Lately I’ve been interested in trying to cluster documents, and to find similar documents based on their contents.  
In this blog post, I will use [Seneca’s *Moral letters to Lucilius*](https://en.wikisource.org/wiki/Moral_letters_to_Lucilius)  
and compute the pairwise [cosine similarity](https://en.wikipedia.org/wiki/Cosine_similarity) of his 124 letters.  
Computing the cosine similarity between two vectors returns how similar these vectors are. A cosine  
similarity of 1 means that the angle between the two vectors is 0, and thus both vectors have the  
same direction.  
Seneca’s Moral letters to Lucilius deal mostly with philosophical topics, as Seneca was, among many other  
things, a philosopher of the stoic school. The stoic school of philosophy is quite  
interesting, but it has been unfortunately misunderstood, especially in modern times. There is now a renewed interest for  
this school, see [Modern Stoicism](https://en.wikipedia.org/wiki/Modern_Stoicism).

The first step is to scrape the letters. The code below scrapes the letters, and saves them into a list.  
I first start by writing a function that gets the raw text. Note the xpath argument of the html\_nodes()  
function. I obtained this complex expression by using the [SelectorGadget](https://selectorgadget.com/)  
extension for Google Chrome, and then selecting the right element of the web page.  
See this [screenshot](https://i.imgur.com/2cntugt.png) if my description was not very clear.

Then, the extract\_text() function extracts the text from the letter. The only line that might be  
a bit complex is discard(~`==`(., "")) which removes every empty line.

Finally, there’s the get\_letter() function that actually gets the letter by calling the first two  
functions. In the last line, I get all the letters into a list by mapping the list of urls to the  
get\_letter() function.

library(tidyverse)

library(rvest)

base\_url <- "https://en.wikisource.org/wiki/Moral\_letters\_to\_Lucilius/Letter\_"

letter\_numbers <- seq(1, 124)

letter\_urls <- paste0(base\_url, letter\_numbers)

get\_raw\_text <- function(base\_url, letter\_number){

paste0(base\_url, letter\_number) %>%

read\_html() %>%

html\_nodes(xpath ='//\*[contains(concat( " ", @class, " " ), concat( " ", "mw-parser-output", " " ))]') %>%

html\_text()

}

extract\_text <- function(raw\_text, letter\_number){

raw\_text <- raw\_text %>%

str\_split("\n") %>%

flatten\_chr() %>%

discard(~`==`(., ""))

start <- 5

end <- str\_which(raw\_text, "Footnotes\*")

raw\_text[start:(end-1)] %>%

str\_remove\_all("\\[\\d{1,}\\]") %>%

str\_remove\_all("\\[edit\\]")

}

get\_letter <- function(base\_url, letter\_number){

raw\_text <- get\_raw\_text(base\_url, letter\_number)

extract\_text(raw\_text, letter\_number)

}

letters\_to\_lucilius <- map2(base\_url, letter\_numbers, get\_letter)

Now that we have the letters saved in a list, we need to process the text a little bit. In order  
to compute the cosine similarity between the letters, I need to somehow represent them as vectors.  
There are several ways of doing this, and I am going to compute the tf-idf of each letter. The tf-idf  
will give me a vector for each letter, with zero and non-zero values. Zero values represent words  
that are common to all letters, and thus do not have any *predictive power*. Non-zero values are  
words that are not present in all letters, but maybe only a few. I expect that letters that  
discuss death for example, will have the word death in them, and letters that do not discuss death  
will not have this word. The word death thus has what I call *predictive power*, in that it helps  
us distinguish the letters discussing death from the other letters that do not discuss it. The same  
reasoning can be applied for any topic.

So, to get the tf-idf of each letter, I first need to put them in a tidy dataset. I will use the  
{tidytext} package for this. First, I load the required packages, convert each letter to a  
dataframe of one column that contains the text, and save the letter’s titles into another list:

library(tidytext)

library(SnowballC)

library(stopwords)

library(text2vec)

letters\_to\_lucilius\_df <- map(letters\_to\_lucilius, ~tibble("text" = .))

letter\_titles <- letters\_to\_lucilius\_df %>%

map(~slice(., 1)) %>%

map(pull)

Now, I add this title to each dataframe as a new column, called title:

letters\_to\_lucilius\_df <- map2(.x = letters\_to\_lucilius\_df, .y = letter\_titles,

~mutate(.x, title = .y)) %>%

map(~slice(., -1))

I can now use unnest\_tokens() to transform the datasets. Before, I had the whole text of the letter  
in one column. After using unnest\_tokens() I now have a dataset with one row per word. This will  
make it easy to compute frequencies by letters, or what I am interested in, the tf-idf of each letter:

tokenized\_letters <- letters\_to\_lucilius\_df %>%

bind\_rows() %>%

group\_by(title) %>%

unnest\_tokens(word, text)

I can now remove stopwords, using the data containing in the {stopwords} package:

stopwords\_en <- tibble("word" = stopwords("en", source = "smart"))

tokenized\_letters <- tokenized\_letters %>%

anti\_join(stopwords\_en) %>%

filter(!str\_detect(word, "\\d{1,}"))

## Joining, by = "word"

Next step, wordstemming, meaning, going from “dogs” to “dog”, or from “was” to “be”. If you do not  
do wordstemming, “dogs” and “dog” will be considered different words, even though they are not.  
wordStem() is a function from {SnowballC}.

tokenized\_letters <- tokenized\_letters %>%

mutate(word = wordStem(word, language = "en"))

Finally, I can compute the tf-idf of each letter and cast the data as a sparse matrix:

tfidf\_letters <- tokenized\_letters %>%

count(title, word, sort = TRUE) %>%

bind\_tf\_idf(word, title, n)

sparse\_matrix <- tfidf\_letters %>%

cast\_sparse(title, word, tf)

Let’s take a look at the sparse matrix:

sparse\_matrix[1:10, 1:4]

## 10 x 4 sparse Matrix of class "dgCMatrix"

## thing

## CXIII. On the Vitality of the Soul and Its Attributes 0.084835631

## LXVI. On Various Aspects of Virtue 0.017079890

## LXXXVII. Some Arguments in Favour of the Simple Life 0.014534884

## CXVII. On Real Ethics as Superior to Syllogistic Subtleties 0.025919732

## LXXVI. On Learning Wisdom in Old Age 0.021588946

## CII. On the Intimations of Our Immortality 0.014662757

## CXXIV. On the True Good as Attained by Reason 0.010139417

## XCIV. On the Value of Advice 0.009266409

## LXXXI. On Benefits 0.007705479

## LXXXV. On Some Vain Syllogisms 0.013254786

## live

## CXIII. On the Vitality of the Soul and Its Attributes 0.0837751856

## LXVI. On Various Aspects of Virtue .

## LXXXVII. Some Arguments in Favour of the Simple Life 0.0007267442

## CXVII. On Real Ethics as Superior to Syllogistic Subtleties 0.0050167224

## LXXVI. On Learning Wisdom in Old Age 0.0025906736

## CII. On the Intimations of Our Immortality 0.0019550342

## CXXIV. On the True Good as Attained by Reason .

## XCIV. On the Value of Advice 0.0023166023

## LXXXI. On Benefits 0.0008561644

## LXXXV. On Some Vain Syllogisms 0.0022091311

## good

## CXIII. On the Vitality of the Soul and Its Attributes 0.01166490

## LXVI. On Various Aspects of Virtue 0.04132231

## LXXXVII. Some Arguments in Favour of the Simple Life 0.04578488

## CXVII. On Real Ethics as Superior to Syllogistic Subtleties 0.04849498

## LXXVI. On Learning Wisdom in Old Age 0.04663212

## CII. On the Intimations of Our Immortality 0.05180841

## CXXIV. On the True Good as Attained by Reason 0.06717364

## XCIV. On the Value of Advice 0.01081081

## LXXXI. On Benefits 0.01626712

## LXXXV. On Some Vain Syllogisms 0.01472754

## precept

## CXIII. On the Vitality of the Soul and Its Attributes .

## LXVI. On Various Aspects of Virtue .

## LXXXVII. Some Arguments in Favour of the Simple Life .

## CXVII. On Real Ethics as Superior to Syllogistic Subtleties .

## LXXVI. On Learning Wisdom in Old Age .

## CII. On the Intimations of Our Immortality .

## CXXIV. On the True Good as Attained by Reason 0.001267427

## XCIV. On the Value of Advice 0.020463320

## LXXXI. On Benefits .

## LXXXV. On Some Vain Syllogisms .

We can consider each row of this matrix as the vector representing a letter, and thus compute the  
cosine similarity between letters. For this, I am using the sim2() function from the {text2vec}  
package. I then create the get\_similar\_letters() function that returns similar letters for a  
given reference letter:

similarities <- sim2(sparse\_matrix, method = "cosine", norm = "l2")

get\_similar\_letters <- function(similarities, reference\_letter, n\_recommendations = 3){

sort(similarities[reference\_letter, ], decreasing = TRUE)[1:(2 + n\_recommendations)]

}

get\_similar\_letters(similarities, 19)

## XXX. On Conquering the Conqueror

## 1.0000000

## XXIV. On Despising Death

## 0.6781600

## LXXXII. On the Natural Fear of Death

## 0.6639736

## LXX. On the Proper Time to Slip the Cable

## 0.5981706

## LXXVIII. On the Healing Power of the Mind

## 0.4709679

get\_similar\_letters(similarities, 99)

## LXI. On Meeting Death Cheerfully

## 1.0000000

## LXX. On the Proper Time to Slip the Cable

## 0.5005015

## XCIII. On the Quality, as Contrasted with the Length, of Life

## 0.4631796

## CI. On the Futility of Planning Ahead

## 0.4503093

## LXXVII. On Taking One's Own Life

## 0.4147019

get\_similar\_letters(similarities, 32)

## LIX. On Pleasure and Joy

## 1.0000000

## XXIII. On the True Joy which Comes from Philosophy

## 0.4743672

## CIX. On the Fellowship of Wise Men

## 0.4526835

## XC. On the Part Played by Philosophy in the Progress of Man

## 0.4498278

## CXXIII. On the Conflict between Pleasure and Virtue

## 0.4469312

get\_similar\_letters(similarities, 101)

## X. On Living to Oneself

## 1.0000000

## LXXIII. On Philosophers and Kings

## 0.3842292

## XLI. On the God within Us

## 0.3465457

## XXXI. On Siren Songs

## 0.3451388

## XCV. On the Usefulness of Basic Principles

## 0.3302794

As we can see from these examples, this seems to be working quite well: the first title is the  
title of the reference letter, will the next 3 are the suggested letters. The problem is that my  
matrix is not in the right order, and thus reference letter 19 does not correspond to letter 19  
of Seneca…