Week 5: Assignment 4: Word relationship analysis

Scout Leonard

2022-05-02

Load Libraries

The code chunk below loads the packages required for this homeowrk assignment:

```
packages=c("tidyr",
           "pdftools",
           "lubridate",
           "tidyverse",
           "tidytext",
           "readr",
           "quanteda",
           "readtext",
           "quanteda.textstats",
           "quanteda.textplots",
           "ggplot2",
           "forcats",
           "stringr",
           "quanteda.textplots",
           "widyr",
           "igraph",
           "ggraph",
           "here")

for (i in packages) {
  if (require(i,character.only=TRUE)==FALSE) {
    install.packages(i,repos='http://cran.us.r-project.org')
  }
  else {
    require(i,character.only=TRUE)
  }
}
```

Read in data

Read in files and make corpus

The code chunk below reads in the EPA report PDFs, makes a corpus of the documents, and removes stop words from the corpus. It also prints a dataframe of the corpus without stop words:

```
#filepath to data
files <- list.files(path = here("data/week5_data/"),
                    pattern = "*pdf$",
                    full.names = TRUE)
```

```
#renders all textboxes on a text canvas and returns a character vector of equal length to the number of
ej_reports <- lapply(files, pdf_text)

#read texts and (if any) associated document-level meta-data from one or more source files - makes a df
ej_pdf <- readtext(file = here("data/week5_data/*.pdf"),
                   docvarsfrom = "filenames",
                   docvarnames = c("type", "subj", "year"),
                   sep = "_")

#creating an initial corpus containing our data
epa_corp <- corpus(x = ej_pdf, text_field = "text" )

#return details of the corpus
summary(epa_corp) %>%
  knitr::kable()
```

Text	Types	Tokens	Sentences	type	subj	year
EPA_EJ_2015.pdf	2136	8944	263	EPA	EJ	2015
EPA_EJ_2016.pdf	1599	7965	176	EPA	EJ	2016
EPA_EJ_2017.pdf	2774	16658	447	EPA	EJ	2017
EPA_EJ_2018.pdf	3973	30564	653	EPA	EJ	2018
EPA_EJ_2019.pdf	3773	22648	672	EPA	EJ	2019
EPA_EJ_2020.pdf	4493	30523	987	EPA	EJ	2020

```
#I'm adding some additional, context-specific stop words to stop word lexicon
more_stops <-c("2015","2016", "2017", "2018",
               "2019", "2020", "www.epa.gov", "https")

#add the additional stopwords to the stop word lexicon
add_stops <- tibble(word = c(stop_words$word, more_stops))

stop_vec <- as_vector(add_stops)
```

Count and tokenize words

The code chunk below converts the EPA report corpus data to tidy format:

```
#convert to tidy format and apply my stop words
raw_text <- tidy(epa_corp)

#Distribution of most frequent words across documents
raw_words <- raw_text %>%
  mutate(year = as.factor(year)) %>%
  unnest_tokens(word, text) %>%
  anti_join(add_stops, by = 'word') %>%
  count(year, word, sort = TRUE)

#number of total words by document
total_words <- raw_words %>%
  group_by(year) %>%
  summarize(total = sum(n))
```

```
report_words <- left_join(raw_words, total_words)
```

```
## Joining, by = "year"
```

```
par_tokens <- unnest_tokens(raw_text,
                            output = paragraphs,
                            input = text,
                            token = "paragraphs")

par_tokens <- par_tokens %>%
 mutate(par_id = 1:n())

par_words <- unnest_tokens(par_tokens,
                           output = word,
                           input = paragraphs,
                           token = "words")
```

Part 1

What are the most frequent trigrams in the dataset? How does this compare to the most frequent bigrams? Which n-gram seems more informative here, and why?

```
tokens <- tokens(epa_corp, remove_punct = TRUE) #list of character vectors - takes each document and sp

toks1<- tokens_select(tokens, min_nchar = 3)

toks1 <- tokens_tolower(toks1)

toks1 <- tokens_remove(toks1, pattern = (stop_vec))

dfm <- dfm(toks1) #create document feature matrix - rows are number of occurances of each word within e

#first the basic frequency stat
tstat_freq <- textstat_frequency(dfm, n = 5, groups = year)

head(tstat_freq, 10) %>%
  knitr::kable()
```

feature	frequency	rank	docfreq	group
environmental	127	1	1	2015
communities	99	2	1	2015
epa	92	3	1	2015
justice	84	4	1	2015
community	47	5	1	2015
environmental	109	1	1	2016
communities	85	2	1	2016
justice	71	3	1	2016
epa	48	4	1	2016
federal	31	5	1	2016

```
toks2 <- tokens_ngrams(toks1, n = 3)

dfm2 <- dfm(toks2)

dfm2 <- dfm_remove(dfm2, pattern = c(stop_vec))
#gives more coherent terms - power of chunking at a different token level

freq_words2 <- textstat_frequency(dfm2, n = 20)

freq_words2$token <- rep("trigram", 20)
#tokens1 <- tokens_select(tokens1,pattern = stopwords("en"), selection = "remove")

head(freq_words2, 5) %>%
  knitr::kable()
```

feature	frequency	rank	docfreq	group	token
justice_fy2017_progress	51	1	1	all	trigram
fy2017_progress_report	51	1	1	all	trigram
environmental_public_health	50	3	6	all	trigram
environmental_justice_fy2017	50	3	1	all	trigram

feature	frequency	rank	docfreq	group	token
national_environmental_justice	37	5	6	all	trigram

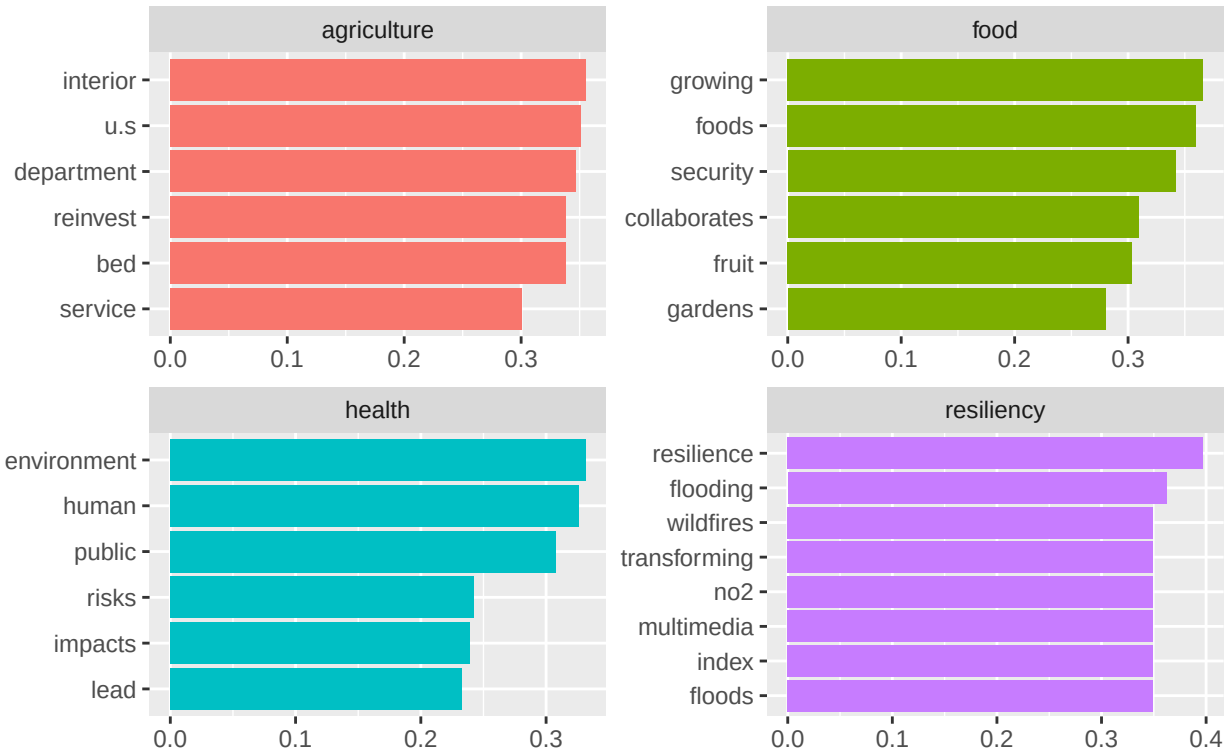
The most frequent trigrams in the dataset are shown in the table above, with justice_fy2017_progress as the most frequently occurring trigram. This is less informative than the bigram frequency because the word groupings are slightly more nonsensical, probably because linguistically bigrams are more common and trigrams.

Part 2

Choose a new focal term to replace "justice" and recreate the correlation table and network (see corr_paragraphs and corr_network chunks). Explore some of the plotting parameters in the cor_network chunk to see if you can improve the clarity or amount of information your plot conveys. Make sure to use a different color for the ties!

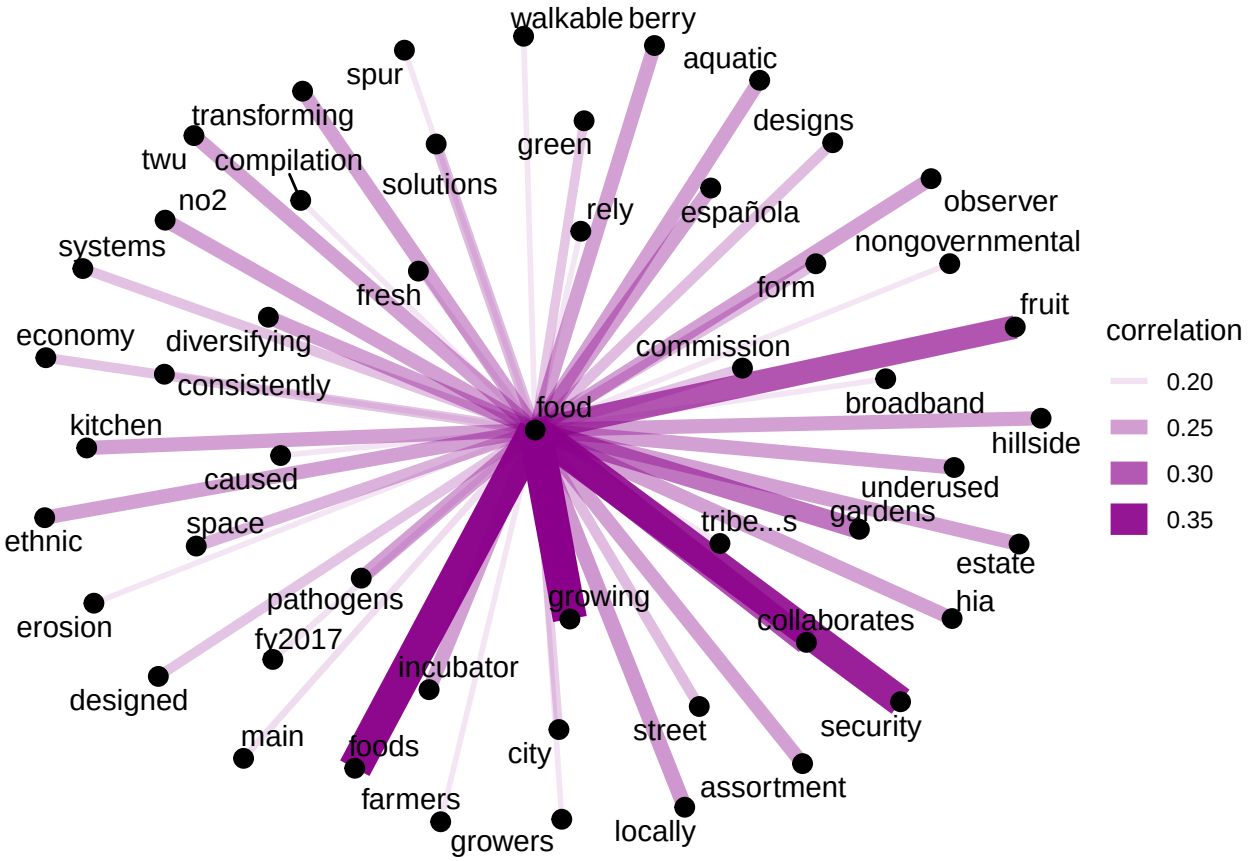
```
#correlation between co-occuring words
word_cors <- par_words %>%
  add_count(par_id) %>%
  filter(n >= 50) %>%
  select(-n) %>%
  pairwise_cor(word, par_id, sort = TRUE)

#now we can select words cooccurring with the word justice and get correlation coefficients
food_cors <- word_cors %>%
  filter(item1 == "food")

  word_cors %>%
  filter(item1 %in% c("food", "agriculture", "health", "resiliency")) %>%
  group_by(item1) %>%
  top_n(6) %>%
  ungroup() %>%
  mutate(item1 = as.factor(item1),
  name = reorder_within(item2, correlation, item1)) %>%
  ggplot(aes(y = name, x = correlation, fill = item1)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~item1, ncol = 2, scales = "free")+
  scale_y_reordered() +
  labs(y = NULL,
         x = NULL,
         title = "Correlations with key words",
         subtitle = "EPA EJ Reports")
```

```
## Selecting by correlation
```




```
  #let's zoom in on just one of our key terms
   food_cors <- word_cors %>%
  filter(item1 == "food") %>%
   mutate(n = 1:n())
```

```
food_cors  %>%
  filter(n <= 50) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = correlation,
                     edge_width = correlation),
                 edge_colour = "darkmagenta") +
  geom_node_point(size = 3) +
  geom_node_text(aes(label = name),
                 repel = TRUE,
                 point.padding = unit(0.2,
                                      "lines")) +
  theme_void()
```


walkable
berry
spur
aquatic
transforming
designs
green
twu
compilation
solutions
no2
rely
española
observer
systems
nongovernmental
fresh
form
fruit
economy
diversifying
commission
consistently
broadband
kitchen
food
hillside
caused
underused
gardens
space
tribe...s
ethnic
estate
pathogens
growing
hia
erosion
collaborates
fy2017
incubator
designed
security
main
foods
street
city
assortment
farmers
growers
locally
correlation
0.20
0.25
0.30
0.35

Part 3

Write a function that allows you to conduct a keyness analysis to compare two individual EPA reports (hint: that means target and reference need to both be individual reports). Run the function on 3 pairs of reports, generating 3 keyness plots.

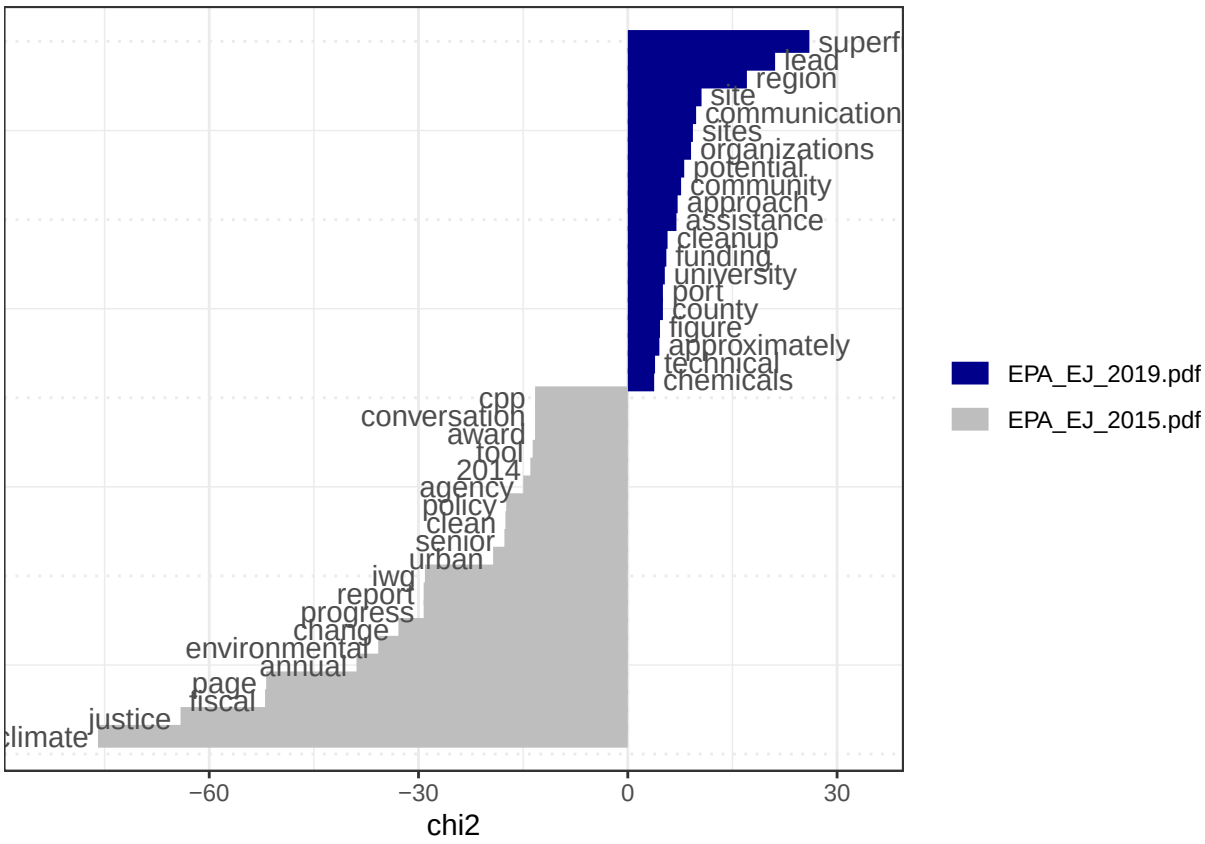
```
#test function
dfm_subset <- corpus_subset(epa_corp, grepl("2018|2019", docnames(epa_corp)))
```

```
#write function
keyness_comparison <- function(text1_year, text2_year) {

  #subset the corpus
  corpus_subset <- corpus_subset(epa_corp, grepl(paste0(text1_year, "|", text2_year), docnames(epa_corp)

  #tokenize corpus
  tokens <- tokens(corpus_subset, remove_punct = TRUE) #list of character vectors - takes each document
  toks<- tokens_select(tokens, min_nchar = 3)
  toks <- tokens_tolower(toks)
  toks <- tokens_remove(toks, pattern = (stop_vec))
  dfm <- dfm(toks) #create document feature matrix - rows are number of occurances of each word within

  keyness <- textstat_keyness(dfm, target = 2)

  textplot_keyness(keyness)

}
```

Keyness plot 1

Test running the function on 2015 and 2019

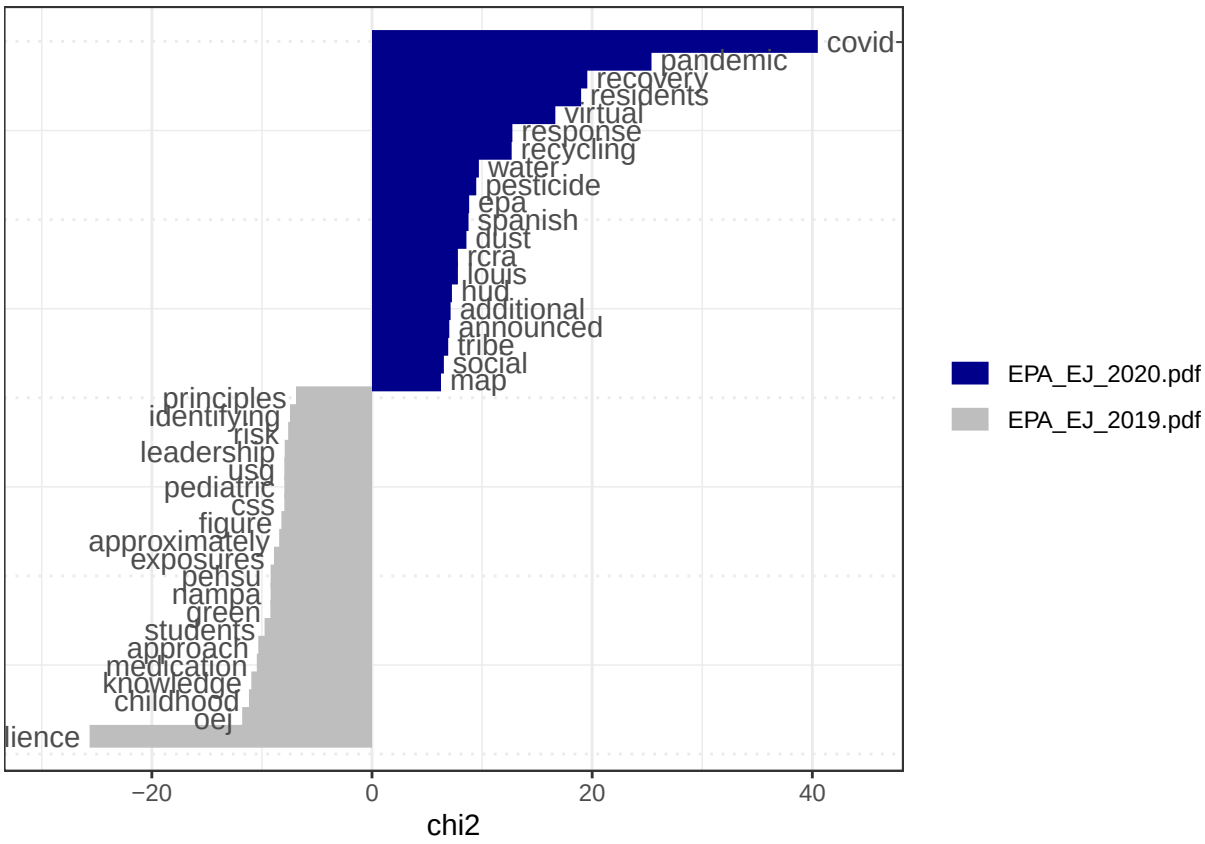
```
keyness_comparison(text1_year = 2015, text2_year = 2019)
```



Keyness plot 2

Test running the function on 2019 and 2020

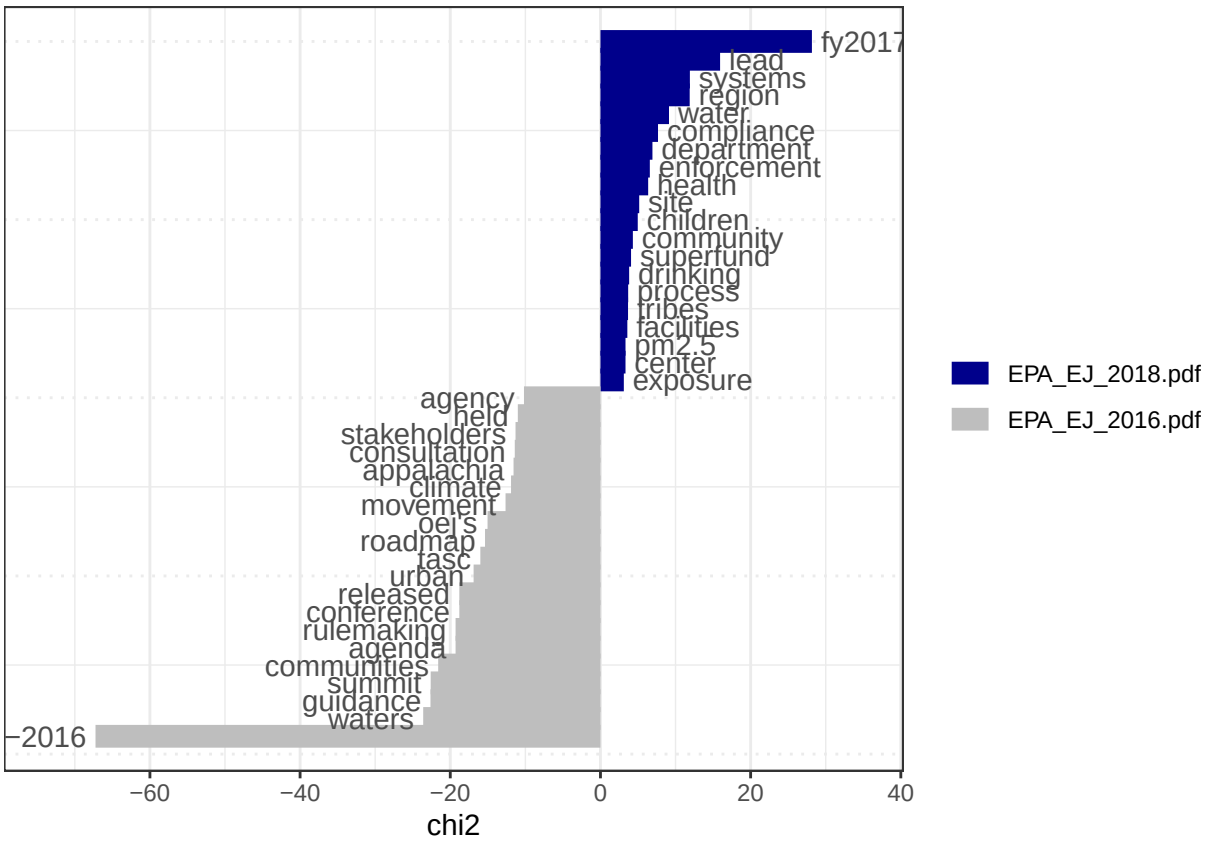
```
keyness_comparison(text1_year = 2019, text2_year = 2020)
```



Keyness plot 3

Test running the function on 2016 and 2018:

```
keyness_comparison(text1_year = 2016, text2_year = 2018)
```



Part 4

Select a word or multi-word term of interest and identify words related to it using windowing and keyness comparison. To do this you will create two objects: one containing all words occurring within a 10-word window of your term of interest, and the second object containing all other words. Then run a keyness comparison on these objects. Which one is the target, and which the reference? Hint

food systems

Create an object containing all words occurring within a 10 word window of **food systems**.

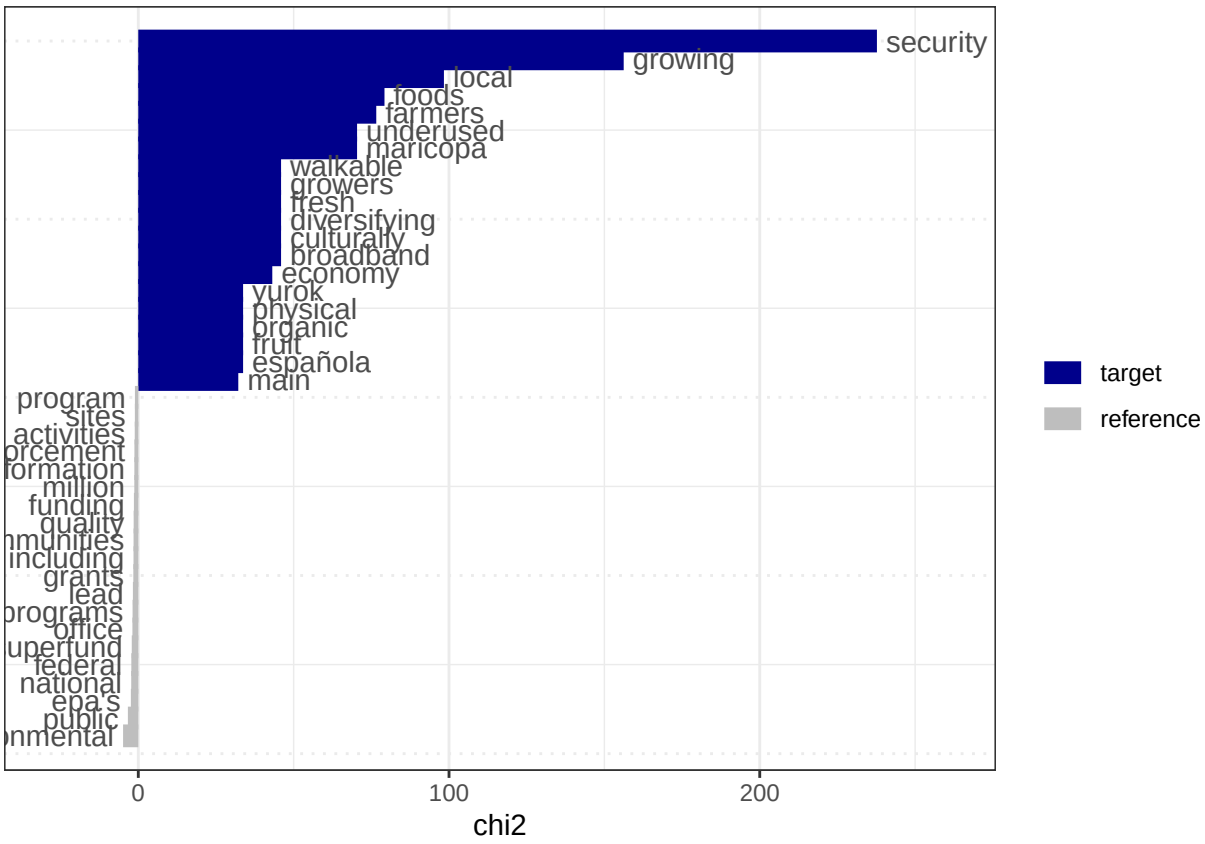
```
term = "food"

toks_inside <- tokens_keep(toks1,
                           pattern = term,
                           window = 10)

toks_inside <- tokens_remove(toks_inside,
                             pattern = term)
```

Create an object containing all other words:

```
toks_outside <- tokens_remove(toks1,
                              pattern = term,
                              window = 10)
```

Run a keyness comparison of the objects:

```
dfmat_inside <- dfm(toks_inside)
dfmat_outside <- dfm(toks_outside)

tstat_key_inside <- textstat_keyness(rbind(dfmat_inside, dfmat_outside),
                                     target = seq_len(ndoc(dfmat_inside)))

textplot_keyness(tstat_key_inside)
```



`toks_inside` is the target, and `toks_outside` is the reference.