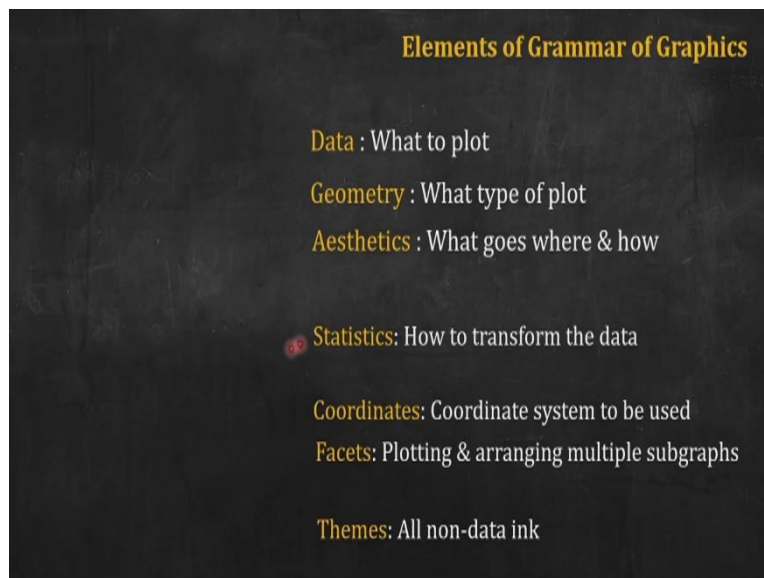


**Data Analysis for Biologists**  
**Professor Biplab Bose**  
**Department of Bioscience & Bioengineering**  
**Mehta Family School of Data Science & Artificial Intelligence**  
**Indian Institute of Technology Guwahati**  
**Data visualization using ggplot2 - II**

Welcome back. In the last lecture, I introduced you to ggplot2, the data visualization package of R, which is based upon gg, the grammar of graphics. While discussing ggplot2 and learning about the grammar of graphics, we discussed that just like a natural language, we can divide a graph, a plot into different elements. And those elements come together in different layers to create a data visualization, a graph or a plot.

And if you remember I said there are three key elements, which you must have in your visualization. One is the data itself. The second thing is the geometry or geom in case of ggplot2, which tells what type of visualization you want, which type of graph you want to plot, and the third one is aesthetic, which tells ggplot function that what to put where.

(Refer Slide Time: 1:43)



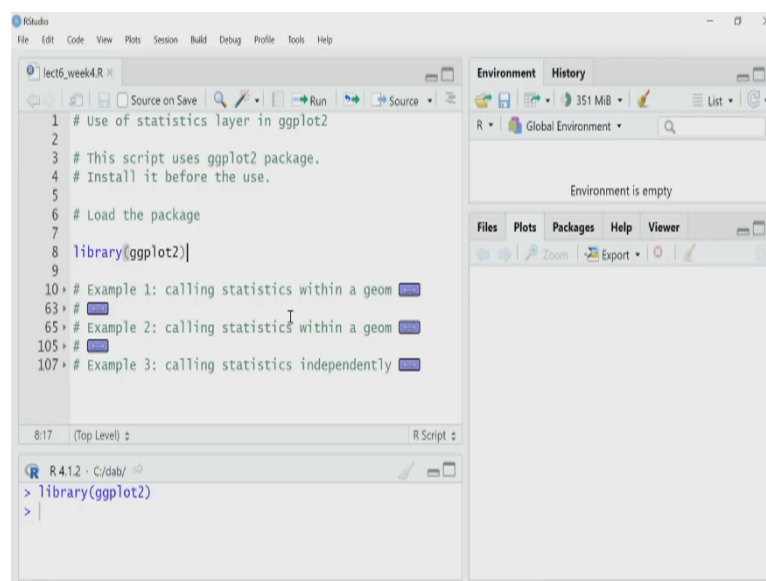
Now, apart from that, I said that there are some other elements also. For example, statistics, coordinate, facets and themes. Now, if you remember, I said the statistics is used when I want to add some layer some information, where the data statistics is used to create the plot. When you simply create a plot using the raw data, you do not need statistics.

But sometimes you may not plot the raw data rather extract some statistics like mean median, some quartile value or some other thing and plot that, then raw data will not be useful, you

have to use the statistics element to create that data and then visualize it. And then we have coordinates where you may want to change the coordinate, and you may want to make multiple sub plot, you may need the facet, and theme is for all non-data ink.

Now, in the last lecture, we discussed a detail about how to use data, geometry and aesthetic and stitch them together to create a plot. And today, I will discuss about the fourth most important thing, the statistics.

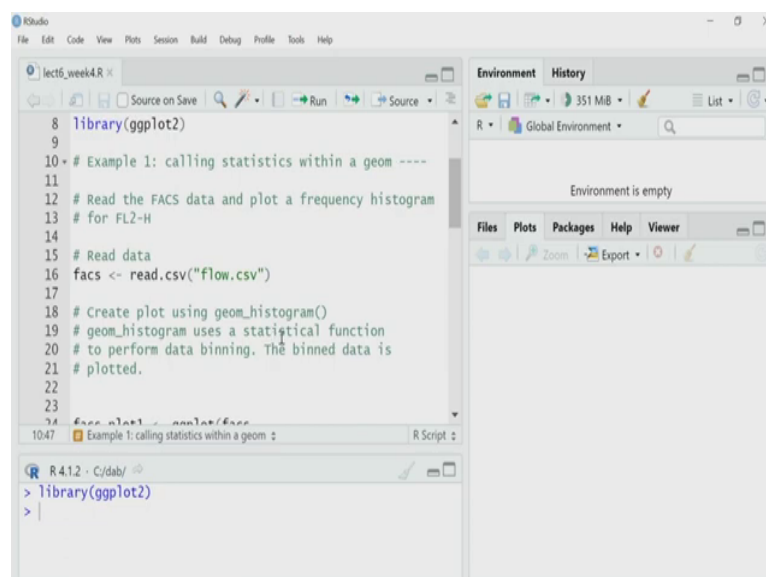
(Refer Slide Time: 2:53)



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

```
1 # Use of statistics layer in ggplot2
2
3 # This script uses ggplot2 package.
4 # Install it before the use.
5
6 # Load the package
7
8 library(ggplot2)
9
10 # Example 1: calling statistics within a geom
11 #
12 # Example 2: calling statistics within a geom
13 #
14 # Example 3: calling statistics independently
```

The console shows the command `library(ggplot2)` being executed.



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

```
8 library(ggplot2)
9
10 # Example 1: calling statistics within a geom ----
11
12 # Read the FACS data and plot a frequency histogram
13 # for FL2-H
14
15 # Read data
16 facts <- read.csv("flow.csv")
17
18 # Create plot using geom_histogram()
19 # geom_histogram uses a statistical function
20 # to perform data binning. The binned data is
21 # plotted.
22
23
24 # Example 1: calling statistics within a geom
```

The console shows the command `library(ggplot2)` being executed.

```
library(ggplot2)
```

```
facts <- read.csv("flow.csv")
```

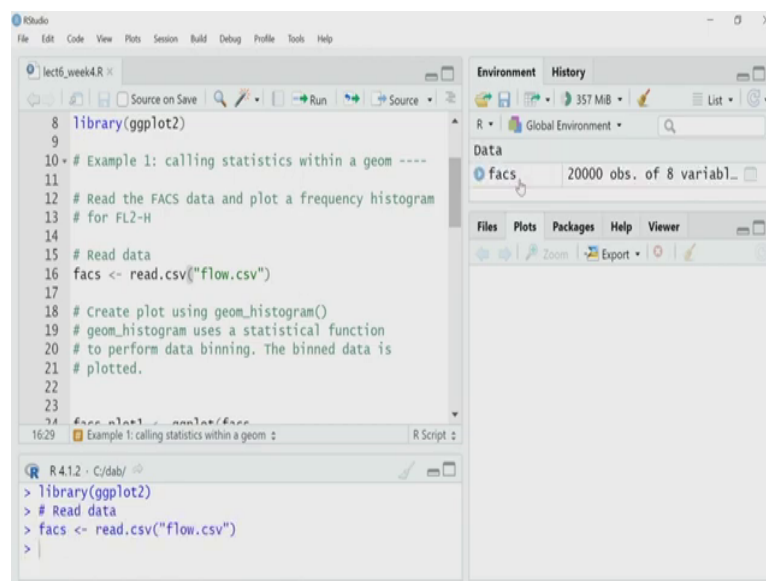
So, let us get into R code and learn about it. So, I will use `ggplot2`, I have already installed that in my machine. Now, let me load that using the `library` function. Now, remember the statistics functions, which are the part of these statistical layers in my graphics may come from multiple way.

One is that there are a suppose, sometimes you are using a particular geometry, that geometry function itself will call some statistic function, and will calculate the statistics even without telling you. So, it will do perform the statistical analysis using some statistical function under the hood. So, it is an indirect way of calling the statistical functions.

On the other hand, you can directly call a statistical function and then actually use that to create a plot. So, I will give both the example in this script. So, first what I will do, first example that I will deal with is where I am calling a `geom` function, a geometry function, a geometry element, and that in itself will call a statistical function, and then we'll create the plot.

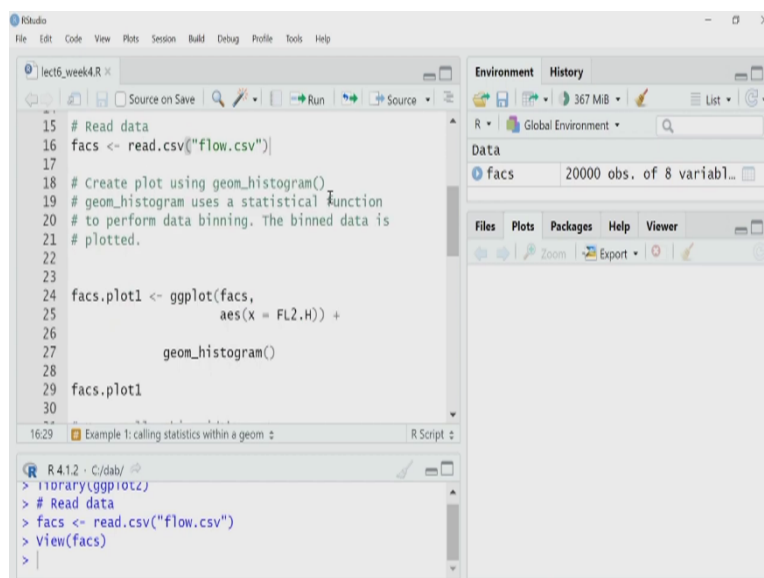
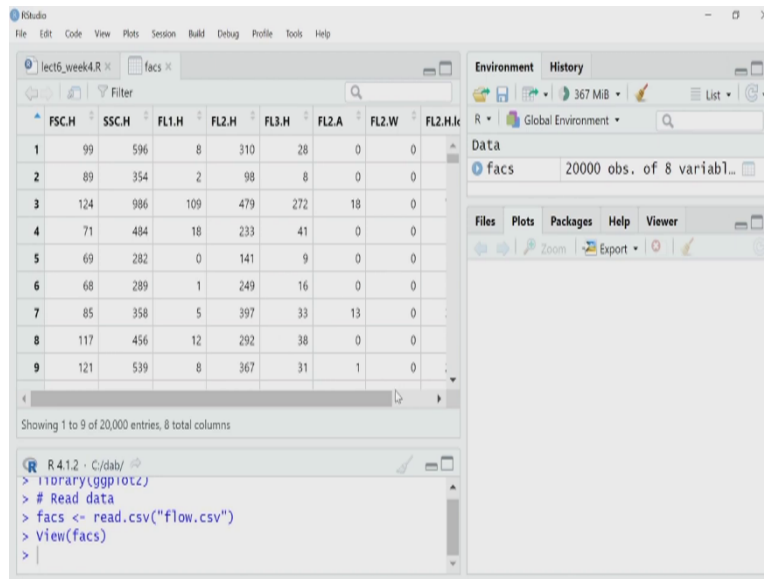
So, the example is involving drawing histogram using a flow cytometry data, we have plotted earlier. So, what you have to do? You have to bring the data in different bins and then you take the bin counts, the counts of data in each of these bin and plot them as sequential bar plots, and you get a histogram.

Refer Slide Time: 4:31)



```
8 library(ggplot2)
9
10 # Example 1: calling statistics within a geom ----
11
12 # Read the FACS data and plot a frequency histogram
13 # for FL2-H
14
15 # Read data
16 facs <- read.csv("flow.csv")
17
18 # Create plot using geom_histogram()
19 # geom_histogram uses a statistical function
20 # to perform data binning. The binned data is
21 # plotted.
22
23
24 facs$plot1 <- geom_bar(facs
```

The screenshot shows the RStudio interface. The main editor window contains R code for loading the `ggplot2` library and reading a CSV file named `flow.csv` into a data frame called `facs`. The code includes comments explaining the steps: loading the library, reading the data, and creating a plot using `geom_histogram()`. The Environment pane on the right shows the `facs` data frame with 20000 observations and 8 variables. The Console at the bottom shows the execution of the code, including the `library(ggplot2)` command and the `read.csv` function call.



```
facs.plot1 ← ggplot(facs,
                    aes(x = FL2.H)) +
                    geom_histogram()
```

facs.plot1

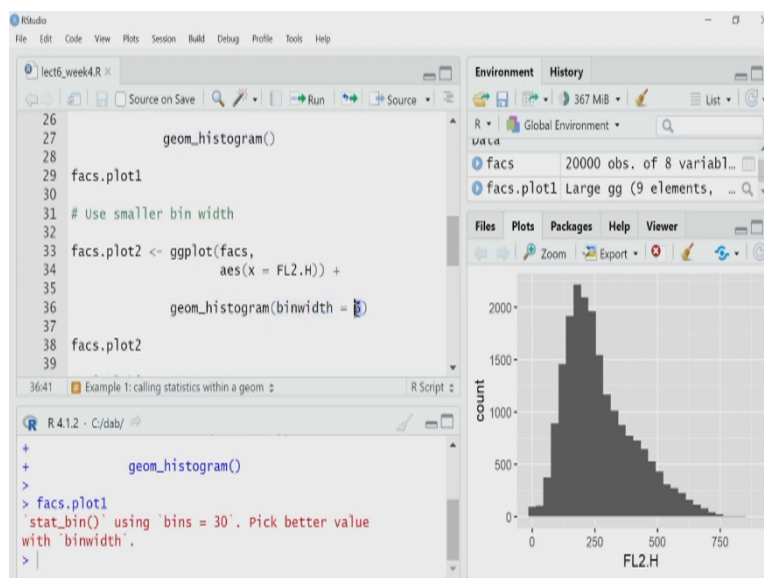
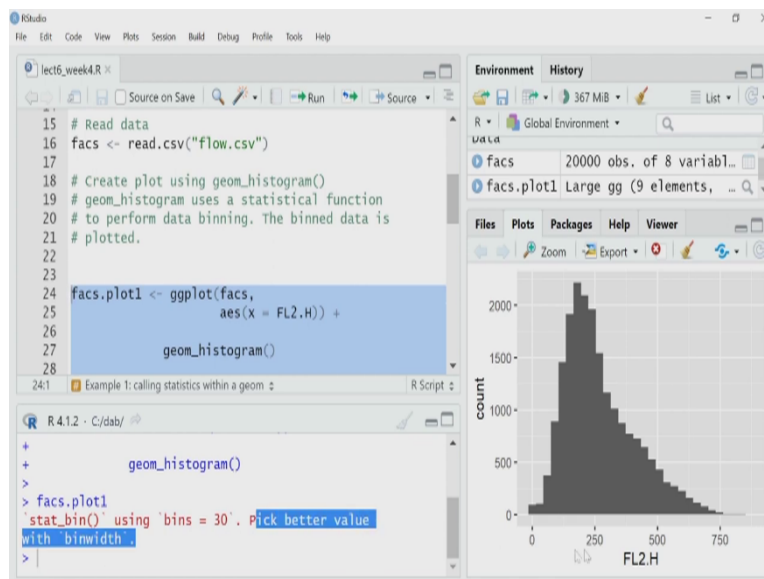
So, let me read that data, its the flow cytometry data. I am reading it as a way to read dot CSV function. So here the data Facs variable as you can see, it has multiple variables in multiple columns, and suppose I want to plot the histogram for FL2h. So, to do that, what I will do? I will use a function called Geom underscore histogram.

So, this is a geometry function. And remember to create histogram it has to bin the data. So, now this function will call a statistical function, which has been written to create the binning.

Without even let me know that it is doing that. So, let me execute it. So, the first thing would be ggplot, I have to call the ggplot function if you remember.

And then I am giving the first most important element, the data, the facts variable and then I am declaring the aesthetic element, I am saying in the horizontal axis, the x axis, you put a FL2 H variable, and then I am putting the addition mark, and then I am calling this geom underscore histogram function. So, the geometry is histogram. And I am putting all these and assigning all these information to a variable called facts dot plot 1, and I will plot that.

(Refer Slide Time: 5:39)



```
facts.plot2 ← ggplot(facts,
                     aes(x = FL2.H)) +
```

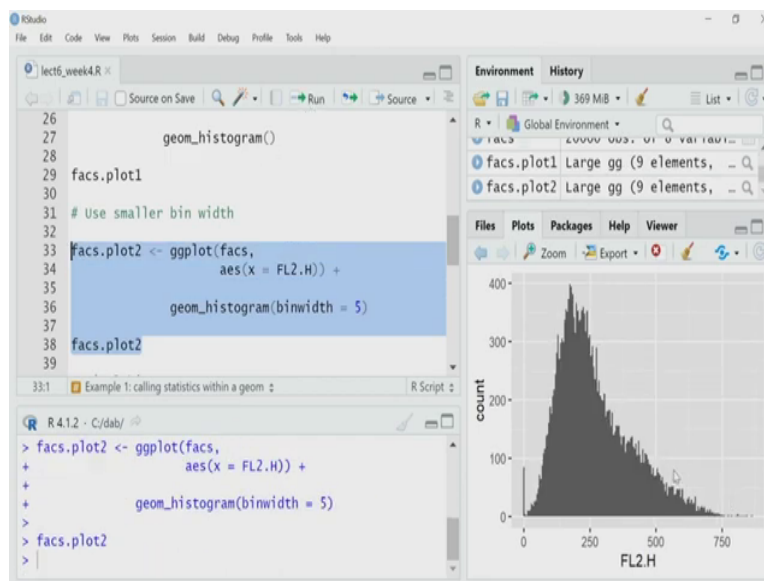
`geom_histogram(binwidth = 5)`

`facs.plot2`

Now, you can see I have got a histogram and also in the console, if you notice, what it has said is that the bins equal to 30 that means the bin size has considered as 30, it has its own algorithm, that statistical function used by geom underscore histogram has its own algorithm to calculate the bin size. So, it has calculated but it is also saying telling me clearly that pick a better value of bin width. As you can see the bin width are here a bit bigger.

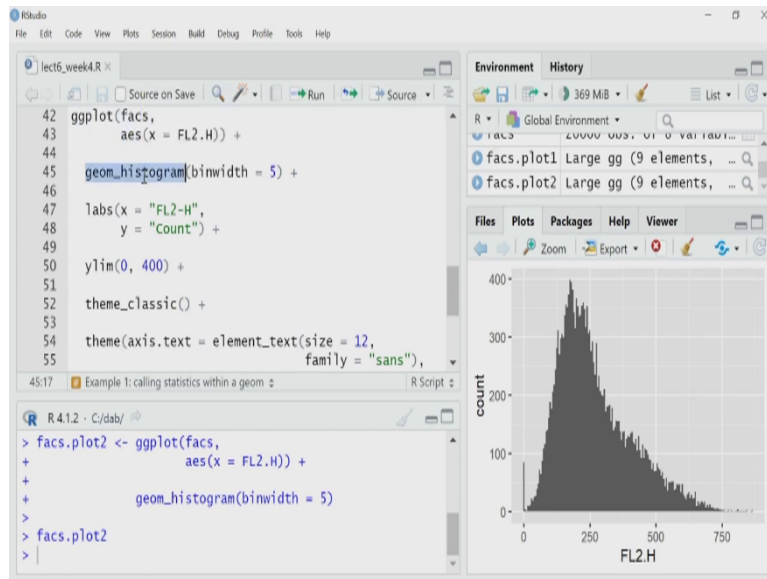
So, I will rather prefer to use a smaller bin width, how can I do that change? Well, I will keep everything same, I will call ggplot function, give the data and then declare the aesthetic element and then when I am calling this geometry element, the geom underscore histogram, I will define the bin width as an argument. So, I am writing bin width equal to 5, and then I am assigning the whole thing in the second plot, and I will plot that.

(Refer Slide Time: 6:47)



Now, you see this diagram is much better, the histogram is much smoother. And I think from my experience that this bin width is good enough for this particular plot.

(Refer Slide Time: 6:58)



```

ggplot(facs,
       aes(x = FL2.H)) +
       geom_histogram(binwidth = 5) +
       labs(x = "FL2-H",
            y = "Count") +
       ylim(0, 400) +
       theme_classic() +
       theme(axis.text = element_text(size=12,
                                       family = "sans"),
            axis.title= element_text(size=12,
                                       family = "sans"))

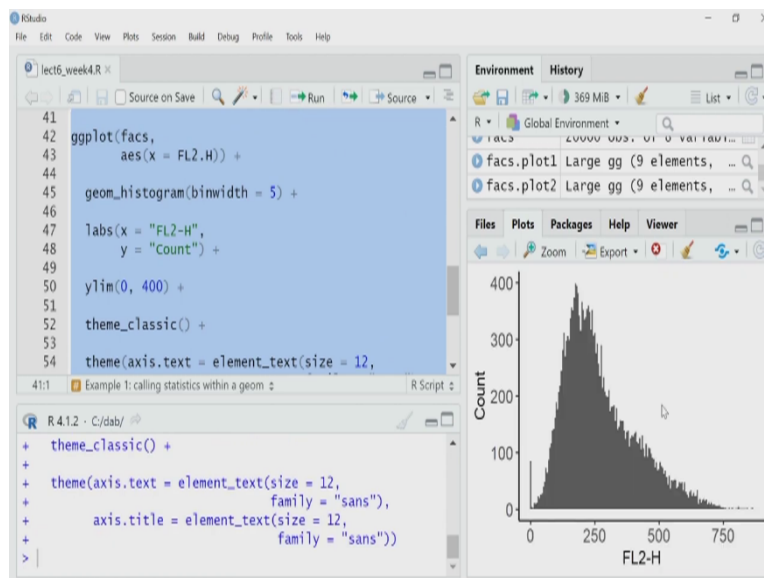
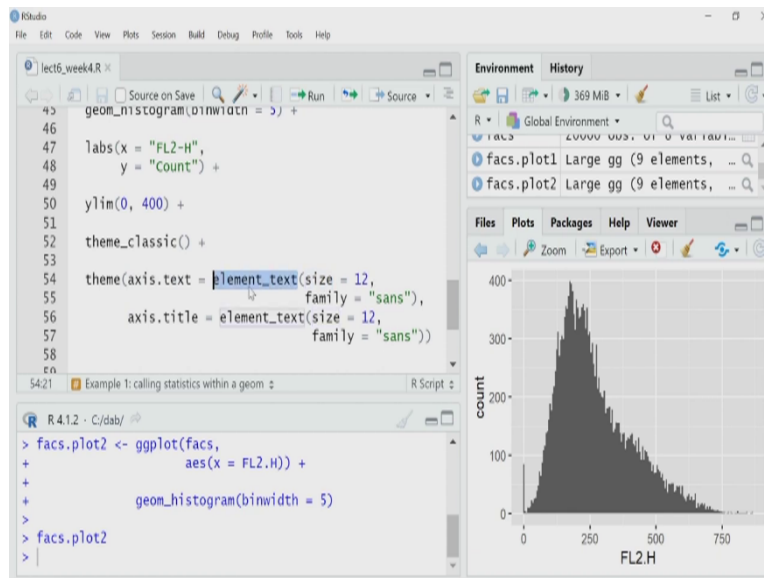
```

So, now, what I will do, I will create the final histogram for my presentation and I will do some cosmetic changes, I will change some text size and format, then labels of the axis. So, what I am doing here? I am keeping the initial things like ggplot function is called, data is given, aesthetic is declared then the histogram geometry is called and with the bin width equal to 5 and then I am adding extra thing for the makeover of this graph.

So, I am writing labs the labels, x axis labels should be FL2 h, and y axis labels should be count C is in capital now that should be. And I am saying y limit is 0 to 400, then I am calling a theme, remember theme involves all non-data inks, like the background and all these things,

I do not like this grey background all with gridlines, so I will get rid of that. So, I will use the inbuilt theme underscore classic, which gives me a very neat and minimal representation of the diagram.

(Refer Slide Time: 8:03)



```
ggplot(facs,  
  aes(x = FL2.H)) +  
  geom_histogram(binwidth = 5) +  
  labs(x = "FL2-H",  
       y = "Count") +
```



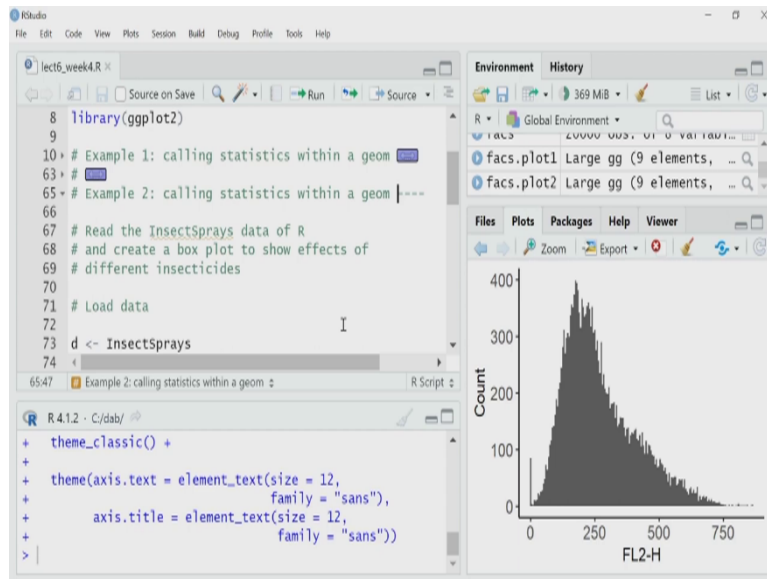
```
ylim(0, 400) +  
theme_classic() +  
theme(axis.text = element_text(size=12,  
family = "sans"),  
axis.title= element_text(size=12,  
family = "sans"))
```

And then I am adding some extra theme elements, what are those? I want to define the font size and the particular type of font that I want to use for this labelling of the axis and the tick marks. So, what I am saying? I am saying, calling the theme function and as argument I am saying axis dot takes that means the text of the axis should be size equal to 12 for that I am using a function element underscore text, and the family should be sans.

Similarly for this axis title, this axis label FL2 H and count you use size equal to 12 and family equal to sans. So, these things will change the way we are using the different fonts and their font sizes. So, let me execute this. Now, I have a decent diagram the way you are habituated to see in paper and reports and PowerPoint presentation and this is ready to go in my presentation.

So, what I have shown in this example? In this example, I have used statistical function, I have incorporated the statistical element, I have not plotted the raw data, rather the bin count data, which is actually statistics, using a geometry element, geom underscore histogram and that function itself has called a statistical function to perform the job.

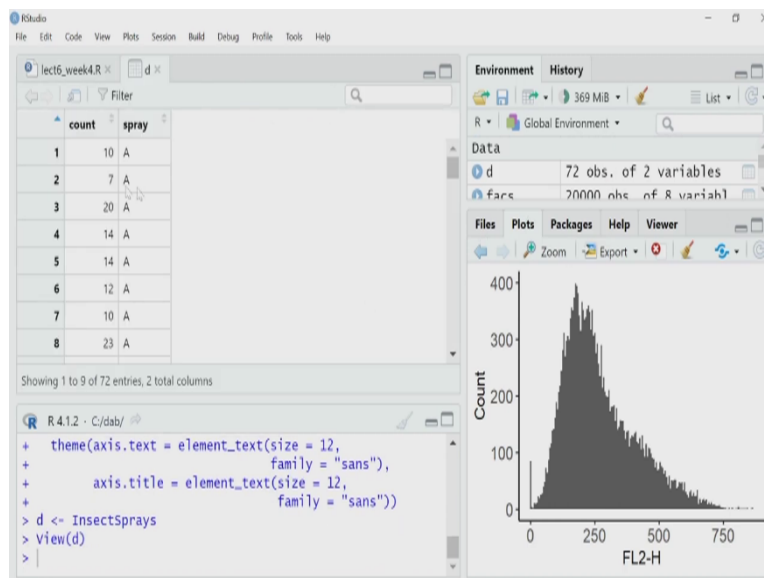
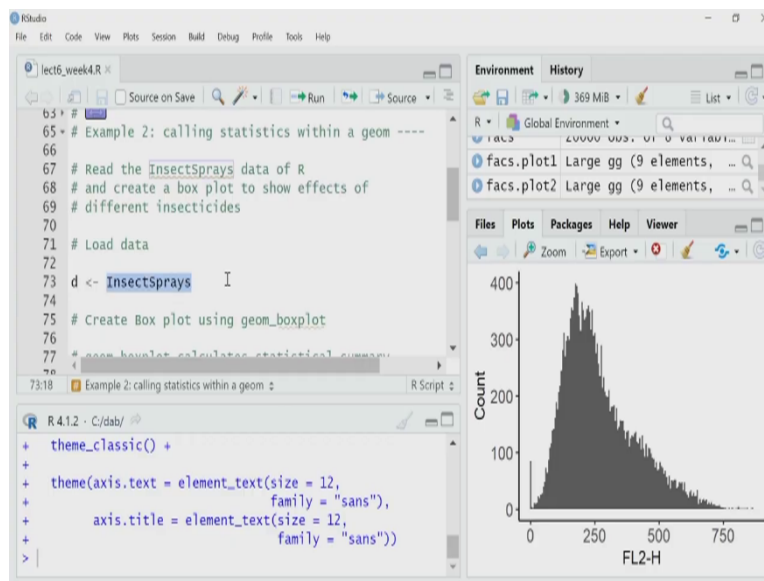
(Refer Slide Time: 9:37)



`library(ggplot2)`

I will give another similar example. So, in the example two what I am doing, again, I am calling this statistical function inside a geometric function and this one involves boxplot, you have learned about boxplot in another lecture. So, in that case, we use the default box plot function in R, here I will use the ggplot to create the same box plot.

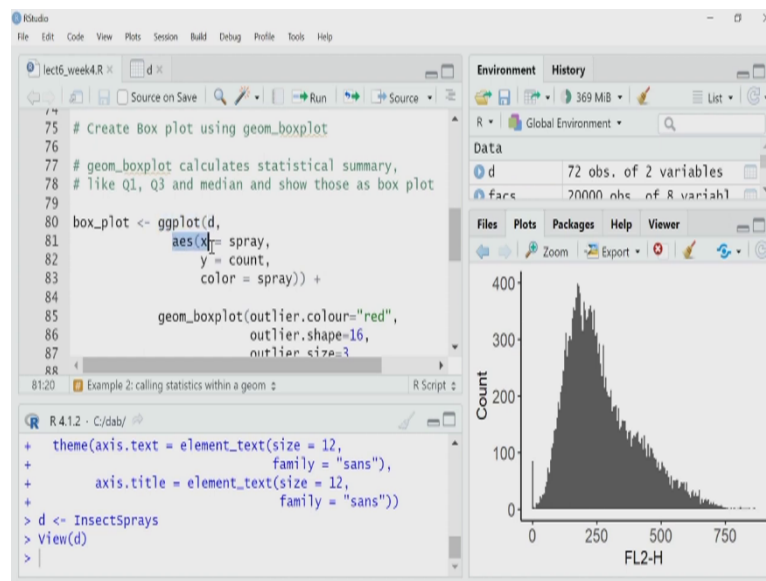
(Refer Slide Time: 9:59)



`d <- InsectSprays`

So, what I am reading, using the data, I am using the insect spray data, the default data set present along with R. So, I will read that and store it in D. So, if you will open D, you can see it is a two-variable thing, the count and the spray, I have multiple types of spray A to F, A, B, C, D, E, F, these are categories, groups, so I have different groups of spray. And each of this spray were applied on the insect, these are insecticide spray, and we have made some counts and that count numerical values are in the count variable.

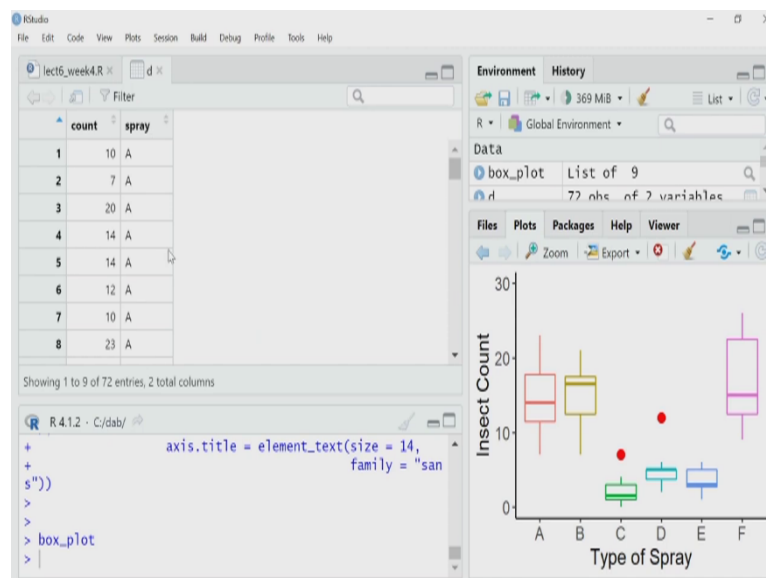
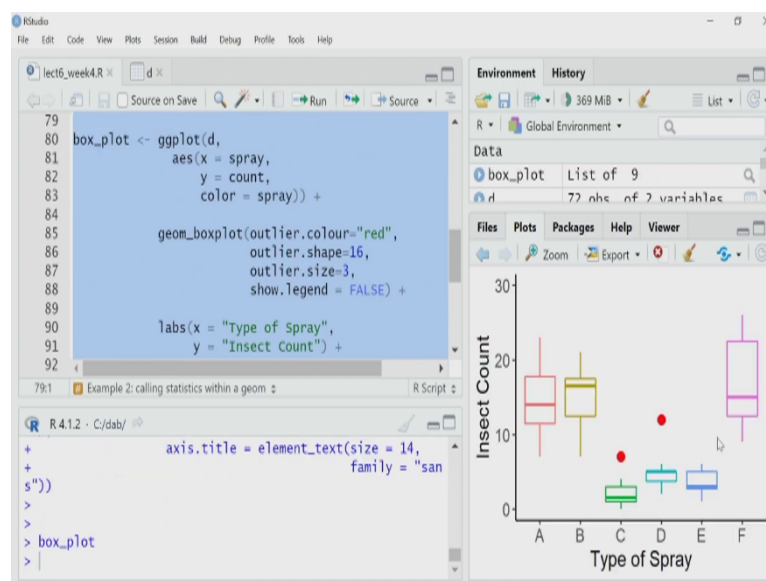
(Refer Slide Time: 10:35)



```
box_plot <- ggplot(d,  
  aes(x = spray,  
      y = count,  
      color = spray)) +  
  geom_boxplot(outlier.color = "red",  
              outlier.shape = 16,  
              outlier.size = 3,  
              show.legend = False) +  
  labs(x = "Type of Spray",  
       y = "Insect Count") +  
  ylim(0, 30) +  
  theme_classic() +  
  theme(axis.text = element_text(size=12,  
                                  family = "sans"),  
        axis.title = element_text(size=14,  
                                   family = "sans"))
```

Now I want to create a boxplot, to do that, I have to use a geometry of boxplot fine. Now, what I will do, I will call ggplot function, give it the data d, and then I will give the aesthetic, I will come and discuss what are there in the argument for the aesthetic function and then I am calling the geometry element, here it is geom underscore boxplot. I have lots of arguments here, let me first make the plot and I will then explain each of these arguments one by one.

(Refer Slide Time: 11:20)



```
box_plot <- ggplot(d,  
  aes(x = spray,  
      y = count,
```

```

        color = spray)) +
geom_boxplot(outlier.color = "red",
             outlier.shape = 16,
             outlier.size = 3,
             show.legend = False) +
labs(x = "Type of Spray",
     y = "Insect Count") +
ylim(0, 30) +
theme_classic() +
theme(axis.text = element_text(size=12,
                                family = "sans"),
      axis.title= element_text(size=14,
                                family = "sans"))

```

Now, I have a very nice box plot here. So, let me explain each of the argument that I have used. Geom underscore box plot is the geometry element, I am adding after ggplot function using the plus sign, here as a argument I have said make the outlier red colored. So, I have written outlier dot color equal to red, so, the outliers are now red colored. Outlier shape I have written as 16, that means it will be a solid circle.

Outlier size is 3, so, I have written outlier dot size equal to 3. If you want bigger you can use 4 or 5, if you smaller if you want you can make it 2 or 1 something like that. And then I have written show dot legend equal to false, why I have written that, let me explain that. To understand that I have to go to the aesthetic element, in the aesthetic element I have written on the horizontal axis in the x axis you put spray.

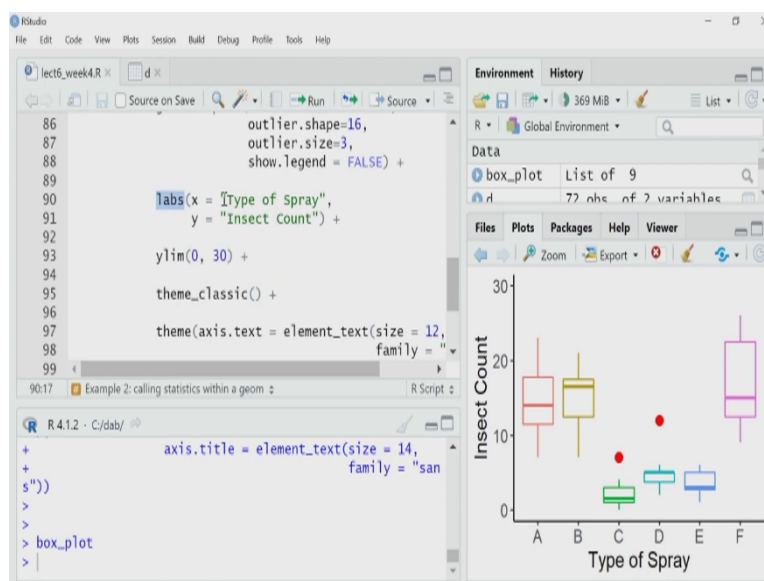
So the spray variable with these six category A, B, C, D, E, F is in the horizontal axis, x axis and the count information is in the y axis. So, the numerical value counts are in the y axis and then I have said in the aesthetic as a argument color equal to spray. Remember here by saying color I am asking to categorize the data and give them a color.

So, how they will categorize it? They will categorize based upon the information present in the spray variable, what do I have in the spray variable? In the spray variable, I have the

labels for each of these group's A, B, C, D, E, F like that. So, based on these labels it will color code the plot, so that is what it has done. So, it has color coded each of these box plot in different color based on the categories, in the spray column, the spray variable.

Now, once you say you have to color code or categorize the data, immediately ggplot2 will also plot a legend along with this figure explaining the meaning of this color code. In some cases that is essential, but in this particular case, I do not need a separate legend to explain which color represent which data set I can easily see this pink one represents the F data set. So, it is just about the F, so I do not need it to be said separately in the legend, that is why I have written show dot legend equal to false.

(Refer Slide Time: 14:00)



```

box_plot ← ggplot(d,
  aes(x = spray,
      y = count,
      color = spray)) +
  geom_boxplot(outlier.color = "red",
    outlier.shape = 16,
    outlier.size = 3,
    show.legend = False) +
  labs(x = "Type of Spray",
    y = "Insect Count") +
  ylim(0, 30) +
  theme_classic() +
  theme(axis.text = element_text(size=12,
    family = "sans"),
    axis.title= element_text(size=14,
    family = "sans"))

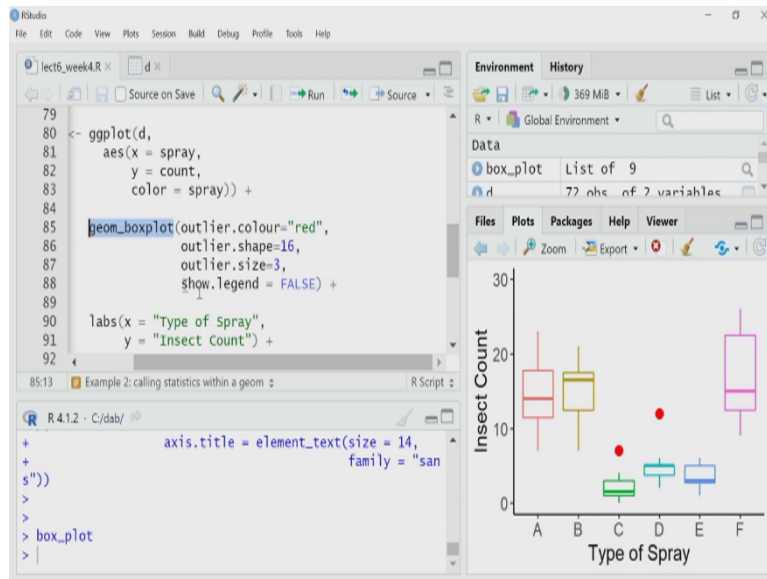
```

Now, apart from that I have added some extra layer for cosmetic changes. For example, I have written label equal to x axis label is type of spray, that is why it is written as type of spray here, Vertical axis I have written as insect count, that is why I have written here in the label y equal to inset count, obviously in apostrophe.

Then I have defined the limit of the vertical axis, I have said y lim is from 0 to 30, then I have added my favorite classic theme of ggplot2, so, theme underscore classic comes here. Then I have added another thin layer where I am saying that use font size 12 and 14 for the text and the title and for both the cases use the sans size font. So that is why these font sizes are bigger and they look neat and clear. So, that is how I have generated these nice-looking box plot.

(Refer Slide Time: 14:59)





```

box_plot ← ggplot(d,
  aes(x = spray,
    y = count,
    color = spray)) +
  geom_boxplot(outlier.colour = "red",
    outlier.shape = 16,
    outlier.size = 3,
    show.legend = False) +
  labs(x = "Type of Spray",
    y = "Insect Count") +
  ylim(0, 30) +
  theme_classic() +
  theme(axis.text = element_text(size=12,
    family = "sans"),
    axis.title= element_text(size=14,
    family = "sans"))

```

So, just to remind you again, while doing the box plot here, I have used the geometry element `geom_boxplot`. And if you remember, in box plot, you are not

plotting the raw data. Take the example of this, F data set, the box has a middle line that is the median. So, this function `geom_underscore_boxplot` has called a statistical function to calculate the median of the data and then only plotted these median, the lower end of the box represent quartile one.

So, again `geom_underscore_boxplot` must have called a statistical function to call calculate the quartile 1 and also the quartile 3, because that is the upper edge of the box. And also, if you remember the whiskers are representing the range of 1.5 of IQR from  $q_3$  and  $q_1$ . So, those values have also been calculated by a statistical function, then only this plot has been created. So, this is another example where the geometry function calls the statistical function and do the work under the hood and create the plot.

(Refer Slide Time: 16:18)

```
65 # Example 2: calling statistics within a geom
105 #
107 # Example 3: calling statistics independently ----
108
109 # Create a bar plot showing variation in Petal
110 # length of different flowers
111
112 # Read data
113 bar.data <- iris
114
115 # Create the bar plot showing mean values
116 # Use stat_summary to get statistics,
117 # Take mean and sd as...
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

`bar.data <- iris`

Now, I will go into the third example, where I will call a statistical element, a statistical function directly and then create the plot. So, what I will do in this example? I will take the inbuilt Iris data set and then I will create bar plot showing the variation in petal length in different species of the iris flowers. If you remember, this data set has three different species and we have different data like petal width, petal length, all these things.

So, I want to make a bar plot showing the petal length of different species of these flowers. So, the first thing I will do, I will read the Iris data and store it in a variable bar dot data. Let

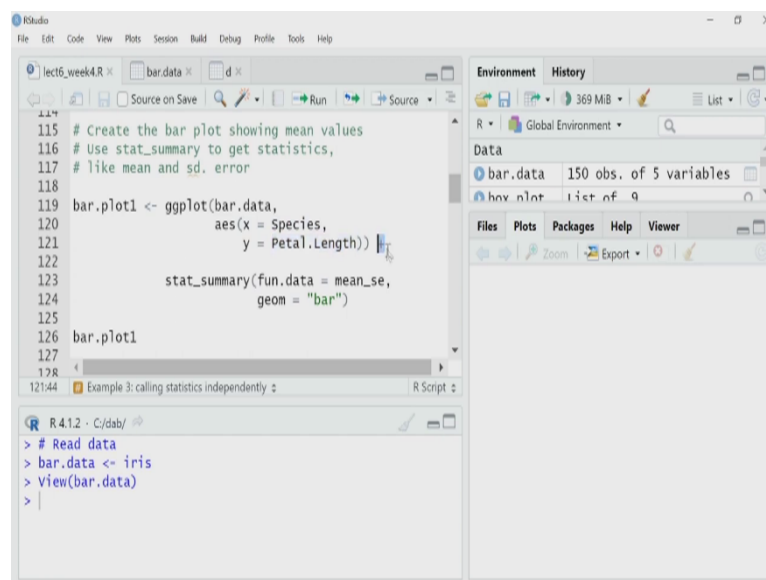
us go and check that check the data once. So, what I have? I have 5 variables sepal length, sepal width, petal length, petal width and the species.

So, these first four variables are numerical data, but the species is category data label data labels, so, I have three species setosa then you go down to versicolor and then you have the data for virginica. So, now I want to create a bar plot showing the difference in petal length of these three species. So, if you have to do that, what you will do?

For each of these species, you will take the data for petal length and calculate the mean of that, because you have multiple data for each of these pieces. For versicolor you can see you have multiple data. So, you will take those values and calculate the mean, and you will draw a bar plot where the bar height will represent the mean value and then you want to add the error bars. So, the error bar can be standard error or standard deviation.

So, that means in this case, you cannot plot these raw data directly, you have to do statistical transformation, you have to calculate mean and also the error. And in this case, I will not do the statistics below the hood, I will call a statistical function directly and I will use that to create the plot.

(Refer Slide Time: 18:11)



The screenshot shows the RStudio interface. The main editor window contains the following R code:

```
115 # Create the bar plot showing mean values
116 # Use stat_summary to get statistics,
117 # like mean and sd. error
118
119 bar.plot1 <- ggplot(bar.data,
120                     aes(x = Species,
121                         y = Petal.Length))
122
123     stat_summary(fun.data = mean_se,
124                 geom = "bar")
125
126 bar.plot1
```

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

Object	Details
bar.data	150 obs. of 5 variables
bar.plot1	list of 9

The console window at the bottom shows the following commands:

```
> # Read data
> bar.data <- iris
> View(bar.data)
>
```

```
bar.plot1 <- ggplot(bar.data,
                    aes(x = Species,
                        y = Petal.Length)) +
  stat_summary(fun.data = mean_se,
```

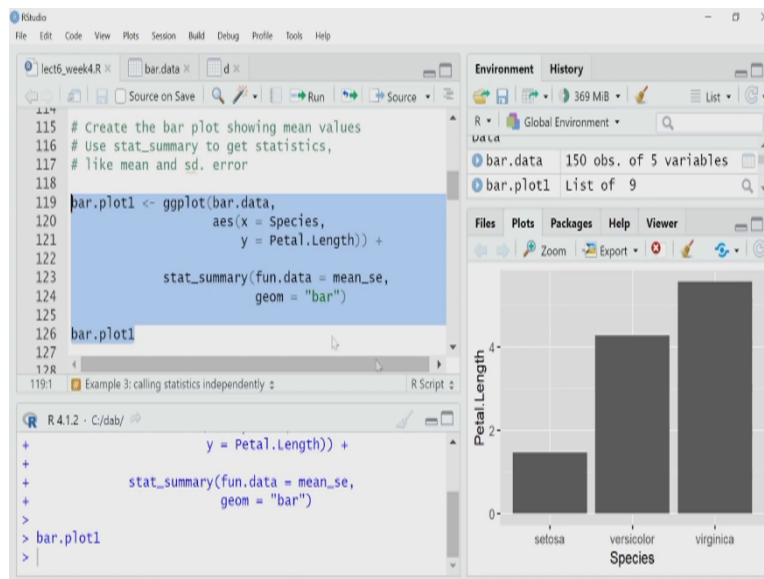
```
geom = "bar")
```

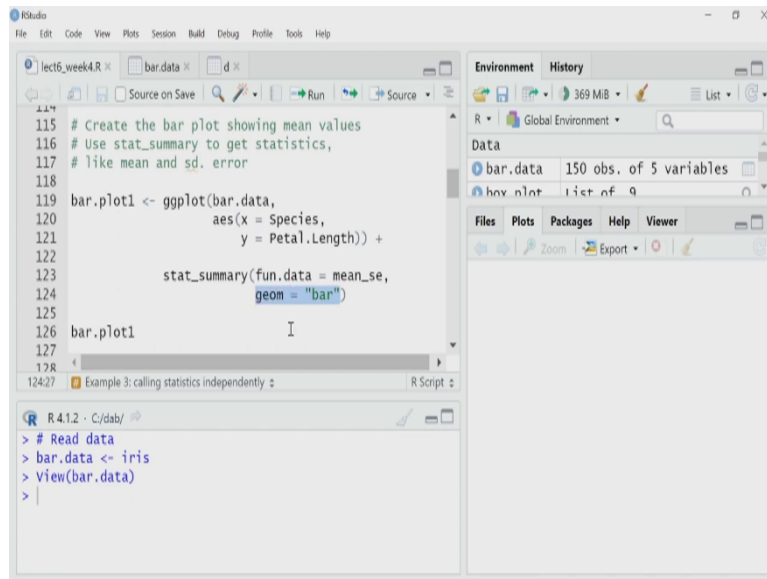
## bar.plot1

So, what I am doing here? Here is my first script, I am calling `ggplot` function, I am giving the data the most important element in my plot, then I am giving the aesthetic element I am saying the species will go in the x axis, horizontal axis, and the petal length will go in the y axis. And then I am putting the addition symbol and then I am calling not a geometry element, I am calling a statistic. I am calling a function called `stat_summary`.

There are many of such statistical function I am using this one, `stat_summary` and it will calculate some summary statistics. What do I wanted to calculate? I wanted to calculate mean and standard error. So, that is why it is written `fun.data = mean_se` this argument `fun.data` is the argument for `stat_summary`, and I am written that that is equal to `mean_se`. So, if I write it this way, then `stat_summary` function will understand that I wanted to calculate the mean and standard error of this data set.

(Refer Slide Time: 19:20)



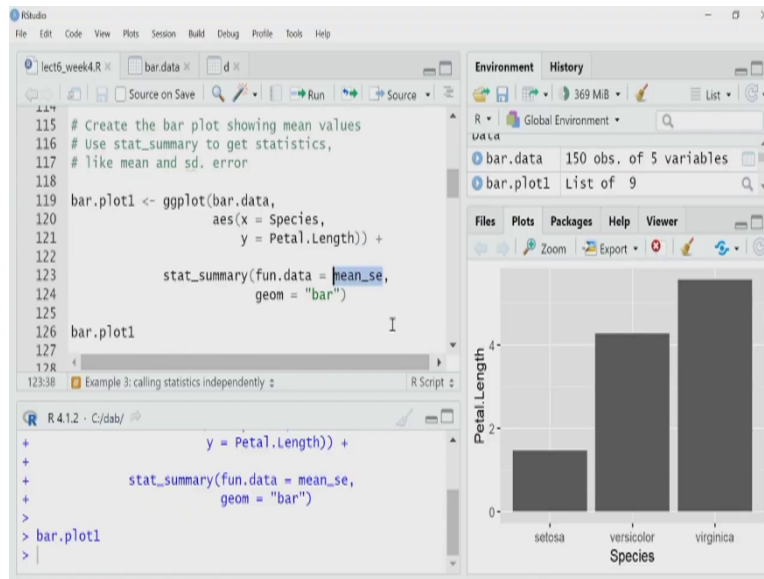


```
bar.plot1 ← ggplot(bar.data,  
                    aes(x = Species,  
                        y = Petal.Length)) +  
                    stat_summary(fun.data = mean_se,  
                                geom = "bar")
```

bar.plot1

And then, I am calling the geometry inside this function, not outside, inside the function. I am saying geom equal to bar. So, it will create a bar plot using the mean calculated by this stat summary function. And then I will plot that, so here I do. Now, you can see I have the bar plot, height of each bar represent the mean of the data for each category, each species of the flower.

(Refer Slide Time: 19:52)



```

bar.plot1 ← ggplot(bar.data,
                    aes(x = Species,
                        y = Petal.Length)) +
                    stat_summary(fun.data = mean_se,
                                geom = "bar")
bar.plot1

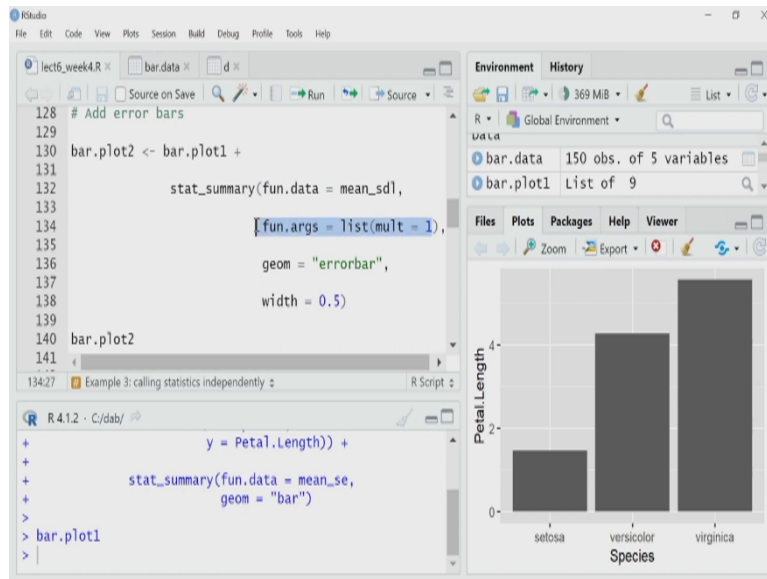
```

bar.plot1

So, now the next step has come that I have to add the error bar, I will again calculate the error bar using the statistical function directly and inside that I will call a geometry element and put the error bar. So, I will add a layer of error bar the geometry is your error bar, but the data is not raw data, but the statistically transformed data, the error.

So, what type of error I can put? For some cases you may want to put the standard error, sometime you may want to put the standard deviation, if I want to calculate standard error, then I can use these mean underscore se in this example, just as an example, I want to show that I want to plot the standard deviation as error bar.

(Refer Slide Time: 20:36)



`bar.plot2 ← bar.plot1 +`

`stat_summary(fun.data = mean_sdl,`

`fun.args = list(mult = 1),`

`geom = "errorbar",`

`width = 0.5)`

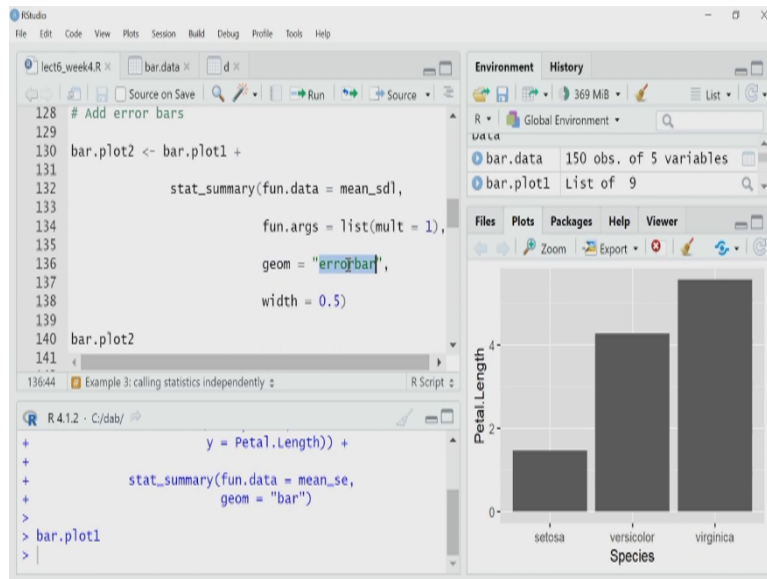
`bar.plot2`

So, to do that what I will do? I will assign mean underscore sdl as the value for the argument fun dot data, right. So, this is the argument for stat underscore summary, again, I am using the same function and then I am saying here fun dot arguments args is equal to list mult equal to 1. Why do I want it?

So, the interesting in this, if you say mean underscore sdl as the function to be used for this statistical summary, R will calculate mean and multiply the standard deviation by a value, by default that value is 2. But so that will be twice a standard deviation, but you do not want that you want the original standard deviation.

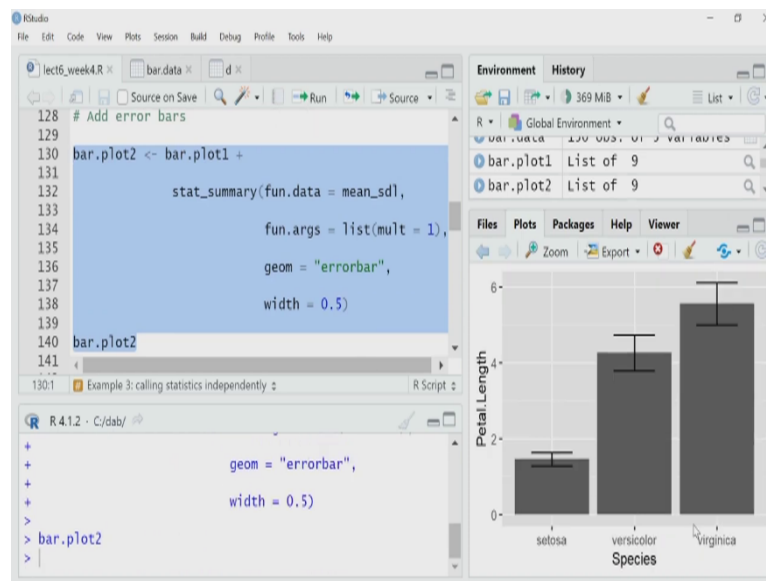
(Refer Slide Time: 21:24)





So, here I am saying multiply by 1. So, that is why I have written mult equal to 1. And again, I am not calling geometry separately; I am calling geom inside the statistical summary function. So, I have written geom equal to error bar. So, I am calling the geometry object of error bar, and I am saying width equal to 0.5 that will define the width of the error bar.

(Refer Slide Time: 21:49)



```
bar.plot2 <- bar.plot1 +
```

```
  stat_summary(fun.data = mean_sdl,
```

```
              fun.args = list(mult = 1),
```

```
              geom = "errorbar",
```

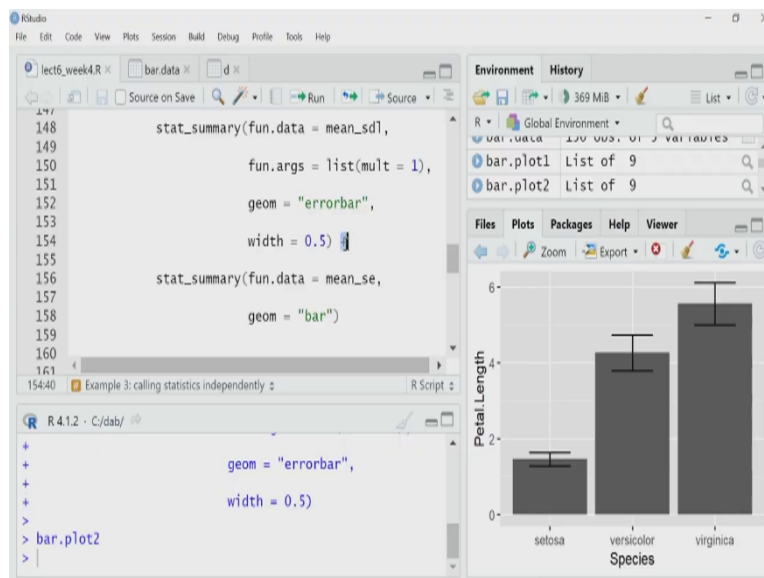
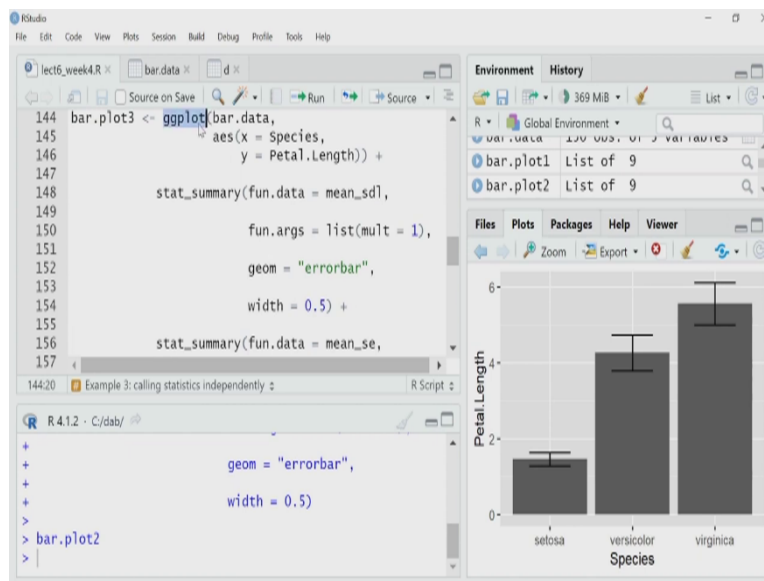
```
              width = 0.5)
```

```
bar.plot2
```

Now, where I am adding this cell here, this statistical summary layer? I am adding it after bar plot 1, the bar plot 1 is already I have drawn and I am adding that using a plus sign. So, this layer will get added over it. So, let me do that now, I have a new plot, the bars were created earlier, the plot one and on that now, I have another statistical layer and the geometries error bar and that is why you can see the error bar in the plot.

Now, you may not be happy with this plot, because usually we try to put the error bar behind the bar. So, as I feel the error bar with solid color, then the lower part of the error bar should not be visible here. So, to do that what I have to do? I have to rearrange these layers. So, the bar layer should be on top, error bar layer should be below. So, that is what I have done, I have rearranged the layerings.

(Refer Slide Time: 22:48)



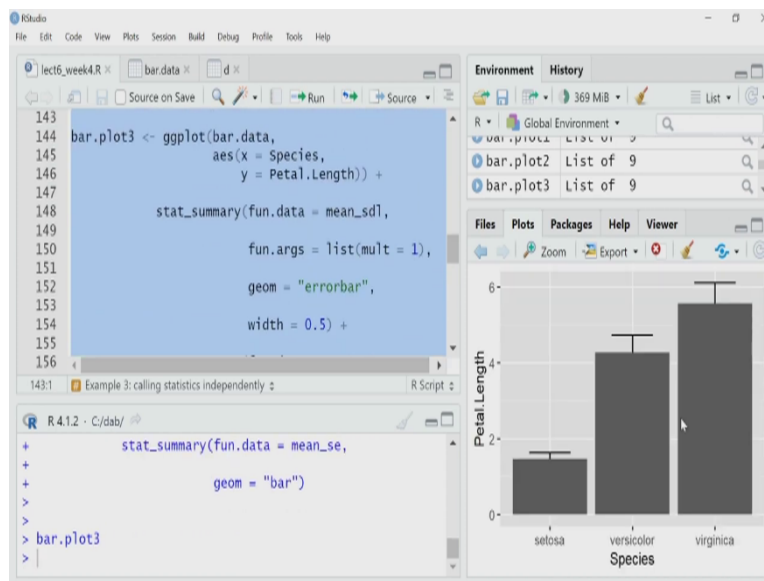
```
bar.plot3 <- ggplot(bar.data,  
  aes(x = Species,  
      y = Petal.Length)) +  
  stat_summary(fun.data = mean_sd1,  
              fun.args = list(mult = 1),  
              geom = "errorbar",  
              width = 0.5) +  
  stat_summary(fun.data = mean_se,  
              geom = "bar",  
              width = 0.5)
```

```
geom = "bar")
```

### bar.plot3

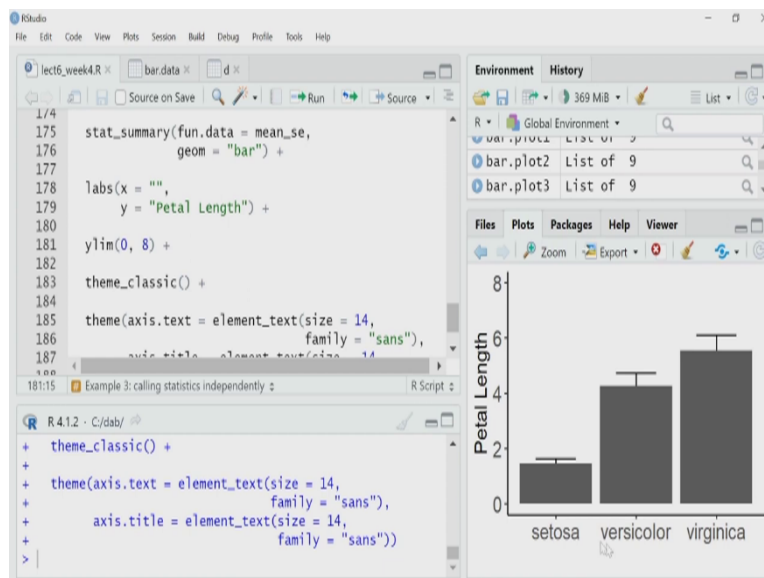
So, the first thing is ggplot function giving the data and then you are defining the aesthetic, then you are adding a layer, which layer you are adding? You are adding the layer for error bar, you are calling the stat\_underscore\_summary function, statistical function and asking it to create an error bar from the standard deviation and then you are adding the last layer by this plus symbol again and this layer will again call is using the statistical element, stat\_underscore\_summary function and it will create a bar plot.

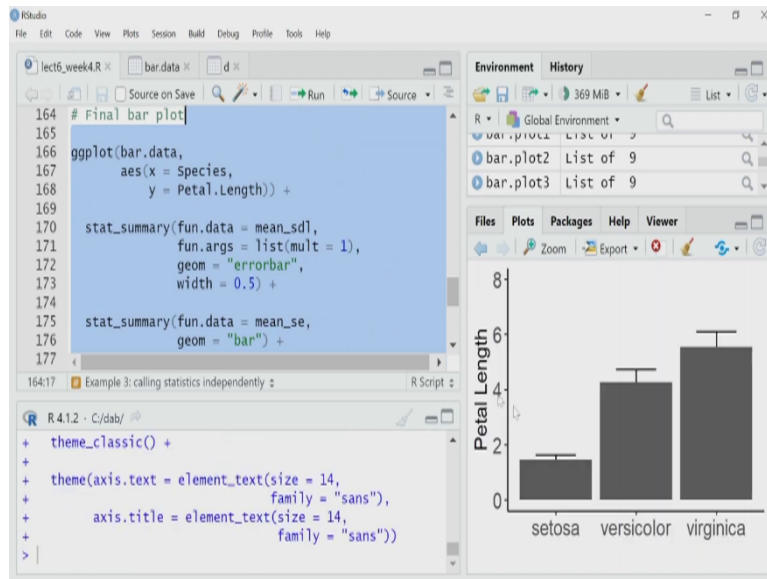
(Refer Slide Time: 23:19)



So, this is how I have rearranged the layer so, that it looks better. So, here is my plot and as you can see, it looks decent. So, what I have done in this case? I have actually called the statistical element, a function for statistical calculation directly and has created this plot and the statistical function has called the geometric property, the geometry inside it to create the geometric object, the error bar and the bar plot.

(Refer Slide Time: 23:51)





```

bar.plot3 ← ggplot(bar.data,
  aes(x = Species,
      y = Petal.Length)) +
  stat_summary(fun.data = mean_sdl,
              fun.args = list(mult = 1),
              geom = "errorbar",
              width = 0.5) +
  stat_summary(fun.data = mean_se,
              geom = "bar") +
  labs(x = "",
      y = "Petal Length") +
  ylim(0, 8) +
  theme_classic() +
  theme(axis.text = element_text(size=14,
                                  family = "sans"),
        axis.title = element_text(size=14,
                                   family = "sans"))

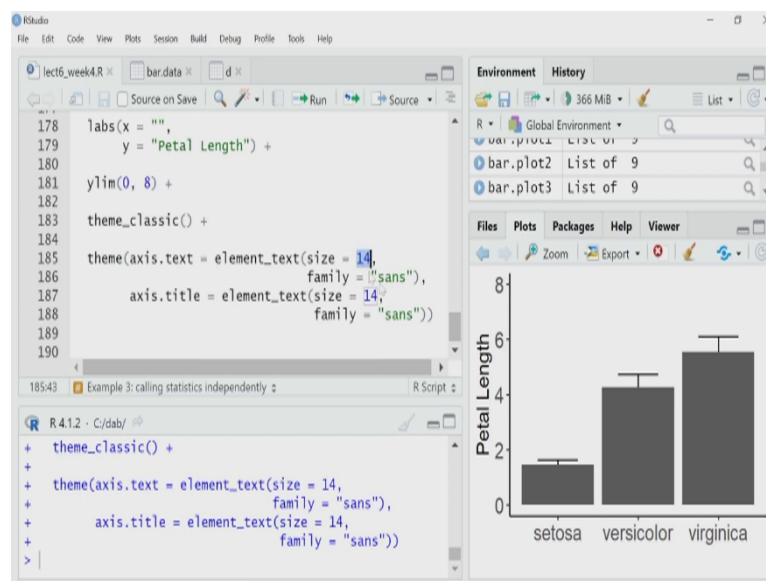
```

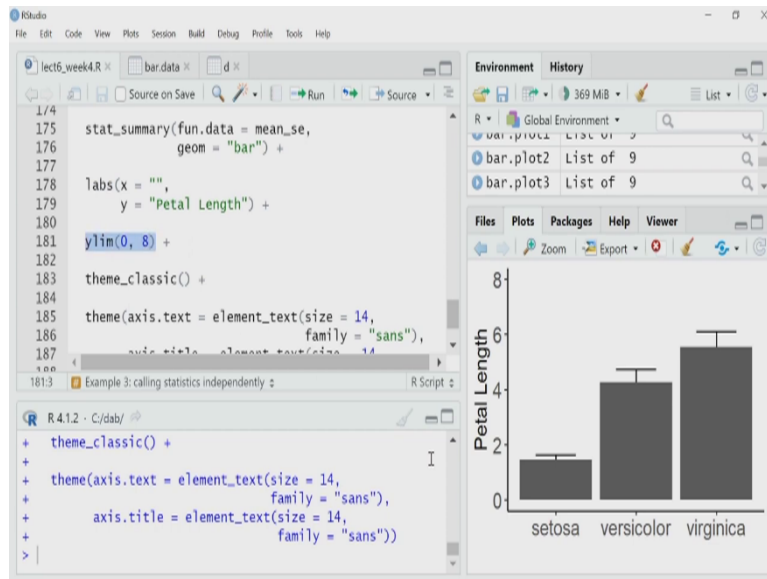
## bar.plot3

Now, at the end of this lecture, what I will do here? I will do some cosmetic changes, so that these diagrams or the plot looks a decent one. So, let me do that. And then I will explain what I am doing. So, here what I have done? Obviously, the layers for statistical summary functions for error bar and the bar remain same.

What I have added? I have added some extra make over thing. So, I do not want any axis label in the horizontal axis, because already I have written them as Setosa, Versicolor and Virginica, I do not need to write species separately in this label, that is why I have kept x axis label empty using these labs function. And on the vertical axis, I have labelled these as petal length.

(Refer Slide Time: 24:41)





I have extended the vertical axis up to 8, so I have said y lim equal to 0 to 8. I am using my favorite classic theme underscore classic. And I am adding another theme layer to set the size of the fonts and also the font type I will use here the sans and that has given me this nice bar diagram. So, that is all for this lecture, this is the extension of previous lecture on ggplot2.

In the first lecture, we have learned about the basics of grammar of graphics and learn how to use three main element data, aesthetic and geometry. In this lecture, I have introduced you to the statistical element, where the data, raw data is not plotted rather a statistical function is called either explicitly or through a geometry function and then that statistically transformed data is plotted. Thank you for being with me today. See you in the next lecture.