**How To Create a Word Cloud For Your Favourite Book With R**

Making a word cloud is fun and easy. It is a way of looking at text data and gain a different perspective. For example, if you have a bunch of customer feedback about your product, you can quickly create a word cloud to get some ideas. When I work with text data, it is often the first step I do to quickly show business users something or to simply get the feeling before I get into more advanced analysis such as sentiment analytics or machine learning.

Creating a word cloud for your favourite book is even more fun if you are a book lover. It is another way to get to know your book and gives you a new creative perspective. In this post, I am building a word cloud from [*On The Road*](https://en.wikipedia.org/wiki/On_the_Road) by [Jack Kerouac](https://en.wikipedia.org/wiki/Jack_Kerouac). This is one of my favourite books. It is beautiful, ragged and free.

The simplest way of building word cloud is counting individual words that appear in the document. The more frequent a word appears on the book, the bigger the size of the word becomes in the cloud. To accomplish this task, you first need to download your favourite book and read it into the memory, process it into an appropriate format, and visualise it. The process is relatively simple. In R, the tm package pretty much handles processing and formatting of text data and the wordcloud package handles the creation of word cloud.

There are heaps of instruction available online, too. [Here](https://www.r-bloggers.com/building-wordclouds-in-r/) is the super simple introduction to word cloud with R from R-bloggers. This post will go further into the customisation of the word cloud and the creation of a document term matrix.

OK, enough intro. Let’s get to coding.

**Step Summary**

* Obtain free pdf copy of On The Road online.
* Convert pdf into text data.
* Split it by a new line, convert it into data frame and do clean up.
* Create corpus and do pre-processing with tm.
* Build word cloud.

**Steps**

1. I used [freeditorial.com](https://freeditorial.com/) to get a copy of On The Road. You can also check out [Project Gutenberg](http://www.gutenberg.org/), which offers heaps of free books for downloading. If you are into Jane Austin, R has a package ([janeaustenr](https://cran.r-project.org/web/packages/janeaustenr/index.html)) that contains her complete works and prepped for text analytics. The [harrypotter](https://github.com/bradleyboehmke/harrypotter) package offers the full text of the first seven Harry Potter books.

After downloading the pdf file, I used [pdftools](https://ropensci.org/blog/2016/03/01/pdftools-and-jeroen/) to convert it into text. It converts each page into a vector element. It is a great package to read pdf with R.

[cc lang="matlab" tab\_size="4" lines="-1"]

library(pdftools)

download.file("https://freeditorial.com/en/books/on-the-road/downloadbookepub/pdf", "/tmp/on\_the\_road.pdf", mode = "wb")

txt <- pdf\_text('/tmp/on\_the\_road.pdf')

[/cc]

1. Split the text by new line and create a data frame. Once you have a data frame, it becomes easier to do clean up. I removed the title page, chapter title and the last two lines which are not the part of the novel. I also removed some special characters. I kept the chapter number because this can be dealt with tm later. For this part, you really need to look at the pdf you download and decide what to remove. When you use strisplit, make sure to add carrage return with new line for Windows like \r\n. For Linux and Unix, \n is sufficient.

[cc lang="matlab" tab\_size="4" lines="-1"]

vec <- character()

for(i in 2:length(txt)){

tmp <- strsplit(txt[i], '\n')

for(line in tmp){

vec <- c(vec, line)

}

}

lines <- data.frame(vec)

len <- length(lines$vec)

lines <- subset(lines, !grepl('PART', vec))

lines$vec <- as.character(trimws(lines$vec, 'both'))

lines <- as.data.frame(lines[-c(len, len-1),])

colnames(lines) <- c('vec')

lines$vec <- gsub("Â«", "", gsub("Â»", "", lines$vec))

[/cc]

1. Create a corpus and do pre-processing (remove punctuation, numbers and stopwords and converting all words to lower case). Stemming is important for machine learning like spam identification. For a word cloud, it generates truncated words and looks wired. So, I always omit it.

[cc lang="matlab" tab\_size="4" lines="-1"]

library(tm)

library(SnowballC)

library(wordcloud)

corpus <- Corpus(VectorSource(lines$vec))

corpus <- tm\_map(corpus, PlainTextDocument)

corpus <- tm\_map(corpus, removePunctuation)

corpus <- tm\_map(corpus, removeNumbers)

corpus <- tm\_map(corpus, content\_transformer(tolower))

corpus <- tm\_map(corpus, removeWords, stopwords('english'))

# Stemming doesn't really work with wordcloud.

# corpus <- tm\_map(corpus, stemDocument)

corpus <- tm\_map(corpus, stripWhitespace)

[/cc]

1. To create a word cloud, you can simply pass corpus to the wordcloud function. For colour, I created my own colour vector and randomiser so that colour changes randomly every time I run it. You can use [RColorBrewer](https://moderndata.plot.ly/create-colorful-graphs-in-r-with-rcolorbrewer-and-plotly/) (see Step 5), but I found none of them really gives me the colour combos that I want. I set min.freq = 80 and max.words = 100. 100 words per word cloud is a good place to start. I usually determine min.freq by looking at the document term matrix (see step 6). The example below creates a png file, which is better than generating the cloud in the viewer. By default, the file is saved by the path of the current R session (check it with the getwd function).

[cc lang="matlab" tab\_size="4" lines="-1"]

[/cc]

1. Here is the example of using [RColorBrewer](https://moderndata.plot.ly/create-colorful-graphs-in-r-with-rcolorbrewer-and-plotly/) for a word cloud.

[cc lang="matlab" tab\_size="4" lines="-1"]

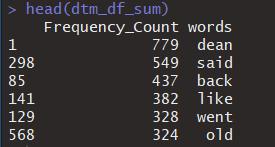
[/cc]

1. A document term matrix has all the words in the text as columns and each line as rows. If a word appears in the row, it puts 1. To create a word count, you can simply aggregate it by columns. The wordcloud function takes column names and column sum as argument instead of corpus as below.

[cc lang="matlab" tab\_size="4" lines="-1"]

[/cc]

You can sort the aggregated data frame to check if the word cloud looks right according to word frequencies.



I also use it to determin the min.freq parameter. 89 words exist for more than 80 frequencies. Hence, I used 80 as min.freq. Each text is different and it is best to check it so that you don’t miss out on words.

Creating DTM is the first step for machine learning on text data (but make sure to include stemming, which is the step I skipped for the word cloud).

[cc lang="matlab" tab\_size="4" lines="-1"]

[/cc]

Here are the word cloud I generated. Most of the books, you will see the characters’ name appearing as the most frequently used word. As expected, you can see dean as the biggest word as the story is centred on the madman, Dean Moriarty. The word cloud sort of makes sense to me. A lot of talking in the book, hence said is the second most frequently used word. Dean always goes back or comes back, hence back comes third. They are always on the move traveling across America, hence the words like went, going, see, around, road, car, way, people, girls, everybody and miles are frequently used words. This is so cool!

**Custom Colour**

**RColorBrewer (Set3)**

As mentioned above, I prefer using my own colour selections than any of the RColorBrewer palette as the palette sometimes give me the colours that are hard to read!

Now it’s your turn to create a word cloud for your favourite book!