# Machine Learning for Soil and Crop Management

**Professor Somsubhra Chakraborty**
**Agricultural and Food Engineering Department**
**Indian Institute of Technology, Kharagpur**

**Lecture 52**

**Digital Soil Mapping with Continuous Variables (Contd.)**

(Refer Slide Time: 0:22)

NPTEL ONLINE CERTIFICATION COURSES

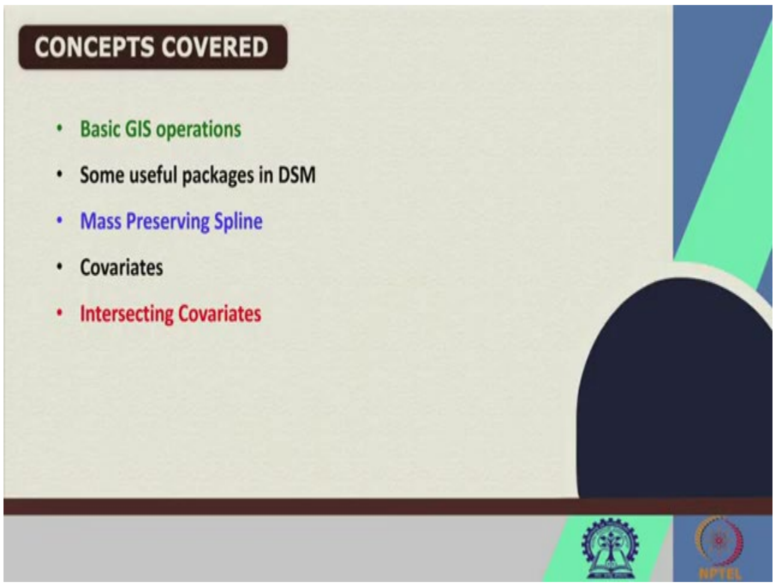
Machine Learning for Soil and Crop Management
Prof. Somsubhra Chakraborty
Agricultural and Food Engineering Department
Indian Institute of Technology Kharagpur

Week 11: DIGITAL SOIL MAPPING WITH CONTINUOUS VARIABLES

LECTURE 52

Welcome friends to this second lecture of week 11 of this online NPTEL certification course of Machine Learning for Soil and Crop Management. And in this week we are discussing about how to model and map the continuous variable in the digital soil mapping domain.

(Refer Slide Time: 0:53)

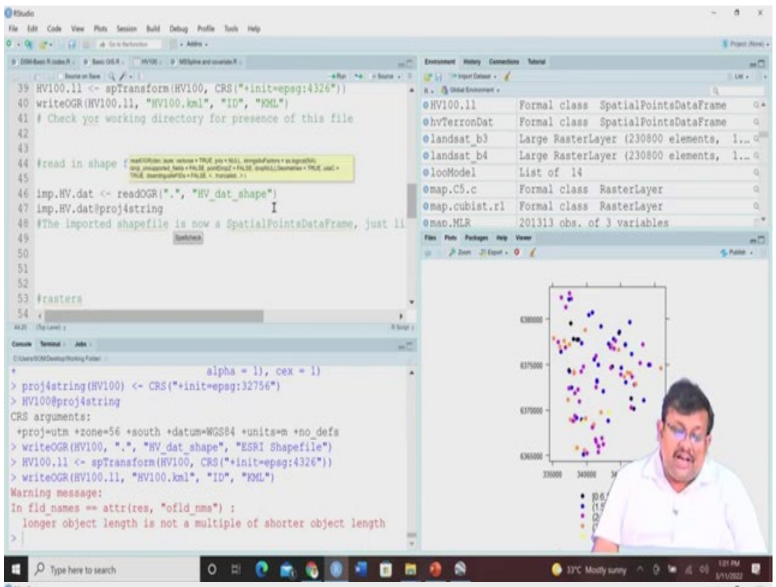


CONCEPTS COVERED

- Basic GIS operations
- Some useful packages in DSM
- Mass Preserving Spline
- Covariates
- Intersecting Covariates

So, in my first lecture of this week we have discussed some basic GIS operation, some codes are remaining and I am going to discuss this first and then in this lecture we are also going to discuss some of the useful packages for digital soil mapping and then we are going to start a very important topic that is mass preserving spline and also we are going to discuss if time permits for the covariates and also how to intersect the covariates for subsequent DSM modelling.

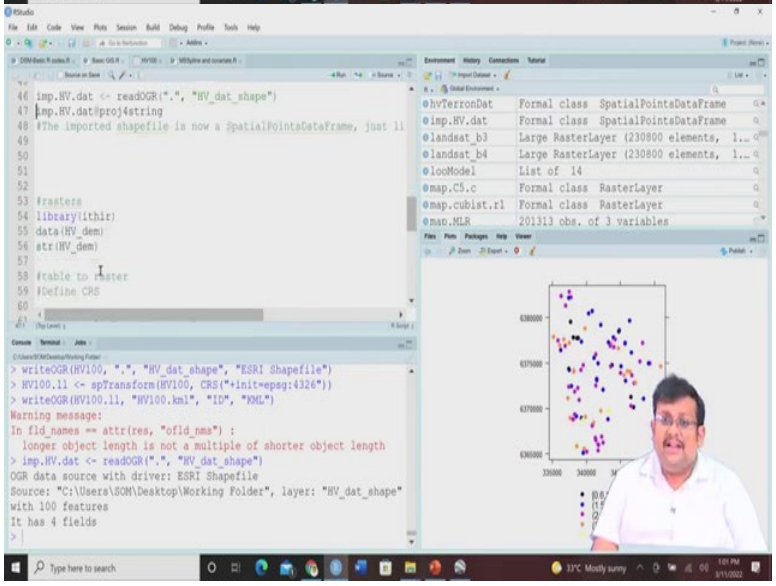
(Refer Slide Time: 1:19)



```
HV100.ll <- spTransform(HV100, CRS("+init=epsg:4326"))
writeOGR(HV100.ll, "HV100.kml", "ID", "KML")
# Check yor working directory for presence of this file


#read in shape f

imp.HV.dat <- readOGR(".", "HV_dat_shape")
imp.HV.dat@proj4string
#The imported shapefile is now a SpatialPointsDataFrame, just li





#rasters
```

```
+                                alpha = 1), cex = 1)
> proj4string(HV100) <- CRS("+init=epsg:32756")
> HV100@proj4string
CRS arguments:
 +proj=utm +zone=56 +south +datum=WGS84 +units=m +no_defs
> writeOGR(HV100, ".", "HV_dat_shape", "ESRI Shapefile")
> HV100.ll <- spTransform(HV100, CRS("+init=epsg:4326"))
> writeOGR(HV100.ll, "HV100.kml", "ID", "KML")
Warning message:
In fld_names == attr(res, "ofld_nms") :
  longer object length is not a multiple of shorter object length
>
```

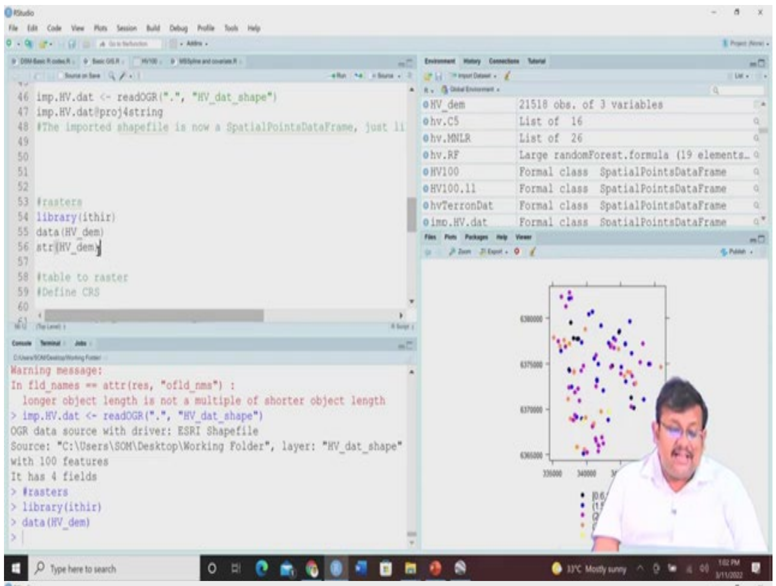
```
HV100.ll       Formal class  SpatialPointsDataFrame
hvTerronDat    Formal class  SpatialPointsDataFrame
landsat_b3     Large RasterLayer (230800 elements,  1...
landsat_b4     Large RasterLayer (230800 elements,  1...
looModel       List of  14
map.C5.c       Formal class  RasterLayer
map.cubist.rl  Formal class  RasterLayer
map.MLR        201313 obs. of 3 variables
```



```
imp.HV.dat <- readOGR(".", "HV_dat_shape")
imp.HV.dat@proj4string
#The imported shapefile is now a SpatialPointsDataFrame, just li




#rasters
library(ithir)
data(HV_dem)
str(HV_dem)

#table to raster
#Define CRS
```

```
> writeOGR(HV100, ".", "HV_dat_shape", "ESRI Shapefile")
> HV100.ll <- spTransform(HV100, CRS("+init=epsg:4326"))
> writeOGR(HV100.ll, "HV100.kml", "ID", "KML")
Warning message:
In fld_names == attr(res, "ofld_nms") :
  longer object length is not a multiple of shorter object length
> imp.HV.dat <- readOGR(".", "HV_dat_shape")
OGR data source with driver: ESRI Shapefile
Source: "C:\Users\SOM\Desktop\Working Folder", layer: "HV_dat_shape"
with 100 features
It has 4 fields
>
```

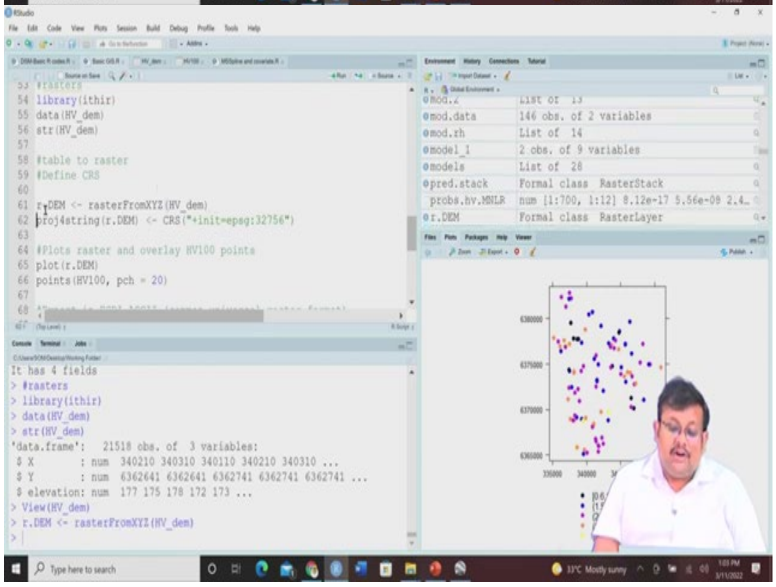
```
hvTerronDat    Formal class  SpatialPointsDataFrame
imp.HV.dat     Formal class  SpatialPointsDataFrame
landsat_b3     Large RasterLayer (230800 elements,  1...
landsat_b4     Large RasterLayer (230800 elements,  1...
looModel       List of  14
map.C5.c       Formal class  RasterLayer
map.cubist.rl  Formal class  RasterLayer
map.MLR        201313 obs. of 3 variables
```



```r
imp.HV.dat <- readOGR(".", "HV_dat_shape")
imp.HV.dat@proj4string
#The imported shapefile is now a SpatialPointsDataFrame, just li




#rasters
library(ithir)
data(HV_dem)
str(HV_dem)

#table to raster
#Define CRS

```

```
Warning message:
In fld_names == attr(res, "ofld_nms") :
  longer object length is not a multiple of shorter object length
> imp.HV.dat <- readOGR(".", "HV_dat_shape")
OGR data source with driver: ESRI Shapefile
Source: "C:\Users\SOM\Desktop\Working Folder", layer: "HV_dat_shape"
with 100 features
It has 4 fields
> #rasters
> library(ithir)
> data(HV_dem)
> |
```

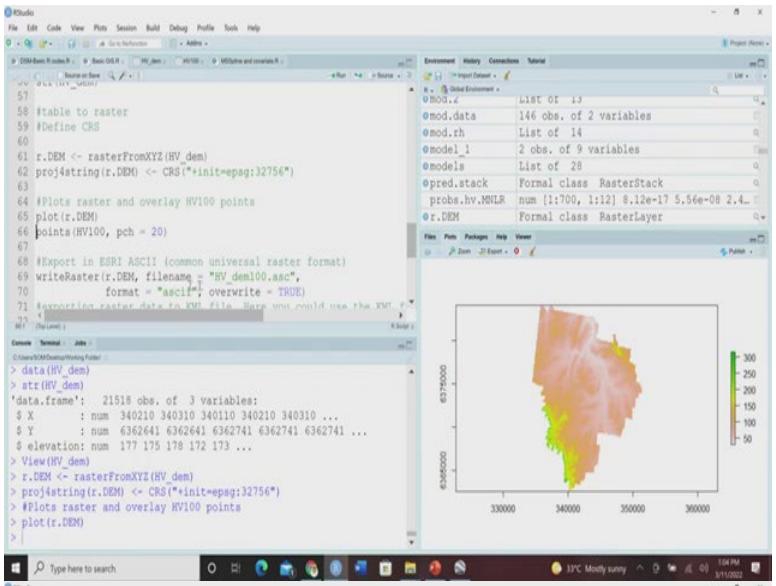
```
HV_dem        21518 obs. of 3 variables
hv.C5         List of  16
hv.MNLR       List of  26
hv.RF         Large randomForest.formula (19 elements…
HV100         Formal class  SpatialPointsDataFrame
HV100.11      Formal class  SpatialPointsDataFrame
hvTerronDat   Formal class  SpatialPointsDataFrame
imp.HV.dat    Formal class  SpatialPointsDataFrame
```



```r
#rasters
library(ithir)
data(HV_dem)
str(HV_dem)

#table to raster
#Define CRS

r.DEM <- rasterFromXYZ(HV_dem)
proj4string(r.DEM) <- CRS("+init=epsg:32756")

#Plots raster and overlay HV100 points
plot(r.DEM)
points(HV100, pch = 20)

```

```
It has 4 fields
> #rasters
> library(ithir)
> data(HV_dem)
> str(HV_dem)
'data.frame':   21518 obs. of  3 variables:
 $ X        : num  340210 340310 340110 340210 340310 ...
 $ Y        : num  6362641 6362641 6362741 6362741 6362741 ...
 $ elevation: num  177 175 178 172 173 ...
> View(HV_dem)
> r.DEM <- rasterFromXYZ(HV_dem)
> |
```

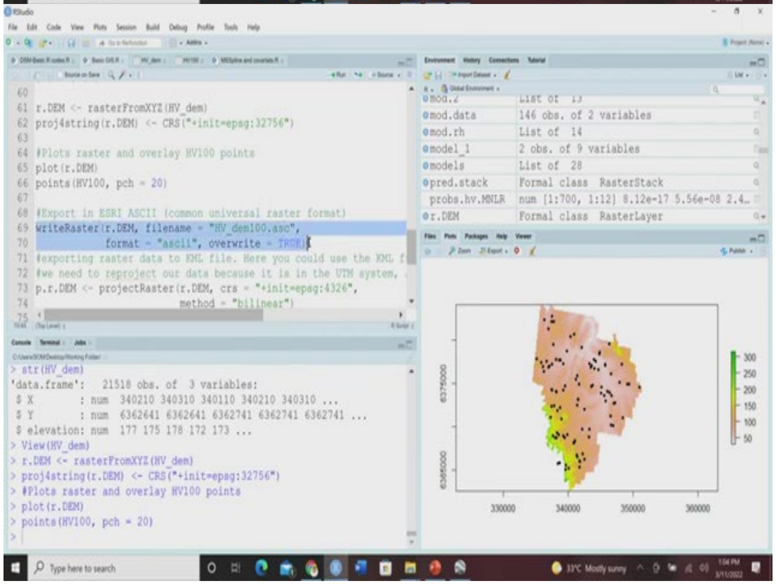
```
mod.2            List of  13
mod.data         146 obs. of 2 variables
mod.rh           List of  14
model_1          2 obs. of 9 variables
models           List of  28
pred.stack       Formal class  RasterStack
probs.hv.MNLR    num [1:700, 1:12] 8.12e-17 5.56e-09 2.4…
r.DEM            Formal class  RasterLayer
```



```
#table to raster
#Define CRS

r.DEM <- rasterFromXYZ(HV_dem)
proj4string(r.DEM) <- CRS("+init=epsg:32756")

#Plots raster and overlay HV100 points
plot(r.DEM)
points(HV100, pch = 20)

#Export in ESRI ASCII (common universal raster format)
writeRaster(r.DEM, filename = "HV_dem100.asc",
            format = "ascii", overwrite = TRUE)
```

```
mod.data      146 obs. of 2 variables
mod.rh        List of 14
model_1       2 obs. of 9 variables
models        List of 28
pred.stack    Formal class RasterStack
probs.hv.MNLR num [1:700, 1:12] 8.12e-17 5.56e-08 2.4...
r.DEM         Formal class RasterLayer
```

```
> data(HV_dem)
> str(HV_dem)
'data.frame':   21518 obs. of  3 variables:
 $ X        : num  340210 340310 340110 340210 340310 ...
 $ Y        : num  6362641 6362641 6362741 6362741 6362741 ...
 $ elevation: num  177 175 178 172 173 ...
> View(HV_dem)
> r.DEM <- rasterFromXYZ(HV_dem)
> proj4string(r.DEM) <- CRS("+init=epsg:32756")
> #Plots raster and overlay HV100 points
> plot(r.DEM)
>
```



```
r.DEM <- rasterFromXYZ(HV_dem)
proj4string(r.DEM) <- CRS("+init=epsg:32756")

#Plots raster and overlay HV100 points
plot(r.DEM)
points(HV100, pch = 20)

#Export in ESRI ASCII (common universal raster format)
writeRaster(r.DEM, filename = "HV_dem100.asc",
            format = "ascii", overwrite = TRUE)
#exporting raster data to KML file. Here you could use the KML f
#we need to reproject our data because it is in the UTM system,
p.r.DEM <- projectRaster(r.DEM, crs = "+init=epsg:4326",
                         method = "bilinear")
```

```
> str(HV_dem)
'data.frame':   21518 obs. of  3 variables:
 $ X        : num  340210 340310 340110 340210 340310 ...
 $ Y        : num  6362641 6362641 6362741 6362741 6362741 ...
 $ elevation: num  177 175 178 172 173 ...
> View(HV_dem)
> r.DEM <- rasterFromXYZ(HV_dem)
> proj4string(r.DEM) <- CRS("+init=epsg:32756")
> #Plots raster and overlay HV100 points
> plot(r.DEM)
> points(HV100, pch = 20)
>
```

So, let us just go back to our previous course which we have ran in R. So, I have showed you that how to use R for producing the, how to download the data and then how to download and install packages, how to call those libraries and we have seen how to create the, how to change the normal data frame to spatial points data frame and then we have seen how to convert that spatial points data frame and give some, give the coordinate reference system, proper coordinate reference system using the EPSG code and how this one map can be reproject in the Google Earth by, we have already seen.

Now, if you want to, now next thing is we have seen that how these shapefiles are being created, ESRI shapefile are being created. So, if you want to read any shapefile so the function is readOGR and then you can give the name of the shapefile, so you can see that this is shapefile which is there in the working folder with 100 features and 4 fields. So, the important shapefile is now in the spatial points data frame. So, just like this HV 100.

Now, we have seen with this vector data, we have seen different operation with the vector data so far. Let us see how to deal with the different raster data. So, again let us call this library ithir and then data. So, here as raster we are going to use the dem file for that hunted valley region.

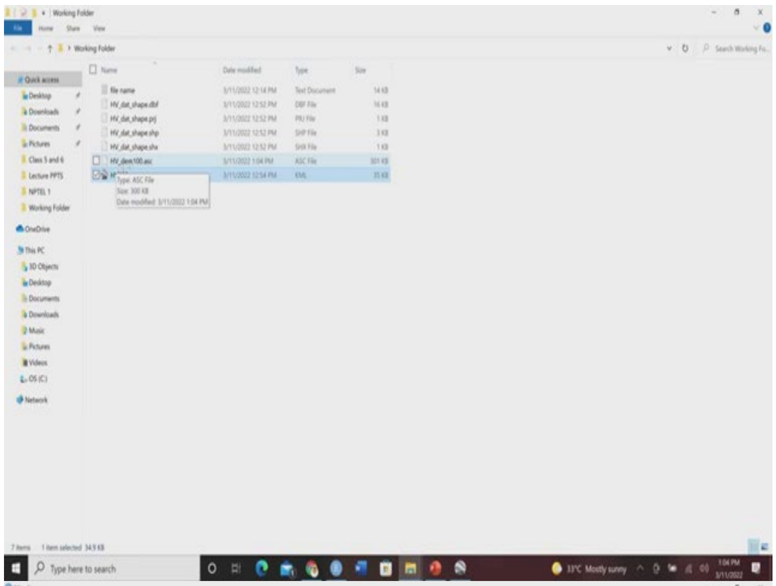
So, we are going to use, we are going to call this data HV underscore dem which is there in the ithir package and let us see the structure of the dem. Structure of the dem is very simple any type of raster file by the way their format is very simple, so x y and elevation so you can see here if you click on this HV underscore dem 21518 observation of 3 variables. So, you can see x and y are the coordinates and the third variable is the elevation. So, using this you can understand that this is only showing the surface roughness of the earth surface.

So, we have seen the structure, now if we want to convert, if you want to plot it again, right now it is in simple tabular format, but we have to instruct R that you should convert this table into raster. So, for that the function is raster from x y z, so here the original x y z will be converted into raster followed by this name. So, we are giving the name as R dot dem.

Now, if we want to just like the vector file here also we are going to assign the coordinate reference system using the EPSG code. So, we have assigned this now if you want to plot raster and then overlay these HV 100 points you will see that plot, we are going to first plot this R dot dem, so you can see this is the plot of this dem file and then we are going to overlay this 100 soil data with the size of the character 20. So, you can see here these are the soil samples and which are overlaid over this dem file.

Now, you can export in ESRI ascii file which is the common universal raster format, so for that you can use this write raster function and here we are having this rdem file name, the new file name will be HV underscore dem 100 dot aac, format will be ascii file. So, we can create this new ascii file.

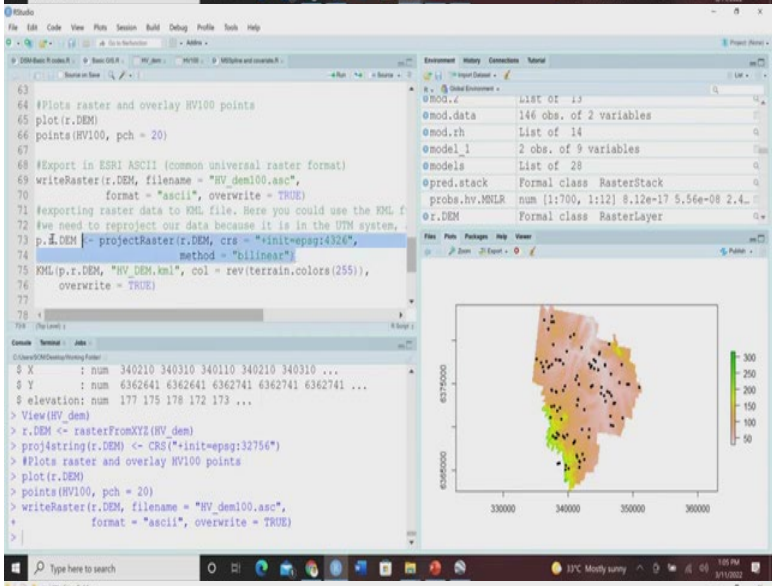
(Refer Slide Time: 6:00)



Working Folder

| Name | Date modified | Type | Size |
|---|---|---|---|
| file name | 3/11/2022 12:14 PM | Text Document | 14 KB |
| HV_dat_shape.dbf | 3/11/2022 12:52 PM | DBF File | 16 KB |
| HV_dat_shape.prj | 3/11/2022 12:52 PM | PRJ File | 1 KB |
| HV_dat_shape.shp | 3/11/2022 12:52 PM | SHP File | 3 KB |
| HV_dat_shape.shx | 3/11/2022 12:52 PM | SHX File | 1 KB |
| HV_dem100.asc | 3/11/2022 1:04 PM | ASC File | 301 KB |
| H[illegible] | 3/11/2022 12:54 PM | KML | 35 KB |

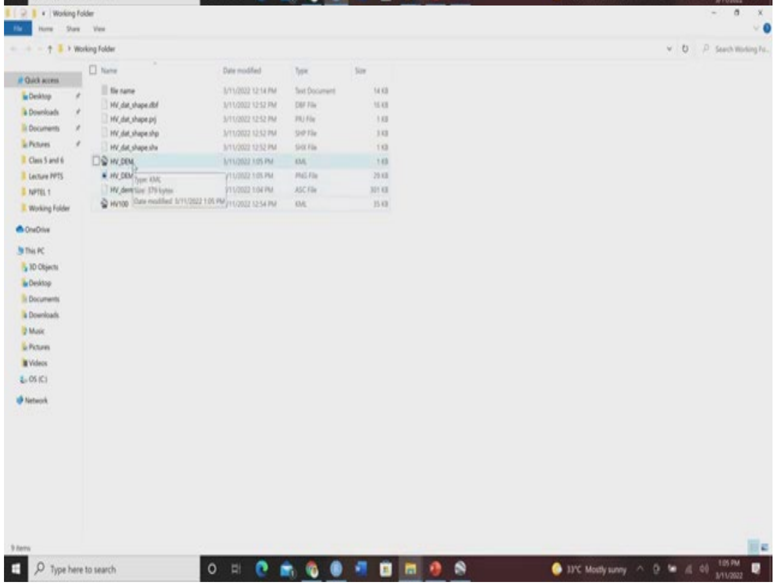
Type: ASC File
Size: 300 KB
Date modified: 3/11/2022 1:04 PM



```
#Plots raster and overlay HV100 points
plot(r.DEM)
points(HV100, pch = 20)

#Export in ESRI ASCII (common universal raster format)
writeRaster(r.DEM, filename = "HV_dem100.asc",
            format = "ascii", overwrite = TRUE)
#exporting raster data to KML file. Here you could use the KML f
#we need to reproject our data because it is in the UTM system,
p.r.DEM <- projectRaster(r.DEM, crs = "+init=epsg:4326",
                         method = "bilinear")
KML(p.r.DEM, "HV_DEM.kml", col = rev(terrain.colors(255)),
    overwrite = TRUE)
```

```
 $ X        : num  340210 340310 340110 340210 340310 ...
 $ Y        : num  6362641 6362641 6362741 6362741 6362741 ...
 $ elevation: num  177 175 178 172 173 ...
> View(HV_dem)
> r.DEM <- rasterFromXYZ(HV_dem)
> proj4string(r.DEM) <- CRS("+init=epsg:32756")
> #Plots raster and overlay HV100 points
> plot(r.DEM)
> points(HV100, pch = 20)
> writeRaster(r.DEM, filename = "HV_dem100.asc",
+             format = "ascii", overwrite = TRUE)
> 
```

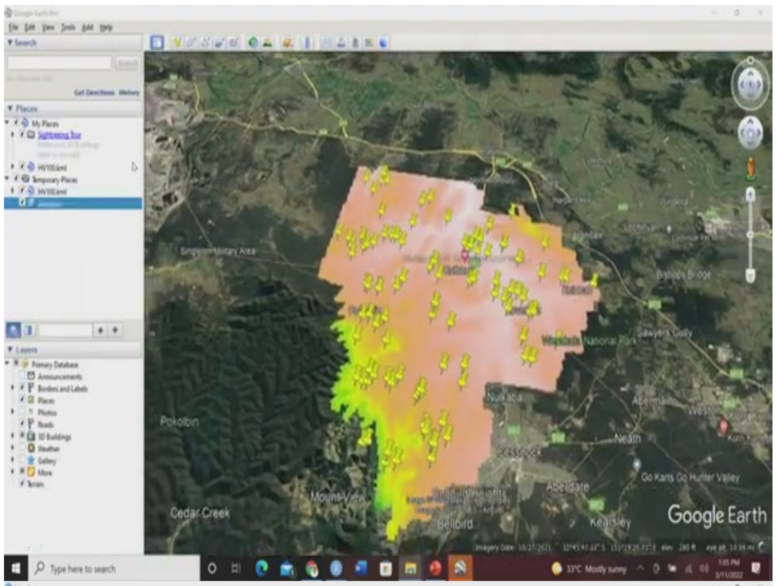
```
mod.2              List of  13
mod.data           146 obs. of 2 variables
mod.rh             List of  14
model_1            2 obs. of 9 variables
models             List of  28
pred.stack         Formal class  RasterStack
probs.hv.MNLR      num [1:700, 1:12] 8.12e-17 5.56e-08 2.4…
r.DEM              Formal class  RasterLayer
```

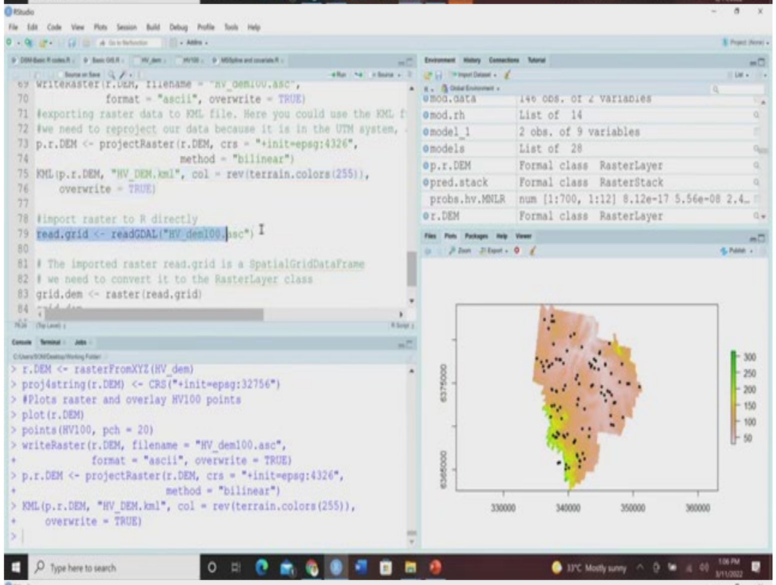


Working Folder

| Name | Date modified | Type | Size |
|---|---|---|---|
| file name | 3/11/2022 12:14 PM | Text Document | 14 KB |
| HV_dat_shape.dbf | 3/11/2022 12:52 PM | DBF File | 16 KB |
| HV_dat_shape.prj | 3/11/2022 12:52 PM | PRJ File | 1 KB |
| HV_dat_shape.shp | 3/11/2022 12:52 PM | SHP File | 3 KB |
| HV_dat_shape.shx | 3/11/2022 12:52 PM | SHX File | 1 KB |
| HV_DEM | 3/11/2022 1:05 PM | KML | 1 KB |
| HV_DEM | 3/11/2022 1:05 PM | PNG File | 29 KB |
| HV_dem | 3/11/2022 1:04 PM | ASC File | 301 KB |
| HV100 | 3/11/2022 12:54 PM | KML | 35 KB |

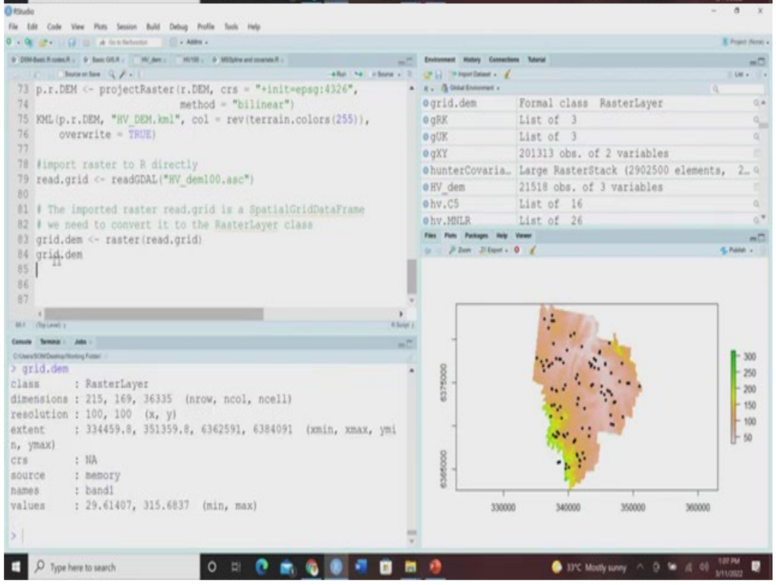
Type: KML
Size: 379 bytes
Date modified: 3/11/2022 1:05 PM




```
writeRaster(r.DEM, filename = "HV_dem100.asc",
            format = "ascii", overwrite = TRUE)
#exporting raster data to KML file. Here you could use the KML f
#we need to reproject our data because it is in the UTM system,
p.r.DEM <- projectRaster(r.DEM, crs = "+init=epsg:4326",
                         method = "bilinear")
KML(p.r.DEM, "HV_DEM.kml", col = rev(terrain.colors(255)),
    overwrite = TRUE)

#import raster to R directly
read.grid <- readGDAL("HV_dem100.asc")

# The imported raster read.grid is a SpatialGridDataFrame
# we need to convert it to the RasterLayer class
grid.dem <- raster(read.grid)
```

```
> r.DEM <- rasterFromXYZ(HV_dem)
> proj4string(r.DEM) <- CRS("+init=epsg:32756")
> #Plots raster and overlay HV100 points
> plot(r.DEM)
> points(HV100, pch = 20)
> writeRaster(r.DEM, filename = "HV_dem100.asc",
+             format = "ascii", overwrite = TRUE)
> p.r.DEM <- projectRaster(r.DEM, crs = "+init=epsg:4326",
+                          method = "bilinear")
> KML(p.r.DEM, "HV_DEM.kml", col = rev(terrain.colors(255)),
+     overwrite = TRUE)
> |
```

```
mod.data          146 obs. of 2 variables
mod.rh            List of 14
model_1           2 obs. of 9 variables
models            List of 28
p.r.DEM           Formal class RasterLayer
pred.stack        Formal class RasterStack
probs.hv.MNLR     num [1:700, 1:12] 8.12e-17 5.56e-08 2.4...
r.DEM             Formal class RasterLayer
```


```
p.r.DEM <- projectRaster(r.DEM, crs = "+init=epsg:4326",
                         method = "bilinear")
KML(p.r.DEM, "HV_DEM.kml", col = rev(terrain.colors(255)),
    overwrite = TRUE)

#import raster to R directly
read.grid <- readGDAL("HV_dem100.asc")

# The imported raster read.grid is a SpatialGridDataFrame
# we need to convert it to the RasterLayer class
grid.dem <- raster(read.grid)
grid.dem
```

```
> grid.dem
class       : RasterLayer
dimensions  : 215, 169, 36335  (nrow, ncol, ncell)
resolution  : 100, 100  (x, y)
extent      : 334459.8, 351359.8, 6362591, 6384091  (xmin, xmax, ymi
n, ymax)
crs         : NA
source      : memory
names       : band1
values      : 29.61407, 315.6837  (min, max)

> |
```

```
grid.dem          Formal class RasterLayer
gRK               List of 3
gUK               List of 3
gXY               201313 obs. of 2 variables
hunterCovaria..   Large RasterStack (2902500 elements, 2...
HV_dem            21518 obs. of 3 variables
hv.C5             List of 16
hv.MNLR           List of 26
```

If you go to your working folder you will see that this ascii file has been created HV underscore dem 100 dot aac and now you can export if you want to see this dem file in the

Google Earth domain so just like previous we have to convert it to kml file. So, we need to this project raster function.

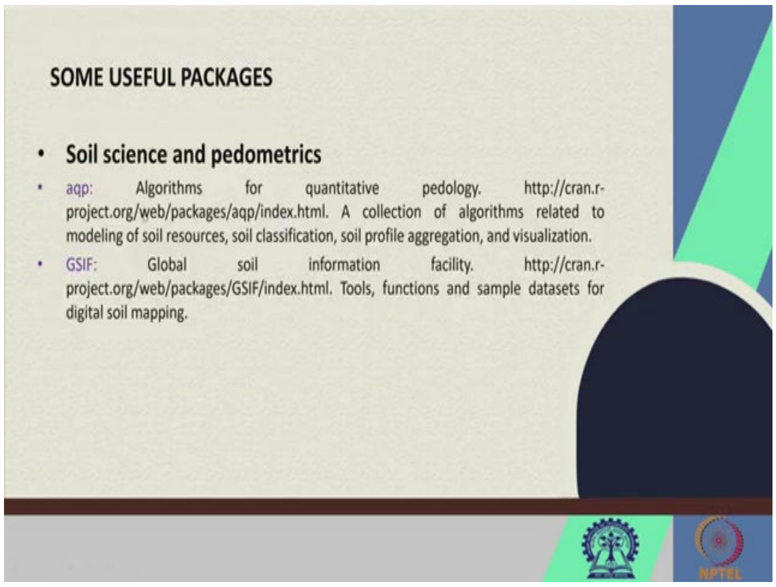
So, again we are assigning this EPSG code of 4326 which is the EPSG code for WGS 1984. So, we are doing this and then we are creating a kml file which is HV underscore dem dot kml file, so this file will be also created in your working folder. So, you can see that HV underscore dem dot kml file has been already created.

So, if you just directly click on it you will see that on the google earth you will get directly this dem file also. So, this is how you develop the dem file and so you can import the raster to R directly, so for that the command is readGDAL, so you can directly read the raster. So, here HV dem 100 has GDAL driver and it has 215 rows with 169 columns, you can see.

Now this imported raster read dot grid is a spatial grid data frame, so it is a spatial grid data frame, the earlier it was spatial points data frame but it is a spatial grid data frame, so we need to convert it to the raster layer class. So, for that you can use this raster function and then this you can see that grid dot dem is now a raster layer.

So, this is how you can play with different types of vector data and raster data. So, in the raster layer you can clearly see the dimension number of rows 215, number of columns is 169, so total number of values or cells are there 36335, resolution is 100 meter by 100 meter extent is given and also the minimum and maximum values are given. So, this is how you can see, this is the dem, grid dot dem file will look like, the raster file of this grid dot dem will look like this.

(Refer Slide Time: 9:12)

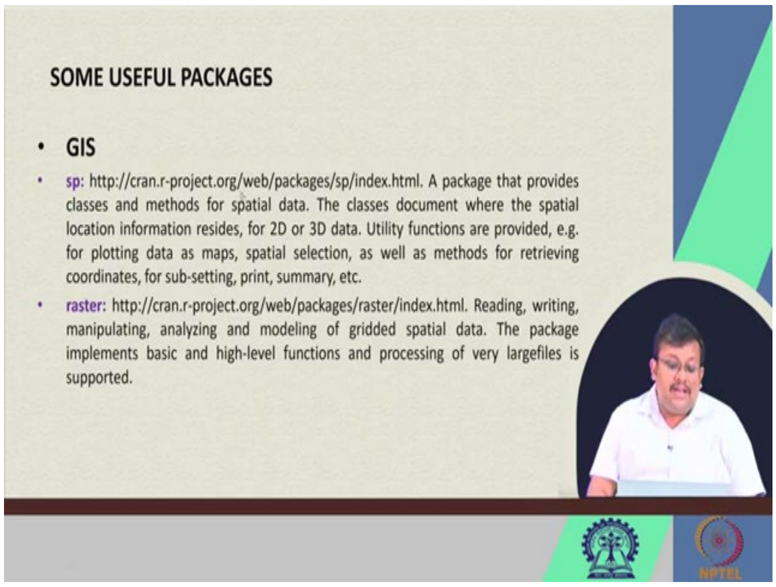


## SOME USEFUL PACKAGES

- **Soil science and pedometrics**
- aqp: Algorithms for quantitative pedology. http://cran.r-project.org/web/packages/aqp/index.html. A collection of algorithms related to modeling of soil resources, soil classification, soil profile aggregation, and visualization.
- GSIF: Global soil information facility. http://cran.r-project.org/web/packages/GSIF/index.html. Tools, functions and sample datasets for digital soil mapping.

Now, we are going to see some of the important packages which we require for the DSM operations so let us see this important packages for soil science and pedometrics we generally going to use this aqp package which is algorithm for quantitative pedology and these are the crayon source from where you can download this package, it is a collection of algorithm related to the modeling of soil sources or classification, soil profile aggregation and visualization. Next is the GSIF, GSIF stands for global soil information facility and it basically shows the, it basically contains the tools, functions and sample data set for digital soil mapping.

(Refer Slide Time: 9:46)

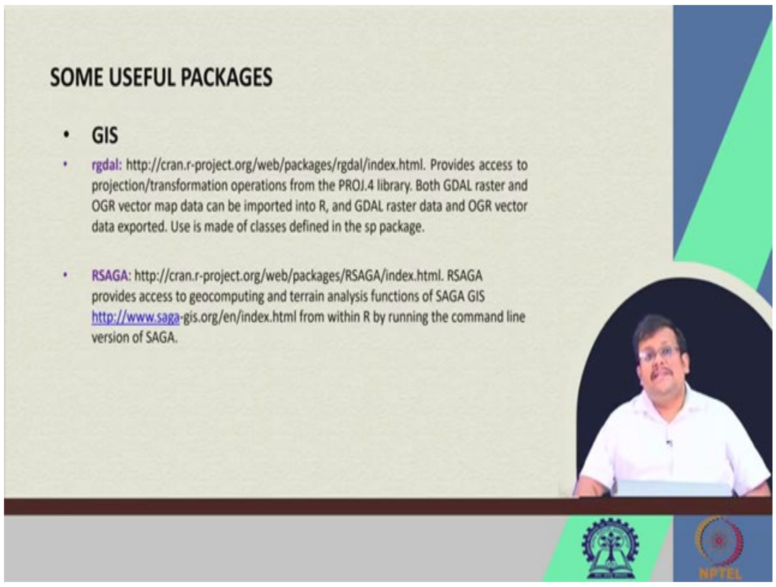


## SOME USEFUL PACKAGES

- **GIS**
- sp: http://cran.r-project.org/web/packages/sp/index.html. A package that provides classes and methods for spatial data. The classes document where the spatial location information resides, for 2D or 3D data. Utility functions are provided, e.g. for plotting data as maps, spatial selection, as well as methods for retrieving coordinates, for sub-setting, print, summary, etc.
- raster: http://cran.r-project.org/web/packages/raster/index.html. Reading, writing, manipulating, analyzing and modeling of gridded spatial data. The package implements basic and high-level functions and processing of very largefiles is supported.

GIS as there are two packages sp package which provides the classes and methods for spatial data and the classes document here the spatial location information resides for 2D and 3D data and so you can do different types of spatial operation in the sp package. Raster package is also there which deals with the manipulating, analyzing and modeling the gridded spatial data. You can do different types of operation in your raster file using this raster package. So, high level functions and processing of very large files also supported in this raster package.

(Refer Slide Time: 10:31)

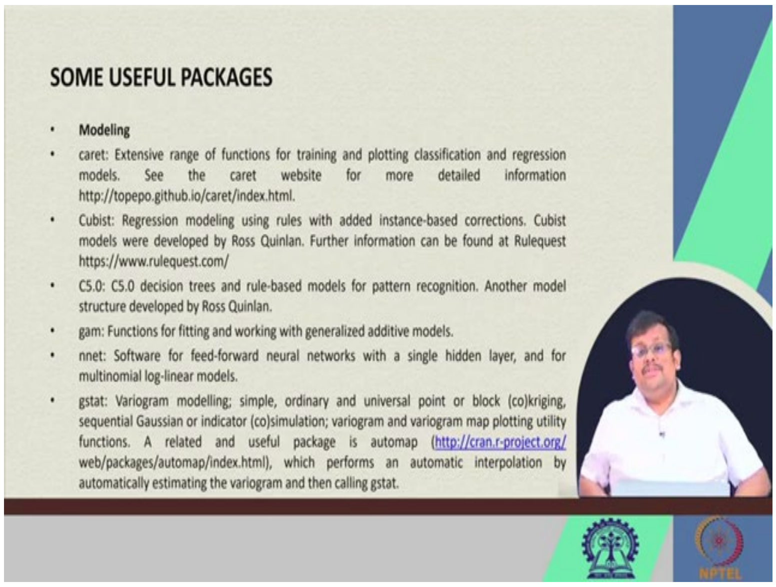


SOME USEFUL PACKAGES

- GIS
  - rgdal: http://cran.r-project.org/web/packages/rgdal/index.html. Provides access to projection/transformation operations from the PROJ.4 library. Both GDAL raster and OGR vector map data can be imported into R, and GDAL raster data and OGR vector data exported. Use is made of classes defined in the sp package.
  - RSAGA: http://cran.r-project.org/web/packages/RSAGA/index.html. RSAGA provides access to geocomputing and terrain analysis functions of SAGA GIS http://www.saga-gis.org/en/index.html from within R by running the command line version of SAGA.

GIS is specifically also dependent on this rgdal package, so and also RSAGA is another package which provides the access to geo computing and terrain analysis function of SAGA GIS which is a software.

(Refer Slide Time: 10:51)



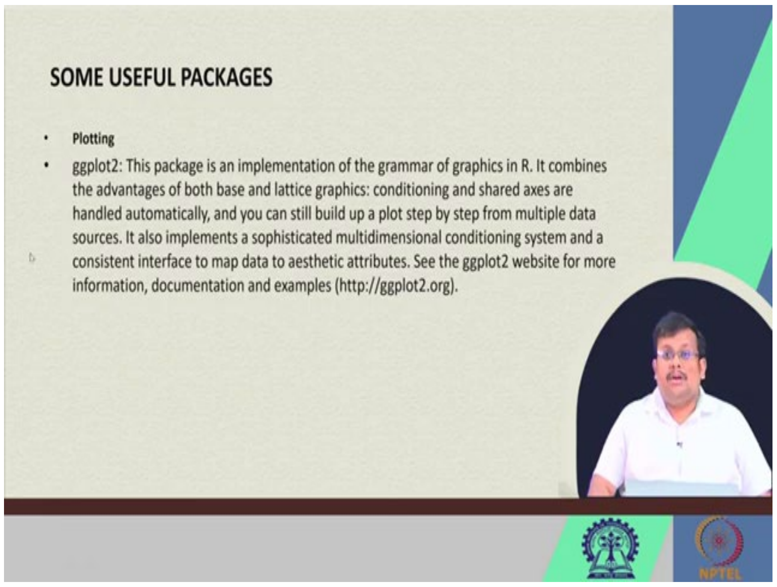
SOME USEFUL PACKAGES

- Modeling
- caret: Extensive range of functions for training and plotting classification and regression models. See the caret website for more detailed information http://topepo.github.io/caret/index.html.
- Cubist: Regression modeling using rules with added instance-based corrections. Cubist models were developed by Ross Quinlan. Further information can be found at Rulequest https://www.rulequest.com/
- C5.0: C5.0 decision trees and rule-based models for pattern recognition. Another model structure developed by Ross Quinlan.
- gam: Functions for fitting and working with generalized additive models.
- nnet: Software for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models.
- gstat: Variogram modelling; simple, ordinary and universal point or block (co)kriging, sequential Gaussian or indicator (co)simulation; variogram and variogram map plotting utility functions. A related and useful package is automap (http://cran.r-project.org/web/packages/automap/index.html), which performs an automatic interpolation by automatically estimating the variogram and then calling gstat.

Now, for modeling you can use different types of packages like caret package. Caret package has extensive range of functions for training and plotting classification regression models. Then cubist model is there, C5 model, we will going to discuss all these. Gam is basically stands for generalized additive models we have already discussed. Nnet for neural network models and for geostatistical models, geostatistical variograms kriging we are going to use the gstat package.

So, for modeling we are going to in this course we are going to see the classification regression or decision tree and then we are going to see the cubist and also we are going to see how to use the random forest to create the map for continuous variables. Also we are going to see the regression kriging which will be also using this gstat package. So, all these packages are really helpful for producing the desired outputs in digital soil mapping and we are going to see.

(Refer Slide Time: 12:04)

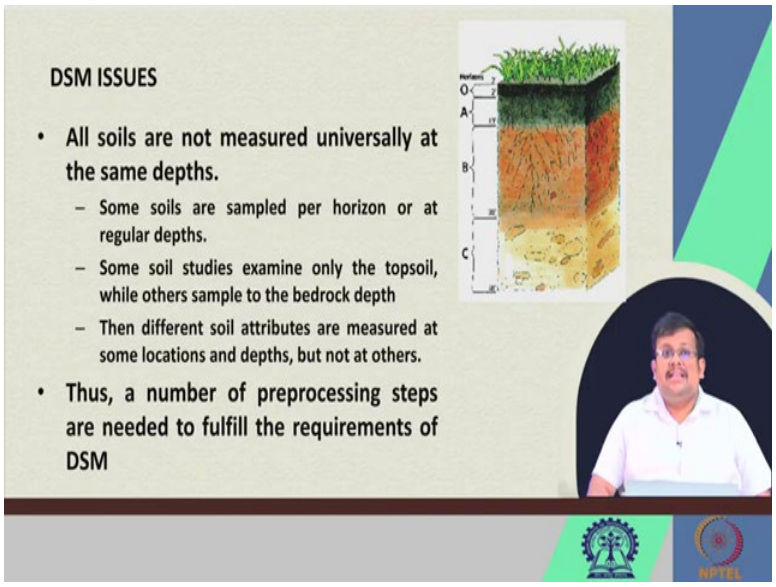


## SOME USEFUL PACKAGES

- **Plotting**
- ggplot2: This package is an implementation of the grammar of graphics in R. It combines the advantages of both base and lattice graphics: conditioning and shared axes are handled automatically, and you can still build up a plot step by step from multiple data sources. It also implements a sophisticated multidimensional conditioning system and a consistent interface to map data to aesthetic attributes. See the ggplot2 website for more information, documentation and examples (http://ggplot2.org).

And for plotting, ggplot2 all these are very important packages for DSM.

(Refer Slide Time: 12:28)



DSM ISSUES

- All soils are not measured universally at the same depths.
  - Some soils are sampled per horizon or at regular depths.
  - Some soil studies examine only the topsoil, while others sample to the bedrock depth
  - Then different soil attributes are measured at some locations and depths, but not at others.
- Thus, a number of preprocessing steps are needed to fulfill the requirements of DSM

Now, let us let us start a very important discussion that is mass preserving spline but before we go to discuss the mass preserving spline we need to understand that there are certain issues which we have to deal with when we talk about the digitized legacy soil data. So, remember that this legacy data which I have already told you is basically the data of the soil which has been gathered by standard laboratory analysis and some of them are digitized and some of them are not.

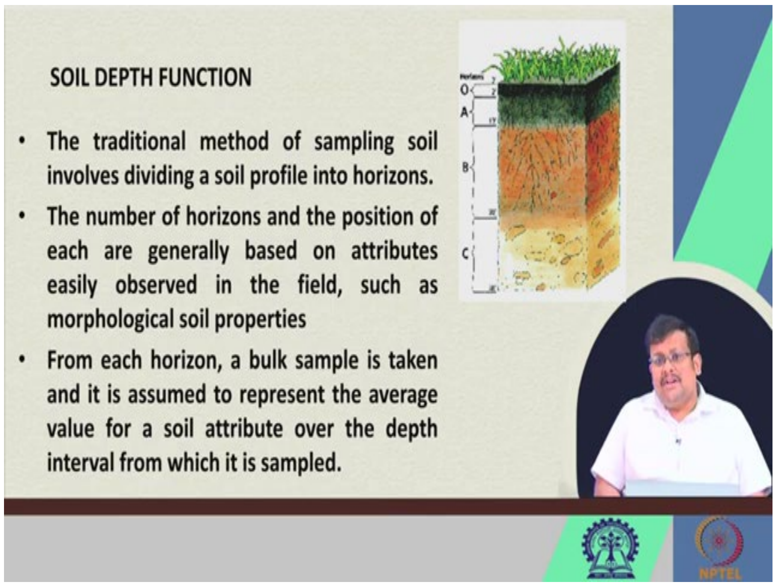
So, these original data set which are measured and maintained by different organizations for soil is known as the legacy soil data. But the problem is all soils are not measured universally at same depths, sometime you will see that some places will have the data for 0 to 30 centimeters, some places will have the data for 30 to 60 centimeters and also some places will have the data from 50 to 100 centimeter.

So, the depth at which this legacy data is available to us varies widely from one organization to another organization and within on one organization also from one data set to another data set. So, some soils are sample per horizon or at regular depth. So, basically you can see this is the soil profile, it has different horizons like O horizon, A horizon, B horizon, C horizon. So, some soils are sample, one sample is taken from each of this horizon and they assume that the sample property will represent the whole horizon, so this is one issue.

Another issue is some soil studies examine only the top soil while other sample to the bedrock depth. So, sometimes some database will have only the top soil information but some other sources may have up to this 2 meter depth or bedrock depth. The different soil attributes are measured at some location and depth but not at other.

So, these are some of the non-uniformity issues which poses problem for proper digital soil mapping and their execution. So, that is a number of preprocessing steps are needed to fulfill the requirement of digital soil mapping.

(Refer Slide Time: 15:07)



## SOIL DEPTH FUNCTION

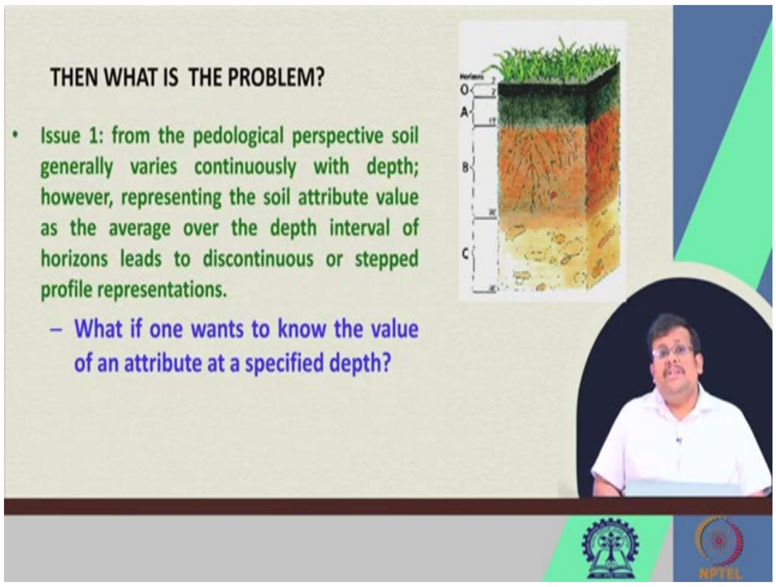
- The traditional method of sampling soil involves dividing a soil profile into horizons.
- The number of horizons and the position of each are generally based on attributes easily observed in the field, such as morphological soil properties
- From each horizon, a bulk sample is taken and it is assumed to represent the average value for a soil attribute over the depth interval from which it is sampled.

Now, what is the soil depth function. So, remember that the traditional method of sampling soil involves dividing a soil profile into horizon like O horizon, A horizon, B horizon, then C horizon. So, the number of horizon and the position of each are generally based on attributes easily observed in the fields such as morphological and soil properties.

So, what happens the surveyors or the pedalologies generally go to the field and then they see the what is going, they visually observe different characteristics features and then and then define the different horizons and from this each horizon a bulk sample is taken and it is assumed to represent the average value for a soil attribute over the depth interval from which it is sampled.

So, what happens they take 1 sample from each horizon and they assume that it represent the average value for a soil attribute over the depth interval from which it is sampled. However, given soil properties are highly variable that may not be the true case.

(Refer Slide Time: 16:21)



## THEN WHAT IS THE PROBLEM?

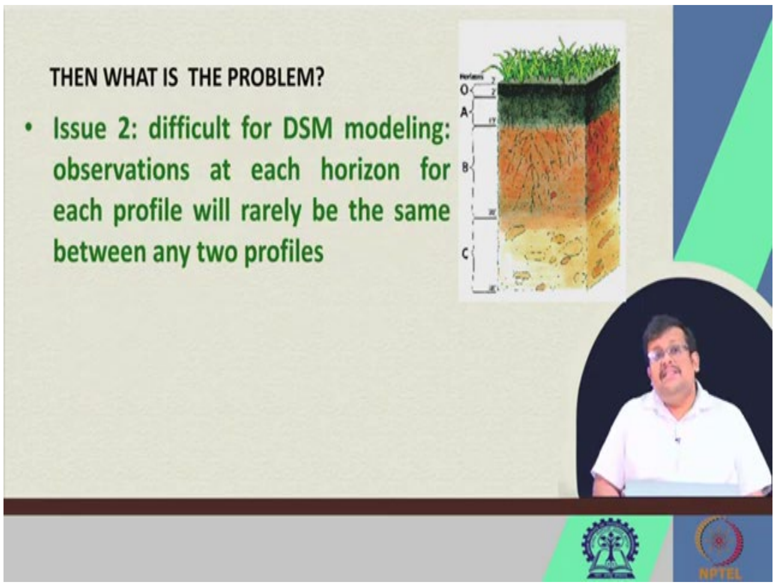
- Issue 1: from the pedological perspective soil generally varies continuously with depth; however, representing the soil attribute value as the average over the depth interval of horizons leads to discontinuous or stepped profile representations.
  - What if one wants to know the value of an attribute at a specified depth?

So, what is the issue? Issue is from the pedagogical perspective soil generally varies continuously with depth. However, representing the soil attribute value as the average over the depth interval of horizon leads to discontinuous or stepped profile representation. So, it is assumed that the pedagogical, pedagogically soil properties varies continuously with the depth.

So, for example, if you want to have higher organic carbon, organic carbon is always generally high at the top soil and then it decreases in the subsoil. So, representing the soil attribute value at the average over the depth interval of horizon leads to discontinuous and stepped profile representation.

So, what if one wants to know the value of an attribute at a specified depth? Suppose I want to have the value of a variable at 35 centimeter depth. So, how do I get that? If I want to pinpoint what is there in that 35 centimeter depth, how do I get that?

(Refer Slide Time: 17:33)

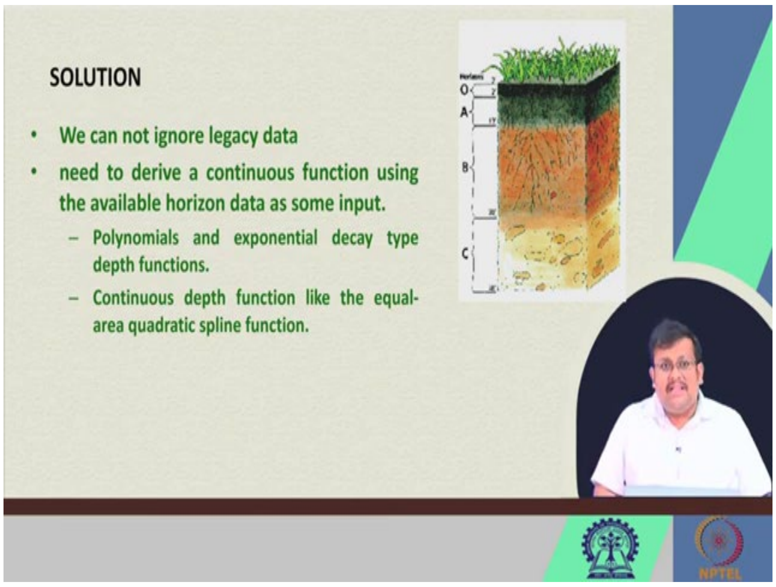


THEN WHAT IS THE PROBLEM?

- Issue 2: difficult for DSM modeling: observations at each horizon for each profile will rarely be the same between any two profiles

Another issue is difficulty for DSM modeling, that is observation at each horizon for each profile will rarely be the same between any two profile, so that is another issue.

(Refer Slide Time: 17:57)



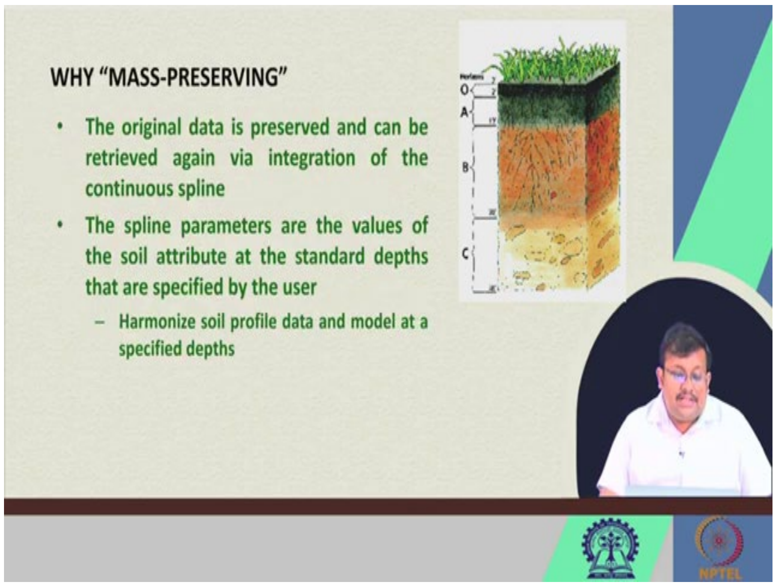
SOLUTION

- We can not ignore legacy data
- need to derive a continuous function using the available horizon data as some input.
  - Polynomials and exponential decay type depth functions.
  - Continuous depth function like the equal-area quadratic spline function.

So, due to the problem this type of problem we generally take the help of digital soil mapping or mass preserving spline. Because we cannot ignore the legacy data, legacy data is the original data, which is available to us, which is measured using the standard laboratory methods, so we cannot no ignore this legacy data. So, we need to derive a continuous function using the available horizon data at some inputs.

So, generally we use the polynomial and exponential decay type of depth function or continuous depth function like equal area quadratic spline function. So, let us see how this works.

(Refer Slide Time: 18:34)



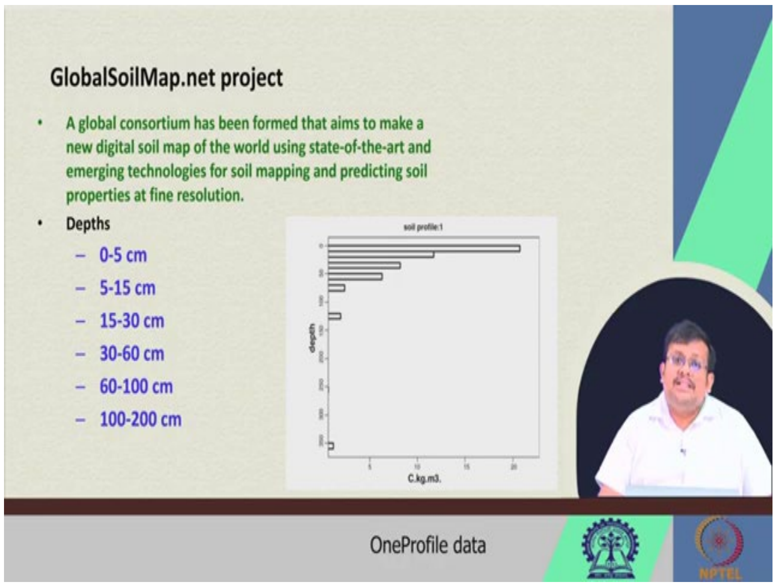
## WHY "MASS-PRESERVING"

- The original data is preserved and can be retrieved again via integration of the continuous spline
- The spline parameters are the values of the soil attribute at the standard depths that are specified by the user
  - Harmonize soil profile data and model at a specified depths

So, this equal area quadratics spline function is also known as the mass preserving spline. Why it is called mass preserving spline? I will show you that the original data is preserved and can be retrieved again by integration of the continuous spline. So, basically we fit a spline from the top of the soil profile to the desired depth generally for DSM we go from 0 centimeter to 2 meter depth, we fit a continuous spline and then we interpolate the values at any given depth.

Now, why it is called the mass preserving spline? It is called mass preserving spline because the original data can be preserved and can be retrieved again via integration of the continuous spline. So, the spline parameters are the values of the soil attribute at the standard depths that are specified by the user. And basically it needs to harmonize the soil profile data and model at a specified depths. So, how do I do that, I will show you.

(Refer Slide Time: 19:43)



## GlobalSoilMap.net project

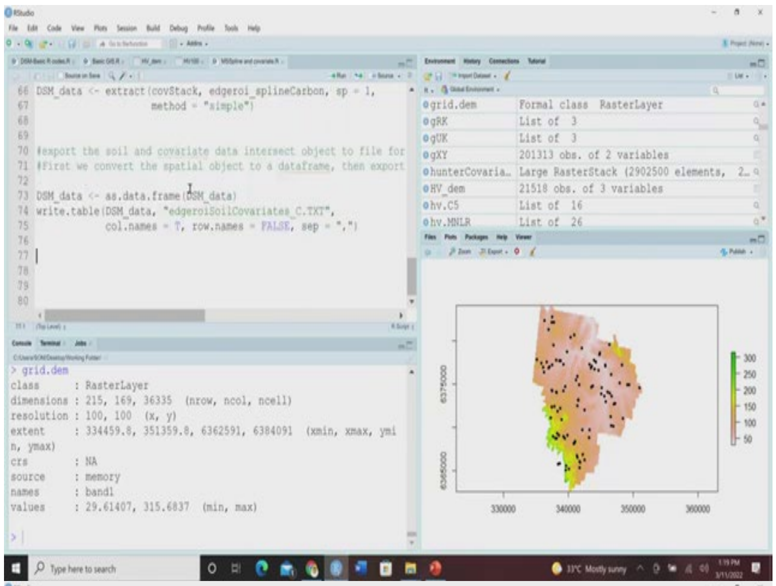
- A global consortium has been formed that aims to make a new digital soil map of the world using state-of-the-art and emerging technologies for soil mapping and predicting soil properties at fine resolution.
- Depths
  - 0-5 cm
  - 5-15 cm
  - 15-30 cm
  - 30-60 cm
  - 60-100 cm
  - 100-200 cm

OneProfile data

So, globalsoilmap dot net project is there, there is a project, it is a global consortium that has been formed that aim to make a new digital soil map of the world using the state of the art emerging technologies for soil mapping and predicting soil properties at fine resolution. So, this globalsoilmap dot net project helps in identifying or producing the digital soil maps at 6 different depths, so they have prescribed 6 different depths like 0 to 5 centimeter, 5 to 15 centimeter, 15 to 30 centimeter, 30 to 60 centimeter, 60 to 100 centimeter and 100 to 200 centimeter.

So, the maps for digital soil mapping are generally produced at this 6 depths and we are going to see how we can develop this maps for this. So, to deal more with this profile spline function or mass preserving spline let us go to the, let us go to the codes.

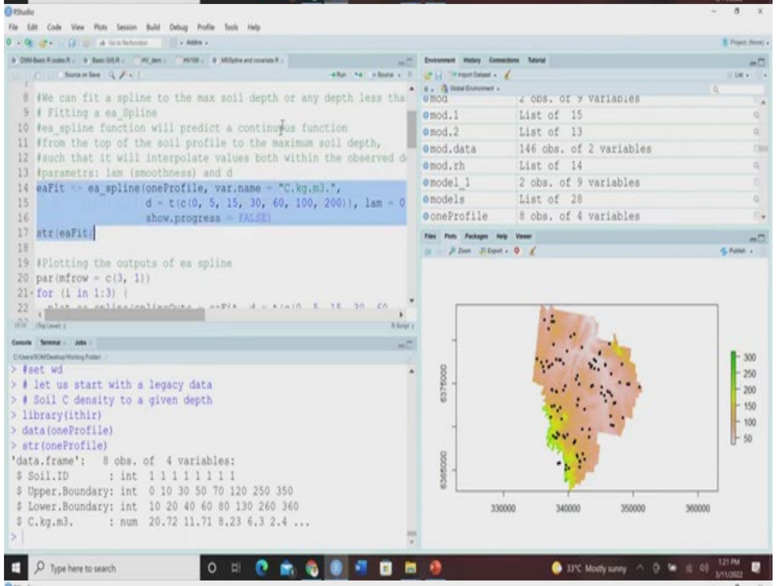
(Refer Slide Time: 20:53)



```
DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
                    method = "simple")


#export the soil and covariate data intersect object to file for
#First we convert the spatial object to a dataframe, then export

DSM_data <- as.data.frame(DSM_data)
write.table(DSM_data, "edgeroiSoilCovariates_C.TXT",
            col.names = T, row.names = FALSE, sep = ",")
```

```
> grid.dem
class      : RasterLayer
dimensions : 215, 169, 36335  (nrow, ncol, ncell)
resolution : 100, 100  (x, y)
extent     : 334459.8, 351359.8, 6362591, 6384091  (xmin, xmax, ymi
n, ymax)
crs        : NA
source     : memory
names      : band1
values     : 29.61407, 315.6837  (min, max)

>
```

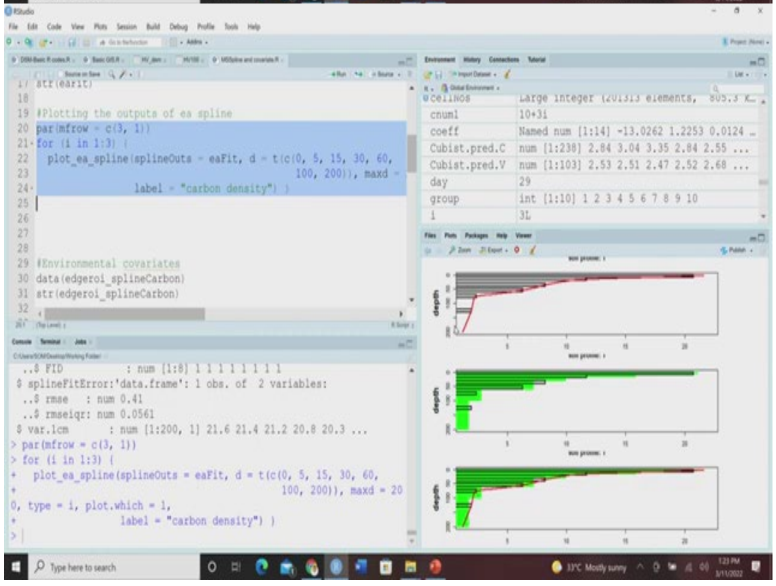
| | |
|---|---|
| grid.dem | Formal class  RasterLayer |
| gRK | List of  3 |
| gUK | List of  3 |
| gXY | 201313 obs. of 2 variables |
| hunterCovaria… | Large RasterStack (2902500 elements,  2… |
| HV_dem | 21518 obs. of 3 variables |
| hv.C5 | List of  16 |
| hv.MNLR | List of  26 |



```
#We can fit a spline to the max soil depth or any depth less tha
# Fitting a ea_Spline
#ea_spline function will predict a continuous function
#from the top of the soil profile to the maximum soil depth,
#such that it will interpolate values both within the observed d
#parametrs: lam (smoothness) and d
eaFit <- ea_spline(oneProfile, var.name = "C.kg.m3.",
                   d = t(c(0, 5, 15, 30, 60, 100, 200)), lam = 0
                   show.progress = FALSE)
str(eaFit)

#Plotting the outputs of ea spline
par(mfrow = c(3, 1))
for (i in 1:3) {
```

```
> #set wd
> # let us start with a legacy data
> # Soil C density to a given depth
> library(ithir)
> data(oneProfile)
> str(oneProfile)
'data.frame':   8 obs. of  4 variables:
 $ Soil.ID       : int  1 1 1 1 1 1 1 1
 $ Upper.Boundary: int  0 10 30 50 70 120 250 350
 $ Lower.Boundary: int  10 20 40 60 80 130 260 360
 $ C.kg.m3.      : num  20.72 11.71 8.23 6.3 2.4 ...
>
```

| | |
|---|---|
| mod | 2 obs. of 9 variables |
| mod.1 | List of  15 |
| mod.2 | List of  13 |
| mod.data | 146 obs. of 2 variables |
| mod.rh | List of  14 |
| model_1 | 2 obs. of 9 variables |
| models | List of  28 |
| oneProfile | 8 obs. of 4 variables |



```
str(eaFit)

#Plotting the outputs of ea spline
par(mfrow = c(3, 1))
for (i in 1:3) {
  plot_ea_spline(splineOuts = eaFit, d = t(c(0, 5, 15, 30, 60,
                                              100, 200)), maxd =
               label = "carbon density") }




#Environmental covariates
data(edgeroi_splineCarbon)
str(edgeroi_splineCarbon)
```

```
  ..$ FID          : num [1:8] 1 1 1 1 1 1 1 1
 $ splineFitError:'data.frame': 1 obs. of  2 variables:
  ..$ rmse   : num 0.41
  ..$ rmseiqr: num 0.0561
 $ var.1cm       : num [1:200, 1] 21.6 21.4 21.2 20.8 20.3 ...
> par(mfrow = c(3, 1))
> for (i in 1:3) {
+   plot_ea_spline(splineOuts = eaFit, d = t(c(0, 5, 15, 30, 60,
+                                              100, 200)), maxd = 20
0, type = i, plot.which = 1,
+                 label = "carbon density") }
>
```

| | |
|---|---|
| cellNos | Large integer (201313 elements,  805.3 K… |
| cnum1 | 10+3i |
| coeff | Named num [1:14] -13.0262 1.2253 0.0124 … |
| Cubist.pred.C | num [1:238] 2.84 3.04 3.35 2.84 2.55 ... |
| Cubist.pred.V | num [1:103] 2.53 2.51 2.47 2.52 2.68 ... |
| day | 29 |
| group | int [1:10] 1 2 3 4 5 6 7 8 9 10 |
| i | 3L |

So, you see in this course we have discuss all these things so we are again going to use this library ithir and the data we are going to use is called the one profile data and let us see the structure of the one profile data. So, the one profile data is having 8 observation and 4 variables, so soil id is given, upper boundary is given, lower boundary is given and carbon concentration or carbon content is given, organic carbon content is given.

So, we can fit a spline to the maximum soil depth of any depth less than that and so whatever maximum depth information is you have you can fit the spline up to that depth from the top of the profile to that maximum soil depth, such that it will interpolate the values both between the observed depths and between the depths where there is no observation. So, interpolation by the nature, by the name interpolation it says that it can predict the value at any given depth between the two extreme conditions.

So, in this we are going to use a parameter called lambda which is smoothness parameter so we can vary the smoothness parameter to see how, to make the spline more flexible or rigid. So, both this extreme have some pros and cons, so we are going to use this ea spline function, so ea stands for equal area underscore spline function, we are going to use this one profile data which we have already downloaded, our variable is carbon and the depth at which we want to, we want to interpolate is 0 5 15 30 60 100 and 200 and our lambda parameter is 0.1.

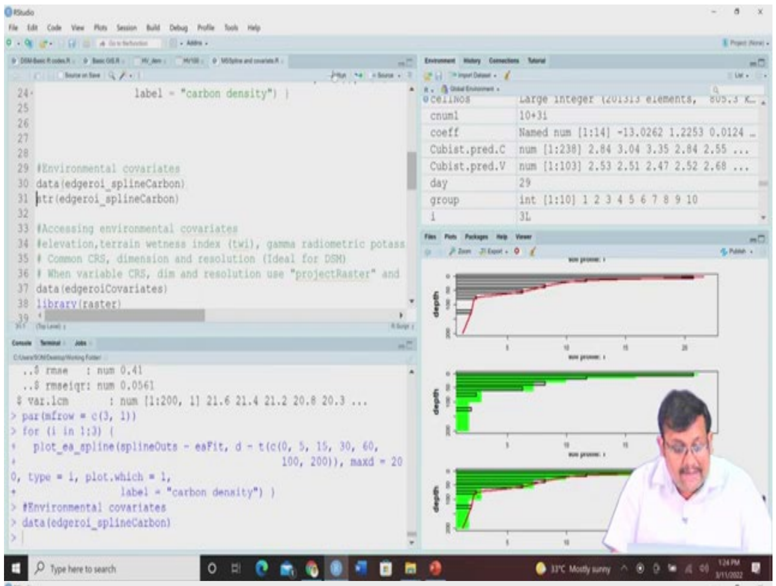
So, and let us see the structure of this. So, ultimately once we develop this thing and then we harmonize the data at this given depths you can see that it is being harmonized and the upper boundary, lower boundary and all these things and the values are being generated. Now, how to plot the output of an ea spline?

So, we are going to use this plot underscore ea underscore spline function. So, we can use this ea fit outputs for these given depth, 6 different depths remember, 0 to 5 centimeter, 5 to 15 centimeter, 15 to 30 centimeter, 30 to 60, 60 to 100 and 100 to 200. And since our maximum depth is 200 will go up to 200 centimeter depth.

So, let us first develop this and one thing you can see here we are using some index to generate the plot. So, you can see here level is carbon density. So, here you can see 3 plots are being produced. So, this is the first spline fitting and then this is how interpolations are made at different depths, so these are combined together to get this type of values.

So, this is step representation, this is the spline representation, using the spline you can get the values at any given depth. So, guys this is how this spline function can take. So, here you can see these are some missing values in the one profile data. So, here there are some missing values but when you fit a spline it can produce, it can predict the variable at any given data up to 200 centimeter up to which it has result, it has observation. So, it is a continuous spline and using this continuous spline you can predict the parameter at any given depth.

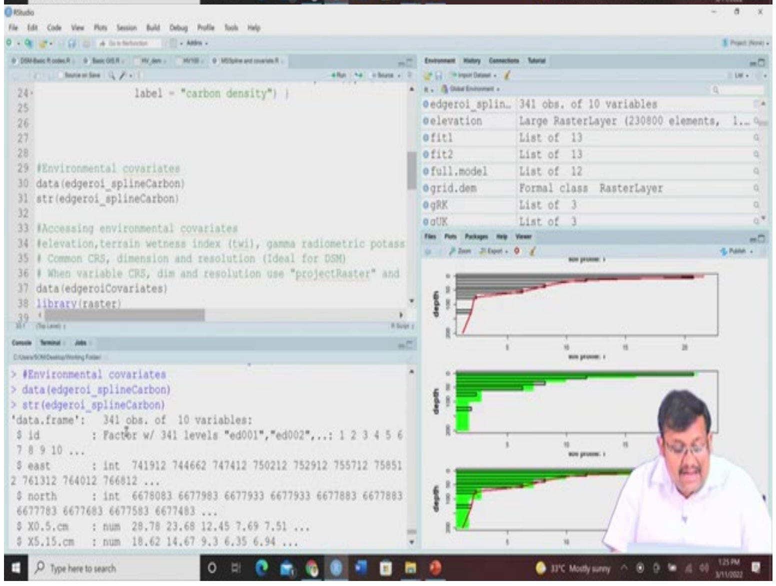
(Refer Slide Time: 25:06)



```
                    label = "carbon density") )




#Environmental covariates
data(edgeroi_splineCarbon)
str(edgeroi_splineCarbon)

#Accessing environmental covariates
#elevation,terrain wetness index (twi), gamma radiometric potass
# Common CRS, dimension and resolution (Ideal for DSM)
# When variable CRS, dim and resolution use "projectRaster" and
data(edgeroiCovariates)
library(raster)
```

```
  ..$ rmse   : num 0.41
  ..$ rmseiqr: num 0.0561
 $ var.1cm      : num [1:200, 1] 21.6 21.4 21.2 20.8 20.3 ...
> par(mfrow = c(3, 1))
> for (i in 1:3) {
+   plot_ea_spline(splineOuts = eaFit, d = t(c(0, 5, 15, 30, 60,
+                                            100, 200)), maxd = 20
0, type = 1, plot.which = i,
+                  label = "carbon density") )
> #Environmental covariates
> data(edgeroi_splineCarbon)
> |
```

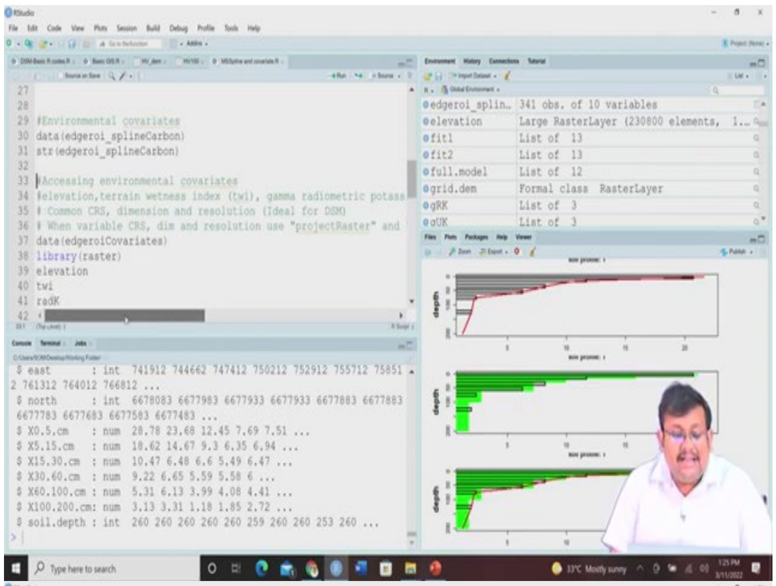
```
cellNos         Large integer (201313 elements,  805.3 K…
cnum1           10+3i
coeff           Named num [1:14] -13.0262 1.2253 0.0124 …
Cubist.pred.C   num [1:238] 2.84 3.04 3.35 2.84 2.55 ...
Cubist.pred.V   num [1:103] 2.53 2.51 2.47 2.52 2.68 ...
day             29
group           int [1:10] 1 2 3 4 5 6 7 8 9 10
i               3L
```



```
                    label = "carbon density") )




#Environmental covariates
data(edgeroi_splineCarbon)
str(edgeroi_splineCarbon)

#Accessing environmental covariates
#elevation,terrain wetness index (twi), gamma radiometric potass
# Common CRS, dimension and resolution (Ideal for DSM)
# When variable CRS, dim and resolution use "projectRaster" and
data(edgeroiCovariates)
library(raster)
```

```
> #Environmental covariates
> data(edgeroi_splineCarbon)
> str(edgeroi_splineCarbon)
'data.frame':   341 obs. of  10 variables:
 $ id         : Factor w/ 341 levels "ed001","ed002",..: 1 2 3 4 5 6
 7 8 9 10 ...
 $ east       : int  741912 744662 747412 750212 752912 755712 75851
2 761312 764012 766812 ...
 $ north      : int  6678083 6677983 6677933 6677933 6677883 6677883
 6677783 6677683 6677583 6677483 ...
 $ X0.5.cm    : num  28.78 23.68 12.45 7.69 7.51 ...
 $ X5.15.cm   : num  18.62 14.67 9.3 6.35 6.94 ...
```

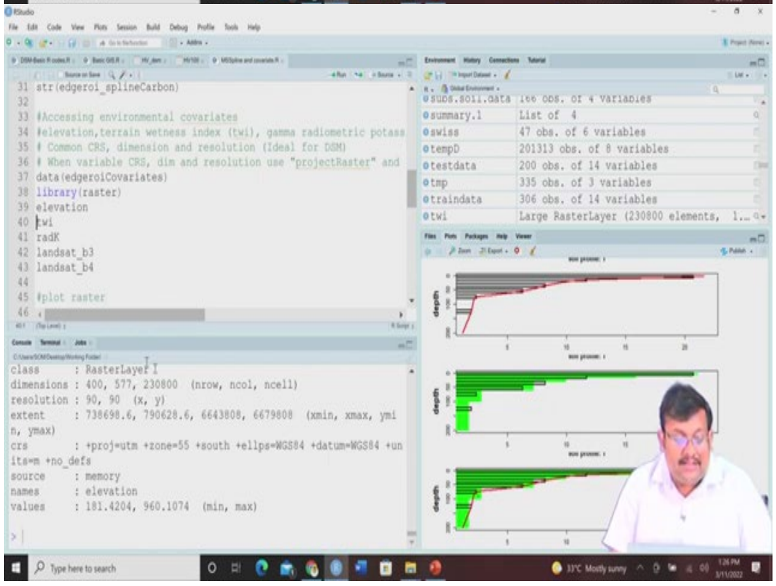
```
edgeroi_splin…  341 obs. of 10 variables
elevation       Large RasterLayer (230800 elements,  1…
fit1            List of  13
fit2            List of  13
full.model      List of  12
grid.dem        Formal class  RasterLayer
gRK             List of  3
gUK             List of  3
```



```
#Environmental covariates
data(edgeroi_splineCarbon)
str(edgeroi_splineCarbon)

#Accessing environmental covariates
#elevation,terrain wetness index (twi), gamma radiometric potass
# Common CRS, dimension and resolution (Ideal for DSM)
# When variable CRS, dim and resolution use "projectRaster" and
data(edgeroiCovariates)
library(raster)
elevation
twi
radK
```

```
 $ east       : int  741912 744662 747412 750212 752912 755712 75851
2 761312 764012 766812 ...
 $ north      : int  6678083 6677983 6677933 6677933 6677883 6677883
 6677783 6677683 6677583 6677483 ...
 $ X0.5.cm    : num  28.78 23.68 12.45 7.69 7.51 ...
 $ X5.15.cm   : num  18.62 14.67 9.3 6.35 6.94 ...
 $ X15.30.cm  : num  10.47 6.48 6.6 5.49 6.47 ...
 $ X30.60.cm  : num  9.22 6.65 5.59 5.58 6 ...
 $ X60.100.cm : num  5.31 6.13 3.99 4.08 4.41 ...
 $ X100.200.cm: num  3.13 3.31 1.18 1.85 2.72 ...
 $ soil.depth : int  260 260 260 260 260 259 260 260 253 260 ...
>
```

```
edgeroi_splin…  341 obs. of 10 variables
elevation       Large RasterLayer (230800 elements,  1…
fit1            List of  13
fit2            List of  13
full.model      List of  12
grid.dem        Formal class  RasterLayer
gRK             List of  3
gUK             List of  3
```



```
str(edgeroi_splineCarbon)

#Accessing environmental covariates
#elevation,terrain wetness index (twi), gamma radiometric potass
# Common CRS, dimension and resolution (Ideal for DSM)
# When variable CRS, dim and resolution use "projectRaster" and
data(edgeroiCovariates)
library(raster)
elevation
twi
radK
landsat_b3
landsat_b4

#plot raster
```

```
class       : RasterLayer
dimensions  : 400, 577, 230800  (nrow, ncol, ncell)
resolution  : 90, 90  (x, y)
extent      : 738698.6, 790628.6, 6643808, 6679808  (xmin, xmax, ymi
n, ymax)
crs         : +proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84 +un
its=m +no_defs
source      : memory
names       : elevation
values      : 181.4204, 960.1074  (min, max)
>
```

```
subs.soil.data  166 obs. of 4 variables
summary.1       List of  4
swiss           47 obs. of 6 variables
tempD           201313 obs. of 8 variables
testdata        200 obs. of 14 variables
tmp             335 obs. of 3 variables
traindata       306 obs. of 14 variables
twi             Large RasterLayer (230800 elements,  1…
```

Now, the next important thing is environmental covariates. Now, I told you about this environmental covariates for doing any type of digital soil mapping you need to use these environmental covariates like either rainfall data or terrain parameters or different types of remote sensing data.

So, I am going to show you why we go for this, how we download the data and how we intercept the data in a single file so that we can use them for future prediction of soil properties. So, here we are going to download the covariates called edgeroi underscore spline carbon. So, the data we are going to use this edgeroi underscore spline carbon data and the structure of the edgeroi spline carbon data you can see here it is having the 341 observation of 10 variables.

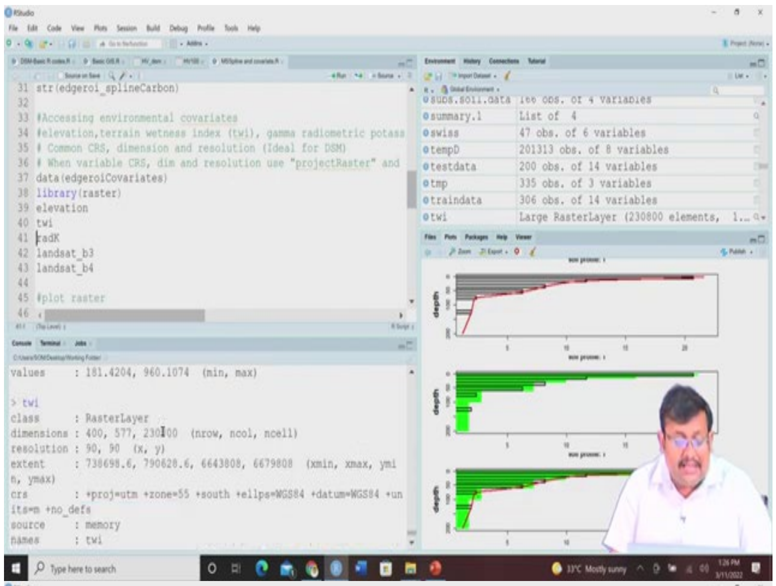
So, it has profile id, then easting and northing and of course it has 0 to 5 centimeter depth organic carbon values, 5 to 15 centimeter, 15 to 30 centimeter, 30 to 60 centimeter, 60 to 100 centimeter, 100 to 200 centimeter and soil depth are also given. Now, how to download these environmental covariates?

So, here we are down, we are going to download the 5 environmental covariates like elevation, then terrain wetness index or twi, gamma radiometric potassium that is potassium on the soil surface measured to gamma radiometry and landsat, band 3, and landsat 7 spectral band 3 and spectral band 4 values.

So, these 5 variables or environmental covariates we are going to use. For the sake of simplicity here all these 5 layers or raster layers are having the same coordinate reference system and dimensions and resolution which is ideal for DSM remember the support size which you have talked about.

Now, when there are variable coordinate reference system dimensions and resolution you have to use the project raster command. Now, here all these are same so let us call, let us first download this data this is called edgeroi covariate data and then let us call this library raster so if we see this elevation let us first go with the elevation. So, here you can see this is the resolution 90 meter by 90 meter, dimension 400, 400 number of rows, 577 columns and 230800 number of cells.

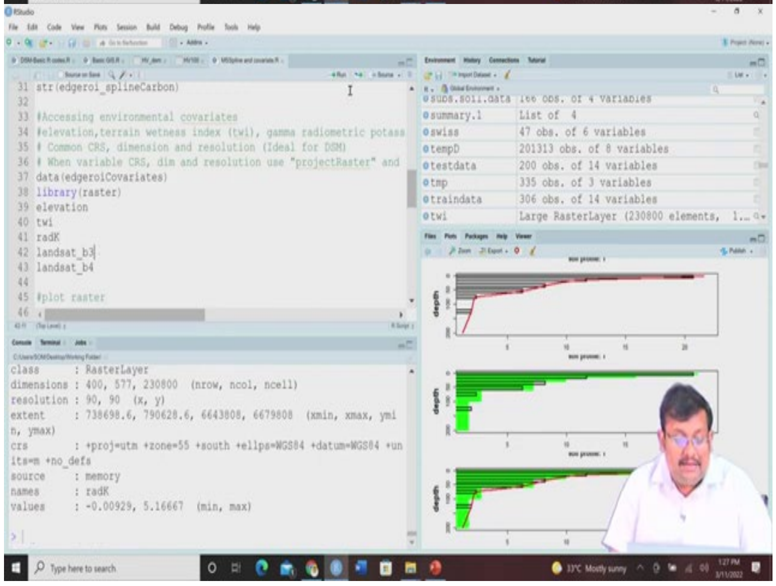
(Refer Slide Time: 28:03)



```
str(edgeroi_splineCarbon)

#Accessing environmental covariates
#elevation,terrain wetness index (twi), gamma radiometric potass
# Common CRS, dimension and resolution (Ideal for DSM)
# When variable CRS, dim and resolution use "projectRaster" and
data(edgeroiCovariates)
library(raster)
elevation
twi
radK
landsat_b3
landsat_b4

#plot raster
```

```
values     : 181.4204, 960.1074  (min, max)

> twi
class      : RasterLayer
dimensions : 400, 577, 230800  (nrow, ncol, ncell)
resolution : 90, 90  (x, y)
extent     : 738698.6, 790628.6, 6643808, 6679808  (xmin, xmax, ymi
n, ymax)
crs        : +proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84 +un
its=m +no_defs
source     : memory
names      : twi
```

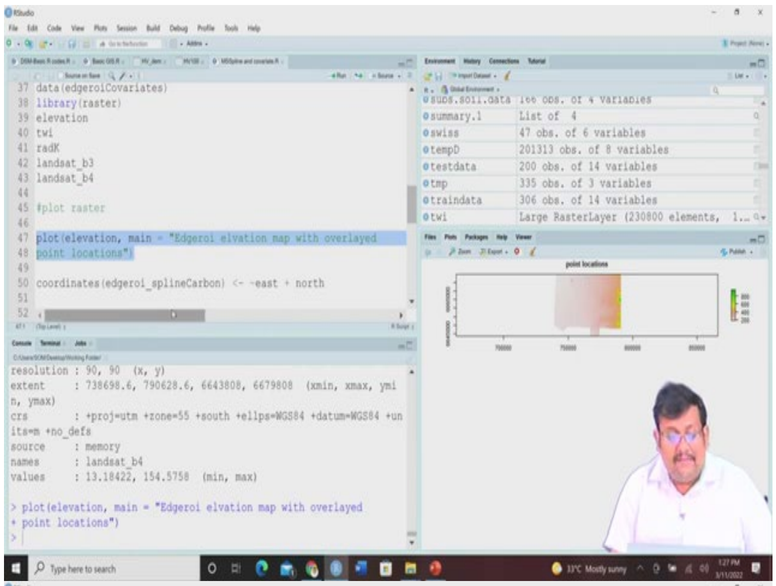
```
subs.soil.data  166 obs. of 4 variables
summary.1       List of  4
swiss           47 obs. of 6 variables
tempD           201313 obs. of 8 variables
testdata        200 obs. of 14 variables
tmp             335 obs. of 3 variables
traindata       306 obs. of 14 variables
twi             Large RasterLayer (230800 elements, 1...
```



```
str(edgeroi_splineCarbon)

#Accessing environmental covariates
#elevation,terrain wetness index (twi), gamma radiometric potass
# Common CRS, dimension and resolution (Ideal for DSM)
# When variable CRS, dim and resolution use "projectRaster" and
data(edgeroiCovariates)
library(raster)
elevation
twi
radK
landsat_b3
landsat_b4

#plot raster
```

```
class      : RasterLayer
dimensions : 400, 577, 230800  (nrow, ncol, ncell)
resolution : 90, 90  (x, y)
extent     : 738698.6, 790628.6, 6643808, 6679808  (xmin, xmax, ymi
n, ymax)
crs        : +proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84 +un
its=m +no_defs
source     : memory
names      : radK
values     : -0.00929, 5.16667  (min, max)

>
```

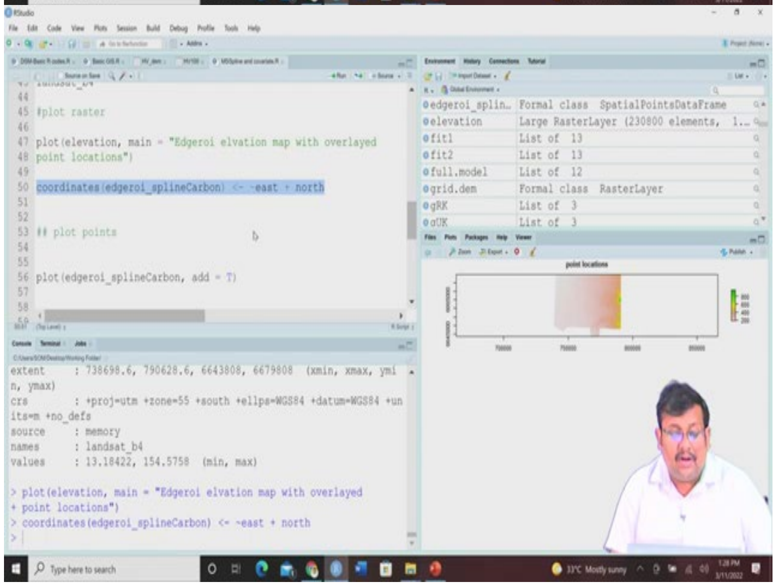
```
subs.soil.data  166 obs. of 4 variables
summary.1       List of  4
swiss           47 obs. of 6 variables
tempD           201313 obs. of 8 variables
testdata        200 obs. of 14 variables
tmp             335 obs. of 3 variables
traindata       306 obs. of 14 variables
twi             Large RasterLayer (230800 elements, 1...
```



```
data(edgeroiCovariates)
library(raster)
elevation
twi
radK
landsat_b3
landsat_b4

#plot raster

plot(elevation, main = "Edgeroi elvation map with overlayed
point locations")

coordinates(edgeroi_splineCarbon) <- ~east + north
```

```
resolution : 90, 90  (x, y)
extent     : 738698.6, 790628.6, 6643808, 6679808  (xmin, xmax, ymi
n, ymax)
crs        : +proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84 +un
its=m +no_defs
source     : memory
names      : landsat_b4
values     : 13.18422, 154.5758  (min, max)

> plot(elevation, main = "Edgeroi elvation map with overlayed
+ point locations")
>
```

```
subs.soil.data  166 obs. of 4 variables
summary.1       List of  4
swiss           47 obs. of 6 variables
tempD           201313 obs. of 8 variables
testdata        200 obs. of 14 variables
tmp             335 obs. of 3 variables
traindata       306 obs. of 14 variables
twi             Large RasterLayer (230800 elements, 1…
```



```
#plot raster

plot(elevation, main = "Edgeroi elvation map with overlayed
point locations")

coordinates(edgeroi_splineCarbon) <- ~east + north


## plot points


plot(edgeroi_splineCarbon, add = T)
```

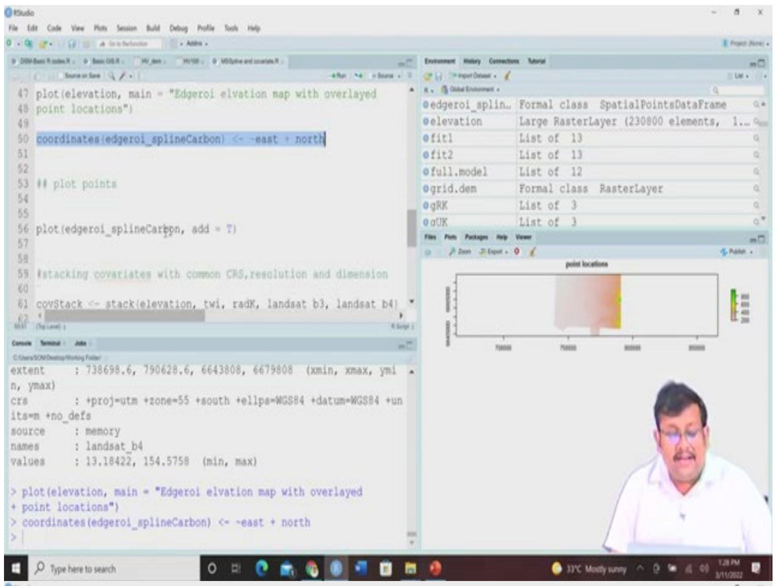
```
extent     : 738698.6, 790628.6, 6643808, 6679808  (xmin, xmax, ymi
n, ymax)
crs        : +proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84 +un
its=m +no_defs
source     : memory
names      : landsat_b4
values     : 13.18422, 154.5758  (min, max)

> plot(elevation, main = "Edgeroi elvation map with overlayed
+ point locations")
> coordinates(edgeroi_splineCarbon) <- ~east + north
>
```

```
edgeroi_splin…  Formal class  SpatialPointsDataFrame
elevation       Large RasterLayer (230800 elements, 1…
fit1            List of  13
fit2            List of  13
full.model      List of  12
grid.dem        Formal class  RasterLayer
gRK             List of  3
gUK             List of  3
```

Similarly, if you go to terrain wetness index, so then you can see here the same resolution just like so 400, 577, 230800, so 90 meter by 90 meter resolution, so for all of these we are having the same resolution landsat b3 and also landsat b4 we are having the same resolution. So, this will be, now how to overlay them?

So, the procedure of plot them or overlay them and then extract their values together in a single file is known as the intersection. So, let us plot this raster elevation, let us do this so if we plot this raster elevation so you can see here this is the elevation file which you have plotted and the coordinates are of course easting and northing we can identify, we can instruct R that you should understand this x and y are the coordinates.

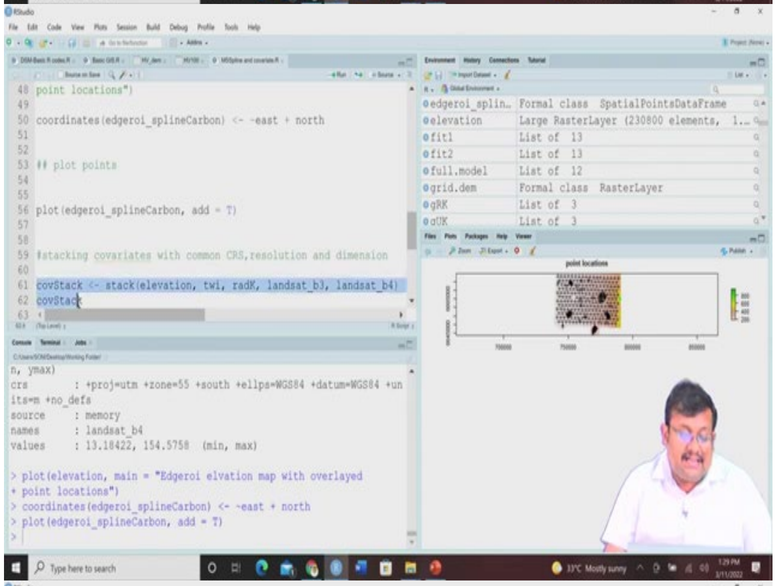
(Refer Slide Time: 29:16)



```
plot(elevation, main = "Edgeroi elvation map with overlayed
point locations")

coordinates(edgeroi_splineCarbon) <- ~east + north


## plot points


plot(edgeroi_splineCarbon, add = T)


#stacking covariates with common CRS,resolution and dimension

covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
```

```
extent     : 738698.6, 790628.6, 6643808, 6679808  (xmin, xmax, ymi
n, ymax)
crs        : +proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84 +un
its=m +no_defs
source     : memory
names      : landsat_b4
values     : 13.18422, 154.5758  (min, max)

> plot(elevation, main = "Edgeroi elvation map with overlayed
+ point locations")
> coordinates(edgeroi_splineCarbon) <- ~east + north
>
```

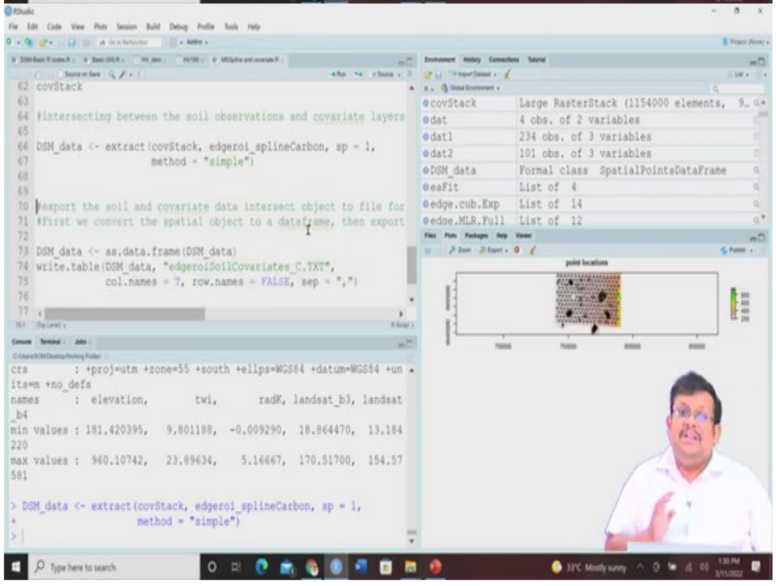
```
edgeroi_splin…   Formal class  SpatialPointsDataFrame
elevation        Large RasterLayer (230800 elements,  1…
fit1             List of  13
fit2             List of  13
full.model       List of  12
grid.dem         Formal class  RasterLayer
gRK              List of  3
aUK              List of  3
```



```
point locations")

coordinates(edgeroi_splineCarbon) <- ~east + north


## plot points


plot(edgeroi_splineCarbon, add = T)


#stacking covariates with common CRS,resolution and dimension

covStack <- stack(elevation, twi, radK, landsat_b3, landsat_b4)
covStack
```

```
n, ymax)
crs        : +proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84 +un
its=m +no_defs
source     : memory
names      : landsat_b4
values     : 13.18422, 154.5758  (min, max)

> plot(elevation, main = "Edgeroi elvation map with overlayed
+ point locations")
> coordinates(edgeroi_splineCarbon) <- ~east + north
> plot(edgeroi_splineCarbon, add = T)
>
```

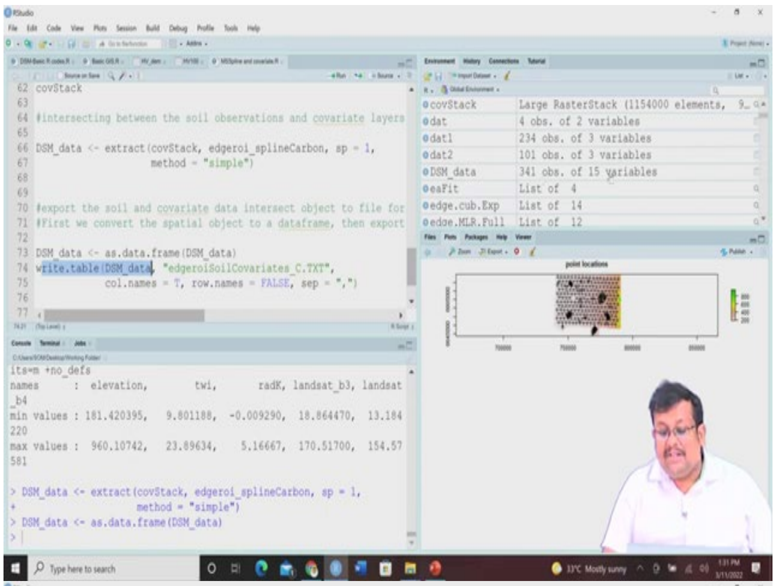
```
edgeroi_splin…   Formal class  SpatialPointsDataFrame
elevation        Large RasterLayer (230800 elements,  1…
fit1             List of  13
fit2             List of  13
full.model       List of  12
grid.dem         Formal class  RasterLayer
gRK              List of  3
aUK              List of  3
```



```
covStack

#intersecting between the soil observations and covariate layers

DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
                    method = "simple")


#export the soil and covariate data intersect object to file for
#First we convert the spatial object to a dataframe, then export

DSM_data <- as.data.frame(DSM_data)
write.table(DSM_data, "edgeroiSoilCovariates_C.TXT",
            col.names = T, row.names = FALSE, sep = ",")
```

```
crs        : +proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84 +un
its=m +no_defs
names      :  elevation,       twi,       radK, landsat_b3, landsat
_b4
min values : 181.420395,   9.801188,  -0.009290,  18.864470,  13.184
220
max values :  960.10742,   23.89634,    5.16667,  170.51700,  154.57
581

> DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
+                     method = "simple")
>
```

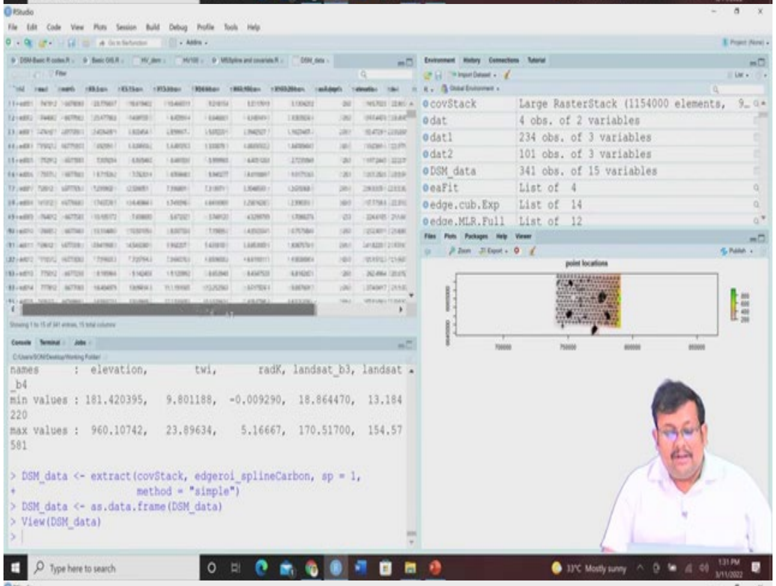
```
covStack         Large RasterStack (1154000 elements,  9…
dat              4 obs. of 2 variables
dat1             234 obs. of 3 variables
dat2             101 obs. of 3 variables
DSM_data         Formal class  SpatialPointsDataFrame
eaFit            List of  4
edge.cub.Exp     List of  14
edge.MLR.Full    List of  12
```



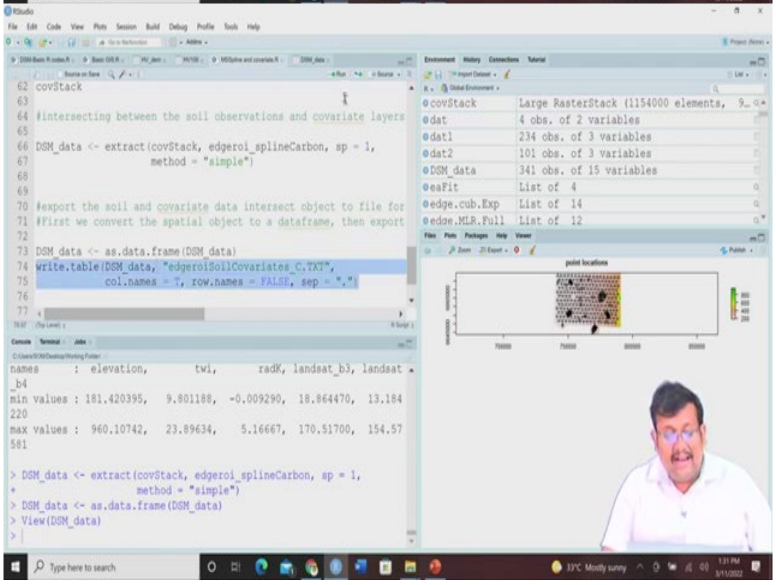
```
covStack

#intersecting between the soil observations and covariate layers

DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
                    method = "simple")


#export the soil and covariate data intersect object to file for
#First we convert the spatial object to a dataframe, then export

DSM_data <- as.data.frame(DSM_data)
write.table(DSM_data, "edgeroiSoilCovariates_C.TXT",
            col.names = T, row.names = FALSE, sep = ",")
```

| | |
|---|---|
| covStack | Large RasterStack (1154000 elements, 9… |
| dat | 4 obs. of 2 variables |
| dat1 | 234 obs. of 3 variables |
| dat2 | 101 obs. of 3 variables |
| DSM_data | 341 obs. of 15 variables |
| eaFit | List of 4 |
| edge.cub.Exp | List of 14 |
| edge.MLR.Full | List of 12 |

```
its=m +no_defs
names      :  elevation,        twi,       radK, landsat_b3, landsat
_b4
min values : 181.420395,   9.801188,  -0.009290,  18.864470,  13.184
220
max values :  960.10742,   23.89634,    5.16667,  170.51700,  154.57
581

> DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
+                     method = "simple")
> DSM_data <- as.data.frame(DSM_data)
> |
```



```
names      :  elevation,        twi,       radK, landsat_b3, landsat
_b4
min values : 181.420395,   9.801188,  -0.009290,  18.864470,  13.184
220
max values :  960.10742,   23.89634,    5.16667,  170.51700,  154.57
581

> DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
+                     method = "simple")
> DSM_data <- as.data.frame(DSM_data)
> View(DSM_data)
> |
```



```
covStack

#intersecting between the soil observations and covariate layers

DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
                    method = "simple")


#export the soil and covariate data intersect object to file for
#First we convert the spatial object to a dataframe, then export

DSM_data <- as.data.frame(DSM_data)
write.table(DSM_data, "edgeroiSoilCovariates_C.TXT",
            col.names = T, row.names = FALSE, sep = ",")
```

```
names      :  elevation,        twi,       radK, landsat_b3, landsat
_b4
min values : 181.420395,   9.801188,  -0.009290,  18.864470,  13.184
220
max values :  960.10742,   23.89634,    5.16667,  170.51700,  154.57
581

> DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
+                     method = "simple")
> DSM_data <- as.data.frame(DSM_data)
> View(DSM_data)
> |
```

Now, if we want to plot these points so these points or sampling points are plotted over this elevation raster layer. Now, remember that guys there are 5 different raster layers and we have to bind them together so that we can use them for future prediction, for all of these points from where these point data is available.

So, what happens sampling points are there so for the sampling points we have suppose they have collected, here you can see 341 observation they have collected from this region and they have measured the organic carbon say for 0 to 5 centimeter. Now, for this 341 observation we need all these the 5 variable values from these 5 raster layers. So, how to do that?

So, the way to do that is to overlay and stack them together 5 layers and then intercept the values of these 5 variables for those all 347 observation. So, we are going to use this stack function and this stacked and for all these 5 raster layers and then we are going to name them as covstack that is stacked covariate and let us see how it looks like.

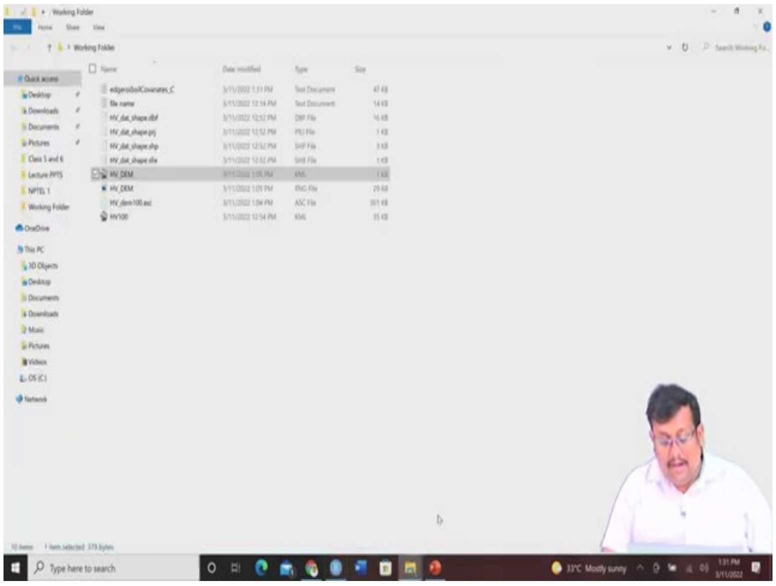
So, one thing you can see that this covstack which is also created the large raster stack so here you can see the same dimension 400 number of rows, 500 number of columns and 230800 number of cells. However, here one more field has been added that is 5 layers. So, now we have stacked all these different, 5 different covariates. And now we have point data, we have extracted stacked covariate data, so we want to combine them together by using the extract function or in other words we have these 5 layers stacked together, how to extract the values of those 5 variables from those all 341 observations.

So, for that we are going to use this extract function. So, you can see we are extracting from the stacked covariates and based on the 341 observation which are there in the edgeroi spline carbon and our method is simple extraction. So, we are doing that, so the data has been created, so DSM underscore data, it is also spatial points data frame right now.

But for doing as I have mentioned for doing different modeling things we need to convert it to the normal data frame and so here from the DSM data we are converting into the normal data frame so for that the function is as dot data dot frame. So, we are doing that and then we are writing the table.

So, now you can see here DSM data 341 observation of 15 variables it is a very, it is a normal tabular format. In this normal tabular format you can see all these first variables which are there in the edgeroi spline carbon apart from that you can see elevation, twi, radk, landsat b3, landsat b4 have been combined so that is the resultant of these ms, this extraction and you can write this whole file as a txt file by using write dot table function.

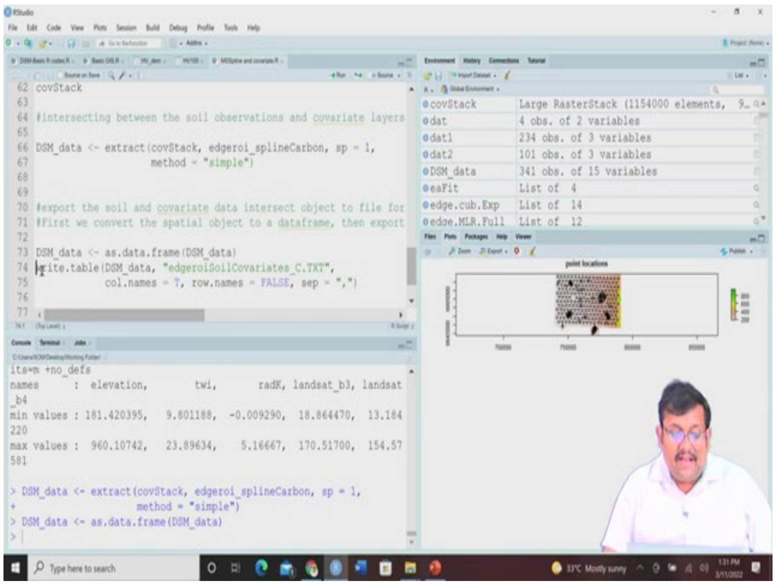
(Refer Slide Time: 33:00)



| Name | Date modified | Type | Size |
|---|---|---|---|
| edgeroiSoilCovariates_C | 3/11/2022 1:31 PM | Text Document | 47 KB |
| file name | 3/11/2022 12:14 PM | Text Document | 14 KB |
| HV_dat_shape.dbf | 3/11/2022 12:52 PM | DBF File | 16 KB |
| HV_dat_shape.prj | 3/11/2022 12:52 PM | PRJ File | 1 KB |
| HV_dat_shape.shp | 3/11/2022 12:52 PM | SHP File | 3 KB |
| HV_dat_shape.shx | 3/11/2022 12:52 PM | SHX File | 1 KB |
| HV_DEM | 3/11/2022 1:05 PM | KML | 1 KB |
| HV_DEM | 3/11/2022 1:05 PM | PNG File | 29 KB |
| HV_dem100.asc | 3/11/2022 1:04 PM | ASC File | 301 KB |
| HV100 | 3/11/2022 12:54 PM | KML | 35 KB |

And if you go to your working folder you will see that edgeroi soil covariates underscore C has been already as a text file have been saved. So, you can use this file for further operation and you can do different types of modeling which we are going to see in our upcoming lectures.

(Refer Slide Time: 33:21)



```
covStack

#intersecting between the soil observations and covariate layers

DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
                    method = "simple")


#export the soil and covariate data intersect object to file for
#First we convert the spatial object to a dataframe, then export

DSM_data <- as.data.frame(DSM_data)
write.table(DSM_data, "edgeroiSoilCovariates_C.TXT",
            col.names = T, row.names = FALSE, sep = ",")
```

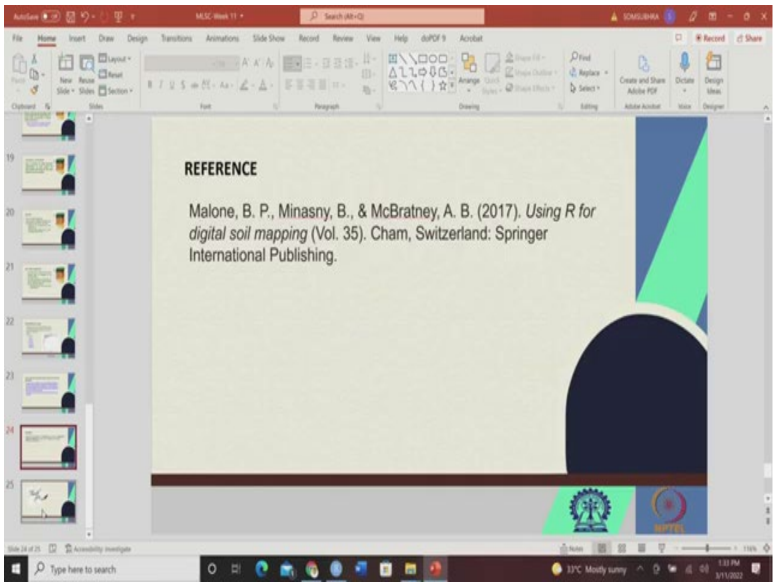
```
its=m +no_defs
names       :  elevation,        twi,       radK, landsat_b3, landsat
_b4
min values : 181.420395,   9.801188,  -0.009290,  18.864470,  13.184
220
max values :  960.10742,   23.89634,    5.16667,  170.51700,  154.57
581

> DSM_data <- extract(covStack, edgeroi_splineCarbon, sp = 1,
+                     method = "simple")
> DSM_data <- as.data.frame(DSM_data)
> 
```

```
covStack        Large RasterStack (1154000 elements, 9…
dat             4 obs. of 2 variables
dat1            234 obs. of 3 variables
dat2            101 obs. of 3 variables
DSM_data        341 obs. of 15 variables
eaFit           List of 4
edge.cub.Exp    List of 14
edge.MLR.Full   List of 12
```

So, these covariates interception in order to carry out the DSM in terms of evaluating the significance of environmental variables in explaining the spatial variation of the target soil variables under investigation we need to link both the sets of data together and extract the values of the covariates at the location of the soil point data.

And the first task of course is to bring to our working environment and some soil point data and then you bring the covariate data, stack them together using the stacked function and then from the stack layer you extract those values using the extract function. And finally you convert them into the normal data frame for doing all the operations. So, this is how you prepare a final data set for different types of modeling, this will be our starting point from different types of modeling, I will show you how to do different types of quantitative modeling in our next lecture.

(Refer Slide Time: 34:23)



REFERENCE

Malone, B. P., Minasny, B., & McBratney, A. B. (2017). *Using R for digital soil mapping* (Vol. 35). Cham, Switzerland: Springer International Publishing.

This is a reference and I hope that guys you have learnt something new in this lecture. We will be sharing you all these codes and please perform these operations, please run this codes to boost your confidence and see how these codes can be utilized for different types of DSM applications. So, thank you for joining and let us meet in our next lecture where we will continue from here. Thank you very much.