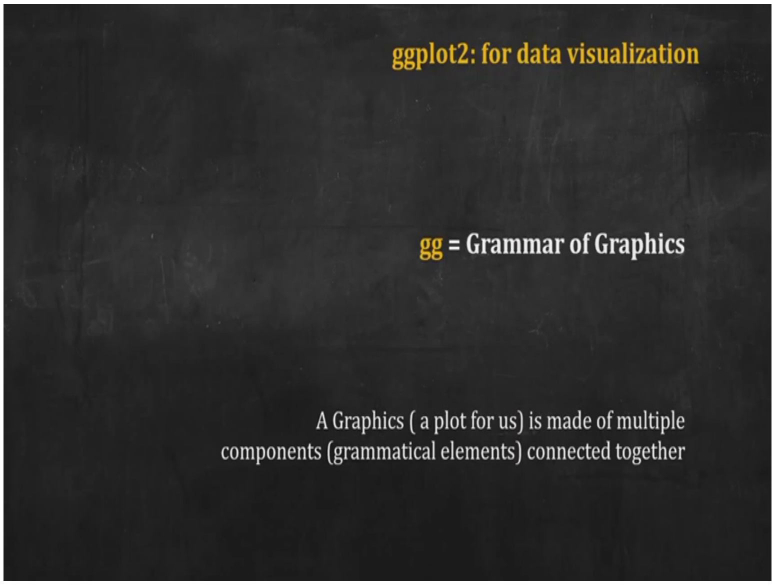
Data Analysis for Biologists
Professor Biplab Bose
Department of Biosciences and Bioengineering
Mehta Family School of Data Sciences and Artificial Intelligence
Indian Institute of Technology, Guwahati
Lecture 24
Data visualization using ggplot2 - I

Welcome back. In last couple of lectures, we have learned how to visualize data using scatter plot, bar plot something like those using R. And in those example I have mostly used the base R function that is the R functions that comes by default with your R installation, but there are certain dedicated R packages for high end data visualization, for example, Plotly, ggplot2.

In this lecture, I will discuss about ggplot2, which can be used for generating high quality visualization, graphs, which are of publication quality, its very versatile and very powerful.

(Refer Slide Time: 01:24)



ggplot2: for data visualization

gg = Grammar of Graphics

A Graphics (a plot for us) is made of multiple components (grammatical elements) connected together

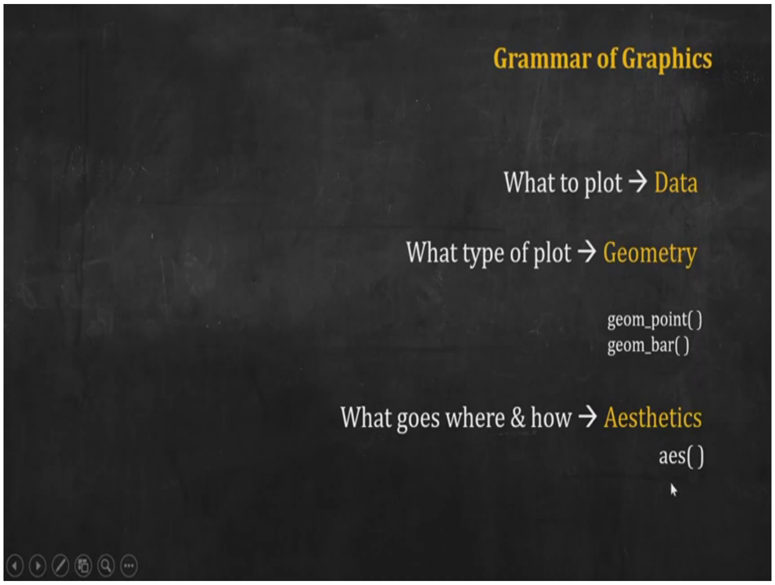
gg = Grammar of Graphics

The gg in ggplot2 stands for grammar of graphics. Every human language has some certain grammar. For example, in English we have noun, pronoun, adjectives, verbs, adverbs, and when you construct a sentence, these elements of grammar this noun, pronoun, verbs, adverbs, these things come and get arranged in a particular order, then only you get a meaningful sentence. The idea, the principle behind the grammar of graphic is that if I take a graphic, in our case, the graphic is a plot, a diagram based on your data; it also has some basic elements.

I can break down the graphics into our visualization into different elements. And these elements are like grammatical element, they have a purpose and they have structures and they all come together they bound together to give me a particular plot or a graphic visualization. So ggplot2 uses this principle of grammar of graphics that is why it is named as ggplot2.

But the grammar of graphics is a broad principle and not just ggplot2, there are many other tools for visualization of data, which are very flexible, scalable and can interactive data visualization, those uses the grammar of graphics. However, in this lecture, we will focus only on ggplot2. So, let me take you into some 3 key elements of the grammar of graphics that is used in ggplot2.

(Refer Slide Time: 03:00)



Grammar of Graphics

What to plot → Data

What type of plot → Geometry

geom_point( )
geom_bar( )

What goes where & how → Aesthetics

aes( )

The first one is obviously the data that should be the first, the foremost element of your graphics, the graphic visualization in your plot. So, you should know, you should tell the machine what to plot. Once you have said that the second element come that I have to tell what type of plot, what type of graphics I want. In case of ggplot2, we call it geometry, and in short, it is called geom.

So, for example, in ggplot2 you have functions called geom underscore point, if you call this function, you are essentially telling R, that see I want a scatter plot. So, this scatter, the dots, the circles or any other symbol used to create the scatter plot is the geometry. And for example, take the other examples, geom underscore bar.

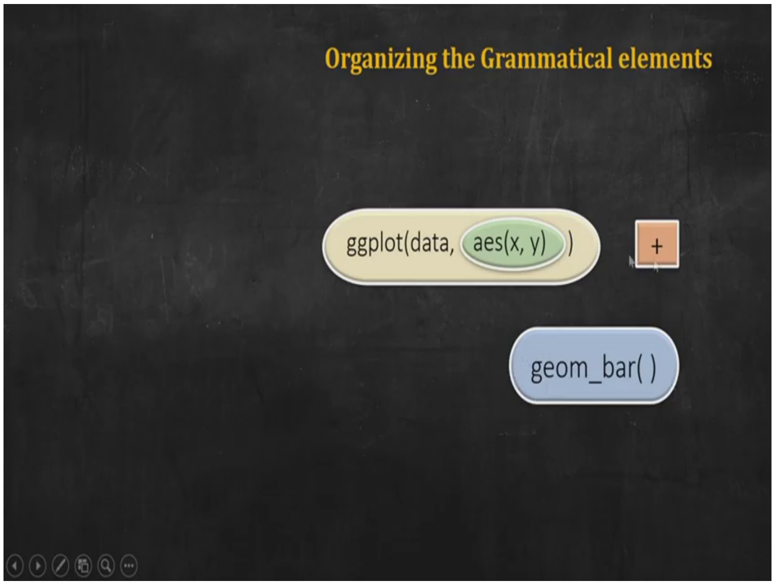
This function is a geometry function, it will add a layer of geometric objects to your graphics and it will tell R that, we want to represent the data as bar plot. Now, you have told the data, you have given the data, defined the data, you have defined the geometry. Next comes the aesthetic part, this is the third key element in grammar of graphics for ggplot2.

What is aesthetic? You have to tell the R, that this is data, this is the type of plot I want, but you have to tell which one should go where, for example, you have three variables in your data, then you have to tell which, and you want to make a scatter plot, then you have to tell which variable should go in X axis and which variable should go in Y axis.

This comes under aesthetic. So essentially what aesthetic functions are doing is that it is mapping your data on the geometric object, geometric layers that you are creating. You can also add other arguments in this aesthetic function, for example, the feel of the symbol, type of symbol and all these things. So in general ggplot2 aes stands for the aesthetic.

So aes function is actually a function for aesthetic. So as I said in any language take English which we have different grammatical elements, a noun, pronoun, verb, adverb, you do not use them abruptly anywhere, you have a structure, you join them together in a particular order to create a sentence. Similarly, in ggplot2, also you have to join these three elements - data, aesthetic and geometry in a particular fashion.

(Refer Slide Time: 05:38)



Organizing the Grammatical elements

ggplot(data, aes(x, y)) +

geom_bar( )

ggplot(data, aes(x, y)) +

geom_bar()

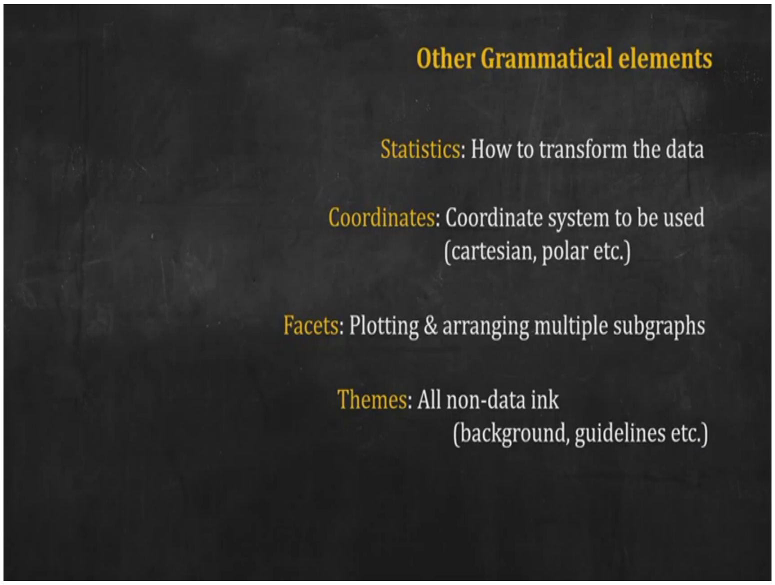
So let us see how we do it, I will give a generic example. Suppose I have a data and I want to create a bar plot. So the first thing that you have to do, you have to call the ggplot function. Remember, although the package is called ggplot2, the name of the function is ggplot, and this is the heart of ggplot2, so you have to anyway call this function, ggplot2 function.

And in as argument what I have given to that ggplot function, I have given the data, the key element of your graphics as one argument, and then I have given, called the aesthetic function as a argument for this ggplot2 function, so that here I am telling which variable will go in X, which variable will go in Y.

And then I can add the geometric element, the geom underscore bar here because I want to create a bar plot by simply putting an addition symbol. Remember, I have to put the addition symbol just after this ggplot function, not before geom bar. So, I put a addition symbol here and then I call geom bar underscore bar a function to tell the machine, tell R that this is my data, this is the geometry and this is the aesthetic.

So what is the structure, call ggplot function, within ggplot function, you declare the data, you can call the aesthetic function, and then you add the geometry function. In the geometry function also you can add aesthetic function as an argument, if required. We will discuss those separately.

(Refer Slide Time: 07:29)



Other Grammatical elements

Statistics: How to transform the data

Coordinates: Coordinate system to be used (cartesian, polar etc.)

Facets: Plotting & arranging multiple subgraphs

Themes: All non-data ink (background, guidelines etc.)

Now, apart from these three key elements in grammar of graphics for ggplot2, there are a couple of others key elements also. One of them is called statistics. See when I was talking previously about creating a bar plot from a data set, and the example also that we will discuss today on scatter plot, today, there we will use the raw data, but some time you want to transform the data and create some statistics out of it, and then plot that transformed data.

So in that case, you have to use the statistics element. So there will be functions for that, we have to call those, you simply add them by this addition symbol. In the most of the examples that we have discussed in our graphic visualization, we have used Cartesian coordinate, but that does not mean that always you have to use Cartesian coordinate, in some cases, you may have polar coordinate also. So there can be different coordinate systems.

So you have to tell R, that, this is the coordinate system I want to use. So that is why coordinates is another key element, which I can assign a particular coordinate system to my graphics. Then come one interesting thing called facets in ggplot2 what is that? In many publications, you may have seen that one figure has multiple sub graphs, so you can create those sub graphs separately. And then you can use a image editor to stick these 4 or 5 different sub graphs and to create a composite diagram.

But the facet element in ggplot2 allow me to create the sub graphs using some part of the data and to create different plots in one single diagram. So I have not to take the trouble to separately assemble then after I have drawn created the images. The last one, which comes very handy, and we will use it regularly is called themes. And as I have written here, all non-data ink goes there, what do I mean by non-data ink?

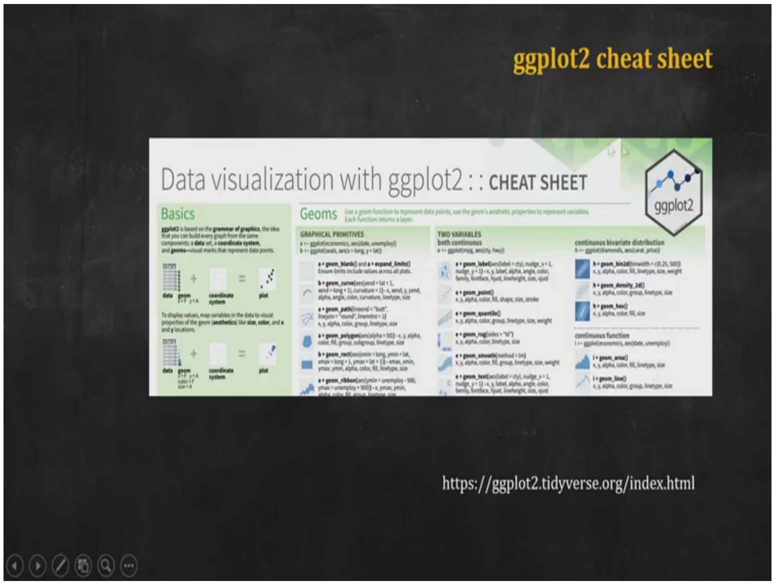
See, when you create a diagram a plot, the key thing is obviously, how you visualize your data, but at the same time beyond that, there can be something else also for example, what should be the color of the background of your bar plot? You may have some black color bars and you can put a yellow color as a background, white color as a background.

So similarly, there can be guidelines also, grids that helps him to visualize the data. So these are not related to the data itself. So, these are additional thing that you add on the graphics on your plot to make it more easily readable, so that you can easily draw attention to people. So, and aesthetically it looks good, something like those.

So, all these things comes under the elements of themes, there are some inbuilt themes in ggplot2 which we can use, you can build your own theme define the themes and use that also. Now, all these things seems a bit complicated, is not it? If you think of English language, there are 1000s of grammatical rule, lakhs of words. So, in that sense, learning English and to be able to speak it correctly is also a difficult job.

But obviously, you and me all do not know all the grammars of English, we know, we are mostly concerned about the grammar which we have, bare minimum grammar that we have to know to communicate, read and write and speak. Now, if you have to do some complicated sentence construction or if you realize you are doing, creating a sentence which has a particular grammatical error, you may sometime go back and look into a grammar book or ask somebody who is more learned than you.

(Refer Slide Time: 11:32)



ggplot2 cheat sheet

Data visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the grammar of graphics, the idea that you can build every graph from the same components: a data set, a coordinate system, and geoms—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (aesthetics) like size, color, and x and y locations.

Geoms Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank() and a + expand_limits()
Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)) - x, y, alpha, color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

TWO VARIABLES
both continuous
e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl")
x, y, alpha, color, linetype, size

e + geom_smooth(method = lm)
x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

continuous bivariate distribution
h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d()
x, y, alpha, color, group, linetype, size

h + geom_hex()
x, y, alpha, color, fill, size

continuous function
i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

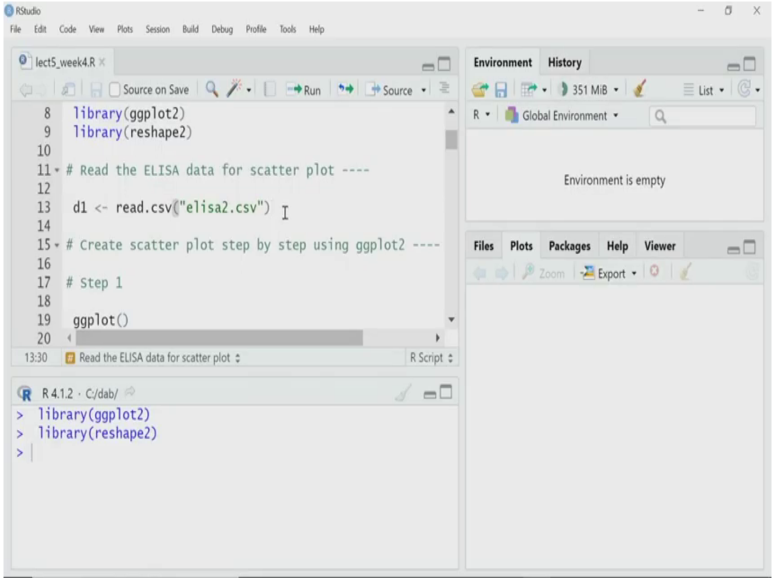
https://ggplot2.tidyverse.org/index.html

Similarly, in case of ggplot2, as it is based on grammar of graphics, it may seems intimidating initially that I have to know so many things, how do should I do that, it is not like that, most of the things that are required for day to day graphics, you will get habituated to it is as you use.

Apart from that, there is a very handy thing we call cheat sheet ggplot2 cheat sheet, you can go to ggplot2 website and you can download that PDF file, it is a two page file, you can simply print it and keep it on your desk and graphically and visually representing provide you information, what type of function what type of grammatical element has to be used and their purposes.

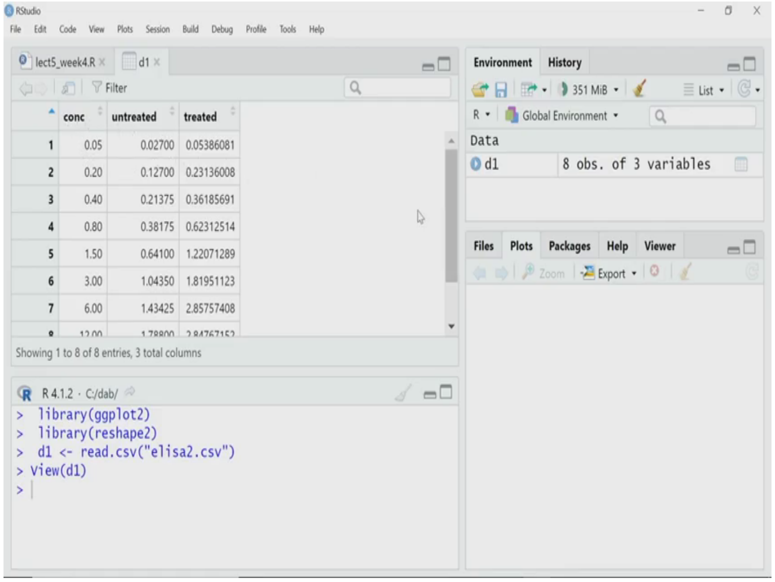
So, if you want to create a particular type of graphic element in your graph, you can look into the cheat sheet and get an idea what you should use which function you should use. So, now, till now, what I have done, I have given a brief description of ggplot2, the concept of grammar of graphics. And I have discussed about some of the elements which are using grammar of graphics in ggplot2.

(Refer Slide Time: 12:27)



```
library(ggplot2)
library(reshape2)

# Read the ELISA data for scatter plot ----

d1 <- read.csv("elisa2.csv")

# Create scatter plot step by step using ggplot2 ----

# Step 1

ggplot()
```

```
R 4.1.2 · C:/dab/
> library(ggplot2)
> library(reshape2)
>
```

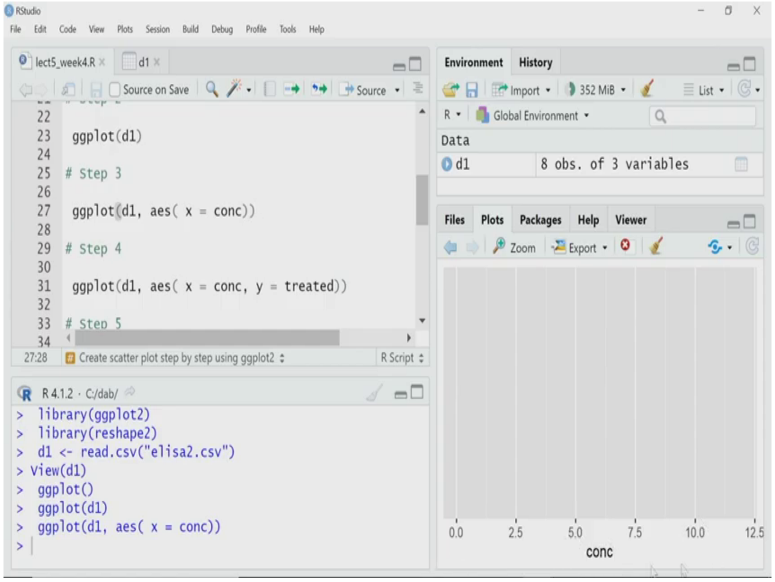


	conc	untreated	treated
1	0.05	0.02700	0.05386081
2	0.20	0.12700	0.23136008
3	0.40	0.21375	0.36185691
4	0.80	0.38175	0.62312514
5	1.50	0.64100	1.22071289
6	3.00	1.04350	1.81951123
7	6.00	1.43425	2.85757408
8	12.00	1.78800	2.84767152

Showing 1 to 8 of 8 entries, 3 total columns

```
R 4.1.2 · C:/dab/
> library(ggplot2)
> library(reshape2)
> d1 <- read.csv("elisa2.csv")
> View(d1)
>
```

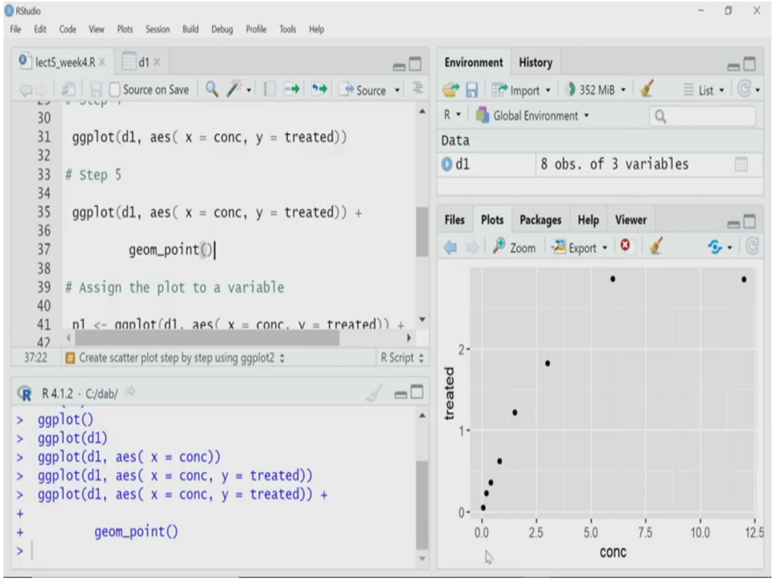
Data
d1 8 obs. of 3 variables



```
ggplot(d1)

# Step 3

ggplot(d1, aes( x = conc))

# Step 4

ggplot(d1, aes( x = conc, y = treated))

# Step 5
```

Data
d1 8 obs. of 3 variables

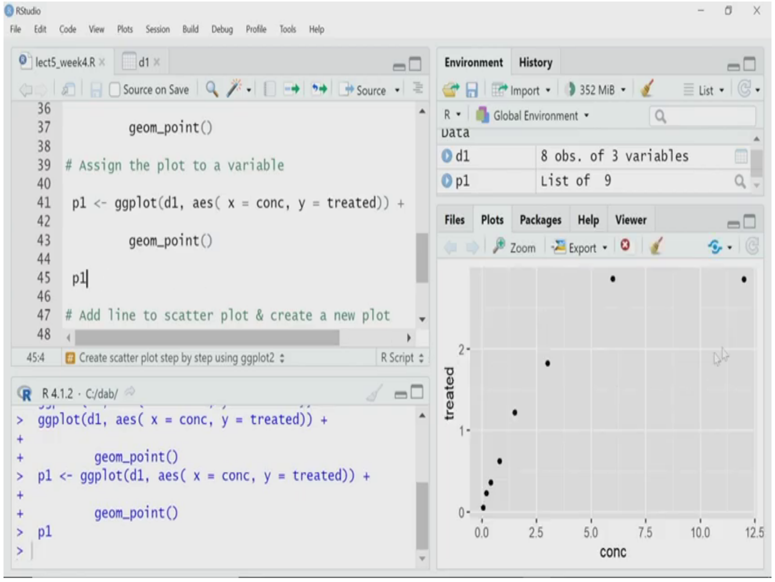
```
> library(ggplot2)
> library(reshape2)
> d1 <- read.csv("elisa2.csv")
> View(d1)
> ggplot()
> ggplot(d1)
> ggplot(d1, aes( x = conc))
>
```



```
ggplot(d1, aes( x = conc, y = treated))

# Step 5

ggplot(d1, aes( x = conc, y = treated)) +

        geom_point()

# Assign the plot to a variable

p1 <- ggplot(d1, aes( x = conc, y = treated)) +
```

Data
d1 8 obs. of 3 variables

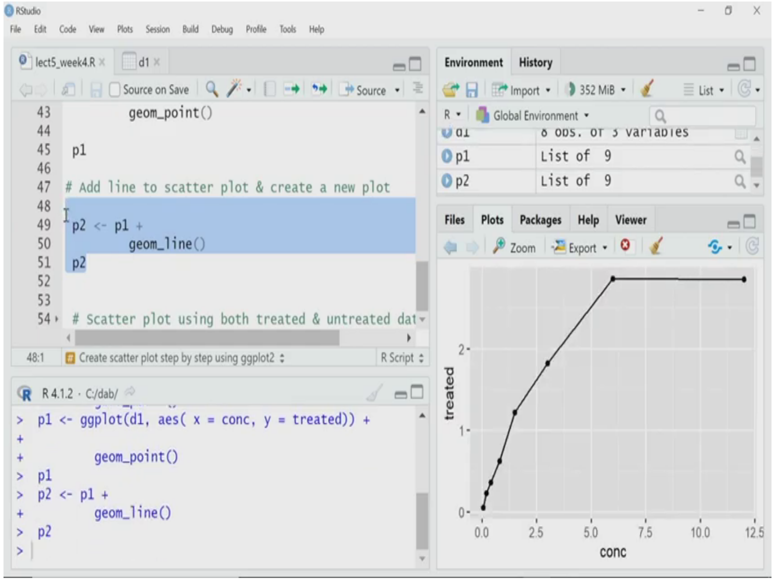
```
> ggplot()
> ggplot(d1)
> ggplot(d1, aes( x = conc))
> ggplot(d1, aes( x = conc, y = treated))
> ggplot(d1, aes( x = conc, y = treated)) +
+
+         geom_point()
>
```



```
        geom_point()

# Assign the plot to a variable

 p1 <- ggplot(d1, aes( x = conc, y = treated)) +

        geom_point()

 p1

# Add line to scatter plot & create a new plot
```

```
> ggplot(d1, aes( x = conc, y = treated)) +
+
+         geom_point()
> p1 <- ggplot(d1, aes( x = conc, y = treated)) +
+
+         geom_point()
> p1
>
```

```
Data
d1    8 obs. of 3 variables
p1    List of  9
```



```
        geom_point()

 p1

# Add line to scatter plot & create a new plot

 p2 <- p1 +
        geom_line()
 p2


 # Scatter plot using both treated & untreated dat
```

```
> p1 <- ggplot(d1, aes( x = conc, y = treated)) +
+
+         geom_point()
> p1
> p2 <- p1 +
+         geom_line()
> p2
>
```

```
d1    8 obs. of 3 variables
p1    List of  9
p2    List of  9
```

library(ggplot2)

library(reshape2)

d1 ← read.csv("elisa2.csv”)

# Step 1

ggplot()

# Step 2

ggplot(d1)

```
# Step 3
ggplot(d1, aes(x = conc))


# Step 4
ggplot(d1, aes(x = conc, y = treated))


# Step 5
ggplot(d1, aes(x = conc, y = treated)) +
geom_point()


p1 ← ggplot(d1, aes(x = conc, y = treated)) +
geom_point()
p1


p2 ← p1 +
        geom_line()
p2
```

Now I will move into R script and give a demonstration that how these elements are used to create a simple scatter plot. In this R script, I will be using 2 libraries, 2 packages, ggplot2 and reshape2. So if you have not installed them earlier, please install ggplot2 and reshape2 before you use this. I have already done that. So let me load those first before I proceed into something else. So I have loaded them.

Now earlier, I have used a Elisa data set to create scatter plot. That time I have used the default functions in R to create the scatter plot. So I will again read that file and use that data and demonstrate different aspects of these grammar of graphics in ggplot2. So here I am reading this data Elisa2 dot CSV file and storing that data in d2, let us check d2 data what I have. It is the experiment, Elisa experiment where the samples are different concentrations starting from 0.05 to 12.

And untreated and treated samples are 2, untreated and treated samples and their absorbance are given, the values are given. So the 3 column data set and I want to create a scatter plot out of it. So suppose initially, I want to create a scatter plot for these treated or untreated1 sample only. So let us see how should we do? Let us do this thing step by step. As I said, at the heart of ggplot2 is the ggplot function.

So let me first call the ggplot function without doing anything. So I will call here ggplot, plot function. If you can notice carefully in this plot tab, a gray box has been created. So ggplot function is empty I have not given it any arguments. So this function does not know anything, what is the data what is the aesthetic nothing. So what it has done it has just created a background in which a graphic can be created.

Now, let me move to next step. Now I will provide it the data. If you remember in grammar of graphics. The first key element is the data. And I am using d1 as the data. So I am written ggplot and as a argument I have given d1. So I run that, if you see we have created a new image but nothing has changed. It is still a gray box because what I have done I have given ggplot the data, but I have not told it how to create a plot and what is the, which variable will go to which axis all this information. So, it could not create anything.

Now, let me move to step three, here, I add an additional argument to my ggplot, what argument? I am calling the aesthetic function and I am saying x equal to, that means the x coordinate should be equal to concentration conc, if I look into data d1 I have the first column the variable is conc, so that is the concentration I am saying it should go to the x axis, if you run this you can see we do not have a scatter plot yet, but what do we have, we have some scaling and concentration is written in the horizontal line.

So, the horizontal line is now scaled from 0 to 12.5. If I go to my data, I can see my lower value is 0.05 And the highest value is 12. So, ggplot2 automatically has scaled it from 0 to 12.5. That is what it has done because you have only provided information on the x axis. In the next line, what

I am doing now, I will provide the y axis information also in the aesthetic function. So, now, I have edited the aesthetic function I am writing x equal to concentration and y equal to treated.

Now, if I execute this a new image is created, which has a gray background and some grid line and now, you can see the vertical axis has also some scaling some numbers are there and it is marked as treated. So, what I have done till now, I have called ggplot and I have provided it the data and I have given it aesthetic information, what should get mapped to where but till now, I have not got a scatter plot.

I have not got the scatter plot, because I have not told ggplot2 that I want a scatter plot, remember that information is part of the geometry element, I have to specify what type of plot I want, bar plot, pie chart, box plot, scatter plot, whatever, I have to specify that as a geometric element. So in the next line, I do that. So here what I am doing, in the step five I have edited the code and I have written ggplot2.

Again, I am calling that function giving the data, aesthetic remains same, I have already defined x axis and y axis and then note here, I have added a addition symbol, plus and then I am saying geom point, geometry point, geom point. Now this geom point function as I am calling here as an additional layer, ggplot2 will understand that now I have to create a scatter plot.

So, if I execute it, now, you see you have a scatter plot, the concentration is varying along the horizontal axis, the absorbance of the treated sample is varying on the vertical axis and each of these black dots, black circle is my data point. So, you have seen here by step by step, I have added all the three key elements of ggplot2 while using ggplot function. Till now what I was doing is that I was calling ggplot function just like that.

Now, what I will do? I will call ggplot function, but I will not plot it immediately, I will assign whatever ggplot function is creating to a variable with the plot name. Why I want to do it? Because then what we can happen that, I can obviously draw that figure calling that particular variable. At the same time I can step by step add a new layer to my existing graph by simply adding a particular function with the addition symbol. I will show it.

So what I am doing here, I am calling the ggplot function with the aesthetic information and the geometric information. But now this time I am assigning this whole thing using this assign symbol to a variable P1. And then in the next line I am printing that that P1, so that I can visualize it. Let me clean this plot tab. So now if I execute this first thing that ggplot is assigning something to P1.

You can see it has been executed but nothing has been plotted in the Plot tab. If I now click execute the P1 command then I will get the plot, fine. Now suppose I want to add another layer to it, another geometric layer it, I want to add lines between these dots, straight lines, So I will create a line plot that are rather than just a scatter plot.

So what I can do now, rather than writing the whole code what I can say, take P1, P1 is my existing plot, and then add with the Addition symbol geom underscore line. So what I am adding? I am adding the geometric layer of line plot. So I am adding geom underscore line function. And I use just some addition symbol after the my previous plot P1, and I assigned this whole thing to a new variable called P2.

So, this is my new plot P2, and in the next line, I print that. So if I execute both of these, now, you see I have a new plot, where the dots are, there just like a scatter plot and they are connected by a line just like in a line plot. So, this is how you actually can keep on building upon a existing plot, you first get P1, then add some extra element that becomes P2, then you add another element it become P3. So step by step, you can build it and whenever you want to stop or you want to edit, you can edit a small part of it rather than editing the whole code.

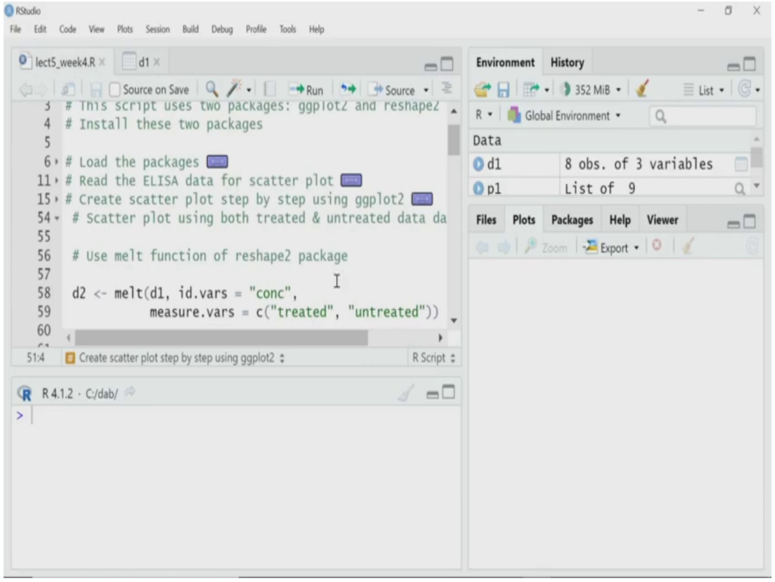
Till now, what I have tried to show you is that how these three key element of data, aesthetic and geometry is used to create a plot. But if you remember our data, in our data, we have two types of sample treated and untreated. The scatter plot I have created here was only for the treated sample. Now, I want to plot both the data in a single scatter plot. Let me show you how I will do that using again ggplot2. I will first go back to the data d1 what I have.

Now, in this data set, the way I have arranged this is what we are more habituated to arrange data when we work on Excel or some other spreadsheet, you have one variable, which is the concentration and that goes in your horizontal axis, x axis, and you have two sample untreated and treated, and you want them to be in the, corresponding value should be in the vertical axis that you want. But ggplot2 cannot use these data in this fashion.

So I have to transform the, rearrange these data, I have to recast these data in a particular way. So that ggplot2 can use that to create a scatter plot, where in one single scatter plot, both the sample data should be there. I will do it step by step. And while I do it, I will explain what type of recasting I want to do.

Now, I can do this recasting manually, because we have only 8 data point somewhere, I could do it manually, but I will not do that I will show you a better way to do it, because sometimes new data can be large, these recasting or reshaping can be done by the tools present in the reshape2 package and the function of reshape2 that I will use for recasting my data is called melt. So it will melt variables to create a new variable data, a data set.

(Refer Slide Time: 23:23)



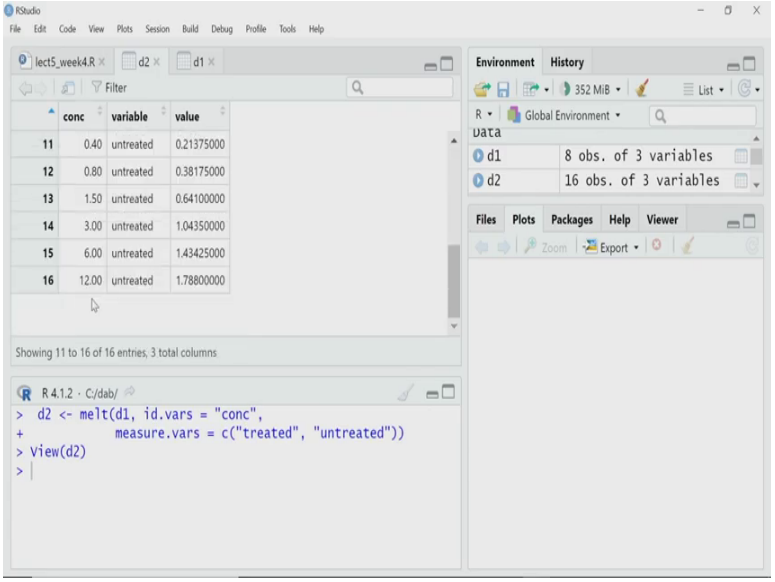
```
# This script uses two packages: ggplot2 and reshape2
# Install these two packages

# Load the packages
# Read the ELISA data for scatter plot
# Create scatter plot step by step using ggplot2
 # Scatter plot using both treated & untreated data da

 # Use melt function of reshape2 package

 d2 <- melt(d1, id.vars = "conc",
            measure.vars = c("treated", "untreated"))
```

Create scatter plot step by step using ggplot2

Data

d1	8 obs. of 3 variables
p1	List of 9

```
R 4.1.2 · C:/dab/
>
```



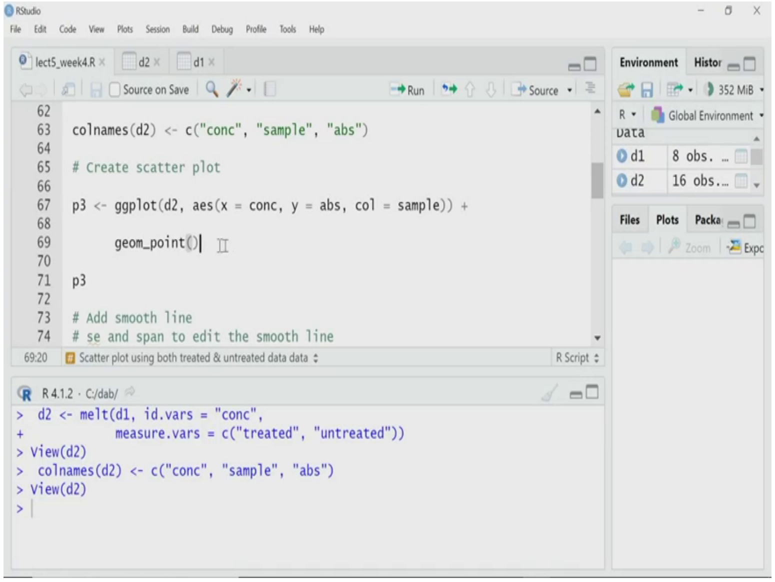
	conc	variable	value
11	0.40	untreated	0.21375000
12	0.80	untreated	0.38175000
13	1.50	untreated	0.64100000
14	3.00	untreated	1.04350000
15	6.00	untreated	1.43425000
16	12.00	untreated	1.78800000

Showing 11 to 16 of 16 entries, 3 total columns

Data

d1	8 obs. of 3 variables
d2	16 obs. of 3 variables

```
R 4.1.2 · C:/dab/
> d2 <- melt(d1, id.vars = "conc",
+            measure.vars = c("treated", "untreated"))
> View(d2)
>
```

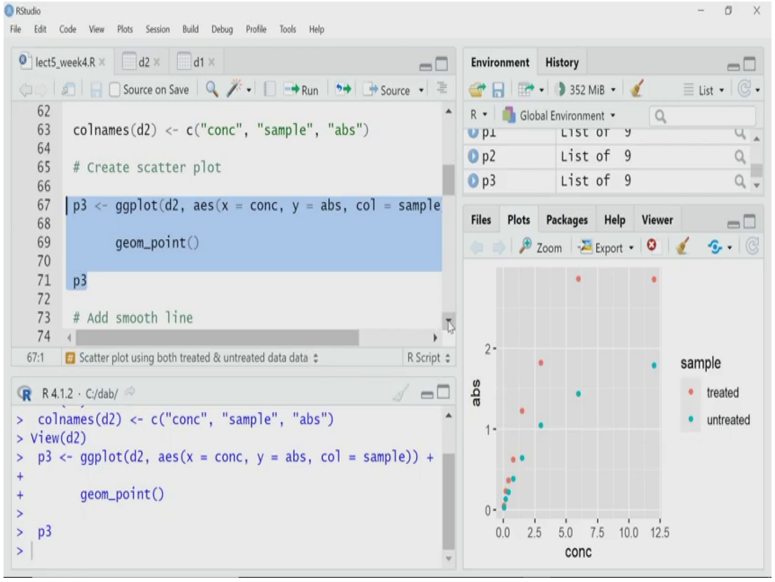


```
colnames(d2) <- c("conc", "sample", "abs")

# Create scatter plot

p3 <- ggplot(d2, aes(x = conc, y = abs, col = sample)) +

      geom_point()

p3

# Add smooth line
# se and span to edit the smooth line
```

69:20 Scatter plot using both treated & untreated data data R Script

Data	
d1	8 obs. …
d2	16 obs.…

```
> d2 <- melt(d1, id.vars = "conc",
+            measure.vars = c("treated", "untreated"))
> View(d2)
> colnames(d2) <- c("conc", "sample", "abs")
> View(d2)
>
```

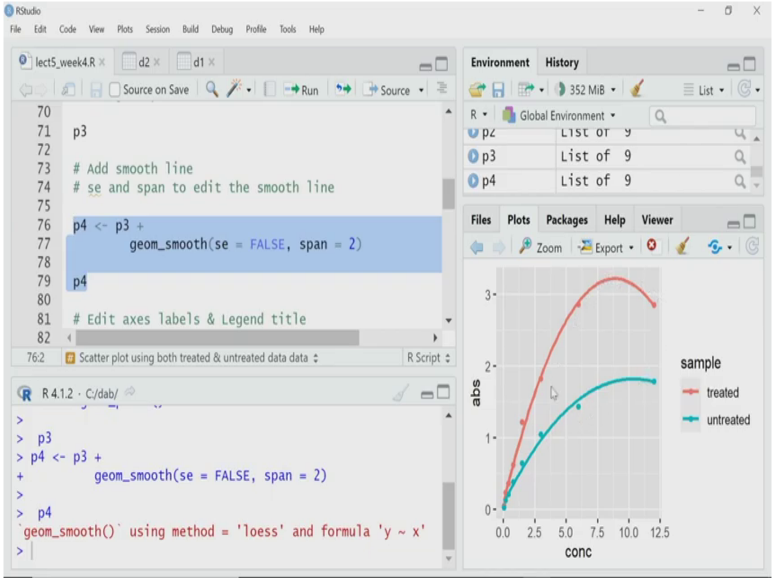


```
colnames(d2) <- c("conc", "sample", "abs")

# Create scatter plot

p3 <- ggplot(d2, aes(x = conc, y = abs, col = sample

      geom_point()

p3

# Add smooth line
```

67:1 Scatter plot using both treated & untreated data data R Script

p1	List of 9
p2	List of 9
p3	List of 9

```
> colnames(d2) <- c("conc", "sample", "abs")
> View(d2)
> p3 <- ggplot(d2, aes(x = conc, y = abs, col = sample)) +
+
+       geom_point()
>
> p3
>
```



```
p3

# Add smooth line
# se and span to edit the smooth line

p4 <- p3 +
        geom_smooth(se = FALSE, span = 2)

p4

# Edit axes labels & Legend title
```

p2	List of 9
p3	List of 9
p4	List of 9

```
>
>  p3
> p4 <- p3 +
+          geom_smooth(se = FALSE, span = 2)
>
>  p4
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> |
```

```
d2 ← melt(d1, id.vars = “conc”,
        measure.vars = c(“treated”, “untreated”)
        colnames(d2) ← c(“conc”, “sample”, “abs”)

p3 ← ggplot(d2, aes(x = conc, y = abs, col = sample)) +
        geom_point()
p3

p4 ← p3 +
        geom_smooth(se = FALSE, span = 2)
p4
```

So, what I will do, I have already called the library, called the package reshape2 at the very beginning of running this script. So now what I am doing here, I am calling the melt function from reshape2 and I am providing d1 as my first arguments. So this is my raw data that I want to recast, reshape into any particular way so that ggplot2 can use it. Then what I am saying id dot variables, I can have multiple variables as id’s, but we have only one. I am saying the concentration column is my id identification variable.

That means I am saying that, these are the variable with respect to which other variables should be rearranged. So, you should not rearrange this concentration variable. And then what I am saying the major variables are treated and untreated. That means this untreated and treated, these two are measured variable and each of these measured variable have these values. Fine, it may look a bit confusing now, let me execute it, create a new data set, it will be very clear to you.

So now if I execute it, I create a new data set d2. Let us look into that. So now your d2 you see you have the first column called concentration. So you have all these concentration. But notice that they are, initially I have 8 rows because I have 8 concentration, but now I have 16. So if you notice carefully each of them has been taken twice, so 0.5 to 12 and then again 0.5 to 12. Why it has been done that way?

Because in the variable part, what it has done? Variable was earlier in the column header, the name, the measured thing, the untreated and treated, those were the name of the header. Now, what has happened? They have become factors. So, these first 8 measurements are for treated and the next 8 measurement are for untreated. And the corresponding values of absorbance that has come is in the last column.

So, when the sample is treated and the concentration was 0.05, the value is 0.05386. Similarly, when the untreated sample has concentration 0.4, the reading is 0.213. So, you can now understand how these melt function in reshape2 has re-casted the data. This is usually in our terminology it is called, this form is called wide form, whereas this is called the long form of data. So, now, I have re-casted the data, rearranged the data so that it ggplot2 can use it.

Before I move into what it has happened is that the concentration column has a nice name easily one can you understand that it is a concentration, but variable values these are the less meaning, these are not really meaningful. So, what I will do I will change these column headings so that we can easily understand what they represent.

So, what I am doing here again I am using the call names function. So, column names of d2 I am changing with what, I have created a vector, the first thing will be concentration, second thing will be sample, third thing will be absorbance. So, if I execute it, now, if I go to d2 I can see the first column is concentration, second column is sample the last column is absorbance, good enough.

Now, I am ready to go and work with ggplot2 again and to create a scatter plot where both treated and untreated sample data will be shown. So, what I am doing here, the first thing is I am creating a scatter plot. So, again it will come that I have to call ggplot, use aesthetic data and geometry. So, I am calling ggplot function I am giving dd2 now as data, in the aesthetic I am saying concentration should go in X axis absorbance should go in Y axis.

Remember in absorbance, absorbance of both treated and untreated data is there now in my this recasted data in d2. So, absorbance goes in Y axis and this col represent color, this another aesthetic element what I am saying that the color the dots because I will have a scatter plot, color the same symbols based on the information in sample.

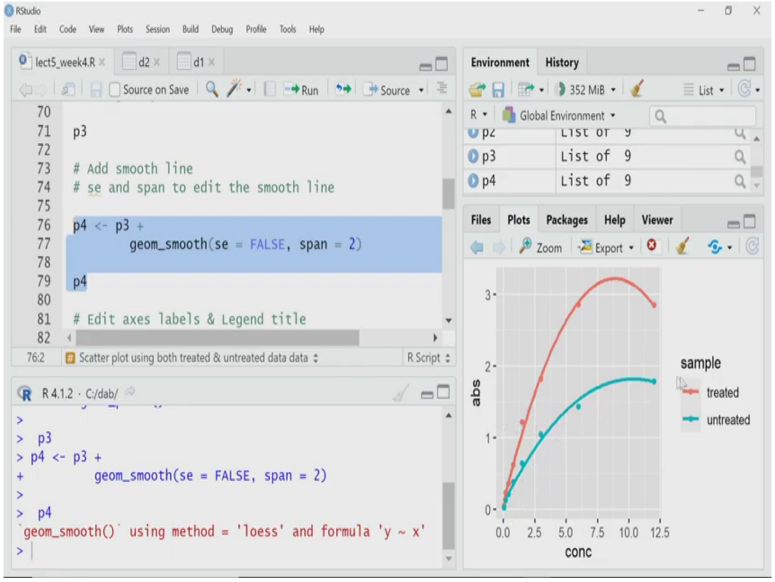
So, if you look into the sample column, I have two factors treated untreated. So, that means, I am telling ggplot, see for treated sample readings use one color and for untreated sample reading use another color, its simple, is not it? Then I am adding by Addition symbol the geometric element I want scatter plot So, I am using geom underscore point and then I am assigning this whole thing T to P3 variable and then I will print P3. Here it goes.

Now I have a scatter plot where both the data is present and treated is red colored field circles, blue colored field circle is for untreated. Now, many a time I want to add a smooth lines to these dataset means I want that, one smooth line should go to these blue dots, another one will go to these red dots. There are many ways you can do that.

What I will do, I will, here in this example, I will add another geometric layer for smooth layer, smooth line and that function is geom underscore smooth. And what I am doing, I have earlier created a P3 variable where the graph is there. So I am taking that P3 and adding this layer just by this addition symbol. Now let me explain what are the arguments here, SE is equal to false, span equal to 2.

SE stands for here. In this case you have some confidence interval that will be calculated by default by geom underscore smooth that I want to not to visualize, so I have set false. And span has very a particular utility. I will explain that., first, let me plot it. So I am creating a new variable P4, where all these things will be added and stored, and then I will print it. Now you can see beautiful smooth line has been added, 1 blue line for the untreated and other one for the treated one.

(Refer Slide Time: 30:21)

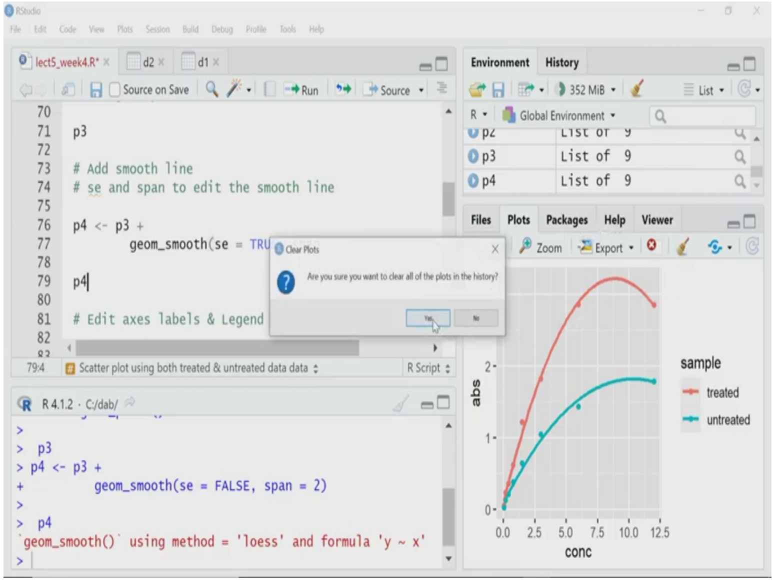


```
p3

# Add smooth line
# se and span to edit the smooth line

p4 <- p3 +
         geom_smooth(se = FALSE, span = 2)

p4

# Edit axes labels & Legend title
```

76:2 Scatter plot using both treated & untreated data data R Script

p2	List of	9
p3	List of	9
p4	List of	9

```
>
> p3
> p4 <- p3 +
+         geom_smooth(se = FALSE, span = 2)
>
> p4
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```



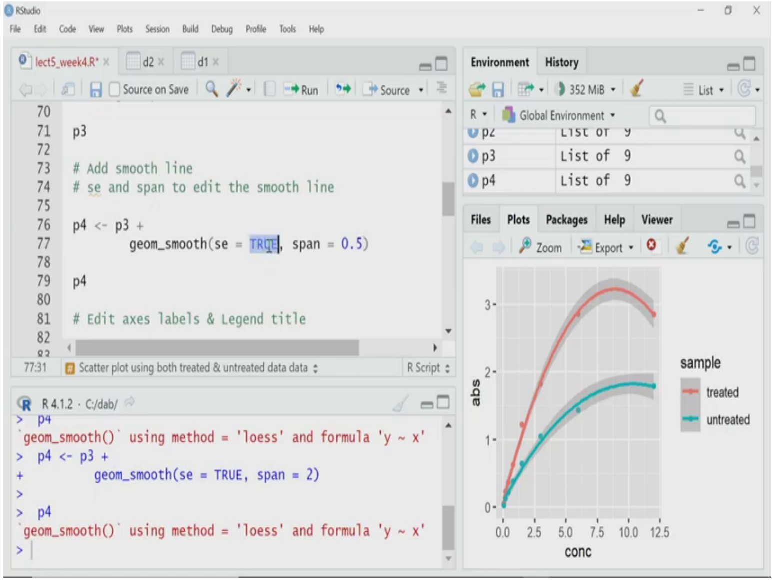
```
p3

# Add smooth line
# se and span to edit the smooth line

p4 <- p3 +
         geom_smooth(se = TRU

p4

# Edit axes labels & Legend
```

Are you sure you want to clear all of the plots in the history?

79:4 Scatter plot using both treated & untreated data data R Script

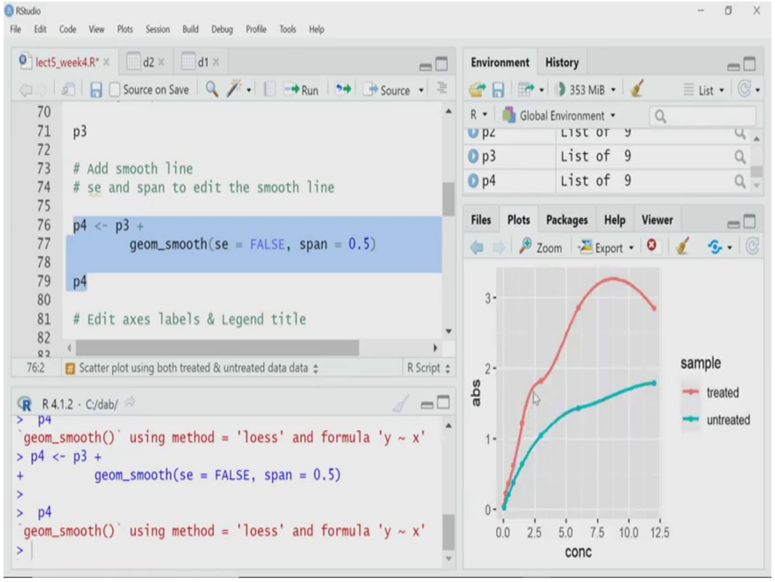
p2	List of	9
p3	List of	9
p4	List of	9

```
>
> p3
> p4 <- p3 +
+         geom_smooth(se = FALSE, span = 2)
>
> p4
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```



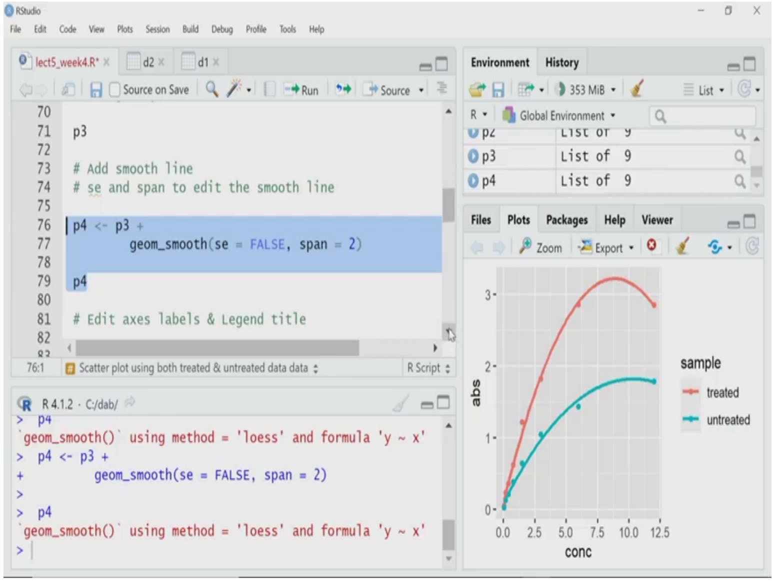
```
p3

# Add smooth line
# se and span to edit the smooth line

p4 <- p3 +
        geom_smooth(se = TRUE, span = 0.5)

p4

# Edit axes labels & Legend title
```

```
> p4
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> p4 <- p3 +
+         geom_smooth(se = TRUE, span = 2)
>
> p4
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```



```
p3

# Add smooth line
# se and span to edit the smooth line

p4 <- p3 +
        geom_smooth(se = FALSE, span = 0.5)

p4

# Edit axes labels & Legend title
```

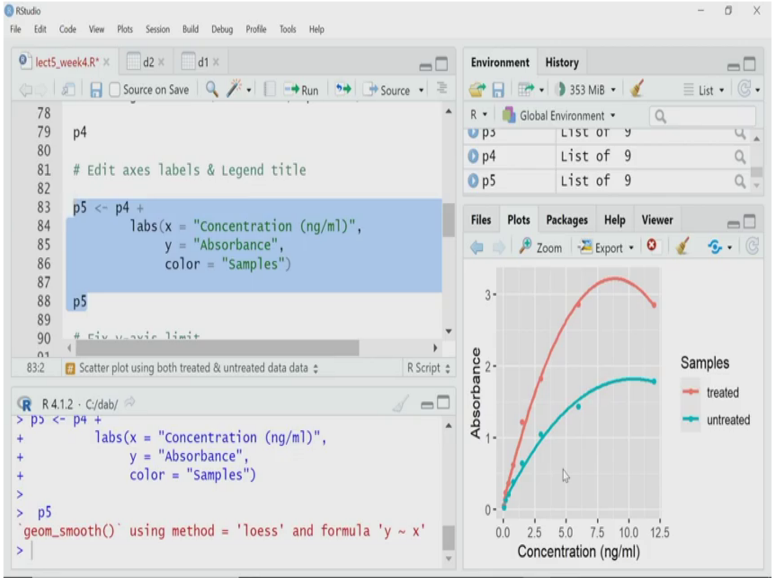
```
> p4
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> p4 <- p3 +
+         geom_smooth(se = FALSE, span = 0.5)
>
> p4
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```



```
p3

# Add smooth line
# se and span to edit the smooth line

p4 <- p3 +
        geom_smooth(se = FALSE, span = 2)

p4

# Edit axes labels & Legend title
```

```
> p4
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> p4 <- p3 +
+         geom_smooth(se = FALSE, span = 2)
>
> p4
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```

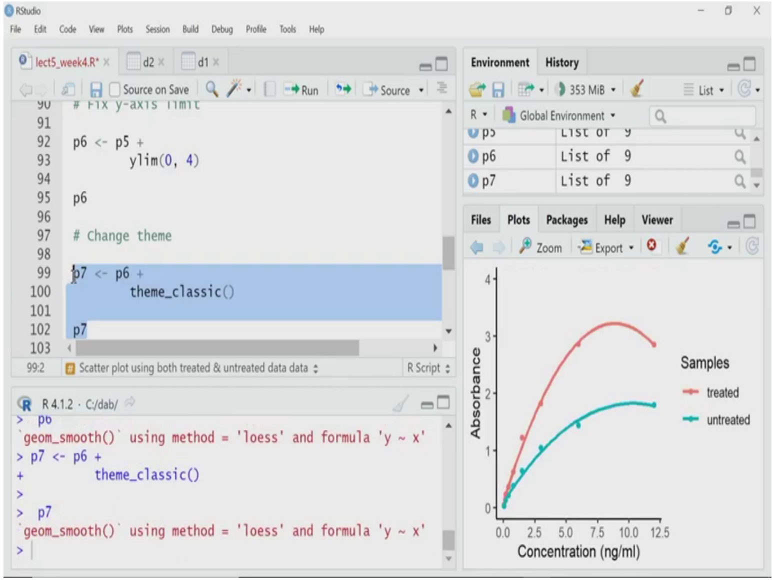
```
p2    List of  9
p3    List of  9
p4    List of  9
```



```
p4

# Edit axes labels & Legend title

p5 <- p4 +
        labs(x = "Concentration (ng/ml)",
             y = "Absorbance",
             color = "Samples")

p5
```

```
> p5 <- p4 +
+         labs(x = "Concentration (ng/ml)",
+              y = "Absorbance",
+              color = "Samples")
>
> p5
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```

```
p3    List of  9
p4    List of  9
p5    List of  9
```



```
# Fix y-axis limit

p6 <- p5 +
        ylim(0, 4)

p6

# Change theme

p7 <- p6 +
        theme_classic()

p7
```

p5	List of 9
p6	List of 9
p7	List of 9

```
> p6
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> p7 <- p6 +
+         theme_classic()
>
> p7
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> |
```

```
p4 ← p3 +
        geom_smooth(se = FALSE, span = 2)
p4


p5 ← p4 +
        labs(x = “Concentration (ng/ml)”,
                y = “Absorbance”,
                Color = “Samples”)
p5


p6 ← p5 +
        ylim(0, 4)
p6


p7 ← p6 +
        theme_classic()
p7
```

Now, if you go to the console, the console is saying that it has used a method called Loess, its a regression method, which is used. And when it is performing Loess, it can actually calculate the confidence interval also. And some time we want to show that confidence interval in the plot itself. So, if you want to show the confidence interval, what you have to do, you have to keep this SE equal to true. Actually, by default this SE is true, but explicitly I am writing here.

Now, if I execute it you can see this gray zone has come, that is the confidence interval calculated by the algorithm which has performed the Loess data smoothing. What is the role of Span 2? Span 2 actually help decide that what should the window that Loess, window of data that the Loess algorithm should use, if you make it smaller, it will have very wiggly data fitting, if you make it bigger, it will the data fitting will be smoother.

Let me give you one example. So if I make span equal to 0.5, and now again, and let me keep SE equal to false, we do not need that. So I reduced span from 2 and I will execute it again. Now you can see we have very wiggly, an undulating data fitting, you may not need that, you do not want that. So I will go back and stick to 2, we got back the same plot.

Now what I will do, I will add some axis labels and legend title something like that, here is fine the axis is labeled absorbance and concentration. But if you want to make it a publication quality image, I have to explicitly write it concentration and the unit also and absorbance have to be written clearly. And maybe here it is written as sample.

It is not right to use a small letter at the very beginning. So I want to make S a capital here. So what I will do? This plot is P4. So this P4 I will take and then I will put the Addition symbol and then I will add some other elements, other information. What are those? This is labels, labs. And then I am saying labs function I am calling and I then am seeing x equal to concentration, nanogram per ml.

So now ggplot2 understand, the title of the x axis should be concentration, nanogram per ml, y equal to absorbance. And the color, color mean the legend because see, you have color coded your data. And that has gone into this legend in the hand side. So the heading of that can be changed by saying writing something for color equal to that. So I have written samples now, S I have made capital letter. So I will execute that and store it in P5, and then I will plot P5.

Now it looks better, is not it? So it looks much more professional. But still, you may have noticed that my line that is smooth line, the rich smooth lines just going above 3, But my axis scaling is still up to 3, is not it? It would be better that if I increase the limit of Y axis, so that is what I will do. Again, I am doing step by step modification.

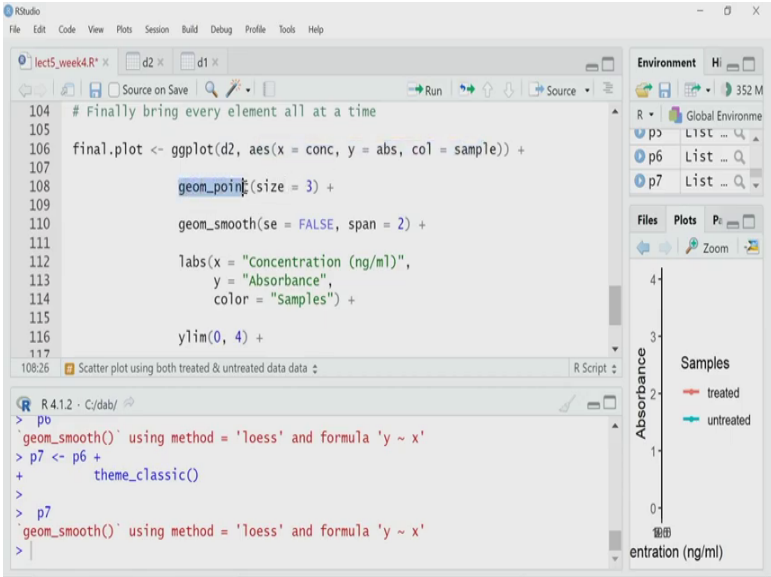
So what I will do, I will take the P5 plot, P5 variable, add y limit, and as a argument I will say, we will start from 0, go up to 4 and make these things P6, 6 plot. And if I execute, that axis is has got corrected. Now I will introduce themes. If you remember, at the very beginning, we said apart from those three basic elements, there are some couple of other elements like statistics, themes, facet.

Theme is all non-data ink. For example, in this case, if you see carefully in that plot, you can see there is a gray background with these grid lines, white color grid line, sometimes these background color is useful. Sometimes these gridlines are helpful to understand the data, but in this particular case, it is not really helping me anyway. And personally for me, it is not aesthetically pleasing. So I want to get rid of this.

So those are non-data thing, non-data ink that is why. So I want to use a theme where these things will not be there. So what I will do, ggplot2 has an inbuilt theme called classic theme. It is a simple plotting theme. So I will use that, to use that what I will do P6 plot I will take simply add that theme underscore classic. So I will create a new variable and plot that variable.

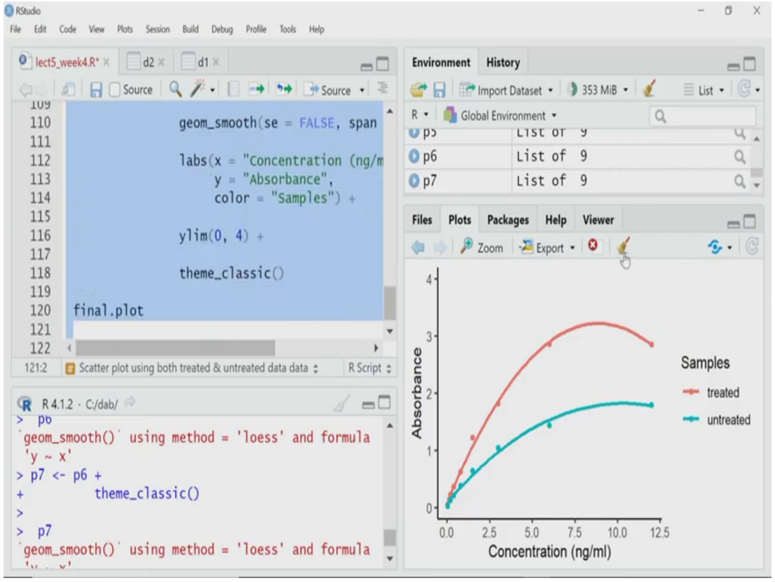
I think personally, this is much better, this plot is much better than the previous one, there is no background, no grayness, no grid line, only thing still sticking to me is that, the size of these dots circles are a bit small. What I will do, I will now put all those elements that I have joined one by one together to create this plot. I will bring them all together to create a final plot. And in that final plot, I will do that editing.

(Refer Slide Time: 35:55)



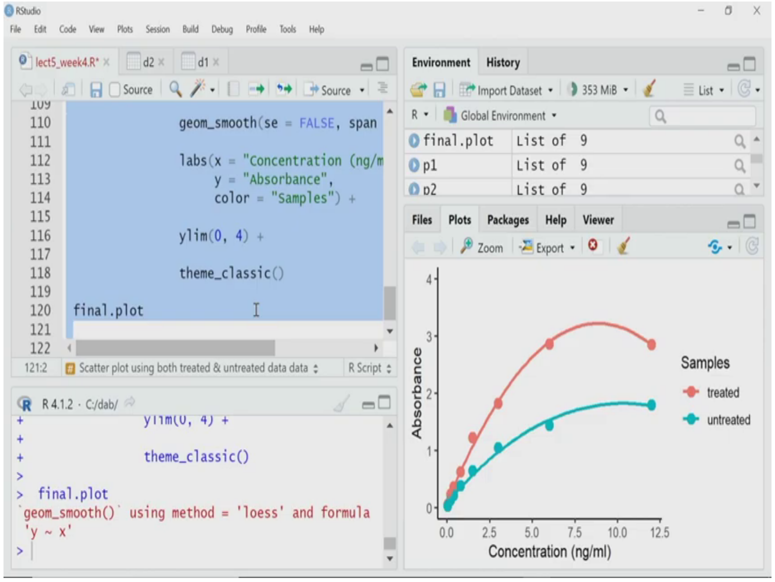
```
# Finally bring every element all at a time

final.plot <- ggplot(d2, aes(x = conc, y = abs, col = sample)) +

              geom_point(size = 3) +

              geom_smooth(se = FALSE, span = 2) +

              labs(x = "Concentration (ng/ml)",
                   y = "Absorbance",
                   color = "Samples") +

              ylim(0, 4) +
```

```
> p6
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> p7 <- p6 +
+         theme_classic()
>
>  p7
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```



```
              geom_smooth(se = FALSE, span

              labs(x = "Concentration (ng/m
                   y = "Absorbance",
                   color = "Samples") +

              ylim(0, 4) +

              theme_classic()

final.plot
```

```
> p6
`geom_smooth()` using method = 'loess' and formula
'y ~ x'
> p7 <- p6 +
+         theme_classic()
>
>  p7
`geom_smooth()` using method = 'loess' and formula
```



```
                    geom_smooth(se = FALSE, span

                    labs(x = "Concentration (ng/m
                         y = "Absorbance",
                         color = "Samples") +

                    ylim(0, 4) +

                    theme_classic()

final.plot
```

```
+                    ylim(0, 4) +
+
+                    theme_classic()
>
>  final.plot
`geom_smooth()` using method = 'loess' and formula
 'y ~ x'
>
```

```
final.plot   List of  9
p1           List of  9
p2           List of  9
```

```
final.plot ← ggplot(d2, aes(x = conc, y = abs, col = sample)) +

                    geom_point(size=3) +

                    geom_smooth(se = FALSE, span = 2) +

                    labs(x = "Concentration (ng/ml)",

                         y = "Absorbance",

                         Color = "Samples") +

                    ylim(0, 4) +

                    theme_classic()

final.plot
```

So what I am doing here, I am creating a final plot, where I have now joined all those things that I was doing step by step from P1 to P6. So, I have first calling ggplot2 ggplot giving d2 data defining the aesthetic, then I am saying, make scatter plot using geom underscore point, I have made an edit here, as an argument to geom underscore point earlier there is not this argument, no argument was there, I am saying size equal to 3, that means the symbol size that will be used of size 3.

Then I am adding geom underscore smooth we have done that, then I am adding labs then I am adding ylim limits and I am using the theme classic I am adding at the end. So all these things now come together in one point because step by step I have worked, I am satisfied with it. Now I have joined them together to create a final code that I can reuse later on also.

So what I will do, I will execute this final plot code and plot it. Before I plot, let me clean this tab so that I can see only that plot. Nice. This is what I will call a publication quality graph. This is almost ready to go in your paper. So that is all for this lecture. What I have discussed here, I have introduced you to the grammar of graphics which is used in ggplot2. As I said, ggplot2 is a R package, which is versatile and very powerful for scalable data visualization.

And although it seems initially that, learning the grammar of graphics may be difficult, but you must have realized here you can actually learn it step by step if you understand what is the purpose of each element. So go back, try the script and manipulate it. Take the help of the cheat sheet of ggplot2, the link I have given and getting yourself ready for our next video where we will discuss about ggplot2 further. Thank you. Happy learning.