Text mining, sentiment analysis, and visualization Sophie Giambona

created on 22 November 2020 and updated 17 maj, 2025

Note: for more text analysis, you can fork & work through Casey O'Hara and Jessica Couture's eco-data-sci workshop (available here https://github.com/oharac/text_workshop (https://github.com/oharac/text_workshop))

Get the IPCC report:

```
got_path <- here("data","got.pdf")
got_text <- pdf_text(got_path)
```

Some things to notice:

- How cool to extract text out of a PDF! Do you think it will work with any PDF?
- Each row is a page of the PDF (i.e., this is a vector of strings, one for each page)
- The pdf_text() function only sees text that is "selectable"

Example: Just want to get text from a single page (e.g. Page 9)?

```
got_p9 <- got_text[9]
got_p9
```

```
## [1] "one before, when I was half a boy. Everyone talks about snows forty foot deep, and ho
w\nthe ice wind comes howling out of the north, but the real enemy is the cold. It steals up
\non you quieter than Will, and at first you shiver and your teeth chatter and you stamp\nyou
r feet and dream of mulled wine and nice hot fires. It burns, it does. Nothing burns\nlike th
e cold. But only for a while. Then it gets inside you and starts to fill you up, and\nafter a
while you don't have the strength to fight it. It's easier just to sit down or go to\nsleep.
They say you don't feel any pain toward the end. First you go weak and drowsy,\nand everythin
g starts to fade, and then it's like sinking into a sea of warm milk. Peaceful,\nlike."\n\n"S
uch eloquence, Gared," Ser Waymar observed. "I never suspected you had it in you."\n\n"I've h
ad the cold in me too, lordling." Gared pulled back his hood, giving Ser Waymar a\ngood long
look at the stumps where his ears had been. "Two ears, three toes, and the\nlittle finger off
my left hand. I got off light. We found my brother frozen at his watch,\nwith a smile on his
face."\n\nSer Waymar shrugged. "You ought dress more warmly, Gared."\n\nGared glared at the l
ordling, the scars around his ear holes flushed red with anger where\nMaester Aemon had cut t
he ears away. "We'll see how warm you can dress when the\nwinter comes." He pulled up his hoo
d and hunched over his garron, silent and sullen.\n\n"If Gared said it was the cold . . . " W
ill began.\n\n"Have you drawn any watches this past week, Will?"\n\n"Yes, m'lord." There neve
r was a week when he did not draw a dozen bloody watches.\nWhat was the man driving at?\n\n"A
nd how did you find the Wall?"\n\n"Weeping," Will said, frowning. He saw it clear enough, now
that the lordling had\npointed it out. "They couldn't have froze. Not if the Wall was weepin
g. It wasn't cold\nenough."\n\nRoyce nodded. "Bright lad. We've had a few light frosts this p
ast week, and a quick flurry\nof snow now and then, but surely no cold fierce enough to kill
eight grown men. Men\nclad in fur and leather, let me remind you, with shelter near at hand,
and the means of\nmaking fire." The knight's smile was cocksure. "Will, lead us there. I woul
d see these\ndead men for myself."\n"
```

See how that compares to the text in the PDF on Page 9. What has pdftools added and where?

From Jessica and Casey's text mining workshop: "pdf_text() returns a vector of strings, one for each page of the pdf. So we can mess with it in tidyverse style, let's turn it into a dataframe, and keep track of the pages. Then we can use stringr::str_split() to break the pages up into individual lines. Each line of the pdf is concluded with a backslash-n, so split on this. We will also add a line number in addition to the page number."

Some wrangling:

- Split up pages into separate lines (separated by `\n`) using `stringr::str_split()`
- Unnest into regular columns using `tidyr::unnest()`
- Remove leading/trailing white space with `stringr::str_trim()`

```
got_df <- data.frame(got_text) %>%
  mutate(text_full = str_split(got_text, pattern = '\n')) %>%
  unnest(text_full) %>%
  mutate(text_full = str_trim(text_full))

# More information: https://cran.r-project.org/web/packages/stringr/vignettes/regular-expressions.html
```

Now each line, on each page, is its own row, with extra starting & trailing spaces removed.

Get the tokens (individual words) in tidy format

Use `tidytext::unnest_tokens()` (which pulls from the `tokenizer`) package, to split columns into tokens. We are interested in *words*, so that's the token we'll use:

```
got_tokens <- got_df %>%
  unnest_tokens(word, text_full)

# See how this differs from `ipcc_df`
# Each word has its own row!
```

Let's count the words!

```
got_wc <- got_tokens %>%
  count(word) %>%
  arrange(-n)
got_wc
```

```
## # A tibble: 11,826 × 2
##    word      n
##    <chr> <int>
##  1 the   17988
##  2 and    8997
##  3 to     6640
##  4 a      6461
##  5 of     6200
##  6 he     5166
##  7 his    5161
##  8 was    3904
##  9 her    3659
## 10 you    3267
## # ℹ 11,816 more rows
```

OK…so we notice that a whole bunch of things show up frequently that we might not be interested in (“a”, “the”, “and”, etc.). These are called *stop words*. Let’s remove them.

Remove stop words:

See `?stop_words` and `View(stop_words)` to look at documentation for stop words lexicons.

We will *remove* stop words using `tidyr::anti_join()`:

```
got_stop <- got_tokens %>%
  anti_join(stop_words) %>%
  select(-got_text)
```

Now check the counts again:

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

What if we want to get rid of all the numbers (non-text) in `ipcc_stop`?

```
# This code will filter out numbers by asking:
# If you convert to as.numeric, is it NA (meaning those words)?
# If it IS NA (is.na), then keep it (so all words are kept)
# Anything that is converted to a number is removed

got_no_numeric <- got_stop %>%
  filter(is.na(as.numeric(word)))
```

A word cloud of IPCC report words (non-numeric)

See more: https://cran.r-project.org/web/packages/ggwordcloud/vignettes/ggwordcloud.html (https://cran.r-project.org/web/packages/ggwordcloud/vignettes/ggwordcloud.html)

```
# There are almost 2000 unique words
length(unique(got_no_numeric$word))
```

```
## [1] 11209
```

```
# We probably don't want to include them all in a word cloud. Let's filter to only include th
e top 100 most frequent?
got_top100 <- got_no_numeric %>%
  count(word) %>%
  arrange(-n) %>%
  head(100)
got_top100
```

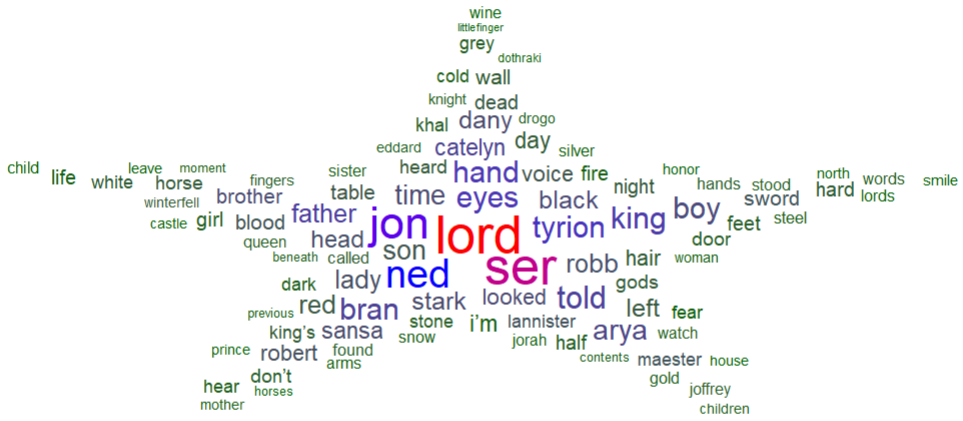
```
## # A tibble: 100 × 2
##    word       n
##    <chr>  <int>
##  1 lord    1341
##  2 ser     1023
##  3 jon      787
##  4 ned      743
##  5 tyrion   591
##  6 eyes     567
##  7 hand     567
##  8 king     542
##  9 father   512
## 10 told     504
## # ℹ 90 more rows
```

```
got_cloud <- ggplot(data = got_top100, aes(label = word)) +
  geom_text_wordcloud() +
  theme_minimal()

got_cloud
```

That's underwhelming. Let's customize it a bit:

```
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()
```



Cool! And you can facet wrap (for different reports, for example) and update other aesthetics. See more here: https://cran.r-project.org/web/packages/ggwordcloud/vignettes/ggwordcloud.html (https://cran.r-project.org/web/packages/ggwordcloud/vignettes/ggwordcloud.html)

Sentiment analysis

First, check out the 'sentiments' lexicon. From Julia Silge and David Robinson (https://www.tidytextmining.com/sentiment.html (https://www.tidytextmining.com/sentiment.html)):

"The three general-purpose lexicons are

- AFINN from Finn Årup Nielsen,
- bing from Bing Liu and collaborators, and
- nrc from Saif Mohammad and Peter Turney

All three of these lexicons are based on unigrams, i.e., single words. These lexicons contain many English words and the words are assigned scores for positive/negative sentiment, and also possibly emotions like joy, anger, sadness, and so forth. The AFINN lexicon assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment. The bing lexicon categorizes words in a binary fashion into positive and negative categories. The nrc lexicon categorizes words in a binary fashion ("yes"/"no") into categories of positive, negative, anger, anticipation, disgust, fear, joy,

sadness, surprise, and trust. All of this information is tabulated in the sentiments dataset, and tidytext provides a function get_sentiments() to get specific sentiment lexicons without the columns that are not used in that lexicon."

Let's explore the sentiment lexicons. "bing" is included, other lexicons ("afinn", "nrc", "loughran") you'll be prompted to download.

WARNING: These collections include very offensive words. I urge you to not look at them in class.

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

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

```
## # A tibble: 2,477 × 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
```

```
# You should be prompted to install lexicon - choose yes!


# Let's look at the pretty positive words:
afinn_pos <- get_sentiments("afinn") %>%
  filter(value %in% c(3,4,5))

# Do not look at negative words in class.
afinn_pos
```

```
## # A tibble: 222 × 2
##    word         value
##    <chr>        <dbl>
##  1 admire           3
##  2 admired          3
##  3 admires          3
##  4 admiring         3
##  5 adorable         3
##  6 adore            3
##  7 adored           3
##  8 adores           3
##  9 affection        3
## 10 affectionate     3
## # ℹ 212 more rows
```

bing: binary, "positive" or "negative"

```
get_sentiments(lexicon = "bing")
```

```
## # A tibble: 6,786 × 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # ℹ 6,776 more rows
```

```
# You should be prompted to install lexicon - choose yes!
```

nrc:https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm (https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm) Includes bins for 8 emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust) and positive / negative.

Citation for NRC lexicon: Crowdsourcing a Word-Emotion Association Lexicon, Saif Mohammad and Peter Turney, Computational Intelligence, 29 (3), 436-465, 2013.

Now nrc:

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

```
## # A tibble: 13,872 × 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
```

```
# You should be prompted to install lexicon - choose yes!
```

Let's do sentiment analysis on the IPCC text data using afinn, and nrc.

Sentiment analysis with afinn:

First, bind words in `ipcc_stop` to `afinn` lexicon:

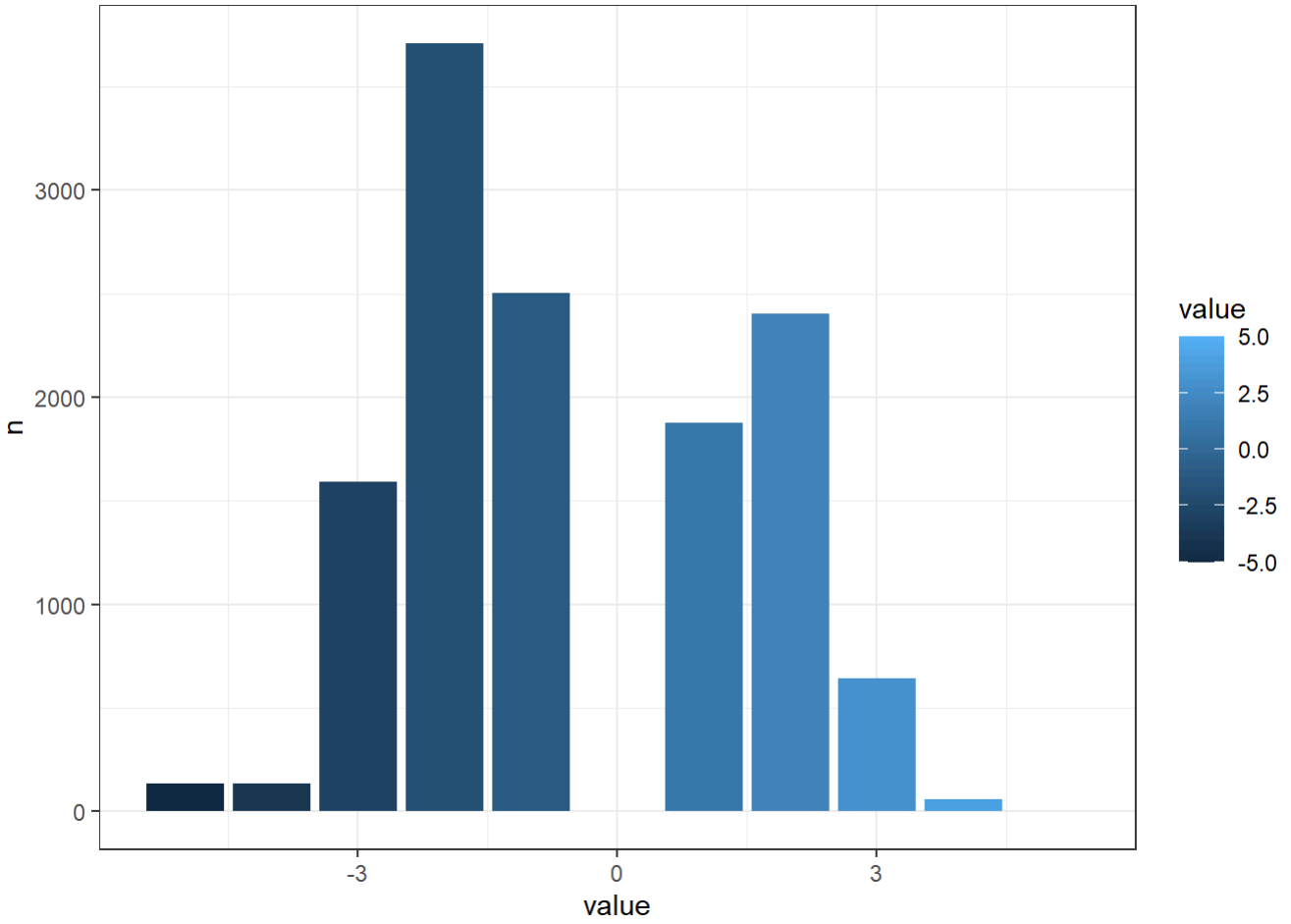
```
got_afinn <- got_stop %>%
  inner_join(get_sentiments("afinn"))
got_afinn
```

```
## # A tibble: 13,058 × 2
##    word  value
##    <chr> <dbl>
##  1 fire     -2
##  2 dead     -3
##  3 dead     -3
##  4 smile     2
##  5 dead     -3
##  6 dead     -3
##  7 dead     -3
##  8 dead     -3
##  9 dead     -3
## 10 drag     -1
## # ℹ 13,048 more rows
```

Let's find some counts (by sentiment ranking):

```
got_afinn_hist <- got_afinn %>%
  count(value)

# Plot them:
ggplot(data = got_afinn_hist, aes(x = value, y = n)) +
  geom_col(aes(fill = value)) +
  theme_bw()
```



Investigate some of the words in a bit more depth:

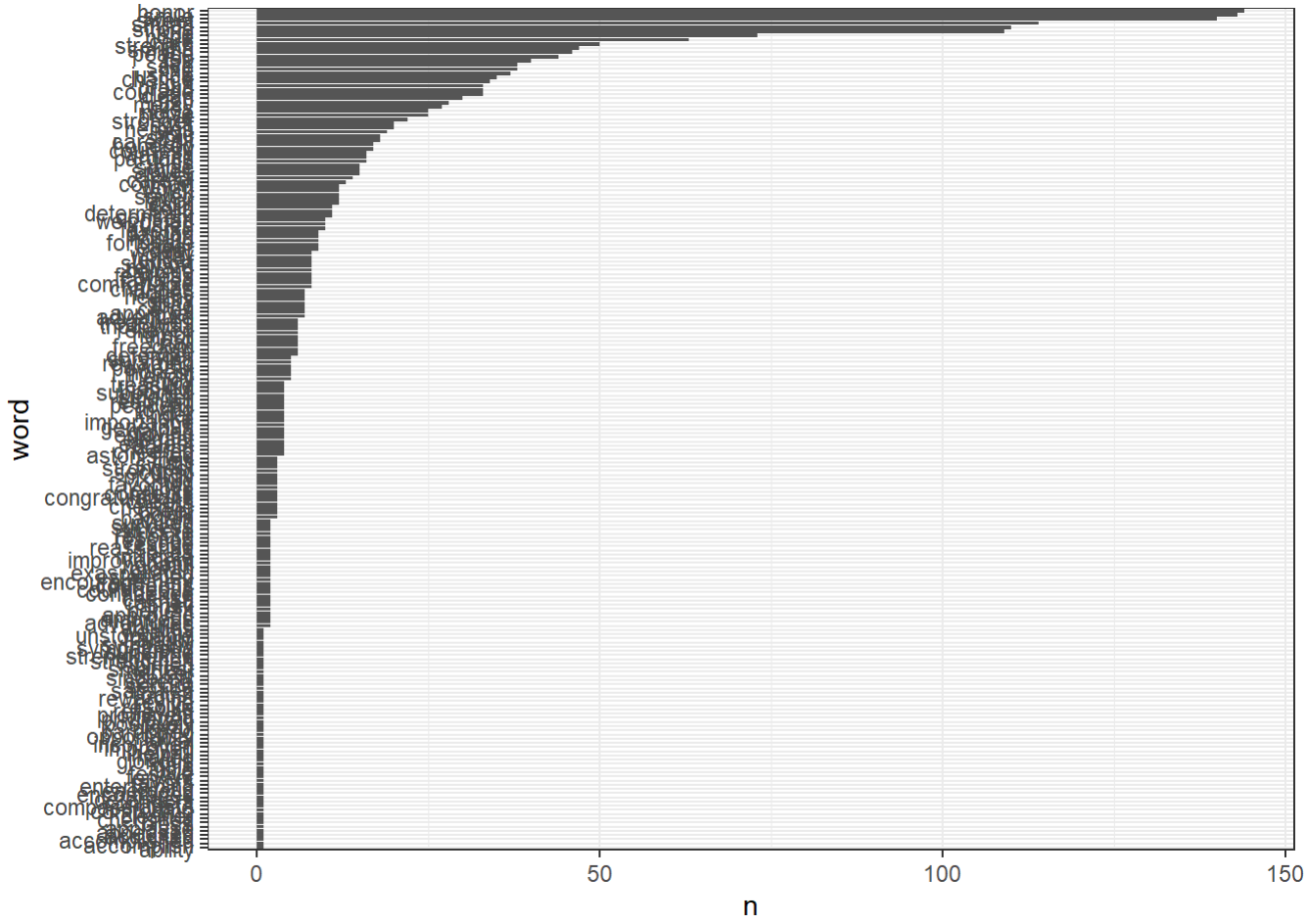
```
# What are these '2' words?
got_afinn2 <- got_afinn %>%
  filter(value == 2)
```

```
# Check the unique 2-score words:
unique(got_afinn2$word)
```

```
##   [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"
```

```
# Count & plot them
got_afinn2_n <- got_afinn2 %>%
  count(word, sort = TRUE) %>%
  mutate(word = fct_reorder(factor(word), n))


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




```
# OK so what's the deal with confidence? And is it really "positive" in the emotion sense?
```

Look back at the IPCC report, and search for “confidence.” Is it typically associated with emotion, or something else?

We learn something important from this example: Just using a sentiment lexicon to match words will not differentiate between different uses of the word…(ML can start figuring it out with context, but we won’t do that here).

Or we can summarize sentiment for the report:

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

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

The mean and median indicate *slightly* positive overall sentiments based on the AFINN lexicon.

NRC lexicon for sentiment analysis

We can use the NRC lexicon to start "binning" text by the feelings they're typically associated with. As above, we'll use inner_join() to combine the IPCC non-stopword text with the nrc lexicon:

```
got_nrc <- got_stop %>%
  inner_join(get_sentiments("nrc"))
```

Wait, won't that exclude some of the words in our text? YES! We should check which are excluded using `anti_join()`:

```
got_exclude <- got_stop %>%
  anti_join(get_sentiments("nrc"))

# View(ipcc_exclude)

# Count to find the most excluded:
got_exclude_n <- got_exclude %>%
  count(word, sort = TRUE)

head(got_exclude_n)
```

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

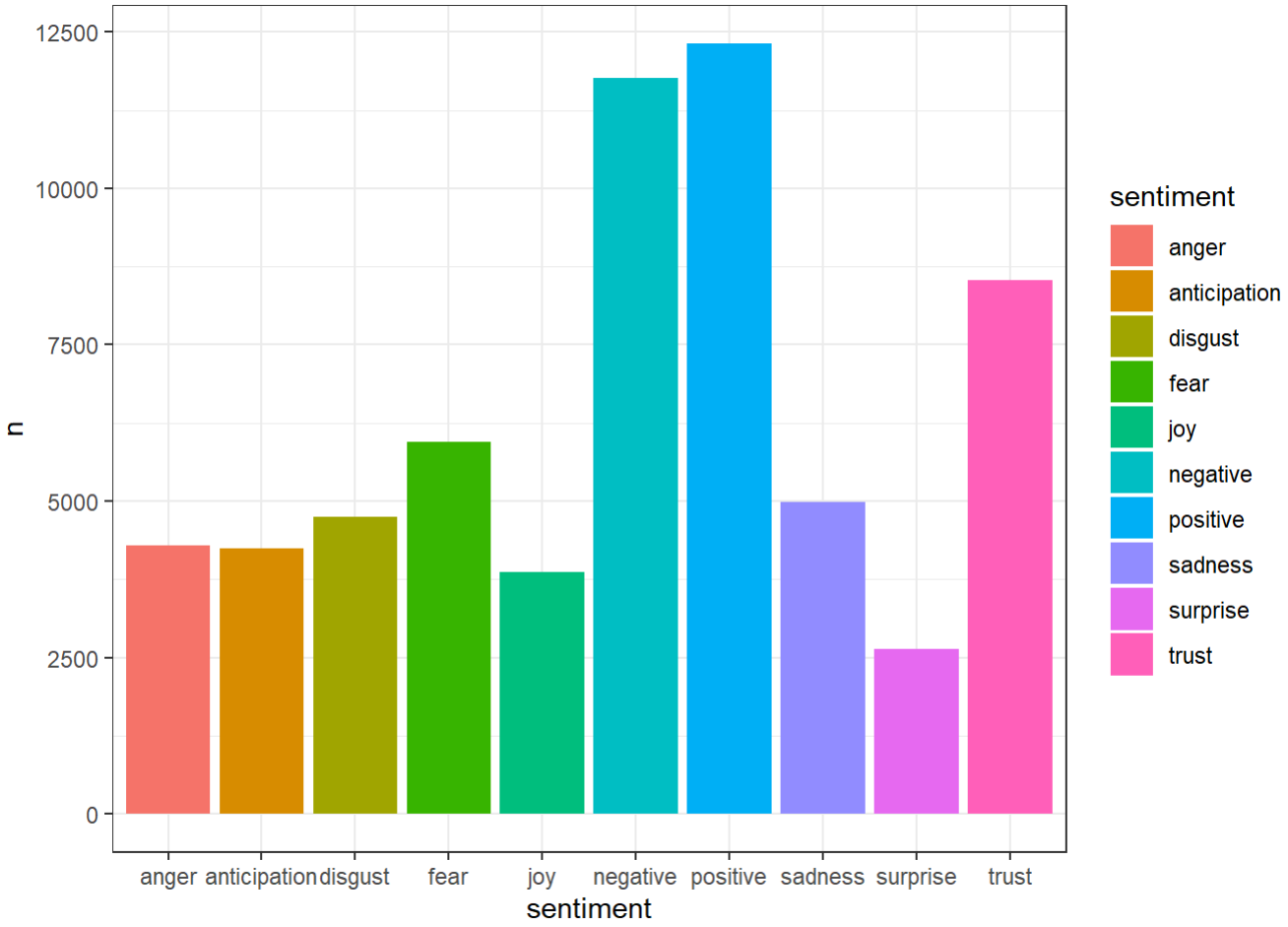
Lesson: always check which words are EXCLUDED in sentiment analysis using a pre-built lexicon!

Now find some counts:

```
got_nrc_n <- got_nrc %>%
  count(sentiment, sort = TRUE)

# And plot them:

ggplot(data = got_nrc_n, aes(x = sentiment, y = n)) +
  geom_col(aes(fill = sentiment))+
  theme_bw()
```

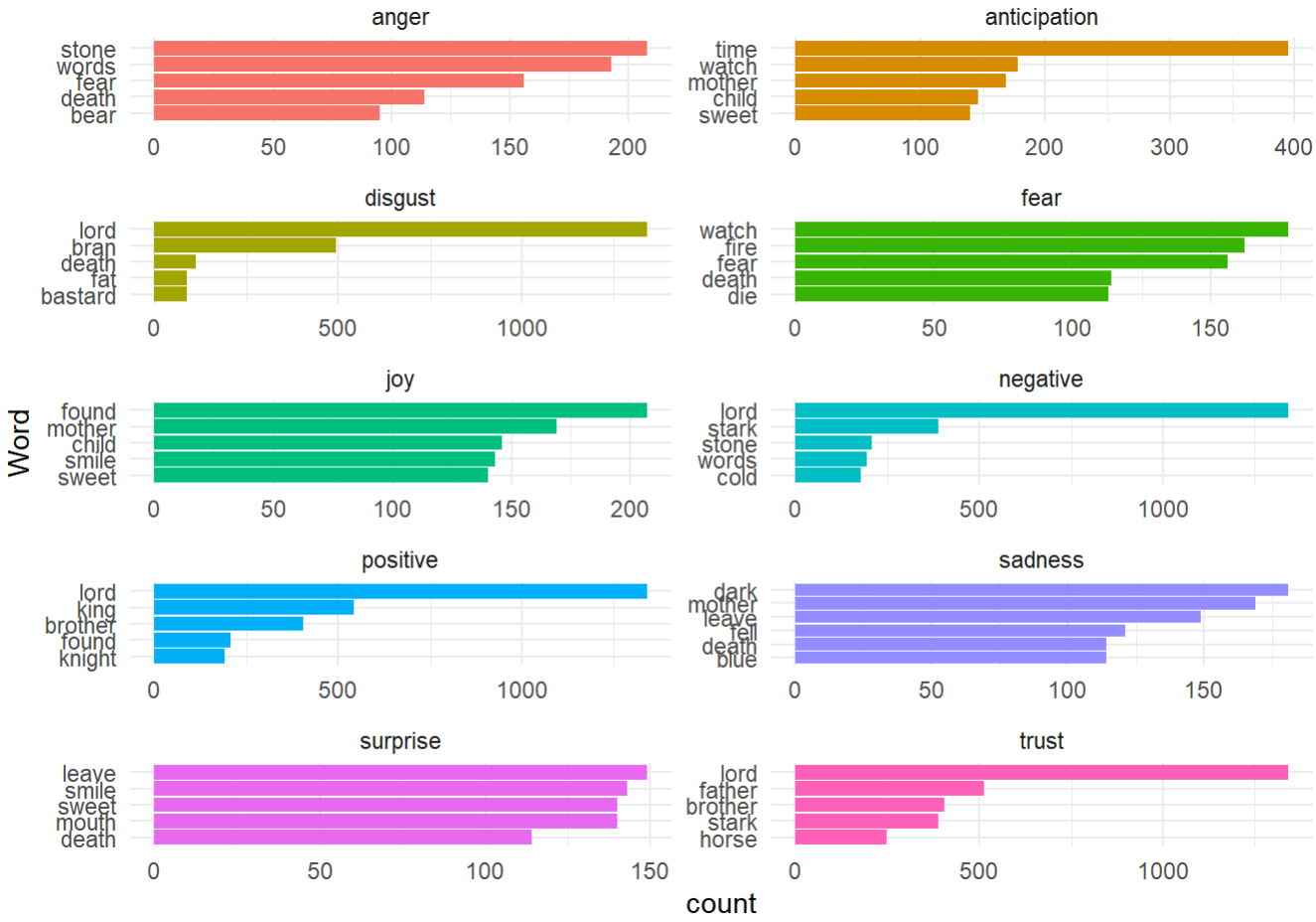


Or count by sentiment *and* word, then facet:

```
got_nrc_n5 <- got_nrc %>%
  count(word,sentiment, sort = TRUE) %>%
  group_by(sentiment) %>%
  top_n(5) %>%
  ungroup()

got_nrc_gg <- ggplot(data = got_nrc_n5, aes(x = reorder(word,n), y = n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, ncol = 2, scales = "free") +
  coord_flip() +
  theme_minimal() +
  labs(x = "Word", y = "count")

# Show it
got_nrc_gg
```




```
# Save it
ggsave(plot = got_nrc_gg,
       here("figures","got_sentiment.png"),
       height = 8,
       width = 5)
```

Wait, so “confidence” is showing up in NRC lexicon as “fear”? Let’s check:

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

# Yep, check it out:
lord
```

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

Big picture takeaway

There are serious limitations of sentiment analysis using existing lexicons, and you should **think really hard** about your findings and if a lexicon makes sense for your study. Otherwise, word counts and exploration alone can be useful!

Your 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?

Credits:

This tutorial is inspired by Allison Horst’s Advanced Statistics and Data Analysis.