

Machine Learning for Soil and Crop Management
Professor Somsubhra Chakraborty
Agricultural and Food Engineering Department
Indian Institute of Technology, Kharagpur
Lecture 55
Digital Soil Mapping with Continuous Variables (Contd.)

Welcome friends to this last 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. We have in our previous lectures we have seen many you know R codes for exploratory data analysis, GIS operation, geo statistical operations, spline fitting, we have seen also, we have seen how to do the model validation using random holdout and then leave one out cross validation.

We have seen, we have seen how to produce the model using simple linear regression, we have seen how to produce the model using multiple linear regression and map based on multiple linear regression. Also, we have seen how to develop the classification regression to a decision tree using the continuous data and how to produce the map using that continuous, that decision tree model.

(Refer Slide Time: 01:25)



Now, in this lecture we are going to cover these concepts we are going to cover this cubist model, we are going to cover the random forest model and then we are going to discuss a hybrid model that is a hybrid approach that is Universal Kriging.

(Refer Slide Time: 01:38)

KEYWORDS

- Cubist
- Random Forest
- ntree
- cubistControl
- Universal Kriging

The slide features a speaker video inset in the bottom right corner and logos for IITM and NPTEL at the bottom.

So, guys these are the cube these are the keywords for this lecture, cubist, random forest, ntree then cubist control Universal Kriging these we are going to discuss in this lecture.

(Refer Slide Time: 01:55)

CUBIST

- A very popular model structure used within the DSM community.
- Its popularity is due to its ability to “mine” non-linear relationships in data, but does not have the issues of finite predictions that occur for other decision and regression tree models

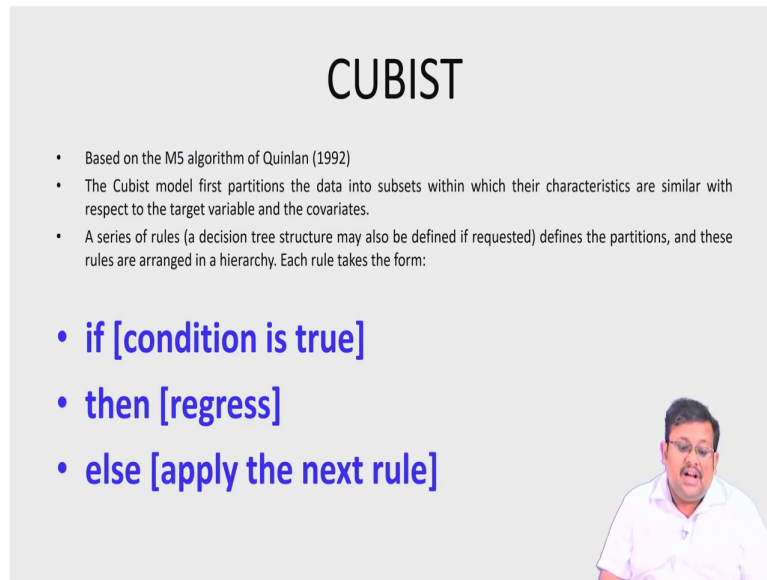
43

Now, first we will start discussing about the Cubist, cubist is a very popular model structure used in the within the DSM community and its popularity is due to its ability to mine nonlinear relationship data but does not have the issue of finite prediction that occur for other decision and regression tree models.

Now, in case of decision tree models, the finite predictions is one of the major issue. So, here cubist can address that problem. So, it is ability to mine that non-linear relationship data, but


it does not have the issue of this finite prediction. So, this is very popular method nowadays for DSM operations.

(Refer Slide Time: 02:39)



CUBIST

- Based on the M5 algorithm of Quinlan (1992)
- The Cubist model first partitions the data into subsets within which their characteristics are similar with respect to the target variable and the covariates.
- A series of rules (a decision tree structure may also be defined if requested) defines the partitions, and these rules are arranged in a hierarchy. Each rule takes the form:
 - **if [condition is true]**
 - **then [regress]**
 - **else [apply the next rule]**




So, this cubist is based on the M5 algorithm of Quinlan, and also this cubist model... how this cubist model works. So, this cubist model works first by partitioning the data into subsets within which their characteristics are similar with respect to the target variable and the covariates and a series of rules, defines these partitions and these rules are managed in a hierarchy. So, each rule takes this form.

So, you can see here, if a condition is true, then you know you go for regression otherwise or else apply the next rule. So, basically partition the data by these if then else rule and when some observations satisfy some clustered by this partition, then we fit the linear regression in each of these nodes. So, it is basically kind of a hybrid between the nonlinear cart as well as linear model. So, that is why it is able to generalise the nonlinear relationship, but at the same time it can maintain the linearity. So, this is a cubist rule, this is the this is the cubist algorithm.

(Refer Slide Time: 03:55)

CUBIST

- The condition may be a simple one based on one covariate or, more often, it comprises a number of covariates. If a condition results in being true then the next step is the prediction of the soil property of interest by ordinary least-squares regression from the covariates within that partition. If the condition is not true then the rule defines the next node in the tree, and the sequence of if, then, else is repeated.
- The result is that the regression equations, though general in form, are local to the partitions and their errors smaller than they would otherwise be.




Now, the conditions may be a simple one based on the one covariate or more often it comprises a number of covariates if a condition results in being true then the next step is the prediction of the soil property of interest by ordinary least squares regression from the covariates within that partition, just like I have seen, I have told you that we partition the data based on the rule if else and you know if then else rule and then we partition the data and based on that partition, we predict the target parameter using our covariate data using lived you know least squares regression linear least squares regression model.

So, if the condition is not true, then the rule defines the next node of the tree and the sequence of it and then else is repeated. So, the results is that regression equation through general inform are very local to the partition and they are error is smaller than they would otherwise be. So, of course, you can see that we are partitioning the data and we are fitting it individual models, linear models in each of this partition. So, though they are general linear, you know, you know this regression equation though their general form they are very much local, because we are fitting the individual models within the individual partitions and as a result of that our error becomes much smaller.

(Refer Slide Time: 05:23)

CUBIST

- Luckily, fitting a Cubist model in R is not too difficult—although it will be useful to spend some time playing around with many of the controllable parameters the function has.
- In the example we will try today we can control the number of potential rules that could potentially partition the data (note this limits the number of possible rules, and does not necessarily mean that those number of rules will actually be realized i.e. the outcome is internally optimised).
- We can also limit the extrapolation of the model predictions, which is a useful model constraint feature. These various control parameters plus others can be adjusted within the cubistControl parameter.
- Does not unnecessarily overfits the data



Now, luckily fitting this cubist model in R is not too difficult although it will be useful to spend some time for playing around with many of the controllable parameters the function says, so there is a parameter called cubist control. So, use this cubist control parameter to tune the model. So, if the example will try today, we can control the number of potential rules that could potentially partition the data so, we can control the number of rules this limits the number of possible rules and does not necessarily means that those number of rules will actually be realised, maybe we can give 10 rules, but the algorithm may go with only two rules.

So, when you know after the optimization and also, we can limit the excerpt of extrapolation of the model prediction which is an useful model constraint feature and these variates various control parameters can be adjusted within this cube control parameters. And the one of the benefit of using this cube is easily does not overfit and unnecessarily overfit the data.

(Refer Slide Time: 06:44)

```
209 ##### CUBIST #####
210 install.packages("Cubist")
211 library(Cubist)
212 library(MASS)
213 set.seed(123)
214 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
215 mDat <- DSM_data[training, ]
216 # fit the model
217 edge.cub.Exp <- cubist(x = mDat[, c("elevation", "twi", "radK",
218                                "landsat_b4")], y = mDat$log_cstock0_5,
219                       cubistControl(rules = 5, extrapolation =
220                                     ))
221 summary(edge.cub.Exp)
222
223 # Internal validation
224
225 R2 concordance MSE RMSE bias
226 1 0.2924207 0.4506149 0.2304465 0.4800485 -4.440892e-16
227 > RT.pred.V <- predict(edge.RT.Exp, DSM_data[-training, ])
228 > goof(observed = DSM_data$log_cstock0_5[-training], predicted = RT.
229     pred.V)
230
231 R2 concordance MSE RMSE bias
232 1 0.2306111 0.4266996 0.1990041 0.4460987 -0.06496294
233 > map.RT.rl <- predict(covStack, edge.RT.Exp, "cstock_0_5_RT.tif",
234                       format = "Griff", datatype = "FLT4S", overwri
235                       te = TRUE)
236 > plot(map.RT.rl, main = "Decision tree predicted 0-5cm log carbon s
237     tocks")
238 >
```

Environment: Global Environment

@map.RT.rl	Formal class RasterLayer
@mDat	Formal class SpatialPointsDataFrame
@mod	2 obs. of 9 variables
@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

Console: Terminal Jobs

```
> map.RT.rl <- predict(covStack, edge.RT.Exp, "cstock_0_5_RT.tif",
+ format = "Griff", datatype = "FLT4S", overwri
+ te = TRUE)
> plot(map.RT.rl, main = "Decision tree predicted 0-5cm log carbon s
+ tocks")
>
```

```
209 ##### CUBIST #####
210 install.packages("Cubist")
211 library(Cubist)
212 library(MASS)
213 set.seed(123)
214 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
215 mDat <- DSM_data[training, ]
216 # fit the model
217 edge.cub.Exp <- cubist(x = mDat[, c("elevation", "twi", "radK",
218                                "landsat_b4")], y = mDat$log_cstock0_5,
219                       cubistControl(rules = 5, extrapolation =
220                                     ))
221 summary(edge.cub.Exp)
222
223 # Internal validation
224
225 R2 concordance MSE RMSE bias
226 1 0.2306111 0.4266996 0.1990041 0.4460987 -0.06496294
227 > map.RT.rl <- predict(covStack, edge.RT.Exp, "cstock_0_5_RT.tif",
228                       format = "Griff", datatype = "FLT4S", overwri
229                       te = TRUE)
230 > plot(map.RT.rl, main = "Decision tree predicted 0-5cm log carbon s
231     tocks")
232 > library(Cubist)
233 Loading required package: lattice
234 > library(MASS)
235 > set.seed(123)
236 > training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
237 > mDat <- DSM_data[training, ]
238 >
```

Environment: Global Environment

pred.nv.MALL	factor w/ 12 levels "1","2","3","4",...
predl	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...
RF.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 [1:238] 99 269 139 299 317 16 177 33...

Console: Terminal Jobs

```
> map.RT.rl <- predict(covStack, edge.RT.Exp, "cstock_0_5_RT.tif",
+ format = "Griff", datatype = "FLT4S", overwri
+ te = TRUE)
> plot(map.RT.rl, main = "Decision tree predicted 0-5cm log carbon s
+ tocks")
> library(Cubist)
Loading required package: lattice
> library(MASS)
> set.seed(123)
> training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
> mDat <- DSM_data[training, ]
>
```

```
209 ##### CUBIST #####
210 install.packages("Cubist")
211 library(Cubist)
212 library(MASS)
213 set.seed(123)
214 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
215 mDat <- DSM_data[training, ]
216 # fit the model
217 edge.cub.Exp <- cubist(x = mDat[, c("elevation", "twi", "radK",
218                                "landsat_b3",
219                                "landsat_b4")], y = mDat$log_cstock0_5,
220                       cubistControl(rules = 5, extrapolation = 5, committees = 1)
221                                     ))
222 summary(edge.cub.Exp)
223
224 # Internal validation
225
226 R2 concordance MSE RMSE bias
227 1 0.2306111 0.4266996 0.1990041 0.4460987 -0.06496294
228 > map.RT.rl <- predict(covStack, edge.RT.Exp, "cstock_0_5_RT.tif",
229                       format = "Griff", datatype = "FLT4S", overwri
230                       te = TRUE)
231 > plot(map.RT.rl, main = "Decision tree predicted 0-5cm log carbon s
232     tocks")
233 > library(Cubist)
234 Loading required package: lattice
235 > library(MASS)
236 > set.seed(123)
237 > training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
238 > mDat <- DSM_data[training, ]
239 >
```

Environment: Global Environment

@mDat	238 obs. of 8 variables
@mod	2 obs. of 9 variables
@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

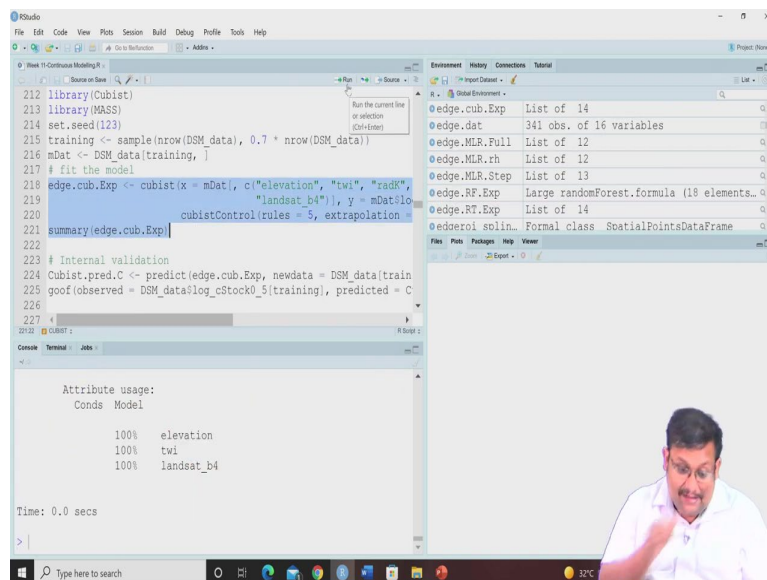
Console: Terminal Jobs

```
> map.RT.rl <- predict(covStack, edge.RT.Exp, "cstock_0_5_RT.tif",
+ format = "Griff", datatype = "FLT4S", overwri
+ te = TRUE)
> plot(map.RT.rl, main = "Decision tree predicted 0-5cm log carbon s
+ tocks")
> library(Cubist)
Loading required package: lattice
> library(MASS)
> set.seed(123)
> training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
> mDat <- DSM_data[training, ]
>
```

So, let us go ahead and see how we can run this model. So, we have done this card and sorry decision tree and based prediction in our previous lecture. So, here we are going to start with the cubist. So, for cubist we are going to install this cubist package. So, once we have installed the cubist package, let us run this let me just you know, we can call this library cubist, and then we can call this library mass and then we are setting the seed as 123 and then again we are selecting the 70 percent of the samples in the training data set and let us call these training data as model data or m dat. So, these training data calibration data let us call it as m dat or model data.

Now, let us fit the cubist model. So, for fitting the cubist model here we are having our x is our model data and our predictors are elevation twi, radk, landset b3, landset b4 and our y is the target parameter that is the carbon stock of 0 to 5 centimetre within this model data our cubist control parameters you can see it is cubist control parameter, we are specifying the 5 number of rows and we are specifying the five extrapolation and number of committees will be 1. So, and so, this is very simple and let us run this summary of this cubist model.

(Refer Slide Time: 08:30)



```
212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # fit the model
218 edge.cub.Exp <- cubist(x = mDat, c("elevation", "twi", "radk",
219                               "landset_b4"), y = mDat$log_cStock0_5,
220                       cubistControl(rules = 5, extrapolation =
221                                     5))
222
223 # Internal validation
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 goof(observed = DSM_data$log_cStock0_5[training], predicted = C)
226
227 # Summary
228 summary(edge.cub.Exp)
```

Attribute usage:

Conds	Model
100%	elevation
100%	twi
100%	landset_b4

Time: 0.0 secs

>

```

212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # fit the model
218 edge.cub.Exp <- cubist(x = mDat[, c("elevation", "twi", "radK",
219                               "landsat_b4")], y = mDat$log_cstock0_5,
220                      cubistControl(rules = 5, extrapolation = 5))
221 summary(edge.cub.Exp)
222
223 # Internal validation
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 goof(observed = DSM_data$log_cstock0_5[training], predicted = Cubist.pred.C)
226
227

```

Call:

```

cubist.default(x = mDat[, c("elevation", "twi",
mDat$log_cstock0_5, committees = 1, control
= cubistControl(rules = 5, extrapolation = 5))

```

Cubist [Release 2.07 GPL Edition] Sat Mar 12 12:12:41 2022

```

Call:
cubist.default(x = mDat[, c("elevation", "twi",
mDat$log_cstock0_5, committees = 1, control
= cubistControl(rules = 5, extrapolation = 5))

Cubist [Release 2.07 GPL Edition] Sat Mar 12 12:12:41 2022

Target attribute 'outcome'

Read 238 cases (6 attributes) from undefined.data

```

```

Cubist [Release 2.07 GPL Edition] Sat Mar 12 12:12:41 2022

Target attribute 'outcome'

Read 238 cases (6 attributes) from undefined.data

Model:

Rule 1: [238 cases, mean 2.7634952, range -1.147828 to 4.533301, est err 0.3358926]

```



```

212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # fit the model
218 edge.cub.Exp <- cubist(x = mDat, c("elevation", "twi", "radK",
219                               "landsat_b4"), y = mDat$log_cstock0_5,
220                      cubistControl(rules = 5, extrapolation =
221                                "summary(edge.cub.Exp)"))
222
223 # Internal validation
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 goof(observed = DSM_data$log_cstock0_5[training], predicted = Cubist.pred.C)
226
227 # Output:
228
229 Model:
230
231 Rule 1: [238 cases, mean 2.7634952, range -1.147828 to 4.533301, error 0.3358926]
232
233 outcome = -0.409619 + 0.0066 elevation + 0.063 twi + 0.0042 landsat_b4
234
235 Evaluation on training data (238 cases):
236
237 Average |error|      0.3968150
238 Relative |error| 0.99
239 Correlation coefficient 0.28

```

```

212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # fit the model
218 edge.cub.Exp <- cubist(x = mDat, c("elevation", "twi", "radK",
219                               "landsat_b4"), y = mDat$log_cstock0_5,
220                      cubistControl(rules = 5, extrapolation =
221                                "summary(edge.cub.Exp)"))
222
223 # Internal validation
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 goof(observed = DSM_data$log_cstock0_5[training], predicted = Cubist.pred.C)
226
227 # Output:
228
229 Model:
230
231 Rule 1: [238 cases, mean 2.7634952, range -1.147828 to 4.533301, error 0.3358926]
232
233 outcome = -0.409619 + 0.0066 elevation + 0.063 twi + 0.0042 landsat_b4
234
235 Evaluation on training data (238 cases):
236
237 Average |error|      0.3968150
238 Relative |error| 0.99
239 Correlation coefficient 0.28

```

```

212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # fit the model
218 edge.cub.Exp <- cubist(x = mDat, c("elevation", "twi", "radK",
219                               "landsat_b4"), y = mDat$log_cstock0_5,
220                      cubistControl(rules = 5, extrapolation =
221                                "summary(edge.cub.Exp)"))
222
223 # Internal validation
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 goof(observed = DSM_data$log_cstock0_5[training], predicted = Cubist.pred.C)
226
227 # Output:
228
229 Model:
230
231 Rule 1: [238 cases, mean 2.7634952, range -1.147828 to 4.533301, error 0.3358926]
232
233 outcome = -0.409619 + 0.0066 elevation + 0.063 twi + 0.0042 landsat_b4
234
235 Evaluation on training data (238 cases):
236
237 Average |error|      0.3968150
238 Relative |error| 0.99
239 Correlation coefficient 0.28

```

So, if you run this summary of the cubist model, this is the model results. So, you can see this is the model initial model input and then you can see that target attribute in this case is outcome and we have 238 cases with 6 attributes and then rule 1 is basically using all the 238 cases the mean is 2.76 the range of values is given and estimated error is also given outcome is this model.

So, this model which they have fit this linear regression model which they have fit for this first rule is you know outcome is this is the intercept plus elevation with the slope then twi with the slope and landset b4 with the slope and then you can see that what are the average error what are the linearity error and that is the correlation coefficient.

(Refer Slide Time: 09:34)

```

212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # fit the model
218 edge.cub.Exp <- cubist(x = mDat, c("elevation", "twi", "radk",
219                               "landsat_b4"), y = mDat$log_cstock0_5,
220                      cubistControl(rules = 5, extrapolation = 1))
221 summary(edge.cub.Exp)
222
223 # Internal validation
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 goof(observed = DSM_data$log_cstock0_5[training], predicted = Cubist.pred.C)
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

```

Console output:

```

Relative error|      0.99
Correlation coefficient      0.28

Attribute usage:
Conds Model      I
      100% elevation
      100% twi
      100% landsat_b4

```

```

212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # fit the model
218 edge.cub.Exp <- cubist(x = mDat, c("elevation", "twi", "radk", "landsat_b3",
219                               "landsat_b4"), y = mDat$log_cstock0_5,
220                      cubistControl(rules = 5, extrapolation = 5, committees = 1))
221 summary(edge.cub.Exp)
222
223 # Internal validation
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 goof(observed = DSM_data$log_cstock0_5[training], predicted = Cubist.pred.C)
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

```

Console output:

```

Relative error|      0.99
Correlation coefficient      0.28

Attribute usage:
Conds Model      I
      100% elevation
      100% twi
      100% landsat_b4

```

And then you can see that you know, how much these variables have been utilised in the model. So, from there you can see one important thing that although we have fixed you know, five rules, it has used only one role after optimization. So, this is one of the important feature of cubist.

(Refer Slide Time: 10:00)

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

```
212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # fit the model
218 edge.cub.Exp <- cubist(x = mDat[, c("elevation", "twi", "radK",
219                               "landsat_b4")], y = mDat$logStock0_5)
220 cubistControl(rules = 5, extrapolation = 5)
221 summary(edge.cub.Exp)
222 # Internal validation
223 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
224 goof(observed = DSM_data$logStock0_5[training], predicted = Cubist.pred.C)
```

The Environment pane on the right shows the following objects:

- @edge.cub.Exp List of 14
- @edge.dat 341 obs. of 16 variables
- @edge.MLR.Full List of 12
- @edge.MLR.rh List of 12
- @edge.MLR.Step List of 13
- @edge.RF.Exp Large randomForest.formula (18 elements...
- @edge.RF.Exp List of 14
- @edge.roi_solin_ Formal class SpatialPointsDataFrame

The Console shows the output of the summary function:

```
Attribute usage:
Conds Model
      100% elevation
      100% twi
      100% landsat_b4
```

Time: 0.0 secs

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

```
212 Cubist)
213 MASS)
214 (123)
215 } <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 DSM_data[training, ]
217 ie model)
218 ).Exp <- cubist(x = mDat[, c("elevation", "twi", "radK", "landsat_b4")], y = mDat$log_cStock0_5)
219 cubistControl(rules = 5, extrapolation = 5), comm
220 edge.cub.Exp)
221 al validation
222 red.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
223 erved = DSM_data$log_cStock0_5[training], predicted = Cubist.pr
```

The Environment pane on the right shows the following objects:

- @edge.cub.Exp List of 14
- @edge.dat 341 obs. of 16 variables
- @edge.MLR.Full List of 12
- @edge.MLR.rh List of 12
- @edge.MLR.Step List of 13
- @edge.RF.Exp Large randomForest.formula (18 elements...
- @edge.RF.Exp List of 14
- @edge.roi_solin_ Formal class SpatialPointsDataFrame

The Console shows the output of the summary function:

```
Attribute usage:
Conds Model
      100% elevation
      100% twi
      100% landsat_b4
```

Time: 0.0 secs

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
0 Week 11-Continuous Modeling R
Source Editor: 212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # fit the model
218 edge.cub.Exp <- cubist(x = mDat[, c("elevation", "twi", "radK",
219 "landsat_b4")], y = mDat$log_cStock0_5,
220 cubistControl(rules = 5, extrapolation =
221 summary(edge.cub.Exp)
222
223 # Internal validation
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 goof(observed = DSM_data$log_cStock0_5[training], predicted = C
226
227
228
229
3058
Cubist:
Conds Model
100% elevation
100% twi
100% landsat_b4
Time: 0.0 secs
> Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
>

```

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
0 Week 11-Continuous Modeling R
Source Editor: 212 }
213
214
215 mple(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 ata[training, ]
217 l
218 - cubist(x = mDat[, c("elevation", "twi", "radK", "landsat_b3",
219 "landsat_b4")], y = mDat$log_cStock0_5,
220 cubistControl(rules = 5, extrapolation = 5), committees
221 ub.Exp)
222
223 idation
224 <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 = DSM_data$log_cStock0_5[training], predicted = Cubist.pred.C)
226
227
228
229
3057
Cubist:
100% landsat_b4
Time: 0.0 secs
> Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
> goof(observed = DSM_data$log_cStock0_5[training], predicted = Cubist.pred.C)
R2 concordance MSE RMSE bias
1 0.1774503 0.3173758 0.2678904 0.5175813 -0.009558701
>

```

Now, let us do some internal validation. So, for this internal validation We are going to use again the you know first we are going to predict using the predict our dataset based on these training samples. And then internal validation means again the calibration so again goodness of fit statistics observed is our training data set 0 to 5 centimetre carbon data predicted which we have just predicted here cubist dot predict dot C here.

(Refer Slide Time: 10:30)

```
212 library(Cubist)
213 library(MASS)
214 set.seed(123)
215 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 mDat <- DSM_data[training, ]
217 # Fit the model
218 edge.cub.Exp <- cubist(x = mDat[, c("elevation", "twi", "radK",
219                                "landsat_b4")], y = mDat$log_cStock0_5,
220                       cubistControl(rules = 5, extrapolation = 5))
221 summary(edge.cub.Exp)
222
223 # Internal validation
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 goof(observed = DSM_data$log_cStock0_5[training], predicted = Cubist.pred.C)
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
```

Conds Model

100%	elevation
100%	twi
100%	landsat_b4

Time: 0.0 secs

```
> Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
>
```

```
212 }
213 }
214 }
215 mple(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 ta[training, ]
217 ll
218 - cubist(x = mDat[, c("elevation", "twi", "radK", "landsat_b3",
219                    "landsat_b4")], y = mDat$log_cStock0_5,
220          cubistControl(rules = 5, extrapolation = 5), committees = 5)
221 cub.Exp)
222
223 # Internal validation
224 <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 = DSM_data$log_cStock0_5[training], predicted = Cubist.pred.C)
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
```

100% landsat_b4

Time: 0.0 secs

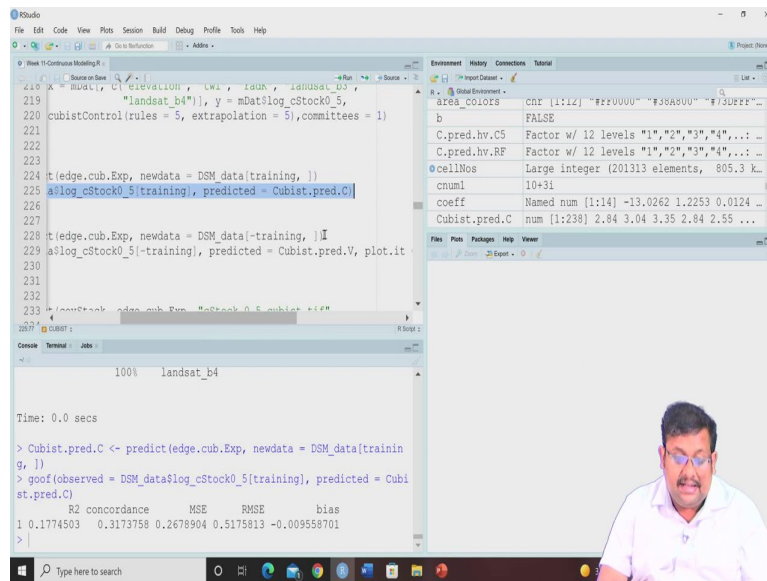
```
> Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
> goof(observed = DSM_data$log_cStock0_5[training], predicted = Cubist.pred.C)
      R2 concordance    MSE    RMSE    bias
1 0.1774503 0.3173758 0.2678904 0.5175813 -0.009558701
>
```

```
212 }
213 }
214 }
215 mple(nrow(DSM_data), 0.7 * nrow(DSM_data))
216 ta[training, ]
217 ll
218 - cubist(x = mDat[, c("elevation", "twi", "radK", "landsat_b3",
219                    "landsat_b4")], y = mDat$log_cStock0_5,
220          cubistControl(rules = 5, extrapolation = 5), committees = 5)
221 cub.Exp)
222
223 # Internal validation
224 <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
225 = DSM_data$log_cStock0_5[training], predicted = Cubist.pred.C)
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
```

100% landsat_b4

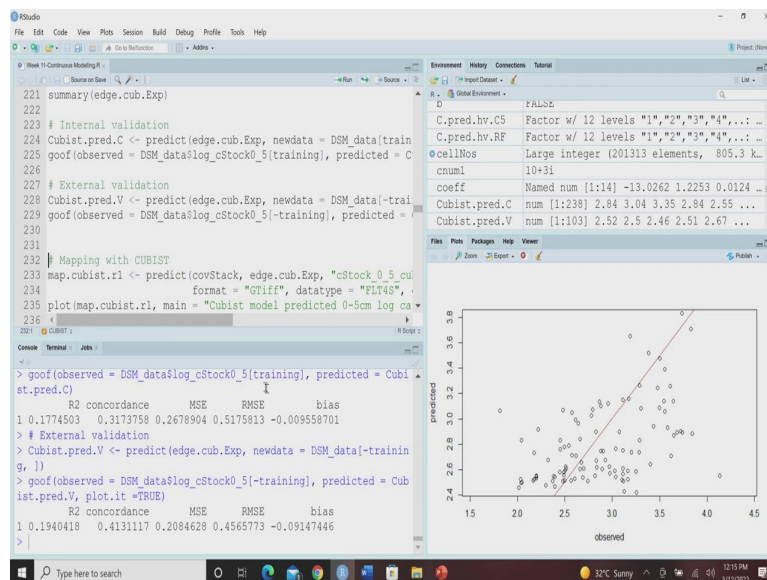
Time: 0.0 secs

```
> Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training, ])
> goof(observed = DSM_data$log_cStock0_5[training], predicted = Cubist.pred.C)
      R2 concordance    MSE    RMSE    bias
1 0.1774503 0.3173758 0.2678904 0.5175813 -0.009558701
>
```



So, if we run this thing, we will see we will see the model results So, R square 0.17 concordance 0.31, MSE 0.26, RMSE 0.51, bias is 0.00 external validation again for validation data set minus training you can see we are first predicting the values and then from the observed values and the predicted values again we are going to use this goof function and we want to plot it simultaneously.

(Refer Slide Time: 11:05)



```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
0 Week 11-Continuous Modelling R
Source
224 lic(edge.cub.Exp, newdata = DSM_data[training,])
225 lats$log_cStock0_5[training, predicted = Cubist.pred.C)
226
227
228 lic(edge.cub.Exp, newdata = DSM_data[-training,])
229 lats$log_cStock0_5[-training, predicted = Cubist.pred.V, plot.it
230
231
232
233 lic(covStack, edge.cub.Exp, "cStock_0_5_cubist.tif",
234     format = "GTiff", datatype = "FLT4S", overwrite = TRUE)
235 main = "Cubist model predicted 0-5cm log carbon
236
237
238
239-
281 cubist
Console Terminal Jobs
> goof(observed = DSM_data$log_cStock0_5[training, predicted = Cubi
st.pred.C)
R2 concordance MSE RMSE bias
1 0.1774503 0.3173758 0.2678904 0.5175813 -0.009558701
> # External validation
> Cubist.pred.V <- predict(edge.cub.Exp, newdata = DSM_data[-traini
g,])
> goof(observed = DSM_data$log_cStock0_5[-training, predicted = Cubi
st.pred.V, plot.it = TRUE)
R2 concordance MSE RMSE bias
1 0.1940418 0.4131117 0.2084628 0.4565773 -0.09147446
>

```

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
0 Week 11-Continuous Modelling R
Source
224 Cubist.pred.C <- predict(edge.cub.Exp, newdata = DSM_data[training,])
225 goof(observed = DSM_data$log_cStock0_5[training, predicted = Cubi
st.pred.C)
226
227 # External validation
228 Cubist.pred.V <- predict(edge.cub.Exp, newdata = DSM_data[-traini
g,])
229 goof(observed = DSM_data$log_cStock0_5[-training, predicted = Cubi
st.pred.V, plot.it = TRUE)
230
231
232 # Mapping with CUBIST
233 map.cubist.r1 <- predict(covStack, edge.cub.Exp, "cStock_0_5_cu
bist.tif",
234     format = "GTiff", datatype = "FLT4S",
235     overwrite = TRUE)
236 plot(map.cubist.r1, main = "Cubist model predicted 0-5cm log ca
rbon stocks (0-5cm)")
237
238
239-
281 cubist
Console Terminal Jobs
> Cubist.pred.V <- predict(edge.cub.Exp, newdata = DSM_data[-traini
g,])
> goof(observed = DSM_data$log_cStock0_5[-training, predicted = Cubi
st.pred.V, plot.it = TRUE)
R2 concordance MSE RMSE bias
1 0.1940418 0.4131117 0.2084628 0.4565773 -0.09147446
> # Mapping with CUBIST
> map.cubist.r1 <- predict(covStack, edge.cub.Exp, "cStock_0_5_cu
bist.tif",
+     format = "GTiff", datatype = "FLT4S", ove
rwrite = TRUE)
> plot(map.cubist.r1, main = "Cubist model predicted 0-5cm log carbo
n stocks (0-5cm)")
>

```

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
0 Week 11-Continuous Modelling R
Source
228 Cubist.pred.V <- predict(edge.cub.Exp, newdata = DSM_data[-traini
g,])
229 goof(observed = DSM_data$log_cStock0_5[-training, predicted = Cubi
st.pred.V, plot.it = TRUE)
230
231
232 # Mapping with CUBIST
233 map.cubist.r1 <- predict(covStack, edge.cub.Exp, "cStock_0_5_cu
bist.tif",
234     format = "GTiff", datatype = "FLT4S",
235     overwrite = TRUE)
236 plot(map.cubist.r1, main = "Cubist model predicted 0-5cm log carbo
n stocks (0-5cm)")
237
238
239-
240
241 install.packages("randomForest")
242 library(randomForest)
243
281 Random Forest
Console Terminal Jobs
ist.pred.V, plot.it = TRUE)
R2 concordance MSE RMSE bias
1 0.1940418 0.4131117 0.2084628 0.4565773 -0.09147446
> # Mapping with CUBIST
> map.cubist.r1 <- predict(covStack, edge.cub.Exp, "cStock_0_5_cu
bist.tif",
+     format = "GTiff", datatype = "FLT4S", ove
rwrite = TRUE)
> plot(map.cubist.r1, main = "Cubist model predicted 0-5cm log carbo
n stocks (0-5cm)")
>

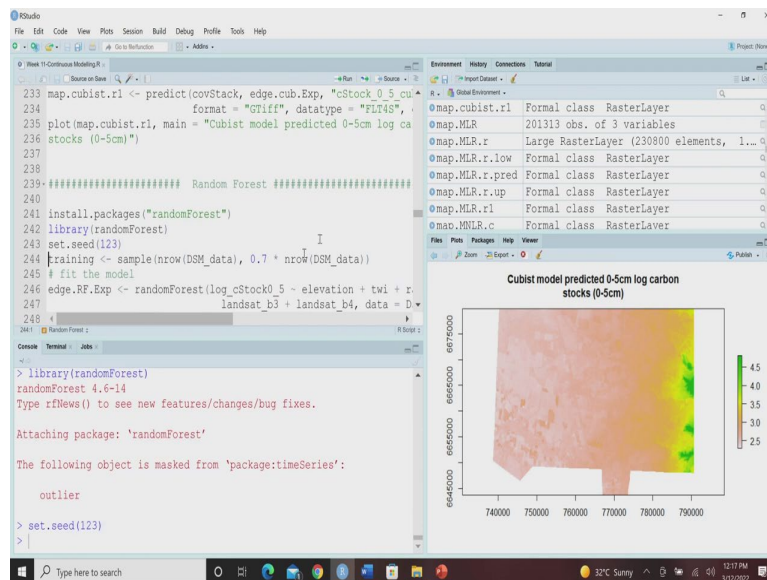
```

So, you can see this is the result 0.19 concordance I may see RMSE bias, when you do not one thing when using the goof function, when you do not specify any argument of type, then by default it will give you the result for digital soil mapping or DSM if you specify the argument at either DSM or spec, then only they will they will take it into consideration, but if you do not specify the argument, then it will by default it will take the DSM argument.

So, this is a predicted versus measured results and then if you want to map based on this cube is values, so, basically you have to use this predict function and then you know based on this predict function our model is edge dot cube dot exp our covStack remember whatever modelling, whatever mapping you are doing, you have to based on these covariate data.

So, covariate data stack covariate is taking into consideration so, we are running it and then if you want to plot this, you just have to use this plot function and see how it will appear. So, this is the cubist model predicted 0 to 5 centimetre no log carbonStock. So, using the cubist model, so, this is how you can guys you can predict the view based on the cubist model and you can produce the map of soil properties using the cubist predicted values. So, we have completed the cubist model.

(Refer Slide Time: 12:58)



RStudio interface showing a script for Random Forest modeling and a map of predicted carbon stocks.

```

237
238
239##### Random Forest #####
240
241install.packages("randomForest")
242library(randomForest)
243set.seed(123)
244training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
245# fit the model
246edge.RF.Exp <- randomForest(log_cStock0_5 ~ elevation + twi + r
247                           landsat_b3 + landsat_b4, data = D
248                           importance = TRUE, ntree = 1000)
249print(edge.RF.Exp)
250
251#Visualize the importance of covariates
252
253##### Random Forest #####
254
255randomForest 4.6-14
256Type rfNews() to see new features/changes/bug fixes.
257
258Attaching package: 'randomForest'
259
260The following object is masked from 'package:timeSeries':
261
262   outlier
263
264> set.seed(123)
265> training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
266>
  
```

Environment window:

preo.nv.MNLK	factor w/ 12 levels "1", "2", "3", "4", ...
pred1	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...
RR.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 [1:238] 99 269 139 299 317 16 177 33...

Map: Cubist model predicted 0-5cm log carbon stocks (0-5cm)

RStudio interface showing a script for Random Forest modeling and a map of predicted carbon stocks.

```

237
238
239##### Random Forest #####
240
241install.packages("randomForest")
242library(randomForest)
243set.seed(123)
244training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
245# fit the model
246edge.RF.Exp <- randomForest(log_cStock0_5 ~ elevation + twi + radK +
247                           landsat_b3 + landsat_b4, data = DSM_data(tr
248                           importance = TRUE, ntree = 1000)
249print(edge.RF.Exp)
250
251# the importance of covariates
252
253##### Random Forest #####
254
255randomForest 4.6-14
256Type rfNews() to see new features/changes/bug fixes.
257
258Attaching package: 'randomForest'
259
260The following object is masked from 'package:timeSeries':
261
262   outlier
263
264> set.seed(123)
265> training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
266>
  
```

Environment window:

preo.nv.MNLK	factor w/ 12 levels "1", "2", "3", "4", ...
pred1	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...
RR.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 [1:238] 99 269 139 299 317 16 177 33...

Map: Cubist model predicted 0-5cm log carbon stocks (0-5cm)

RStudio interface showing a script for Random Forest modeling and a map of predicted carbon stocks.

```

237
238
239##### Random Forest #####
240
241install.packages("randomForest")
242library(randomForest)
243set.seed(123)
244training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
245# fit the model
246edge.RF.Exp <- randomForest(log_cStock0_5 ~ elevation + twi + radK +
247                           landsat_b3 + landsat_b4, data = DSM_data(training, ),
248                           importance = TRUE, ntree = 1000)
249print(edge.RF.Exp)
250
251# the importance of covariates
252
253##### Random Forest #####
254
255randomForest 4.6-14
256Type rfNews() to see new features/changes/bug fixes.
257
258Attaching package: 'randomForest'
259
260The following object is masked from 'package:timeSeries':
261
262   outlier
263
264> set.seed(123)
265> training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
266>
  
```

Environment window:

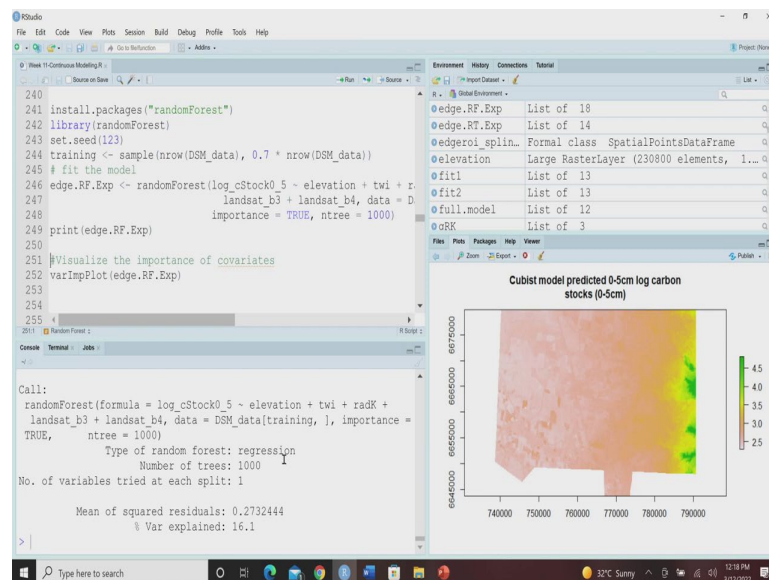
preo.nv.MNLK	factor w/ 12 levels "1", "2", "3", "4", ...
pred1	Named num [1:101] 1.18 4.27 2.1 1.27 1.1...
RF.pred.C	Named num [1:238] 2.85 3.16 3.21 2.98 2...
RF.pred.V	Named num [1:103] 2.51 2.46 2.51 2.42 2...
RR.preds.fin	num [1:103] 2.54 2.52 2.48 2.54 2.7...
RT.pred.C	Named num [1:238] 3.15 3.3 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 [1:238] 99 269 139 299 317 16 177 33...

Map: Cubist model predicted 0-5cm log carbon stocks (0-5cm)

Now, the next important model, which has been widely used in DSM domain is the random forest for Random Forests the in the you have to install this package, random forest. So, please go ahead and install, I have already installed so, I am not going to further install I am just going to call this library random forest.

Again, I am setting the seed 123 for selecting randomly selecting the data, the calibration set. So, here we are selecting the 70 percent of the data in the calibration set and then we are fitting the model so, for fitting the model the function is random forest function here we are targeting these log carbon stocks 0 to 5 centimetre our parameters or variables or elevation twi, radk, landsat b3, landsat b4 data is the training data set of the you know of the of these DSM underscore data and importance true and the number of tree we want to grow is 1000 number of trees. So, the model is now built.

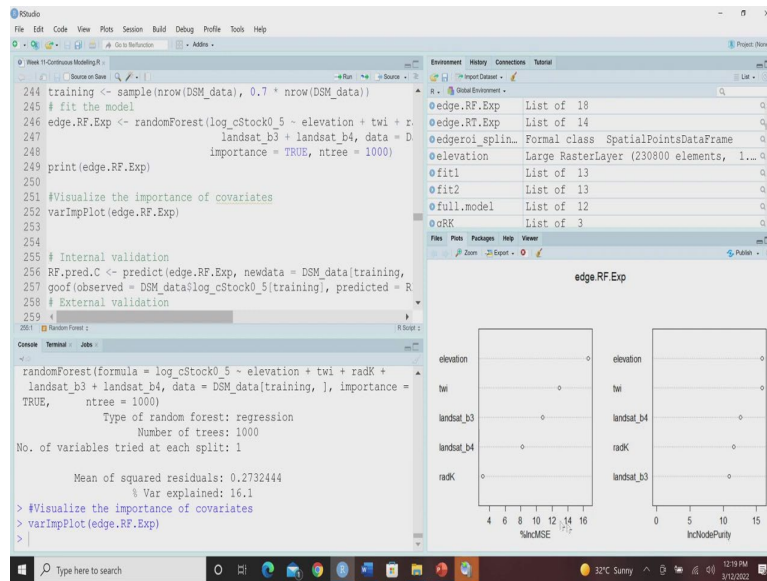
(Refer Slide Time: 14:00)



Now, let us see how this model will. So, here you can see this is the output of the model. So, type of random forest is a regression because our target is a continuous variable number of trees we have grown 1000, number of variables tried at each split is 1, mean of squared residuals 0.27, so MSE is basically 0.27.

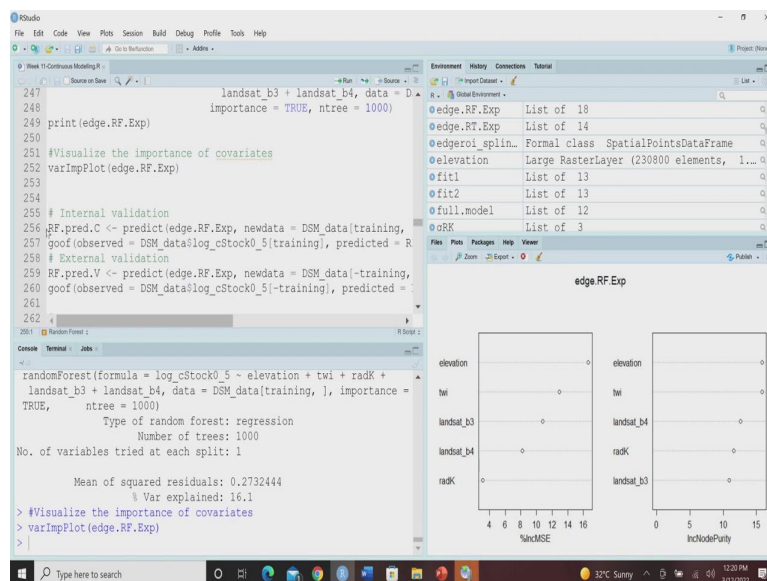
So, if you take the square root of it, it will give you the RMSE and number of way and the percentage of variance explained is 16.1 percentage. So, you can see here this is the result from the random forest. Now, another very important thing is using the random forest, you will be able to get the variable importance that means you can map or you can produce the plot showing which variable is more important than other variables. So, for that we are going to use this varImpPlot function and this is the our full model.

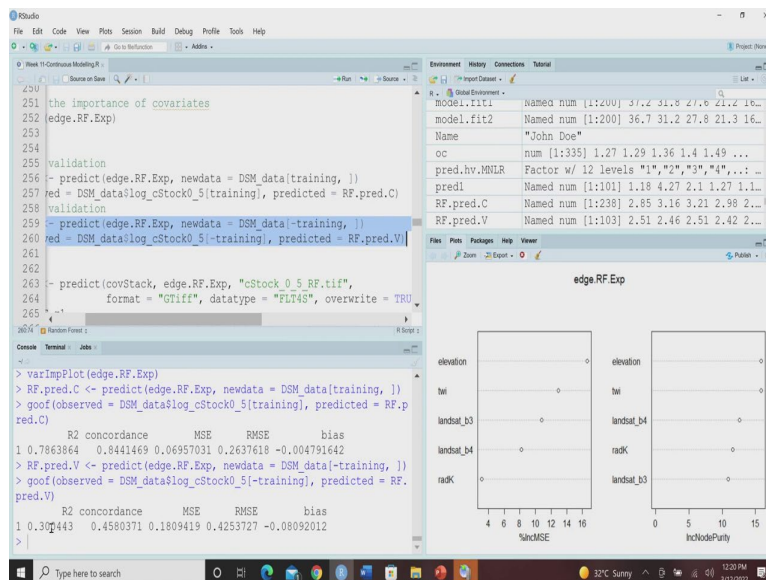
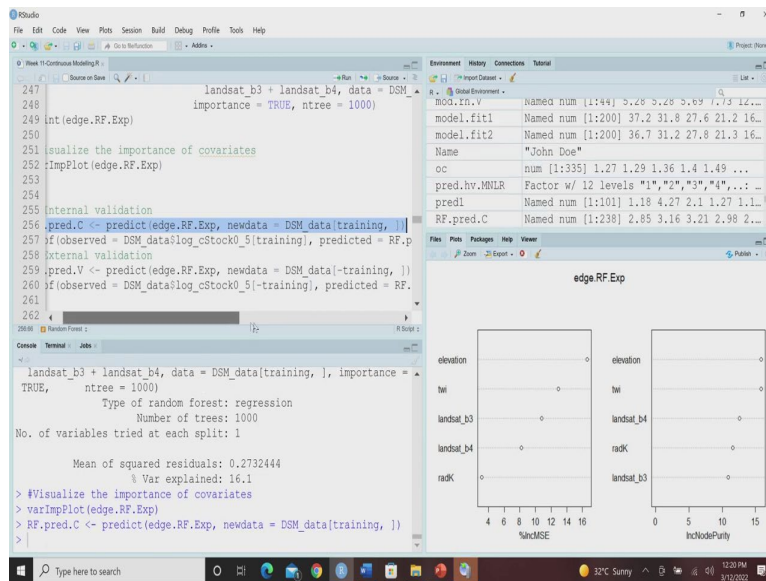
(Refer Slide Time: 15:00)



So, let us run it and you can see here these two expressions or these two plots will be generated based on percentage increase in MSE and increase in Node purity. So, you see that percent increase in MSE basically that means a based on that elevation has the maximum importance followed by twi followed by landset b3, landset b4 and radiometric potassium similar type of results we can get an increase in node purity, INC node purity means increasing node purity. So, you can also get that elevation is the highest important variable followed by twi, landset b4 radk, landset b3. So, based on this, we can say that elevation is the highest important variable followed by twi, landset b3 and twi and then landset b4 and so on, so forth.

(Refer Slide Time: 16:05)





So, once we have done that, next is doing the internal validation the same just like previous, we are going to do this with the calibration set we are going to first predict based on the calibration set, and then we are going to use this cubist, we are going to use this goof function to predict the results and then we are going to see that R square is 0.78, concordance is 0.8, for MSE 0.06, RMSE 0.26, bias minus 0.00 and then external validation that means original validation with the holdout validation sample.

So, if you can see that, it is also 30 percent. Now, one thing you can see for clear here that here the model is showing somewhat you know, over prediction or you know sorry overfitting So, here you can see R squared value is 0.78 however, for the calibration data set for what the validation data set, we are getting an R square value of 0.30, so that showing some amount of overfitting. So, you should be very very careful by checking these... calibration validation statistics. So, you know, so we have seen how to do how to check the

variable importance and we have also seen how to you know, how to get the model output from the calibration and the validation.

(Refer Slide Time: 17:37)

The screenshot shows the RStudio interface with the following code in the console:

```

254
255 # Internal validation
256 RF.pred.C <- predict(edge.RF.Exp, newdata = DSM_data(training, ))
257 goof(observed = DSM_data$log_cStock0_5(training), predicted = RF.pred.C)
258 # External validation
259 RF.pred.V <- predict(edge.RF.Exp, newdata = DSM_data[-training, ])
260 goof(observed = DSM_data$log_cStock0_5[-training], predicted = RF.pred.V)
261
262 #map
263 map.RF.rl <- predict(covStack, edge.RF.Exp, "cStock_0_5_RF.tif",
264                   format = "GTiff", datatype = "FLT4S", overwrite = TRUE)
265 plot(map.RF.rl,
266      main = "Random Forest model predicted 0-5cm log carbon stocks")
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

```

The Environment pane shows the following objects:

- model.fit1: Named num [1:200] 31.2 31.8 21.6 21.2 16...
- model.fit2: Named num [1:200] 36.7 31.2 27.8 21.3 16...
- Name: "John Doe"
- oc: num [1:335] 1.27 1.29 1.36 1.4 1.49 ...
- pred.hv.MNLR: Factor w/ 12 levels "1","2","3","4",... ..
- pred1: 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...

The console output for `varImpPlot` is as follows:

```

> varImpPlot(edge.RF.Exp)
> RF.pred.C <- predict(edge.RF.Exp, newdata = DSM_data(training, ))
> goof(observed = DSM_data$log_cStock0_5(training), predicted = RF.pred.C)
      R2 concordance      MSE      RMSE      bias
1 0.7863864 0.8441469 0.06957031 0.2637618 -0.004791642
> RF.pred.V <- predict(edge.RF.Exp, newdata = DSM_data[-training, ])
> goof(observed = DSM_data$log_cStock0_5[-training], predicted = RF.pred.V)
      R2 concordance      MSE      RMSE      bias
1 0.3004443 0.4580371 0.1809419 0.4253727 -0.08092012

```

The figure shows two side-by-side plots titled "edge.RF.Exp". The left plot shows the relationship between `%IncMSE` (x-axis, 4 to 16) and several variables (y-axis: elevation, hwi, landsat_b3, landsat_b4, radK). The right plot shows the relationship between `IncNodePurity` (x-axis, 0 to 15) and the same variables (y-axis: elevation, hwi, landsat_b3, landsat_b4, radK).

The screenshot shows the RStudio interface with the following code in the console:

```

254
255 # Internal validation
256 predict(edge.RF.Exp, newdata = DSM_data(training, ))
257 = DSM_data$log_cStock0_5(training), predicted = RF.pred.C)
258 # External validation
259 predict(edge.RF.Exp, newdata = DSM_data[-training, ])
260 = DSM_data$log_cStock0_5[-training], predicted = RF.pred.V)
261
262 #map
263 predict(covStack, edge.RF.Exp, "cStock_0_5_RF.tif",
264         format = "GTiff", datatype = "FLT4S", overwrite = TRUE)
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

```

The Environment pane shows the following objects:

- model.fit1: Named num [1:200] 31.2 31.8 21.6 21.2 16...
- model.fit2: Named num [1:200] 36.7 31.2 27.8 21.3 16...
- Name: "John Doe"
- oc: num [1:335] 1.27 1.29 1.36 1.4 1.49 ...
- pred.hv.MNLR: Factor w/ 12 levels "1","2","3","4",... ..
- pred1: 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...

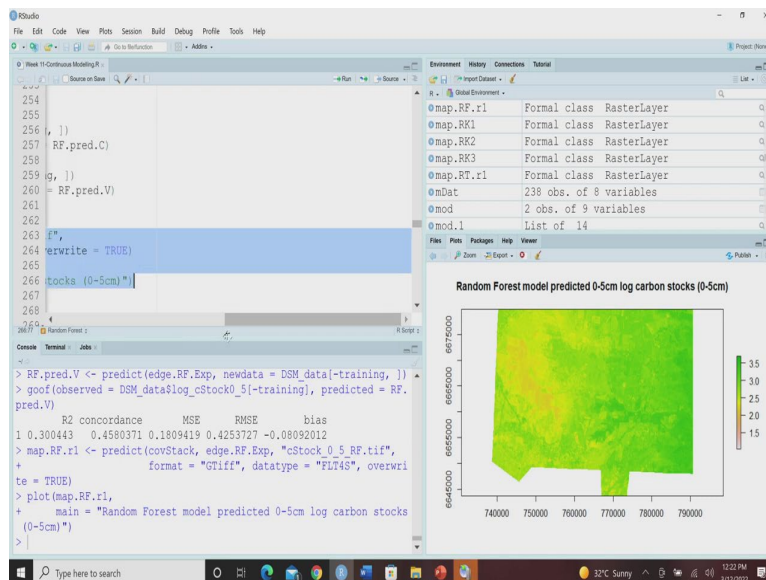
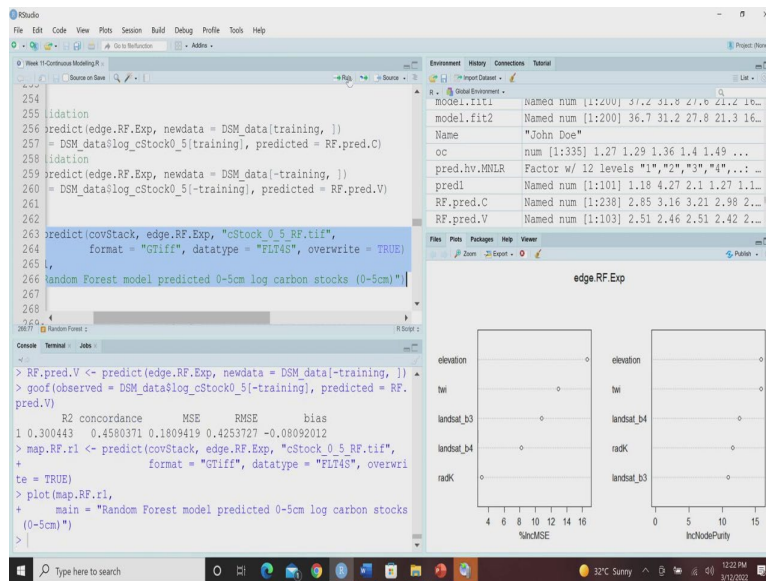
The console output for `varImpPlot` is as follows:

```

> varImpPlot(edge.RF.Exp)
> RF.pred.C <- predict(edge.RF.Exp, newdata = DSM_data(training, ))
> goof(observed = DSM_data$log_cStock0_5(training), predicted = RF.pred.C)
      R2 concordance      MSE      RMSE      bias
1 0.7863864 0.8441469 0.06957031 0.2637618 -0.004791642
> RF.pred.V <- predict(edge.RF.Exp, newdata = DSM_data[-training, ])
> goof(observed = DSM_data$log_cStock0_5[-training], predicted = RF.pred.V)
      R2 concordance      MSE      RMSE      bias
1 0.3004443 0.4580371 0.1809419 0.4253727 -0.08092012

```

The figure shows two side-by-side plots titled "edge.RF.Exp". The left plot shows the relationship between `%IncMSE` (x-axis, 4 to 16) and several variables (y-axis: elevation, hwi, landsat_b3, landsat_b4, radK). The right plot shows the relationship between `IncNodePurity` (x-axis, 0 to 15) and the same variables (y-axis: elevation, hwi, landsat_b3, landsat_b4, radK).



Now, let us see how to produce the map using this using the random forest, just like previously we have also used this call stack function, so call stack function sorry call stack or stack covariate we are going to predict based on the stack covariate using these edge dot rf dot exp model, our format data type all these things are given here. So, let us first predict and then map using and then plot using the plot function.

So, you can see here the plot will be generated momentarily showing the random forest predicted map of 0 to 5-centimetre block carbon stock. So, this is how this map looks like this is the random forest predicted organic carbon map of soil. So, guys this is how you produce the maps using these cubist and Random Forest. Now, next thing is Universal Kriging.

Now, Universal Kriging is basically that it is a kind of a hybrid approach. Now, what is a hybrid approach it is basically combination of both regression as well as Kriging. So, here we do the regression by using the covariates and whatever residuals are there, we are going to Kriging those residuals and then combine the results from the regression as well as from the Kriging interpolation.

So, the final output will be the combination of the model output plus the Kriging interpolation output. Now, this in general this is called regression Kriging we are going to talk about regression Kriging in next week of lectures, but just remember that in this case, if we are very much strict about the model, that is this regression model, which should be linear then it will be called a Universal model problem.

So, Universal Kriging approach, but, if that prediction model is variable, maybe most of the time you will see that nonlinear models are more you know, more appropriate for handling or for predicting the soil properties, then we can tell that, in that case it will be a true regression Kriging but the difference between a regression Kriging and then a Universal Kriging is in case of regression Kriging the prediction model could be any way you know, it could be nonlinear, but in case of Universal Kriging the prediction model will be always linear.

So, I will show you now, how to you know deal with these Universal Kriging in R. So, remember that in case of Universal Kriging, there is a strict requirement of Universal you know Universal Kriging in gstat is that the coordinate reference system of the point data and the covariate must be exactly the same. So, for maintaining, these coordinate, the coordinate reference system for these for the of the point data set as well as the covariate data must be same, so, we need to periodically check this if they are same or not.

(Refer Slide Time: 21:18)

The screenshot shows an RStudio session with the following code in the editor:

```
266 main = "Random forest model predicted U-5cm log carbon sto
267
268
269 ##### UNIVERSAL KRIGING #####
270
271 #strict requirements of universal kriging in gstat is that the
272
273 install.packages("gstat")
274 library(gstat)
275 set.seed(123)
276 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
277 cDat <- DSM_data[training, ]
278 coordinates(cDat) <- x + y
279 crs(cDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS
280 +units=m +no_defs"
281 type=covStack(cDat)
```

The console output shows the following results:

```
pred.V)
R2 concordance MSE RMSE bias
1 0.300443 0.4580371 0.1809419 0.4253727 -0.08092012
> map.RF.r1 <- predict(covStack, edge.RF.Exp, "cStock_0_5_RF.tif",
+ format = "Gtiff", datatype = "FLT4S", overwri
te = TRUE)
> plot(map.RF.r1,
+ main = "Random Forest model predicted 0-5cm log carbon stocks
(0-5cm)")
> library(gstat)
> set.seed(123)
>
```

The Environment pane shows the following objects:

Object	Class	Attributes
map.RF.r1	Formal class RasterLayer	
map.RK1	Formal class RasterLayer	
map.RK2	Formal class RasterLayer	
map.RK3	Formal class RasterLayer	
map.RT.r1	Formal class RasterLayer	
modDat	238 obs. of 8 variables	
mod	2 obs. of 9 variables	
mod.1	List of 14	

The plot shows a map titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)" with a color scale from 15 to 35. The x-axis ranges from 740000 to 770000 and the y-axis from 6640000 to 6670000.

The screenshot shows an RStudio session with the following code in the editor:

```
266 main = "Random forest model predicted U-5cm log carbon sto
267
268
269 ##### UNIVERSAL KRIGING #####
270
271 #strict requirements of universal kriging in gstat is that the
272
273 install.packages("gstat")
274 library(gstat)
275 set.seed(123)
276 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
277 cDat <- DSM_data[training, ]
278 coordinates(cDat) <- x + y
279 crs(cDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS
280 +units=m +no_defs"
281 type=covStack(cDat)
```

The console output shows the following results:

```
pred.V)
R2 concordance MSE RMSE bias
1 0.300443 0.4580371 0.1809419 0.4253727 -0.08092012
> map.RF.r1 <- predict(covStack, edge.RF.Exp, "cStock_0_5_RF.tif",
+ format = "Gtiff", datatype = "FLT4S", overwri
te = TRUE)
> plot(map.RF.r1,
+ main = "Random Forest model predicted 0-5cm log carbon stocks
(0-5cm)")
> library(gstat)
> set.seed(123)
>
```

The Environment pane shows the following objects:

Object	Class	Attributes
map.RF.r1	Formal class RasterLayer	
map.RK1	Formal class RasterLayer	
map.RK2	Formal class RasterLayer	
map.RK3	Formal class RasterLayer	
map.RT.r1	Formal class RasterLayer	
modDat	238 obs. of 8 variables	
mod	2 obs. of 9 variables	
mod.1	List of 14	

The plot shows a map titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)" with a color scale from 15 to 35. The x-axis ranges from 740000 to 770000 and the y-axis from 6640000 to 6670000.

The screenshot shows an RStudio session with the following code in the editor:

```
266 main = "Random forest model predicted U-5cm log carbon sto
267
268
269 ##### UNIVERSAL KRIGING #####
270
271 #strict requirements of universal kriging in gstat is that the
272
273 install.packages("gstat")
274 library(gstat)
275 set.seed(123)
276 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
277 cDat <- DSM_data[training, ]
278 coordinates(cDat) <- x + y
279 crs(cDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS
280 +units=m +no_defs"
281 type=covStack(cDat)
```

The console output shows the following results:

```
pred.V)
R2 concordance MSE RMSE bias
1 0.300443 0.4580371 0.1809419 0.4253727 -0.08092012
> map.RF.r1 <- predict(covStack, edge.RF.Exp, "cStock_0_5_RF.tif",
+ format = "Gtiff", datatype = "FLT4S", overwri
te = TRUE)
> plot(map.RF.r1,
+ main = "Random Forest model predicted 0-5cm log carbon stocks
(0-5cm)")
> library(gstat)
> set.seed(123)
> training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
>
```

The Environment pane shows the following objects:

Object	Class	Attributes
factor w/ 12 levels	factor w/ 12 levels	"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"
pred1	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...
RF.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	[1:238] 99 269 139 299 317 16 177 33...

The plot shows a map titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)" with a color scale from 15 to 35. The x-axis ranges from 740000 to 770000 and the y-axis from 6640000 to 6670000.

So, for this we have to install this gstat package, and then we have to call this library gstat. And then again, setting the seed 123 again just like previously, we are going to select the model and the training model let us call it as a calibration dataset. See that, and then we are going to see the coordinate of the see that so, of course, it will be we can see that we are going to assign these coordinates that utm zone 55 south WGS 84.

(Refer Slide Time: 21:57)

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

```

270
271 #strict requirements of universal kriging in gstat is that the
272
273 install.packages("gstat")
274 library(gstat)
275 set.seed(123)
276 training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
277 cDat <- DSM_data[training, ]
278 coordinates(cDat) <- ~x + y
279 crs(cDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS
280 +units=m +no_defs"
281 crs(covStack) = crs(cDat)
282 # check
283 crs(cDat)
284 crs(covStack)
285
286
287 te = TRUE
288 > plot(maf.RF.r1,
289 + main = "Random Forest model predicted 0-5cm log carbon stocks
290 (0-5cm)")
291 > library(gstat)
292 > set.seed(123)
293 > training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
294 > cDat <- DSM_data[training, ]
295 > coordinates(cDat) <- ~x + y
296 > crs(cDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84
297 +units=m +no_defs"
298 >

```

The Environment pane on the right shows the following objects:

cDat	Formal class SpatialPointsDataFrame
con.mat	num [1:4, 1:4] 5 0 1 2 0 15 0 5 0 1 ...
covStack	Large RasterStack (1154000 elements, 5...
dat	234 obs. of 3 variables
dat1	234 obs. of 3 variables
dat2	101 obs. of 3 variables
DSM_data	341 obs. of 8 variables
edge.cub.Exo	List of 14

The map on the right is titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)" and shows a spatial distribution of values ranging from 15 to 35. A person's face is visible in the bottom right corner of the RStudio window.

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

```

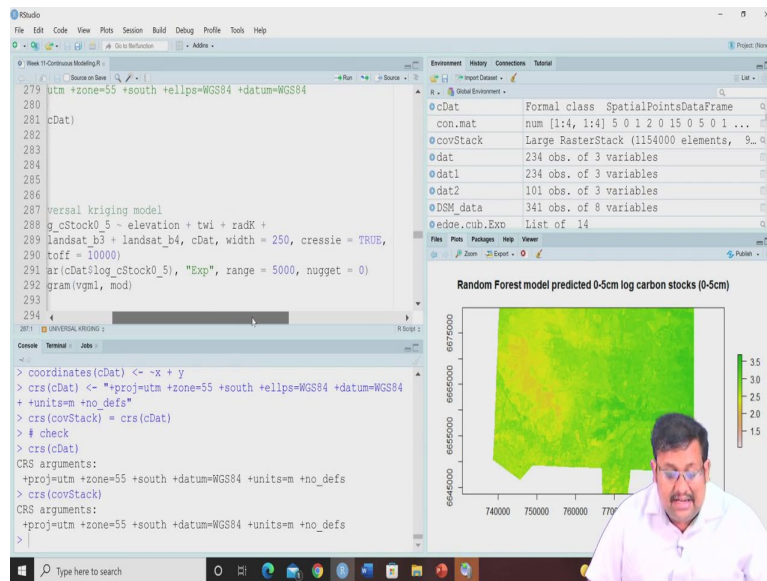
279 crs(cDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS
280 +units=m +no_defs"
281 crs(covStack) = crs(cDat)
282 # check
283 crs(cDat)
284 crs(covStack)
285
286
287 # parameterise the universal kriging model
288 vgm1 <- variogram(log_cStock0_5 ~ elevation + twi + radK +
289 landsat_b3 + landsat_b4, cDat, width = 250,
290 cutoff = 10000)
291 mod <- vgm(psill = var(cDat$log_cStock0_5), "Exp", range = 5000)
292 model_1 <- fit.variogram(vgm1, mod)
293 model_1
294
295
296 > coordinates(cDat) <- ~x + y
297 > crs(cDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84
298 +units=m +no_defs"
299 > crs(covStack) = crs(cDat)
300 # check
301 > crs(cDat)
302 CRS arguments:
303 +proj=utm +zone=55 +south +datum=WGS84 +units=m +no_defs
304 > crs(covStack)
305 CRS arguments:
306 +proj=utm +zone=55 +south +datum=WGS84 +units=m +no_defs
307 >

```

The Environment pane on the right shows the following objects:

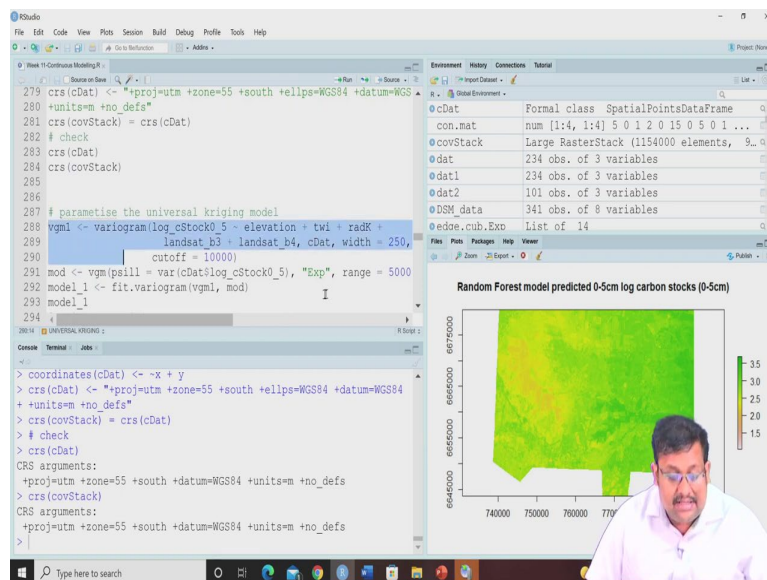
cDat	Formal class SpatialPointsDataFrame
con.mat	num [1:4, 1:4] 5 0 1 2 0 15 0 5 0 1 ...
covStack	Large RasterStack (1154000 elements, 5...
dat	234 obs. of 3 variables
dat1	234 obs. of 3 variables
dat2	101 obs. of 3 variables
DSM_data	341 obs. of 8 variables
edge.cub.Exo	List of 14

The map on the right is titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)" and shows a spatial distribution of values ranging from 15 to 35. A person's face is visible in the bottom right corner of the RStudio window.



So, let us see whether our coordinate reference system of this calibration data set matches well with the coordinate reference system of the of the stack covariate or not. So, if we give this command we will cDat yes, they are both same. So, if we check further the coordinate reference system of cDat and then coordinate reference system of covStack we will see that in both the condition there will be zone 55 South datum a WGS 84 in both cases.

(Refer Slide Time: 22:30)



Now, next thing is to parameterize the Universal Kriging model. So, first we have to fit the variogram we have already seen how to fit the variogram so we are going to use these variogram function. Our target is logged in... log converted carbonStocks 0 to 5 centimetre our variables are elevation twi, radk then landsat b3, landsat b4 our model is calibration data, our data set its calibration data, width is again default 250 cressie variogram cutoff values is 10,000 and then here, we are again going to fix, the fit the model using an exponential model, and then let us see how this variogram parameters will come.

So, here you can see negative 0 which is almost... which is ideal, and then here are the for a exponential model, we are gating the partial sill. So, the total sill will also be 0.16 and range parameter is 233. So, from there we can have any idea about the variogram parameters. And based on these variogram parameters, we are going to feed the Universal Kriging model.

(Refer Slide Time: 23:46)

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

```

287 # parameterize the universal kriging model
288 vgm1 <- variogram(log_cStock0_5 ~ elevation + twi + radK +
289   landsat_b3 + landsat_b4, cDat, width = 250,
290   cutoff = 10000)
291 mod <- vgm(psill = var(cDat$log_cStock0_5), "Exp", range = 5000
292 model_1 <- fit.variogram(vgm1, mod)
293 model_1
294 # Universal kriging model
295 gUK <- gstat(NULL, "log.carbon", log_cStock0_5 ~ elevation + twi
296   landsat_b3 + landsat_b4, cDat, model = model_1)
297 |
298 |
299 #validation
300 vDat <- DSM_data[-training_1]
301 |

```

The console output shows the results of the variogram fitting:

```

> vgm1 <- variogram(log_cStock0_5 ~ elevation + twi + radK +
+   landsat_b3 + landsat_b4, cDat, width = 250, cr
+   cressie = TRUE,
+   cutoff = 10000)
> mod <- vgm(psill = var(cDat$log_cStock0_5), "Exp", range = 5000, n
+   ugget = 0)
> model_1 <- fit.variogram(vgm1, mod)
> model_1
model    psill  range
1  Nug 0.0000000  0.000
2  Exp 0.1611552 233.794
> |

```

The Environment pane on the right lists the objects in the workspace:

- gUK: Large RasterLayer (230800 elements, 1...)
- UK.F.map: Large RasterLayer (230800 elements, 1...)
- UK.preds.V: 103 obs. of 4 variables
- UK.F.var.map: Large RasterLayer (230800 elements, 1...)
- USYD_soil1: 166 obs. of 16 variables
- vats: 230800 obs. of 5 variables
- vDat: Formal class 'SpatialPointsDataFrame'
- vgm1: 40 obs. of 6 variables

The Plots pane shows a map titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)". The map displays a spatial distribution of predicted values, with a color scale ranging from 1.5 (green) to 3.5 (red). The map axes are labeled with coordinates: X-axis (740000 to 770000) and Y-axis (6640000 to 6675000).

```

286
287 # Universal kriging model
288 ram(log_cStock0_5 ~ elevation + twi + radK +
289     landsat_b3 + landsat_b4, cDat, width = 250, cressie = TRUE,
290     cutoff = 10000)
291 ll = var(cDat$log_cStock0_5, "Exp", range = 5000, nugget = 0)
292 vvariogram(vgml, mod)
293
294 # Fitting model
295 LL, "log.carbon", log_cStock0_5 ~ elevation + twi + radK +
296     landsat_b3 + landsat_b4, cDat, model = model_1)
297
298
299
300 # Training
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

```

```

291 mod <- vgm(psill = var(cDat$log_cStock0_5), "Exp", range = 5000)
292 model_1 <- fit.variogram(vgml, mod)
293 model_1
294 # Universal kriging model
295 gUK <- gstat(NULL, "log.carbon", log_cStock0_5 ~ elevation + twi +
296     landsat_b3 + landsat_b4, cDat, model = model_1)
297
298
299 # validation
300 vDat <- DSM_data[-training, ]
301 coordinates(vDat) <- ~x + y
302 crs(vDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84
303     +units=m +no_defs"
304 crs(vDat)
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

```

So, for fitting the Universal Kriging model, the function is gstat function we are going to use our target is a natural log converted carbonStock, our predictors are elevation twi, radk, landsat b3, landsat b4 our data set is calibration data set our model is model 1 which we have just fitted using the exponential function.

So, once we have this fitted this model, let us validate so validation data set is vDat again, minus training samples coordinates again we are x we are, instructing R that these x and y you should understand these add the coordinates. So, we are assigning these coordinates same just like previously utm zone beautified south and from there WGS 84 and then we want to see the coordinate reference system, so you will see that utm zone 35 south WGS 84.

(Refer Slide Time: 24:45)

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

```
299 #validation
300 vDat <- DSM_data[-training, ]
301 coordinates(vDat) <- ~x + y
302 crs(vDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84
303 +units=m +no_defs"
304 crs(vDat)
305
306
307 # make the predictions
308 UK.preds.V <- as.data.frame(krige(log_cStock0_5 ~ elevation + t
309                               + radK + landsat_b3 + landsat_b4,
310                               newdata = vDat))
311 ## [using universal kriging]
312 goof(observed = DSM_data$log_cStock0_5[-training],
313      predicted = UK.preds.V[,3])
314
```

The console shows the execution of the `gUK` function:

```
> gUK <- gstat(NULL, "log.carbon", log_cStock0_5 ~ elevation + twi +
+ radK +
+ landsat_b3 + landsat_b4, cDat, model = model_1)
> #validation
> vDat <- DSM_data[-training, ]
> coordinates(vDat) <- ~x + y
> crs(vDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84
+ units=m +no_defs"
> crs(vDat)
CRS arguments:
+proj=utm +zone=55 +south +datum=WGS84 +units=m +no_defs
>
```

The Environment pane on the right lists the following objects:

- `trainData`: 306 obs. of 14 variables
- `twi`: Large RasterLayer (230800 elements, 1...)
- `UK.F.map`: Large RasterLayer (230800 elements, 1...)
- `UK.preds.V`: 103 obs. of 4 variables
- `UK.F.var.map`: Large RasterLayer (230800 elements, 1...)
- `USYD_soil1`: 166 obs. of 16 variables
- `oVals`: 230800 obs. of 5 variables
- `vDat`: Formal class SpatialPointsDataFrame

The Plots pane shows a map titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)" with a color scale from 15 to 35. A small video inset of the presenter is visible in the bottom right corner.

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

```
299 #validation
300 -training, ]
301 ~x + y
302 proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84
303 "
304
305
306
307 # make the predictions
308 data.frame(krige(log_cStock0_5 ~ elevation + twi
309               + radK + landsat_b3 + landsat_b4, cDat, model
310               newdata = vDat))
311 ## [using universal kriging]
312 DSM_data$log_cStock0_5[-training],
313 UK.preds.V[,3])
314
```

The console shows the execution of the `gUK` function:

```
> gUK <- gstat(NULL, "log.carbon", log_cStock0_5 ~ elevation + twi +
+ radK +
+ landsat_b3 + landsat_b4, cDat, model = model_1)
> #validation
> vDat <- DSM_data[-training, ]
> coordinates(vDat) <- ~x + y
> crs(vDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84
+ units=m +no_defs"
> crs(vDat)
CRS arguments:
+proj=utm +zone=55 +south +datum=WGS84 +units=m +no_defs
>
```

The Environment pane on the right lists the following objects:

- `trainData`: 306 obs. of 14 variables
- `twi`: Large RasterLayer (230800 elements, 1...)
- `UK.F.map`: Large RasterLayer (230800 elements, 1...)
- `UK.preds.V`: 103 obs. of 4 variables
- `UK.F.var.map`: Large RasterLayer (230800 elements, 1...)
- `USYD_soil1`: 166 obs. of 16 variables
- `oVals`: 230800 obs. of 5 variables
- `vDat`: Formal class SpatialPointsDataFrame

The Plots pane shows a map titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)" with a color scale from 15 to 35. A small video inset of the presenter is visible in the bottom right corner.

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

```
302 crs(vDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84
303 +units=m +no_defs"
304 crs(vDat)
305
306
307 # make the predictions
308 UK.preds.V <- as.data.frame(krige(log_cStock0_5 ~ elevation + t
309                               + radK + landsat_b3 + landsat_b4,
310                               newdata = vDat))
311 ## [using universal kriging]
312 goof(observed = DSM_data$log_cStock0_5[-training],
313      predicted = UK.preds.V[,3])
314
315 #map interpolation and prediction variance
316 par(mfrow = c(1, 2))
317
```

The console shows the execution of the `gUK` function:

```
> coordinates(vDat) <- ~x + y
> crs(vDat) <- "+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84
+ units=m +no_defs"
> crs(vDat)
CRS arguments:
+proj=utm +zone=55 +south +datum=WGS84 +units=m +no_defs
> UK.preds.V <- as.data.frame(krige(log_cStock0_5 ~ elevation + twi
+ radK + landsat_b3 + landsat_b4, cDat, model = model_1,
+ newdata = vDat))
[using universal kriging]
>
```

The Environment pane on the right lists the following objects:

- `trainData`: 306 obs. of 14 variables
- `twi`: Large RasterLayer (230800 elements, 1...)
- `UK.F.map`: Large RasterLayer (230800 elements, 1...)
- `UK.preds.V`: 103 obs. of 4 variables
- `UK.F.var.map`: Large RasterLayer (230800 elements, 1...)
- `USYD_soil1`: 166 obs. of 16 variables
- `oVals`: 230800 obs. of 5 variables
- `vDat`: Formal class SpatialPointsDataFrame

The Plots pane shows a map titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)" with a color scale from 15 to 35. A small video inset of the presenter is visible in the bottom right corner.

And now we can predict based on the predict this validation data based on our Universal Kriging model which have already fitted, so we are going to you know, predict the validation data set based on the previously fitted model.

(Refer Slide Time: 25:13)

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

```
309 + radK + landsat_b3 + landsat
310 newdata = vDat))
311 ## [using universal kriging]
312 goof(observed = DSM_data$log_cStock0_5[-training],
313 predicted = UK.preds.V[,3])
314
315 #map interpolation and prediction variance
316 par(mfrow = c(1, 2))
317 # predictions
318 UK.P.map <- interpolate(covStack, gUK, xyOnly = FALSE, index =
319 plot(UK.P.map, main = "Universal kriging predictions")
320 # prediction variance
321 UK.Pvar.map <- interpolate(covStack, gUK, xyOnly = FALSE, index
322 plot(UK.Pvar.map, main = "Universal kriging prediction variance"
323
324
```

The Environment pane on the right lists the following objects:

- traindata: 306 obs. of 14 variables
- twi: Large RasterLayer (230800 elements, 1...
- UK.P.map: Large RasterLayer (230800 elements, 1...
- UK.preds.V: 103 obs. of 4 variables
- UK.Pvar.map: Large RasterLayer (230800 elements, 1...
- USYD_soil1: 166 obs. of 16 variables
- vvals: 230800 obs. of 5 variables
- vDat: Formal class 'SpatialPointsDataFrame'

The plot window displays a map titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)". The map shows a spatial distribution of carbon stocks, with a color scale ranging from 1.5 (dark green) to 3.5 (light green). The plot axes are labeled with coordinates: x-axis from 740000 to 770000 and y-axis from 6645000 to 6675000.

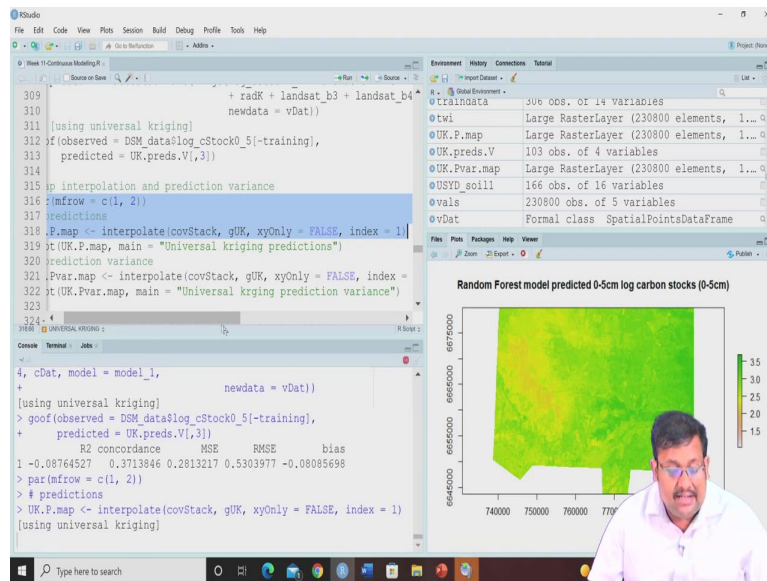
The screenshot shows the RStudio interface. The console window contains the following R code:

```
309 + radK + landsat_b3 + landsat
310 newdata = vDat))
311 ## [using universal kriging]
312 goof(observed = DSM_data$log_cStock0_5[-training],
313 predicted = UK.preds.V[,3])
314
315 #map interpolation and prediction variance
316 par(mfrow = c(1, 2))
317 # predictions
318 UK.P.map <- interpolate(covStack, gUK, xyOnly = FALSE, index =
319 plot(UK.P.map, main = "Universal kriging predictions")
320 # prediction variance
321 UK.Pvar.map <- interpolate(covStack, gUK, xyOnly = FALSE, index
322 plot(UK.Pvar.map, main = "Universal kriging prediction variance"
323
324
```

The Environment pane on the right lists the following objects:

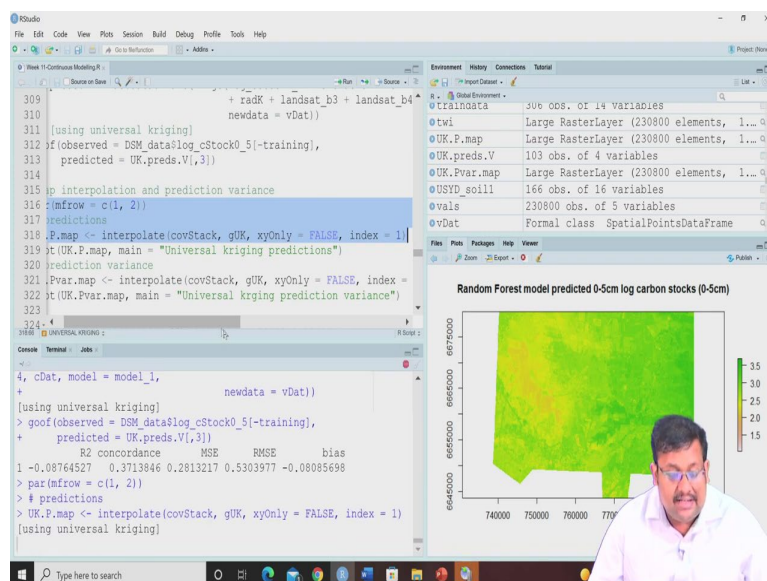
- traindata: 306 obs. of 14 variables
- twi: Large RasterLayer (230800 elements, 1...
- UK.P.map: Large RasterLayer (230800 elements, 1...
- UK.preds.V: 103 obs. of 4 variables
- UK.Pvar.map: Large RasterLayer (230800 elements, 1...
- USYD_soil1: 166 obs. of 16 variables
- vvals: 230800 obs. of 5 variables
- vDat: Formal class 'SpatialPointsDataFrame'

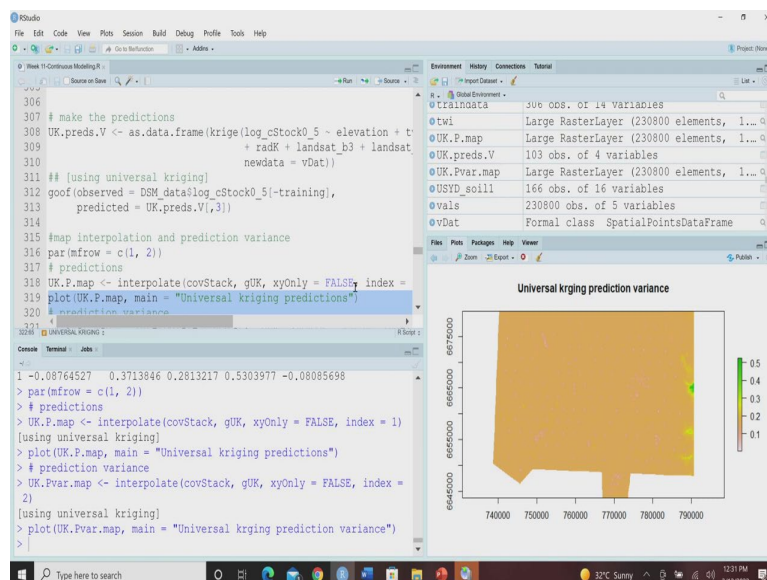
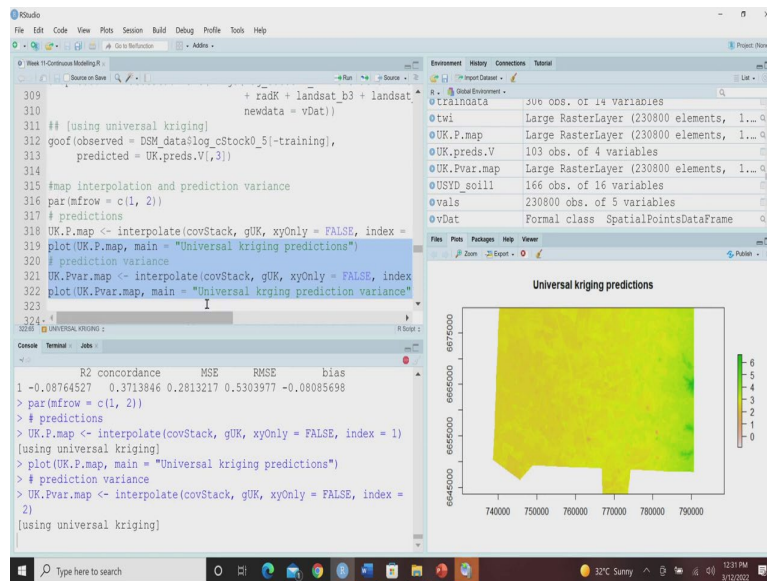
The plot window displays a map titled "Random Forest model predicted 0-5cm log carbon stocks (0-5cm)". The map shows a spatial distribution of carbon stocks, with a color scale ranging from 1.5 (dark green) to 3.5 (light green). The plot axes are labeled with coordinates: x-axis from 740000 to 770000 and y-axis from 6645000 to 6675000.



Now, we are going to see the goodness of fit statistics of the observed as well as predicted results. So, you can see that here the goodness of fit statistics is not that high we are seeing some extrapolation. So, that is why we are getting the negative R square negative R square value, but, you know, you will have it varies from data set to data set in some data set you will may have some good R squared values in this just in this case, we are not getting very good result that shows the prior the linear model may not be the ideal model to handle this dataset. So, once we have these now, we can map the interpolation and prediction variance. So, first we want to predict based on this Universal Kriging model, and once we do that we can produce.

(Refer Slide Time: 26:13)

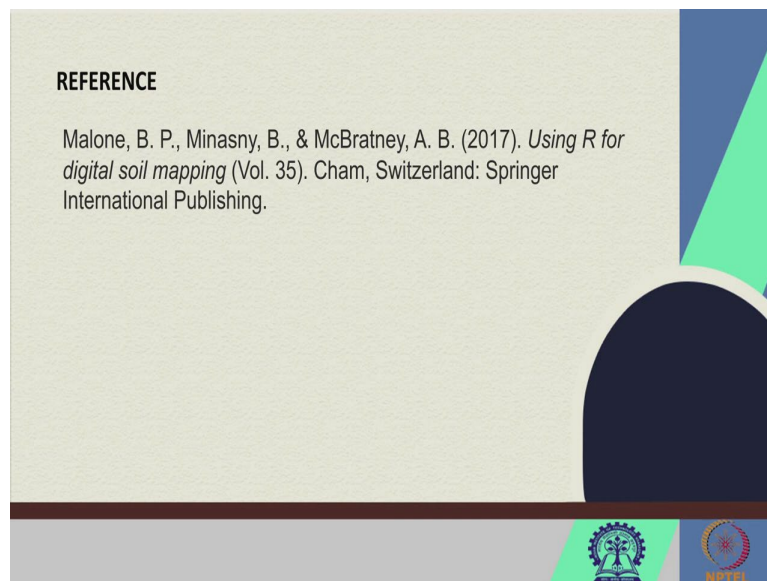




So, let us first remove this plot so, that, so first you have to predict or interpolate based on these covStack and then you are using these gUK or Universal Kriging and so here you are giving the index 1 for producing the predictions, you can also produce the prediction variance by using the same by changing only the index and you can see the both the plots. So, let us first run all these and we will see their plot together.

So, this is the Universal Kriging predictions and it will do some calculation and this is the Universal Kriging prediction variance. So, not only you will get the prediction result, but at the same time you will get the prediction variance also. So, this is how you use the Universal Kriging to produce the hybrid results.

(Refer Slide Time: 27:39)



So, guys, I hope that the things are you have got some good knowledge from this lecture and you will be now able to run these codes and grow your confidence on these type of approaches, where we have already seen the cubist which is an important machine learning approach we have already seen the random forests which is an advanced machine learning approach and we have also seen an hybrid approach called Universal Kriging.

So, let us wrap up our lecture here. And let us wrap up our week 11 and in week 12 we will be discussing the Regression Kriging which is the most advanced, one of the advanced hybrid approach for digital soil mapping. And then we will be also discussing how to deal with the categorical models using R so please stay tuned and let us meet in the lectures of week 12. Thank you.