ps3

Malvika Rajeev

9/27/2018

QUESTION 1

After reading up on reproducilibility in scientific computation, I had a few thoughts. Generally speaking, 'reproducibility' of any scientific process depends on two things: whether it is at all reproducible, and the way it's manufactured.

Replicating an experiment is becoming an important foundation of the scientific method. While it is important to value reproducibility, it raises two questions. Firstly, that the scientist is infact interested or even cares about bringing about reproducibility. Assuming that reproducible research is the main aim and the paper or the manuscript is the byproduct is a very heavy assumption. While the community in general may consider that 'unethical' etc, should the reputation of the scientist's work suffer? Should preconceived notions be excused in such cases? Secondly, experiments in social sciences have fundamental differences with those of physical sciences, but require the same if not more practice of computational (especially statistical) tools. The 'problem' of reproducibility then ceases to be one of following best practices. Some social studies cannot be repeated due to problems with the initial study, while others aren't replicable because the follow-up research did not follow the methods or use the same tools as the original study, or maybe that the study simply cannot be replicated? For example, a study of race and affirmative action performed at Stanford University was 'replicated' at the University of Amsterdam in the Netherlands, in another country with different racial diversity. When the study was later repeated at Stanford, the original published results were indeed replicated. (Source: How scientists are addressing the 'reproducibility problem' By Deborah Berry)

I came accross a study that had to be retracted because of some minor error in the code that it used. The retraction read: "An in-house data reduction program introduced a change in sign for anomalous differences. This program, which was not part of a conventional data processing package, converted the anomalous pairs (I+ and I-) to (F- and F+, thereby introducing a sign change. As the diffraction data collected for each set of MsbA crystals1., 2., 3. were processed with the same program, the structures reported had the wrong hand". Instances like these stress upon the importance of having accurate computational tools for your study. And since programming, in general, is ever growing, dynamic and application based, accuracy can be thought of as a function of how many people can use your code and get the same results. This goes hand in hand with being updated with the open source online community, as such because the distinction between 'users' and 'core developers' isn't very rigid. The main benefit of following conventions is obvious: if code isn't easily readable, it is less conducive to modification and constructive criticism.

I like researching and knowing exactly what my function is supposed to do beforehand by getting some domain knowledge. I also like code to be crisp and short. To do this I realise that sometimes my code gets confusing to read. Also, possibly because I don???t have a lot of experience, names of functions don???t seem as pressing to me, because that's contextual.

Cohort Linear Model

I read the housing model paper and the code, and I personally thought it was a good read. The github ReadMe was quite informative, and the code was neat and comprehensible, replete with comments, which is great. They even provide cleaned data set ready for download. In section however, during the discussion we saw that were some minor lapses, like undefined variables being used inside a function, but nothing majorly disastrous. Overall, I think this study is a great example of good coding practices. Even in the paper, they acknowledge the circumstances in which their study will not be entirely reproducible: "However, when thinking about external validity of the CLM, one should refer the two basic assumptions that this model was developed upon. Our first assumption that housing career increases over a household???s life span should hold globally. Yet, our second assumption about how housing services are being offered across metropolitan regions is US-based and in order for this model to work in other context, this assumption needs to be modified accordingly". Which, again, I think, is helpful.

QUESTION 2

```
library(testthat)
## Note that this code uses the XML package rather than xml2 and rvest
## simply because I had this code sitting around from a previous demonstration.

## @knitr download

moderators <- c("LEHRER", "LEHRER", "LEHRER", "MODERATOR", "LEHRER", "HOLT")

candidates <- rbind(c(Dem = "CLINTON", Rep = "TRUMP"),
                   c(Dem = "OBAMA", Rep = "ROMNEY"),
                   c(Dem = "OBAMA", Rep = "MCCAIN"),
                   c(Dem = "KERRY", Rep = "BUSH"),
                   c(Dem = "GORE", Rep = "BUSH"),
                   c(Dem = "CLINTON", Rep = "DOLE"))



library(XML)
library(stringr)
library(assertthat)

url <- "http://www.debates.org/index.php?page=debate-transcripts"

yrs <- seq(1996, 2012, by = 4)
type <- 'first'
main <- htmlParse(url)
listOfANodes <- getNodeSet(main, "//a[@href]")
labs <- sapply(listOfANodes, xmlValue)
inds_first <- which(str_detect(labs, "The First"))
## debates only from the specified years
inds_within <- which(str_extract(labs[inds_first], "\\d{4}")
                     %in% as.character(yrs))
inds <- inds_first[inds_within]
## add first 2016 debate, which is only in the sidebar
ind_2016 <- which(str_detect(labs, "September 26, 2016"))
inds <- c(ind_2016, inds)
debate_urls <- sapply(listOfANodes, xmlGetAttr, "href")[inds]

n <- length(debate_urls)

assert_that(n == length(yrs)+1)
```

```
## [1] TRUE
## @knitr extract

debates_html <- sapply(debate_urls, htmlParse)

get_content <- function(html) {
  # get core content containing debate text
  contentNode <- getNodeSet(html, "//div[@id = 'content-sm']")
  if(length(contentNode) > 1)
    stop("Check why there are multiple chunks of content.")
  text <- xmlValue(contentNode[[1]])
  # sanity check:
  print(xmlValue(getNodeSet(contentNode[[1]], "//h1")[[1]]))
  return(text)
}

debates_body <- sapply(debates_html, get_content)
```

```
## [1] "September 26, 2016 Debate Transcript"
## [1] "October 3, 2012 Debate Transcript"
## [1] "September 26, 2008 Debate Transcript"
## [1] "September 30. 2004 Debate Transcript"
## [1] "October 3, 2000 Transcript"
## [1] "October 6, 1996 Debate Transcript"
```

I decided to do all the analyses we need to do on the basis of year (names are repeated, in some instance the moderator is literally refered to as “moderator”, etc). My basic plan was to make several small functions. I ended up calling a function inside functions many times, so that that particular function would work on its own.

```
library(stringr)
required_year <- function(x) {
  'if'(!(x %in% seq(2016, 1996, by = -4)), {print("Please enter an election year betw
een 1996 and 2016.")},{
  y <- (-0.25*x) + 505
  transcript = str_replace_all(debates_body[y],"(\n)+"," ")
  return(transcript)})
  expect_gt(nchar(transcript),1)
  }
```

As an example, I’m working with the 2004 data. Also, instead of combining all the data into one dataframe, I thought it would be better if I get all the data for any specific year by using generalisable functions.

_____restructuring the data_____

My idea is to map every year to an index in the table that I created, which consists of democractic candidate, republican candidate, and moderator, as they appear in the transcripts.

```
N = 6 #set this to the number of elections years we are analysing
candidates <- rbind(c(Dem = "CLINTON", Rep = "TRUMP"),
                                         c(Dem = "OBAMA", Rep = "ROMNEY"),
                                         c(Dem = "OBAMA", Rep = "MCCAIN"),
                                         c(Dem = "KERRY", Rep = "BUSH"),
                                        c(Dem = "GORE", Rep = "BUSH"),
                                        c(Dem = "CLINTON", Rep = "DOLE"))

moderators <- rbind("HOLT","LEHRER","LEHRER","LEHRER","MODERATOR","LEHRER")
tableOfSpeakers <- cbind(candidates, moderators)

#So now, starting from 2016, the speakers are in order till 2016.
```

_____finding out markers for the transcript, according to the year._____

Every chunk starts with the name of the speakers, which I then use to separate the data into literal chunks. First I find the index of the chunks. I get an n by 2 matrix. I dont need the second column. So I weed it out.

```
library(stringr)


find_markers <- function(x){
  if (!(x %in% seq(2016, 1996, by = -4))) {
      print("Please enter an election year between 1996 and 2016, else expect an error message!")

  }
  else {
  y <- (-0.25*x) + 505
  transcript <- required_year(x)
  pattern <- paste(tableOfSpeakers[y, ],":", sep="",collapse="|")  ##the names of the speakers made regex frieNDLY
  markers <- str_locate_all(transcript, pattern)
  markers <- markers[[1]] ##unnesting it
  markers <- markers[ ,1] #Now markers has the starting index of when a particular speaker starts talking.
  expect_is(markers, "integer")
  return(markers)
  }}


#Now creating the metadata

listOfMarkers <- function(){
  temp = list()

  for (i in seq(2016, 1996, by = -4)){
    j <- (-0.25*i) + 505
    temp[[j]] <- find_markers(i)
    names(temp)[[j]] <- paste(i) #changing index to the year.
  }

  temp

}
expect_length(listOfMarkers(),nrow(tableOfSpeakers))
listOfMarkers()["2012"]
```

```
## $`2012`
##   [1]   292   614  2012  2138  4205  4243  6064  6464  6610  8038  8890
##  [12]  9167 10173 12547 12570 13802 13821 14082 14734 14815 14885 14919
##  [23] 15024 15043 16575 16938 17042 18353 19331 19350 19429 19484 19540
##  [34] 19638 19666 19941 21490 22440 23049 23140 23210 23362 24572 25068
##  [45] 26456 26474 26947 27471 27507 29900 30127 30161 30232 30266 30280
##  [56] 30356 30420 30471 30676 30814 30893 31180 32536 32593 32605 32628
##  [67] 32675 32780 32912 32944 33531 33674 35248 36560 36688 36724 37036
##  [78] 37064 37374 38176 38193 38305 39455 39473 39616 39720 39731 39947
##  [89] 41308 42275 42390 44550 44573 44887 44923 44963 44999 45020 45073
## [100] 46028 47057 47186 47342 47391 47875 49158 50162 50199 50237 50249
## [111] 50266 50339 50397 50456 50474 50525 50561 50604 50652 50759 50778
## [122] 50872 50905 51030 51151 51171 51203 51223 51242 51622 52064 52107
## [133] 52154 52186 52953 52995 53255 53302 53396 53530 54682 55347 56747
## [144] 57074 57152 59211 59263 61165 61212 61748 61926 61983 62996 64123
## [155] 66806 67195 67212 67262 67322 67338 67414 68097 69277 69583 70893
## [166] 71931 71965 72009 72028 73886 74310 74365 74393 75041 76657 76726
## [177] 77213 78914 78933 79018 79261 80218 80355 80593 80738 81403 81767
## [188] 81809 81858 83373 83431 83449 85463 85507 85715 85747 86324 86673
## [199] 86717 87511 87530 87813 87836 89642 89660 90003 90181 92420 92470
## [210] 94847
```

So the idea hopefully is clear. I am executing each activity for a particular year, which can then be collated to a bigger database with just a few lines of code.

Example year:

```
markers <- find_markers(2004)
```

_____segregating the transcript according to markers._____

Now I get the actual chunks by the index matrix I just created.

```
get_chunks <- function(x){

  markers <- find_markers(x)
  transcript <- required_year(x)##so that "markers" and "transcript" exist in the fun
ction"

  s <- ((-0.25)*x) + 505   #to help me index through the table
  l <- length(markers)
  chunks <- vector(mode = "character", length = l) #initialise a character vector

  #we substract one to make sure the first letter of the next marker isnt included
  for (i in 1:(l-1)){
    chunks[i] <- substr(transcript,markers[i],markers[i+1]-1)
    }
  chunks[l] <- substr(transcript,markers[l],nchar(transcript))
  return(chunks)
}

#Example year
chunks <- get_chunks(2004)
```

Collating same speakers: I compare the first word of consecutive chunk, which is the name of the speaker. If its the same, i collate it, and pop out the second one. The reason i’m not looping through the length of chunks is that I was afraid that if the length reduces, it might break the code?

```
collate = TRUE  ##Set this to false if you don't want to collate
while (collate){
     i <- 1

     while (i > 0){
     if (word(chunks[i],1) == word(chunks[(i+1)],1)){
          chunks[i] = paste(chunks[i], chunks[i+1], sep = " ");
          chunks <- chunks[-(i+1)];
           }
     i <- i+1;

     if (i == length(chunks)){
          break
     }
     }
     break
}
```

Creating a nested list of chunks indexed by speaker name:

```
final_output <- function(x){
  'if'(!(x %in% seq(2016, 1996, by = -4)), {print("Please enter an election year betw
een 1996 and 2016.")},{
    chunks <- get_chunks(x)
    s <- (-0.25*x) + 505
    final <- list(chunks[sapply(chunks,function(m) grepl(paste(tableOfSpeakers[s,1]),
m))],  #Look for the speakers from that year
                  chunks[sapply(chunks,function(m) grepl(paste(tableOfSpeakers[s,2]),
m))],
                  chunks[sapply(chunks,function(m) grepl(paste(tableOfSpeakers[s,3]),
m))])
    names(final) = c(str_to_title(paste(tableOfSpeakers[s,1])), str_to_title(paste(ta
bleOfSpeakers[s,2])),           str_to_title(paste(tableOfSpeakers[s,3])))
    expect_length(final,3)
    is.null(final)
    return(final)
  })}


#_____________________________

##for all the years
listOfFinalChunks <- function(){
  temp = list()
  for (i in seq(2016, 1996, by = -4)){
    j <- (-0.25*i) + 505
    temp[[j]] <- final_output(i)
    names(temp)[[j]] <- paste(i)
  }
 temp
}

expect_length(listOfFinalChunks(),N)


##Printing out the number of each candidate's responses.

want_all_responses = TRUE #could set to false

while (want_all_responses){
  for (i in seq(2016, 1996, by = -4)){
      for (j in seq(1,2)){
       stat <- sprintf("Candidate %s had %d responses.", names(listOfFinalChunks()[pa
ste(i)][[1]])[[j]],                                        length(listOfFinalChunks()
[paste(i)][[1]][[j]]));
       print(stat)
      }
  }
  break
}
```

```
## [1] "Candidate Clinton had 87 responses."
## [1] "Candidate Trump had 124 responses."
## [1] "Candidate Obama had 56 responses."
## [1] "Candidate Romney had 71 responses."
## [1] "Candidate Obama had 127 responses."
## [1] "Candidate Mccain had 127 responses."
## [1] "Candidate Kerry had 33 responses."
## [1] "Candidate Bush had 41 responses."
## [1] "Candidate Gore had 49 responses."
## [1] "Candidate Bush had 56 responses."
## [1] "Candidate Clinton had 45 responses."
## [1] "Candidate Dole had 46 responses."
```

```
#____________________________


#Example year
final <- final_output(1996)
names(final)
```

```
## [1] "Clinton" "Dole"    "Lehrer"
```

```
head(final$Dole)
```

```
## [1] "DOLE: Thank you. Thank you, Mr. President, for those kind words. Thank the pe
ople of Hartford, the Commission, and all those out here who may be listening or watc
hing.It's a great honor for me to be here standing here as the Republican nominee.
I'm very proud to be the Republican nominee reaching out to Democrats and Independent
s.I have three very special people with me tonight: My wife, Elizabeth; my daughter,
Robin, who has never let me down, and a fellow named Frank Carafa from New York, alon
g with Ollie Manninen who helped me out in the mountains of Italy a few years back.
I've learned from them that people do have tough times. And sometimes you can't go it
alone. And that's what America is all about.I remember getting my future back from do
ctors and nurses and a doctor in Chicago named Dr. Kalikian . And ever since that tim
e, I've tried to give something back to my country, to the people who are watching us
tonight.America is the greatest place on the face of the earth. Now, I know millions
of you still have anxieties. You work harder and harder to make ends meet and put foo
d on the table. You worry about the quality and the safety of your children, and the
quality of education. But even more importantly, you worry about the future and will
they have the same opportunities that you and I have had.And Jack Kemp and I want to
share with you some ideas tonight. Jack Kemp is my running mate, doing an outstanding
job. Now, I'm a plain-speaking man and I learned long ago that your word was your bon
d. And I promise you tonight that I'll try to address your concerns and not try to ex
ploit them. It's a tall order, but I've been running against the odds for a long time
and, again, I'm honored to be here this evening."
## [2] "DOLE: I think the basic difference is, and I have had some experience in thi
s, I think the basic difference, I trust the people. The President trusts the governm
ent. We go back and look at the healthcare plan that he wanted to impose on the Ameri
can people. One seventh the total economy, 17 new taxes, price controls, 35 to 50 new
bureaucracies that cost $1.5 trillion. Don't forget that, that happened in 1993. A ta
x increase, a tax on everybody in America. Not just the rich. If you made 25,000 as t
he original proposal, you got your Social Security taxes increased. We had a BTU tax
we turned into a $35 million gas tax, a $265 billion tax increase.I guess I rely more
on the individual. I carry a little card in my pocket called the Tenth Amendment. Whe
re possible, I want to give power back to the states and back to the people. That's m
y difference with the President. We'll have specific differences later. He noted a fe
w, but there are others."

## [3] "DOLE: Well, he's better off than he was four years ago."
```

## [4] "DOLE: And I may be better off four years from now, but I don't know. I looked at the slowest growth in the century. He inherited a growth of 4.7 4.8 percent, now it's down to about 2.4 percent. We're going to pass a million bankruptcies this year for the first time in history. We've got stagnant wages. In fact, women's wages have dropped 2.2 percent. Men's wages haven't gone up, gone down. So we have stagnation.We have the highest foreign debt in history. And it seems to me that if you take a look, are you better off? Well, I guess some may be better off. Saddam Hussein is probably better off than he was four years ago. Renee Proval (ph) is probably better off than he was four years ago. But are the American people? They're working harder and higher and harder paying more taxes. For the first time in history, you pay about 40 percent of what you earn. More than you spend for food, clothing and shelter combined for taxes under this administration. So some may be better off.They talk about family income being up. That's not true in Connecticut, family income is down. And it's up in some cases because both parents are working. One works for the family, and one works to pay taxes for the government. We're going to give them tax cuts so they can spend more time with their children, maybe even take a vacation. That's what America is all about."

## [5] "DOLE: I doubt that I acknowledged that this year. But in any event, I think we just look at the facts. We ask the people that are viewing tonight, are you better off than you were four years ago. It's not whether we're better off, it's whether they're better off.Are you working harder to put food on the table, feed your children. Are your children getting a better education. Drug use has doubled the past 44 months all across America. Crime has gone down but it's because the mayors like Rudy Giuliani where one third of the drop happened in one city, New York City.So, yes, some may be better off. But of the people listening tonight, the working families who will benefit from economic packages, they'll be better off when Bob Dole is president and Jack Kemp is vice president."

## [6] "DOLE: Well, I must say, I look back at the vote on Medicare in 1965, we had a program called Eldercare that also provided drugs and means tests to people who needed medical attention received it. I thought it was a good program. But I've supported Medicare ever since.In fact, I used to go home, my mother would tell me, Bob, all I've got is my Social Security and my Medicare, don't cut it. I wouldn't violate anything my mother said. In fact, we had a conversation about our mothers one day, a very poignant conversation in the White House. I'm concerned about healthcare. I've had the best healthcare from government hospitals, Army hospitals and I know its importance, but we've got to fix it. It's his trustees, the President's trustees, not mine, who says it's going to go broke. He doesn't fix it for ten years. We ought to appoint a co

```
mmission, just as we did in Social Security in 1983, when we rescued Social Security,
and I was proud to be on that commission, along with Claude Pepper, the champion of s
enior citizens from Florida. And we can do it again, if we take politics out of it. S
top scaring the seniors, Mr. President. You've already spent $45 million scaring seni
ors and tearing me apart. I think it's time to have a truce."
```

Counting laughter and applause.

I create a function that takes a list of strings and counts the number of occurrences of the words we want. .

```
laugh <- function(y){
  sum(sapply(y, function(x) str_count(x, "\\[|\\(laughter\\]|\\)|\\[|\\(LAUGHTER\\]|
\\)")))
}

applause <- function(y){
  sum(sapply(y, function(x) str_count(x, "\\[|\\(applause\\]|\\)|\\[|\\(APPLAUSE\\]|
\\)")))
}

count_laughter_applause<- function(x){
  'if'(!(x %in% seq(2016, 1996, by = -4)), {print("Please enter an election year betw
een 1996 and 2016.")},{
  s <- (-0.25*x) + 505
  final <- final_output(x)
  for (m in seq(1,2)){
      stat <- sprintf("Candidate %s got %d laughs and %d applauses", names(final)[m],
 laugh(final[[m]]), applause(final[[m]]))
      print(stat)
  }})}

count_laughter_applause(2004)
```

```
## [1] "Candidate Kerry got 2 laughs and 2 applauses"
## [1] "Candidate Bush got 1 laughs and 1 applauses"
```

```
count_laughter_applause(1996)
```

```
## [1] "Candidate Clinton got 0 laughs and 0 applauses"
## [1] "Candidate Dole got 4 laughs and 4 applauses"
```

I also created a nested list for all the years, where the data for every year can be easily accessed through index.

```
laughter_applause <- function(x){
  row <- list()
  for (i in seq(2016, 1996, by = -4)){
    y <- (-0.25*i) + 505

    final <- final_output(i)
    temp <- list()
    for (j in seq(1,2)){

      temp[[j]] <- rbind(names(final)[j], laugh(final[[j]]), applause(final[[j]]))

    }
    row[[y]] <- temp
    names(row)[[y]] <- paste(i)
    }
row
}

laughter_applause()["2008"]
```

```
## $`2008`
## $`2008`[[1]]
##      [,1]
## [1,] "Obama"
## [2,] "12"
## [3,] "12"
##
## $`2008`[[2]]
##      [,1]
## [1,] "Mccain"
## [2,] "12"
## [3,] "12"
```

Removing silly symbols, and the laughter and applause.

```
strip_all <- function(x){
  sapply(x, function(y) str_replace_all(y,"\\[|\\(laughter\\]|\\)|\\[|\\(LAUGHTER\\]|
\\)|\\[|\\(applause\\]|\\)|\\[|\\(APPLAUSE\\]|\\)|(\\(|\\[).+(\\)|\\])|",""))
}

##creating a clean data set.

for (i in seq(1,2)){
  final[[i]] <- as(strip_all(final[[i]]),"list")
}

cleandata <- listOfFinalChunks()

for (i in seq(2016, 1996, by = -4)) {
  for (j in seq(1,2)) {
    cleandata[paste(i)][[1]][[j]] = as.list(strip_all(cleandata[paste(i)][[1]][[j]]))
  }
}

expect_length(cleandata, N)
```

Storing all the words, characters and sentences for the candidates.

I use the str method “boundary”. It extracts out words, characters and even sentences. I love this method.

```
words <- function(x){
  temp <- sapply(x, function(y) str_split(y, boundary("word"))) ##extract all words f
rom each chunk of the list
  names(temp) = NULL
  return(unlist(temp))
}

head(words(final$Dole))
```

```
## [1] "DOLE"  "Thank" "you"   "Thank" "you"   "Mr"
```

```
sen <- function(x){
  sentences <- sapply(x, function(y) str_split(y,"\\.")) ##split by "."
  temp <- unlist(sentences)
  names(temp) = NULL
  return(temp[lapply(temp,function(y) str_count(y))>1]) ##remove chunks of one letter
}
head(sen(final$Dole))
```

```
## [1] "DOLE: Thank you"

## [2] " Thank you, Mr"

## [3] " President, for those kind words"

## [4] " Thank the people of Hartford, the Commission, and all those out here who may be listening or watching"
## [5] "It's a great honor for me to be here standing here as the Republican nominee"

## [6] " I'm very proud to be the Republican nominee reaching out to Democrats and Independents"
```

```r
char <- function(x){
 m <-gsub("[[:space:]]?", "", x) ##remove all white spaces
 characters <- sapply(m, function(y) str_split(y, boundary("character"))) ##str_trim first removes the white spaces, then stores the characters.
 names(characters) = NULL
 return(unlist(characters))
 }

avg_word <- function(x){
  return(round(length(char(x))/length(words(x)),2))
}



counting_stuff <- function(x) {
  if (!(x %in% seq(2016, 1996, by = -4))) {print("Please enter an election year between 1996 and 2016.")}
      else {
  y <- (-0.25*x) + 505;
  ##create the dataset
  col <- data.frame(matrix(1:10, nrow = 5, ncol = 2),row.names=c("name","words spoken","sentences","characters", "average word length"),stringsAsFactors=FALSE)
  final <- final_output(x)
  for (i in seq(1,2)){
  temp <-rbind(names(final)[[i]],length(words(final[[i]])),length(sen(final[[i]])),length(char(final[[i]])), avg_word(final[[i]]))
  col[i] <- temp}
  colnames(col) <- c("Democratic Candidate", "Republican Candidate")
  return(col)}}





tableOfCount <- counting_stuff(2004)
counting_stuff(2008)
```

	Democratic Candidate <chr>	Republican Candidate <chr>
name	Obama	Mccain
words spoken	15232	14282
sentences	866	904

	Democratic Candidate	Republican Candidate
	<chr>	<chr>
characters	70538	67052
average word length	4.63	4.69

5 rows

```
listOfTableOfCount = list()
for (i in seq(2016, 1996, by = -4)){
  j <- (-0.25*i) + 505
  listOfTableOfCount[[j]] <- counting_stuff(i)
  names(listOfTableOfCount)[[j]] <- paste(i)
}
listOfTableOfCount
```

```
## $`2016`
##                     Democratic Candidate Republican Candidate
## name                             Clinton                Trump
## words spoken                        6408                 8627
## sentences                            436                  733
## characters                         29319                38539
## average word length                 4.58                 4.47
##
## $`2012`
##                     Democratic Candidate Republican Candidate
## name                               Obama               Romney
## words spoken                        7370                 7901
## sentences                            365                  577
## characters                         34028                35850
## average word length                 4.62                 4.54
##
## $`2008`
##                     Democratic Candidate Republican Candidate
## name                               Obama               Mccain
## words spoken                       15232                14282
## sentences                            866                  904
## characters                         70538                67052
## average word length                 4.63                 4.69
##
## $`2004`
##                     Democratic Candidate Republican Candidate
## name                               Kerry                 Bush
## words spoken                        7001                 6220
## sentences                            479                  500
## characters                         31855                28600
## average word length                 4.55                  4.6
##
## $`2000`
##                     Democratic Candidate Republican Candidate
## name                                Gore                 Bush
## words spoken                        7290                 7537
## sentences                            426                  536
## characters                         32555                33412
## average word length                 4.47                 4.43
##
## $`1996`
##                     Democratic Candidate Republican Candidate
## name                             Clinton                 Dole
## words spoken                        7694                 8122
## sentences                            412                  642
## characters                         34842                36531
## average word length                 4.53                  4.5
```

The avergae word count remains the same in almost all cases (4.5), which is the average word count of the English language. The republican candidate always utters more sentences.

Creating another list to store the actual words and characters of each candidate

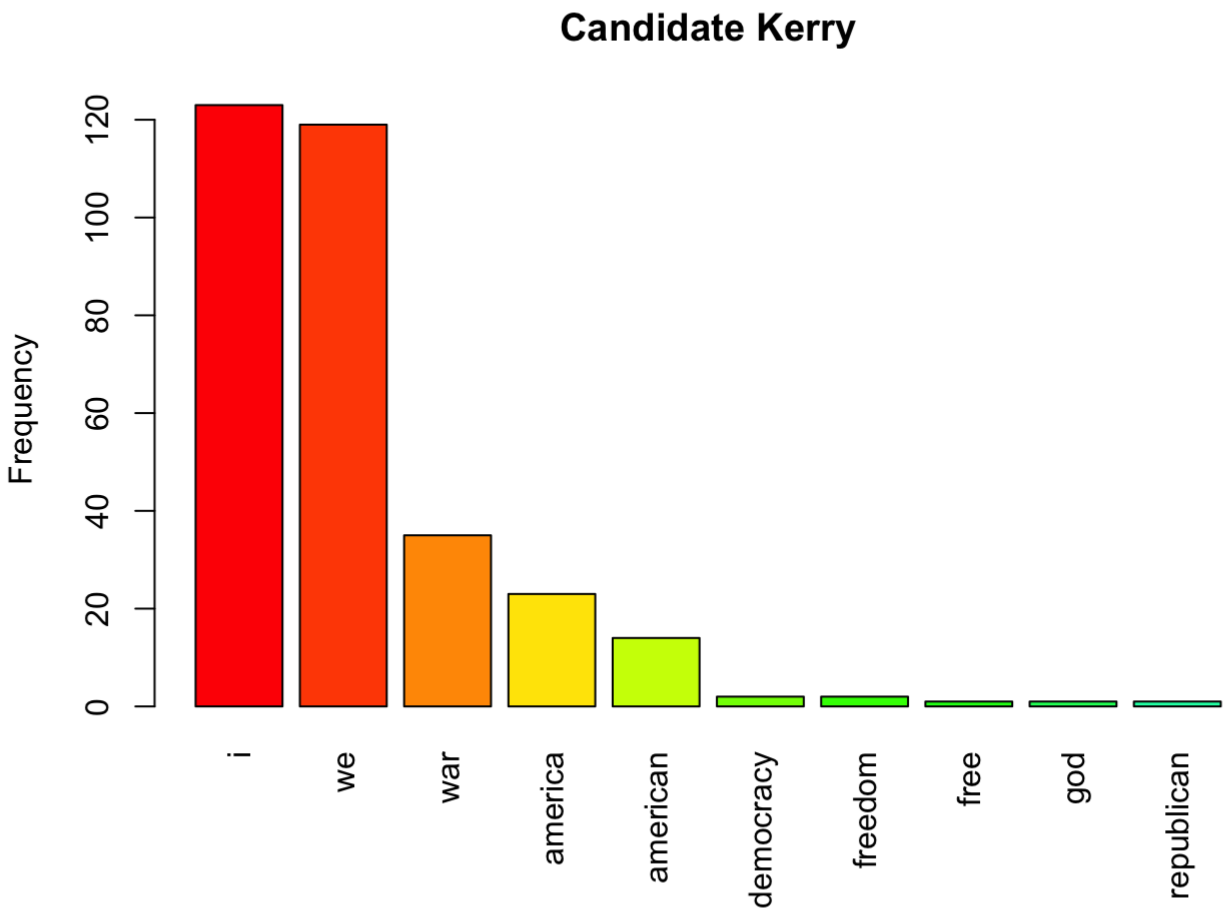
```r
store_stuff <- function(x){

  dat <- list()  #initialise a list
  final <- final_output(x) #get the "final" list (in case that function hasn't been called)

  'if'(!(x %in% seq(2016, 1996, by = -4)), {print("Please enter an election year between 1996 and 2016.")},{
  s = (-0.25*x) + 505

  for (i in seq(1,2)){
       temp <- list(words(final[[i]]), char(final[[i]]))
       dat[[i]] <- temp
  }


  names(dat) = c(paste(tableOfSpeakers[s,1]), paste(tableOfSpeakers[s,2]))
  return(dat)
  })}


##creating metadata using this function, easily indexable by year

listOfWordsAndChars = list()
for (i in seq(2016, 1996, by = -4)){
  j <- (-0.25*i) + 505
  listOfWordsAndChars[[j]] <- store_stuff(i)
  names(listOfWordsAndChars)[[j]] <- paste(i)
}

store_stuff(2005)
```

```
## [1] "Please enter an election year between 1996 and 2016."
## [1] "Please enter an election year between 1996 and 2016."
```

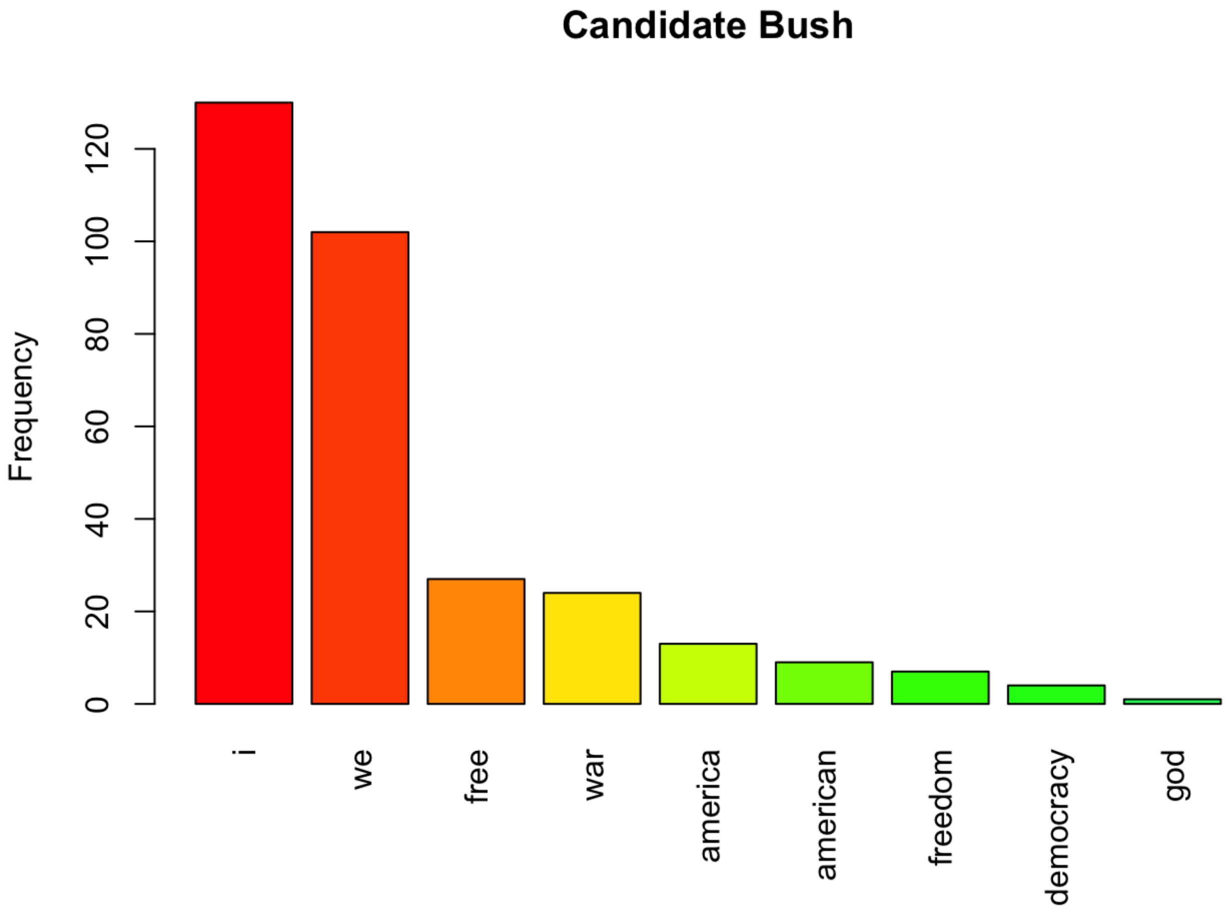
Function to Create frequencies and histograms by year

I create a function that looks for the words specified in the question, uses plyr package to calculate frequencies, and then finally loops through to create a list of those words and plots a graph for each candidate.

```
candidateFig <- function(x){
    if (!(x %in% seq(2016, 1996, by = -4))) {
      print("Please enter an election year between 1996 and 2016.")
    }

    else {
      library(plyr)
      library(grid)
      library(gridBase)
      s <- (-0.25*x) + 505
      final <- final_output(x)
      pattern <- "i|we|(america|american)|(democracy|democratic)|republic|democrat|re
publican|(free|freedom)|war|god|god bless|(jesus|christ|christian)"

      word_frequencies = list() #it's easier to add stuff to a list
      for (i in seq(1,2)){
          temp_1 <- str_to_lower(words(final[i]))
          temp_2 <- grep(paste("^(",pattern,")$",sep=""), temp_1, value = TRUE)
          word_freq = plyr::count(temp_2) %>% arrange(desc(freq))
          word_frequencies[[i]] = word_freq
          fig <- barplot(word_freq$freq, ylab= "Frequency" ,
                         main = paste("Candidate",names(final)[[i]]), col=rainbow(20
))
          vps <- baseViewports()
          pushViewport(vps$inner, vps$figure, vps$plot)
          grid.text(word_freq$x,
                    x = unit(fig, "native"), y=unit(-1, "lines"),
                    just="right", rot=90)

          popViewport(3)
          cat("\n")

       }
      names(word_frequencies) = c("Democratic Candidate","Republican Candidate")
      return(word_frequencies)}}

candidateFig(2004)
```



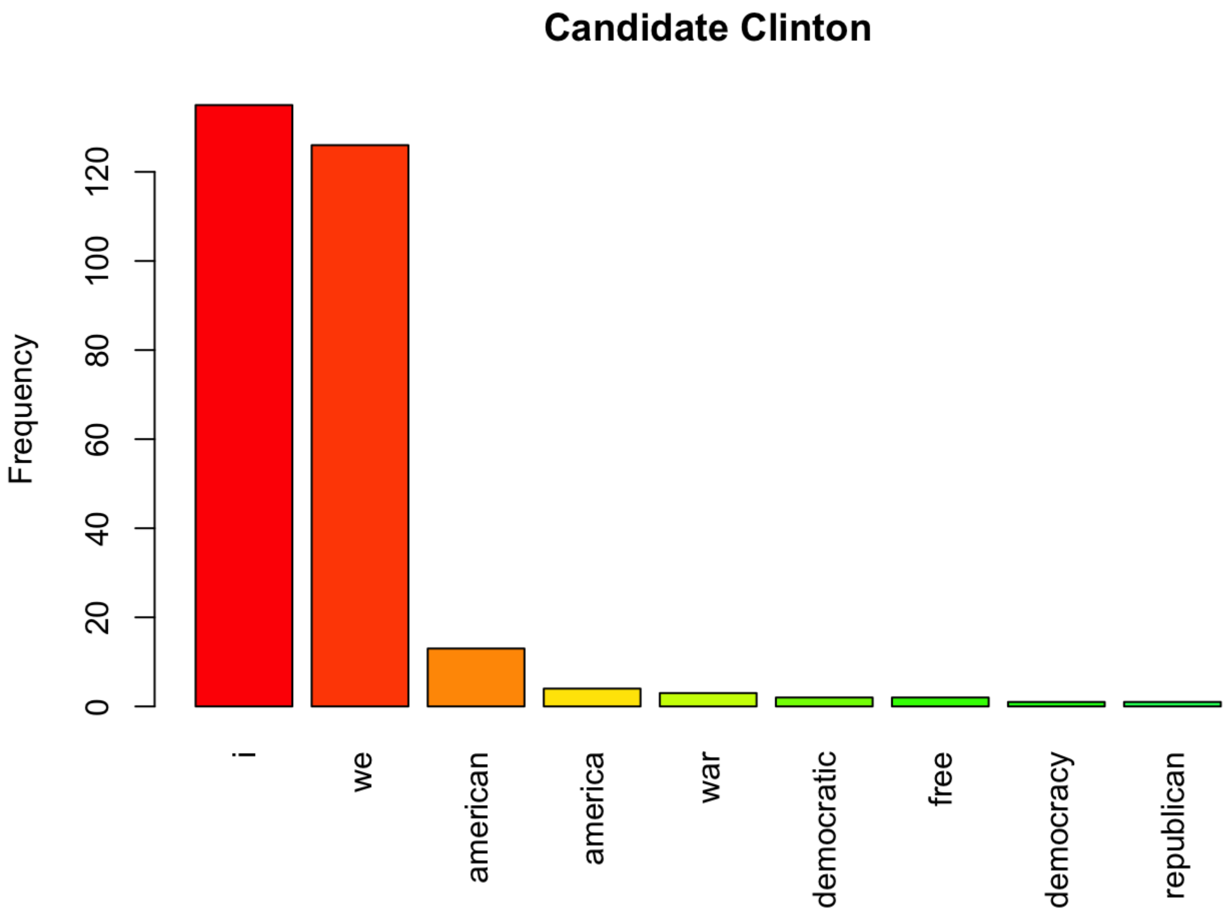
Candidate Kerry



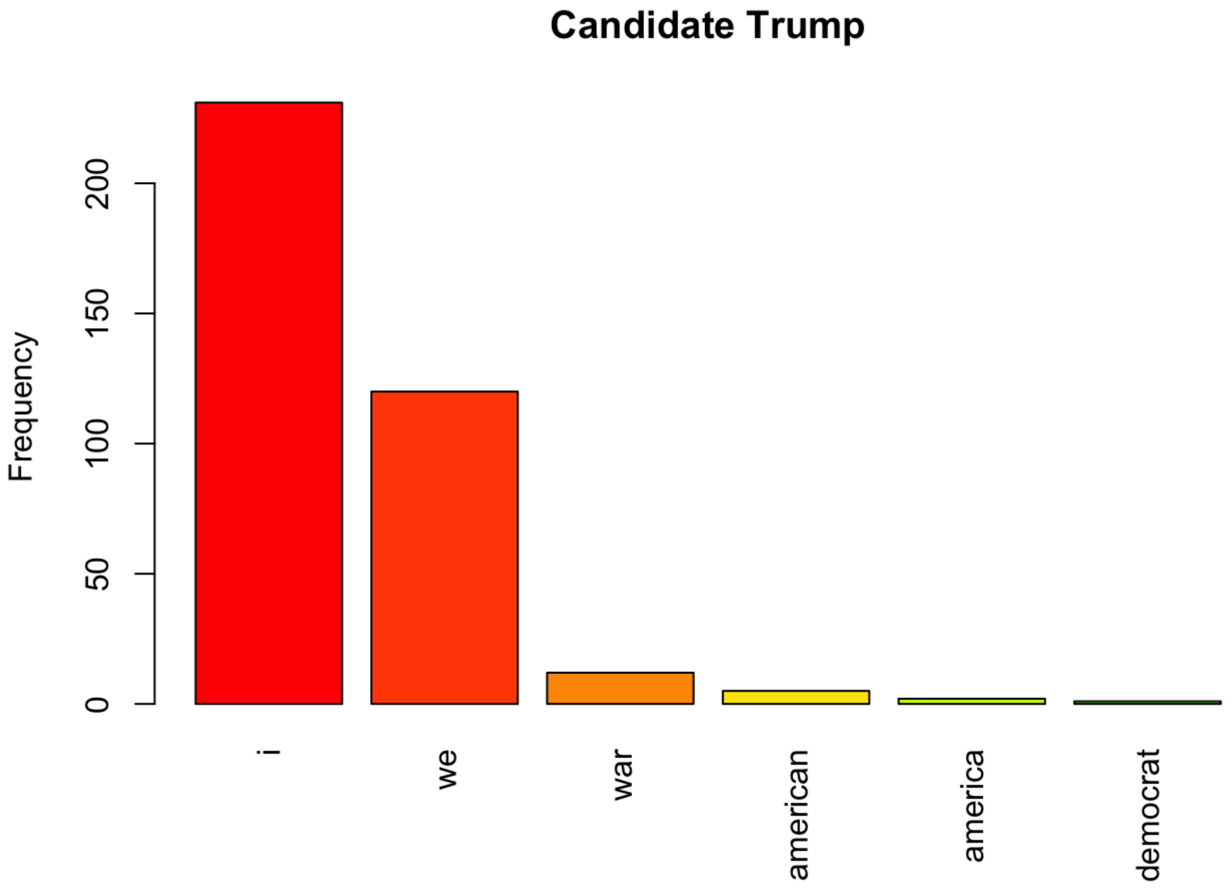
Candidate Bush

```
## $`Democratic Candidate`
##             x freq
## 1           i  123
## 2          we  119
## 3         war   35
## 4     america   23
## 5    american   14
## 6   democracy    2
## 7     freedom    2
## 8        free    1
## 9         god    1
## 10 republican    1
##
## $`Republican Candidate`
##           x freq
## 1         i  130
## 2        we  102
## 3      free   27
## 4       war   24
## 5   america   13
## 6  american    9
## 7   freedom    7
## 8 democracy    4
## 9       god    1
```

```
candidateFig(2016)
```



Candidate Clinton



Candidate Trump

```
## $`Democratic Candidate`
##            x freq
## 1          i  135
## 2         we  126
## 3   american   13
## 4    america    4
## 5        war    3
## 6 democratic    2
## 7       free    2
## 8  democracy    1
## 9 republican    1
##
## $`Republican Candidate`
##          x freq
## 1        i  231
## 2       we  120
## 3      war   12
## 4 american    5
## 5  america    2
## 6 democrat    1
```

I noticed consistently that for each year, the republican candidate utters “i” more times than “we”, and the democratic candidate utters “we” more. War is commonly mentioned by both parties.

QUESTION 3

From an Object Oriented approach, we could have a Debate class. This class could have the following attributes: 1. No of Speakers 2. No of Moderators 3. No of Adjudicators

A “presidential” debate would then be a subclass where there are two speakers and one moderator and zero adjudicator.

The data we would therefore need (fields) this object to represent are:

1. The year in which it takes place
2. The moderator
3. The candidates, and the parties to which they belong
4. Their actual responses
5. The location of the debate

.

The methods on the debate could be:

1. Segregate: Group by speaker. - returns a list of responses
2. Word count: Count the words spoken by each speaker. - returns an integer
3. Specific word count - specify a word and count how many times each speaker said it
4. Interruptions - count interruptions/ crosstalk
5. Fact-Check (I’m not entirely sure how that could work though).