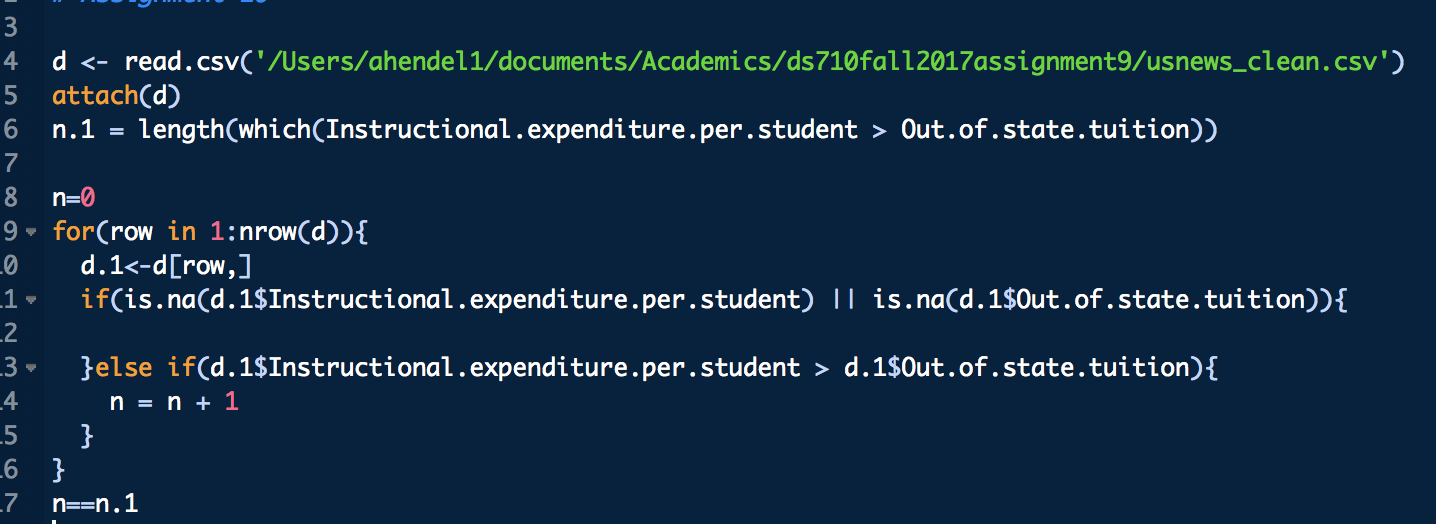
**Adam Hendel**

**DS 710**

**Homework 10**

**1.a Read the data into R and attach it.  Use *length* and *which* to determine how many schools have a per-student instructional expenditure higher than their out-of-state tuition.  Then use control flow to answer the same question.  (Check that the two methods give the same answer!)**

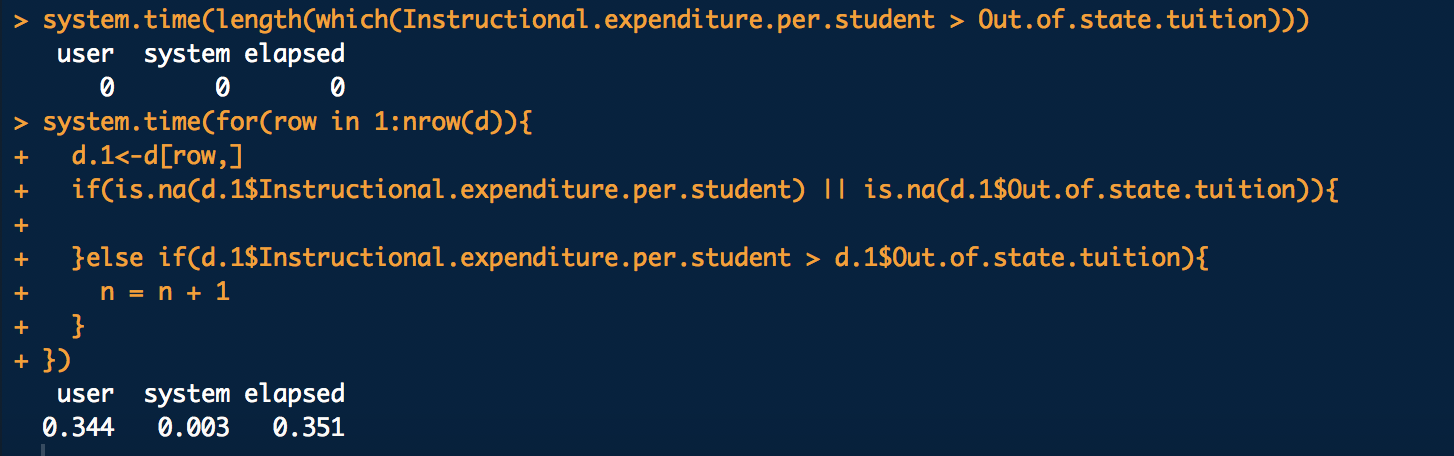




**1.b Use *system.time* to compare the running times of the two methods you wrote in part a.  Iterate each method enough times that you can see a difference in the running times.  Report the user time + system time for each method.  Which is more efficient?**

Both tested on 2.7ghz Intel Core i7:

The first method is by far faster…10 iterations yielded results of 0 seconds. The second method was much slower, as expected. I would not typically expect a for loop to be faster than vector computation.



**1.c Consider three different methods of finding the mean of each numeric column of the data:**

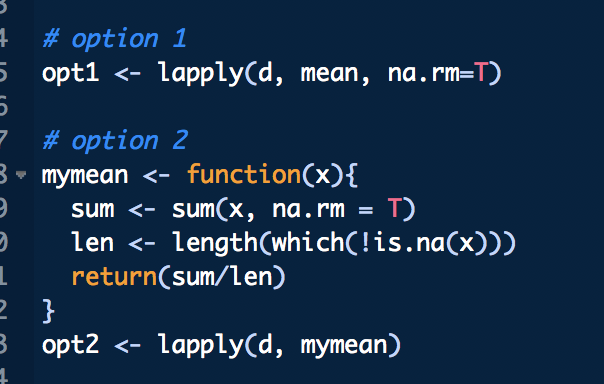
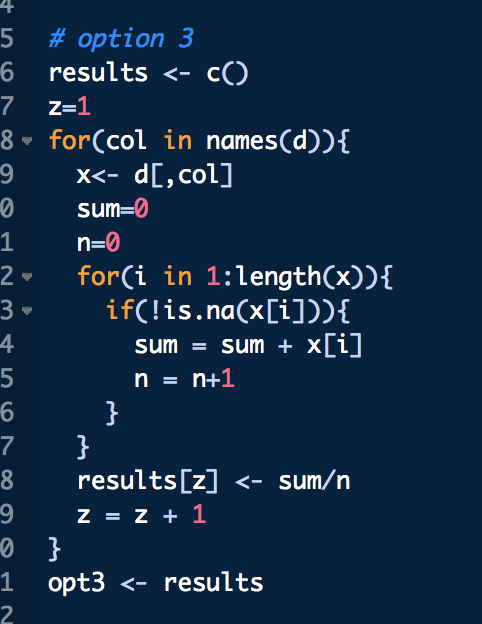
1. Using apply() and the built-in function mean()
2. Using apply() and a function you write, called mymean(), which takes the sum of all of the non-missing values and divides by the number of non-missing values, without using the built-in function mean().
3. Using a for() loop to iterate over the numeric columns, and a for() loop inside it to iterate over the values within that column, without using the built-in functions mean() or sum().

Which do you expect to be most efficient?  Explain your answer in 1-3 sentences.

Disclaimer: Apply would probably not work on the structure (data frame) I read the data in as. The method will look to coerce the object to a matrix, and since there are strings in the data frame (particularly in the first column) the matrix will end up as type string. The matrix structure is homogenous, which means all the datatypes in the structure must be the same. Instead, we can use lapply().

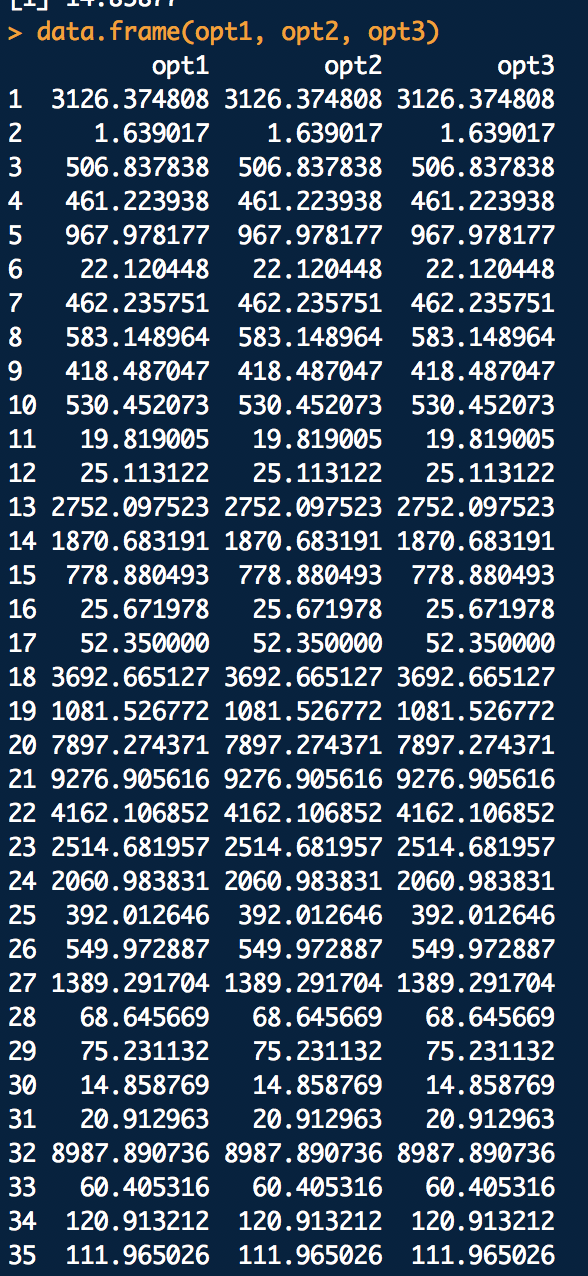
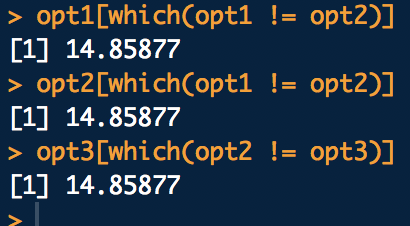
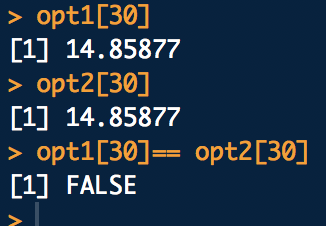
Generally, I would expect any build in R function to be faster than a function I’d write myself. This is more of a statement about the efficiency of my own code than it is about the quality of built-in R functions, though the built in functions are usually efficient. I would expect the for() loop to be the slowest since it will be repeating an operation many times.

**1.d Write functions for each of the 3 methods in part c. Apply them to the US News and World Report data and check that all 3 methods give the same answers.**





Lets inspect the one value that is wrong across all vectors.



By visual inspection, all values are equal.

**1.e Use *microbenchmark* to compare the median running time of the methods in part c.  Write 1-3 sentences describing which method is most efficient.**

As expected, the for loop which has very high repetition of tasks is the slowest (attention to the units for these measurements). Surprisingly, mymean() was faster than the build in mean function. This is likely due to some of the additional functionality provided by built in mean(), such as its ability to trim the dataset and method of handling NA values. It is not surprising that both option 1 and option 2 were faster than the for() loop, given the amount of repeated tasks in the for loop.

