

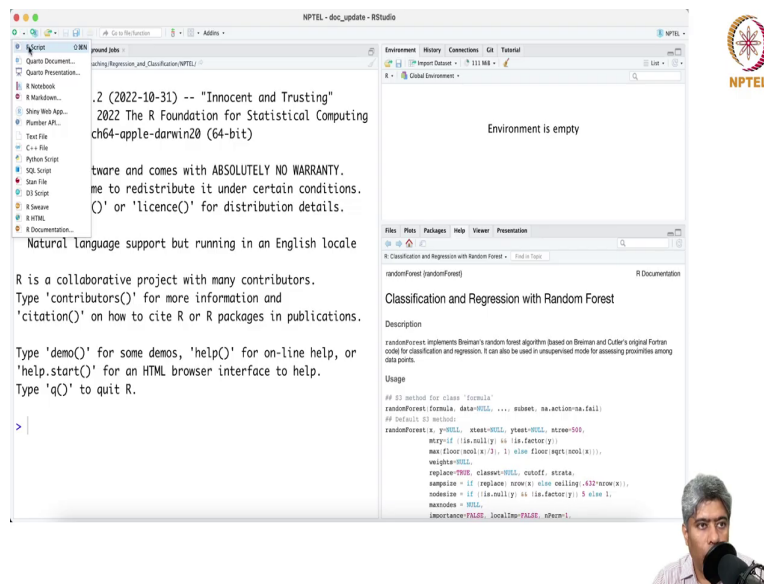
Predictive Analytics - Regression and Classification
Prof. Sourish Das
Department of Mathematics
Mathematical Institute, Chennai

Lecture - 52

Hands on with R: Implement Tree Regression and Random Forest with Simulated Data

Hello all, welcome back to the lecture 16 Part B. In this video, we are going to do some hands-on with R.

(Refer Slide Time: 00:34)



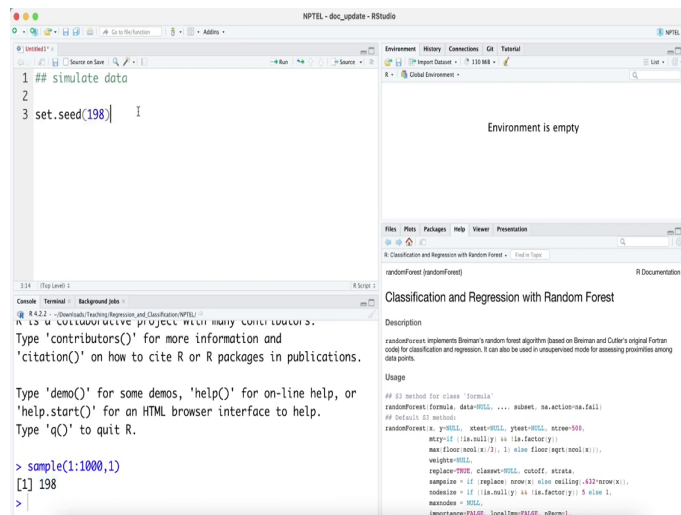
The image shows a screenshot of the RStudio interface. The script editor on the left contains the following R code:

```
## 2 (2022-10-31) -- "Innocent and Trusting"  
## 2022 The R Foundation for Statistical Computing  
## ch64-apple-darwin20 (64-bit)  
  
## Software and comes with ABSOLUTELY NO WARRANTY.  
## You may redistribute it under certain conditions.  
## See the file 'COPYING' for distribution details.  
  
## Natural language support but running in an English locale  
  
## R is a collaborative project with many contributors.  
## Type 'contributors()' for more information and  
## 'citation()' on how to cite R or R packages in publications.  
  
## Type 'demo()' for some demos, 'help()' for on-line help, or  
## 'help.start()' for an HTML browser interface to help.  
## Type 'q()' to quit R.  
  
> |
```

The console window on the right displays the message "Environment is empty". The RStudio logo is visible in the top right corner, and a small inset image of Prof. Sourish Das is in the bottom right corner.

So, let me first start R, start a new script. Ok,

(Refer Slide Time: 00:38)



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

```
1 ## simulate data
2
3 set.seed(198) |
```

The console window shows the output of the code:

```
> sample(1:1000,1)
[1] 198
>
```

The right-hand pane displays the documentation for the `randomForest` package, titled "Classification and Regression with Random Forest". It includes a description of the algorithm and its usage.

Environment: Environment is empty

Files: Files Packages Help Viewer Presentation

R Documentation: randomForest (randomForest)

Classification and Regression with Random Forest

Description

randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing similarities among data points.

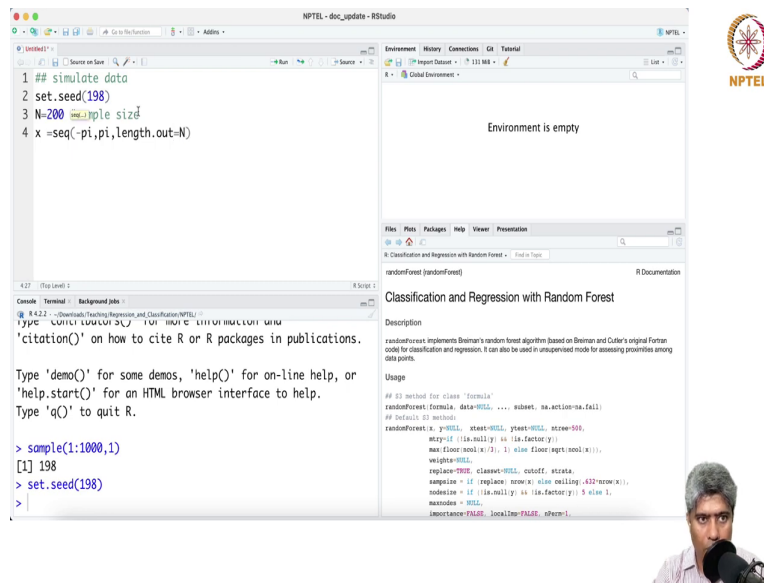
Usage

```
# S3 method for class 'formula'
randomForest(formula, data=NULL, ... subset, na.action=na.fail)
# default S3 method
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
  mtry=if (is.null(y)) is.factor(y)
  else floor(sqrt(ncol(x))),
  max[,log=TRUE],
  replace=TRUE, classwt=NULL, cutoff=NULL,
  sampsize = if (replace) nrow(x) else ceiling(.02*nrow(x)),
  nodesize = if (is.null(y)) is.factor(y) ? 4096 : 1,
  maxnodes = NULL,
  importance=FALSE, localImp=FALSE, sapply=)
```

Now. So, in this hands-on, we will try to understand how the tree structured regression and decision tree and random forest works. So, first and we will do this with simulated data and in the next video, we will work with real life Football data or UK Premier League data, simulated data, simulate data.

So, first I need to set up a seed, set dot seed, say, let me just sample a number between 1 to 1000, just once number is fine, 198 fine.

(Refer Slide Time: 01:35)



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

```
1 ## simulate data
2 set.seed(198)
3 N=200 ## sample size
4 x =seq(-pi,pi,length.out=N)
```

The console window shows the following output:

```
> sample(1:1000,1)
[1] 198
> set.seed(198)
>
```

The right-hand pane displays the help page for the `randomForest` package, titled "Classification and Regression with Random Forest". The description states: "randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing similarities among data points."

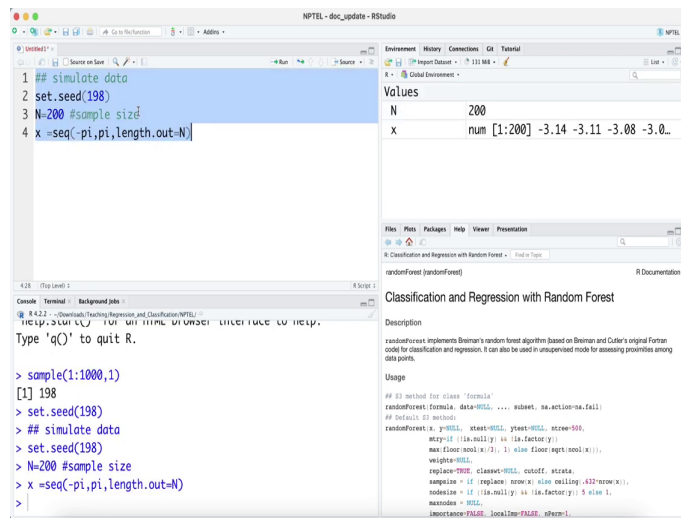
Usage information is provided below the description:

```
## S3 method for class 'formula'
randomForest(formula, data=NULL, ... subset, na.action=na.fail)
## default S3 method
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
            mtry=if (is.null(y)) is.factor(y)
            else floor(sqrt(ncol(x))),
            weights=NULL,
            replace=TRUE, classwt=NULL, cutoff=NULL,
            sampsize = if (replace) nrow(x) else ceiling(.02*nrow(x)),
            nodesize = if (is.null(y)) is.factor(y) ? 5 else 1,
            maxnodes = NULL,
            importance=FALSE, localImp=FALSE, sparam=)
```

The NPTEL logo is visible in the top right corner of the slide, and a small inset image of a man speaking into a microphone is located in the bottom right corner.

So, now once I have this, I will first decide what will be the simulation size, capital N equal to say 200, this is my simulation sample size, this is going to our sample size. We can play with the detector. So, first what I am going to do, I am going to create sequence of number between minus pi and the length dot out equal to N alright.

(Refer Slide Time: 02:22)



The screenshot shows the RStudio interface with the following content:

```
## simulate data
1 ## simulate data
2 set.seed(198)
3 N=200 #sample size
4 x =seq(-pi,pi,length.out=N)
```

Values

N	200
x	num [1:200] -3.14 -3.11 -3.08 -3.0...

randomForest (randomForest)

Classification and Regression with Random Forest

Description

randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing similarities among data points.

Usage

```
# S3 method for class "formula"
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
# default S3 method
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
  mtry=if (is.null(y)) is.factor(y)
    max(100, ncol(x)/3), 1) else floor(sqrt(ncol(x))),
  weights=NULL,
  replace=TRUE, classwt=NULL, cutoff=NULL,
  amplicon = if (replace) show %>% summarise(amt=ntree*N),
  nodesize = if (is.null(y)) is.factor(y) %>% summarise(
    maxnodes = NULL,
    importance=FALSE, localImp=FALSE, _other=)
```



(Refer Slide Time: 02:26)

The screenshot shows an RStudio interface with the following components:

- Source Editor:** Contains R code for simulating data:

```
1 ## simulate data
2 set.seed(198)
3 N=200 #sample size
4 x =seq(-pi,pi,length.out=N)
5 z = sin(x)
6 y = x + rnorm(N,mean=0,sd=0.3)
```
- Environment:** Shows the 'Values' table:

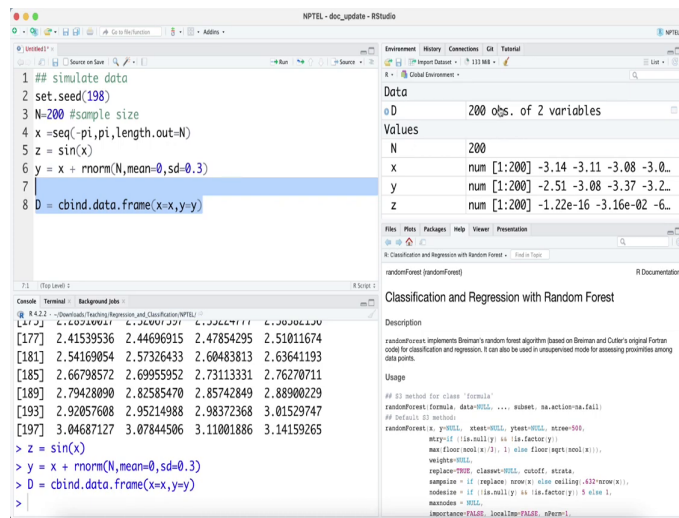
Variable	Value
N	200
x	num [1:200] -3.14 -3.11 -3.08 -3.0...
- Console:** Displays the output of the random forest model, showing a grid of values for each variable:

```
[165] 2.03650981 2.06808361 2.09965740 2.13123120
[169] 2.16280499 2.19437879 2.22595258 2.25752638
[173] 2.28910017 2.32067397 2.35224777 2.38382156
[177] 2.41539536 2.44696915 2.47854295 2.51011674
[181] 2.54169054 2.57326433 2.60483813 2.63641193
[185] 2.66798572 2.69955952 2.73113311 2.76270711
[189] 2.79428090 2.82585470 2.85742849 2.88900229
[193] 2.92057608 2.95214988 2.98372368 3.01529747
[197] 3.04687127 3.07844506 3.11001886 3.14159265
```
- Documentation:** Shows the documentation for the `randomForest` function, including a description and usage instructions.

NPTEL logo is visible in the top right corner. A small inset image of a man speaking into a microphone is located in the bottom right corner of the RStudio window.

So, I have created bunch of numbers between negative pi and pi with equal interval ok. Now, I am going to create z equal to sin x and then also y equal to z plus r norm n, mean equal to 0 and standard deviation equal to 0.3 ok.

(Refer Slide Time: 03:12)



```
1 ## simulate data
2 set.seed(198)
3 N=200 #sample size
4 x =seq(-pi,pi,length.out=N)
5 z = sin(x)
6 y = x + rnorm(N,mean=0,sd=0.3)
7
8 D = cbind.data.frame(x=x,y=y)
```

Environment: Global Environment

Data

#D					200 obs. of 2 variables
N					200
x	num	[1:200]	-3.14	-3.11	-3.08 -3.0...
y	num	[1:200]	-2.51	-3.08	-3.37 -3.2...
z	num	[1:200]	-1.22e-16	-3.16e-02	-6...

randomForest (randomForest)

Classification and Regression with Random Forest

Description

randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points.

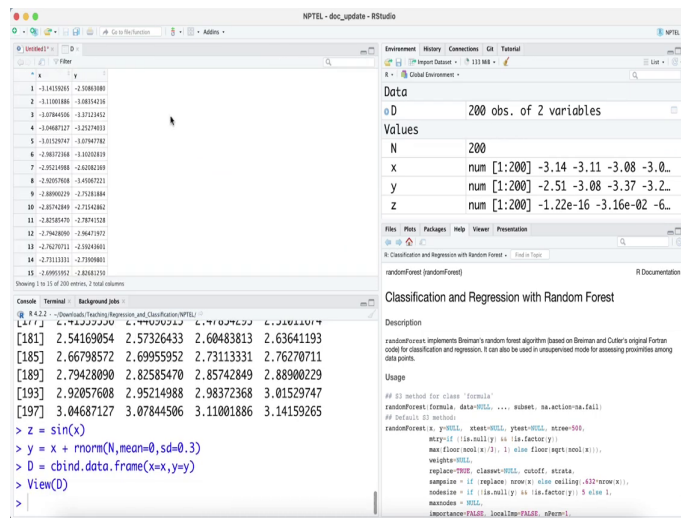
Usage

```
# S3 method for class 'formula'
randomForest(formula, data=NULL, ... subset, na.action=na.fail)
# default S3 method
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
  mtry=if (is.null(y)) 1 else floor(sqrt(ncol(x))),
  max[,col=ncol(x)/2], n) else floor(sqrt(ncol(x))),
  weights=NULL,
  replace=TRUE, classwt=NULL, cutoff=as.na,
  amplicon = if (replace) show(x) else ceiling(-0.02*ncol(x)),
  nodesize = if (is.null(y)) 1 else 1,
  maxnodes = NULL,
  importance=FALSE, localImp=FALSE, sparam=)
```



So, first and then we are going to set up a data set data cbind, cbind dot data dot frame is x equal to x and equal to y.

(Refer Slide Time: 03:35)



The screenshot shows the RStudio interface with a data frame and a random forest model. The data frame has 200 observations and 2 variables. The random forest model is trained on the data and its output is displayed in the console.

id	x	y
1	-1.14159385	-2.50893880
2	-1.11001888	-1.81542426
3	-1.07844356	-3.37231652
4	-1.04686823	-3.21254933
5	-1.01529247	-3.0747782
6	-2.9872164	-1.30282819
7	-2.95214988	-4.82082169
8	-2.92057608	-1.45082221
9	-2.88900229	-2.71211844
10	-2.85742849	-2.71561862
11	-2.82585470	-2.78451128
12	-2.79428090	-2.96471932
13	-2.76270711	-2.35241861
14	-2.73113331	-2.73980801
15	-2.69955952	-2.26281270

```
## R 4.2.2 -- Show the Teaching Regression and Classification NPTEL ##
> x
[1] 2.54169054 2.57326433 2.60483813 2.63641193
[181] 2.66798572 2.69955952 2.73113331 2.76270711
[189] 2.79428090 2.82585470 2.85742849 2.88900229
[193] 2.92057608 2.95214988 2.98372368 3.01529747
[197] 3.04687127 3.07844506 3.11001886 3.14159265
> y = sin(x)
> y = x + rnorm(N, mean=0, sd=0.3)
> D = cbind(data.frame(x=x, y=y))
> View(D)
>
```

Classification and Regression with Random Forest

Description

randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing similarities among data points.

Usage

```
## S3 method for class 'formula'
randomForest(formula, data=NULL, ... subset, na.action=na.fail)
## default S3 method
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
  mtry=if (is.null(y)) 3 else (factor(y) %>%
    max(table(x[1:n()])) else floor(sqrt(table(x))))
  weights=NULL,
  replace=TRUE, classwt=NULL, cutoff=as.numeric(),
  amplicon = if (replace) mtry * else ceiling(.02 * mtry * n()),
  nodesize = if (is.null(y)) 3 else 1,
  maxnodes = NULL,
  importance=FALSE, localImp=FALSE, sparam=)
```



So, here is my data set x and y ok, and now what I am going to do?

(Refer Slide Time: 03:45)

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

```
1 # simulate data
2 set.seed(198)
3 N=200 #sample size
4 x = seq(-pi,pi,length.out=N)
5 z = sin(x)
6 y = x + rnorm(N,mean=0,sd=0.3)
7
8 D = cbind.data.frame(x=x,y=y)
9 max(y)
10
11 plot(NULL,xlim=c(-3.2,3.2),ylim=)
```

The console window shows the execution of these commands:

```
> z = sin(x)
[197] 3.04687127 3.07844506 3.11001886 3.14159265
> y = x + rnorm(N,mean=0,sd=0.3)
> D = cbind.data.frame(x=x,y=y)
> View(D)
> min(y)
[1] -3.450672
> max(y)
[1] 3.394861
>
```

The right pane shows the documentation for the 'randomForest' package, titled 'Classification and Regression with Random Forest'. It includes a description of the algorithm and usage instructions.

I am going to first plot, plot, NULL ok, between xlim will be x limit will be minus 3.2 to 3.2 and ylim will be. So, let us first check what will be the minimum and maximum of, what is the minimum of, minimum of y that we got, negative 3.4 and max is also 3.4.

(Refer Slide Time: 04:46)

```
1 # simulate data
2 set.seed(198)
3 N=200 #sample size
4 x =seq(-pi,pi,length.out=N)
5 z = sin(x)
6 y = x + rnorm(N,mean=0,sd=0.3)
7
8 D = cbind.data.frame(x=x,y=y)
9 max(y)
10
11 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-3.5,3.5),xlab="x"
12      ,ylab="y")
```

Environment | History | Connections | Global | Tutorial

Global Environment

Data

Variable	Class	Length	Summary
D	data.frame	200	obs. of 2 variables
N	numeric	1	200
x	numeric	[1:200]	-3.14 -3.11 -3.08 -3.0...
y	numeric	[1:200]	-2.51 -3.08 -3.37 -3.2...
z	numeric	[1:200]	-1.22e-16 -3.16e-02 -6...

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export | Refresh | Publish

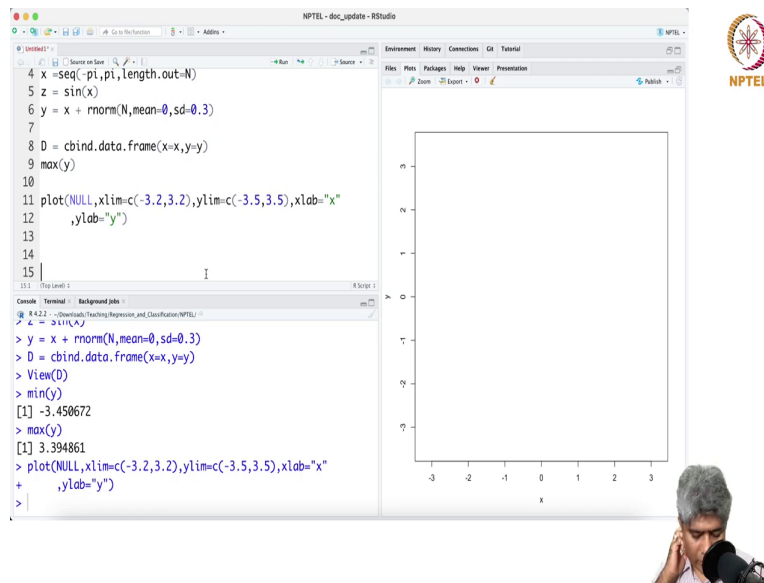
Console

```
R 4.2.2 -- Download/Teaching/Regression_and_Classification/NPTEL
> y = x + rnorm(N,mean=0,sd=0.3)
> D = cbind.data.frame(x=x,y=y)
> View(D)
> min(y)
[1] -3.450672
> max(y)
[1] 3.394861
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-3.5,3.5),xlab="x"
+       ,ylab="y")
>
```

NPTEL

So, we can just take ylim equal to c from maybe 3.5 to 3.5 ok and x lab equal to x and then y lab equal to y. Alright.

(Refer Slide Time: 05:15)



The image shows a screenshot of an RStudio interface. The main editor window contains the following R code:

```
4 x = seq(-pi, pi, length.out=N)
5 z = sin(x)
6 y = x + rnorm(N, mean=0, sd=0.3)
7
8 D = cbind.data.frame(x=x, y=y)
9 max(y)
10
11 plot(NULL, xlim=c(-3.2, 3.2), ylim=c(-3.5, 3.5), xlab="x"
12       , ylab="y")
13
14
15 ]
```

The console window shows the following output:

```
> y = x + rnorm(N, mean=0, sd=0.3)
> D = cbind.data.frame(x=x, y=y)
> View(D)
> min(y)
[1] -3.450672
> max(y)
[1] 3.394861
> plot(NULL, xlim=c(-3.2, 3.2), ylim=c(-3.5, 3.5), xlab="x"
+       , ylab="y")
>
```

The plot window on the right is empty, showing only the axes with labels 'x' and 'y'. The x-axis ranges from -3 to 3, and the y-axis ranges from -3.5 to 3.5. The NPTEL logo is visible in the top right corner of the RStudio window.

(Refer Slide Time: 05:28)

```
4 x = seq(-pi, pi, length.out=N)
5 z = sin(x)
6 y = x + rnorm(N, mean=0, sd=0.3)
7
8 D = cbind.data.frame(x=x, y=y)
9 max(y)
10
11 plot(NULL, xlim=c(-3.2, 3.2), ylim=c(-3.5, 3.5), xlab="x"
12       , ylab="y")
13
14 points(D, col="red", pch=20)
15
```

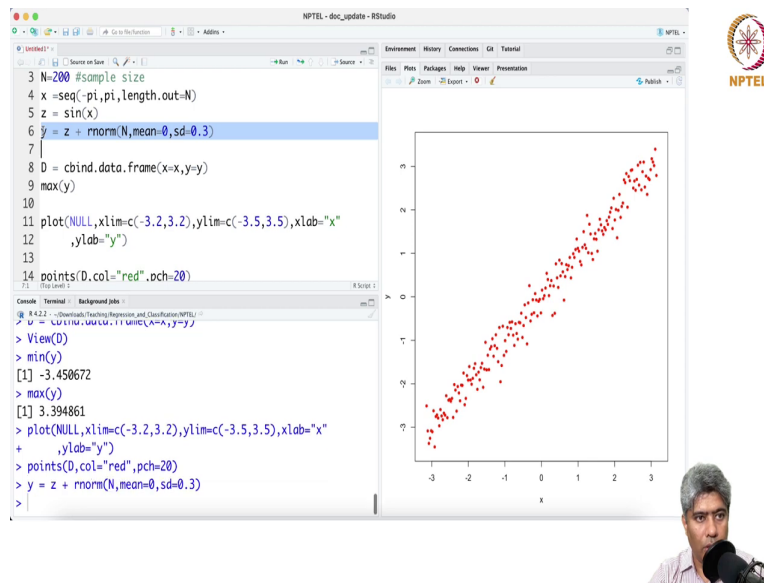
Console Terminal | Background jobs

```
R 4.2.2 -- Download Teaching Regression and Classification NPTEL |
> x = seq(-pi, pi, length.out=N)
> z = sin(x)
> y = x + rnorm(N, mean=0, sd=0.3)
> D = cbind.data.frame(x=x, y=y)
> View(D)
> min(y)
[1] -3.450672
> max(y)
[1] 3.394861
> plot(NULL, xlim=c(-3.2, 3.2), ylim=c(-3.5, 3.5), xlab="x"
+       , ylab="y")
> points(D, col="red", pch=20)
>
```

The scatter plot displays a positive linear relationship between x and y. The x-axis ranges from approximately -3.2 to 3.2, and the y-axis ranges from approximately -3.5 to 3.5. The data points are red circles with a size of 20. The NPTEL logo is visible in the top right corner of the RStudio window. A small video inset of a man speaking is located in the bottom right corner of the slide.

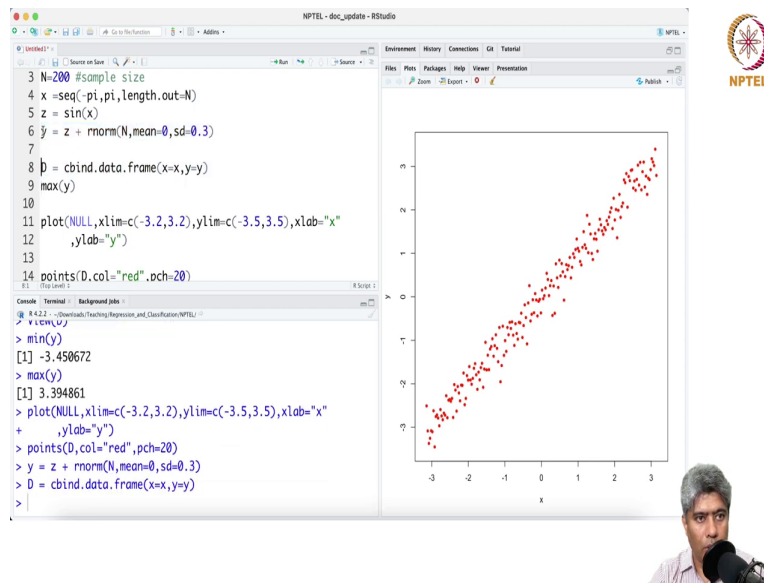
Next, what I am going to do is, I am going to point the, plot the points. Point plot these points. Plot this point, maybe I will just put a color ok. So, color equal to red and p c h equal to 20. So, I made a mistake here.

(Refer Slide Time: 06:06)



I am sorry, I should have been taken this. Actually, I should take, yeah, it was mistake.

(Refer Slide Time: 06:17)



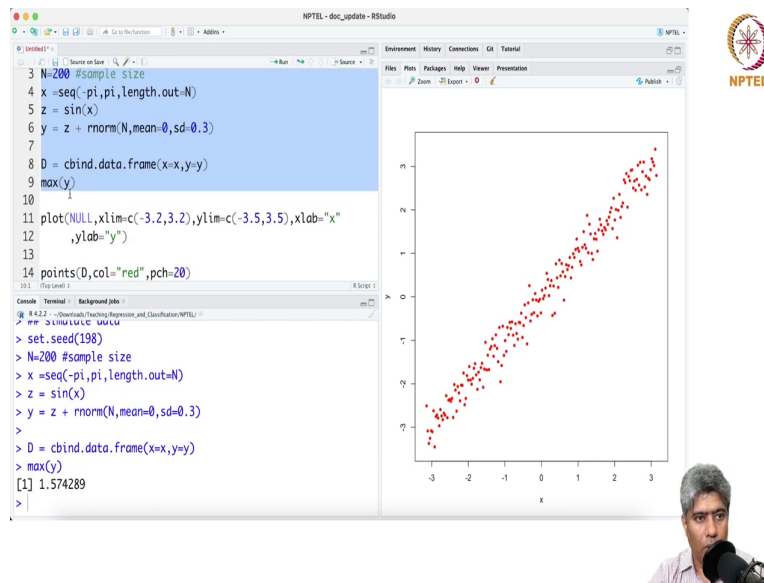
```
3 N=200 #sample size
4 x =seq(-pi,pi,length.out=N)
5 z = sin(x)
6 y = z + rnorm(N,mean=0,sd=0.3)
7
8 D = cbind.data.frame(x=x,y=y)
9 max(y)
10
11 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-3.5,3.5),xlab="x"
12       ,ylab="y")
13
14 points(D,col="red",pch=20)
```

Console Terminal | Background job | R Script |

```
R 4.2.2 --(Download)Teaching-Regression_and-Classification-NPTEL
> v1cwn(y)
> min(y)
[1] -3.450672
> max(y)
[1] 3.394861
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-3.5,3.5),xlab="x"
+       ,ylab="y")
> points(D,col="red",pch=20)
> y = z + rnorm(N,mean=0,sd=0.3)
> D = cbind.data.frame(x=x,y=y)
>
```

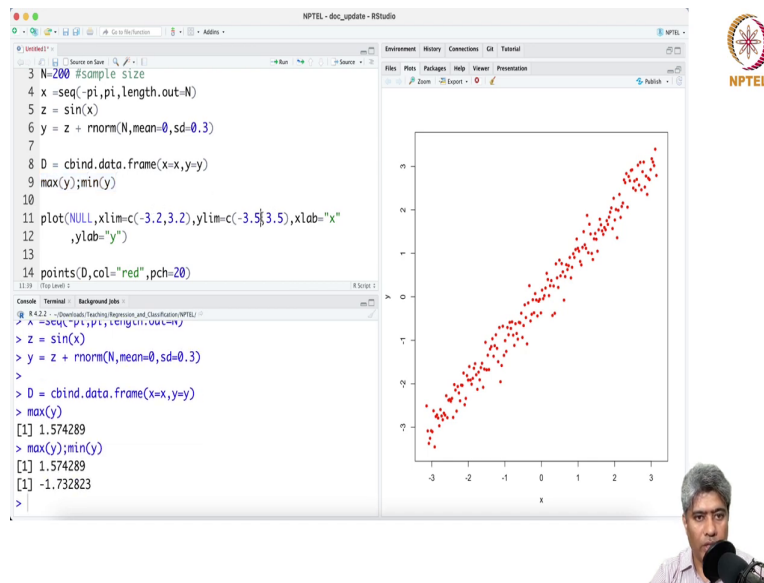
The scatter plot displays a positive linear relationship between x and y. The x-axis ranges from approximately -3.5 to 3.5, and the y-axis ranges from approximately -3.5 to 3.5. The data points are red dots with a size of 20. The NPTEL logo is located in the top right corner of the RStudio window.

(Refer Slide Time: 06:21)



And then; so, actually, yeah, it was a mistake.

(Refer Slide Time: 06:27)





The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains the following R code:

```
3 N=200 #sample size
4 x =seq(-pi,pi,length.out=N)
5 z = sin(x)
6 y = z + rnorm(N,mean=0,sd=0.3)
7
8 D = cbind.data.frame(x=x,y=y)
9 max(y);min(y)
10
11 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-3.5,3.5),xlab="x"
12       ,ylab="y")
13
14 points(D,col="red",pch=20)
```

The console pane shows the execution of the code:

```
R 4.2.2 -- Download/Teaching/Regression_and_Classification/NPTEL
> x = seq(-pi, pi, length.out = N)
> z = sin(x)
> y = z + rnorm(N, mean = 0, sd = 0.3)
>
> D = cbind.data.frame(x = x, y = y)
> max(y)
[1] 1.574289
> max(y); min(y)
[1] 1.574289
[1] -1.732823
>
```

The plot pane displays a scatter plot of red points (pch=20) showing a positive linear correlation between x and y. The x-axis ranges from approximately -3.2 to 3.2, and the y-axis ranges from approximately -3.5 to 3.5. The points are scattered around a diagonal line, indicating a noisy linear relationship.

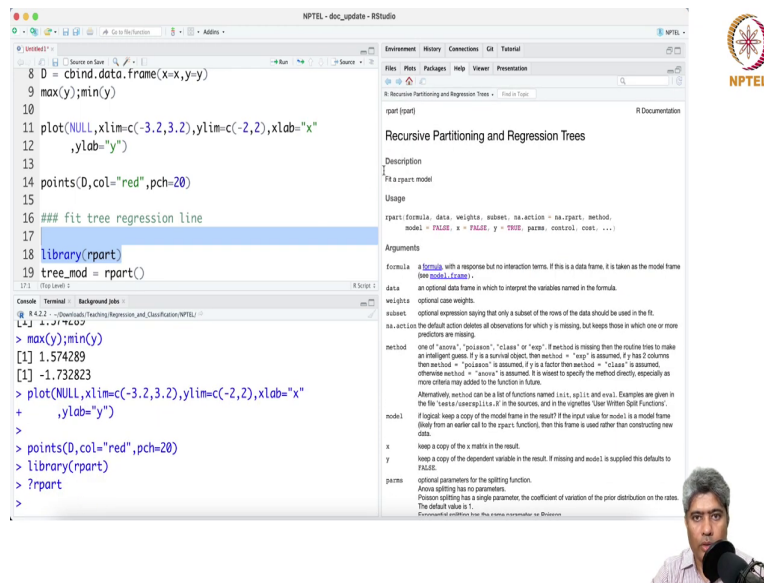


(Refer Slide Time: 06:39)



And minimum of y will be negative 1.5 to, So, I can just take minus 2 to 2. Yeah, so, yeah. Now, this is sort of ok. Alright. So, its basically sample sine x, but with some noise ok. So, its a non-linear complete non-linear relationship between x and y.

(Refer Slide Time: 07:13)



The screenshot shows the RStudio interface with the following content:

```
8 D = cbind.data.frame(x=x,y=y)
9 max(y);min(y)
10
11 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
12 ,ylab="y")
13
14 points(D,col="red",pch=20)
15
16 ## fit tree regression line
17
18 library(rpart)
19 tree_mod = rpart()
```

The console output shows:

```
> max(y);min(y)
[1] 1.574289
[1] -1.732823
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
+ ,ylab="y")
+
> points(D,col="red",pch=20)
> library(rpart)
> ?rpart
```

The right-hand pane displays the documentation for the `rpart` package, titled "Recursive Partitioning and Regression Trees". It includes a description, usage, arguments, and method details.

Recursive Partitioning and Regression Trees

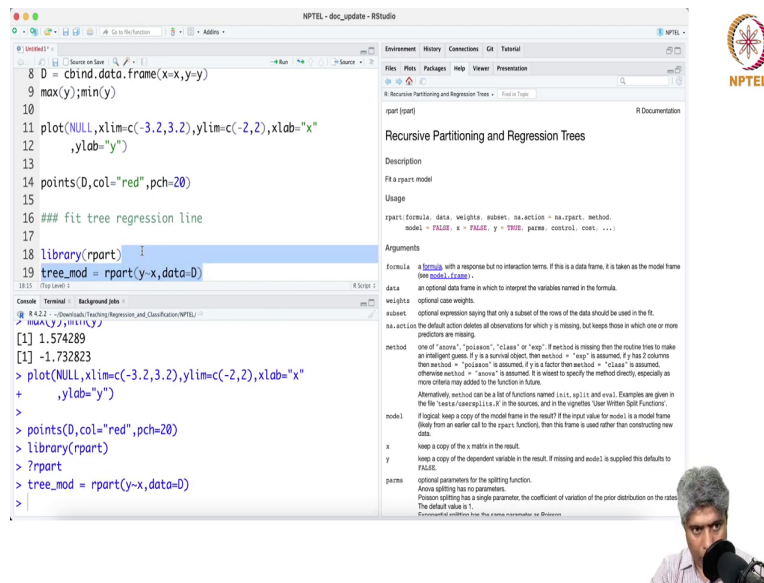
Description
Fit a rpart model

Usage
rpart: formula, data, weights, subset, na.action = na.rpart, method, model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...

Arguments
formula a formula with a response but no interaction terms. If this is a data frame, it is taken as the model frame (see `models::formula`).
data an optional data frame in which to interpret the variables named in the formula.
weights optional case weights.
subset optional expression saying that only a subset of the rows of the data should be used in the fit.
na.action the default action deletes all observations for which y is missing, but keeps those in which one or more predictors are missing.
method one of "anova", "poisson", "class" or "exp". If method is missing then the routine tries to make an intelligent guess. If y is a survival object, then method = "exp" is assumed. If y has 2 columns then method = "poisson" is assumed. If y is a factor then method = "class" is assumed, otherwise method = "anova" is assumed. It is wisest to specify the method directly, especially as more criteria may be added to the function in future.
Alternatively, method can be a list of functions named `list.uplit` and `eval`. Examples are given in the file `NEWS/asecplists.R` in the sources, and in the vignettes "User Written Split Functions".
model If logical, keep a copy of the model frame in the result? If the logical value for model is a model frame (likely from an earlier call to the rpart function), then this frame is used rather than constructing new data.
x keep a copy of the x matrix in the result.
y keep a copy of the dependent variable in the result. If missing and `model` is supplied this defaults to FALSE.
parms optional parameters for the splitting function.
Across splitting has no parameters.
Precision splitting has a single parameter, the coefficient of variation of the prior distribution on the rates. The default value is 1.
Environment `utils` for the `write.parameters` and `show` methods.

Now, what I am going to do, I am going to fit a regression line, tree regression line, regression line ok. What is that tree regression line? Yes, library rpart equal to tree model equal to rpart. So, rpart if the is the package. So, if I just load this package and question mark rpart. So, it's a recursive partitioning and regression tree.

(Refer Slide Time: 08:16)



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


```
8 D = cbind.data.frame(x=x,y=y)
9 max(y);min(y)
10
11 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
12 ,ylab="y")
13
14 points(D,col="red",pch=20)
15
16 ## fit tree regression line
17
18 library(rpart)
19 tree_mod = rpart(y~x,data=D)
```

The console window shows the output of the code:

```
[1] 1.574289
[1] -1.732823
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
+ ,ylab="y")
>
> points(D,col="red",pch=20)
> library(rpart)
> rpart
> tree_mod = rpart(y~x,data=D)
>
```

The help pane on the right displays the documentation for the `rpart` function, titled "Recursive Partitioning and Regression Trees". It includes a description, usage, arguments, and method details.

The NPTEL logo is visible in the top right corner of the RStudio window.



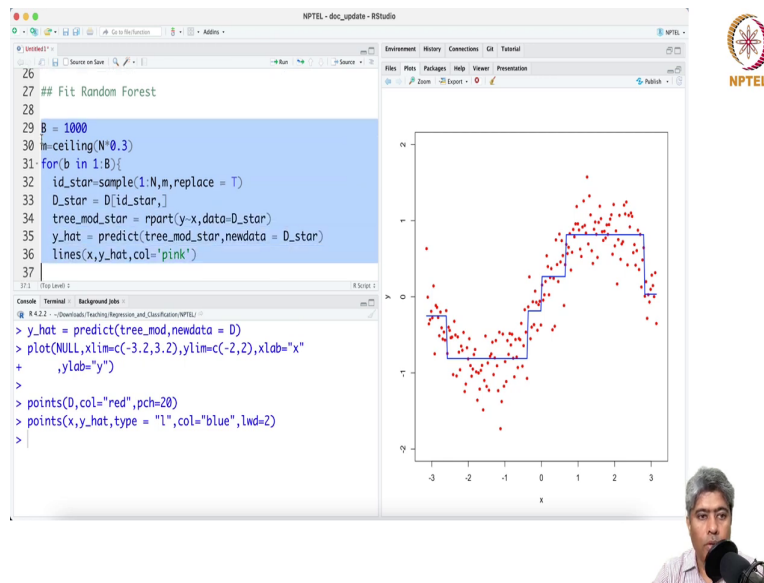
So, it says recursive partitioning and regression trees. So, it fits the regression tree structured model till the x comma data equal to d ok. Now, well let us not going to, I will tell you what it is me later, but let us try to plot, make some plot and then you will get a much better sense what exactly happening.

(Refer Slide Time: 08:40)



So, let me calculate \hat{y} equals to predict ok tree mod. This is the model and new data equal to D. Alright, next. Let me just take the plot. And then points, what I have is x. comma y hat type equal to L ok. And color equal to blue and line width equal to 2. So, let me just. So, you can see that its trying to fit through this ok. So, now, so, this is the decision tree.

(Refer Slide Time: 10:03)



The screenshot displays the RStudio interface. The source editor on the left contains the following R code:

```
26  
27 ## Fit Random Forest  
28  
29 B = 1000  
30 m=ceiling(N*0.3)  
31 for(b in 1:B){  
32   id_star=sample(1:N,m,replace = T)  
33   D_star = D[id_star,]  
34   tree_mod_star = rpart(y~x,data=D_star)  
35   y_hat = predict(tree_mod_star,newdata = D_star)  
36   lines(x,y_hat,col='pink')  
37 }
```

The console on the bottom left shows the execution of the code:

```
> y_hat = predict(tree_mod,newdata = D)  
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"  
+ ,ylab="y")  
> points(D,col="red",pch=20)  
> points(x,y_hat,type = "l",col="blue",lwd=2)  
>
```

The plot on the right shows a scatter plot of red points (D) and a blue step function (y_hat) representing the predicted values. The x-axis ranges from -3 to 3, and the y-axis ranges from -2 to 2. The NPTEL logo is visible in the top right corner of the RStudio window.

Now, if I have to fit a say random forest, let us try to fit random forest fit random forest. And so, first is my three number, say B is 1000. And then what I am going to do, I am just saying for b in 1 is to capital B, what you do is first bunch of simulate id star equal to sample from 1 is to N, sizes say N and replace equal to true.

So, maybe I will just take a small m and m equal to maybe ceiling capital N times 0.3 ok. So, I will take less number of samples, but random samples I will take. So, sort, we will just sort them. I do need to sort actually, why should I sort. And then D star D underscore star equal to D id star. And then what I will do, I will just fit this tree model here, but instead of tree model star with D star ok.

So, I will just do D star and then I am going to ask it as a lines ok. I have to do some prediction also. So, I have to, fitting I have to do prediction, three mod star D star ok and then x comma y hat and color equal to say pink ok. So, let me just run and see what happens ok.

(Refer Slide Time: 12:44)



There are some issues x and y. Of course, there is a issue here with this ok. So, I have to say D star dollar x.

(Refer Slide Time: 13:01)



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

```
29 B = 1000
30 m=ceiling(N*0.3)
31 for(b in 1:B){
32   id_star=sample(1:N,m,replace = T)
33   D_star = D[id_star,]
34   tree_mod_star = rpart(y~x,data=D_star)
35   y_hat = predict(tree_mod_star,newdata = D_star)
36   lines(D_star$x,y_hat,col='pink')
37 }
38 }
39 }
40 }
```

The console window shows the execution of the code:

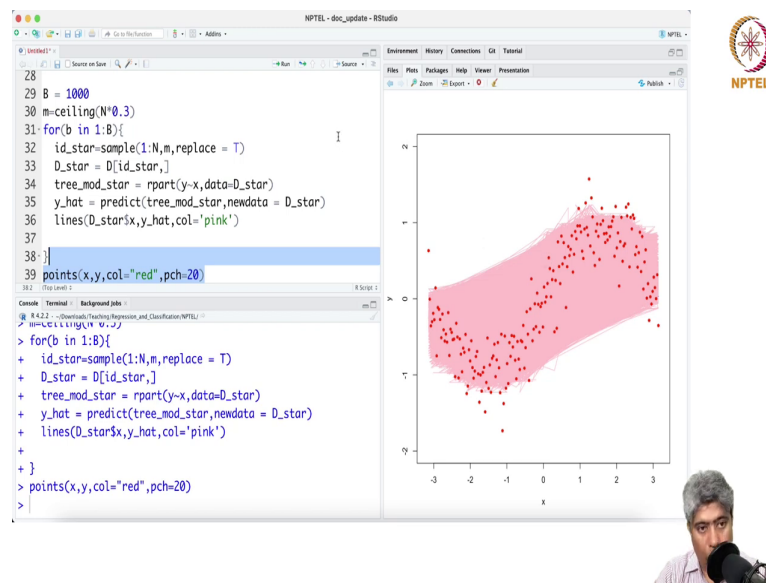
```
> m=ceiling(N*0.3)
> for(b in 1:B){
+ id_star=sample(1:N,m,replace = T)
+ D_star = D[id_star,]
+ tree_mod_star = rpart(y~x,data=D_star)
+ y_hat = predict(tree_mod_star,newdata = D_star)
+ lines(D_star$x,y_hat,col='pink')
+ }
+ }
+ }
> }
```

The plot window displays a scatter plot of data points (red dots) and a fitted model (pink shaded area). The x-axis ranges from -3 to 3, and the y-axis ranges from -2 to 4. The pink shaded area represents the predicted values for the data points.

The NPTEL logo is visible in the top right corner of the RStudio window.

Now, I think this will be fine yeah. So, it is just too many Perhaps.

(Refer Slide Time: 13:27)



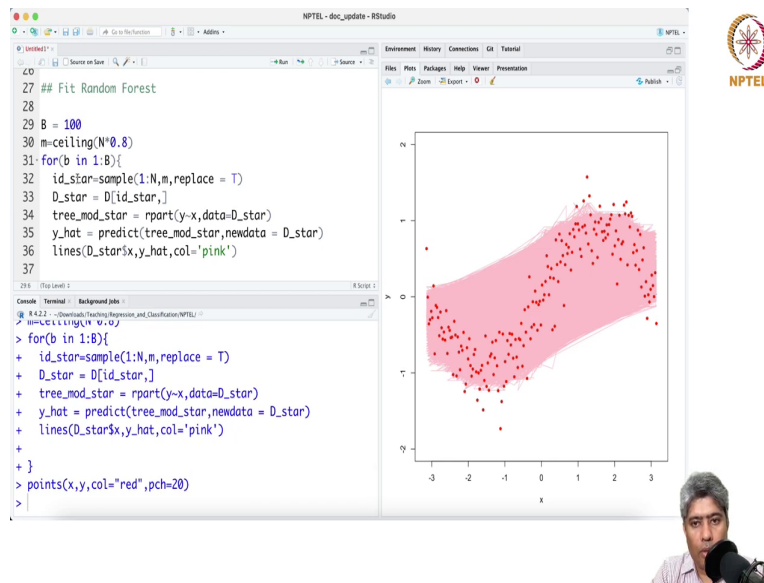
Now, if I use little bit and now if I put the points on this points x comma y color equal to red and pch equal to 20, because just kind of yeah. So, it is just too many over fitting is happening. So, what we can do instead of that, we can just increase the training samples to up to maybe 7 or 0.8 ok.

(Refer Slide Time: 14:03)



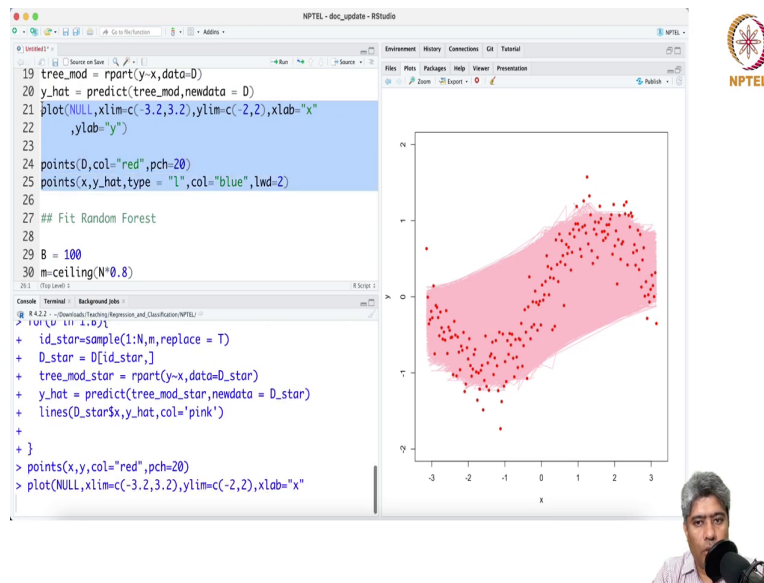
Maybe we can reduce the number for a while.

(Refer Slide Time: 14:17)



Maybe I am just let me reduce the number, because we can do it more and get a just sense. So, let me just run it. So, you will not get this.

(Refer Slide Time: 14:35)



The screenshot displays an RStudio interface with the following R code in the editor:

```
19 tree_mod = rpart(y~x,data=D)
20 y_hat = predict(tree_mod,newdata = D)
21 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
22       ,ylab="y")
23
24 points(D,col="red",pch=20)
25 points(x,y_hat,type="l",col="blue",lwd=2)
26
27 ## Fit Random Forest
28
29 B = 100
30 m=ceiling(N*0.8)
```

The console shows the execution of a function:

```
> id_star=sample(1:N,m,replace = T)
+ D_star = D[id_star,]
+ tree_mod_star = rpart(y~x,data=D_star)
+ y_hat = predict(tree_mod_star,newdata = D_star)
+ lines(D_star$x,y_hat,col='pink')
+
+ }
> points(x,y,col="red",pch=20)
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
```

The plot on the right shows a scatter plot of red points (D) and a blue line representing the predicted values (y_hat). A pink shaded region is overlaid on the plot, representing the fitted model's prediction interval. The x-axis ranges from -3.2 to 3.2, and the y-axis ranges from -2 to 2.

The NPTEL logo is visible in the top right corner of the RStudio window.

So, let me just run this first.

(Refer Slide Time: 14:37)

```
17
18 library(rpart)
19 tree_mod = rpart(y~x,data=D)
20 y_hat = predict(tree_mod,newdata = D)
21 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
22       ,ylab="y")
23
24 points(D,col="red",pch=20)
25 points(x,y_hat,type = "l",col="blue",lwd=2)
26
27 ## Fit Random Forest
28
29
30
31
32
33
```

Console Terminal | Background jobs

```
R 4.2.2 -- Downloads/Teaching/Regression_and_Classification/NPTEL/
+ xlim=c(-3.2,3.2),y_hat,col= "pink"
+
+ }
+ }
> points(x,y,col="red",pch=20)
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
+       ,ylab="y")
>
> points(D,col="red",pch=20)
> points(x,y_hat,type = "l",col="blue",lwd=2)
Error in xy.coords(x, y) : 'x' and 'y' lengths differ
> |
```




(Refer Slide Time: 14:39)

```
17
18 library(rpart)
19 tree_mod = rpart(y~x,data=D)
20 y_hat = predict(tree_mod,newdata = D)
21 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
22 ,ylab="y")
23
24 points(D,col="red",pch=20)
25 points(x,y_hat,type="l",col="blue",lwd=2)
26
27 ## Fit Random Forest
28
29
```

Console Terminal | Background jobs

```
R 4.2.2 -- Download/Teaching Regression and Classification NPTEL
> points(x,y_hat,type="l",col="blue",lwd=2)
Error in xy.coords(x, y) : 'x' and 'y' lengths differ
> library(rpart)
> tree_mod = rpart(y~x,data=D)
> y_hat = predict(tree_mod,newdata = D)
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
+ ,ylab="y")
>
> points(D,col="red",pch=20)
> points(x,y_hat,type="l",col="blue",lwd=2)
>
```



So, maybe I have just first run it few steps.

(Refer Slide Time: 14:46)



The image displays the RStudio interface. The left pane shows the following R code:

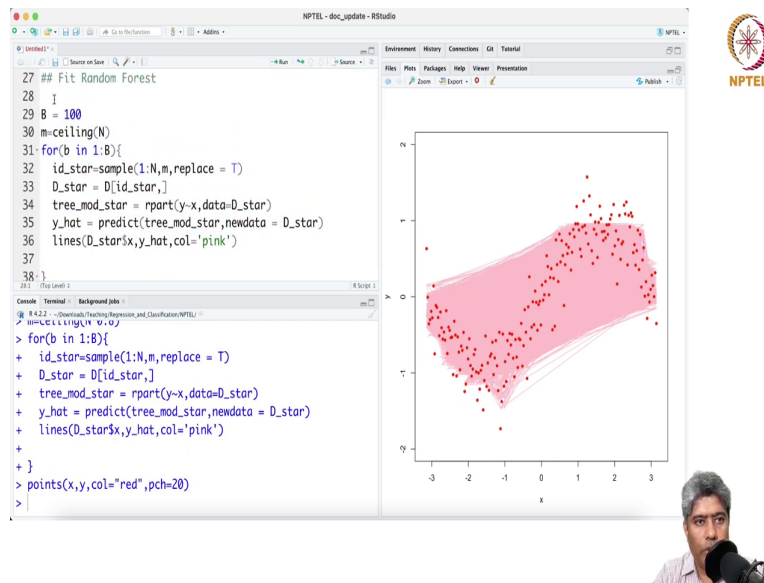
```
29 B = 100
30 m = ceiling(N*0.8)
31 for(b in 1:B){
32   id_star = sample(1:N,m,replace = T)
33   D_star = D[id_star,]
34   tree_mod_star = rpart(y~x,data=D_star)
35   y_hat = predict(tree_mod_star,newdata = D_star)
36   lines(D_star$x,y_hat,col='pink')
37 }
38 }
39 points(x,y,col='red',pch=20)
40 }
```

The right pane shows a scatter plot of the data points (red dots) and the predicted values (pink lines) for the bootstrapped model. The x-axis ranges from -3 to 3, and the y-axis ranges from -2 to 2. The plot shows a positive correlation between x and y, with a shaded pink region representing the confidence interval around the regression line.

Below the RStudio window, a small video feed shows a man speaking into a microphone.

And then I will just run this guy ok.

(Refer Slide Time: 14:52)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains the following R code:

```
27 ## Fit Random Forest
28 I
29 B = 100
30 m=ceiling(N)
31 for(b in 1:B){
32   id_star=sample(1:N,m,replace = T)
33   D_star = D[id_star,]
34   tree_mod_star = rpart(y~x,data=D_star)
35   y_hat = predict(tree_mod_star,newdata = D_star)
36   lines(D_star$x,y_hat,col='pink')
37 }
38 }
```

The console pane shows the execution of the code:

```
> for(b in 1:B){
+   id_star=sample(1:N,m,replace = T)
+   D_star = D[id_star,]
+   tree_mod_star = rpart(y~x,data=D_star)
+   y_hat = predict(tree_mod_star,newdata = D_star)
+   lines(D_star$x,y_hat,col='pink')
+ }
> points(x,y,col="red",pch=20)
>
```

The plot pane displays a scatter plot with red points (pch=20) and a pink shaded region representing the predicted values. The x-axis ranges from -3 to 3, and the y-axis ranges from -2 to 2. The pink region is roughly bounded by x from -1 to 2 and y from -1 to 1.5. The NPTEL logo is visible in the top right corner of the RStudio window.

And then I just run the entire thing. Just take the full samples. Let us take that. So, let me just clean it.

(Refer Slide Time: 15:05)



The image shows an RStudio window titled "NPTEL - doc_update - RStudio". The editor pane contains the following R code:

```
10 ## Fit tree regression line
17
18 library(rpart)
19 tree_mod = rpart(y~x,data=D)
20 y_hat = predict(tree_mod,newdata = D)
21 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
22      ,ylab="y")
23
24 points(D,col="red",pch=20)
25 points(x,y_hat,type = "l",col="blue",lwd=2)
26
27 ## Fit Random Forest
```

The console pane shows the execution of the code:

```
R 4.2.2 -- Downloads/Teaching/Regression_and_Classification/NPTEL/
> points(x,y,col="red",pch=20)
> library(rpart)
> tree_mod = rpart(y~x,data=D)
> y_hat = predict(tree_mod,newdata = D)
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
+      ,ylab="y")
>
> points(D,col="red",pch=20)
> points(x,y_hat,type = "l",col="blue",lwd=2)
>
```

The plot pane displays a scatter plot of red points (D) and a blue step function (y_hat) representing the fitted tree regression model. The x-axis ranges from -3 to 3, and the y-axis ranges from -2 to 2. The step function follows the general trend of the data points, showing a non-linear relationship.



(Refer Slide Time: 15:10)



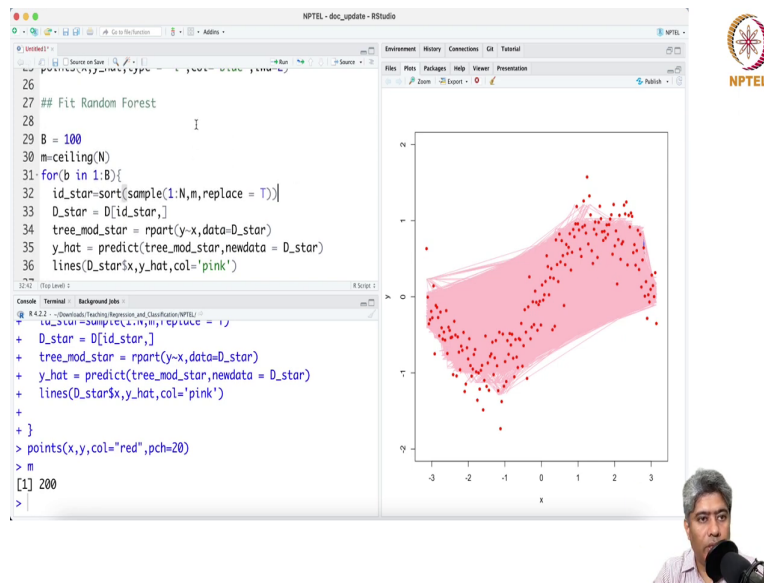
The image displays the RStudio interface. The left pane shows the following R code:

```
30 m=ceiling(N)
31 for(b in 1:B){
32   id_star=sample(1:N,m,replace = T)
33   D_star = D[id_star,]
34   tree_mod_star = rpart(y~x,data=D_star)
35   y_hat = predict(tree_mod_star,newdata = D_star)
36   lines(D_star$x,y_hat,col='pink')
37
38 }
39 points(x,y,col="red",pch=20)
40
41
```

The right pane shows a scatter plot with red points and a pink shaded region. The x-axis ranges from -3 to 3, and the y-axis ranges from -2 to 2. The pink shaded region is roughly bounded by x from -1 to 2 and y from -1 to 1.5. The NPTEL logo is visible in the top right corner of the RStudio window.

Below the RStudio window, a small video feed shows a man speaking into a microphone.

(Refer Slide Time: 15:15)

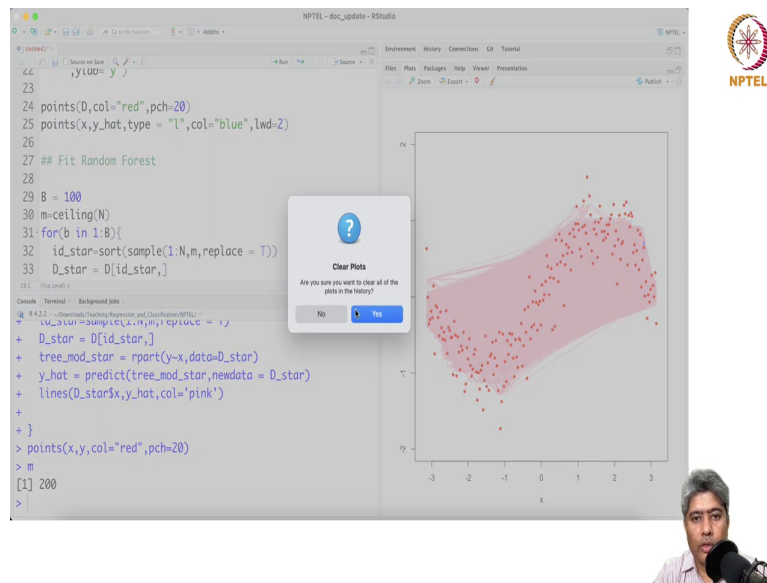


The image shows a screenshot of the RStudio interface. The left pane contains R code for fitting a random forest model. The code includes a loop over B iterations, where each iteration involves sampling data with replacement, fitting a tree model, and predicting on the new data. The right pane displays a scatter plot of data points (red dots) with a pink shaded region representing the predicted area. The x-axis ranges from -3 to 3, and the y-axis ranges from -2 to 2. The pink region is roughly centered around $x=0$ and $y=0$. The console shows the execution of the code, including the output of `points(x,y,col="red",pch=20)` and `m`.

```
26
27 ## Fit Random Forest
28
29 B = 100
30 m=ceiling(N)
31 for(b in 1:B){
32   id_star=sort(sample(1:N,m,replace = T))
33   D_star = D[id_star,]
34   tree_mod_star = rpart(y~x,data=D_star)
35   y_hat = predict(tree_mod_star,newdata = D_star)
36   lines(D_star$x,y_hat,col='pink')
}
> points(x,y,col="red",pch=20)
> m
[1] 200
>
```

I think it will be better if I just take a sorting. Otherwise, there will be. Yeah, so that is why it is creating this problem ok.

(Refer Slide Time: 15:26)



The image shows a screenshot of the RStudio interface. The main window displays a scatter plot with a fitted model. The plot shows a set of red points and a pink shaded region representing the fitted model. A dialog box titled "Clear Plots" is overlaid on the plot, asking "Are you sure you want to clear all of the plots in the history?" with "No" and "Yes" buttons. The R console on the left shows the following code:

```
23
24 points(D,col="red",pch=20)
25 points(x,y_hat,type="l",col="blue",lwd=2)
26
27 ## Fit Random Forest
28
29 B = 100
30 m=ceiling(N)
31 for(b in 1:B){
32   id_star=sort(sample(1:N,m,replace = T))
33   D_star = D[id_star,]
34
35   + D_star = D[id_star,]
36   + tree_mod_star = rpart(y~x,data=D_star)
37   + y_hat = predict(tree_mod_star,newdata = D_star)
38   + lines(D_star$x,y_hat,col='pink')
39   +
40   + }
41 }
42 > points(x,y,col="red",pch=20)
43 > m
44 [1] 200
45 >
```

I think I made a mistake by not. If I do the sorting.

(Refer Slide Time: 15:39)



Then it will be better in a sense that the color the drawing will be better. So, it is doing some random stuff because I am not sorting it. Yeah, now you can see this is how the random forest will work.

(Refer Slide Time: 15:48)



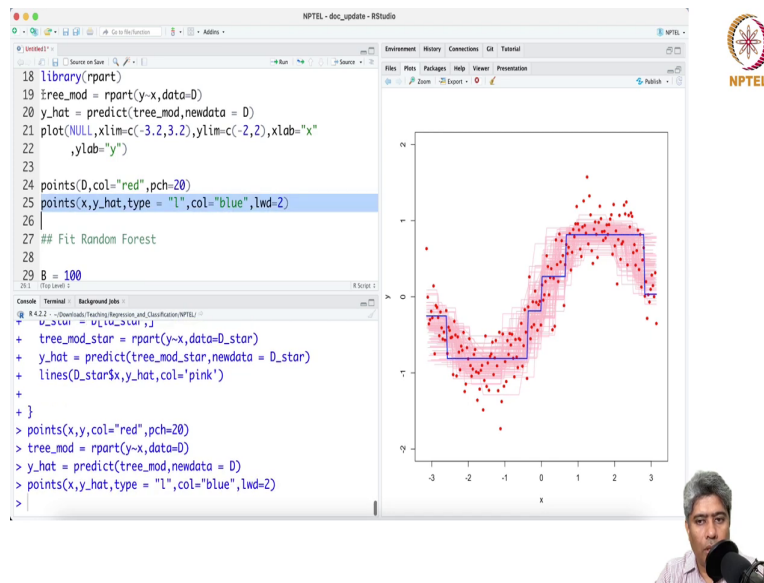
So, I can just reduce it to maybe 0.3. And if I just run this entire thing for 0.3.

(Refer Slide Time: 15:57)



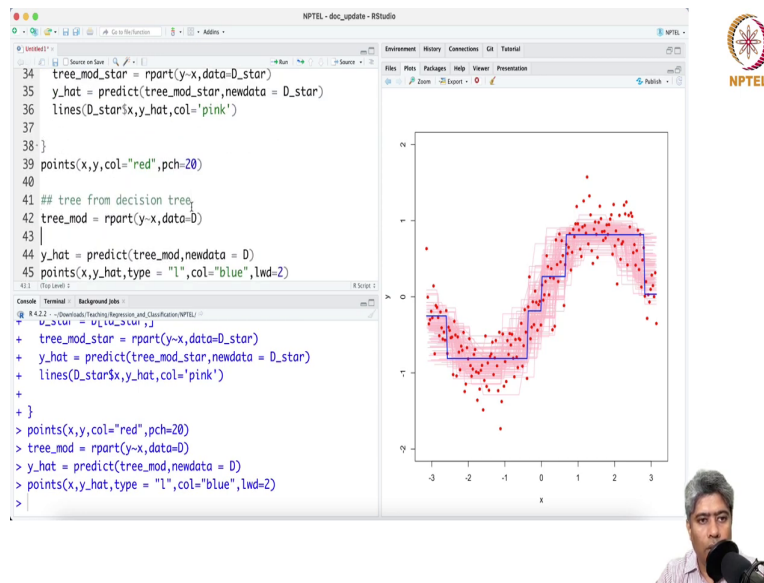
Then you will see something like this ok. So, you will see little bit more variability, right? And then if you just run the this guy and so, this is the decision tree.

(Refer Slide Time: 16:14)



So, the blue one is the decision tree, whereas, the red one is the random forest tree, tree from the random forest ok.

(Refer Slide Time: 16:23)



So, let me just quickly put those things. Tree from decision tree. So, let me just call it here. Yeah, so, now, I think you can just play around with the numbers a little bit. Maybe I will just say 500 trees. And let me just draw this first.

(Refer Slide Time: 17:02)



The image displays the RStudio interface with the following R code in the editor:

```
20 y_hat = predict(tree_mod,newdata = D)
21 plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
22 ,ylab="y")
23
24 points(D,col="red",pch=20)
25 points(x,y_hat,type = "l",col="blue",lwd=2)
26
27 ## Fit Random Forest
28
29 B = 500
30 m=ceiling(N*0.3)
```

The console shows the execution of the following commands:

```
+ }
> points(x,y,col="red",pch=20)
> tree_mod = rpart(y~x,data=D)
> y_hat = predict(tree_mod,newdata = D)
> points(x,y_hat,type = "l",col="blue",lwd=2)
> plot(NULL,xlim=c(-3.2,3.2),ylim=c(-2,2),xlab="x"
+ ,ylab="y")
+
> points(D,col="red",pch=20)
>
```

The plot on the right shows a scatter plot of red points (predicted values) and a blue line (observed values) on a coordinate system with x-axis from -3 to 3 and y-axis from -2 to 2. The data points form a parabolic shape opening upwards, with the blue line following the general trend of the red points.



(Refer Slide Time: 17:08)



The image shows an RStudio window with the following R code in the editor:

```
30 m = ceiling(N/0.3)
31 for(b in 1:B){
32   id_star = sort(sample(1:N,m,replace = T))
33   D_star = D[id_star,]
34   tree_mod_star = rpart(y~x,data=D_star)
35   y_hat = predict(tree_mod_star,newdata = D_star)
36   lines(D_star$x,y_hat,col='pink')
37
38 }
39 points(x,y,col='red',pch=20)
40
41 # tree from decision tree
```

The console shows the execution of the code:

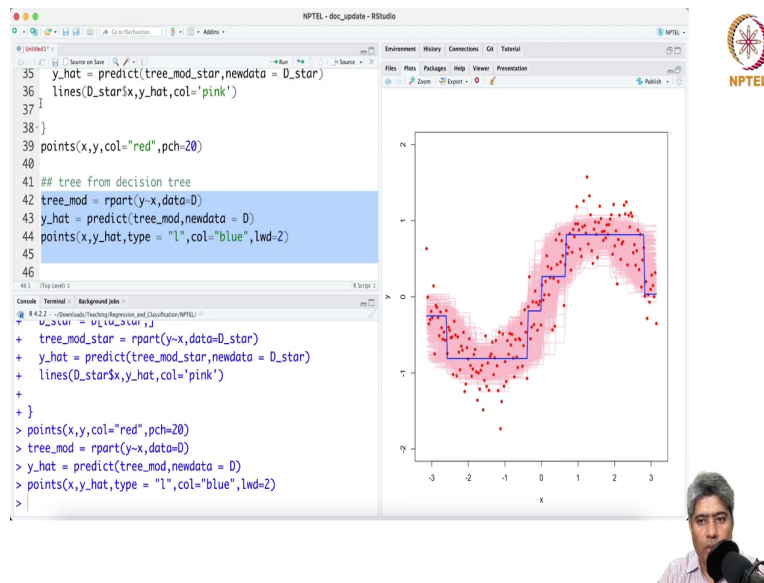
```
> for(b in 1:B){
+   id_star = sort(sample(1:N,m,replace = T))
+   D_star = D[id_star,]
+   tree_mod_star = rpart(y~x,data=D_star)
+   y_hat = predict(tree_mod_star,newdata = D_star)
+   lines(D_star$x,y_hat,col='pink')
+ }
> points(x,y,col='red',pch=20)
>
```

The plot on the right shows a scatter plot of red points (pch=20) forming a curved shape. Pink lines represent the decision tree boundaries, which are piecewise constant and approximate the curve of the data points. The x-axis ranges from -3 to 3, and the y-axis ranges from -2 to 4.

The NPTEL logo is visible in the top right corner of the RStudio window.

And then I will just run this random forest. So, this is a random forest. And here is the decision tree ok.

(Refer Slide Time: 17:14)



So, now you can see how this whole thing is happening. So, we got a sense that why random forest is. So, popular because its much more agile and its much more, you know, it can capture the non-linear and non-monotonic behavior between the x and y. Then linear model, linear model regression, you have to put the feature engineering in you have to built in the feature engineering itself.

But the decision tree and random forest has its own fine interesting behavior which captured the non-linear behavior very nicely. So, this is the advantage of using decision tree and random forest. So, I hope you enjoyed the video. Hands on with the random, tree random forest. This was on the simulated dataset. In the next video, we will work on the real dataset with the Football dataset from English Premier League data. So, see you in the next video.

