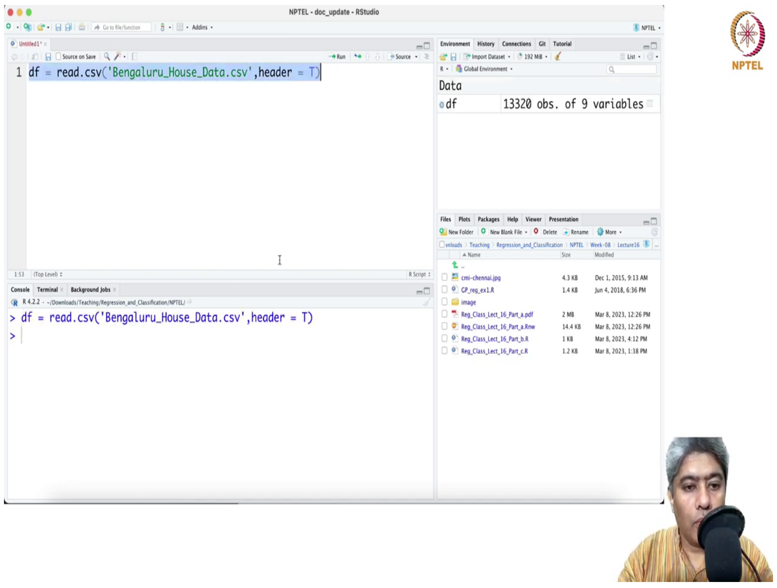
Predictive Analytics - Regression and Classification

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)



```
df = read.csv('Bengaluru_House_Data.csv',header = T)
```

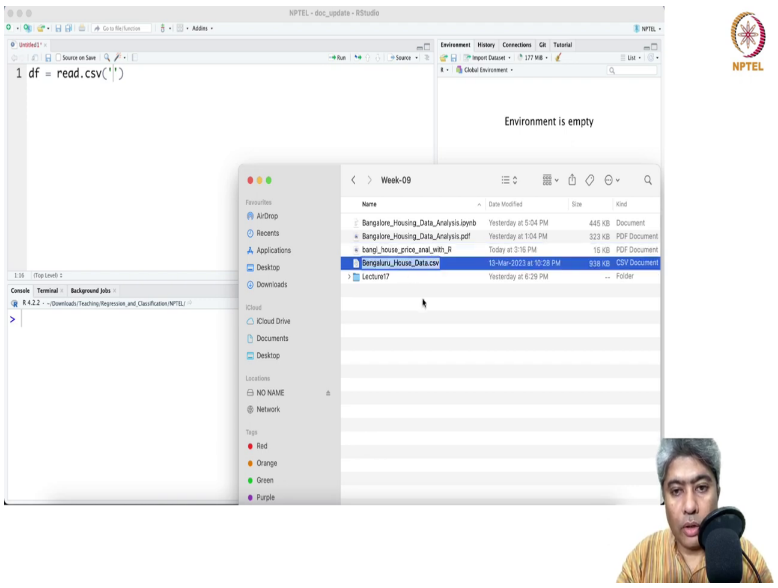
```
> df = read.csv('Bengaluru_House_Data.csv',header = T)
>
```

```
Data
df        13320 obs. of 9 variables
```

Name	Size	Modified
cmi-chennai.jpg	4.3 KB	Dec 1, 2015, 9:13 AM
GP_reg_ex1.R	1.4 KB	Jun 4, 2018, 6:36 PM
image		
Reg_Class_Lect_16_Part_a.pdf	2 MB	Mar 8, 2023, 12:26 PM
Reg_Class_Lect_16_Part_a.Rnw	14.4 KB	Mar 8, 2023, 12:26 PM
Reg_Class_Lect_16_Part_b.R	1 KB	Mar 8, 2023, 4:12 PM
Reg_Class_Lect_16_Part_c.R	1.2 KB	Mar 8, 2023, 1:38 PM

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

(Refer Slide Time: 01:02)



```
df = read.csv('')
```

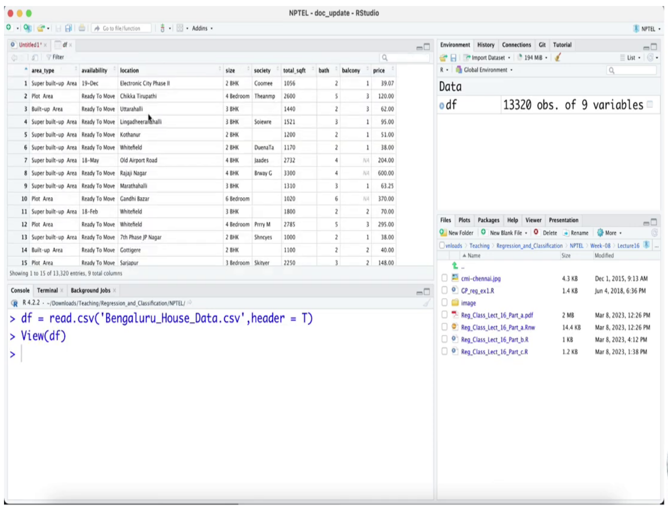
Environment is empty

Week-09

Name	Date Modified	Size	Kind
Bangalore_Housing_Data_Analysis.ipynb	Yesterday at 5:04 PM	445 KB	Document
Bangalore_Housing_Data_Analysis.pdf	Yesterday at 1:04 PM	323 KB	PDF Document
bangl_house_price_anal_with_R	Today at 3:16 PM	15 KB	PDF Document
Bengaluru_House_Data.csv	13-Mar-2023 at 10:28 PM	938 KB	CSV Document
Lecture17	Yesterday at 6:29 PM	--	Folder

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)



	area_type	availability	location	size	society	total_sqft	bath	balcony	price
1	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2	1	39.07
2	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5	3	120.00
3	Built-up Area	Ready To Move	Uttarahalli	3 BHK		1440	2	3	62.00
4	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3	1	95.00
5	Super built-up Area	Ready To Move	Kothanur	2 BHK		1200	2	1	51.00
6	Super built-up Area	Ready To Move	Whitefield	2 BHK	DuenaTa	1170	2	1	38.00
7	Super built-up Area	18-May	Old Airport Road	4 BHK	Jaades	2732	4	NA	204.00
8	Super built-up Area	Ready To Move	Rajaji Nagar	4 BHK	Brway G	3300	4	NA	600.00
9	Super built-up Area	Ready To Move	Marathahalli	3 BHK		1310	3	1	63.25
10	Plot Area	Ready To Move	Gandhi Bazar	6 Bedroom		1020	6	NA	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	Prrry M	2785	5	3	295.00
13	Super built-up Area	Ready To Move	7th Phase JP Nagar	2 BHK	Shncyes	1000	2	1	38.00
14	Built-up Area	Ready To Move	Gottigere	2 BHK		1100	2	2	40.00
15	Plot Area	Ready To Move	Sarjapur	3 Bedroom	Skityer	2250	3	2	148.00

Showing 1 to 15 of 13,320 entries, 9 total columns

```
R 4.2.2 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
> df = read.csv('Bengaluru_House_Data.csv',header = T)
> View(df)
>
```

Data

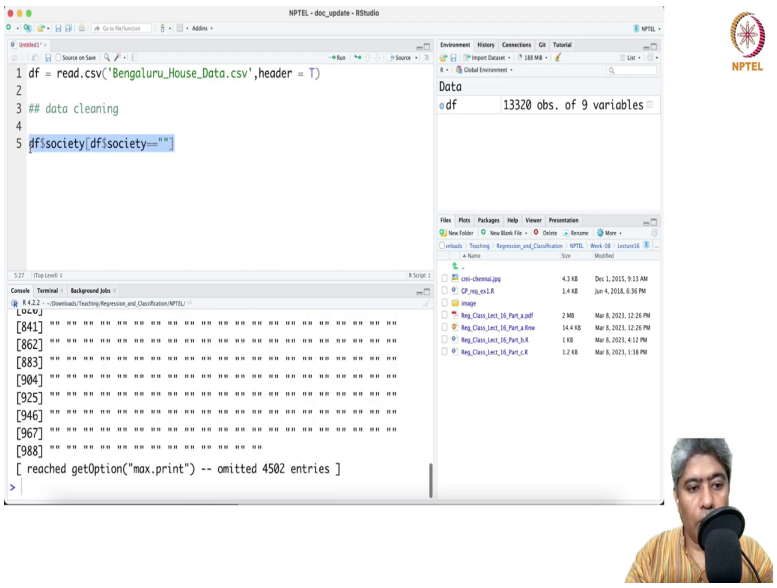
df	13320 obs. of 9 variables

vnloads > Teaching > Regression_and_Classification > NPTEL > Week-08 > Lecture16

Name	Size	Modified
..		
cmi-chennai.jpg	4.3 KB	Dec 1, 2015, 9:13 AM
GP_reg_ex1.R	1.4 KB	Jun 4, 2018, 6:36 PM
image		
Reg_Class_Lect_16_Part_a.pdf	2 MB	Mar 8, 2023, 12:26 PM
Reg_Class_Lect_16_Part_a.Rnw	14.4 KB	Mar 8, 2023, 12:26 PM
Reg_Class_Lect_16_Part_b.R	1 KB	Mar 8, 2023, 4:12 PM
Reg_Class_Lect_16_Part_c.R	1.2 KB	Mar 8, 2023, 1:38 PM

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)



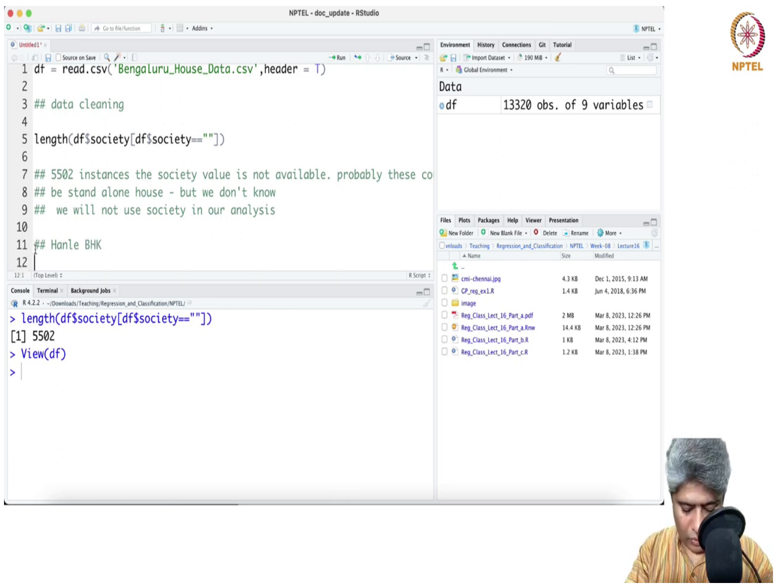
```
df = read.csv('Bengaluru_House_Data.csv',header = T)

## data cleaning

df$society[df$society==""]
```

```
Data
df            13320 obs. of 9 variables
```

```
[841] "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
[862] "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
[883] "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
[904] "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
[925] "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
[946] "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
[967] "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
[988] "" "" "" "" "" "" "" "" "" "" "" "" ""
[ reached getOption("max.print") -- omitted 4502 entries ]
>
```

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)



```
df = read.csv('Bengaluru_House_Data.csv',header = T)

## data cleaning

length(df$society[df$society==""])

## 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

```

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

Data

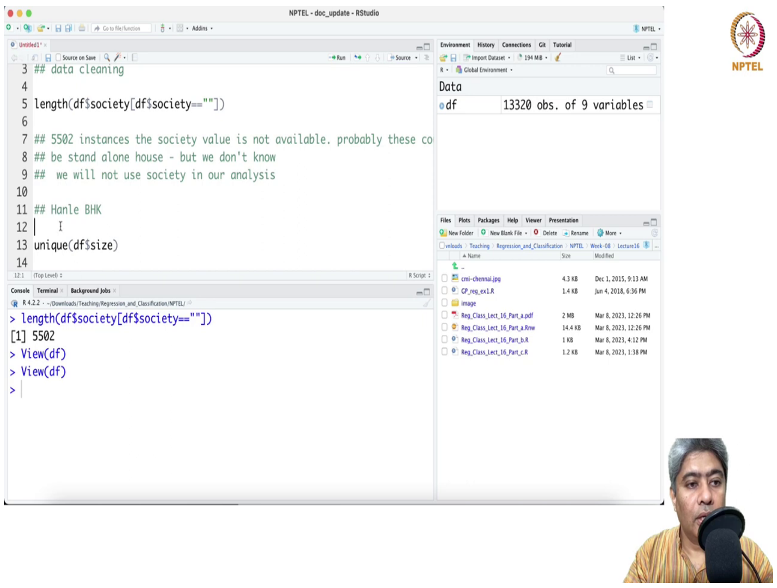
df	13320 obs. of 9 variables

Name	Size	Modified
cmi-chennai.jpg	4.3 KB	Dec 1, 2015, 9:13 AM
GP_reg_ex1.R	1.4 KB	Jun 4, 2018, 6:36 PM
image		
Reg_Class_Lect_16_Part_a.pdf	2 MB	Mar 8, 2023, 12:26 PM
Reg_Class_Lect_16_Part_a.Rnw	14.4 KB	Mar 8, 2023, 12:26 PM
Reg_Class_Lect_16_Part_b.R	1 KB	Mar 8, 2023, 4:12 PM
Reg_Class_Lect_16_Part_c.R	1.2 KB	Mar 8, 2023, 1:38 PM

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)



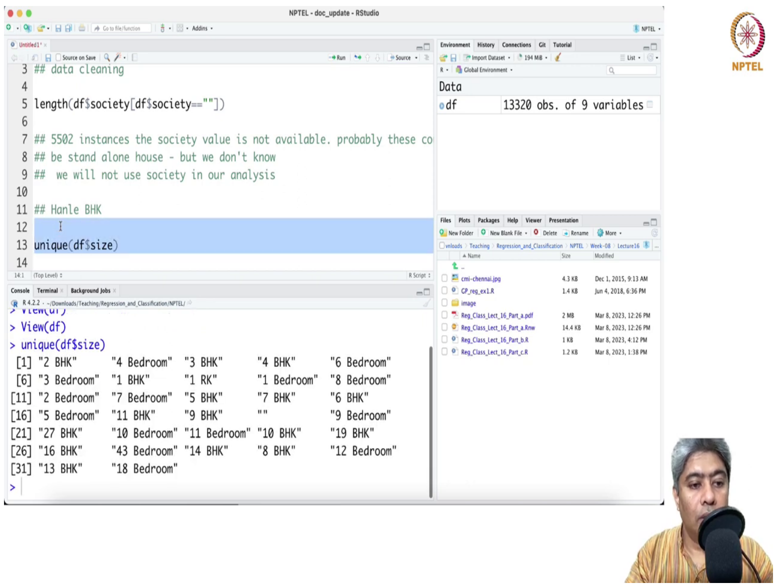
```
## data cleaning

length(df$society[df$society==""])

## 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)
```

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

```
df    13320 obs. of 9 variables
```

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)



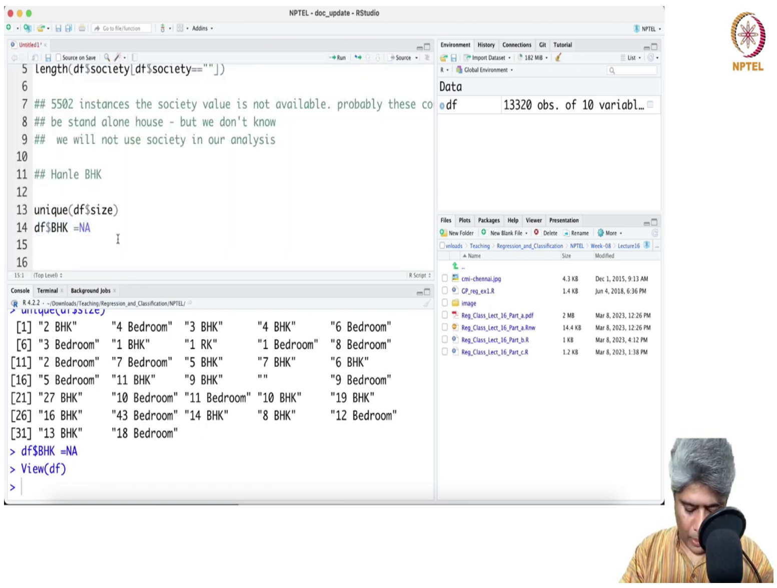
```
## data cleaning

length(df$society[df$society==""])

## 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)
```

```
> 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] "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"
>
```

Data
df 13320 obs. of 9 variables

So, these are the BHK things.

(Refer Slide Time: 05:14)

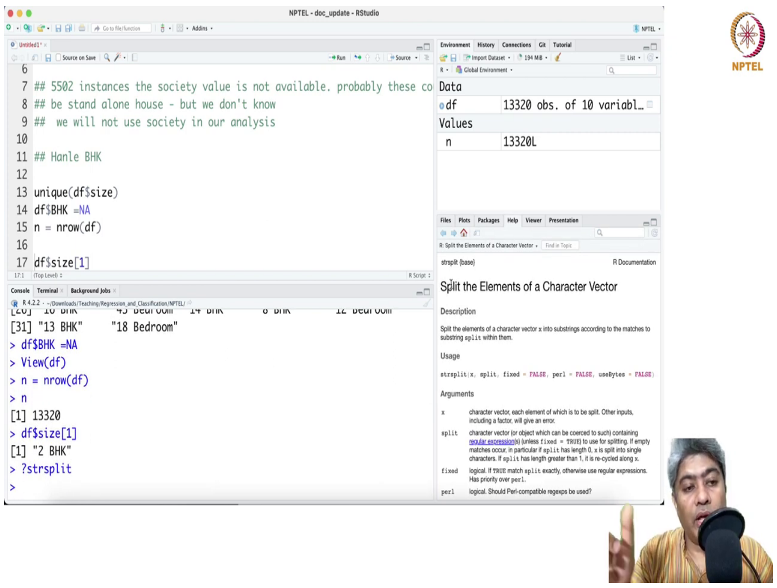

```
length(df$society[df$society==""])

## 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
```

```
Data
df    13320 obs. of 10 variabl…
```

```
> unique(df$size)
 [1] "2 BHK"      "4 Bedroom"  "3 BHK"      "4 BHK"      "6 Bedroom"
 [6] "3 Bedroom"  "1 BHK"      "1 RK"       "1 Bedroom"  "8 Bedroom"
[11] "2 Bedroom"  "7 Bedroom"  "5 BHK"      "7 BHK"      "6 BHK"
[16] "5 Bedroom"  "11 BHK"     "9 BHK"      ""           "9 Bedroom"
[21] "27 BHK"     "10 Bedroom" "11 Bedroom" "10 BHK"     "19 BHK"
[26] "16 BHK"     "43 Bedroom" "14 BHK"     "8 BHK"      "12 Bedroom"
[31] "13 BHK"     "18 Bedroom"
> df$BHK =NA
> View(df)
>
```

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)



```
## 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]
```

```
Data
df        13320 obs. of 10 variabl…
Values
n         13320L
```

```
[31] "13 BHK"     "18 Bedroom"
> df$BHK =NA
> View(df)
> n = nrow(df)
> n
[1] 13320
> df$size[1]
[1] "2 BHK"
> ?strsplit
>
```

strsplit {base}   R Documentation

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

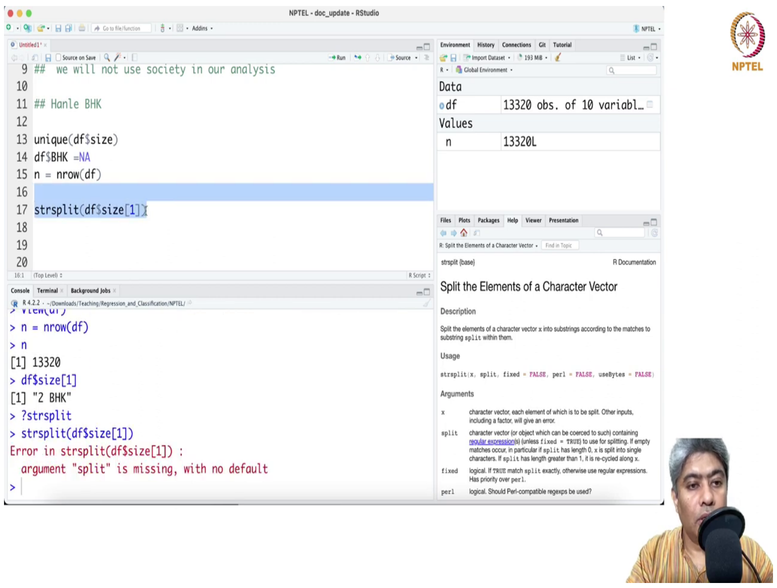
```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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?

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)



```r
##  we will not use society in our analysis

## Hanle BHK

unique(df$size)
df$BHK =NA
n = nrow(df)

strsplit(df$size[1])
```

```
> 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
>
```

```
Data
df      13320 obs. of 10 variabl…
Values
n       13320L
```

strsplit {base} R Documentation

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

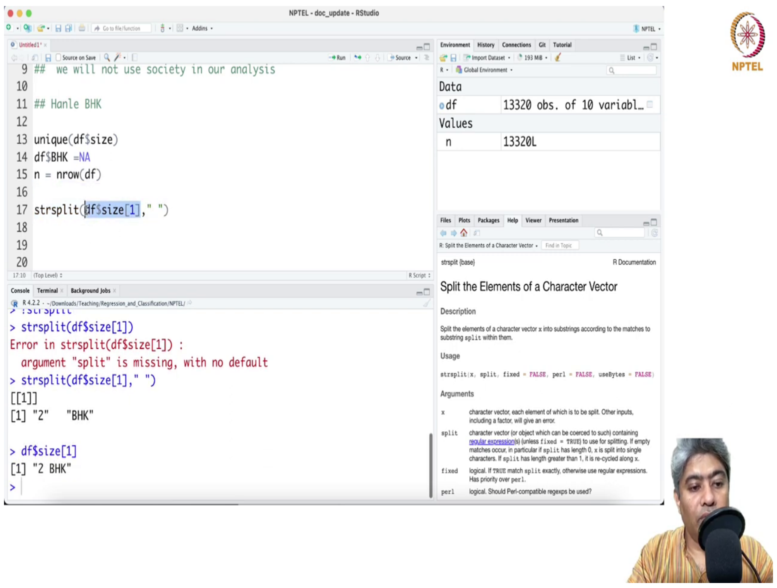
```r
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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?

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)


NPTEL - doc_update - RStudio
Environment History Connections Git Tutorial
Files Plots Packages Help Viewer Presentation
NPTEL

```
## we will not use society in our analysis

## Hanle BHK

unique(df$size)
df$BHK =NA
n = nrow(df)

strsplit(df$size[1]," ")
```

```
> 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"

> df$size[1]
[1] "2 BHK"
>
```

```
Data
df        13320 obs. of 10 variabl…
Values
n         13320L
```

strsplit {base}   R Documentation

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

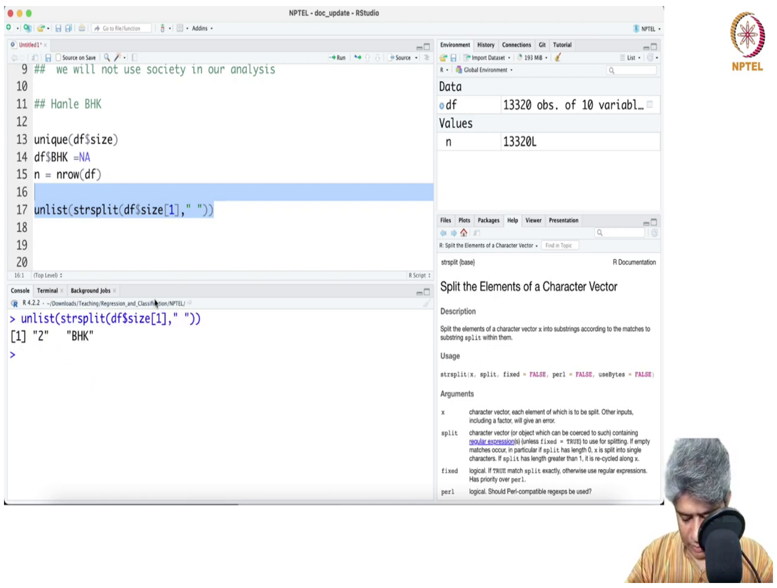
`strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)`

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 expression(s) (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?

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)



```
## we will not use society in our analysis

## Hanle BHK

unique(df$size)
df$BHK =NA
n = nrow(df)

unlist(strsplit(df$size[1]," "))
```

```
> unlist(strsplit(df$size[1]," "))
[1] "2"   "BHK"
>
```

Data
df 13320 obs. of 10 variabl…
Values
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

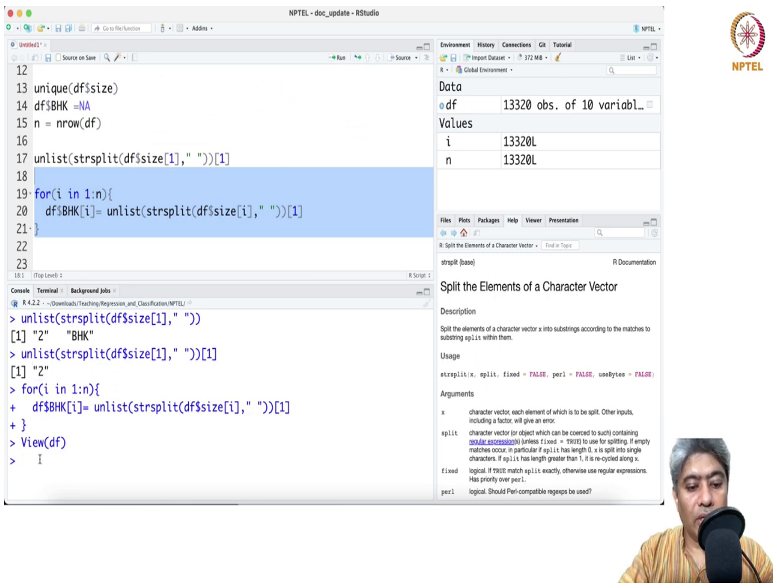
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)

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 expression(s) (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?

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

(Refer Slide Time: 07:10)



```
unique(df$size)
df$BHK =NA
n = nrow(df)

unlist(strsplit(df$size[1]," "))[1]

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

```
> 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)
>
```

Data

df	13320 obs. of 10 variabl…

Values

i	13320L
n	13320L

strsplit {base} R Documentation

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

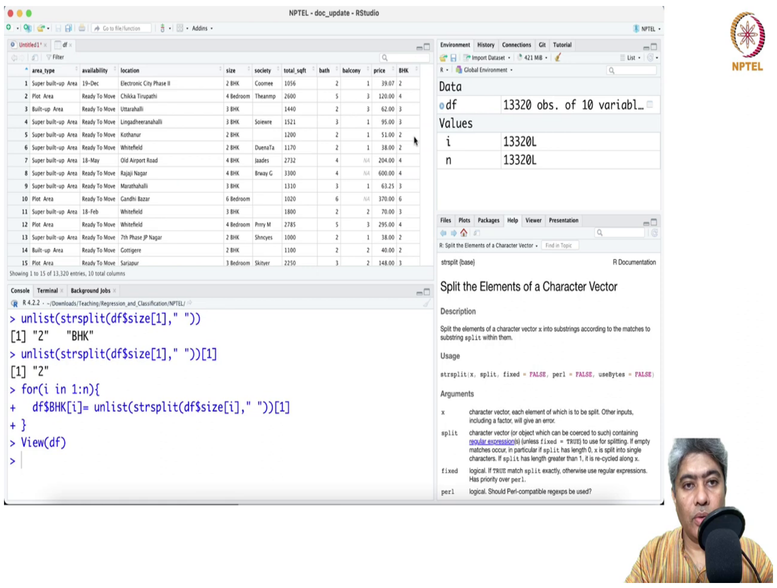
```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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?

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)


NPTEL - doc_update - RStudio

	area_type	availability	location	size	society	total_sqft	bath	balcony	price	BHK
1	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2	1	39.07	2
2	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5	3	120.00	4
3	Built-up Area	Ready To Move	Uttarahalli	3 BHK		1440	2	3	62.00	3
4	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3	1	95.00	3
5	Super built-up Area	Ready To Move	Kothanur	2 BHK		1200	2	1	51.00	2
6	Super built-up Area	Ready To Move	Whitefield	2 BHK	DuenaTa	1170	2	1	38.00	2
7	Super built-up Area	18-May	Old Airport Road	4 BHK	Jaades	2732	4	NA	204.00	4
8	Super built-up Area	Ready To Move	Rajaji Nagar	4 BHK	Brway G	3300	4	NA	600.00	4
9	Super built-up Area	Ready To Move	Marathahalli	3 BHK		1310	3	1	63.25	3
10	Plot Area	Ready To Move	Gandhi Bazar	6 Bedroom		1020	6	NA	370.00	6
11	Super built-up Area	18-Feb	Whitefield	3 BHK		1800	2	2	70.00	3
12	Plot Area	Ready To Move	Whitefield	4 Bedroom	Prrry M	2785	5	3	295.00	4
13	Super built-up Area	Ready To Move	7th Phase JP Nagar	2 BHK	Shncyes	1000	2	1	38.00	2
14	Built-up Area	Ready To Move	Gottigere	2 BHK		1100	2	2	40.00	2
15	Plot Area	Ready To Move	Sarjapur	3 Bedroom	Skityer	2250	3	2	148.00	3

Showing 1 to 15 of 13,320 entries, 10 total columns

```
R 4.2.2 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
> 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)
>
```

Data

df	13320 obs. of 10 variabl…

Values

i	13320L
n	13320L

strsplit {base} R Documentation

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

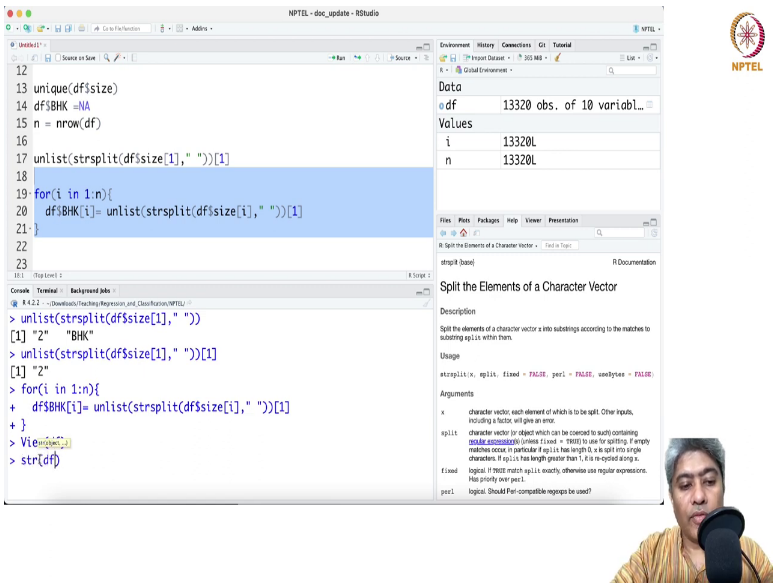
```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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?

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)



```
unique(df$size)
df$BHK =NA
n = nrow(df)

unlist(strsplit(df$size[1]," "))[1]

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

Data	
df	13320 obs. of 10 variabl…
Values	
i	13320L
n	13320L

```
> 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]
+ }
> Vie
> str(df)
```

strsplit {base} R Documentation

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

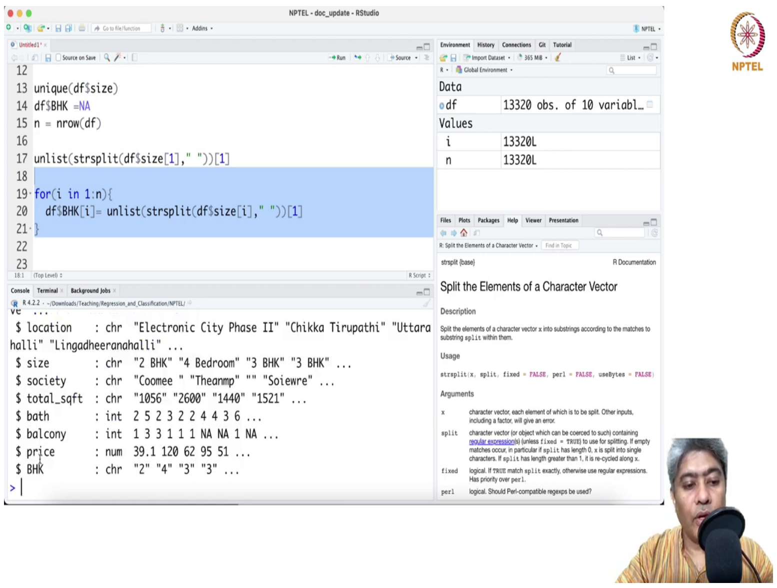
```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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?

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

(Refer Slide Time: 08:13)



```r
unique(df$size)
df$BHK =NA
n = nrow(df)

unlist(strsplit(df$size[1]," "))[1]

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

```
 $ location    : chr  "Electronic City Phase II" "Chikka Tirupathi" "Uttara
halli" "Lingadheeranahalli" ...
 $ size        : chr  "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
 $ society     : chr  "Coomee " "Theanmp" "" "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" ...
>
```

Data	
df	13320 obs. of 10 variabl…
Values	
i	13320L
n	13320L

strsplit {base} R Documentation

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

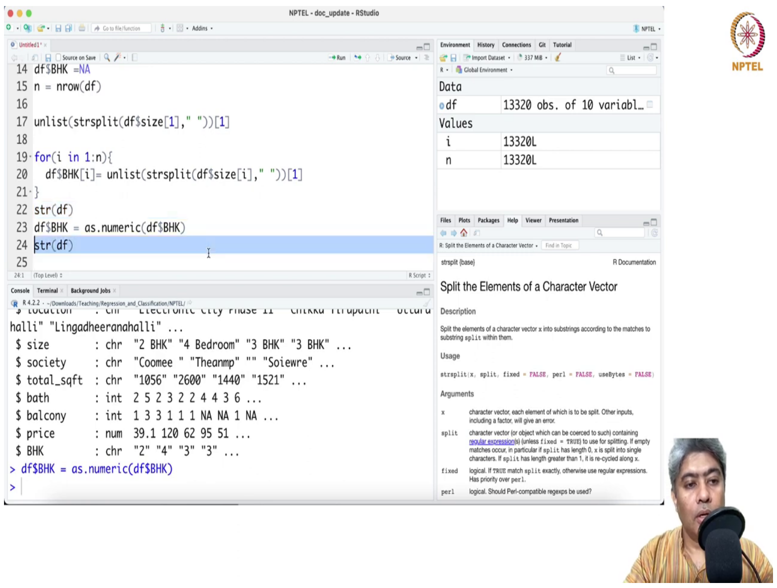
```r
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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?

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

(Refer Slide Time: 08:22)



```r
df$BHK =NA
n = nrow(df)

unlist(strsplit(df$size[1]," "))[1]

for(i in 1:n){
  df$BHK[i]= unlist(strsplit(df$size[i]," "))[1]
}
str(df)
df$BHK = as.numeric(df$BHK)
str(df)
```

```
halli" "Lingadheeranahalli" ...
 $ size       : chr  "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
 $ society    : chr  "Coomee " "Theanmp" "" "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" ...
> df$BHK = as.numeric(df$BHK)
>
```

Data

df	13320 obs. of 10 variabl…

Values

i	13320L
n	13320L

strsplit {base} R Documentation

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

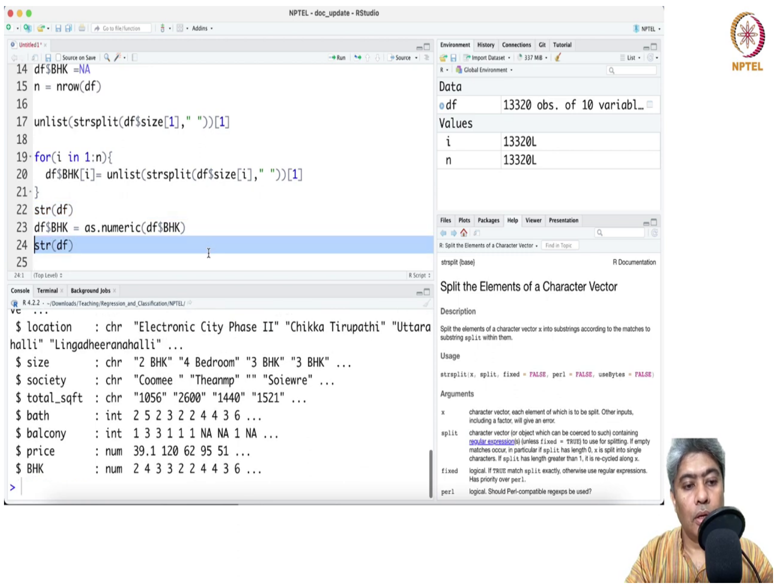
```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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?

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)



```r
df$BHK =NA
n = nrow(df)

unlist(strsplit(df$size[1]," "))[1]

for(i in 1:n){
  df$BHK[i]= unlist(strsplit(df$size[i]," "))[1]
}
str(df)
df$BHK = as.numeric(df$BHK)
str(df)
```

```
ve  ...
 $ location    : chr  "Electronic City Phase II" "Chikka Tirupathi" "Uttara
halli" "Lingadheeranahalli" ...
 $ size        : chr  "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
 $ society     : chr  "Coomee " "Theanmp" "" "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         : num  2 4 3 3 2 2 4 4 3 6 ...
>
```

```
Data
df      13320 obs. of 10 variabl…
Values
i       13320L
n       13320L
```

strsplit {base}   R Documentation

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

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

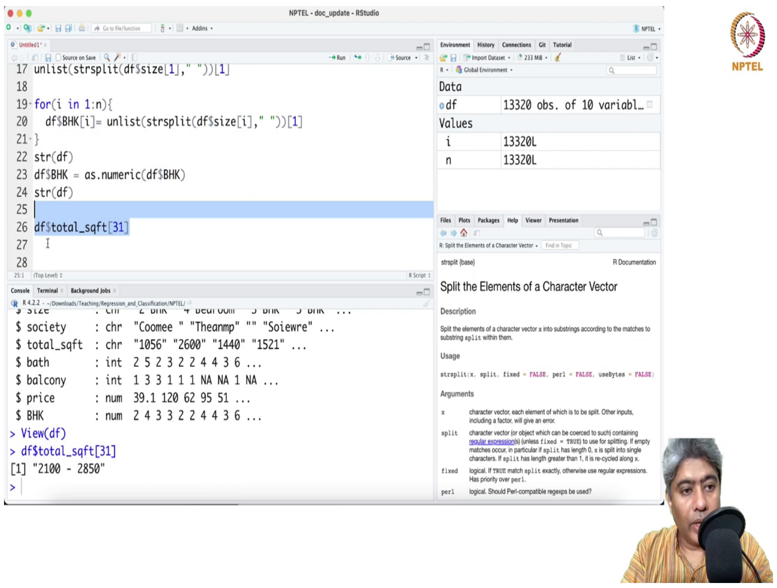
Arguments

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 expression(s) (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?

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)



```
unlist(strsplit(df$size[1]," "))[1]

for(i in 1:n){
  df$BHK[i]= unlist(strsplit(df$size[i]," "))[1]
}
str(df)
df$BHK = as.numeric(df$BHK)
str(df)

df$total_sqft[31]


```

```
 $ society    : chr  "Coomee " "Theanmp" "" "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        : num  2 4 3 3 2 2 4 4 3 6 ...
> View(df)
> df$total_sqft[31]
[1] "2100 - 2850"
>
```

Data	
df	13320 obs. of 10 variabl…
Values	
i	13320L
n	13320L

strsplit {base} R Documentation

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

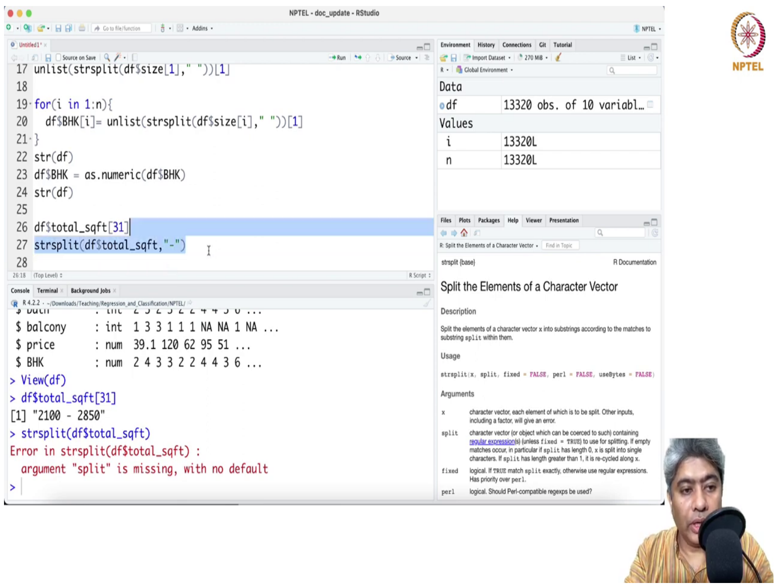
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)

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 expression(s) (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?

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)



```r
unlist(strsplit(df$size[1]," "))[1]

for(i in 1:n){
  df$BHK[i]= unlist(strsplit(df$size[i]," "))[1]
}
str(df)
df$BHK = as.numeric(df$BHK)
str(df)

df$total_sqft[31]
strsplit(df$total_sqft,"-")
```

```
 $ 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 ...
> 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
>
```

Data	
df	13320 obs. of 10 variabl…
Values	
i	13320L
n	13320L

strsplit {base} R Documentation

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

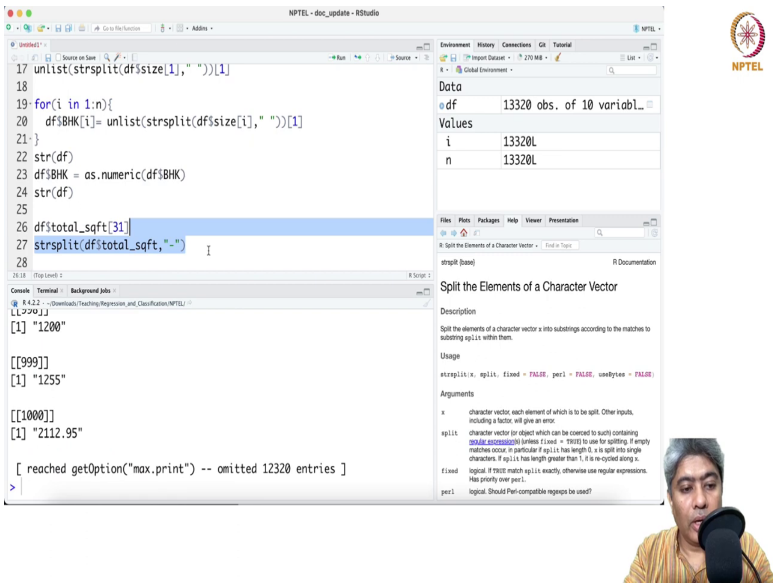
```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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?

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)



```r
unlist(strsplit(df$size[1]," "))[1]

for(i in 1:n){
  df$BHK[i]= unlist(strsplit(df$size[i]," "))[1]
}
str(df)
df$BHK = as.numeric(df$BHK)
str(df)

df$total_sqft[31]
strsplit(df$total_sqft,"-")
```

```
[1] "1200"

[[999]]
[1] "1255"

[[1000]]
[1] "2112.95"

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

Data	
df	13320 obs. of 10 variabl…
Values	
i	13320L
n	13320L

strsplit {base} R Documentation

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.

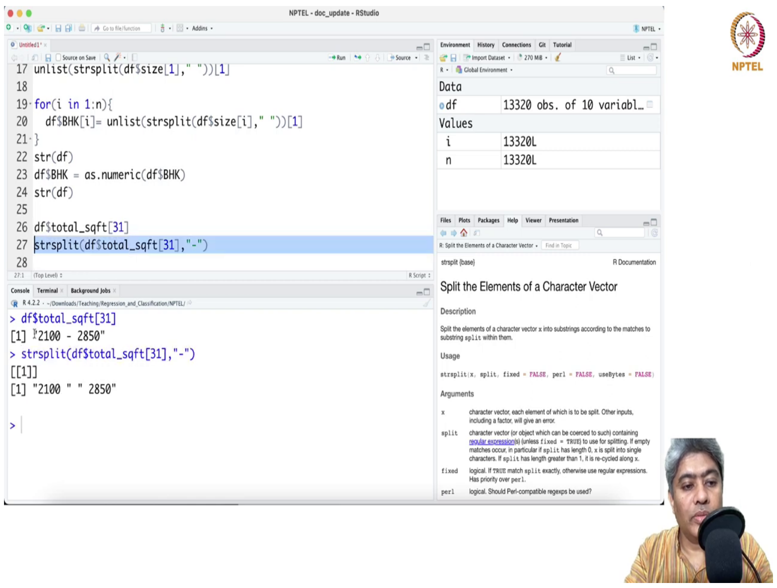
Usage

`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 expression(s) (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?

(Refer Slide Time: 10:38)



```
unlist(strsplit(df$size[1]," "))[1]

for(i in 1:n){
  df$BHK[i]= unlist(strsplit(df$size[i]," "))[1]
}
str(df)
df$BHK = as.numeric(df$BHK)
str(df)

df$total_sqft[31]
strsplit(df$total_sqft[31],"-")
```

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

>
```

```
Data
df      13320 obs. of 10 variabl…
Values
i       13320L
n       13320L
```

strsplit {base}   R Documentation

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

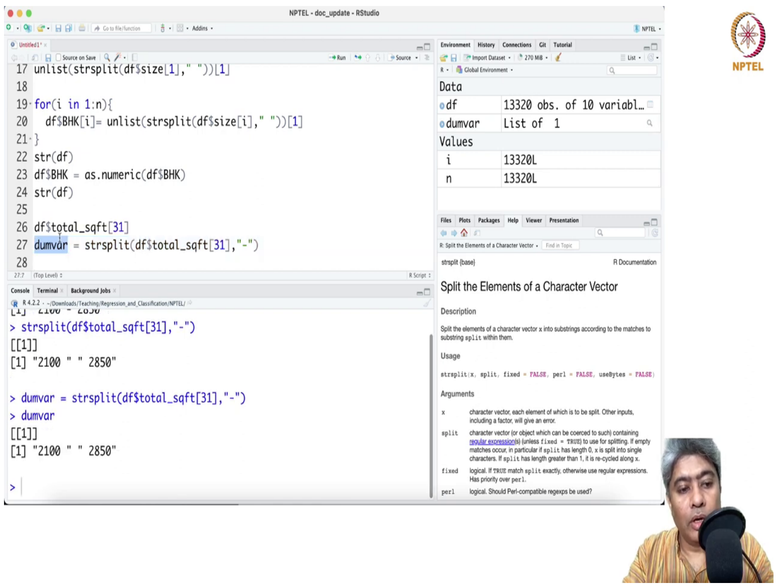
```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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)



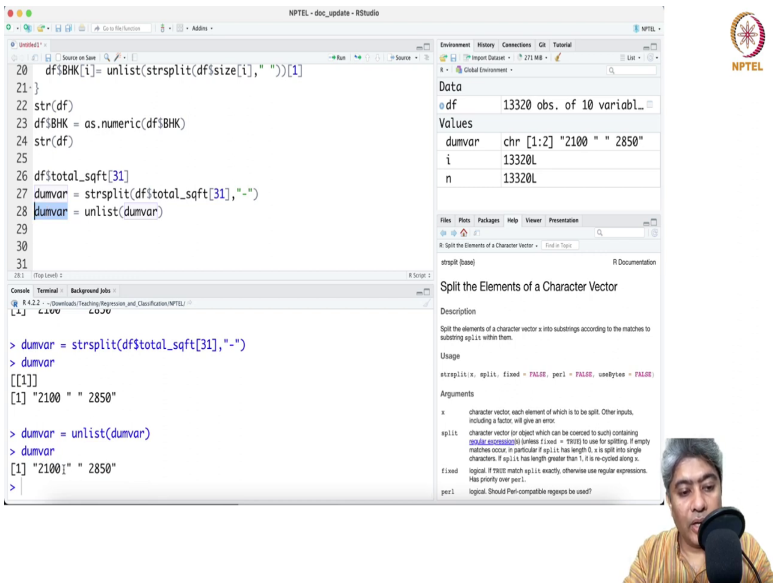
```
unlist(strsplit(df$size[1]," "))[1]

for(i in 1:n){
  df$BHK[i]= unlist(strsplit(df$size[i]," "))[1]
}
str(df)
df$BHK = as.numeric(df$BHK)
str(df)

df$total_sqft[31]
dumvar = strsplit(df$total_sqft[31],"-")
```

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

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

>
```

NPTEL - doc_update - RStudio
Data
df 13320 obs. of 10 variabl…
dumvar List of 1
Values
i 13320L
n 13320L
Split the Elements of a Character Vector
NPTEL

So, let us put it in some dummy variable.

(Refer Slide Time: 11:17)



```
  df$BHK[i]= unlist(strsplit(df$size[i]," "))[1]
}
str(df)
df$BHK = as.numeric(df$BHK)
str(df)

df$total_sqft[31]
dumvar = strsplit(df$total_sqft[31],"-")
dumvar = unlist(dumvar)
```

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

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

```
Data
df        13320 obs. of 10 variabl…
Values
dumvar    chr [1:2] "2100 " " 2850"
i         13320L
n         13320L
```

strsplit {base} R Documentation

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

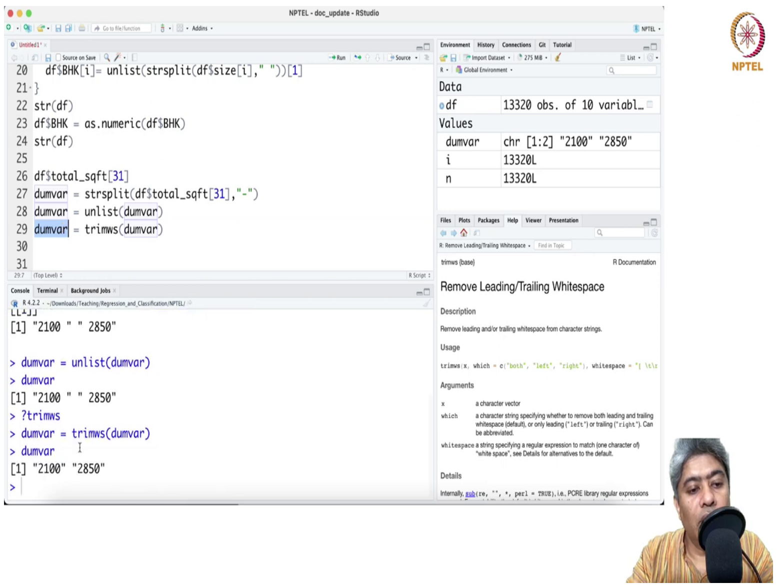
```
strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)
```

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 expression(s) (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?

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)



```
  df$BHK[i]= unlist(strsplit(df$size[i]," "))[1]
}
str(df)
df$BHK = as.numeric(df$BHK)
str(df)

df$total_sqft[31]
dumvar = strsplit(df$total_sqft[31],"-")
dumvar = unlist(dumvar)
dumvar = trimws(dumvar)
```

```
[1] "2100 " " 2850"

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

```
Data
df        13320 obs. of 10 variabl…
Values
dumvar    chr [1:2] "2100" "2850"
i         13320L
n         13320L
```

trimws {base} R Documentation

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
```

Arguments

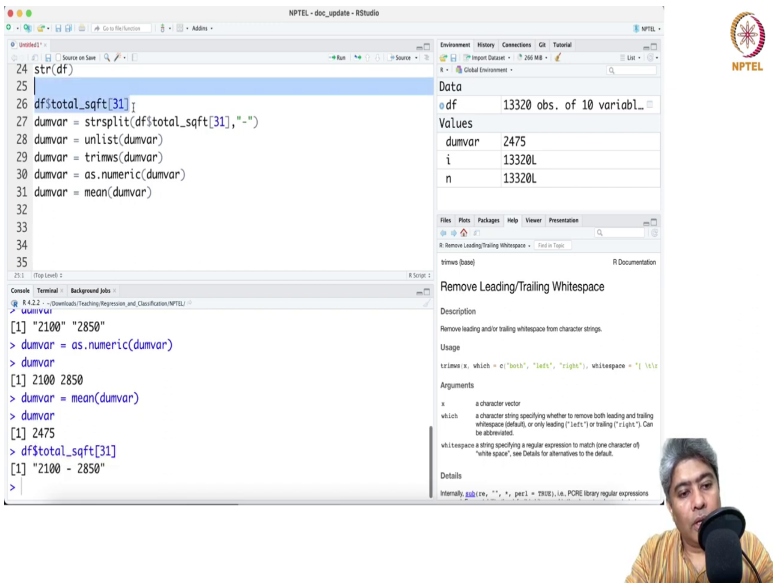
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

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)



```
str(df)

df$total_sqft[31]
dumvar = strsplit(df$total_sqft[31],"-")
dumvar = unlist(dumvar)
dumvar = trimws(dumvar)
dumvar = as.numeric(dumvar)
dumvar = mean(dumvar)
```

```
[1] "2100" "2850"
> dumvar = as.numeric(dumvar)
> dumvar
[1] 2100 2850
> dumvar = mean(dumvar)
> dumvar
[1] 2475
> df$total_sqft[31]
[1] "2100 - 2850"
>
```

```
Data
df        13320 obs. of 10 variabl…
Values
dumvar    2475
i         13320L
n         13320L
```

trimws {base} R Documentation

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
```

Arguments

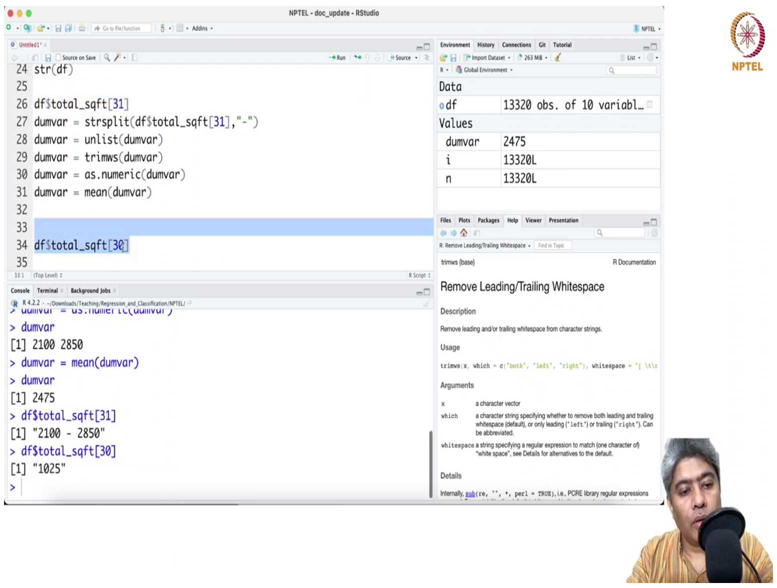
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

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)



```r
str(df)

df$total_sqft[31]
dumvar = strsplit(df$total_sqft[31],"-")
dumvar = unlist(dumvar)
dumvar = trimws(dumvar)
dumvar = as.numeric(dumvar)
dumvar = mean(dumvar)


df$total_sqft[30]

```

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

```
Data
df        13320 obs. of 10 variabl…
Values
dumvar    2475
i         13320L
n         13320L
```

trimws {base} R Documentation

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
```

Arguments

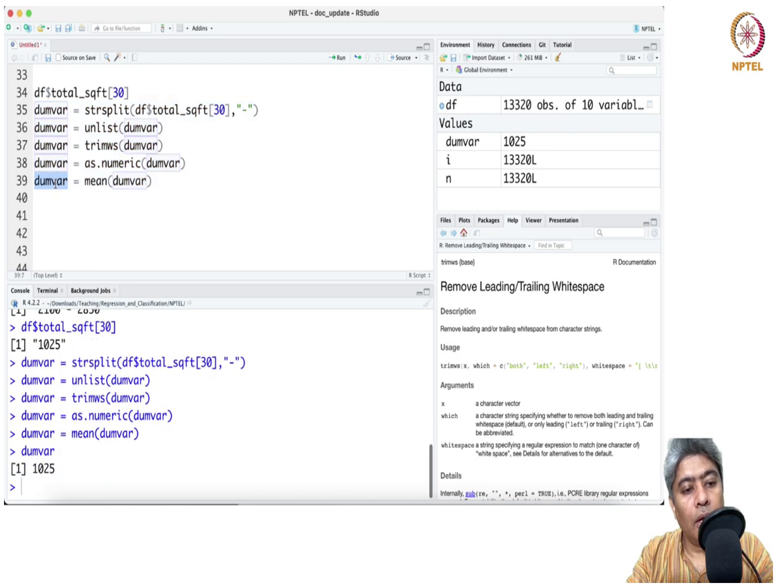
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, 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)



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

```
[1]  2100 - 2850
> 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
>
```

Data

```
df      13320 obs. of 10 variabl…
```

Values

```
dumvar  1025
i       13320L
n       13320L
```

trimws {base} R Documentation

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
```

Arguments

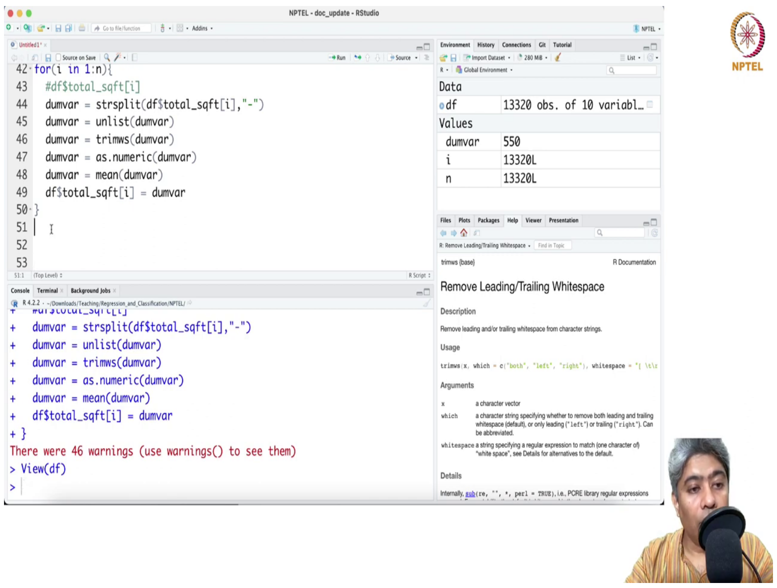
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

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)



```
for(i in 1:n){
  #df$total_sqft[i]
  dumvar = strsplit(df$total_sqft[i],"-")
  dumvar = unlist(dumvar)
  dumvar = trimws(dumvar)
  dumvar = as.numeric(dumvar)
  dumvar = mean(dumvar)
  df$total_sqft[i] = dumvar
}
```

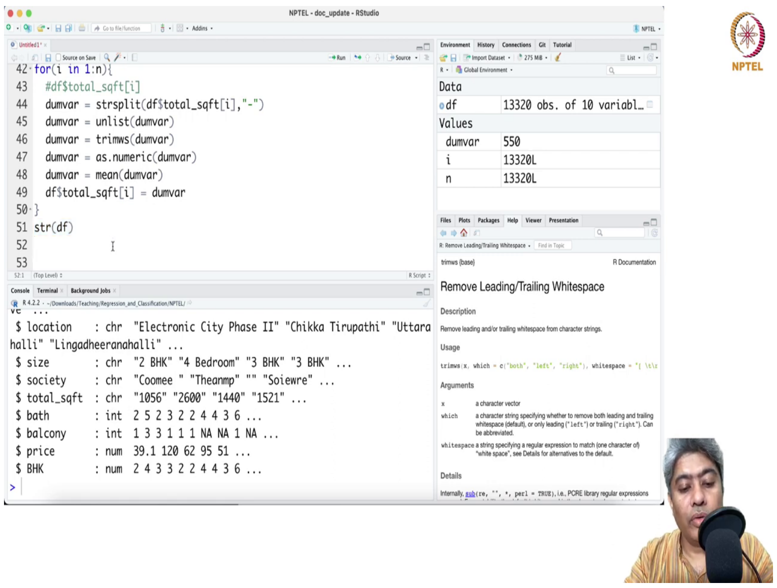
```
+   dumvar = strsplit(df$total_sqft[i],"-")
+   dumvar = unlist(dumvar)
+   dumvar = trimws(dumvar)
+   dumvar = as.numeric(dumvar)
+   dumvar = mean(dumvar)
+   df$total_sqft[i] = dumvar
+ }
There were 46 warnings (use warnings() to see them)
> View(df)
>
```

```
Data
df      13320 obs. of 10 variabl…
Values
dumvar  550
i       13320L
n       13320L
```

```
trimws {base}                                          R Documentation

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

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, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions
```

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)



```r
for(i in 1:n){
  #df$total_sqft[i]
  dumvar = strsplit(df$total_sqft[i],"-")
  dumvar = unlist(dumvar)
  dumvar = trimws(dumvar)
  dumvar = as.numeric(dumvar)
  dumvar = mean(dumvar)
  df$total_sqft[i] = dumvar
}
str(df)
```

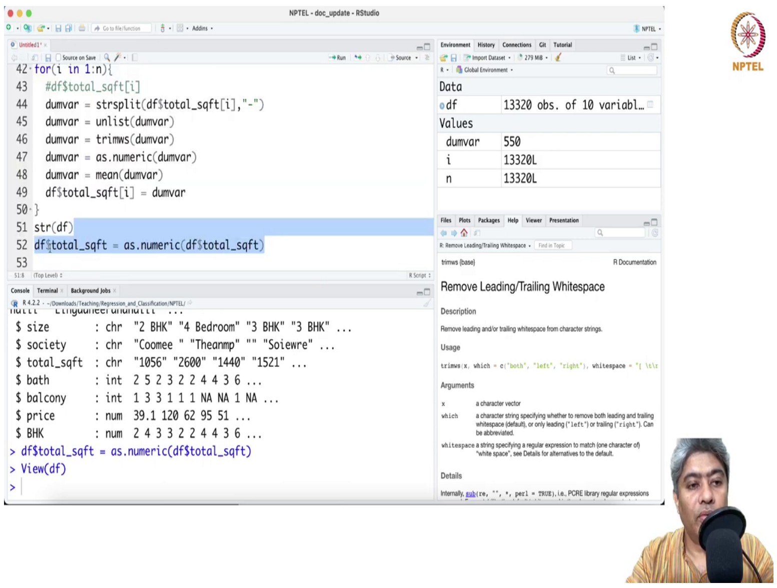
```
ve  ...
 $ location    : chr  "Electronic City Phase II" "Chikka Tirupathi" "Uttara
halli" "Lingadheeranahalli" ...
 $ size        : chr  "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
 $ society     : chr  "Coomee " "Theanmp" "" "Soiewre" ...
 $ total_sqft  : chr  "1056" "2600" "1440" "1521" ...
 $ bath        : 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         : num  2 4 3 3 2 2 4 4 3 6 ...
>
```

df	13320 obs. of 10 variabl…
dumvar	550
i	13320L
n	13320L

Remove Leading/Trailing Whitespace

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)



```
for(i in 1:n){
  #df$total_sqft[i]
  dumvar = strsplit(df$total_sqft[i],"-")
  dumvar = unlist(dumvar)
  dumvar = trimws(dumvar)
  dumvar = as.numeric(dumvar)
  dumvar = mean(dumvar)
  df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)
```

```
 $ size       : chr  "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
 $ society    : chr  "Coomee " "Theanmp" "" "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        : num  2 4 3 3 2 2 4 4 3 6 ...
> df$total_sqft = as.numeric(df$total_sqft)
> View(df)
>
```

Data

df	13320 obs. of 10 variabl…

Values

dumvar	550
i	13320L
n	13320L

trimws {base} R Documentation

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
```

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.

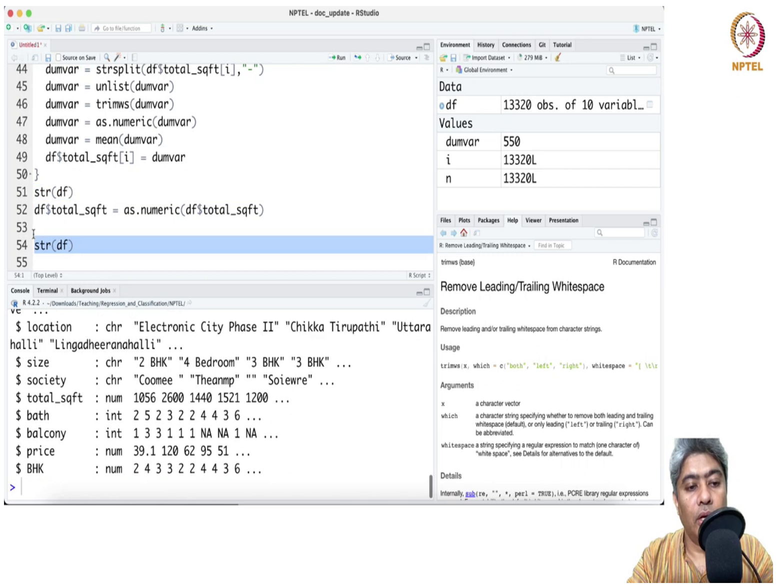
Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

NPTEL - doc_update - RStudio
Environment History Connections Git Tutorial
Files Plots Packages Help Viewer Presentation
Console Terminal Background Jobs

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)



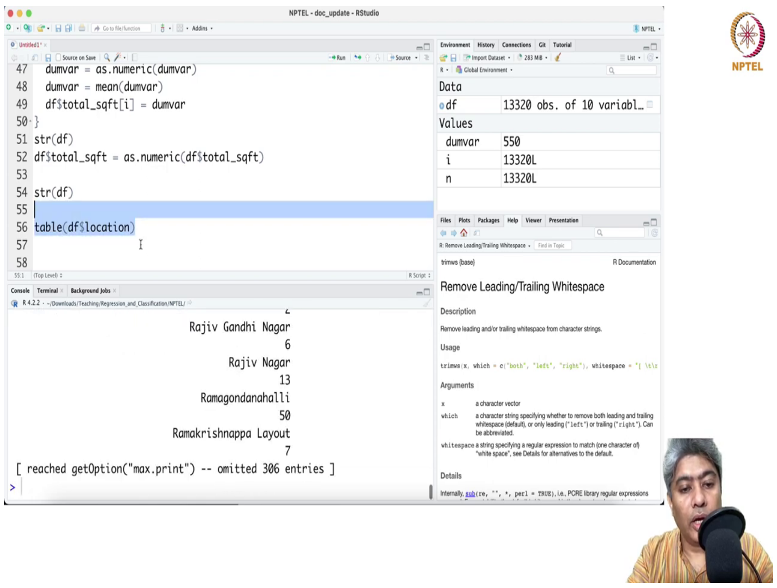
```
  dumvar = strsplit(df$total_sqft[i],"-")
  dumvar = unlist(dumvar)
  dumvar = trimws(dumvar)
  dumvar = as.numeric(dumvar)
  dumvar = mean(dumvar)
  df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)
```

```
 $ location   : chr  "Electronic City Phase II" "Chikka Tirupathi" "Uttara
halli" "Lingadheeranahalli" ...
 $ size       : chr  "2 BHK" "4 Bedroom" "3 BHK" "3 BHK" ...
 $ society    : chr  "Coomee " "Theanmp" "" "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 ...
>
```

```
Data
df        13320 obs. of 10 variabl…
Values
dumvar    550
i         13320L
n         13320L
```

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)



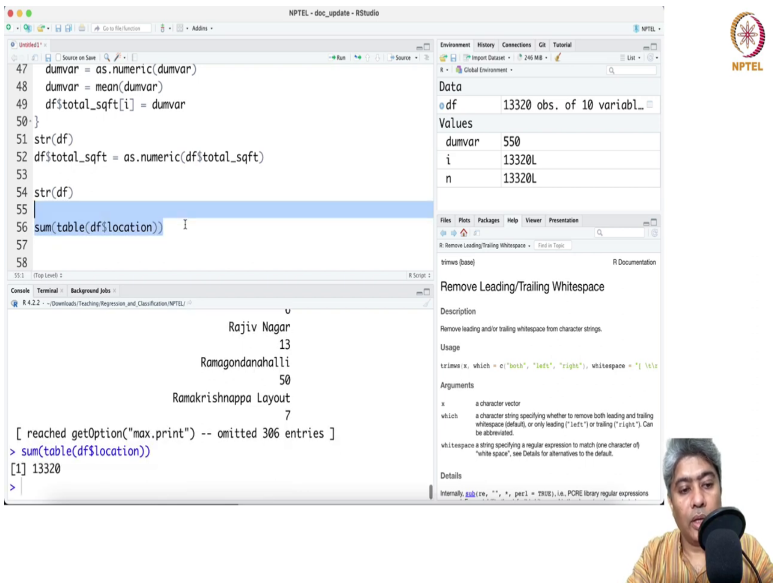
```
  dumvar = as.numeric(dumvar)
  dumvar = mean(dumvar)
  df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

table(df$location)
```

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

df	13320 obs. of 10 variabl…
dumvar	550
i	13320L
n	13320L

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)



```
    dumvar = as.numeric(dumvar)
    dumvar = mean(dumvar)
    df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

sum(table(df$location))
```

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

Data	
df	13320 obs. of 10 variabl…
Values	
dumvar	550
i	13320L
n	13320L

trimws {base} R Documentation

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
```

Arguments

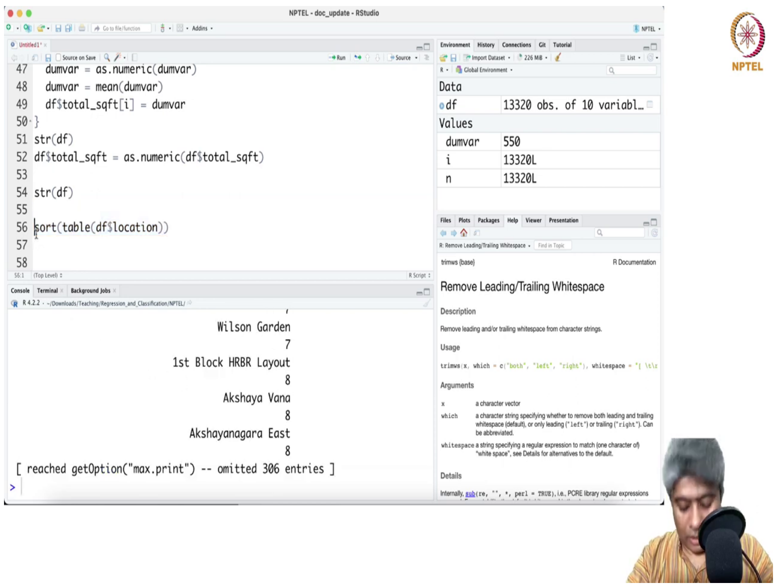
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

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)



```
  dumvar = as.numeric(dumvar)
  dumvar = mean(dumvar)
  df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

sort(table(df$location))
```

```
                 Wilson Garden
                             7
            1st Block HRBR Layout
                             8
                  Akshaya Vana
                             8
             Akshayanagara East
                             8
[ reached getOption("max.print") -- omitted 306 entries ]
>
```

Data	
df	13320 obs. of 10 variabl…
Values	
dumvar	550
i	13320L
n	13320L

trimws {base}   R Documentation

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
```

Arguments

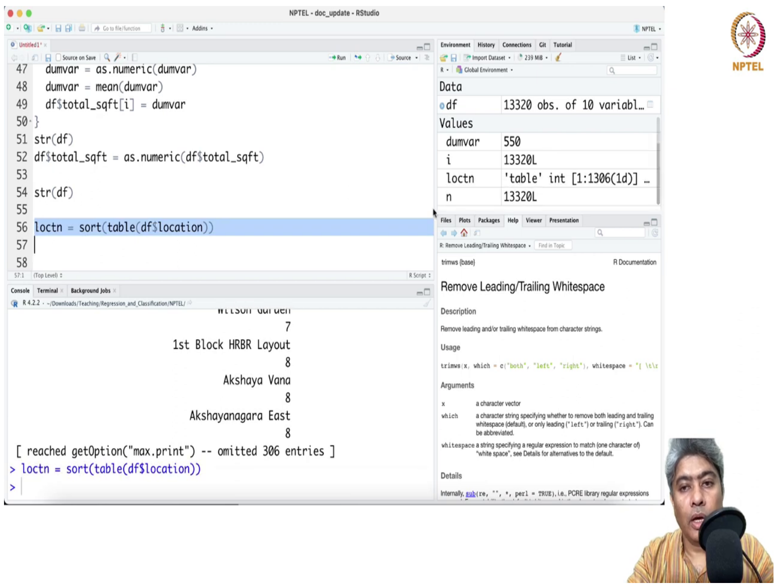
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

And then I can just sort them.

(Refer Slide Time: 16:20)



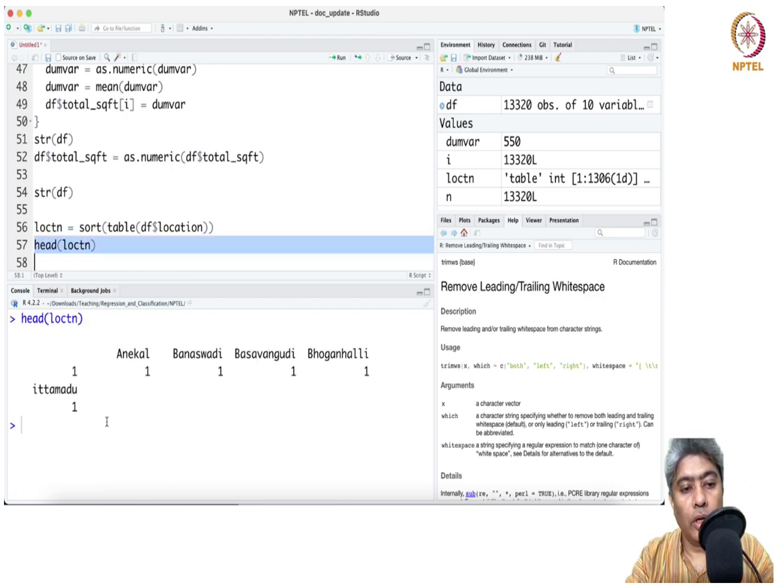
```
47   dumvar = as.numeric(dumvar)
48   dumvar = mean(dumvar)
49   df$total_sqft[i] = dumvar
50 }
51 str(df)
52 df$total_sqft = as.numeric(df$total_sqft)
53
54 str(df)
55
56 loctn = sort(table(df$location))
57
58
```

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

```
Data
df       13320 obs. of 10 variabl…
Values
dumvar   550
i        13320L
loctn    'table' int [1:1306(1d)] …
n        13320L
```

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

(Refer Slide Time: 16:33)


NPTEL - doc_update - RStudio
Untitled1*
Source on Save
Run
Source
Environment History Connections Git Tutorial
Import Dataset
238 MiB
List
Global Environment
Files Plots Packages Help Viewer Presentation
R: Remove Leading/Trailing Whitespace
Find in Topic
Console Terminal Background Jobs
R 4.2.2 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
R Script

```
    dumvar = as.numeric(dumvar)
    dumvar = mean(dumvar)
    df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

loctn = sort(table(df$location))
head(loctn)

```

```
> head(loctn)

                   Anekal    Banaswadi  Basavangudi  Bhoganhalli
          1             1            1            1            1
 ittamadu
          1
>
```

Data

df	13320 obs. of 10 variabl…

Values

dumvar	550
i	13320L
loctn	'table' int [1:1306(1d)] …
n	13320L

trimws {base} R Documentation

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
```

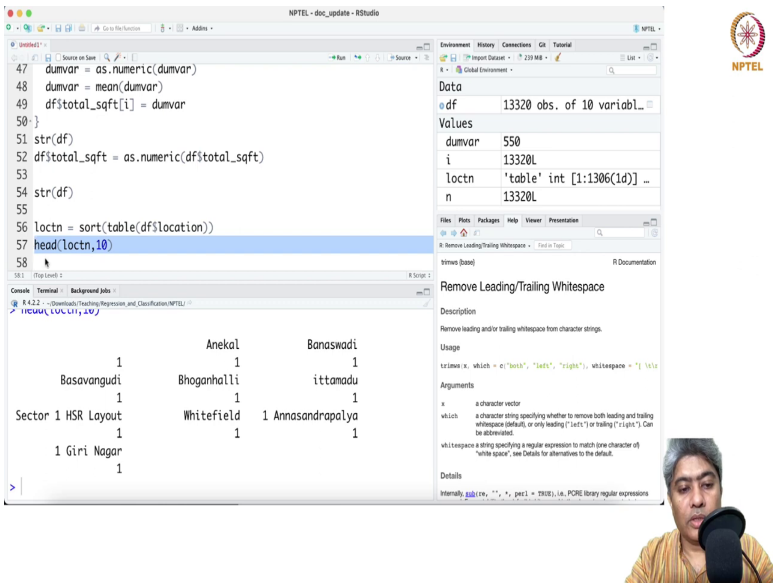
Arguments

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, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

(Refer Slide Time: 16:44)



```
  dumvar = as.numeric(dumvar)
  dumvar = mean(dumvar)
  df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

loctn = sort(table(df$location))
head(loctn,10)
```

```
> head(loctn,10)

                          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
>
```

Data

df	13320 obs. of 10 variabl…

Values

dumvar	550
i	13320L
loctn	'table' int [1:1306(1d)] …
n	13320L

trimws {base} R Documentation

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`

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.

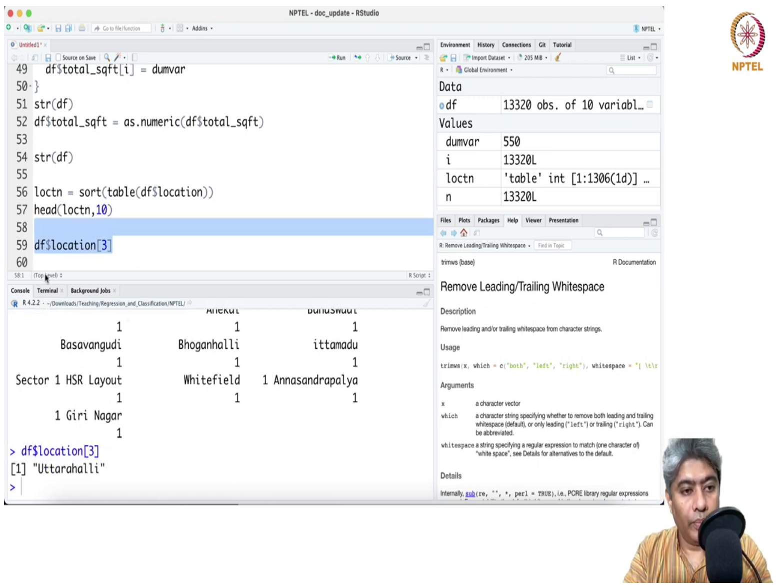
Details

Internally, `sub(re, "", *, perl = TRUE)`, i.e., PCRE library regular expressions

And so, if you just say head of loctn 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)



```
   df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

loctn = sort(table(df$location))
head(loctn,10)

df$location[3]
```

```
                 1                 1                 1
       Basavangudi       Bhoganhalli          ittamadu
                 1                 1                 1
Sector 1 HSR Layout        Whitefield  1 Annasandrapalya
                 1                 1                 1
      1 Giri Nagar
                 1
> df$location[3]
[1] "Uttarahalli"
>
```

df	13320 obs. of 10 variabl…
dumvar	550
i	13320L
loctn	'table' int [1:1306(1d)] …
n	13320L

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

Arguments

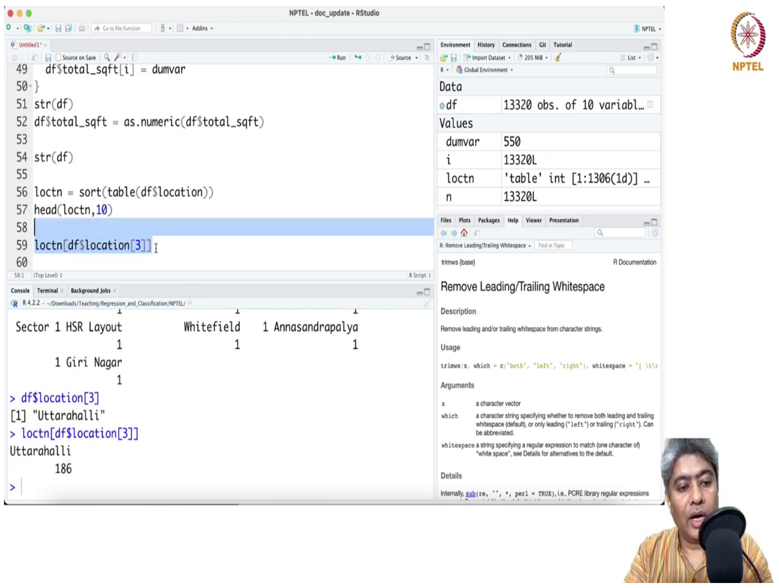
x a character vector

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.

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

(Refer Slide Time: 18:01)



```
  df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

loctn = sort(table(df$location))
head(loctn,10)

loctn[df$location[3]]
```

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

Data	
df	13320 obs. of 10 variabl…
Values	
dumvar	550
i	13320L
loctn	'table' int [1:1306(1d)] …
n	13320L

trimws {base} R Documentation

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
```

Arguments

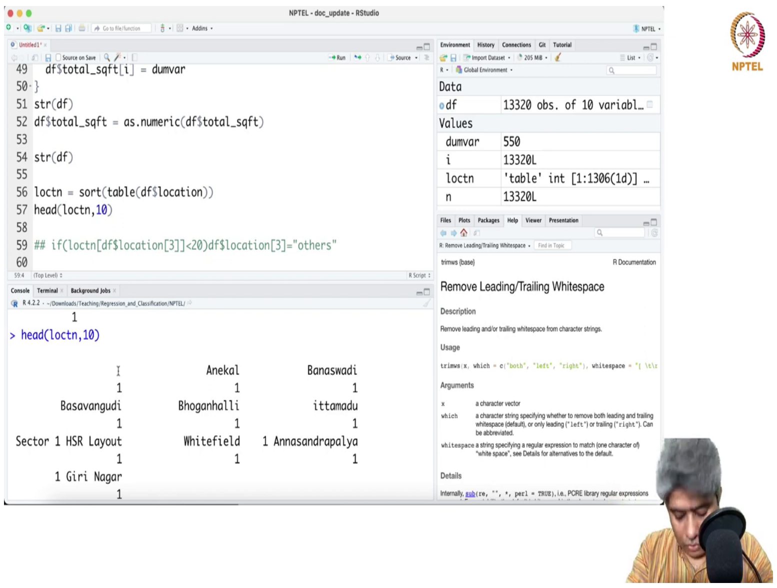
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

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)



```
  df$total_sqft[i] = dumvar
}
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

loctn = sort(table(df$location))
head(loctn,10)

## if(loctn[df$location[3]]<20)df$location[3]="others"

```

```
           1
> head(loctn,10)


                  I               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
```

```
Data
df       13320 obs. of 10 variabl…
Values
dumvar   550
i        13320L
loctn    'table' int [1:1306(1d)] …
n        13320L
```

trimws {base} R Documentation

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
```

Arguments

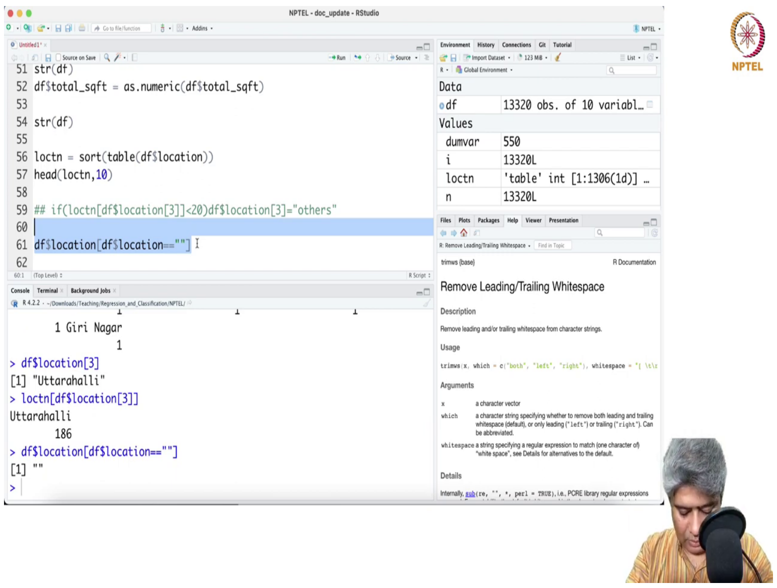
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

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)



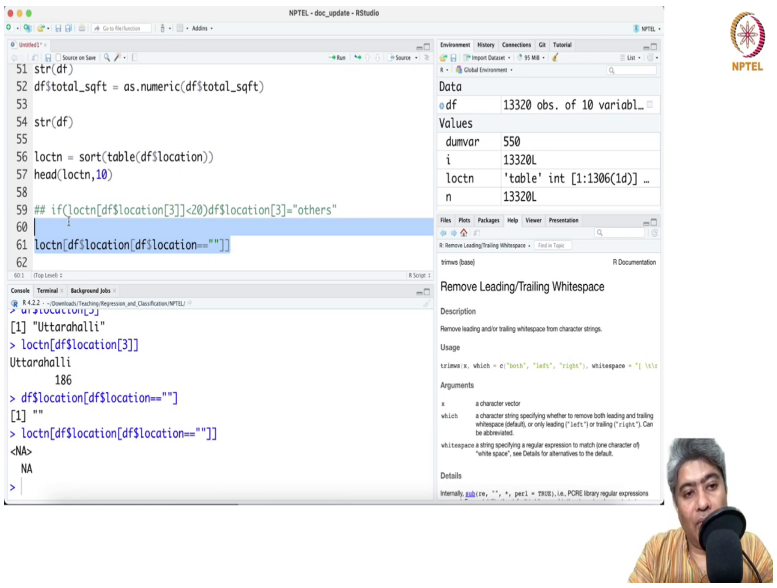
```
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

loctn = sort(table(df$location))
head(loctn,10)

## if(loctn[df$location[3]]<20)df$location[3]="others"

df$location[df$location==""]
```

```
        1 Giri Nagar
                   1
> df$location[3]
[1] "Uttarahalli"
> loctn[df$location[3]]
Uttarahalli
        186
> df$location[df$location==""]
[1] ""
>
```

NPTEL - doc_update - RStudio
Environment History Connections Git Tutorial
Data
df 13320 obs. of 10 variabl…
Values
dumvar 550
i 13320L
loctn 'table' int [1:1306(1d)] …
n 13320L
Remove Leading/Trailing Whitespace
NPTEL

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)



```
str(df)
df$total_sqft = as.numeric(df$total_sqft)

str(df)

loctn = sort(table(df$location))
head(loctn,10)

## if(loctn[df$location[3]]<20)df$location[3]="others"

loctn[df$location[df$location==""]]
```

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

```
Data
df        13320 obs. of 10 variabl…
Values
dumvar    550
i         13320L
loctn     'table' int [1:1306(1d)] …
n         13320L
```

trimws {base} R Documentation

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
```

Arguments

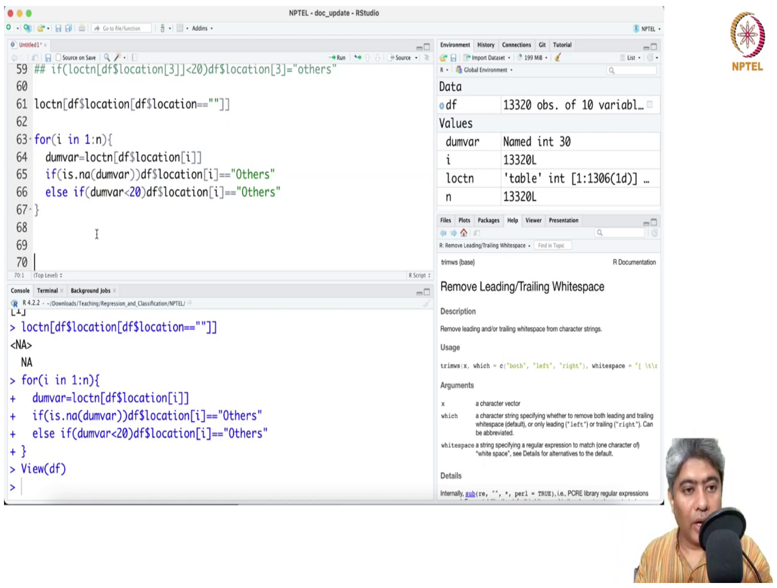
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

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)



```
## if(loctn[df$location[3]]<20)df$location[3]="others"

loctn[df$location[df$location==""]]

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"
}
```

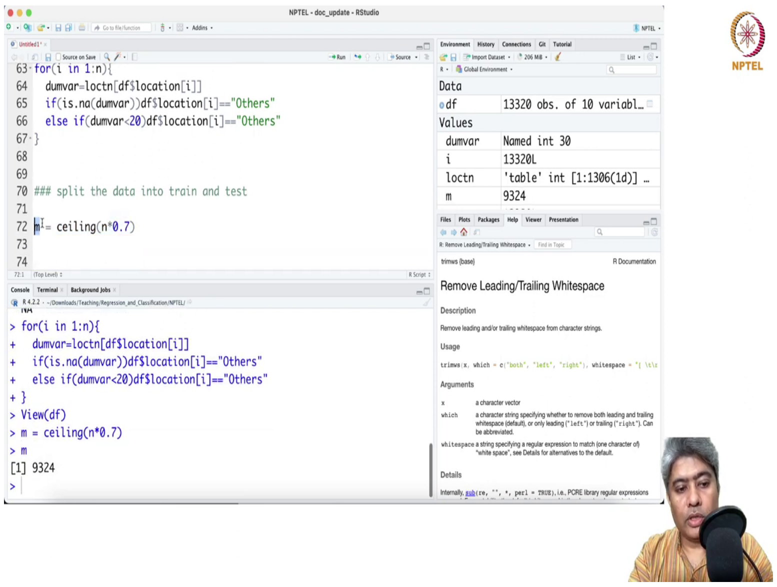
```
> loctn[df$location[df$location==""]]
<NA>
  NA
> for(i in 1:n){
+   dumvar=loctn[df$location[i]]
+   if(is.na(dumvar))df$location[i]=="Others"
+   else if(dumvar<20)df$location[i]=="Others"
+ }
> View(df)
>
```

```
Data
df        13320 obs. of 10 variabl…
Values
dumvar    Named int 30
i         13320L
loctn     'table' int [1:1306(1d)] …
n         13320L
```

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)



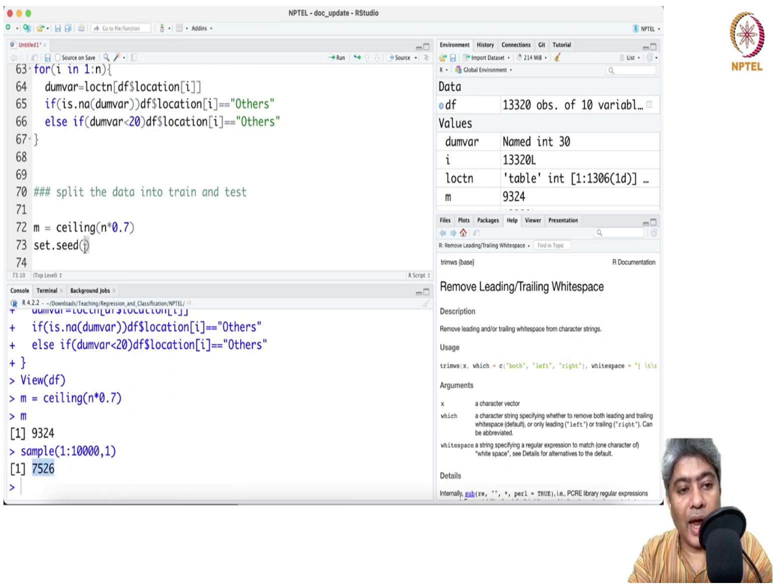
```
for(i in 1:n){
  dumvar=loctn[df$location[i]]
  if(is.na(dumvar))df$location[i]=="Others"
  else if(dumvar<20)df$location[i]=="Others"
}


### split the data into train and test

m = ceiling(n*0.7)
```

```
> 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
>
```

```
Data
df        13320 obs. of 10 variabl…
Values
dumvar    Named int 30
i         13320L
loctn     'table' int [1:1306(1d)] …
m         9324
```

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)



```
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"
}


### split the data into train and test

m = ceiling(n*0.7)
set.seed()

```

```
+   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
> sample(1:10000,1)
[1] 7526
>
```

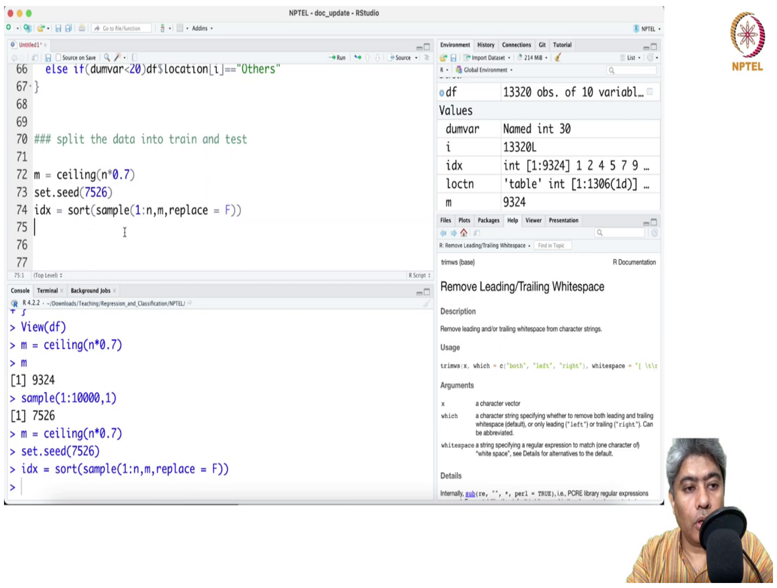
```
Data
df       13320 obs. of 10 variabl…
Values
dumvar   Named int 30
i        13320L
loctn    'table' int [1:1306(1d)] …
m        9324
```

Remove Leading/Trailing Whitespace

Remove leading and/or trailing whitespace from character strings.

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)



```
  else if(dumvar<20)df$location[i]=="Others"
}


### split the data into train and test

m = ceiling(n*0.7)
set.seed(7526)
idx = sort(sample(1:n,m,replace = F))
```

```
> View(df)
> m = ceiling(n*0.7)
> m
[1] 9324
> sample(1:10000,1)
[1] 7526
> m = ceiling(n*0.7)
> set.seed(7526)
> idx = sort(sample(1:n,m,replace = F))
>
```

```
df        13320 obs. of 10 variabl…
Values
dumvar    Named int 30
i         13320L
idx       int [1:9324] 1 2 4 5 7 9 …
loctn     'table' int [1:1306(1d)] …
m         9324
```

trimws {base} R Documentation

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
```

Arguments

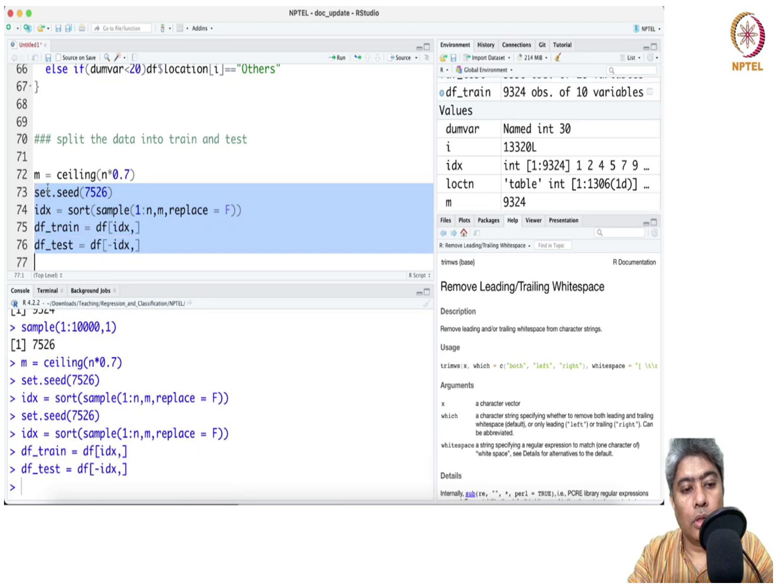
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions

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)



```r
  else if(dumvar<20)df$location[i]=="Others"
}



### split the data into train and test

m = ceiling(n*0.7)
set.seed(7526)
idx = sort(sample(1:n,m,replace = F))
df_train = df[idx,]
df_test = df[-idx,]
```

```
[1] 9324
> 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,]
>
```

```
df_train   9324 obs. of 10 variables
Values
dumvar     Named int 30
i          13320L
idx        int [1:9324] 1 2 4 5 7 9 …
loctn      'table' int [1:1306(1d)] …
m          9324
```

trimws {base} R Documentation

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`

Arguments

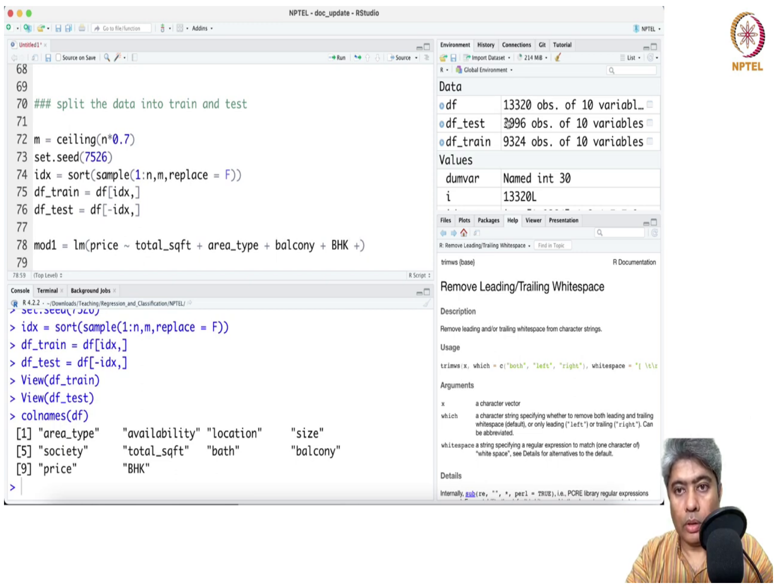
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, `sub(re, "", *, perl = TRUE)`, i.e., PCRE library regular expressions

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)



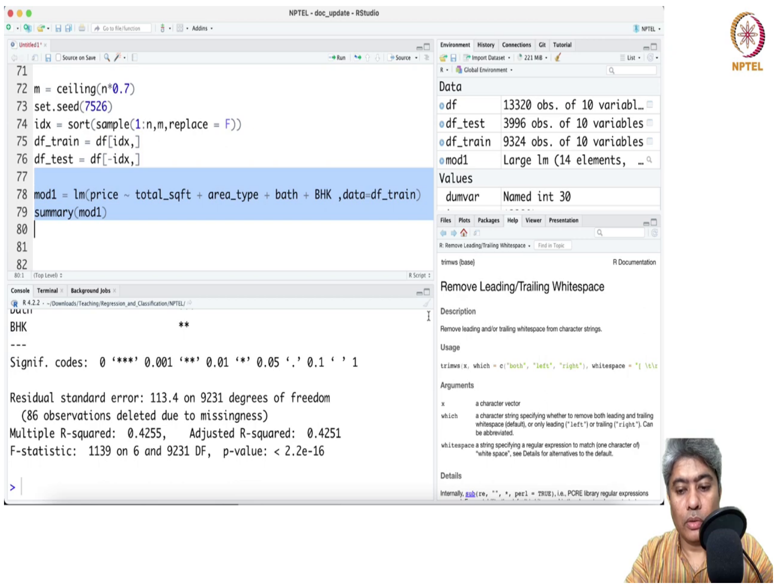
```
### split the data into train and test

m = ceiling(n*0.7)
set.seed(7526)
idx = sort(sample(1:n,m,replace = F))
df_train = df[idx,]
df_test = df[-idx,]

mod1 = lm(price ~ total_sqft + area_type + balcony + BHK +)
```

```
> 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"
>
```

```
Data
df        13320 obs. of 10 variabl…
df_test   3996 obs. of 10 variables
df_train  9324 obs. of 10 variables
Values
dumvar    Named int 30
i         13320L
```

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)



```r
m = ceiling(n*0.7)
set.seed(7526)
idx = sort(sample(1:n,m,replace = F))
df_train = df[idx,]
df_test = df[-idx,]

mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
summary(mod1)
```

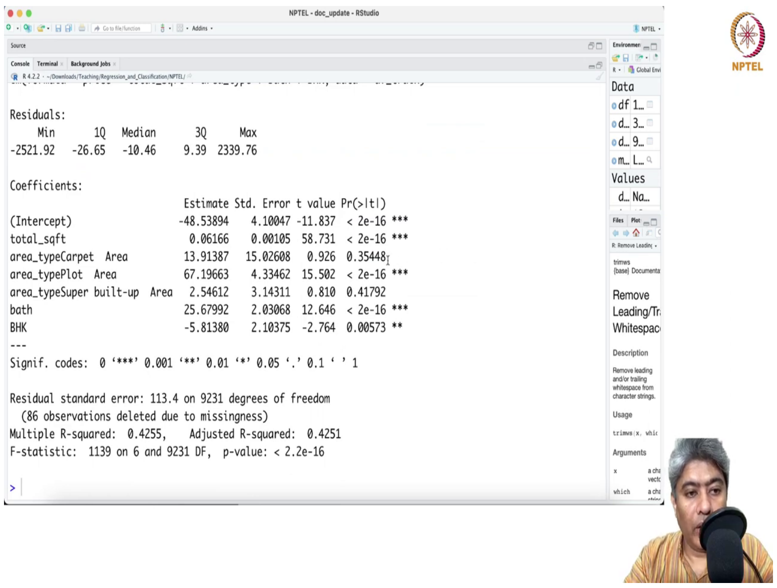
```
BHK                              **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

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
```

df	13320 obs. of 10 variabl…
df_test	3996 obs. of 10 variables
df_train	9324 obs. of 10 variables
mod1	Large lm (14 elements, …
dumvar	Named int 30

NPTEL - doc_update - RStudio
Remove Leading/Trailing Whitespace
NPTEL

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.

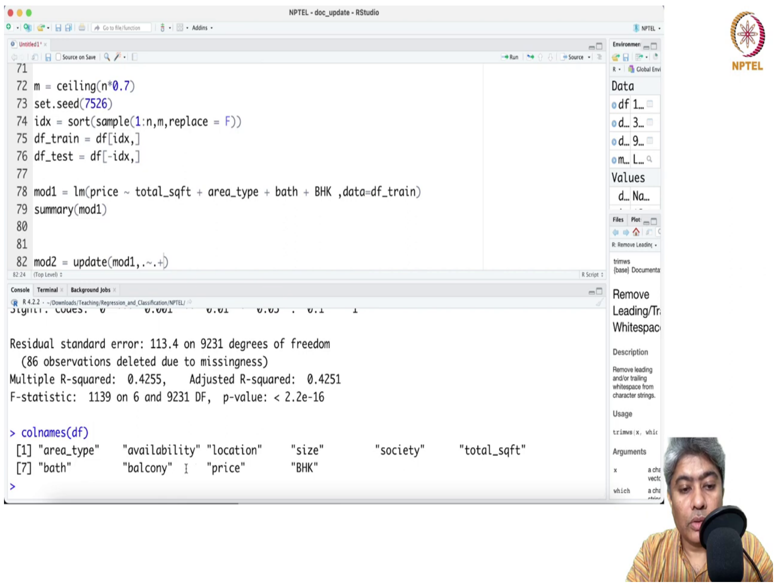
(Refer Slide Time: 26:54)



```
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)

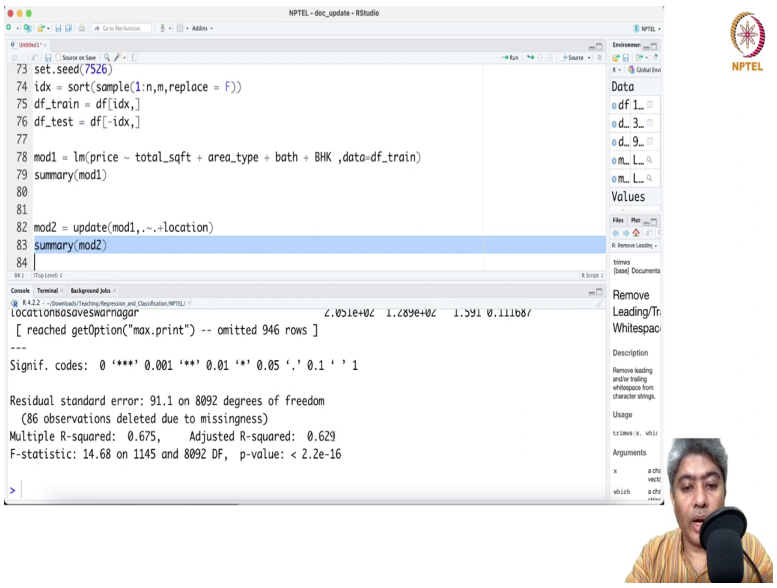


```
m = ceiling(n*0.7)
set.seed(7526)
idx = sort(sample(1:n,m,replace = F))
df_train = df[idx,]
df_test = df[-idx,]

mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
summary(mod1)


mod2 = update(mod1,.~.+)
```

```
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

> colnames(df)
 [1] "area_type"    "availability" "location"     "size"         "society"      "total_sqft"
 [7] "bath"         "balcony"      "price"        "BHK"
>
```

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)

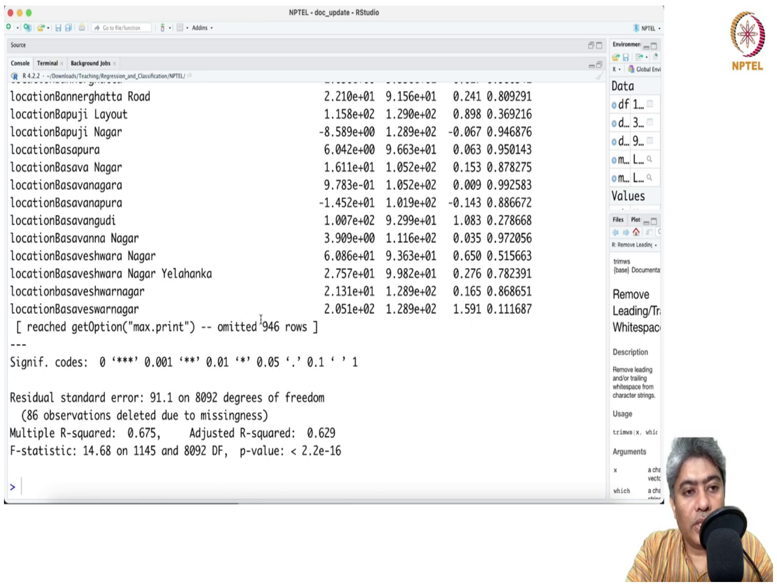


```
set.seed(7526)
idx = sort(sample(1:n,m,replace = F))
df_train = df[idx,]
df_test = df[-idx,]

mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
summary(mod1)


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

```
locationBasaveswarnagar                        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

>
```

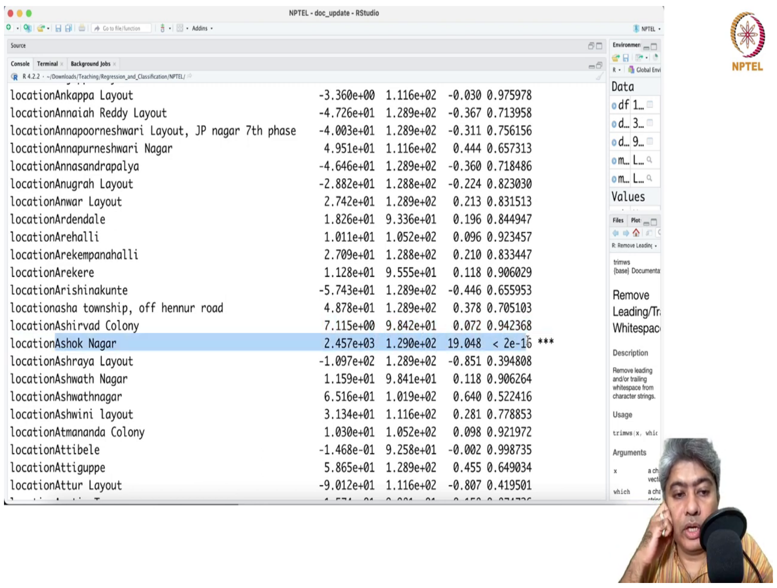
So, you just put location and it should work alright and then let us see summary mod2 of location and you can see that adjusted r square now went up to 0.629 right ok.

(Refer Slide Time: 29:00)



```
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
locationBasaveswarnagar                    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

> 
```

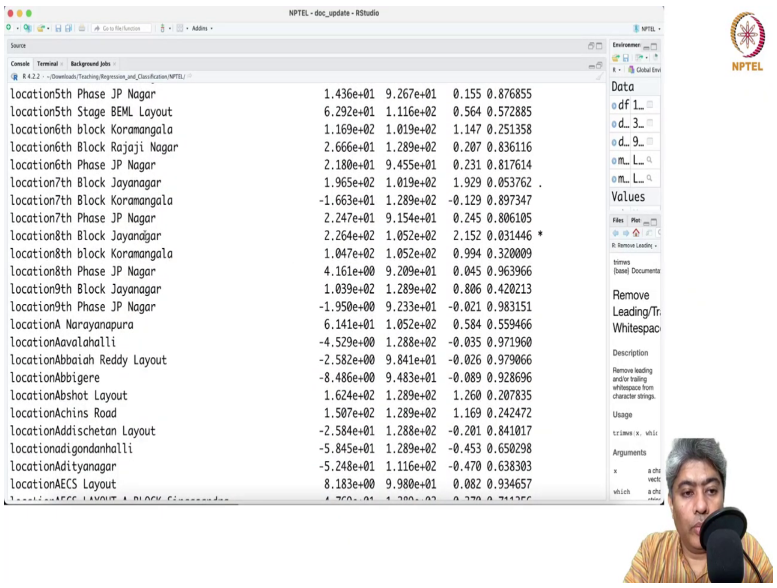
(Refer Slide Time: 29:06)



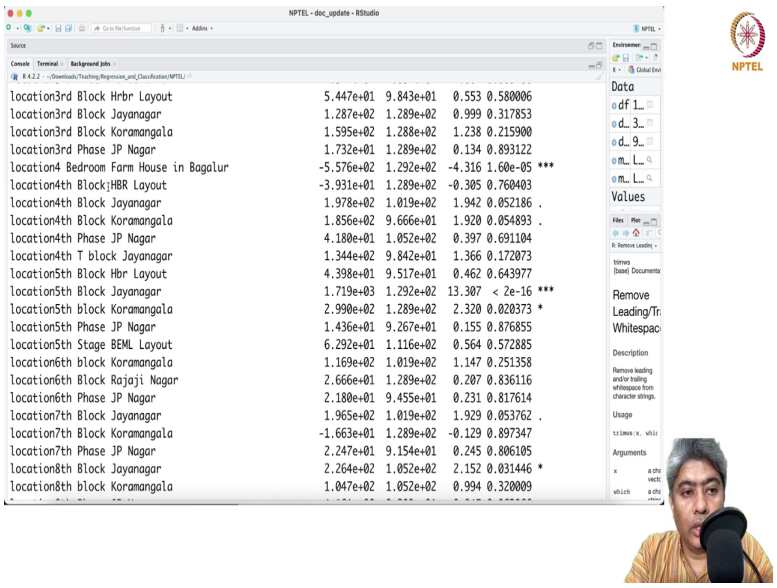
```
locationAnkappa Layout                              -3.360e+00  1.116e+02  -0.030 0.975978
locationAnnaiah Reddy Layout                        -4.726e+01  1.289e+02  -0.367 0.713958
locationAnnapoorneshwari Layout, JP nagar 7th phase -4.003e+01  1.289e+02  -0.311 0.756156
locationAnnapurneshwari 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
locationArekempanahalli                              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)


```
location5th Phase JP Nagar             1.436e+01  9.267e+01   0.155 0.876855
location5th Stage BEML Layout          6.292e+01  1.116e+02   0.564 0.572885
location6th block Koramangala          1.169e+02  1.019e+02   1.147 0.251358
location6th Block Rajaji Nagar         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
locationAavalahalli                   -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
locationAddischetan Layout            -2.584e+01  1.288e+02  -0.201 0.841017
locationadigondanhalli                -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  9.980e+01   0.082 0.934657
```

(Refer Slide Time: 29:35)

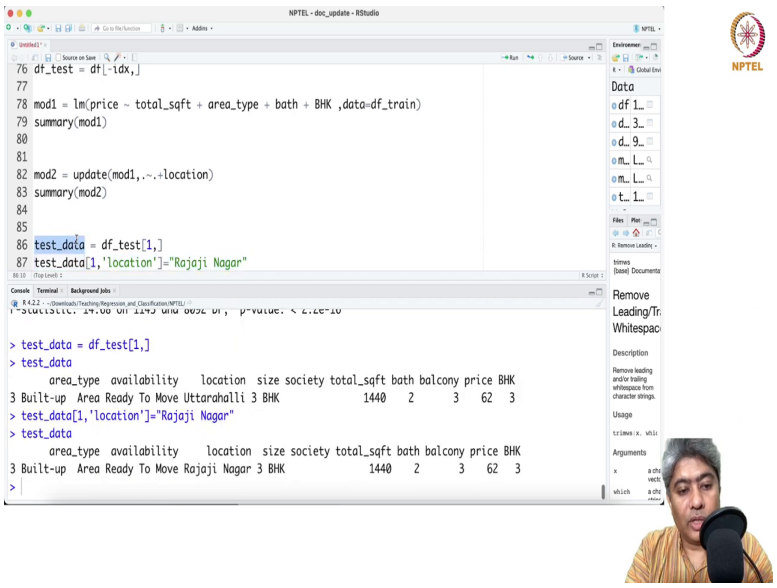

NPTEL - doc_update - RStudio
Console
Terminal
Background Jobs
R 4.2.2 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/

```
location3rd Block Hrbr Layout                   5.447e+01  9.843e+01   0.553 0.580006
location3rd Block Jayanagar                     1.287e+02  1.289e+02   0.999 0.317853
location3rd Block Koramangala                   1.595e+02  1.288e+02   1.238 0.215900
location3rd Phase JP Nagar                      1.732e+01  1.289e+02   0.134 0.893122
location4 Bedroom Farm House in Bagalur        -5.576e+02  1.292e+02  -4.316 1.60e-05 ***
location4th Block HBR Layout                   -3.931e+01  1.289e+02  -0.305 0.760403
location4th Block Jayanagar                     1.978e+02  1.019e+02   1.942 0.052186 .
location4th Block Koramangala                   1.856e+02  9.666e+01   1.920 0.054893 .
location4th Phase JP Nagar                      4.180e+01  1.052e+02   0.397 0.691104
location4th T block Jayanagar                   1.344e+02  9.842e+01   1.366 0.172073
location5th Block Hbr Layout                    4.398e+01  9.517e+01   0.462 0.643977
location5th Block Jayanagar                     1.719e+03  1.292e+02  13.307  < 2e-16 ***
location5th block Koramangala                   2.990e+02  1.289e+02   2.320 0.020373 *
location5th Phase JP Nagar                      1.436e+01  9.267e+01   0.155 0.876855
location5th Stage BEML Layout                   6.292e+01  1.116e+02   0.564 0.572885
location6th block Koramangala                   1.169e+02  1.019e+02   1.147 0.251358
location6th Block Rajaji Nagar                  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
```

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)



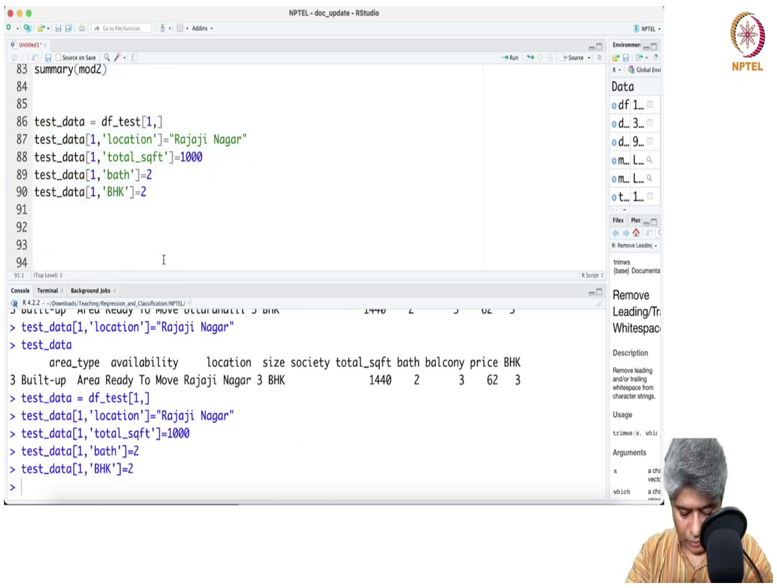
```
df_test = df[-idx,]

mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
summary(mod1)


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


test_data = df_test[1,]
test_data[1,'location']="Rajaji Nagar"
```

```
F-statistic: 14.08 on 1145 and 8092 DF,  p-value: < 2.2e-16

> 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)

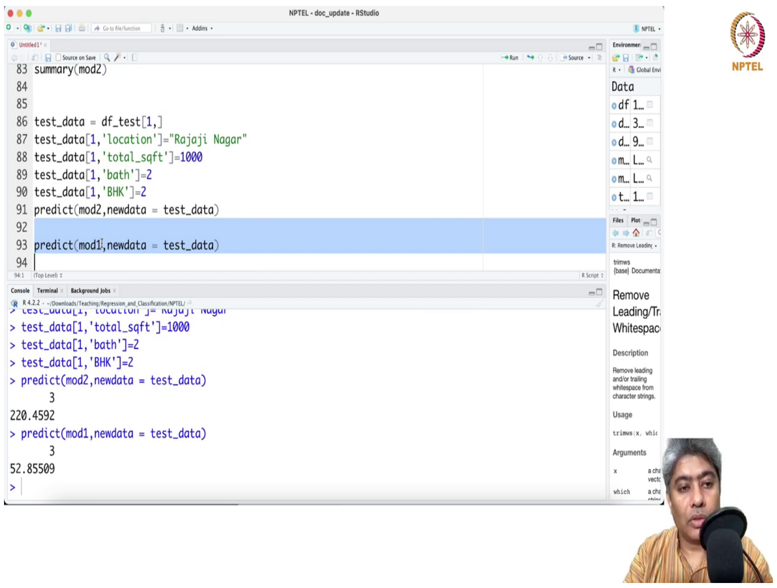


```
summary(mod2)


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
```

```
> 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
> 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
>
```

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)

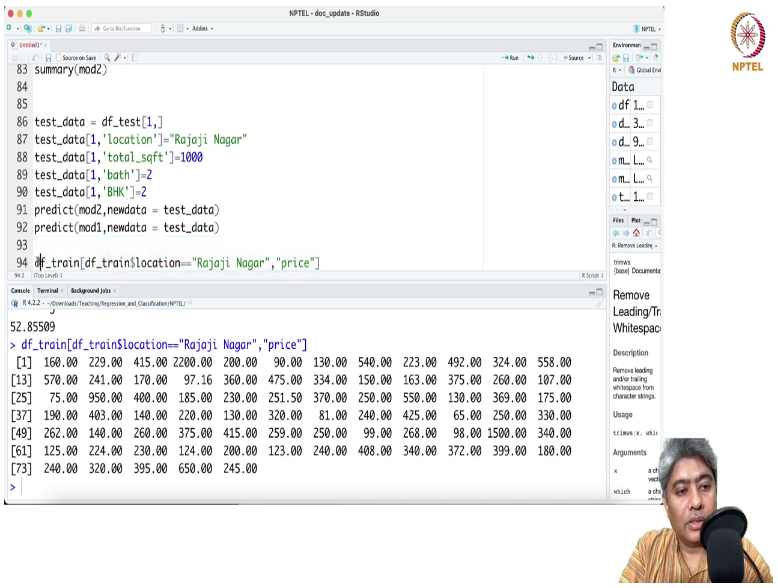


```
summary(mod2)


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
predict(mod2,newdata = test_data)

predict(mod1,newdata = test_data)
```

```
> 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
>
```

So, let us run this through and now if I just predict; predict mod2 with newdata equal to test data 220.45. So, it is 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)

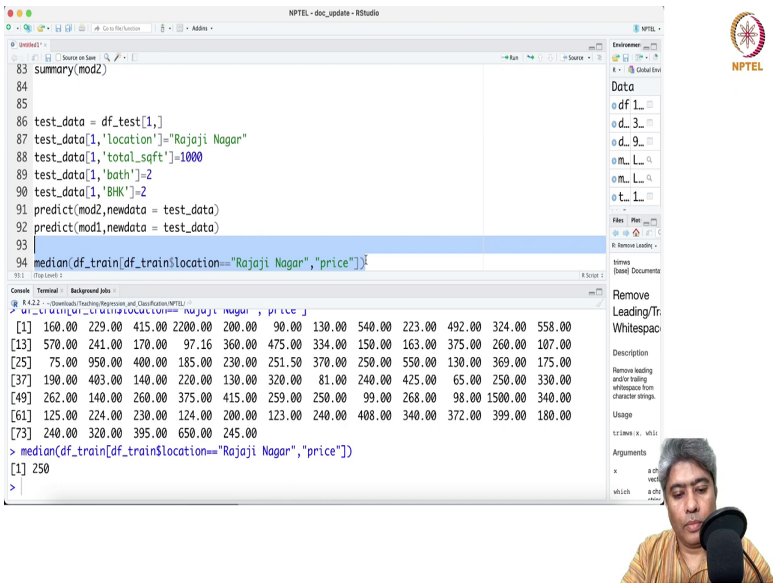


```
summary(mod2)


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
predict(mod2,newdata = test_data)
predict(mod1,newdata = test_data)

df_train[df_train$location=="Rajaji Nagar","price"]
```

```
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)

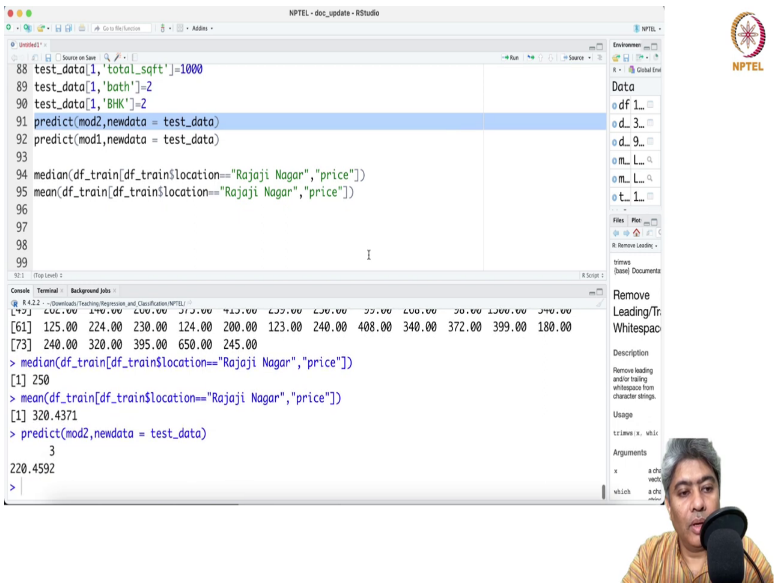


```r
summary(mod2)


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
predict(mod2,newdata = test_data)
predict(mod1,newdata = test_data)

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

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

(Refer Slide Time: 34:47)

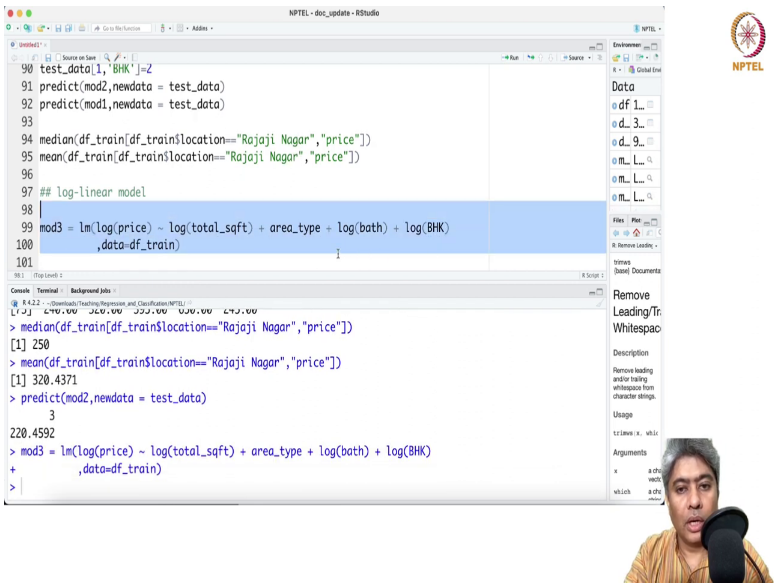


```
test_data[1,'total_sqft']=1000
test_data[1,'bath']=2
test_data[1,'BHK']=2
predict(mod2,newdata = test_data)
predict(mod1,newdata = test_data)

median(df_train[df_train$location=="Rajaji Nagar","price"])
mean(df_train[df_train$location=="Rajaji Nagar","price"])
```

```
[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
>
```

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)

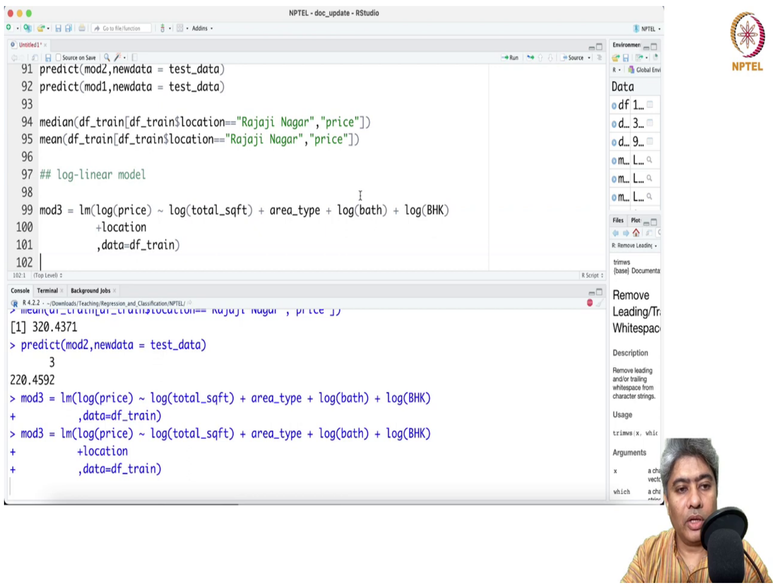


```
test_data[1,'BHK']=2
predict(mod2,newdata = test_data)
predict(mod1,newdata = test_data)

median(df_train[df_train$location=="Rajaji Nagar","price"])
mean(df_train[df_train$location=="Rajaji Nagar","price"])

## log-linear model

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

```
> 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)
>
```

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)

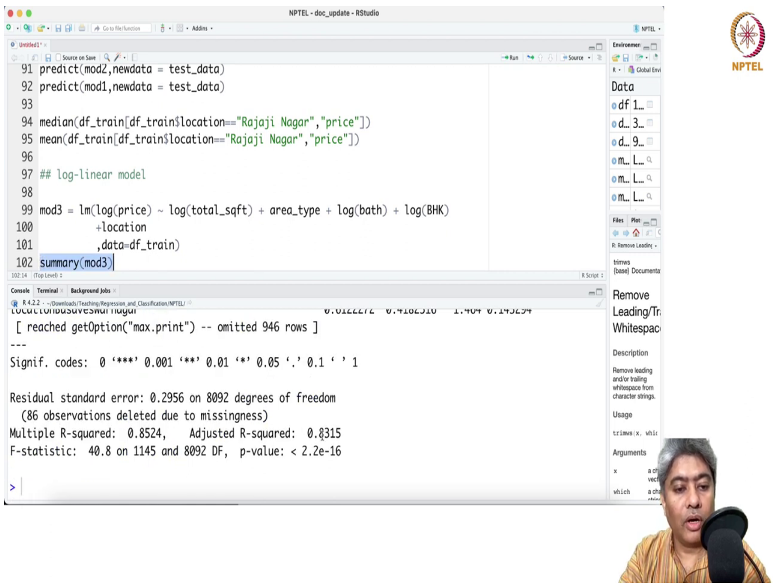


```
predict(mod2,newdata = test_data)
predict(mod1,newdata = test_data)

median(df_train[df_train$location=="Rajaji Nagar","price"])
mean(df_train[df_train$location=="Rajaji Nagar","price"])

## log-linear model

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

```
[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)
```

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)

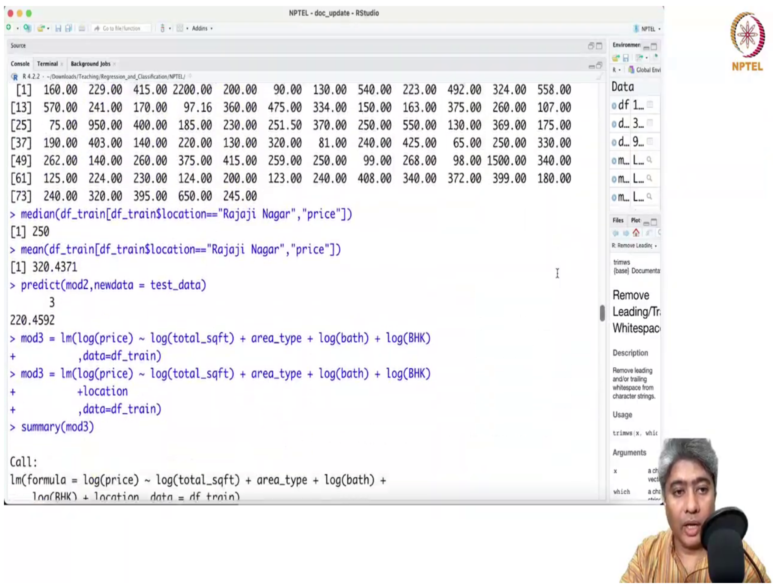


```r
predict(mod2,newdata = test_data)
predict(mod1,newdata = test_data)

median(df_train[df_train$location=="Rajaji Nagar","price"])
mean(df_train[df_train$location=="Rajaji Nagar","price"])

## log-linear model

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

```
 [ 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

>
```

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

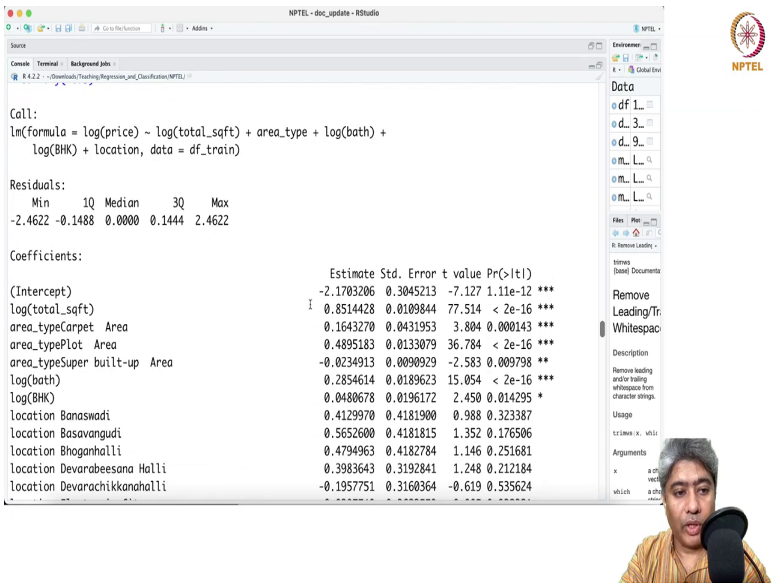
(Refer Slide Time: 36:50)



```
 [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.

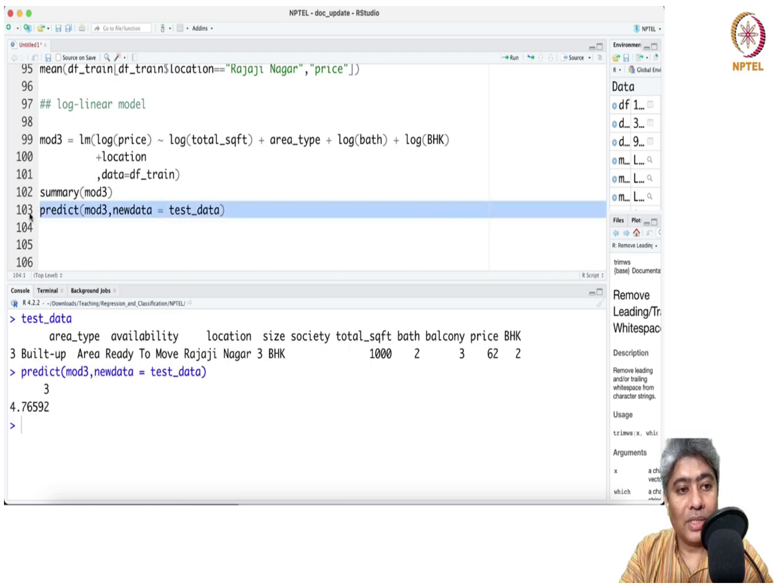
(Refer Slide Time: 36:52)



```
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
```

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)

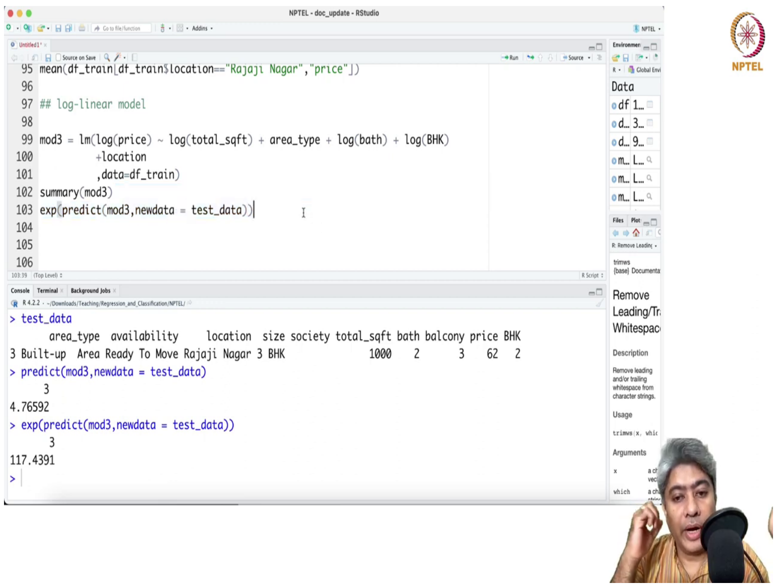


```r
mean(df_train[df_train$location=="Rajaji Nagar","price"])

## log-linear model

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

```
> 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
>
```

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)

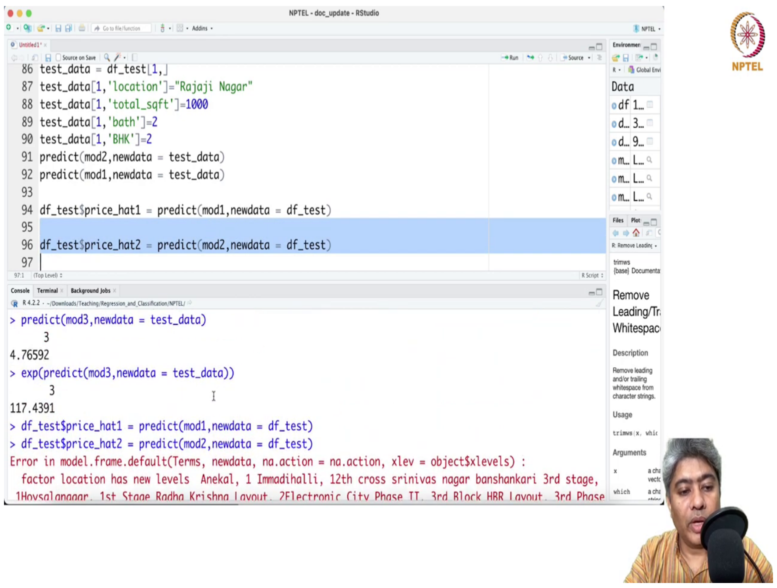


```
mean(df_train[df_train$location=="Rajaji Nagar","price"])

## log-linear model

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

```
> 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
>
```

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 slighlty 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)

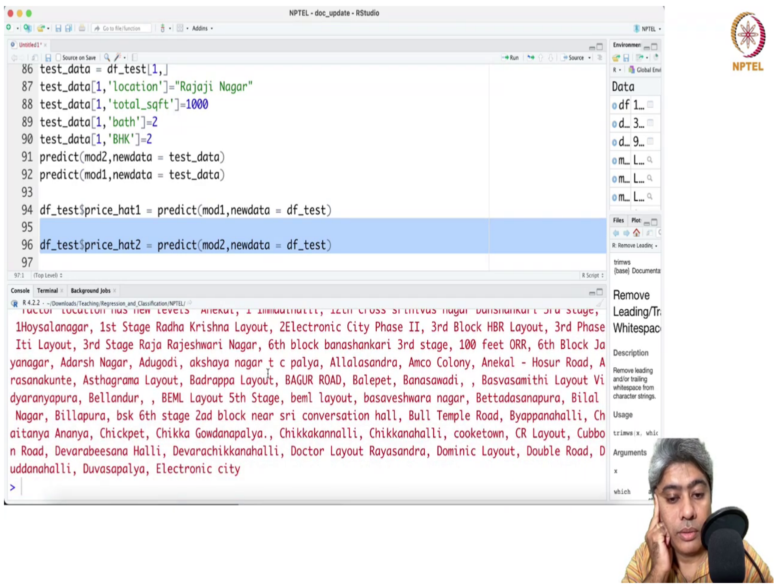


```
test_data = df_test[1,]
test_data[1,'location']="Rajaji Nagar"
test_data[1,'total_sqft']=1000
test_data[1,'bath']=2
test_data[1,'BHK']=2
predict(mod2,newdata = test_data)
predict(mod1,newdata = test_data)

df_test$price_hat1 = predict(mod1,newdata = df_test)

df_test$price_hat2 = predict(mod2,newdata = df_test)

```

```
> 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 Immadihalli, 12th cross srinivas nagar banshankari 3rd stage,
 1Hoysalanagar, 1st Stage Radha Krishna Layout, 2Electronic City Phase II, 3rd Block HBR Layout, 3rd Phase
```

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.

(Refer Slide Time: 38:57)



```
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
predict(mod2,newdata = test_data)
predict(mod1,newdata = test_data)

df_test$price_hat1 = predict(mod1,newdata = df_test)

df_test$price_hat2 = predict(mod2,newdata = df_test)

```

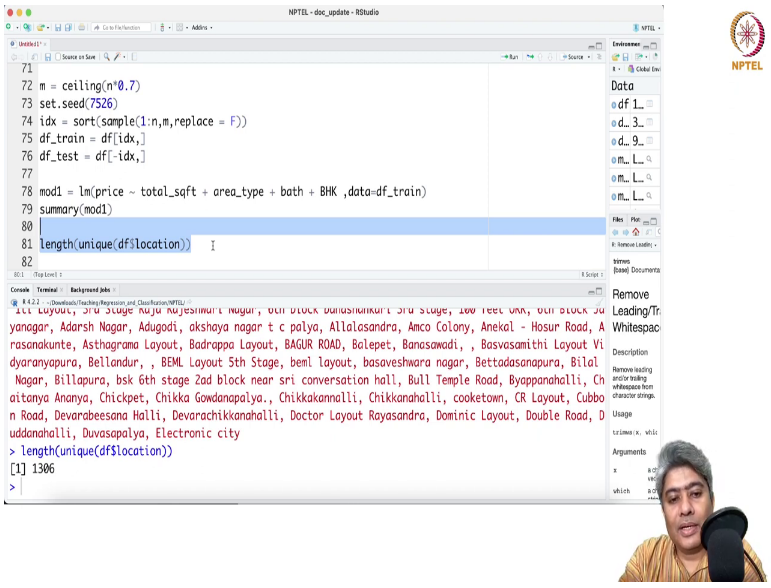
```
factor location has new levels  Anekal, 1 Immadihalli, 12th cross srinivas nagar banshankari 3rd stage,
 1Hoysalanagar, 1st Stage Radha Krishna Layout, 2Electronic City Phase II, 3rd Block HBR Layout, 3rd Phase
 Iti Layout, 3rd Stage Raja Rajeshwari Nagar, 6th block banashankari 3rd stage, 100 feet ORR, 6th Block Ja
yanagar, Adarsh Nagar, Adugodi, akshaya nagar t c palya, Allalasandra, Amco Colony, Anekal - Hosur Road, A
rasanakunte, Asthagrama Layout, Badrappa Layout, BAGUR ROAD, Balepet, Banasawadi, , Basvasamithi Layout Vi
dyaranyapura, Bellandur, , BEML Layout 5th Stage, beml layout, basaveshwara nagar, Bettadasanapura, Bilal
 Nagar, Billapura, bsk 6th stage 2ad block near sri conversation hall, Bull Temple Road, Byappanahalli, Ch
aitanya Ananya, Chickpet, Chikka Gowdanapalya., Chikkakannalli, Chikkanahalli, cooketown, CR Layout, Cubbo
n Road, Devarabeesana Halli, Devarachikkanahalli, Doctor Layout Rayasandra, Dominic Layout, Double Road, D
uddanahalli, Duvasapalya, Electronic city
>
```

There is some error its throwing because in the test second model its same location has new level ok that is this is bit of a problem always I have found when we have too many you have too many. So, basically what is happening the issue is I will tell you if there are cases where the test dataset has some values which are not there in the not there in the train data set.

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)



```
m = ceiling(n*0.7)
set.seed(7526)
idx = sort(sample(1:n,m,replace = F))
df_train = df[idx,]
df_test = df[-idx,]

mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
summary(mod1)

length(unique(df$location))
```

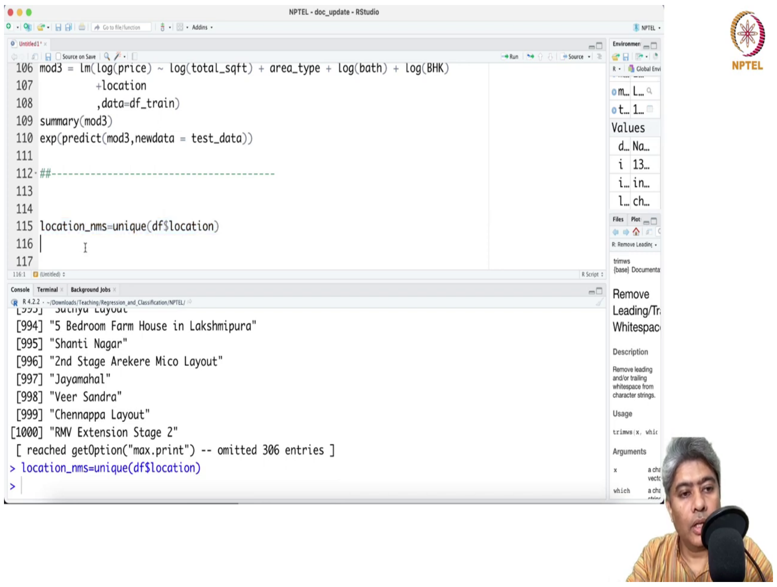
```
yanagar, Adarsh Nagar, Adugodi, akshaya nagar t c palya, Allalasandra, Amco Colony, Anekal - Hosur Road, A
rasanakunte, Asthagrama Layout, Badrappa Layout, BAGUR ROAD, Balepet, Banasawadi, , Basvasamithi Layout Vi
dyaranyapura, Bellandur, , BEML Layout 5th Stage, beml layout, basaveshwara nagar, Bettadasanapura, Bilal
 Nagar, Billapura, bsk 6th stage 2ad block near sri conversation hall, Bull Temple Road, Byappanahalli, Ch
aitanya Ananya, Chickpet, Chikka Gowdanapalya., Chikkakannalli, Chikkanahalli, cooketown, CR Layout, Cubbo
n Road, Devarabeesana Halli, Devarachikkanahalli, Doctor Layout Rayasandra, Dominic Layout, Double Road, D
uddanahalli, Duvasapalya, Electronic city
> length(unique(df$location))
[1] 1306
>
```

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)

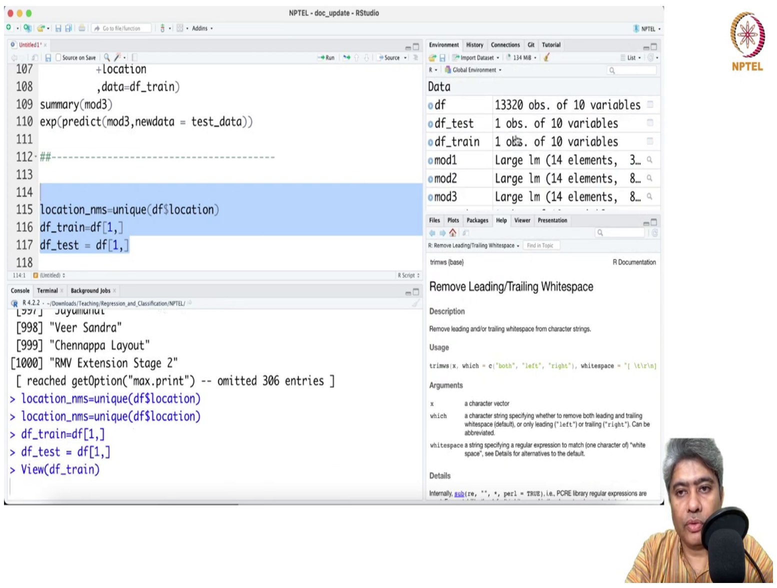


```
mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
          +location
          ,data=df_train)
summary(mod3)
exp(predict(mod3,newdata = test_data))

##---------------------------------------


location_nms=unique(df$location)

```

```
[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)
>
```

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)



```
            +location
            ,data=df_train)
summary(mod3)
exp(predict(mod3,newdata = test_data))

##---------------------------------------


location_nms=unique(df$location)
df_train=df[1,]
df_test = df[1,]
```

```
 [997] "Jayamahal"
 [998] "Veer Sandra"
 [999] "Chennappa Layout"
[1000] "RMV Extension Stage 2"
 [ reached getOption("max.print") -- omitted 306 entries ]
> location_nms=unique(df$location)
> location_nms=unique(df$location)
> df_train=df[1,]
> df_test = df[1,]
> View(df_train)
```

```
Data
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…
```

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

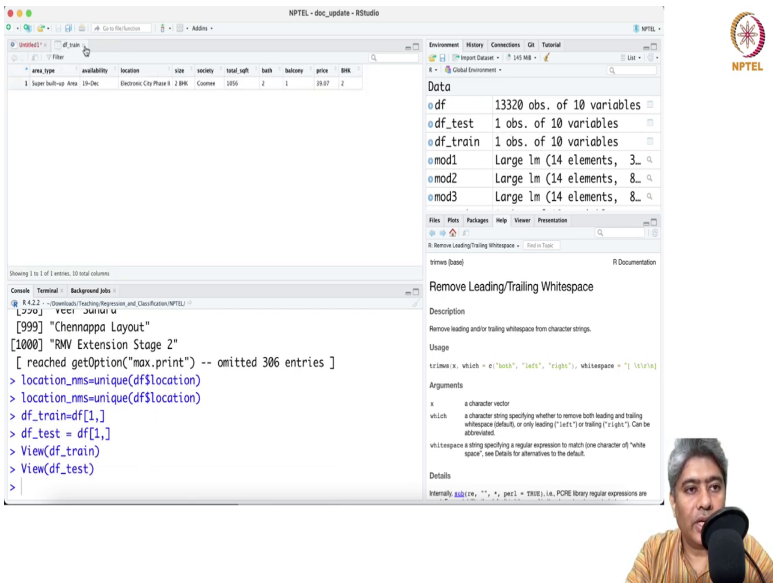
x — a character vector

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.

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)



NPTEL - doc_update - RStudio

	area_type	availability	location	size	society	total_sqft	bath	balcony	price	BHK
1	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2	1	39.07	2

Showing 1 to 1 of 1 entries, 10 total columns

```
 [999] "Chennappa Layout"
[1000] "RMV Extension Stage 2"
 [ reached getOption("max.print") -- omitted 306 entries ]
> location_nms=unique(df$location)
> location_nms=unique(df$location)
> df_train=df[1,]
> df_test = df[1,]
> View(df_train)
> View(df_test)
>
```

Data

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…

R: Remove Leading/Trailing Whitespace

trimws {base} R Documentation

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

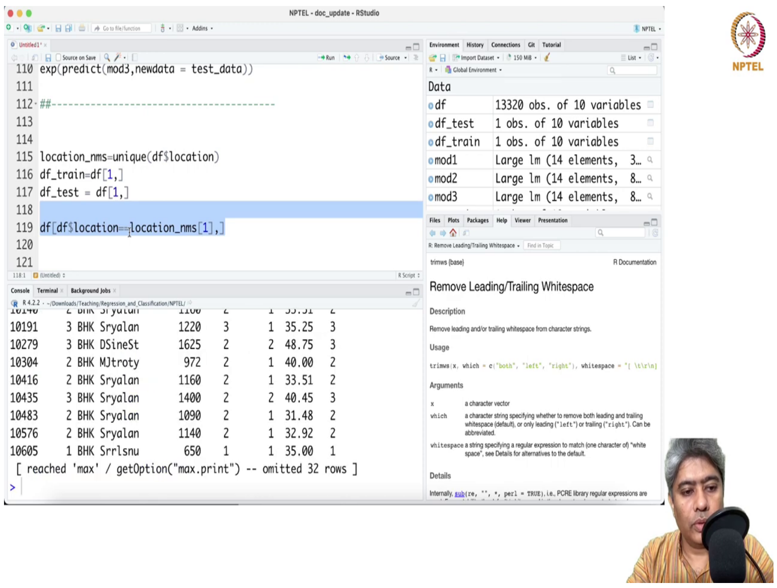
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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)



```
exp(predict(mod3,newdata = test_data))

##--------------------------------------


location_nms=unique(df$location)
df_train=df[1,]
df_test = df[1,]

df[df$location==location_nms[1],]


```

```
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 Srrlsnu        650   1        1  35.00   1
 [ reached 'max' / getOption("max.print") -- omitted 32 rows ]
>
```

```
Data
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…
```

Remove Leading/Trailing Whitespace

Description

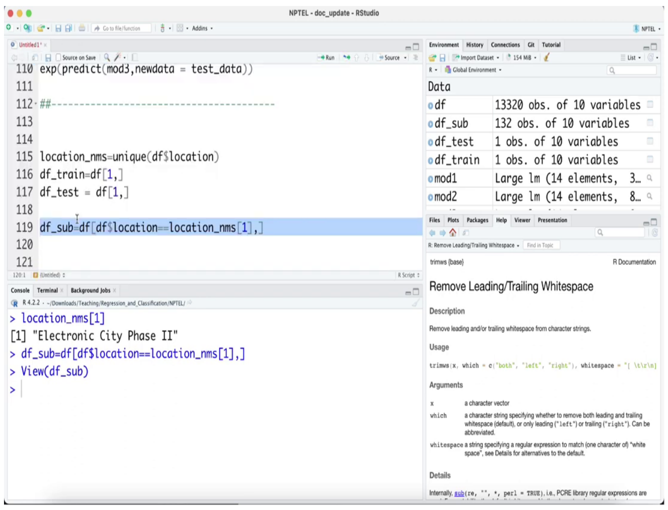
Remove leading and/or trailing whitespace from character strings.

Usage

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

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)



```
exp(predict(mod3,newdata = test_data))

##-------------------------------------


location_nms=unique(df$location)
df_train=df[1,]
df_test = df[1,]

df_sub=df[df$location==location_nms[1],]
```

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

```
Data
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…
```

trimws {base}   R Documentation

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

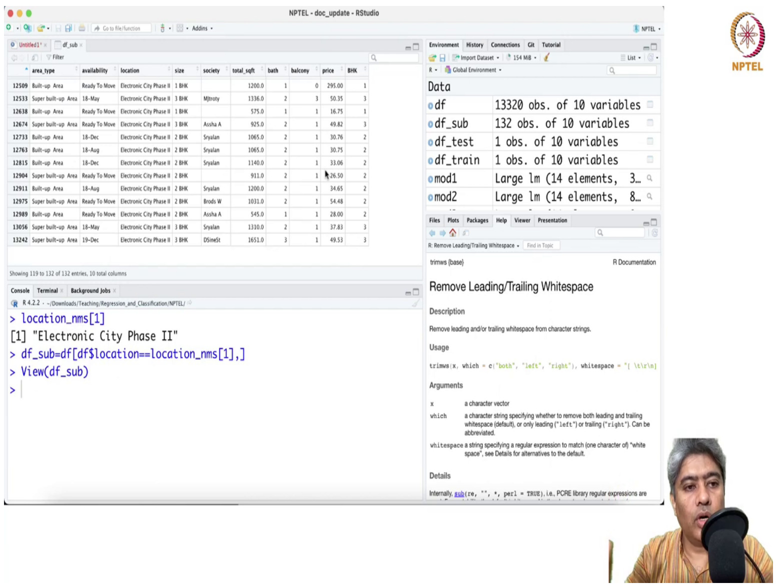
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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)



	area_type	availability	location	size	society	total_sqft	bath	balcony	price	BHK
12509	Built-up Area	Ready To Move	Electronic City Phase II	1 BHK		1200.0	1	0	295.00	1
12533	Super built-up Area	18-May	Electronic City Phase II	3 BHK	MJtroty	1336.0	2	3	50.35	3
12638	Built-up Area	Ready To Move	Electronic City Phase II	1 BHK		575.0	1	1	16.75	1
12674	Super built-up Area	Ready To Move	Electronic City Phase II	3 BHK	Assha A	925.0	2	1	49.82	3
12733	Built-up Area	18-Dec	Electronic City Phase II	2 BHK	Sryalan	1065.0	2	1	30.76	2
12763	Built-up Area	18-Aug	Electronic City Phase II	2 BHK	Sryalan	1065.0	2	1	30.75	2
12815	Built-up Area	18-Dec	Electronic City Phase II	2 BHK	Sryalan	1140.0	2	1	33.06	2
12904	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK		911.0	2	1	26.50	2
12911	Built-up Area	18-Aug	Electronic City Phase II	2 BHK	Sryalan	1200.0	2	1	34.65	2
12975	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Brods W	1031.0	2	1	54.48	2
12989	Built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Assha A	545.0	1	1	28.00	2
13056	Super built-up Area	18-May	Electronic City Phase II	3 BHK	Sryalan	1310.0	2	1	37.83	3
13242	Super built-up Area	19-Dec	Electronic City Phase II	3 BHK	DSineSt	1651.0	3	1	49.53	3

Showing 119 to 132 of 132 entries, 10 total columns

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

Data

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…

trimws {base} R Documentation

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

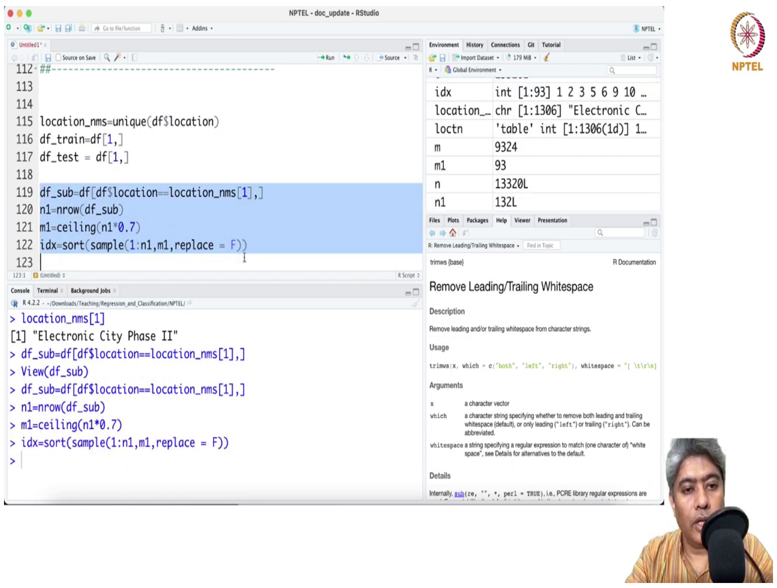
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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)



```
##------------------------------------------


location_nms=unique(df$location)
df_train=df[1,]
df_test = df[1,]

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))
```

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

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

trimws {base} R Documentation

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

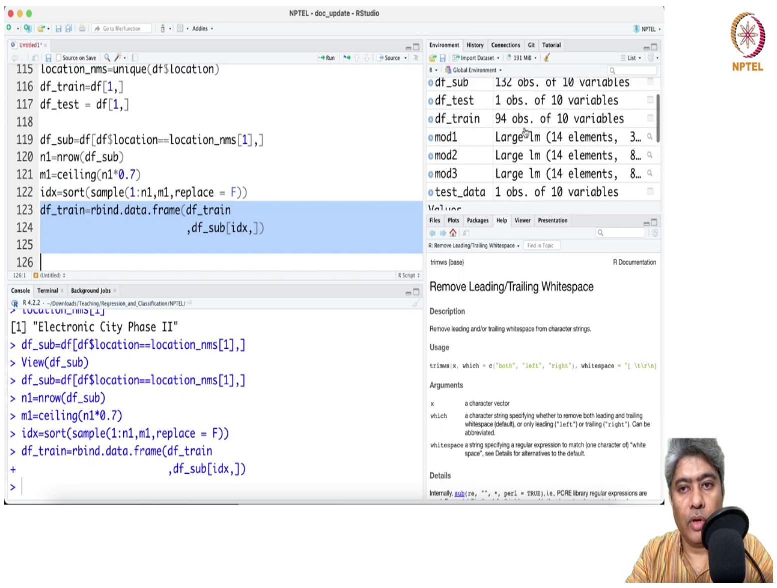
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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)



```r
location_nms=unique(df$location)
df_train=df[1,]
df_test = df[1,]

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,])
```

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

```
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
```

trimws {base} R Documentation

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

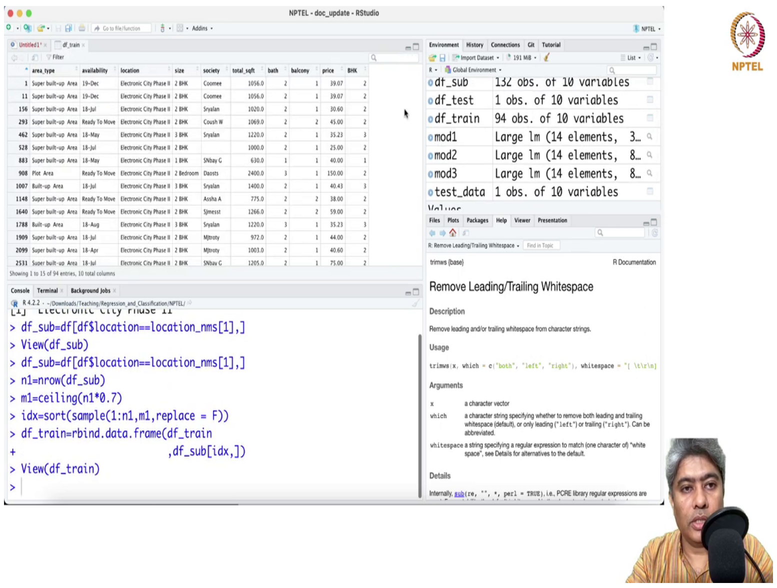
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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)



	area_type	availability	location	size	society	total_sqft	bath	balcony	price	BHK
1	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056.0	2	1	39.07	2
11	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056.0	2	1	39.07	2
156	Super built-up Area	18-Jul	Electronic City Phase II	2 BHK	Sryalan	1020.0	2	1	30.60	2
293	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Coush W	1069.0	2	2	45.00	2
462	Super built-up Area	18-May	Electronic City Phase II	3 BHK	Sryalan	1220.0	2	1	35.23	3
528	Super built-up Area	18-Jul	Electronic City Phase II	2 BHK		1000.0	2	1	25.00	2
883	Super built-up Area	18-May	Electronic City Phase II	1 BHK	SNbay G	630.0	1	1	40.00	1
908	Plot Area	Ready To Move	Electronic City Phase II	2 Bedroom	Daosts	2400.0	3	1	150.00	2
1007	Built-up Area	18-Jul	Electronic City Phase II	3 BHK	Sryalan	1400.0	2	1	40.43	3
1148	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Assha A	775.0	2	2	38.00	2
1640	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	SJmesst	1266.0	2	2	59.00	2
1788	Built-up Area	18-Aug	Electronic City Phase II	3 BHK	Sryalan	1220.0	3	1	35.23	3
1909	Super built-up Area	18-Jul	Electronic City Phase II	2 BHK	MJtroty	972.0	2	1	44.00	2
2099	Super built-up Area	18-Apr	Electronic City Phase II	2 BHK	MJtroty	1003.0	2	1	40.60	2
2531	Super built-up Area	18-Jul	Electronic City Phase II	2 BHK	SNbay G	1205.0	2	1	75.00	2

Showing 1 to 15 of 94 entries, 10 total columns

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

```
R 4.2.2 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
[1] "Electronic City Phase II"
> df_sub=df[df$location==location_nms[1],]
> View(df_sub)
> df_sub=df[df$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)
>
```

R: Remove Leading/Trailing Whitespace

trimws {base} R Documentation

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

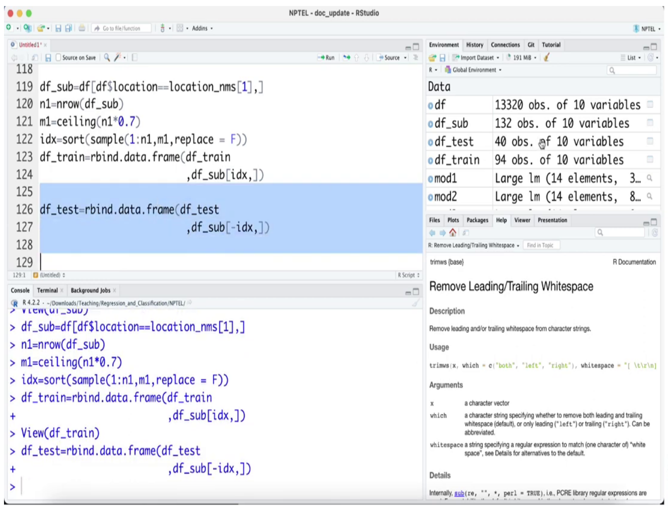
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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

(Refer Slide Time: 47:31)



```
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,])

df_test=rbind.data.frame(df_test
                          ,df_sub[-idx,])

```

```
> 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,])
>
```

```
Data
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…
```

trimws {base} R Documentation

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 a character vector

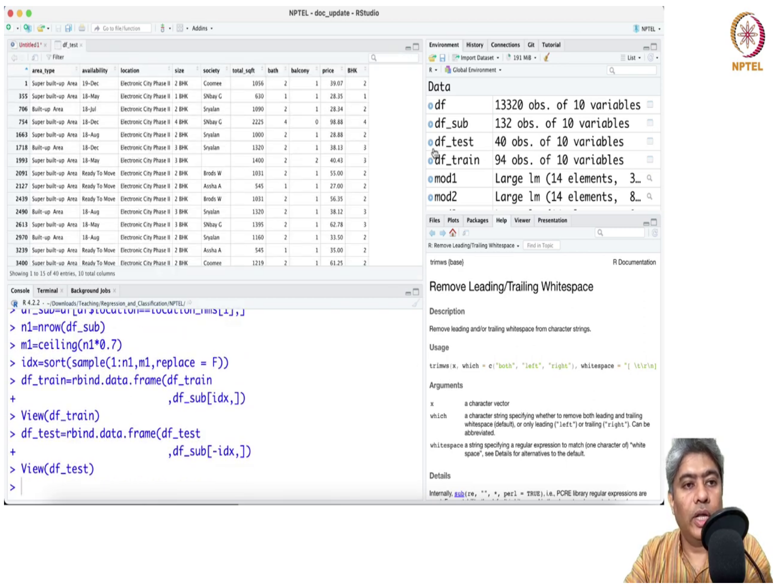
which a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.

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

Details

Internally, `sub(re, "", *, perl = TRUE)`, i.e., PCRE library regular expressions are

(Refer Slide Time: 47:42)



	area_type	availability	location	size	society	total_sqft	bath	balcony	price	BHK
1	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2	1	39.07	2
355	Super built-up Area	18-May	Electronic City Phase II	1 BHK	SNbay G	630	1	1	28.35	1
706	Built-up Area	18-Jul	Electronic City Phase II	2 BHK	Sryalan	1090	2	1	28.34	2
754	Super built-up Area	18-Dec	Electronic City Phase II	4 BHK	SNbay G	2225	4	0	98.88	4
1663	Super built-up Area	18-Aug	Electronic City Phase II	2 BHK	Sryalan	1000	2	1	28.88	2
1718	Built-up Area	18-Dec	Electronic City Phase II	3 BHK	Sryalan	1320	2	1	38.13	3
1993	Super built-up Area	18-May	Electronic City Phase II	3 BHK		1400	2	2	40.43	3
2091	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Brods W	1031	2	1	55.00	2
2127	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Assha A	545	1	1	27.00	2
2439	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Brods W	1031	2	1	56.35	2
2490	Built-up Area	18-Aug	Electronic City Phase II	3 BHK	Sryalan	1320	2	1	38.12	3
2613	Super built-up Area	18-May	Electronic City Phase II	3 BHK	SNbay G	1395	2	1	62.78	3
2970	Built-up Area	18-Aug	Electronic City Phase II	2 BHK	Sryalan	1160	2	1	33.50	2
3239	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Assha A	545	1	1	35.00	2
3400	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Coomee	1219	2	1	61.25	2

Showing 1 to 15 of 40 entries, 10 total columns

```
> 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,])
> View(df_test)
>
```

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…

trimws {base} R Documentation

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

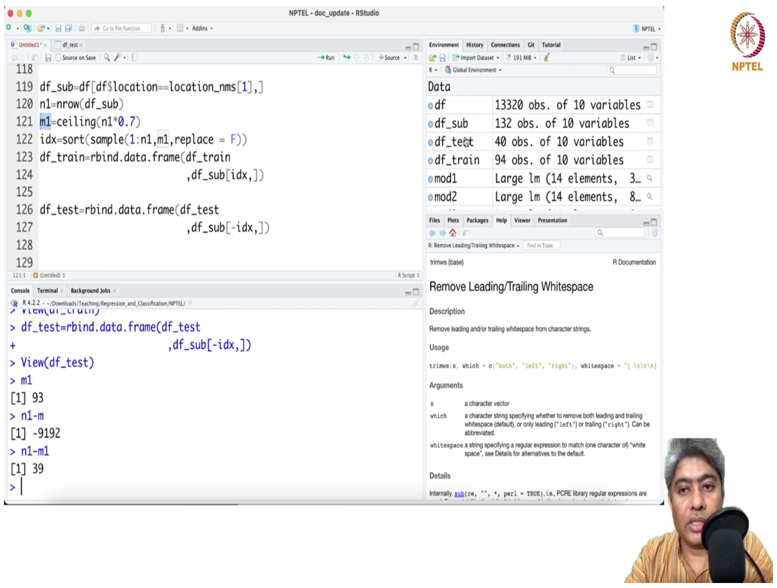
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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)



```
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,])

df_test=rbind.data.frame(df_test
                         ,df_sub[-idx,])
```

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

Data	
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…

trimws {base} R Documentation

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

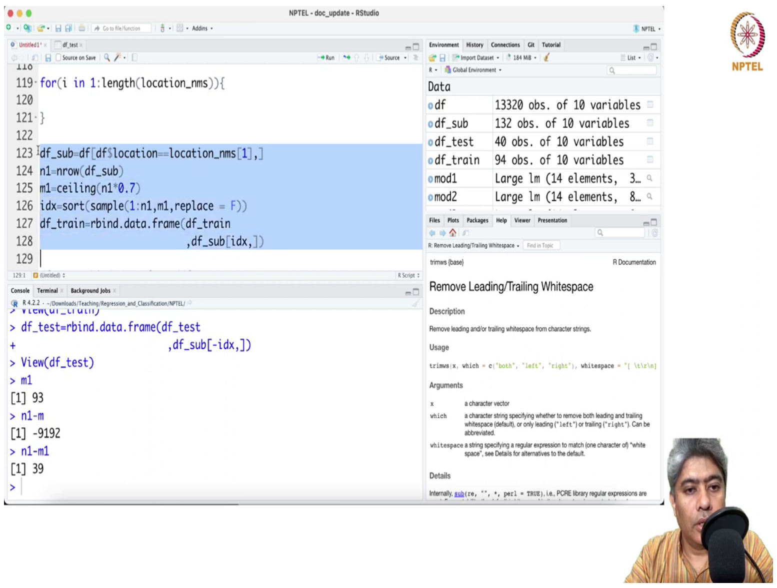
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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)



```
for(i in 1:length(location_nms)){

}

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,])
```

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

```
Data
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…
```

trimws {base} R Documentation

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

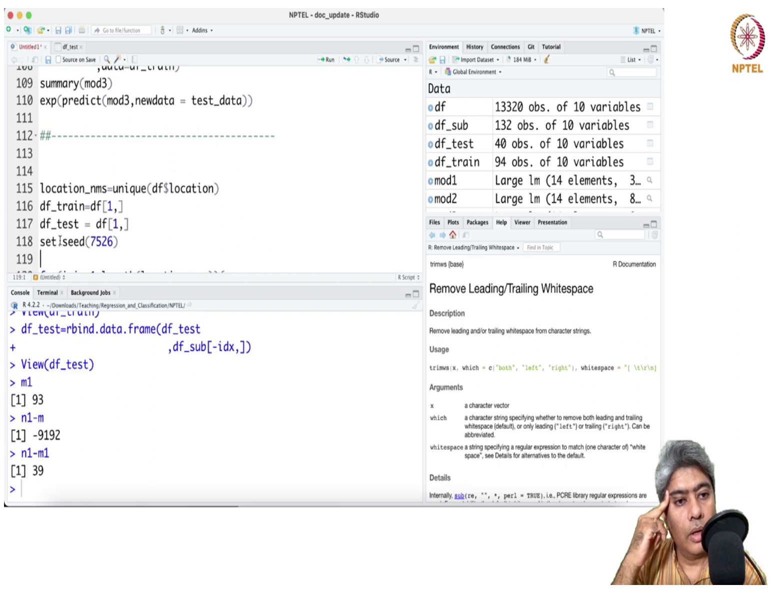
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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)



```
summary(mod3)
exp(predict(mod3,newdata = test_data))

##---------------------------------------


location_nms=unique(df$location)
df_train=df[1,]
df_test = df[1,]
set.seed(7526)

```

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

```
Data
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…
```

trimws {base} R Documentation

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 a character vector

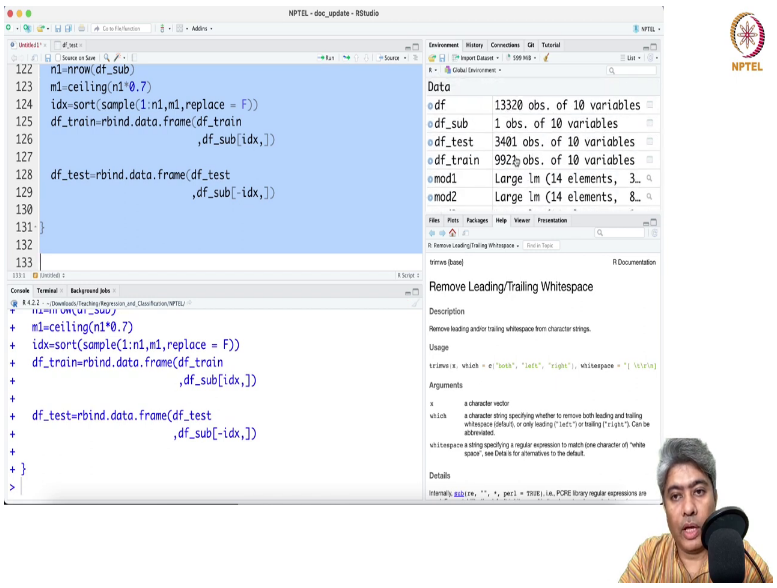
which a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.

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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

(Refer Slide Time: 49:28)



```
  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,])

  df_test=rbind.data.frame(df_test
                           ,df_sub[-idx,])

}

```

```
+   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,])
+
+ }
>
```

```
Data
df        13320 obs. of 10 variables
df_sub    1 obs. of 10 variables
df_test   3401 obs. of 10 variables
df_train  9921 obs. of 10 variables
mod1      Large lm (14 elements, 3…
mod2      Large lm (14 elements, 8…
```

trimws {base} R Documentation

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

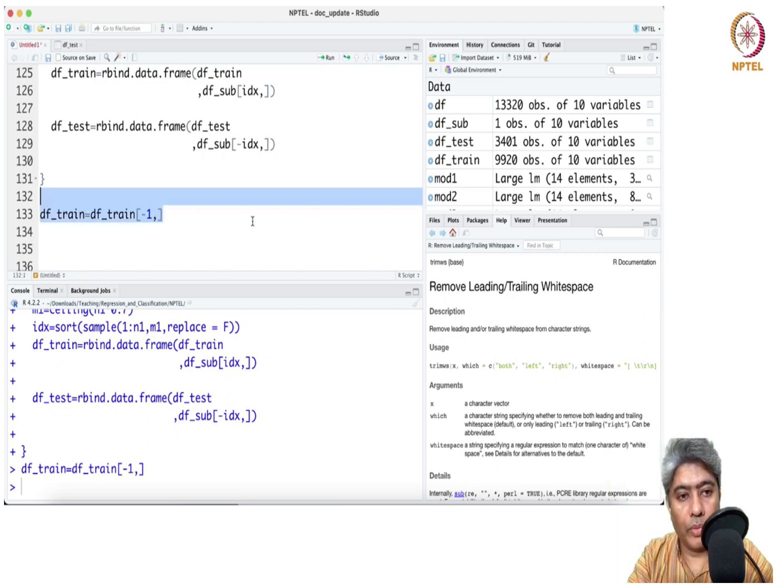
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, `sub(re, "", *, perl = TRUE)`, i.e., PCRE library regular expressions are

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)



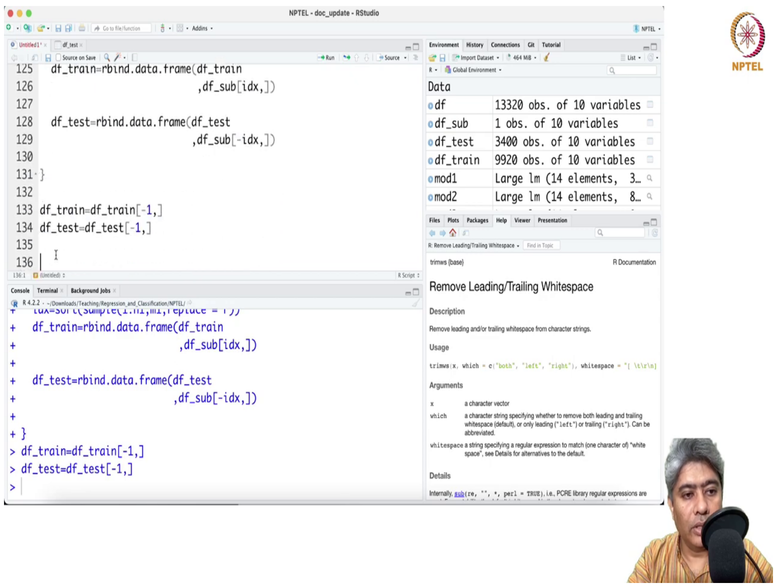
```
df_train=rbind.data.frame(df_train
                          ,df_sub[idx,])

df_test=rbind.data.frame(df_test
                          ,df_sub[-idx,])

}

df_train=df_train[-1,]
```

```
+   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,]
>
```

```
Data
df        13320 obs. of 10 variables
df_sub    1 obs. of 10 variables
df_test   3401 obs. of 10 variables
df_train  9920 obs. of 10 variables
mod1      Large lm (14 elements, 3…
mod2      Large lm (14 elements, 8…
```

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)



```r
  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,]
df_test=df_test[-1,]

```

```
+   idx=sort(sample(1:n1,m1,replace = T))
+   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,]
> df_test=df_test[-1,]
>
```

```
Data
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…
```

trimws {base} R Documentation

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

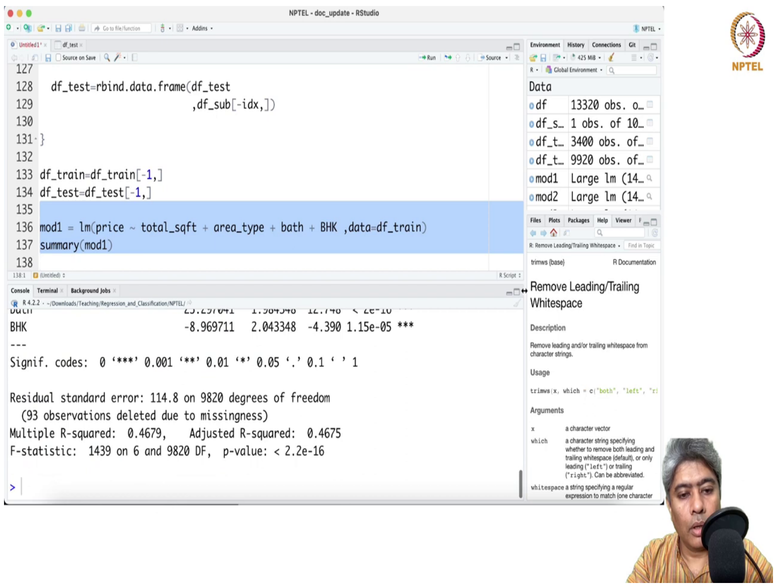
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are

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

(Refer Slide Time: 50:15)

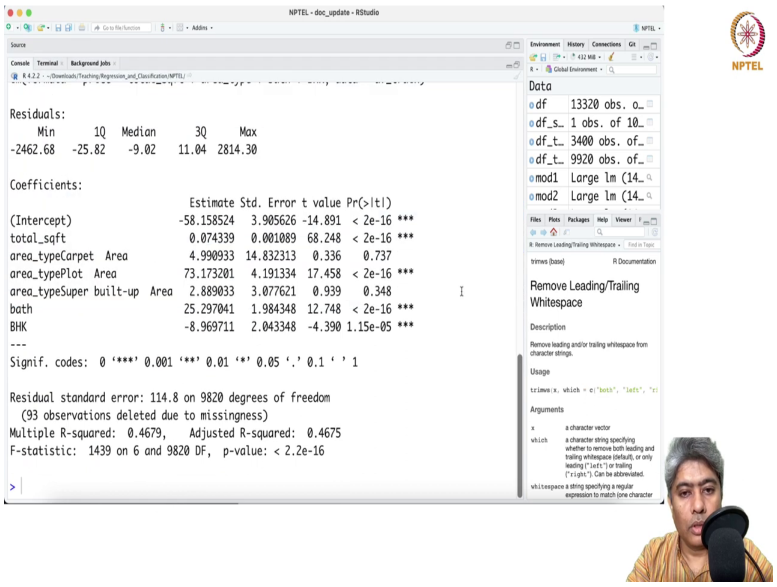


```
  df_test=rbind.data.frame(df_test
                           ,df_sub[-idx,])

}

df_train=df_train[-1,]
df_test=df_test[-1,]

mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
summary(mod1)
```

```
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

>
```

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

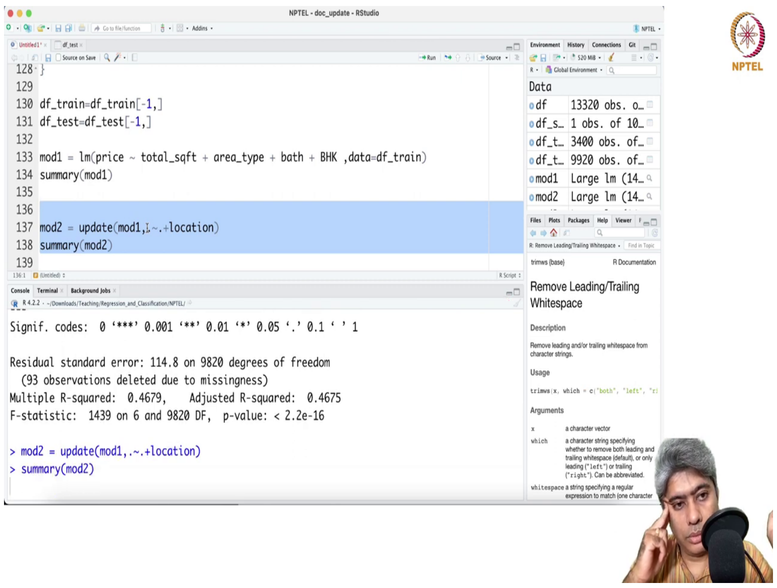
(Refer Slide Time: 50:25)



```
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

>
```

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)

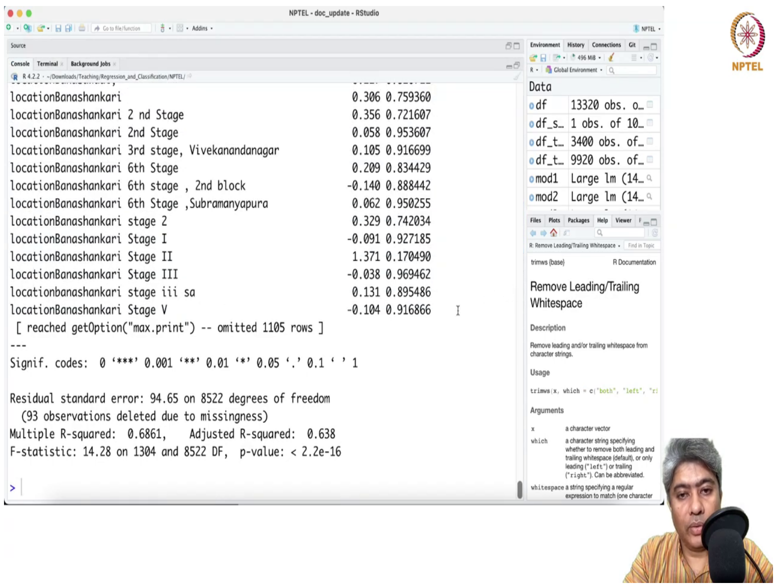


```r
}

df_train=df_train[-1,]
df_test=df_test[-1,]

mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
summary(mod1)


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

```
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,.~.+location)
> summary(mod2)
```

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

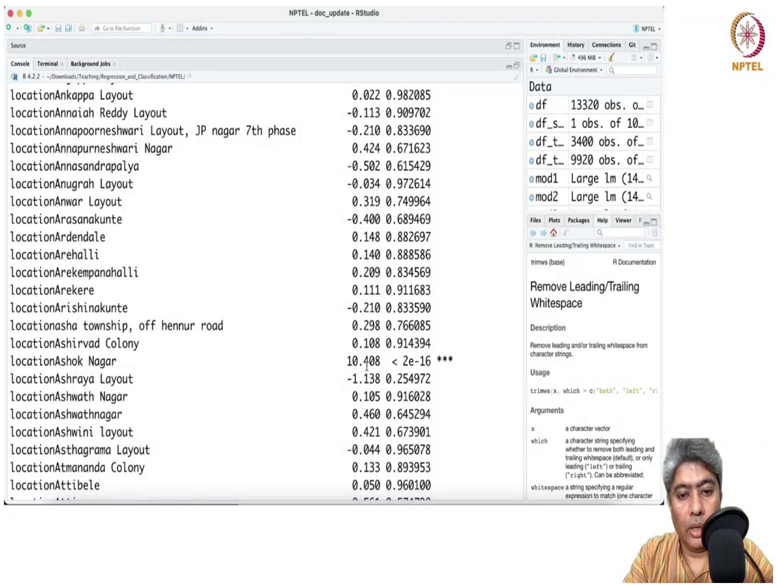
(Refer Slide Time: 51:04)



```
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

>
```

Now, it is 63.8 percent accuracy 63.8 percent accuracy.

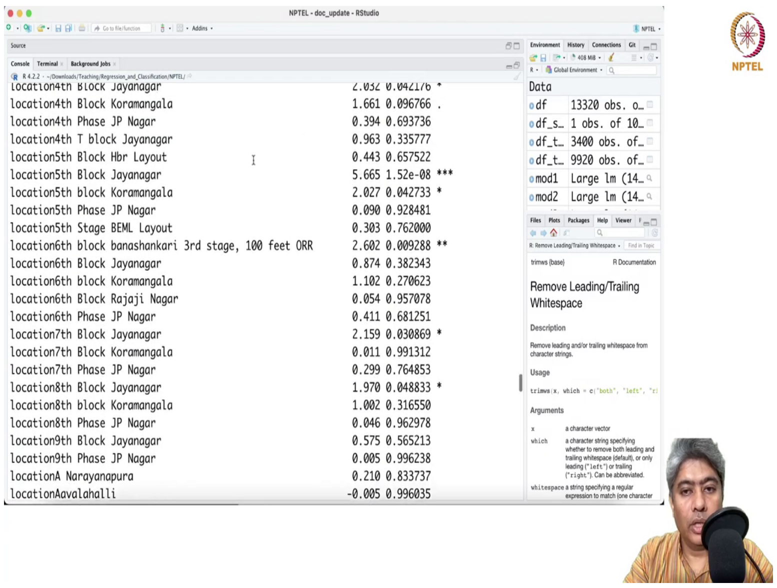
(Refer Slide Time: 51:13)



```
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
locationArdendale                                      0.148 0.882697
locationArehalli                                       0.140 0.888586
locationArekempanahalli                                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 0.645294
locationAshwini layout                                 0.421 0.673901
locationAsthagrama Layout                             -0.044 0.965078
locationAtmananda Colony                               0.133 0.893953
locationAttibele                                       0.050 0.960100
```

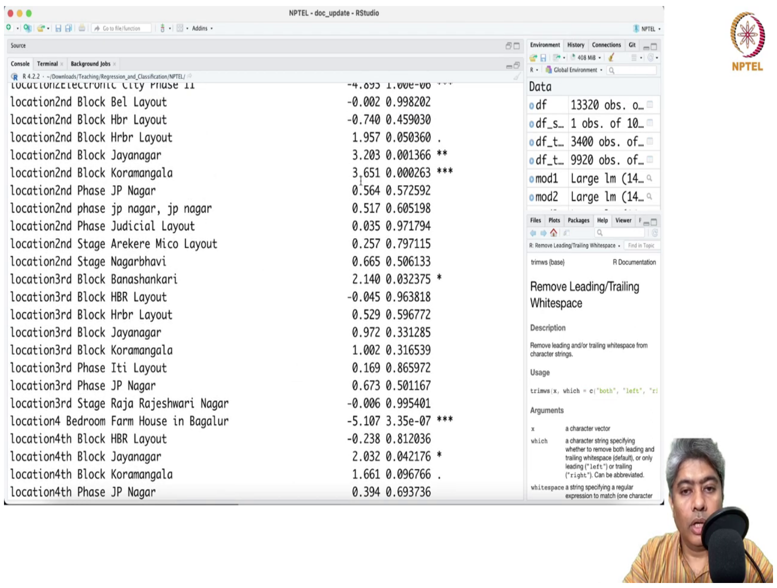
And Ashok Nagar has a very high p value.

(Refer Slide Time: 51:21)



```
location4th Block Jayanagar                               2.032 0.042176 *
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
locationAavalahalli                                      -0.005 0.996035
```

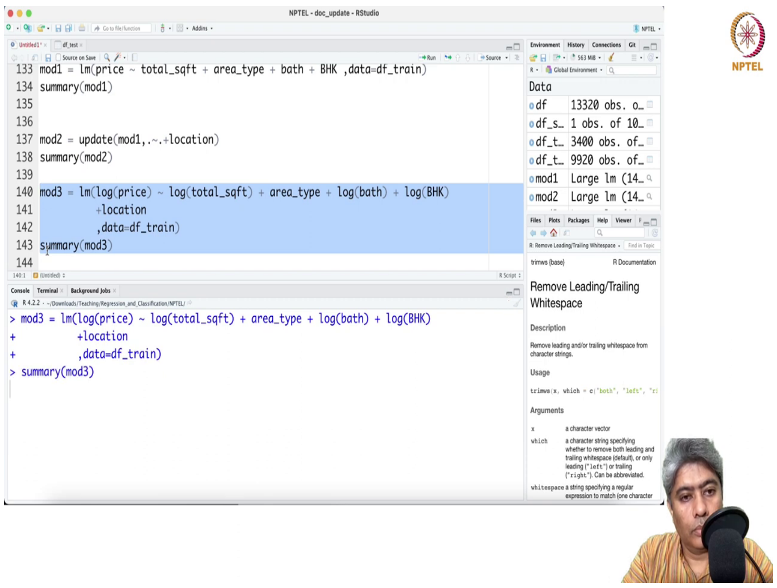
(Refer Slide Time: 51:26)



```
location2Electronic City Phase II                   -4.895 1.00e-06 ***
location2nd Block Bel Layout                        -0.002 0.998202
location2nd Block Hbr Layout                        -0.740 0.459030
location2nd Block Hrbr Layout                        1.957 0.050360 .
location2nd Block Jayanagar                          3.203 0.001366 **
location2nd Block Koramangala                        3.651 0.000263 ***
location2nd Phase JP Nagar                           0.564 0.572592
location2nd phase jp nagar, jp nagar                 0.517 0.605198
location2nd Phase Judicial Layout                    0.035 0.971794
location2nd Stage Arekere Mico Layout                0.257 0.797115
location2nd Stage Nagarbhavi                         0.665 0.506133
location3rd Block Banashankari                       2.140 0.032375 *
location3rd Block HBR Layout                        -0.045 0.963818
location3rd Block Hrbr Layout                        0.529 0.596772
location3rd Block Jayanagar                          0.972 0.331285
location3rd Block Koramangala                        1.002 0.316539
location3rd Phase Iti Layout                         0.169 0.865972
location3rd Phase JP Nagar                           0.673 0.501167
location3rd Stage Raja Rajeshwari Nagar             -0.006 0.995401
location4 Bedroom Farm House in Bagalur             -5.107 3.35e-07 ***
location4th Block HBR Layout                        -0.238 0.812036
location4th Block Jayanagar                          2.032 0.042176 *
location4th Block Koramangala                        1.661 0.096766 .
location4th Phase JP Nagar                           0.394 0.693736
```

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)



```
mod1 = lm(price ~ total_sqft + area_type + bath + BHK ,data=df_train)
summary(mod1)


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

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

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

```
Data
df      13320 obs. o…
df_s…   1 obs. of 10…
df_t…   3400 obs. of…
df_t…   9920 obs. of…
mod1    Large lm (14…
mod2    Large lm (14…
```

Remove Leading/Trailing Whitespace

Description

Remove leading and/or trailing whitespace from character strings.

Usage

`trimws(x, which = c("both", "left", "ri`

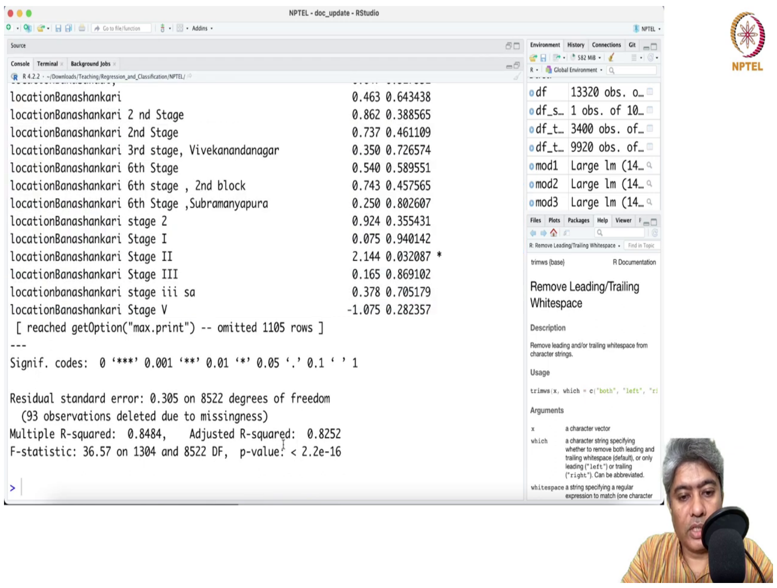
Arguments

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

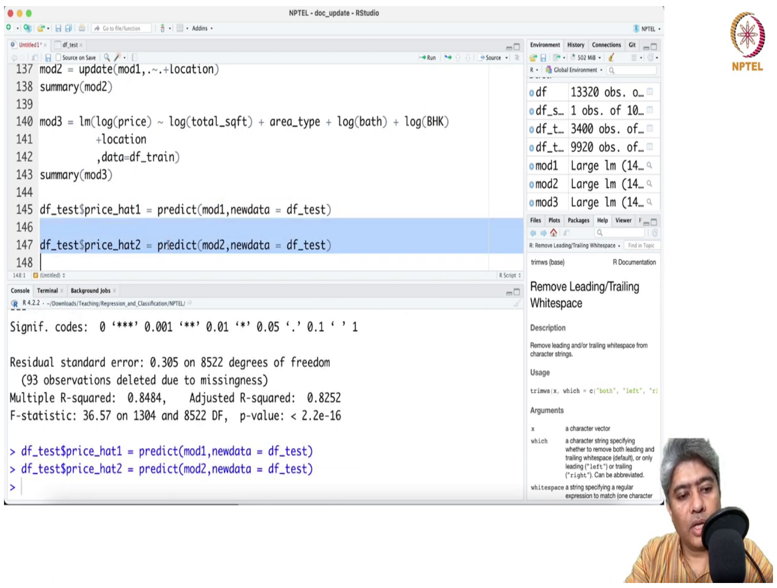
(Refer Slide Time: 52:13)



```
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
 [ reached getOption("max.print") -- omitted 1105 rows ]
---
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)



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

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

df_test$price_hat1 = predict(mod1,newdata = df_test)

df_test$price_hat2 = predict(mod2,newdata = df_test)

```

```
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)
>
```

```
df     13320 obs. o…
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…
```

Remove Leading/Trailing Whitespace

Description

Remove leading and/or trailing whitespace from character strings.

Usage

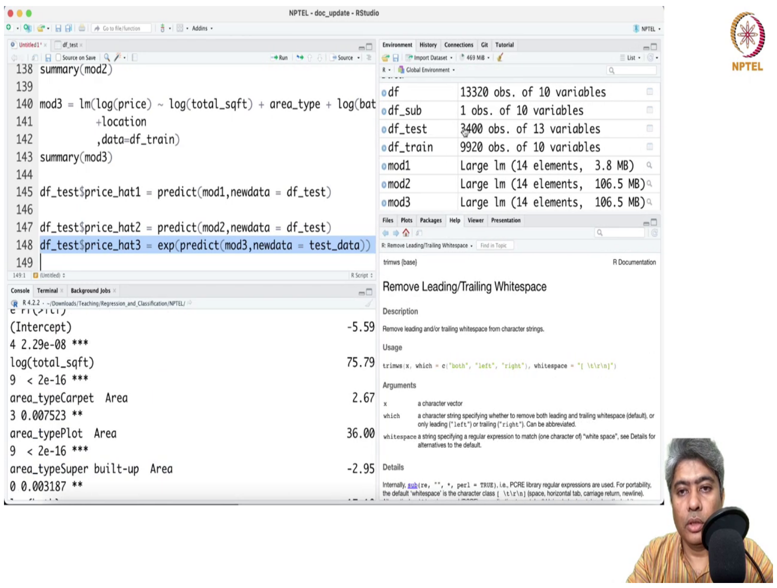
`trimws(x, which = c("both", "left", "r`

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 (one 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)



```
summary(mod2)

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

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))
```

```
(Intercept)                                   -5.59
4 2.29e-08 ***
log(total_sqft)                               75.79
9  < 2e-16 ***
area_typeCarpet  Area                          2.67
3 0.007523 **
area_typePlot  Area                           36.00
9  < 2e-16 ***
area_typeSuper built-up  Area                 -2.95
0 0.003187 **
```

```
df        13320 obs. of 10 variables
df_sub    1 obs. of 10 variables
df_test   3400 obs. of 13 variables
df_train  9920 obs. of 10 variables
mod1      Large lm (14 elements,   3.8 MB)
mod2      Large lm (14 elements, 106.5 MB)
mod3      Large lm (14 elements, 106.5 MB)
```

trimws {base} R Documentation

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

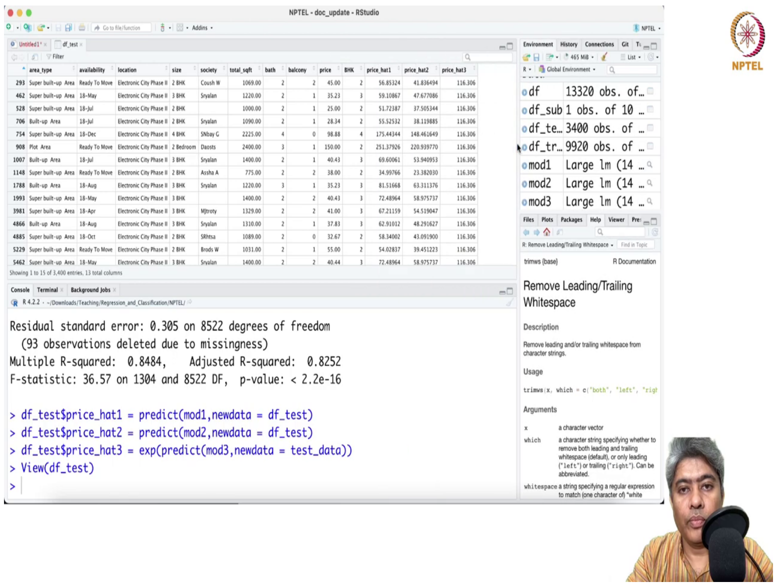
x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for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

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are used. For portability, the default 'whitespace' is the character class [\t\r\n] (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)



	area_type	availability	location	size	society	total_sqft	bath	balcony	price	BHK	price_hat1	price_hat2	price_hat3
293	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Coush W	1069.00	2	2	45.00	2	56.85324	41.836494	116.306
462	Super built-up Area	18-May	Electronic City Phase II	3 BHK	Sryalan	1220.00	2	1	35.23	3	59.10867	47.677086	116.306
528	Super built-up Area	18-Jul	Electronic City Phase II	2 BHK		1000.00	2	1	25.00	2	51.72387	37.505344	116.306
706	Built-up Area	18-Jul	Electronic City Phase II	2 BHK	Sryalan	1090.00	2	1	28.34	2	55.52532	38.119885	116.306
754	Super built-up Area	18-Dec	Electronic City Phase II	4 BHK	SNbay G	2225.00	4	0	98.88	4	175.44344	148.461649	116.306
908	Plot Area	Ready To Move	Electronic City Phase II	2 Bedroom	Daosts	2400.00	3	1	150.00	2	251.37926	220.939770	116.306
1007	Built-up Area	18-Jul	Electronic City Phase II	3 BHK	Sryalan	1400.00	2	1	40.43	3	69.60061	53.940953	116.306
1148	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Assha A	775.00	2	2	38.00	2	34.99766	23.382030	116.306
1788	Built-up Area	18-Aug	Electronic City Phase II	3 BHK	Sryalan	1220.00	3	1	35.23	3	81.51668	63.311376	116.306
1993	Super built-up Area	18-May	Electronic City Phase II	3 BHK		1400.00	2	2	40.43	3	72.48964	58.975737	116.306
3981	Super built-up Area	18-Apr	Electronic City Phase II	3 BHK	MJtroty	1329.00	2	2	41.00	3	67.21159	54.519047	116.306
4866	Built-up Area	18-Aug	Electronic City Phase II	3 BHK	Sryalan	1310.00	2	1	37.83	3	62.91012	48.291627	116.306
4885	Super built-up Area	18-Oct	Electronic City Phase II	2 BHK	SRhtsa	1089.00	2	0	32.67	2	58.34002	43.091900	116.306
5229	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Brods W	1031.00	2	1	55.00	2	54.02837	39.451223	116.306
5462	Super built-up Area	18-May	Electronic City Phase II	3 BHK	Sryalan	1400.00	2	2	40.44	3	72.48964	58.975737	116.306

Showing 1 to 15 of 3,400 entries, 13 total columns

```
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)
> df_test$price_hat3 = exp(predict(mod3,newdata = test_data))
> View(df_test)
>
```

```
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 …
```

Remove Leading/Trailing Whitespace

Description

Remove leading and/or trailing whitespace from character strings.

Usage

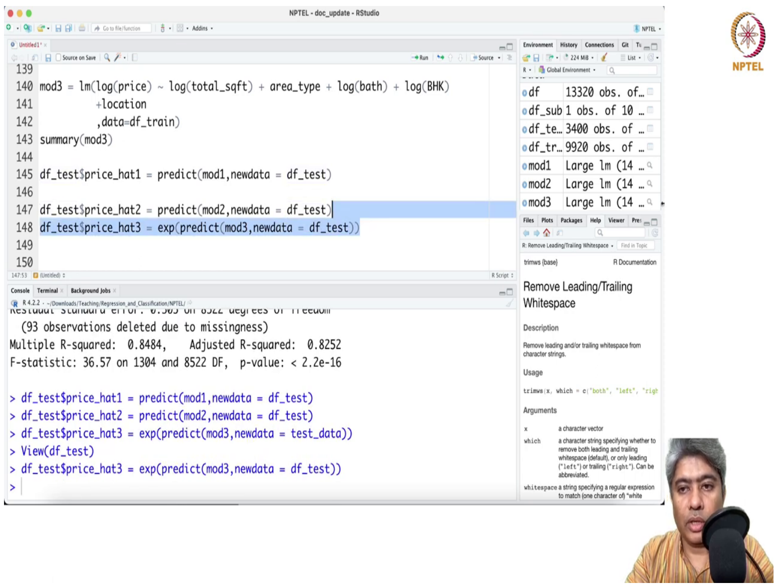
`trimws(x, which = c("both", "left", "rig`

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

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)



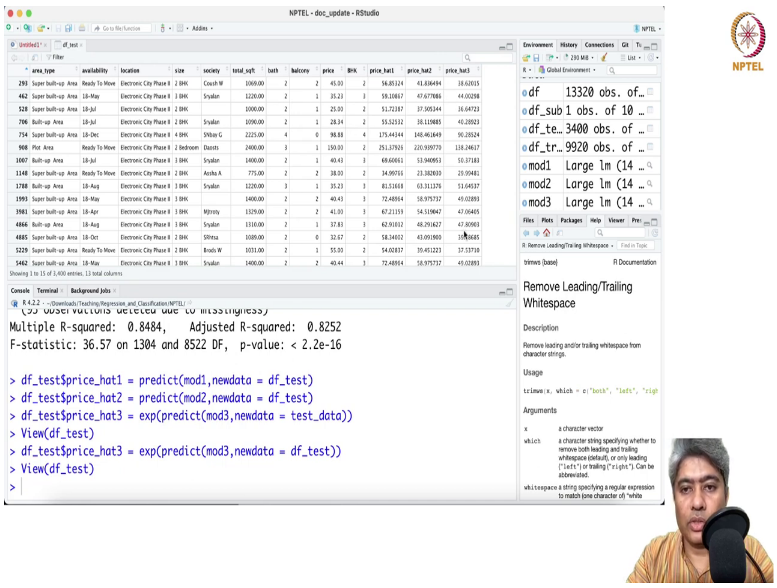
```r
mod3 = lm(log(price) ~ log(total_sqft) + area_type + log(bath) + log(BHK)
          +location
          ,data=df_train)
summary(mod3)

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))
```

```
  (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)
> df_test$price_hat3 = exp(predict(mod3,newdata = test_data))
> View(df_test)
> df_test$price_hat3 = exp(predict(mod3,newdata = df_test))
>
```

```
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 …
```

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)



	area_type	availability	location	size	society	total_sqft	bath	balcony	price	BHK	price_hat1	price_hat2	price_hat3
293	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Coush W	1069.00	2	2	45.00	2	56.85324	41.836494	38.62015
462	Super built-up Area	18-May	Electronic City Phase II	3 BHK	Sryalan	1220.00	2	1	35.23	3	59.10867	47.677086	44.00298
528	Super built-up Area	18-Jul	Electronic City Phase II	2 BHK		1000.00	2	1	25.00	2	51.72387	37.505344	36.64723
706	Built-up Area	18-Jul	Electronic City Phase II	2 BHK	Sryalan	1090.00	2	1	28.34	2	55.52532	38.119885	40.28923
754	Super built-up Area	18-Dec	Electronic City Phase II	4 BHK	SNbay G	2225.00	4	0	98.88	4	175.44344	148.461649	90.28524
908	Plot Area	Ready To Move	Electronic City Phase II	2 Bedroom	Daosts	2400.00	3	1	150.00	2	251.37926	220.939770	138.24617
1007	Built-up Area	18-Jul	Electronic City Phase II	3 BHK	Sryalan	1400.00	2	1	40.43	3	69.60061	53.940953	50.37183
1148	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Assha A	775.00	2	2	38.00	2	34.99766	23.382030	29.99481
1788	Built-up Area	18-Aug	Electronic City Phase II	3 BHK	Sryalan	1220.00	3	1	35.23	3	81.51668	63.311376	51.64537
1993	Super built-up Area	18-May	Electronic City Phase II	3 BHK		1400.00	2	2	40.43	3	72.48964	58.975737	49.02893
3981	Super built-up Area	18-Apr	Electronic City Phase II	3 BHK	MJtroty	1329.00	2	2	41.00	3	67.21159	54.519047	47.06405
4866	Built-up Area	18-Aug	Electronic City Phase II	3 BHK	Sryalan	1310.00	2	1	37.83	3	62.91012	48.291627	47.80903
4885	Super built-up Area	18-Oct	Electronic City Phase II	2 BHK	SRhtsa	1089.00	2	0	32.67	2	58.34002	43.091900	39.18685
5229	Super built-up Area	Ready To Move	Electronic City Phase II	2 BHK	Brods W	1031.00	2	1	55.00	2	54.02837	39.451223	37.53710
5462	Super built-up Area	18-May	Electronic City Phase II	3 BHK	Sryalan	1400.00	2	2	40.44	3	72.48964	58.975737	49.02893

Showing 1 to 15 of 3,400 entries, 13 total columns

```
  (95 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)
> 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)
>
```

```
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 …
```

trimws {base} R Documentation

Remove Leading/Trailing Whitespace

Description

Remove leading and/or trailing whitespace from character strings.

Usage

trimws(x, which = c("both", "left", "righ

Arguments

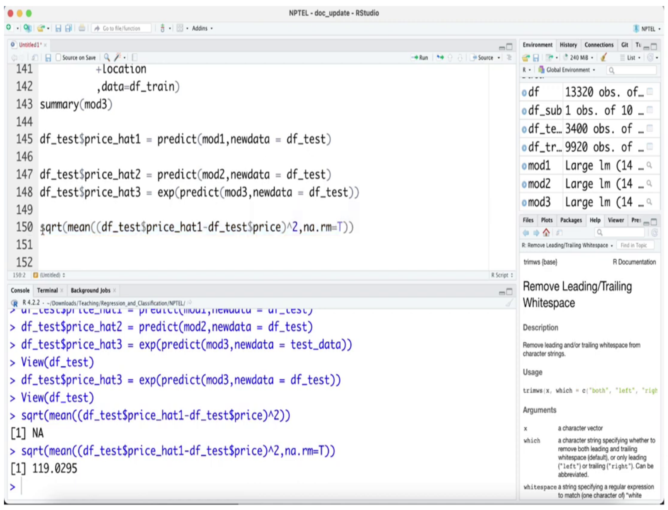
x a character vector

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

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)

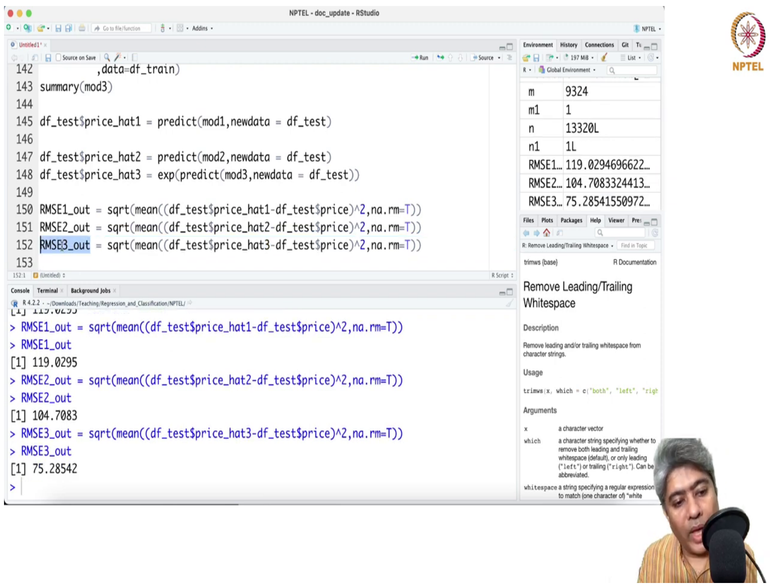


```
         +location
         ,data=df_train)
summary(mod3)

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))

sqrt(mean((df_test$price_hat1-df_test$price)^2,na.rm=T))
```

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

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)



```
             ,data=df_train)
summary(mod3)

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))

RMSE1_out = sqrt(mean((df_test$price_hat1-df_test$price)^2,na.rm=T))
RMSE2_out = sqrt(mean((df_test$price_hat2-df_test$price)^2,na.rm=T))
RMSE3_out = sqrt(mean((df_test$price_hat3-df_test$price)^2,na.rm=T))
```

```
> 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
>
```

m	9324
m1	1
n	13320L
n1	1L
RMSE1…	119.0294696622…
RMSE2…	104.7083324413…
RMSE3…	75.28541550972…

Remove Leading/Trailing Whitespace

Description

Remove leading and/or trailing whitespace from character strings.

Usage

`trimws(x, which = c("both", "left", "righ`

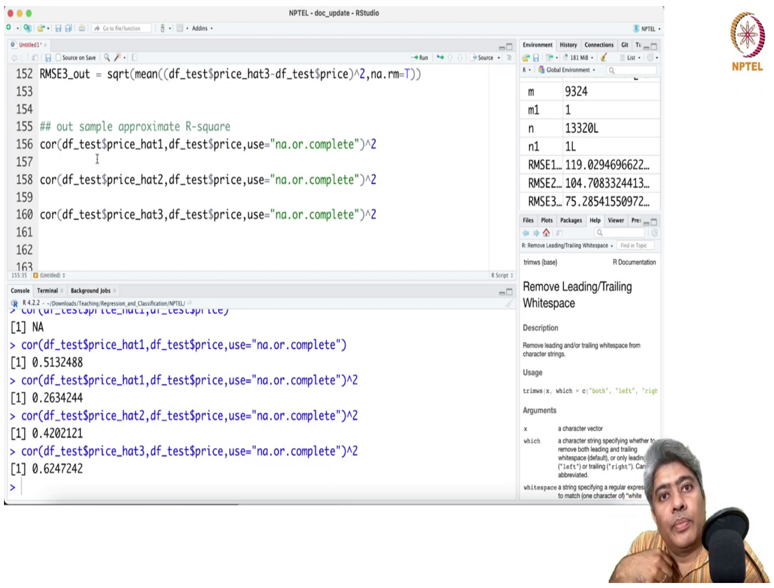
Arguments

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

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)



```
RMSE3_out = sqrt(mean((df_test$price_hat3-df_test$price)^2,na.rm=T))


## out sample approximate R-square
cor(df_test$price_hat1,df_test$price,use="na.or.complete")^2

cor(df_test$price_hat2,df_test$price,use="na.or.complete")^2

cor(df_test$price_hat3,df_test$price,use="na.or.complete")^2
```

```
[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
>
```

m	9324
m1	1
n	13320L
n1	1L
RMSE1…	119.0294696622…
RMSE2…	104.7083324413…
RMSE3…	75.28541550972…

trimws {base} R Documentation

Remove Leading/Trailing Whitespace

Description

Remove leading and/or trailing whitespace from character strings.

Usage

trimws(x, which = c("both", "left", "righ

Arguments

x a character vector

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

whitespace a string specifying a regular expres to match (one character of) "white

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.