LING 410X: Language as Data

Semester: Spring '18

Instructor: Sowmya Vajjala

Iowa State University, USA

8 February 2018

Class Outline

- writing R functions
- Rmarkdown: introduction and practice exercise
- Reminder: Assignment 2 due on 10th!

Interesting stuff I found

- a 1994 article that did text analysis on the "Book of Genesis"! `http://projecteuclid.org/download/pdf_1/euclid.ss/1177010393`

Writing R functions

What is a function? Why use it?

- Functions are reusable pieces of code you can just "call" and use instead of writing everything line by line.
- sort(), unlist(), table() and all these things you saw are such "built-in" functions.

What is a function? Why use it?

- Functions are reusable pieces of code you can just "call" and use instead of writing everything line by line.
- sort(), unlist(), table() and all these things you saw are such "built-in" functions.
- Let us say you want to sort a list of words and frequencies in decreasing order.
- You don't now start writing code for the sorting process from scratch.
- You just use sort() function, and you can use it again and again.
- If we already have so many functions in R, why write new ones?

What is a function? Why use it?

- Functions are reusable pieces of code you can just "call" and use instead of writing everything line by line.
- sort(), unlist(), table() and all these things you saw are such "built-in" functions.
- Let us say you want to sort a list of words and frequencies in decreasing order.
- You don't now start writing code for the sorting process from scratch.
- You just use sort() function, and you can use it again and again.
- If we already have so many functions in R, why write new ones?
- To do some custom tasks that we want, for which some such function does not already exist in R.

Functions vs Loops - your responses

- In some sense, a loop is also some sort of function to repeatedly do something.
- A loop can call some functions repeatedly inside it
- A function can have loops inside it

Functions vs Loops - your responses

- In some sense, a loop is also some sort of function to repeatedly do something.
- A loop can call some functions repeatedly inside it
- A function can have loops inside it
- Difference is that: your function, once defined and saved somewhere on your computer, can be "imported" into any R code you are writing at a later point in time, and used as if it is a built-in R function.

Functions vs Loops - your responses

- In some sense, a loop is also some sort of function to repeatedly do something.
- A loop can call some functions repeatedly inside it
- A function can have loops inside it
- Difference is that: your function, once defined and saved somewhere on your computer, can be "imported" into any R code you are writing at a later point in time, and used as if it is a built-in R function.
- With loops, you cannot do that unless you put that loop inside a function.

Writing a function - Example

```
my_square_function <- function(number)
{
  return(number * number)
}

my_square_function(4) #Gives 16
```

Writing a function - Example

```
my_number_function <- function(number)
{
  return (c(number*number, number*number*number, number*number*number*number))
}

my_number_function(4) #Gives a vector with values 16, 64, 256
```

So, how do I use functions?

- Let us take this example from last week, for creating dispersion plots.
- When I wanted to create a dispersion plot for the word ”nora” in Dolls House, here is what I did:

```
progress <- seq(1:length(dollshouse_words_vector))
nora <- which(dollshouse_words_vector == "nora")
length(nora)
nora_progression <- rep(NA, length(progress))
nora_progression[nora] <- 1
plot(nora_progression, main="Dispersion plot for word ’nora’ in ’A Doll\’s House’ play",
    xlab="position in text", ylab="nora", type="h", ylim=c(0,1), yaxt = ’n’)
```

Where are functions useful?

When I wanted to repeat the same process for another word ”rank”, here is what I had to do:

```
progress <- seq(1:length(dollshouse_words_vector))
rank <- which(dollshouse_words_vector == "rank")
rank_progression <- rep(NA, length(progress))
rank_progression[rank] <- 1
plot(rank_progression, main="Dispersion plot for word ’rank’ in ’A Doll\’s House’ play",
     xlab="position in text", ylab="rank", type="h", ylim=c(0,1), yaxt = ’n’)
```

If I had a third word, I should do all these again. What if I can put this piece of code as a custom function, and use that function wherever I want?

Writing a function for dispersion plot

```
get_dispersion_plot <- function(wordsvector, word)
{
  progress <- seq(1:length(wordsvector))
  word_presence <- which(wordsvector == word)
  length(word_presence)
  word_progression <- rep(NA, length(progress))
  word_progression[word_presence] <- 1
  plot(word_progression, main="Dispersion plot for the given word in ’A Doll\’s House’ play",
    xlab="position in text", ylab=word, type="h", ylim=c(0,1), yaxt = ’n’)
}
```

Now, I can use this dispersion function to get the plot for any word, once I create that vector of words.
- Note that this is not a loop.

A function for getting the word vector from a file

```
get_words_vector <- function(file_path)
{
  fulltext <- scan(file_path, what = "character", sep = "\n")
  fulltext_as_string <- paste(fulltext, collapse = " ")
  words_vector <- unlist(strsplit(tolower(fulltext_as_string), "\\W+"))
  return (words_vector)
}
```

Using these two functions

- Let us say I put these functions in a file called TextAnalFunctions.R.
- Using these functions in R console follows the following steps:
  1. Set the working directory to where your file is (else, use the full path)
  2. First, I "source" or load the R file with my functions
  3. use get_words_vector to get words vector for the file
  4. use that words vector from the previous step to get dispersion plot for a given word from this vector.
- in R

```
source("TextAnalFunctions.R").
words <- get_words_vector("DollsHouse-Eng.txt")
get_dispersion_plot(words, "nora")
#or whatever word you want.
```

Functions: Conclusion

- Functions also make your R code more readable.
- When to use (and whether to use) functions - depends on what you are doing.
- More examples: `http://www.statmethods.net/management/userfunctions.html`

Using functions to find Hapax Legomena (Chapter 7)

Hapax Legomena

- We looked at average word frequency and TTR on Tuesday.
- Another way of looking at vocabulary richness is to look at the number of words that occur very infrequently in the text.
- If we consider words that appeared only once, we call them singleton/one-zies/hapax legomena
- How do you get such information? There is no such pre-defined function like mean() or sum() to return frequencies that are 1.

sapply, with custom function definition

Consider this line below (chapters.raw - is the variable from our last class):

```
hapax <- sapply(chapters.raw, function(x) sum(x == 1))
```

Alternatively, the following also works:

```
hapaxfunction <- function(x)
{
  return(sum(x ==1))
}
hapax <- sapply(chapters.raw, hapaxfunction)
```

-What this says is: for each item in chapters.raw, i.e., for each chapter, count the number of words whose

frequency is 1.

From what we saw so far...

What will these lines below do?:

```
hapax <- sapply(chapters.raw, function(x) sum(x == 1))
hapax <- unname(hapax)
lengths <- unname(sapply(chapters.raw, sum)) #What will this have?
new <- hapax/lengths
```

-What will "new" have eventually??

R Markdown

What is R Markdown?

- R markdown is a formatting system to create HTML, PDF, or Word reports that use R code.
- Why use it?: Typically used to share your R based analysis and results with others. Supports reproducible research.
- Advantage: R code can be embedded inside the report. So, you can just keep adding R code, its output, and your comments, and prepare a neatly formatted report.

What is R Markdown?

- R markdown is a formatting system to create HTML, PDF, or Word reports that use R code.
- Why use it?: Typically used to share your R based analysis and results with others. Supports reproducible research.
- Advantage: R code can be embedded inside the report. So, you can just keep adding R code, its output, and your comments, and prepare a neatly formatted report.
- This is how all those tutorial documents I shared have been created.

How to start?

- First, install the Rmarkdown package by going to tools$-$ $>$install packages and selecting rmarkdown or by typing: install.packages(”rmarkdown”) on R console.

How to start?

- First, install the Rmarkdown package by going to tools− >install packages and selecting rmarkdown or by typing: install.packages(”rmarkdown”) on R console.
- Now, in your R studio, go to File − > New − > Rmarkdown. Give some title to your report, write your name, and create a document.

How to start?

- First, install the Rmarkdown package by going to tools− >install packages and selecting rmarkdown or by typing: install.packages("rmarkdown") on R console.
- Now, in your R studio, go to File − > New − > Rmarkdown. Give some title to your report, write your name, and create a document.
- There are three major types of data in this - it starts with some metadata about the document, and then there is either Rcode or some plain text.
- Resource: `http://rmarkdown.rstudio.com/lesson-1.html`

R markdown exercise

- With this elementary introduction to Rmarkdown, do a small exercise (individually or as groups)
- Prepare a R markdown report that shows the top ten words in any text file you take as input (sample text file provided)
- It need not necessarily be from Gutenberg.org - so, you don't have to handle the removal of metadata part.
- Note: We already did this before, and you also have my Tutorial document as a reference to do this.
- Once you are done, go to Canvas course page, open Discussion forums, look for a forum titled 8thFeb2018 and post your doc (or html or pdf) reports there along with the .Rmd file you created.

Next Week

- Text analysis topics:
  - Searching for keywords and their contexts inside texts (Read chapters: 8–9 in textbook)
  - Moving beyond single words and looking at word pairs, word triplets etc.
- R specifics: creating working with .R files. Continuing to work with R functions.
- For those who want to do more:
  1. Convert all text analyses we did so far into functions and store them in one file, calling it text analysis or something.
  2. Use loops to loop through all files in a folder, and repeat whatever text analyses you want by calling these functions in the loop.