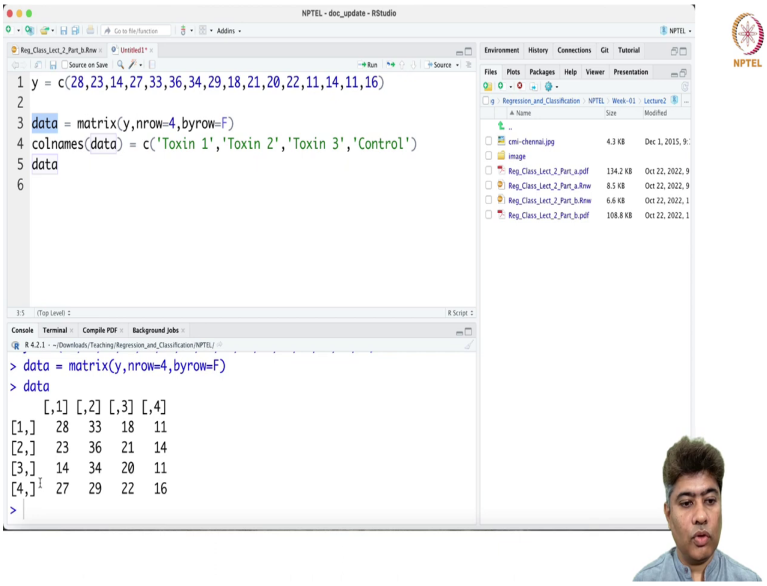
Predictive Analytics - Regression and Classification
Prof. Sourish Das
Department of Mathematics
Chennai Mathematical Institute

Lecture - 07
Hand-on with R Part -2

Hi guys welcome back to lecture 2 in this part of the lecture I am going to do some hands on using R.

(Refer Slide Time: 00:27)

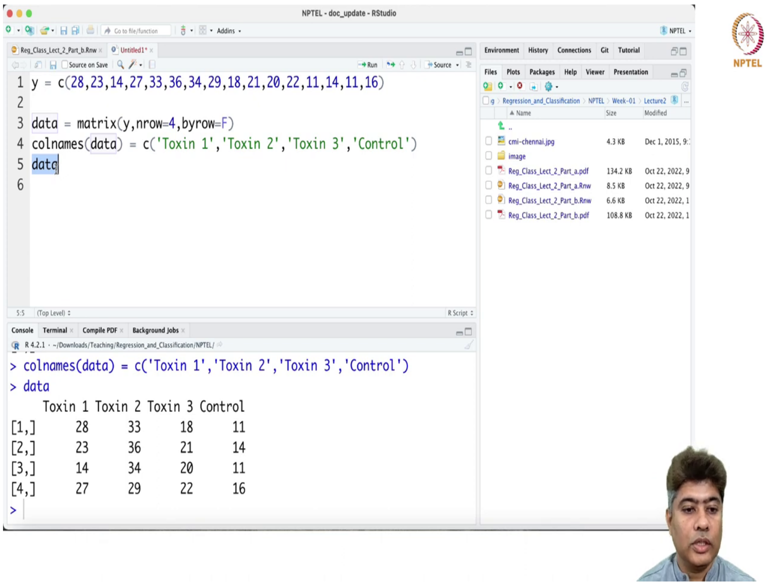


```
y = c(28,23,14,27,33,36,34,29,18,21,20,22,11,14,11,16)

data = matrix(y,nrow=4,byrow=F)
colnames(data) = c('Toxin 1','Toxin 2','Toxin 3','Control')
data
```

```
> data = matrix(y,nrow=4,byrow=F)
> data
     [,1] [,2] [,3] [,4]
[1,]   28   33   18   11
[2,]   23   36   21   14
[3,]   14   34   20   11
[4,]   27   29   22   16
>
```

So, what I have done, I have essentially put all the data in a vector y and then I rate that data as kind of a matrix and put. So, you can see the data here as a matrix with 4 rows and by row equal to false because I want to read them as a column. So now if you run that, so I rate the data as a matrix and then I put the column names here.

(Refer Slide Time: 00:58)



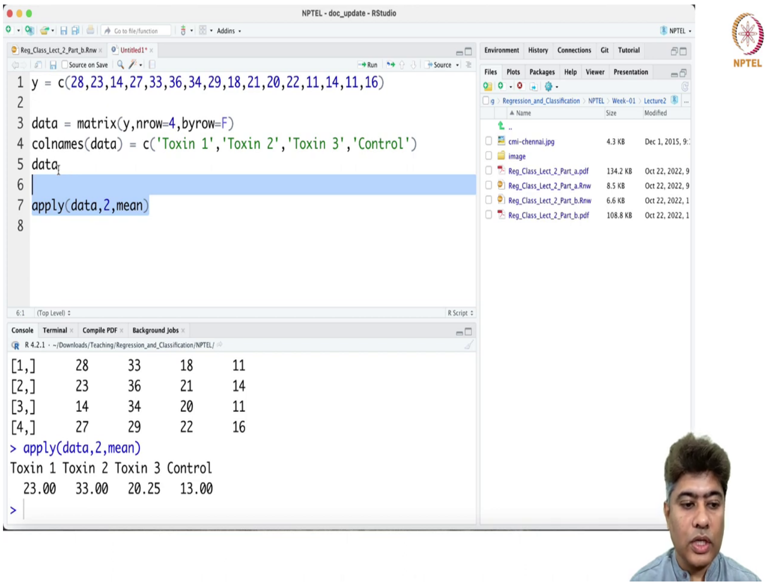
```
y = c(28,23,14,27,33,36,34,29,18,21,20,22,11,14,11,16)

data = matrix(y,nrow=4,byrow=F)
colnames(data) = c('Toxin 1','Toxin 2','Toxin 3','Control')
data
```

```
> colnames(data) = c('Toxin 1','Toxin 2','Toxin 3','Control')
> data
     Toxin 1 Toxin 2 Toxin 3 Control
[1,]      28      33      18      11
[2,]      23      36      21      14
[3,]      14      34      20      11
[4,]      27      29      22      16
>
```

NPTEL - doc_update - RStudio
Reg_Class_Lect_2_Part_b.Rnw
Untitled1*
Source on Save
Run
Source
Console
Terminal
Compile PDF
Background Jobs
R 4.2.1 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
Environment
History
Connections
Git
Tutorial
Files
Plots
Packages
Help
Viewer
Presentation
Regression_and_Classification > NPTEL > Week-01 > Lecture2
Name
Size
Modified
cmi-chennai.jpg
image
Reg_Class_Lect_2_Part_a.pdf
Reg_Class_Lect_2_Part_a.Rnw
Reg_Class_Lect_2_Part_b.Rnw
Reg_Class_Lect_2_Part_b.pdf
NPTEL

So, if you put the data here. So, you know you see the each the way exactly I presented in the you know theory part the data is kind of looks exactly something like that.

(Refer Slide Time: 01:16)

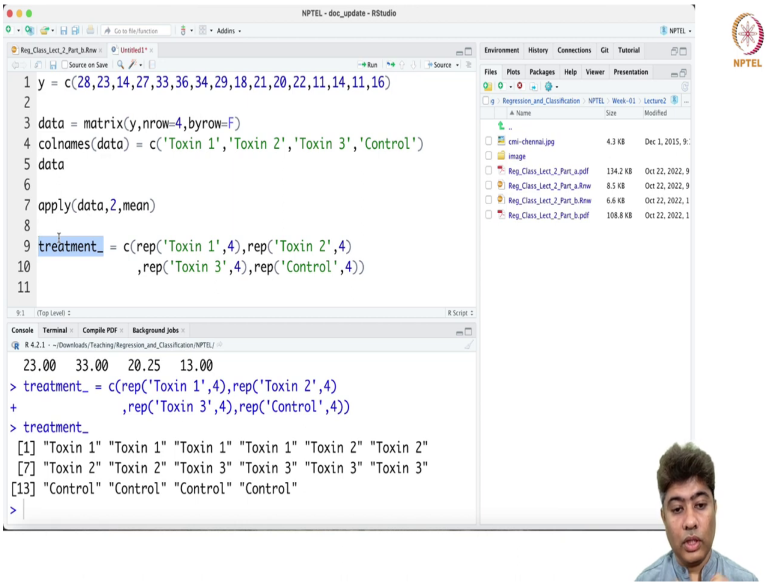

NPTEL - doc_update - RStudio
Reg_Class_Lect_2_Part_b.Rnw
Untitled1*
Environment History Connections Git Tutorial
Files Plots Packages Help Viewer Presentation
Console Terminal Compile PDF Background Jobs
NPTEL

```
y = c(28,23,14,27,33,36,34,29,18,21,20,22,11,14,11,16)

data = matrix(y,nrow=4,byrow=F)
colnames(data) = c('Toxin 1','Toxin 2','Toxin 3','Control')
data

apply(data,2,mean)
```

```
[1,]      28      33      18      11
[2,]      23      36      21      14
[3,]      14      34      20      11
[4,]      27      29      22      16
> apply(data,2,mean)
Toxin 1 Toxin 2 Toxin 3 Control
  23.00   33.00   20.25   13.00
>
```

Now, if I apply if I use the apply function and to calculate the mean by columns it will give me the group means. So, if I just say data comma 2 will give me the column wise mean and I just call for mean. So, it gives me the column means for each group. So, Toxin 2 has a group mean of 23 Toxin 2 has a group mean of 33 Toxin 3 has a group mean of 20 and control group has average of 13.

(Refer Slide Time: 02:05)


NPTEL - doc_update - RStudio
Reg_Class_Lect_2_Part_b.Rnw
Untitled1*
Run
Source
Environment
History
Connections
Git
Tutorial
Files
Plots
Packages
Help
Viewer
Presentation
cmi-chennai.jpg
image
Reg_Class_Lect_2_Part_a.pdf
Reg_Class_Lect_2_Part_a.Rnw
Reg_Class_Lect_2_Part_b.Rnw
Reg_Class_Lect_2_Part_b.pdf
Console
Terminal
Compile PDF
Background Jobs
R 4.2.1 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
NPTEL

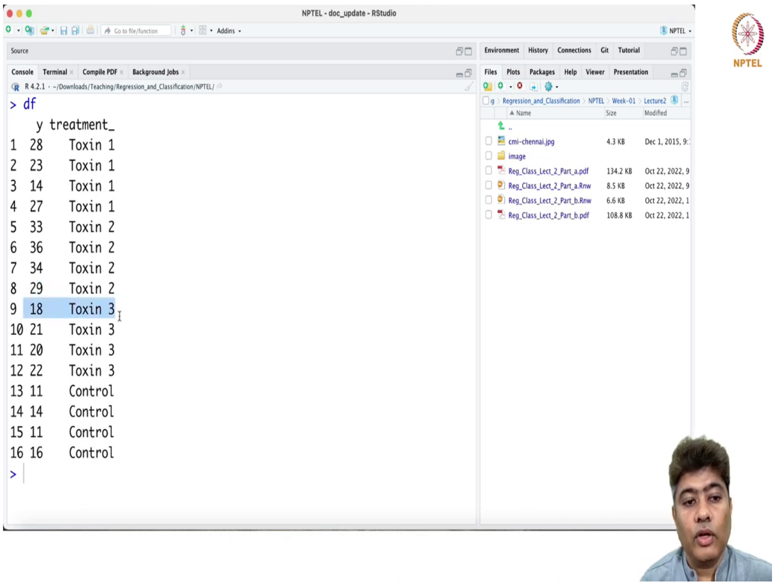
```r
y = c(28,23,14,27,33,36,34,29,18,21,20,22,11,14,11,16)

data = matrix(y,nrow=4,byrow=F)
colnames(data) = c('Toxin 1','Toxin 2','Toxin 3','Control')
data

apply(data,2,mean)

treatment_ = c(rep('Toxin 1',4),rep('Toxin 2',4)
                 ,rep('Toxin 3',4),rep('Control',4))
```

```
  23.00   33.00   20.25   13.00
> treatment_ = c(rep('Toxin 1',4),rep('Toxin 2',4)
+                 ,rep('Toxin 3',4),rep('Control',4))
> treatment_
 [1] "Toxin 1" "Toxin 1" "Toxin 1" "Toxin 1" "Toxin 2" "Toxin 2"
 [7] "Toxin 2" "Toxin 2" "Toxin 3" "Toxin 3" "Toxin 3" "Toxin 3"
[13] "Control" "Control" "Control" "Control"
>
```

Now, what I am going to do I am going to apply the mod equal to say mod ln and ok; before that I need to define a treatment, say treatment equal to replicate and say our first I want the first 4 values in the y variable is belongs to Toxin 1. So, I am going to create Toxin 1 4 variable then next 4 way the observation are going to Toxin 2. So, I am going to create this Toxin this is Toxin 1 this is Toxin 2 then I will just copy this 2 here this is Toxin 3 and this is control group this is control group.

So, let me just run this. So, what I have here is Toxin 1 4 Toxin 1 then 4 Toxin 2 then 4 Toxin 3 and then 4 Toxin.

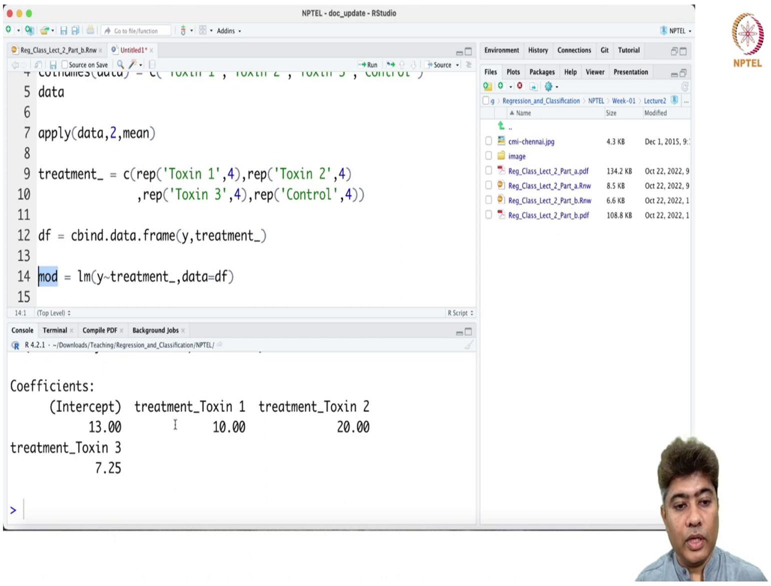
(Refer Slide Time: 03:42)



```
> df
    y treatment_
1  28    Toxin 1
2  23    Toxin 1
3  14    Toxin 1
4  27    Toxin 1
5  33    Toxin 2
6  36    Toxin 2
7  34    Toxin 2
8  29    Toxin 2
9  18    Toxin 3
10 21    Toxin 3
11 20    Toxin 3
12 22    Toxin 3
13 11    Control
14 14    Control
15 11    Control
16 16    Control
>
```

Now, what I am going to do is if I just cbind dot data dot frame comma y equal to dot treatment as a df I am going to keep that as a data frame ok. Now, if I just run the data frame. So, you can see that my y variable is all the values that we have in the data and treatment which group this particular value belongs to the first value belongs to group Toxin 1 second value belongs to group Toxin 1 Toxin 1 Toxin 1 then 5th value belongs to Toxin group 2 6th value belongs to Toxin group 2 similarly the 8th value belongs to Toxin group 2 then the 9th value belongs to Toxin group three that is how I am pre processing the data.

(Refer Slide Time: 04:44)

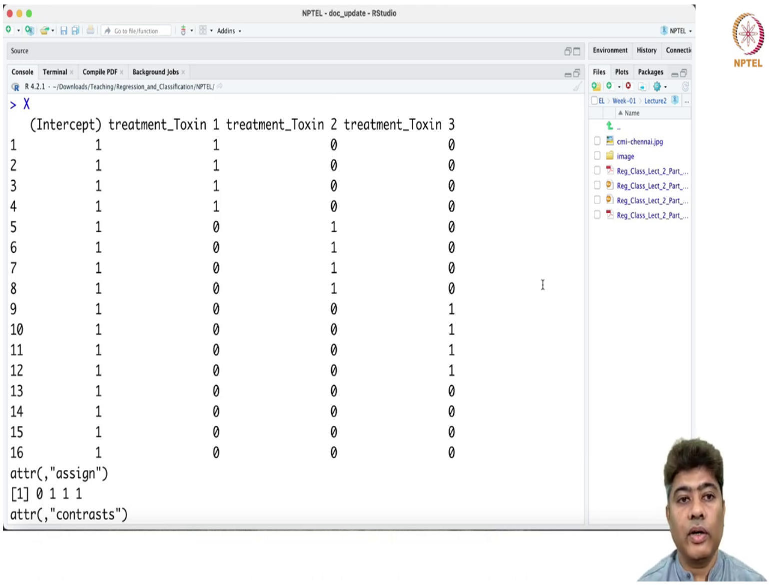


```
colnames(data) = c('Toxin 1','Toxin 2','Toxin 3','Control')
data

apply(data,2,mean)

treatment_ = c(rep('Toxin 1',4),rep('Toxin 2',4)
               ,rep('Toxin 3',4),rep('Control',4))

df = cbind.data.frame(y,treatment_)

mod = lm(y~treatment_,data=df)

```

```
Coefficients:
      (Intercept)  treatment_Toxin 1  treatment_Toxin 2
            13.00              10.00              20.00
treatment_Toxin 3
             7.25

>
```

Now, I am going to fit a model using lm saying that y colon treatment 1 delta equal to df, this df I am going to give. So, this gives me model and if I just run this it gives me the coefficient values, but I will come back to this before that.

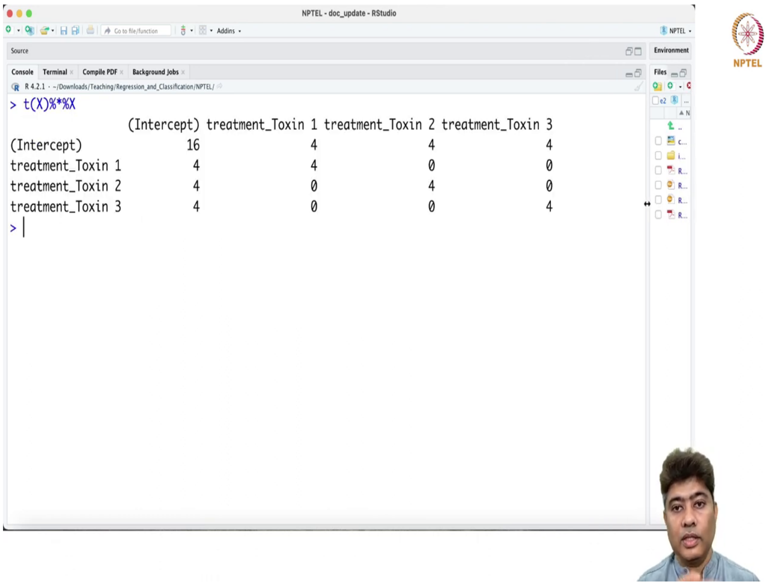
(Refer Slide Time: 05:17)



```
> X
  (Intercept) treatment_Toxin 1 treatment_Toxin 2 treatment_Toxin 3
1           1                 1                 0                 0
2           1                 1                 0                 0
3           1                 1                 0                 0
4           1                 1                 0                 0
5           1                 0                 1                 0
6           1                 0                 1                 0
7           1                 0                 1                 0
8           1                 0                 1                 0
9           1                 0                 0                 1
10          1                 0                 0                 1
11          1                 0                 0                 1
12          1                 0                 0                 1
13          1                 0                 0                 0
14          1                 0                 0                 0
15          1                 0                 0                 0
16          1                 0                 0                 0
attr(,"assign")
[1] 0 1 1 1
attr(,"contrasts")
```

So, first what I am going to do I am going to from the mod; from the mod I am going to extract the model matrix model dot matrix and as the x or design matrix. Now, you will see if you look into this first is intercept, then the second column is treatment Toxin 1 then treatment Toxin 2 and treatment Toxin 3, let me just increase the fonts window size and run this. You can see this is the exact design matrix that we got in the theory we designed and this is exactly what we are getting.

(Refer Slide Time: 06:11)

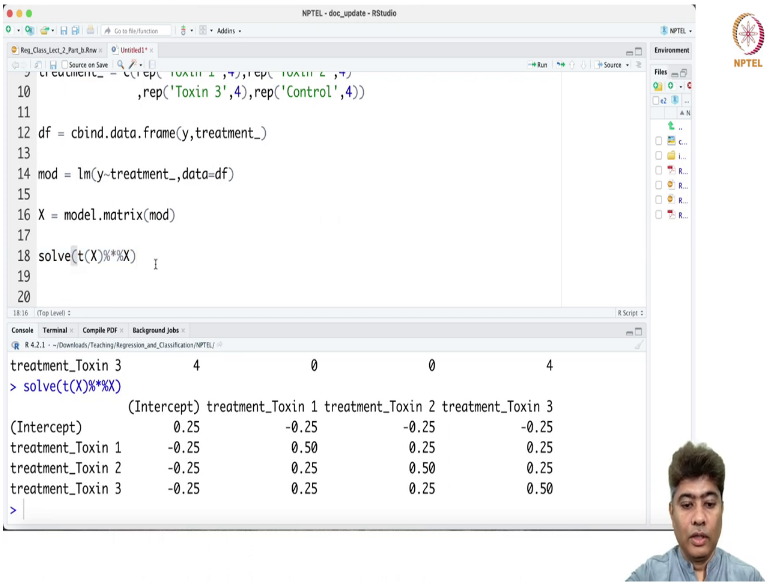


NPTEL - doc_update - RStudio
Source
Console
Terminal
Compile PDF
Background Jobs
R 4.2.1 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
Environment
Files
NPTEL

```
> t(X)%*%X
                 (Intercept) treatment_Toxin 1 treatment_Toxin 2 treatment_Toxin 3
(Intercept)               16                 4                 4                 4
treatment_Toxin 1          4                 4                 0                 0
treatment_Toxin 2          4                 0                 4                 0
treatment_Toxin 3          4                 0                 0                 4
> 
```

Now, what we are going to do I am going to compute the beta hat for that I need to compute the x transpose x percentage star percentage x. So, this is my x transpose x this is exactly what we got 16 4 4 4. So, if I let me see if I can yeah this is the exactly the x transpose x we got.

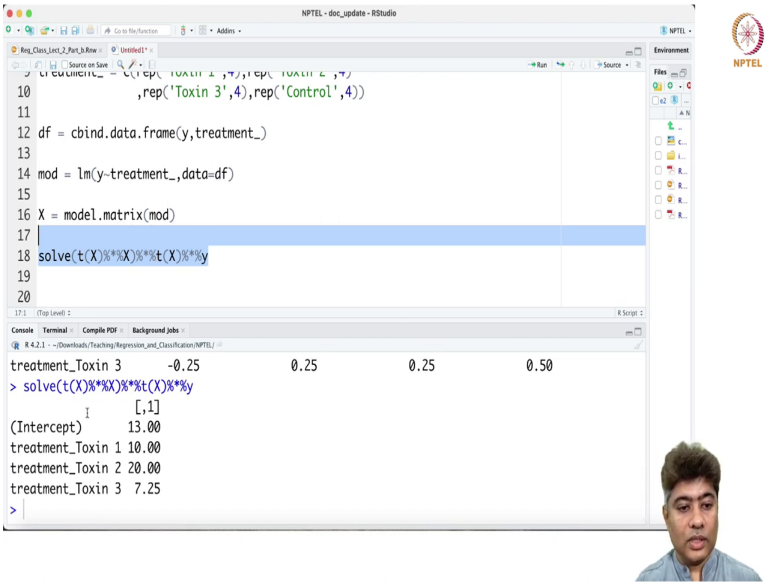
(Refer Slide Time: 06:42)



```
treatment_ = c(rep('Toxin 1',4),rep('Toxin 2',4)
                ,rep('Toxin 3',4),rep('Control',4))

df = cbind.data.frame(y,treatment_)

mod = lm(y~treatment_,data=df)

X = model.matrix(mod)

solve(t(X)%*%X)
```

```
treatment_Toxin 3           4                0                0                4
> solve(t(X)%*%X)
                  (Intercept) treatment_Toxin 1 treatment_Toxin 2 treatment_Toxin 3
(Intercept)              0.25             -0.25             -0.25             -0.25
treatment_Toxin 1       -0.25              0.50              0.25              0.25
treatment_Toxin 2       -0.25              0.25              0.50              0.25
treatment_Toxin 3       -0.25              0.25              0.25              0.50
>
```

(Refer Slide Time: 06:52)

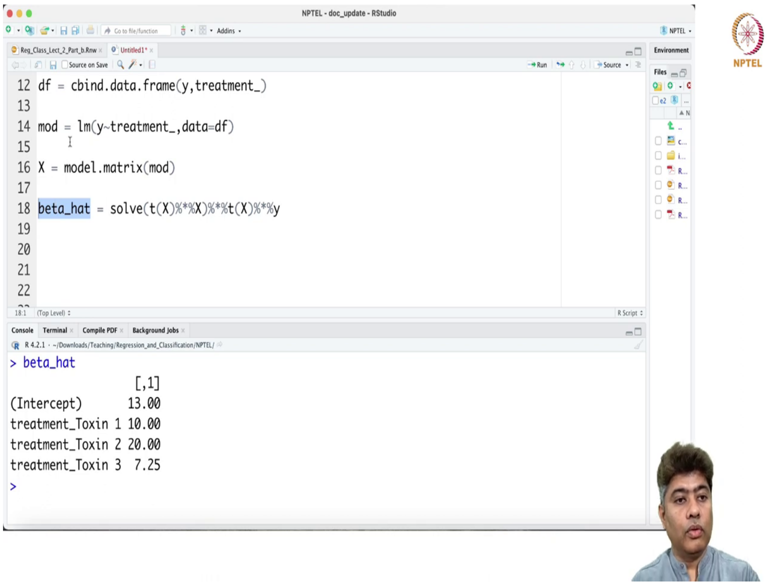


```
treatment_ = c(rep('Toxin 1',4),rep('Toxin 2',4)
               ,rep('Toxin 3',4),rep('Control',4))

df = cbind.data.frame(y,treatment_)

mod = lm(y~treatment_,data=df)

X = model.matrix(mod)


solve(t(X)%*%X)%*%t(X)%*%y


```

```
treatment_Toxin 3       -0.25             0.25             0.25             0.50
> solve(t(X)%*%X)%*%t(X)%*%y
                   [,1]
(Intercept)       13.00
treatment_Toxin 1 10.00
treatment_Toxin 2 20.00
treatment_Toxin 3  7.25
> 
```

And then we solve this this is exactly what x transpose x inverse we got x, then we get x transpose multiplied with x transpose x transpose y. So, this is exactly the beta hat that we got. So, let me see this is exactly the beta hat we got in our previous method ok.

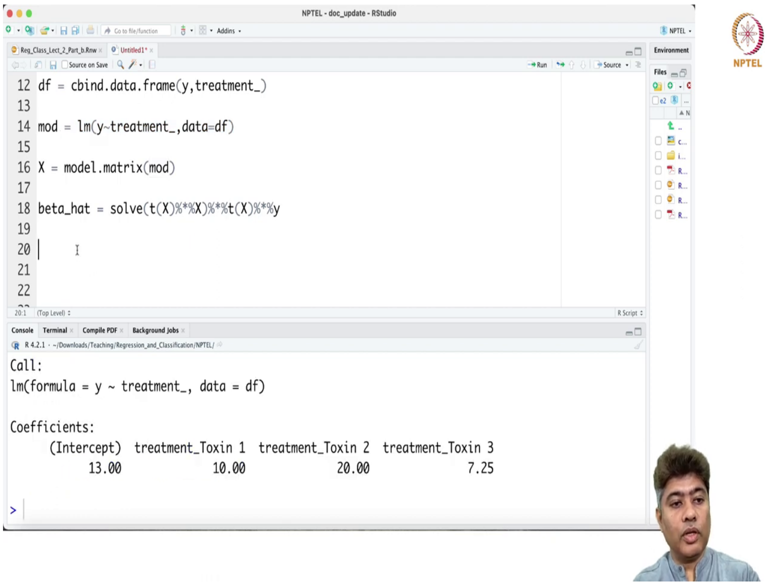
(Refer Slide Time: 07:14)



```r
df = cbind.data.frame(y,treatment_)

mod = lm(y~treatment_,data=df)

X = model.matrix(mod)

beta_hat = solve(t(X)%*%X)%*%t(X)%*%y
```

```
> beta_hat
                   [,1]
(Intercept)       13.00
treatment_Toxin 1 10.00
treatment_Toxin 2 20.00
treatment_Toxin 3  7.25
>
```

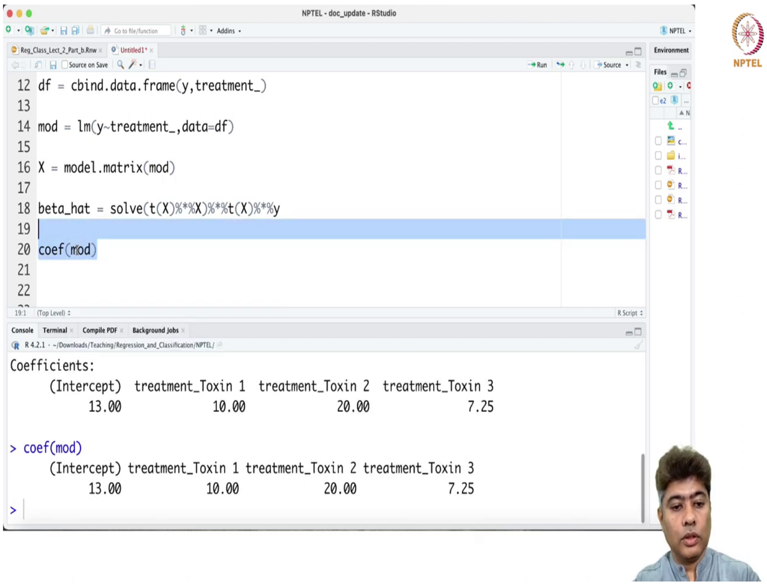
(Refer Slide Time: 07:35)


NPTEL - doc_update - RStudio
Reg_Class_Lect_2_Part_b.Rnw
Untitled1*
Source on Save
Run
Source
Environment
Files
Console
Terminal
Compile PDF
Background Jobs
R 4.2.1 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
(Top Level)
R Script

```
df = cbind.data.frame(y,treatment_)

mod = lm(y~treatment_,data=df)

X = model.matrix(mod)

beta_hat = solve(t(X)%*%X)%*%t(X)%*%y
```

```
Call:
lm(formula = y ~ treatment_, data = df)

Coefficients:
     (Intercept)  treatment_Toxin 1  treatment_Toxin 2  treatment_Toxin 3
           13.00              10.00              20.00               7.25

>
```

(Refer Slide Time: 07:36)

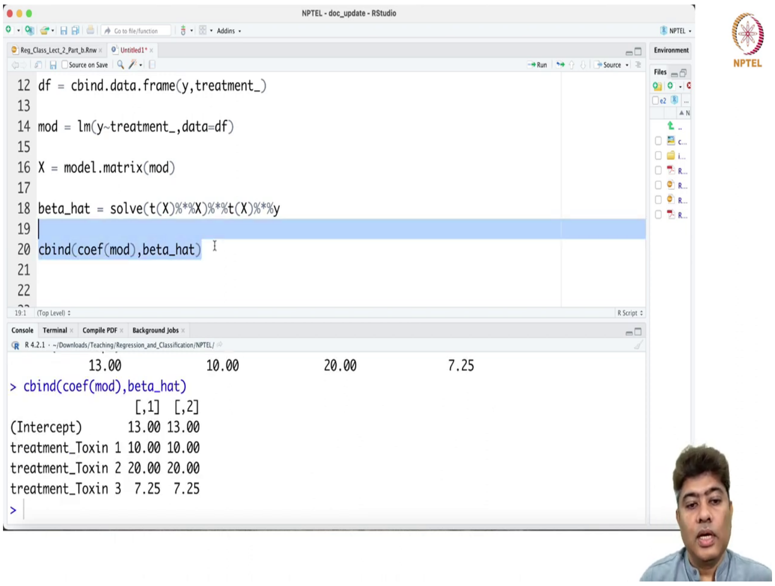

NPTEL - doc_update - RStudio
Reg_Class_Lect_2_Part_b.Rnw
Untitled1*
Source on Save
Run
Source
Environment
Files
Console
Terminal
Compile PDF
Background Jobs
R 4.2.1 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
(Top Level)
R Script

```
df = cbind.data.frame(y,treatment_)

mod = lm(y~treatment_,data=df)

X = model.matrix(mod)

beta_hat = solve(t(X)%*%X)%*%t(X)%*%y

coef(mod)
```

```
Coefficients:
      (Intercept)  treatment_Toxin 1  treatment_Toxin 2  treatment_Toxin 3  
            13.00              10.00              20.00               7.25  

> coef(mod)
      (Intercept) treatment_Toxin 1 treatment_Toxin 2 treatment_Toxin 3 
            13.00             10.00             20.00              7.25 
> 
```

Now, what we will do from the also we fit the model here remember that. So, from the model we extract the coefficient and we also have the beta hat.

(Refer Slide Time: 07:46)


NPTEL - doc_update - RStudio
Reg_Class_Lect_2_Part_b.Rnw
Untitled1*
Console
Terminal
Compile PDF
Background Jobs
R 4.2.1 · ~/Downloads/Teaching/Regression_and_Classification/NPTEL/
Environment
Files
NPTEL

```
df = cbind.data.frame(y,treatment_)

mod = lm(y~treatment_,data=df)

X = model.matrix(mod)

beta_hat = solve(t(X)%*%X)%*%t(X)%*%y

cbind(coef(mod),beta_hat)
```

```
            13.00          10.00          20.00          7.25
> cbind(coef(mod),beta_hat)
                   [,1]  [,2]
(Intercept)       13.00 13.00
treatment_Toxin 1 10.00 10.00
treatment_Toxin 2 20.00 20.00
treatment_Toxin 3  7.25  7.25
>
```

So, we will just cbind them. So, you can see that lm is also using the same method. So, our coefficient from the lm and our calculation exactly matches. So, now you understand that in the categorical variable, you when you use categorical variable as predictor in your regression model, you have to sacrifice one of the level out of many levels.

First of all you have to have at least 1 level more than 1 level and you have to if you have four levels you have to at least predict sacrifice one of the label to make the design matrix full rank that is very important. And if you do not do that then and if you create dummy variable or the one hot encoding for all of the label, then your matrix will automatically become less than full rank and you will not have a solution for your coefficients. So, you have to be careful when you are creating the dummy variables.