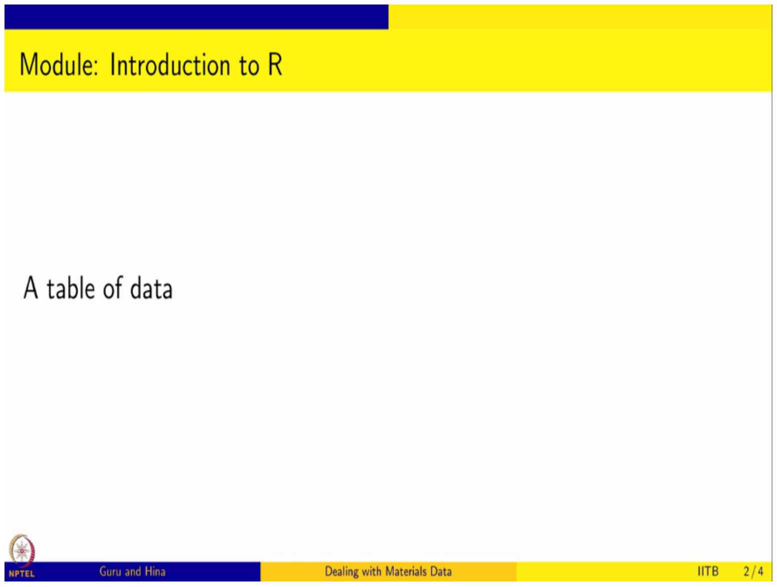
Dealing with Materials Data: Collection, Analysis and Interpretation

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

Lecture 12
Data in tabular form: Properties of elements

Welcome, this is a course on Dealing with Materials Data. We are going to talk about collection, analysis and interpretation of data and we are looking at R to do this analysis and interpretation. We have started learning R and we have seen how to use the R as a calculator and interpreter and in this session of the module, we are going to look at how to enter data into R and how to manipulate data and visualize data and so on and so forth. So, for doing that, we are going to use a table of data.

(Refer Slide Time: 0:52)



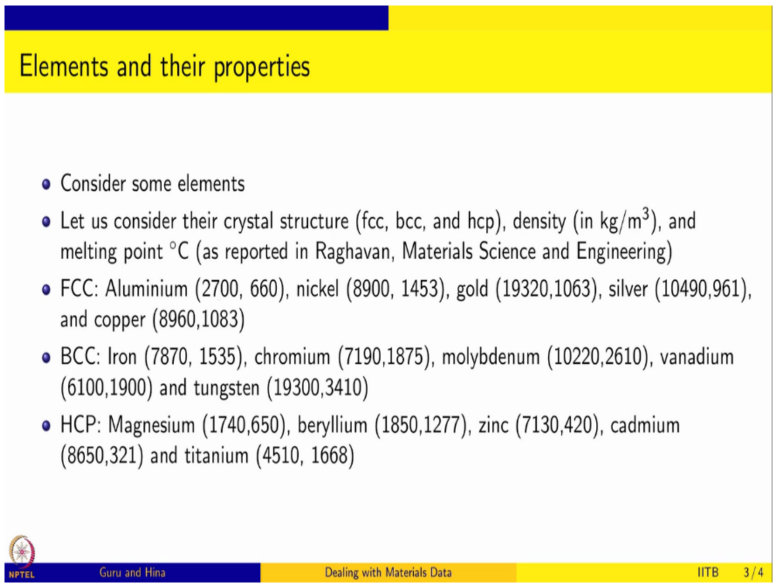
Module: Introduction to R

A table of data

Guru and Hina   Dealing with Materials Data   IITB 2/4

So, and how to introduce this table into R and how to work with it is what we are going to learn.

(Refer Slide Time: 1:01)



Elements and their properties

- Consider some elements
- Let us consider their crystal structure (fcc, bcc, and hcp), density (in kg/m^3), and melting point °C (as reported in Raghavan, Materials Science and Engineering)
- FCC: Aluminium (2700, 660), nickel (8900, 1453), gold (19320,1063), silver (10490,961), and copper (8960,1083)
- BCC: Iron (7870, 1535), chromium (7190,1875), molybdenum (10220,2610), vanadium (6100,1900) and tungsten (19300,3410)
- HCP: Magnesium (1740,650), beryllium (1850,1277), zinc (7130,420), cadmium (8650,321) and titanium (4510, 1668)

Guru and Hina | Dealing with Materials Data | IITB 3/4

And the data that we are going to use, the table of data that we are going to use is the crystal structures of some elements right? Crystal structures could be FCC, BCC or HCP. These are the only three that I am going to consider in this table. And for each we will take 5 elements. For HCC for example, we will take aluminum, nickel, gold, silver and copper. For BCC we will take iron, chromium, molybdenum, vanadium and tungsten and for HCP we will take magnesium, beryllium, zinc, cadmium and titanium.

And the density and melting point, density in kilograms per meter cube and melting point in degree C, that is reported in Raghavan, in the material science and engineering textbook, is what is given here. For example, aluminum has a density of 2700 kilogram per meter cube and melting point of 660 degree C, nickel has 8900 kilograms per meter cube and a melting point of 1453 degree C. And similarly gold has 19320 kilogram per meter cube.

It is much heavier, denser and the melting temperature is 1063 degrees C and so on. So we have this data. What we want to do is to get this data into R and so we can start work with that.

(Refer Slide Time: 2:30)

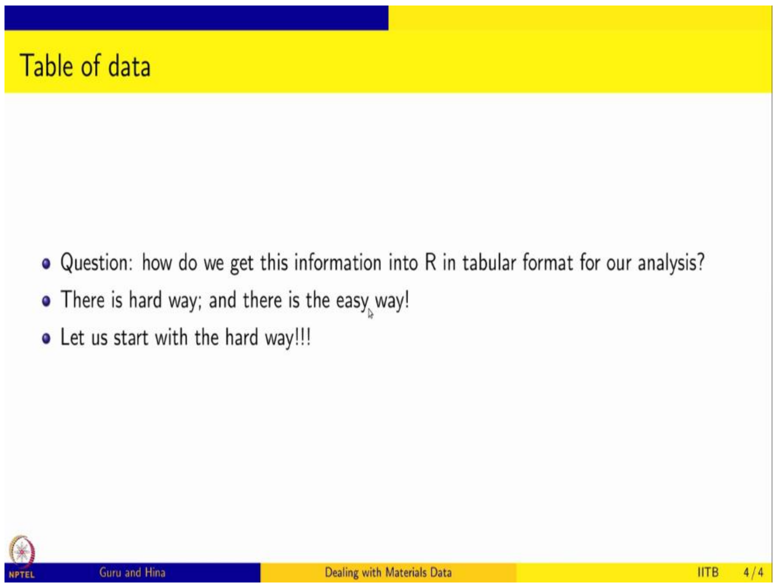


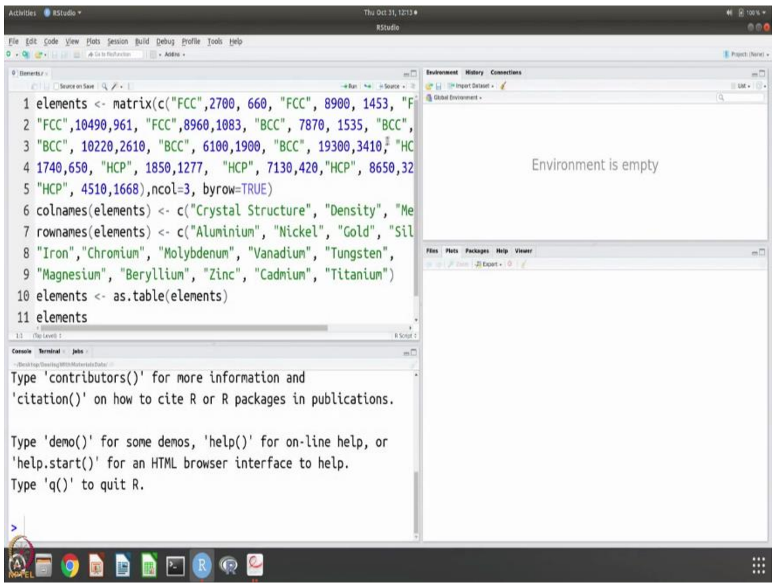
Table of data

- Question: how do we get this information into R in tabular format for our analysis?
- There is hard way; and there is the easy way!
- Let us start with the hard way!!!

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

And how to get this information into R in tabular format for further analysis is the question. There is a hard way and there is an easy way. Of course, today we are going to start with the hard way. The easy way is the one that we will do in one of the following sessions. So the hard way is, of course to enter everything into R. So let us get that.

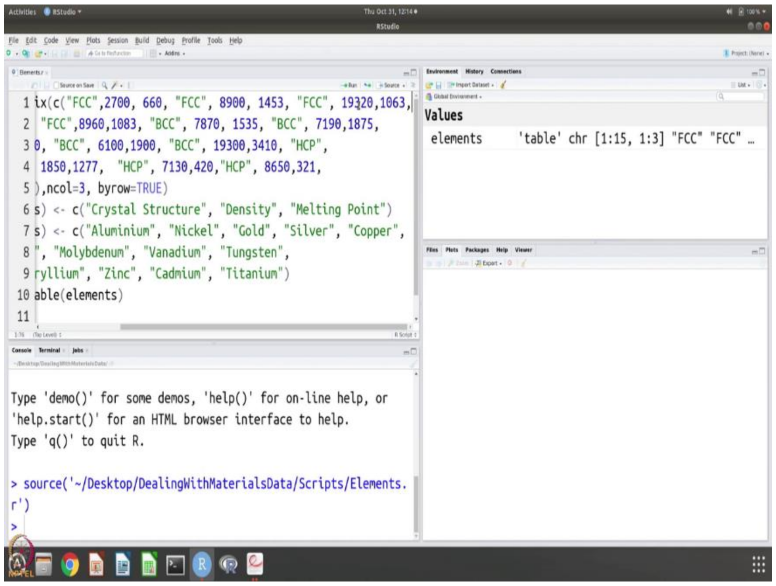
(Refer Slide Time: 2:54)



```
elements <- matrix(c("FCC",2700, 660, "FCC", 8900, 1453, "F
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

```
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.

>
```

Environment is empty



```
ix(c("FCC",2700, 660, "FCC", 8900, 1453, "FCC", 19320,1063,
 "FCC",8960,1083, "BCC", 7870, 1535, "BCC", 7190,1875,
0, "BCC", 6100,1900, "BCC", 19300,3410, "HCP",
 1850,1277,  "HCP", 7130,420,"HCP", 8650,321,
),ncol=3, byrow=TRUE)
s) <- c("Crystal Structure", "Density", "Melting Point")
s) <- c("Aluminium", "Nickel", "Gold", "Silver", "Copper",
", "Molybdenum", "Vanadium", "Tungsten",
ryllium", "Zinc", "Cadmium", "Titanium")
able(elements)
```

```
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.

> source('~/Desktop/DealingWithMaterialsData/Scripts/Elements.
r')
>
```

```
Values
elements      'table' chr [1:15, 1:3] "FCC" "FCC" …
```

And because this involves lots of data and things like that, so I have already entered this data and I have made a script. So let us go to the script. And it is called elements. So we will now work through with this. Okay, so what is it that we are doing in the script? So we can source the script. Okay. And it is sourced. So let us start line by line. So this is the line which says elements is the name that I have given and it is a matrix, right?

And this is a data for the matrix. So what does the matrix consists of? So it says, for example, its FCC 2700 660, FCC 8900, 1453. Right? So like that we have all these values listed. FCC 1932

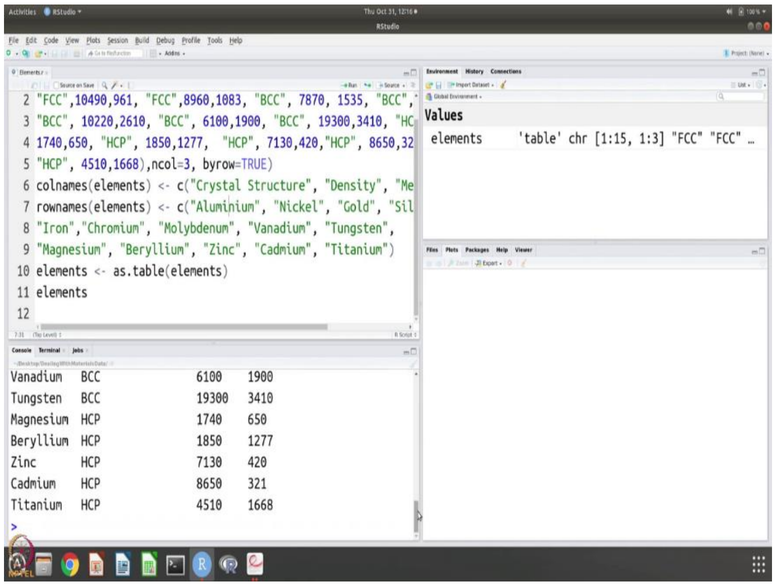
190320, 1063, FCC 10490, 961 etc. So we have and at the end after entering all these 15 elements, their density, melting temperature, we are saying that there are 3 columns and the data is entered by row, right?

We said this is, for example, aluminum, this is for nickel. So each data is like 3 columns. So these 3 makes 1 row, next 3 makes the next row and so on, right? So we said that number of columns is 3 and it is by row the data is entered. So it is R knows that this matrix is made up of 3 columns. And so every three it has to bunch and it has to make into a matrix. Columns have names, right?

Columns of these object elements have names, and what are the names? Crystal structure, density and melting point, right? Right, so that is what it is. And the rows also have names, right? There are 4 row elements and the rows also have names, for example aluminum, nickel, gold etc. So, it is also possible for example, you can make number of columns 4 and include the name here itself as aluminum FCC 2700, 660, nickel FCC 8900, 1453, etc.

And then you could have given the row names so column names as element and then crystal structure density melting point and then you do not have to name the rows at all. So, this is an exercise that (we) you are going to do. You are going to take this and modify it a little bit and then you are going to work with it. So, and we are saying that, this object elements it should be stored, this matrix that we have prepared in the name of element should be stored as a table and then if you say elements, okay.

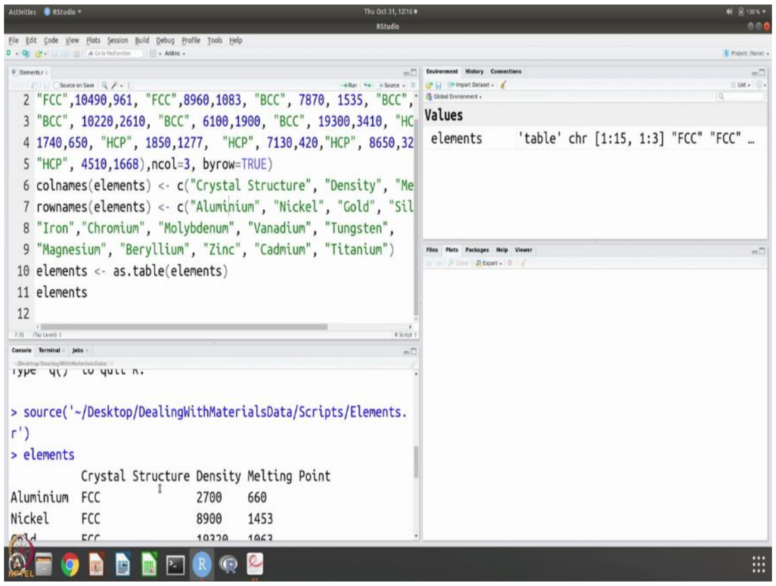
(Refer Slide Time: 6:01)



```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

```
Vanadium   BCC                6100    1900
Tungsten   BCC                19300   3410
Magnesium  HCP                1740    650
Beryllium  HCP                1850    1277
Zinc       HCP                7130    420
Cadmium    HCP                8650    321
Titanium   HCP                4510    1668
>
```

```
Values
 elements      'table' chr [1:15, 1:3] "FCC" "FCC" …
```



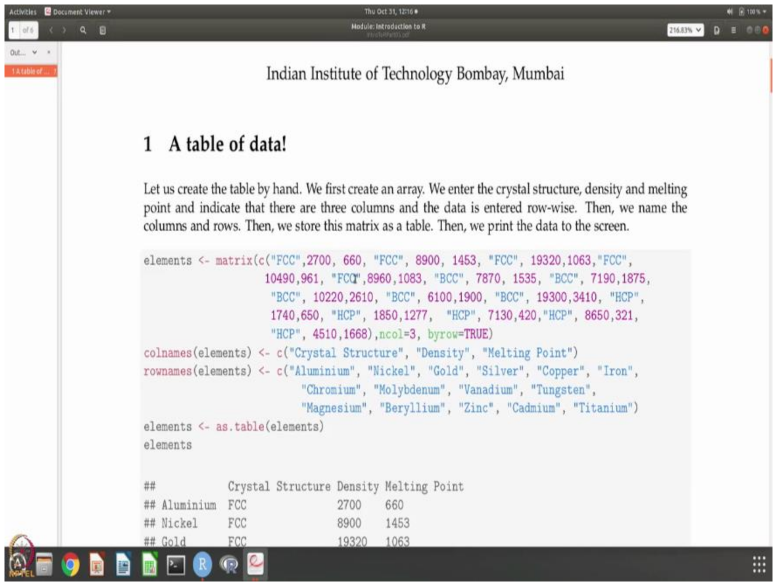
```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

```
Type 'q()' to quit R.

> source('~/Desktop/DealingWithMaterialsData/Scripts/Elements.
r')
> elements
          Crystal Structure Density Melting Point
Aluminium  FCC               2700    660
Nickel     FCC               8900    1453
Gold       FCC               19320   1063
```

```
Values
 elements      'table' chr [1:15, 1:3] "FCC" "FCC" …
```

So it gives you the list. Right? So, it says that there is crystal structure density melting point and there are all these elements-aluminum, nickel, gold, silver, copper, iron, chromium, molybdenum, vanadium, etc and their crystal structure and their density and melting point is given.

(Refer Slide Time: 6:26)

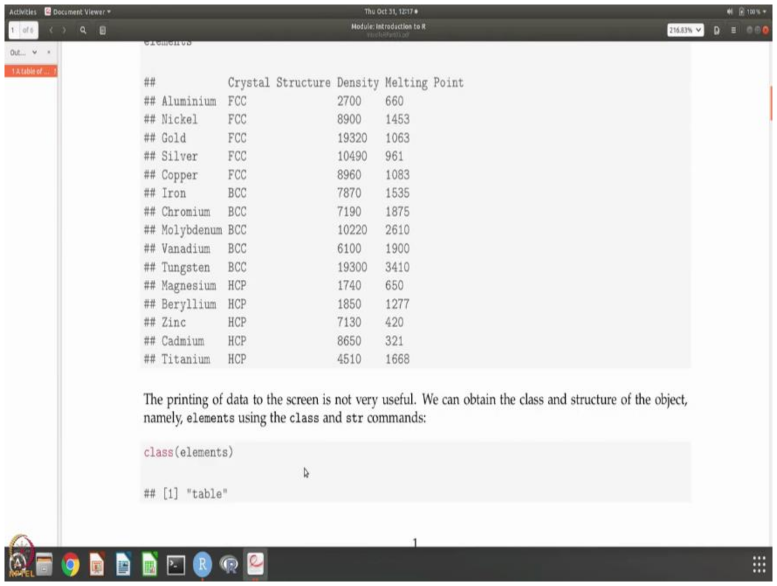


Indian Institute of Technology Bombay, Mumbai

1 A table of data!

Let us create the table by hand. We first create an array. We enter the crystal structure, density and melting point and indicate that there are three columns and the data is entered row-wise. Then, we name the columns and rows. Then, we store this matrix as a table. Then, we print the data to the screen.

```
elements <- matrix(c("FCC",2700, 660, "FCC", 8900, 1453, "FCC", 19320,1063,"FCC",
                    10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC", 7190,1875,
                     "BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HCP",
                     1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,321,
                     "HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Melting Point")
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Silver", "Copper", "Iron",
                         "Chromium", "Molybdenum", "Vanadium", "Tungsten",
                         "Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements


##           Crystal Structure Density Melting Point
## Aluminium  FCC               2700    660
## Nickel     FCC               8900    1453
## Gold       FCC               19320   1063
```



```
##           Crystal Structure Density Melting Point
## Aluminium  FCC               2700    660
## Nickel     FCC               8900    1453
## Gold       FCC               19320   1063
## Silver     FCC               10490   961
## Copper     FCC               8960    1083
## Iron       BCC               7870    1535
## Chromium   BCC               7190    1875
## Molybdenum BCC               10220   2610
## Vanadium   BCC               6100    1900
## Tungsten   BCC               19300   3410
## Magnesium  HCP               1740    650
## Beryllium  HCP               1850    1277
## Zinc       HCP               7130    420
## Cadmium    HCP               8650    321
## Titanium   HCP               4510    1668
```

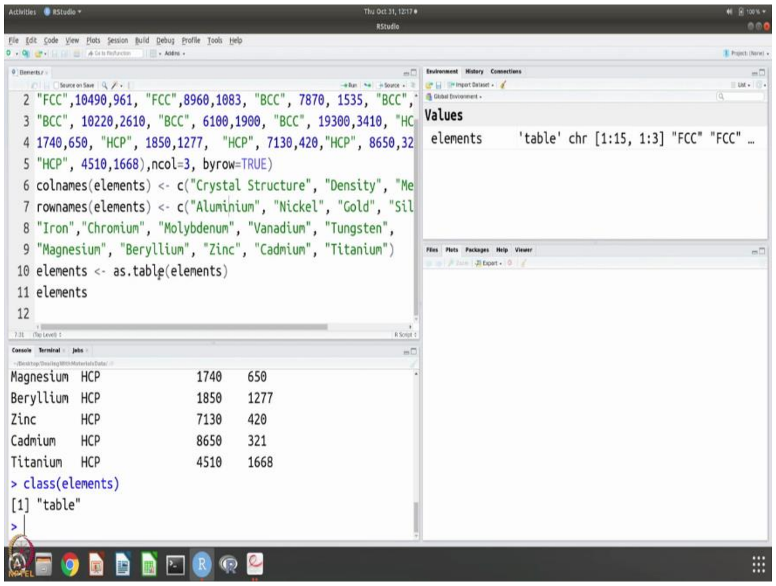
The printing of data to the screen is not very useful. We can obtain the class and structure of the object, namely, `elements` using the `class` and `str` commands:

```
class(elements)

## [1] "table"
```

1

Okay, so let us go back to the notes. So, this is what we have done. So, we have given the information by hand and we have prepared a table and this is the table that we are going to work with for the rest of this session, right?

(Refer Slide Time: 6:43)



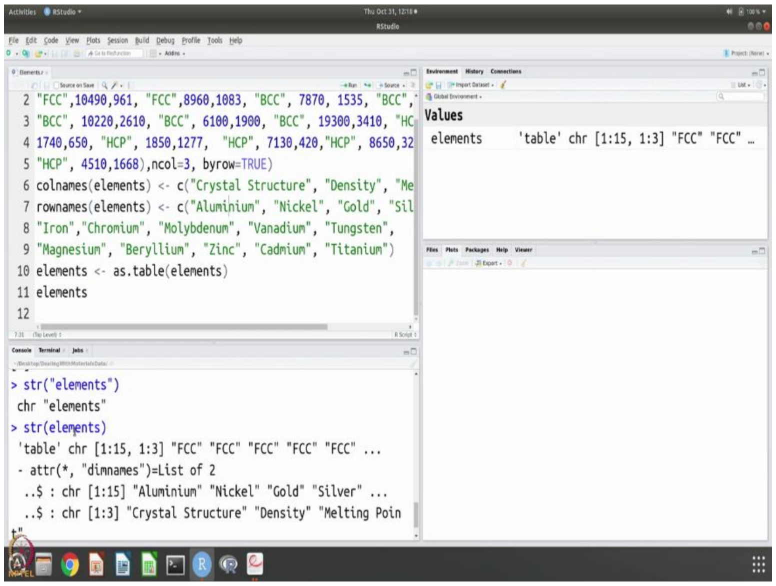
```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements

```

```
Magnesium  HCP              1740    650
Beryllium  HCP              1850    1277
Zinc       HCP              7130    420
Cadmium    HCP              8650    321
Titanium   HCP              4510    1668
> class(elements)
[1] "table"
>
```

```
Values
elements     'table' chr [1:15, 1:3] "FCC" "FCC" …
```

So, you can use this command class elements, so element, it is a table obviously, because we said that the matrix should be stored as a table here, so it is stored as a table. So there is no surprise there. Okay, then what do we do?

(Refer Slide Time: 7:09)



```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements

```

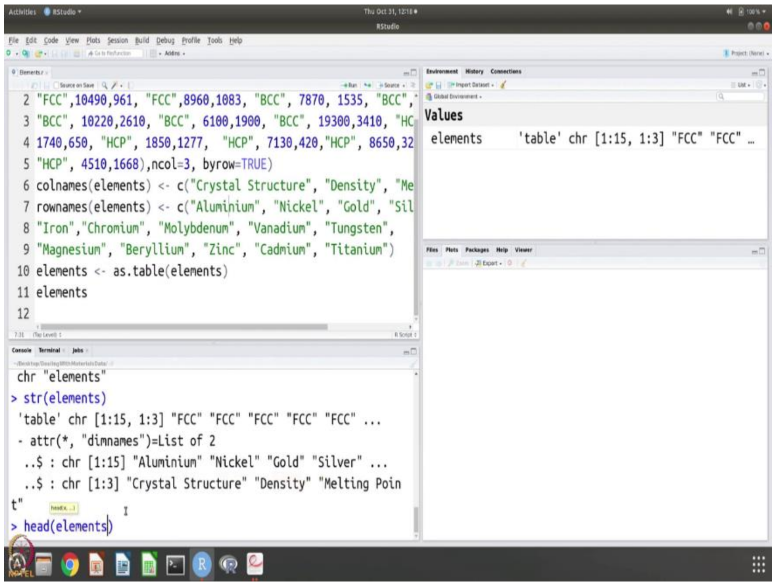
```
> str("elements")
 chr "elements"
> str(elements)
 'table' chr [1:15, 1:3] "FCC" "FCC" "FCC" "FCC" "FCC" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:15] "Aluminium" "Nickel" "Gold" "Silver" ...
  ..$ : chr [1:3] "Crystal Structure" "Density" "Melting Poin
t"
```

```
Values
elements     'table' chr [1:15, 1:3] "FCC" "FCC" …
```

Then we can say the structure of elements. Okay, so when you ask what the structure of elements says, it says it is a table and it has this 15 rows and 3 columns and so those 3 columns have information like string like FCC, etc and the numbers. And the columns and the rows are labeled and the labels are crystal structure density, melting point, aluminum, nickel, gold, silver, etc. So, this is what we have done and this is the information that it gives.

So, notice the difference when I say str() within quote elements, it just considers this as a string. So it is just some characters, but when I say str() of this table object name elements, then it gives me the information about this data. So that is what is done here. Okay.

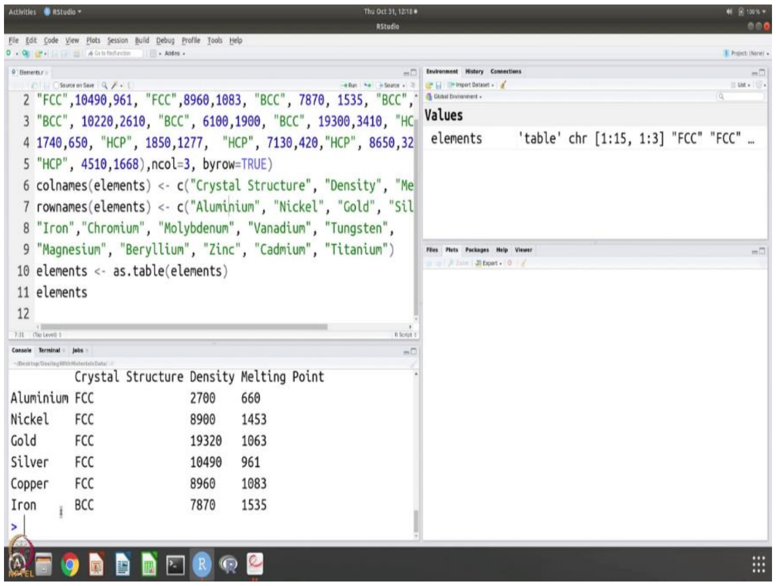
(Refer Slide Time: 8:25)



```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

```
 chr "elements"
> str(elements)
 'table' chr [1:15, 1:3] "FCC" "FCC" "FCC" "FCC" "FCC" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:15] "Aluminium" "Nickel" "Gold" "Silver" ...
  ..$ : chr [1:3] "Crystal Structure" "Density" "Melting Poin
t"
> head(elements)
```

```
elements     'table' chr [1:15, 1:3] "FCC" "FCC" …
```



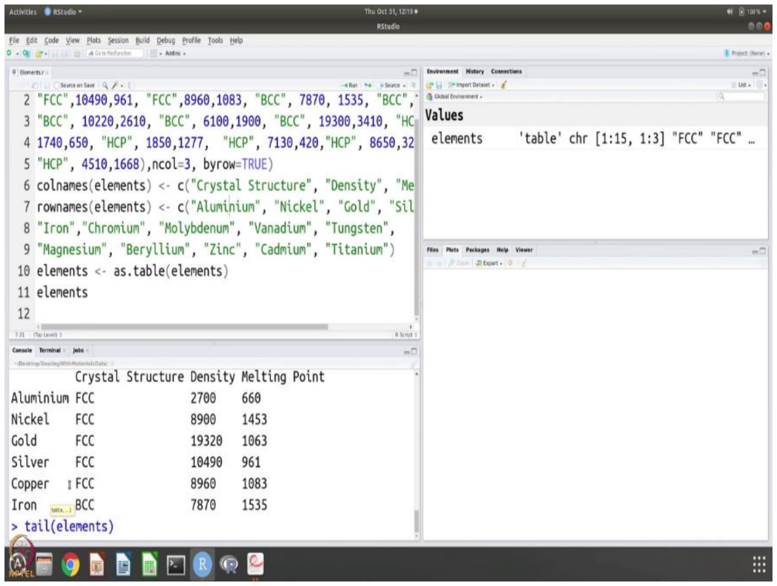
```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

```
          Crystal Structure Density Melting Point
Aluminium FCC               2700    660
Nickel    FCC               8900    1453
Gold      FCC               19320   1063
Silver    FCC               10490   961
Copper    FCC               8960    1083
Iron      BCC               7870    1535
>
```

```
elements     'table' chr [1:15, 1:3] "FCC" "FCC" …
```

Now, let us use two more commands, head elements. So, it gives you the first 6 rows of the data.

(Refer Slide Time: 08:36)

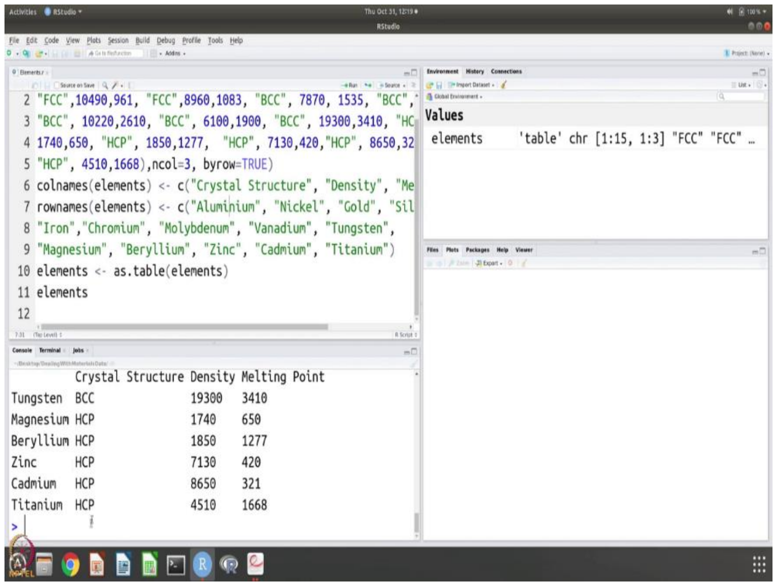


```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

Values

elements   'table' chr [1:15, 1:3] "FCC" "FCC" …

```
         Crystal Structure Density Melting Point
Aluminium FCC               2700    660
Nickel    FCC               8900    1453
Gold      FCC               19320   1063
Silver    FCC               10490   961
Copper    FCC               8960    1083
Iron      BCC               7870    1535
> tail(elements)
```



```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

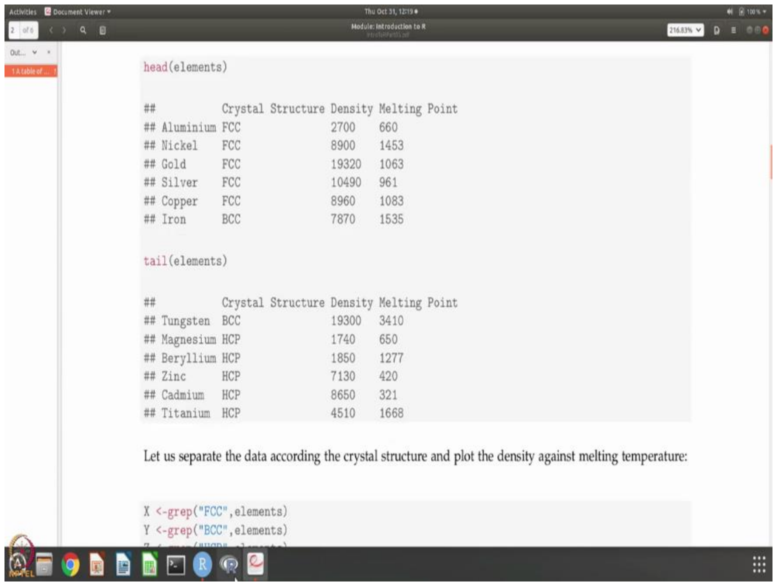
Values

elements   'table' chr [1:15, 1:3] "FCC" "FCC" …

```
         Crystal Structure Density Melting Point
Tungsten  BCC               19300   3410
Magnesium HCP               1740    650
Beryllium HCP               1850    1277
Zinc      HCP               7130    420
Cadmium   HCP               8650    321
Titanium  HCP               4510    1668
>
```

Of course, you can also use tail and it gives you the last 6 rows. So remember it was FCC, BCC, HCP. So the last 5 are HCP and the one before that was BCC. So you get all this information. So you can look at some aspects of the data using head, tail, etc these commands. So, okay so we can do that.

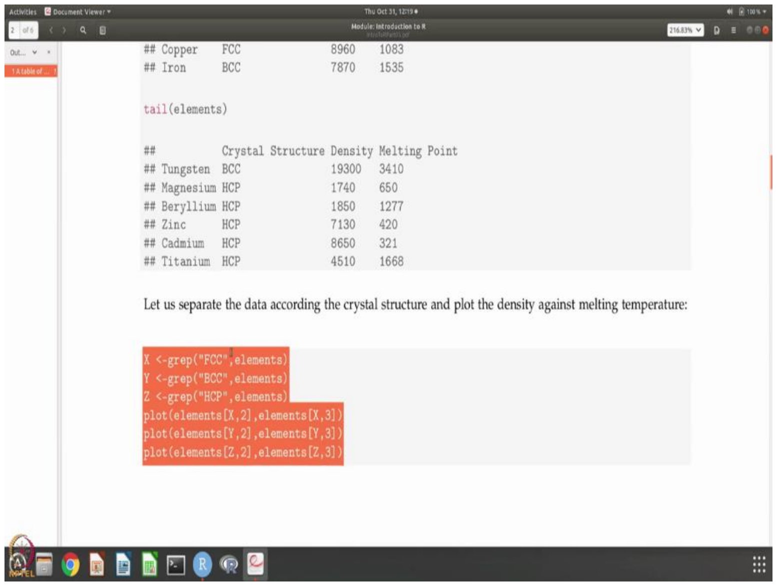
(Refer Slide Time: 9:07)



```
head(elements)

##          Crystal Structure Density Melting Point
## Aluminium FCC               2700    660
## Nickel    FCC               8900    1453
## Gold      FCC               19320   1063
## Silver    FCC               10490   961
## Copper    FCC               8960    1083
## Iron      BCC               7870    1535

tail(elements)

##          Crystal Structure Density Melting Point
## Tungsten  BCC               19300   3410
## Magnesium HCP               1740    650
## Beryllium HCP               1850    1277
## Zinc      HCP               7130    420
## Cadmium   HCP               8650    321
## Titanium  HCP               4510    1668
```

Let us separate the data according the crystal structure and plot the density against melting temperature:

```
X <-grep("FCC",elements)
Y <-grep("BCC",elements)
```



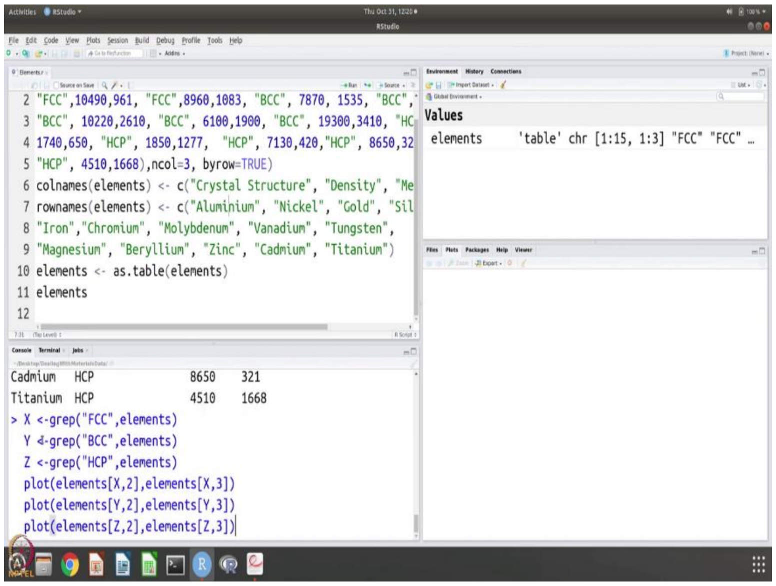
```
## Copper    FCC               8960    1083
## Iron      BCC               7870    1535

tail(elements)

##          Crystal Structure Density Melting Point
## Tungsten  BCC               19300   3410
## Magnesium HCP               1740    650
## Beryllium HCP               1850    1277
## Zinc      HCP               7130    420
## Cadmium   HCP               8650    321
## Titanium  HCP               4510    1668
```

Let us separate the data according the crystal structure and plot the density against melting temperature:

```
X <-grep("FCC",elements)
Y <-grep("BCC",elements)
Z <-grep("HCP",elements)
plot(elements[X,2],elements[X,3])
plot(elements[Y,2],elements[Y,3])
plot(elements[Z,2],elements[Z,3])
```

Now what do we want to do? We want to separate the data according to crystal structure and plot the density against the melting temperature. Okay, so this is the command that does that, okay.

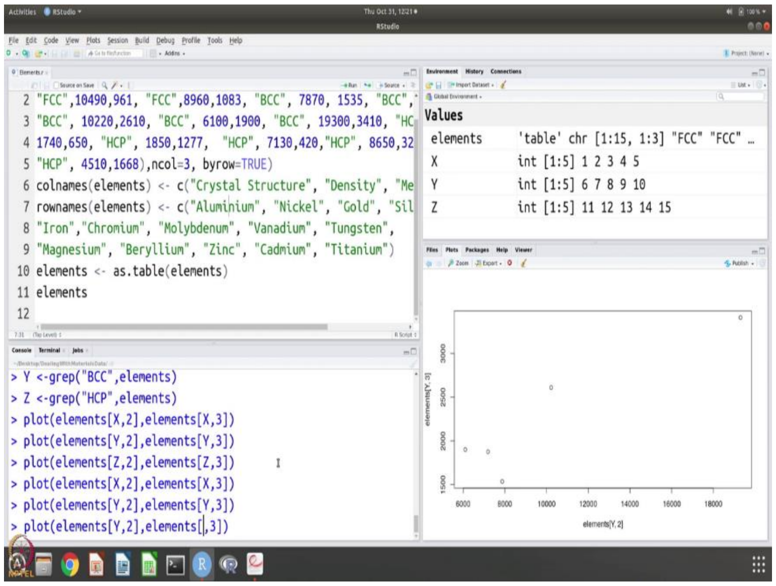
(Refer Slide Time: 09:32)



```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

```
Cadmium   HCP               8650    321
Titanium  HCP               4510    1668
> X <-grep("FCC",elements)
  Y <-grep("BCC",elements)
  Z <-grep("HCP",elements)
  plot(elements[X,2],elements[X,3])
  plot(elements[Y,2],elements[Y,3])
  plot(elements[Z,2],elements[Z,3])
```

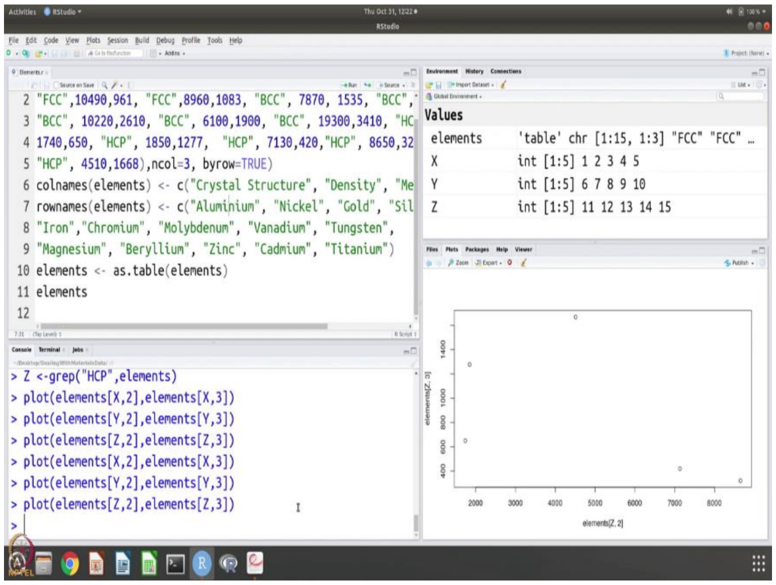
```
Values
elements    'table' chr [1:15, 1:3] "FCC" "FCC" …
```



```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

```
> Y <-grep("BCC",elements)
> Z <-grep("HCP",elements)
> plot(elements[X,2],elements[X,3])
> plot(elements[Y,2],elements[Y,3])
> plot(elements[Z,2],elements[Z,3])
> plot(elements[X,2],elements[X,3])
> plot(elements[Y,2],elements[Y,3])
> plot(elements[Y,2],elements[,3])
```

```
Values
elements    'table' chr [1:15, 1:3] "FCC" "FCC" …
X           int [1:5] 1 2 3 4 5
Y           int [1:5] 6 7 8 9 10
Z           int [1:5] 11 12 13 14 15
```



```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

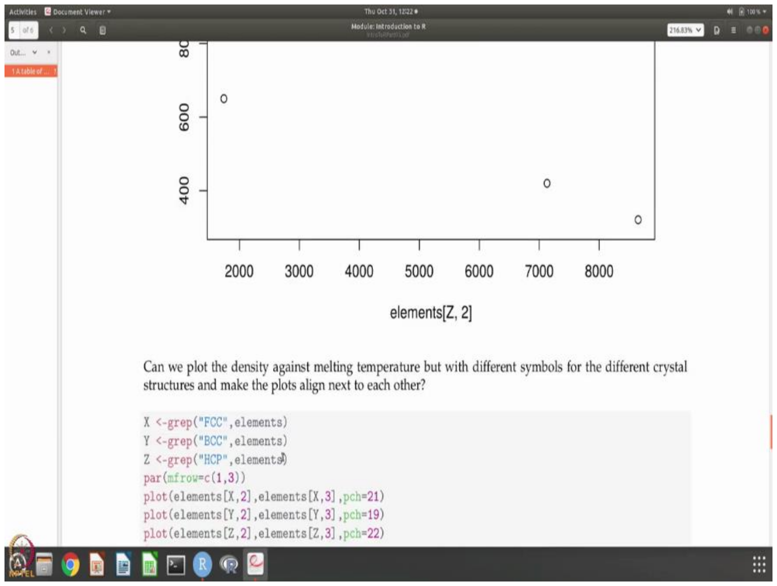
```
> Z <-grep("HCP",elements)
> plot(elements[X,2],elements[X,3])
> plot(elements[Y,2],elements[Y,3])
> plot(elements[Z,2],elements[Z,3])
> plot(elements[X,2],elements[X,3])
> plot(elements[Y,2],elements[Y,3])
> plot(elements[Z,2],elements[Z,3])
>
```

```
Values
elements    'table' chr [1:15, 1:3] "FCC" "FCC" …
X           int [1:5] 1 2 3 4 5
Y           int [1:5] 6 7 8 9 10
Z           int [1:5] 11 12 13 14 15
```

So let me copy this and put it here. Okay, so what does it do? It stores in the variables X, Y, Z etc. The rows which have in the table elements, the character FCC, or the string FCC or BCC or HCP, then it is going to plot by taking only those data, the second column of that data will be the density of that element and third column of that data will be the melting point of that element. So this is what it is going to do and let us do one by one.

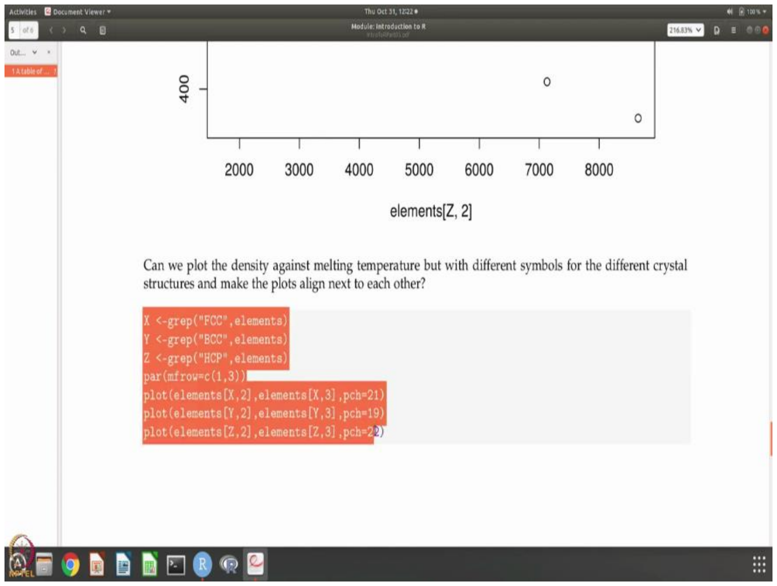
So that we still have this information Okay. Okay, so this is so let us run through one by one. Okay, so this is for FCC, do you see a trend? With density does the melting temperature increase? That is a good question to ask. We will ask and answer questions like this as we go along. Let us look at all bcc. So they also have this data. Let us take a look at HCP and they also have this data. And as you can see, in the case of HCP, it looks like lower density has higher melting point and higher density has lower melting point. I do not know if this is an actual trend, but at least for the data that we have, we can see something like that.

(Refer Slide Time: 11:58)



Can we plot the density against melting temperature but with different symbols for the different crystal structures and make the plots align next to each other?

```
X <-grep("FCC",elements)
Y <-grep("BCC",elements)
Z <-grep("HCP",elements)
par(mfrow=c(1,3))
plot(elements[X,2],elements[X,3],pch=21)
plot(elements[Y,2],elements[Y,3],pch=19)
plot(elements[Z,2],elements[Z,3],pch=22)
```

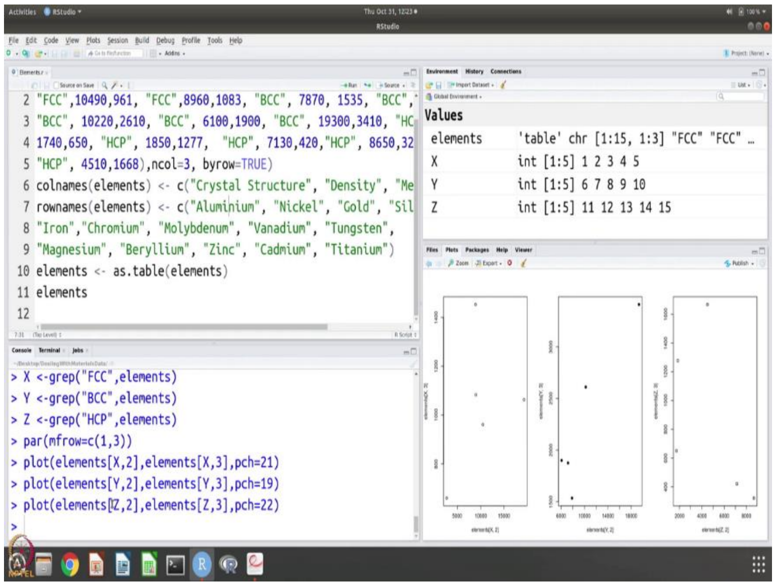


Can we plot the density against melting temperature but with different symbols for the different crystal structures and make the plots align next to each other?

```
X <-grep("FCC",elements)
Y <-grep("BCC",elements)
Z <-grep("HCP",elements)
par(mfrow=c(1,3))
plot(elements[X,2],elements[X,3],pch=21)
plot(elements[Y,2],elements[Y,3],pch=19)
plot(elements[Z,2],elements[Z,3],pch=22)
```

Of course, suppose you wanted to get all these three figures, not like this individually, that is what is shown here. There are three figures. This is what we got just now. But suppose we wanted to get them next to each other. And here is a code does that.

(Refer Slide Time: 12:09)



```
"FCC",10490,961, "FCC",8960,1083, "BCC", 7870, 1535, "BCC",
"BCC", 10220,2610, "BCC", 6100,1900, "BCC", 19300,3410, "HC
1740,650, "HCP", 1850,1277,  "HCP", 7130,420,"HCP", 8650,32
"HCP", 4510,1668),ncol=3, byrow=TRUE)
colnames(elements) <- c("Crystal Structure", "Density", "Me
rownames(elements) <- c("Aluminium", "Nickel", "Gold", "Sil
"Iron","Chromium", "Molybdenum", "Vanadium", "Tungsten",
"Magnesium", "Beryllium", "Zinc", "Cadmium", "Titanium")
elements <- as.table(elements)
elements
```

```
Values
elements     'table' chr [1:15, 1:3] "FCC" "FCC" …
X            int [1:5] 1 2 3 4 5
Y            int [1:5] 6 7 8 9 10
Z            int [1:5] 11 12 13 14 15
```

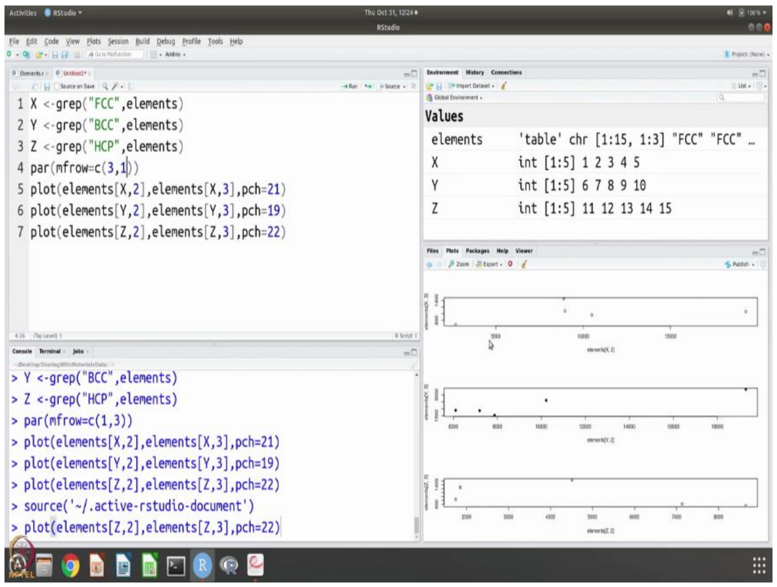
```
> X <-grep("FCC",elements)
> Y <-grep("BCC",elements)
> Z <-grep("HCP",elements)
> par(mfrow=c(1,3))
> plot(elements[X,2],elements[X,3],pch=21)
> plot(elements[Y,2],elements[Y,3],pch=19)
> plot(elements[Z,2],elements[Z,3],pch=22)
>
```

Okay, so let me copy and paste it here. So this, this command power MF row column of 1 by 3, so it is going to put all these 3 figures in a row. And so it is one row and there are 3 columns to this row. So that is what the figure is going to look like. And as you can see, so now it becomes easier to compare.

So for example, this is FCC, this BCC, this is HCP and you can see that FCC densities, the range is slightly different. So you are below 5000. Here, everything is above 6000 and here you go up to more than 18000. Of course, there was gold here which was very high and then here you have going from 2000 to 8000 right? There is nothing about 10,000 in the case of HCP and then in the case of melting temperature also, so the range is 1400. Here it is above 3000 and here it is 1600.

So, so it is easier to compare the data if you have these figures right next to each other. You can also plot the other way.

(Refer Slide Time: 13:38)



```
X <-grep("FCC",elements)
Y <-grep("BCC",elements)
Z <-grep("HCP",elements)
par(mfrow=c(3,1))
plot(elements[X,2],elements[X,3],pch=21)
plot(elements[Y,2],elements[Y,3],pch=19)
plot(elements[Z,2],elements[Z,3],pch=22)
```

```
Values
elements    'table' chr [1:15, 1:3] "FCC" "FCC" ...
X           int [1:5] 1 2 3 4 5
Y           int [1:5] 6 7 8 9 10
Z           int [1:5] 11 12 13 14 15
```

```
> Y <-grep("BCC",elements)
> Z <-grep("HCP",elements)
> par(mfrow=c(1,3))
> plot(elements[X,2],elements[X,3],pch=21)
> plot(elements[Y,2],elements[Y,3],pch=19)
> plot(elements[Z,2],elements[Z,3],pch=22)
> source('~/.active-rstudio-document')
> plot(elements[Z,2],elements[Z,3],pch=22)
```

Let me try that. I think you can, let me make another script and I want to make it. So, let us source this. Now you can see that you can also make these plots by aligning them this way, right. So, you have now 3 rows and in each row you have only one plot. And so, you can play with it and you can get many different kinds of plots and you would have seen in the demo, sometimes there were plots like this, more than one plot of different aspects in the figure and one can ask many other questions.

For example, can I get all this data in one single plot but with different symbols or different colors? You have seen in the demos that it is possible. How to do that is a question. So that we will do in one of the following sessions.

So this module is just to show you that you can take data, and you can by hand put data, of course it is very painful to enter data like this, and it is also not possible to enter data like this if you have a large amount of data, this is some 15 lines and 3-4 columns, so it is okay. But if suppose you have to have some hundreds of lines or thousands of lines or you are just getting data from an experiment, so it already comes in an electronic format.

So how to get that data into R? So import data into R is something that also we will do in one of these sessions. So but for now, this is to enter data by hand and work with the data. And like I said that there is a small exercise based on this that you can do yourself by modifying this table a little bit and working with such a table. Thank you.