ps3

Baoyue Liang

9/26/2018

Problemm 1

(d)

Strength: 1. Have intermedia data, which allow others to start halfway. 2. Use graph to vasualize result, which is very interpretive. 3. Separate code with different function into different parts. 4. Use Github to do version control. 5. Nice indentation to make code readable.

Weakness: 1. Hard code the file path in windows system. 2. Too much code in each chunk. It may be better if those chunks are separated into smaller parts.

Problem 2

```
moderators <- c("LEHRER", "LEHRER", "LEHRER", "MODERATOR", "LEHRER", "HOLT")

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"))


library(XML)
library(stringr)
library(assertthat)

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[5], 1, 1000))
```

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

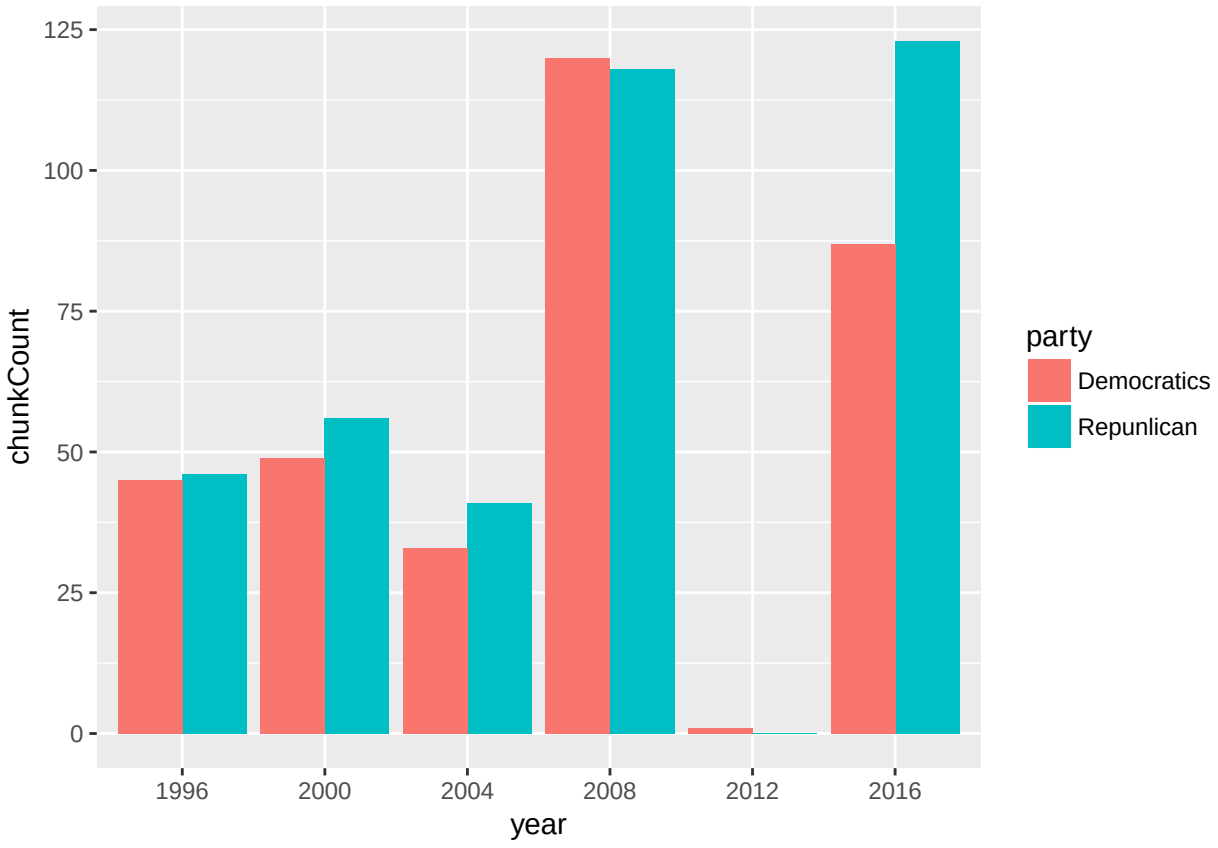
(a)

```
library(stringr)
cddSpeechD = cddSpeechR = mdrSpeech = list()
countDL = countDA = countDC = countRL = countRA = countRC = chunkD = chunkR =  matrix()

for (i in 1:6){
  # finding anything start with "moderator name :" and the next : to seperate the speech of moderators
  pattern1 = paste0(moderators[i],"\\:\\s.*?\\:")
  mdrSpeech[[i]] = t(str_extract_all(debates_body[7-i], pattern1, simplify = TRUE))

  pattern2 = paste0(candidates[[7-i]][[1]],"\\:\\s.*?\\:")

  pattern3 = paste0(candidates[[7-i]][[2]],"\\:\\s.*?\\:")

  cddSpeechD[[i]]= t(str_extract_all(debates_body[7-i], pattern2 , simplify = TRUE))
  cddSpeechR[[i]]= t(str_extract_all(debates_body[7-i], pattern3 , simplify = TRUE))
```

```
   countDL[i] = sum(str_count(cddSpeechD[[i]], "laughter(?i)"))
   countDA[i] = sum(str_count(cddSpeechD[[i]], "applause(?i)"))
   countDC[i] = sum(str_count(cddSpeechD[[i]], "crosstalk(?i)"))
   countRL[i] = sum(str_count(cddSpeechR[[i]], "laughter(?i)"))
   countRA[i] = sum(str_count(cddSpeechR[[i]], "applause(?i)"))
   countRC[i] = sum(str_count(cddSpeechR[[i]], "crosstalk(?i)"))
# remove the extra part that belongs to the start of next chunk
   patternD1 = paste0(candidates[[7-i]][[1]], "\\:")
   patternD2 = paste0(candidates[[7-i]][[2]], "\\:")
   patternD3 = paste0(moderators[i], "\\:")

   mdrSpeech[[i]] = str_remove(mdrSpeech[[i]], patternD1)
   mdrSpeech[[i]] = str_remove(mdrSpeech[[i]], patternD2)
   mdrSpeech[[i]] = str_remove(mdrSpeech[[i]], patternD3)

   cddSpeechD[[i]] = str_remove(cddSpeechD[[i]], patternD1)
   cddSpeechD[[i]] = str_remove(cddSpeechD[[i]], patternD2)
   cddSpeechD[[i]] = str_remove(cddSpeechD[[i]], patternD3)

   cddSpeechR[[i]] = str_remove(cddSpeechR[[i]], patternD1)
   cddSpeechR[[i]] = str_remove(cddSpeechR[[i]], patternD2)
   cddSpeechR[[i]] = str_remove(cddSpeechR[[i]], patternD3)

   chunkD[i] = length(cddSpeechD[[i]])
   chunkR[i] = length(cddSpeechR[[i]])

   # remove all the non-spoken text in the square bracket
   cddSpeechD[[i]] = str_replace_all(cddSpeechD[[i]],"\\[(.*?)\\]","")
   cddSpeechR[[i]] = str_replace_all(cddSpeechD[[i]],"\\[(.*?)\\]","")

  cddSpeechD[[i]] = str_replace_all(cddSpeechD[[i]],"\\((.*?)\\)","")
   cddSpeechR[[i]] = str_replace_all(cddSpeechD[[i]],"\\((.*?)\\)","")
}

# remove all the non-spoken text in the square bracket
cddSpeechD = str_replace_all(cddSpeechD,"\\[(.*?)\\]","")
```

```
## Warning in stri_replace_all_regex(string, pattern,
## fix_replacement(replacement), : argument is not an atomic vector; coercing
```

```
cddSpeechR = str_replace_all(cddSpeechD,"\\[(.*?)\\]","")

chunkCount = as.vector(cbind(t(chunkD),t(chunkR)))
year = c("1996","2000","2004","2008","2012","2016", "1996","2000","2004","2008","2012","2016")
party = c("Democratics", "Democratics","Democratics","Democratics","Democratics","Democratics","Repunli

plotChunk = data.frame(chunkCount,year,party)
library(ggplot2)
ggplot(plotChunk,aes(x = year,y = chunkCount,fill = party))+geom_bar(position="dodge",stat="identity")
```



(b) & (c)

```
cddWordD = cddWordR = avgWdLD = avgWdLR =  list()

for(i in 1:6){
  #split into words
  cddWordD[[i]] = str_split(cddSpeechD[[i]], " ")
  cddWordD[[i]] = unlist(cddWordD[[i]])

  cddWordR[[i]] = str_split(cddSpeechR[[i]], " ")
  cddWordR[[i]] = unlist(cddWordR[[i]])

  #delete punctuation
  cddWordD[[i]] = gsub("[^(\\w)]$","",cddWordD[[i]], perl = T)
  cddWordR[[i]] = gsub("[^(\\w)]$","",cddWordR[[i]], perl = T)

  #calculate average word length
  avgWdLD[i] = sum(str_count(cddWordD[i]))/lengths(cddWordD[i])
  avgWdLR[i] = sum(str_count(cddWordR[i]))/lengths(cddWordR[i])
}
```

```
## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
```

```
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing

## Warning in stri_count_boundaries(string, opts_brkiter = opts(pattern)):
## argument is not an atomic vector; coercing
```

Comments: Average word length does not vary much among candidates or over year.

(d)

```
specialWord= c( "^I$", "^America(n)?$","^democra((cy) | (tic))?$","republic","Democrat(ic)?", "Republica

countSpcWordR = countSpcWordD =  matrix()


for(j in 1:6){
  for(i in 1:length(specialWord)){
    countSpcWordD[i,j] = length(grep(specialWord[i],cddWordD[j]))
    countSpcWordR[i,j] = length(grep(specialWord[i],cddWordR[j]))
  }
  countSpcWordD[(length(specialWord)+1),j] = length(grep("God Bless",cddSpeechD[j]))
  countSpcWordR[(length(specialWord)+1),j] = length(grep("God Bless",cddSpeechR[j]))
}
```

Comments: the candidates who use more I than we may be more self centered and who use more “war”" may be more aggressive.

Problem 3

Debate

- field: year
- field: candidates(a list of Candidate)
- field: text
- method: getDebateFromYear(int year)
- method: getChunksForCandidate(Candidate candidate)
- method: countNumberOfWordsForCandidate(Candidate candidate)
- method: countNumberOfCharactersForCandidate(Candidate candidate)
- method: ComputeWordLengthForCandidate(Candidate candidate)
  - use countNumberOfCharactersForCandidate(Candidate candidate) countNumberOfWordsForCandidate(Candidate candidate)

Candidate

- field: name
- field: years(a list of int)
- method: getCandidateFromName(string name)
- method: getNumberOfWords(String word)