

Machine Learning for Soil and Crop Management
Professor Somsubhra Chakraborty
Agricultural and Food Engineering Department
Indian Institute of Technology, Kharagpur
Lecture 51
Digital Soil Mapping with Continuous Variables

(Refer Slide Time: 0:24)



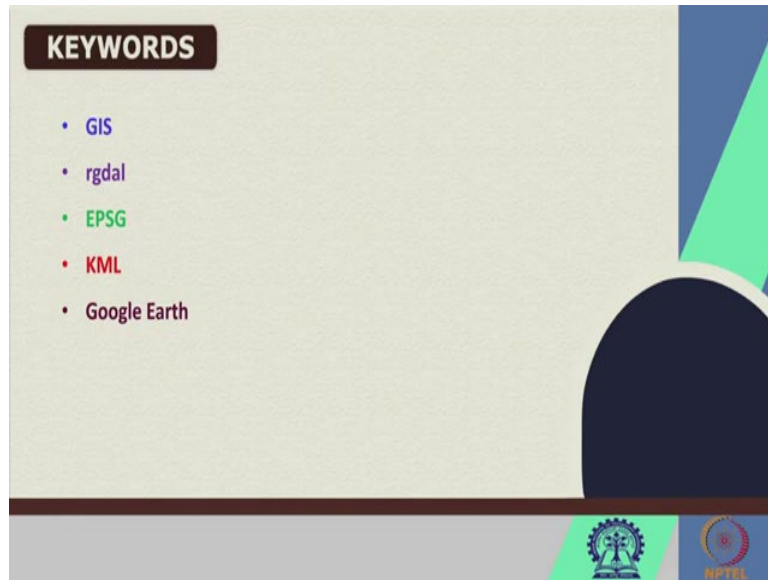
Welcome friends to this new week, week 11 of lectures of Digital Soil Mapping with continuous variables in this NPTEL online certification course of Machine Learning for Soil and Crop Management. And in this week we are going to see how to use, what is the continuous modeling and how we can use continuous modeling of soil properties and for their mapping, for their creating the special mapping of those properties.

(Refer Slide Time: 1:15)



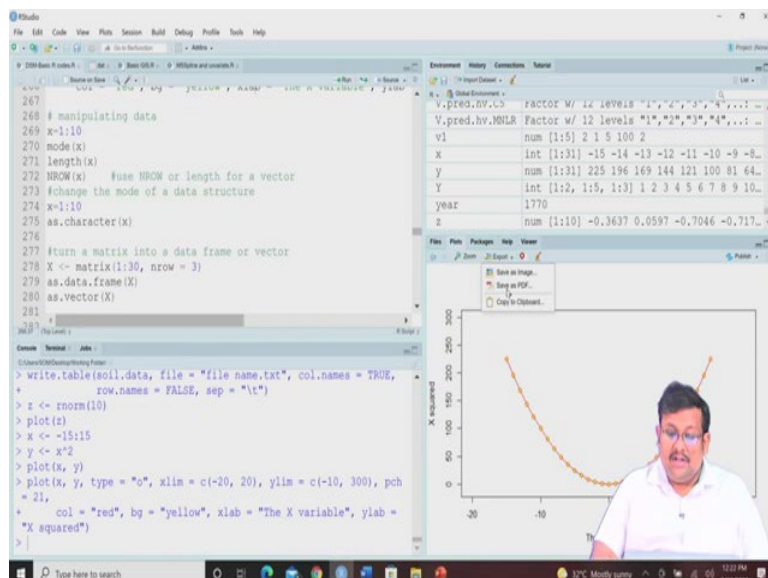
So, in my previous week we have already discussed the basics of DSM and basics of R programming basic operations of R. So, in this lecture this will be again using R. So, I will show you how to handle the data for DSM and how to do the continuous modeling for DSM and then will see the GIS with a very basic overview of GIS operation in R.

(Refer Slide Time: 1:34)



So, these are some of the keywords for this lecture so GIS, rgdal, EPSG, KML, Google Earth, so we will be learning, we will be discussing these things.

(Refer Slide Time: 1:54)



The image displays three sequential screenshots of an RStudio environment. In the first screenshot, the R console contains the following code:

```

267
268 # manipulating data
269 x=1:10
270 mode(x)
271 length(x)
272 nrow(x) #use nrow or length for a vector
273 #change the mode of a data structure
274 x=1:10
275 as.character(x)
276
277 #turn a matrix into a data frame or vector
278 X <- matrix(1:30, nrow = 3)
279 as.data.frame(X)
280 as.vector(X)
281
282 #random numbers
283 z <- rnorm(10)
284
285 #plotting
286 plot(z)
287
288 #plotting
289 x <- -15:15
290 y <- x^2
291 plot(x, y)
292 plot(x, y, type = "o", xlim = c(-20, 20),
+ col = "red", bg = "yellow", xlab = "The X variable", ylab =
+ "X squared")
293 >

```

The plot shows a parabolic curve with red points and a yellow background. A 'Save Plot as Image' dialog box is open, showing the file name 'plot.png' and the directory 'C:\Users\MO\Desktop\Working Folder'. The second screenshot shows the same dialog box with the 'File format' dropdown menu open, listing options: PNG, JPEG, TIFF, GIF, EPS, and PDF. The third screenshot shows the R console with updated code:

```

285 #random numbers
286 z <- rnorm(10)
287
288 #plotting
289 x <- -15:15
290 y <- x^2
291 plot(x, y, type = "o", xlim = c(-20, 20), ylim = c(-10, 300), pch
+ = 21,
+ col = "red", bg = "yellow", xlab = "The X variable", ylab =
+ "X squared")
292 >

```

The plot now includes a y-axis label 'X squared' and a y-axis range from -10 to 300. A 'Save Plot as Image' dialog box is open with the 'File format' dropdown set to 'PDF'.

Okay guys so let me start with where we left in our previous lecture. So, I showed you how to deal with different types of data set, how to select a particular, how to create a data frame,

how to call a library and then how to install a package, we have discussed and also we have seen that how to change the name of the character, how to create a matrix, how to do different types of matrix calculations.

So, we will start from there and then we will see the sorting of the data and we will go from there. Now, before going further I would like to show you that there is an export function in R so you can use this to this every plot to you can either save it as an image or you can save it as pdf.

So, when you click on save it as image you can have different types of, you can save it in different format like png, jpg, tif, bmp, meta file, svgs and you can change the aspect ratio and all these things. So, this is one of the way of saving the plot or you can directly clip it directly save it as pdf also or you can copy into the clipboard and then paste it later. So, guys we have seen that how to create this type of plots. Now, let us see how to manipulate the data.

(Refer Slide Time: 3:30)

The screenshot shows the RStudio interface. The console on the left contains the following R code:

```
264 plot(x, y, type = "o", xlim = c(-20, 20), ylim = c(-10, 300), pch = 21, col = "red", bg = "yellow", xlab = "The X variable", ylab = "X squared")
265
266
267
268 # manipulating data
269 x=1:10
270 mode(x)
271 length(x)
272 NROW(x) #use NROW or length for a vector
273 #change the mode of a data structure
274 x=1:10
275 as.character(x)
276
277 #turn a matrix into a data frame or vector
278 #matrix(1:30, nrow=3)
279 #colnames(m)
280 #rownames(m)
```

The environment pane on the right shows the following objects:

Object	Class	Value
V.pred.nv.L3	factor w/ 12 levels	"1", "2", "3", "4", ...
V.pred.hv.MMLR	Factor w/ 12 levels	"1", "2", "3", "4", ...
v1	num	[1:5] 2 1 5 100 2
X	int	[1:11] -15 -14 -13 -12 -11 -10 -9 -8...
y	num	[1:11] 225 196 169 144 121 100 81 64...
Y	int	[1:2, 1:5, 1:3] 1 2 3 4 5 6 7 8 9 10...
year		1770
z	num	[1:10] -0.3637 0.0597 -0.7046 -0.717...

The plot pane shows a scatter plot of X squared versus X. The x-axis is labeled "The X variable" and ranges from -20 to 20. The y-axis is labeled "X squared" and ranges from 0 to 300. The data points are red circles with yellow centers, forming a parabolic curve. A man's face is visible in the bottom right corner of the plot area.

RStudio interface showing R code and environment pane. The code defines a vector `x` from 1 to 10 and plots a parabolic curve. The environment pane shows the objects created.

```

265 plot(x, y, type = "o", xlim = c(-20, 20), ylim = c(-10, 300), p
266 col = "red", bg = "yellow", xlab = "The X variable", ylab
267
268 # manipulating data
269 x=1:10
270 mode(x)
271 length(x)
272 NROW(x) #use NROW or length for a vector
273 #change the mode of a data structure
274 x=1:10 I
275 as.character(x)
276
277 #turn a matrix into a data frame or vector
278 X <- matrix(1:30, nrow = 3)
279 as.data.frame(X)
280 as.vector(X)
281
282 nrow(X)
283 ncol(X)
284
285 # Indexing
286 NROW[x]
287
288
289 # Indexing
290
291

```

Environment pane:

```

Global Environment
V.pred.hv.V1 factor w/ 12 levels "1","2","3","4"...
V.pred.hv.MNLR Factor w/ 12 levels "1","2","3","4"...
v1 num [1:5] 2 1 5 100 2
X int [1:10] 1 2 3 4 5 6 7 8 9 10
y num [1:31] 225 196 169 144 121 100 81 64...
Y int [1:2, 1:5, 1:3] 1 2 3 4 5 6 7 8 9 10...
year 1770
z num [1:10] -0.3637 0.0597 -0.7046 -0.717...

```

RStudio interface showing R code for creating a matrix and its environment pane. The code creates a matrix `X` and converts it to a data frame.

```

269 x=1:10
270 mode(x)
271 length(x)
272 NROW(x) #use NROW or length for a vector
273 #change the mode of a data structure
274 x=1:10
275 as.character(x)
276
277 #turn a matrix into a data frame or vector
278 X <- matrix(1:30, nrow = 3)
279 as.data.frame(X)
280 as.vector(X)
281
282 nrow(X)
283 ncol(X)
284
285 # Indexing
286 NROW[x]
287
288
289 # Indexing
290
291

```

Environment pane:

```

Global Environment
X int [1:3, 1:10] 1 2 3 4 5 6 7 8 9 10 ...
z num [1:3, 1:3] 1 1 1 1 1 1 1 1 1
Values
a logi [1:10] FALSE FALSE FALSE FALSE FALS...
area_colors chr [1:12] "#FF0000" "#3BA800" "#730FFF"...
b 2
C.pred.hv.CS Factor w/ 12 levels "1","2","3","4"...
C.ored.hv.RF Factor w/ 12 levels "1","2","3","4"...

```

RStudio interface showing R code for indexing a matrix and its environment pane. The code uses `NROW[x]` for indexing and checks dimensions.

```

275 as.character(x)
276
277 #turn a matrix into a data frame or vector
278 X <- matrix(1:30, nrow = 3)
279 as.data.frame(X)
280 as.vector(X)
281
282 nrow(X)
283 ncol(X)
284 dim(X)
285 x=1:10
286 NROW[x]
287
288
289 # Indexing
290
291

```

Environment pane:

```

Global Environment
M1.pred.t named num [1:239] 3.15 3.3 3.3 3.15 2.35...
RT.pred.V Named num [1:103] 2.4 2.4 2.65 2.4 2.92 ...
soil chr [1:4] "Chromosol" "Vertosol" "Organo...
training int [1:238] 99 269 139 299 317 16 177 33...
V.pred.hv.CS Factor w/ 12 levels "1","2","3","4"...
V.pred.hv.MNLR Factor w/ 12 levels "1","2","3","4"...
v1 num [1:5] 2 1 5 100 2
X int [1:10] 1 2 3 4 5 6 7 8 9 10

```

So, let us see that there is a vector called `x` from 1 to 10 and then there is a mode of `x` let us see that is of course numeric, length of `x` as the name suggests so it is 10. So, here the number

of rows or with the n row is a function to identify the number of rows. So, here it will be 10 number of rows, of course. And if you change the mode of the data structure, so let us create this vector again and let us let us instruct R that you should remember this x as a character so we are going to use this as dot character function of x and then you will see that all the values are within the code. So, this is how you can change the numerical data to character data.

You can turn a matrix into data frame or vector. So, let us consider there is a matrix using the values of 1 to 30 and then a number of rows is 3. So, we are going to see this matrix. Now, let us see this matrix as a data frame so you can see that this is the data frames of v1 v2, v3, v4 variable up to v10 and these are the number of rows because number of rows are 3.

So, you can see that how a matrix can be represented as a data frame. So, here you can see the number of rows is 3, number of column which is n col is 10 and you can use this dim command or dimension command to see the dimension that is 3 and 10. Again, we are creating a vector of x equal to 1 to 10, number of rows is, number of row of x is 10.

So, you can do a lot of things with play, I am just showing a very limited number of functions but you see there are plenty of sources for learning R and there are plenty of functions in R you can explore from different websites and YouTube videos so please go ahead and learn those functions and utilize in your data set to feel more and more confident. I am just showing a handful of them because of time constraint but please pay attention and also practice this code so that you can feel more and more confident.

(Refer Slide Time: 6:07)

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

```
284 dim(X)
285 x=1:10
286 NROW(x)
287
288 # Indexing
289 v1 <- c(5, 1, 3, 8)
290 v1
291 v1[3]
292 v1[1:3]
293 v1[-4]
294 length(v1)
295 v1[8]=10
296 v1
```

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

```
> dim(X)
[1] 3
> ncol(X)
[1] 10
> dim(X)
[1] 3 10
> x=1:10
> NROW(x)
[1] 10
>
```

The Environment pane on the right shows the objects created:

Object	Class	Attributes
KI.pred.C	named num	[1:239] 3.15 3.3 3.3 3.13 2.35...
RT.pred.V	Named num	[1:103] 2.4 2.4 2.65 2.4 2.92 ...
soil	chr	[1:4] "Chromosol" "Vertosol" "Organo...
training	int	[1:238] 99 269 139 299 317 16 177 33...
V.pred.hv.CS	Factor w/ 12 levels	"1","2","3","4",... :
V.pred.hv.MNLR	Factor w/ 12 levels	"1","2","3","4",... :
v1	num	[1:5] 2 1 5 100 2
x	int	[1:10] 1 2 3 4 5 6 7 8 9 10

The bottom right pane shows a plot of a parabolic curve with orange data points. A small video inset of the presenter is visible in the bottom right corner of the plot area.

RStudio interface showing R code execution and environment variables.

```

289 # Indexing
290 v1 <- c(5, 1, 3, 8)
291 v1
292 v1[3]
293 v1[1:3]
294 v1[-4]
295 length(v1)
296 v1[8]=10
297 v1
298
299
300 library(ithir)
301 data(USYD_soil1)
302 soil.data <- USYD_soil1
303 dim(soil.data)
304 #
305
306 > v1[3]
[1] 3
307 > v1[1:3]
[1] 5 1 3
308 > v1[-4]
[1] 5 1 3
309 > length(v1)
[1] 4
310 > v1[8]=10
311 > v1
[1] 5 1 3 8 NA NA NA 10
312 >

```

Environment pane showing variables:

- RT.pred.V: Named num [1:103] 3.15 3.3 3.3 3.13 2.39...
- soil: chr [1:4] "Chromosol" "Vertosol" "Organo..."
- V.pred.hv.CS: Factor w/ 12 levels "1","2","3","4",...:
- V.pred.hv.MNLR: Factor w/ 12 levels "1","2","3","4",...:
- v1: num [1:8] 5 1 3 8 NA NA NA 10
- x: int [1:10] 1 2 3 4 5 6 7 8 9 10

RStudio interface showing R code execution and environment variables.

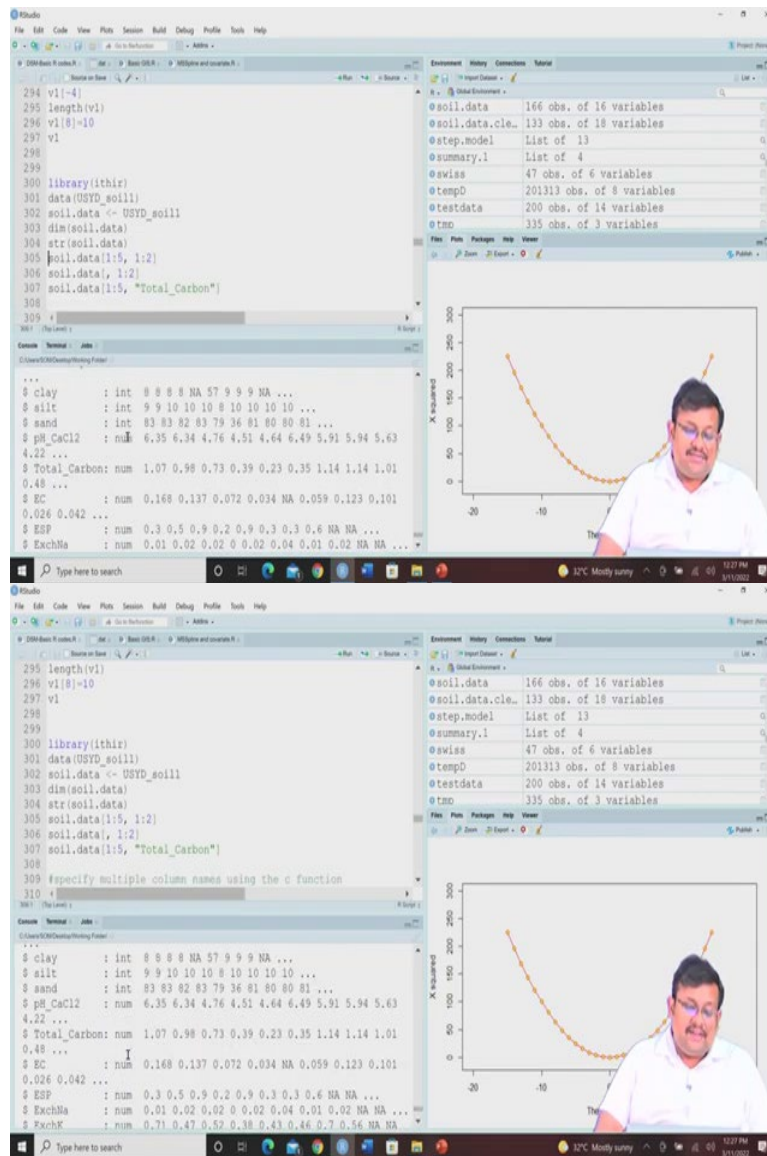
```

292 v1[3]
293 v1[1:3]
294 v1[-4]
295 length(v1)
296 v1[8]=10
297 v1
298
299
300 library(ithir)
301 data(USYD_soil1)
302 soil.data <- USYD_soil1
303 dim(soil.data)
304 #tr(soil.data)
305 soil.data[1:5, 1:2]
306 soil.data[, 1:2]
307 #
308
309 > v1[3]
[1] 3
310 > length(v1)
[1] 4
311 > v1[8]=10
312 > v1
[1] 5 1 3 8 NA NA NA 10
313 > library(ithir)
314 > data(USYD_soil1)
315 > soil.data <- USYD_soil1
316 > dim(soil.data)
[1] 166 16
317 >

```

Environment pane showing variables:

- @soil.data: 166 obs. of 16 variables
- @soil.data.cle: 133 obs. of 18 variables
- @step.model: List of 13
- @summary.1: List of 4
- @swiss: 47 obs. of 6 variables
- @tempD: 201313 obs. of 8 variables
- @testdata: 200 obs. of 14 variables
- @tmo: 335 obs. of 3 variables



Then we can see that we can index any number of the vectors. So, here you can see we are creating a vector v1 and this is a vector and then if we want to see the third value so we put this 3 in the parenthesis and we will see that this will be 3 and then if you want to see the first third value you can see that is from 5, 1, 3.

If you want to see the minus 4 so minus 4 index means all these values except the fourth one. So, if we run it you will see that 5, 1, 3, then if we want to see the length of v1 then we will see it is 4, if we want to see the eighth value in this if you want to assign the eighth value in this vector as 10 we can assign it but if you want to see it you will see that it is 5, 1, 3, 8 but this fifth, sixth and seventh values are missing, so eighth value is 10.

So, this is how you can create a vector. We are dealing with the library ithir, so let us call this library again ithir and then data USYD underscore soil and then soil data USYD soil 1 and let

us see the dimension of the soil data you can see here there are 166 observations and 16 variables so there will be 166 rows and 16, and there will be 16 variables.

So, if you want to see the structure of the data you will see this is the structure of the data, what are the integer, what are the numeric variables they will be clearly shown here. And if you want to reduce the data set, suppose you want to have only 1 to fifth variable and then 1 to 2 variables. So, you can just see, you can just select those variables only. So, profile you can see from 1 to 5 and also the number of columns 1 and 2 are given here. So, here that is the number of rows and number of columns you have selected.

(Refer Slide Time: 8:44)

The image displays two screenshots of the RStudio environment. The top screenshot shows the following R code in the editor:

```
296 v1[8]=10
297 v1
298
299
300 library(ihvir)
301 data(USYD_soil1)
302 soil.data <- USYD_soil1
303 dim(soil.data)
304 str(soil.data)
305 soil.data[1:5, 1:2]
306 soil.data[, 1:2]
307 #soil.data[1:5, "Total_Carbon"]
308
309 #specify multiple column names using the c function
310 soil.data[1:5, c("Total_Carbon", "CEC")]
311 #
```

The console output shows a table with columns 'Date', 'Treatment', and 'Jobs':

Date	Treatment	Jobs
140	25	Forest
141	25	Forest
142	25	Forest
143	25	Forest
144	26	Cropping
145	26	Cropping
146	26	Cropping
147	26	Cropping
148	26	Cropping
149	27	Improved pasture
150	27	Improved pasture
151	27	Improved pasture
152	27	Improved pasture

The environment pane shows the following objects:

- @soil.data: 166 obs. of 16 variables
- @soil.data.cle.: 133 obs. of 18 variables
- @step.model: List of 13
- @summary.1: List of 4
- @swias: 47 obs. of 6 variables
- @tempD: 201313 obs. of 8 variables
- @testdata: 200 obs. of 14 variables
- @tmo: 335 obs. of 3 variables

The plot shows 'X Required' on the y-axis (0 to 300) and 'The' on the x-axis (-20 to 10). A yellow curve is plotted, and a man's face is overlaid on the plot.

The bottom screenshot shows the same R code, but the console output table is different:

Date	Treatment	Jobs
99	17	Forest
100	17	Forest
101	18	Improved pasture
102	18	Improved pasture
103	18	Improved pasture
104	18	Improved pasture
105	18	Improved pasture
106	18	Improved pasture
107	19	Native pasture
108	19	Native pasture
109	19	Native pasture
110	19	Native pasture
111	19	Native pasture

The environment pane shows the same objects as the top screenshot. The plot is identical to the top screenshot.

And then if you want to select only the number of columns for all the rows so you keep it blank in the place of row and then you select this. So you will see that for all the 166

observation these two variable that is profile and land class are appearing. So, I hope it is clear.

(Refer Slide Time: 9:03)

The image displays two screenshots of an RStudio interface. The top screenshot shows the R console with the following code:

```
301 data(USYD_soil1)
302 soil.data <- USYD_soil1
303 dim(soil.data)
304 str(soil.data)
305 soil.data[1:5, 1:2]
306 soil.data[, 1:2]
307 soil.data[1:5, "Total_Carbon"]
308
309 #specify multiple column names using the c function
310 soil.data[1:5, c("Total_Carbon", "CEC")]
311
312 subset(soil.data, ESP > 10)
313 subset(soil.data, ESP > 10 & Lower.Depth > 0.3)
314
315 #sorting the data
316 x <- rnorm(5)
```

The Environment pane on the right shows the following objects:

- soil.data: 166 obs. of 16 variables
- soil.data.cle: 133 obs. of 18 variables
- estep.model: List of 13
- osummary.l: List of 4
- oswiss: 47 obs. of 6 variables
- otempD: 201313 obs. of 8 variables
- otestdata: 200 obs. of 14 variables
- otrm: 335 obs. of 3 variables

The console output for the code in the top screenshot is:

```
> soil.data[1:5, "Total_Carbon"]
[1] 1.07 0.98 0.73 0.39 0.23
> #specify multiple column names using the c function
> soil.data[1:5, c("Total_Carbon", "CEC")]
  Total_Carbon CEC
1      1.07 5.29
2       0.98 3.70
3       0.73 2.86
4       0.39 2.92
5       0.23 2.60
>
```

The bottom screenshot shows the same RStudio session with the following code:

```
302 soil.data <- USYD_soil1
303 dim(soil.data)
304 str(soil.data)
305 soil.data[1:5, 1:2]
306 soil.data[, 1:2]
307 soil.data[1:5, "Total_Carbon"]
308
309 #specify multiple column names using the c function
310 soil.data[1:5, c("Total_Carbon", "CEC")]
311
312 subset(soil.data, ESP > 10)
313 subset(soil.data, ESP > 10 & Lower.Depth > 0.3)
314
315 #sorting the data
316 x <- rnorm(5)
317 #
```

The console output for the code in the bottom screenshot is:

```
147 26 Cropping 0.15 0.24 51 8 42
148 26 Cropping 0.70 0.80 50 9 40
pH_C12 Total_Carbon EC ESP ExchNa ExchK ExchCa ExchMg
68 4.65 1.49 0.499 13.0 1.00 0.74 2.17 3.76
111 6.10 0.50 0.223 17.4 2.62 0.30 3.74 8.37
113 4.64 1.08 0.301 11.1 0.31 0.87 1.01 0.63
117 6.30 0.32 0.214 12.1 1.66 0.27 4.19 7.61
118 7.17 0.09 0.292 21.2 2.88 0.33 2.86 7.50
123 5.05 0.25 0.073 13.2 0.95 0.30 2.00 3.92
147 6.27 0.63 0.134 10.4 2.09 0.74 5.09 12.23
148 7.81 0.84 0.820 16.4 4.93 0.91 7.52 16.72
CEC
```

Both screenshots include a plot of 'X' (Y-axis, 0 to 300) versus 'The' (X-axis, -30 to 10). The plot shows a parabolic curve with orange dots. A video overlay of a man speaking is present in the bottom right corner of both screenshots.

The image displays three sequential screenshots of an RStudio environment, illustrating data manipulation and visualization steps.

Top Screenshot: The console shows the following R code:

```

304 str(soil.data)
305 soil.data[1:5, 1:2]
306 soil.data[, 1:2]
307 soil.data[1:5, "Total_Carbon"]
308
309 #specify multiple column names using the c function
310 soil.data[1:5, c("Total_Carbon", "CEC")]
311
312 subset(soil.data, ESP > 10)
313 subset(soil.data, ESP > 10 & Lower.Depth > 0.3)
314
315 #sorting the data
316 x <- rnorm(5)
317 x
318 y <- sort(x)
319 y

```

The Environment pane shows objects: soil.data (166 obs. of 16 variables), soil.data.cle... (133 obs. of 18 variables), ostep.model (List of 13), osummary.1 (List of 4), oswias (47 obs. of 6 variables), otempD (20133 obs. of 8 variables), otestdata (200 obs. of 14 variables), and otm (335 obs. of 3 variables). The console output shows a data frame with columns: Cropping, pH_CaCl2, Total_Carbon, EC, ESP, ExchM, ExchK, ExchCa, ExchMg, and CEC. A plot of X vs Y is shown with a parabolic curve.

Middle Screenshot: The console shows the following R code:

```

317 soil.data[1:5, "Total_Carbon"]
318
319 #specify multiple column names using the c function
320 soil.data[1:5, c("Total_Carbon", "CEC")]
321
322 subset(soil.data, ESP > 10)
323 subset(soil.data, ESP > 10 & Lower.Depth > 0.3)
324
325 #sorting the data
326 x <- rnorm(5)
327 x
328 y <- sort(x)
329 y

```

The Environment pane shows objects: Ki.pred.t (named num [1:258]), RT.pred.V (named num [1:103]), soil (chr [1:4]), training (int [1:238]), V.pred.hv.CS (Factor w/ 12 levels), V.pred.hv.MNLR (Factor w/ 12 levels), vl (num [1:8]), and x (num [1:5]). The console output shows the same data frame as the top screenshot. A plot of X vs Y is shown with a parabolic curve.

Bottom Screenshot: The console shows the following R code:

```

314
315 #sorting the data
316 x <- rnorm(5)
317 x
318 y <- sort(x)
319 y
320
321
322 head(soil.data[order(soil.data$clay),])
323 order(soil.data$clay)
324
325 # to know where the data resides in a particular data structure
326 match(max(soil.data$CEC, na.rm = TRUE), soil.data$CEC)
327 soil.data[CEC[95]]
328 which(soil.data$ESP>5)
329 y

```

The Environment pane shows objects: Ki.pred.v (named num [1:103]), soil (chr [1:4]), training (int [1:238]), V.pred.hv.CS (Factor w/ 12 levels), V.pred.hv.MNLR (Factor w/ 12 levels), vl (num [1:8]), x (num [1:5]), and y (num [1:5]). The console output shows the same data frame as the top screenshot. A plot of X vs Y is shown with a parabolic curve.

Now, if you want to see the first 5 rows of total carbon variable we can use it so these are the values of the total carbon for first 5 observations and you can specify multiple column names

using the C function. So, if you want to combine multiple column if you want to see the multiple column combinedly you can use this C function and you can get so this is 1 to 5 and then the first 5 values of both total carbon and CEC.

So, if you want to see which soil data, which subset of the soil data is having the ESP values greater than 10, so you can use this subset function and you can see these are the observation for which we are having the ESP values of greater than 10. You can add more and more constraints that is ESP greater than 10 and lower depth is greater than 0.3 using the same subset function and you can see the number of samples are getting down.

You can sort the data using the random norm, R norm or random normal and then you can sort them, you can sort, further you can create y and then you can sort x and then you will get the results of sorting.

(Refer Slide Time: 10:36)

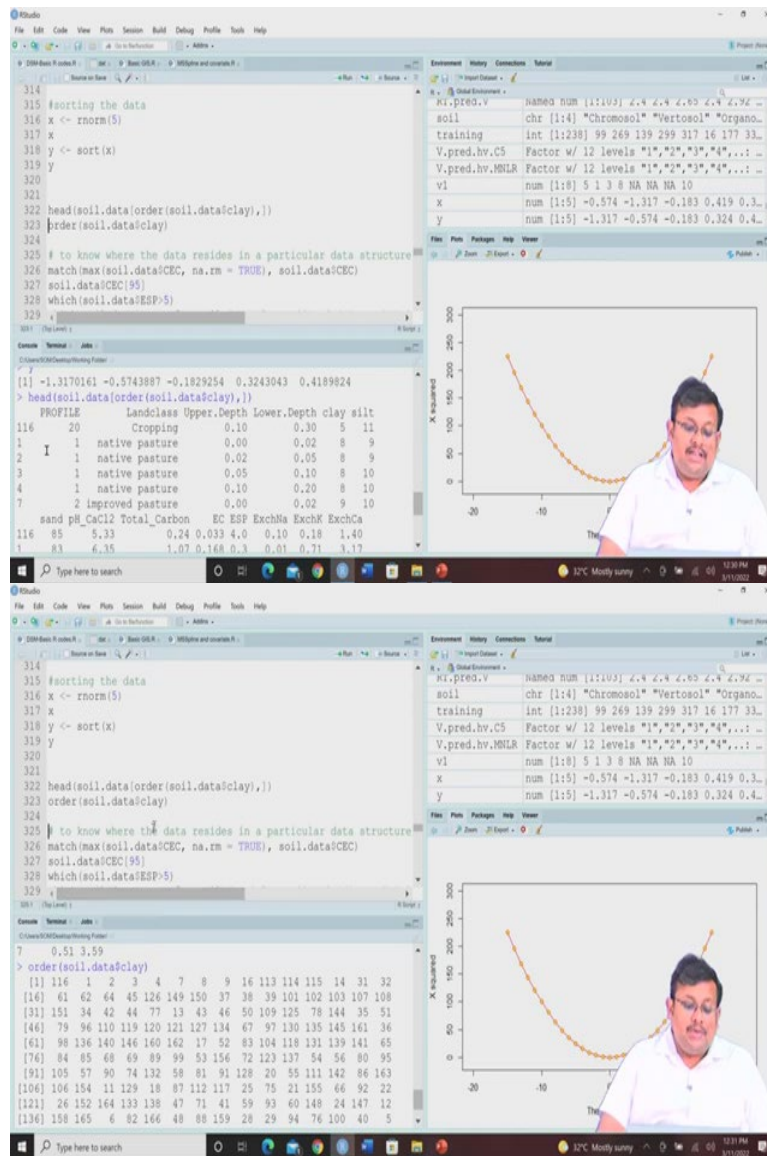
The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for data manipulation and analysis:

```
314  
315 #sorting the data  
316 x <- rnorm(5)  
317 x  
318 y <- sort(x)  
319 y  
320  
321  
322 head(soil.data[order(soil.data$clay),])  
323 |rder(soil.data$clay)  
324  
325 # to know where the data resides in a particular data structure  
326 match(max(soil.data$CEC, na.rm = TRUE), soil.data$CEC)  
327 soil.data[95]  
328 which(soil.data$ESP>5)  
329
```
- Console:** Shows the output of the code, including a table of soil data values:

```
2 83 6.34 0.98 0.137 0.5 0.02 0.47 3.50  
3 82 4.76 0.73 0.072 0.9 0.02 0.52 1.34  
4 83 4.51 0.39 0.034 0.2 0.00 0.38 1.03  
7 81 5.91 1.14 0.123 0.3 0.01 0.70 2.92  
ExchMg CEC  
116 0.70 1.90  
1 0.59 5.29  
2 0.60 3.70  
3 0.22 2.86  
4 0.22 2.92  
7 0.51 3.59
```
- Environment Pane:** Lists objects in the workspace, including a data frame named 'soil' with columns: 'chr', 'training', 'V.pred.hv.CS', 'V.pred.hv.MNLR', 'v1', 'x', and 'y'.

```
soil      chr [1:4] "Chromosol" "Vertosol" "Organo...  
training int [1:238] 99 269 139 299 317 16 177 33...  
V.pred.hv.CS Factor w/ 12 levels "1","2","3","4",... :  
V.pred.hv.MNLR Factor w/ 12 levels "1","2","3","4",... :  
v1       num [1:8] 5 1 3 8 NA NA NA 10  
x        num [1:5] -0.574 -1.317 -0.183 -0.419 0.3...  
y        num [1:5] -1.317 -0.574 -0.183 0.324 0.4...
```
- Plot:** A scatter plot with 'X' on the x-axis and 'Y' on the y-axis. The data points form a parabolic curve opening upwards. A person's face is visible in the bottom right corner of the plot area.



Then if you want to, if you want to order the clay data and want to see only the first 6 observation after ordering so you can see we are specifying the clay and then we are ordering this and then from there we can want to see the first 6 observation by head function. So, head will give you the first 6 observation and then soil dot data and then we are first ordering by taking the clay variables only, so this is how let me show you.

So, this is the 6 observation, the first observation will be 116th observation according to the clay content followed by the number 1 observation, number 2, number 3, number 4 and number 7. So, this is how you do this type of data manipulation. And then you can order the soil data according to the low to high or high to low in this fashion by ordering the data.

So, to know the data residue, sorry data resides in a particular data structure or not so you can use this match function so you can use this match function you can see we are specifying CEC data and then we are trying to see whether in the CEC data there is a missing values and

so after removing these missing values from the CEC we want to see what is the maximum value.

(Refer Slide Time: 12:16)

The image displays two screenshots of an RStudio environment. The top screenshot shows the R console with the following code and output:

```
317 x
318 y <- sort(x)
319 y
320
321
322 head(soil.data[order(soil.data$clay),])
323 order(soil.data$clay)
324
325 # to know where the data resides in a particular data structure
326 match(max(soil.data$CEC, na.rm = TRUE), soil.data$CEC)
327 soil.data$CEC[95]
328 which(soil.data$ESP>5)
329 which(soil.data$ESP>5 & soil.data$clay > 30) #adding multipl
330 which(is.na(soil.data$ESP)) #identifying the missing value obs
331 #factors
332 a <- c(rep(0, 4), rep(1, 4))
333 a
334 x<- factor(a)
335 x
336 soil.drainage<- c("well drained", "imperfectly drained", "poorl
337 "poorly drained", "well drained", "poorly dra
338 soil.drainage1 <- factor(soil.drainage)
```

The output shows the maximum CEC value is 95, and the corresponding ESP values are 5 and 30. A plot of X (required) vs Y (observed) is visible in the background.

The bottom screenshot shows the R console with the following code and output:

```
324
325 # to know where the data resides in a particular data structure
326 match(max(soil.data$CEC, na.rm = TRUE), soil.data$CEC)
327 soil.data$CEC[95]
328 which(soil.data$ESP>5)
329 which(soil.data$ESP > 5 & soil.data$clay > 30)
330 which(is.na(soil.data$ESP)) #identifying the missing value observ
331 #factors
332 a <- c(rep(0, 4), rep(1, 4))
333 a
334 x<- factor(a)
335 x
336 soil.drainage<- c("well drained", "imperfectly drained", "poorl
337 "poorly drained", "well drained", "poorly dra
338 soil.drainage1 <- factor(soil.drainage)
```

The output shows the maximum CEC value is 95, and the corresponding ESP values are 5 and 30. A plot of X (required) vs Y (observed) is visible in the background.

So, if you see that it will see that it is 95 and then 95th variable and if you want to see the data of the CEC of the 95th variable just use the index and you will get the value of 28.21 you can check it from the data set. Also you can use this ESP you can identify this the variable, the values which are greater than, ESP values which are greater than 5 so which soil ESP data in the soil data set is greater than 5 you can identify, these are the observation for which ESP is greater than 5.

So, you can add more multiple conditions by using this AND sign and then you can use that within the ESP which are the missing values so you can identify the missing values like this.

(Refer Slide Time: 13:17)

The image displays three sequential screenshots of an RStudio environment, illustrating the process of data manipulation and plotting. Each screenshot includes a console window with R code and its output, a variable viewer on the right, and a plot of 'X required' versus 'X'.

Top Screenshot: The console shows the following code:

```
327 soil.data$CEC[95]
328 which(soil.data$ESP>5)
329 which(soil.data$ESP > 5 & soil.data$clay > 30) #adding multi-
330 which(is.na(soil.data$ESP)) #identifying the missing value obs
331 #factors
332 a <- c(rep(0, 4), rep(1, 4))
333 a
334 k<- factor(a)
335 x
336 soil.drainage<- c("well drained", "imperfectly drained", "poorl
337 "poorly drained", "well drained", "poorly dra
338 soil.drainage1 <- factor(soil.drainage)
339 soil.drainage1
340 as.numeric(soil.drainage1)
341
342 #
343 #
```

The console output shows the values of 'a' and the indices of missing values in 'soil.data\$ESP'. The variable viewer shows the structure of 'a' and 'k'. The plot shows a parabolic curve with orange data points.

Middle Screenshot: The console shows the following code:

```
330 [soil.data$ESP]) #identifying the missing value observations
331
332 ), 4), rep(1, 4))
333
334 s)
335
336 pe<- c("well drained", "imperfectly drained", "poorly drained",
337 "poorly drained", "well drained", "poorly drained")
338 pe1 <- factor(soil.drainage)
339 pe1
340 soil.drainage1)
341
342 #control the order of the levels of factor
343 pe2 <- factor(soil.drainage, levels = c("well drained",
344 "imperfectly drained",
345 #
```

The console output shows the levels of the factor 'pe1' and the resulting factor 'pe2'. The variable viewer shows the structure of 'pe1' and 'pe2'. The plot is identical to the top screenshot.

Bottom Screenshot: The console shows the following code:

```
330 which(is.na(soil.data$ESP)) #identifying the missing value obs
331 #factors
332 a <- c(rep(0, 4), rep(1, 4))
333 a
334 x<- factor(a)
335 x
336 soil.drainage<- c("well drained", "imperfectly drained", "poorl
337 "poorly drained", "well drained", "poorly dra
338 soil.drainage1 <- factor(soil.drainage)
339 soil.drainage1
340 #s.numeric(soil.drainage1)
341
342 #control the order of the levels of factor
343 soil.drainage2 <- factor(soil.drainage, levels = c("well draine
344 "imperfectly
345 #
```

The console output shows an error message: "Error in factor(soil.drainage) : object 'soil.drainage' not found". The variable viewer shows the structure of 'a' and 'x'. The plot is identical to the top screenshot.

You can you can create some factors like replication of 0, 4 times and replication of 1, 4 times and give them the name a. So, a will look like this 0 0 0 0 1 1 1 1 and let us see x is also equal to factor equal to a so let us see it will be 0 0 0 0 1 1 1 1 so here we can see two levels 0 and 1.

So, let us create a vector called, let us create a variable which is soil drainage and you can see that is well there are couple of categories like well drained, imperfectly drained, poor drained, poorly drained, well drained, poorly drained, so these are the val, these are the categorical levels within the soil drainage and then we want to see the factors of the soil drainage.

So, let us create a factor and then let us see the soil drainage. So, you can see these are the factors, well drained, imperfectly drained, poor drained, poorly drained, well drained and poorly drained and so here levels are there imperfectly drained, poorly drained and well drained, so there are three levels.

(Refer Slide Time: 14:53)

The screenshot shows the RStudio interface with the following content:

```

334 a <- factor(a)
335 x
336 soil.drainage <- c("well drained", "imperfectly drained", "poorly
337   drained", "well drained", "poorly dra
338 soil.drainage1 <- factor(soil.drainage)
339 soil.drainage1
340 as.numeric(soil.drainage1)
341
342 #control the order of the levels of factor
343 soil.drainage2 <- factor(soil.drainage, levels = c("well drained",
344   "imperfectly drained", "poorly drained"))
345 as.numeric(soil.drainage2)
346
347
348 #combine the data
349 soil
350
351 #combine the data
352 soil
353
354 #combine the data
355 soil
356
357 #combine the data
358 soil
359
360 #combine the data
361 soil
362
363 #combine the data
364 soil
365
366 #combine the data
367 soil
368
369 #combine the data
370 soil
371
372 #combine the data
373 soil
374
375 #combine the data
376 soil
377
378 #combine the data
379 soil
380
381 #combine the data
382 soil
383
384 #combine the data
385 soil
386
387 #combine the data
388 soil
389
390 #combine the data
391 soil
392
393 #combine the data
394 soil
395
396 #combine the data
397 soil
398
399 #combine the data
400 soil
401
402 #combine the data
403 soil
404
405 #combine the data
406 soil
407
408 #combine the data
409 soil
410
411 #combine the data
412 soil
413
414 #combine the data
415 soil
416
417 #combine the data
418 soil
419
420 #combine the data
421 soil
422
423 #combine the data
424 soil
425
426 #combine the data
427 soil
428
429 #combine the data
430 soil
431
432 #combine the data
433 soil
434
435 #combine the data
436 soil
437
438 #combine the data
439 soil
440
441 #combine the data
442 soil
443
444 #combine the data
445 soil
446
447 #combine the data
448 soil
449
450 #combine the data
451 soil
452
453 #combine the data
454 soil
455
456 #combine the data
457 soil
458
459 #combine the data
460 soil
461
462 #combine the data
463 soil
464
465 #combine the data
466 soil
467
468 #combine the data
469 soil
470
471 #combine the data
472 soil
473
474 #combine the data
475 soil
476
477 #combine the data
478 soil
479
480 #combine the data
481 soil
482
483 #combine the data
484 soil
485
486 #combine the data
487 soil
488
489 #combine the data
490 soil
491
492 #combine the data
493 soil
494
495 #combine the data
496 soil
497
498 #combine the data
499 soil
500
501 #combine the data
502 soil
503
504 #combine the data
505 soil
506
507 #combine the data
508 soil
509
510 #combine the data
511 soil
512
513 #combine the data
514 soil
515
516 #combine the data
517 soil
518
519 #combine the data
520 soil
521
522 #combine the data
523 soil
524
525 #combine the data
526 soil
527
528 #combine the data
529 soil
530
531 #combine the data
532 soil
533
534 #combine the data
535 soil
536
537 #combine the data
538 soil
539
540 #combine the data
541 soil
542
543 #combine the data
544 soil
545
546 #combine the data
547 soil
548
549 #combine the data
550 soil
551
552 #combine the data
553 soil
554
555 #combine the data
556 soil
557
558 #combine the data
559 soil
560
561 #combine the data
562 soil
563
564 #combine the data
565 soil
566
567 #combine the data
568 soil
569
570 #combine the data
571 soil
572
573 #combine the data
574 soil
575
576 #combine the data
577 soil
578
579 #combine the data
580 soil
581
582 #combine the data
583 soil
584
585 #combine the data
586 soil
587
588 #combine the data
589 soil
590
591 #combine the data
592 soil
593
594 #combine the data
595 soil
596
597 #combine the data
598 soil
599
600 #combine the data
601 soil
602
603 #combine the data
604 soil
605
606 #combine the data
607 soil
608
609 #combine the data
610 soil
611
612 #combine the data
613 soil
614
615 #combine the data
616 soil
617
618 #combine the data
619 soil
620
621 #combine the data
622 soil
623
624 #combine the data
625 soil
626
627 #combine the data
628 soil
629
630 #combine the data
631 soil
632
633 #combine the data
634 soil
635
636 #combine the data
637 soil
638
639 #combine the data
640 soil
641
642 #combine the data
643 soil
644
645 #combine the data
646 soil
647
648 #combine the data
649 soil
650
651 #combine the data
652 soil
653
654 #combine the data
655 soil
656
657 #combine the data
658 soil
659
660 #combine the data
661 soil
662
663 #combine the data
664 soil
665
666 #combine the data
667 soil
668
669 #combine the data
670 soil
671
672 #combine the data
673 soil
674
675 #combine the data
676 soil
677
678 #combine the data
679 soil
680
681 #combine the data
682 soil
683
684 #combine the data
685 soil
686
687 #combine the data
688 soil
689
690 #combine the data
691 soil
692
693 #combine the data
694 soil
695
696 #combine the data
697 soil
698
699 #combine the data
700 soil
701
702 #combine the data
703 soil
704
705 #combine the data
706 soil
707
708 #combine the data
709 soil
710
711 #combine the data
712 soil
713
714 #combine the data
715 soil
716
717 #combine the data
718 soil
719
720 #combine the data
721 soil
722
723 #combine the data
724 soil
725
726 #combine the data
727 soil
728
729 #combine the data
730 soil
731
732 #combine the data
733 soil
734
735 #combine the data
736 soil
737
738 #combine the data
739 soil
740
741 #combine the data
742 soil
743
744 #combine the data
745 soil
746
747 #combine the data
748 soil
749
750 #combine the data
751 soil
752
753 #combine the data
754 soil
755
756 #combine the data
757 soil
758
759 #combine the data
760 soil
761
762 #combine the data
763 soil
764
765 #combine the data
766 soil
767
768 #combine the data
769 soil
770
771 #combine the data
772 soil
773
774 #combine the data
775 soil
776
777 #combine the data
778 soil
779
780 #combine the data
781 soil
782
783 #combine the data
784 soil
785
786 #combine the data
787 soil
788
789 #combine the data
790 soil
791
792 #combine the data
793 soil
794
795 #combine the data
796 soil
797
798 #combine the data
799 soil
800
801 #combine the data
802 soil
803
804 #combine the data
805 soil
806
807 #combine the data
808 soil
809
810 #combine the data
811 soil
812
813 #combine the data
814 soil
815
816 #combine the data
817 soil
818
819 #combine the data
820 soil
821
822 #combine the data
823 soil
824
825 #combine the data
826 soil
827
828 #combine the data
829 soil
830
831 #combine the data
832 soil
833
834 #combine the data
835 soil
836
837 #combine the data
838 soil
839
840 #combine the data
841 soil
842
843 #combine the data
844 soil
845
846 #combine the data
847 soil
848
849 #combine the data
850 soil
851
852 #combine the data
853 soil
854
855 #combine the data
856 soil
857
858 #combine the data
859 soil
860
861 #combine the data
862 soil
863
864 #combine the data
865 soil
866
867 #combine the data
868 soil
869
870 #combine the data
871 soil
872
873 #combine the data
874 soil
875
876 #combine the data
877 soil
878
879 #combine the data
880 soil
881
882 #combine the data
883 soil
884
885 #combine the data
886 soil
887
888 #combine the data
889 soil
890
891 #combine the data
892 soil
893
894 #combine the data
895 soil
896
897 #combine the data
898 soil
899
900 #combine the data
901 soil
902
903 #combine the data
904 soil
905
906 #combine the data
907 soil
908
909 #combine the data
910 soil
911
912 #combine the data
913 soil
914
915 #combine the data
916 soil
917
918 #combine the data
919 soil
920
921 #combine the data
922 soil
923
924 #combine the data
925 soil
926
927 #combine the data
928 soil
929
930 #combine the data
931 soil
932
933 #combine the data
934 soil
935
936 #combine the data
937 soil
938
939 #combine the data
940 soil
941
942 #combine the data
943 soil
944
945 #combine the data
946 soil
947
948 #combine the data
949 soil
950
951 #combine the data
952 soil
953
954 #combine the data
955 soil
956
957 #combine the data
958 soil
959
960 #combine the data
961 soil
962
963 #combine the data
964 soil
965
966 #combine the data
967 soil
968
969 #combine the data
970 soil
971
972 #combine the data
973 soil
974
975 #combine the data
976 soil
977
978 #combine the data
979 soil
980
981 #combine the data
982 soil
983
984 #combine the data
985 soil
986
987 #combine the data
988 soil
989
990 #combine the data
991 soil
992
993 #combine the data
994 soil
995
996 #combine the data
997 soil
998
999 #combine the data
1000 soil

```

The console output shows the following results:

```

> soil.drainage1
[1] well drained      imperfectly drained poorly drained
[4] poorly drained    well drained        poorly drained
Levels: imperfectly drained poorly drained well drained
>

```

The plot shows a quadratic function with orange data points and a blue curve. The x-axis ranges from -20 to 20, and the y-axis ranges from 0 to 300. The curve is a parabola opening upwards, with its vertex at approximately (-10, 100). The data points are scattered around the curve, showing a clear upward trend.

RStudio interface showing R code and a plot. The code defines soil drainage levels and creates a data frame. The plot shows a parabolic relationship between soil drainage and a response variable.

```
343 factor(soil.drainage, levels = c("well drained",
344 "imperfectly drained", "poorly
345 drained"))
346
347
348 #
349 .frame(soil = c("Vertosol", "Hydrosol", "Sodosol"),
350 response = 1:3)
351
352 .frame(soil = c("Chromosol", "Dermosol", "Tenosol"),
353 response = 4:6)
354
355 (soil.info1, soil.info2)
356
357 3.2, 1.2, 2.1, 2, 0.5)
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
```

```
> soil.drainage1 <- factor(soil.drainage)
> soil.drainage2
[1] well drained      imperfectly drained poorly drained
[4] poorly drained    well drained      poorly drained
Levels: imperfectly drained poorly drained well drained
> soil.drainage2 <- factor(soil.drainage, levels = c("well drained",
+ "imperfectly dr
ained", "poorly drained"))
> as.numeric(soil.drainage2)
[1] 1 2 3 3 1 3
>
```

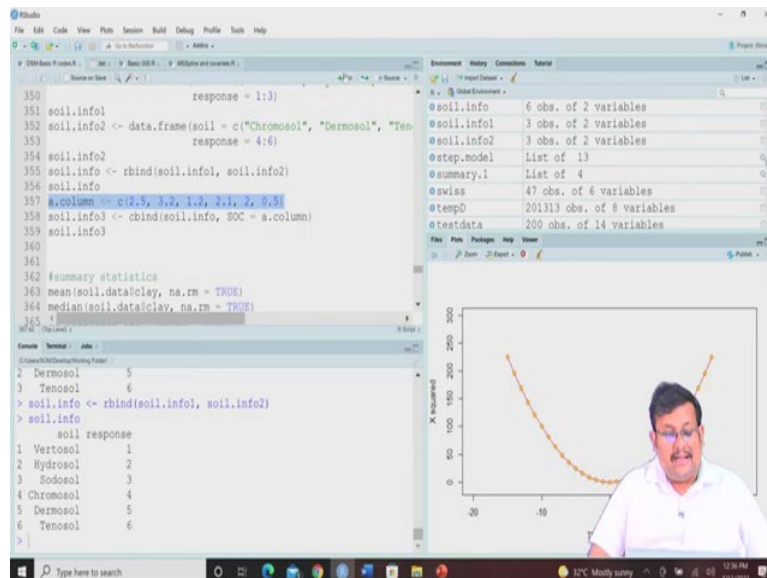

drainage and levels are well drained, imperfectly drained and poorly drained. And let us use this soil drain.

So, you can see based on your data it will be labeled as 1 2 3 3 1 3 or not. So, you can assign some values and based on that you can create a new variable that soiled or drainage and you can numerically arrange those factors if you have already assigned. Now, if you want to combine the data you can combine the data in form of a table.

So, let us assume that we can create a data frame with the, with this three observation that is soil and response, so vertosol, hydrosol, sodosol and response is 1 2 3, so 1 2 3. So, let us see how it looks like. So, it will look like this and suppose there is another called soil info which is the data frame which you have created with the chromosol, dermosol, and tenosol with the response 4 to 6. So, let us see how it looks like soil info 2, it will look like this.

Now if we use this R bind function it will give you combine of rows. So, here you can see there are 3 rows in soil info, in soil 2, soil info 1 there is 3 rows and soil info 2 there are also 3 rows. So, if we use this R bind function it will combine all these 6 rows together so you can see this is how R info will look like 1 2 3 4 5 6.

(Refer Slide Time: 16:55)



The screenshot displays an RStudio interface. The main editor window contains R code for creating data frames and combining them. The console shows the execution of these commands, resulting in a data frame with 6 rows and 2 columns. The Environment pane on the right lists the objects created, including @soil.info, @soil.info1, @soil.info2, @step.model, @summary.l, @swiss, @temp0, and @testdata. A plot window in the bottom right shows a scatter plot of X vs Y with a yellow curve fitted to the data points. A small video inset of a man is visible in the bottom right corner of the plot window.

```
350 response = 1:3)
351 soil.info1
352 soil.info2 <- data.frame(soil = c("Chromosol", "Dermosol", "Tenosol"),
353 response = 4:6)
354 soil.info2
355 soil.info <- rbind(soil.info1, soil.info2)
356 soil.info
357 a.column <- c(2.5, 3.3, 1.2, 2.1, 2, 0.5)
358 soil.info3 <- cbind(soil.info, SOC = a.column)
359 soil.info3
360
361
362 #summary statistics
363 mean(soil.data$clay, na.rm = TRUE)
364 median(soil.data$clay, na.rm = TRUE)
365 #summary statistics
366
```

Environment

- @soil.info 6 obs. of 2 variables
- @soil.info1 3 obs. of 2 variables
- @soil.info2 3 obs. of 2 variables
- @step.model List of 13
- @summary.l List of 4
- @swiss 47 obs. of 6 variables
- @temp0 20133 obs. of 8 variables
- @testdata 200 obs. of 14 variables

Console

```
> soil.info <- rbind(soil.info1, soil.info2)
> soil.info
  soil response
1 Vertosol   1
2 Hydrosol   2
3 Sodosol    3
4 Chromosol  4
5 Dermosol   5
6 Tenosol    6
>
```

```

357 a.col1000 <- c(1,2,3,4,4,4,4,4,4,4,0,0)
358 soil.info3 <- chind(soil.info, SOC = a.column)
359 soil.info3
360
361
362 #summary statistics
363 mean(soil.data$clay, na.rm = TRUE)
364 median(soil.data$clay, na.rm = TRUE)
365 sd(soil.data$clay, na.rm = TRUE)
366 var(soil.data$clay, na.rm = TRUE)
367 summary(soil.data)
368 summary(soil.data, 1:6)
369
370 #Histogram and boxplots
371 hist(soil.data$clay)
372 boxplot(soil.data$clay)

```

Environment: 6 obs. of 3 variables
 @soil.info3 List of 13
 @step.model List of 4
 @summary.l List of 4
 @swiss 47 obs. of 6 variables
 @tempD 201313 obs. of 8 variables
 @testdata 200 obs. of 14 variables
 @tmp 335 obs. of 3 variables
 @traindata 306 obs. of 14 variables

```

PROFILE
  Min. : 1.00 Cropping :49 Min. :0.0000
  1st Qu.: 8.00 Forest :50 1st Qu.:0.0200
  Median :15.00 improved pasture:15 Median :0.0500
  Mean :14.73 native pasture :32 Mean :10.1816
  3rd Qu.:22.00 3rd Qu.:0.2000
  Max. :38.00 Max. :16.3000

```

```

373
374 #When plotting one variable by some factor in a box-plot
375 boxplot(Total_Carbon ~ Landclass, data = soil.data)
376
377
378

```

Lower.Depth clay silt
 Min. :0.0200 Min. : 5.00 Min. : 6.0
 1st Qu.:0.0500 1st Qu.:15.00 1st Qu.:11.0
 Median :0.1000 Median :21.00 Median :15.0
 Mean :0.2464 Mean :26.95 Mean :16.5
 3rd Qu.:0.3000 3rd Qu.:37.00 3rd Qu.:20.0
 Max. :0.8000 Max. :68.00 Max. :32.0
 NA's :17 NA's :1

```

380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

1st Qu.:0.0500 1st Qu.:15.00 1st Qu.:11.0
 Median :0.1000 Median :21.00 Median :15.0
 Mean :0.2464 Mean :26.95 Mean :16.5
 3rd Qu.:0.3000 3rd Qu.:37.00 3rd Qu.:20.0
 Max. :0.8000 Max. :68.00 Max. :32.0
 NA's :17 NA's :1

So, you can create another variable that is a dot column which is a vector and you can see that is soil info and these are the values so you can combined these values this column with this

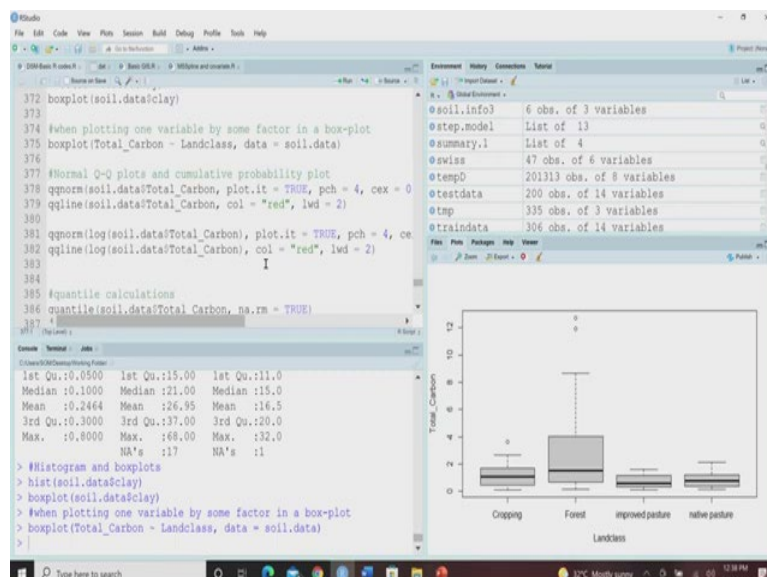
soil info file so you can see that how it look like, soil info data set so ultimately the soil info 3 will look like this. So, earlier it is soil info and then we have added 1 column that is soil info 3.

So, you can do some summary statistics by selecting specific web, specific variables like mean, median, standard deviation, variance, summary so by selecting a specific variable and removing the, and removing the missing values so you can see here these are the results it is a mean of soil clay, median of soil clay, standard deviation of soil clay, variance of soil clay and this is the summary statistics of the soil data for all the variables.

So, if you want to see the summary statistics of the selected data set from 1 to 6 variable you can also use this and you can see that this is for 1 to 6 variables summary statistics. You can do some histograms for using histogram the command is hist so the function is histogram function so you can see this histogram is created by using the clay values. You can create the box plots for the clay and this you can see this is the box plot showing the distribution.

When plotting 1 variable by some factor in the box plots let us consider that let us plot this total carbon by different land classes and our data set is soil dot data so we are going to use this box plot and you can see for different land class like cropping, forest, improve posture, native posture, we are getting the distribution of total carbon.

(Refer Slide Time: 19:02)

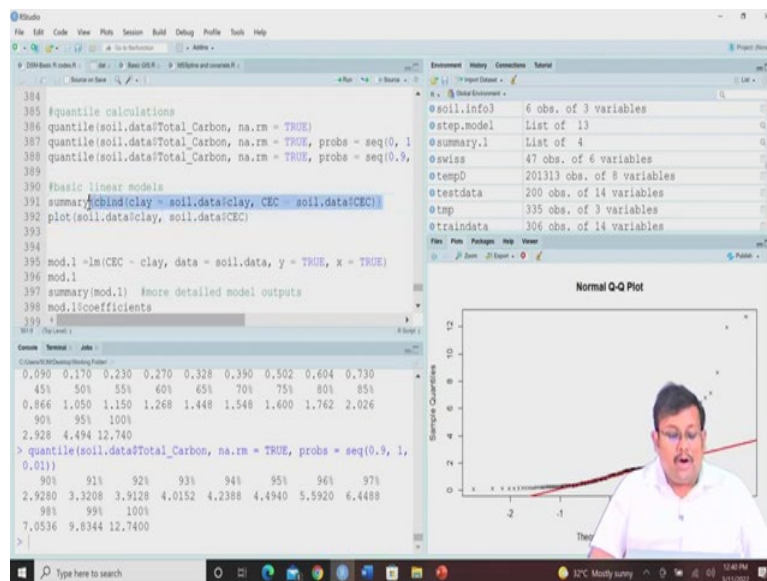


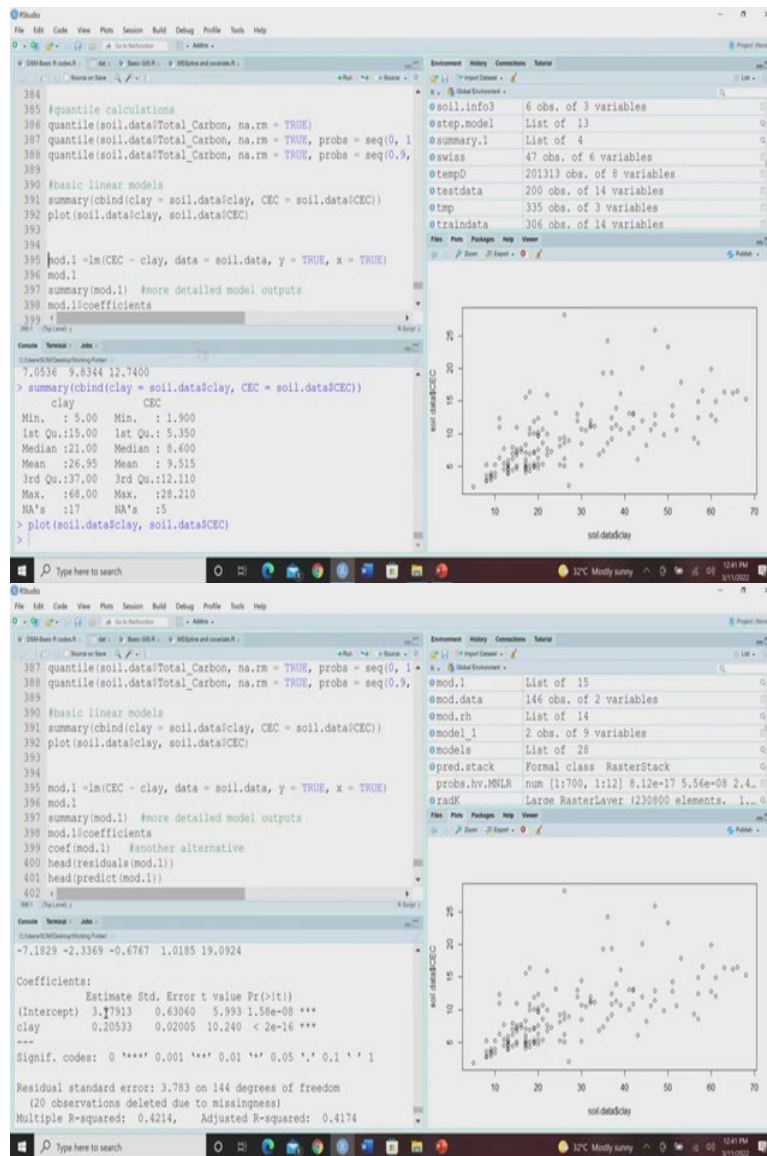
norm we specify the variable, we plot it and then plot character 4 we use then the size also 0.7 so let us see how it looks like.

So, this is the qq norm and then qq line if you click on it then you will see that we will fit a line to see whether they are normally distributed or not. So, this is how you can do different types of operation with the data set, you can do quantile calculation of the total carbon, you can do the quantile, if you just by default use this quantile it will give you from 0 percent, 25 percent, 50 percent, 75 percent, 100 percent.

So, therefore quartile it will give but if you specify the range and the sequence it will give. So, here you can see from 0 to 1 with the interval of 0.05 you will get the quantile values so if you run it you will get from 0 to 100 percent with 5 percent increment you will get and similarly here another example is given from 0.9 to 1 with 0.01 so from 90 percent to 100 percent for each 1 percent interval you will get the value.

(Refer Slide Time: 20:40)





So, basic linear model again so summary you can use the c bind function, clay is data. So, suppose you want to predict the CEC based on the clay content. So, let us first combined the clay and CEC so we are combining using the columbine function and let us see the summary of these two.

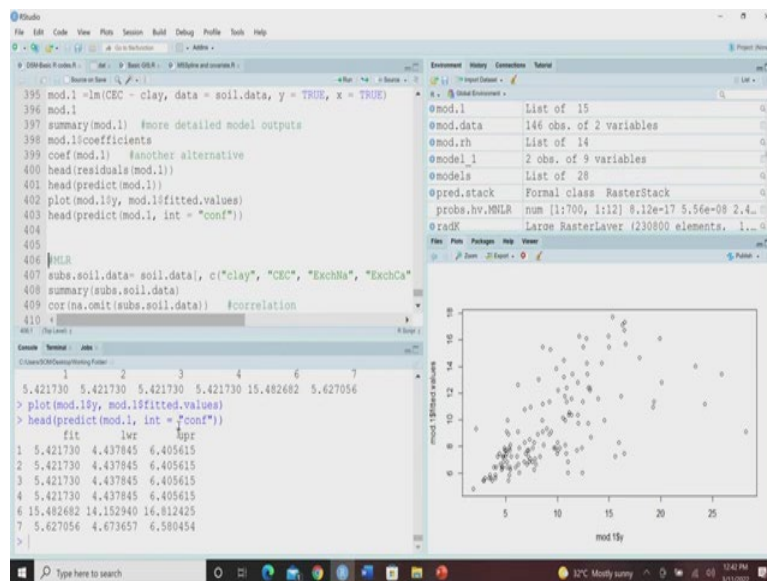
So, you can see the summary of clay the minimum value, first quartile, median, mean, third quartile, maximum and number of missing values you can see here. You can directly plot the clay versus CEC by using this plot command and this is the simple plot between clay versus CEC.

But if you want to use the model mod 1 so we are using this linear model lm command to predict the CEC using the clay, our data is soil dot data so we are going to use this and then you can see here this will be CEC Clay and model data so here you can see the intercept and the slope.

So, if you want to have the summary of the model dot 1 so you will get the summary of the model of 1 and then coefficient so this is an offset and this is the coefficient of the clay, what are the standard error, what are the t values, whether they are significant or not at different like significant level, these will be also mentioned.

So, multiple R square, adjusted R square, all these things will be mentioned. So, in this case we are getting multiple R square of 0.42, adjusted R square is 0.41. But it is a simple linear regression because we are using only clay.

(Refer Slide Time: 22:27)



So, you can have only the coefficient by specifying the coefficients from the model also. You can, another alternative is using this coeff function, you can use this coeff function to get the same results, you can use this head function to get the first 6 residuals of the first 6 observation or you can get the first 6 predicted values of the models from first 6 observation.

You can use these predicted values versus fitted values by using the plot command you can see here. You can also see the head that means first 6 observation of model 1 with the confidence interval you can see here a lower confidence interval and upper limit of confidence interval they are same with the fitted values.

(Refer Slide Time: 23:19)

The image displays three sequential screenshots of an RStudio environment, illustrating the workflow of data analysis and model fitting.

Top Screenshot: The R console shows the following code and output:

```
401 head(predict(mod.1))
402 plot(mod.l$y, mod.l$fitted.values)
403 head(predict(mod.1, int = "conf"))
404
405
406 #MLR
407 subs.soil.data= soil.data[, c("clay", "CEC", "ExchNa", "ExchCa")
408 summary(subs.soil.data)
409 cor(na.omit(subs.soil.data)) #correlation
410 pairs(na.omit(subs.soil.data))
411 mod.2 <- lm(CEC ~ clay + ExchNa + ExchCa, data = subs.soil.data)
412 summary(mod.2)
413
414
415
```

The Environment pane on the right lists the following objects:

- @mod.1: List of 15
- @mod.data: 146 obs. of 2 variables
- @mod.rh: List of 14
- @model.1: 2 obs. of 9 variables
- @models: List of 28
- @pred.stack: Formal class 'RasterStack'
- probs.hv.MNLR: num [1:700, 1:12] 8.12e-17 5.56e-08 2.4...
- @raster: Large RasterLayer (230800 elements, 1...

The console output for `summary(mod.2)` is:

```
1      2      3      4      5      6      7
5.421730 5.421730 5.421730 5.421730 15.482682 5.627056
> plot(mod.l$y, mod.l$fitted.values)
> head(predict(mod.1, int = "conf"))
      fit      lwr      upr
1  5.421730  4.437845  6.405615
2  5.421730  4.437845  6.405615
3  5.421730  4.437845  6.405615
4  5.421730  4.437845  6.405615
6  15.482682 14.152940 16.812425
7  5.627056  4.673657  6.580454
```

Middle Screenshot: This screenshot is identical to the top one, but the variable `CEC` in the `summary(subs.soil.data)` line is highlighted in blue.

Bottom Screenshot: The R console shows the following code and output:

```
401 head(predict(mod.1))
402 plot(mod.l$y, mod.l$fitted.values)
403 head(predict(mod.1, int = "conf"))
404
405
406 #MLR
407 subs.soil.data= soil.data[, c("clay", "CEC", "ExchNa", "ExchCa")
408 summary(subs.soil.data)
409 cor(na.omit(subs.soil.data)) #correlation
410 pairs(na.omit(subs.soil.data))
411 mod.2 <- lm(CEC ~ clay + ExchNa + ExchCa, data = subs.soil.data)
412 summary(mod.2)
413
414
415
```

The Environment pane on the right lists the following objects:

- @soil.data: 146 obs. of 16 variables
- @soil.data.cle: 133 obs. of 18 variables
- @soil.info: 6 obs. of 2 variables
- @soil.info1: 3 obs. of 2 variables
- @soil.info2: 3 obs. of 2 variables
- @soil.info3: 6 obs. of 3 variables
- @step.model: List of 13
- @subs.soil.data: 166 obs. of 4 variables

The console output for `cor(na.omit(subs.soil.data))` is:

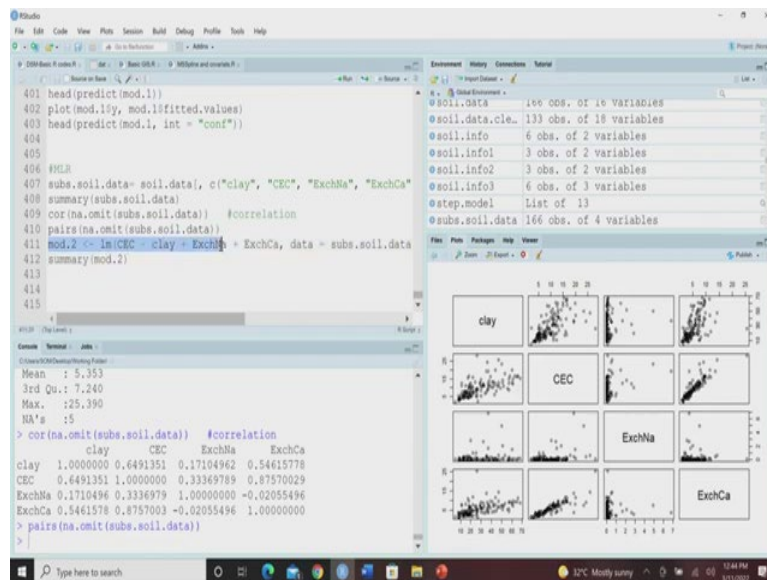
```
      clay      CEC      ExchNa      ExchCa
clay  1.0000000  0.6491351  0.17104962  0.54615778
CEC   0.6491351  1.0000000  0.33369789  0.87570029
ExchNa 0.1710496  0.3336979  1.00000000 -0.02055496
ExchCa 0.5461578  0.8757003 -0.02055496  1.00000000
> pairs(na.omit(subs.soil.data))
```

The Environment pane at the bottom shows a grid of scatter plots for the variables: clay, CEC, ExchNa, and ExchCa.

For the multiple linear regression again we are going to use this, so we are going to subset the soil data where we are combining clay, CEC, exchangeable sodium and exchangeable calcium for all the 166 observations. So, when all the rows are involved so we do not give anything and we are only combining these 4 or 5, 4 variables together.

So, we want to see the summary and these are the summary of all the variables and then we want to see the correlation the function is COR we want to omit and then do the correlation and you can see this is the correlation matrix and you can want, you also want to have a pair scatter plot using these pair commands and you can get this scatter plot. So, this is a clay versus CEC, clay versus exchangeable sodium, exchangeable sodium versus exchangeable calcium and so on. So, this type of paired scatter plot you can also get.

(Refer Slide Time: 24:21)



Model 2 is again linear model where CEC is being predicted using clay exchangeable sodium, exchangeable calcium and data is subset of the soil data and then we want to have, we want to have the summary statistics so you can see multiple R square is 0.90, adjusted R square is again 0.90 and also you can get the values of the offset as well as intercept and as well as the coefficient values and their significance levels also.

So, guys this is how you do the different types of data handling and basic very basic modeling in R for DSM. Now, let us move to basics of GIS.

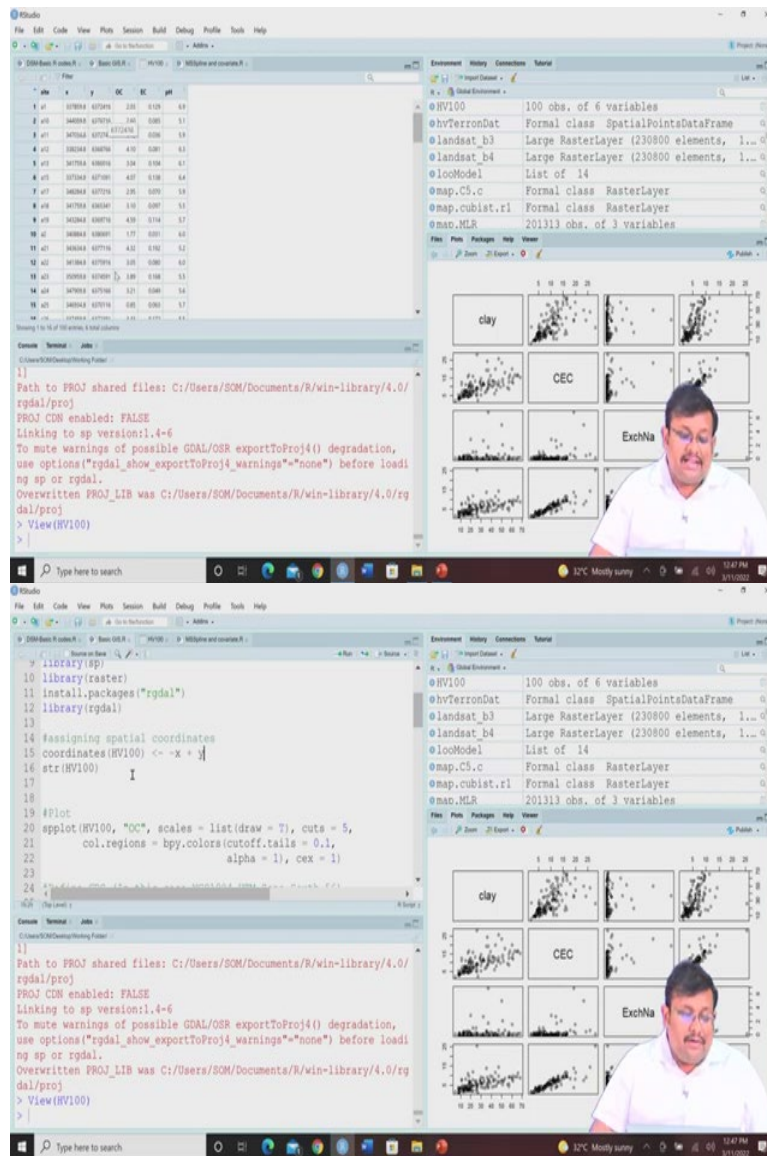
(Refer Slide Time: 25:08)

The image displays three sequential screenshots of an RStudio environment, illustrating the process of installing and using the `rgdal` package for GIS operations.

Top Screenshot: The R console shows the initial setup code, including loading `sp` and `raster` packages, installing `rgdal`, and assigning spatial coordinates. The Environment pane lists objects like `@mod.2`, `@mod.data`, and `@mod.rh`. The console output shows the results of a regression model, including coefficients for `clay`, `ExchNa`, and `ExchCa`, along with statistical metrics like the F-statistic and p-value.

Middle Screenshot: The R console shows the execution of `str(HV100)`, displaying the structure of the data frame with 100 observations and 6 variables. The Environment pane shows the loaded objects, including `@HV100` and various raster layers. A dialog box titled "Updating Loaded Packages" is visible, indicating that some packages need to be updated before installing `rgdal`.

Bottom Screenshot: The R console shows the successful installation of `rgdal` and the execution of `library(rgdal)`. The Environment pane shows the updated list of objects, including `@HV100` and `@mo.MLR`. The console output includes a message about the retirement of `rgdal` and the plan to transition to `sf` and `terra` functions.



Now, you already have the basic, you already know the basics of GIS and so again we are going to use this library `ithir` so I am calling this library `ithir` and here we are going to use this HV 100 data set actually this is the data, soil data, 100 soil data was collected from hunted valley of Australia so that is why the name is HV 100.

So, we can use this HV, so we let us consider this HV 100 data set and let us see the structure of this HV 100 data set. So, you will see that there are 100 observation of 6 variables and 100 observation like and there are x and y, x and y are basically coordinates and then organic carbon EC and then pH so these are the different variables.

So, we are going to load the necessary packages so we are going to install this package `sp` so we can install but I just skip it because it is already installed, so you can try and then you can call this library `sp`, it is basically doing some special operations, special and then library `raster` you can for dealing with the raster data and then you can use this `rgdal` package also so

this rgdal package will be helpful for doing the GIS operation I have already installed that so I am just going to upload this rgdal package.

Now, if you see this hunt HV 100 data set so this HV 100 dataset you can see these are different sites, their x and y coordinates and then organic carbon EC and ph. So, first of all we want to, this is a basically in a data, in a basically it is a tabular format. So, this table R will understand this as a table but we have to instruct R that these x and y are coordinates, only then the R will consider it as a special points data frame.

It is only simple data frame right now but when will instruct R that you should understand this x and y as the coordinates then they will understand that yes these are this is the whole data set frame is the spatial points data frame. So, we can we can use this HV, we can assign by these coordinates by using this coordinate function, so coordinate HV 100 we are telling R that x and y are the coordinates.

(Refer Slide Time: 28:02)

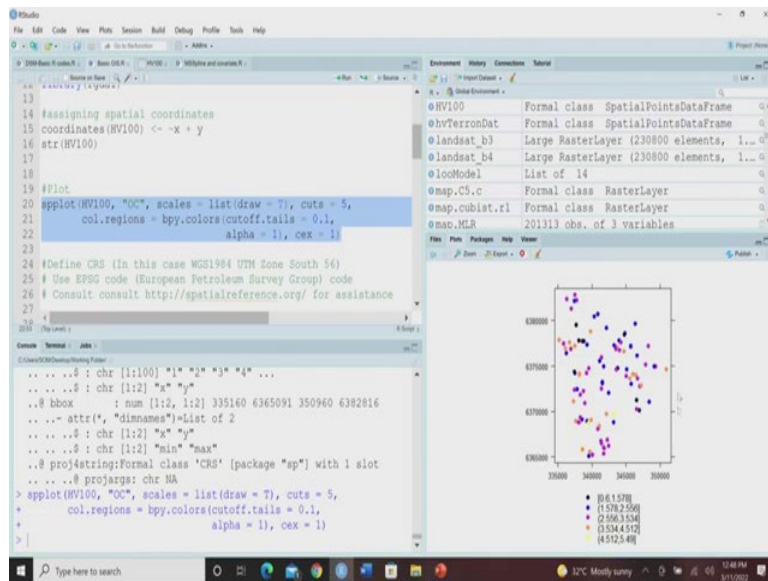
The screenshot displays an RStudio interface with the following components:

- Source Editor:** Contains R code for loading the `rgdal` package, assigning spatial coordinates to the `HV100` dataset, and plotting a spatial points data frame.
- Environment:** Lists loaded objects including `HV100` (SpatialPointsDataFrame), `ohvTerrorDat` (SpatialPointsDataFrame), `landsat_b3` (RasterLayer), `landsat_b4` (RasterLayer), `locoModel` (List), `omap.C5.c` (RasterLayer), `omap.cubist.ri` (RasterLayer), and `omao.MLR` (matrix).
- Console:** Shows the execution of the code, including the output of the `coordinates` function and the `str` function applied to `HV100`.
- Plots:** A grid of plots showing spatial data for variables like `clay`, `CEC`, and `ExoNa`.

```
library(rgdal)
install.packages("rgdal")
library(rgdal)
# Assigning spatial coordinates
coordinates(HV100) <- ~x + y
str(HV100)
# Plot
spplot(HV100, "OC", scales = list(draw = T), cuts = 5,
       col.regions = bpy.colors(cutoff.tails = 0.1,
                               alpha = 1), cex = 1)
```

Console output:

```
> coordinates(HV100) <- ~x + y
> str(HV100)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
..@ data      : 'data.frame': 100 obs. of  4 variables:
.. ..@ site: Factor w/ 100 levels "a1","a10","a11",...: 1 2 3 4 5 6
7 8 9 10 ...
.. ..@ OC  : num [1:100] 2.03 2.6 3.42 4.1 3.04 4.07 2.95 3.1 4.59
1.77 ...
.. ..@ EC  : num [1:100] 0.129 0.085 0.036 0.081 0.104 0.138 0.07
0.097 0.114 0.031 ...
.. ..@ pH  : num [1:100] 6.9 5.1 5.9 6.3 6.1 6.4 5.9 5.5 5.7 6 ...
.. ..@ coords.nrs : int [1:2] 2 3
```



Now, let us see the structure you will see now this is not appearing as a normal structure of what we generally see in case of any normal data frame. Here you can see it is a special points data frame since we have assigned these x and y as the coordinates then the R will understand that this is a, it is not a simple data frame, now it is a special points data frame.

Remember for doing all this GIS operation you need the special points data frame, for plotting you need the spatial points data frame. So, we are going to use this sp plot, sp plot using this HV 100 data set we are going to focus on this organic carbon, our scales are given and then colors which we have specified and the size of the markers are also given.

So, let us see, let us run it and you will see that this plot of organic carbon is appearing and based on their values from 0.6 to 1.5 we are getting black dots from 1.5 to 2.5 we are getting blue dots, from 2.5 to 3.53 we are getting pink dots and then orange dots from 3.5 to 4.5 so for different interval we are getting these, their colors are being coded.

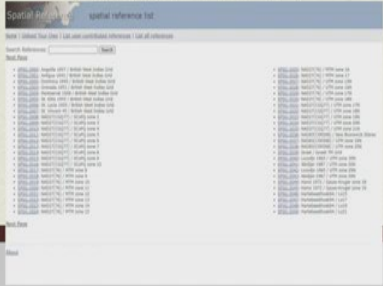
Now, the next important thing is how to define the coordinate reference system. R understand that it is a spatial points data frame but how to instruct R that this is the coordinate reference system, because unless we assign the coordinate reference system it will not be projected properly.

(Refer Slide Time: 30:23)


EPSG CODE

- EPSG stands for European Petroleum Survey Group and is an organization that maintains a geodetic parameter database with standard codes, the EPSG codes, for coordinate systems, and datums


<http://spatialreference.org/>



The screenshot shows the 'Spatial Reference List' page on the website. It features a search bar at the top and a table of coordinate systems. The table has columns for 'Name', 'Code', 'Datum', 'Spheroid', 'Projection', and 'Units'. The first few rows are: 'WGS 1984 UTM zone 56S', 'WGS 1984 UTM zone 56N', 'WGS 1984 UTM zone 57S', and 'WGS 1984 UTM zone 57N'. The 'Code' column contains the EPSG codes: 31466, 31467, 31468, and 31469 respectively.



The speaker icon shows a man in a white shirt speaking.

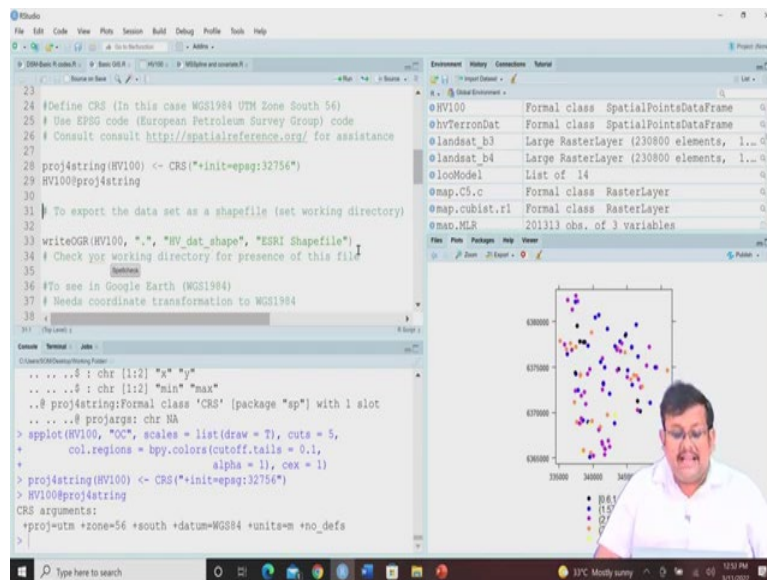
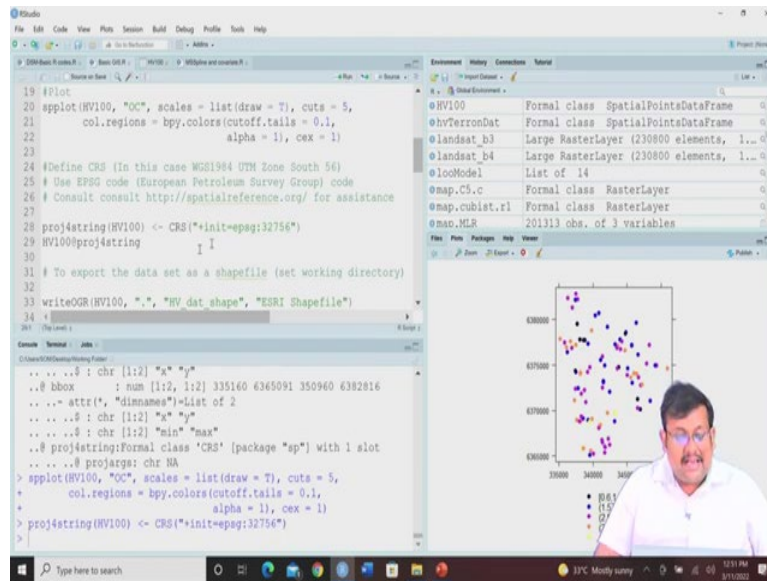


Logos of institutions are visible at the bottom right of the slide.

So, we have to define this coordinate reference system in this case this data set has the coordinate reference system of WGS 1984 UTM zone south 56. So, to define this coordinate reference system we are going to use the EPSG code. So, what are the EPSG code? EPSG stands for the European petroleum survey group and basically this is an organization of that maintains the geodetic parameter database with standard codes, so these are known as the EPSG codes for coordinate systems and datum.

So, you can check this website with the, and then you will see this special reference and you can get for all the different location what are the EPSG code. So, you can instead of giving all the details of the coordinate reference system and datum and all this thing you can specifically mention the EPSG code.

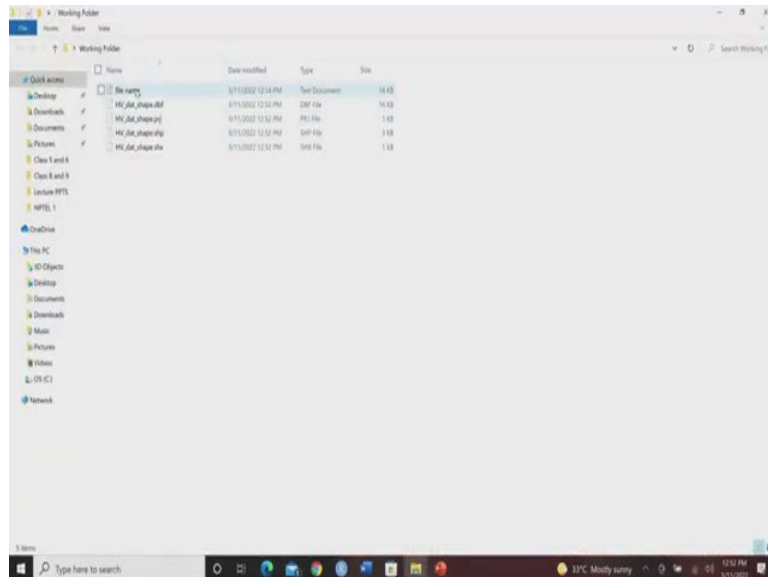
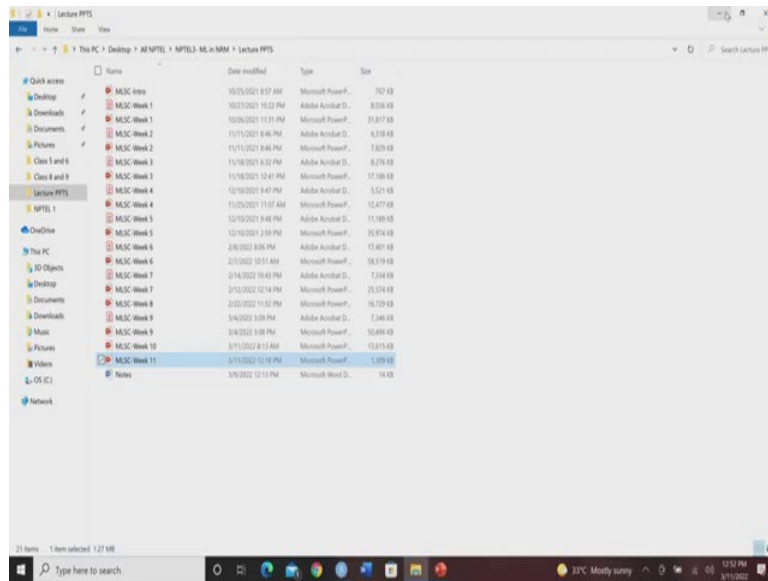
(Refer Slide Time: 31:06)



So, here you can see we are using this project for string function for this HV 100 and we are defining this CRS and then EPSG code is 32756, we are assigning and then let us see, now it has been already updated. So, you see that coordinate reference system it is UTM zone 56 south datum WGS 84 unit in meter. So, you can see that instead of giving all this information if you just give the EPSG code it will be a lot easier.

Now, we can export the data as a shape file, so our working directory will be there, this shape file or ESRI shape file will be saved, the vector shape file so this shape file will be saved in our directory so for that the function is write ogr function, here we are using this HV 100 data set, so the name of this created data, created file will be HV underscore dat underscore shape and it will be saved as a ESRI shape file, which is an universal format.

(Refer Slide Time: 32:26)



ESRI SHAPE FILE

- A shapefile is an Esri vector data storage format for storing the location, shape, and attributes of geographic features. It is stored as a set of related files and contains one feature class.

- *.shp: contains the feature geometries.
- *.dbf: contains feature attribute data, as a table.
- *.shx: indexation data for iterations across the features.
- *.prj: the coordinate reference system represented as text.

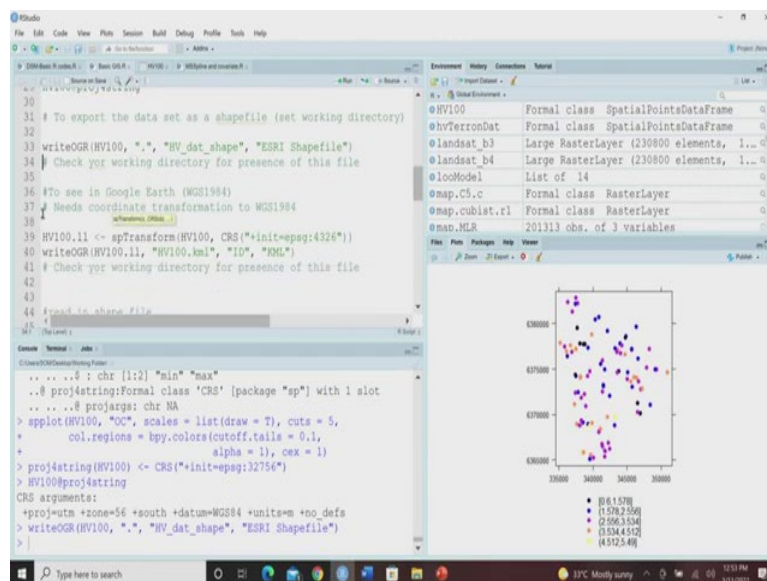


So, if you open this working directory you will see that the working folder that these files have been created, four files have been created like HV dot shape dot shp, dot shx, dot prj, dot dbf. Now, why these are, what are these four files? So, again these ESRI shape file is an ESRI vector data ESRI means those organization who have developed this GIS software called RGIS.

So, this shapefile is an ESRI vector data storage format for storing the location, shape and attribute of geographical features. It is stored as a set of related files so you can see four different types of file, shp file is showing the feature geometries, so it is basically point, polygons and lines.

So, dbf is the database file which is containing the attribute data, shx is a indexation data for iteration across the features that is index file and dot prj is the coordinate reference system represented as text file. So, these are four different files. So, once you create this shapefile, automatically these four files will be created. So, that is why you see in your working directory those four files have been created.

(Refer Slide Time: 33:52)



RStudio interface showing R code for exporting a data set as a shapefile and projecting it to WGS1984. The code includes comments and function calls like `writeOGR` and `spTransform`. The Environment pane on the right lists objects such as `HV100`, `landsat_b3`, and `landsat_b4`. A plot of the data points is visible in the bottom right corner.

```
30
31 # To export the data set as a shapefile (set working directory)
32
33 writeOGR(HV100, ".", "HV_dat_shape", "ESRI Shapefile")
34 # Check your working directory for presence of this file
35
36 # To see in Google Earth (WGS1984)
37 # Needs coordinate transformation to WGS1984
38
39 HV100.11 <- spTransform(HV100, CRS("+init=epsg:4326"))
40 writeOGR(HV100.11, "HV100.kml", "ID", "KML")
41 # Check your working directory for presence of this file
42
43
44 # read in shape file
```

Environment:


- @HV100: Formal class 'SpatialPointsDataFrame'
- @hvTerrorDat: Formal class 'SpatialPointsDataFrame'
- @landsat_b3: Large RasterLayer (230800 elements, 1..)
- @landsat_b4: Large RasterLayer (230800 elements, 1..)
- @looModel: List of 14
- @map.C5.c: Formal class 'RasterLayer'
- @map.cubist.rl: Formal class 'RasterLayer'
- @mas.MLR: 201313 obs. of 3 variables

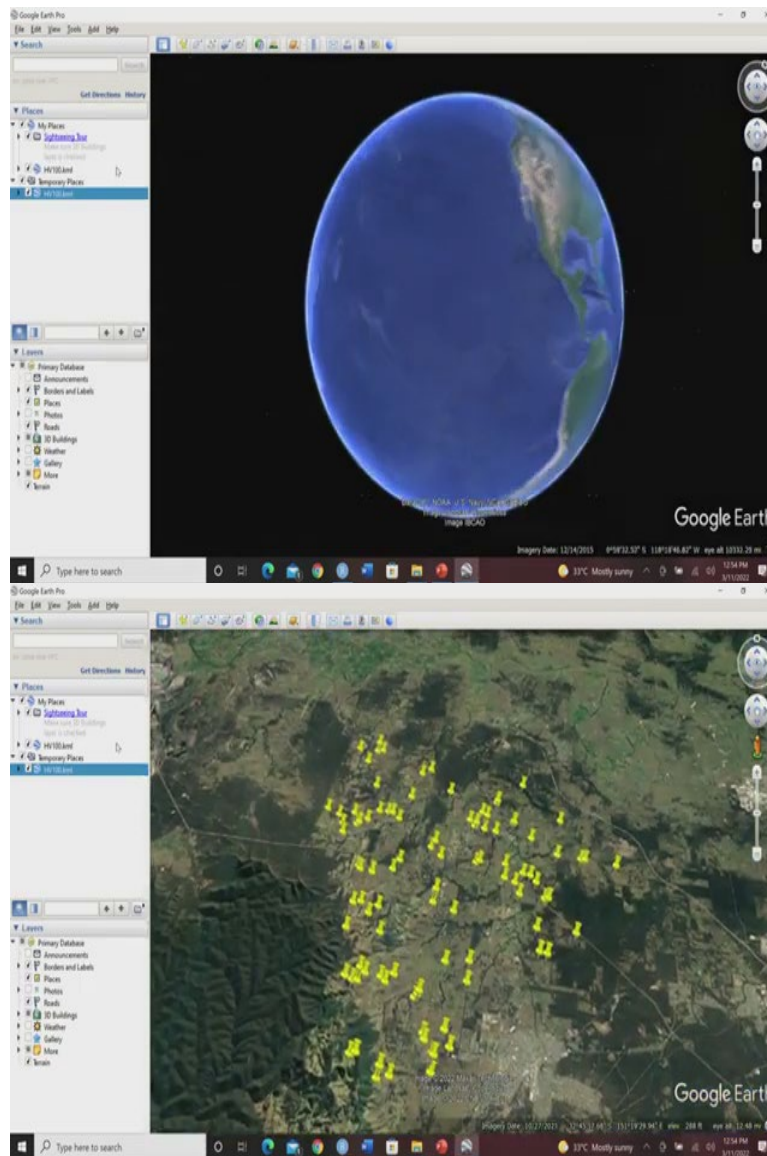
Plot: A scatter plot showing data points in a 2D space. The x-axis ranges from 33000 to 34000, and the y-axis ranges from 620000 to 627000. Points are colored by region, with a legend on the right showing categories B.6, B.7, B.8, and B.9.

Windows File Explorer showing the contents of a 'Working Folder'. The folder contains several files related to the R script, including `Bk name`, `HV_dat_shape.dbf`, `HV_dat_shape.prj`, `HV_dat_shape.shp`, and `HV100`. The `HV100` file is selected.

Name	Date modified	Type	Size
Bk name	3/11/2022 12:14 PM	Text Document	14 KB
HV_dat_shape.dbf	3/11/2022 12:52 PM	DBF File	16 KB
HV_dat_shape.prj	3/11/2022 12:52 PM	PRJ File	1 KB
HV_dat_shape.shp	3/11/2022 12:52 PM	SHP File	3 KB
HV100	3/11/2022 12:54 PM	KML	35 KB

Windows File Explorer showing the 'Working Folder' with a large 'Google Earth' logo overlaid on the image. The logo features the text 'Google Earth' above a view of the Earth from space.

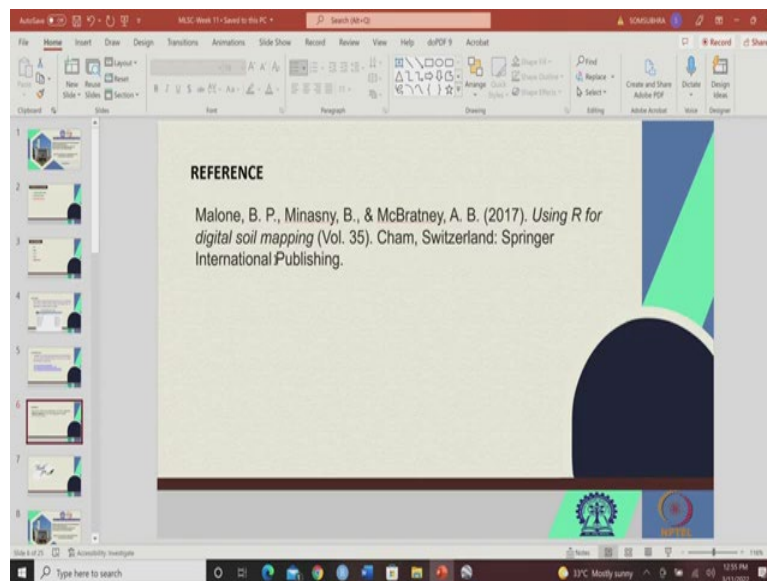




Now, if you want to see this file in a Google Earth, so you need further coordinate transformation to WGS 1984 because Google Earth can identify only dot kml file and for dot kml file you need the transformation to WGS 1984 only. So, for these WGS 1984 we are going to use the EPSG code of 4326 and then we are again using this writeOGR function to develop this HV 100 dot kml file.

So, let us see, let us run this codes and then let us see if this file has been created or not. Yes, this HV 100 file has been created. So, if you just click on it you will see that these points will be projected directly in the google earth surface. Now, you can see how exciting these operations are and you can directly project these sampling points over the Google Earth and so this type of representation is so fascinating that you can use it in any type of your academic presentation or, so this type of things you can do using the using the R software and also digital soil mapping will help you to understand all these things.

(Refer Slide Time: 35:36)



So, guys let us wrap up this thing, this is the reference again that book if you want to have more detailed information of these GIS operations you should consult that book and so I hope that you have gathered some important information from this lecture and let us wrap up here and we will see, we will again, will go from here.

So, these codes will be continuously discussed in throughout these upcoming lectures so wherever we left will start from there and then we will discuss further about these codes and how to utilize this code for different types of operation. Thank you very much.