Sentiment Analysis

Prateek

6/6/2020

SENTIMENT ANALYSIS

This is a project to understand the basics of R language. I referred to the various websites particulary Datacamp and Data-Flair for getting the most of the codes. From my end I read and understood the working and functioning each of the packages and R functions used in this project.

By end of this project I won't claim I learnt everything about R language, however the self paced learning taught me the art of exploring, understanding and finding my way out when working on R. Although the entire code on Sentiment Analysis is available on many websites, I found most of them failed to give a thorough explanation of:

"why a particular function is used?",

"Role of a particular step in the end result", and

"The various attributes available with a particular function".

In this article I will try to explain the usage of each step of code w.r.t. to the above questions that comes to any first time R language learner.

So, Let's begin

```
library(tidytext)
library(janeaustenr)
library(stringr)
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
```

First step is to load these 4 packages to the global environment. "tidytext" is an important part of this project. The package and its associated functions lets us handle data comprising of texts. Remember that in tidytext package the table of tidy data is stored in a token format and is in a format of #one #token #per #row. Token usually means a one single word but it can also be a sentence, paragraph or even one complete chapter.

In our project, we aim to analyse each word in the book and rate those words on a sentiment scale. Hence, we would be tokenizing the tidydata to one word/one token in our case.

"janeaustenr" loads a package containing each word published in 6 books written by Jane Austen. We will be doing sentiment analysis of words in one of the above 6 books.

"stringr" loads a package which lets us use the pipe operator(%>%) in next step and other string functions.

"dplyr" is a package that lets us use the "group_by" function.

```
tidy_data<- austen_books()%>%group_by(book)%>% mutate(linenumber=row_number(),
          chapter=cumsum(str_detect(text,regex("^chapter [\\divxcl]", #\\d includes all decimal digits
ignore_case = T)))%>% #ignorecase considers both small and big case
 ungroup()%>%unnest_tokens(word,text)
```

The above chunks of code is an important step in this project. We are aiming to create a tibble in which the words in janeausten package are grouped by order of book, provide chapter number against each word depending upon the chapter it appears in, and tokenize the janeausten package to one word level. In simple terms, we are *tidying* the text data in tidy_data tibble. Pipe Operator(%>%) - The pipe operator takes the output of one statement and makes it the input of next statement. This is somewhat similar to chaining. e.g. f(g(h(x))) can be piped as x%>%h()%>%g()%>%f(). Pipe operator hence allows the chaining, without needing intermediate variables to store the value. Also, it eliminates the use of lots of parenthesis making it *readable* in a code chunk and presents the chained code in a *logically sequenced* format. group_by- The function groups the austen_books() data frame by order of the books present in data. This is not a visual change i.e. group_by function won't make a readable change to the structure of data.

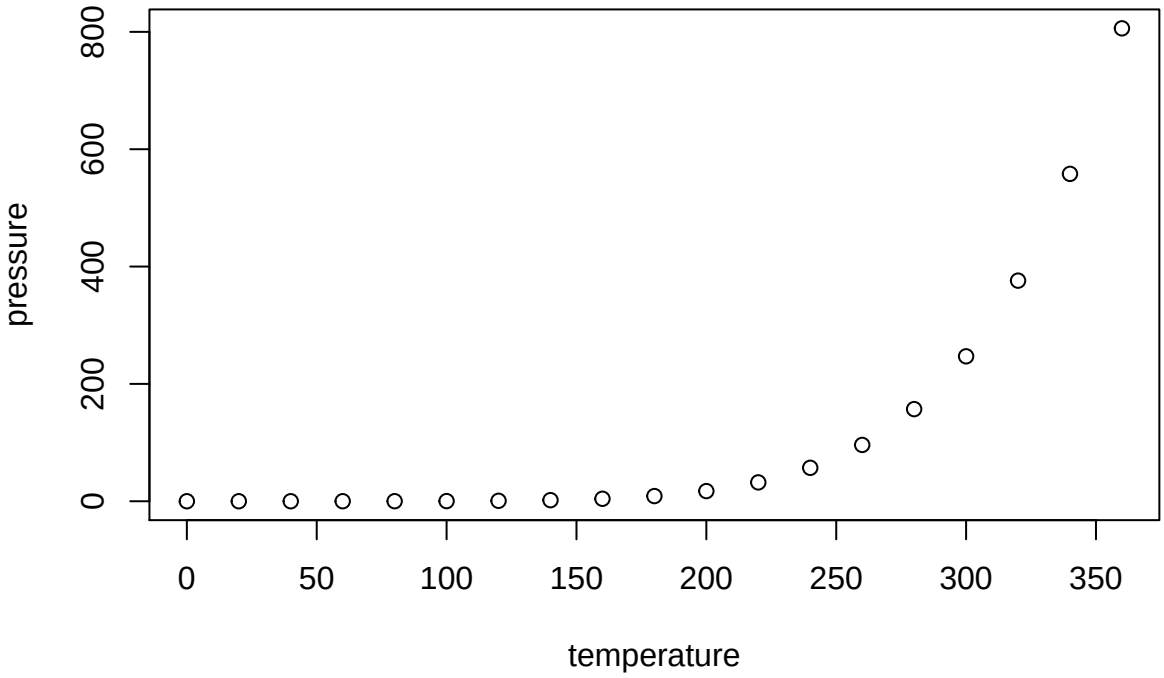
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.