Texts, Functions, and Probabilities: Problem Set

Ben Schmidt

February 19, 2015

1. Find a feature in the "titles" or "whaling crews" data set for which Zipf's law holds. Plot it using the text element we've used and `geom_point` which should present a clearer path.

2. Certain words always appear in the same immediate context: for example, "dearth" is almost always followed by the word "of." Use the SOTU concordance frame to find words that are most often preceded by the same word before them as a *share* of the times they appear. (Note the general strategy in the probability section for calculating a local probability: group by different elements and use the `n()` function to see how often particular combinations appear.)

3. (Challenge–skippable) Write a function called `concordance` that, fed a text data frame with a column called words and an integer `n`, uses `lead` to add `n` columns. For instance, given the Declaration of Independence in `data.frame` format, `concordance(declaration,n=4)` would return:

```
    word  word2  word3   word4
1   When     in    the  course
2     in    the course      of
3    the course     of   human
4 course     of  human  events
5     of  human events      it
6  human events     it becomes
```

```
concordance = function(frame,n) {
  #Answer here.
}
```

4. Copy and paste together the various functions we wrote to read in all the files of the State of the Unions: rewrite it as a function `collect_directory_into_text_frame` that will take a directory name, and then read in all the text files in it into a data.frame. A sample folder of works by Charles Dickens is supplied. But you can (and should) be able to run it on any set of texts that you like. Project Gutenberg is a good source of free public domain texts, but there are many others out there.

And let me emphasize–**copy and paste the code from the handout.**. You'll need to change one or two things (the directory name) and understand the order to execute in, but this should require you to write almost no new code. (You'll have to delete the bit that assigns a year to the state of the union, but not much else.) Don't worry if you end up having internal variables in your function that are called things like "allSOTUs"; what matters is what comes out of the function, not what it looks like internally. Writing new code when existing code exists is *bad practice*.

Likewise, if you can wrangle the code for this from a classmate, feel free–if you do, though, acknowledge them and try to use a different folder of texts than them.

```
collect_directory_into_text_frame = function(dirname) {

}
```

5. Using some source of text in this format, at least begin to try implementing one of Ramsay's many potential forms of reading. (An easy one would be the "entropic version" of the poem on page 37); a harder one would be the TF-IDF lists of the waves but applied to a different group of texts.

6. Create a random walk generator like the State of the Union one that will operate on one of these new sources of texts.

A random Dickens generator is an obvious choice. But I'd be particularly interested to see one that generates random titles from the titles set, if you can figure out how to reformat that data as text.

Bonus points if you can think about changing the regular expression splitter in your version to include punctuation.

Run it for a while, and then write a blog post **briefly** describing your source texts and illustrating some of the most interesting portions of the walk online.

If you want to go nuts, make it into a Twitter bot.