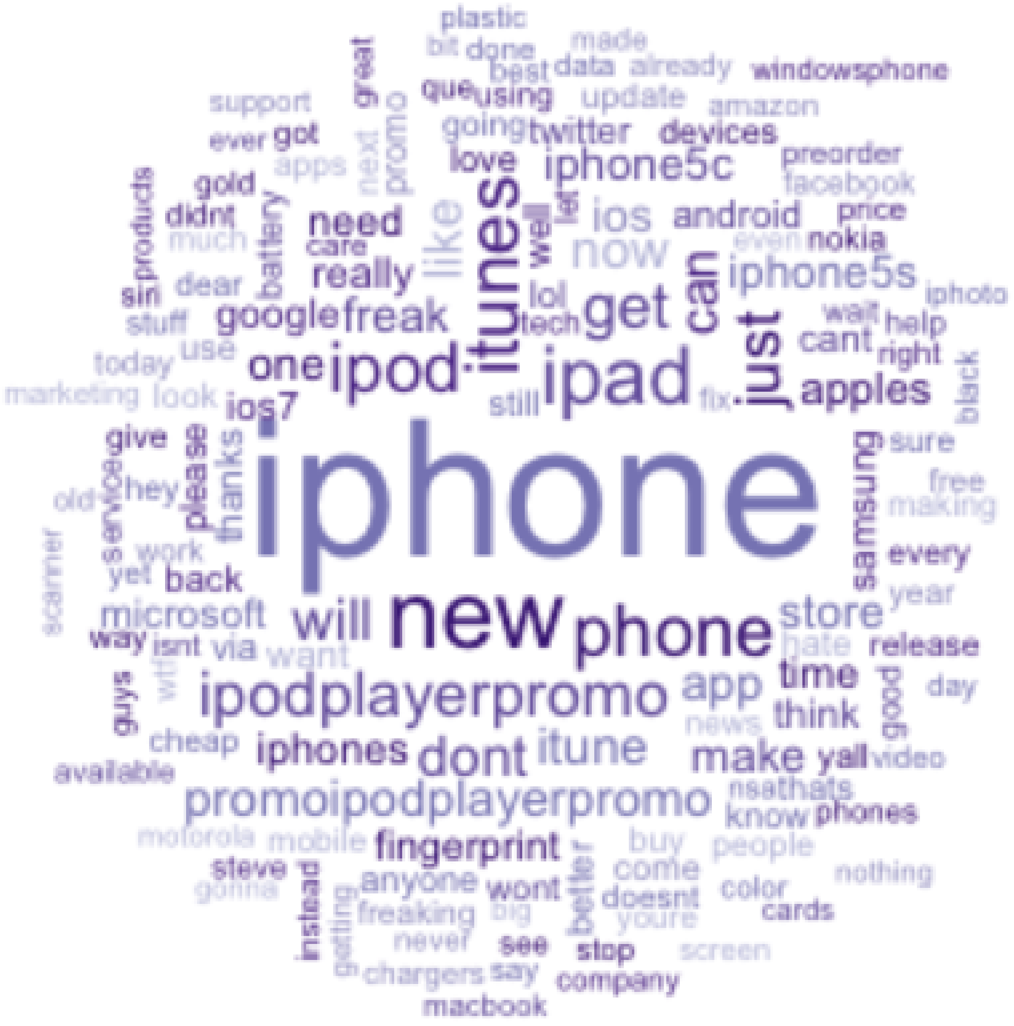
Word Cloud C

```r
include_graphics('wcC.png')
```



Word Cloud D

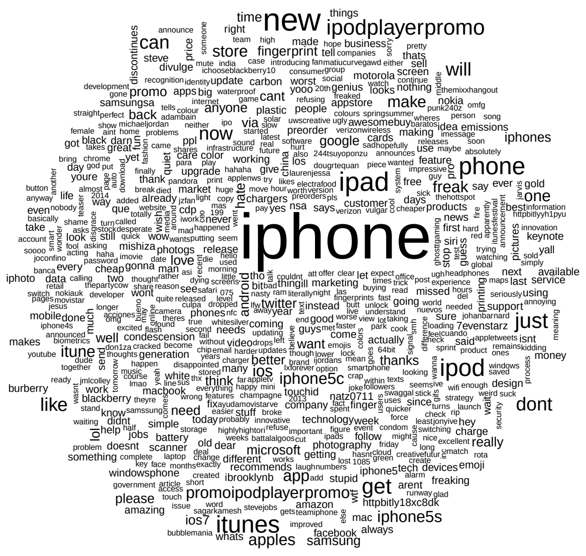
```
include_graphics('wcD.png')
```



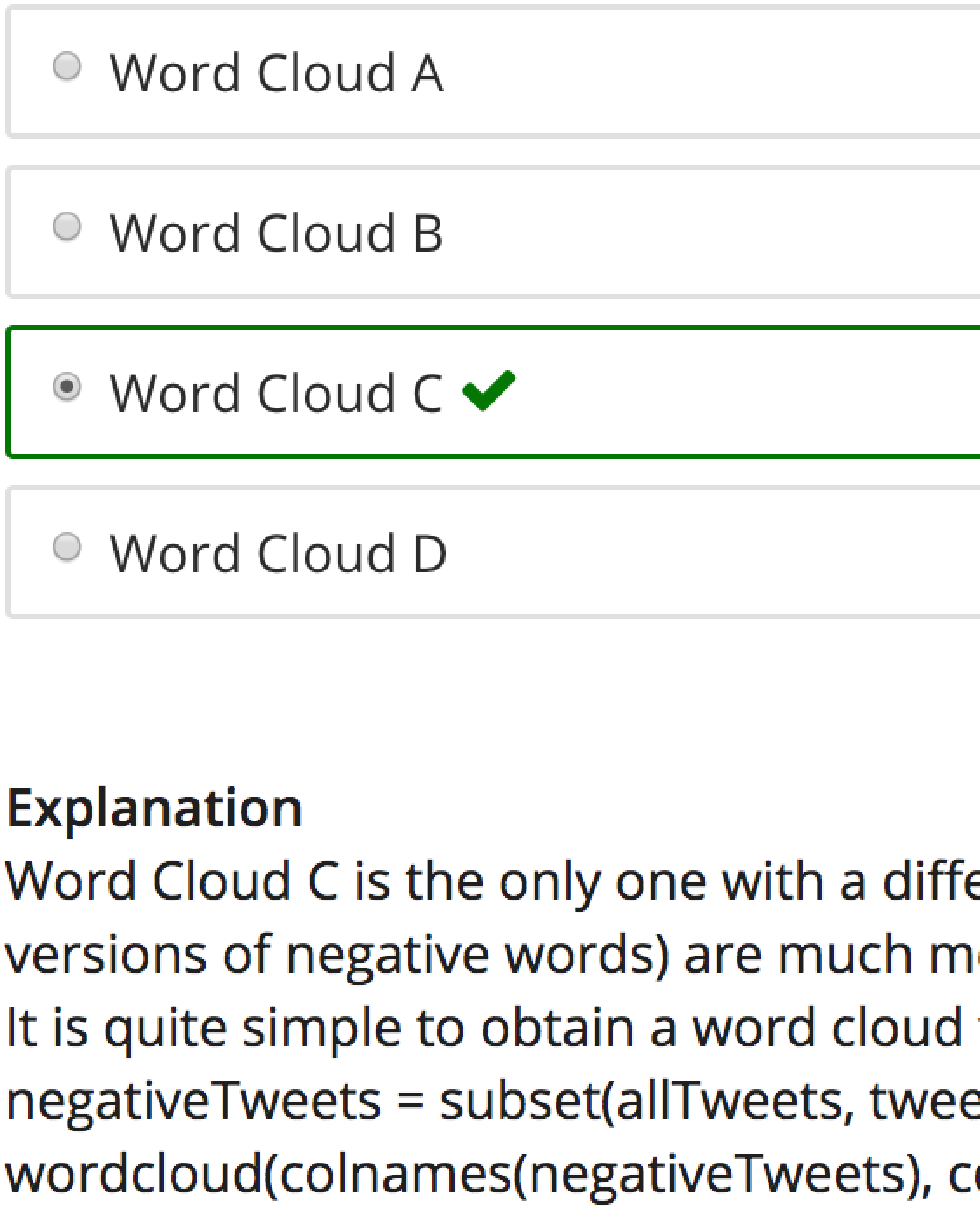
3.1) Size and Color

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

```r
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25))
```



```r
include_graphics('3.1.png')
```



- Word Cloud A
- Word Cloud B
- [✓ Word Cloud C] ✓
- Word Cloud D

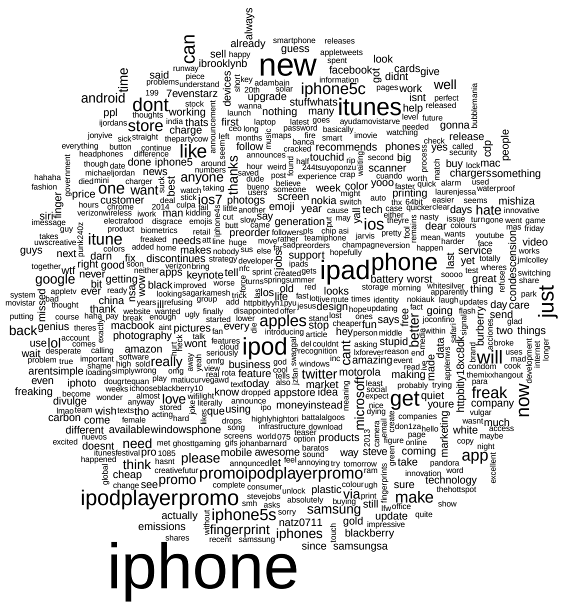
Explanation

Word Cloud C is the only one with a diffe
versions of negative words) are much m
It is quite simple to obtain a word cloud
negativeTweets = subset(allTweets, twee
wordcloud(colnames(negativeTweets), c

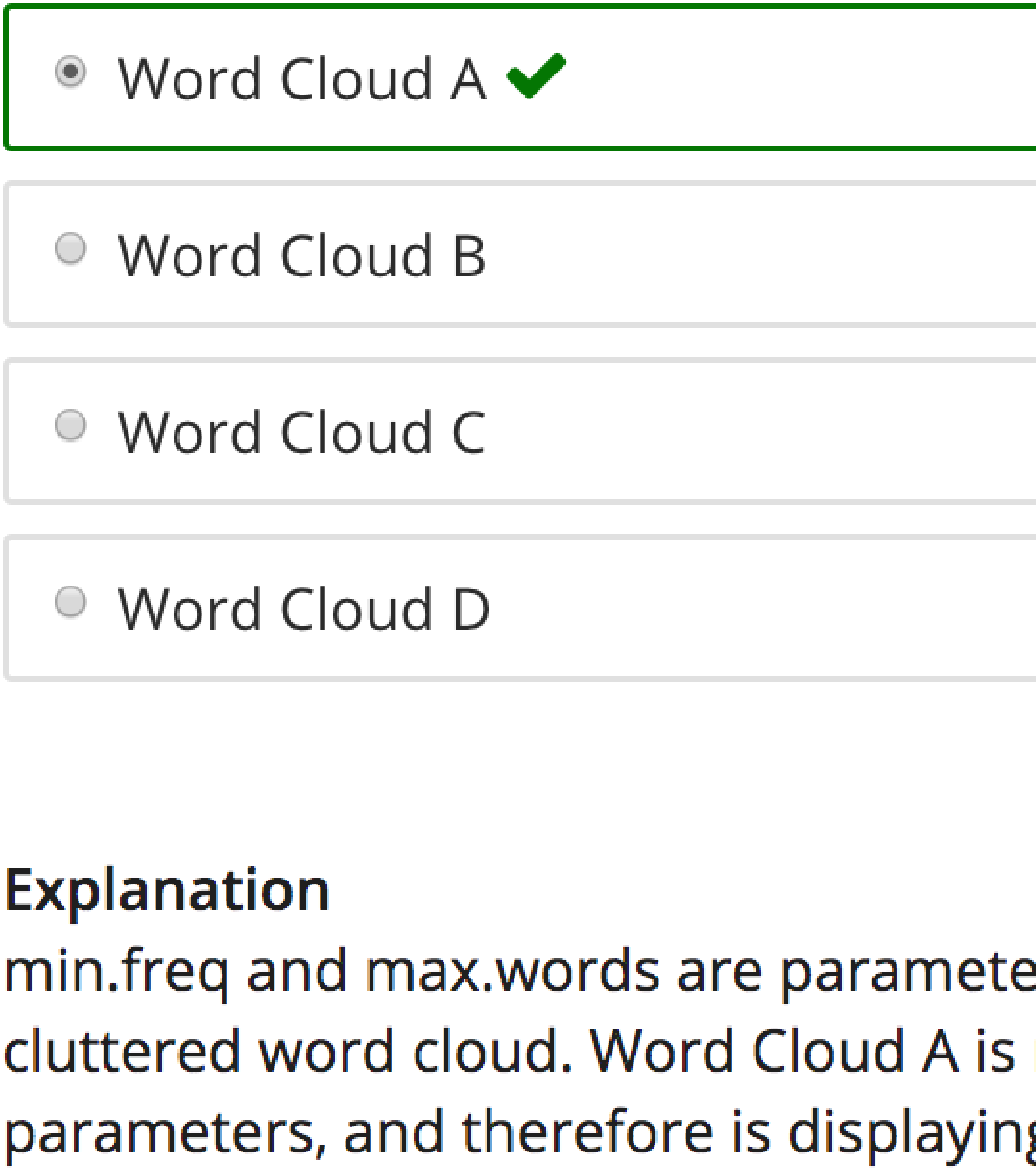
3.2) Size and Color

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

```r
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25))
```



```r
include_graphics('3.2.png')
```



[✓ Word Cloud A] ✓

Word Cloud B

Word Cloud C

Word Cloud D

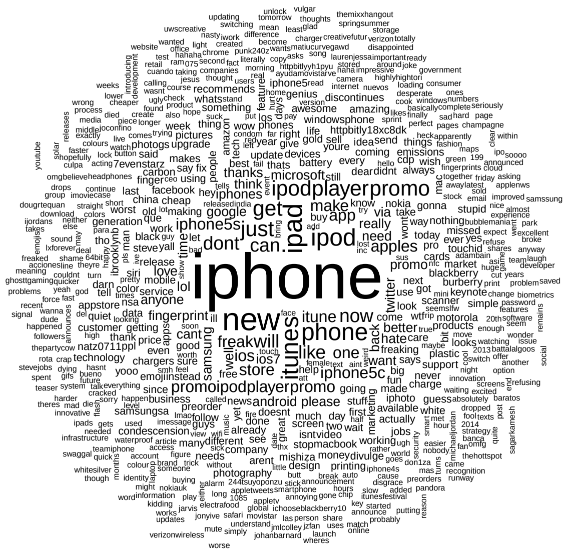
Explanation

min.freq and max.words are paramete
cluttered word cloud. Word Cloud A is
parameters, and therefore is displayin

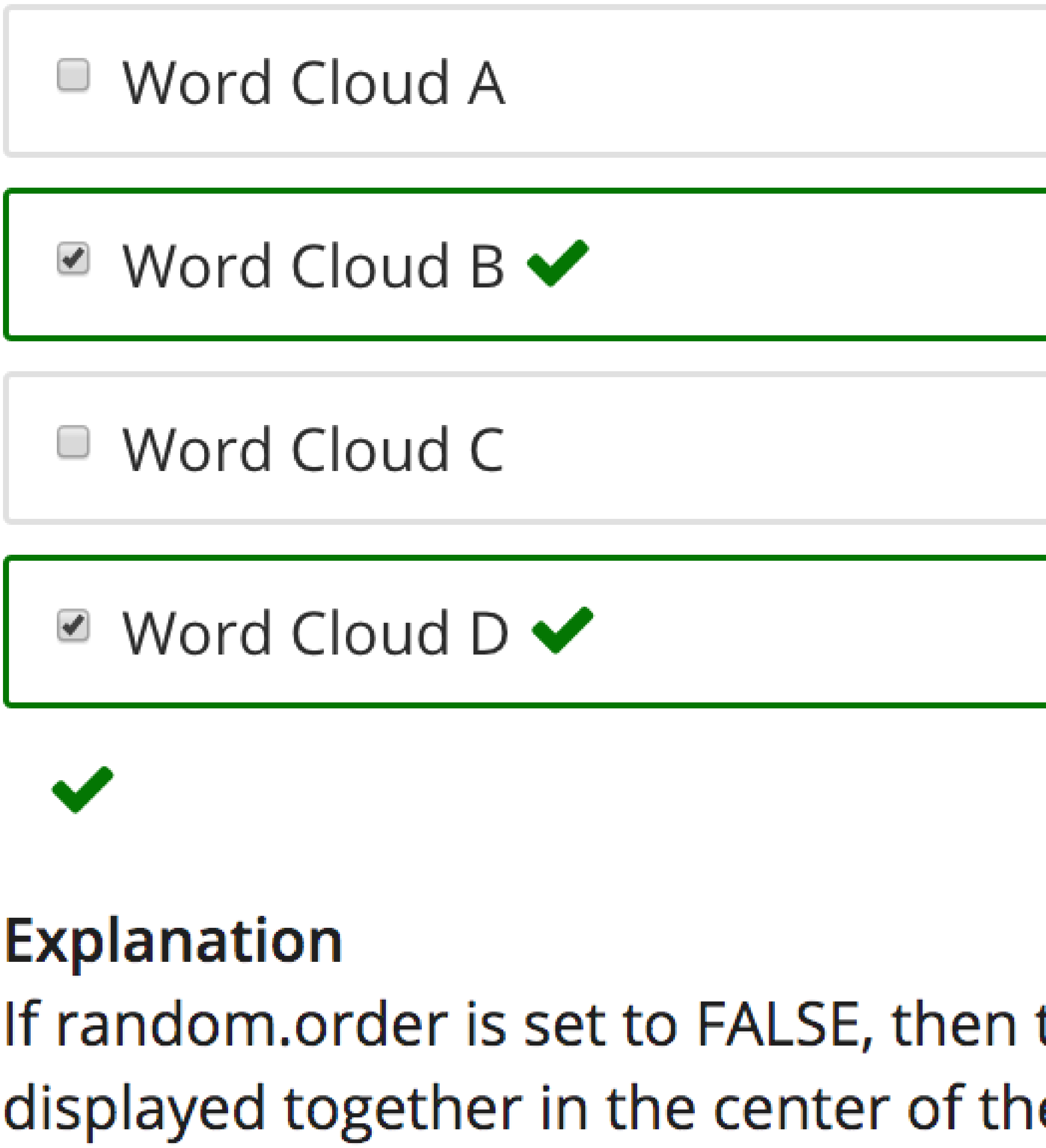
3.3) Size and Color

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

```
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25),
          random.order = FALSE)
```



```
include_graphics('3.3.png')
```



- ☐ Word Cloud A
- ☑ Word Cloud B ✓
- ☐ Word Cloud C
- ☑ Word Cloud D ✓

✓

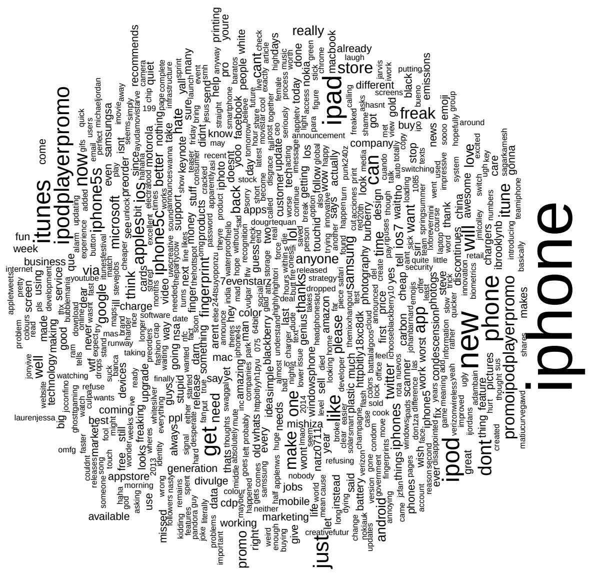
Explanation

If random.order is set to FALSE, then t
displayed together in the center of the

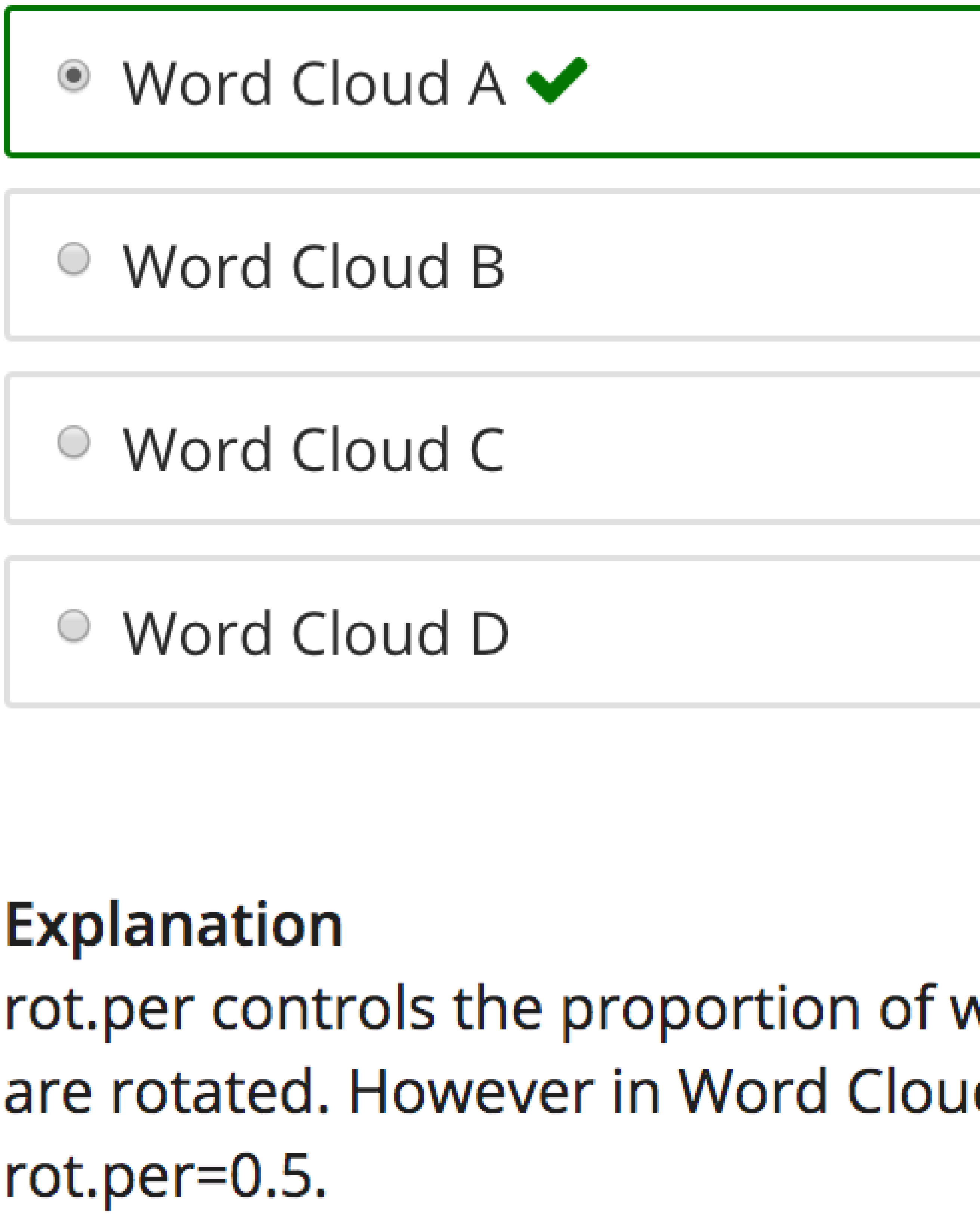
3.4) Size and Color

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

```r
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25),
          rot.per = 0.9)
```



```r
include_graphics('3.4.png')
```



[✓ Word Cloud A]

Word Cloud B

Word Cloud C

Word Cloud D

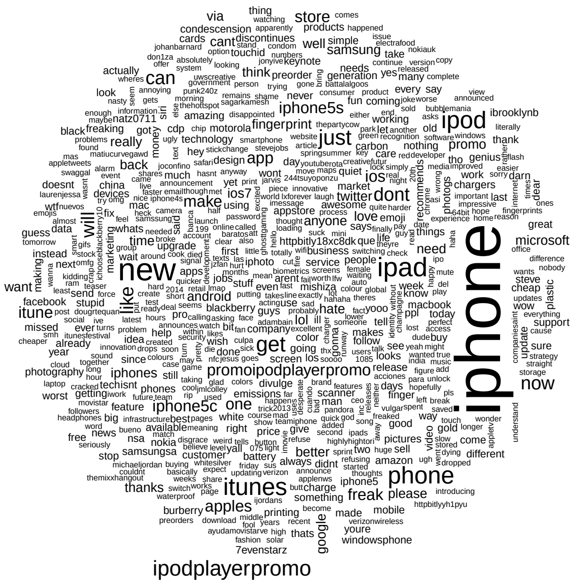
Explanation

rot.per controls the proportion of w
are rotated. However in Word Clou
rot.per=0.5.

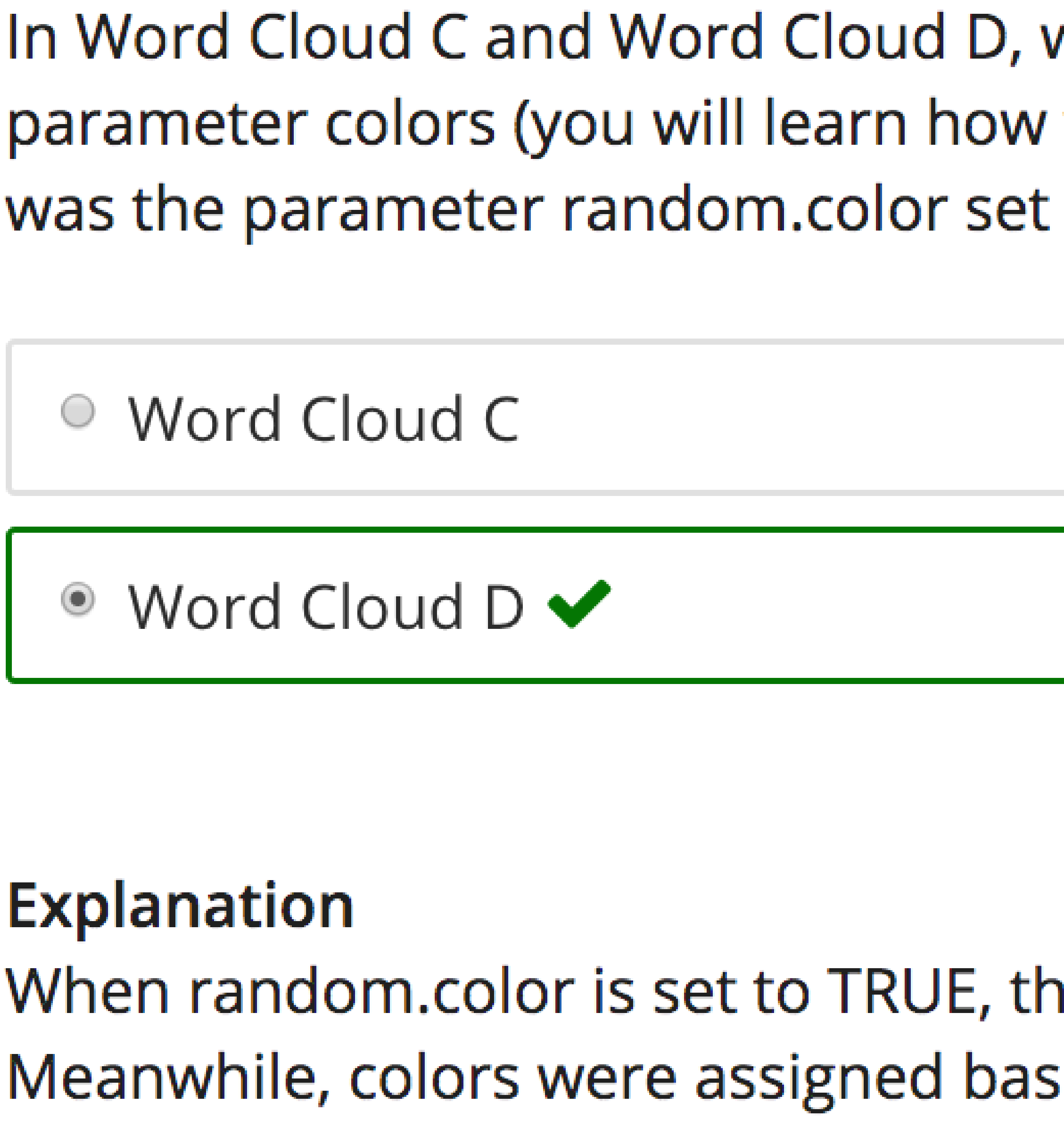
3.5) Size and Color

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?

```r
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25),
          random.color = TRUE)
```



```r
include_graphics('3.5.png')
```


In Word Cloud C and Word Cloud D, w
parameter colors (you will learn how
was the parameter random.color set

Word Cloud C

[✓ Word Cloud D]

Explanation

When random.color is set to TRUE, th
Meanwhile, colors were assigned bas

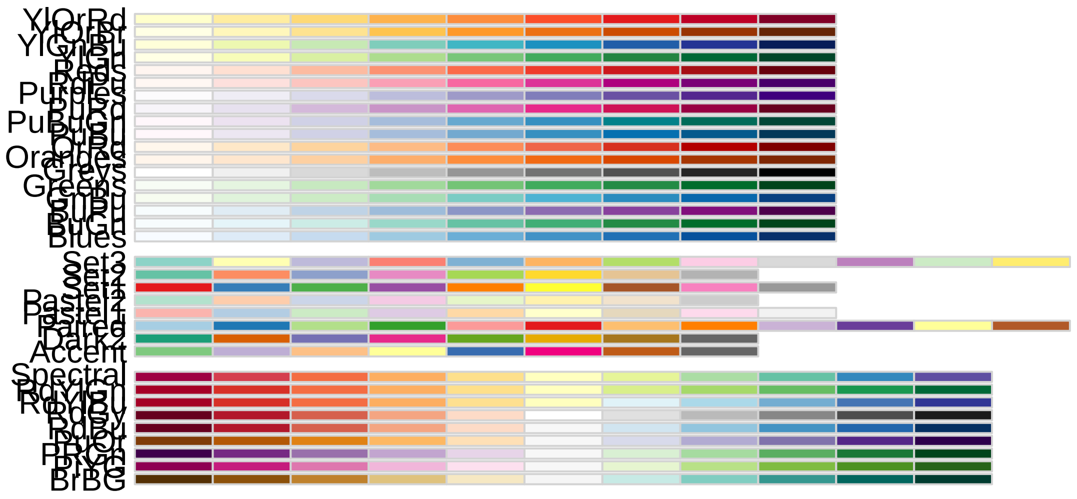
4.1) Setting a Color Palette

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.

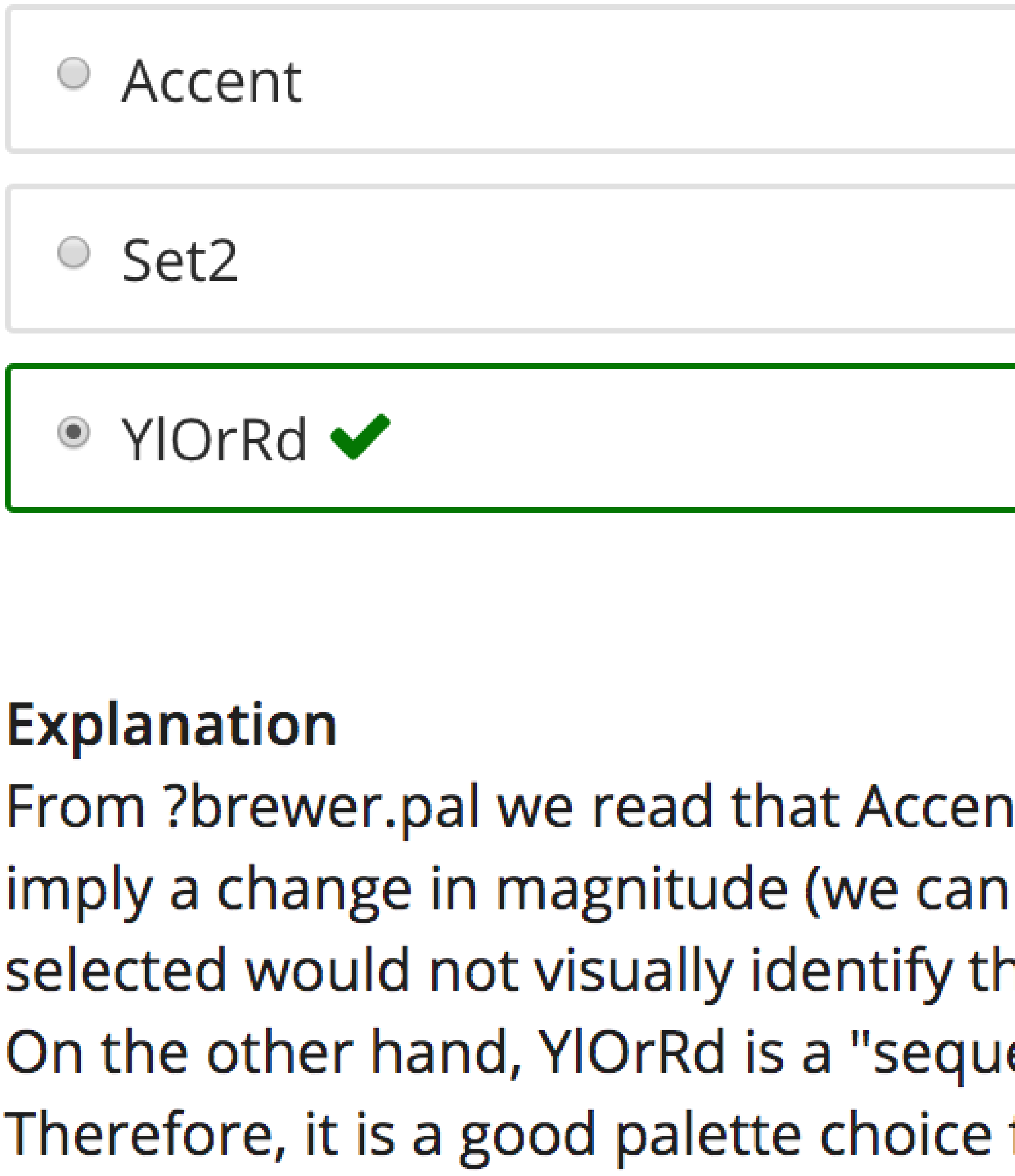
The function brewer.pal() returns color palettes from the ColorBrewer project when provided with appropriate parameters, and the function display.brewer.all() displays the palettes we can choose from.

```r
display.brewer.all()
```



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

```r
include_graphics('4.1.png')
```



- Accent
- Set2
- [✓ YlOrRd] ✓

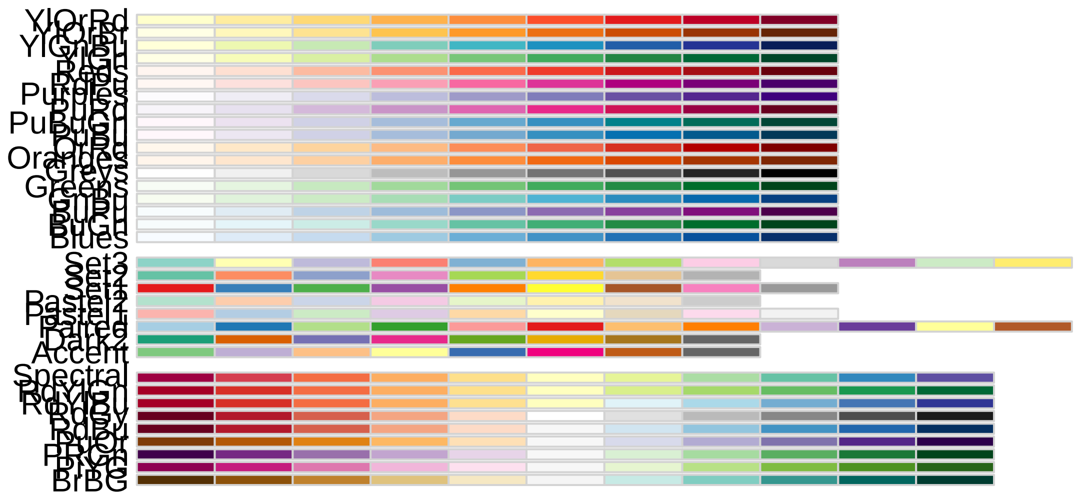
Explanation

From ?brewer.pal we read that Accen
imply a change in magnitude (we can
selected would not visually identify th
On the other hand, YlOrRd is a "sequ
Therefore, it is a good palette choice f

4.2) Setting a Color Palette

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

```r
display.brewer.all()
```



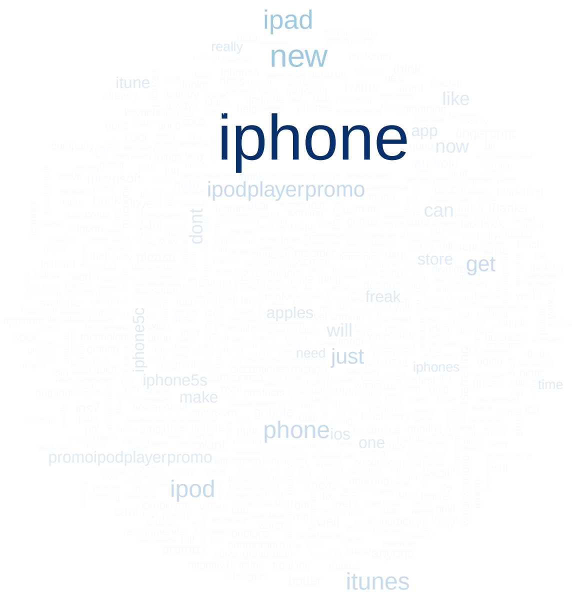
Greys

4.3) Setting a Color Palette

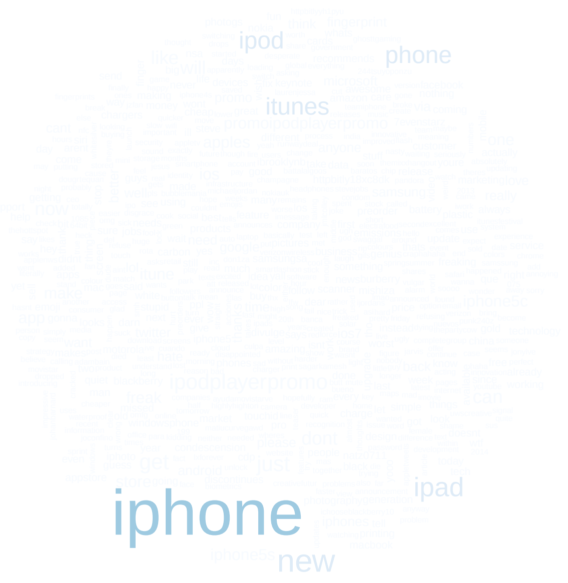
In sequential palettes, sometimes there is an undesirably large contrast between the lightest and darkest colors. You can see this effect when plotting a word cloud for allTweets with parameter colors=brewer.pal(9, "Blues"), which returns a sequential blue palette with 9 colors.

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.

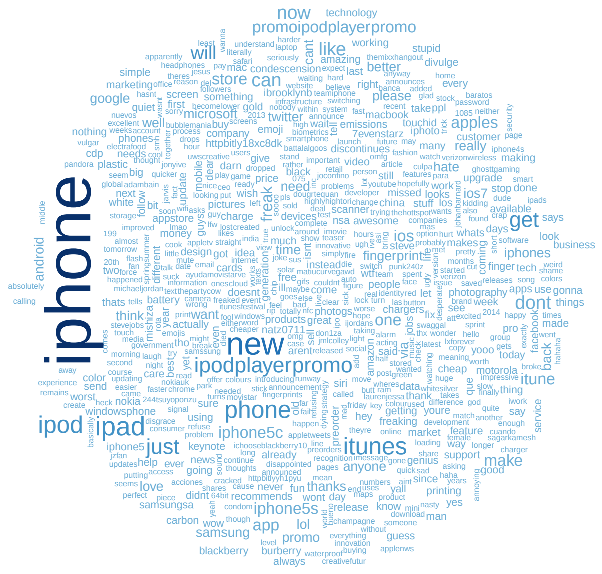
```r
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25),
          colors = brewer.pal(9, "Blues"))
```



```r
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25),
          colors = brewer.pal(9, 'Blues')[c(-5,-6,-7,-8,-9)])
```



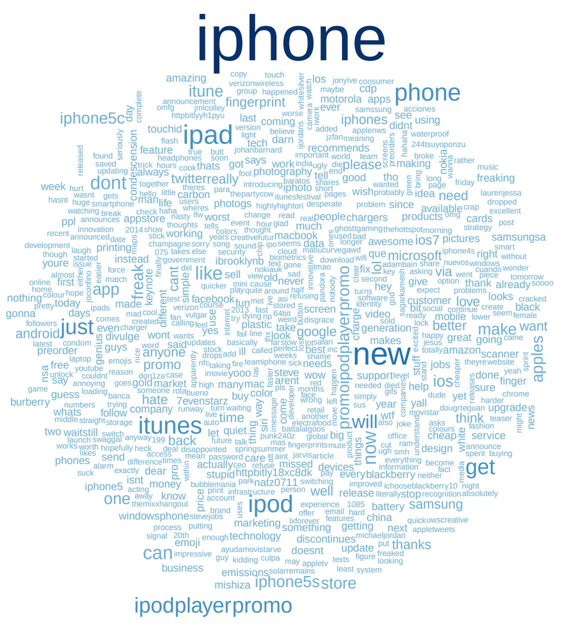
```r
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25),
          colors = brewer.pal(9, 'Blues')[c(-1,-2,-3,-4)])
```



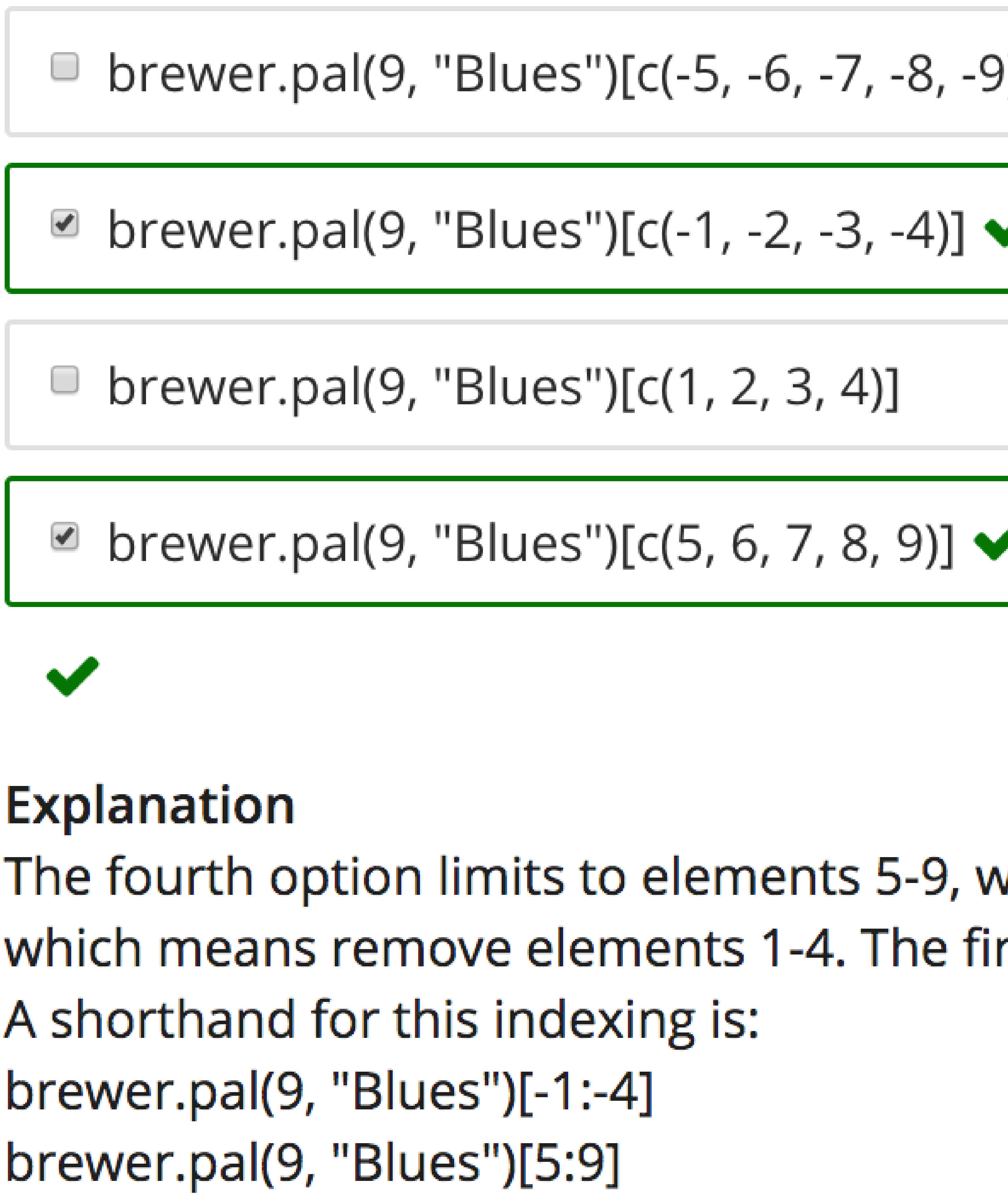
```
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25),
          colors = brewer.pal(9, 'Blues')[c(1,2,3,4)])
```

iphone
new
phone
itunes
ipad
ipod
ipodplayerpromo
promoipodplayerpromo

```
wordcloud(colnames(all_tweets),
          colSums(all_tweets),
          scale = c(2, 0.25),
          colors = brewer.pal(9, 'Blues')[c(5,6,7,8,9)])
```



```r
include_graphics('4.3.png')
```



☐ brewer.pal(9, "Blues")[c(-5, -6, -7, -8, -9

☑ brewer.pal(9, "Blues")[c(-1, -2, -3, -4)] ✓

☐ brewer.pal(9, "Blues")[c(1, 2, 3, 4)]

☑ brewer.pal(9, "Blues")[c(5, 6, 7, 8, 9)] ✓

✓

Explanation

The fourth option limits to elements 5-9, w
which means remove elements 1-4. The fir
A shorthand for this indexing is:

```
brewer.pal(9, "Blues")[-1:-4]
brewer.pal(9, "Blues")[5:9]
```