Tweet Assignment - Widyr

Tabitha Hagen

4/19/2023

1 - Load the libraries

2 - Read-in the following datasets in R (the dataset is available on Canvas).

```
#read in the data
tweets_original <- read_csv("Twitter Data-Corona.csv")

# tidy and view data
tweets <- tweets_original %>%
    filter(nonrts.lang=="en")%>%
     select (created_at=nonrts.created_at, text=nonrts.text)

#describe new dataframe
names(tweets) # gives the column/variable names

## [1] "created_at" "text"

dim(tweets)   # gives the observation/rows count

## [1] 964   2
```

3 - Create a tidytext dataset of Tweets - Tokenize by unigrams (one-word-per-row). (10 points)

```
#Create a tidytext dataframe
tidy_unigrams <- tweets %>%
  unnest_tokens(word, text, token = "ngrams", n = 1) #create unigrams of 1
word

tidy_unigrams #view tidytext dataframe

## # A tibble: 15,974 × 2
##    created_at                    word
##    <chr>                         <chr>
##  1 Fri Mar 27 00:48:56 +0000 2020 at
##  2 Fri Mar 27 00:48:56 +0000 2020 least
##  3 Fri Mar 27 00:48:56 +0000 2020 he
##  4 Fri Mar 27 00:48:56 +0000 2020 s
```

```
##  5 Fri Mar 27 00:48:56 +0000 2020 being
##  6 Fri Mar 27 00:48:56 +0000 2020 honest
##  7 Fri Mar 27 00:48:56 +0000 2020 u
##  8 Fri Mar 27 00:48:56 +0000 2020 0001f937
##  9 Fri Mar 27 00:48:56 +0000 2020 u
## 10 Fri Mar 27 00:48:56 +0000 2020 0001f3fd
## # … with 15,964 more rows
```

4 - Pre-Process Text by Removing numbers, whitespaces, undesirable words, and stop words, words with less than 3 characters, etc.

```
undesirable_words <-
c("https","t.co","amp","rt","tco","coronavirus","corona","covid","virus",
"ass", "fucking", "fuck", "bitch", "0001f525", "horny", "tdk", "bro", "aye",
"neekolul", "12ct", "toilet", "ccp", "ccp_is_terrorist", "tempestwynn_",
"stopdlow", "q3lzl2crpd", "n04kfkdfia", "kmov", "fai", "bra", "3fle7en7nu",
"south32", "ya'll", "rendon63rd", "lorenaad80", "gavinnewsom", "ab5",
"anymore", "ocasio", "yay", "lululan09", "    tomforutah", "andrewyang",
"rand", "porky", "dab", "mrecmann", "hyundai", "alyssa_milano", "bts",
"threadreaderapp", "jamierodr14", "glov", "lmfaooo", "karaharagu1",
"jopbyfl", "iiqidaigw", "esvrmbfpb", "callmeanqiiex", "cabujnhnj",
"gqkzrfegc", "ylgukcir", "rjvgemll", "eumilsi", "sexier", "lvklqhs",
"afezlralb", "liijwmzo", "meatbjhzlf", "dwuhlfelderlaw", "lxzelujpz",
"coscklyxyr", "cvtjyjjkh", "kgqleuchk", "igwofruyc", "ofgihsj", "safjltdy",
"realcandaceo", "neqbz", "ivhfxjcp", "snoqxiwq", "euca", "oqsgopcm",
"couwmwxap", "ogxnxctmc", "mrtzgbshth", "dlvdbsxj", "ccpisterrorist",
"tempestwynn", "sgsuwbnuec", "pyxvjiprch", "aczzaq", "gxyjuhnuv", "qlzlcrpd",
"fboidbfbas", "kvlhnuue","", "muthafuckas")

tidy_unigrams$word <- gsub("[^a-zA-Z]","", tidy_unigrams$word) # only use
alpha characters
tidy_unigrams$word <- gsub("\\s+","", tidy_unigrams$word) # get rid of
whitespace



tidy_filtered <- tidy_unigrams %>%
  filter(!word %in% stop_words$word) %>%  # leave out stop_words
  filter(nchar(word) > 3)%>% # leave out words less than 3 characters
  filter(!word %in% undesirable_words) # leave out undesirable words

tidy_filtered
```

```
## # A tibble: 5,761 × 2
##    created_at                     word
##    <chr>                          <chr>
##  1 Fri Mar 27 00:48:56 +0000 2020 honest
##  2 Fri Mar 27 00:48:56 +0000 2020 septa
##  3 Fri Mar 27 00:48:56 +0000 2020 yall
##  4 Fri Mar 27 00:48:56 +0000 2020 greedy
```

```
##  5 Fri Mar 27 00:48:56 +0000 2020 america
##  6 Fri Mar 27 00:48:56 +0000 2020 issues
##  7 Fri Mar 27 00:48:56 +0000 2020 silly
##  8 Fri Mar 27 00:48:56 +0000 2020 estimated
##  9 Fri Mar 27 00:48:56 +0000 2020 figures
## 10 Fri Mar 27 00:48:56 +0000 2020 confirmed
## # … with 5,751 more rows
```

5 - Count the most frequently used words. (10 points)

```
# Count the most frequently used words.
tidy_counts <- tidy_filtered %>%
  count(word, sort = TRUE)
tidy_counts
```

```
## # A tibble: 3,568 × 2
##    word          n
##    <chr>     <int>
##  1 world        47
##  2 trump        46
##  3 people       44
##  4 china        39
##  5 pandemic     26
##  6 news         24
##  7 confirmed    23
##  8 care         17
##  9 country      17
## 10 bill         16
## # … with 3,558 more rows
```

6 - Load the package widyr.

```
library(widyr)
```

7 - Use pairwise_count() from the widyr package to count how many times each pair of words occurs together in the tweets. Explain the results.{it may take a few minutes to run this chunck of code} (20 points)

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

word_pairs
```

```
## # A tibble: 230,020 × 3
##    item1     item2         n
##    <chr>     <chr>     <dbl>
```

```
##  1 world     confirmed    13
##  2 people    trump        13
##  3 confirmed world        13
##  4 trump     people       13
##  5 china     world        12
##  6 world     china        12
##  7 trump     pandemic     10
##  8 people    pandemic     10
##  9 pandemic  trump        10
## 10 pandemic  people       10
## # … with 230,010 more rows
```

The word_pairs tibble shows us pairs of words that occur together and how many times they occur. The phrases repeated the most (13 times) are: “world confirmed”, “confirmed world”, “people Trump” and “Trump people”. This was followed by “people China” and “China people” (12 times).

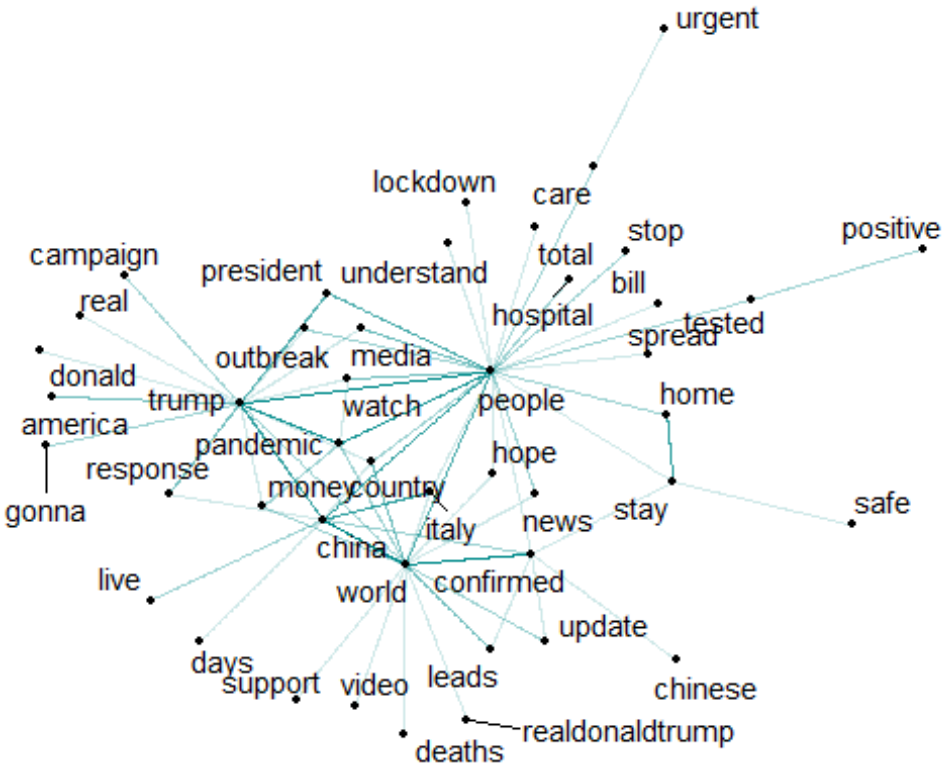
visualize the results. Filter the less common pairs. Explain the graph. (10 points)

```
library(igraph)
library(ggraph)

set.seed(2016)

word_pairs %>%
  filter(n > 4) %>%  # only use words that repeat more than 3 times
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
     #add edge_alpha to make links transparent based on how common or rare
the bigram is
     geom_edge_link(aes(edge_alpha = n), edge_colour = "cyan4", show.legend =
FALSE) +
     geom_node_point( size = 1) +  # size of each node
     geom_node_text(aes(label = name), repel = TRUE) +  # label nodes
     theme_void()  # apply a color theme
```

```
## Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2
3.4.0.
## ℹ Please use `linewidth` in the `default_aes` field and elsewhere instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



In this part, we are looking at certain words to analyze how common or rare the word pairs are. We are only looking at the less common words by choosing words that appear more than 4 times. We are using the color cyan with the higher level of transparency reflecting the less common or rare the words are. The most common words are: people, trump, china, world and confirmed.

8 - Use pairwise_cor() from the widyr package to calculate the correlation among words in the tweets (filter for at least relatively common words first n() >= 10).Explain the results.{it may take a few minutes to run this chunck of code}. (20 points)

```
word_cors <- tidy_filtered %>%
  group_by() %>%
  filter(n() >= 10) %>%
  pairwise_cor( word,created_at, sort=TRUE )

word_cors #view new tibble of the ϕ or phi-coefficient by Karl Pearson
```

```
## # A tibble: 12,727,056 × 3
##    item1     item2     correlation
##    <chr>     <chr>           <dbl>
##  1 brother   host                1
##  2 useless   worried             1
##  3 idea      approval            1
```

```
##  4 worried   useless             1
##  5 approval  idea                1
##  6 host      brother             1
##  7 institute letter              1
##  8 letter    institute           1
##  9 blames    kathy               1
## 10 kathy     blames              1
## # … with 12,727,046 more rows
```

9 - load the packages igraph and ggraph

```
library (igraph)
library (ggraph)
```

10 - Plot networks of these correlations among words (use the ggraph package and the layout="fr"")).Explain the results. (10 points)

Even on Coursera's Lab Sandbox, I could not install ggraph, igraph, tidytext or widyr. I tried doing this assignment there since I struggled with the memory for ggraph for the Twitter Data-Corona.csv" file. I also tried to create a smaller subset to make it work. I finally created the smallest subset to prove that my simplest graph could work.

```
# create a subset df from tidy data to conserve memory
word_cors_subset <- word_cors %>%
  filter(correlation > 0.5) # the correlation was increased from 0.2 to be
able to create graphs
word_cors_subset
```

```
## # A tibble: 111,934 × 3
##    item1     item2     correlation
##    <chr>     <chr>           <dbl>
##  1 brother   host                1
##  2 useless   worried             1
##  3 idea      approval            1
##  4 worried   useless             1
##  5 approval  idea                1
##  6 host      brother             1
##  7 institute letter              1
##  8 letter    institute           1
##  9 blames    kathy               1
## 10 kathy     blames              1
## # … with 111,924 more rows
```

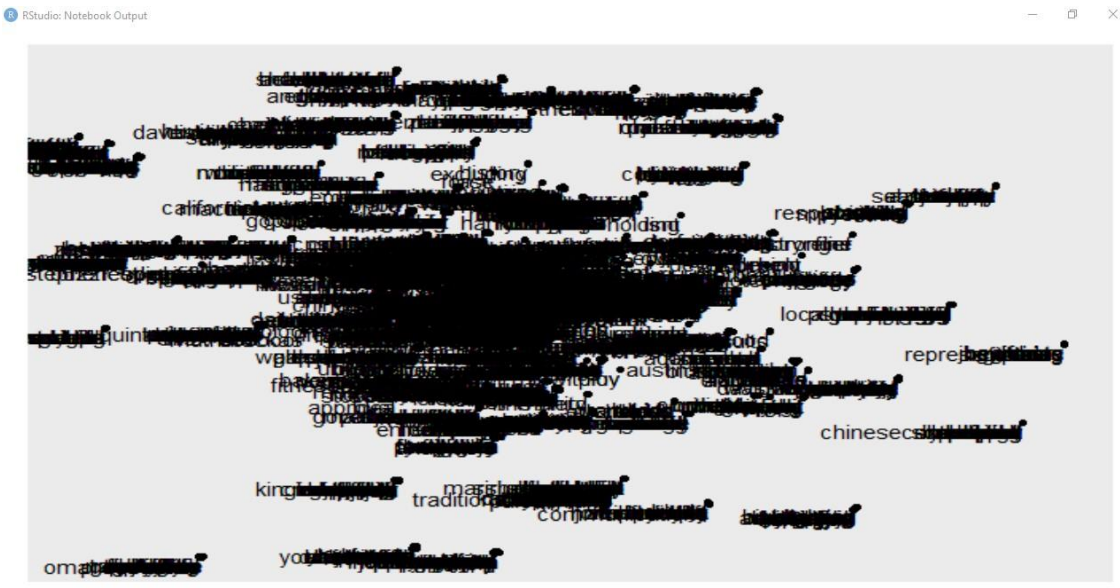
```
#save word_cors_subset to save memory
word_cors_subset <- word_cors %>%
  filter(correlation > 0.5)
write.csv(word_cors_subset,"saved_word_cors_subset.csv", row.names=FALSE )
#retrieve word_cors from external file
```

```
word_cors_graph <- read_csv("saved_word_cors_subset.csv")

# create an igraph object from tidy data
#word_cors_graph <- word_cors_subset
graph_from_data_frame(word_cors_graph)

## IGRAPH 19cc173 DN-- 3452 111934 --
## + attr: name (v/c), correlation (e/n)
## + edges from 19cc173 (vertex names):
##  [1] brother    ->host      useless    ->worried   idea       ->approval
##  [4] worried    ->useless   approval   ->idea      host       ->brother
##  [7] institute  ->letter    letter     ->institute blames     ->kathy
## [10] kathy      ->blames    desist     ->cease     cease      ->desist
## [13] bpats      ->pssg      brgy       ->pssg      vigan      ->pssg
## [16] implemented->pssg      executive  ->pssg      pssg       ->bpats
## [19] brgy       ->bpats     vigan      ->bpats     implemented->bpats
## [22] executive  ->bpats     pssg       ->brgy      bpats      ->brgy
## + ... omitted several edges

#Simple graph
#set.seed(1234)
#ggraph(word_cors_graph, layout = "fr") +  #convert the igraph object into a
ggraph object and add layers
     #geom_edge_link() +
     #geom_node_point() +
     #geom_node_text(aes(label = name), repel = TRUE)
knitr::include_graphics("Simple_word_cors_Network_Graph.jpg")
```




```
# advanced graph aesthetics
#set.seed(1234)
#ggraph(word_cors_graph, layout = "fr") +  #convert the igraph object into a
ggraph object and add layers
     #geom_edge_link(aes(edge_alpha = correlation), edge_colour = "cyan4",
```

```
show.legend = FALSE) +
     #geom_node_point(size = 1) +
     #geom_node_text(aes(label = name), check_overlap=TRUE) +
     #geom_node_text(aes(label = name), repel = TRUE) +  # label nodes +
     #theme_graph()
knitr::include_graphics("Advanced_word_cors_Network_Graph.jpg")
```


```
# errors:
# Using the 'size' aesthetic in this geom was deprecated in ggplot2 3.4.0.
# ggrepel: 3452 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```

11 - Reflect on this assignment and text networks.Provide a summary of your findings.(20 points)

In this assignment, we create unigrams, or single words, pair them up using pair_wise, and analyze their correlations to each other. This assignment needs a high memory machine to run the unigrams because of how many words are in the unigram dataframe. When you pair up item 1 to item 2 you also pair up item 2 to item 1, in essence, doubling your dataframe. Even when the attenpt to minimize the word correlation object to minimize memory, it was ineffective because there were simply too many words that the graphing could not handle all the objects. It did not matter whether the seed was changed, or whether the memory was contained using a saved file (so as not to let the previous work tie up the memory). Ultimately, there were unreadable graphs.