Gadsby Text Mining

Ameer Dharamshi

1 Introduction and Motivation

Traditional introductory statistics courses tend to focus on analyzing data in a numeric format or in frequencies of categorical data (categorical information presented in numeric form). However, there is endless information stored and communicated through text. Humans are constantly bombarded with textual information in the forms of news articles, e-mails and social media, among others. In most cases there is far more textual information available on any given subject than can be manually processed by a human. As such, employing the aid of computers to assist in sifting through text is a valuable skill. Given the notions of intention and subtext, it may at first seem odd to employ a computer to interepret text. However, there are techniques developed to address these concepts and they will be be illustrated below.

Regarding this exercise specifically, it is meant to offer a framework for introducing students to text mining techniques. To illustrate these concepts, we will select some literary piece and walk through the process of extracting the text from an online source, decomposing the text into characters, words, paragraphs, etc. and analyze the resulting data set for attributes of interest. The hope is that upon completing a similar exercise, a student will be equipped with the tools to be able to extract, process and analyze online text based information for their own purposes.

2 The Data Set

As the motivation of this project is to dive into the world of text mining, it seems natural to choose a book or other literary work that is publicly accessible electronically. In order to help guide the analysis portion, a piece that has some noteworthy feature is an optimal choice. The novel *Gadsby* by Ernest Vincent Wright is famous for being "A Story of 50,000 Words Without Using the Letter "E"". The lack of the letter "E" offers an interesting quirk that can be proven using text mining techniques. In the process of testing this fact, a student will be required to separate the data set into words, characters or other groupings. Having already processed the data into a easily manipulated from offers a good starting point for further analysis which is only limited by the student's imagination.

Gadsby is available for public use on Wikisource at https://en.wikisource.org/wiki/Gadsby. We can analyze some of the features of the novel such as word/character concentration and other categorical attributes along with the content of the novel. For these purposes, we will need to obtain data sets consisting of all the words and characters with the ability to sort. We can use the tidytext library to organize the text in the tidy text format. Tidy text tables are tables organized with one token (usually words) per row. Ideally, we want to construct a tidy dataset to facilitate easier manipulation as we will then be able to perform tasks such as sorting and searching.

3 Wikisource Extraction

There are numerous packages and resources in R that can be used to extract information from webpages. Before selecting an approach, we should examine the source to search for features in the structure that can be used to automate extraction. Opening up the Wikisource page and clicking through the chapters, we can see that the book is organized into 43 webpages, 1 per chapter with URL "https://en.wikisource.org/wiki/Gadsby/Chapter_X" where "X" is the chapter numbers. Given that the URL is structured very cleanly, we can iterate through each chapter when extracting. However, the webpage has provided little insight on how to perform the extraction within our for loop. As can be seen in Figure 1 below, the webpage is very clean

looking but we need to examine the source code to identify how the data is stored and search for any patters than can be used to extract and clean the data.

Figure 1: Sample Wikisource Chapter Layout (Chapter 1)

Figure 2 below shows a small piece of the HTML source containing the text. We can see the first few sentences present. However, we can see that the text is not stored cleanly. There are links embedded in the paragraphs as hyperlinks on specific words as well as text formatting tags. These are noteworthy features that will complicate the extraction and cleaning process as we must be able to guarantee that we do not catch these unwanted features.

At this point, I think it would be useful to introduce students to reading through HTML Source Code and introducing them to the range of tools available in R. While I don't think this is the primary takeway of thise exercise, the students need exposure to the extent that they are capable of coming up with a sufficient extraction strategy. In performing this exercise, my first attempt at extracting clearly demonstrates my lack of experience in this task. After reading up about the tools available in R, my second attempt offers a much cleaner solution.

3.1 First Attempt

Given that the data is embedded in the HTML source similarly for each chapter, at first glance it makes sense to extract all of the HTML source. After extraction, we can then remove all of the unwanted html syntax once the text has been retrieved. Extraction can be performed using the "RCurl" package and cleaning can be done using regex processing methods and "stringr" package tools. As discussed above, we can iterate through each chapter's URL, extract the source code and paste it in a vector.

```
#base URL for the Gadsby chapters
base <- "https://en.wikisource.org/wiki/Gadsby/Chapter_"

#extract_chapters function extracts the chapters specified by the users
extract_chapters <- function(chapterlist, baseURL){
  n <- length(chapterlist)
```

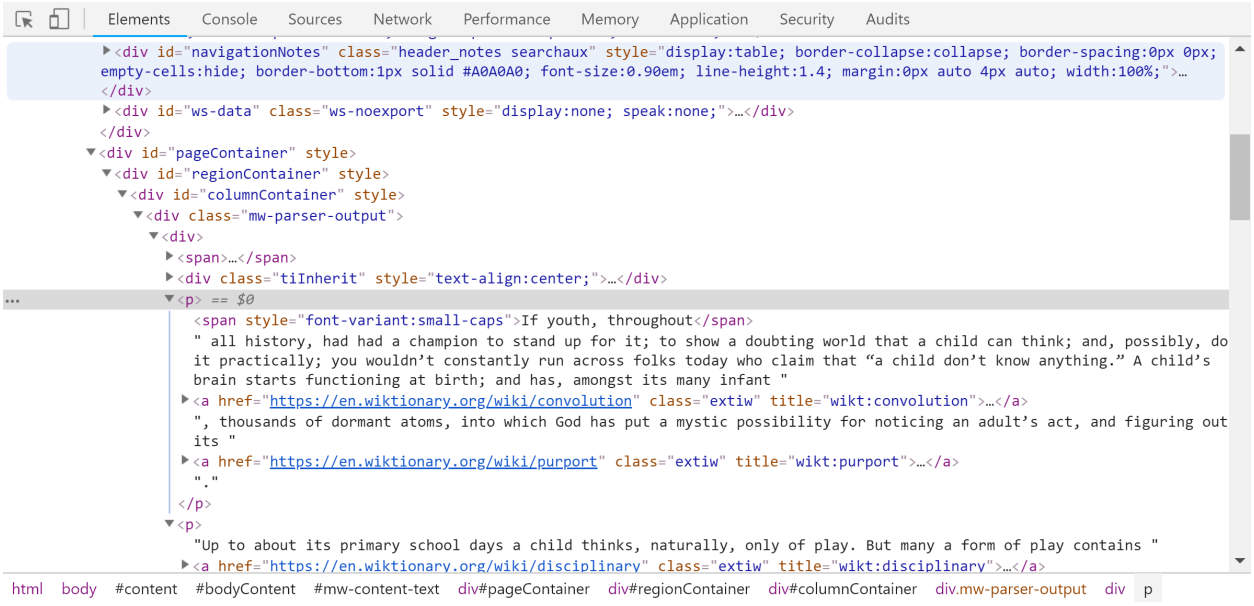


Figure 2: Sample Wikisource HTML Source Code (Chapter 1)

```
  chaps <- rep(NA, n)
  for(i in 1:n){
    url <- paste(baseURL,chapterlist[i],sep="")
    chaps[i] = getURL(url)
  }
  chaps
}

chapters <- extract_chapters(c(1:43), base)
```

At this point, the entire Gadsby Wikisource page is contained in chapters. We can now move on to cleaning. To get a sense of the task at hand, we can take a look at the first chapter. The first few lines of the chapter highlight the flaw in this approach. As the text is retrieved in one large string, we must use the strwrap function to even see the data.

```
strwrap(chapters[1])[90:100]
```

```
##  [1] "style=\"display:table; border-collapse:collapse; border-spacing:0px"
##  [2] "0px; empty-cells:hide; border-bottom:1px solid #A0A0A0;"
##  [3] "font-size:0.90em; line-height:1.4; margin:0px auto 4px auto;"
##  [4] "width:100%;\"> <div style=\"display:table-row-group;"
##  [5] "background-color:#FAFAFF;\"> <div style=\"display:table-row;\"> <div"
##  [6] "class=\"searchaux\" style=\"display:table-cell;\"></div> </div> </div>"
##  [7] "</div> <div id=\"ws-data\" class=\"ws-noexport\" style=\"display:none;"
##  [8] "speak:none;\"><span id=\"ws-article-id\">1449702</span><span"
##  [9] "id=\"ws-title\"><a href=\"/wiki/Gadsby\" title=\"Gadsby\">Gadsby</a> <U+0097>"
## [10] "<i>Chapter 1</i></span><span id=\"ws-author\">Ernest Vincent"
## [11] "Wright</span></div> </div> <div><span><span class=\"pagenum"
```

As we can see, there is a significant amount of HTML syntax that is not of interest. At first one might think you can filter out pieces of it using regex (Ex. setting all of the “” substrings to “”) but from a practicality

perspective, this is not feasible. There is no consistent pattern in the HTML tags as can be seen above. As such, if we opt to use regex, we will be forced to manually check each paragraph of each chapter to guarantee that we have cleaned all of the HTML.

3.2 Revised Solution

As discussed above, the sheer volume of HTML buried in the extracted data in the above attempt reduces the practicality of cleaning the data with regex techniques. In researching other web scraping R packages aside from RCurl, I came across rvest. Rvest is designed to enable easier harvesting of html embedded data. Chaining Rvest functions with the dplyr piping operator "%>%" can simplify the process significantly. In particular here we use the read_html function to access the webpage, html_nodes("p") to access all the of the "p" containers and then finally flow into html_text to retrieve only the textual data without the unwanted HTML code. Referring back to Figure 2, we can see that the paragraphs are stored in the "p" containers. While this was not a pattern that can be easily exploited using regex, it is a structural pattern that can be taken advantage of with the right tools.

```
#extract_chapters2 implements our second attempt at extracting the text from the
#Wikisource page using the strategy described in the above paragraph
extract_chapters2 <- function(chapterlist, baseURL){
  n <- length(chapterlist)
  chaps <- data.frame(chapter = c(), paragraph = c(), text = c(), stringsAsFactors = FALSE)
  for(i in 1:n){
    url <- paste(baseURL,chapterlist[i],sep="")
    site <- read_html(url) %>% html_nodes("p") %>% html_text()
    #The first node contains the chapter number and is not relevant to ur analysis,
    #thus we remove it
    site <- site[-1]
    l <- length(site)
    #We maintain a record of the chapter number
    ch <- rep(i, l)
    #We maintain a record of the paragraph number within a chapter
    para <- c(1:l)
    df <- data.frame(chapter = ch, paragraph = para, text = site, stringsAsFactors = FALSE)
    #Append the current chapter to the data frame with all chapters
    chaps <- bind_rows(chaps, df)
  }
  chaps
}

chapters2 <- extract_chapters2(c(1:43), base)
```

```
print(colnames(chapters2))
```

```
## [1] "chapter"   "paragraph" "text"
```

From the labels of the chapters2 data frame, we currently have a data frame with all of the text from the book organized by chapter and paragraph. This structure is a conscious choice as it will allow for easy searching if referencing the book is needed.

This is a sample paragraph from chapters2. It is the first paragraph of Chapter 1 and can be compared to both Figure 1 and Figure 2 above.

```
print(chapters2[1,])
```

```
##   chapter paragraph
## 1       1         1
```

```
##
## 1 If youth, throughout all history, had had a champion to stand up for it; to show a doubting world
```

It is important to briefly examine the data to ensure it is complete and accurate. A quick scan of the source along with chapters2 yields an unwanted line at the end.

```
## Note the last line in chapters2
chapters2[nrow(chapters2),]
```

```
##     chapter paragraph
## 964      43        23
##                                                                                                  text
## 964 \nNote: Not a word containing the letter <U+0093>E<U+0094> has appeared in this story of over 50
```

```
## This is just a comment by the author stating that he abstained from using the letter 'E'
## A quick scan of the Chapter 43 webpage confirms this. We will now remove this line to
## ensure we can validate the no 'E' claim.

chapters2 <- chapters2[-nrow(chapters2),]

#The new final row is:
chapters2[nrow(chapters2),]
```

```
##     chapter paragraph  text
## 963      43        22 FINIS
```

3.3 Lessons Learned

There's an important lesson to be learned here. There is no dobut that regex is a powerful tool, but this example provides insight on a situation where regex is not suitable. The fact that we are looking to extract the text in bulk means that there is no specific, identifiable text pattern to extract such as a date or name. At best, we can search for text in quotes, but as discussed above, the formatting and links within the text will still need to be processed. Given the number of irregularities, we would have to spend time on each paragraph individually when cleaning to ensure that we don't miss any residual HTML. This defeats the purpose of automatating the extraction. At this point we would be better off copy-pasting the data manually! However, further exploration into the structure of the source revealed a pattern that perfectly suited the tools available under the "rvest" package. If we decided we still wanted to clean the data ourselves, we could have considered that HTML is not a regular language but it is context-free. As such, we could have built a parser that removed the HTML using a context-free grammar approach.

The key lesson to take away from this section is that examining the structure of the data source in depth is imperative. Without taking the time to plan preprocessing, wasting time in inevitable. That being said, regex processing has a place in isolating specific pieces of information in text data. Specifically, finding repeated patters between many documents. This case just happens to be one large document and thus regex has no use.

Regarding student exercises, it would be beneficial to have students simulate this experience by offering a data set and have them decide on a suitable course of action with justification. If this creates too much variability, students could instead be prompted to try a few different methods and end with a comparison of each.

4 Cleaning & Processing the Data

Having extracted the text from Wikisource, we next focus on preprocessing the data into the work and character summaries mentioned in the introduction. These will act as the user-friendly base to perform

analysis.

We discussed the concept of a tidy text data structure. This is the optimal structure for performing text based analysis. Using the unnest_tokens function, we can further organize our data by breaking down the text column of chapters2 into a word column. This then follows the criteria for tidy text in that there is one token per row.

```
words <- chapters2 %>% unnest_tokens(word, text)
head(words)
```

```
##     chapter paragraph       word
## 1         1         1         if
## 1.1       1         1      youth
## 1.2       1         1 throughout
## 1.3       1         1        all
## 1.4       1         1    history
## 1.5       1         1        had
```

Looping through the alphabet, we can count the number of each letter in each word as follows.

```
chardecomp <- function(word_df, wordcol){
  temp <- word_df
  for (l in letters){
    temp[[l]] = str_count(temp[,3], l)
  }
  temp
}

chars <- chardecomp(words, 3)
head(chars)
```

```
##     chapter paragraph       word a b c d e f g h i j k l m n o p q r s t u
## 1         1         1         if 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
## 1.1       1         1      youth 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1
## 1.2       1         1 throughout 0 0 0 0 0 0 1 2 0 0 0 0 0 0 2 0 0 1 0 2 2
## 1.3       1         1        all 1 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0
## 1.4       1         1    history 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0
## 1.5       1         1        had 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
##     v w x y z
## 1   0 0 0 0 0
## 1.1 0 0 0 1 0
## 1.2 0 0 0 0 0
## 1.3 0 0 0 0 0
## 1.4 0 0 0 1 0
## 1.5 0 0 0 0 0
```

At this point, we have created a data frame with a column for each letter of the alphabet as was planned in the introductory stages of the exercise. This will be the core data frame used in our analysis.

4.1 Lessons Learned

This section was quite simple. Given how clean our data was upon extraction, there was minimal effort required to separate the words into characters. However, in other cases it will likely be more challenging and more advanced tools may be required. I did not need to use any dplyr tools here but they are powerful and can be helpful in preprocessing. Dplyr will be discussed more below in the analysis sections.

5 Analysis

There is a very wide range of features that can be investigated using the techniques we will be going through. In this case, character frequency is of particular interest due to the unique nature of Gadsby. However, this will not be the case for all text mining exercises. For example, character frequencies are not of particular interest if we are investigating news articles. The benefit of walking through an exercise lies in the experience gained through using the array of tools provided in R.

5.1 Characters

We can analyse the decomposition of Gadsby at varying levels based on the datasets constructed above. Given the emphasis placed on characters, we will first examine features dealing with character frequencies.

5.1.1 A Story of 50,000 Words Without Using the Letter "E"

To verify the claim that Gadsby is "A Story of 50,000 Words Without Using the Letter"E"", we will begin by summarizing our data sets prepared above. In particular, the character dataset can be summarized using Dplyr tools. For the sake of completeness, we will first ensure that the book does in fact have over 50,000 words.

```
print(nrow(words))
```

```
## [1] 50775
```

With the word count verfied, we can turn to the more interesting claim that there are no "E"s present.

```
#charsum removes the word and paragraph columns and aggregates the number of each letter by chapter
charsum <- chars %>% select(-word) %>% select(-paragraph) %>% group_by(chapter) %>% summarise_all(funs(s
head(charsum)
```

```
## # A tibble: 6 x 27
##   chapter     a     b     c     d     e     f     g     h     i     j
##     <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1       1  1965   386   606   753     0   434   596   997  1717    30
## 2       2   864   195   247   359     0   187   273   369   747    16
## 3       3  1059   203   292   367     0   227   340   459   781    18
## 4       4   660   133   203   242     1   163   200   317   565    11
## 5       5   875   167   251   285     0   198   250   353   737    16
## 6       6   533    93   116   207     0    97   170   248   384    10
## # ... with 16 more variables: k <int>, l <int>, m <int>, n <int>, o <int>,
## #   p <int>, q <int>, r <int>, s <int>, t <int>, u <int>, v <int>,
## #   w <int>, x <int>, y <int>, z <int>
```

```
#charsumtot takes the above charsum data frame and computes a total count for each letter for the entir
charsumtot <- charsum %>% select(letters) %>% summarise_all(funs(sum))
kable(charsumtot[,1:13], "latex", booktabs=T, caption="Letter Count, A-M") %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 1: Letter Count, A-M

a	b	c	d	e	f	g	h	i	j	k	l	m
23229	4543	5645	8731	4	4565	7647	10411	18657	487	2499	11276	4403

```
kable(charsumtot[,14:26], "latex", booktabs=T, caption="Letter Count, N-Z") %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 2: Letter Count, N-Z

n	o	p	q	r	s	t	u	v	w	x	y	z
18235	22102	4039	109	10093	14779	18018	8820	665	5934	100	6734	228

Just printing the first few lines of charsum and displaying charsumtot, we can see something rather unexpected. The total count for the letter "E" is 4 and not 0 as expected. As this is a direct contradiction to the claims of the author, we will need to verify with the source.

Referring back to the chars data frame, we can filter for the words containing one or more "E"s.

```
letterE <- chars %>%
  select(chapter, paragraph, word, e) %>%
  filter(e > 0)

kable(letterE, "latex", booktabs=T, caption="Words Containing the Letter E") %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 3: Words Containing the Letter E

chapter	paragraph	word	e
4	5	the	1
12	4	the	1
15	1	the	1
32	1	officers	1

Having identified the chapters, paragraphs and words containing the letter "E", we can verify their presence on Wikisource. Wikisource offers images of the actual pages of the book. The 4 instances we are interested in are displayed in the figures below.

cloudburst. Small plots sprang into public light which couldn't hold a poultry barn, to say nothing of a big City Hospital. But no grasping landlord can fool physicians in talking up a hospital location, so it was finally built, on high land, with a charming vista across Branton Hills' suburbs and distant hills; amongst which Gadsby's charity auto and bus trips took so many happy invalids on past hot days.

Now it is only fair that our boys and girls of this famous Organization of Youth, should walk forward for an introduction to you. So I will bring forth such bright and loyal girls as Doris Johnson, Dorothy Fitts, Lucy Donaldson, Marian Hopkins, Priscilla Standish, Abigail Worthington, Sarah Young, and Virginia Adams. Amongst the boys, cast a fond look upon Arthur Rankin, Frank Morgan, John Hamilton, Paul Johnson, Oscar Knott and William Snow; as smart a bunch of Youth as you could find in a month of Sundays.

As soon as our big hospital was built and functioning, Sarah Young and Priscilla Standish, in talking with groups of girls, had found a longing for a night-school, as so many folks had to work all day, so couldn't go to our Manual Training School. So Mayor Gadsby took it up with Branton Hills' School Board. Now school boards do not always think in harmony with Mayors and Councils; in

half lion and half woman; and a mountainous mass of masonry, built for a king's tomb. So, standing right in front of both, Nancy and Frank got that wondrous thrill coming from attaining a long, long wish. From Cairo to Italy, Spain, London, Paris, and that grand Atlantic sail, landing at Boston, and hustling by fast train (but *how* slow it did go!!) to Branton Hills! So, along about Thanksgiving Day, about half of its population was again at its big railway station, for Nancy was coming back. (And Frank, too, if anybody should ask you.)

And with that big Municipal Band a-booming and blaring, and the crowd of our old Organization girls pushing forward, did Branton Hills look good to Nancy? *And did Nancy look good to Branton Hills? What* a glorious tan, from days and days on shipboard! And was that old Atlantic ugly? Ask Frank, poor chap, who, as on that big Pacific, had found out just what a ship's rail is for! And that stomachs can turn most amazing flip-flops if an old boat is too frisky!

In just an instant, actual count, Nancy was in Lady Gadsby's arms, fighting valiantly to hold back a flood of big, happy sobs; and Frank was busy, grabbing a cloud of hands surging towards him.

XV

As THIS IS a history of a city I must not stay around any part too long. So, as it was almost "a small morning hour," Nina Adams, a widow, was sitting up; for Virginia, a High School girl, was still out; and, around two-thirty, was brought back in a fast car; two youths actually *dumping* an unconscious form on Nina's front porch, and dashing madly away. But Nina Adams saw it; and, calling for aid in carrying Virginia indoors, put in a frantic call for old Doc Wilkins, an old, long-ago school pal, who found Nina frantic from not knowing Virginia's condition, nor why the pair of youths shot madly away without calling anybody. But it only took Doctor Wilkins an instant to find out what was wrong; and Nina, noting his tight lips and growing scowl was in an agony of doubt.

"What *is* it, Tom? *Quick!!* I'm almost crazy!!"

Dr. Wilkins, standing by Virginia's couch, said, slowly:—

"It's nothing to worry about, Nina. Virginia will pull through all right, by morning."

But that didn't satisfy Nina Adams, *not for*

XXXII

A CROWD WAS standing around in City Park, for a baby was missing. Patrol cars roaring around Branton Hills; many a woman hunting around through sympathy; kidnapping rumors flying around. His Honor was out of town; but on landing at our railroad station, and finding patrol cars drawn up at City Park, saw, in that crowd's midst, a tiny girl, of about six, with a bunch of big shouting patrol officers, asking:—

"Who took that baby?"

"Did you do it?"

"Which way did it go?"

"How long ago did you miss it?"

"Say, kiddo!! *Why don't you talk?*"

An adult brain can stand a lot of such shouting, but a baby's is not in that class; so, totally dumb, and shaking with fright, this tot stood, thumb in mouth, and two big brown baby orbs just starting to grow moist, as His Honor, pushing in, said:

"Wait a bit!!" and that bunch in uniform, knowing him, got up and Gadsby sat down on a rock, saying:—

"You can't find out a *thing* from a young child by such hard, gruff ways. This tiny lady is

Image Links:

- https://en.wikisource.org/wiki/Page%3AGadsby.djvu/59
- https://en.wikisource.org/wiki/Page%3AGadsby.djvu/111
- https://en.wikisource.org/wiki/Page%3AGadsby.djvu/132
- https://en.wikisource.org/wiki/Page%3AGadsby.djvu/221

Unfortunately, the above reveals the claim Gadsby does not contain the letter "E" is false and it actually is present in four instances. While not as impressive as completely forgoing the letter "E", only using it four times is still remarkable. For context, the last sentence had more "E"s than the entire Gadsby book!

5.1.2 Letter Frequencies

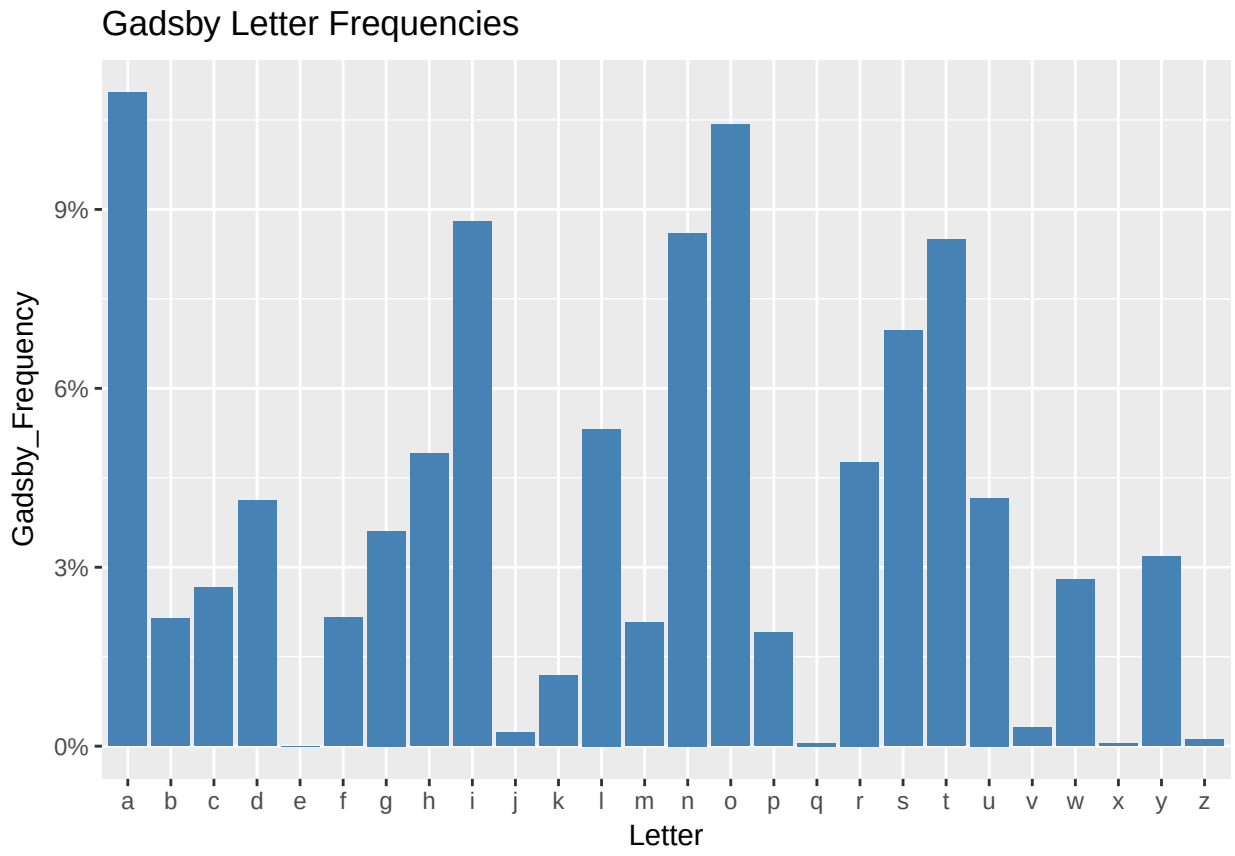
While we have discovered that the claim that Gadsby does not contain the letter "E" is false, there are interesting implications that arise from the almost non-existant presence of the letter "E". The first feature of the data we may be interested in is the frequencies of other letters. It is obvious that frequencies should increase as there are really only 25 letters with notable frequencies as opposed to the standard 26. However, the magnitude of increase in frequency will be dependent on how often a given letter appears with the letter "E" in the English language. For example, we may expect a lower increase or perhaps a decrease in the letter "H" due to the elimination of common words such as "he", "she", "the", etc. from the vocabulary available in Gadsby.

We can begin to gain an understanding of character frequencies by plotting the character totals computed earlier in a histogram.

```
charsumtot2 <- data.frame(Letter = colnames(charsumtot),
                          Gadsby_Count = t(charsumtot[1,]))

charsumtot2 <- charsumtot2 %>%
  mutate(Gadsby_Frequency = Gadsby_Count/sum(Gadsby_Count))

charsumtot2 %>% ggplot(aes(x=Letter, y=Gadsby_Frequency)) +
  geom_bar(stat="identity", fill="steelblue") +
  scale_y_continuous(labels=scales::percent) +
  ggtitle("Gadsby Letter Frequencies")
```



Unsurprisingly, eliminating "E" gives way for other vowels such as "A" and "O" to be the most frequent letters. On its own, this histogram has no meaning. We need to provide context by comparing to some other frequency distribution. A quick internet search offers many standard letter frequencies for the English language. We will use the dataset offered by Cornell (http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html). This data is based off of a sample of 40,000 words in the English language.

```
url <- "http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html"

English_Data <- read_html(url) %>% html_nodes("table") %>% html_table(header = TRUE)

English_Data <- as.data.frame(English_Data) %>%
  select(Letter, Count, Frequency) %>%
  rename(English_Count=Count, English_Frequency=Frequency) %>%
  mutate(Letter = tolower(Letter)) %>%
  mutate(English_Frequency = English_Frequency/100)

charsumtot2 <- charsumtot2 %>% inner_join(English_Data, by="Letter")
```

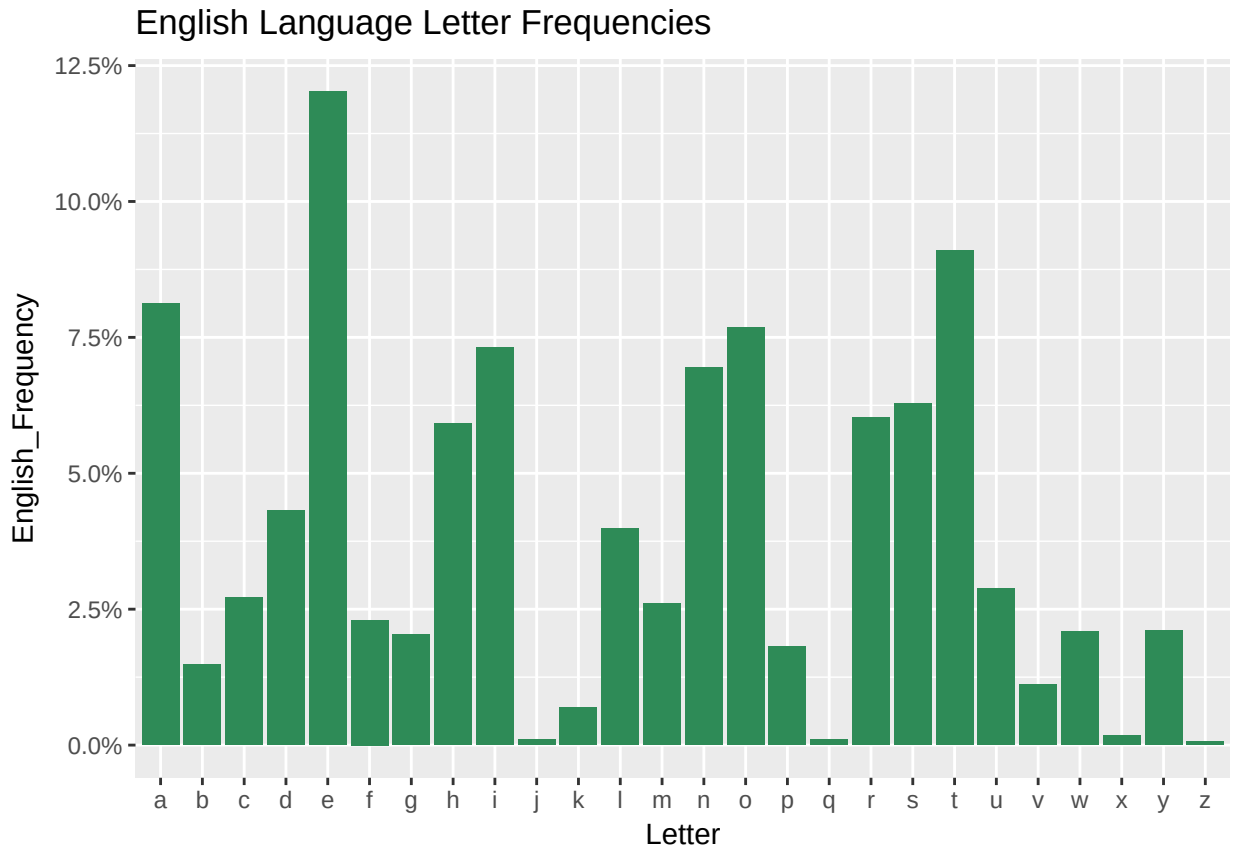
```
## Warning: Column `Letter` joining factor and character vector, coercing into
## character vector
```

```
head(charsumtot2)
```

```
##   Letter Gadsby_Count Gadsby_Frequency English_Count English_Frequency
## 1      a        23229     1.095951e-01         14810            0.0812
## 2      b         4543     2.143400e-02          2715            0.0149
## 3      c         5645     2.663326e-02          4943            0.0271
## 4      d         8731     4.119309e-02          7874            0.0432
```

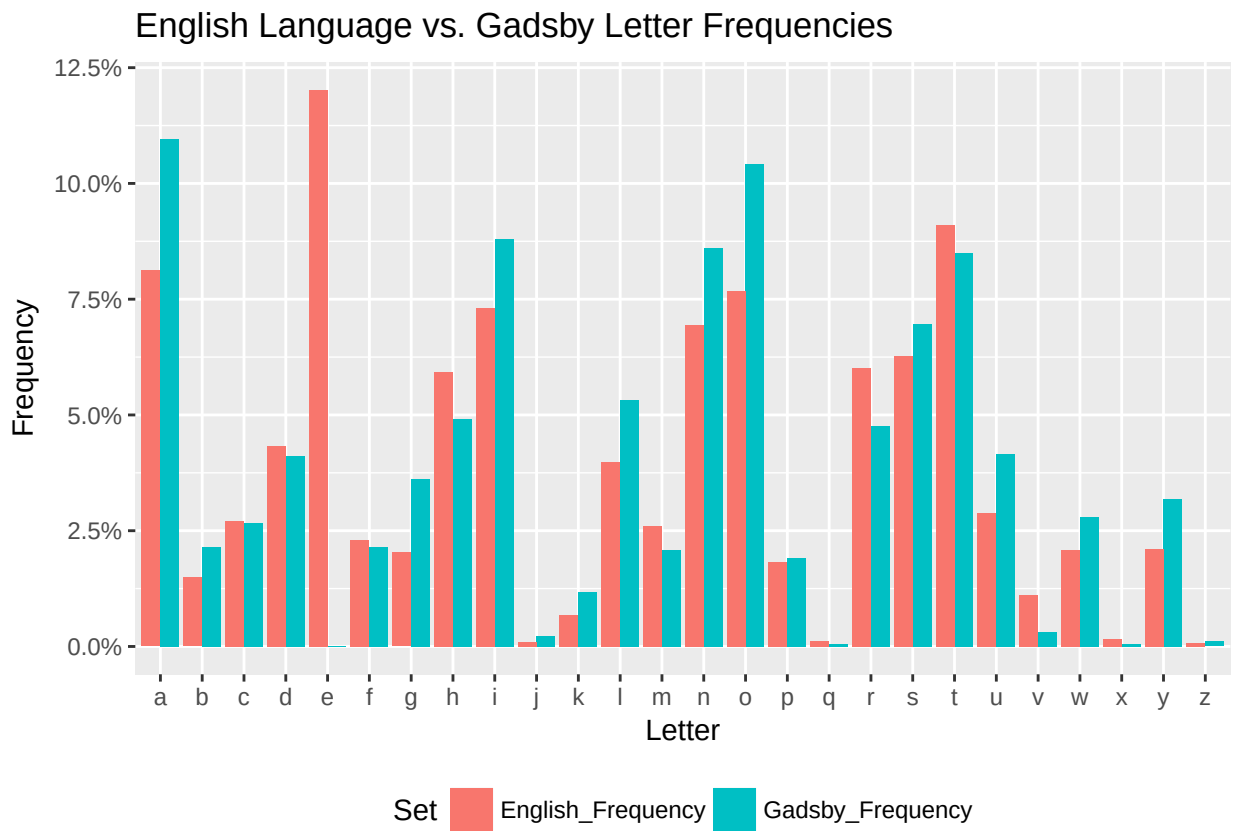
```
## 5      e            4     1.887211e-05         21912            0.1202
## 6      f         4565     2.153779e-02          4200            0.0230
```

```
charsumtot2 %>% ggplot(aes(x=Letter, y=English_Frequency)) +
  geom_bar(stat="identity", fill="sea green") +
  scale_y_continuous(labels=scales::percent) +
  ggtitle("English Language Letter Frequencies")
```

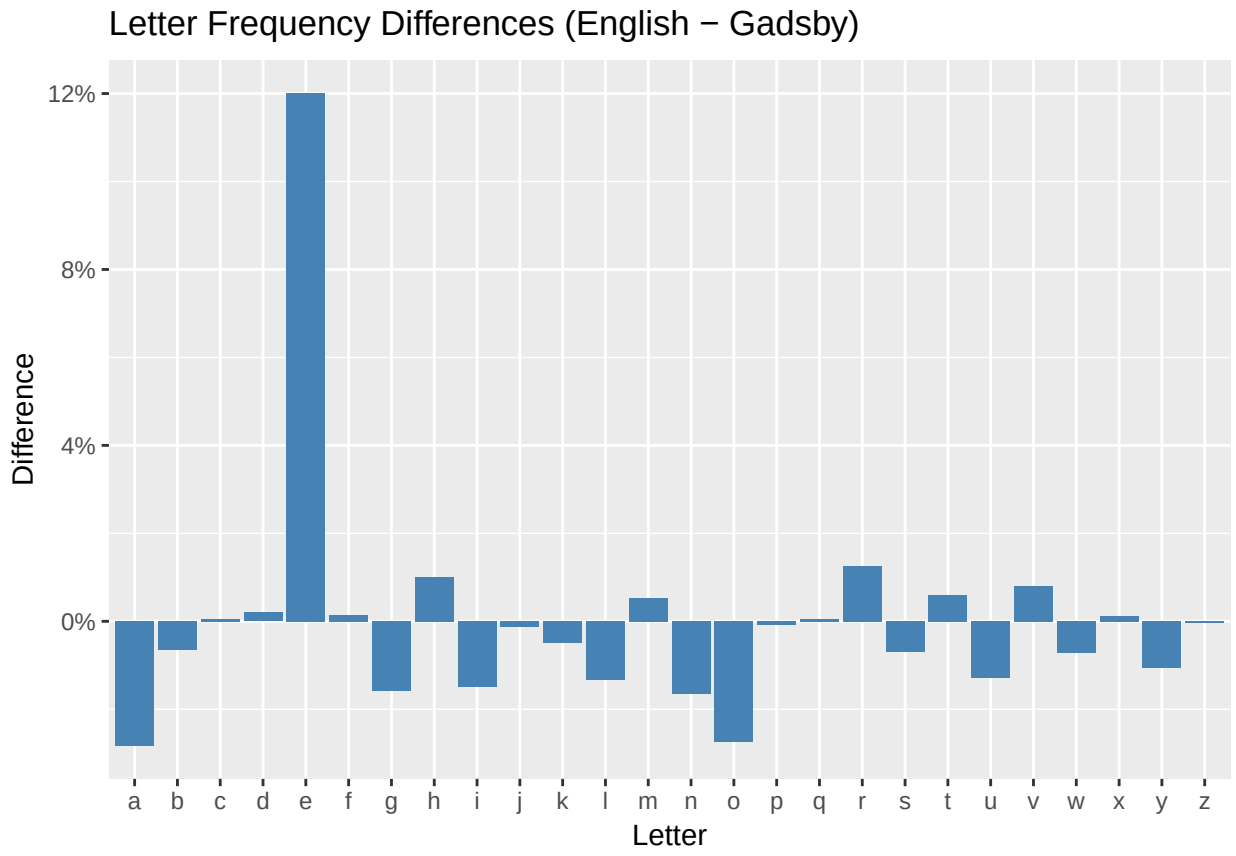


To facilitate easier comparison, we can plot the two distributions together:

```
charsumtot2 %>% select(Letter, Gadsby_Frequency, English_Frequency) %>%
  gather(Set, Frequency, Gadsby_Frequency:English_Frequency) %>%
  ggplot(aes(x=Letter, y=Frequency, fill=Set)) +
  geom_bar(stat="identity", position="dodge") +
  scale_y_continuous(labels=scales::percent) +
  ggtitle("English Language vs. Gadsby Letter Frequencies") +
  theme(legend.position="bottom")
```




```
charsumtot2 %>% mutate(Difference = English_Frequency - Gadsby_Frequency) %>%
  ggplot(aes(x=Letter, y=Difference)) +
  geom_bar(stat="identity", fill="steelblue") +
  scale_y_continuous(labels=scales::percent) +
  ggtitle("Letter Frequency Differences (English - Gadsby)")
```



The above comparison of letter frequency in the standard English language data set and the Gadsby dataset reveals a number of interesting effects that eliminating a key letter from the alphabet has on the frequency of the remaining letters. We can see from the English plot that "E" is the most common letter with a frequency of approximately 12%. Upon removing "E" in Gadsby, the 12% will be redistributed amongst the 25 other letters. However, not all letters experienced an increase in frequency with the elimination "E". In fact, many letters experienced a sharp decline in frequency. In the difference plot, we can see that aside from the obvious decline in the frequency of the letter "E", the letters "H", "M", "R", "T" and "V" experienced significant declines in frequency as well. Earlier we hypothesized that the letter "H" would decline due to the fact that many common words containing the letter "H" such as "he", "she" and "the" also contain "E" and thus are not part of the Gadsby vocabulary (ignoring the 4 exceptions identified earlier). Regarding the letter "R", many common words end with the suffix "-ER" or "-RE". Given this fact, we expect the frequency of "R" to be highly dependent on the presence of "E". The other letters that experienced reductions in frequency can be explained using similar logic.

Another interesting observation we can make is that the frequency of all other vowels, "A", "I", "O" and "U" increased. Also worth noting is that "A" and "O" experienced the largest increases across the alphabet. If we consider the structure of the English language, each word must contain at least one vowel. By eliminating the most common vowel, the others must experience an increase in ferquency to compensate.

5.1.3 Probabilistic Interpretation

This section is likely not within the scope of this exercise but is a thought I had while going through the character frequency section. For interest purposes I included it but it does not add much to the core objectives of the exercise.

We can consider the English language frequencies obtained above as the pmf of a discrete distribtution

representing the probability that a given letter is selected. Denote the probabilities as follows: $P(X = "a") = p_{Ea}$ where $p_{Ea} = 0.0812$ as shown in the table earlier. Note that the "E"" in the subscript represents that the value is from the English set and not Gadsby and the "a" in the subscript represents the letter in question. Then, the pmf for the character decomposition of a given word of length n can be described by a multinomial distribution since each letter can be thought of as an independent draw from the 26 letters with the probilities described above. The parameters are n, the length of the word, $x_a, ..., x_z$, the number of each letter present and $p_a, ...p_z$, the probability of each letter.

$$P(X_a = x_a, ..., X_z = x_z) \sim MVN(n, p_a, ...p_z)$$

This becomes interesting when we consider the second set of frequencies from Gadsby. Given the negligible frequency of "E"s, this set of frequencies can be treated as the pmf of the conditional distribution representing the probability that a given letter is selected given the letter is not "E". We can denote this as follows: $P(X = "a"|X \neq "e") = p_{Ga} = 0.1096$. We can use this conditional characteristic to develop a relationship between the two distributions:

$$p_{Ga} = P(X = "a"|X \neq "e") = \frac{P(X = "a" \cap X \neq "e")}{P(X \neq "e")} = \frac{P(X = "a" \cap X \neq "e")}{1 - P(X = "e")} = \frac{P(X = "a" \cap X \neq "e")}{1 - p_{Ee}}$$

While not directly related to our analysis of Gadsby, it is quite interesting to see the beginnings of a model that describes the effect "E" has on the frequencies of other letters. However, this becomes more complicated as we attempt to generate words using a model due to the quirks of the English language. For example, there are high levels of dependency between letters which will drastically change the probabilities of subsequent letters in a word. One noteworthy example is that "q" is almost always followed by "u". As a probability statement, $P(X_2 = "u"|X_1 = "q") \approx 1$. Another way of describing these relationships is to investigate the Index of Coincidence which measures the likelihood of drawing two letters one after the other in a document.

5.2 Words

We may now be interested in some of the features present at the word level. We've seen that the elimination of a letter from the alphabet causes a shift in the frequencies of other letters. We can expect a similar effect to occur in the word frequency analysis. This will be most apparant in the stop words section where many of the most common stop words will no longer be available such as "the" since they contain "E"s. Instead synonyms are likely to take their place.

5.2.1 Word Frequency

To frame this analysis, we first need to get a sense of the most frequent words in the book. Using Dplyr's count function, we can create a term count by chapter and for the book.

```
words2 <- words %>% count(chapter, word, sort=TRUE)
words3 <- words %>% count(word, sort=TRUE)
head(words2)
```

```
## # A tibble: 6 x 3
##   chapter  word     n
##     <int> <chr> <int>
## 1       1     a   246
## 2       3     a   134
## 3       1   and   131
## 4       1    of   126
## 5      13     a   123
```

```
## 6       2     a   108
```

```
head(words3)
```

```
## # A tibble: 6 x 2
##    word     n
##   <chr> <int>
## 1     a  2475
## 2   and  1687
## 3  that  1189
## 4    of  1168
## 5    in   949
## 6    to   924
```

Note that the most common words are all frequently used words in the English language. While we don't gain information on the use of obscure words used in Gadsby, we do have some insight on the strategy the author used in avoiding "E"s. For example, the word "a" (or "an" if the following word begins with a vowel) is an indefinite article that is used to reference something non-specific whereas "the" is the defintie equivalent. The fact that "a" is the most common word with almost 2500 instances suggests that the author structured sentences in a way that references objects indirectly. Other highly frequent words such as "and" are expected as they are the most common conjunctions. This is a feature that will be present in most books (unless they decide to avoid the letter "a")!

5.2.2 Stop Words

It was noted above that many of the most frequent words are simply common in the English language and thus do not provide too much insight in the unique wording used in Gadsby. One approach to overcoming this hurdle is to remove a list of stop words from our words dataset and only consider the remaining words. For reference, stop words are highly frequent words that typically do not add much value when analysing textual information. If anything, they hinder analysis as it is difficult to extract patterns in word usage if all documents present stop words as their most frequent words. As such we will remove them before proceeding in our analysis.

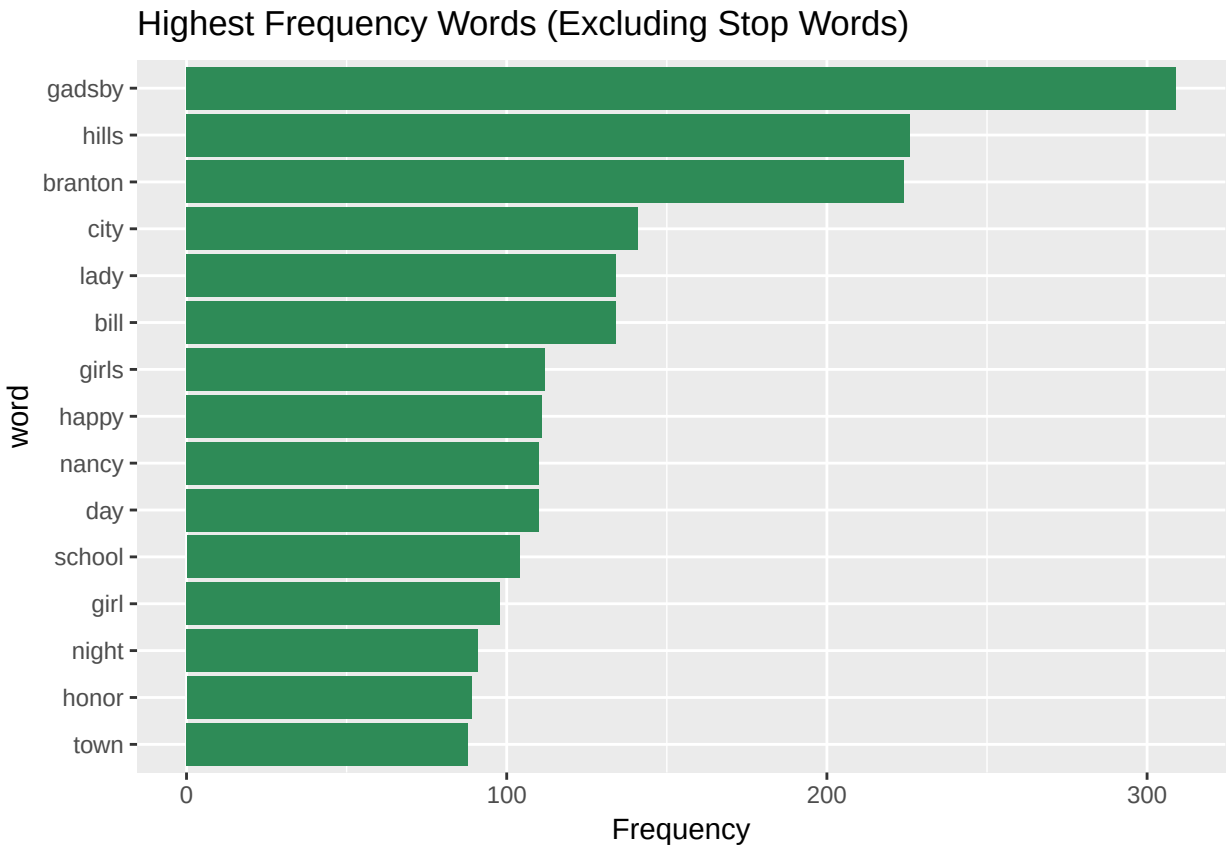
The tidytext package offers a dataset with three sets of stop words that can be used for this purpose.

```
data(stop_words)

words4 <- words3 %>% anti_join(stop_words, by="word")

words4 %>% top_n(15) %>% mutate(word = reorder(word,n)) %>%
  ggplot(aes(word, n)) +
  geom_col(fill="sea green") +
  coord_flip() +
  ylab("Frequency") +
  ggtitle("Highest Frequency Words (Excluding Stop Words)")
```

```
## Selecting by n
```



As expected, there are no longer and stop words and now we can see some more plot specific words. There are a few names present representing the main characters and we can see words reflecting the setting such as "branton" and "hills" (Branton Hills is the primary town where Gadsby takes place). Note that most of these words do not have context and thus it is challenging to find key patterns. We can examine bigrams to see if there is anything interesting there.

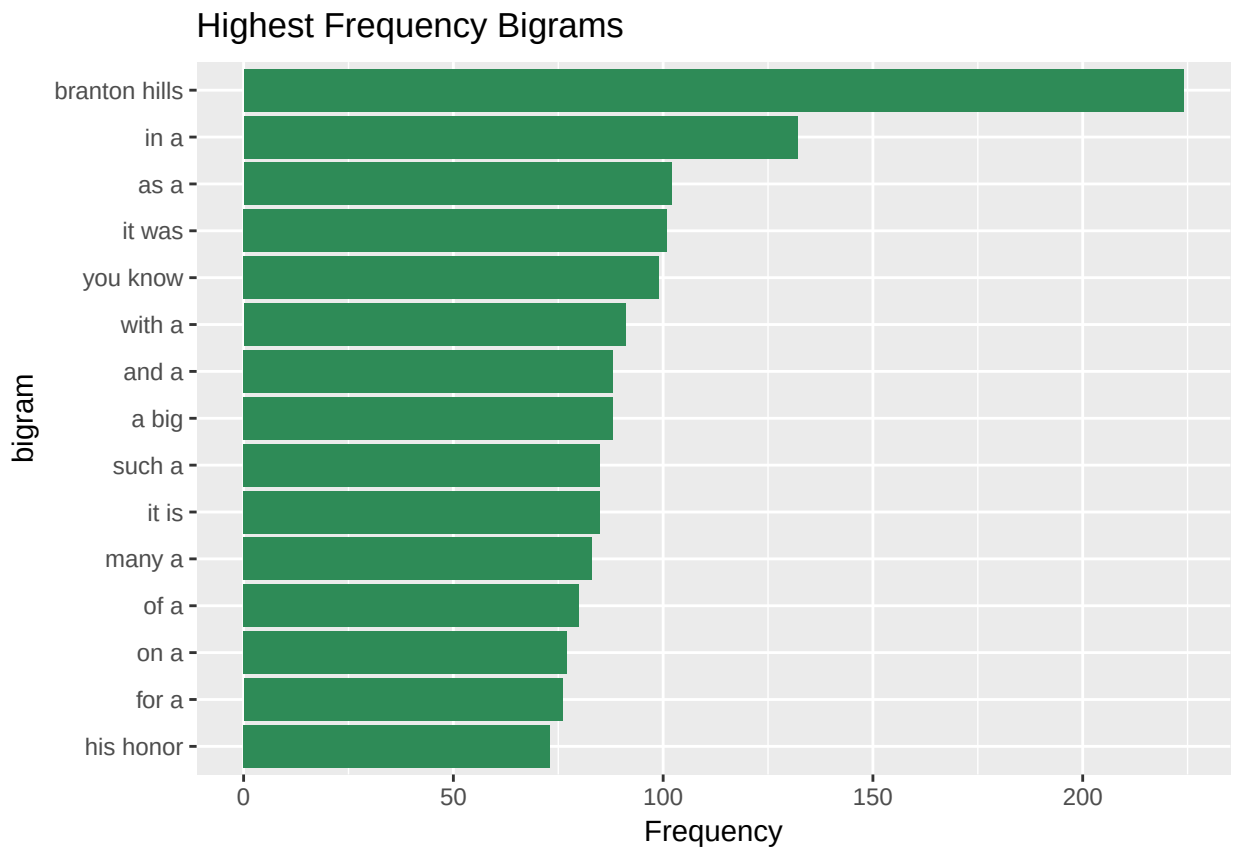
5.2.3 Bigrams

Bigrams are ordered pairs of words found in the text. They may be of interest as there may be some unusual pairings used by the author to circumvent using the letter "E". We will need to construct a new dataset with bigrams as tokens to perform this analysis.

```
bigrams <- chapters2 %>%
  unnest_tokens(bigram, text, token="ngrams", n=2) %>%
  count(bigram, sort=TRUE)

bigrams %>% top_n(15) %>%
  mutate(bigram = reorder(bigram,n)) %>%
  ggplot(aes(bigram, n)) +
  geom_col(fill="sea green") +
  coord_flip() +
  ylab("Frequency") +
  ggtitle("Highest Frequency Bigrams")
```

```
## Selecting by n
```

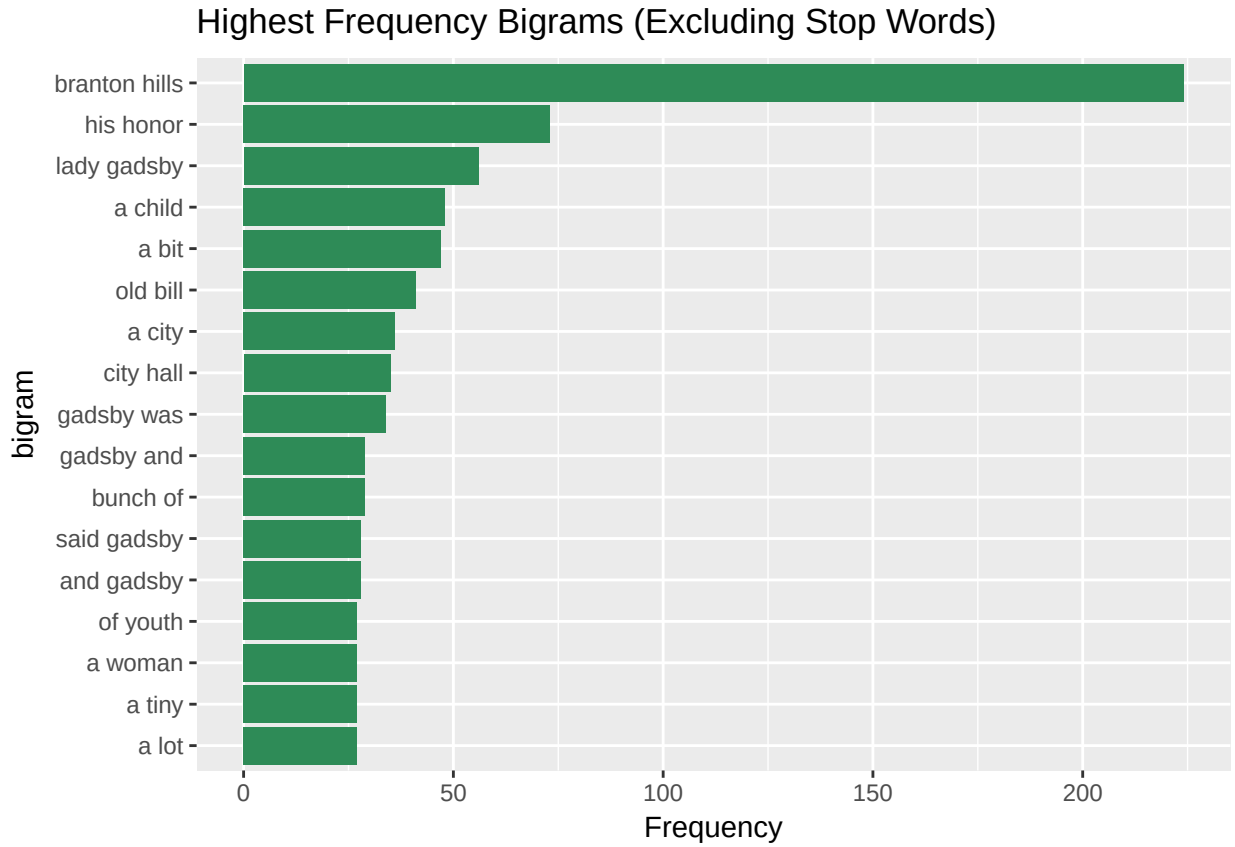


As mentioned above, "Branton Hills" is the main setting and thus we would expect it to be near the top of the bigram list. The other bigrams mostly contain stop words and in particular the word "a" which was discussed above.

```
bigrams2 <- bigrams %>%
  separate(bigram, c("word1", "word2"), sep=" ") %>%
  filter((!word1 %in% stop_words$word) | (!word2 %in% stop_words$word)) %>%
  unite(bigram, word1, word2, sep=" ")

bigrams2 %>% top_n(15) %>%
  mutate(bigram = reorder(bigram,n)) %>%
  ggplot(aes(bigram, n)) +
  geom_col(fill="sea green") +
  coord_flip() +
  ylab("Frequency") +
  ggtitle("Highest Frequency Bigrams (Excluding Stop Words)")
```

```
## Selecting by n
```



In removing bigrams where both words are stop words, we can see evidence of a possible writing strategy employed by the author. Note that outside of titles, and locations, many of the bigrams are references to objects using the word "a" such as "a woman" or "a child". It is possible that the author explicitly tried to reference objects indirectly to eliminate "the" altogether.

5.2.4 tf-idf

An alternative to the approach of removing all stop words is to use the term frequency - inverse document frequency statistic (tf-idf). The tf-idf statistic is a measure of how frequent a word is a document with its weight decreasing based on how frequent the word is across a range of documents. Thus words that are unique to a document have a much higher tf-idf as compared to something common such as a stop word. While this sounds attractive, there are a number of flaws with the tf-idf statistic in this particular exercise. First, we would require a corpus of similarly structured documents. In this case, we will need a number of books written in the early-mid 1900s to provide a comparable set. Second, the tf-idf statistic has a crucial flaw when applied to novels. Novels often contain fictional characters and locations that will not be present in any other novel. For example, the name Gadsby is unlikely to be present in any other novel collected for our corpus. As such, the unique names and locations will most likely have the highest tf-idf value. This does not add much value to the analysis as names and locations were already identified in our term frequency analysis above.

5.3 Sentiment Analysis

The final piece of analysis that we want to perform on the Gadsby text is basic sentiment analysis. Sentiment analysis is the idea of extracting intent the author is trying to convey. For example, there may be a positive or negative inclination in a news article. Fundamentally, sentiment analysis is the process of classifying tokens

in a document into pre-determined classes and extracting insights based on the classification results. These classes may be positive and negative or basic emotions such as happy or angry. The analysis is performed by separating the document into smaller components and computing sentiment scores by aggregating the sentiment scores of the tokens in each component. In the context of a novel, we can attempt to trace the plot through the novel by examining changes in the sentiment scores across chapters. To facilitate this task, the tidytext library has three sentiment datasets available. We will use the AFINN sentiments as they are numerical values that can be transformed into useful visualizations.

```
sent <- get_sentiments("afinn")

#Note that we use the mean of the sentiment scores and not the total to adjust for the
#differing sizes of chapters.
Gadsby_Sent <- words %>%
  inner_join(sent, by="word") %>%
  select(-paragraph) %>%
  group_by(chapter) %>%
  summarise(Sentiment = mean(score))

Gadsby_Sent %>% ggplot(aes(chapter, Sentiment)) +
  geom_bar(stat="identity", fill="sea green") +
  xlab("Chapter") +
  ggtitle("AFINN Sentiments by Chapter")
```

From the histogram, we can see that the book is mostly positive with a noticeable dip in sentiment in the chapters in the mid-late 20s. Perhaps this is the location of some of the struggles faced by Gadsby and the other characters. Based on plot summaries, we know that the book ends positively with Gadsby becoming mayor in the much imporoved Branton Hills which grew from a small town to a city. The positive ending is

clear in the histogram.

5.4 Lessons Learned

Hopefully the above analysis successfully highlights some of the interesting features of textual data that can be examined in R. One of the most important lessons to be taken away from this section is the importance of having a well structured dataset. Given how particular we were in building our words and characters datasets, analysis can be performed quite efficiently using Dplyr's data frame manipultion tools. In particular, using the mutate function and group_by/summarise functions, we have the ability to build interesting quantities into our data frames.

Another key takeaway is that it is important to have some sort of baseline distribution to compare to. Having the frequencies of each letter in Gadsby is great but there really was nothing too interesting until we compared to the standard English Lnguage distribution. Our interest was then in the difference between the two and allowed us to focus our analysis on the impact of removing "E" from the language. Off of this point, I think it is imporant to acknowledge that what we end up investigating may be different that what is originally intended. In this instance, I was not expecting such dramatic drops in frequency in any of the letters. Upon seeing this, I spent more time investigating the cause of this phenomenon.

6 Summary

In this exercise, we have demonstrated a complete text mining process from extraction through to analysis. We began by selecting the novel Gadsby as our subject of interest as the claim that there are no "E"s present in the novel offers an interesting starting point for analysis. As the novel is available in full on Wikisource, we extracted the text in full using tools from the rvest library. At first, extraction was attempted using the RCurl tools but due to the overwhelming amount of HTML clutter and the fact that we simply wanted all of the text, rvest offered a better solution. From there, we decomposed the data into a tidy format where each word acted as a token and was assigned a character count. At this point, the dataset was processed and ready for analysis.

In the analysis stage, we focused on three core areas. First we attempted to verify the claim that Gadsby does not contain the letter "E" but unfortunately found four instances of the letter "E". Regardless, we investigated the effects of a negligible frequency of "E"s on the other letters in the alphabet. We found that consonants in Gadsby either increased in frequency or decreased relative to the standard English language frequencies depending on the relationship the given letter has with the letter "E". For example, the letter "H" is commonly found in words where "E" is also present. Thus the frequency of "H" decreases in Gadsby. In contrast, all vowels increased in frequency which is in line with expectations as there must be a vowel in each word and thus removing "E" from the alphabet will naturally drive an increase in the frequency of other vowels to compensate. In investigating the word decomposition, we noted that the author seemed to refer to objects indirectly to circumvent the need for the word "the". This trend was present in both the term frequency and bigram frequency histograms. Other common words included key names and locations. The final area of interest was the changes in sentiment across the book. We were able to find a high-level sentiment value for each chapter and identify the positive and negative chapters.

Finally, we can take this exercise and consider the benefits and challenges that came with it to decide what would be useful for student learning. In addition, there are a few topics that may need to be covered further in depth to enable students to perform similar tasks in the future.

7 Text Mining Teaching Notes

Having gone through the text mining process from end to end, there are a few concepts that I think should be taught to students in advance of such a project.

- In the STAT courses I have taken to date, datasets tend to be preprocessed and clean numerical data. In a text mining exercise this will not be the case. The data will usually be categorical. I think it would be useful to take some time and discuss approach to working with categorical data, introduce techniques usch as one-hot encoding and to disucss areas of interest in textual information. In doing this, covering key statistics such as tf-idf could be useful. However, I do think it is imporant to cover the advantages and disadvantages of using such popular statistics. For example, I think there are quite a few shortcomings in tf-idf such as its focus on typos, names and locations that restrict its value. It also relies on having a corpus of related documents available.
- Sentiment analysis is an interesting topic that has a lot of value in tasks that involve opinionated pieces such as news articles but there are quite a few nuances that must be addressed. I think educating the students on the common hurdles such as negation in sentences can't assist them in coming up with more sophisticated approaches to sentiment analysis other than simply assigning scores to words. For example, a bigram approach can catch negations and prevent net cancellations in bigrams such as "not good". Another apporach could be to build a classifier that scores sentences at a time. There are fairly sophisticated text encoding models available that leverage classification techniques. I think that discussion could be interesting but I do not know if we can guarantee that students will have taken 441/841 or some equivalent.
- While not technically a statistics topic, I think it would be useful to have a brief discussion on linguistics to impart some domain knowledge on students. While it is possible for a statistician to perform analysis on any given dataset, interpreting results to extract useful insights requires some level of domain expertise. One point I think is particularly important is exploring the difference between natural and formal languages. Students should be aware of some of the challenges presented by natural languages such as the level of ambiguity, concept of literalness and the importance of sentence structure in interpreting a sentence. In contrast, formal languages are designed to convey a specific meaning with little or no ambiguity. This difference should play into the students' approach to parsing through text.

From the software side, it depends on the average students' level of proficiency in R. By the time students take this course, I would expect them to have some background in R but I find many students to be highly uncomfortable with R even in 400 level STAT courses. Regardless, I think a tutorial that briefly covers a few of the needed packages would be useful.

- RCurl and rvest: I think it is reasonable to expect that there will be students with no web scraping experience at this point. Exposing them to the range of tools available will be useful. There is little value in having them rebuild web scraping tools simply because they were unaware of what is available to them.
- Dplyr: Dplyr implements a number of methods that enable fluid data frame manipulation. This is quite different and much cleaner that working with other types of data structures in other languages. Since it is very intuitive, I don't think it needs to be explicitly taught but I do think it should be brought to the attention of the students.
- Regex: While we didn't use any regex processing in this exercise, I cannot deny that it is a powerful tool that is worth learning. I think that this is a topic that is worth covering in a bit more depth just to show the extent of text manipulation that regex can perform. As an aside, regex was briefly covered in CS 246 as part of bash scripting. We could go over it at a similra level of depth.

One of the trends I've noticed in STAT courses is that students tend to complete assignments by copying code out of the notes and repurposing it for the specific question. This approach may produce the correct results but the student loses out on the learning experience of building their solutions from the ground up. While there is limited time avaialbe for any course, I think it will be beneficial to allocate some time to directly exposing students to the tools available to them so they are more willing to come up with solutions independently. This will in turn reinforce conceptual learning.

8 Assignment and Project Thoughts

I think the objective in any text-mining assignment should be to have the students become comfortable with text based data manipulation and be comfortable deciding what is worth investigating and figuring out how to go about structuring their analysis in a logical way. In an assignment, this could be accomplished by using structured questions to guide students through some of the challenging steps such as data extraction and preprocessing. In terms of analysis, I think it would be best to relax the structure to allow better self-teaching. I think the core objective should be for the students to be able to come up with their own approach as opposed to reproducing one that has been given to them. That being said, I do think that any analysis questions should be asked in stages with explicit checkpoints so that it is still easy to mark. The motivation for this is that in many courses, R-based assignment questions can sometimes be answered by copying code out of notes. I don't think this is a great way to learn and so relaxing the structure to avoid direct copying should be beneficial to students.

Below are a few interesting dataset ideas that could accomplish these goals:

- Novel decomposition such as the exercise performed here. Gadsby is a really good choice as their is an immediately interesting quirk in the dataset to investigate.
- If there is a large group of acturial science students or math/finance students, considering a public stock and tracing sentiments in articles/press releases over a period of time could be very interesting. This could then be paired with the stock performance over the same period allowing the student to determine if there is a relationship between sentiments in articles/press releases and actual performance.
- Major world events or scandals offer interesting datasets with a wide variety of features to consider. We've discussed the Clinton e-mails and Panama/Paradise Papers as examples. The benefit of this exercise is that it tests the students' ability to perform sophisticated data extraction as the data is buried in the documents.

If this is intended as a project as opposed to an assignment, I think an open-ended project should be pursued. Much of the learning in this type of exercise comes through deciding what could be interesting and what information is neeed. This also opens up the door for more complicated data sources to facilitate more interesting analysis.