Introduction to sentiment analysis

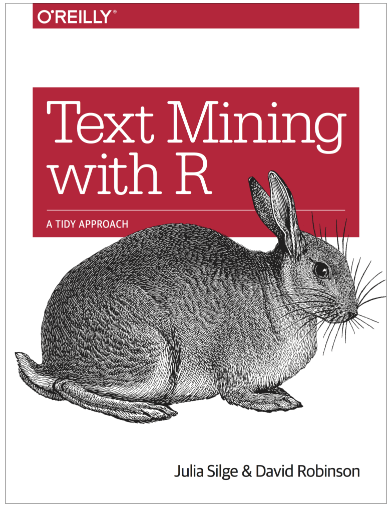
Rei Sanchez-Arias, Ph.D.

Sentiment analysis with the `tidytext` package

Sentiment analysis

Text emotional tone

"Text Mining with R: A Tidy Approach" by Julia Silge and David Robinson

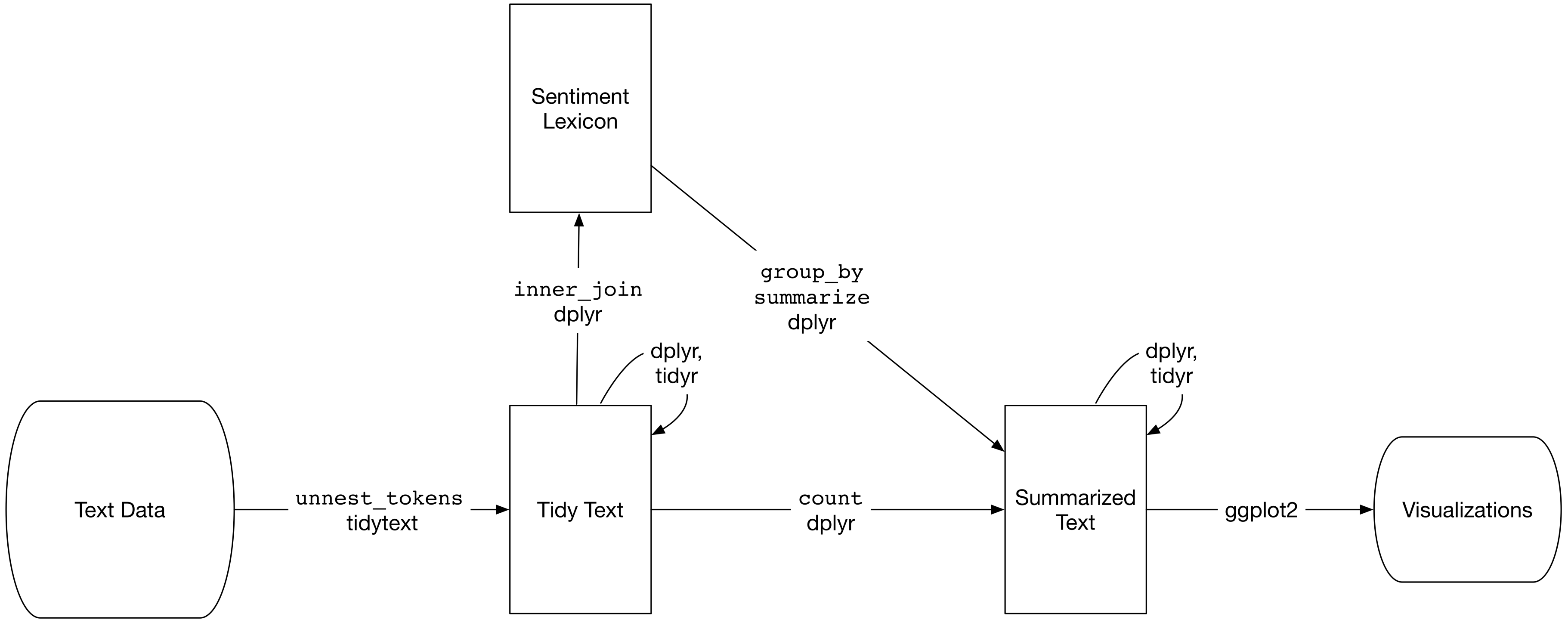


Sentiment analysis can be thought of as the exercise of taking a sentence, paragraph, document, or any piece of natural language, and **determining whether that text's emotional tone is positive, negative or neutral**. One way to analyze the sentiment of a text is to consider the text as a combination of its individual words and the sentiment content of the whole text as the sum of the sentiment content of the individual words.

> Examples and materials in this set of slides are adapted from the book by Silge and Robinson

Emotional intent

When human readers approach a text, we use our understanding of the *emotional intent* of words to infer whether a section of text is positive or negative, or perhaps characterized by some other more nuanced emotion like surprise or disgust.



https://www.tidytextmining.com/sentiment.html

Lexicons

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

The `tidytext` package contains `sentiments` dataset. Three general-purpose lexicons are included:

- `AFINN` from Finn Arup Nielsen: scores a word with a number, which may range from -5 to +5.
- `bing` from Bing Liu and collaborators: scores a word as either positive or negative.
- `nrc` from Saif Mohammad and Peter Turney: categorizes a word under sentiment type categories such as positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

These sentiment lexicons, based on unigrams (i.e. single words) are derived from a single English word and are assigned different scores of positive/negative sentiments.

Lexicons (cont.)

All of this information is tabulated in the `sentiments` dataset, and `tidytext` provides the `get_sentiments()` function to get specific sentiment lexicons with the appropriate measures for each one.

```
sample_n( get_sentiments("afinn") , 5)
```

```
## # A tibble: 5 x 2
##   word          value
##   <chr>         <dbl>
## 1 aggravate        -2
## 2 criminal         -3
## 3 growth            2
## 4 significance      1
## 5 glamorous         3
```

```
sample_n( get_sentiments("nrc") , 5)
```

```
## # A tibble: 5 x 2
##   word      sentiment
##   <chr>     <chr>
## 1 jealousy  anger
## 2 disgraced sadness
## 3 gall      negative
## 4 snag      negative
## 5 athletic  positive
```

```
sample_n( get_sentiments("bing"), 5)
```

```
## # A tibble: 5 x 2
##   word        sentiment
##   <chr>       <chr>
## 1 deprave     negative
## 2 floundering negative
## 3 treacherous negative
## 4 plusses     positive
## 5 bungling    negative
```

More on lexicons

- These lexicons are available under different licenses, so be sure that the license for the lexicon you want to use is appropriate for your project. *You may be asked to agree to a license before downloading data.*
- There are also some **domain-specific sentiment lexicons** available, constructed to be used with text from a specific content area. Not every English word is in the lexicons because many English words are pretty neutral. It is important to keep in mind that these methods do not take into account qualifiers before a word, such as in "no good" or "not true"; a lexicon-based method like this is based on unigrams only.
- One last caveat is that *the size of the chunk of text that we use to add up unigram sentiment scores can have an effect* on an analysis. A text the size of many paragraphs can often have positive and negative sentiment averaged out to about zero, while sentence-sized or paragraph-sized text often works better.

Example: song lyrics example

Billboard top 100

```r
# top 100 Billboard songs in 2015
lyrics_2015 <- read_csv("https://raw.githubusercontent.com/reisanar/datasets/master/BB_top100_
lyrics_2015 <- select(lyrics_2015, -Year, -Source)
```

	Rank	Song	Artist
1	1	uptown funk	mark ronson featuring bruno mars
2	2	thinking out loud	ed sheeran
3	3	see you again	wiz khalifa featuring charlie puth
4	4	trap queen	fetty wap
5	5	sugar	maroon 5
6	6	shut up and dance	walk the moon
7	7	blank space	taylor swift

Previous 1 2 3 4 5 … 15 Next

Sample of lyrics

```
lyrics_2015 %>%
  select(Song, Lyrics) %>%
  top_n(5)
```

```
## # A tibble: 5 x 2
##   Song              Lyrics
##   <chr>             <chr>
## 1 earned it         you make it look like its magic cause i see nobody nobody bu…
## 2 the hills         your man on the road he doin promo you said keep our busines…
## 3 love me like yo… youre the light youre the night youre the color of my blood …
## 4 hotline bling     you used to call me on my you used to you used to you used t…
## 5 house party       youre on the couch blowing up my phone you dont want to come…
```

> **Goal:** study the sentiment of the top 10 songs lyrics

Sentiment in lyrics for the top 10

```r
lyrics_2015 %>%
  filter(Rank %in% 1:10)
```

```
## # A tibble: 10 x 4
##     Rank Song         Artist              Lyrics
##    <dbl> <chr>        <chr>               <chr>
##  1     1 uptown funk  mark ronson featur… this hit that ice cold michelle pfeiff…
##  2     2 thinking o…  ed sheeran          when your legs dont work like they use…
##  3     3 see you ag…  wiz khalifa featur… its been a long day without you my fri…
##  4     4 trap queen   fetty wap           im like hey wassup hello seen yo prett…
##  5     5 sugar        maroon 5            im hurting baby im broken down i need …
##  6     6 shut up an…  walk the moon       oh dont you dare look back just keep y…
##  7     7 blank space  taylor swift        nice to meet you where you been i coul…
##  8     8 watch me     silento             now watch me whip kill it now watch me…
##  9     9 earned it    the weeknd          you make it look like its magic cause …
## 10    10 the hills    the weeknd          your man on the road he doin promo you…
```

Tokenization

First: convert the text to the **tidy format** using `unnest_tokens()`. Notice that we choose the name `word` for the output column from `unnest_tokens()`. This is a convenient choice because the **sentiment lexicons** and **stop-words datasets** have columns named `word`; so performing inner-joins and anti-joins is thus easier.

```
lyrics_2015 %>%
  filter(Rank %in% 1:10) %>%
  unnest_tokens(word, Lyrics)
```

	Rank	Song	Artist	word
1	1	uptown funk	mark ronson featuring bruno mars	this
2	1	uptown funk	mark ronson featuring bruno mars	hit
3	1	uptown funk	mark ronson featuring bruno mars	that
4	1	uptown funk	mark ronson featuring bruno mars	ice

Previous 1 2 3 4 5 … 1104 Next

Remove stop words

```r
lyrics_2015 %>%
  filter(Rank %in% 1:10) %>%
  unnest_tokens(word, Lyrics, token = "words") %>%
  filter(!word %in% stop_words$word, str_detect(word, "[a-z]"))
```

	Rank	Song	Artist	word
1	1	uptown funk	mark ronson featuring bruno mars	hit
2	1	uptown funk	mark ronson featuring bruno mars	ice
3	1	uptown funk	mark ronson featuring bruno mars	cold
4	1	uptown funk	mark ronson featuring bruno mars	michelle
5	1	uptown funk	mark ronson featuring bruno mars	pfeiffer
6	1	uptown funk	mark ronson featuring bruno mars	white

Previous 1 2 3 4 5 … 297 Next

Exploration

First let us store our tidy text data set in a new object called `bb_top_10`

```
# BB top 10 songs lyrics in 2015 (no stop-wo
bb_top_10 <- lyrics_2015 %>%
  filter(Rank %in% 1:10) %>%
  unnest_tokens(word,
                Lyrics,
                token = "words") %>%
  filter(!word %in% stop_words$word,
         str_detect(word, "[a-z]"))
```

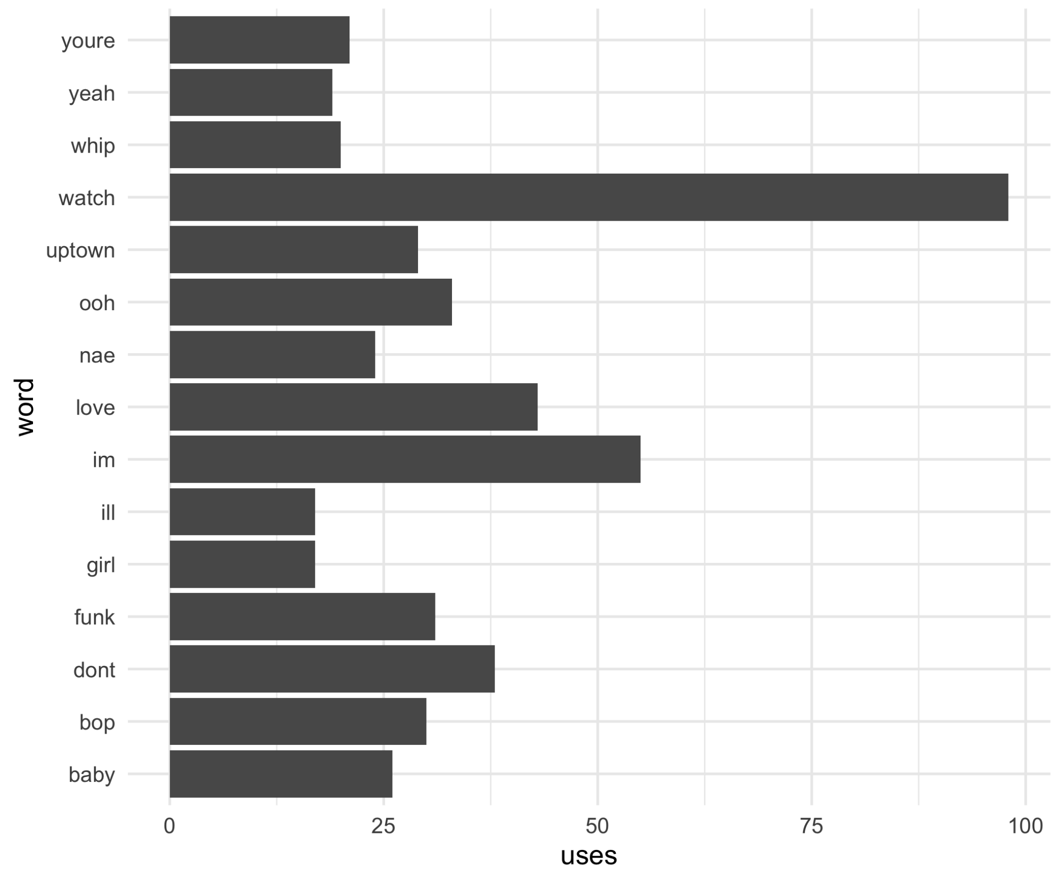
Check the most used words:

```
# check some of the most frequent words
bb_top_10 %>%
  group_by(word) %>%
  summarise(uses = n()) %>%
  arrange(desc(uses)) %>%
  head(10)
```

```
## # A tibble: 10 x 2
##      word     uses
##      <chr>   <int>
##    1 watch      98
##    2 im         55
##    3 love       43
##    4 dont       38
##    5 ooh        33
##    6 funk       31
##    7 bop        30
##    8 uptown     29
##    9 baby       26
##   10 nae        24
```

Most frequent words: visualization

```r
bb_top_10 %>%
  group_by(word) %>%
  summarise(uses = n()) %>%
  arrange(desc(uses)) %>%
  slice(1:15) %>%
  ggplot() +
  geom_bar(aes(x = word, y = uses),
           stat = "identity") +
  coord_flip() +
  theme_minimal()
```



get_sentiments("bing")

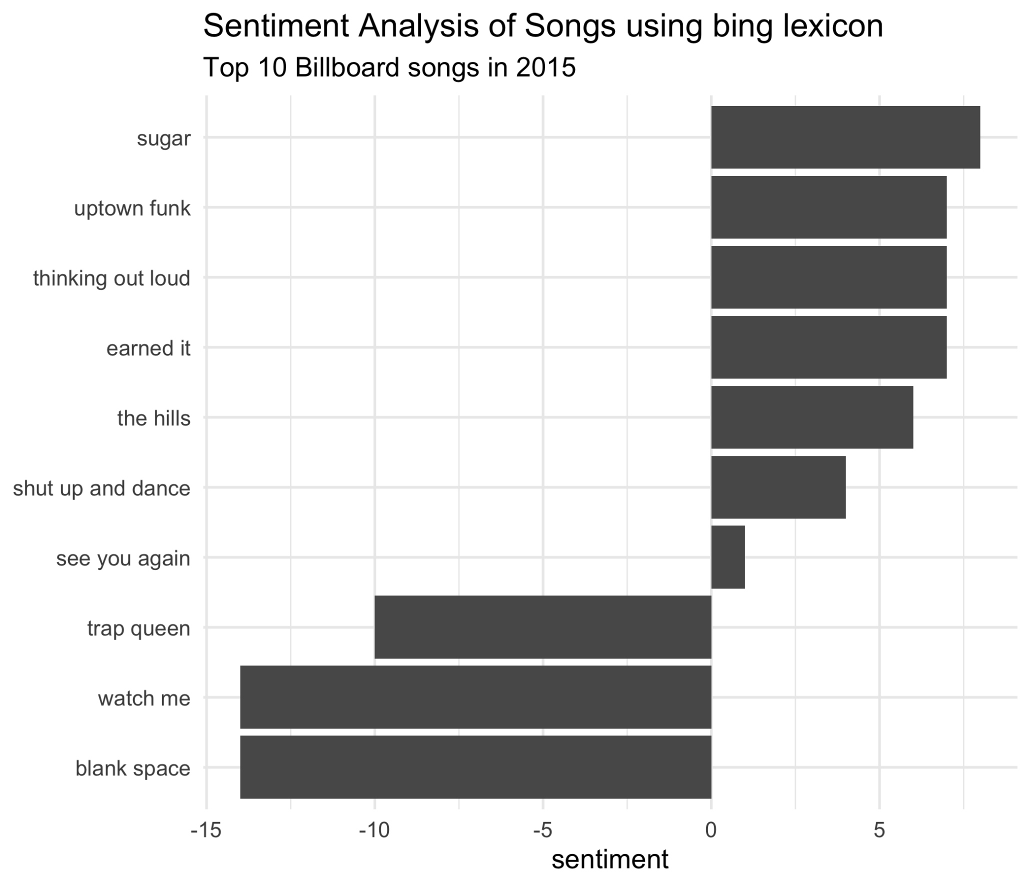
Much as removing stop words is an anti-join operation, performing sentiment analysis is an **inner join operation**.

```
bb_top_10 %>%
  inner_join(get_sentiments("bing")) %>%
  count(Song, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

```
## # A tibble: 10 x 4
##    Song                negative positive
##    <chr>                  <dbl>    <dbl>
##  1 blank space               34       20
##  2 earned it                 10       17
##  3 see you again              4        5
##  4 shut up and dance          1        5
##  5 sugar                     11       19
##  6 the hills                  3        9
##  7 thinking out loud         11       18
##  8 trap queen                18        8
##  9 uptown funk               16       23
## 10 watch me                  14        0
```

Sentiment analysis visualization

```
bb_top_10 %>%
  inner_join(get_sentiments("bing")) %>%
  count(Song, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative) %>
  ggplot() +
  geom_bar(aes(x = reorder(Song, sentiment),
               y = sentiment),
           stat = "identity") +
  coord_flip() +
  labs(x = "",
       title = "Sentiment Analysis of Songs 
       subtitle = "Top 10 Billboard songs ir
  theme_minimal()
```



get_sentiments("afinn")

What if we want to find out how the sentiment of the song varies as the song progresses? Let us use now the `afinn` lexicon

```
bb_top_10 %>%
  inner_join(
    get_sentiments("afinn")
    )
```

	Rank	Song	Artist	word	value
1	1	uptown funk	mark ronson featuring bruno mars	straight	1
2	1	uptown funk	mark ronson featuring bruno mars	masterpieces	4
3	1	uptown funk	mark ronson featuring bruno mars	kiss	2
4	1	uptown funk	mark ronson featuring bruno mars	damn	-4
5	1	uptown funk	mark ronson featuring bruno mars	damn	-4
6	1	uptown funk	mark ronson featuring bruno mars	damn	-4

Previous 1 2 3 4 5 … 41 Next

Keep track of words used

```
# add an index column to keep track of the words used
bb_top_10 %>%
  group_by(Song) %>%
  mutate(index = row_number() ) %>%
  inner_join(get_sentiments("afinn"))
```

	Rank	Song	Artist	word	index	value
1	1	uptown funk	mark ronson featuring bruno mars	straight	11	1
2	1	uptown funk	mark ronson featuring bruno mars	masterpieces	12	4
3	1	uptown funk	mark ronson featuring bruno mars	kiss	20	2
4	1	uptown funk	mark ronson featuring bruno mars	damn	25	-4
5	1	uptown funk	mark ronson featuring bruno mars	damn	32	-4
6	1	uptown funk	mark ronson featuring bruno mars	damn	39	-4

Previous 1 2 3 4 5 … 41 Next

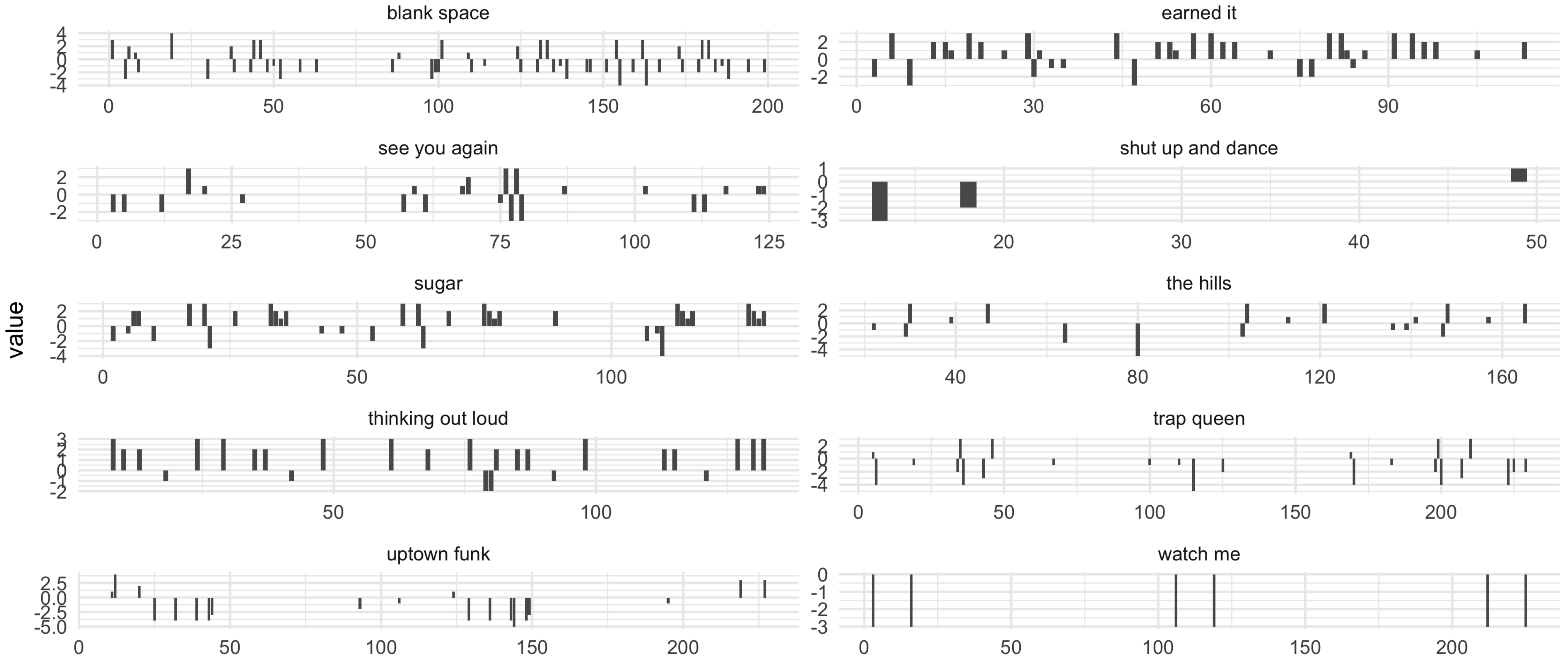
Visualizing words used in the song

```r
bb_top_10 %>%
  group_by(Song) %>%
  mutate(index = row_number() ) %>%
  inner_join(get_sentiments("afinn")) %>%
  ggplot() +
  geom_col(aes(x = index, y = value)) +
  facet_wrap(~Song, scales = "free", ncol = 2) +
  labs(title = "Sentiment Analysis of Songs using AFINN lexicon",
       subtitle = "Top 10 Billboard songs in 2015",
       x = "") +
  theme_minimal()
```

Notice that several songs in this top 10 list, contain many words that are not part of the lexicon, and therefore were not scored

Sentiment Analysis of Songs using AFINN lexicon

Top 10 Billboard songs in 2015



Example: using the `nrc` lexicon

Focus on one artist

Goal: perform sentiment analysis using the `nrc` lexicon for songs by Taylor Swift

```
lyrics_2015 %>%
  filter(Artist == "taylor swift")
```

```
## # A tibble: 4 x 4
##    Rank Song         Artist     Lyrics
##   <dbl> <chr>        <chr>      <chr>
## 1     7 blank space taylor sw… nice to meet you where you been i could show you…
## 2    18 shake it o… taylor sw… i stay up too late got nothing in my brain thats…
## 3    29 style       taylor sw… midnight you come and pick me up no headlights l…
## 4    57 wildest dr… taylor sw… he said lets get out of this town drive out of t…
```

unnest_tokens()

First, let us convert the text to the tidy format using `unnest_tokens()`. Notice that we choose the name `word` for the output column from `unnest_tokens()`. This is a convenient choice because the sentiment lexicons and stop word datasets have columns named `word`; performing inner joins and anti-joins is thus easier.

```
lyrics_2015 %>%
  filter(Artist == "taylor swift") %>%
  unnest_tokens(word, Lyrics)
```

	Rank	Song	Artist	word
1	7	blank space	taylor swift	nice
2	7	blank space	taylor swift	to
3	7	blank space	taylor swift	meet

Previous 1 2 3 4 5 ... 597 Next

anti_join(stop_words)

```r
# taylor swift songs (no stop-words)
swift <- lyrics_2015 %>%
  filter(Artist == "taylor swift") %>%
  unnest_tokens(word, Lyrics) %>%
  anti_join(stop_words)
```

Check the most used words:

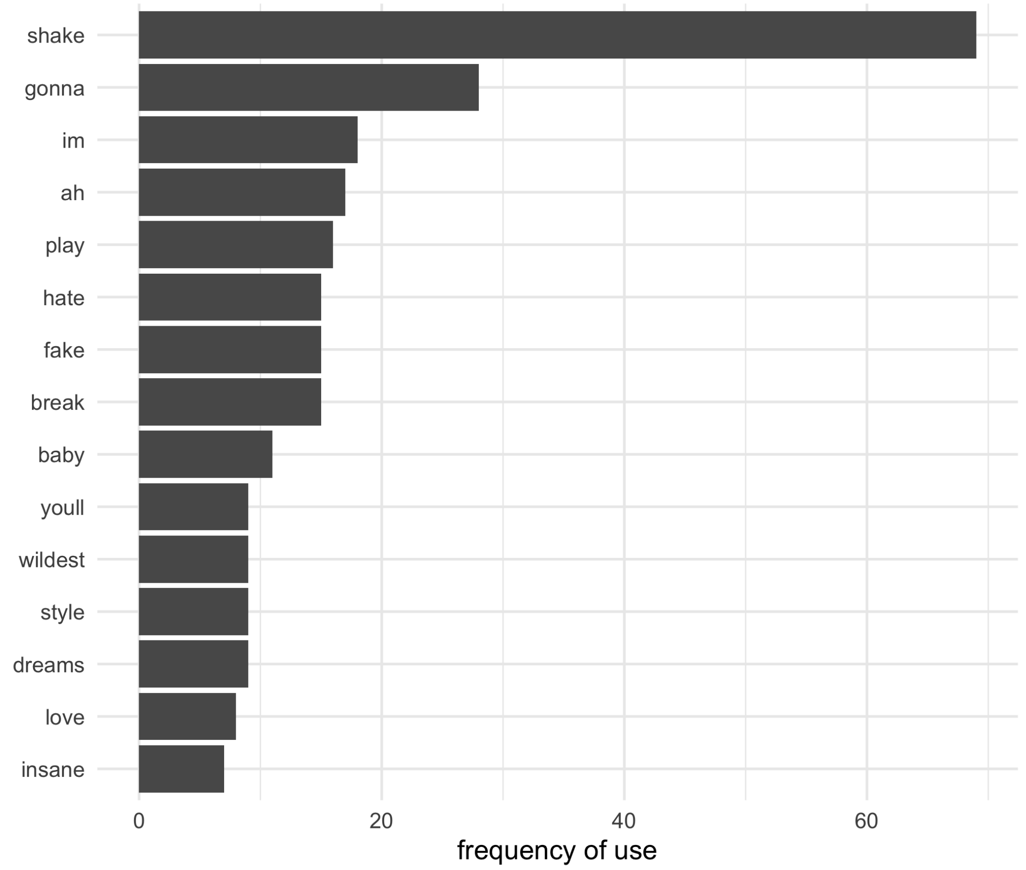
```r
# Check the most used words
swift %>%
  group_by(word) %>%
  summarise(uses = n()) %>%
  arrange(desc(uses))
```

	word	uses
1	shake	69
2	gonna	28
3	im	18
4	ah	17
5	play	16
6	break	15
7	fake	15
8	hate	15
9	baby	11

Previous 1 2 3 4 5 … 29 Next

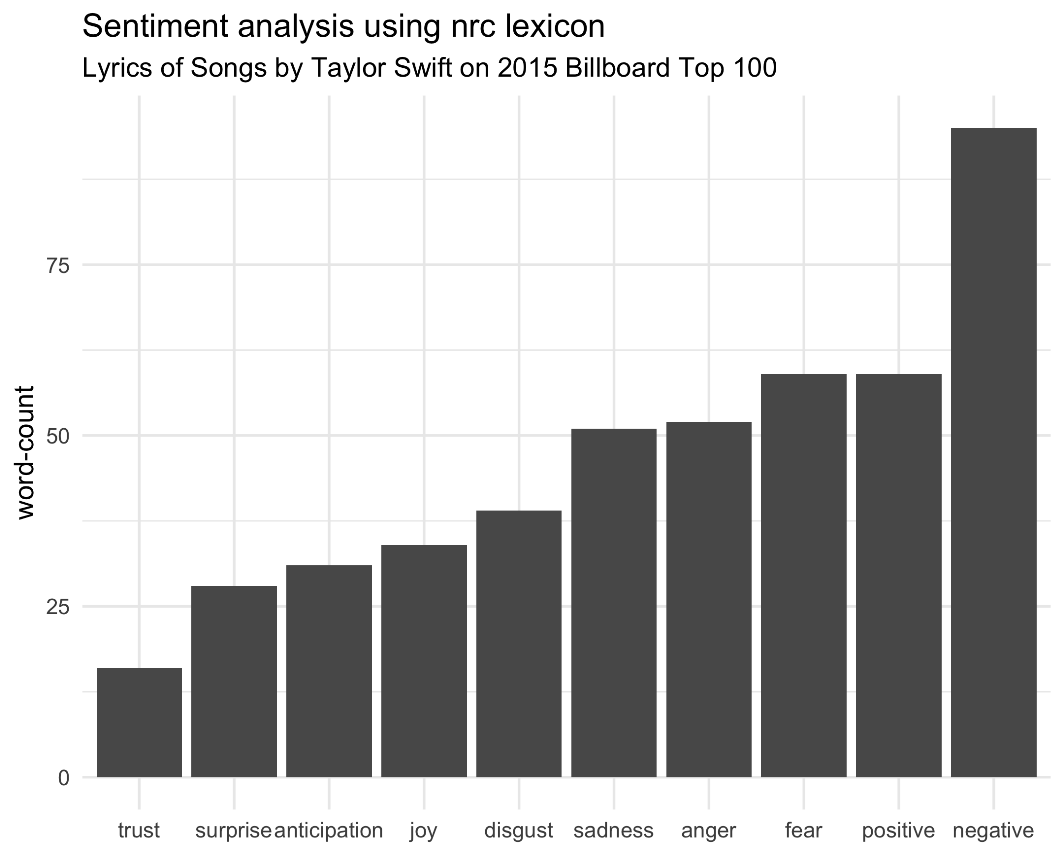
Visualization: common words in songs

```r
swift %>%
  group_by(word) %>%
  summarise(uses = n()) %>%
  arrange(desc(uses)) %>%
  slice(1:15) %>%
  ggplot() +
  geom_bar(
    aes(x = reorder(word, uses),
        y = uses),
    stat = "identity") +
  coord_flip() +
  labs(y = "frequency of use",
       x = "") +
  theme_minimal()
```



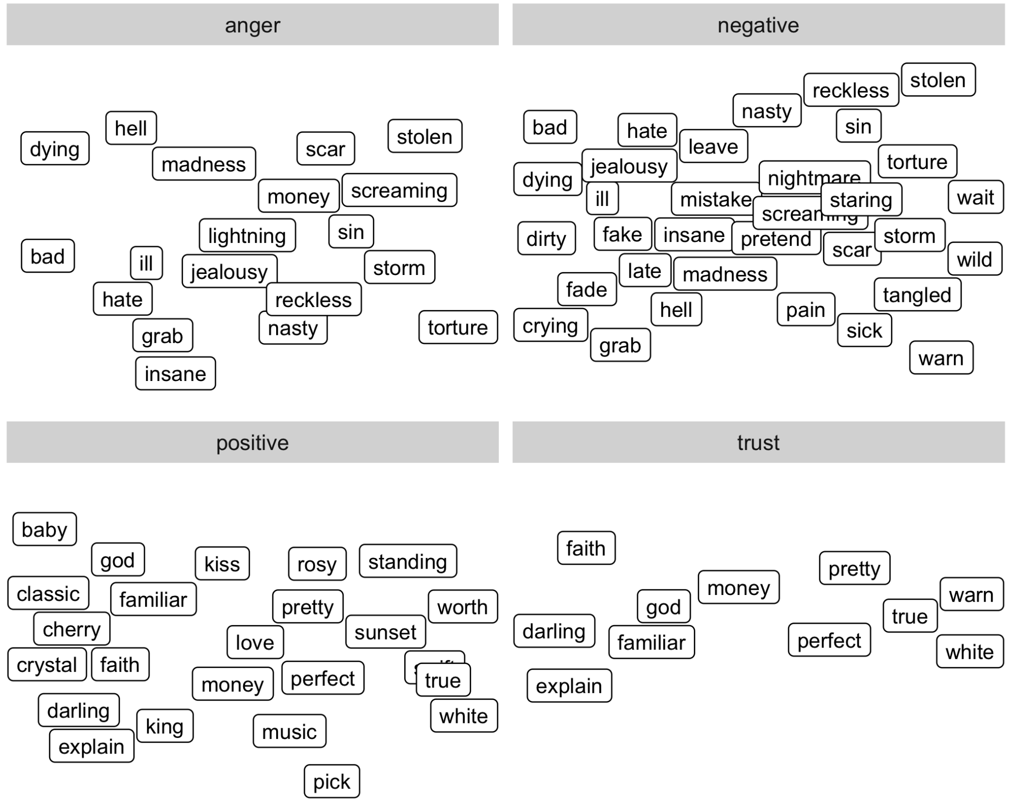
Sentiment analysis with `nrc` lexicon

```
swift %>%
  inner_join(get_sentiments("nrc")) %>%
  group_by(sentiment) %>%
  count() %>%
  ggplot() +
  geom_bar(
    aes(x = reorder(sentiment, n),
        y = n),
    stat = "identity") +
  labs(title = "Sentiment analysis using nrc
       subtitle = "Lyrics of Songs by Taylor
       x = "",
       y = "word-count") +
  theme_minimal()
```



Words in different sentiments

```r
swift_words <- swift %>%
  inner_join(get_sentiments("nrc")) %>%
  group_by(sentiment, word) %>%
  count(mycount = n()) %>%
  distinct() %>%
  filter(sentiment %in%
           c("negative", "anger",
             "positive", "trust"))
# plot
ggplot(data = swift_words, aes(label = word)) +
  ggrepel::geom_label_repel(
    aes(x = word, y = rnorm(nrow(swift_words)),
        label = word),
    direction = "both", box.padding = 0.04,
    segment.color = "transparent", size = 3) +
  facet_wrap(~sentiment, ncol = 2)+
  labs(x = "", y = "") +
  theme(axis.text.y = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks = element_blank(),
        panel.grid = element_blank(),
        panel.background = element_blank()
```


anger
dying
hell
madness
scar
stolen
money
screaming
bad
ill
lightning
sin
jealousy
storm
hate
reckless
grab
nasty
torture
insane
negative
bad
hate
nasty
reckless
stolen
leave
sin
dying
jealousy
nightmare
torture
ill
mistake
screaming
staring
wait
dirty
fake
insane
pretend
scar
storm
wild
fade
late
madness
tangled
crying
hell
pain
sick
grab
warn
positive
baby
god
kiss
rosy
standing
classic
familiar
pretty
worth
cherry
love
sunset
crystal
faith
money
perfect
true
darling
king
music
white
explain
pick
trust
faith
money
pretty
god
warn
darling
familiar
true
perfect
white
explain