Homework 3

Due: Wednesday February 13 at 11:59pm via compass2g

Use RStudio for all exercises. Efficiency is important. Use efficient programming techniques and modular programming as discussed in class, and make use of functions we have already created when possible.

You should provide one script (a `.R` or `.rmd` file) that contains all the code and includes code comments noting which code is for which exercises. You will also need to show and comment on the results, so place the results in a Word (or Open Office or HTML or PDF) document and write sentences to answer the questions, or use `knitr` to programmatically create your document. **Script files must be the actual script files**, not unevaluated code pasted into some other document.

Include your name in the name for each file submitted ('<Your-First-Name> <Your-Last-Name> HW#.R', e.g. 'JaneDoeHW3.R'). Any code based on code from elsewhere (e.g. code provided with the text) **must reference in code comments** the source of the original code.

Some initial setup code is provided in **HW3Setup.R** in the Homework 3 directory in compass.

Note: the `unname` function can be used to return values from a named vector without the names (e.g. file names) for the entries.

Exercises for All Students

For these 3 exercises, only basic comments are needed. State what the functions are, what they do, show your results and give short statements (in sentences) of what the results are for.

1) Write a function to extract the subject from messages. This is the value for the Subject key (Be sure to allow for capital or lower case s when searching for Subject in the header). The function should take the header from a message and return the subject value, and also handle the case where no Subject key is present in the header.
   Demonstrate your function on the headers from the 15 sample emails used in the chapter.
2) Write a function to count the letters from a message subject line. The function should take in a message header and return an integer (the number of letters in the subject line). Make use of the function created in exercise 1 and use modular programming to create your function. Again make sure to handle the missing Subject key case.
   Demonstrate your function on the 15 sample emails from the chapter.
3) Write a function to count the number of digits (0, 1, 2, 3, 4, 5, 6, 7, 8, or 9) in an email message body. The function should take a split email message (the result from the `splitMessage` function in the chapter) and return the number of digits in the body. Drop attachments and only count digits not in attachments.
   Again, show the results of your function for the sample emails.

Additional Exercises for Graduate Students

More extensive comments and explanations are needed for this one than the other 3 exercises.

4) Write a function to obtain the number of digits in each email in a directory (e.g. `easy_ham`, `easy_ham_2`, etc.) that relies on the function written in exercise 3. The function should take a directory name as its input and return a vector of the digits counts for messages in that directory as its output. (Note: the initial processing from `processAllWords` in section 3.5.4 of the text is a good example for processing a directory of messages.)
   Obtain basic summary statistics for the digits per message for each of the 5 email directories, and comment on differences in digits per message in the 5 directories. What do these differences suggest about using number of digits in a message as a basis for classifying spam and ham messages?