STAT 243: Problem Set 3

Jin Rou New

September 28, 2014

1 Problem 1

I broke down the problem into the following steps:

1. Extract all speech URLs from the speech directory page with `GetSpeechURLs`.
2. Download a speech given the URL, extract and format the speech text and other information such as president, year and number of laughter and applause tags with `GetSpeech`.
3. Extract individual words from a speech with `ExtractWordsFromSpeech`.
4. Extract individual sentences from a speech with `ExtractSentencesFromSpeech`.
5. Get all relevant textual information about the speech with `GetSpeechData`. Information extracted includes:
   - Number of words in speech.
   - Number of sentences in speech.
   - Average length of words in speech.
   - Average length of sentences in speech.
   - Average length of sentences in speech.
   - Count of word/phrase occurrences of given words/phrases, including the count of the number of figures/statistics used. This was done with two functions `CountWordOccurrences` and `CountPhraseOccurrences`
   - (and for extra credit):
   - Count of word/phrase occurrences of given words/phrases, normalised by the total number of words in the text.
   - Lexical diversity, which is the ratio of the number of unique words to the total number of words (a crude calculation because words should be stemmed before counting the number of unique words, but not done in this case).
   - Flesch-Kincaid readability score, a measured of the readability level of text, expressed as a U.S. grade level, i.e. The level of education generally required to understand the text (may be a number greater than 12th grade).
   - Flesch-Kincaid readability score, expressed as an age.
   - Word frequencies of all words in each speech sorted in order of decreasing word frequencies.
6. Combine steps 2-5 in an overall function to download and process a speech into a list containing the speech text and information about it with `GetAndProcessSpeech`.
7. Do step 6 for all speech URLs obtained from step 1 with `lapply` on `GetAndProcessSpeech`.
8. Pull information about all speeches into a data frame for data analysis and visualization with `GetSpeechDataFrame`

9. Finally, make exploratory plots of the data and select interesting ones to present.

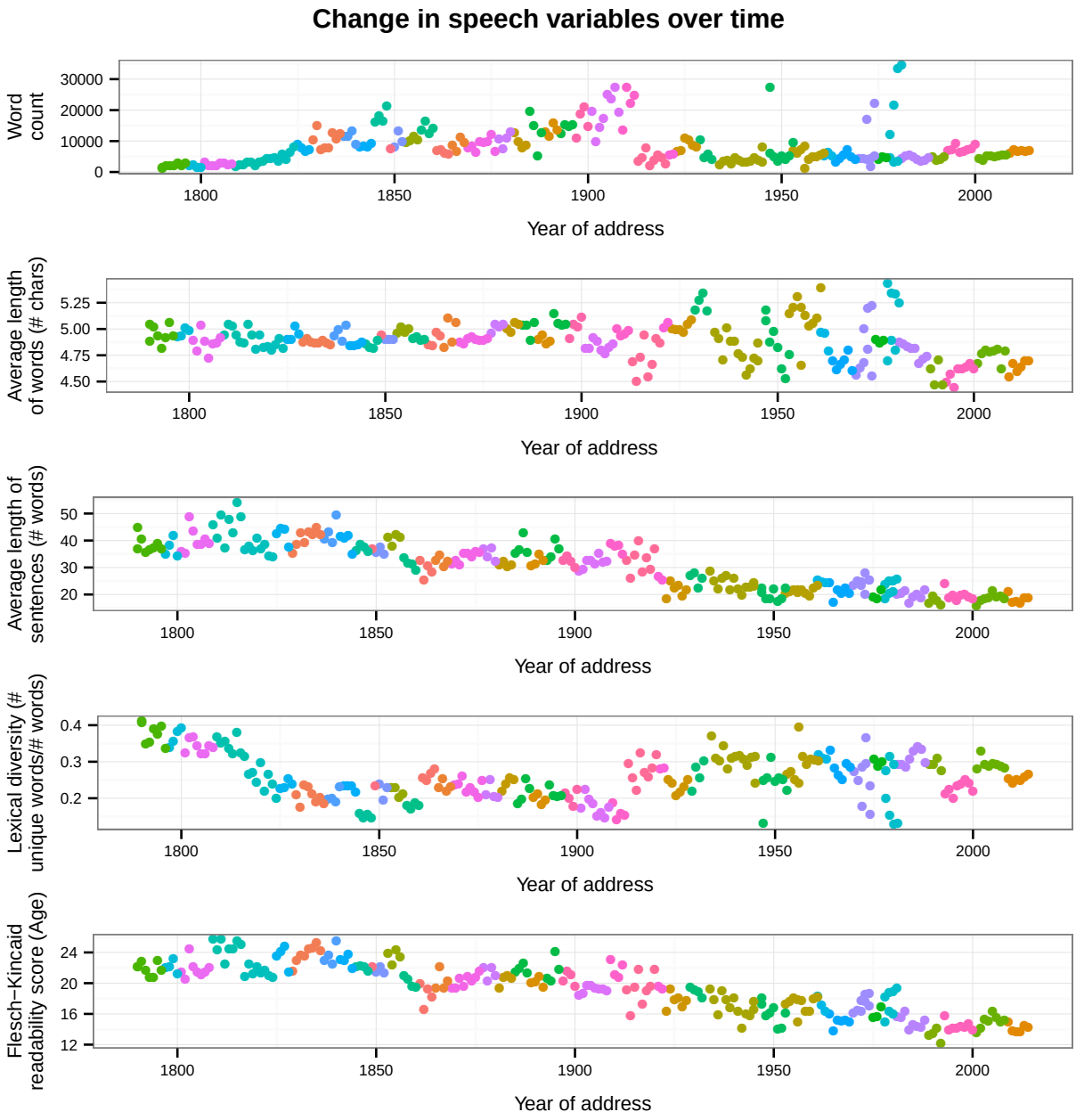
In my main script, I download and process all speeches.

```
library(XML)
library(stringr)
library(koRpus)
library(ggplot2)
source("speech-functions.R")
output_dir <- "output"
# Get list of speech objects
dir.create(output_dir, showWarnings = FALSE)
if (!file.exists(file.path(output_dir, "speech_list_all.rda"))) {
  urls_speech <- GetSpeechURLs()
  speech_list_all <- lapply(urls_speech, GetAndProcessSpeech)
  names(speech_list_all) <- sapply(speech_list_all, function(speech_list)
    paste(speech_list$speech_data_scalar$president, speech_list$speech_data_scalar$year))
  save(speech_list_all, file = file.path(output_dir, "speech_list_all.rda"))
} else {
  load(file.path(output_dir, "speech_list_all.rda"))
}

# Get speech data frame
if (!file.exists(file.path(output_dir, "speech_df.rda"))) {
  speech_df <- GetSpeechDataFrame(speech_list_all)
  save(speech_df, file = file.path(output_dir, "speech_df.rda"))
} else {
  load(file.path(output_dir, "speech_df.rda"))
}
```

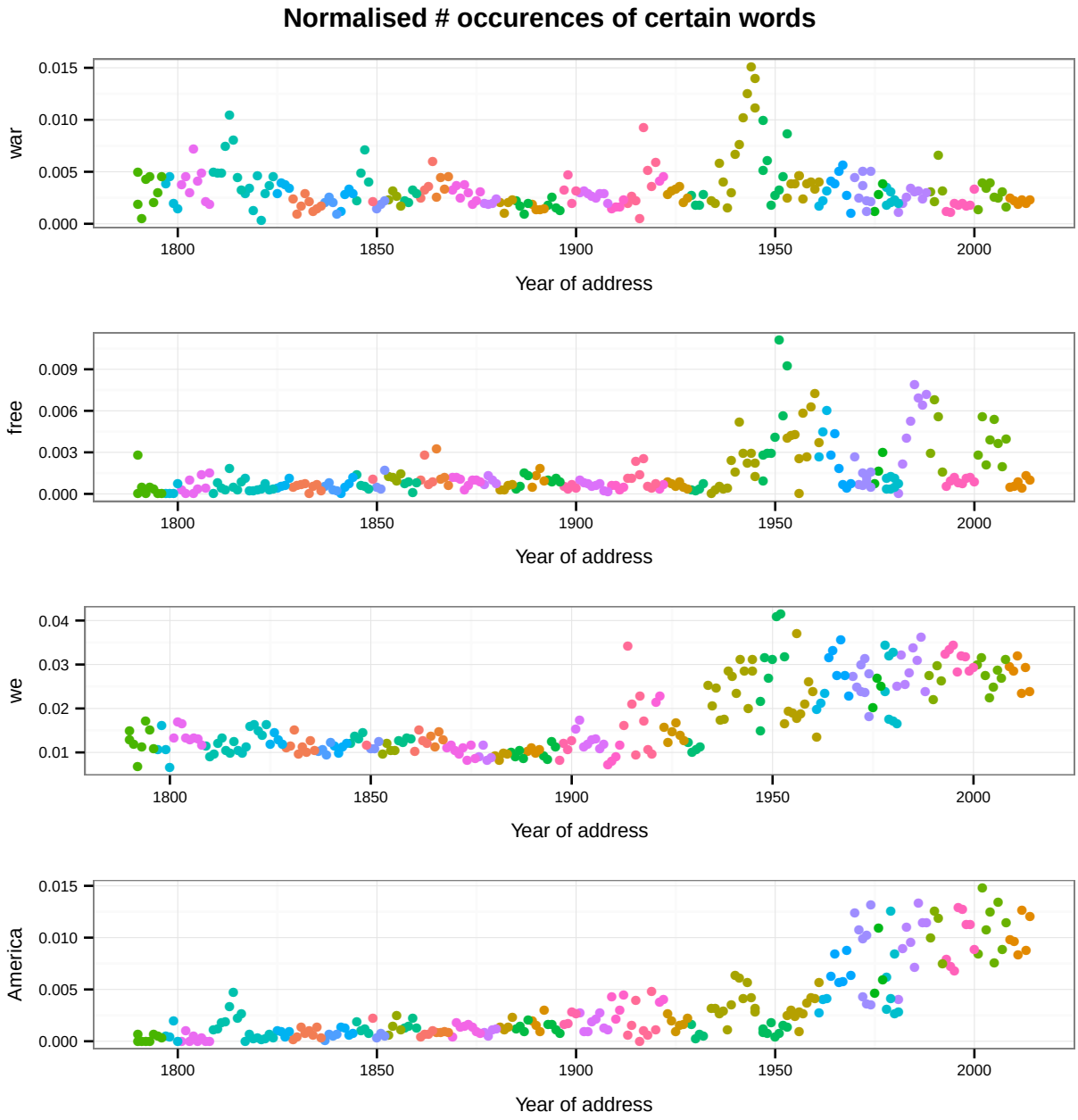
Next, I make exploratory plots of the data.

```
# Make exploratory plots of variables over time
vars_to_plot <- c("num_words", "avg_word_nchars", "avg_sentence_nwords",
                  "lexical_diversity", "readability_age",
                  "war", "free", "we", "America")
labels_to_plot <- c("Word\ncount", "Average length\nof words (# chars)",
                    "Average length of\nsentences (# words)",
                    "Lexical diversity (#\nunique words/# words)",
                    "Flesch-Kincaid\nreadability score (Age)",
                    "war", "free", "we", "America")

theme_set(theme_bw(base_size = 9) +
            theme(axis.title.x = element_text(vjust = -0.3),
                  axis.title.y = element_text(vjust = 1.1)))
p <- list()
for (i in seq_along(vars_to_plot)) {
  p[[i]] <- ggplot(speech_df,
                   aes_string(x = "year", y = vars_to_plot[i],
                              color = "president")) +
    geom_point() +
    xlab("Year of address") + ylab(labels_to_plot[i]) +
    scale_colour_hue(guide = FALSE)
}
```

```
# Plots of text variables
do.call(grid.arrange,
        c(p[1:5], list(nrow = 5, ncol = 1,
                       main = textGrob("Change in speech variables over time",
                                       gp = gpar(font=2)))))
```

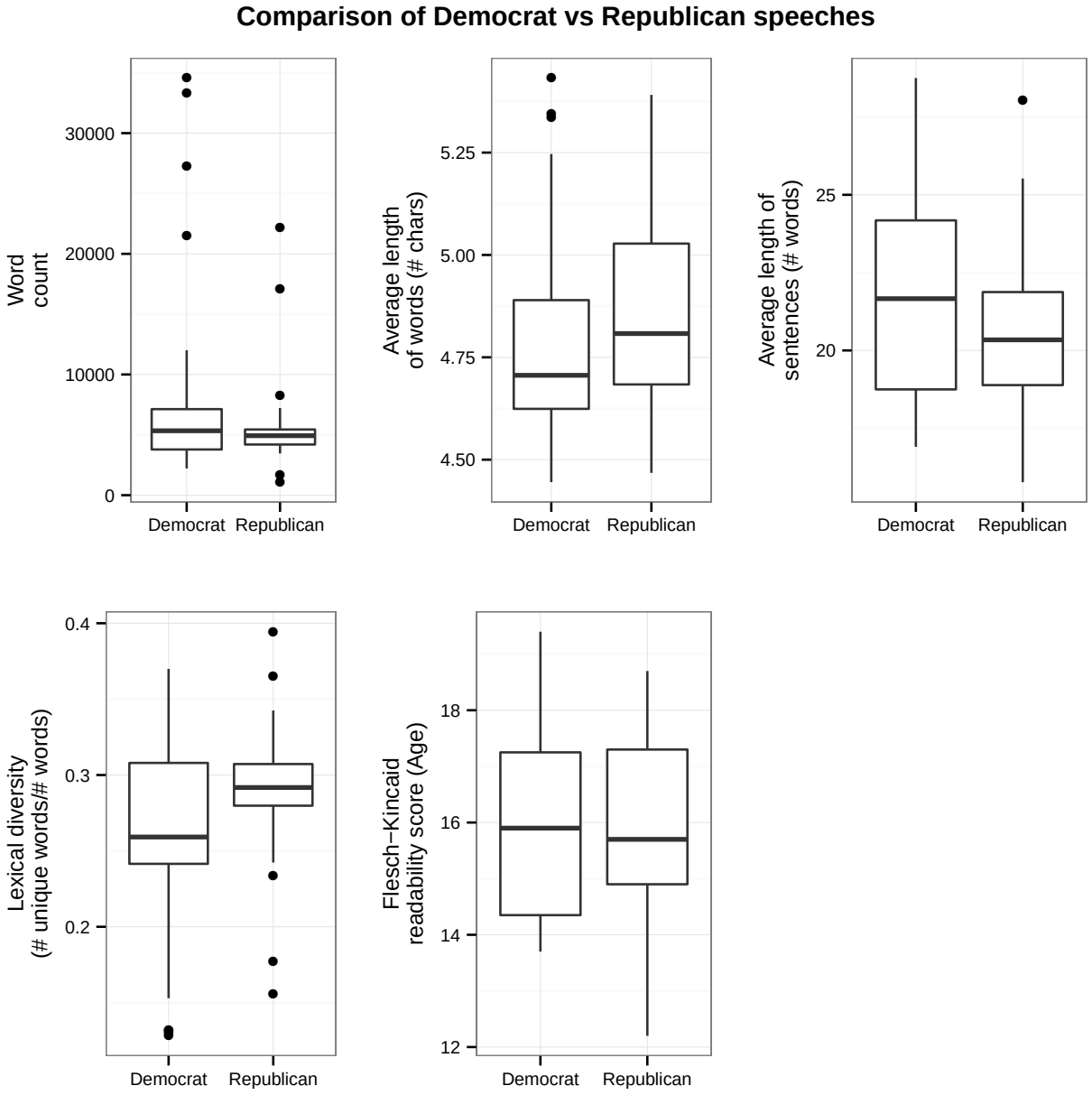



```
# Plots of word occurences
do.call(grid.arrange,
        c(p[6:9], list(nrow = 4, ncol = 1,
                       main = textGrob("Normalised # occurences of certain words",
                                       gp = gpar(font=2)))))
```




```
theme_set(theme_bw(base_size = 10) +
            theme(axis.title.x = element_text(vjust = -0.3),
                  axis.title.y = element_text(vjust = 1.1)))
# Make boxplots comparing Republican and Democratic presidents
comp_vars_to_plot <- vars_to_plot[1:5]
comp_labels_to_plot <- labels_to_plot[1:5]
boxplots <- vector("list", length(comp_vars_to_plot))
for (i in seq_along(comp_vars_to_plot)) {
  boxplots[[i]] <- ggplot(speech_df[!is.na(speech_df$party), ],
                          aes_string(x = "party", y = comp_vars_to_plot[i])) +
    geom_boxplot() +
    xlab("") +
    ylab(comp_labels_to_plot[i])
}
```

```
do.call(grid.arrange,
        c(boxplots, list(nrow = 2, ncol = 3,
                         main = textGrob("Comparison of Democrat vs Republican speeches",
                                         gp = gpar(font=2)))))
```



The functions called from the main script are given in the following pages.

```
# speech-functions.R

#' Download speech, then process it and return data about it.
#'
#' Download speech from URL, format it in various forms, then get data about it,
#' including number of words/sentences, average word/sentence length,
#' number of word/phrase occurences of given words/phrases.
#'
```

```r
#' @param words_to_count A character vector containing non-case-sensitive words to count.
#' @param words_to_count_cs A character vector containing case-sensitive words to count.
#' @param phrases_to_count A character vector containing non-case-sensitive phrases to count.
#' @param phrases_to_count_cs A character vector containing case-sensitive phrases to count.
#' @return A list object containing:
#' \describe{
#'   \item{\code{text_formatted}}{The full formatted speech text.}
#'   \item{\code{text_words}}{A character vector with words of the speech as individual elements.}
#'   \item{\code{text_sentences}}{A character vector with sentences of the speech as individual elements
#'   \item{\code{speech_data_scalar}}{Scalar data about the speech, which also include
#'   \describe{
#'      \item{\code{president}}{President who made the speech.}
#'      \item{\code{year}}{Year the speech was made.}
#'      \item{\code{count_audience_response}}{Numeric vector with number of times
#'      the audience laughed or applauded.}
#'   }}
#'   \item{\code{speech_data_vector}}{Vector data about the speech.}
#' }
GetAndProcessSpeech <- function(
  url_speech,
  words_to_count = c("we", "you", "they", "free", "war", "god",
                     "republic[[:punct:] ]", "[0-9]{1,}"),
  words_to_count_cs = c("I", "America", "Republican", "Democrat",
                        "democra", "Jesus|Christ|Christian"),
  phrases_to_count = c("god bless"),
  phrases_to_count_cs = NULL
) {
  speech_list <- GetSpeech(url_speech)

  # Get plain speech text in the form of a character vector of words/sentences
  text_words <- ExtractWordsFromSpeech(speech_list$text_formatted)
  text_sentences <- ExtractSentencesFromSpeech(speech_list$text_formatted)

  # Get speech-related data
  speech_data_all <- GetSpeechData(speech_list$text_formatted, text_words, text_sentences,
                                   words_to_count, words_to_count_cs,
                                   phrases_to_count, phrases_to_count_cs)
  speech_data_scalar <- c(list(president = speech_list$president,
                               year = speech_list$year,
                               num_laughter = speech_list$count_audience_response[["num_laughter"]],
                               num_applause = speech_list$count_audience_response[["num_applause"]]),
                          speech_data_all$speech_data_scalar)
  speech_list_final <- list(text_formatted = speech_list$text_formatted,
                            text_words = text_words,
                            text_sentences = text_sentences,
                            speech_data_scalar = speech_data_scalar,
                            speech_data_vector = speech_data_all$speech_data_vector)
  message(paste0("Processed speech by ", speech_list_final$speech_data_scalar$president,
                 " in ", speech_list_final$speech_data_scalar$year, "."))
  return(speech_list_final)
}

#' Extract scalar speech data and output a data frame.
```

```r
#'
#' Reads in a list of speech objects and extract scalar speech data into
#' a data frame for ease of analysis.
#'
#' @param speech_list_all A list of speech objects/lists.
#' @return A data frame containing scalar speech data.
GetSpeechDataFrame <- function(speech_list_all) {
  vars <- names(speech_list_all[[1]]$speech_data_scalar)
  speech_df_scalar <- vector("list", length = length(vars))
  for (i in seq_along(vars)) {
    speech_df_scalar[[i]] <- sapply(speech_list_all,
                                    function(speech_list, var) speech_list$speech_data_scalar[[var]],
                                    var = vars[i])
  }
  vars_count <- names(speech_list_all[[1]]$speech_data_vector$count_word_normalised)
  speech_df_count <- vector("list", length = length(vars_count))
  for (i in seq_along(vars_count)) {
    speech_df_count[[i]] <- sapply(speech_list_all,
                                       function(speech_list, var)
                                         speech_list$speech_data_vector$count_word_normalised[[var]],
                                       var = vars_count[i])
  }
  names(speech_df_scalar) <- vars
  names(speech_df_count) <- vars_count
  speech_df <- as.data.frame(c(speech_df_scalar, speech_df_count))
  # Add in party of presidents after 1932, set as NA for those before 1932
  speech_df$party <-
    ifelse(speech_df$year > 1932 &
             str_detect(speech_df$president,
                        "Eisenhower|Nixon|Ford|Reagan|Bush|Bush"),
           "Republican",
           ifelse(speech_df$year > 1932 &
                    str_detect(speech_df$president,
                               "Roosevelt|Truman|Kennedy|Johnson|Carter|Clinton|Obama"),
                  "Democrat", NA))
  print(head(speech_df))
  return(speech_df)
}

#' Get speech from the speech URL.
#'
#' Reads in the URL of the page with a listing of all State of the Union
#' Address speeches and returns all URLs of the speeches.
#'
#' @param url_speech_dir A character scalar containing the URL linking to a
#' page with a listing of all State of the Union Address speeches.
#' @return A character vector containing URLs of all State of the Union
#' Address speeches.
GetSpeechURLs <- function(url_speech_dir = "http://www.presidency.ucsb.edu/sou.php") {
  xml_speech_dir <- htmlParse(url_speech_dir, useInternalNodes = TRUE)
  urls_all <- xpathSApply(xml_speech_dir, "//a/@href")
  urls_speech <- unique(urls_all[str_detect(urls_all, "pid")])
  message(paste("There are", length(urls_speech), "speeches on url_speech_dir."))
```

```
  return(urls_speech)
}

#' Get speech from the speech URL.
#'
#' Reads in an URL of a page with a State of the Union Address and returns
#' the speech text and president and year of the speech.
#'
#' @param url_speech A character scalar containing the URL linking to a
#' State of the Union Address speech.
#' @return A list object containing:
#' \describe{
#'   \item{\code{text_formatted}}{The full formatted speech text.}
#'   \item{\code{president}}{President who made the speech.}
#'   \item{\code{year}}{Year the speech was made.}
#'   \item{\code{count_audience_response}}{Numeric vector with number of times
#'   the audience laughed or applauded.}
#' }
GetSpeech <- function(url_speech) {
  xml_speech <- htmlParse(url_speech, useInternalNodes = TRUE)

  # Extract speech text, stripping all HTML tags and convert text encoding to UTF-8
  text_paragraphs_raw <- xpathSApply(xml_speech, "//span[@class='displaytext']/p", xmlValue)
  text_paragraphs <- sapply(text_paragraphs_raw, iconv, to = "UTF-8")
  names(text_paragraphs) <- NULL

  # Get full formatted speech text with new paragraphs indicated by a new line
  text_formatted <- paste(text_paragraphs, collapse = " \n")
  text_formatted <- str_replace_all(text_formatted, "  ", " ")

  # Get number of occurences of "[Laughter]" and "[Applause]"
  # before removing them from speech text
  count_audience_response <- sapply(c("\\[Laughter\\]", "\\[Applause\\]"),
                                    str_count, string = text_formatted)
  names(count_audience_response) <- c("num_laughter", "num_applause")
  text_formatted <- str_replace_all(text_formatted, " \\[Laughter\\]| \\[Applause\\]", "")

  # Extract president name and year of speech
  citation <- paste(xpathSApply(xml_speech, "//span[@class='ver10']", xmlValue), collapse = "")
  president <- str_replace_all(str_extract_all(citation, perl("Citation:[a-zA-Z\\.\\s]+:"))[[1]],
                               "Citation:\\s|\\:", "")
  year <- as.integer(str_extract(citation, perl("[[:digit:]]{4}")))

  speech_list <- list(text_formatted = text_formatted,
                      president = president,
                      year = year,
                      count_audience_response = count_audience_response)
  return(speech_list)
}

#' Extract individual words from speech.
#'
#' Reads in the full formatted speech text and returns it as individual words.
```

```r
#'
#' @param text_formatted A character scalar containing the full formatted speech text
#' from \code{\link{GetSpeech}}.
#' @return A character vector with words of the speech as individual elements.
ExtractWordsFromSpeech <- function(text_formatted) {
  # Convert all commas and periods sandwiched by digits and periods following initlals
  # into &&&, ###, ~~~
  text_plain <- str_replace_all(text_formatted, "([0-9]+)\\.([0-9]+)", "\\1&&&\\2")
  text_plain <- str_replace_all(text_plain, "([0-9]+),([0-9]{3})", "\\1###\\2")
  text_plain <- str_replace_all(text_plain, "([A-Z])\\.", "\\1~~~")

  # Remove all punctuation marks that separate words
  text_plain <- str_replace_all(text_plain,
                                "\\.|,|\\?|!|:|;|\"|\n||--|\\[|\\]|\\(|\\)", "")

  # Replace &&& and ### back by periods and commas respectively
  text_plain <- str_replace_all(text_plain, "&&&", "\\.")
  text_plain <- str_replace_all(text_plain, "###", ",")
  text_plain <- str_replace_all(text_plain, "~~~", "\\.")
  text_words <- strsplit(text_plain, split = " ")[[1]]

  return(text_words)
}

#' Extract individual sentences from speech.
#'
#' Reads in the full formatted speech text and returns it as individual sentences.
#'
#' @param text_formatted A character scalar containing the full formatted speech text
#' from \code{\link{GetSpeech}}.
#' @return A character vector with sentences of the speech as individual elements.
ExtractSentencesFromSpeech <- function(text_formatted) {
  # Remove \n and quotation marks
  text_plain <- str_replace_all(text_formatted,
                                "\n|\"", "")
  # Remove period after honorifics to avoid splitting on them
  text_plain <- str_replace_all(text_plain,
                                "(Mr|Mrs|Ms|Mdm|Dr|Lt|Hon|Jr)\\.",
                                "\\1")
  # Substitute period after initials to avoid splitting on them
  text_plain <- str_replace_all(text_plain, "([A-Z])\\.", "\\1~~~")

  # Prepend an extra period/question mark/exclamation mark to each of these punctuation marks
  # then split into individual sentences
  text_temp <- str_replace_all(text_plain, "\\. ", "\\.\\. ")
  text_temp <- str_replace_all(text_temp, "\\? ", "\\?\\? ")
  text_temp <- str_replace_all(text_temp, "! ", "!! ")
  text_sentences <- strsplit(text_temp, split = "\\. |\\? |! ")[[1]]

  # Put back period after initials
  text_sentences <- str_replace_all(text_sentences, "~~~", "\\.")
  return(text_sentences)
}
```

```r
#' Get data about a speech.
#'
#' Get data about a speech, including number of words/sentences,
#' average word/sentence length, number of word occurences of given words.
#'
#' @param text_formatted A character scalar containing the full formatted speech text
#' from \code{\link{GetSpeech}}.
#' @param text_words A character vector with sentences of the speech as individual elements
#' from \code{\link{ExtractSentencesFromSpeech}}.
#' @param text_sentences A character vector with sentences of the speech as individual elements
#' from \code{\link{ExtractWordsFromSpeech}}.
#' @param words_to_count A character vector containing non-case-sensitive words to count.
#' @param words_to_count_cs A character vector containing case-sensitive words to count.
#' @param phrases_to_count A character vector containing non-case-sensitive phrases to count.
#' @param phrases_to_count_cs A character vector containing case-sensitive phrases to count.
#' @return A list containing:
#' \describe{
#'   \item{\code{speech_data_scalar}}{A list containing:
#'     \describe{
#'       \item{\code{num_words}}{Numeric scalar. Number of words in speech.}
#'       \item{\code{num_sentences}}{Numeric scalar. Number of sentences in speech.}
#'       \item{\code{avg_word_nchars}}{Numeric scalar. Average length of words in speech.}
#'       \item{\code{avg_sentence_nwords}}{Numeric scalar. Average length of sentences in speech.}
#'       \item{\code{num_unique_words}}{Numeric scalar. Average length of sentences in speech.}
#'       \item{\code{lexical_diversity}}{Numeric scalar. Lexical diversity,
#'       which is the ratio of the number of unique words to the total number of words.}
#'       \item{\code{readability_grade}}{Numeric scalar. Flesch-Kincaid readability score
#'       expressed as a U.S. grade level.}
#'       \item{\code{readability_age}}{Numeric scalar. Flesch-Kincaid readability score
#'       expressed as an age.}
#'     }
#'   }
#'   \item{\code{speech_data_vector}}{A list containing:
#'     \describe{
#'       \item{\code{word_frequencies}}{Named numeric vector of word frequencies
#'       sorted in order of decreasing word frequencies.}
#'       \item{\code{count_word}}{A numeric vector with the count of word/phrase
#'       occurrences of given words/phrases.}
#'       \item{\code{count_word_normalised}}{A numeric vector with the normalised
#'       count of word/phrase occurrences of given words/phrases.}
#'     }
#'   }
#' }
GetSpeechData <- function(text_formatted, text_words, text_sentences,
                          words_to_count, words_to_count_cs,
                          phrases_to_count, phrases_to_count_cs) {
  # Get count of words and sentences in speech text
  num_words <- length(text_words)
  num_sentences <- length(text_sentences)

  # Get average word and sentence lengths
  avg_word_nchars <- mean(nchar(text_words))
  text_sentences_words <- strsplit(text_sentences, " ")
```

```
  avg_sentence_nwords <- mean(sapply(text_sentences_words, length))

  # Get lexical diversity
  num_unique_words <- length(unique(tolower(text_words)))
  lexical_diversity <- num_unique_words/num_words

  # Get readability score
  tokens <- tokenize(text_formatted, format = "obj", lang = "en")
  readability <- flesch.kincaid(tokens)
  readability_grade <- as.numeric(summary(readability)$grade)
  readability_age <- as.numeric(summary(readability)$age)

  # Get word count of each word in decreasing order of frequency
  word_frequencies <- sort(table(tolower(text_words)), decreasing = TRUE)

  # Get number of word/phrase occurences
  count_word <- NULL
  if (!is.null(phrases_to_count))
    count_word <- c(count_word, CountPhraseOccurrences(text_formatted, phrases_to_count,
                                                       ignore_case = TRUE, perl = TRUE))
  if (!is.null(phrases_to_count_cs))
    count_word <- c(count_word, CountPhraseOccurrences(text_formatted, phrases_to_count_cs,
                                                       ignore_case = FALSE, perl = TRUE))
  if (!is.null(words_to_count))
    count_word <- c(count_word, CountWordOccurrences(text_words, words_to_count,
                                                     ignore.case = TRUE, perl = TRUE))
  if (!is.null(words_to_count_cs))
    count_word <- c(count_word, CountWordOccurrences(text_words, words_to_count_cs,
                                                     ignore.case = FALSE, perl = TRUE))
  # Exclude count of "god bless" from count of "god"
  count_word[["god"]] <- count_word[["god"]] - count_word[["god bless"]]
  # Normalise word count
  count_word_normalised <- count_word/num_words

  speech_data_scalar <- list(num_words = num_words,
                             num_sentences = num_sentences,
                             avg_word_nchars = avg_word_nchars,
                             avg_sentence_nwords = avg_sentence_nwords,
                             num_unique_words = num_unique_words,
                             lexical_diversity = lexical_diversity,
                             readability_grade = readability_grade,
                             readability_age = readability_age)
  speech_data_vector <- list(word_frequencies = word_frequencies,
                             count_word = count_word,
                             count_word_normalised = count_word_normalised)
  return(list(speech_data_scalar = speech_data_scalar,
              speech_data_vector = speech_data_vector))
}

#' Count word occurences of multiple words.
#'
#' Count the number of times each word provided in a character vector
#' occurs in a given text.
```

```r
#'
#' @param text_words A character vector with words of the speech as individual elements
#' from \code{\link{ExtractWordsFromSpeech}}.
#' @param words_to_count A character vector containing words to count.
#' @param ... Other arguments for \code{grepl}
#' @return A numeric vector with counts of each word.
CountWordOccurrences <- function(text_words, words_to_count, ...) {
  count_words <- sapply(words_to_count, function(string, pattern, ...) {
    sum(grepl(pattern, string, ...))
  }, string = text_words, ...)
  return(count_words)
}

#' Count phrase occurences of multiple phrases.
#'
#' Count the number of times each phrase provided in a character vector
#' occurs in a given text.
#'
#' @param text A character scalar containing speech text.
#' @param phrases_to_count A character vector containing words to count.
#' @param ignore_case A logical scalar. Ignore case of matches?
#' @param perl A logical scalar. Use Perl-like regular expressions?
#' @return A numeric vector with counts of each phrase.
CountPhraseOccurrences <- function(text, phrases_to_count, ignore_case, perl = TRUE) {
  count_phrases <- sapply(phrases_to_count, function(string, pattern, ignore_case, perl) {
    if (ignore_case & perl) {
      count_phrases <- str_count(string, ignore.case(perl(pattern)))
    } else if (ignore_case) {
      count_phrases <- str_count(string, ignore_case(pattern))
    } else if (perl) {
      count_phrases <- str_count(string, perl(pattern))
    } else {
      count_phrases <- str_count(string, pattern)
    }
  }, string = text, ignore_case = ignore_case, perl = perl)
  return(count_phrases)
}
```