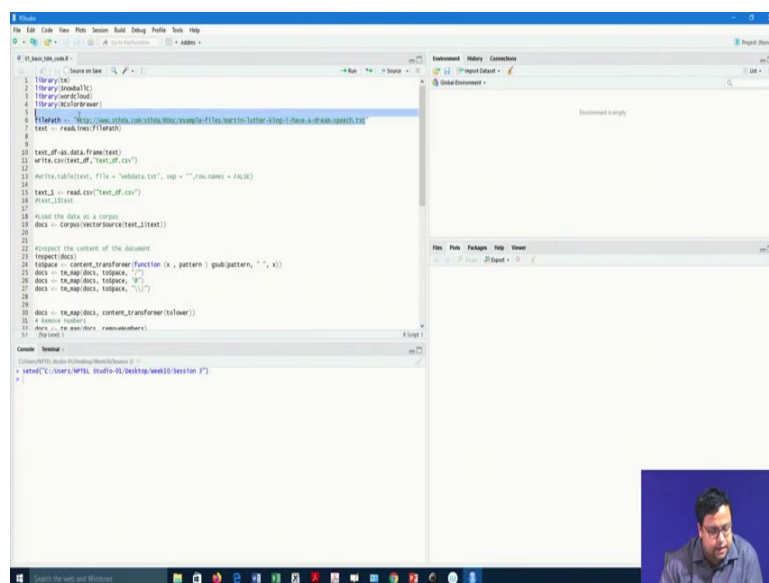


Marketing Analytics
Professor Swagato Chatterjee
Vinod Gupta School of Management
Indian Institute of Technology, Kharagpur
Lecture 53
Text Mining and Sentiment Analytics (Contd.)

Hello everybody welcome to Marketing Analytics course, this is Dr. Swagato Chatterjee from VGSOM, IIT Kharagpur who is taking this course. We are in week 10 and this is session 3 we are discussing about Text Mining in the context of marketing.

(Refer Slide Time: 0:32)



```
1 library(tm)
2 library(snowballc)
3 library(wordcloud)
4 library(RColorBrewer)
5 #urlpath <- "http://www.utdallas.edu/~davidson/1301/lectures/lect13/king-1-have-a-dream-speech.txt"
6 text <- readLines(urlpath)
7
8
9
10 text_df <- data.frame(text)
11 write.csv(text_df, "text_df.csv")
12
13 #url <- "http://www.utdallas.edu/~davidson/1301/lectures/lect13/king-1-have-a-dream-speech.txt"
14 #url <- "http://www.utdallas.edu/~davidson/1301/lectures/lect13/king-1-have-a-dream-speech.txt"
15 text_1 <- read.csv("text_df.csv")
16 #print(text_1)
17
18 #load the data as a corpus
19 docs <- corpus(VectorSource(text_1$text))
20
21
22 #inspect the content of the document
23 inspect(docs)
24 tokenize <- content_transformer(function(x, pattern) gsub(pattern, "", x))
25 docs <- tm_map(docs, tokenize, "/")
26 docs <- tm_map(docs, tospace, "/")
27 docs <- tm_map(docs, tospace, "\\|")
28
29
30 docs <- tm_map(docs, content_transformer(tolower))
31 # return a number
32 docs <- tm_map(docs, removeNumbers)
33 # the text
```

So, if you see the file there is something called basic TDMcode.r, I have to thank my friend, Mr. Saptarsi Sarkar for sharing this this file with me, the actual file has been made by him but I am using this for the course and I think, he has given me the permission.

So the first thing in this particular thing, but in this particular class, what we are trying to do is that we will play with some text data and this text data is available online, freely available text data, and it is about Martin Luther King, Martin Luther King, I Have a Dream speech.

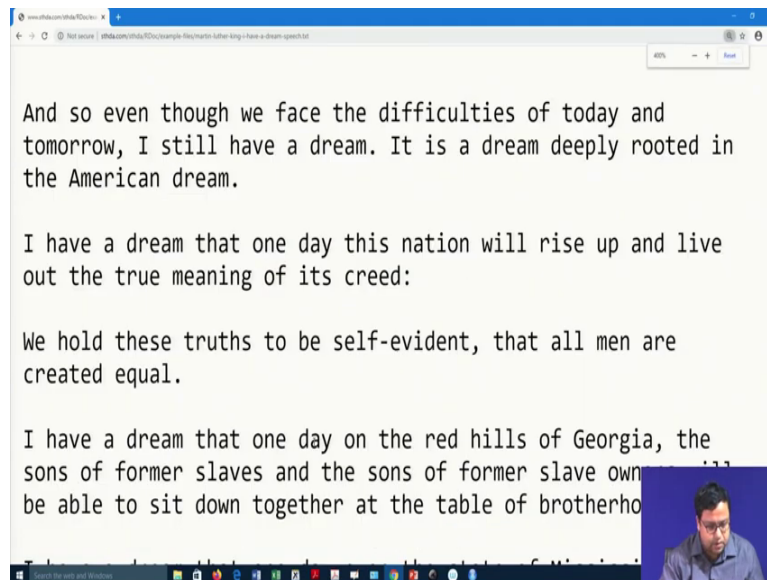
So that particular speech is what is there in the text and we will try to analyze that. So to do that, there are certain libraries that you require, you require a library called tm then Snowballc, wordcloud and RColorBrewer, we will discuss about them one by one. So first things, we have to check that whether these libraries are already installed or not. So I can see probably tm library, you can see here that tm library is not installed in my this thing. So I will try to call this library and it will say that error there is no package called tm.

So what I will I am going to do now in the next probably 2, 3 minutes, is to install all these packages so install .packages then tm , so we have to install quite a few package. So tm what else, Snowballc, so S capital, Snowballc, and then the next one is wordcloud, so wordcloud and the next one is RColorbrewer, so let us just install these files. So it might take some time depending on the internet connection.

So tm library is the classic library that we will be using. And they have, it has dependency called nlp and slam, so it has downloaded that, and the other ones have also been downloaded. So if you see that nlp, slam, tm, Snowballc, wordcloud and Colorbrewer all have been downloaded. So if I now call these four libraries together, I think they will work. So some of them saying that was built under 3.6.3, so tm built was 3.6.3, so you have to check whether your R version is compatible to this, but warning sign means it is okay we can go ahead.

So, I have till now what I did is, I install certain libraries, then what I will do? Then I will actually is not set working directory. So this I will delete and then I will set working directory to my source file location wherever this file is currently residing this port file and then I am putting this thing at file path is called to this particular link.

(Refer Slide Time: 4:06)

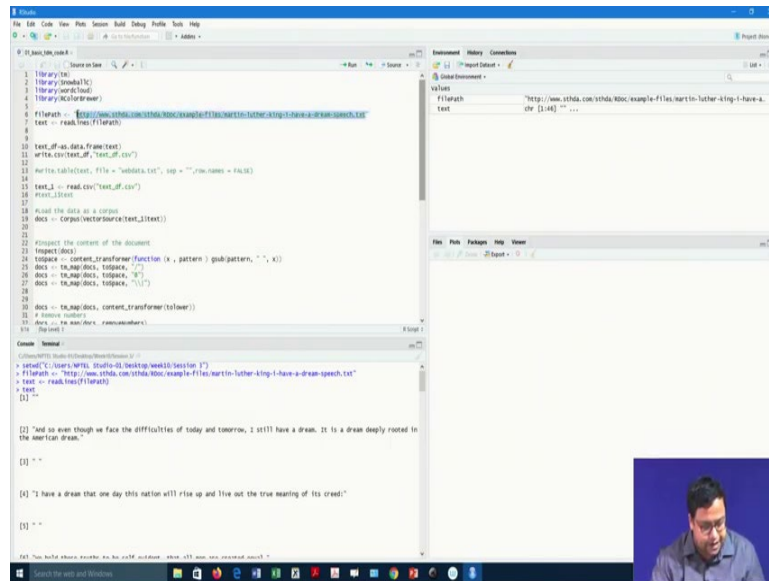


So if I just go to the link, what does it look like? Let us just check that. So if I just go to this link, so in this link, this whole text is there, I can probably make it bigger. And okay so, and so even though we face the difficulties of today and tomorrow I still have a dream it is a dream deeply rooted in the in the American dream and so on. So the great American Dream

Martin Luther King's speech, I have a dream that one day this nation will rise up and leave out the true meaning of its creed blah, blah, blah, is given that particular thing is written in this particular.

So we can read a text, it could have been some other file also. So there are different ways to read files from internet, text we can directly do that.

(Refer Slide Time: 4:52)



```
1 library(tar)
2 library(rvest)
3 library(xml2)
4 library(xml2)
5 # Fetch the text from the URL
6 text = readLines("http://www.s3.amazonaws.com/example-files/martin-luther-king-i-have-a-dream-speech.txt")
7
8
9
10 test_of_no_data_frame(text)
11 # Write the text to a file
12 writeLines(text, "test_of_csv")
13 # Create a data frame from the file
14 df = read.csv("test_of_csv", sep = ",", as.is = TRUE)
15 # Print the data frame
16 print(df)
17
18 # Read the data as a string
19 docs = corpus(VectorSource(text))
20
21 # Inspect the content of the document
22 inspect(docs)
23 # Create a document transformer
24 tokenize <- content_transformer(function(x, pattern) gsub(pattern, "", x))
25 docs <- tm_map(docs, tokenize)
26 docs <- tm_map(docs, tolower)
27 docs <- tm_map(docs, tospace, " ")
28
29 # Create a document transformer
30 docs <- tm_map(docs, content_transformer(tolower))
31 # Create a document transformer
32 docs <- tm_map(docs, removeNumbers)
```

Console Output:

```
> print("Using rvest, I can fetch the text from the URL")
[1] "Using rvest, I can fetch the text from the URL"
> # Fetch the text from the URL
[1] "I have a dream that one day this nation will rise up and live out the true meaning of its creed!"
[2] ""
[3] "And so even though we face the difficulties of today and tomorrow, I still have a dream. It is a dream deeply rooted in the American dream."
[4] ""
[5] "I have a dream that one day this nation will rise up and live out the true meaning of its creed!"
[6] ""
```

So I can I have to give file path. So file path is basically the link of this particular data. That is my file path and the text is read lines with lines fields line by line it will read, so read lines file path.

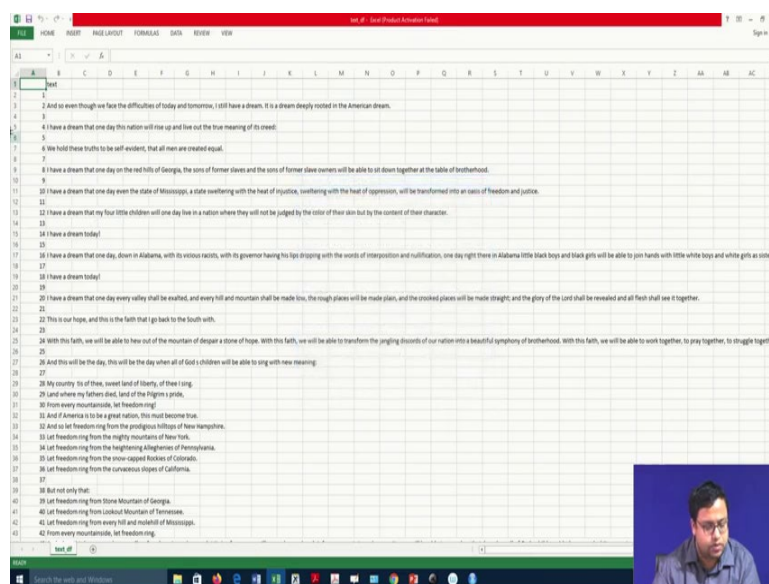
And that gives, if I now see the text, it will see the line by line has it has written. So, the first line was blank and then, so even though we face the difficulties of today, tomorrow, I shall have a dream that comes in the second line, third line was also blank, because if you remember if you if you remember how the text was there, you will see that the first line was blank.

And then second line onwards, there was some text is there here it was blank. There is some text then one blank then some text then again blank, then again text, so one single line by line it will just try to read. So whenever you press an enter in the text, when whoever created this text document whenever he has pressed and enter, another line got generated. So if that line was nothing written in that, then that line is blank, and then you go to the next line. So that is how we did this thing. So, I have some blanks and some text in this particular data.

Next work, next I will change this text data which is a character variable to a data frame. So as data frame text, text df = ask a different text. It is nothing but one single observation the same things with some blank values and some text values are there, why we do that? Because I can now save this.

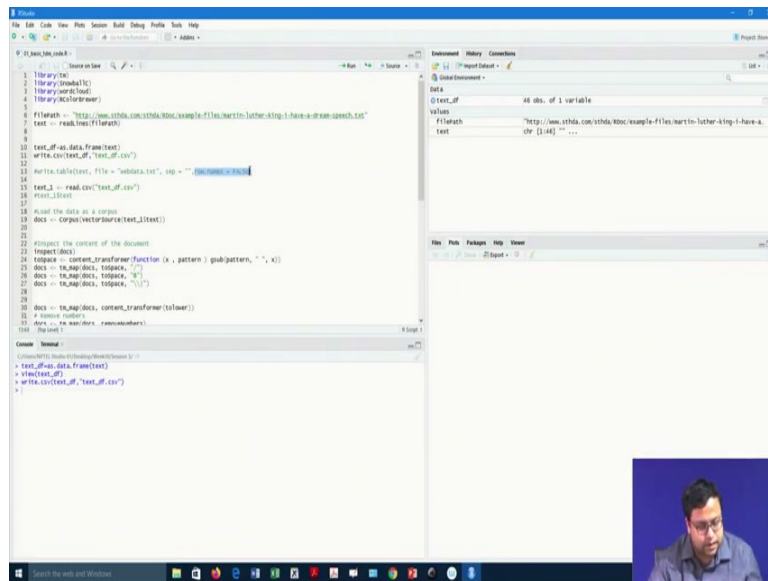
So I will save this in my file wherever the files are stored. I can save this so that the next time by chance the internet is off, and I want to do this thing once more, I do not have to read the file again from internet once more because if internet is not there, then there is a problem. So this is how the data is now saved here, text_df.

(Refer Slide Time: 7:16)



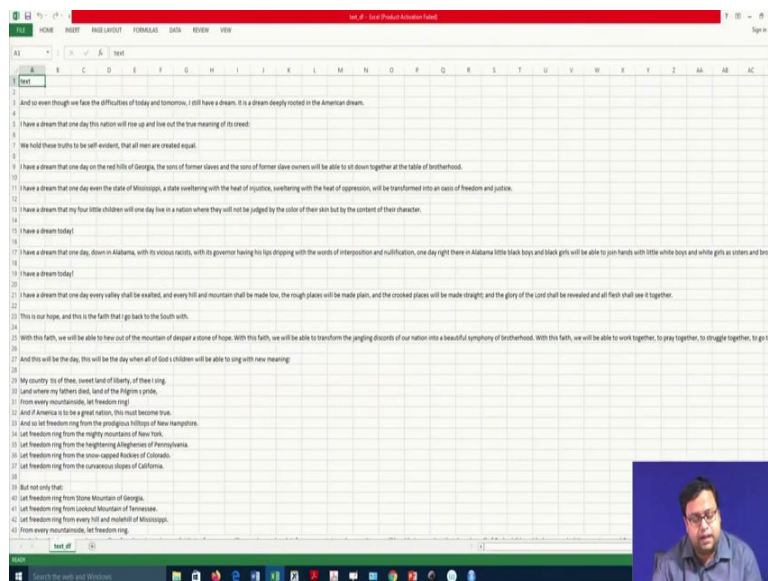
Okay, so if you check it carefully here, there is another column that has come up, which is this 1, 2, 3, 4, 5 thing things, these things have come up. So these are basically row names and then when there is no row names, the serial numbers of the rows comes up.

(Refer Slide Time: 7:32)



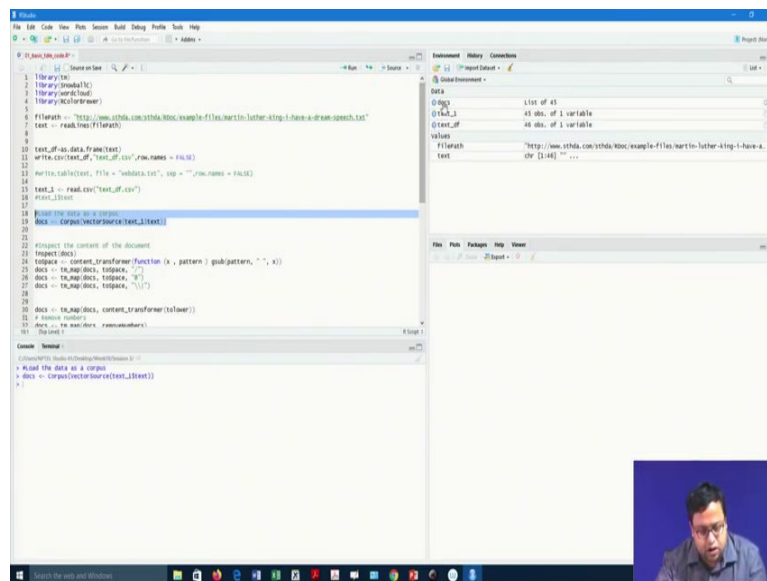
If you do not want that, if you do not want that, you have to write . CSV text df then CSV and then row names row.names = false you can write this that , row.names = false. It will just append the previous thing.

(Refer Slide Time: 7:57)



Now see there is no row names. So 1, 2, 3, 4, 5, 6, 7, 8 has gone. So, this is something that you can try to do.

(Refer Slide Time: 8:05)



Now, what I will be doing is that my text one I will, I am reading the text once more. So text one = read.CSV, text_df .CSV I am reading the text once more, why? I am reading the text once more, because now I can play with this text. So, this is text document that I have bought. So, if I just want to see what are the texts that are there, that is basically text 1 dollar text is the column name. So text one not from, sorry, text_one text_1 dollar text and that is this text.

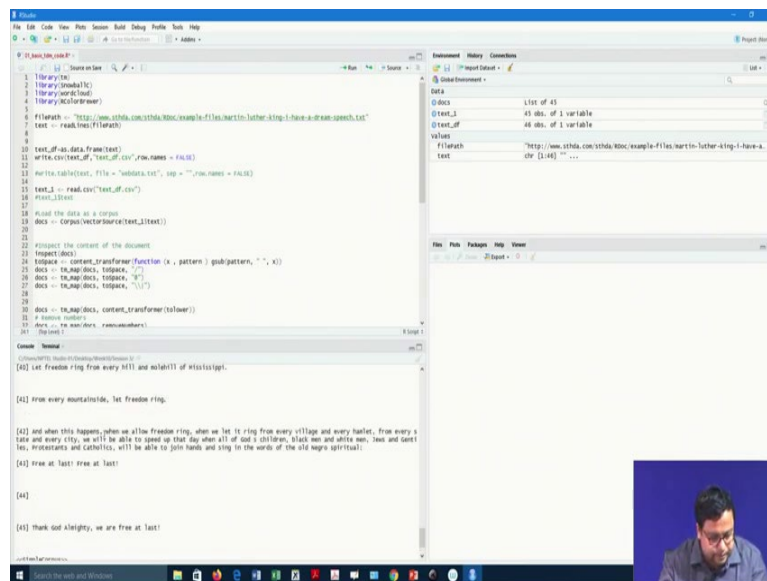
This is what I will be playing with. Now once I have a column of data set, there is something called corpus corpus is a special form like if you remember factor was a special form of screen variable while you can analyze the screen data with numerical representations. Similarly here, this text_df is one kind of data set whose text I can convert to something called corpus.

Corpus is a special form which can be worked with tm library tm library can easily analyze this corpus form of that data. And this is generally takes data and corpus can be formed from text files from CSV files, from PDF files from folders. If you put lots of folders inside lots of text documents inside a folder, and then if you read a folder and then change that folders content into corpus that can also create lots of text.

So, it is basically a bunch of text put together in its very raw form is called corpus. Why it is required? Because it reduces size and the moment it reduces size, you can actually analyze the data with faster speed and lots of accuracy. So, I will fast change his data to its corpus corresponding.

So, if it is vector source, because my column with there is a column of data, when for other things this vector source things will change, but now, because they are in vector in law most of the situation in marketing, you will get customer reviews let say and customer reviews will be in a column or our blog reports will be in a column. So, that can be a vector and you can use vector source as your function to convert the vector while screen data to a corpus data.

(Refer Slide Time: 11:00)



So, this is what we are doing the first thing I am creating a docs, so, you check are lots of con things are there in docs. So if I just inspect docs, this is how it looks like. And along this thing, it will be very difficult to inspect the whole thing. So doc is a metadata which is a corpus specific = 1 document level 0. So it is not indexed. And the documents is 45 there are 45 documents. And if I inspect let us first 10 generally, that is what we do because the corpus in real life situation will be very long.

So then I say let us have 1 to 10, I get the, sorry, not this one. So if I just write within docs, yeah. So I get the first 10 documents in this thing. So that is how we create the corpus. Then there are lots of things that we have to do, you have to understand that there are some changes that we have to do.

So we are doing content transformation. So content transformer we are transforming some content. How? So it will be doing this thing that particular function is called two space, what it will do? It will be a basically a function of x and pattern where it will change a pattern with a blank. Whenever you give a source a pattern under blank it will change that in that in the source called x.

It will find out the pattern and corresponding pattern it will replace it with a blank space for example, is a function just a function that we have written for example, let us say doc tm_map and then doc's 2 space and then this. So whenever this particular sign comes up, it will use this to space function here. Okay and that is how it will convert it. So, that means it will convert this whenever this particular backslash is coming up, it will convert it to space.

(Refer Slide Time: 13:25)

```

1 source("code.R")
2 library(devtools)
3 library(roxygen2)
4 library(RCTE)
5
6 filepath <- "http://www.sthda.com/sthda/kboc/example-files/martin-luther-king-i-have-a-dream-speech.txt"
7 text <- readLines(filepath)
8
9
10 test_df <- data.frame(text)
11 write.csv(test_df, "text_df.csv", row.names = FALSE)
12
13 url <- table(text, ffile = "table.txt", sep = "\t", row.names = FALSE)
14
15 text_2 <- read.csv("text_df.csv")
16 print(text_2)
17
18 # Load the docs as a corpus
19 docs <- corpus(Vectorsource(text_2$text))
20
21
22 # Inspect the content of the document
23 inspect(docs)
24 tospace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
25 docs <- tm_map(docs, tospace, "[[:punct:]]")
26 docs <- tm_map(docs, tospace, "[[:punct:]]")
27 docs <- tm_map(docs, tospace, "\\|")
28
29
30 docs <- tm_map(docs, content_transformer(tolower))
31 # Remove numbers
32 docs <- tm_map(docs, removeNumbers)
33 # Remove english common stopwords
34 stopwords
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

Console

```

> inspect(docs)
<- inspect(docs)
<- tm_map(docs, tospace, "[[:punct:]]")
<- tm_map(docs, tospace, "\\|")
<- tm_map(docs, content_transformer(tolower))
<- removeNumbers
<- removeCommonStopwords

```

Preview

```

[1] and so even though we face the difficulties of today and tomorrow, I still have a dream. It is a dream deeply rooted in the american dream.

[2]

[3] I have a dream that one day this nation will rise up and live out the true meaning of its creed:

[4]

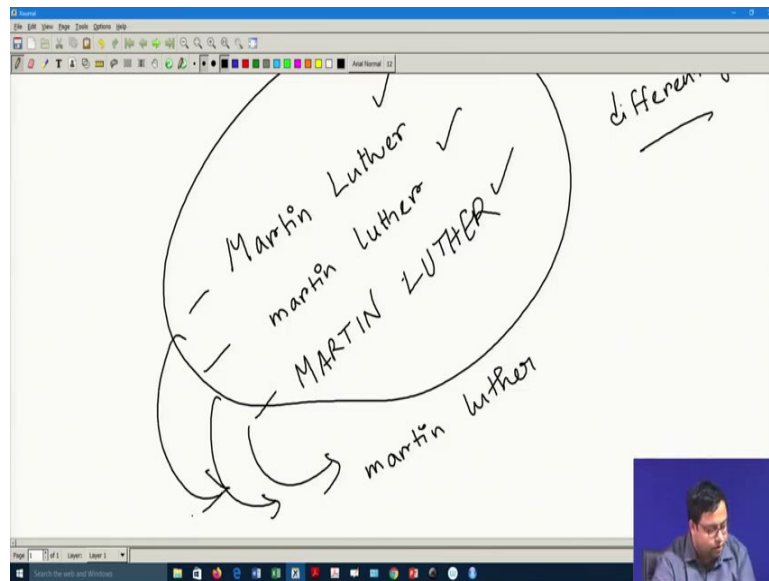
[5] we hold these truths to be self-evident, that all men are created equal.

```

So this is what I am writing, so, convert all this funk punctuations this one and then this one and probably this one also all of them to space admits that they will be blank spaces. Now if I try to inspect, if I just try to inspect you will see that those kind of signs are probably removed from the data and there are specific things that are still there.

So you have to check that when whenever this particular thing was there and now it has got removed. So, that is how we are slowly cleaning the data so that the data can be used for further analysis.

(Refer Slide Time: 14:20)

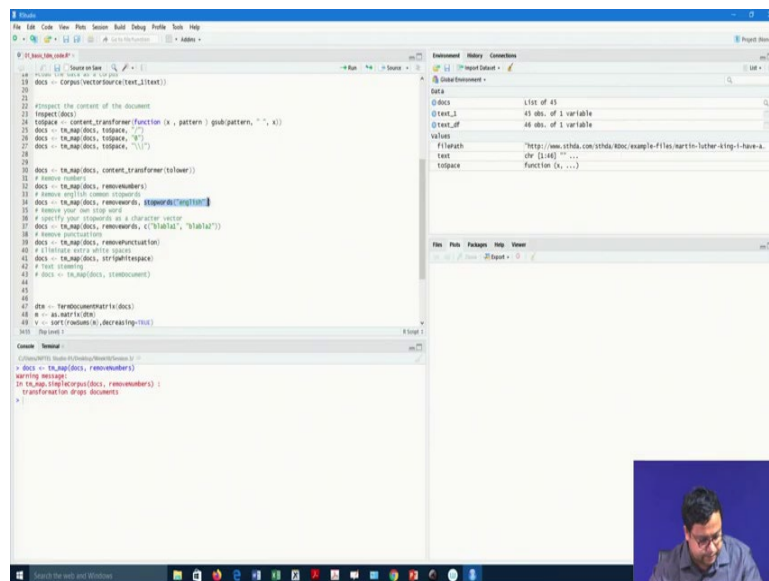


What next? Now, there can be so many different kinds of transformation, why this transformation is needed? Let us say somebody says that there is Martin Luther and some other guys have written martin Luther and then some other guys have written, M. Luther. So ideally these 3 tests will be different or M. Luther might be should be different actually M, I am not saying, let us say this one. So that M. Luther will be different in any way. But let us say MARTIN and LUTHER.

Now, how do you deal with this kind of a situation? You think that see all these 3 are different and if you ask the guy to analyze it, it will analyze it differently. And you have a problem that you do not know that whether you should analyze them differently or not. There are situations which, where you want to analyze them differently.

There are situations where you do not want. If you do not want, then if you want to analyze them as a same thing, then what do you have to do basic thing you have to change all of them to their lower form. So let us say I write it, all of them becomes Martin and then Luther. Something like that. So, to do that, while I have to do is convert them to their lower form. So, that is something that I have to do as that fast.

(Refer Slide Time: 16:12)

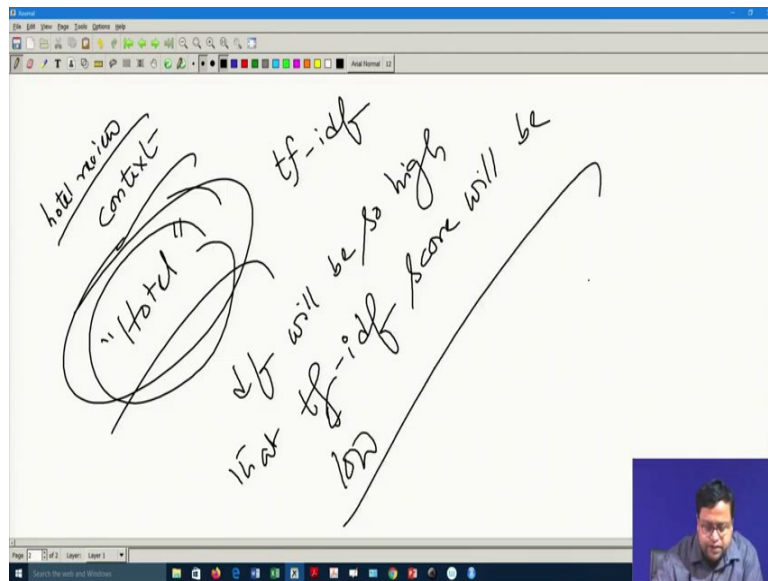


So I am here if you check the text, I am converting them to their lower form, docks = tm map docks content transformer to tool lower. So this is something that I am doing for the first thing done, if I now inspect my content, so let us say dock, if I inspect docks, everything is in lowercase. Everything all the text is in lowercase there is nobody which is in uppercase, then what then there are certain things which are numbers.

So I will just check whether numbers are there or not. In this particular text numbers is not there but number is probably a very standard cleaning process here that we have to follow because sometimes we have numbers, I would say pretty often times, we have numbers here we do not have numbers. But still, if you have to remove numbers, the code is like this doc, tm map docs is comma removed number.

So you pick up docs, that particular corpus, find out the numbers, remove them, and then again, save it in docs. So that is what I am doing. Then what then there is something called stopwords and we are focusing on English language. So stopwords_English suffers within bracket English, so I will just copy that and paste it here.

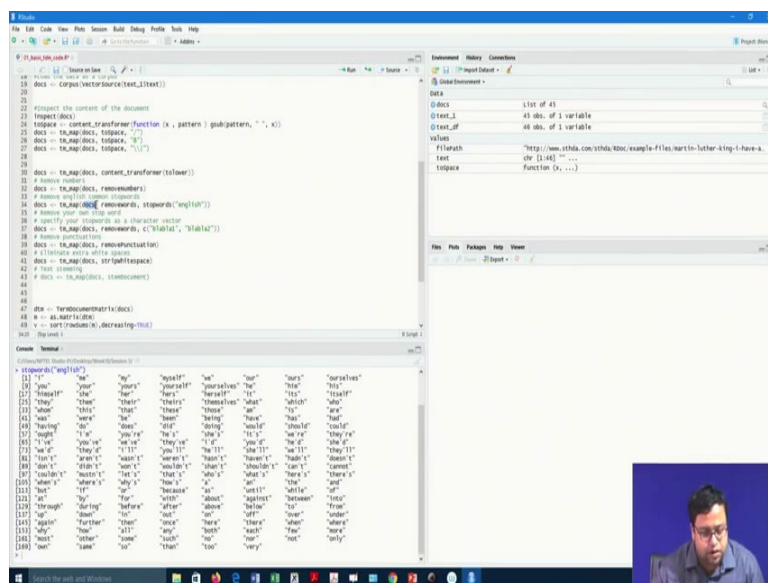
(Refer Slide Time: 18:40)



Now, you can remove stopwords by just mentioning that this some of the cases there can be certain words, which itself are stopwords but for example, if you remember I told in a hotel review context in a hotel review context, one stopwords itself is hotel. This itself is a stopwords.

So, if this itself is a stopwords that that particular thing has to be removed because hotel is very common. If we do tf idf the guy's df will be so high that tf idf score will be much low. So in such kind of a situation, we have to remove them. So this is something that I am doing now.

(Refer Slide Time: 19:29)

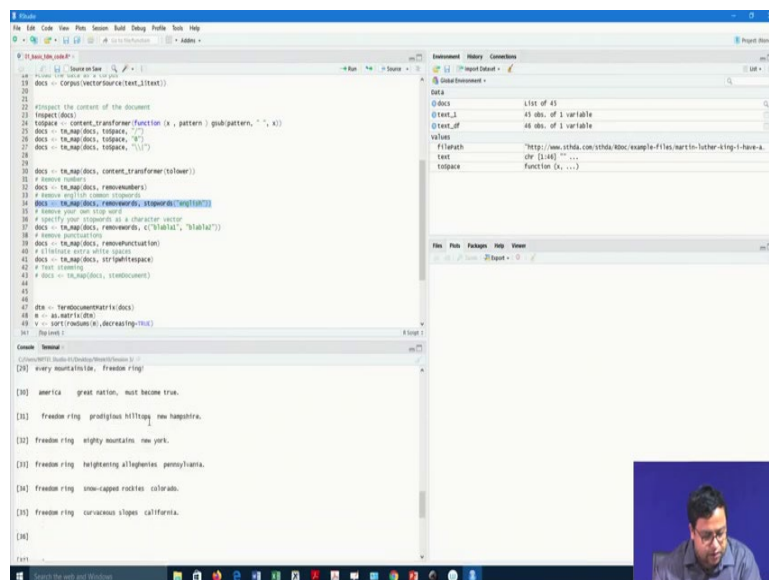


These are my stopwords, and if you want that, okay, there is one more stopword, which is hotel let us say. So you said they say, so remove stopword, tm then docs is your x y from where you will remove the function is remove words and what will you remove stopwords and then this English, so if I just run this in the whole docs, this particular words have been removed. So thank you Almighty free last, free last free last every mountainside let freedom ring.

So, there are certain words you can understand now let, freedom ring every heel. So, let freedom ring from every hill was there to another to heal Mississippi or something like that that kind of to form the all the pronouns all the prepositions conjunctions has been removed. Let us say in this particular thing I also want to remove this let this let is coming very common and let has no meaning for my case. So, let us remove let so, how will I remove let I will just create docs and here instead of stopwords I would write let let has got removed.

Now in my introduction Let Freedom Ring blah blah blah this has also gone way. Now let us say, I want to remove 2 words instead of 1 word, what will I so will let us say will, will be free, will be the day, will be able to, so we will has nothing to do. And then there is another word let us say, which is let us say another word can be will is 1 word it is coming up pretty common, Four will and four this is another one that I will be removing, let us just say that.

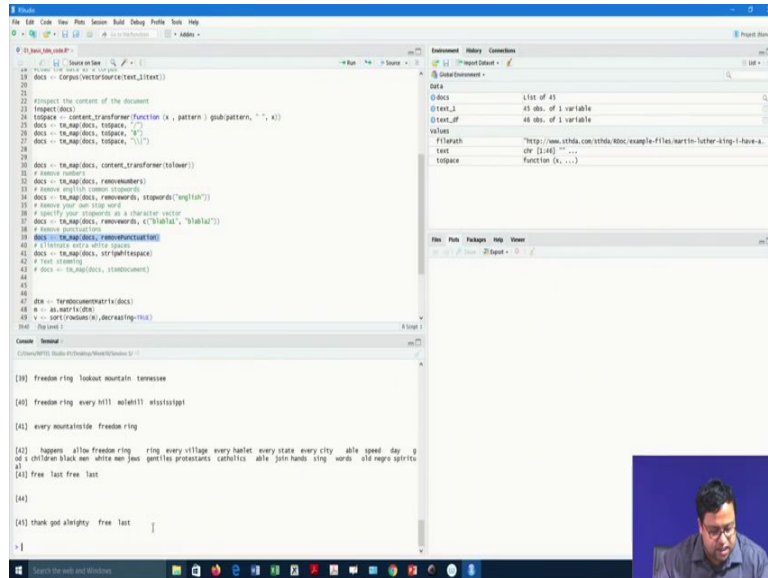
(Refer Slide Time: 21:42)



So then what I will do is, here I will write C will and then four now, if I inspect the docs, the will has gone away and the four has gone away. So all of these things are so, that is how you clean unnecessary words. Now after cleaning necessary words, so see here you see that blah

blah blah and blah blah2, these words are not there, but this is remove it some example has been given here.

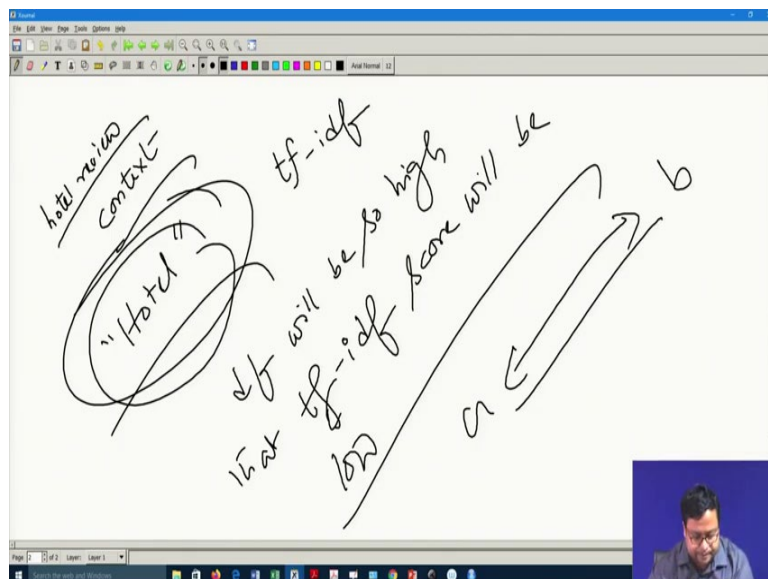
(Refer Slide Time: 22:18)



So then the remove punctuations, so we will punctuation. So, you include just trying that, and then if you inspect the document, all the punctuation in this document has been removed. So, that is what we are doing.

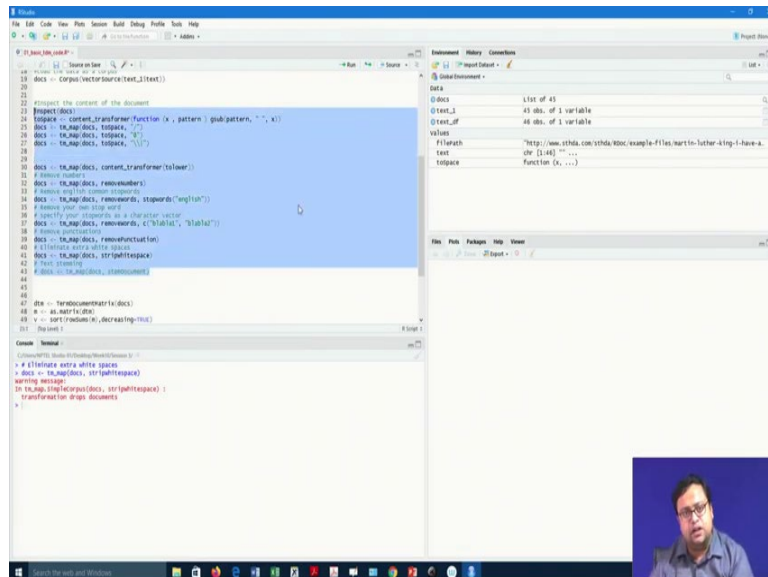
And then there are white spaces also blank spaces. So, tm map docs pre white space, so that will actually reduce remove all the white spaces, not the enters but if you have given tabs in between, let us say a tab b, so, a tab b will create a situation like this.

(Refer Slide Time: 22:50)



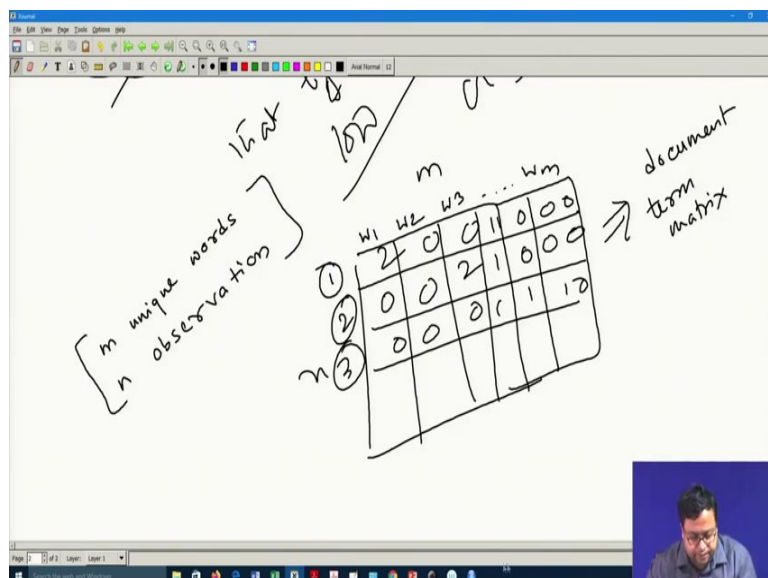
You write a and then a blank and then B so this blank will be removed by whitespace remove whitespace.

(Refer Slide Time: 23:04)



So, after this cleaning this part of to this part is called pre processing of the data after this pre processing, what we try to do is the first thing that I will do is we will create a document term matrix.

(Refer Slide Time: 23:20)

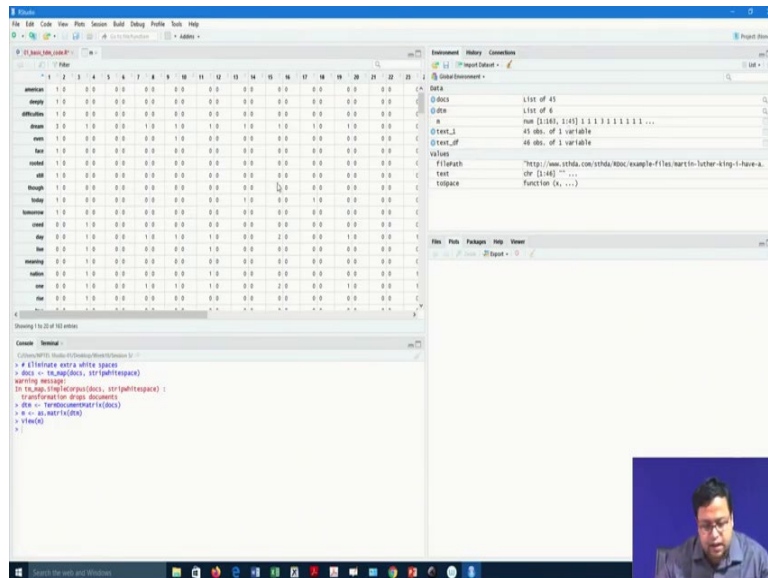


What is document term matrix? Document term matrix is something which looks like this, let us say I have in my in the whole corpus, I have m unique words.

And in observations, that means n number of variables, m number of unique words, and n number of observations. So what we do in this case is a m by n matrix where here each word is written word 1, word 2, word 3, up to word m. So each word is given here and correspondingly, we just see that in observation number 1, whether some of the words has occurred or not or how many times it has occurred.

So, this has occurred 2 times let us say 00100 something 0 and then in a second entry 0021000, the third entry it is 0001110 and the fourth entry and so on these are basically frequencies. So, if this particular matrix is called document term matrix generally we do not convert this matrix to visible matrix form, but we do is a special form of matrix which is document term matrix and I will tell you what that particular thing means.

(Refer Slide Time: 24:52)

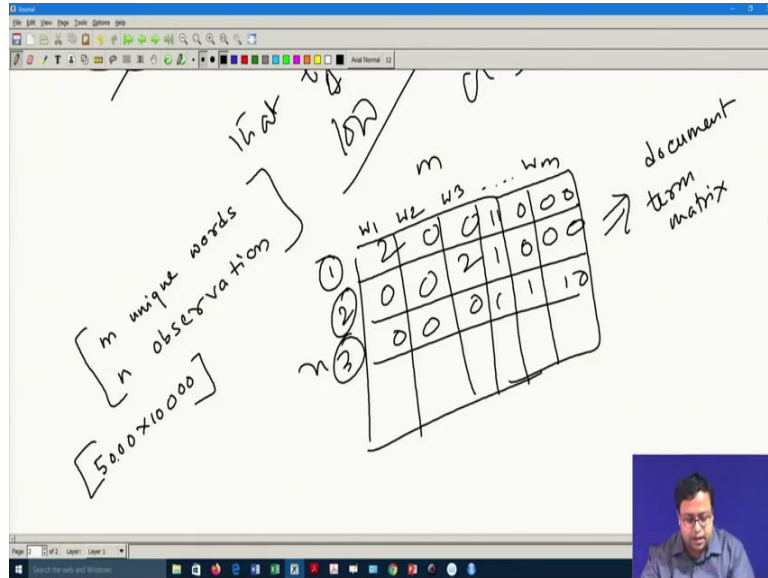


So, this is something and the purpose is also something that I will tell you so, DTM converts is to D document of matrix and DTM is this thing if I by chance convert DTM to its matrix form m, then the M will look like this, see there are I, so if you check here 163 unique text is there unique words are there and this 1 to 45 is done number of observations, so, sorry in the other way it has done in the other way.

So, 163 unique rows are unique words are there, which are rows and 45 columns are there and this particular column is particularly saying that in the first row, which words are occurring, so, 111311111, So, which words are occurring, so if I do the row sums of this particular matrix form that you are seeing, then what you will get is the whether this particular were is the total frequency of that particular word, you will get that, but actually in

document of matrix if you see, what is DTM? DTM gives me the i , the j and the v . See remember, if it is a if there were a number of words with 45 observations, I got 163 unique words.

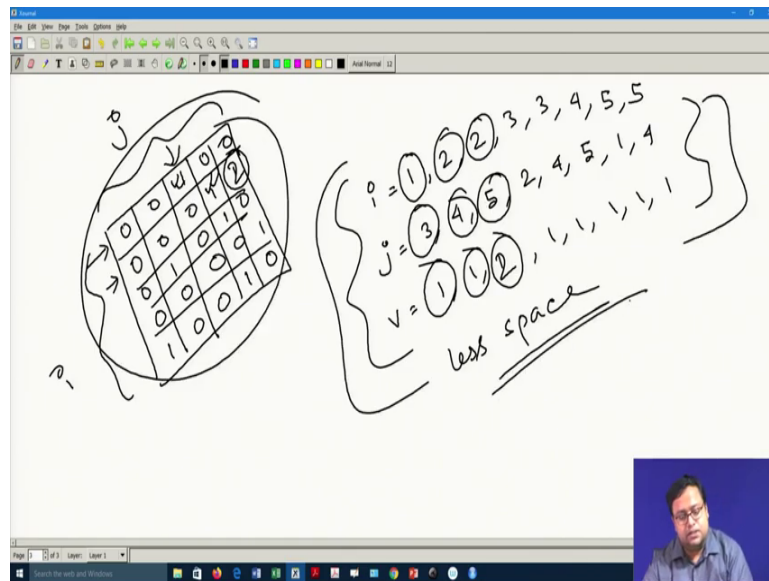
(Refer Slide Time: 26:26)



So now, if I have let us say around our data set of 5000 observations, and how many unique what will I have? I will have probably 10,000 15,000 unique part. So, then $5000 \times 10,000$ matrix will be so, big matrix that that analyzing that will be a difficult thing forget about analyzing, reading that in your art will be a very hectic thing.

So, then also you have to remember that there will be lots of 0 you see that the it will see this particular document or matrix most of the words not occurring in a particular observation in a particular column probably at max 100 words will occur, but even 5000 words. So in a particular observation, maximums will be 0. So there are lots of 0 and some of the positions where these numbers are positive, then why do not why do this?

(Refer Slide Time: 27:20)



So instead of, let us say creating let us I have a five by five matrix 12345 12345, so this is 0100, this is 11000,01010,00001,10010. So this can maximum 0 if you want.

Instead of writing this thing can I write that these are my i's, these are my j's and these are values. So $i = 1$, then $i = 2$, 2 , then 3 , 3 , then 4 , and then 5 5 and $j =$ for the first one, it is 3 , then 4 , then 5 , then two then 4 , then 5 , then 1 then 4 , and the V , which is value is on 1 , or let us this is 2 , so $1,1,2,1,1,1,1,1$ but how do I read this these 3 vectors, how do I read this?

So, the first entry is $i = 1$ $j = 3$ value = 1 . So, first row, third column the value will be 1 , the second is second row, fourth column value will be 1 , the second row, fourth column value is 1 , second row, fifth column values 2 , second rows fifth column there is 2 and that is how you create the matrix and exactly.

Now, this formation this formation of matrix takes less space, you have equally whatever information you have here, you have the same information you have here, but it takes less space.

(Refer Slide Time: 29:17)

The screenshot shows an R script in RStudio. The script performs the following steps:

- Imports the `tm` package.
- Creates a `docs` object from a directory of documents.
- Removes stopwords and non-ASCII characters.
- Creates a `dtm` (Document Term Matrix) from the processed documents.
- Calculates the row sums of the `dtm` to find the most frequent words.
- Sorts the words by frequency and displays the top 10.

The console output shows a table of the top 10 words and their frequencies:

word	freq
mountain	4
shall	4
fat	4
free	4
today	4
her	4
state	4
children	4
little	4
black	4

The screenshot shows an R script in RStudio. The script performs the following steps:

- Imports the `tm` package.
- Creates a `docs` object from a directory of documents.
- Removes stopwords and non-ASCII characters.
- Creates a `dtm` (Document Term Matrix) from the processed documents.
- Calculates the row sums of the `dtm` to find the most frequent words.
- Sorts the words by frequency and displays the top 10.

The console output shows a table of the top 10 words and their frequencies:

word	freq
right	1
sisters	1
vicious	1
crooked	1
exalted	1
flash	1
glory	1
lord	1
low	1
plain	1

So, to do that, this kind of a document of matrix, you see i, j and v that values i_{th} , so that values are given and then you can calculate. So, that is what we have created. Now, I have created the matrix formation of this particular observation also in real life situation this matrix will be so large and you will be not able to see like this, so, the suggestion is do not do this conversion.

Now, the row sums will be the basically the, as I told the row sums will be the value so, if I see that the most common this thing is freedom Martin Luther King, he will talk about freedom, right? When ring the dream, then day one day will come that day, we will do these that day we will do that.

So, day was, so, every morning will be very good and blah, blah blah. So, these kind of words are coming at the top. And then if I convert these two a data frame where frequency is called v and the data is called the data, the data from D looks like this, the words and their corresponding frequency. So, this is called the document term frequency data set.

So, we can create this data set the same thing in a much smaller code also I will show you at a later video that how to do it in a smaller code.

(Refer Slide Time: 30:57)



And now, with this D , but I will do is I will ask you to plot a simple bar plot which looks like this, which is basically talking about which words are common and which words are not common. So, here what I have done till this point is actually we have read the data we cleaned the data pre processing of the data has been done and we created a bar plot to show that which word is most common which one is less common, how to find out the term frequency of the words we have found out.

In the next video, we will try to see that how this kind of concepts can be done and be used for analyzing the data in a better way. Thank you for being with me. I will see you in the next video.