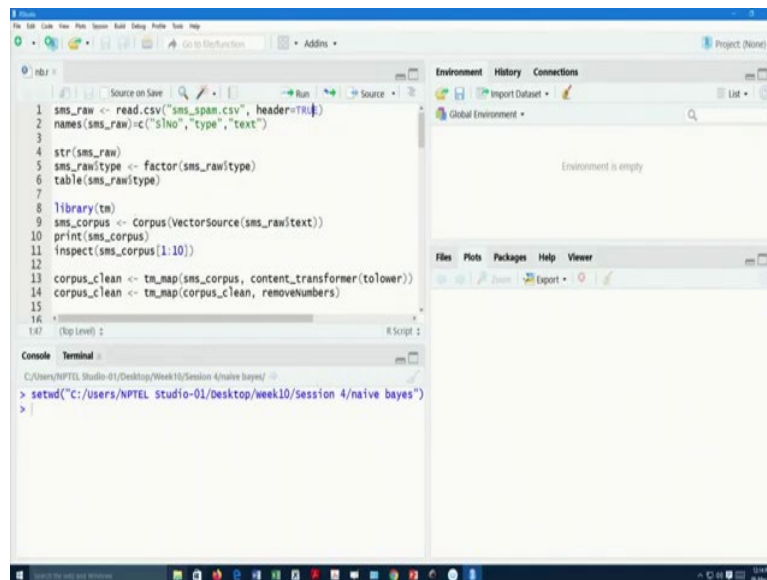


Marketing Analytics
Professor Swagato Chatterjee
Vinod Gupta School of Management,
Indian Institute of Technology, Kharagpur
Lecture 55
Text Mining and Sentiment Analytics (Contd.)

Hello everybody, welcome to Marketing Analytics course. I am Doctor Swagato Chatterjee from VGSOM, IIT, Kharagpur who is taking this course for you. We are in week 10, session 5 and I will discuss about how to use R programming to apply the Naive Bayes algorithm in a data set.

(Refer Slide Time: 00:37)

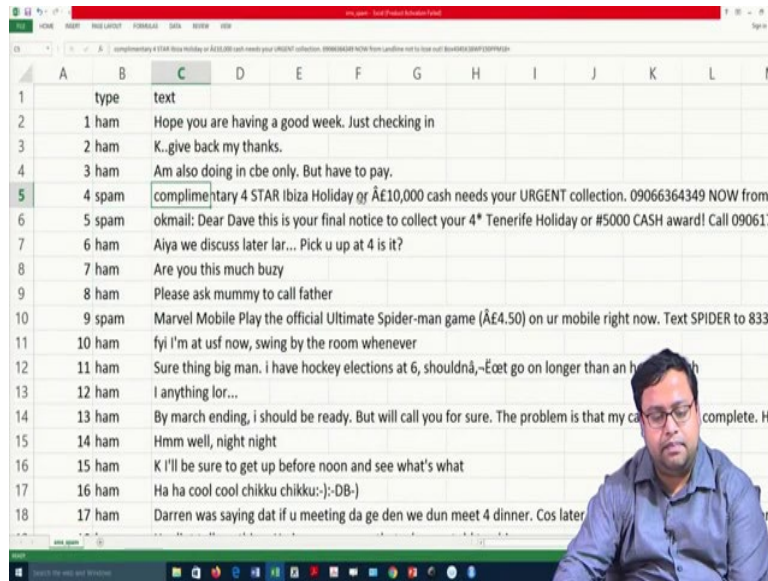


```
1 sms_raw <- read.csv("sms_spam.csv", header=TRUE)
2 names(sms_raw)=c("sno", "type", "text")
3
4 str(sms_raw)
5 sms_raw$type <- factor(sms_raw$type)
6 table(sms_raw$type)
7
8 library(tm)
9 sms_corpus <- Corpus(VectorSource(sms_raw$text))
10 print(sms_corpus)
11 inspect(sms_corpus[1:10])
12
13 corpus_clean <- tm_map(sms_corpus, content_transformer(to_lower))
14 corpus_clean <- tm_map(corpus_clean, remove_numbers)
15
16
17
```

The screenshot shows the R Studio interface. The main editor window contains the R code above. The Environment pane on the right shows 'Global Environment' with 'Environment is empty'. The Console pane at the bottom shows the command prompt with the directory path: `> setwd("C:/Users/NPTEL Studio-01/Desktop/week10/session 4/naive bayes")`.

A data set is given to you, it is the SMS spam data, so we will first put the set working directory and then I will read the data. The data set looks like this.

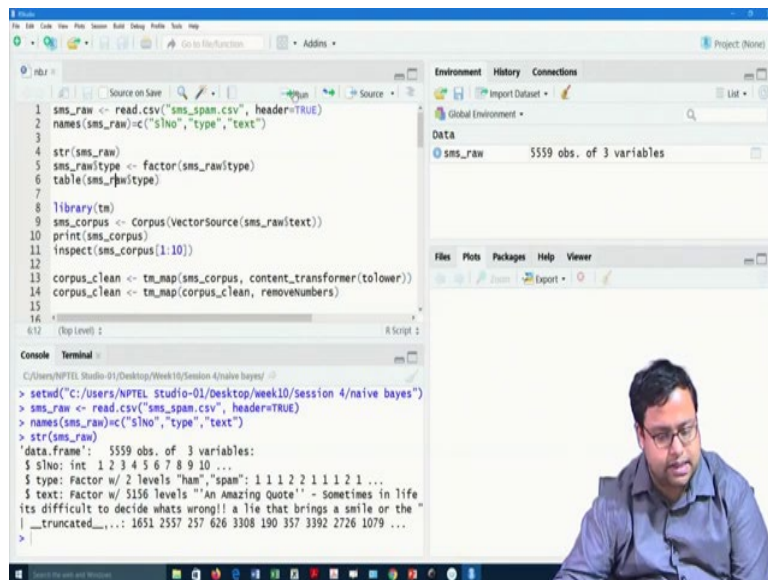
(Refer Slide Time: 00:46)



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		type	text										
2	1	ham	Hope you are having a good week. Just checking in										
3	2	ham	K..give back my thanks.										
4	3	ham	Am also doing in cbe only. But have to pay.										
5	4	spam	complimentary 4 STAR Ibiza Holiday or £10,000 cash needs your URGENT collection. 09066364349 NOW from										
6	5	spam	okmail: Dear Dave this is your final notice to collect your 4* Tenerife Holiday or #5000 CASH award! Call 090617										
7	6	ham	Aiya we discuss later lar... Pick u up at 4 is it?										
8	7	ham	Are you this much buzy										
9	8	ham	Please ask mummy to call father										
10	9	spam	Marvel Mobile Play the official Ultimate Spider-man game (Â£4.50) on ur mobile right now. Text SPIDER to 833										
11	10	ham	fyi i'm at usf now, swing by the room whenever										
12	11	ham	Sure thing big man. i have hockey elections at 6, shouldná, -Ëæt go on longer than an h										
13	12	ham	! anything lor...										
14	13	ham	By march ending, i should be ready. But will call you for sure. The problem is that my ca complete. H										
15	14	ham	Hmm well, night night										
16	15	ham	K i'll be sure to get up before noon and see what's what										
17	16	ham	Ha ha cool chikku chikku:-)-DB-)										
18	17	ham	Darren was saying dat if u meeting da ge den we dun meet 4 dinner. Cos later										

Where if you see it carefully there are the type whether it is spam or ham, and the text is written. For example, complementary 4-star Ibiza holiday, blah-blah-blah, this is spam. But are you this much busy or please ask mummy to call father, this is ham. So, we will be playing with that to see that whether.

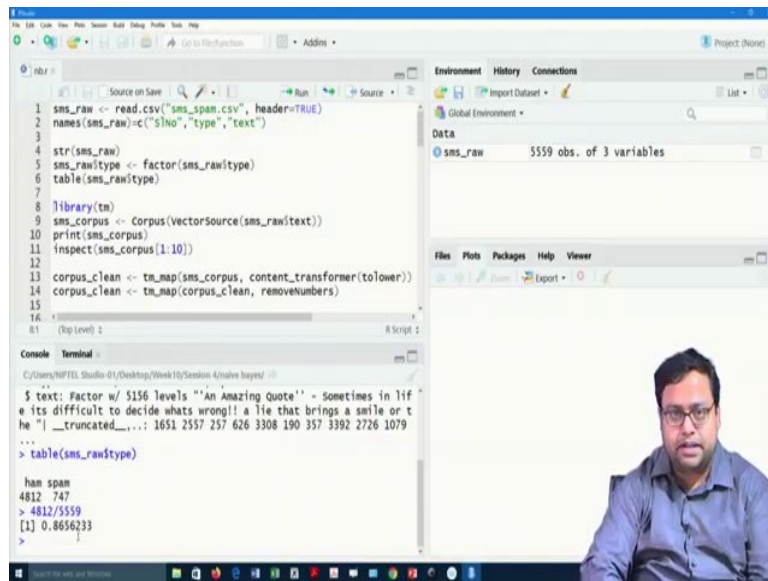
(Refer Slide Time: 01:14)



```
1 sms_raw <- read.csv("sms_spam.csv", header=TRUE)
2 names(sms_raw)=c("sNo", "type", "text")
3
4 str(sms_raw)
5 sms_raw$type <- factor(sms_raw$type)
6 table(sms_raw$type)
7
8 library(tm)
9 sms_corpus <- Corpus(VectorSource(sms_raw$text))
10 print(sms_corpus)
11 inspect(sms_corpus[1:10])
12
13 corpus_clean <- tm_map(sms_corpus, content_transformer(tolower))
14 corpus_clean <- tm_map(corpus_clean, removeNumbers)
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
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
100
```

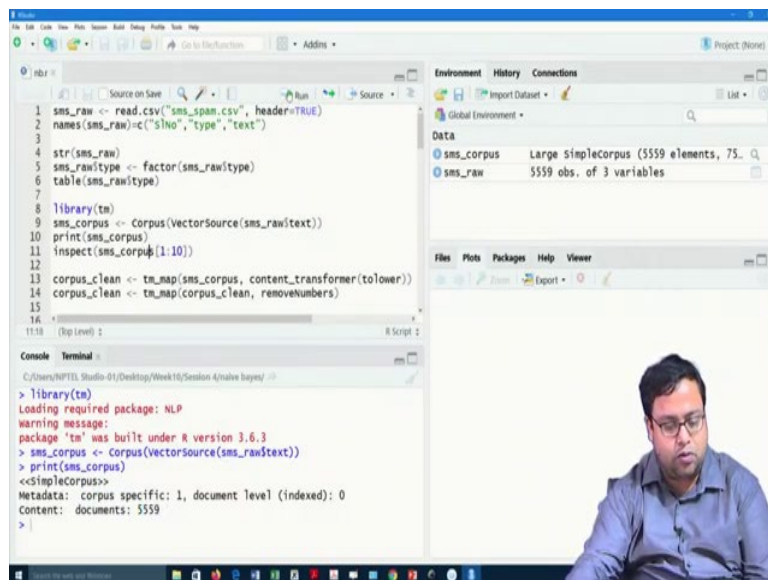
Environment History Connections
Global Environment
Data
sms_raw 5559 obs. of 3 variables
Files Plots Packages Help Viewer

```
> setwd("C:/Users/NPTEL Studio-01/Desktop/week10/session 4/naive bayes")
> sms_raw <- read.csv("sms_spam.csv", header=TRUE)
> names(sms_raw)=c("sNo", "type", "text")
> str(sms_raw)
'data.frame': 5559 obs. of 3 variables:
 $ sNo: int 1 2 3 4 5 6 7 8 9 10 ...
 $ type: Factor w/ 2 levels "ham", "spam": 1 1 1 2 2 1 1 1 2 1 ...
 $ text: Factor w/ 5156 levels "An Amazing Quote" - Sometimes in life
 its difficult to decide whats wrong!! a lie that brings a smile or the
 | _truncated,...: 1651 2557 257 626 3308 190 357 3392 2726 1079 ...
```



So I read the data first and then I put the names properly. The structure of the data looks like this, there are 3 columns, serial number, the type which is factor variable spam and ham, there are two factors and then the text. And then what we do is, if I just check the table that there are 4812 spams so to get an idea there are 4812 / 5559 that means around 86 % ham and then 14 % spam or 86.5 % ham and 13.5 % ham is there, fair enough.

(Refer Slide Time: 01:57)



```
1 sms_raw <- read.csv("sms.csv", as.is = TRUE)
2 names(sms_raw) = c("id", "type", "text")
3
4 str(sms_raw)
5 sms_raw$type <- factor(sms_raw$type)
6 table(sms_raw$type)
7
8 library(tm)
9 sms_corpus <- corpus(VectorSource(sms_raw$text))
10 print(sms_corpus)
11 inspect(sms_corpus[1:10])
12
13 corpus_clean <- tm_map(sms_corpus, content_transformer(to_lower))
14 corpus_clean <- tm_map(corpus_clean, remove_numbers)
15
16 corpus_clean <- tm_map(corpus_clean, remove_words, stopwords())
17
18 # (Tip Level) z
19 # A script z
```

Environment History Connections

Global Environment

DATA

- sms_corpus Large simpleCorpus (5559 elements, 75...
- sms_raw 5559 obs. of 3 variables

Files Plots Packages Help Viewer

Console Terminal

```
C:/Users/NPTEL/Studio-01/Desktop/Week10/Session 4/nave bayes/
> inspect(sms_corpus[1:10])
<<simpleCorpus>>
Metadata: corpus specific: 1,; document level (indexed): 0
Content: documents: 10

[1] Hope you are having a good week. Just checking in

[2] K..give back my thanks.

[3] Am also doing in cbe only. But have to pay.
```

So, now as usual, Library tm, we are calling it changing it to corpus, we have done this before. And then, I will print the corpus at least the, print the corpus says that it has a corpus specific bond, document level zero and that documents are 5559 and if I inspect the first 10 documents, it looks like this, which is basically the first 10 observations.

(Refer Slide Time: 02:26)

```
1 sms_raw <- read.csv("sms.csv", as.is = TRUE)
2 names(sms_raw) = c("id", "type", "text")
3
4 str(sms_raw)
5 sms_raw$type <- factor(sms_raw$type)
6 table(sms_raw$type)
7
8 library(tm)
9 sms_corpus <- corpus(VectorSource(sms_raw$text))
10 print(sms_corpus)
11 inspect(sms_corpus[1:10])
12
13 corpus_clean <- tm_map(sms_corpus, content_transformer(to_lower))
14 corpus_clean <- tm_map(corpus_clean, remove_numbers)
15
16 corpus_clean <- tm_map(corpus_clean, remove_words, stopwords())
17
18 # (Tip Level) z
19 # A script z
```

Environment History Connections

Global Environment

DATA

- sms_corpus Large simpleCorpus (5559 elements, 75...
- sms_raw 5559 obs. of 3 variables

Files Plots Packages Help Viewer

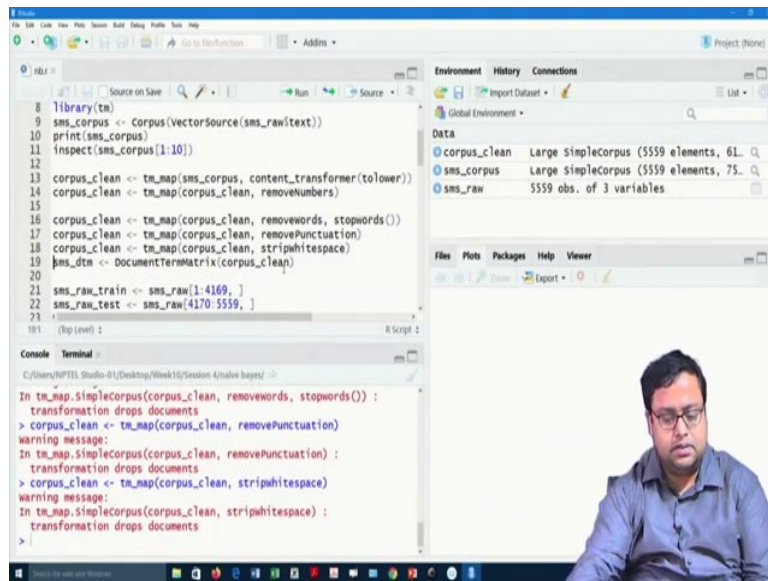
Console Terminal

```
C:/Users/NPTEL/Studio-01/Desktop/Week10/Session 4/nave bayes/
[8] Please ask nummy to call father

[9] Marvel Mobile Play the official Ultimate Spider-man game (A£4.50)
on ur mobile right now. Text SPIDER to 83338 for the game & we ll sen
d u a FREE 8ball wallpaper

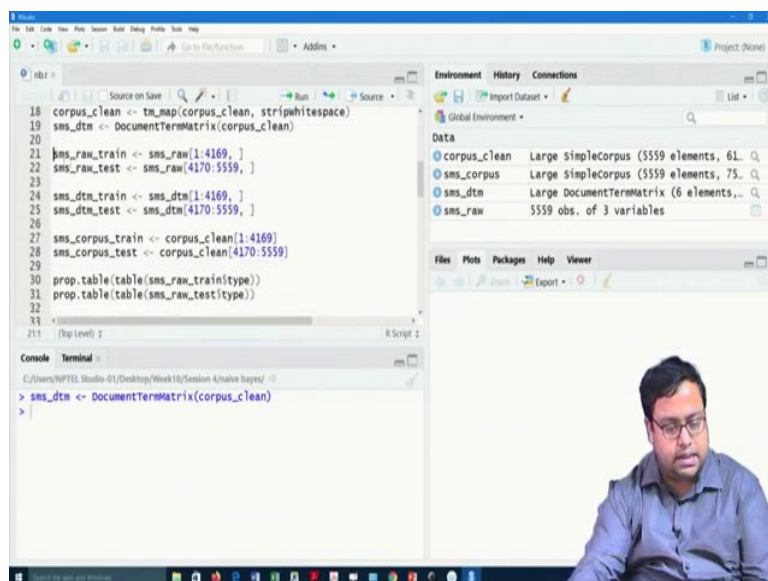
[10] fyi I'm at usf now, swing by the room whenever

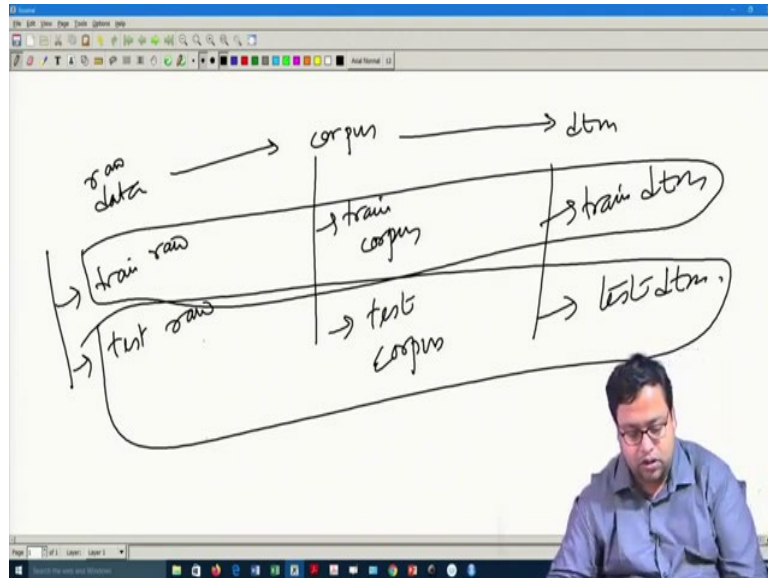
>
```



Now, with this what I do is I transform it to their lower, I change, remove the numbers, I remove the stop words, I remove punctuation, we have done this before, right in session 2, if we are not wrong, session 2 or 3 I have done this. So, strip white space, so we are removing this and now with this thing I am creating a document on matrix.

(Refer Slide Time: 02:57)

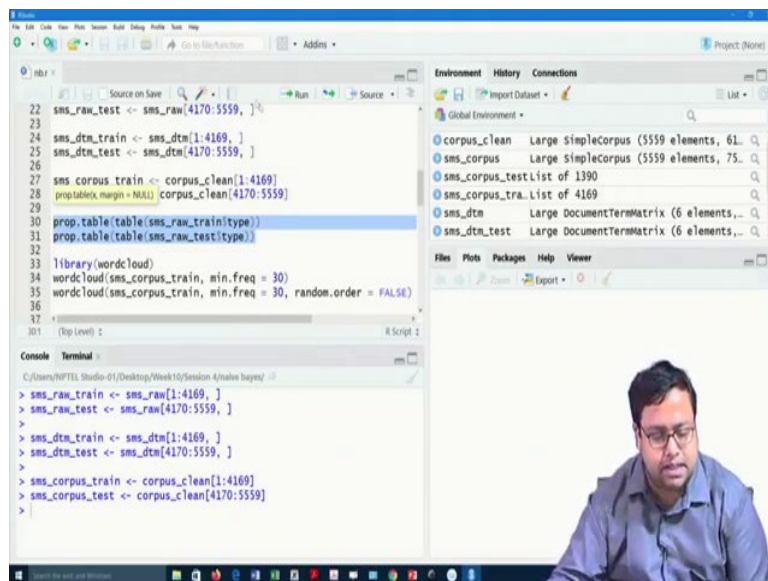




So, SMS DTM is a large matrix. Now, next what I am doing here is there are some purpose of doing this. So, there are 3 things, you check, there is a raw data, from there we created the corpus, we clean the corpus and created data documental matrix. Now I will break this raw data into training raw and testing raw. Training corpus the same, exactly same row numbers, its training corpus and testing corpus. And training DTM and testing DTM, this is something that I am going to create.

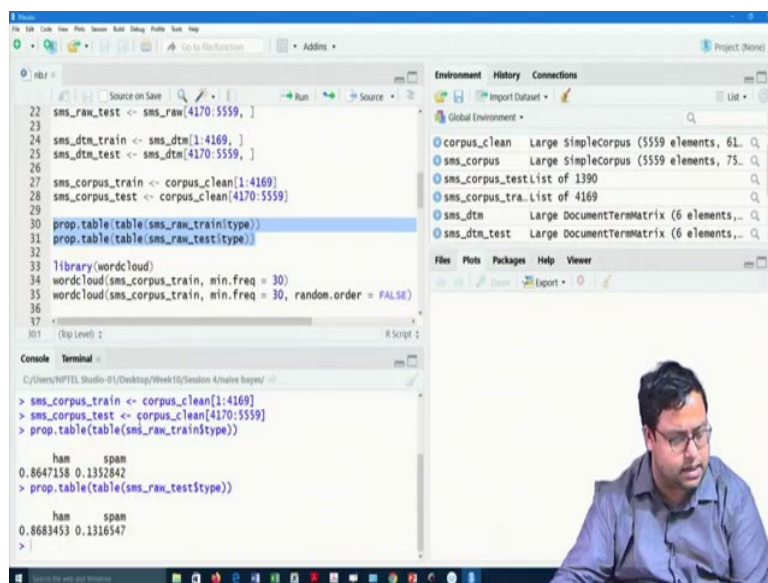
Why? Because some of this raw one will be used to create word cloud. Corpus might be used to something else. DTM might be used to create something else so there is certain purpose. But there should be some matching so that you see, these three guys should be related to each other and these three guys should be related to each other.

(Refer Slide Time: 04:09)



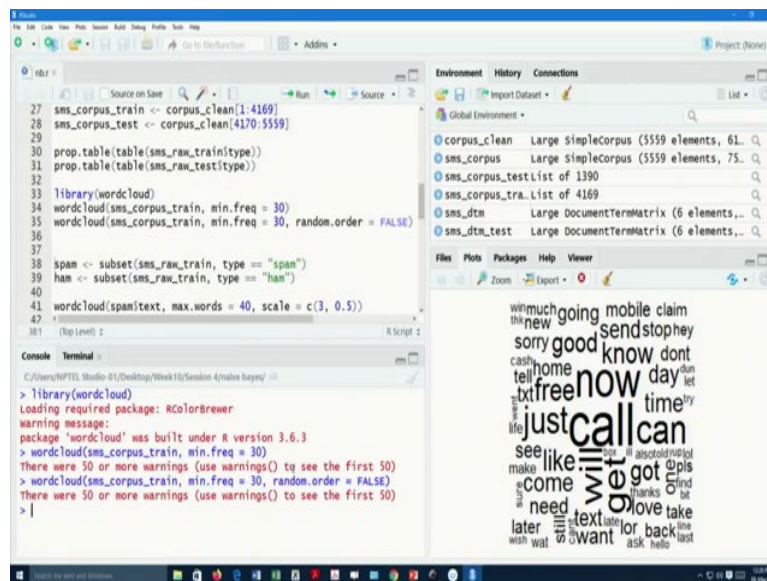
So, I am using the same column numbers, row numbers 1 to 4169, and 4170 to 5599 so, I am just, so if I just do it 4169 / 5559. That means around 75 % goes to my train and 25 % goes to my test. So, I am breaking this thing. So, the first one is the raw data, next is DTM and third is corpus. Fair enough. I have broken them. Why have I broken them? So, after breaking them I have to just see that whether I have not done randomness, see. So, I am just checking that whether even the training and testing data the I would say distribution of spams and hams are similar or not.

(Refer Slide Time: 04:51)



If you remember, earlier it was 86.5 % ham in the training data it is 86.47 similar. Here is it 86.8 one and the similar. So, I can deal with this thing.

(Refer Slide Time: 05:05)



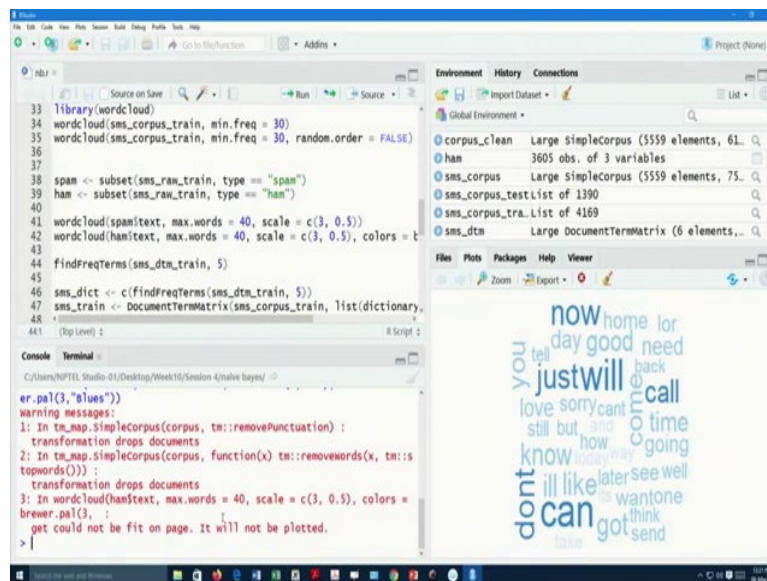
So, now with this dataset, what I do is next is I call the library, the library is word cloud. I am calling the library and with the corpus, I am trying to draw a, so you see this is the corpus of my training data. This is how a word cloud looks like.

This is a word cloud this is how exactly it looks like. So, word cloud, let us say the, this guy is the bigger one, the bigger the one, the bigger the word in word cloud the more frequency of that word is there. Now, if I just random one is equal to false if I do that, what that will do, so before you plot the next plot, it is better to clean this plotting area. So, what it will do is, it will put the important ones at the middle.

So, you see call, now, just, will, can, these are the. Now what is my purpose? My purpose is using words to analyse whether it is spam or ham. So by that logic exploratorily or visually if I see that the words which are coming common in spam and the words which are coming common in ham if that words are different in other words, the word cloud of a spam message and word cloud of all the ham message, if the prominent words are different then only I can say that words are the important variable to predict whether it is spam or ham.

Understand carefully what I am saying once more, I am saying that I can use words as my predicting variable for spam or ham only when the word cloud is giving me different sets of words for spam or ham that is the only case.

(Refer Slide Time: 07:06)

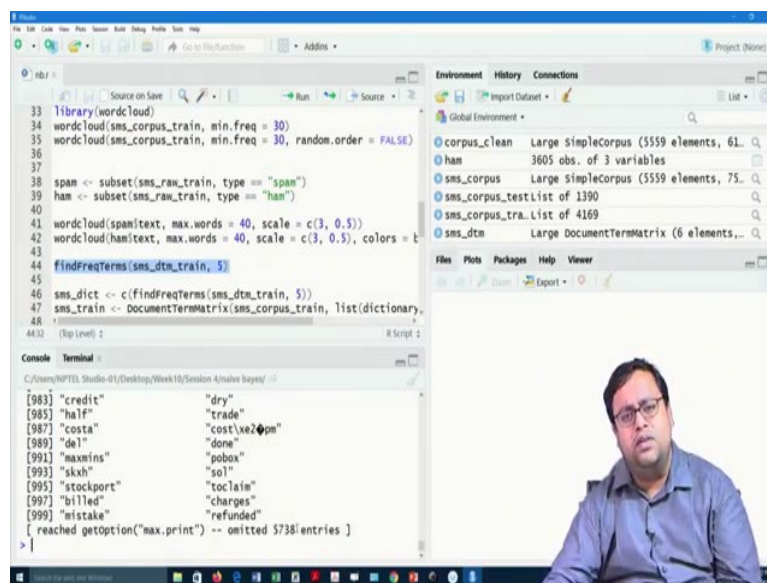


So, what I do next is, I divide the dataset into spam and ham. Okay, so subset the raw data into spam and ham and now with once I will draw it with spam and then I will draw it with ham.

So, I have drawn it with spam, okay so let me draw it once more, it did not come properly, so I will draw it with spam and spam if you see carefully, the words are coming up call, free, now, mobile So, some words just remember the claim, just remember this words and for ham, the words that are coming up are just, will, do not, can, so call and now are same but claim or let us say mobile, these words are going away.

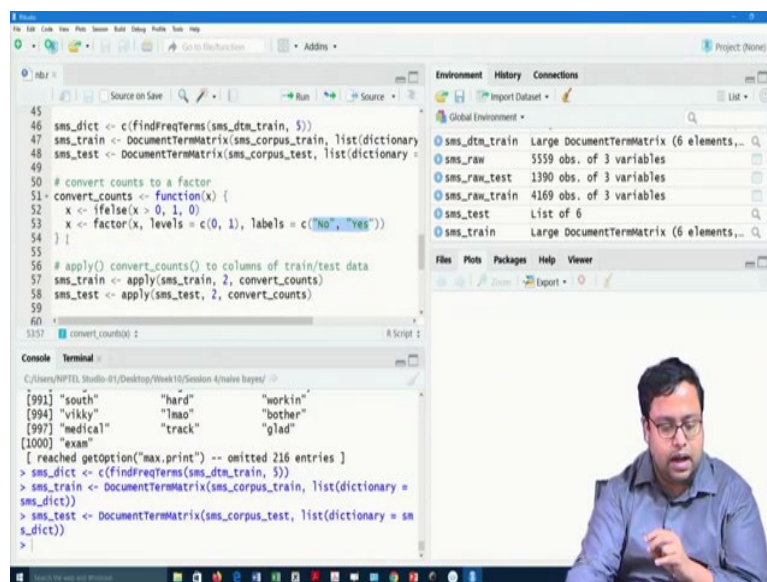
So, there is some difference between the words that are common. So, that suggest that words can be one variable to use that. I have also shown how to create very beautiful word clouds which can be used for your reporting purpose. Now, imagine how many words are there?

(Refer Slide Time: 08:31)



```
library(wordcloud)
wordcloud(sms_corpus_train, min.freq = 30)
wordcloud(sms_corpus_train, min.freq = 30, random.order = FALSE)
span <- subset(sms_raw_train, type == "spam")
ham <- subset(sms_raw_train, type == "ham")
wordcloud(span$text, max.words = 40, scale = c(3, 0.5))
wordcloud(ham$text, max.words = 40, scale = c(3, 0.5), colors = t
findFreqTerms(sms_dtm_train, 5)
sms_dict <- c(findFreqTerms(sms_dtm_train, 5))
sms_train <- DocumentTermMatrix(sms_corpus_train, list(dictionary,
```

```
[983] "credit"      "dry"
[985] "half"         "trade"
[987] "costa"        "cost\xe2\x20pm"
[989] "de1"          "done"
[991] "maxmins"     "pobox"
[993] "skxh"        "sol"
[995] "stockport"   "toclaim"
[997] "billed"      "charges"
[999] "mistake"     "refunded"
[reached getOption("max.print") -- omitted 5738 entries ]
```



```
sms_dict <- c(findFreqTerms(sms_dtm_train, 5))
sms_train <- DocumentTermMatrix(sms_corpus_train, list(dictionary
sms_test <- DocumentTermMatrix(sms_corpus_test, list(dictionary
# convert counts to a factor
convert_counts <- function(x) {
  x <- ifelse(x > 0, 1, 0)
  x <- factor(x, levels = c(0, 1), labels = c("No", "Yes"))
}
# apply() convert_counts() to columns of train/test data
sms_train <- apply(sms_train, 2, convert_counts)
sms_test <- apply(sms_test, 2, convert_counts)
```

```
[991] "south"      "hard"      "workin"
[994] "vikky"     "lmao"      "bother"
[997] "medical"   "track"     "glad"
[1000] "axax"
[reached getOption("max.print") -- omitted 216 entries ]
> sms_dict <- c(findFreqTerms(sms_dtm_train, 5))
> sms_train <- DocumentTermMatrix(sms_corpus_train, list(dictionary =
sms_dict))
> sms_test <- DocumentTermMatrix(sms_corpus_test, list(dictionary = sm
s_dict))
```

So if I just find out, find frequent words and then put it one here it will find out all those words which has a frequency of at least 1. And there are 5738 words are there. So, sorry not six thousand, thousand are printed and omitted are 5738 so total 6738 words are there. I mean there is no point on playing with so many words. If they are occurring only once, why will have an importance on them. So, what I do is, I find out only those words which have up to 5 frequency and those words are 1 2 1 6, 1216 words.

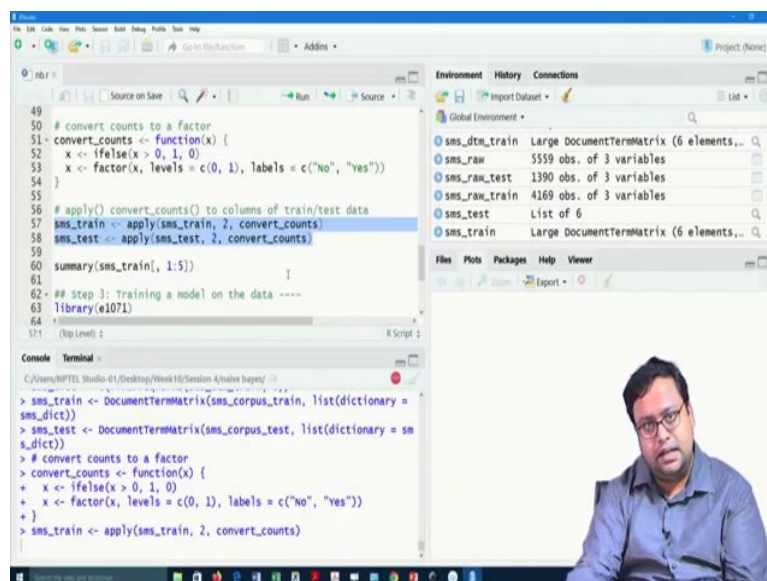
So, what I do next is put these words in a dictionary and then again create this document from matrix training and testing, whichever I have created here, this document on matrix DTM, training and testing. I am recreating them but now with the dictionary of this data, not the whole set of words.

So, I am creating a document on matrix, if you remember document on matrix was the terms in one side, the document numbers and their corresponding counts, Counts okay, not whether it occurred or not. How many times it occurs was there? Now, remember in my Naive Bayes algorithm when I taught Naïve and Bayes equation it was just telling Viagra was there or not? Yes or no.

Grocery was there or not? Yes or no. Unsubscribe word was there or not? Yes or no. it was not talking about whether that word is occurring 5 times in a particular row or 2 times in a particular row. The frequency was not there. The only thing that was a variable was the occurrence whether it occurred or not. So, I have to also do something to change this document on practice which is frequency oriented to a occurrence oriented.

So, I am creating a function called convert counts which takes a value of x, if the x's value is > 0 , any positive value, it will make it 1, otherwise 0. And then 0 and 1, it is leveling at as no and yes.

(Refer Slide Time: 10:42)



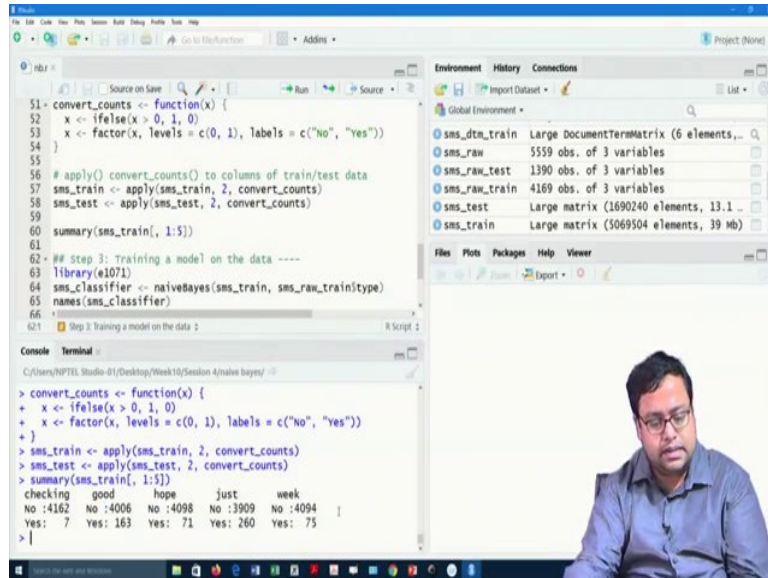
```
49
50 # convert counts to a factor
51 convert_counts <- function(x) {
52   x <- ifelse(x > 0, 1, 0)
53   x <- factor(x, levels = c(0, 1), labels = c("No", "Yes"))
54 }
55
56 # apply() convert_counts() to columns of train/test data
57 sms_train <- apply(sms_train, 2, convert_counts)
58 sms_test <- apply(sms_test, 2, convert_counts)
59
60 summary(sms_train[, 1:5])
61
62 ## step 3: Training a model on the data ----
63 library(e1071)
64
65 (Rip Level) :
```

Environment History Connections

- Global Environment
- sms_dtm_train Large DocumentTermMatrix (6 elements, ...)
- sms_raw 5539 obs. of 3 variables
- sms_raw_test 1390 obs. of 3 variables
- sms_raw_train 4169 obs. of 3 variables
- sms_test List of 6
- sms_train Large DocumentTermMatrix (6 elements, ...)

Files Plots Packages Help Viewer

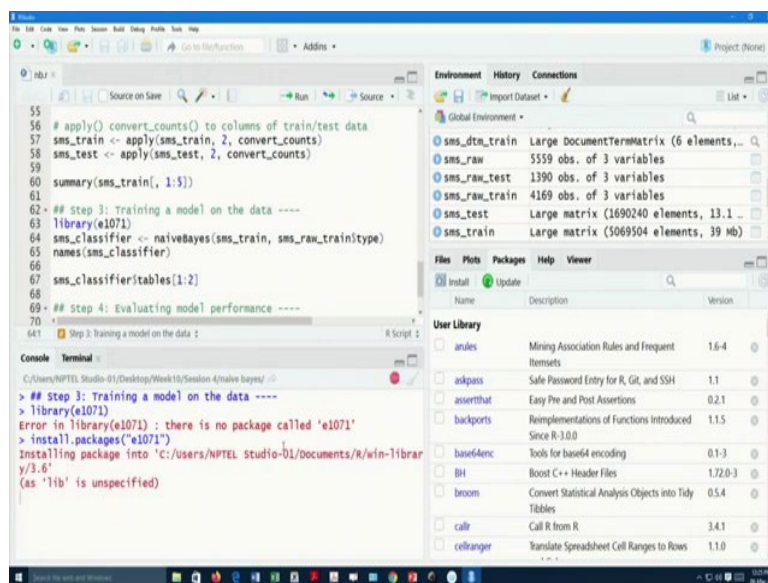
```
> sms_train <- DocumentTermMatrix(sms_corpus_train, list(dictionary = sms_dict))
> sms_test <- DocumentTermMatrix(sms_corpus_test, list(dictionary = sms_dict))
> # convert counts to a factor
> convert_counts <- function(x) {
+   x <- ifelse(x > 0, 1, 0)
+   x <- factor(x, levels = c(0, 1), labels = c("No", "Yes"))
+ }
> sms_train <- apply(sms_train, 2, convert_counts)
```



So, that is how I am writing a function and then I will use this function on my training and testing DTM document on matrix, to convert them to this kind of a thing. So, if I just run this, it will take little bit of time. If I just run this one and convert ten to ES no basically instead of the counts, the training one has been done but testing one also has been done, the summary of the training one looks like this.

So, this are the word, first 5 words, first 5 randomly chosen, no, no it is not like the first 5 words based on the frequency or something. So, this word is occurring, not occurring for 162 times, occurring 7 times. Would is occurring 163 times and not occurring 4006 n times and so on. So that is how I have created the summary.

(Refer Slide Time: 11:46)



```

55 # apply() convert_counts() to columns of train/test data
56 sms_train <- apply(sms_train, 2, convert_counts)
57 sms_test <- apply(sms_test, 2, convert_counts)
58
59 summary(sms_train[, 1:5])
60
61 ## Step 3: Training a model on the data ----
62 library(e1071)
63 sms_classifier <- naiveBayes(sms_train, sms_raw_train$type)
64 names(sms_classifier)
65
66 sms_classifier$tables[1:2]
67
68 ## Step 4: Evaluating model performance ----
69
70

```

Environment History Connections

Name	Description	Version
sms_dtm_train	Large DocumentTermMatrix (6 elements, ...)	
sms_raw	5559 obs. of 3 variables	
sms_raw_test	1390 obs. of 3 variables	
sms_raw_train	4169 obs. of 3 variables	
sms_test	Large matrix (1690240 elements, 13.1 ...)	
sms_train	Large matrix (5069504 elements, 39 Mb)	

User Library

Name	Description	Version
arules	Mining Association Rules and Frequent Itemsets	1.6-4
askpass	Safe Password Entry for R, Git, and SSH	1.1
assertthat	Easy Pre and Post Assertions	0.2.1
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.5
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.72.0-3
broom	Convert Statistical Analysis Objects into Tidy Tables	0.5.4
callr	Call R from R	3.4.1
cellranger	Translate Spreadsheet Cell Ranges to Rows	1.1.0

```

56 # apply() convert_counts() to columns of train/test data
57 sms_train <- apply(sms_train, 2, convert_counts)
58 sms_test <- apply(sms_test, 2, convert_counts)
59
60 summary(sms_train[, 1:5])
61
62 ## Step 3: Training a model on the data ----
63 library(e1071)
64 sms_classifier <- naiveBayes(sms_train, sms_raw_train$type)
65 names(sms_classifier)
66
67 sms_classifier$tables[1:2]
68
69 ## Step 4: Evaluating model performance ----
70 sms_test_pred <- predict(sms_classifier, sms_test)
71

```

Environment History Connections

Name	Description	Version
sms_classifier	Large naiveBayes (5 elements, 1.7 Mb)	
sms_corpus	Large SimpleCorpus (5559 elements, 75...)	
sms_corpus_test	List of 1390	
sms_corpus_train	List of 4169	
sms_dtm	Large DocumentTermMatrix (6 elements, ...)	
sms_dtm_test	Large DocumentTermMatrix (6 elements, ...)	

Console Terminal

```

C:\Users\NPTEL_Studio-01\Desktop\Week10\Session 4\Naive Bayes/ >
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/e1071_1.7-3.zip'
content type 'application/zip' length 1022345 bytes (998 kB)
downloaded 998 kB

package 'e1071' successfully unpacked and MD5 sums checked

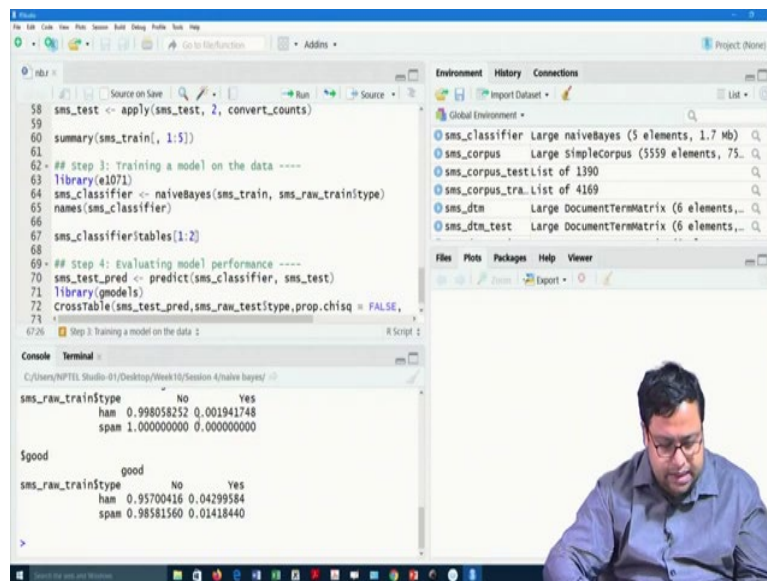
The downloaded binary packages are in
  c:\Users\NPTEL_Studio-01\AppData\Local\Temp\rtmpuiv5wv\downloaded_packages
>
> # Step 3: Training a model on the data ----
> library(e1071)
warning message:
package 'e1071' was built under R version 3.6.3
> sms_classifier <- naiveBayes(sms_train, sms_raw_train$type)
> names(sms_classifier)
[1] "apriori" | "tables" | "levels" | "isnumeric" "call"
>

```

Now what will I do? I will run a Naive Bayes now, the library is E 1071, there is no such package, so I have to install the package. So, E 1071 is the package name. I am installing it. Okay, it has been installed I am calling the library now. It has been called, there is a warning sign. Well do not care the warning sign probably and the function is Naive Bayes but the DTM is by x variable and the y variable is the SMS raw train, the raw data, raw training data, type variable is my basically the y variable.

So, now if I just run this thing, Naive Bayes, it will take some time to train, it is training right now. It will take some time. Okay, that is trained. And if I just want to see the tables.

(Refer Slide Time: 12:51)



```
58 sms_test <- apply(sms_test, 2, convert_counts)
59
60 summary(sms_train[, 1:5])
61
62 ## Step 3: Training a model on the data ----
63 library(e1071)
64 sms_classifier <- naiveBayes(sms_train, sms_raw_train$type)
65 names(sms_classifier)
66 sms_classifier$tables[1:2]
67
68 ## Step 4: Evaluating model performance ----
69 sms_test_pred <- predict(sms_classifier, sms_test)
70 library(models)
71 crossTable(sms_test_pred, sms_raw_test$type, prop.chisq = FALSE,
72           )
73
```

Console Terminal

```
C:/Users/HPTEL/Studio-01/Desktop/Week 10/Session 4/naive bayes/ >
sms_raw_train$type      No      Yes
ham 0.998058252 0.001941748
spam 1.000000000 0.000000000

$good
sms_raw_train$type      No      Yes
ham 0.95700416 0.04299584
spam 0.98581560 0.01418440
```

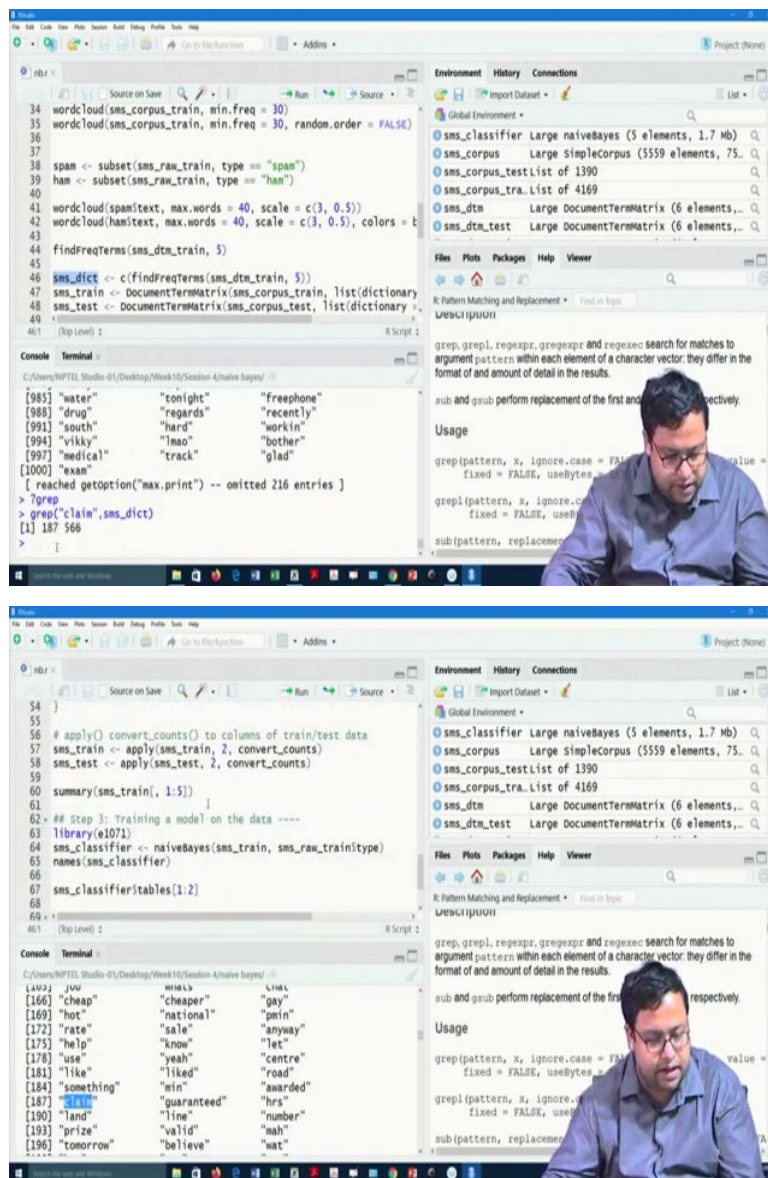
So, the first two tables, it will create, if there are 4 words, if you remember, there were 4 tables. For Viagra word there was 1 table, and there was Viagra, Unsubscribe, Grocery and so Money, there were 4 tables.

Here I have taken 1216 words, top words which has at least 5 frequency. So, there will be 1216 these things. And I have taken only first 2. You can you can choose which word you want to choose. So, if you see carefully, that when there is ham, given it is ham, the probability of checking occurring is 0.99 and checking occurring is 0.002 and checking not occurring is 0.998.

And given it is spam, its always is it is 0 and it is 1. Similarly given it is spam, the probability often is a numerator, so P of n given s, remember. So, given it is spam, the probability of this occurring is 0.1 and 0.98. So, there is some distance here, 0.01 and 0.04, some distance here but there is probably very lower distance here.

If the 0.000 and 0.002, so we have to check for different, so if you remember there was a word called, there was a word called mobile or there was a word called, claim. So, let us find out what the word thing is? Okay, so I will just quickly find out the position of the word called claim. So, how will I find out that?

(Refer Slide Time: 14:32)



So, find frequent terms or SMS DICT. So claim, wait, grep, grep function is written like this, grep so the help will tell me that how pattern and then x, okay so grep, then the pattern is claimed and what I am searching SMS underscore dict. So, if it is 187 okay so 187 and 566. So, if I just check 566 word, 566 is claimed to claim, so that is not so important word, but 187, 187 is the word called claim.

So, I will just check this particular table whatever I created for 187, word number 187 let us see, and then you will understand why I am saying this.

(Refer Slide Time: 15:46)

```
summary(sms_train[, 1:5])
## Step 3: Training a model on the data ----
library(e1071)
sms_classifier <- naiveBayes(sms_train, sms_raw_train$type)
names(sms_classifier)
sms_classifier$tables[1:2]
## Step 4: Evaluating model performance ----
sms_test_pred <- predict(sms_classifier, sms_test)
library(models)
crossTable(sms_test_pred, sms_raw_test$type, prop.chisq = FALSE,
           dnn = c('predicted', 'actual'))
```

```
[1000] "exam"
[ reached getOption("max.print") -- omitted 216 entries ]
> ?grep
> grep("claim", sms_dict)
[1] 187 366
Error in gsub(restrip, "", completions, perl = TRUE) :
  input string 558 is invalid UTF-8
> summary(sms_train[, 187])
  Length class      Mode
 4169 character character
> sms_classifier$tables[187]
```

```
4169 character character
Error in gsub(restrip, "", completions, perl = TRUE) :
  input string 558 is invalid UTF-8
> sms_classifier$tables[187]
$claim
      claim
sms_raw_train$type No Yes
ham 1.0000000 0.0000000
spam 0.8457447 0.1542553
```

So, if I just write 187, this is the claim word, summary of SMS train, see 187. Oh no, not sorry, no, no, sorry-sorry, not summary, this one, not summary, this one and if I just write 187 here, you see that for claim, when it is spam, the probability is 0.5, that the claim word will be there.

Ham is 0.84 and when it is ham the probability is 0 and 1 so there is a huge distance 0.00 and 0.15, 15 % distance is a very high distance. So, that is something that we are trying to find out here.

(Refer Slide Time: 16:42)

```
59 summary(sms_train[, 1:5])
60
61 ## Step 3: Training a model on the data ----
62 library(e1071)
63 sms_classifier <- naiveBayes(sms_train, sms_raw_train$type)
64 names(sms_classifier)
65
66 sms_classifier$tables[1:2]
67
68
69 ## Step 4: Evaluating model performance ----
70 sms_test_pred <- predict(sms_classifier, sms_test)
71 library(gmodels)
72 crosstable(sms_test_pred, sms_raw_test$type, prop.chisq = FALSE,
73           dnm = c("predicted", "actual"))
74
75
76 Step 4: Evaluating model performance : R script
```

Console Terminal

```
C:\Users\NPTEL Studio-01\Desktop\Week10\Session 4\naive bayes/ >
> ## Step 4: Evaluating model performance ----
> sms_test_pred <- predict(sms_classifier, sms_test)
> library(gmodels)
Error in library(gmodels) : there is no package called 'gmodels'
> crosstable(sms_test_pred, sms_raw_test$type, prop.chisq = FALSE,
+           dnm = c("predicted", "actual"))
Error in Crosstable(sms_test_pred, sms_raw_test$type, prop.chisq = FALSE
+ :
+ could not find function "Crosstable"
>
```

Environment

Object	Class	Attributes
sms_test	Large matrix	(1690240 elements, 13.1 MB)
sms_train	Large matrix	(5069504 elements, 39 MB)
span	Factor	564 obs. of 3 variables

User Library

Package	Description	Version
arules	Mining Association Rules and Frequent Itemsets	1.6-4
askpass	Safe Password Entry for R, Git, and SSH	1.1
assertthat	Easy Pre and Post Assertions	0.2.1
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.5
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.72.0-3
broom	Convert Statistical Analysis Objects into Tidy Tibbles	0.5.4
callr	Call R from R	3.4.1
cellranger	Translate Spreadsheet Cell Ranges to Rows	1.1.0

```
59 summary(sms_train[, 1:5])
60
61 ## Step 3: Training a model on the data ----
62 library(e1071)
63 sms_classifier <- naiveBayes(sms_train, sms_raw_train$type)
64 names(sms_classifier)
65
66 sms_classifier$tables[1:2]
67
68
69 ## Step 4: Evaluating model performance ----
70 sms_test_pred <- predict(sms_classifier, sms_test)
71 library(gmodels)
72 crosstable(sms_test_pred, sms_raw_test$type, prop.chisq = FALSE,
73           dnm = c("predicted", "actual"))
74
75
76 Step 4: Evaluating model performance : R script
```

Console Terminal

```
C:\Users\NPTEL Studio-01\Desktop\Week10\Session 4\naive bayes/ >
content type 'application/zip' length 114408 bytes (111 kB)
downloaded 111 kB

package 'gtools' successfully unpacked and MD5 sums checked
package 'gdata' successfully unpacked and MD5 sums checked
package 'gmodels' successfully unpacked and MD5 sums checked

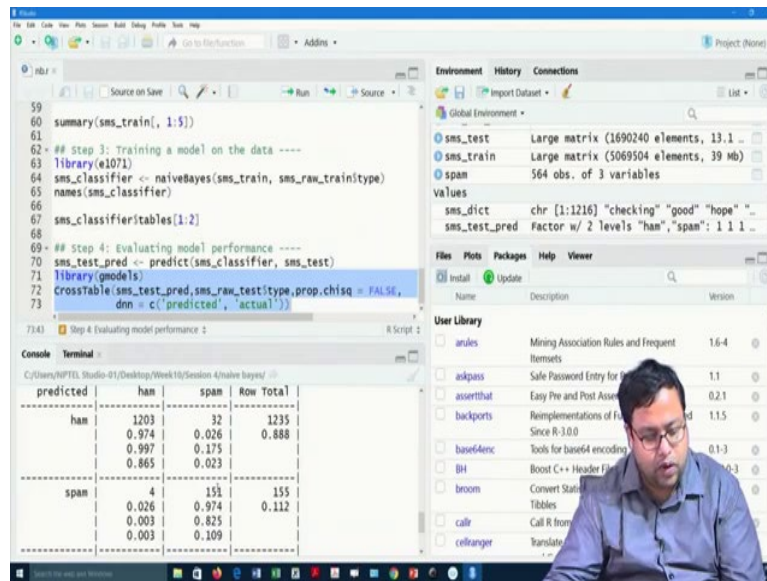
The downloaded binary packages are in
c:\Users\NPTEL Studio-01\AppData\Local\Temp\ktmpuiv5w\downloaded_packages
>
```

Environment

Object	Class	Attributes
sms_test	Large matrix	(1690240 elements, 13.1 MB)
sms_train	Large matrix	(5069504 elements, 39 MB)
span	Factor	564 obs. of 3 variables

User Library

Package	Description	Version
arules	Mining Association Rules and Frequent Itemsets	1.6-4
askpass	Safe Password Entry for R, Git, and SSH	1.1
assertthat	Easy Pre and Post Assertions	0.2.1
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.5
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.72.0-3
broom	Convert Statistical Analysis Objects into Tidy Tibbles	0.5.4
callr	Call R from R	3.4.1
cellranger	Translate Spreadsheet Cell Ranges to Rows	1.1.0



So, now based on these, I will predict with my SMS test as my testing data, I will predict and then in the testing data, we will try to see that whether prediction is working well or not. So, let it predict a little bit, it might take for only another 1 minute time. So, we are handling pretty large dataset, 5000 data set, even if it is taking 30 second, 40 second, that is okay, good enough to deal with. So, let it predict. So, right now it is predicting for the testing data only and SMS test is around 1390 observations.

But there are 1216 variables that are there. So, based on that, it is predicting, yes, the predictions has been done and now if I just plot them, okay, so G model's package is not there I will quickly do G models. Install the package. Now if I run this you see out of 1390 observations, I am correct 1203 times and 151 times. So, 1203 plus 151 comes up to be how much? 1354?

(Refer Slide Time: 18:37)

```
59 summary(sms_train[, 1:5])
60
61 ## Step 3: Training a model on the data ----
62 library(e1071)
63 sms_classifier <- naiveBayes(sms_train, sms_raw_train$type)
64 names(sms_classifier)
65
66 sms_classifier$tables[1:2]
67
68 ## Step 4: Evaluating model performance ----
69 sms_test_pred <- predict(sms_classifier, sms_test)
70 library(gmodels)
71 CrossTable(sms_test_pred, sms_raw_test$type, prop.chisq = FALSE,
72           dnn = c("predicted", "actual"))
```

Console Terminal

```
C:/Users/NPTEL Studio-01/Desktop/Week10/Session 4/naive bayes/ >>
-----
Column Total |      1207 |      183 |      1390 |
-----
               | 0.868    | 0.132    |             |
-----
> 1354/1390
[1] 0.9741007
> 151/183
[1] 0.8251366
```

Environment History Connections

- Global Environment
- sms_test: Large matrix (1690240 elements, 13.1 ...)
- sms_train: Large matrix (5069504 elements, 39 Mb)
- span: 564 obs. of 3 variables

Values

- sms_dict: chr [1:1216] "checking" "good" "hope" "
- sms_test_pred: Factor w/ 2 levels "ham", "spam": 1 1 1 ...

User Library

- arules: Mining Association Rules and Frequent Itemsets 1.6-4
- askpass: Safe Password Entry for R 1.1
- assertthat: Easy Pre and Post Assertions 0.2.1
- backports: Reimplementations of Functions Since R 3.0.0 1.1.5
- base64enc: Tools for base64 encoding 1.1-3
- BH: Boost C++ Header Files 1.62.0
- broom: Convert Statistical Objects to Tidy Tibbles 1.0.0
- callr: Call R from R 3.0.0
- collanger: Translate Statistical Objects 1.0.0

So, $1354 / 1390$, I am correct almost 97 % times on this data set. Okay and then if I just think about spam detection there was this is actual, this is predicted, so actually there was spam 183 and I am correctly predicting 151 out of them. So, $151 / 183$, I am also correct 82 % times in the spam detection part. So, no correct in the ham detection part, there is correctness spam detection part but still is giving pretty good result. So that is how we can use Naive Bayes algorithm in spam detection.

So, that is all for this particular video, in the next video or probably in the next week, we will use all whatever we have learnt in further more advance application like sentiment mining and emotion mining to find out newer insights. So, thank you very much for being with me in this particular video and I will see you in the next week. Thank you.