Problem Set 3

Cameron Adams

September 29, 2017

1 On Monday, September 25, section will consist of a discussion of good practices in reproducible research and computing, led by Andrew and Chris. In preparation, please do the following...

I read:

- Wilson et.al. (http://arxiv.org/pdf/1210.0530v3.pdf)
- Millman and Perez (https://github.com/berkeley-stat243/stat243-fall-2014/blob/master/section/millman-perez.pdf)

I try to incorporate good coding practices, reproducible research ideals, etc., into my research work. However, the last stage of research-report production-prevents me from fully incorporating these practices. My lab still mostly uses .doc (or Google docs equivalent) file format to produce and submit research drafts. Do you have advice on how to create elegant/efficient work flow to go from .R/python code to latex typesetting/tables to .doc format without cutting and pasting. This is by far the biggest barrier I've seen to fully incorporating the "R/python computation –¿ completed journal articles" workflow.

I found "Best Practices for Scientific Computing" informative. My code is usually just for me, and is rarely viewed by others. Despite that, I do try to uses wrappers/functions to prevent repeated copy and pasting code. What are recommendations, besides version control for small changes to code. For example, in the first set of analyses I used one set of covariates, and the next week, I want to change those covariates. Is this a scenario where you simply recommend version control, another script (analysis1, analysis2)?

2 DRAMATIS PERSONAE: INSTRUCTOR, STUDENTS, GSI

2.1 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).

```r
#download plays in .txt file
URL   <- "http://www.gutenberg.org/cache/epub/100/pg100.txt"
download <- getURL(URL)
txt <- download[1]

#############
#extract plays from "the sonnets" to "lover's complaint"

#each starts with year published (eg., 1609)
id <- str_extract_all(txt, "\\n[0-9]{4}\\r")[[1]]

#each play also appears to end with "THE END"
```

```r
theEnd <- str_extract_all(txt, "THE END")[[1]]
length(id)==length(theEnd) #there are 38 works in this txt file

## [1] TRUE

##set custom delimter for start of plays
txt<-str_replace_all(txt, "\\n([0-9]{4}[\\r\\n]+)",
                      "\\|\\|\\1")
length(str_extract_all(txt, "\\|\\|")[[1]]) #38 delimeters

## [1] 38

#Set custom delimter for end of plays
txt<-str_replace_all(txt, "THE END", "THE END\\|\\|")
length(str_extract_all(txt, "THE END\\|\\|")[[1]]) #38 delimeters

## [1] 38

#fix inconsistenances and remove extraneous repeated text
txt<-str_replace_all(txt, "ACT [0-9]{1,2}\\.", "ACT I.") #ACT 1 -> ACT I.
txt<-str_replace_all(txt, "Dramatis Personae", "DRAMATIS PERSONAE")
txt <- str_replace_all(txt, "Scene", "SCENE")
txt <- str_replace_all(txt, "Scene:", "SCENE:")
txt=str_replace_all(txt,"<<[\\s[:alnum:][:punct:]]*>>","") #remove copyright

#we've isolated the extranous text with delimeters
plays <- unlist(strsplit(txt, split="\\|\\|")) #odd numbered elements can be removed
plays <- plays[seq(4, (length(plays)-2), 2)] #remove first/last sonnets

#now we have all the plays as elements of a character vector
substr(plays[1], 0, 50)

## [1] "1603\r\n\r\nALLS WELL THAT ENDS WELL\r\n\r\nby William Sha"

substr(plays[2], 0, 50)

## [1] "1607\r\n\r\nTHE TRAGEDY OF ANTONY AND CLEOPATRA\r\n\r\nby "

substr(plays[length(plays)], 0, 50)

## [1] "1611\r\n\r\nTHE WINTER'S TALE\r\n\r\nby William Shakespear"

#removing the 4th play, "THE COMEDY OF ERRORS" and
# and "THE SECOND PART OF KING HENRY THE SIXTH" I can't process properly
plays <- plays[-c(4,12)]
```

2.2 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.

```r
############
```

```r
#extract metadata
#titles
title <- sapply(1:length(plays), function(x) str_extract(plays[x], "[^0-9\\r\\n]+"))

#year
year <- sapply(1:length(plays), function(x) str_extract(plays[x], "^[0-9]{4}"))

#num act per play
acts <- sapply(1:length(plays), function(x)
  length(unique(c(str_extract_all(plays[x], "ACT [IV]{0,3}\\.", simplify = T)))))

#num scenes per play
scenes <- sapply(1:length(plays), function(x)
  length(c(str_extract_all(plays[x], "SCENE\\s[0-9IV]{1,3}", simplify = T))))
#str_extract_all(plays[3], "SCENE\\s[0-9IV]{1,3}\\.", simplify = T)

#combine and view
metaData <- list(year = year, title = title, acts = acts, scenes = scenes)

############
#extract body of play
body=sapply(1:length(plays), function(x)
  str_extract_all(plays[x],"SCENE(:|\\.)([[:alnum:][:punct:]\\s]+)",simplify=T))
```

2.3 Extract the actual text spoken by the characters into your object.

```r
############
#extract chunks
extractChunks <- function(x) {

    #extract speaking chunks
    chunks <- str_extract_all(body[x], "(?<=\\r\\n\\s{2})([^%]+?)(\\r\\n(?!\\s{4}))",
                              simplify = T)

    #remove act, scene, stage directions, etc.
    chunks <- chunks[-grep("(ACT|SCENE|Enter|Exit|ENTER|EXIT)", chunks)]

    return(chunks) #return value
}
chunks <- sapply(1:length(body), FUN = extractChunks) #chunks for each play
```

Strategy for extracting chunks was to use lookaheads and lookbehind to look for indents at beginning and end of speaking chunks. They start and end with carriage return, new line, and 2 space indent. Continuing indents are 4 spaces. Then removed ACT, and scene information, along with stage directions.

```r
#extract actors
extractActors <-  function(x) {

    #search for JOHN., KING JOHN., Rom. , etc
    actor_pattern <- paste0("([A-Z]{1}[a-z]+?\\.{1})|",
                            "([A-Z]{1}[a-z]+?\\s{1}[A-Z]{1}[a-z]+?\\.{1})|",
                            "([A-Z ]{3,}?\\.{1})")
    actors <-
        unique(str_replace(
            str_extract(chunks[[x]], actor_pattern),"^\\s+", "")) #remove whitespace

    #remove NA's
    actors <- actors[!is.na(actors)]

    #remove unwanted stuff
    bad_actor_pattern <- paste0("^(ACT|I\\.|II|VI|X|",
                                "EPILOGUE|PROLOGUE|ENTER|Enter|",
                                "EXIT|Exit|[[:punct:]])")
    actors <- actors[!grepl(bad_actor_pattern, actors)]
}
actors <- sapply(1:length(chunks), FUN = extractActors)
```

Strategy for extracting actors was to search for single words which end with a period, and the removing obvious non-actor results such as NA, ACT, IV, and things that begin with punctuation.

2.4 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. When extracting words, make sure you remove punctuation such as commas and periods but not apostrophes.

2.4.1 The number of unique speakers.

```
#numer of actors per play
num_actors_play <- unlist(lapply(actors, function(x) length(x)))
```

Length of actor vectors per play gives you these quantities.

2.4.2 The number of spoken chunks.

```
#number of spoken chunks per play
num_spoken_chunks_play <- unlist(lapply(chunks, function(x) length(x)))
```

For number of spoken chunks, I counted the number of elements in each vector of chunks for each play.

2.4.3 The number of sentences and words spoken and average number of words per chunk.

```
#number of setences per play
num_sentences_play <- sapply(1:length(chunks),
       function(x) sum(unlist(lapply(
           str_extract_all(chunks[x],"[//.]+"), #extract all periods
            function(x) length(x)-1 #apply to each list and remove period for speaker
))))
```

For number of sentences, I used periods to define if a sentence occurred in a chunk, and I subtracted one to account for actor name at the start of a line.

```r
#number of total words spoken per play
extractWords <- function(x) { #input vector of chunk for a play

    tmp <-
        sapply(x, function(x)
            str_extract_all(x, "([A-Z]{1}[a-z']+|[a-z']+)")[[1]])
            #takes in vector of chunks for play_n,
            #extracts words with regex, and then counts them
            #and then sum

    #unlist and collapse into single character element per play
    tmp <- paste0(unlist(tmp), collapse = " ")

    #remove unneed stuff and combine into character vector
    tmp <- str_extract_all(tmp, "[A-z']+[^ ']{1}")

    #return charcter vector of words for each play
    return(tmp)
}

#get all words in each play
words_play <- sapply(chunks, extractWords)

#count number of words in each play
num_words_play <- sapply(words_play, function(x) length(x))
```

For total words spoken, I regex'ed for words in each chunk and counted them.

```r
#get num words per chunk per play
avg_num_words_chunk_play <- num_words_play /
                                sapply(chunks, function(x) length(x))
```

For average number of words per chunk, simply divide number of words per play by number of chunks per play.

2.4.4 The number of unique words.

```r
#num. unique words per play
num_uniq_words_play <- sapply(words_play, function(x) length(unique(x)))
```

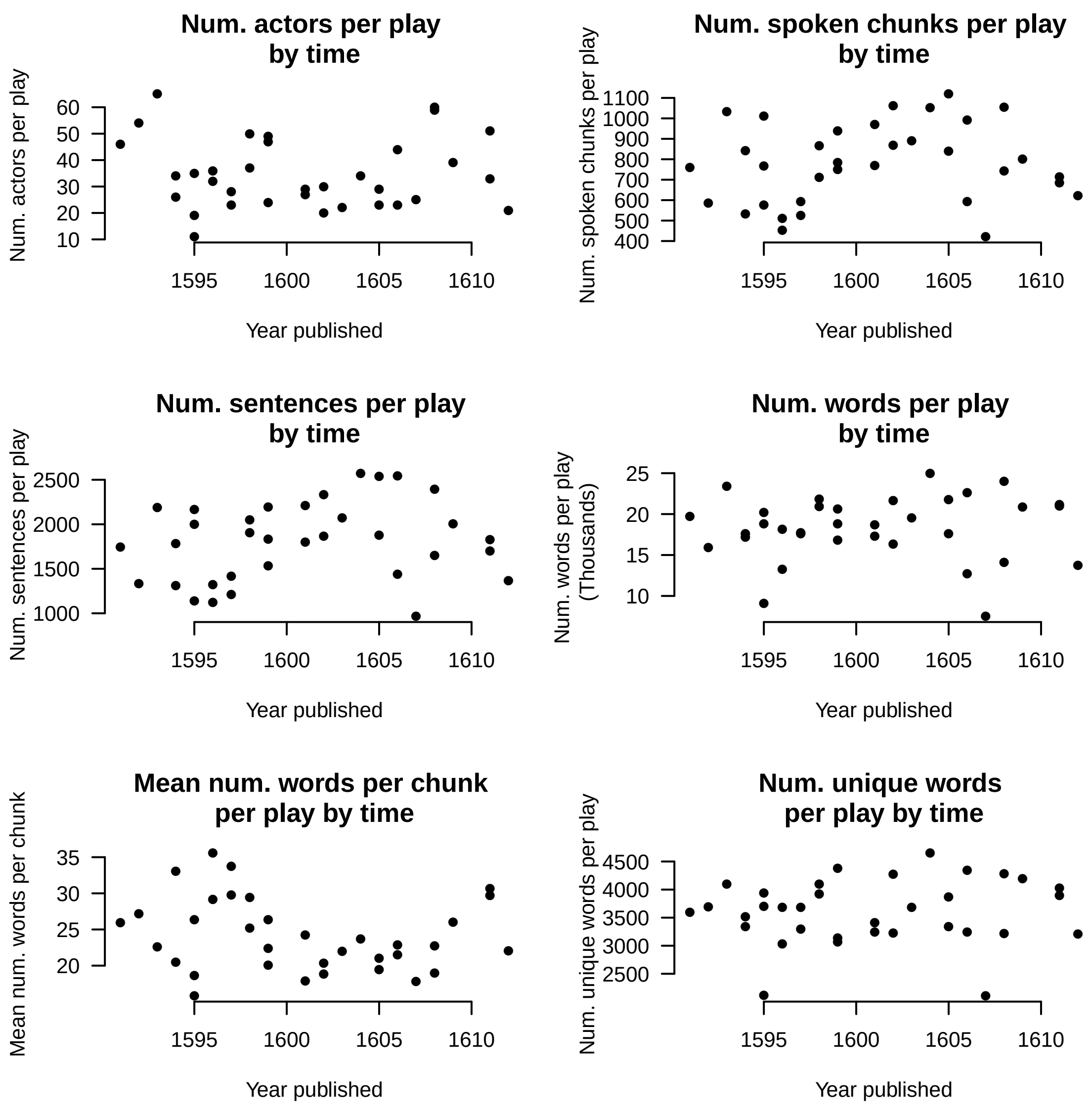
For number of unique words, I used the character vector of words from earlier, removed duplicates, and then counted remaining unique words.

2.5 Plot some of your summary statistics as a function of time...

```r
#plot fxn
pplot <- function(x, y, pch = 16, cex = NULL,
                  xlab = "", ylab = "", xaxt = "n",
                  yaxt = "n", bty = "n", ...) {

  plot(x, y, pch = pch, cex = cex, xlab = xlab, ylab = ylab,
       xaxt = xaxt, yaxt = yaxt, bty = "n",...)
  axis(1)
  axis(2, las = 2)

}
par(mar = c(5, 5, 4, 1), mfrow = c(3, 2)) #set plotting area
#plot characteristics of plays
pplot(x = year[order(year)], y = num_actors_play[order(year)],
      xlab = "Year published", ylab = "Num. actors per play",
      main = "Num. actors per play \nby time")
pplot(x = year[order(year)], y = num_spoken_chunks_play[order(year)],
      xlab = "Year published", ylab = "Num. spoken chunks per play",
      main = "Num. spoken chunks per play \nby time")
pplot(x = year[order(year)], y = num_sentences_play[order(year)],
      xlab = "Year published", ylab = "Num. sentences per play",
      main = "Num. sentences per play \nby time")
pplot(x = year[order(year)], y = num_words_play[order(year)]/1000,
      xlab = "Year published", ylab = "Num. words per play \n (Thousands)",
      main = "Num. words per play \nby time")
pplot(x = year[order(year)], y = avg_num_words_chunk_play[order(year)],
      xlab = "Year published", ylab = "Mean num. words per chunk",
      main = "Mean num. words per chunk \nper play by time")
pplot(x = year[order(year)], y = num_uniq_words_play[order(year)],
      xlab = "Year published", ylab = "Num. unique words per play",
      main = "Num. unique words \nper play by time")
```



```
par(mar = c(5, 4, 4, 1), mfrow = c(1,1))
```

Plots don't indicate any obvious time trends, but the number of spoken chunks, and number of sentences peaked around year 1605, at the same time the number of words per chunk was at its smallest. The number of actors, unique words, and number of words were generally consistently noisy.

```
#combine num. of acts and scenes, unique speakers, chunks for each play
report <- cbind(acts, scenes, num_actors_play, num_spoken_chunks_play)[order(year),]

colnames(report) <- c("Acts", "Scenes", "Actors", "Spoken chunks")
rownames(report) <- substr(title[order(year)], 0, 20) #shorten titles to fit

#print table of characteristics
require(xtable)
tbl <- xtable(report, caption = "Characteristics of Shapkespeare plays extracted using regex",
           comment = F, align = c("l", rep("c", dim(report)[2])))
print(tbl, floating = T, table.placement = "H", comment = F,
    row.names = T, caption.placement = "top", caption.width = "35em")
```

Table 1: Characteristics of Shapkespeare plays extracted using regex

	Acts	Scenes	Actors	Spoken chunks
THE THIRD PART OF KI	5	28	46	760
THE FIRST PART OF HE	5	27	54	585
KING RICHARD III	5	25	65	1034
THE TAMING OF THE SH	4	14	34	841
THE TRAGEDY OF TITUS	5	14	26	532
LOVE'S LABOUR'S LOST	5	9	19	1012
THE TRAGEDY OF ROMEO	5	24	35	768
THE TWO GENTLEMEN OF	5	20	11	575
A MIDSUMMER NIGHT'S	5	9	32	454
KING RICHARD THE SEC	5	19	36	510
KING JOHN	5	16	28	525
THE MERCHANT OF VENI	5	20	23	592
THE FIRST PART OF KI	5	19	37	712
SECOND PART OF KING	5	19	50	866
THE LIFE OF KING HEN	5	23	47	783
THE TRAGEDY OF JULIU	5	18	49	749
MUCH ADO ABOUT NOTHI	5	17	24	938
AS YOU LIKE IT	5	22	27	770
THE MERRY WIVES OF W	4	23	29	970
THE HISTORY OF TROIL	5	24	30	1063
TWELFTH NIGHT; OR, W	5	18	20	868
ALLS WELL THAT ENDS	5	23	22	890
THE TRAGEDY OF HAMLE	4	20	34	1052
MEASURE FOR MEASURE	4	17	23	839
THE TRAGEDY OF OTHEL	5	15	29	1119
THE TRAGEDY OF KING	5	26	23	991
THE TRAGEDY OF MACBE	5	29	44	593
THE TRAGEDY OF ANTON	5	32	25	421
THE TRAGEDY OF CORIO	5	28	60	1055
THE LIFE OF TIMON OF	5	17	59	743
CYMBELINE	5	27	39	801
KING HENRY THE EIGHT	5	17	51	684
THE WINTER'S TALE	5	15	33	713
THE TEMPEST	5	9	21	623

2.6 Extra credit:

3 This problem asks you to design an object-oriented programming (OOP) approach to the Shakespeare analysis of problem 2.

3.1 What are the fields (i.e., member data, slots, etc.) for the class? ...

The fields would be the objects containing parts of each play (or one with the entire plays) and objects with metadata about the plays (title, number of acts, year published, etc.). Going with the suggestion in the question, there could also be an array object which is comprised of the components of each play. And we use these components of each play to extract information about the play, such as number of chunks, actors, etc.

Example of object structure:

- plays (array which contains lists of components of each play)
  - body (list of character vectors, each element the body only of a play)
  - chunks (list of character vectors)
  - actors (list of character vectors)
  - titles (list of character vectors)

We would then need a field for the meta Data (list) we wish to extract from the plays array

- meta Data (list of integer vectors)
  - num_actors (integer vector)
  - num_acts (integer vector)
  - num_scenes (integer vector)
  - num_words (integer vector)
  - num_sentences (integer vector)
  - ..., etc.

3.2 What are the methods for the class? ...

First we would have methods which would process the plays into its components (titles, body, actors, chunks, etc.)

Example pseudo code:

```r
#Input: Entire text downloaded from web
#Output: list of plays
processWebData <- function(EntireText) {

    plays <- getPlays(EntireText)            #subset plays

    return(plays)                            #return a list of plays)
}

#Input: list of plays
#Output: array of plays with their component parts
processWebData <- function(plays) {

    titles <- getTitles(plays)               #get titles

    body <- getBody(plays)                   #get body of plays

    chunks <- getChunks(body)                #get chunks

    actors <- getActors(chunks)              #get actors/speakers

    return(list(body, actors,                #return lists of play components
                chunks, titles))

}
```

Then we would need a method for extracting metaData info from the play components

```r
#intput: plays array
#output: list of metaData for each play
extractMetaData <- function(plays) {

    num_actors <- extractNumActors(actors)

    num_setences <- extractNumSentences(body)

    num_words <- extractNumWords(body)

    ...

    return(list(num_actors, num_sentences, num_words, ..., etc.))
}
```