CS3907/CS6444 BIG DATA AND ANALYTICS

Class Project #3
Due December 13, 2019 COB

Text Analytics in R

1. Data Set: Rikki-Tikki-Tavi.txt
The problem is to process a large document and analyze it.

You can start by following the slides in Lecture 8.

You should do at least the following:
a. Try the functions in lecture 8.
- What happens?
- Do they yield anything understandable about the document.

b. Prior to removing the punctuation, find the 10 longest words and 10 longest sentences in the document. Print a table of this data as well as showing these items.

c. After filtering out stop words and other elements as suggested in Lecture 8, work through the examples given in Lecture 8 to display the dendrogram and the WordCloud.

For the following you will need to write R functions to help you compute the results. Use the packages textreuse, wordnet, zipfR

d. Use WordNet to mark the parts of speech for the 10 longest sentences found in part b for nouns and verbs having a length of 5 or greater.

e. Analyze word frequency using functions from package zipfR.

f. Generate bigrams and trigrams for all words whose length is greater than 6 characters in the 10 longest sentences

Now, there are several other packages that you can use. By now, you know to load packages, read their PDF files, and apply their methods to do analysis.

Read text_analysis_in_R.pdf.

g. Process the text from the document using corpusTools, stringi, corpustools, quanteda, and tidytext. Describe the methods you use, the results, you get, and what you understand about the theme of the book.

By now, you should see that Data Science is an empirical science. So, these packages

functions from each package and apply them to the document.

2. Deliverables: You will deliver your results by putting a zipfile in your group's Blackboard file, with the following naming convention: Group-N-Project-3.zip, where N is your group number. Your deliverable should encompass the following items:

A listing of all R functions that you have written
A document giving your results which should include your assessment of applying the different techniques to the data provided.

Remember to save your workspace! In your Group area would be a good place so all members can get to it.

Include in your Word document the results required
(use a CTRL-ALT-PrintScreen) to grab the screen
You may use Irfanview 4.40, irfanview@gmx.net. Paste in the screen image, and copy the image as JPEG to drop into your Word document.

3. Due Date: December 13, 2019 COB (Saturday before the Final Exam)

4. Project #3 Value: 25 points
a. **Document R functions: 2 points**

b. **Presentation and discussion of results from the experiments that you run using the different functions from Lecture 8: parts (a) through (f)** - 2 points each. **Include plots where applicable**. 16 points.

c. Presentation and discussion of part (g). 8 points.

d. **Analysis of what this project helped you learn about text analytics**, e.g., the exploration of data which is what you have been doing: 3 points

a. Preparing data and trying functions:

Setting the document path:

setwd("/Users/aarvithadeshwar/big data/final project")

Creating a corpus, inspecting the document's characteristics:

rtt<-VCorpus(DirSource(".", ignore.case=TRUE, mode="text"))

rtt

inspect(rtt)

```
> rtt
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 1
> inspect(rtt)
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 1

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 29900
```

str(rtt)

```
> str(rtt)
List of 1
 $ analysethis.txt:List of 2
  ..$ content: chr [1:611] "RIKKI-TIKKI-TAVI" "" "This is the story of the great war that Rikki-tik
i-tavi fought single-handed, through the bath-rooms of the big " "bungalow in Segowlee cantonment.
arzee, the tailor-bird, helped him, and Chuchundra, the musk-rat, who never " ...
  ..$ meta   :List of 7
  .. ..$ author       : chr(0)
  .. ..$ datetimestamp: POSIXlt[1:1], format: "2019-12-12 23:46:50"
  .. ..$ description  : chr(0)
  .. ..$ heading      : chr(0)
  .. ..$ id           : chr "analysethis.txt"
  .. ..$ language     : chr "en"
  .. ..$ origin       : chr(0)
  .. ..- attr(*, "class")= chr "TextDocumentMeta"
  ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
 - attr(*, "class")= chr [1:2] "VCorpus" "Corpus"
```

Creating the document term matrix:

rttdtm <- DocumentTermMatrix(rtt)

rttdtm

```
> rttdtm <- DocumentTermMatrix(rtt)
> rttdtm
<<DocumentTermMatrix (documents: 1, terms: 1602)>>
Non-/sparse entries: 1602/0
Sparsity           : 0%
Maximal term length: 31
Weighting          : term frequency (tf)
```

Creating the term document matrix:

rtttdm<- TermDocumentMatrix(rtt)

rtttdm

```
> rtttdm
<<TermDocumentMatrix (terms: 1602, documents: 1)>>
Non-/sparse entries: 1602/0
Sparsity           : 0%
Maximal term length: 31
Weighting          : term frequency (tf)
```

Describing the first 20 terms and their frequencies in the TD matrix:

inspect(rtttdm[1:20,1])

```
<<TermDocumentMatrix (terms: 20, documents: 1)>>
Non-/sparse entries: 20/0
Sparsity           : 0%
Maximal term length: 31
Weighting          : term frequency (tf)
Sample             :
                                Docs
Terms                            analysethis.txt
  _ding-dong-tock!_'                           1
  _hear_,                                      1
  _our_                                        1
  _rikk-tck-tck!_'                             1
  _rikki-tikki-tck-tck!_                       1
  _scratch-scratch_                            1
  '_rikk-tikk-tikki-tikki-tchk!_'              2
  'all                                         3
  'and                                         5
  'but                                         2
```

Converting document characters to lowercase:

rttlow<- tm_map(rtt, content_transformer(tolower))
rttlow
str(rttlow)

```
> rttlow
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 1
> str(rttlow)
List of 1
 $ analysethis.txt:List of 2
  ..$ content: chr [1:611] "rikki-tikki-tavi" "" "this is the story of the great war that rikki-tikk
i-tavi fought single-handed, through the bath-rooms of the big " "bungalow in segowlee cantonment. c
arzee, the tailor-bird, helped him, and chuchundra, the musk-rat, who never " ...
  ..$ meta   :List of 7
  .. ..$ author       : chr(0)
  .. ..$ datetimestamp: POSIXlt[1:1], format: "2019-12-12 23:46:50"
  .. ..$ description  : chr(0)
  .. ..$ heading      : chr(0)
  .. ..$ id           : chr "analysethis.txt"
  .. ..$ language     : chr "en"
  .. ..$ origin       : chr(0)
  .. ..- attr(*, "class")= chr "TextDocumentMeta"
  ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
 - attr(*, "class")= chr [1:2] "VCorpus" "Corpus"
```

Removing any numbers and punctuation from the document:

remNumPunc <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
rttclean<- tm_map(rttlow, content_transformer(remNumPunc))
rttclean
str(rttclean)

```
> str(rttclean)
List of 1
 $ analysethis.txt:List of 2
  ..$ content: chr [1:611] "rikkitikkitavi" "" "this is the story of the great war that rikkitikki
vi fought singlehanded through the bathrooms of the big " "bungalow in segowlee cantonment darzee
e tailorbird helped him and chuchundra the muskrat who never " ...
  ..$ meta   :List of 7
  .. ..$ author       : chr(0)
  .. ..$ datetimestamp: POSIXlt[1:1], format: "2019-12-12 23:46:50"
  .. ..$ description  : chr(0)
  .. ..$ heading      : chr(0)
  .. ..$ id           : chr "analysethis.txt"
  .. ..$ language     : chr "en"
  .. ..$ origin       : chr(0)
  .. ..- attr(*, "class")= chr "TextDocumentMeta"
  ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
 - attr(*, "class")= chr [1:2] "VCorpus" "Corpus"
```

Listing stop words:

rttstopwords<- c(stopwords('english'))

rttstopwords

```
> rttstopwords
  [1] "i"          "me"         "my"         "myself"     "we"         "our"
  [7] "ours"       "ourselves"  "you"        "your"       "yours"      "yourself"
 [13] "yourselves" "he"         "him"        "his"        "himself"    "she"
 [19] "her"        "hers"       "herself"    "it"         "its"        "itself"
 [25] "they"       "them"       "their"      "theirs"     "themselves" "what"
 [31] "which"      "who"        "whom"       "this"       "that"       "these"
 [37] "those"      "am"         "is"         "are"        "was"        "were"
 [43] "be"         "been"       "being"      "have"       "has"        "had"
 [49] "having"     "do"         "does"       "did"        "doing"      "would"
 [55] "should"     "could"      "ought"      "i'm"        "you're"     "he's"
 [61] "she's"      "it's"       "we're"      "they're"    "i've"       "you've"
 [67] "we've"      "they've"    "i'd"        "you'd"      "he'd"       "she'd"
 [73] "we'd"       "they'd"     "i'll"       "you'll"     "he'll"      "she'll"
 [79] "we'll"      "they'll"    "isn't"      "aren't"     "wasn't"     "weren't"
 [85] "hasn't"     "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
 [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"      "cannot"
 [97] "couldn't"   "mustn't"    "let's"      "that's"     "who's"      "what's"
[103] "here's"     "there's"    "when's"     "where's"    "why's"      "how's"
[109] "a"          "an"         "the"        "and"        "but"        "if"
[115] "or"         "because"    "as"         "until"      "while"      "of"
[121] "at"         "by"         "for"        "with"       "about"      "against"
```

Removing stop words:

rttstop<-tm_map(rttclean, removeWords, rttstopwords)

inspect(rttstop)

The character count reduces by approximately 9000 characters.

```
> rttstop<-tm_map(rttclean, removeWords, rttstopwords)
> inspect(rttstop)
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 1

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 20257
```

Constructing a term frequency vector from a text:
termFreq()

test<-rttpun[[1]]
termF<-termFreq(test)
termF

```
> test<-rttpun[[1]]
> termF<-termFreq(test)
> termF
```

	1		4
	act		advantage
	1		1
	advice		afraid
	1		5
	after		afternoon
	4		1
	afterward		again
	1		11
	against		ago
	4		1
	ahh		air
	1		1
	alice		all
	1		38
	almost		along
	3		3
	always		among
	3		1
	amused		and
	1		273
	andoh		angle
	1		1

Finding frequent terms in a document-term matrix:
findFreqTerms()

docterm<-TermDocumentMatrix(rtt_lower, control = list(wordlengths=c(1,Inf)))
freq<-findFreqTerms(docterm, lowfreq=3)
freq

```
> freq<-findFreqTerms(docterm, lowfreq=3)
> freq
  [1] "about"          "across"         "afraid"
  [4] "after"          "again"          "against"
  [7] "all"            "almost"         "along"
 [10] "always"         "and"            "angry"
 [13] "any"            "anything"       "are"
 [16] "away"           "babies"         "back"
 [19] "bath"           "bathroom"       "because"
 [22] "bed"            "been"           "before"
 [25] "began"          "behind"         "better"
 [28] "between"        "big"            "bird"
 [31] "birds"          "bit"            "bite"
 [34] "body"           "bottom"         "boy"
 [37] "broken"         "bungalow"       "but"
 [40] "came"           "can"            "care"
 [43] "catch"          "caught"         "chua"
 [46] "chuchundra"     "clear"          "climbed"
 [49] "close"          "cobra"          "cobras"
 [52] "coiled"         "cold"           "come"
 [55] "comes"          "could"          "cried"
 [58] "dark"           "darzee"         "darzees"
 [61] "day"            "dead"           "death"
```

Removing words which occur infrequently:
removeSparseTerms()

rttdm<-TermDocumentMatrix(stoprtt, control = list(wordlengths=c(1,Inf)))
spar<-removeSparseTerms(rttdm, sparse = 0.75)
inspect(spar)

```
c(1,Inf)))
> spar<-removeSparseTerms(rttdm, sparse = 0.75)
> inspect(spar)
<<TermDocumentMatrix (terms: 1041, documents: 1)>>
Non-/sparse entries: 1041/0
Sparsity           : 0%
Maximal term length: 22
Weighting          : term frequency (tf)
Sample             :
            Docs
Terms        Rikki-Tikki-Tavi.txt
  big                          21
  came                         19
  darzee                       23
  head                         25
  little                       22
  nag                          39
  nagaina                      36
  rikkitikki                   86
  said                         49
  teddys                       27
```

Converting text to tokens:

tokens()

trial1<-rttpun[[1]]$content
tok<-tokens(trial1)
tok
tok[1:20]

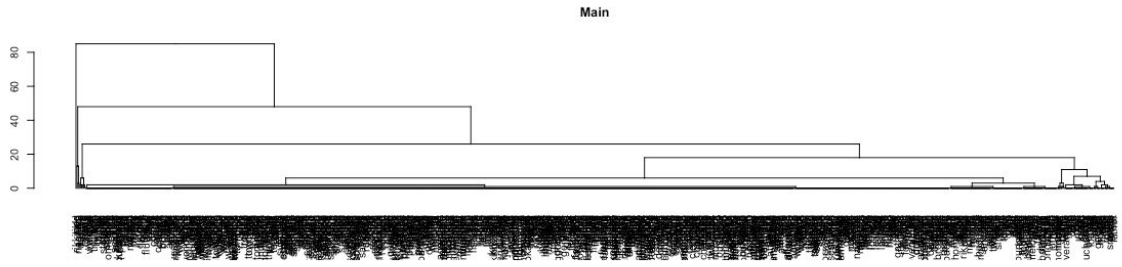
```
> trial1<-rttpun[[1]]$content
> tok<-tokens(trial1)
> tok
```

```
tokens from 20 documents.
text1 :
[1] "rikkitikkitavi"

text2 :
character(0)

text3 :
 [1] "this"           "is"             "the"
 [4] "story"          "of"             "the"
 [7] "great"          "war"            "that"
[10] "rikkitikkitavi" "fought"

text4 :
 [1] "singlehanded" "through"      "the"
 [4] "bathrooms"    "of"           "the"
 [7] "big"          "bungalow"     "in"
[10] "segowlee"

text5 :
[1] "cantonment" "darzee"     "the"        "tailorbird"
[5] "helped"     "him"        "and"        "chuchundra"
```

b. Dendrograms and Word Cloud

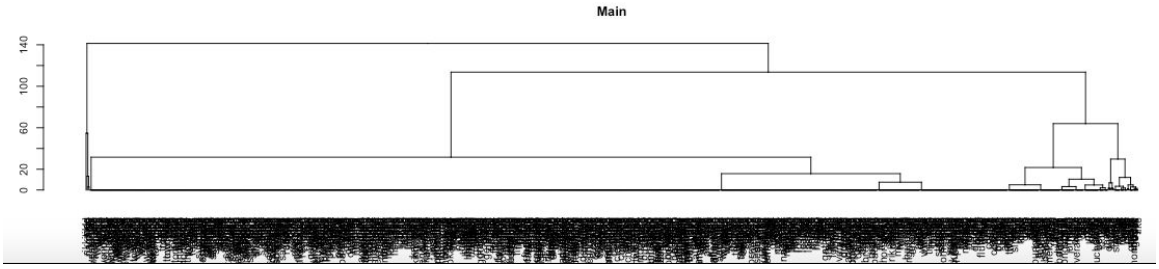
i. Dendrogram using complete method

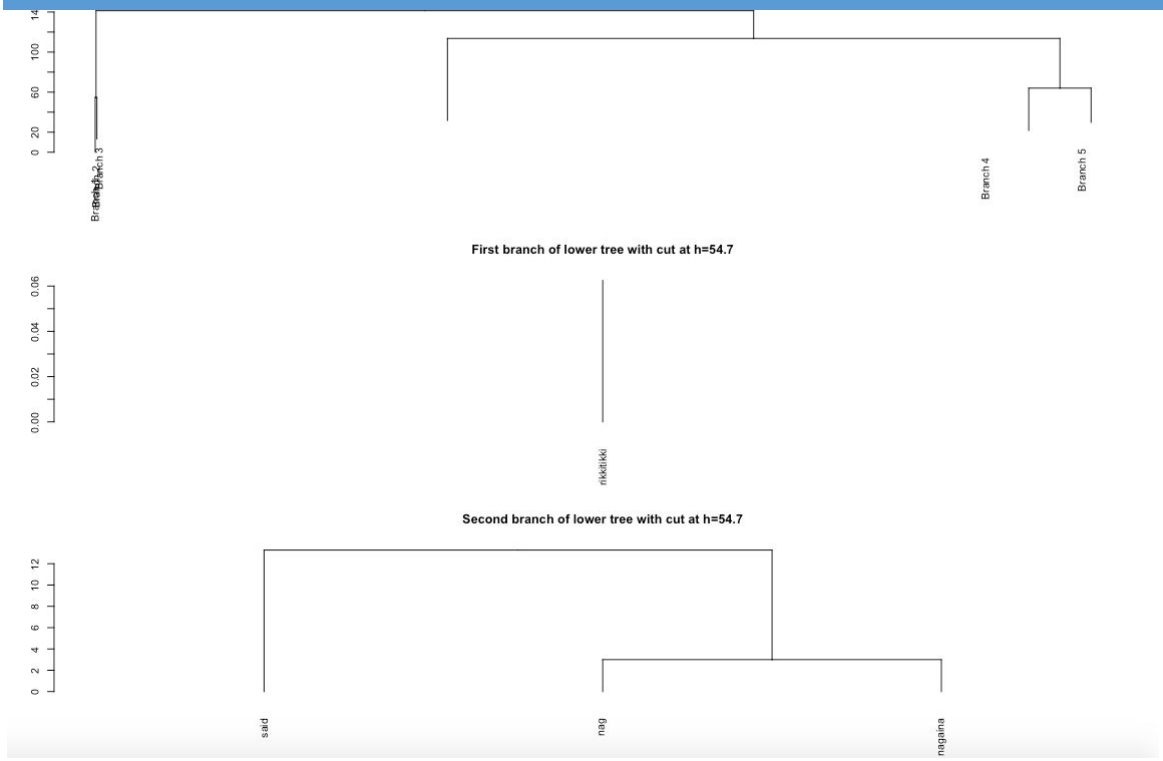
```
hc<-hclust(dist(rtttdm2))
hcd<-as.dendrogram(hc)
plot(hcd, main="Main")
plot(cut(hcd, h=40)$upper,
    main="Upper tree of cut at h=40")
plot(cut(hcd, h=40)$lower[[2]],
    main="Second branch of lower tree with cut at h=30")
```



ii. Dendrogram using ward's method:

```
hc<-hclust(dist(rtttdm2, method = "euclidean"), method = "ward.D2")
hcd<-as.dendrogram(hc)
plot(hcd, main="Main")
plot(cut(hcd, h=54.7)$upper,
    main="Upper tree of cut at h=54.7")
plot(cut(hcd, h=54.7)$lower[[2]],
    main="Second branch of lower tree with cut at h=54.7")
```




0 20 60 100 14
Branch 4
Branch 5
First branch of lower tree with cut at h=54.7
0.00 0.02 0.04 0.06
nkkitkki
Second branch of lower tree with cut at h=54.7
0 2 4 6 8 10 12
said
nag
nagaina

iii. Word cloud generation:

Finding the word frequency, we see that rikkitikki, said, nag and nagaina are the four most frequently occurring words. This is also seen in the dendrograms found above.

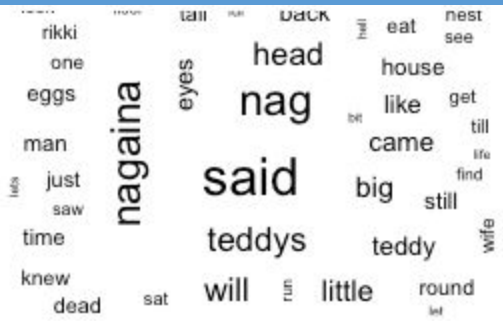
```
rtttm<-as.matrix(rtttdm2)
word.freq<-sort(rowSums(rtttm), decreasing=T)
word.freq
```

```
> word.freq<-sort(rowSums(rtttm), decreasing=T)
> word.freq
```

rikkitikki	said	nag	nagaina
86	49	39	36
teddys	will	head	darzee
27	26	25	23
little	big	came	teddy
22	21	19	18
eyes	like	back	mongoose
17	17	16	16
mother	garden	house	never
16	15	15	15
still	eggs	man	time
15	14	14	14
dead	just	round	tail
13	13	13	13
eat	father	grass	knew
12	12	12	12
snakes	went	heard	one

Now, plotting clouds with frequency>3, freq>10:

```
rttpal<- brewer.pal(9, "BuGn")
rttpal<-rttpal[-(1:4)]
wordcloud(words = names(word.freq), freq = word.freq, min.freq=10, random.order=F,
        colours=rttpal)
```



c. Finding the 10 longest words and sentences

10 longest words:

```
dtm1<-TermDocumentMatrix(rtt_lower)
m2<-as.matrix(dtm1)
v2<-sort(rowSums(m2), decreasing = TRUE)
d2<- data.frame(word = names(v2), freq=v2)
dfm<-as.data.frame(d3[, 1], drop = False, colnames = c('word'))
dfm$length<-nchar(as.character(dfm$word))
new_df<- dfm %>%
arrange(desc(length)) %>%
select(word,length)
```

```
> m2<-as.matrix(dtm1)
> v2 <- sort(rowSums(m2),decreasing=TRUE)
> d2 <- data.frame(word = names(v2),freq=v2)
> View(d2)
> dfm<-as.data.frame(d2[, 1], drop = False)
> View(dfm)
> View(d2)
> dfm<-as.data.frame( d2[, 1], drop = False, )
> dfm<-as.data.frame( d2[, 1], drop = False)
> View(dfm)
> dfm<-as.data.frame( name = d2[, 1], drop = False)
Error in as.data.frame(name = d2[, 1], drop = False) :
  argument "x" is missing, with no default
> dfm<-as.data.frame(d2[, 1], drop = False, colnames("Word
s"))
> View(dfm)
> dfm<-as.data.frame(d2[, 1], drop = False, colnames = c('Wor
ds'))
> View(dfm)
> dfm$length <- nchar(as.character(dfm$word))
> View(dfm)
> plot(dfm)
> library(dplyr)

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

> new_df <- dfm %>%
+ arrange(desc(length)) %>%
+ select(word, length)
> View(new_df)
>
```

	word	length
1	'_rikk-tikk-tikki-tikki-tchk!_'	31
2	_rikki-tikki-tck-tck!_	22
3	'_ding-dong-lock!_	18
4	_ding-dong-tock!_'	18
5	_scratch-scratch_	17
6	rikki-tikki-tavi.	17
7	window-pane,--the	17
8	rikki-tikki-tavi	16
9	_rikk-tck-tck!_'	16
10	hunting-ground,'	16

10 Longest Sentences:

```
s <- as.String(inp)
sent_token_annotator <- Maxent_Sent_Token_Annotator()
sent_token_annotator
a1 <- annotate(inp, sent_token_annotator)
a1
s[a1]
sdf<-as.data.frame(s[a1])
colnames(sdf)<-c("sentence")
sdf$length = nchar(as.character(sdf$sentence))
new_sdf<-sdf %>%
arrange(desc(length)) %>%
select(sentence,length)
```

```
> is.vector(s[i])
Error in is.Span(i) : object 'i' not found
> is.vector(s[a1])
[1] TRUE
> sdf<-as.data.frame(s[a1])
> View(sdf)
> colnames(sdf) <- c("sentence")
> sdf$length = nchar(as.character)
Error in nchar(as.character) :
  cannot coerce type 'builtin' to vector of type 'character'
> sdf$length = nchar(as.character(sdf$sentence))
>
```

```
> sdf$length = nchar(as.character)
Error in nchar(as.character) :
  cannot coerce type 'builtin' to vector of type 'character'
> sdf$length = nchar(as.character(sdf$sentence))
> library(dplyr)

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

> new_sdf <- sdf %>%
+ arrange(desc(length)) %>%
+ select(sentence, length)
>
```

	sentence	length
1	That night, at dinner, walking to and fro among the ...	487
2	Rikki-tikki held on with his eyes shut, for now he wa...	485
3	Early in the morning Rikki-tikki came to early breakf...	461
4	At last there were only three eggs left, and Rikki-tik...	425
5	When Rikki got to the house, Teddy and Teddy's mot...	342
6	Then he was battered to and fro as a rat is shaken b...	313
7	I must get to the melon-bed, and if I went there now...	310
8	His eyes and the end of his restless nose were pink;...	305
9	Still, the instant's delay brought Rikki-tikki up to her...	299
10	He was afraid for the minute; but it is impossible for...	289

e. Applying functions from zipfR

i. bootstrap.confint()

This function calculates the confidence intervals using a normal approximation of the bootstrap distribution. The central tendency is given by the sample mean, and the spread is given by the standard deviation.

bootstrap.confint(rtttm, level=0.95, method = "normal")

```
docs
 analysethis.txt
2.5%         -6.299871
97.5%        11.566666
center        2.633397
spread        4.557874
>
```

ii. Finding the document term matrix and word frequencies to further create a tfl file:

rttdtm<-DocumentTermMatrix(rttstop)
View(rttdtm)
freq<-colSums(as.matrix(rttdtm))
rttdtm
length(freq)
ord<-order(freq,decreasing=TRUE)
freq[head(ord)]
freq[tail(ord)]

```
> rttdtm
<<DocumentTermMatrix (documents: 1, terms: 1042)>>
Non-/sparse entries: 1042/0
Sparsity           : 0%
Maximal term length: 22
Weighting          : term frequency (tf)
> length(freq)
[1] 1042
> ord<-order(freq,decreeasing=TRUE)
Error in order(freq, decreeasing = TRUE) : argument lengths differ
> ord<-order(freq,decreasing=TRUE)
> freq[head(ord)]
rikkitikki       said        nag    nagaina     teddys       will
        86         49         39         36         27         26
> freq[tail(ord)]
     writing writingtable          yes    yesterday          yet        youve
           1            1            1            1            1            1
>
```

Keeping only words with lengths 4-20:
rttdtmr<-DocumentTermMatrix(rttstop, control = list(wordLengths=c(4,20)))
rttdtmr
freqr<-colSums(as.matrix(rttdtmr))
ordr<-order(freqr,decreasing=TRUE)
freqr[head(ordr)]
freqr[tail(ordr)]

```
> freqr[head(ordr)]
rikkitikki       said    nagaina     teddys       will       head
        86         49         36         27         26         25
> freqr[tail(ordr)]
     wouldnt      wrapped      writing writingtable    yesterday        youve
           1            1            1            1            1            1
```

Converting to a tfl file, then importing it back to observe the frequency statistics using **read.tfl:**

```
write.table(freqr, file = "freqr.txt", sep = "\n",
            row.names = FALSE)

rtt.tfl<-read.tfl("/Users/aarvithadeshwar/Desktop/freqr.txt")
summary(rtt.tfl)
```

```
> summary(rtt.tfl)
zipfR object for frequency spectrum
Sample size:     N  = 2415
Vocabulary size: V  = 962
Range of freq's: f  = 1 ... 86
Mean / median:   mu = 2.510395 ,  M = 1
Hapaxes etc.:    V1 = 590 ,  V2 = 145
```

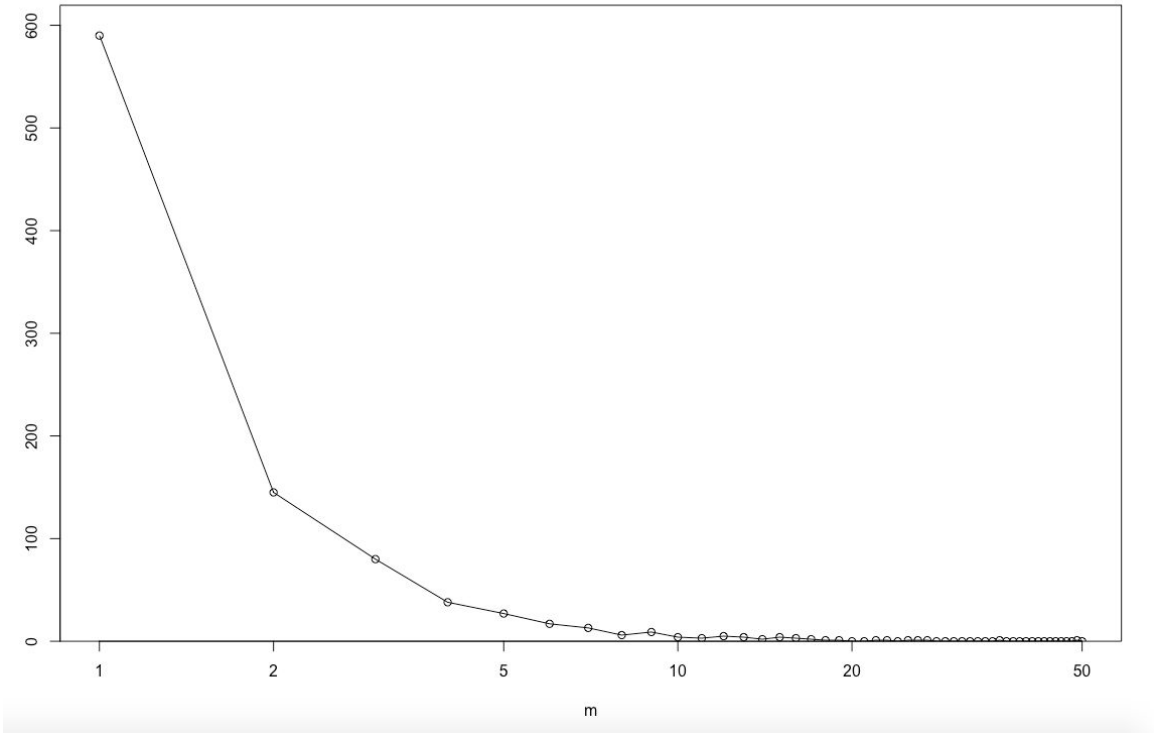
iii. Generating a spectrum using tfl2spc:

```
rtt.spc<- tfl2spc(rtt.tfl)
summary(rtt.spc)
```

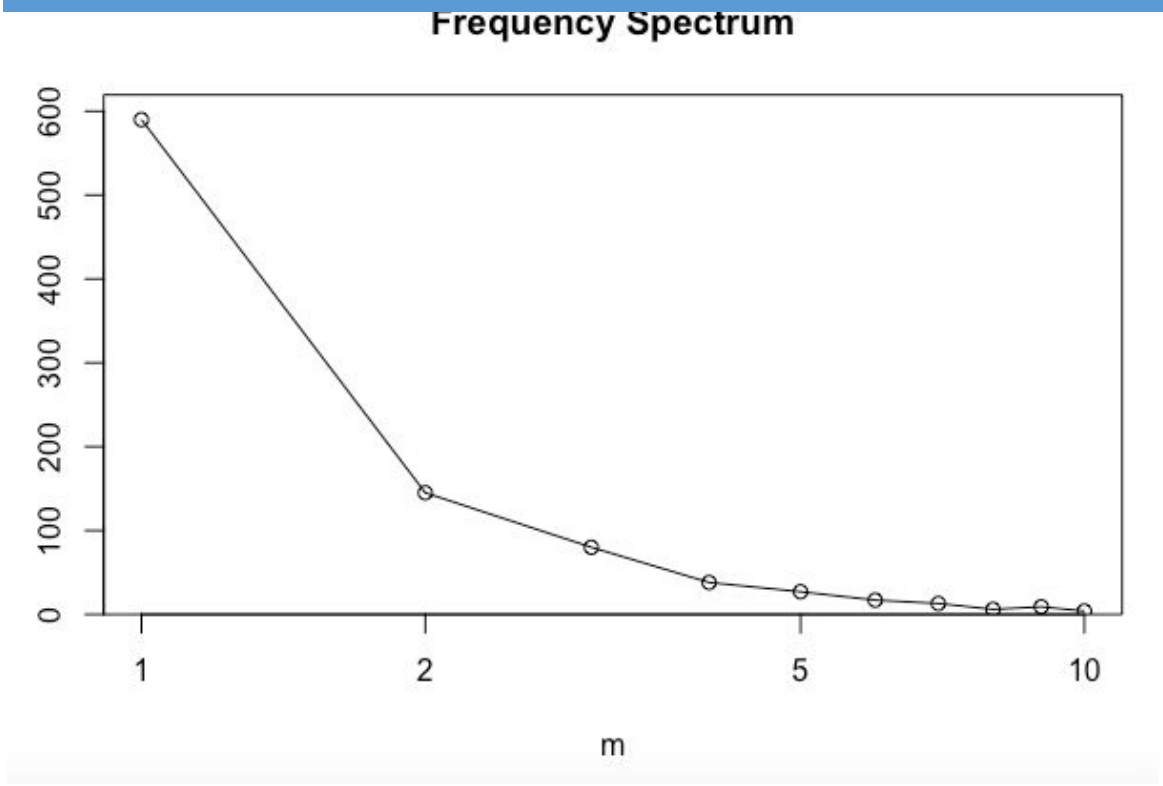
```
> rtt.spc<- tfl2spc(rtt.tfl)
> summary(rtt.spc)
zipfR object for frequency spectrum
Sample size:     N  = 2415
Vocabulary size: V  = 962
Class sizes:     Vm = 590 145 80 38 27 17 13 6 ...
```

iv. Plotting the frequency spectrum:

```
plot(rtt.spc,log="x")
```

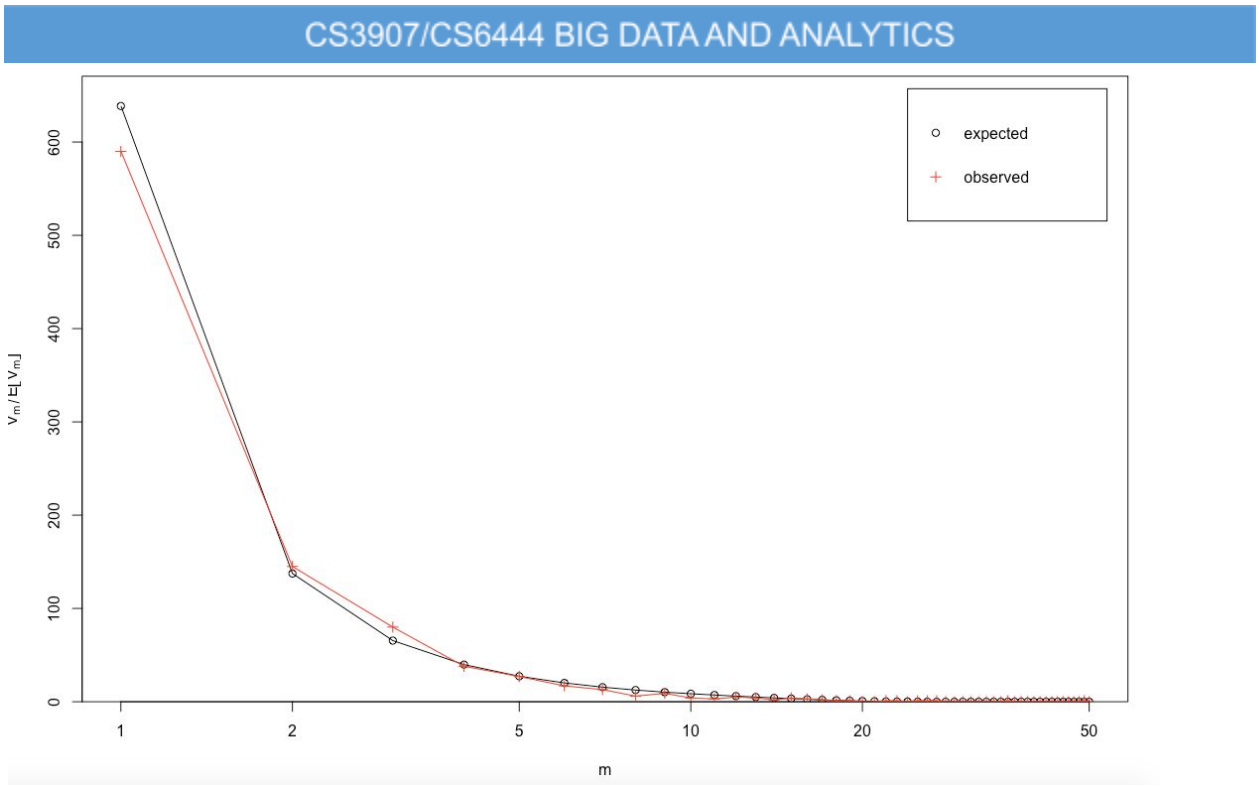


Plotting only the first 10 elements:
plot(rtt.spc,log="x", m.max=10)

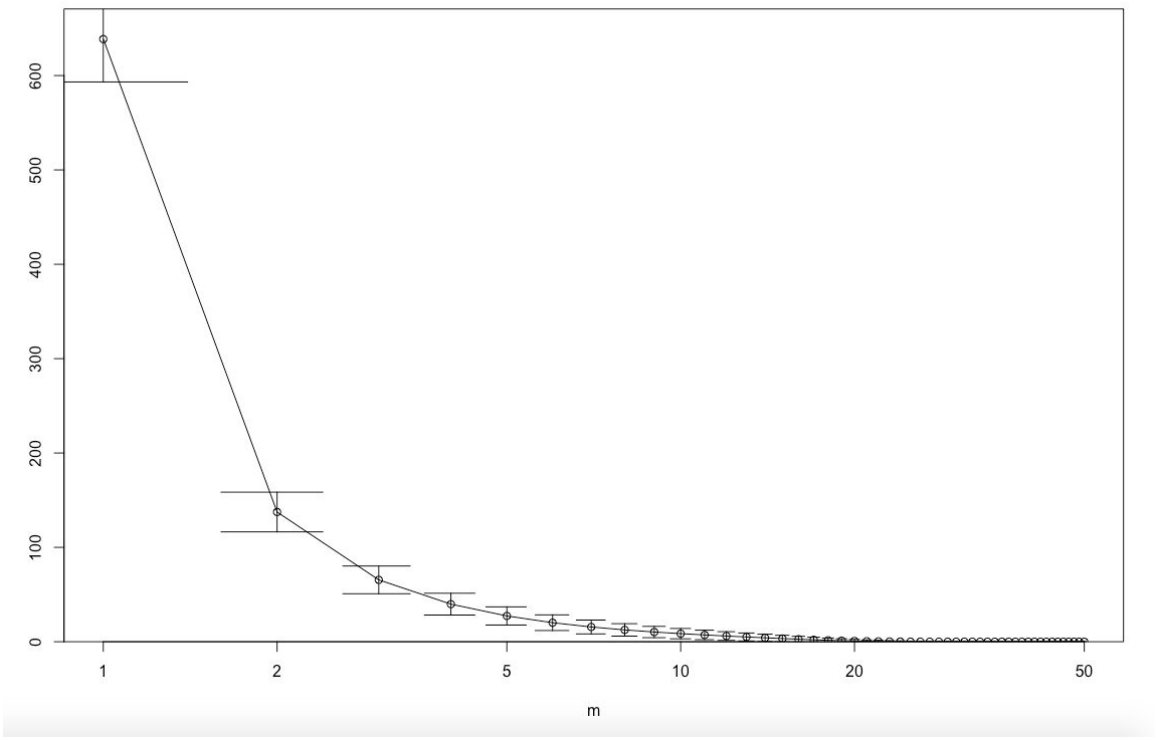


Computing the zm model and expected spectrum, then comparing the observed and expected spectra:

```
zm<-lnre("zm",rtt.spc)
zm.spc<-lnre.spc(zm,N(rtt.spc))
plot(zm.spc, rtt.spc, legend = c("expected", "observed"), log="x")
```

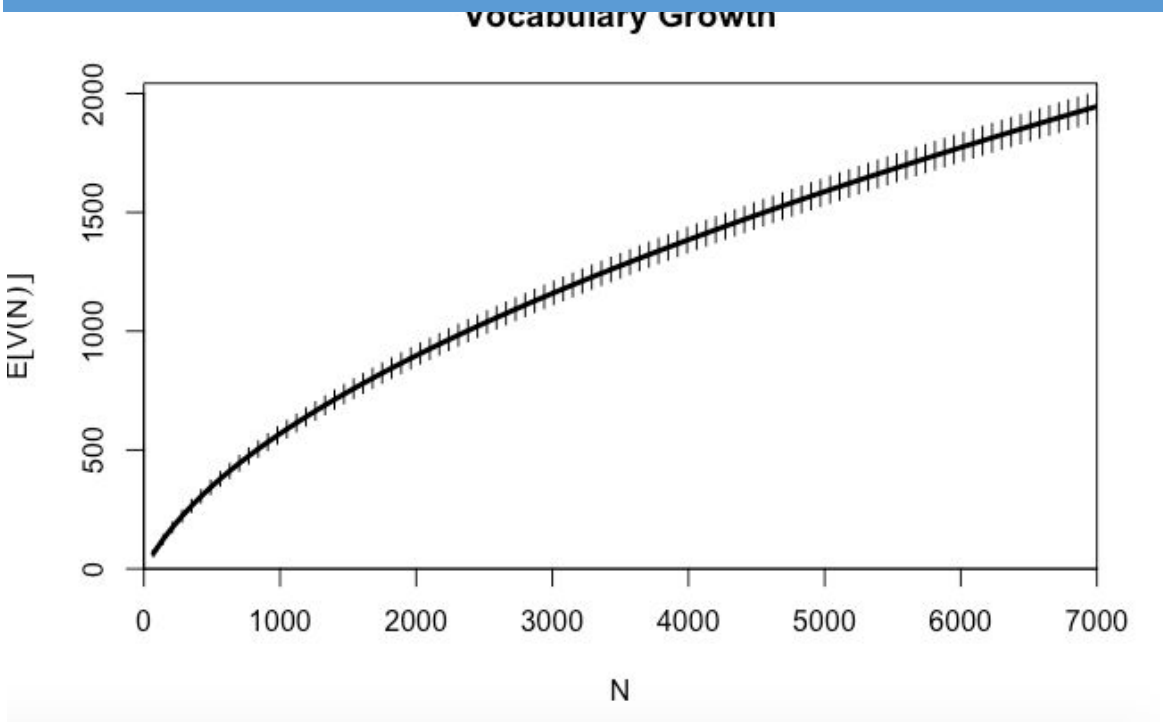


Now, plotting the variances of the spectra to also obtain the 95% confidence interval:
zm.spc<-lnre.spc(zm,N(rtt.spc), variances = TRUE)
plot(zm.spc, log = "x")



v. Plotting vocabulary growth curves:

```
zm.vgc<- lnre.vgc(zm, (1:100)*70, variances=TRUE)
summary(zm.vgc)
print(zm.vgc)
plot(zm.vgc)
```



f. Bigrams and trigrams words with length is greater than 6 characters in the 10 longest sentences

Bigrams:

```
newcorp<-Vcorpus(VectorSource(dfst$sentence))
tm<-TermDocumentMatrix(newcorp)
m3<-as.matrix(tm)
v3<-sort(rowSums(m3), decreasing = TRUE)
d3<- data.frame(word = names(v3), freq=v3)
dgen<-as.data.frame(d3[, 1], drop = False, colnames = c('word'))
dgen$length<-nchar(as.character(dgen$word))
dgen_new<- dgen %>%
arrange(desc(length)) %>%
select(word,length)
gen<-as.data.frame(dgen_new[1:86,])
library("quanteda")
myDfm <- tokens(txt) %>%
        tokens_remove("\\p{P}", valuetype = "regex", padding = TRUE) %>%
        tokens_remove(stopwords("english"), padding  = TRUE) %>%
        tokens_ngrams(n = 2) %>%
        dfm()
featnames(myDfm)
```

81	stuffed	7
82	things;	7
83	through	7
84	tumbled	7
85	walking	7
86	whether	7
87	mother	6
88	little	6

Showing 80 to 89 of 323 entries, 2 total c

Console | **Terminal** | **Jobs**

```
> txt<-corp[[1]]$content
> myDfm <- tokens(txt) %>%
+     tokens_remove("\\p{P}", valuetype = "regex", padding = TRU
E) %>%
+     tokens_remove(stopwords("english"), padding  = TRUE) %>%
+     tokens_ngrams(n = 2) %>%
+     dfm()
> featnames(myDfm)
 [1] "fro_among"         "three_times"       "nice_things"
 [4] "great_circles"     "remembered_nag"    "tin_dipper"
 [7] "get_red"           "tin_side"          "long_war"
[10] "war_cry"           "teddy_carried"     "eyes_shut"
[13] "quite_sure"        "must_get"          "big_man"
[16] "man_picked"        "little_fellow"     "little_chap"
[19] "never_hold"        "lives_now"         "one_idea"
[22] "mother_came"       "white_face"        "eggs_like"
[25] "spent_half"        "night_shaking"     "restless_nose"
[28] "really_broken"     "forty_pieces"      "tail_till"
[31] "looked_like"       "verandah_riding"   "long_grass"
[34] "boiled_egg"        "delay_brought"     "laps_one"
[37] "mongoose_always"   "always_hopes"      "nag_used"
[40] "carefully_told"    "told_rikki"        "came_across"
```

Trigrams:

```
myDfm <- tokens(txt) %>%
        tokens_remove("\\p{P}", valuetype = "regex", padding = TRUE) %>%
        tokens_remove(stopwords("english"), padding  = TRUE) %>%
        tokens_ngrams(n = 3) %>%
        dfm()
featnames(myDfm)
```

```
> myDfm3 <- tokens(txt) %>%
+     tokens_remove("\\p{P}", valuetype = "regex", padding = TRU
E) %>%
+     tokens_remove(stopwords("english"), padding  = TRUE) %>%
+     tokens_ngrams(n = 3) %>%
+     dfm()
> featnames(myDfm3)
 [1] "long_war_cry"          "big_man_picked"
 [3] "little_white_teeth"    "mongoose_always_hopes"
 [5] "carefully_told_rikki"  "came_across_white"
 [7] "across_white_men"      "three_eggs_left"
 [9] "led_nagaina_toward"    "smashed_two_eggs"
>
```

g.

Function quanteda()

i. dfm()

The dfm() function derives a document-feature matrix. The features vary, based on the input format. In this instance, the input is a corpus and the features are the words in the corpus.

```
rcon<-rttpun[[1]]$content
rdfm<-dfm(rcon)
rdfm
```

```
> rcon<-rttpun[[1]]$content
> rdfm<-dfm(rcpm)
Error in is(x, "dfm") : object 'rcpm' not found
> rdfm<-dfm(rcon)
> rdfm
Document-feature matrix of: 622 documents, 1,153 features (99.3%
 sparse).
>
```

ii. featnames()

This function returns all the feature names in the document-feature matrix. Here, the words in the document have been displayed as featnames.

featnames(rdfm)

```
> featnames(rdfm)
   [1] "rikkitikkitavi"         "this"
   [3] "is"                     "the"
   [5] "story"                  "of"
   [7] "great"                  "war"
   [9] "that"                   "fought"
  [11] "singlehanded"           "through"
  [13] "bathrooms"              "big"
  [15] "bungalow"               "in"
  [17] "segowlee"               "cantonment"
  [19] "darzee"                 "tailorbird"
  [21] "helped"                 "him"
  [23] "and"                    "chuchundra"
  [25] "muskrat"                "who"
  [27] "never"                  "comes"
  [29] "out"                    "into"
  [31] "middle"                 "floor"
  [33] "but"                    "always"
  [35] "creeps"                 "round"
  [37] "by"                     "wall"
```

iii. kwic()

For a given text and keyword, the function kwic() returns a list of its instances in context, with source and word index number. There are 15 occurrences of the term “house” in this document and as seen below, each of them have been represented in context (adjacent words). The index and term numbers have also been cited for each.

qk<-kwic(rttpun[[1]]$content, "house")
qk

```
> qk

   [text29, 6]    they took him into the | house |
   [text58, 5]            and out of the | house |
  [text65, 11]    find out about in this | house |
   [text69, 9] that day roaming over the | house |
  [text91, 10]   to live in the generals | house |
 [text178, 10]  the gravel path near the | house |
  [text213, 5]      teddy shouted to the | house |
  [text244, 2]                      the | house |
  [text277, 4]   rikkitikki listened the | house |
  [text293, 3]                 when the | house |
 [text438, 10]     broken the boy in the | house |
 [text450, 11]      night the boy in the | house |
  [text474, 7]  i led nagaina toward the | house |
 [text588, 13]       will go back to the | house |
  [text602, 6]     when rikki got to the | house |

and a big man picked
all day long lets give
he said to
he nearly drowned
```

iv. textstat_collocations()

This function identifies and scores phrases or adjacent collocations from a given text. As illustrated below, the count of each of the phrases, along with attributes like length have been cited. A log-linear model is used to calculate the “lambda” the values.

textstat_collocations(rcon)

	collocation	count	count_nested	length	lambda
1	teddys mother	11	0	2	5.694107
2	big man	10	0	2	6.448211
3	go away	3	0	2	5.070193
4	never come	3	0	2	4.808450
5	teddys father	4	0	2	4.072099
6	singing song	2	0	2	6.118502
7	middle room	2	0	2	6.792344
8	used live	2	0	2	6.792344
9	white teeth	2	0	2	6.203659
10	move strike	2	0	2	6.203659
11	bathroom sluice	2	0	2	6.203659
12	shall wait	2	0	2	6.966698
13	five feet	2	0	2	7.814893
14	teddys shoulder	3	0	2	4.541256
15	long grass	2	0	2	4.977184
16	round wall	2	0	2	5.117273
17	knew better	2	0	2	6.202760
18	flew nest	2	0	2	6.202760
19	man killed	2	0	2	5.353662
20	darzees wife	6	0	2	8.566589
21	one side	2	0	2	6.077147

v. topfeatures()

The function topfeatures() can return a list of the most or least frequently used features in a document-feature matrix. Here, 50 of the most frequently-occurring features are lists in a decreasing order.

topfeatures(rdfm, n = 50, decreasing = TRUE)

```
> topfeatures(rdfm, n = 50, decreasing = TRUE)
rikkitikki       said        nag    nagaina     teddys
        86         49         39         36         27
      head     darzee     little        big       came
        25         23         22         21         19
     teddy       like       eyes   mongoose       back
        18         17         17         16         16
    mother      never     garden      house      still
        16         15         15         15         15
       man       time       eggs         go      round
        14         14         14         14         13
      tail       dead       just      grass     father
        13         13         13         12         12
       eat       went     snakes       knew        one
        12         12         12         12         11
     rikki      heard       wife       till     looked
        11         11         11         10         10
       get       bite       nest chuchundra        now
        10         10         10          9          9
       see        sat      night       look        egg
         9          9          9          9          9
> |
```

Function stringi():

i stri_read_lines(filename)
Reads a text file, re-encodes it, and splits it into text lines

```
[555] "thorn-bush, and as he was running Rikki-tikki heard Darzee still"
[556] "singing his foolish little song of triumph. But Darzee's wife was"
[557] "wiser. She flew off her nest as Nagaina came along and flapped her"
[558] "wings about Nagaina's head. If Darzee had helped they might have"
[559] "turned her; but Nagaina only lowered her hood and went on. Still, the"
[560] "instant's delay brought Rikki-tikki up to her, and as she plunged"
[561] "into the rat-hole where she and Nag used to live, his little white"
[562] "teeth were clenched on her tail, and he went down with her--and very"
[563] "few mongooses, however wise and old they may be, care to follow a"
[564] "cobra into its hole. It was dark in the hole; and Rikki-tikki never"
[565] "knew when it might open out and give Nagaina room to turn and strike"
[566] "at him. He held on savagely, and struck out his feet to act as brakes"
[567] "on the dark slope of the hot, moist earth."
[568] ""
[569] "Then the grass by the mouth of the hole stopped waving, and Darzee"
[570] "said: 'It is all over with Rikki-tikki! We must sing his death-song."
[571] "Valiant Rikki-tikki is dead! For Nagaina will surely kill him"
[572] "underground.'"
[573] ""
[574] "So he sang a very mournful song that he made up on the spur of the"
[575] "minute, and just as he got to the most touching part the grass"
[576] "quivered again, and Rikki-tikki, covered with dirt, dragged himself"
[577] "out of the hole leg by leg, licking his whiskers. Darzee stopped with"
[578] "a little shout. Rikki-tikki shook some of the dust out of his fur and"
[579] "sneezed. 'It is all over,' he said. 'The widow will never come out"
[580] "again.' And the red ants that live between the grass stems heard him,"
[581] "and began to troop down one after another to see if he had spoken the"
[582] "truth."
[583] ""
[584] "Rikki-tikki curled himself up in the grass and slept where he"
```

ii stri_flatten

Flatten a String,Joins the elements of a character vector into one string.

```
> rikki_flat_string <- stri_flatten(rikki_as_string)
> rikki_flat_string
[1] "RIKKI-TIKKI-TAVIThis is the story of the great war that Rikki-tikki-tavi foughtsingle-handed, through the bath-rooms of the big bungal
ow in Segowleecantonment. Darzee, the tailor-bird, helped him, and Chuchundra, themusk-rat, who never comes out into the middle of the floo
r, butalways creeps round by the wall, gave him advice; but Rikki-tikki didthe real fighting.He was a mongoose, rather like a little cat in
 his fur and his tail,but quite like a weasel in his head and habits. His eyes and the endof his restless nose were pink; he could scratch
 himself anywhere hepleased, with any leg, front or back, that he chose to use; he couldfluff up his tail till it looked like a bottle-brus
h, and hiswar-cry, as he scuttled through the long grass, was:'_Rikk-tikk-tikki-tikki-tchk!_'One day, a high summer flood washed him out of
 the burrow where helived with his father and mother, and carried him, kicking andclucking, down a roadside ditch. He found a little wisp o
f grassfloating there, and clung to it till he lost his senses. When herevived, he was lying in the hot sun on the middle of a garden path,
very draggled indeed, and a small boy was saying: 'Here's a deadmongoose. Let's have a funeral.''No,' said his mother; 'let's take him in a
nd dry him. Perhaps heisn't really dead.'They took him into the house, and a big man picked him up between hisfinger and thumb, and said he
 was not dead but half choked; so theywrapped him in cotton-wool, and warmed him and he opened his eyes andsneezed.'Now,' said the big man
 (he was an Englishman who had just moved intothe bungalow); 'don't frighten him and we'll see what he'll do.'It is the hardest thing in th
e world to frighten a mongoose, becausehe is eaten up from nose to tail with curiosity. The motto of all themongoose family is 'Run and fin
d out'; and Rikki-tikki was a truemongoose. He looked at the cotton-wool, decided that it was not goodto eat, ran all round the table, sat
 up and put his fur in order,scratched himself, and jumped on the small boy's shoulder.'Don't be frightened, Teddy,' said his father. 'Tha
t's his way ofmaking friends.''Ouch! He's tickling under my chin,' said Teddy.Rikki-tikki looked down between the boy's collar and neck, sn
uffed athis ear, and climbed down to the floor, where he sat rubbing hisnose.'Good gracious,' said Teddy's mother, 'and that's a wild creat
ure! Isuppose he's so tame because we've been kind to him.''All mongooses are like that,' said her husband. 'If Teddy doesn'tpick him up by
 the tail, or try to put him in a cage,  he'll run inand out of the house all day long. Let's give him something to eat.'They gave him a li
ttle piece of raw meat. Rikki tikki liked itimmensely, and when it was finished he went out into the verandah andsat in the sunshine and fl
uffed up his fur to make it dry to theroots. Then he felt better.'There are more things to find out about in this house,' he said tohimsel
f, 'than all my family could find out in all their lives. Ishall certainly stay and find out.'He spent all that day roaming over the house.
 He nearly drownedhimself in the bath tubs, put his nose into the ink on awriting-table, and burnt it on the end of the big man's cigar, fo
r heclimbed up in the big man's lap to see how writing was done. Atnightfall he ran into Teddy's nursery to watch how the kerosene-lampswer
e lighted, and when Teddy went to bed Rikki-tikki climbed up too;but he was a restless companion, because he had to get up and attendto eve
ry noise all through the night, and find out what made it.Teddy's mother and father came in, the last thing, to look at theirboy, and Rikki
-tikki was awake on the pillow. 'I don't like that,'said Teddy's mother; 'he may bite the child.' 'He'll do no suchthing,' said the father.
 'Teddy's safer with that little beast thanif he had a bloodhound to watch him. If a snake came into the nurserynow----'But Teddy's mother
 wouldn't think of anything so awful.Early in the morning Rikki-tikki came to early breakfast-in theverandah riding on Teddy's shoulder, an
```

iii stri_enc_detect()

Detect Character Set and Language,This function uses the ICU engine to determine the character set, or encoding, of character data in an unknown format.

```
   Encoding Language Confidence
1  ISO-8859-1       en       0.77
2       UTF-8                0.15
3  ISO-8859-9       tr       0.15
4  ISO-8859-2       hu       0.14
5    UTF-16BE                0.10
6    UTF-16LE                0.10
7   Shift_JIS       ja       0.10
8     GB18030       zh       0.10
9      EUC-JP       ja       0.10
10     EUC-KR       ko       0.10
11       Big5       zh       0.10
```

from this function we can confidently say that the text is english

iv stri_count_boundaries()

These functions determine the number of text boundaries (like character, word, line, or sentence boundaries) in a string.

```
> stri_count_boundaries(rikki_flat_string, "sentence")
[1] 300
> stri_count_boundaries(rikki_flat_string, "word")
[1] 11932
> stri_count_boundaries(rikki_flat_string, "character")
[1] 29879
>
```

this can be used to talk about the overall length of the text using different boudaries

v. stri_extract_all_words

extracts all words from a string

```
> stri_extract_all_words(rikki_flat_string, simplify=FALSE)
[[1]]
  [1] "RIKKI"     "TIKKI"     "TAVI"      "This"      "is"        "the"       "story"     "of"        "the"       "great"
 [11] "war"       "that"      "Rikki"     "tikki"     "tavi"      "fought"    "single"    "handed"    "through"   "the"
 [21] "bath"      "rooms"     "of"        "the"       "big"       "bungalow"  "in"        "Segowlee"  "cantonment" "Darzee"
 [31] "the"       "tailor"    "bird"      "helped"    "him"       "and"       "Chuchundra" "the"       "musk"      "rat"
 [41] "who"       "never"     "comes"     "out"       "into"      "the"       "middle"    "of"        "the"       "floor"
 [51] "but"       "always"    "creeps"    "round"     "by"        "the"       "wall"      "gave"      "him"       "advice"
 [61] "but"       "Rikki"     "tikki"     "did"       "the"       "real"      "fighting"  "He"        "was"       "a"
 [71] "mongoose"  "rather"     "like"      "a"         "little"    "cat"       "in"        "his"       "fur"       "and"
 [81] "his"       "tail"      "but"       "quite"     "like"      "a"         "weasel"    "in"        "his"       "head"
 [91] "and"       "habits"    "His"       "eyes"      "and"       "the"       "end"       "of"        "his"       "restless"
[101] "nose"      "were"      "pink"      "he"        "could"     "scratch"   "himself"   "anywhere"  "he"        "pleased"
[111] "with"      "any"       "leg"       "front"     "or"        "back"      "that"      "he"        "chose"     "to"
[121] "use"       "he"        "could"     "fluff"     "up"        "his"       "tail"      "till"      "it"        "looked"
[131] "like"      "a"         "bottle"    "brush"     "and"       "his"       "war"       "cry"       "as"        "he"
```

Function tidytext():

i. unnset_tokens()

Split a column into tokens using the tokenizers package, splitting the table into one-token-per-row. This function supports non-standard evaluation through the tidyeval framework

```
> text_df %>%  unnest_tokens(word, value)
# A tibble: 5,883 x 1
   word
   <chr>
 1 rikki
 2 tikki
 3 tavi
 4 this
 5 is
 6 the
 7 story
 8 of
 9 the
10 great
# ... with 5,873 more rows
>
```

ii. get_sentiments('bing')

Get a tidy data frame of a single sentiment lexicon

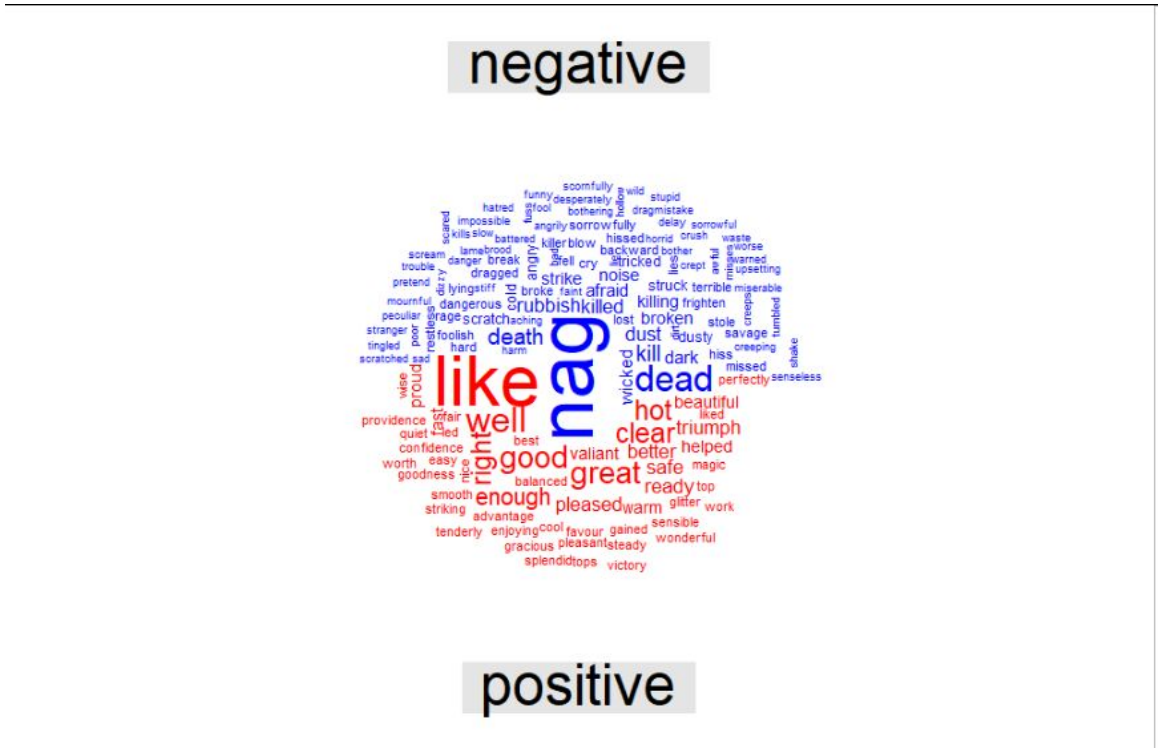
```
> get_sentiments("bing")
# A tibble: 6,786 x 2
   word        sentiment
   <chr>       <chr>
 1 2-faces     negative
 2 abnormal    negative
 3 abolish     negative
 4 abominable  negative
 5 abominably  negative
 6 abominate   negative
 7 abomination negative
 8 abort       negative
 9 aborted     negative
10 aborts      negative
# ... with 6,776 more rows
>
```

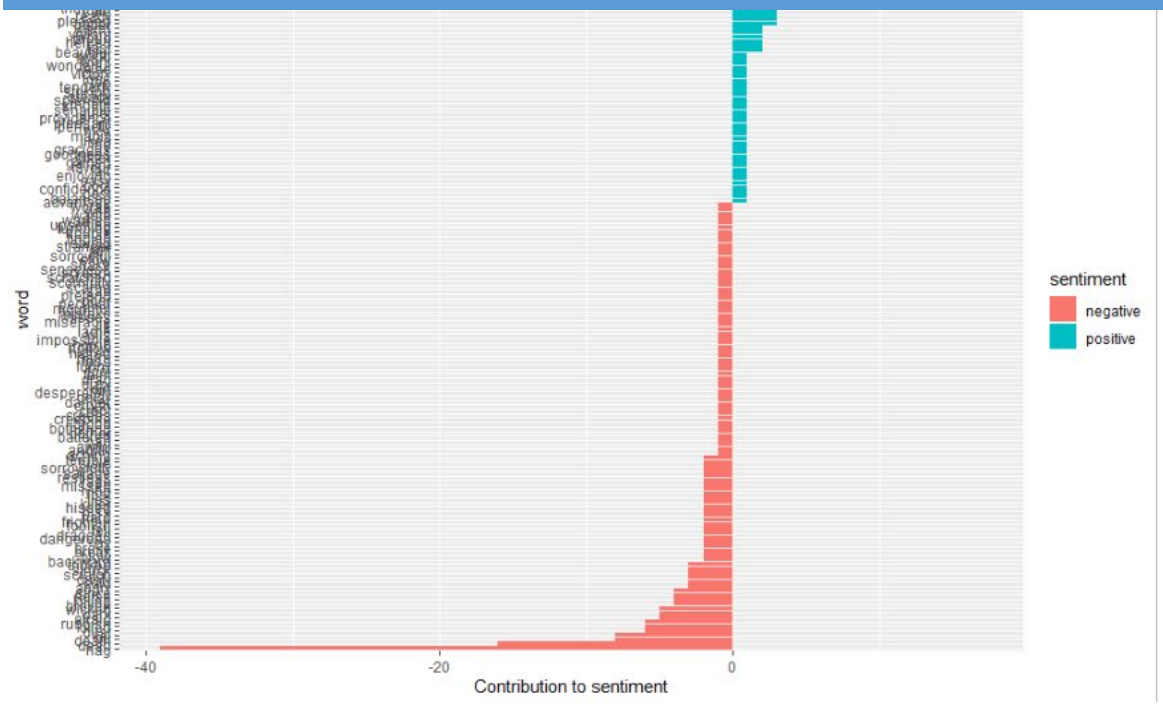
can be join with actual words in the text

```
# A tibble: 154 x 3
   word       n sentiment
   <chr>  <int> <chr>
 1 nag       39 negative
 2 like      17 positive
 3 dead      16 negative
 4 death      8 negative
 5 kill       8 negative
 6 well       7 positive
 7 dust       6 negative
 8 good       6 positive
 9 great      6 positive
10 killed     6 negative
# ... with 144 more rows
```

this can then be used with plotting to get the overall sentiment of the text

```
library(ggplot2)
sentiments %>%
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col() +
  coord_flip() +
  labs(y = "Contribution to sentiment")
```





We can see there are way more uses of negative words. I would assume from this that story is overall negative.

iii stop_words

English stop words from three lexicons, as a data frame. The snowball and SMART sets are pulled from the tm package. Note that words with non-ASCII characters have been removed

```
> words %>% anti_join(stop_words)
Joining, by = "word"
# A tibble: 2,130 x 1
   word
   <chr>
 1 rikki
 2 tikki
 3 tavi
 4 story
 5 war
 6 rikki
 7 tikki
 8 tavi
 9 fought
10 single
# ... with 2,120 more rows
```

Function Corpus:

i. create_tcorpus

```
tCorpus containing 7200 tokens
grouped by documents (n = 1) and sentences (n = 341)
contains:
  - 4 columns in $tokens:       doc_id, sentence, token_id, token
  - 1 column in $meta:          doc_id
> |
```

ii subset

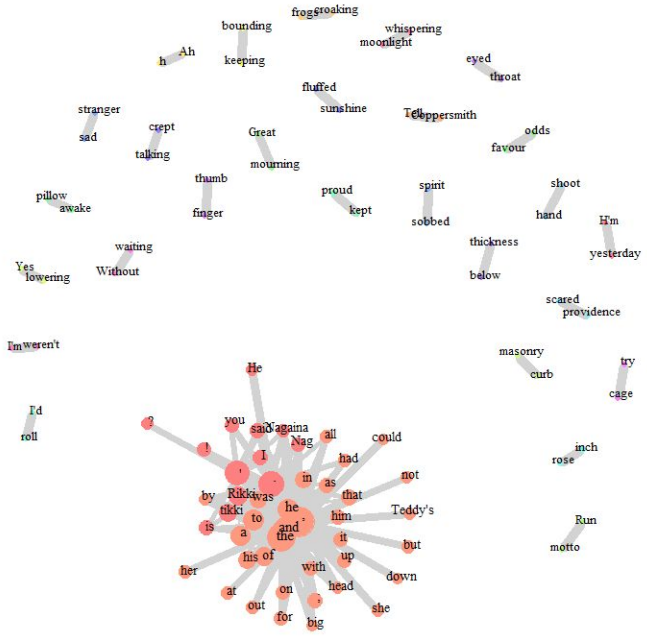
this can be used to filter out certain features. In this case i remove the nulls

```
> head(tc$tokens)
   doc_id sentence token_id token
1:      1        1        1 RIKKI
2:      1        1        2     -
3:      1        1        3 TIKKI
4:      1        1        4     -
5:      1        1        5  TAVI
6:      1        1        6  This
> tc2 = subset(tc, token!='-')
> head(tc2$tokens)
   doc_id sentence token_id token
1:      1        1        1 RIKKI
2:      1        1        3 TIKKI
3:      1        1        5  TAVI
4:      1        1        6  This
5:      1        1        7    is
6:      1        1        8   the
```

iii semnet_window

This function calculates the co-occurence of features and returns a network/graph in the igraph for-mat, where nodes are tokens and edges represent the similarity/adjacency of tokens. Co-occurence is calcuated based on how often two tokens co-occurr within a given token distance.

```
> g = semnet_window(tc2, 'token')
> gb = backbone_filter(g, alpha = 0.0001, max_vertices = 100)
Used cutoff edge-weight 6.01803586833063e-05 to keep number of vertices under 100
(For the edges the original weight is still used)
> plot_semnet(gb)
> |
```



using the semnet we can plot tokens to see which ones are related to each other. In this case we see that the Proper nouns are most connect which leads me to believe it is a very character driven story.
We one interesting thing we see is that great and mourning are highly connect which seems to reinforce the finds.

OVERALL ANALYSIS:
The initial process of reading the data and cleaning it for consistency involved elimination of punctuation marks, numbers and uppercase letters. Next, stop words were removed from the data, resulting in a clean data set.
Generation of TDM and DTM helped summarize data according to frequencies. This was represented using word clouds and different levels of dendrograms. Word frequencies were then used to plot frequency distribution graphs and vocabulary growth curvess using zipfR.
Many packages have similar capabilities some of the packages make it easier to do things. stringi was the best for doing simple text clean up and ingesting text, because the data object were less opinionated. TidyText was great for tokenizing data and for adding more information to text such as semantics. CorpusTools had capabilities that were not readily available in the other package like building networks for text analysis. From doing text analysis on Rikki Tikki using these packages I would claim is that is has an overall negative sentiment and is very character driven story.