post02-jenny-yang

Jenny Yang

December 3, 2017

A Text to Analyze Text Analysis: Introduction to Working with Textual Data

Introduction

When someone thinks of statistics and computational techniques, they're more often than not imagining applications to quantitative, number-heavy fields. After all, stats is often grounded in mathematical theory, and we intuitively think that we can only use such techniques to analyze and manipulate numbers.

However, data analytics tools need not be limited to the realm of numbers; they can also be applied to texts as well. For example, suppose a company wants to get a general feel on how the public feels about a particular product, and wants a more comprehensive evaluation than just a number on a 1-5 scale. To get an overview, they could conduct text analysis on their product reviews in order to pick out major themes. Of course, if they wanted specific, detailed examples, they should directly read through the comments and reviews, but text analysis provides a guide that enables people to quickly understand the sentiments capured by large data sets.

The business world is often a blend of numbers and words. But can we apply the same techniques to purely literary data? The purpose of this post is to demonstrate that we can apply R and data visualization techniques to traditionally non-quantitative–in particular, literary–fields as well. I'll be walking you through some of the basic analysis and techniques that can be used to analyze any type of literature.

We'll use a package called `janeaustenr`. This package consists of her six published works, which were retrieved from Project Gutenberg, a platform that offers a variety of free eBooks. Each row corresponds to a physically printed line. In particular, we'll be using *Pride and Prejudice*, as it is her most well-known work (admittedly, I've also recently finished reading this novel for the third time, and I'm particularly partial to it.) Using these packages and techniques, we'll be able to visualize unstructured texts just as easily as we could visualize numbers in a data table.

As a brief summary, *Pride and Prejudice* offers a satirical take on 19th century society. Austen's protagonist, Elizabeth (Lizzy) Bennet, is a strong-willed, vivacious female with a sense of independence that is quite surprising for a woman of her time. Due to a combination of her familiy's financial situation and societal expectations, it is imperative that the five Bennet daughters marry well. *Pride and Prejudice* details the courtship of Elizabeth Bennet and Mr. Darcy, a rather proud and haughty (but fabulously wealthy) gentleman, and the complications that arise because of their conflicting personalities and clashing pride. Although the premise may sound rather dull, insipid, and shallow, *Pride and Prejudice* is often listed as one of the most-loved literary works, and its popularity is such that it transcends the test of time.

Let's now load the packages necessary to perform this analysis. Among the new packages, we'll need to install the new packages `janeaustenr`, `tidytext`, `tidyr`, and `wordcloud` in order to proceed. We'll also be using `dplyr` and `stringr` which we've covered in class, and `SnowballC`, which should be built into the package library depending on which version of R Studio you have. If you don't have it, go ahead and install it as well.

A note: many of the online resources surrounding tidytext uses pipe operators (%>%); while this is undoubtedly a powerful and convenient tool for `dplyr` and packages built around `dplyr`, for the purpose of this blog, I'll be using the conventional function form, just to offer another syntax option for those who might not yet be comfortable with pipe operators.

Install Packages:

```
#install packages
#install.packages('janeaustenr')
#install.packages('tidytext')
#install.packages("tm")

#load packages
library(janeaustenr)
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
```

```
library(tidytext)
library(stringr)
library(SnowballC)
library(ggplot2)
library(tidyr)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

Introduce our data set:

```
#Obtain Pride and Prejudice from janeaustenr
pp <- group_by(austen_books(), book)
pp <- filter(pp, book == "Pride & Prejudice")
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.1
```

```
head(pp, 10)
```

```
## # A tibble: 10 x 2
## # Groups:   book [1]
##                                                                    text
##                                                                   <chr>
##  1                                                 PRIDE AND PREJUDICE
##  2
##  3                                                       By Jane Austen
##  4
##  5
##  6
##  7                                                            Chapter 1
##  8
##  9
## 10 It is a truth universally acknowledged, that a single man in possession
## # ... with 1 more variables: book <fctr>
```

As you can see, the format of the file essentially reads like a novel. How do we convert it into a more conventional format so we can run an analysis?

In order to structure and clean the unstructured data into something we can work with, we'll need to transform the data set so that each row corresponds to a single word in the novel. Using the language of `tidyr`, we can "tokens" (a single word). The process of creating these tokens is called "tokenization."

Cleaning Data

```
#Tokenize our data
pp <- unnest_tokens(tbl = pp, output = word, input = text)
head(pp, 10)
```

```
## # A tibble: 10 x 2
## # Groups:   book [1]
##                 book      word
##               <fctr>     <chr>
##  1 Pride & Prejudice     pride
##  2 Pride & Prejudice       and
##  3 Pride & Prejudice prejudice
##  4 Pride & Prejudice        by
##  5 Pride & Prejudice      jane
##  6 Pride & Prejudice    austen
##  7 Pride & Prejudice   chapter
##  8 Pride & Prejudice         1
##  9 Pride & Prejudice        it
## 10 Pride & Prejudice        is
```

Our data has now been tokenized–each row represents a separate word in a format that we can analyze. Notice that we no longer have to deal with punctuation and that all words have been converted to lowercase, which will make our analysis easier (for instance, we won't have to worry about R thinking that "Pride" and "pride" are different.) However, note that words such as "and" and "it" are displayed in the table above. Unfortunately, words such as these, called "stop words," do not provide much interesting information, and yet are common in English language. Let's try removing them from `pp` so they don't clutter our analysis.

A large list of stop words are included in the tidytext data set `stop_words`; these words are pulled from three separate lexicons (dictionaries).

```
#preview stop words
head(stop_words, 10)
```

```
## # A tibble: 10 x 2
##           word lexicon
##          <chr>   <chr>
##  1           a   SMART
##  2         a's   SMART
##  3        able   SMART
##  4       about   SMART
##  5       above   SMART
##  6   according   SMART
##  7 accordingly   SMART
##  8      across   SMART
##  9    actually   SMART
## 10       after   SMART
```

To remove stop words, we'll use the `anti_join` function from the `dplyr` package.

```
pp <- anti_join(x = pp, y = stop_words, by = "word")
head(pp, 10)
```

```
## # A tibble: 10 x 2
## # Groups:   book [1]
##                 book         word
##               <fctr>        <chr>
##  1 Pride & Prejudice        pride
##  2 Pride & Prejudice    prejudice
##  3 Pride & Prejudice         jane
##  4 Pride & Prejudice       austen
##  5 Pride & Prejudice      chapter
##  6 Pride & Prejudice            1
##  7 Pride & Prejudice        truth
##  8 Pride & Prejudice  universally
##  9 Pride & Prejudice acknowledged
## 10 Pride & Prejudice       single
```

We no longer have irrelevant stop words such as “and” or “it” in our data set.

Now that our data has been formatted and cleaned, the real analysis can begin. To start, let’s take a look at the twenty most commonly words used in *Pride and Prejudice*.

```
#count each word and sort from most to least commonly used
count_words <- count(pp, word, sort = TRUE)

#take a look at the top twenty most commonly used words
head(count_words, 20)
```

```
## # A tibble: 20 x 3
## # Groups:   book [1]
##                 book      word     n
##               <fctr>     <chr> <int>
##  1 Pride & Prejudice elizabeth   597
##  2 Pride & Prejudice     darcy   373
##  3 Pride & Prejudice    bennet   294
##  4 Pride & Prejudice      miss   283
##  5 Pride & Prejudice      jane   264
##  6 Pride & Prejudice   bingley   257
##  7 Pride & Prejudice      time   203
##  8 Pride & Prejudice      lady   183
##  9 Pride & Prejudice    sister   180
## 10 Pride & Prejudice   wickham   162
## 11 Pride & Prejudice      dear   158
## 12 Pride & Prejudice   collins   156
## 13 Pride & Prejudice    family   151
## 14 Pride & Prejudice       day   140
## 15 Pride & Prejudice     lydia   133
## 16 Pride & Prejudice      hope   121
## 17 Pride & Prejudice    father   116
## 18 Pride & Prejudice    letter   116
## 19 Pride & Prejudice    mother   112
## 20 Pride & Prejudice catherine   110
```

Unsurprisingly, a majority of these words consists of character names. Words such as “lady” and “sister” are common as well, as those roles play a large part in Elizabeth’s society. We can assume that “letter” is within the top twenty because *Pride and Prejudice* is an epistolary novel, and Austen often uses letters to introduce major plot points.

Let’s also see how often Elizabeth’s two nicknames, Lizzy and Eliza, appear.

```
filter(count_words, word == "lizzy" | word == "eliza")
```

```
## # A tibble: 2 x 3
## # Groups:   book [1]
##                book  word     n
##              <fctr> <chr> <int>
## 1 Pride & Prejudice lizzy    95
## 2 Pride & Prejudice eliza    22
```

Considering that only her family calls her “Lizzy” and only one friend calls her “Eliza”, it’s unsurprising that these names appear much less often than “Elizabeth.”

Now let’s take a look at the ten least commonly used words:

```
tail(count_words, 10)
```

```
## # A tibble: 10 x 3
## # Groups:   book [1]
##                 book          word     n
##               <fctr>         <chr> <int>
##  1 Pride & Prejudice       womanly     1
##  2 Pride & Prejudice       women's     1
##  3 Pride & Prejudice worthlessness     1
##  4 Pride & Prejudice    wretchedly     1
##  5 Pride & Prejudice        writes     1
##  6 Pride & Prejudice         yards     1
##  7 Pride & Prejudice        yawned     1
##  8 Pride & Prejudice       yawning     1
##  9 Pride & Prejudice          york     1
## 10 Pride & Prejudice        youths     1
```

We see that “yawned” and “yawning” are listed in the last ten–however, these words capture the same meaning. In order to correct for this, let’s try **stemming** our words. Stemming is the process of changing words to their root form. For example, “yawned” and “yawning” would both become “yawn.” We’ll need the package `SnowballC`, which we loaded earlier, to do this.

```
#adds a column "stem" of stemmed words
pp <- mutate(pp, stem = wordStem(word, language = "english"))

#counts the number of occurences of each stemmed word
count_stem <- count(pp, stem, sort = TRUE)

#display least common stemmed words
tail(count_stem, 10)
```

```
## # A tibble: 10 x 3
## # Groups:   book [1]
##                 book      stem     n
##               <fctr>     <chr> <int>
##  1 Pride & Prejudice     wiser     1
##  2 Pride & Prejudice    wisher     1
##  3 Pride & Prejudice  withdraw     1
##  4 Pride & Prejudice withstood     1
##  5 Pride & Prejudice     witti     1
##  6 Pride & Prejudice    wittic     1
##  7 Pride & Prejudice      wive     1
##  8 Pride & Prejudice       woe     1
##  9 Pride & Prejudice      yard     1
## 10 Pride & Prejudice      york     1
```

“Yawned” and “yawning” are no longer in the least commonly used words! Let’s see how often the action of yawning appears now:

```
#number of times "yawn" appears as a stemmed word
filter(count_stem, stem == "yawn")
```

```
## # A tibble: 1 x 3
## # Groups:   book [1]
##                book  stem     n
##              <fctr> <chr> <int>
## 1 Pride & Prejudice  yawn     4
```

Is this accurate? Let’s look at how many times any tense of yawning appeared in the original:

```
#show instances in which any form of yawn was used
filter(pp, stem == "yawn")
```

```
## # A tibble: 4 x 3
## # Groups:   book [1]
##                book    word  stem
##              <fctr>   <chr> <chr>
## 1 Pride & Prejudice    yawn  yawn
## 2 Pride & Prejudice  yawned  yawn
## 3 Pride & Prejudice yawning  yawn
## 4 Pride & Prejudice    yawn  yawn
```

And yawning does indeed appear four times! We have therefore successfully stemmed and checked our data.

After stemming, has the top twenty list changed?

```
head(count_stem, 20)
```

```
## # A tibble: 20 x 3
## # Groups:   book [1]
##                 book      stem     n
##               <fctr>     <chr> <int>
##  1 Pride & Prejudice elizabeth   635
##  2 Pride & Prejudice     darci   417
##  3 Pride & Prejudice    bennet   333
##  4 Pride & Prejudice   bingley   311
##  5 Pride & Prejudice    sister   297
##  6 Pride & Prejudice      jane   292
##  7 Pride & Prejudice      miss   287
##  8 Pride & Prejudice      ladi   265
##  9 Pride & Prejudice      time   224
## 10 Pride & Prejudice   wickham   194
## 11 Pride & Prejudice    collin   180
## 12 Pride & Prejudice       day   174
## 13 Pride & Prejudice    friend   174
## 14 Pride & Prejudice     lydia   171
## 15 Pride & Prejudice      hope   168
## 16 Pride & Prejudice      feel   167
## 17 Pride & Prejudice      dear   161
## 18 Pride & Prejudice    famili   158
## 19 Pride & Prejudice     happi   155
## 20 Pride & Prejudice    manner   142
```

Elizabeth appears 38 more times than she did without stemming (the 38 additional times likely came from instances in which “Elizabeth’s” was used. However, some words, such as “darci” and “ladi”, have been stemmed incorrectly. This is an unfortunate byproduct of using our current cools. Cons of stemming include:

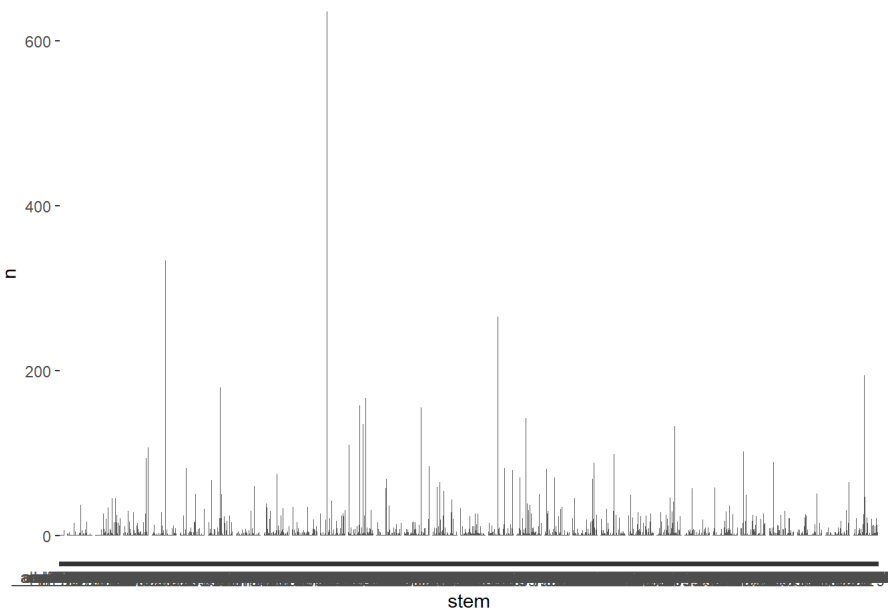
- stems are not always actual words (e.g., “ladi”, “famili”, “happi”)
- sometimes unrelated words are grouped together
- sometimes related words are not grouped together.

Nevertheless, stemming is undoubtedly a useful tool to aggregate our data and to eliminate a large portion of unnecessary repeated ideas.

Text Visualization

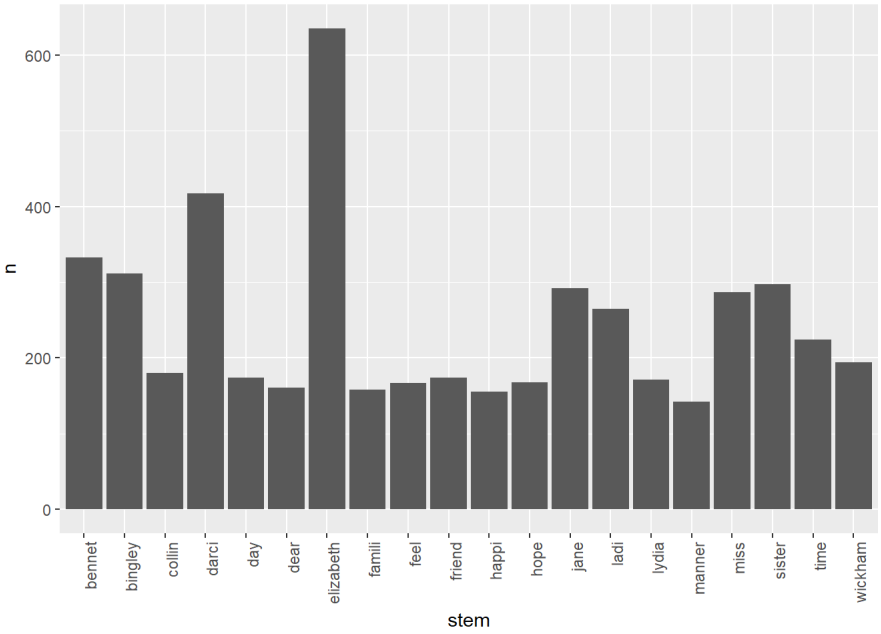
Let’s start with the basics–the simplest way we can visualize our data is with a simple bar chart, with the frequencies for each word. We’ll use `ggplot` to do this.

```
#plot occurrences of all words (may take awhile to run)
ggplot(count_stem, aes(x = stem, y = n)) + geom_col()
```



This is quite messy to read, so let’s just take a look at the twenty most commonly used words:

```
ggplot(head(count_stem, 20), aes(x = stem, y = n)) +
  geom_col() + theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



This is a visual representation of the top twenty most commonly used words.

So far, we've been examining inidviual words. What if we're interested in the *pairs* of words that Austen uses most commonly? Consecutive sequences of words are called "n-grams." We can see this simply by adding "token = ngrams" to unnest_tokens() when we first tokenize our text.

```
#obtain new data set to work with
pp_bigram <- group_by(austen_books(), book)
pp_bigram <- filter(pp_bigram, book == 'Pride & Prejudice')

#unnest_tokens to obtain pairs of words
pp_bigram <- unnest_tokens(tbl = pp_bigram, output = bigram,
                           input = text, token = "ngrams", n = 2)

#show the top twenty bigrams
head(pp_bigram, 20)
```

```
## # A tibble: 20 x 2
## # Groups:   book [1]
##                 book                  bigram
##               <fctr>                   <chr>
##  1 Pride & Prejudice               pride and
##  2 Pride & Prejudice            and prejudice
##  3 Pride & Prejudice             prejudice by
##  4 Pride & Prejudice                  by jane
##  5 Pride & Prejudice              jane austen
##  6 Pride & Prejudice           austen chapter
##  7 Pride & Prejudice                chapter 1
##  8 Pride & Prejudice                    1 it
##  9 Pride & Prejudice                  it is
## 10 Pride & Prejudice                   is a
## 11 Pride & Prejudice                a truth
## 12 Pride & Prejudice        truth universally
## 13 Pride & Prejudice universally acknowledged
## 14 Pride & Prejudice        acknowledged that
## 15 Pride & Prejudice                  that a
## 16 Pride & Prejudice               a single
## 17 Pride & Prejudice             single man
## 18 Pride & Prejudice                 man in
## 19 Pride & Prejudice           in possession
## 20 Pride & Prejudice           possession of
```

Note that these bigrams overlapp–i.e., "pride and" is one token, whereas "and prejudice" is another.

Let's perform the same analysis as above to analyze these bigrams. First, we'll count the number of occurences of each pair.

```
count_bigram <- count(pp_bigram, bigram, sort = TRUE)
head(count_bigram)
```

```
## # A tibble: 6 x 3
## # Groups:   book [1]
##                book bigram     n
##              <fctr>  <chr> <int>
## 1 Pride & Prejudice of the   464
## 2 Pride & Prejudice  to be   443
## 3 Pride & Prejudice in the   382
## 4 Pride & Prejudice   i am   302
## 5 Pride & Prejudice of her   260
## 6 Pride & Prejudice to the   252
```

Once again, the most common word pairs are quite uninteresting–not much meaning can be taken from pairs such as “of the” and “to be”. We’ll have to remove the stop words. To do this, we’ll separate the words into two columns by using the " " delimiter, and remove instances in which one of the two words is a stop word.

```
#separate pairs into two columns
count_bigram_sep <- separate(data = count_bigram, col = bigram,
                             into = c('word1', 'word2'), sep = " ")

#filter out stop words
count_bigram_sep <- filter(count_bigram_sep,
                       !word1 %in% stop_words$word &
                         !word2 %in% stop_words$word)

#show pairs of words that occur more than 10 times
filter(count_bigram_sep, n > 10)
```

```
## # A tibble: 20 x 4
## # Groups:   book [1]
##                  book     word1       word2     n
##                <fctr>     <chr>       <chr> <int>
##  1 Pride & Prejudice      lady   catherine   100
##  2 Pride & Prejudice      miss     bingley    72
##  3 Pride & Prejudice      miss      bennet    60
##  4 Pride & Prejudice       sir     william    38
##  5 Pride & Prejudice        de      bourgh    35
##  6 Pride & Prejudice      miss       darcy    34
##  7 Pride & Prejudice   colonel     forster    26
##  8 Pride & Prejudice   colonel fitzwilliam    25
##  9 Pride & Prejudice     cried   elizabeth    24
## 10 Pride & Prejudice      miss       lucas    23
## 11 Pride & Prejudice      miss          de    20
## 12 Pride & Prejudice  thousand      pounds    20
## 13 Pride & Prejudice      lady       lucas    18
## 14 Pride & Prejudice   replied   elizabeth    18
## 15 Pride & Prejudice      lady catherine's    16
## 16 Pride & Prejudice      dear       lizzy    15
## 17 Pride & Prejudice      miss   bingley's    15
## 18 Pride & Prejudice catherine          de    14
## 19 Pride & Prejudice      miss   elizabeth    12
## 20 Pride & Prejudice       ten    thousand    11
```

The most common pairs of words are character titles followed by character names–for example, “Lady Catherine” and “Miss Bingley.” We also see that Elizabeth does a lot of exclaiming and talking, based off of “cried Elizabeth” and “replied Elizabeth.” Finally, note that “thousand pounds” and “ten thousand” appear relatively frequently. This refers to Darcy’s wealth (he would’ve been the equivalent of a millionaire today), and considering how imperative it was that women married well, it’s unsurprising that his wealth is a large factor and mentioned quite a bit in character interactions.

How do we combine the two filtered columns into a single column, so we have the original format but without stop words? We simply use the unite() function.

```
#combine two word columns into one column called "bigram"
count_bigram <- unite(count_bigram_sep, col = bigram, word1, word2, sep = " ")

#preview table to confirm
head(count_bigram)
```

```
## # A tibble: 6 x 3
## # Groups:   book [1]
##                book         bigram     n
##              <fctr>          <chr> <int>
## 1 Pride & Prejudice lady catherine   100
## 2 Pride & Prejudice   miss bingley    72
## 3 Pride & Prejudice    miss bennet    60
## 4 Pride & Prejudice    sir william    38
## 5 Pride & Prejudice      de bourgh    35
## 6 Pride & Prejudice     miss darcy    34
```

Now that we’ve gone over the basic fundamentals of text visualization, let’s move onto more advanced methods.

Sentiment Analysis

Using sentiment analysis, we can classify different sections of the text by their emotional content. For example, if a section contains many instances of “happy,” “joy,” or “delight,”, we assume that something good has happened in the plot. However, sometimes these texts can be misclassified–for example, if the context is “Elizabeth was not happy”, the word “happy” might cause us to misclassify this as a happy moment. Despite this, sentiment analysis is a good way to get a broad sense of key plot points.

Let’s take a look at the `sentiments` data set in the `tidytext` library, which consists of words and their associated sentiments.

```
head(sentiments, 10)
```

```
## # A tibble: 10 x 4
##           word sentiment lexicon score
##          <chr>     <chr>   <chr> <int>
##  1      abacus     trust     nrc    NA
##  2     abandon      fear     nrc    NA
##  3     abandon  negative     nrc    NA
##  4     abandon   sadness     nrc    NA
##  5   abandoned     anger     nrc    NA
##  6   abandoned      fear     nrc    NA
##  7   abandoned  negative     nrc    NA
##  8   abandoned   sadness     nrc    NA
##  9 abandonment     anger     nrc    NA
## 10 abandonment      fear     nrc    NA
```

What does the column “lexicon” mean here, and how many different types of lexicons are there?

```
unique(sentiments$lexicon)
```

```
## [1] "nrc"      "bing"     "AFINN"    "loughran"
```

The nrc lexicon simply tells us whether or not a word is categorized by a particular sentiment (i.e., trust, fear, etc.) The bing lexicon categorizes words into positive or negative connotations, the AFINN lexicon assigns a word with a score on a scale from -5 to 5, with 5 indicating a postitive conotation. The loughran lexicon extends sentiment categories to words such as “litigation” and “uncertainty”, which may be used more commonly in a financial setting, so we won’t concern ourselves with it here.

What are the different sentiment classifications are there in this data set?

```
#obtains the sentiments in the nrc lexicon
unique(filter(sentiments, lexicon == "nrc")$sentiment)
```

```
##  [1] "trust"        "fear"         "negative"     "sadness"
##  [5] "anger"        "surprise"     "positive"     "disgust"
##  [9] "joy"          "anticipation"
```

Because large chunks of text, such as a page, can contain several emotions, it’s better to deal with smaller chunks. We’ll just focus on analyzing single words.

Remember, we’ve already created a dataset “pp” that contains the words in *Pride and Prejudice* and their stems.

```
head(pp)
```

```
## # A tibble: 6 x 3
## # Groups:   book [1]
##                book      word     stem
##              <fctr>     <chr>    <chr>
## 1 Pride & Prejudice     pride    pride
## 2 Pride & Prejudice prejudice prejudic
## 3 Pride & Prejudice      jane     jane
## 4 Pride & Prejudice    austen   austen
## 5 Pride & Prejudice   chapter  chapter
## 6 Pride & Prejudice         1        1
```

What are the most common words associated with a positive feeling in *Pride and Prejudice*?

```
#obtain positive words in sentiments data set
positive <- filter(sentiments, sentiment == "positive" & lexicon == "nrc")

#match positive words in Pride and Prejudice with positive words in sentiments
positive_pp <- inner_join(pp, positive, by = "word")
positive_pp
```

```
## # A tibble: 6,789 x 6
## # Groups:   book [?]
##                 book       word    stem sentiment lexicon score
##               <fctr>      <chr>   <chr>     <chr>   <chr> <int>
##  1 Pride & Prejudice      pride   pride  positive     nrc    NA
##  2 Pride & Prejudice      truth   truth  positive     nrc    NA
##  3 Pride & Prejudice    fortune  fortun  positive     nrc    NA
##  4 Pride & Prejudice      truth   truth  positive     nrc    NA
##  5 Pride & Prejudice   rightful   right  positive     nrc    NA
##  6 Pride & Prejudice       dear    dear  positive     nrc    NA
##  7 Pride & Prejudice invitation   invit  positive     nrc    NA
##  8 Pride & Prejudice       dear    dear  positive     nrc    NA
##  9 Pride & Prejudice    fortune  fortun  positive     nrc    NA
## 10 Pride & Prejudice  delighted delight  positive     nrc    NA
## # ... with 6,779 more rows
```

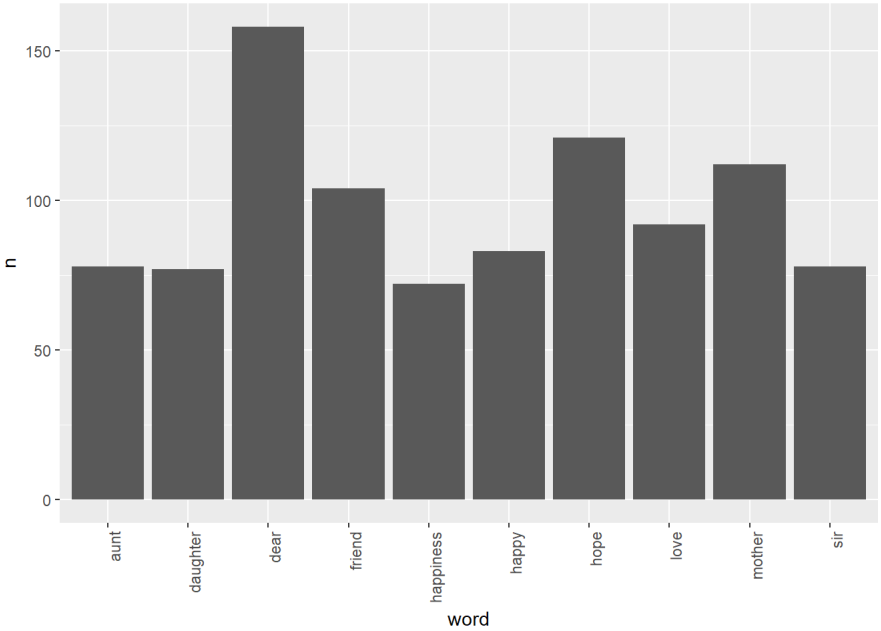
```
#count the number of instances a word appears
count_positive_pp <- head(count(positive_pp, word, sort = TRUE), 10)

head(count_positive_pp)
```

```
## # A tibble: 6 x 3
## # Groups:   book [1]
##                 book   word     n
##               <fctr>  <chr> <int>
## 1 Pride & Prejudice   dear   158
## 2 Pride & Prejudice   hope   121
## 3 Pride & Prejudice mother   112
## 4 Pride & Prejudice friend   104
## 5 Pride & Prejudice   love    92
## 6 Pride & Prejudice  happy    83
```

These are the most common words associated with a “positive” sentiment in *Pride and Prejudice*.

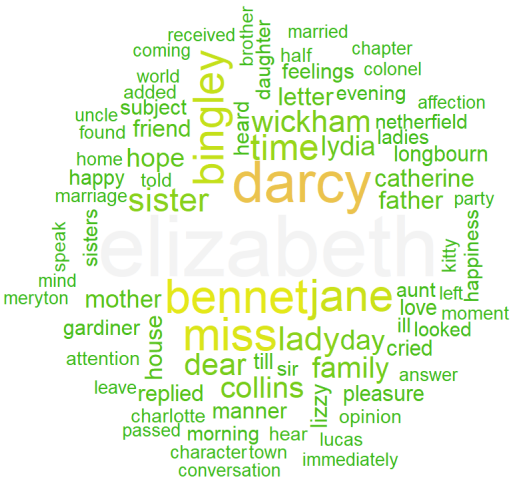
Let’s plot them to make them more visually understandable.

```
ggplot(head(count_positive_pp, 20), aes(x = word, y = n)) + geom_col() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



So far we’ve just done simple bar plots for all the analysis we’ve done. While standard, let’s try to make our visuals a bit more exciting by making a word cloud.

```
#create word cloud with the top 100 words
wordcloud(words = count_words$word, freq = count_words$n, max.words = 80, col = terrain.colors(length(count_words$
word), alpha = 0.9), random.order = FALSE)
```



Conclusion

Here, I've only presented the basic principles of working with tidyr, cleaning textual data, and visualizing data. To summarize, we first introduced and our data set. We then had to reformat it and remove stop words to make our analysis more meaningful. We created some basic plots, before moving on to sentiment analysis and creating more visually exciting plots.

Although we've only scratched the surface, there's so much room for more exploration using text analysis. For example, using machine learning, if we were given a paragraph from a random source, we might be able predict which author wrote that passage. Text analysis is still a growing and developing field, and there's still more work to be done and explored.

References

Pride and Prejudice Wikipedia

Text Mining in R: A Tutorial by Shubham Simar Tomar

Cleaning Words with R: Stemming, Lemmatization & Replacing with More Common Synonym

Text Mining with R by Julia Silge and David Robinson

Data Manipulation with tidyr by Teja Kodali

tidytext: Word Clouds and Sentiment Analysis in R by Michael Grogan

Intro to Text Analysis with R by Tal Galili