Business Case: Aunt Serena Pancakes

Text Analytics and Natural Language Processing (NLP) | Team 10 | MsBA1

16/02/2021

1.) Import of Libraries & Data Set

As a first step we imported all the necessary libraries and the text files that stored our survey answers.

```
############################################
##########Loading Survey Data ##############
############################################

library(shinyBS)
library(rintrojs)
library(shinydashboard)
library(shiny)
library(shinyWidgets)
library(shinythemes)
library(DT)
library(dplyr)
library(purrr)
library(tidyverse)
library(tidytext)
library(textdata)
library(widyr)
library(tidyr)
library(stringr)
library(scales)
library(twitteR)
library(rtweet)
library(tm)
library(ggplot2)
library(igraph)
library(ggraph)
library(reshape2)
library(wordcloud)
library(readr)
library(plotly)

Question_1 <- read_delim("~/Documents/Hult Master Spring 2020/Text Analytics/T10 Group Assignment/Questi
                         "\t", escape_double = FALSE, col_names = FALSE,
                         trim_ws = TRUE)


Question_2 <- read_delim("~/Documents/Hult Master Spring 2020/Text Analytics/T10 Group Assignment/Questi
                         "\t", escape_double = FALSE, col_names = FALSE,
                         trim_ws = TRUE)

Question_3 <- read_delim("~/Documents/Hult Master Spring 2020/Text Analytics/T10 Group Assignment/Questi
                         "\t", escape_double = FALSE, col_names = FALSE,
                         trim_ws = TRUE)

Question_4 <- read_delim("~/Documents/Hult Master Spring 2020/Text Analytics/T10 Group Assignment/Questi
                         "\t", escape_double = FALSE, col_names = FALSE,
                         trim_ws = TRUE)

Question_5 <- read_delim("~/Documents/Hult Master Spring 2020/Text Analytics/T10 Group Assignment/Questi
                         "\t", escape_double = FALSE, col_names = FALSE,
                         trim_ws = TRUE)

Question_6 <- read_delim("~/Documents/Hult Master Spring 2020/Text Analytics/T10 Group Assignment/Questi
                         "\t", escape_double = FALSE, col_names = FALSE,
                         trim_ws = TRUE)


Question_7 <- read_delim("~/Documents/Hult Master Spring 2020/Text Analytics/T10 Group Assignment/Questi
                         "\t", escape_double = FALSE, col_names = FALSE,
                         trim_ws = TRUE)

Question_8 <- read_delim("~/Documents/Hult Master Spring 2020/Text Analytics/T10 Group Assignment/Questi
                         "\t", escape_double = FALSE, col_names = FALSE,
                         trim_ws = TRUE)
```

2.) Data Massaging

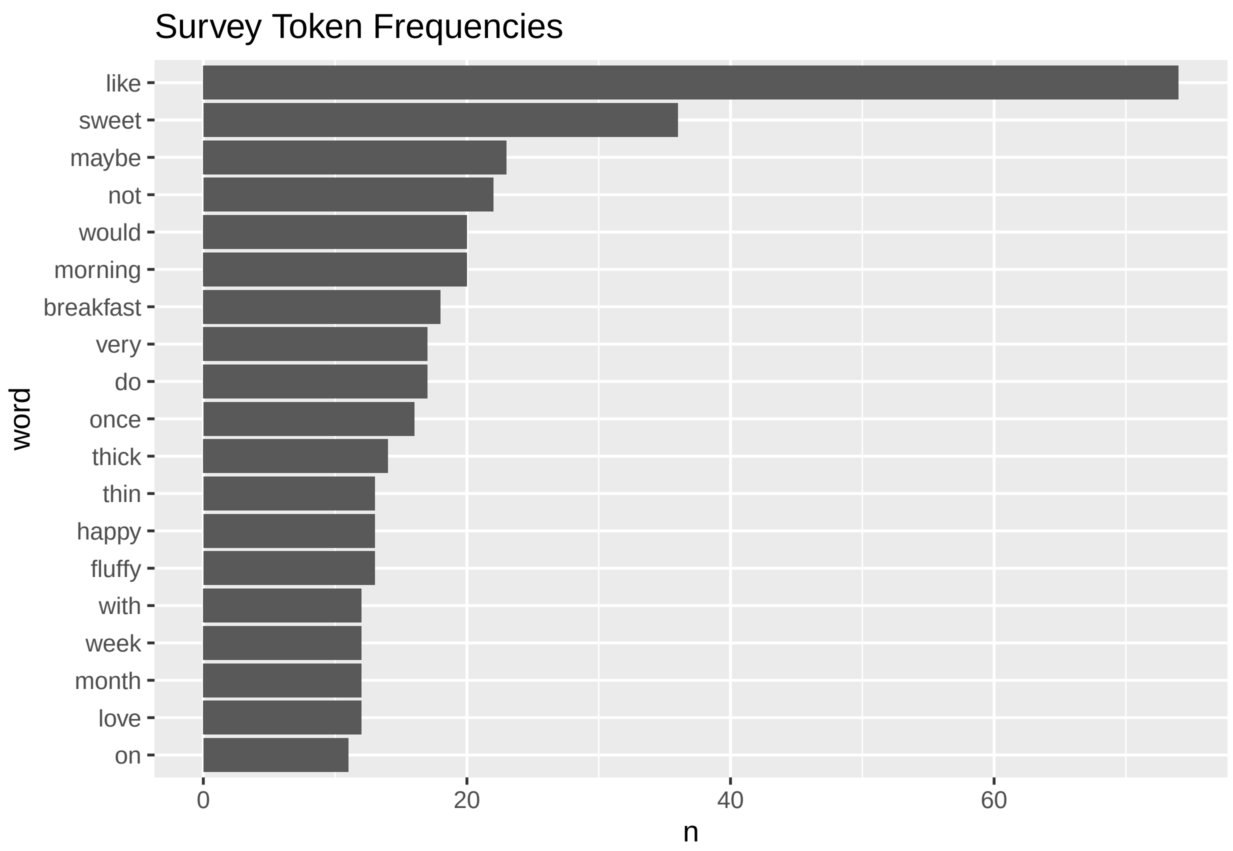
As a second step we are going to massage our survey data.

```
options(knitr.duplicate.label = 'allow')

#Assigning question number to each row
Question_1$question <- "Q1"
Question_2$question <- "Q2"
Question_3$question <- "Q3"
Question_4$question <- "Q4"
Question_5$question <- "Q5"
Question_6$question <- "Q6"
Question_7$question <- "Q7"
Question_8$question <- "Q8"

#Creating a data frame with survey data
survey_df <- rbind.data.frame(Question_1,
                              Question_2,
                              Question_3,
                              Question_4,
                              Question_5,
                              Question_6,
                              Question_7,
                              Question_8)
#calling the stop words library
data(stop_words)

#creating an object with a txt file of custom stop words

to_keep <- read_delim("~/Documents/Hult Master Spring 2020/Text Analytics/T10 Group Assignment/T10_Shin
                                "\t", escape_double = FALSE, col_names = FALSE,
                                trim_ws = TRUE) %>%
  rename(word = X1)



#creating my own stop_words
custom_stop_words <- tribble(
    ~word, ~lexicon,
    "yeah", "CUSTOM",
    "pancakes", "CUSTOM",
    "pancake", "CUSTOM",
    "eat", "CUSTOM",
    "prefer", "CUSTOM",
    "feel", "CUSTOM",
    "favorite", "CUSTOM",
    "toppings", "CUSTOM",
)



#joining the custom stop words to the stop words
stop_words2 <- stop_words  %>%
    anti_join(to_keep) %>%
    bind_rows(custom_stop_words)
```

3.) Tokenizing

Next, we tokenized our survey data.

```
############################################
#######Tokenizing Survey Data ############
############################################

survey_counts <- survey_df %>%
  unnest_tokens(word, X1) %>%
  anti_join(stop_words2)%>%
  count(word, sort=TRUE)

tok_Q1 <- survey_df %>%
    filter(question == "Q1") %>%
    unnest_tokens(word, X1) %>%
    anti_join(stop_words2)

tok_Q2 <- survey_df %>%
    filter(question == "Q2") %>%
    unnest_tokens(word, X1) %>%
    anti_join(stop_words2)

tok_Q3 <- survey_df %>%
    filter(question == "Q3") %>%
    unnest_tokens(word, X1) %>%
    anti_join(stop_words2)

tok_Q4 <- survey_df %>%
    filter(question == "Q4") %>%
    unnest_tokens(word, X1) %>%
    anti_join(stop_words2)

tok_Q5 <- survey_df %>%
    filter(question == "Q5") %>%
    unnest_tokens(word, X1) %>%
    anti_join(stop_words2)

tok_Q6 <- survey_df %>%
    filter(question == "Q6") %>%
    unnest_tokens(word, X1) %>%
    anti_join(stop_words2)

tok_Q7 <- survey_df %>%
    filter(question == "Q7") %>%
    unnest_tokens(word, X1) %>%
    anti_join(stop_words2)

tok_Q8 <- survey_df %>%
    filter(question == "Q8") %>%
    unnest_tokens(word, X1)

############################################
#########Token frequency histograms#########
############################################

freq_survey <-survey_counts %>%
  filter(n > 10) %>% # we need this to eliminate all the low count words
  mutate(word = reorder(word,n )) %>%
  ggplot(aes(word, n))+
  geom_col()+
  labs(title = "Survey Token Frequencies")+
  coord_flip()
print(freq_survey)
```



4.) N-grams, Zip Law & TFIDF

We did an analysis for all our questions together concerning the n-grams, zip law and tf-idf.

```
###############TFIDF ALL QUESTIONS######################
questions <- bind_rows(mutate(Question_1, author = "Question 1"),
                      mutate(Question_2, author = "Question 2"),
                      mutate(Question_3, author = "Question 3"),
                      mutate(Question_4, author = "Question 4"),
                      mutate(Question_5, author = "Question 5"),
                      mutate(Question_6, author = "Question 6"),
                      mutate(Question_7, author = "Question 7"),
                      mutate(Question_8, author = "Question 8"),)
questions_tokens <- questions %>%
  unnest_tokens(word, X1) %>%
  anti_join(stop_words2) %>%
  count(word, sort=T)

#############################################
###### N-grams and tokenizing ###############
#############################################

questions_bigrams <- questions %>%
  unnest_tokens(bigram, X1, token = "ngrams", n=2)%>%
  filter(!is.na(bigram))
```

```r
questions_bigrams #We want to see the bigrams (words that appear together, "pairs")
```

```
## # A tibble: 2,087 x 3
##    question author     bigram
##    <chr>    <chr>      <chr>
##  1 Q1       Question 1 i do
##  2 Q1       Question 1 do like
##  3 Q1       Question 1 like pancakes
##  4 Q1       Question 1 i like
##  5 Q1       Question 1 like pancakes
##  6 Q1       Question 1 i don’t
##  7 Q1       Question 1 don’t like
##  8 Q1       Question 1 like pancakes
##  9 Q1       Question 1 pancakes because
## 10 Q1       Question 1 because the
## # ... with 2,077 more rows
```

```r
questions_bigrams %>%
  count(bigram, sort = TRUE) #this has many stop words, need to remove them
```

```
## # A tibble: 1,102 x 2
##    bigram               n
##    <chr>            <int>
##  1 i prefer            38
##  2 i like              30
##  3 me feel             30
##  4 eat pancakes        25
##  5 make me             24
##  6 in the              23
##  7 sweet pancakes      23
##  8 pancakes because    22
##  9 to eat              22
## 10 my favorite         21
## # ... with 1,092 more rows
```

```r
#to remove stop words from the bigram data, we need to use the separate function:
questions_separated <- questions_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

questions_filtered <- questions_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

#creating the new bigram, "no-stop-words":
questions_counts <- questions_filtered %>%
  count(word1, word2, sort = TRUE)
#want to see the new bigrams
questions_counts
```

```
## # A tibble: 104 x 3
##    word1    word2        n
##    <chr>    <chr>    <int>
##  1 eat      pancakes    25
##  2 sweet    pancakes    23
##  3 prefer   sweet       16
##  4 feel     happy       10
##  5 love     pancakes     9
##  6 favorite toppings     8
##  7 fluffy   pancakes     8
##  8 pancake  toppings     8
##  9 favorite pancake      7
## 10 thin     pancakes     7
## # ... with 94 more rows
```

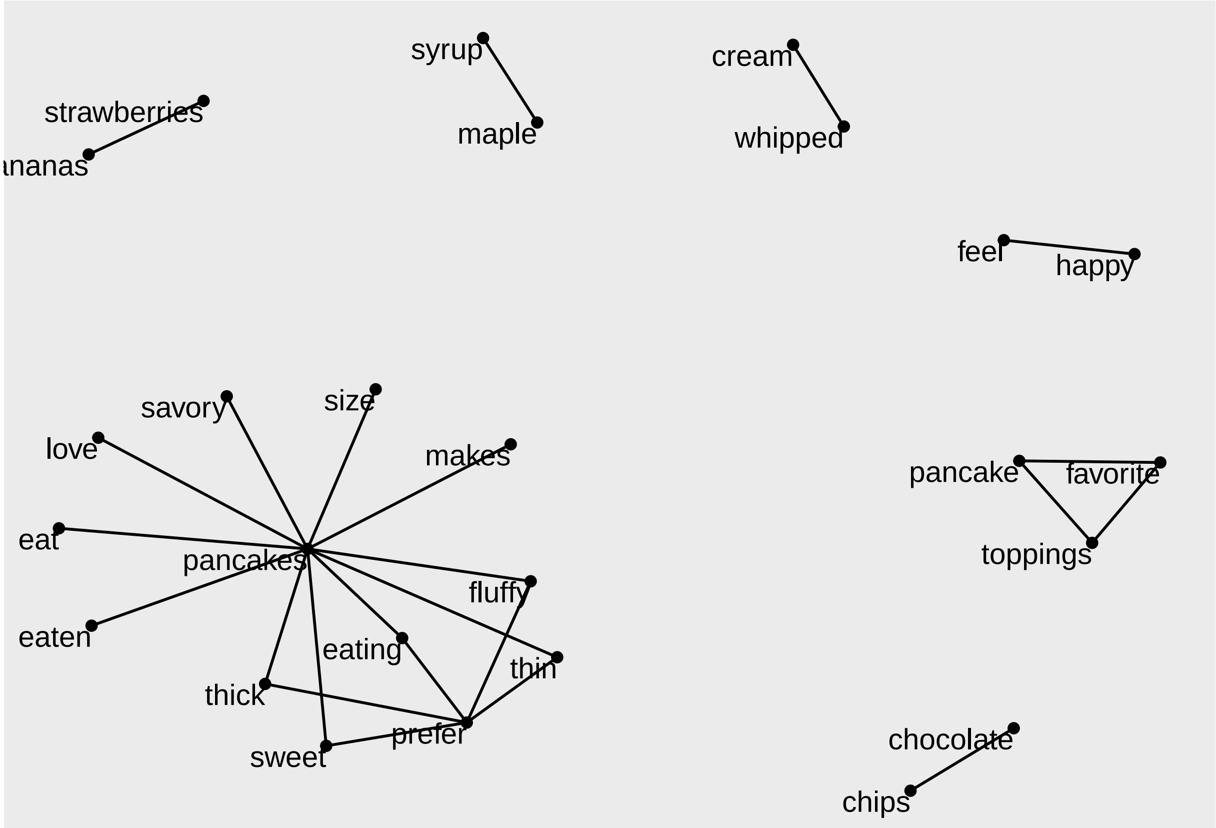
```
###########################################################
###### We can also apply the tf_idf framework  ############
################### on our bigram  ########################
###########################################################

questions_united <- questions_filtered %>%
  unite(bigram, word1, word2, sep=" ") #we need to unite what we split in the previous section

questions_bigram_tf_idf <- questions_united %>%
  count(author, bigram) %>%
  bind_tf_idf(bigram, author, n) %>%
  arrange(desc(tf_idf))

questions_bigram_tf_idf
```

```
## # A tibble: 111 x 6
##    author     bigram                 n    tf   idf tf_idf
##    <chr>      <chr>              <int> <dbl> <dbl>  <dbl>
##  1 Question 4 prefer sweet          16 0.327 1.95   0.635
##  2 Question 7 feel happy            10 0.323 1.95   0.628
##  3 Question 4 sweet pancakes        22 0.449 1.25   0.562
##  4 Question 1 love pancakes          7 0.438 1.25   0.548
##  5 Question 3 eat pancakes          16 0.64  0.847  0.542
##  6 Question 2 eat pancakes           8 0.4   0.847  0.339
##  7 Question 6 fluffy pancakes        8 0.174 1.95   0.338
##  8 Question 6 thin pancakes          7 0.152 1.95   0.296
##  9 Question 5 favorite toppings      8 0.138 1.95   0.268
## 10 Question 5 pancake toppings       8 0.138 1.95   0.268
## # ... with 101 more rows
```

```
######################################################
####### VISUALISING A BIGRAM NETWORK #################
######################################################

questions_bigram_graph <- questions_counts %>%
  filter(n>1) %>%
  graph_from_data_frame()

questions_bigram_graph
```

```
## IGRAPH 9e511f1 DN-- 26 24 --
## + attr: name (v/c), n (e/n)
## + edges from 9e511f1 (vertex names):
##  [1] eat         ->pancakes sweet       ->pancakes prefer      ->sweet
##  [4] feel        ->happy    love        ->pancakes favorite    ->toppings
##  [7] fluffy      ->pancakes pancake     ->toppings favorite    ->pancake
## [10] thin        ->pancakes prefer      ->fluffy   thick       ->pancakes
## [13] chocolate   ->chips    maple       ->syrup    eating      ->pancakes
## [16] prefer      ->eating   eaten       ->pancakes pancakes    ->makes
## [19] prefer      ->thick    prefer      ->thin     savory      ->pancakes
## [22] size        ->pancakes strawberries->bananas  whipped     ->cream
```

```
ggraph(questions_bigram_graph, layout = "fr") +
  geom_edge_link()+
  geom_node_point()+
  geom_node_text(aes(label=name), vjust =1, hjust=1)
```

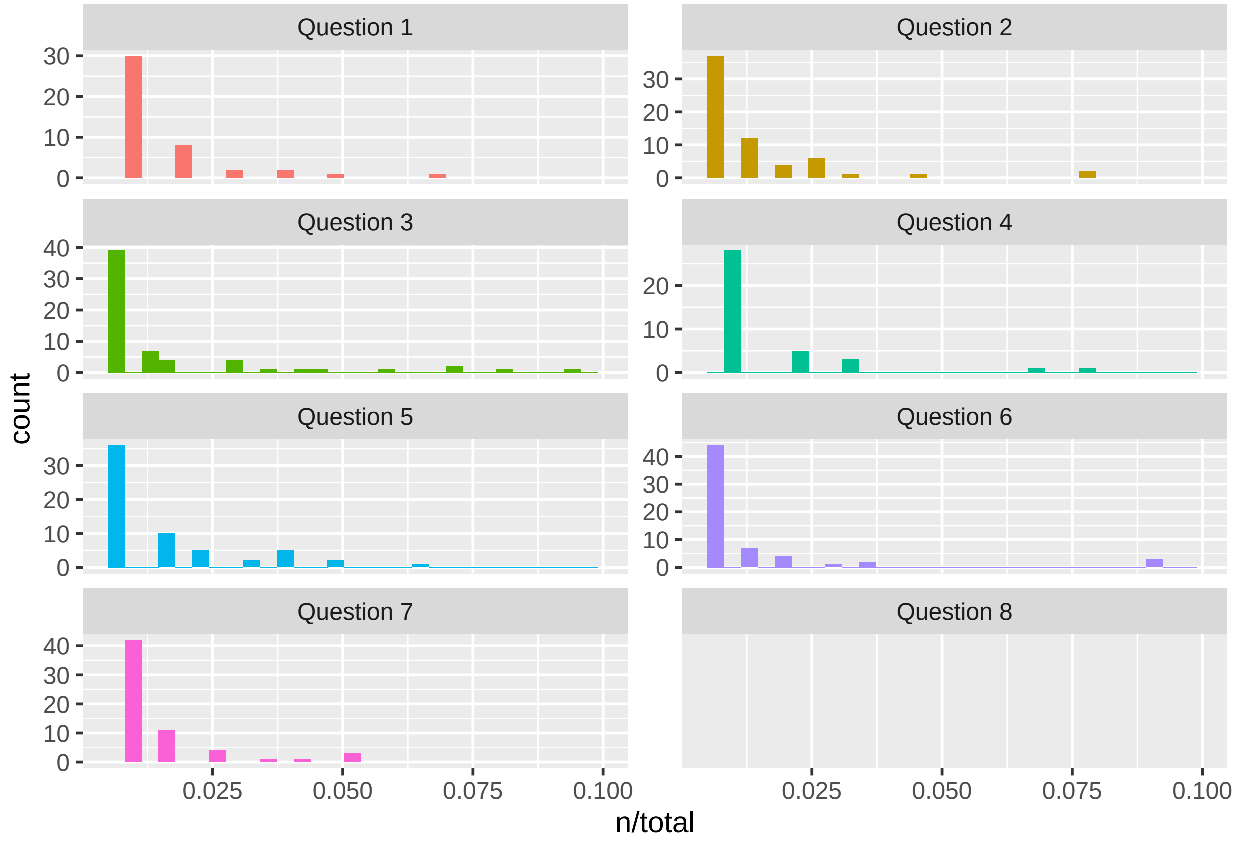


```
##########################################################
################ tf_idf Analysis#########################
##########################################################

tf_idf_questions <- bind_rows(mutate(Question_1, author = "Question 1"),
                              mutate(Question_2, author = "Question 2"),
                              mutate(Question_3, author = "Question 3"),
                              mutate(Question_4, author = "Question 4"),
                              mutate(Question_5, author = "Question 5"),
```

```r
                              mutate(Question_6, author = "Question 6"),
                              mutate(Question_7, author = "Question 7"),
                              mutate(Question_8, author = "Question 8")) %>%
  unnest_tokens(word, X1) %>%
  anti_join(stop_words2) %>%
  count(author, word, sort=TRUE) %>%
  ungroup()

total_words <- tf_idf_questions %>%
  group_by(author) %>%
  summarize(total=sum(n))

questions_words <- left_join(tf_idf_questions, total_words)

print(questions_words)
```

```
## # A tibble: 398 x 4
##    author     word           n total
##    <chr>      <chr>      <int> <int>
##  1 Question 4 sweet         30    90
##  2 Question 1 like          22   105
##  3 Question 6 like          20   143
##  4 Question 2 morning       19   152
##  5 Question 3 once          16   170
##  6 Question 3 maybe         14   170
##  7 Question 6 fluffy        13   143
##  8 Question 6 thick         13   143
##  9 Question 6 thin          13   143
## 10 Question 2 breakfast     12   152
## # ... with 388 more rows
```

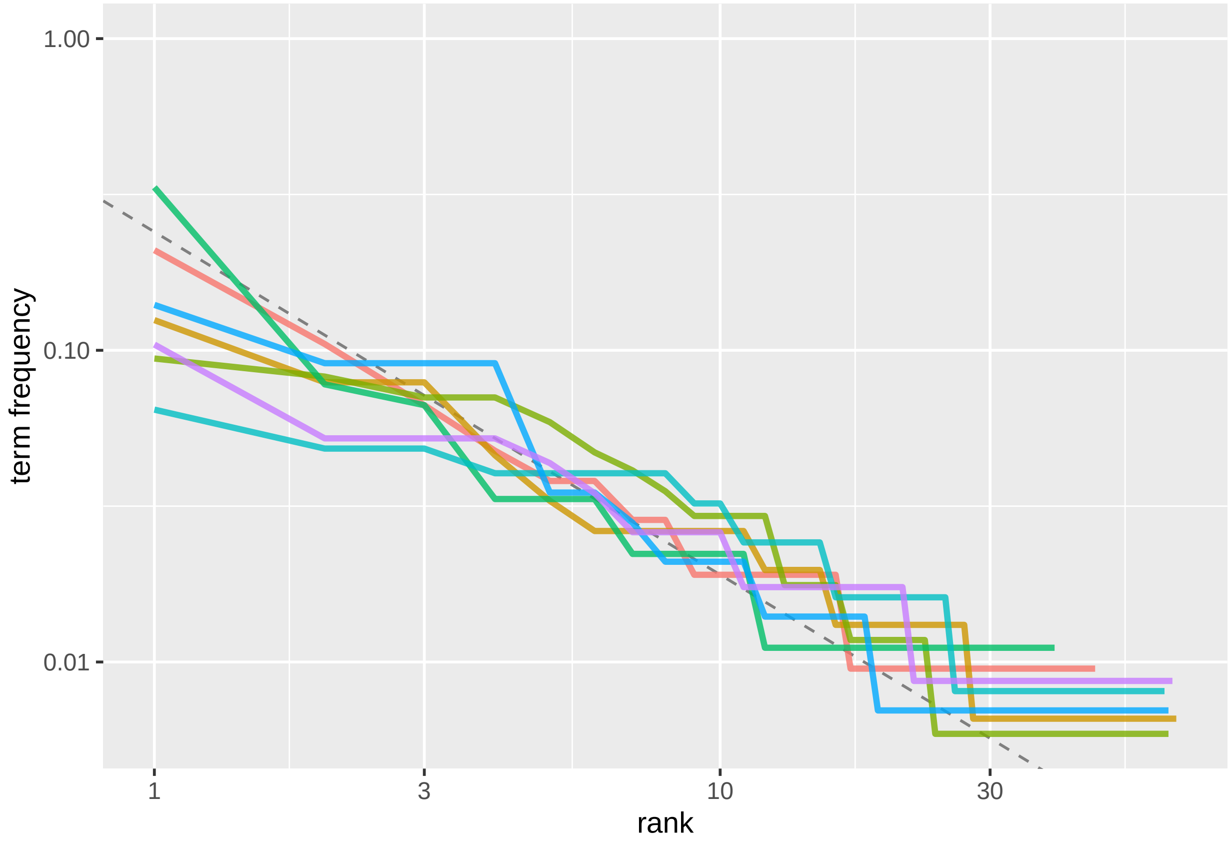
```r
ggplot(questions_words, aes(n/total, fill = author))+
  geom_histogram(show.legend=FALSE)+
  xlim(NA, 0.1) +
  facet_wrap(~author, ncol=2, scales="free_y")
```



```
#######################################
########## ZIPF's law ###############
#######################################

freq_by_rank <- questions_words %>%
  group_by(author) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total)
freq_by_rank
```

```
## # A tibble: 398 x 6
## # Groups:   author [8]
##    author     word          n total  rank `term frequency`
##    <chr>      <chr>     <int> <int> <int>            <dbl>
##  1 Question 4 sweet        30    90     1           0.333
##  2 Question 1 like         22   105     1           0.210
##  3 Question 6 like         20   143     1           0.140
##  4 Question 2 morning      19   152     1           0.125
##  5 Question 3 once         16   170     1           0.0941
##  6 Question 3 maybe        14   170     2           0.0824
##  7 Question 6 fluffy       13   143     2           0.0909
##  8 Question 6 thick        13   143     3           0.0909
##  9 Question 6 thin         13   143     4           0.0909
## 10 Question 2 breakfast    12   152     2           0.0789
## # ... with 388 more rows
```

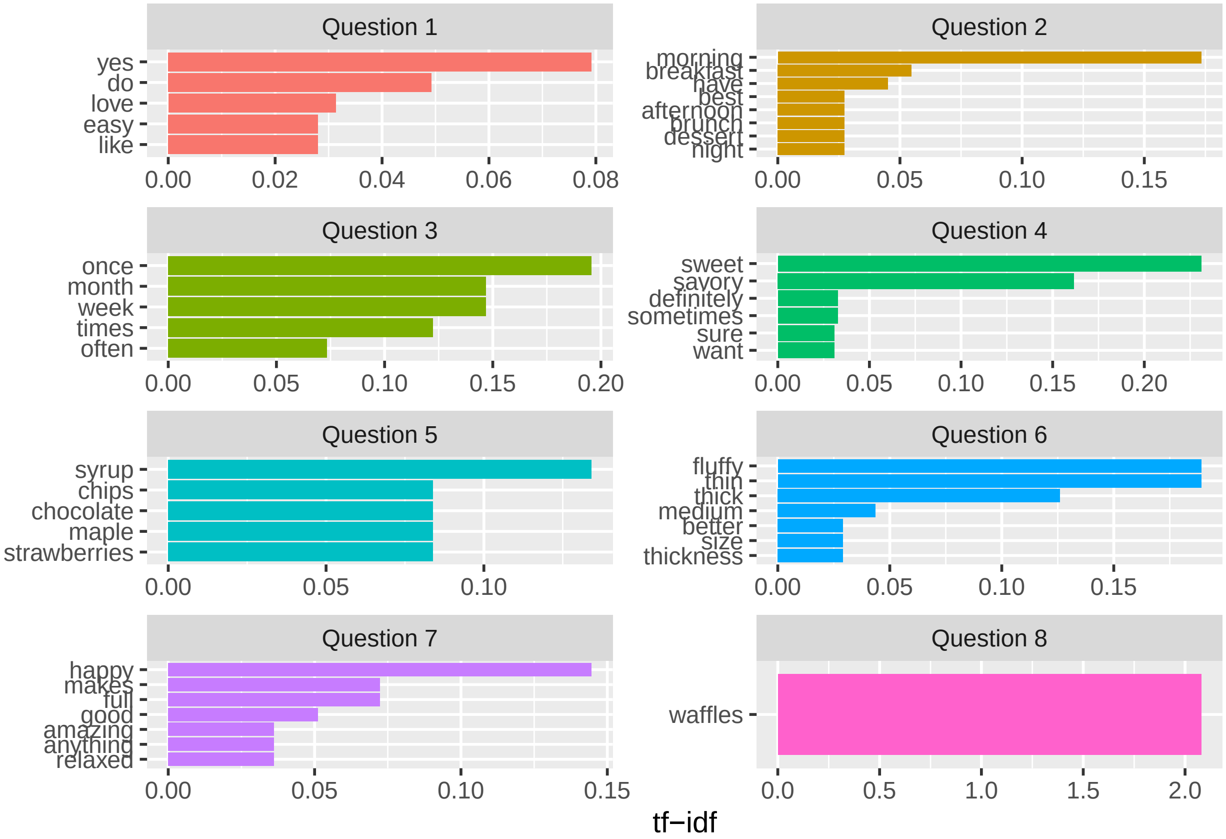
```
# plot ZIPF's Law
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color=author))+
  geom_abline(intercept=-0.62, slope= -1.1, color='gray50', linetype=2)+
  geom_line(size= 1.1, alpha = 0.8, show.legend = FALSE)+
  scale_x_log10()+
  scale_y_log10()
```



```
################################################
################# TF_IDF #########################
################################################
questions_words_idf <-questions_words %>%
  bind_tf_idf(word, author, n)

#reorganize the table
questions_words_idf %>%
  arrange(desc(tf_idf))
```

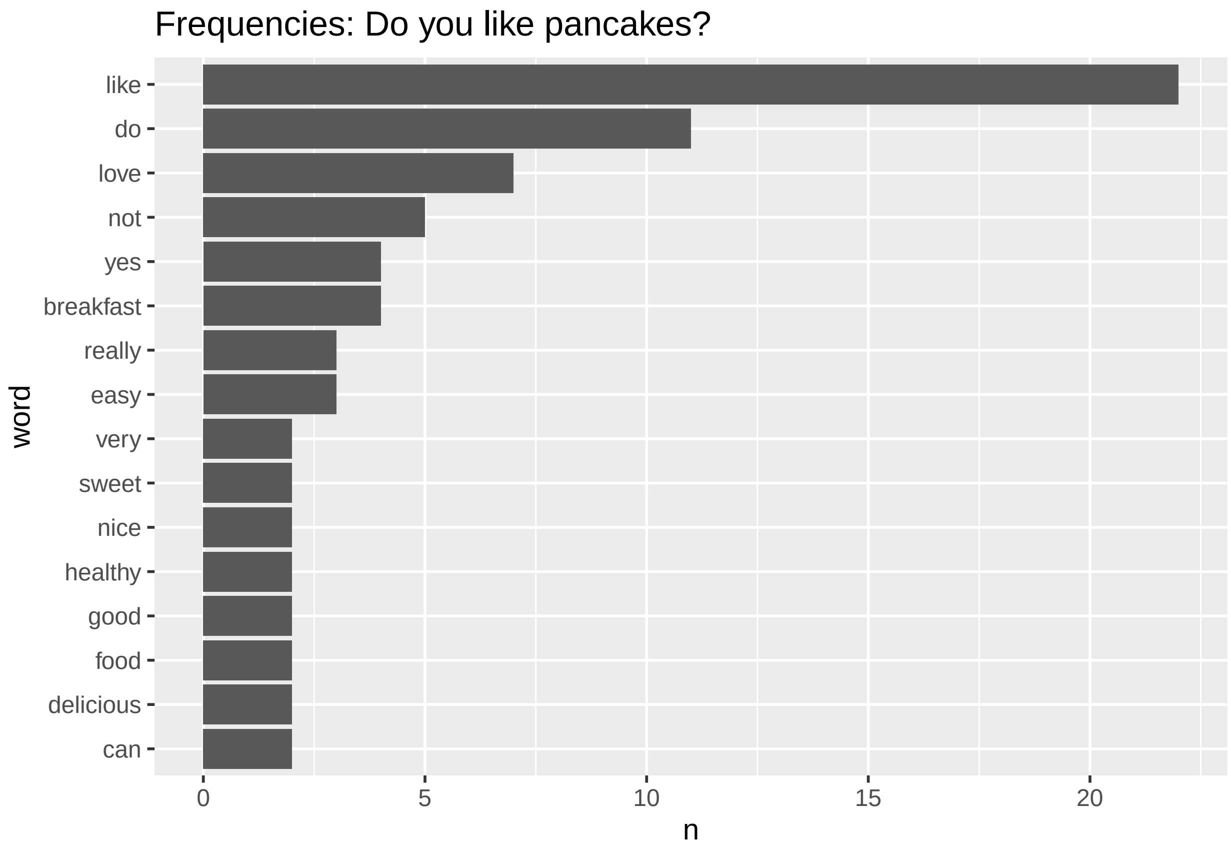
```
## # A tibble: 398 x 7
##    author     word         n total     tf   idf tf_idf
##    <chr>      <chr>    <int> <int>  <dbl> <dbl>  <dbl>
##  1 Question 8 waffles      8     8 1      2.08   2.08
##  2 Question 4 sweet       30    90 0.333  0.693  0.231
##  3 Question 3 once        16   170 0.0941 2.08   0.196
##  4 Question 6 fluffy      13   143 0.0909 2.08   0.189
##  5 Question 6 thin       13   143 0.0909 2.08   0.189
##  6 Question 2 morning    19   152 0.125  1.39   0.173
##  7 Question 4 savory      7    90 0.0778 2.08   0.162
##  8 Question 3 month      12   170 0.0706 2.08   0.147
##  9 Question 3 week       12   170 0.0706 2.08   0.147
## 10 Question 7 happy      12   115 0.104  1.39   0.145
## # ... with 388 more rows
```

```
#graphical approach
questions_words_idf %>%
  anti_join(stop_words2) %>%
  arrange(desc(tf_idf)) %>%
  mutate(word=factor(word, levels=rev(unique(word)))) %>%
  group_by(author) %>%
  top_n(5) %>% #top highest tfidf tokens
  ungroup %>%
  ggplot(aes(word, tf_idf, fill=author))+
  geom_col(show.legend=FALSE)+
  labs(x=NULL, y="tf-idf")+
  facet_wrap(~author, ncol=2, scales="free")+
  coord_flip()
```

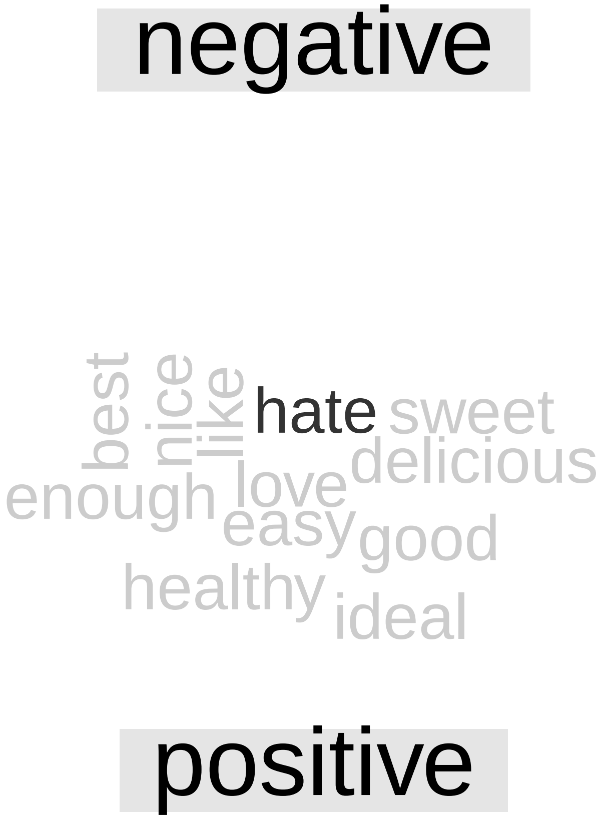


5.) Question 1: Do you like pancakes?

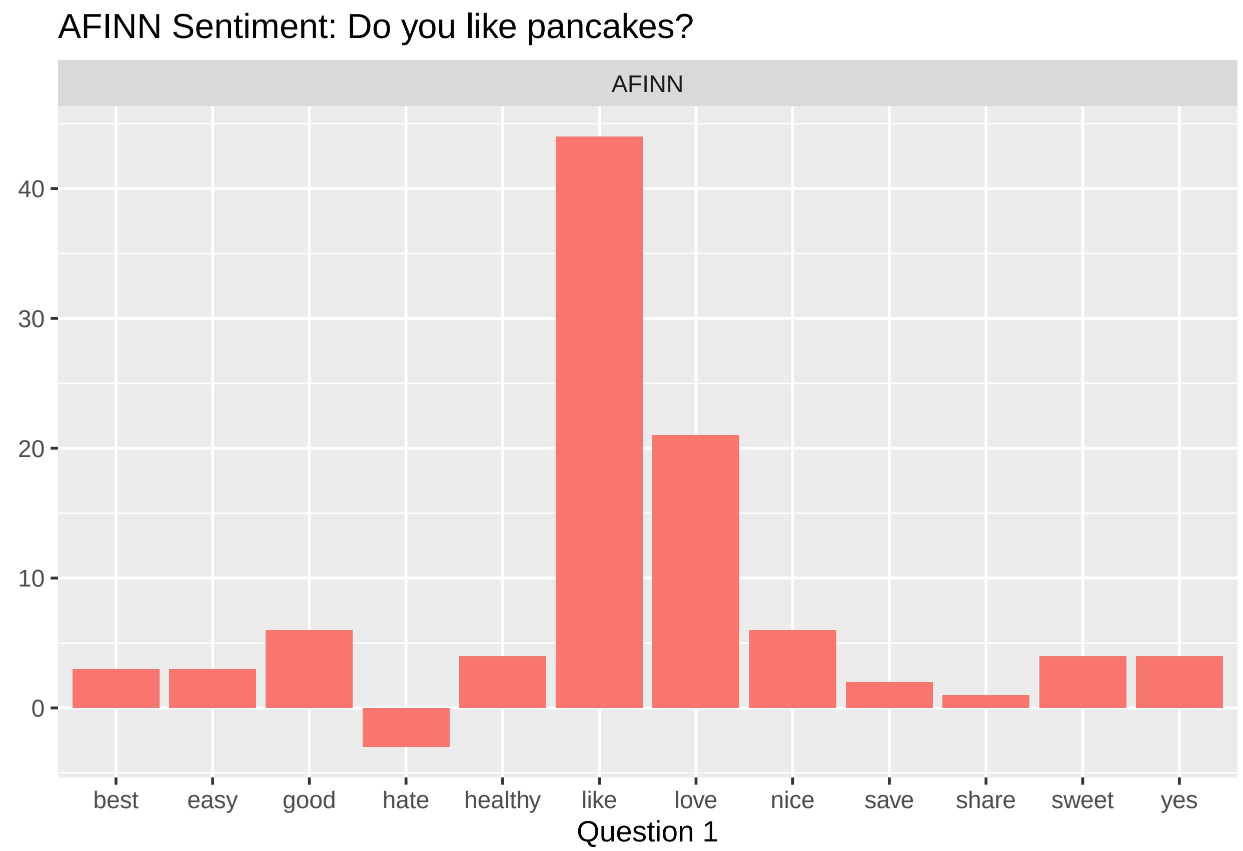
```
############################################
################ Question 1 ################
############################################

###########Question 1: Frequency##########
freq_Q1 <- tok_Q1 %>%
  count(word, sort=TRUE) %>%
  filter(n > 1) %>% # we need this to eliminate all the low count words
  mutate(word = reorder(word,n)) %>%
  ggplot(aes(word, n))+
  geom_col()+
  labs(title = "Frequencies: Do you like pancakes?")+
  coord_flip()
print(freq_Q1)
```



```
######Question 1: Bing Sentiment##########
bing_Q1 <- tok_Q1 %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var="n", fill=0) %>%
  comparison.cloud(colors = c("grey20", "grey80"),
                   max.words=500, scale=c(2, 2),
                   title.size=3)
```



```r
#######Question 1: Afinn Sentiment###########
afinn_Q1 <- tok_Q1 %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(word) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")

afinn_Q1_plot <- afinn_Q1 %>%
  ggplot(aes(word, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")+
  labs(x = "Question 1", y = NULL, title = "AFINN Sentiment: Do you like pancakes?")

print(afinn_Q1_plot )
```



```
#########Question 1: Bigrams############
Q1_bigrams <- survey_df %>%
  filter(question == "Q1") %>%
  unnest_tokens(bigram, X1, token = "ngrams", n=2)

Q1_bigrams #We want to see the bigrams (words that appear together, "pairs")
```

```
## # A tibble: 255 x 2
##    question bigram
##    <chr>    <chr>
##  1 Q1       i do
##  2 Q1       do like
##  3 Q1       like pancakes
##  4 Q1       i like
##  5 Q1       like pancakes
##  6 Q1       i don’t
##  7 Q1       don’t like
##  8 Q1       like pancakes
##  9 Q1       pancakes because
## 10 Q1       because the
## # ... with 245 more rows
```

```
Q1_bigrams %>%
  count(bigram, sort = TRUE) #this has many stop words, need to remove them
```

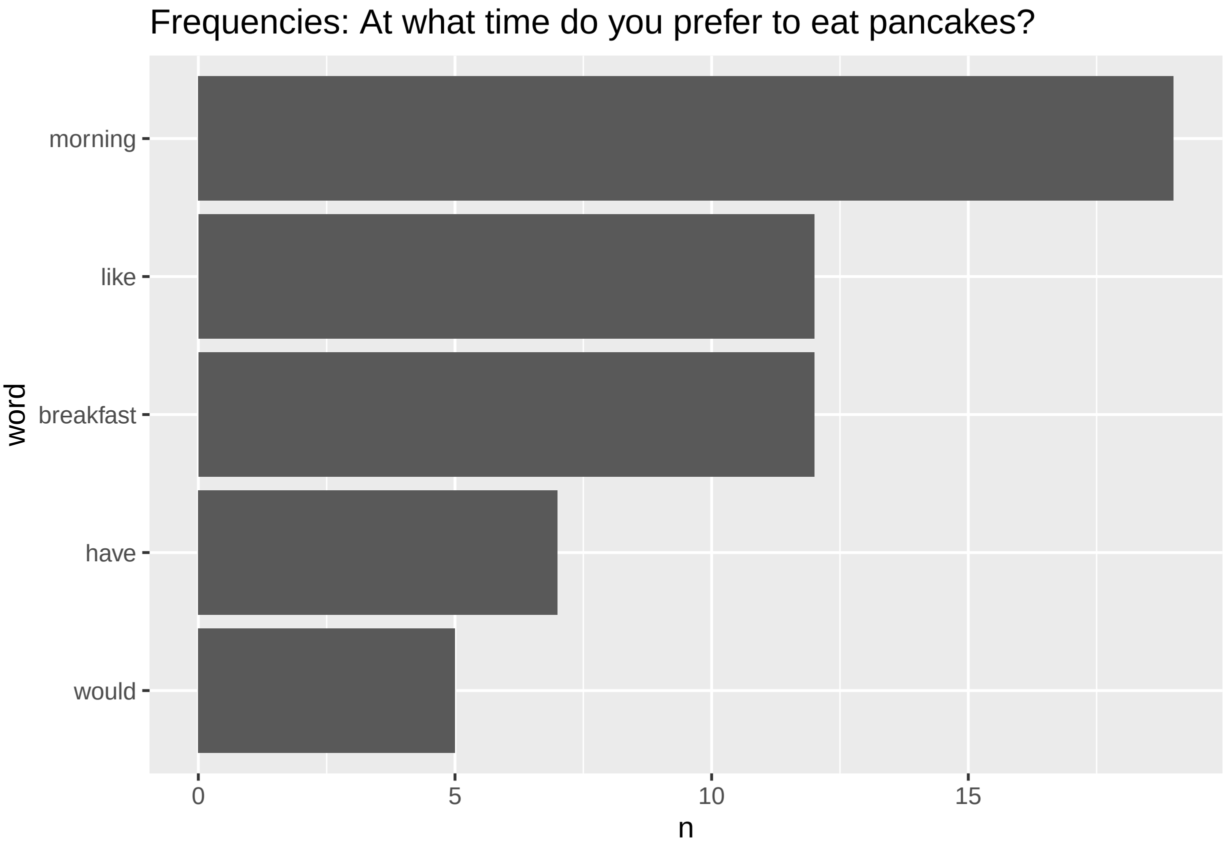
```
## # A tibble: 165 x 2
##    bigram               n
##    <chr>            <int>
##  1 like pancakes       18
##  2 i do                11
##  3 pancakes because    11
##  4 do like              9
##  5 i like               9
##  6 love pancakes        7
##  7 i love               6
##  8 because they         4
##  9 i don’t              4
## 10 they are             4
## # ... with 155 more rows
```

```
#to remove stop words from the bigram data, we need to use the separate function:
Q1_separated <- Q1_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

Q1_filtered <- Q1_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

#creating the new bigram, "no-stop-words":
Q1_counts <- Q1_filtered %>%
  count(word1, word2, sort = TRUE) %>%
  filter(n > 1)
#want to see the new bigrams
Q1_counts
```

```
## # A tibble: 1 x 3
##   word1 word2        n
##   <chr> <chr>    <int>
## 1 love  pancakes     7
```

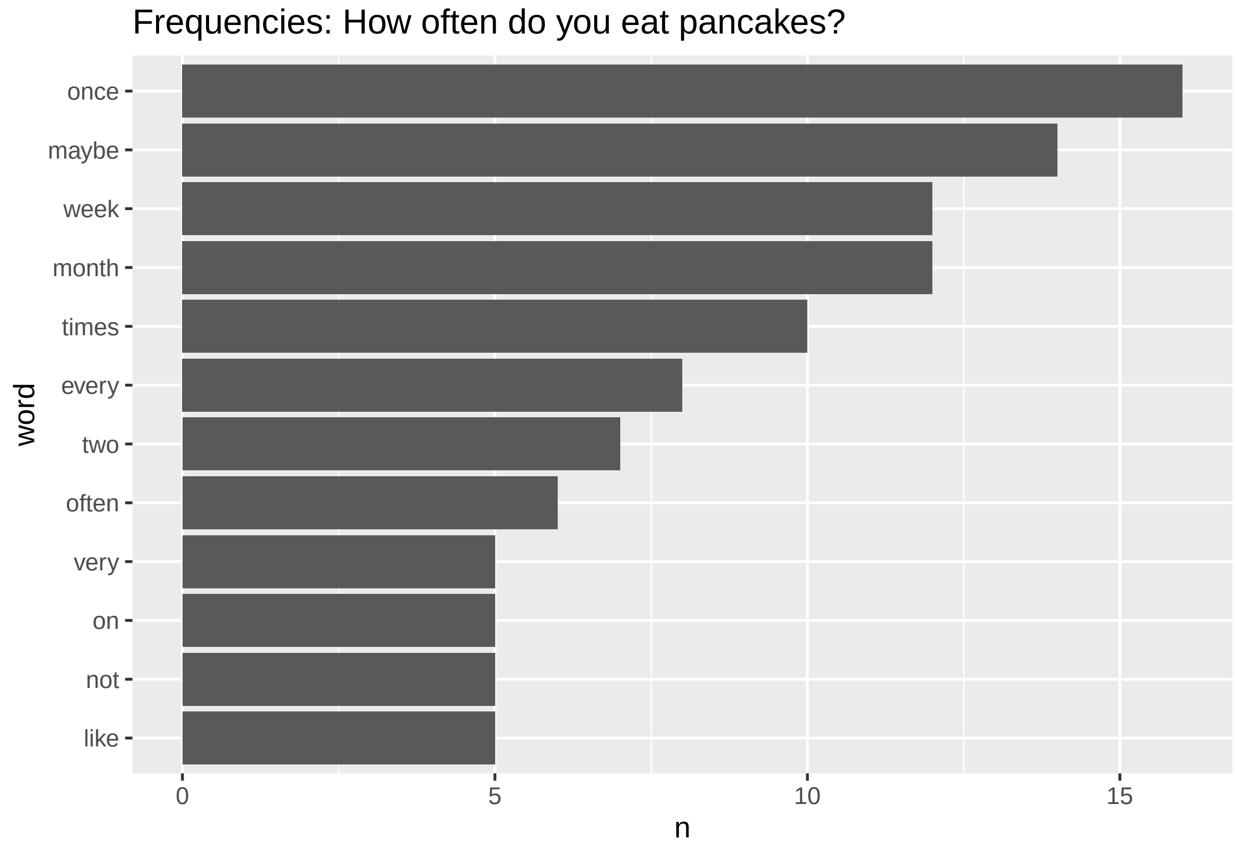
6.) Question 2: At what time do you prefer to eat pancakes?

```
#############################################
################ Question 2 ################
############################################

############Question 2: Frequency###########
freq_Q2 <- tok_Q2 %>%
  count(word, sort=TRUE) %>%
  filter(n > 4) %>% # we need this to eliminate all the low count words
  mutate(word = reorder(word,n)) %>%
  ggplot(aes(word, n))+
  geom_col()+
  labs(title = "Frequencies: At what time do you prefer to eat pancakes?")+
  coord_flip()
print(freq_Q2)
```



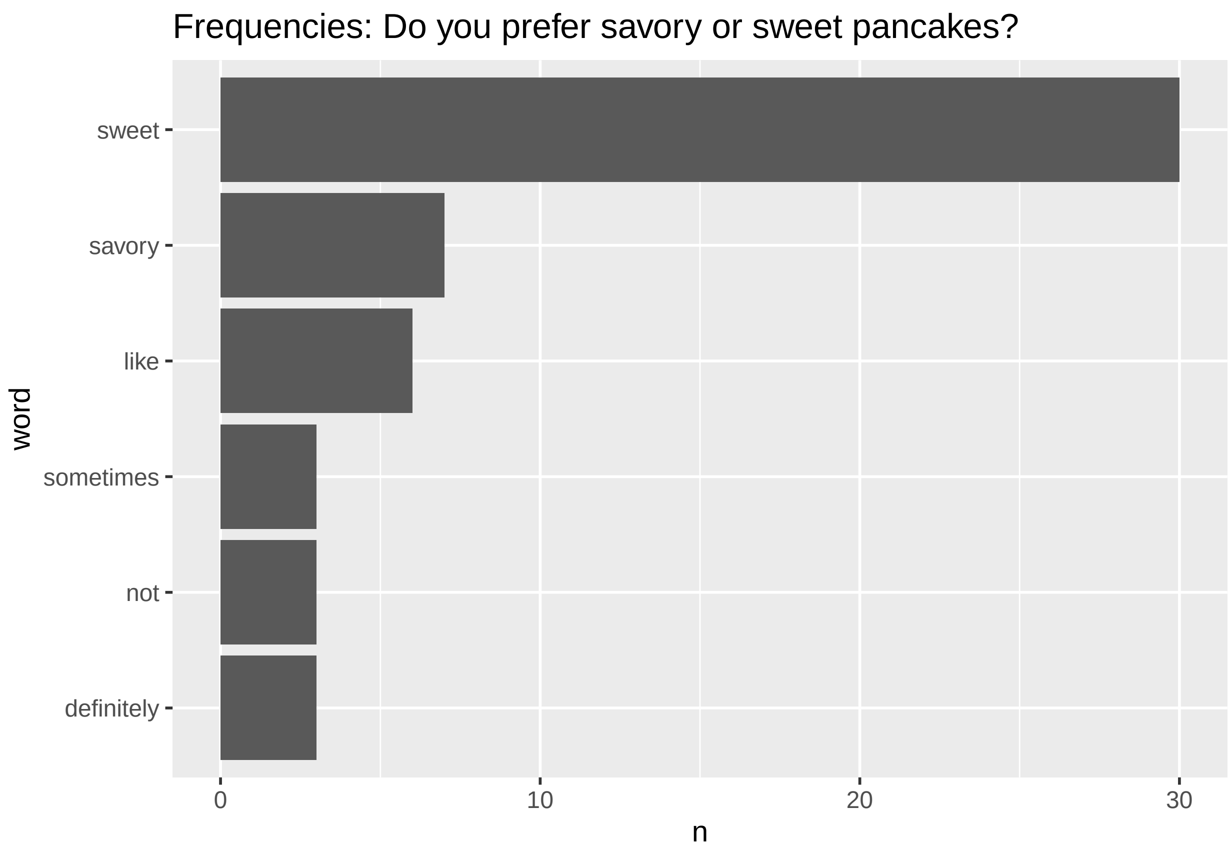
7.) Question 3: How often do you eat pancakes?

```
############################################
################ Question 3 ################
############################################

###########Question 3: Frequency##########
freq_Q3 <- tok_Q3 %>%
  count(word, sort=TRUE) %>%
  filter(n > 4) %>% # we need this to eliminate all the low count words
  mutate(word = reorder(word,n)) %>%
  ggplot(aes(word, n))+
  geom_col()+
  labs(title = "Frequencies: How often do you eat pancakes?")+
  coord_flip()
print(freq_Q3)
```



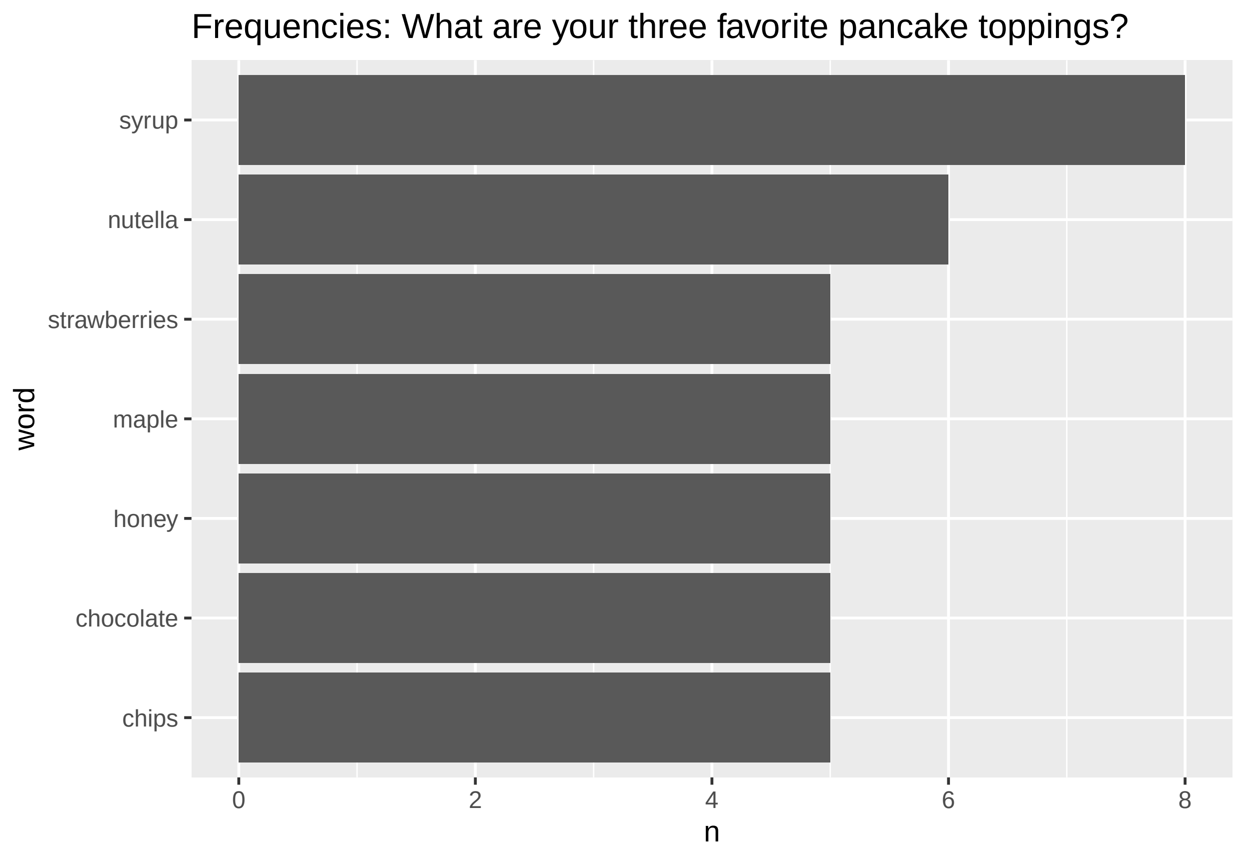
8.) Question 4: Do you prefer savory or sweet pancakes?

```
#############################################
################ Question 4 ################
#############################################

############Question 4: Frequency##########
freq_Q4 <- tok_Q4 %>%
  count(word, sort=TRUE) %>%
  filter(n > 2) %>% # we need this to eliminate all the low count words
  mutate(word = reorder(word,n)) %>%
  ggplot(aes(word, n))+
  geom_col()+
  labs(title = "Frequencies: Do you prefer savory or sweet pancakes?")+
  coord_flip()
print(freq_Q4)
```



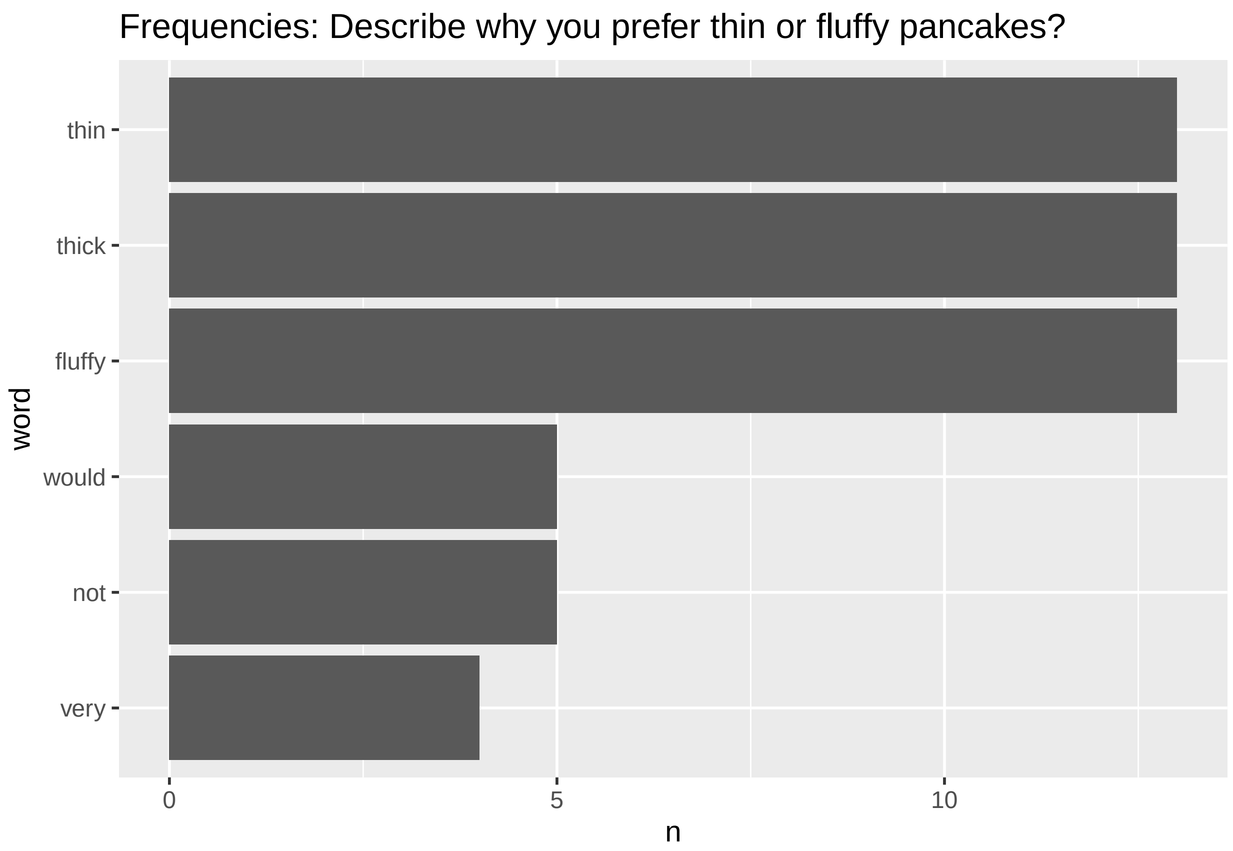
9.) Question 5: What are your three favorite pancake toppings?

```
############################################
################ Question 5 ################
############################################
custom_stop_words2 <- tribble(
  ~word, ~lexicon,
  "would", "CUSTOM"
)

###########Question 5: Frequency##########
freq_Q5 <- tok_Q5 %>%
  count(word, sort=TRUE) %>%
  anti_join(custom_stop_words2) %>%
  filter(n > 4) %>% # we need this to eliminate all the low count words
  mutate(word = reorder(word,n)) %>%
  ggplot(aes(word, n))+
  geom_col()+
  labs(title = "Frequencies: What are your three favorite pancake toppings?")+
  coord_flip()
print(freq_Q5)
```

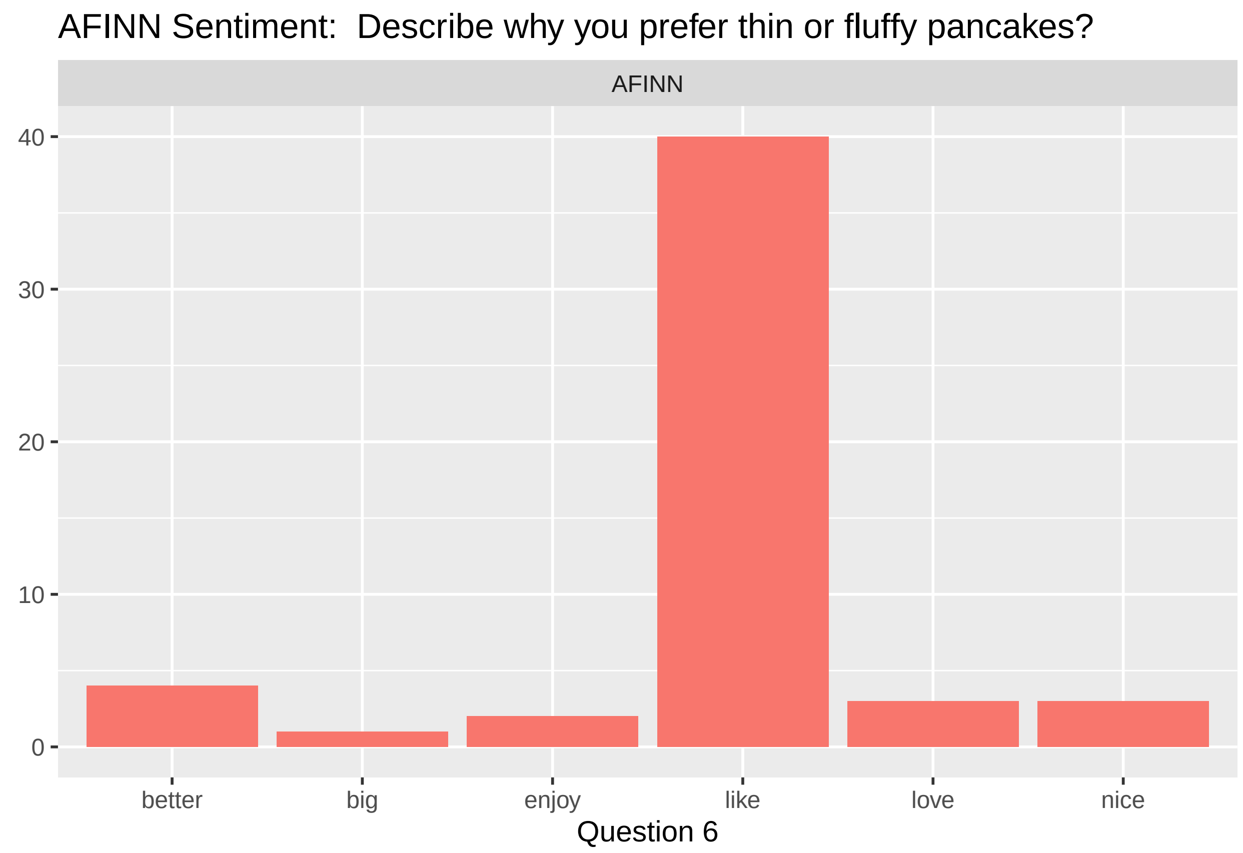


10.) Question 6: Describe why you prefer thin or fluffy pancakes?

```
##############################################
################ Question 6 ################
##############################################

###########Question 6: Frequency##########
custom_stop_words3 <- tribble(
  ~word, ~lexicon,
  "like", "CUSTOM"
)

freq_Q6 <- tok_Q6 %>%
  count(word, sort=TRUE) %>%
  anti_join(custom_stop_words3)%>%
  filter(n > 3) %>% # we need this to eliminate all the low count words
  mutate(word = reorder(word,n)) %>%
  ggplot(aes(word, n))+
  geom_col()+
  labs(title = "Frequencies: Describe why you prefer thin or fluffy pancakes?")+
  coord_flip()
print(freq_Q6)
```

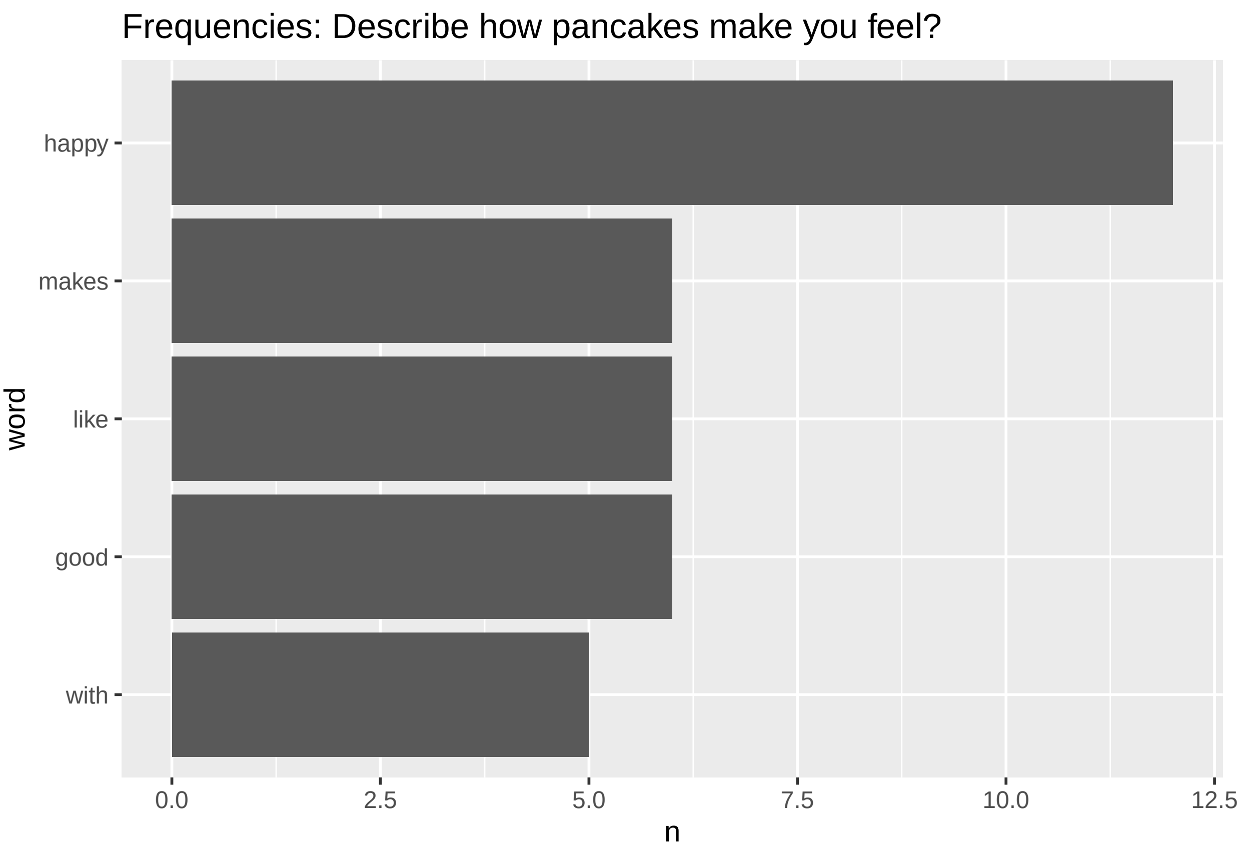


```r
######Question 6: Afinn Sentiment##########
afinn_Q6 <- tok_Q6 %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(word) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")

afinn_Q6_plot <- afinn_Q6 %>%
  ggplot(aes(word, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")+
  labs(x = "Question 6", y = NULL, title = "AFINN Sentiment:  Describe why you prefer thin or fluffy pa

print(afinn_Q6_plot )
```

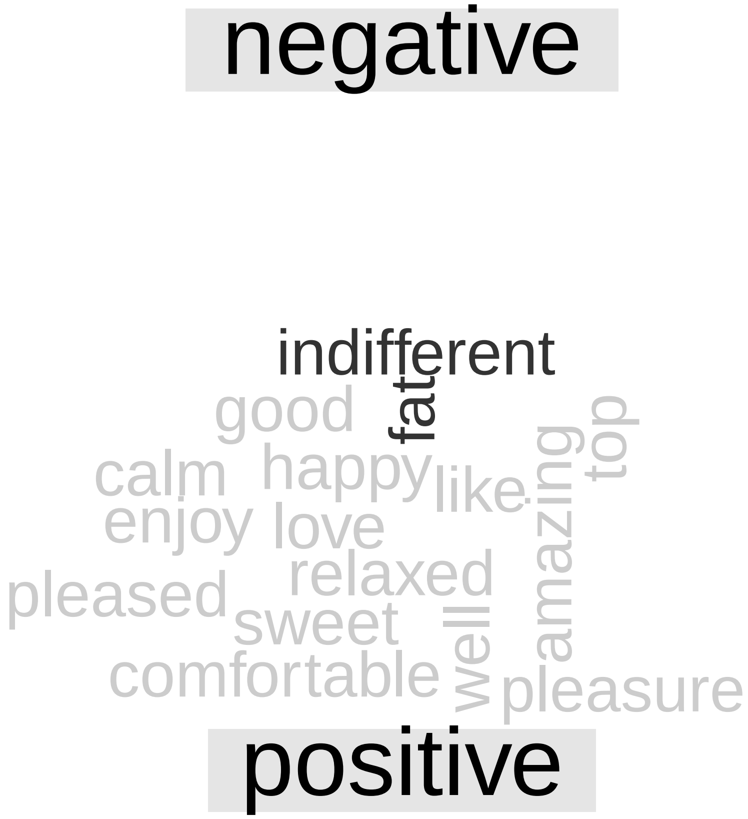


11.) Question 7: Describe how pancakes make you feel?

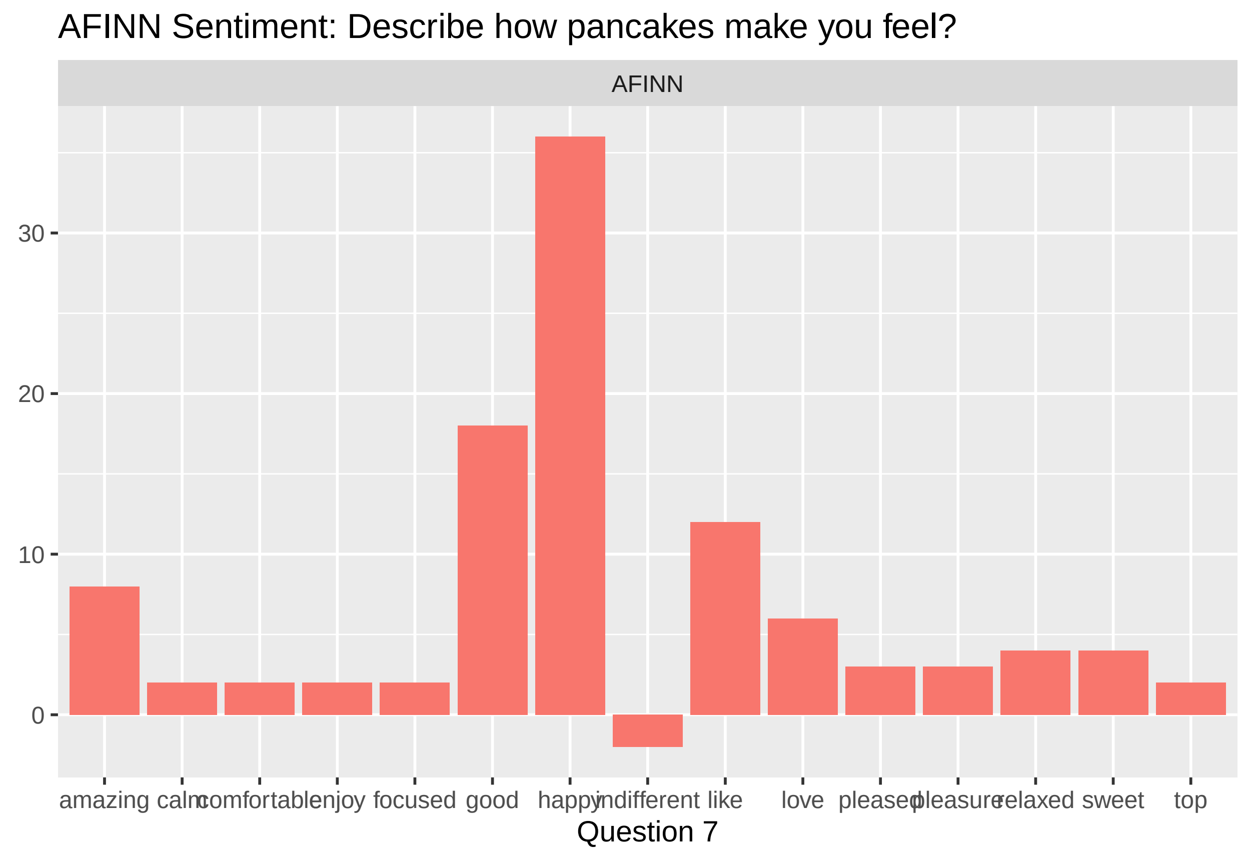
```r
############################################
################ Question 7 ################
############################################

############Question 7: Frequency###########
freq_Q7 <- tok_Q7 %>%
  count(word, sort=TRUE) %>%
  filter(n > 4) %>% # we need this to eliminate all the low count words
  mutate(word = reorder(word,n)) %>%
  ggplot(aes(word, n))+
  geom_col()+
  labs(title = "Frequencies: Describe how pancakes make you feel?")+
  coord_flip()
print(freq_Q7)
```



```
###########Question 7: Bing Sentiment ###########
bing_Q7 <- tok_Q7 %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var="n", fill=0) %>%
  comparison.cloud(colors = c("grey20", "grey80"),
                   max.words=500, scale=c(2, 2),
                   title.size=3)
```

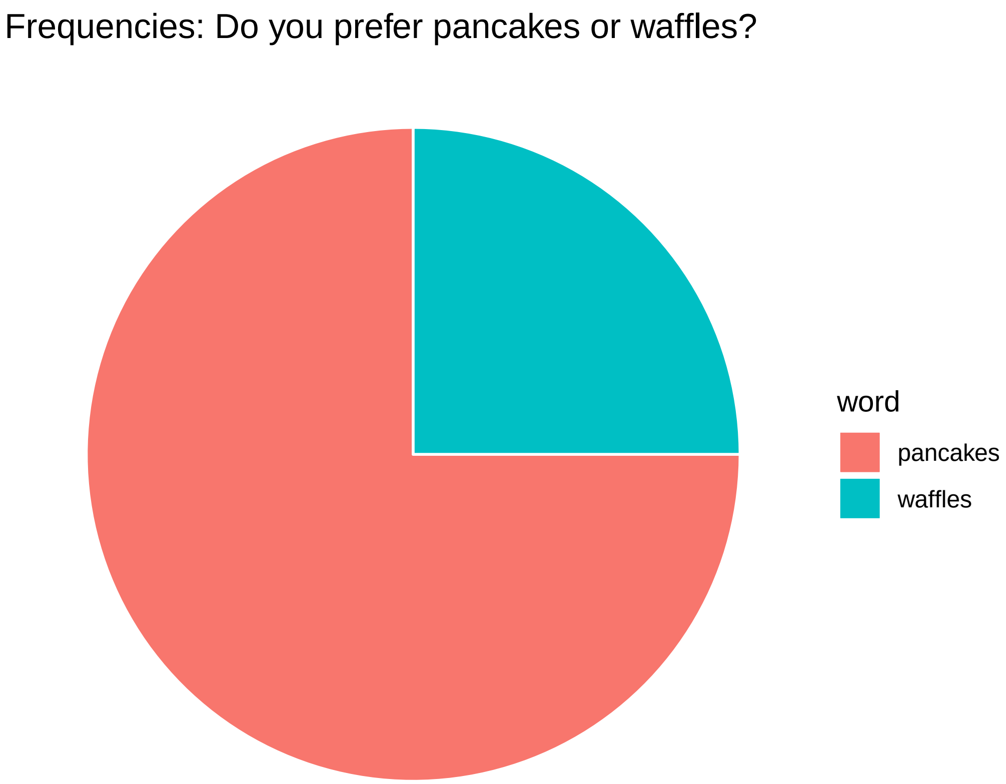


```r
#######Question 7: Afinn Sentiment###########
afinn_Q7 <- tok_Q7 %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(word) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")

afinn_Q7_plot <- afinn_Q7 %>%
  ggplot(aes(word, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")+
  labs(x = "Question 7", y = NULL, title = "AFINN Sentiment: Describe how pancakes make you feel?")

print(afinn_Q7_plot )
```



12.) Question 8: Do you prefer pancakes or waffles?

```
############################################
################ Question 8 ################
############################################

#######Question 8: Frequency###########
freq_Q8 <- tok_Q8 %>%
  count(word, sort=TRUE) # we need this to eliminate all the low count words


plot_Q8 <- ggplot(freq_Q8, aes(x="", y=n, fill=word))+
  geom_bar(stat="identity", width = 1, color="white")+
  labs(title = "Frequencies: Do you prefer pancakes or waffles?")+
  coord_polar("y", start = 0) +
  theme_void()
print(plot_Q8)
```



13.) Shiny App

```
total_counts <- rbind.data.frame(tok_Q1 %>%
                                     count(word, sort=TRUE) %>%
                                     mutate(question = "Question 1"),
                                 tok_Q2 %>%
                                     count(word, sort=TRUE) %>%
                                     mutate(question = "Question 2"),
                                 tok_Q3 %>%
                                     count(word, sort=TRUE) %>%
                                     mutate(question = "Question 3"),
                                 tok_Q4 %>%
                                     count(word, sort=TRUE) %>%
                                     mutate(question = "Question 4"),
                                 tok_Q5 %>%
                                     count(word, sort=TRUE) %>%
                                     mutate(question = "Question 5"),
                                 tok_Q6 %>%
                                     count(word, sort=TRUE) %>%
                                     mutate(question = "Question 6"),
                                 tok_Q7 %>%
                                     count(word, sort=TRUE) %>%
                                     mutate(question = "Question 7"),
                                 tok_Q8 %>%
                                     count(word, sort=TRUE) %>%
                                     mutate(question = "Question 8"))

questionsapp <- c("1. Do you like pancakes ?",
               "2. At what time do you prefer to eat pancakes? Why?",
               "3. How often do you eat pancakes ?",
               "4. Do you prefer savory or sweet pancakes?",
               "5. What are your three favorite pancake toppings",
               "6. Describe why you prefer thin or fluffy pancakes?",
               "7. Describe how pancakes make you feel.",
               "8. Do you prefer pancakes or waffles?")

questionsapp_df <- cbind.data.frame(questionsapp, c("Question 1", "Question 2",
                                                    "Question 3", "Question 4",
                                                    "Question 5", "Question 6",
                                                    "Question 7", "Question 8"))
names(questionsapp_df)[1] <- "name"
names(questionsapp_df)[2] <- "number"

########################################
##########UI for application###########
########################################

ui <- fluidPage(
  theme = shinythemes:: shinytheme("superhero"),
  img(src = "pancakes.png", align = "left",height=’80px’,width=’80px’),
  tags$h1(strong("Aunt Serena’s Pancakes")),
  br(),
  tags$h5(em(h5("Aunt Serena is a brand that produces pancake mix, syrup, and other breakfast foods, ba
  br(),"A survey was conducted to gather insight on flavour of pancake preferences from MsBA Hult studen
  br(),
    sidebarLayout(
        sidebarPanel(
            selectInput("questionchoice",tags$h5(strong("1. Which survey question you want to explore?")
            sliderInput(’n_words’, tags$h5(strong("2. Select the number of words you want to see")),  mi
        ),#closing the sidebarPanel
        mainPanel(tags$h4(textOutput("questiontitle")),
            tabsetPanel(
                tabPanel("Token Frequency Bar Plot", plotlyOutput("frequency"),
                         br(),
                         br(),
                         conditionalPanel(condition="input.questionchoice==’Question 1’",
                                          p(tags$h5("AFINN Sentiment Analysis")), plotlyOutput("afinn_q
                         conditionalPanel(condition="input.questionchoice==’Question 7’",
                                          p(tags$h5("AFINN Sentiment Analysis")), plotlyOutput("afinn_q
                         conditionalPanel(condition="input.questionchoice==’Question 8’",
                                          p(tags$h5("Token Proportion",br(),br(), plotlyOutput("pie_q8")
                ),#closing tabPanel
                tabPanel(
                  "TFIDF Bar Plot", plotlyOutput("tfidf")
                ),#closing tabPanel
                tabPanel(
                  "Insights & Recommendation",
                  br(),
                  strong(em("Insights:",br(),br())),
                  tags$div(
                    tags$ul(
                      tags$li("There is a notable positive association with pancakes amongst respondents
                      tags$li("The majority of respondents prefer to eat pancakes for breakfast/morning
                      tags$li("Pancakes are not an everyday food for the majority of respondents. People
                      tags$li("The vast majority of respondents prefer sweet over savoury flavours coupl
                      tags$li("Sweet toppings are dominating: syrup, chocolate and fruits being the crow
                      tags$li("A small majority prefers fluffy and thick pancakes over thin, but this is
                      tags$li("3/4 of respondents prefer pancakes over waffles",br(),br())
                    )
                  ),
                  br(),
                  br(),
                  br(),
                  strong(em("Next Steps:",br(),br())),
                  tags$ul(
                    tags$li("Deals with supermarkets/grocery stores: Place pancake mixes next to cereals
                    tags$li("A range of premium flavoured mixes to capitalise on popular flavours: maple
                    tags$li("Potential to sell a more premium version of our product ,  as this isn't an
                    tags$li("Diversify our product lines within the sweet category; with a mixture of sw
                    tags$li("Create a waffle mix; low R&D costs because of high similarity to appeal to
                    tags$li("Increasing marketing across social media platforms of other topping possibi
                    tags$li("On product packaging and on social media: clearly state the two distinctive
                  )
                )#closing tabPanel
            )#closing tabsetPanel
        )#closing mainPanel
    )#closing the sidebarLayout
)#closing fluidPage

########################################
###############Server#################
########################################

server <- function(input, output) {

    rval_questiontitle <- reactive({
        questionsapp_df %>%
            filter(number == input$questionchoice)
    })

    output$questiontitle <- renderText({
        paste(rval_questiontitle()[1])
    })


    # filter with a reactive expression
    rval_question <- reactive({
        total_counts %>%
            filter(question == input$questionchoice) %>%
            arrange(desc(n)) %>%
            head(input$n_words)
    })

    rval_tf_idf <- reactive({
        questions_words_idf %>%
            filter(author == input$questionchoice) %>%
            arrange(desc(tf_idf)) %>%
            head(input$n_words)
    })



    #  Render a text output, greeting
    output$frequency <- renderPlotly({
        rval_question() %>%
        filter(n > 1) %>%
        mutate(word = reorder(word,n, fill=n))%>%
        ggplot(aes(word, n))+
                geom_col(fill="steelblue")+
                labs(x=NULL, y="n", title = "Word Frequencies")+
                theme(text = element_text(family = "Helvetica", size=10, colour="black", face = "bold")
                theme(plot.title = element_text(hjust = 0.5))+
                theme(axis.title.y = element_text(family = "Helvetica", size=9.5, colour="black", face =
                theme(axis.title.x = element_text(family = "Helvetica", size=9.5, colour="black", face =
                scale_y_continuous(expand = expansion(mult = c(0, 0.1)))+
                coord_flip()}
    )#closing frequency

    output$tfidf <- renderPlotly({
        rval_tf_idf() %>%
        filter(word > 1) %>%
        mutate(word = reorder(word, tf_idf))%>%
        ggplot(aes(word, tf_idf))+
                geom_col(fill="steelblue", show.legend=FALSE)+
                labs(x=NULL, y="tf-idf")+
                facet_wrap(~author, ncol=2, scales="free")+
                theme(text = element_text(family = "Helvetica", size=10.5, colour="black", face = "bold
                theme(plot.title = element_text(hjust = 0.5))+
                theme(axis.title.y = element_text(family = "Helvetica", size=9.5, colour="black", face =
                theme(axis.title.x = element_text(family = "Helvetica", size=9.5, colour="black", face =
                scale_y_continuous(expand = expansion(mult = c(0, 1)))+
                coord_flip()}
    )#closing frequency


    output$afinn_q1 <- renderPlotly({
      ggplot(afinn_Q1_plot, aes(word, sentiment)) +
        geom_col(fill="steelblue", show.legend = FALSE) +
        facet_wrap(~method, ncol = 1, scales = "free_y")+
        labs(x = NULL, y = "Sentiment Score")+
        theme(text = element_text(family = "Helvetica", size=10.5, colour="black", face = "bold"))+
        theme(plot.title = element_text(hjust = 0.5))+
        theme(axis.title.y = element_text(family = "Helvetica", size=9.5, colour="black", face = "bold"
        theme(axis.title.x = element_text(family = "Helvetica", size=9.5, colour="black", face = "bold"
        scale_y_continuous(expand = expansion(mult = c(0, 0.3)))+
        coord_flip()
    })


    output$afinn_q7 <- renderPlotly({
      ggplot(afinn_Q7_plot, aes(word, sentiment)) +
        geom_col(fill="steelblue", show.legend = FALSE) +
        facet_wrap(~method, ncol = 1, scales = "free_y")+
        labs(x = NULL, y = "Sentiment Score")+
        theme(text = element_text(family = "Helvetica", size=10.5, colour="black", face = "bold"))+
        theme(plot.title = element_text(hjust = 0.5))+
        theme(axis.title.y = element_text(family = "Helvetica", size=9.5, colour="black", face = "bold"
        theme(axis.title.x = element_text(family = "Helvetica", size=9.5, colour="black", face = "bold"
        scale_y_continuous(expand = expansion(mult = c(0, 0.3)))+
        coord_flip()
    })

    output$pie_q8 <- renderPlotly({
      plot_ly(
        pie_Q8_plot,
        labels=~word,
        values=~n,
        type = "pie",
        marker = list(colors = colors)) %>%
      layout(title = " <b>Token Counts: Do you prefer pancakes or waffles?</b>", font=plotly_layout)
    })


}#closing server

# Run the application
shinyApp(ui = ui, server = server)
```

Shiny applications not supported in static R Markdown documents