Assignment_week_12

Kristoffer Segerstroem 2025-03-21

```
knitr::opts_chunk$set(echo = TRUE,
                      warning = TRUE,
                      message = TRUE)

library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# For text mining:
library(pdftools)  # Used to extract text from PDF files
```

```
## Using poppler version 0.86.1
```

```
library(tidytext)  # Facilitates text analysis by working with words in a 'tidy' format
library(textdata)  # Contains various sentiment dictionaries
library(ggwordcloud)  # Used to create word clouds
```

Get the Game of Thrones text:

```
got_path <- "data/got.pdf"
got_text <- pdf_text(got_path)  # Extracts text from the PDF file as a vector of strings (one per page)
```

Some wrangling:

```
got_df <- data.frame(got_text) %>%
  mutate(text_full = str_split(got_text, pattern = '\\n')) %>%  # Splits the text by line breaks
  unnest(text_full) %>%  # 'Unnests' the listed text so each line becomes a row in the dataframe
  mutate(text_full = str_trim(text_full))  # Removes leading and trailing spaces from each line
```

Get the tokens (individual words) in tidy format

```
got_tokens <- got_df %>%
  unnest_tokens(word, text_full)  # Splits the text into individual words (tokens), so each row contains
```

```
got_tokens
```

```
## # A tibble: 297,815 x 2
##    got_text                                                          word
##    <chr>                                                             <chr>
##  1 "                A GAME OF THRONES\n            Book One of A Song o~ a
##  2 "                A GAME OF THRONES\n            Book One of A Song o~ game
##  3 "                A GAME OF THRONES\n            Book One of A Song o~ of
##  4 "                A GAME OF THRONES\n            Book One of A Song o~ thro~
##  5 "                A GAME OF THRONES\n            Book One of A Song o~ book
##  6 "                A GAME OF THRONES\n            Book One of A Song o~ one
##  7 "                A GAME OF THRONES\n            Book One of A Song o~ of
##  8 "                A GAME OF THRONES\n            Book One of A Song o~ a
##  9 "                A GAME OF THRONES\n            Book One of A Song o~ song
## 10 "                A GAME OF THRONES\n            Book One of A Song o~ of
## # i 297,805 more rows
```

Remove stop words:

```
got_stop <- got_tokens %>%
  anti_join(stop_words) %>%  # Removes stop words (commonly used words like 'the', 'and', 'of') that do
  select(-got_text)  # Removes the original text column as it is no longer needed
```

```
## Joining with `by = join_by(word)`
```

Count the words

```
got_stop %>%
  count(word) %>%
  arrange(-n)
```

```
## # A tibble: 11,294 x 2
##    word       n
##    <chr>  <int>
##  1 lord    1341
##  2 ser     1023
##  3 jon      787
##  4 ned      743
##  5 tyrion   591
##  6 eyes     567
##  7 hand     567
##  8 king     542
##  9 father   512
## 10 told     504
## # i 11,284 more rows
```

Word cloud of GoT words

```
got_top100 <- got_stop %>%
  count(word) %>%  # Counts the occurrences of each word
```

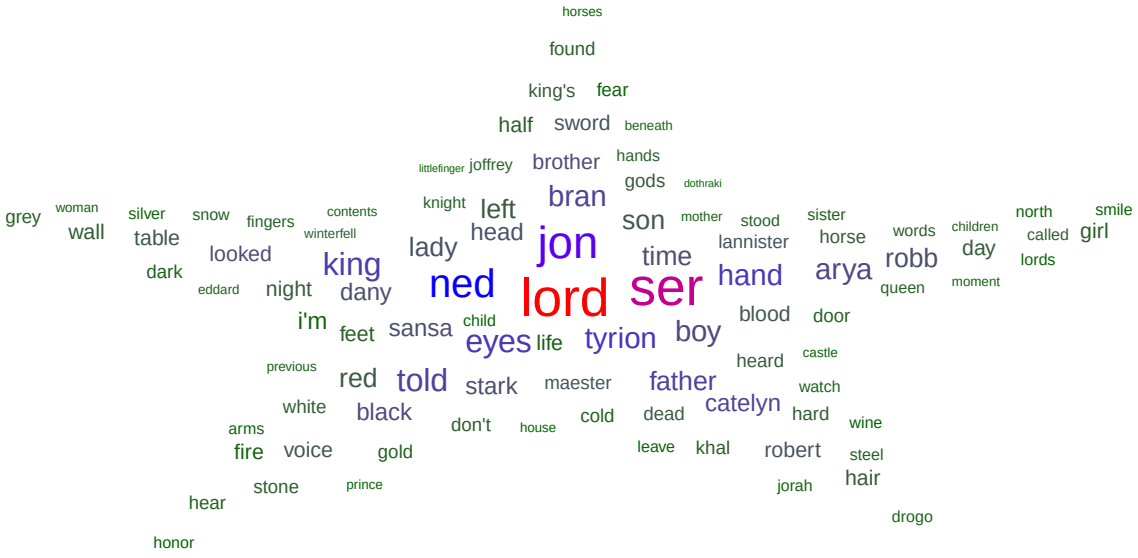
```
  arrange(-n) %>%  # Sorts words by frequency
  head(100)  # Keeps only the 100 most frequent words
```

```
got_cloud <- ggplot(data = got_top100, aes(label = word)) +
  geom_text_wordcloud() +  # Visualizes the words in a word cloud
  theme_minimal()

got_cloud
```

Let's make the word cloud af star

```
ggplot(data = got_top100, aes(label = word, size = n)) +
  geom_text_wordcloud_area(aes(color = n), shape = "star") +
  scale_size_area(max_size = 12) +
  scale_color_gradientn(colors = c("darkgreen","blue","red")) +
  theme_minimal()
```



Sentiment analysis with afinn:

"afinn": Words ranked from -5 (very negative) to +5 (very positive)

```
got_afinn <- got_stop %>%
  inner_join(get_sentiments("afinn"))  # Matches words with sentiment scores from the AFINN lexicon (-5
```

```
## Joining with `by = join_by(word)`
```

The negative words

```
get_sentiments(lexicon = "afinn")
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # i 2,467 more rows
```

```
# Note: may be prompted to download (yes)
```

The positive words

```
library(tidytext)
afinn <- get_sentiments("afinn")
afinn_pos <- afinn %>% filter(value > 0) #finds the sentiments with a greater value than 0
head(afinn_pos)
```

```
## # A tibble: 6 x 2
##   word      value
##   <chr>     <dbl>
## 1 abilities     2
## 2 ability       2
## 3 aboard        1
## 4 absolve       2
## 5 absolved      2
## 6 absolves      2
```

Word association with NRC

```
get_sentiments(lexicon = "nrc")
```

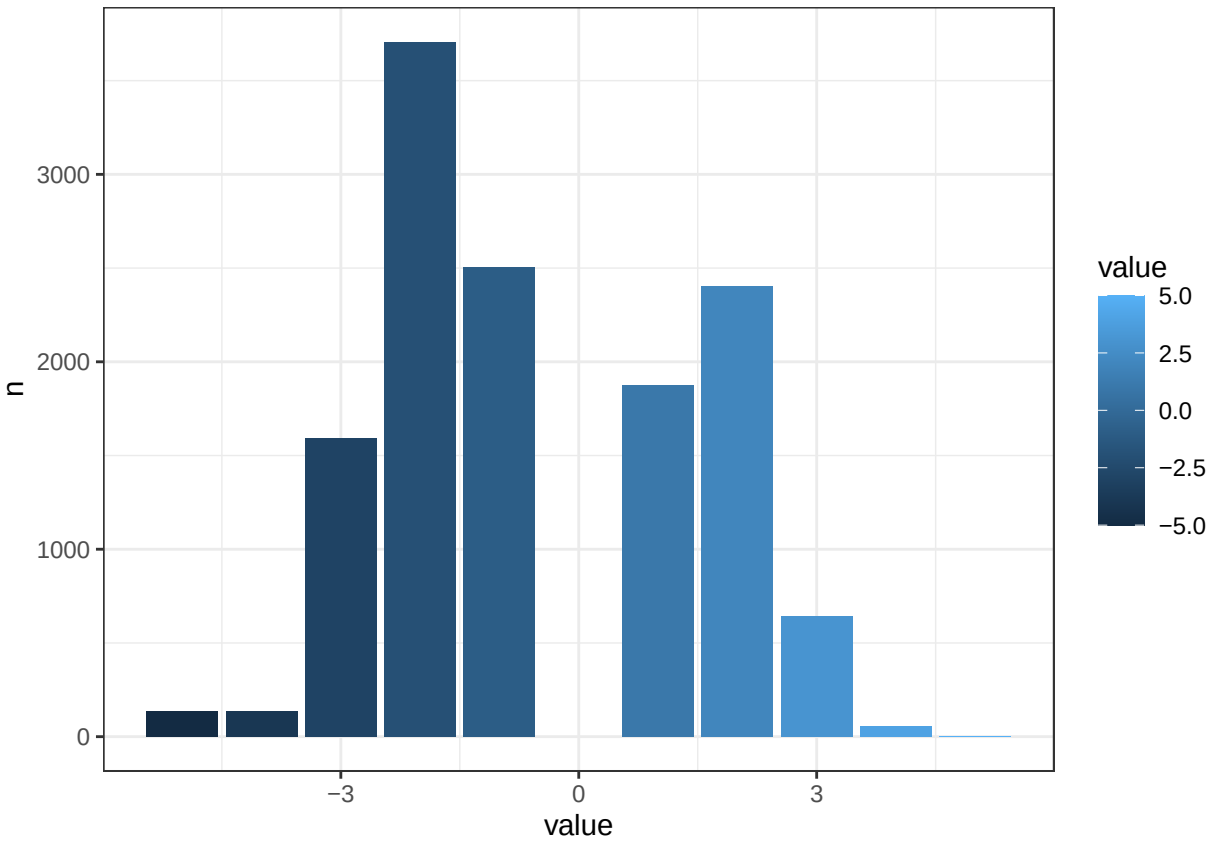
```
## # A tibble: 13,872 x 2
##    word        sentiment
##    <chr>       <chr>
```

```
##  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
## # i 13,862 more rows
```

Plot sentiment scores:

```
got_afinn_hist <- got_afinn %>%
  count(value)  # Counts the number of words for each sentiment score

ggplot(data = got_afinn_hist, aes(x = value, y = n)) +
  geom_col(aes(fill = value)) +  # Visualizes sentiment scores with a bar chart
  theme_bw()
```



#

Investegate which words have a sentiment score of 2 (quite positive)

```
got_afinn2 <- got_afinn%>%
  filter(value == 2)
got_afinn2%>%
  distinct(word)
```

```
## # A tibble: 201 x 1
##    word
##    <chr>
##  1 smile
##  2 fine
##  3 glory
##  4 hope
##  5 smiled
##  6 care
##  7 strength
##  8 peaceful
##  9 honor
## 10 carefully
## # i 191 more rows
```

```
#These commandoes isolates the 2-score words
```

Finding the unique 2-score words

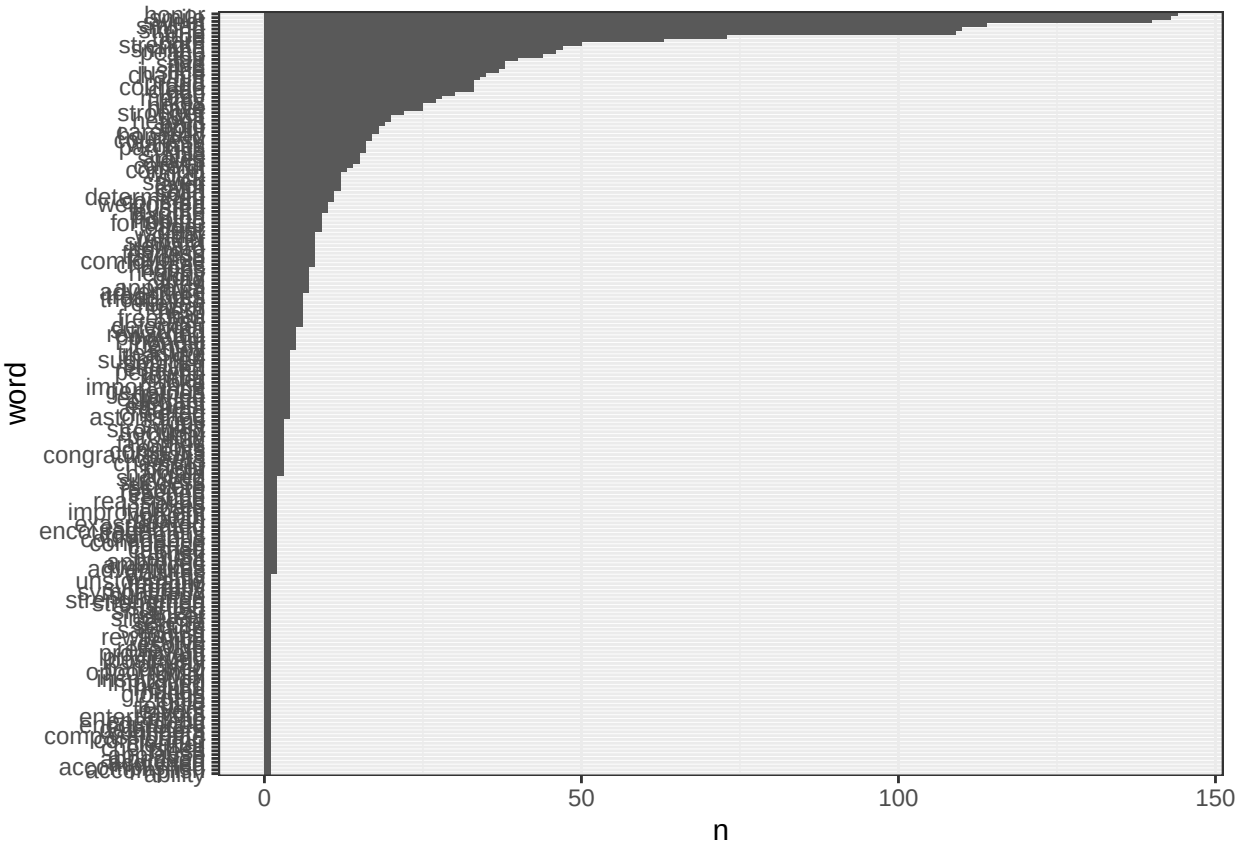
```
unique(got_afinn2$word) #finds the unique words
```

```
##   [1] "smile"          "fine"           "glory"          "hope"
##   [5] "smiled"         "care"           "strength"       "peaceful"
##   [9] "honor"          "carefully"      "slick"          "top"
##  [13] "gained"         "comfort"        "sweet"          "courage"
##  [17] "daring"         "elegant"        "justice"        "heroes"
##  [21] "fair"           "strong"         "brave"          "solid"
##  [25] "proud"          "mercy"          "rescue"         "swift"
##  [29] "smiling"        "true"           "noble"          "saved"
##  [33] "gift"           "treasures"      "favorite"       "clean"
##  [37] "rich"           "fearless"       "fortunate"      "likes"
##  [41] "earnest"        "generous"       "chances"        "smiles"
##  [45] "hug"            "kiss"           "approved"       "fond"
##  [49] "honored"        "consent"        "peace"          "powerful"
##  [53] "worthy"         "humor"          "entertaining"   "save"
##  [57] "sincerely"      "festive"        "careful"        "stronger"
##  [61] "bold"           "eager"          "favored"        "warmth"
##  [65] "pardon"         "pardons"        "healthy"        "loving"
##  [69] "chance"         "thoughtful"     "enjoy"          "privileged"
##  [73] "positively"     "stout"          "encouragement"  "stable"
##  [77] "smarter"        "ease"           "ambitious"      "improvement"
##  [81] "hopeful"        "hopes"          "relieved"       "helping"
##  [85] "cares"          "importance"     "favor"          "tender"
##  [89] "welcomed"       "treasure"       "spirited"       "secured"
##  [93] "courtesy"       "calm"           "resolved"       "courageous"
##  [97] "comfortable"    "sympathy"       "reassuring"     "resolute"
## [101] "brisk"          "appeased"       "enjoying"       "hoping"
## [105] "intricate"      "rescued"        "glorious"       "adventures"
## [109] "friendly"       "astonished"     "reward"         "trusted"
## [113] "honest"         "clever"         "dear"           "favors"
## [117] "determined"     "strengthen"     "approval"       "slicker"
## [121] "sincere"        "jokes"          "joke"           "smartest"
## [125] "favorites"      "hero"           "adventure"      "abilities"
## [129] "strongest"      "courteous"      "exasperated"    "enjoys"
```

```
## [133] "rewarded"        "cherished"       "comforting"      "robust"
## [137] "cherish"         "sympathetic"     "surviving"       "cheered"
## [141] "worth"           "boldly"          "acquitted"       "unstoppable"
## [145] "cheer"           "fervent"         "applause"        "cheers"
## [149] "proudly"         "compassionate"   "bless"           "success"
## [153] "supported"       "kinder"          "improved"        "defender"
## [157] "tranquil"        "helpful"         "hail"            "tops"
## [161] "thankful"        "calmed"          "sunshine"        "opportunity"
## [165] "inspiration"     "survived"        "gain"            "freedom"
## [169] "growth"          "futile"          "swiftly"         "satisfied"
## [173] "congratulations" "confident"       "pardoned"        "energetic"
## [177] "esteemed"        "benefit"         "secure"          "accomplished"
## [181] "support"         "rewarding"       "ability"         "jovial"
## [185] "cheering"        "hailed"          "playful"         "confidence"
## [189] "consents"        "bargain"         "encouraged"      "relieving"
## [193] "accomplish"      "resolve"         "cleaner"         "prominent"
## [197] "serene"          "defenders"       "strengthened"    "wealthy"
## [201] "revive"
```

```
got_afinn2_n <- got_afinn2 %>%
  count(word, sort = TRUE) %>%
  mutate(word = fct_reorder(factor(word), n)) #set up for the ggplot


ggplot(data = got_afinn2_n, aes(x = word, y = n)) +
  geom_col() +
  coord_flip() +
  theme_bw()
```



This was too big - I dont quite know how to cut down the amount of words

Let's find the median and mean of the sentiment of the words

```
got_summary <- got_afinn %>%
  summarize(
    mean_score = mean(value),
    median_score = median(value)
  )

print(got_summary)
```

```
## # A tibble: 1 x 2
##   mean_score median_score
##        <dbl>        <dbl>
## 1     -0.542           -1
```

The words in the GoT.pdf are not quite as positive, as they have a median score of -1

NRC lexicon for sentiment analysis

```
got_nrc <- got_stop %>%
  inner_join(get_sentiments("nrc"))  # Matches words with the NRC lexicon, which categorizes words into
```

```
## Joining with `by = join_by(word)`
```

```
## Warning in inner_join(., get_sentiments("nrc")): Detected an unexpected many-to-many relationship be
```

```
## i Row 147 of `x` matches multiple rows in `y`.
## i Row 9803 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

Before we do the sentiment analysis, I will find out, which words are excluded

```
got_exclude <- got_stop %>%
  anti_join(get_sentiments("nrc")) #finds the excluded wrods
```

```
## Joining with `by = join_by(word)`
```

```
got_exclude_n <- got_exclude %>%
  count(word, sort = TRUE) #counts the excluded words

head(got_exclude_n) #shows the result
```

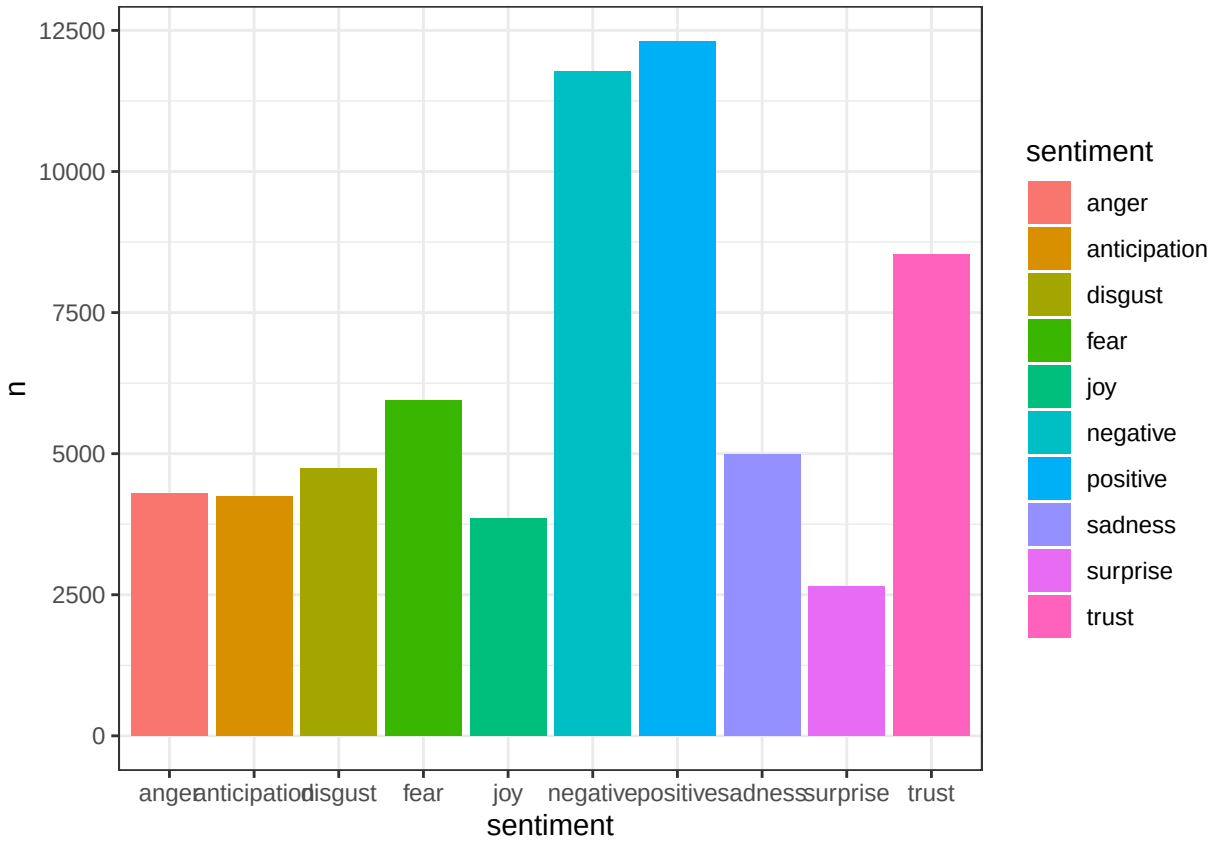
```
## # A tibble: 6 x 2
##   word       n
##   <chr>  <int>
## 1 ser     1023
## 2 jon      787
## 3 ned      743
## 4 tyrion   591
## 5 eyes     567
## 6 hand     567
```

Above are the excluded words

Now we can continue analysing the sentiment

```
got_nrc_n <- got_nrc %>%
  count(sentiment, sort = TRUE)  # Counts how many words belong to each sentiment category

ggplot(data = got_nrc_n, aes(x = sentiment, y = n, fill = sentiment)) +
  geom_col() +  # Visualizes the results in a bar chart
  theme_bw()
```



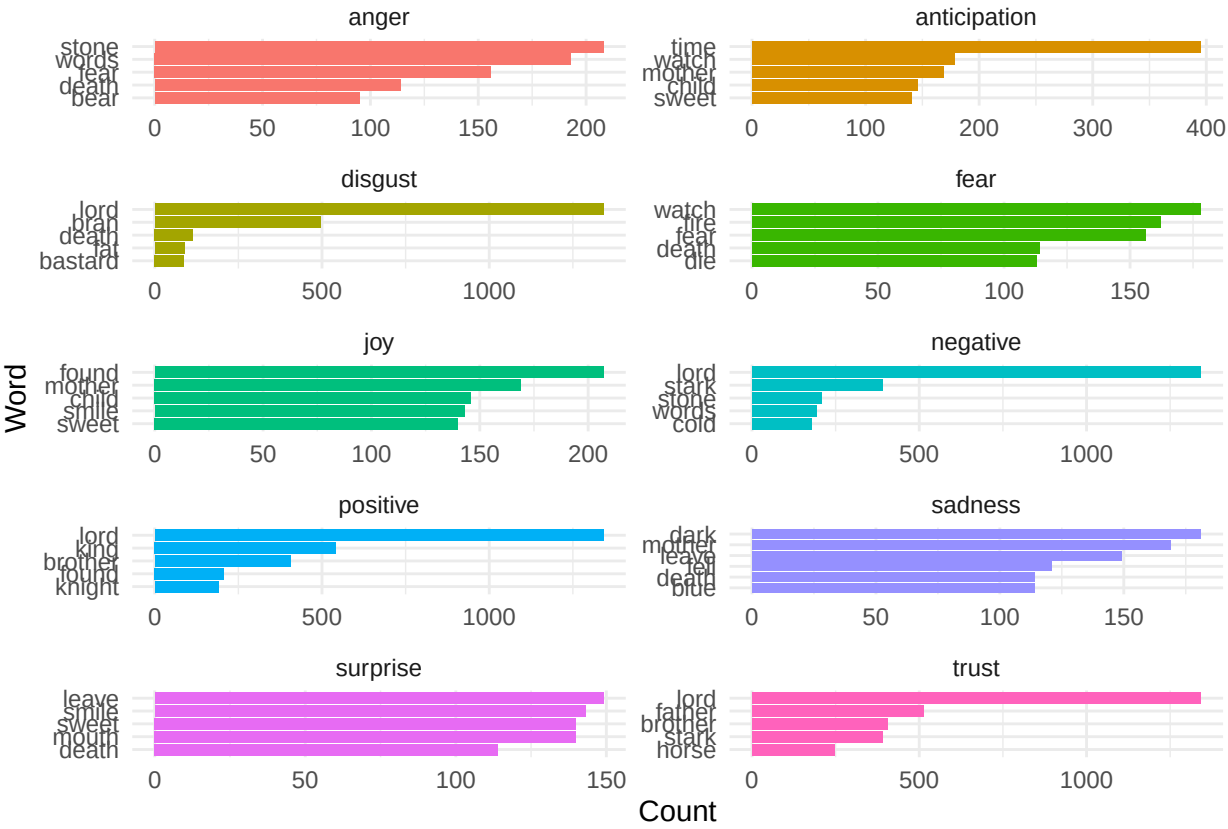
Creating a bar plot of the top words per sentiment category

```
got_nrc_n5 <- got_nrc %>%
  count(word, sentiment, sort = TRUE) %>%  # Counts occurrences of each word categorized by sentiment
  group_by(sentiment) %>%  # Groups by sentiment category
  top_n(5) %>%  # Selects the top 5 most frequent words for each sentiment
  ungroup()  # Removes grouping to allow further independent operations
```

```
## Selecting by n
```

```
# Create a bar plot of the top words per sentiment category
got_nrc_gg <- ggplot(data = got_nrc_n5, aes(x = reorder(word, n), y = n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +  # Creates a bar plot without legend
  facet_wrap(~sentiment, ncol = 2, scales = "free") +  # Creates separate panels for each sentiment
  coord_flip() +  # Rotates the bar chart for better readability
  theme_minimal() +  # Applies a minimalistic theme
  labs(x = "Word", y = "Count")  # Labels the axes

# Show the plot
got_nrc_gg
```



I notice that the word "lord" is in many of the charts...

```
conf <- get_sentiments(lexicon = "nrc") %>%
  filter(word == "lord")

# Yep, check it out:
conf
```

```
## # A tibble: 4 x 2
##   word  sentiment
##   <chr> <chr>
## 1 lord  disgust
## 2 lord  negative
## 3 lord  positive
## 4 lord  trust
```

It was true

Answering the task

My task

Taking this script as a point of departure, apply sentiment analysis on the Game of Thrones. You will find a pdf in the data folder. What are the most common meaningful words and what emotions do you expect will dominate this volume? Are there any terms that are similarly ambiguous to the 'confidence' above?

My answer

Using this script, we applied sentiment analysis to Game of Thrones. The most common meaningful words likely include character names, titles (e.g., “king,” “lord”), and thematic words such as “battle” or “death.”

In terms of emotions, we expect a dominance of fear, anger, and trust, as the book revolves around political intrigue, betrayal, and loyalty.

An ambiguous term similar to “confidence” is “lord.” It appears frequently but does not inherently convey a positive or negative sentiment—it depends on context.