Introduction to Proteogenomics

Dr. Sanjeeva Srivastava
Dr. David Fenyo
Department of Biosciences and Bioengineering
New York University
Indian Institute of Technology, Bombay

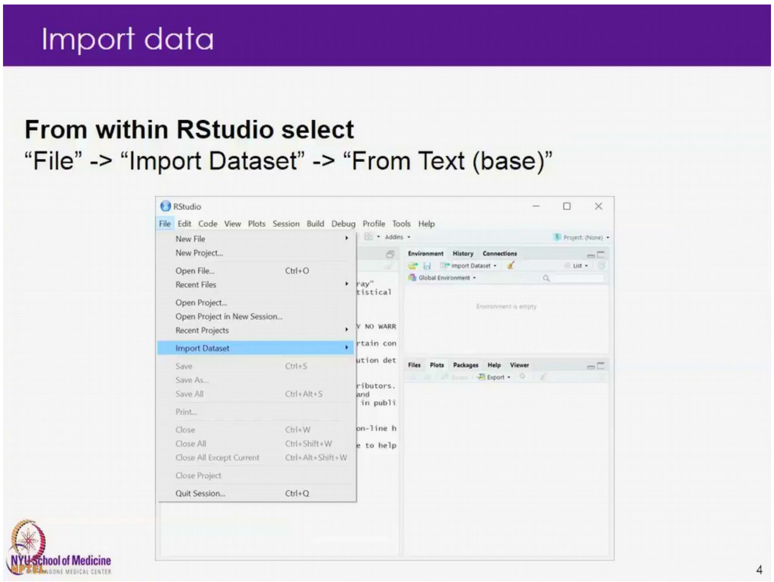
Supplementary - 13
Predictive Analysis

Welcome to MOOC course on Introduction to Proteogenomics. Today we have a hands on session by Doctor David Fenyo. In this session he will talk to you about some basic informations like how to open a file in R, how to transpose your data and how to run a command. A background knowledge in R will help you to understand these things must quickly next he will also discuss about how we can generate different kinds of plot and correlation study using R.

Finally, he will discuss about binary classification model and how to plot a binarized copy number. So, let us welcome Doctor David Fenyo for today's hands on session.

You should start our RStudio everyone should have that installed from previously.

(Refer Slide Time: 01:25)

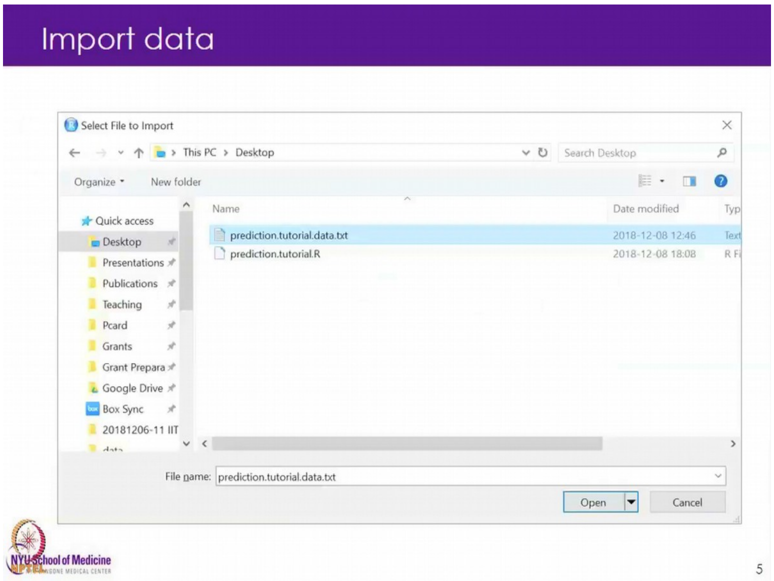


Import data

From within RStudio select
"File" -> "Import Dataset" -> "From Text (base)"

And then you need to do two things one is to import the data and if you can look at the PowerPoint yourself allow I will show it quickly here but and but you should have the PowerPoint also. So, you can look at it how you import data.

(Refer Slide Time: 01:35)



Import data

Select File to Import

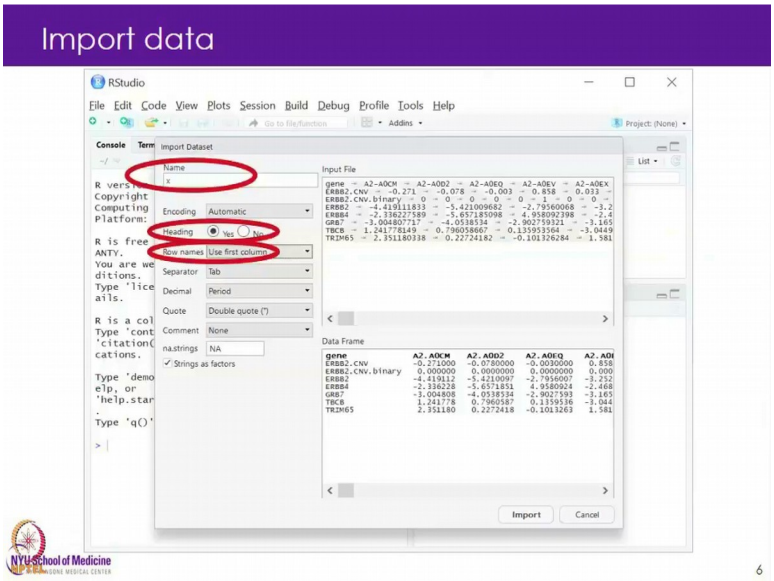
This PC › Desktop

Name	Date modified	Typ
prediction.tutorial.data.txt	2018-12-08 12:46	Text
prediction.tutorial.R	2018-12-08 18:08	R Fi

File name: prediction.tutorial.data.txt

And it is the dot txt file that is the data.

(Refer Slide Time: 01:41)



Import data

RStudio

Import Dataset

Name: x

Encoding: Automatic

Heading: ● Yes ○ No

Row names: Use first column

Separator: Tab

Decimal: Period

Quote: Double quote (")

Comment: None

na.strings: NA

☑ Strings as factors

Input File

```
gene → A2-A0CM → A2-A0D2 → A2-A0EQ → A2-A0EV → A2-A0EX
ERBB2.CNV → -0.271 → -0.078 → -0.003 → 0.858 → 0.033 →
ERBB2.CNV.binary → 0 → 0 → 0 → 0 → 0 → 1 → 0 → 0 →
ERBB2 → -4.419111833 → -5.421009682 → -2.79560068 → -3.2
ERBB4 → -2.336227589 → -5.657185098 → 4.958092398 → -2.4
GRB7 → -3.004807717 → -4.0538534 → -2.902759321 → -3.165
TBCB → 1.241778149 → 0.796058667 → 0.135953564 → -3.0449
TRIM65 → 2.351180338 → 0.22724182 → -0.101326284 → 1.581
```

Data Frame

```
gene              A2.A0CM    A2.A0D2    A2.A0EQ     A2.A0I
ERBB2.CNV        -0.271000  -0.0780000  -0.0030000  0.858
ERBB2.CNV.binary  0.000000   0.0000000   0.0000000  0.000
ERBB2            -4.419112  -5.4210097  -2.7956007  -3.252
ERBB4            -2.336228  -5.6571851   4.9580924  -2.468
GRB7             -3.004808  -4.0538534  -2.9027593  -3.165
TBCB              1.241778   0.7960587   0.1359536  -3.044
TRIM65            2.351180   0.2272418  -0.1013263   1.581
```

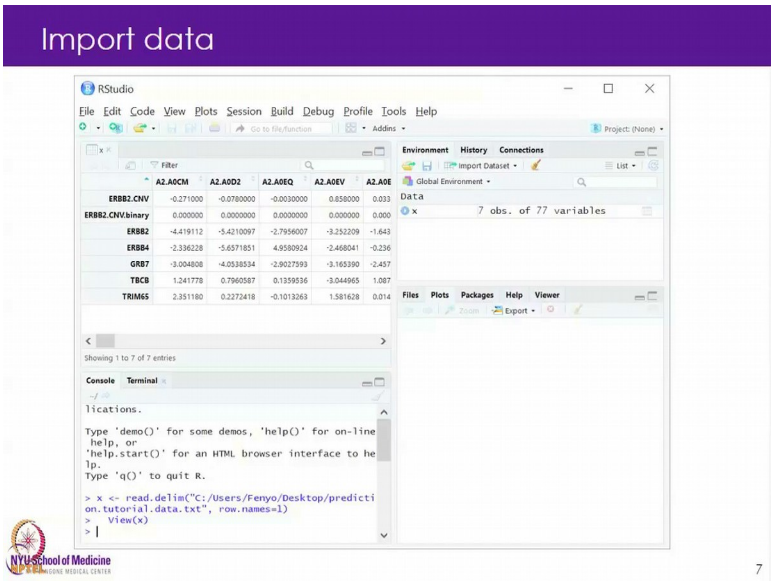
And the only thing that is important to do when you import open the data file is that, you change the name and call it x just for so, that it is short and you should click yes here on the heading.

So, then it will take the sample names as the column headings and you should select use the first column as the row name. So, these are the gene names. So, so the data is a small data set. So, it is the 77 breast cancer samples from CPTAC that you have seen already before and I have all just done a small selection of genes.

So, these are from the bottom here there are 5 genes and these are the protein levels that we measured and it is the log 2 protein levels. So, it is ERBB 2, ERBB 4, GRB 7, TBCB and TRIM 65. So, there was are the 5 protein and then they have a value for each sample here and then the other thing we have is copy number. So, we have copy number only for ERBB 2 and we also I have binarized the copy number.

So, divided it into low and high. So, that is that is all the data and then you just click on import but if you do not for example, if you change do not change the name to x, it is going to be complicated than later for you and.

(Refer Slide Time: 03:35)



Import data

	A2.A0CM	A2.A0D2	A2.A0EQ	A2.A0EV	A2.A0E
ERBB2.CNV	-0.271000	-0.0780000	-0.0030000	0.858000	0.033
ERBB2.CNV.binary	0.000000	0.0000000	0.0000000	0.000000	0.000
ERBB2	-4.419112	-5.4210097	-2.7956007	-3.252209	-1.643
ERBB4	-2.336228	-5.6571851	4.9580924	-2.468041	-0.236
GRB7	-3.004808	-4.0538534	-2.9027593	-3.165390	-2.457
TBCB	1.241778	0.7960587	0.1359536	-3.044965	1.087
TRIM65	2.351180	0.2272418	-0.1013263	1.581628	0.014

Showing 1 to 7 of 7 entries

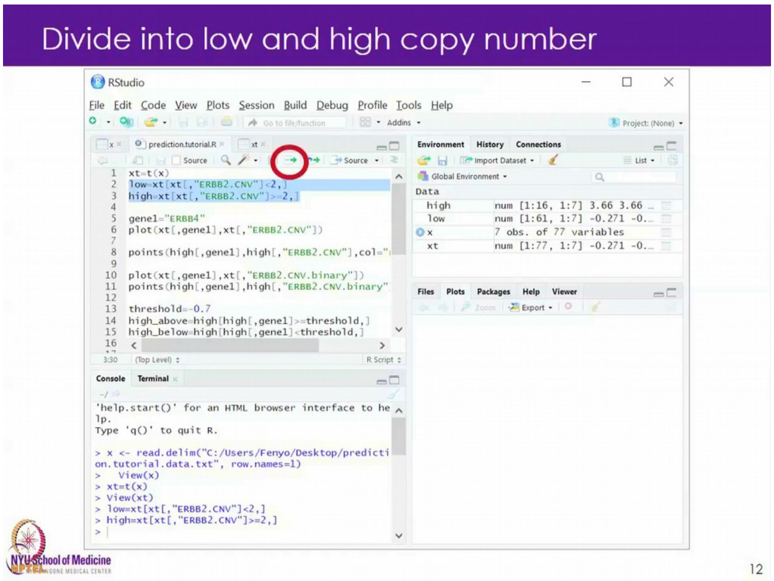
Data
x 7 obs. of 77 variables

```
lications.

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

> x <- read.delim("C:/Users/Fenyo/Desktop/predicti
on.tutorial.data.txt", row.names=1)
>   View(x)
> |
```

So, that is what you should see if after you import it is you should see x here there should be 7; 77 columns and 7 rows and this is a showing a small part of it is.

Now, the next step is to open there the code which is the dot R file.

(Refer Slide Time: 04:03)



Divide into low and high copy number

```
xt=t(x)
low=xt[xt[,"ERBB2.CNV"]<2,]
high=xt[xt[,"ERBB2.CNV"]>=2,]

gene1="ERBB4"
plot(xt[,gene1],xt[,"ERBB2.CNV"])

points(high[,gene1],high[,"ERBB2.CNV"],col="

plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"

threshold=-0.7
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
```

```
Data
  high     num [1:16, 1:7] 3.66 3.66 ...
  low      num [1:61, 1:7] -0.271 -0...
  x        7 obs. of 77 variables
  xt       num [1:77, 1:7] -0.271 -0...
```

```
~/
'help.start()' for an HTML browser interface to he
lp.
Type 'q()' to quit R.

> x <- read.delim("C:/Users/Fenyo/Desktop/predicti
on.tutorial.data.txt", row.names=1)
>   View(x)
> xt=t(x)
> View(xt)
> low=xt[xt[,"ERBB2.CNV"]<2,]
> high=xt[xt[,"ERBB2.CNV"]>=2,]
>
```

And then when you open the code you will see this and we are going to go through that and oh yes. So, there is one more thing that is important is, if you select a few lines and then click on here, I think I have seen some people have it is different ways that it says run may be, but that will then execute the line that you have marked and you can mark several lines.

So, now can anyone explain what this first line means?

And with a we are applying a function called t to x. So, x we can look at x it is where is it, here it looks like this. So, what is going to happen when we applied t the function t?

Student: we are extracting xt.

Yes yes.

Student: Input.

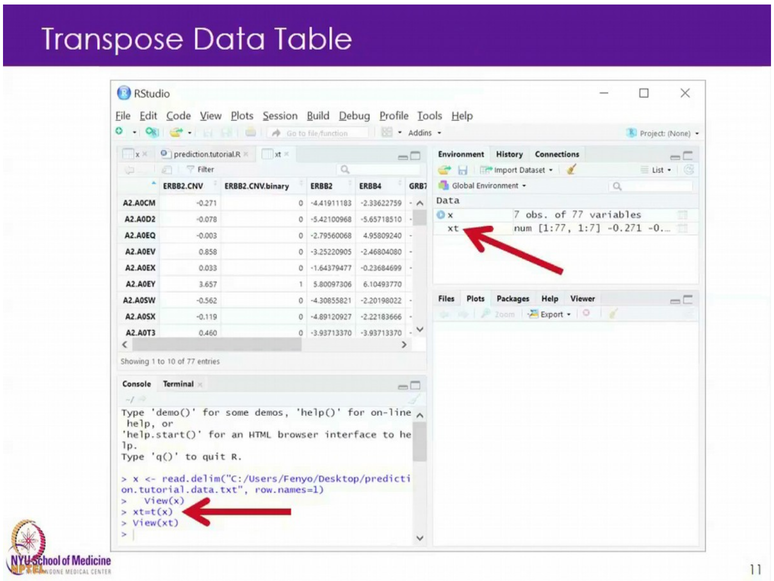
So, everyone knows what transpose is, you guys have transposed it already and I will do it now. So, like that now it appeared here. So, it we can look at it and it is now has the gene names as columns and the samples as rows, then what we want to do is what we are going to predict try to predict copy number that we will try to predict that is the samples that have a high copy number and so, we have this column now in the transposed one and we are going to say that any copy when the copy number is smaller than 2, then it is low and when it is larger than 2 it is high.

And this is now the log version of it. So, we can run these 2 and see that we get in the high we have 16. So, if you remember my slides from the presentation there were a few now few dots and it turns out that they were 16 that had high ERBB 2 copy number and, but most of them 61 or low.

So, now let us try our first plot. So, you should see definitely look here that is there are 16 that have high copy number you should get that and 61 that had low. So, this taking a subset is very useful and we can that is something you will need very often and so, we just. So, what this notation means is that, we take xt and then buts we are only interested in the ones where the ERBB2 copy number is less than 2.

And those we assigned to a new variable that is called low. So, we can actually we can look at let us look at the high because it is a little bit smaller.

(Refer Slide Time: 07:31)



Transpose Data Table

	ERBB2.CNV	ERBB2.CNV.binary	ERBB2	ERBB4
A2.A0CM	-0.271	0	-4.41911183	-2.33622759
A2.A0D2	-0.078	0	-5.42100968	-5.65718510
A2.A0EQ	-0.003	0	-2.79560068	4.95809240
A2.A0EV	0.858	0	-3.25220905	-2.46804080
A2.A0EX	0.033	0	-1.64379477	-0.23684699
A2.A0EY	3.657	1	5.80097306	6.10493770
A2.A0SW	-0.562	0	-4.30855821	-2.20198022
A2.A0SX	-0.119	0	-4.89120927	-2.22183666
A2.A0T3	0.460	0	-3.93713370	-3.93713370

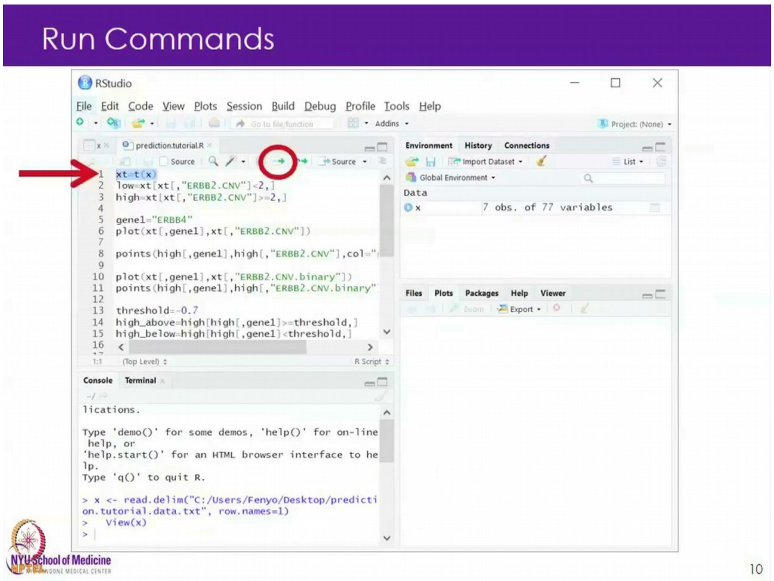
Showing 1 to 10 of 77 entries

```
Data
x          7 obs. of 77 variables
xt         num [1:77, 1:7] -0.271 -0...
```

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

> x <- read.delim("C:/Users/Fenyo/Desktop/predicti
on.tutorial.data.txt", row.names=1)
>   View(x)
> xt=t(x)
> View(xt)
>
```

So, the; however, now we should only have 16 samples here, which probably we have switch this now is a subset is a view of the overall table that has 77 samples but that we have selected only the 16 that are high copy number.

(Refer Slide Time: 07:55)



Run Commands

```
xt=t(x)
low=xt[xt[,"ERBB2.CNV"]<2,]
high=xt[xt[,"ERBB2.CNV"]>=2,]

gene1="ERBB4"
plot(xt[,gene1],xt[,"ERBB2.CNV"])

points(high[,gene1],high[,"ERBB2.CNV"],col="

plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"

threshold=-0.7
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
```

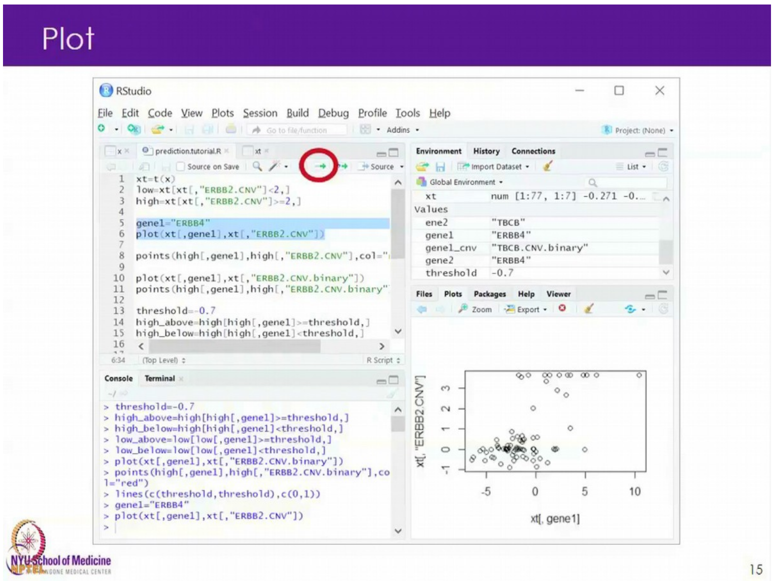
```
lications.

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

> x <- read.delim("C:/Users/Fenyo/Desktop/predicti
on.tutorial.data.txt", row.names=1)
>   View(x)
>
```

```
Data
x          7 obs. of 77 variables
```

So, now we just going to plot we defined gene one as ERBB 4 you just picked one out of the 5 proteomics tracks.

And we are going to plot on the x axis, we are going to plot ERBB 4 and on the y axis we are going to plot the copy number.

(Refer Slide Time: 08:23)



Plot

```
xt=t(x)
low=xt[xt[,"ERBB2.CNV"]<2,]
high=xt[xt[,"ERBB2.CNV"]>=2,]

gene1="ERBB4"
plot(xt[,gene1],xt[,"ERBB2.CNV"])

points(high[,gene1],high[,"ERBB2.CNV"],col="

plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"

threshold=-0.7
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
```

```
> threshold=-0.7
> high_above=high[high[,gene1]>=threshold,]
> high_below=high[high[,gene1]<threshold,]
> low_above=low[low[,gene1]>=threshold,]
> low_below=low[low[,gene1]<threshold,]
> plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
> points(high[,gene1],high[,"ERBB2.CNV.binary"],co
l="red")
> lines(c(threshold,threshold),c(0,1))
> gene1="ERBB4"
> plot(xt[,gene1],xt[,"ERBB2.CNV"])
>
```

```
  xt          num [1:77, 1:7] -0.271 -0.…
Values
  ene2        "TBCB"
  gene1       "ERBB4"
  gene1_cnv   "TBCB.CNV.binary"
  gene2       "ERBB4"
  threshold   -0.7
```

So, and that is plot is going to appear here. So, now, we see we have x axis is ERBB 4 and a copy number is for ERBB 2 on the y axis and we see that is there are these we know that there are 16 in our high group which are these here that is have a higher a copy number.

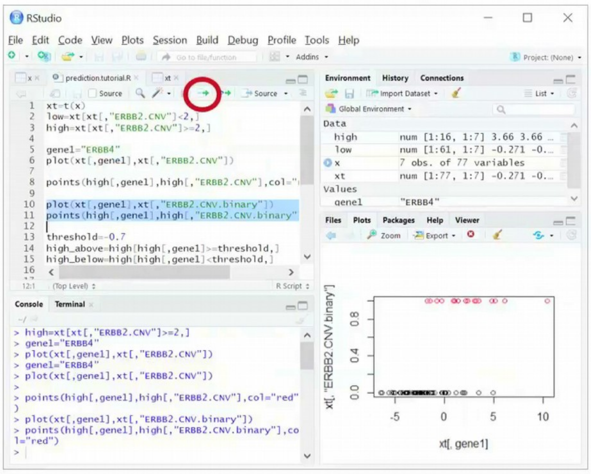
So, we have those 2 um. So, now, we can do one more thing here is to colour these in red. So, let me run it and then I will explain it. So, now, all the ones we are considering high or shown in red. So, what we did is we first plotted everything in the black here in the plot command, but then we overlaids in red but just for the ones that are in this the 16 in this high group.

So, we have now separated them. You can put search for maybe R points and then you will get some and then it will show you that that is what I did yesterday that to set define the colour, you write ”col” short for colour equal to red and the red has to be in quotations and there are lots of other things you can do you can change the shape of these you can have them filled and all that you can look up on the internet how to how to do ok.

So, now we prop that copy number here, but we are really only interested if it is the copy number is low or high and so, there is one other column. So, here we plot the ERBB 2 dot CNV that is column but we have another one that is was ERBB 2 dot CNV dot binary. So, now, we are going to plot that. So, it is the same plot but we changing from the actual copy number to either 0 if the copy number is low or 1 if it is high.

(Refer Slide Time: 10:43)

Plot binarized copy number



```
xt=t(x)
low=xt[xt[,"ERBB2.CNV"]<2,]
high=xt[xt[,"ERBB2.CNV"]>=2,]

gene1="ERBB4"
plot(xt[,gene1],xt[,"ERBB2.CNV"])

points(high[,gene1],high[,"ERBB2.CNV"],col="

plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"

threshold=-0.7
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
```

```
> high=xt[xt[,"ERBB2.CNV"]>=2,]
> gene1="ERBB4"
> plot(xt[,gene1],xt[,"ERBB2.CNV"])
> gene1="ERBB4"
> plot(xt[,gene1],xt[,"ERBB2.CNV"])
>
> points(high[,gene1],high[,"ERBB2.CNV"],col="red"
)
> plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
> points(high[,gene1],high[,"ERBB2.CNV.binary"],co
l="red")
>
```

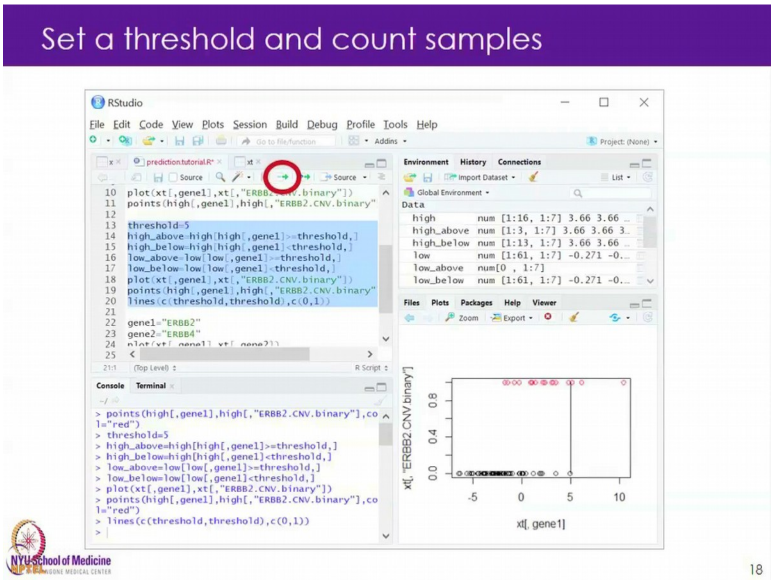
```
Data
  high     num [1:16, 1:7] 3.66 3.66 ...
  low      num [1:61, 1:7] -0.271 -0....
  x        7 obs. of 77 variables
  xt       num [1:77, 1:7] -0.271 -0....
Values
  gene1    "ERBB4"
```

17

So, then we run this and now the y axis will either be 0 or 1 and all the red ones will be up here and that is have high copy number they will have one and the other ones will be 0. So, you remember in my lecture I showed some slides where we had exactly these kinds of plots and then what is we want to do is to find a threshold here that where we have most of the red points above the thresholds and most of the black points below the threshold.

Now, as you see it is impossible to find a perfect thing for this, but let us try just one. So, now, there is a whole section here that is we are going to run, we are going to just define the threshold at 5 because all the black points are below 5. So, we are going to run this and get another plot where we put the line here at 5 and then we see we are going to get these define this low above.

(Refer Slide Time: 11:51)



Set a threshold and count samples

```
plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"

threshold=5
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
low_above=low[low[,gene1]>=threshold,]
low_below=low[low[,gene1]<threshold,]
plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"
lines(c(threshold,threshold),c(0,1))

gene1="ERBB2"
gene2="ERBB4"
plot(xt[,gene1],xt[,gene2])
```

```
> points(high[,gene1],high[,"ERBB2.CNV.binary"],co
l="red")
> threshold=5
> high_above=high[high[,gene1]>=threshold,]
> high_below=high[high[,gene1]<threshold,]
> low_above=low[low[,gene1]>=threshold,]
> low_below=low[low[,gene1]<threshold,]
> plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
> points(high[,gene1],high[,"ERBB2.CNV.binary"],co
l="red")
> lines(c(threshold,threshold),c(0,1))
>
```

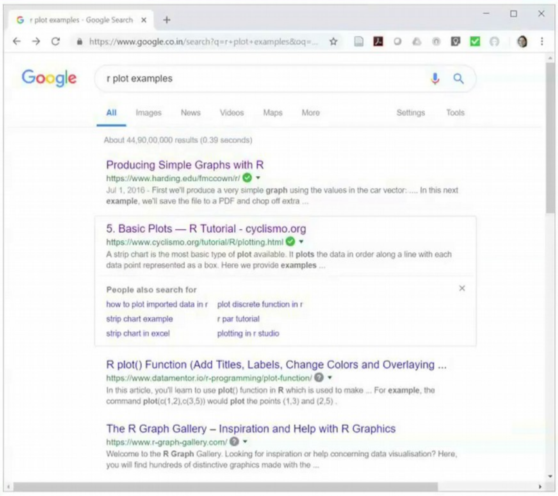
```
Data
  high         num [1:16, 1:7] 3.66 3.66 ...
  high_above   num [1:3, 1:7] 3.66 3.66 3...
  high_below   num [1:13, 1:7] 3.66 3.66 ...
  low          num [1:61, 1:7] -0.271 -0....
  low_above    num[0 , 1:7]
  low_below    num [1:61, 1:7] -0.271 -0....
```

18

So, the num that is the number of black points above the line which is 0 and low below which is 61 which is all the black, then we have from the high group above we have only 3 which is a bit disappointing 3 out of 16 and that were classified right and then 13 is below. So, this is not a good choice of a thresholds but. So, what I want you to do is now find a good threshold oh yeah.

(Refer Slide Time: 12:43)

Plot



r plot examples - Google Search
https://www.google.co.in/search?q=r+plot+examples&oq=...
Google
All
Images
News
Videos
Maps
More
Settings
Tools

r plot examples

About 44,90,00,000 results (0.39 seconds)

Producing Simple Graphs with R
https://www.harding.edu/fmccown/r/
Jul 1, 2016 - First we'll produce a very simple **graph** using the values in the car vector: .... In this next **example**, we'll save the file to a PDF and chop off extra ...

5. Basic Plots — R Tutorial - cyclismo.org
https://www.cyclismo.org/tutorial/R/plotting.html
A strip chart is the most basic type of **plot** available. It **plots** the data in order along a line with each data point represented as a box. Here we provide **examples** ...

People also search for

how to plot imported data in r	plot discrete function in r
strip chart example	r par tutorial
strip chart in excel	plotting in r studio

R plot() Function (Add Titles, Labels, Change Colors and Overlaying ...
https://www.datamentor.io/r-programming/plot-function/
In this article, you'll learn to use **plot**() function in **R** which is used to make ... For **example**, the command **plot**(c(1,2),c(3,5)) would **plot** the points (1,3) and (2,5) .

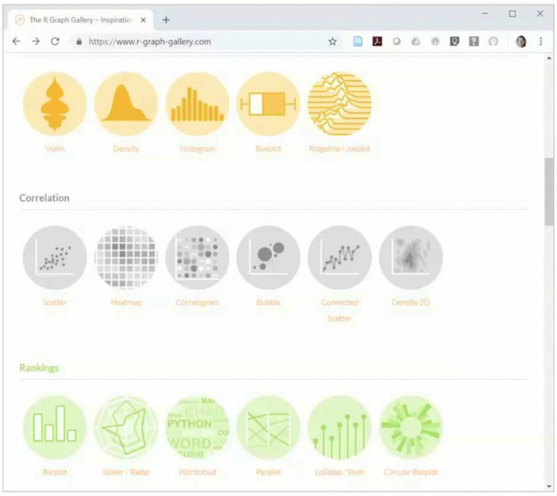
The R Graph Gallery – Inspiration and Help with R Graphics
https://www.r-graph-gallery.com/
Welcome to the **R Graph** Gallery. Looking for inspiration or help concerning data visualisation? Here, you will find hundreds of distinctive graphics made with the ...

13

So, this is another Google search I did is to how to plot with R and you get a lot of different answers.

(Refer Slide Time: 12:53)

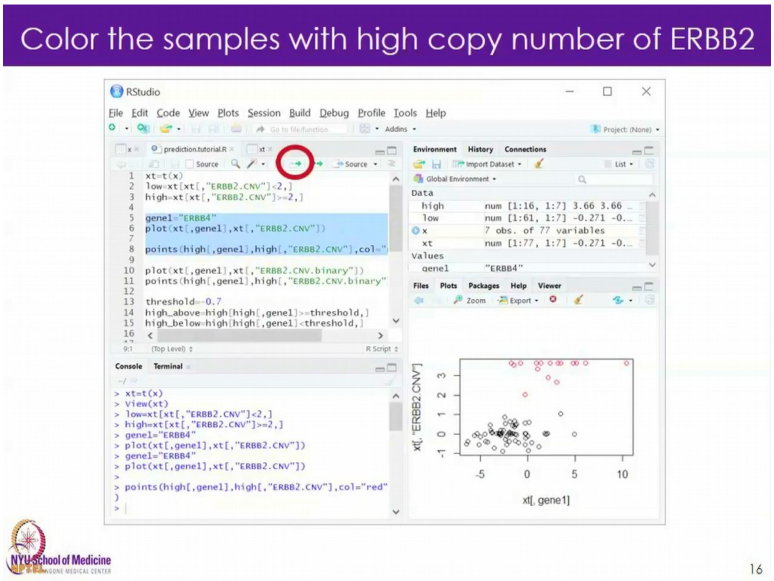
Plot



14

So, and one very nice thing is that they have a graph gallery that shows different types of graphs. So, you can look at the graphs and see I want to do that is kind of graph, then they have the code there you can just copy and paste and modify it a little bit.

(Refer Slide Time: 13:11)



Color the samples with high copy number of ERBB2

```
xt=t(x)
low=xt[xt[,"ERBB2.CNV"]<2,]
high=xt[xt[,"ERBB2.CNV"]>=2,]

gene1="ERBB4"
plot(xt[,gene1],xt[,"ERBB2.CNV"])

points(high[,gene1],high[,"ERBB2.CNV"],col="

plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"

threshold=-0.7
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
```

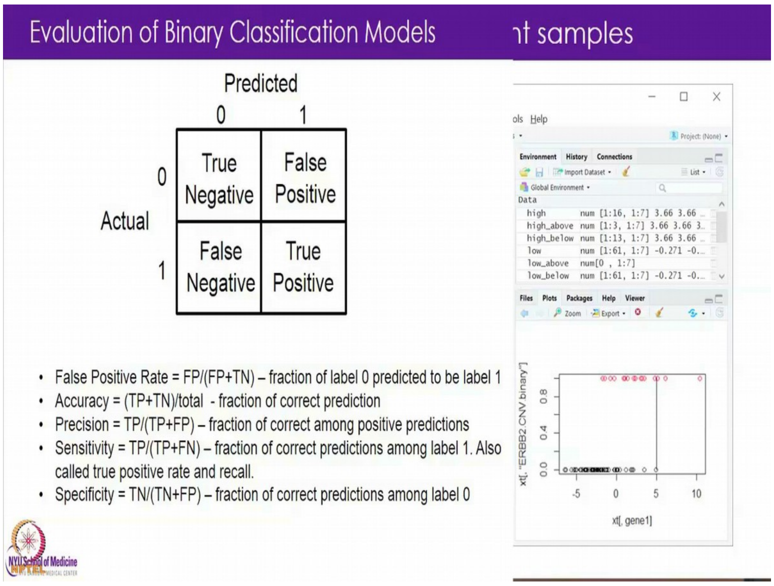
```
> xt=t(x)
> View(xt)
> low=xt[xt[,"ERBB2.CNV"]<2,]
> high=xt[xt[,"ERBB2.CNV"]>=2,]
> gene1="ERBB4"
> plot(xt[,gene1],xt[,"ERBB2.CNV"])
> gene1="ERBB4"
> plot(xt[,gene1],xt[,"ERBB2.CNV"])
>
> points(high[,gene1],high[,"ERBB2.CNV"],col="red"
)
>
```

```
Data
  high        num [1:16, 1:7] 3.66 3.66 ...
  low         num [1:61, 1:7] -0.271 -0....
  x           7 obs. of 77 variables
  xt          num [1:77, 1:7] -0.271 -0....
Values
  gene1       "ERBB4"
```

16

And so, these will just now show the exactly what we have done.

(Refer Slide Time: 13:21)



Evaluation of Binary Classification Models

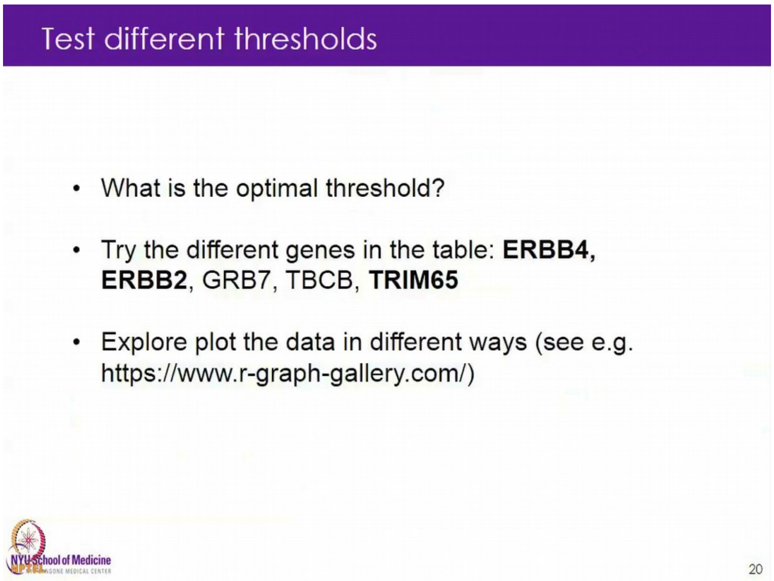
		Predicted 0	Predicted 1
Actual	0	True Negative	False Positive
	1	False Negative	True Positive

- False Positive Rate = FP/(FP+TN) – fraction of label 0 predicted to be label 1
- Accuracy = (TP+TN)/total - fraction of correct prediction
- Precision = TP/(TP+FP) – fraction of correct among positive predictions
- Sensitivity = TP/(TP+FN) – fraction of correct predictions among label 1. Also called true positive rate and recall.
- Specificity = TN/(TN+FP) – fraction of correct predictions among label 0

```
Data
  high        num [1:16, 1:7] 3.66 3.66 ...
  high_above  num [1:3, 1:7] 3.66 3.66 3...
  high_below  num [1:13, 1:7] 3.66 3.66 ...
  low         num [1:61, 1:7] -0.271 -0....
  low_above   num[0 , 1:7]
  low_below   num [1:61, 1:7] -0.271 -0....
```

And then if you remember one of the slides what we have done now is actually may a table like this. So, we had our actual groups which are the 0 would correspond to the lower group, the copy num ERBB2 copy numbers low and the one is when the copy number is high and then the predicted group is when we set or threshold now at 5. So, we get this table.

(Refer Slide Time: 13:57)



Test different thresholds

- What is the optimal threshold?
- Try the different genes in the table: **ERBB4, ERBB2**, GRB7, TBCB, **TRIM65**
- Explore plot the data in different ways (see e.g. https://www.r-graph-gallery.com/)

NYU School of Medicine

20

So, what I want you to do now is to change the threshold and just find what you think is a good threshold to choose for this example and this is for ERBB 4 then I want you to redo it for ERBB 2 protein values which of course, are going to be better that is what we expect, but also maybe try TRIM5 TRIM 65 and see find that for each of these the best threshold and also if you have more time you can explore the to plot the data in different ways by looking here at gallery yes.

So,. So, then every time you try and new thresholds you get a table like this. So, maybe fill out the table if it on paper and pen and forever and try a few and then move on to from ERBB 4 to ERBB 2 which is going to give a better result this because there is going to be more separation not surprisingly because the copy number effects the same gene more but as we saw it is before I show that ERBB 4 does not have a copy number change, but it is still affected by the copy number change in indirect way and so, on yeah.

So, that is what I want you to do and also TRIM 65 is negatively correlated, but not that strongly. So, that will be then the red and the black will switch it will be the other way around, but yeah. So, that is just explore and think about definitely think about what does it mean that the threshold is optimal. So, what we want to we want to get the values of these 4 values for every threshold.

And so and we have calculated these for a copy number high the ones that are above the threshold the for a copy number high advance that are below the threshold. So, can anyone tell me where. So, there are 3 that are above the threshold these 3 here of the copy number high, 3 of the red points. So, where does that3 go in the in the in the confusion matrix which we have 4 different positions where would we put the number 3 here?

Student: may be in the true positives

Yes true positives. So, we put the 3 here and then we have the copy number high below which are the 13 and what are those.

Student: false negatives

Yes false negatives. So, we put 3 it in through positives and thirteen in false negatives and then we have the copy number low which then are the other two. So, we have the low above which are 0 the where does that end up.

Student: false positives.

True negative yes no I am sorry false false positive sorry yes. So, those are the false. So, we do not have any false positives and then the 61 the low below or the 61 true negatives and always with these we want as many as possible on the diagonal and as few as possible off the diagonal. Let us say that we have a very in additional test that we do for all the positive on that is rather quick and easy then we do not worry so, much about false positives then maybe we can allow more false positive, but we are worried about the false negatives.

So, it is not always, but the taking the sum is definitely one option. So, the other example is let us say the consequence of having a false positive is that the PhD student will spend 5 years investigating this protein and then we of course, want to have very few false positives. So, it is it always depends on the situation.

So, we had the high the copy number high that are above the threshold that is our true positives and then we have the copy number high that are below the threshold those was are the false negatives; then we have the copy number low above those are the false positives.

So, in this case with this initial threshold we had no false positives and then the copy number low that are below or the true negatives you should try to find is to minimize the sum of the false positives and the false negatives, that I think for this case we could call as the optimal case, but remember that is that is not the general statement that is it is not vary from case to case.

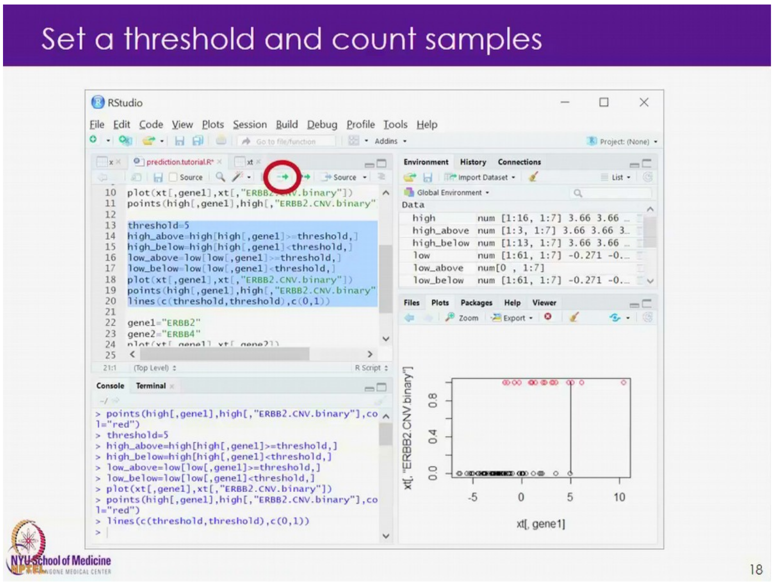
So, what you have done now is actually optimized a very simple machine learning algorithm. It is probably the simplest machine learning algorithm one can imagine, but it is still an algorithm that is where we separate try to predict what is positive and negative. And so, the what I want you to people talk a lot about machine learning and its, but and there is a lot of hype but there is no magic to it this is; it is I mean this is really when if you understand this the rest is just tricks to do it better.

Student: Sir. So, when you are calculating the threshold locking a threshold now based on that table if you take that precision value and I want to get the most precise values can that be used as the point for calculating on optimization?

Yeah, yeah that is a yeah if you want to do that definitely yeah.

So, I think this will kind of show this will kind of show the power of R. So, that is in. So, what David and I will show you all kind of showing of the power of R. So, David went through how you can manually.

(Refer Slide Time: 21:27)



Set a threshold and count samples

```
plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"

threshold=5
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
low_above=low[low[,gene1]>=threshold,]
low_below=low[low[,gene1]<threshold,]
plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"
lines(c(threshold,threshold),c(0,1))

gene1="ERBB2"
gene2="ERBB4"
```

```
> points(high[,gene1],high[,"ERBB2.CNV.binary"],co
l="red")
> threshold=5
> high_above=high[high[,gene1]>=threshold,]
> high_below=high[high[,gene1]<threshold,]
> low_above=low[low[,gene1]>=threshold,]
> low_below=low[low[,gene1]<threshold,]
> plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
> points(high[,gene1],high[,"ERBB2.CNV.binary"],co
l="red")
> lines(c(threshold,threshold),c(0,1))
>
```

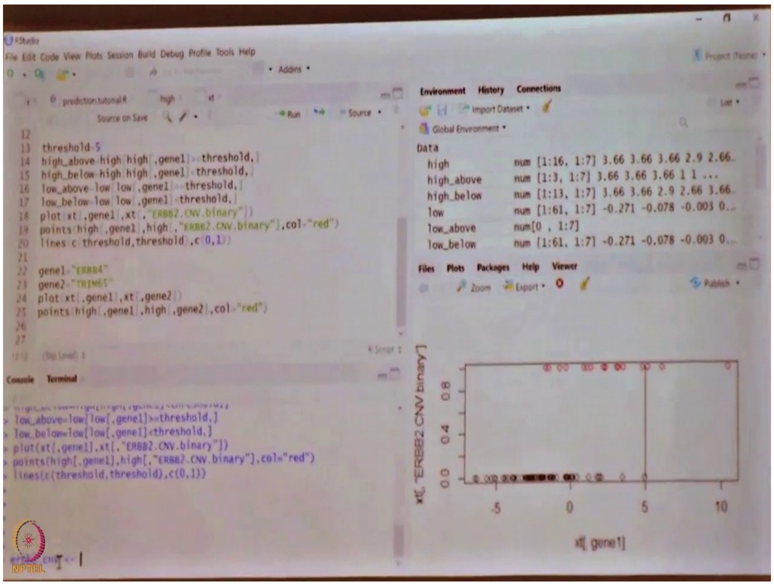
```
Data
  high         num [1:16, 1:7] 3.66 3.66 ...
  high_above   num [1:3, 1:7] 3.66 3.66 3...
  high_below   num [1:13, 1:7] 3.66 3.66 ...
  low          num [1:61, 1:7] -0.271 -0....
  low_above    num[0 , 1:7]
  low_below    num [1:61, 1:7] -0.271 -0....
```

Fill out that that confusion matrix orthe table for all programming languages like R that comes with the fact that you can fill that the table with single line of code and so, now, if you have to optimize this we have to know the table let us say 10 lines if you do it by hands, some may take you a month or maybe you are faster but still it will take you a lot time but you can very quickly do it with 10 lines of code , and if you learn this thing for a for loop you can do it with 2 lines of code and it all let you go through everything if you want to calculate precision or you want to calculate accuracy or sensitivity specificity there are formulas based on that table.

And R will automatically do those for you there are functions that calculate all those things. And we can do pretty much everything and you can even plot all the values as you are threshold changes and then see where the maximum or minimum occurs. So, that is the power of programming. So, I think David and I are basically introducing to you to what it can do but it provides power of those wave beyond brought excel or people can do and like David said it is all do by hand, but if you do it by code it is much faster one thing. The more important thing is the next and few each later you are trying to figure out what exactly you did it would be there in code if you did it in excel what ever you say is what we did before.

So, re reproducibility in terms of or remembering what you did in showing others what you did and in the publication making it available to others is made much easier if you can code I think that is where the power of programming comes. So, I will just show you how to create that table automatically and then I let you and Google deal with the best. To do this to do the table you need to have 2 vectors one that gives you the high and low based on copy number and another one gives you the high and low based on this threshold that we have chosen and so, when you have these 2 vectors you can call a function them create the table for you

(Refer Slide Time: 24:03)



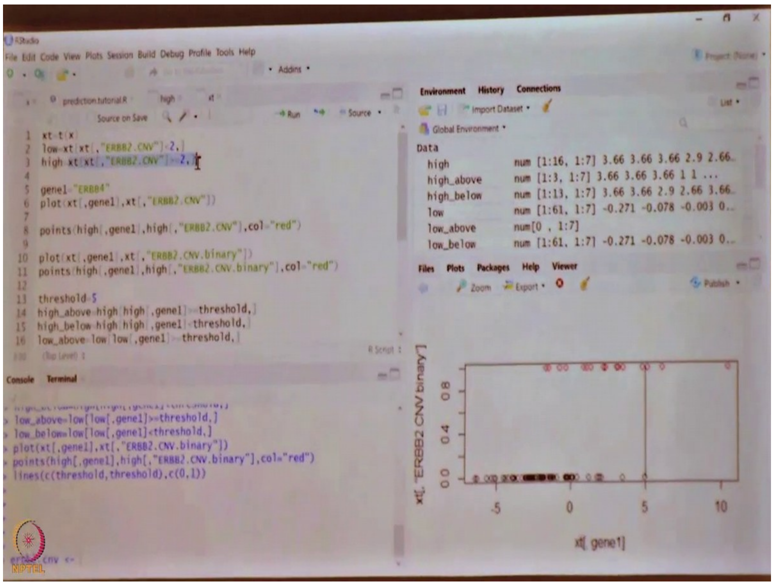
```
threshold=5
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
low_above=low[low[,gene1]>=threshold,]
low_below=low[low[,gene1]<threshold,]
plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"],col="red")
lines(c(threshold,threshold),c(0,1))

gene1="ERBB4"
gene2="TRIM65"
plot(xt[,gene1],xt[,gene2])
points(high[,gene1],high[,gene2],col="red")
```

```
> low_above=low[low[,gene1]>=threshold,]
> low_below=low[low[,gene1]<threshold,]
> plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
> points(high[,gene1],high[,"ERBB2.CNV.binary"],col="red")
> lines(c(threshold,threshold),c(0,1))
> erbb2_cnv <- 
```

```
Data
  high         num [1:16, 1:7] 3.66 3.66 3.66 2.9 2.66...
  high_above   num [1:3, 1:7] 3.66 3.66 3.66 1 1 ...
  high_below   num [1:13, 1:7] 3.66 3.66 2.9 2.66 3.66...
  low          num [1:61, 1:7] -0.271 -0.078 -0.003 0...
  low_above    num[0 , 1:7]
  low_below    num [1:61, 1:7] -0.271 -0.078 -0.003 0...
```

So, the first vector threshold was 5 and then create another vector that says when it is when that gene is high and when that gene is low and once you have these 2 vectors that is the single line of code for the data plotting.

(Refer Slide Time: 24:31)



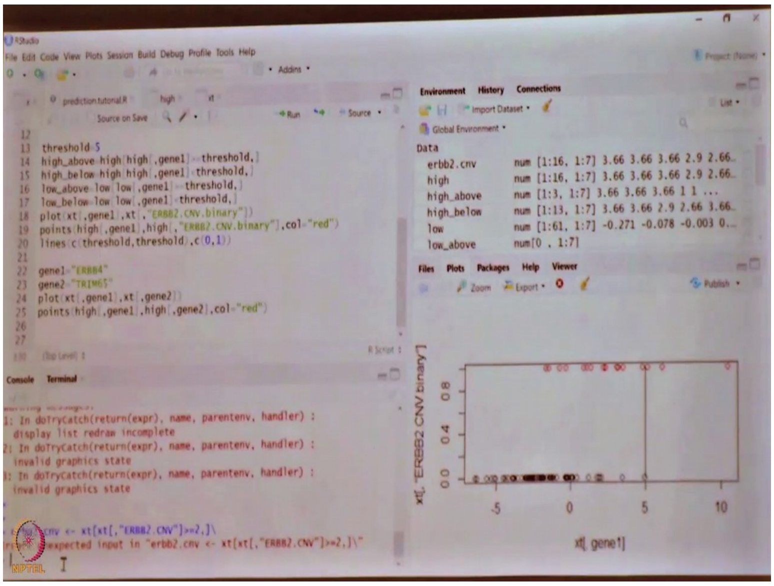
```
xt=t(x)
low=xt[xt[,"ERBB2.CNV"]<2,]
high=xt[xt[,"ERBB2.CNV"]>=2,]

gene1="ERBB4"
plot(xt[,gene1],xt[,"ERBB2.CNV"])

points(high[,gene1],high[,"ERBB2.CNV"],col="red")

plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"],col="red")

threshold=5
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
low_above=low[low[,gene1]>=threshold,]
```

```
> low_above=low[low[,gene1]>=threshold,]
> low_below=low[low[,gene1]<threshold,]
> plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
> points(high[,gene1],high[,"ERBB2.CNV.binary"],col="red")
> lines(c(threshold,threshold),c(0,1))
> erbb2_cnv <-
```

```
Data
  high         num [1:16, 1:7] 3.66 3.66 3.66 2.9 2.66...
  high_above   num [1:3, 1:7] 3.66 3.66 3.66 1 1 ...
  high_below   num [1:13, 1:7] 3.66 3.66 2.9 2.66 3.66...
  low          num [1:61, 1:7] -0.271 -0.078 -0.003 0...
  low_above    num[0 , 1:7]
  low_below    num [1:61, 1:7] -0.271 -0.078 -0.003 0...
```

So, basically what you need to do is you have to say high is 1. So, you just stick that and just copy it over ok.

(Refer Slide Time: 24:43)



```r
threshold<-5
high_above<-high[high[,gene1]>=threshold,]
high_below<-high[high[,gene1]<threshold,]
low_above<-low[low[,gene1]>=threshold,]
low_below<-low[low[,gene1]<threshold,]
plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"],col="red")
lines(c(threshold,threshold),c(0,1))

gene1<-"ERBB4"
gene2<-"TRIM65"
plot(xt[,gene1],xt[,gene2])
points(high[,gene1],high[,gene2],col="red")
```

```
1: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
2: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
3: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
>
> cnv <- xt[xt[,"ERBB2.CNV"]>=2,]\
rror: expected input in "erbb2.cnv <- xt[xt[,"ERBB2.CNV"]>=2,]\"
```

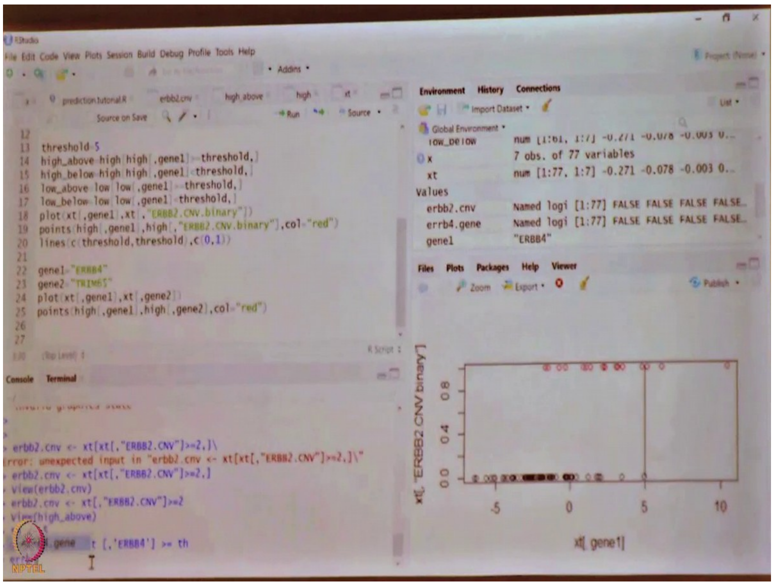
```
Data
  erbb2.cnv      num [1:16, 1:7] 3.66 3.66 3.66 2.9 2.66…
  high           num [1:16, 1:7] 3.66 3.66 3.66 2.9 2.66…
  high_above     num [1:3, 1:7] 3.66 3.66 3.66 1 1 ...
  high_below     num [1:13, 1:7] 3.66 3.66 2.9 2.66 3.66…
  low            num [1:61, 1:7] -0.271 -0.078 -0.003 0.…
  low_above      num[0 , 1:7]
```

So, I now have a variable called erbb2 cnv

I just.

So, I want to mark things that are more than 2 as 1 and. So, so this statement is saying there is this matrix called xt and.

(Refer Slide Time: 25:35)



```r
threshold<-5
high_above<-high[high[,gene1]>=threshold,]
high_below<-high[high[,gene1]<threshold,]
low_above<-low[low[,gene1]>=threshold,]
low_below<-low[low[,gene1]<threshold,]
plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"],col="red")
lines(c(threshold,threshold),c(0,1))

gene1<-"ERBB4"
gene2<-"TRIM65"
plot(xt[,gene1],xt[,gene2])
points(high[,gene1],high[,gene2],col="red")
```

```
>
>
> erbb2.cnv <- xt[xt[,"ERBB2.CNV"]>=2,]\
rror: unexpected input in "erbb2.cnv <- xt[xt[,"ERBB2.CNV"]>=2,]\"
> erbb2.cnv <- xt[xt[,"ERBB2.CNV"]>=2,]
> View(erbb2.cnv)
> erbb2.cnv <- xt[,"ERBB2.CNV"]>=2
> View(high_above)
> .gene    t [,'ERBB4'] >= th
```

```
  low_below      num [1:61, 1:7] -0.271 -0.078 -0.003 0.…
  x              7 obs. of 77 variables
  xt             num [1:77, 1:7] -0.271 -0.078 -0.003 0.…
Values
  erbb2.cnv      Named logi [1:77] FALSE FALSE FALSE FALSE…
  errb4.gene     Named logi [1:77] FALSE FALSE FALSE FALSE…
  gene1          "ERBB4"
```

I am picking the erbb 2 cnv column and if that column is greater than or equal to 2, then the Boolean value is true in other words whenever that is true for any sample, then that will have a 1 otherwise it will have a 0. So, it is just a different way of representing what David had before. So, if you look here. So, now, you can see erbb 2 cnv is logical and it has false false false true and so, far.

So, for every sample it will say whether it was greater than 2 or less than 2. So, it now you have taken values and discretized it. So, it has become true or false I will do the same for the other I am going to set my threshold to 5 I will call it th. So, one of the tricks that R programmers use is to how names that are really small. So, you do not have to type too much when you are writing code.

But that is not a good idea when you are writing code to keep this is like just to try things out when you are writing code to publish if you have th then you do not know whether it was a what it meant. So, then you need a longer name, but when you are just trying things out it helps to how smaller names and so, now, I am going to say erbb 4.

Student: What does that arrow means?

No, the error happened because I had an extra xt here.

Student: no the arrow arrow.

Or the arrow means get the result and put it into some name. So, what I am saying here is. So, this line is basically saying I have the number 5, I want to call it th. So, that way when I write code using th I can go on change it to 3 and then next time I can run the same code and it will do it for 3 instead of 5.

Student: So, I will use the equal to sign.

Equal to sign is also the same.

Actually equal to is a more newer thing they would not allow equal to in R before.

I think, but the later versions allow equal to, but I am used to doing the less than minus is the same as equal to.

Student: Yeah.

So, I will used the equal to because I do not really know R.

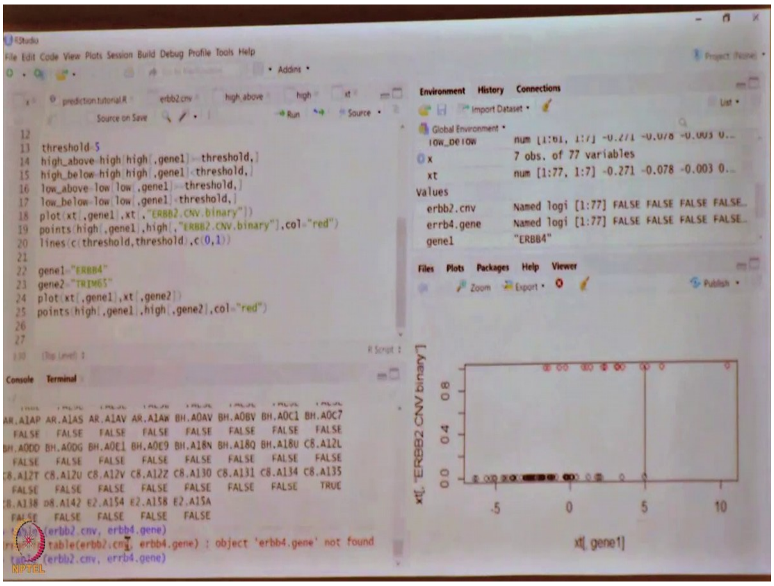
In many languages it is equal to.

Yeah.

And all I think realize that people are getting confused. So, only recently they have allowed the equal to sign till than it was the less than minus. So, that shows that I am I am a little old.

So, for the erbb 4 gene I am going to say I am going to take the array xt and I want the gene ERBB 4 and I want this to be greater than or equal to threshold in order for it to be called high. So, now, for ERBB 4 gene again you see here there are 77 things and they are all either true false or some true or false. So, you can even look at it by just typing the name here. So, now, we save the result in ERBB 4 dot gene when.

(Refer Slide Time: 28:41)



```
threshold<-5
high_above<-high[high[,gene1]>=threshold,]
high_below<-high[high[,gene1]<threshold,]
low_above<-low[low[,gene1]>=threshold,]
low_below<-low[low[,gene1]<threshold,]
plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"],col="red")
lines(c(threshold,threshold),c(0,1))

gene1<-"ERBB4"
gene2<-"TRIM65"
plot(xt[,gene1],xt[,gene2])
points(high[,gene1],high[,gene2],col="red")
```

```
AR.A1AP AR.A1AS AR.A1AV AR.A1AW BH.A0AV BH.A0BV BH.A0C1 BH.A0C7
  FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE
BH.A0DD BH.A0DG BH.A0E1 BH.A0E9 BH.A18N BH.A18Q BH.A18U C8.A12L
  FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE
C8.A12T C8.A12U C8.A12V C8.A12Z C8.A130 C8.A131 C8.A134 C8.A135
  FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE    TRUE
C8.A138 D8.A142 E2.A154 E2.A158 E2.A15A
  FALSE   FALSE   FALSE   FALSE   FALSE
> table(erbb2.cnv, erbb4.gene)
Error in table(erbb2.cnv, erbb4.gene) : object 'erbb4.gene' not found
> table(erbb2.cnv, errb4.gene)
```

```
low_below    num [1:61, 1:7] -0.271 -0.078 -0.003 0...
x            7 obs. of 77 variables
xt           num [1:77, 1:7] -0.271 -0.078 -0.003 0...
Values
erbb2.cnv    Named logi [1:77] FALSE FALSE FALSE FALSE...
errb4.gene   Named logi [1:77] FALSE FALSE FALSE FALSE...
gene1        "ERBB4"
```

So, you can see for which sample it is true for which sample it is false it has the whole thing encoded in that one name.

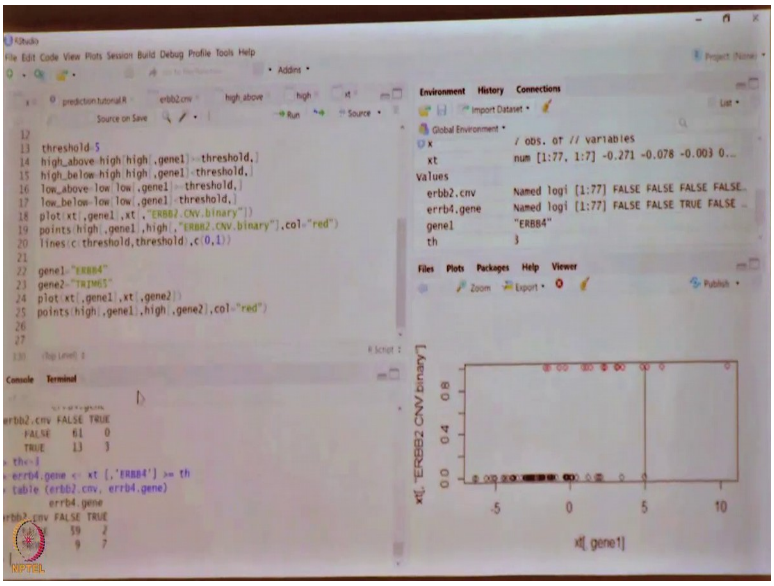
So, now all we need to do to get your table that you did manually before is to say I want a table that compares erbb 2 copy number and erbb 4 the gene what did I called.

Student: So, actually I think that erbb.

Er yeah I spelt it wrong the first time. So, these are things that you have to keep in mind. So, many of your had issues when you load at the table for example, if you did not remove the column that said genes and make it the row name then when you do the transpose of the matrix, the transpose loads only when you have a real numbered matrix.

So, there are all these perks and debugging that you need to learn along the way, it is not like as straightforward as one would think because the error that comes out does not really tell you what is happening or what is wrong. So, you need to kind of logically think through and do it, but you do it a couple of times you will you will get it. So, there is your table.

(Refer Slide Time: 29:49)



```
threshold=5
high_above=high[high[,gene1]>=threshold,]
high_below=high[high[,gene1]<threshold,]
low_above=low[low[,gene1]>=threshold,]
low_below=low[low[,gene1]<threshold,]
plot(xt[,gene1],xt[,"ERBB2.CNV.binary"])
points(high[,gene1],high[,"ERBB2.CNV.binary"],col="red")
lines(c(threshold,threshold),c(0,1))

gene1="ERBB4"
gene2="TRIM65"
plot(xt[,gene1],xt[,gene2])
points(high[,gene1],high[,gene2],col="red")
```

```
         errb4.gene
erbb2.cnv FALSE TRUE
    FALSE    61    0
    TRUE     13    3
> th<-3
> errb4.gene <- xt [,'ERBB4'] >= th
> table (erbb2.cnv, errb4.gene)
         errb4.gene
erbb2.cnv FALSE TRUE
                59    2
                 9    7
```

```
x            7 obs. of 77 variables
xt           num [1:77, 1:7] -0.271 -0.078 -0.003 0...
Values
  erbb2.cnv    Named logi [1:77] FALSE FALSE FALSE FALSE...
  errb4.gene   Named logi [1:77] FALSE FALSE TRUE FALSE ...
  gene1        "ERBB4"
  th           3
```

So, you can see the true false here is the predicted and the true false on the on the rows or the actual. So, in our case the this is for the copy number and this is for the gene and so, you can this is the table that you manually created by counting dots before. So, now, let us say I want to change the threshold to 3 I said threshold to 3 and then I just repeat the same command I had before.

And then I just plot the table again. So, now, if you set it to 3 this is the result you will get and so, on and there is a construct called a for loop where you can say start with my threshold of 1 and go up to 10 and it will calculate this table automatically and print it for values from 1 to 10.

But I thought one should never use for loops in R.

That is true. That is more complicatedSo, there is a way to do it without doing for loops in a single line of code and I think that is when you come for the advanced R course next time.

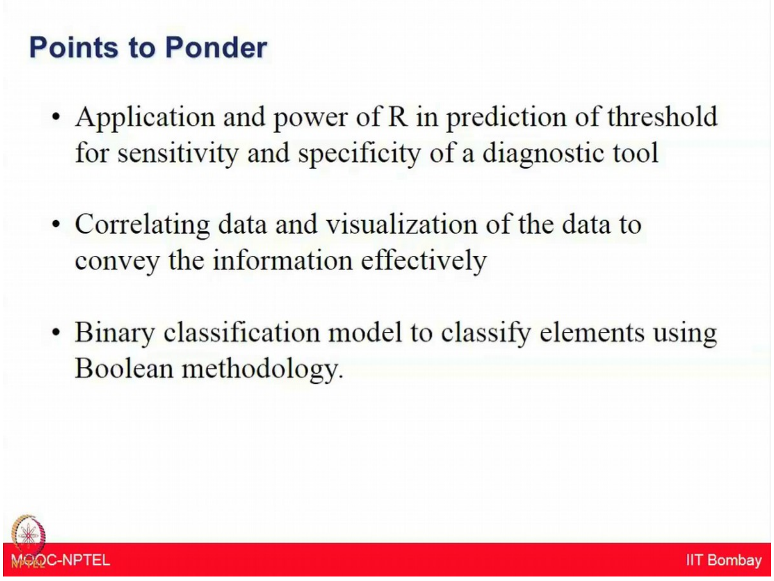
Yes yeah.

That is for yeah. So, you are allowed to use for loops until next year.

Yeah.

But after that is no more.

(Refer Slide Time: 30:57)



Points to Ponder

- Application and power of R in prediction of threshold for sensitivity and specificity of a diagnostic tool
- Correlating data and visualization of the data to convey the information effectively
- Binary classification model to classify elements using Boolean methodology.

MOOC-NPTEL IIT Bombay

I hope this session was informative for you all, where you learned some basics of R followed by the prediction analysis. Doctor Fenyo showed you how to set an optimal threshold and count the sample we have also learned how to colour code the samples that is coming with high copy number. Further he showed you how binary classification model can be used to classify the elements of a given set into 2 groups.

There are many matrices that can be used to measure the performance of a classifier or predictor. Different fields have provided different preferences for a specific matrices due to different goals. For example, for the clinical applications the sensitivity and a specificity are often used. I will recommend you all to learn some basics of R that will help you a lot for doing the statistical analysis and prediction analysis very easily.

There are many publically available resources where various software various codes are already available, but you at least need some basic ability to run those codes in r. In the next supplementary lecture, we will have TA for this course Deeptarup Biswas who will discuss about pathway enrichment and network analysis.

Thank you.