SKYTRAX AIRLINE REVIEW ANALYSIS

Rounak Saha

2024-12-17

```
options(repos = c(CRAN = "https://cloud.r-project.org"))
install.packages("tidyverse")
```

```
##
## The downloaded binary packages are in
##  /var/folders/fm/k4r2n81j77q8l80mh7tnvq2r0000gn/T//RtmpXdsoRR/downloaded_packages
```

```
install.packages("rvest")
```

```
##
## The downloaded binary packages are in
##  /var/folders/fm/k4r2n81j77q8l80mh7tnvq2r0000gn/T//RtmpXdsoRR/downloaded_packages
```

```
install.packages("data.table")
```

```
##
## The downloaded binary packages are in
##  /var/folders/fm/k4r2n81j77q8l80mh7tnvq2r0000gn/T//RtmpXdsoRR/downloaded_packages
```

```
install.packages("wordcloud")
```

```
##
## The downloaded binary packages are in
##  /var/folders/fm/k4r2n81j77q8l80mh7tnvq2r0000gn/T//RtmpXdsoRR/downloaded_packages
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(rvest)
```

```
##
## Attaching package: 'rvest'
##
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```r
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.3.3
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(wordcloud)
```

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

```r
dataloop <- data.frame()

# for loop
for (i in 1:30) {

  # construct the URL with the current value of start
  urlloop <- paste0("https://www.airlinequality.com/airline-reviews/british-airways/page/",
                    i, "/?sortby=post_date%3ADesc&pagesize=100")

  # Read the html content of the webpage
  webpageloop <- read_html(urlloop)

  #Extract the data from the webpage using CSS selectors
  Review = webpageloop %>% html_nodes(".text_content") %>% html_text()
  # Extract all recommendations from multiple reviews
  Recommended <- webpageloop %>%
    html_nodes(".recommended+ .review-value") %>%  # Select review recommendation nodes
    html_text(trim = TRUE)
```

```
  Date_Flown = webpageloop %>% html_nodes(".date_flown+ .review-value") %>% html_text()
  Seat_Type =webpageloop %>% html_nodes(".cabin_flown+ .review-value") %>% html_text()
  Rating = webpageloop %>% html_nodes(".position-content .rating-10 span:nth-child(1)") %>% html_text()

  #combine extracted data into a data table
  data <- data.frame(Review,Date_Flown,Rating,Recommended,Seat_Type)

  data <- data %>%
    mutate(Review = ifelse(grepl("\\|", Review), Review, paste("|", Review))) %>%  # Add a placeholder
    separate(Review, into = c("Status", "Details"), sep = "\\|", fill = "right", extra = "merge") %>%
    mutate(Status = trimws(Status), Details = trimws(Details))  # Trim whitespace


  # add the data to overall data table
  dataloop <- bind_rows(dataloop,data)

  #Pause for few secs to avoid overload
  Sys.sleep(3)
}

view(dataloop)
```

```
str(dataloop)
```

```
## 'data.frame':    3000 obs. of  6 variables:
##  $ Status     : chr  "  Trip Verified" "  Trip Verified" "  Trip Verified" "  Trip Verified" ...
##  $ Details    : chr  "On a recent flight from Cyprus BA621 on 23/11/24, the second the cabin door was
##  $ Date_Flown : chr  "November 2024" "December 2024" "December 2024" "December 2024" ...
##  $ Rating     : chr  "1" "1" "2" "8" ...
##  $ Recommended: chr  "no" "no" "no" "yes" ...
##  $ Seat_Type  : chr  "Economy Class" "Economy Class" "Business Class" "Business Class" ...
```

```
dataloop$Rating <- as.numeric(dataloop$Rating)
dataloop <- as.data.table(dataloop)
```

```
dataloop[, .N, Rating]
```

```
##     Rating     N
##      <num> <int>
##  1:      1   830
##  2:      2   366
##  3:      8   279
##  4:      7   216
##  5:      4   202
##  6:     10   218
##  7:      3   354
##  8:      6   144
##  9:      9   209
## 10:      5   182
```

```
dataloop[, .N, Recommended]
```

```
##    Recommended     N
##         <char> <int>
## 1:          no  1951
## 2:         yes  1049
```

```
dataloop[, .(Average_rating= mean(Rating)), by= Seat_Type]
```

```
##          Seat_Type Average_rating
##              <char>          <num>
## 1:   Economy Class       3.924066
## 2:  Business Class       4.792657
## 3:     First Class       5.694805
## 4: Premium Economy       4.487805
```

```
#We will now convert this into a tibble. One column, called line, will have the line number (from 1 to

Review_text <- tibble(review_no = 1: length(dataloop$Details), text= dataloop$Details)
Review_text
```

```
## # A tibble: 3,000 x 2
##    review_no text
##        <int> <chr>
##  1         1 "On a recent flight from Cyprus BA621 on 23/11/24, the second the ~
##  2         2 "Flight BA 0560 arrived in Rome on 11 December where ALL passenger~
##  3         3 "This was the first time I flew British Airways, and it was a huge~
##  4         4 "Pretty good flight but still some small things that can be improv~
##  5         5 "Check in was fine, but no priority/fast track lines for security ~
##  6         6 "British Airways is absolute rubbish. I had to fly to Amsterdam fo~
##  7         7 "The flight time was changed at the last minute without warning an~
##  8         8 "   I'm so fraustrated. My flight was cancelled last minute, which~
##  9         9 "We have sat on this plane for an hour and forty five minutes awai~
## 10        10 "British Airways stranding my wife and I at Heathrow Airport for 2~
## # i 2,990 more rows
```

```
#We can now tokenise the text using unnest_tokens again, and then count the numbers
library(tidytext)

tokens<- Review_text %>% unnest_tokens(word, text)
tokens
```

```
## # A tibble: 499,910 x 2
##    review_no word
##        <int> <chr>
##  1         1 on
##  2         1 a
##  3         1 recent
##  4         1 flight
##  5         1 from
##  6         1 cyprus
```

```
##  7         1 ba621
##  8         1 on
##  9         1 23
## 10         1 11
## # i 499,900 more rows
```

```
#As we are only interested in words that are text, let's remove all words with
#digits and special characters from our set of words. We can do this using grepl().
tokens <- as.data.table(tokens)
tokens <- tokens[grepl("\\d", word)==FALSE, ]
tokens <- tokens[grepl("[:alpha:]", word), ]
tokens
```

```
##         review_no     word
##             <int>   <char>
##      1:         1        a
##      2:         1   flight
##      3:         1   cyprus
##      4:         1      the
##      5:         1      the
##     ---
## 300598:      3000     home
## 300599:      3000       at
## 300600:      3000 midnight
## 300601:      3000      was
## 300602:      3000 starving
```

```
#Now that the text is tidy, it is easy to count the number of occurrences of each word. We can do this

tokens[, .N, word]
```

```
##           word     N
##         <char> <int>
##    1:        a 11385
##    2:   flight  5766
##    3:   cyprus     6
##    4:      the 25110
##    5:    cabin  1527
##   ---
## 9185: appaling     1
## 9186:   arabia     1
## 9187:      ruh     2
## 9188:  degrade     1
## 9189:  cheddar     1
```

```
#Analysis is made difficult because the text contains lots of short words, like "the", "of" and "and",

data(stop_words)
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
## # i 1,139 more rows
```

```
#remove certain words that will not be used to determine the positive or negative sentiment
# Custom stopwords
custom_stopwords <- tibble(word = c('flight', 'ba', 'passenger', 'u', 'london',
                                    'airway', 'british', 'airline', 'heathrow',
                                    'plane', 'lhr', 'review', 'airways', 'flights'))

# Combine with default stopwords
updated_stopwords <- stop_words %>%
  bind_rows(custom_stopwords)

# Check the updated stopwords
print(updated_stopwords)
```

```
## # A tibble: 1,163 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
## # i 1,153 more rows
```

```
#We can remove these stop words from word column by performing an anti-join between tokens and stop_wor

review_tokens <- tokens %>% anti_join(updated_stopwords)
```

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

```
review_tokens
```

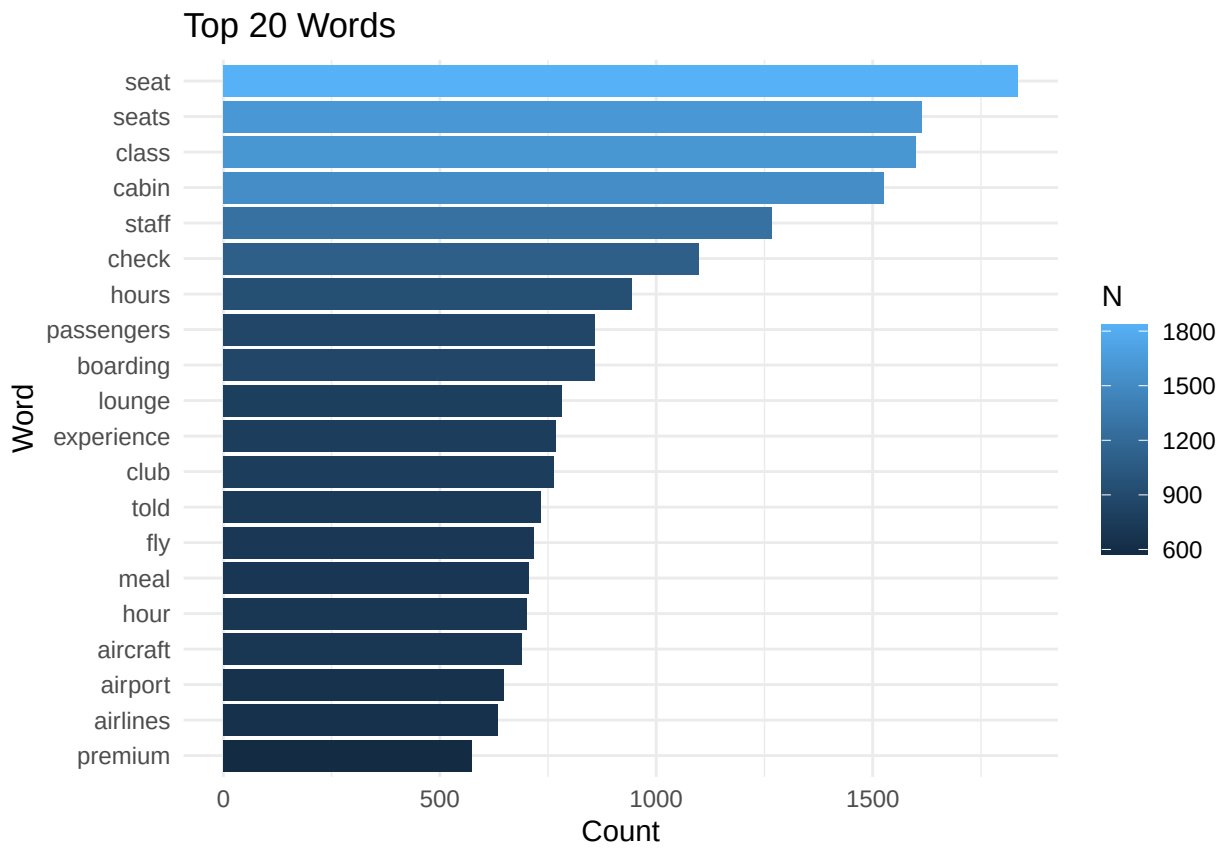
```
##         review_no      word
##             <int>    <char>
##      1:         1    cyprus
```

```
##      2:         1     cabin
##      3:         1    closed
##      4:         1     pilot
##      5:         1 announced
##     ---
## 123897:      3000   airport
## 123898:      3000   arrived
## 123899:      3000      home
## 123900:      3000  midnight
## 123901:      3000  starving
```

```
#Now, when we count the words, we will only get the meaningful words;
review_tokens[, .N, word][order(-N)]
```

```
##              word     N
##            <char> <int>
##    1:        seat  1836
##    2:       seats  1613
##    3:       class  1599
##    4:       cabin  1527
##    5:       staff  1266
##   ---
## 8761: undertaking     1
## 8762:    appaling     1
## 8763:      arabia     1
## 8764:     degrade     1
## 8765:     cheddar     1
```

```
review_tokens[, .N, word][order(-N)] %>% head(20) %>% ggplot(aes(N, reorder(word, N))) + geom_col(aes(f
  labs(x = "Count", y = "Word", title = "Top 20 Words") +
  theme_minimal()
```




```
# Sentiment Analysis
#The tidytext library provides the get_sentiments function. This can be used to download one of many di

sentiments<- get_sentiments("nrc")
sentiments
```

```
## # A tibble: 13,872 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abacus      trust
##  2 abandon     fear
##  3 abandon     negative
##  4 abandon     sadness
##  5 abandoned   anger
##  6 abandoned   fear
##  7 abandoned   negative
##  8 abandoned   sadness
##  9 abandonment anger
## 10 abandonment fear
## # i 13,862 more rows
```

```
# We can classify each word by sentiment by joining together the tidy text sherlock tibble with the dic

review_sentiments <- review_tokens %>% inner_join(sentiments)
```

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

```
## Warning in inner_join(., sentiments): Detected an unexpected many-to-many relationship between `x` ar
## i Row 4 of `x` matches multiple rows in `y`.
## i Row 715 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
review_sentiments
```

```
##        review_no      word    sentiment
##            <int>    <char>       <char>
##     1:         1     pilot     positive
##     2:         1     pilot        trust
##     3:         1      hell        anger
##     4:         1      hell      disgust
##     5:         1      hell         fear
##    ---
## 66679:      3000 appalling     negative
## 66680:      3000       hot        anger
## 66681:      3000       eat     positive
## 66682:      3000   airport anticipation
## 66683:      3000  starving     negative
```

```
#We can then count how many words of different sentiments there are using filter and count.
review_sentiments %>% filter(sentiment =="positive") %>% count(word, sort = TRUE)
```

```
##              word     n
##            <char> <int>
##   1:        flying   535
##   2:      friendly   462
##   3: entertainment   441
##   4:     breakfast   428
##   5:           pay   421
##  ---
## 726:    vocabulary     1
## 727:      warranty     1
## 728:     wholesome     1
## 729:      wireless     1
## 730:   wonderfully     1
```

```
# We could get the total number of words of each type using;
review_sentiments %>% filter(sentiment =="positive") %>% count(word, sort = TRUE) %>% summarise(total= s
```

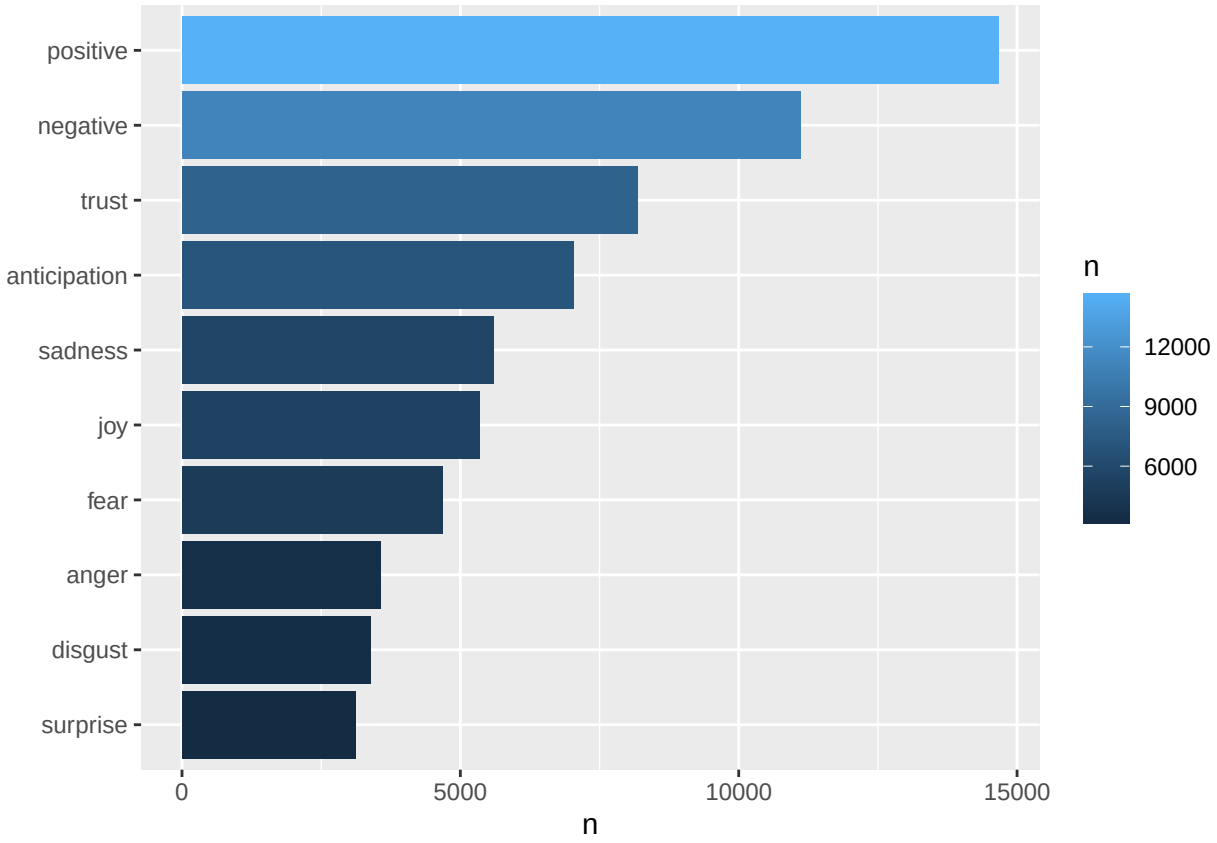
```
##   total
## 1 14670
```

```
#Alternatively (and much more simply) we could just count the number of occurrences of each sentiment i
review_sentiments %>% count(sentiment)
```

```
##        sentiment     n
##           <char> <int>
##  1:        anger  3558
```

```
##  2: anticipation  7034
##  3:      disgust  3391
##  4:         fear  4673
##  5:          joy  5351
##  6:     negative 11107
##  7:     positive 14670
##  8:      sadness  5594
##  9:     surprise  3121
## 10:        trust  8184
```

```
#We could plot this using a similar technique as the last section (this time converting the sentiments
review_sentiments %>% count(sentiment) %>% ggplot(aes(n, reorder(sentiment, n))) + geom_col(aes(fill= n)
```




```
#What we want to do now is to count the number of words with different sentiments for differnet review
review_sentiments_pivot<- review_sentiments %>% count(review_no,sentiment)

#Next, we can now pivot the tibble so that the sentiments are the columns, and we have one row per bloc

review_sentiments_pivot <- review_sentiments_pivot %>% pivot_wider(names_from = sentiment, values_from =
review_sentiments_pivot
```

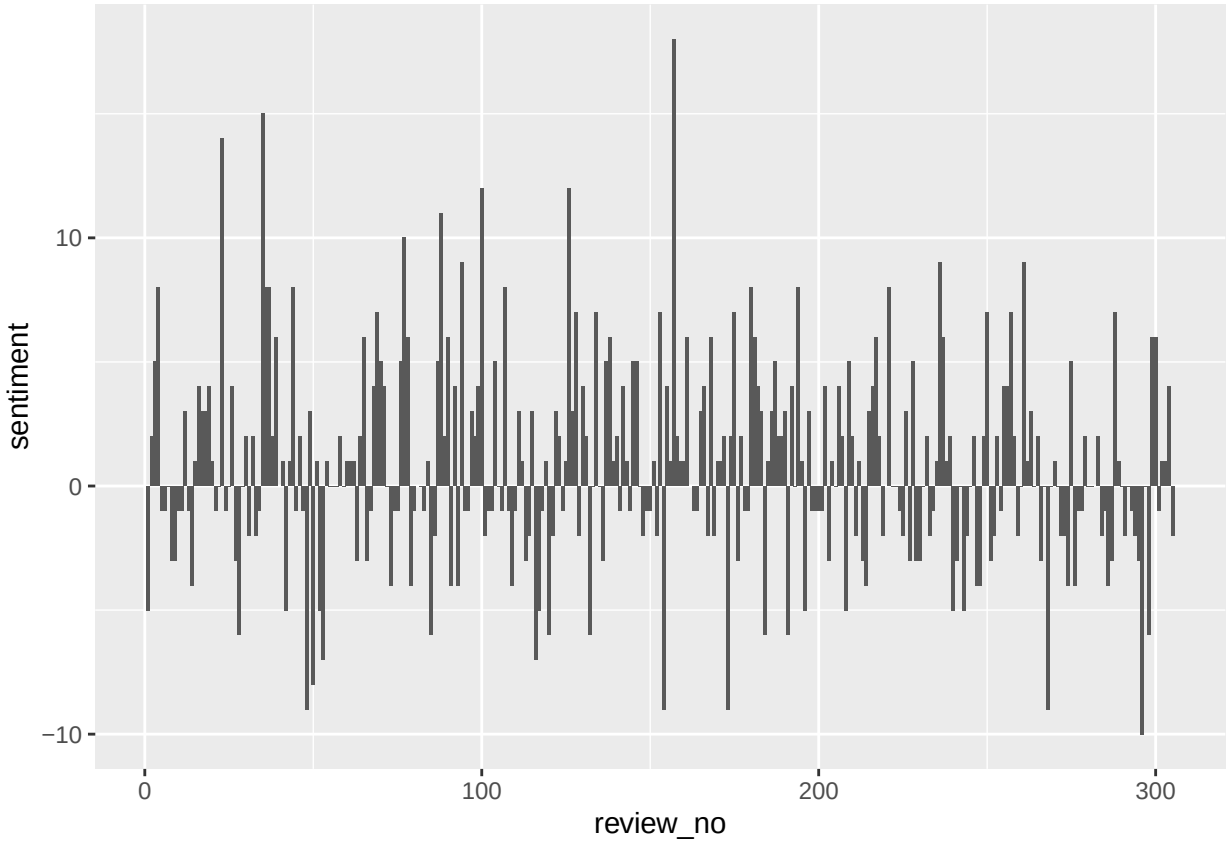
```
## # A tibble: 2,983 x 11
##    review_no anger anticipation disgust  fear negative positive sadness surprise
##        <int> <int>        <int>   <int> <int>    <int>    <int>   <int>    <int>
##  1         1     4            3       3     4        7        2       5        1
```

```
##  2         2     2            5       2     2        5        7       4        3
##  3         3     0            3       1     0        1        6       1        0
##  4         4     3            4       0     4        9       17       3        1
##  5         5     2            4       3     4        6        5       3        0
##  6         6     3            2       4     3        4        3       4        0
##  7         7     1            0       1     2        2        2       1        0
##  8         8     4            2       4     4        5        2       2        1
##  9         9     3            0       2     2        4        1       2        1
## 10        10     2            1       2     5        4        3       3        0
## # i 2,973 more rows
## # i 2 more variables: trust <int>, joy <int>
```

```
#We can now do things like calculate the difference between the number of positive and negative sentimes
review_sentiments_pivot <- review_sentiments_pivot %>% mutate(sentiment = positive - negative)
review_sentiments_pivot
```

```
## # A tibble: 2,983 x 12
##    review_no anger anticipation disgust  fear negative positive sadness surprise
##        <int> <int>        <int>   <int> <int>    <int>    <int>   <int>    <int>
##  1         1     4            3       3     4        7        2       5        1
##  2         2     2            5       2     2        5        7       4        3
##  3         3     0            3       1     0        1        6       1        0
##  4         4     3            4       0     4        9       17       3        1
##  5         5     2            4       3     4        6        5       3        0
##  6         6     3            2       4     3        4        3       4        0
##  7         7     1            0       1     2        2        2       1        0
##  8         8     4            2       4     4        5        2       2        1
##  9         9     3            0       2     2        4        1       2        1
## 10        10     2            1       2     5        4        3       3        0
## # i 2,973 more rows
## # i 3 more variables: trust <int>, joy <int>, sentiment <int>
```

```
#This could then be graphed, with the "sentiment" column on the y axis, and the review no on the x axis
# checking for first 300 reviews
review_sentiments_pivot %>% head(300) %>% ggplot(aes(review_no, sentiment)) + geom_col()
```




```
#Word clouds
#We are going to create word clouds from the text reviews
# The wordcloud function is very easy to use. You just need to pass in a column of words (review_tokens
review_tokens_wordcloud <- review_tokens[, .N, word][order(-N)]
suppressWarnings(review_tokens_wordcloud %>% with(wordcloud(word,N, min.freq = 50,random.order=FALSE,co
```


```
#n-grams
#It is very common that you want to tokenise using tokens that represent pairs of words. This is becaus
#A token comprising n words is called an "n-gram" (or "ngram"). Tokenising on bigrams or n-grams enable

Review_ngrams <- Review_text %>% unnest_tokens(ngram, text, token = "ngrams", n=3)
Review_ngrams
```

```
## # A tibble: 493,910 x 2
##    review_no ngram
##        <int> <chr>
##  1         1 on a recent
##  2         1 a recent flight
##  3         1 recent flight from
##  4         1 flight from cyprus
##  5         1 from cyprus ba621
##  6         1 cyprus ba621 on
##  7         1 ba621 on 23
##  8         1 on 23 11
##  9         1 23 11 24
## 10         1 11 24 the
## # i 493,900 more rows
```

```
#In this case we have placed the ngrams into the column ngram.

#Now we want to remove pairs of words where one in the pair is a stop word. To do this we must first se
```

```
Review_ngrams <- Review_ngrams %>% separate(ngram, c("word1", "word2", "word3"), sep = " ")
Review_ngrams
```

```
## # A tibble: 493,910 x 4
##    review_no word1  word2  word3
##        <int> <chr>  <chr>  <chr>
##  1         1 on     a      recent
##  2         1 a      recent flight
##  3         1 recent flight from
##  4         1 flight from   cyprus
##  5         1 from   cyprus ba621
##  6         1 cyprus ba621  on
##  7         1 ba621  on     23
##  8         1 on     23     11
##  9         1 23     11     24
## 10         1 11     24     the
## # i 493,900 more rows
```

```
#Next we need to filter only the rows where neither word1 or word2 or word3 have a word in stop_words.
Review_ngrams <- Review_ngrams %>%
                 filter(!word1 %in% updated_stopwords$word) %>%
                 filter(!word2 %in% updated_stopwords$word) %>%
                 filter(!word3 %in% updated_stopwords$word)

Review_ngrams
```

```
## # A tibble: 17,687 x 4
##    review_no word1  word2   word3
##        <int> <chr>  <chr>   <chr>
##  1         1 23     11      24
##  2         1 6      hours   due
##  3         1 2      hours   waiting
##  4         1 8      hour    hell
##  5         1 3      oatmeal biscuits
##  6         1 4      cabin   crew
##  7         1 23     11      24
##  8         1 email  system  unsurprisingly
##  9         2 sunday 15      december
## 10         4 chosen wines   disappointingly
## # i 17,677 more rows
```

```
#Next, we can see that there are a lot of NAs in the tibble. These are caused by blank lines or other t
Review_ngrams <- Review_ngrams %>%
                     filter(!is.na(word1)) %>%
                     filter(!is.na(word2)) %>%
                     filter(!is.na(word3))
Review_ngrams
```

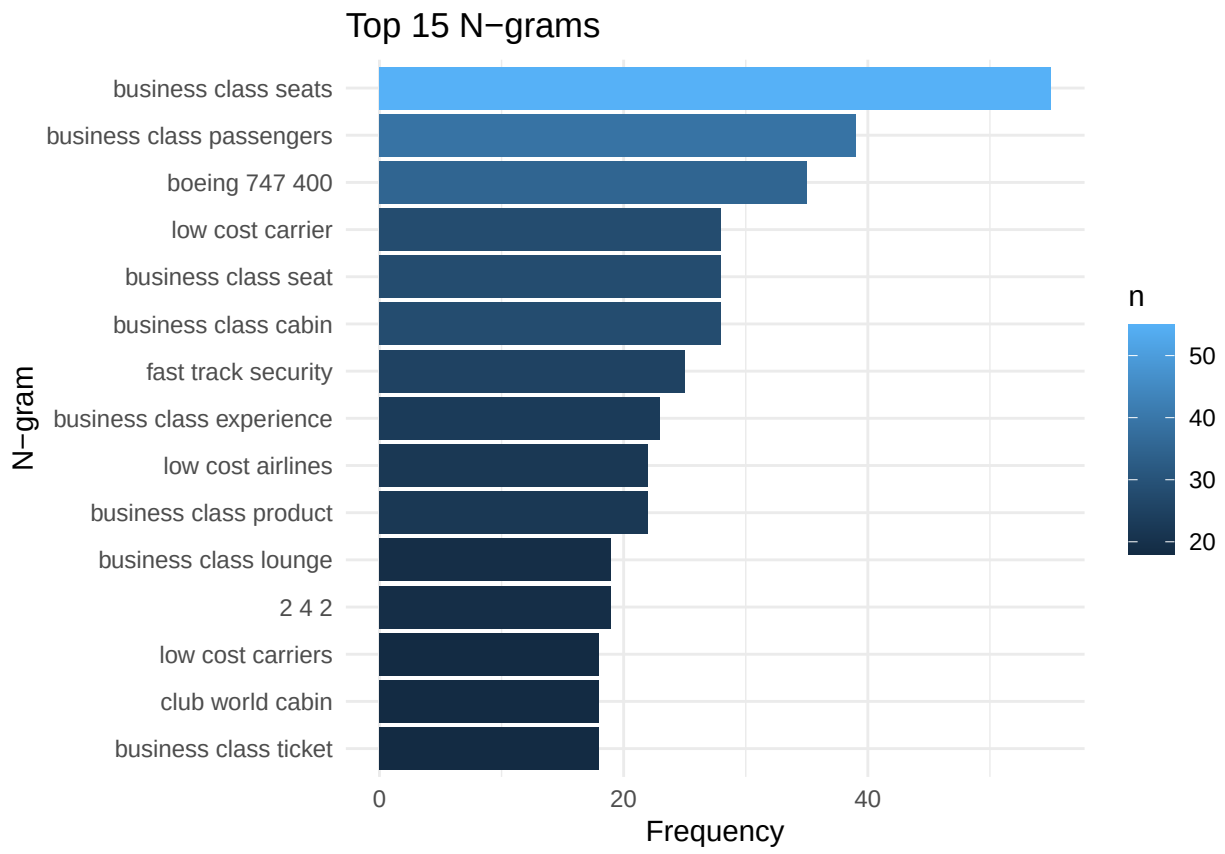
```
## # A tibble: 17,687 x 4
##    review_no word1  word2   word3
##        <int> <chr>  <chr>   <chr>
```

```
##  1         1 23     11      24
##  2         1 6      hours   due
##  3         1 2      hours   waiting
##  4         1 8      hour    hell
##  5         1 3      oatmeal biscuits
##  6         1 4      cabin   crew
##  7         1 23     11      24
##  8         1 email  system  unsurprisingly
##  9         2 sunday 15      december
## 10         4 chosen wines   disappointingly
## # i 17,677 more rows
```

```
#Finally! we will rejoin the word1 and word2 columns into a single "ngram" column, using the "unite" fu
Review_ngrams <- Review_ngrams %>% unite(ngram, word1,word2, word3, sep = " ")
Review_ngrams
```

```
## # A tibble: 17,687 x 2
##    review_no ngram
##        <int> <chr>
##  1         1 23 11 24
##  2         1 6 hours due
##  3         1 2 hours waiting
##  4         1 8 hour hell
##  5         1 3 oatmeal biscuits
##  6         1 4 cabin crew
##  7         1 23 11 24
##  8         1 email system unsurprisingly
##  9         2 sunday 15 december
## 10         4 chosen wines disappointingly
## # i 17,677 more rows
```

```
#We can now count the ngrams as before, e.g.
ngram_counts <- Review_ngrams %>% count(ngram, sort=TRUE)
ngram_counts %>%
  head(15) %>%
  ggplot(aes(n, reorder(ngram, n))) + # Reorder ngram based on n in descending order
  geom_col(aes(fill= n)) +
  labs(x = "Frequency", y = "N-gram", title = "Top 15 N-grams") +
  theme_minimal()
```




```
# Visualising correlations between words
#We can go beyond just counting, and can draw graphs that visualise the connections and correlations be

Review_ngrams_count <- Review_text %>% unnest_tokens(ngram, text, token = "ngrams", n=3) %>%
  separate(ngram, c("word1", "word2", "word3"), sep = " ") %>%
                  filter(!word1 %in% updated_stopwords$word) %>%
                  filter(!word2 %in% updated_stopwords$word) %>%
                  filter(!word3 %in% updated_stopwords$word) %>%
                      filter(!is.na(word1)) %>%
                      filter(!is.na(word2)) %>%
                      filter(!is.na(word3)) %>%
                      count(word1, word2, word3, sort = TRUE)

Review_ngrams_count
```

```
## # A tibble: 15,604 x 4
##    word1    word2 word3          n
##    <chr>    <chr> <chr>      <int>
##  1 business class seats         55
##  2 business class passengers    39
##  3 boeing   747   400           35
##  4 business class cabin         28
##  5 business class seat          28
##  6 low      cost  carrier       28
##  7 fast     track security      25
```

```
##  8 business class experience    23
##  9 business class product       22
## 10 low      cost  airlines      22
## # i 15,594 more rows
```

```
#This table can now be converted into a directed graph. A directed graph is one where nodes (in this ca

#To prevent the graph becoming too large, we will only graph pairs of words that appear seven or more t
install.packages("igraph")
```

```
##
## The downloaded binary packages are in
##  /var/folders/fm/k4r2n81j77q8l80mh7tnvq2r0000gn/T//RtmpXdsoRR/downloaded_packages
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.3.3
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

```
## The following object is masked from 'package:tidyr':
##
##     crossing
```

```
## The following object is masked from 'package:tibble':
##
##     as_data_frame
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

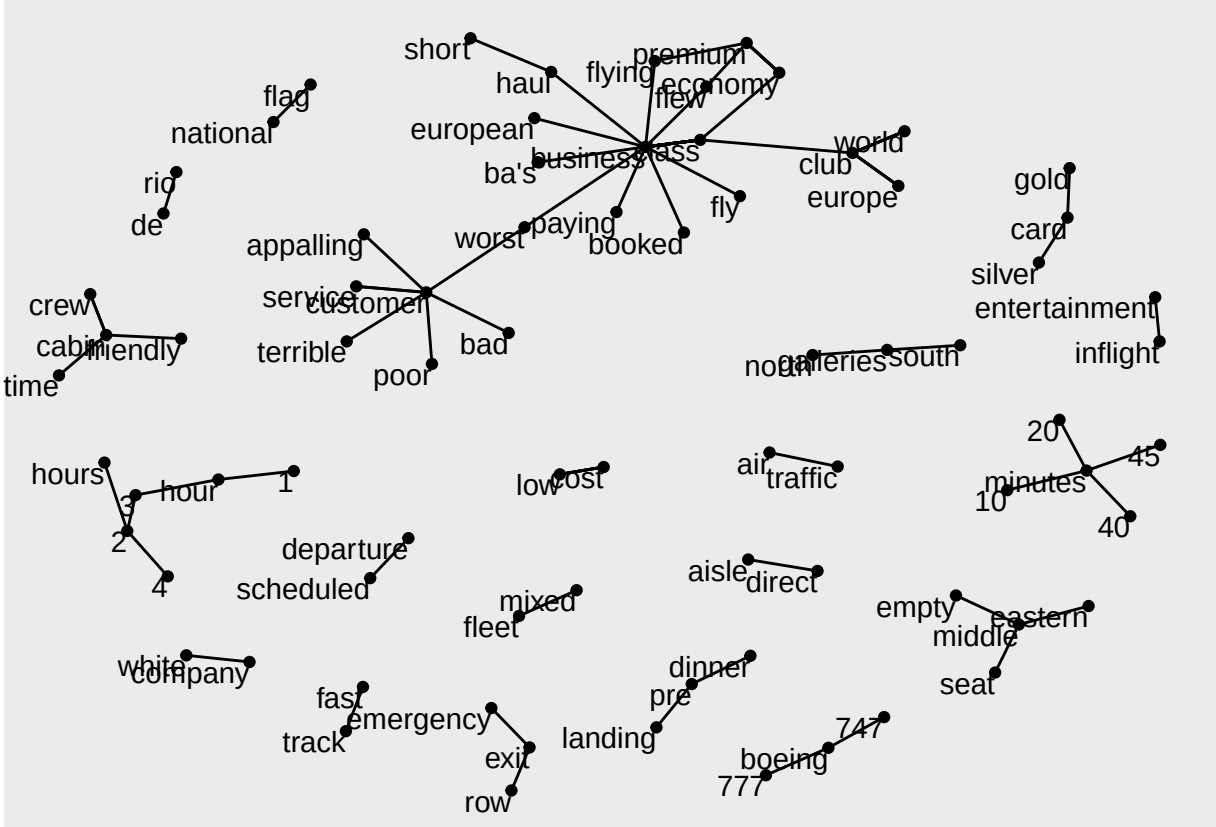
```
ngram_graph <- Review_ngrams_count %>% filter(n > 5) %>%
  graph_from_data_frame()
ngram_graph
```

```
## IGRAPH 25e6669 DN-- 77 85 --
## + attr: name (v/c), word3 (e/c), n (e/n)
## + edges from 25e6669 (vertex names):
##  [1] business->class    business->class    boeing  ->747      business->class
##  [5] business->class    low     ->cost     fast    ->track    business->class
##  [9] business->class    low     ->cost     2       ->4        business->class
## [13] business->class    club    ->world    low     ->cost     worst   ->business
## [17] premium ->economy  gold    ->card     business->class    club    ->world
## [21] haul    ->business club    ->europe   premium ->economy  premium ->economy
## [25] 3       ->3        booked  ->business cabin   ->crew     economy ->class
## [29] middle  ->seat     poor    ->customer boeing  ->777      business->class
## + ... omitted several edges
```

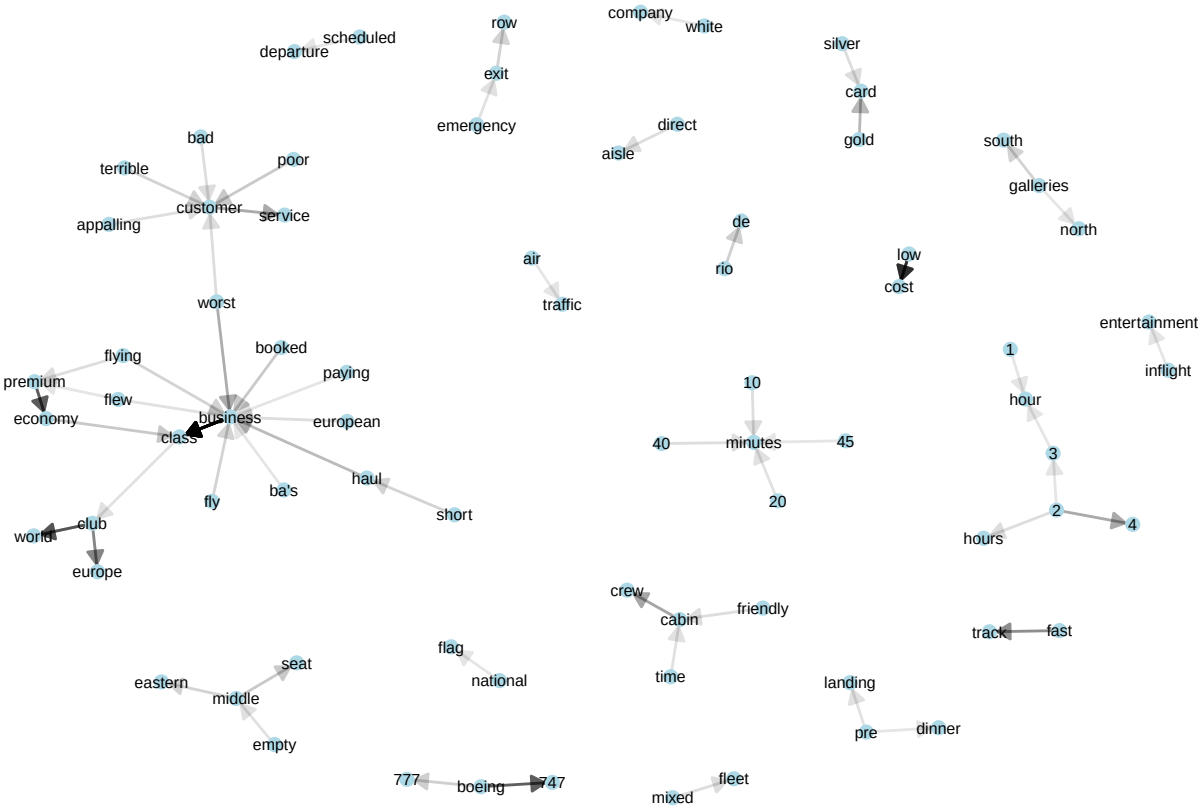
```
#The graph_from_data_frame takes a tibble where the first three columns name the nodes and specify thei

#Now that we have built the directed graph, we can visualise it using the ggraph function from the ggra

#There is a grammar for these graphs, e.g. we pass the data into ggraph, and then add layers, such as g
install.packages("ggraph")
```

```
##
## The downloaded binary packages are in
##  /var/folders/fm/k4r2n81j77q8l80mh7tnvq2r0000gn/T//RtmpXdsoRR/downloaded_packages
```

```
library(ggraph)

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




```
#This graph isn't very pretty. By looking online at the options to improve ggraph, and comparing agains

ggraph(ngram_graph, layout = "fr") +
     geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                    arrow = grid::arrow(type = "closed", length = unit(2, "mm")),
                    end_cap = circle(1, "mm")) +
     geom_node_point(color = "lightblue", size = 2) +
     geom_node_text(aes(label = name), size = 2) +
     theme_void()
```


scheduled
departure
row
exit
emergency
company
white
silver
card
gold
direct
aisle
south
galleries
north
bad
terrible
poor
customer
service
appalling
worst
de
rio
air
traffic
low
cost
entertainment
inflight
1
hour
3
2
4
hours
flying
premium
economy
flew
booked
paying
business
class
european
haul
short
ba's
fly
club
world
europe
10
40
minutes
45
20
crew
cabin
friendly
time
track
fast
flag
national
seat
eastern
middle
empty
landing
pre
dinner
777
boeing
747
fleet
mixed