

Sentiment Analysis Project in R

Today, we are starting our series of R projects and the first one is Sentiment analysis. So, in this article, we will develop our very own project of sentiment analysis using R. We will make use of the `tiny text` package to analyze the data and provide scores to the corresponding words that are present in the dataset. In the end, you will become industry ready to solve any problem related to R programming.

Before we start, you must take a quick revision to *R concepts*.

R Project – Sentiment Analysis

The aim of this project is to build a sentiment analysis model which will allow us to categorize words based on their sentiments, that is whether they are positive, negative and also the magnitude of it. Before we start with our R project, let us understand sentiment analysis in detail.

What is Sentiment Analysis?

Sentiment Analysis is a process of extracting opinions that have different polarities. By polarities, we mean positive, negative or neutral. It is also known as opinion mining and polarity detection. With the help of sentiment analysis, you can find out the nature of opinion that is reflected in documents, websites, social media feed, etc. Sentiment Analysis is a type of classification where the data is classified into different classes. These

classes can be binary in nature (positive or negative) or, they can have multiple classes (happy, sad, angry, etc.).

Developing our Sentiment Analysis Model in R

We will carry out sentiment analysis with R in this project. The dataset that we will use will be provided by the R package ‘janeaustenR’.

In order to build our project on sentiment analysis, we will make use of the tidytext package that comprises of sentiment lexicons that are present in the dataset of ‘sentiments’.

Syntax:

```
library(tidytext)
```

```
Sentiments
```

Screenshot:

[Hide](#)

```
library(tidytext)
sentiments
```

word <chr>	sentiment <chr>
ache	negative
ached	negative
aches	negative
achey	negative
achievable	positive
achievement	positive
achievements	positive
achievable	positive
aching	negative
acrid	negative

61-70 of 6,786 rows

[Previous](#)
[1](#)
[...](#)
[5](#)
[6](#)
[7](#)
[8](#)
[9](#)
[...](#)
[100](#)
[Next](#)

We will make use of three general purpose lexicons like –

- AFINN
- Bing
- Loughran

These three lexicons make use of the unigrams. Unigrams are a type of n-gram model that consists of a sequence of 1 item, that is, a word collected from a given textual data. In the AFINN lexicon model scores the words in a range from -5 to 5. The increase in negativity corresponds the negative sentiment whereas an increase in positivity corresponds the positive one.

The bing lexicon model on the other hand, classifies the sentiment into a binary category of negative or positive. And finally, the loughran model that performs analysis of the shareholder's reports. In this project, we will make use of the bing lexicons to extract the sentiments out of our data. We can retrieve these lexicons using the `get_sentiments()` function. We can implement this as follows –

Syntax:

```
get_sentiments("bing")
```

Screenshot:

[Hide](#)

<code>get_sentiments("bing")</code>	
word <chr>	sentiment <chr>
2-faces	negative
abnormal	negative
abolish	negative
abominable	negative
abominably	negative
abominate	negative
abomination	negative
abort	negative
aborted	negative
aborts	negative

1-10 of 6,786 rows

Previous
1
2
3
4
5
6
...
100
Next

While practicing the R concepts through this project, I recommend you to start your R interview preparations also. Here are some of the best R interview questions that are mostly asked and will surely help you in the future.

Let's continue to R sentiment analysis

Performing Sentiment Analysis with the Inner Join

In this step, we will import our libraries ‘janeaustenr’, ‘stringr’ as well as ‘tidytext’. The janeaustenr package will provide us with the textual data in the form of books authored by the novelist [Jane Austen](#). Tidytext will allow us to perform efficient text analysis on our data. We will convert the text of our books into a tidy format using `unnest_tokens()` function.

Syntax:

```
library(janeaustenr)

library(stringr)

library(tidytext)

tidy_data <- austen_books() %>%

group_by(book) %>%

mutate(linenumber = row_number(),

       chapter = cumsum(str_detect(text, regex("^chapter [\\divxlc]",

                                              ignore_case = TRUE)))) %>%

ungroup() %>%

unnest_tokens(word, text)
```

Screenshot:

```
library(janeaustenr)
library(stringr)
library(tidytext)

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

We have performed the tidy operation on our text such that each row contains a single word. We will now make use of the “bing” lexicon to and implement filter() over the words that correspond to joy. We will use the book Sense and Sensibility and derive its words to implement our sentiment analysis model.

Syntax:

```
positive_senti <- get_sentiments("bing") %>%

filter(sentiment == "positive")

tidy_data %>%

filter(book == "Emma") %>%

semi_join(positive_senti) %>%

count(word, sort = TRUE)
```

Screenshot:

```
positive_senti <- get_sentiments("bing") %>%
  filter(sentiment == "positive")

tidy_data %>%
  filter(book == "Emma") %>%
  semi_join(positive_senti) %>%
  count(word, sort = TRUE)
```

Joining, by = "word"

word	n
<chr>	<int>
well	401
good	359
great	264
like	200
better	173
enough	129
happy	125
love	117
pleasure	115
right	92

1-10 of 668 rows

Previous 1 2 3 4 5 6 ... 67 Next

From our above result, we observe many positive words like “good”, “happy”, “love” etc. In the next step, we will use `spread()` function to segregate our data into separate columns of positive and negative sentiments. We will then use the `mutate()` function to calculate the total sentiment, that is, the difference between positive and negative sentiment.

Syntax:

```
library(tidyr)

bing <- get_sentiments("bing")

Emma_sentiment <- tidy_data %>%

inner_join(bing) %>%
```

```
count(book = "Emma" , index = linenumbr %% 80, sentiment) %>%

spread(sentiment, n, fill = 0) %>%

mutate(sentiment = positive - negative)
```

Screenshot:

Hide

```
library(tidyr)
bing <- get_sentiments("bing")
Emma_sentiment <- tidy_data %>%
  inner_join(bing) %>%
  count(book = "Emma" , index = linenumbr %% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

Have you checked the [importance of R for Data Scientists](#)?

In the next step, we will visualize the words present in the book “Emma” based on their corresponding positive and negative scores.

Syntax:

```
library(ggplot2)

ggplot(Emma_sentiment, aes(index, sentiment, fill = book)) +

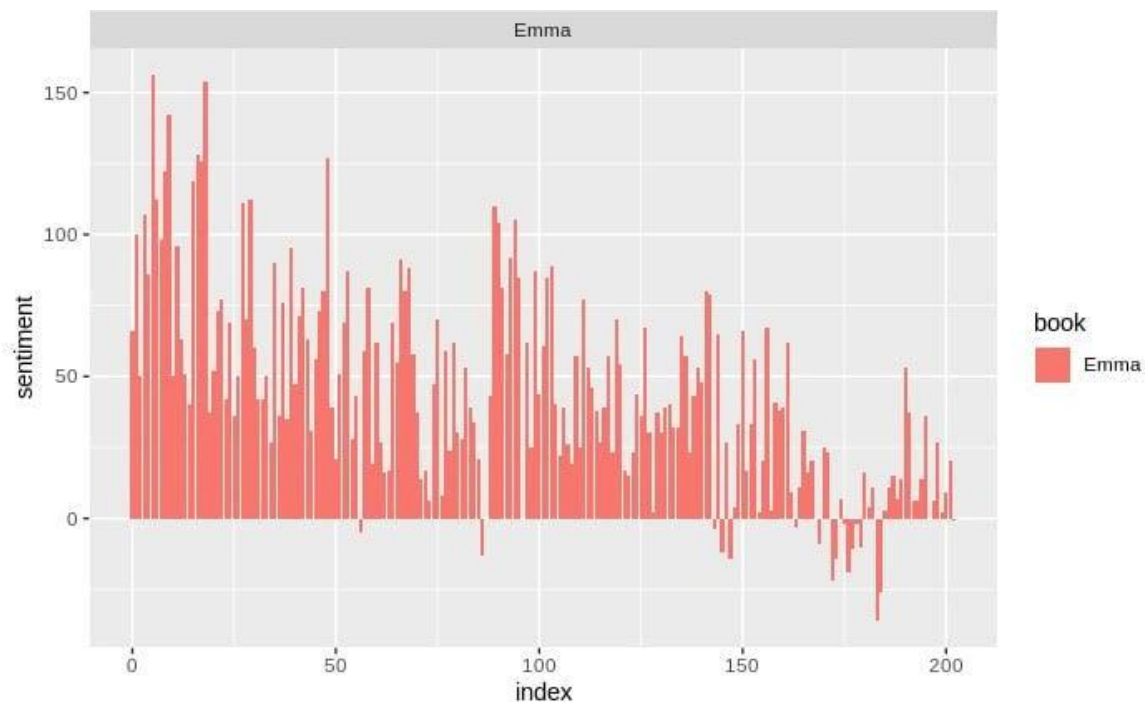
geom_bar(stat = "identity", show.legend = TRUE) +

facet_wrap(~book, ncol = 2, scales = "free_x")
```

Screenshot:

```
library(ggplot2)

ggplot(Emma_sentiment, aes(index, sentiment, fill = book)) +
  geom_bar(stat = "identity", show.legend = TRUE) +
  facet_wrap(~book, ncol = 2, scales = "free_x")
```

Let us now proceed towards counting the most common positive and negative words that are present in the novel.

```
counting_words <- tidy_data %>%
```

```
inner_join(bing) %>%
```

```
count(word, sentiment, sort = TRUE)
```

```
head(counting_words)
```

Screenshot:

Hide

```
counting_words <- tidy_data %>%
  inner_join(bing) %>%
  count(word, sentiment, sort = TRUE)
```

Joining, by = "word"

Hide

```
head(counting_words)
```

word <chr>	sentiment <chr>	n <int>
miss	negative	1855
well	positive	1523
good	positive	1380
great	positive	981
like	positive	725
better	positive	639

6 rows

In the next step, we will perform visualization of our sentiment score. We will plot the scores along the axis that is labeled with both positive as well as negative words. We will use `ggplot()` function to visualize our data based on their scores.

Don't forget to check our latest guide on [data visualization using R](#).

Syntax:

```
counting_words %>%

filter(n > 150) %>%

mutate(n = ifelse(sentiment == "negative", -n, n)) %>%

mutate(word = reorder(word, n)) %>%

ggplot(aes(word, n, fill = sentiment))+

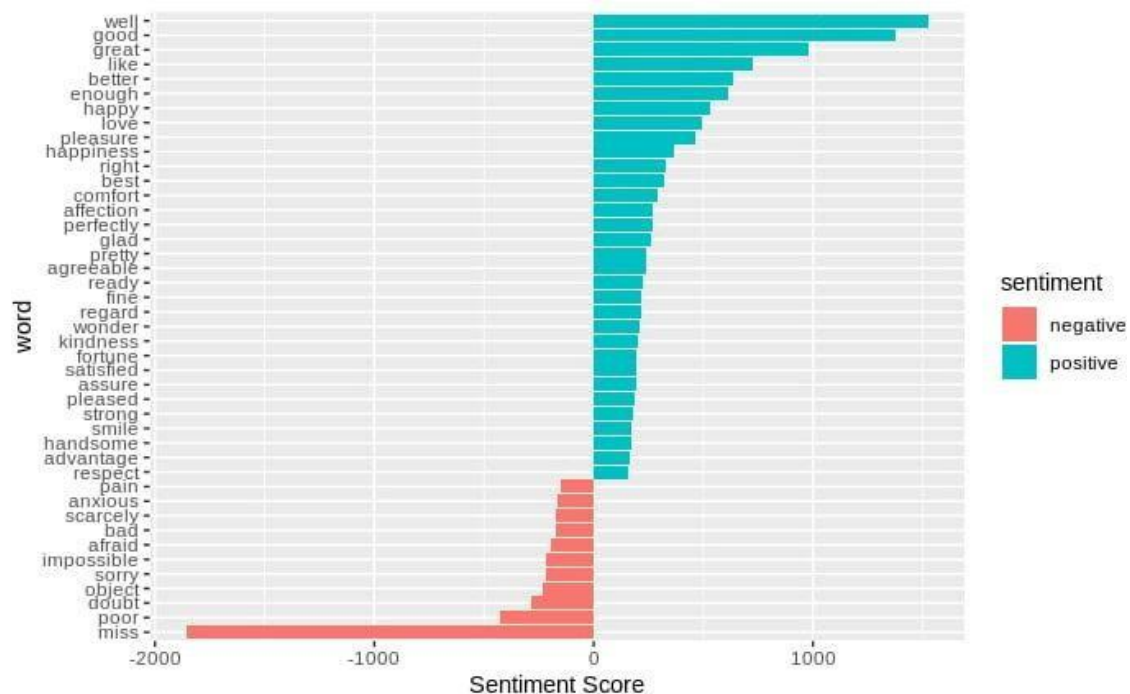
geom_col() +

coord_flip() +
```

```
labs(y = "Sentiment Score")
```

Screenshot:

```
counting_words %>%
  filter(n > 150) %>%
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col() +
  coord_flip() +
  labs(y = "Sentiment Score")
```



In the final visualization, let us create a wordcloud that will delineate the most recurring positive and negative words. In particular, we will use the `comparision.cloud()` function to plot both negative and positive words in a single wordcloud as follows:



Summary

In this blog, we went through our project of sentiment analysis in R. We learnt about the concept of sentiment analysis and implemented it over the dataset of Jane Austen's books. We were able to delineate it through various visualizations after we performed data wrangling on our data. We used a lexical analyzer – 'bing' in this instance of our project. Furthermore, we also represented the sentiment score through a plot and also made a visual report of wordcloud.