Lab 5

Conie O'Malley

2025-02-16

Table of contents

```r
# install packages
packages <- c("quanteda", "quanteda.textmodels", "quanteda.textstats", "quanteda.textplots",


for (i in packages) {
    if (!requireNamespace(i, quietly = TRUE)) {
        renv::install(i)
    }
    library(i, character.only = TRUE)  # Load the package
}
```

```
Warning in .recacheSubclasses(def@className, def, env): undefined subclass
"ndiMatrix" of class "replValueSp"; definition not updated
```

```
Warning: package 'quanteda' was built under R version 4.3.3
```

```
Package version: 4.2.0
Unicode version: 14.0
ICU version: 71.1
```

```
Parallel computing: disabled
```

```
See https://quanteda.io for tutorials and examples.
```

```
Warning: package 'quanteda.textmodels' was built under R version 4.3.3

Warning in .recacheSubclasses(def@className, def, env): undefined subclass
"ndiMatrix" of class "replValueSp"; definition not updated

Warning: package 'quanteda.textstats' was built under R version 4.3.3

Warning: package 'quanteda.textplots' was built under R version 4.3.3

Warning: package 'textdata' was built under R version 4.3.3

Warning: package 'wordcloud' was built under R version 4.3.3

Loading required package: RColorBrewer


Attaching package: 'readtext'

The following object is masked from 'package:quanteda':

    texts
```

```r
remotes::install_github("quanteda/quanteda.sentiment")
```

```
Using GitHub PAT from the git credential store.

Skipping install of 'quanteda.sentiment' from a github remote, the SHA1 (934c1e1f) has not cl
  Use `force = TRUE` to force installation
```

```r
remotes::install_github("quanteda/quanteda.tidy")
```

```
Using GitHub PAT from the git credential store.

Skipping install of 'quanteda.tidy' from a github remote, the SHA1 (c3c28f0f) has not change
  Use `force = TRUE` to force installation
```

```r
renv::install("reshape2")
```

```
The following package(s) will be installed:
- reshape2 [1.4.4]
These packages will be installed into "~/Library/Caches/org.R-project.R/R/renv/library/AdvTx

# Installing packages --------------------------------------------------------
- Installing reshape2 ...                       OK [linked from cache]
Successfully installed 1 package in 8.8 milliseconds.
```

```r
library(tm)
```

```
Warning: package 'tm' was built under R version 4.3.3
```

```
Loading required package: NLP
```

```
Warning: package 'NLP' was built under R version 4.3.3
```

```

Attaching package: 'NLP'
```

```
The following objects are masked from 'package:quanteda':

    meta, meta<-
```

```

Attaching package: 'tm'
```

```
The following object is masked from 'package:quanteda':

    stopwords
```

```r
library(tidyverse)
```

```
Warning: package 'lubridate' was built under R version 4.3.3
```

```
-- 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   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.2
```

```
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x ggplot2::annotate() masks NLP::annotate()
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 becom
```

```r
library(tidytext)
library(reshape2)
```

```
Attaching package: 'reshape2'

The following object is masked from 'package:tidyr':

    smiths
```

```r
library(janeaustenr)
```

```
Warning: package 'janeaustenr' was built under R version 4.3.3
```

```r
library(reticulate)

#use_condaenv("datascience", required = FALSE) # set my environment
```

1 Part 1: Data Preparation, Text Mining and Dictionary Development in tm

1.1 Deliverable 1: Get your working directory and paste below:

```
getwd()
```

```
[1] "/Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Main/023. Prog
```

1.2 Deliverable 2: Create Files For Use from Reuters

```
reut21578 <- system.file("texts","crude", package = "tm")
```

1.3 Deliverable 3: Create VCorpus Object

```
reuters <- VCorpus(DirSource(reut21578,mode = "binary"),
                            readerControl = list(reader=readReut21578XMLasPlain))
reuters
```

```
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 20
```

1.4 Deliverable 4: Prepare and Preprocess the Corpus

```
reuters <- tm_map(reuters, content_transformer(tolower)) # make all text lowercase

reuters <- tm_map(reuters, removeWords, tm::stopwords("english")) # remove stopwords

myStopwords = c(tm::stopwords(),"") # alternate preprocessing method
tdm3 = TermDocumentMatrix(reuters,
                        control = list(weighting = weightTfIdf,
                        stopwords = myStopwords,
                        removePunctuation = T,
                        removeNumbers = T,
                        stemming = T))
inspect(tdm3)
```

```
<<TermDocumentMatrix (terms: 781, documents: 20)>>
Non-/sparse entries: 1501/14119
Sparsity           : 90%
Maximal term length: 16
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
Sample             :
         Docs
Terms            144        211         237        242        246        273
  billion 0.00000000 0.00000000 0.000000000 0.00000000 0.12501880 0.00000000
  crude   0.00000000 0.00000000 0.000000000 0.00000000 0.00000000 0.02380172
  grade   0.00000000 0.00000000 0.000000000 0.00000000 0.00000000 0.00000000
  januari 0.00000000 0.00000000 0.000000000 0.00000000 0.00000000 0.05490790
  mln     0.01752096 0.04266678 0.004430781 0.00000000 0.00000000 0.04284309
  opec    0.05703422 0.00000000 0.003846154 0.02298851 0.01075269 0.02066116
  post    0.00000000 0.00000000 0.000000000 0.00000000 0.00000000 0.00000000
  reserv  0.00000000 0.12899601 0.000000000 0.00000000 0.01248348 0.00000000
  saudi   0.00000000 0.00000000 0.000000000 0.02298851 0.00000000 0.05785124
  west    0.00000000 0.00000000 0.000000000 0.00000000 0.00000000 0.00000000
         Docs
Terms     368        502 704        708
  billion    0 0.00000000   0 0.19540753
  crude      0 0.00000000   0 0.03388244
  grade      0 0.00000000   0 0.00000000
  januari    0 0.00000000   0 0.39081507
  mln        0 0.03170651   0 0.06776489
  opec       0 0.00000000   0 0.00000000
  post       0 0.00000000   0 0.00000000
  reserv     0 0.06390628   0 0.00000000
  saudi      0 0.00000000   0 0.00000000
  west       0 0.00000000   0 0.00000000
```

1.5 Deliverable 5: Create Document Term Matrix with TF and TF*IDF

```r
dtm <- DocumentTermMatrix(reuters) # create dtm
inspect(dtm)
```

```
<<DocumentTermMatrix (documents: 20, terms: 1183)>>
Non-/sparse entries: 1908/21752
Sparsity           : 92%
Maximal term length: 17
```

```
Weighting          : term frequency (tf)
Sample             :
     Terms
Docs  crude dlrs last mln oil opec prices reuter said saudi
  144     0    0    1   4  11   10      3      1    9     0
  236     1    2    4   4   7    6      2      1    6     0
  237     0    1    3   1   3    1      0      1    0     0
  242     0    0    0   0   3    2      1      1    3     1
  246     0    0    2   0   4    1      0      1    4     0
  248     0    3    1   3   9    6      7      1    5     5
  273     5    2    7   9   5    5      4      1    5     7
  489     0    1    0   2   4    0      2      1    2     0
  502     0    1    0   2   4    0      2      1    2     0
  704     0    0    0   0   3    0      2      1    3     0
```

```
dtm2 <- DocumentTermMatrix(reuters, control = list(weighting=weightTfIdf)) # dtm with idf we
inspect(dtm2)
```

```
<<DocumentTermMatrix (documents: 20, terms: 1183)>>
Non-/sparse entries: 1868/21792
Sparsity           : 92%
Maximal term length: 17
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
Sample             :
     Terms
Docs  1.50    billion       crude   january         mln        opec posted
  144    0 0.00000000 0.000000000 0.0000000 0.017258473 0.037453184      0
  211    0 0.00000000 0.000000000 0.0000000 0.041891022 0.000000000      0
  236    0 0.00000000 0.004314618 0.0000000 0.017258473 0.022471910      0
  237    0 0.00000000 0.000000000 0.0000000 0.004314618 0.003745318      0
  242    0 0.00000000 0.000000000 0.0000000 0.000000000 0.020618557      0
  246    0 0.09770377 0.000000000 0.0000000 0.000000000 0.004901961      0
  349    0 0.00000000 0.034909185 0.0000000 0.000000000 0.015151515      0
  368    0 0.00000000 0.000000000 0.0000000 0.000000000 0.000000000      0
  704    0 0.00000000 0.000000000 0.0000000 0.000000000 0.000000000      0
  708    0 0.14764125 0.025600069 0.2952825 0.051200137 0.000000000      0
     Terms
Docs    power      saudi west
  144 0.000000 0.00000000    0
  211 0.000000 0.00000000    0
  236 0.000000 0.00000000    0
  237 0.000000 0.00000000    0
  242 0.000000 0.02061856    0
  246 0.000000 0.00000000    0
  349 0.000000 0.03030303    0
  368 0.261935 0.00000000    0
  704 0.000000 0.00000000    0
  708 0.000000 0.00000000    0
```

1.6 Deliverable 6: Find the Most Frequent Terms

```r
findFreqTerms(dtm,5) # find all terms mentioned > 5 times
```

```
 [1] "15.8"          "abdul-aziz"    "ability"       "accord"
 [5] "agency"        "agreement"     "ali"           "also"
 [9] "analysts"      "arab"          "arabia"        "barrel."
[13] "barrels"       "billion"       "bpd"           "budget"
[17] "company"       "crude"         "daily"         "demand"
[21] "dlrs"          "economic"      "emergency"     "energy"
[25] "exchange"      "expected"      "exports"       "futures"
[29] "government"    "group"         "gulf"          "help"
[33] "hold"          "industry"      "international" "january"
[37] "kuwait"        "last"          "market"        "may"
[41] "meeting"       "minister"      "mln"           "month"
[45] "nazer"         "new"           "now"           "nymex"
[49] "official"      "oil"           "one"           "opec"
[53] "output"        "pct"           "petroleum"     "plans"
[57] "posted"        "present"       "price"         "prices"
[61] "prices,"       "prices."       "production"    "quota"
[65] "quoted"        "recent"        "report"        "research"
[69] "reserve"       "reuter"        "said"          "said."
[73] "saudi"         "sell"          "sheikh"        "sources"
[77] "study"         "traders"       "u.s."          "united"
[81] "west"          "will"          "world"
```

1.7 Deliverable 7: Find Terms Associated with a Specific Term

```r
findAssocs(dtm, "opec", 0.8) # find terms associated with "opec"
```

```
$opec
```

```
  meeting emergency       oil      15.8  analysts    buyers      said   ability
     0.88      0.87      0.87      0.85      0.85      0.83      0.82      0.80
```

```r
findAssocs(dtm2, "opec", 0.8) # find terms associated with "opec"
```

```
$opec
emergency   meeting  analysts     quota
     0.85      0.85      0.84      0.81
```

1.7.0.1 Which do you find more useful?

The weighted version weeds out certain words that may not be critical to analysis, like “said”, “oil”, and “15.8”. Presumably “oil” will be highly relational to opec (since its the first word of the opec acronym), said is a verb likely to come after the mention of opec, and 15.8 is a unknown float. The `TF*IDF` weighting method helps to reduce noise in the data like these terms above.

1.8 Deliverable 8: Remove Sparse Terms

```r
inspect(removeSparseTerms(dtm, 0.4)) # remove sparse terms
```

```
<<DocumentTermMatrix (documents: 20, terms: 3)>>
Non-/sparse entries: 58/2
Sparsity           : 3%
Maximal term length: 6
Weighting          : term frequency (tf)
Sample             :
     Terms
Docs  oil reuter said
  127   5      1    1
  144  11      1    9
  236   7      1    6
  242   3      1    3
  246   4      1    4
  248   9      1    5
  273   5      1    5
  352   5      1    1
  489   4      1    2
  502   4      1    2
```

```r
inspect(removeSparseTerms(dtm2, 0.4)) # remove sparse terms
```

```
<<DocumentTermMatrix (documents: 20, terms: 1)>>
Non-/sparse entries: 18/2
Sparsity           : 10%
Maximal term length: 4
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
Sample             :
     Terms
Docs         said
  144 0.005123700
  191 0.003800077
  194 0.003234108
  211 0.008291078
  236 0.003415800
  242 0.004701127
  248 0.003518590
  368 0.004606154
  489 0.003415800
  543 0.005241486
```

1.9 Deliverable 9: Develop a Simple Dictionary in tm

```r
inspect(DocumentTermMatrix(reuters, list(dictionary = c("prices","crude","oil"))))
```

```
<<DocumentTermMatrix (documents: 20, terms: 3)>>
Non-/sparse entries: 41/19
Sparsity           : 32%
Maximal term length: 6
Weighting          : term frequency (tf)
Sample             :
     Terms
Docs  crude oil prices
  127     2   5      3
  144     0  11      3
  236     1   7      2
  248     0   9      7
  273     5   5      4
  352     0   5      4
  353      2   4       1
  489      0   4       2
  502      0   4       2
  543      2   2       2
```

2 Part 2: Understanding Tidyverse Dictionary Construction and Sentiment Analysis

```
sentiments
```

```
# 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
# i 6,776 more rows
```

```
head(sentiments)
```

```
# A tibble: 6 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
```

```r
tail(sentiments)
```

```
# A tibble: 6 x 2
  word      sentiment
  <chr>     <chr>
1 zealous   negative
2 zealously negative
3 zenith    positive
4 zest      positive
5 zippy     positive
6 zombie    negative
```

```r
class(sentiments)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

2.1 Deliverable 10: Download Individual Lexicons within Sentiments

```r
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
# i 2,467 more rows
```

```r
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
# i 6,776 more rows
```

```r
get_sentiments("nrc")
```

```
# 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
```

2.2 Deliverable 11: Create an object called tidy_books from the janeaustenr package

```r
tidy_books <- janeaustenr::austen_books() %>%
    group_by(book) %>%
    mutate(linenumber = row_number(),
        chapter = cumsum(str_detect(text, regex("^chapter [\\divxlc]", ignore_case = TRUE))))
    ungroup() %>%
    unnest_tokens(word, text)
tidy_books
```

```
# A tibble: 725,055 x 4
   book                linenumber chapter word
   <fct>                    <int>   <int> <chr>
 1 Sense & Sensibility          1       0 sense
 2 Sense & Sensibility          1       0 and
 3 Sense & Sensibility          1       0 sensibility
 4 Sense & Sensibility          3       0 by
 5 Sense & Sensibility          3       0 jane
 6 Sense & Sensibility          3       0 austen
 7 Sense & Sensibility          5       0 1811
 8 Sense & Sensibility         10       1 chapter
 9 Sense & Sensibility         10       1 1
10 Sense & Sensibility         13       1 the
# i 725,045 more rows
```

2.3 Deliverable 12: Create nrcjoy Sentiment Dictionary

```
nrcjoy <- get_sentiments("nrc") %>%
    filter(sentiment == "joy")
nrcjoy
```

```
# A tibble: 687 x 2
   word          sentiment
   <chr>         <chr>
 1 absolution    joy
 2 abundance     joy
 3 abundant      joy
 4 accolade      joy
 5 accompaniment joy
 6 accomplish    joy
 7 accomplished  joy
 8 achieve       joy
 9 achievement   joy
10 acrobat       joy
# i 677 more rows
```

2.4 Deliverable 13: Applying NRC Joy Extract to Emma

```r
tidy_books %>%
    filter(book == "Emma") %>%
    inner_join(nrcjoy) %>%
    count(word, sort = TRUE)
```

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

```
# A tibble: 301 x 2
   word          n
   <chr>     <int>
 1 good        359
 2 friend      166
 3 hope        143
 4 happy       125
 5 love        117
 6 deal         92
 7 found        92
 8 present      89
 9 kind         82
10 happiness    76
# i 291 more rows
```

```r
tidy_books %>%
    filter(book == "Persuasion") %>%
    inner_join(nrcjoy) %>%
    count(word, sort = TRUE)
```

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

```
# A tibble: 256 x 2
   word        n
   <chr>   <int>
 1 good      187
 2 found      83
 3 friend     77
 4 present    65
 5 happy      64
 6 hope       53
 7 deal       45
 8 love       42
 9 spirits    41
10 feeling    37
# i 246 more rows
```

2.4.0.1 This result is interesting, but how does the book Emma compare to other books by Jane Austen on the specific sentiment of joy?

After reviewing the sentiment analysis for the book *Persuasion*, we can see that 8/10 words from *Emma* are on the list. We can infer that these novels have a level of similarity in terms of the emotions they evoke, but also must account for the Austen's writing style to account for some of the similarities.

2.5 Deliverable 14: Sentiment Analysis of Jane Austen Books

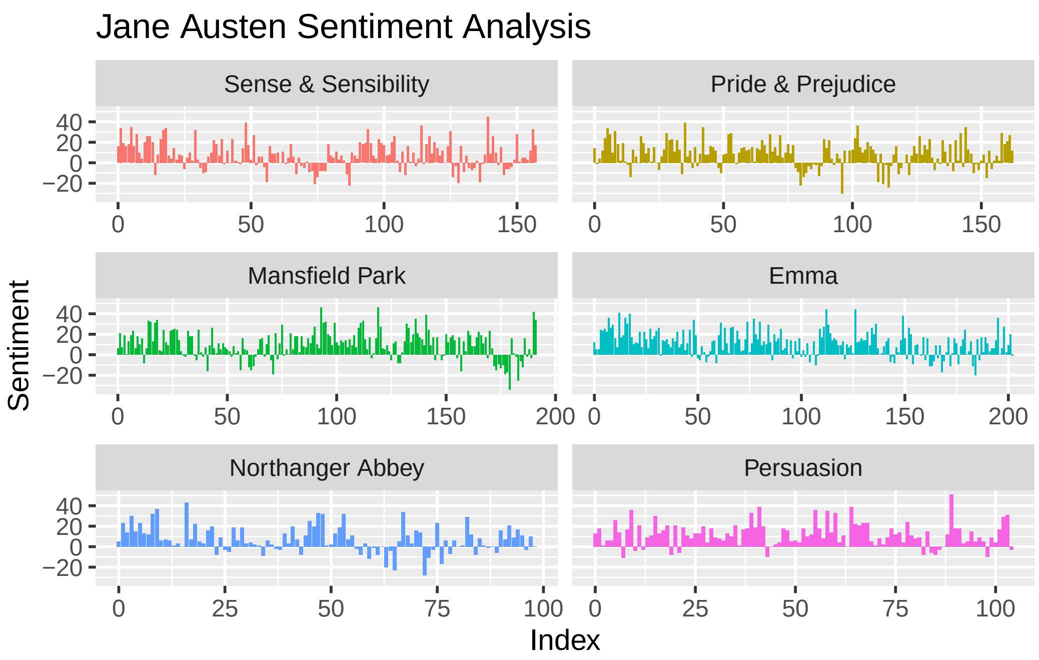
```
janeaustensentiment <- tidy_books %>%
    inner_join(get_sentiments("bing")) %>%
    count(book, index = linenumber %/% 80, sentiment) %>%
    spread(sentiment, n, fill = 0) %>%
    mutate(sentiment = positive - negative)
```

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

```
Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relati
i Row 435434 of `x` matches multiple rows in `y`.
i Row 5051 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.
```

2.6 Deliverable 15: Visualize Jane Austen Sentiment

```
ggplot(janeaustensentiment, aes(index, sentiment, fill = book)) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~book, ncol = 2, scales = "free_x") +
    labs(title = "Jane Austen Sentiment Analysis", x = "Index", y = "Sentiment")
```



2.7 Deliverable 16: Calculate and Visualize Sentiment and Words

```
bing_word_counts <- tidy_books %>%
    inner_join(get_sentiments("bing")) %>%
    count(word, sentiment, sort = TRUE) %>%
    ungroup()
```

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

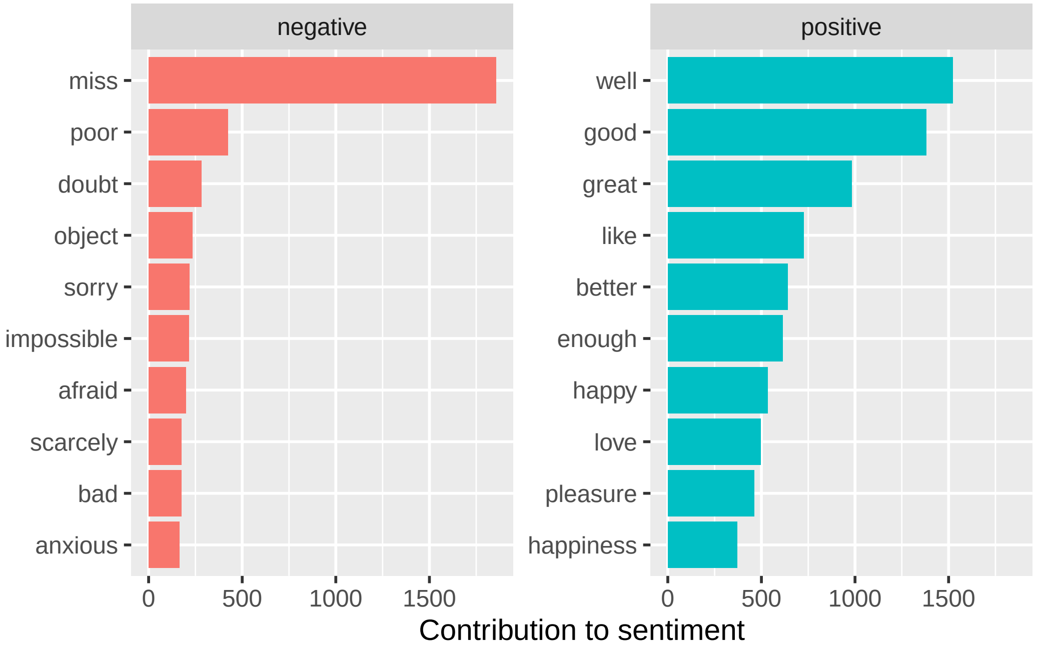
```
Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relati
i Row 435434 of `x` matches multiple rows in `y`.
i Row 5051 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```
bing_word_counts
```

```
# A tibble: 2,585 x 3
   word     sentiment     n
   <chr>    <chr>     <int>
 1 miss     negative   1855
 2 well     positive   1523
 3 good     positive   1380
 4 great    positive    981
 5 like     positive    725
 6 better   positive    639
 7 enough   positive    613
 8 happy    positive    534
 9 love     positive    495
10 pleasure positive    462
# i 2,575 more rows
```

```r
bing_word_counts %>%
    group_by(sentiment) %>%
    top_n(10) %>%
    ungroup() %>%
    mutate(word = reorder(word,n)) %>%
    ggplot(aes(word, n, fill = sentiment)) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~sentiment, scales = "free_y") +
    labs(y = "Contribution to sentiment", x = NULL) +
    coord_flip()
```

```
Selecting by n
```



2.8 Deliverable 17: Create a Custom Stopword Dictionary

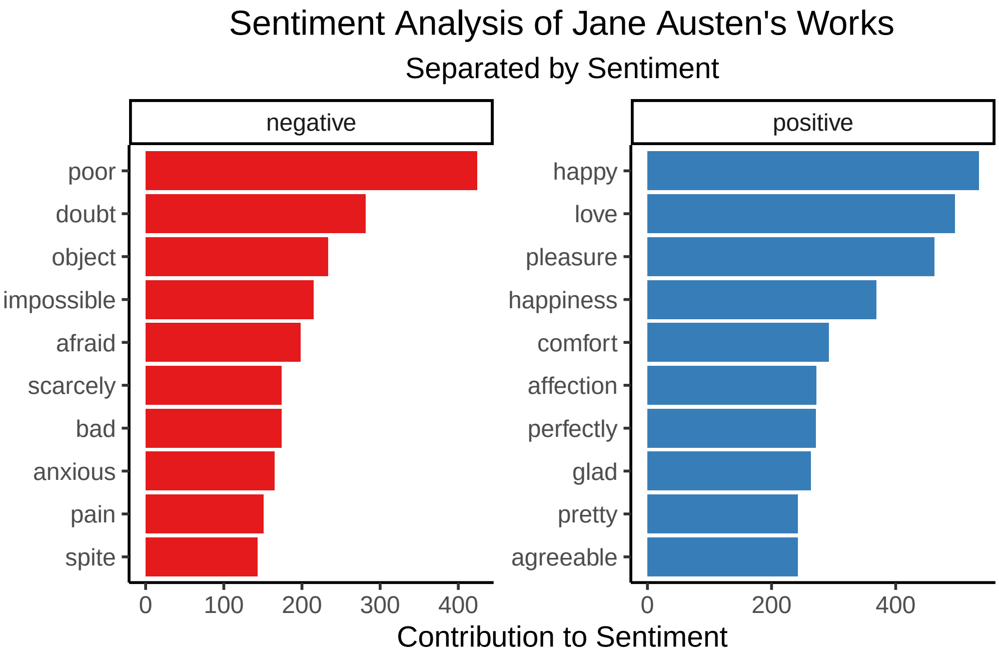
```
custom_stop_words <- bind_rows(tibble(word = c("miss"), lexicon = c("custom")), stop_words)
custom_stop_words
```

```
# A tibble: 1,150 x 2
   word        lexicon
   <chr>       <chr>  
 1 miss        custom 
 2 a           SMART  
 3 a's         SMART  
 4 able        SMART  
 5 about       SMART  
 6 above       SMART  
 7 according   SMART  
 8 accordingly SMART  
 9 across      SMART  
10 actually    SMART  
# i 1,140 more rows
```

2.9 Deliverable 18: Apply Custom Stopword Dictionary

```
bing_word_counts %>%
    anti_join(custom_stop_words) %>%
    group_by(sentiment) %>%
    top_n(10) %>% ungroup() %>%
    mutate(word = reorder(word, n)) %>%
    ggplot() +
    geom_col(aes(word, n, fill = sentiment), show.legend = F) +
    labs(title = "Sentiment Analysis of Jane Austen's Works",
        subtitle = "Separated by Sentiment",
        x = "",
        y = "Contribution to Sentiment") +
    theme_classic() +
    theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5)) +
    scale_fill_brewer(palette = "Set1") +
    facet_wrap(~sentiment, scales = "free") +
    coord_flip()
```

```
Joining with `by = join_by(word)`
Selecting by n
```



2.10 Deliverable 19: Data Visualization with WordClouds

```
tidy_books %>%
    anti_join(stop_words) %>%
    count(word) %>%
    with(wordcloud(word, n, max.words = 100))
```

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

```
Warning in wordcloud(word, n, max.words = 100): pleasure could not be fit on
page. It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): attention could not be fit on
page. It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): edmund could not be fit on
page. It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): world could not be fit on page.
It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): family could not be fit on
page. It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): passed could not be fit on
page. It will not be plotted.
```

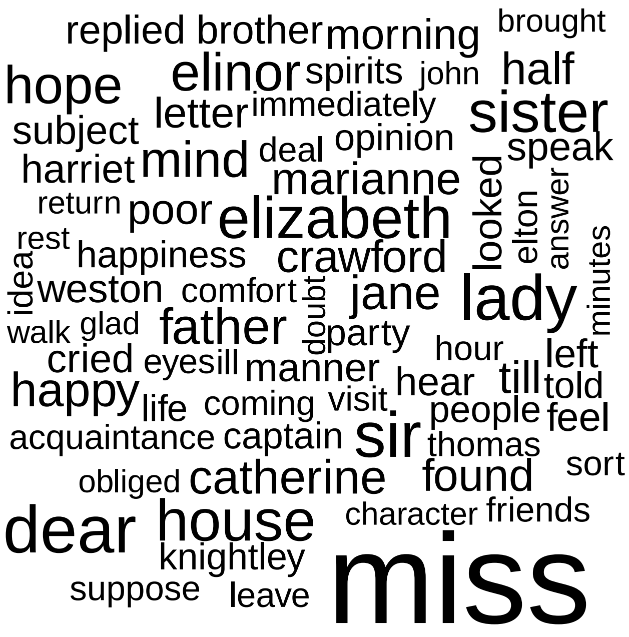
```
Warning in wordcloud(word, n, max.words = 100): woodhouse could not be fit on
page. It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): moment could not be fit on
page. It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): friend could not be fit on
page. It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): emma could not be fit on page.
It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): time could not be fit on page.
It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): evening could not be fit on
page. It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): darcy could not be fit on page.
It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): perfectly could not be fit on
page. It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): mother could not be fit on
page. It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): chapter could not be fit on
page. It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): affection could not be fit on
page. It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): anne could not be fit on page.
It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): heart could not be fit on page.
It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): woman could not be fit on page.
It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): feelings could not be fit on
page. It will not be plotted.
```



```
Warning in wordcloud(word, n, max.words = 100): word could not be fit on page.
It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): short could not be fit on page.
It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): day could not be fit on page.
It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): bennet could not be fit on
page. It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): colonel could not be fit on
page. It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): love could not be fit on page.
It will not be plotted.

Warning in wordcloud(word, n, max.words = 100): fanny could not be fit on page.
It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): home could not be fit on page.
It will not be plotted.
```

```
Warning in wordcloud(word, n, max.words = 100): heard could not be fit on page.
It will not be plotted.
```

```r
tidy_books %>%
    inner_join(get_sentiments("bing")) %>%
    count(word, sentiment, sort = TRUE) %>%
    acast(word ~ sentiment, value.var = "n", fill = 0) %>%
    comparison.cloud(colors = c("gray20","gray80"), max.words = 100)
```

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

```
Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relatio
i Row 435434 of `x` matches multiple rows in `y`.
i Row 5051 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.
```

```
Warning in comparison.cloud(., colors = c("gray20", "gray80"), max.words =
100): amiable could not be fit on page. It will not be plotted.
```

```
Warning in comparison.cloud(., colors = c("gray20", "gray80"), max.words =
100): favour could not be fit on page. It will not be plotted.
```

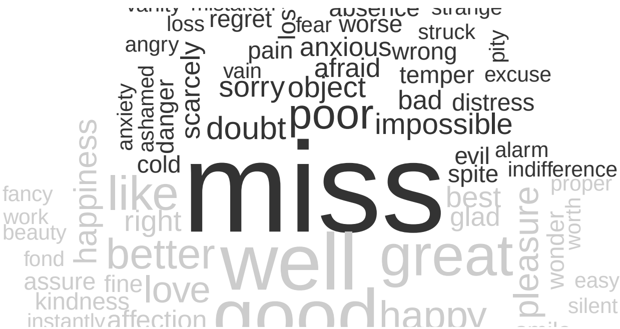
```
Warning in comparison.cloud(., colors = c("gray20", "gray80"), max.words =
100): admiration could not be fit on page. It will not be plotted.
```

```
Warning in comparison.cloud(., colors = c("gray20", "gray80"), max.words =
100): pride could not be fit on page. It will not be plotted.
```

```
Warning in comparison.cloud(., colors = c("gray20", "gray80"), max.words =
100): delight could not be fit on page. It will not be plotted.
```

```
Warning in comparison.cloud(., colors = c("gray20", "gray80"), max.words =
100): superior could not be fit on page. It will not be plotted.
```

```
Warning in comparison.cloud(., colors = c("gray20", "gray80"), max.words =
100): gratitude could not be fit on page. It will not be plotted.
```



3 Part 3: Text Mining with quanteda, Including Variable Creation and Dictionaries

```
global_path <- "/Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Ma
```

3.1 Deliverable 20: Create an Object for the UNGD Speeches

```
UNGDspeeches <- readtext(paste0(global_path, "**/*.txt"),
                docvarsfrom = "filenames",
                docvarnames = c("country","session","year"),
                dvsep = "_",
                encoding = "UTF-8")
UNGDspeeches
```

```
readtext object consisting of 2329 documents and 3 docvars.
# A data frame: 2,329 x 5
  doc_id          text                 country session  year
  <chr>           <chr>                <chr>     <int> <int>
1 AFG_26_1971.txt "\"82.\tMr. Pr\"..." AFG          26  1971
2 ALB_26_1971.txt "\"110.\t  Thi\"..." ALB          26  1971
3 ARG_26_1971.txt "\"33.\t On be\"..." ARG          26  1971
4 AUS_26_1971.txt "\"38.\t  I sh\"..." AUS          26  1971
5 AUT_26_1971.txt "\"112.\t  Mr.\"..." AUT          26  1971
6 BDI_26_1971.txt "\"1.\tMr. Pre\"..." BDI          26  1971
# i 2,323 more rows
```

```r
class(UNGDspeeches)
```

```
[1] "readtext"   "data.frame"
```

3.2 Deliverable 21: Generate a Corpus from UNGDspeeches

```r
mycorpus <- corpus(UNGDspeeches)

docvars(mycorpus, "Textno") <- sprintf("%02d", 1:ndoc(mycorpus))
mycorpus
```

```
Corpus consisting of 2,329 documents and 4 docvars.
AFG_26_1971.txt :
"82. Mr. President, at the outset, I wish to congratulate you..."

ALB_26_1971.txt :
"110.   This session of the General Assembly is meeting at a ..."

ARG_26_1971.txt :
"33.  On behalf of the Argentine Government I wish to congrat..."

AUS_26_1971.txt :
"38.   I should like, on behalf of Australia,, to extend my c..."

AUT_26_1971.txt :
"112.   Mr. President. I am happy to convey to you our sincer..."
```

```
BDI_26_1971.txt :
"1. Mr. President, this great Assembly made a very wise choic..."

[ reached max_ndoc ... 2,323 more documents ]
```

```
mycorpus.stats <- summary(mycorpus)
head(mycorpus.stats, n=10)
```

```
              Text Types Tokens Sentences country session year Textno
1  AFG_26_1971.txt  1180   4475       181     AFG      26 1971     01
2  ALB_26_1971.txt  1804   8687       263     ALB      26 1971     02
3  ARG_26_1971.txt  1495   5344       227     ARG      26 1971     03
4  AUS_26_1971.txt  1086   3857       180     AUS      26 1971     04
5  AUT_26_1971.txt  1104   3616       154     AUT      26 1971     05
6  BDI_26_1971.txt  1825   6420       266     BDI      26 1971     06
7  BEL_26_1971.txt  1312   4543       190     BEL      26 1971     07
8  BEN_26_1971.txt   781   2184        81     BEN      26 1971     08
9  BFA_26_1971.txt  1319   5035       195     BFA      26 1971     09
10 BGR_26_1971.txt  1158   4505       182     BGR      26 1971     10
```

3.3 Deliverable 22: Preprocess the Text

```
token <-tokens(mycorpus,
               split_hyphens = TRUE,
               remove_numbers = TRUE,
               remove_punct = TRUE,
               remove_symbols = TRUE,
               remove_url = TRUE,
               include_docvars = TRUE
)

token_ungd <- tokens_select(token, c("[\\d-]", "[[:punct:]]", "^.{1,2}$"),
                        selection = "remove",
                        valuetype = "regex",
                        verbose = TRUE
)
```

```
tokens_remove() changed from 6,943,345 tokens (2,329 documents) to 5,481,398 tokens (2,329 d
```

3.4 Deliverable 23: Tokenize the Dataset by N-Grams

```
toks_ngram <- tokens_ngrams(token, n = 2:4)
head(toks_ngram[[1]], 30)
```

```
 [1] "Mr_President"          "President_at"          "at_the"
 [4] "the_outset"            "outset_I"              "I_wish"
 [7] "wish_to"               "to_congratulate"       "congratulate_you"
[10] "you_whole"             "whole_heartedly"       "heartedly_on"
[13] "on_your"               "your_election"         "election_as"
[16] "as_President"          "President_of"          "of_the"
[19] "the_General"           "General_Assembly"      "Assembly_the"
[22] "the_most"              "most_esteemed"         "esteemed_and"
[25] "and_highest"           "highest_international" "international_post"
[28] "post_Our"              "Our_congratulations"   "congratulations_do"
```

```
tail(toks_ngram[[1]], 30)
```

```
 [1] "inside_and_outside_the"      "and_outside_the_United"
 [3] "outside_the_United_Nations"  "the_United_Nations_Only"
 [5] "United_Nations_Only_then"    "Nations_Only_then_will"
 [7] "Only_then_will_mankind"      "then_will_mankind_be"
 [9] "will_mankind_be_confident"   "mankind_be_confident_enough"
[11] "be_confident_enough_to"      "confident_enough_to_look"
[13] "enough_to_look_forward"      "to_look_forward_hopefully"
[15] "look_forward_hopefully_to"   "forward_hopefully_to_seeing"
[17] "hopefully_to_seeing_a"       "to_seeing_a_world"
[19] "seeing_a_world_united"       "a_world_united_in"
[21] "world_united_in_order"       "united_in_order_to"
[23] "in_order_to_achieve"         "order_to_achieve_its"
[25] "to_achieve_its_common"       "achieve_its_common_goals"
[27] "its_common_goals_of"         "common_goals_of_peace"
[29] "goals_of_peace_and"          "of_peace_and_prosperity"
```

3.5 Deliverable 24: Create a Document Feature Matrix

```
mydfm <- dfm(token_ungd, tolower = TRUE,)
mydfm <- dfm_remove(mydfm, pattern = stopwords("english"))
mydfm <- dfm_wordstem(mydfm)
```

3.6 Deliverable 25: Trim the DFM

```r
mydfm.trim <- dfm_trim(mydfm, min_docfreq = 0.075,
                    max_docfreq = 0.90,
                    docfreq_type = "prop"
)

head(dfm_sort(mydfm.trim, decreasing = TRUE, margin = "both"), n = 10, nf = 10)
```

```
Warning: nf argument is not used.
```

```
Document-feature matrix of: 10 documents, 1,959 features (51.71% sparse) and 4 docvars.
                 features
docs               problem region conflict africa global council hope situat
  CUB_34_1979.txt      36     16        1     13      3       0    8     23
  BFA_29_1974.txt      25     20        1     15      0       4   20      9
  PRY_38_1983.txt      30      7       12      0      3       3   10     16
  LBY_64_2009.txt       5      1        3      8      2      76    3      5
  LUX_35_1980.txt      21     19       13     11     10      12    9     15
  DEU_38_1983.txt      11     12       11      7     12       3   10      6
                 features
docs               resolut relat
  CUB_34_1979.txt      10    18
  BFA_29_1974.txt      10     8
  PRY_38_1983.txt      21     5
  LBY_64_2009.txt      13     1
  LUX_35_1980.txt      13    17
  DEU_38_1983.txt       4    23
[ reached max_ndoc ... 4 more documents, reached max_nfeat ... 1,949 more features ]
```

3.6.0.1 Which country refers most to the economy in this snapshot of the data?

Cuba.

3.7 Deliverable 26: Text Classification Using a Dictionary

```r
dict <- dictionary(file = "/Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Docu
```

3.8 Deliverable 27: Apply Dictionary

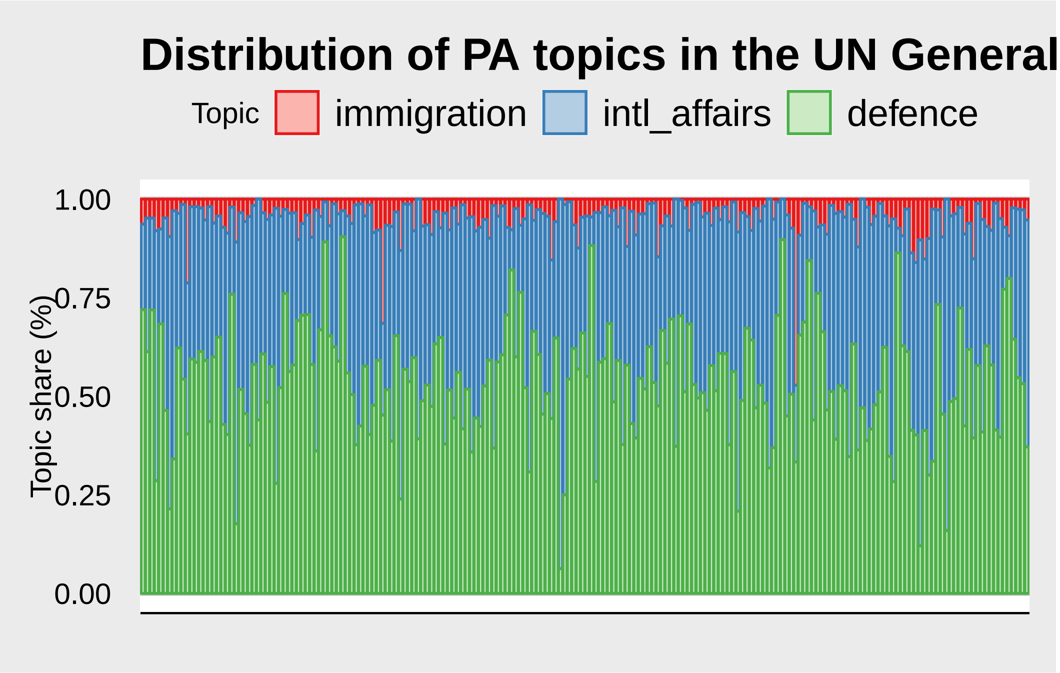
```
mydfm.un <- dfm(mydfm.trim) # create DFM w/o grouping or applying dictionary
mydfm.un <- dfm_lookup(mydfm.un, dictionary = dict) # apply dictionary
mydfm.un <- dfm_group(mydfm.un, groups = docvars(mydfm.un, "country")) # group the DFM by "c
```

3.9 Deliverable 28: Convert the DFM to a Data Frame

```
un.topics.pa <- convert(mydfm.un, "data.frame") %>%
    dplyr::rename(country = doc_id) %>%
    select(country, immigration, intl_affairs, defence) %>%
    tidyr::gather(immigration:defence, key = "Topic", value = "Share") %>%
    group_by(country) %>%
    mutate(Share = Share/ sum(Share)) %>%
    mutate(Topic = haven::as_factor(Topic))
```

3.10 Deliverable 29: Visualize the Results

```
un.topics.pa %>%
    ggplot(aes(country, Share, colour = Topic, fill = Topic))+
    geom_bar(stat = "identity")+
    ggthemes::theme_economist_white() +
    scale_color_brewer(palette = "Set1")+
    scale_fill_brewer(palette = "Pastel1")+
    ggtitle("Distribution of PA topics in the UN General Debate corpus")+
    xlab("")+
    ylab("Topic share (%)")+
    theme(axis.text.x = element_blank(),
    axis.ticks.x = element_blank())
```



4 Part 4: Using nltk and TextBlob to conduct sentiment analysis in Python

4.1 Deliverable 30: Creating a Custom Lexicon and Applying it to a Sample Dataset

```
custom_lexicon = {
'positive': ['good', 'great', 'awesome', 'fantastic', 'terrific'],
'negative': ['bad', 'terrible', 'awful', 'dreadful', 'horrible'],
'neutral': ['okay', 'alright', 'fine', 'decent', 'satisfactory'],
'uncertain': ['maybe', 'perhaps', 'possibly', 'probably', 'likely'],
'conjunctions': ['and', 'but', 'or', 'so', 'yet']
}
```

```
import nltk
nltk.download('punkt')
```

```
True
```

```
nltk.download('punkt_tab')
```

```
True
```

```
def preprocess_and_token(text):
    text = text.lower()
    tokens = text.split()
    return tokens
```

```
def preprocess_and_tokenize(text):
    text = text.lower()
    tokens = text.split()
    return tokens
```

```
def categorize_text(text, lexicon):
    tokens = preprocess_and_tokenize(text)
    categories = {category: 0 for category in lexicon}
    for token in tokens:
        for category, words in lexicon.items():
            if token in words:
                categories[category] += 1
    return categories
```

```
sample_texts = [
    'The movie was good and the acting was great.',
    'The movie was terrible and the acting was dreadful.',
    'The movie was okay and the acting was satisfactory.',
    'The movie was perhaps good and the acting was probably great.',
    'The movie was fine and the acting was decent.',
    'The movie was good but the acting was terrible.',
    'The movie was good or the acting was bad.',
    'The movie was good so the acting was bad.',
    'The movie was good yet the acting was bad.'
]
for text in sample_texts:
    categorize = categorize_text(text, custom_lexicon)
    print(categorize_text(text, custom_lexicon))
```

```
{'positive': 1, 'negative': 0, 'neutral': 0, 'uncertain': 0, 'conjunctions': 1}
{'positive': 0, 'negative': 1, 'neutral': 0, 'uncertain': 0, 'conjunctions': 1}
```

```
{'positive': 0, 'negative': 0, 'neutral': 1, 'uncertain': 0, 'conjunctions': 1}
{'positive': 1, 'negative': 0, 'neutral': 0, 'uncertain': 2, 'conjunctions': 1}
{'positive': 0, 'negative': 0, 'neutral': 1, 'uncertain': 0, 'conjunctions': 1}
{'positive': 1, 'negative': 0, 'neutral': 0, 'uncertain': 0, 'conjunctions': 1}
{'positive': 1, 'negative': 0, 'neutral': 0, 'uncertain': 0, 'conjunctions': 1}
{'positive': 1, 'negative': 0, 'neutral': 0, 'uncertain': 0, 'conjunctions': 1}
{'positive': 1, 'negative': 0, 'neutral': 0, 'uncertain': 0, 'conjunctions': 1}
```

4.2 Deliverable 31: Adding N-Grams to the Custom Lexicon

```python
custom_lexicon = {
    'positive': ['good', 'great', 'awesome', 'fantastic', 'terrific', 'good and', 'great and
    'negative': ['bad', 'terrible', 'awful', 'dreadful', 'horrible', 'bad and', 'terrible and
    'neutral': ['okay', 'alright', 'fine', 'decent', 'satisfactory', 'okay and', 'alright and
    'uncertain': ['maybe', 'perhaps', 'possibly', 'probably', 'likely', 'maybe and', 'perhaps
    'conjunctions': ['and', 'but', 'or', 'so', 'yet', 'but and', 'or and', 'so and', 'yet and
}
```

4.3 Deliverable 32: Applying the Custom Lexicon with N-Grams to the Sample Sentences

```python
from nltk.util import ngrams
```

```python
def generate_ngrams(tokens, n):
   return [' '.join(gram) for gram in ngrams(tokens, n)]
```

```python
def preprocess_and_tokenize(text):
    text = text.lower()
    tokens = text.split()
    bigrams = generate_ngrams(tokens, 2)
    trigrams = generate_ngrams(tokens, 3)
    all_tokens = tokens + bigrams + trigrams
    return all_tokens
```

```python
def categorize_text(text, lexicon):
    tokens = preprocess_and_tokenize(text)
    categories = {category: 0 for category in lexicon}
    for token in tokens:
        for category, phrases in lexicon.items():
            if token in phrases:
                categories[category] += 1
    return categories
```

4.4 Deliverable 33: Downloading NLTK Data and Preparing the Dataset

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon') # Download VADER lexicon
```

```
True
```

```
# Initialize VADER sentiment analyzer
sia = SentimentIntensityAnalyzer()
# Sample text
text = "I love this product! It's absolutely amazing :)"
# Get sentiment scores
sentiment = sia.polarity_scores(text)
print(sentiment)
```

```
{'neg': 0.0, 'neu': 0.252, 'pos': 0.748, 'compound': 0.9163}
```

```
import nltk
from nltk.corpus import movie_reviews
from nltk.sentiment import SentimentIntensityAnalyzer
import pandas as pd
nltk.download('movie_reviews')
```

```
True
```

```
nltk.download('vader_lexicon')
```

```
True
```

```
documents = [(list(movie_reviews.words(fileid)), category)
            for category in movie_reviews.categories()
            for fileid in movie_reviews.fileids(category)]
```

```
reviews = pd.DataFrame(documents, columns = ['text', 'sentiment'])
reviews['text'] = reviews['text'].apply(lambda x: ' '.join(x))
```

4.5 Deliverable 34: Display the first Five Rows of the Reviews Dataframe

```
print(reviews.head())
```

```
                                                text sentiment
0  plot : two teen couples go to a church party ,...       neg
1  the happy bastard ' s quick movie review damn ...       neg
2  it is movies like these that make a jaded movi...       neg
3  " quest for camelot " is warner bros . ' first...       neg
4  synopsis : a mentally unstable man undergoing ...       neg
```

```
print(reviews.tail())
```

```
                                                   text sentiment
1995  wow ! what a movie . it ' s everything a movie...       pos
1996  richard gere can be a commanding actor , but h...       pos
1997  glory -- starring matthew broderick , denzel w...       pos
1998  steven spielberg ' s second epic film on world...       pos
1999  truman ( " true - man " ) burbank is the perfe...       pos
```

4.6 Deliverable 35: Sentiment Analysis with VADER

```
sid = SentimentIntensityAnalyzer()
reviews['scores'] = reviews['text'].apply(lambda review: sid.polarity_scores(review))
reviews['compound'] = reviews['scores'].apply(lambda score_dict: score_dict['compound'])
reviews['comp_score'] = reviews['compound'].apply(lambda c: 'pos' if c >=0 else 'neg')

print(reviews[['text', 'sentiment', 'compound', 'comp_score']].head())
```

```
                                                text  ... comp_score
0  plot : two teen couples go to a church party ,...  ...        pos
1  the happy bastard ' s quick movie review damn ...  ...        pos
2  it is movies like these that make a jaded movi...  ...        pos
3  " quest for camelot " is warner bros . ' first...  ...        neg
4  synopsis : a mentally unstable man undergoing ...  ...        pos

[5 rows x 4 columns]
```

4.7 Deliverable 36: Quick Exploration of Sentiment Analysis in TextBlob

```python
import nltk
from textblob import TextBlob
nltk.download('gutenberg')
```

```
True
```

```python
from nltk.corpus import gutenberg
```

```python
text = gutenberg.raw('austen-emma.txt') # import text

sentences = nltk.sent_tokenize(text) # split into sentences

for sentence in sentences[:25]:
    blob = TextBlob(sentence)
    print(f"Sentence: {sentence}\nPolarity: {blob.sentiment.polarity}\n")
```

```
Sentence: [Emma by Jane Austen 1816]

VOLUME I

CHAPTER I


Emma Woodhouse, handsome, clever, and rich, with a comfortable home
and happy disposition, seemed to unite some of the best blessings
of existence; and had lived nearly twenty-one years in the world
with very little to distress or vex her.
Polarity: 0.3872395833333333
```

```
Sentence: She was the youngest of the two daughters of a most affectionate,
indulgent father; and had, in consequence of her sister's marriage,
been mistress of his house from a very early period.
Polarity: 0.315

Sentence: Her mother
had died too long ago for her to have more than an indistinct
remembrance of her caresses; and her place had been supplied
by an excellent woman as governess, who had fallen little short
of a mother in affection.
Polarity: 0.2525

Sentence: Sixteen years had Miss Taylor been in Mr. Woodhouse's family,
less as a governess than a friend, very fond of both daughters,
but particularly of Emma.
Polarity: 0.06666666666666667

Sentence: Between _them_ it was more the intimacy
of sisters.
Polarity: 0.5

Sentence: Even before Miss Taylor had ceased to hold the nominal
office of governess, the mildness of her temper had hardly allowed
her to impose any restraint; and the shadow of authority being
now long passed away, they had been living together as friend and
friend very mutually attached, and Emma doing just what she liked;
highly esteeming Miss Taylor's judgment, but directed chiefly by
her own.
Polarity: 0.20305555555555554

Sentence: The real evils, indeed, of Emma's situation were the power of having
rather too much her own way, and a disposition to think a little
too well of herself; these were the disadvantages which threatened
alloy to her many enjoyments.
Polarity: 0.2625

Sentence: The danger, however, was at present
so unperceived, that they did not by any means rank as misfortunes
with her.
Polarity: -0.4

Sentence: Sorrow came--a gentle sorrow--but not at all in the shape of any
```

```
disagreeable consciousness.--Miss Taylor married.
Polarity: 0.225

Sentence: It was Miss
Taylor's loss which first brought grief.
Polarity: -0.275

Sentence: It was on the wedding-day
of this beloved friend that Emma first sat in mournful thought
of any continuance.
Polarity: 0.475

Sentence: The wedding over, and the bride-people gone,
her father and herself were left to dine together, with no prospect
of a third to cheer a long evening.
Polarity: -0.016666666666666666

Sentence: Her father composed himself
to sleep after dinner, as usual, and she had then only to sit
and think of what she had lost.
Polarity: -0.125

Sentence: The event had every promise of happiness for her friend.
Polarity: 0.7

Sentence: Mr. Weston
was a man of unexceptionable character, easy fortune, suitable age,
and pleasant manners; and there was some satisfaction in considering
with what self-denying, generous friendship she had always wished
and promoted the match; but it was a black morning's work for her.
Polarity: 0.3875

Sentence: The want of Miss Taylor would be felt every hour of every day.
Polarity: 0.0

Sentence: She recalled her past kindness--the kindness, the affection of sixteen
years--how she had taught and how she had played with her from five
years old--how she had devoted all her powers to attach and amuse
her in health--and how nursed her through the various illnesses
of childhood.
Polarity: -0.125

Sentence: A large debt of gratitude was owing here; but the
```

```
intercourse of the last seven years, the equal footing and perfect
unreserve which had soon followed Isabella's marriage, on their
being left to each other, was yet a dearer, tenderer recollection.
Polarity: 0.18154761904761904

Sentence: She had been a friend and companion such as few possessed: intelligent,
well-informed, useful, gentle, knowing all the ways of the family,
interested in all its concerns, and peculiarly interested in herself,
in every pleasure, every scheme of hers--one to whom she could speak
every thought as it arose, and who had such an affection for her
as could never find fault.
Polarity: 0.2

Sentence: How was she to bear the change?--It was true that her friend was
going only half a mile from them; but Emma was aware that great must
be the difference between a Mrs. Weston, only half a mile from them,
and a Miss Taylor in the house; and with all her advantages,
natural and domestic, she was now in great danger of suffering
from intellectual solitude.
Polarity: 0.20606060606060606

Sentence: She dearly loved her father, but he
was no companion for her.
Polarity: 0.7

Sentence: He could not meet her in conversation,
rational or playful.
Polarity: 0.0

Sentence: The evil of the actual disparity in their ages (and Mr. Woodhouse had
not married early) was much increased by his constitution and habits;
for having been a valetudinarian all his life, without activity
of mind or body, he was a much older man in ways than in years;
and though everywhere beloved for the friendliness of his heart
and his amiable temper, his talents could not have recommended him
at any time.
Polarity: 0.005952380952380947

Sentence: Her sister, though comparatively but little removed by matrimony,
being settled in London, only sixteen miles off, was much beyond
her daily reach; and many a long October and November evening must
be struggled through at Hartfield, before Christmas brought the next
visit from Isabella and her husband, and their little children,
to fill the house, and give her pleasant society again.
Polarity: 0.11203703703703703

Sentence: Highbury, the large and populous village, almost amounting to a town,
to which Hartfield, in spite of its separate lawn, and shrubberies,
and name, did really belong, afforded her no equals.
Polarity: 0.20714285714285713
```

4.8 Deliverable 37: Sentiment Analysis on the UN Data with TextBlob

```python
import os
from textblob import TextBlob
```

```python
folder_path = "/Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Mai
results = [] # List to store the results
# Walk through all subdirectories and files
for root, dirs, files in os.walk(folder_path):
    for filename in files:
        if filename.endswith('.txt'):
            file_path = os.path.join(root, filename)
            with open(file_path, 'r', encoding='utf-8') as file:
                text = file.read()
                blob = TextBlob(text)
                polarity = blob.sentiment.polarity
                subjectivity = blob.sentiment.subjectivity
                results.append({
                    'file_path': file_path,
                    'polarity': polarity,
                    'subjectivity': subjectivity
                    })

for result in results[:5]:
    print(f"File: {result['file_path']}")
    print(f"Polarity: {result['polarity']}")
    print(f"Subjectivity: {result['subjectivity']}")
    print('---')
```

```
File: /Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Main/023. Pro
Polarity: 0.056829403519977284
Subjectivity: 0.42787665886026544
```

```
---
File: /Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Main/023. Pro
Polarity: 0.12719839684125395
Subjectivity: 0.48171411921411905
---
File: /Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Main/023. Pro
Polarity: 0.10939538239538237
Subjectivity: 0.39115632515632504
---
File: /Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Main/023. Pro
Polarity: 0.07273518148518145
Subjectivity: 0.4263486513486513
---
File: /Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Main/023. Pro
Polarity: 0.09259146767211283
Subjectivity: 0.39663748079877126
---
```