Lecture Notes on Neural Networks: Part 2B, Import Into R, Trim Rows, Encode Easy Columns

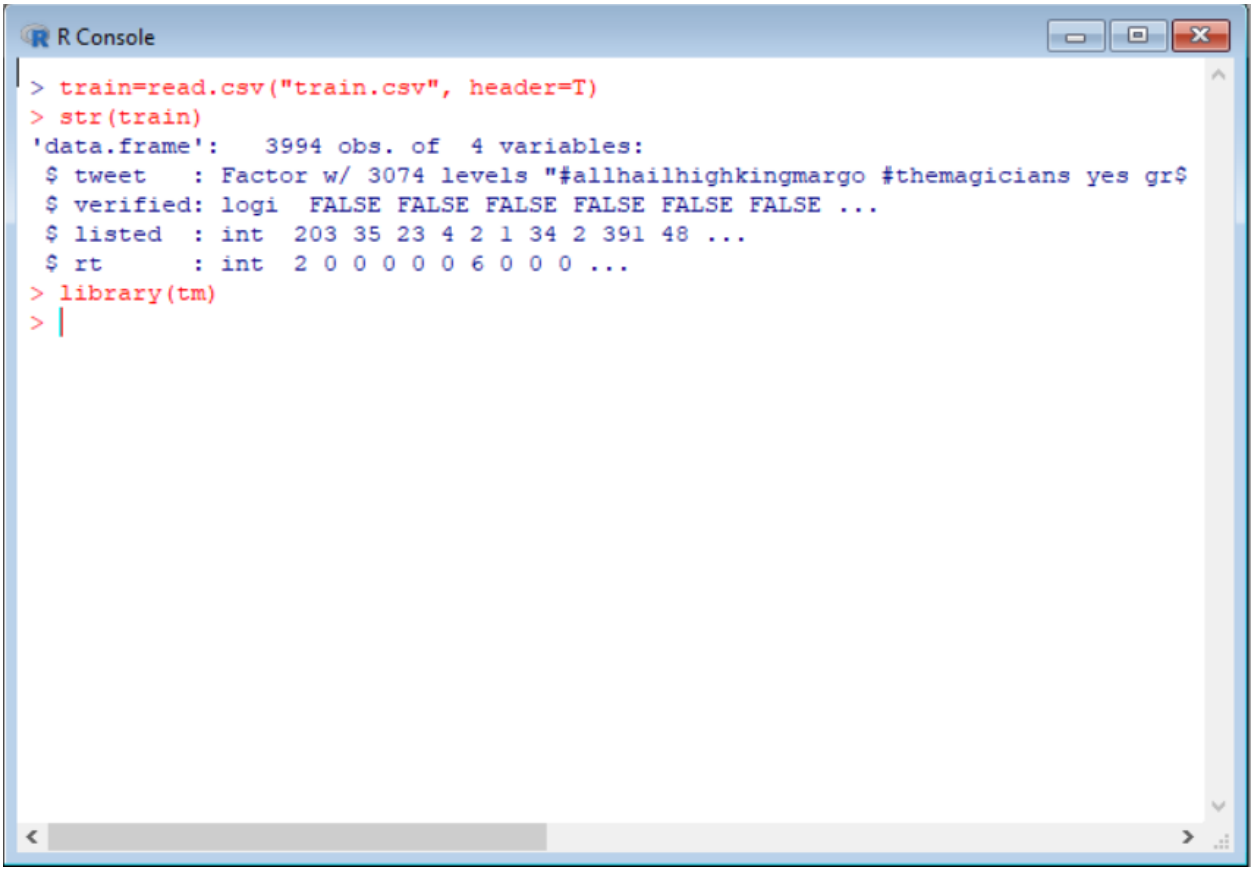
Nick V. Flor, nickflor@unm.edu

In the previous tutorial, we did some data cleaning, and copied into a new sheet just the columns of data we would use as inputs (tweet, listed, verified) as well as output (rt). We saved the file as *train.csv*.

In this tutorial, we'll import the file into R, then encode the "easy" columns. The easy columns are any column other than the tweet column—because converting text into numbers is not easy.

PART B. IMPORT INTO R, TRIM ROWS

Step 1. Import into R



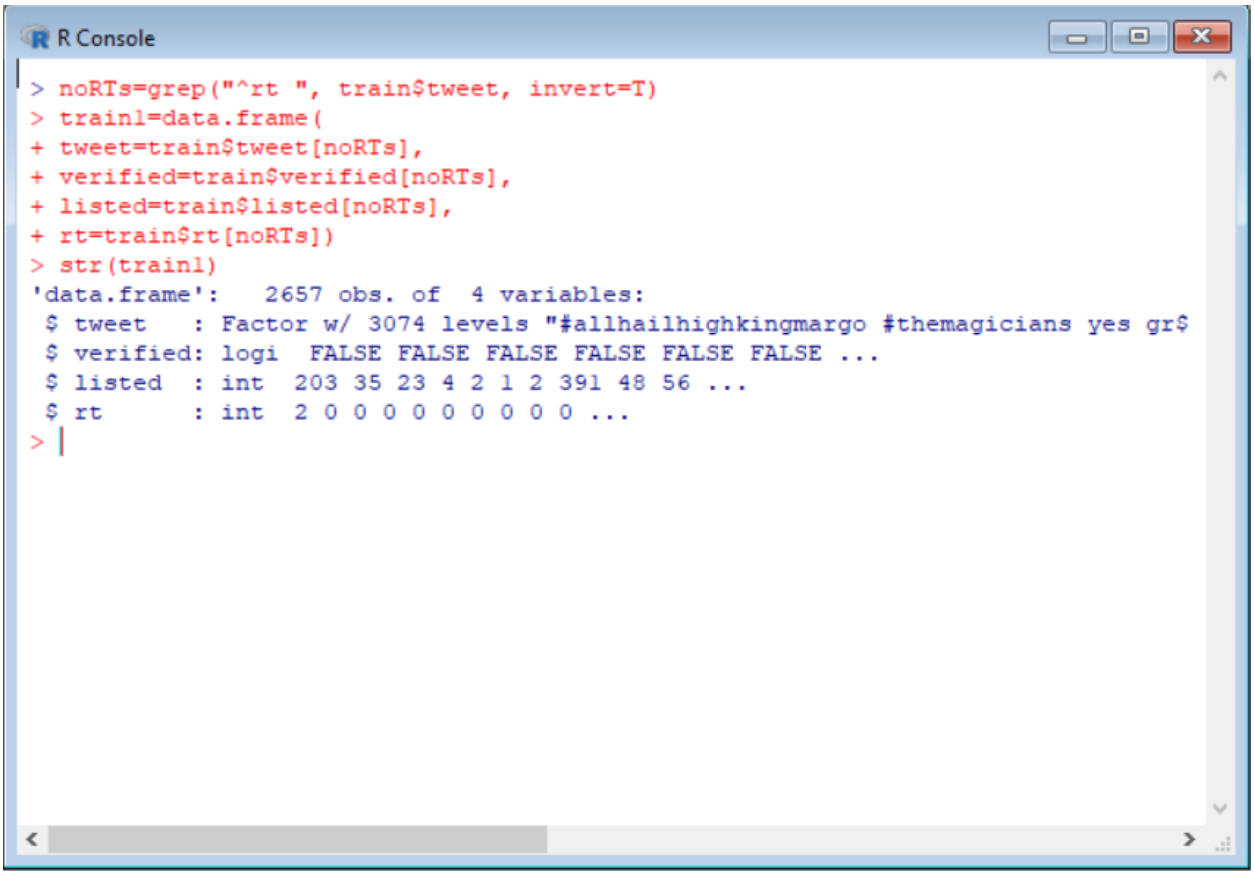
Explanation:

I used *read.csv* to import into R, *str* to check the structure, and then *library* to load the text mining package (tm). Remember, we did an *install.packages("tm")*, early in the semester during the text-mining lectures.

<u>Bottom Line</u>: *train* contains our spreadsheet.

(continued on the next page)

Step 2. Trim Rows and Create a New Data Frame



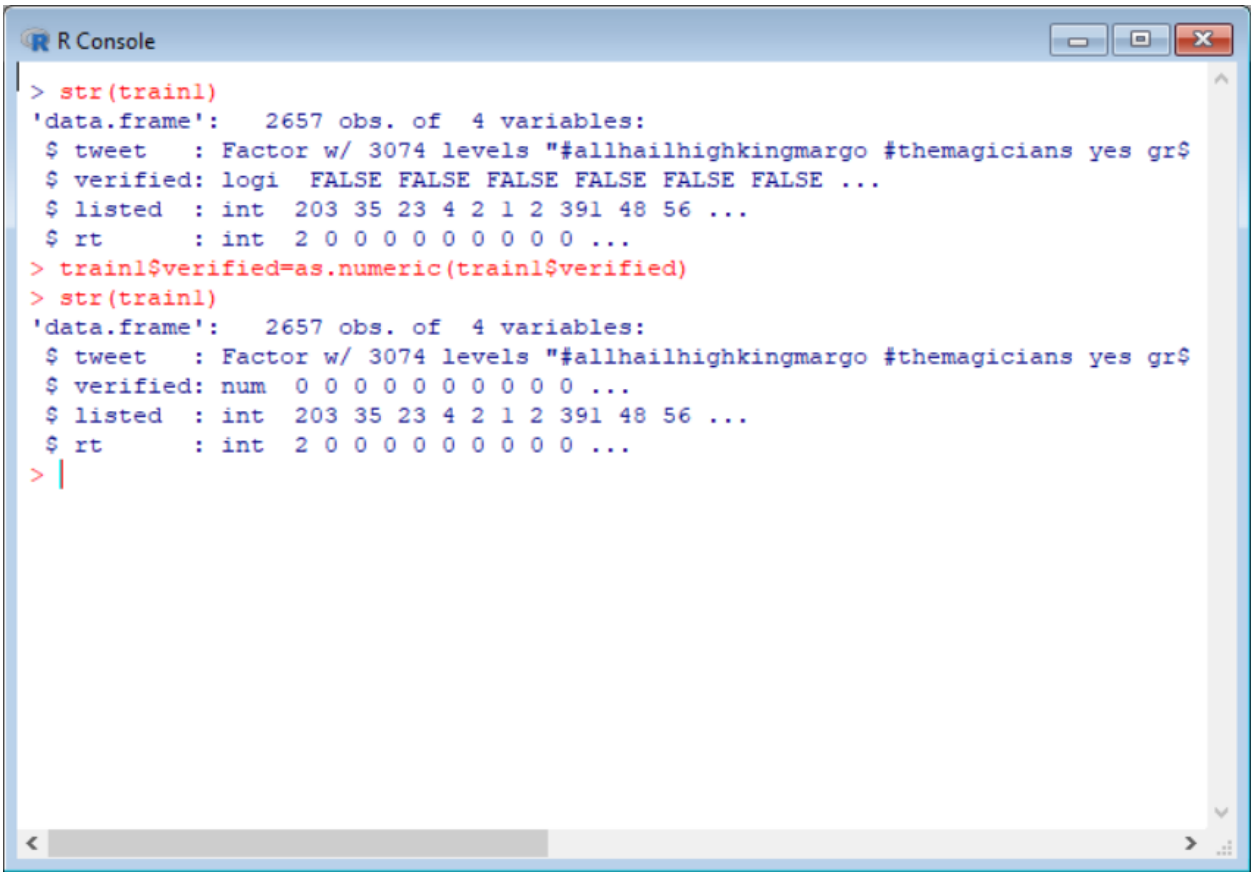
Explanation:

We do not want to train our network on retweets. A retweet always starts with the letter *rt* (*"^rt "* as a regular expression); the *grep* returns the row number of any tweet which does not (*invert=T*) start with an rt. I store these row numbers in the variable *noRTs*.

Finally, I create a new *data.frame* named *train1*, with the same column headings as *train* (*tweet, verified, listed, rt*) but the *[noRTs]* means the tweets are not retweets.

Bottom line: *train1* is *train* without the retweets.

Step 3A. Encode Booleans (TRUE/FALSE) as 1/0



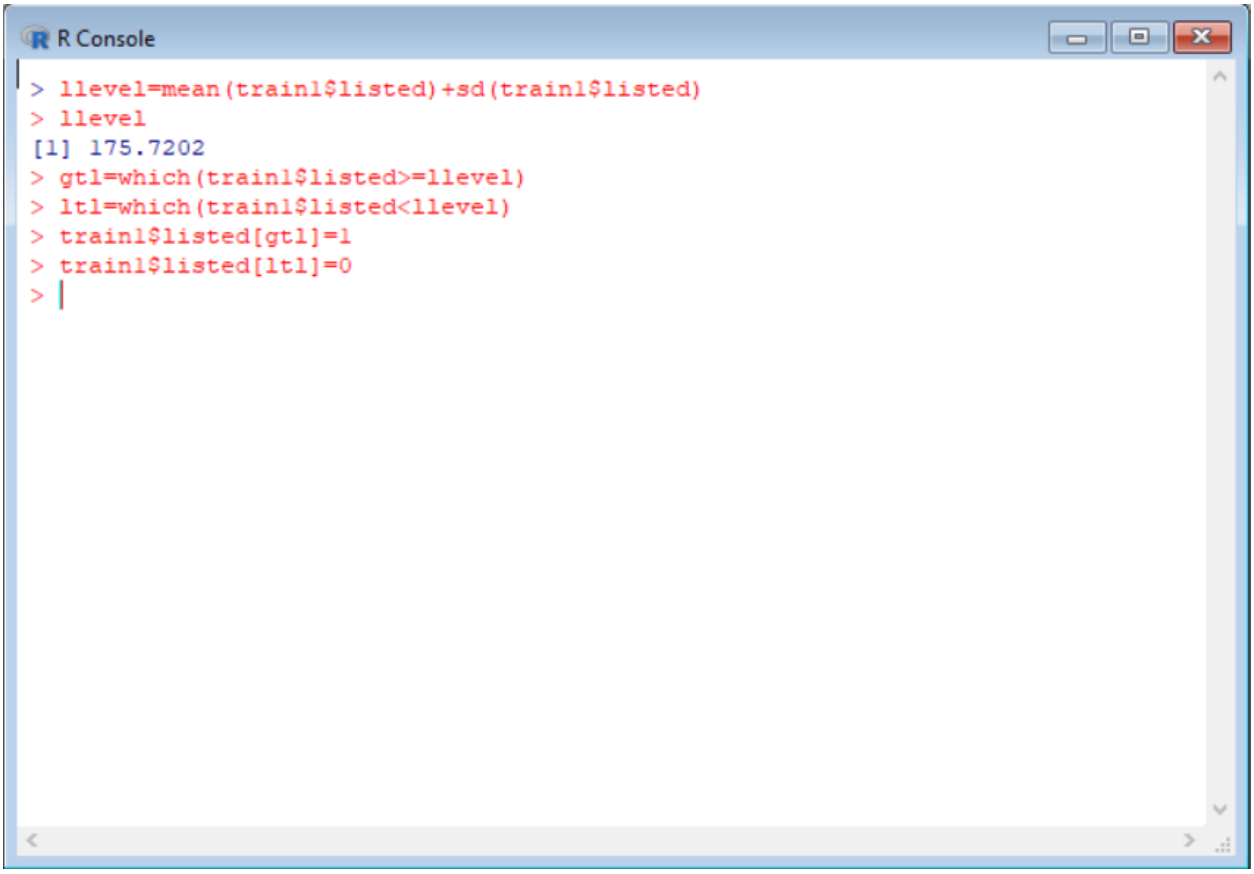
Explanation:

The function *as.numeric* coerces TRUE/FALSE to 1/0. I call *str* before and after *as.numeric* just to show you the effects.

Bottom Line: Whenever you want to convert a column of Booleans to 1's and 0's, use *as.numeric*.

(continued)

Step 3B. Encode the listed column as 1 if high, 0 otherwise



Explanation:

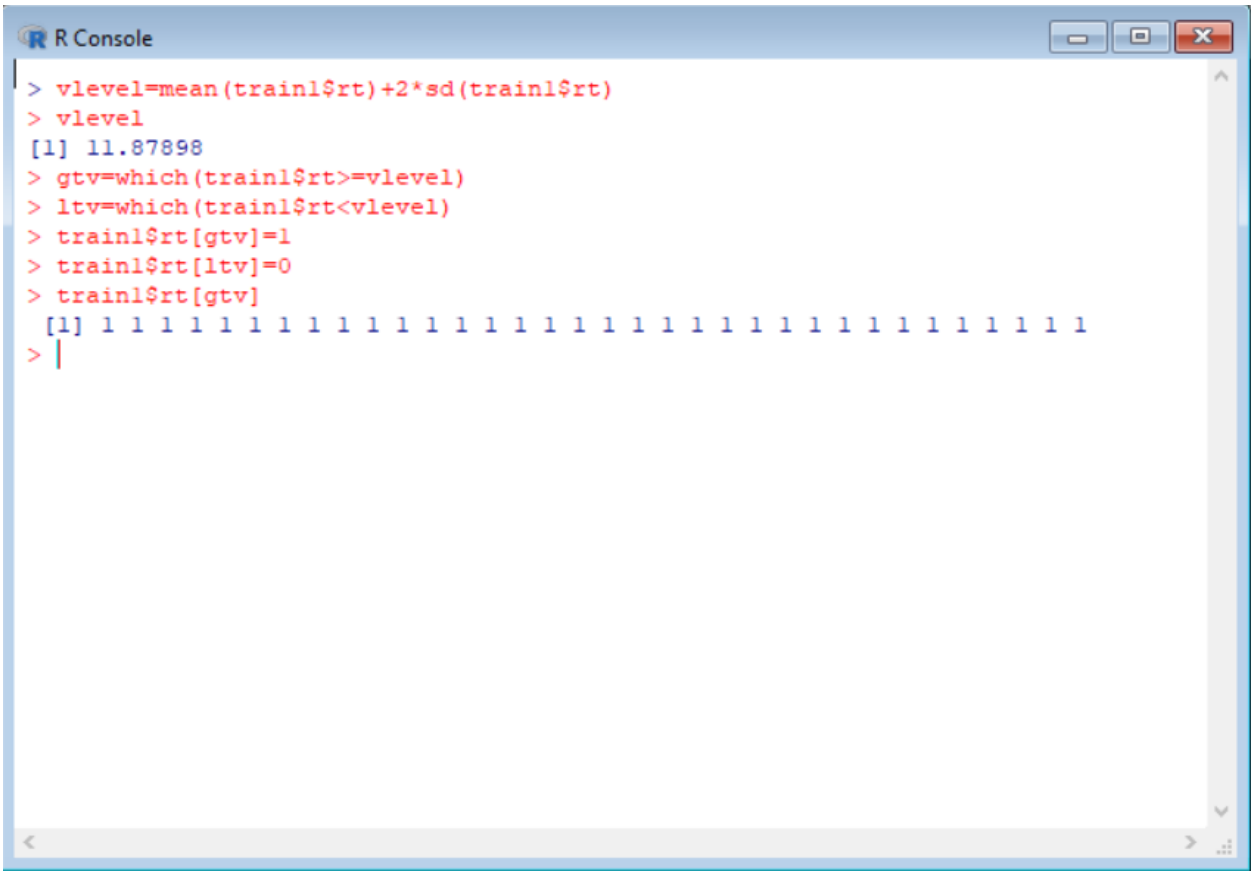
What does it mean for a user to have a high level of being listed? Instead of arbitrarily picking a cut-off, I decided to define one standard deviation (*sd*) from the *mean* as the cut-off. I stored this cut-off in *llevel*. Thus, any users listed more than 175.72 times has a high listing (1), anything less than that is a low listing level (0).

I used the *which* function to return the listed row numbers greater than *llevel* (*gtl*). I also used the *which* function to return the listed row numbers less than *llevel* (*ltl)*.

Finally, I set all rows greater than *llevel* to 1, and all rows less than *llevel* to 0.

Bottom Line: the listed values for all rows greater than *llevel* are now 1 (vs say 200), otherwise 0.

Step 3C. Encode RT column as 1 if Viral, 0 if Not Viral



Explanation:

What does it mean for a tweet to be viral? We are using the number-of-retweets (column *rt*) to determine whether a tweet is viral. But what is our cut-off level? Instead of arbitrarily picking a cut-off, I decided to define two standard deviations (*sd*) from the *mean* as the cut-off. I stored this cut-off in *vlevel*. Thus, any tweet with more than 11.87 retweets is viral, anything less than that is not viral.

I used the *which* function to return the row numbers of retweets greater than *vlevel* (*gtv*). I also used the *which* function to return the row numbers of retweets less than *vlevel* (*ltv)*.

Finally, I set all rows greater than *vlevel* to 1, and all rows less than *vlevel* to 0.

<u>*Bottom Line*</u>: all rows greater than *vlevel* are now 1 (instead of say 94), otherwise 0.