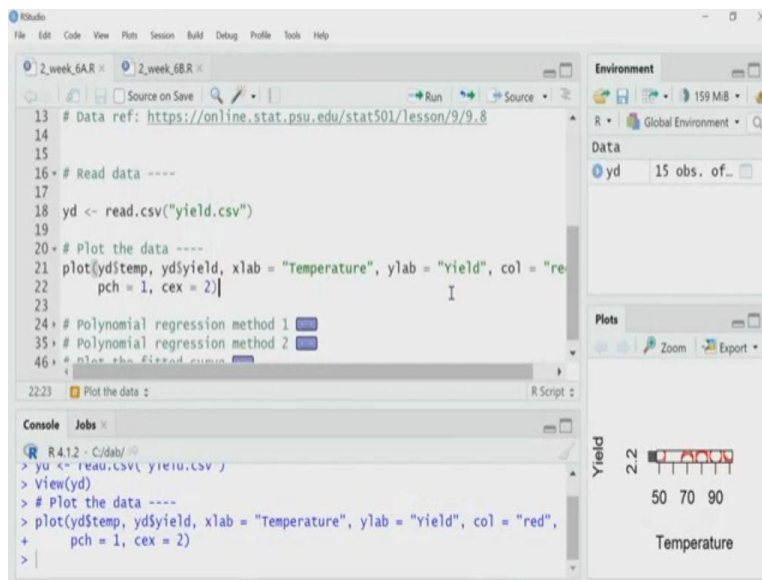


**Data Analysis for Biologists**  
**Professor Biplab Bose**  
**Department of Biosciences and Bioengineering**  
**Mehta Family School of Data Science and Artificial Intelligence**  
**Indian Institute of Technology, Guwahati**  
**Lecture – 33**  
**Nonlinear Regression using R**

Welcome back to our lectures on data analysis for biologists. In this lecture, I will show how to implement a nonlinear regression using R. I will perform two types of nonlinear regression. One is polynomial regression that one I will perform first, and then I will move into a nonlinear least square problem. So, let us start with nonlinear regression for polynomial equations or what we call polynomial regression.

So, if you remember there can be some time the data where y suppose is the dependent variable and x is the independent variable. And you want to fit that data to some equation polynomial for example,  $y$  equal to  $a$  plus  $b_1$  into  $x$  plus  $b_2$  into  $x$  square plus  $b_3$  into  $x$  cube something like that. So, this is a polynomial and you want to fit your data to that polynomial. I will load a data set and plot that and then it will be itself evident, that why I want to go for polynomial regression for that particular problem.

(Refer Slide Time: 01:41)



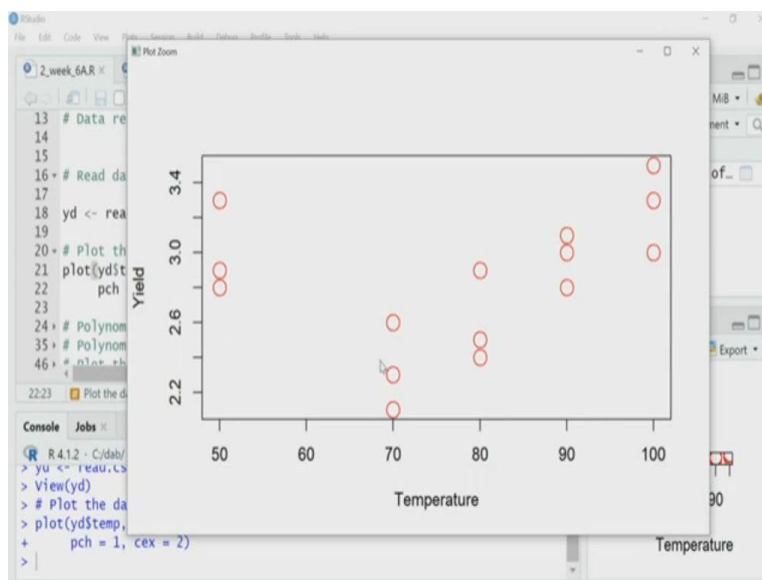
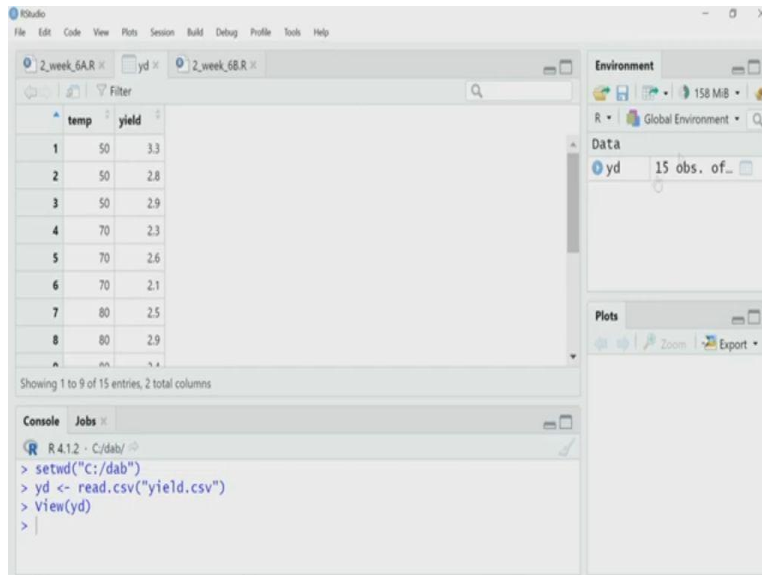
```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help

2_week_6A.R 2_week_6B.R
Source on Save Run Source
13 # Data ref: https://online.stat.psu.edu/stat501/lesson/9/9.8
14
15
16 # Read data ----
17
18 yd <- read.csv("yield.csv")
19
20 # Plot the data ----
21 plot(yd$temp, yd$yield, xlab = "Temperature", ylab = "Yield", col = "red",
22      pch = 1, cex = 2)
23
24 # Polynomial regression method 1
25 # Polynomial regression method 2
26 # Plot the fitted curve
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

Environment
R Global Environment 159 MB
Data
yd 15 obs. of ...

Plots
Zoom Export

Console Jobs
R 4.1.2 - C:\dab\
> yd <- read.csv("yield.csv")
> View(yd)
> # Plot the data ----
> plot(yd$temp, yd$yield, xlab = "Temperature", ylab = "Yield", col = "red",
+      pch = 1, cex = 2)
> |
```



`yd <- read.csv("yield.csv")`

`plot(yd$temp, yd$yield, xlab = "Temperature", ylab = "Yield", col = "red", pch = 1, cex = 2)`

So, in my working directory, there is a data file called yield dot csv; let me give you a brief introduction to this data. Somebody has performed some experiments and suppose chemical reaction at different temperatures; and at different temperatures the yield of the reaction has been measured. So, I have this two variable yield and temperature; temperature is the independent variable, you can call it x.

And whereas the dependent variable is the yield and it depends upon temperature. So, I will first read that csv file using read dot csv and then I will plot that data; and then I will decide why I need a polynomial regression for this and what type of polynomial I will go for. So, here I am calling the read dot csv function to read this csv file and storing that data to a variable called y

Let me check this data yd, this is a two columns data set; first column is the temperature 50, 70 something like that; these are the temperature at which the experiment has been performed. And the next column the next variable is yield and we have 15 observations here. Now, I will plot that data as a scatter plot, so I am using the plot function. The X variable the horizontal variable, X is a variable is the temperature; so, I am writing yd, yd is the variable.

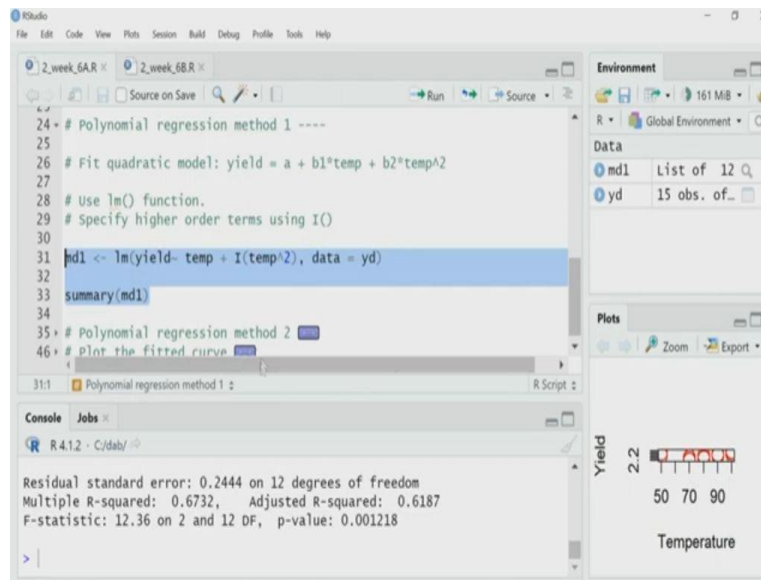
And I am asking the temperature from that and yd is storing the data from that I am calling the yield variable. So, that will be the vertical axis. I am labeling x-axis with temperature and I am labeling y-axis as yield. And I want to use red colour and open circle I want to use; that is why pch is equal to 1 and the size of these symbols is equal to 2.

So, here is my plot, let me zoom it to see it clearly; let me expand it a bit. So, what we can see here is that see the temperature is increasing from 50 to 100. And if you look into the data, the yield is slowly decreasing from 50 and going almost like minimum at 70, and then it is again increasing. So, the data is something like this a conic section, you can imagine a part of a circle or something like that.

So, that means this is a some sort of conic section equation will fit here. So, I will go for a quadratic equation and quadratic equation is a second order polynomial. A quadratic equation will have y equal to a plus b1 into x plus b2 into x square; there will not be x cube, x to the power 4 something like that; we will stop at x square. So, that is my equation that is the polynomial I want to fit to this data, and that is why I have to perform polynomial regression.

Now, so what I am trying to fit is that I want to fit a quadratic equation, and let me write it, show it in a written form. So, I want that yield is equal to a; a is the intercept, a constant, plus b1 into temperature plus b2 into temperature square. Now, if you remember from my lecture on nonlinear regression, polynomial regression is one of the simplest nonlinear regression; because here we will perform same linear regression but with a trick.

(Refer Slide Time: 05:15)



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

```
24 # Polynomial regression method 1 ----
25
26 # Fit quadratic model: yield = a + b1*temp + b2*temp^2
27
28 # Use lm() function.
29 # Specify higher order terms using I()
30
31 md1 <- lm(yield~ temp + I(temp^2), data = yd)
32
33 summary(md1)
34
35 # Polynomial regression method 2
36 # Plot the fitted curve
```

The Environment pane shows the following objects:

Object	Class	Attributes
md1	lm	List of 12
yd	data.frame	15 obs. of...

The Console pane shows the following output:

```
Residual standard error: 0.2444 on 12 degrees of freedom
Multiple R-squared: 0.6732, Adjusted R-squared: 0.6187
F-statistic: 12.36 on 2 and 12 DF, p-value: 0.001218
```

The Plots pane shows a plot of Yield versus Temperature. The x-axis is labeled 'Temperature' and ranges from 50 to 90. The y-axis is labeled 'Yield' and ranges from 2 to 2. The plot shows a scatter of data points and a fitted quadratic curve.

```
md1 <- lm(yield~ temp + I(temp^2), data = yd)
```

```
summary(md1)
```

The trick is, see I have temperature data; so, this temperature square will be considered as an independent variable. So, that means I can consider temperature as one variable independent variable and temperature square as another independent variable. So, this equation becomes yield equal to a plus b1 into one independent variable plus b2 into another independent variable.

So, it is just nothing but multiple linear regression. So, that means, I can use the lm linear model function of R, just that I have to tell the lm function that, see I do not have multiple independent variable, I have just one independent variable; but you have to create multiple independent variable from the data.

That means, I have to tell it, let us see, temperature should be one independent variable, and you will generate temperature square and consider that as another independent variable; and I will use the same lm function. So, how do I implement that? So, here what I will do? I will use the I function within the lm function; let me explain that with the script.

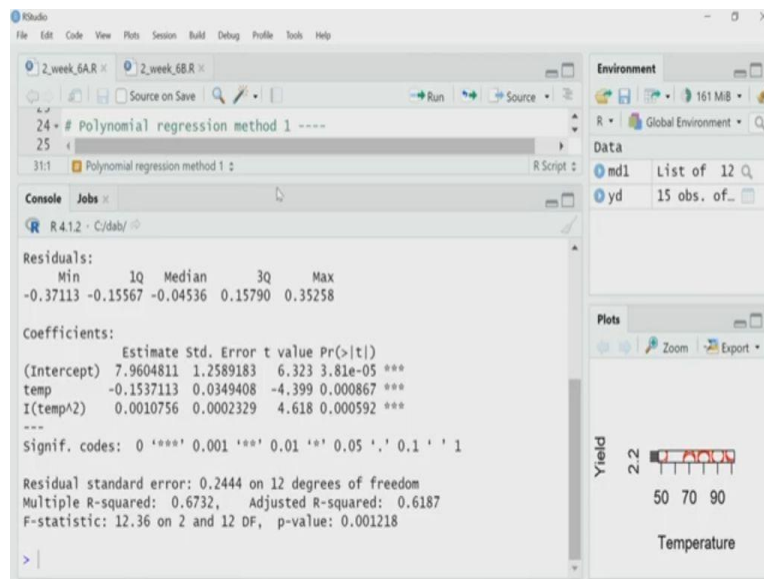
So, I am calling the lm function. And if you remember when you call the linear model function, the first thing you have to define is the model, the linear model. So, yield is the dependent

variable that is why I have written yield then tilde. So, a tilde is after the yield, so, yield is a dependent variable here.

And then I am defining the independent variable; I have said temperature is a independent variable plus, I in bracket I am writing temperature square. This I function and temperature square written inside it will tell the lm that see I have temperature data; I have a variable called temperature. Take that and square it and use it in this model as a independent variable; this is how you define the model.

And then, obviously, by default it will consider the intercept and the data is the yd. And then what I will do after performing the linear model regression, linear regression using this lm function? I will store the whole data in md1 variable; that is why I am using the assignment sign here. And then I will extract the summary of md1; I will implement them both of them together. So, I have got the summary result of my regression.

(Refer Slide Time: 07:44)



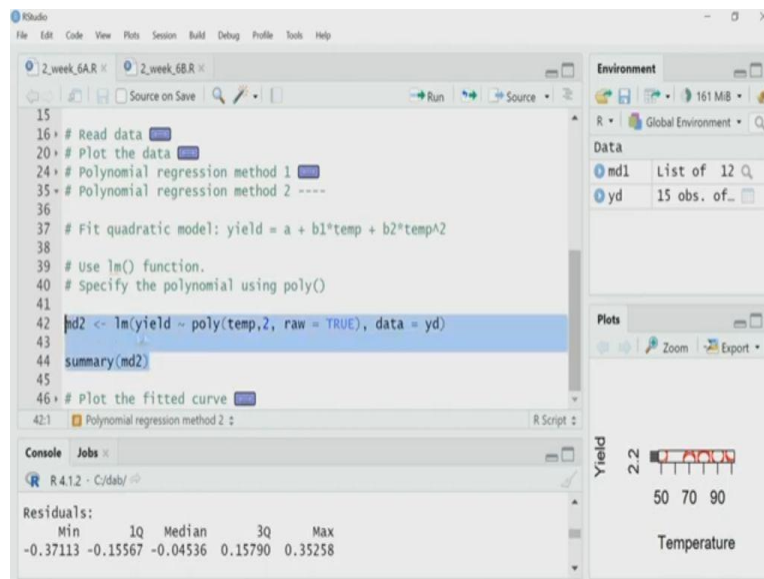
It is a polynomial regression, but it is just like multiple linear regression. And if you look at the output of the summary, this is seen, you must have seen it earlier when I was doing simple linear regression or multiple linear regression. The important thing that we have to notice here is that the estimated coefficients values, the intercept is the constant is 7.96.

Temperature, the coefficient of temperature is minus 0.15. Remember, initially when the temperature increases, the yield falls; that is why this minus is there. And then temperature square the coefficient of that is a positive value and that is equal to point double nought 107.

And it has performed the statistical test. And as you can see the p value for all these three estimated coefficient, they are all passing the statistical test; and that means all these coefficients are statistically significant, and I have to keep all this term in my quadratic equation. You can look into adjusted R square; you can do the confidence interval of these coefficients. I am not going into those, but you can do all those thing; because it is nothing but the simple lm function that you have used earlier for linear regression. What I will do now? I will show you another way of defining the same model and using the lm function. See if I have a lengthy polynomial.

Suppose I have up to the eighth order, x to the power 8, x to the power 7 all these things are there; then you have to repeatedly write those things in your equation. i in x square, i in x cube, something like that that is tedious. There is a function called poly and I will use that function to do a shortcut, and I will show it here.

(Refer Slide Time: 09:38)



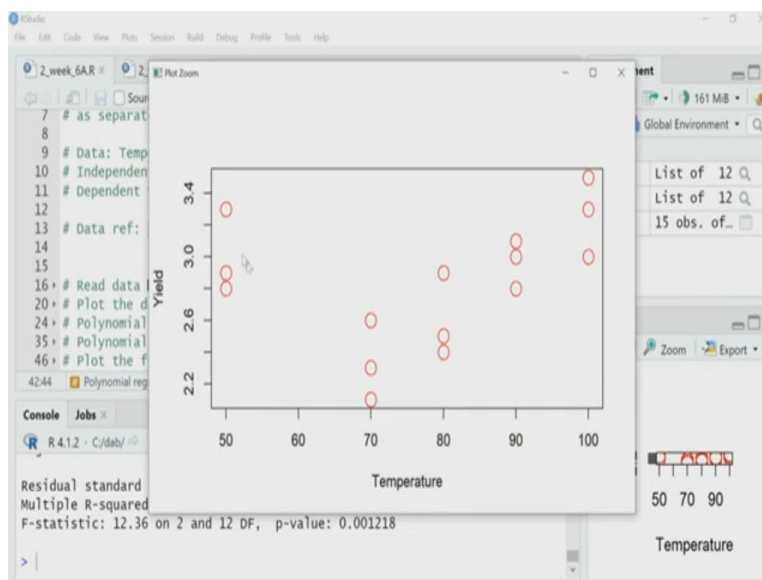
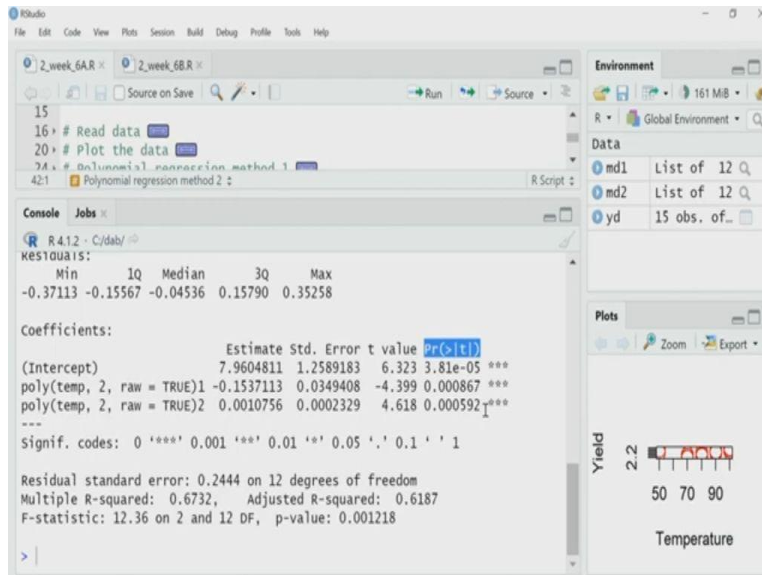
The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for reading data, plotting, and fitting a quadratic model using the `lm()` function with a polynomial of degree 2. The code includes comments and a `summary(md2)` command.
- Environment:** Shows the global environment with objects `md1` (List of 12) and `yd` (15 obs. of ...).
- Plots:** A plot titled "Polynomial regression method 2" showing Yield vs Temperature. The y-axis ranges from 22 to 90, and the x-axis ranges from 50 to 90.
- Console:** Displays the output of `summary(md2)`, showing the residuals distribution.

```
15
16 # Read data
20 # Plot the data
24 # Polynomial regression method 1
35 # Polynomial regression method 2 ----
36
37 # Fit quadratic model: yield = a + b1*temp + b2*temp^2
38
39 # Use lm() function.
40 # Specify the polynomial using poly()
41
42 md2 <- lm(yield ~ poly(temp,2, raw = TRUE), data = yd)
43
44 summary(md2)
45
46 # Plot the fitted curve
42:1 Polynomial regression method 2
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.37113	-0.15567	-0.04536	0.15790	0.35258



```
md2 ← lm(yield~ poly(temp, 2, raw=TRUE), data = yd)
```

```
summary(md2)
```

So again, I am using the `lm` function, and then first thing I am doing is I am defining the model. Here what I am doing? I am writing `yield tilde`; `yield` is a dependent variable and it depend upon a polynomial. That is why I am asking the `poly` function polynomial of temperature, polynomial of temperature; that is I am saying at the first argument of `poly`.

So, the first argument of `poly` function is temperature and 2, 2 is the highest order that I want in the polynomial. And then I am saying `raw` equal to `true`, I will explain why I have to write this

one. The poly function has been created to create something called orthogonal polynomial. We will not go into the detail of that on the mathematics of that.

Just to tell you that orthogonal polynomial are many a time very useful in performing polynomial regression. So, that is why poly function has been created and it has other utilities also. So, when you give data and you define what is the independent variable; and you define the order and you call the poly function, it creates a orthogonal polynomial for that particular independent variable.

Now, in this case, I do not want it to create orthogonal polynomial; but what I am asking it that see you do not need to calculate orthogonal polynomial, you use a raw form, you use the raw data and create a polynomial. So, that is why I am saying raw equal to true. So, it will not create orthogonal polynomial, but it will simple polynomial that we have created just few seconds back in the other example, where I have used I; so that same polynomial it will create.

So, it will not create orthogonal polynomial, it will create the normal polynomial for temperature up to the second order; that means the quadratic equation. So, now you can imagine suppose I have to use up to eighth order, so I can simply replace 2 by 8; and by one simple function call a poly, it will create that polynomial. And again, as I have to define the data for lm function, the data is same, the data is yd.

I will execute that regression and then I will extract the summary of that by summary function, so here is the result. Again, you must check that you will get the same result. Obviously, these are the first column is for the coefficient, and then the last column here is for the p value of the statistical test, and all of them are significant. So, I have performed the polynomial regression using the lm function, the linear model function.

So, now once I have this linear model, that means I have the polynomial regress model; I want to obviously fit that or show that on the plot. I have this plot here that I have just created a few minutes back. So, I now want to overlay that fitted curve, and that is what I will do now. To overlay I have to create some data. So what I will do? I will create a x vector, vector for the horizontal axis.



(Refer Slide Time: 12:55)

The screenshot shows the RStudio environment with the following R code in the editor:

```
46 # Plot the fitted curve ----
47
48 # Create data (x and y vectors) as per the fitted model
49
50 xv = seq(50, 100, 0.1)
51
52 yv = predict(md2, data.frame(temp = xv), type = "response")
53
54
55 # Draw the line
56
57 lines(xv,yv,col = "blue", lwd = 2)
58
59
```

The console shows the execution of these commands:

```
> xv = seq(50, 100, 0.1)
>
> yv = predict(md2, data.frame(temp = xv), type = "response")
> lines(xv,yv,col = "blue", lwd = 2)
>
```

The Environment pane shows variables: md1 (List of 12), md2 (List of 12), and yd (15 obs. of ...). The Plots pane shows a small plot of Yield vs Temperature with a blue line.

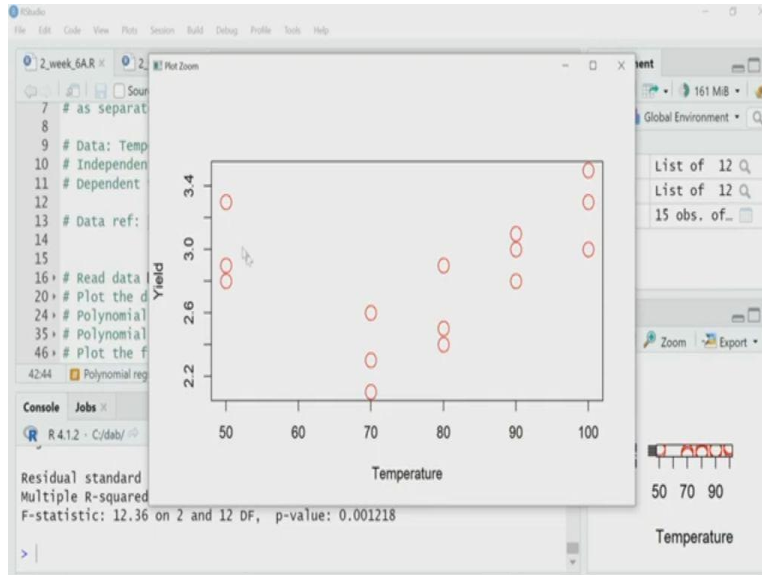
The screenshot shows the RStudio environment with the same R code as above. The console shows the execution of the commands, followed by the output of the plot:

```
> 1
```

The plot shows Yield vs Temperature with a blue fitted curve and red data points. The plot is zoomed in, showing the x-axis (Temperature) from 50 to 100 and the y-axis (Yield) from 2.2 to 3.4. The plot is titled "Plot Zoom".

The Environment pane shows variables: md1 (List of 12), md2 (List of 12), and yd (15 obs. of ...). The Plots pane shows a small plot of Yield vs Temperature with a blue line.

Residual standard  
Multiple R-squared: 0.6732, Adjusted R-squared: 0.6187  
F-statistic: 12.36 on 2 and 12 DF, p-value: 0.001218



```
xv = seq(50, 100, 0.1)
```

```
yv = predict(md2, data.frame(temp = xv), type = "response")
```

```
lines(xv,yv,col = "blue", lwd = 2)
```

And I am naming it as xv, and I am assigning some values from 50 to 100; why 50 to 100 let us look into the original picture. So, the data starts from 50 and ends at 100. So I want to create numerical values from 50 to 100 with an interval of 0.1; so that is 0.1, the third argument of sequence function. So, this sequence function will create a series of numbers starting from 50 to 100. So, it is a vector and the increment is 0.1, and I am saving that in xv.

Now, I will use this xv, this 50 to 100 sequence as a input to my regressed model; because in the regressed model, I have got a polynomial. So, in that polynomial there is a independent variable, which is temperature. So, this xv will be each value of this xv will be one of the case of those independent variable. So, I will put these individual value of xv and calculate the value of the yv, the yield.

And I will not do it individually; I will do it by one single function, which will do the calculation for me. So, to do that, I will call the predict function; so the predict function is something like this. You tell it the model, which has the equation in it; a model is nothing but a mathematical equation. And then, you give it the input values; so, you want that y equal to a plus bx plus cx square.

So you call predict, give this model and give the different values of x, it will calculate the value of y. So in this case, I will give the model is md2; just now I have created that by regression. And my data should be here, the way they have written the code; my data, input data should be a data frame. So, I am calling data frame function and saying that xv that just now I created a sequence xv is that data, and the name of that data is actually temperature.

So, temperature variable is equal to xv. So in my md2 model, I have a polynomial equation, quadratic equation involving temperature. So, now this function will take this value of xv as the temperature and will predict the yield. And I want to predict, to calculate the response that means I want to calculate the yield in this case; so, I will execute both of these together.

I have done, so you can see here it has xv I have created 50 to 100 at 0.1 interval, so I have 500 data points. Similarly, yv also has 500 data points, so now I want to plot this data. So, what I want to do? I want to plot lines as lines will be a function, which will plot this data, and connect them by a smooth line. So, lines function I will call, the horizontal axis variable is xv, the vertical axis variable is yv. I want the color to be blue and I want the thickness to be 2, execute it.

The figure has been drawn. Let me zoom it, you can easily see that quadratic equation has nicely fitted to my model. This is how you can actually perform polynomial regression by simply using the lm function. Now, I will move to the second part of this lecture, where I will perform nonlinear least square. So, I hope you remember what is nonlinear least square.

If I have a polynomial, it is very easy to perform polynomial regression using simple linear regression approach. But, all nonlinear equation will not be polynomial, there can be some complicated functions. One of the commonest function that you face in biology are exponential growth or exponential decay, you cannot actually perform regression for them by using polynomial regression.

In that case, this approach of nonlinear regression, nonlinear least square becomes very useful. So I will load the data, explain the data, and then it will be obvious that why I want to go for nonlinear least square, and what type of equation I want to fit. So in this data, what we have? We have some experiment where we have measured the growth of plant over the weeks; and growth has been measured in terms of suppose mass. So, I will have a mass versus, the growth versus the time plot; and I want to fit a curve to the smooth curve to that.

(Refer Slide Time: 17:49)

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

```
15 # Read the data ----
16
17 plant <- read.csv("plant.csv")
18
19
20 # Plot the data ----
21
22 plot(plant$week, plant$mass, xlab = "weeks", ylab = "
23       col = "red", pch = 1, cex = 2)
24
25
26 # Perform non-linear least square fit
46
```

The Environment pane on the right shows the 'plant' object with 16 observations and 2 variables. The Console shows the execution of the code:

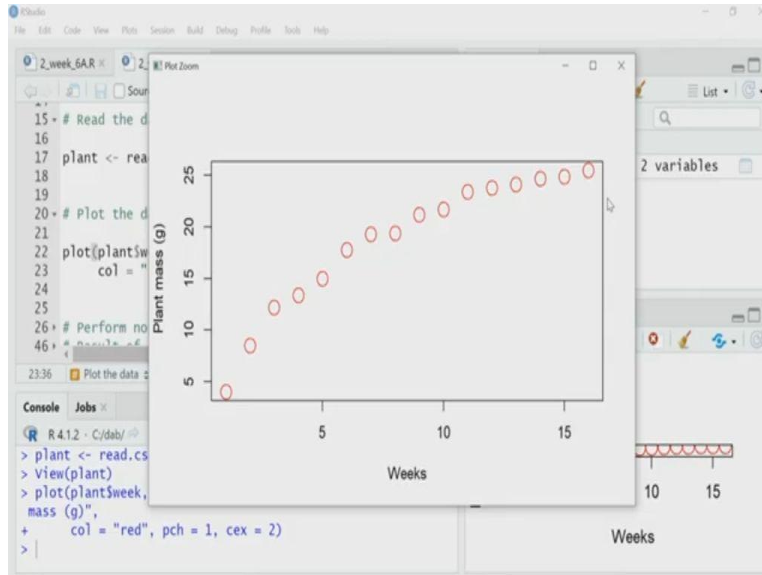
```
> plant <- read.csv("plant.csv")
> view(plant)
> |
```

The screenshot shows the RStudio interface with the 'plant' data frame in view. The data is displayed as follows:

week	mass
11	23.4
12	23.8
13	24.1
14	24.7
15	24.9
16	25.5

The Environment pane on the right shows the 'plant' object with 16 observations and 2 variables. The Console shows the execution of the code:

```
> plant <- read.csv("plant.csv")
> view(plant)
> |
```



```
plant ← read.csv("plant.csv")
```

```
plot(plant$week, plant$mass, xlab = "weeks", ylab = "Plant mass (g)", col = "red", pch = 1, cex = 2)
```

So, let me load the data, that data is in plant dot csv, so it is a csv file. I am calling a read dot csv function and storing that data as a variable plant; let us check that data. This a two column data, the first column is week; week is the first variable. So, it will be my independent variable and the mass, mass is the measure of growth of the plant in this case, and this is the second column; and I have 16 observations.

So, I will plot that data using that simple plot, scatter plot function plot; and in the horizontal axis, I will put the week. And on the vertical axis, I will put the mass; and I will label them as week and mass with color red and other things, as I have done for the earlier figure. So here is the plot, let me zoom in you can see.

How it is behaving? It is behaving the way usually growth of a organism behaves. Initially, there can be sharp rise, sharp growth that is almost like exponential growth; and then it starts saturating. So, I have a growth initially growing very fast, the slope is really high; and then it saturate and saturates almost like at 25.

Maybe if I have, we have done the experiment for a long time. it may have settled at 26, 27 something like that. So, what type of equation we want to fit to this? There are standard

equations which have been used, and there are the reason why we use those to fit this type of growth model. So, the equation that I will fit here is written here.

(Refer Slide Time: 19:28)

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

```
3 # Dataset: growth of a plant over time
4 # Independent variable: week
5 # Dependent variable: Mass (g)
6
7 # Non-linear model:  $mass = a \cdot (1 - \exp(-b \cdot week))$ 
8
9 # Dataset ref: Gregg Hartvigsen, A Primer in Biological
10 # Data Analysis and Visualization Using R
11
12 #####
13
14
15 # Read the data ----
16
```

The console shows the execution of the following commands:

```
> plant <- read.csv("plant.csv")
> View(plant)
> plot(plant$week, plant$mass, xlab = "weeks", ylab = "Plant
+ mass (g)",
+ col = "red", pch = 1, cex = 2)
>
```

The Environment pane shows a data object named 'plant' with 16 observations and 2 variables. The Plots pane displays a scatter plot of Plant mass (g) versus Weeks, with red circles representing the data points.

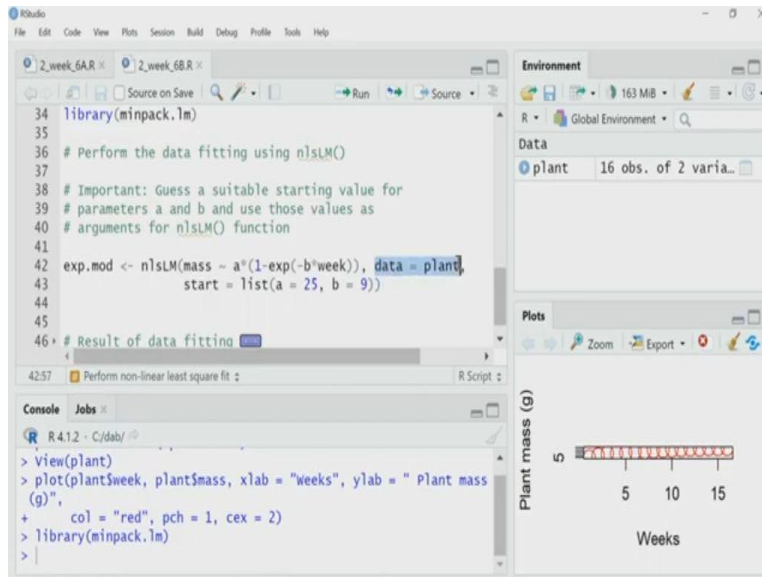
The screenshot shows the RStudio interface with the following code in the editor:

```
24
25
26 # Perform non-linear least square fit ----
27
28 # Using the nlsLM() function of R-package minpack.lm
29 # It uses Levenberg-Marquardt algorithm
30 # Install the package and load it before using
31
32 # Load the package
33
34 library(minpack.lm)
35
36 # Perform the data fitting using nlsLM()
37
```

The console shows the execution of the following commands:

```
> plant <- read.csv("plant.csv")
> View(plant)
> plot(plant$week, plant$mass, xlab = "weeks", ylab = "Plant
+ mass (g)",
+ col = "red", pch = 1, cex = 2)
>
```

The Environment pane shows a data object named 'plant' with 16 observations and 2 variables. The Plots pane displays a scatter plot of Plant mass (g) versus Weeks, with red circles representing the data points.



`library(minpack.lm)`

`exp.mod ← nlsM(mass~ a*(1-exp(-b*week)), data = plant,  
start = list(a = 25, b = 9))`

I want to fit an equation where the mass is the dependent variable. So, mass is equal to  $a$ ,  $a$  is a constant into  $1 - \exp(-b \cdot \text{week})$ ; week is the independent variable. So, this is the form  $y = a(1 - \exp(-bx))$ ; so this type of equation will actually fit very well to a growth phenomenon of an organism.

So, I want to fit this equation to my data that I have shown in the plot just now. And I want to use the nonlinear least squares method; because I cannot use a polynomial regression for this. To perform nonlinear least squares what I will use? I will use a function call `nlsLM`; it is, there is another function built in R, which is called `nls`, nonlinear least square function.

But, I am using a special function which uses the LM algorithm. So, the name of the function is `nlsLM` and it is not by default comes with the R; so, you have to install the `minpack.lm` package. I have already installed it, you know how to install the extra packages, please install it before you execute this script. So, I have installed it, now I will call this, I will load this package in my workspace. So, let me load it, I have loaded the library.

Now, I will call this `nlsLM` function will give some argument to perform nonlinear least squares using the LM algorithm. So, what should be the arguments for this function? The first argument just

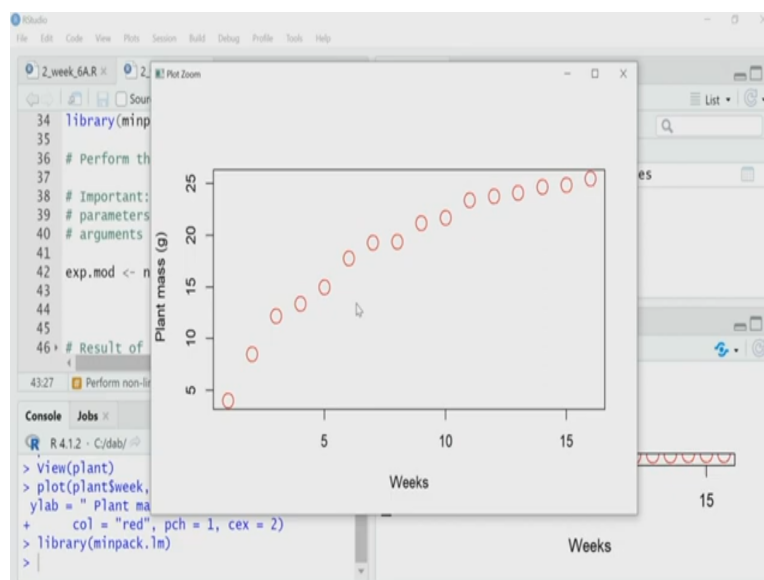


like linear model function is to define the model and I have done that here. I am saying mass tilde, so mass is a dependent variable, and it is a function of a into 1 minus exponential of e to the power minus b into week.

So, this is my, this first argument is my model, and then comma; and then I am defining the data, data is the plant variable. Now, when you are doing nonlinear least square, it is actually using a optimization algorithm. It is not a linear regression that you have learned in the earlier and used them; it is the optimization algorithm it is using. And to perform optimization, you have to specify a starting point, to specify the starting value of the unknown.

So in my model, the unknowns are a and b these two coefficient, so I have to specify these two; and I have to specify them correctly or not correctly, judiciously I will say, so that you know the optimization algorithm converge to a result very easily. Now, you can ask how should I guess this value; yes, in most cases it maybe very difficult to guess. But in this particular example, I can actually guess, actually something I can guess, let me look into this.

(Refer Slide Time: 22:25)



RStudio interface showing the execution of an R script for non-linear least squares fitting.

```

35
36 # Perform the data fitting using nlsLM()
37
38 # Important: Guess a suitable starting value for
39 # parameters a and b and use those values as
40 # arguments for nlsLM() function
41
42 exp.mod <- nlsLM(mass ~ a*(1-exp(-b*week)), data = plant,
43                 start = list(a = 25, b = 9))
44
45
46 # Result of data fitting
47
48 # Plot the fitted model

```

Environment:

- Global Environment
- exp.mod: List of 5
- plant: 16 obs. of 2 varia...

Plots:

Console:

```

R 4.1.2 - C:/dab/
s (g)",
+   col = "red", pch = 1, cex = 2)
> library(minpack.lm)
> exp.mod <- nlsLM(mass ~ a*(1-exp(-b*week)), data = plant,
+                 start = list(a = 25, b = 9))
>

```

43:46 Perform non-linear least square fit

RStudio interface showing the execution of an R script for non-linear least squares fitting, including a summary of the model.

```

40 # arguments for nlsLM() function
41
42 exp.mod <- nlsLM(mass ~ a*(1-exp(-b*week)), data = plant,
43                 start = list(a = 25, b = 9))
44
45
46 # Result of data fitting ----
47
48 summary(exp.mod)
49
50
51
52 # Non-linear fitted model

```

Environment:

- Global Environment
- exp.mod: List of 5
- plant: 16 obs. of 2 varia...

Plots:

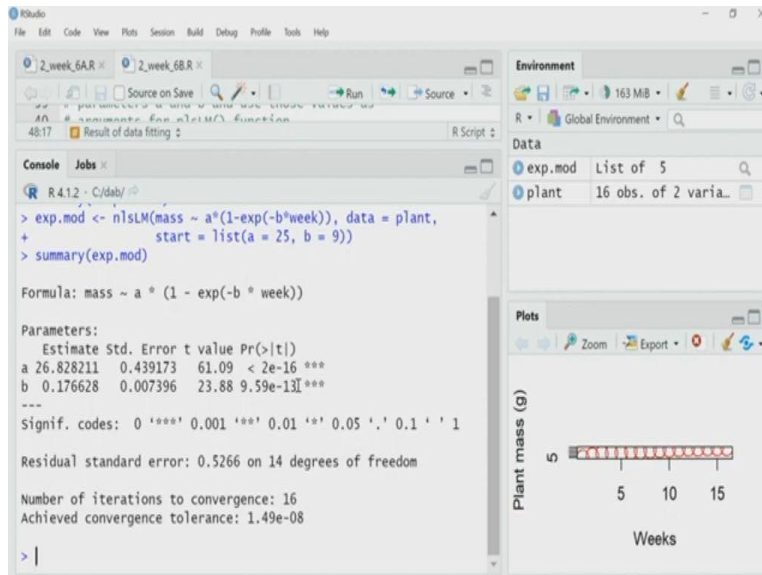
Console:

```

R 4.1.2 - C:/dab/
s (g)",
+   col = "red", pch = 1, cex = 2)
> library(minpack.lm)
> exp.mod <- nlsLM(mass ~ a*(1-exp(-b*week)), data = plant,
+                 start = list(a = 25, b = 9))
>

```

48:17 Result of data fitting



`summary(exp.mod)`

So, see the data is saturating is near 25 remember that. Now, let us look into the equation that we are fitting. My model is  $a \cdot (1 - \exp(-b \cdot \text{week}))$ . So, when week will become very large means suppose week is infinity, then this whole term becomes 0; so, then this mass will be equal to a. So, that is what is happening as time is becoming very big, it is saturating as to 25; so, I am guessing a maybe 25.

And whereas, b I have given a rational size value, you could have given 1, 2, 3, 4, 5 something like that; obviously I will not give 900 or 600 that does not make sense. From experience we have learned maybe the value maybe 2, 3 or 5 or something like that; so I have given 9. So, I will execute that and then what I will do? I will plot that data. So, here I perform that nonlinear least square just perform like a wink, very fast. Now, let me see the result by calling summary.

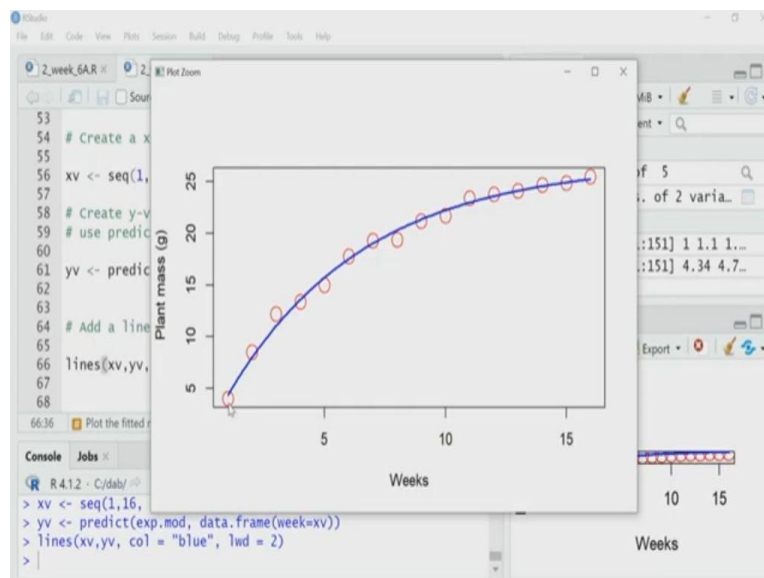
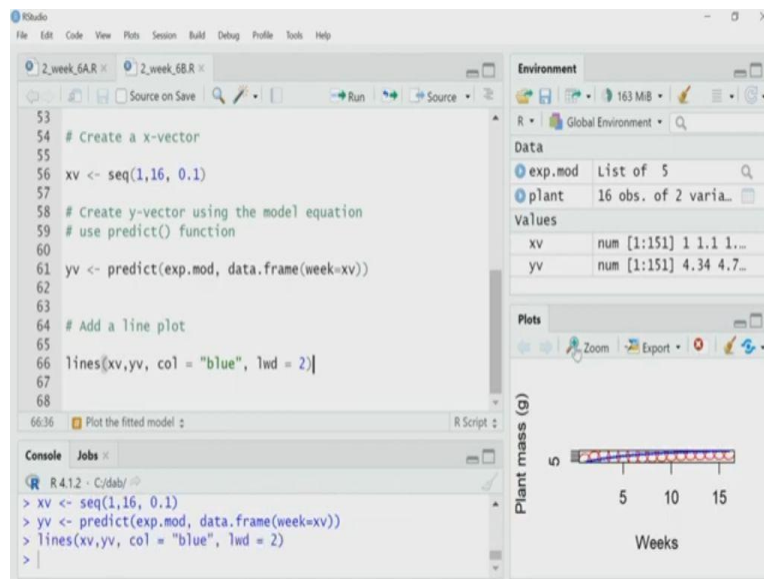
So, my nonlinear model is stored in experiment dot mod, exp dot mod. So, I will call the summary function to see the summary of this result, let us look into the result. So, this is the formula, there is a model; it has fitted that sort of we wanted. Now, the first column give me the estimated parameters. So a, I guess 25 and it is saying 26.8 roughly correct. Our logic was that it is saturating near 25, so it is 26.8 roughly 27, and b is 0.17.

It has also performed statistical test and you can see the probabilities are given for the null hypothesis, and you can reject the null hypothesis in both cases. That means both the coefficient a and b which are estimated are statistically significant. This information is good enough for us

to now proceed and plot the result. So, I have the nonlinear model now, so I will plot that on an overlay over the existing diagram.

So, in the previous example, we created two vectors; the one is the horizontal axis vector, x vector, where I have different data points. And then we use those data point as an input to predict the vertical axis data; so, we will do the same thing here. So, if you look into our data, it starts from somewhere around 1 and ends around 15 or 16, something like that.

(Refer Slide Time: 25:19)



$xv = seq(1, 16, 0.1)$

```
yv = predict(exp.mod, data.frame(week = xv))
```

```
lines(xv,yv,col = "blue", lwd = 2)
```

So, what I am doing? I am using the sequence function to generate a series of number 1 to 16 with an increment a 0.1, and I am storing that as the xv variable. And now again, I am calling the predict function and I am giving the model that just now I created by nonlinear least square as a model. So, that model has already this exponential equation in built there.

And I am creating a data frame where I am saying this xv vector that I created is equal to the week variable. So, every data point of this xv variable will be used as a value of week and will be used to predict the value of the mass of the plant or y, the vertical axis data from this nonlinear least squares model.

So, and then the whole thing will be stored in the yv variable. So, I execute that, I have not executed the xv; so I have to first create the xv, xv is created, now yv. So, I will plot these two as a line, smooth line. So for that, I will call the lines function, I will have xv and yv as a horizontal and vertical axis data correspond respectively, color will be blue; and I want a line thickness lwd equal to 2.

So, let us zoom into the plot, you can see so nicely, this blue line which has been created from the nonlinear least square model is fitting with the data. So, that is all for this particular lecture. In this we have performed two types of nonlinear regression; one is polynomial regression, where we have used the same old linear model function to perform polynomial regression.

And I have also shown a nonlinear least square using nlsLM function for exponential model. I hope you will try this script and with a different data set and practice them yourself. Thank you for joining me today and learning something new.