**Dictionary of functions for text analysis with R, in the order they appear**

**Day 2. Exercises 1-20**

|  |  |  |  |
| --- | --- | --- | --- |
| **Exercise Number** | **Line of code** | **What it does** | **Package the function is from** |
|  | group\_by(title) | Groups the data by title | **dplyr** |
|  | %>% | Pipe operator. Allows you to pass data through multiple functions. A more efficient way to write code. | **magrittr (used in dplyr)** |
|  | unnest\_tokens(word, text) | Split a line of text into tokens using the tokenizers package. Splits the table into one-token-per-row. | **tidytext** |
|  | anti\_join(stop\_words) | Removes common stop words (a, an, the, etc) from the data set | **tidytext** |
|  | ungroup() | Removes any grouping of the data specified earlier using a group\_by() function | **dplyr** |
|  | count(word, sort=TRUE) | Does a word count and sorts based on highest to lowest count | **dplyr** |
|  | filter() | Filters the data set based on a parameter you specify in the parentheses | **dplyr** |
|  | mutate(word= reorder(word, n) | Creates a new variable “word” in the data frame. Also, reorder() sorts the data on the new column by count (n) | **dplyr** |
|  | ggplot(aes(word, n)) + geom\_col() + xlab(NULL) + coord\_flip() | Creates a column chart with word on the X axis, and count on the Y axis, but flips X and Y so columns are horizontal | **ggplot2** |
|  | summarize(total = sum(n)) | There are two functions here. Summarize() produces a summary statistic. total = sum(n) adds up all the n values and assigns that to total. So the summary statistic is on the total variable | **dplyr** |
|  | left\_join(freq\_words, total\_words) | Creates a data join of two tables: freq\_words and total\_words | **dplyr** |
|  | bind\_tf\_idf(word, title, n) | creates 3 ratios for each term: tf, idf and tf\_idf. adds each ratio as its own column | **tidytext** |
|  | select(-total) | drops the “total” variable from data set | **dplyr** |
|  | arrange(desc(tf\_idf) | sorts the data on tf\_idf column, in descending order | **dplyr** |

**Exercises 21-30**

|  |  |  |  |
| --- | --- | --- | --- |
| **Exercise Number** | **Line of code** | **What it does** | **Package** |
|  | |  | | --- | |  | | unnest\_tokens(bigram, text, token = "ngrams", n = 2) | |  | | Create bigrams from every line of text in all of Stoker’s works | **tidytext** |
|  | head(bigrams, 40) | Prints out 40 rows of the bigrams data | **base R** |
|  | separate(bigram, c("word1", "word2"), sep = " ") | Separates bigrams into two words, as “word1” and “word2” | **tidyr** |
|  | |  | | --- | | mutate(word1 = str\_extract(word1, "[a-z']+")) %>% mutate(word2 = str\_extract(word2, "[a-z']+")) | |  |  | | Creates 2 new columns: word1, word2 and stores the separated bigrams in those columns.  Strips punctuation and non-text characters off of the words | **dplyr** |
|  | |  | | --- | |  | | filter(!word1 %in% stop\_words$word) %>% filter(!word2 %in% stop\_words$word) | |  |  | |  |  | | Removes stop words from columns word1 and word2  %in% looks for a match | **dplyr** |
|  | |  | | --- | | count(word1,word2, sort = TRUE) | |  |  | | Runs a word count function on columns word1 and word2 and sorts them | **dplyr** |
|  | graph\_from\_data\_frame() | Makes a network of the bigram\_counts data frame – note, this is not a chart, it’s a network | **igraph** |
|  | |  | | --- | |  | | ggraph(bigram\_graph, layout = "fr")… | |  |  | |  |  | |  |  | |  |  | | This creates a visual network graph from the network data that is stored in the bigram\_graph variable. Using a fruchterman reingold layout algorithm | **ggraph** |
|  | |  | | --- | |  | | filter(title=="Dracula", str\_detect(text, "death")) %>%   |  |  | | --- | --- | |  | select(text, title) %>% | |  | write.csv(.,file = "~/text-analysis-with-R/strings.csv") | |  |  | | | Filters out only the text from Dracula, searches for the word “death” and saves lines from the text that contain “death” into a CSV file. | **filter() and select() are from dplyr**  **str\_detect is stringr (part of tidyverse)**  **write.csv() is base R** |

**Exercises 31-45**

|  |  |  |  |
| --- | --- | --- | --- |
| **Exercise Number** | **Line of code** | **What it does** | **Package** |
| 32 | |  | | --- | | mutate(linenumber = row\_number()) %>% filter(linenumber>156) | |  |  | | Creates a new column “linenumber” and assigns the row number value as linenumber for each line. Filters out lines 1-156 – the title and table of contents | **dplyr** |
| 33 | |  | | --- | |  | | mutate(chapter = cumsum(str\_detect(text, regex("^chapter [\\divxlc]", ignore\_case = TRUE)))) | |  |  | | Creates a new column “chapter”  Assigns every line a chapter number by looking for the word “chapter” followed by Roman numerals and then adding 1 to the chapter number each time. Ignores whether chapter is upper or lower case. | **mutate() is from dplyr**  **regex() is base R** |
| 35 | unite(document, chapter) | This effectively changes the variable “chapter” to “document” to avoid confusion when doing the document-term matrix. It essentially collapses the chapter column into a new column, “document” | **dplyr** |
| 36 | |  | | --- | |  | |  | | cast\_dtm(document, word, n) |  | |  |  | | Creates a document-term matrix along with word counts for each term | **tidytext** |
| 37 | LDA(chapters\_dtm, k = 12, control = list(seed = 1234)) | Runs a Latent Dirichlet Allocation model, which assumes that each document is a mixture of a small number of topics | **tidytext** |
| 38 | tidy(chapters\_lda, matrix = "beta") | Returns a ‘tidy’ data frame with one row per topic and term from the LDA model | **tidytext** |
| 39 | chapter\_topics %>%  group\_by(topic) %>%  top\_n(5, beta) %>%  ungroup() %>%  arrange(topic, -beta) | Creates a data frame where you get the top 5 terms per chapter | **dplyr (all 4 functions in this line)** |
| 40 | mutate(term = reorder(term, beta)) | This reorders the term factor based on the beta variable. Purpose: to order the bars of the graph that’s created in the next line | **dplyr** |
| 40 | ggplot(aes(term, beta, fill = factor(topic))) + geom\_col(show.legend = FALSE) + facet\_wrap(~ topic, scales = "free") + coord\_flip() | Creates a column chart with no legend, broken into subplots (facets) by topic. Each plot has its own scale | **ggplot2** |
| 41 | Do not run this section during the workshop  The code for this section comes from this Stack Overflow post: <https://stackoverflow.com/questions/28764056/could-not-find-function-tagpos> | This section tags the text of Dracula with POS tags, specifying the part of speech for every word. | **NLP**  **openNLP** |
| 43 | sapply(strsplit(tagged,"[[:punct:]]\*/NN.?"),function(x) {res = sub("(^.\*\\s)(\\w+$)", "\\2", x); res[!grepl("\\s",res)]} ) | sapply() allows you to apply a function to a vector and get a vector result back. In this case, the function gets only the words tagged as nouns (//NN) and returns just those words, minus the tag.  This uses grep: a Unix command used to search files for the occurrence of a string of characters that matches a pattern. | **base** |
| 45 | nouns[!nouns %in% stop\_words]  nouns[!nouns %in% stopwords()] | Removes specified stop words in the stop\_words list we created, and it removes the words in the stopwords() data set from the tm package | **base**  **tm** |
| 46 | data.frame(text=nouns, stringsAsFactors = F) | Makes a data frame from nouns, but imports the categorical data without factors | **base** |
| 47 | nouns\_dtm <- nouns\_df %>% cast\_dtm(document, text, n) | Creates a document-term matrix of nouns\_df | **tidytext** |
| 48 | LDA() | See #37 | **tidytext** |
| 49 | mutate(term = reorder(term, beta)) %>%  ggplot(aes(term, beta, fill = factor(topic))) + geom\_col(show.legend = FALSE) + facet\_wrap(~ topic, scales = "free") + coord\_flip() | Does the same thing as #40, creates a graph of the top topics in the novel, but hopefully better since all terms but nouns were removed | **dplyr**  **ggplot2** |