Halloween Books Book

Justin Minsk   Andrew Innes   Jerrin Varghese

Contents

Chapter 1

Poe

Edgar Allen Poe is a well known short story and poem writer that tended to write poems in the horror genre such as ”The Raven”, ”The Cask of Amontillado”, ”The Tell-Tale Heart”, and many other horror fan’s favorites. The purpose of this paper is to look at the book, ”The Complete Poetical Works Of Edgar Allen Poe” and find the most used words in his poetical works, just in time for Halloween.

1.1 ”The Complete Poetical Works Of Edgar Allen Poe”

The first task is to get ”The Complete Poetical Works Of Edgar Allen Poe” into R to be able to mine the text. First we need the libraries that we will use to mine the text.

```
library(tidytext)
library(tm)
library(wordcloud)
library(stringr)
library(dplyr)
library(knitr)
library(gutenbergr)
```

Then we use gutenbergr to extract the data into a data frame.

```
ids <- gutenberg_works(author == str_extract(author, "Poe, Edgar Allan"))
#get Poe work's IDs

head(ids$title)

## [1] "The Fall of the House of Usher"
## [2] "First Project Gutenberg Collection of Edgar Allan Poe"
## [3] "The Cask of Amontillado"
## [4] "The Masque of the Red Death"
## [5] "The Raven"
## [6] "The Works of Edgar Allan Poe <U+0097> Volume 1"

#see Poe work's IDs

df <- gutenberg_download(10031)
#get the complete poetical works of Edgar Allan Poe
```

The data frame now looks like this:

```
words_df <- df%>%
  unnest_tokens(word, text)
#split the lines into words

head(words_df)

## # A tibble: 6 x 2
##   gutenberg_id     word
##          <int>    <chr>
## 1        10031      the
## 2        10031 complete
## 3        10031 poetical
## 4        10031    works
## 5        10031       of
## 6        10031    edgar
```

Then we need to get rid of all the common words that do not give us anything unique in the text compared to other texts. We do this by using a list of words that are commonly used. This text is unique since it is written in an older English and has some extra common words that need to be filtered out past the normal list.

```
words_df <- words_df%>%
  filter(!(word %in% stop_words$word))
#get rid of common words that are not unique (the, a, etc.)
```

```
words_df <- words_df%>%
  filter(!word == "thy" & !word == "thou" & !word == "thee")
#some older words not in out stop_word list that are not useful
```

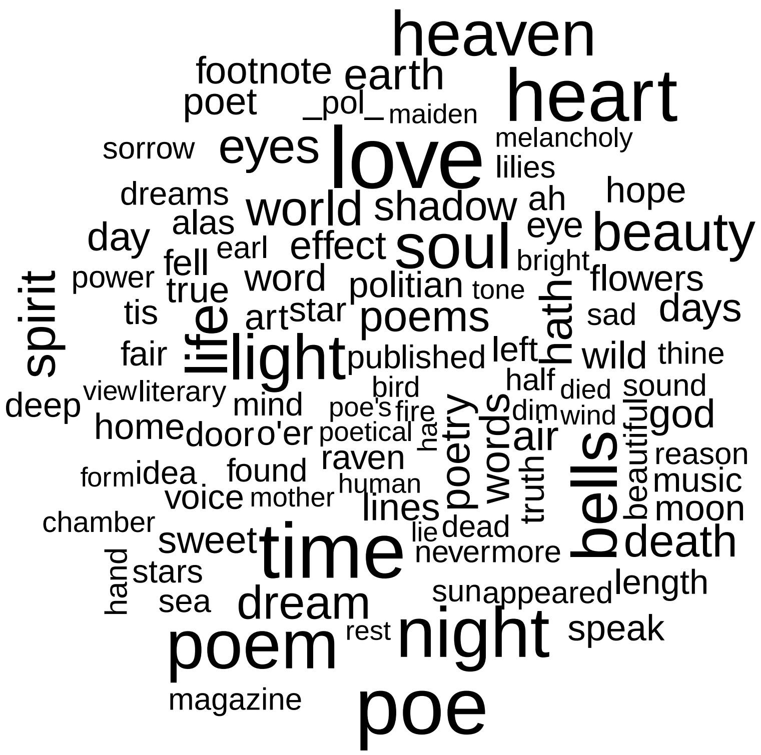
Now to create a count of each instance of a word. Using simple dplyr functions this can easily be done.

```
words_free <- words_df%>%
  group_by(word)%>%
  summarise(count = n())%>%
  arrange(-count)
#make a count of the word

head(words_free)

## # A tibble: 6 x 2
##    word count
##   <chr> <int>
## 1  love   116
## 2   poe   104
## 3  time   101
## 4 heart    96
## 5 night    90
## 6  poem    89
```

1.3 Wordcloud

Now we can create a wordcloud, a good visual representation of the most common words. The bigger the word the higher the frequency. We will use the library wordcloud to generate this image.

```
wordcloud(words_free$word, words_free$count, min.freq = 25)
```



```
#make a word cloud
```

Chapter 2

The Shunned House

In this article we construct a wordcloud, using the tidytext R package, for H.P. Lovecraft's The Shunned House.

The Shunned House is a horror fiction novelette by American author H.P. Lovecraft, published in 1937[1]. Below we craft a wordcloud for the most common words appearing in the novelette.

2.1 The Gutenberg Package

The Gutenberg Package is a package for R, gutenbergr, that gives one access to the books in Project Gutenberg. One has to first install this package and bring it in with library. You may then call the following function and store the result. Since we will be using The Shunned House we will download it using its unique integer identifier. In order to do this we must execute the following code:

```
library(gutenbergr)
library(stringr)
gutenberg_works(str_detect(title,'The Shunned House'))

## # A tibble: 1 x 8
##   gutenberg_id             title                                 author
##          <int>             <chr>                                  <chr>
## 1        31469 The Shunned House Lovecraft, H. P. (Howard Phillips)
## # ... with 5 more variables: gutenberg_author_id <int>, language <chr>,
## #   gutenberg_bookshelf <chr>, rights <chr>, has_text <lgl>

House<-gutenberg_download(31469)

House
```

[1]The novel was published in the October 1937 issue of Weird Tales.

```
## # A tibble: 1,065 x 2
##    gutenberg_id
##           <int>
##  1        31469
##  2        31469
##  3        31469
##  4        31469
##  5        31469
##  6        31469
##  7        31469
##  8        31469
##  9        31469
## 10        31469
## # ... with 1,055 more rows, and 1 more variables: text <chr>
```

This dataframe has two columns, one for the The ID Number of the book, and one containing the text from the book. Now we are ready for very litte data cleaning.

2.2 Very Little Data Cleaning

We would like to remove the front matter of the book. By inspection, we have determined that the front matter ends on line 6. Therefore we can redefine House to begin on line 7:

```
library(dplyr)
House<-House[7:1055,]
```

2.3 The Wordcloud

To make the wordcloud, we first have to break up the lines into words. We can use a function from the tidytext package for this:

```
library(tidytext)
words_df<-House%>%
  unnest_tokens(word,text)

words_df

## # A tibble: 10,968 x 2
##    gutenberg_id      word
##           <int>     <chr>
##  1        31469        _a
## 2          31469 posthumous
## 3          31469      story
## 4          31469         of
## 5          31469    immense
## 6          31469      power
## 7          31469    written
## 8          31469         by
## 9          31469          a
## 10         31469     master
## # ... with 10,958 more rows
```

We can remove the common, unimportant words with the stop_words data frame and some dplyr:

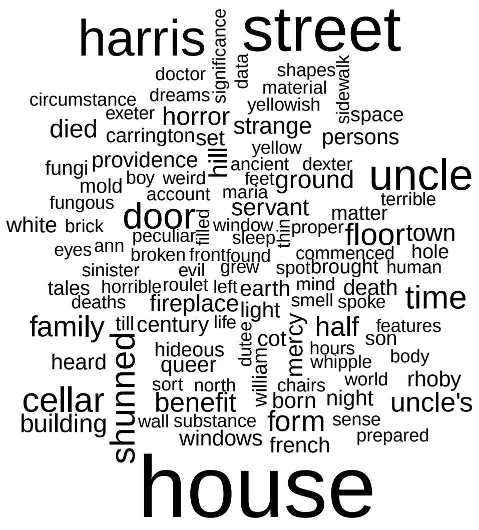
```
words_df<-words_df%>%
  filter(!(word %in% stop_words$word))

words_df

## # A tibble: 4,547 x 2
##    gutenberg_id       word
##           <int>      <chr>
##  1        31469         _a
##  2        31469 posthumous
##  3        31469      story
##  4        31469    immense
##  5        31469      power
##  6        31469    written
##  7        31469     master
##  8        31469      weird
##  9        31469    fiction
## 10        31469       tale
## # ... with 4,537 more rows
```

Now, we need to calculate the frequencies of the words in the novelette. Again, we can use standard dplyr techniques for this:

```
word_freq<-words_df%>%
  group_by(word)%>%
  summarize(count=n())

word_freq

## # A tibble: 2,652 x 2
##           word count
##          <chr> <int>
## 1            _a     1
## 2      _cellar_     1
## 3       _daily      1
## 4      _elbow_      1
## 5  _emanation_     1
## 6      _gaspee_     1
## 7         _had_     1
## 8           _in     1
## 9     _jacques      1
## 10 _providence      1
## # ... with 2,642 more rows
```

Finally, it's time to generate the wordcloud:

```
library(wordcloud)
wordcloud(word_freq$word,word_freq$count,min.freq = 5)
```



Chapter 3

Romeo And Juliet

In this article we are constructing a wordcloud using the tidytext R package. Here we will be taking the words from the famous book Romeo and Juliets written by shakespear. We will extract all the words and convert it to a cloud of words in different shape and colurs using the package wordcloud.

3.1 Romeo and Juliet

is a tragic love story written by William Shakespeare early in his career.

3.2 The Gutenberg Package

This is a relatively new package for R,Gutenberg , that gives one access to all of the novels written by different authors.

```
library(janeaustenr)
library(dplyr)
library(tidytext)
library(gutenbergr)
library(wordcloud)
library(ggplot2)
library(stringr)
library(knitr)
```

Let's find the book for Romeo and Juliets by shakespear using gutenberg.

```
gutenberg_works(str_detect(title,'Romeo'))

## # A tibble: 3 x 8
##   gutenberg_id
##          <int>
## 1         1513
## 2        22274
## 3        47960
## # ... with 7 more variables: title <chr>, author <chr>,
## #   gutenberg_author_id <int>, language <chr>, gutenberg_bookshelf <chr>,
## #   rights <chr>, has_text <lgl>
```

Let's now download and store it into a data frame.

```
  Romeo<-gutenberg_download(1513)
```

3.3 Some Data Cleaning

Adding a new column of line to get the line numbers and clean the id to Null.

```
Romeo$line<-1:5268
Romeo$gutenberg_id<-NULL
```

3.4 The wordcloud

To make the word cloud, we first need to break up the lines into words. We can use the function from package tidytext for this.

```
  Romeo_words<-Romeo%>%
  unnest_tokens(word,text)
```

We can remove common, unimportant words with the stop_words from the data frame with dplyr.

```
  Romeo_words<-Romeo_words%>%
  filter(!(word %in% stop_words$word))
```

Since this is a tragic love story, so lets only take the joy and sadness word sentiments out of it.

```
nrc<-get_sentiments('nrc')
joy_sad<-nrc%>%
filter(sentiment == 'joy' | sentiment == 'sadness')
```

Lets takeout all the joy and sadness words to a dataframe.

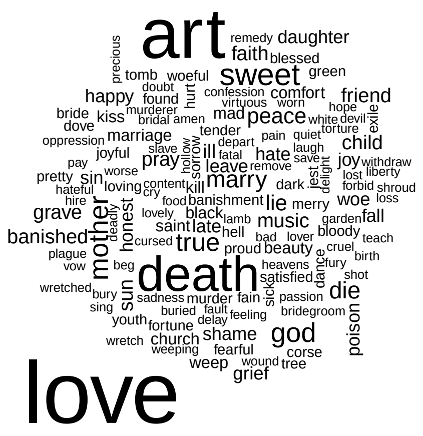
```
  Romeo_joy_sad<-inner_join(Romeo_words,joy_sad)
```

Now we need to calculate frequencies of the words in the novel. Again, we can use standard techniques for this:

```
Romeo_frequency<-Romeo_joy_sad%>%
group_by(word)%>%
summarise(frequency = n())
```

Finally lets view our wordcloud.

```
  wordcloud(Romeo_frequency$word,Romeo_frequency$frequency)
```


art
remedy
daughter
faith
blessed
precious
tomb
woeful
sweet
green
doubt
happy
found
hurt
confession
comfort
friend
virtuous
worn
hope
bride
kiss
murderer
bridal
amen
mad
peace
white
devil
exile
dove
tender
torture
oppression
marriage
pain
quiet
child
slave
hollow
depart
laugh
joyful
pray
ill
fatal
hate
save
joy
withdraw
pay
sorrow
leave
remove
jest
lost
liberty
pretty
sin
worse
delight
hateful
loving
content
cry
kill
marry
dark
forbid
shroud
hire
food
banishment
lie
merry
woe
loss
grave
mother
deadly
honest
lovely
black
lamb
music
garden
fall
saint
late
hell
bloody
banished
cursed
true
bad
lover
teach
plague
proud
beauty
cruel
vow
beg
death
heavens
birth
fury
dance
satisfied
shot
wretched
bury
sun
sadness
murder
fain
sick
passion
die
sing
buried
fault
feeling
bridegroom
youth
delay
fortune
poison
wretch
church
shame
god
weeping
fearful
corse
weep
wound
tree
grief
love