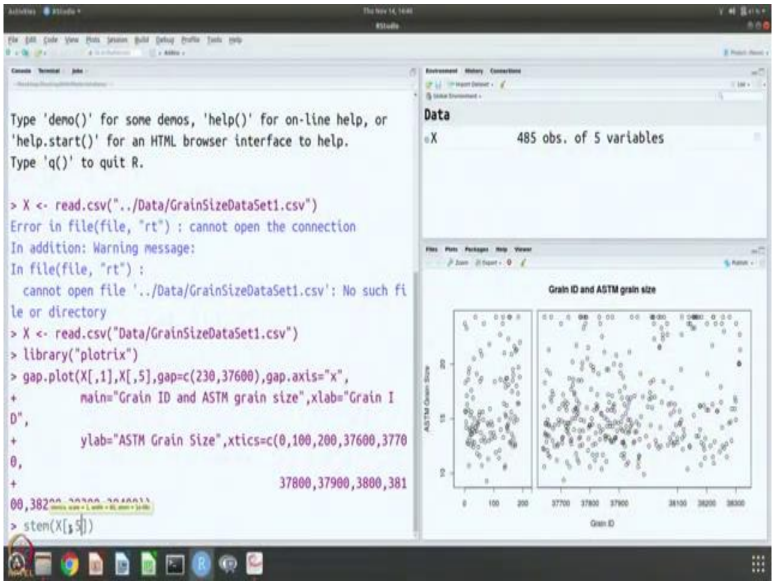
Dealing with Materials Data: Collection, Analysis and Interpretation

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

Lecture No 23

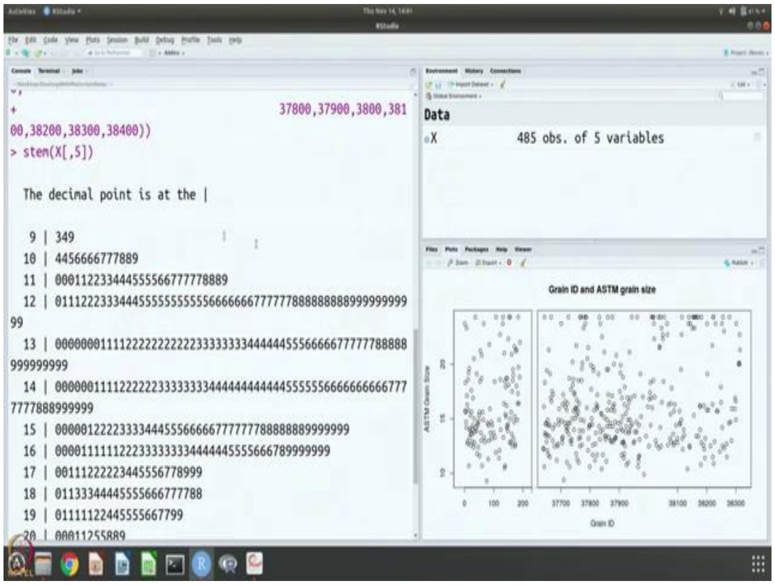
Grain size data: Property and rank based reports

(Refer Slide Time: 0:30)



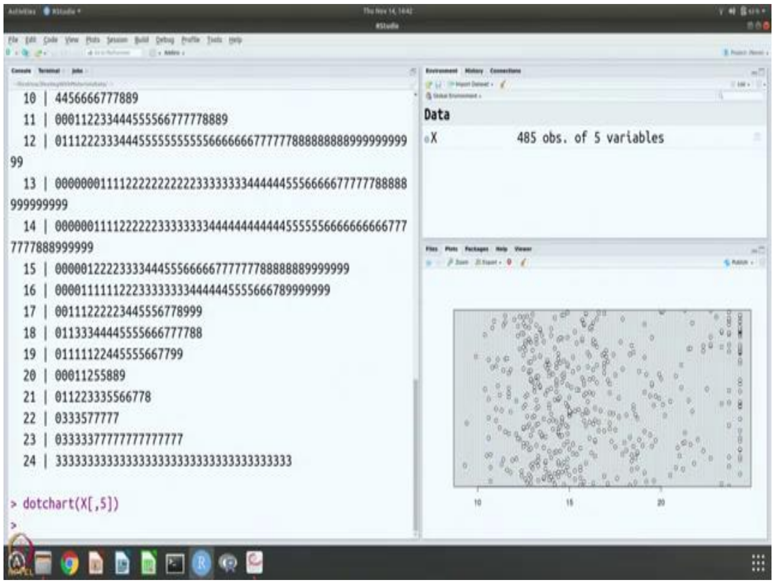
```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> X <- read.csv("../Data/GrainSizeDataSet1.csv")
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file '../Data/GrainSizeDataSet1.csv': No such fi
le or directory
> X <- read.csv("Data/GrainSizeDataSet1.csv")
> library("plotrix")
> gap.plot(X[,1],X[,5],gap=c(230,37600),gap.axis="x",
+          main="Grain ID and ASTM grain size",xlab="Grain I
D",
+          ylab="ASTM Grain Size",xtics=c(0,100,200,37600,3770
0,
+                                         37800,37900,3800,381
00,38200,38300,38400))
> stem(X[,5])
```

```
Data
X          485 obs. of 5 variables
```



```
+                                         37800,37900,3800,381
00,38200,38300,38400))
> stem(X[,5])

  The decimal point is at the |

   9 | 349
  10 | 4456666777889
  11 | 0001122334445555667777778889
  12 | 0111222333444555555555556666666777778888888889999999
99
  13 | 0000000111122222222222333333334444445556666677777788888
999999999
  14 | 000000111122222223333333344444444444455555666666666777
7777888999999
  15 | 00000122223333444555666667777777888888889999999
  16 | 0000111111222333333334444445555666789999999
  17 | 001112222223445556778999
  18 | 01133344445555666777788
  19 | 0111112244555566 7799
  20 | 00011255889
```

```
Data
X          485 obs. of 5 variables
```



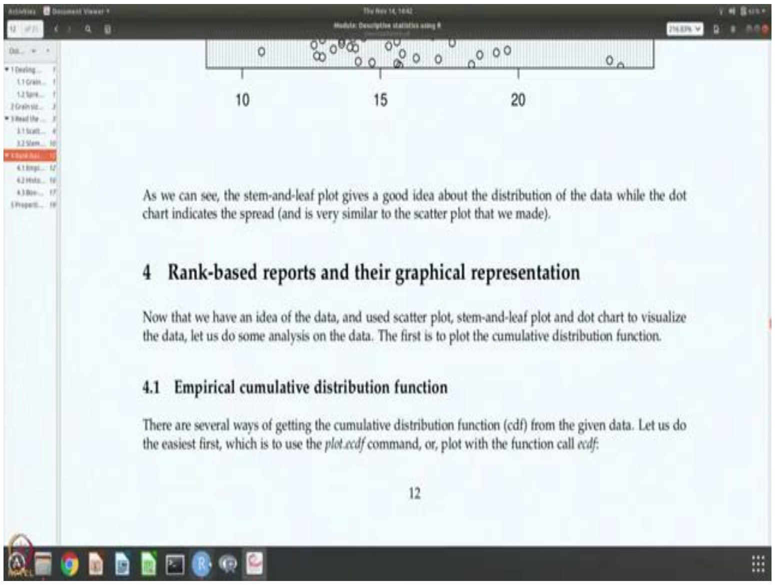
```
  10 | 4456666777889
  11 | 0001122334445555566777778889
  12 | 0111222333444555555555556666666777777888888888899999999
99
  13 | 00000001111222222222222233333333444444555666667777778888
999999999
  14 | 000000111122222223333333344444444444445555555666666666677
7777888999999
  15 | 0000012222333344455566666777777788888889999999
  16 | 00001111112223333333344444455555666789999999
  17 | 0011122222344555678899
  18 | 011333444455555666777788
  19 | 011111224455555667799
  20 | 00011255889
  21 | 011223335566778
  22 | 0333577777
  23 | 033333777777777777777
  24 | 33333333333333333333333333333333333333

> dotchart(X[,5])
>
```

Let us continue exploring we just made a scatterplot and now we are going to make a stem plot, stem and leaf plot is very easy. So, you have to just say which is the variable. So, in this case we are plotting the grain size and so we just say stem and you get the state. So, now you can see the decimal point is at the pipe symbol itself so, it is 9.3, 9.4, 9.9, 10.4, 10.4, 10.5, 10.6, 10.6, etc., ok.

So, this gives you a good idea also about the data spread and you can already see that, ok. So, this looks like peak and there seems to be another peak towards the end. So, it looks like it has two peaks that is what the stem plot indicates and of course, you can also make a dot chart. So, we are going to use the dot chart and the same quantity. So, we want to look at the ASTM grain size in dot chart and so, this is the dot chart.

So, this is the grain size somewhere about 10 to somewhere about probably 27, 28 and, and the dot chart shows you previously I showed you that dot chart can show you the, the extreme points or the outliers, but it is very difficult from this figure to figure out which one is an outlier. So, dot chart and the stem and leaf plot are two other ways of visualizing data and once we have done that, of course, let us go do some rank based reports and represent them graphically.

(Refer Slide Time: 2:09)



As we can see, the stem-and-leaf plot gives a good idea about the distribution of the data while the dot chart indicates the spread (and is very similar to the scatter plot that we made).

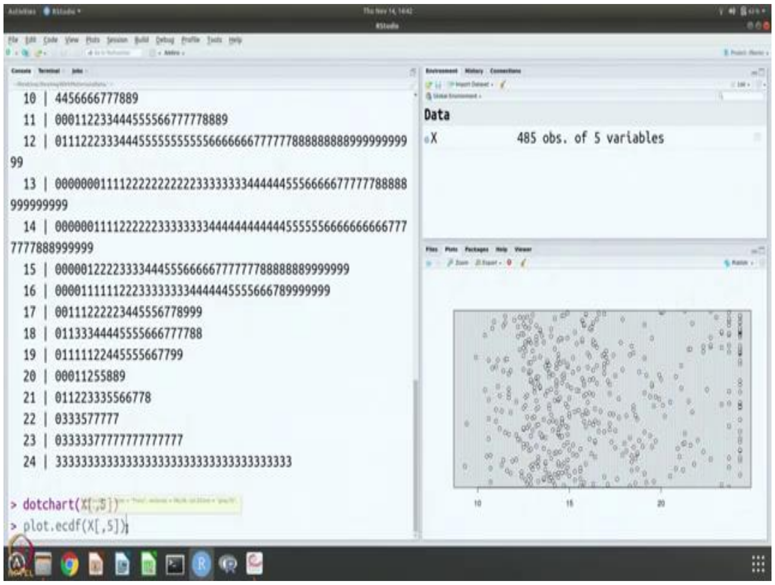
4 Rank-based reports and their graphical representation

Now that we have an idea of the data, and used scatter plot, stem-and-leaf plot and dot chart to visualize the data, let us do some analysis on the data. The first is to plot the cumulative distribution function.

4.1 Empirical cumulative distribution function

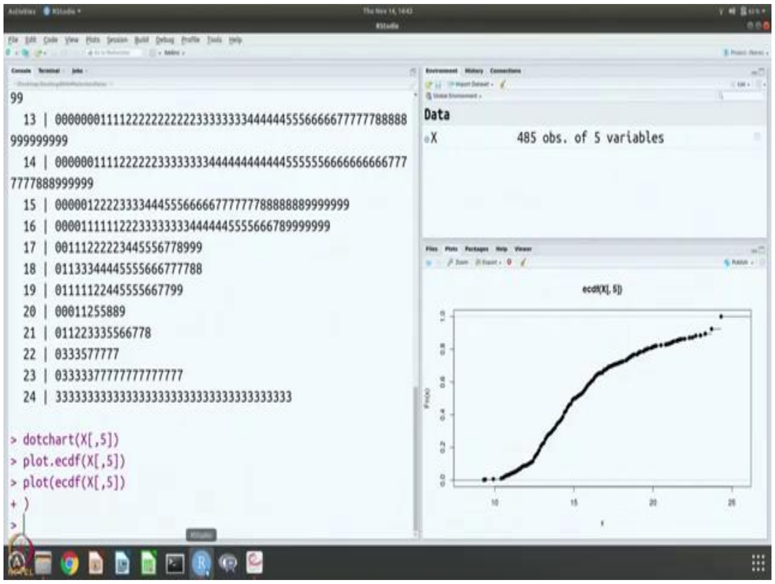
There are several ways of getting the cumulative distribution function (cdf) from the given data. Let us do the easiest first, which is to use the *plot.ecdf* command, or, plot with the function call *ecdf*:

12



```
  10 | 4456666777889
  11 | 000112233444555566777778889
  12 | 0111222333444555555555556666666777777888888888999999999
99
  13 | 0000000111122222222222233333333344444455566666777777788888
999999999
  14 | 0000001111222222333333333444444444444455555566666666666777
7777888999999
  15 | 0000012222333344455566666777777788888889999999
  16 | 0000111111222333333334444445555666789999999
  17 | 001112222223445556778999
  18 | 01133344445555666777788
  19 | 011111224455555667799
  20 | 00011255889
  21 | 011223335566778
  22 | 0333577777
  23 | 0333337777777777777
  24 | 3333333333333333333333333333333333333

> dotchart(X[,5])
> plot.ecdf(X[,5])
```

X   485 obs. of 5 variables

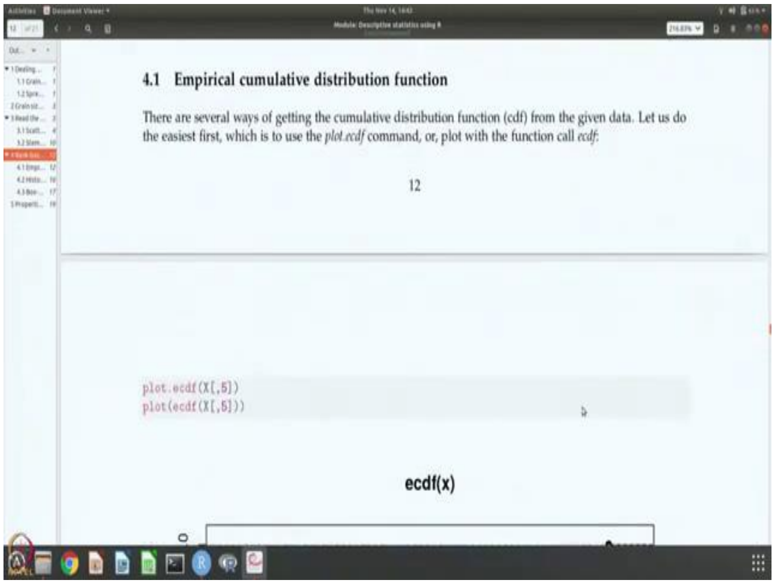


```
99
  13 | 00000001111222222222222333333334444445556666677777788888
999999999
  14 | 000000111122222223333333344444444444455555566666666666777
7777888999999
  15 | 000001222233334445556666677777778888888899999999
  16 | 000011111122233333333444444555566789999999
  17 | 0011122222344555677899
  18 | 0113334444555566677788
  19 | 0111112244555567799
  20 | 00011255889
  21 | 011223335566778
  22 | 0333577777
  23 | 0333337777777777777
  24 | 3333333333333333333333333333333333333

> dotchart(X[,5])
> plot.ecdf(X[,5])
> plot(ecdf(X[,5])
+ )
>
```

Data
X 485 obs. of 5 variables

We have seen the empirical cumulative distribution function and we know that the easiest way to generate one is to say plot is ecdf and we say 5. This is to plot the empirical cumulative distribution function. So, you see this and so you can see the there is another way which also we have learned, we can just say plot ecdf of this data. So, that is no different and we also saw yesterday how to make our own cumulative distribution plot just to remind ourselves how it is done. Let us do it once more.

(Refer Slide Time: 3:05)

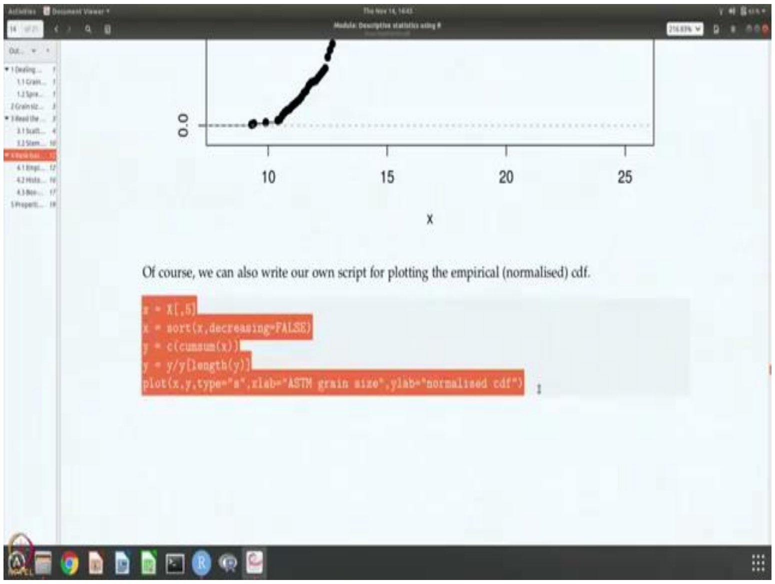


4.1 Empirical cumulative distribution function

There are several ways of getting the cumulative distribution function (cdf) from the given data. Let us do the easiest first, which is to use the *plot.ecdf* command, or, plot with the function call *ecdf*:

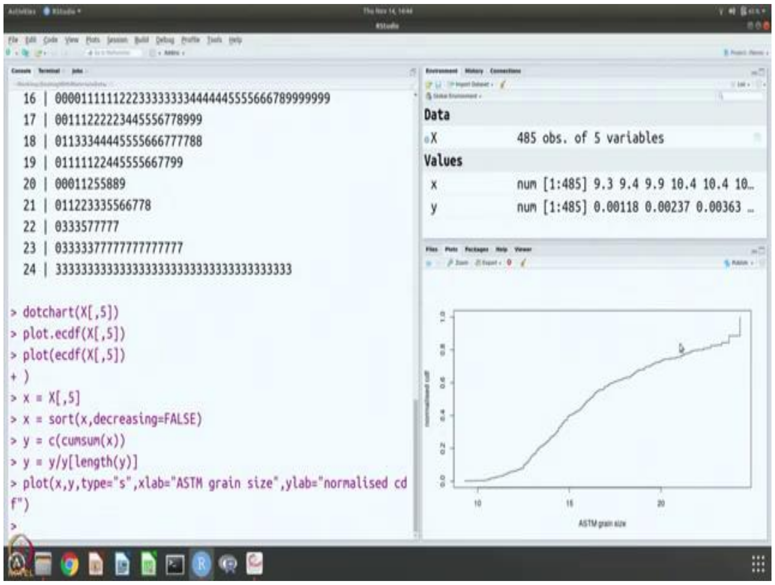
12

```
plot.ecdf(X[,5])
plot(ecdf(X[,5]))
```



Of course, we can also write our own script for plotting the empirical (normalised) cdf.

```
x = X[,5]
x = sort(x,decreasing=FALSE)
y = c(cumsum(x))
y = y/y[length(y)]
plot(x,y,type="s",xlab="ASTM grain size",ylab="normalised cdf")
```



```
  16 | 0000111111222333333334444445555666789999999
  17 | 0011122222344555677899 9
  18 | 0113334444555566677778 8
  19 | 0111112244555566779 9
  20 | 00011255889
  21 | 011223335566778
  22 | 0333577777
  23 | 03333377777777777777
  24 | 3333333333333333333333333333333333333

> dotchart(X[,5])
> plot.ecdf(X[,5])
> plot(ecdf(X[,5])
+ )
> x = X[,5]
> x = sort(x,decreasing=FALSE)
> y = c(cumsum(x))
> y = y/y[length(y)]
> plot(x,y,type="s",xlab="ASTM grain size",ylab="normalised cd
f")
>
```

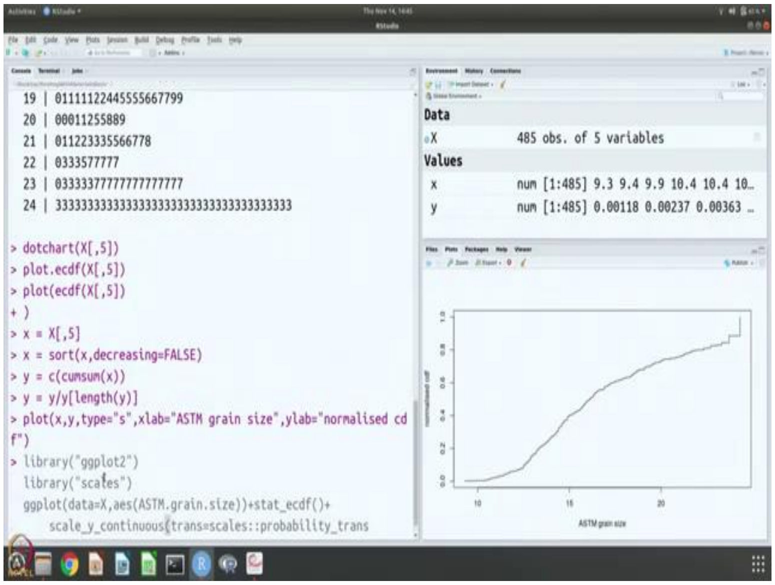
```
Data
X        485 obs. of 5 variables
Values
x        num [1:485] 9.3 9.4 9.9 10.4 10.4 10…
y        num [1:485] 0.00118 0.00237 0.00363 …
```

So, it is a good idea to know what this empirical, empirical cumulative distribution function is. So, let us generate it ourselves. So, this is the one. So, let us look at it. So, first I say that, ok. Take the fifth column and call that as x and sort that x in increasing order, because decreasing its false and store it back in x, y is the cumulative sum of x and we are going to take the final value, and we are going to divide by that so that the numbers go from 0 to 1 you see, so it is normalized.

So, this is the normalization step and then we are going to plot the, the grain size versus the cumulative function and we are going to use the step type for plotting and of course, the x label is ASTM grain size and y label is normalized, y label is normalized CDF. So, you get this. So, this is no different from the previous figures that you generated, but except that now we have written the code ourselves.

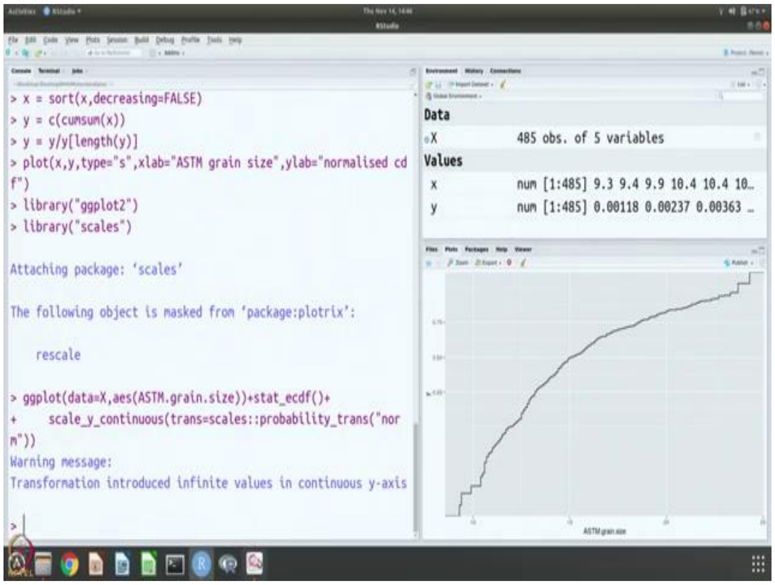
So, R is both a software and a programming language. So, you can just call a function or you can write your own code to do the same thing. So, this we have done earlier also once but this is just to remind you of how a CDF is generated.

(Refer Slide Time: 4:40)



```
 19 | 01111122445555667799
 20 | 00011255889
 21 | 011223335566778
 22 | 0333577777
 23 | 033333777777777777777
 24 | 333333333333333333333333333333333333

> dotchart(X[,5])
> plot.ecdf(X[,5])
> plot(ecdf(X[,5])
+ )
> x = X[,5]
> x = sort(x,decreasing=FALSE)
> y = c(cumsum(x))
> y = y/y[length(y)]
> plot(x,y,type="s",xlab="ASTM grain size",ylab="normalised cd
f")
> library("ggplot2")
  library("scales")
  ggplot(data=X,aes(ASTM.grain.size))+stat_ecdf()+
      scale_y_continuous(trans=scales::probability_trans
```

```
Data
X           485 obs. of 5 variables
Values
x           num [1:485] 9.3 9.4 9.9 10.4 10.4 10…
y           num [1:485] 0.00118 0.00237 0.00363 …
```



```
> x = sort(x,decreasing=FALSE)
> y = c(cumsum(x))
> y = y/y[length(y)]
> plot(x,y,type="s",xlab="ASTM grain size",ylab="normalised cd
f")
> library("ggplot2")
> library("scales")

Attaching package: ‘scales’

The following object is masked from ‘package:plotrix’:

    rescale

> ggplot(data=X,aes(ASTM.grain.size))+stat_ecdf()+
+     scale_y_continuous(trans=scales::probability_trans("nor
m"))
Warning message:
Transformation introduced infinite values in continuous y-axis

> 
```

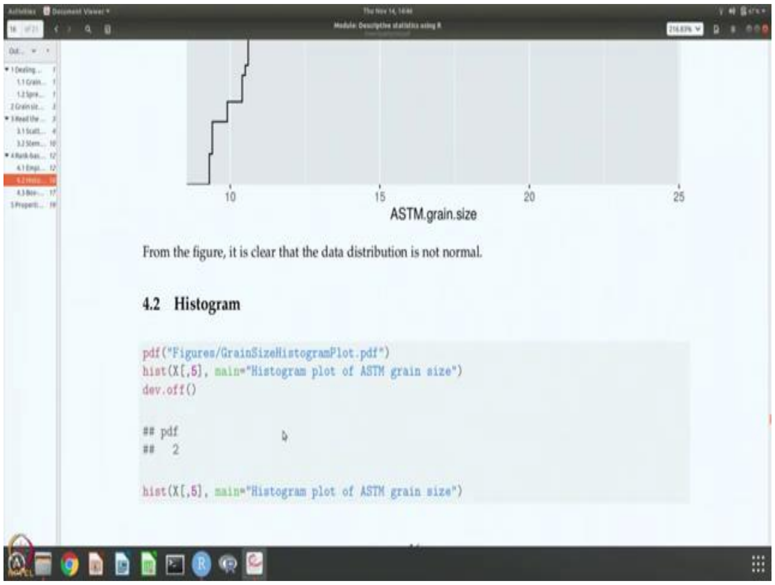
```
Data
X           485 obs. of 5 variables
Values
x           num [1:485] 9.3 9.4 9.9 10.4 10.4 10…
y           num [1:485] 0.00118 0.00237 0.00363 …
```

Of course, one can use ggplot and we learned yesterday that using ggplot it is easier to change the Y scale and so we have to use the library scales also and ggplot you have to tell which is the data, you have to tell which is the aesthetics. So, we want to applaud the ASTM grain size and what is the plot? It is a ECDF plot. And so you have to do the statistical analysis for the cumulative distribution.

And the scale of y should be probability scale. So, if it is an actually a normal distribution then this will look like a straight line so by looking at it, you will know what the distribution of the data is. So, that is the reason why we want to put this scale and see if it actually shows that or it shows any deviation.

So, let us do that and we see that here also there is a deviation. So, it is not a straight line. So, you do not expect the grain size distribution to be normal and of course, there is a warning message. So, the transformation introduced infinite values in the y axis. So, but that is not crucial. So, we do not worry about it.

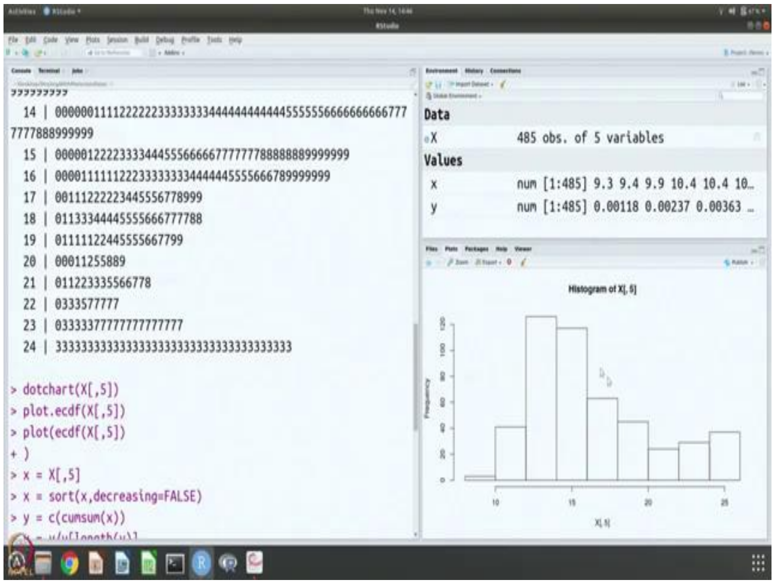
(Refer Slide Time: 5:57)



From the figure, it is clear that the data distribution is not normal.

4.2 Histogram

```
pdf("Figures/GrainSizeHistogramPlot.pdf")
hist(X[,5], main="Histogram plot of ASTM grain size")
dev.off()

## pdf
##   2

hist(X[,5], main="Histogram plot of ASTM grain size")
```



```
999999999
  14 | 0000001111222222333333334444444444445555556666666666777
7777888999999
  15 | 0000012222333344455566666777777788888889999999
  16 | 0000111111222333333334444445555666789999999
  17 | 001112222223445556778999
  18 | 01133344445555666777788
  19 | 011111122445555667799
  20 | 00011255889
  21 | 011223335566778
  22 | 0333577777
  23 | 03333377777777777777
  24 | 333333333333333333333333333333333333

> dotchart(X[,5])
> plot.ecdf(X[,5])
> plot(ecdf(X[,5])
+ )
> x = X[,5]
> x = sort(x,decreasing=FALSE)
> y = c(cumsum(x))
```

Data

X 485 obs. of 5 variables

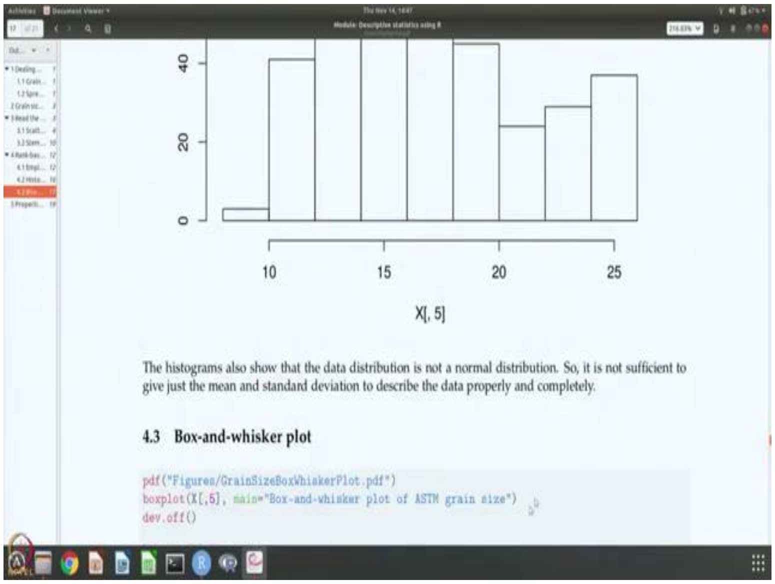
Values

x num [1:485] 9.3 9.4 9.9 10.4 10.4 10…

y num [1:485] 0.00118 0.00237 0.00363 …

So let us plot histogram and that is easy. This is a histogram. And as we saw when we did the stem and leaf plot. So, there is a peak, there is this value then goes down. And then there is a smaller peak here, which is what you saw here, it came down and then it went to a peak here. So, you can see in the histogram also this sort of second peak, it is not quite a peak, but it is a rather large tail and very fat tail. So, if this is distribution, then this is a much larger tail, fatter tail. So, this is very common, sometimes data does not really show nice well shaped curves. And here is an example.

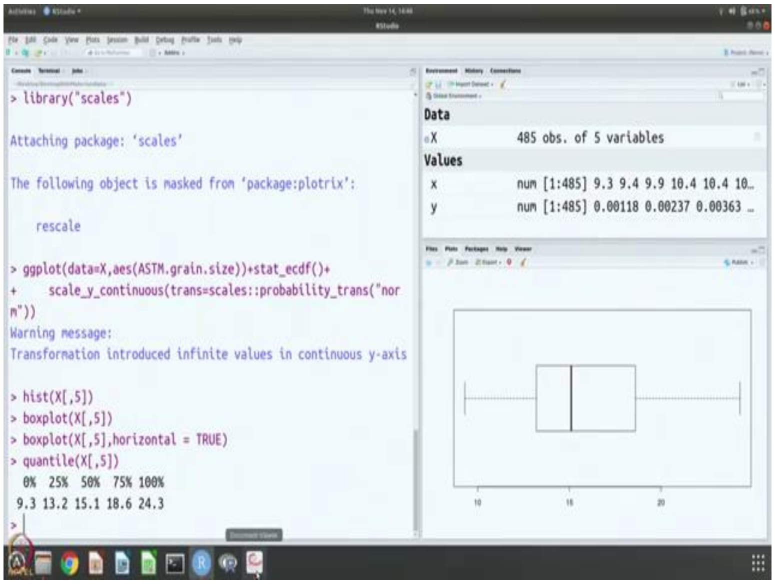
(Refer Slide Time: 6:58)



The histograms also show that the data distribution is not a normal distribution. So, it is not sufficient to give just the mean and standard deviation to describe the data properly and completely.

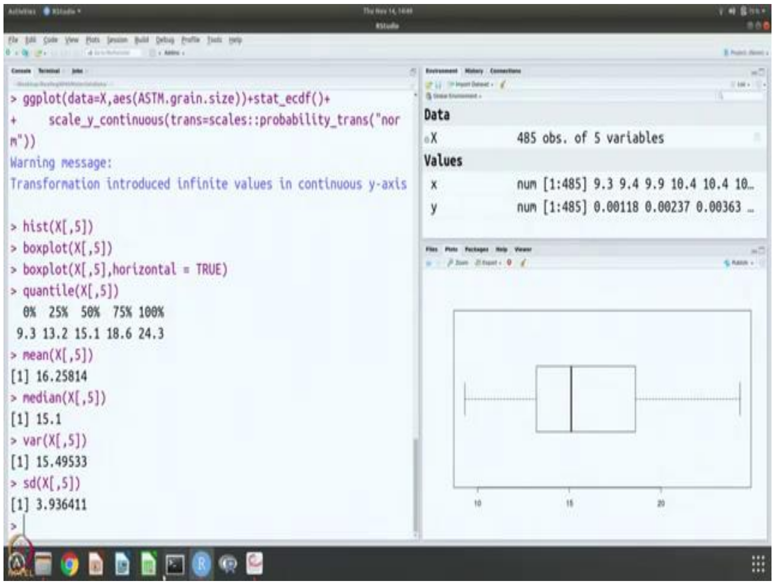
4.3 Box-and-whisker plot

```
pdf("Figures/GrainSizeBoxWhiskerPlot.pdf")
boxplot(X[,5], main="Box-and-whisker plot of ASTM grain size")
dev.off()
```



```
> library("scales")

Attaching package: ‘scales’

The following object is masked from ‘package:plotrix’:

    rescale

> ggplot(data=X,aes(ASTM.grain.size))+stat_ecdf()+
+     scale_y_continuous(trans=scales::probability_trans("nor
m"))
Warning message:
Transformation introduced infinite values in continuous y-axis

> hist(X[,5])
> boxplot(X[,5])
> boxplot(X[,5],horizontal = TRUE)
> quantile(X[,5])
  0%  25%  50%  75% 100% 
 9.3 13.2 15.1 18.6 24.3 
> 
```

```
Data
X        485 obs. of 5 variables
Values
x        num [1:485] 9.3 9.4 9.9 10.4 10.4 10…
y        num [1:485] 0.00118 0.00237 0.00363 …
```



```
> ggplot(data=X,aes(ASTM.grain.size))+stat_ecdf()+
+     scale_y_continuous(trans=scales::probability_trans("nor
m"))
Warning message:
Transformation introduced infinite values in continuous y-axis

> hist(X[,5])
> boxplot(X[,5])
> boxplot(X[,5],horizontal = TRUE)
> quantile(X[,5])
  0%  25%  50%  75% 100% 
 9.3 13.2 15.1 18.6 24.3 
> mean(X[,5])
[1] 16.25814
> median(X[,5])
[1] 15.1
> var(X[,5])
[1] 15.49533
> sd(X[,5])
[1] 3.936411
> 
```

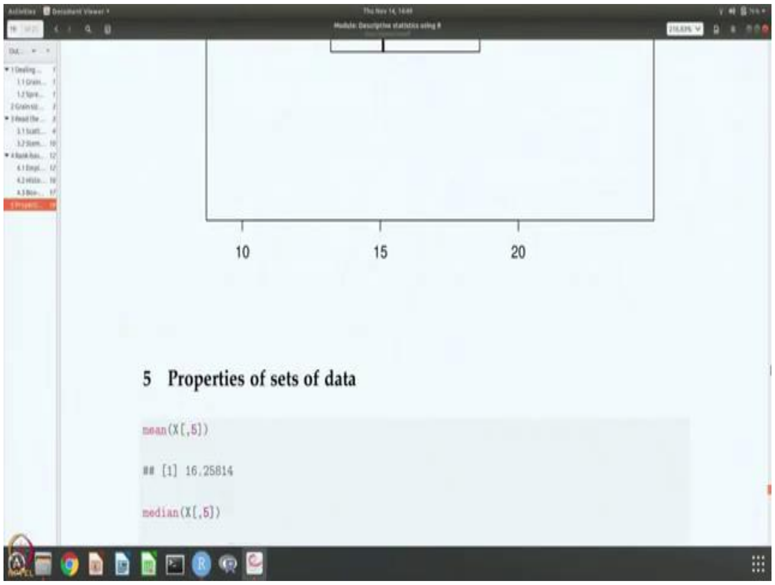
```
Data
X        485 obs. of 5 variables
Values
x        num [1:485] 9.3 9.4 9.9 10.4 10.4 10…
y        num [1:485] 0.00118 0.00237 0.00363 …
```

After histogram plot, of course, yesterday in the previous session, we did the box-and-whisker plot. So, let us do that. So, let us say boxplot. So, we have the boxplot and as usual, so we can make the boxplot with the horizontal to be true. So, you can see that this is the mean and this, box actually represents the second and third the quantiles, ok. So, to get this idea, let us do this command now quantile which will clearly show what is happening.

So, 50 percent of the data is somewhere here 15.1. And 25 percent of the data is from 25 to 50 happens between 13.2 somewhere here to 15.1 and 18.6 is by the time 75 so 25 to 75 percent of the data lies here and this is on one side the first quantile and this is the last quantile. So, that is what this boxplot actually represents. So, it gives you an idea of spread of the data. So, so, this is another way of looking at the spread of the data. So, that is what we have seen here.

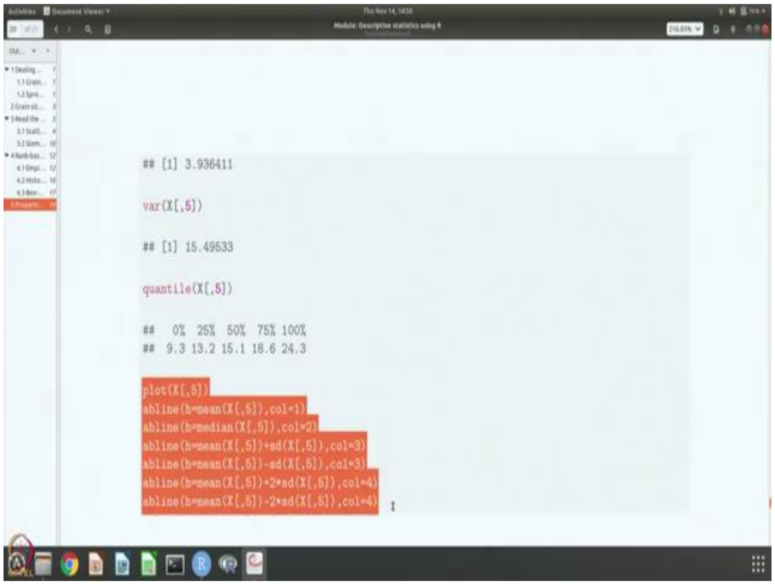
So, once we have the So, we have exhausted all the rank based reports that one can prepare and we have even looked at one of the summary values and of course, we can get the other ones the one is mean. So that is 16.3. So, that is the mean value. Let us look at the medium value that is 15.1 that is where this line is there. This dot line actually represents the median and variance. So, variance is 15.5 and the standard deviation and that is some 3.9. So, that is the standard deviation.

(Refer Slide Time: 9:36)

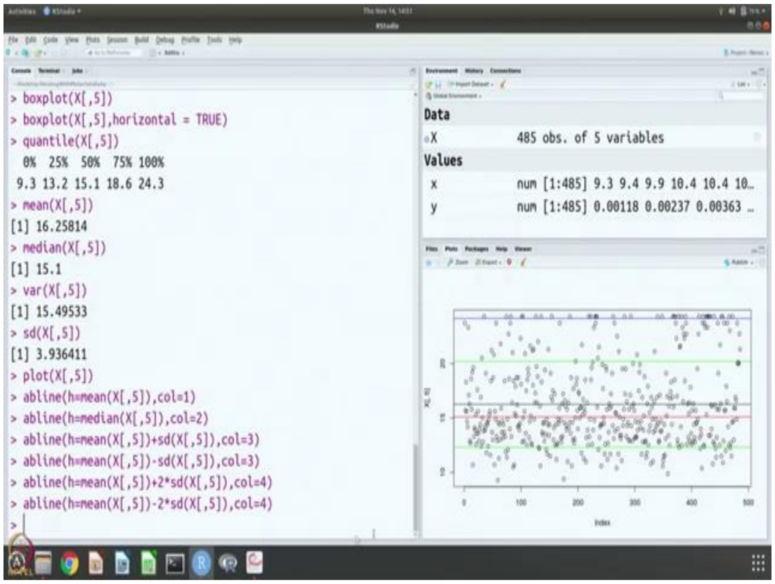


5 Properties of sets of data

```
mean(X[,5])

## [1] 16.25814

median(X[,5])
```



```
## [1] 3.936411

var(X[,5])

## [1] 15.49533

quantile(X[,5])

##   0%  25%  50%  75% 100%
##  9.3 13.2 15.1 18.6 24.3

plot(X[,5])
abline(h=mean(X[,5]),col=1)
abline(h=median(X[,5]),col=2)
abline(h=mean(X[,5])+sd(X[,5]),col=3)
abline(h=mean(X[,5])-sd(X[,5]),col=3)
abline(h=mean(X[,5])+2*sd(X[,5]),col=4)
abline(h=mean(X[,5])-2*sd(X[,5]),col=4)
```



```
> boxplot(X[,5])
> boxplot(X[,5],horizontal = TRUE)
> quantile(X[,5])
  0%  25%  50%  75% 100%
 9.3 13.2 15.1 18.6 24.3
> mean(X[,5])
[1] 16.25814
> median(X[,5])
[1] 15.1
> var(X[,5])
[1] 15.49533
> sd(X[,5])
[1] 3.936411
> plot(X[,5])
> abline(h=mean(X[,5]),col=1)
> abline(h=median(X[,5]),col=2)
> abline(h=mean(X[,5])+sd(X[,5]),col=3)
> abline(h=mean(X[,5])-sd(X[,5]),col=3)
> abline(h=mean(X[,5])+2*sd(X[,5]),col=4)
> abline(h=mean(X[,5])-2*sd(X[,5]),col=4)
>
```

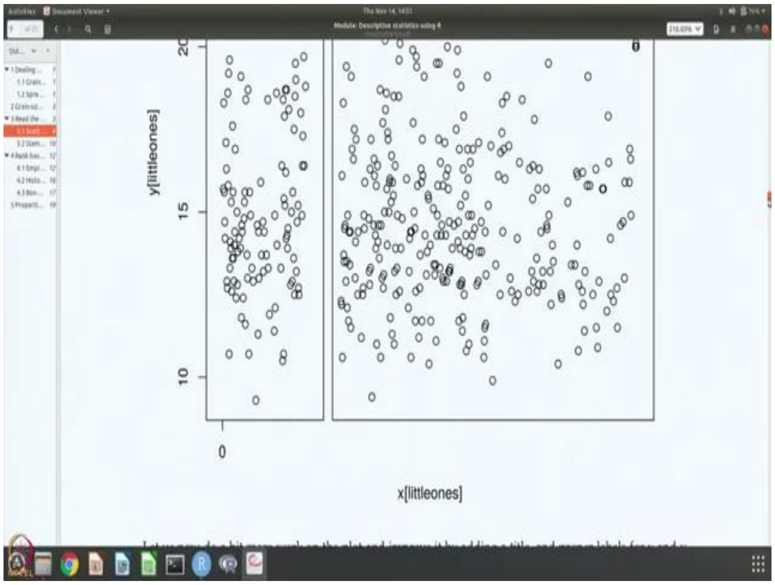
```
Data
X      485 obs. of 5 variables
Values
x      num [1:485] 9.3 9.4 9.9 10.4 10.4 10…
y      num [1:485] 0.00118 0.00237 0.00363 …
```

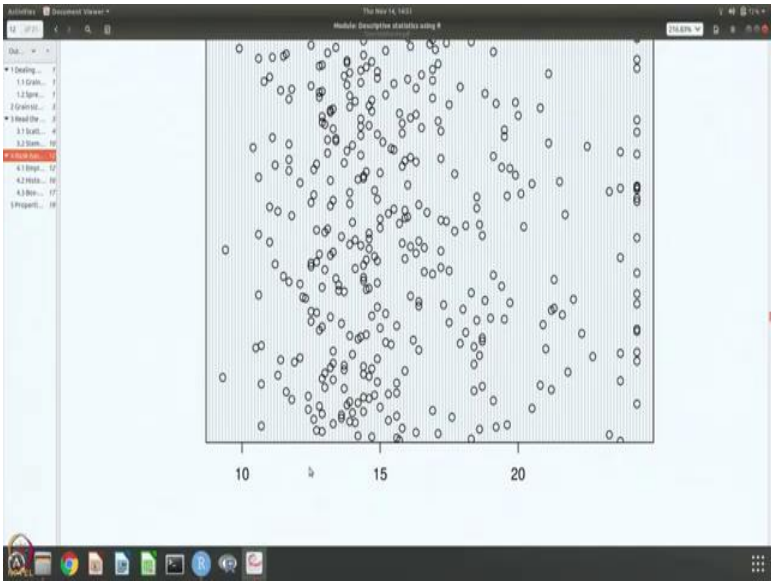
Of course, we want to plot these numbers along with this scatterplot to get a better idea. So, let us do that. So, what are we doing? We are plotting the data like a scatterplot and then we are drawing lines for the mean and the median and mean plus standard deviation, mean minus standard deviation, mean plus 2 standard deviation, and mean minus 2 standard deviation.

So, so, you can see that mean is here 16 something 16.3 and the median is 15.1. And these green lines represent points which are within one standard deviation and the blue lines represent points which are within the 2 times standard deviation. So, this blue line you cannot even see here. So, these points are all lying between mean and minus 2 times standard deviation. But on the other side, you can see large number of data points that are lying just outside of this 2 sigma.

So, these are basically the points that are outliners. So, so, summarize, we have already looked at quantile, we have plotted the data.

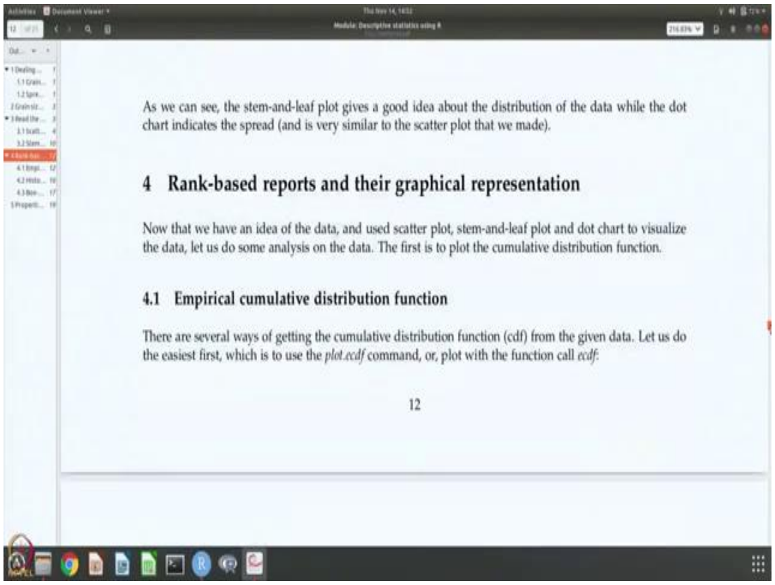
(Refer Slide Time: 11:16)





So, that is the first thing that we did and we had a gap x axis. But, you do not have to do that because your dot chart, for example, does the same thing without any introducing any gap or anything. Just looking at the numbers. This is because it is really not putting the grain IDs if you have to have grain ID here and the numbers there, then you have to use a dot chart, the gap chart gap plot, but a dot chart otherwise can give you the complete data in one go. So, these are ways of looking at the data.

(Refer Slide Time: 11:48)



As we can see, the stem-and-leaf plot gives a good idea about the distribution of the data while the dot chart indicates the spread (and is very similar to the scatter plot that we made).

4 Rank-based reports and their graphical representation

Now that we have an idea of the data, and used scatter plot, stem-and-leaf plot and dot chart to visualize the data, let us do some analysis on the data. The first is to plot the cumulative distribution function.

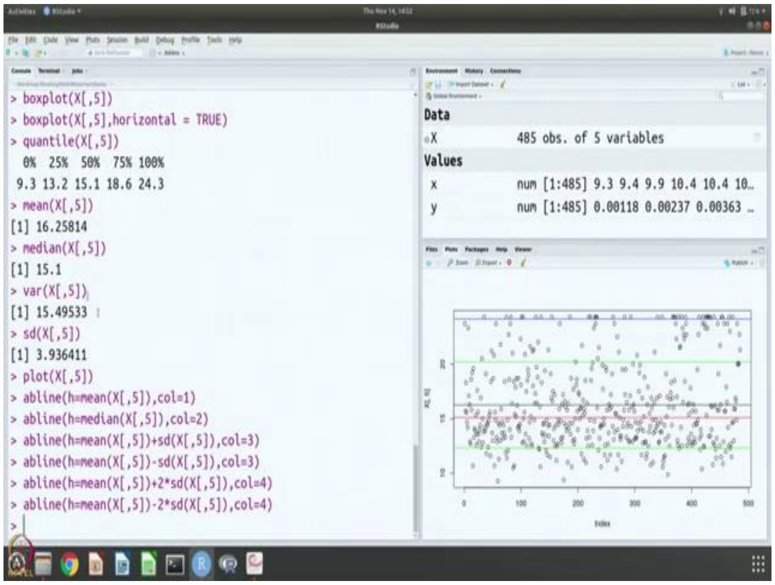
4.1 Empirical cumulative distribution function

There are several ways of getting the cumulative distribution function (cdf) from the given data. Let us do the easiest first, which is to use the *plot.ecdf* command, or, plot with the function call *ecdf*:

12

Then we made the several rank based reports and represented them graphically; CDF, histogram, box-plot and things like that.

(Refer Slide Time: 11:57)



```
> boxplot(X[,5])
> boxplot(X[,5],horizontal = TRUE)
> quantile(X[,5])
  0%  25%  50%  75% 100%
 9.3 13.2 15.1 18.6 24.3
> mean(X[,5])
[1] 16.25814
> median(X[,5])
[1] 15.1
> var(X[,5])
[1] 15.49533
> sd(X[,5])
[1] 3.936411
> plot(X[,5])
> abline(h=mean(X[,5]),col=1)
> abline(h=median(X[,5]),col=2)
> abline(h=mean(X[,5])+sd(X[,5]),col=3)
> abline(h=mean(X[,5])-sd(X[,5]),col=3)
> abline(h=mean(X[,5])+2*sd(X[,5]),col=4)
> abline(h=mean(X[,5])-2*sd(X[,5]),col=4)
>
```

Data	
X	485 obs. of 5 variables
Values	
x	num [1:485] 9.3 9.4 9.9 10.4 10.4 10...
y	num [1:485] 0.00118 0.00237 0.00363 ...

And then we have made the summary based reports, like mean, median, variance, start deviation, quantiles, etc. And then we actually plot all the data points and also these summary based numbers on the same plot to have an idea about the spread of the data and outliers.

So, this completes the analysis, descriptive data analysis for data set 1. Now, let us take the more complicated data set 2 which is meant for two different phases and do the analysis and see what that has to tell us. Thank you.