# LING 410X: Language as Data

## Semester: Spring '18

Instructor: Sowmya Vajjala

Iowa State University, USA

1 February 2018

# Class Outline

- Some R functions
- Analyzing word frequencies, distribution of word usage throughout the text.
- Splitting a text using regular expressions and then analyzing word usage across parts.
- target: instead of counting frequent words for entire text, we now learn to do chapter by chapter (i.e., answering questions such as: where does "Nora" appear more frequently in the play? what chapter has most references to lead character in novel X etc.)

# News: new book published on corpus analysis with R

- Title: Corpus Linguistics and Statistics with R: Introduction to Quantitative Methods in Linguistics
- Author: Guillaume Desagulier
- Published by Springer, accessible in University network for free.
- `https://link.springer.com/content/pdf/10.1007%2F978-3-319-64572-8.pdf`
- table of contents and quick summary: `https://mailman.uib.no/public/corpora/attachments/20180131/aab68a32/attachment.txt`

# old stuff: revision

- which() function: we used it to get line numbers matching a string.

```
u <- c(1,2,3,3,4,5,6,66,7,7,8)
which(u ==3) #gives: 3, 4
u ==3 #what does that give, without which?
```

# old stuff: revision

- which() function: we used it to get line numbers matching a string.

```
u <- c(1,2,3,3,4,5,6,66,7,7,8)
which(u ==3) #gives: 3, 4
u ==3 #what does that give, without which?
```

- gsub() function: used for substitution within a string.

```
gsub("[[:punct:]]", " ", dollshouse_string)
-replaces all punctuation with a space, in dollshouse_string
```

# old stuff: revision

- which() function: we used it to get line numbers matching a string.

```
u <- c(1,2,3,3,4,5,6,66,7,7,8)
which(u ==3) #gives: 3, 4
u ==3 #what does that give, without which?
```

- gsub() function: used for substitution within a string.

```
gsub("[[:punct:]]", " ", dollshouse_string)
-replaces all punctuation with a space, in dollshouse_string
```

- length() function - gives you length (Number of items) in a vector/list
- seq() function (e.g., seq(1,10) - gives a vector with items 1 ... 10
- rep() function (e.g., rep(1,10) - gives a vector with items 1 ... 1

# new: few more simple functions

- sum() function takes a numeric vector and returns its sum.

```
v4 = c(1,2,3,4)
sum(v4)
10
```

- unlist() function: converts list to a vector.

```
t <- strsplit("this is a string", " ")
t
unlist(t)
```

# new: apply() functions in R

- apply functions in R (lapply, sapply, vapply etc. ) are used for looping through of lists, vectors, and other such structures in R.
- When are they used: typically used instead of writing loops.
- one of these: lapply - you want to perform some operation on a list, and get back the output as another list.

# new: apply() functions in R

- apply functions in R (lapply, sapply, vapply etc. ) are used for looping through of lists, vectors, and other such structures in R.
- When are they used: typically used instead of writing loops.
- one of these: lapply - you want to perform some operation on a list, and get back the output as another list.

```
v1 = c("This", "is", "an", "example")
v2 = c("This", "is", "another", "sentence")
v3 = c("Third", "string", "for", "this", "example")
temp = list(strsplit(v1, " "),strsplit(v2, " "),strsplit(v3, " "))
lapply(temp, '[', 2) #gives me all 2nd items within each item in temp.
```

- We can also write our own functions where we have that square bracket. More on that next week or later.

# Analyzing word distribution throughout a text

(based on Chapter 4 of the textbook)

```
english <- scan("DollsHouse-Eng.txt", what = "character", sep = "\n")
english.start <- which (english ==  "DRAMATIS PERSONAE")
english.end <- which (english == "(The sound of a door shutting is heard from below.)")
dollshouse_text <- english[english.start:english.end]
dollshouse_string <- paste(dollshouse_text, collapse = " ")
dollshouse_string <- tolower(dollshouse_string)
dollshouse_string <- gsub("[[:punct:]]", " ", dollshouse_string)
dollshouse_string <- gsub(" +", " ", dollshouse_string)
dollshouse_words <-strsplit(dollshouse_string, "\\W")
sorted_wordfreqs_dollshouse <-sort(table(dollshouse_words), decreasing = TRUE)
```

At the end, what do you expect to see in sorted_wordfreqs_dollshouse?
(note: there are minor differences from what I used before - but nothing extremely new!)

# What do you think the following lines will show as output?

Let us go line by line:

```
length(sorted_wordfreqs_dollshouse)
sum(sorted_wordfreqs_dollshouse)
sorted_wordfreqs_dollshouse["doll"]
sorted_wordfreqs_dollshouse["he"]/sorted_wordfreqs_dollshouse["she"]
sorted_wordfreqs_dollshouse[1:10]
plot(sorted_wordfreqs_dollshouse[1:10], type="b")
```

# Raw counts to relative frequencies

The table of words and their frequencies contains raw counts of the number of times a word appeared in this text. Let us say I want to convert this into counts showing: "number of times I will see a word for every 100 words"

```
sorted_wordfreqs_dollshouse_100 <- 100 * (sorted_wordfreqs_dollshouse)/ sum(sorted_wordfreqs_dollshouse)
sorted_wordfreqs_dollshouse["the"]
sorted_wordfreqs_dollshouse_100["the"]
```

Remember "vector recycling" and vector division from your swirl lessons? We are using them here to do something useful.
What will happen if I plot with these new counts instead of raw counts?

```
plot(sorted_wordfreqs_dollshouse_100[1:10], type="b")
```

# Disperson Plots

- We saw how to know how frequently a word is used in a text.
- We also saw how to compare two words in terms of their frequencies.

# Disperson Plots

- We saw how to know how frequently a word is used in a text.
- We also saw how to compare two words in terms of their frequencies.
- Let us say I now want to create plots showing where a word occurs in the text - where is it more frequent, less frequent etc.
- Dispersion plots are useful for doing this kind of analysis.

# Disperson Plots

- We saw how to know how frequently a word is used in a text.
- We also saw how to compare two words in terms of their frequencies.
- Let us say I now want to create plots showing where a word occurs in the text - where is it more frequent, less frequent etc.
- Dispersion plots are useful for doing this kind of analysis.
- X-axis: shows progression of the text. Let us say progression is shown from first word to last word of the entire text.
- Y-axis: In this case, we are only noting presence/absence of a word throughout the text. So, that is our Y-axis

# How to create a dispersion plot?

1. Step 0: create a sequence vector for the number of words in the text (what is a sequence vector?)

# How to create a dispersion plot?

1. Step 0: create a sequence vector for the number of words in the text (what is a sequence vector?)
2. Step 1: get all locations of occurrence of a word (e.g., "is")
3. Step 2: Have a output vector that has the same dimension as sequence vector from Step 0.
4. Step 3: Replace the locations where we see the word (is) in output with 1.

```
dollshouse_words_vector <- unlist(dollshouse_words)
progress <- seq(1:length(dollshouse_words_vector)))
nora <- which(dollshouse_words_vector == "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')
```

You can do this with any word you want in the text.

# Let us do a little bit more

## Chapter 4.2

- Let us take a book with Chapters, do start looking at how many times a word appears in each chapter, and plot again.
- I am going to use Moby Dick that the author also uses in his examples.

# Let us do a little bit more

## Chapter 4.2

- Let us take a book with Chapters, do start looking at how many times a word appears in each chapter, and plot again.
- I am going to use Moby Dick that the author also uses in his examples.
- What do you think the following lines are doing?

```
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]
```

# Let us do a little bit more

## Chapter 4.2

- Let us take a book with Chapters, do start looking at how many times a word appears in each chapter, and plot again.
- I am going to use Moby Dick that the author also uses in his examples.
- What do you think the following lines are doing?

```
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.chap.positions is a vector that will hold line numbers of the lines matching the regular expression. So, the last line will print you the actual lines matching those line numbers.

This will give us info about where the chapters are starting and ending. But, how do we know what is the end of last chapter?

This will give us info about where the chapters are starting and ending. But, how do we know what is the end of last chapter? One simple solution is to just look at the length of the vector. The last line should indicate the end of the last chapter.

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

# For loop and If condition

- To go through chapter by chapter and count word frequencies instead of counting throughout the text, we need to change our approach a little bit.
- We should use a for loop to tell R that you want to do the same analysis for each chapter instead of full text at once.

# For loop and If condition

- To go through chapter by chapter and count word frequencies instead of counting throughout the text, we need to change our approach a little bit.
- We should use a for loop to tell R that you want to do the same analysis for each chapter instead of full text at once.
- "If" condition is similar to regular English. It just says: if condition X is met, do this.

# Looping through the chapters of Moby Dick

Let us slowly go through line by line.

```
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
}
```

At the end of this, chapters.raw should have the words and their frequencies for each and every individual chapter!!

# What did we do so far?

- We started with the aim of looking at the distribution of words chapter by chapter in Moby Dick.
- We first split the text into chapters by using a regular expression.
- Then, we wrote a "loop" that goes through each chapter, does the same set of actions:
  - get chapter start and end positions
  - get chapter text from full book based on these line positions
  - lowercase text, remove punctuations
  - split the text into words
  - count frequencies of words for each chapter

- At the end, we had chapters.raw, a big variable which contained all we wanted.
- So, how do we get what we want from this??
- Remember the discussion about vectors and lists from tuesday?

# Accessing List items

- chapters.raw is a **list** where we stored data in the form: $chapters.raw[[title]] <- chapter.freqs$
- So, how do we access the first chapter and its word frequency list here? - chapters.raw [[1]]
- chapters.raw [[1]]["whale"]- What will this then give me?

# Accessing List items

- chapters.raw is a **list** where we stored data in the form: $chapters.raw[[title]] <- chapter.freqs$
- So, how do we access the first chapter and its word frequency list here? - chapters.raw [[1]]
- chapters.raw [[1]]["whale"]- What will this then give me?
- chapters.raw [[1]][1]- what will this give me?

# Accessing List items

- chapters.raw is a **list** where we stored data in the form: $chapters.raw[[title]] <- chapter.freqs$
- So, how do we access the first chapter and its word frequency list here? - chapters.raw [[1]]
- chapters.raw [[1]]["whale"]- What will this then give me?
- chapters.raw [[1]][1]- what will this give me?
- chapters.raw [[1]][[1]]- what will this give me?

# Accessing List items

- chapters.raw is a **list** where we stored data in the form: $chapters.raw[[title]] <- chapter.freqs$
- So, how do we access the first chapter and its word frequency list here? - chapters.raw [[1]]
- chapters.raw [[1]]["whale"]- What will this then give me?
- chapters.raw [[1]][1]- what will this give me?
- chapters.raw [[1]][[1]]- what will this give me?

# What next?

- Okay, we have some way of getting chapter wise frequencies of words now.
- How do we get them all at once, instead of writing that line one by one for each chapter like this:
  chapters.raw [[1]]["whale"]
  chapters.raw [[2]]["whale"]
  .... ....

# lapply

- We want all occurrences of some word ("whale" in my example) from all chapters using chapters.raw.

# lapply

- We want all occurrences of some word ("whale" in my example) from all chapters using chapters.raw.
- using lapply for that:
  $whaledetails <- lapply(chapters.raw, "[", "whale")$
- What this is saying is:
  - Go through the elements of chapters.raw one by one (e.g., chapters.raw [[1]], chapters.raw [[2]]etc)

# lapply

- We want all occurrences of some word ("whale" in my example) from all chapters using chapters.raw.
- using lapply for that:
  $whaledetails <- lapply(chapters.raw, "[", "whale")$
- What this is saying is:
  - Go through the elements of chapters.raw one by one (e.g., chapters.raw [[1]], chapters.raw [[2]]etc)
  - "[" says: for each of these element, do bracketed subsetting: (e.g., chapters.raw [[1]][1]or chapters.raw [[1]]["whale"])
  - "whale" is an additional argument that says: look for the word "whale" in each of these bracketed subset of each element of chapters.raw
    (i.e., look only for these: chapters.raw [[1]]["whale"], chapters.raw [[2]]["whale"]etc.)

```
$`CHAPTER 48. The First Lowering.`
whale
    9

$`CHAPTER 49. The Hyena.`
whale
    7

$`CHAPTER 50. Ahab's Boat and Crew. Fedallah.`
whale
    3

$`CHAPTER 51. The Spirit-Spout.`
whale
    2

$`CHAPTER 52. The Albatross.`
whale
    2

$`CHAPTER 53. The Gam.`
whale
    5

$`CHAPTER 54. The Town-Ho's Story.`
whale
   23
```

# what next?

What is happening in this code snippet below?:

```
whale_counts = c()
for (i in 1:length(whale))
{
  whale_counts[i] <- whale[[i]]
}
whale_counts
plot(whale_freqs_per_chap, type="h")
```

# what next?

What is happening in this code snippet below?:

```
whale_counts = c()
for (i in 1:length(whale))
{
  whale_counts[i] <- whale[[i]]
}
whale_counts
plot(whale_freqs_per_chap, type="h")
```

Note: There are several ways of doing this in R. The textbook describes another way of achieving this which requires you to know more about using functions like rbind(), do.call() etc.

# Rest of the class

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?

# Next Week

- Measures of lexical variety, and Hapax legomena (Chapters 6 and 7)
- Searching for key words in context (Chapter 8)
- ToDo: Read Chapter 6 and 7 by Tuesday.
- Note: I am leaving Chapter 5 out of my classwork.