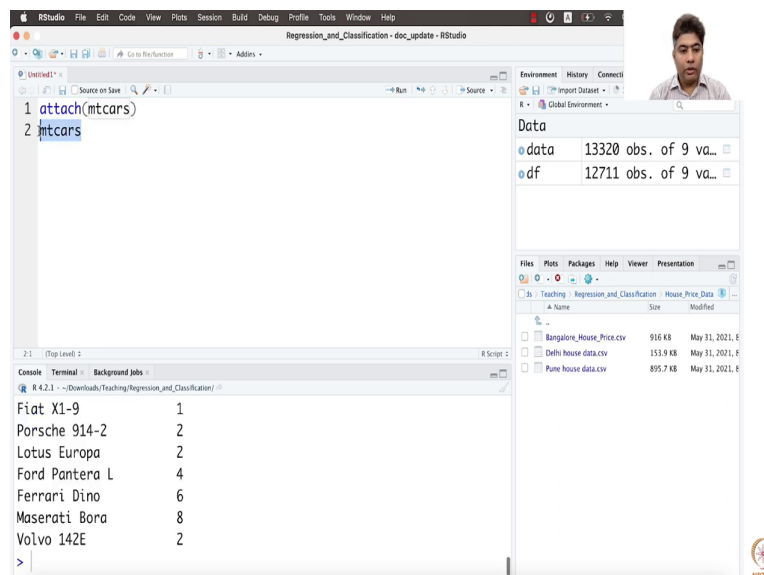


Predictive Analytics - Regression and Classification
Prof. Sourish Das
Department of Mathematics
Chennai Mathematical Institute

Lecture - 04
Hands-on with R Part - 1

Welcome back. In this part of the lecture video, we are going to check out how to implement regression using R, particularly the OLS estimator, how to implement those OLS estimator in R.

(Refer Slide Time: 00:36)



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

```
1 attach(mtcars)
2 mtcars
```

The console output shows the following data:

Fiat X1-9	1
Porsche 914-2	2
Lotus Europa	2
Ford Pantera L	4
Ferrari Dino	6
Maserati Bora	8
Volvo 142E	2

The Environment pane on the right shows the following data objects:

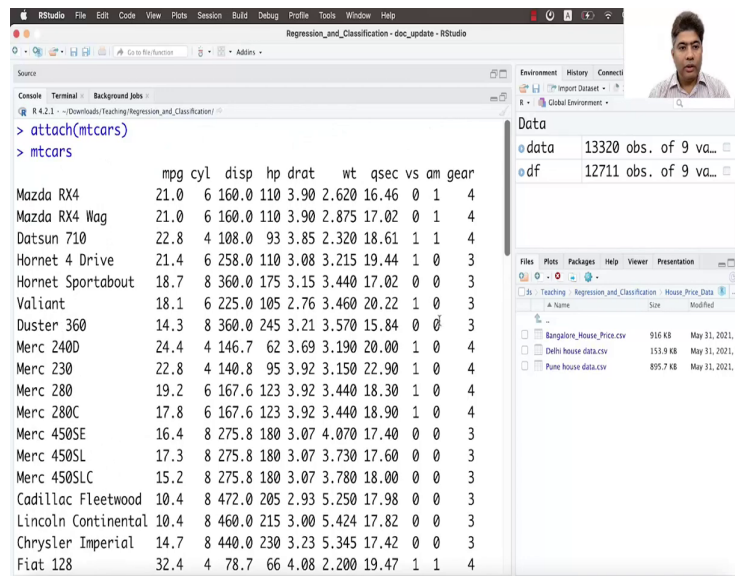
data	13320 obs. of 9 va...
df	12711 obs. of 9 va...

The Files pane on the right shows the following files:

Name	Size	Modified
Bangalore_House_Price.csv	916 KB	May 31, 2021, E
Delhi house data.csv	153.9 KB	May 31, 2021, E
Pune house data.csv	895.7 KB	May 31, 2021, E

So, now, we are going to look into this R, we are going to open R script and one good thing about R is this mtcars data set is already available in R. So, you just say mtcars and run this as if you, if here is a run I can just put a run there. And then if I just say mtcars, then and run it.

(Refer Slide Time: 01:16)



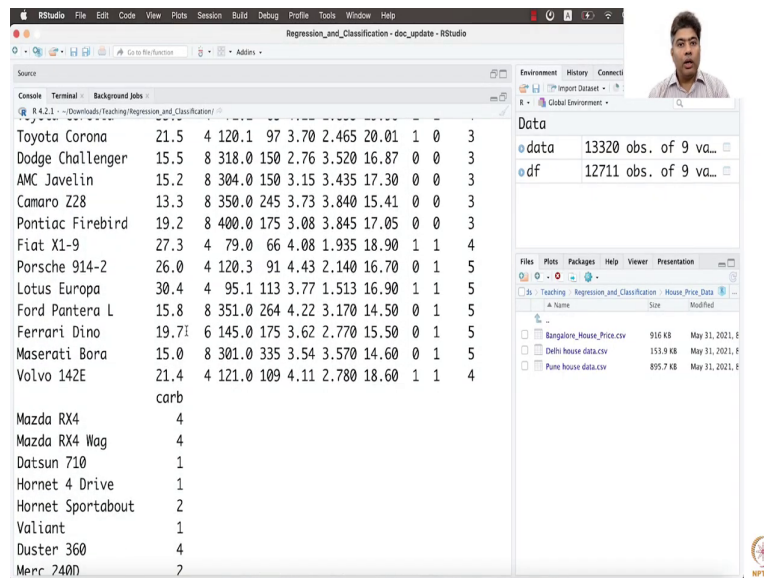
The screenshot shows the RStudio interface with the following components:

- Console:** Shows the execution of `attach(mtcars)` and `mtcars`, resulting in a printed table of car specifications.
- Data Viewer:** Shows the loaded data with 13320 observations and 9 variables.
- Files Panel:** Lists files in the current directory, including `Bangalore_House_Price.csv`, `Delhi_House_data.csv`, and `Pune_House_data.csv`.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4

So, I will get the all the data here. You can see, you know miles per gallon cylinder displacement horsepower all these variables. And there are 32 models or each models we have the specifications.

(Refer Slide Time: 01:33)

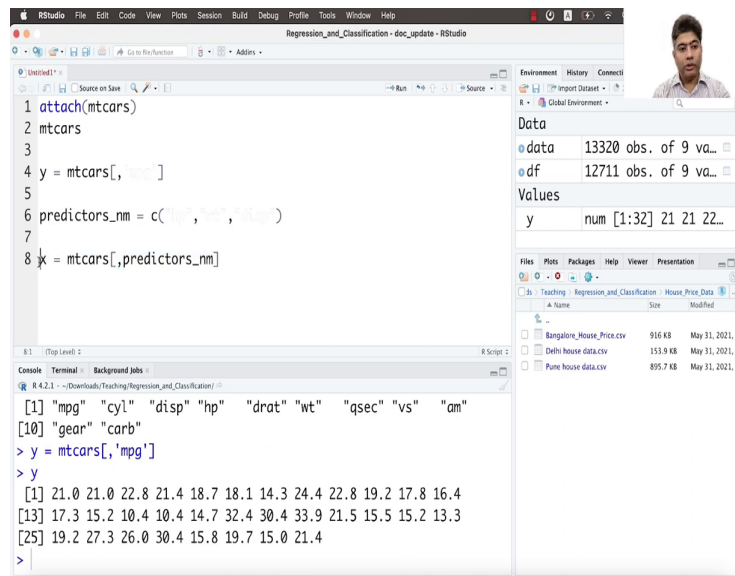


The screenshot shows the RStudio interface with a console window displaying the following data:

Model	mpg	displacement	horsepower	weight	acceleration	year	origin	carb
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1 0 3
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0 0 3
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0 0 3
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0 0 3
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0 0 3
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1 1 4
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0 1 5
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1 1 5
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0 1 5
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0 1 5
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0 1 5
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1 1 4
carb								
Mazda RX4		4						
Mazda RX4 Wag		4						
Datsun 710		1						
Hornet 4 Drive		1						
Hornet Sportabout		2						
Valiant		1						
Duster 360		4						
Merc 240D		7						

Now, what I am, just let me clean the console. Now, what we want is we want to keep miles per gallon in the y vector.

(Refer Slide Time: 01:59)



```
1 attach(mtcars)
2 mtcars
3
4 y = mtcars[, 'mpg']
5
6 predictors_nm = c('hp', 'wt', 'disp')
7
8 x = mtcars[,predictors_nm]
```

Environment History Connect
R • Global Environment •

Data
data 13320 obs. of 9 va...
df 12711 obs. of 9 va...

Values
y num [1:32] 21 21 22...

Files Plots Packages Help Viewer Presentation
Teaching Regression_and_Classification House_Price_Data

Name Size Modified
Bangalore_House_Price.csv 916 KB May 31, 2021, 6
Delhi_house_data.csv 153.9 KB May 31, 2021, 6
Pune_house_data.csv 895.7 KB May 31, 2021, 6

R 4.2.1 - Downloads/Teaching/Regression_and_Classification/

```
[1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am"
[10] "gear" "carb"
> y = mtcars[, 'mpg']
> y
[1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4
[13] 17.3 15.2 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3
[25] 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4
>
```

So, before we can before do that, [FL] we can let us pick this put this things mtcars and then let us call it mpg. So, we can just look into the col names of mtcars. So, this is the mpg that has put it up here and run this and (Refer Time: 02:30) on this. So, these are the all variables are there in the y variable.

And then what I can do. So, now, what I will do, I will bring predictors name, predictors name and maybe horsepower, horsepower, weight and displacement disp, you know and then we can just write mtcars as with predictor names here. So, we have this property (Refer Time: 03:33), the design (Refer Time: 03:34) sorry; you know we will take it as small x.

(Refer Slide Time: 03:40)

The screenshot shows the RStudio interface with the following code in the script editor:

```
1 attach(mtcars)
2 mtcars
3
4 y = mtcars[, "mpg"]
5
6 predictors_nm = c("hp", "wt", "disp")
7
8 x = mtcars[,predictors_nm]
```

The console output shows the execution of the code:

```
R 4.2.1 -- (64-bit) Teaching Regression and Classification
> object 'predictors_nm' not found
> predictors_nm = c("hp", "wt", "disp")
>
> x = mtcars[,predictors_nm]
> x
```

	hp	wt	disp
Mazda RX4	110	2.620	160.0
Mazda RX4 Wag	110	2.875	160.0
Datsun 710	93	2.320	108.0

The Environment pane on the right shows the following data:

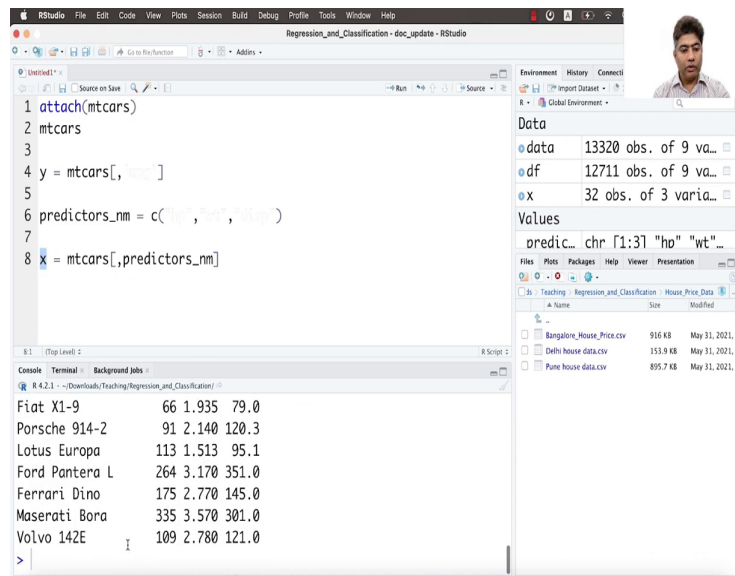
Object	Value
data	13320 obs. of 9 variables
df	12711 obs. of 9 variables
x	32 obs. of 3 variables

The Files pane shows a list of files in the current directory:

Name	Size	Modified
Bangalore_House_Price.csv	916 KB	May 31, 2021, 6:00 AM
Delhi_House_data.csv	153.9 KB	May 31, 2021, 6:00 AM
Pune_House_data.csv	895.7 KB	May 31, 2021, 6:00 AM

And sorry, I need to run this on the thing, and now I got this 3 variables here.

(Refer Slide Time: 03:52)



The screenshot shows the RStudio interface with the following code in the editor:

```
1 attach(mtcars)
2 mtcars
3
4 y = mtcars[, "mpg"]
5
6 predictors_nm = c("wt", "hp", "displ")
7
8 x = mtcars[, predictors_nm]
```

The console output shows the first six rows of the selected columns:

```
Fiat X1-9      66 1.935 79.0
Porsche 914-2  91 2.140 120.3
Lotus Europa   113 1.513  95.1
Ford Pantera L 264 3.170 351.0
Ferrari Dino   175 2.770 145.0
Maserati Bora  335 3.570 301.0
Volvo 142E     109 2.780 121.0
```

The Environment pane on the right shows the following data objects:

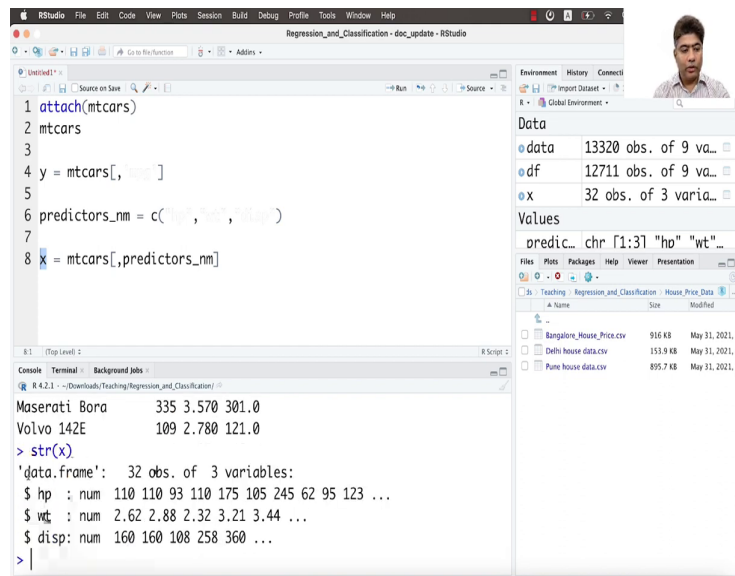
Object	Value
data	13320 obs. of 9 variables
df	12711 obs. of 9 variables
x	32 obs. of 3 variables

The Files pane shows a list of files in the current project:

Name	Size	Modified
Bangalore_House_Price.csv	916 KB	May 31, 2021, 6:00 AM
Delhi_House_data.csv	153.9 KB	May 31, 2021, 6:00 AM
Pune_House_data.csv	895.7 KB	May 31, 2021, 6:00 AM

But remember that these are still at a data frame structure.

(Refer Slide Time: 03:54)



The screenshot shows the RStudio interface with the following code in the editor:

```
1 attach(mtcars)
2 mtcars
3
4 y = mtcars[, "mpg"]
5
6 predictors_nm = c("hp", "wt", "disp")
7
8 x = mtcars[,predictors_nm]
```

The console output shows the result of the code:

```
Maserati Bora      335 3.570 301.0
Volvo 142E        109 2.780 121.0
> str(x)
'data.frame':   32 obs. of  3 variables:
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ disp: num  160 160 108 258 360 ...
> |
```

The Environment pane on the right shows the following data objects:

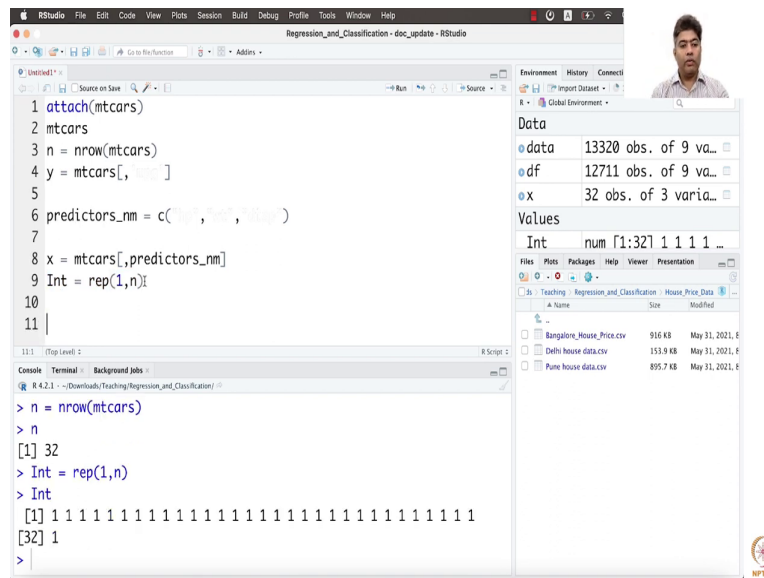
Object	Value
data	13320 obs. of 9 variables
df	12711 obs. of 9 variables
x	32 obs. of 3 variables

The Files pane shows a list of files in the current directory:

Name	Size	Modified
Bangalore_House_Price.csv	916 KB	May 31, 2021, 6
Delhi_House_Price.csv	153.9 KB	May 31, 2021, 6
Pune_House_Price.csv	895.7 KB	May 31, 2021, 6

So, if we just say structure x, you see it is calling data dot frame. So, this is still at a data frame. So, we have to convert it into a matrix. Before that what we have to do? Remember that we have to add a intercept here, ok.

(Refer Slide Time: 04:16)



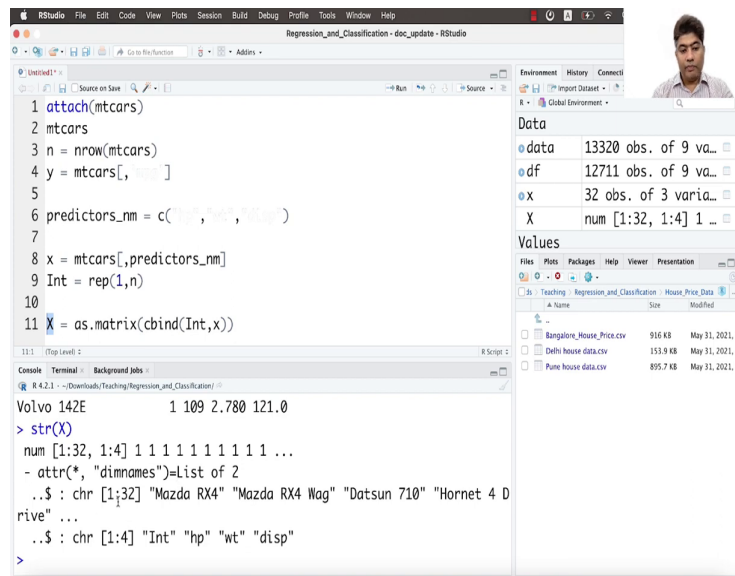
```
1 attach(mtcars)
2 mtcars
3 n = nrow(mtcars)
4 y = mtcars[, "mpg"]
5
6 predictors_nm = c("wt", "qsec", "dis")
7
8 x = mtcars[,predictors_nm]
9 Int = rep(1,n)
10
11
```

```
> n = nrow(mtcars)
> n
[1] 32
> Int = rep(1,n)
> Int
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[32] 1
>
```

The screenshot shows the RStudio interface. The script editor contains the code above. The console shows the execution of the code, resulting in the output shown. The Environment pane on the right shows the 'Data' environment with variables 'data', 'df', and 'x'. The Files pane shows a list of files in the current directory.

So, we will define intercept as replicate 1 comma n, but I do not know what is n. So, before that we have to say n column of mtcars row and in row of mtcars is 32. So, we will put that in n, so n as 32. So, now if I just run this, so I have intercept as 1.

(Refer Slide Time: 05:10)



The screenshot shows the RStudio interface with the following code in the editor:

```
1 attach(mtcars)
2 mtcars
3 n = nrow(mtcars)
4 y = mtcars[, "mpg"]
5
6 predictors_nm = c("wt", "hp", "displ")
7
8 x = mtcars[,predictors_nm]
9 Int = rep(1,n)
10
11 X = as.matrix(cbind(Int,x))
```

The console shows the output of the code:

```
Volvo 142E      1 109 2.780 121.0
> str(X)
num [1:32, 1:4] 1 1 1 1 1 1 1 1 1 1 ...
- attr(*, "dimnames")=List of 2
 ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 D
rive" ...
 ..$ : chr [1:4] "Int" "hp" "wt" "disp"
>
```

The Environment pane on the right shows the following data:

Object	Class	Attributes
data	data.frame	13320 obs. of 9 variables
df	data.frame	12711 obs. of 9 variables
x	matrix	32 obs. of 3 variables
X	matrix	num [1:32, 1:4] 1 ...

The Files pane shows a list of files:

Name	Size	Modified
Bangalore_House_Price.csv	916 KB	May 31, 2021, 6
Delhi_House_data.csv	153.9 KB	May 31, 2021, 6
Pune_House_data.csv	895.7 KB	May 31, 2021, 6

And now what I will do cbind, the Int and x, this will give me the thing that I am looking for. So, and we can put it into x as dot matrix. So, now, we have x as matrix and if we just say here structure of x, it is a numeric array with 32 rows and 44 columns. It has dimension names, the least in the rows these are the dimension names and in columns you have these 3 columns intercept, horsepower, weight and displacement. Now, my data is ready, both x and y is ready. So, what we can do?

(Refer Slide Time: 06:13)

The screenshot shows the RStudio interface with the following code in the script editor:

```
7  
8 x = mtcars[,predictors_nm]  
9 Int = rep(1,n)  
10  
11 X = as.matrix(cbind(Int,x))  
12  
13  
14 ## Calculate X'X  
15  
16 t(X)  
17
```

The console output shows the dimensions and structure of the matrix X:

```
R 4.2.1 - Downloads/Teaching/Regression_and_Classification/  
- attr(*, "dimnames")=List of 2  
..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 D  
rive"  
..$ : chr [1:4] "Int" "hp" "wt" "disp"  
> t(X)  
Mazda RX4 Mazda RX4 Wag Datsun 710 Hornet 4 Drive  
Int 1.00 1.000 1.00 1.000  
hp 110.00 110.000 93.00 110.000
```

The Environment pane on the right shows the following data objects:

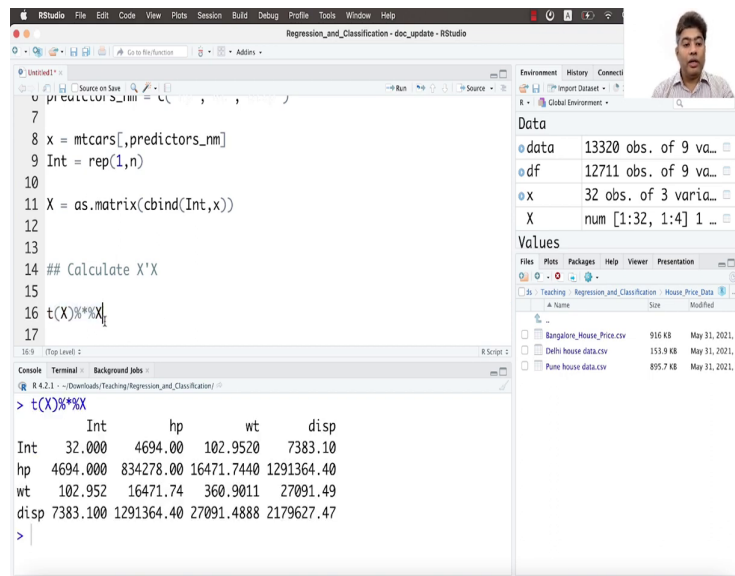
Object	Dimensions
data	13320 obs. of 9 variables
df	12711 obs. of 9 variables
x	32 obs. of 3 variables
X	num [1:32, 1:4] 1 ...

The Files pane shows the following files:

Name	Size	Modified
Bangalore_House_Price.csv	916 KB	May 31, 2021, 6
Delhi_House_Price.csv	153.9 KB	May 31, 2021, 6
Pune_House_Price.csv	895.7 KB	May 31, 2021, 6

First calculate, so in X transpose X. So, first thing I will do, I will say t of X.

(Refer Slide Time: 06:34)



The screenshot shows the RStudio interface with the following code in the script editor:

```
7  
8 x = mtcars[,predictors_nm]  
9 Int = rep(1,n)  
10  
11 X = as.matrix(cbind(Int,x))  
12  
13  
14 ## Calculate X'X  
15  
16 t(X)%*%X  
17
```

The console output shows the result of the matrix multiplication:

```
> t(X)%*%X  
      Int      hp      wt      disp  
Int  32.000  4694.00  102.9520  7383.10  
hp   4694.000  834278.00  16471.7440  1291364.40  
wt   102.952  16471.74  360.9011  27091.49  
disp 7383.100 1291364.40  27091.4888  2179627.47  
>
```

The Environment pane on the right shows the following data:

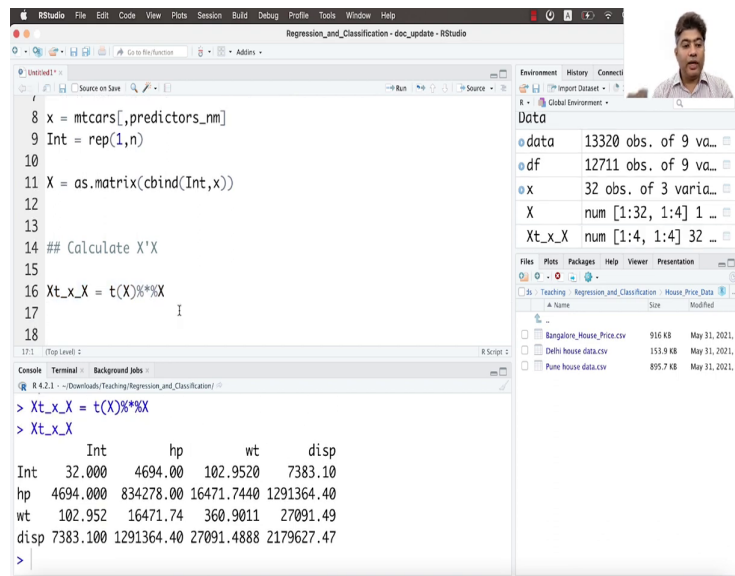
Object	Value
data	13320 obs. of 9 variables
df	12711 obs. of 9 variables
x	32 obs. of 3 variables
X	num [1:32, 1:4] 1 ...

The Files pane shows a list of files:

Name	Size	Modified
Bangalore_House_Price.csv	916 KB	May 31, 2021, 6
Delhi_House_Price.csv	153.9 KB	May 31, 2021, 6
Pune_House_Price.csv	895.7 KB	May 31, 2021, 6

So, now, we can see this is transpose of X, and then I will multiply in the matrix multiplication in R is percentage star percentage. So, if I just do that, it gives me X transpose X.

(Refer Slide Time: 06:48)



The screenshot shows the RStudio interface with the following code in the editor:

```
8 x = mtcars[,predictors_nm]
9 Int = rep(1,n)
10
11 X = as.matrix(cbind(Int,x))
12
13
14 ## Calculate X^X
15
16 Xt_x_X = t(X)%*%X
17
18
```

The console output shows the result of the calculation:

```
> Xt_x_X = t(X)%*%X
> Xt_x_X
      Int      hp      wt      disp
Int 32.000 4694.00 102.9520 7383.10
hp 4694.000 834278.00 16471.7440 1291364.40
wt 102.952 16471.74 360.9011 27091.49
disp 7383.100 1291364.40 27091.4888 2179627.47
```

The Environment pane on the right shows the following data objects:

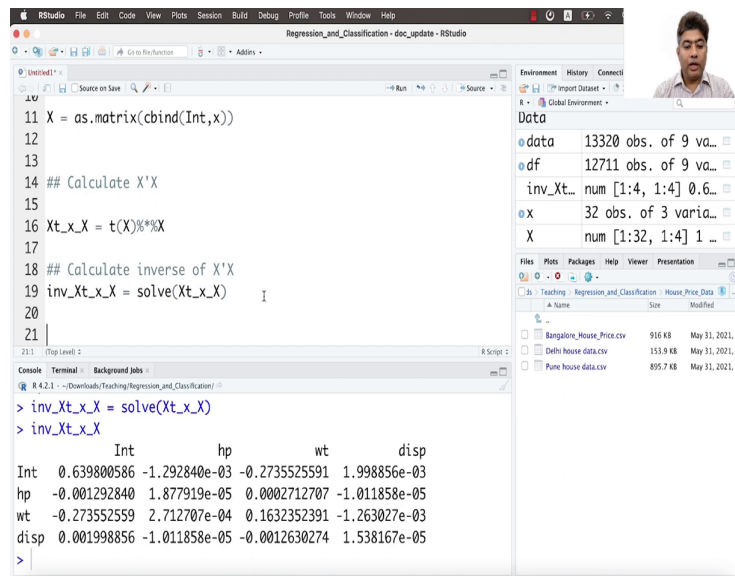
Object	Dimensions
data	13320 obs. of 9 variables
df	12711 obs. of 9 variables
x	32 obs. of 3 variables
X	num [1:32, 1:4] 1 ...
Xt_x_X	num [1:4, 1:4] 32 ...

The Files pane shows the following files:

Name	Size	Modified
Bangalore_House_Price.csv	916 KB	May 31, 2021, 6
Delhi_House_Price.csv	153.9 KB	May 31, 2021, 6
Pune_House_Price.csv	895.7 KB	May 31, 2021, 6

And then what I will do, let me just put it in $X^T X$; let me define this variable $X^T X$.

(Refer Slide Time: 07:09)



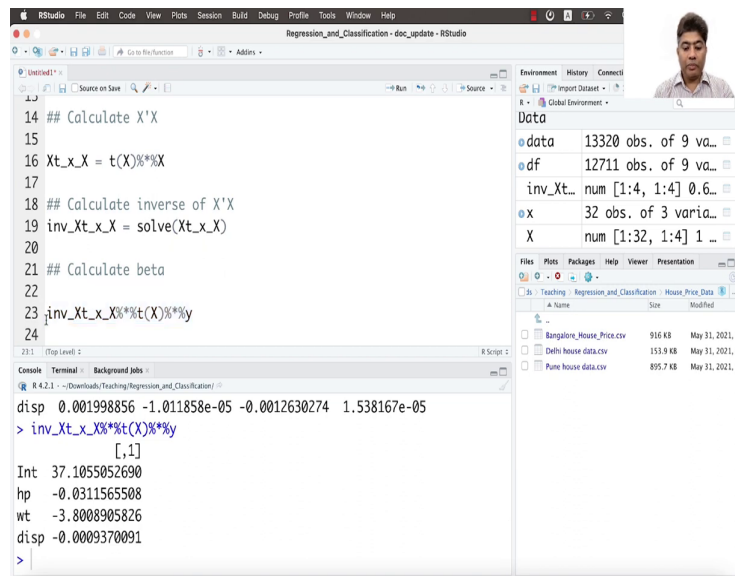
```
11 X = as.matrix(cbind(Int,x))
12
13
14 ## Calculate X'X
15
16 Xt_x_X = t(X)%*%X
17
18 ## Calculate inverse of X'X
19 inv_Xt_x_X = solve(Xt_x_X)
20
21
221 |
```

```
> inv_Xt_x_X = solve(Xt_x_X)
> inv_Xt_x_X
```

	Int	hp	wt	disp
Int	0.639800586	-1.292840e-03	-0.2735525591	1.998856e-03
hp	-0.001292840	1.877919e-05	0.0002712707	-1.011858e-05
wt	-0.273552559	2.712707e-04	0.1632352391	-1.263027e-03
disp	0.001998856	-1.011858e-05	-0.0012630274	1.538167e-05

So, this is X transpose X and then solve that inverse of X t X will be basically you solve this X transpose X , this will give you calculate X transpose X , inverse of X transpose X , so this will give you the X transpose X inverse.

(Refer Slide Time: 07:45)



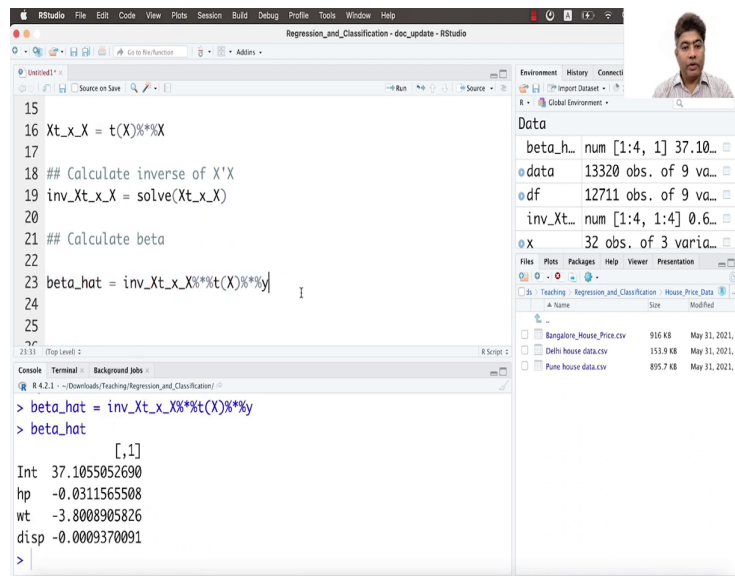
```
14 ## Calculate X'X
15
16 Xt_x_X = t(X)%*%X
17
18 ## Calculate inverse of X'X
19 inv_Xt_x_X = solve(Xt_x_X)
20
21 ## Calculate beta
22
23 inv_Xt_x_X%*%t(X)%*%y
24
251 (Top Level)
```

```
R 4.2.1 -> Downloads/Teaching/Regression_and_Classification/
disp 0.001998856 -1.011858e-05 -0.0012630274 1.538167e-05
> inv_Xt_x_X%*%t(X)%*%y
[1,]
Int 37.1055052690
hp -0.0311565508
wt -3.8008905826
disp -0.0009370091
>
```

The screenshot shows the RStudio interface. The main editor window contains R code for calculating the beta coefficients in a linear regression model. The code includes comments and uses the `t()` function for matrix transposition and `solve()` for matrix inversion. The console window shows the output of the code, including the values of the beta coefficients for the variables `disp`, `Int`, `hp`, `wt`, and `disp`. The Environment pane on the right shows the data objects created, including `data`, `df`, `inv_Xt_x_X`, `x`, and `X`.

And then, you calculate $\beta = (X^T X)^{-1} X^T y$. So, if I just run that, so this is the values that I am getting.

(Refer Slide Time: 08:09)



The screenshot shows the RStudio interface with the following code in the editor:

```
15
16 Xt_x_X = t(X)%*%X
17
18 ## Calculate inverse of X'X
19 inv_Xt_x_X = solve(Xt_x_X)
20
21 ## Calculate beta
22
23 beta_hat = inv_Xt_x_X*t(X)%*%y
24
25
```

The console output shows the result of the beta_hat calculation:

```
> beta_hat = inv_Xt_x_X*t(X)%*%y
> beta_hat
      [,1]
Int 37.1055052690
hp  -0.0311565508
wt  -3.8008905826
disp -0.0009370091
>
```

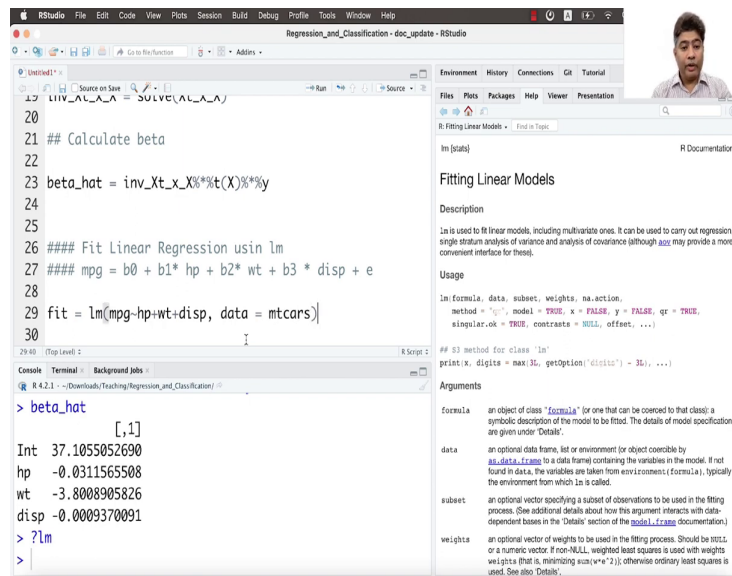
The Environment pane on the right shows the following data objects:

Object	Class	Dimensions	Value
beta_h...	num	[1:4, 1]	37.10...
data	data.frame	13320 obs. of 9 va...	
df	data.frame	12711 obs. of 9 va...	
inv_Xt_...	num	[1:4, 1:4]	0.6...
x	data.frame	32 obs. of 3 varia...	

The Files pane shows a directory structure with files like Bangalore_House_Price.csv, Delhi_house_data.csv, and Pune_house_data.csv.

So, this is I am calling it beta hat. So, if I just run this, so this is my beta hat.

(Refer Slide Time: 08:23)



```
20
21 ## Calculate beta
22
23 beta_hat = inv_Xt_X%*%t(X)%*%y
24
25
26 #### Fit Linear Regression using lm
27 ##### mpg = b0 + b1* hp + b2* wt + b3 * disp + e
28
29 fit = lm(mpg~hp+wt+disp, data = mtcars)
30
```

```
> beta_hat
      [,1]
Int 37.1055052690
hp -0.0311565508
wt -3.8008905826
disp -0.0009370091
> ?lm
```

The screenshot shows the RStudio interface. The script editor contains R code for calculating the beta vector and fitting a linear model. The console shows the output of the beta_hat variable. The help pane on the right displays the documentation for the lm function, including its description, usage, and arguments.

And now, what I am going to do is going to call there is a something called lm. If you just go to console to question mark lm, it will open the manual for lm. It fits the linear models. So, what model fit linear regression using lm the model? I want to fit is miles per gallon as a function of intercept plus b 1 times, with what was there? Horsepower, horsepower plus b 2 times weight and displacement, weight and b 3 times displacement and plus some error, ok.

So, this is the model I am trying to fit. So, what I will do is fit equal to lm you have to say miles per gallon, tilde horsepower plus weight plus displacement comma, now data you have to provide the data, data is mtcars.

(Refer Slide Time: 09:56)

The screenshot shows the RStudio interface with the following code in the script editor:

```
20  
21 ## Calculate beta  
22 beta_hat = inv Xt_X %t(X) %y  
23  
24  
25  
26 #### Fit Linear Regression using lm  
27 ##### mpg = b0 + b1* hp + b2* wt + b3 * disp + e  
28  
29 fit = lm(mpg~hp+wt+disp, data = mtcars)  
30  
31
```

The console output shows the results of the linear regression fit:

```
      [,1]  
Int 37.1055052690  
hp  -0.0311565508  
wt  -3.8008905826  
disp -0.0009370091  
> ?lm  
> fit = lm(mpg~hp+wt+disp, data = mtcars)  
>
```

The right-hand pane displays the documentation for the `lm` function, titled "Fitting Linear Models". It includes a description, usage instructions, and a list of arguments.

Fitting Linear Models

Description

`lm` is used to fit linear models, including multivariate ones. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although `glm` may provide a more convenient interface for these).

Usage

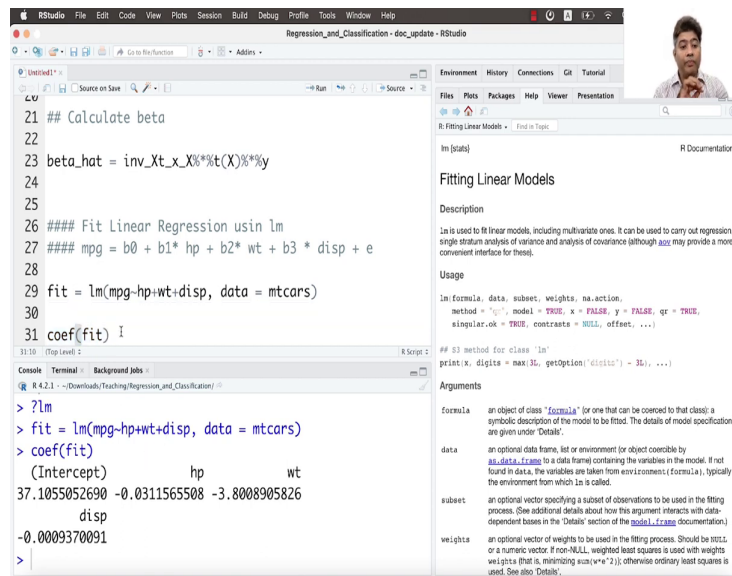
```
lm(formula, data, subset, weights, na.action,  
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,  
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

Arguments

- `formula`: an object of class "formula" (for one that can be coerced to that class); a symbolic description of the model to be fitted. The details of model specification are given under "Details".
- `data`: an optional data frame, list or environment for object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `lm` is called.
- `subset`: an optional vector specifying a subset of observations to be used in the fitting process. (See additional details about how this argument interacts with case-dependent bases in the "Details" section of the `model.frame` documentation.)
- `weights`: an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If non-`NULL`, weighted least squares is used with weights `weights` that is, minimizing $\sum w_i (y_i - \hat{y}_i)^2$; otherwise ordinary least squares is used. See also "Details".

And now if you just run this that is good enough.

(Refer Slide Time: 10:01)



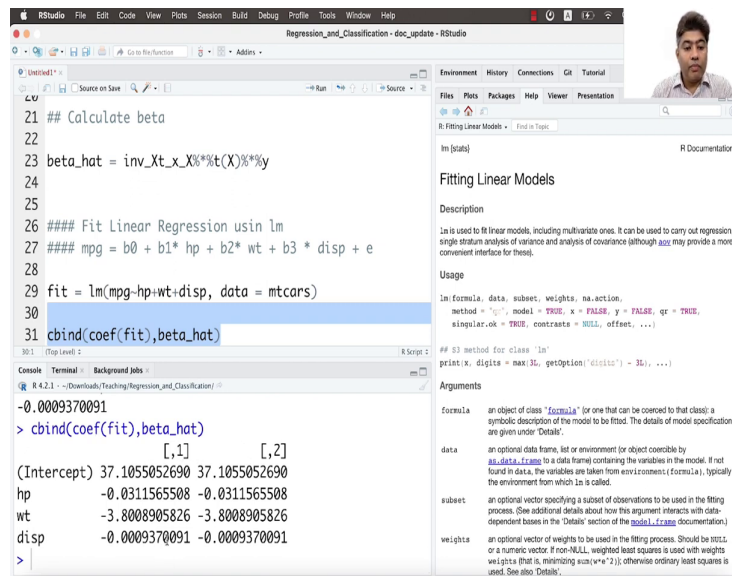
```
21 ## Calculate beta
22
23 beta_hat = inv_Xt_x_X%*t(X)%*%y
24
25
26 #### Fit Linear Regression using lm
27 #### mpg = b0 + b1* hp + b2* wt + b3 * disp + e
28
29 fit = lm(mpg~hp+wt+disp, data = mtcars)
30
31 coef(fit) |
```

```
> ?lm
> fit = lm(mpg~hp+wt+disp, data = mtcars)
> coef(fit)
(Intercept)      hp      wt
37.1055052690 -0.0311565508 -3.8008905826
      disp
-0.0009370091
|
```

The screenshot shows the RStudio interface. The main editor window contains R code for calculating the beta coefficients of a linear regression model. The console window shows the execution of the code, including the fitting of the model and the extraction of coefficients. The right-hand pane displays the documentation for the `lm` function, titled "Fitting Linear Models".

So, no and then, if you just say `coef` equal to `fit`, it will give you all the coefficient values that you are looking for.

(Refer Slide Time: 10:14)



```
21 ## Calculate beta
22
23 beta_hat = inv_Xt_x_X%*%t(X)%*%y
24
25
26 ### Fit Linear Regression using lm
27 ### mpg = b0 + b1* hp + b2* wt + b3 * disp + e
28
29 fit = lm(mpg~hp+wt+disp, data = mtcars)
30
31 cbind(coef(fit),beta_hat)
```

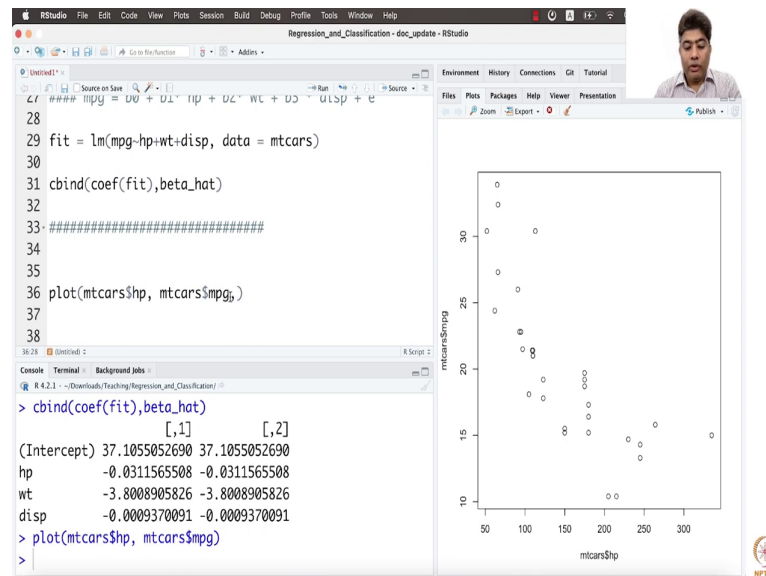
```
30:1 (Top Level) >
-0.0009370091
> cbind(coef(fit),beta_hat)
      [,1]      [,2]
(Intercept) 37.1055052690 37.1055052690
hp          -0.0311565508 -0.0311565508
wt          -3.8008905826 -3.8008905826
disp        -0.0009370091 -0.0009370091
>
```

The screenshot shows the RStudio interface. The script editor contains R code for calculating beta_hat and fitting a linear regression model. The console shows the output of the cbind function, which combines the coefficients from the lm model and the manually calculated beta_hat. The output shows that the two values are identical up to the 10th decimal place. The right-hand pane displays the documentation for the lm function, including its description, usage, and arguments.

Now, what we will do? We will just cbind them, again beta hat and cbind them. Now, the first one is coming from the Rs lm models. You can blindly apply the R lm models. And this is we the second one is beta hat we calculated all by ourselves. And you can see these two values are matching till the last digit 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, till 10 digit, both the method are matching exactly.

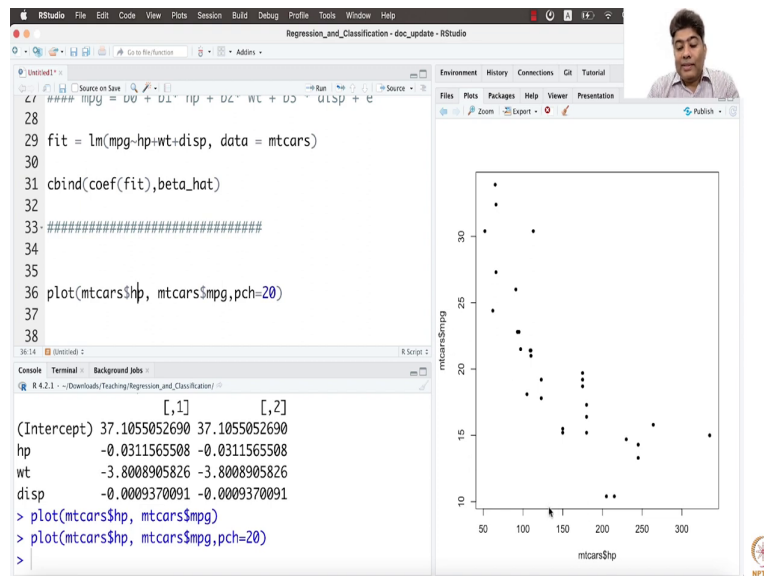
So, this is a this is how lm or in both in R and python, they are calculate the coefficient values through these matrix operation. I just showed you that how to do that. Next, I want you to learn few things.

(Refer Slide Time: 11:19)



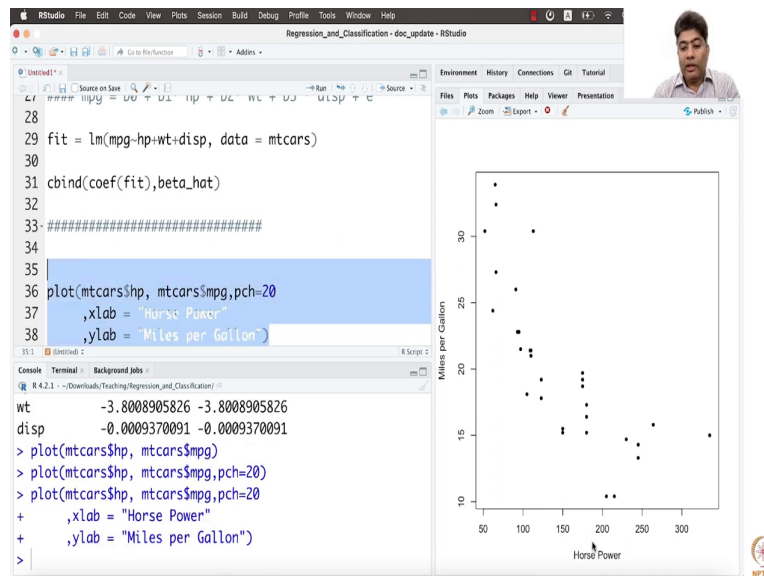
Say, suppose you want to before fitting a model, probably you would like to plot some of the you know more some of the data that you have to develop a intuition. So, maybe we want to plot horsepower and mtcars dollar miles per gallon and if we just plot that. So, we get this kind of plot.

(Refer Slide Time: 12:00)



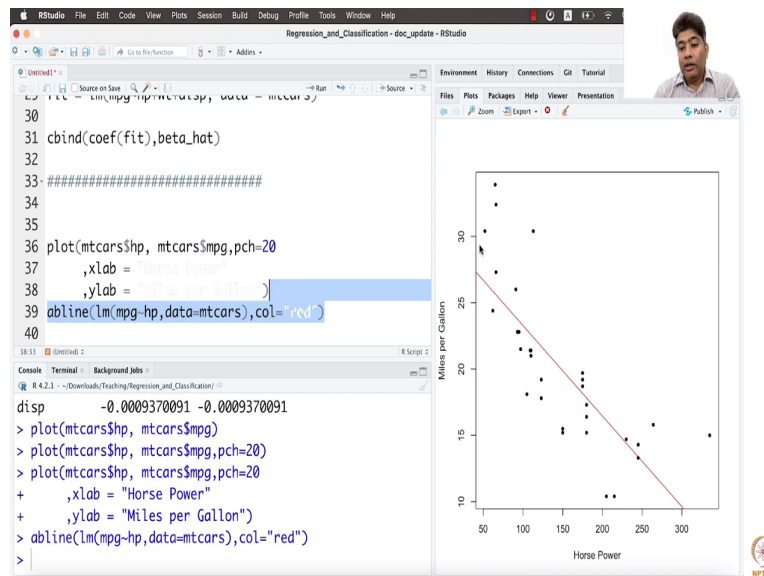
If we just put say `pch` equal to 20, you can and run again you will see this now plots are kind of filled up with the black, so like more prominent. But you see, the way we have provided the code that, ok from the `mtcars` you extract the horsepower and that will be your x axis, and `mtcars` you extract the miles per gallon that will be your y axis.

(Refer Slide Time: 12:35)



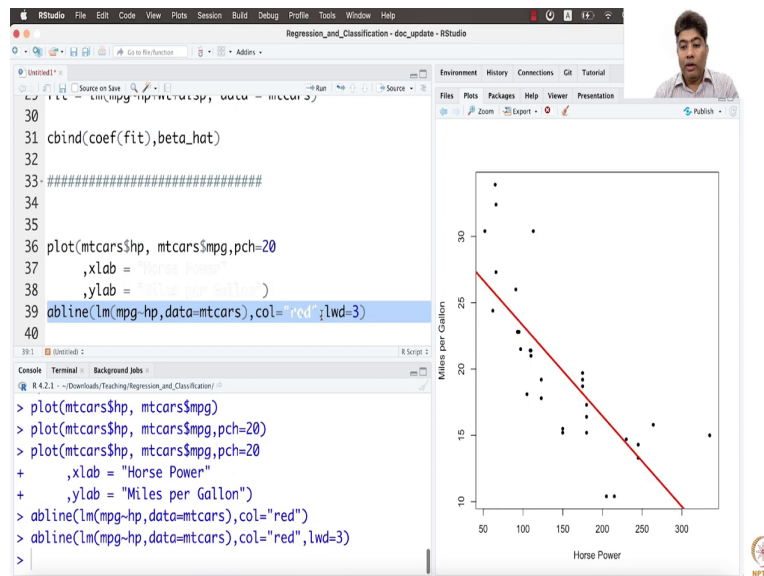
We do not want to see it in this way, maybe we just want to give a name for this xlab equal to the label, stands for labels, horsepower and y label equal to miles per gallon, miles per gallon. So, now if we run you can see that labels are now look nice. And then, what I am going to do?

(Refer Slide Time: 13:10)



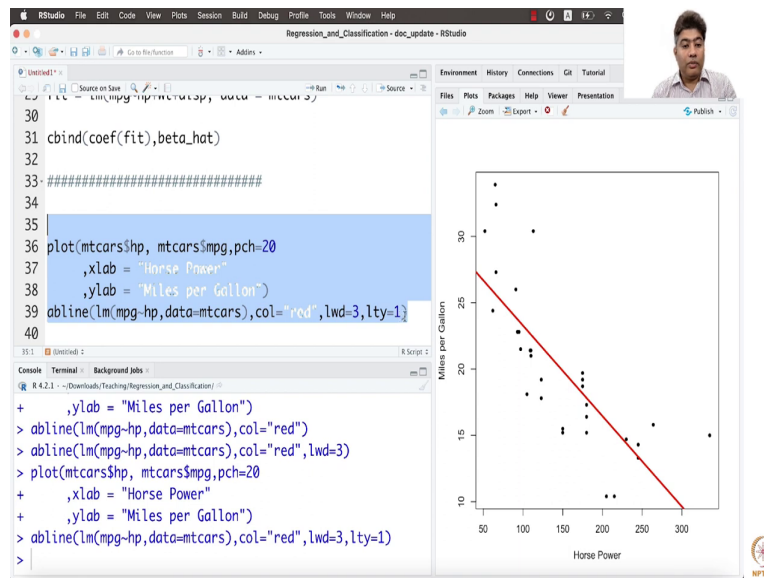
I am going to call a function called abline, abline, and in that abline I am going to call lm which is a miles per gallon as a function of horsepower and I have to provide the data equal to data equal to mtcars. And I have to provide a color, I will provide a maybe I will provide a red color. And if I do that, you can see its drawing a straight line here.

(Refer Slide Time: 13:45)



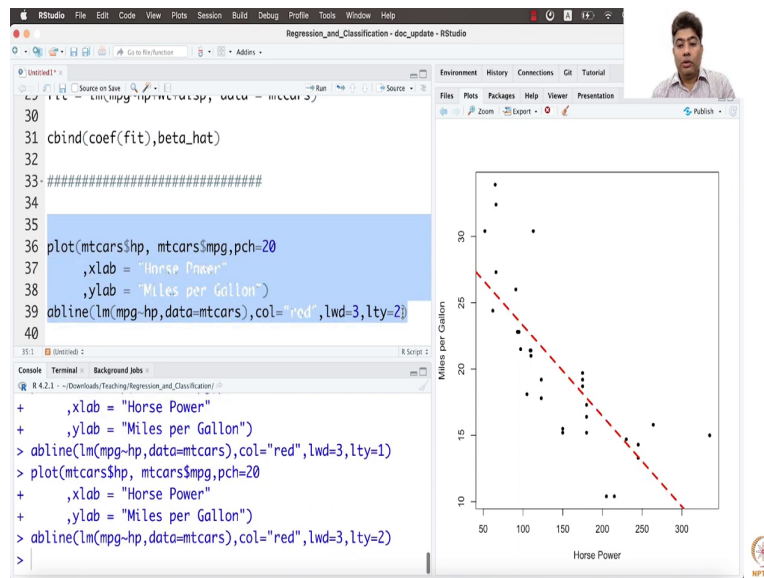
So, maybe what I will do, I will just increase the line width to 3 maybe. And now, it is a little bit better.

(Refer Slide Time: 13:56)



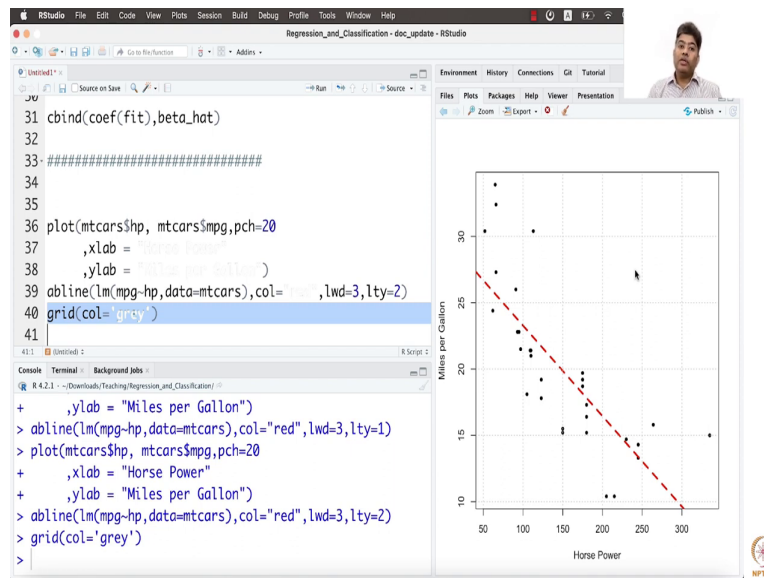
If you increase, if you may consider lty equal to maybe 1 or 2, and if you just want this might be it maybe 2.

(Refer Slide Time: 14:11)



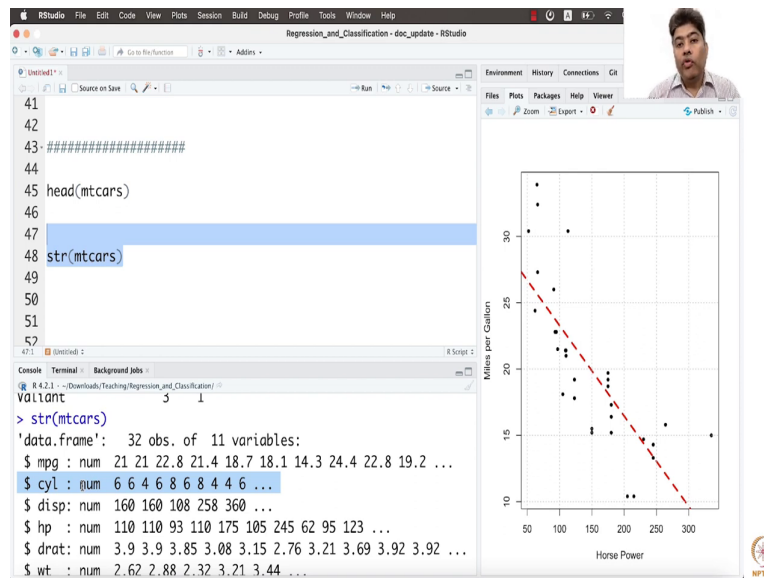
So, now, you can create a dash line, if you choose line type equal to 2, 3, 4, this will give you different kind of dashed line.

(Refer Slide Time: 14:24)



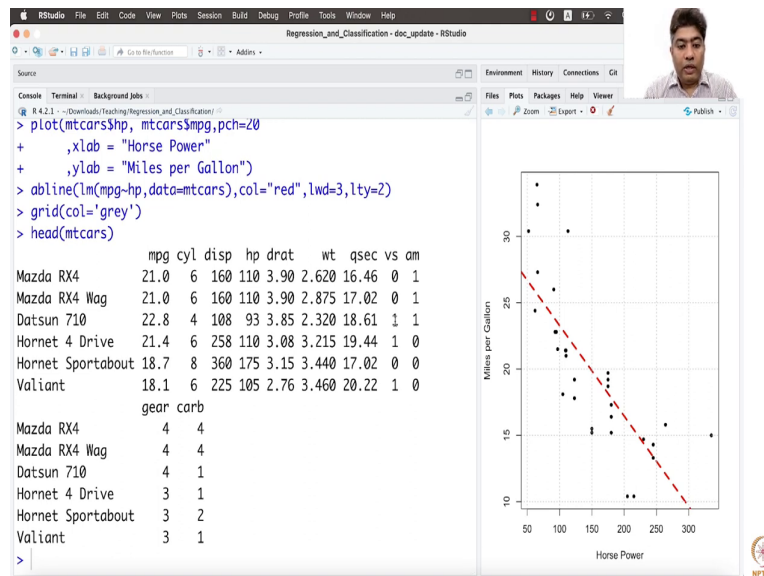
And last one thing if you want to do some grid, if you just say grid with color equal to maybe grey and you run this. So, you will see that there will be a grid kind of thing comes in. So, lot of things you can do with these kind of plots. And these kind of plots are very useful in linear regression.

(Refer Slide Time: 14:59)



Now, I am going to show few more things, like a little bit more on the exploration of the of the design matrix. So, if we look into this data set mtcars are little bit more, ok; mtcars data set, maybe we just say head.

(Refer Slide Time: 15:27)



So, we see that cylinder is typically 6 cylinder, 4 cylinder or 8 cylinder kind of things. And then you know vs is v shaped or not, am stands for its automatic car or manual car; gear stands for whether it is a 3 gear car or 4 gear car, and carburettor stands for how many carburettors they have.

So, suppose, and so far our regression model that we have discussed are all in our predictors are all predictors are all essentially continuous variable like horsepower or you know weight or displacement. They you can consider them as a continuous variable. But what happens if my predictors are not continuous variable, rather my variables these are all categorical variable. So, let us see how these things happen.

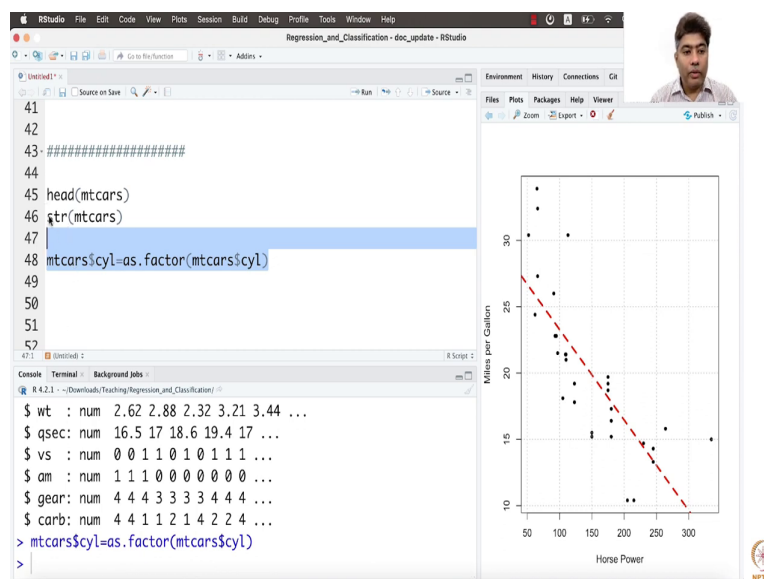
So, first let us work with the cylinder, it could be 6 cylinder, 4 cylinder or 8 cylinder. These cylinder is a ordinal categorical variable. You cannot have a 5 cylinder, generally you do not

have a 5 or 4 cylinder, well car or; 5.5 cylinder car. You can have a 5, you may design a 5 cylinder car, but you cannot have really 5.5 cylinder cars.

So, it is a sort of a discrete kind of things. And how; and if you want to give them as a categorical variable, you want to, if your predictor has a categorical variable, how you handle categorical variable in the linear model setup or the in the design matrix. Now, how do you do that?

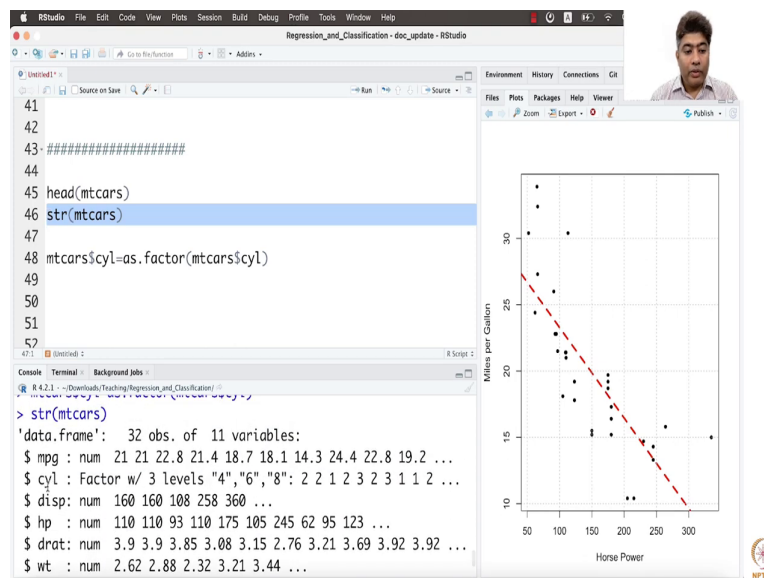
So, like one good thing is, one possibility is you just in the m; if you look into the structure of the data set mtcars, here a cylinder is being coded as numeric variable. So, but what we will do, we will transform them it into factor variable.

(Refer Slide Time: 17:57)



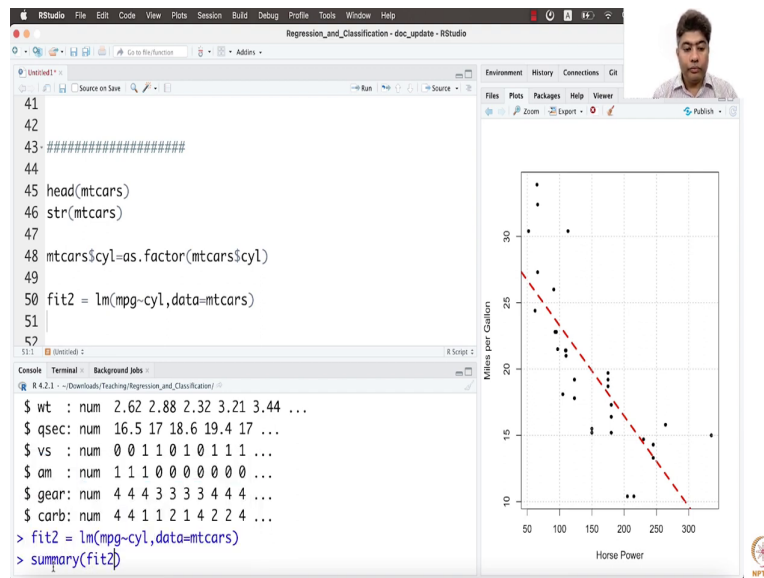
So, now, what I will do, I will just take it as mtcars dollar cylinder as dot factor mtcars dollar cylinder. Now, what is happening?

(Refer Slide Time: 18:18)



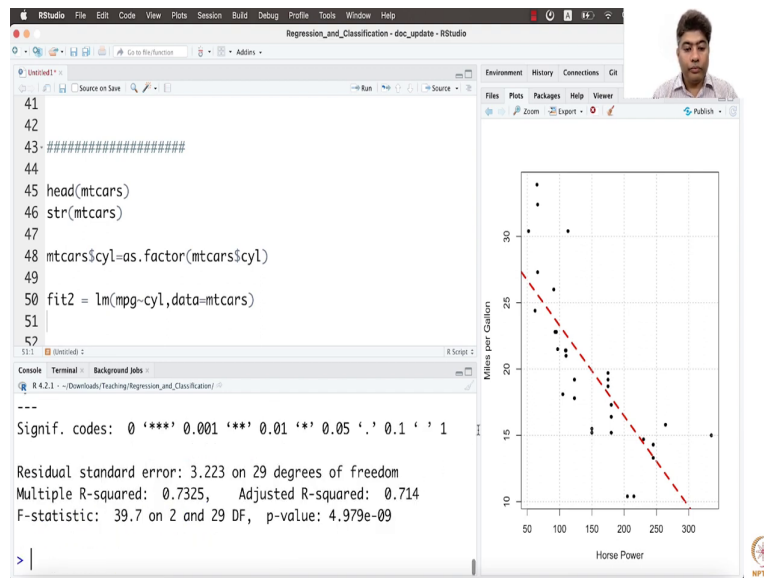
If you look into the structure of the data set, you see a factor and it has 3 levels, 4, 6, and 8. Now, what we will do?

(Refer Slide Time: 18:38)



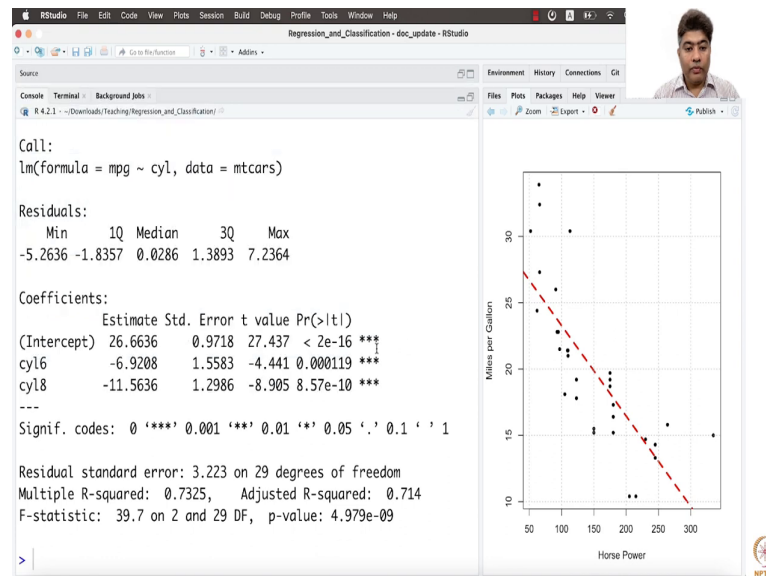
We will if we run the lm between say miles per gallon, and say cylinder, then it is called data equal to mtcars and we call it fit 2.

(Refer Slide Time: 19:10)



And then if we run say summary of it, so what we will see that for 6 cylinder, we got something and for 8 cylinder and then there is a intercept.

(Refer Slide Time: 19:13)



So, what kind of; we will try to understand what kind of model are is fitting here. So, you have to first understand how the data set is. So, what we will do now, we will go back to the theory and we will try to understand how to handle this kind of categorical variable as predictor.

Thank you. Let us continue on the next video on the theory side of how to handle categorical variable in the predictor.