# Greeshma Chowdary Peddineni

mcu\_ethics <- readRDS("C:/Users/Dell/Downloads/mcu\_ethics.rds")

library(tidytext)

## Warning: package 'tidytext' was built under R version 4.4.3

library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)  
library(textdata)

## Warning: package 'textdata' was built under R version 4.4.3

library(wordcloud)

## Warning: package 'wordcloud' was built under R version 4.4.3

## Loading required package: RColorBrewer

library(reshape2)

## Warning: package 'reshape2' was built under R version 4.4.3

##   
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':  
##   
## smiths

library(tm)

## Warning: package 'tm' was built under R version 4.4.3

## Loading required package: NLP

##   
## Attaching package: 'NLP'

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

library(widyr)

## Warning: package 'widyr' was built under R version 4.4.3

library(igraph)

## Warning: package 'igraph' was built under R version 4.4.3

##   
## Attaching package: 'igraph'

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

## The following objects are masked from 'package:dplyr':  
##   
## as\_data\_frame, groups, union

## 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 4.4.3

get\_sentiments("nrc")

## # A tibble: 13,872 × 2  
## word sentiment  
## <chr> <chr>   
## 1 abacus trust   
## 2 abandon fear   
## 3 abandon negative   
## 4 abandon sadness   
## 5 abandoned anger   
## 6 abandoned fear   
## 7 abandoned negative   
## 8 abandoned sadness   
## 9 abandonment anger   
## 10 abandonment fear   
## # ℹ 13,862 more rows

get\_sentiments("bing")

## # A tibble: 6,786 × 2  
## word sentiment  
## <chr> <chr>   
## 1 2-faces negative   
## 2 abnormal negative   
## 3 abolish negative   
## 4 abominable negative   
## 5 abominably negative   
## 6 abominate negative   
## 7 abomination negative   
## 8 abort negative   
## 9 aborted negative   
## 10 aborts negative   
## # ℹ 6,776 more rows

get\_sentiments("afinn")

## # A tibble: 2,477 × 2  
## word value  
## <chr> <dbl>  
## 1 abandon -2  
## 2 abandoned -2  
## 3 abandons -2  
## 4 abducted -2  
## 5 abduction -2  
## 6 abductions -2  
## 7 abhor -3  
## 8 abhorred -3  
## 9 abhorrent -3  
## 10 abhors -3  
## # ℹ 2,467 more rows

# Let's find how many English tweets are there?   
mcuEN <- mcu\_ethics %>% filter(lang=="en")   
print(paste("English tweets are", count(mcuEN)))

## [1] "English tweets are 49859"

# What is the ratio of retweets to total tweets?  
retweets <- mcu\_ethics %>% filter(grepl("^RT", msg)) %>% nrow()  
total\_tweets <- nrow(mcu\_ethics)  
retweet\_ratio <- retweets / total\_tweets  
print(paste("Ratio of retweets to total tweets:", retweet\_ratio))

## [1] "Ratio of retweets to total tweets: 0.65786"

# How many tweets contain a hyperlink?  
hyperlink\_tweets <- mcu\_ethics %>% filter(grepl("http[s]?://", source)) %>% nrow()  
print(paste("Number of tweets containing a hyperlink:", hyperlink\_tweets))

## [1] "Number of tweets containing a hyperlink: 49857"

# what is the most popular “source” used to post?   
  
most\_popular\_source <- mcu\_ethics %>%   
 count(source, sort = TRUE) %>%   
 slice\_max(n, n = 1)  
  
cat("Most popular source:", most\_popular\_source$source, "\n")

## Most popular source: <a href=http://twitter.com/download/iphone>Twitter for iPhone</a>

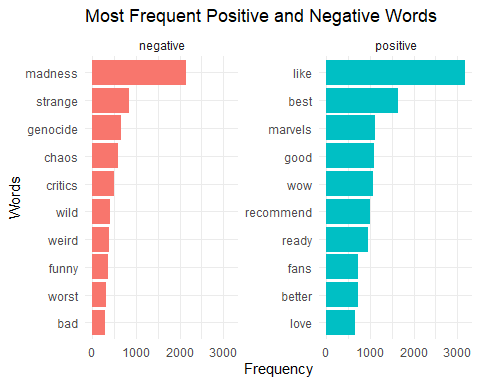
cat("Count:", most\_popular\_source$n, "\n")

## Count: 19351

# Create a bar graph of most frequent Positive and Negative words (using BING)  
# Load the Bing sentiment lexicon  
bing\_lexicon <- get\_sentiments("bing")  
  
tweet\_words <- mcu\_ethics %>%   
 unnest\_tokens(word, msg) %>%   
 filter(word != "marvel") %>% # Remove the word "marvel, because we cannot say it is a positive or negative word."  
 inner\_join(bing\_lexicon, by = "word")

## Warning in inner\_join(., bing\_lexicon, by = "word"): Detected an unexpected many-to-many relationship between `x` and `y`.  
## ℹ Row 292916 of `x` matches multiple rows in `y`.  
## ℹ Row 3805 of `y` matches multiple rows in `x`.  
## ℹ If a many-to-many relationship is expected, set `relationship =  
## "many-to-many"` to silence this warning.

# Get top 10 positive and negative words separately  
top\_positive <- tweet\_words %>%  
 filter(sentiment == "positive") %>%  
 count(word, sentiment, sort = TRUE) %>%  
 top\_n(10, n)  
  
top\_negative <- tweet\_words %>%  
 filter(sentiment == "negative") %>%  
 count(word, sentiment, sort = TRUE) %>%  
 top\_n(10, n)  
  
# Combine both into one dataframe  
top\_words <- bind\_rows(top\_positive, top\_negative)  
  
# Ensure sentiment is a factor to avoid issues in facet\_wrap  
top\_words$sentiment <- factor(top\_words$sentiment, levels = c("negative", "positive"))  
  
# Create bar plot  
ggplot(top\_words, aes(x = reorder(word, n), y = n, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 coord\_flip() +  
 labs(title = "Most Frequent Positive and Negative Words", x = "Words", y = "Frequency") +  
 theme\_minimal()



# Let's create a wordcloud to get most frequent positive and negative words  
library(tm)  
library(textdata)  
  
tweet\_words %>%  
 count(word, sentiment, sort = TRUE) %>%  
 acast(word ~ sentiment, value.var = "n", fill = 0) %>%  
 comparison.cloud(colors = c("red", "blue"), scale = c(4, 1),  
 random.order = FALSE, max.words = 60)

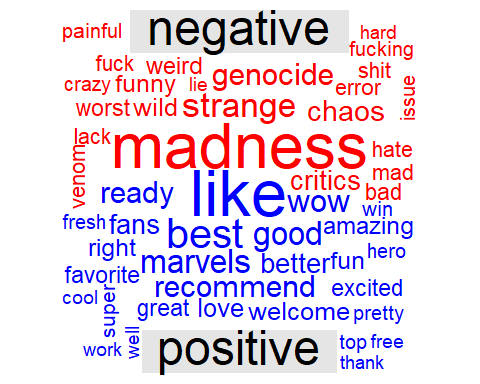
## Warning in comparison.cloud(., colors = c("red", "blue"), scale = c(4, 1), :  
## refreshing could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(., colors = c("red", "blue"), scale = c(4, 1), :  
## wrong could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(., colors = c("red", "blue"), scale = c(4, 1), :  
## overloaded could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(., colors = c("red", "blue"), scale = c(4, 1), :  
## dark could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(., colors = c("red", "blue"), scale = c(4, 1), :  
## disrespectful could not be fit on page. It will not be plotted.



## Warning in comparison.cloud(., colors = c("red", "blue"), scale = c(4, 1), :  
## happy could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(., colors = c("red", "blue"), scale = c(4, 1), :  
## breaking could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(., colors = c("red", "blue"), scale = c(4, 1), :  
## available could not be fit on page. It will not be plotted.

## Warning in comparison.cloud(., colors = c("red", "blue"), scale = c(4, 1), :  
## obsession could not be fit on page. It will not be plotted.

# Let's do create a network diagram (adjust the filter so that individual words are legible. ie: don't have a tangled mess of a plot)   
tweet\_bigrams <- mcu\_ethics %>%  
 unnest\_tokens(bigram, msg, token = "ngrams", n = 2) %>%  
 separate(bigram, into = c("word1", "word2"), sep = " ") %>%  
 filter(!is.na(word1) & !is.na(word2)) %>%  
 count(word1, word2, sort = TRUE)  
  
# Filter for frequent bigrams   
sotuBigramCount <- tweet\_bigrams %>% filter(n >= 2000)  
  
# Convert to graph object  
graph <- graph\_from\_data\_frame(sotuBigramCount)  
  
# Create network plot  
ggraph(graph, layout = "fr") +  
 geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n), show.legend = FALSE) +  
 geom\_node\_point(color = "darkslategray4", size = 3) +  
 geom\_node\_text(aes(label = name), vjust = 1.8, size = 3) +  
 labs(title = "Word Network: Tweets",  
 subtitle = "Text Mining of Tweet Bigrams",  
 x = "", y = "") +  
 theme\_void()

## Warning: The `trans` argument of `continuous\_scale()` is deprecated as of ggplot2 3.5.0.  
## ℹ Please use the `transform` argument instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

