Detecting Fake News Using Sentiment Analysis

Andrea Boskovic and Peter Cho

12/18/2018

```
# Loading the dataset
fake <- read_csv("fake.csv") #all fake
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   ord_in_thread = col_integer(),
##   published = col_datetime(format = ""),
##   crawled = col_datetime(format = ""),
##   domain_rank = col_integer(),
##   spam_score = col_double(),
##   replies_count = col_integer(),
##   participants_count = col_integer(),
##   likes = col_integer(),
##   comments = col_integer(),
##   shares = col_integer()
## )
```

```
## See spec(...) for full column specifications.
```

```
real <- read_csv("Articles.csv") #all real
```

```
## Parsed with column specification:
## cols(
##   Article = col_character(),
##   Date = col_character(),
##   Heading = col_character(),
##   NewsType = col_character()
## )
```

```
new_ds <- read_csv("data.csv") #combination of real and fake
```

```
## Parsed with column specification:
## cols(
##   URLs = col_character(),
##   Headline = col_character(),
##   Body = col_character(),
##   Label = col_integer()
## )
```

```
fake_type <- c("fake", "satire", "bias", "bs", "conspiracy", "state", "junksci", "hate")
real_type <- c("sports", "business")

# Merging the datasets and removing unnecessary columns
real <- real %>%
  mutate(binary_type = ifelse(NewsType %in% fake_type, 0, 1)) #now fake = 0 and real = 1
fake <- fake %>%
  mutate(binary_type = ifelse(type %in% fake_type, 0, 1)) #now fake = 0 and real = 1
new_ds <- new_ds %>%
  filter(Label == 1)
```

```r
real <- full_join(real, new_ds, by = c("Heading" = "Headline", "Article" = "Body", "binary_type" = "Labe
real <- real %>%
  mutate(id = as.character(seq(1:4564))) %>%
  mutate(realtype = "real")

# Making a combined dataset with both fake and real articles and selecting only for the uuid (unique id
combined <- full_join(fake, real, by = c("text" = "Article", "title" = "Heading", "uuid" = "id", "binary
  select(uuid, binary_type, type, title, text)

# Making a tidy dataset where we have the the words in their own column for facilitated data analysis a
tidy_combined <- combined %>%
  unnest_tokens(word, text)
```

```r
# Basic Data Exploration:

# This allows us to see how many observations are in each type of fake news.
combined %>%
  group_by(type) %>%
  summarize(n = n())
```

```
## # A tibble: 9 x 2
##   type           n
##   <chr>      <int>
## 1 bias         443
## 2 bs         11492
## 3 conspiracy   430
## 4 fake          19
## 5 hate         246
## 6 junksci      102
## 7 real        4564
## 8 satire       146
## 9 state        121
```

```r
typetotals <- combined %>%
  group_by(type) %>%
  summarize(n = n())
```

```r
# What are the most common words for each basic emotion?
# We will use the nrc lexicon to categorize each documented word into on of the basic human emotions ca

# Anger
nrc_anger <- get_sentiments("nrc") %>%
  filter(sentiment == "anger")

tidy_combined %>%
  inner_join(nrc_anger) %>%
  count(word, sort = TRUE)
```

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

```
## # A tibble: 1,220 x 2
##    word         n
##    <chr>    <int>
##  1 vote      4969
##  2 money     4835
##  3 force     3189
```

```
##   4 court     2721
##   5 attack    2548
##   6 defense   2242
##   7 death     2176
##   8 bad       2175
##   9 politics  2058
##  10 fight     2054
## # ... with 1,210 more rows
```

```r
# Fear
nrc_fear <- get_sentiments("nrc") %>%
  filter(sentiment == "fear")

tidy_combined %>%
  inner_join(nrc_fear) %>%
  count(word, sort = TRUE)
```

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

## # A tibble: 1,430 x 2
##    word           n
##    <chr>      <int>
##  1 government 11656
##  2 war         9845
##  3 military    5880
##  4 police      4902
##  5 change      4442
##  6 case        4177
##  7 force       3189
##  8 court       2721
##  9 attack      2548
## 10 problem     2381
## # ... with 1,420 more rows
```

```r
# Anticipation
nrc_anticipation <- get_sentiments("nrc") %>%
  filter(sentiment == "anticipation")

tidy_combined %>%
  inner_join(nrc_anticipation) %>%
  count(word, sort = TRUE)
```

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

## # A tibble: 816 x 2
##    word              n
##    <chr>         <int>
##  1 time          14159
##  2 white          6547
##  3 public         6039
##  4 good           5802
##  5 long           5706
##  6 vote           4969
##  7 money          4835
##  8 investigation  3968
##  9 top            3822
## 10 continue       3439
```

```
## # ... with 806 more rows
```

```r
# Trust
nrc_trust <- get_sentiments("nrc") %>%
  filter(sentiment == "trust")

tidy_combined %>%
  inner_join(nrc_trust) %>%
  count(word, sort = TRUE)
```

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

```
## # A tibble: 1,191 x 2
##    word          n
##    <chr>     <int>
##  1 president 12344
##  2 united     7803
##  3 white      6547
##  4 good       5802
##  5 law        5181
##  6 system     5088
##  7 vote       4969
##  8 police     4902
##  9 money      4835
## 10 fact       4673
## # ... with 1,181 more rows
```

```r
# Surprise
nrc_surprise <- get_sentiments("nrc") %>%
  filter(sentiment == "surprise")

tidy_combined %>%
  inner_join(nrc_surprise) %>%
  count(word, sort = TRUE)
```

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

```
## # A tibble: 518 x 2
##    word      n
##    <chr> <int>
##  1 trump 23953
##  2 good   5802
##  3 vote   4969
##  4 money  4835
##  5 deal   2802
##  6 death  2176
##  7 leave  2080
##  8 hope   1902
##  9 young  1859
## 10 shot   1604
## # ... with 508 more rows
```

```r
# Sadness
nrc_sadness <- get_sentiments("nrc") %>%
  filter(sentiment == "sadness")

tidy_combined %>%
```

```r
  inner_join(nrc_sadness) %>%
  count(word, sort = TRUE)
```

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

## # A tibble: 1,151 x 2
##    word          n
##    <chr>     <int>
##  1 vote       4969
##  2 black      4196
##  3 case       4177
##  4 problem    2381
##  5 lost       2260
##  6 tax        2211
##  7 death      2176
##  8 bad        2175
##  9 leave      2080
## 10 violence   1955
## # ... with 1,141 more rows
```

```r
# Joy
nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")

tidy_combined %>%
  inner_join(nrc_joy) %>%
  count(word, sort = TRUE)
```

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

## # A tibble: 668 x 2
##    word      n
##    <chr> <int>
##  1 white  6547
##  2 good   5802
##  3 vote   4969
##  4 money  4835
##  5 found  4192
##  6 share  3090
##  7 deal   2802
##  8 food   2756
##  9 pay    2339
## 10 true   2234
## # ... with 658 more rows
```

```r
# Disgust
nrc_disgust <- get_sentiments("nrc") %>%
  filter(sentiment == "disgust")

tidy_combined %>%
  inner_join(nrc_disgust) %>%
  count(word, sort = TRUE)
```

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

## # A tibble: 1,023 x 2
##    word            n
```

```
##    <chr>      <int>
##  1 john        3108
##  2 congress    2473
##  3 death       2176
##  4 bad         2175
##  5 criminal    1805
##  6 illegal     1756
##  7 powerful    1611
##  8 corruption  1571
##  9 finally     1442
## 10 remains     1244
## # ... with 1,013 more rows
```

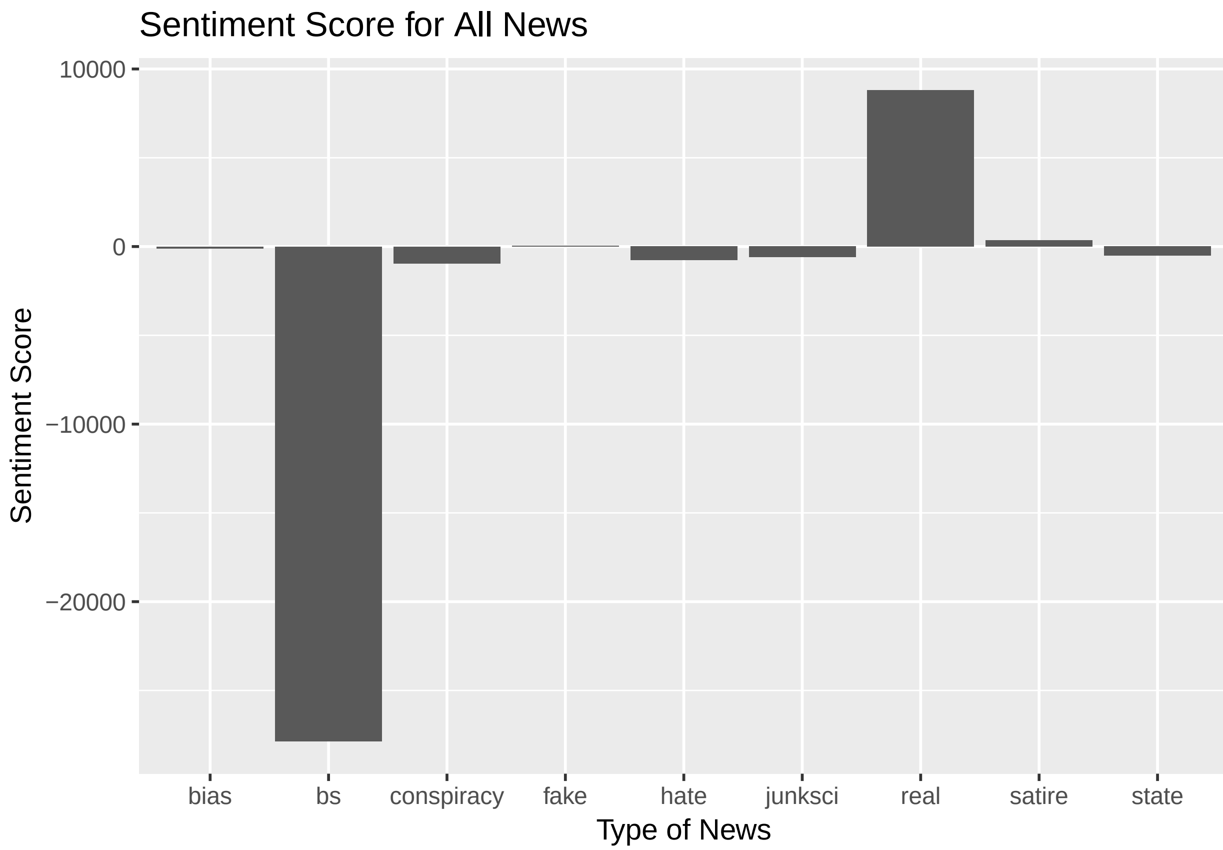
```
# Find net sentiment for each type of fake news documented in the dataset using the bing lexicon. The b
# Note that some types, such as bs (> 400,000), have more corresponding observations than other types,
combined_sentiment <- tidy_combined %>%
  inner_join(get_sentiments("bing")) %>%
  count(type, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

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

```
combined_sentiment
```

```
## # A tibble: 9 x 4
##   type       negative positive sentiment
##   <chr>         <dbl>    <dbl>     <dbl>
## 1 bias           5422     5322      -100
## 2 bs           247391   219536    -27855
## 3 conspiracy     4805     3851      -954
## 4 fake            148      199        51
## 5 hate           8765     7998      -767
## 6 junksci        3070     2469      -601
## 7 real          45896    54690      8794
## 8 satire         1148     1487       339
## 9 state          1215      704      -511
```

```
# Plot of the sentiment score for each type of news
ggplot(combined_sentiment, aes(x = type, y = sentiment)) + geom_col() + labs(title = "Sentiment Score fo
```




```
# We can also get the sentiment score on a scale of -5 to 5 from the AFINN lexicon. The AFINN lexicon h
afinn <- tidy_combined %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(type) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(method = "AFINN")
```

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

```
afinn
```

```
## # A tibble: 9 x 3
##   type       sentiment method
##   <chr>          <int> <chr>
## 1 bias           -1507 AFINN
## 2 bs            -62021 AFINN
## 3 conspiracy     -1846 AFINN
## 4 fake             108 AFINN
## 5 hate           -1625 AFINN
## 6 junksci           41 AFINN
## 7 real           28457 AFINN
## 8 satire           868 AFINN
## 9 state          -1089 AFINN
```

```
# It may be useful to investigate the basic contents of the lexicons.

# Positive and negative words in nrc lexicon
get_sentiments("nrc") %>%
     filter(sentiment %in% c("positive",
```

```r
                              "negative")) %>%
  count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   3324
## 2 positive   2312
```

```r
# Positive and negative words in bing lexicon
get_sentiments("bing") %>%
  count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   4782
## 2 positive   2006
```

```r
# Both lexicons have more negative words than positive words, but the bing lexicon has a higher ratio o
```

```r
# Counting the most frequently appearing words and which sentiment they correspond to (positive or nega
bing_word_counts <- tidy_combined %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

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

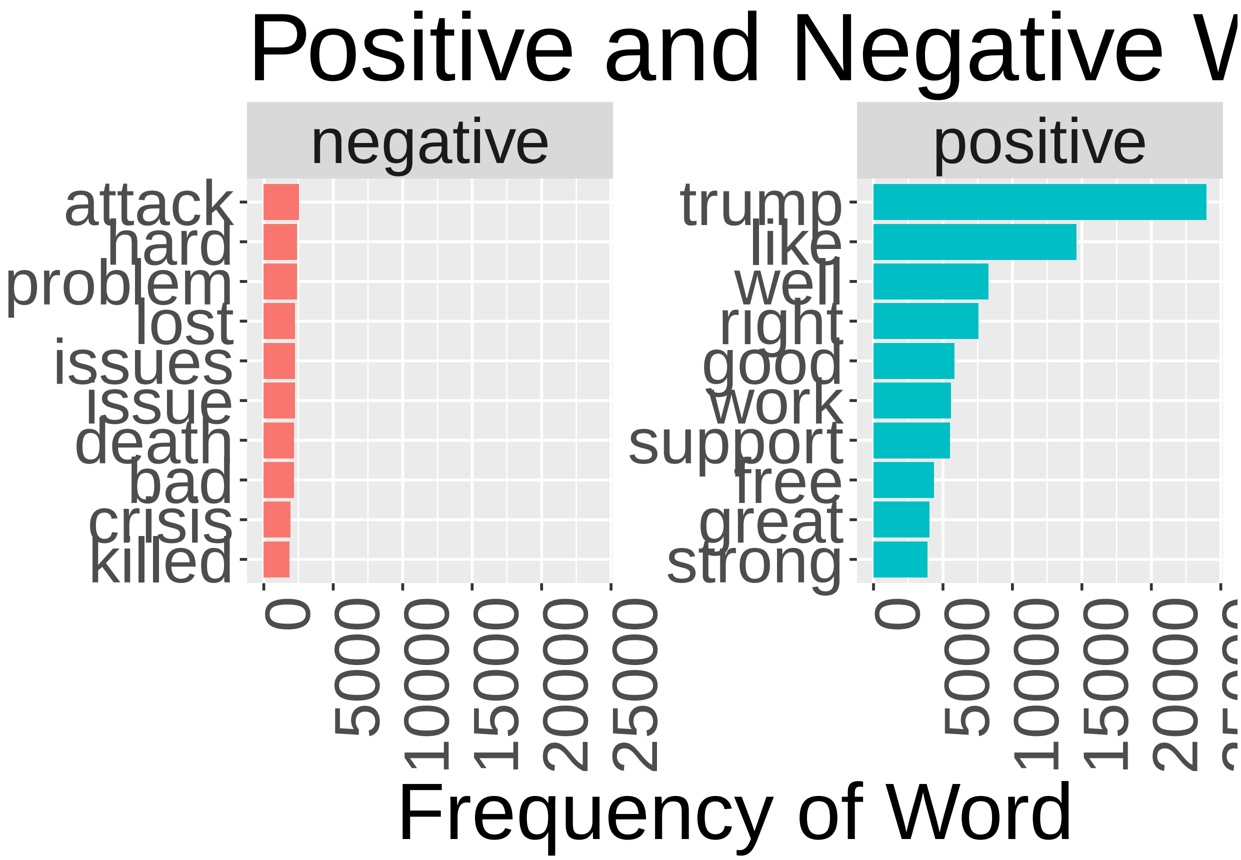
```r
bing_word_counts
```

```
## # A tibble: 5,552 x 3
##    word    sentiment     n
##    <chr>   <chr>     <int>
##  1 trump   positive  23953
##  2 like    positive  14612
##  3 well    positive   8250
##  4 right   positive   7530
##  5 good    positive   5802
##  6 work    positive   5544
##  7 support positive   5504
##  8 free    positive   4327
##  9 great   positive   4007
## 10 strong  positive   3862
## # ... with 5,542 more rows
```

```r
bing_word_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  ggtitle("Positive and Negative Word Frequency") +
  labs(y = "Frequency of Word",
       x = NULL) +
```

```r
  theme(text = element_text(size=30),
        axis.text.x = element_text(angle=90, hjust=1)) +
  coord_flip()
```

```
## Selecting by n
```

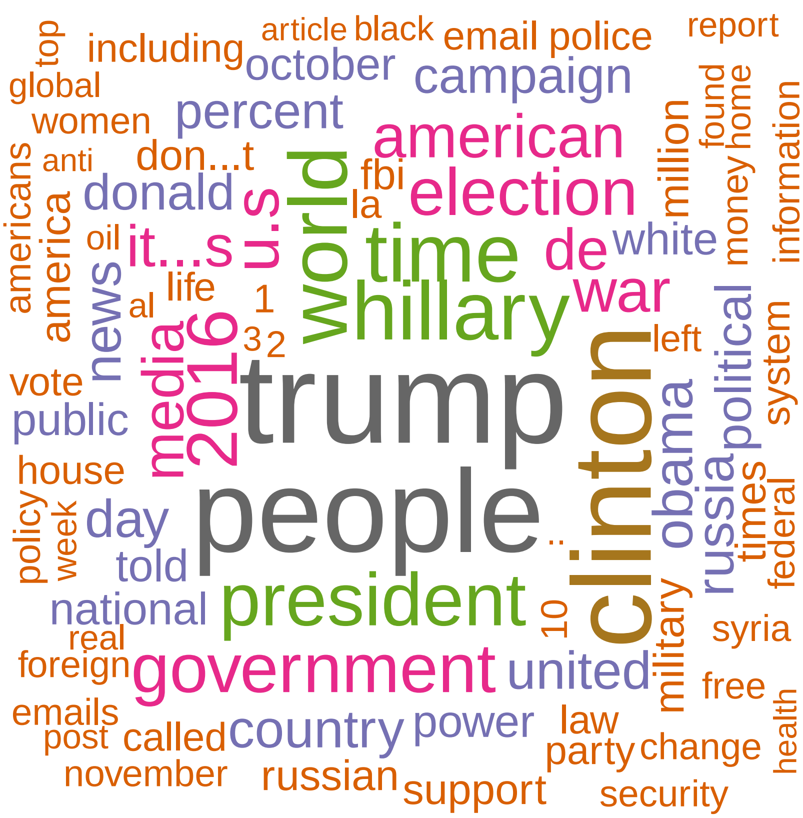



```r
# Wordcloud with most frqeuently appearing words
tidy_combined %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(words = word, freq = n, max.words = 100, min.freq = 1, random.order=FALSE, rot.per = 0
```

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

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'it’s' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'it’s' in 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'it’s' in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'it’s' in 'mbcsToSbcs': dot substituted
## for <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'it’s' in 'mbcsToSbcs': dot substituted
## for <80>
```

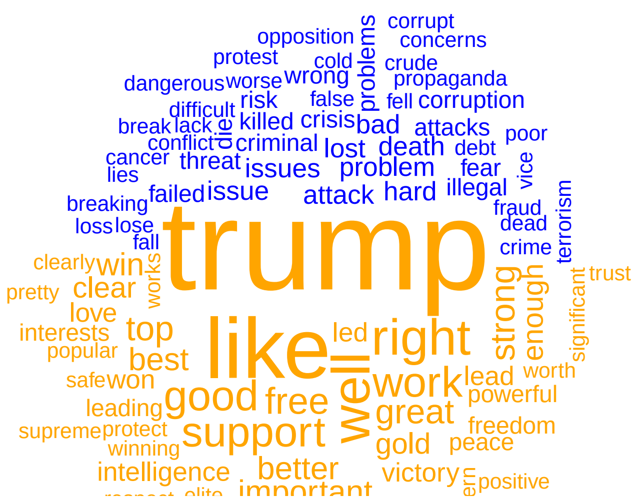
```
## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'it’s' in 'mbcsToSbcs': dot substituted
## for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'don’t' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'don’t' in 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'don’t' in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted
## for <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted
## for <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted
## for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : presidential could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : washington could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : international could not be fit on page. It will not be plotted.

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## ' ' in 'mbcsToSbcs': dot substituted for <d0>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## ' ' in 'mbcsToSbcs': dot substituted for <b2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on ' ' in 'mbcsToSbcs': dot substituted for
## <d0>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on ' ' in 'mbcsToSbcs': dot substituted for
## <b2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+0432

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : investigation could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : democratic could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : control could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : york could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : days could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : economic could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : strong could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : story could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : china could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : countries could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : human could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : history could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = word, freq = n, max.words = 100, min.freq =
## 1, : team could not be fit on page. It will not be plotted.
```



```
# Wordcloud faceted into positive and negative with color (blue corresponds to a negative sentiment whi
tidy_combined %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("blue", "orange"),
                   max.words = 100)
```

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




```
# Now, it is time to start the machine learning aspect of the project.

# Using the AFINN lexicon to append the sentiment score of each word to a new dataset called tidy_combi
tidy_combined_a <- tidy_combined %>%
  inner_join(get_sentiments("afinn"))
```

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

```
# Categorize article as positive or negative overall based on the average of the AFINN score of the wor
tidy_combined_final <- tidy_combined_a %>%
  select(uuid, score, binary_type) %>%
  group_by(uuid) %>%
  summarise(n_words = n(), avgscore = sum(score) / n_words,
            type = mean(binary_type),
            positive_score = sum(score[score > 0]),
            negative_score = sum(score[score < 0]),
            n_positive = sum(score > 0),
            n_negative = sum(score < 0)
            ) %>%
  mutate(articlesent = ifelse(avgscore < 0, "Negative", "Positive")) %>%
  mutate(txt_type = as.factor(type)) %>%
  select(-type)
tidy_combined_final
```

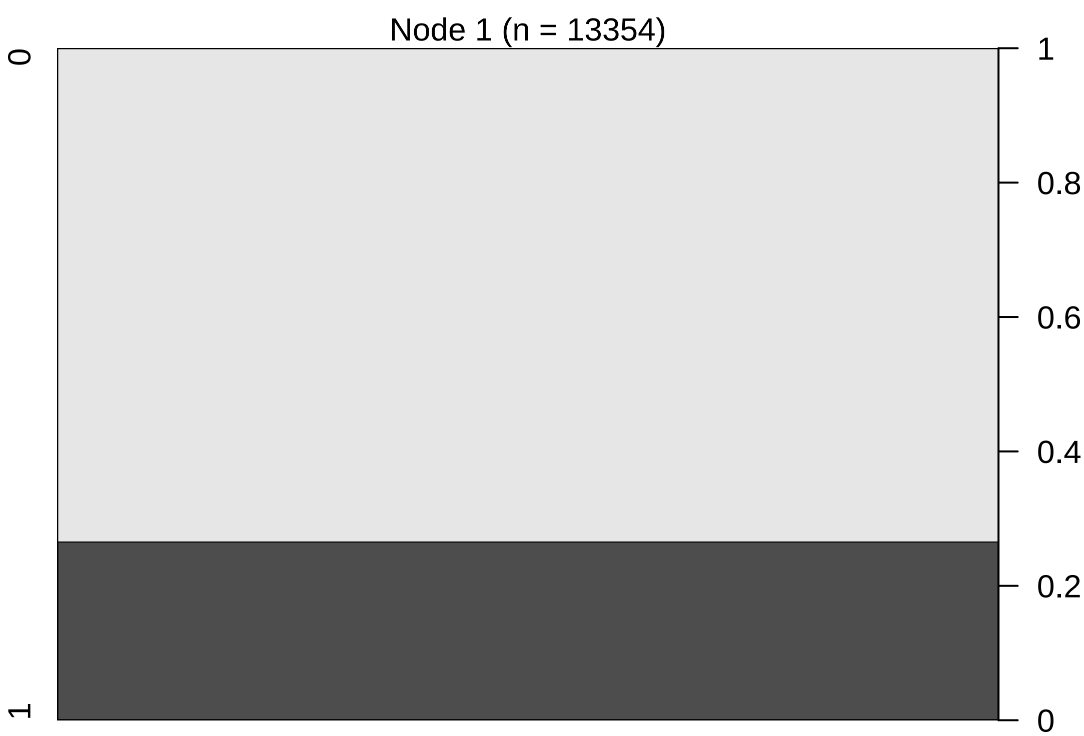
```
## # A tibble: 16,693 x 9
##    uuid  n_words avgscore positive_score negative_score n_positive
##    <chr>   <int>    <dbl>          <int>          <int>      <int>
##  1 0005~      21    0.286             19            -13         13
```

```
##   2 0020~      24   -0.667            12           -28          7
##   3 0021~      87    0.379           109           -76         49
##   4 002d~      88    0.261            99           -76         50
##   5 0033~       9    0                 8            -8          5
##   6 0033~      58   -0.759            36           -80         20
##   7 0037~      14    0.714            16            -6          8
##   8 0038~      30   -0.667            20           -40          9
##   9 003d~      10    0.7              14            -7          7
##  10 0048~      58    0.172            50           -40         34
## # ... with 16,683 more rows, and 3 more variables: n_negative <int>,
## #   articlesent <chr>, txt_type <fct>
```

```
tidy_combined_final %>%
  filter(txt_type == 0) %>%
  summarise(n_negative = n())
```

```
## # A tibble: 1 x 1
##   n_negative
##        <int>
## 1      12248
```

```
# Decision tree training process
n <- nrow(tidy_combined_final)
train_id <- sample(1:n, size = round(n * 0.8))
train <- tidy_combined_final[train_id,]
test <- tidy_combined_final[-train_id,]


tree <- rpart(txt_type ~ avgscore + n_words + n_positive + n_negative + negative_score + positive_score
plot(as.party(tree))
```



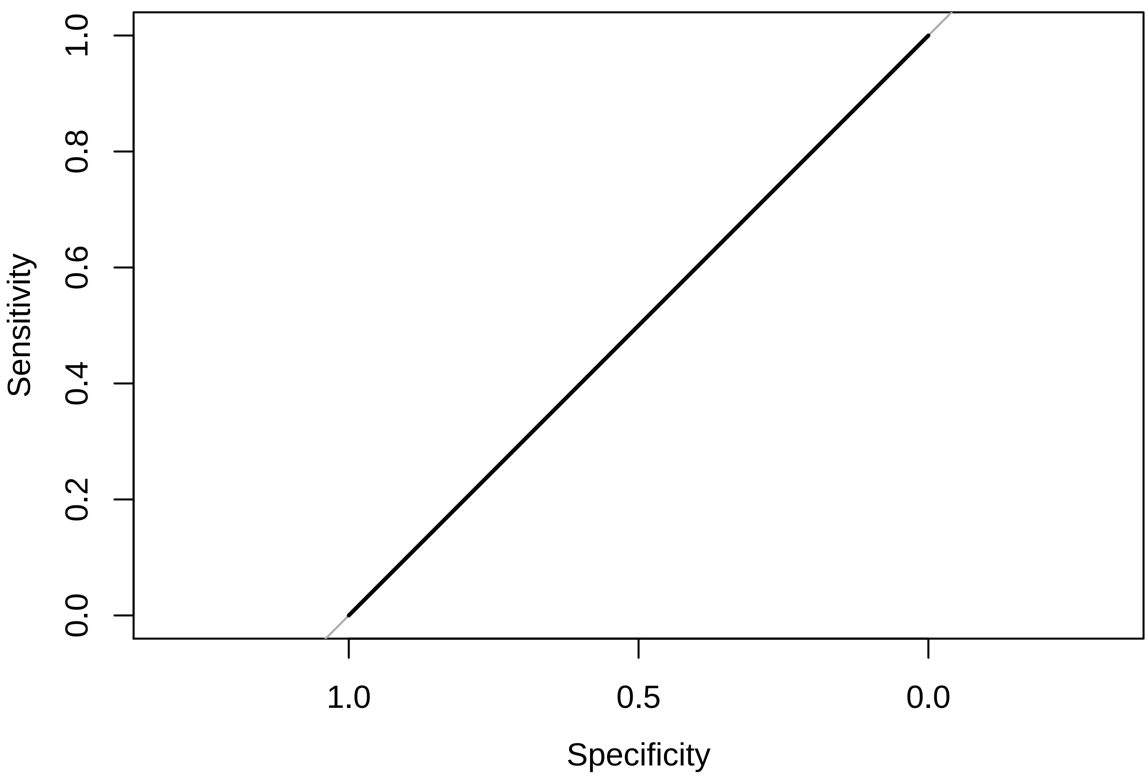
```
tree
```

```
## n= 13354
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 13354 3550 0 (0.7341620 0.2658380) *
```

```
saveRDS(tree, file = "tree.rds")
saveRDS(train, file = "train.rds")
prediction <- predict(tree, test)

test <- test %>%
  mutate(prediction = prediction[1])
roc_obj <- roc(test$txt_type, test$prediction)
auc(roc_obj)
```

```
## Area under the curve: 0.5
```

```
plot(roc_obj)
```

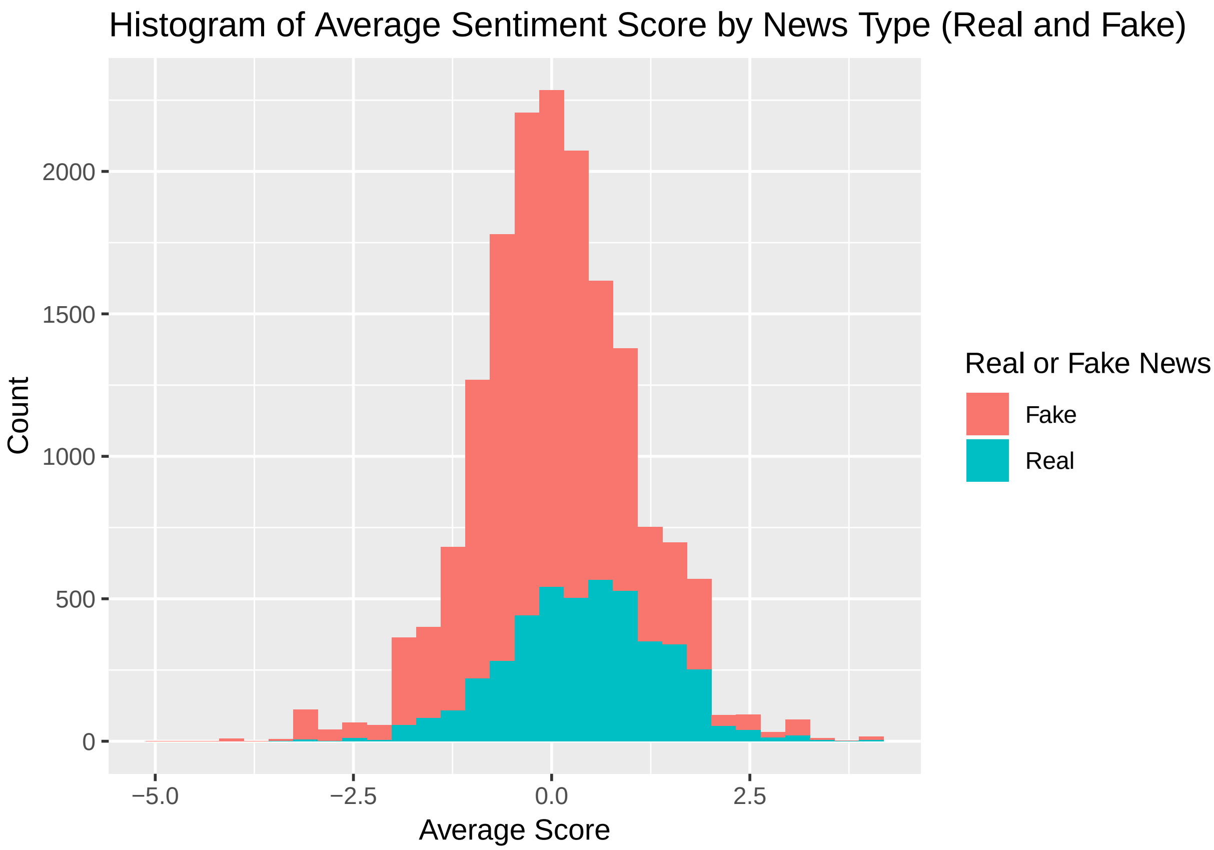



```
# Based on this tree, we can see that none of the predictors (average score, number of words, number of
```

```
# Why is this true? Below, we will do some exploration using visualizations to display the poor relatio

# Histogram of Average Sentiment Score by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = avgscore, fill = txt_type)) +
  geom_histogram() +
  xlab("Average Score") +
  ylab("Count") +
```

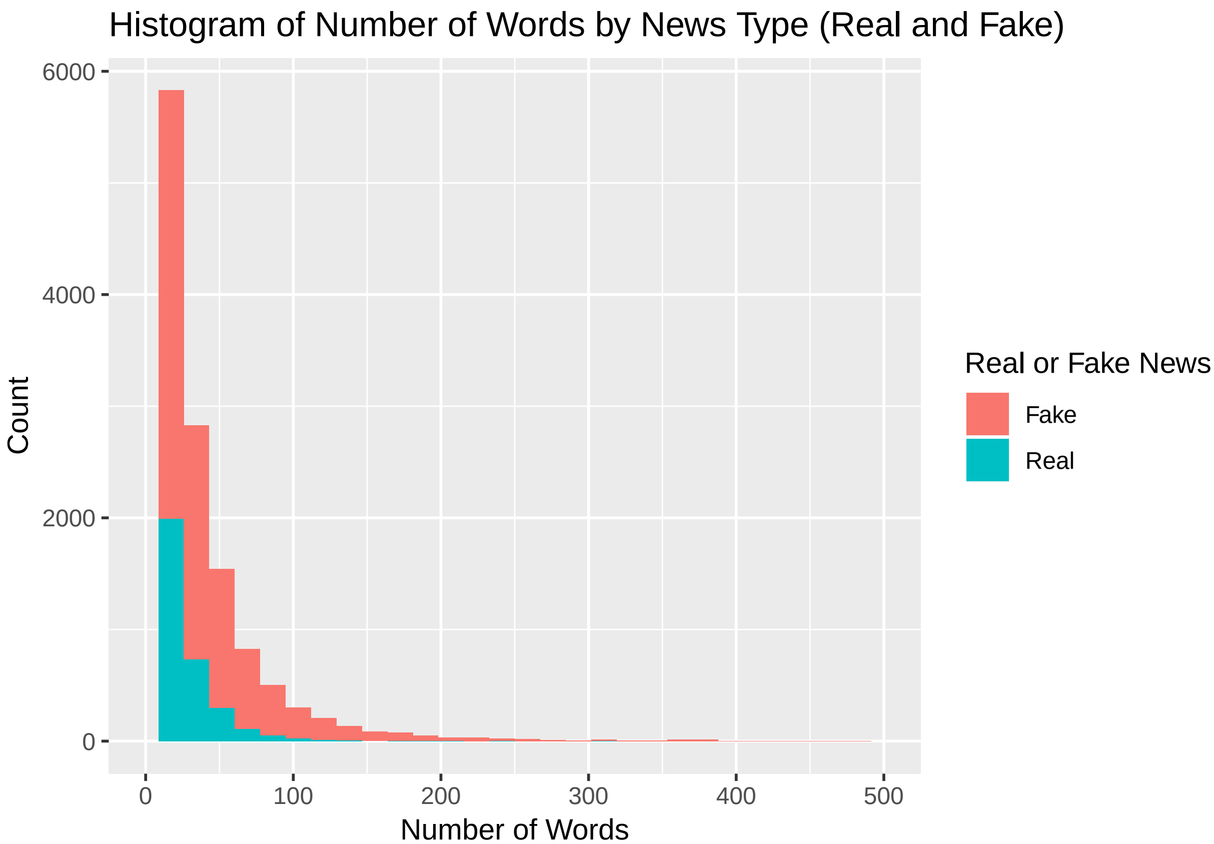
```r
  ggtitle("Histogram of Average Sentiment Score by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



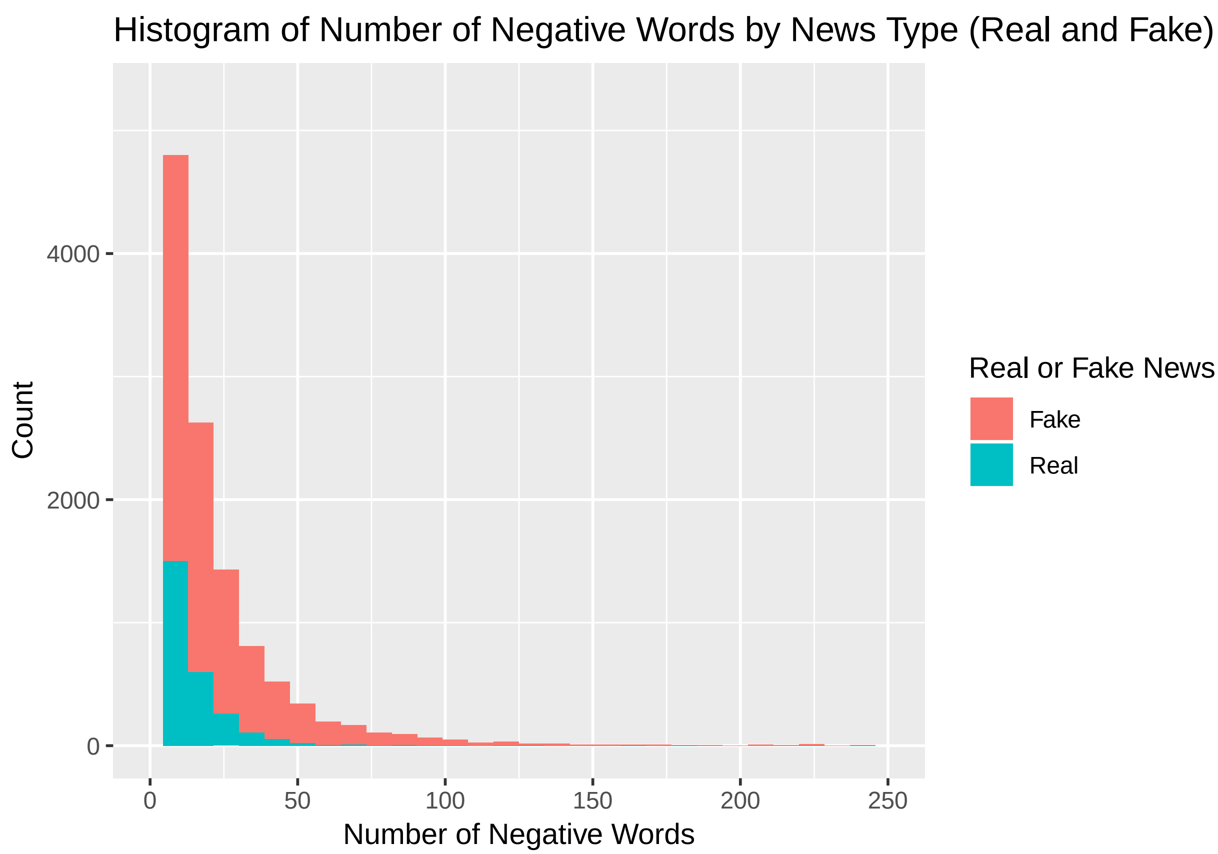

```r
# Histogram of Number of Words by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = n_words, fill = txt_type)) +
  geom_histogram() +
  xlim(0, 500) +
  xlab("Number of Words") +
  ylab("Count") +
  ggtitle("Histogram of Number of Words by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 25 rows containing non-finite values (stat_bin).

## Warning: Removed 4 rows containing missing values (geom_bar).
```



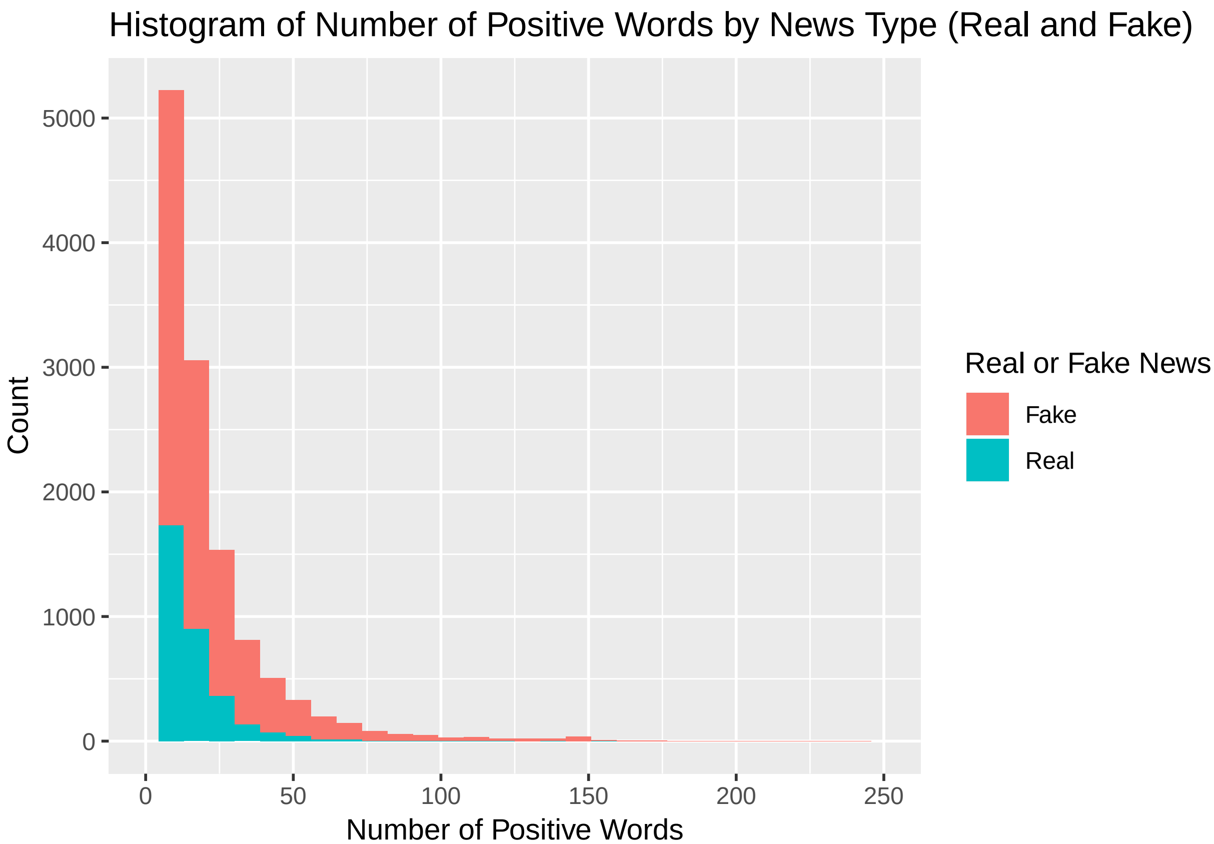

```r
# Histogram of Number of Negative Words by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = n_negative, fill = txt_type)) +
  geom_histogram() +
  xlim(0, 250) +
  xlab("Number of Negative Words") +
  ylab("Count") +
  ggtitle("Histogram of Number of Negative Words by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 27 rows containing non-finite values (stat_bin).

## Warning: Removed 4 rows containing missing values (geom_bar).
```




```
# Histogram of Number of Positive Words by News Type (Real and Fake
ggplot(tidy_combined_final, aes(x = n_positive, fill = txt_type)) +
  geom_histogram() +
  xlim(0, 250) +
  xlab("Number of Positive Words") +
  ylab("Count") +
  ggtitle("Histogram of Number of Positive Words by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 27 rows containing non-finite values (stat_bin).

## Warning: Removed 4 rows containing missing values (geom_bar).
```

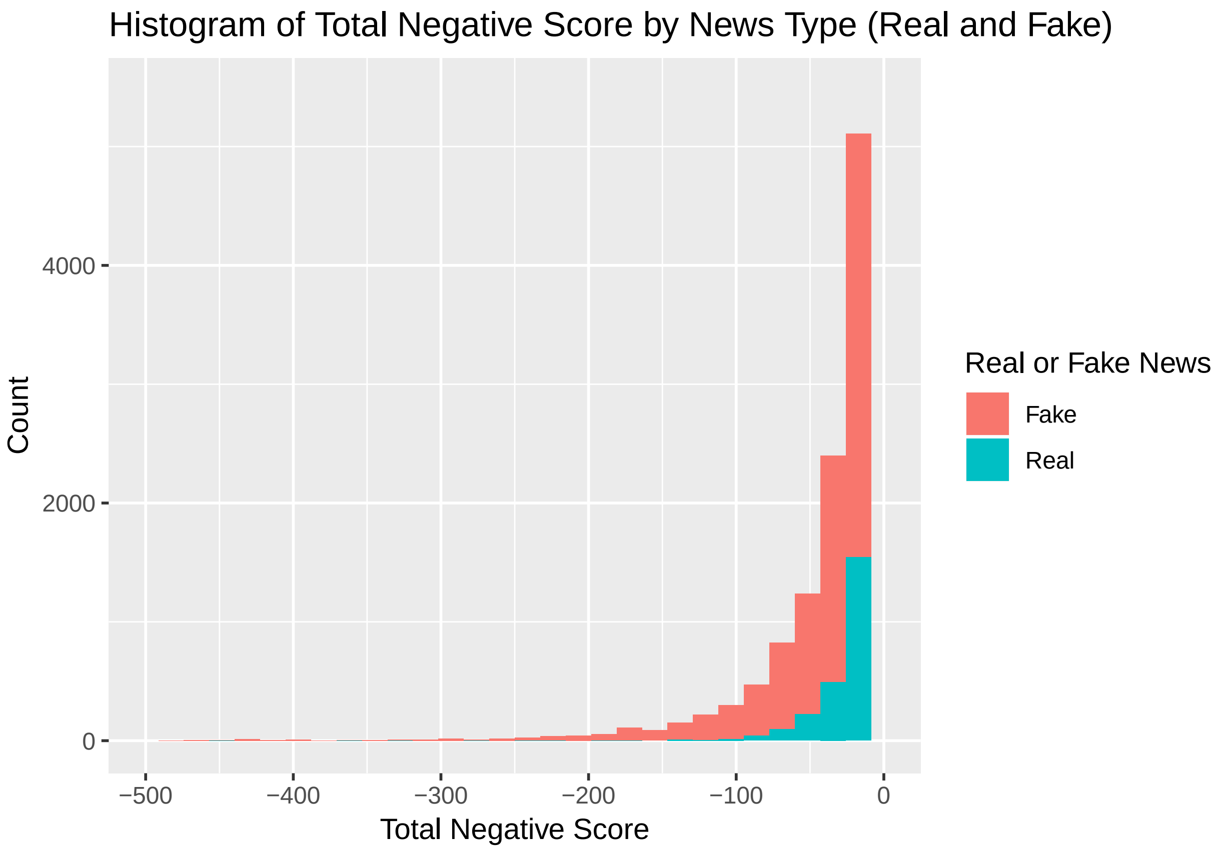



```r
# Histogram of Total Negative Score by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = negative_score, fill = txt_type)) +
  geom_histogram() +
  xlim(-500, 0) +
  xlab("Total Negative Score") +
  ylab("Count") +
  ggtitle("Histogram of Total Negative Score by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

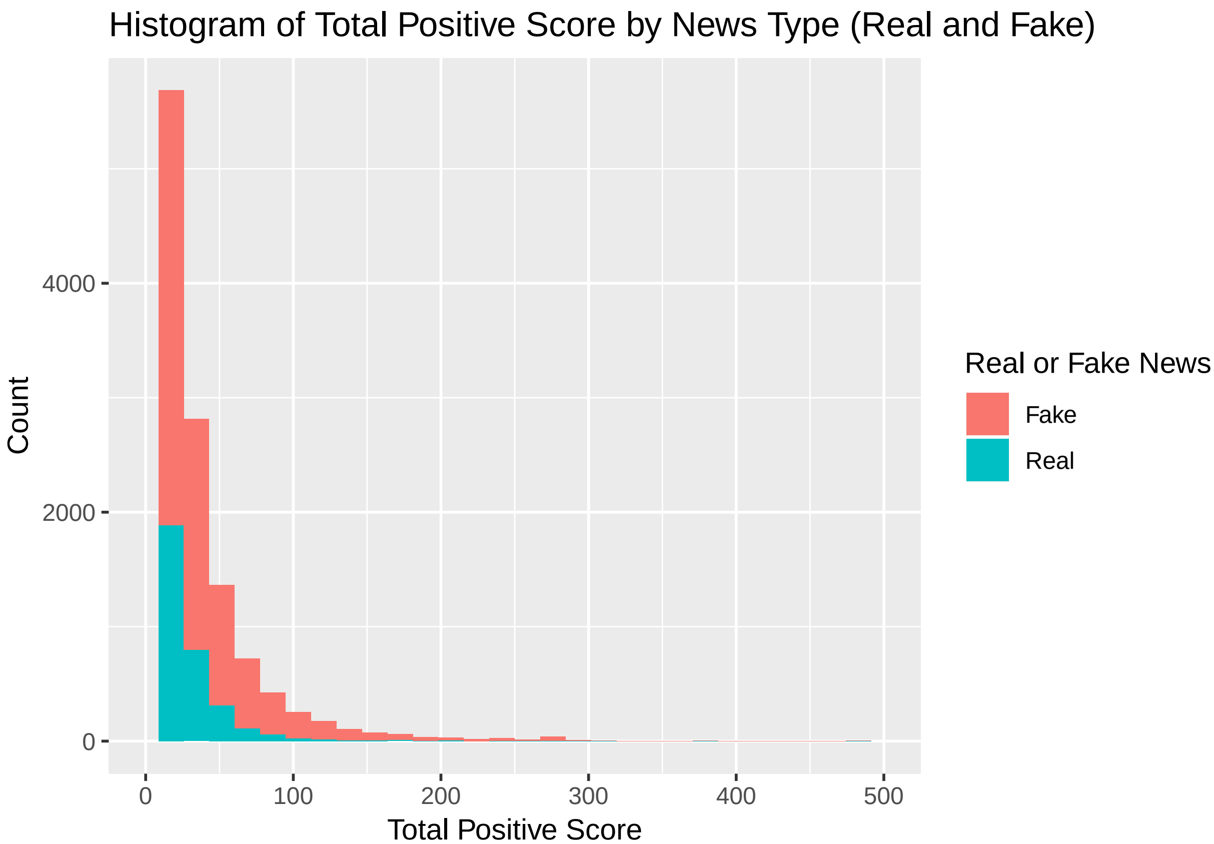
```
## Warning: Removed 25 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 4 rows containing missing values (geom_bar).
```




```r
# Histogram of Total Positive Score by News Type (Real and Fake)
ggplot(tidy_combined_final, aes(x = positive_score, fill = txt_type)) +
  geom_histogram() +
  xlim(0, 500) +
  xlab("Total Positive Score") +
  ylab("Count") +
  ggtitle("Histogram of Total Positive Score by News Type (Real and Fake)") +
  scale_fill_discrete(name = "Real or Fake News", labels = c("Fake", "Real"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 21 rows containing non-finite values (stat_bin).

## Warning: Removed 4 rows containing missing values (geom_bar).
```


Histogram of Total Positive Score by News Type (Real and Fake)
Count
4000
2000
0
0
100
200
300
400
500
Total Positive Score
Real or Fake News
Fake
Real