```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

```
## Warning: package 'purrr' was built under R version 4.0.5
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
## Warning: package 'stringr' was built under R version 4.0.5
```

```
## Warning: package 'forcats' was built under R version 4.0.5
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(sqldf)
```

```
## Warning: package 'sqldf' was built under R version 4.0.5
```

```
## Loading required package: gsubfn
```

```
## Warning: package 'gsubfn' was built under R version 4.0.5
```

```
## Loading required package: proto
```

```
## Warning: package 'proto' was built under R version 4.0.5
```

```
## Loading required package: RSQLite
```

```
## Warning: package 'RSQLite' was built under R version 4.0.5
```

```r
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.0.5
```

```r
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 4.0.5
```

```
## Loading required package: RColorBrewer
```

```
## Warning: package 'RColorBrewer' was built under R version 4.0.3
```

```r
library(textdata) #for Afinn library
```

```
## Warning: package 'textdata' was built under R version 4.0.5
```

```r
library(topicmodels)
```

```
## Warning: package 'topicmodels' was built under R version 4.0.5
```

```r
Sys.setenv(TZ= "Europe/Warsaw")
Sys.getenv("TZ")
```

```
## [1] "Europe/Warsaw"
```

```r
#as.POSIXct(t, tz=getOption("TZ"))
Data <- read.csv("Data_april.csv")
Data_tibble <- read_csv("Data_april.csv",locale=locale(tz="Europe/Warsaw"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   conversationId = col_character(),
##   timeL = col_double(),
##   text = col_character(),
##   sentBy = col_character(),
##   prev_text = col_character(),
##   prev_sentBy = col_character(),
##   escalation = col_double(),
##   totalCharacters = col_double(),
##   rank = col_double()
## )
```

```r
Total_user = as.numeric(Data_tibble %>%
  filter(sentBy == "Consumer") %>%
  count())

sprintf("Total messages sent by consumer = %i",Total_user)
```

```
## [1] "Total messages sent by consumer = 25871"
```

```
Total_agent = as.numeric(Data_tibble %>%
  filter(sentBy == "Agent") %>%
  count())

sprintf("Total messages sent by agent = %i",Total_agent)
```

```
## [1] "Total messages sent by agent = 55464"
```

```
TAB = as.numeric(Data_tibble %>%
  filter(sentBy == "Agent", escalation == "0") %>%
  count())
sprintf("Total messages sent by agent before escalation = %i",TAB)
```

```
## [1] "Total messages sent by agent before escalation = 35072"
```

```
TAA = as.numeric(Data_tibble %>%
  filter(sentBy == "Agent", escalation == "1") %>%
  count())
sprintf("Total messages sent by agent after escalation = %i",TAA)
```

```
## [1] "Total messages sent by agent after escalation = 20392"
```

```
TAB = as.numeric(Data_tibble %>%
  filter(sentBy == "Consumer", escalation == "0") %>%
  count())
sprintf("Total messages sent by consumer before escalation = %i",TAB)
```

```
## [1] "Total messages sent by consumer before escalation = 17836"
```

```
TAA = as.numeric(Data_tibble %>%
  filter(sentBy == "Consumer", escalation == "1") %>%
  count())
sprintf("Total messages sent by consumer after escalation = %i",TAA)
```

```
## [1] "Total messages sent by consumer after escalation = 8035"
```

```
#Average no. of msgs shared by agent per each unique conversation"

Data_tibble %>%
  group_by(conversationId)%>%
  filter(sentBy == "Agent") %>%
  summarize(max_ranks = max(rank)) %>%
  summarize(mean_avg = mean(max_ranks))
```

```
## # A tibble: 1 x 1
##   mean_avg
##      <dbl>
## 1     16.5
```

```r
#Average no. of msgs shared by agent per each unique conversation before escalation was made"

Data_tibble %>%
  group_by(conversationId)%>%
  filter(sentBy == "Agent", escalation== "0") %>%
  summarize(max_ranks = max(rank)) %>%
  summarize(mean_avg = mean(max_ranks))
```

```
## # A tibble: 1 x 1
##   mean_avg
##      <dbl>
## 1     10.2
```

```r
#as escalation = 1, ranks of messages where escalation was initially 0 will already be considered.
#Average no. of msgs shared by agent per each unique conversation in which an escalation was made"

Data_tibble %>%
  group_by(conversationId)%>% #To divide in unique conversations
  filter(sentBy == "Agent", escalation== "1") %>%
  summarize(max_ranks = max(rank)) %>%
  summarize(mean_avg = mean(max_ranks))
```

```
## # A tibble: 1 x 1
##   mean_avg
##      <dbl>
## 1     20.1
```

```r
#Average no. of msgs shared by consumer per each unique conversation in which an escalation was made"
Data_tibble %>%
  group_by(conversationId)%>% #To divide in unique conversations
  filter(sentBy == "Consumer", escalation== "1") %>%
  summarize(max_ranks = max(rank)) %>% #In each conversation, find max rank- total msgs shared
  summarize(mean_avg = mean(max_ranks)) #get average rank of all conversations
```

```
## # A tibble: 1 x 1
##   mean_avg
##      <dbl>
## 1     18.3
```

```r
#Average no. of msgs shared by consumer per each unique conversation before escalation was made"
Data_tibble %>%
  group_by(conversationId)%>% #To divide in unique conversations
  filter(sentBy == "Consumer", escalation== "0") %>%
  summarize(max_ranks = max(rank)) %>%
  summarize(mean_avg = mean(max_ranks))
```

```
## # A tibble: 1 x 1
##   mean_avg
##      <dbl>
## 1     9.97
```

```r
#Max number of msgs in any conversation before escalation was made
Data_tibble %>%
  group_by(sentBy) %>% filter(escalation == "0") %>%
  summarize(x = max(rank))
```

```
## # A tibble: 2 x 2
##   sentBy       x
##   <chr>    <dbl>
## 1 Agent       49
## 2 Consumer    55
```

```r
#Max no of msgs shared in any conversation

Data_tibble %>%
  group_by(sentBy) %>% filter(escalation == "1") %>%
  summarize(x = max(rank))
```

```
## # A tibble: 2 x 2
##   sentBy       x
##   <chr>    <dbl>
## 1 Agent       66
## 2 Consumer    67
```

```r
#Unique conversations in data

def <- as.numeric(sqldf("select count(distinct(conversationId)) from Data_tibble"))
def
```

```
## [1] 4876
```

```r
abc <- as.numeric(sqldf("select count(distinct(conversationId)) from Data_tibble where sentBy = 'Consume

abc
```

```
## [1] 3884
```

```r
cat("In total of", def , "conversations", (abc/def)*100, "% of conversations had a long input being type
```

```
## In total of 4876 conversations 79.65546 % of conversations had a long input being typed in the very i
```

```r
def2 <- as.numeric(sqldf("select count(distinct(conversationId)) from Data_tibble where  sentBy = 'Consu

def2
```

```
## [1] 4518
```

```r
cat("In total of", def2 , "out of", def,  "conversations- a long input was entered, and in", (abc/def2)*
```

```
## In total of 4518 out of 4876 conversations- a long input was entered, and in 85.96724 % of those conv
```

```r
#Total rows
Data_tibble %>% summarize(Total_msgs = n())
```

```
## # A tibble: 1 x 1
##   Total_msgs
##        <int>
## 1      81335
```

```r
#Let us tokenize and separate words of reviews

Data_review <- Data_tibble %>%
  unnest_tokens(words, text)
Data_review
```

```
## # A tibble: 1,237,760 x 9
##    conversationId          timeL sentBy prev_text         prev_sentBy escalation
##    <chr>                   <dbl> <chr>  <chr>             <chr>            <dbl>
##  1 d1f1d0a5-e5de-45de~   1.62e12 Consu~ <NA>              <NA>                 0
##  2 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  3 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  4 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  5 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  6 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  7 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  8 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  9 d1f1d0a5-e5de-45de~   1.62e12 Agent  ?? Hey there! I'~ Agent                0
## 10 d1f1d0a5-e5de-45de~   1.62e12 Agent  ?? Hey there! I'~ Agent                0
## # ... with 1,237,750 more rows, and 3 more variables: totalCharacters <dbl>,
## #   rank <dbl>, words <chr>
```

```r
# least used words in reviews
Data_review %>% count(words) %>% arrange(n)
```

```
## # A tibble: 18,181 x 2
##    words                                      n
##    <chr>                                  <int>
##  1 ________________________________________     1
##  2 0.07                                         1
##  3 0.18                                         1
##  4 0.57                                         1
##  5 0.5mbps                                      1
##  6 0000                                         1
##  7 0000000000                                   1
##  8 0004                                         1
##  9 0005                                         1
## 10 00137135597                                  1
## # ... with 18,171 more rows
```

```r
#Most used words in reviews
Data_review %>% count(words) %>% arrange(desc(n))
```

```
## # A tibble: 18,181 x 2
##    words     n
##    <chr> <int>
##  1 to    51828
##  2 you   45305
##  3 i     35454
##  4 your  31750
##  5 the   27854
##  6 a     23052
##  7 can   18047
##  8 for   15012
##  9 and   14319
## 10 me    13600
## # ... with 18,171 more rows
```

```
#As we notice, most often used words are useless filler words, lets remove them

Data_review2 <- Data_tibble %>%
  unnest_tokens(output = word, input = text) %>%  #don't know why but can only use output = word, no ot
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```
#Anti-join is opposite of join, it fishes out values of A that didn't match with B

#Notice how number of rows reduce drastically
Data_review2
```

```
## # A tibble: 451,777 x 9
##    conversationId          timeL sentBy prev_text         prev_sentBy escalation
##    <chr>                   <dbl> <chr>  <chr>             <chr>            <dbl>
##  1 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  2 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  3 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  4 d1f1d0a5-e5de-45de~   1.62e12 Agent  Help              Consumer             0
##  5 d1f1d0a5-e5de-45de~   1.62e12 Agent  ?? Hey there! I'~ Agent                0
##  6 d1f1d0a5-e5de-45de~   1.62e12 Agent  ?? Hey there! I'~ Agent                0
##  7 d1f1d0a5-e5de-45de~   1.62e12 Agent  ?? Hey there! I'~ Agent                0
##  8 d1f1d0a5-e5de-45de~   1.62e12 Agent  ?? Hey there! I'~ Agent                0
##  9 d1f1d0a5-e5de-45de~   1.62e12 Agent  ?? Hey there! I'~ Agent                0
## 10 d1f1d0a5-e5de-45de~   1.62e12 Agent  ?? Hey there! I'~ Agent                0
## # ... with 451,767 more rows, and 3 more variables: totalCharacters <dbl>,
## #   rank <dbl>, word <chr>
```

```
#Most used imp words
Data_review2 %>% count(word) %>% arrange(desc(n))
```

```
## # A tibble: 17,545 x 2
##    word          n
##    <chr>     <int>
##  1 service    8406
##  2 account    8359
```

```
##  3 free       6189
##  4 optimum    5808
##  5 phone      5165
##  6 questions  4765
##  7 hey        4578
##  8 allie      4575
##  9 virtual    4542
## 10 assistant  4539
## # ... with 17,535 more rows
```

```
#least used imp words
Data_review2 %>% count(word) %>% arrange(n)
```

```
## # A tibble: 17,545 x 2
##    word                                      n
##    <chr>                                 <int>
##  1 ________________________________________     1
##  2 0.07                                        1
##  3 0.18                                        1
##  4 0.57                                        1
##  5 0.5mbps                                     1
##  6 0000                                        1
##  7 0000000000                                  1
##  8 0004                                        1
##  9 0005                                        1
## 10 00137135597                                 1
## # ... with 17,535 more rows
```

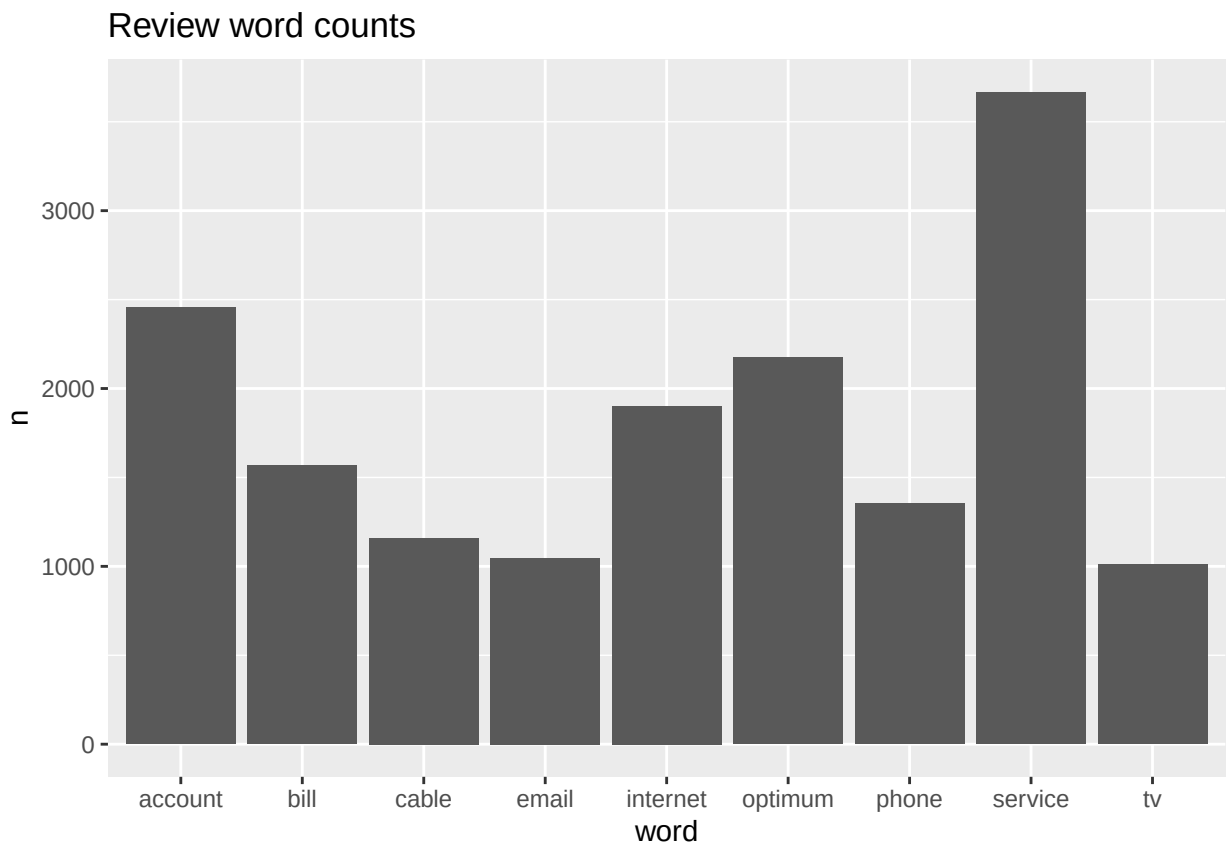
```
#Most used imp words by consumer
Data_review2 %>% filter(sentBy == 'Consumer') %>% count(word) %>% arrange(desc(n))
```

```
## # A tibble: 16,803 x 2
##    word         n
##    <chr>    <int>
##  1 service   3666
##  2 account   2456
##  3 optimum   2176
##  4 internet  1902
##  5 bill      1567
##  6 phone     1352
##  7 cable     1160
##  8 email     1046
##  9 tv        1010
## 10 call       828
## # ... with 16,793 more rows
```

```
#Perfect, Most requested services can thus be guessed to be for internet, bills, phone, cable , tv etc

#Lets say we want words that were used more than 900 times
imp_words <- Data_review2 %>% filter(sentBy == 'Consumer') %>% count(word) %>% filter(n>900) %>% arrang
imp_words
```

```
## # A tibble: 9 x 2
##   word         n
##   <chr>    <int>
## 1 service   3666
## 2 account   2456
## 3 optimum   2176
## 4 internet  1902
## 5 bill      1567
## 6 phone     1352
## 7 cable     1160
## 8 email     1046
## 9 tv        1010
```

```
ggplot(imp_words, aes(x= word, y = n))+ geom_col() + ggtitle("Review word counts")
```



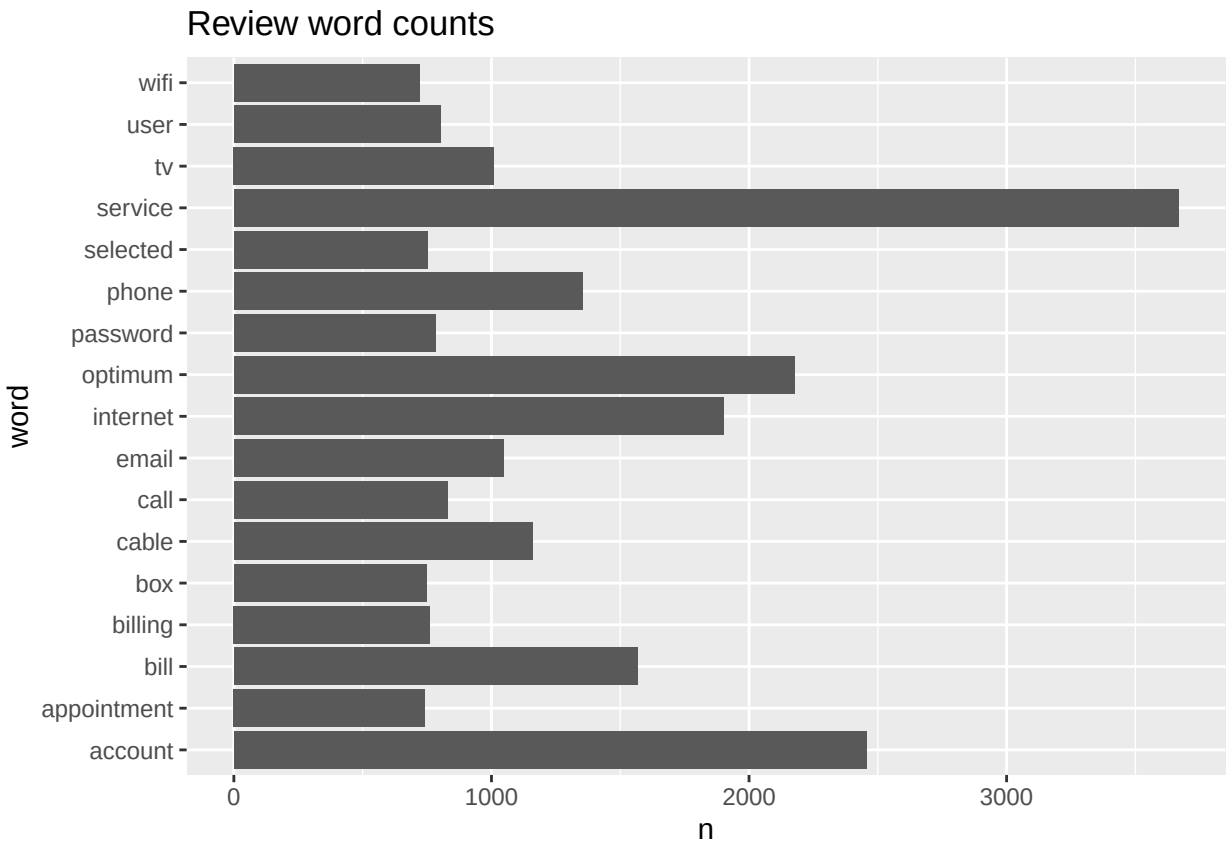

```
#To include more words, lets flip the coordinates

Data_review2 <- Data_tibble %>%
  unnest_tokens(output = word, input = text) %>%  #text is from our file under column text
                                                  #word is by default the output for unnest_tokens
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

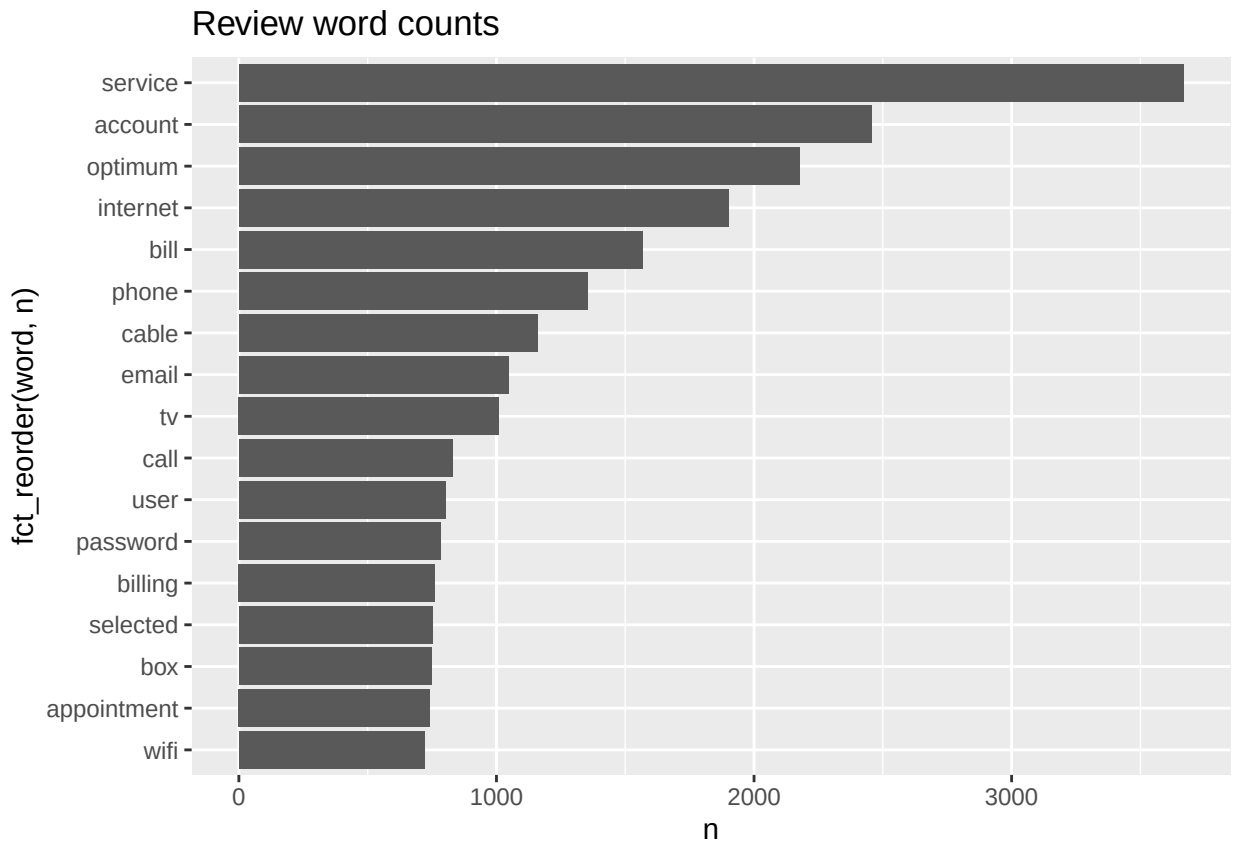
```
imp_words2 <- Data_review2 %>% filter(sentBy == 'Consumer') %>% count(word) %>% filter(n>700) %>% arran
imp_words2
```

```
## # A tibble: 17 x 2
##    word            n
##    <chr>       <int>
##  1 service      3666
##  2 account      2456
##  3 optimum      2176
##  4 internet     1902
##  5 bill         1567
##  6 phone        1352
##  7 cable        1160
##  8 email        1046
##  9 tv           1010
## 10 call          828
## 11 user          802
## 12 password      784
## 13 billing       761
## 14 selected      751
## 15 box           749
## 16 appointment   742
## 17 wifi          720
```

```
ggplot(imp_words2, aes(x= word, y = n))+ geom_col() + coord_flip() + ggtitle("Review word counts")
```



```
#Lets arrange in a readable sequence

ggplot(imp_words2, aes(x= fct_reorder(word, n), y = n))+ geom_col() + coord_flip() + ggtitle("Review wo
```




```
stop_words
```

```
## # A tibble: 1,149 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 a           SMART
##  2 a's         SMART
##  3 able        SMART
##  4 about       SMART
##  5 above       SMART
##  6 according   SMART
##  7 accordingly SMART
##  8 across      SMART
##  9 actually    SMART
## 10 after       SMART
## # ... with 1,139 more rows
```

```
#As we don't like a few words, we make custom stop words now
custom_stop_words <- tribble(
  ~word, ~lexicon,
```

```
  "service", "CUSTOM",
  "account", "CUSTOM")

stop_words2 <- stop_words %>%
  bind_rows(custom_stop_words)
stop_words2
```

```
## # A tibble: 1,151 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 a           SMART
##  2 a's         SMART
##  3 able        SMART
##  4 about       SMART
##  5 above       SMART
##  6 according   SMART
##  7 accordingly SMART
##  8 across      SMART
##  9 actually    SMART
## 10 after       SMART
## # ... with 1,141 more rows
```

```
Data_review2 <- Data_tibble %>%
  unnest_tokens(output = word, input = text) %>%  #text is from our file under column text
                                                  #word is by default the output for unnest_tokens
  anti_join(stop_words2)
```

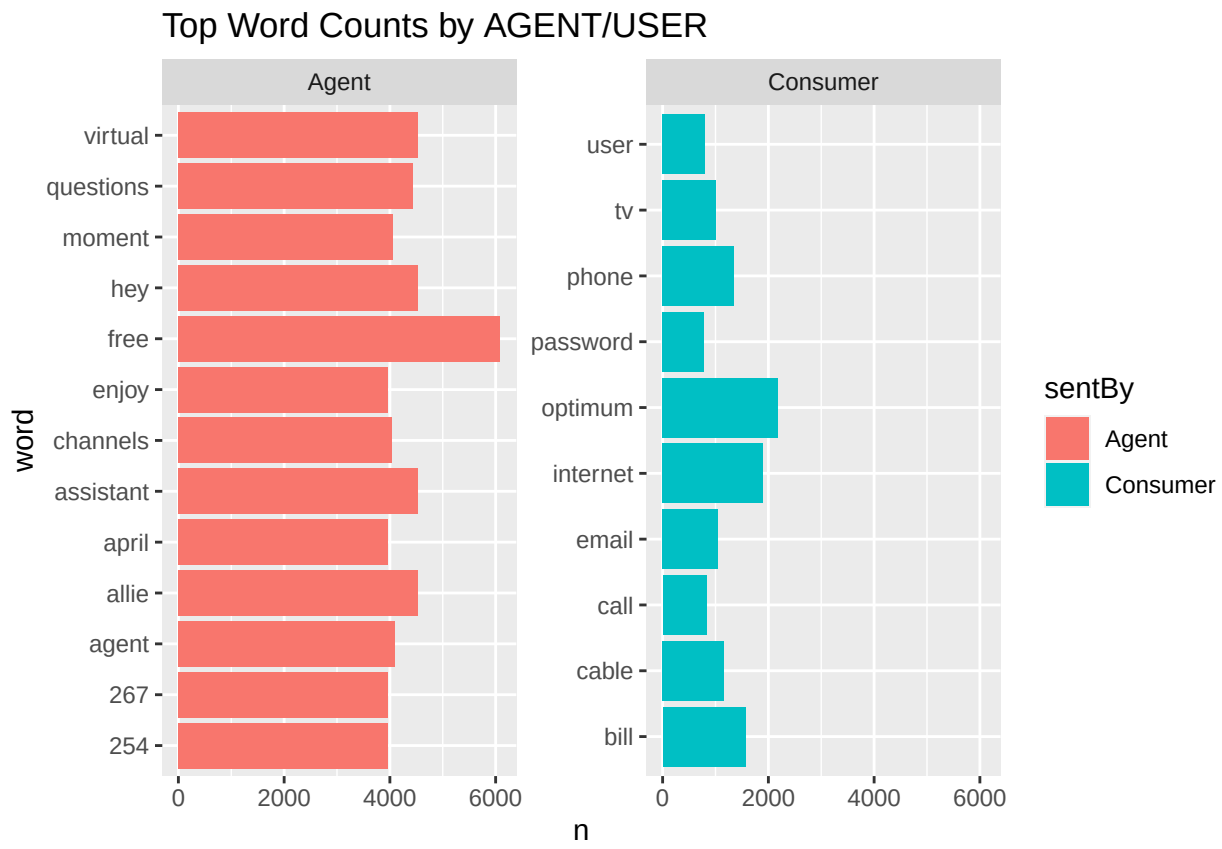
```
## Joining, by = "word"
```

```
imp_words2 <- Data_review2 %>% filter(sentBy == 'Consumer') %>% count(word) %>% filter(n>700) %>% arran
imp_words2
```

```
## # A tibble: 15 x 2
##    word            n
##    <chr>       <int>
##  1 optimum      2176
##  2 internet     1902
##  3 bill         1567
##  4 phone        1352
##  5 cable        1160
##  6 email        1046
##  7 tv           1010
##  8 call          828
##  9 user          802
## 10 password      784
## 11 billing       761
## 12 selected      751
## 13 box           749
## 14 appointment   742
## 15 wifi          720
```

```
#Could also have used mutate
#imp_words2 <- Data_review2 %>% filter(sentBy == 'Consumer') %>% count(word) %>% filter(n>700) %>% muta


#fct_reorder is to reorder word column, acc to decreasing n when plotting graph too, as just arrange(de

#ggplot(imp_words2, aes(x= word2, y = n))+ geom_col() + coord_flip() + ggtitle("Review word counts")
```

```
abc <- Data_review2 %>% group_by(sentBy) %>%
  count(word) %>% filter(n>700) %>% arrange(desc(n)) %>% top_n(10,n) %>%
  #Until here job is to find top 10 in both categories
  ungroup()

ggplot(abc, aes(word, n, fill = sentBy)) + geom_col() +
  facet_wrap(~sentBy, scales = "free_y")+ # So plot 2 y axis separately
  coord_flip()+ ggtitle("Top Word Counts by AGENT/USER")
```



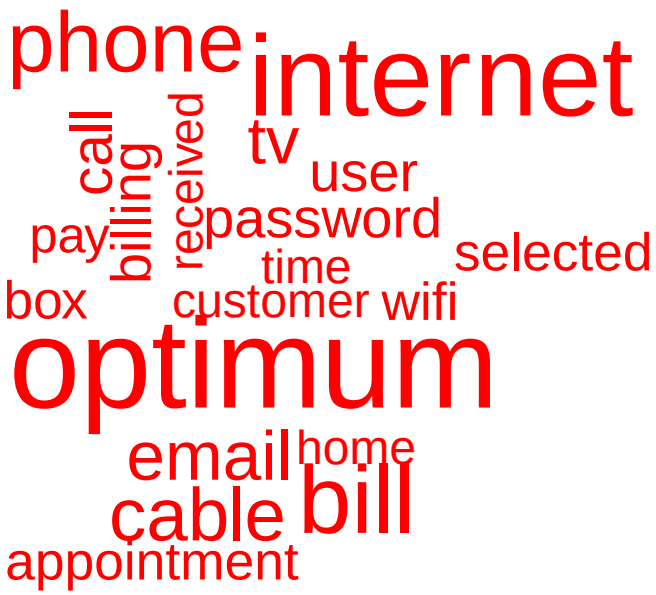

```
#Using word cloud to show most used words

Data_review2 <- Data_tibble %>%
  unnest_tokens(output = word, input = text) %>%
  anti_join(stop_words2)
```

```
## Joining, by = "word"
```

```
abc <- Data_review2 %>% filter(sentBy == "Consumer") %>% count(word) %>% arrange(desc(n))
abc
```

```
## # A tibble: 16,801 x 2
##    word         n
##    <chr>    <int>
##  1 optimum   2176
##  2 internet  1902
##  3 bill      1567
##  4 phone     1352
##  5 cable     1160
##  6 email     1046
##  7 tv        1010
##  8 call       828
##  9 user       802
## 10 password   784
## # ... with 16,791 more rows
```

```
wordcloud(
  words = abc$word,
  freq = abc$n,
  max.words = 20,
  colors = "red"
)
```



```r
#For sentiment analysis, let's get some of saved ones
get_sentiments("bing")
```

```
## # A tibble: 6,786 x 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
## # ... with 6,776 more rows
```

```r
#So we see more negative words than positive
get_sentiments("bing") %>% count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   4781
## 2 positive   2005
```

```r
options(readr.default_locale=readr::locale(tz="Europe/Warsaw"))
get_sentiments("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
## # ... with 2,467 more rows
```

```r
get_sentiments("afinn") %>%
  summarize(
   min = min(value),
    max=max(value)

  )
```

```
## # A tibble: 1 x 2
##     min   max
##   <dbl> <dbl>
## 1    -5     5
```

```
#This shows that afinn has emotions ranging from value -5 to +5

#sentiment_counts <- get_sentiments("loughran") %>%
 #count(sentiment) %>%
 #mutate(sentiment2 = fct_reorder(sentiment, n))


#ggplot(sentiment_counts, aes(x= sentiment2, y = n))+
#geom_col() +
  #coord_flip() +
  #labs(
    #title ="Sentiment Counts",
    #x ="Counts",
    #y ="Sentiment"
  #)
```

```
#install.Rtools(TRUE, FALSE, page_with_download_url = "https://cran.r-project.org/bin/windows/Rtools/")
#inst
#Sys.setenv(TZ = "Europe/Warsaw")


#Add this line to install and get over the TZ issue
options(readr.default_locale=readr::locale(tz="Europe/Warsaw"))
get_sentiments("nrc")
```

```
## # A tibble: 13,901 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
## # ... with 13,891 more rows
```

```
get_sentiments("nrc") %>%
  count(sentiment) %>%
  arrange(desc(n))
```

```
## # A tibble: 10 x 2
##    sentiment        n
##    <chr>        <int>
```

```
##  1 negative      3324
##  2 positive      2312
##  3 fear          1476
##  4 anger         1247
##  5 trust         1231
##  6 sadness       1191
##  7 disgust       1058
##  8 anticipation   839
##  9 joy            689
## 10 surprise       534
```

```
#Join to keep all rows where match was found
sentiment_review <- Data_review2 %>% inner_join(get_sentiments("bing"))
```

```
## Joining, by = "word"
```

```
sentiment_review
```

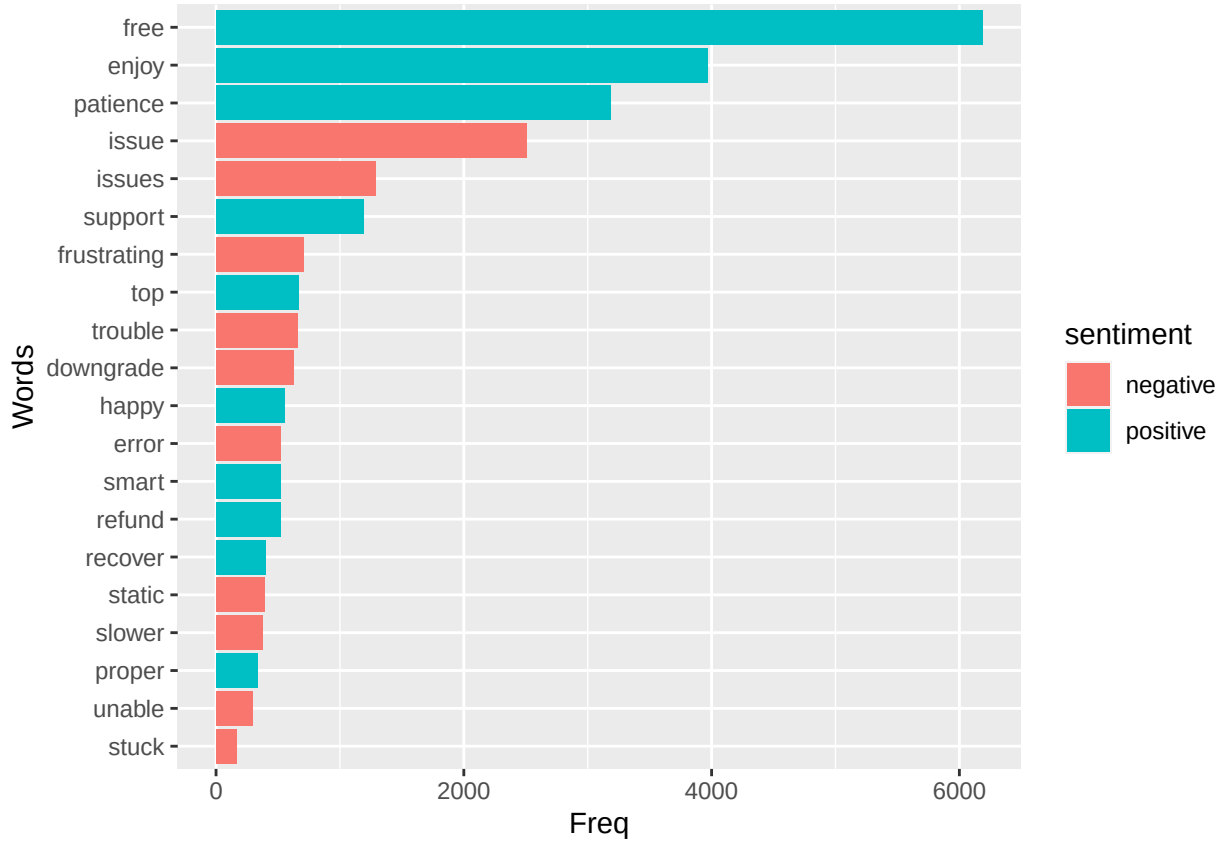
```
## # A tibble: 32,177 x 10
##    conversationId        timeL sentBy prev_text           prev_sentBy escalation
##    <chr>                 <dbl> <chr>  <chr>               <chr>            <dbl>
##  1 d1f1d0a5-e5de-45d~  1.62e12 Agent  "?? Hey there! I'm~ Agent                0
##  2 d1f1d0a5-e5de-45d~  1.62e12 Agent  "Internet"          Consumer             1
##  3 d1f1d0a5-e5de-45d~  1.62e12 Consu~ "Before I transfer~ Agent                1
##  4 d1f1d0a5-e5de-45d~  1.62e12 Agent  "Great! I\x92ll pa~ Agent                1
##  5 0bb7fdaa-967b-489~  1.62e12 Agent  "Sorry, I'm not qu~ Agent                0
##  6 0bb7fdaa-967b-489~  1.62e12 Consu~ "Feel free to ask ~ Agent                0
##  7 0bb7fdaa-967b-489~  1.62e12 Agent  "unless thank you ~ Consumer             0
##  8 0bb7fdaa-967b-489~  1.62e12 Agent  "unless thank you ~ Consumer             0
##  9 cafbf819-f7d6-49a~  1.62e12 Consu~  <NA>               <NA>                 0
## 10 cafbf819-f7d6-49a~  1.62e12 Agent  "Spam is generally~ Agent                0
## # ... with 32,167 more rows, and 4 more variables: totalCharacters <dbl>,
## #   rank <dbl>, word <chr>, sentiment <chr>
```

```
sentiment_review %>% count(sentiment)
```

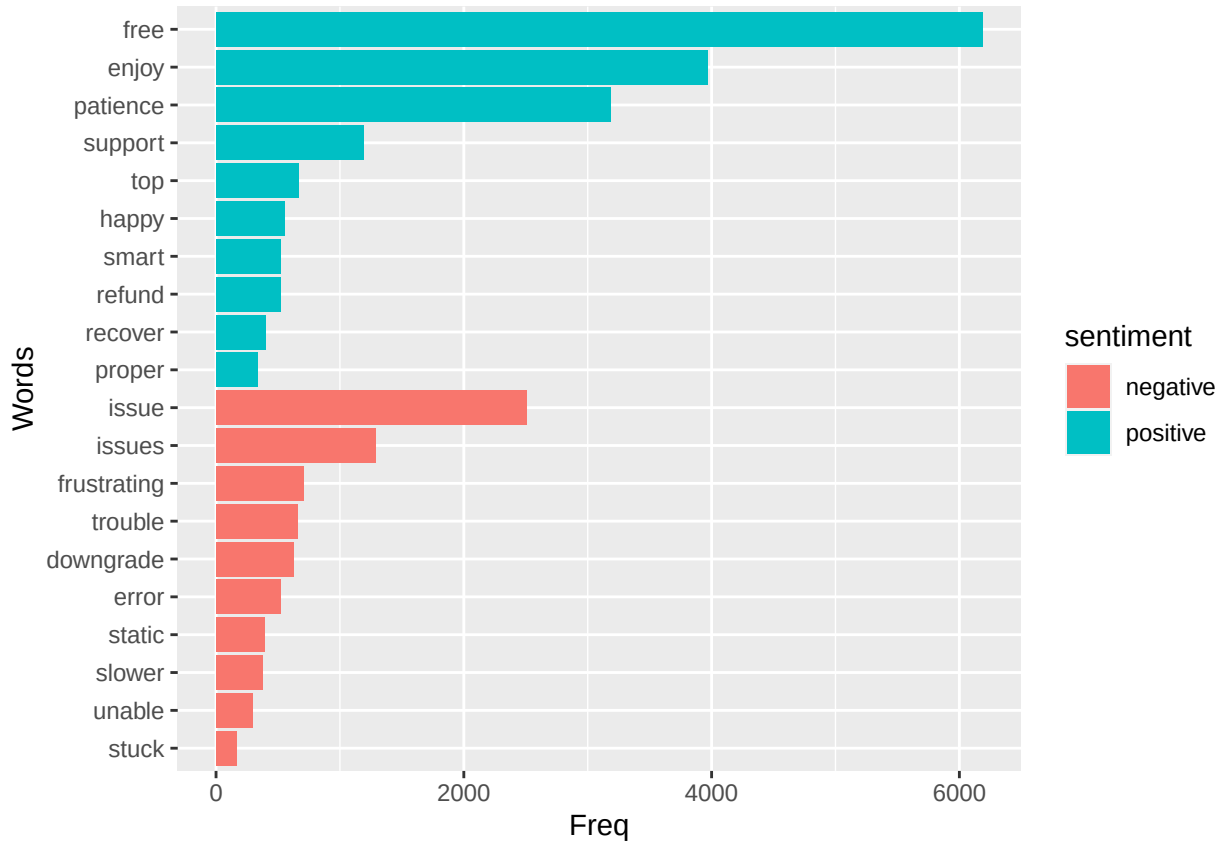
```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative  11662
## 2 positive  20515
```

```
#Wow, that's nice..Not so many negative words and more of positive ones
```

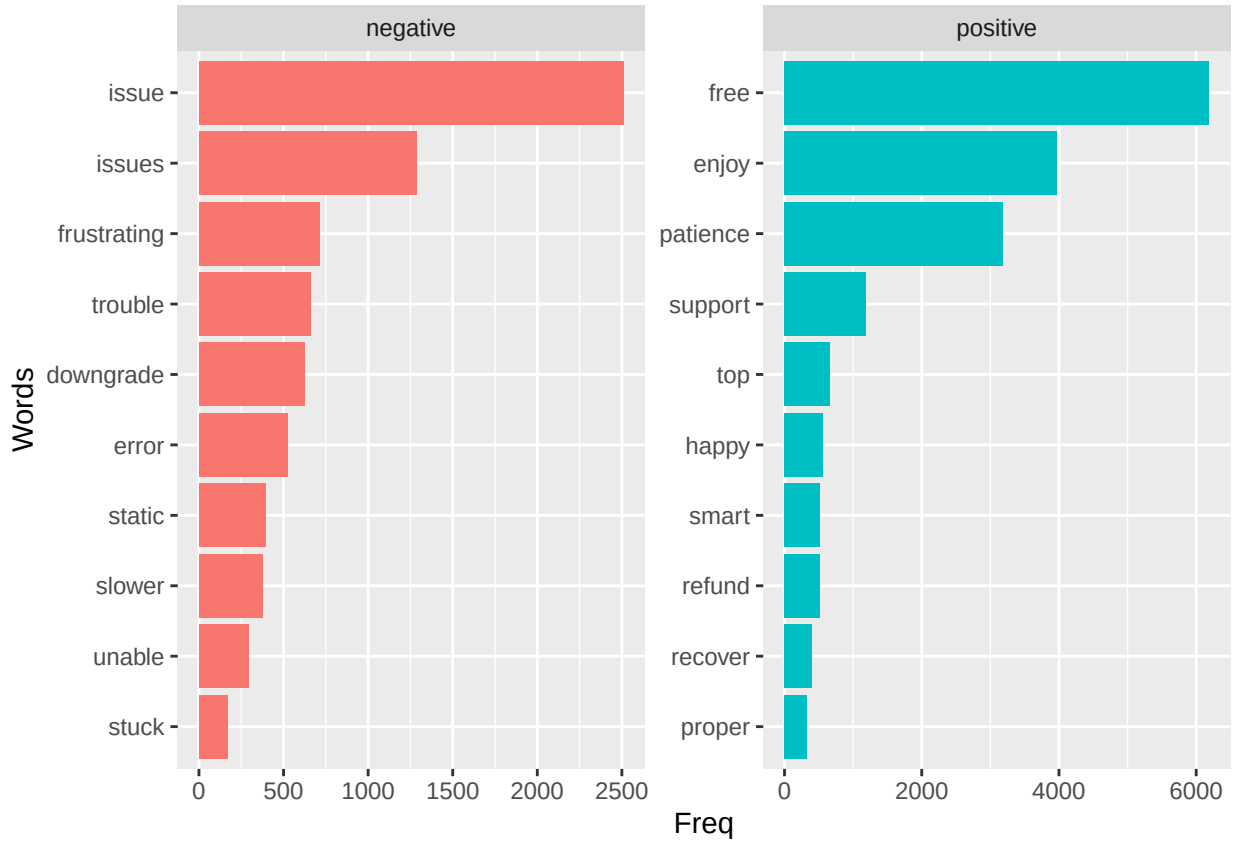
```
#Let's see which words are most used for a particular sentiment
abc = sentiment_review %>% count(word, sentiment) %>% arrange(desc(n)) %>% group_by(sentiment) %>% top_

#Notice how top_n is used after grouping by sentiment..so as to group first and then find top 10

ggplot(abc, aes(word2, n, fill = sentiment))+ geom_col()+  coord_flip() +labs(x ="Words", y ="Freq")
```




```
#Let's see which words are most used for a particular sentiment
abc = sentiment_review %>% count(word, sentiment) %>% arrange(desc(n)) %>% group_by(sentiment) %>% top_

#Notice how top_n is used after grouping by sentiment..so as to group first and then find top 10

ggplot(abc, aes(word2, n, fill = sentiment))+ geom_col()+  coord_flip() +labs(x ="Words", y ="Freq")
```




```
ggplot(abc, aes(word2, n, fill = sentiment))+ geom_col(show.legend = FALSE)+ facet_wrap(~sentiment, sca
```




```
#How many negative and positive messages were shared before and after escalation

 Data_review2 %>% inner_join(get_sentiments("bing")) %>%  count(escalation, sentiment)
```

```
## Joining, by = "word"
```

```
## # A tibble: 4 x 3
##   escalation sentiment     n
##        <dbl> <chr>     <int>
## 1          0 negative  11164
## 2          0 positive  16721
## 3          1 negative    498
## 4          1 positive   3794
```

```
#Using spread, we can spread sentiment column into 2- positive and negative, based on n, that comes fro

Z = Data_review2 %>% inner_join(get_sentiments("bing")) %>%  count(rank, sentiment) %>%
  spread(sentiment, n)
```

```
## Joining, by = "word"
```

```
Z %>% filter(rank == "1")
```

```
## # A tibble: 1 x 3
```

```
##    rank negative positive
##   <dbl>    <int>    <int>
## 1     1     2722     1340
```

```
#Z %>% filter(is.na == FALSE) %>% summarise(sum = sum(Z$negative))
Z %>% mutate(overall_sentiment = positive - negative)
```

```
## # A tibble: 50 x 4
##     rank negative positive overall_sentiment
##    <dbl>    <int>    <int>             <int>
##  1     1     2722     1340             -1382
##  2     2       21       20                -1
##  3     3      115     7922              7807
##  4     4      786     1293               507
##  5     5      698     1781              1083
##  6     6      672      754                82
##  7     7      858      655              -203
##  8     8      809      419              -390
##  9     9      801      627              -174
## 10    10      684      495              -189
## # ... with 40 more rows
```

```
total_neg = as.numeric(Data_review2 %>% inner_join(get_sentiments("bing")) %>%  filter( sentiment == "n
```

```
## Joining, by = "word"
```

```
total_neg
```

```
## [1] 11662
```

```
#This shows that a lot of negative words are being used in the very first message
cat("Infact out of ", total_neg, "negative reviews, ", (2722/total_neg)*100 , " % were entered in the v
```

```
## Infact out of  11662 negative reviews,  23.34076  % were entered in the very first message
```

```
OS1 = as.numeric(Data_review2 %>% inner_join(get_sentiments("bing")) %>%  count(rank, sentiment) %>%
  spread(sentiment, n) %>%  mutate(overall_sentiment = positive - negative)  %>% summarise(OS = sum(!is
```

```
## Joining, by = "word"
```

```
OS2 = as.numeric(Data_review2 %>% inner_join(get_sentiments("bing")) %>%  count(rank, sentiment) %>%
  spread(sentiment, n) %>%  mutate(overall_sentiment = positive - negative)  %>% summarise(OS = mean(!is
```

```
## Joining, by = "word"
```

```
cat("Overall sentiment rating for consumers' requests can be termed across all ranks as (if summed) +",
```

```
## Overall sentiment rating for consumers' requests can be termed across all ranks as (if summed) + 41 
```

#Clustering #Unsupervised learning #Topic Modeling

```r
#Latent Dirischlet allocation
#LDA Topic modeling
#each topic is a collection of word probabilities for all of the unique words used in the corpus.
```

```r
#install.packages("tm")
#First step - Tokenize reviews to separate it word by word
Data_review2 <- Data_tibble %>% filter(sentBy == "Consumer") %>%
  unnest_tokens(output = word, input = text) %>%
  anti_join(stop_words2)
```

```
## Joining, by = "word"
```

```r
Data_review2 %>% count(word, conversationId) %>% cast_dtm(conversationId, word, n)
```

```
## <<DocumentTermMatrix (documents: 4876, terms: 16801)>>
## Non-/sparse entries: 96811/81824865
## Sparsity           : 100%
## Maximal term length: NA
## Weighting          : term frequency (tf)
```

```r
#Let us understand what happened here
#First we specify the document column, ie conversationId then,the term column ie word, last word counts
#Sequence is imp
#cast into dtm basing on ID, tokenized words and count

#Output saysm 4876 reviewsm 17543 words
```

```r
#Let us save our dtm as matrix, and access data as it will be a huge matrix,

dtm_review <- Data_review2 %>% count(word, conversationId) %>%
  cast_dtm(conversationId, word, n) %>% as.matrix()

#check 1st row, and 4th word ( rows are conversation ids or documents and columns are unique words)

dtm_review[1,4]
```

```
## [1] 0
```

```r
#Gives meaningless output, so lets see multiple rows and words
dtm_review[1:4, 2000:2004]
```

```
##                                     Terms
## Docs                                 3473732794 3473742943 3473874640
##   af5486f3-5c27-4d8c-a674-8cac43ff3e53          0          0          0
##   073a3fd1-ff1e-4939-92fe-ce23becde877          0          0          0
##   0891d0ac-61fb-4284-bae8-5f0d9043f2f1          0          0          0
##   13d18750-97bc-47bf-8742-c04272a7bc2e          0          0          0
##                                     Terms
## Docs                                 3473935319 3474176064
##   af5486f3-5c27-4d8c-a674-8cac43ff3e53          0          0
##   073a3fd1-ff1e-4939-92fe-ce23becde877          0          0
##   0891d0ac-61fb-4284-bae8-5f0d9043f2f1          0          0
##   13d18750-97bc-47bf-8742-c04272a7bc2e          0          0
```

```
dtm_review[1:4, 9000:9004]
```

```
##                                     Terms
## Docs                                 finding fine finesse fing finger
##   af5486f3-5c27-4d8c-a674-8cac43ff3e53       0    0       0    0      0
##   073a3fd1-ff1e-4939-92fe-ce23becde877       0    0       0    0      0
##   0891d0ac-61fb-4284-bae8-5f0d9043f2f1       0    0       0    0      0
##   13d18750-97bc-47bf-8742-c04272a7bc2e       0    0       0    0      0
```

```
#^ so we see that in first 4 conversations how many times these words (9000-9004) appeared.
#As many are 0, it means they appeared sparsely, but word email appeared 5 times in the 1st conversatio
```

```
lda_out <- LDA(
  dtm_review, k = 4, method = "Gibbs", control=list(seed=42)
)

lda_out
```

```
## A LDA_Gibbs topic model with 4 topics.
```

```
glimpse(lda_out)
```

```
## Formal class 'LDA_Gibbs' [package "topicmodels"] with 16 slots
##   ..@ seedwords      : NULL
##   ..@ z              : int [1:122369] 3 4 4 1 3 3 3 2 3 3 ...
##   ..@ alpha          : num 12.5
##   ..@ call           : language LDA(x = dtm_review, k = 4, method = "Gibbs", control = list(seed = 4
##   ..@ Dim            : int [1:2] 4876 16801
##   ..@ control        :Formal class 'LDA_Gibbscontrol' [package "topicmodels"] with 14 slots
##   ..@ k              : int 4
##   ..@ terms          : chr [1:16801] "_____________________________________" "0" "0.00" "0.07" ..
##   ..@ documents      : chr [1:4876] "af5486f3-5c27-4d8c-a674-8cac43ff3e53" "073a3fd1-ff1e-4939-92fe-
##   ..@ beta           : num [1:4, 1:16801] -12.7 -12.7 -10.3 -12.7 -12.7 ...
##   ..@ gamma          : num [1:4876, 1:4] 0.147 0.336 0.379 0.343 0.188 ...
##   ..@ wordassignments:List of 5
##   .. ..$ i   : int [1:96811] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..$ j   : int [1:96811] 1 1740 2445 3944 5828 6054 7796 7899 7901 8528 ...
##   .. ..$ v   : num [1:96811] 3 4 1 3 3 3 2 3 3 3 ...
##   .. ..$ nrow: int 4876
##   .. ..$ ncol: int 16801
##   .. ..- attr(*, "class")= chr "simple_triplet_matrix"
##   ..@ loglikelihood  : num -835949
##   ..@ iter           : int 2000
##   ..@ logLiks        : num(0)
##   ..@ n              : int 122369
```

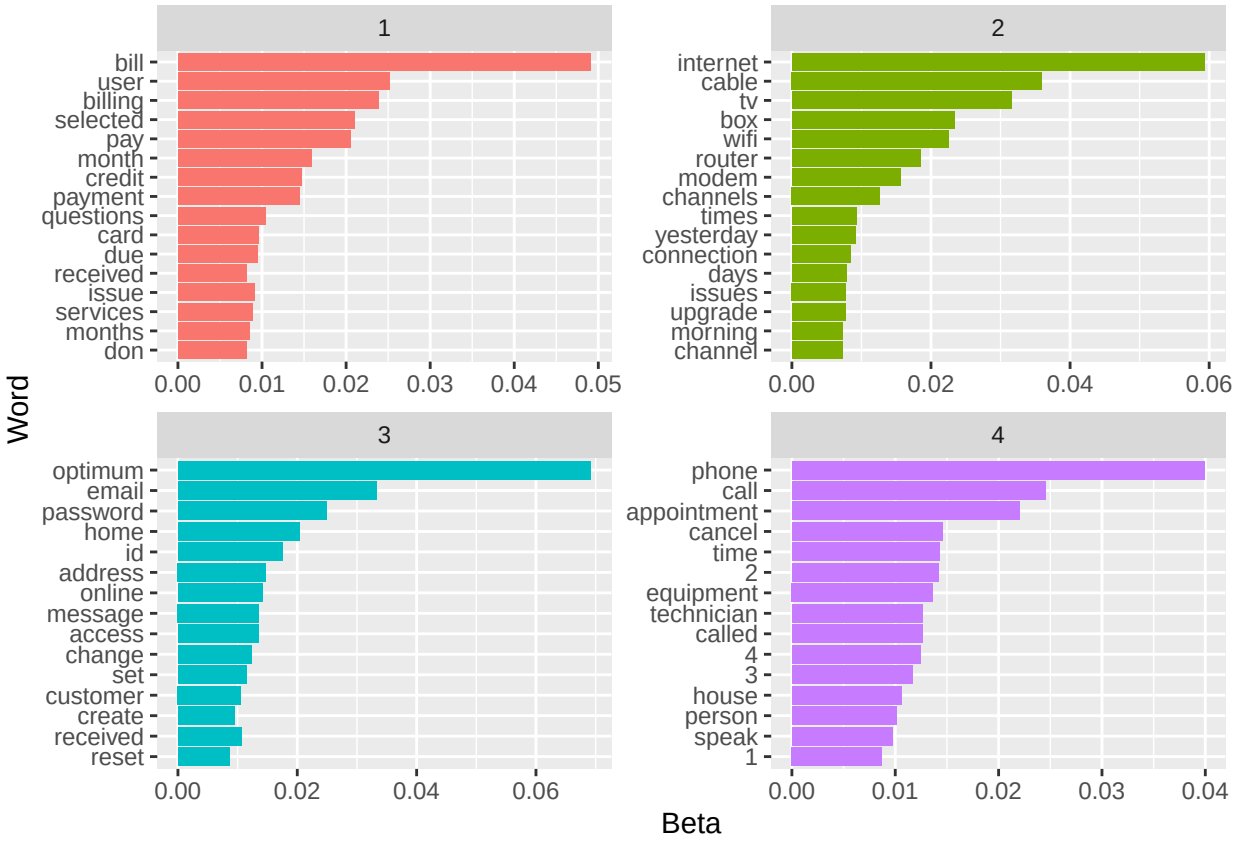
```
#as not much can be inferred this way, lets convert it to tidy
```

```
#install.packages("reshape2")
```

```r
#This step is done to cast it into dtm and then tidy it up for better readability,

#take out as a matrix, the beta column so it can be represented graphically etc
lda_topics <- lda_out %>%
  tidy(matrix = "beta")
lda_topics %>% arrange(desc(beta))
```

```
## # A tibble: 67,204 x 3
##    topic term       beta
##    <int> <chr>     <dbl>
##  1     3 optimum  0.0692
##  2     2 internet 0.0594
##  3     1 bill     0.0491
##  4     4 phone    0.0399
##  5     2 cable    0.0360
##  6     3 email    0.0333
##  7     2 tv       0.0315
##  8     1 user     0.0251
##  9     3 password 0.0249
## 10     4 call     0.0246
## # ... with 67,194 more rows
```

```r
word_probs <- lda_topics %>% group_by(topic) %>%
  top_n(15, beta) %>% ungroup() %>%
  mutate(term2 = fct_reorder(term, beta))

ggplot(word_probs, aes(x= term2, y = beta, fill = as.factor(topic))) +
  geom_col(show.legend = FALSE) + facet_wrap(~topic, scales = "free") + coord_flip() + labs(y ='Beta', x
```




```
#Since with k = 3, optimum appears in topic 1 and 3 and we don't want our words repeating topics, let u

#Based on highest probabilities in 4 topics, lets us classify them this way:

#Topic 1 - Payment/Billing queries
#Topic 2 - Internet/Subscription queries
#Topic 3 - Access Issues
#Topic 4 - Escalation to human agent requests
tinytex::install_tinytex()
```