ps3

Ramon Crespo

9/28/2018

Problem2. Various different problems. For the different parts see comments in the code.

```
## Note that this code uses the XML package rather than xml2 and rvest
## simply because I had this code sitting around from a previous demonstration.

## @knitr download

moderators <- c("LEHRER", "LEHRER", "LEHRER", "MODERATOR", "LEHRER", "HOLT")
words <- c("I", "we", "America", "American", "democracy","democratic", "republic", "Democrat","Democrati
words[1]
```

```
## [1] "I"
```

```
moderators[[3]]
```

```
## [1] "LEHRER"
```

```
candidates <- list(c(Dem = "CLINTON", Rep = "TRUMP"),
                   c(Dem = "OBAMA", Rep = "ROMNEY"),
                   c(Dem = "OBAMA", Rep = "MCCAIN"),
                   c(Dem = "KERRY", Rep = "BUSH"),
                   c(Dem = "GORE", Rep = "BUSH"),
                   c(Dem = "CLINTON", Rep = "DOLE"))

candidates[[3]][[1]]
```

```
## [1] "OBAMA"
```

```
library(XML)
library(stringr)
```

```
##
## Attaching package: 'stringr'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     words
```

```
library(assertthat)
library(gsubfn)
```

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

```
## Warning in doTryCatch(return(expr), name, parentenv, handler): unable to load shared object '/Library
##   dlopen(/Library/Frameworks/R.framework/Resources/modules//R_X11.so, 6): Library not loaded: /opt/X1
##   Referenced from: /Library/Frameworks/R.framework/Resources/modules//R_X11.so
##   Reason: image not found
```

```
## Could not load tcltk.  Will use slower R code instead.
```

```
library(tidyr)

url <- "http://www.debates.org/index.php?page=debate-transcripts"
```

```
yrs <- seq(1996, 2012, by = 4)
type <- 'first'
main <- htmlParse(url)
listOfANodes <- getNodeSet(main, "//a[@href]")
labs <- sapply(listOfANodes, xmlValue)
inds_first <- which(str_detect(labs, "The First"))
## debates only from the specified years
inds_within <- which(str_extract(labs[inds_first], "\\d{4}")
                     %in% as.character(yrs))
inds <- inds_first[inds_within]
## add first 2016 debate, which is only in the sidebar
ind_2016 <- which(str_detect(labs, "September 26, 2016"))
inds <- c(ind_2016, inds)
debate_urls <- sapply(listOfANodes, xmlGetAttr, "href")[inds]

n <- length(debate_urls)

assert_that(n == length(yrs)+1)
```

```
## [1] TRUE
```

```
## @knitr extract

debates_html <- sapply(debate_urls, htmlParse)

get_content <- function(html) {
  # get core content containing debate text
  contentNode <- getNodeSet(html, "//div[@id = 'content-sm']")
  if(length(contentNode) > 1)
    stop("Check why there are multiple chunks of content.")
  text <- xmlValue(contentNode[[1]])
  # sanity check:
  print(xmlValue(getNodeSet(contentNode[[1]], "//h1")[[1]]))
  return(text)
}

debates_body <- sapply(debates_html, get_content)
```

```
## [1] "September 26, 2016 Debate Transcript"
## [1] "October 3, 2012 Debate Transcript"
## [1] "September 26, 2008 Debate Transcript"
## [1] "September 30. 2004 Debate Transcript"
## [1] "October 3, 2000 Transcript"
## [1] "October 6, 1996 Debate Transcript"
```

```
## sanity check
print(substring(debates_body[6], 1, 1000))
```

```
##
## "\nOctober 6, 1996 Debate Transcript\n\nOctober 6, 1996The First Clinton-Dole Presidential DebateLEH
```

```
setClass(Class="Debate",
         representation(
            text_of_debate="character",
            words="numeric",
```

```
            special_words = "numeric"
          )
)

return_word_frequency <- function(input_data){
  words_freq<-table(unlist(input_data))
  d2 <- as.data.frame(words_freq)
  d2 <- spread(data = d2,key = Var1, value = Freq)
  words_length = length(words)
  vec <- c(1:words_length)
  count <- 0

  for (val in words){
    count <- count + 1
    temp <- try(d2[[val]])
    if (is.null(temp)){
      vec[count] <- 0
    } else{
      vec[count] <- temp
    }
  }
  return(vec)
}


get_debate_information <- function(text,republican_candidate,democrat_candidate,moderator){
  # The function takes in a string of text from the presidential debates and returns information about
  # Input:
  #     republican -> name of the republican pressident in the debate
  #     democrat   -> name of the democrat pressident in the debate
  #Output:
  #     democrat_text -> text spoken by democrat
  #     republican_text -> text spoken by republican candidate
  #     moderator -> text spoken by moderator
  # This function will save two files in your corren working directory
  #     i) the general text, parsed in a dataframe, named after the yearof the debate
  #     i) A second data frame containing the results of short analysis on the word count and word leng
  #replace names with usefull tags

  current_debate <- text
  current_debate <- gsub(democrat_candidate, "<d>", current_debate)
  current_debate <- gsub(republican_candidate, "<r>", current_debate)
  current_debate <- gsub(moderator, "<m>", current_debate)

  #split text according to tags
  current_debate_split <- str_split(current_debate, "<")

  #store candidates text into variables
  applause_tag = "APPLAUSE"
  laughter_tag = "LAUGHTER"
  republican_text <- grep("^r",current_debate_split[[1]], value=TRUE)
  democrat_text <- grep("^d",current_debate_split[[1]], value=TRUE)
  moderator_text <- grep("^r",current_debate_split[[1]], value=TRUE)
```

```
  #Store relevant values before deleting them
  applause_republican <- str_count(republican_text, applause_tag)
  laughter_republican <- str_count(republican_text, laughter_tag)
  applause_democrat <- str_count(democrat_text, applause_tag)
  laughter_democrat <- str_count(democrat_text, laughter_tag)

  print(length(republican_text))
  print(length(democrat_text))
  print(length(moderator_text))
}

get_debate_statistics <- function(text,republican_candidate,democrat_candidate,moderator){
  current_debate <- text
  current_debate <- gsub(democrat_candidate, "<d>", current_debate)
  current_debate <- gsub(republican_candidate, "<r>", current_debate)
  current_debate <- gsub(moderator, "<m>", current_debate)

  #split text according to tags
  current_debate_split <- str_split(current_debate, "<")

  #store candidates text into variables
  applause_tag = "APPLAUSE"
  laughter_tag = "LAUGHTER"
  republican_text <- grep("^r",current_debate_split[[1]], value=TRUE)
  democrat_text <- grep("^d",current_debate_split[[1]], value=TRUE)
  moderator_text <- grep("^r",current_debate_split[[1]], value=TRUE)

  #This nex line will save the results in your current working diretor

  #Part b. The following lines are designed to parse the data into individual strings. Since I included
  republican_text <- gsub("\\.","",republican_text)
  democrat_text <- gsub("\\.","",democrat_text)
  moderator_text <- gsub("\\.","",moderator_text)
  republican_text <- gsub("\\,","",republican_text)
  moderator_text <- gsub("\\,","",republican_text)
  republican_text <- gsub("\\,","",republican_text)

  republican_words <- unlist(strsplit(republican_text, " "))
  democrat_words <- unlist(strsplit(democrat_text, " "))
  moderator_words <- unlist(strsplit(moderator_text, " "))

  #Part c. Use the already split data to count the number of words letters and compute the average word
  #c.1 count words
  republican_word_count <- sum(sapply(republican_words, length))
  democrat_word_count <- sum(sapply(democrat_words, length))
  moderator_word_count <- sum(sapply(moderator_words, length))

  #c.2 count letters
  characters_republican <- sum(nchar(republican_words,type="chars"))
  characters_democrat <- sum(nchar(democrat_words,type="chars"))
  characters_moderator <- sum(nchar(moderator_words,type="chars"))

  #c.3 compute avg word length
```

```
  avg_wrd_lngth_rep <- characters_republican/republican_word_count
  avg_wrd_lngth_dem <- characters_democrat/democrat_word_count
  avg_wrd_lngth_mod <- characters_moderator/moderator_word_count

  d1 <- data.frame("republican_avg_word"=avg_wrd_lngth_rep,"democrat_avg_word" = avg_wrd_lngth_dem,"mod
  return(d1)
}

get_debate_special_words <- function(text,republican_candidate,democrat_candidate,moderator){
  current_debate <- text
  current_debate <- gsub(democrat_candidate, "<d>", current_debate)
  current_debate <- gsub(republican_candidate, "<r>", current_debate)
  current_debate <- gsub(moderator, "<m>", current_debate)

  #split text according to tags
  current_debate_split <- str_split(current_debate, "<")

  #store candidates text into variables
  applause_tag = "APPLAUSE"
  laughter_tag = "LAUGHTER"
  republican_text <- grep("^r",current_debate_split[[1]], value=TRUE)
  democrat_text <- grep("^d",current_debate_split[[1]], value=TRUE)
  moderator_text <- grep("^r",current_debate_split[[1]], value=TRUE)


  #Part b. The following lines are designed to parse the data into individual strings. Since I included
  republican_text <- gsub("\\.","",republican_text)
  democrat_text <- gsub("\\.","",democrat_text)
  moderator_text <- gsub("\\.","",moderator_text)
  republican_text <- gsub("\\,","",republican_text)
  moderator_text <- gsub("\\,","",republican_text)
  republican_text <- gsub("\\,","",republican_text)

  republican_words <- unlist(strsplit(republican_text, " "))
  democrat_words <- unlist(strsplit(democrat_text, " "))
  moderator_words <- unlist(strsplit(moderator_text, " "))
  #Partd.
  #count the words in the global variable words
  specific_words_count_rep <- return_word_frequency(republican_words)
  specific_words_count_dem <- return_word_frequency(democrat_words)
  specific_words_count_mod <- return_word_frequency(moderator_words)

  d2 <- data.frame("republican"=specific_words_count_rep,"democrat_avg_word" = specific_words_count_dem
  return(d2)
}


sixteen <- substring(debates_body[1], 1)
twelve <-substring(debates_body[2], 1)
eight <-substring(debates_body[3], 1)
four <-substring(debates_body[4], 1)
two <-substring(debates_body[5], 1)
ninetysix <- substring(debates_body[6], 1)
```

```
sixteen_data <- get_debate_information(sixteen,candidates[[1]][[1]],candidates[[1]][[2]],moderators[[1]]
```

```
## [1] 87
## [1] 124
## [1] 87
```

```
sixteen_length <- get_debate_statistics(sixteen,candidates[[1]][[1]],candidates[[1]][[2]],moderators[[1]
sixteen_words <- get_debate_special_words(sixteen,candidates[[1]][[1]],candidates[[1]][[2]],moderators[
```

```
twelve_data <- get_debate_information(twelve,candidates[[2]][[1]],candidates[[2]][[2]],moderators[[2]])
```

```
## [1] 58
## [1] 73
## [1] 58
```

```
twelve_length <- get_debate_statistics(twelve,candidates[[2]][[1]],candidates[[2]][[2]],moderators[[2]]
twelve_words <- get_debate_special_words(twelve,candidates[[2]][[1]],candidates[[2]][[2]],moderators[[2
print(twelve_data)
```

```
## [1] 58
```

```
eight_data <- get_debate_information(eight,candidates[[3]][[1]],candidates[[3]][[2]],moderators[[3]])
```

```
## [1] 128
## [1] 128
## [1] 128
```

```
eight_length <- get_debate_statistics(eight,candidates[[3]][[1]],candidates[[3]][[2]],moderators[[3]])
eight_words <- get_debate_special_words(eight,candidates[[3]][[1]],candidates[[3]][[2]],moderators[[3]]
print(eight_data)
```

```
## [1] 128
```

```
four_data <- get_debate_information(four,candidates[[4]][[1]],candidates[[4]][[2]],moderators[[4]])
```

```
## [1] 34
## [1] 42
## [1] 34
```

```
four_length <- get_debate_statistics(four,candidates[[4]][[1]],candidates[[4]][[2]],moderators[[4]])
four_words <- get_debate_special_words(four,candidates[[4]][[1]],candidates[[4]][[2]],moderators[[4]])
print(four_data)
```

```
## [1] 34
```

```
two_data <- get_debate_information(two,candidates[[5]][[1]],candidates[[5]][[2]],moderators[[5]])
```

```
## [1] 49
## [1] 56
## [1] 49
```

```
two_length <- get_debate_statistics(two,candidates[[5]][[1]],candidates[[5]][[2]],moderators[[5]])
two_words <- get_debate_special_words(two,candidates[[5]][[1]],candidates[[5]][[2]],moderators[[5]])
print(two_data)
```

```
## [1] 49
```

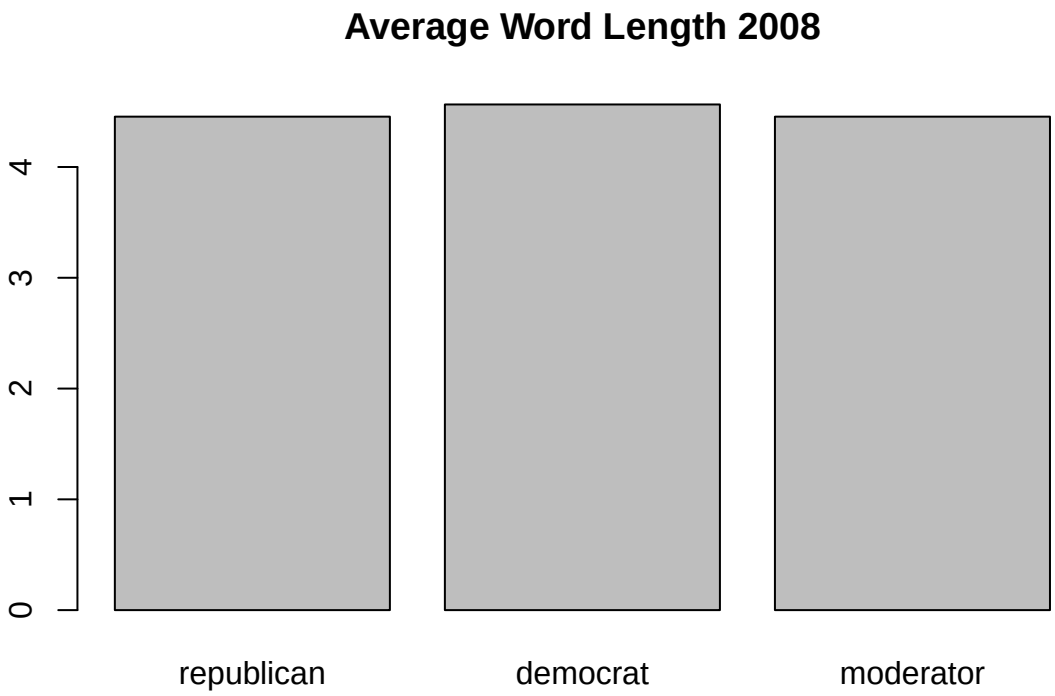
```
ninetysix_data <- get_debate_information(ninetysix,candidates[[6]][[1]],candidates[[6]][[2]],moderators
```

```
## [1] 45
## [1] 46
## [1] 45
```

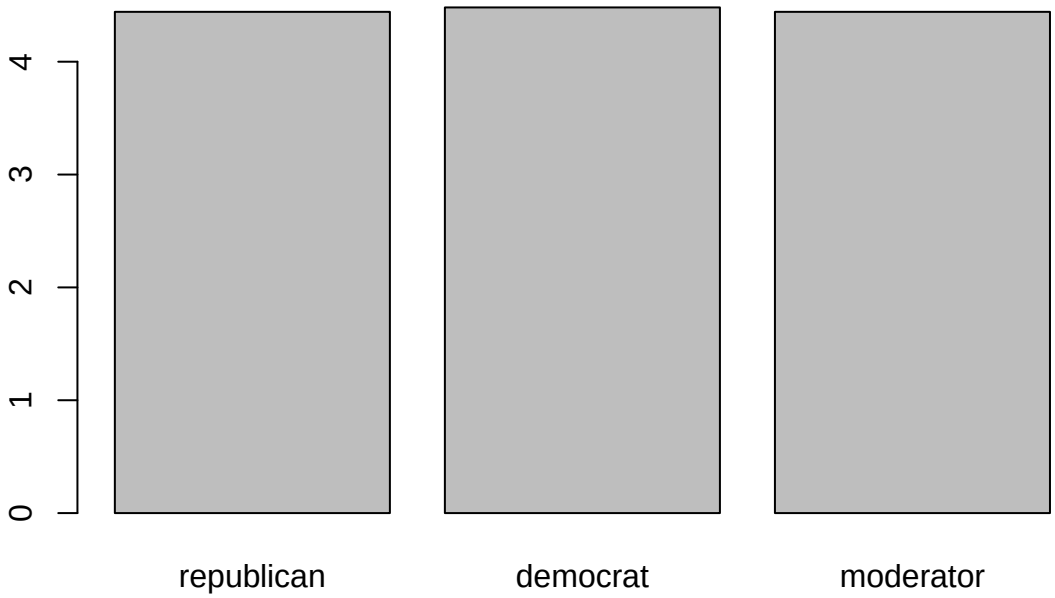
```
ninetysix_length <- get_debate_statistics(ninetysix,candidates[[6]][[1]],candidates[[6]][[2]],moderators
ninetysix_words <- get_debate_special_words(ninetysix,candidates[[6]][[1]],candidates[[6]][[2]],moderato
print(ninetysix_data)
```

```
## [1] 45
```

```
average_word_length_08 <- c(eight_length$republican_avg_word,eight_length$democrat_avg_word,eight_lengt
barplot(average_word_length_08, main="Average Word Length 2008", names.arg=c("republican", "democrat",
```




```
average_word_list_16 <- c(sixteen_length$republican_avg_word,sixteen_length$democrat_avg_word,sixteen_l
barplot(average_word_list_16, main="Average Word Length 2016", names.arg=c("republican", "democrat", "m
```



Problem3 Making this code in an oop fashion would have had a positive effect on the structure of the code. To transform my code into an oop approcha I would first make one class for each of the debates. The classes would have three different functions included inside of the class. 1) obtain data. The first function of the class would be to collect the information related to the data into a format that could be accessed by the two functions that would do the cleanup and analysis. 2) Parsde data. This function (that is still part of the class) would parse out the data and create a data frame containing the first section of the analysis which is the average lenght of the words per candidate. This class will return a dataframe with each candidate and the average length of the words. 3) Analysis. The last function of the class will call the data obtained at the first step (1) and generate some analysis to produce and to generate a vector with the number of ocurrances of a specific word. This would return a datafram with the required information.