Novels by Dan Brown Textmining - Shiny

This R Markdown document is made interactive using Shiny.

Make Packages Available

Here I load all the packages necessary for this project. The first thing I need to do is load some packages that I'm going to be using. I use pacman simply to manage packages, and tm is a text mining and that will give us most of our functionality. SnowballC adds some additional text analysis, and dplyr is for manipulating data and for arranging the code using pipes, where the output of one command feeds directly into the input of another one.

```
library(shiny)
library(wordcloud)
library(devtools)
library(tidyverse)
library(stringr)
library(tidytext)
library(dplyr)
library(reshape2)
library(igraph)
library(ggraph)
library(memoise)
if (packageVersion("devtools") < 1.6) {
  install.packages("devtools")
}
pacman::p_load(pacman, tm, SnowballC, dplyr)
```

Import Three Books

The books this project is going to do textmining and sentiment analysis on are three novels by Dan Brown - “Angels & Demons”, “The Da Vinci Code”, and “The Lost Symbol”. I'll start by importing book data, which is the full content of the three books. I have everything in the same directory, so there's no need to give a specific file path. I've already removed the metadata at the beginning and the end of the documents, so all that's left is the novels themselves.

```
# "Angels & Demons" by Dan Brown, published 2000
bookAAD <- readLines('ANGELS AND DEMONS.txt')

# "The Da Vinci Code" by Dan Brown, published 2003
bookDVC <- readLines('The Da Vinci Code.txt')

# "The Lost Symbol" by Dan Brown, pubished 2009
bookTLS <- readLines('The Lost Symbol.txt')
```

I'll begin by giving the data of every single book respectively, and then I'll compare their features in a set of 2 books and 3 later. First, I'm going to create a Corpus, which is a body of text for each book. I'll begin by creating what I call a preliminary corpus, because I'm going to do some later clean-up on it. These commands come from tm, for text mining. I'm going to remove the punctuation, any numbers, change everything to lowercase, and remove stopwords. Stopwords are words such as “the”, “I”, “but”, which are usually meaningless when doing text mining.

I'm also going to stem the documents, and what that does is it takes a word like, "Stop" and it takes the variations of it, "Stops, stopped, stopping," and it cuts off those end parts and leaves us with just the beginning, "Stop."

```
# CORPUS FOR ANGELS & DEMONS

# Preliminary corpus
corpusAAD <- Corpus(VectorSource(bookAAD)) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace) %>%
  tm_map(stemDocument)

# Create term-document matrices & remove sparse terms
tdmAAD <- DocumentTermMatrix(corpusAAD) %>%
  removeSparseTerms(1 - (5/length(corpusAAD)))
```

Word Frequencies

Now I'm going to get absolute frequencies for each word, and then relative frequencies

```
# Calculate and sort by word frequencies
word.freqAAD <- sort(colSums(as.matrix(tdmAAD)),
                     decreasing = T)

# Create frequency table
tableAAD <- data.frame(word = names(word.freqAAD),
                       absolute.frequency = word.freqAAD,
                       relative.frequency =
                         word.freqAAD/length(word.freqAAD))

# Remove the words from the row names
rownames(tableAAD) <- NULL

# Show the 10 most common words
head(tableAAD, 10)
```

```
##      word absolute.frequency relative.frequency
## 1   angel                211               0.84
## 2     god                126               0.50
## 3  heaven                 71               0.28
## 4   earth                 62               0.25
## 5    bodi                 55               0.22
## 6    will                 51               0.20
## 7     one                 42               0.17
## 8    bibl                 42               0.17
## 9    know                 40               0.16
## 10   time                 39               0.15
```

As we can see in the table, Dan Brown uses "angel" 211 times and the relative frequency of "angel" is about 0.84.

I'm now going to create a csv file that has the most common words together with their absolute and relative frequencies. The file name will be AAD_1000 in which AAD stands for Angels And Demons. The file will be saved to the same directory where I have my other documents.

```
# Export the 1000 most common words in CSV files
write.csv(tableAAD[1:1000, ], "AAD_1000.csv")
```

I'll repeat the same steps described above on the other two books in the following codes.

```
# CORPUS FOR THE DA VINCI CODE

corpusDVC <- Corpus(VectorSource(bookDVC)) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace) %>%
  tm_map(stemDocument)
tdmDVC <- DocumentTermMatrix(corpusDVC) %>%
  removeSparseTerms(1 - (5/length(corpusDVC)))
word.freqDVC <- sort(colSums(as.matrix(tdmDVC)),
                     decreasing = T)
tableDVC <- data.frame(word = names(word.freqDVC),
                       absolute.frequency = word.freqDVC,
                       relative.frequency =
                         word.freqDVC/length(word.freqDVC))
rownames(tableDVC) <- NULL
head(tableDVC, 10)
```

```
##       word absolute.frequency relative.frequency
## 1  langdon               1579               0.61
## 2    sophi               1127               0.43
## 3     teab                601               0.23
## 4     said                536               0.21
## 5      now                430               0.17
## 6     look                418               0.16
## 7     fach                398               0.15
## 8      one                325               0.13
## 9     back                295               0.11
## 10   grail                290               0.11
```

```
write.csv(tableDVC[1:1000, ], "DVC_1000.csv")
```

```
# CORPUS FOR THE LOST SYMBOL

corpusTLS <- Corpus(VectorSource(bookTLS)) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace) %>%
  tm_map(stemDocument)
tdmTLS <- DocumentTermMatrix(corpusTLS) %>%
  removeSparseTerms(1 - (5/length(corpusTLS)))
word.freqTLS <- sort(colSums(as.matrix(tdmTLS)),
                     decreasing = T)
tableTLS <- data.frame(word = names(word.freqTLS),
                       absolute.frequency = word.freqTLS,
                       relative.frequency =
                         word.freqTLS/length(word.freqTLS))
rownames(tableTLS) <- NULL
head(tableTLS, 10)
```

```
##        word absolute.frequency relative.frequency
## 1   langdon               1365               0.50
## 2  katherin                762               0.28
## 3      said                696               0.25
## 4       now                555               0.20
## 5     peter                553               0.20
## 6       man                441               0.16
## 7      look                437               0.16
## 8   pyramid                415               0.15
## 9   solomon                394               0.14
## 10      one                389               0.14
```

```
write.csv(tableTLS[1:1000, ], "TLS_1000.csv")
```

Here’s the part where I’ll compare their features in a set of 2 books to find out the most distinctive words. I’m going to create one called dProp, which is for a difference in proportions. Now, in this case, I’m simply taking the difference, a subtraction.

“Angels & Demons” vs “The Da Vinci Code”

```
# MOST DISTINCTIVE WORDS ################################

# Set number of digits for output
options(digits = 2)

# Compare relative frequencies (via subtraction)
# ("Angels & Demons" vs "The Da Vinci Code")
AADvsDVC <- tableAAD %>%
  merge(tableDVC, by = "word") %>%
  mutate(dProp =
           relative.frequency.x -
           relative.frequency.y,
         dAbs = abs(dProp)) %>%
  arrange(desc(dAbs)) %>%
  rename(AAD.freq = absolute.frequency.x,
         AAD.prop = relative.frequency.x,
         DVC.freq = absolute.frequency.y,
         DVC.freq = relative.frequency.y)

# Show the 10 most distinctive terms
head(AADvsDVC, 10)
```

```
##      word AAD.freq AAD.prop DVC.freq DVC.freq dProp dAbs
## 1   angel      211    0.837        6   0.0023  0.83 0.83
## 2     god      126    0.500      105   0.0404  0.46 0.46
## 3  heaven       71    0.282       22   0.0085  0.27 0.27
## 4   earth       62    0.246       36   0.0139  0.23 0.23
## 5    bodi       55    0.218       76   0.0293  0.19 0.19
## 6    said       11    0.044      536   0.2065 -0.16 0.16
## 7    bibl       42    0.167       26   0.0100  0.16 0.16
## 8   creat       33    0.131       25   0.0096  0.12 0.12
## 9    will       51    0.202      228   0.0878  0.11 0.11
## 10   lord       28    0.111       18   0.0069  0.10 0.10
```

```
# Save full table to CSV
write.csv(AADvsDVC, "AAD vs DVC.csv")
```

As we can see in the table above, “angel” appears 211 times in Angels & Demons, while it only appears 6 times in The Da Vinci Code, which makes sense given the story. That’s why it has a positive dProp, or difference in proportions. The full table is going to be saved as a csv file in the same directory.

I’ll continue the same steps for the other two sets.

“Angels & Demons” vs “The Lost Symbol”

```
# ("Angels & Demons" vs "The Lost Symbol")
AADvsTLS <- tableAAD %>%
  merge(tableTLS, by = "word") %>%
  mutate(dProp =
           relative.frequency.x -
           relative.frequency.y,
         dAbs = abs(dProp)) %>%
  arrange(desc(dAbs)) %>%
  rename(AAD.freq = absolute.frequency.x,
         AAD.prop = relative.frequency.x,
         TLS.freq = absolute.frequency.y,
         TLS.freq = relative.frequency.y)
head(AADvsTLS, 10)
```

```
##      word AAD.freq AAD.prop TLS.freq TLS.freq dProp dAbs
## 1   angel      211    0.837       13   0.0048  0.83 0.83
## 2     god      126    0.500      156   0.0571  0.44 0.44
## 3  heaven       71    0.282       48   0.0176  0.26 0.26
## 4   earth       62    0.246       63   0.0231  0.22 0.22
## 5    said       11    0.044      696   0.2549 -0.21 0.21
## 6   peter        5    0.020      553   0.2026 -0.18 0.18
## 7    bodi       55    0.218      142   0.0520  0.17 0.17
## 8    bibl       42    0.167       46   0.0168  0.15 0.15
## 9    will       51    0.202      227   0.0832  0.12 0.12
## 10  jesus       31    0.123       24   0.0088  0.11 0.11
```

```
write.csv(AADvsTLS, "AAD vs TLS.csv")
```

“The Da Vinci Code” vs “The Lost Symbol”

```
# ("The Da Vinci Code" vs "The Lost Symbol")
DVCvsTLS <- tableDVC %>%
  merge(tableTLS, by = "word") %>%
  mutate(dProp =
           relative.frequency.x -
           relative.frequency.y,
         dAbs = abs(dProp)) %>%
  arrange(desc(dAbs)) %>%
  rename(DVC.freq = absolute.frequency.x,
         DVC.prop = relative.frequency.x,
         TLS.freq = absolute.frequency.y,
         TLS.freq = relative.frequency.y)
head(DVCvsTLS, 10)
```

```
##       word DVC.freq DVC.prop TLS.freq TLS.freq  dProp  dAbs
## 1    peter       13   0.0050      553   0.2026 -0.198 0.198
## 2  solomon       17   0.0065      394   0.1443 -0.138 0.138
## 3  pyramid       40   0.0154      415   0.1520 -0.137 0.137
## 4    mason       16   0.0062      320   0.1172 -0.111 0.111
## 5  langdon     1579   0.6082     1365   0.5000  0.108 0.108
## 6   church      236   0.0909       11   0.0040  0.087 0.087
## 7  ancient       68   0.0262      260   0.0952 -0.069 0.069
## 8      man      254   0.0978      441   0.1615 -0.064 0.064
## 9  brother       14   0.0054      166   0.0608 -0.055 0.055
## 10 teacher      146   0.0562        9   0.0033  0.053 0.053
```

```
write.csv(DVCvsTLS, "DVC vs TLS.csv")
```

Here’s the part where I’ll compare their features in a set of 3 books

```
# Three BOOKS DATA
titles <- c("Angels & Demons", "The Da Vinci Code", "The Lost Symbol")

books <- list(bookAAD, bookDVC, bookTLS)

##Each book is an array in which each value in the array is a chapter
series <- tibble()
for(i in seq_along(titles)) {

  temp <- tibble(chapter = seq_along(books[[i]]),
                 text = books[[i]]) %>%
    unnest_tokens(word, text) %>%
    ##Here we tokenize each chapter into words
    mutate(book = titles[i]) %>%
    select(book, everything())

  series <- rbind(series, temp)
}
# set factor to keep books in order of publication
series$book <- factor(series$book, levels = rev(titles))
series
```

```
## # A tibble: 311,918 x 3
##    book              chapter word
##    <fct>               <int> <chr>
##  1 Angels & Demons         2 angels
##  2 Angels & Demons         3 and
##  3 Angels & Demons         3 demons
##  4 Angels & Demons         5 their
##  5 Angels & Demons         5 nature
##  6 Angels & Demons         5 origin
##  7 Angels & Demons         5 mtntsfry
##  8 Angels & Demons         6 ond
##  9 Angels & Demons         6 classification
## 10 Angels & Demons        10 four
## # ... with 311,908 more rows
```

We can get counts for each word using the count function.

```
series %>% count(word, sort = TRUE)
```

```
## # A tibble: 17,039 x 2
##    word      n
##    <chr> <int>
##  1 the   22244
##  2 of     7648
##  3 a      7134
##  4 to     7114
##  5 and    6528
##  6 in     4476
##  7 he     4066
##  8 was    4059
##  9 his    3494
## 10 had    3072
## # ... with 17,029 more rows
```

Many of the words in the top 10 most common words are stopwords. I’m going to remove the stopwords here and make a word cloud.

```
# Remove stopwords and make a word cloud
series$book <- factor(series$book, levels = rev(titles))
series %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```

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

Sentiment derived from the NRC

```
(hp_nrc <- series %>%
   inner_join(get_sentiments("nrc")) %>%
   group_by(book, chapter, sentiment))
```

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

```
## # A tibble: 69,460 x 4
## # Groups:   book, chapter, sentiment [48,451]
##    book             chapter word     sentiment
##    <fct>              <int> <chr>    <chr>
##  1 Angels & Demons       10 radio    positive
##  2 Angels & Demons       32 throne   positive
##  3 Angels & Demons       32 throne   trust
##  4 Angels & Demons       33 elders   positive
##  5 Angels & Demons       33 elders   trust
##  6 Angels & Demons       38 subject  negative
##  7 Angels & Demons       41 spirit   positive
##  8 Angels & Demons       45 heavenly anticipation
##  9 Angels & Demons       45 heavenly joy
## 10 Angels & Demons       45 heavenly positive
## # ... with 69,450 more rows
```

Sentiment analysis using the AFINN dictionary

Here I want to visualize the positive/negative sentiment for each book over time using the AFINN dictionary.

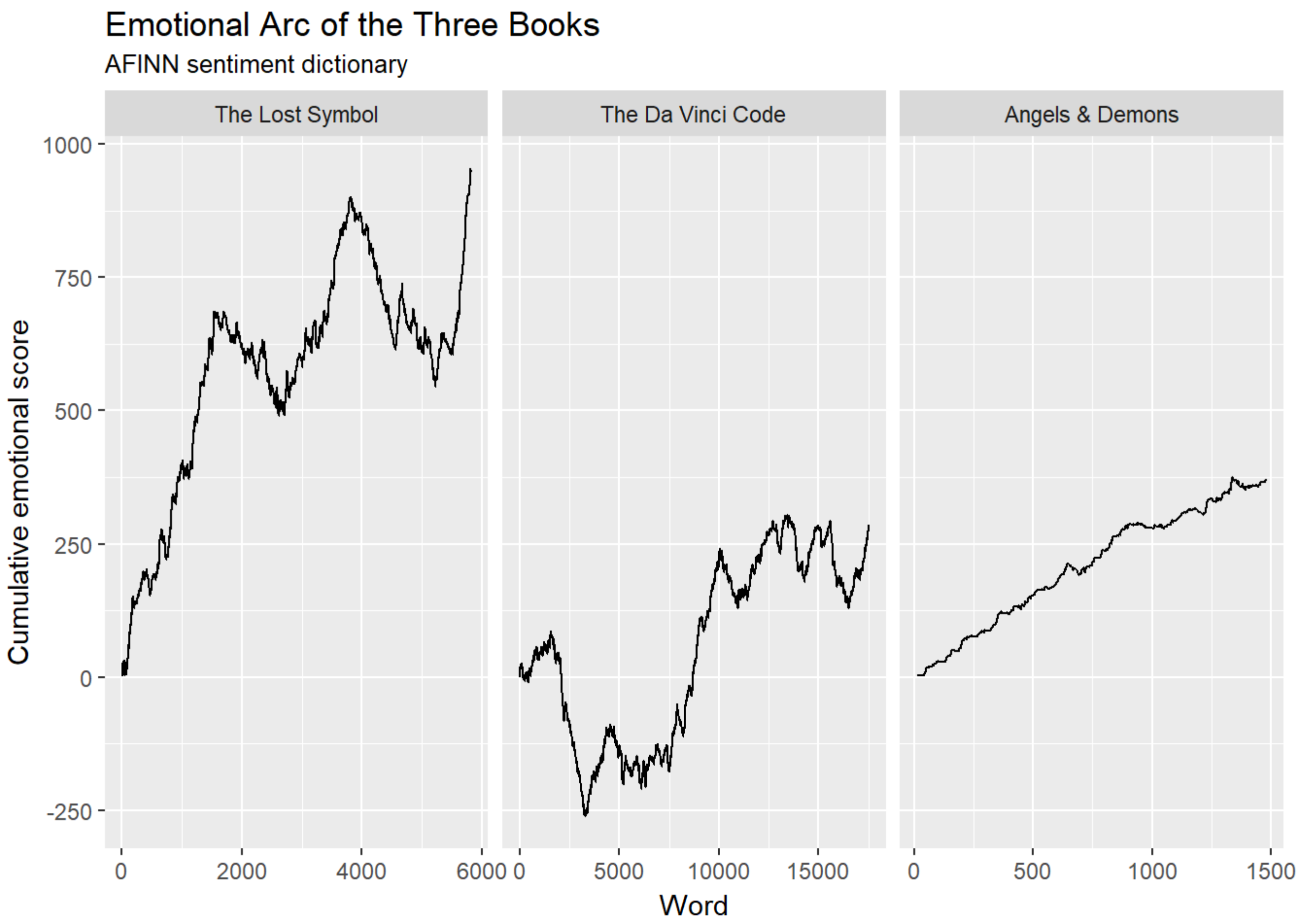
```
series %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(book, chapter) %>%
  summarize(score = sum(score)) %>%
  ggplot(aes(chapter, score, fill = book)) +
  geom_col() +
  facet_wrap(~ book, scales = "free_x") +
  labs(title = "Emotional Arc of the Three Books",
       subtitle = "AFINN sentiment dictionary",
       x = "Word",
       y = "Emotional score") +
  theme(legend.position = "none")
```

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

Here I'm going to show the cumulative score

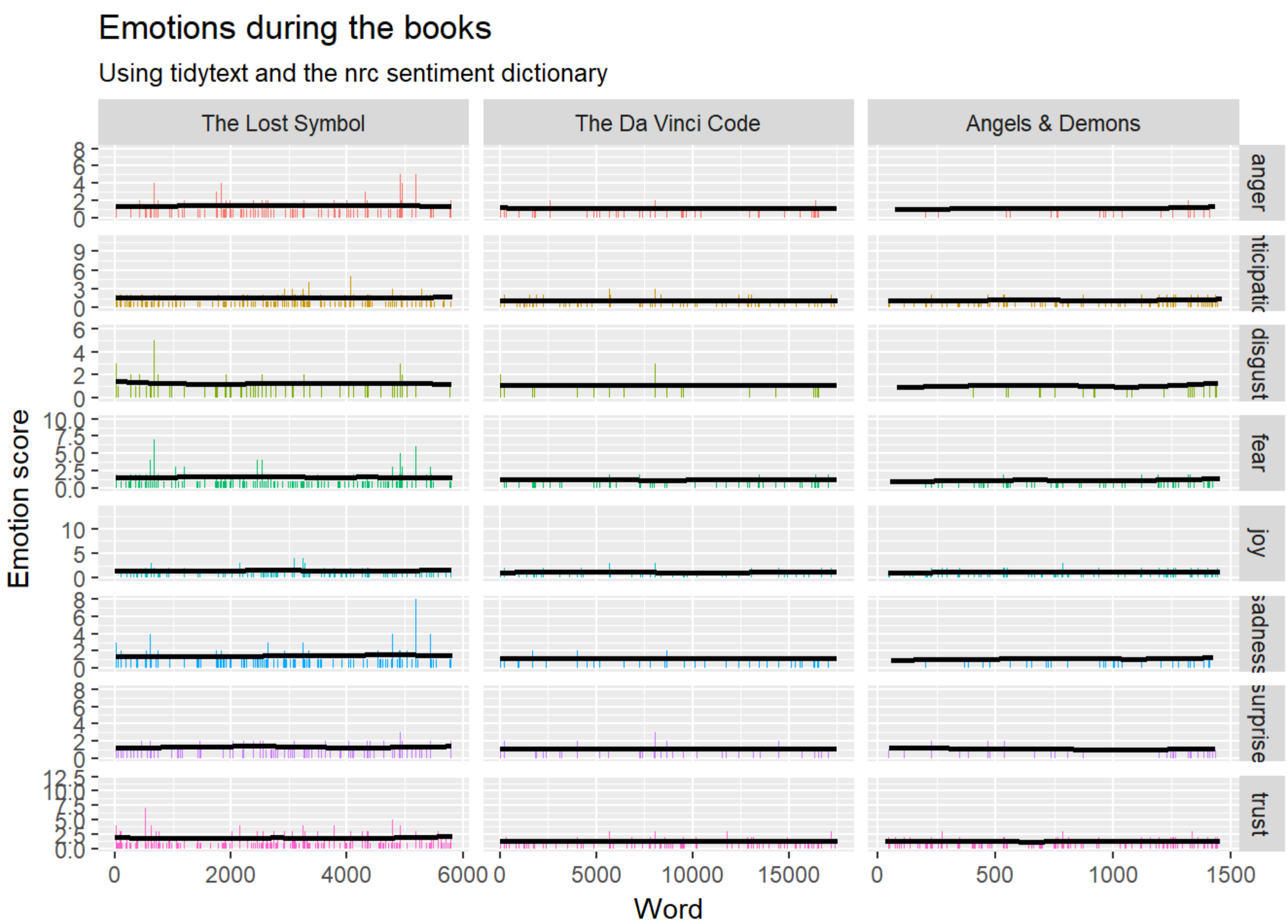
```
# Cumulative score
series %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(book) %>%
  mutate(cumscore = cumsum(score)) %>%
  ggplot(aes(chapter, cumscore, fill = book)) +
  geom_step() +
  facet_wrap(~ book, scales = "free_x") +
  labs(title = "Emotional Arc of the Three Books",
       subtitle = "AFINN sentiment dictionary",
       x = "Word",
       y = "Cumulative emotional score")
```

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



After this, I want to visualize the sentimental content of each chapter in each book using the NRC dictionary.

```
hp_nrc %>%
  count(sentiment, book, chapter) %>%
  filter(!(sentiment %in% c("positive", "negative"))) %>%
  # create area plot
  ggplot(aes(x = chapter, y = n)) +
  geom_col(aes(fill = sentiment)) +
  # add black smoothing line without standard error
  geom_smooth(aes(fill = sentiment), method = "loess", se = F, col = 'black') +
  theme(legend.position = 'none') +
  labs(x = "Word", y = "Emotion score", # add labels
       title = "Emotions during the books",
       subtitle = "Using tidytext and the nrc sentiment dictionary") +
  # seperate plots per sentiment and book and free up x-axes
  facet_grid(sentiment ~ book, scale = "free")
```




```
### This chunk takes longer to execute
```

Another sentiment analysis

```
series %>%
  right_join(get_sentiments("nrc")) %>%
  filter(!is.na(sentiment)) %>%
  count(sentiment, sort = TRUE)
```

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

```
## # A tibble: 10 x 2
##    sentiment        n
##    <chr>        <int>
##  1 positive     16491
##  2 negative     12039
##  3 trust        10829
##  4 anticipation  7330
##  5 fear          7103
##  6 joy           5749
##  7 sadness       5451
##  8 anger         4594
##  9 surprise      3763
## 10 disgust       3254
```

The ‘bing’ lexicon only classifies words as positive or negative.

```
series %>%
  right_join(get_sentiments("bing")) %>%
  filter(!is.na(sentiment)) %>%
  count(sentiment, sort = TRUE)
```

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

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative  12204
## 2 positive   9978
```

Next I’m going to Use the the ‘bing’ lexicon for sentiment analysis and make a comparison cloud.

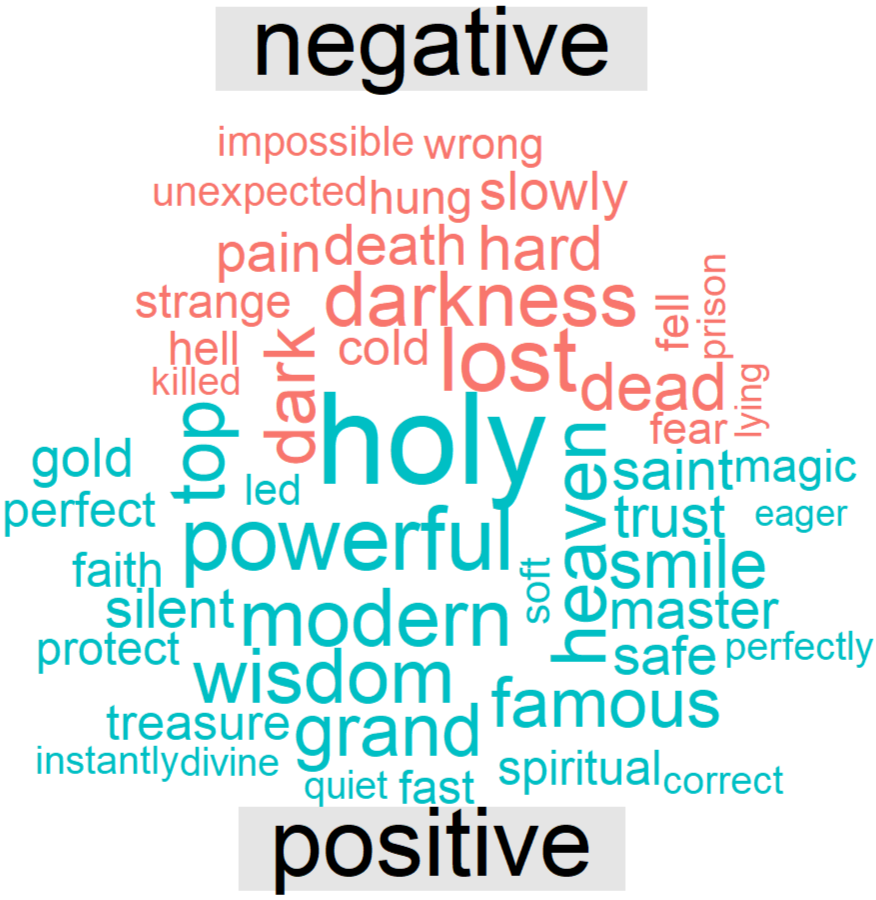
```
series %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("#F8766D", "#00BFC4"),
                   max.words = 50)
```

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

The comparison above still contains stopwords, and now I'm going to remove them and make a new cloud. I also use colors o separate words that are positive or negative. We can see here that character names don't appear in the following word cloud, because 'bing' doesn't classify names as positive or negative.

```
series %>%
  anti_join(stop_words) %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("#F8766D", "#00BFC4"),
                   max.words = 50)
```

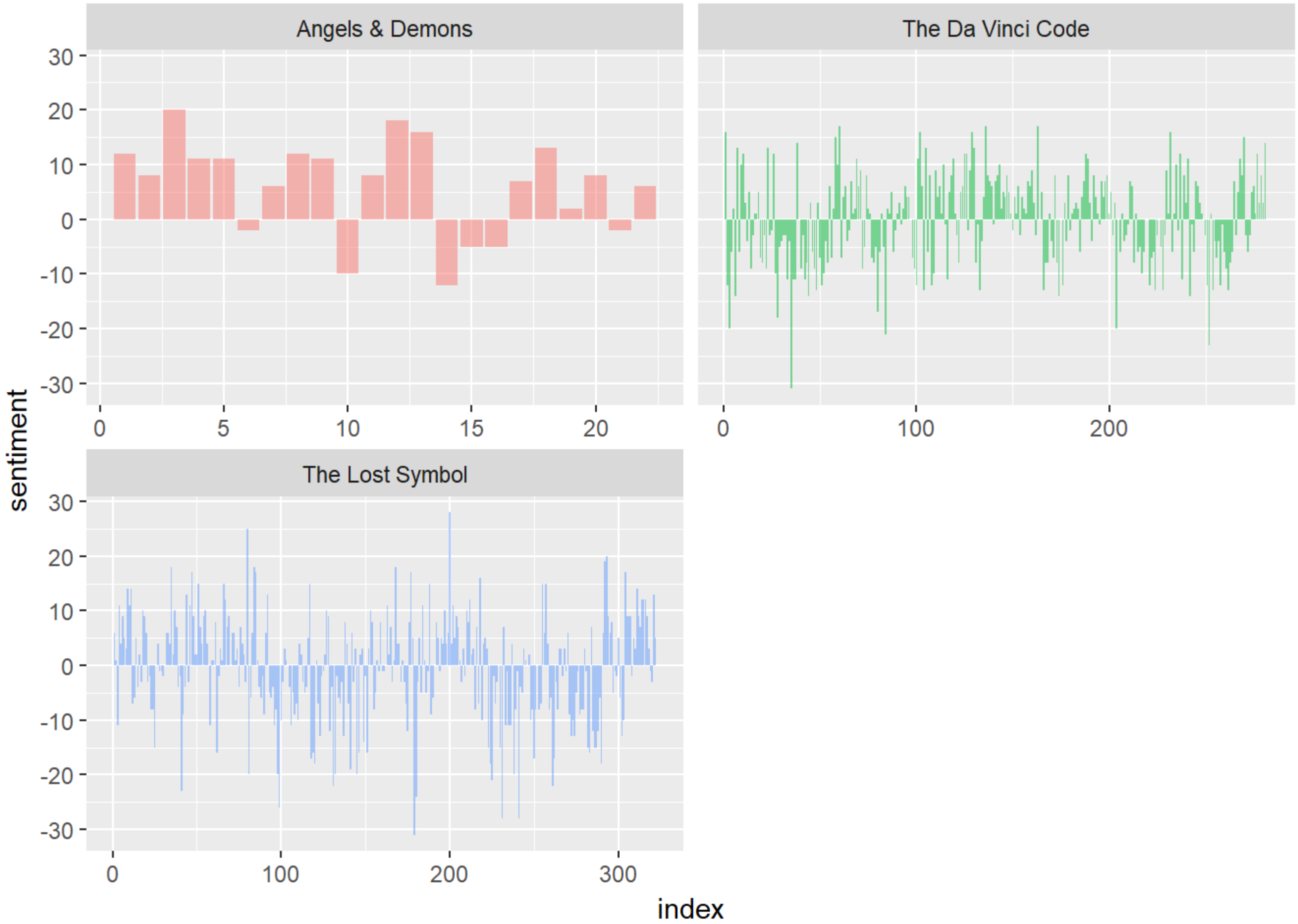
```
## Joining, by = "word"
## Joining, by = "word"
```



Calculating Sentiment Score.

```
series %>%
  group_by(book) %>%
  mutate(word_count = 1:n(),
         index = word_count %/% 500 + 1) %>%
  inner_join(get_sentiments("bing")) %>%
  count(book, index = index , sentiment) %>%
  ungroup() %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative,
         book = factor(book, levels = titles)) %>%
  ggplot(aes(index, sentiment, fill = book)) +
  geom_bar(alpha = 0.5, stat = "identity", show.legend = FALSE) +
  facet_wrap(~ book, ncol = 2, scales = "free_x")
```

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



Now, when we are doing sentiment analysis, using single words as tokens can be misleading sometimes. For example, if we say “This movie is not good,” we will know that this is a negative statement, while using single words as tokens can lead us to believe that this is a positive statemenet. To solve the problem, I’m going to consider pairs of words (bigrams). In the sentence “I love yellow cars,” the bigrams we can extract would be (I, love), (love, yellow), (yellow cars).

```
# Pairs of words (Bigrams)
series <- tibble()
for(i in seq_along(titles)) {

  temp <- tibble(chapter = seq_along(books[[i]]),
                 text = books[[i]]) %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
    ##Here we tokenize each chapter into bigrams
    mutate(book = titles[i]) %>%
    select(book, everything())

  series <- rbind(series, temp)
}
# set factor to keep books in order of publication
series$book <- factor(series$book, levels = rev(titles))
series
```

```
## # A tibble: 301,131 x 3
##    book            chapter bigram
##    <fct>             <int> <chr>
##  1 Angels & Demons       1 <NA>
##  2 Angels & Demons       2 <NA>
##  3 Angels & Demons       3 and demons
##  4 Angels & Demons       4 <NA>
##  5 Angels & Demons       5 their nature
##  6 Angels & Demons       5 nature origin
##  7 Angels & Demons       5 origin mtntsfry
##  8 Angels & Demons       6 ond classification
##  9 Angels & Demons       7 <NA>
## 10 Angels & Demons       8 <NA>
## # ... with 301,121 more rows
```

I'm going to use the count function to find the most common bigrams in the books.

```
series %>%
  count(bigram, sort = TRUE)
```

```
## # A tibble: 133,203 x 2
##    bigram       n
##    <chr>    <int>
##  1 <NA>      7261
##  2 of the    2098
##  3 in the    1309
##  4 to the     988
##  5 on the     901
##  6 at the     691
##  7 had been   459
##  8 he had     440
##  9 into the   423
## 10 and the    416
## # ... with 133,193 more rows
```

As we can see there are still stopwords containted in the table, let's remove them.

```
# bigrams without stopwords
bigrams_separated <- series %>%
  separate(bigram, c("word1", "word2"), sep = " ")
bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
# new bigram counts:
bigrams_united <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ")
bigrams_united %>%
  count(bigram, sort = TRUE)
```

```
## # A tibble: 35,369 x 2
##    bigram                n
##    <chr>             <int>
##  1 NA NA              7261
##  2 robert langdon      140
##  3 peter solomon       130
##  4 holy grail          119
##  5 da vinci             91
##  6 masonic pyramid      83
##  7 opus dei             75
##  8 ancient mysteries    71
##  9 katherine solomon    71
## 10 cell phone           70
## # ... with 35,359 more rows
```

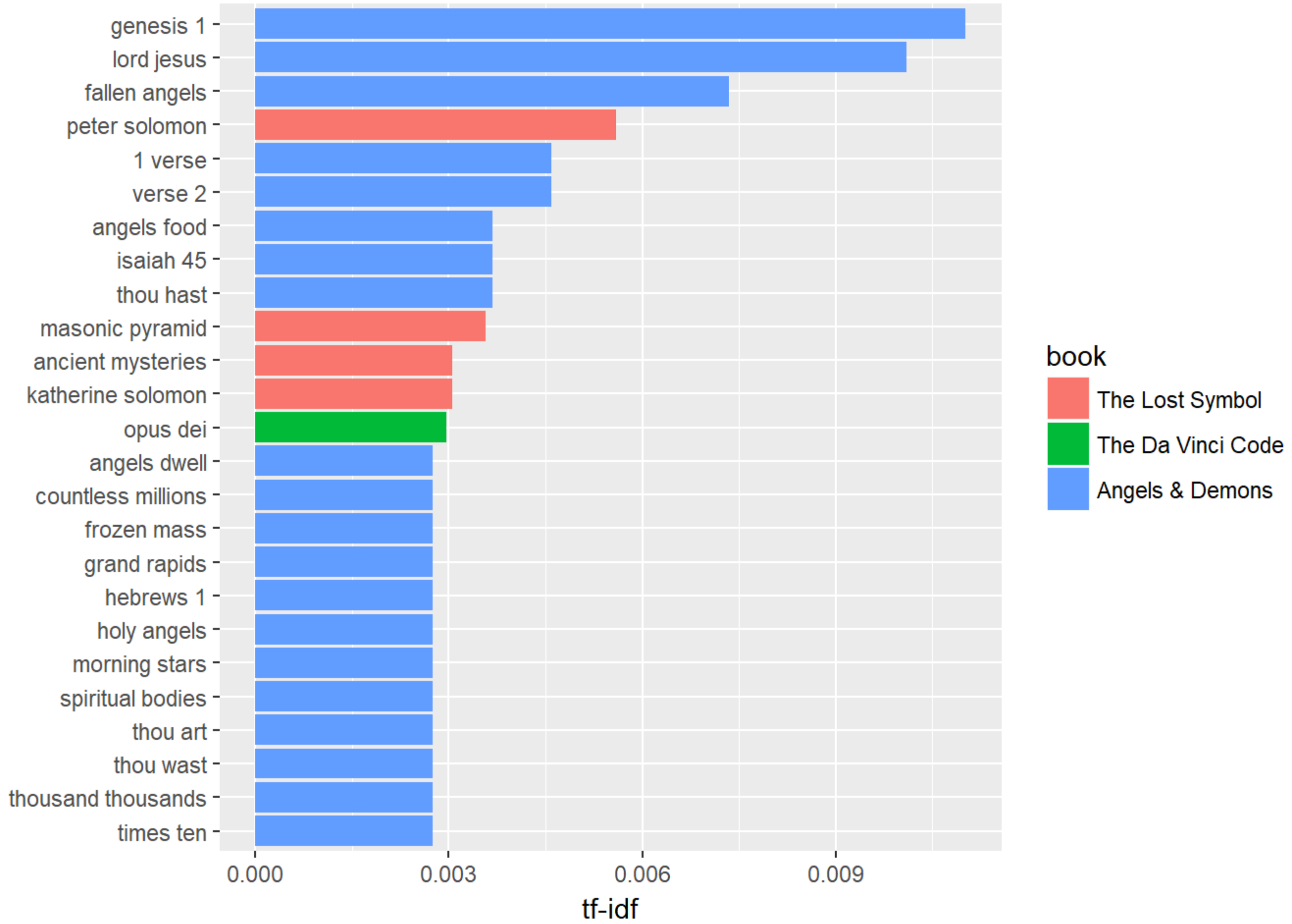
Now, I'll use bigrams to practice tf-idf (term frequency -inverse document frequency). Tf-idf is an analysis that seeks to identify how common a word is in a particular text, given how often it occurs in a group of texts.

```
# Use bigrams to practice tf-idf
# (term frequency -inverse document frequency)
bigram_tf_idf <- bigrams_united %>%
  count(book, bigram) %>%
  bind_tf_idf(bigram, book, n) %>%
  arrange(desc(tf_idf))
bigram_tf_idf
```

```
## # A tibble: 36,511 x 6
##    book            bigram              n      tf   idf  tf_idf
##    <fct>           <chr>           <int>   <dbl> <dbl>   <dbl>
##  1 Angels & Demons genesis 1          12 0.0100   1.10 0.0110
##  2 Angels & Demons lord jesus         11 0.00920  1.10 0.0101
##  3 Angels & Demons fallen angels       8 0.00669  1.10 0.00735
##  4 The Lost Symbol peter solomon     130 0.00509  1.10 0.00559
##  5 Angels & Demons 1 verse             5 0.00418  1.10 0.00459
##  6 Angels & Demons verse 2             5 0.00418  1.10 0.00459
##  7 Angels & Demons angels food         4 0.00334  1.10 0.00367
##  8 Angels & Demons isaiah 45           4 0.00334  1.10 0.00367
##  9 Angels & Demons thou hast           4 0.00334  1.10 0.00367
## 10 The Lost Symbol masonic pyramid    83 0.00325  1.10 0.00357
## # ... with 36,501 more rows
```

```
# Make the chart
plot <- bigram_tf_idf %>%
  arrange(desc(tf_idf)) %>%
  mutate(bigram = factor(bigram, levels = rev(unique(bigram))))
plot  %>%
  top_n(20) %>%
  ggplot(aes(bigram, tf_idf, fill = book)) +
  geom_col() +
  labs(x = NULL, y = "tf-idf") +
  coord_flip()
```

```
## Selecting by tf_idf
```



I have now shown the bigrams, but in order to get an idea of how negations affect sentiment analysis, I want to find the bigrams that have the word “not” as the first word in the bigram to show how sentiment analysis was affected by negations,

```
bigrams_separated %>%
  filter(word1 == "not") %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 445 x 3
##    word1 word2       n
##    <chr> <chr>   <int>
##  1 not   a          73
##  2 not   the        71
##  3 not   to         63
##  4 not   only       44
##  5 not   be         42
##  6 not   yet        41
##  7 not   even       39
##  8 not   sure       29
##  9 not   have       25
## 10 not   imagine    18
## # ... with 435 more rows
```

Removing stopwords,

```
# Remove stopwords
bigrams_separated <- bigrams_separated %>%
  filter(word1 == "not") %>%
  filter(!word2 %in% stop_words$word)%>%
  count(word1, word2, sort = TRUE)
bigrams_separated
```

```
## # A tibble: 289 x 3
##    word1 word2          n
##    <chr> <chr>      <int>
##  1 not   imagine       18
##  2 not   possibly      12
##  3 not   surprised     11
##  4 not   understand     9
##  5 not   coming         7
##  6 not   surprising     6
##  7 not   telling        6
##  8 not   begin          5
##  9 not   breathe        5
## 10 not   fathom         5
## # ... with 279 more rows
```

```
# Use Bing lexicon
BING <- get_sentiments("bing")
not_words <- bigrams_separated %>%
  filter(word1 == "not") %>%
  filter(!word2 %in% stop_words$word)%>%
  inner_join(BING, by = c(word2 = "word")) %>%
  ungroup()
not_words
```

```
## # A tibble: 51 x 4
##    word1 word2        n sentiment
##    <chr> <chr>    <int> <chr>
##  1 not   entrust      3 positive
##  2 not   fail         3 negative
##  3 not   mistaken     3 negative
##  4 not   trust        3 positive
##  5 not   easy         2 positive
##  6 not   fall         2 negative
##  7 not   happy        2 positive
##  8 not   limited      2 negative
##  9 not   lying        2 negative
## 10 not   miss         2 negative
## # ... with 41 more rows
```

Inputs and Outputs

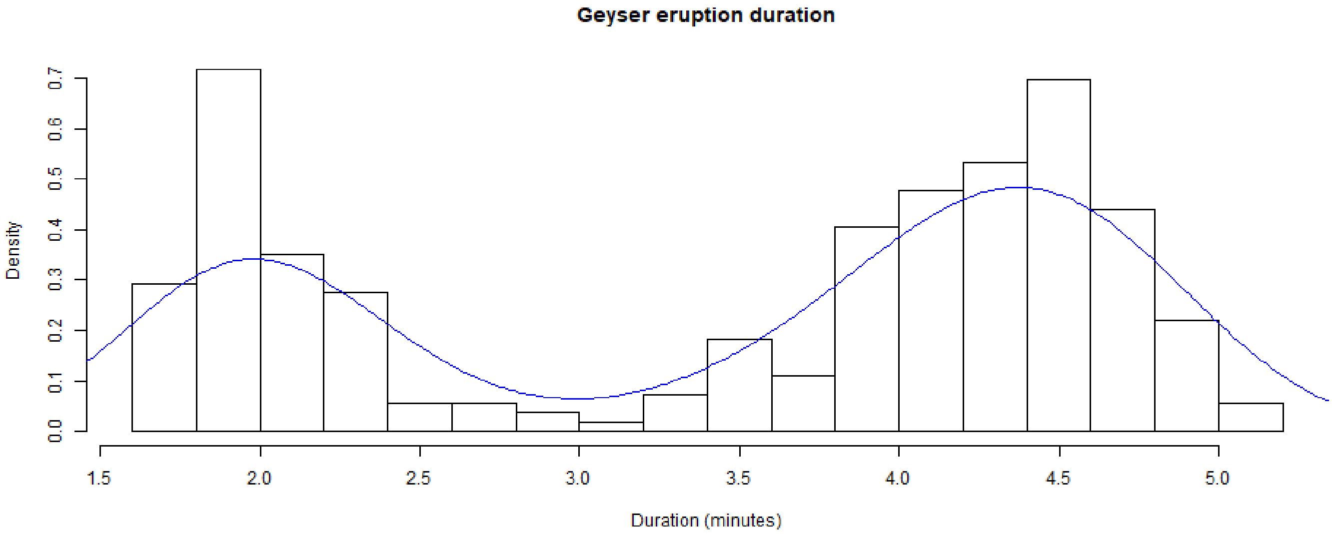
Number of bins:

20

Bandwidth adjustment:

0.2 1 2

0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2



Embedded Application

Tabsets

Distribution type:

- (•) Normal
- ( ) Uniform
- ( ) Log-normal
- ( ) Exponential

Number of observations:

1 500 1,000

1 101 201 301 401 501 601 701 801 901 1,000

Plot Summary Table