```
> library(knitr) # need this for opts_chunk command
> opts_chunk$set(fig.width = 5, fig.height = 5)
```

Problem Set 3

Yue Hu

Sept 2017

1 problem 2

Did all process and calculations with apply or sapply, no loops. When calculating average number of words per chunk used a nested sapply.

```
> library(XML)
> library(stringr)
> ## download and read the whole txt, and maitain the structure of lines
> # download.file("http://www.gutenberg.org/cache/epub/100/pg100.txt", "page100.txt")
> txt <- scan(file="page100.txt", what = "character", sep="\n")
> #constent formating
> txt <- gsub("scene I\\.", "scene 1\\.", txt, ignore.case = TRUE)
> txt <- gsub("ACT I\\.", "act 1\\.", txt, ignore.case = TRUE)
> # grep the index of starts of plays(a year of 4 digit), and index of end with "THE END"
> # Then use the list of index to construct a list of plays using appply.
> start <- grep( "^[[:digit:]]{4}", txt)
> end = grep("THE END", txt)
> exstart <- start[2:(length(start)-1)]
> exend <- end[2:(length(end)-1)]
> index <- data.frame(s = exstart, e = exend)
> allplays <- apply(index,1,  function(x) {txt[x[1]:x[2]]})
> ## alternative using for loop to construct list of all plays
> # allplays <- list()
> # total <- length(start)
> # for (i in 2:(total-1)) {
> #   play <- txt[start[i]:end[i]]
> #   allplays[[i-1]] <- play
> # }
>
>
>
> # cal num of plays
> numofplays <- length(allplays)
> cat(numofplays)
```

```
36
```

```
> # extract years (The first line of each play), and names( second line), and body(from 4th line onwards
> years <- sapply(allplays, '[[', 1)
> names <- sapply(allplays, '[[', 2)
> body <- sapply(allplays, function(x) {x[-(1:3)]})
> #Every act has at least a scene 1, grep them to count num of acts; TRUE is to ignore cases
> acts<- sapply(body, function(x) {grep(regex("scene 1\\.", TRUE), x, ignore.case = TRUE)})
> acts[[4]] <- c(1,2,3,4,5)
```

```
> acts[[30]] <- c(1,2,3,4,5)
> nacts <- sapply(acts, function(x) {length(x)})
> # count number of scenes, a pattern of scene then anything then a dot
> scenes <- sapply(body, function(x) {grep(regex("scene .*\\.$", TRUE), x, ignore.case = TRUE)})
> # for special case in No.4 and No.30 where scenen doesn't end with .
> scenes[[4]] <- grep(regex("scene ", TRUE), body[[30]], ignore.case = TRUE)
> scenes[[30]] <- grep(regex("scene ", TRUE), body[[30]], ignore.case = TRUE)
> nscenes <- sapply(scenes, function(x) {length(x)})
> # merge into dataframe
> df <- data.frame(year = years, titile = names, nacts = nacts, nscenes = nscenes)
> df
```

	year	titile	nacts	nscenes
1	1603	ALLS WELL THAT ENDS WELL	5	23
2	1607	THE TRAGEDY OF ANTONY AND CLEOPATRA	5	41
3	1601	AS YOU LIKE IT	5	22
4	1593	THE COMEDY OF ERRORS	5	9
5	1608	THE TRAGEDY OF CORIOLANUS	5	29
6	1609	CYMBELINE	5	27
7	1604	THE TRAGEDY OF HAMLET, PRINCE OF DENMARK	5	20
8	1598	THE FIRST PART OF KING HENRY THE FOURTH	5	19
9	1598	SECOND PART OF KING HENRY IV	5	19
10	1599	THE LIFE OF KING HENRY THE FIFTH	5	23
11	1592	THE FIRST PART OF HENRY THE SIXTH	5	27
12	1591	THE SECOND PART OF KING HENRY THE SIXTH	5	24
13	1591	THE THIRD PART OF KING HENRY THE SIXTH	5	28
14	1611	KING HENRY THE EIGHTH	5	17
15	1597	KING JOHN	3	14
16	1599	THE TRAGEDY OF JULIUS CAESAR	5	18
17	1606	THE TRAGEDY OF KING LEAR	5	26
18	1595	LOVE'S LABOUR'S LOST	4	10
19	1606	THE TRAGEDY OF MACBETH	5	28
20	1605	MEASURE FOR MEASURE	5	17
21	1597	THE MERCHANT OF VENICE	5	20
22	1601	THE MERRY WIVES OF WINDSOR	5	22
23	1596	A MIDSUMMER NIGHT'S DREAM	5	9
24	1599	MUCH ADO ABOUT NOTHING	5	16
25	1605	THE TRAGEDY OF OTHELLO, MOOR OF VENICE	5	15
26	1596	KING RICHARD THE SECOND	5	19
27	1593	KING RICHARD III	5	24
28	1595	THE TRAGEDY OF ROMEO AND JULIET	5	25
29	1594	THE TAMING OF THE SHREW	6	14
30	1612	THE TEMPEST	5	9
31	1608	THE LIFE OF TIMON OF ATHENS	5	17
32	1594	THE TRAGEDY OF TITUS ANDRONICUS	5	14
33	1602	THE HISTORY OF TROILUS AND CRESSIDA	5	24
34	1602	TWELFTH NIGHT; OR, WHAT YOU WILL	5	18
35	1595	THE TWO GENTLEMEN OF VERONA	5	20
36	1611	THE WINTER'S TALE	5	15

```
> ##2c
>
> ## start from "act 1. scene 1" to precisely extract parts including chunks, excludes headings, copy ri
> bodystart <- grep(regex("act 1\\. scene 1", TRUE), txt, ignore.case = TRUE)
> bodyend <- end[2:(length(end)-1)]
```

```
> index <- data.frame(s = bodystart, e = bodyend)
> bodys <- apply(index,1,  function(x) {txt[x[1]:x[2]]})
> # two indentations indecates a new speaker, substitute them with a $ sign
> bodys <- sapply(bodys, function(x) {gsub("^[[:space:]]{2}(\\w+\\..*)", "\\$\\1", x)})
> # four indentations then a word then anything(word or spaces) indecates a following chunk, substitute
> bodys <- sapply(bodys, function(x) {gsub("^[[:space:]]{4}\\w(.*)", "\\@\\1", x)})
> # paste them form several lines into a long string and mark original lines with '\n' at each line end
> bodys <- sapply(bodys, function(x) {paste(x, collapse = '\n')})
> # each chunk with speaker is a pattern of "$blahblah\n" followed by "@blahblah\n"zero or more times, e
> chunks <- sapply(bodys, function(x) {str_extract_all(x,"\\$(.*?)\n(\\@(.*?)\n)*")})
> #remove all indecators ($ @ \n)
> chunks <- sapply(chunks, function(x) {gsub("(\\@|\\$|\n)", '', x)})
> # split speacker and dialogue by ". ". Set n=2 so creat 2 elements at most.
> chunksum <- sapply(chunks, function(x) {str_split(x, "\\. ", n = 2)})
> ##d
>
> # separate speakers and dialogues into different lists
> speakers <- sapply(chunks, function(x) {gsub("(\\w+)\\.(.*)", "\\1", x)})
> chunktxt <- sapply(chunks, function(x) {gsub("(\\w+)\\.(.*)", "\\2", x)})
> # find uniqie speakers and calculate the number
> uspe <- sapply(speakers,unique)
> nspe <- sapply(uspe,length )
> # Calculate the number of chunks
> nchun <- sapply(chunktxt,length )
> # paste all sentences of chunks, then calculate sentences, indicated by num. or words followed by .?!
> sumsentence <- sapply(chunktxt, function(x) paste(x, collapse = ' ') )
> nsent <- sapply(sumsentence, function(x) {length(str_extract_all(x, "(\\d|\\w)(\\.|\\!|\\?)")[[1]])})
> # extract words (at least one character fllowed by one or more ' then one or more words.)
> words <- sapply(sumsentence, function(t) {str_extract_all(t, "\\w+\\'?\\w*")})
> nwords <- sapply(words, length)
> #split text in each chunk to single words
> chunkword <- sapply(chunktxt, function(t) {str_extract_all(t, "\\w+\\'?\\w*")})
> #count the length. use a nested sapply since each element of play is itself a list
> nchunkword <- sapply(chunkword, function(x) {sapply(x, length)})
> # count the mean of each element list
> aveword <- sapply(nchunkword, mean)
> # find uniqie words and calculate the number
> uwords <- sapply(words,unique)
> nuwords <- sapply(uwords,length )
> ## e
> nyear <- as.numeric(years)
> nspe <- as.numeric(nspe)
> nchun <- as.numeric(nchun)
> summary <- data.frame(year = nyear, acts_num = nacts, scene_num = nscenes, speaker_num = nspe, chunk_n
+                       = nchun)
> # sort by year
> summarys <- summary[order(summary[,1]),]
> summary

   year acts_num scene_num speaker_num chunk_num
1  1603        5        23          16       777
2  1607        5        41          45      1112
3  1601        5        22          19       746
4  1593        5         9           4         4
```

```
5  1608        5        29          29        887
6  1609        5        27          27        792
7  1604        5        20          34       1125
8  1598        5        19          35        757
9  1598        5        19          37        776
10 1599        5        23          44        540
11 1592        5        27          44        605
12 1591        5        24          49        676
13 1591        5        28          33        512
14 1611        5        17          41        575
15 1597        3        14          21        393
16 1599        5        18          39        711
17 1606        5        26          25       1061
18 1595        4        10          17        938
19 1606        5        28          29        477
20 1605        5        17          21        873
21 1597        5        20          22        608
22 1601        5        22          24       1010
23 1596        5         9          31        501
24 1599        5        16          25        975
25 1605        5        15          19       1154
26 1596        5        19          32        451
27 1593        5        24          45        696
28 1595        5        25          37        836
29 1594        6        14          24        812
30 1612        5         9          19        641
31 1608        5        17          33        596
32 1594        5        14          23        558
33 1602        5        24          29       1141
34 1602        5        18          17        761
35 1595        5        20          14        828
36 1611        5        15          26        703
```

```
> # plot
> plot.ts(summarys)
```

2 Problem 3

2.1 a)

1)class: Play fields: name a string charactor of the play tittle year a numeric of the year of the play body a list of Body Objects 2čl'class: Body fields: speakers a list of the speakers of each chunk in sequece chunks a list of the dialogue chunks in sequece

2.2 b)

Methods for Plays: '<-' input is the url link or path of the whole txt and the output is a list of objects in Play class by processing the whole txt. print print the name , year, and the first 10 elements of the body printchunk input is input is Play object and a list of numbers. output is the elements of it's bod y(i.e. speakers and chunks) by number index playSummary input is Play object and out put is a data frame with column $(\text{num}_of_acts, num_of_scenes, num_of_speakers) bodySummary input is Play object and output is a data frame with columns (num_of_c h$