Stat243: Problem Set 3

Qingan Zhao
SID: 3033030808

26 Sep. 2017

Problem 1

Reading question: The item I read was *Code and Data for the Social Sciences: A Practitioner's Guide* written by *Gentzkow* and *Shapiro*. I am a little confused about the version control part: If we change the version too frequently, and the version control software just records every version, then what if we do not want to record some of the version? Because when we would like to change something, we have to find the version (to be edited) we want. So how to deal with this issue?

Problem 2

This problem intends to extract data from plays written by William Shakespeare. Each play including the meta data and body, and chunks with corresponding speakers are extracted in the solution. The numbers of unique speakers, spoken chunks, sentences, words, unique words and average number of words per chunk in each play are calculated. The plots of the summarized statistics are given at the end of the solution. Note that *the sonnts* and *Lover's Complaint* are excluded as required, and *the comedy of Errors* is excluded in Problem 2c∼2f since it has different indentation from other plays.

(a) Download the file from the Internet and scan it in R. Use *grep* and *RegEx* to extract the start and the end of each play. Finally put the extracted plays into a list.

```
download.file('http://www.gutenberg.org/cache/epub/100/pg100.txt', destfile = 'shakes.txt')
#scan the file into characters and seperate by lines
text <- scan(file='shakes.txt', what = character(0), sep = "\n")
#the play always begin with a number of a year with 4 digits
play_start <- grep('^[[:digit:]]{4}', text)
#the plays always end with "THE END"
play_end <- grep('THE END', text)
num_plays <- length(play_start)
#the function that extract each play
extract_play <- function(i){
  return(text[play_start[i]:play_end[i]])
}
#exclude the first piece and last piece as required, and put the rest into a list
plays <- c(2:(num_plays-1))
plays <- lapply(plays, extract_play)
#example output
head(plays[[1]])
```

```
## [1] "1603"                      "ALLS WELL THAT ENDS WELL"
## [3] "by William Shakespeare"    "Dramatis Personae"
## [5] "  KING OF FRANCE"          "  THE DUKE OF FLORENCE"
```

```
head(plays[[9]])
```

```
## [1] "1598"                          "SECOND PART OF KING HENRY IV"
## [3] "by William Shakespeare"        "Dramatis Personae"
## [5] "  RUMOUR, the Presenter"       "  KING HENRY THE FOURTH"
```

(b) To extract the meta data and body of the plays, I made several functions and apply them to a vector of the index of each play. For *year*, *title*, and *body* of the play, I extracted them directly. For *act numbers and scene numbers*, I first used *RegEx* to extract the index of all acts and scences together, and then seperated them based on the index, and finally calculated the numbers.

```
plays_num <- length(plays)
#creat an index based on problem 2a
play_index <- c(1:plays_num)

#this function is to find years, note that the year is the first line of each play
find_year <- function(i){
  return(as.integer(plays[[i]][[1]]))
}
play_year <- sapply(play_index, find_year)
#year output
play_year
```

```
##  [1] 1603 1607 1601 1593 1608 1609 1604 1598 1598 1599 1592 1591 1591 1611
## [15] 1597 1599 1606 1595 1606 1605 1597 1601 1596 1599 1605 1596 1593 1595
## [29] 1594 1612 1608 1594 1602 1602 1595 1611
```

```
#this function is to find titles, note that the title is the second line of each play
find_title <- function(i){
  return(as.character(plays[[i]][[2]]))
}
play_title <- sapply(play_index, find_title)
#title output
play_title
```

```
##  [1] "ALLS WELL THAT ENDS WELL"
##  [2] "THE TRAGEDY OF ANTONY AND CLEOPATRA"
##  [3] "AS YOU LIKE IT"
##  [4] "THE COMEDY OF ERRORS"
##  [5] "THE TRAGEDY OF CORIOLANUS"
##  [6] "CYMBELINE"
##  [7] "THE TRAGEDY OF HAMLET, PRINCE OF DENMARK"
##  [8] "THE FIRST PART OF KING HENRY THE FOURTH"
##  [9] "SECOND PART OF KING HENRY IV"
## [10] "THE LIFE OF KING HENRY THE FIFTH"
## [11] "THE FIRST PART OF HENRY THE SIXTH"
## [12] "THE SECOND PART OF KING HENRY THE SIXTH"
## [13] "THE THIRD PART OF KING HENRY THE SIXTH"
## [14] "KING HENRY THE EIGHTH"
## [15] "KING JOHN"
## [16] "THE TRAGEDY OF JULIUS CAESAR"
## [17] "THE TRAGEDY OF KING LEAR"
## [18] "LOVE'S LABOUR'S LOST"
## [19] "THE TRAGEDY OF MACBETH"
## [20] "MEASURE FOR MEASURE"
```

```
## [21] "THE MERCHANT OF VENICE"
## [22] "THE MERRY WIVES OF WINDSOR"
## [23] "A MIDSUMMER NIGHT'S DREAM"
## [24] "MUCH ADO ABOUT NOTHING"
## [25] "THE TRAGEDY OF OTHELLO, MOOR OF VENICE"
## [26] "KING RICHARD THE SECOND"
## [27] "KING RICHARD III"
## [28] "THE TRAGEDY OF ROMEO AND JULIET"
## [29] "THE TAMING OF THE SHREW"
## [30] "THE TEMPEST"
## [31] "THE LIFE OF TIMON OF ATHENS"
## [32] "THE TRAGEDY OF TITUS ANDRONICUS"
## [33] "THE HISTORY OF TROILUS AND CRESSIDA"
## [34] "TWELFTH NIGHT; OR, WHAT YOU WILL"
## [35] "THE TWO GENTLEMEN OF VERONA"
## [36] "THE WINTER'S TALE"
```

```
#create a vector which will be used in the following function
index <- c()
#this function searches all of the acts and scenes of each play
index_search <- function(play) {
  #all acts (contains several scences) begin with the following 3 types
  index_lines <- grep("^ACT |SCENE |Scene", plays[[play]])
  index_num <- length(index_lines)
  index <- c(1:index_num)
  #a sub-function that extract the index lines of the acts and scenes
  index_index <- function(i){
    return(as.character(plays[[play]][[index_lines[i]]]))
  }
  index <- sapply(index, index_index)
  return(index)
}
#example output
index_search(1)
```

```
##  [1] "ACT I. SCENE 1."   "ACT I. SCENE 2."   "ACT I. SCENE 3."
##  [4] "ACT II. SCENE 1."  "ACT II. SCENE 2."  "ACT II. SCENE 3."
##  [7] "ACT II. SCENE 4."  "ACT II. SCENE 5."  "ACT III. SCENE 1."
## [10] "ACT III. SCENE 2." "ACT III. SCENE 3." "ACT III. SCENE 4."
## [13] "ACT III. SCENE 5." "ACT III. SCENE 6." "ACT III. SCENE 7."
## [16] "ACT IV. SCENE 1."  "ACT IV. SCENE 2."  "ACT IV. SCENE 3."
## [19] "ACT IV SCENE 4."   "ACT IV SCENE 5."   "ACT V. SCENE 1."
## [22] "ACT V SCENE 2."    "ACT V SCENE 3."
```

```
index_search(7)
```

```
##  [1] "ACT I. Scene I."   "Scene II."         "Scene III."
##  [4] "Scene IV."         "Scene V."          "Act II. Scene I."
##  [7] "Scene II."         "ACT III. Scene I." "Scene II."
## [10] "Scene III."        "Scene IV."         "ACT IV. Scene I."
## [13] "Scene II."         "Scene III."        "Scene IV."
## [16] "Scene V."          "Scene VI."         "Scene VII."
## [19] "ACT V. Scene I."   "Scene II."
```

```
#this function separates and counts the index lines of acts
```

```r
find_act_num <- function(i){
  #to count acts, we can count scene 1s instead, because each act must has and only has one scene 1
  #we also have to exclude other scenes begin with 'scene 1' such as scene 10 and scene IV
  return(length(grep("(?i)SCENE 1[^(0-9)]\\.?|(?i)SCENE 1$|(?i)SCENE I[^(A-Z)]", index_search(i))))
}

#this function counts the index lines of scenes
find_scene_num <- function(i)
  #this makes sense because we made the index_search() function based on total scenes
  return(length(index_search(i)))

#put the act numbers and scene numbers into vectors based on the plays
act_num <- sapply(play_index, find_act_num)
scene_num <- sapply(play_index, find_scene_num)

#scene numbers output
scene_num
```

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

```r
#act numbers output
act_num
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 5 6 5 5 5 5 5 5
## [36] 5
```

```r
#this part deals with the bodies
body_start <- c()
body_end <- c()
#each body begins with scene: or scene.
body_start <- grep('SCENE(:|\\.)|Scene(:|\\.)', text)
body_end <- grep('THE END', text)
#this function extract the bodys
find_body <- function(i){
  return(text[body_start[i]:body_end[i+1]])
}
#put the bodies into a list
play_body <- lapply(play_index, find_body)
#example output
head(play_body[[1]])
```

```
## [1] "SCENE:"
## [2] "Rousillon; Paris; Florence; Marseilles"
## [3] "ACT I. SCENE 1."
## [4] "Rousillon. The COUNT'S palace"
## [5] "Enter BERTRAM, the COUNTESS OF ROUSILLON, HELENA, and LAFEU, all in black"
## [6] "  COUNTESS. In delivering my son from me, I bury a second husband."
```

```r
head(play_body[[13]])
```

```
## [1] "SCENE:"
## [2] "England and France"
## [3] "ACT I. SCENE I."
## [4] "London. The Parliament House"
## [5] "Alarum. Enter DUKE OF YORK, EDWARD, RICHARD, NORFOLK, MONTAGUE, WARWICK,"
## [6] "and soldiers, with white roses in their hats"
```

(c) To extract the spoken chunks, I relied on the indentation of the beginning and body of each chunk. The 4th play (i.e., *The Comedy of Errors*) is excluded because of the unusual indentation. Basically, I used the detected beginnings of the chunks to extract the chunk bodies which is between the beginnings.

```
#exclude the 4th play
plays[[4]] <- list()
#this function searches the beginning of each chunk including the corresponding speaker
find_chunk <- function(i){
  #each chunk begins with the speaker and indents 2 spaces
  #the speaker's names are all capital letters in most plays
  #there is only one captial letter (first letter) in the names in some plays
  #some names have spaces (maybe first and last names together)
  #all names end with a dot
  chunks <- grep('^[[:space:]]{2}[[:upper:]]{1,10}[ ]?[[:alpha:]]{1,15}\\.', plays[[i]])
  return(chunks)
}
#put all chunk beginnings into a vector
chunks <- sapply(play_index, find_chunk)
chunks_length <- as.integer(sapply(chunks, length))
#this function returns the chunks of each play
make_chunks <- function(i){
  #exclude the 4th play
  if(i == 4)
    return(0)
  else
    #this sub-function extracts the chunks
    make_chunks_sub <- function (j){
      #use 2 adjacent chunk beginnings to extract each chunk
      chunk_temp <- plays[[i]][chunks[[i]][[j-1]]:(chunks[[i]][[j]]-1)]
      return(chunk_temp)
    }
    #beginning with 2 because I used (j-1) in the make_chunks_sub() function
    chunk_length_temp <- c(2:chunks_length[i])
    chunk_all <- sapply(chunk_length_temp, make_chunks_sub)
    return(chunk_all)
}
#put all chunks into a list
play_chunk <- lapply(play_index, make_chunks)

#example output
head(play_chunk[[1]])

## [[1]]
## [1] "  COUNTESS. In delivering my son from me, I bury a second husband."
##
## [[2]]
## [1] "  BERTRAM. And I in going, madam, weep o'er my father's death anew;"
## [2] "    but I must attend his Majesty's command, to whom I am now in"
## [3] "    ward, evermore in subjection."
##
## [[3]]
## [1] "  LAFEU. You shall find of the King a husband, madam; you, sir, a"
## [2] "    father. He that so generally is at all times good must of"
## [3] "    necessity hold his virtue to you, whose worthiness would stir it"
```

```
## [4] "    up where it wanted, rather than lack it where there is such"
## [5] "    abundance."
##
## [[4]]
## [1] "  COUNTESS. What hope is there of his Majesty's amendment?"
##
## [[5]]
## [1] "  LAFEU. He hath abandon'd his physicians, madam; under whose"
## [2] "    practices he hath persecuted time with hope, and finds no other"
## [3] "    advantage in the process but only the losing of hope by time."
##
## [[6]]
## [1] "  COUNTESS. This young gentlewoman had a father- O, that 'had,' how"
## [2] "    sad a passage 'tis!-whose skill was almost as great as his"
## [3] "    honesty; had it stretch'd so far, would have made nature"
## [4] "    immortal, and death should have play for lack of work. Would, for"
## [5] "    the King's sake, he were living! I think it would be the death of"
## [6] "    the King's disease."
```

```
head(play_chunk[[20]])
```

```
## [[1]]
## [1] "  DUKE. Escalus!"
##
## [[2]]
## [1] "  ESCALUS. My lord."
##
## [[3]]
##  [1] "  DUKE. Of government the properties to unfold"
##  [2] "    Would seem in me t' affect speech and discourse,"
##  [3] "    Since I am put to know that your own science"
##  [4] "    Exceeds, in that, the lists of all advice"
##  [5] "    My strength can give you; then no more remains"
##  [6] "    But that to your sufficiency- as your worth is able-"
##  [7] "    And let them work. The nature of our people,"
##  [8] "    Our city's institutions, and the terms"
##  [9] "    For common justice, y'are as pregnant in"
## [10] "    As art and practice hath enriched any"
## [11] "    That we remember. There is our commission,"
## [12] "    From which we would not have you warp. Call hither,"
## [13] "    I say, bid come before us, Angelo.         Exit an ATTENDANT"
## [14] "    What figure of us think you he will bear?"
## [15] "    For you must know we have with special soul"
## [16] "    Elected him our absence to supply;"
## [17] "    Lent him our terror, dress'd him with our love,"
## [18] "    And given his deputation all the organs"
## [19] "    Of our own power. What think you of it?"
##
## [[4]]
## [1] "  ESCALUS. If any in Vienna be of worth"
## [2] "    To undergo such ample grace and honour,"
## [3] "    It is Lord Angelo."
## [4] "                          Enter ANGELO"
##
```

```
## [[5]]
## [1] "  DUKE. Look where he comes."
##
## [[6]]
## [1] "  ANGELO. Always obedient to your Grace's will,"
## [2] "    I come to know your pleasure."
```

Comment: This method unfortunately cannot extract the last chunk of each play. I tried to make a function that searches the end of each chunk, and use the beginning and end to extract the chunks. To find the end of a chunk, we know that the last line of a chunk indents 4 spaces while the next line does not indents 4 spaces. I wrote the following codes to search the end of each chunk, but the result is bad. (beginnings and ends do not match) So I have to use the previous method.

```
#the code has been split for better view
find_chunk_end <- grep('^[[:space:]]{4}([[:space:]]|[[:graph:]])*\n[[:graph:]]|[[:space:]]
{1,3}|[[:space:]]{4,}$', plays[[i]])
```

(d) I basically use *RegEx* to extract the items required, putting them into vectors or lists, and use length() function to count the numbers. The solution of this part is similar to the previous ones.

```
#this function counts the unique speakers
unique_speaker <- function(i){
  if(i == 4)
    return(0)
  else
    #search the speaker's name using the same method in problem 2c
    speakers <- gregexpr('^[[:space:]]{2}[[:upper:]]{1,10}[ ]?[[:alpha:]]{1,15}\\.',
                         as.character(make_chunks(i)))
    #extract the matched names
    speaker_name <- regmatches(as.character(make_chunks(i)), speakers)
    #find unique names
    unique_name <- unique(speaker_name)
    return(length(unique_name))
}
#put the speaker numbers into a vector
play_unique_speaker <- sapply(play_index, unique_speaker)
#unique speaker numbers output
play_unique_speaker
```

```
##  [1] 23 53 24  0 47 29 28 30 40 35 37 53 38 41 24 46 22 17 36 23 19 25 27
## [24] 25 20 29 51 28 34 16 47 22 29 17 18 26
```

```
#this function counts the spoken chunks
spoken_chunks <- function(i){
  if(i == 4)
    return(0)
  else
    return(length(make_chunks(i)))
}
#put the chunk numbers into a list
play_chunks <- sapply(play_index, spoken_chunks)
#chunk numbers output
play_chunks
```

```
##  [1]  932 1171  804    0 1103  854 1120  745  901  716  645  789  799  703
## [15]  547  792 1052  939  642  894  607 1017  503  956 1180  551 1065  811
## [29]  888  640  754  561 1140  920  856  742
```

```
#this function counts the sentences of each play
sentence <- function(i){
  if(i == 4)
    return(0)
  else
    #use periods to detect sentences
    periods <- sum(sapply(gregexpr('\\.', make_chunks(i)), length))
    return(periods)
}
#put sentence numbers into a vector
play_sentence <- sapply(play_index, sentence)
#sentence numbers output
play_sentence
```

```
##  [1] 2319 2929 1955    0 2638 2405 3151 2330 2290 1800 1714 1924 2020 1898
## [15] 1374 2144 3031 2305 1877 2112 1641 2560 1346 2498 2938 1529 2539 2515
## [29] 2018 1514 1924 1478 2670 2096 1860 2017
```

```
#this function counts the words
word <- function(i){
  if(i == 4)
    return(0)
  else
    #use "\\w+" to detect words
    word_temp <- sum(sapply(gregexpr('\\w+', make_chunks(i)), length))
  return(word_temp)
}
#put the word numbers into a vector
play_word <- sapply(play_index, word)
#word numbers output
play_word
```

```
##  [1] 25822 27907 23790     0 30926 30550 33991 27133 28819 27582 23983
## [12] 28155 27322 26806 22818 21701 29252 24131 19179 24056 23179 25080
## [23] 18096 23816 29230 24428 32649 27494 23584 18596 20777 22735 28817
## [34] 22266 19368 27559
```

```
#this function counts the average number of words per chunk
average_word <- function(i){
  if(i == 4)
    return(0)
  else
    #the number of words divided by the number of chunks of each play
    return(word(i) / chunks_length[i])
}
#put the average numbers of words per chunk into a vector
play_average_word <- sapply(play_index, average_word)
#average numbers of words per chunk output
play_average_word
```

```
##  [1] 27.67631 23.81143 29.55280  0.00000 28.01268 35.73099 30.32203
```

```
##  [8] 36.37131 31.95011 38.46862 37.12539 35.63924 34.15250 38.07670
## [15] 41.63869 27.36570 27.77968 25.67128 29.82737 26.87821 38.12336
## [22] 24.63654 35.90476 24.88610 24.75021 44.25362 30.62758 33.85961
## [29] 26.52868 29.01092 27.51921 40.45374 25.25592 24.17590 22.59977
## [36] 37.09152

#this function counts the unique words
unique_word <- function(i){
  if(i == 4)
    return(0)
  else
    #search the words
    words <- gregexpr('\\w+', as.character(make_chunks(i)))
    #extract the matched words
    match_words <- regmatches(as.character(make_chunks(i)), words)
    #find unique words
    unique_words <- unique(match_words)
    return(length(unique_words))
}
#put the unique word numbers into a vector
play_unique_word <- sapply(play_index, unique_word)
#unique word numbers output
play_unique_word

##  [1]  931 1167  801    0 1101  853 1118  740  898  715  645  789  799  703
## [15]  545  788 1046  937  640  891  607 1015  502  955 1166  550 1061  807
## [29]  888  638  752  561 1133  918  854  740
```

(e) First, I put all the summarized vectors into a data frame. Then I used plot() function to plot act numbers, scene numbers, unique speaker numbers, and chunk numbers (the same items as those to be reported). Finally reported the 4 required items.

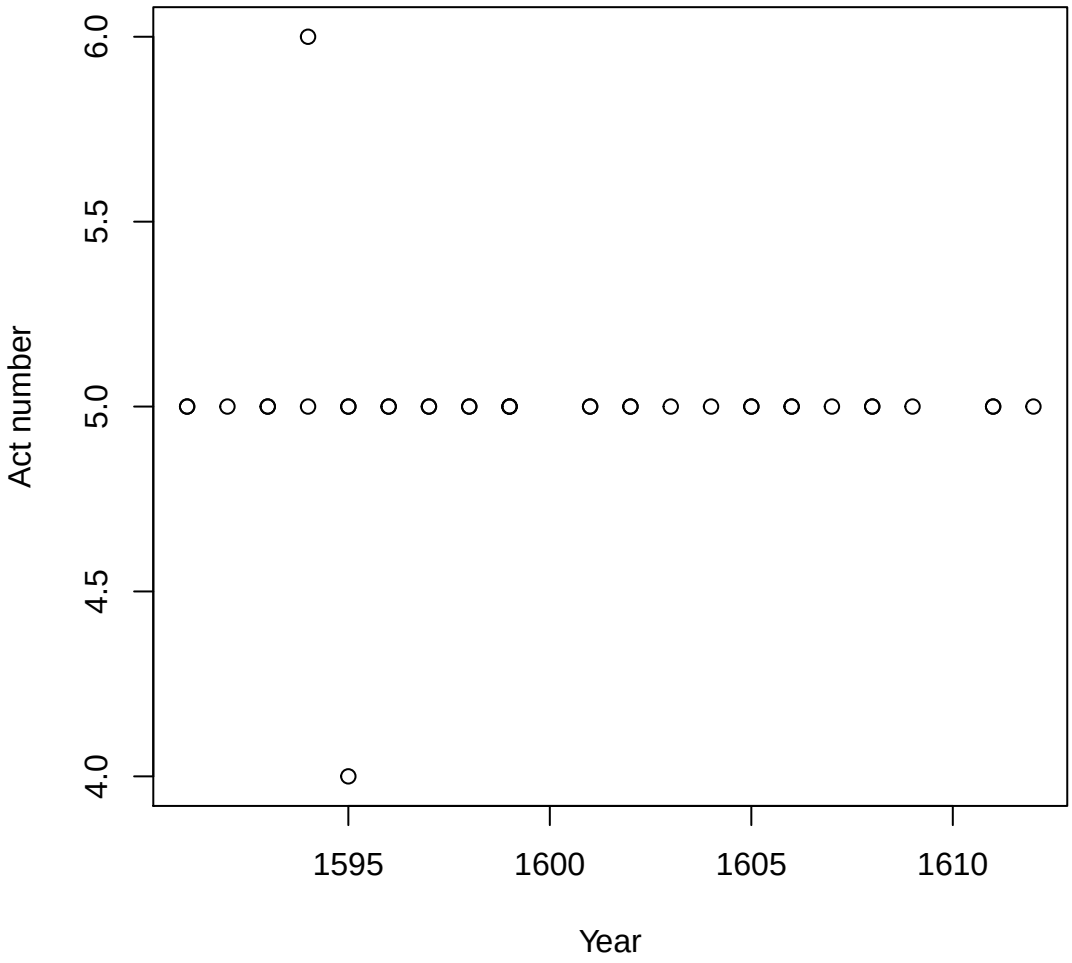
```
#put the summarized statistics into a data frame
results <- data.frame(play_year, play_title, act_num, scene_num, play_unique_speaker,
                      play_chunks, play_sentence, play_word, play_average_word,
                      play_unique_word)
#output a few lines of the data frame, note that '0's are in the 4th play because I exclued it
head(results)

##   play_year                          play_title act_num scene_num
## 1      1603            ALLS WELL THAT ENDS WELL       5        23
## 2      1607 THE TRAGEDY OF ANTONY AND CLEOPATRA       5        42
## 3      1601                       AS YOU LIKE IT       5        22
## 4      1593                THE COMEDY OF ERRORS       5        11
## 5      1608           THE TRAGEDY OF CORIOLANUS       5        29
## 6      1609                            CYMBELINE       5        27
##   play_unique_speaker play_chunks play_sentence play_word
## 1                  23         932          2319     25822
## 2                  53        1171          2929     27907
## 3                  24         804          1955     23790
## 4                   0           0             0         0
## 5                  47        1103          2638     30926
## 6                  29         854          2405     30550
##   play_average_word play_unique_word
```

```
## 1          27.67631              931
## 2          23.81143             1167
## 3          29.55280              801
## 4           0.00000                0
## 5          28.01268             1101
## 6          35.73099              853
```

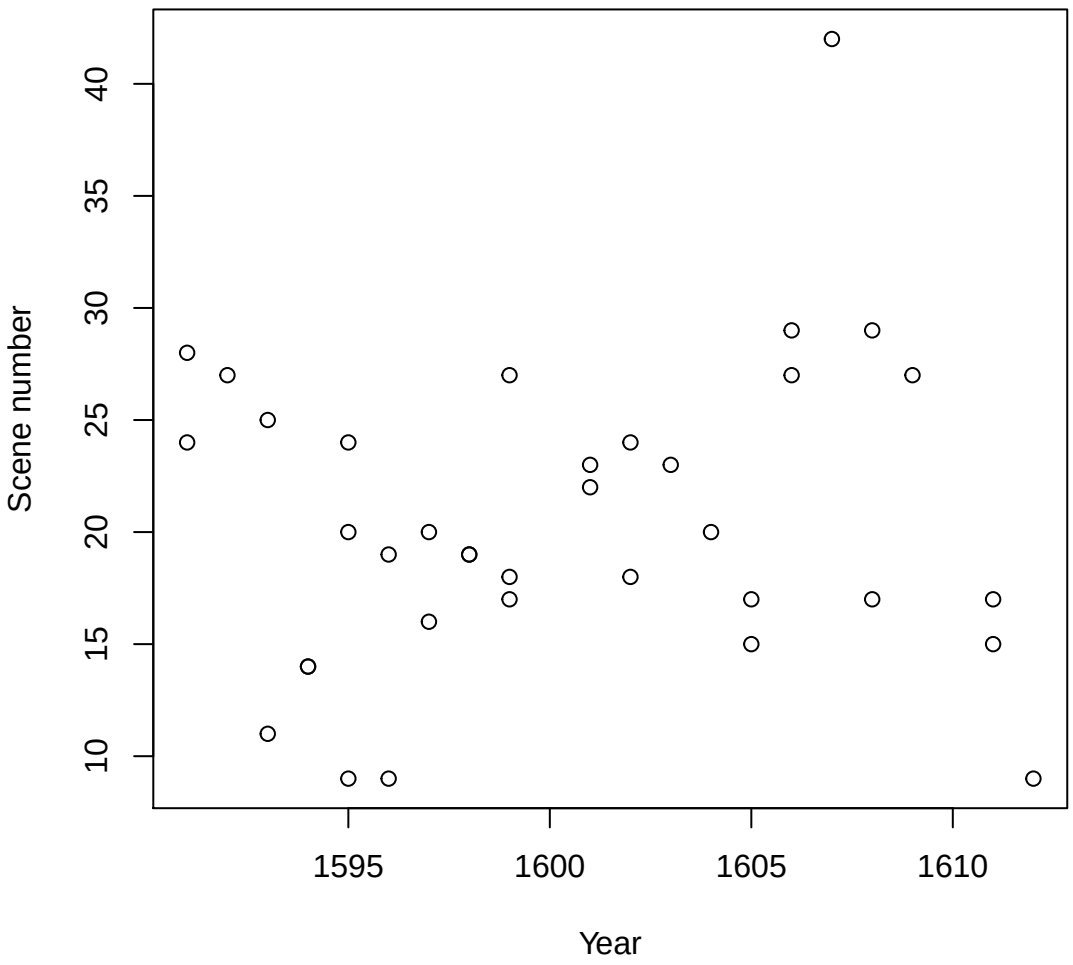
Plot the act numbers as a function of time:

```
plot(results$play_year, results$act_num, xlab='Year', ylab='Act number')
```



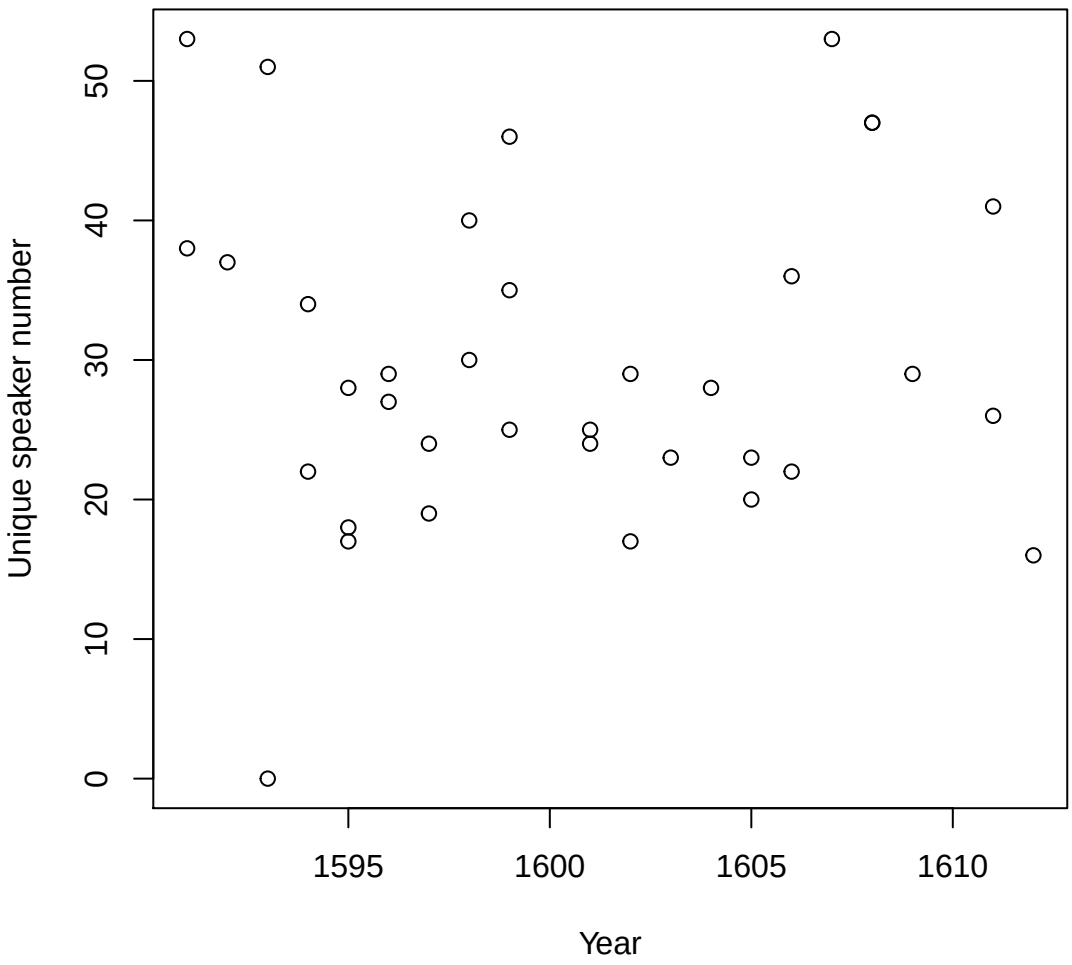
Plot the scene numbers as a function of time:

```
plot(results$play_year, results$scene_num, xlab='Year', ylab='Scene number')
```



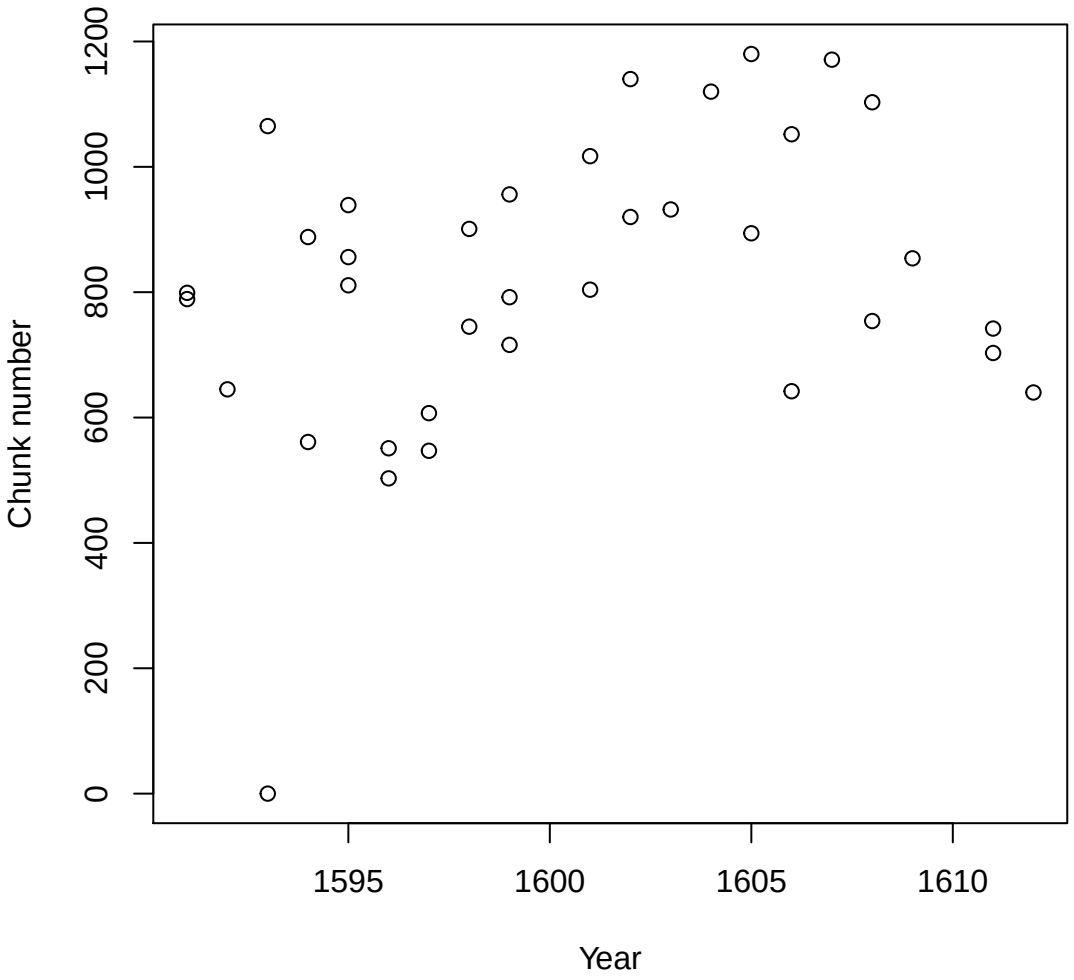
Plot the unique speaker numbers as a function of time (Note that "0" stands for the play excluded):

```
plot(results$play_year, results$play_unique_speaker, xlab='Year', ylab='Unique speaker number')
```



Plot the chunk numbers as a function of time (Note that "0" stands for the play excluded):

```
plot(results$play_year, results$play_chunks, xlab='Year', ylab='Chunk number')
```



Comment: From the plots, it seems that almost all plays have 5 acts. There seems no trends in scene numbers. The unique speaker number and chunk number goes up and down with a period of about 7 years, and it seems that they are in negative correlation.

Report the 4 items: (Note that "0" stands for the play excluded)

```
#report the number of acts for each play:
act_num

##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 5 5 6 5 5 5 5 5 5
## [36] 5

#report the number of scenes for each play:
scene_num

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

```r
#report the number of unique speakers for each play:
play_unique_speaker
```

```
##  [1] 23 53 24  0 47 29 28 30 40 35 37 53 38 41 24 46 22 17 36 23 19 25 27
## [24] 25 20 29 51 28 34 16 47 22 29 17 18 26
```

```r
#report the number of chunks for each play:
play_chunks
```

```
##  [1]  932 1171  804    0 1103  854 1120  745  901  716  645  789  799  703
## [15]  547  792 1052  939  642  894  607 1017  503  956 1180  551 1065  811
## [29]  888  640  754  561 1140  920  856  742
```

Problem 3

(a) Using S4 approach, suppose that we define a new class called *plays*, we let the fields of the class *plays* be *year*, *title*, *act*, *scene*, and *body*.

```r
#define a class called "plays" which contains the following slots
setClass("plays",
         representation(
           year = "numeric",
           title = "character",
           act = "character",
           scene = "character",
           body = "list"
         )
)
#then set the validity of each slots
setValidity("plays",
            function(object){
              #William Shakespeara (1564-1616)
              if(!(object@year %in% 1564:1616))
                return("error: not a valid year")
              if(length(grep('[^a-zA-Z ]', object@title)))
                return("error: title contains non-characters")
              if(!length(grep('^ACT ', object@act)))
                return("error: not a valid act")
              if(!length(grep('(?i)SCENE 1[^(0-9)]\\.?|(?i)SCENE 1$|(?i)SCENE I[^(A-Z)]',
                              object@scene)))
                return("error: not a valid scene")
              if(!length(grep('SCENE(:|\\.)|Scene(:|\\.)', object@body)))
                return("error: not a valid body")
              return(TRUE)
            }
)
```

```
## Class "plays" [in ".GlobalEnv"]
##
## Slots:
##
## Name:       year     title       act     scene      body
## Class:   numeric character character character      list
```

```
#make an object as an example
a_play_act4 <- new('plays', year=1601, title="AS YOU LIKE IT", act="ACT IV.",
              scene=c("SCENE I.", "SCENE II.", "SCENE III"),
              body=list("SCENE: OLIVER'S house..."))
a_play_act4

## An object of class "plays"
## Slot "year":
## [1] 1601
##
## Slot "title":
## [1] "AS YOU LIKE IT"
##
## Slot "act":
## [1] "ACT IV."
##
## Slot "scene":
## [1] "SCENE I."  "SCENE II." "SCENE III"
##
## Slot "body":
## [[1]]
## [1] "SCENE: OLIVER'S house..."
```

(b) The first type of method is to validate the slots so that we can produce the right slots when processing the text. Input values to the slots, and the output should indicate whether the value is valid. The second type is to provide information of a given play (or certain act or scene). Input an argument of class *plays*, and the output should be the exact information of this play. Here I gave two examples of each type:

```
#example 1: validate the year
setGeneric("isYear", function(object){
  standardGeneric("isYear")
})

## [1] "isYear"

#check whether the year is valid
isYear.plays <- function(object){
  if(object@year %in% 1564:1616)
    cat(object@title, "is of valid year")
  else
    cat(object@title, "is not of valid year")
}
setMethod(isYear, signature=c("plays"), definition=isYear.plays)

## [1] "isYear"
## attr(,"package")
## [1] ".GlobalEnv"

#example output
isYear(a_play_act4)

## AS YOU LIKE IT is of valid year

#if we change the number:
a_play_act4@year <- 99999
isYear(a_play_act4)
```

```
## AS YOU LIKE IT is not of valid year

#example 2: provide the scene number information for an act of a play
setGeneric("scenenum", function(object){
  standardGeneric("scenenum")
})

## [1] "scenenum"

scenenum.plays <- function(object){
  cat(object@title, object@act, "has", length(object@scene), "scenes")
}
setMethod(scenenum, signature=c("plays"), definition=scenenum.plays)

## [1] "scenenum"
## attr(,"package")
## [1] ".GlobalEnv"

#example output
scenenum(a_play_act4)

## AS YOU LIKE IT ACT IV. has 3 scenes
```