HW1 Answer Key

Christian Baehr

10/18/2023

Load in required packages.

```
rm(list=ls())
pacman::p_load(quanteda, quanteda.corpora, quanteda.textstats, dplyr, ggplot2,
               readtext, stringr, sotu, gutenbergr, stylest, text.alignment)
```

Question 1

```
## merge the meta and text dataframes from the sotu package
sotu <- cbind(sotu_meta, sotu_text)

## subset to just speeches between 2007 and 2010
sotu <- sotu[which(sotu$year %in% 2007:2010), ]

## convert to corpus, with "sotu_text" as the variable with text data
sotu.corpus <- corpus(sotu, text_field = "sotu_text")

## tokenize the speeches (no pre processing yet)
sotu.tokens <- tokens(sotu.corpus)
```

1a)

```
## function to compute TTR
##
## @param x tokenized quanteda corpus
calculate_TTR <- function(x){
  ntype(x)/lengths(x)
}
calculate_TTR(sotu.tokens)
```

```
##     text1     text2     text3     text4
## 0.2741438 0.2669819 0.2390753 0.2289587
```

```
## function to compute Guiraud's index of lexical richness
##
## @param x tokenized quanteda corpus
```

```r
calculate_G <- function(x){
  ntype(x)/sqrt(lengths(x))
}
calculate_G(sotu.tokens)
```

```
##    text1    text2    text3    text4
## 21.51811 21.26658 19.57646 20.62536
```

1b)

```r
## create dfm of the SOTU speeches
sotu.dfm <- tokens(sotu.corpus, remove_punct = T) |>
  dfm(tolower=F)
textstat_simil(sotu.dfm, margin = "documents", method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##       text1 text2 text3 text4
## text1 1.000 0.974 0.934 0.937
## text2 0.974 1.000 0.944 0.940
## text3 0.934 0.944 1.000 0.968
## text4 0.937 0.940 0.968 1.000
```

Question 2

2a) Calculating TTR & Similarity w/ Stemming

```r
## Processing
sotu.tokens <- tokens(sotu.corpus, remove_punct = TRUE) |>
  tokens_wordstem()

## TTR
ttr <- calculate_TTR(sotu.tokens) %>% setNames(c("Bush-07", "Bush-08", "Obama-09", "Obama-10"))
r <- calculate_G(sotu.tokens) %>% setNames(c("Bush-07", "Bush-08", "Obama-09", "Obama-10"))

## Similarity
rr_dfm <- dfm(sotu.tokens, tolower = FALSE)
sim <- textstat_simil(sotu.dfm, margin = "documents", method = "cosine")

## print results
cat("TTR scores w. stemming: \n", ttr, "\n\n")
```

```
## TTR scores w. stemming:
##  0.2536375 0.2420795 0.2204061 0.209537
```

```r
cat("G scores w. stemming: \n", r, "\n\n")
```

```
## G scores w. stemming:
##  18.92449 18.29743 17.15349 17.82296
```

```r
cat("Cosine similarity w. stemming: \n"); prmatrix(as.matrix(sim))
```

```
## Cosine similarity w. stemming:
```

```
##           text1     text2     text3     text4
## text1 1.0000000 0.9743182 0.9341894 0.9367975
## text2 0.9743182 1.0000000 0.9440385 0.9399558
## text3 0.9341894 0.9440385 1.0000000 0.9676051
## text4 0.9367975 0.9399558 0.9676051 1.0000000
```

2b) Calculating TTR & Similarity w/o Stopwords

```r
## Processing
sotu.tokens <- tokens(sotu.corpus, remove_punct = TRUE) |>
  tokens_remove(stopwords("english"))

## TTR
ttr <- calculate_TTR(sotu.tokens)
r <- calculate_G(sotu.tokens) %>% setNames(c("Bush-07", "Bush-08", "Obama-09", "Obama-10"))

## Similarity
sotu.dfm <- dfm(sotu.tokens, tolower = FALSE)
sim <- textstat_simil(sotu.dfm, margin = "documents", method = "cosine")

## print results
cat("TTR scores w/o stopwords: \n", ttr, "\n\n")
```

```
## TTR scores w/o stopwords:
##  0.5171013 0.4963034 0.4714665 0.4473472
```

```r
cat("G scores w/o stopwords: \n", r, "\n\n")
```

```
## G scores w/o stopwords:
##  28.10006 27.68199 25.88347 27.25883
```

```r
cat("Cosine similarity w/o stopwords: \n"); prmatrix(as.matrix(sim))
```

```
## Cosine similarity w/o stopwords:
```

```
##           text1     text2     text3     text4
## text1 1.0000000 0.7371163 0.6163764 0.6134526
## text2 0.7371163 1.0000000 0.6056913 0.6066282
## text3 0.6163764 0.6056913 1.0000000 0.7485176
## text4 0.6134526 0.6066282 0.7485176 1.0000000
```

2c) Calculating TTR & Similarity w/ all lowercase

```r
## Processing
sotu.tokens <- tokens(sotu.corpus, remove_punct = TRUE) |>
  tokens_tolower()

## TTR
ttr <- calculate_TTR(sotu.tokens)
r <- calculate_G(sotu.tokens)

## Similarity
sotu.dfm <- dfm(sotu.tokens)
sim <- textstat_simil(sotu.dfm, margin = "documents", method = "cosine")

# print results
cat("TTR scores w. lowercase: \n", ttr, "\n\n")
```

```
## TTR scores w. lowercase:
##  0.2857913 0.2804131 0.2519399 0.2403594
```

```r
cat("G scores w. lowercase: \n", r, "\n\n")
```

```
## G scores w. lowercase:
##  21.32355 21.19485 19.60766 20.44468
```

```r
cat("Cosine similarity w. lowercases: \n"); prmatrix(as.matrix(sim))
```

```
## Cosine similarity w. lowercases:

##           text1     text2     text3     text4
## text1 1.0000000 0.9785101 0.9417215 0.9455727
## text2 0.9785101 1.0000000 0.9493788 0.9483864
## text3 0.9417215 0.9493788 1.0000000 0.9716291
## text4 0.9455727 0.9483864 0.9716291 1.0000000
```

2d) TF-IDF

These are the results below, but we might hesitate to use TF-IDF when we have very few documents in the corpus.

```r
sotu.dfm.tfidf <- tokens(sotu.corpus, remove_punct = T) %>%
  dfm() %>%
  dfm_tfidf()
textstat_simil(sotu.dfm.tfidf, margin = "documents", method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##        text1  text2  text3  text4
## text1 1.0000 0.1562 0.0555 0.0602
## text2 0.1562 1.0000 0.0560 0.0657
## text3 0.0555 0.0560 1.0000 0.2163
## text4 0.0602 0.0657 0.2163 1.0000
```

Question 3

3a)

```r
## file names
files <- c("wealth_of_nations.txt", "theory_of_moral_sentiments.txt")

## read each text as a corpus object
smith <- readtext(files) |>
  corpus()

## docvar with titles
smith$title <- c("wealth of nations", "theory of moral sentiments")
```

3b)

```r
## lowercase and remove hyphens
smith <- tolower(smith)
smith <- gsub("-", " ", smith)

## remove symbols/punctuation/numbers/stopwords
smith.tok <- tokens(smith, remove_symbols = T, remove_punct = T, remove_numbers = T) |>
  tokens_remove(stopwords())
```

3c)

```r
## use tfidf weighting with numerator the proportion of
## document tokens of that type
smith.dfm <- dfm(smith.tok) |>
  dfm_tfidf(scheme_tf = "prop", base = exp(1))

topfeatures(smith.dfm[2,]) # pretty close!
```

```
##    quantity        land      silver      market        rent        corn
## 0.003077651 0.002780312 0.002552481 0.002247419 0.001907603 0.001718387
##    colonies      annual commodities consumption
## 0.001262725 0.001235694 0.001189356 0.001073509
```

Question 4

```r
sentence1 <- "Biden Administration Loses Expensive Aircraft Because Pilot Scared of Bad Weather"

sentence2 <- "U.S. Marine Pilot Ejects From F-35 Aircraft Following Mishap in South Carolina"

## remove punctuation and tokenize
sentences.tokens <- corpus(c(sentence1, sentence2)) |>
```

```r
  tokens(remove_punct = T)

sentences.dfm <- dfm(sentences.tokens, tolower = T)

s1 <- as.matrix(sentences.dfm)[1,] # feature vector for sentence1
s2 <- as.matrix(sentences.dfm)[2,] # feature vector for sentence2

## Euclidean distance
euclidean <- sqrt( sum( ( s1-s2 )^2 ) )

## Manhattan distance
manhattan <- sum( abs( s1-s2 ) )

## Jaccard distance
num <- length( intersect(sentences.tokens[[1]], sentences.tokens[[2]]) )
denom <- length( union(sentences.tokens[[1]],sentences.tokens[[2]]) )
jaccard <- num / denom

## Cosine similarity
cosine <- sum(s1 * s2) /( sqrt(sum(s1^2)) * sqrt(sum(s2^2)) )

## Levenshtein distance for surveyance and surveillance
levenshtein <- adist("surveyance", "surveillance")

## print
cat("Euclidean distance:", euclidean, "\n\n",
    "Manhattan distance:", manhattan, "\n\n",
    "Jaccard similarity:", jaccard, "\n\n",
    "Cosine similarity:", cosine, "\n\n",
    "Levenshtein distance:", levenshtein)
```

```
## Euclidean distance: 4.358899
##
##  Manhattan distance: 19
##
##  Jaccard similarity: 0.0952381
##
##  Cosine similarity: 0.1740777
##
##  Levenshtein distance: 3
```

Question 5

5a-b) Prepare Data

```r
## list of authors
author_list <- c("Fitzgerald, F. Scott (Francis Scott)", "Melville, Herman",
                 "Austen, Jane", "Dickens, Charles")

## Prepare data function
##
```

```r
## @param author_name: author's name as it would appear in gutenbergr
## @param num_texts: numeric specifying number of texts to select
## @param num_lines: num_lines specifying number of sentences to sample
prepare.dt <- function(author_name, num_texts, num_lines, removePunct = TRUE){
  meta <- gutenberg_works(author == author_name) %>% slice(1:num_texts)
  meta <- meta %>% mutate(author = unlist(str_split(author, ","))[1] %>% tolower(.))
  texts <- lapply(meta$gutenberg_id, function(x) gutenberg_download(x) %>%
                  select(text) %>%
                  sample_n(num_lines) %>%
                  unlist() %>%
                  paste(., collapse = " ") %>%
                  str_squish(.) %>%
                  str_trim(., side = "both")) # remove white space
  texts <- lapply(texts, function(x) gsub("`|'", "", x)) # remove apostrophes
  if(removePunct) texts <- lapply(texts, function(x) gsub("[^[:alpha:]]", " ", x))
  # remove all non-alpha characters
  output <- tibble(title = meta$title, author = meta$author,
                   text = unlist(texts, recursive = FALSE))
}

## run function
set.seed(0123)
texts.dt <- lapply(author_list, prepare.dt, num_texts = 4, num_lines = 500, removePunct = F)
texts.dt <- do.call(rbind, texts.dt)
```

5c) Select Features

```r
filter <- corpus::text_filter(drop_punct = T, drop_number = T)
set.seed(0123)  # remember to set seed for replicability
vocab <- stylest_select_vocab(texts.dt$text, texts.dt$author,
                              nfold = 5,
                              smooth = 1,
                              filter = filter,
                              cutoff_pcts = c(25, 50, 60, 70, 75, 80, 90, 99))

## percentile with best prediction rate
vocab$cutoff_pct_best
```

```
## [1] 99
```

```r
## rate of INCORRECTLY predicted speakers of held-out texts
vocab$miss_pct
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  100  100  100  100  100   50    0    0
## [2,]   75   75   75   75   75   75   75   50
## [3,]   50   50   50   50   50    0    0    0
## [4,]   50   50   50   25   25   25    0    0
## [5,]    0    0    0    0    0    0    0    0
```

```r
## average misspecification across hold-out sample by percentiles
misspct <- apply(vocab$miss_pct, 2, mean)
names(misspct) <- c("pct25", "pct50", "pct60", "pct70", "pct75", "pct80", "pct90", "pct99")

print(misspct)
```

```
## pct25 pct50 pct60 pct70 pct75 pct80 pct90 pct99
##    55    55    55    50    50    30    15    10
```

5d) Prune Features and Fit Model

```r
## apply threshold to prunce features
prune.vocab <- stylest_terms(texts.dt$text,
                             texts.dt$author,
                             vocab$cutoff_pct_best,
                             filter = filter)

## Fit model
style.model <- stylest_fit(texts.dt$text, texts.dt$author, terms = prune.vocab,
                           filter = filter)

## explore fit
authors <- unique(texts.dt$author)
term_usage <- style.model$rate

## all "stopwords" (high frequency words)
lapply(authors, function(x) head(term_usage[x,][order(-term_usage[x,])])) %>% setNames(authors)
```

```
## $fitzgerald
##        the        and          a         of         to         in
## 0.10005882 0.05901575 0.05627083 0.04921247 0.04019345 0.03561859
##
## $melville
##        the         of        and          a         to         in
## 0.12549000 0.06725042 0.05859594 0.04749784 0.04658148 0.03965789
##
## $austen
##        the         to        and         of          a        her
## 0.06481274 0.06005831 0.05727742 0.05494505 0.03556851 0.03027585
##
## $dickens
##        the        and         of         to          a         in
## 0.11118120 0.06944225 0.05177942 0.04662777 0.04063502 0.03390632
```

5e) Ratio of Rate Vectors

```r
## take ratios
ratio <- term_usage["austen",]/term_usage["dickens",]
head(ratio[order(-ratio)])  # more specific to each author
```

```
##      she      her    could     much      not      can
## 4.444977 4.084663 3.530074 2.425880 2.357258 2.286685
```

```r
head(ratio[order(ratio)]) # less specific to each author
```

```
##       out      down      into      came     night      back
## 0.2745209 0.2937386 0.3027628 0.3702742 0.3824872 0.4128584
```

5f) Mystery Author

```r
load("mystery_excerpt.rds")

## use fitted model to predict author
pred <- stylest_predict(style.model, mystery_excerpt)
pred$predicted
```

```
## [1] austen
## Levels: austen dickens fitzgerald melville
```

```r
pred$log_probs
```

```
## 1 x 4 Matrix of class "dgeMatrix"
##          austen  dickens fitzgerald  melville
## [1,] -0.6889113 -4.42669  -4.908984 -0.737008
```

Question 6

6a) Contingency table for UK Manifestos

```r
## get text from UK political manifestos speeches
corpus <- corpus_subset(data_corpus_ukmanifestos, Year %in% c(1945:1955))
text <- tokens(corpus, remove_punct = T) |>
  paste(collapse = " ") |>
  tolower()

## get entry of contingency table for the collocation
o11 <- str_count(text, "united(?= kingdom)")
o12 <- str_count(text, "united(?! kingdom)")
o21 <- str_count(text, "(?<!united )kingdom")
N <- tokens(text) |>
  tokens_ngrams(n = 2) |>
  ntoken() |>
  unname()
o22 <- N - o21 - o11 - o12

## contingency table
out <- matrix(c(o11, o12, o21, o22),
                 ncol = 2,
```

```r
                  byrow = T)
rownames(out) <- c("United", "Not United")
colnames(out) <- c("Kingdom", "Not Kingdom")
print(out)
```

```
##            Kingdom Not Kingdom
## United          13          44
## Not United       2       51092
```

```r
## expected frequency
E11 <- (o11+o12)/N * (o11 + o21)/N * N
# N12 <- N - (o11 + o21)
# E21 <- (o11+o21)/N * N21/N * N
# E12 <- (o11+o12)/N * N12/N * N
# E22 <- N12/N * N21/N * N

## get Chi-square value
## (o11-E11)^2/E11 + (o21-E21)^2/E21 + (o12-E12)^2/E12 + (o22-E22)^2/E22

## print
cat("Observed frequency:", o11, "\n\n",
    "Expected frequency:", E11)
```

```
## Observed frequency: 13
##
##  Expected frequency: 0.01671522
```

6b) Collocation for "United Kingdom" using quanteda

```r
textstat_collocations(corpus, min_count = 5) %>%
  data.frame() %>%
  select(c("collocation", "lambda", "z")) %>%
  filter(collocation == "united kingdom")
```

```
##        collocation   lambda        z
## 114 united kingdom 8.703808 12.33136
```

6c) Collocations using quanteda

```r
(collout1 <- textstat_collocations(corpus, min_count = 5) |>
   arrange(-lambda) |>
   slice(1:10) |>
   data.frame() |>
   select(c("collocation", "count", "lambda", "z")))
```

```
##          collocation count   lambda         z
## 482   05-apr-2001 00    11 14.57272  7.208407
## 517 northern ireland     6 14.00228  6.870247
```

```
## 349         per cent    12 13.04659  8.284559
## 395           44 GMT     8 12.32452  7.949845
## 378  manifesto index    10 12.28445  8.069520
## 417    hydrogen bomb     6 12.05630  7.718918
## 488   civil aviation     5 10.79051  7.150772
## 132    raw materials    16 10.79025 11.897479
## 487    last modified    11 10.42891  7.164762
## 130    bulk purchase     7 10.13787 11.987849
```

```
(collout2 <- textstat_collocations(corpus, min_count = 5) |>
   arrange(-count) |>
   slice(1:10) |>
   data.frame() |>
   select(c("collocation", "count", "lambda", "z")))
```

```
##      collocation count    lambda         z
## 3         of the   511 1.5791977 29.049959
## 7         in the   321 1.8411879 26.650363
## 1        will be   202 4.0177210 39.888514
## 4       we shall   201 6.4010169 28.892577
## 479       to the   186 0.5788572  7.238386
## 1150     and the   160 0.1970529  2.333199
## 2        must be   157 4.2712557 35.975141
## 57       for the   135 1.4346411 14.432671
## 31        of our   104 1.9796117 17.104630
## 10       we have    93 3.1359640 24.678345
```

Question 7

```
## Prepare data function
##
## @param book_id: book_id as it would appear in gutenbergr
## @param removePunct logical specifying whether to remove punctuation
prepare_dt <- function(book_id, removePunct = TRUE){
  meta <- gutenberg_works(gutenberg_id  == book_id)
  meta <- meta %>% mutate(author = unlist(str_split(author, ","))[1] %>% tolower(.))
  text <- gutenberg_download(book_id) %>%
                    select(text) %>%
                    filter(text!="") %>%
                    unlist() %>%
                    paste(., collapse = " ") %>%
                    str_squish(.) %>%
                    str_trim(., side = "both")
  text <- gsub("`|'", "", text) # remove apostrophes
  text <- gsub("[^[:alpha:]]", " ", text) # remove all non-alpha characters
  output <- tibble(title = meta$title, author = meta$author, text = text)
}

## run function
novels <- lapply(c(64317, 2489), prepare_dt, removePunct = TRUE) %>% do.call(rbind,.)
```

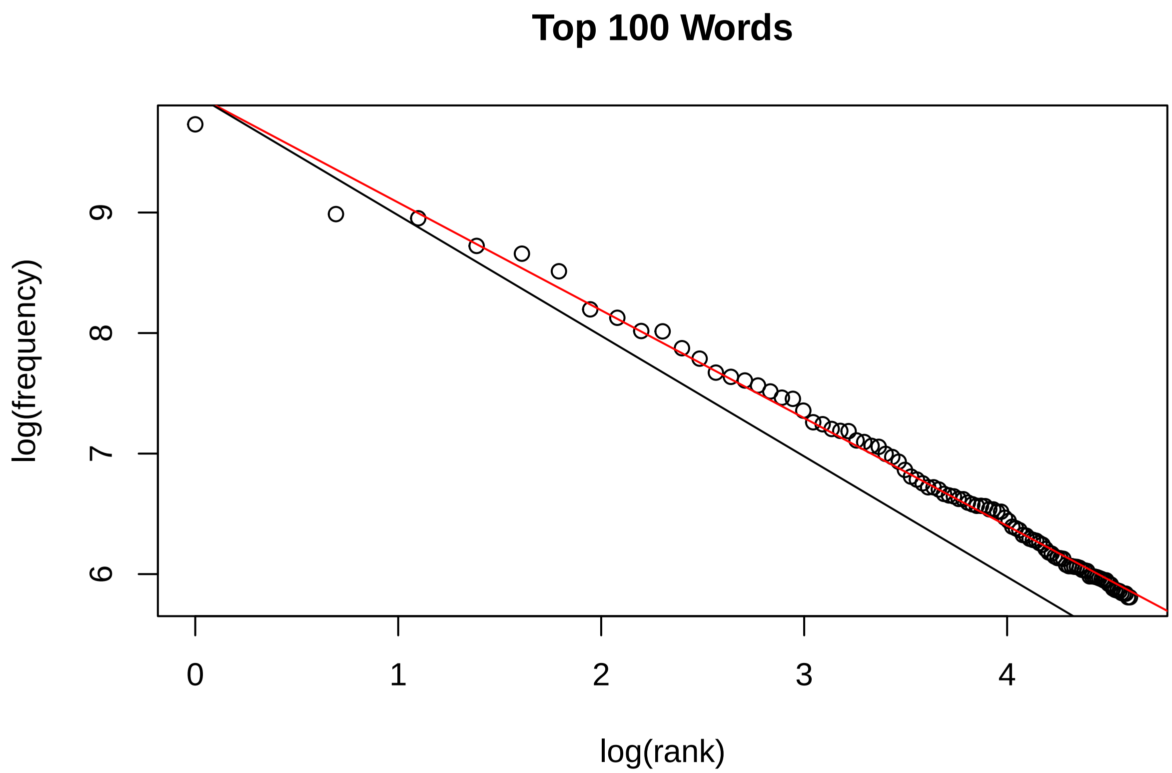
```r
## create dfm
dfm <- tokens(novels$text, remove_punct = T) |>
  dfm(tolower = T)

## regression to check if slope is approx -1.0
regression <- lm(log(topfeatures(dfm, 100)) ~ log(1:100))
summary(regression)
```

```
##
## Call:
## lm(formula = log(topfeatures(dfm, 100)) ~ log(1:100))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37032 -0.03492 -0.00347  0.04543  0.13754
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.976976   0.026751   373.0   <2e-16 ***
## log(1:100)  -0.894054   0.007128  -125.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06582 on 98 degrees of freedom
## Multiple R-squared:  0.9938, Adjusted R-squared:  0.9937
## F-statistic: 1.573e+04 on 1 and 98 DF,  p-value: < 2.2e-16
```

```r
confint(regression)
```

```
##                 2.5 %     97.5 %
## (Intercept)  9.923889 10.0300637
## log(1:100)  -0.908200 -0.8799076
```

```r
# create plot to illustrate zipf's law
plot(log(1:100), log(topfeatures(dfm, 100)),
     xlab="log(rank)", ylab="log(frequency)", main="Top 100 Words")
abline(regression, col="red")
abline(a = regression$coefficients["(Intercept)"], b = -1, col = "black")
```



Question 8

```r
## Heap's Law
## M = kT^b
## where:
## M = vocab size
## T = number of tokens
## k, b are constants

num_tokens <- sum(rowSums(dfm))
M <- nfeat(dfm)
k <- 44

## solve for b
b <- log(M/k)/log(num_tokens)
print(b)
```

```
## [1] 0.4863215
```

Question 9

```r
corpus <- data_corpus_ukmanifestos

## key words in context
corpus_subset(corpus, Party == "Lab") |>
  tokens(remove_punct = T) |>
  kwic("nation", window = 5)
corpus_subset(corpus, Party == "Lab") |>
  tokens(remove_punct = T) |>
  kwic("industry", window = 5)

corpus_subset(corpus, Party == "Con") |>
  tokens(remove_punct = T) |>
  kwic("nation", window = 5)
corpus_subset(corpus, Party == "Con") |>
  tokens(remove_punct = T) |>
  kwic("industry", window = 5)
```

Question 10

10a)

```r
sotu.sub <- data_corpus_sotu
sotu.sub$Date <- format(sotu.sub$Date, "%Y")
names(docvars(sotu.sub))[3] <- "year"

sotu.sub <- sotu.sub |>
  corpus_subset(year %in% 1982:2020) |>
  corpus_reshape("sentence")

sotu.df <- cbind(as.character(sotu.sub), docvars(sotu.sub)["year"]) |>
  setNames(c("text", "year"))

sotu.split <- split(sotu.df, as.factor(sotu.df$year))

boot.fre <- function(year) { # accepts df of texts (year-specific)
  n <- nrow(year) # number of texts
  docnums <- sample(1:n, size=n, replace=T) # sample texts WITH replacement
  docs.boot <- corpus(year[docnums, "text"])
  docnames(docs.boot) <- 1:length(docs.boot) # something you have to do
  fre <- textstat_readability(docs.boot, measure = "Flesch") # compute FRE for each
  return(mean(fre[,"Flesch"])) # return flesch scores only
}

lapply(sotu.split, boot.fre) # apply to each df of party texts
```

```
## $`1982`
## [1] 53.02008
##
## $`1983`
## [1] 50.31358
##
```

```
## $`1984`
## [1] 56.16722
##
## $`1985`
## [1] 55.52416
##
## $`1986`
## [1] 58.51162
##
## $`1987`
## [1] 55.97253
##
## $`1988`
## [1] 51.89491
##
## $`1989`
## [1] 61.36738
##
## $`1990`
## [1] 57.31667
##
## $`1991`
## [1] 54.46893
##
## $`1992`
## [1] 70.90926
##
## $`1993`
## [1] 53.40139
##
## $`1994`
## [1] 64.48437
##
## $`1995`
## [1] 66.28487
##
## $`1996`
## [1] 60.96427
##
## $`1997`
## [1] 57.47489
##
## $`1998`
## [1] 58.1499
##
## $`1999`
## [1] 59.81814
##
## $`2000`
## [1] 62.22857
##
## $`2001`
## [1] 56.80397
##
```

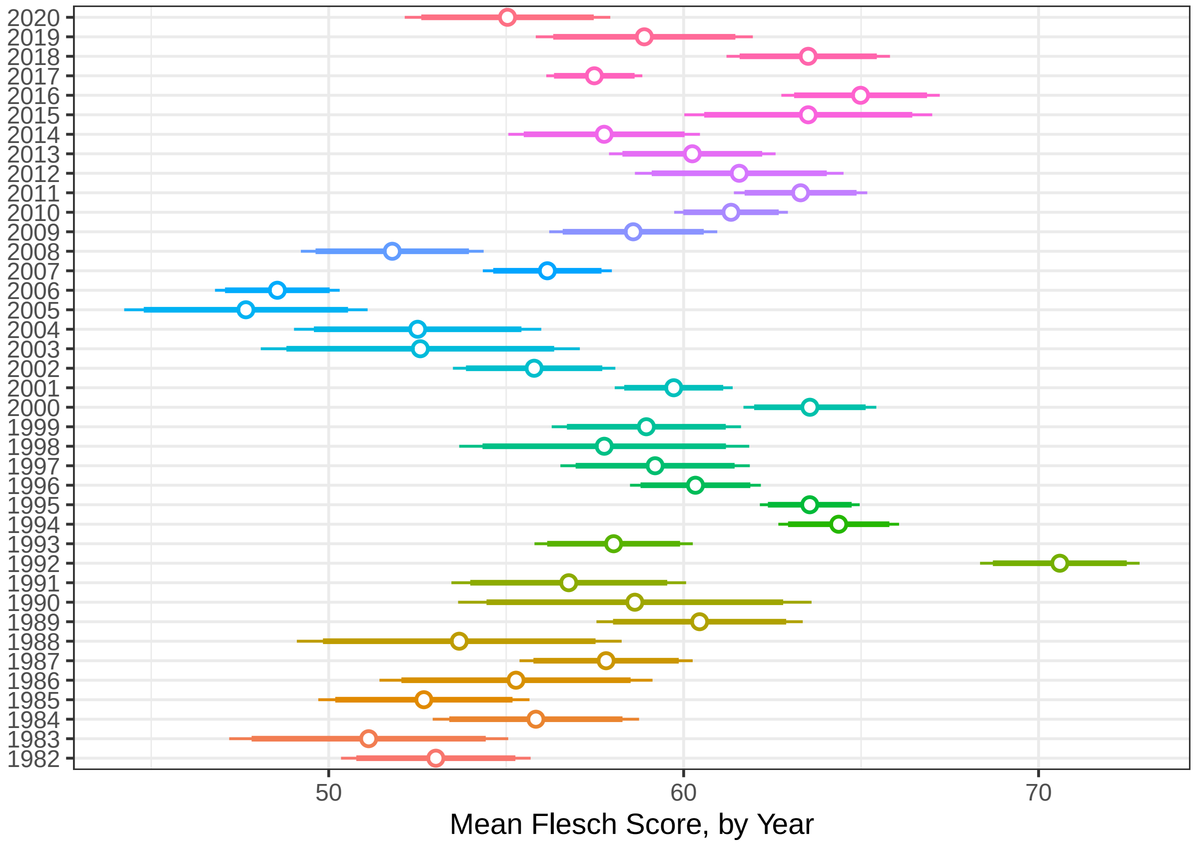
```
## $`2002`
## [1] 56.78444
##
## $`2003`
## [1] 53.8805
##
## $`2004`
## [1] 55.0094
##
## $`2005`
## [1] 47.02655
##
## $`2006`
## [1] 47.60247
##
## $`2007`
## [1] 57.11791
##
## $`2008`
## [1] 50.19596
##
## $`2009`
## [1] 60.85517
##
## $`2010`
## [1] 61.28102
##
## $`2011`
## [1] 62.17157
##
## $`2012`
## [1] 60.94132
##
## $`2013`
## [1] 60.9502
##
## $`2014`
## [1] 58.26783
##
## $`2015`
## [1] 61.74464
##
## $`2016`
## [1] 60.73028
##
## $`2017`
## [1] 59.68024
##
## $`2018`
## [1] 63.93203
##
## $`2019`
## [1] 61.48394
##
```

```
## $`2020`
## [1] 55.88464
```

```
iter <- 10 # NUMBER OF BOOTSTRAP SAMPLES (usually would want more, >=100)

## for loop to compute as many samples as specified
for(i in 1:iter) {
  if(i==1) {boot.means <- list()} # generate new list

  # store the results in new element i
  boot.means[[i]] <- lapply(sotu.split, boot.fre)
  print(paste("Iteration", i))
}
```

```
## [1] "Iteration 1"
## [1] "Iteration 2"
## [1] "Iteration 3"
## [1] "Iteration 4"
## [1] "Iteration 5"
## [1] "Iteration 6"
## [1] "Iteration 7"
## [1] "Iteration 8"
## [1] "Iteration 9"
## [1] "Iteration 10"
```

```
## combine the point estimates to a data frame and compute statistics by party
boot.means.df <- do.call(rbind.data.frame, boot.means)
mean.boot <- apply(boot.means.df, 2, mean)
sd.boot <- apply(boot.means.df, 2, sd)

## create data frame for plot
plot_df <- data.frame(sort(unique(sotu.df$year)), mean.boot, sd.boot) |>
  setNames(c("year", "mean", "se"))

## confidence intervals
ci90 <- qnorm(0.95)
ci95 <- qnorm(0.975)

## ggplot point estimate + variance
ggplot(plot_df, aes(colour = year)) + # general setup for plot
  geom_linerange(aes(x = year,
                     ymin = mean - se*ci90,
                     ymax = mean + se*ci90),
                 lwd = 1, position = position_dodge(width = 1/2)) + # plot 90% interval
  geom_pointrange(aes(x = year,
                      y = mean,
                      ymin = mean - se*ci95,
                      ymax = mean + se*ci95),
                  lwd = 1/2, position = position_dodge(width = 1/2),
                  shape = 21, fill = "WHITE") + # plot point estimates and 95% interval
  coord_flip() + # fancy stuff
  theme_bw() + # fancy stuff
  xlab("") + ylab("Mean Flesch Score, by Year") + # fancy stuff
  theme(legend.position = "none") # fancy stuff
```



10b)

```
## mean Flesch statistic by year
flesch_point <- sotu.df$text %>%
  textstat_readability(measure = "Flesch") %>%
  group_by(sotu.df$year) %>%
  summarise(mean_flesch = mean(Flesch)) %>%
  setNames(c("year", "mean")) %>%
  arrange(as.numeric(year))

cbind(flesch_point, "bs_mean" = plot_df$mean)
```

```
##    year     mean  bs_mean
## 1  1982 52.80153 53.01784
## 2  1983 50.76320 51.12585
## 3  1984 55.86283 55.83683
## 4  1985 53.88412 52.68090
## 5  1986 56.55134 55.27634
## 6  1987 57.53704 57.81484
## 7  1988 53.31507 53.67749
## 8  1989 60.89586 60.44943
## 9  1990 59.29738 58.62461
## 10 1991 56.68247 56.76271
## 11 1992 70.47788 70.59865
```

```
## 12 1993 57.22686 58.02709
## 13 1994 64.29242 64.36846
## 14 1995 64.28937 63.55329
## 15 1996 60.46150 60.33186
## 16 1997 58.66844 59.19581
## 17 1998 57.25828 57.76236
## 18 1999 58.48922 58.94872
## 19 2000 62.93560 63.55672
## 20 2001 58.92055 59.72000
## 21 2002 56.50193 55.78723
## 22 2003 53.11198 52.58009
## 23 2004 52.78702 52.50568
## 24 2005 47.10176 47.66671
## 25 2006 48.22666 48.55229
## 26 2007 55.81772 56.15958
## 27 2008 52.12939 51.78987
## 28 2009 59.06057 58.57935
## 29 2010 61.32735 61.33479
## 30 2011 63.36712 63.29481
## 31 2012 61.45300 61.56613
## 32 2013 60.15540 60.24359
## 33 2014 58.11050 57.75980
## 34 2015 62.81994 63.51062
## 35 2016 64.77663 64.98439
## 36 2017 58.01080 57.48267
## 37 2018 63.02524 63.51011
## 38 2019 59.67678 58.89131
## 39 2020 55.15502 55.03720
```

10c)

```
## calculate the FRE score and the Dale-Chall score.
fre_and_dc_measures <- textstat_readability(sotu.sub, c("Flesch", "FOG"))

## compute correlations
readability_cor <- cor(cbind(fre_and_dc_measures$Flesch, fre_and_dc_measures$FOG))

## print
print(readability_cor[1,2])
```

```
## [1] -0.8681133
```

Question 11

```
docs <- corpus( readtext(paste0(c("melania", "michelle"), ".txt")) )

## set gap to default (-1)
sw2 <- smith_waterman(as.character(docs)[1], as.character(docs)[2],
                      type="words", gap=-1)
```

```r
## increase gap penalty to -5 --> reduces extent of plagiarism. Why?
sw3 <- smith_waterman(as.character(docs)[1], as.character(docs)[2],
                      type="words", gap=-5)


print(sw2$sw)
```

```
## [1] 84
```

```r
print(sw3$sw)
```

```
## [1] 47
```