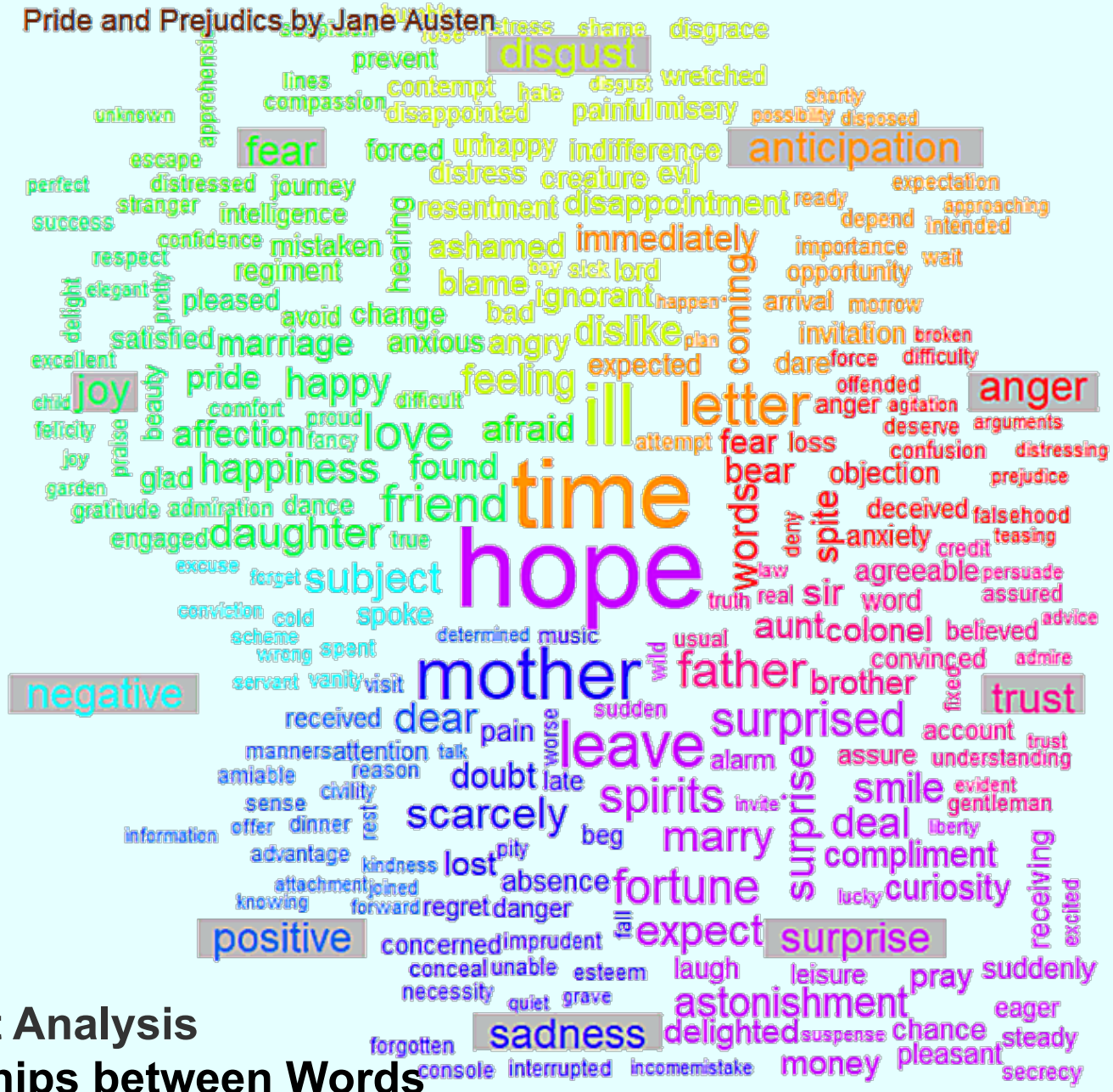
Text Mining Part 2

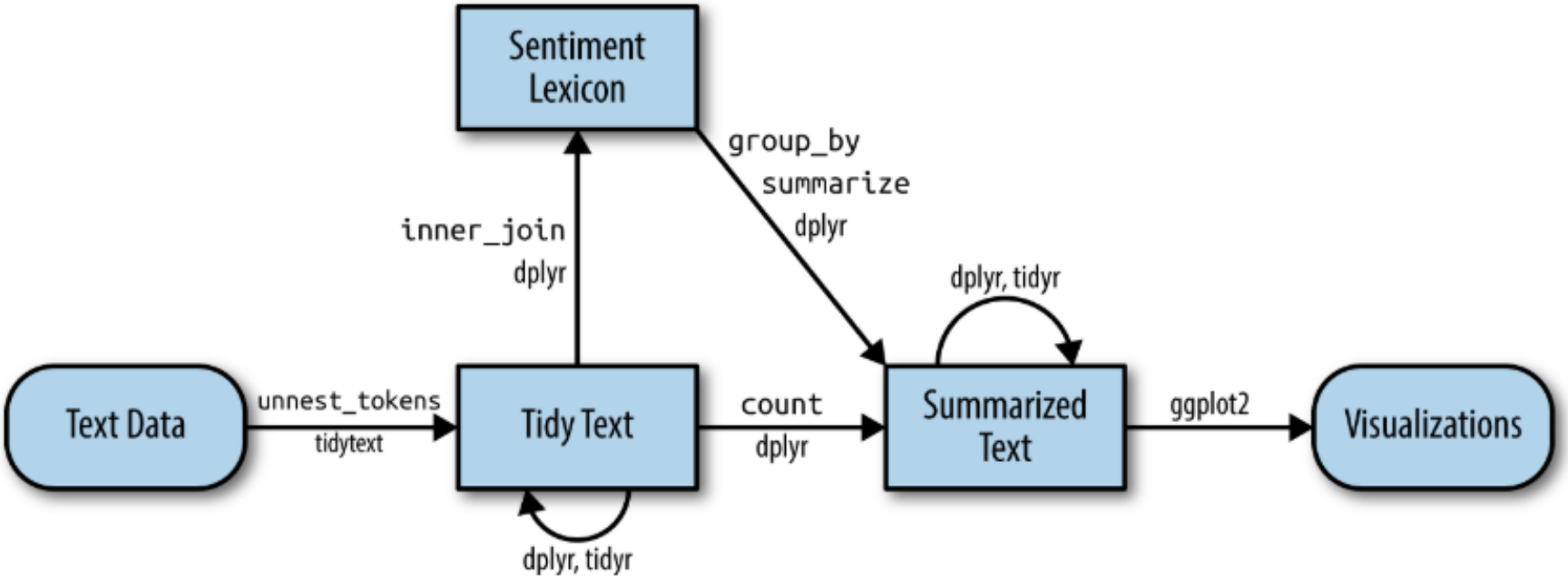


3. Sentiment Analysis
4. Relationships between Words

3. Sentiment Analysis

Sentiment analysis is the process of extracting an author's emotional intent from text.

T. Kwartler, Text Mining in Practice with R (John Wiley, 2017) p.85.



A flowchart of a typical text analysis that uses tidytext for sentiment analysis.
[Figure] J. Silge and D. Robinson, Text Mining with R: A Tidy Approach (O'Reilly, 2017) p.13.

https://www.datacamp.com/community/tutorials/sentiment-analysis-R

(1) The Sentiments Dataset

The tidytext package contains four sentiment lexicons in the sentiments dataset.

sentiments {tidytext}

Four lexicons for sentiment analysis are combined here in a tidy data frame.

```
> library(tidytext)
> sentiments
# A tibble: 27,314 x 4
   word        sentiment lexicon score
   <chr>       <chr>     <chr>   <int>
 1 abacus      trust     nrc        NA
 2 abandon     fear      nrc        NA
 3 abandon     negative  nrc        NA
 4 abandon     sadness   nrc        NA
 5 abandoned   anger     nrc        NA
 6 abandoned   fear      nrc        NA
 7 abandoned   negative  nrc        NA
 8 abandoned   sadness   nrc        NA
 9 abandonment anger     nrc        NA
10 abandonment fear      nrc        NA
# ... with 27,304 more rows
> table(sentiments$lexicon)

   AFINN     bing loughran      nrc
    2476     6788     4149    13901
```

lexicon: a word dictionary supported by the tidytext package labeling sentiments often produced by crowdsourcing
- nrc has multiple labels per word
- bing is binary (positive/negative)
- afinn has -5 to 5 scales

get_sentiments(lexicon=c("afinn","bing","nrc","loughran")) {tidytext}
Get specific sentiment lexicons in a tidy format, with one row per word, in a form that can be joined with a one-word-per-row dataset.

The **nrc** lexicon:

```
> nrc_sword <- get_sentiments("nrc")
> nrc_sword
# A tibble: 13,901 x 2
   word        sentiment
   <chr>       <chr>
 1 abacus      trust
 2 abandon     fear
 3 abandon     negative
 4 abandon     sadness
 5 abandoned   anger
 6 abandoned   fear
 7 abandoned   negative
 8 abandoned   sadness
 9 abandonment anger
10 abandonment fear
# ... with 13,891 more rows
> table(nrc_sword$sentiment)

      anger anticipation      disgust         fear
       1247          839         1058         1476
        joy     negative     positive      sadness
        689         3324         2312         1191
   surprise        trust
        534         1231
```

(2) Text Data: Jane Eyre by Charlotte Brontë

```
> library(gutenbergr)
> jane <- gutenberg_download(1260)
> head(jane,2)
# A tibble: 2 x 2
  gutenberg_id text
         <int> <chr>
1         1260 JANE EYRE
2         1260 AN AUTOBIOGRAPHY
```

Project Gutenberg offers over 58,000 free eBooks.

http://www.gutenberg.org/

Let's split a column into tokens and remove stop_words.

```
> library(dplyr); library(tidytext)
> tm_jane <- jane %>%
+   unnest_tokens(word, text) %>%
+   anti_join(stop_words)
Joining, by = "word"
> tm_jane %>% top_n(5)
Selecting by word
# A tibble: 5 x 2
  gutenberg_id word
         <int> <chr>
1         1260 zembla
2         1260 zone
3         1260 zenith
4         1260 zig
5         1260 zornes
```

Replace ("days", "eyes", "ladies") to ("day", "eye", "lady")

```
tm_jane$word <- gsub("days","day",tm_jane$word)
tm_jane$word <- gsub("eyes","eye",tm_jane$word)
tm_jane$word <- gsub("ladies","lady",tm_jane$word)
```

(3) Frequency Distribution of Sentiments according to Sentiment Categories

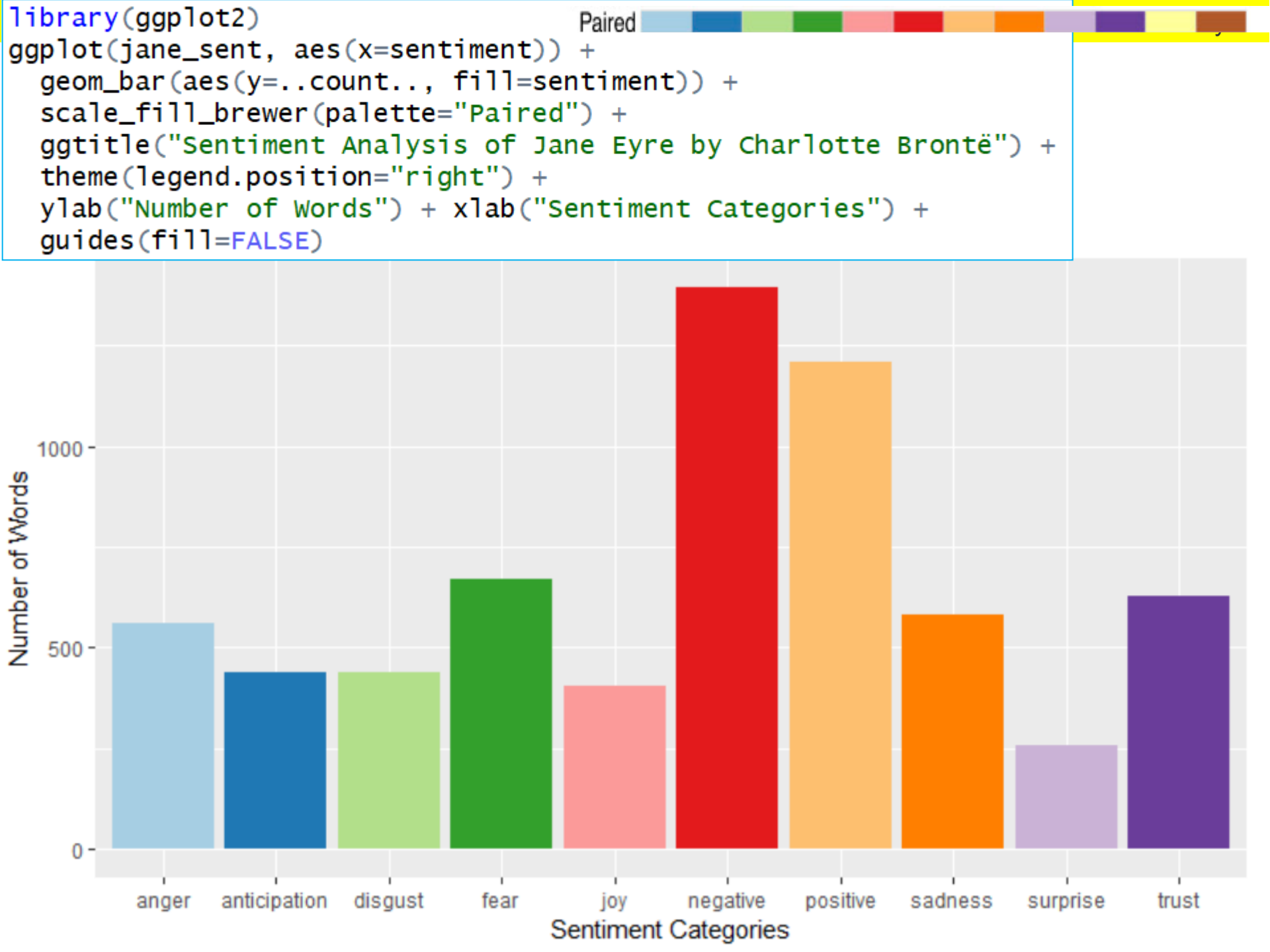
inner_join(x, y, by, ...) {dplyr}
Join two tbls together. Return all rows from x where there are matching values in y, and all columns from x and y.

Join tibbles of Jane Eyre and nrc:

```
> jane_sent <- tm_jane %>%
+   inner_join(nrc_sword,by="word") %>%
+   count(word, sentiment, sort=TRUE)
> jane_sent
# A tibble: 6,588 x 3
   word  sentiment        n
   <chr> <chr>        <int>
 1 sir   positive       315
 2 sir   trust          315
 3 time  anticipation   244
 4 john  disgust        184
 5 john  negative       184
 6 love  joy            151
 7 love  positive       151
 8 found joy            126
 9 found positive       126
10 found trust          126
# ... with 6,578 more rows
```

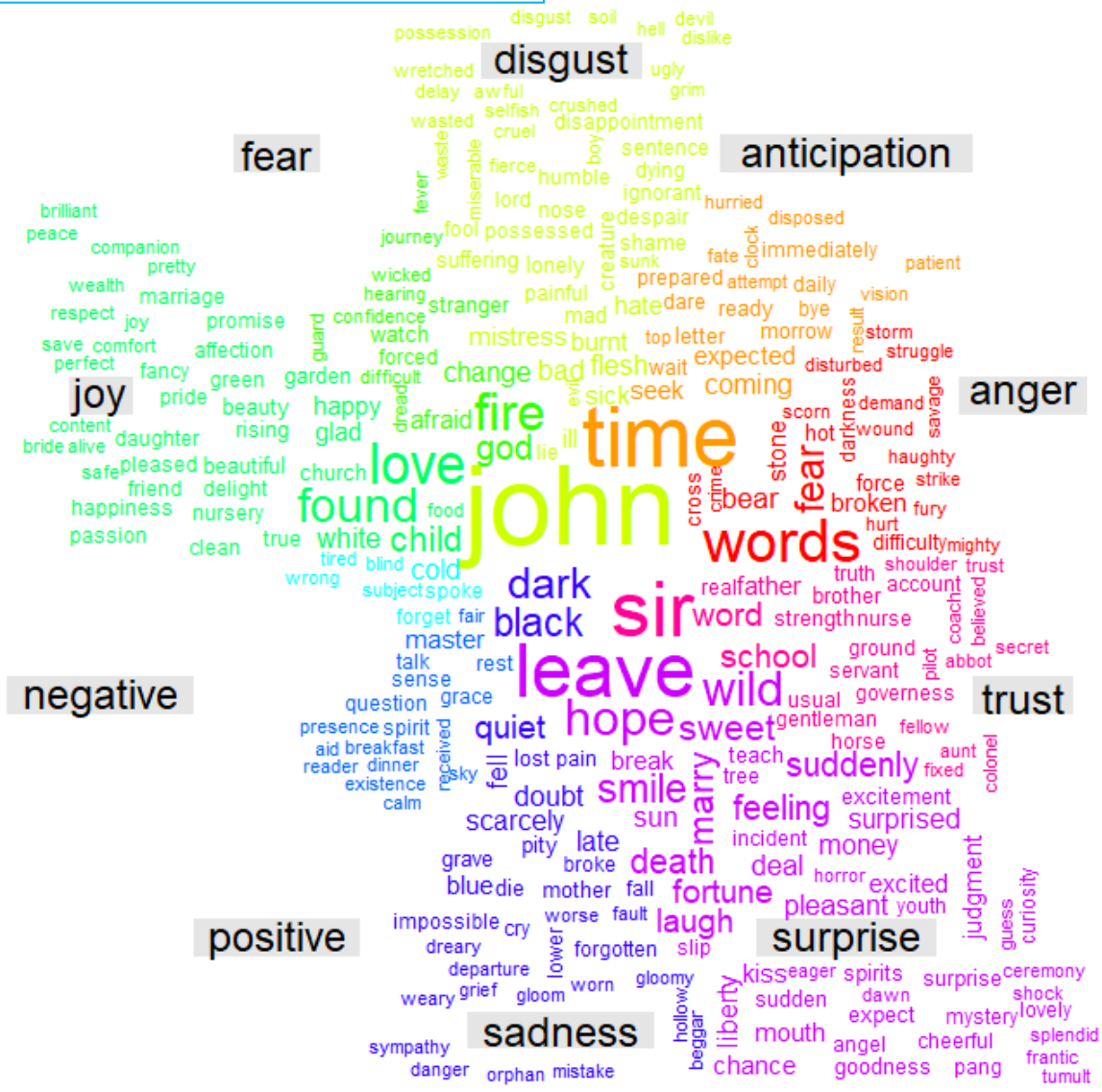
Lexicon approach assumes the total sentiment of a document is a simple summation of individual sentiment scores for each word in the text.
This approach cannot handle complex expressions like sarcasm or 'no good'.

```
library(ggplot2)
ggplot(jane_sent, aes(x=sentiment)) +
  geom_bar(aes(y=..count.., fill=sentiment)) +
  scale_fill_brewer(palette="Paired") +
  ggtitle("Sentiment Analysis of Jane Eyre by Charlotte Brontë") +
  theme(legend.position="right") +
  ylab("Number of Words") + xlab("Sentiment Categories") +
  guides(fill=FALSE)
```



(4) Comparison Word Cloud


```
library(reshape2); library(wordcloud)
jane_sent %>%
  acast(word~sentiment,value.var="n",fill=0) %>%
  comparison.cloud(colors=rainbow(10),title.size=1.5)
```



(5) Positive and Negative Words

The **positive** and **negative** words in **nrc** lexicon:

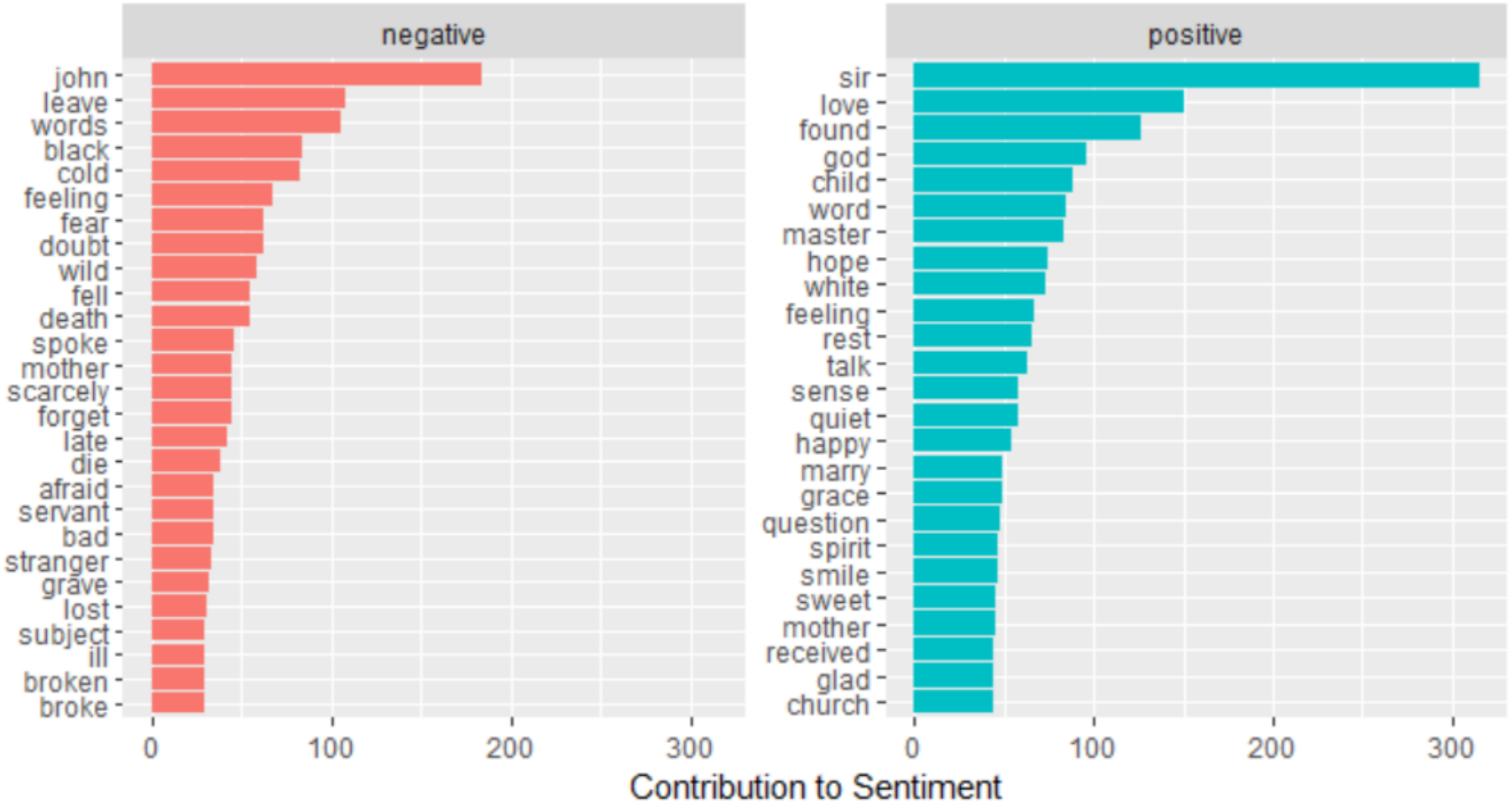
```
posneg <- get_sentiments("nrc") %>%
  filter(sentiment %in% c("positive","negative"))
```

Now let's filter() the data frame with the text from the **Jane Eyre** and then use **inner_join()** to perform the sentiment analysis.

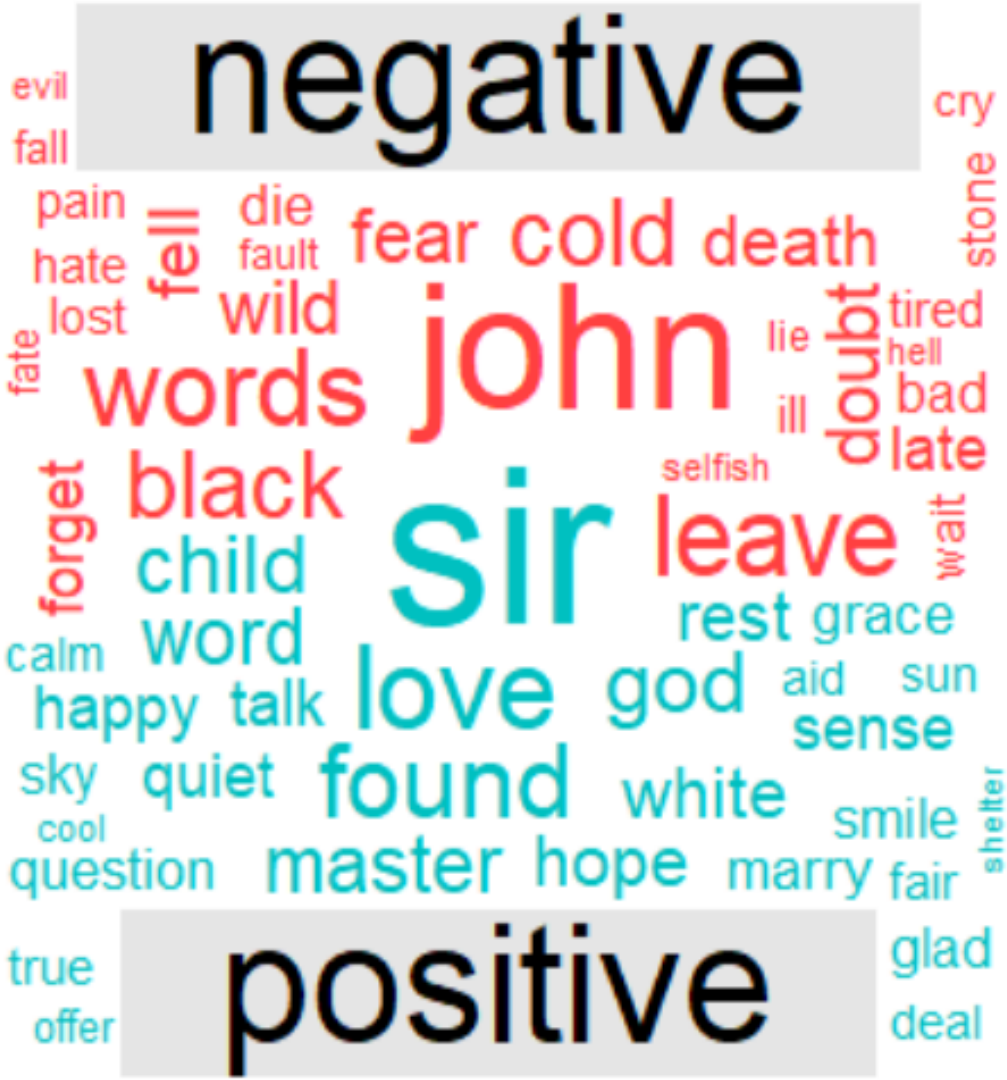
```
> jane_posneg <- tm_jane %>%
+   inner_join(posneg,by="word") %>%
+   count(word, sentiment, sort=TRUE)
> jane_posneg
# A tibble: 2,604 x 3
   word  sentiment     n
   <chr> <chr>     <int>
 1 sir   positive    315
 2 john  negative    184
 3 love  positive    151
 4 found positive    126
 5 leave negative    108
 6 words negative    105
 7 god   positive     96
 8 child positive     88
 9 word  positive     85
10 black negative     84
# ... with 2,594 more rows
```

```
library(ggplot2)
jane_posneg %>% group_by(sentiment) %>%
  top_n(25) %>% 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()
```

Axis **scales** are normally fixed and have the same size and range. They can be made independent by setting scales to free.



```
library(reshape2)
library(wordcloud)
jane_posneg %>%
  acast(word~sentiment,value.var="n",fill=0) %>%
  comparison.cloud(colors=c("#FF4040","#00C0C0"))
```



4. Relationships between Words

Understanding the relationship between words in a corpus:

- What sequences of words are common across our text?
- Given a sequence of words, what word is most likely to follow?
- What words have the strongest relationship with each other?

[Sample Tidy Data] **Alice's Adventures in Wonderland** by Lewis Carroll

```
> library(gutenbergr)
> alice <- gutenberg_download(11)
> alice
# A tibble: 3,339 x 2
   gutenberg_id text
          <int> <chr>
 1           11 ALICE'S ADVENTURES IN WONDE~
 2           11 ""
 3           11 Lewis Carroll
 4           11 ""
 5           11 THE MILLENNIUM FULCRUM EDIT~
 6           11 ""
 7           11 ""
 8           11 ""
 9           11 ""
10           11 CHAPTER I. Down the Rabbit-~
# ... with 3,329 more rows
```

(1) Tokenizing by n-grams

By seeing how often word A is followed by word B, we can then build a model of the relationships between them.
When we set token="ngrams" and n=2 to unnest_tokens(), we are examining pairs of two consecutive words, bigrams:

```
> library(dplyr); library(tidytext)
> tm_alice <- alice %>%
+   unnest_tokens(word,text,token="ngrams",n=2) %>%
+   anti_join(stop_words)
Joining, by = "word"
> tm_alice
# A tibble: 26,693 x 2
   gutenberg_id word
          <int> <chr>
 1           11 alice's adventures
 2           11 adventures in
 3           11 in wonderland
 4           11 wonderland lewis
 5           11 lewis carroll
 6           11 carroll the
 7           11 the millennium
 8           11 millennium fulcrum
 9           11 fulcrum edition
10           11 edition 3.0
# ... with 26,683 more rows
```

(2) Counting and Filtering n-grams

Separate splits a single character column into multiple columns.

```
> library(tidyr)
> bg_alice <- tm_alice %>%
+   separate(word,c("word1","word2"),sep=" ")
> head(bg_alice,3)
# A tibble: 3 x 3
  gutenberg_id word1      word2
         <int> <chr>      <chr>
1           11 alice's    adventures
2           11 adventures in
3           11 in         wonderland
```

Remove when stop-word is included. Count (word1,word2) pairs.

```
> filter_bg <- bg_alice %>%
+   filter(!word1 %in% stop_words$word) %>%
+   filter(!word2 %in% stop_words$word)
> filter_bg %>% top_n(3)
Selecting by word2
# A tibble: 3 x 3
  gutenberg_id word1    word2
         <int> <chr>    <chr>
1           11 apples   yer
2           11 arm      yer
3           11 graceful zigzag
```

```
> # new bigram counts:
> count_bg <- filter_bg %>%
+   count(word1,word2, sort=TRUE)
> count_bg %>% top_n(3)
Selecting by n
# A tibble: 3 x 3
  word1 word2       n
  <chr> <chr>   <int>
1 mock  turtle     54
2 march hare       31
3 white rabbit     22
```

(3) Creating graph object

Visualization shows the relationship among words simultaneously. Words are arranged into a network or "graph."

https://igraph.org/r/doc/

```
> library(igraph)
> graph_bg <- count_bg %>%
+   filter(n > 2) %>%
+   graph_from_data_frame()
> graph_bg
IGRAPH 14fd4e6 DN-- 77 51 --
+ attr: name (v/c), n (e/n)
+ edges from 14fd4e6 (vertex names):
 [1] mock      ->turtle    march     ->hare
 [3] white     ->rabbit    poor      ->alice
 [5] alice     ->replied   alice     ->looked
 [7] beautiful->soup       cried     ->alice
 [9] soo       ->oop       golden    ->key
[11] cheshire  ->cat       evening   ->beautiful
[13] kid       ->gloves    offended  ->tone
[15] play      ->croquet   trembling->voice
+ ... omitted several edges
```

(4) Visualizing a Network of Bigrams

We can convert an igraph object into a ggraph.

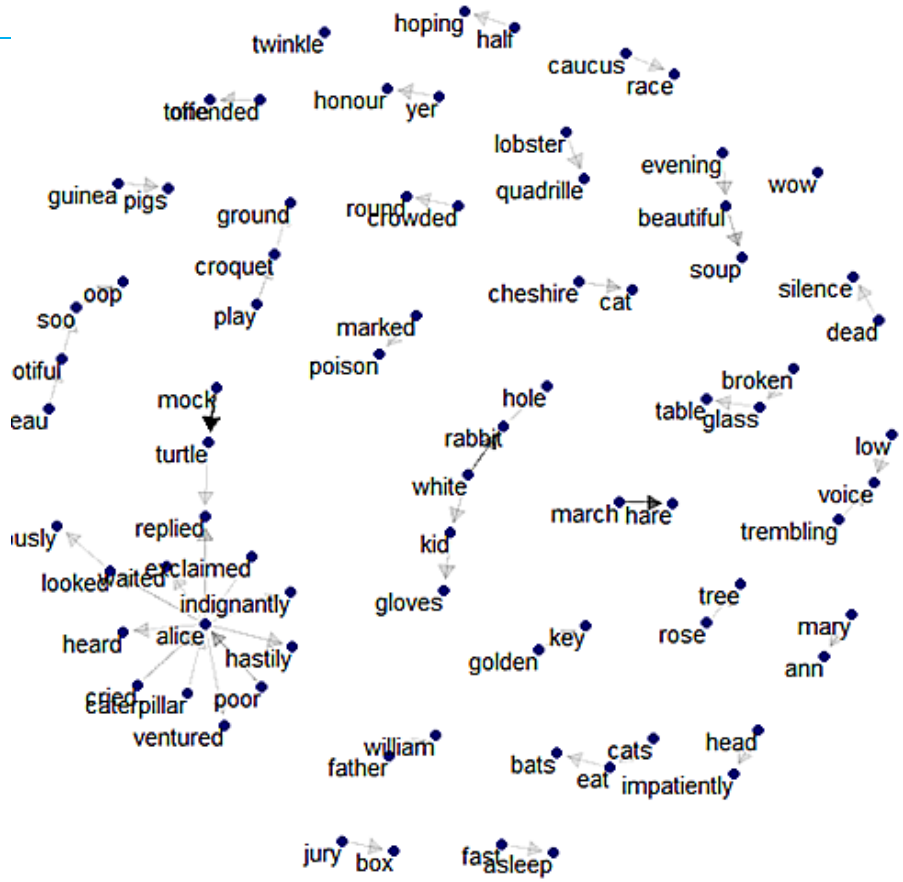
```
library(ggraph)
ar <- grid::arrow(type="closed", length=unit(.1,"inches"))
ggraph(graph_bg,layout='fr') +
  geom_edge_link(aes(edge_alpha=n), show.legend=FALSE,
                 arrow=ar, end_cap=circle(.07,"inches")) +
  geom_node_point(color="navyblue", size=2) +
  geom_node_text(aes(label=name), vjust=1, hjust=1) +
  theme_void()
```

ggraph: https://tinyurl.com/y3n8xu5c

```
1  arrow(angle = 30, length = unit(0.25, "inches"),
2        ends = "last", type = "open")
```

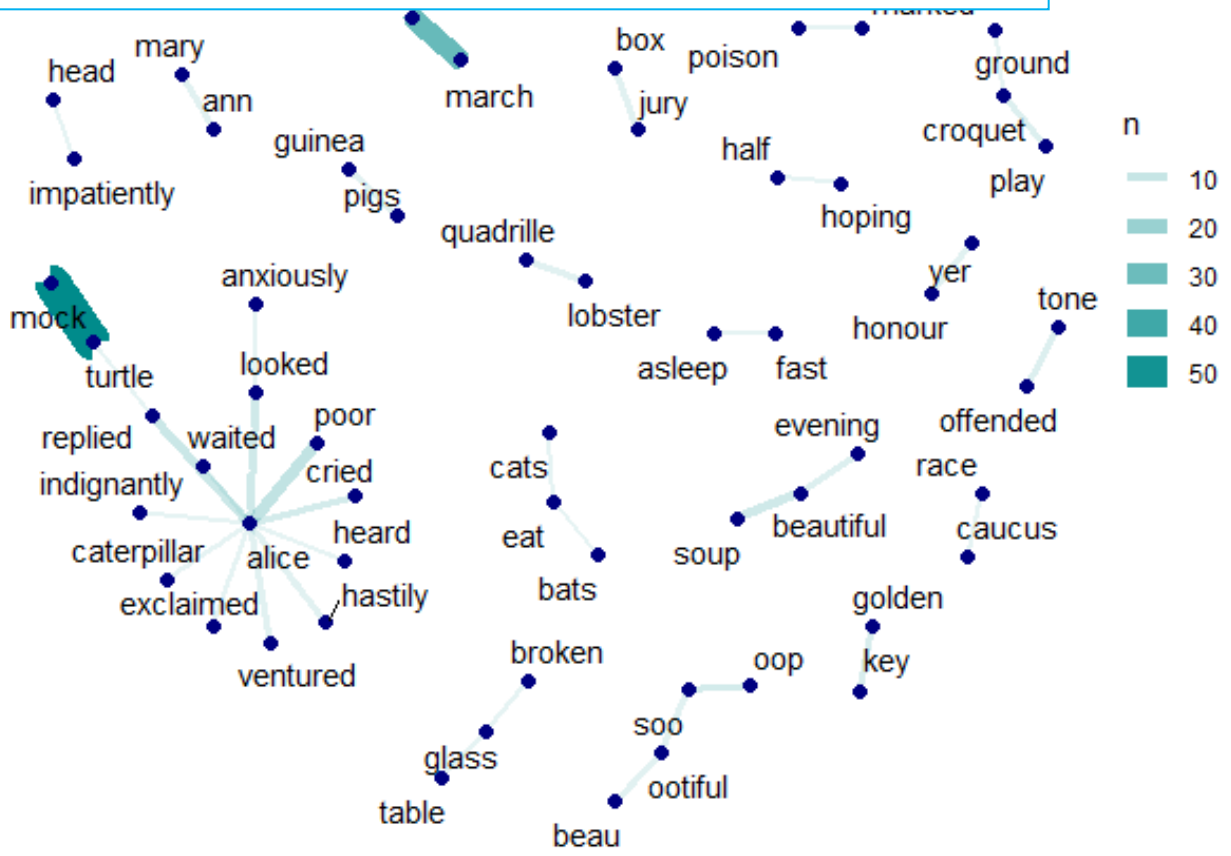
Arguments

`angle`	The angle of the arrow head in degrees (smaller numbers produce narrower, pointier arrows). Essentially describes the width of the arrow head.
`length`	A unit specifying the length of the arrow head (from tip to base).
`ends`	One of `"last"`, `"first"`, or `"both"`, indicating which ends of the line to draw arrow heads.
`type`	One of `"open"` or `"closed"` indicating whether the arrow head should be a closed triangle.



Networks of the co-occurring words:

```
count_bg %>%
  filter(n > 2) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha=n,edge_width=n),edge_colour="cyan4") +
  geom_node_point(color="navyblue", size = 2) +
  geom_node_text(aes(label = name), repel = TRUE,
                 point.padding = unit(0.2, "lines")) +
  theme_void()
```



(5) Sentiment-Associated Words

https://uc-r.github.io/word_relationships

The sentiment analysis simply counts the appearance of positive or negative words, according to a specified lexicon (i.e., AFINN, bing, nrc).
This approach scores the sentiments of words merely on their presence rather than on context.
[Example] The words "happy" and "like" will be counted as positive, even in a sentence like "I am not happy and I do not like it!"

AFINN lexicon gives a numeric **sentiment score** for each word.

```
> library(tidytext)
> AFINN <- get_sentiments("afinn")
> head(AFINN)
# A tibble: 6 x 2
  word       score
  <chr>      <int>
1 abandon       -2
2 abandoned     -2
3 abandons      -2
4 abducted      -2
5 abduction     -2
6 abductions    -2
```

We can assess the most frequent words that have a sentiment score and were preceded by “not”.

```
> nots <- tm_alice %>%
+   separate(word, c("word1","word2"), sep=" ") %>%
+   filter(word1 == "not") %>%
+   inner_join(AFINN, by = c(word2 = "word")) %>%
+   count(word2, score, sort = TRUE)
> nots
# A tibble: 10 x 3
   word2   score     n
   <chr>   <int> <int>
 1 like        2     7
 2 join        1     3
 3 help        2     2
 4 allow       1     1
 5 cried      -2     1
 6 easy        1     1
 7 escape     -1     1
 8 feeling     1     1
 9 mad        -3     1
10 wish        1     1
```

```
library(ggplot2)
nots %>%
  mutate(contribution=n*score) %>%
  arrange(desc(abs(contribution))) %>%
  ggplot(aes(reorder(word2, contribution), n*score, fill=n*score>0)) +
  geom_bar(stat="identity", show.legend=FALSE) +
  xlab("Words preceded by 'not'") +
  ylab("Sentiment score x Number of occurrances") +
  coord_flip()
```