ps3

Yihuan Song

9/20/2018

question 1

a) From the reading of Wilson's 'Best Practices for Scientific Computing', I learned that writing "human-readable" code is not only about writing clearly explained functions and comments, naming meaningfully and writing in consistent style, it is also about controlling the length of code so that it would not be too long to hold in human memory. Therefore, limiting length of code chunks and modularizing code is helpful when others are trying to understand and reproduce your code. What's more, I think the notion of "turning bugs to test cases" is really helpful, because it would be efficient to think of defensive programming in the process of writing functions and debugging. This would help with developing more complete and thorough test cases, and therefore a good practice to develop reproducible research and computing with robustness.

d) In terms of reproducibility, the strengths of the clm.pdf paper include:

```
1. Included clear file descriptions to help understand the purpose of each file
2. Use of the workflow diagram to clarify the procedure of the whole process
3. Use of different styles of comments to seperate chunks of code, making code more readable
4. Breaking code sources into seperate files with different purposes, so the process is modularized
5. Breaking up code to the next line at appropriate length, so all code written are easily visible
6. Included checks/assertions to see if file exists and if file is of the right length,
   providing extra robustness
```

And weaknesses include:

```
1. Used absolute paths while specifying file path,
   introduced difficulty for others to reproduce the analysis
2. Did not assert to see if the url exists, since url could be changed/moved
3. Did not specify version of packages used, since packaged could be modified after a period of time
4. No tests to check if functions work appropriately
5. Introduced ambiguities by using global variables in functions
6. Invoked packages inside the function, making it unclear for reproducing
```

question 2

(a) The "clean_format" function would clean the nonverbal words and formatting of the text(using sapply). The "splt_dbt" function would split the text according to the metadata for a specific speaker, strip out and put to a list for the chunk related to the speaker, and count the number of chunks and laughter and applause tags.

```
#load additional pakages required
library(ggplot2)
library(testthat)
library(knitr)
library(kableExtra)

#load r code prepared
source('~/Desktop/stat_243/stat243-fall-2018/ps/ps3prob2.R')
```

```
## [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"
##
## "\nOctober 3, 2000 Transcript\n\nOctober 3, 2000The First Gore-Bush Presidential DebateMODERATOR: Go
```

```
#clean different data format for the second debate in list
debates_body[2] <- str_replace_all(debates_body[2], "[\r\n]","")
#clean duplicated third debate in the list
debates_body[3] <- substr(debates_body[3], nchar(debates_body[3])/2, nchar(debates_body[3]))

moderators <- c("HOLT", "LEHRER", "LEHRER", "LEHRER", "MODERATOR","LEHRER")
years <- rev(c(yrs,2016))

#function cleans up the nonverbal texts and formatting and returns the cleaned list
clean_format <- function(list){

  clean_list <- sapply(list, str_replace_all, "\\(.*?\\)", "", USE.NAMES = FALSE)
  clean_list <- sapply(clean_list, str_replace_all, "\\[.*?\\]", "", USE.NAMES = FALSE)
  clean_list <- sapply(clean_list, str_replace_all, "[\r\n]", "", USE.NAMES = FALSE)

  return(clean_list)

}

#function will split each debate by speakers's name, put the chunks into lists,
#and print the number of chunks and nonverbal tags
splt_dbt <- function(old_str,num){
  #look for speakers in a specific debate
  dem_cand <- unlist(candidates[num],use.names = FALSE)[1]
  rep_cand <- unlist(candidates[num],use.names = FALSE)[2]
  mod <- moderators[num]

  #split the texts by speakers' NAME
  match <- str_c(c(dem_cand,rep_cand,mod), collapse = "|")
  new_str <- unlist(strsplit(old_str[num], paste0("(?<=.)(?=", match, ")"),
                             perl = TRUE), use.names = FALSE)

  #put each speaker's chunks together into a list and print out number of chunks and nonverbal tags
  dem_chunks <- grep(paste0(dem_cand, ": "), new_str, value = TRUE)
  cat("\nThe number of chunks for", dem_cand,"is", length(dem_chunks),"\n")
  cat(dem_cand, "got", length(grep("applause", dem_chunks, ignore.case = TRUE)),"applause\n")
  cat(dem_cand, "got", length(grep("LAUGHTER", dem_chunks, ignore.case = TRUE)),"laughter\n")
  dem_chunks <- clean_format(dem_chunks)

  rep_chunks <- grep(paste0(rep_cand, ": "),new_str, value = TRUE)
  cat("\nThe number of chunks for", rep_cand,"is", length(rep_chunks),"\n")
  cat(rep_cand, "got", length(grep("applause", rep_chunks, ignore.case = TRUE)),"applause\n")
  cat(rep_cand, "got", length(grep("LAUGHTER", rep_chunks, ignore.case = TRUE)),"laughter\n")
  rep_chunks <- clean_format(rep_chunks)
```

```
  mod_chunks <- grep(paste0(mod, ": "),new_str, value = TRUE)
  cat("\nThe number of chunks for", mod,"is", length(mod_chunks),"\n")
  cat(mod, "got", length(grep("applause", mod_chunks, ignore.case = TRUE)),"applause\n")
  cat(mod, "got", length(grep("LAUGHTER", mod_chunks, ignore.case = TRUE)),"laughter\n")
  mod_chunks <- clean_format(mod_chunks)

  new_list <- list(dem_chunks, rep_chunks, mod_chunks)

  return(new_list)
}

#loop through each debate
chunk_lists<-list()

for(num in 1:length(debates_body)){
  cat("\nHere is some information on debate", years[num], ":")
  chunk_lists[[num]] <- splt_dbt(debates_body, num)
}
```

```
##
## Here is some information on debate 2016 :
## The number of chunks for CLINTON is 87
## CLINTON got 4 applause
## CLINTON got 4 laughter
##
## The number of chunks for TRUMP is 124
## TRUMP got 5 applause
## TRUMP got 1 laughter
##
## The number of chunks for HOLT is 97
## HOLT got 3 applause
## HOLT got 0 laughter
##
## Here is some information on debate 2012 :
## The number of chunks for OBAMA is 56
## OBAMA got 0 applause
## OBAMA got 3 laughter
##
## The number of chunks for ROMNEY is 71
## ROMNEY got 0 applause
## ROMNEY got 1 laughter
##
## The number of chunks for LEHRER is 83
## LEHRER got 1 applause
## LEHRER got 0 laughter
##
## Here is some information on debate 2008 :
## The number of chunks for OBAMA is 63
## OBAMA got 0 applause
## OBAMA got 0 laughter
##
## The number of chunks for MCCAIN is 63
## MCCAIN got 0 applause
## MCCAIN got 1 laughter
```

```
##
## The number of chunks for LEHRER is 63
## LEHRER got 2 applause
## LEHRER got 1 laughter
##
## Here is some information on debate 2004 :
## The number of chunks for KERRY is 33
## KERRY got 0 applause
## KERRY got 2 laughter
##
## The number of chunks for BUSH is 41
## BUSH got 0 applause
## BUSH got 1 laughter
##
## The number of chunks for LEHRER is 68
## LEHRER got 2 applause
## LEHRER got 0 laughter
##
## Here is some information on debate 2000 :
## The number of chunks for GORE is 49
## GORE got 0 applause
## GORE got 0 laughter
##
## The number of chunks for BUSH is 56
## BUSH got 0 applause
## BUSH got 0 laughter
##
## The number of chunks for MODERATOR is 61
## MODERATOR got 2 applause
## MODERATOR got 0 laughter
##
## Here is some information on debate 1996 :
## The number of chunks for CLINTON is 45
## CLINTON got 0 applause
## CLINTON got 0 laughter
##
## The number of chunks for DOLE is 46
## DOLE got 0 applause
## DOLE got 0 laughter
##
## The number of chunks for LEHRER is 53
## LEHRER got 0 applause
## LEHRER got 0 laughter
```

(b) For each speaker in each debate, use "strsplit" and regular expressions to seperate each chunk by sentences and words

```
debate_words <- list()
debate_sentences <- list()
name_pattern <- str_c(c(unique(moderators), unique(unlist(candidates))), collapse = "|")

#loop through debates
for (list in 1:length(chunk_lists)){
  tmp_word <- list()
  tmp_sent <- list()
```

```
  #loop through speakers
  for(num in 1:length(chunk_lists[[list]])){

    #extract by sentence
    prd <- "(Mr|Ms|Dr|Mrs)\\."
    tmp_prd <- unlist(gsub(prd, "\\1#", chunk_lists[[list]][[num]]))
    tmp_prd <- gsub("U.S.","U#S#",tmp_prd)
    tmp_sent[[num]] <- unlist(strsplit(tmp_prd, "(?<=[\\.|\\?|\\!])\\s?(?=[A-Z])", perl = T))

    #extract by words
    punct_rm <- gsub("[[:punct:]]", "",tmp_sent[[num]], perl = TRUE)
    name_rm <- gsub(name_pattern, "", punct_rm)
    tmp_word[[num]] <- unlist(str_split(name_rm, " "))
    tmp_word[[num]] <- tmp_word[[num]][tmp_word[[num]] != ""]

  }

  debate_words[[list]] <- tmp_word
  debate_sentences[[list]] <- tmp_sent

}

debate_words[[1]][[1]][1:10]
```

```
##  [1] "How"    "are"    "you"    "Donald" "Well"   "thank"  "you"
##  [8] "Lester" "and"    "thanks"
```

```
debate_sentences[[1]][[1]][1:10]
```

```
##  [1] "CLINTON: How are you, Donald? "
##  [2] "CLINTON: Well, thank you, Lester, and thanks to Hofstra for hosting us."
##  [3] "The central question in this election is really what kind of country we want to be and what ki
##  [4] "Today is my granddaughter's second birthday, so I think about this a lot."
##  [5] "First, we have to build an economy that works for everyone, not just those at the top."
##  [6] "That means we need new jobs, good jobs, with rising incomes."
##  [7] "I want us to invest in you."
##  [8] "I want us to invest in your future."
##  [9] "That means jobs in infrastructure, in advanced manufacturing, innovation and technology, clean
## [10] "We also have to make the economy fairer."
```

(c) For each speaker in each chunk, the function "get_wordlen" counts the number of words and characters, calculates the average word length, and put all information in to a list. After that, a data frame is created to store the information. According to the table, we can see that the average word length for each candidates are approximately the same, and the word length of speakers in each debate is also of little difference for each year.

```
#the function takes one debate of out of 6 debates, counts word and word length,
#and make a list of all the information correspondingly
get_wordlen <- function(list ,num){
  dem_cand <- unlist(candidates[num],use.names = FALSE)[1]
  rep_cand <- unlist(candidates[num],use.names = FALSE)[2]
  names <- c(dem_cand, rep_cand)
  word_len <- list()
  char_len <- list()
  avg_len <- list()
```

```
  info <- list()
  for(x in 1:2){
    word_len[[x]] <- length(list[[num]][[x]])
    char_len[[x]] <- sum(sapply(debate_words[[num]][[x]],nchar))
    avg_len[[x]] <- char_len[[x]]/word_len[[x]]
    info[[x]] <- c(years[num], names[[x]], word_len[[x]], char_len[[x]], avg_len[[x]])
  }
  return(info)

}

#loop through all the debates and put all information into a list
count_list <- list()

for (x in 1:length(debate_words)){

  y <- 2*x-1
  count_list[[y]] <- get_wordlen(debate_words, x)[[1]]
  count_list[[y+1]] <- get_wordlen(debate_words, x)[[2]]

}

#make a data frame using the list containing all information
df<-as.data.frame(count_list, row.names = c("year", "candidate", "words", "characters", "average"),
                  col.names = 1:12)
count_df <- t(df)
count_df
```

```
##     year   candidate words  characters average
## X1  "2016" "CLINTON" "6314" "27320"    "4.32689261957555"
## X2  "2016" "TRUMP"   "8453" "35744"    "4.22855790843488"
## X3  "2012" "OBAMA"   "7270" "32212"    "4.43081155433287"
## X4  "2012" "ROMNEY"  "7768" "33435"    "4.30419670442842"
## X5  "2008" "OBAMA"   "7604" "33119"    "4.35547080483956"
## X6  "2008" "MCCAIN"  "7117" "31350"    "4.40494590417311"
## X7  "2004" "KERRY"   "7104" "30359"    "4.27350788288288"
## X8  "2004" "BUSH"    "6319" "27163"    "4.2986231998734"
## X9  "2000" "GORE"    "7195" "31270"    "4.34607366226546"
## X10 "2000" "BUSH"    "7437" "31967"    "4.29837299986554"
## X11 "1996" "CLINTON" "7654" "33371"    "4.35994251371832"
## X12 "1996" "DOLE"    "8103" "34627"    "4.27335554732815"
```

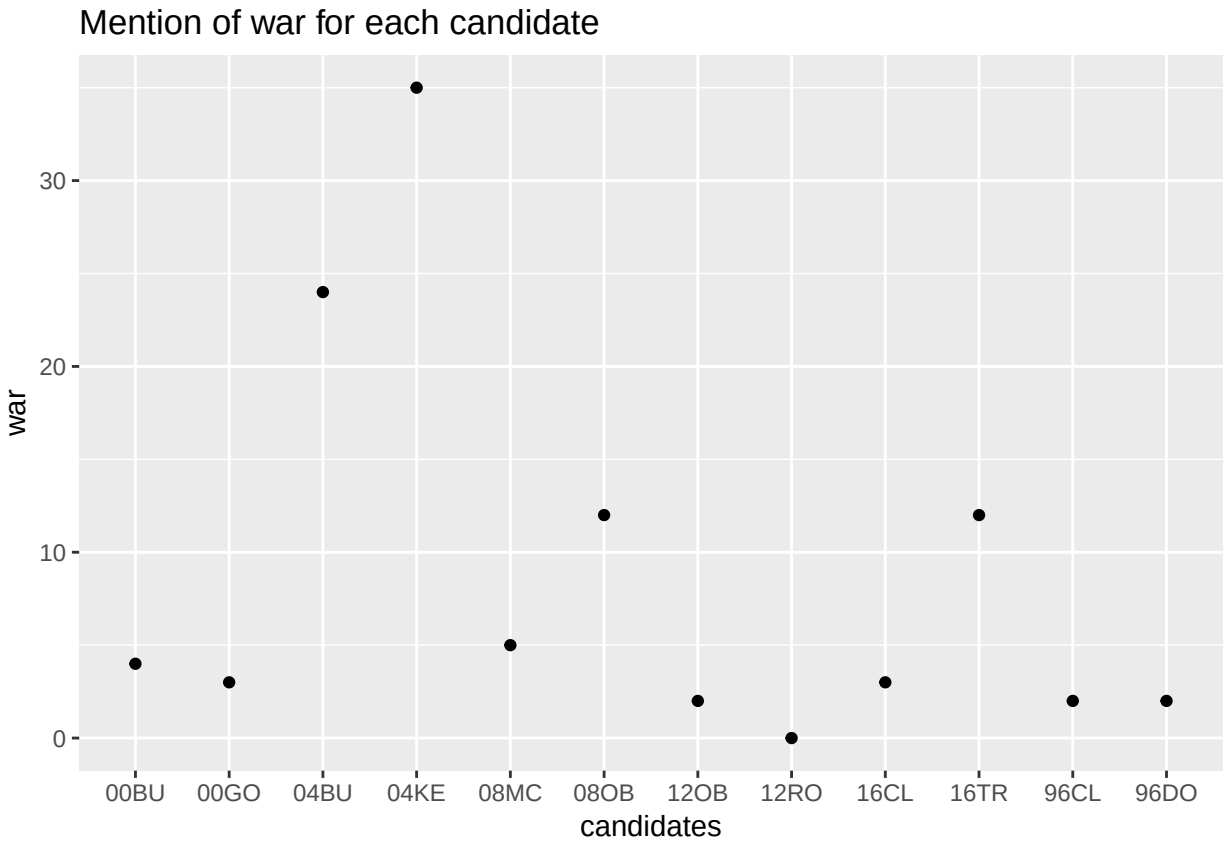
(d) For each candidate in each debate, the function "ct_words" mainly uses "grep" to extract and counts the number of appearance for "I, we, America{,n}, democra{cy,tic}, republic, Democrat{,ic}, Republican, free{,dom}, war, God [not including God bless], God Bless, {Jesus, Christ, Christian}". The information us stored in a data frame. Plots are made to explore the pattern. According to the table and the plot, we can see that the word choice is quite different for each debate in different years, and the word usage and frequence of mentioning different terms are different for each candidate, but the similarity is the high frequency for "I" and "we". For example, "war" and "freedom" is more frequent in year 2004 than other years, and "Republican" is mentioned more frequently in year 1996.

```
#function counts specified words for each speaker in a specified debate,
#return the information by a list
ct_words <- function(lists, num){
  dem_cand <- unlist(candidates[num],use.names = FALSE)[1]
```

```
  rep_cand <- unlist(candidates[num],use.names = FALSE)[2]
  names <- c(dem_cand, rep_cand)
  info <- list()
  for(x in 1:2){
     num_I <- length(grep("^I$",lists[[num]][[x]]))
     num_we <-length(grep("^we$",lists[[num]][[x]]))
     num_Ame <- length(grep("America$|American$",lists[[num]][[x]]))
     num_dem <- length(grep("democracy|democratic}",lists[[num]][[x]]))
     num_rep <- length(grep("republic",lists[[num]][[x]]))
     num_D <- length(grep("Democrat$|Democratic",lists[[num]][[x]]))
     num_R <- length(grep("Republican",lists[[num]][[x]]))
     num_free <- length(grep("^free$|freedom",lists[[num]][[x]]))
     num_war <- length(grep("^war$",lists[[num]][[x]]))
     num_bless <- length(grep("God bless",debate_sentences[[num]][[x]]))
     num_god <- length(grep("God",lists[[num]][[x]])) - num_bless
     num_jchr <- length(grep("Jesus|Christ|Christian",lists[[num]][[x]]))

    info[[x]] <- c(years[num], names[[x]],num_I, num_we, num_Ame, num_dem, num_rep,
                   num_D, num_R, num_free, num_war, num_god, num_bless, num_jchr)
  }
  return(info)

}

#loop through all the debates and stores information into a list
search_words <- list()

for (x in 1:length(debate_words)){

  y <- 2*x-1
  search_words[[y]] <- ct_words(debate_words, x)[[1]]
  search_words[[y+1]] <- ct_words(debate_words, x)[[2]]

}

#turn the list to a dataframe
df <- as.data.frame(search_words, col.names = 1:12, stringsAsFactors = FALSE)
word_table <- as.data.frame(t(df), stringsAsFactors = FALSE)
word_table_char <- word_table[1:2]
word_table_num <- word_table[3:14]
word_table_num <- sapply(word_table_num, as.numeric)
word_table <- cbind(word_table_char, word_table_num)
colnames(word_table) = c("year", "candidate", "num_I", "num_we", "America{,n}", "democra{cy,tic}",
                                "republic", "Democrat{,ic}", "Republican", "free{,dom}", "war", "God",
                                "God Bless", "Jesus, Christ, Christian")
kable(word_table) %>%  kable_styling(latex_options="scale_down")
```

	year	candidate	num_I	num_we	America{,n}	democra{cy,tic}	republic	Democrat{,ic}	Republican	free{,dom}	war	God	God Bless	Jesus, Christ, Christian
X1	2016	CLINTON	133	105	17	1	0	2	2	0	3	0	0	0
X2	2016	TRUMP	231	80	7	0	0	1	1	0	12	0	0	0
X3	2012	OBAMA	93	110	18	0	0	4	9	3	2	0	0	0
X4	2012	ROMNEY	149	71	30	1	0	4	9	7	0	0	0	0
X5	2008	OBAMA	118	171	12	0	0	0	3	2	12	0	0	0
X6	2008	MCCAIN	168	115	18	1	0	1	7	3	5	0	0	2
X7	2004	KERRY	144	92	40	2	0	0	1	3	35	0	1	0
X8	2004	BUSH	150	89	23	4	0	0	0	35	24	1	0	0
X9	2000	GORE	195	61	12	1	0	1	2	1	3	0	0	0
X10	2000	BUSH	172	64	18	1	0	2	9	3	4	0	0	0
X11	1996	CLINTON	210	81	32	3	0	1	10	4	2	0	0	0
X12	1996	DOLE	214	78	41	0	0	7	12	1	2	0	1	0

```
#plot the number of mentioning of "war" by each candidate
yr_cand <- paste0(sapply(word_table$year, substr, 3, 4),sapply(word_table$candidate, substr, 1, 2))
word_table <- cbind(word_table, yr_cand)
war_plot = ggplot(data = word_table, aes(y = word_table$war, x = word_table$yr_cand)) + geom_point()
war_plot = war_plot + ggtitle("Mention of war for each candidate")
war_plot = war_plot + xlab("candidates") + ylab("war")
war_plot
```



(e) The unit tests for the functions are created using the "testthat" package. Tests are created to make sure that each of the functions returns reasonable results, including the type/length of the returned object.

```
test_that("clean_format returns a list", {
  expect_is(clean_format(chunk_lists[[1]]), "list")
})

test_that("splt_dbt returns a list of 3", {

  expect_true(length(splt_dbt(debates_body,1)) == 3 )
```

```
  expect_is(splt_dbt(debates_body,1), "list")
})
```

```
##
## The number of chunks for CLINTON is 87
## CLINTON got 4 applause
## CLINTON got 4 laughter
##
## The number of chunks for TRUMP is 124
## TRUMP got 5 applause
## TRUMP got 1 laughter
##
## The number of chunks for HOLT is 97
## HOLT got 3 applause
## HOLT got 0 laughter
##
## The number of chunks for CLINTON is 87
## CLINTON got 4 applause
## CLINTON got 4 laughter
##
## The number of chunks for TRUMP is 124
## TRUMP got 5 applause
## TRUMP got 1 laughter
##
## The number of chunks for HOLT is 97
## HOLT got 3 applause
## HOLT got 0 laughter
```

```
test_that("get_wordlen returns a list of 2", {

  expect_true(length(get_wordlen(debate_words,1)) == 2 )
  expect_is(get_wordlen(debate_words,1), "list")
})

test_that("ct_words returns a non-empty list", {

  expect_true(length(ct_words(debate_words,1)) != 0 )
  expect_is(ct_words(debate_words,1), "list")
})
```

quwstion 3

We can redesign the code from question 2 to use object-oriented programming, we could create classes as follows:

class: debate

```
*fields:
    -year              // the year index for each debate
    -candidates        // the candidates for each debate
    -moderators        // the moderator for each debate
    -debateChunk       // the spoken chunks for each debate
*methods:
    -splt_dbt         // takes the debate and splits debate by spoken chunks
```

class: debateChunk

```
*fields:
    -metadata          // the metadata in the chunk for each candidate
    -words             // seperated debate_chunk by word
    -sentences         // seperated debate_chunk by sentences
*methods:
    -splt_word        // uses all fields from the debate class, and seperates each chunk by word
    -splt_sentence    // uses fields from the debate class, and seperates each chunk by sentence
```

class: wordInfo

```
*fields:
    -wordLength        // the average word length for each candidate
    -numMentioned      // the number of time one word is mentioned
*methods:
    -get_wordlen       // uses year and candidates fields from the debate class and words field from
                          debateChunk class, and calculate average word length for each candidate
    -ct_words          // uses year and candidates fields from the debate class and words field
                          from debateChunk class, and counts the number of time one candidate
                          speaks a specified word
```