Chapter4

Sowmya

1/2/2018

Purpose of this document is to show how to take a piece of text, do some pre-processing and start analysing how words are distributed across the text. It can be considered as a continuation from where we ended on Week 2. I want you to read Chapter 4 of the textbook, and also take a look at my class slides for today's class, along with this worksheet.

Important instructions: 1) Do not copy paste from this pdf file. Type what you want to run in R console. 2) Set your working directory before proceeding, or 3) Use full file paths where I give file names.

Let us start

Initially I am going to read DollsHouse-Eng.txt and do some preprocessing on that.

```
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)
```

What are these lines above doing?? Try to understand each line. What is dollshouse_string expected to have in the end here? Quick summary of the above code: line 1: reading the file into R line 2-4: getting the actual content of the file, discarding meta data line 5: converting collection of lines into one big string. line 6–8: lowercasing, removing punctuations, removing extra spaces (Why did the extra spaces come up??)

```
dollshouse_words <-strsplit(dollshouse_string, "\\W")
sorted_wordfreqs_dollshouse <-sort(table(dollshouse_words), decreasing = TRUE)
```

What should I expect to see in sorted_wordfreqs_dollshouse after all these?

Number of unique words and total number of words in the text can be obtained by the following two lines once we create a sorted table:

```
length(sorted_wordfreqs_dollshouse)
```

```
## [1] 2409
```

```
sum(sorted_wordfreqs_dollshouse)
```

```
## [1] 27256
```

To access individual word frequencies, and compare them with each other can be done by something like this:

```
sorted_wordfreqs_dollshouse["doll"]
```

```
## doll
##    8
```

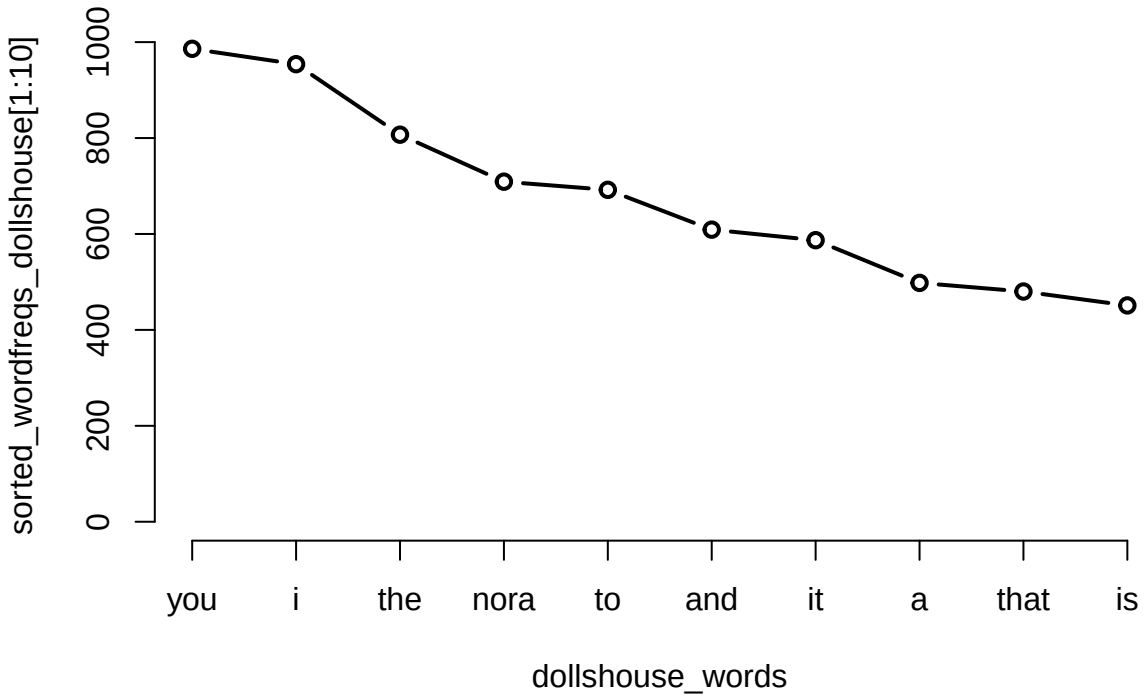
```
sorted_wordfreqs_dollshouse["he"]/sorted_wordfreqs_dollshouse["she"]
```

```
##       he
## 2.378049
```

First line just gets me back number of times "doll" appeared in the text. Second line gives me ratio between "he" vs "she" in this text.

Plotting the 10 most frequent wods and their frequencies is done like below, as you already saw in Week 2.

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



Converting raw counts to relative frequencies: Let us say I want to convert all these counts into numbers indicating: "number of times I will see a word for every 100 words" instead of general counts (you will need this sometimes, in corpus analysis work)

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

```
## the
## 807
```

```
sorted_wordfreqs_dollshouse_100["the"]
```

```
##      the
## 2.960816
```

```
sorted_wordfreqs_dollshouse["doll"]
```

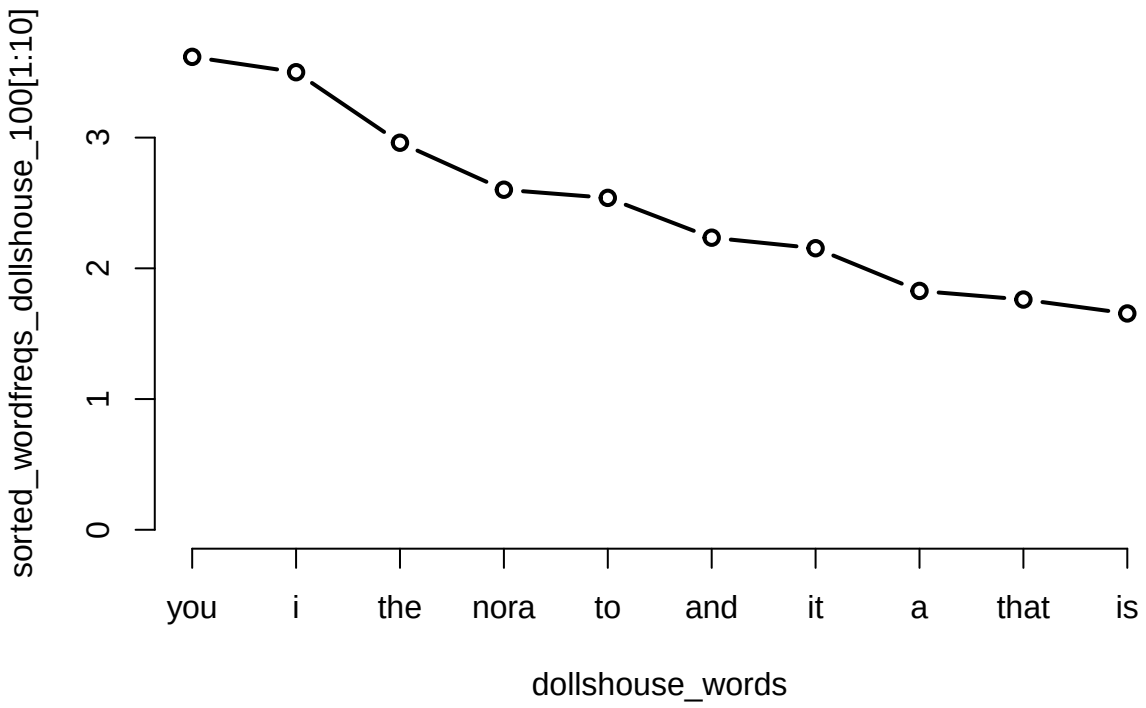
```
## doll
##    8
```

```
sorted_wordfreqs_dollshouse_100["doll"]
```

```
##       doll
## 0.02935134
```

Now, let us see how the plot for 10 most frequent words looks like after this relative counts:

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



Does the shape of the plot change? Why? Why not?

Dispersion Plots

Let us move now to the task of plotting usage of a given word across the whole document in a text.

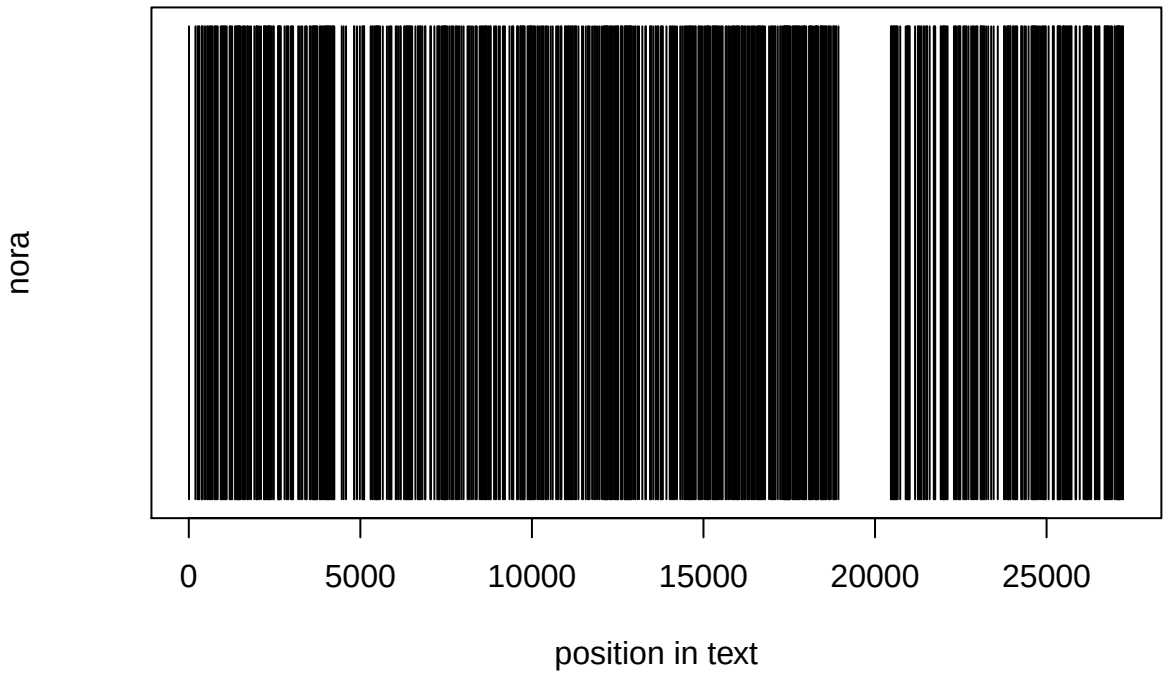
```
dollshouse_words_vector <- unlist(dollshouse_words)
```

What this line is doing is to convert a list to a vector. This is because I want to in the next step create another vector which matches the words in the text, with numbers indicating the position of that word in the whole text.

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

```
## [1] 709
```

```
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')
```

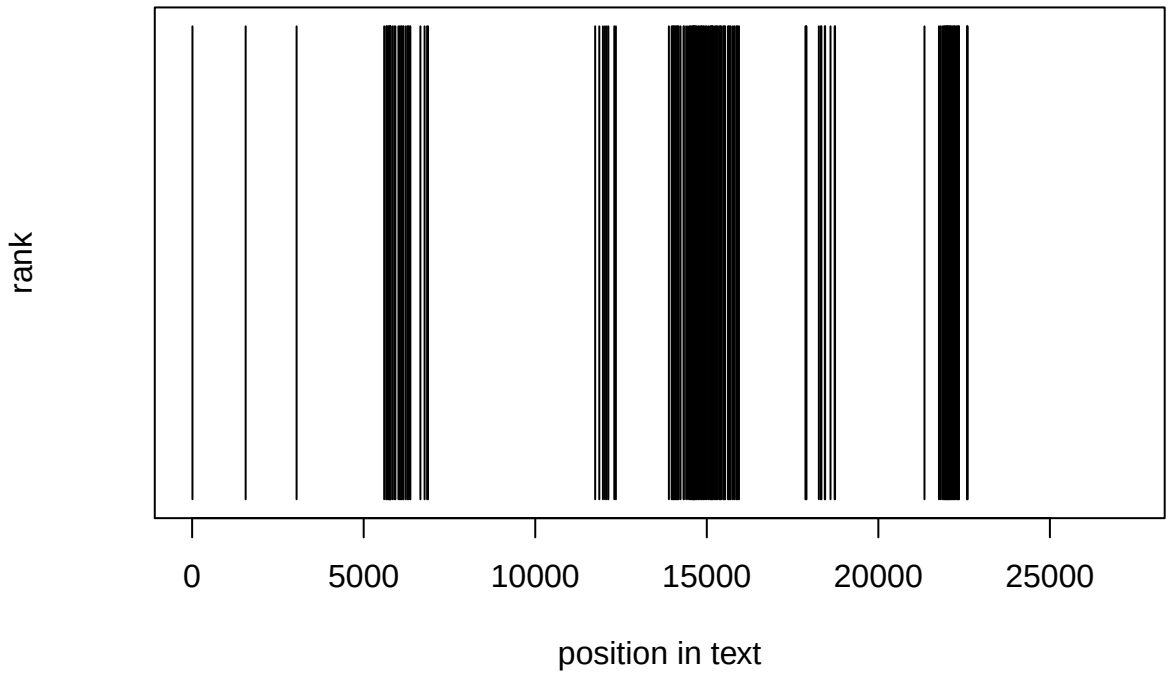



```
#nora_progression
#use the above line if you want to see what that variable nora looks like.
```

nora_progression vector will have a 1 where the word is nora, and NA where it is not, and it is as long as the length of the dolls house play!

Let us do the same analysis for another word, "rank"

```
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')
```



ylim is used to limit my y-axis scale to (0,1) because my nora_progression vector elements only take a value of either 1 or NA. yaxt=n is used to indicate that I do not want to see those numbers (0 to 1) marked on the axis, because I am not looking for "numeric" comparisons anyway.

We don't have to copy-paste repetitively each time we want for a new word. We can write our own R function for that - more on this in the coming weeks!

Using Grep to identify chapter breaks

From here, I will use another book: Moby Dick, following the textbook example. The task here is to first split the novel by chapters, and analyze word usage across chapters (instead of word by word, like in dispersion plot example above). Chapters seem to start with "CHAPTER" followed by a digit sequence. So, we can use that to get the line numbers for where a chapter begins, using regular expressions.

```
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]
```

Okay, I do know where all chapters are beginning and ending, but I do not really know where this last chapter ends. But, I perhaps can get it by just getting the last line of the novel itself. Here is how to find that out:

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

```
##   [1]     1   185   301   790   925   989  1062  1141  1222  1524  1654
##  [12]  1712  1785  1931  1996  2099  2572  2766  2887  2997  3075  3181
##  [23]  3323  3357  3506  3532  3635  3775  3893  3993  4018  4084  4532
```

```
##  [34]  4619  4805  5023  5273  5315  5347  5371  5527  5851  6170  6202
##  [45]  6381  6681  6771  6856  7201  7274  7360  7490  7550  7689  8379
##  [56]  8543  8656  8742  8828  8911  9032  9201  9249  9293  9555  9638
##  [67]  9692  9754  9854  9894  9971 10175 10316 10502 10639 10742 10816
##  [78] 10876 11016 11097 11174 11541 11638 11706 11778 11947 12103 12514
##  [89] 12620 12745 12843 13066 13148 13287 13398 13440 13592 13614 13701
## [100] 13900 14131 14279 14416 14495 14620 14755 14835 14928 15066 15148
## [111] 15339 15377 15462 15571 15631 15710 15756 15798 15873 16095 16113
## [122] 16164 16169 16274 16382 16484 16601 16671 16790 16839 16984 17024
## [133] 17160 17473 17761 18169
```

```
#length(moby.chap.positions) gives you the number of chapters.
```

So, I want to now go through chapter by chapter, and get the sorted word frequency table like we did before (i.e., instead of a single large table, we will have N such tables, where N is the number of chapters). I am doing this using a for loop here. I am going to store all that information in a variable chapters.raw in the below code - since initially, it does not have anything, I just declare a empty list.

How do I get the end line for a given chapter? - it is the line before the next chapter's start!

```
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
}

#10 most frequent words in Chapter 1
sort(chapters.raw[[1]], decreasing = TRUE)[1:10]
```

```
## chapter.words
##  the   of  and    a   to   in    i   is   it that
##  124   81   73   69   53   48   43   34   33   31
```

```
#10 most frequent words in Chapter 2
sort(chapters.raw[[2]], decreasing = TRUE)[1:10]
```

```
## chapter.words
##  the    a  and   of   in    i   to   it that  for
##   98   52   49   47   28   27   26   23   19   15
```

```
#Frequency of "whale" in Chapter 1
chapters.raw[[1]]["whale"]
```

```
## whale
##     3
```

```
#Frequency of "whale" in Chapter 2
chapters.raw[[2]]["whale"]
```

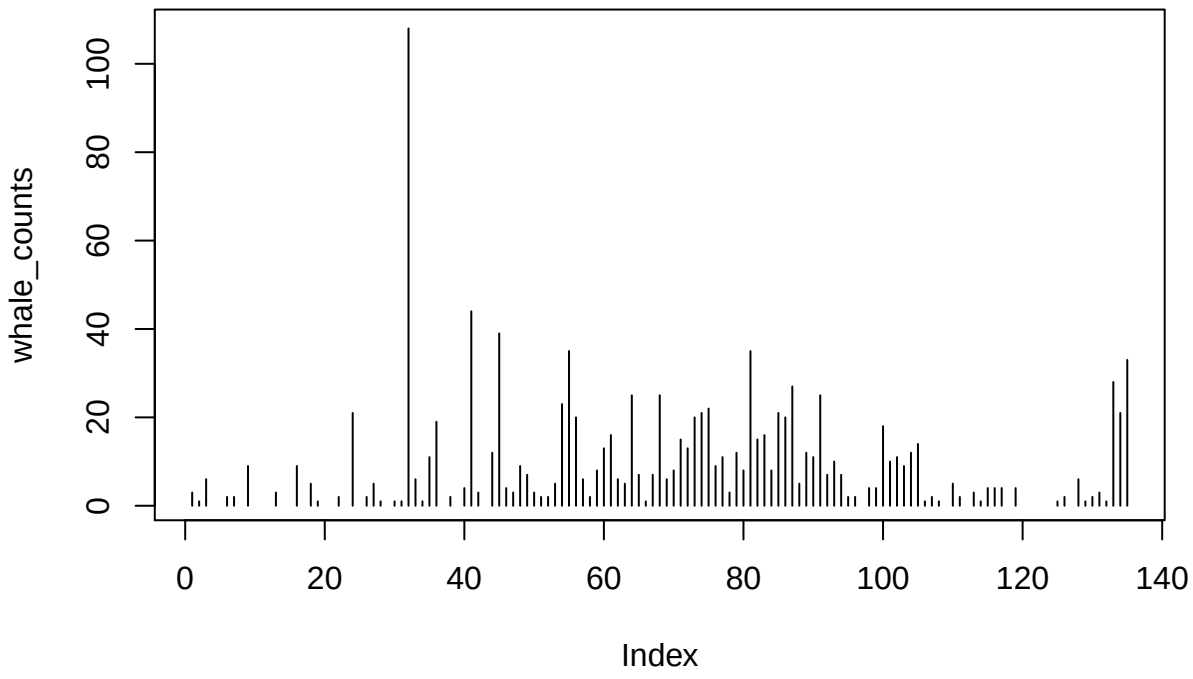
```
## whale
##     1
```

Getting information about specific words across chapters, formatting them and plotting: 1. We will use lapply for this - we learnt about this in today's class. The first line gives us the chapter names and frequency of whales in that chapter as a list. 2. Once i have that, I am writing a for-loop to create a vector of only the chapter wise counts for the word "whale" 3. Then, I use the counts

```
whale <- lapply(chapters.raw, "[", "whale")
whale_counts = c()
for (i in 1:length(whale))
{
  whale_counts[i] <- whale[[i]]
}
whale_counts
```

```
##   [1]   3   1   6  NA  NA   2   2  NA   9  NA  NA  NA   3  NA  NA   9  NA
##  [18]   5   1  NA  NA   2  NA  21  NA   2   5   1  NA   1   1 108   6   1
##  [35]  11  19  NA   2  NA   4  44   3  NA  12  39   4   3   9   7   3   2
##  [52]   2   5  23  35  20   6   2   8  13  16   6   5  25   7   1   7  25
##  [69]   6   8  15  13  20  21  22   9  11   3  12   8  35  15  16   8  21
##  [86]  20  27   5  12  11  25   7  10   7   2   2  NA   4   4  18  10  11
## [103]   9  12  14   1   2   1  NA   5   2  NA   3   1   4   4   4  NA   4
## [120]  NA  NA  NA  NA  NA   1   2  NA   6   1   2   3   1  28  21  33
```

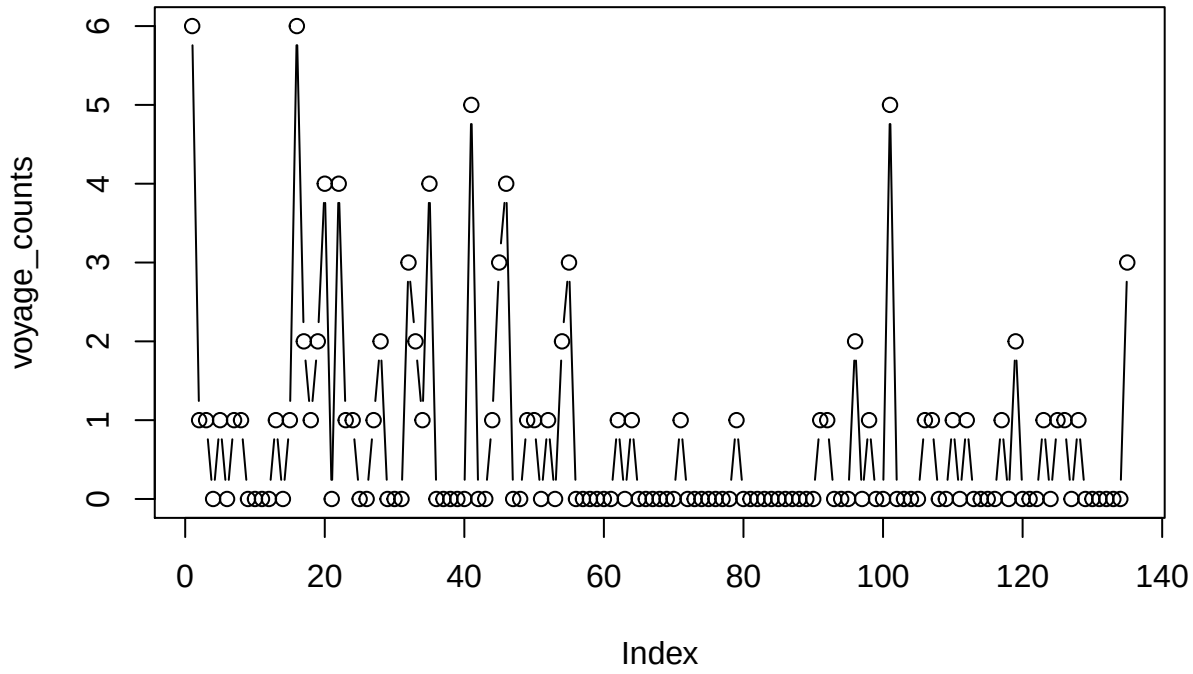
```
plot(whale_counts, type="h")
```



Let me do this for another word:

```
voyage <- lapply(chapters.raw, "[", "voyage")
voyage_counts = c()
for (i in 1:length(voyage))
{
  voyage_counts[i] <- voyage[[i]]
}
```

```r
voyage_counts[is.na(voyage_counts)] <- 0 #Replacing NAs with 0.
plot(voyage_counts, type="b")
```



Note: There are other ways of doing this, and R has several advanced functions that will let you do stuff without writing too many loops. You will get there as you progress further!