607_Fall2021_HW10_Sentiment_Analysis

Mark Schmalfeld

10/28/2021

607 Fall 2021 Week 10 Assignment - Sentiment Analysis

Problem A) Re-create the base analysis using the code from the text book and cite B) Extend analysis to a new text and new lexicon C) Complete sentiment analysis D) Reference the text from a web URL E) Publish in rpubs and github F) Provide overview of approach G) Provide any conclusions and recommendations. Which lexicon was most useful for your text? why?

```
library(tidytext)
library(tidyverse)
```

Install appropriate library to support analysis and plan.

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

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

```
library(tinytex)
library(gutenbergr)
library(SentimentAnalysis)
```

```
##
## Attaching package: 'SentimentAnalysis'

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

```
library(janeaustenr)
library(stringr)
library(lexicon)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(RColorBrewer)
```

Code from Text Mining in R by Julia Silge and David Robinson (Ref 1).

The code below is an example of the coding done in reference 1 to review the sentiment in Jane Austen's novels.

This code uses all the Jane Austen books that are in the R Studio package. After opening the combined Jane Austen books; we add a linenumber and chapter number to the file. We unnest into single words and remove stop words with anti join. A simple word count across the works is completed.

There is an example looking at Emma for sentiment (joy)

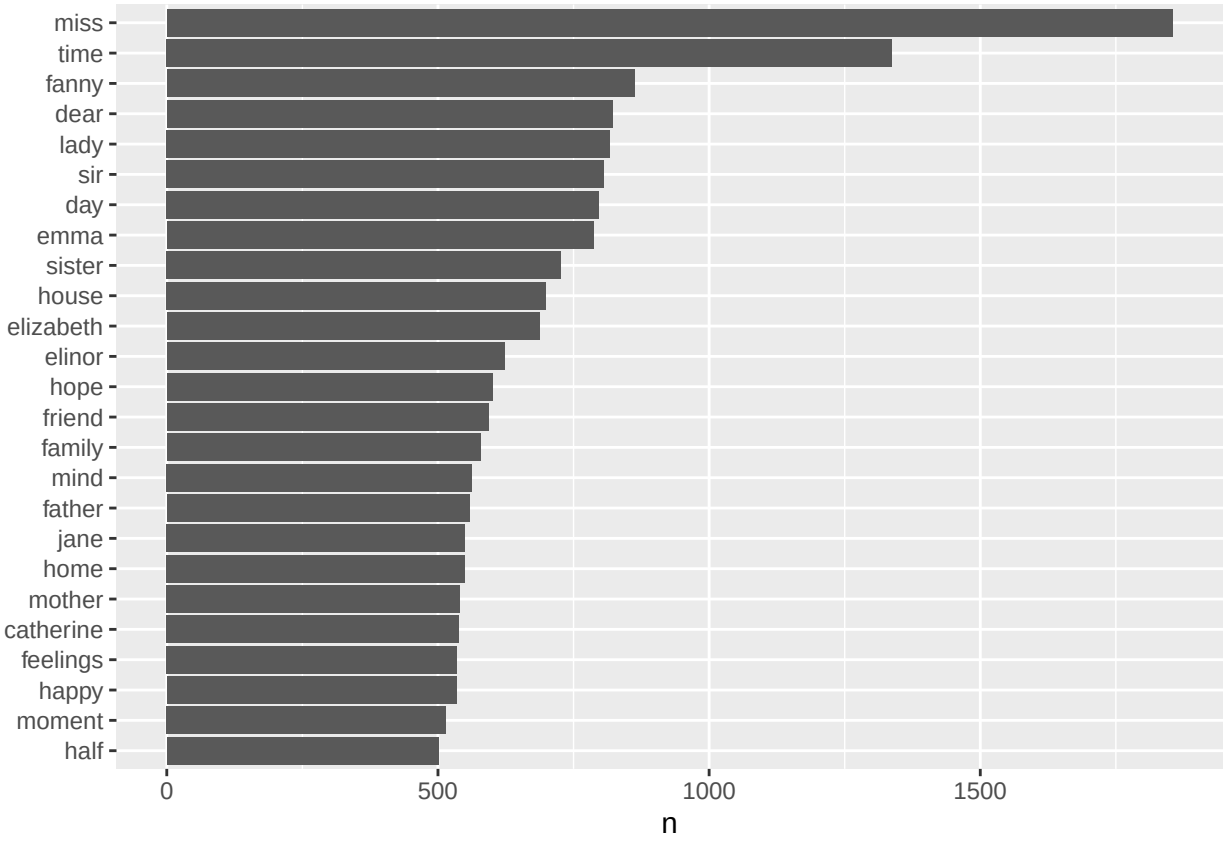
```
original_books<-austen_books() %>%
  group_by(book) %>%
  mutate(linenumber= row_number(),
         chapter= cumsum(str_detect(text, regex("^chapter [\\divxlc]",
                                                ignore_case=TRUE)))) %>%
  ungroup()

tidy_books<-original_books %>%
  unnest_tokens(word, text)
tidy_books<- tidy_books %>%
  anti_join(stop_words)
```

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

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

```
## # A tibble: 13,914 x 2
##    word       n
##    <chr>  <int>
##  1 miss    1855
##  2 time    1337
##  3 fanny    862
##  4 dear     822
##  5 lady     817
##  6 sir      806
##  7 day      797
##  8 emma     787
##  9 sister   727
## 10 house    699
## # ... with 13,904 more rows
```

```
tidy_books %>%
  count(word, sort=TRUE) %>%
  filter(n>500)  %>%
  mutate(word=reorder(word,n)) %>%
  ggplot(aes(word, n))+ geom_col()+xlab(NULL)+coord_flip()
```




```
nrcjoy<-get_sentiments("nrc") %>%
  filter(sentiment=="joy")

tidy_books %>%
  filter(book=="Emma") %>%
  inner_join(nrcjoy)  %>%
  count(word, sort=TRUE)
```

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

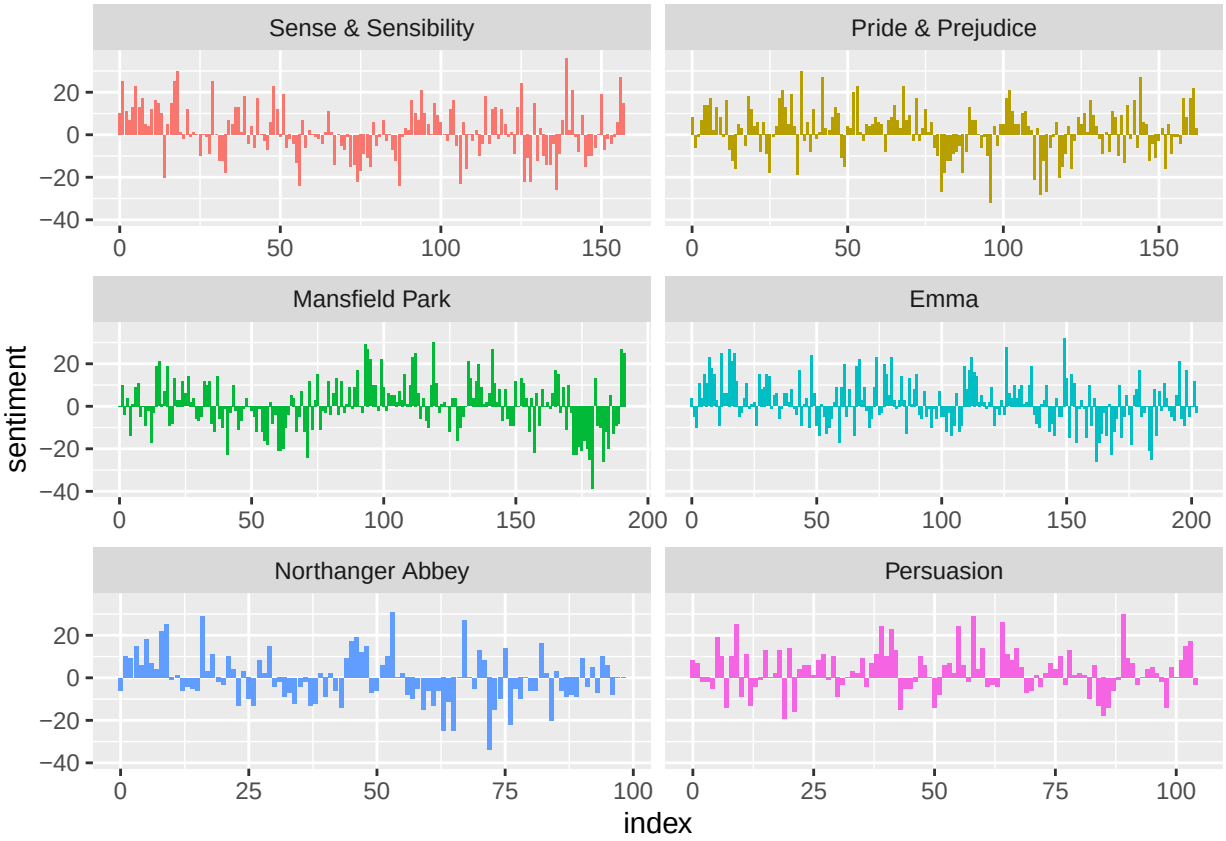
```
## # A tibble: 297 x 2
##    word          n
##    <chr>     <int>
##  1 friend      166
##  2 hope        143
##  3 happy       125
##  4 love        117
##  5 deal         92
##  6 found        92
```

```
##  7 happiness    76
##  8 pretty       68
##  9 true         66
## 10 comfort      65
## # ... with 287 more rows
```

```
janeaustensentiment <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(book, index=linenumber %/% 80, sentiment) %>%
  spread(sentiment,n,fill=0) %>%
  mutate(sentiment=positive-negative)
```

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

```
ggplot(janeaustensentiment,aes(index,sentiment,fill=book))+geom_col(show.legend=FALSE)+facet_wrap(~book
```




```
get_sentiments("nrc") %>%
  filter(sentiment %in% c("positive", "negative")) %>%
count(sentiment)
```

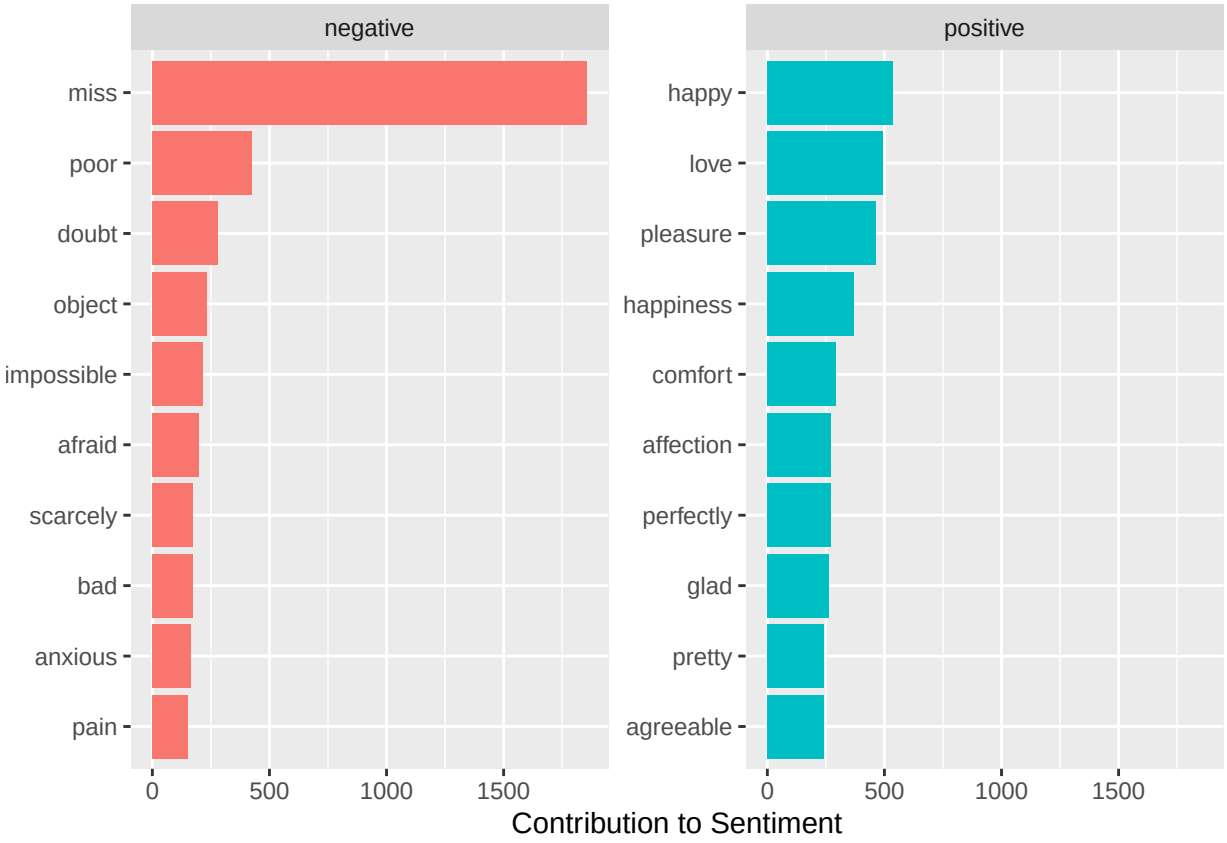
```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   3318
## 2 positive   2308
```

```
bing_word_counts_austen<- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
bing_word_counts_austen %>%
  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")+
  labs(y="Contribution to Sentiment", x=NULL)+
  coord_flip()
```

```
## Selecting by n
```



Compare the two classic Greek texts by Homer

Use gutenberg books library: https://www.gutenberg.org/ebooks/ 1728 is the Odyessy by Homer
2199 is the Iliad by Homer

We can also download all texts by Homer but would then need to eliminate those that are duplicates (or would need to focus on translator style differences)

Process and workflow

1) Download using the gutenberg_download function
2) Group by title and add line number and find chapter number
3) Unnest the words and establish the df to support analysis of the words
4) Complete a simple word count across the two texts by Homer.
5) We start to look at sentiment analysis using different specific emotion categories, positive vs negative overall and also do this with different word databases. These provide the best comparison sets of data to evaluate the different sentiments of the two novels.
6) Wordclouds are created and are mainly to be selected for specific presentation as they do not provide the detail data seen in the specific sentiment analysis.

```
homer<- gutenberg_download (c(2199,1728),mirror =NULL,strip=FALSE,meta_fields="title",)
```

```
## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest
```

```
## Using mirror http://aleph.gutenberg.org
```

```
#homer<-mutate(homer, id = row_number())

homer_book <-homer%>%
  group_by(title) %>%
  mutate(linenumber= row_number(),
         chapter= cumsum(str_detect(text, regex("^Book [\\divxlc]",
                                                ignore_case=TRUE)))) %>%
  ungroup()



tidy_homer<-homer_book %>%
  unnest_tokens(word,text) %>%
  anti_join (stop_words)
```

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

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

```
## # A tibble: 10,635 x 2
##    word         n
##    <chr>    <int>
##  1 son       1403
##  2 thou       936
##  3 odysseus   718
##  4 achaeans   662
##  5 thee       658
##  6 spake      657
##  7 thy        622
##  8 trojans    597
##  9 ships      579
## 10 heart      549
## # ... with 10,625 more rows
```

Sentiment Analysis of the Odyssey and the Iliad by Homer using the NRC sentiment analysis framework.

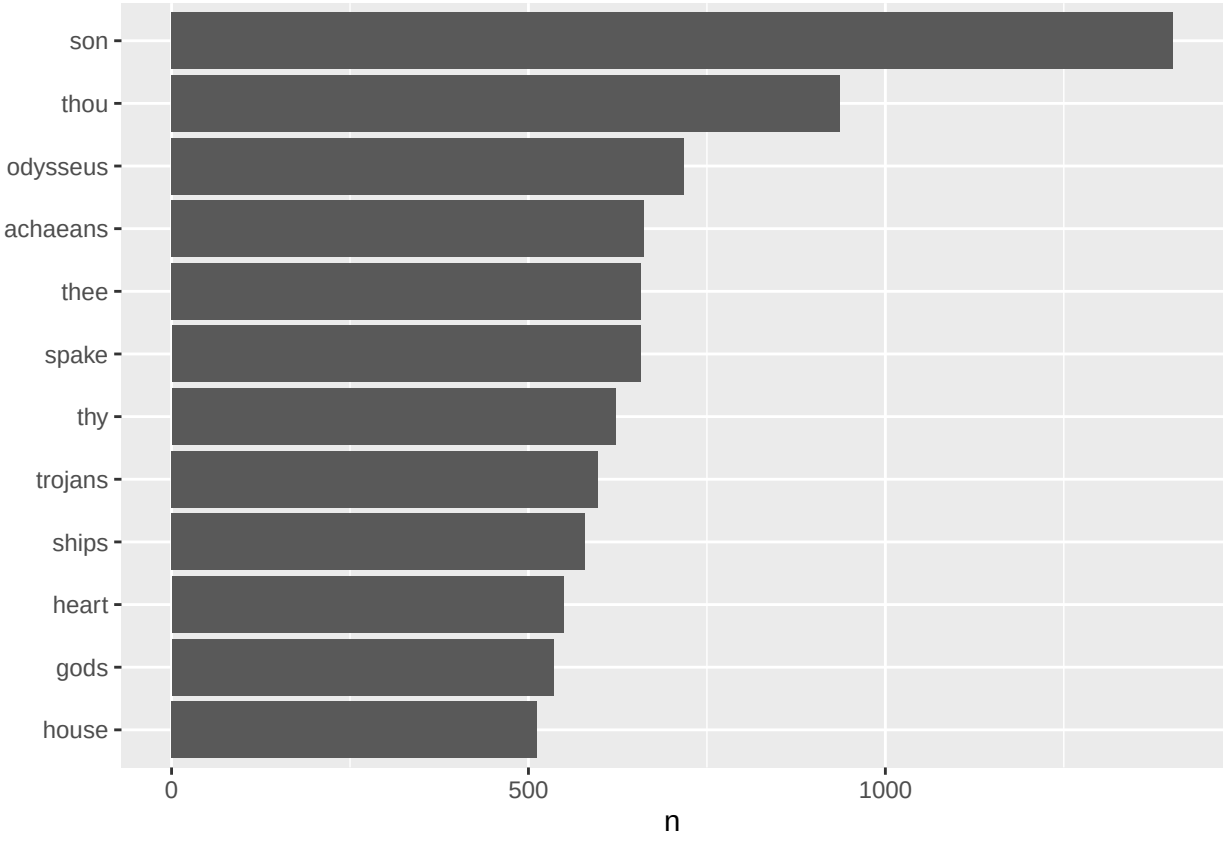
NRC Lexicon cited: Rstudio Lexicons covering all the lexicons used and the NRC page http: //saifmohammad.com/WebPages/lexicons.html were used. Ref details included in references.

#Overview of approach:

Compare two specific historical works by one attributed author - Homer. The two works are occur during a similar historic time period but are very different in context. The Iliad concerns the siege of Troy occurring due to the stolen wife of a Greek King and contains significant periods of battle scenes, unrest, war, and uncertainty - fear through the novel. The Odyessy concerns the period after the Iliad and while it contains many challenges, events that can cause fear but the tone is general thought to be more upbeat and positive as a returning hero (or delayed return) to his home and beseiged wife (by suitors) It has an tone of adventure and eventually leads to Odyessus return to his homeland and home.

Using these two texts allows a chance to evaluate how the sentiment analysis would confirm or not confirm expectations based on my reading of the two texts.

```
library(tidytext)

tidy_homer %>%
  count(word, sort=TRUE) %>%
  filter(n>500)  %>%
  mutate(word=reorder(word,n)) %>%
  ggplot(aes(word, n))+ geom_col()+xlab(NULL)+coord_flip()
```

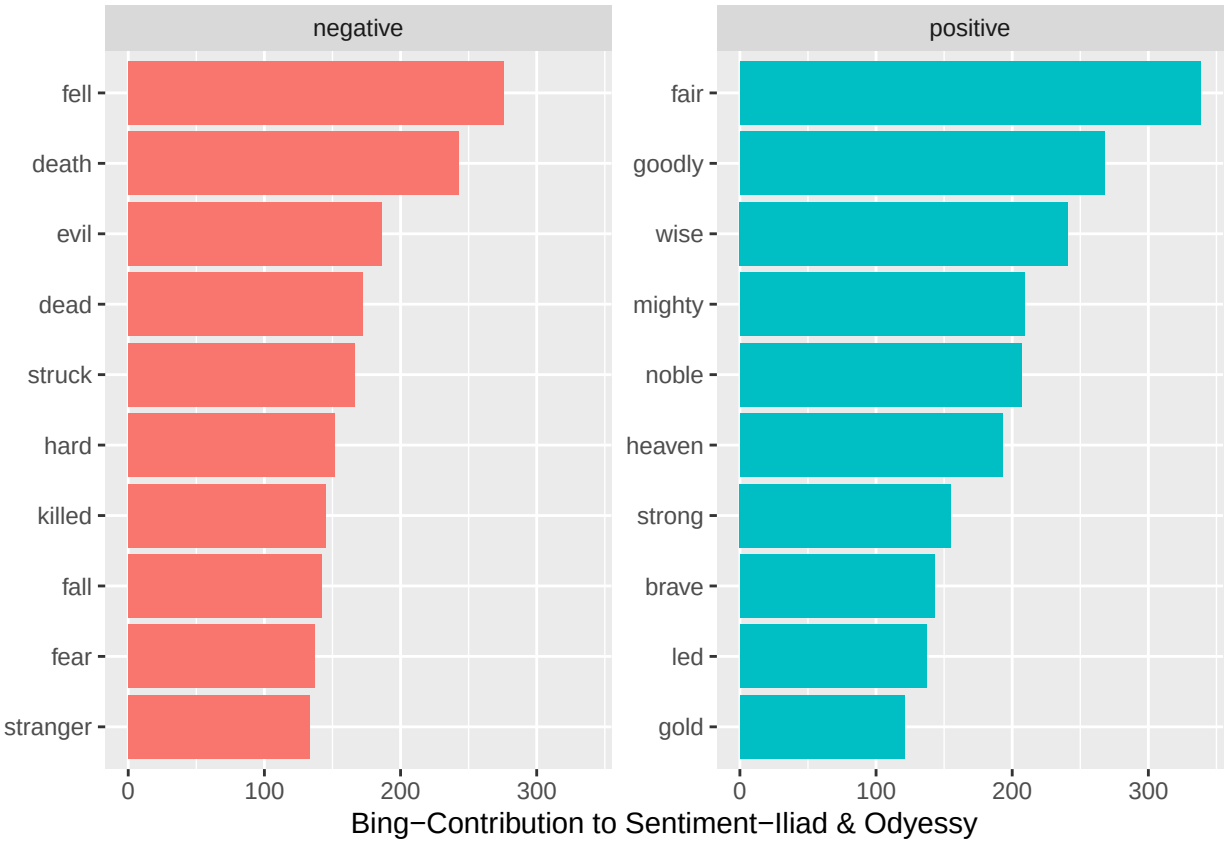


```
# Bing Lexicon
bing_word_counts_homer<- tidy_homer %>%
  inner_join(get_sentiments("bing")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
bing_word_counts_homer %>%
  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")+
  labs(y="Bing-Contribution to Sentiment-Iliad & Odyessy", x=NULL)+
  coord_flip()
```

```
## Selecting by n
```

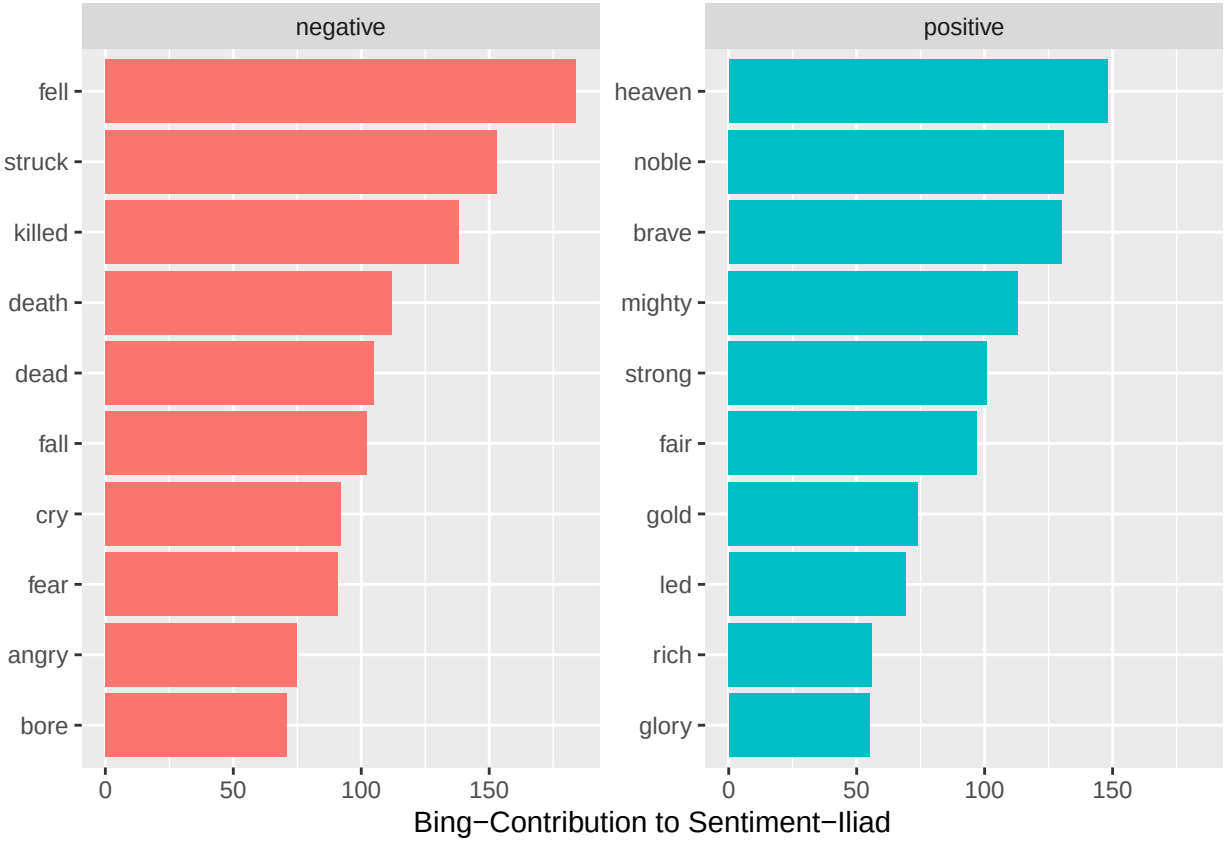



```
bing_word_counts_iliad<- tidy_homer %>%
  filter(gutenberg_id==2199) %>%    # Iliad
  inner_join(get_sentiments("bing")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
bing_word_counts_iliad %>%
  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")+
  labs(y="Bing-Contribution to Sentiment-Iliad", x=NULL)+
  coord_flip()
```

```
## Selecting by n
```



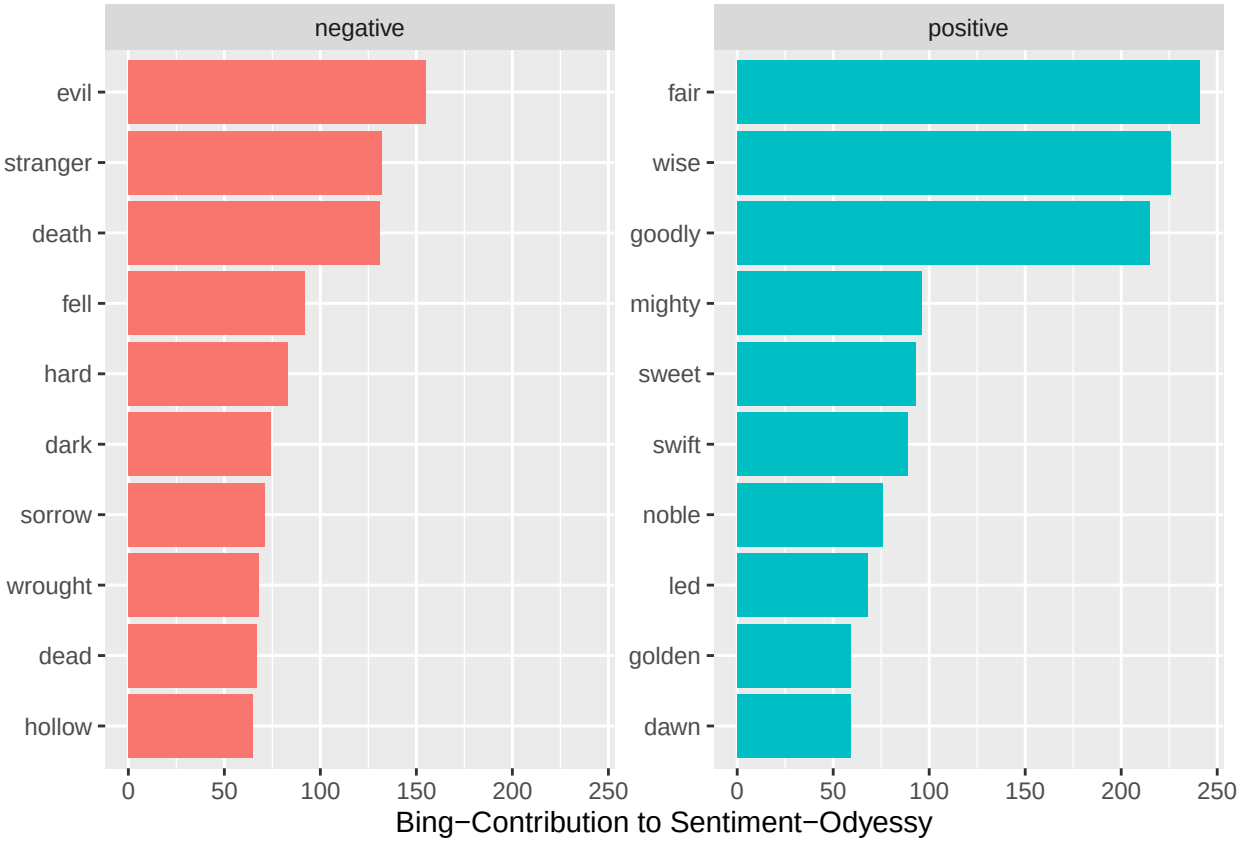

```
bing_word_counts_odyessy<- tidy_homer %>%
  filter(gutenberg_id==1728) %>%    # Odyessy
  inner_join(get_sentiments("bing")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
bing_word_counts_odyessy %>%
  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")+
  labs(y="Bing-Contribution to Sentiment-Odyessy", x=NULL)+
  coord_flip()
```

```
## Selecting by n
```



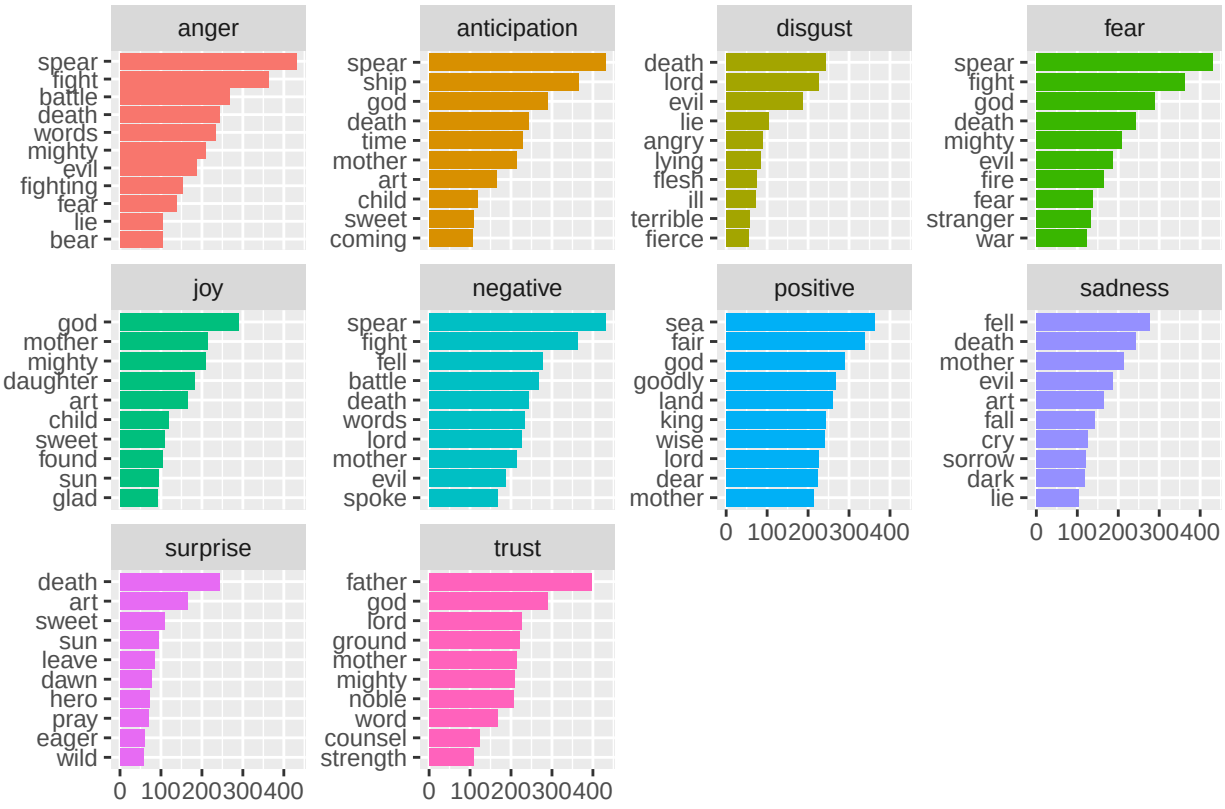

```
#NRC Lexicon

nrc_word_counts_homer<- tidy_homer %>%
  inner_join(get_sentiments("nrc")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
nrc_word_counts_homer %>%
  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")+
  labs(y="NRC-Contribution to Sentiment-Iliad & Odyessy", x=NULL)+
  coord_flip()
```

```
## Selecting by n
```

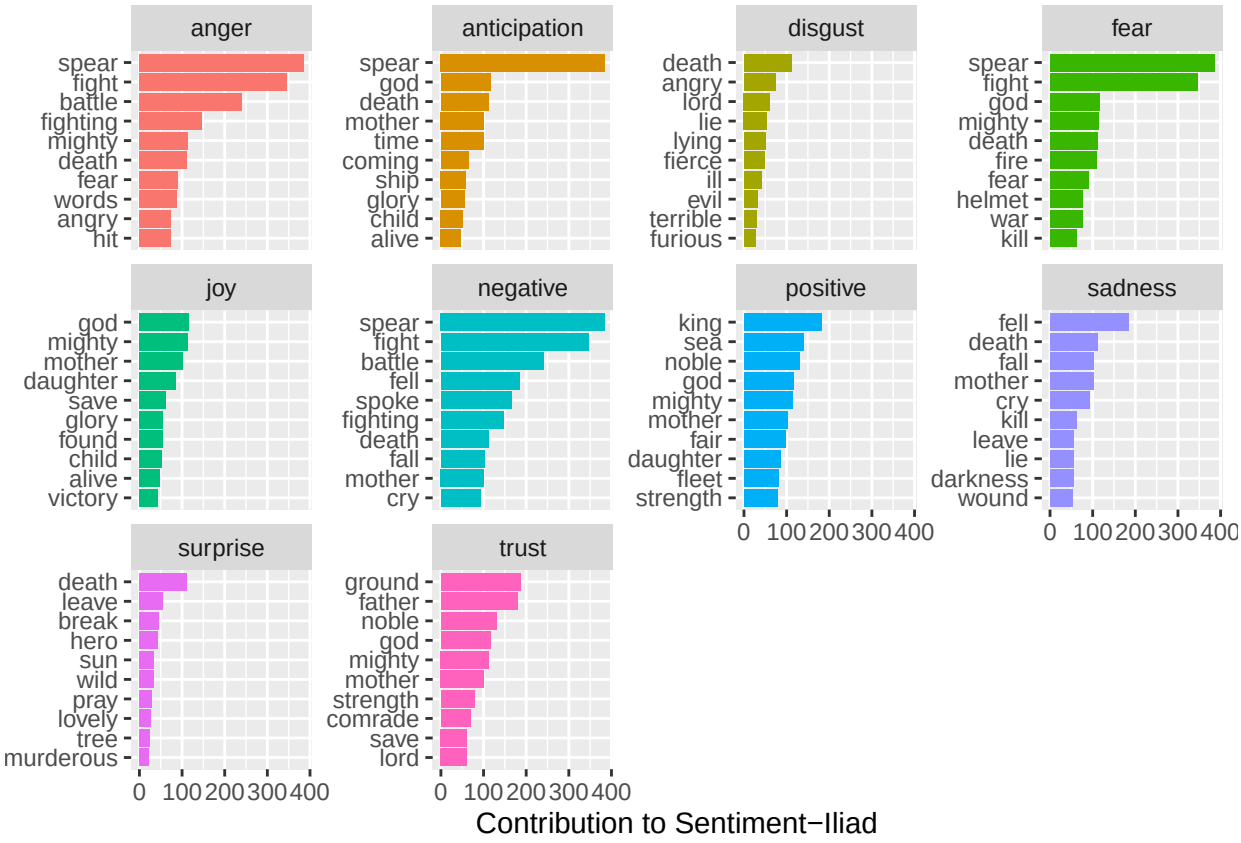



```
nrc_word_counts_iliad<- tidy_homer %>%
  filter(gutenberg_id==2199) %>%    # Iliad
  inner_join(get_sentiments("nrc")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
nrc_word_counts_iliad %>%
  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")+
  labs(y="Contribution to Sentiment-Iliad", x=NULL)+
  coord_flip()
```

```
## Selecting by n
```




```
nrc_word_counts_odyessy<- tidy_homer %>%
  filter(gutenberg_id==1728) %>%    # Odyessy
  inner_join(get_sentiments("nrc")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
nrc_word_counts_odyessy %>%
  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")+
  labs(y="NRC-Contribution to Sentiment-Odyessy", x=NULL)+
  coord_flip()
```

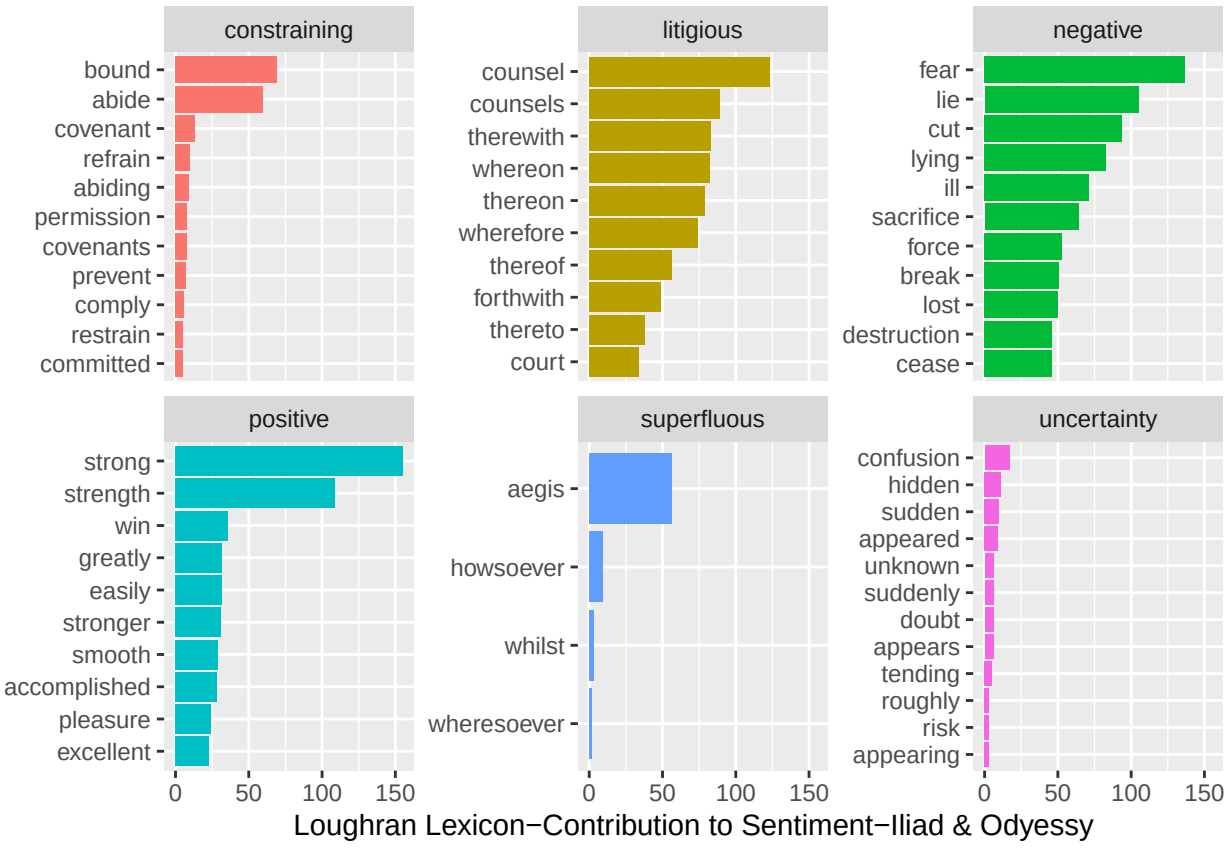
```
## Selecting by n
```


```
#loughran Lexicon

loughran_word_counts_homer<- tidy_homer %>%
  inner_join(get_sentiments("loughran")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
loughran_word_counts_homer %>%
  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")+
  labs(y="Loughran Lexicon-Contribution to Sentiment-Iliad & Odyessy", x=NULL)+
  coord_flip()
```

```
## Selecting by n
```

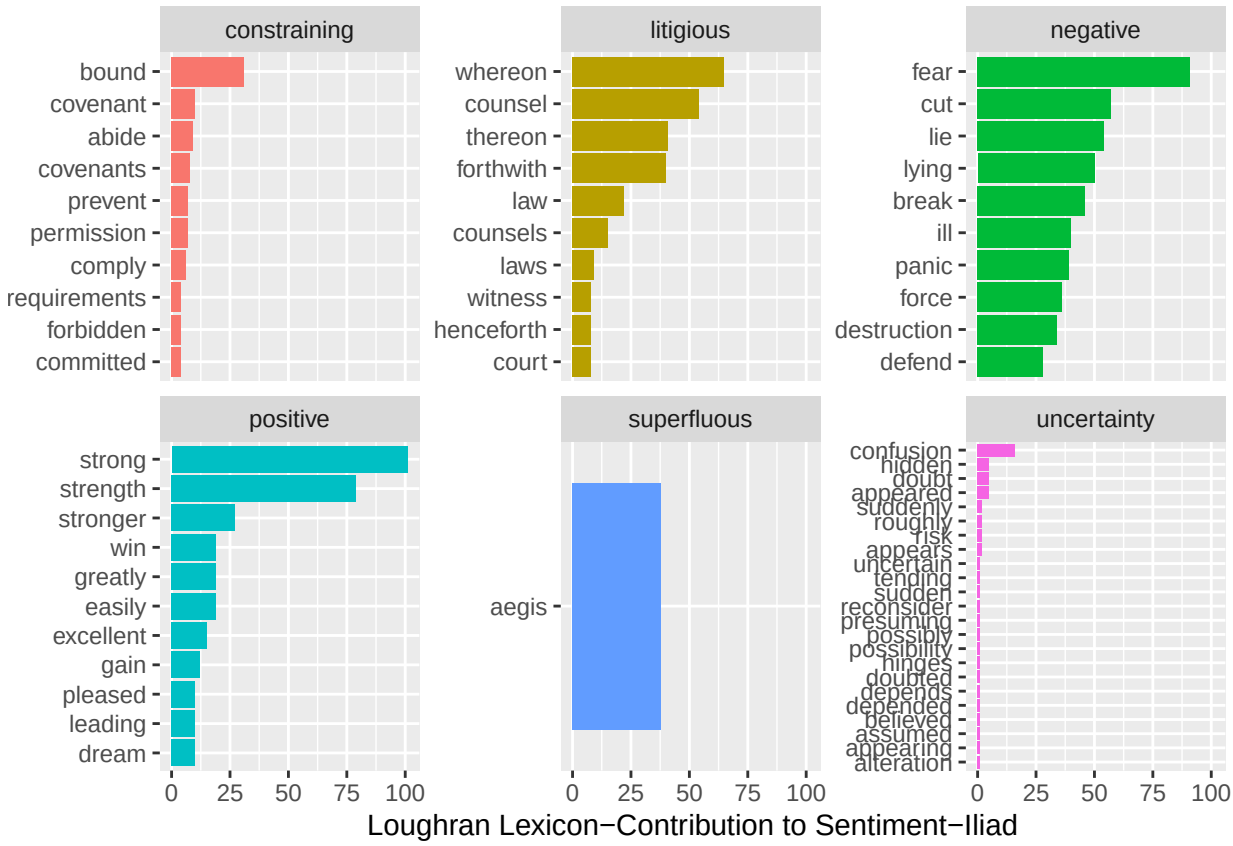



```
loughran_word_counts_iliad<- tidy_homer %>%
  filter(gutenberg_id==2199) %>%    # Iliad
  inner_join(get_sentiments("loughran")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
loughran_word_counts_iliad %>%
  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")+
  labs(y="Loughran Lexicon-Contribution to Sentiment-Iliad", x=NULL)+
  coord_flip()
```

```
## Selecting by n
```

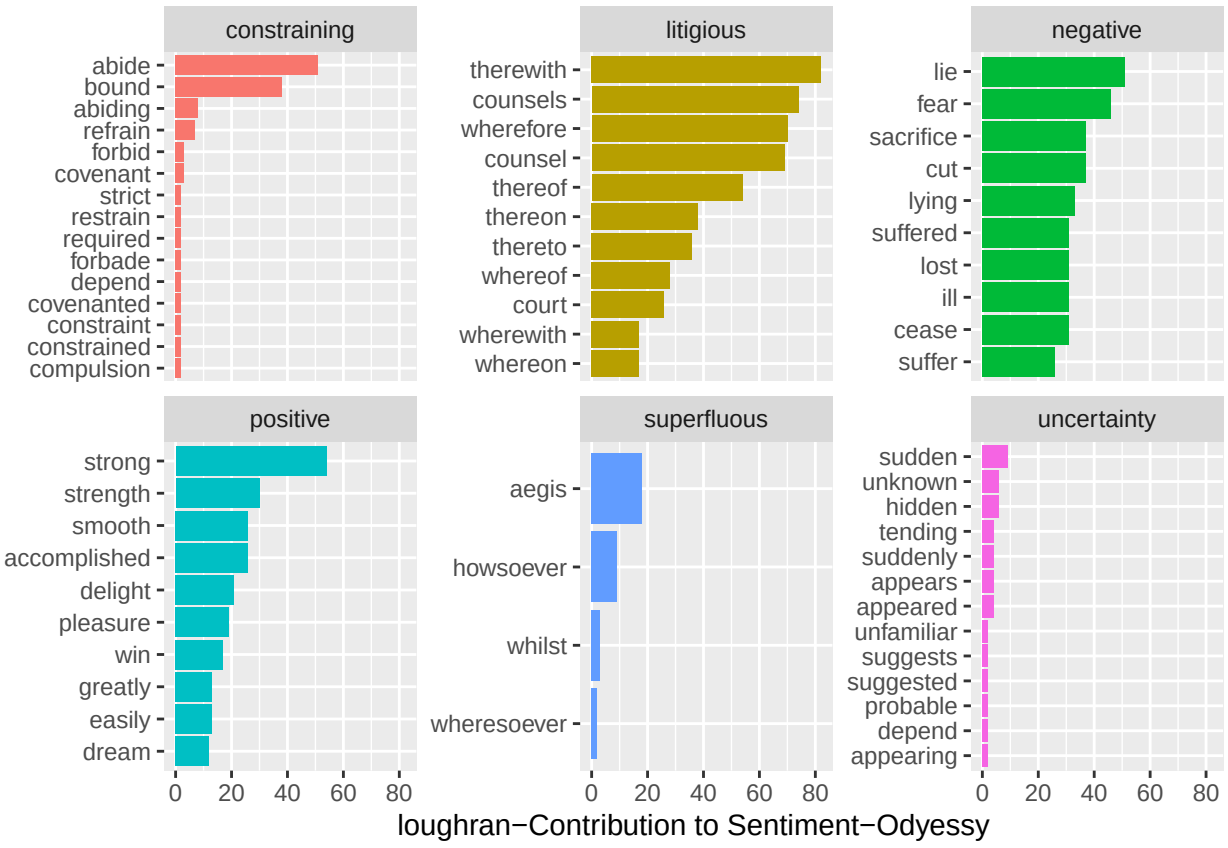



```
loughran_word_counts_odyessy<- tidy_homer %>%
  filter(gutenberg_id==1728) %>%    # Odyessy
  inner_join(get_sentiments("loughran")) %>%
  count(word,sentiment, sort=TRUE) %>%
ungroup()
```

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

```
loughran_word_counts_odyessy %>%
  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")+
  labs(y="loughran-Contribution to Sentiment-Odyessy", x=NULL)+
  coord_flip()
```

```
## Selecting by n
```




```
nrcjoy<-get_sentiments("nrc") %>%
  filter(sentiment=="joy")

joyOdyssey<-tidy_homer %>%
  filter(gutenberg_id==1728) %>%    # The Odyssey
  inner_join((nrcjoy)) %>%
  count(word,sort=TRUE)
```

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

```
joyIliad<-tidy_homer %>%
  filter(gutenberg_id==2199) %>%    # Iliad
  inner_join((nrcjoy)) %>%
  count(word,sort=TRUE)
```

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

```
wordcloud(words = joyOdyssey$word, freq = joyOdyssey$n, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```


```
wordcloud(words = joyIliad$word, freq = joyIliad$n, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```


```
nrcfear<-get_sentiments("nrc") %>%
  filter(sentiment=="fear")

fearOdyssey<-tidy_homer %>%
  filter(gutenberg_id==1728) %>%    # The Odyssey
  inner_join((nrcfear)) %>%
  count(word,sort=TRUE)
```

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

```
fearIliad<-tidy_homer %>%
  filter(gutenberg_id==2199) %>%    # Iliad
  inner_join((nrcfear)) %>%
  count(word,sort=TRUE)
```

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

```
wordcloud(words = fearOdyssey$word, freq = fearOdyssey$n, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```


```
wordcloud(words = fearIliad$word, freq = fearIliad$n, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```


```
nrctrust<-get_sentiments("nrc") %>%
  filter(sentiment=="trust")
```

```
trustodyssey<-tidy_homer %>%
  filter(gutenberg_id==1728) %>%    # The Odyssey
  inner_join((nrctrust)) %>%
  count(word,sort=TRUE)
```

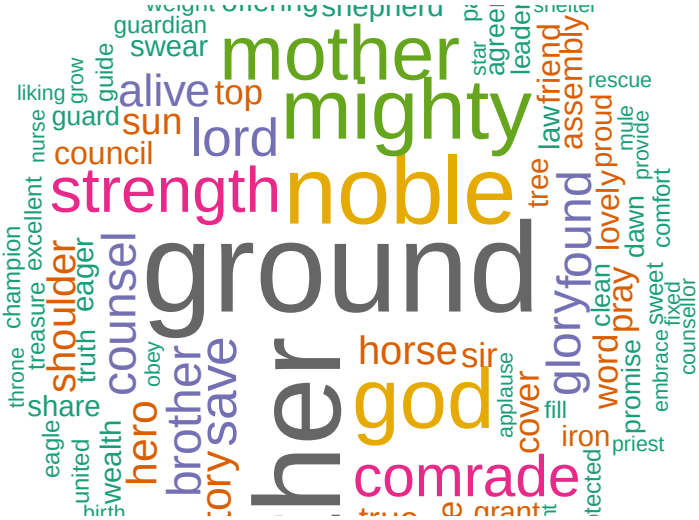
```
## Joining, by = "word"
```

```
trustyIliad<-tidy_homer %>%
  filter(gutenberg_id==2199) %>%    # Iliad
  inner_join((nrctrust)) %>%
  count(word,sort=TRUE)
```

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

```
wordcloud(words = trustodyssey$word, freq = trustodyssey$n, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```


```
wordcloud(words = trustyIliad$word, freq = trustyIliad$n, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"),main="Trust")
```




```
nrcnegative<-get_sentiments("nrc") %>%
  filter(sentiment=="negative")

odyssey_negative<-tidy_homer %>%
  filter(gutenberg_id==1728) %>%    # The Odyssey
  inner_join((nrcnegative)) %>%
  count(word,sort=TRUE)
```

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

```
Iliad_negative<-tidy_homer %>%
  filter(gutenberg_id==2199) %>%    # Iliad
  inner_join((nrcnegative)) %>%
  count(word,sort=TRUE)
```

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

```
nrcpositive<-get_sentiments("nrc") %>%
  filter(sentiment=="positive")


Odyssey_positive<-tidy_homer %>%
  filter(gutenberg_id==1728) %>%    # The Odyssey
  inner_join((nrcpositive)) %>%
  count(word,sort=TRUE)
```

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

```
Iliad_positive<-tidy_homer %>%
  filter(gutenberg_id==2199) %>%    # Iliad
  inner_join((nrcpositive)) %>%
  count(word,sort=TRUE)
```

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

```
lgnegative<-get_sentiments("loughran") %>%
  filter(sentiment=="negative")

odyssey_negativelg<-tidy_homer %>%
  filter(gutenberg_id==1728) %>%    # The Odyssey
  inner_join((lgnegative)) %>%
  count(word,sort=TRUE)
```

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

```
Iliad_negativelg<-tidy_homer %>%
  filter(gutenberg_id==2199) %>%    # Iliad
  inner_join((lgnegative)) %>%
  count(word,sort=TRUE)
```

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

```
#generate word cloud
library(wordcloud)
set.seed(1234)
wordcloud(words = nrc_word_counts_homer$word, freq = nrc_word_counts_homer$n, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```

```
## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : battle could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : death could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : death could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : death could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : death could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : wise could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : words could not be fit on page. It will not be
## plotted.
```

```
## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : words could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : dear could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : ground could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mother could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mother could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mother could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mother could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mother could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mother could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mighty could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mighty could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mighty could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mighty could not be fit on page. It will not be
## plotted.
```

```
## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : mighty could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : noble could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : noble could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : evil could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : evil could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : daughter could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : daughter could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : spoke could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : word could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : word could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : art could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : art could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : fire could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : fighting could not be fit on page. It will not be
```

```
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : fighting could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : nay could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : fall could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : fear could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : stranger could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : stranger could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : cry could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : cry could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : war could not be fit on page. It will not be plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : counsel could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : counsel could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : sorrow could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : sorrow could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : gold could not be fit on page. It will not be
## plotted.
```

```
## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : child could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : child could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : child could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : spirit could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : rest could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : strength could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : strength could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : sweet could not be fit on page. It will not be
## plotted.

## Warning in wordcloud(words = nrc_word_counts_homer$word, freq =
## nrc_word_counts_homer$n, : sweet could not be fit on page. It will not be
## plotted.
```

```
homersentiment <- tidy_homer %>%
  inner_join(get_sentiments("bing")) %>%
  count(title, index=linenumber %/% 80, sentiment) %>%
  spread(sentiment,n,fill=0) %>%
  mutate(sentiment=positive-negative)
```

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

```
ggplot(homersentiment,aes(index,sentiment,fill=title))+geom_col(show.legend=FALSE)+facet_wrap(~title,nc
```

Conclusions

There were two different types of analysis that I found most useful:

NCR Sentiment by emotion and positive/negative sentiment:

The NRC sentiment analysis was the most useful to me to see the variation in count of the types of emotions and positive/negative sentiment at a more detailed level.

The higher negative count seems to confirm the Iliad has a darker tone and sentiment than the Odyessy which was more positive overall in sentiment. The Iliad had higher indications of fear and anger while the Odyessy had higher indications of Joy and Trust. The Iliad had a much lower indication of joy / surprise than the Odyessy.

Bing analysis through the book plotting change over time was also very useful

This plot of the sentiment through the two books very directly shows how negative the sentiment is in the Iliad throughout the entire book. The Iliad is not only negative it is also much more negative than any cases seen in the Odyessy.

The plot of the sentiment seen in the Odyessy is more typical of a novel with periods of up and down sentiment but the overall sentiment is more balances with periods of positive and negative sentiment. The negative periods are not as strong as seen in the Iliad. The Odyessy also has some periods of very positive sentiment. The book is interesting that after such a large number of highs and lows the end is a relative calm ending with the household sort of returning to normal in the book which seems to be confirmed by the sentiment analysis.

Recommendations are to evaluate some of the color sentiment as this would also be expected to provide some indications of the differences in the two books. The Odyessy takes place on the sea for a large part of it and the descriptions of the sea and would be intersting to understand as they did not describe it as blue. The Iliad occurred with an encampment on the beach and in a fort - so the analysis here would also be intersting to compare and see if there are any noticable trends between the two works.

###References:

1) Silge, J. & Robinson, D. (2017). "Text Mining with R". Oreilly.
2) Project Gutenberg site https://www.gutenberg.org/ebooks/
3) Hill, C. "Sentiment Analysis Lexicon", Rstudio. https://rpubs.com/chelseyhill/676279
4) Mhatre, S. (2020). "Text Mining and Sentiment Analysis: Analysis with R". https://www.red-gate.com/simple-talk/databases/sql-server/bi-sql-server/text-mining-and-sentiment-analysis-with-r/
5) Mohammad, S. M. (2014) "Sentiment and emotion Lexicon". http://saifmohammad.com/WebPages/lexicons.html