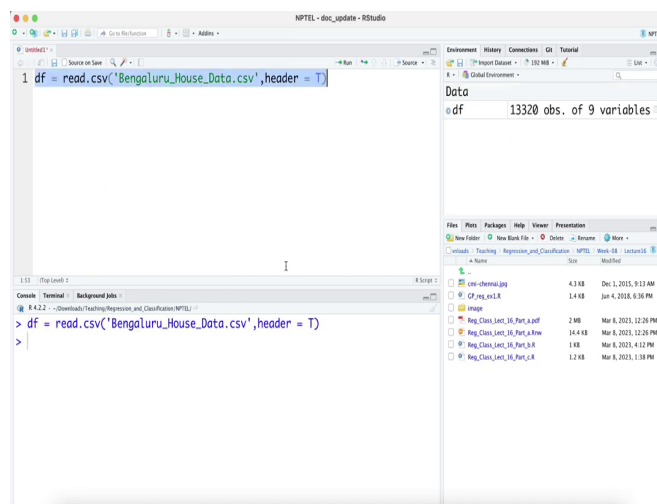


Predictive Analytics - Regression and Classification
Prof. Sourish Das
Department of Mathematics
Chennai Mathematical Institute

Lecture - 55
Hands on with R : Prediction of Bangalore House Price

Hello all, in this video we are going to do the Bangalore House Price data analysis using R. In the previous video we did the same analysis with Python. Now, if we do using R, we will get a sense of both language, how both language works and some of the subtle difference in the language.

(Refer Slide Time: 00:46)



The screenshot shows the RStudio interface. The script editor on the left contains the following R code:

```
1 df = read.csv("Bengaluru_House_Data.csv", header = T)
```

The console at the bottom shows the execution of the code:

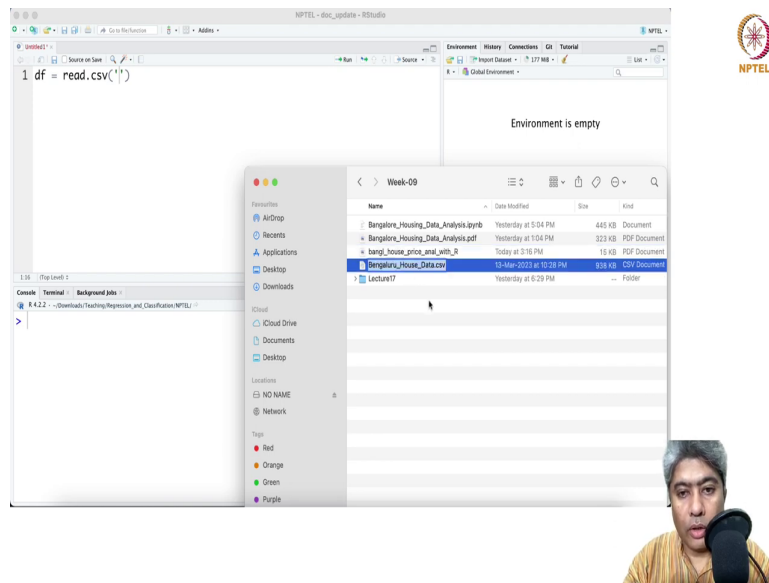
```
> df = read.csv("Bengaluru_House_Data.csv", header = T)
>
```

The environment pane on the right shows the data frame 'df' with 13320 observations and 9 variables. The file explorer on the right shows the project files.



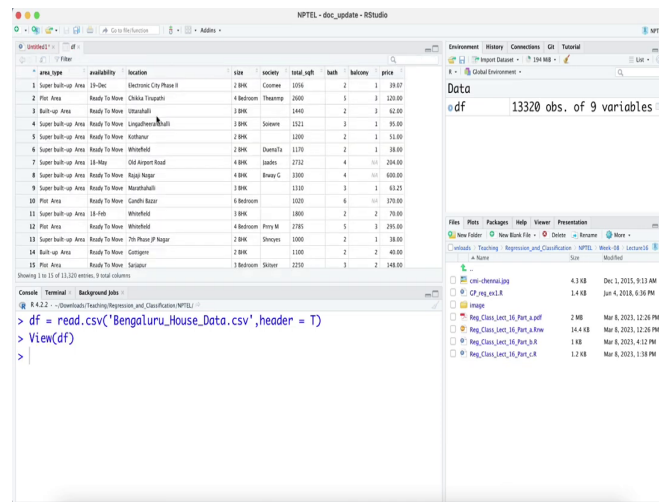
So, first I will call data set read dot csv.

(Refer Slide Time: 01:02)



So, let me just go and take the copy the name of the data set and let me just call header equal to true. So, now you can see the data has come here.

(Refer Slide Time: 01:20)



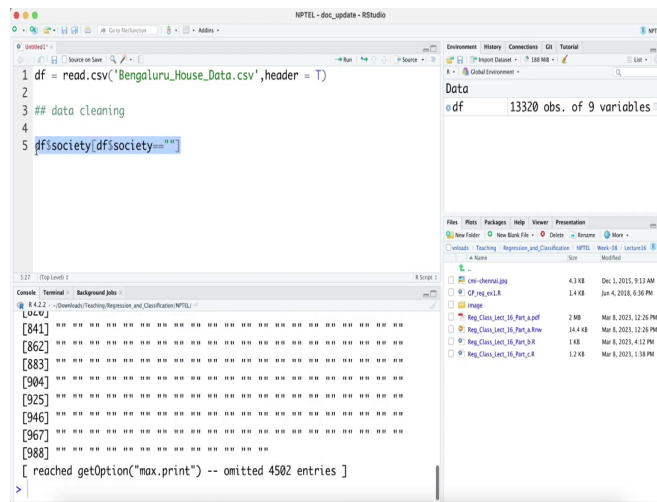
The screenshot shows the RStudio interface. The main window displays a data frame 'df' with 13320 observations and 9 variables. The console shows the command 'df = read.csv('Bengaluru_House_Data.csv', header = T)' and 'View(df)'. The Environment pane shows the data frame 'df'.

| area_type | availability | location | size | society | total_sqft | bath | balcony | price | |
|-----------|---------------------|---------------|--------------------------|-----------|------------|------|---------|--------|--------|
| 1 | Super built-up Area | 19-Dec | Electronic City Phase II | 2 BHK | Cosmos | 1056 | 2 | 1 | 39.87 |
| 2 | Plot Area | Ready To Move | Chikka Tappa | 4 Bedroom | Theanp | 2000 | 5 | 3 | 120.00 |
| 3 | Plot Area | Ready To Move | Chikka Tappa | 3 BHK | 1440 | 2 | 3 | 62.00 | |
| 4 | Super built-up Area | Ready To Move | Lingappaiah | 3 BHK | Sirens | 1521 | 3 | 1 | 95.00 |
| 5 | Super built-up Area | Ready To Move | Kothanur | 2 BHK | 1209 | 2 | 1 | 51.00 | |
| 6 | Super built-up Area | Ready To Move | Whitefield | 2 BHK | Durga | 1170 | 2 | 1 | 38.00 |
| 7 | Super built-up Area | 18-May | Old Airport Road | 4 BHK | Islands | 2732 | 4 | 1 | 204.00 |
| 8 | Super built-up Area | Ready To Move | Kappa Nagar | 4 BHK | Brave | 1700 | 4 | 1 | 600.00 |
| 9 | Super built-up Area | Ready To Move | Manshulali | 3 BHK | 1110 | 3 | 1 | 63.25 | |
| 10 | Plot Area | Ready To Move | Candolim Bazar | 6 Bedroom | 1020 | 6 | 1 | 370.00 | |
| 11 | Super built-up Area | 18-Feb | Whitefield | 3 BHK | 1800 | 2 | 2 | 70.00 | |
| 12 | Plot Area | Ready To Move | Whitefield | 4 Bedroom | Pray W | 2700 | 5 | 3 | 295.00 |
| 13 | Super built-up Area | Ready To Move | Tin Manu Nagar | 2 BHK | 1000 | 2 | 1 | 38.00 | |
| 14 | Plot Area | Ready To Move | Comptons | 2 BHK | 1100 | 2 | 2 | 40.00 | |
| 15 | Plot Area | Ready To Move | Satapur | 3 Bedroom | Skinner | 2250 | 3 | 2 | 148.00 |



So, it has 13,320 observation and 9 variables. So, 9 columns you can see all of them here. Now, first thing we will do since we have done quite a view of visualization in Python we will not spend too much time on the visualization. We can do little bit, but first thing probably what we will try to do. We will see we will try to do a some data cleaning.

(Refer Slide Time: 02:05)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The source editor contains the following R code:

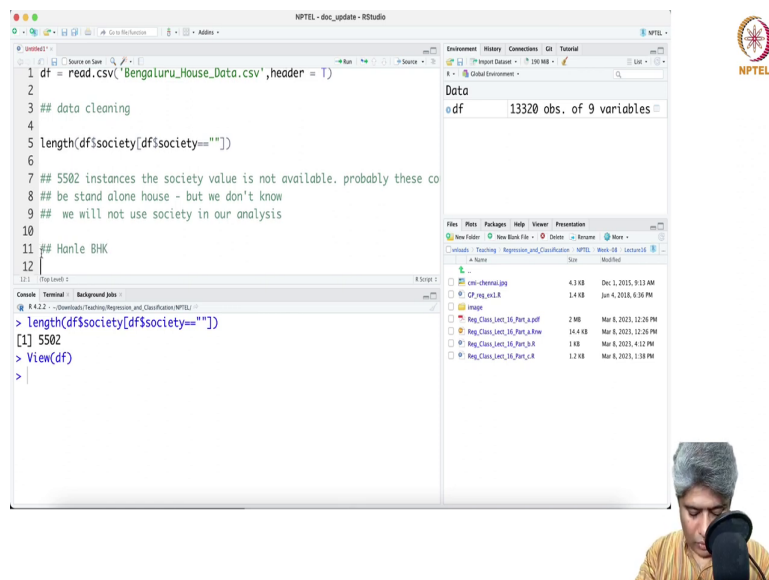
```
1 df = read.csv("Bengaluru_House_Data.csv", header = T)
2
3 ## data cleaning
4
5 df$society[df$society==""] = NA
```

The console shows the output of the code, indicating that 4502 entries were omitted due to reaching the maximum print limit. The environment pane on the right shows the data frame 'df' with 13320 observations and 9 variables. The file pane shows the project structure, including a folder named 'NPTEL' and a file named 'Bengaluru_House_Data.csv'.



First we have to do some data cleaning and if you look into the data you can see in the society there are quite a few you know empty variables are there empty cells are there. So, we can check how many empty cells are there by calling the extracting the society column equated to quote unquote and then take the length of just df society ok.

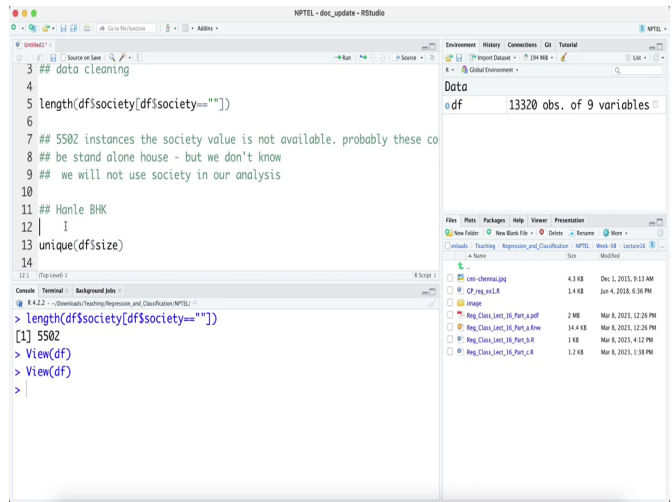
(Refer Slide Time: 02:59)



So, this will give you all the cases where you have quote unquote kind of thing and then if you just say length then it will give you. So, 51 5502 cases where society value is not available. So, you can see it in this way we can write it down have a note then write it down with a note in 5502 instances the society value is not available. We probably these are probably these could be; these could be stand alone house, but we do not know ok. So, the next thing is so, probably we will not use society in our analysis.

So, we will not use society as we did in Python we will not use society in our analysis as we did in Python. So, I am not ok. Next is BHK and you can see most of these BHKs are you know exactly like at the bedroom sometime BHK. So, we have to handle that. So, we have to handling the handle BHK variable.

(Refer Slide Time: 04:48)





The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The editor pane contains the following R code:

```
3 ## data cleaning
4
5 length(df$society[df$society==""])
6
7 ## 5502 instances the society value is not available. probably these co
8 ## be stand alone house - but we don't know
9 ## we will not use society in our analysis
10
11 ## Handle BHK
12 |
13 unique(df$size)
14
```

The console pane shows the output of the code:

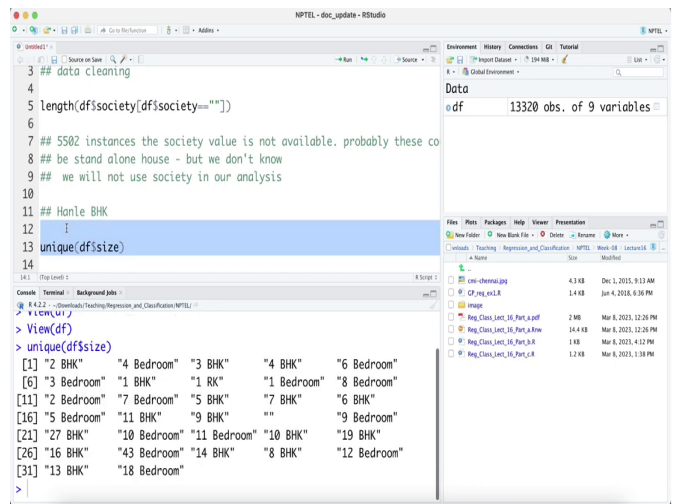
```
> length(df$society[df$society==""])
[1] 5502
> View(df)
> View(df)
>
```

The Environment pane on the right shows a data frame "df" with 13320 observations and 9 variables. The File pane on the right shows a list of files and folders, including "cnn-chemical.jpg", "CJ_mg_m1.8", "image", "Reg_Class_Lant_H_Pant_a.pdf", "Reg_Class_Lant_H_Pant_a.km", "Reg_Class_Lant_H_Pant_a.R", and "Reg_Class_Lant_H_Pant_a.R".



So, how we do that first let us see how many unique BHK columns are there. It is written as size I believe, ok. Yeah size correct.

(Refer Slide Time: 05:06)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:



```
3 ## data cleaning
4
5 length(df$society[df$society==""])
6
7 ## $502 instances the society value is not available. probably these co
8 ## be stand alone house - but we don't know
9 ## we will not use society in our analysis
10
11 ## Handle BHK
12 I
13 unique(df$size)
14
```

The console shows the output of the code:

```
> View(df)
> unique(df$size)
[1] "2 BHK"      "4 Bedroom"  "3 BHK"      "4 BHK"      "6 Bedroom"
[6] "3 Bedroom"  "1 BHK"      "1 RK"       "1 Bedroom"  "8 Bedroom"
[11] "2 Bedroom"  "7 Bedroom"  "5 BHK"      "7 BHK"      "6 BHK"
[16] "5 Bedroom"  "11 BHK"     "9 BHK"      ""           "9 Bedroom"
[21] "7 BHK"      "10 Bedroom" "11 Bedroom" "10 BHK"     "19 BHK"
[26] "16 BHK"     "43 Bedroom" "14 BHK"     "8 BHK"      "12 Bedroom"
[31] "13 BHK"     "18 Bedroom"

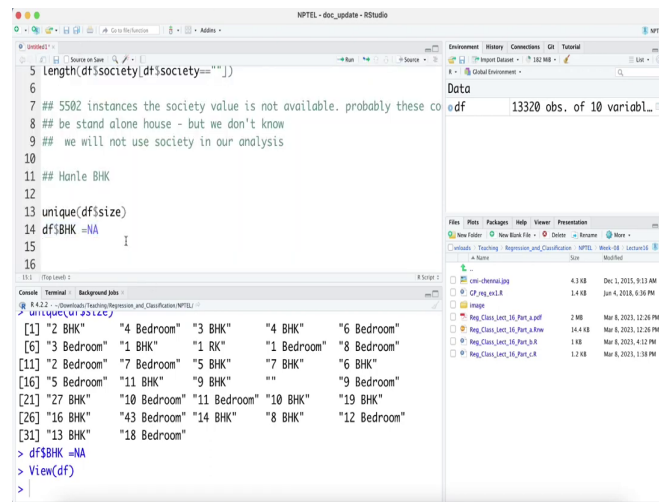
>
```

The Environment pane on the right shows a data frame "df" with 13320 observations and 9 variables. The File pane on the right shows a list of files in the "NPTEL" directory.



So, these are the BHK things.

(Refer Slide Time: 05:14)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The editor pane contains the following R code:

```
5 length(df$society[df$society==""])
6
7 ## 5502 instances the society value is not available. probably these co
8 ## be stand alone house - but we don't know
9 ## we will not use society in our analysis
10
11 ## Handle BHK
12
13 unique(df$size)
14 df$BHK =NA
15
16
```

The console pane shows the output of the code:

```
[1] "2 BHK" "4 Bedroom" "3 BHK" "4 BHK" "6 Bedroom"
[6] "3 Bedroom" "1 BHK" "1 RK" "1 Bedroom" "8 Bedroom"
[11] "2 Bedroom" "7 Bedroom" "5 BHK" "7 BHK" "6 BHK"
[16] "5 Bedroom" "11 BHK" "9 BHK" "" "9 Bedroom"
[21] "27 BHK" "10 Bedroom" "11 Bedroom" "10 BHK" "19 BHK"
[26] "16 BHK" "43 Bedroom" "14 BHK" "8 BHK" "12 Bedroom"
[31] "13 BHK" "18 Bedroom"

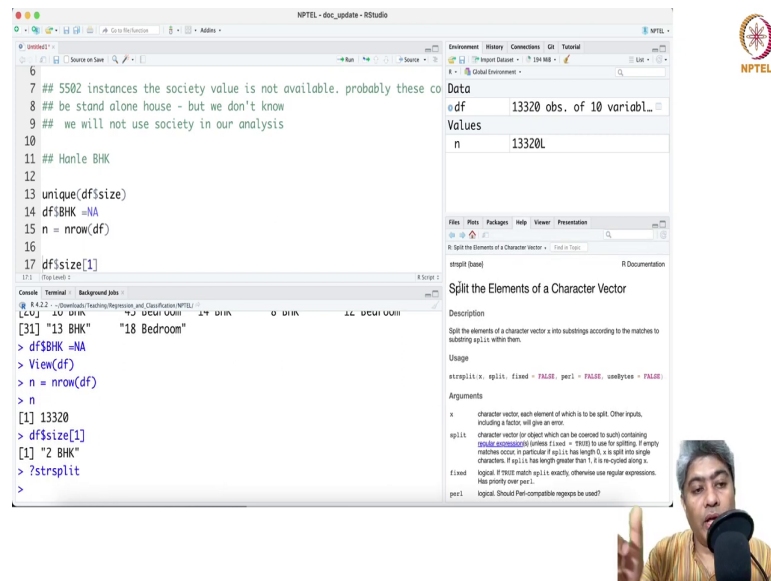
> df$BHK =NA
> View(df)
>
```

The Environment pane on the right shows the 'Data' environment with 'df' containing 13320 observations of 10 variables. The File pane on the right shows a list of files including 'cse-chemical.jpg', 'CF_MIS_m1.8', 'image', 'Reg_Class_Lent_H_Prot_a.pdf', 'Reg_Class_Lent_H_Prot_a.Rnw', 'Reg_Class_Lent_H_Prot_a.R', and 'Reg_Class_Lent_H_Prot_a.Rproj'.



So, now what I will do as we did in Python we will create another column called BHK equal to NA first. So, now there are 10th column there will be a 10th column with BHK all values are NA just we created a placeholder, ok.

(Refer Slide Time: 05:32)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains R code for data cleaning and inspection. The console shows the execution of these commands, resulting in the dimensions of the data frame and the first few rows. The right-hand pane displays the documentation for the `strsplit` function.

```
# 5502 instances the society value is not available, probably these co
## be stand alone house - but we don't know
## we will not use society in our analysis
## Hanle BHK
unique(df$size)
df$BHK = NA
n = nrow(df)
df$size[1]
```

Console output:

```
[1] "13 BHK" "18 Bedroom"
> df$BHK = NA
> View(df)
> n = nrow(df)
> n
[1] 13320
> df$size[1]
[1] "2 BHK"
> ?strsplit
>
```

Documentation for `strsplit`:

Split the Elements of a Character Vector

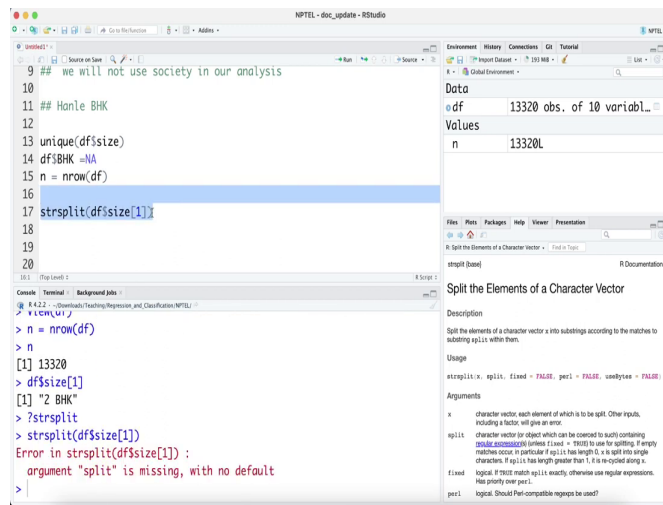
Description: Split the elements of a character vector `x` into substrings according to the matches to substring `split` within them.

Usage: `strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)`

Arguments: `x` character vector, each element of which is to be split. Other inputs, including a factor, will give an error. `split` character vector or object which can be coerced to such containing regular expressions (unless `fixed = TRUE`) to use for splitting. If empty matches occur, in particular if `split` has length 0, `x` is split into single characters. If `split` has length greater than 1, it is re-cycled along `x`. `fixed` logical. If `TRUE` match `split` exactly, otherwise use regular expressions. Help priority over `perl`. `perl` logical. Should Perl-compatible regexps be used?

Now, `n` equal to `nrow` of `df` is the number of row in the data set 13,320, ok. So, first thing we will do is let us take the first case it is to quote unquote BHK right. Now, what I am going to do I am going to called function called string, string split, ok. It split the elements of a character vector, ok.

(Refer Slide Time: 06:20)



The image shows the RStudio interface. The script editor on the left contains the following code:

```
9 ## we will not use society in our analysis
10
11 ## Hanle BHK
12
13 unique(df$size)
14 df$BHK = NA
15 n = nrow(df)
16
17 strsplit(df$size[1])
18
19
20
```

The console on the bottom left shows the following output:

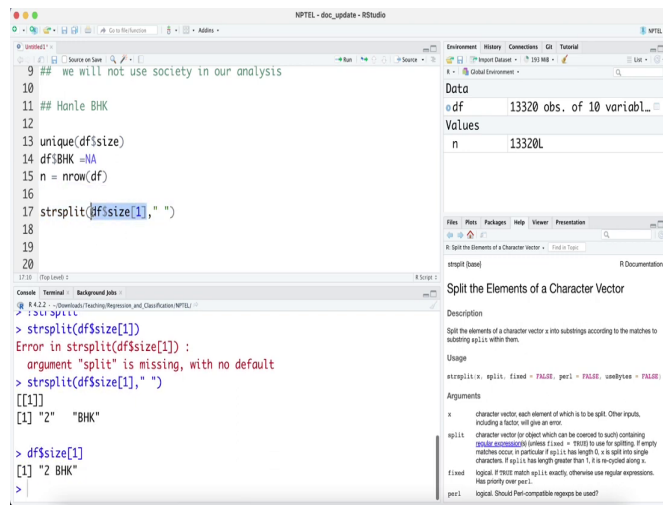
```
> n = nrow(df)
> n
[1] 13320
> df$size[1]
[1] "2 BHK"
> ?strsplit
> strsplit(df$size[1])
Error in strsplit(df$size[1]) :
  argument "split" is missing, with no default
>
```

The environment pane on the right shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' section shows 'n' as 13320L. The help pane on the right shows the documentation for the 'strsplit' function.



So, what it will do is string, (Refer Time: 06:25) string split 1 ok. Let me just first run this.

(Refer Slide Time: 06:32)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
9 ## we will not use society in our analysis
10
11 ## Hanle BHK
12
13 unique(df$size)
14 df$BHK = NA
15 n = nrow(df)
16
17 strsplit(df$size[1], " ")
18
19
20
```

The console shows the execution of the code, with an error message for the first `strsplit` call and the output for the second call:

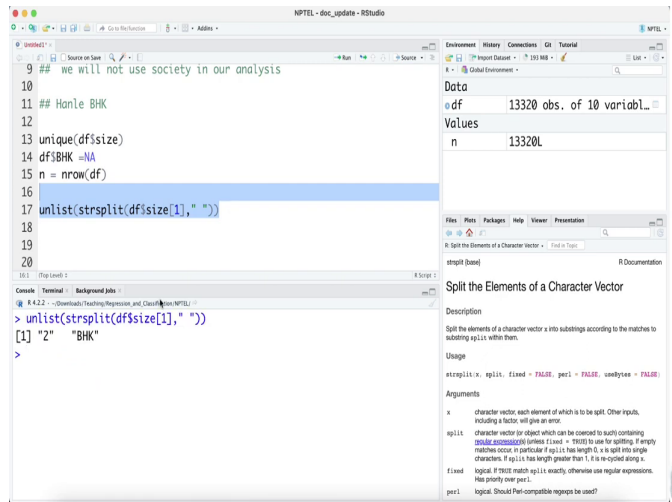
```
> strsplit(df$size[1])
Error in strsplit(df$size[1]) :
  argument "split" is missing, with no default
> strsplit(df$size[1], " ")
[[1]]
[1] "2" "BHK"
```

The environment pane shows the data frame `df` with 13320 observations and 10 variables. The values pane shows the value `n` as 13320L. The right pane shows the documentation for the `strsplit` function.





I have to give like by what they would split. So, now we can see. So, that this was my original df size to BHK. Now, they splitted into 2 and BHK. So, now, but they have kept it as a list.

(Refer Slide Time: 06:54)



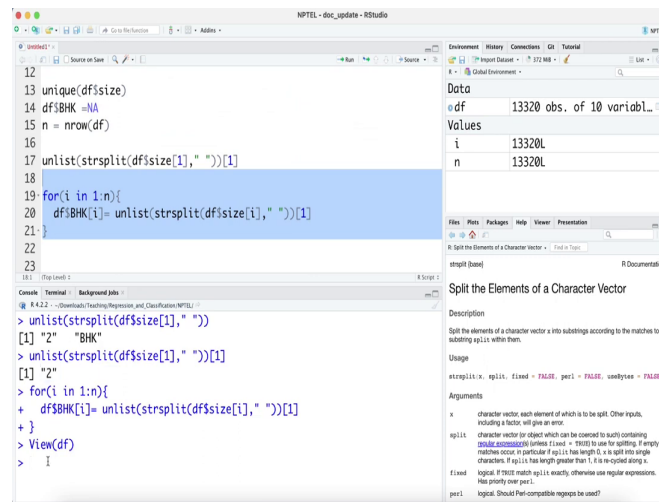
The image shows the RStudio interface with the following components:

- Source Editor:** Contains R code for data manipulation. Line 17 is highlighted: `unlist(strsplit(df$size[1], " "))`.
- Environment:** Shows a data frame `df` with 13320 observations and 10 variables. The variable `n` has a value of 13320.
- Console:** Shows the execution of `unlist(strsplit(df$size[1], " "))`, resulting in a character vector: `[1] "2" "BHK"`.
- Help Panel:** Displays the documentation for the `strsplit` function, titled "Split the Elements of a Character Vector".



So, we have to put it as a unlist. We have to put it as a unlist, ok.

(Refer Slide Time: 07:10)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The script editor contains the following R code:

```
12
13 unique(df$size)
14 df$BHK = NA
15 n = nrow(df)
16
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22
23
```

The console shows the execution of the code:

```
> unlist(strsplit(df$size[1], " "))
[1] "2" "BHK"
> unlist(strsplit(df$size[1], " "))[1]
[1] "2"
> for(i in 1:n){
+   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
+ }
> View(df)
> i
```

The environment pane on the right shows the 'Data' tab with a table:

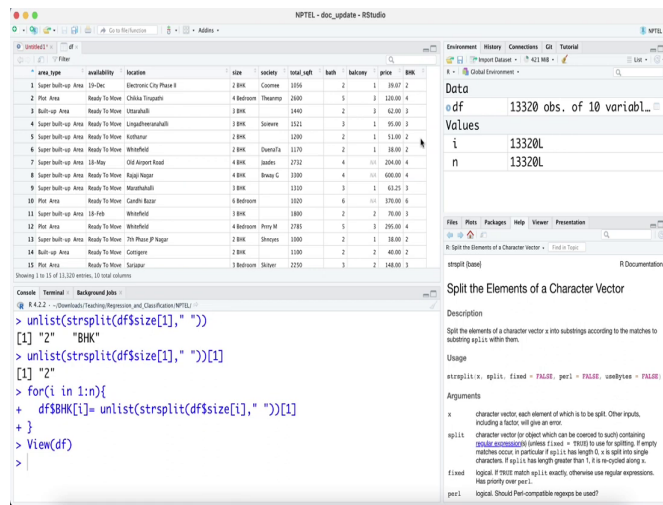
| | df |
|--------|-----------------------------|
| obs. | 13320 obs. of 10 variabl... |
| Values | |
| i | 13320L |
| n | 13320L |

The right pane also shows the documentation for the 'strsplit' function, titled 'Split the Elements of a Character Vector'.



Now, if you just put it as a unlist. So, the first element is two next element is BHK. Now, what I am going to do and then if we just put say 1 then it will give me the first element which is 2. So, now what I am going to do I am going to run it through this line basically over the all cases 1 is to n; unlist and this will be like df dollar BHK on the ith cell and this will be ith component. So, let us run this thing. It is done.

(Refer Slide Time: 08:00)



The screenshot displays the RStudio interface. The top-left pane shows a data frame 'df' with 13320 observations and 10 variables. The top-right pane shows the environment with 'df' having dimensions 13320 x 10. The bottom-left pane shows the console with the following R code:

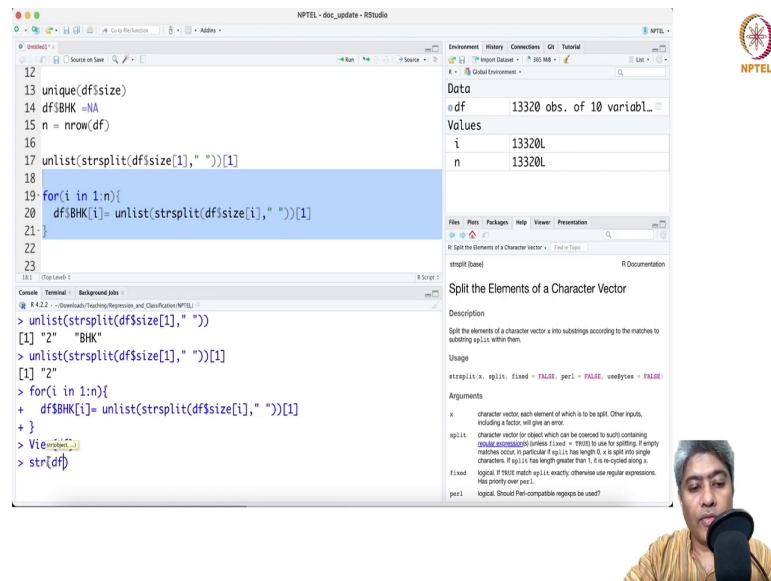
```
> unlist(strsplit(df$size[1], ""))
[1] "2" "BHK"
> unlist(strsplit(df$size[1], " "))[1]
[1] "2"
> for(i in 1:n){
+   df$BHK[i]= unlist(strsplit(df$size[i], " "))[1]
+ }
> View(df)
>
```

The bottom-right pane shows the documentation for the 'strsplit' function, titled 'Split the Elements of a Character Vector'.



Now, if I just go there you see it is filled up. But remember that when it was done it was character.

(Refer Slide Time: 08:08)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
12
13 unique(df$size)
14 df$BHK = NA
15 n = nrow(df)
16
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22
23
```

The console shows the execution of the code:

```
> unlist(strsplit(df$size[1], " "))
[1] "2" "BHK"
> unlist(strsplit(df$size[1], " "))[1]
[1] "2"
> for(i in 1:n){
+   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
+ }
> View(df)
> str(df)
```

The Environment pane shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' pane shows the first few values of 'i' and 'n'.

The R Documentation pane shows the documentation for the `strsplit` function:

Split the Elements of a Character Vector

Description

Split the elements of a character vector `x` into substrings according to the matches to substring `split` within them.

Usage

```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

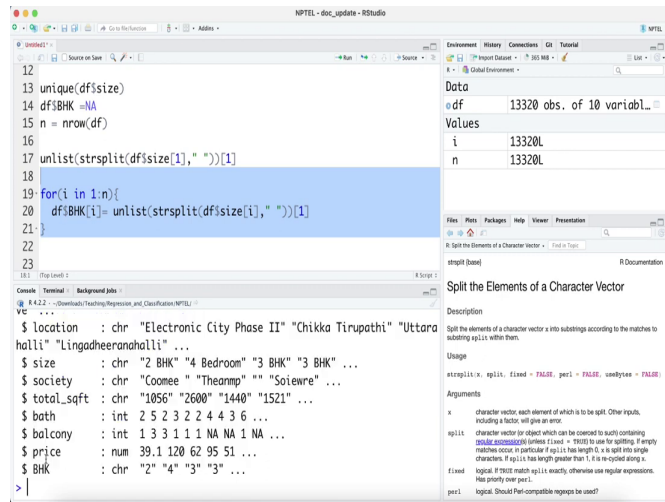
Arguments

- `x`: character vector, each element of which is to be split. Other inputs, including a factor, will give an error.
- `split`: character vector or object which can be coerced to such containing regular expressions (unless `fixed = TRUE`) to use for splitting. If empty matches occur, in particular if `split` has length 0, `x` is split into single characters. If `split` has length greater than 1, it is re-cycled along `x`.
- `fixed`: logical. If `TRUE` match `split` exactly, otherwise use regular expressions. Also priority over `perl`.
- `perl`: logical. Should Perl-compatible regexes be used?

A video inset in the bottom right corner shows a man speaking into a microphone.

So, if I just check the structure of the data set df.

(Refer Slide Time: 08:13)



The image shows an RStudio session. The script editor contains the following R code:

```
12
13 unique(df$size)
14 df$BHK = NA
15 n = nrow(df)
16
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22
23
```

The console shows the output of the code:

```
$ location : chr "Electronic City Phase II" "Chikka Tirupathi" "Uttara
halli" "Lingadheeranahalli" ...
$ size : chr "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
$ society : chr "Coomee" "Theamp" "" "Soiewre" ...
$ total_sqft : chr "1056" "2600" "1440" "1521" ...
$ bath : int 2 5 2 3 2 2 4 4 3 6 ...
$ balcony : int 1 3 3 1 1 1 NA NA 1 NA ...
$ price : num 39.1 120 62 95 51 ...
$ BHK : chr "2" "4" "3" "3" ...
>
```

The Environment pane shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' pane shows the values for 'i', 'df', and 'n'.

The R Documentation pane shows the documentation for the `strsplit` function:

Split the Elements of a Character Vector

Description


Split the elements of a character vector `x` into substrings according to the matches to substring `split` within them.

Usage

```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

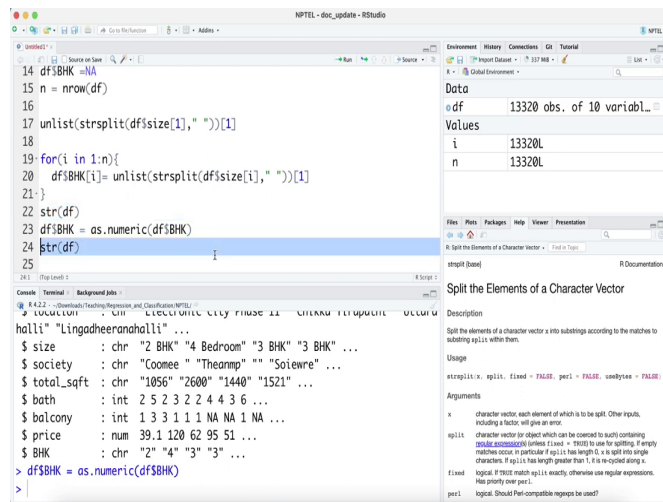
Arguments

- `x`: character vector, each element of which is to be split. Other inputs, including a factor, will give an error.
- `split`: character vector or object which can be coerced to such containing special characters (unless `fixed = TRUE`) to use for splitting. If empty matches occur, in particular if `split` has length 0, `x` is split into single characters. If `split` has length greater than 1, it is re-cycled along `x`.
- `fixed`: logical. If `TRUE` match `split` exactly; otherwise use regular expressions. Help priority over `perl`.
- `perl`: logical. Should Perl-compatible regexps be used?



We will see BHK is kind of used as a is being stored as a character, ok.

(Refer Slide Time: 08:22)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The editor pane contains the following R code:

```
14 df$BHK = NA
15 n = nrow(df)
16
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22 str(df)
23 df$BHK = as.numeric(df$BHK)
24 str(df)
25
```

The console pane shows the output of the code:

```
halhi" "Lingadheeranahalli" ...
$ size      : chr "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
$ society   : chr "Coomee" "Theanmp" "" "Soienre" ...
$ total_sqft : chr "1056" "2600" "1440" "1521" ...
$ bath      : int  2 5 2 3 2 2 4 4 3 6 ...
$ balcony   : int  1 3 3 1 1 1 NA NA 1 NA ...
$ price     : num  39.1 120 62 95 51 ...
$ BHK       : chr  "2" "4" "3" "3" ...
> df$BHK = as.numeric(df$BHK)
>
```

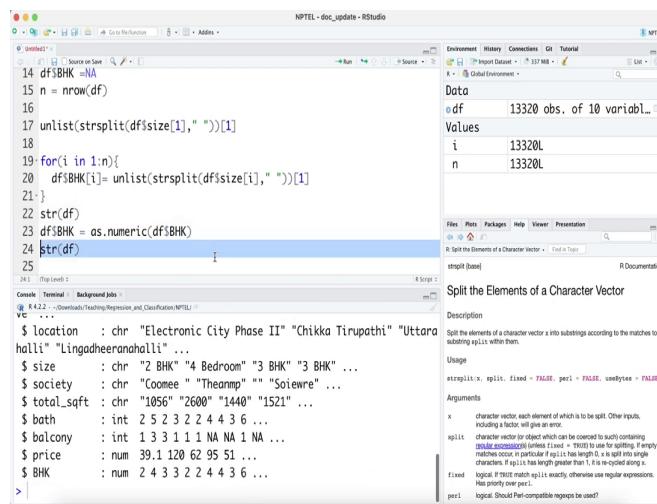
The environment pane on the right shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' section shows the values for 'i', 'df', and 'n'.

The help pane on the right shows the documentation for the 'strsplit' function, titled 'Split the Elements of a Character Vector'.



So, now what I am going to do df as a df dollar BHK as dot numeric df dollar BHK. I think no problem. Now, if I just run this.

(Refer Slide Time: 08:51)



```
14 df$BHK = NA
15 n = nrow(df)
16
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22 str(df)
23 df$BHK = as.numeric(df$BHK)
24 str(df)
25
```

Environment: Global Environment
Data
df 13320 obs. of 10 variables
Values
i 13320L
n 13320L

Split the Elements of a Character Vector

Description
Split the elements of a character vector x into substrings according to the matches to substring expr within them.

Usage
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)

Arguments
x character vector, each element of which is to be split. Other inputs, including a factor, will give an error.
split character vector or object which can be coerced to such containing special characters (lines <code>\\</code> must be used for splitting. If empty matches occur, in particular if split has length 0, x is split into single characters. If split has length greater than 1, it is re-cycled along x.
fixed logical. If TRUE match split exactly, otherwise use regular expressions. Help priority over perl.
perl logical. Should Perl-compatible regulars be used?

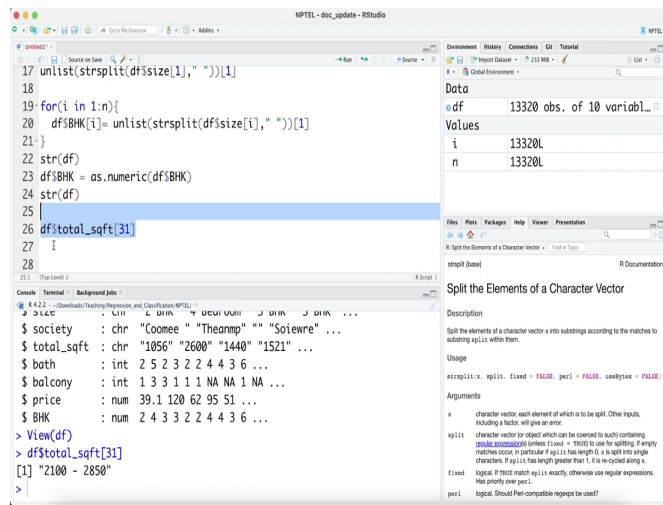
Console
\$ location : chr "Electronic City Phase II" "Chikka Tirupathi" "Uttara halli" "Lingadheeranahalli" ...
\$ size : chr "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
\$ society : chr "Coomee" "Theamp" " " "Soleure" ...
\$ total_sqft : chr "1056" "2600" "1440" "1521" ...
\$ bath : int 2 5 2 3 2 2 4 4 3 6 ...
\$ balcony : int 1 3 3 1 1 1 NA NA 1 NA ...
\$ price : num 39.1 120 62 95 51 ...
\$ BHK : num 2 4 3 3 2 2 4 4 3 6 ...



Now, you can see BHK is numeric no problem, ok alright. Next we have to if let us look into the data set. So, BHK now size was split and you know put it into two and you know the size of the like number numeric it was converted into numeric. Now, we have to look into the total square feet now because if you can see total square feets are considered as a character variable.

So, because there must there were lot of dash sign or you know very weird way of stored values were done. For example, this one 3010 or 2100 to 2850 this 31 row ok. The 31 row they have done it in a very way characteristic or string way of substituting the data, ok.

(Refer Slide Time: 10:02)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor contains the following R code:

```
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22 str(df)
23 df$BHK = as.numeric(df$BHK)
24 str(df)
25
26 df$total_sqft[31]
27 I
28
```

The console output shows the structure of the data frame 'df':

```
$ society : chr "Coomer" "Theamp" "Soienre" ...
$ total_sqft : chr "1056" "2600" "1440" "1521" ...
$ bath : int 2 5 2 3 2 2 4 4 3 6 ...
$ balcony : int 1 3 3 1 1 1 NA NA 1 NA ...
$ price : num 39.1 120 62 95 51 ...
$ BHK : num 2 4 3 3 2 2 4 4 3 6 ...
```

The user has executed the command `df$total_sqft[31]`, and the console shows the output: `[1] "2100 - 2850"`.

The right-hand pane shows the 'Data' tab with a summary of the data frame 'df':

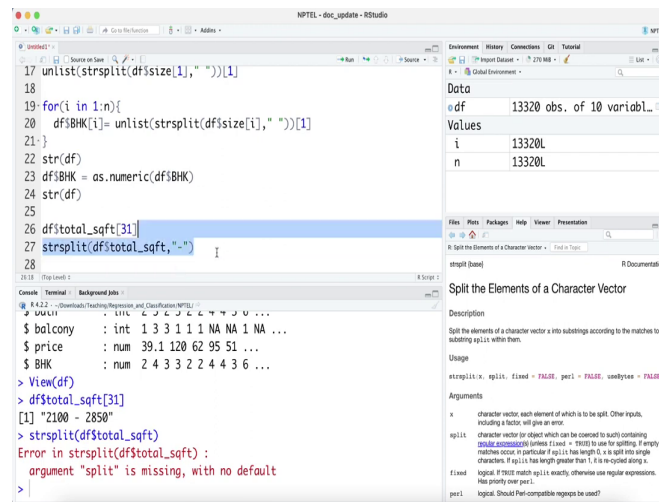
| Variable | Value |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| i | 13320L |
| n | 13320L |

The 'Help' tab is open, showing the documentation for the `strsplit` function.



So, if you look into the df dollar total square feet 31 if you look into this it is a very string way of doing data.

(Refer Slide Time: 10:16)



The image shows an RStudio window with the following content:

```
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22 str(df)
23 df$BHK = as.numeric(df$BHK)
24 str(df)
25
26 df$total_sqft[31]
27 strsplit(df$total_sqft, "-")
28
```

The console shows the following output:

```
> View(df)
> df$total_sqft[31]
[1] "2100 - 2850"
> strsplit(df$total_sqft)
Error in strsplit(df$total_sqft) :
  argument "split" is missing, with no default
>
```

The environment pane shows the following data:

| Variable | Value |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| i | 13320L |
| n | 13320L |

The right pane shows the documentation for the `strsplit` function:

Split the Elements of a Character Vector

Description

Split the elements of a character vector `x` into substrings according to the matches to substring `split` within them.

Usage

```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

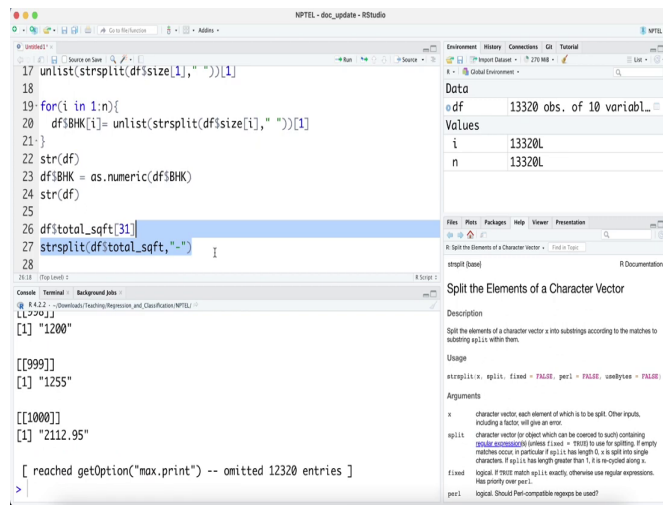
Arguments

- `x`: character vector, each element of which is to be split. Other inputs, including a factor, will give an error.
- `split`: character vector or object which can be coerced to such containing special characters (unless `fixed = TRUE`) to use for splitting. If empty matches occur, in particular if `split` has length 0, `x` is split into single characters. If `split` has length greater than 1, it is re-cycled along `x`.
- `fixed`: logical. If `TRUE` match `split` exactly, otherwise use regular expressions. Help priority over `perl`.
- `perl`: logical. Should Perl-compatible regexes be used?



So, now what we can do? We can just say ok string split df dollar total square feet, ok. I have to give up yes we have to give a say dash right.

(Refer Slide Time: 10:34)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The editor pane contains the following R code:

```
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22 str(df)
23 df$BHK = as.numeric(df$BHK)
24 str(df)
25
26 df$total_sqft[31]
27 strsplit(df$total_sqft, "-")
28
```

The console pane shows the output of the code:

```
[[[999]]]
[1] "1255"

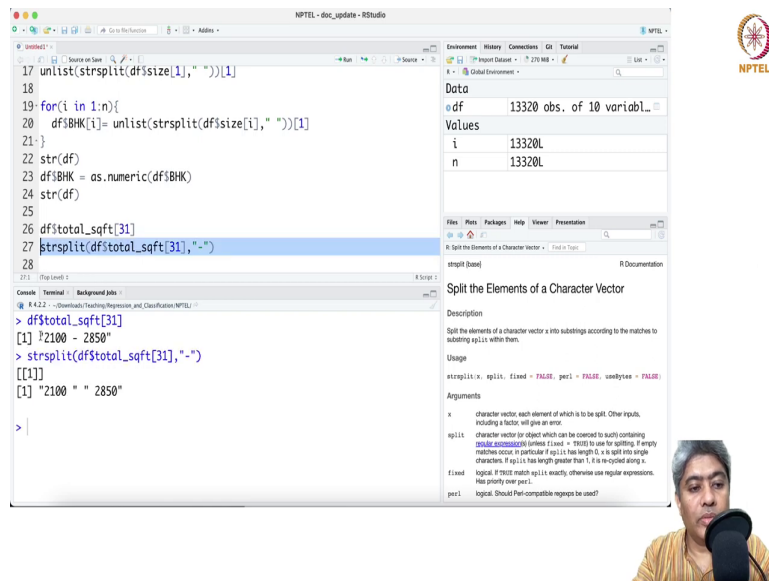
[[[1000]]]
[1] "2112.95"

[ reached getOption("max.print") -- omitted 12320 entries ]
>
```

The Environment pane shows the data frame "df" with 13320 observations and 10 variables. The Variables pane shows the values of "i" and "n". The Help pane shows the documentation for the "strsplit" function.



(Refer Slide Time: 10:38)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The editor pane contains R code for processing a data frame. The console shows the execution of the code, resulting in a character vector with two elements: "2100" and "2850". The environment pane on the right shows the data frame "df" with 13320 observations and 10 variables. The help pane on the right displays the documentation for the `strsplit` function.

```
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22 str(df)
23 df$BHK = as.numeric(df$BHK)
24 str(df)
25
26 df$total_sqft[31]
27 strsplit(df$total_sqft[31], "-")
28
```

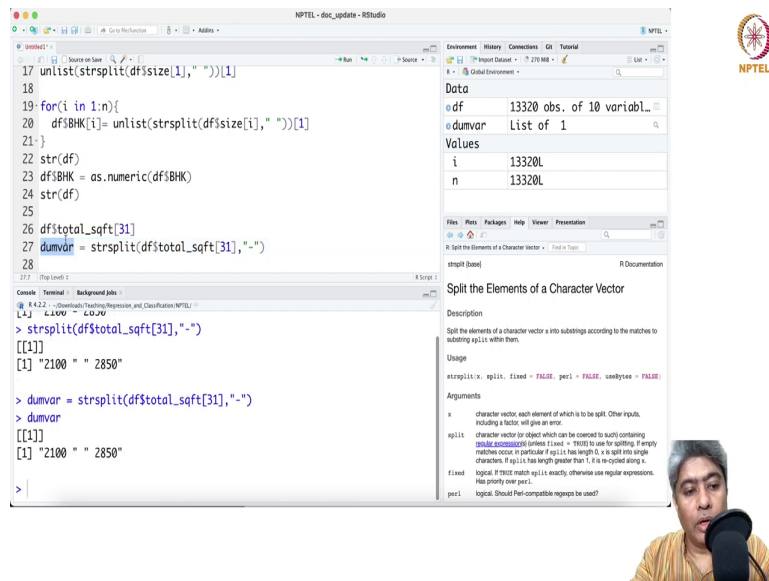
```
> df$total_sqft[31]
[1] "2100 - 2850"
> strsplit(df$total_sqft[31], "-")
[[1]]
[1] "2100" "2850"
```

Data
df 13320 obs. of 10 variables
Values
i 13320L
n 13320L

Split the Elements of a Character Vector
Description
Split the elements of a character vector `x` into substrings according to the matches to substring `split` within them.
Usage
`strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)`
Arguments
`x` character vector, each element of which is to be split. Other inputs, including a factor, will give an error.
`split` character vector or object which can be coerced to such containing regular expressions (unless `fixed = TRUE`) to use for splitting. If empty matches occur, in particular if `split` has length 0, `x` is split into single characters. If `split` has length greater than 1, it is re-cycled along `x`.
`fixed` logical. If `TRUE` match `split` exactly; otherwise use regular expressions. Has priority over `perl`.
`perl` logical. Should Perl-compatible regexps be used?

Of course not this is I have to say 31. Sorry about that. So, this is my df 31 and then if I string split. So, by the dash it was splitted. The first element was 2100 and the second element was 2850, but both are character and it included some space and it was kept in screen. So, let us put it in some dummy variable, ok.

(Refer Slide Time: 11:05)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
17 unlist(strsplit(df$size[1], " "))[1]
18
19 for(i in 1:n){
20   df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22 str(df)
23 df$BHK = as.numeric(df$BHK)
24 str(df)
25
26 df$total_sqft[31]
27 dumvar = strsplit(df$total_sqft[31], "-")
28
```

The console shows the output of the code:

```
> strsplit(df$total_sqft[31], "-")
[[1]]
[1] "2100 " " 2850"

> dumvar = strsplit(df$total_sqft[31], "-")
> dumvar
[[1]]
[1] "2100 " " 2850"

>
```

The Environment pane on the right shows the following objects:

- df**: 13320 obs. of 10 variables
- dumvar**: List of 1

The Values pane shows the values for the objects:

| Object | Value |
|--------|--------|
| i | 13320L |
| n | 13320L |

The R Documentation pane on the right shows the documentation for the `strsplit` function:

Split the Elements of a Character Vector

Description
Split the elements of a character vector `x` into substrings according to the matches to substring `split` within them.

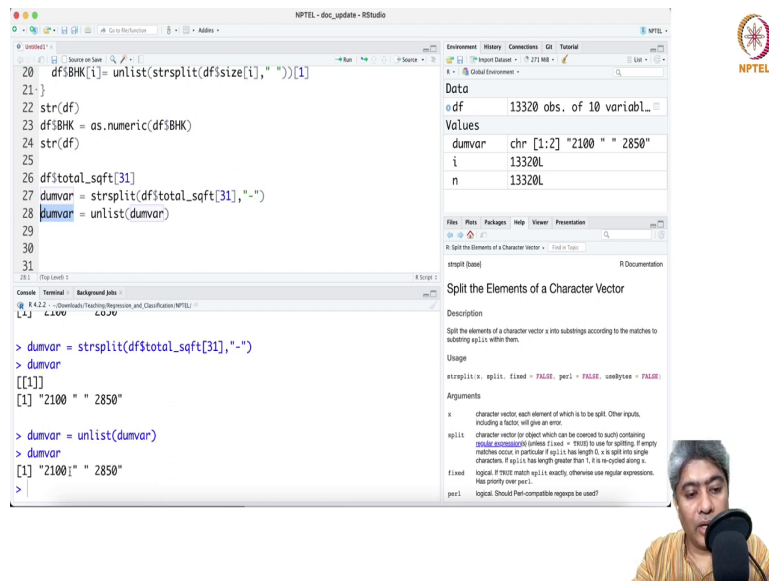
Usage
`strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)`

Arguments
`x`: character vector, each element of which is to be split. Other inputs, including a factor, will give an error.
`split`: character vector or object which can be coerced to such containing regular expressions (unless `fixed = TRUE`) to use for splitting. If empty matches occur, in particular if `split` has length 0, `x` is split into single characters. If `split` has length greater than 1, it is re-cycled along `x`.
`fixed`: logical. If `TRUE` match `split` exactly, otherwise use regular expressions. Also priority over `perl`.
`perl`: logical. Should Perl-compatible regexes be used?

A video inset in the bottom right corner shows a man speaking into a microphone.

So, let us put it in some dummy variable.

(Refer Slide Time: 11:17)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
20 df$BHK[i] = unlist(strsplit(df$size[i], " "))[1]
21 }
22 str(df)
23 df$BHK = as.numeric(df$BHK)
24 str(df)
25
26 df$total_sqft[31]
27 dumvar = strsplit(df$total_sqft[31], "-")
28 dumvar = unlist(dumvar)
29
30
31
```

The console shows the execution of the code:

```
> dumvar = strsplit(df$total_sqft[31], "-")
> dumvar
[[1]]
[1] "2100 " " 2850"


> dumvar = unlist(dumvar)
> dumvar
[1] "2100!" " 2850"
>
```

The Environment pane on the right shows the data structure:

| Variable | Value |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| dumvar | chr [1:2] "2100 " " 2850" |
| i | 13320L |
| n | 13320L |

The right pane displays the documentation for the `strsplit` function, titled "Split the Elements of a Character Vector". It includes a description, usage, and arguments.

NPTEL



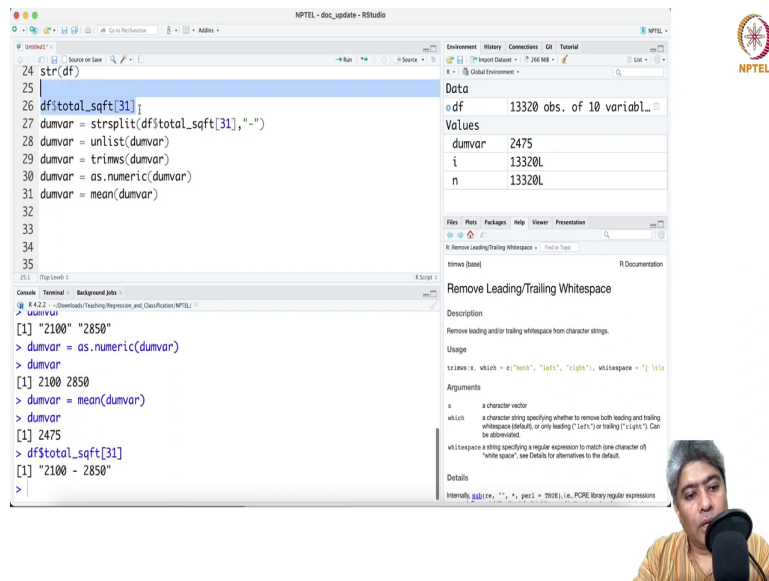
And then first let us unlist the dummy variable; unlist the dummy variable. Now, it has the two element as a vector, but it both of them have some white space added to it.

(Refer Slide Time: 11:39)

A man with grey hair is speaking into a black microphone. He is wearing a yellow and orange striped shirt. The background is a plain, light-colored wall. In the top left corner, there is a small inset image showing a close-up of a person's face with the text "character of)" and "ar expressions" below it.

So, what we will do? We will just trim the white space, ok. We will trim white space, ok. So, you can question mark trim white space yeah you see remove leading or trailing white space that is called trimws. So, now if you run it you can see that you know the white space are removed and it just one single element.

(Refer Slide Time: 12:18)



The screenshot displays an RStudio interface with the following components:

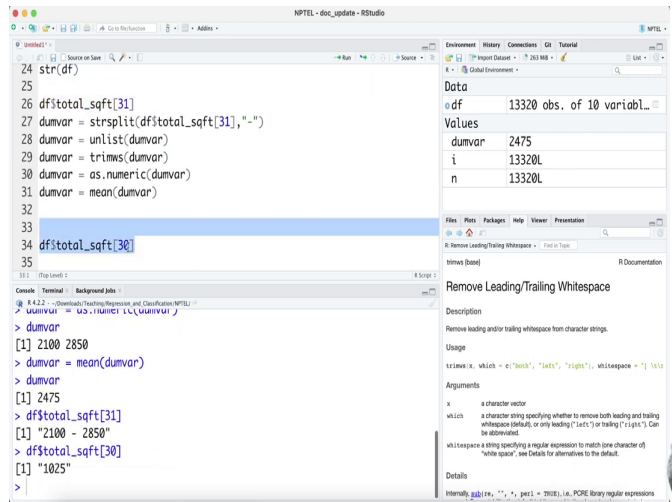
- Source Editor:** Contains R code for data cleaning:

```
24 str(df)
25
26 df$total_sqft[31];
27 dumvar = strsplit(df$total_sqft[31], "-")
28 dumvar = unlist(dumvar)
29 dumvar = trimws(dumvar)
30 dumvar = as.numeric(dumvar)
31 dumvar = mean(dumvar)
32
33
34
35
```
- Environment:** Shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' pane shows 'dumvar' with values 2475, 13320L, and 13320L.
- Console:** Shows the execution of the code, resulting in the output:

```
[1] "2100" "2850"
> dumvar = as.numeric(dumvar)
> dumvar
[1] 2100 2850
> dumvar = mean(dumvar)
> dumvar
[1] 2475
> df$total_sqft[31]
[1] "2100 - 2850"
```
- Documentation:** The right pane shows the documentation for the `trimws` function, titled "Remove Leading/Trailing Whitespace".
- Video Inset:** A small video window in the bottom right corner shows a man speaking into a microphone.

Now, what you can do what we can do is we can convert it to numeric, ok. And then we can take the mean of the 2 2475 is the number. And then what we can do we can just do the same operation over all i because for example, this is the one.

(Refer Slide Time: 12:54)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The editor pane contains the following R code:

```
24 str(df)
25
26 df$total_sqft[31]
27 dumvar = strsplit(df$total_sqft[31], "-")
28 dumvar = unlist(dumvar)
29 dumvar = trimws(dumvar)
30 dumvar = as.numeric(dumvar)
31 dumvar = mean(dumvar)
32
33
34 df$total_sqft[30]
35
```

The console pane shows the following output:

```
> dumvar
[1] 2100 2850
> dumvar = mean(dumvar)
> dumvar
[1] 2475
> df$total_sqft[31]
[1] "2100 - 2850"
> df$total_sqft[30]
[1] "1025"
```

The Environment pane shows the following data:

| Variable | Value |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| dumvar | 2475 |
| i | 13320L |
| n | 13320L |

The right pane shows the documentation for the `trimws` function, titled "Remove Leading/Trailing Whitespace".


Remove Leading/Trailing Whitespace

Description
Remove leading and/or trailing whitespace from character strings.

Usage
`trimws(x, which = c("both", "left", "right"), whitespace = "[\\t\\n]")`

Arguments
`x` a character vector
`which` a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
`whitespace` a string specifying a regular expression to match (one character of) "white space", see Details for alternatives to the default.

Details
Internally `gsub`, `""`, `*`, `perl = TRUE`, i.e., PCRE library regular expressions



And then if I just run it through say 30. It is 1025. Now, if I run it through in this is perfectly fine.

(Refer Slide Time: 13:07)

The screenshot shows an RStudio interface. The console on the left contains the following R code:

```
33
34 df$total_sqft[30]
35 dumvar = strsplit(df$total_sqft[30], "-")
36 dumvar = unlist(dumvar)
37 dumvar = trimws(dumvar)
38 dumvar = as.numeric(dumvar)
39 dumvar = mean(dumvar)
40
41
42
43
44
45
```

The environment pane on the right shows a data frame with 13320 observations and 10 variables. The variables listed are:

| Variable | Value |
|----------|--------|
| dumvar | 1025 |
| i | 13320L |
| n | 13320L |

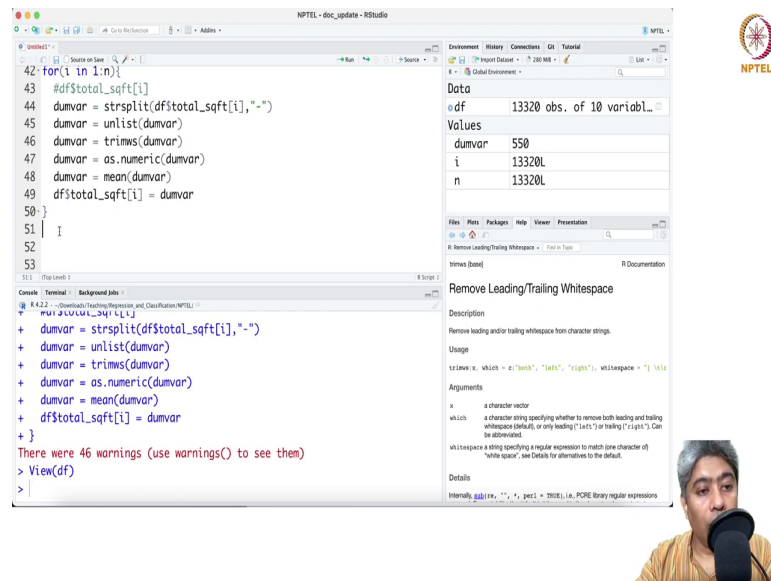
The console output shows the result of the code execution:

```
> df$total_sqft[30]
[1] "1025"
> dumvar = strsplit(df$total_sqft[30], "-")
> dumvar = unlist(dumvar)
> dumvar = trimws(dumvar)
> dumvar = as.numeric(dumvar)
> dumvar = mean(dumvar)
dumvar
[1] 1025
```

The right-hand pane shows the documentation for the `removeLeadingTrailingWhitespace` function, which removes leading and trailing whitespace from character strings.

I just convert it to numeric that should be fine, but if I just even if I run it through this process then it is just still 10020 , 1025 square feet. So, I can run this operation through the entire thing ok.

(Refer Slide Time: 13:24)



The screenshot shows an RStudio interface with the following components:

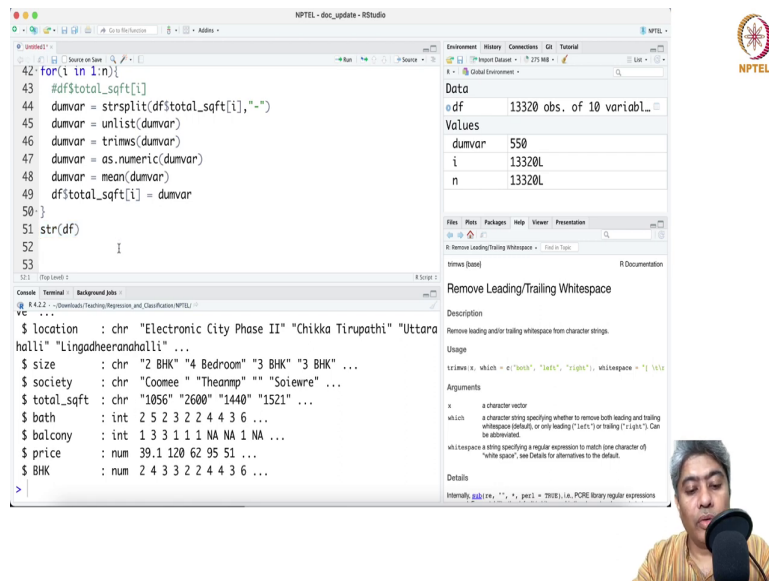
- Source Editor:** Contains R code for a loop that processes data. The code is as follows:

```
42: for(i in 1:n){
43:   #df$total_sqft[i]
44:   dumvar = strsplit(df$total_sqft[i], "-")
45:   dumvar = unlist(dumvar)
46:   dumvar = trimws(dumvar)
47:   dumvar = as.numeric(dumvar)
48:   dumvar = mean(dumvar)
49:   df$total_sqft[i] = dumvar
50: }
51: |
52:
53:
```
- Environment:** Shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' section shows 'dumvar' as 550, 'i' as 13320L, and 'n' as 13320L.
- Console:** Shows the execution of the code, with a message indicating 46 warnings. The output is as follows:

```
+ dumvar = strsplit(df$total_sqft[i], "-")
+ dumvar = unlist(dumvar)
+ dumvar = trimws(dumvar)
+ dumvar = as.numeric(dumvar)
+ dumvar = mean(dumvar)
+ df$total_sqft[i] = dumvar
+ }
There were 46 warnings (use warnings() to see them)
> View(df)
>
```
- Documentation:** Shows the documentation for the 'trimws' function, which removes leading and trailing whitespace from character strings.

For i in 1 is to n. So, let me just copy this and paste it and ok I do not need this guy. I just need to put i here and then into finally, df total square feet i you just put this dummy variable and then you are done essentially. So, let me run this operation there 46 warnings, but you can see that now more or less there being there are no much. So, those 31 is 2475.754. So, now most of the cases have been taken care of no dash business are there. So, there is a big problem solving.

(Refer Slide Time: 14:37)



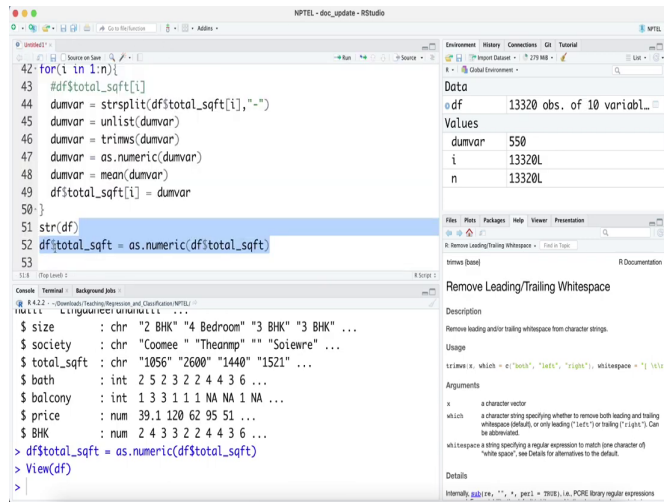
The screenshot displays an RStudio interface with the following components:

- Source Editor:** Contains R code for cleaning the 'total_sqft' column.

```
42 for(i in 1:n){
43   #df$total_sqft[i]
44   dumvar = strsplit(df$total_sqft[i], "-")
45   dumvar = unlist(dumvar)
46   dumvar = trimws(dumvar)
47   dumvar = as.numeric(dumvar)
48   dumvar = mean(dumvar)
49   df$total_sqft[i] = dumvar
50 }
51 str(df)
52
53
```
- Environment:** Shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' section shows 'dumvar' as 550, 'i' as 13320L, and 'n' as 13320L.
- Console:** Displays the output of the R code, showing the structure of the data frame 'df' with columns: location, size, society, total_sqft, bath, balcony, price, and BHK.
- Documentation:** A sidebar showing the documentation for the 'trimws()' function, titled 'Remove Leading/Trailing Whitespace'.
- Inset:** A small video inset in the bottom right corner shows a person speaking into a microphone.

But still if you look into the structure of the df I think I have to just check whether it is still total character.


(Refer Slide Time: 14:48)



The image shows the RStudio interface with the following components:

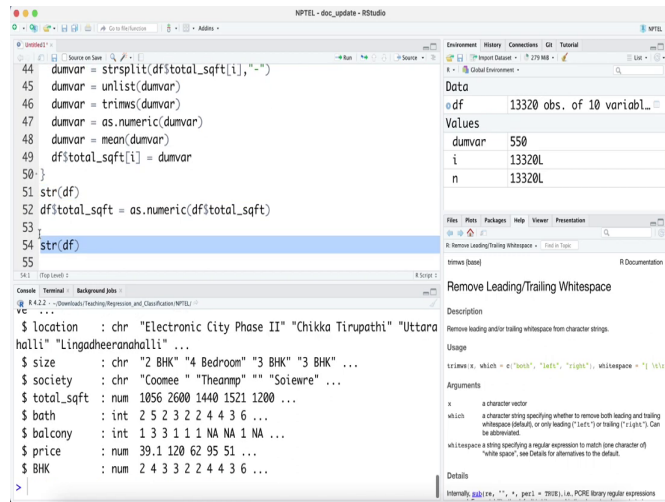
- Source Editor:** Contains R code for processing a data frame. The code includes a loop to process each row, splitting a string, trimming it, converting it to numeric, and then converting it back to a character vector. The final line is `df$total_sqft = as.numeric(df$total_sqft)`.
- Environment:** Shows the data frame `df` with 13320 observations and 10 variables. The variable `dumvar` is of type `factor` with 550 levels. The variable `i` is of type `integer` with 13320 levels. The variable `n` is of type `integer` with 13320 levels.
- Console:** Shows the output of the code execution, including the structure of the data frame and the results of the `as.numeric` conversion.
- Documentation:** Shows the documentation for the `strtrim` function, which removes leading and trailing whitespace from character strings.

NPTEL



So, what we have to do? We have to say ok boss we need to put as dot numeric alright. And now we can see probably they have indeed gone into as dot numeric ok.

(Refer Slide Time: 15:08)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
44 dumvar = strsplit(df$total_sqft[, "-"])
45 dumvar = unlist(dumvar)
46 dumvar = trimws(dumvar)
47 dumvar = as.numeric(dumvar)
48 dumvar = mean(dumvar)
49 df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
```


The console shows the output of the code, displaying the structure of the data frame:


```
$ location : chr "Electronic City Phase II" "Chikka Tirupathi" "Uttara
halli" "Lingadheeranahalli" ...
$ size : chr "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
$ society : chr "Coomee" "Theamp" "" "Soiewre" ...
$ total_sqft : num 1056 2600 1440 1521 1200 ...
$ bath : int 2 5 2 3 2 2 4 4 3 6 ...
$ balcony : int 1 3 3 1 1 1 NA NA 1 NA ...
$ price : num 39.1 120 62 95 51 ...
$ BHK : num 2 4 3 3 2 2 4 4 3 6 ...
```

The right-hand pane shows the "Data" tab with the following summary:

| Variable | Value |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| dumvar | 550 |
| i | 13320L |
| n | 13320L |

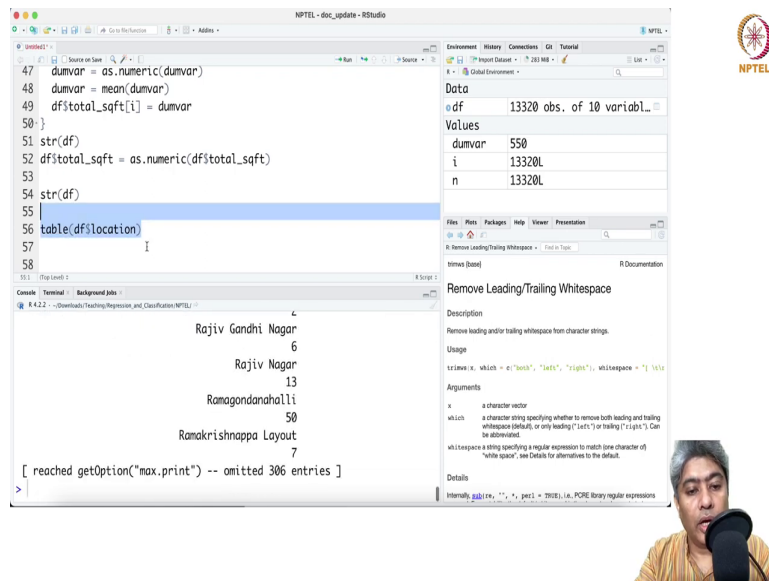
The "Files" tab shows the "Remove Leading/Trailing Whitespace" function, which is used to remove leading and trailing whitespace from character strings.





So, total square feet is now numeric. So, this is a big problem solving. Next we have to handle the location ok. So, location was location is very important and we have to handle the location. Why? Because there are few location few new quite a few location where the number of instances were very few.

(Refer Slide Time: 15:42)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The editor pane contains the following R code:

```
47 dumvar = as.numeric(dumvar)
48 dumvar = mean(dumvar)
49 df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 table(df$location)
57
58
```



The console pane shows the output of the code:

```
4
Rajiv Gandhi Nagar
6
Rajiv Nagar
13
Ramagondanahalli
50
Ramakrishnappa Layout
7
[ reached getOption("max.print") -- omitted 306 entries ]
>
```

The Environment pane shows the data frame "df" with 13320 observations and 10 variables. The "Values" pane shows the values for "dumvar" (550), "i" (13320L), and "n" (13320L).

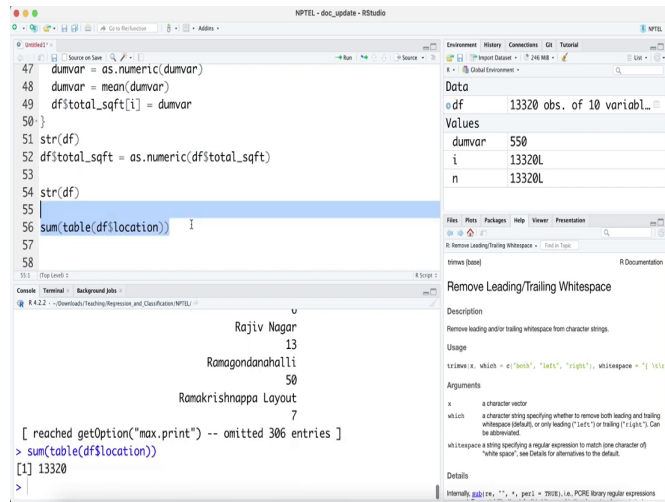
The right sidebar shows the "Remove Leading/Trailing Whitespace" package documentation. The description states: "Remove leading and/or trailing whitespace from character strings." The usage is: "straw <- which = c('left', 'right', 'both'), whitespace = ' ' | '\n'". The arguments are: "x" is a character vector, "which" is a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated. "whitespace" is a string specifying a regular expression to match (one character of) "white space", see Details for alternatives to the default.

Internally: `gsub(x, "", y, perl = TRUE)`, i.e., POSIX regular expressions



So, how to handle it? df dollar location and first what we can do? We can just say table. So, it will give me a table of each location how many instances are there.

(Refer Slide Time: 16:03)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
47 dumvar = as.numeric(dumvar)
48 dumvar = mean(dumvar)
49 df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 sum(table(df$location))
57
58
```

The console shows the output of the code:

```


      Rajiv Nagar
      13
Ramagondanahalli
      50
Ramakrishnappa Layout
      7
[ reached getOption("max.print") -- omitted 306 entries ]
> sum(table(df$location))
[1] 13320
>
```

The environment pane on the right shows the following data:

| Variable | Value |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| dumvar | 550 |
| i | 13320L |
| n | 13320L |

The right pane shows the documentation for the `str_remove` function, which removes leading and trailing whitespace from character strings.

NPTEL



So, if I just say sum it will give me 13,320 you see, but I do not want it. I want the table in each locations are there.

(Refer Slide Time: 16:12)

The screenshot shows an RStudio interface with the following components:

- Source Editor:** Contains R code for data cleaning:

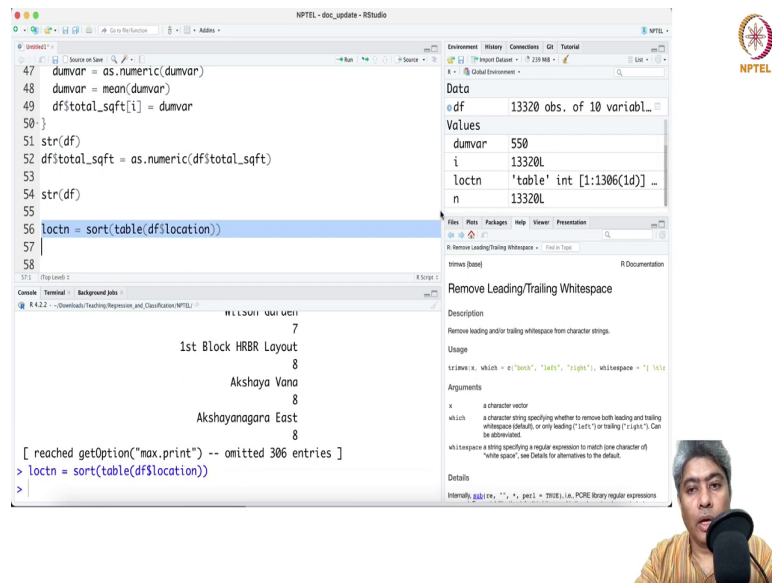

```
47 dumvar = as.numeric(dumvar)
48 dumvar = mean(dumvar)
49 df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 sort(table(df$location))
57
58
```
- Environment:** Shows a data frame 'df' with 13320 observations and 10 variables. A preview of the data is shown:

| Variables | Values |
|-----------|--------|
| dumvar | 550 |
| i | 13320L |
| n | 13320L |
- Console:** Shows the output of the R code:


```
Wilson Garden
7
1st Block HRBR Layout
8
Akshaya Vana
8
Akshayanagara East
8
[ reached getOption("max.print") -- omitted 306 entries ]
>
```
- Help:** The 'Remove Leading/Trailing Whitespace' help page is open, showing the 'trimws()' function and its arguments.

And then I can just sort them.

(Refer Slide Time: 16:20)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains the following R code:

```
47 dumvar = as.numeric(dumvar)
48 dumvar = mean(dumvar)
49 df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 locn = sort(table(df$location))
57
58
```

The console pane shows the output of the code:


```
7
1st Block HRBR Layout
8
Akshaya Vana
8
Akshayanagara East
8
[ reached getOption("max.print") -- omitted 306 entries ]
> locn = sort(table(df$location))
>
```

The environment pane shows the following data:

| Variable | Value |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| dumvar | 550 |
| i | 13320L |
| locn | 'table' int [1:1306(1d)] |
| n | 13320L |

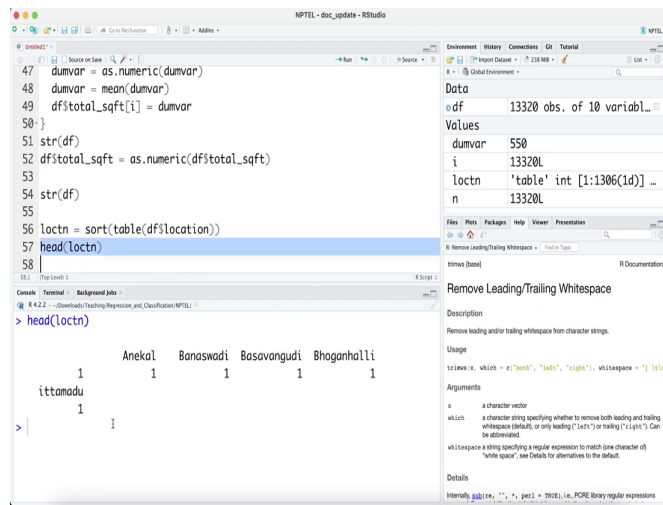
The right pane shows the documentation for the `str_remove` function, which removes leading and trailing whitespace from character strings.

NPTEL



And let me put it in some variable or some object locn location ok.

(Refer Slide Time: 16:33)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The editor pane contains the following R code:

```
47 dumvar = as.numeric(dumvar)
48 dumvar = mean(dumvar)
49 df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 loctn = sort(table(df$location))
57 head(loctn)
58
```

The console shows the output of the `head(loctn)` command:

```
> head(loctn)
      1      1      1      1      1
ittanadu 1
```

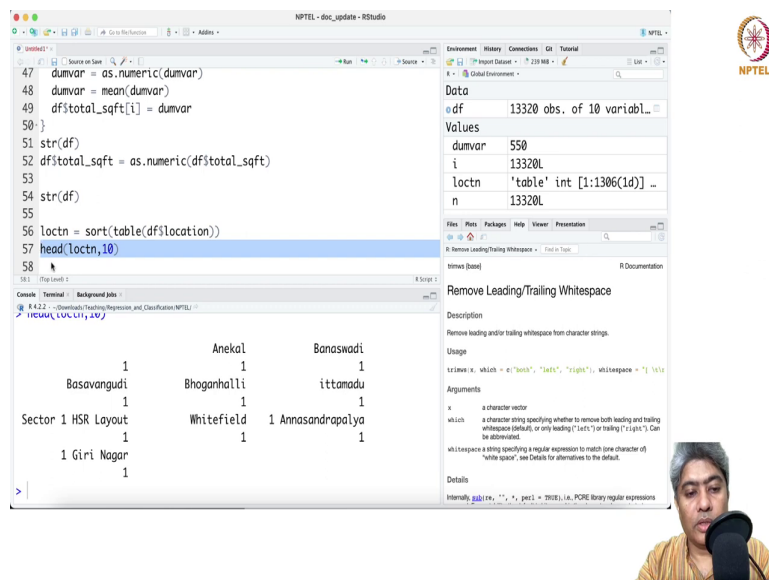
The environment pane on the right shows the following data:

| Variable | Value |
|----------|-----------------------------|
| df | 13320 obs. of 10 variables |
| dumvar | 550 |
| i | 13320L |
| loctn | 'table' int [1:1306(1d)] .. |
| n | 13320L |

The right pane also shows the documentation for the `str_remove` function, which removes leading and trailing whitespace from character strings.



(Refer Slide Time: 16:44)



The screenshot shows the RStudio interface. The script editor on the left contains the following R code:

```
47 dumvar = as.numeric(dumvar)
48 dumvar = mean(dumvar)
49 df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 locn = sort(table(df$location))
57 head(locn, 10)
58
```

The console on the right displays the output of the code, showing a table of locations and their counts:

| | Anekal | Banaswadi |
|---------------------|-------------|-------------------|
| 1 | 1 | 1 |
| Basavangudi | Bhoganhalli | ittamadu |
| 1 | 1 | 1 |
| Sector 1 HSR Layout | Whitefield | 1 Annasandrapalya |
| 1 | 1 | 1 |
| 1 Giri Nagar | | |
| 1 | | |

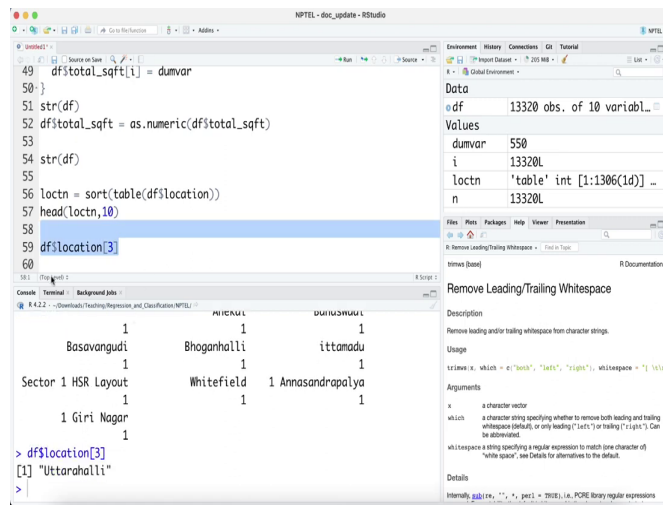
The environment pane on the right shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' pane shows the distribution of 'dumvar' (550), 'i' (13320), 'locn' (table with 13320 unique values), and 'n' (13320).

The R documentation pane on the right shows the 'Remove Leading/Trailing Whitespace' function, which removes leading and trailing whitespace from character strings.

And so, if you just say head of locn then it will be like few first few maybe 10. These are all places where you have only one instance. So, these are the cases where you have only single instances and there are one instances where you have no space like location is not known. So, you just have some quote unquote kind of thing.

So, what we can do wherever the location is belongs to this say Anekal or any one Giri Nagar or something the location is less than say 10 or 20 or 15 some number. We consider is or say 20 let us take 20 then we will consider it as that we do not have enough instances in those case we will put it into others ok. So, how can we do that?

(Refer Slide Time: 17:48)



The image shows the RStudio interface with the following components:

- Source Editor:** Contains R code for data manipulation:

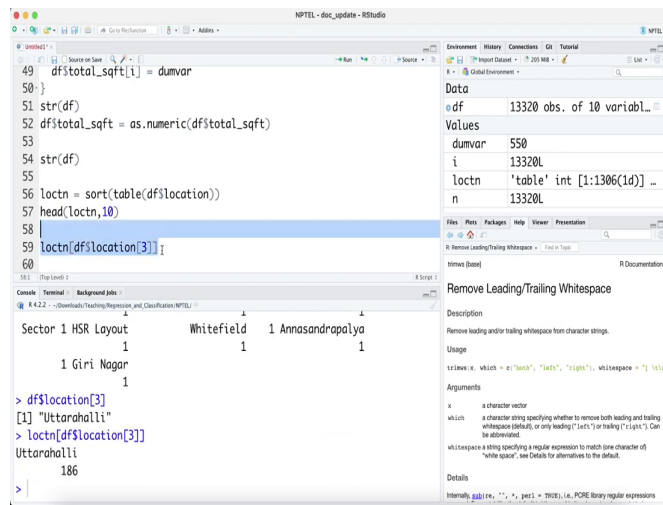
```
49 df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 locn = sort(table(df$location))
57 head(locn, 10)
58
59 df$location[3]
```
- Environment:** Shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' pane shows the structure of 'dumvar' (550), 'i' (13320L), 'locn' (table of integers), and 'n' (13320L).
- Console:** Displays the output of the code, showing a table of location counts and the result of the indexing operation:

```
> df$location[3]
[1] "Uttarahalli"
```
- Help Pane:** Displays the documentation for the `str_remove` function, titled "Remove Leading/Trailing Whitespace".



So, let us take one particular location say 3 ok on the third case is Uttarahalli.

(Refer Slide Time: 18:01)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The editor pane contains the following R code:

```
49 df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 loctn = sort(table(df$location))
57 head(loctn, 10)
58
59 loctn[df$location[3]]
60
```

The console pane shows the output of the code:

```
Sector 1 HSR Layout      Whitefield      1 Annasandrapalya
      1
      1 Giri Nagar
      1
> df$location[3]
[1] "Uttarahalli"
> loctn[df$location[3]]
Uttarahalli
      186
```

The Environment pane on the right shows the following data:

| Variable | Value |
|----------|-----------------------------|
| df | 13320 obs. of 10 variables |
| dumvar | 550 |
| i | 13320L |
| loctn | 'table' int [1:1306(1d)] .. |
| n | 13320L |

The right pane shows the documentation for the `str_remove` function, which removes leading and trailing whitespace from character strings.



Now, in the location; so, this is the name of the case in the location table if I just put that guy 186; that means, out of 13,320 cases instances there are 186 house, which belongs to the location Uttarahalli ok.

(Refer Slide Time: 18:27)

[illegible]

Now, if what we will try to do? We will try to secure that good boss if this location is say less than 20 then we will say this location as others ok. This is something what we will trying want to do, but we do not want to do it here we will do it in a little bit more fashionable way. We have to also take care of one problem that there is a one quote unquote thing is there. So, and how location table is handling this guy? So, if I just say df location.

(Refer Slide Time: 19:17)

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for data cleaning:

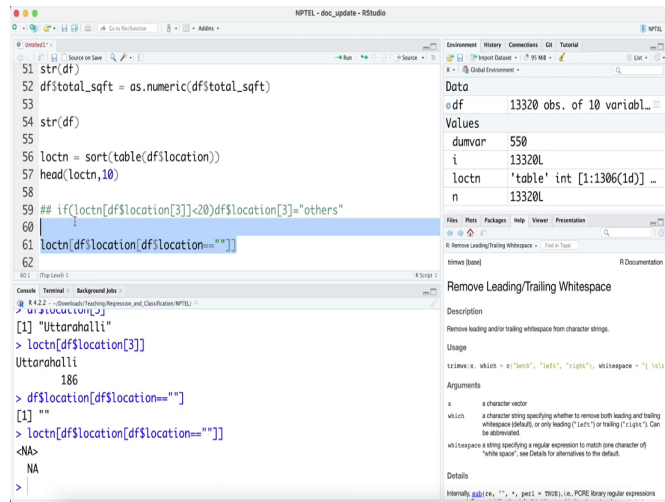

```
df$total_sqft = as.numeric(df$total_sqft)
df$location[df$location == ""] = NA
```
- Console:** Shows the output of the code:


```
1 Giri Nagar
1
> df$location[3]
[1] "Uttarahalli"
> locn[df$location[3]]
Uttarahalli
186
> df$location[df$location==""]
[1]
>
```
- Environment Pane:** Displays the structure of the data frame 'df':

| Variable | Value |
|----------|-----------------------------|
| df | 13320 obs. of 10 variables |
| Values | |
| dimvar | 550 |
| i | 13320L |
| locn | 'table' int [1:1306(1d)] .. |
| n | 13320L |
- Files Pane:** Shows the file 'NPTEL.doc_update.R'.
- Help Pane:** Displays the documentation for 'Remove Leading/Trailing Whitespaces', including a description, usage, arguments, and details.

If I just say df location equal to quote unquote and df dollar location, ok. So, there is one case.

(Refer Slide Time: 19:44)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 locn = sort(table(df$location))
57 head(locn, 10)
58
59 ## if[locn[df$location[3]]<20]df$location[3]="others"
60 {
61   locn[df$location[df$location==""]]
62 }
```


The console shows the output of the code:

```
[1] "Uttarahalli"
> locn[df$location[3]]
Uttarahalli
186
> df$location[df$location==""]
[1] ""
> locn[df$location[df$location==""]]
<NA>
NA
>
```

The Environment pane on the right shows the data frame "df" with 13320 observations and 10 variables. The "Values" pane shows the distribution of the "location" variable:

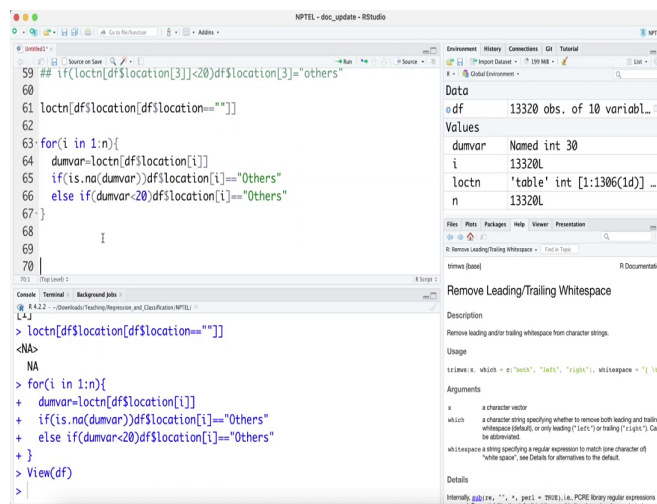
| Variable | Value |
|----------|-----------------------------|
| df | 13320 obs. of 10 variables |
| dumvar | 550 |
| i | 13320L |
| locn | 'table' int [1:1306(1d)] .. |
| n | 13320L |

The "Remove Leading/Trailing Whitespace" documentation is also visible on the right.



Now, if I just put it there if I just put it there then it is a NA; that means, and we have to; that means, handle NA inside the if condition and that could be a bit of a challenge that could be a bit of a challenge.

(Refer Slide Time: 20:09)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The editor pane contains R code for processing a data frame 'df'. The code uses a loop to iterate over each row 'i' from 1 to n. For each row, it extracts the value from the 'location' column and assigns it to a variable 'dumvar'. It then checks if 'dumvar' is NA. If it is NA, it assigns 'Others' to the 'location' column. If 'dumvar' is less than 20, it also assigns 'Others' to the 'location' column. Otherwise, it leaves the value as is. The console shows the execution of the code, resulting in NA values for the 'location' column. The environment pane on the right shows the data frame 'df' with 13320 observations and 10 variables. The 'location' variable is of type 'table' with integer values from 1 to 1306. The 'dumvar' variable is of type 'integer' with values from 1 to 30. The 'n' variable is of type 'integer' with a value of 13320. The console also shows the output of the code, which is NA for the 'location' column.

```
59 ## if(loctn[df$location[3]]<20)df$location[3]="Others"
60
61 loctn[df$location[df$location==""]]
62
63 for(i in 1:n){
64   dumvar=loctn[df$location[i]]
65   if(is.na(dumvar))df$location[i]="Others"
66   else if(dumvar<20)df$location[i]="Others"
67 }
68
69
70
71
```

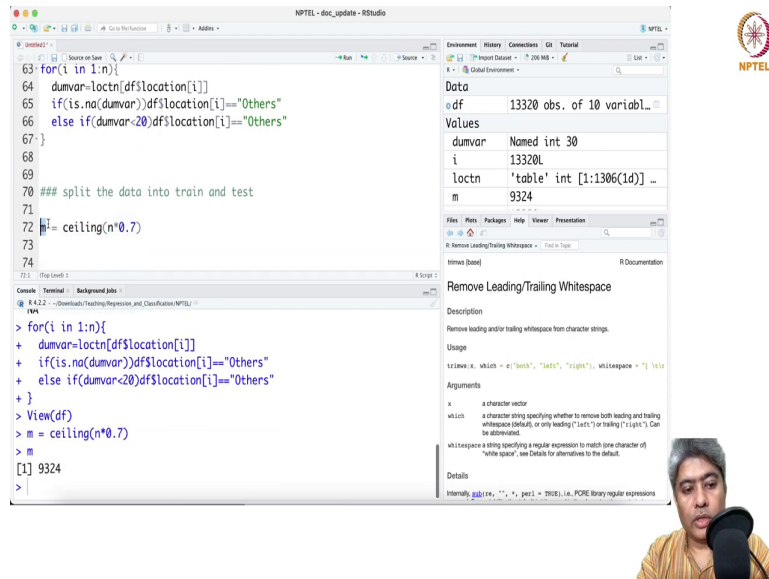
```
> loctn[df$location[df$location==""]]
<NA>
NA
> for(i in 1:n){
+   dumvar=loctn[df$location[i]]
+   if(is.na(dumvar))df$location[i]="Others"
+   else if(dumvar<20)df$location[i]="Others"
+ }
> View(df)
>
```



So, there will be one NA that we will encounter. So, let us you know try to handle it for i in 1 is to n. First thing what we will do? We will take the location of the ith instances and put it into the location table and we will see what is the count give me the count and that count I will put it as a dum variable. So, either it will give me NA or it will give me an some number ok.

So, if dum variable is dot na then we will say ok boss in this particular case you just put others. Else if dum variable is less than 20; that means, the number of instances in the entire data set for that particular location is less than 20 then also you put it others else you just leave it there you do not worry ok. So, you know if I just run it, it run very fast and you see if there any others have come there may be one or something, ok we will see. So, at least I think the location thing is now kind of ok.

(Refer Slide Time: 22:15)



The screenshot shows an RStudio interface with the following components:

- Source Editor:** Contains R code for data cleaning and splitting:

```
63 for(i in 1:n){
64   dumvar=loctn[df$location[i]]
65   if(is.na(dumvar))df$location[i]="Others"
66   else if(dumvar<20)df$location[i]="Others"
67 }
68
69
70 ## split the data into train and test
71
72 m= ceiling(n*0.7)
73
74
```
- Environment:** Shows the data frame 'df' with 13320 observations and 10 variables. The 'Values' section lists:
 - dumvar: Named int 30
 - i: 13320L
 - loctn: 'table' int [1:1306(1d)] ..
 - m: 9324
- Console:** Shows the execution of the code:

```
> for(i in 1:n){
+   dumvar=loctn[df$location[i]]
+   if(is.na(dumvar))df$location[i]="Others"
+   else if(dumvar<20)df$location[i]="Others"
+ }
> View(df)
> m = ceiling(n*0.7)
> m
[1] 9324
>
```
- Documentation Panel:** Displays the documentation for the `removeLeadingTrailingWhitespace` function, including its description, usage, arguments, and details.

In the bottom right corner, there is a small video feed of a person speaking into a microphone.

We as most of the data cleaning thing has been done. Now, we will do the model training. So, let us split the data in train and test. Split the data into train and test ok. Split the data into train and test. Now, so, the first thing I will do m equal to ceiling n is to 0.7 ok. So, that will give me. So, out of 13,320, 3 9324 instances we will take as a test data set sorry training data set. We will take it randomly and rest we will take it as a test data set.

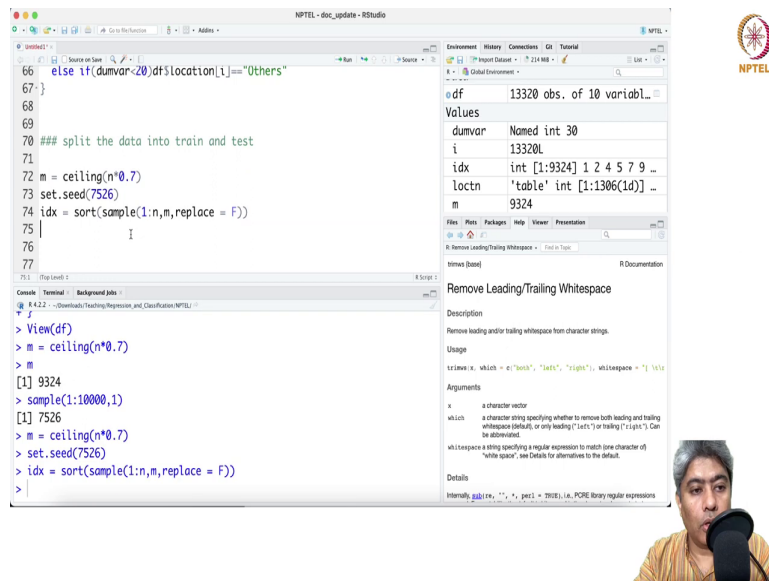
(Refer Slide Time: 23:17)

The screenshot displays the RStudio IDE with the following components:

- Source Editor:** Contains R code for cleaning a dataset 'df'. It uses a for loop to check for missing values in 'dumvar' and 'location' columns, replacing them with 'Others'. It then splits the data into training and testing sets using `sample()` and `ceiling()`.
- Environment:** Shows the data frame 'df' with 13320 observations and 10 variables. It lists the objects 'dumvar', 'location', and 'm'.
- Console:** Shows the execution of the code, resulting in the dimensions of 'df' (9324 rows, 10 columns) and the first few rows of the data.
- Documentation:** The 'Remove Leading/Trailing Whitespace' function is highlighted, showing its description and usage.
- Video Inset:** A small video window in the bottom right corner shows a man speaking into a microphone.

So, we have to first set the seed. So, how we will do that? So, typically the way we do it I do it is 1 is to say 10,000 I just draw one sample all I need 7526.

(Refer Slide Time: 23:31)



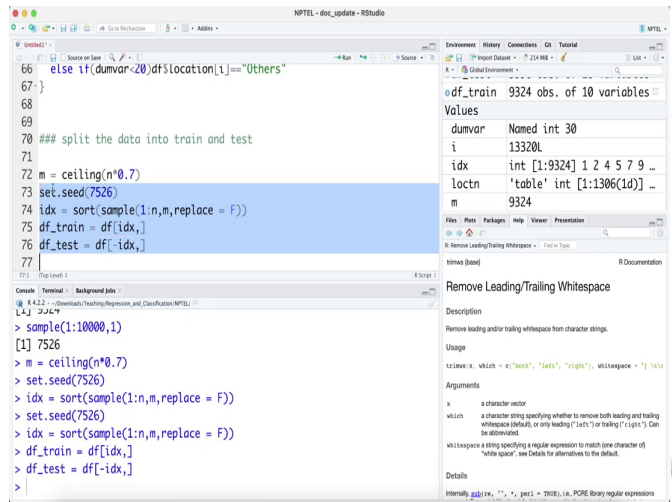
The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for splitting data into train and test sets. The code includes a function definition and a call to `sample()` to draw a random sample.
- Environment:** Shows the data frame `df` with 13320 observations and 10 variables. The variables `dumvar`, `i`, `idx`, `loctn`, and `m` are listed with their respective data types and values.
- Console:** Displays the execution of the code, including the output of `View(df)`, `m = ceiling(n*0.7)`, `set.seed(7526)`, and `sort(sample(1:n,m,replace = F))`.
- Help Window:** Shows the documentation for the `str_remove()` function, which removes leading and trailing whitespace from character strings.

A small inset image of a person speaking is visible in the bottom right corner of the RStudio window.

And that I will set up as a seed. Then what I am going to do? I am going to draw sample from 1 is to n, m, many sample and replace equal to false and then I must do a. It is a one time sort do not worry about that one time sort is fine and I just do idx. So, these are the idx I am going to draw and if you use the same seed you will get the same data set same split exactly my result.

(Refer Slide Time: 24:13)



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for splitting data. Lines 74-76 are highlighted in blue:

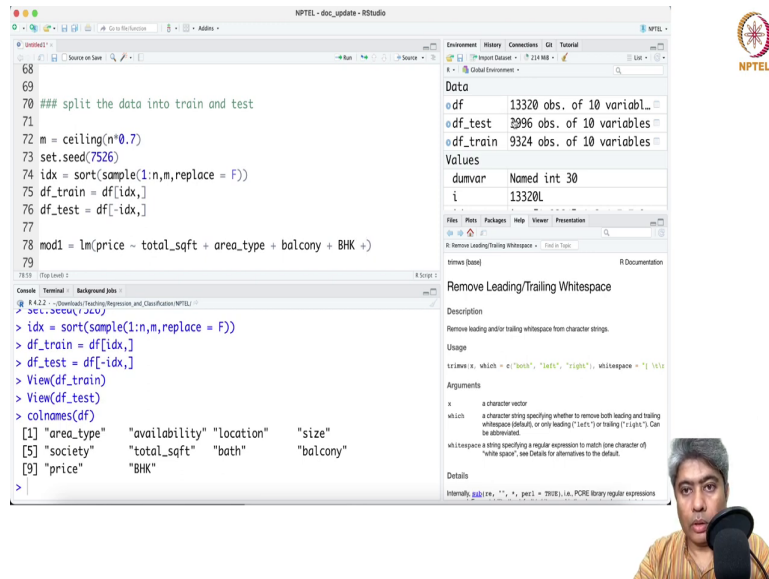
```
74 idx = sort(sample(1:n,m,replace = F))
75 df_train = df[idx,]
76 df_test = df[-idx,]
```
- Environment Pane:** Shows the data frame `df_train` with 9324 observations and 10 variables. The variables listed are `dumvar` (Named int 30), `i` (13320L), `idx` (int [1:9324] 1 2 4 5 7 9 ...), `loctn` ('table' int [1:1306(1d)] ...), and `m` (9324).
- Console:** Shows the execution of the following commands:

```
> sample(1:10000,1)
[1] 7526
> m = ceiling(n*0.7)
> set.seed(7526)
> idx = sort(sample(1:n,m,replace = F))
> set.seed(7526)
> idx = sort(sample(1:n,m,replace = F))
> df_train = df[idx,]
> df_test = df[-idx,]
```
- Help Pane:** Displays the documentation for the `str_remove` function, titled "Remove Leading/Trailing Whitespace".

In the bottom right corner, there is a small video feed of a person speaking into a microphone.

And your result will be same. So, df train will be df with these idx comma. So, these are the row values and column values all columns will come in and df test will be minus of these idx. So, now this is df train ok and this is df test, ok alright next what I am going to do is we are going to fit our first model `mod1 lm price`.

(Refer Slide Time: 25:10)



The screenshot shows an RStudio interface with the following components:

- Source Editor:** Contains R code for splitting data and fitting a model.

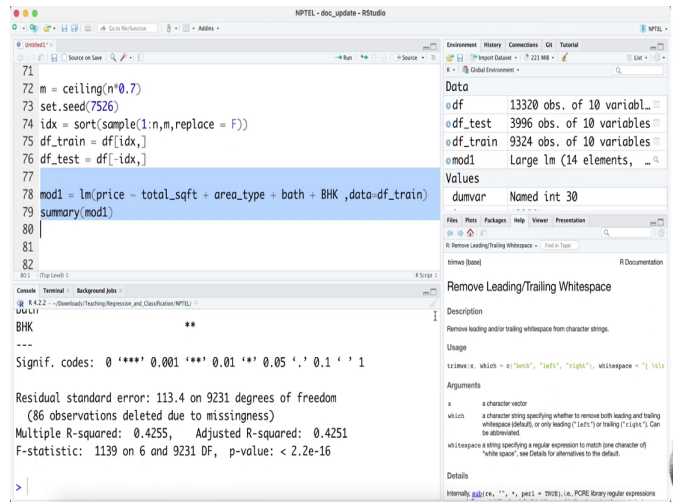
```
68  
69  
70 ## split the data into train and test  
71  
72 m = ceiling(n*0.7)  
73 set.seed(7526)  
74 idx = sort(sample(1:n,m,replace = F))  
75 df_train = df[idx,]  
76 df_test = df[-idx,]  
77  
78 mod1 = lm(price ~ total_sqft + area_type + balcony + BHK +)  
79
```
- Environment:** Shows the data objects: `df` (13320 obs. of 10 variables), `df_test` (2996 obs. of 10 variables), and `df_train` (9324 obs. of 10 variables). It also shows a variable `i` with value 13320L.
- Console:** Shows the execution of the code, including the output of `colnames(df)`:

```
> set.seed(7526)  
> idx = sort(sample(1:n,m,replace = F))  
> df_train = df[idx,]  
> df_test = df[-idx,]  
> View(df_train)  
> View(df_test)  
> colnames(df)  
[1] "area_type" "availability" "location" "size"  
[5] "society" "total_sqft" "bath" "balcony"  
[9] "price" "BHK"
```
- Help:** Displays the documentation for the `remove_leading_trailing_whitespace` function.

NPTEL logo is visible in the top right corner.

Let me just get the colnames of the df so, price. So, that you know we do not make spelling mistake price total square feet plus area type plus balcony plus and BHK plus your bathroom. I think we have to take bathroom the sometime there are bathroom and then BHK there are yeah.

(Refer Slide Time: 26:25)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The script editor contains the following R code:

```
71  
72 m = ceiling(n*0.7)  
73 set.seed(7526)  
74 idx = sort(sample(1:n,m,replace = F))  
75 df_train = df[idx,]  
76 df_test = df[-idx,]  
77  
78 mod1 = lm(price ~ total_sqft + area_type + bath + BHK, data=df_train)  
79 summary(mod1)  
80  
81  
82
```

The console output shows the summary of the linear model:



```
BHK  
----  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 113.4 on 9231 degrees of freedom  
(86 observations deleted due to missingness)  
Multiple R-squared:  0.4255,    Adjusted R-squared:  0.4251  
F-statistic: 1139 on 6 and 9231 DF,  p-value: < 2.2e-16
```

The environment pane on the right shows the following data:

| Object | Class | Attributes |
|----------|------------|----------------------------|
| df | data.frame | 13320 obs. of 10 variables |
| df_test | data.frame | 3996 obs. of 10 variables |
| df_train | data.frame | 9324 obs. of 10 variables |
| mod1 | lm | Linear model object |

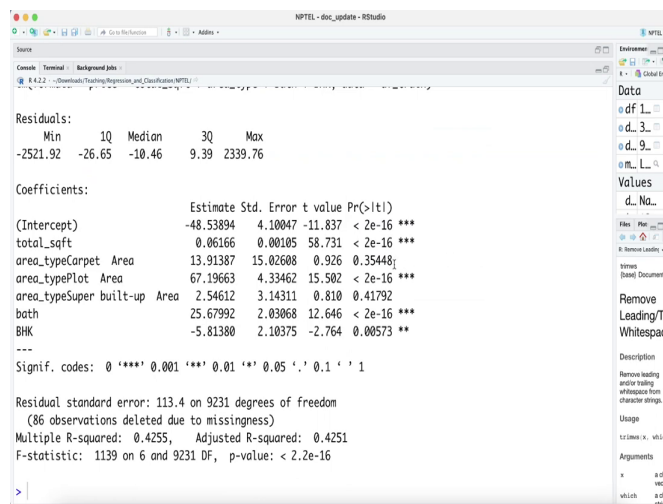
The console also shows the output of the `summary(mod1)` command, which includes the following information:

```
Linear model using ordinary least squares  
Model: lm(  
  formula = price ~ total_sqft + area_type + bath + BHK,  
  data = df_train  
)  
Call: lm(  
  formula = price ~ total_sqft + area_type + bath + BHK,  
  data = df_train  
)  
Linear model using ordinary least squares  
1 (Intercept)  
total_sqft  
area_type  
bath  
BHK  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 113.4 on 9231 degrees of freedom  
(86 observations deleted due to missingness)  
Multiple R-squared:  0.4255,    Adjusted R-squared:  0.4251  
F-statistic: 1139 on 6 and 9231 DF,  p-value: < 2.2e-16
```



So, we can take bathroom instead of bathroom BHK and data equal to df train ok and at the end you just say summary of mod1. Remember note that I did not put location yet ok. I did not put location yet.

(Refer Slide Time: 26:54)



```
Residuals:
    Min       1Q   Median       3Q      Max
-2521.92  -26.65  -10.46    9.39  2339.76

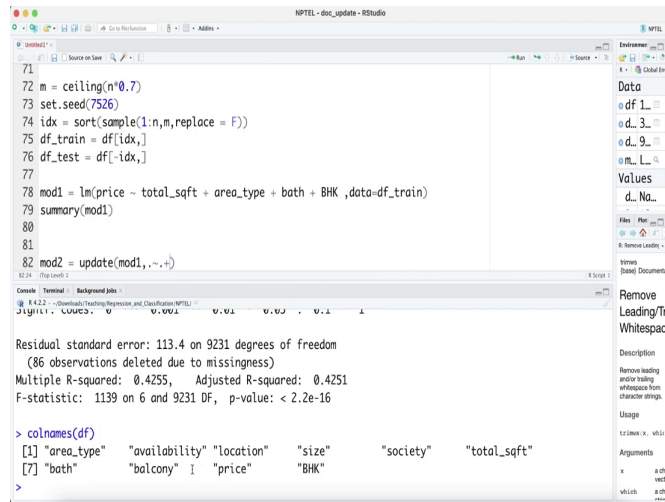
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -48.53894    4.10047  -11.837 < 2e-16 ***
total_sqft      0.06166    0.00105   58.731 < 2e-16 ***
area_typeCarpet Area  13.91387    15.02608    0.926  0.35448
area_typePlot Area  67.19663    4.33462   15.502 < 2e-16 ***
area_typeSuper built-up Area  2.54612    3.14311    0.810  0.41792
bath           25.67992    2.03068   12.646 < 2e-16 ***
BHK            -5.81380    2.10375   -2.764  0.00573 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 113.4 on 9231 degrees of freedom
(86 observations deleted due to missingness)
Multiple R-squared:  0.4255,    Adjusted R-squared:  0.4251
F-statistic: 1139 on 6 and 9231 DF,  p-value: < 2.2e-16
```



So, if you see either put area total square feet does have a strong correlation, strong effect very strong effect t value is 58, t value is very small area type plot type area has quite significant effect plot area the people the things, which are plot area they have a very little bit premium price people are paying. If bath is a number of bathroom increases then the premium or the price goes up, but if the number of BHK increases it looks like it is a negative this is bit worry the some I do not know why it is negative? And adjusted R squared we got 0.4251 with no location.

(Refer Slide Time: 28:00)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
71  
72 m = ceiling(n*0.7)  
73 set.seed(7526)  
74 idx = sort(sample(1:n,m,replace = F))  
75 df_train = df[idx,]  
76 df_test = df[-idx,]  
77  
78 mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)  
79 summary(mod1)  
80  
81  
82 mod2 = update(mod1, ~.+)
```



The console output shows the summary of the linear model:

```
Residual standard error: 113.4 on 9231 degrees of freedom  
(36 observations deleted due to missingness)  
Multiple R-squared: 0.4255, Adjusted R-squared: 0.4251  
F-statistic: 1139 on 6 and 9231 DF, p-value: < 2.2e-16
```

The command `colnames(df)` is executed, resulting in the following output:

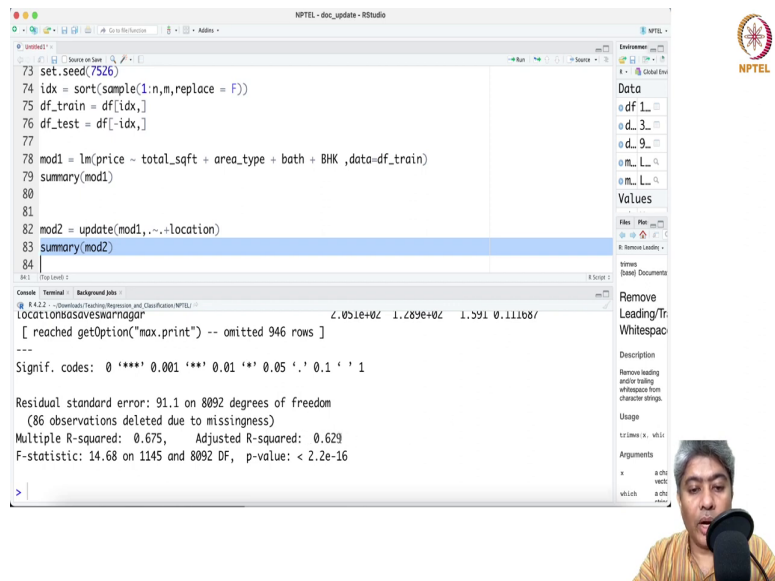
```
[1] "area_type" "availability" "location" "size" "society" "total_sqft"  
[7] "bath" "balcony" "price" "BHK"
```

The right-hand pane shows the "Environment" tab with a list of objects: `df_1`, `df_3`, `df_9`, `m_1`, and `d_Na`. The "Data" tab is selected, showing the structure of the data frame.



So, now, if we add location if we want to add location so, mod2 equal to what we will do update the model1 you update model1 how you do that dot comma dot plus you just put the location ok. So, what is the col names of df. So, this is the location.

(Refer Slide Time: 28:30)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
73 set.seed(7526)
74 idx = sort(sample(1:n,m,replace = F))
75 df_train = df[idx,]
76 df_test = df[-idx,]
77
78 mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
79 summary(mod1)
80
81
82 mod2 = update(mod1, ~. + location)
83 summary(mod2)
84
```

The console output shows the summary of the linear model 'mod2':

```
location: saseswarnagar 2.051e+04 1.289e+04 1.391 0.1118 /
[ reached getOption("max.print") -- omitted 946 rows ]
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 91.1 on 8092 degrees of freedom
(86 observations deleted due to missingness)
Multiple R-squared:  0.675,    Adjusted R-squared:  0.629
F-statistic: 14.68 on 1145 and 8092 DF,  p-value: < 2.2e-16
```

On the right side of the RStudio window, there is a sidebar with sections for "Data", "Values", "Files", "Plots", "Environments", "Packages", "Viewer", "Console", "Terminal", "Background Jobs", "Help", "Documentation", "Remove Leading/Trailing Whitespace", "Description", "Usage", and "Arguments". The "Data" section shows a list of data frames: "df_1", "df_3", "df_9", "m_1", "m_2", "m_3", "m_4", "m_5", "m_6", "m_7", "m_8", "m_9", "m_10", "m_11", "m_12", "m_13", "m_14", "m_15", "m_16", "m_17", "m_18", "m_19", "m_20", "m_21", "m_22", "m_23", "m_24", "m_25", "m_26", "m_27", "m_28", "m_29", "m_30", "m_31", "m_32", "m_33", "m_34", "m_35", "m_36", "m_37", "m_38", "m_39", "m_40", "m_41", "m_42", "m_43", "m_44", "m_45", "m_46", "m_47", "m_48", "m_49", "m_50", "m_51", "m_52", "m_53", "m_54", "m_55", "m_56", "m_57", "m_58", "m_59", "m_60", "m_61", "m_62", "m_63", "m_64", "m_65", "m_66", "m_67", "m_68", "m_69", "m_70", "m_71", "m_72", "m_73", "m_74", "m_75", "m_76", "m_77", "m_78", "m_79", "m_80", "m_81", "m_82", "m_83", "m_84", "m_85", "m_86", "m_87", "m_88", "m_89", "m_90", "m_91", "m_92", "m_93", "m_94", "m_95", "m_96", "m_97", "m_98", "m_99", "m_100". The "Values" section shows a list of values: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Files" section shows a list of files: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Plots" section shows a list of plots: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Environments" section shows a list of environments: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Packages" section shows a list of packages: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Viewer" section shows a list of viewers: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Console" section shows a list of console outputs: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Terminal" section shows a list of terminal outputs: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Background Jobs" section shows a list of background jobs: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Help" section shows a list of help topics: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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". The "Documentation" section shows a list of documentation topics: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "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".

So, you just put location and it should work alright and then let us see summary mod2 of location and you can see that adjusted r square now went up to 0.629 right ok.

(Refer Slide Time: 29:00)

```
NPTEL - doc_update - RStudio

Source | Terminal | Background jobs | Environment | Global Env |

R 4.1.2 | ... | Download/Training/Regression_and_Classification/NPTEL |

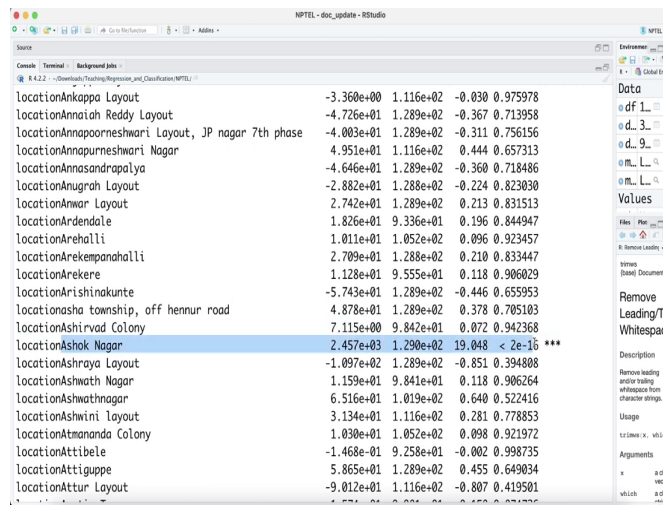
locationBannerghatta Road      2.210e+01  9.156e+01  0.241  0.809291
locationBapuji Layout          1.158e+02  1.290e+02  0.898  0.369216
locationBapuji Nagar          -8.589e+00  1.289e+02 -0.067  0.946876
locationBasapura               6.042e+00  9.663e+01  0.063  0.950143
locationBasava Nagar           1.611e+01  1.052e+02  0.153  0.878275
locationBasavanagara           9.783e-01  1.052e+02  0.009  0.992583
locationBasavanapura          -1.452e+01  1.019e+02 -0.143  0.886672
locationBasavangudi            1.007e+02  9.299e+01  1.083  0.278668
locationBasavanna Nagar        3.909e+00  1.116e+02  0.035  0.972056
locationBasaveshwara Nagar     6.086e+01  9.363e+01  0.650  0.515663
locationBasaveshwara Nagar Yelahanka 2.757e+01  9.982e+01  0.276  0.782391
locationBasaveshwarnagar       2.131e+01  1.289e+02  0.165  0.868651
locationBasaveshwarnagar       2.051e+02  1.289e+02  1.591  0.111687
[ reached getOption("max.print") -- omitted 946 rows ]
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 91.1 on 8092 degrees of freedom
(86 observations deleted due to missingness)
Multiple R-squared:  0.675,    Adjusted R-squared:  0.629
F-statistic: 14.68 on 1145 and 8092 DF,  p-value: < 2.2e-16

> |
```



(Refer Slide Time: 29:06)

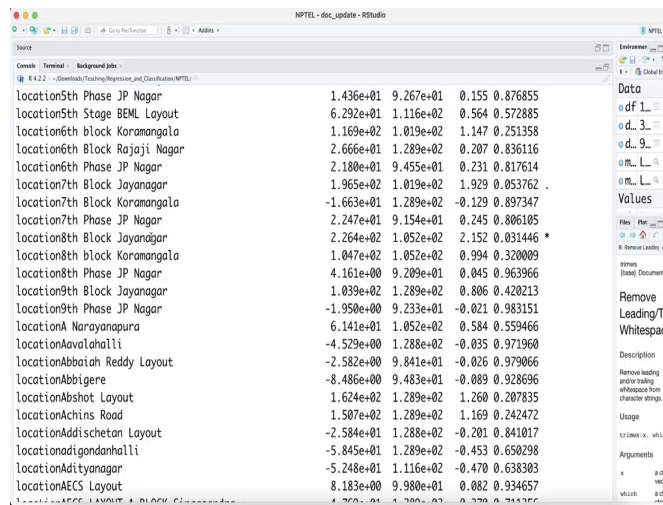


| | | | | |
|---|------------|-----------|--------|-------------|
| locationAnkappa Layout | -3.360e+00 | 1.116e+02 | -0.030 | 0.975978 |
| locationAnnaiah Reddy Layout | -4.726e+01 | 1.289e+02 | -0.367 | 0.713958 |
| locationAnnapoorneshwari Layout, JP nagar 7th phase | -4.003e+01 | 1.289e+02 | -0.311 | 0.756156 |
| locationAnnapurneshwari Nagar | 4.951e+01 | 1.116e+02 | 0.444 | 0.657313 |
| locationAnnasandrapalya | -4.646e+01 | 1.289e+02 | -0.360 | 0.718486 |
| locationAnugrah Layout | -2.882e+01 | 1.288e+02 | -0.224 | 0.823030 |
| locationAnwar Layout | 2.742e+01 | 1.289e+02 | 0.213 | 0.831513 |
| locationArdendale | 1.826e+01 | 9.336e+01 | 0.196 | 0.844947 |
| locationArehalli | 1.011e+01 | 1.052e+02 | 0.096 | 0.923457 |
| locationArekempahalli | 2.709e+01 | 1.288e+02 | 0.210 | 0.833447 |
| locationArekere | 1.128e+01 | 9.555e+01 | 0.118 | 0.906029 |
| locationArishinakunte | -5.743e+01 | 1.289e+02 | -0.446 | 0.655953 |
| locationasha township, off hennur road | 4.878e+01 | 1.289e+02 | 0.378 | 0.705103 |
| locationAshirvad Colony | 7.115e+00 | 9.842e+01 | 0.072 | 0.942368 |
| locationAshok Nagar | 2.457e+03 | 1.290e+02 | 19.048 | < 2e-16 *** |
| locationAshraya Layout | -1.097e+02 | 1.289e+02 | -0.851 | 0.394808 |
| locationAshwath Nagar | 1.159e+01 | 9.841e+01 | 0.118 | 0.906264 |
| locationAshwathnagar | 6.516e+01 | 1.019e+02 | 0.640 | 0.522416 |
| locationAshwini Layout | 3.134e+01 | 1.116e+02 | 0.281 | 0.778853 |
| locationAtmananda Colony | 1.030e+01 | 1.052e+02 | 0.098 | 0.921972 |
| locationAttibele | -1.468e-01 | 9.258e+01 | -0.002 | 0.998735 |
| locationAttiguppe | 5.865e+01 | 1.289e+02 | 0.455 | 0.649034 |
| locationAttur Layout | -9.012e+01 | 1.116e+02 | -0.807 | 0.419501 |



See Ashok Nagar has very strong positive premium looks like Ashok Nagar ok.

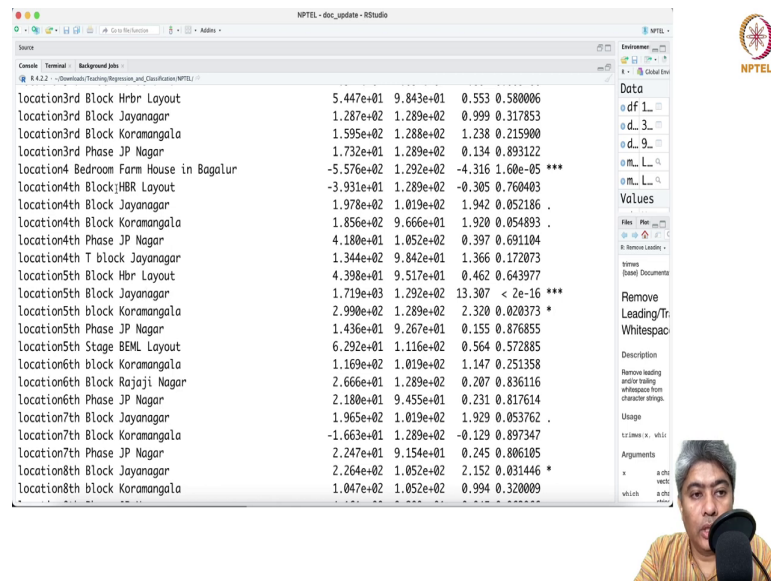
(Refer Slide Time: 29:30)



| | 1.436e+01 | 9.267e+01 | 0.155 | 0.876855 |
|-------------------------------|------------|-----------|--------|----------|
| location5th Phase JP Nagar | 6.292e+01 | 1.116e+02 | 0.564 | 0.572885 |
| location5th Stage BEML Layout | 1.169e+02 | 1.019e+02 | 1.147 | 0.251358 |
| location6th block Koramangala | 2.666e+01 | 1.289e+02 | 0.207 | 0.836116 |
| location6th Phase JP Nagar | 2.180e+01 | 9.455e+01 | 0.231 | 0.817614 |
| location7th Block Jayanagar | 1.965e+02 | 1.019e+02 | 1.929 | 0.053762 |
| location7th Block Koramangala | -1.663e+01 | 1.289e+02 | -0.129 | 0.897347 |
| location7th Phase JP Nagar | 2.247e+01 | 9.154e+01 | 0.245 | 0.806105 |
| location8th Block Jayanagar | 2.264e+02 | 1.052e+02 | 2.152 | 0.031446 |
| location8th block Koramangala | 1.047e+02 | 1.052e+02 | 0.994 | 0.320009 |
| location8th Phase JP Nagar | 4.161e+00 | 9.209e+01 | 0.045 | 0.963966 |
| location9th Block Jayanagar | 1.039e+02 | 1.289e+02 | 0.806 | 0.420213 |
| location9th Phase JP Nagar | -1.950e+00 | 9.233e+01 | -0.021 | 0.983151 |
| locationA Narayanapura | 6.141e+01 | 1.052e+02 | 0.584 | 0.559466 |
| locationAvalahalli | -4.529e+00 | 1.288e+02 | -0.035 | 0.971960 |
| locationAbbaiah Reddy Layout | -2.582e+00 | 9.841e+01 | -0.026 | 0.979066 |
| locationAbbigere | -8.486e+00 | 9.483e+01 | -0.089 | 0.928696 |
| locationAbshot Layout | 1.624e+02 | 1.289e+02 | 1.260 | 0.207835 |
| locationAchins Road | 1.507e+02 | 1.289e+02 | 1.169 | 0.242472 |
| locationAddisichetan Layout | -2.584e+01 | 1.288e+02 | -0.201 | 0.841017 |
| locationadigondanahalli | -5.845e+01 | 1.289e+02 | -0.453 | 0.650298 |
| locationAdityanagar | -5.248e+01 | 1.116e+02 | -0.470 | 0.638303 |
| locationAECS Layout | 8.183e+00 | 9.900e+01 | 0.082 | 0.934657 |



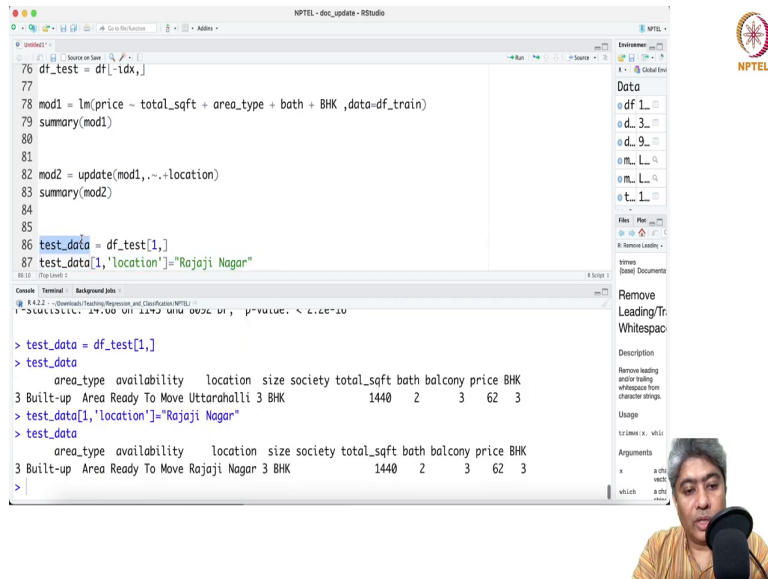
(Refer Slide Time: 29:35)



8th Block Jayanagar 5th Block Jayanagar 4 Bedroom house in Bengaluru ok then these are the in some of the location still required you know I think still the bedroom things are 7th block, 6th block this 4 Bedroom Farm House in Bagalur ok they have put the bedroom information in the location which is not right second stage are in the ok.

So, alright I think there are lot of things are being missed actually lot of areas are being missed, but adjusted R squared as now 0.629, which is similar to that of similar to that of Python.

(Refer Slide Time: 30:47)



```
76 df_test = df[-idx,]
77
78 mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
79 summary(mod1)
80
81
82 mod2 = update(mod1, ~. + location)
83 summary(mod2)
84
85
86 test_data = df_test[1,]
87 test_data[1, 'location'] = "Rajaji Nagar"
```

```
> test_data = df_test[1,]
> test_data
  area_type availability location size society total_sqft bath balcony price BHK
3 Built-up Area Ready To Move Uttarahalli 3 BHK      1440    2      3     62    3

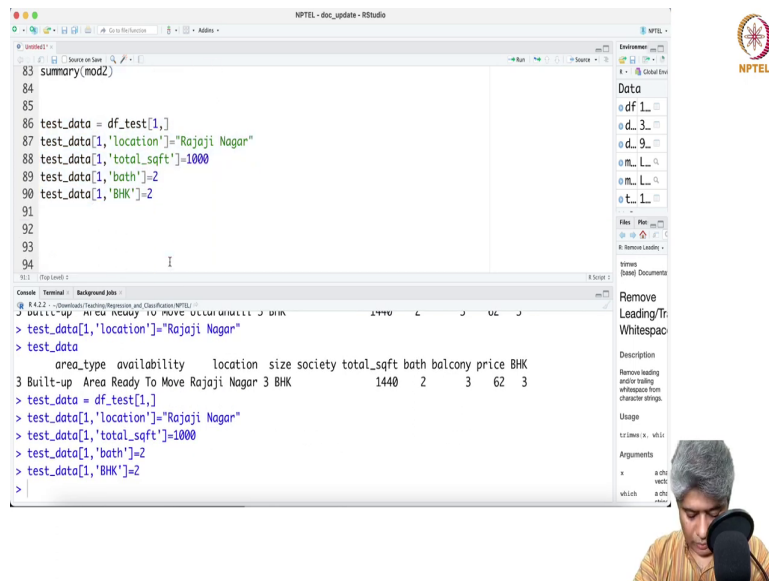
> test_data[1, 'location'] = "Rajaji Nagar"
> test_data
  area_type availability location size society total_sqft bath balcony price BHK
3 Built-up Area Ready To Move Rajaji Nagar 3 BHK      1440    2      3     62    3

>
```

Now, what I am going to do is very simple that I am going to like check some price like test data ok test data I am going to create some df test 1. So, this is df test data now this test data is a test point that we are going to try out the location we are going to use say first Rajaji Nagar.

I remember in Rajaji Nagar case we got a value of near about 200 lakhs or 2.4 crores something like that let us see in this time what happens what is the situation. So, now, we have changed it was Uttarahalli, but now in the test data we have given a location Rajaji Nagar.

(Refer Slide Time: 31:53)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The script editor contains the following R code:

```
83 summary(mod2)
84
85
86 test_data = df_test[1,]
87 test_data[1,'location']="Rajaji Nagar"
88 test_data[1,'total_sqft']=1000
89 test_data[1,'bath']=2
90 test_data[1,'BHK']=2
91
92
93
94
```

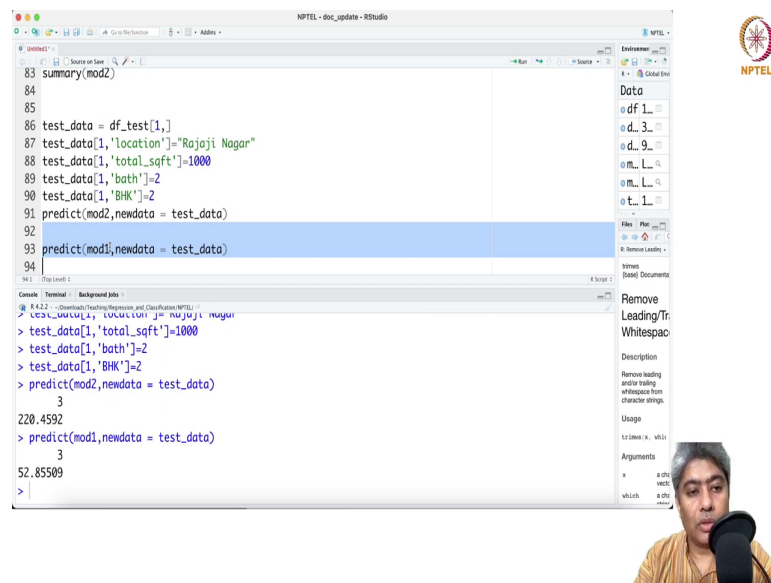
The console shows the output of the code:

```
> built_up = read_csv("move_location.csv")
> test_data
  area_type availability location size society total_sqft bath balcony price BHK
3 Built-up Area Ready To Move Rajaji Nagar 3 BHK      1440    2      3     62    3
> test_data = df_test[1,]
> test_data[1,'location']="Rajaji Nagar"
> test_data[1,'total_sqft']=1000
> test_data[1,'bath']=2
> test_data[1,'BHK']=2
>
```

On the right side, the 'Data' pane shows a table with columns: df_1, d_3, d_9, m_1, m_2, t_1. Below it, the 'Remove Leading/Trailing Whitespace' pane shows a description of the function and its arguments.

And the total square feet will be total. So, total square feet will be say 1000 square feet ok.
And then bathroom will be there will be two bathroom and it will be 2 BHK it will be 2 BHK.

(Refer Slide Time: 32:33)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The script editor contains the following R code:

```
83 summary(mod2)
84
85
86 test_data = df_test[1,]
87 test_data[1, 'location'] = "Rajaji Nagar"
88 test_data[1, 'total_sqft'] = 1000
89 test_data[1, 'bath'] = 2
90 test_data[1, 'BHK'] = 2
91 predict(mod2, newdata = test_data)
92
93 predict(mod1, newdata = test_data)
94
```

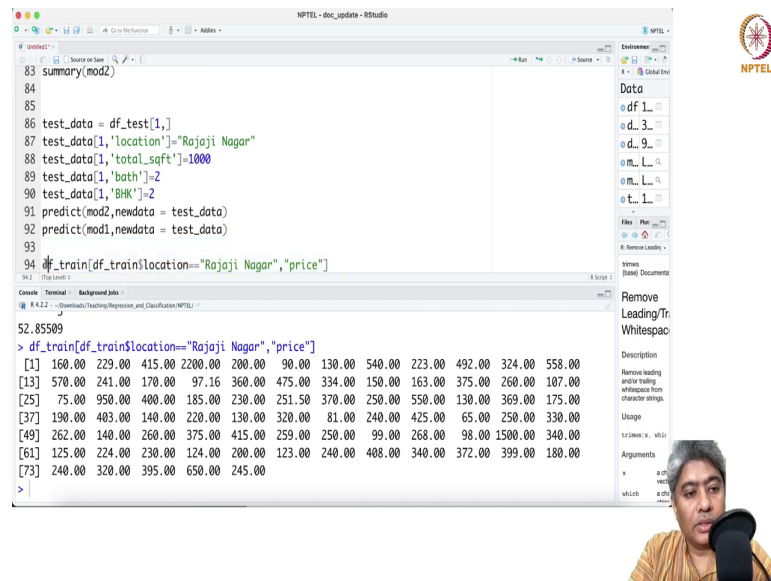
The console output shows the following results:

```
> test_data[1, 'total_sqft'] = 1000
> test_data[1, 'bath'] = 2
> test_data[1, 'BHK'] = 2
> predict(mod2, newdata = test_data)
3
220.4592
> predict(mod1, newdata = test_data)
3
52.85509
>
```

On the right side of the RStudio window, there is a sidebar with the 'Data' pane showing a data frame with columns 'df_1', 'd_3', 'd_9', 'm_1', 'm_2', and 't_1'. Below the 'Data' pane is the 'Environment' pane showing the 'test_data' object. At the bottom right, there is a small video feed of a person speaking into a microphone.

So, let us run this through and now if I just predict; predict mod2 with newdata equal to test data 220.45. So, it gives me a price of 220 lakh rupees so; that means, 2.2 crore in Python it was giving me 2.4 crore. So, it is somewhat in the same range remember that we also had made a model 1 initial model base model what happens if we use model 1. And model1 had a very low R square 45 percent in model1 it is giving me 52 lakh. So, model1 is looks like giving me a not so good also model.

(Refer Slide Time: 33:30)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The script editor contains the following R code:

```
83 summary(mod2)
84
85
86 test_data = df_test[1,]
87 test_data[1,'location']="Rajaji Nagar"
88 test_data[1,'total_sqft']=1000
89 test_data[1,'bath']=2
90 test_data[1,'bhk']=2
91 predict(mod2,newdata = test_data)
92 predict(mod1,newdata = test_data)
93
94 df_train[df_train$location=="Rajaji Nagar","price"]
```

The console output shows the result of the last command, displaying a data frame of house prices for Rajaji Nagar:

```
52.85509
> df_train[df_train$location=="Rajaji Nagar","price"]
[1] 160.00 229.00 415.00 2200.00 200.00 90.00 130.00 540.00 223.00 492.00 324.00 558.00
[13] 570.00 241.00 170.00 97.16 360.00 475.00 334.00 150.00 163.00 375.00 260.00 107.00
[25] 75.00 950.00 400.00 185.00 230.00 251.50 370.00 250.00 550.00 130.00 369.00 175.00
[37] 190.00 403.00 140.00 220.00 130.00 320.00 81.00 240.00 425.00 65.00 250.00 330.00
[49] 262.00 140.00 260.00 375.00 415.00 259.00 250.00 99.00 268.00 98.00 1500.00 340.00
[61] 125.00 224.00 230.00 124.00 200.00 123.00 240.00 408.00 340.00 372.00 399.00 180.00
[73] 240.00 320.00 395.00 650.00 245.00
```

The environment pane on the right shows the following objects:

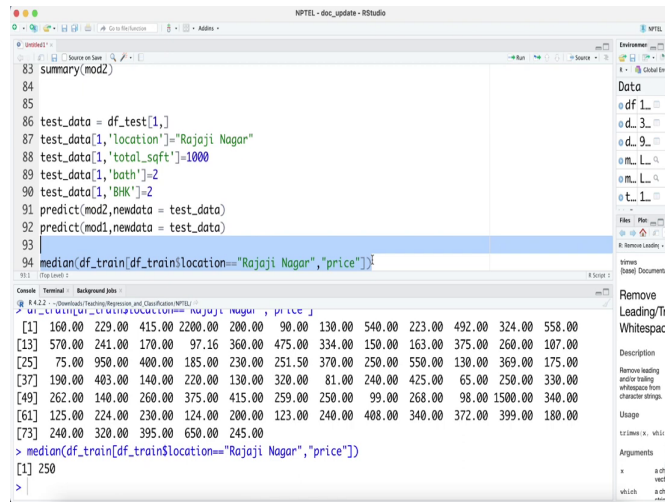
- df_1
- df_3
- df_9
- m_1
- m_2
- t_1

The console pane shows the following output:

```
52.85509
> df_train[df_train$location=="Rajaji Nagar","price"]
[1] 160.00 229.00 415.00 2200.00 200.00 90.00 130.00 540.00 223.00 492.00 324.00 558.00
[13] 570.00 241.00 170.00 97.16 360.00 475.00 334.00 150.00 163.00 375.00 260.00 107.00
[25] 75.00 950.00 400.00 185.00 230.00 251.50 370.00 250.00 550.00 130.00 369.00 175.00
[37] 190.00 403.00 140.00 220.00 130.00 320.00 81.00 240.00 425.00 65.00 250.00 330.00
[49] 262.00 140.00 260.00 375.00 415.00 259.00 250.00 99.00 268.00 98.00 1500.00 340.00
[61] 125.00 224.00 230.00 124.00 200.00 123.00 240.00 408.00 340.00 372.00 399.00 180.00
[73] 240.00 320.00 395.00 650.00 245.00
```

Accuracy for model1 is adjusted R square is very low compared to may at least 20 percent lower than model2. So, location is very important location if you based on the location the house price varied extremely ok. Now one thing we can do is we can look into the train dataset ok in the train dataset among the df train location equal to Rajaji Nagar and we can look into the price. So, these are the instances which is the houses which belongs to the Rajaji Nagar and their prices.

(Refer Slide Time: 34:28)





The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
83 summary(mod2)
84
85
86 test_data = df_test[1,]
87 test_data[1,'location']="Rajaji Nagar"
88 test_data[1,'total_sqft']=1000
89 test_data[1,'bath']=2
90 test_data[1,'BHK']=2
91 predict(mod2,newdata = test_data)
92 predict(mod1,newdata = test_data)
93
94 median(df_train[df_train$location=="Rajaji Nagar","price"])
```

The console output shows the result of the median calculation:

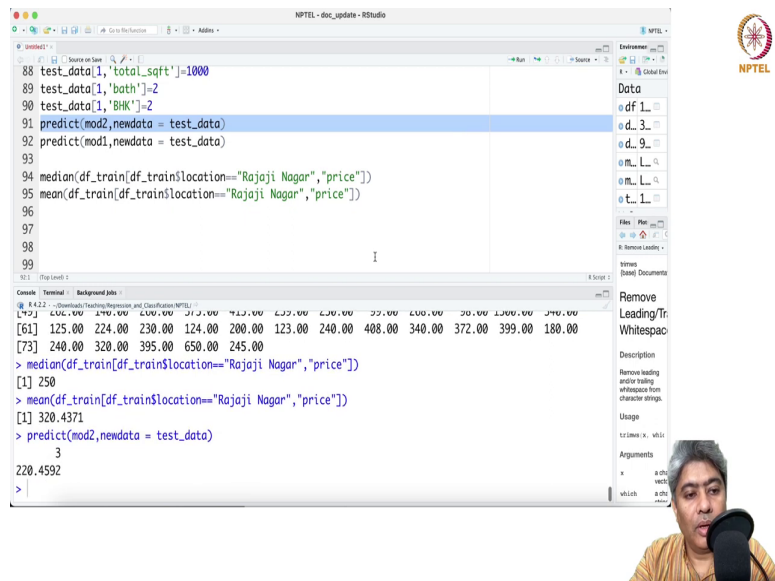
```
> median(df_train[df_train$location=="Rajaji Nagar","price"])
[1] 250
```

The environment pane on the right shows a data frame with columns: df_1, d_3, d_9, m_1, m_2, t_1. The file pane shows a file named "R Remove Leading Whitespace".



Now, if I take the mean price or median price. So, the median price is near about 250.

(Refer Slide Time: 34:47)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The editor pane contains the following R code:

```
88 test_data[1, 'total_sqft'] = 1000
89 test_data[1, 'bath'] = 2
90 test_data[1, 'BHK'] = 2
91 predict(mod2, newdata = test_data)
92 predict(mod1, newdata = test_data)
93
94 median(df_train[df_train$location == "Rajaji Nagar", "price"])
95 mean(df_train[df_train$location == "Rajaji Nagar", "price"])
96
97
98
99
```

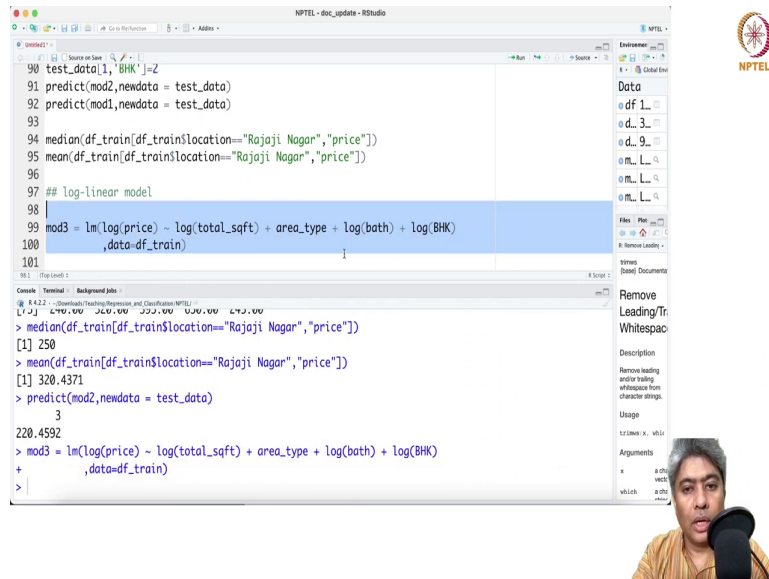
The console pane shows the output of the code:

```
[61] 125.00 224.00 230.00 124.00 200.00 123.00 240.00 408.00 340.00 372.00 399.00 180.00
[73] 240.00 320.00 395.00 650.00 245.00
> median(df_train[df_train$location == "Rajaji Nagar", "price"])
[1] 250
> mean(df_train[df_train$location == "Rajaji Nagar", "price"])
[1] 320.4371
> predict(mod2, newdata = test_data)
3
220.4592
>
```

On the right side of the RStudio window, there is a sidebar with a 'Data' pane showing a list of data frames (df_1, df_3, df_9, etc.) and a 'Files' pane. A video inset in the bottom right corner shows a man speaking into a microphone.

And mean price average or average price is 320. So, given this I think predict model predicted model is giving 220 I think is a decently the correct value probably. And we can try few other models for example, we have not tried log linear model.

(Refer Slide Time: 35:16)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The editor pane contains the following R code:

```
90 test_data[, "BHK"] = 2
91 predict(mod2, newdata = test_data)
92 predict(mod1, newdata = test_data)
93
94 median(df_train[df_train$location == "Rajaji Nagar", "price"])
95 mean(df_train[df_train$location == "Rajaji Nagar", "price"])
96
97 ## log-linear model
98
99 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
100           , data = df_train)
101
```

The console pane shows the output of the code execution:

```
> median(df_train[df_train$location == "Rajaji Nagar", "price"])
[1] 250
> mean(df_train[df_train$location == "Rajaji Nagar", "price"])
[1] 320.4371
> predict(mod2, newdata = test_data)
3
220.4592
> mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
+           , data = df_train)
>
```

On the right side of the RStudio window, there is a sidebar with a "Data" pane showing variables like "df_1", "d_3", "d_9", "m_1", and "m_2". Below it is a "Remove Leading/Trailing Whitespace" button. At the bottom right, there is a small video feed of a person speaking into a microphone.

So, we can try log linear model. So, let us try that. So, what we can do? We can just take the mod1 and this will be the third model and we are trying log of price right log of price and log of total square feet. So, this will be log of bathroom I am giving 1 because in case there is 0 bathroom it should not happen [FL], let me not take if it there is 0 then it will throw error, but let us take this for just for fun let us see how this turns ok. I have to put location also.

(Refer Slide Time: 36:13)

The screenshot displays the RStudio interface with a script editor on the left and a console on the right. The script file is named "NPTEL - doc_update - RStudio".

```
# NPTEL - doc_update - RStudio

91 predict(mod2,newdata = test_data)
92 predict(mod1,newdata = test_data)
93
94 median(df_train[df_train$location=="Rajaji Nagar","price"])
95 mean(df_train[df_train$location=="Rajaji Nagar","price"])
96
97 ## log-linear model
98
99 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
100           ,location
101           ,data=df_train)
102
103 # log-linear
```

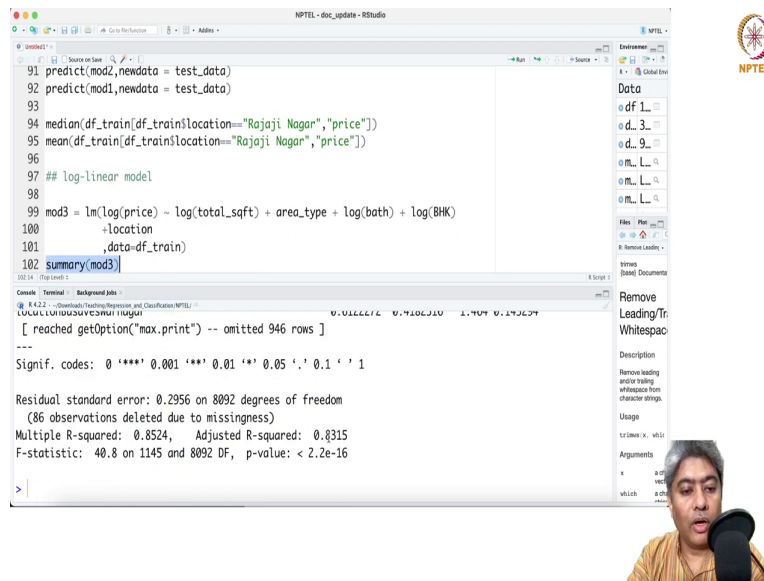
The console output shows the results of the predictions:

```
[1] 320.4371
> predict(mod2,newdata = test_data)
      3
220.4592
> mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
+         ,data=df_train)
> mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
+         ,location
+         ,data=df_train)
```

In the bottom right corner, there is a small video inset showing a man speaking into a microphone.

Plus location ok, so it will take little time because it will create all the dummy variables and then it will fit the model, alright.

(Refer Slide Time: 36:31)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
91 predict(mod2,newdata = test_data)
92 predict(mod1,newdata = test_data)
93
94 median(df_train[df_train$location=="Rajaji Nagar","price"])
95 mean(df_train[df_train$location=="Rajaji Nagar","price"])
96
97 ## log-linear model
98
99 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
100           +location
101           ,data=df_train)
102 summary(mod3)
```

The console output shows the summary of the linear model 'mod3':

```
U.UUUUUUU  U.UUUUUUU  U.UUUUUUU  U.UUUUUUU  U.UUUUUUU
[ reached getOption("max.print") -- omitted 946 rows ]
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2956 on 8092 degrees of freedom
(86 observations deleted due to missingness)
Multiple R-squared:  0.8524,    Adjusted R-squared:  0.8315
F-statistic: 40.8 on 1145 and 8092 DF,  p-value: < 2.2e-16
```

On the right side of the RStudio window, there is a sidebar with a "Data" pane showing variables like "df_1", "d_3", "d_9", "m_1", "m_2", "m_3", and "m_4". Below it is a "Files" pane showing "R Remove Leading Whitespace". At the bottom right, there is a small video feed of a man speaking into a microphone.

Summary mod3 so, now, after log written it has gone up to 85 percent ok so, this is a huge jump another 20 percent.

(Refer Slide Time: 36:50)

```
[1] 160.00 229.00 415.00 2200.00 200.00 90.00 130.00 540.00 223.00 492.00 324.00 558.00
[13] 570.00 241.00 170.00 97.16 360.00 475.00 334.00 150.00 163.00 375.00 260.00 107.00
[25] 75.00 950.00 400.00 185.00 230.00 251.50 370.00 250.00 550.00 130.00 369.00 175.00
[37] 190.00 403.00 140.00 220.00 130.00 320.00 81.00 240.00 425.00 65.00 250.00 330.00
[49] 262.00 140.00 260.00 375.00 415.00 259.00 250.00 99.00 268.00 98.00 1500.00 340.00
[61] 125.00 224.00 230.00 124.00 200.00 123.00 240.00 408.00 340.00 372.00 399.00 180.00
[73] 240.00 320.00 395.00 650.00 245.00

> median(df_train[df_train$location=="Rajaji Nagar", "price"])
[1] 250

> mean(df_train[df_train$location=="Rajaji Nagar", "price"])
[1] 320.4371

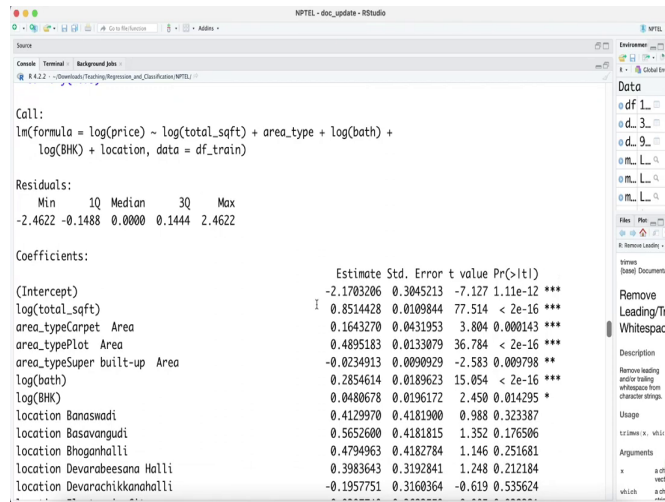
> predict(mod2, newdata = test_data)
3
220.4592

> mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
+ , data=df_train)
> mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
+ +location
+ , data=df_train)
> summary(mod3)

Call:
lm(formula = log(price) ~ log(total_sqft) + area_type + log(bath) +
    log(BHK) + location, data = df_train)
```

Just because I am getting, but that may not be a correct thing because remember that.

(Refer Slide Time: 36:52)



The image shows a screenshot of an RStudio session. The console displays the following output:

```
Call:
lm(formula = log(price) ~ log(total_sqft) + area_type + log(bath) +
    log(BHK) + location, data = df_train)
```

Residuals:

| Min | 1Q | Median | 3Q | Max |
|---------|---------|--------|--------|--------|
| -2.4622 | -0.1488 | 0.0000 | 0.1444 | 2.4622 |

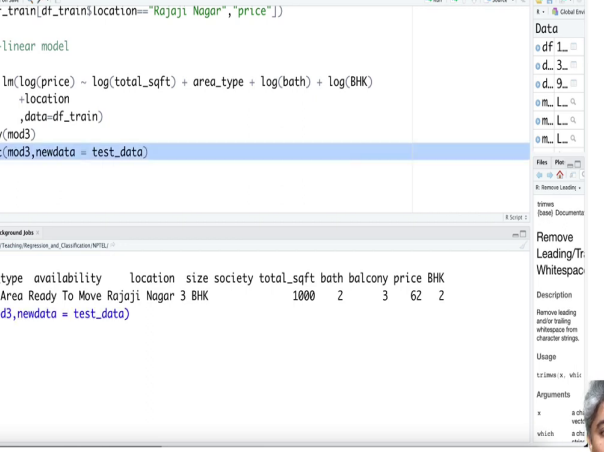
Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) |
|------------------------------|------------|------------|---------|--------------|
| (Intercept) | -2.1703206 | 0.3045213 | -7.127 | 1.11e-12 *** |
| log(total_sqft) | 0.8514428 | 0.0109844 | 77.514 | < 2e-16 *** |
| area_typeCarpet Area | 0.1643270 | 0.0431953 | 3.804 | 0.000143 *** |
| area_typePlot Area | 0.4895183 | 0.0133079 | 36.784 | < 2e-16 *** |
| area_typeSuper built-up Area | -0.0234913 | 0.0090929 | -2.583 | 0.009798 ** |
| log(bath) | 0.2854614 | 0.0189623 | 15.054 | < 2e-16 *** |
| log(BHK) | 0.0480678 | 0.0196172 | 2.450 | 0.014295 * |
| location Banaswadi | 0.4129970 | 0.4181900 | 0.988 | 0.323387 |
| location Basavangudi | 0.5652600 | 0.4181815 | 1.352 | 0.176506 |
| location Bhoganhalli | 0.4794963 | 0.4182784 | 1.146 | 0.251681 |
| location Devarabeesana Halli | 0.3983643 | 0.3192841 | 1.248 | 0.212184 |
| location Devarachikkanahalli | -0.1957751 | 0.3160364 | -0.619 | 0.535624 |

The RStudio interface also shows the 'Data' pane on the right with 'df_1' selected. A small video inset of a person is visible in the bottom right corner.

These are in the log scale log scale it is linear it is becoming linear and that is why and in the log bath and log BHK they are all now positive and everything is now pretty cool actually. So, maybe it will give us a much better how accuracy. So, what I will do? I will just take this thing.

(Refer Slide Time: 37:24)



The screenshot shows an RStudio window with a script editor on the left and an environment pane on the right. The script editor contains R code for fitting a linear model and predicting values. The environment pane shows the data frame 'Data' with 10 rows and 10 columns. The code in the script editor is as follows:

```

95 mean(df_train[df_train$location=="Kajaji Nagar","price"])
96
97 ## log-linear model
98
99 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
100           ~ location
101           ,data=df_train)
102 summary(mod3)
103 predict(mod3,newdata = test_data)
104
105
106

```

The environment pane shows the following data frame:

| area_type | availability | location | size | society | total_sqft | bath | balcony | price | BHK |
|------------|--------------|----------|--------------|---------|------------|------|---------|-------|-----|
| 3 Built-up | Area Ready | To Move | Kajaji Nagar | 3 BHK | 1000 | 2 | 3 | 62 | 2 |

The output of the script is shown in the console:

```

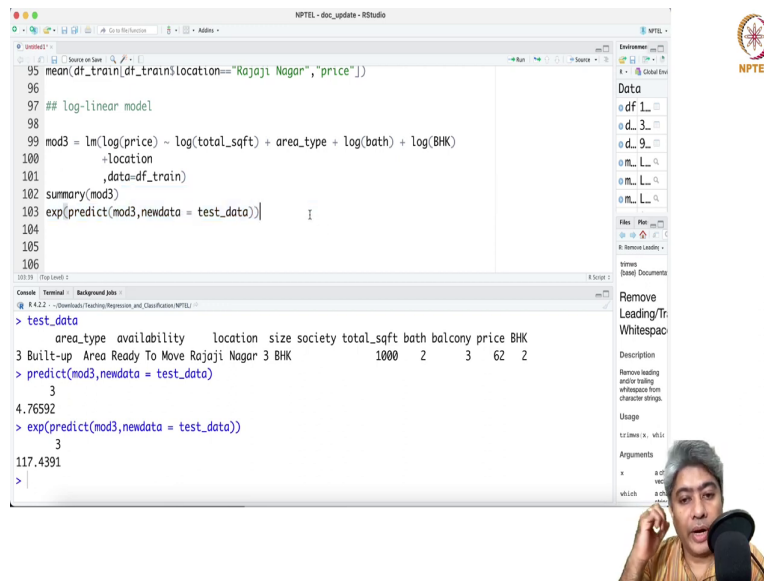
> test_data
      area_type availability      location size society total_sqft bath balcony price BHK
3 Built-up Area Ready To Move Kajaji Nagar 3 BHK      1000      2      3      62      2
> predict(mod3,newdata = test_data)
      3
4.76592
>

```

The environment pane also shows a list of variables: Data, df, log, lm, predict, summary, test_data, and test_data\$price. The 'Data' variable is highlighted, and its structure is shown as a data frame with 10 rows and 10 columns.

And we can just use this model 3 to predict the same data set. So, test data is still that same Rajaji Nagar with 100 square feet and 2 bathroom and 2 BHK, but now it will give me log scale the price will be log scale.

(Refer Slide Time: 37:41)



The image shows an RStudio window with the following code in the script editor:

```
95 mean(df_train[df_train$location=="Rajaji Nagar", "price"])
96
97 ## log-linear model
98
99 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
100           + location
101           , data=df_train)
102 summary(mod3)
103 exp(predict(mod3, newdata = test_data))
104
105
106
```

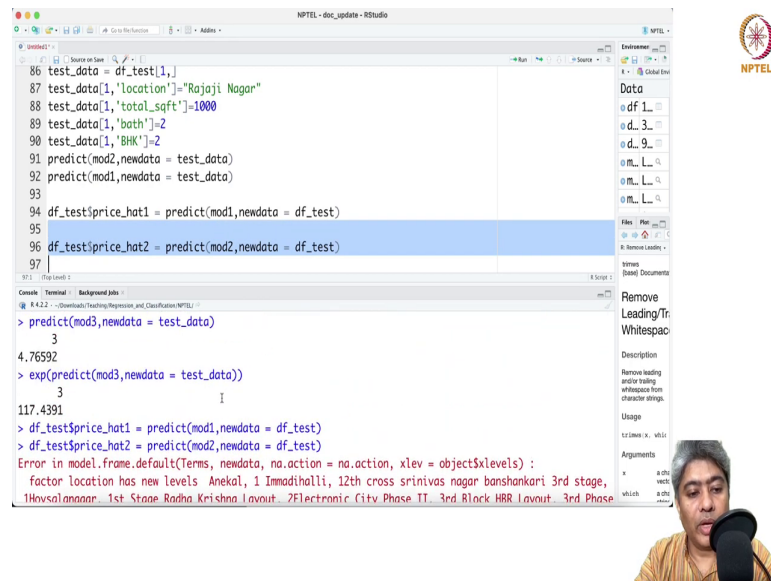
The console output shows the following results:

```
> test_data
  area_type availability location size society total_sqft bath balcony price BHK
3 Built-up Area Ready To Move Rajaji Nagar 3 BHK      1000    2      3    62    2
> predict(mod3, newdata = test_data)
3
4.76592
> exp(predict(mod3, newdata = test_data))
3
117.4391
>
```

The right sidebar shows the 'Data' pane with a list of objects: df_1, d_3, d_9, m_1, m_2, m_3. The 'Environment' pane shows the 'Remove Leading/Trailing Whitespace' option. The 'Description' pane shows the 'Remove leading and trailing whitespace from character strings' option. The 'Usage' pane shows the 'trimws' function. The 'Arguments' pane shows the 'x' argument as a character vector.

So, I have to put it in e to the power remember that I have to put it in e to the power. So, it is now giving me 107s ok. So, I do not know how good or how bad it is. So, this is looks like this could model looks good, but it is looks slightly slightly giving me under value now is it a right model. So, best way to check it out what will be the test cases in the test data frame what would be the predicted price.

(Refer Slide Time: 38:12)



The image shows an RStudio interface with a script editor, console, and environment pane. The script editor contains the following code:

```
86 test_data = df_test[1,]
87 test_data[1,'location']="Rajaji Nagar"
88 test_data[1,'total_sqft']=1000
89 test_data[1,'bath']=2
90 test_data[1,'BHK']=2
91 predict(mod2,newdata = test_data)
92 predict(mod1,newdata = test_data)
93
94 df_test$price_hat1 = predict(mod1,newdata = df_test)
95
96 df_test$price_hat2 = predict(mod2,newdata = df_test)
97
```

The console shows the output of the commands:

```
> predict(mod3,newdata = test_data)
3
4.76592
> exp(predict(mod3,newdata = test_data))
3
117.4391
> df_test$price_hat1 = predict(mod1,newdata = df_test)
> df_test$price_hat2 = predict(mod2,newdata = df_test)
Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = object$xlevels) :
  factor location has new levels Anekal, 1 Imadiahalli, 12th cross srinivas nagar banshankari 3rd stage,
  1Hovsalanagar, 1st Stage Radha Krishna Layout, 2Electronic City Phase II, 3rd Block HBR Layout, 3rd Phase
```

The environment pane on the right shows the following objects:

- df_1
- df_3
- df_9
- m_1
- m_2
- m_3

A video inset in the bottom right corner shows a man speaking into a microphone.

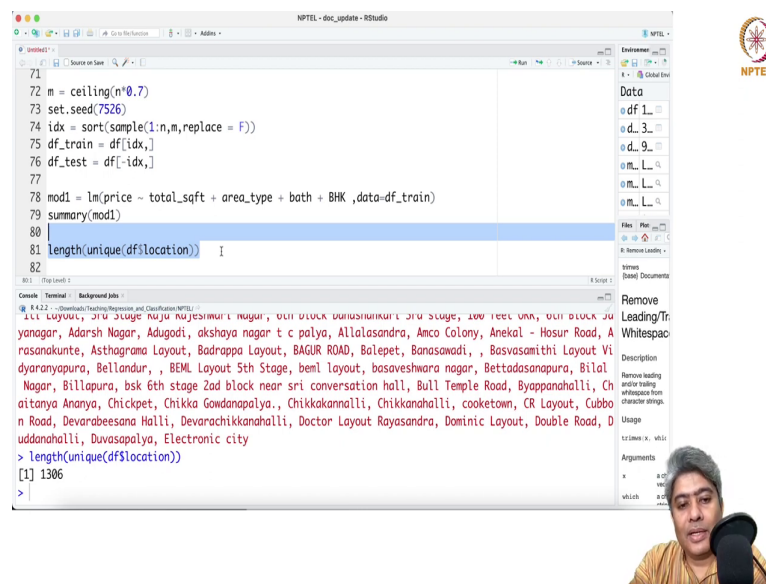
So, test df test dollar price hat what I will do? I will just put the test hat1 ok predict 1 df test here instead of that I will just put df test and then df test 2 that will be model 2 ok.



So, when the split happened the split was not correctly I mean has not considered every everybody in from the thing. So, one way of doing it is from each test area each area we can split it in train and test. So, that is a one way of definitely one way of doing it that will be a yeah then that is a good idea actually let me try that yeah.

So, this splitting what is happening let me repeat the problem ok please pause your video you may take a break come back, but let me give you tell you this problem I face this problem of 10 time, but let me tell you this problem. In the location there are too many locations.

(Refer Slide Time: 40:42)





The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The script editor contains the following R code:

```
71  
72 m = ceiling(n*0.7)  
73 set.seed(7526)  
74 idx = sort(sample(1:n,m,replace = F))  
75 df_train = df[idx,]  
76 df_test = df[-idx,]  
77  
78 mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)  
79 summary(mod1)  
80  
81 length(unique(df$location))  
82
```

The console output shows a list of location names and the result of the command:

```
> length(unique(df$location))  
[1] 1306
```

On the right side of the RStudio window, there is a sidebar with 'Data' and 'Environment' tabs. The 'Data' tab shows 'df_1' and 'df_2'. The 'Environment' tab shows 'df_1' and 'df_2'.



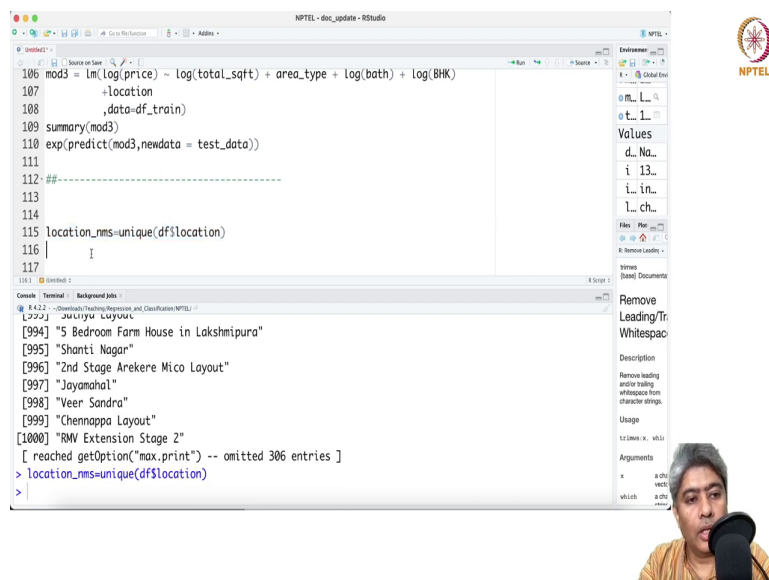
In fact, we can look into the location there are df dollar location and if I just say unique length ok length there are 1306 unique location. Now, when I am splitting the data I am randomly splitting and quite a few of them few locations all samples of locations have gone into the test so; that means, those locations are completely missing in the training dataset.

So, for those location the training dataset did not able to train and as a result now it is throwing that in the test dataset there are because if you look into what it is saying it is saying

that model data frame default error in model factors location has new level the test dataset has new levels, which is not available in the training datasets.

So, that is what it is talking about. So, we have to split the data in so, way that all 1306 location will split into 7030 way then each location will have a representative of 7 representative of samples in their training dataset ok you understand. So, let us try to make this thing it is going to be a bit of a let me try this let us try this ok.

(Refer Slide Time: 42:30)



The image shows an RStudio interface with a script editor, a console, and a video inset of a person speaking. The script editor contains the following R code:

```
106 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
107 +location
108 ,data=df_train)
109 summary(mod3)
110 exp(predict(mod3,newdata = test_data))
111
112 ##-----
113
114 location_nms=unique(df$location)
115 |
116 |
117 |
```

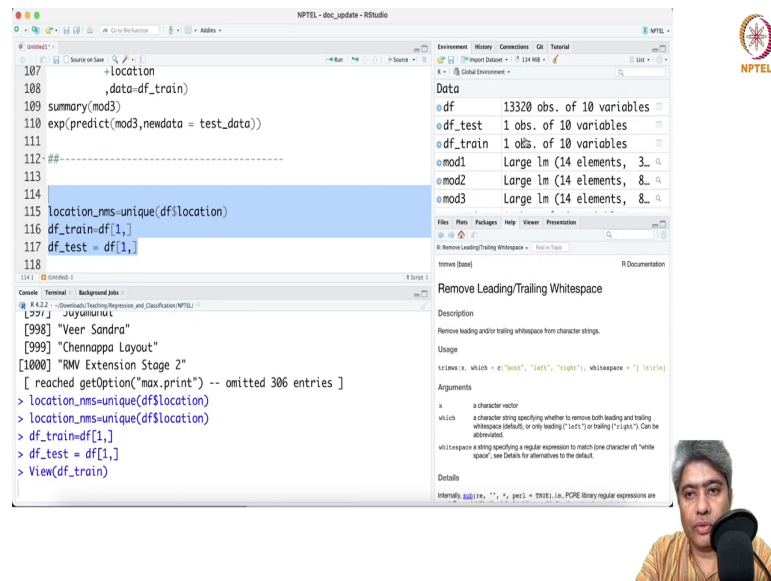
The console output shows the following messages:

```
[994] "5 Bedroom Farm House in Lakshmipura"
[995] "Shanti Nagar"
[996] "2nd Stage Arekere Mico Layout"
[997] "Jayamahal"
[998] "Veer Sandra"
[999] "Chennappa Layout"
[1000] "RMV Extension Stage 2"
[ reached getOption("max.print") -- omitted 306 entries ]
> location_nms=unique(df$location)
>
```

The video inset shows a person with a microphone, likely the speaker, in the bottom right corner.

Now, so, first is location names equal to what I will do? I will just take a yeah this guy and put it there. So, here I have all the names now I will just put it in there ok.

(Refer Slide Time: 43:17)



The screenshot displays an RStudio interface with the following components:

- Source Editor:** Contains R code for data manipulation and model training. Lines 115-117 are highlighted in blue.
- Environment:** Lists objects in the workspace: `df` (13320 obs. of 10 variables), `df_test` (1 obs. of 10 variables), `df_train` (1 obs. of 10 variables), `mod1` (Large lm (14 elements, 3...)), `mod2` (Large lm (14 elements, 8...)), and `mod3` (Large lm (14 elements, 8...)).
- Console:** Shows the output of the code, including the removal of leading and trailing whitespace from the `location` variable.
- Documentation:** Displays the documentation for the `removeLeadingTrailingWhitespace` function.

```
107 +location
108 ,data=df_train)
109 summary(mod3)
110 exp(predict(mod3,newdata = test_data))
111
112 ##-----
113
114
115 location_rms=unique(df$location)
116 df_train=df[,]
117 df_test = df[,]
118
```

Console Output:

```
[998] "Veer Sandra"
[999] "Chennappa Layout"
[1000] "RMV Extension Stage 2"
[ reached getOption("max.print") -- omitted 306 entries ]
> location_rms=unique(df$location)
> location_rms=unique(df$location)
> df_train=df[,]
> df_test = df[,]
> View(df_train)
```

Documentation: Remove Leading/Trailing Whitespace

Description: Remove leading and/or trailing whitespace from character strings.

Usage: `removeLeadingTrailingWhitespace(x, which = c("both", "left", "right"), whitespace = "\\s+")`

Arguments: `x` is a character vector; `which` is a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.

whitespace is a string specifying a regular expression to match (one character of "white space"; see Details for alternatives to the default).

Details: Internally `gsub(" ", "", x, perl = TRUE)` (i.e. PCRE library regular expressions are used).

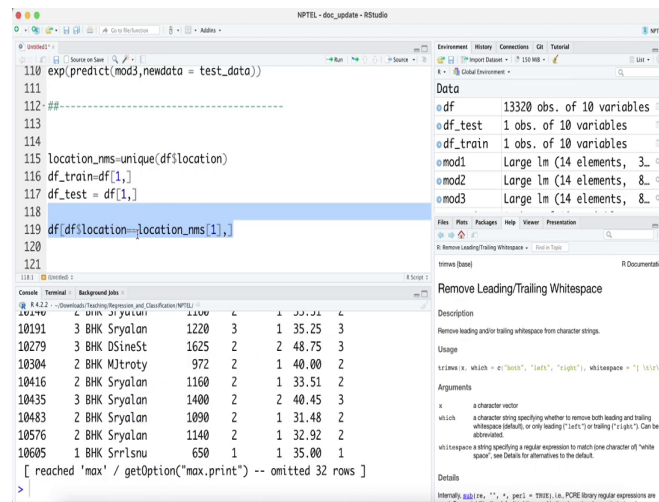
And then what I am going to do is for it is going to be a bit of a challenge df train df train equal to df 1 and kind of df test equal to df 1 we will just later put it as a drop these first row just to I just need a data structure at a framework ok. So, if I now I have just the framework the structure.

(Refer Slide Time: 43:59)

The image shows a dual-screen setup. On the left, a terminal window with a dark background displays R code being executed. The code reads a file named 'NPTEL.docx' into a data frame, filters it to keep only rows where 'location' is 'Chennappa Layout' or 'RW Extension Stage 2', and then prints the first few rows of the resulting data frame. On the right, a web browser window shows the NPTEL website. The website has a header with the NPTEL logo and navigation links. The main content area lists documents, with 'NPTEL.docx' at the top, showing its size (1.2 MB) and a download button. A small inset video of a person speaking is visible in the bottom right corner of the browser window.

So, if you just see I have some structure and later I will keep appending on that. So, that is that is my idea and later I will just drop the first row from both frame.

(Refer Slide Time: 44:15)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The script editor on the left contains the following R code:

```
110 exp(predict(mod3,newdata = test_data))
111
112 #####-----
113
114
115 location_nms=unique(df$location)
116 df_train=df[,]
117 df_test = df[,]
118
119 df[df$location==location_nms[1],]
120
121
```

The Environment pane on the right shows the following objects:

- df: 13320 obs. of 10 variables
- df_test: 1 obs. of 10 variables
- df_train: 1 obs. of 10 variables
- mod1: Large lm (14 elements, 3...)
- mod2: Large lm (14 elements, 8...)
- mod3: Large lm (14 elements, 8...)

The Console pane at the bottom shows the output of the code, displaying a table of data for the first location name 'BHK Sryalan'.

| 10191 | 3 | BHK Sryalan | 1220 | 3 | 1 35.25 3 |
|-------|---|-------------|------|---|-----------|
| 10279 | 3 | BHK DSineSt | 1625 | 2 | 2 48.75 3 |
| 10304 | 2 | BHK MJtroty | 972 | 2 | 1 40.00 2 |
| 10416 | 2 | BHK Sryalan | 1160 | 2 | 1 33.51 2 |
| 10435 | 3 | BHK Sryalan | 1400 | 2 | 2 40.45 3 |
| 10483 | 2 | BHK Sryalan | 1090 | 2 | 1 31.48 2 |
| 10576 | 2 | BHK Sryalan | 1140 | 2 | 1 32.92 2 |
| 10605 | 1 | BHK Srslsnu | 650 | 1 | 1 35.00 1 |

The right pane shows the documentation for the 'Remove Leading/Trailing Whitespace' function.

Remove Leading/Trailing Whitespace

Description: Remove leading and/or trailing whitespace from character strings.

Usage: `trimws(x, which = c("both", "left", "right"), whitespace = "[\\t\\r\\n]")`

Arguments: `x` is a character vector; `which` is a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.

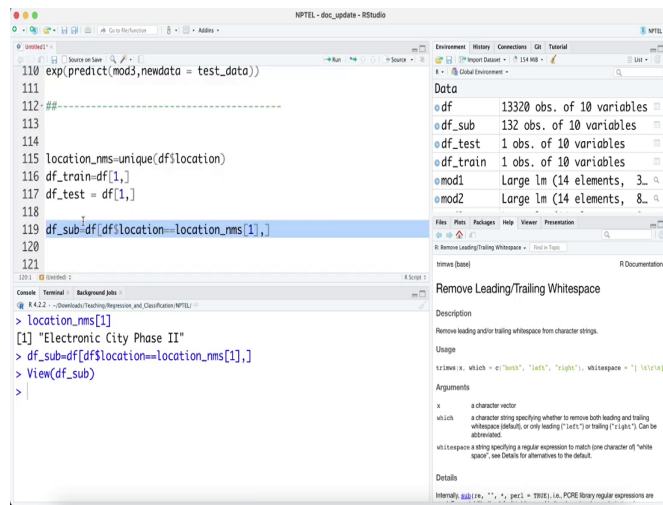
`whitespace` is a string specifying a regular expression to match (one character of "white space"; see Details for alternatives to the default).

Details: Internally `gsub("^[\\t\\r\\n]+", "", x, perl = TRUE)`, i.e. PCRE binary regular expressions are used.



Now what I am going to do I am going to say that ok df sub df equal to df dollar location equal to location names comma. So, if I just do that. So, these are the cases where you have bunch of names.

(Refer Slide Time: 44:52)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The editor pane contains R code for data manipulation. The environment pane on the right shows the following objects:

| Object | Value |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| df_sub | 132 obs. of 10 variables |
| df_test | 1 obs. of 10 variables |
| df_train | 1 obs. of 10 variables |
| mod1 | Large lm (14 elements, 3_) |
| mod2 | Large lm (14 elements, 8_) |

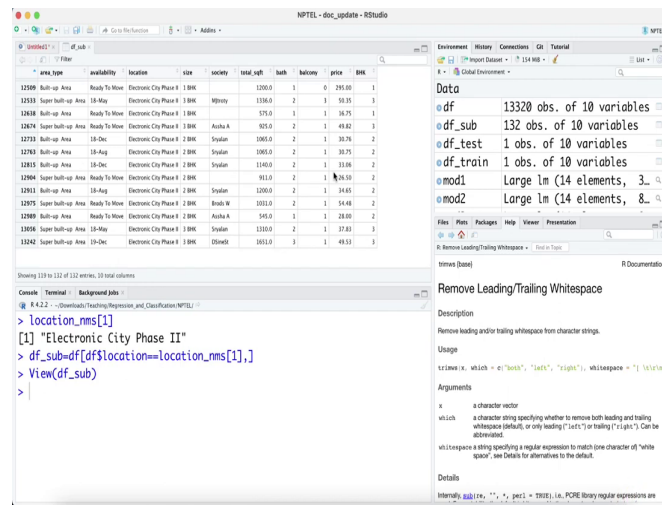
The console pane shows the following commands and output:

```
> location_rms[1]
[1] "Electronic City Phase II"
> df_sub=df[df$location==location_rms[1],]
> View(df_sub)
>
```



Say electronic phase let me just df call it df sub ok. So, let me just put it in df sub.

(Refer Slide Time: 45:09)



The screenshot shows the RStudio interface with the following components:

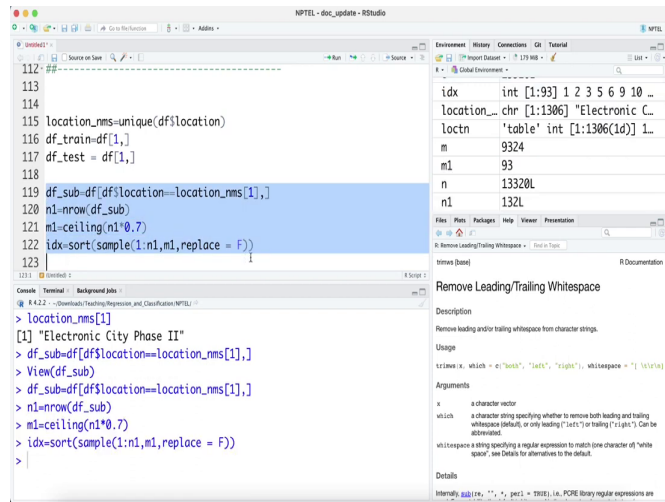
- Environment:** Lists data frames: `df` (13320 obs. of 10 variables), `df_sub` (132 obs. of 10 variables), `df_test` (1 obs. of 10 variables), `df_train` (1 obs. of 10 variables), `mod1` (Large lm (14 elements, 3...)), and `mod2` (Large lm (14 elements, 8...)).
- Files:** Shows a file named `NPTEL - disc_update - RStudio`.
- Terminal:** Contains the following R code:

```
> location_rms[1]
[1] "Electronic City Phase II"
> df_sub=df[df$location==location_rms[1],]
> View(df_sub)
>
```
- Documentation:** Shows the documentation for the `removeLeadingTrailingWhitespace` function, including its description, usage, arguments, and details.



So, what is that 132 observations. So, these are the instances there are 132 observations, which all belongs to Electronic City Phase 2 this location. Now, I have to split this guy into 2.

(Refer Slide Time: 45:25)



The screenshot shows the RStudio interface. The script editor on the left contains the following R code:

```
112 ##
113
114 location_rms=unique(df$location)
115 df_train=df[1,]
116 df_test = df[1,]
117
118 df_sub=df[df$location==location_rms[1,]]
119 n1=nrow(df_sub)
120 m1=ceiling(n1*0.7)
121 idx=sort(sample(1:n1,m1,replace = F))
122
123
```

The console on the bottom left shows the execution of the code:

```
> location_rms[1]
[1] "Electronic City Phase II"
> df_sub=df[df$location==location_rms[1,]]
> View(df_sub)
> df_sub=df[df$location==location_rms[1,]]
> n1=nrow(df_sub)
> m1=ceiling(n1*0.7)
> idx=sort(sample(1:n1,m1,replace = F))
>
```

The environment pane on the right shows the following variables:

| Variable | Value |
|--------------|-------------------------------|
| idx | int [1:93] 1 2 3 5 6 9 10 .. |
| location_rms | chr [1:1306] "Electronic C... |
| loctn | 'table' int [1:1306(1d)] 1.. |
| m | 9324 |
| m1 | 93 |
| n | 13320L |
| n1 | 132L |

The R Documentation pane on the right shows the documentation for the `strtrim` function, which removes leading and trailing whitespace from character strings.

Remove Leading/Trailing Whitespace

Description
Remove leading and/or trailing whitespace from character strings.

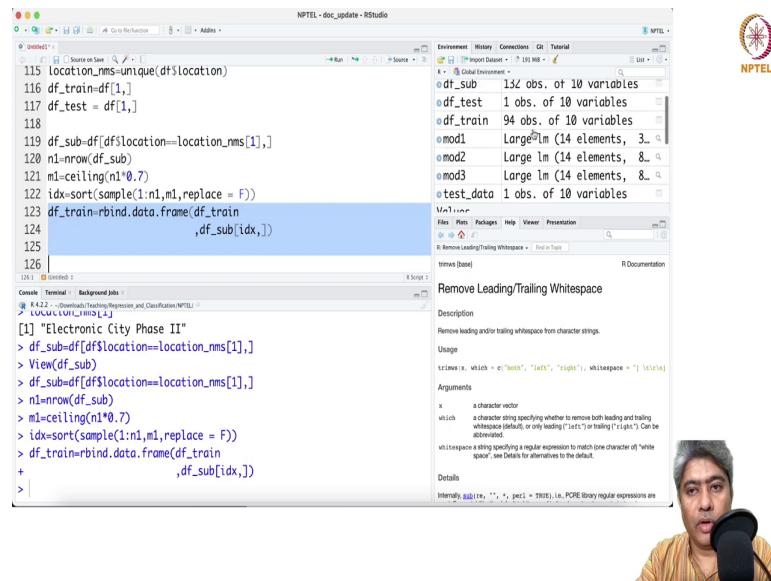
Usage
`strtrim(x, which = c("both", "left", "right"), whitespace = "[\\t\\r\\n]")`

Arguments
`x` a character vector
`which` a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
`whitespace` a string specifying a regular expression to match (one character of "white space"; see Details for alternatives to the default).

Details
Internally `strtrim`, `strtrimleft`, `strtrimright`, `strtrimboth`, `strtrimleftboth`, `strtrimrightboth`, `strtrimbothleft`, `strtrimbothright`, `strtrimbothleftboth`, `strtrimbothrightboth` are used.

So, what I am going to do n1 equal to nrow of df sub oopsie df sub. And then what I am going to do is m1 is equal to ceiling m n1 times 0.7 and then idx equal to 1 sort sample 1 is to n n1 comma m1 replace equals to false alright.

(Refer Slide Time: 46:26)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
115 location_rms=unique(df$location)
116 df_train=df[1,]
117 df_test = df[1,]
118
119 df_sub=df[df$location==location_rms[1,]]
120 n1=nrow(df_sub)
121 m1=ceiling(n1*0.7)
122 idx=sort(sample(1:n1,m1,replace = F))
123 df_train=rbind.data.frame(df_train
124                           ,df_sub[idx,])
125
126
```

The Environment pane on the right shows the following objects:

- df_sub: 132 obs. of 10 variables
- df_test: 1 obs. of 10 variables
- df_train: 94 obs. of 10 variables
- mod1: Large lm (14 elements, 3...)
- mod2: Large lm (14 elements, 8...)
- mod3: Large lm (14 elements, 8...)
- test_data: 1 obs. of 10 variables

The Console pane shows the following output:

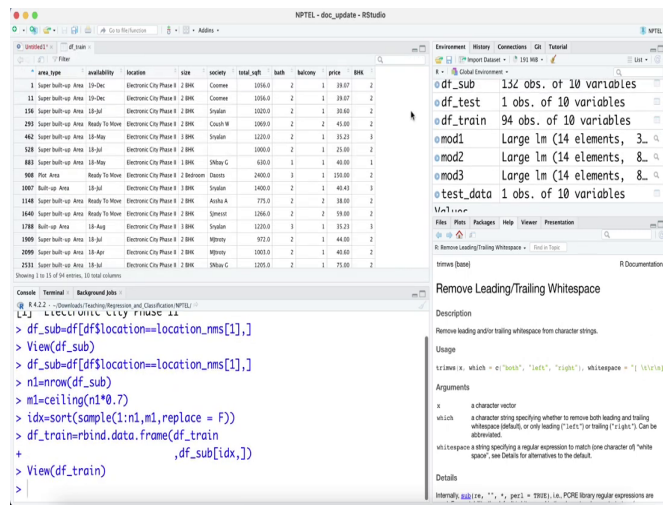
```
[1] "Electronic City Phase II"
> df_sub=df[df$location==location_rms[1,]]
> View(df_sub)
> df_sub=df[df$location==location_rms[1,]]
> n1=nrow(df_sub)
> m1=ceiling(n1*0.7)
> idx=sort(sample(1:n1,m1,replace = F))
> df_train=rbind.data.frame(df_train
+                           ,df_sub[idx,])
+
>
```

The R Documentation pane on the right shows the documentation for the `strtrim` function, titled "Remove Leading/Trailing Whitespace".

In the bottom right corner, there is a small video inset of a person speaking into a microphone.

So, this is what I am going to do and then df train is equal to df sub rbind dot data dot frame and df train comma df sub with idx that we have here ok. So, if I just now do that.

(Refer Slide Time: 47:09)



The screenshot shows the RStudio interface with the following components:

- Environment:** Lists objects including `df_sub` (132 obs. of 10 variables), `df_test` (1 obs. of 10 variables), `df_train` (94 obs. of 10 variables), `mod1` (Large lm (14 elements, 3...)), `mod2` (Large lm (14 elements, 8...)), `mod3` (Large lm (14 elements, 8...)), and `test_data` (1 obs. of 10 variables).
- Source:** Contains the following R code:

```
> df_sub=df[df$location==location_rms[1],]  
> View(df_sub)  
> df_sub=df[df$location==location_rms[1],]  
> n1=nrow(df_sub)  
> m1=ceiling(n1*0.7)  
> idx=sort(sample(1:n1,m1,replace = F))  
> df_train=bind.data.frame(df_train  
+                               ,df_sub[idx,])  
> View(df_train)  
>
```
- Console:** Shows the execution of the code in the Source pane.
- Help:** Displays the documentation for the `removeLeadingTrailingWhitespace` function, including its description, usage, arguments, and details.



So, now, you have df train as this so, the first thing and then ok.

(Refer Slide Time: 47:31)

The screenshot shows an RStudio interface with a script editor and a console window. The script editor contains the following R code:

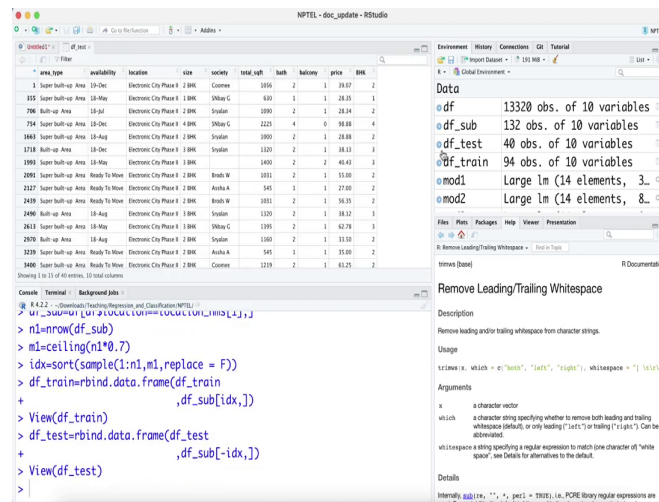
```
118
119 df_sub=df[df$location==location_nms[1,],]
120 n1=nrow(df_sub)
121 m1=ceiling(n1*0.7)
122 idx=sort(sample(1:n1,m1,replace = F))
123 df_train=rbind.data.frame(df_train
124                               ,df_sub[idx,])
125
126 df_test=rbind.data.frame(df_test
127                           ,df_sub[-idx,])
128
129
```

The console window shows the output of the code, including the structure of the data frame and the result of the whitespace removal function:

```
R 4.2.2 - Downloads\TrainingRegression_and_Classification\NPTSL1 //
> R console [R62] x86_64-suse-linux-gnu
> df_sub=df[df$location==location_nms[1,],]
> n1=nrow(df_sub)
> m1=ceiling(n1*0.7)
> idx=sort(sample(1:n1,m1,replace = F))
> df_train=rbind.data.frame(df_train
+                               ,df_sub[idx,])
+
> View(df_train)
> df_test=rbind.data.frame(df_test
+                           ,df_sub[-idx,])
>
```



(Refer Slide Time: 47:42)



The screenshot shows the RStudio interface with the following components:

- Environment:** Lists objects in the global environment:
 - `df`: 13320 obs. of 10 variables
 - `df_sub`: 132 obs. of 10 variables
 - `df_test`: 40 obs. of 10 variables
 - `df_train`: 94 obs. of 10 variables
 - `mod1`: Large lm (14 elements, 3...)
 - `mod2`: Large lm (14 elements, 8...)
- Source:** Contains the following R code:

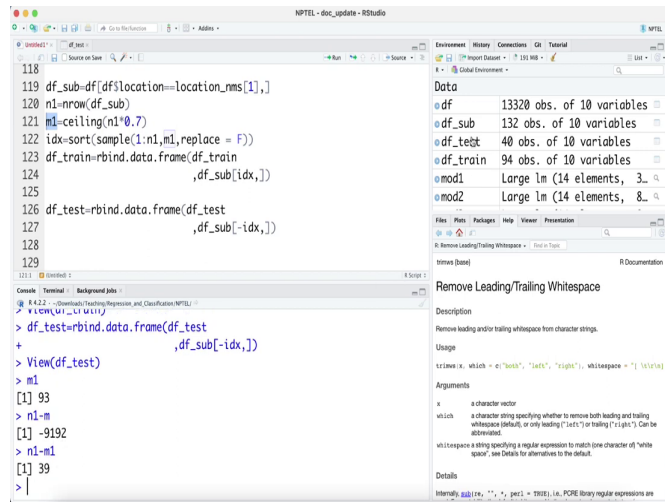

```

      # R 4.2.2 - Generated: Training, Test, and Validation Data
      # df_train = df[1:nrow(df)*0.7,]
      # df_test = df[(nrow(df)*0.7):(nrow(df)),]
      > n1=nrow(df_sub)
      > m1=ceiling(n1*0.7)
      > idx=sort(sample(1:n1,m1,replace = F))
      > df_train=bind.data.frame(df_train
      +                               ,df_sub[idx,])
      > View(df_train)
      > df_test=bind.data.frame(df_test
      +                               ,df_sub[-idx,])
      > View(df_test)
      >
      
```
- Console:** Shows the execution of the above code.
- Help:** Displays the documentation for the `str_remove` function, titled "Remove Leading/Trailing Whitespace".



And then similarly df test equal to df test and minus of that is going to give me the all test data set that half 40, 40 out of 94 of them goes to or 93 of them goes to train data set and 39 of them goes to test data set ok, alright.

(Refer Slide Time: 48:00)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The editor pane contains R code for data manipulation. The environment pane on the right shows the following objects:

| Object | Description |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| df_sub | 132 obs. of 10 variables |
| df_test | 40 obs. of 10 variables |
| df_train | 94 obs. of 10 variables |
| mod1 | Large lm (14 elements, 3_) |
| mod2 | Large lm (14 elements, 8_) |

The console shows the following commands and output:

```
> df_test=bind.data.frame(df_test, df_sub[-idx,])
> View(df_test)
> m1
[1] 93
> n1-m
[1] -9192
> n1-m1
[1] 39
>
```

The right pane shows the documentation for the `trimws` function, titled 'Remove Leading/Trailing Whitespace'.



Remove Leading/Trailing Whitespace

Description: Remove leading and/or trailing whitespace from character strings.

Usage: `trimws(x, which = c("both", "left", "right"), whitespace = "[\\t\\r\\n]")`

Arguments: `x` is a character vector; `which` is a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated; `whitespace` is a string specifying a regular expression to match (one character of "white space" - see Details for alternatives to the default).

Details: Internally `gsub("^[\\t\\r\\n]+", "", x, perl = TRUE)`, i.e. PCRE binary regular expressions are used.



So, m m1 is 93 and n1 minus m1 n1 minus m1 is 39. So, 39 goes to so, that is why it is 39 plus above on 40 and it is 93 plus 140 because we have one row added first we will drop that later.

(Refer Slide Time: 48:25)

The screenshot shows an RStudio interface with the following components:

- Source Editor:** Contains R code for data manipulation. Lines 123-129 are highlighted in blue.


```

123 df_sub$df[df$location==location_rms[1],]
124 n1=nrow(df_sub)
125 n1=ceiling(n1*0.7)
126 idx=sort(sample(1:n1,m1,replace = F))
127 df_train=rbind.data.frame(df_train
128                        ,df_sub[idx,])
129

```
- Environment:** Shows a data frame 'df' with 13320 observations and 10 variables.

| Variable | Observations | Variables |
|----------|--------------|-------------|
| df | 13320 | 10 |
| df_sub | 132 | 10 |
| df_test | 40 | 10 |
| df_train | 94 | 10 |
| mod1 | Large lm | 14 elements |
| mod2 | Large lm | 14 elements |
- Console:** Shows the execution of the code from the source editor.


```

> view(df_sub)
> df_test=rbind.data.frame(df_test
+                        ,df_sub[-idx,])
> View(df_test)
> m1
[1] 93
> n1=m
[1] -9192
> n1=m1
[1] 39
>

```
- Help Panel:** Displays the documentation for the `removeLeadingTrailingWhitespace` function.

removeLeadingTrailingWhitespace

Description

Remove leading and/or trailing whitespace from character strings.

Usage

```
trimws(x, which = c("both", "left", "right"), whitespace = "[ \\t\\n\\f]")
```

Arguments

 - `x`: a character vector
 - `which`: a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
 - `whitespace`: a string specifying a regular expression to match one character of "white space". See Details for alternatives to the default.

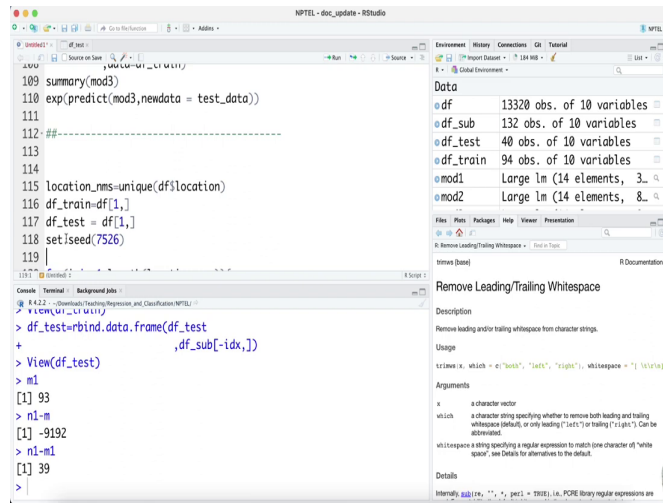
Details

Internally `trimws` uses `perl = TRUE` in `perl` to remove binary expressions on

In the bottom right corner, there is a small inset video of a man with grey hair speaking into a microphone.

So, now, whole thing I am going to repeat in a row for loop for i in 1 is to length of n g t h length of location names ok. And this is going to be the I alright and you have df train and test here and I just need a set dot seed here remember that this is this method is ok.

(Refer Slide Time: 49:15)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains R code for a linear model analysis. The console shows the execution of the code, resulting in a data frame with 93 rows and 39 columns. The environment pane on the right lists the objects in the workspace: df (13320 obs. of 10 variables), df_sub (132 obs. of 10 variables), df_test (40 obs. of 10 variables), df_train (94 obs. of 10 variables), mod1 (Large lm (14 elements, 3...)), and mod2 (Large lm (14 elements, 8...)). The help pane on the right displays the documentation for the `strtrim` function, which removes leading and trailing whitespace from character strings.

```
109 summary(mod3)
110 exp(predict(mod3,newdata = test_data))
111
112 ##-----
113
114
115 location_rms=unique(df$location)
116 df_train=df[1,]
117 df_test = df[1,]
118 set.seed(7526)
119 |
```

Environment

| Object | Class | Attributes |
|----------|------------|------------------------------|
| df | data.frame | 13320 obs. of 10 variables |
| df_sub | data.frame | 132 obs. of 10 variables |
| df_test | data.frame | 40 obs. of 10 variables |
| df_train | data.frame | 94 obs. of 10 variables |
| mod1 | lm | Large lm (14 elements, 3...) |
| mod2 | lm | Large lm (14 elements, 8...) |

Files

Remove Leading/Trailing Whitespace

Description

Remove leading and/or trailing whitespace from character strings.

Usage

```
strtrim(x, which = c("both", "left", "right"), whitespace = "[ \\t\\r\\n]")
```

Arguments

x

a character vector

which

a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.

whitespace

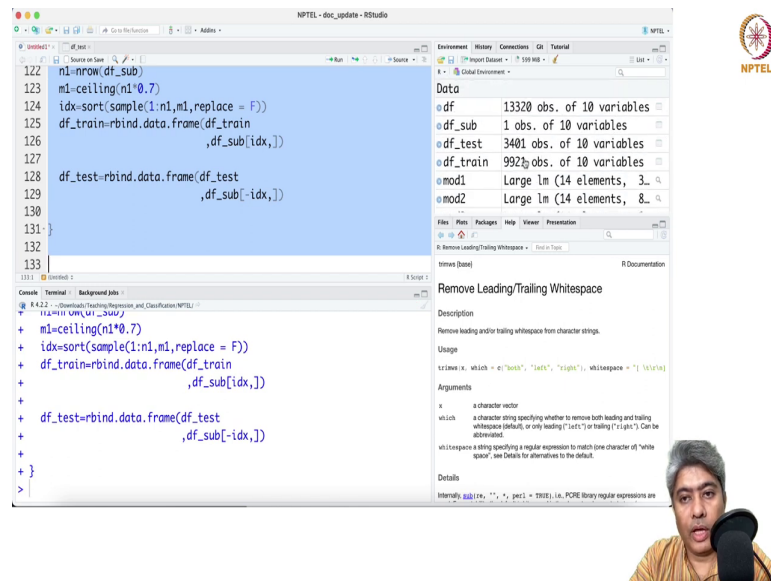
a string specifying a regular expression to match (one character of "white space"; see Details for alternatives to the default).

Details

Internally `strtrim`, `strtrimleft`, `strtrimright`, `strtrimboth`, `strtrimleftboth`, `strtrimrightboth`, `strtrimbothleft`, `strtrimbothright`, `strtrimbothleftboth`, `strtrimbothrightboth` are implemented using `strtrim`, `strtrimleft`, `strtrimright`, `strtrimboth`, `strtrimleftboth`, `strtrimrightboth`, `strtrimbothleft`, `strtrimbothright`, `strtrimbothleftboth`, `strtrimbothrightboth` respectively.



(Refer Slide Time: 49:28)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains R code for partitioning a dataset into training and testing sets. The code is as follows:

```
122 n1=nrow(df_sub)
123 m1=ceiling(n1*0.7)
124 idx=sort(sample(1:n1,m1,replace = F))
125 df_train=rbind.data.frame(df_train
126 ,df_sub[idx,])
127
128 df_test=rbind.data.frame(df_test
129 ,df_sub[-idx,])
130
131 }
132
133 }
```

The Environment pane on the right shows the following objects:

| Object | Class | Attributes |
|----------|------------|----------------------------|
| df | data.frame | 13320 obs. of 10 variables |
| df_sub | data.frame | 1 obs. of 10 variables |
| df_test | data.frame | 3401 obs. of 10 variables |
| df_train | data.frame | 9929 obs. of 10 variables |
| mod1 | lm | Large lm (14 elements, 3_) |
| mod2 | lm | Large lm (14 elements, 8_) |

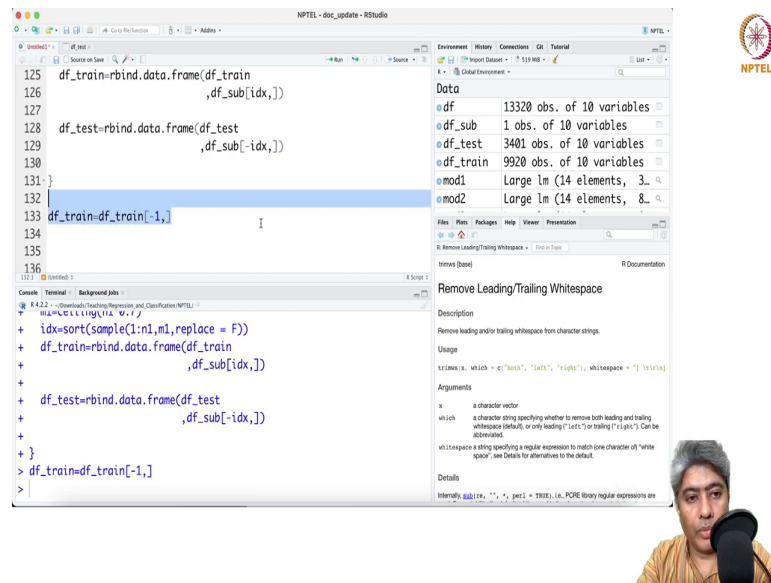
The Console pane shows the execution of the code, with the following output:

```
R 4.2.2 -- Generated by RStudio and R
> m1=ceiling(n1*0.7)
+ m1=ceiling(n1*0.7)
+ idx=sort(sample(1:n1,m1,replace = F))
+ df_train=rbind.data.frame(df_train
+ ,df_sub[idx,])
+
+ df_test=rbind.data.frame(df_test
+ ,df_sub[-idx,])
+
+ }
+ }
```

The video inset in the bottom right corner shows a man with a beard and a yellow shirt, speaking into a microphone.

Now, you have train is this many and test is this many and what I will do? I will just say in df train equal to df train minus 1.

(Refer Slide Time: 49:40)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The script editor contains the following R code:

```
125 df_train=rbind.data.frame(df_train
126                             ,df_sub[idx,])
127
128 df_test=rbind.data.frame(df_test
129                           ,df_sub[-idx,])
130
131 }
132
133 df_train=df_train[-1,]
134
135
136
```

The Environment pane on the right shows the following objects:

| Object | Class | Attributes |
|----------|------------|----------------------------|
| df | data.frame | 13320 obs. of 10 variables |
| df_sub | data.frame | 1 obs. of 10 variables |
| df_test | data.frame | 3401 obs. of 10 variables |
| df_train | data.frame | 9920 obs. of 10 variables |
| mod1 | lm | Large lm (14 elements, 3_) |
| mod2 | lm | Large lm (14 elements, 8_) |

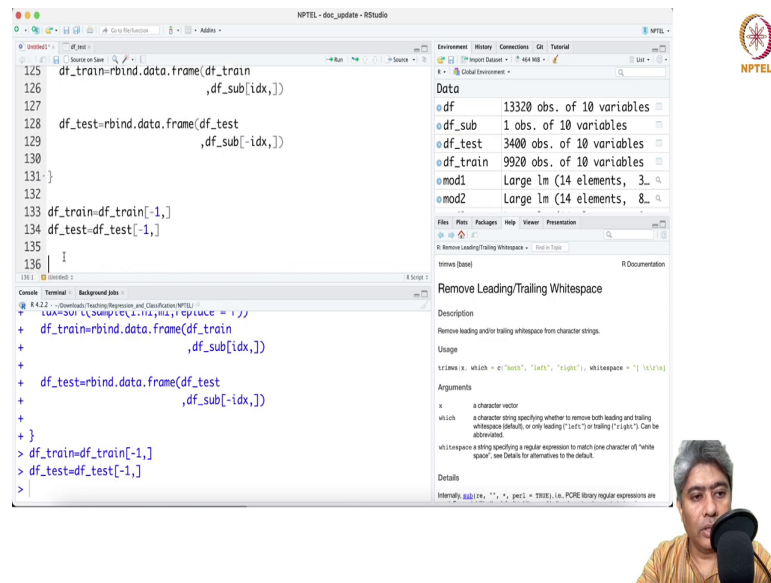
The Console pane shows the following output:

```
R 4.2.2 -- Generated by RStudio and R
+ ml=c(1,1,1,1,1,1,1,1,1,1)
+ idx=sort(sample(1:n1,m1,replace = F))
+ df_train=rbind.data.frame(df_train
+                             ,df_sub[idx,])
+
+ df_test=rbind.data.frame(df_test
+                           ,df_sub[-idx,])
+
+ }
+ > df_train=df_train[-1,]
+
+ >
```

A video inset in the bottom right corner shows a person speaking into a microphone.

So, I have 9921. Now, it is 9920 because remember that the first row I want to drop right I want to drop the first row.

(Refer Slide Time: 50:06)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains R code for data manipulation. The environment pane on the right shows the following objects:

| Object | Description |
|----------|----------------------------|
| df | 13320 obs. of 10 variables |
| df_sub | 1 obs. of 10 variables |
| df_test | 3400 obs. of 10 variables |
| df_train | 9920 obs. of 10 variables |
| mod1 | Large lm (14 elements, 3_) |
| mod2 | Large lm (14 elements, 8_) |

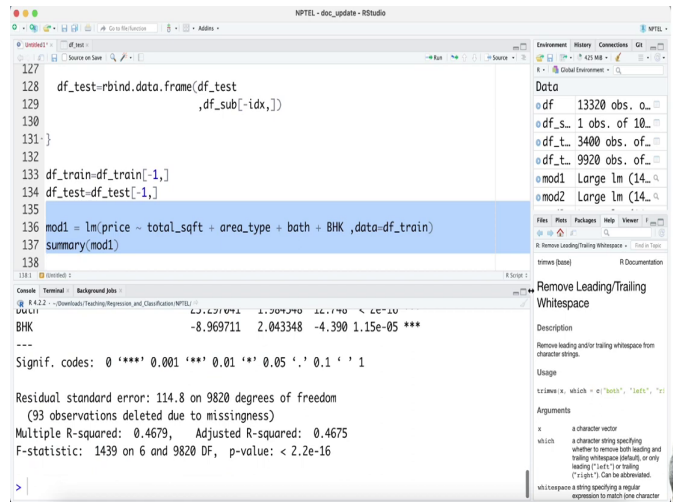
The console pane shows the following R code being executed:

```
125 df_train=rbind.data.frame(df_train
126                               ,df_sub[idx,])
127
128 df_test=rbind.data.frame(df_test
129                           ,df_sub[-idx,])
130
131 }
132
133 df_train=df_train[-1,]
134 df_test=df_test[-1,]
135
136 |
```

The video inset shows a man with grey hair and a beard, wearing a yellow shirt, speaking into a black microphone.

And similarly df test I want to drop the first row. Now my things are ready.

(Refer Slide Time: 50:15)




The screenshot shows an RStudio session. The script editor contains the following code:

```
127  
128 df_test=rbind(data.frame(df_test  
129 ,df_sub[-idx,])  
130 }  
131 }  
132  
133 df_train=df_train[-1,]  
134 df_test=df_test[-1,]  
135  
136 mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)  
137 summary(mod1)  
138
```

The console output shows the summary of the linear model:

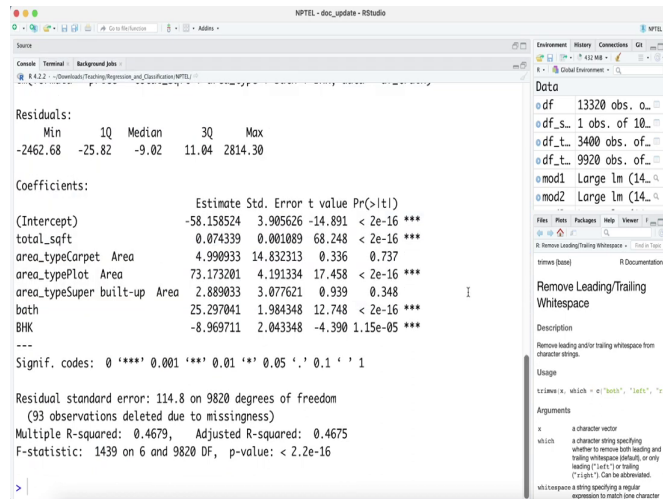
```
lm  
BHK  
-8.969711 2.043348 -4.390 1.15e-05 ***  
---  
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 114.8 on 9820 degrees of freedom  
(93 observations deleted due to missingness)  
Multiple R-squared: 0.4679, Adjusted R-squared: 0.4675  
F-statistic: 1439 on 6 and 9820 DF, p-value: < 2.2e-16
```

The sidebar on the right shows the Data environment with objects: df (13320 obs. of 10), df_s (1 obs. of 10), df_t (3400 obs. of 10), df_t (9920 obs. of 10), mod1 (Large lm (14)), and mod2 (Large lm (14)). A help window for 'Remove Leading/Trailing Whitespace' is also visible.



So, let me now run the model let me just copy this model and run it here ok.

(Refer Slide Time: 50:25)



The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The console displays the following output:

```
Residuals:
    Min       1Q   Median       3Q      Max
-2462.68  -25.82   -9.02   11.04  2814.30

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -58.158524    3.905626  -14.891 < 2e-16 ***
total_sqft      0.074339    0.001089   68.248 < 2e-16 ***
area_typeCarpet Area  4.990933    14.832313    0.336  0.737
area_typePlot Area  73.173201    4.191334   17.458 < 2e-16 ***
area_typeSuper built-up Area  2.889033    3.077621    0.939  0.348
bath           25.297041    1.984348   12.748 < 2e-16 ***
BHK            -8.969711    2.043348   -4.390 1.15e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

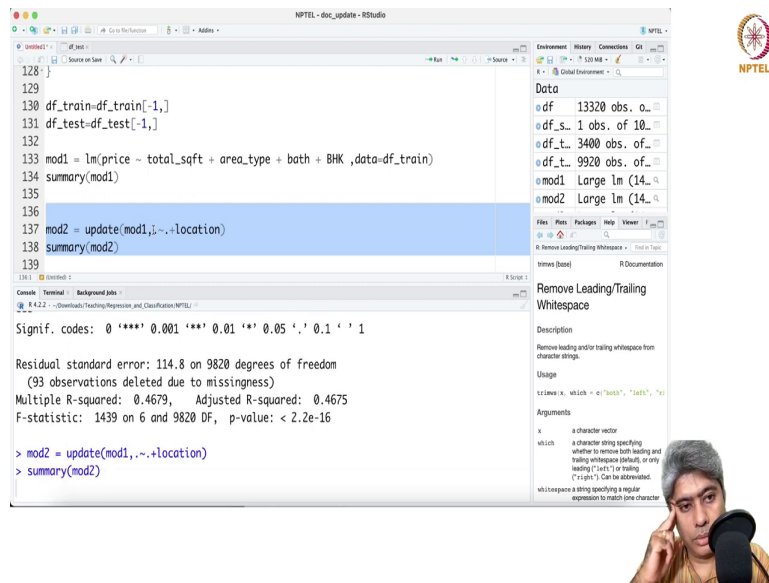
Residual standard error: 114.8 on 9820 degrees of freedom
(93 observations deleted due to missingness)
Multiple R-squared:  0.4679,    Adjusted R-squared:  0.4675
F-statistic: 1439 on 6 and 9820 DF,  p-value: < 2.2e-16
```

The right-hand pane shows the 'Data' tab with a list of objects: df, df_s, df_t, df_t, mod1, and mod2. The 'mod1' and 'mod2' objects are of class 'lm' and have 14 observations.



So, this is 46 percent about 46 percent accuracy total square feet BHK is negative which is bit of a I do not know this model has some problem and then we go and try to fit model2 ok.

(Refer Slide Time: 50:51)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
128 }
129
130 df_train=df_train[-1,]
131 df_test=df_test[-1,]
132
133 mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
134 summary(mod1)
135
136
137 mod2 = update(mod1,1,~.+location)
138 summary(mod2)
139
```

The console output shows the results of the linear model:

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 114.8 on 9820 degrees of freedom
(93 observations deleted due to missingness)
Multiple R-squared:  0.4679,    Adjusted R-squared:  0.4675
F-statistic: 1439 on 6 and 9820 DF,  p-value: < 2.2e-16

> mod2 = update(mod1,1,~.+location)
> summary(mod2)
```

The right-hand pane shows the "Data" tab with the following information:

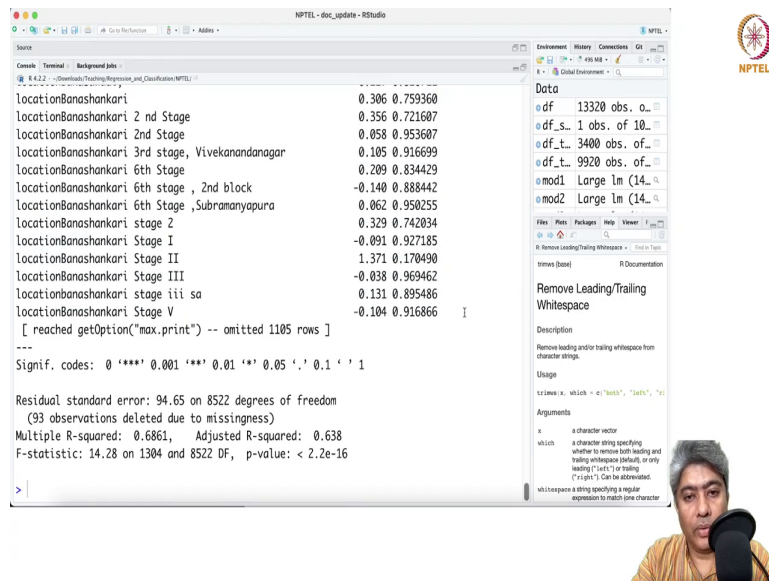
- df: 13320 obs. of...
- df_s: 1 obs. of 10...
- df_t: 3400 obs. of...
- df_t: 9920 obs. of...
- mod1: Large lm (14...
- mod2: Large lm (14...

Below the "Data" tab, the "Remove Leading/Trailing Whitespace" function is shown. The description states: "Remove leading and/or trailing whitespace from character strings." The usage is: `trimws(x, which = c("both", "left", "right"))`. The arguments are: `x` is a character vector; `which` is a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.

In the bottom right corner, there is a small image of a person with a hand on their chin, appearing to be in deep thought.

It will take little bit time because it will create all the dummy variables.

(Refer Slide Time: 51:04)



The screenshot shows an RStudio window titled "NPTEL - doc_update - RStudio". The console displays the output of a linear model fit, including coefficients, significance codes, and model statistics. The environment pane on the right shows the data frame "df" with 13320 observations and two model objects, "mod1" and "mod2". A video inset in the bottom right corner shows a man speaking into a microphone.

```
locationBanashankari          0.306 0.759360
locationBanashankari 2 nd Stage 0.356 0.721607
locationBanashankari 2nd Stage  0.058 0.953607
locationBanashankari 3rd stage, Vivekanandanagar 0.105 0.916699
locationBanashankari 6th Stage  0.209 0.834429
locationBanashankari 6th stage , 2nd block -0.140 0.888442
locationBanashankari 6th Stage ,Subramanyapura 0.062 0.950255
locationBanashankari stage 2     0.329 0.742034
locationBanashankari Stage I     -0.091 0.927185
locationBanashankari Stage II    1.371 0.170490
locationBanashankari Stage III   -0.038 0.969462
locationBanashankari stage iii sa 0.131 0.895486
locationBanashankari Stage V     -0.104 0.916866
[ reached getOption("max.print") -- omitted 1105 rows ]

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 94.65 on 8522 degrees of freedom
(93 observations deleted due to missingness)
Multiple R-squared:  0.6861,    Adjusted R-squared:  0.638 
F-statistic: 14.28 on 1304 and 8522 DF,  p-value: < 2.2e-16
```

Remove Leading/Trailing Whitespace

Description
Remove leading and/or trailing whitespace from character strings.

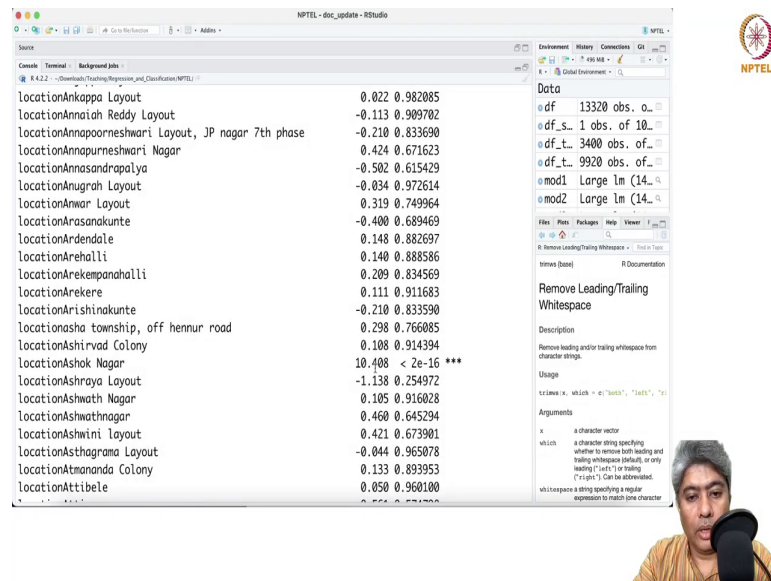
Usage
`trimws(x, which = c("both", "left", "right"))`

Arguments
`x` a character vector
`which` a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.

`whitespace` a string specifying a regular expression to match (pipe character

Now, it is 63.8 percent accuracy 63.8 percent accuracy.

(Refer Slide Time: 51:13)

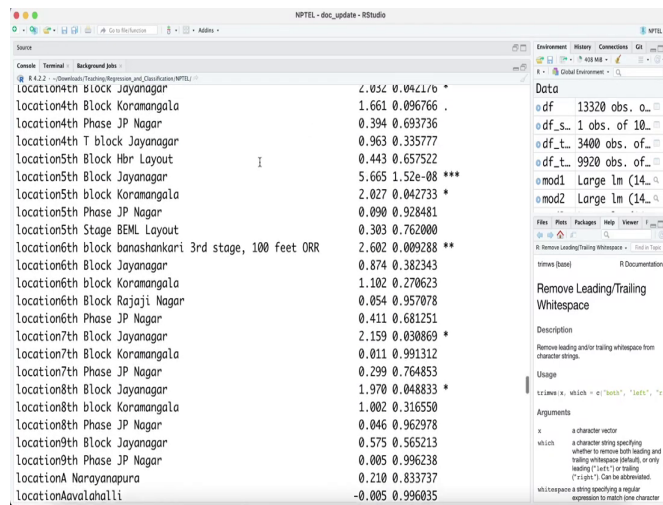


The RStudio interface displays a data frame with the following columns: location, p-value, and another p-value. The 'locationAshok Nagar' row is highlighted with a red box, indicating a very high p-value of 10.408.

| location | p-value | another p-value |
|---|---------|-----------------|
| locationAnkappa Layout | 0.022 | 0.982085 |
| locationAnnaiah Reddy Layout | -0.113 | 0.909702 |
| locationAnnapoorneshwari Layout, JP nagar 7th phase | -0.210 | 0.833690 |
| locationAnnapurneshwari Nagar | 0.424 | 0.671623 |
| locationAnnasandrapalya | -0.502 | 0.615429 |
| locationAnugrah Layout | -0.034 | 0.972614 |
| locationAnwar Layout | 0.319 | 0.749964 |
| locationArasanakunte | -0.400 | 0.689469 |
| locationArendale | 0.148 | 0.882697 |
| locationArehalli | 0.140 | 0.88586 |
| locationArekempandahalli | 0.209 | 0.834569 |
| locationArekere | 0.111 | 0.911683 |
| locationArishinakunte | -0.210 | 0.833590 |
| locationasha township, off hennur road | 0.298 | 0.766085 |
| locationAshirvad Colony | 0.108 | 0.914394 |
| locationAshok Nagar | 10.408 | < 2e-16 *** |
| locationAshraya Layout | -1.138 | 0.254972 |
| locationAshwath Nagar | 0.105 | 0.916028 |
| locationAshwathnagar | 0.460 | 0.645234 |
| locationAshwini layout | 0.421 | 0.673901 |
| locationAsthagrama Layout | -0.044 | 0.965078 |
| locationAtmananda Colony | 0.133 | 0.893953 |
| locationAttibele | 0.050 | 0.960100 |

And Ashok Nagar has a very high p value.

(Refer Slide Time: 51:21)

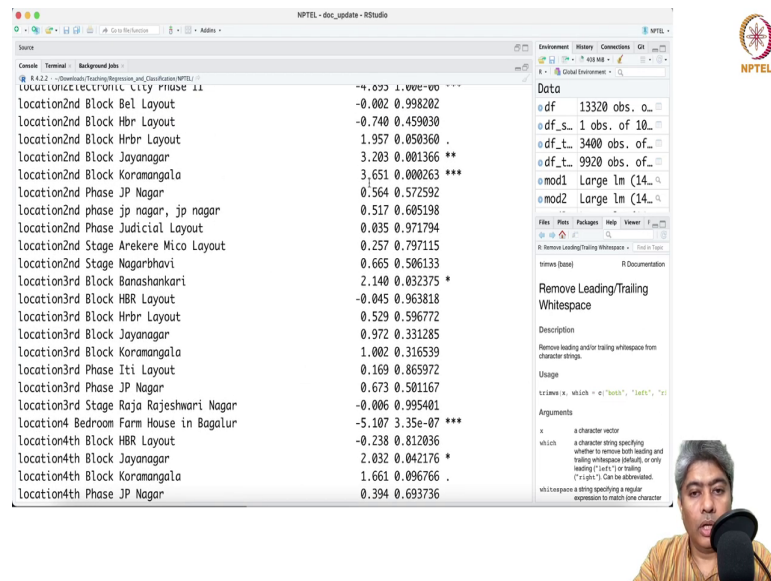


The image shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The main editor displays a data frame with 20 rows and 2 columns. The rows are labeled with location names and the columns contain numerical values. The right sidebar shows the 'Data' pane with a list of objects: 'df' (13320 obs. of 2), 'df_s' (1 obs. of 10), 'df_t' (3400 obs. of 2), 'df_t_1' (9920 obs. of 2), 'mod1' (Large lm (14)), and 'mod2' (Large lm (14)). Below the 'Data' pane is the 'Remove Leading/Trailing Whitespace' function help page, which includes a description, usage, and arguments.

| Location | Value 1 | Value 2 |
|--|---------|--------------|
| location4th Block Jayanagar | 2.032 | 0.0421/b * |
| location4th Block Koramangala | 1.661 | 0.096766 . |
| location4th Phase JP Nagar | 0.394 | 0.693736 |
| location4th T block Jayanagar | 0.963 | 0.335777 |
| location5th Block Hbr Layout | 0.443 | 0.657522 |
| location5th Block Jayanagar | 5.665 | 1.52e-08 *** |
| location5th block Koramangala | 2.027 | 0.042733 * |
| location5th Phase JP Nagar | 0.090 | 0.928481 |
| location5th Stage BEML Layout | 0.303 | 0.762000 |
| location6th block banashankari 3rd stage, 100 feet ORR | 2.602 | 0.009288 ** |
| location6th Block Jayanagar | 0.874 | 0.382343 |
| location6th block Koramangala | 1.102 | 0.270623 |
| location6th Block Rajaji Nagar | 0.054 | 0.957078 |
| location6th Phase JP Nagar | 0.411 | 0.681251 |
| location7th Block Jayanagar | 2.159 | 0.030869 * |
| location7th Block Koramangala | 0.011 | 0.991312 |
| location7th Phase JP Nagar | 0.299 | 0.764853 |
| location8th Block Jayanagar | 1.970 | 0.048833 * |
| location8th block Koramangala | 1.002 | 0.316550 |
| location8th Phase JP Nagar | 0.046 | 0.962978 |
| location9th Block Jayanagar | 0.575 | 0.565213 |
| location9th Phase JP Nagar | 0.005 | 0.996238 |
| locationA Narayanapura | 0.210 | 0.833737 |
| locationAvalahalli | -0.005 | 0.996035 |



(Refer Slide Time: 51:26)



The screenshot displays the RStudio interface. The main console window shows the output of a linear model fit, with the following data:

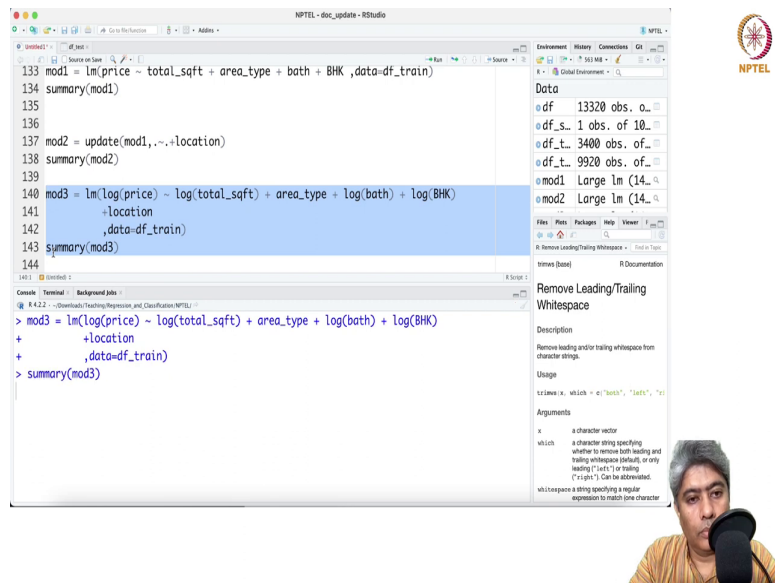
| Location | Estimate | Std. Error | t-value | Pr(> t) |
|---|----------|------------|-----------|----------|
| location2nd Block Bel Layout | -0.002 | 0.998202 | -0.002 | 1.000000 |
| location2nd Block Hbr Layout | -0.740 | 0.459030 | -1.612 | 0.111000 |
| location2nd Block Hbr Layout | 1.957 | 0.050360 | 38.850 | <.0001 |
| location2nd Block Jayanagar | 3.203 | 0.001366 | 2343.000 | <.0001 |
| location2nd Block Koramangala | 3.651 | 0.000263 | 13878.000 | <.0001 |
| location2nd Phase JP Nagar | 0.564 | 0.572592 | 0.986 | 0.325000 |
| location2nd phase jp nagar, jp nagar | 0.517 | 0.605198 | 0.854 | 0.395000 |
| location2nd Phase Judicial Layout | 0.035 | 0.971794 | 0.036 | 0.972000 |
| location2nd Stage Arekere Mico Layout | 0.257 | 0.797115 | 0.322 | 0.745000 |
| location2nd Stage Nagarbhavi | 0.665 | 0.506133 | 1.313 | 0.190000 |
| location3rd Block Banashankari | 2.140 | 0.032375 | 66.150 | <.0001 |
| location3rd Block HBR Layout | -0.045 | 0.963818 | -0.047 | 0.962000 |
| location3rd Block Hbr Layout | 0.529 | 0.596772 | 0.887 | 0.375000 |
| location3rd Block Jayanagar | 0.972 | 0.331285 | 2.934 | 0.003000 |
| location3rd Block Koramangala | 1.002 | 0.316539 | 3.166 | 0.001000 |
| location3rd Phase Iti Layout | 0.169 | 0.865972 | 0.195 | 0.845000 |
| location3rd Phase JP Nagar | 0.673 | 0.501167 | 1.343 | 0.180000 |
| location3rd Stage Raja Rajeshwari Nagar | -0.006 | 0.995401 | -0.006 | 0.999000 |
| location4 Bedroom Farm House in Bagalur | -5.107 | 3.35e-07 | -1524.000 | <.0001 |
| location4th Block HBR Layout | -0.238 | 0.812036 | -0.293 | 0.770000 |
| location4th Block Jayanagar | 2.032 | 0.042176 | 48.190 | <.0001 |
| location4th Block Koramangala | 1.661 | 0.096766 | 17.080 | <.0001 |
| location4th Phase JP Nagar | 0.394 | 0.693736 | 0.568 | 0.575000 |

The right-hand pane shows the 'Data' tab with a list of objects: df (13320 obs. of 10), df_s (1 obs. of 10), df_t (3400 obs. of 10), df_t (9920 obs. of 10), mod1 (Large lm (14)), and mod2 (Large lm (14)). Below this is the 'Remove Leading/Trailing Whitespace' function documentation.

In the bottom right corner, there is a small video feed of a man with grey hair, wearing a yellow shirt, speaking into a black microphone.

Jayanagar, Koramangala has a very high p value alright. So, we will see what is it is happening here ok and then finally, we will take the third model and we will see what is happening here.

(Refer Slide Time: 51:58)



The image displays the RStudio interface with the following components:

- Source Editor:** Contains R code for fitting and updating linear models. Lines 140-144 are highlighted in blue.
- Environment:** Shows the objects in the global environment, including data frames and model objects.
- Console:** Displays the output of the R commands.
- Documentation:** Shows the help page for the `strtrim` function.

R Code (Lines 140-144):

```
mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
          + location
          ,data=df_train)
summary(mod3)
```

Environment:

- df: 13320 obs. of...
- df_s: 1 obs. of 10...
- df_t: 3400 obs. of...
- df_t: 9920 obs. of...
- mod1: Large lm (14...
- mod2: Large lm (14...

Console:

```
> mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
+           + location
+           ,data=df_train)
> summary(mod3)
```

Documentation: Remove Leading/Trailing Whitespace

Description: Remove leading and/or trailing whitespace from character strings.

Usage:

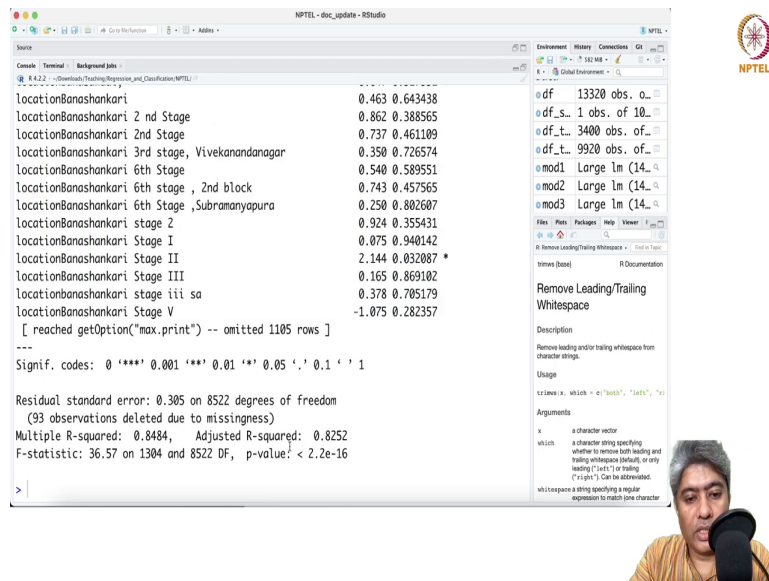
```
strtrim(x, which = c("both", "left", "r"))
```

Arguments:

- `x`: a character vector
- `which`: a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
- `whitespace`: a string specifying a regular expression to match (pipe character

A video inset in the bottom right corner shows a man speaking into a microphone.

(Refer Slide Time: 52:13)



The screenshot displays an RStudio window titled 'NPTEL - disc_update - RStudio'. The main console area shows the output of a linear model fit, including a table of coefficients, significance codes, and model statistics. The coefficients table lists 13 predictors with their estimates and standard errors. The significance codes are: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1. The model statistics include: Residual standard error: 0.305 on 8522 degrees of freedom (93 observations deleted due to missingness), Multiple R-squared: 0.8484, Adjusted R-squared: 0.8252, and F-statistic: 36.57 on 1304 and 8522 DF, p-value: < 2.2e-16. On the right side, the 'Environment' pane shows the 'Global Environment' with several objects: 'df' (13320 obs. of a...), 'df_s' (1 obs. of 10...), 'df_t' (3400 obs. of...), 'df_t' (9920 obs. of...), 'mod1' (Large lm (14...)), 'mod2' (Large lm (14...)), and 'mod3' (Large lm (14...)). Below the environment pane, the 'Remove Leading/Trailing Whitespace' function is shown with its description and usage. In the bottom right corner, there is a small video inset of a man with grey hair, wearing a yellow shirt, speaking into a black microphone.

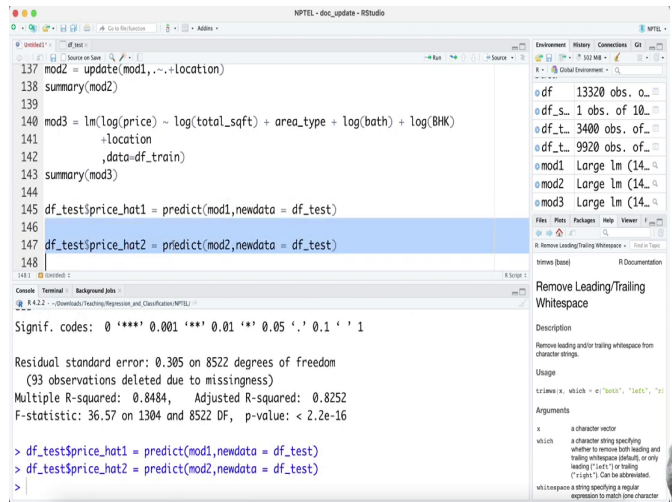
| locationBanashankari | 0.463 | 0.643438 |
|--|--------|------------|
| locationBanashankari 2 nd Stage | 0.862 | 0.388565 |
| locationBanashankari 2nd Stage | 0.737 | 0.461109 |
| locationBanashankari 3rd stage, Vivekanandanagar | 0.350 | 0.726574 |
| locationBanashankari 6th Stage | 0.540 | 0.589551 |
| locationBanashankari 6th stage , 2nd block | 0.743 | 0.457565 |
| locationBanashankari 6th Stage ,Subramanyapura | 0.250 | 0.802607 |
| locationBanashankari stage 2 | 0.924 | 0.355431 |
| locationBanashankari Stage I | 0.075 | 0.940142 |
| locationBanashankari Stage II | 2.144 | 0.032087 * |
| locationBanashankari Stage III | 0.165 | 0.869102 |
| locationBanashankari stage iii sa | 0.378 | 0.705179 |
| locationBanashankari Stage V | -1.075 | 0.282357 |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.305 on 8522 degrees of freedom
(93 observations deleted due to missingness)
Multiple R-squared: 0.8484, Adjusted R-squared: 0.8252
F-statistic: 36.57 on 1304 and 8522 DF, p-value: < 2.2e-16

So, in the third model it is giving us accuracy up to 82 percent, but these are in sample accuracy remember that. So, you have to be bit careful about these kind of accuracies. So, if you run. So, first we have to check out of the sample accuracy. So, let me just copy these two lines and paste it here ok.

(Refer Slide Time: 52:41)



The screenshot shows an RStudio session titled "NPTEL - disc_update - RStudio". The script editor contains the following code:

```
137 mod2 = update(mod1, ~., +location)
138 summary(mod2)
139
140 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
141         +location
142         ,data=df_train)
143 summary(mod3)
144
145 df_test$price_hat1 = predict(mod1,newdata = df_test)
146
147 df_test$price_hat2 = predict(mod2,newdata = df_test)
148
```

The console output shows the results of the model training and prediction:

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.305 on 8522 degrees of freedom
(93 observations deleted due to missingness)
Multiple R-squared:  0.8484,    Adjusted R-squared:  0.8252
F-statistic: 36.57 on 1304 and 8522 DF,  p-value: < 2.2e-16



> df_test$price_hat1 = predict(mod1,newdata = df_test)
> df_test$price_hat2 = predict(mod2,newdata = df_test)
>
```

The environment pane on the right shows the following objects:

- df: 13320 obs. of...
- df_s: 1 obs. of 10...
- df_t: 3400 obs. of...
- df_t: 9920 obs. of...
- mod1: Large lm (14...
- mod2: Large lm (14...
- mod3: Large lm (14...

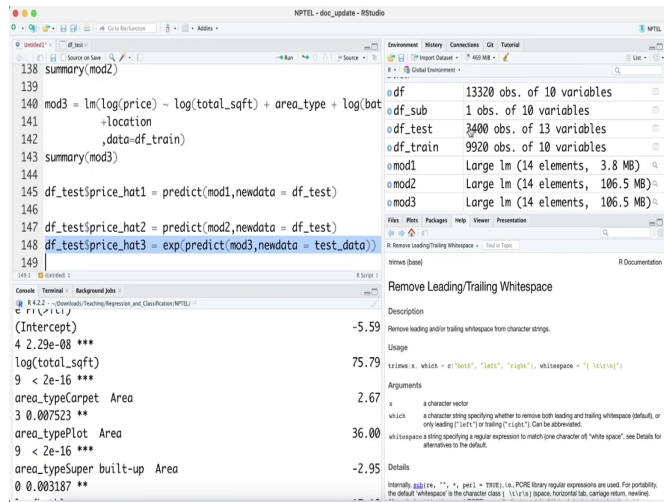
The console output also includes the following text:

```
Remove Leading/Trailing Whitespace
Description
Remove leading and/or trailing whitespace from character strings.
Usage
trimws(x, which = c("both", "left", "right"))
Arguments
x
a character vector
which
a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right") whitespace. Can be abbreviated.
whitespace
a string specifying a regular expression to match (pipe character
```



And similarly if I see now there is no problem now there is no problem and it is it gives us the exact accuracy thing we wish it.

(Refer Slide Time: 53:00)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The editor pane contains R code for fitting three linear models (mod1, mod2, mod3) and predicting values. The environment pane shows the objects created, including data frames (df, df_sub, df_test, df_train) and linear models (mod1, mod2, mod3). The console pane shows the output of the code, including the summary of mod3 and the predicted values for df_test. The right pane shows the documentation for the `remove_leading_trailing_whitespace` function.

```
138 summary(mod2)
139
140 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bat
141 +location
142 ,data=df_train)
143 summary(mod3)
144
145 df_test$price_hat1 = predict(mod1,newdata = df_test)
146
147 df_test$price_hat2 = predict(mod2,newdata = df_test)
148 df_test$price_hat3 = exp(predict(mod3,newdata = test_data))
149
```

Environment:

| Object | Class | Attributes |
|----------|------------|----------------------------------|
| df | data.frame | 13320 obs. of 10 variables |
| df_sub | data.frame | 1 obs. of 10 variables |
| df_test | data.frame | 3400 obs. of 13 variables |
| df_train | data.frame | 9920 obs. of 10 variables |
| mod1 | lm | Large lm (14 elements, 3.8 MB) |
| mod2 | lm | Large lm (14 elements, 106.5 MB) |
| mod3 | lm | Large lm (14 elements, 106.5 MB) |

Console Output:

```
(Intercept) -5.59
log(total_sqft) 75.79
area_typeCarpet Area 2.67
area_typeLot Area 36.00
area_typeSuper built-up Area -2.95
```

Documentation: Remove Leading/Trailing Whitespace

Description: Remove leading and/or trailing whitespace from character strings.

Usage: `remove_leading_trailing_whitespace(x, which = c("both", "left", "right"), whitespace = "[\\t\\n\\r]")`

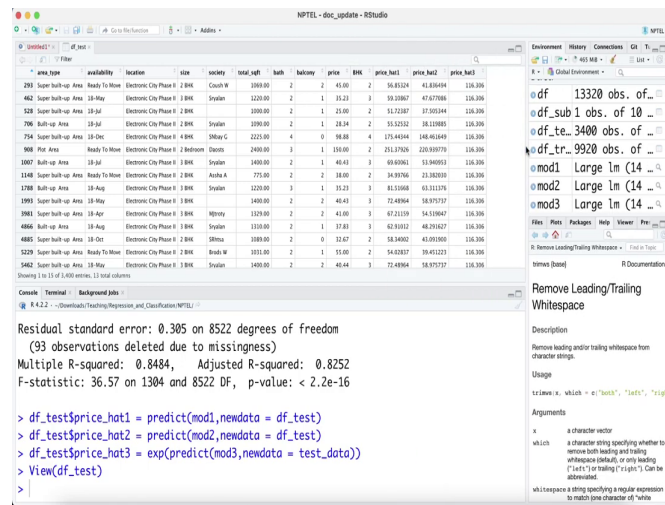
Arguments:

- `x`: a character vector
- `which`: a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
- `whitespace`: a string specifying a regular expression to match (one character of "white space", see Details for alternatives to the default).

Details: Internally, `gsub` (i.e., `perl = TRUE`) is used. PCRE library regular expressions are used. For portability, the default "whitespace" is the character class `[\\t\\n\\r]` (space, horizontal tab, carriage return, newline).

And then this is the third one third model and if we just look into the test.

(Refer Slide Time: 53:13)



The screenshot shows the RStudio interface with the following components:

- Environment Pane:** Displays a data frame with columns: area_type, availability, location, size, society, total_sqft, bath, balcony, price, bhk, price_hat1, price_hat2, price_hat3. It lists 1300 observations.
- Console:** Shows the output of a linear regression model fit. The text reads:

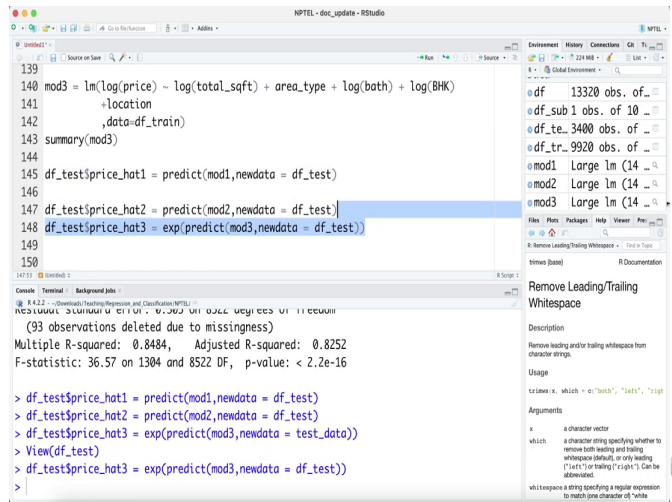

```
Residual standard error: 0.305 on 8522 degrees of freedom
(93 observations deleted due to missingness)
Multiple R-squared: 0.8484, Adjusted R-squared: 0.8252
F-statistic: 36.57 on 1304 and 8522 DF, p-value: < 2.2e-16
```

 Below this, the following R code is entered:


```
> df_test$price_hat1 = predict(mod1, newdata = df_test)
> df_test$price_hat2 = predict(mod2, newdata = df_test)
> df_test$price_hat3 = exp(predict(mod3, newdata = test_data))
> View(df_test)
```
- Sidebar:** Contains a video feed of a person speaking into a microphone, likely the presenter.

So, we have price hat from first model second model and the third model ok. So, the third model looks like it is little bit maybe it is not that great model I do not know some issue is there it is all same places maybe, we will see what is happening there not test data.

(Refer Slide Time: 53:53)



The image shows an RStudio window titled "NPTEL - disc_update - RStudio". The script editor contains the following R code:

```
139
140 mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
141           + location
142           ,data=df_train)
143 summary(mod3)
144
145 df_test$price_hat1 = predict(mod1,newdata = df_test)
146
147 df_test$price_hat2 = predict(mod2,newdata = df_test)
148 df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
149
150
```

The console output shows the summary of the linear model 'mod3':

```
lm model data: df_train
(93 observations deleted due to missingness)
Multiple R-squared: 0.8484, Adjusted R-squared: 0.8252
F-statistic: 36.57 on 1304 and 8522 DF, p-value: < 2.2e-16
```



Below the summary, the following commands are entered in the console:

```
> df_test$price_hat1 = predict(mod1,newdata = df_test)
> df_test$price_hat2 = predict(mod2,newdata = df_test)
> df_test$price_hat3 = exp(predict(mod3,newdata = test_data))
> View(df_test)
> df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
>
```

The environment pane on the right shows the following objects:

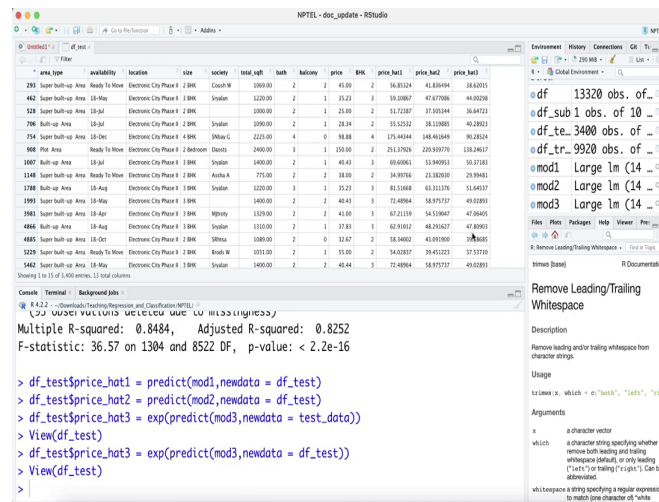
- df: 13320 obs. of ...
- df_sub: 1 obs. of 10 ...
- df_te: 3400 obs. of ...
- df_tr: 9920 obs. of ...
- mod1: Large lm (14 ...)
- mod2: Large lm (14 ...)
- mod3: Large lm (14 ...)

The help pane on the right shows the documentation for the `strLines` function, which is used to remove leading and trailing whitespace from character strings.



We want a df test we want df test to be here df test there was a mistake now it is doing well alright.

(Refer Slide Time: 53:58)



The screenshot shows an RStudio interface with the following components:

- Environment:** Shows a data frame 'df' with 13320 observations and 10 variables.
- Console:** Contains R code and output:


```

      > df_test$price_hat1 = predict(mod1,newdata = df_test)
      > df_test$price_hat2 = predict(mod2,newdata = df_test)
      > df_test$price_hat3 = exp(predict(mod3,newdata = test_data))
      > View(df_test)
      > df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
      > View(df_test)
      >
      
```
- Terminal:** Shows the output of the R code:

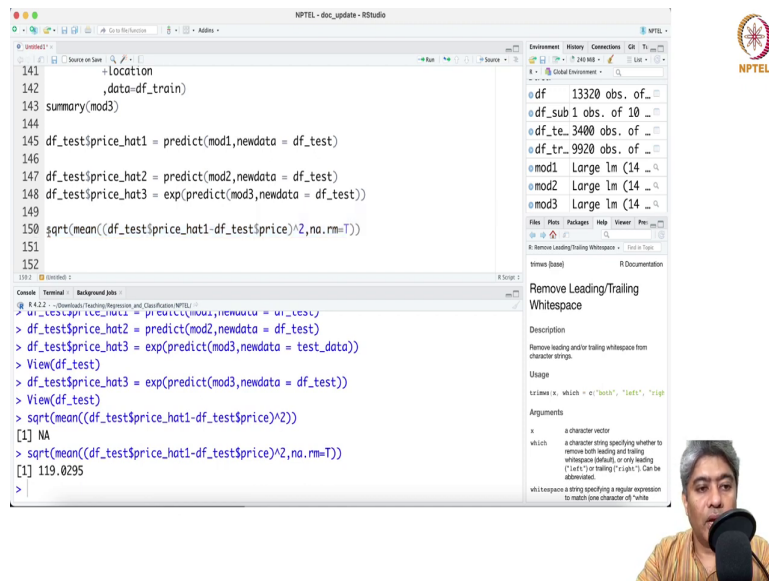

```

      Multiple R-squared:  0.8484,    Adjusted R-squared:  0.8252
      F-statistic: 36.57 on 1304 and 8522 DF,  p-value: < 2.2e-16
      
```
- Background Jobs:** Shows a job running in the background.
- Remove Leading/Trailing Whitespace:** A panel on the right side of the console showing the description and usage of the `trimws` function.



Now, what I am going to do now if you look into this particular case the price is 150 price hat to the first model predicted 251, second model predicted 220 and third model predicted 138. So, this from this one case looks like the first third model has done good job, but we have to first compute the root mean square error. So, what we will do?

(Refer Slide Time: 54:40)



The image shows an RStudio window titled 'NPTEL - disc_update - RStudio'. The editor pane contains R code for model evaluation. The console shows the execution of the code, resulting in the RMSE value of 119.0295. The environment pane on the right lists several objects: 'df' (13320 obs. of 10 variables), 'df_sub' (1 obs. of 10 variables), 'df_test' (3400 obs. of 10 variables), 'df_train' (9920 obs. of 10 variables), 'mod1' (Large lm (14 parameters)), 'mod2' (Large lm (14 parameters)), and 'mod3' (Large lm (14 parameters)). A video inset in the bottom right corner shows a man speaking into a microphone.

```
141 +location
142 ,data=df_train)
143 summary(mod3)
144
145 df_test$price_hat1 = predict(mod1,newdata = df_test)
146
147 df_test$price_hat2 = predict(mod2,newdata = df_test)
148 df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
149
150 sqrt(mean((df_test$price_hat1-df_test$price)^2,na.rm=T))
151
152
```

```
> df_test$price_hat1 = predict(mod1,newdata = df_test)
> df_test$price_hat2 = predict(mod2,newdata = df_test)
> df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
> View(df_test)
> View(df_test)
> View(df_test)
> sqrt(mean((df_test$price_hat1-df_test$price)^2))
[1] NA
> sqrt(mean((df_test$price_hat1-df_test$price)^2,na.rm=T))
[1] 119.0295
>
```

Remove Leading/Trailing Whitespace

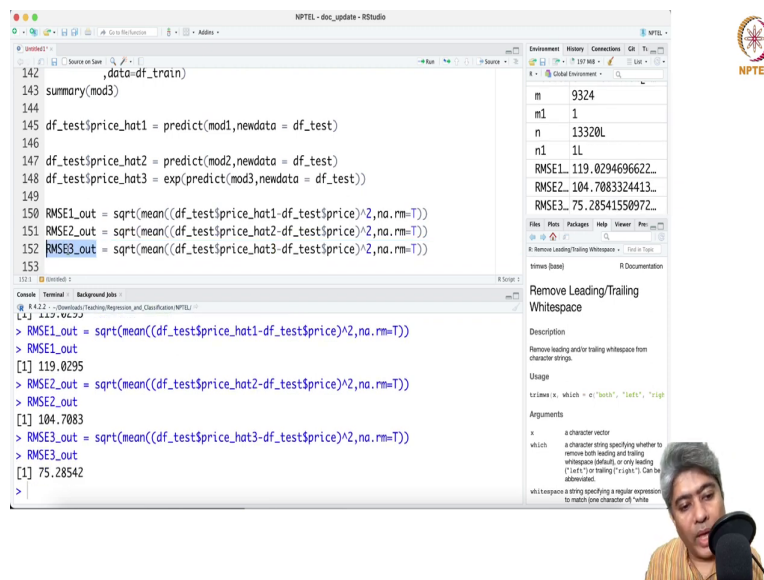
Description
Remove leading and/or trailing whitespace from character strings.

Usage
`trimws(x, which = c("both", "left", "right"))`

Arguments
`x` a character vector
`which` a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
`whitespace` a string specifying a regular expression to match one character of "white"

We will take the df test price 1 hat minus df test dollar price square it mean it take a mean of that take square root of that. So, ok I have to take na dot rm equal true some places probably it did not able to. So, this is my RMSE1.

(Refer Slide Time: 55:22)



The screenshot shows an RStudio window with the following code in the editor:

```
142 ,data=df_train)
143 summary(mod3)
144
145 df_test$price_hat1 = predict(mod1,newdata = df_test)
146
147 df_test$price_hat2 = predict(mod2,newdata = df_test)
148 df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
149
150 RMSE1_out = sqrt(mean((df_test$price_hat1-df_test$price)^2,na.rm=T))
151 RMSE2_out = sqrt(mean((df_test$price_hat2-df_test$price)^2,na.rm=T))
152 RMSE3_out = sqrt(mean((df_test$price_hat3-df_test$price)^2,na.rm=T))
153
```

The console output shows the results of the RMSE calculations:

```
> RMSE1_out = sqrt(mean((df_test$price_hat1-df_test$price)^2,na.rm=T))
> RMSE1_out
[1] 119.0295
> RMSE2_out = sqrt(mean((df_test$price_hat2-df_test$price)^2,na.rm=T))
> RMSE2_out
[1] 104.7083
> RMSE3_out = sqrt(mean((df_test$price_hat3-df_test$price)^2,na.rm=T))
> RMSE3_out
[1] 75.28542
>
```

The Environment pane on the right shows the following variables:

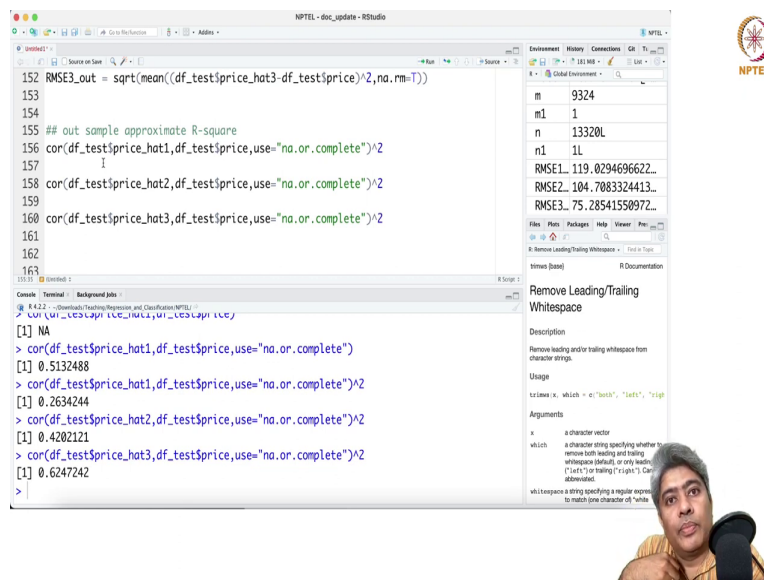
| Variable | Value |
|----------|-------------------|
| m | 9324 |
| m1 | 1 |
| n | 13320L |
| n1 | 1L |
| RMSE1_ | 119.0294696622... |
| RMSE2_ | 104.7083324413... |
| RMSE3_ | 75.28541550972... |

A video inset in the bottom right corner shows a man speaking into a microphone.

Out of the sample RMSE RMSE1 this is out sample remember that this is my out sample RMSE and second is from the second model. This is little less and the third one is from the third model ok and this is very less. So, third model is not only doing well it is doing pretty well to great extent.

So, log; that means, basically the total square feet bathroom BHK and price they all have a log linear relationship and that is why even in the out of the sample they are doing far superior than the just sample linear regression ok. There and let us we can also compute the correlation between the two.

(Refer Slide Time: 56:35)



The screenshot shows an RStudio session. The script editor contains the following code:

```
152 RMSE3_out = sqrt(mean((df_test$price_hat3-df_test$price)^2,na.rm=T))
153
154
155 ## out sample approximate R-square
156 cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
157
158 cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
159
160 cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
161
162
163
```

The console shows the output of the correlation calculations:

```
[1] NA
> cor(df_test$price_hat1,df_test$price,use="na.or.complete")
[1] 0.5132488
> cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2
[1] 0.2634244
> cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2
[1] 0.4202121
> cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
[1] 0.6247242
>
```

The Environment window shows the following variables:

| Variable | Value |
|----------|-------------------|
| m | 9324 |
| m1 | 1 |
| n | 13320L |
| n1 | 1L |
| RMSE1_ | 119.0294696622... |
| RMSE2_ | 104.7083324413... |
| RMSE3_ | 75.28541550972... |

The Help window shows the documentation for the `strstrip` function:

Remove Leading/Trailing Whitespace

Description: Remove leading and/or trailing whitespace from character strings.

Usage: `strstrip(x, which = c("both", "left", "right"))`

Arguments: `x` is a character vector; `which` is a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right") whitespace.

We can just take cor complete yeah and if you just take square. So, this is really less and then if you have just take 2 42 percent and then 3 this is approximate r square is 62 percent out of the sample approximate R square these are like out of the sample out sample approximate R square s q u a r e.

So, it shows that third model is has pretty good predictive power in even out of the sample it can predict the price with 62 percent variability can be explained, which is very good some out of over fitting is happening, but that is can be handled. So, with that I will stop here I hope you enjoyed this hands-on session. So, see you in the next video.

Thank you.

