**Aly6040 Data Mining**

**Assignment 5**

**Technique Practice**

**Professor: - Prof Harpreet Sharma**

**Submitted by: - Harshil Bhatt**

**NEUID: - 002780601**

**Introduction**

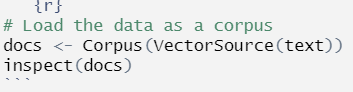
[***http://www.sthda.com/sthda/RDoc/example-files/martin-luther-king-i-have-a-dream-speech.txt***](http://www.sthda.com/sthda/RDoc/example-files/martin-luther-king-i-have-a-dream-speech.txt)

Above text file containing the famous speech titled "I Have a Dream" delivered by Martin Luther King Jr. This speech holds significant historical and cultural importance, as it was given during the Civil Rights Movement in the United States on August 28, 1963.

In the speech, Martin Luther King Jr. passionately advocates for racial equality and an end to discrimination against African Americans. He eloquently expresses his dream of a future where people are judged by the content of their character rather than the color of their skin. The speech has become an iconic representation of the struggle for civil rights and has left a lasting impact on the fight for equality and justice worldwide.

**Code Walkthrough**

**Step 1: Create a text file**



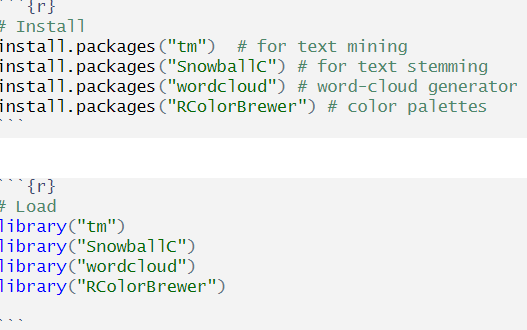
***Fig1.1 Loading text***

First creating a corpus object named **docs** by reading the content of the text file. Here's what each component means:

* **VectorSource(text)**: This creates a source object called **VectorSource** from the provided **text**. The **text** variable should contain the actual text content you want to load.
* **Corpus(VectorSource)**: This function creates a corpus object, which is a collection of text documents, from the **VectorSource** object. In this case, it creates a corpus containing a single document.

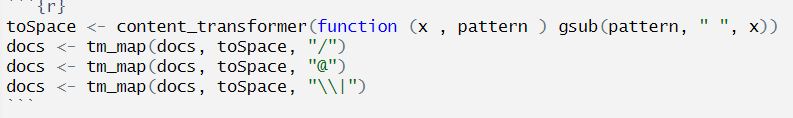
And lastly inspecting the document in R environment

**Step 2 : Install and load the required packages**



Cleaning and normalize the text data before further analysis or modelling tasks

**Text transformation**



***Fig 1.2 replacing specific patterns with spaces in the docs’ corpus object.***

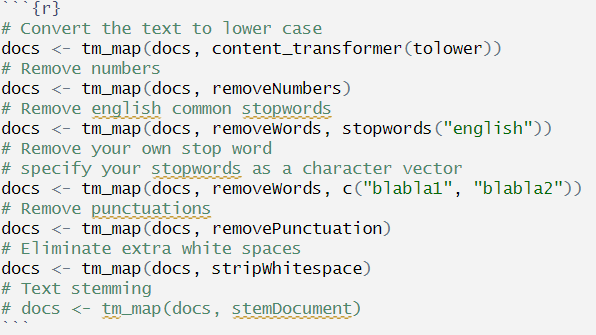
gsub(pattern, " ", x): toSpace - content\_transformer(function (x, pattern) gsub(pattern, " ", x)): This line declares the toSpace custom content transformer function. It accepts two arguments: x for the text to be converted and pattern for the pattern to be substituted. The gsub() method is used to substitute the pattern with a space (" ") in the string x.

documents - tm\_map(docs, toSpace, "/"): Using the tm\_map() function, this line applies the toSpace transformation to the docs corpus. It substitutes forward slashes ("/") in the text with spaces. This is handy for separating words or sentences separated by forward slashes.

documents - tm\_map(docs, toSpace, "@"): This line performs the toSpace transformation to the docs corpus once again, substituting spaces for the "@" sign. This might be done to eliminate email addresses or usernames from the text.

documents - tm\_map(docs, toSpace, "|"): This line performs the toSpace transformation again on the docs corpus, substituting vertical bars ("|") with spaces. This can be handy for separating vertically separated components in a text.

**Cleaning the text**

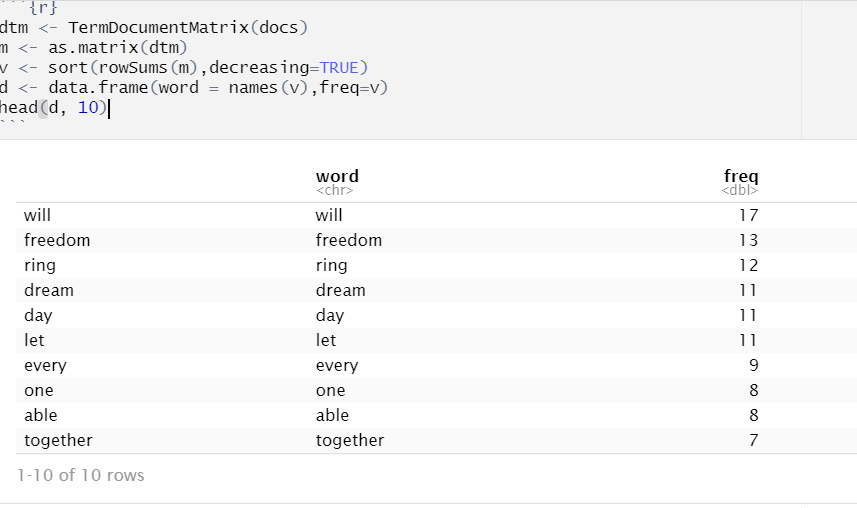


***Fig1.3 converting text for processing***

The provided code performs several pre-processing steps on the **docs** corpus object, resulting in a transformed text data ready for analysis or modelling tasks. These steps include converting the text to lowercase, removing numbers, eliminating common English stop words, removing user-defined stop words, removing punctuation, and eliminating extra white spaces. Additionally, the code includes a commented-out step for text stemming, which reduces words to their base or root form.

The purpose of these pre-processing steps is to standardize the data, remove noise and irrelevant elements, and simplify the text for further analysis. By converting the text to lowercase, case differences are ignored, avoiding duplications. Removing numbers helps focus on the textual content rather than numerical values. Eliminating common English stopwords reduces noise and allows for a more meaningful analysis of the remaining words. Removing user-defined stop words allows for the removal of specific words that are not relevant to the analysis or modelling goals. Removing punctuation simplifies the data by eliminating non-essential characters. Eliminating extra white spaces ensures consistent spacing between words. Lastly, stemming, if enabled, reduces words to their base form, aiding in consolidating the vocabulary size.

### Step 4 : Build a term-document matrix



***Fig1.4 Table formation***

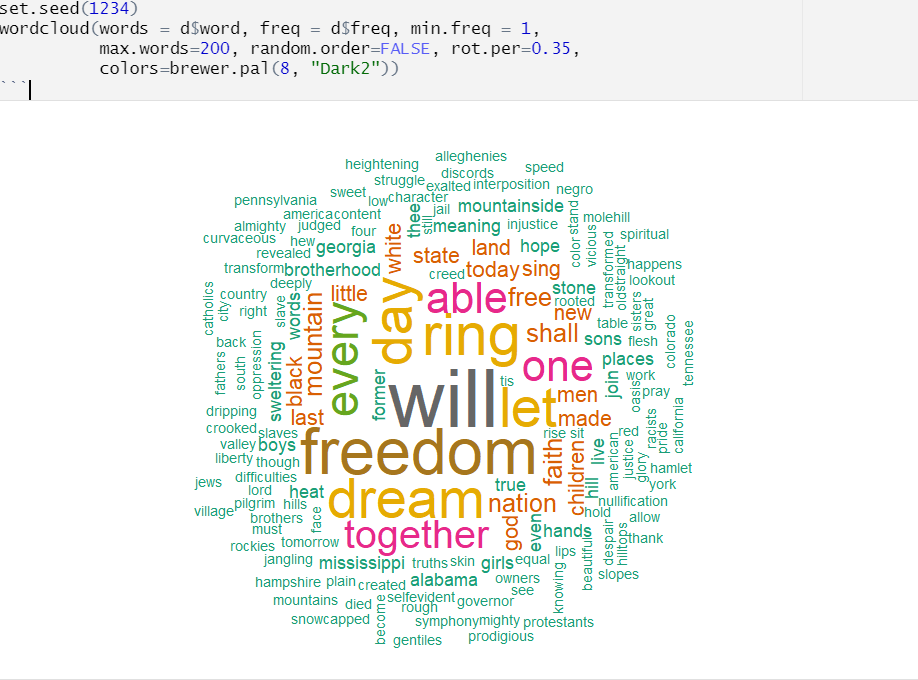
TermDocumentMatrix(docs): Creates a term-document matrix (dtm) from the docs corpus. A term-document matrix is a text data structure in which rows represent terms (words) and columns represent documents. The frequency of a phrase in a document is represented by each cell in the matrix. as.matrix(dtm): m The term-document matrix (dtm) is converted into a regular matrix (m) by this line. It is done to make future calculations and analyses easier.

v - sort(rowSums(m),descending=TRUE): By computing the row sums of the matrix m, this line computes the sum of word frequencies over all texts. The rowSums() method computes the overall frequency of each word across all texts for each row. The word frequencies are sorted in decreasing order in the resultant vector v.

data.frame(word = names(v), frequency=v): This line generates a data frame d that contains two columns: "word" and "freq." The "word" column holds the v vector's names (words), whereas the "freq" column has the matching frequencies. head(d, 10): This line displays the first ten rows of the data frame d, displaying the top ten most frequently occurring terms.

table with the word frequencies sorted in descending order. The table would contain two columns: "word" and "freq." The "word" column would display the individual words, and the "freq" column would display their corresponding frequencies.

### Step 5 : Generate the Word cloud



***Fig1.5 Word cloud***

d words =$word: This option defines which words will appear in the word cloud. It reads the column "word" from the d data frame. freq = d$freq: The size of each word in the word cloud is determined by its frequency. It reads the column "freq" from the d data frame. min.freq = 1: Sets the minimum frequency threshold for a word to appear in the word cloud. Words having a frequency of less than or equal to one will be shown.

max.words = 200: The maximum number of words displayed in the word cloud is limited by this option. It enables a maximum of 200 words in this scenario. random.order = FALSE: This option controls whether the words in the word cloud are presented in random order or by frequency. When set to FALSE, the terms are shown in decreasing order of frequency. rot.per = 0.35: This option determines the proportion of words that will be presented vertically (rotated). 35% of the text will be shown vertically in this situation. brewer.pal(8, "Dark2"): colours The colour palette for the word cloud is determined by this option. It employs the "Dark2" palette from the brewer.pal() function, which has eight colours.

**Output:-**

The word cloud visualization would provide insights into the prominent words in the text data. The size of each word in the word cloud corresponds to its frequency, with larger words indicating higher frequencies. This helps identify the most common and significant words in the corpus. The colors used in the word cloud add visual appeal and can help distinguish different words or themes.

### Explore frequent terms and their associations

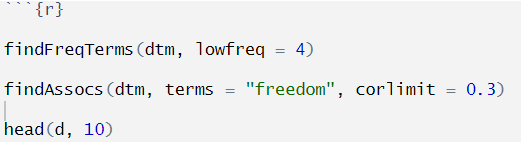
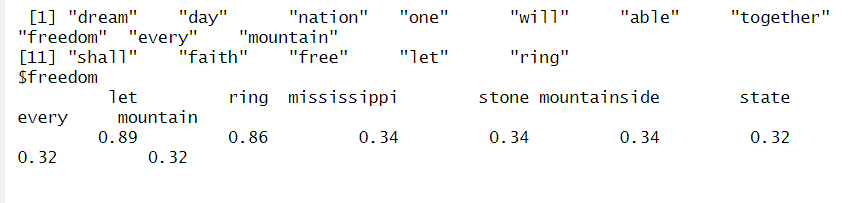


Fig 1.7

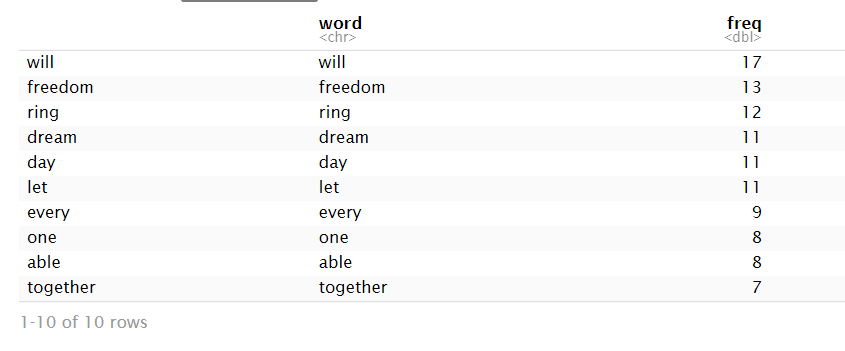
**findFreqTerms(dtm, lowfreq = 4)**: This line identifies terms (words) in the document-term matrix **dtm** that have a frequency greater than or equal to 4. It returns a vector of terms that occur frequently enough in the corpus. **findAssocs(dtm, terms = "freedom", corlimit = 0.3)**: This line finds terms in the document-term matrix **dtm** that are associated with the term "freedom" based on their co-occurrence. The **corlimit** parameter specifies the minimum correlation threshold, with a value of 0.3 in this case. It returns a vector of terms that have a positive correlation with "freedom" above the specified threshold.

**Output**:-



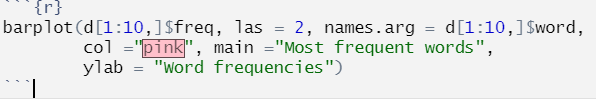
vector of terms that are positively correlated with the term "freedom" above the specified correlation threshold. These terms are likely to be associated with the concept of freedom in the text data. The output of **head(d, 10)** would be a table containing the first 10 rows of the **d** data frame, which represents word frequencies. This table shows the top 10 words with the highest frequencies in the corpus. As shown in below figure

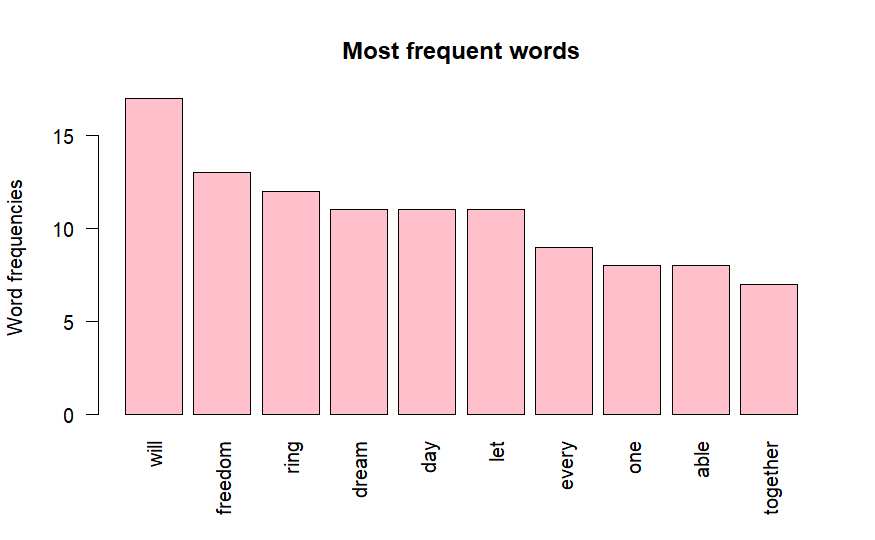
**The frequency table of words**



***Fig 1.8 Table containing word frequencies***

### Plot word frequencies





d[1:10,]$freq: The frequency values for the top ten words to be plotted are specified by this parameter. It chooses the frequency column of the d data frame for rows 1 through 10.

This option determines the orientation of the x-axis labels. When set to 2, the labels are rotated vertically for easier reading. d[1:10,] names.arg = d[1:10,]$word: This option specifies the x-axis labels. It chooses the word column of the d data frame for rows 1 through 10. col = "pink": This option determines the colour of the plot's bars. The colour is set to "pink" in this scenario. You may change the colour name or code to make it your own.

main = "Most frequent words": This option changes the plot's title to "Most frequent words." It gives the plot a descriptive title. ylab = "Word frequencies": This option determines the y-axis label. It denotes that the y-axis numbers represent word frequencies.

**References**

*Text mining and word cloud fundamentals in R : 5 simple steps you should know*. STHDA. (n.d.). http://www.sthda.com/english/wiki/wiki.php?id\_contents=20546

**Conclusions**

Word cloud generation and text mining are powerful approaches for analysing unstructured text data. Using these approaches, the primary themes and words that featured the most frequently in Martin Luther King Jr.'s "I Have a Dream" speech could be identified. This study gives crucial information on the meaning and effect of the speech while highlighting the significance of text mining in finding hidden patterns and relationships in large text sets.