Stats243 Problem Set 3

Mengling Liu

Sept 2017

Collaboration: I disscussed with Rui Chen for this homework set.

2.(a) Extract the plays into a character vector or a list, with one element for each play. Skip the information at the start of the file, the first piece (the sonnets), and the last piece (Lover's Complaint).

```
library(curl)
filename <- "http://www.gutenberg.org/cache/epub/100/pg100.txt"
# Read each line of the text file into a vector
plays <- readLines(filename)
# Note that each play starts with a year and ends with "THE END"
# Use regular expression to locate the start row and the
# end row of each play
start <- grep('^1[[:digit:]]{3}$', plays)
end <- grep('THE END', plays)
# Rows in between the start row and end row will be the
# play content
# Extract each play into a vector and compile all the plays
# into a list
# Also will take out the first and the last piece
compileplays <- list()
for (i in 1:(length(start)-2)) {
   compileplays[[i]] <- plays[start[i+1]:end[i+1]]
}
```

(b) Extract meta data about each play and extract the body of the play. The result should be in the form of an R object, with one element for each play. By meta data I mean the year of the play, the title, the number of acts, and the number of scenes.

```
library(stringr)

# Use funcion as a constructor to create a class called
# playinfo which contains fields such as year, title,
# actsnum, scenesum, and body
# Pls note that the compiletext is a list created in
```

```
# part c which will return the speaker name and spoken
# chunks of each play
# Will add the compiletext as a new field into the class

playinfo <- function(year = NA, title = NA, actsnum = NA,
                     scenesnum = NA, body = NA,
                     compiletext = NA){
     obj <- list(year = year, title = title, actsnum =
     actsnum, scenesnum = scenesnum, body = body,
     compiletext = compiletext)
     class(obj) <- 'playinfo'
     return(obj)
}

# I created the fields for the first play first
# For the 1st play, need to delete the empty lines first
empty_lines <- grepl('^\\s*$', compileplays[[1]])
compileplays[[1]] <- compileplays[[1]][! empty_lines]
# Year is the first line of each play list
year1 <- compileplays[[1]][1]
# Title is the second line of each play list
title1 <- compileplays[[1]][2]
# compileplays[[1]]
# Use regular expression to locate ACT and count
# the number of Act
acts1 <- str_extract(compileplays[[1]],
                     'A(?i)(CT)\\s([MDCLXVIl]+|[0-9]+)')
acts1 <- acts1[! is.na(acts1)]
acts1 <- unique(acts1)
actsnum1 <- length(acts1)
# Use regular expression to locate Scene and count
# the number of Scene
scenes1 <- str_extract(compileplays[[1]],
                       'S(?i)(CENE)\\s([MDCLXVI]+|[0-9]+)')
scenes1 <- scenes1[! is.na(scenes1)]
scenesnum1 <- length(scenes1)
# Extract the body of the play
bodystart <- grep('S(?i)(CENE)', compileplays[[1]])
body1 <- compileplays[[1]][(bodystart[1]):
                           (length(compileplays[[1]])-1)]
# Insert the info for each fields into the class
# for the first play
playlist1 <- playinfo(year=year1, title=title1,
actsnum=actsnum1, scenesnum=scenesnum1, body=body1,
compiletext=NA )
```

```
# Create a list called playlist which stores the info
# of each play as an element under the list
# Loop over 36 plays to extract year, title, number
# of acts, number of scenes, body ad compiletext
playlist <- list()
for (i in 1:36){
  empty_lines <- grepl('^\\s*$', compileplays[[i]])
  compileplays[[i]] <- compileplays[[i]][! empty_lines]
  year <- compileplays[[i]][1]
  title <- compileplays[[i]][2]
  acts <- str_extract(compileplays[[i]],
                  'A(?i)(CT)\\s([MDCLXVI1]+|[0-9]+)')
  acts <- acts[! is.na(acts)]
  acts <- unique(acts)
  actsnum <- length(acts)
  scenes <- str_extract(compileplays[[i]],
                 'S(?i)(CENE)\\s([MDCLXVI]+|[0-9]+)')
  scenes <- scenes[! is.na(scenes)]
  scenesnum <- length(scenes)
  bodystart <- grep('S(?i)CENE', compileplays[[i]])
  body <- compileplays[[i]][(bodystart[1]):
                          (length(compileplays[[i]])-1)]
  playlist[[i]] <- playinfo(year=year, title=title,
    actsnum=actsnum, scenesnum=scenesnum, body=body,
    compiletext=NA)
}
```

(c) Extract the actual text spoken by the characters into your object. You should store each chunk of spoken text and store the speaker of each chunk. Discard stage directions, Dramatis personae information, and scene information. Note that the stage directions come in many forms, with inconsistent formatting, so we're not expecting perfection here - you'll likely accidentally include a small number of stage directions as part of your spoken text.

```
library(stringr)

# For the 1st play, extract the speaker name from the play body
# Then find the cooresponding rows forming the actual text spoken
# under each speaker
# Indentation of 2 indicates the speaker name
# Indentatio of 4 indicates the rest of the spoken words
bodystart <- grep('S(?i)CENE', compileplays[[1]])
body <- compileplays[[1]][(bodystart[1]):
                            (length(compileplays[[1]])-1)]
speaker <- str_extract(body, '^(\\s){2}[A-Z]+')
```

```
speaker <- speaker[!is.na(speaker)]
# Use regular expression to locate the row number of each speaker
speakertrue <- which(str_detect(body, '^(\\s){2}[A-Z]+'))
# Create a text list, and insert the actual text under each speaker
# into the list
text <- list()
for (i in 1:(length(speakertrue)-1)){
  text[[i]] <- body[(speakertrue[i]):(speakertrue[i+1]-1)]
  text[[length(speakertrue)]] <- body[speakertrue
                        [length(speakertrue)]:length(body)]
}


# Loop over all 36 plays to extract speaker name and actual text
# compiletext here is a list of list that the first layer contains
# 36 plays and second layer contains the speaker and actual text
# under that play

# Note that the 4th play, there is no space in front of speaker name
# For spoken words, still 4 space of indentation
# Below used if statment to change the regex to capture that

# Note that the 17th play, there is 5 indenation for spoken words
# But indentation for speaker name remains 2
# Below used if statment to change the regex to capture spoken words

compiletext <- list()
for (j in 1:36){
# Use the body from part b
empty_lines <- grepl('^\\s*$', compileplays[[j]])
compileplays[[j]] <- compileplays[[j]][! empty_lines]
bodystart <- grep('S(?i)CENE', compileplays[[j]])
body <- compileplays[[j]][(bodystart[1]):(length(compileplays[[j]])-1)]
# Use if statement to locate the row number of speaker name
# and actual words spoken body for different scenarios
if (! j == 4 && !j == 17 ) {
  nametext<-grepl('(^\\s{2}[A-Za-z]+)|(^\\s{4}[A-Za-z]+)',body)
  body <- body[nametext]
  speakertrue <- which(str_detect(body, '^(\\s){2}[A-Z]+'))
} else if( j==4 ) {
  stage <- grepl('^<[A-Za-z]+', body)
  body <-body[! stage]
  speakertrue <- which(str_detect(body, '^(\\s)*[A-Z(\\s)]+\\.'))
} else if (j==17){
  nametext<-grepl('(^\\s{2}[A-Za-z]+)|(^\\s{5}[A-Za-z]+)',body)
```

```
  body <- body[nametext]
  speakertrue <- which(str_detect(body, '^(\\s){2}[A-Z]+'))
}
# Combine the lines that falls under the speaker together
text <- list()
for (i in 1:(length(speakertrue)-1)){
  text[[i]] <- body[(speakertrue[i]):(speakertrue[i+1]-1)]
  text[[length(speakertrue)]] <- body[speakertrue[length
                                      (speakertrue)]:length(body)]
}
# Split out the speaker name from the spoken words and save to
# speakername under subtext, and save the spoken words to content
# under subtext for each chunk
# Then save the subtext to a new list called newtext which compiles
# all the chunks for each play
subtext <- list(speakername=NA, content=NA)
newtext <- list()
# use if statement to capture speaker name different for the 4th
# play and the rest
if(!j==4){
for (a in 1:length(speakertrue)){
  pattern <- str_extract(text[[a]],
                '^(\\s){2}(([A-Z(\\s)]+)|([A-Z][a-z]+))\\.')
  subtext$speakername <- pattern[!is.na(pattern)]
  subtext$content <- text[[a]]
  newtext[[a]]<- subtext
}
}else{
  for (a in 1:length(speakertrue)){
  pattern <- str_extract(text[[a]],
                 '^(\\s)*(([A-Z(\\s)]+)|([A-Z][a-z]+))\\.')
  subtext$speakername <- pattern[!is.na(pattern)]
  subtext$content <- text[[a]]
  newtext[[a]]<- subtext
}
}
# Assign the newtext to compiletext which contains 36 plays
# as a list that contains list of all chunks for each play
compiletext[[j]] <- newtext
}
# compiletext[[17]]

## Insert this compiletext as a new field into the class object
## created in part b of the problem
## I copied the codes over again to make it clear
```

```
playlist <- list()
for (i in 1:36){
  empty_lines <- grepl('^\\s*$', compileplays[[i]])
  compileplays[[i]] <- compileplays[[i]][! empty_lines]
  year <- compileplays[[i]][1]
  title <- compileplays[[i]][2]
  acts <- str_extract(compileplays[[1]],
                      'A(?i)(CT)\\s([MDCLXVIl]+|[0-9]+)')
  acts <- acts[! is.na(acts)]
  acts <- unique(acts)
  actsnum <- length(acts)
  scenes <- str_extract(compileplays[[i]],
                      'S(?i)(CENE)\\s([MDCLXVI]+|[0-9]+)')
  scenes <- scenes[! is.na(scenes)]
  scenesnum <- length(scenes)
  bodystart <- grep('S(?i)CENE', compileplays[[i]])
  body <- compileplays[[i]][(bodystart[1]):(length
                                      (compileplays[[i]])-1)]

  ###### Insert the compiletext field here.
  empty_lines <- grepl('^\\s*$', compileplays[[i]])
  compileplays[[i]] <- compileplays[[i]][! empty_lines]
  bodystart <- grep('S(?i)CENE', compileplays[[j]])
  body <- compileplays[[i]][(bodystart[1]):(length
                                      (compileplays[[i]])-1)]
  if (! i == 4 && !i == 17 ) {
    nametext<-grepl('(^\\s{2}[A-Za-z]+)|(^\\s{4}[A-Za-z]+)',body)
    body <- body[nametext]
    speakertrue <- which(str_detect(body, '^(\\s){2}[A-Z]+'))
  } else if( i==4 ) {
    stage <- grepl('^<[A-Za-z]+', body)
    body <-body[! stage]
    speakertrue <- which(str_detect(body, '^(\\s)*[A-Z(\\s)]+\\.'))
  } else if (i==17){
    nametext<-grepl('(^\\s{2}[A-Za-z]+)|(^\\s{5}[A-Za-z]+)',body)
    body <- body[nametext]
    speakertrue <- which(str_detect(body, '^(\\s){2}[A-Z]+'))
  }
  text <- list()
  for (q in 1:(length(speakertrue)-1)){
    text[[q]] <- body[(speakertrue[q]):(speakertrue[q+1]-1)]
    text[[length(speakertrue)]] <- body[speakertrue
                                  [length(speakertrue)]:length(body)]
  }
  subtext <- list(speakername=NA, content=NA)
```

```
  newtext <- list()
  if(!i==4){
  for (a in 1:length(speakertrue)){
    pattern <- str_extract(text[[a]],
                           '^(\\s){2}(([A-Z(\\s)]+)|([A-Z][a-z]+))\\.')
    subtext$speakername <- pattern[!is.na(pattern)]
    subtext$content <- text[[a]]
    newtext[[a]]<- subtext
  }
  }else{
    for (a in 1:length(speakertrue)){
    pattern <- str_extract(text[[a]],
                           '^(\\s)*(([A-Z(\\s)]+)|([A-Z][a-z]+))\\.')
    subtext$speakername <- pattern[!is.na(pattern)]
    subtext$content <- text[[a]]
    newtext[[a]]<- subtext
  }
  }
  ######

  # assign newtext here to compiletext filed
  playlist[[i]] <- playinfo(year=year, title=title, actsnum=actsnum,
                  scenesnum=scenesnum, body=body, compiletext=newtext)
}

# Below gives the content of the first spoken word of the first play
playlist[[1]]$compiletext[[1]]$content

## [1] "  COUNTESS. In delivering my son from me, I bury a second husband."

# Below gives the 1000th speaker name and content of the spoken word
# of the 17th play
playlist[[17]]$compiletext[[1000]]

## $speakername
## [1] "  Edg."
##
## $content
##  [1] "  Edg. By nursing them, my lord. List a brief tale;"
##  [2] "     And when 'tis told, O that my heart would burst!"
##  [3] "     The bloody proclamation to escape"
##  [4] "     That follow'd me so near (O, our lives' sweetness!"
##  [5] "     That with the pain of death would hourly die"
##  [6] "     Rather than die at once!) taught me to shift"
##  [7] "     Into a madman's rags, t' assume a semblance"
##  [8] "     That very dogs disdain'd; and in this habit"
```

```
##  [9] "     Met I my father with his bleeding rings,"
## [10] "     Their precious stones new lost; became his guide,"
## [11] "     Led him, begg'd for him, sav'd him from despair;"
## [12] "     Never (O fault!) reveal'd myself unto him"
## [13] "     Until some half hour past, when I was arm'd,"
## [14] "     Not sure, though hoping of this good success,"
## [15] "     I ask'd his blessing, and from first to last"
## [16] "     Told him my pilgrimage. But his flaw'd heart"
## [17] "     Burst smilingly."
```

(d) Now use the constructed data object to calculate summary statistics about each play. These should include the following: i. The number of unique speakers. ii. The number of spoken chunks. iii. The number of sentences and words spoken and average number of words per chunk. iv. The number of unique words.

```
# i. the number of unique speakers
# For the first play, write a lapply function get all
# the speaker names, and then count the unique names
get_speakername <- function(x){
  return(x$speakername)
}
speaker <- lapply(compiletext[[1]],get_speakername)
speakeruni <- unique(speaker)
speakernum <- length(speakeruni)

# Looping over 36 plays to get the number of unique names
# for each play
speakernum <- rep(0,36)
for (i in 1:36) {
speaker <- lapply(compiletext[[i]],get_speakername)
speakeruni <- unique(speaker)
speakernum[i] <- length(speakeruni)
}
# speakernum

#-----------------------------------------------------------------
# ii.The number of spoken chunks.
# For the first play, write a lapply function get all
# the spoken chunks, and then count the number of chunks
get_chunks <- function(y){
  return(y$content)
}
chunks <- lapply(compiletext[[1]],get_chunks)
chunksnum <- length(chunks)
```

```
# Looping over 36 plays to get the number of spoken chunks
# for each play
chunksnum <- rep(0,36)
for (i in 1:36) {
chunks <- lapply(compiletext[[i]],get_chunks)
chunksnum[i] <- length(chunks)
}
# chunksnum

#-----------------------------------------------------------------
# iii. The number of sentences and words spoken and average
# number of words per chunk.

# Number of sentences for each play ---------------------------
# Get spoken chunks for the first play
get_chunks <- function(y){
  return(y$content)
}
chunks <- lapply(compiletext[[1]],get_chunks)
# count the sentence number for each chunk
sentence_count <- function(z){
  count1 <- sum(str_count(z,'[\\.\\!\\?]'))
  return(count1-1)
}
sentencecount <- lapply(chunks,sentence_count)
# Sum over the sentence number for each chunk to get the sentence
# number for the first play
sum_sentencecount <- sum(as.numeric(sentencecount))

# Loop over 36 plays to get the number of sentences for each play
sum_sentencecount <- rep(0,36)
for (i in 1:36){
chunks <- lapply(compiletext[[i]],get_chunks)
sentencecount <- lapply(chunks,sentence_count)
sum_sentencecount[i] <- sum(as.numeric(sentencecount))
}
# sum_sentencecount

# Number of words spoken for each play ------------------------
# Get speaker name and spoken chunks for the first play
chunks <- lapply(compiletext[[1]],get_chunks)
speaker <- lapply(compiletext[[1]],get_speakername)
# count the words spoken for each chunk
word_count <- function(w,x){
  count2 <- sum(str_count(w,"[A-Za-z\\']+"))
```

```
  count3 <- sum(str_count(x,"[A-Za-z\\']+"))
  return(count2-count3)
}
wordcount <- mapply(word_count, chunks,speaker)
# Sum over all the chunks to get the total word count
# for the first play
sum_wordcount <- sum(as.numeric(wordcount))

# loop over 36 plays to get the total word cout for
# all plays
sum_wordcount <- rep(0,36)
for (i in 1:36){
chunks <- lapply(compiletext[[i]],get_chunks)
speaker <- lapply(compiletext[[i]],get_speakername)
wordcount <- mapply(word_count, chunks, speaker)
sum_wordcount[i] <- sum(as.numeric(wordcount))
}
# sum_wordcount


# Average number of words per chunk ------------------------------
# Get speaker name and spoken chunks for the first play
chunks <- lapply(compiletext[[1]],get_chunks)
speaker <- lapply(compiletext[[1]],get_speakername)
# count the words spoken for each chunk
word_count <- function(w,x){
  count2 <- sum(str_count(w,"[A-Za-z\\']+"))
  count3 <- sum(str_count(x,"[A-Za-z\\']+"))
  return(count2-count3)
}
wordcount <- mapply(word_count, chunks,speaker)
# Take the average of all the chunks to get the average word count
# per chunk for the first play
ave_wordcount <- mean(as.numeric(wordcount))

# loop over 36 plays to get the average word cout per chunk
# for all plays
ave_wordcount <- rep(0,36)
for (i in 1:36){
chunks <- lapply(compiletext[[i]],get_chunks)
speaker <- lapply(compiletext[[i]],get_speakername)
wordcount <- mapply(word_count, chunks, speaker)
ave_wordcount[i] <- mean(as.numeric(wordcount))
}
# ave_wordcount
```

```
# Number of unique words per play ------------------------------
# Get spoken chunks for the first play
chunks <- lapply(compiletext[[1]],get_chunks)
# extract all the single words for each chunk
extract <- str_extract_all(chunks,"[A-Za-z\\']+")
# Unlist the list to gather all words together
# in order to find the unique words per play
uniqword <- length(unique(unlist(extract)))

# Loop over the 36 plays to obtain the unique words per play
uniqword <- rep(0,36)
for (j in 1:36){
chunks <- lapply(compiletext[[j]],get_chunks)
extract <- str_extract_all(chunks,"[A-Za-z\\']+")
uniqword[j] <- length(unique(unlist(extract)))
}
# uniqword
```

(e) Plot some of your summary statistics as a function of time to see if there are trends in Shakespeare's plays over the course of his writing career. Also in your solution, please report the number of acts and scenes, number of unique speakers, and number of chunks for each play;

```
playyear <- rep(0,36)
numofacts <- rep(0,36)
numofscenes <- rep(0,36)
for (i in 1:36){
  playyear[i] <- playlist[[i]]$year
  numofacts[i] <- playlist[[i]]$actsnum
  numofscenes[i] <- playlist[[i]]$scenesnum
}
playyear # year of the play
```

```
##  [1] "1603" "1607" "1601" "1593" "1608" "1609" "1604" "1598" "1598" "1599"
## [11] "1592" "1591" "1591" "1611" "1597" "1599" "1606" "1595" "1606" "1605"
## [21] "1597" "1601" "1596" "1599" "1605" "1596" "1593" "1595" "1594" "1612"
## [31] "1608" "1594" "1602" "1602" "1595" "1611"
```

```
numofacts # number of acts for each play
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [36] 5
```

```
numofscenes # number of scenes for each play
```

```
##  [1] 23 42 22 11 29 27 20 19 19 23 27 24 28 17 16 18 26  9 29 17 20 23  9
## [24] 17 15 19 25 24 14  9 17 14 24 18 20 15
```

```
# More stats from part d
speakernum # number of speakers per play
```

```
##  [1] 27 61 28 28 65 40 34 34 51 49 56 67 47 48 28 49 23 20 44 25 25 34 33
## [24] 24 27 37 67 35 37 20 59 30 29 21 17 35
```

```
chunksnum # number of chunks per play
```

```
##  [1]  934 1172  807  628 1106  857 1121  755  907  718  663  794  816  704
## [15]  548  796 1062 1044  646  897  633 1024  504  957 1181  553 1087  822
## [29]  892  643  800  565 1141  921  857  744
```

```
sum_sentencecount # total number of sentences per play
```

```
##  [1] 1655 2260 1457 1067 2039 2073 2473 1871 1885 1384 1366 1569 1582 1487
## [15] 1191 1668 2555 1706 1438 1647 1279 2060 1069 1791 2496 1293 2045 2210
## [29] 1602 1234 1573 1268 2202 1592 1329 1655
```

```
ave_wordcount # average number of spoken words per play
```

```
##  [1] 24.10278 20.37116 25.21933 24.01274 24.02622 30.86581 25.17395
##  [8] 31.70464 27.98126 34.34680 31.15988 30.82620 28.57598 32.58381
## [15] 37.39964 23.96985 23.11394 19.47031 25.51703 23.66221 32.39494
## [22] 21.07129 30.06746 21.32706 21.71973 39.47378 26.05980 28.87835
## [29] 22.74215 24.81649 22.21875 34.96283 22.20859 20.63409 19.61377
## [36] 32.73118
```

```
uniqword # number of unique words per play
```

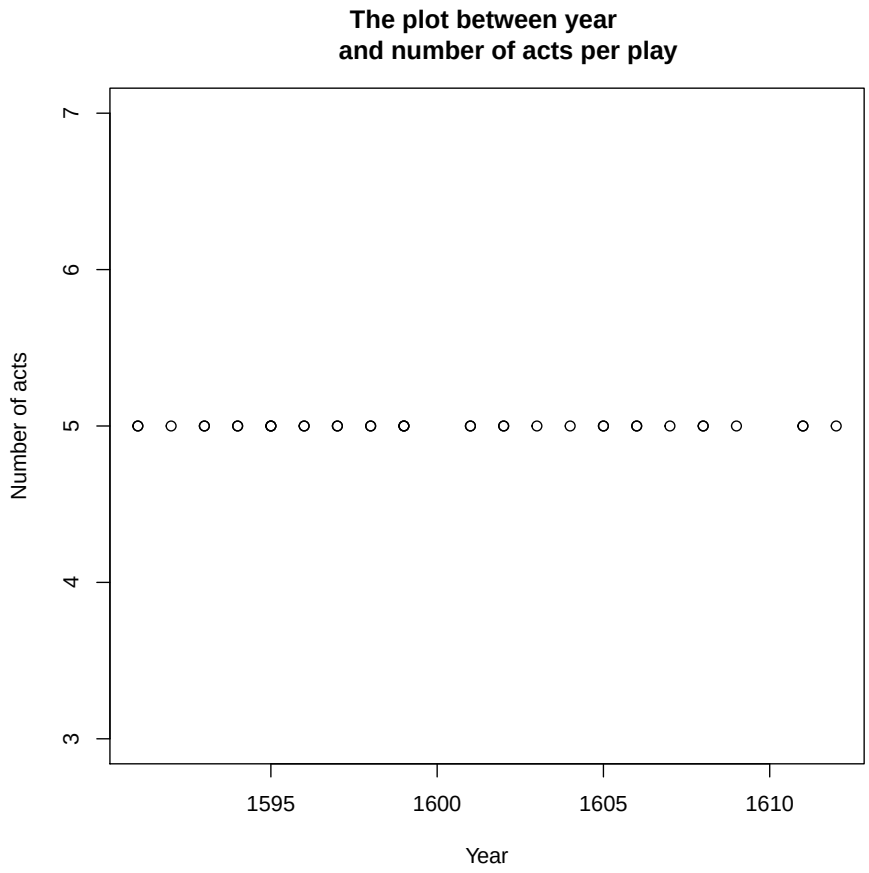
```
##  [1] 3927 4514 3486 2936 4568 4737 5046 4247 4518 4897 4329 4569 3983 4037
## [15] 3993 3294 4624 4074 3772 3716 3589 3628 3190 3294 4195 4153 4617 4155
## [29] 3614 3505 3742 3768 4684 3464 3053 4318
```

```
# Create a new data frame with all the data points above
playstats <- cbind(playyear,numofacts,numofscenes,speakernum,
     chunksnum,sum_sentencecount, ave_wordcount, uniqword)
playstats
```

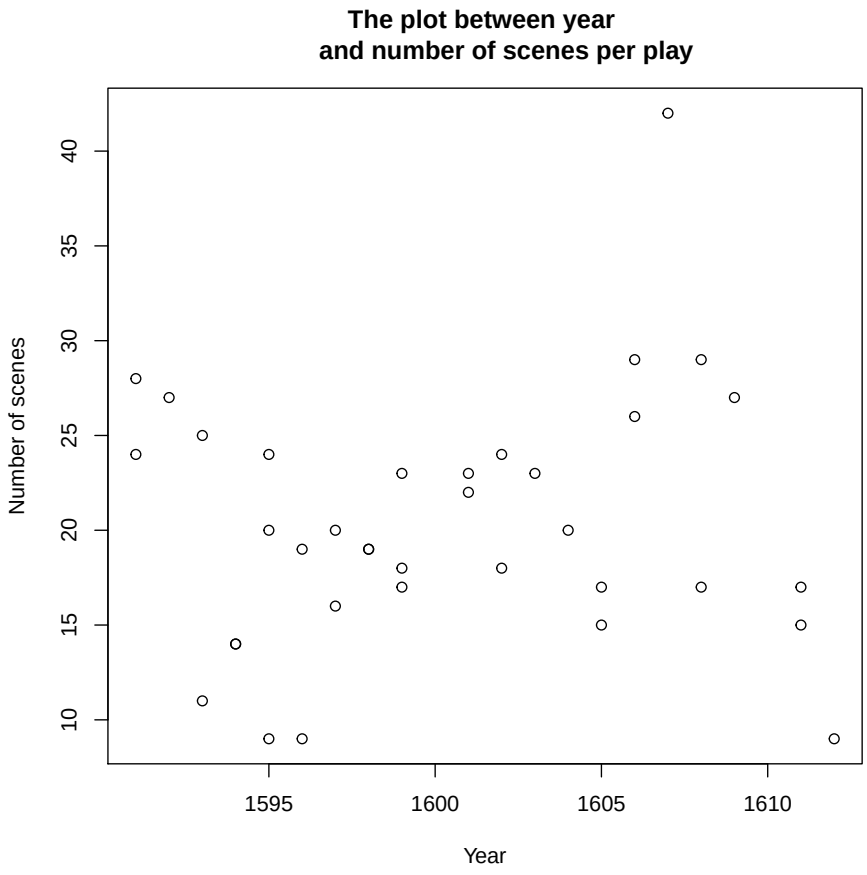
```
##       playyear numofacts numofscenes speakernum chunksnum
##  [1,] "1603"   "5"       "23"        "27"       "934"
##  [2,] "1607"   "5"       "42"        "61"       "1172"
##  [3,] "1601"   "5"       "22"        "28"       "807"
##  [4,] "1593"   "5"       "11"        "28"       "628"
##  [5,] "1608"   "5"       "29"        "65"       "1106"
##  [6,] "1609"   "5"       "27"        "40"       "857"
##  [7,] "1604"   "5"       "20"        "34"       "1121"
##  [8,] "1598"   "5"       "19"        "34"       "755"
##  [9,] "1598"   "5"       "19"        "51"       "907"
```

```
## [10,] "1599"   "5"       "23"        "49"       "718"
## [11,] "1592"   "5"       "27"        "56"       "663"
## [12,] "1591"   "5"       "24"        "67"       "794"
## [13,] "1591"   "5"       "28"        "47"       "816"
## [14,] "1611"   "5"       "17"        "48"       "704"
## [15,] "1597"   "5"       "16"        "28"       "548"
## [16,] "1599"   "5"       "18"        "49"       "796"
## [17,] "1606"   "5"       "26"        "23"       "1062"
## [18,] "1595"   "5"       "9"         "20"       "1044"
## [19,] "1606"   "5"       "29"        "44"       "646"
## [20,] "1605"   "5"       "17"        "25"       "897"
## [21,] "1597"   "5"       "20"        "25"       "633"
## [22,] "1601"   "5"       "23"        "34"       "1024"
## [23,] "1596"   "5"       "9"         "33"       "504"
## [24,] "1599"   "5"       "17"        "24"       "957"
## [25,] "1605"   "5"       "15"        "27"       "1181"
## [26,] "1596"   "5"       "19"        "37"       "553"
## [27,] "1593"   "5"       "25"        "67"       "1087"
## [28,] "1595"   "5"       "24"        "35"       "822"
## [29,] "1594"   "5"       "14"        "37"       "892"
## [30,] "1612"   "5"       "9"         "20"       "643"
## [31,] "1608"   "5"       "17"        "59"       "800"
## [32,] "1594"   "5"       "14"        "30"       "565"
## [33,] "1602"   "5"       "24"        "29"       "1141"
## [34,] "1602"   "5"       "18"        "21"       "921"
## [35,] "1595"   "5"       "20"        "17"       "857"
## [36,] "1611"   "5"       "15"        "35"       "744"
##       sum_sentencecount ave_wordcount      uniqword
##  [1,] "1655"            "24.1027837259101" "3927"
##  [2,] "2260"            "20.3711604095563" "4514"
##  [3,] "1457"            "25.2193308550186" "3486"
##  [4,] "1067"            "24.0127388535032" "2936"
##  [5,] "2039"            "24.0262206148282" "4568"
##  [6,] "2073"            "30.8658109684947" "4737"
##  [7,] "2473"            "25.1739518287244" "5046"
##  [8,] "1871"            "31.7046357615894" "4247"
##  [9,] "1885"            "27.981256890849"  "4518"
## [10,] "1384"            "34.3467966573816" "4897"
## [11,] "1366"            "31.1598793363499" "4329"
## [12,] "1569"            "30.8261964735516" "4569"
## [13,] "1582"            "28.5759803921569" "3983"
## [14,] "1487"            "32.5838068181818" "4037"
## [15,] "1191"            "37.3996350364963" "3993"
## [16,] "1668"            "23.9698492462312" "3294"
## [17,] "2555"            "23.1139359698682" "4624"
```

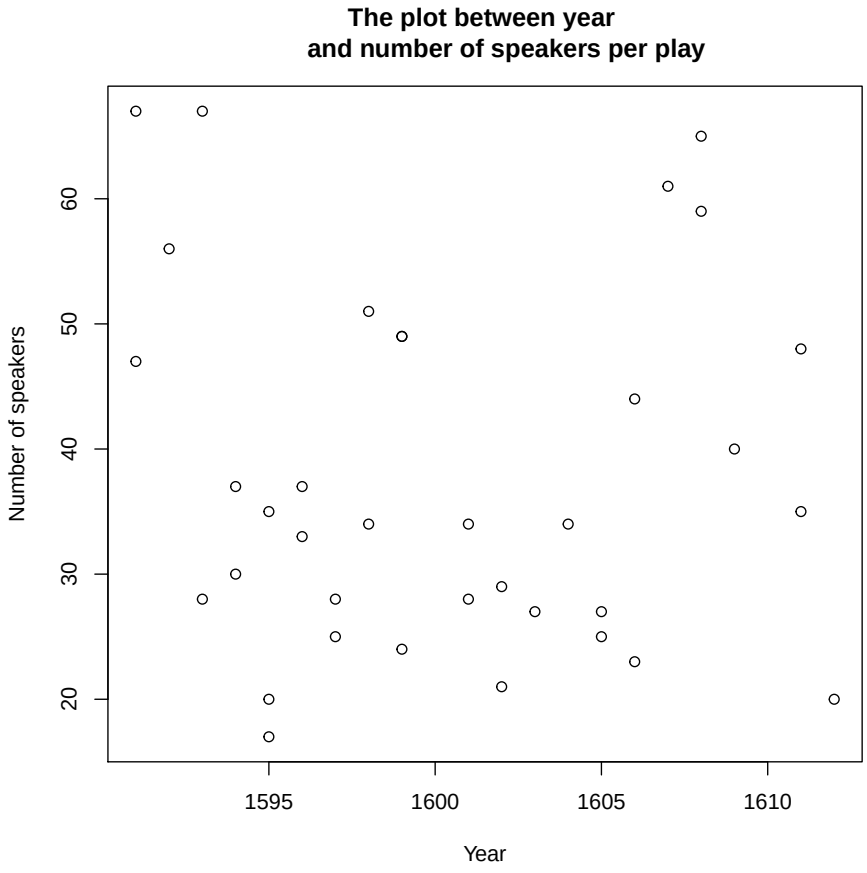
```
## [18,] "1706"            "19.47030651341"   "4074"
## [19,] "1438"            "25.5170278637771" "3772"
## [20,] "1647"            "23.6622073578595" "3716"
## [21,] "1279"            "32.3949447077409" "3589"
## [22,] "2060"            "21.0712890625"    "3628"
## [23,] "1069"            "30.0674603174603" "3190"
## [24,] "1791"            "21.3270637408568" "3294"
## [25,] "2496"            "21.7197290431837" "4195"
## [26,] "1293"            "39.4737793851718" "4153"
## [27,] "2045"            "26.0597976080957" "4617"
## [28,] "2210"            "28.8783454987835" "4155"
## [29,] "1602"            "22.7421524663677" "3614"
## [30,] "1234"            "24.8164852255054" "3505"
## [31,] "1573"            "22.21875"         "3742"
## [32,] "1268"            "34.9628318584071" "3768"
## [33,] "2202"            "22.2085889570552" "4684"
## [34,] "1592"            "20.6340933767644" "3464"
## [35,] "1329"            "19.6137689614936" "3053"
## [36,] "1655"            "32.7311827956989" "4318"

p1 <- plot(playstats[,1], playstats[,2], xlab="Year",
      ylab="Number of acts", main="The plot between year
      and number of acts per play")
```



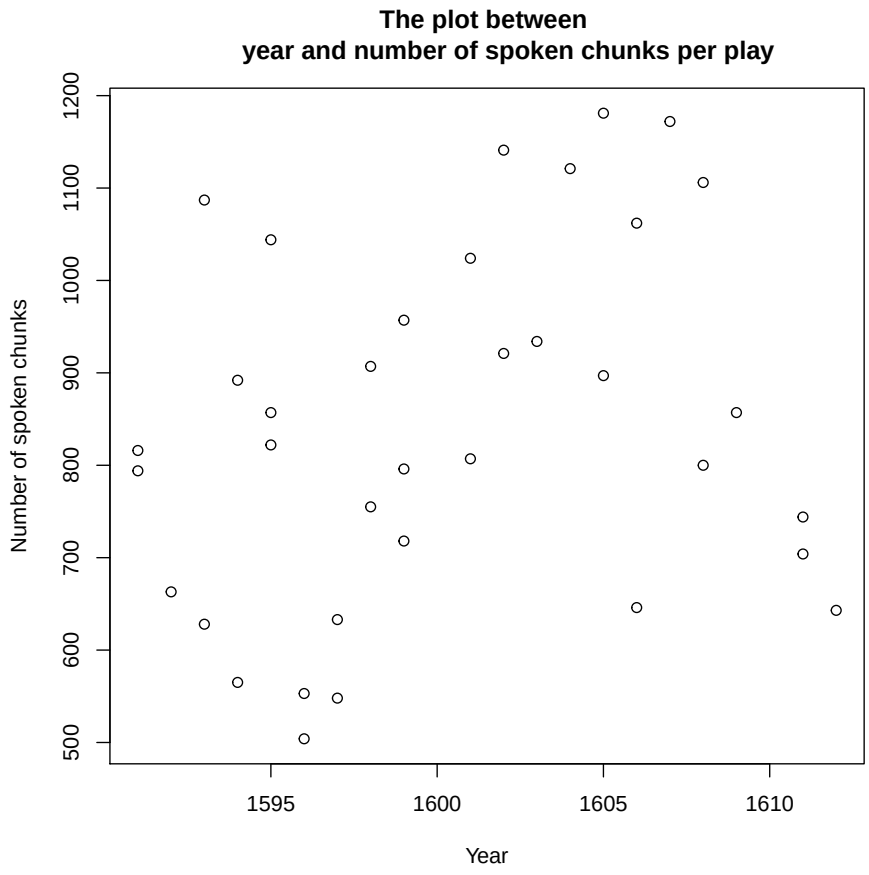

```
p2 <- plot(playstats[,1], playstats[,3], xlab="Year",
      ylab="Number of scenes", main="The plot between year
      and number of scenes per play")
```



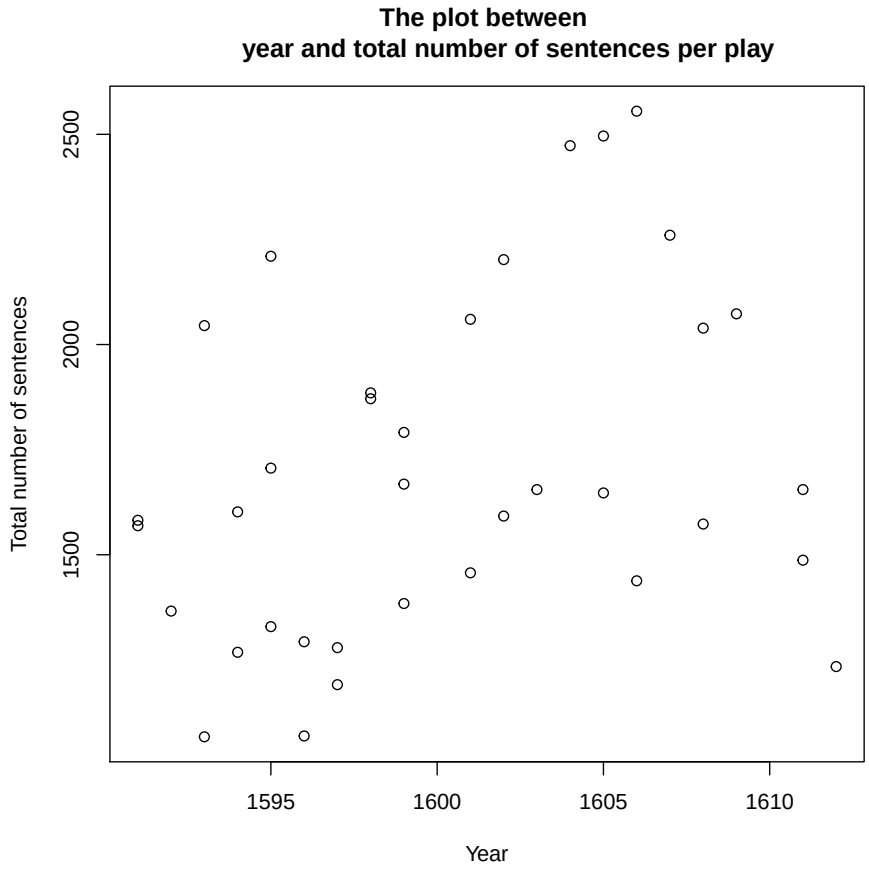

```
p3 <- plot(playstats[,1], playstats[,4], xlab="Year",
      ylab="Number of speakers", main="The plot between year
      and number of speakers per play")
```



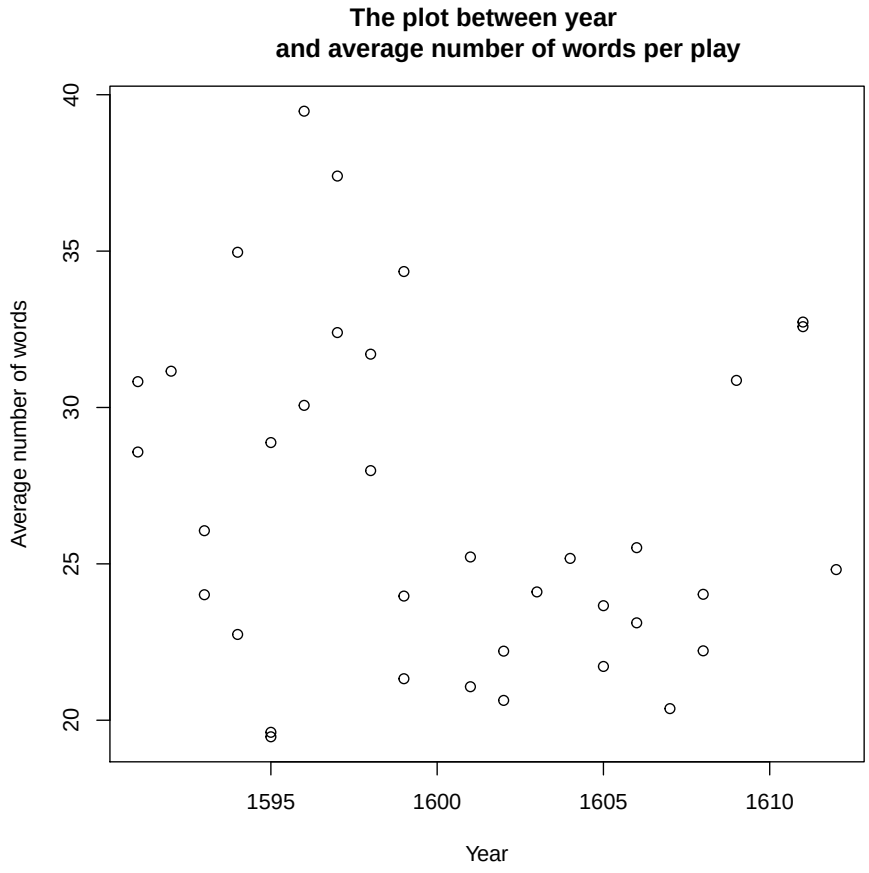

```
p4 <- plot(playstats[,1], playstats[,5], xlab="Year",
      ylab="Number of spoken chunks", main="The plot between
      year and number of spoken chunks per play")
```



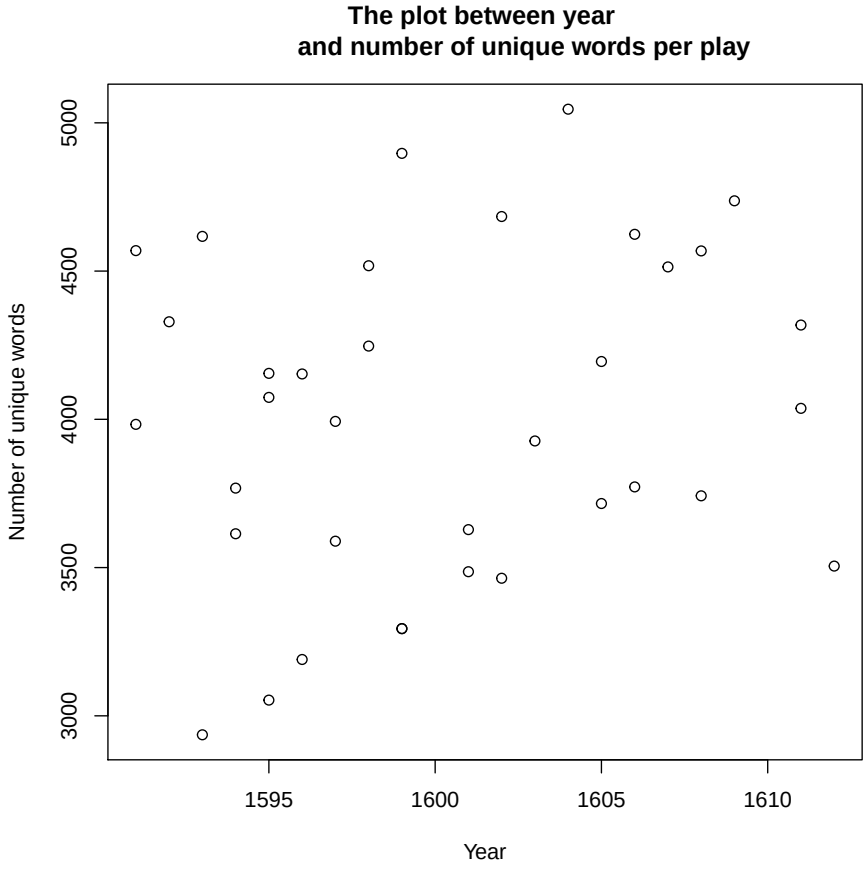

```
p5 <- plot(playstats[,1], playstats[,6], xlab="Year",
      ylab="Total number of sentences", main="The plot between
      year and total number of sentences per play")
```




```
p6 <- plot(playstats[,1], playstats[,7], xlab="Year",
      ylab="Average number of words", main="The plot between year
      and average number of words per play")
```




```
p7 <- plot(playstats[,1], playstats[,8], xlab="Year",
           ylab="Number of unique words", main="The plot between year
           and number of unique words per play")
```



3(a) What are the fields (i.e., member data, slots, etc.) for the class? Describe the kind of variable of each field (e.g., it might be a named numeric vector, or an unnamed list). You might decide to have a field contain an object of yet another class that you should describe.

I created a class in part b with fields such as year, title, number of acts, number of scenes, body of the play and compiletext which contains the speaker name and spoken words for each play. Each field represents a feature of the play. The data type of variables of each play are as follows: year: character string title: character string number of acts (actsnum): integer number of scenes (scenesnum): integer body: character string compiletext: list contianing character string We can consider the field compiletext contains object of another class which includes two fields: the speaker name and spoken words for each play. Both of speaker name and spoken words are character strings.

(b) What are the methods for the class? Some of the methods should relate to processing the text of the plays to produce the fields and other methods should relate to providing information to a user who wants to know something about a play or see the text of the play. Indicate very briefly what the method does,

and any input arguments to each method, fields that are created or modified by the method, and any output that is produced.

Processing the text: We can define a method to return the average number of words spoken by a specific speaker for each play. Input of the method is the title of the play and name of the speaker, of which both are string characters. The output would be the average number of words spoken by that speaker. So for each play, we need to select all the chunks spoken by that specific speaker and count the word number in each chunk to obtian the average word counts for that specific speaker. Output will be an integer. The method can be defined as `ave_wordcount_by_speaker`(x,y)

Provide information: We can create a search method to query the information on Shakespear's works based on the play titles. Input is Shakespear's work titles which are character strings. The method should return the year of the work, number of acts, number of scenes and the body of the play. Year and number of acts/scenes are integers and the body of the play is character string. The method can be defined as `search_info`(x)

Also the statistics we calculated in part 2d can be turned into methods too: Calculating the number of speakers per play can be turned into a method, where the input is play title and the output is the number of speakers of that play. Calculating the number of chunks per play can be turned into a method, where the input is the play title and the output is the number of chunks of that play. Calculating the total number of sentences per play can be turned into a method, where the input is the play title and the output is the number of sentences of that play. Calculating average spoken words per play can be turned into a method, where the input is the play title and the outout is the average number of spoken words of that play. Calculating the number of unique words can be turned into a method too, where the input is the play title and the output is the number of unique words of that play.