

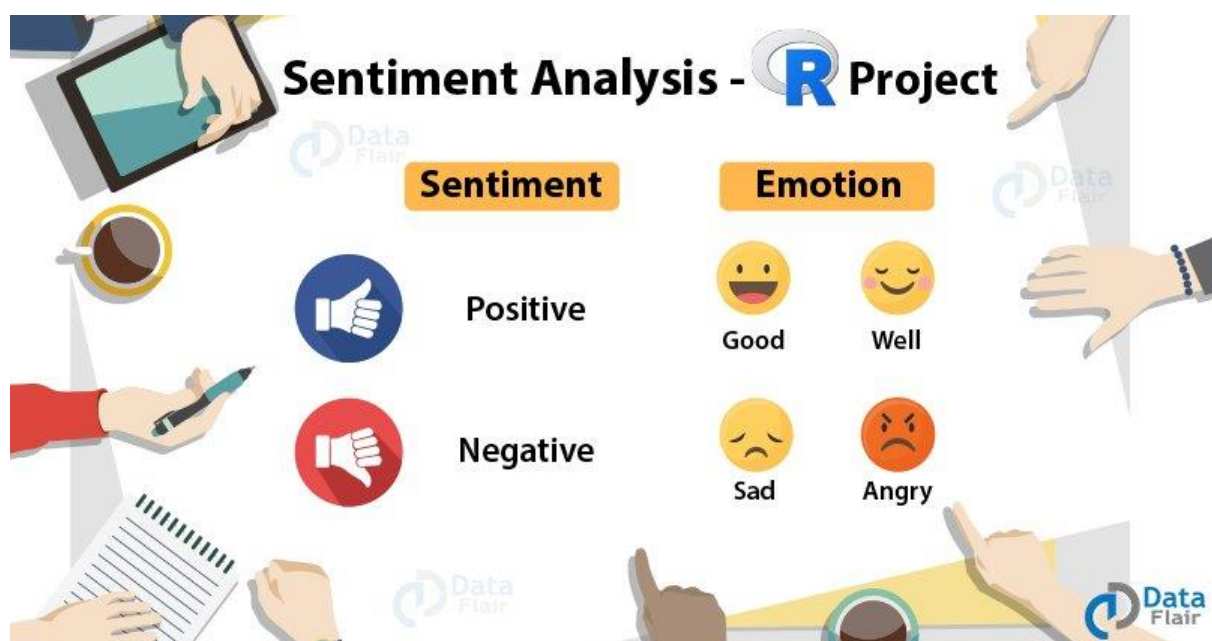
Data Science Project – Sentiment Analysis Project in R

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.).



If you are not aware of the topic classification in R, here is the best guide – [R Classification](#).

Developing our Sentiment Analysis Model in R

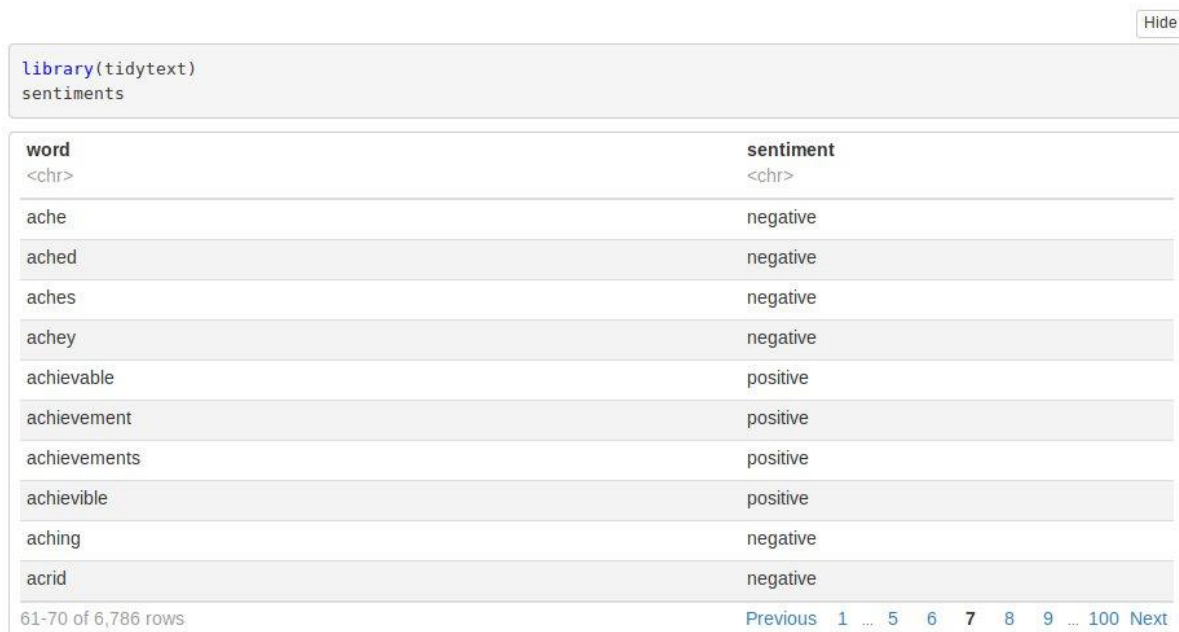
We will carry out sentiment analysis with R in this project. The dataset that we will use will be provided by the R package ‘janeaustenR’.

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:



The screenshot shows the RStudio interface. At the top, the console displays the command `library(tidytext)` and the package `sentiments` is loaded. Below the console, a table preview of the `Sentiments` dataset is shown. The table has two columns: `word` (type `<chr>`) and `sentiment` (type `<chr>`). The preview shows 10 rows of data, with words like 'ache', 'ached', 'aches', 'achey', 'achievable', 'achievement', 'achievements', 'achievable', 'aching', and 'acrid' mapped to either 'negative' or 'positive' sentiments. At the bottom of the table, it indicates '61-70 of 6,786 rows' and provides navigation links: 'Previous', '1', '...', '5', '6', '7', '8', '9', '...', '100', and 'Next'.

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

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:

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

Screenshot:

[Hide](#)

<code>get_sentiments("bing")</code>	
word	sentiment
<chr>	<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(linenumber = row_number(),
         chapter = cumsum(str_detect(text, regex("^chapter [\\divxlc]",
                                                ignore_case = TRUE)))) %>%
  ungroup() %>%
  unnest_tokens(word, text)
```

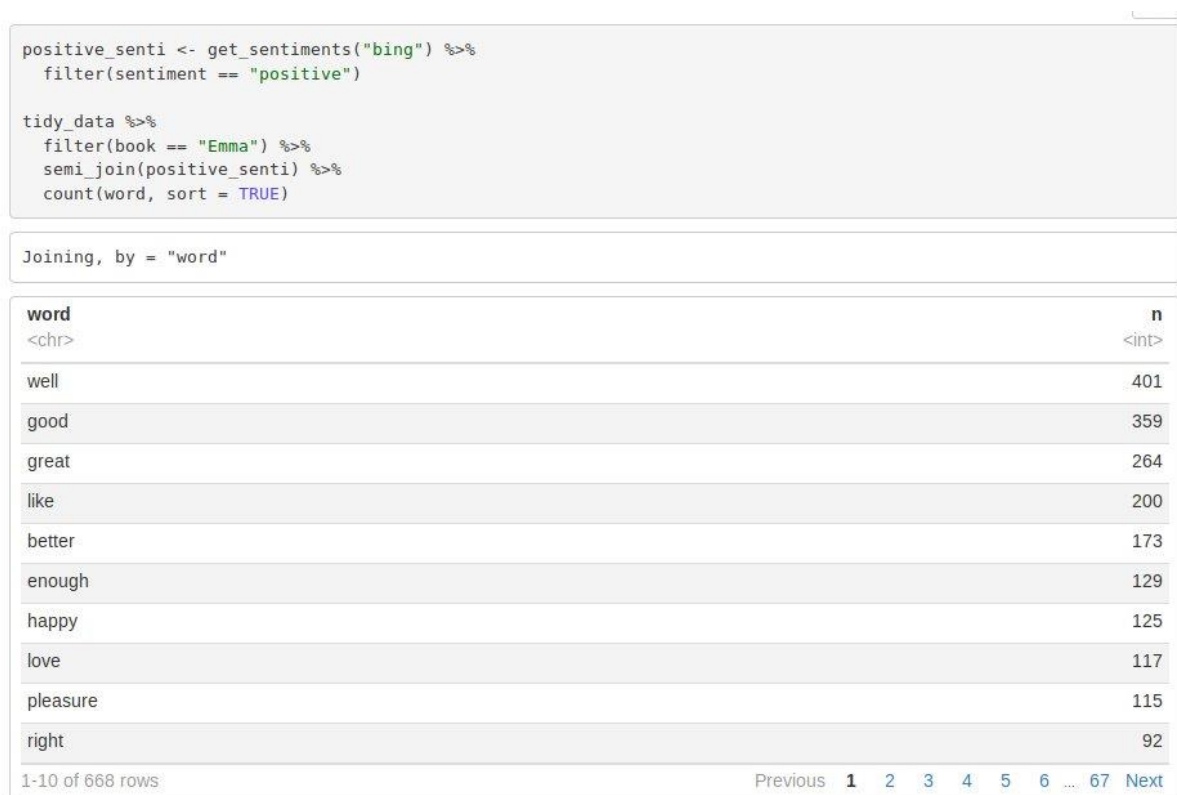
We have performed the tidy operation on our text such that each row contains a single word. We will now make use of the “bing” lexicon to and implement filter() over the words that correspond to joy. We will use the book Sense and Sensibility and derive its words to implement our sentiment analysis model.

Syntax:

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

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

Screenshot:



The screenshot displays the RStudio interface. At the top, a code editor shows the following R code:

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

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

Below the code editor, a message box indicates "Joining, by = 'word'". The main area shows a table with the results of the count operation:

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

At the bottom of the table, it indicates "1-10 of 668 rows". Navigation links for the table are visible: "Previous", "1", "2", "3", "4", "5", "6", "...", "67", and "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 = linenummer %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

Screenshot:

A screenshot of an R console window showing the execution of the same R code as in the Syntax block. The code is: library(tidyr); bing <- get_sentiments("bing"); Emma_sentiment <- tidy_data %>% inner_join(bing) %>% count(book = "Emma" , index = linenummer %/% 80, sentiment) %>% spread(sentiment, n, fill = 0) %>% mutate(sentiment = positive - negative). The output is not visible, and a "Hide" button is present in the top right corner of the console window.

```
library(tidyr)
bing <- get_sentiments("bing")
Emma_sentiment <- tidy_data %>%
  inner_join(bing) %>%
  count(book = "Emma" , index = linenummer %/% 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:

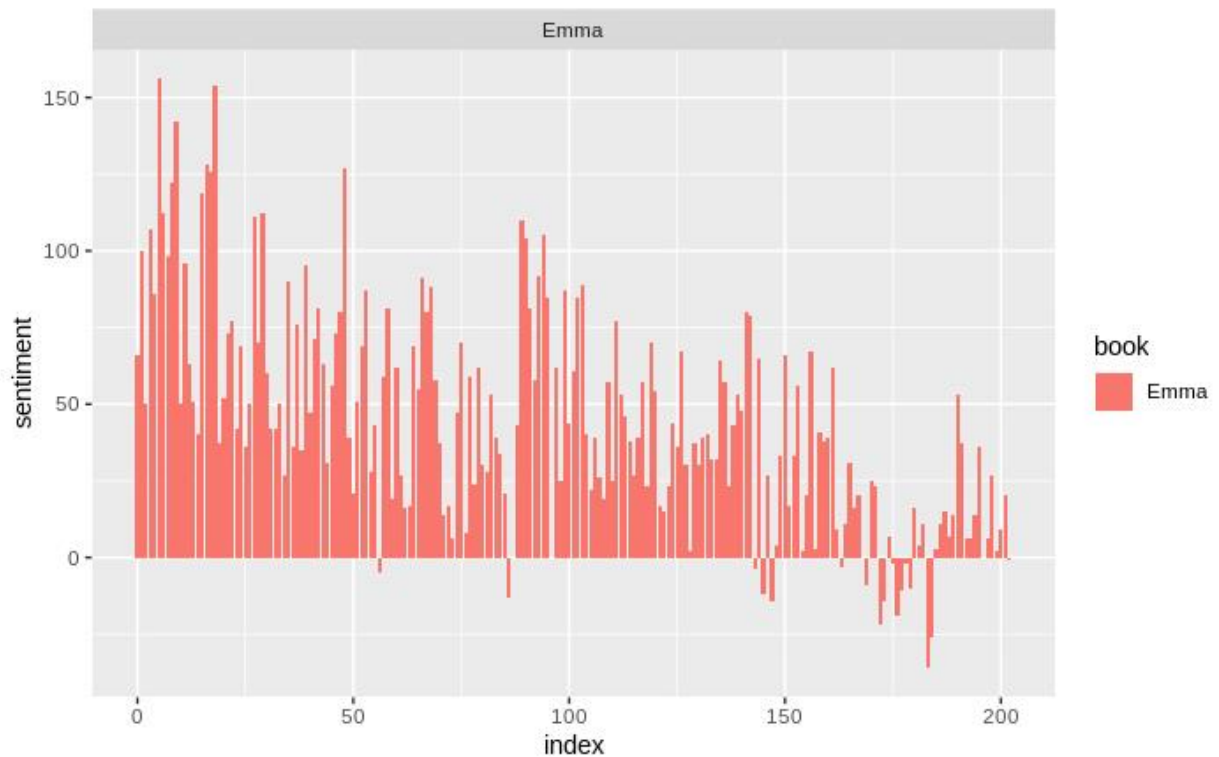
```
library(ggplot2)

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	sentiment	n
<chr>	<chr>	<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.

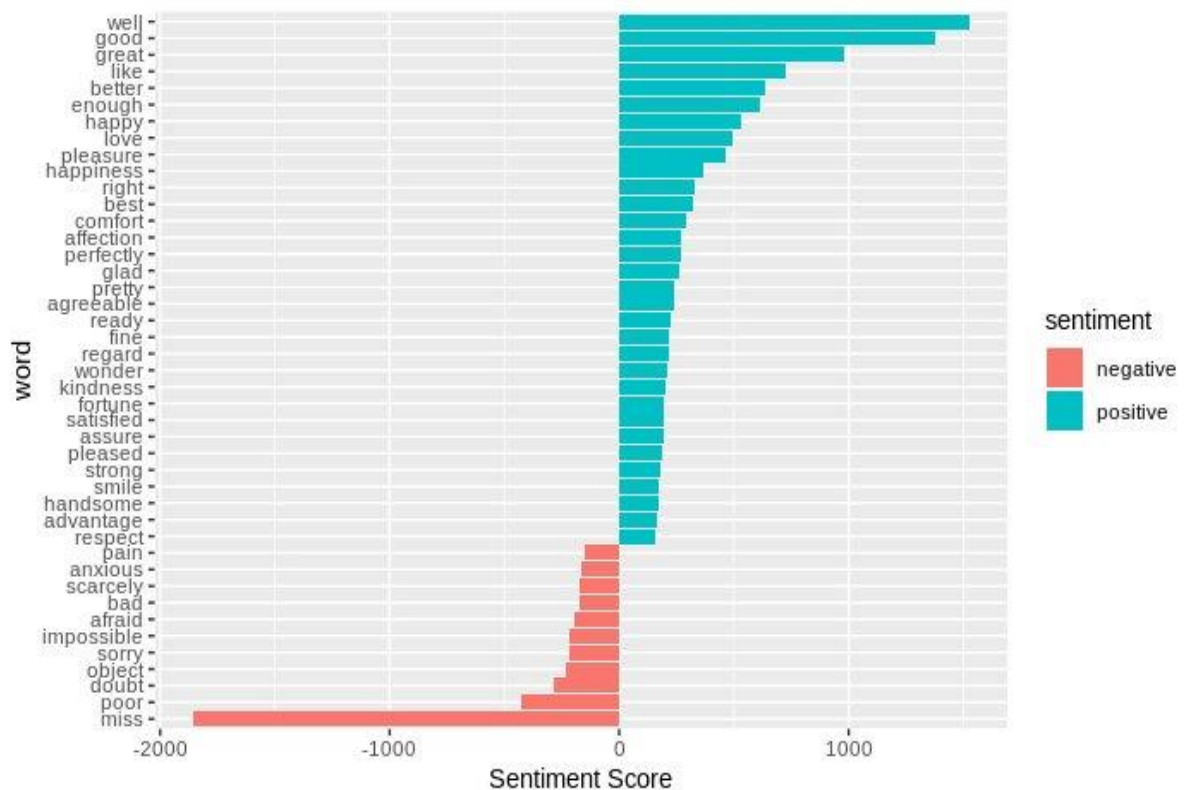
Don't forget to check our latest guide on [data visualization using R](#).

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 `comparison.cloud()` function to plot both negative and positive words in a single wordcloud as follows:

Syntax:

```
library(reshape2)
library(wordcloud)
tidy_data %>%
  inner_join(bing) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("red", "dark green"),
    max.words = 100)
```

Screenshot:

Joining, by = "word"



Summary

Tags: [machine learning Sentiment analysis](#) [R project](#) [Sentiment analysis in R](#) [Sentiment analysis project](#)