18 Jan 2018 - Most Frequent Words in a Text

In the previous version of this writeup, I tried to stick to the procedure explained in your textbook (chapter 2) making minor modifications for better coding. Today, I am going to show you how to read any gutenberg book in plain text format into R environment, calculate the number of characters, words, and plot the frequencies of words in the text. I will still use "A Doll's House", a play by a Norwegian author Henrik Ibsen for illustration, but it should work well with any text from Gutenberg.org (and also perhaps for any language).

Please note: I am writing this under the assumption that you already attended the class before this, and also went through the earlier version of this tutorial.

First, set your working directory to where your files are, using setwd(). You may need "stringr" library, so load that library

```
library(stringr)
```

Once you do these, first task is to "Load" the file I want into my R environment so that I can start analysing them. We do this using the scan function in R (there are other ways too. We will discuss later.)

```
english <- scan("DollsHouse-Eng.txt", what = "character", sep = "\n")
```

What we are doing here is saying R to read these files character by character, and breakup the text into lines based on wherever a newline character is seen in text. Newline character is indicated by "\n" and store each file as a character vector. Is the variable "english" a vector? How do we find out? [hint: isvector(english) will give you this information.]

```
english[1]
```

```
## [1] "The Project Gutenberg EBook of A Doll's House, by Henrik Ibsen"
```

```
english[34]
```

```
## [1] "and on the same side, nearer the footlights, a stove, two easy chairs"
```

-these print the first and 34th lines of english variable.

There is a lot of metadata at the beginning and end of Gutenberg.org texts which indicate licensing information and other such stuff. We don't need it here. However, they all seem to follow a certain format. The last line of the meta-data before text starts with: "*** START OF THIS PROJECT GUTENBERG EBOOK" and the first line after the book ends starts with: "End of the Project Gutenberg EBook" (Note the case differences). We can use this information to remove metadata in gutenberg.org ebooks.

which() is a function in R, that will help us here. Given a vector, and given a logical expression, which gives us where this expression holds true in this vector. Try this out before we go further:

```
something = LETTERS #what is in this something now? type something and press enter to check out!
which(something == "A")
```

```
## [1] 1
```

```
which(something == "Y")
```

```
## [1] 25
```

These two lines return 1 and 25 respectively, because those are the locations of the strings "A" and "Y". "==" is a symbol used for "comparison". The output of a == b kind of expressions is always a True or False.

Let us get back to our question. So, we know how it looks just before the start of the book and right after the end. We can use the which function to get those line numbers. However, there is a small issue. The above which example, and the one we used in last class, both give the full exact line in the quotes. We cannot do that here, if we want this to work with all gutenberg texts (Why?). Thankfully, there is a function for strings in R, called startsWith(string, pattern), which returns a True or False, depending on whether a string of words start with a given pattern

```
meta.top.end <- which(startsWith(english,"*** START OF THIS PROJECT GUTENBERG EBOOK"))
meta.bottom.start <- which(startsWith(english, "End of the Project Gutenberg EBook"))
actual_english <- english[meta.top.end+1:meta.bottom.start+1]
```

Why did I add +1?

actual_english now contins the actual play in English, with all the line breaks preserved. However, if our task is to just have a look at words and frequencies, we don't need line breaks. We can use the paste() function to eliminate line breaks and convert the whole text into one string, and then lowercase everything as before.

```
actual_english_string <- paste(actual_english, collapse = " ")
english_lower <- tolower(actual_english_string)
```

Why did I use "collapse" instead of "sep"? A simple answer is- paste with a collapse returns vector with one long string. But using with sep option returns a vector with multiple smaller strings (See the end of this tutorial for example). We need the text as one full string first, as we want to split it into words by removing punctuation, and extra white spaces.

Next task is to split the string into words. "\W+" is a regular expression that matches "one or more non word characters" in a long string. We use that to split the string into words, such that all punctuation markers and white spaces are removed, and not considered for our further calculations.

```
english_words <- strsplit(english_lower, "\\W+")
```

Now, from here, I convert it into word-frequency table like this:

```
sorted_freqs_english <- sort(table(english_words), decreasing = TRUE)
```

We can get the frequency of individual words in these texts using the below commands:

```
sorted_freqs_english["nora"]
```

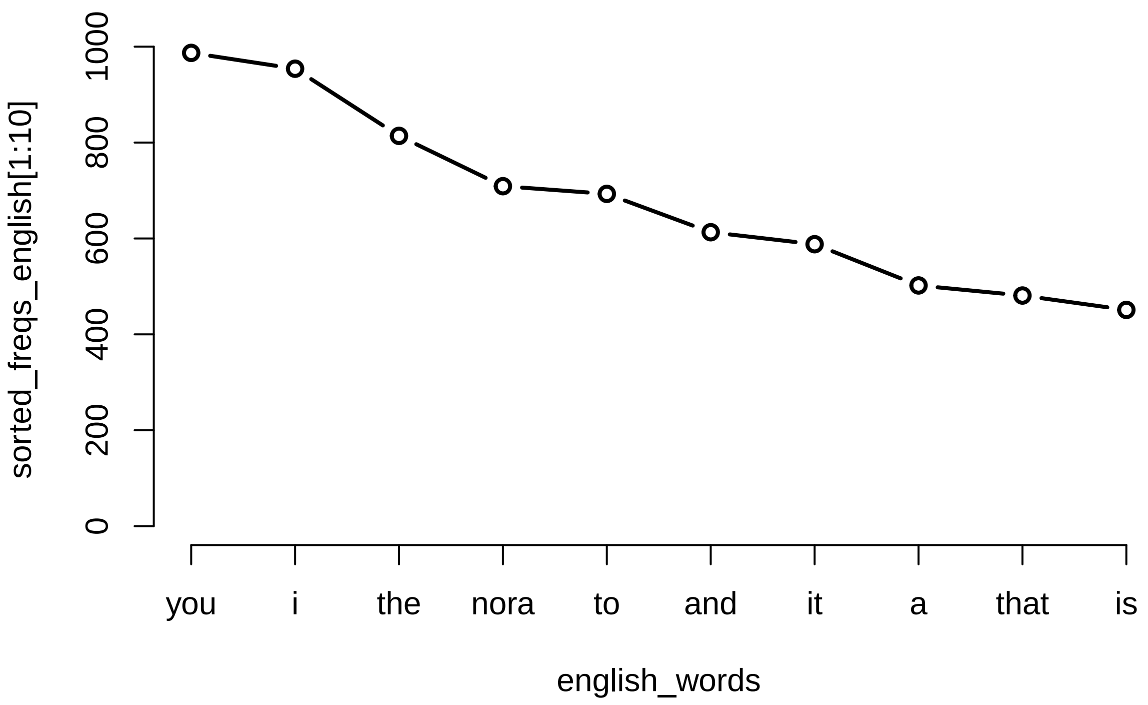
```
## nora
##  709
```

```
sorted_freqs_english["helmer"]
```

```
## helmer
##    332
```

Now to the final plots (type = b indicates join the points by lines.)

```
plot(sorted_freqs_english[1:10], type="b")
```



An aside to demonstrate collapse vs sep in paste:

```
v1 = c("LING","ENGL","ENG","JLMC")
v2 = c("school","college","university")
paste(v1,v2)
```

```
## [1] "LING school"    "ENGL college"   "ENG university" "JLMC school"
```

```
paste(v1,v2,collapse=",")
```

```
## [1] "LING school,ENGL college,ENG university,JLMC school"
```

```
paste(v1,v2,sep=",")
```

```
## [1] "LING,school"    "ENGL,college"   "ENG,university" "JLMC,school"
```

```
length(paste(v1,v2,collapse=","))
```

```
## [1] 1
```

```
length(paste(v1,v2,sep=","))
```

```
## [1] 4
```