Homework Assignment #10

Matt Kline

May 5, 2017

Part 1. Performance Enhancement: Speed and Memory

1.) Use system.time() to time the expression

```
system.time(
  sort(runif(1000000))
)

##    user  system elapsed
##    0.31    0.00    0.44
```

for values of n ranging over $2^1, 2^2, ..., 2^18. Make a plot of the'user'execution time (y axis) versus n (x axis). Specify type =$
$"b" in plot() (for "both" points and lines). How does the execution time grow with n, i.e. is it linear or some other growth patt$

2.) Let n be a large integer. Use runif() to create a vector x containing n (positive) uniformly distributed random numbers. Then embed the following code in a call to the system.time() function:

```
x <- runif(1000000)
y <- rep(NA, 1000000)
system.time(
  for(i in 1:1000000) {
    y[i] <- sqrt(x[i])
  }
)

##    user  system elapsed
##    2.53    0.04    3.55
```

Do this a few times. Now, use system.time() to determine the user execution time for the command

```
x <- runif(1000000)
system.time(
  y <- sqrt(x)
)

##    user  system elapsed
##    0.01    0.00    0.01
```

Which is faster?

The second one is faster because it is a vector operation.

3.) The function expression() will take an expression (or command), like 2 + 2, and return it unevaluated as an object belonging to the class "expression" that can later be evaluated using the eval() function. For example:

```
e <- expression(2 + 2)
class(e)

## [1] "expression"

eval(e)

## [1] 4
```

Write a function that will take an expression argument e and a number n, evaluate e n times, and return the 'user' execution times as a vector.

```
func <- function(e, n){
  expr <- expression(e)
  time <- system.time(
    for (i in 1:n){
      eval(expr)
    }
  )
  return(time)
}
```

Part 2. Text Manipulation

Project Gutenberg (Project Gutenberg Literary Archive Foundation, 2009) makes available the State of the Union Address by United States Presidents from 1790-2001, which can be augmented to include the recent addresses. The text data can be summarized into word counts for each speech, and the speeches can be compared (e.g. via multi-dimensional scaling and hierarchical clustering) to see how they differ across time and party. The goal of the following problems is for you to analyze two "State of the Union" speeches: the first one (George Washington, 1776) and the most recently available one from Project Gutenberg (George W. Bush, 2005). In particular, you will be examining and comparing the words that each president used in his address and their frequency of use. The speeches are in the file "sotu gw gwb.txt".

4.) Set your working directory to wherever you saved the file. Then use scan() to read in the file and assign it to an object data1 (remember to specify the what argument; also specify sep = "" and blank.lines.skip = T). Then run the following:

```
data1 <- scan(file = "/Users/Matt/Desktop/sotu_gw_gwb(1).txt", what = "")
data2 <- paste(data1, collapse = " ")
```

What did the above code do to data1?

This took a list of words and made them into a long string

5.) Use strsplit() to split the two speeches (separated by ***; hint: use backslashes). Unlist the resulting list and assign it to the object speeches. What are the class and length of speeches?

```
speeches <- unlist(strsplit(x = data2, split = "\\*"))
```

6.) Now use gsub() to replace all periods, commas, semi-colons, parentheses, and hyphens (you will have to run gsub() several times, assigning the output to a new object each time). Then run tolower() on the last object you create to make everything lower case; call this final object speeches2.

```r
speechesa <- gsub(x = speeches, pattern = ".", replacement = " ")
speechesb <- gsub(x = speechesa, pattern = ",", replacement = " ")
speechesc <- gsub(x = speechesb, pattern = ";", replacement = " ")
speechesd <- gsub(x = speechesc, pattern = "()", replacement = " ")
speechese <- gsub(x = speechesd, pattern = "-", replacement = " ")
speeches2 <- tolower(speechese)
```

7.) Run the following, which splits each speech into a vector of individual words: What are the class and length of speeches3?

```r
speeches3 <- sapply(speeches2[-1], strsplit, " ")
class(speeches3)

## [1] "list"

length(speeches3)

## [1] 6
```

8.) We want to compare the two speeches but, with so many words in each speech, looking at a barplot might not be that helpful. Instead, we'll use a "word cloud" to visually represent the text. In a word cloud, large (small) words indicate high (low) frequency.

a Install the packages wordcloud and tm; load the wordcloud library. Then make two wordclouds using wordcloud(): one wordcloud for each of the two elements of speeches3.

```r
library(tm)

## Warning:  package 'tm' was built under R version 3.3.3
## Loading required package:  NLP

library(wordcloud)

## Warning:  package 'wordcloud' was built under R version 3.3.3
## Loading required package:  RColorBrewer

wordcloud(speeches3)

## Error in UseMethod("TermDocumentMatrix", x):  no applicable method for 'TermDocumentMatrix'
applied to an object of class "list"

wordcloud(speeches3)

## Error in UseMethod("TermDocumentMatrix", x):  no applicable method for 'TermDocumentMatrix'
applied to an object of class "list"
```

b Use the wordclouds from part (a) to discuss how the two speeches compare. As you think about how they compare, you might consider answering questions like the following: is one person more verbose than the other? did they use similar words? which words did they use the most/least; are those the same between the two? does anything odd stand out about either set of words? etc.