MIS 506_ Final_Project

Final Project

Load the dataframe of tweets by President Trump:

```
load("Trump_Tweets.Rdata")
```

The "load()" command opens up the data set "Trump_Tweets.Rdata", consisting of 3,196 observations and 42 variables, up in the global environment and allows for future manipulation/analysis.

Load the libraries

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 3.5.3
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.2
```

```
## -- Attaching packages ---------------------------------------------------
------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.3.1
## v tibble  2.0.1     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'tibble' was built under R version 3.5.2
```

```
## Warning: package 'tidyr' was built under R version 3.5.2
```

```
## Warning: package 'readr' was built under R version 3.5.2
```

```
## Warning: package 'purrr' was built under R version 3.5.2
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
## Warning: package 'stringr' was built under R version 3.5.2
```

```
## Warning: package 'forcats' was built under R version 3.5.2
```

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

The above code is used to load the packages necessary to do our analysis. In this case, it is the “tidytext” and the “tidyverse” packages.

Tidy text

Transform them into tidytext format. Pre process the data - Remove numbers, whitespaces, and words with less that three characters. Also remove stop words (frequent words such as “the”, and “and”) as well as other unmeaningful words (e.g. https):

```
tidy_trump_tweets<- trumptweets %>%
   select(created_at,text) %>%
     unnest_tokens("word", text)%>%
     anti_join(stop_words) %>%
     filter(!nchar(word) < 3,
         !str_detect(word, "^\\b\\d+\\b"),
         !str_detect(word, "\\s+"),
         !str_detect(word, "[^a-zA-Z]"),
         !str_detect(word,"https|t.co|amp|rt"))

## Joining, by = "word"
```

The above code creates a new tibble titled, “tidy_trump_tweets”, from the “trumptweets” data set. By using the “unnest_tokens” command on the original data set, the strings of text are separated into one-word-per-row resulting in 31,639 observations. The “select()” command also chooses the variables we want to analyze in the new tibble, created_at and text. The remainder of the code is used to remove unwanted characters, numbers, and words from the new tibble.

Word Frequency

Count the top words:

```
trump_tweet_top_words<- tidy_trump_tweets %>%
         count(word) %>%
         arrange(desc(n))
```

In order to determine the individual words used the most by trump on his Twitter, the “count()” command was used and then arranged in descending order. This new tibble, “trump_tweet_top_words”, has only 6,170 observations and still 2 variables.

Make a graph of the top 20 words

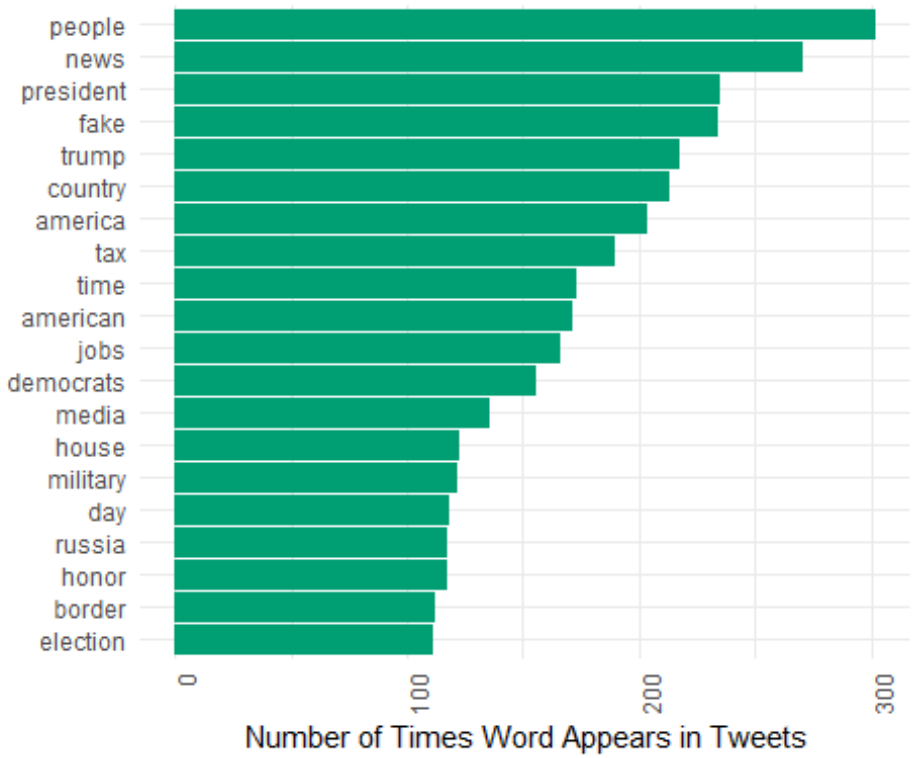
```
#select only top words
top_20<-trump_tweet_top_words[1:20,]

#create factor variable to sort by frequency
trump_tweet_top_words$word <- factor(trump_tweet_top_words$word, levels =
trump_tweet_top_words$word[order(trump_tweet_top_words$n,decreasing=TRUE)])

#define a list of colors
my_colors <- c("#E69F00", "#56B4E9", "#009E73", "#CC79A7", "#D55E00",
```

```
"#D65E00")

library(ggplot2)
top_20%>%
mutate(word = reorder(word, n)) %>%
  ggplot() +
  geom_bar(aes(word, n), fill = my_colors[3], stat="identity")+
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  ylab("Number of Times Word Appears in Tweets")+
  xlab("")+
  guides(fill=FALSE)+
  coord_flip()
```



In order to further isolate the most frequently used words on Trump's Twitter to the top 20, as well as, provide a more clear visual of how often the words are used, the above code was used. First, the tibble "top_20" used the "trump_tweet_top_words" tibble and only selected the top 20 rows. Next, the code sorts the words found in the tibble in descending order. Then, we used hexa-decimal color codes to select specific colors to use on our graph. Finally, after loading the "ggplot()" package, the "geom_bar" code was used to create a bar graph with the Y-axis labled as, "Number of Times Word Appers in Tweets."

In looking at the graph, it is clear that "people" is by far the most used word found on Trump's Twitter being found 300 different times. "People" is followed second by the word, "news" being found around 250 times and then the number of occurrences begins to quickly decrease.

TF-IDF * Calculate the tf-idf for the tweets databased in tidytext:

```
tidy_trump_tfidf<- trumptweets %>%
    select(created_at,text) %>%
      unnest_tokens("word", text) %>%
        anti_join(stop_words) %>%
           count(word, created_at) %>%
              bind_tf_idf(word, created_at, n)

## Joining, by = "word"
```

The above code creates a tibble called "tidy_trump_tfidf" from the original data set "trumptweets". It then selects the columns, "created_at" and "text" and tokenizes the text. Next, the data is parsed of stop words and then counted. Finally, the bind_tf_idf() command is used to determine the most important words. It does this by decrerasing the weight for commonly used words and increasing the weight for words that are found rarely within a collection.

The resulting tibble contains, 37,634 observations and 6 variables. In analyzing the words with the highest tf_idf scores, "standforouranthem", "cpac", and "karen" top the list with tf_idfs all greater than 6. We can tell there still needs to be some cleaning done as "cpac" was listed twice back-to-back as the #2 and #3 highest tf_idf.

what is the most unusual word:

```
top_tfidf<-tidy_trump_tfidf %>%
  arrange(desc(tf_idf))

top_tfidf$word[1]

## [1] "standforouranthem"
```

The above code analyzes the tf-idf scores of the words found in Trump's Tweets and returns the one with the highest score to determine the most unique word, "standforouranthem". In this case, it appears we may have an error in that "standforouranthem" is actually 4 words that are missing spaces.

Sentiment Analysis

Apply the bing sentiment dictionary to our database of tweets:

```
trump_tweet_sentiment <- tidy_trump_tweets %>%
  inner_join(get_sentiments("bing")) %>%
    count(created_at, sentiment)

## Joining, by = "word"

head(trump_tweet_sentiment)

## # A tibble: 6 x 3
##   created_at          sentiment     n
##   <dttm>              <chr>     <int>
## 1 2017-02-05 22:49:42 positive      1
```

```
## 2 2017-02-06 03:36:54 positive      4
## 3 2017-02-06 12:01:53 negative      2
## 4 2017-02-06 12:07:55 negative      2
## 5 2017-02-06 16:32:24 negative      3
## 6 2017-02-06 23:33:52 positive      2
```

The above code creates a new tibble, "trump_tweet_sentiment", by doing "bing" sentiment analysis on the "tidy_trump_tweets" tibble. The bing sentiment analysis categorizes words in a yes/no fashion into positive or negatve categories. In this case, there are an equal number of positive and negative sentiments.

Make a visual that compares the frequency of positive and negative tweets by day. To do this, we'll need to work a bit with the created_at variable—more specifically, we will need to transform it into a "date" object that we can use to pull out the day during which each tweet was made:

```
tidy_trump_tweets$date<-as.Date(tidy_trump_tweets$created_at,
                                              format="%Y-%m-%d %x")
```

The format argument here tells R how to read in the date character string, since dates can appear in a number of different formats, time zones, etc. For more information about how to format data with other dates, see ?as.Date()

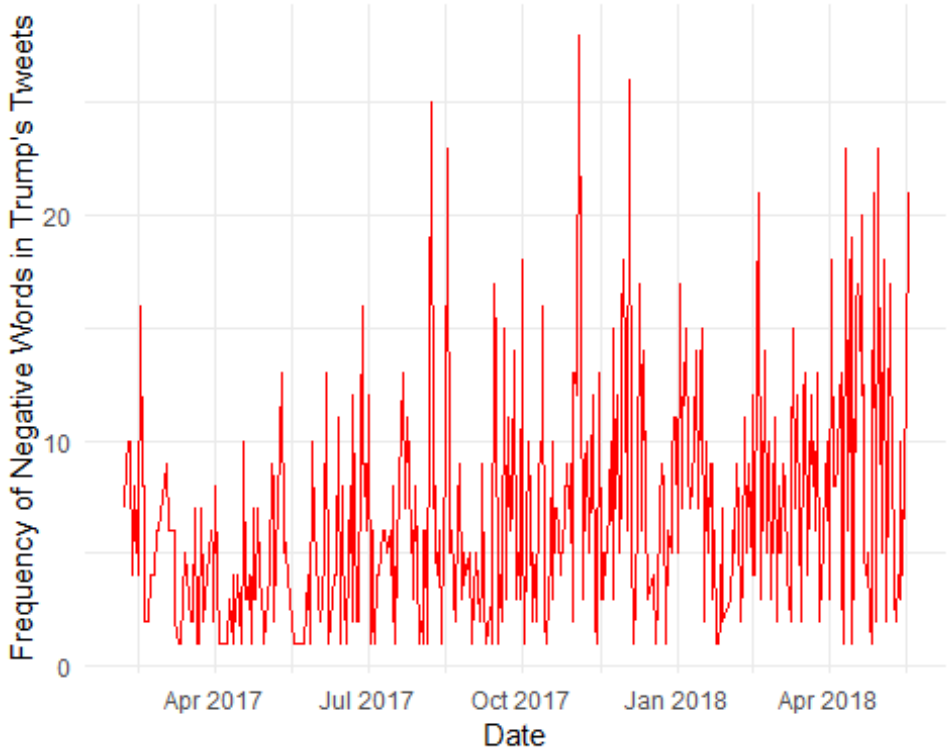
Aggregate negative sentiment by day:

```
trump_sentiment_plot <-  tidy_trump_tweets %>%
    inner_join(get_sentiments("bing")) %>%
      filter(sentiment=="negative") %>%
          count(date, sentiment)

## Joining, by = "word"

library(ggplot2)

ggplot(trump_sentiment_plot, aes(x=date, y=n))+
  geom_line(color="red")+
    theme_minimal()+
      ylab("Frequency of Negative Words in Trump's Tweets")+
        xlab("Date")
```



After the data was adjusted to be read in date form, the negative sentiment was aggregated by individual words and graphed onto a line graph. This provides a more clear visualization as to when the most negative sentiment was found in words. It appears that as time has gone on, the frequency of negative words appearing has increased.

The Document_Term Matrix

Create a document-term matrix:

```
tidy_trump_DTM<-  tidy_trump_tweets %>%
  count(created_at, word) %>%
  cast_dtm(created_at, word, n)
```

Topic Modeling

Find the common themes in the tweets. Apply the LDA() function and set the number of topics (k) to 3:

```
library(topicmodels)

## Warning: package 'topicmodels' was built under R version 3.5.3

topic_model<-LDA(tidy_trump_DTM, k=3, control = list(seed = 321))
```

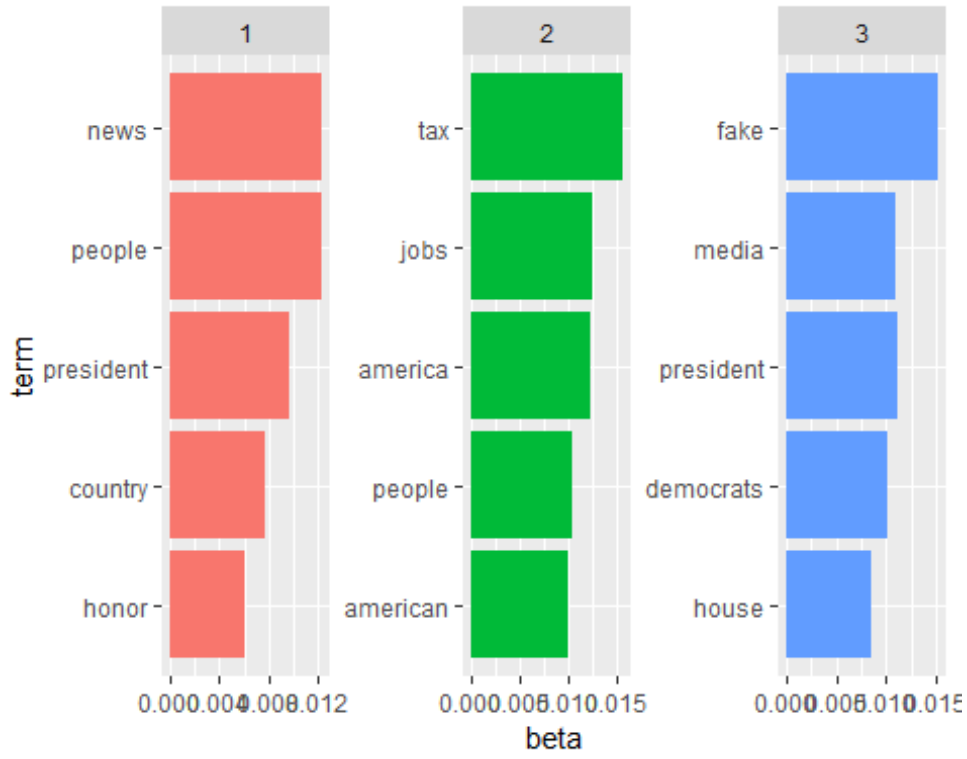
In order to discover common themes across the Tweets, the above code was run. It uss Latent Dirichlet Allocation (LDA) to look at each document as a mixture of topics and each

topic is viewed as a mixture of words. In this case, we separate the topics into 3 different sections.

```
topics <- tidy(topic_model, matrix = "beta")

top_terms <-
  topics %>%
  group_by(topic) %>%
  top_n(5, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)


top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```



After using LDA() command, the top 5 terms from each topic were selected and graphed on a bar graph in descending order, each topic being it's own unique color. "News", "Tax", "people", "jobs", "media" are all words one would assume would appear often in politcal context. "Fake" stood out to me as being unique most likely to Trump's presidency.

Word Correlations and Coocurance

Count how many times each pair of words occurs together in the tweets. {it may take a few minutes to run this chunck of code}

```
#Load the package widyr.
library(widyr)
```

```
## Warning: package 'widyr' was built under R version 3.5.3
```

```
word_pairs <- tidy_trump_tweets %>%
  pairwise_count(word, created_at, sort = TRUE)

word_pairs
```

```
## # A tibble: 259,336 x 3
##    item1 item2     n
##    <chr> <chr> <dbl>
##  1 news  fake    195
##  2 fake  news    195
##  3 cuts  tax      94
##  4 tax   cuts     94
##  5 media fake     79
##  6 fake  media    79
##  7 media news     66
##  8 news  media    66
##  9 white house    65
## 10 house white    65
## # ... with 259,326 more rows
```

Although looking at the most individually used words provides a lot of insight into the sentiment of a document, looking at words pairs can provide a new lense. The above code is used to look at combinations of word pairs and then aggregates them in descending order. Not suprising, many of the top word pairs are also the most common individual words. “Fake News”, “Tax Cuts”, “Fake Media”, and “White House” are the most frequently found word pairs in Trump’s Tweets. Being that media and news are essentially the same thing, Trump certainly mentions these outlets as being “fake” quite often.

Calculate the correlation among words in the comments field (filter for at least relatively common words first).{it may take a few minutes to run this chunck of code}

```
word_cors <- tidy_trump_tweets %>%
  group_by(word) %>%
  filter(n() >= 80) %>%
  pairwise_cor(word, created_at, sort = TRUE)

word_cors
```

```
## # A tibble: 1,406 x 3
##    item1 item2 correlation
```

```
##    <chr> <chr>       <dbl>
##  1 news  fake        0.808
##  2 fake  news        0.808
##  3 cuts  tax         0.717
##  4 tax   cuts        0.717
##  5 media fake        0.444
##  6 fake  media       0.444
##  7 media news        0.331
##  8 news  media       0.331
##  9 china trade       0.316
## 10 trade china       0.316
## # ... with 1,396 more rows
```

Not only is it important to look at word combinations, it is also pertinent to analyze the correrlation amongst these word pairs. The closer the correlation of the word pair is to 1, the stronger a relationship the two words have together. “Fake News” again tops the list with having the strongest correlation.

Plot networks of these correlations among words (use the ggraph package and the layout=”fr”)).

```
library (igraph)
```

```
## Warning: package 'igraph' was built under R version 3.5.3
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

```
## The following object is masked from 'package:tidyr':
##
##     crossing
```

```
## The following object is masked from 'package:tibble':
##
##     as_data_frame
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

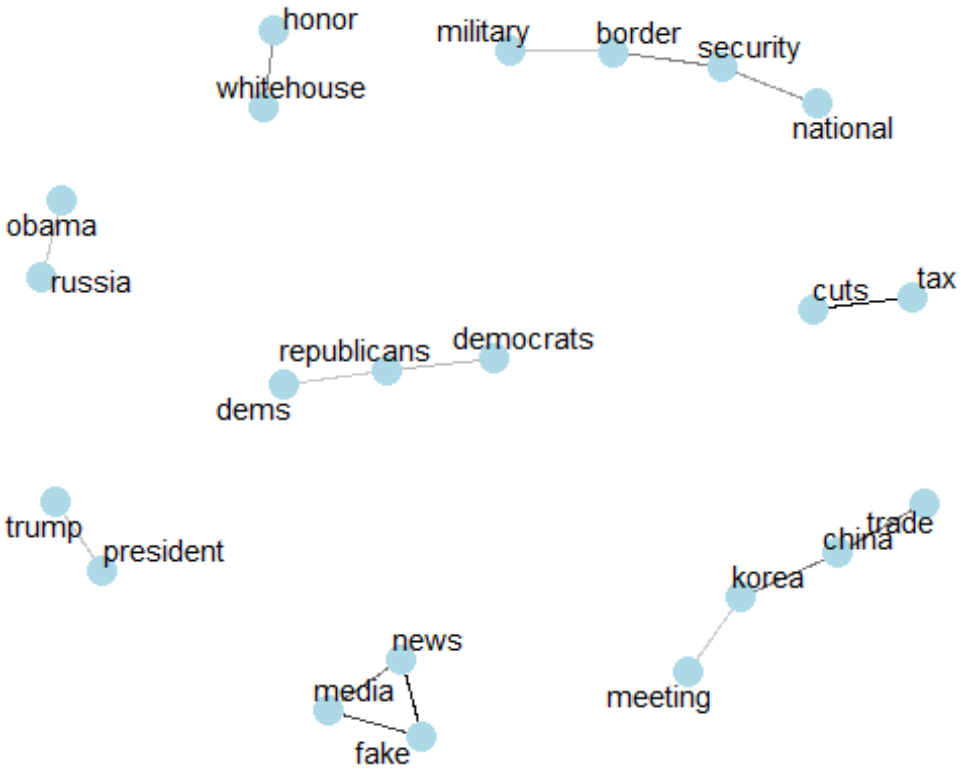
```
## The following object is masked from 'package:base':
##
##     union
```

```
library (ggraph)

## Warning: package 'ggraph' was built under R version 3.5.3

set.seed(2016)

word_cors %>%
  filter(correlation > .15) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()
```



In order to better visualize the correlation amongst the word pairs simulataneously, the graph_from_data_frame() command is used to create a word-flow map. This visual clearly shows connections between words like, “military”, “boarder”, “security”, and “national”. We can see other streams connecting “cuts” and “taxes” and of course the triangular connection between, “fake”, “news”, and “media”.

Bigrams

Tokenize by bigrams, filter the data, and find the most common bigrams in the tweets.

```
bigrams_filtered <- trumptweets %>%
  select(created_at,text) %>%
```

```
  unnest_tokens(bigram, text, token = "ngrams", n = 2)%>%
  separate(bigram, c("word1", "word2"), sep = " ")%>%
  filter(!word1 %in% stop_words$word,
         !nchar(word1) < 3,
         !str_detect(word1, "^\\b\\d+\\b"),
         !str_detect(word1, "\\s+"),
         !str_detect(word1, "[^a-zA-Z]"),
         !str_detect(word1,"https|t.co|amp|rt")) %>%
  filter(!word2 %in% stop_words$word,
         !nchar(word2) < 3,
         !str_detect(word2, "^\\b\\d+\\b"),
         !str_detect(word2, "\\s+"),
         !str_detect(word2, "[^a-zA-Z]"),
         !str_detect(word2,"https|t.co|amp|rt")) %>%
  count(word1, word2,  sort = TRUE)


bigrams_united <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ")
bigrams_united
```

```
## # A tibble: 7,579 x 2
##    bigram              n
##    <chr>           <int>
##  1 fake news         195
##  2 tax cuts           92
##  3 white house        65
##  4 news media         57
##  5 stock market       52
##  6 prime minister     45
##  7 jobs jobs          44
##  8 tax cut            41
##  9 witch hunt         40
## 10 crooked hillary    37
## # ... with 7,569 more rows
```

Bigram - Graph

Build a network of common bigrams [filter for only relatively common combinations (based on n) – use lines instead of directed arrows between nodes (graph_from_data_frame (directed = FALSE))].

```
bigrams<- trumptweets %>%
  select(created_at,text) %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)
bigrams
```

```
## # A tibble: 77,977 x 2
##    created_at          bigram
##    <dttm>              <chr>
```

```
##  1 2017-02-05 22:49:42 enjoy the
##  2 2017-02-05 22:49:42 the superbowl
##  3 2017-02-05 22:49:42 superbowl and
##  4 2017-02-05 22:49:42 and then
##  5 2017-02-05 22:49:42 then we
##  6 2017-02-05 22:49:42 we continue
##  7 2017-02-05 22:49:42 continue make
##  8 2017-02-05 22:49:42 make america
##  9 2017-02-05 22:49:42 america great
## 10 2017-02-05 22:49:42 great again
## # ... with 77,967 more rows
```

```
#Load the package igraph:
library(igraph)

# filter for only relatively common combinations
bigram_counts <- bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")%>%
  filter(!word1 %in% stop_words$word,
         !nchar(word1) < 3,
         !str_detect(word1, "^\\b\\d+\\b"),
         !str_detect(word1, "\\s+"),
         !str_detect(word1, "[^a-zA-Z]"),
         !str_detect(word1,"https|t.co|amp|rt")) %>%
  filter(!word2 %in% stop_words$word,
         !nchar(word2) < 3,
         !str_detect(word2, "^\\b\\d+\\b"),
         !str_detect(word2, "\\s+"),
         !str_detect(word2, "[^a-zA-Z]"),
         !str_detect(word2,"https|t.co|amp|rt")) %>%
  count(word1, word2, sort = TRUE)

bigram_counts
```

```
## # A tibble: 7,579 x 3
##    word1   word2        n
##    <chr>   <chr>    <int>
##  1 fake    news       195
##  2 tax     cuts        92
##  3 white   house       65
##  4 news    media       57
##  5 stock   market      52
##  6 prime   minister    45
##  7 jobs    jobs        44
##  8 tax     cut         41
##  9 witch   hunt        40
## 10 crooked hillary     37
## # ... with 7,569 more rows
```

```
bigram_graph <- bigram_counts %>%
  filter(n > 50) %>%
  graph_from_data_frame((directed = FALSE))

bigram_graph
```

```
## IGRAPH ea67fac UN-- 9 5 --
## + attr: name (v/c), n (e/n)
## + edges from ea67fac (vertex names):
## [1] fake --news   tax  --cuts   white--house  news --media  stock--market
```

The graph_from_dataframe function within the igraph package, takes the data frame of edges with columns for “from”, “to”, and edge attributes.This can be useful when trying to visualize words simultaneously as compared to comparing just the top ranked at a time. By having the directed set equal to false, we are able to eliminate the directional arrows normally found in an igraph output.

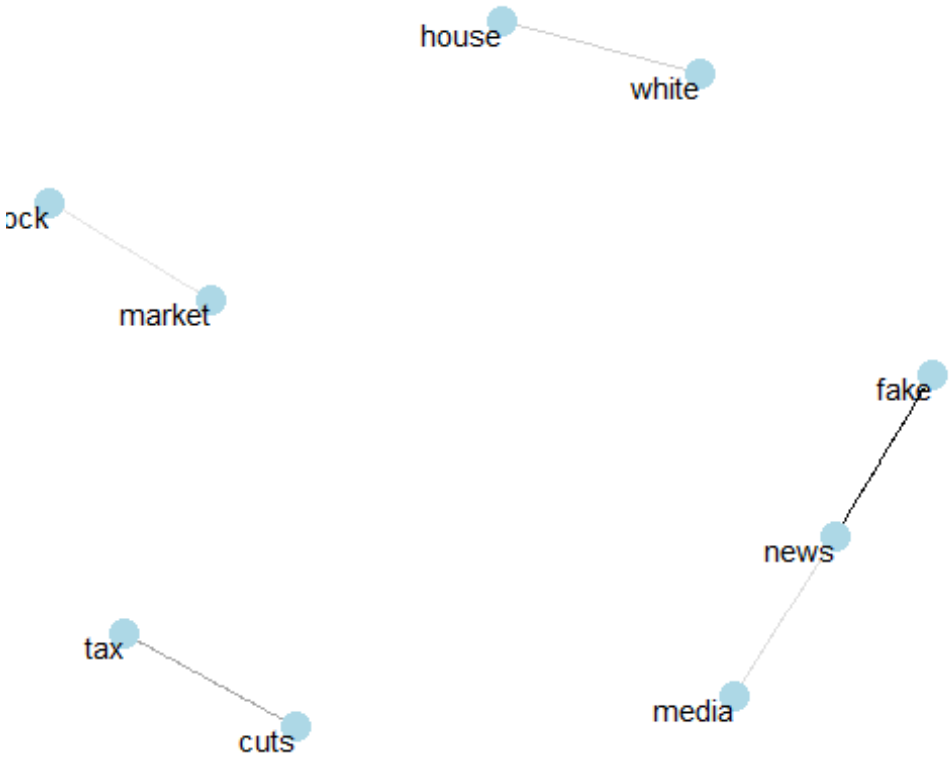
Visualize the graph - Use the Fruchterman-Reingold to visualize the nodes and ties (“fr”).

- add the edge_alpha aesthetic to the link layer to make links transparent based on how common or rare the bigram is
- add the edge_width aesthetic to the link layer to show the weight of the ties between bigrams
- tinker with the options to the node layer to make the nodes more attractive (larger, blue points)
- add a theme that’s useful for plotting networks, theme_void()

```
#Load the package ggraph
library(ggraph)


set.seed(2016)

ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 end_cap = circle(.07, 'inches')) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  theme_void()
```



Even though the igraph package has functions able to create plots, it is not its' optimal task. A superior method for visualizing the network of words is using the ggraph package found in the code above. The ggraph package is able to convert our igraph object andthen can add layers to the ggraph like nodes, edges, and text. The results show a word map of common phrases and nodes.

In looking at the visualization of the bigrams we can again see the connection between words like "tax" and "cuts", "house" and "white", and of course, "fake", "news" and "media".