Stat243: Problem Set 3, Due Wednesday Sep. 29

September 18, 2021

This covers the first half of Unit 5.

It's due **as PDF submitted to Gradescope** and with all materials submitted via GitHub at 10 am on Sep. 29.

Comments:

1. The formatting requirements are the same as previous problem sets.
2. Please note my comments in the syllabus about when to ask for help and about working together. In particular, **please give the names of any other students that you worked with on the problem set and indicate in comments any ideas or code you borrowed from another student.**

Problems

1. The goal of this problem is two-fold: first to give you practice with regular expressions and string processing and the second to have you thinking about writing well-structured, readable code (similar to question 4 of PS1). Regarding the latter, please focus your attention on writing short, modular functions that operate in a vectorized manner and also making use of *lapply()*/*sapply()* (or *purrr:map()* if you a fan of the tidyverse's approach to functional programming) to apply functions to your data structures. Think carefully about how to structure your objects to store the speech information. Note that my solution is around 100 lines of code (including comments but not empty lines) - it's possible to answer the problem with concise but readable code.

   The website The American Presidency Project at UCSB has the text from all of the State of the Union speeches by US presidents. (These are the annual speeches in which the president speaks to Congress to "report" on the situation in the country.) Your task is to process the information and produce data on the speeches. Note that while I present the problem below as subparts (a)-(i), your solution does NOT need to be divided into subparts in the same way. Your solution should do all of the downloading and processing from within R so that your operations are self-contained and reproducible.

   (a) From the website, you need to get the HTML file for each of the speeches. You'll need to start with this HTML file and extract the individual URLs for each speech. Then use that information to read each speech into R.

   (b) For each speech, extract the body of the speech and the year of the speech.

   (c) Convert the text so that all text that was not spoken by the president is stripped out and saved separately. For the Laughter and Applause cases, count the number of times each occurred in the speech.

(d) Extract the words and sentences from each speech as character vectors, one element per sentence and one element per word. Note that there are probably some special cases here. Try to deal with as much as you can, but we're not expecting perfection.

(e) For each speech count the number of words and characters and compute the average word length.

(f) Count the following words or word stems: I, we, America{,n}, democra{cy,tic}, republic, Democrat{,ic}, Republican, free{,dom}, war, God [not including God bless], God Bless, {Jesus, Christ, Christian}, and any others that you think would be interesting.

(g) The result of all of this activity should be well-structured data object(s) containing the information about the speeches.

(h) Make some basic plots that show how the variables have changed over time and (for presidents since Franklin Roosevelt in 1932) whether they seem to differ between Republican and Democratic presidents (the Republican presidents have been {Eisenhower, Nixon, Ford, Reagan, G.H.W. Bush, G.W. Bush, Trump} and the Democrats have been {Roosevelt, Truman, Kennedy, Johnson, Carter, Clinton, Obama, Biden}. Your response here does not have to be extensive but should illustrate what you would do if you were to proceed on to do extensive exploratory data analysis.

(i) Extra credit: Do some additional research and/or additional thinking to come up with additional variables that quantify speech in interesting ways. Do some plotting that illustrates how the speeches have changed over time.

Hint: Depending on how your process the speeches, you may end up with lists for which the name of a list element is very long, such as the entire text of the speech or the entire HTML. Syntax such as `names(myObj) <- NULL` may be helpful.

2. This problem asks you to design an object-oriented programming (OOP) approach to the text analysis in Problem 1. You don't need to code anything up, but rather to decide what the classes would be and the fields and methods of those classes. You can think of this in the context of R6 classes, or in the context of classes in an object-oriented language like Python or C++. To be clear, you do **not** have to write any of the code for the methods nor even formal code for the class structure; the idea is to design the formats of the classes. Basically if you were to redesign your functional programming code from problem 1 to use OOP, how would you design it? As your response for each class, please provide a bulleted list of methods and bulleted list of fields and for each item briefly comment what the purpose is. Also note how each method uses other fields or methods.

3. This problem provides practice in hacking the features of a languge to do something that was not really intended by the developers of the language.

   (a) Suppose I want to be lazy and when I type "k" in the R or RStudio console, R or RStudio should quit without asking any further questions. Write a bit of R code to achieve this.
   Hints: (a) What standard R function call, with what arguments, do I need to run to get R to quit without any questions? (b) If you type "k", what sequence of steps are carried out by R? How can I get something different (namely the code in hint (a)) to be run? (c) You will probably want this code to be in an 'eval=FALSE' chunk, since it if it successful, its operation will cause R to quit and presumably your document will not knit.

   (b) Now suppose that instead of writing "k", you want to be able to write "q" instead and achieve the same result. Does that work? If so, why can your "q" and R's "*q()*" co-exist?