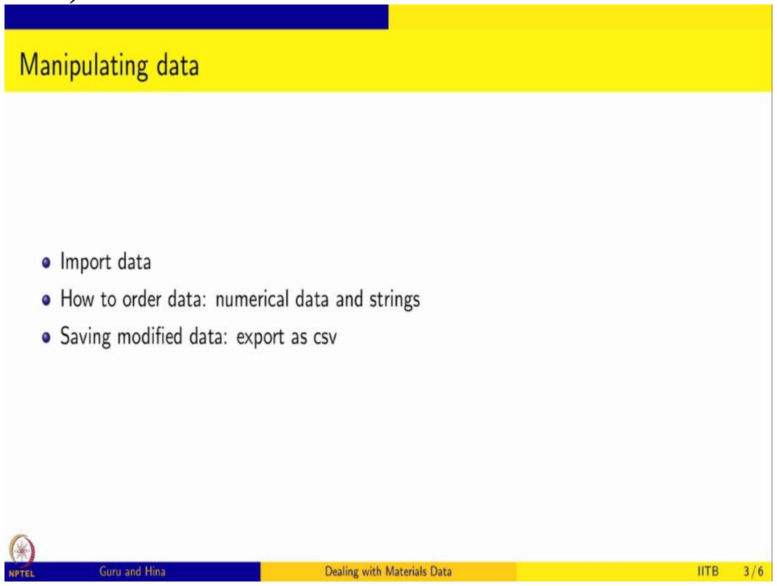
Dealing with Materials Data: Collection, Analysis and Interpretation

Professor M.P. Gururajan
Professor Hina A Gokhale
Department of Metallurgical Engineering and Materials Science
Indian Institute of Technology, Bombay

Lecture No. 17
Property charts: Importing and plotting data

Welcome to dealing with materials data. We are going to look at collection and analysis and interpretation of materials data. Specifically, we are going to look at how to use R to do these things. And this is the first module. This is the module to introduction to R and we are coming almost to the end of this module. So, I want to do a few more data sets and case studies using R to look at the data, import the data, plot the data or manipulate the data and save the figures and save the data and so on and so forth. So, that is what we are going to do in this session. So this is session on case studies in importing and plotting data.

(Refer Slide Time: 1:01)



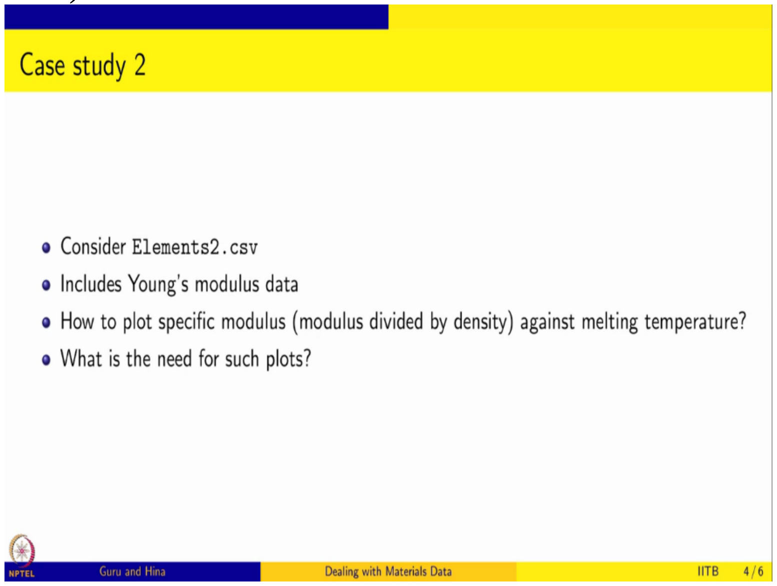
Manipulating data

- Import data
- How to order data: numerical data and strings
- Saving modified data: export as csv

NPTEL Guru and Hina Dealing with Materials Data IITB 3/6

Once you import the data, you can manipulate the data. You can manipulate the data, both the numerical data and the strings. So, I am going to show you an example of how we can do that. And we can save the modified data. So sometimes after you have done the manipulation maybe you want to save this data and you can save it as a CSV file. Like I mentioned, it is also possible to use “fwrite” and write it to a file. But that is something that we are not going to look at, at least at the moment. So, if it is needed later, we will do it. So that is the first one that we are going to do, take a data and do some manipulation on the data and save the modified data.

(Refer Slide Time: 1:50)



Case study 2

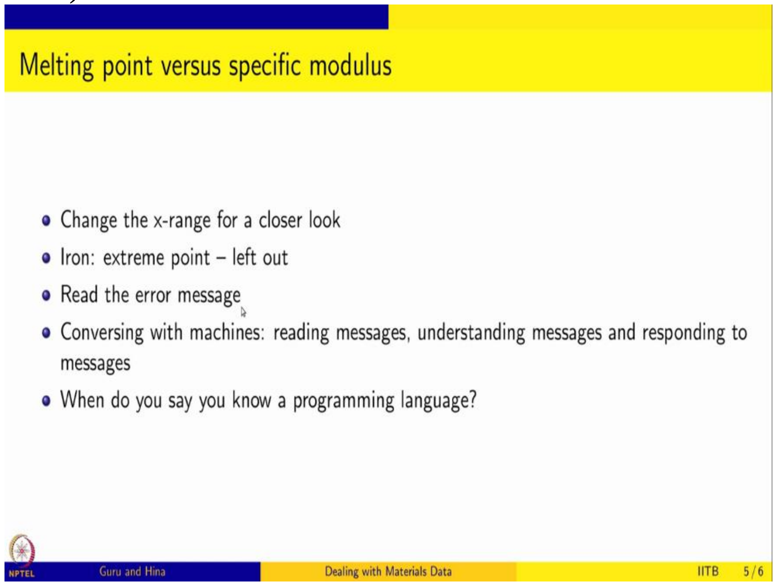
- Consider `Elements2.csv`
- Includes Young's modulus data
- How to plot specific modulus (modulus divided by density) against melting temperature?
- What is the need for such plots?

NPTEL Guru and Hina Dealing with Materials Data IITB 4 / 6

And then the second case study is to consider another data set. It is the same as the elements data set, except that it has now also the Young's modulus data. Once you have the Young's modulus data, we want to know how to plot specific modulus, that is modulus divided by density against the melting temperature.

So previously, we just took the two columns and plotted, but now, we want to do some algebraic manipulations on the columns and such modified data we want to plot and you might be wondering why such plots are needed. These plots are very useful, and they are known as property charts or Ashby plots. And they are used in learning about the properties as well as in the selection of materials for engineering applications. And some of you might have seen in properties and selection courses such plots and so you will see how to generate such plots for yourself.

(Refer Slide Time: 3:01)



Melting point versus specific modulus

- Change the x-range for a closer look
- Iron: extreme point – left out
- Read the error message
- Conversing with machines: reading messages, understanding messages and responding to messages
- When do you say you know a programming language?

Guru and Hina | Dealing with Materials Data | IITB 5 / 6

And so we are going to use the melting point versus specific modulus plot and once we do that, such a plot will also show you the extreme points. So, we are going to plot and we are going to look how the points look like and specifically we are going to learn that iron is an extreme point and so, then we are going to manipulate the plot in such a way that we leave out iron and look at what happens.

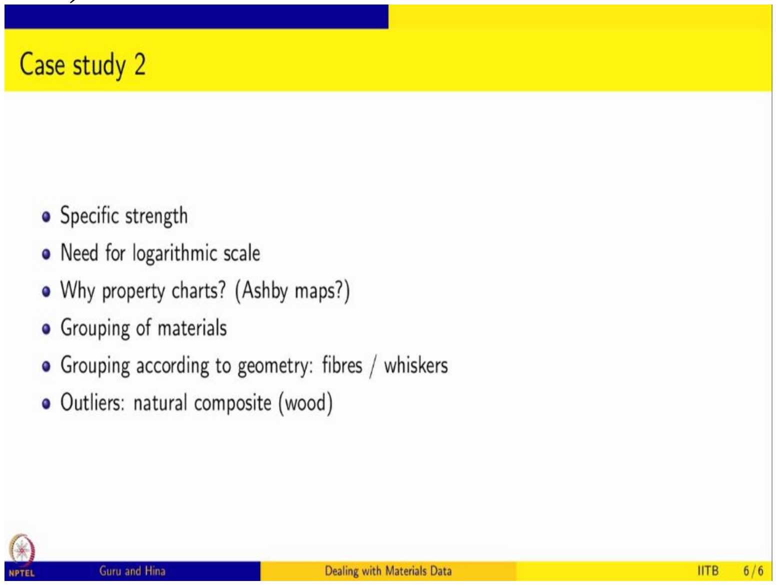
And in the process, there is some programming thing that one learns and I want to emphasize it at this point. Whenever R returns any error message, you should read it and take the corresponding action or if it gives some warnings you have to learn what the warning is about. So, this is what conversing with machines means. You have to read the messages, you have to understand the messages and you have to respond to the messages. So, it is exactly like knowing a language.

If you say you know a specific language, it means that if people ask you something or people tell you something, you will be able to understand and respond to them. In a similar fashion when you say I know R programming language, what is expected is that you are able to converse with the machine if it gives some messages, be it error or warning messages you are able to read, understand and respond.

I know that the tendency for beginning programmers is to go look everything up on the internet, and then copy paste the solutions. It is sometimes useful and it can help you pick things fast. But if you want to be a good programmer, at some point, it is better to read the error messages and try to decode them yourself before taking help from the net. Of course, net is a very good resource and you should utilize it but before doing that it is useful sometimes to try it out on your own.

Because that way even when you get the solution from the net, you will have a better understanding. So, it is important to read the error and warning messages and you will see that when we do some of the manipulations, it does give some messages and you should pay attention to those messages.

(Refer Slide Time: 5:22)



Case study 2

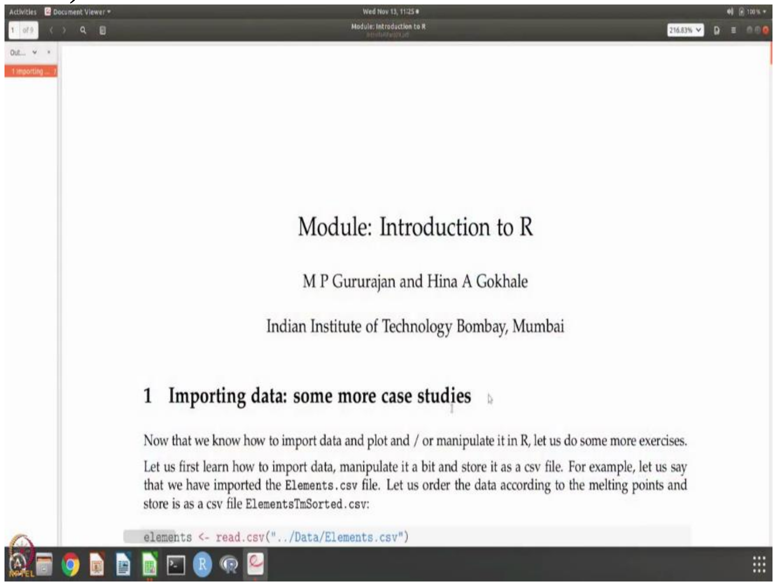
- Specific strength
- Need for logarithmic scale
- Why property charts? (Ashby maps?)
- Grouping of materials
- Grouping according to geometry: fibres / whiskers
- Outliers: natural composite (wood)

NPTEL Guru and Hina Dealing with Materials Data IITB 6 / 6

Okay, the second case study that we are going to do is for specific strength, and we will see that for things like Ashby plot, sometimes the logarithmic scale is needed. And you will also see why property charts or Ashby maps are useful and you are going to notice that the materials are grouped according to the kind of material it is, and you will also see some grouping according to geometry, for example, all fibers and whiskers irrespective of what type of material they are going to fall in one way and it also is very useful to notice the outliers and in this case the outlier happens to be a natural composite, namely wood. So, this is what we want to do in this session.

We are going to take some data in CSV format, we are going to import them, we are going to use GG plot for plotting and we are going to do with a little bit of manipulation and we are going to pay more attention to the data and try to learn about the data by doing this and so this is going to be our almost the last session on introduction to R. Okay, so let us go, do that.

(Refer Slide Time: 6:41)



Module: Introduction to R

M P Gururajan and Hina A Gokhale

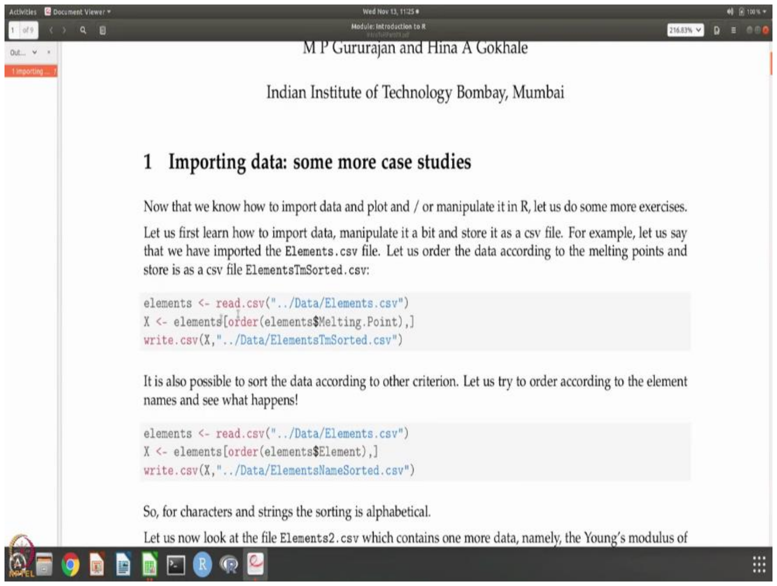
Indian Institute of Technology Bombay, Mumbai

1 Importing data: some more case studies

Now that we know how to import data and plot and / or manipulate it in R, let us do some more exercises.

Let us first learn how to import data, manipulate it a bit and store it as a csv file. For example, let us say that we have imported the `Elements.csv` file. Let us order the data according to the melting points and store is as a csv file `ElementsTmSorted.csv`:

```
elements <- read.csv("../Data/Elements.csv")
```



M P Gururajan and Hina A Gokhale

Indian Institute of Technology Bombay, Mumbai

1 Importing data: some more case studies

Now that we know how to import data and plot and / or manipulate it in R, let us do some more exercises.

Let us first learn how to import data, manipulate it a bit and store it as a csv file. For example, let us say that we have imported the `Elements.csv` file. Let us order the data according to the melting points and store is as a csv file `ElementsTmSorted.csv`:

```
elements <- read.csv("../Data/Elements.csv")
X <- elements[order(elements$Melting.Point),]
write.csv(X,"../Data/ElementsTmSorted.csv")
```

It is also possible to sort the data according to other criterion. Let us try to order according to the element names and see what happens!

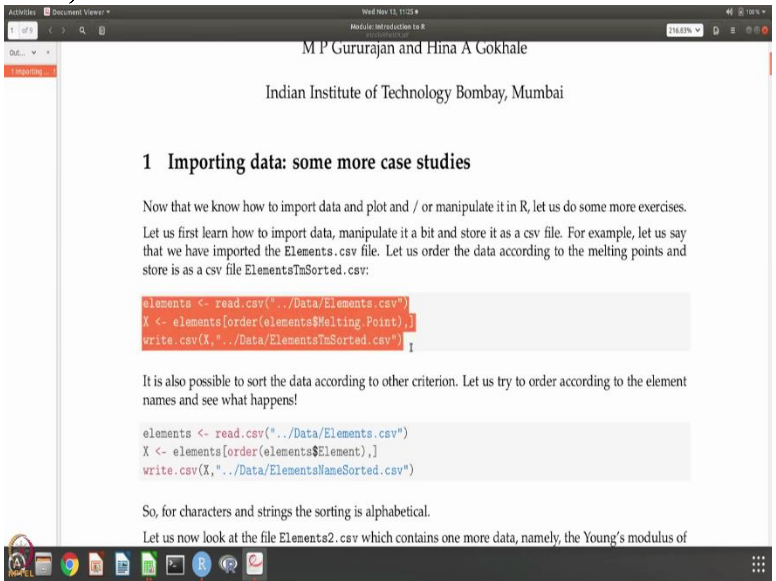
```
elements <- read.csv("../Data/Elements.csv")
X <- elements[order(elements$Element),]
write.csv(X,"../Data/ElementsNameSorted.csv")
```

So, for characters and strings the sorting is alphabetical.

Let us now look at the file `Elements2.csv` which contains one more data, namely, the Young's modulus of

So, this is the first one is to take the elements data that we have already done.

(Refer Slide Time: 6:51)



M P Gururajan and Hina A Gokhale

Indian Institute of Technology Bombay, Mumbai

1 Importing data: some more case studies

Now that we know how to import data and plot and / or manipulate it in R, let us do some more exercises.

Let us first learn how to import data, manipulate it a bit and store it as a csv file. For example, let us say that we have imported the `Elements.csv` file. Let us order the data according to the melting points and store is as a csv file `ElementsTmSorted.csv`:

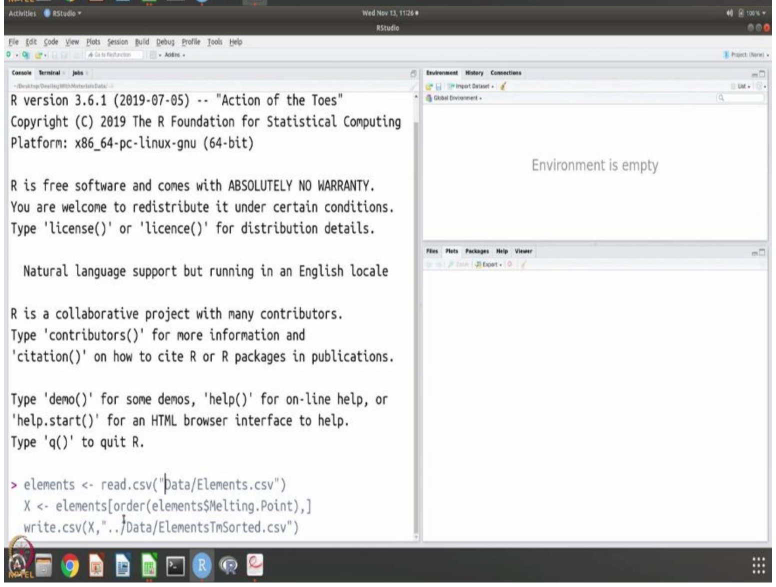
```
elements <- read.csv("../Data/Elements.csv")
X <- elements[order(elements$Melting.Point),]
write.csv(X,"../Data/ElementsTmSorted.csv")
```

It is also possible to sort the data according to other criterion. Let us try to order according to the element names and see what happens!

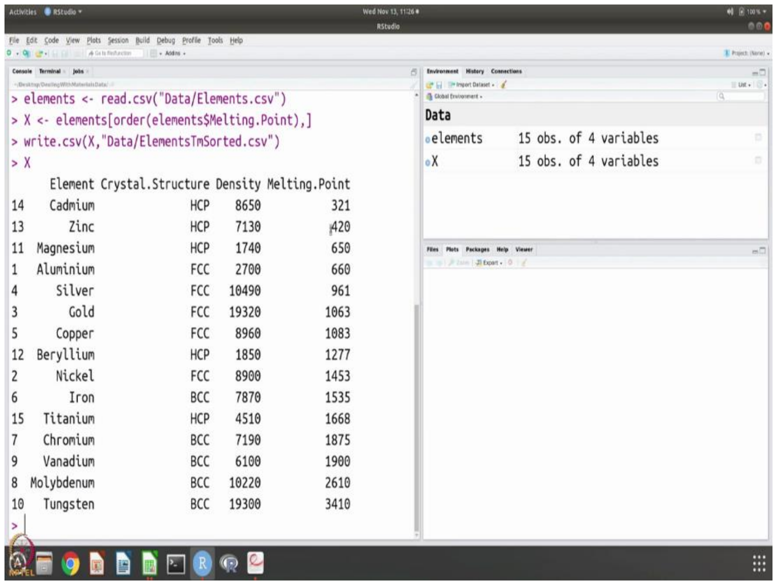
```
elements <- read.csv("../Data/Elements.csv")
X <- elements[order(elements$Element),]
write.csv(X,"../Data/ElementsNameSorted.csv")
```

So, for characters and strings the sorting is alphabetical.

Let us now look at the file `Elements2.csv` which contains one more data, namely, the Young's modulus of



```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> elements <- read.csv("Data/Elements.csv")
  X <- elements[order(elements$Melting.Point),]
  write.csv(X,"../Data/ElementsTmSorted.csv")
```



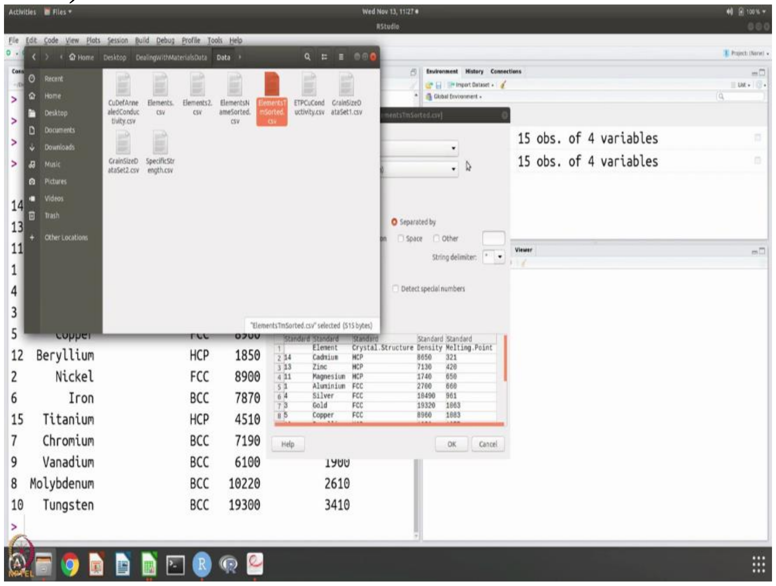
```
> elements <- read.csv("Data/Elements.csv")
> X <- elements[order(elements$Melting.Point),]
> write.csv(X,"Data/ElementsTmSorted.csv")
> X
      Element Crystal.Structure Density Melting.Point
14    Cadmium               HCP    8650           321
13       Zinc               HCP    7130           420
11  Magnesium               HCP    1740           650
1   Aluminium               FCC    2700           660
4      Silver               FCC   10490           961
3        Gold               FCC   19320          1063
5      Copper               FCC    8960          1083
12  Beryllium               HCP    1850          1277
2      Nickel               FCC    8900          1453
6        Iron               BCC    7870          1535
15   Titanium               HCP    4510          1668
7    Chromium               BCC    7190          1875
9    Vanadium               BCC    6100          1900
8  Molybdenum               BCC   10220          2610
10   Tungsten               BCC   19300          3410
>
```

```
Data
elements      15 obs. of 4 variables
X             15 obs. of 4 variables
```

So, let me open R and let me take these commands and put them here. So elements, so we are going to read the CSV file from data and then we are going to order the elements according to melting point and we are going to write this data into a new CSV file called elements melting point sorted. So, that is the thing we want to do. And so you can see X, of course we can see X and you can see that it is sorted according to melting point.

You know previously it was first aluminum or something and you can see the original numbering it was 1 here and then 2 was nickel and 3 was gold and so on, 4 was silver and so on. But that ordering has been changed because now it is going by the increasing order of the melting point. So, it is 321, 426, 650, 660, etc. And so we have written a new data file and that is there in the elements tm sorted.

(Refer Slide Time: 8:16)

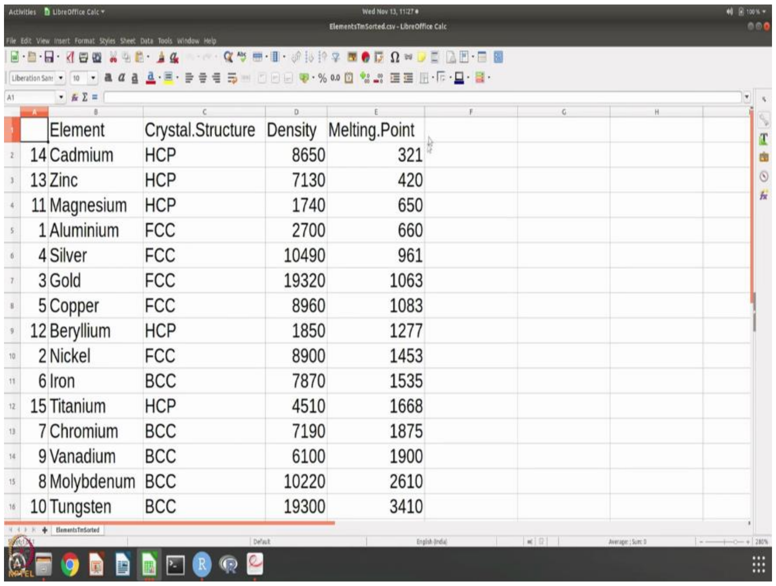


"ElementsTmSorted.csv" selected (515 bytes)

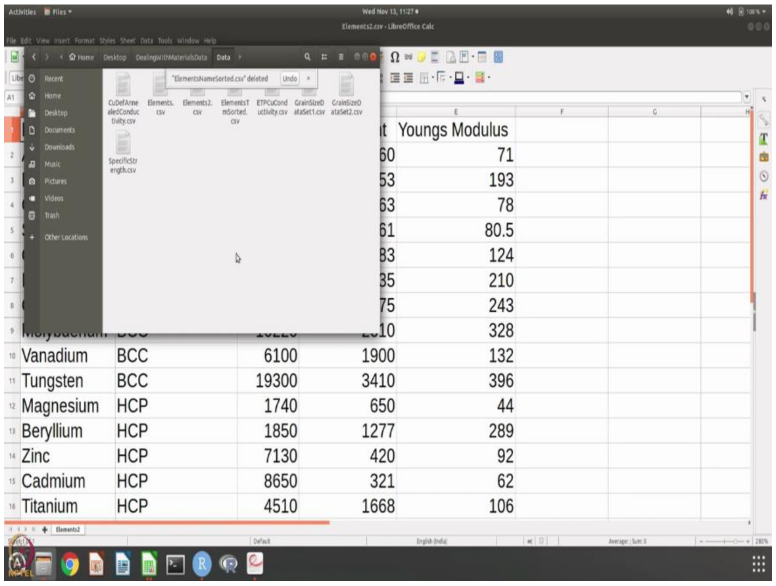
```
15 obs. of 4 variables
15 obs. of 4 variables
```

```
14
13
11
1
4
3
5     Copper         FCC    8960
12  Beryllium        HCP    1850
2      Nickel        FCC    8900
6        Iron        BCC    7870
15   Titanium        HCP    4510
7    Chromium        BCC    7190
9    Vanadium        BCC    6100         1900
8  Molybdenum        BCC   10220         2610
10   Tungsten        BCC   19300         3410
>
```

Standard	Standard	Standard	Standard	Standard
	Element	Crystal.Structure	Density	Melting.Point
14	Cadmium	HCP	8650	321
13	Zinc	HCP	7130	420
11	Magnesium	HCP	1740	650
1	Aluminium	FCC	2700	660
4	Silver	FCC	10490	961
3	Gold	FCC	19320	1063
5	Copper	FCC	8960	1083


Activities
LibreOffice Calc
Wed Nov 13, 11:27
ElementsTmSorted.csv - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
ElementsTmSorted

	Element	Crystal.Structure	Density	Melting.Point
14	Cadmium	HCP	8650	321
13	Zinc	HCP	7130	420
11	Magnesium	HCP	1740	650
1	Aluminium	FCC	2700	660
4	Silver	FCC	10490	961
3	Gold	FCC	19320	1063
5	Copper	FCC	8960	1083
12	Beryllium	HCP	1850	1277
2	Nickel	FCC	8900	1453
6	Iron	BCC	7870	1535
15	Titanium	HCP	4510	1668
7	Chromium	BCC	7190	1875
9	Vanadium	BCC	6100	1900
8	Molybdenum	BCC	10220	2610
10	Tungsten	BCC	19300	3410

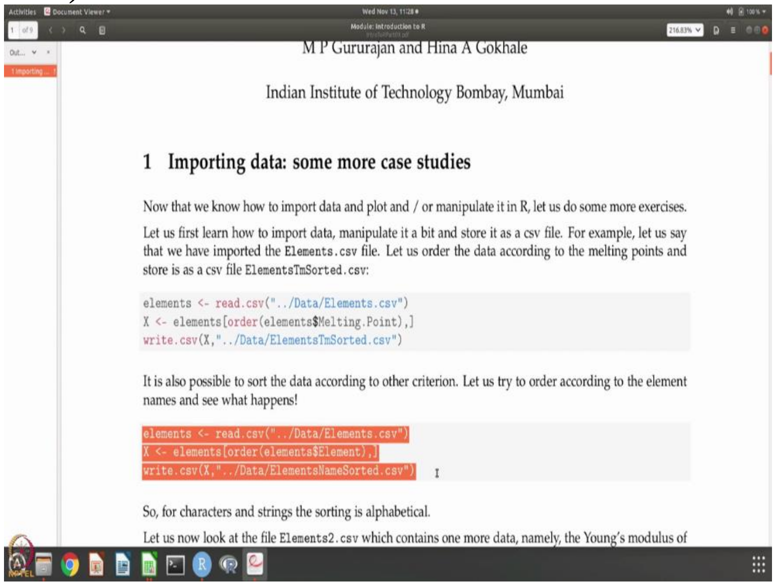


"ElementsNameSorted.csv" deleted

				Youngs Modulus
				71
				193
				78
				80.5
				124
				210
				243
				328
Vanadium	BCC	6100	1900	132
Tungsten	BCC	19300	3410	396
Magnesium	HCP	1740	650	44
Beryllium	HCP	1850	1277	289
Zinc	HCP	7130	420	92
Cadmium	HCP	8650	321	62
Titanium	HCP	4510	1668	106

So, let us go there and see. So, there is this element, TM sorted, so let us say So according to melting point they are now listed and this data is available and to do this, okay, so I also have this name sorted, let me remove this file. Okay, so I am going to remove this file. So in this directory, there is no name sorted now. But let us do sorting according to names and write that file.

(Refer Slide Time: 9:05)



M P Gururajan and Hina A Gokhale

Indian Institute of Technology Bombay, Mumbai

1 Importing data: some more case studies

Now that we know how to import data and plot and / or manipulate it in R, let us do some more exercises.

Let us first learn how to import data, manipulate it a bit and store it as a csv file. For example, let us say that we have imported the `Elements.csv` file. Let us order the data according to the melting points and store is as a csv file `ElementsTmSorted.csv`:

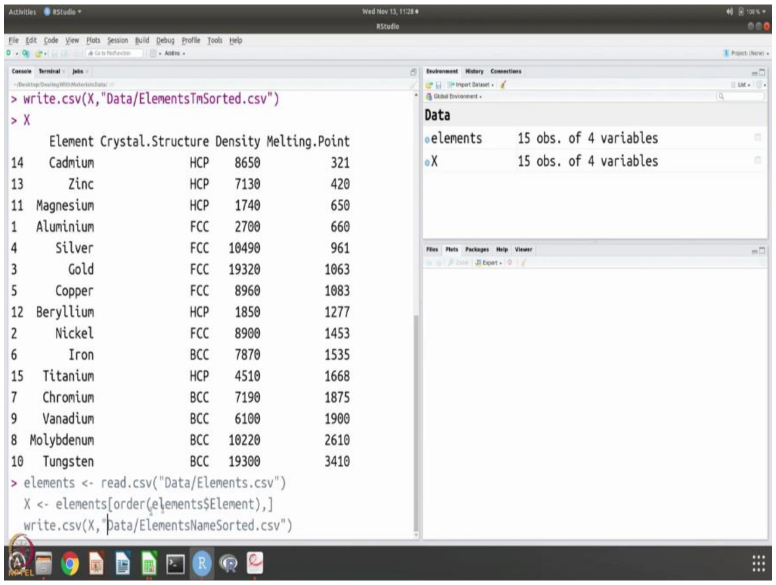
```
elements <- read.csv("../Data/Elements.csv")
X <- elements[order(elements$Melting.Point),]
write.csv(X,"../Data/ElementsTmSorted.csv")
```

It is also possible to sort the data according to other criterion. Let us try to order according to the element names and see what happens!

```
elements <- read.csv("../Data/Elements.csv")
X <- elements[order(elements$Element),]
write.csv(X,"../Data/ElementsNameSorted.csv")
```

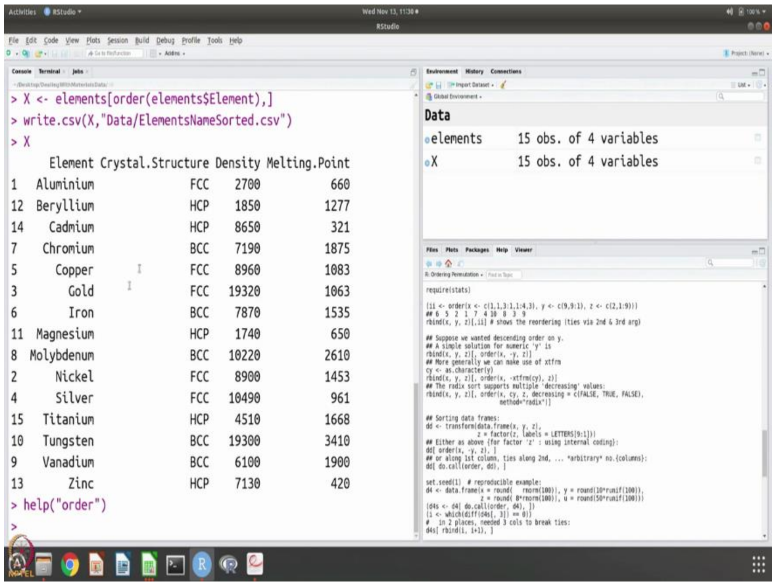
So, for characters and strings the sorting is alphabetical.

Let us now look at the file `Elements2.csv` which contains one more data, namely, the Young's modulus of



```
> write.csv(X,"Data/ElementsTmSorted.csv")
> X
      Element Crystal.Structure Density Melting.Point
14    Cadmium               HCP    8650           321
13       Zinc               HCP    7130           420
11  Magnesium               HCP    1740           650
1   Aluminium               FCC    2700           660
4      Silver               FCC   10490           961
3        Gold               FCC   19320          1063
5      Copper               FCC    8960          1083
12  Beryllium               HCP    1850          1277
2      Nickel               FCC    8900          1453
6        Iron               BCC    7870          1535
15   Titanium               HCP    4510          1668
7    Chromium               BCC    7190          1875
9    Vanadium               BCC    6100          1900
8  Molybdenum               BCC   10220          2610
10   Tungsten               BCC   19300          3410
> elements <- read.csv("Data/Elements.csv")
  X <- elements[order(elements$Element),]
  write.csv(X,"Data/ElementsNameSorted.csv")
```

```
Data
elements     15 obs. of 4 variables
X            15 obs. of 4 variables
```



```
> X <- elements[order(elements$Element),]
> write.csv(X,"Data/ElementsNameSorted.csv")
> X
      Element Crystal.Structure Density Melting.Point
1   Aluminium               FCC    2700           660
12  Beryllium               HCP    1850          1277
14    Cadmium               HCP    8650           321
7    Chromium               BCC    7190          1875
5      Copper               FCC    8960          1083
3        Gold               FCC   19320          1063
6        Iron               BCC    7870          1535
11  Magnesium               HCP    1740           650
8  Molybdenum               BCC   10220          2610
2      Nickel               FCC    8900          1453
4      Silver               FCC   10490           961
15   Titanium               HCP    4510          1668
10   Tungsten               BCC   19300          3410
9    Vanadium               BCC    6100          1900
13       Zinc               HCP    7130           420
> help("order")
>
```

```
Data
elements     15 obs. of 4 variables
X            15 obs. of 4 variables
```

```
require(stats)

(ii <- order(x <- c(1,1,3:1,1:4,3), y <- c(9,9:1), z <- c(2,1:9)))
## 6  5  2  1  7  4 10  8  3  9
rbind(x, y, z)[,ii] # shows the reordering (ties via 2nd & 3rd arg)

## Suppose we wanted descending order on y.
## A simple solution for numeric 'y' is
rbind(x, y, z)[, order(x, -y, z)]
## More generally we can make use of xtfrm
cy <- as.character(y)
rbind(x, y, z)[, order(x, -xtfrm(cy), z)]
## The radix sort supports multiple 'decreasing' values:
rbind(x, y, z)[, order(x, cy, z, decreasing = c(FALSE, TRUE, FALSE),
                       method="radix")]

## Sorting data frames:
dd <- transform(data.frame(x, y, z),
                z = factor(z, labels = LETTERS[9:1]))
## Either as above {for factor 'z' : using internal coding}:
dd[ order(x, -y, z), ]
## or along 1st column, ties along 2nd, ... *arbitrary* no.{columns}:
dd[ do.call(order, dd), ]

set.seed(1)  # reproducible example:
d4 <- data.frame(x = round(   rnorm(100)), y = round(10*runif(100)),
                 z = round( 8*rnorm(100)), u = round(50*runif(100)))
(d4s <- d4[ do.call(order, d4), ])
(i <- which(diff(d4s[, 3]) == 0))
#   in 2 places, needed 3 cols to break ties:
d4s[ rbind(i, i+1), ]
```

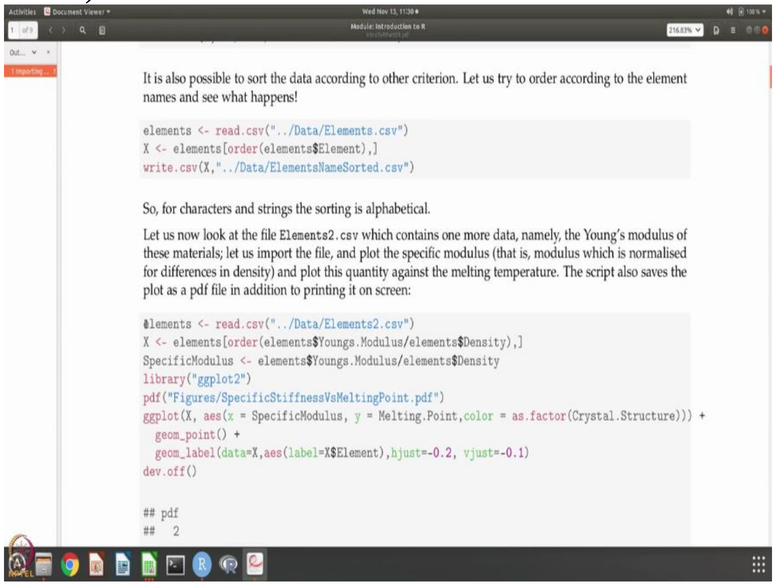
So, let us do that. It is here, where we are going to do. So, again we are going to read the data and we are going to write that data and we are going to, so the difference between the previous one and this one is that previously we said okay elements according to melting point you have to order and here we are going to say that order according to the names of element, element right.

So, what does it mean to say order according to this? As you will see it will alphabetically ordered and the file will then be written to a CSV file. So, now let us say x and you can see aluminum, beryllium, cadmium, chromium, copper, etc. So, this is now alphabetically ordered and this is what would be written here.

So, element name sorted or CSV is available for us now, and as you can see now it is sorted according to the alphabetical order. So, in other words, it is possible to take data, import it and manipulate it and you can manipulate numerical data as well as strings. So, you can alphabetically order or order according to the numbers, increasing, decreasing or ascending, descending order and then you can write them. So, the write dot CSV is the equivalent of read.CSV for writing CSV files.

And you can also, so, help order will also tell you, for example, if you want to do not ascending order but descending order, what should you do? So, decreasing equal to false, which would mean that increasing is true. So, you can order. So should be, should be sort order be increasing or decreasing. Okay, so that is what this does, so you can use the increasing command. And so you can read the help file to get more information. But this tells you how to manipulate data and order them and write them and so on, so forth.

(Refer Slide Time: 11:40)



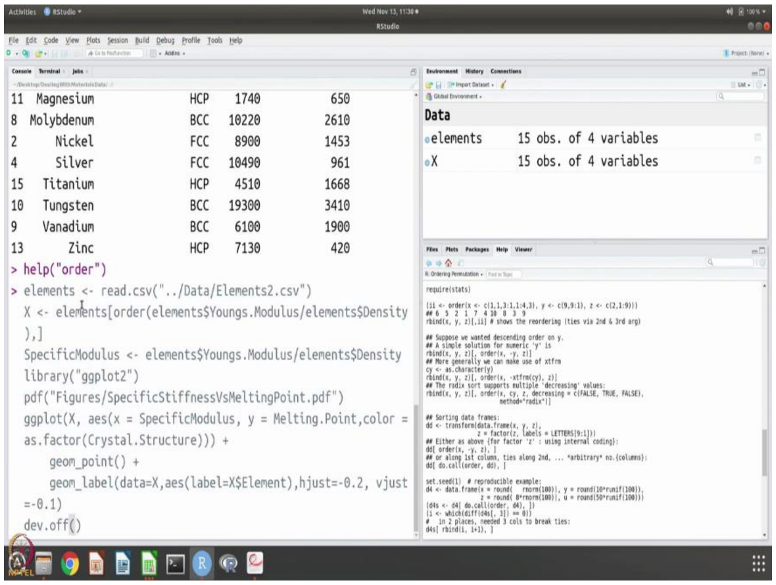
It is also possible to sort the data according to other criterion. Let us try to order according to the element names and see what happens!

```
elements <- read.csv("../Data/Elements.csv")
X <- elements[order(elements$Element),]
write.csv(X,"../Data/ElementsNameSorted.csv")
```

So, for characters and strings the sorting is alphabetical.

Let us now look at the file `Elements2.csv` which contains one more data, namely, the Young's modulus of these materials; let us import the file, and plot the specific modulus (that is, modulus which is normalised for differences in density) and plot this quantity against the melting temperature. The script also saves the plot as a pdf file in addition to printing it on screen:

```
elements <- read.csv("../Data/Elements2.csv")
X <- elements[order(elements$Youngs.Modulus/elements$Density),]
SpecificModulus <- elements$Youngs.Modulus/elements$Density
library("ggplot2")
pdf("Figures/SpecificStiffnessVsMeltingPoint.pdf")
ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color = as.factor(Crystal.Structure))) +
  geom_point() +
  geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust=-0.1)
dev.off()


## pdf
##   2
```



```
11  Magnesium            HCP   1740         650
8  Molybdenum            BCC  10220        2610
2      Nickel            FCC   8900        1453
4      Silver            FCC  10490         961
15   Titanium            HCP   4510        1668
10   Tungsten            BCC  19300        3410
9    Vanadium            BCC   6100        1900
13       Zinc            HCP   7130         420
> help("order")
> elements <- read.csv("../Data/Elements2.csv")
  X <- elements[order(elements$Youngs.Modulus/elements$Density
  ),]
  SpecificModulus <- elements$Youngs.Modulus/elements$Density
  library("ggplot2")
  pdf("Figures/SpecificStiffnessVsMeltingPoint.pdf")
  ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
  as.factor(Crystal.Structure))) +
      geom_point() +
      geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
  =-0.1)
  dev.off()
```

```
Data
elements    15 obs. of 4 variables
X           15 obs. of 4 variables
```

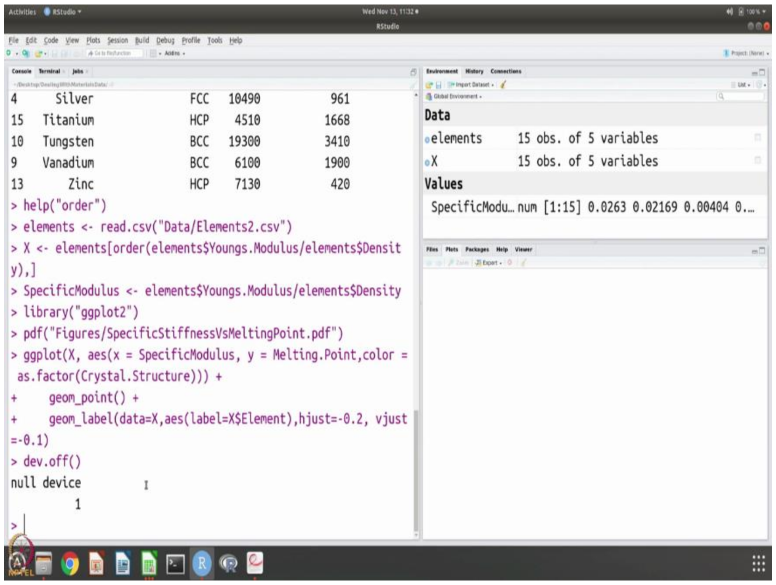
So, now let us look at this second data file. Like I said, there is an elements2 dot CSV data file, which has the extra information. So, in addition to this data, it also has the Young’s modulus. And so we want to read this data and work with this data. So, this is what we are going to do. Okay, so let us do this. Let us find out what this command is.

So, we are going to read into elements the data from elements2 dot CSV and we are going to order and the ordering is according to the specific modulus because elements Young’s modulus by elements density is the specific modulus. So, we are going to order the data according to the specific modulus, this just as an exercise. So, it is not essential that you have to order. And we are going to say specific modulus is this quantity.

Okay, that is what we are going to show, store. And we are going to now use ggplot2 to plot this specific stiffness, that is the specific modulus versus melting point and we are going to save it as a PDF file. And so the device off means that it will execute all these plotting commands and then close the PDF file and the plotting commands as usual.

So it is ggplot, it says take the X data, aesthetics is x is specific modulus, Y is melting point and color should be according to crystal structure and the geometry is that it is a scatterplot, it is a point and you are going to label the points of course according to element and there is a justification for the label. So, that is what this plot commands are. So, this we have done several times.

(Refer Slide Time: 13:30)

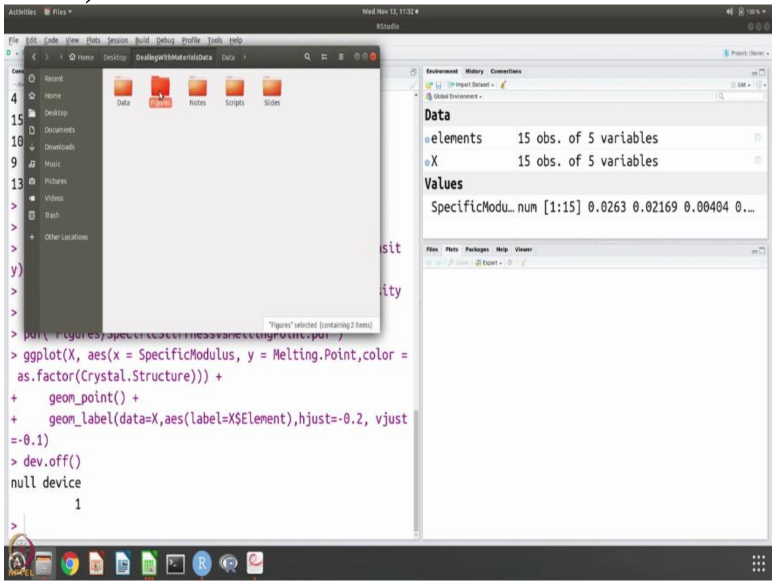


```
4      Silver                  FCC   10490           961
15   Titanium                  HCP    4510          1668
10   Tungsten                  BCC   19300          3410
9    Vanadium                  BCC    6100          1900
13       Zinc                  HCP    7130           420
> help("order")
> elements <- read.csv("Data/Elements2.csv")
> X <- elements[order(elements$Youngs.Modulus/elements$Densit
y),]
> SpecificModulus <- elements$Youngs.Modulus/elements$Density
> library("ggplot2")
> pdf("Figures/SpecificStiffnessVsMeltingPoint.pdf")
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
> dev.off()
null device
          1
>
```

```
Data
elements     15 obs. of 5 variables
X            15 obs. of 5 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```

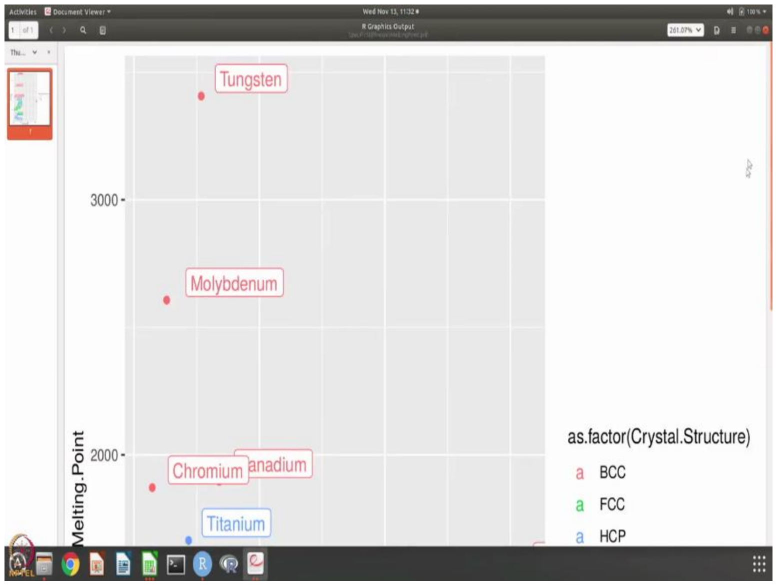
And see when you do this, now, there is no plot that you can see here.

(Refer Slide Time: 13:36)



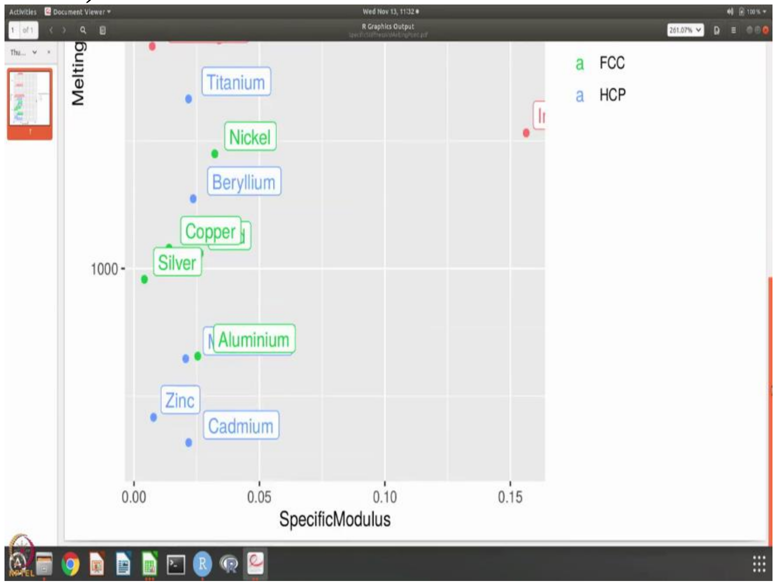
```
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
> dev.off()
null device
          1
>
```

```
Data
elements     15 obs. of 5 variables
X            15 obs. of 5 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```



That is because the PDF file actually wrote the figure here.

(Refer Slide Time: 13:45)

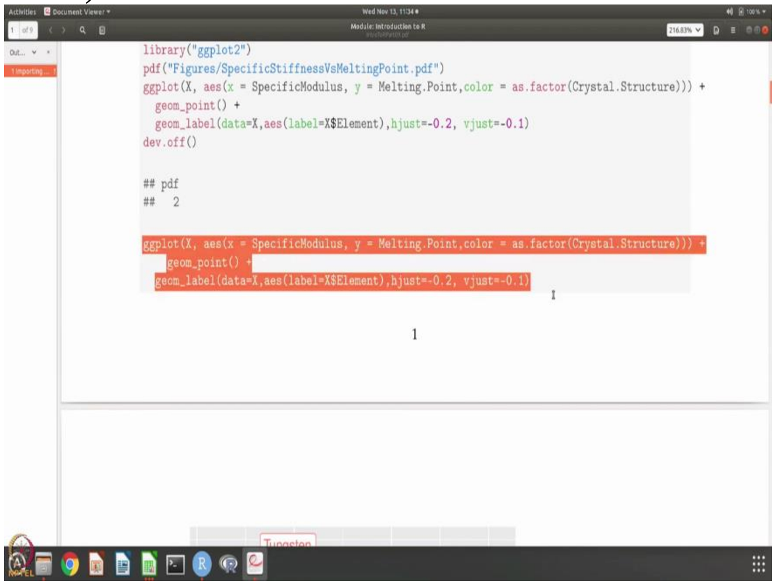


So, this is what the data you have. And as you can see, the specific modulus versus melting point, most of them fall in this column. There is nothing beyond the 0.05 or 06, nothing beyond 0.05 actually except for iron which is really really an outlier. So, its specific modulus is like, that is modulus divided by its density is very high, it is falling somewhere beyond 0.15. So, as compared to all the other elements, iron is really an outlier.

So, this you can see. So, in other words, the modulus with respect to density, if you normalize by the density values, different elements have different densities. And if you account for those differences, then you see that the relative ratio is different for iron as compared to all these other materials. And within of course a given specific modulus now the melting temperature is different for different metals.

So, you can see that there are many points, cadmium, beryllium, titanium, and tungsten, they are all falling almost on the similar values but their melting points is different. So this is what we see. And so and there was no plot because we plotted it to the file.

(Refer Slide Time: 15:33)

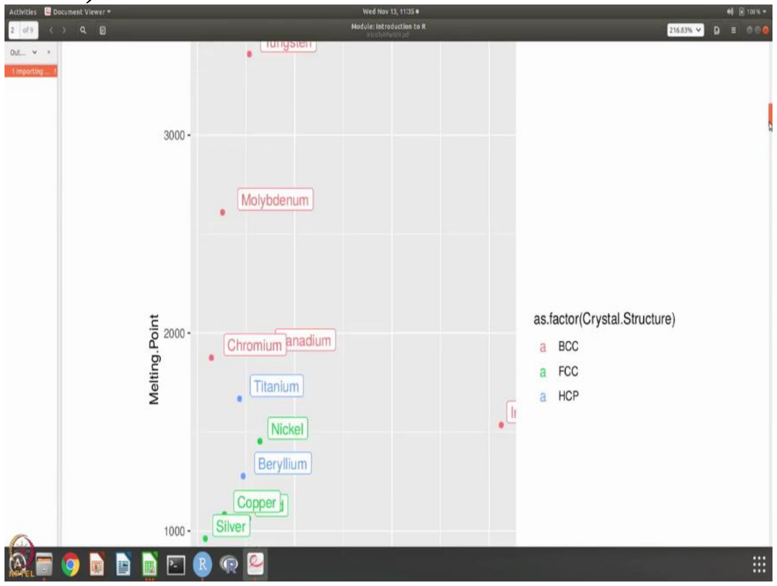


```
library("ggplot2")
pdf("Figures/SpecificStiffnessVsMeltingPoint.pdf")
ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color = as.factor(Crystal.Structure))) +
  geom_point() +
  geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust=-0.1)
dev.off()

## pdf
##   2

ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color = as.factor(Crystal.Structure))) +
    geom_point() +
  geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust=-0.1)
```

1

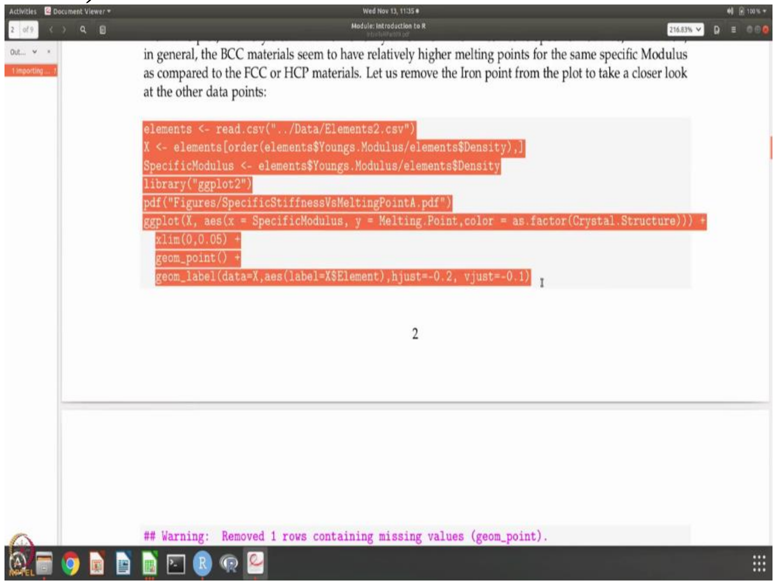
So, let us now go back and do some more manipulation. So of course, you can get a plot by now, after a dev off. If you put the command then you will get the plot here. Again, showing that iron is really, really an outlier (())(15:55).

(Refer Slide Time: 15:57)



So, let us do the next one. Next one is that, so we want to take a closer look at these values and because that is an outlier, we really do not want beyond this point.

(Refer Slide Time: 16:15)


in general, the BCC materials seem to have relatively higher melting points for the same specific Modulus as compared to the FCC or HCP materials. Let us remove the Iron point from the plot to take a closer look at the other data points:

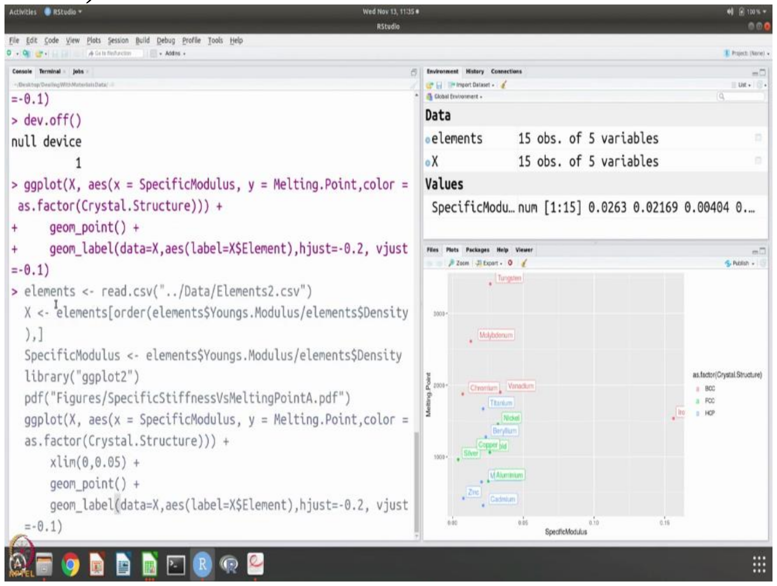
```
elements <- read.csv("../Data/Elements2.csv")
X <- elements[order(elements$Youngs.Modulus/elements$Density),]
SpecificModulus <- elements$Youngs.Modulus/elements$Density
library("ggplot2")
pdf("Figures/SpecificStiffnessVsMeltingPointA.pdf")
ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color = as.factor(Crystal.Structure))) +
  xlim(0,0.05) +
  geom_point() +
  geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust=-0.1)
```

2

```
## Warning:  Removed 1 rows containing missing values (geom_point).
```

So, we are going to change the range when we are doing the X plot. So that is what we are going to do.

(Refer Slide Time: 16:21)

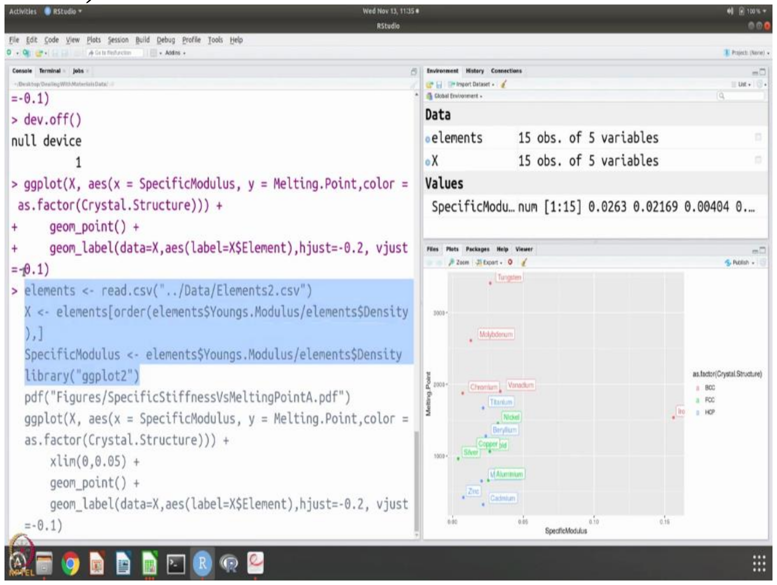


```
=-0.1)
> dev.off()
null device
          1
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
> elements <- read.csv("../Data/Elements2.csv")
  X <- elements[order(elements$Youngs.Modulus/elements$Density
  ),]
  SpecificModulus <- elements$Youngs.Modulus/elements$Density
  library("ggplot2")
  pdf("Figures/SpecificStiffnessVsMeltingPointA.pdf")
  ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
  as.factor(Crystal.Structure))) +
      xlim(0,0.05) +
      geom_point() +
      geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
  =-0.1)
```

```
Data
elements      15 obs. of 5 variables
X             15 obs. of 5 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```

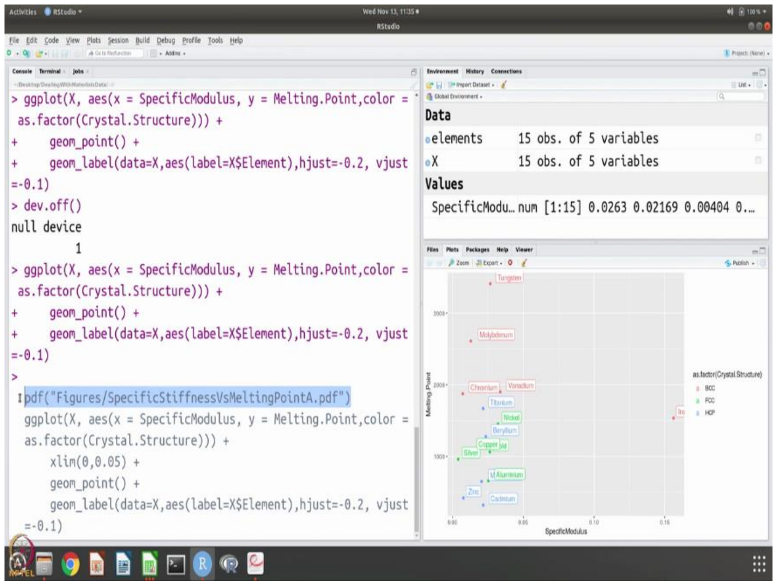
Now, let us go back and do it here. So, we are going to read, so the reading and this thing has been done already in the data.

(Refer Slide Time: 16:33)



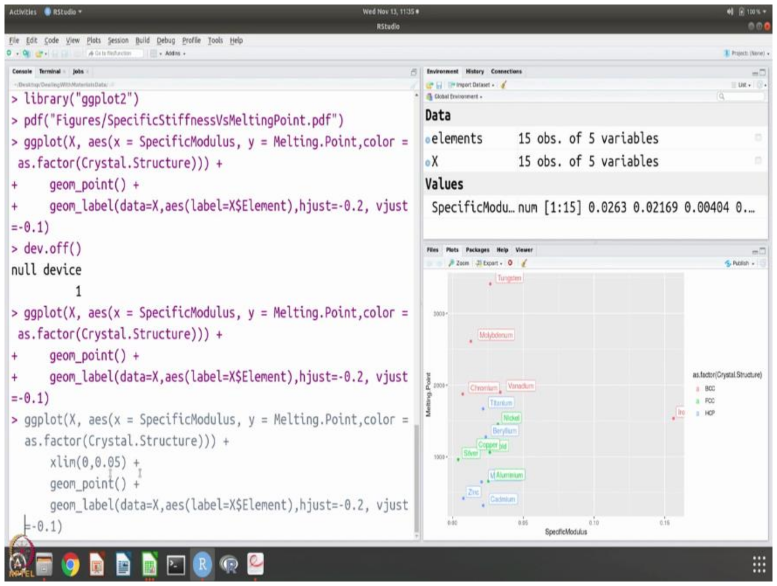
```
=-0.1)
> dev.off()
null device
          1
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
> elements <- read.csv("../Data/Elements2.csv")
  X <- elements[order(elements$Youngs.Modulus/elements$Density
  ),]
  SpecificModulus <- elements$Youngs.Modulus/elements$Density
  library("ggplot2")
  pdf("Figures/SpecificStiffnessVsMeltingPointA.pdf")
  ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
  as.factor(Crystal.Structure))) +
      xlim(0,0.05) +
      geom_point() +
      geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
  =-0.1)
```

```
Data
elements      15 obs. of 5 variables
X             15 obs. of 5 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```



```
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
> dev.off()
null device
          1
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
>
  pdf("Figures/SpecificStiffnessVsMeltingPointA.pdf")
  ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
  as.factor(Crystal.Structure))) +
      xlim(0,0.05) +
      geom_point() +
      geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
  =-0.1)
```

```
Data
elements      15 obs. of 5 variables
X             15 obs. of 5 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```

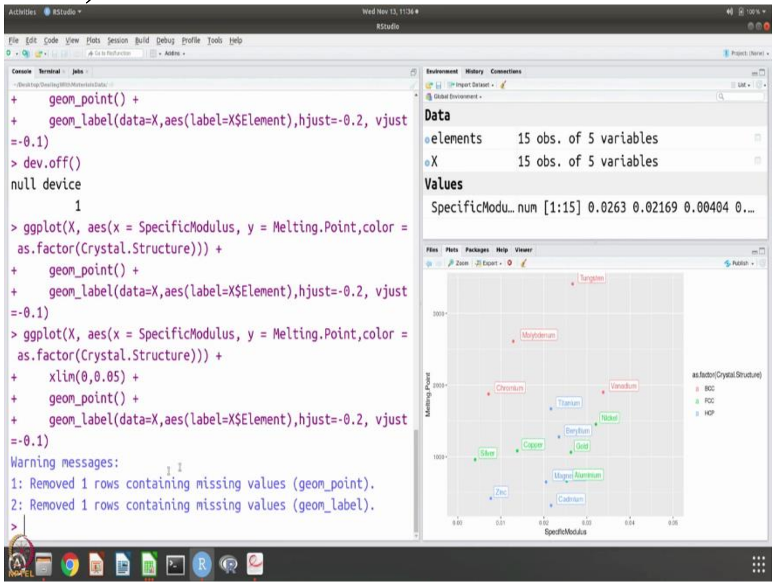


```
> library("ggplot2")
> pdf("Figures/SpecificStiffnessVsMeltingPoint.pdf")
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
> dev.off()
null device
          1
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
  as.factor(Crystal.Structure))) +
      xlim(0,0.05) +
      geom_point() +
      geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
  =-0.1)
```

```
Data
elements      15 obs. of 5 variables
X             15 obs. of 5 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```

So, probably we do not need up to this. So we have and we also do not need this. Let us start with ggplot. And so what we want to do is that specific modulus, this is melting point according to color, but now we want to restrict ourselves to 0.05. We do not want to go beyond this. And geometry is point and labeling is according to element. So, everything else is the same as earlier as here, except that we have introduced a new limit for the X range.

(Refer Slide Time: 17:13)



```
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
> dev.off()
null device
          1
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
> ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color =
 as.factor(Crystal.Structure))) +
+     xlim(0,0.05) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust
=-0.1)
Warning messages:
1: Removed 1 rows containing missing values (geom_point).
2: Removed 1 rows containing missing values (geom_label).
>
```

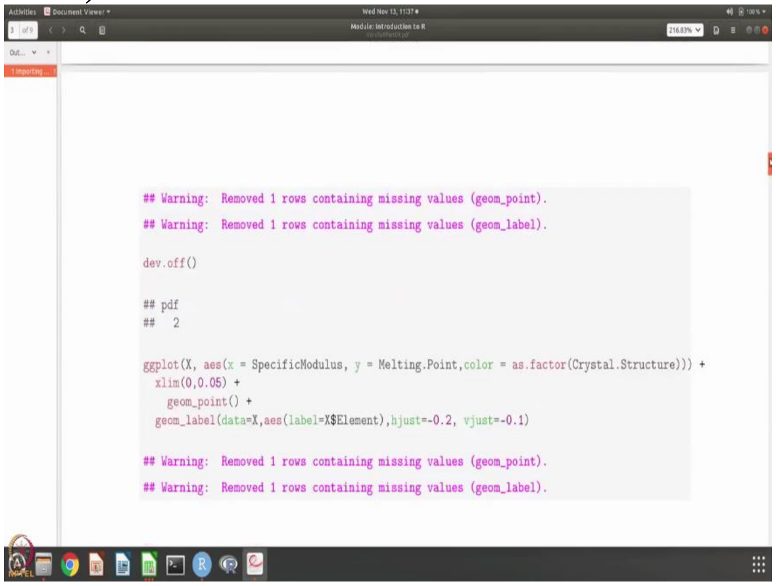
```
Data
elements      15 obs. of 5 variables
X             15 obs. of 5 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```

So, when you do that, it gives you a warning message. It says removed 1 rows containing missing values and removed 1 rows containing missing values for both the point and label. That is because it had the iron point and it had the label iron. And R is just telling you that when you did this rescaling of the X axis, you actually missed one data point. And this is very useful because sometimes if we do not pay attention and if we just give some X limit or Y limit, in the process, if you leave out some data, it is better to know that we have left out some data and that is what R is giving us a warning.

So in this case, we know we wanted that data to be left out so it is okay for us. But in some cases, you should not inadvertently leave out to data points. And so it is always a good idea to read the warning messages and to know. Now, because we have expanded the X axis in the relevant regime, you can see that the data has spread out.

And previously we thought that all these points were at the same level, but now you can see that cadmium and tungsten are actually slightly apart. Probably tungsten and gold are at the same level. And so this kind of gives you a better picture and a more closer look at the data and ggplot just by adding one more layer allows you to do that.

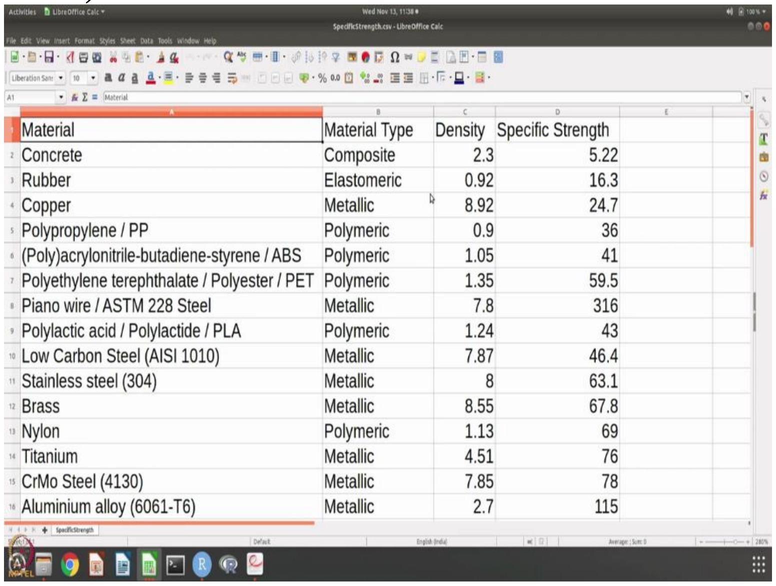
(Refer Slide Time: 18:43)



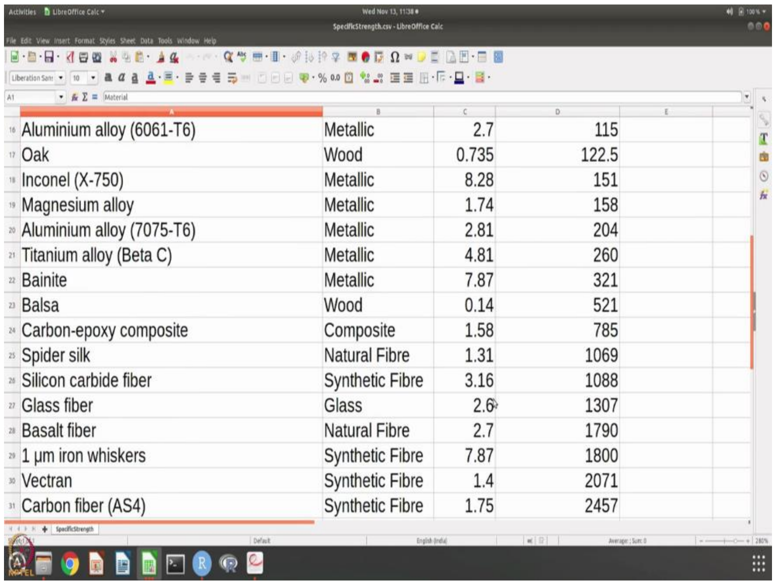
```
## Warning:  Removed 1 rows containing missing values (geom_point).
## Warning:  Removed 1 rows containing missing values (geom_label).

dev.off()

## pdf
##   2

ggplot(X, aes(x = SpecificModulus, y = Melting.Point,color = as.factor(Crystal.Structure))) +
  xlim(0,0.05) +
    geom_point() +
  geom_label(data=X,aes(label=X$Element),hjust=-0.2, vjust=-0.1)

## Warning:  Removed 1 rows containing missing values (geom_point).
## Warning:  Removed 1 rows containing missing values (geom_label).
```

So that is what we have done and this is what is shown here also. So, now, we have done.

(Refer Slide Time: 18:58)



Material	Material Type	Density	Specific Strength
Concrete	Composite	2.3	5.22
Rubber	Elastomeric	0.92	16.3
Copper	Metallic	8.92	24.7
Polypropylene / PP	Polymeric	0.9	36
(Poly)acrylonitrile-butadiene-styrene / ABS	Polymeric	1.05	41
Polyethylene terephthalate / Polyester / PET	Polymeric	1.35	59.5
Piano wire / ASTM 228 Steel	Metallic	7.8	316
Polylactic acid / Polylactide / PLA	Polymeric	1.24	43
Low Carbon Steel (AISI 1010)	Metallic	7.87	46.4
Stainless steel (304)	Metallic	8	63.1
Brass	Metallic	8.55	67.8
Nylon	Polymeric	1.13	69
Titanium	Metallic	4.51	76
CrMo Steel (4130)	Metallic	7.85	78
Aluminium alloy (6061-T6)	Metallic	2.7	115


SpecificStrength.csv - LibreOffice Calc

	A	B	C	D
16	Aluminium alloy (6061-T6)	Metallic	2.7	115
17	Oak	Wood	0.735	122.5
18	Inconel (X-750)	Metallic	8.28	151
19	Magnesium alloy	Metallic	1.74	158
20	Aluminium alloy (7075-T6)	Metallic	2.81	204
21	Titanium alloy (Beta C)	Metallic	4.81	260
22	Bainite	Metallic	7.87	321
23	Balsa	Wood	0.14	521
24	Carbon-epoxy composite	Composite	1.58	785
25	Spider silk	Natural Fibre	1.31	1069
26	Silicon carbide fiber	Synthetic Fibre	3.16	1088
27	Glass fiber	Glass	2.6	1307
28	Basalt fiber	Natural Fibre	2.7	1790
29	1 μm iron whiskers	Synthetic Fibre	7.87	1800
30	Vectran	Synthetic Fibre	1.4	2071
31	Carbon fiber (AS4)	Synthetic Fibre	1.75	2457

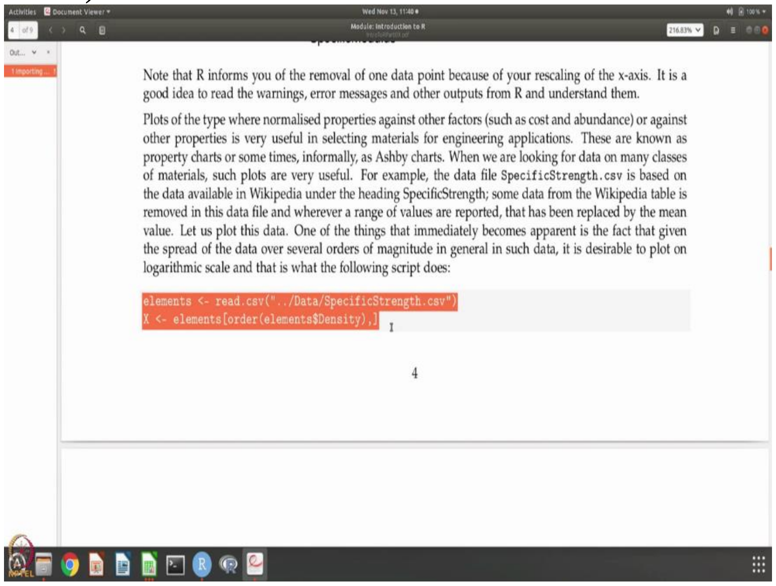
Now, let us take another data. It is called specific strength and here is the data. So, this data file I have generated by reading the data table that is given in one of the Wikipedia pages about specific strength and it has the material like concrete, rubber, copper, polypropylene etc. As you can see they are of different material types, this is a composite, this is an elastomeric, this is a metallic material, this is a polymeric material and some of them are fibrous, some of them are synthetic fibers and so on.

So, for example, spider silk is a natural fiber, silicon carbide fiber is a synthetic fiber and glass is a glass material and so on and so forth. So, and I have even put iron whiskers as synthetic fibers and so on so forth. Then there is the density of these materials and the specific strength of these materials that is strength by density. As you know, strength is the resistance that a material gives to permanent deformation or plastic deformation

So, we want to know the specific strength, how do these materials perform. And as you can see the numbers change a lot, I mean, you have something that is going in thousands, then you have numbers in hundreds, then you have numbers in tens. And then you might even have numbers in ones. So, there is a large range. And by the way, in these cases, when the numbers were given in a range for some of these materials, I have just taken the average to be the value just to make life easy for us when we are doing the plotting.

And so here also you can see the numbers can change from point something to some large number like 8 point something. So, there is, two orders of magnitude here, here it is more, it is about three orders of magnitude. So, the numbers vary a lot. So, it is difficult to get them all on the same plot unless you do some manipulation which is to plot them on a logarithmic scale. Okay, so that is what we are going to do.

(Refer Slide Time: 21:29)

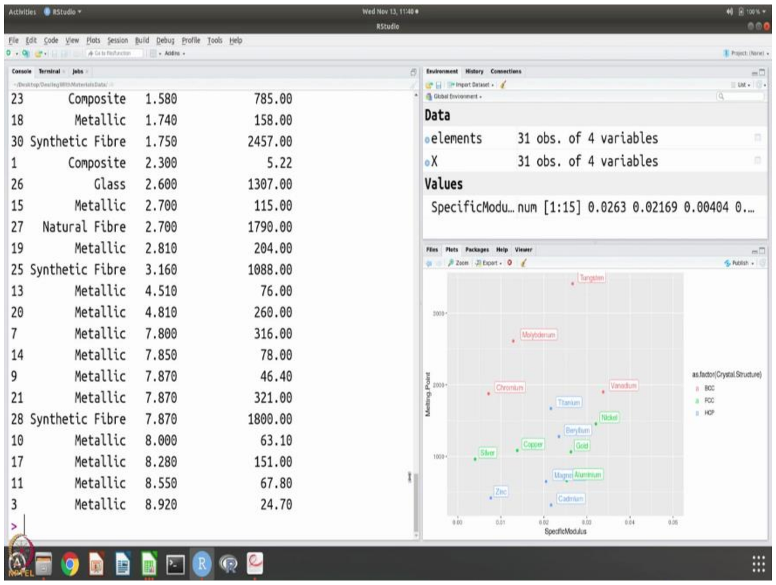


Note that R informs you of the removal of one data point because of your rescaling of the x-axis. It is a good idea to read the warnings, error messages and other outputs from R and understand them.

Plots of the type where normalised properties against other factors (such as cost and abundance) or against other properties is very useful in selecting materials for engineering applications. These are known as property charts or some times, informally, as Ashby charts. When we are looking for data on many classes of materials, such plots are very useful. For example, the data file `SpecificStrength.csv` is based on the data available in Wikipedia under the heading SpecificStrength; some data from the Wikipedia table is removed in this data file and wherever a range of values are reported, that has been replaced by the mean value. Let us plot this data. One of the things that immediately becomes apparent is the fact that given the spread of the data over several orders of magnitude in general in such data, it is desirable to plot on logarithmic scale and that is what the following script does:

```
elements <- read.csv("../Data/SpecificStrength.csv")
X <- elements[order(elements$Density),]
```

4

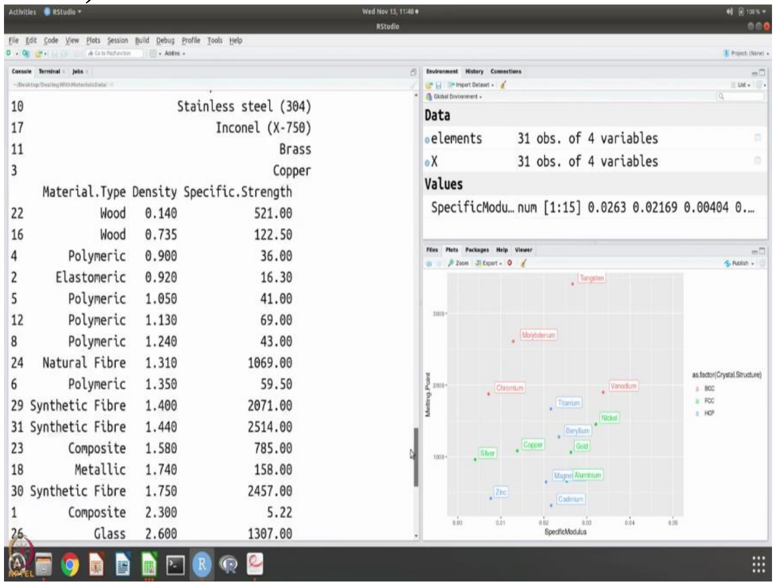


```
23       Composite  1.580         785.00
18        Metallic  1.740         158.00
30 Synthetic Fibre  1.750        2457.00
1        Composite  2.300           5.22
26           Glass  2.600        1307.00
15        Metallic  2.700         115.00
27   Natural Fibre  2.700        1790.00
19        Metallic  2.810         204.00
25 Synthetic Fibre  3.160        1088.00
13        Metallic  4.510          76.00
20        Metallic  4.810         260.00
7         Metallic  7.800         316.00
14        Metallic  7.850          78.00
9         Metallic  7.870          46.40
21        Metallic  7.870         321.00
28 Synthetic Fibre  7.870        1800.00
10        Metallic  8.000          63.10
17        Metallic  8.280         151.00
11        Metallic  8.550          67.80
3         Metallic  8.920          24.70
>
```

Data
elements   31 obs. of 4 variables
X   31 obs. of 4 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…

So, let us start, let us take this data and load them So, we are going to do the specific strength dot CSV file, we are going to load and we are going to order that and store it as X. And now if you say X here, so it has all this data.

(Refer Slide Time: 21:53)

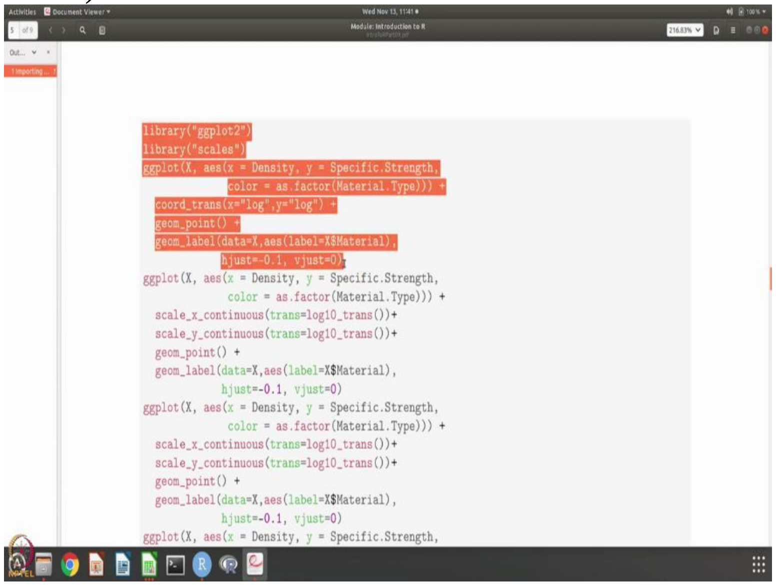


```
10                     Stainless steel (304)
17                          Inconel (X-750)
11                                    Brass
3                                    Copper
     Material.Type Density Specific.Strength
22            Wood   0.140            521.00
16            Wood   0.735            122.50
4        Polymeric   0.900             36.00
2      Elastomeric   0.920             16.30
5        Polymeric   1.050             41.00
12       Polymeric   1.130             69.00
8        Polymeric   1.240             43.00
24   Natural Fibre   1.310           1069.00
6        Polymeric   1.350             59.50
29 Synthetic Fibre   1.400           2071.00
31 Synthetic Fibre   1.440           2514.00
23       Composite   1.580            785.00
18        Metallic   1.740            158.00
30 Synthetic Fibre   1.750           2457.00
1        Composite   2.300              5.22
26           Glass   2.600           1307.00
```

```
Data
elements      31 obs. of 4 variables
X             31 obs. of 4 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```

So it has the material type, density and specific strength.

(Refer Slide Time: 22:01)

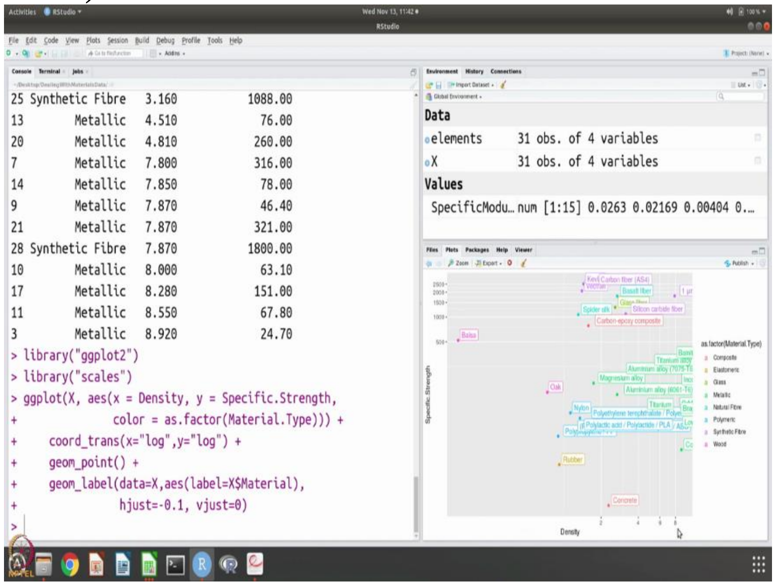


```
library("ggplot2")
library("scales")
ggplot(X, aes(x = Density, y = Specific.Strength,
              color = as.factor(Material.Type))) +
  coord_trans(x="log",y="log") +
  geom_point() +
  geom_label(data=X,aes(label=X$Material),
             hjust=-0.1, vjust=0)
ggplot(X, aes(x = Density, y = Specific.Strength,
              color = as.factor(Material.Type))) +
  scale_x_continuous(trans=log10_trans())+
  scale_y_continuous(trans=log10_trans())+
  geom_point() +
  geom_label(data=X,aes(label=X$Material),
             hjust=-0.1, vjust=0)
ggplot(X, aes(x = Density, y = Specific.Strength,
              color = as.factor(Material.Type))) +
  scale_x_continuous(trans=log10_trans())+
  scale_y_continuous(trans=log10_trans())+
  geom_point() +
  geom_label(data=X,aes(label=X$Material),
             hjust=-0.1, vjust=0)
ggplot(X, aes(x = Density, y = Specific.Strength,
```

Now, we want to use ggplot and we are also going to use the library scales because we want to change the X and Y axis to logarithmic scale. So, let us do this and let us plot and see what happens. So, ggplot and we are using library scales as usual ggplot you have to tell what is the data, what is the aesthetics, it is density or specific strength we want to do. And now we are going to color according to material type. Metallic materials in one, glass in another, concrete in another and so on.

And the coordinate transformation both x and y we are taking logarithmic axis. So, we are plotting on logarithmic scale and the geometry is of course points and labeling has to be done according to the name of the material. So, from the data it will take the name of the material and do the plotting. So, let us do this.

(Refer Slide Time: 23:00)

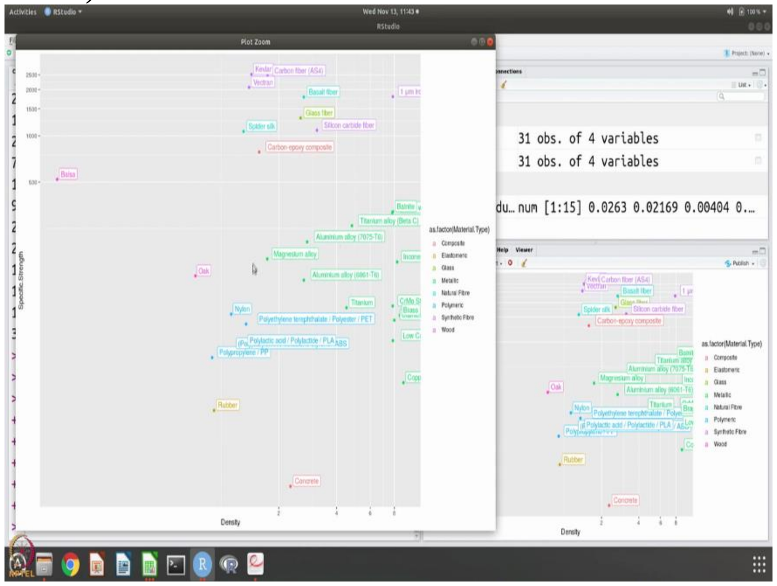


```
25 Synthetic Fibre   3.160          1088.00
13        Metallic   4.510            76.00
20        Metallic   4.810           260.00
7         Metallic   7.800           316.00
14        Metallic   7.850            78.00
9         Metallic   7.870            46.40
21        Metallic   7.870           321.00
28 Synthetic Fibre   7.870          1800.00
10        Metallic   8.000            63.10
17        Metallic   8.280           151.00
11        Metallic   8.550            67.80
3         Metallic   8.920            24.70
> library("ggplot2")
> library("scales")
> ggplot(X, aes(x = Density, y = Specific.Strength,
+               color = as.factor(Material.Type))) +
+     coord_trans(x="log",y="log") +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Material),
+                hjust=-0.1, vjust=0)
>
```

```
Data
elements      31 obs. of 4 variables
X             31 obs. of 4 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```

So, now you can see that okay, so it is a logarithmic scale obviously. So, 1, 2 and up to 10. So, here you saw that its density was not changing, some 0.7 to something, but here there was a large scale difference. And so, you can see that and according to material type the coloring is done. Composite, elastomeric, glass, metallic, etc they are all.

(Refer Slide Time: 23:33)



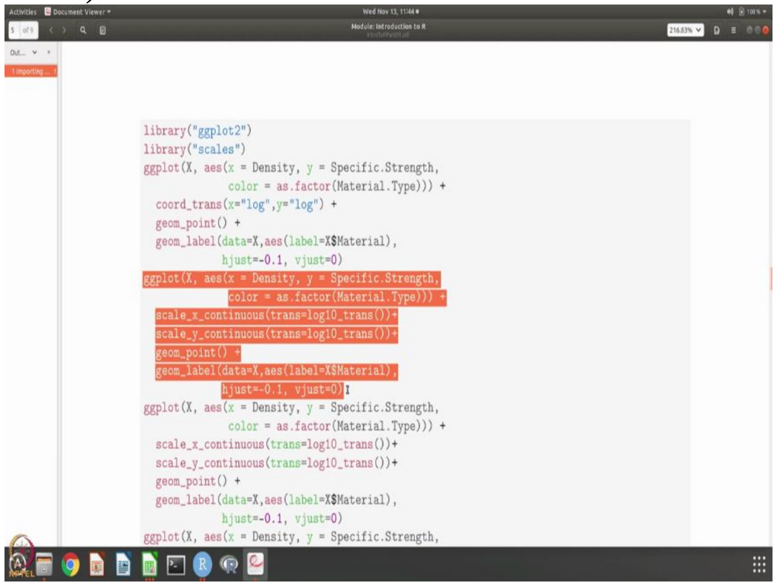
```
31 obs. of 4 variables
31 obs. of 4 variables
du… num [1:15] 0.0263 0.02169 0.00404 0.…
```

So, we can zoom and see So, you can see that there are different colors and the moment you put it in colors, you can see that all these polymers, for example, are clustering together. All this metallic materials, for example, are clustering together and all the fibers, for examples, are clustering together, I mean, they are all of different type, glass fiber, basalt fiber, silicon carbide, spider silk and iron whisker.

So, they are all have different type of materials, but in terms of their structure or geometry or morphology, they are different. And you can see that they are also clustering together. So, there is an effect of the material type, there is an effect in which form that material is, and both these come out very, very nicely using these property charts. And the other thing that comes out very, very nicely is that wood is really an outlier.

So, in terms of density, very low densities, but specific strengths comparable even more than, I mean comparable to metals or more than and in terms of density, there is a huge difference between these but they do have very good specific strength. So, this kind of information is very useful and it is also nice to have if you are trying to choose a material for a particular engineering problem, and so that is what this plot does. So yes, so we are going to do some more plots.

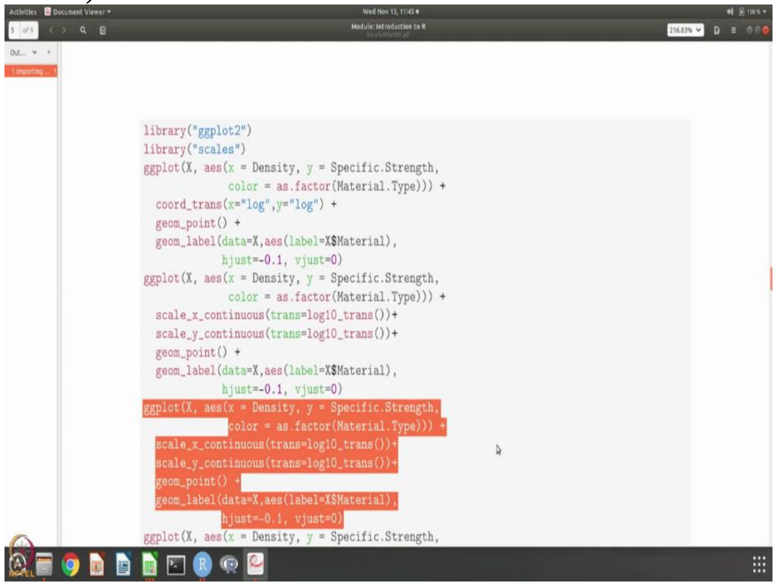
(Refer Slide Time: 25:14)



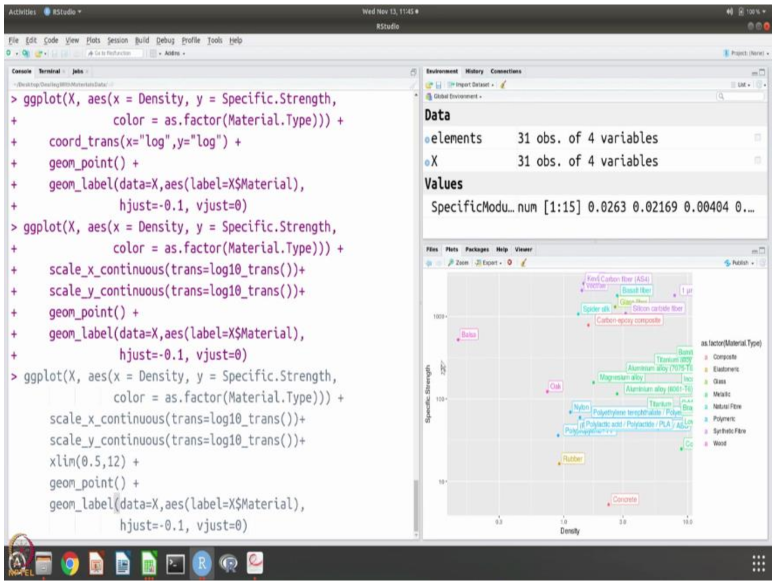
```
library("ggplot2")
library("scales")
ggplot(X, aes(x = Density, y = Specific.Strength,
           color = as.factor(Material.Type))) +
  coord_trans(x="log",y="log") +
  geom_point() +
  geom_label(data=X,aes(label=X$Material),
          hjust=-0.1, vjust=0)
ggplot(X, aes(x = Density, y = Specific.Strength,
           color = as.factor(Material.Type))) +
  scale_x_continuous(trans=log10_trans())+
  scale_y_continuous(trans=log10_trans())+
  geom_point() +
  geom_label(data=X,aes(label=X$Material),
          hjust=-0.1, vjust=0)
ggplot(X, aes(x = Density, y = Specific.Strength,
           color = as.factor(Material.Type))) +
  scale_x_continuous(trans=log10_trans())+
  scale_y_continuous(trans=log10_trans())+
  geom_point() +
  geom_label(data=X,aes(label=X$Material),
          hjust=-0.1, vjust=0)
ggplot(X, aes(x = Density, y = Specific.Strength,
```

So I want to, that is a log plot, now I am going to make it logarithm to the base 10. So it is possible to do that. So, everything is the same except that the transformation is to logarithm to the base 10. So, you can do and you can get the data. So, you can see now it is 10, 100, 1000, etc. So, it is on the logarithm to the base 10. So, we can do the next exercise, which is the same.

(Refer Slide Time: 26:04)

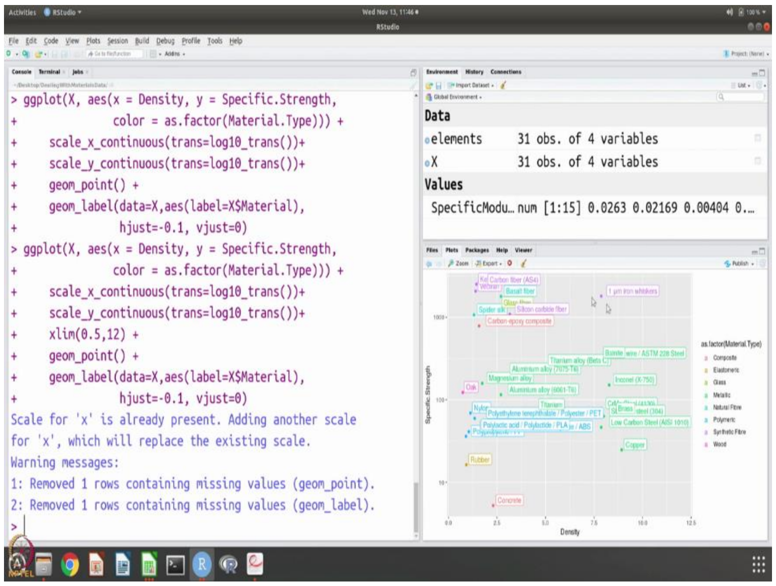


```
library("ggplot2")
library("scales")
ggplot(X, aes(x = Density, y = Specific.Strength,
           color = as.factor(Material.Type))) +
  coord_trans(x="log",y="log") +
  geom_point() +
  geom_label(data=X,aes(label=X$Material),
          hjust=-0.1, vjust=0)
ggplot(X, aes(x = Density, y = Specific.Strength,
           color = as.factor(Material.Type))) +
  scale_x_continuous(trans=log10_trans())+
  scale_y_continuous(trans=log10_trans())+
  geom_point() +
  geom_label(data=X,aes(label=X$Material),
          hjust=-0.1, vjust=0)
ggplot(X, aes(x = Density, y = Specific.Strength,
           color = as.factor(Material.Type))) +
  scale_x_continuous(trans=log10_trans())+
  scale_y_continuous(trans=log10_trans())+
  geom_point() +
  geom_label(data=X,aes(label=X$Material),
          hjust=-0.1, vjust=0)
ggplot(X, aes(x = Density, y = Specific.Strength,
```



```
> ggplot(X, aes(x = Density, y = Specific.Strength,
+               color = as.factor(Material.Type))) +
+     coord_trans(x="log",y="log") +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Material),
+                hjust=-0.1, vjust=0)
> ggplot(X, aes(x = Density, y = Specific.Strength,
+               color = as.factor(Material.Type))) +
+     scale_x_continuous(trans=log10_trans())+
+     scale_y_continuous(trans=log10_trans())+
+     geom_point() +
+     geom_label(data=X,aes(label=X$Material),
+                hjust=-0.1, vjust=0)
> ggplot(X, aes(x = Density, y = Specific.Strength,
                color = as.factor(Material.Type))) +
    scale_x_continuous(trans=log10_trans())+
    scale_y_continuous(trans=log10_trans())+
    xlim(0.5,12) +
    geom_point() +
    geom_label(data=X,aes(label=X$Material),
               hjust=-0.1, vjust=0)
```

```
Data
elements     31 obs. of 4 variables
X            31 obs. of 4 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```



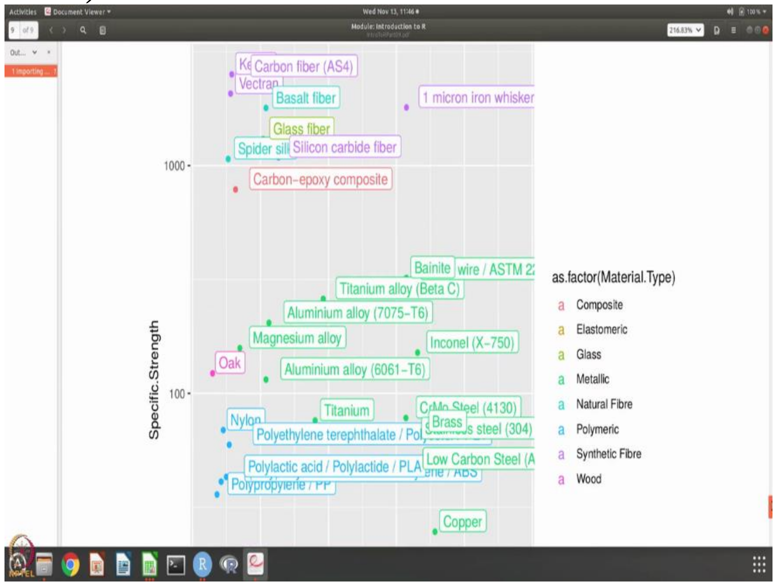
```
> ggplot(X, aes(x = Density, y = Specific.Strength,
+               color = as.factor(Material.Type))) +
+     scale_x_continuous(trans=log10_trans())+
+     scale_y_continuous(trans=log10_trans())+
+     geom_point() +
+     geom_label(data=X,aes(label=X$Material),
+                hjust=-0.1, vjust=0)
> ggplot(X, aes(x = Density, y = Specific.Strength,
+               color = as.factor(Material.Type))) +
+     scale_x_continuous(trans=log10_trans())+
+     scale_y_continuous(trans=log10_trans())+
+     xlim(0.5,12) +
+     geom_point() +
+     geom_label(data=X,aes(label=X$Material),
+                hjust=-0.1, vjust=0)
Scale for 'x' is already present. Adding another scale
for 'x', which will replace the existing scale.
Warning messages:
1: Removed 1 rows containing missing values (geom_point).
2: Removed 1 rows containing missing values (geom_label).
>
```

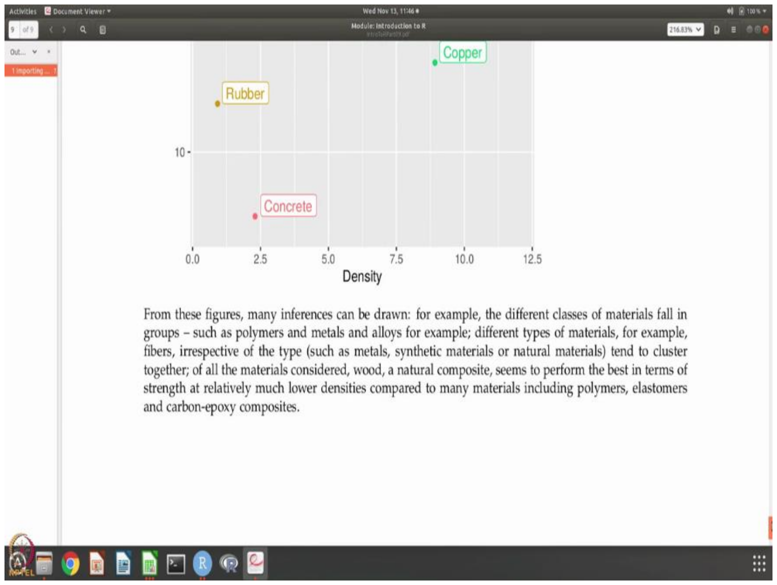
```
Data
elements     31 obs. of 4 variables
X            31 obs. of 4 variables
Values
SpecificModu… num [1:15] 0.0263 0.02169 0.00404 0.…
```

So, once we have this, let us do this and let us change the X limit. So, why do we want to do? Like we did earlier so we want to remove some of these outliers and take a closer look at the other data. Because of this balsa we are having this range and there is nothing much that is happening here. So, let us remove that and look at the closer data and again, it tells you that you know, you have already missing a point which we already know.

And the scale is already present. Of course, obviously it was already present. So, we do not have to worry about it, but it is saying, telling you that X was already present. So, it is adding another scale for the X according to what we get. So, now you can see that things are now spread out a little bit. And you can see again the greens clustering together, the blues clustering together and so on and so forth. So, it is really nice to have this kind of property charts.

(Refer Slide Time: 27:32)





From these figures, many inferences can be drawn: for example, the different classes of materials fall in groups – such as polymers and metals and alloys for example; different types of materials, for example, fibers, irrespective of the type (such as metals, synthetic materials or natural materials) tend to cluster together; of all the materials considered, wood, a natural composite, seems to perform the best in terms of strength at relatively much lower densities compared to many materials including polymers, elastomers and carbon-epoxy composites.

So, that is the figure we have generated. And that brings us to the end of this session. So, what we have learned now is that you can read data and before plotting you can also manipulate the data and plot the manipulated data and do an analysis. So, this pretty much brings us to the end of the introduction to R session. We will have a summary session and complete this module. Thank you.