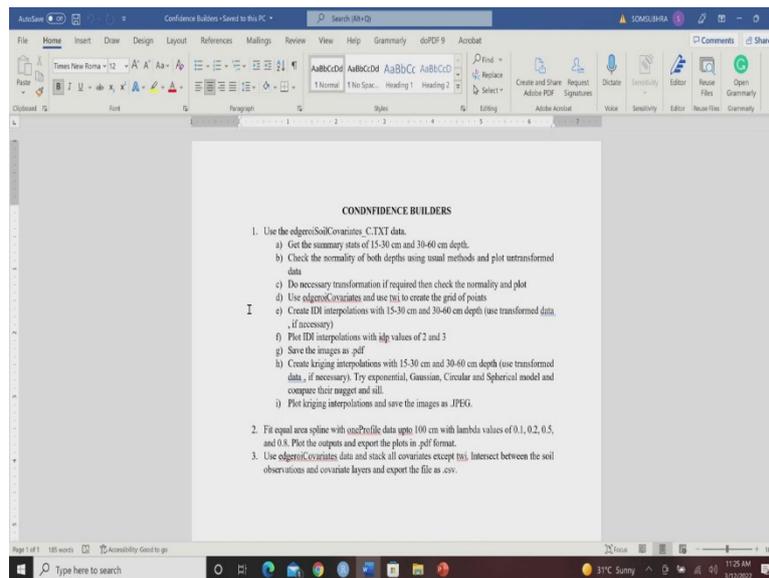


Machine Learning for Soil and Crop Management
Professor Somsubhra Chakraborty
Agricultural and Food Engineering Department
Indian Institute of Technology, Kharagpur
Lecture 54
Digital Soil Mapping with Continuous Variables (Contd.)

Welcome friends to this fourth lecture of week 11 of NPTEL online certification course of Machine Learning for Soil and Crop Management. And in this week, we are dealing with Digital Soil Mapping with Continuous Variables. And in our previous lectures, we have already seen how to deal with different types of... exploratory data analysis, basic GIS operation, basic geo statistical operation using R and also we have seen how to use R for producing the Kriging interpolation, IDI interpolation, all this. And I will be sharing all these codes with these annotations in the forum, but at the same time, I also have some practice set for you for boosting your confidence. So, that you can execute these codes for solving this question and feel more confident.

(Refer Slide Time: 01:23)



So, let me show you this is the these confidence builders which I have developed from you. So, you got you are going to use these edgeroiSoilCovariates underscore C dot txt data. Once you run all the codes which I have already shared, you can be able to get these edgeroiSoilCovariates underscore C dot txt data and then the first question is get the summary stats of 15 to 30 centimetre and 30 to 60 centimetre depth check the normality of both depths using usual methods and plot and transform data do necessary transformation if required, then check the normality and plot then edgeroiSoilCovariates and use twi to create

the grid of points and then create idea interpolation with 15 to 30 centimetre and 30 to 60 centimetre depth, use transform data if necessary.

And plot idea interpolation with idp values of 2 and 3 and save the image as dot pdf create Kriging interpolation with 15 to 30 and 30 to 60-centimetre depth use transform data if necessary, try exponential Gaussian circular spherical model and compare their nugget and sill. Plot Kriging interpolation and save the images as dot JPEG.

And the second question is fit equal area spline with oneProfile data up to 100 centimetre with lambda values of 0.1, 0.2, 0.5 and 0.8 plot the outputs and export the plots in dot pdf format. And then third question is used edgeroiCovariates data and stack all the covariates except twi intercept between the soil observation and covariate layers and export the file as dot csv.

So, whatever we have covered so far, you will see these you know, these confidence builders are basically showing an extract. So, you know, if you go through these confidence builders you will be able to perform all the tasks which we have already seen. So, please go through these questions, the problem set and solve it and once you solve it, then you will feel more confident about this DSM model application using R.

(Refer Slide Time: 03:51)

CONCEPTS COVERED

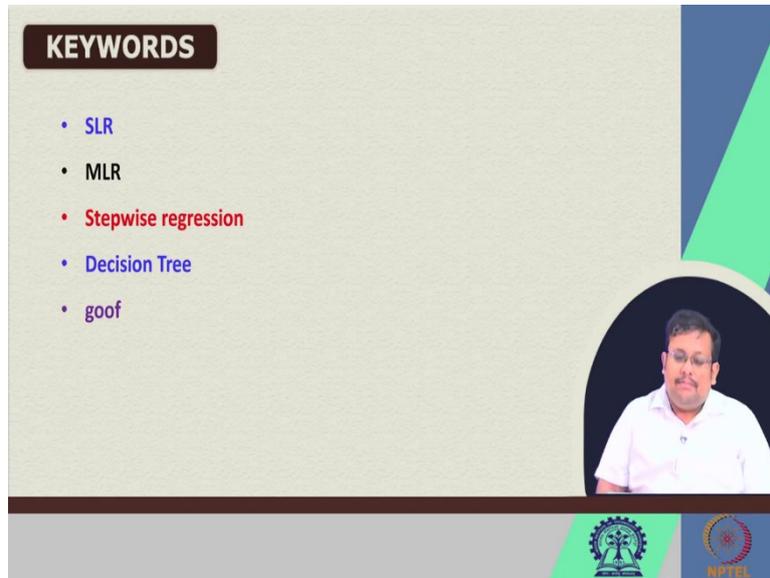
- Model validation
- SLR and MLR based mapping
- Stepwise regression based mapping
- Decision Tree based mapping

The slide features a video inset of a man in a white shirt speaking. At the bottom, there are logos for IIT Bombay and NPTEL.

Now, in this lecture we are going to cover these concepts, we are going to cover the model validation, how to do the model validation. We are going to start with the Simple Linear Regression, Multiple Linear Regression. And I am going to show you how to do the mapping based on this Simple Linear Regression and Multiple Linear Regression. Then, we are going

to see the stepwise regression-based mapping and then finally, we are going to talk about that decision tree-based mapping. So, decision tree is also known as the classification regression tree when it is a classification problem, then we call it a classification tree and when it is a regression problem, then we call it a regression tree.

(Refer Slide Time: 04:32)



So, the keywords which we are going to discuss are SLR, MLR, Stepwise regression, Decision Tree and goof. Goof is a function which we are going to discuss.

(Refer Slide Time: 04:44)

Model Validation

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (obs_i - pred_i)^2}{n}}$$

where *obs*=the observed soil property
pred =predicted soil property from a given model
n =the number of observations *i*.

Bias, also called the mean error of prediction and is defined as:

$$bias = \frac{\sum_{i=1}^n (pred_i - obs_i)}{n}$$

Pearson's correlation coefficient

$$r = \frac{\sum_{i=1}^n (obs_i - \bar{obs})(pred_i - \bar{pred})}{\sqrt{\sum_{i=1}^n (obs_i - \bar{obs})^2} \sqrt{\sum_{i=1}^n (pred_i - \bar{pred})^2}}$$

R²= coefficient of determination
 Lin's concordance correlation coefficient (Lin 1989):is a single statistic that both evaluates the accuracy and precision of the relationship. It is often referred to as the goodness of fit along a 45 degree line. Thus it is probably a more useful statistic than the R² alone

$$\rho_c = \frac{2\sigma_{pred} \sigma_{obs} \rho}{\sigma_{pred}^2 + \sigma_{obs}^2 + (\mu_{pred} - \mu_{obs})^2}$$

where μ_{pred} and μ_{obs} are the means of the predicted and observed values respectively. σ_{pred}^2 and σ_{obs}^2 are the corresponding variances. ρ is the correlation coefficient between the predictions and observations.

KEYWORDS

- SLR
- MLR
- Stepwise regression
- Decision Tree
- goof

So, remember this goof function we are going to use for goodness of fit... for identifying the goodness of fit statistics. So, the short form of this goof is, the short form is goodness of fit is goof and there are different types of model indicators which we are going to use you know about RMSE bias is basically the mean error.

So, it is predicted minus observed values for all the observation divided by n. So, an IBL model should have 0 bias and so, Pearson correlation coefficient we have already discussed, another model coefficient we are going to discuss is the Lin's concordance correlation coefficient or CCC. So, it is a single statistic that both evaluates the accuracy and precision of the relationship, it is often referred to as the goodness of fit along a 45-degree line. Thus, it is probably a more useful statistic than the R square alone.

So, this is the formula this is the Lin's concordance correlation coefficient and this is how you calculate it. So, $2\rho\sigma_{\text{pred}}\sigma_{\text{obs}}$ then $\sigma_{\text{pred}}^2 + \sigma_{\text{obs}}^2 + \mu_{\text{pred}} - \mu_{\text{obs}}$ whole square, so were these μ_{pred} and μ_{obs} are the measures of the predicted and observed values respectively and this σ_{pred}^2 and σ_{obs}^2 are the corresponding variances of the predicted values and observed values and ρ is the correlation coefficient between the predictors and the observations.

(Refer Slide Time: 06:34)

RStudio interface showing R code for simple linear regression. The code includes library calls for 'lme4', 'MASS', and 'USYD_soil', data loading, model fitting with `mod.l <- lm(CEC ~ clay, data = mod.data, y = TRUE, x = TRUE)`, and validation using the `goof` function. The Environment pane on the right lists objects like `@testdata` (200 obs. of 14 variables), `@tmp` (335 obs. of 3 variables), `@traindata` (306 obs. of 14 variables), `@twi` (Large RasterLayer), `@UK.P.map` (Large RasterLayer), `@UK.preds.V` (103 obs. of 4 variables), `@UK.Fvar.map` (Large RasterLayer), and `@USYD_soil1` (166 obs. of 16 variables).

RStudio interface showing a data table with columns: `PKBRI`, `Location`, `UpperDepth`, `LowerDepth`, `day`, `alk`, `acid`, `pH CaCl2`, `Total Carbon`, `EC`, `ESP`, `Enkha`. The Environment pane is the same as in the previous screenshot.

RStudio interface showing a detailed data table with columns: `th`, `LowerDepth`, `day`, `alk`, `acid`, `pH CaCl2`, `Total Carbon`, `EC`, `ESP`, `Enkha`, `Enkha`, `Enkha`, `Enkha`, `CEC`. The Environment pane is the same as in the previous screenshots.

```

1 # Simple Linear Regression
2 library(ithir)
3 library(MASS)
4 data(USYD_soil1)
5 soil.data <- USYD_soil1
6 mod.data <- na.omit(soil.data[, c("clay", "CEC")]) #omit miss
7 mod.l <- lm(CEC ~ clay, data = mod.data, y = TRUE, x = TRUE)
8 mod.l
9
10 # Now let us use goof function in ithir package for model valid
11 #type="DSM" will output only the R2, RMSE, MSE, bias and concor
12 #type="spec" for additional statistics
13
14 goof(observed = mod.data$CEC, predicted = mod.l$fitted.values,
15      type = "DSM")
16
17
18 data
19 > mod.l <- lm(CEC ~ clay, data = mod.data, y = TRUE, x = TRUE)
20 > mod.l
21
22 Call:
23 lm(formula = CEC ~ clay, data = mod.data, x = TRUE, y = TRUE)
24
25 Coefficients:
26 (Intercept)      clay
27      3.7791      0.2053
28
29
30

```

Now, let us go ahead and see how to execute this thing in R. So, we are going to start with the simple linear regression, simple linear regression again we are going to call the library ithir then library mass, the mass package we have already installed before then the data USYD soil 1 we are going to call it.

So, USYD soil 1 you know 166 observations 16 variables, we have previously seen these data profile, land class, upper depth, lower depth, clay, silt sand, pH CaCl2 total carbon EC, ESP, then extendable sodium, extendable potassium, extendable calcium, extendable magnesium and mineral magnesium and CEC, we know it already.

So, here we are going to first the omit the missing values. So, the missing values for omitting the missing values we are going to use these na dot omit function and we are going to keep only the clay and CEC because now, we are interested to predict the CEC based on the clay content. So, we are going to keep this only clay and CEC and now we are going to fit the linear model for fitting the linear model the function is lm and here we are targeting the cation exchange capacity by clay.

So, we are putting this sign to indicate that CEC is the target whereas clay is the predicted and our data is mod dot data. So, because mod dot data is basically the USYD soil 1 data after removing the missing values and so, let us run these and let us see what is the what are the model parameters. So, you can see the model parameters the intercept you are getting 3.77 and the slope of the clay is 0.20. So, from there you can have an idea about the model accuracy.

(Refer Slide Time: 08:30)

```
4 data[USYD_soil1]
5 soil.data <- USYD_soil1
6 mod.data <- na.omit(soil.data, c("clay", "CEC")) #omit miss
7 mod.l <- lm(CEC ~ clay, data = mod.data, y = TRUE, x = TRUE)
8 mod.l
9
10 # Now let us use goof function in ithr package for model valid
11 #type="DSM" will output only the R2, RMSE, MSE, bias and concor
12 #type="spec" for additional statistics
13
14 goof(observed = mod.data$CEC, predicted = mod.l$fitted.values,
15      type = "DSM")
16 goof(observed = mod.data$CEC, predicted = mod.l$fitted.values,
17      type = "spec")
18 #On average the predictions are 3.75 cmol(+)/kg off the true va
19
```

Call:
lm(formula = CEC ~ clay, data = mod.data, x = TRUE, y = TRUE)

Coefficients:
(Intercept) clay
3.7791 0.2053

Object	Class	Attributes
@mod.l	List of 14	
@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...	
@radK	Large RasterLayer (230800 elements, 1...	

```
> goof(observed = mod.data$CEC, predicted = mod.l$fitted.values,
+      type = "DSM")
+      R2 concordance MSE RMSE bias
1 0.4213764 0.5888521 14.11304 3.756733 0
```

```
> goof(observed = mod.data$CEC, predicted = mod.l$fitted.values,
+      type = "spec")
+      R2 concordance MSE RMSE bias MSEc RMSEc
1 0.4213764 0.5888521 14.11304 3.756733 0 14.11304 3.756733
+      RPD RPIQ
1 1.31915 1.774148
```

Now, we are going to use this goof function in ithir package for model validation. Now, when we use the goof function, there is a specific argument where we can either type DSM or we can use the spec. So, if we use the DSM then only it will produce 4 to 5 different... model indicators like R square, RMSE and then and also the R square, RMSE, RPD and bias but in case of you are using the spec as a you know in the type argument then you will have more and more results.

So, if you are using again the DSM you will get R square, RMSE, MSE and bias for concur and concordance correlation statistics, but in case of spec you will be getting the additional statistics. So, let us see how this looks like. So, we are going to use this goof function our observed is mod dot data and we are specifying CEC our predicted values our mod fitted values and that type is DSM.

So, the... so, let us run this. So, you can see R square is 0.42 and we are getting the concordance correlation coefficient 0.58 MSE 14.11 RMSE 3.75 and bias 0 and if we use the spec, so you can see here we are using the DSM in the type argument, but if we want to use the spec, you will see that it will give you more statistical output. So, R square concurred MSE, RMSE, bias MSE calibration RMSE calibration RPD and RPIQ.

Now, since we have not divided the data set into calibration validation, so whole data set has been already killed you know, they have they have calculated they have considered R... have considered that the whole dataset as calibration as well as validation file. So, you will see both calibration and validation will give you the same results. So, here the MSE of the validation is 14.11 also MSE of calibration is 14.11. Then RMSE 3.75 here also you can get 3.75 RPD 1.31 and RPIQ 1.77.

So, from this result we can see that the on average the prediction of R 3.75 centimole per kg of to the true value and the model on average is neither over or undefeated, if you highs CEC values are influencing the concordance and R square. So, this is the interpretation from this from this simple linear regression.

(Refer Slide Time: 11:40)

```
17 type = "spec")
18 #On average the predictions are 3.75 cmol(+)/kg off the true va
19 #The model on average is neither over- nor under-predictive
20 #A few high CEC values are influencing the concordance and R2.
21
22
23 #####Model Validation#####
24
25 #Sub-setting into calibration and validation sets
26 set.seed(123)
27 training <- sample(nrow(mod.data), 0.7 * nrow(mod.data))
28 training
29
30 # fit the calibration model
31 mod.rh <- lm(CEC ~ clay, data = mod.data[training, ],
32
```

```
Model Validation:
+
+ type = "DSM"
+ R2 concordance MSE RMSE bias
1 0.4213764 0.5888521 14.11304 3.756733 0
+ goof(observed = mod.data$CEC, predicted = mod.lsfitted.values,
+ type = "spec")
+ R2 concordance MSE RMSE bias MSEc RMSec
1 0.4213764 0.5888521 14.11304 3.756733 0 14.11304 3.756733
RSD REIQ
1 1.31915 1.774148
+ #Sub-setting into calibration and validation sets
+ set.seed(123)
+ >
```

Object	Class	Details
@mod.l	List of 14	
@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...	
@radk	Large RasterLayer (230800 elements, 1...	

```
29 #Sub-setting into calibration and validation sets
26 set.seed(123)
27 training <- sample(nrow(mod.data), 0.7 * nrow(mod.data))
28 training
29
30 # fit the calibration model
31 mod.rh <- lm(CEC ~ clay, data = mod.data[training, ],
32 y = TRUE, x = TRUE)
33 #evaluate the calibration model
34 goof(predicted = mod.rh$fit.values,
35 observed = mod.data$CEC[training])
36 # validation performance
37 mod.rh.V <- predict(mod.rh, mod.data[-training, ])
38 goof(predicted = mod.rh.V, observed = mod.data$CEC[-training],p
39
40 #A few of the high observed values contribute greatly to the va
41 # outlier removal and bootstrapping
42
43 #bootcv
44
```

```
Model Validation:
+
+ #Sub-setting into calibration and validation sets
+ set.seed(123)
+ training <- sample(nrow(mod.data), 0.7 * nrow(mod.data))
+ training
+ [1] 42 115 59 127 134 7 74 125 77 63 131 62 91 138 14
+ [16] 118 32 6 146 122 113 87 80 123 124 86 66 71 35 18
+ [31] 112 104 79 90 3 54 84 24 136 25 16 44 105 38 106
+ [46] 15 109 47 27 110 5 43 76 12 52 19 93 68 114 33
+ [61] 58 9 139 23 67 37 65 143 135 34 121 48 53 1 108
+ [76] 102 98 95 100 8 17 145 70 50 141 64 60 140 92 10
+ [91] 82 36 142 72 120 57 40 96 83 107 28 101
+ >
```

Object	Class	Details
@pred.nv.MNLR	factor w/ 12 levels "1","2","3","4",...	
@pred.l	Named num [1:101] 1.18 4.27 2.1 1.27 1.1...	
@RF.pred.C	Named num [1:238] 2.85 3.16 3.21 2.98 2...	
@RF.pred.V	Named num [1:103] 2.51 2.46 2.51 2.42 2...	
@RK.preds.fin	num [1:103] 2.54 2.52 2.48 2.54 2.7 ...	
@RT.pred.C	Named num [1:238] 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...	
@training	int [1:102] 42 115 59 127 134 74 125 7...	

```
29
30 # fit the calibration model
31 mod.rh <- lm(CEC ~ clay, data = mod.data[training, ],
32 y = TRUE, x = TRUE)
33 #evaluate the calibration model
34 goof(predicted = mod.rh$fit.values,
35 observed = mod.data$CEC[training])
36 # validation performance
37 mod.rh.V <- predict(mod.rh, mod.data[-training, ])
38 goof(predicted = mod.rh.V, observed = mod.data$CEC[-training],p
39
40 #A few of the high observed values contribute greatly to the va
41 # outlier removal and bootstrapping
42
43 #bootcv
44
```

```
Model Validation:
+
+ mod.rh <- lm(CEC ~ clay, data = mod.data[training, ],
+ y = TRUE, x = TRUE)
+ goof(predicted = mod.rh$fit.values,
+ observed = mod.data$CEC[training])
+ R2 concordance MSE RMSE bias
1 0.4512779 0.6158071 12.31952 3.509917 0
+ >
```

Object	Class	Details
@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...	
@rat	8 obs. of 2 variables	
@RK.preds.V	103 obs. of 4 variables	

```

29
30 |
31 data = mod.data[training, ],
32 TRUE)
33 model
34 fitted.values,
35 EC(training)
36
37 , mod.data[-training, ]
38 observed = mod.data$CEC[-training], plot.it = TRUE)
39
40 d values contribute greatly to the validation diagnostics
41 strapping
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

```

```

> mod.rh <- lm(CEC ~ clay, data = mod.data[training, ],
+             y = TRUE, x = TRUE)
> goof(predicted = mod.rh$fitted.values,
+      observed = mod.data$CEC[training])
+
+      R2 concordance      MSE      RMSE bias
+ 1 0.4512779 0.6158071 12.31952 3.509917 0
> # validation performance
> mod.rh.V <- predict(mod.rh, mod.data[-training, ])
>

```

Environment

Global Environment	num	[1:105]	2.53 2.91 2.47 2.52 2.69 ...
mod.rh	lm		
mod.rh.V	num	[1:146]	5.42 5.45 5.47 5.47 15.5 ...
MLR.pred.rhC	Named num	[1:238]	2.86 3.11 3.18 2.86 2...
MLR.pred.rhV	Named num	[1:103]	2.53 2.55 2.52 2.52 2...
mod.rh.V	Named num	[1:44]	5.28 5.28 5.69 7.73 12...

Console

```

[61] 58 9 139 23 67 37 65 143 135 34 121 48 53 1 108
[76] 102 98 95 100 8 17 145 70 50 141 64 60 140 92 10
[91] 82 36 142 72 120 57 40 96 83 107 28 101
> mod.rh <- lm(CEC ~ clay, data = mod.data[training, ],
+             y = TRUE, x = TRUE)
> goof(predicted = mod.rh$fitted.values,
+      observed = mod.data$CEC[training])
+
+      R2 concordance      MSE      RMSE bias
+ 1 0.4512779 0.6158071 12.31952 3.509917 0
> # validation performance
> mod.rh.V <- predict(mod.rh, mod.data[-training, ])
>

```

Now, so here we did not do any kind of model validation, but if we want to do the model validation separately, so for that we need to divide the data into calibration and validation states. So, for calibration and validation set identification, we are going to randomly divide the data set into calibration file and validation file. So, for that we are going to use the set seed comment.

So, set seed 123. So, set seed 123 basically when we use the set seed, set dot seed 123 it is these values specifies the initial value of the random number seed. So, here we are using these 123, that means 123 is set as the random number value and the main point of using the seed is to be able to reproduce a particular sequence of random numbers.

So, if you want to change the sequence of random numbers, you can use other seeds also. So, here we are using these 123 as a sequence for producing these random number seed and then you can see we are separating the data, we are first selecting the data called training data and training data we are we are we are separating the data 70 percent of the total data. So, for separating the 70 percent of the total data, we just have to multiply with 0.7, 0.7 means 70 percent with the number of rows number of rows means the number of samples.

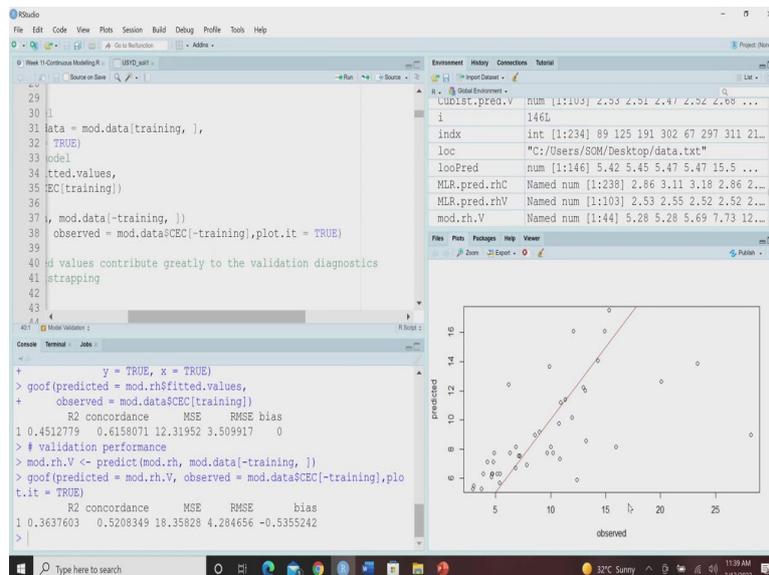
So, when we multiply the number of samples with 0.7 that will give you that number of training data. So, we are getting that and then we are we are we want to see how this training data will look like. So, here you can see in that training data set these are the observations so 40 second observation, 115th observation, 59th observation, 127th observation, 134. So, these are selected very randomly, so okay.

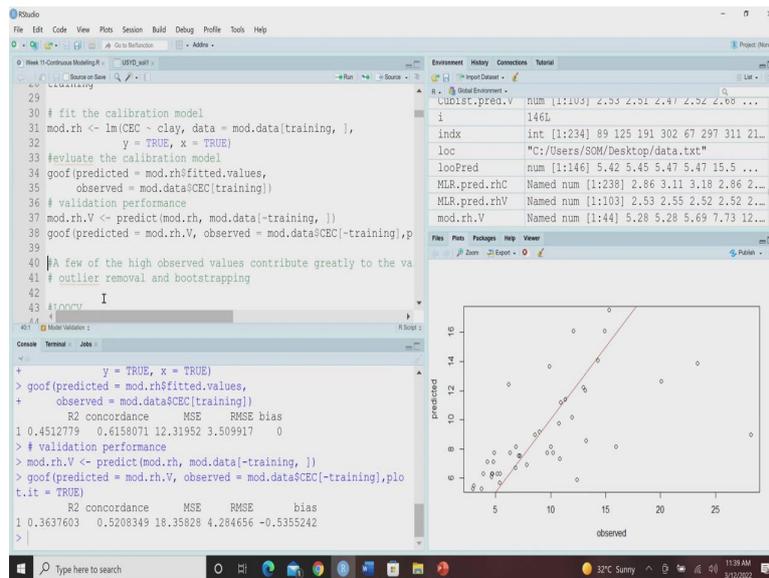
Now, once we have divided the world, we have selected the calibration model, let us fit the calibration model using this linear model. So, here again, the CEC is our target clay is our predictor, our data is mod dot data. And we are focusing now only with the training dataset for all the columns. So, all the variables are included. So, let us run it first.

And then we will let us run these goof function for these predicted values and missing values and you can see this is the model calibration statistics. So, we are getting R square concordance MSE RMSE bias. Now, we are going to see the validation performance. So, for validation performance again, same thing, except we are using minus training.

So, minus training means if we have the total number of samples, and if we subtract the number of training samples on or the specific training samples, then we will be left with only 30 percent of the data that is the testing samples. So, these testing samples or the validation samples we are going to use and then we are going to predict for this testing samples and the goof function we are going to use, now... you see one thing, that the argument of goofy is you have to give predicted values and observed values that is it. So, here we are giving the predicted values which we have already predicted in our previous step, observed values is already you know, this mod dot data minus training. And then you want to plot that also. So, if you want to plot then you can use this plot is true.

(Refer Slide Time: 15:21)





So, here you can see this is the model validation results R squared 0.36 concordance 0.52 and then MSE is 18.35 RMSE 4.28 bias is minus 0.53. And this is a predicted versus observed results. So, you can see from this calibration statistics and validation statistics is that a few of this high and also from this plot that a few of these high observed values contribute greatly. So, here here here, so these high observed values contribute greatly to the validation diagnostics.

So, whatever we are getting here 0.36 they may be so, this low comparatively low validation score could be due to these high values. These are the validation samples these are the 30 percent validation samples. So, how to deal with this problem, so they may be some outliers. So, you can see there some unex, abnormal higher values or outliers. So, what are the two ways to deal with this type of problem either you can go with the outlier removal or you can go with the bootstrapping, both the methods.

(Refer Slide Time: 16:40)

The screenshot shows an R script in RStudio. The script performs LOOCV by iterating over each row of the data, fitting a model, and predicting the value for that row. The console output shows the results of the LOOCV process, including R-squared, concordance, MSE, RMSE, and bias. A scatter plot on the right shows predicted values on the y-axis and observed values on the x-axis, with a diagonal line representing perfect prediction. The plot shows a strong positive correlation between observed and predicted values.

```
35 observed = mod.data$CEC[training])
36 # validation performance
37 mod.rh.V <- predict(mod.rh, mod.data[-training, ])
38 goof(predicted = mod.rh.V, observed = mod.data$CEC[-training], plo
39
40 #A few of the high observed values contribute greatly to the va
41 # outlier removal and bootstrapping
42
43 #LOOCV
44
45 looPred <- numeric(nrow(mod.data))
46 for (i in 1:nrow(mod.data)) {
47   looModel <- lm(CEC ~ clay, data = mod.data[-i, ], y = TRUE,
48                 x = TRUE)
49   looPred[i] <- predict(looModel, newdata = mod.data[i, ])
50 }
51
52 #Assess the performance of LOOCV
53 goof(predicted = looPred, observed = mod.data$CEC, plot.it = TRUE)
54
55 # LOOCV is less sensitive to outliers
56
```

	R2 concordance	MSE	RMSE	bias	
> goof(predicted = mod.rh.V, observed = mod.data\$CEC[training], plot.it = TRUE)	1 0.4512779	0.6158071	12.31952	3.509917	0
> goof(predicted = looPred, observed = mod.data\$CEC[-training], plot.it = TRUE)	1 0.3637603	0.5208349	18.35828	4.284656	-0.5355242

The screenshot shows an R script in RStudio. The script performs LOOCV by iterating over each row of the data, fitting a model, and predicting the value for that row. The console output shows the results of the LOOCV process, including R-squared, concordance, MSE, RMSE, and bias. A scatter plot on the right shows predicted values on the y-axis and observed values on the x-axis, with a diagonal line representing perfect prediction. The plot shows a strong positive correlation between observed and predicted values.

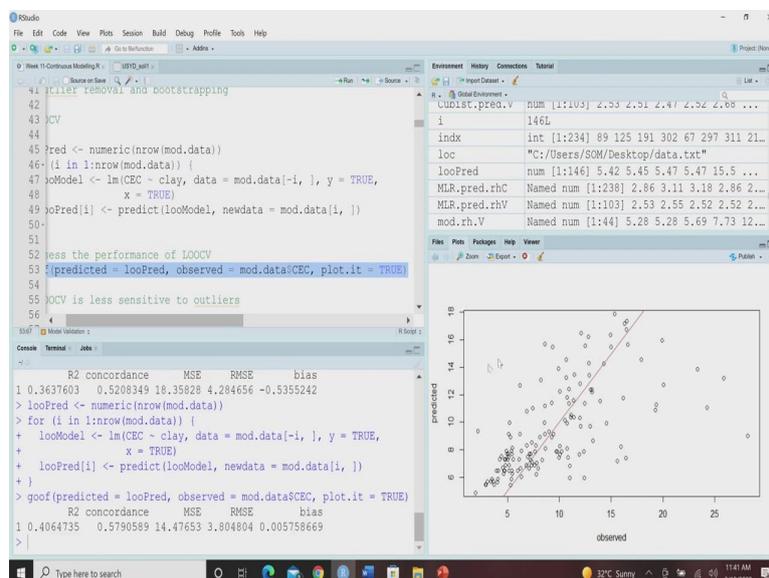
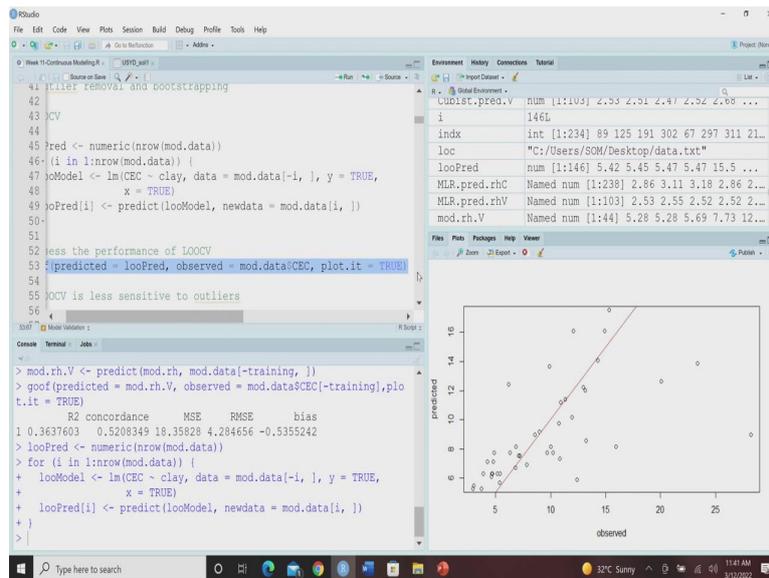
```
38 goof(predicted = mod.rh.V, observed = mod.data$CEC[-training], plo
39
40 #A few of the high observed values contribute greatly to the va
41 # outlier removal and bootstrapping
42
43 #LOOCV
44
45 looPred <- numeric(nrow(mod.data))
46 for (i in 1:nrow(mod.data)) {
47   looModel <- lm(CEC ~ clay, data = mod.data[-i, ], y = TRUE,
48                 x = TRUE)
49   looPred[i] <- predict(looModel, newdata = mod.data[i, ])
50 }
51
52 #Assess the performance of LOOCV
53 goof(predicted = looPred, observed = mod.data$CEC, plot.it = TRUE)
54
55 # LOOCV is less sensitive to outliers
56
```

	R2 concordance	MSE	RMSE	bias	
> goof(predicted = mod.rh.V, observed = mod.data\$CEC[training], plot.it = TRUE)	1 0.3637603	0.5208349	18.35828	4.284656	-0.5355242
> goof(predicted = looPred, observed = mod.data\$CEC[-training], plot.it = TRUE)	1 0.4512779	0.6158071	12.31952	3.509917	0

The screenshot shows an R script in RStudio. The script performs LOOCV by iterating over each row of the data, fitting a model, and predicting the value for that row. The console output shows the results of the LOOCV process, including R-squared, concordance, MSE, RMSE, and bias. A scatter plot on the right shows predicted values on the y-axis and observed values on the x-axis, with a diagonal line representing perfect prediction. The plot shows a strong positive correlation between observed and predicted values.

```
41 # outlier removal and bootstrapping
42
43 #LOOCV
44
45 looPred <- numeric(nrow(mod.data))
46 for (i in 1:nrow(mod.data)) {
47   looModel <- lm(CEC ~ clay, data = mod.data[-i, ], y = TRUE,
48                 x = TRUE)
49   looPred[i] <- predict(looModel, newdata = mod.data[i, ])
50 }
51
52 #Assess the performance of LOOCV
53 goof(predicted = looPred, observed = mod.data$CEC, plot.it = TRUE)
54
55 # LOOCV is less sensitive to outliers
56
```

	R2 concordance	MSE	RMSE	bias	
> goof(predicted = mod.rh.V, observed = mod.data\$CEC[training], plot.it = TRUE)	1 0.3637603	0.5208349	18.35828	4.284656	-0.5355242
> goof(predicted = looPred, observed = mod.data\$CEC[-training], plot.it = TRUE)	1 0.4512779	0.6158071	12.31952	3.509917	0

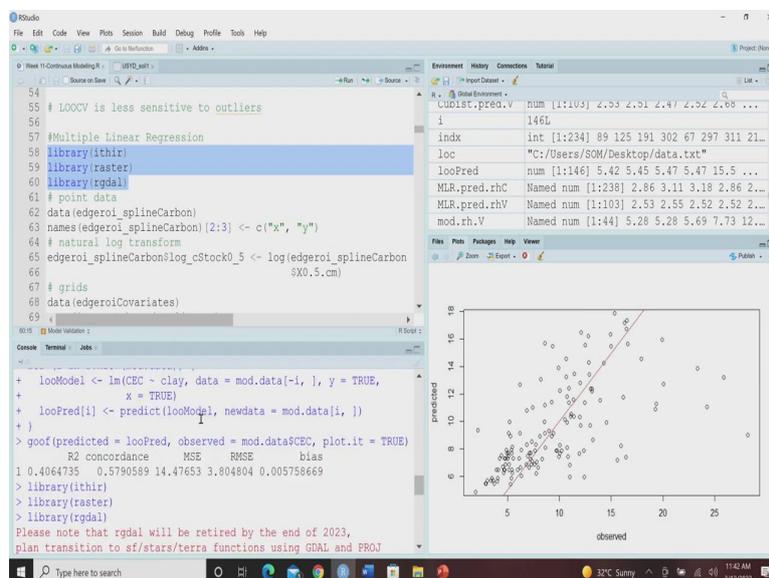
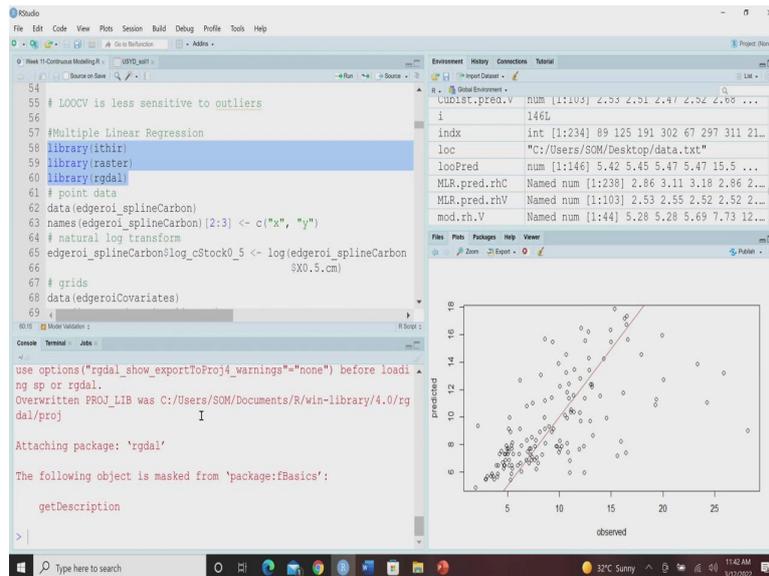


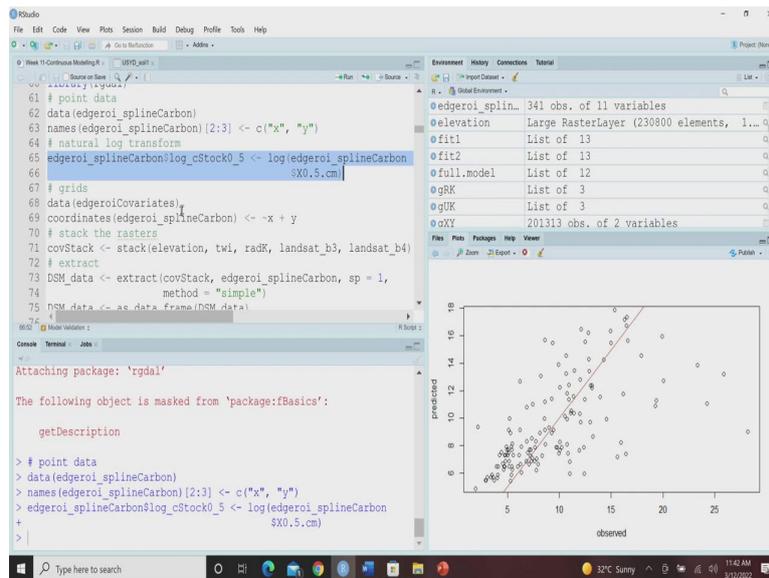
Now, guys you have seen that how we have done the you know model fitting without dividing the data into calibration validation. Now, you have seen how we can do that using the separate training set and testing set. Now, what about doing this model validation using leave one out cross validation. So, this scrip stands for leave one out cross validation. So, you will repeat this step in the steps in loop for each and individual sample and we get the leave one out cross validation prediction parameters.

So, I am going to run the script. So, the predicted values for this leave one out cross validation has been already selected has been already executed and we have stored now, again we are using this goof function our predicted is the LW a looPred value observed is mod dot data and specific to CEC and then we are also want to plot it.

So, here are all the 166 observations we have seen and then you can see this is the leave one out cross validation results. So, here you can see the results 0.40, 0.57 is a concordance MSE is 14.47, RMSE is 3.80 and then bias is 0.05. So, remember that that leave one out cross validation is less sensitive to outliers. So, you can use this leave one out cross validation. So, guys, we are done with the simple linear regression, let us go with the multiple linear regression.

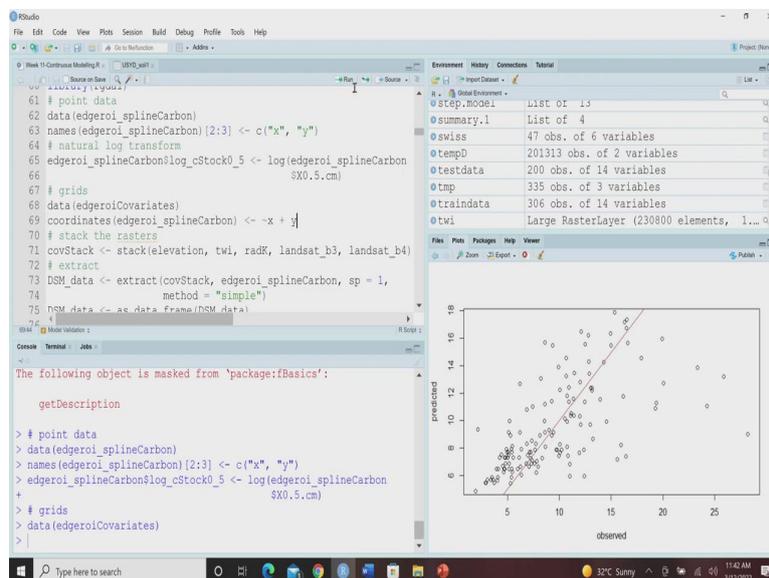
(Refer Slide Time: 18:21)





So, again, we are going to run this library, either library, raster library, rgdal let us call this point data that is that is so, we have already uploaded these packages. Now, let us download the data edgeroi underscore splineCarbon names and then we instruct R that the second and third column are x, y or location, then we do the natural log transformation of the 0 to 5 centimetre organic carbon data, then we need to take the grids.

(Refer Slide Time: 19:02)



RStudio interface showing R code and environment. The code includes:

```

64 natural log transform
65 edgeroi_splineCarbon$log_cStock0_5 <- log(edgeroi_splineCarbon
66 $X0.5.cm)
67 grids
68 data(edgeroiCovariates)
69 coordinates(edgeroi_splineCarbon) <- ~x + y
70 stack the rasters
71 covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
72 extract
73 DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
74 method = "simple")
75 DSM_data <- as.data.frame(DSM_data)
76 str(DSM_data)
77
78 #reduce the dataframe to what is necessary
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

The Environment pane shows:

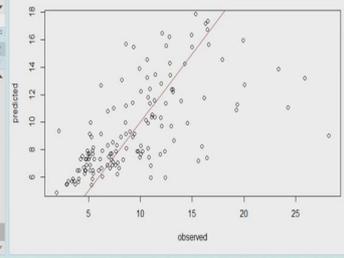
- edgeroi_spline... Formal class SpatialPointsDataFrame
- elevation Large RasterLayer (230800 elements, 1...
- fit1 List of 13
- fit2 List of 13
- full.model List of 12
- gRK List of 3
- gUK List of 3
- oXY 201313 obs. of 2 variables

The Console shows the output of `getDescription`:

```

> # point data
> data(edgeroi_splineCarbon)
> names(edgeroi_splineCarbon)[2:3] <- c("x", "y")
> edgeroi_splineCarbon$log_cStock0_5 <- log(edgeroi_splineCarbon
+ $X0.5.cm)
> # grids
> data(edgeroiCovariates)
> coordinates(edgeroi_splineCarbon) <- ~x + y
>

```



The plot shows predicted values on the y-axis (ranging from 0 to 18) and observed values on the x-axis (ranging from 0 to 25). A diagonal line represents the 1:1 relationship. Most data points are clustered below the diagonal line, indicating that the model tends to predict lower values than what was observed.

RStudio interface showing R code and environment. The code includes:

```

64 # natural log transform
65 edgeroi_splineCarbon$log_cStock0_5 <- log(edgeroi_splineCarbon
66 $X0.5.cm)
67 # grids
68 data(edgeroiCovariates)
69 coordinates(edgeroi_splineCarbon) <- ~x + y
70 # stack the rasters
71 covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
72 # extract
73 DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
74 method = "simple")
75 DSM_data <- as.data.frame(DSM_data)
76 str(DSM_data)
77
78 #reduce the dataframe to what is necessary
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

The Environment pane shows:

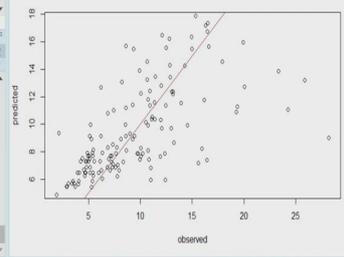
- covStack Large RasterStack (1154000 elements, 9...
- odat 234 obs. of 3 variables
- odat1 234 obs. of 3 variables
- odat2 101 obs. of 3 variables
- DSM_data 341 obs. of 16 variables
- edge.cub.Exp List of 14
- edge.dat 341 obs. of 16 variables
- edge.MLR.Full List of 12

The Console shows the output of `getDescription`:

```

> # point data
> data(edgeroi_splineCarbon)
> names(edgeroi_splineCarbon)[2:3] <- c("x", "y")
> edgeroi_splineCarbon$log_cStock0_5 <- log(edgeroi_splineCarbon
+ $X0.5.cm)
> # grids
> data(edgeroiCovariates)
> coordinates(edgeroi_splineCarbon) <- ~x + y
> covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
>

```



The plot shows predicted values on the y-axis (ranging from 0 to 18) and observed values on the x-axis (ranging from 0 to 25). A diagonal line represents the 1:1 relationship. The distribution of points is similar to the first screenshot, with a concentration of points below the diagonal line.

RStudio interface showing R code and environment. The code includes:

```

63 names(edgeroi_splineCarbon)[2:3] <- c("x", "y")
64 # natural log transform
65 edgeroi_splineCarbon$log_cStock0_5 <- log(edgeroi_splineCarbon
66 $X0.5.cm)
67 # grids
68 data(edgeroiCovariates)
69 coordinates(edgeroi_splineCarbon) <- ~x + y
70 # stack the rasters
71 covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
72 # extract
73 DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
74 method = "simple")
75 DSM_data <- as.data.frame(DSM_data)
76 str(DSM_data)
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

The Environment pane shows:

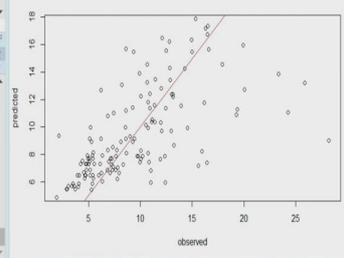
- covStack Large RasterStack (1154000 elements, 9...
- odat 234 obs. of 3 variables
- odat1 234 obs. of 3 variables
- odat2 101 obs. of 3 variables
- DSM_data 341 obs. of 16 variables
- edge.cub.Exp List of 14
- edge.dat 341 obs. of 16 variables
- edge.MLR.Full List of 12

The Console shows the output of `getDescription`:

```

> # point data
> data(edgeroi_splineCarbon)
> names(edgeroi_splineCarbon)[2:3] <- c("x", "y")
> edgeroi_splineCarbon$log_cStock0_5 <- log(edgeroi_splineCarbon
+ $X0.5.cm)
> # grids
> data(edgeroiCovariates)
> coordinates(edgeroi_splineCarbon) <- ~x + y
> covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
>

```



The plot shows predicted values on the y-axis (ranging from 0 to 18) and observed values on the x-axis (ranging from 0 to 25). A diagonal line represents the 1:1 relationship. The distribution of points is similar to the previous screenshots, with a concentration of points below the diagonal line.

So, for the grids we are going to download the edgeroi covariates we have already seen how to do that then we want to have the coordinate we want to instruct R that these x and y are basically the coordinates. So, we run that now, you can see here you know now it has been considered as a special points data frame it is not a simple data frame right now, these edgeroi underscore splineCarbon dataset.

Now, we are doing the stacking of the raster using the stack function. So, again just like previously we are doing the stacking and then we are extracting based on the edgeroi splineCarbon data using simple method. We have already executed this code previously in our previous in our previous lectures. So, we are going to use this... stacked covariate and then we are going to extract the values from the stack covariates for all the five covariates using simple method and then we are creating the whole data frame that is called DSM underscore data.

(Refer Slide Time: 20:10)

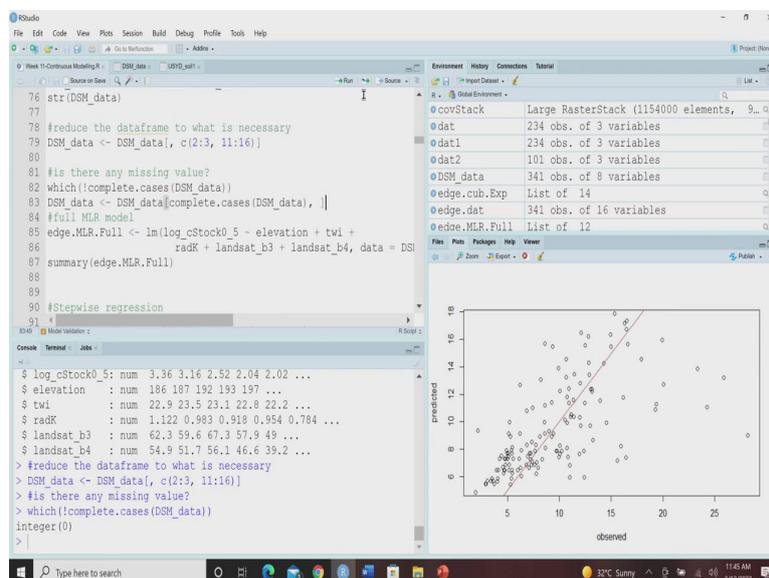
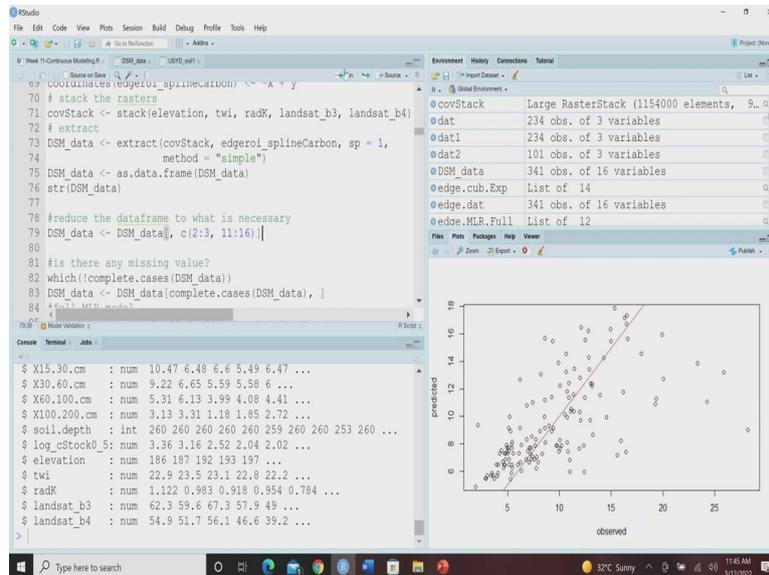
The screenshot displays the RStudio interface. The top-left pane shows a data frame with columns: id, x, y, X1.5m, X1.5cm, X3.0m, X3.0cm, X6.0m, X6.0cm, X10.0m, X10.0cm, vol_dry, log_stack0_5, and elevat. The top-right pane shows the Environment window with a list of objects including @covStack, @dat, @dat1, @dat2, @DSM_data, @edge_cub.Exp, @edge.dat, and @edge_MLR.Phil. The bottom-left pane shows the R console with the following code:

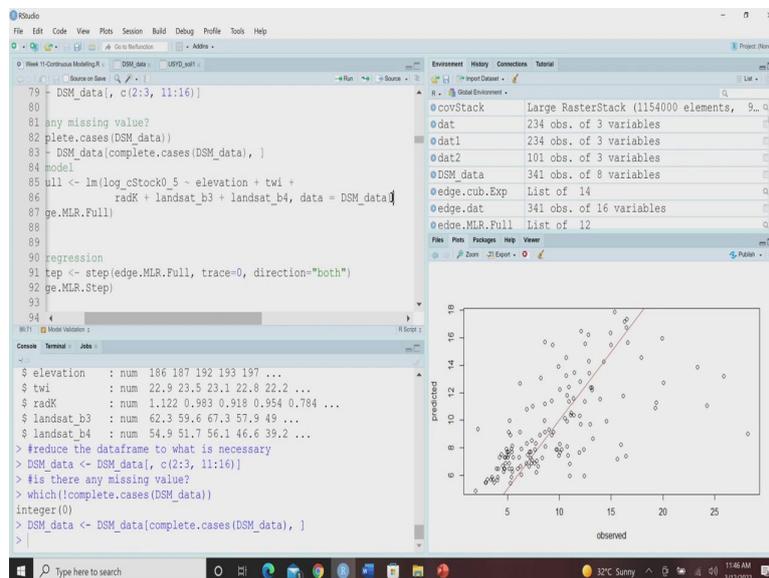
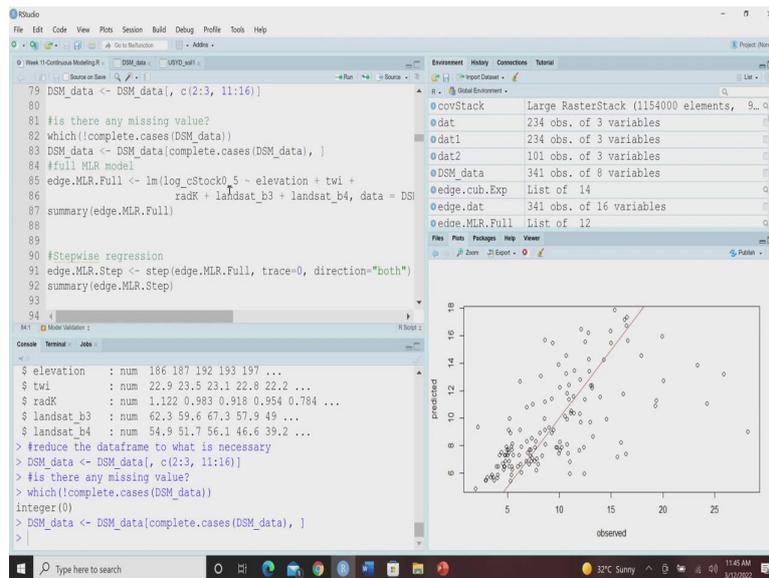
```
> # point data
> data(edgeroi_splineCarbon)
> names(edgeroi_splineCarbon)[2:3] <- c("x", "y")
> edgeroi_splineCarbon$log_stack0_5 <- log(edgeroi_splineCarbon
+ $X0.5cm)
+
> # grids
> data(edgeroiCovariates)
> coordinates(edgeroi_splineCarbon) <- ~x + y
> covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
> View(DSM_data)
>
```

The bottom-right pane shows a scatter plot with 'observed' on the x-axis and 'predicted' on the y-axis. Both axes range from 0 to 18. A diagonal line representing y=x is drawn, and numerous data points are scattered around it, showing a positive correlation between observed and predicted values.

So, this DSM underscore data. So, you can see 341 observation with 16 variables where we have all the value all the variables as well as the all the covariates. So, once our data is ready, now, next step is to see the structure of the data. So, you can see this is the structure of the data, we have these 341 observations of 16 variables.

(Refer Slide Time: 20:37)



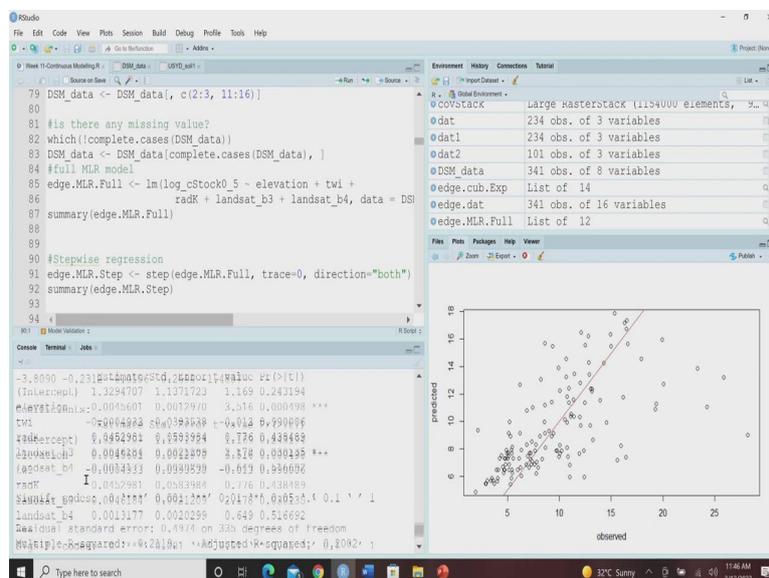
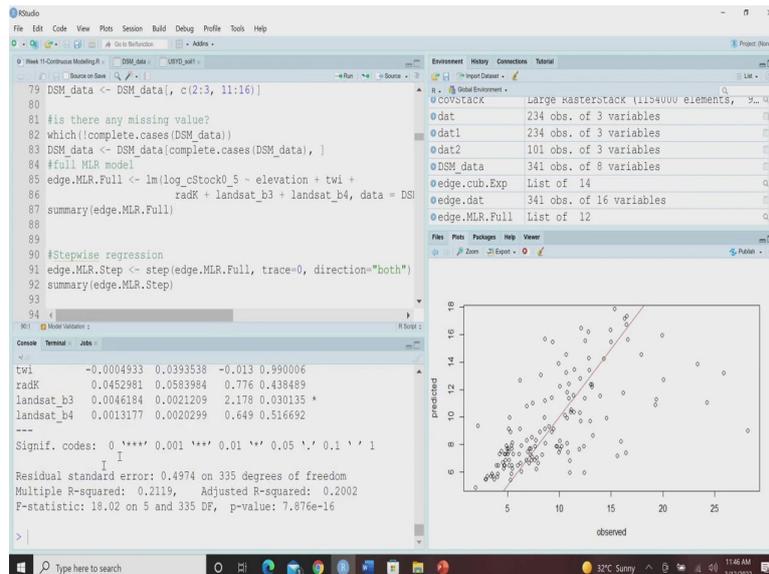


Now, suppose we want to deal with only logarithmic... log converted 0 to 5-centimetre data and only with the covariates. So, we are going to specify the columns, which we are going to keep for further calculations. So, we are going to use another data set called DSM underscore data. So, that keeps all the observation but keeping only the location and context that is x and y and from 11 to 16.

So, you know 11 To 16 stands for these five variable covariates as well as the log converted carbon stock. So, let us run this. So, this reduced data set has been already developed, then we want to keep only the complete cases. So, we want to inquire R that whether there is any missing values or not. So, when we use the exclamatory sign that means it opposite. So, we want to see whether there any missing values in the DSM data or not. So, R returns integer 0 that means there is no missing value.

So, anyway, we want to keep all these complete values complete data set in the DSM data and then we are going to keep these DSM complete cases of the DSM data for subsequent modelling. Now, in the full model you can see we are going to use this linear model again just like the simple linear regression. So, here our target is logarithmic converted stock, 0 to 5 centimetre and our predictors are elevation twi, radk, landset b3, landset b4 and our data is the DSM data, let us run it, let us see how it looks like.

(Refer Slide Time: 22:25)



```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
DSM_data | DSM_data | USTC_soil |
79 DSM_data <- DSM_data[, c(2:3, 11:16)]
80
81 #is there any missing value?
82 which(!complete.cases(DSM_data))
83 DSM_data <- DSM_data[complete.cases(DSM_data), ]
84 #full MLR model
85 edge.MLR.Full <- lm(log_cStock0_5 ~ elevation + twi +
86 radK + landsat_b3 + landsat_b4, data = DSM_data)
87 summary(edge.MLR.Full)
88
89
90 #Stepwise regression
91 edge.MLR.Step <- step(edge.MLR.Full, trace=0, direction="both")
92 summary(edge.MLR.Step)
93
94
Residuals:
  Min       1Q   Median       3Q      Max
-3.8090 -0.2312 -0.0156  0.2590  1.4837

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.3294707  1.1371723   1.169  0.243194
elevation    0.0045601  0.0012970   3.516  0.000498 ***
twi          -0.0004933  0.0393538  -0.013  0.990006
radK         0.0452981  0.0583984   0.776  0.438489
landsat_b3   0.0046184  0.0021209   2.178  0.030135 *

```

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
DSM_data | DSM_data | USTC_soil |
79 DSM_data <- DSM_data[, c(2:3, 11:16)]
80
81 #is there any missing value?
82 which(!complete.cases(DSM_data))
83 DSM_data <- DSM_data[complete.cases(DSM_data), ]
84 #full MLR model
85 edge.MLR.Full <- lm(log_cStock0_5 ~ elevation + twi +
86 radK + landsat_b3 + landsat_b4, data = DSM_data)
87 summary(edge.MLR.Full)
88
89
90 #Stepwise regression
91 edge.MLR.Step <- step(edge.MLR.Full, trace=0, direction="both")
92 summary(edge.MLR.Step)
93
94
Call:
lm(formula = log_cStock0_5 ~ elevation + twi + radK + landsat_b3 +
landsat_b4, data = DSM_data)

Residuals:
  Min       1Q   Median       3Q      Max
-3.8090 -0.2312 -0.0156  0.2590  1.4837

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.3294707  1.1371723   1.169  0.243194
elevation    0.0045601  0.0012970   3.516  0.000498 ***
twi          -0.0004933  0.0393538  -0.013  0.990006
radK         0.0452981  0.0583984   0.776  0.438489
landsat_b3   0.0046184  0.0021209   2.178  0.030135 *
landsat_b4   0.0013177  0.0020299   0.649  0.516692

```

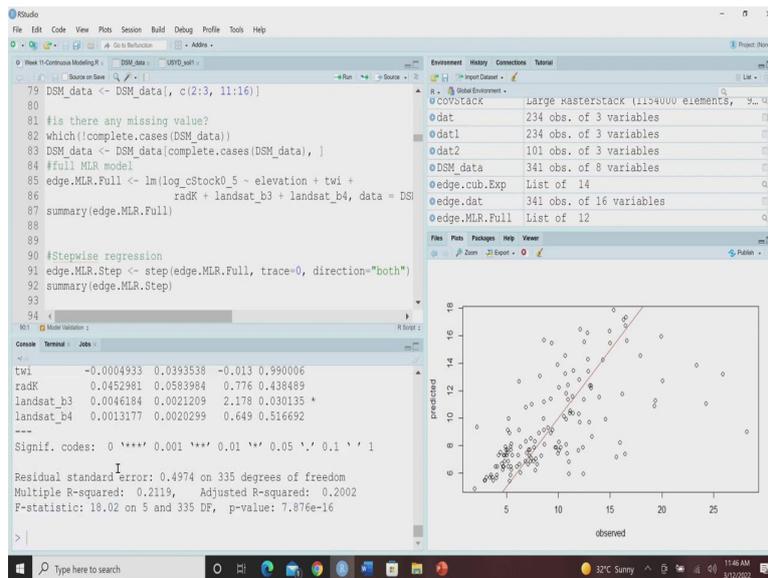
```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
DSM_data | DSM_data | USTC_soil |
79 DSM_data <- DSM_data[, c(2:3, 11:16)]
80
81 #is there any missing value?
82 which(!complete.cases(DSM_data))
83 DSM_data <- DSM_data[complete.cases(DSM_data), ]
84 #full MLR model
85 edge.MLR.Full <- lm(log_cStock0_5 ~ elevation + twi +
86 radK + landsat_b3 + landsat_b4, data = DSM_data)
87 summary(edge.MLR.Full)
88
89
90 #Stepwise regression
91 edge.MLR.Step <- step(edge.MLR.Full, trace=0, direction="both")
92 summary(edge.MLR.Step)
93
94
Residuals:
  Min       1Q   Median       3Q      Max
-3.8090 -0.2312 -0.0156  0.2590  1.4837

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.3294707  1.1371723   1.169  0.243194
elevation    0.0045601  0.0012970   3.516  0.000498 ***
twi          -0.0004933  0.0393538  -0.013  0.990006
radK         0.0452981  0.0583984   0.776  0.438489
landsat_b3   0.0046184  0.0021209   2.178  0.030135 *
landsat_b4   0.0013177  0.0020299   0.649  0.516692

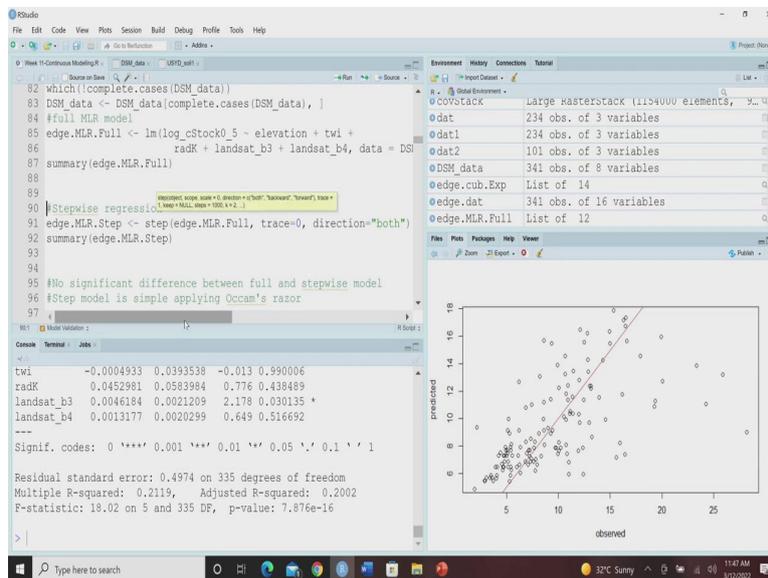
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

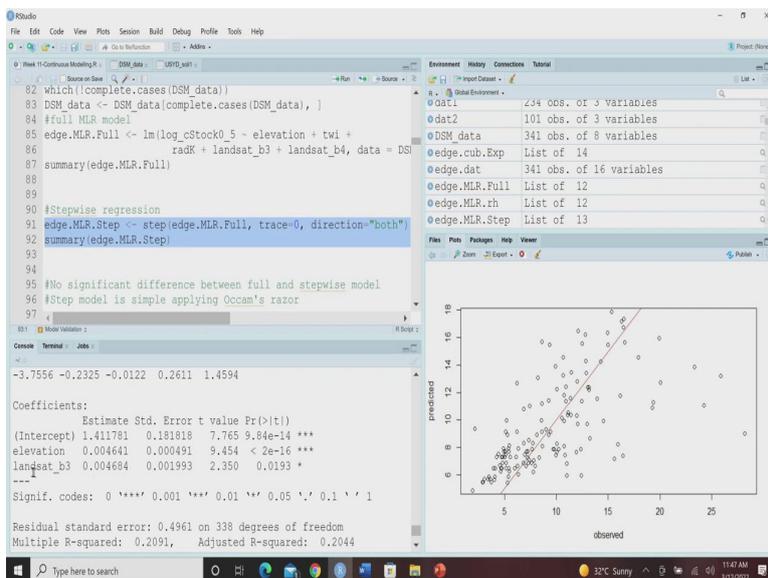
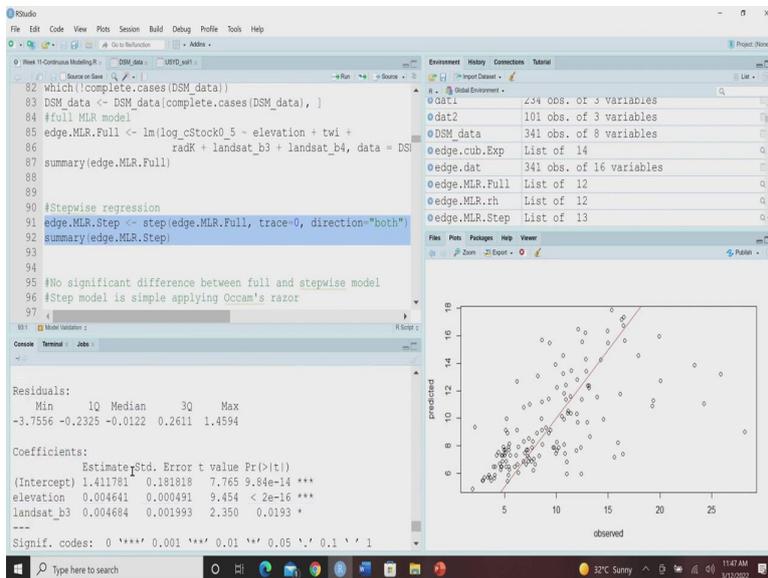
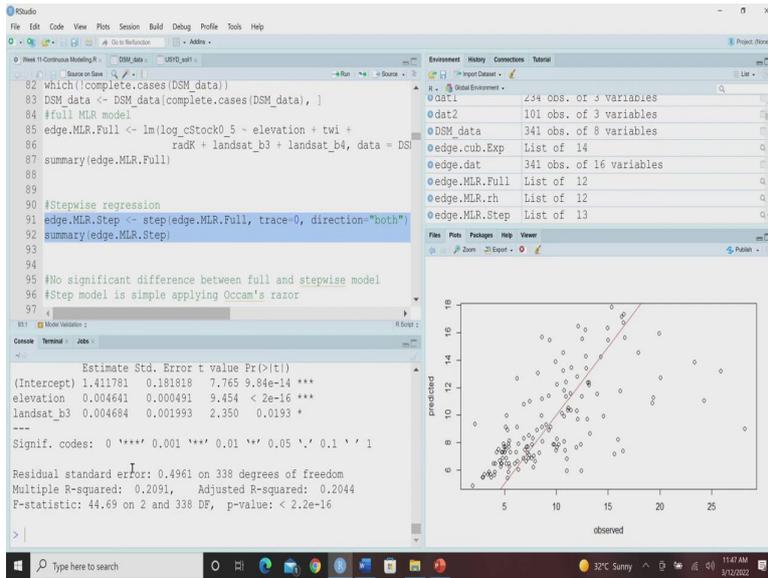
```

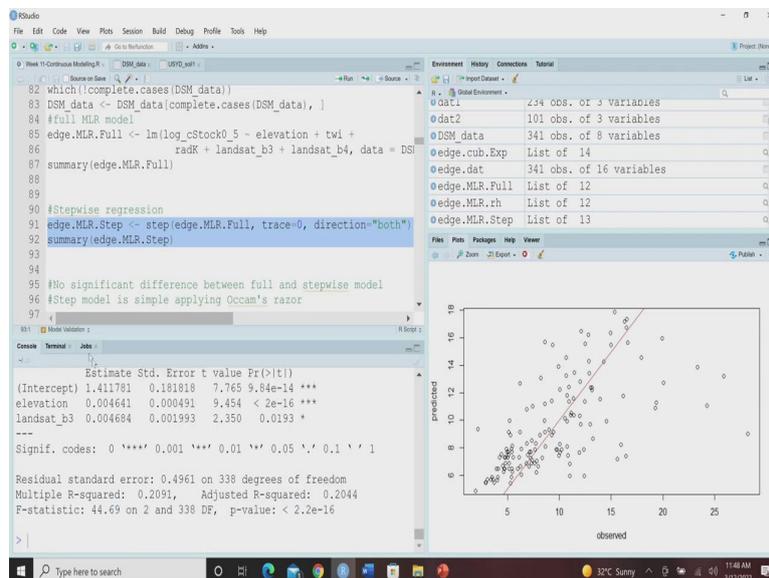
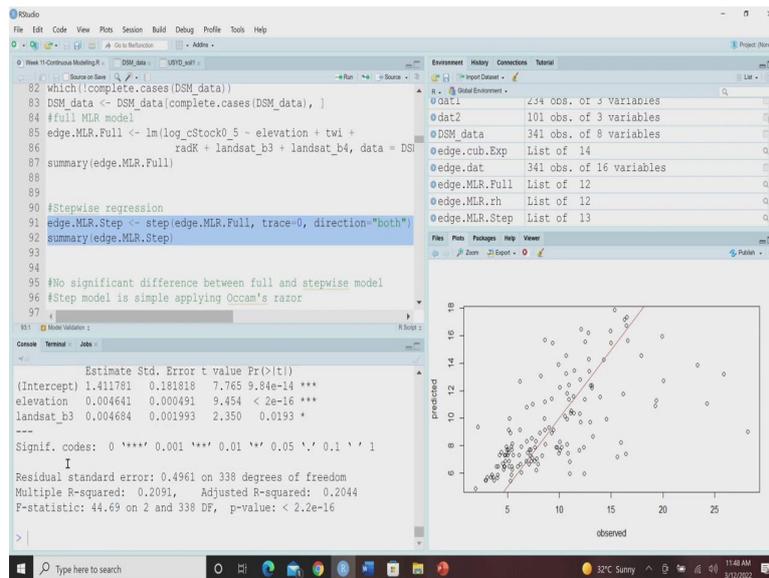


Now, the summary of the full model, you can see this is the summary of the full model, where you can have minimum value maximum value, then first quartile median and third quartile, you can have the coefficient of intercepts and other variables also their standard error their t values and you can have the multiple R square and adjusted R square and their P values also. So, you can get all the information using this.

(Refer Slide Time: 22:50)







Now, here in the multiple linear regression model, we have used all the model parameters, but there is another regression method that is called stepwise regression method. So, in the stepwise regression method, it moves step by step and tries to see whether addition or removal of any variable can significantly affect the model outcome or not.

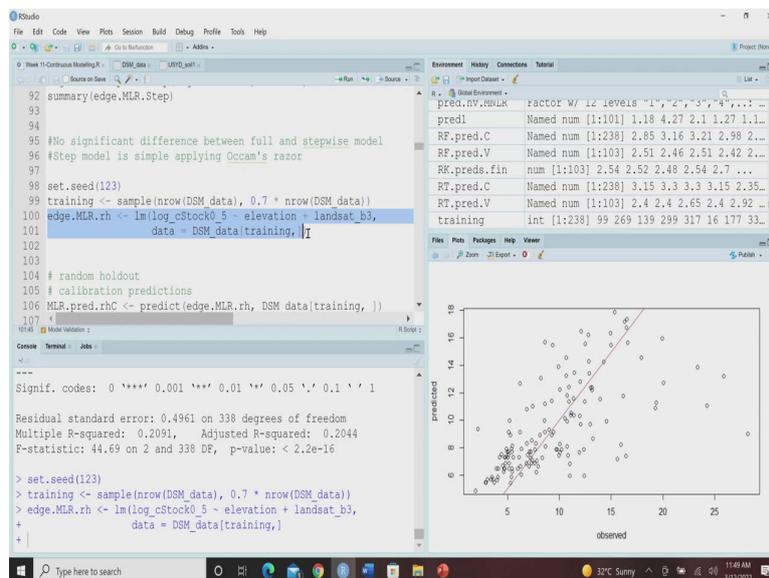
And based on that, it can select the most parsimonious model or simplified model than that of the full multi multiple linear model which keep all the variables. So, here we are going to use the step function and then we are going to use these edge dot MLR dot full models. So, here we have already used these edge dot MLR dot full model and then we want to use it in the both direction.

So, the stepwise regression, let us keep this stepwise regression. And then we will see that the stepwise regression will give you this output from this output we can see except you know,

instead of all the covariates the stepwise regression kept only the two covariates like elevation and landsat b3 which are giving the significant you know impact which has significant impact on the organic carbon.

So, the stepwise regression will simplify the model in earlier, in the full model we have kept all the five variables, but in case of stepwise regression, we are getting less number of variables by this individual step by step basis they have selected these two variables. So, you can see here the results from this to... from the full model and the stepwise model. In case of stepwise model, we are getting the adjusted R squared values 0.20 however, in the full model we are getting the adjusted outskirts R squared is again same 0.20. So, we cannot see any difference between the full model In the stepwise model, but step model is more simpler than full model.

(Refer Slide Time: 25:07)



RStudio

```

96 #Step model is simple applying Occam's razor
97
98 set.seed(123)
99 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
100 edge.MLR.rh <- lm(log_cStock0_5 ~ elevation + landsat_b3,
101                 data = DSM_data[training,])
102
103
104 # random holdout
105 # calibration predictions
106 MLR.pred.rhC <- predict(edge.MLR.rh, DSM_data[training,])
107 # validation predictions
108 MLR.pred.rhV <- predict(edge.MLR.rh, DSM_data[-training,])
109
110 # calibration
111

```

Environment

- edge.MLR.rh List of 12
- edge.MLR.Step List of 13
- edge.RF.Exp Large randomForest.formula (18 elements...
- edge.RT.Exp List of 14
- edgeroi_splim Formal class SpatialPointsDataFrame
- elevation Large RasterLayer (230800 elements, 1...
- fit1 List of 13
- fit2 List of 13

RStudio

```

96 #Step model is simple applying Occam's razor
97
98 set.seed(123)
99 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
100 edge.MLR.rh <- lm(log_cStock0_5 ~ elevation + landsat_b3,
101                 data = DSM_data[training,])
102
103
104 # random holdout
105 # calibration predictions
106 MLR.pred.rhC <- predict(edge.MLR.rh, DSM_data[training,])
107 # validation predictions
108 MLR.pred.rhV <- predict(edge.MLR.rh, DSM_data[-training,])
109
110 # calibration
111

```

Environment

- edge.MLR.rh List of 12
- edge.MLR.Step List of 13
- edge.RF.Exp Large randomForest.formula (18 elements...
- edge.RT.Exp List of 14
- edgeroi_splim Formal class SpatialPointsDataFrame
- elevation Large RasterLayer (230800 elements, 1...
- fit1 List of 13
- fit2 List of 13

RStudio

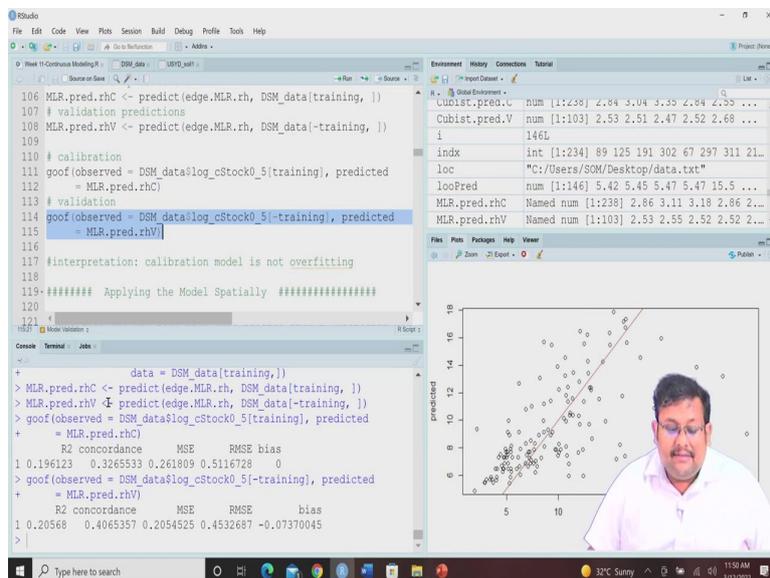
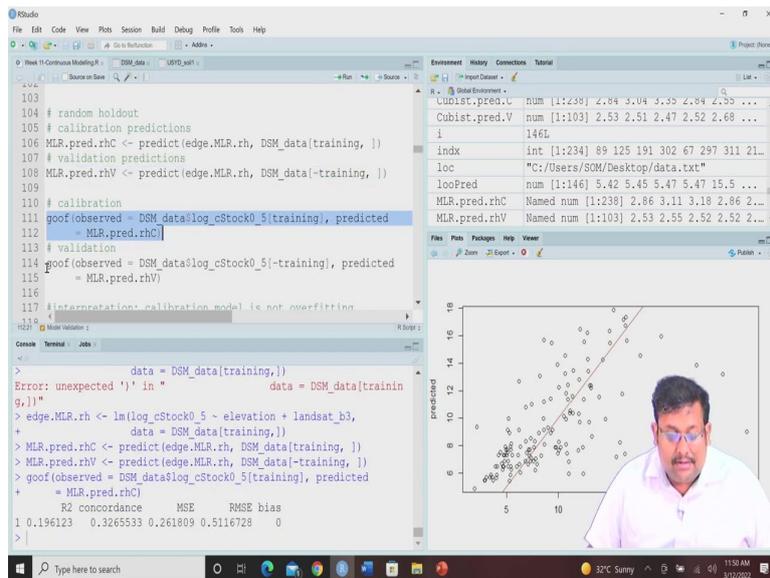
```

96 #Step model is simple applying Occam's razor
97
98 set.seed(123)
99 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
100 edge.MLR.rh <- lm(log_cStock0_5 ~ elevation + landsat_b3,
101                 data = DSM_data[training,])
102
103
104 # random holdout
105 # calibration predictions
106 MLR.pred.rhC <- predict(edge.MLR.rh, DSM_data[training,])
107 # validation predictions
108 MLR.pred.rhV <- predict(edge.MLR.rh, DSM_data[-training,])
109
110 # calibration
111

```

Environment

- coef1 Named num [1:14] -13.0262 1.2293 0.0124 ...
- Cubist.pred.C num [1:238] 2.84 3.04 3.35 2.84 2.55 ...
- Cubist.pred.V num [1:103] 2.53 2.51 2.47 2.52 2.68 ...
- i 146L
- indx int [1:234] 89 125 191 302 67 297 311 21...
- loc "C:/Users/SOM/Desktop/data.txt"
- looPred num [1:146] 5.42 5.45 5.47 5.47 15.5 ...
- MLR.pred.rhC Named num [1:238] 2.86 3.11 3.18 2.86 2...



So, according to the Occam's Razor principle, we already we will select the step model then the full model. So, next step is to map based on, next step is to develop the separate training samples by selecting 70 percent of the data, so again we are setting the seed 123, then we are selecting the training samples that is 0 you know 70 percent of the data and then we are fitting this you know model now, here you can see instead of all the five variables, we are keeping only the two variables elevation and landset b3 based on our stepwise model you know output we are keeping only these two and just a minute sorry guys.

So, then we can also do the random holdout. So, based on the, this calibration data set, we can develop this you know, we can we can predict based on this calibration data set, but at the same time we can predict based on the validation data set. So, validation data set is you can see here this is minus training. So, once you do that, then you can do the goodness of his calculation based on both calibration and also based on variation also separately.

So, you can you can you can compare their results and you can see from this result is the calibration R squared 0.19 or almost 0.20 validation R squared is also 0.20 RMSE 0.51 here 0.45 so, we can see there is no the difference, a significant difference between the calibration model statistics and validation model statistics. So, we can infer that there is no you know, there is no overfitting because in case of overfitting this calibration model performs over optimistically than the validation model.

(Refer Slide Time: 27:18)

```

119 ##### Applying the Model Spatially #####
120
121 #What does the map look like that results from a particular mod
122 #Applying the model parameters to the grids of the covariates t
123
124
125 #Covariate table
126 data(edgeroiCovariates)
127 covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
128 tempD <- data.frame(cellNos = seq(1:ncell(covStack)))
129 vals <- as.data.frame(getValues(covStack))
130 tempD <- cbind(tempD, vals)
131 tempD <- tempD[complete.cases(tempD), ]
132 cellNos <- c(tempD$cellNos)
133 gXY <- data.frame(xyFromCell(covStack, cellNos, spatial = FALSE)
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
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
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

```

```

122 #Applying the model parameters to the grids of the covariates t
123
124
125 #Covariate table
126 data(edgeroiCovariates)
127 covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
128 tempD <- data.frame(cellNos = seq(1:ncell(covStack)))
129 vals <- as.data.frame(getValues(covStack))
130 tempD <- cbind(tempD, vals)
131 tempD <- tempD[complete.cases(tempD), ]
132 cellNos <- c(tempD$cellNos)
133 gXY <- data.frame(xyFromCell(covStack, cellNos, spatial = FALSE)
134 tempD <- cbind(gXY, tempD)
135 str(tempD)
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
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
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

```

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
0 -> R -> Global Environment -> Import Dataset -> List...
128 tempD <- data.frame(cellNos = seq(1:nrow(covStack)))
129 vals <- as.data.frame(getValues(covStack))
130 tempD <- cbind(tempD, vals)
131 tempD <- tempD[complete.cases(tempD), ]
132 cellNos <- c(tempD$cellNos)
133 gXY <- data.frame(xyFromCell(covStack, cellNos, spatial = FALSE))
134 tempD <- cbind(gXY, tempD)
135 str(tempD)
136
137
138 map_MLR <- predict(edge_MLR.rh, newdata = tempD)
139 map_MLR <- cbind(data.frame(tempD[, c("x", "y")]), map_MLR)
140
141 #rasterize the table
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
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
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

```

Environment History Connections Tutorial

- Global Environment
- Import Dataset
- List...
- Named num [1:14] = 13.0262 1.2253 0.0124 ...
- Cubist.pred.C num [1:238] 2.84 3.04 3.35 2.84 2.55 ...
- Cubist.pred.V num [1:103] 2.53 2.51 2.47 2.52 2.68 ...
- i 146L
- indx int [1:234] 89 125 191 302 67 297 311 21...
- loc "C:/Users/SOM/Desktop/data.txt"
- looPred num [1:146] 5.42 5.45 5.47 5.47 15.5 ...
- map_MLR Large numeric (201313 elements, 14.5 MB)

Files Plots Packages Help Viewer

32°C Sunny 11:52 AM 3/12/2022

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
0 -> R -> Global Environment -> Import Dataset -> List...
131 tempU <- tempU[complete.cases(tempU), ]
132 cellNos <- c(tempU$cellNos)
133 gXY <- data.frame(xyFromCell(covStack, cellNos, spatial = FALSE))
134 tempD <- cbind(gXY, tempD)
135 str(tempD)
136
137
138 map_MLR <- predict(edge_MLR.rh, newdata = tempD)
139 map_MLR <- cbind(data.frame(tempD[, c("x", "y")]), map_MLR)
140
141 #rasterize the table
142
143 map_MLR.r <- rasterFromXYZ(as.data.frame(map_MLR[, 1:3]))
144 plot(map_MLR.r, main = "MLR predicted log SOC stock (0-5cm)")
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
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
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

```

Environment History Connections Tutorial

- Global Environment
- Import Dataset
- List...
- map_MLR 201313 obs. of 3 variables
- map_MLR.r Large RasterLayer (230800 elements, 1...)
- map_MLR.r.low Formal class RasterLayer
- map_MLR.r.pred Formal class RasterLayer
- map_MLR.r.up Formal class RasterLayer
- map_MLR.r.l Formal class RasterLayer
- map_MNLR.c Formal class RasterLayer
- map_MNLR.o Formal class RasterLayer

Files Plots Packages Help Viewer

32°C Sunny 11:52 AM 3/12/2022

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
0 -> R -> Global Environment -> Import Dataset -> List...
135 str(tempD)
136
137
138 map_MLR <- predict(edge_MLR.rh, newdata = tempD)
139 map_MLR <- cbind(data.frame(tempD[, c("x", "y")]), map_MLR)
140
141 #rasterize the table
142
143 map_MLR.r <- rasterFromXYZ(as.data.frame(map_MLR[, 1:3]))
144 plot(map_MLR.r, main = "MLR predicted log SOC stock (0-5cm)")
145
146
147 # set the projection (set the CRS to WGS84 Zone 55)
148 crs(map_MLR.r) <- "+proj=utm +zone=55 +south +ellps=WGS84
149 +datum=WGS84 +units=m +no_defs"
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
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
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

```

Environment History Connections Tutorial

- Global Environment
- Import Dataset
- List...
- map_MLR 201313 obs. of 3 variables
- map_MLR.r Large RasterLayer (230800 elements, 1...)
- map_MLR.r.low Formal class RasterLayer
- map_MLR.r.pred Formal class RasterLayer
- map_MLR.r.up Formal class RasterLayer
- map_MLR.r.l Formal class RasterLayer
- map_MNLR.c Formal class RasterLayer
- map_MNLR.o Formal class RasterLayer

Files Plots Packages Help Viewer

32°C Sunny 11:53 AM 3/12/2022

Now, if we want to if we want to apply this model for producing the map specially, then again, we have to run the script again for downloading the data using stacking of the data and then creating the data frame, selecting the cells for which we had the values and extracting those values and extracting their location. And so, once we have it this you know this is the temporary file which we have created with the, their location.

So, g x y is basically you know the location x and y and the temporary data file you can see here the cell numbers for which we have this location and their elevation and twi, radk, landset b3, landset b4 values and then we want to map based on this MLR model and then so, then columbine, this dataset.

And then finally, we are going to raster from the x, y, z from this data set and then we are going to plot so, this is the MLR predicted logarithmic SOC stock for 0 to 5 centimetre. So, this is how guys you can produce the map using the full model or stepwise model using this and also you can check the model performance based on the you know the model performance by either simple holdout or random holdout and also by leave one out cross validation.

(Refer Slide Time: 29:18)

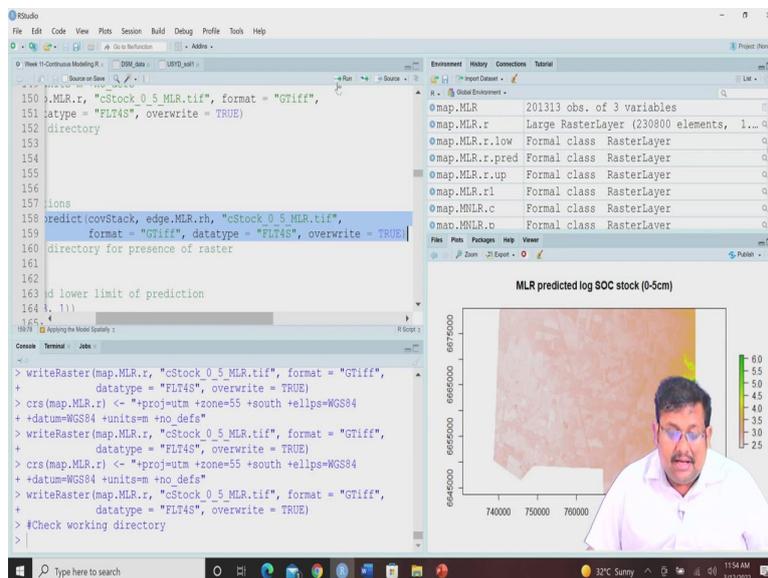
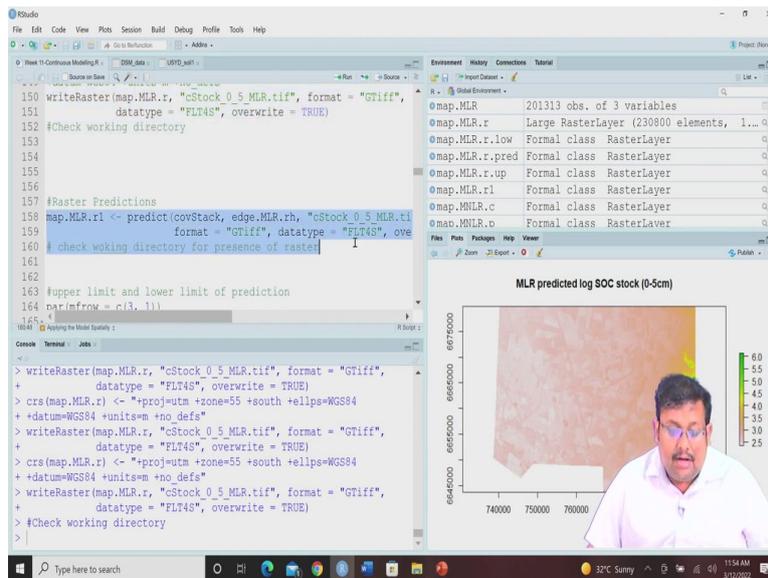
The screenshot displays the RStudio interface. The main editor window contains the following R code:

```
141 #rasterize the table
142
143 map.MLR.r <- rasterFromXYZ(as.data.frame(map.MLR[, 1:3]))
144 plot(map.MLR.r, main = "MLR predicted log SOC stock (0-5cm)")
145
146
147 # set the projection (set the CRS to WGS84 Zone 55)
148 crs(map.MLR.r) <- "+proj=utm +zone=55 +south +ellps=WGS84
149 +datum=WGS84 +units=m +no_defs"
150 writeRaster(map.MLR.r, "cStock_0_5_MLR.tif", format = "GTiff",
151            datatype = "FLT4S", overwrite = TRUE)
152 #check working directory
153
154
155
156
```

The Environment pane on the right shows the following objects:

- map.MLR: 201313 obs. of 3 variables
- map.MLR.r: Large RasterLayer (230800 elements, 1...
- map.MLR.r.low: Formal class RasterLayer
- map.MLR.r.pred: Formal class RasterLayer
- map.MLR.r.up: Formal class RasterLayer
- map.MLR.rl: Formal class RasterLayer
- map.MNLR.c: Formal class RasterLayer
- map.MNLR.o: Formal class RasterLayer

The console shows the execution of the code, including the rasterization and plotting steps. The plot window displays a map titled "MLR predicted log SOC stock (0-5cm)" with a color scale ranging from 25 to 60. The map shows a geographical area with varying values, and a person's face is visible in the bottom right corner of the plot area.



Now, once we have produced this project, you know produce this map next, we are going to set the projections, coordinate reference system as well as you know they are so here you can see we are setting the coordinate reference system as UTM zone 55 south with the WGS 84. And we can we can create you know we can we can write the raster also which will be saved in our working directory.

So, this is how we do this. And so, we can we can also do the rest of predictions So, we can produce this and you can, you can see that raster file will be created in your working folder and then the upper limit and the lower limit of the prediction. So, if we want to see the, what is the upper limit of your prediction, what is the lower limit of the prediction, you can also do that by running the script and you can see in the script, they are all basically the same, but we are changing the index argument from 2, 1 and 3.

(Refer Slide Time: 30:38)

This screenshot shows the RStudio interface with the following code in the editor:

```
163 #upper limit and lower limit of prediction
164 par(mfrow = c(3, 1))
165 predfun <- function(model, data) {
166   v <- predict(model, data, interval = "prediction", level = 0.9)
167 }
168 map.MLR.r.low <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_low.tif",
169   fun = predfun, index = 2, format = "GTiff", datatype = "FLT4S",
170   main = "MLR predicted log SOC stock (0-5cm) lower limit")
171 map.MLR.r.pred <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_pred.tif",
172   fun = predfun, index = 1, format = "GTiff", datatype = "FLT4S",
173   main = "MLR predicted log SOC stock (0-5cm)")
174 map.MLR.r.up <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_up.tif",
175   fun = predfun, index = 3, format = "GTiff", datatype = "FLT4S",
176   main = "MLR predicted log SOC stock (0-5cm) upper limit")
177
```

The console shows the execution of the following commands:

```
> + datum=WGS84 + units=m + no_defs
> writeRaster(map.MLR.r, "cStock_0_5_MLR.tif", format = "GTiff",
+   datatype = "FLT4S", overwrite = TRUE)
> crs(map.MLR.r) <- "+proj=utm +zone=55 +south +ellps=WGS84
+ datum=WGS84 +units=m +no_defs"
> writeRaster(map.MLR.r, "cStock_0_5_MLR.tif", format = "GTiff",
+   datatype = "FLT4S", overwrite = TRUE)
> #Check working directory
> map.MLR.r1 <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR.tif",
+   format = "GTiff", datatype = "FLT4S", overwrite = TRUE)
>
```

The Environment pane lists several objects: @map.MLR (201313 obs. of 3 variables), @map.MLR.r (Large RasterLayer), @map.MLR.r.low (Formal class RasterLayer), @map.MLR.r.pred (Formal class RasterLayer), @map.MLR.r.up (Formal class RasterLayer), @map.MLR.r1 (Formal class RasterLayer), @map.MNLR.c (Formal class RasterLayer), and @map.MNLR.o (Formal class RasterLayer).

The plot shows a map titled "MLR predicted log SOC stock (0-5cm)" with a color scale from 25 to 60. The x-axis ranges from 740000 to 790000 and the y-axis from 6645000 to 6675000.

This screenshot shows the RStudio interface with the following code in the editor:

```
163 lower limit of prediction
164 1)
165 predict(model, data) {
166   model, data, interval = "prediction", level = 0.9)
167 }
168 predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_low.tif",
169   fun = predfun, index = 2, format = "GTiff", datatype = "FLT4S",
170   main = "MLR predicted log SOC stock (0-5cm) lower limit")
171 predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_pred.tif",
172   fun = predfun, index = 1, format = "GTiff", datatype = "FLT4S",
173   main = "MLR predicted log SOC stock (0-5cm)")
174 predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_up.tif",
175   fun = predfun, index = 3, format = "GTiff", datatype = "FLT4S",
176   main = "MLR predicted log SOC stock (0-5cm) upper limit")
177
```

The console shows the execution of the following commands:

```
> + datum=WGS84 + units=m + no_defs
> writeRaster(map.MLR.r, "cStock_0_5_MLR.tif", format = "GTiff",
+   datatype = "FLT4S", overwrite = TRUE)
> crs(map.MLR.r) <- "+proj=utm +zone=55 +south +ellps=WGS84
+ datum=WGS84 +units=m +no_defs"
> writeRaster(map.MLR.r, "cStock_0_5_MLR.tif", format = "GTiff",
+   datatype = "FLT4S", overwrite = TRUE)
> #Check working directory
> map.MLR.r1 <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR.tif",
+   format = "GTiff", datatype = "FLT4S", overwrite = TRUE)
>
```

The Environment pane lists several objects: @map.MLR (201313 obs. of 3 variables), @map.MLR.r (Large RasterLayer), @map.MLR.r.low (Formal class RasterLayer), @map.MLR.r.pred (Formal class RasterLayer), @map.MLR.r.up (Formal class RasterLayer), @map.MLR.r1 (Formal class RasterLayer), @map.MNLR.c (Formal class RasterLayer), and @map.MNLR.o (Formal class RasterLayer).

The plot shows a map titled "MLR predicted log SOC stock (0-5cm)" with a color scale from 25 to 60. The x-axis ranges from 740000 to 790000 and the y-axis from 6645000 to 6675000.

This screenshot shows the RStudio interface with the following code in the editor:

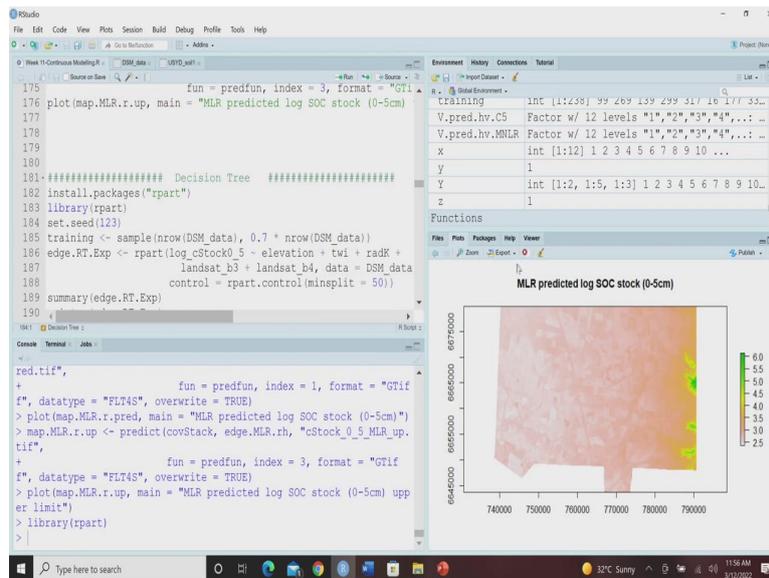
```
163 #lower limit of prediction
164 (3, 1))
165 predict(model, data) {
166   model, data, interval = "prediction", level = 0.9)
167 }
168 map.MLR.r.low <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_low.tif",
169   fun = predfun, index = 2, format = "GTiff", datatype = "FLT4S",
170   main = "MLR predicted log SOC stock (0-5cm) lower limit")
171 map.MLR.r.pred <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_pred.tif",
172   fun = predfun, index = 1, format = "GTiff", datatype = "FLT4S",
173   main = "MLR predicted log SOC stock (0-5cm)")
174 map.MLR.r.up <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_up.tif",
175   fun = predfun, index = 3, format = "GTiff", datatype = "FLT4S",
176   main = "MLR predicted log SOC stock (0-5cm) upper limit")
177
```

The console shows the execution of the following commands:

```
> + datum=WGS84 + units=m + no_defs
> writeRaster(map.MLR.r, "cStock_0_5_MLR.tif", format = "GTiff",
+   datatype = "FLT4S", overwrite = TRUE)
> crs(map.MLR.r) <- "+proj=utm +zone=55 +south +ellps=WGS84
+ datum=WGS84 +units=m +no_defs"
> writeRaster(map.MLR.r, "cStock_0_5_MLR.tif", format = "GTiff",
+   datatype = "FLT4S", overwrite = TRUE)
> #Check working directory
> map.MLR.r1 <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR.tif",
+   format = "GTiff", datatype = "FLT4S", overwrite = TRUE)
>
```

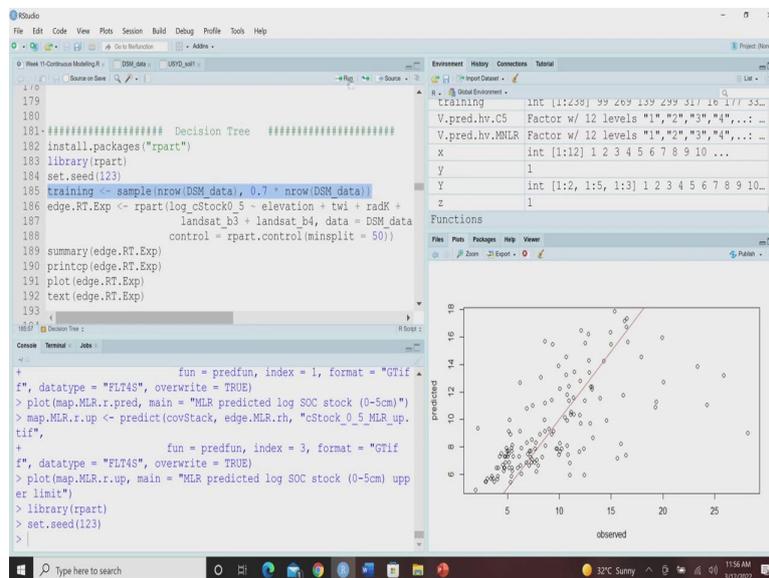
The Environment pane lists several objects: @map.MLR (201313 obs. of 3 variables), @map.MLR.r (Large RasterLayer), @map.MLR.r.low (Formal class RasterLayer), @map.MLR.r.pred (Formal class RasterLayer), @map.MLR.r.up (Formal class RasterLayer), @map.MLR.r1 (Formal class RasterLayer), @map.MNLR.c (Formal class RasterLayer), and @map.MNLR.o (Formal class RasterLayer).

The plot shows a map titled "MLR predicted log SOC stock (0-5cm)" with a color scale from 25 to 60. The x-axis ranges from 740000 to 790000 and the y-axis from 6645000 to 6675000.



So, 2 1 and 3 one will give you the you know, the one will give you the predicted values and when the 2 and 3 will give you the lower limit and the upper limit of the prediction. So, let us run this script and you can see that 3 maps will be produced with the lower limit and original prediction and the upper limit. So, this is how you can produce the map there is for using the multiple linear regression. And for decision trees you know, you need to install this package called rpart and I have already installed so, I am not going to install it further.

(Refer Slide Time: 31:28)



```

179
180
181 ##### Decision Tree #####
182 library("rpart")
183 y(rpart)
184 ed(123)
185 ng <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
186 T.Exp <- rpart(log_cStock0_5 ~ elevation + twi + radK +
187               landsat_b3 + landsat_b4, data = DSM_data[training, ],
188               control = rpart.control(minsplit = 50))
189 y(edge.RT.Exp)
190 p(edge.RT.Exp)
191 gge.RT.Exp)
192 gge.RT.Exp)
193
194
195 f", datatype = "FL74S", overwrite = TRUE)
196 > plot(map.MLR.r.pred, main = "MLR predicted log SOC stock (0-5cm)")
197 > map.MLR.r.up <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_up.
198 tif",
199               fun = predfun, index = 3, format = "Gtif
200 f", datatype = "FL74S", overwrite = TRUE)
201 > plot(map.MLR.r.up, main = "MLR predicted log SOC stock (0-5cm) up
202 per limit")
203 > library(rpart)
204 > set.seed(123)
205 > training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
206 >

```

Environment

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",...
x	int [1:12]	1 2 3 4 5 6 7 8 9 10 ...
y		1
Y	int [1:2, 1:5, 1:3]	1 2 3 4 5 6 7 8 9 10...
z		1

```

179
180
181 ##### Decision Tree #####
182 library("rpart")
183 y(rpart)
184 ed(123)
185 le(nrow(DSM_data), 0.7 * nrow(DSM_data))
186 part(log_cStock0_5 ~ elevation + twi + radK +
187       landsat_b3 + landsat_b4, data = DSM_data[training, ],
188       control = rpart.control(minsplit = 50))
189 Exp)
190 Exp)
191 )
192 )
193
194
195 f", datatype = "FL74S", overwrite = TRUE)
196 > plot(map.MLR.r.pred, main = "MLR predicted log SOC stock (0-5cm)")
197 > map.MLR.r.up <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_up.
198 tif",
199               fun = predfun, index = 3, format = "Gtif
200 f", datatype = "FL74S", overwrite = TRUE)
201 > plot(map.MLR.r.up, main = "MLR predicted log SOC stock (0-5cm) up
202 per limit")
203 > library(rpart)
204 > set.seed(123)
205 > training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
206 >

```

Environment

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",...
x	int [1:12]	1 2 3 4 5 6 7 8 9 10 ...
y		1
Y	int [1:2, 1:5, 1:3]	1 2 3 4 5 6 7 8 9 10...
z		1

```

179
180
181 ##### Decision Tree #####
182 install.packages("rpart")
183 library(rpart)
184 set.seed(123)
185 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
186 edge.RT.Exp <- rpart(log_cStock0_5 ~ elevation + twi + radK +
187                     landsat_b3
188                     control = rpa
189 summary(edge.RT.Exp)
190 printcp(edge.RT.Exp)
191 plot(edge.RT.Exp)
192 text(edge.RT.Exp)
193
194
195 f", datatype = "FL74S", overwrite = TRUE)
196 > plot(map.MLR.r.pred, main = "MLR predicted log SOC stock (0-5cm)")
197 > map.MLR.r.up <- predict(covStack, edge.MLR.rh, "cStock_0_5_MLR_up.
198 tif",
199               fun = predfun, index = 3, format = "Gtif
200 f", datatype = "FL74S", overwrite = TRUE)
201 > plot(map.MLR.r.up, main = "MLR predicted log SOC stock (0-5cm) up
202 per limit")
203 > library(rpart)
204 > set.seed(123)
205 > training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
206 > install.packages("rpart")

```

Updating Loaded Packages

One or more of the packages to be updated are currently loaded. Restarting R prior to install is highly recommended.

RStudio can restart R before installing the requested packages. All work and data will be preserved during restart.

Do you want to restart R prior to install?

Yes No Cancel

Environment

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",...
x	int [1:12]	1 2 3 4 5 6 7 8 9 10 ...
y		1
Y	int [1:2, 1:5, 1:3]	1 2 3 4 5 6 7 8 9 10...
z		1

```

179
180
181 ##### Decision Tree #####
182 install.packages("rpart")
183 library(rpart)
184 set.seed(123)
185 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
186 edge.RT.Exp <- rpart(log_cStock0_5 ~ elevation + twi + radK +
187   landsat_b3 + landsat_b4, data = DSM_data
188   control = rpart.control(minsplit = 50))
189 summary(edge.RT.Exp)
190 printcp(edge.RT.Exp)
191 plot(edge.RT.Exp)
192 text(edge.RT.Exp)
193

```

```

Call:
rpart(formula = log_cStock0_5 ~ elevation + twi + radK + landsat_b3 +
  landsat_b4, data = DSM_data[training, ], control = rpart.control
(minsplit = 50))
n = 238

      CP nsplit rel error  xerror  xstd
1 0.17447214    0 1.0000000 1.0033526 0.2377129
2 0.03394317    1 0.8255279 0.8558919 0.2347981
3 0.01800947    3 0.7576415 0.8818461 0.2442330
4 0.01740352    4 0.7396320 0.9024220 0.2448611

```

And then you can call this library rpart and then again we are setting the seed that is 123 and then we are selecting the training samples randomly 70 percent just like before and then we are going to use these rpart function here we are trying to predict the log converted 0 to 5 centimetre data organic carbon data with elevation twi, radk, landset b3, landset b4 our data is DSM data with the training dataset.

Here one thing you can see we are using a... an argument that is control, which is rpart dot control minimum split equal to 50. So, that means we are instructing r that you should not go for further splitting the node if the minimum sample is less than 50. So, in this way we can we can control the overfitting of the data because, if it is less than 50 You know, they are might be some kind of overfitting. So, you can play with the number we can change the number at the minimum number of samples to be to be further splitted.

So, let us run this and you can see the summary when you let me just cancel it, so you can see the summary of the results also. So, it will give you the all the details, so this is the formula of and also you can see here the CP parameter also the number of split and also the relative error then x error and then x standard.

So, this x error is the cross-validation error, rpart basically has a built-in cross validation. So, there is a thumb rule that you select the CP value for which that minimize this X error, so this is one thumb rule another thumb rule is you can choose the lowest level, so you can see the lowest level, so you can see the... as the number of splits are increasing that means, the model is becoming more and more complex.

So, you can choose where to stop and where to prune to prevent the overfitting of the model. So, one thumb rule is you select the CP for which you have the you minimise the x error. Another thumb rule is you choose the lowest level for which these rel error plus e standard you know the summation of both of them will be less than this e error.

So, based on these model values, you can select the optimum number of split for which you should run the model and you can prevent the overfitting and also the variable importance you can see here elevation has the maximum variable importance followed by twi, the landset b3, radk and landset b2. So, from there you can have any idea about how much important those variables are for predicting the carbon stock of logarithmic converted carbon stock for 0 to 5 centimetre.

(Refer Slide Time: 34:52)

```

179
180
181 ##### Decision Tree #####
182 install.packages("rpart")
183 library(rpart)
184 set.seed(123)
185 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
186 edge.RT.Exp <- rpart(log_cStock0_5 ~ elevation + twi + radK +
187   landsat_b3 + landsat_b4, data = DSM_data
188   control = rpart.control(minsplit = 50))
189 summary(edge.RT.Exp)
190 printcp(edge.RT.Exp)
191 plot(edge.RT.Exp)
192 text(edge.RT.Exp)
193

```

```

3 0.01800947 3 0.7576415 0.8818461 0.2442330
4 0.01740352 4 0.7396320 0.9024220 0.2448611
5 0.01464920 5 0.7222285 0.9127623 0.2448140
6 0.01000000 6 0.7075793 0.9175818 0.2508899

```

```

I
Variable importance
elevation twi landsat_b3 radK landsat_b4
40 36 14 8 2

```

```

Node number 1: 238 observations, complexity param=0.1744721
mean=2.763495, MSE=0.325683
left son=2 (145 obs) right son=3 (93 obs)

```

```

179
180
181 ##### Decision Tree #####
182 install.packages("rpart")
183 library(rpart)
184 set.seed(123)
185 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
186 edge.RT.Exp <- rpart(log_cStock0_5 ~ elevation + twi + radK +
187   landsat_b3 + landsat_b4, data = DSM_data
188   control = rpart.control(minsplit = 50))
189 summary(edge.RT.Exp)
190 printcp(edge.RT.Exp)
191 plot(edge.RT.Exp)
192 text(edge.RT.Exp)
193

```

```

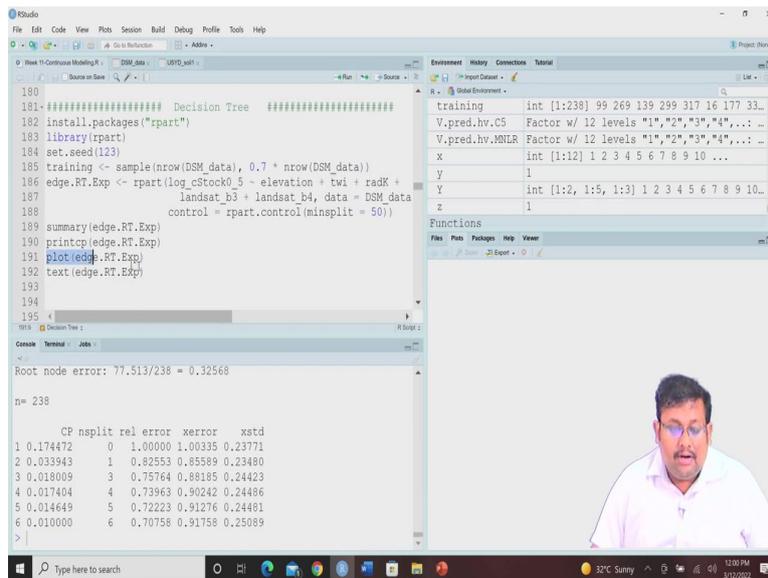
elevation twi landsat_b3 radK landsat_b4
40 36 14 8 2

```

```

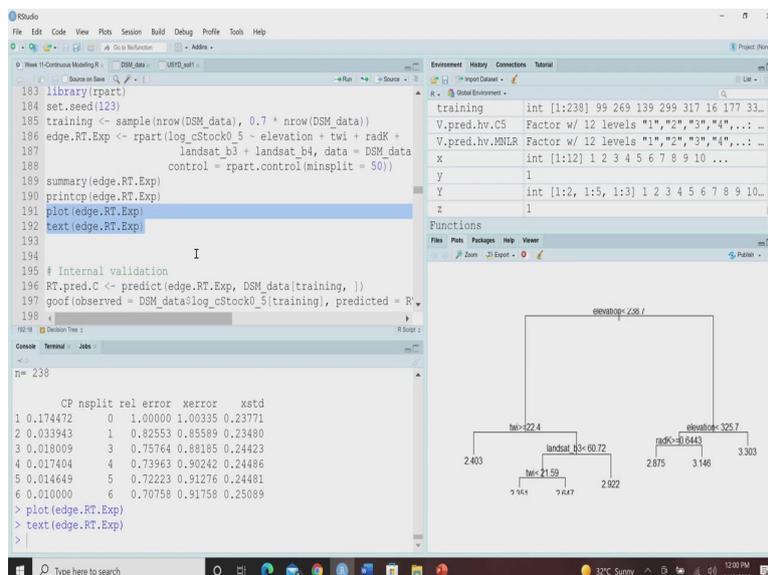
Node number 1: 238 observations, complexity param=0.1744721
mean=2.763495, MSE=0.325683
left son=2 (145 obs) right son=3 (93 obs)
Primary splits:
elevation < 238.7122 to the left, improve=0.17447210, (0 mis
sing)
twi < 21.42162 to the right, improve=0.12716690, (0 mis
sing)
radK < 1.47288 to the left, improve=0.04141532, (0 mis
sing)

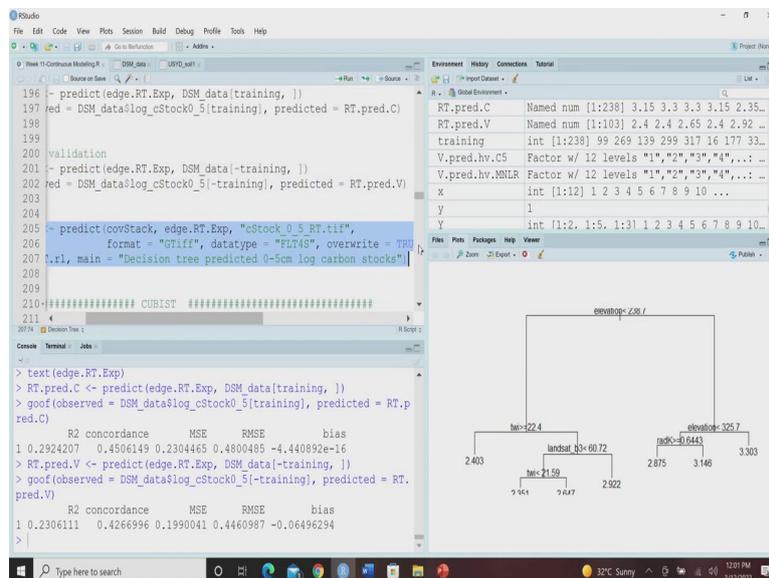
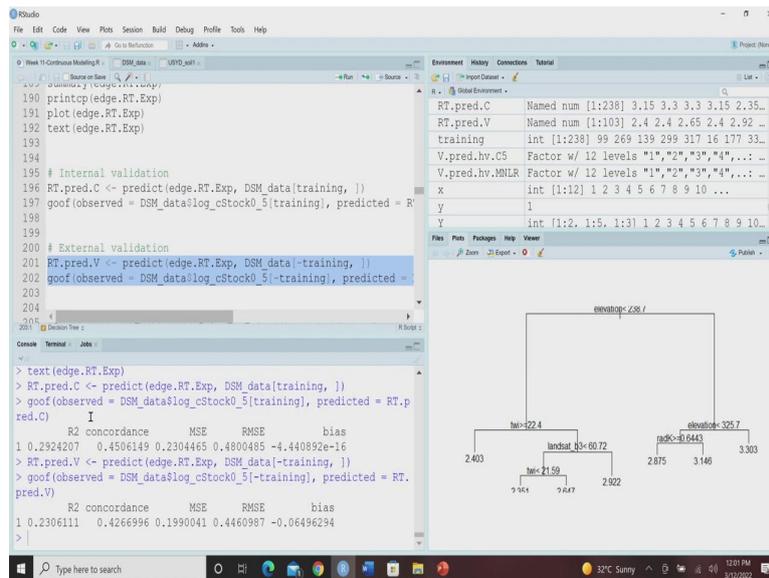
```



Then for each of these nodes, how many observations were there? What was the complexity parameter, what is the mean of observation, what was the MSE of observation, mean squared error, how many observations were in the one side and other how many observation in the other side and then primary splitting rules and all these things you can get from and for all the nodes, you will have these details. So, this is how you create this decision tree, you can print this decision tree also. And also, you can plot this decision tree by using this plot and text comment.

(Refer Slide Time: 35:32)





Of course, there are many other packages called part e part k 8 packages for producing more aesthetically beautiful, decision tree representation, but here you can see some ideas basic ideas. So, now, if we do the internal validation, remember guys internal validation is synonymous to the calibration. So, here we are going to use these you know, calibration.

So, basically using the training samples, we are going to do the internal, internal validation. So, we are getting the results here an external validation is the origin of validation using minus training. So, you can see how the results are. So, you can see here, this is the, results from the calibration and validation.

(Refer Slide Time: 36:35)

This screenshot shows the RStudio interface with the following content:

- Code Editor:** Lines 196-209 show the prediction of `red.C` and `red.V` using `edge.RT.Exp` and `DSM_data`. Lines 201-209 show external validation using `map.RT.r1` and `covStack`. Lines 210-211 show the execution of `CUBIST`.
- Console:** Displays the output of `text(edge.RT.Exp)`, showing R2 concordance, MSE, RMSE, and bias for both training and validation datasets.
- Environment:** Lists objects like `RT.pred.C`, `RT.pred.V`, `training`, `V.pred.hv.C5`, `V.pred.hv.MNLR`, `x`, `y`, and `Y`.
- Plots:** A decision tree plot titled "Decision tree predicted 0-5cm log carbon stocks" with a root node split on `elevation <= 287`.

This screenshot is identical to the one above, showing the same R code, console output, and decision tree plot.

This screenshot shows the RStudio interface with the following content:

- Code Editor:** Lines 196-209 show the prediction of `red.C` and `red.V`. Lines 201-209 show external validation. Lines 210-211 show `CUBIST`. Lines 212-213 show the prediction of `map.RT.r1` using `covStack`. Lines 214-215 show the plot of `map.RT.r1`.
- Console:** Displays the output of `text(edge.RT.Exp)` and the execution of `CUBIST`.
- Environment:** Lists objects like `map.RT.r1`, `mDat`, `mod`, `mod.l`, `mod.data`, `mod.rh`, `model.l`, and `models`.
- Plots:** A raster map plot titled "Decision tree predicted 0-5cm log carbon stocks" showing a spatial distribution of carbon stocks with a color scale from 24 to 32.

And also you can map it for you know, you can map the result map based on the cart model, you can predict based on the stack covariates and you can plot the decision tree predicted organic carbon stock so, this is the decision tree predicted 0 to 5 centimetre log carbon stock and which is using the covariates values for fitting the model and based on those module cart model we have predicted the results and this is the map based on the classification regression tree.

(Refer Slide Time: 37:15)



So, let us wrap up this lecture, I think you have learned something new, please run these codes to gain more and more confidence all these codes are having these annotation for your better understanding, this is the reference and let us wrap up this lecture. And thank you for joining. And in the next lecture we will be going from here we will be discussing the Cubist model and random forest using R. And will see how we can use this cubist model and random forests to produce the map of soil properties. Thank you.