PS3

Mengying Yang

September 29, 2017

Problem 1

a)

The paper I read is Best Practice for Scientific Computing by Greg Wilson, D.A. Aruliah, C. Titus Brown, Neil P. Chue Hong, and so on. This paper indicated building scientific software should be carefully and seriously to avoid errors happened when, revise, reuse and improve the software. Therefore, the author provides and explains 10 practices that we help as to reduce the probability that error happened.

I think currently I have used several practices that the author mentioned, such as 'Write programs for People, not Computer' and 'Document Design and Purpose, not mechanics'. I think I need to thanks to the professor who taught me the first course of programming. The professor taught us the good behaviors similar as how the author describe in practice 1 and practice 10 . Although I haven't written very complicated scientific software, but I gain benefit each time when I reviewed my code that I wrote before. Usually, I feel this behavior saved me lots of time on reminding and understanding the code.

In addition to that, about version control, what I learnt from my personal experience is when it comes to collaborating, version control is important. For example, before the beginning to use google doc, when I wrote documents with my teammates, we often worked on our own task first. In the end, we would sit together and merged the document content. However, this is way less efficient than with version control features google docs provided. As we can track our changes, we don't have to rename our document at the time of our meeting just to track what's the progress we made so far. Also, as we are using git for this course, I can edit the content on one machine and push changes to remote branch then continue working on it with another machine later by simply pulling the change. Without it I would have to store all my changes in an email draft.

I think this suggestions are very useful, but it will take time to get to used them and let them become our behavior.

Problem 2

a)

This part helps to read the Shakespear's plays txt file on the website and splits each play into a single element and save them into a list. My idea at here is find the number of lines of the begining of the play so that we just need to get all the strings between that two lines

```
#Loading required library
library(readr)
library(stringr)
library(tibble)

URL <- "http://www.gutenberg.org/cache/epub/100/pg100.txt"
html <- readLines(URL)

#find the number of line of years at the beginning of each play in the whole string vector
```

```r
num <- grep('^[[:digit:]]{4}$', html)

#delete the sonnet and the last play(the lover's complaints)
num <- num[2:38]

play <- list()

#extract all the strings between the number of year of each play so that we can
#have the content of each play and save the them into the element i of the play list
for(i in seq(length(num)-1)){
  play[[i]] <- html[num[i]:num[i+1]-1]
}
head(play[[1]])
```

```
## [1] ""                          "1603"
## [3] ""                          "ALLS WELL THAT ENDS WELL"
## [5] ""                          "by William Shakespeare"
```

1 Problem 2

b)

This part I extract the year and title, count the number of Act and Scene, and extract the body of the Shakespear play start from 'Scene:' and end at 'THE END'

```r
num_Scene <- 1 : 36
num_Acts <- 1 : 36
year <- 1 : 36
num_bodyst <-1:36
body <- list()
title <- 1:36

#This is the function to find the title of each play, basically is finding the first
#non-empty string, which  will be the title. write a function because some of the
#title is one line after year, but some is two line after year

playtitle <- function(i){
    t <- 3
    hvtitle <- TRUE
    while(hvtitle){
      title <- play[[i]][t]
      if(title != ""){
        hvtitle <- FALSE
      }
      t = t + 1
    }
    return(title)
}


#There are 36 plays in total, so loop 36 times to get information of each play
for(i in seq(36)){
```

```
  #I found year is in the second line of each play I extract in a)
  year[i] <- play[[i]][2]

  #There are several different format of Scene(eg: Scene 1 or SCENE III ...); Use
  #regular expression to represent them; grep and count the number of lines.
  num_Scene[i] <- length(grep('(([A][Cc][Tt]\\s*.*)?\\s*SCENE\\s[0-9IVX]{1,4}[\\.]?|Scene\\s[0-9IVX]{1,4

  #count the number of Act of assigned play. The method I use at here is count the
  #Scene 1. The number of Scene 1 should equal to number of Act. Note: However, there
  #is text mistake in the txt file , there is no Act II Scene I in play 18, which
  #directly starts from ACT II Scene II.
  num_Acts[i] <- length(grep('(SCENE\\s[1I][\\.]?$|Scene\\s[1I]\\.)',play[[i]]))

  #find the number of line of the 'THE END', which actually is the end of the body
  num_End <- grep('THE END', play[[i]])
  #find the number of line of the Sence:/SCENE:/SCENE.-,which is start of the body
  num_bodyst[i] <- grep('(SCENE:|SCENE.-|Scene:)',play[[i]])

  #use the same method as A to extract body of each play and save them in body list
  body[[i]] <- unlist(play[[i]][ as.integer(num_bodyst[i]) : num_End])

  #use playtitle function to find title
  title[i] <- playtitle(i)
}


year <- as.numeric(year)
#output
meta_data <- data.frame(year, title , num_Scene, num_Acts)
head(body[[1]])

## [1] "SCENE:"
## [2] "Rousillon; Paris; Florence; Marseilles"
## [3] ""
## [4] ""
## [5] "ACT I. SCENE 1."
## [6] "Rousillon. The COUNT'S palace"

tail(body[[1]])

## [1] "    Ours be your patience then, and yours our parts;"
## [2] "    Your gentle hands lend us, and take our hearts."
## [3] "                                                 Exeunt omnes"
## [4] ""
## [5] ""
## [6] "THE END"

print(meta_data)

##    year                                  title num_Scene num_Acts
## 1  1603                ALLS WELL THAT ENDS WELL        23        5
## 2  1607     THE TRAGEDY OF ANTONY AND CLEOPATRA        42        5
## 3  1601                           AS YOU LIKE IT        22        5
## 4  1593                     THE COMEDY OF ERRORS        11        5
```

```
## 5  1608                THE TRAGEDY OF CORIOLANUS        29        5
## 6  1609                                  CYMBELINE        27        5
## 7  1604 THE TRAGEDY OF HAMLET, PRINCE OF DENMARK        20        5
## 8  1598  THE FIRST PART OF KING HENRY THE FOURTH        19        5
## 9  1598             SECOND PART OF KING HENRY IV        19        5
## 10 1599         THE LIFE OF KING HENRY THE FIFTH        23        5
## 11 1592        THE FIRST PART OF HENRY THE SIXTH        27        5
## 12 1591  THE SECOND PART OF KING HENRY THE SIXTH        24        5
## 13 1591   THE THIRD PART OF KING HENRY THE SIXTH        28        5
## 14 1611                    KING HENRY THE EIGHTH        17        5
## 15 1597                                KING JOHN        16        5
## 16 1599             THE TRAGEDY OF JULIUS CAESAR        18        5
## 17 1606                 THE TRAGEDY OF KING LEAR        26        5
## 18 1595                     LOVE'S LABOUR'S LOST         9        4
## 19 1606                   THE TRAGEDY OF MACBETH        29        5
## 20 1605                      MEASURE FOR MEASURE        17        5
## 21 1597                   THE MERCHANT OF VENICE        20        5
## 22 1601                THE MERRY WIVES OF WINDSOR        23        5
## 23 1596                 A MIDSUMMER NIGHT'S DREAM         9        5
## 24 1599                    MUCH ADO ABOUT NOTHING        16        5
## 25 1605   THE TRAGEDY OF OTHELLO, MOOR OF VENICE        15        5
## 26 1596                  KING RICHARD THE SECOND        19        5
## 27 1593                        KING RICHARD III        25        5
## 28 1595          THE TRAGEDY OF ROMEO AND JULIET        24        5
## 29 1594                  THE TAMING OF THE SHREW        14        6
## 30 1612                              THE TEMPEST         9        5
## 31 1608              THE LIFE OF TIMON OF ATHENS        17        5
## 32 1594          THE TRAGEDY OF TITUS ANDRONICUS        14        5
## 33 1602      THE HISTORY OF TROILUS AND CRESSIDA        24        5
## 34 1602         TWELFTH NIGHT; OR, WHAT YOU WILL        18        5
## 35 1595              THE TWO GENTLEMEN OF VERONA        20        5
## 36 1611                    THE WINTER'S TALE        15        5
```

Problem 2

c)

This part is to extract actual spoken text by characters. The method I use is: first, find the Character at the beginning of each dialogue and get the number of line. Second, find the following spoken text between the two line I found in first. and paste them togethor and save them.

```
# This function extract the number of line of the begining of each character's talk
# based on the Character name. and then based on the pattern to find the following
# spoken text between the two lines. paste them to one chunk
characterspoken <- function(j){

  num_Char <- grep('(^\\s+[[:upper:]]+([ ][A-Z]+)*\\.\\s*\'?[[:upper:]][^A-Z]|^\\s{2}([1-9]\\.)?[A-Z][a-
  spokentext <- list()
  i=1
  for(i in seq(length(num_Char)-1)){
    dlog <- play[[j]][num_Char[i]: as.integer(num_Char[i+1]-1)]
    followsent <- grep('^\\s{2,8}[^ ]', dlog)
    spokentext[[i]] <- paste(dlog[followsent], collapse = ' ')
```

```
  }
  return(spokentext)
}

chunk <- list()

#run the function for each play
for(j in seq(length(play))){
  chunk[[j]] <- characterspoken(j)
}

head(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;     but I must attend his Ma
##
## [[3]]
## [1] "  LAFEU. You shall find of the King a husband, madam; you, sir, a     father. He that so general
##
## [[4]]
## [1] "  COUNTESS. What hope is there of his Majesty's amendment?"
##
## [[5]]
## [1] "  LAFEU. He hath abandon'd his physicians, madam; under whose     practices he hath persecuted t
##
## [[6]]
## [1] "  COUNTESS. This young gentlewoman had a father- O, that 'had,' how     sad a passage 'tis!-whos
```

Problem 2

d

After extract spoken chunks of each play, we can find number of sentence, number of characters, number of chunks, number of words, number of unique words, number of average words. In this part, I used lapply function.

```
#4)
#a)#b)#c)#d)

#This function is for extracting  character's name from each chunk.
extractfunction <- function(playnum){
  All_spoken_start <- grep('(^\\s+[[:upper:]]+([ ][A-Z]+)*\\.\\s*\'?[[:upper:]][^A-Z]|^\\s{2}[A-Z][a-z]+
  All_Char_name <- str_extract_all(All_spoken_start, '^\\s*[[:upper:]]+([ ][A-Z]+)*\\.|^\\s{2}[A-Z][a-z]
  All_Char_name <- str_extract_all(All_Char_name, '[[:upper:]]+([ ][A-Z]+)*\\.|[A-Z][a-z]+\\.')
  return(unlist(All_Char_name))
}

Num_Charactor <- 1:36
Num_Chunks <- 1:36
```

```
Num_Scentence <- 1:36
Num_word <- 1:36
Ave_word <- 1:36
Num_unique_word <-1:36

# run for each play
for(i in seq(length(play))){

  #count the number of charactor
  Num_Charactor[i] <- length(unique(lapply(chunk[[i]], extractfunction)))

  #count the number of chunks
  Num_Chunks[i] <- length(chunk[[i]])

  #count the number of sentence. I split the charactor string based on '.', '!', '?'
  Num_Scentence[i] <-  length(unlist(lapply(chunk[[i]], strsplit, split = "[.!?]"))) - Num_Chunks[i]

  #count the number of words. I extract all the words first (eg. Mengying's count as 1 word. "I'll" cou
  Num_word[i] <- length(unlist(lapply(chunk[[i]], str_extract_all, pattern = '[A-Z:a-z]+(\'[a-z])?')))#

  #calculate average words. total words/ number of chunks
  Ave_word[i] <- Num_word[i]/Num_Chunks[i]

  #count unique words. extract all the word , let all of themm become uppercase
  #and use unique function to find the unique words
  Num_unique_word[i] <- length(unique(toupper(unlist(lapply(chunk[[i]], str_extract_all, pattern = '[A-
}

  #create a dataframe contain all of the values
  table <- data.frame(year, num_Acts, num_Scene, Num_Charactor, Num_Chunks, Num_word, Num_Scentence, Ave
  print(table)
```

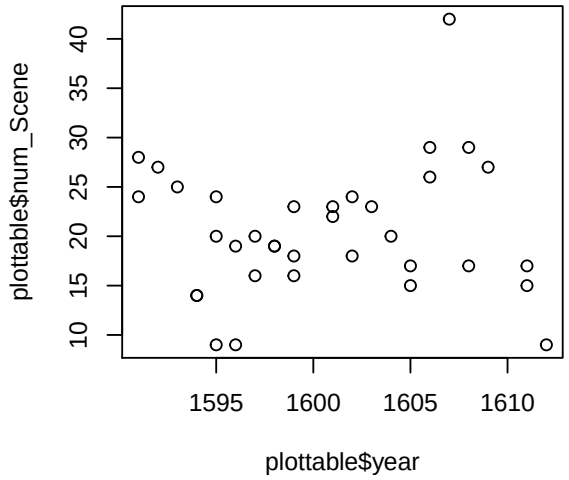
```
##    year num_Acts num_Scene Num_Charactor Num_Chunks Num_word Num_Scentence
## 1  1603        5        23            24        920    23900          1738
## 2  1607        5        42            60       1150    25602          2410
## 3  1601        5        22            26        795    21645          1528
## 4  1593        5        11             1          1     8240           539
## 5  1608        5        29            63       1091    28320          2138
## 6  1609        5        27            39        817    28004          2238
## 7  1604        5        20            31       1082    30836          2646
## 8  1598        5        19            32        730    25231          1987
## 9  1598        5        19            50        885    26785          1972
## 10 1599        5        23            49        738    26643          1454
## 11 1592        5        27            52        624    21759          1482
## 12 1591        5        24            65        755    25882          1695
## 13 1591        5        28            46        791    24931          1681
## 14 1611        5        17            48        682    24738          1613
## 15 1597        5        16            27        536    21454          1248
## 16 1599        5        18            48        781    20213          1725
## 17 1606        5        26            23       1036    26340          2674
## 18 1595        4         9            20       1016    22324          1848
## 19 1606        5        29            42        624    17548          1497
## 20 1605        5        17            22        855    22407          1749
## 21 1597        5        20            24        618    21661          1376
```

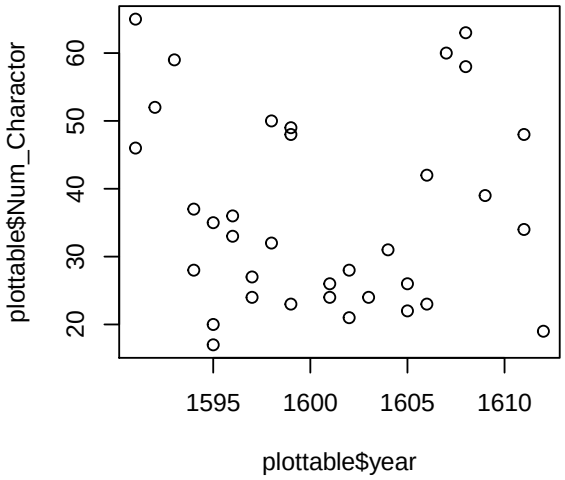
```
## 22 1601        5        23            24        805    23106           2358
## 23 1596        5         9            33        494    15793           1144
## 24 1599        5        16            23        943    21969           1926
## 25 1605        5        15            26       1157    27288           2576
## 26 1596        5        19            36        542    22812           1355
## 27 1593        5        25            59       1051    30254           2166
## 28 1595        5        24            35        803    25243           2361
## 29 1594        6        14            37        868    21698           1703
## 30 1612        5         9            19        616    16926           1317
## 31 1608        5        17            58        746    19089           1702
## 32 1594        5        14            28        552    20837           1342
## 33 1602        5        24            28       1118    26895           2339
## 34 1602        5        18            21        892    20401           1718
## 35 1595        5        20            17        837    17976           1432
## 36 1611        5        15            34        725    25632           1743
##      Ave_word Num_unique_word
## 1    25.97826            3556
## 2    22.26261            3975
## 3    27.22642            3197
## 4  8240.00000            1851
## 5    25.95784            4068
## 6    34.27662            4238
## 7    28.49908            4706
## 8    34.56301            3867
## 9    30.26554            4092
## 10   36.10163            4592
## 11   34.87019            3875
## 12   34.28079            4133
## 13   31.51833            3619
## 14   36.27273            3665
## 15   40.02612            3614
## 16   25.88092            2896
## 17   25.42471            4103
## 18   21.97244            3738
## 19   28.12179            3341
## 20   26.20702            3348
## 21   35.05016            3279
## 22   28.70311            3298
## 23   31.96964            2845
## 24   23.29692            3006
## 25   23.58513            3800
## 26   42.08856            3709
## 27   28.78592            4120
## 28   31.43587            3729
## 29   24.99770            3286
## 30   27.47727            3136
## 31   25.58847            3325
## 32   37.74819            3419
## 33   24.05635            4222
## 34   22.87108            3123
## 35   21.47670            2741
## 36   35.35448            3896
```

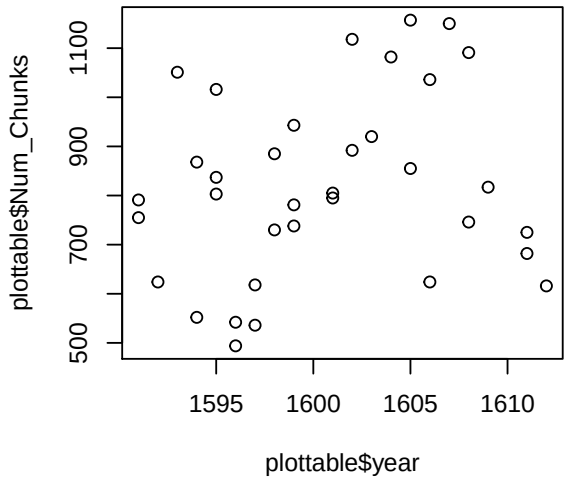
Problem 2

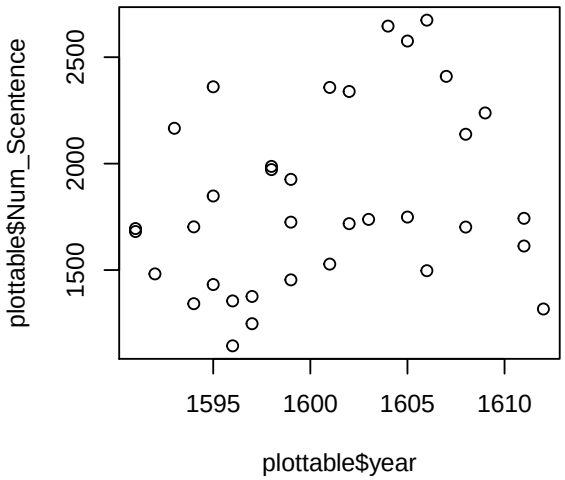
e)

```
#e)
  par(mfrow = c(2,2))
  # remove the fourth, because the fourth is not correct
  plottable <- table[c(1:3,5:36),]
  plot(plottable$year, plottable$num_Scene)
  plot(plottable$year, plottable$Num_Charactor)
  plot(plottable$year, plottable$Num_Chunks)
  plot(plottable$year, plottable$Num_Scentence)
```

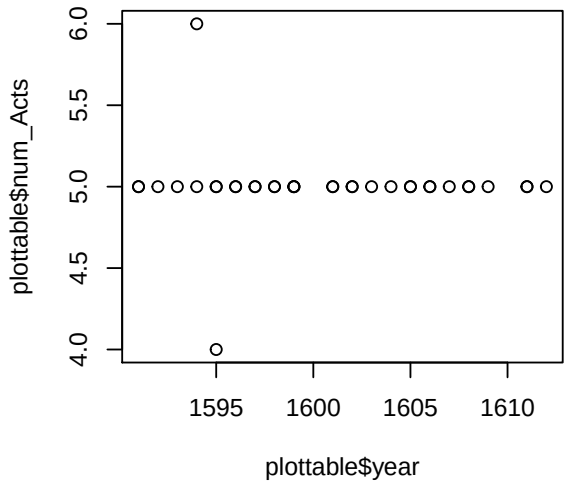


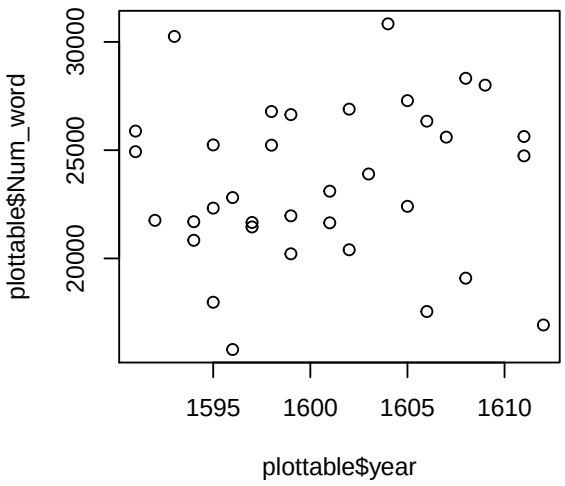


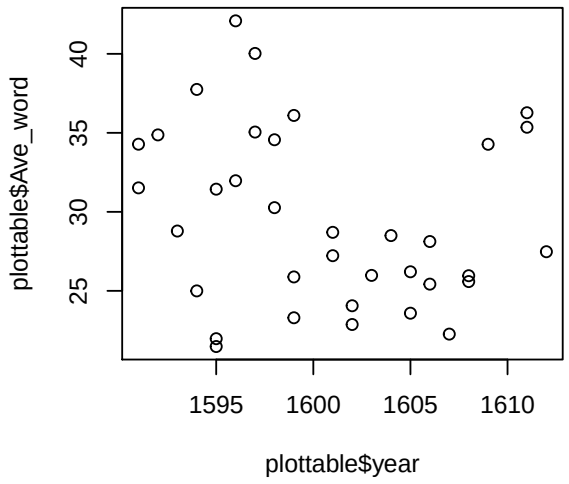


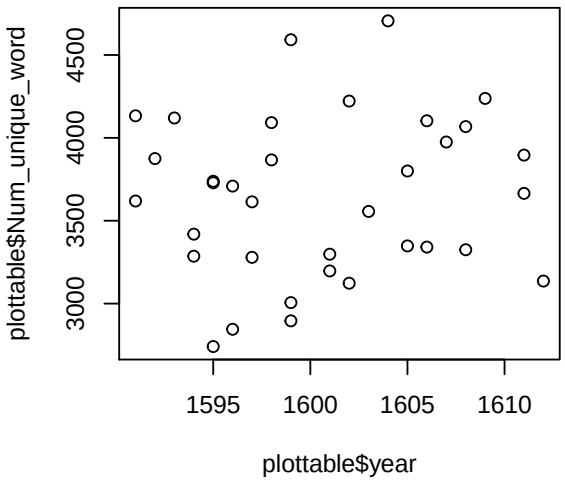


```
plot(plottable$year, plottable$num_Acts)
plot(plottable$year, plottable$Num_word)
plot(plottable$year, plottable$Ave_word)
plot(plottable$year, plottable$Num_unique_word)
```









Besides the Act are remained the same overtime, number of word have a not very obevious pattern going down a little bit, rest of the graphs, I cannot detect any pattern. Therefore, we cannot conclude anything.

Problem 3

1.1 a)

This part I define Shakespeare's plays class. The class name is Shakespeare.

```
library(methods)
  setClass("Shakespeare", representation(
    #The field of this class are year, title, Scene, act, author and body
    #the kind of each field are decribe in the following code:
    year = "numeric",
    title = "character",
    Scene = "vector",
    act = "integer",
    author = "character",
    body = "list"

    #prototype =
    )
  )
    #this part is used to check if the particular field is valid or not
  setValidity("Shakespeare",
          function(object){
            if(!(object@year) > 1580 && object@year < 1620)
              return("error:: Invalid year entered. Year of the play must between 1580 and 1620")
            if(grepl('(Shakespeare|shakespeare)',object@author))
              return("error:: Author of the play has to be Shakespeare")
            return(TRUE)
          }
  )
```

There are lots of methods Shakespeare could have. I set two method at here. The first is for counting the total number of words in the play. The second is for creating a information vector based on the object. The vector contain the year, title and number of Act. The method also could be counting the total number of characters as we did before, the function in the method could be similar with the 'extractfunction' function I created above.

Problem 3

b)

```
#b)
setGeneric("numberofwords", function(objects,...){
  standardGeneric("numberofwords")
})
numberofwords.Shakespeare <- function(object){
  num_words <- length(unlist(lapply(object@body, str_extract_all, pattern = '[A-Z:a-z]+(\'[a-z])?')))
  cat(object@author,"'s ",objec@title,"has", num_words, "words")
}
setMethod(numberofwords, signature = c("Shakespeare"),
          definition = numberofwords.Shakespeare)
#the input argument in this method should be the object belong to the class
#the output is print the a string with number of total words
# no field will be modified

playvector.Shakespeare <- function(object){
  playvector <- 1:3
  playvector[1] <- object@title
```

```
  playvector[2] <- object@year
  playvector[3] <- objecactt@
  cat(playvector)
}
setMethod(playvector, signature = c("Shakespeare"),
          definition = playvector.Shakespeare)
#the input argument in this method should be the object belong to the class
#the output is print the vector with number of total words
# no field will be modified

play1 <- new("Shakespeare", year = 1603,  title = "ALLS WELL THAT ENDS WELL", Scene = 23, chunk = 920 )
```