LING 410X: Language as Data

Semester: Spring '18

Instructor: Sowmya Vajjala

Iowa State University, USA

6 February 2018

Class Outline

- Recap of last week
- A note on the various data structures in R
- Calculating lexical richness in a text (Chapters 6 in Textbook)
- Reminder: Assignment 2 submission is due on 10th Feb.
- news: `https://www.r-bloggers.com/analysis-of-trumps-state-of-the-union-speech-with-r/`

Recap of last week-1

- Text analysis questions: Analyzing word usage in different parts of a text (chapter by chapter, word by word etc)
- R specific concepts:
  1. Differences between a vector and a list
  2. Usage of lapply
  3. Writing a for- loop
  4. Replacing NAs with zero

Recap of last week-2

My Question: Using R program, try to find out:

- How many chapters are there in David Copperfield (`http://www.gutenberg.org/files/766/766-0.txt`)?
- What chapter has the most number of occurrences of the name Murdstone?

Recap of last week-2

My Question: Using R program, try to find out:

- How many chapters are there in David Copperfield (`http://www.gutenberg.org/files/766/766-0.txt`)?
- What chapter has the most number of occurrences of the name Murdstone?
- My solution for 1st question (I actually do not need all that other stuff to answer just this question!)

  ```
  davidc <- scan("davidc.txt", what = "character", sep = "\n")
  davidc.chap.positions <- grep("^CHAPTER \\d", davidc)
  length(davidc.chap.positions)
  ```
- For second part, Brody will explain how he found out.
- Additionally, you can check: 01FebSolution.R for today's canvas (.. and modify it according to your specifications)

Data Structures in R

What is a data structure?

- a datastructure refers any way of organizing data.
- examples: a list, a vector, a matrix, a table etc.
- purpose of a data structure is to store data in some organized manner such that it is easy to access it later
- What data structure you need will depend on how your input and output should look like.
- When we use pre-defined R functions such as sort, lapply etc., we should know what do they take as input, and what they give as output.

R Datastructures: Lists and Vectors

- What is a vector?
- What is a list?
- What is the difference between vector and list?

R Datastructures: Lists and Vectors

- What is a vector?
- What is a list?
- What is the difference between vector and list?
- What does a sort() function take as input? a vector or a list? How can I find out?

R Datastructures: Lists and Vectors

- What is a vector?
- What is a list?
- What is the difference between vector and list?
- What does a sort() function take as input? a vector or a list? How can I find out?
- What does seq(1:5) return? a list or a vector? How can I find out?

R Datastructures: Matrices and Arrays

- A matrix is a 2 dimensional datastructure (arranged in rows and columns) where all elements are of the same data type (integers or strings or decimal numbers etc)
- If we extend this idea to more than 3-dimensions, we have a data structure called array.
- How does a matrix look like? Try this: matrix(1:6, ncol=3)
- How does an array look like? Try this for a 3-D array: array(1:8, dim=c(2, 2, 2))

R Datastructures: Data Frames

- A "Data Frame" is useful to combine several vectors together as columns in a table.
- We use data.frame() function to create a data frame.

R Datastructures: Data Frames

- A "Data Frame" is useful to combine several vectors together as columns in a table.
- We use data.frame() function to create a data frame.
- Let us look at this example:

```
line_number <- seq(1:5)
names <- c("Santa", "Banta", "Manta", "Tanta", "Anton")
college <- c("LAS", "Business", "Engineering", "LAS", "VetMed")
df <- data.frame(line_number,names, college)
```

R Datastructures: Factors

- A "Factor" is a data structure useful to store categorical data (what is that?)

R Datastructures: Factors

- A "Factor" is a data structure useful to store categorical data (what is that?)
- In my previous data frame example, the column college has 5 values, but 4 unique values.
- The 4 unique values can be seen as different factors/categories of colleges.
- We use factor() function to work with factor data structure.
- Example usage: factor(df$college) from previous slide.

Lexical Variety

What is lexical variety or vocabulary richness

- Lexical variety refers to some quantification of the diverseness of the vocabulary used in a text/corpus.
- There are two common ways of defining lexical variety:
  - Mean word frequency: average frequency of words in a document.
  - Type-token ratio: (number of unique words/total number of words)*100

What is lexical variety or vocabulary richness

- Lexical variety refers to some quantification of the diverseness of the vocabulary used in a text/corpus.
- There are two common ways of defining lexical variety:
  - Mean word frequency: average frequency of words in a document.
  - Type-token ratio: (number of unique words/total number of words)*100
- Let us say this is the text I have: "Here is a sentence with sentence coming twice".
- What is the mean-word frequency? What is the type-token ratio?

What is lexical variety or vocabulary richness

- Lexical variety refers to some quantification of the diverseness of the vocabulary used in a text/corpus.
- There are two common ways of defining lexical variety:
  - Mean word frequency: average frequency of words in a document.
  - Type-token ratio: (number of unique words/total number of words)*100
- Let us say this is the text I have: "Here is a sentence with sentence coming twice".
- What is the mean-word frequency? What is the type-token ratio?

Calculating Lexical Variety

Let us start from last week's example again.

```
moby <- scan("mobydick.txt", what = "character", sep = "\n")
moby.start <- which (moby ==  "CHAPTER 1. Loomings.")
moby.end <- which (moby == "orphan.")
moby.actual <- moby[moby.start:moby.end]
moby.chap.positions <- grep("^CHAPTER \\d", moby.actual)
moby.actual[moby.chap.positions]

moby.last.position <- length(moby.actual)
moby.chap.positions <- c(moby.chap.positions, moby.last.position)
moby.actual[moby.chap.positions]

chapters.raw <- list()
for (i in 1:(length(moby.chap.positions) -1))
{
  titleline <- moby.chap.positions[i]
  title <- moby.actual[titleline]
  start <- titleline+1
  end   <- moby.chap.positions[i+1]-1
  chapter.lines <- moby.actual[start:end]
  chapter.string <- tolower(paste(chapter.lines, collapse = " "))
  chapter.string <- gsub(" +", " ", gsub("[[:punct:]]", " ", chapter.string))
  chapter.words <- unlist(strsplit(chapter.string, "\\W"))
  chapter.freqs <- table(chapter.words)
  chapters.raw[[title]] <- chapter.freqs
}
```

Mean Word Frequency

- Let us say I want to calculate mean (average) word frequency for each chapter in Mobydick.
- What information do I need for this?

Mean Word Frequency

- Let us say I want to calculate mean (average) word frequency for each chapter in Mobydick.
- What information do I need for this?
- Okay, let us take one chapter as an example.

```
sum_of_all_word_freqs_chapter1 <- sum(chapters.raw[[1]])
num_words_in_chapter1 <- length(chapters.raw[[1]])
mean_word_freq_chapter1 <- sum_of_all_word_freqs_chapter1/num_words_in_chapter1
another_way_for_this <- mean(chapters.raw[[1]])
ttr_chapter1 <- (num_words_in_chapter1/sum_of_all_word_freqs_chapter1)*100
```

- If we do this for all chapters, we get the mean word frequency per chapter.

How do we get mean word frequency for all chapters?

- Use a for loop like last week:

```
means = c()
for(i in 1:length(chapters.raw))
{
 means[i] <- sum(chapters.raw[[i]])/length(chapters.raw[[i]])
}
```

How do we get mean word frequency for all chapters?

- Use a for loop like last week:

```
means = c()
for(i in 1:length(chapters.raw))
{
 means[i] <- sum(chapters.raw[[i]])/length(chapters.raw[[i]])
}
```

- Use lapply: lapply(chapters.raw, mean)
- What does this give?

unname and unlist

- lapply returns a list. unlist() converts it into a vector (plotting function in the next slide looks for a vector).
- unname() removes the chapter names and retains only the numbers (Not needed here, Just doing to show this)

```
means <- unlist(means)
means <- unname(means)
```

- At this point, means is just a vector of numbers, which we can use to plot.

unname and unlist

- lapply returns a list. unlist() converts it into a vector (plotting function in the next slide looks for a vector).
- unname() removes the chapter names and retains only the numbers (Not needed here, Just doing to show this)

```
means <- unlist(means)
means <- unname(means)
```

- At this point, means is just a vector of numbers, which we can use to plot.
- Instead of using lapply + unlist, we can use sapply() which directly returns a vector.

note: textbook uses rbind and do.call() functions to do the same.

Plotting the mean word frequency

```
par(mfrow=c(1,2))
plot(means, type="h")
plot(scale(means), type = "h")
```

Look at the two plots - what are the differences? What is scale() doing?

scaling

- Scaling a numeric vector means subtracting the average value from each number in the vector
- So, if we plot the scaled vector, it will show us the deviations from average.

scaling

- Scaling a numeric vector means subtracting the average value from each number in the vector
- So, if we plot the scaled vector, it will show us the deviations from average.
- In this case, the positive values after scaling indicate the chapters which have a higher mean word frequency than the overall mean.
- negative values indicate chapters which have a lower mean word frequency than the overall mean.

scaling

- Scaling a numeric vector means subtracting the average value from each number in the vector
- So, if we plot the scaled vector, it will show us the deviations from average.
- In this case, the positive values after scaling indicate the chapters which have a higher mean word frequency than the overall mean.
- negative values indicate chapters which have a lower mean word frequency than the overall mean.
- if mean word frequency is used as a measure of lexical difficulty, we can say those positive valued chapters are difficult.

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.

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
```

We already have loops

- We already have loops to do something repetitively.
- What is the difference between functions and loops according to you, as of now?

We already have loops

- We already have loops to do something repetitively.
- What is the difference between functions and loops according to you, as of now?
- Answer this question in the forum for today. We will continue this discussion on Thursday.

Hapax Legomena

- 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

custom functions can be put directly into other functions such as sapply as well! (More on this in the next class!) Consider this line below (chapters.raw - is the variable from our last class):

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

-What this says is: for each item in chapters.raw, i.e., for each chapter, count the number of words whose frequency is 1.

Thursday

- R: Writing our own functions in R
- Creating R markdown reports (like my tutorial pdfs)
- Practice what we learnt so far