Problem Set 3

Dongyu Lang
SID: 24174288

September, 28, 2017

1 Question 1 (c)

I read the third article : Wilson et.al. (http://arxiv.org/pdf/1210.0530v3.pdf)

Comment: The authors wrote down many effective tips. From my personal experience, there is also one side effect of pair programming, that is, your partner may not devotes as much as you do, and can rely on you too much; thus, in this situation, it is best for partners to do their parts separately to increase the efficiency.

Question: In some situation, it is hard for us to break our code into 1-hour-long code. Also, for those who are not good at writing code, it is often that they barely finish anything in this one hour period. So are there some recommendations in these situations?

2 Question 2 (a)

I frist read the data using readLines. Then I greped the index numbers for "THE END". And, there is a play between every "THE END". In addition, I deleted everything between "<< "*and*" >>" using strsplit. Then I created a list to store the plays.

```
## Read the txt file
rawdata = readLines("http://www.gutenberg.org/cache/epub/100/pg100.txt")

## Grep the index number for "THE END"
THEEND_index = grep("^THE END", rawdata)
plays = c(rep(NA, length(THEEND_index)-1))

for (i in 1:(length(plays)-1)) {
  ## Get the content between the ith index and the i+1th index
  ith_plays = paste(rawdata[(THEEND_index[i]+1):(THEEND_index[i+1])],
                    collapse = "\n")
  ## Delete everything inside "<<" and ">>"
  plays[i] = paste(strsplit(ith_plays, "<<[^>]*>>")[[1]], collapse = "\n")
```

```
}
plays = plays[1:(length(plays)-1)]

## Return the number of plays
length(plays)

## [1] 35
```

3 Question 2 (b)

I used regular expression to find the year, title, number of acts and number of scenes in each play. And I also extracted the body of each play using the method similarly to that in part (a).

```
## Since I deleted everthing inside "<< >>", thus, the first number
## appeared in each play is the year. So I grepped the first
## 4 digits of each play.
years = as.numeric(unlist(lapply(plays,
        function(x) str_match(x,"[0-9]{4}"))))

## All the titles has the same pattern, that is between the year
## and by Shakespeare, so I used the sub to get the title bewteen
## them in each play.
titles = unlist(lapply(plays,
    function(x) sub(".*[0-9]{4} *(.*?) *by.*", "\\1", x)))

## I first grepped the last ACT using regex "ACT.*\n" and tail.
## Then I used regex to get the Roman number or the number of
## the last act. The regular expression also allowed me to grep
## some inconsisting formats.
num_act = unlist(lapply(plays,
  function(x) as.numeric(as.roman(sub("ACT[ ]*[_]*([0-9A-Z]?)[.]*[ ]*[|]*.*",
  "\\1", tail(str_match_all(x, "ACT.*\n")[[1]],1))))))

## I grepped all the lines that contained Scene or SCENE,
## and get the number of results I got.
num_scene = unlist(lapply(plays,
  function(x) length(str_match_all(x,"[S][C|c][E|e][N|n][E|e].*\n")[[1]])))

## I noticed that when I strsplit the "<< >>", starting
## from the 3rd element of the list, it is the body of
## a play. So I used a similar method as that in part (a)
body = c(rep(NA, length(THEEND_index)-1))
for (i in 1:(length(body)-1)) {
```

```
  ith_body = paste(rawdata[(THEEND_index[i]+1):(THEEND_index[i+1])],
                   collapse = "\n")
  body[i] = paste(strsplit(ith_body, "<<[^>]*>>")[[1]][-(1:2)], collapse = "\n")
}
body = body[1:(length(body)-1)]

## I also created a list that contains all the information.
meta_data = list(year=years,title = titles, num_act = num_act, num_scene = num_scene)
```

4 Question 2 (c)

I noticed that almost all the plays follows the same indentation, except for the fourth one. For the same spoken chunk, each line except the first line has a newline plus four whitespaces to the previous line. Thus, I used this indentation to make each spoken chunk, and separated each spoken chunk using newline. Thus, I only had empty strings, stage lines and spoken chunk left. It was easy to get rid of the empty strings. For stage lines, I noticed that most of them don't have a period at the end, also, for each spoken chunk, it also follows that there is a period after the speaker in each chunk. Thus, I used this to separate the stage lines and spoken chunks. Then I separated speakers and spoken text using the periods after each speakers. For the fourth play, it has a unique indentation, so I used the same method but different regex.

```
index = 1:length(plays)
speaker = list(NA)
text = list(NA)
for (i in index[-4]) {
  ## Deleted the whitespaces in each spoken chunk.
  clear_indentation = gsub("\n    ", " ", body[i])
  ## Separated spoken chunks.
  sep_indentation = str_split(clear_indentation,"\n")
  ## Cleared the stage lines
  clear_stage = sep_indentation[[1]][grep("^[ ]{2}[A-Z]",sep_indentation[[1]])]
  clear_stage = trimws(clear_stage)
  clear_stage = clear_stage[grep("[.]",clear_stage)]
  clear_stage = clear_stage[grep("^[A-Za-z]+[.]",clear_stage)]
  ## Separated the speakers and spoken texts by the periods.
  sep_speaker_lines = unlist(stri_split_fixed(clear_stage, ".", n = 2))
  speaker[[i]] = sep_speaker_lines[seq(1,length(sep_speaker_lines),2)]
  text[[i]] = sep_speaker_lines[seq(2,length(sep_speaker_lines),2)]
}

## Same method as above.
play4_clear_indent = gsub("\n  ", " ", body[4])
```

```
play4_sep_indent = str_split(play4_clear_indent,"\n")
play4_clear_stage = play4_sep_indent[[1]][grep("[A-Z]{4,}[.]",play4_sep_indent[[1]])]
play4_sep_speaker_lines = unlist(stri_split_fixed(play4_clear_stage, ".", n = 2))
speaker[[4]] = play4_sep_speaker_lines[seq(1,length(play4_sep_speaker_lines),2)]
text[[4]] = play4_sep_speaker_lines[seq(2,length(play4_sep_speaker_lines),2)]

## add the speaker and text into meta_data
meta_data = list.append(meta_data, speaker= speaker, text = text)
```

5 Question 2 (d)

```
## The number of unique speakers in each play
unique_speakers = unlist(lapply(speaker,function(x) length(unique(x))))
## The number of spoken chunks in each play
num_chunks = unlist(lapply(text, length))
## The number of sentences in each play
num_sentences = unlist(lapply(text, function(x) sum(str_count(x, "[.!?-]"))))
## Remove punctuation but not apostrophes
replace_punc = lapply(text, function(x) gsub("[^[:alnum:][:space:]']", "", x))
## The number of words spoken in each play
num_words = unlist(sapply(replace_punc,
                function(x) length(unlist(str_match_all(x, "\\S+" )))))
## Average number of words per chunk in each play
ave_words_per_chunk = num_words/num_chunks
## The number of unique words in each play
lower_replace_punc = lapply(replace_punc, tolower)
num_uniq_words = unlist(sapply(lower_replace_punc,function(x)
  length(unique(unlist(
  str_match_all(x, "\\S+" ))))))
```

6 Question 2 (e)

```
## The number of acts
num_act

##  [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 5

## The number of scenes
num_scene

##  [1] 24 43 23 12 30 28 21 20 20 24 28 25 29 18 17 19 27 10 48 21 24 10 18
## [24] 16 20 26 25 15 10 18 15 25 19 21 16
```

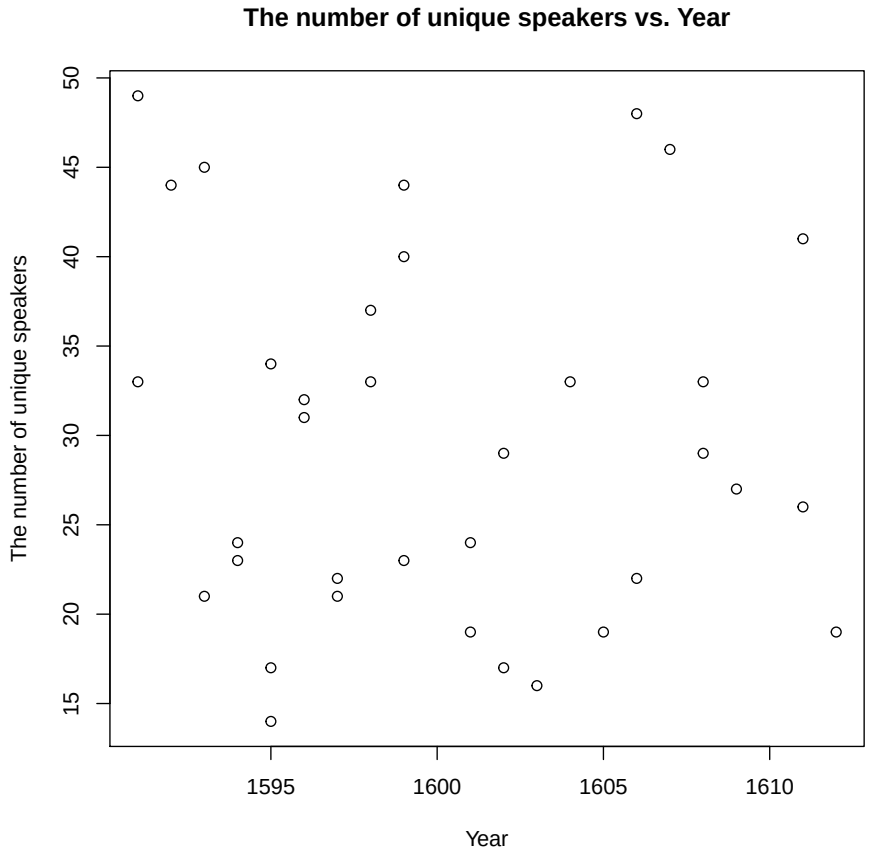
```
## The number of unique speakers
unique_speakers
```

```
##  [1] 16 46 19 21 29 27 33 33 37 44 44 49 33 41 21 40 22 17 48 22 24 31 23
## [24] 19 32 45 34 24 19 33 23 29 17 14 26
```

```
## The number of chunks for each play
num_chunks
```

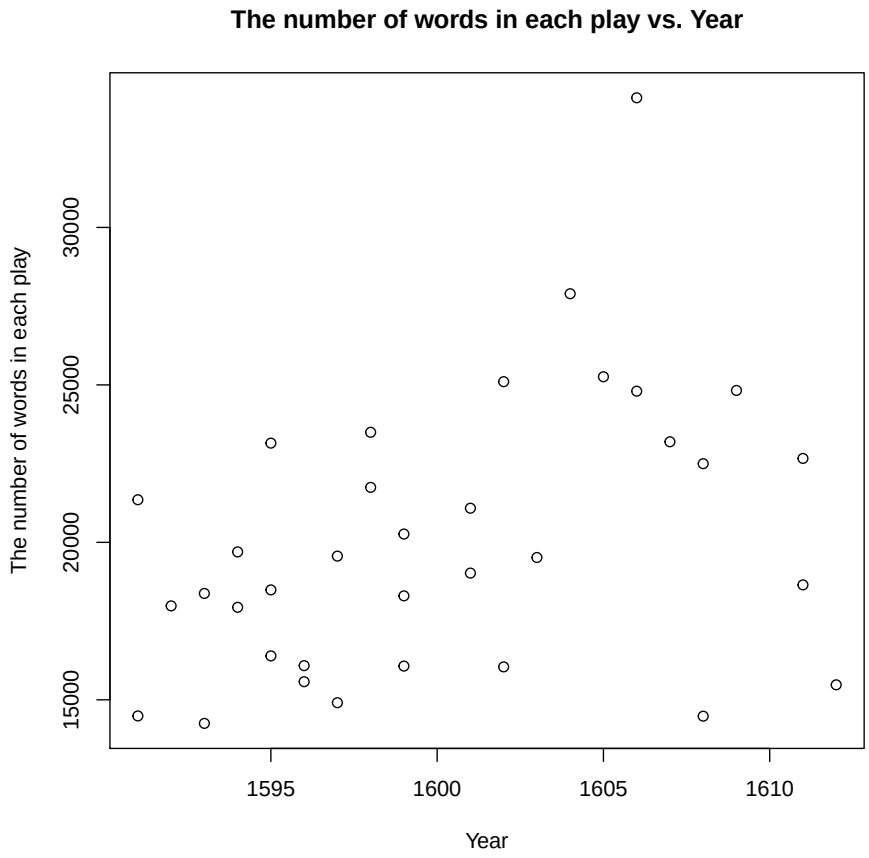
```
##  [1]  777 1113  746  610  887  792 1119  743  776  540  605  676  512  575
## [15]  393  712 1053  938 1350  608 1010  501  955 1154  451  696  812  812
## [29]  641  596  558 1141  761  828  703
```

```
plot(meta_data$year, unique_speakers,
     xlab = "Year", ylab = "The number of unique speakers",
     main = "The number of unique speakers vs. Year")
```



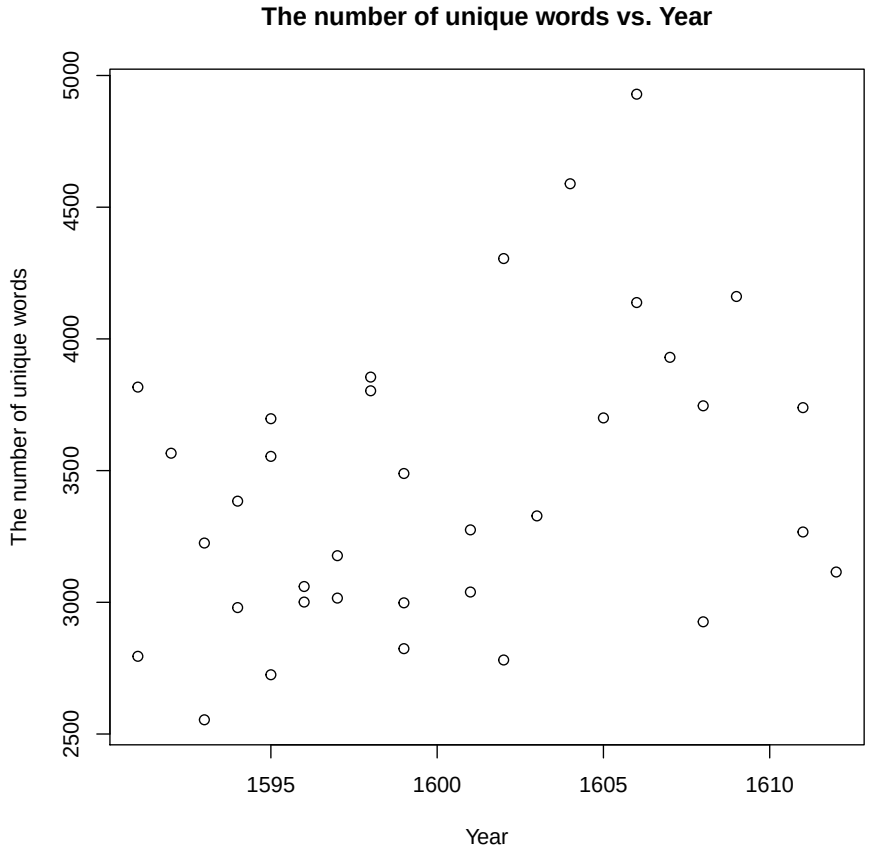
The plot shows that there is no specific pattern for the number of unique speakers over time.

```
plot(meta_data$year, num_words,
     xlab = "Year", ylab = "The number of words in each play",
     main = "The number of words in each play vs. Year")
```



The plot shows that the number of words in his plays actually increased with the course of his writing career. However, in his late writing career, he prefered to write some short plays.

```
plot(meta_data$year, num_uniq_words,
     xlab = "Year", ylab = "The number of unique words",
     main = "The number of unique words vs. Year")
```



It is also interesting that he began to use more unique words when he wrote more plays. Also, in his late writing career, he used less unique words than before.

7 Question 2 (f)

Extra credit: for part (c), I exculded all the stage directions, when I detected that almost all the stage direction don't have their first words in upper-case. In addition, they don't have a period followed the first upper-case words. Thus, I used this finding to perfectly separate stage directions and spoken chunks. In addition, I avoid excluding any plays except for the fourth one.

8 Question 3

The fields should have years when Shakespeare wrote the plays, titles of the plays, the body of the plays. Years should be a 4-digit numberic vector, titles

should be a list that contains characters, and body should be a list. In addition, we should have speakers and texts, which are both lists.

For methods, we should have a method called body_processed, which can separate the body into spoken chunks and furthermore separate the speakers and texts. The input should contain the body of each play, and it should return the speakers and spoken texts.

In addition, we should have a method called related_stats, which can return the statistics in question 2 part (d), the input should contain the speakers and texts, which are returned in method body_processed.

Moreover, we should have a method called make_plots, which can return some plots I did in part (e), the input should be the output of related_stats. And the output for make_plots should be some plots like part(e).