Data processing for DSM

Bas Kempen

Version 2.0, 16 May 2018

This tutorial shows how to process and prepare soil point and covariate data for digital soil mapping (DSM) and to generate a regresssion matrix that is used to calibrate a statistical DSM model. The processing steps are illustrated with a sample dataset from Macedonia.

0 Setting up an R session

Before starting scripting, it is recommended to empty the computer memory, clean up the workspace and load the required libraries.

```
# empty memory and workspace
gc()
rm(list=ls())

# load libraries
require(sp)
require(raster)
require(plyr)
require(caret)
```

1 Processing soil sample data

1.1 Read the soil property point data

The Macedonia soil sample data is stored in two tables. One for the site data, *profiles*, and another for the soil horizon data, *soil_horizons*. We start by reading these tables in and inspecting the content. Then we join the horizon data to the site data based on the common attribute `ProfID`, using the function **join()** from the **plyr** package.

```r
# read profile and horizon data
d_prof <- read.csv("./Data/soil_data/profiles.csv", header = TRUE)
d_hor <- read.csv("./Data/soil_data/soil_horizons.csv", header = TRUE)

# inspect profile data
str(d_prof)
head(d_prof)

# inspect horizon data
str(d_hor)
head(d_hor)
summary(d_hor)

# join the two tables
d1 <- join(d_prof, d_hor, by='ProfID', type = "left")

# show first six rows of the data.frame
head(d1)
```

1.2 Cleaning-up the soil sample data

The next step is to clean-up the data table. First, there are several columns with data that we do not need for our analyses; these can be removed. Second, the summary statistics show that some sampling sites have a soil organic carbon (SOC) value of -9999, which indicates NoData.

```r
# remove some unnecessary columns
d1 <- d1[,-c(2,5)]

# remove NoData values, which is the value -9999 in this case
d1 <- subset(d1, d1$SOC != -9999)
```

Third, displaying the first six rows of the dataset revealed that some layers are duplicated (first and second row, third and fourth row, and the fifth and sixth row). There might be more duplications in the dataset. We can identify duplicated records using the function **duplicated**. Here we use data in the columns `ProfID`, `DepthFrom` and `DepthTo` columns.

```r
# Check if there are duplicate rows present in the dataframe
dum <- duplicated(d1[c(1,6,7)])
summary(dum)
```

```
##    Mode   FALSE    TRUE    NA's
## logical   10075     923       0
```

The output shows there are 923 duplicated entries in the soil property table. We will use the output of the **duplicated** function to excluded these from the dataset, and then save the processed dataset in an **.rda** output file.

```r
# remove duplicated observations based on similar depth and profile id
d1 <-  d1[!duplicated(d1[c(1,6,7)]),]

# save
save(d1, file = "pointData.rda")
```

Clean-up the environment by removing unnecessary objects.

1.3 Deriving soil property values for the top 30 cm

Our target variable for mapping is the soil organic carbon (SOC) content. Here we choose to map the SOC content for the 0 – 30-cm layer. The next step in the data preparation process is to compute the SOC content for this depth interval from the soil horizon layers. This can be done by fitting splines or by calculating the weighted average of the horizons that fall within the depht layer. Here we will use weighted averaging.

A function for weighted averaging is prepared and stored in R script **funComputeSoilProperty.R**. Note that this is not a generic function. In order to work, it requires specific column names. The soil profile identifier should be stored in a column `ProfID`, the X and Y coordinates in columns `X_coord` and `Y_coord`, and the sampling depths in columns `DepthFrom` (in cm) and `DepthTo` (in cm).

Start by import this script into your workspace and run this script. You will see that the function is added to your *Global Environment list*.

```
source("funComputeSoilProperty.R")
```

The **computeSoilProperty** function needs 5 inputs:

- the soil property that need to be averaged
- the data frame that stores the soil data
- the upper depth (cm) of the soil layer of interest
- the lower depth (cm) of the soil layer of interest
- a thickness criterion (cm)

The thickness criterion specifies the minimum thickness within the depht interval of interest for which there should be data available to calculate the weighted average. For instance, if we take as thickness criterion 10 cm to calculate the weighted average for a 30-cm thick layer, then this means that if we have soil data for at least 10 cm within the 30 cm interval we will use these data to compute the weighted average, even if these data cover only 10 cm of the interval.

We run the function for SOC and use 25 cm as the thickness criterion.

```
# run the function for weighted averaging soil properties
d2 <- computeSoilProperty("SOC", d1, 0, 30, 25)
```

Finally, we exclude duplicate sampling sites (i.e. sampling sites that have identical coordinates; coordinates are in columns 2 and 3), and then save the processed soil sample data in a new `.rda` file.

```
# remove observations that have identical coordinates
d <- d2[!duplicated(d2[c(2,3)]),]

# save dataset
save(d, file = "pointData_SOC.rda")
```

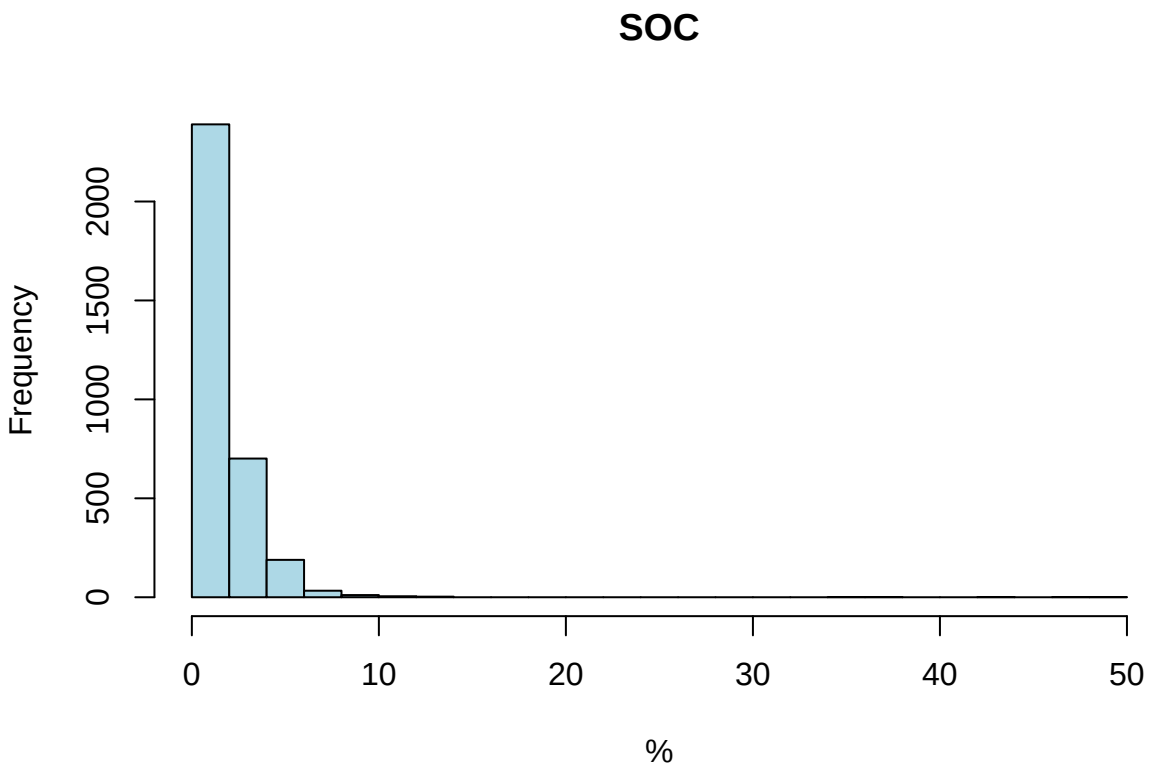
1.4 Inspect point data

Inspect the SOC sample data by calculating summary statistics, plotting a histogram and making a spatial point plot.

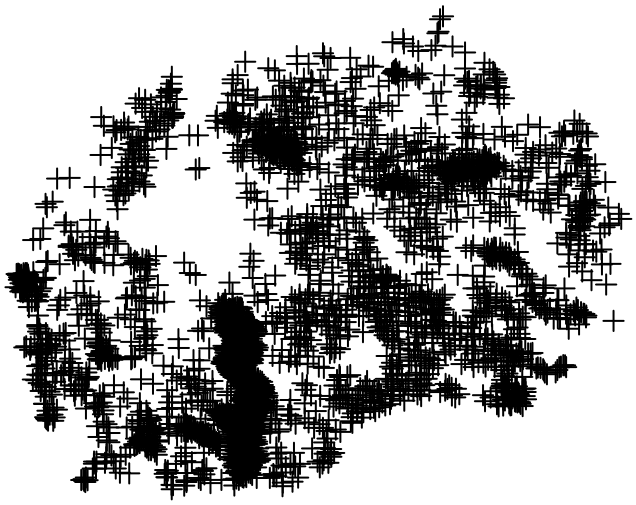
```
# summary statistics
summary(d$SOC)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.450   1.875   2.150  48.030
```

```
# histogram
hist(d$SOC, col = "lightblue", xlab="%", main="SOC", breaks=20)
```




```
# spatial plot (first convert to SpatialPointsDataFrame)
coordinates(d) <- ~ X_coord+Y_coord
plot(d)
```

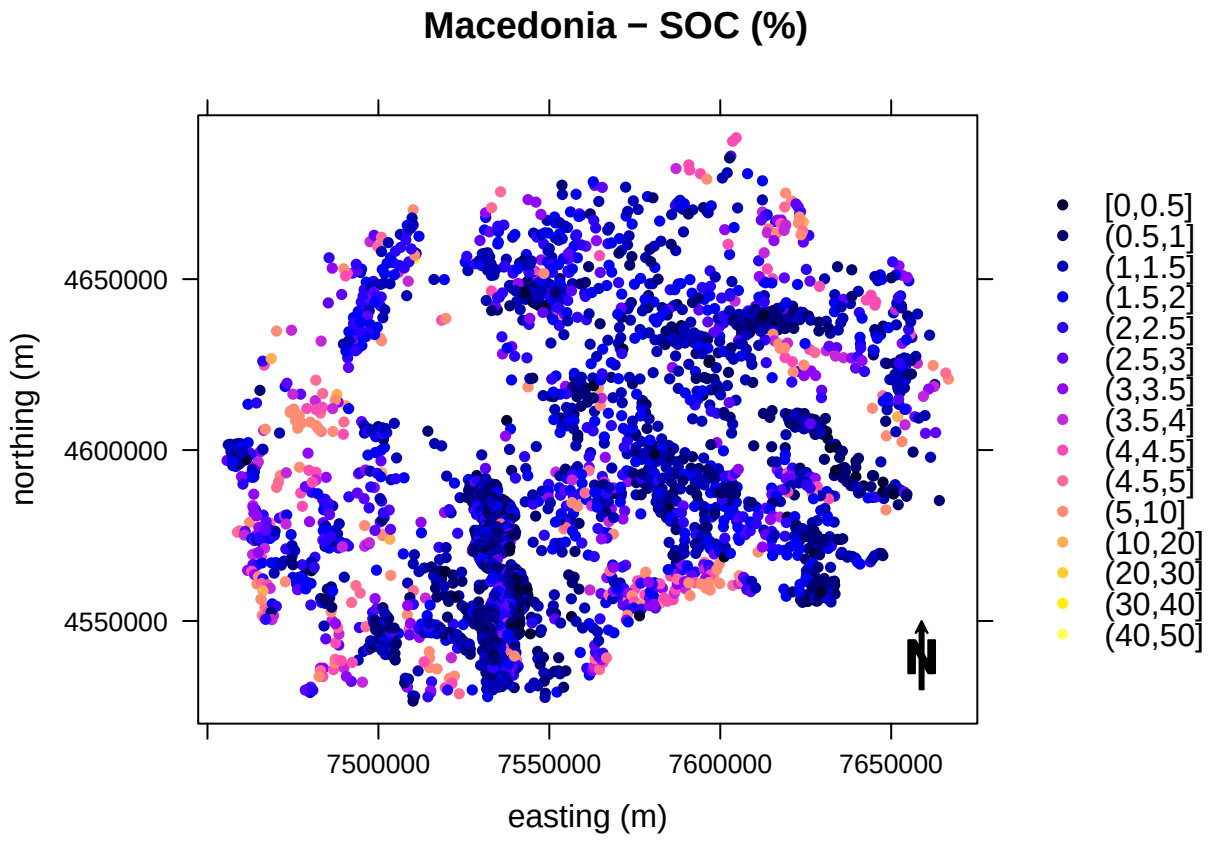


This quick inspection of the point data does not reveal any issues with the data such as unrealistic values, or sampling sites outside the Macedonian border. Also, the summary statistics show that there is a SOC value for all sampling sites. If one or more sites would have had missing SOC data then these could be removed from the dataset using the following command: `d <- d[complete.cases(d[,4]),]` (SOC data is stored in the fourth column).

We can make a nicer plot that shows the SOC values with the **spplot()** function of the **sp** package. We first create an object that stores the specifications of the north arrow maker, and a second object that defines the breaks for the color legend (to enhance contrast). These two objects are used as inputs for the plotting function. The arrow is passed to the `sp.layout` argument and the color breaks to the `cuts` argument. Try to figure out what the function arguments of `spplot` mean by using the help: `?spplot`.

```
# define arrow marker
arrow <- list(
  "SpatialPolygonsRescale",
  layout.north.arrow(),
  offset = c(7655000,4530000),
  scale = 20000
)

# define lengend breaks
breaks <- c(seq(0,5,0.5),seq(10,50,10))

# plot
spplot(d, c("SOC"),
  cuts = breaks,
  cex = 0.7,
```

```
  sp.layout = list(arrow),
  key.space = "right",
  main = "Macedonia - SOC (%)",
  xlab = "easting (m)",
  ylab = "northing (m)",
  scales = list(draw=TRUE)
)
```




```
# convert back to data.frame
d <- as(d, Class = "data.frame")
```

2 Creating a covariate stack

2.1 Reading the covariate layers

Covariate data are typcially GIS layers of biophysical land surface properties that are used in digital soil mapping to predict the soil property of interest across the full extent of the prediction area, using the soil property data from the sampling locations.

For Macedonia, a set of 59 covariate layers is prepared in GeoTiff format and have a spatial resolution of 1 km x 1 km. We start by creating an object that stores the file names of these using the **list.files()** function. We then use the **stack()** function of the **raster** package. Note we can only use the `stack` function if all layers have the same coordinate system, spatial resolution and exactly the same extent.

```
# list the files in the covariate folder
cov.lst <- list.files(path = "./Data/covariates/1km/wgs84/stack1", pattern =".tif")

# explore the set
head(cov.lst )
```

```
## [1] "B02CHE3.tif" "B04CHE3.tif" "B07CHE3.tif" "B13CHE3.tif" "B14CHE3.tif"
## [6] "BARL10.tif"
```

```
length(cov.lst )
```

```
## [1] 59
```

```
# create raster stack
r1 <- stack(paste0("./Data/covariates/1km/wgs84/stack1/", cov.lst ))

# check object class
class(r1)
```

```
## [1] "RasterStack"
## attr(,"package")
## [1] "raster"
```

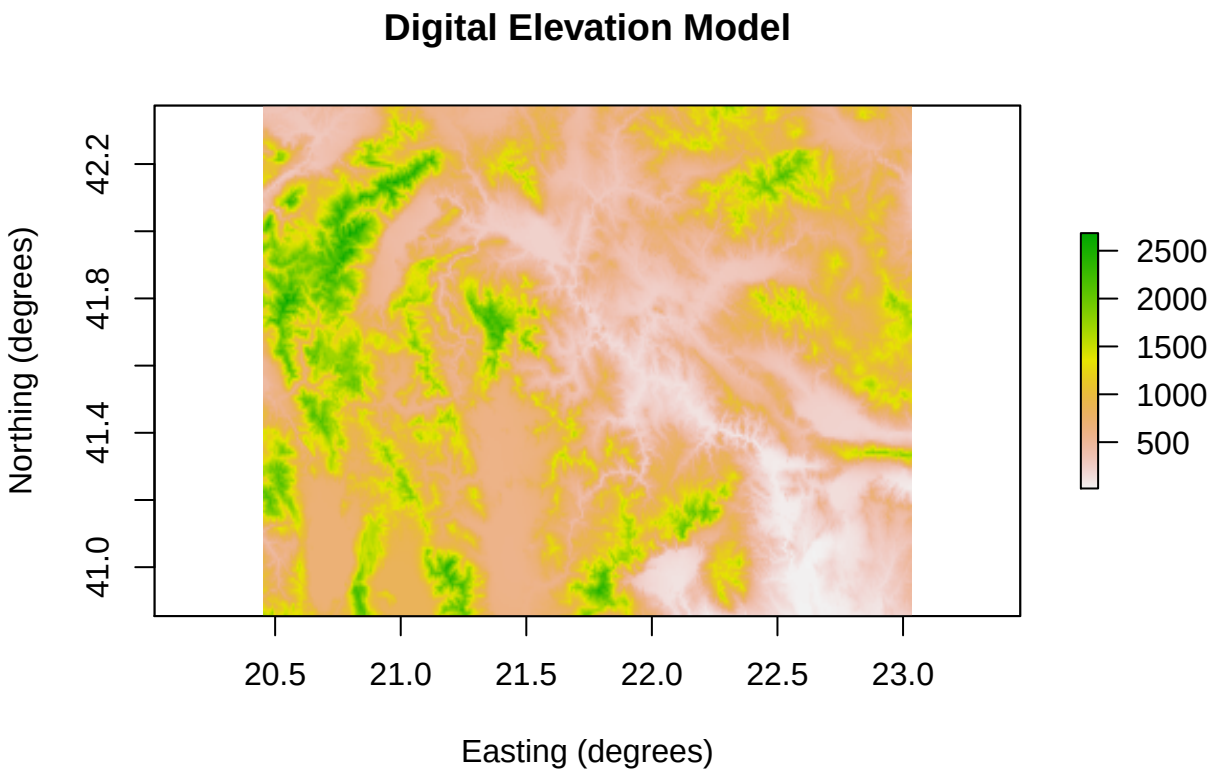
```
# show dimensions
dim(r1)
```

```
## [1] 182 310  59
```

The stack is saved as a **RasterStack** class. We see that the raster stack has rows, 310 columns and 59 layers.

Create a plot of one of the covariates (e.g. elevation).

```
plot(r1[["DEMENV5"]], main = "Digital Elevation Model", xlab = "Easting (degrees)", ylab = "Northing (d
```



2.2 Clipping and projecting the covariate stack

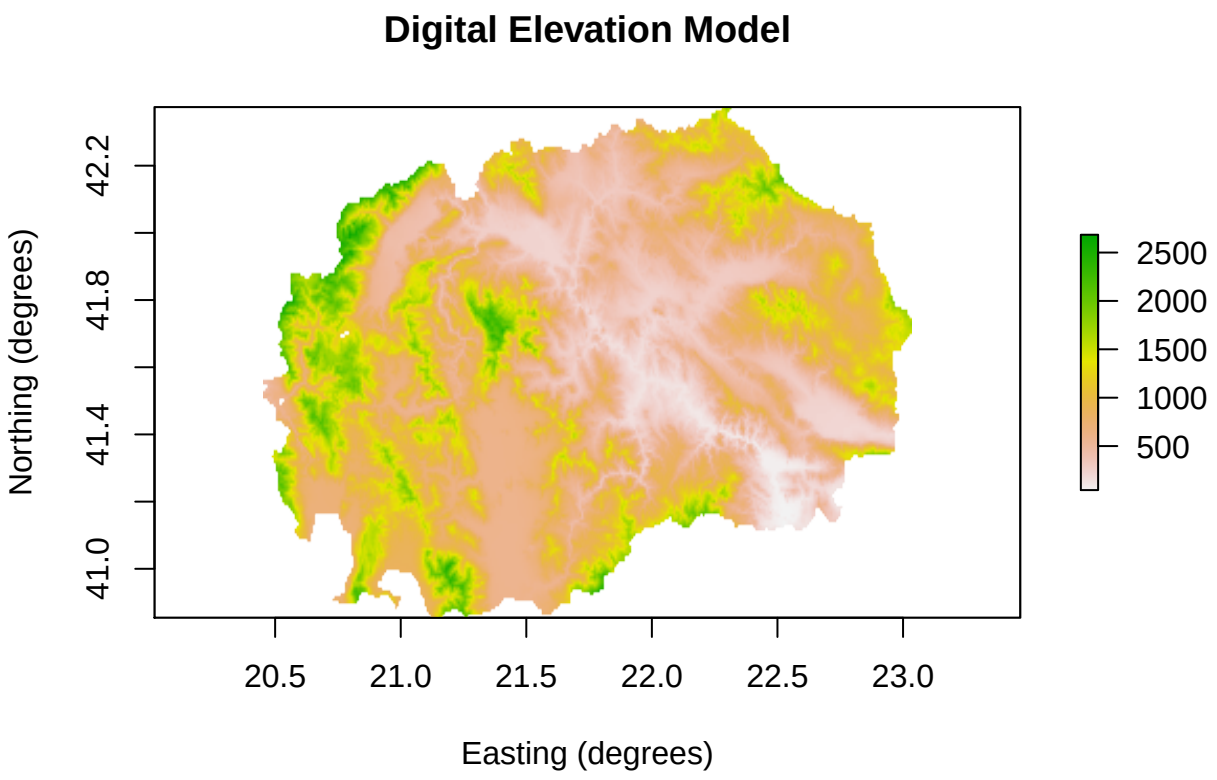
The covariate layers were extracted from global layers using a rectangular bounding box based on the country extent. For prediction, we want to predict for the country so we would like to exclude the pixels that fall outside the country border. In addition we would like to exclude the pixels representing non-soil areas such as built-up areas an water bodies, i.e. we want to mask out the area which is not of our interest. For this purpose we use a **mask** layer which is a raster file that follows the boundary of the country. Masking can be done with the **mask()** function of the **raster** package. Note, masking can take some time, depending on the specifications of your computer.

```
# read mask
m <- raster(paste0("./Data/covariates/1km/wgs84/mask/", "mask.tif"))

# mask the stack
r2 <- mask(r1, m)

# inspect the stack object
r2
```

```
## class       : RasterBrick
## dimensions  : 182, 310, 56420, 59  (nrow, ncol, ncell, nlayers)
## resolution  : 0.008327968, 0.008353187  (x, y)
## extent      : 20.45242, 23.03409, 40.8542, 42.37448  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       :     B02CHE3,     B04CHE3,     B07CHE3,     B13CHE3,     B14CHE3,      BARL10,     CHAG
```

```
## min values  :    20.25891,   536.16620,    36.87946,    44.79326,    21.44841,     0.00000,     0.000
## max values  :    21.94757,   803.28107,    45.40100,   169.11386,    61.78397,   100.00000,   255.000
```

```
# plot one covariate layer
plot(r2[["DEMENV5"]], main = "Digital Elevation Model", xlab = "Easting (degrees)", ylab = "Northing (de
```



Now we check the projection of the raster stack with the **proj4string()** function of the **sp** package.

```
# check projection
proj4string(r2)
```

```
## [1] "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
```

The stack is projected in the WGS84 projection with unit in decimal degrees. The soil sample data is projected in a local projection Macedonia State Coordinate System zone 7. The `EPSG` code for this coordinate system is 6316. You can reproject the using the **projectRaster()** function of the **raster** package. Alternatively, the **spTransfrom()** function of the **sp** package can be used. For this, the `r2` object must first be converted to an sp object of class `SpatialPixelsDataFrame` or `SpatialGridDataFrame` (Try it out!).

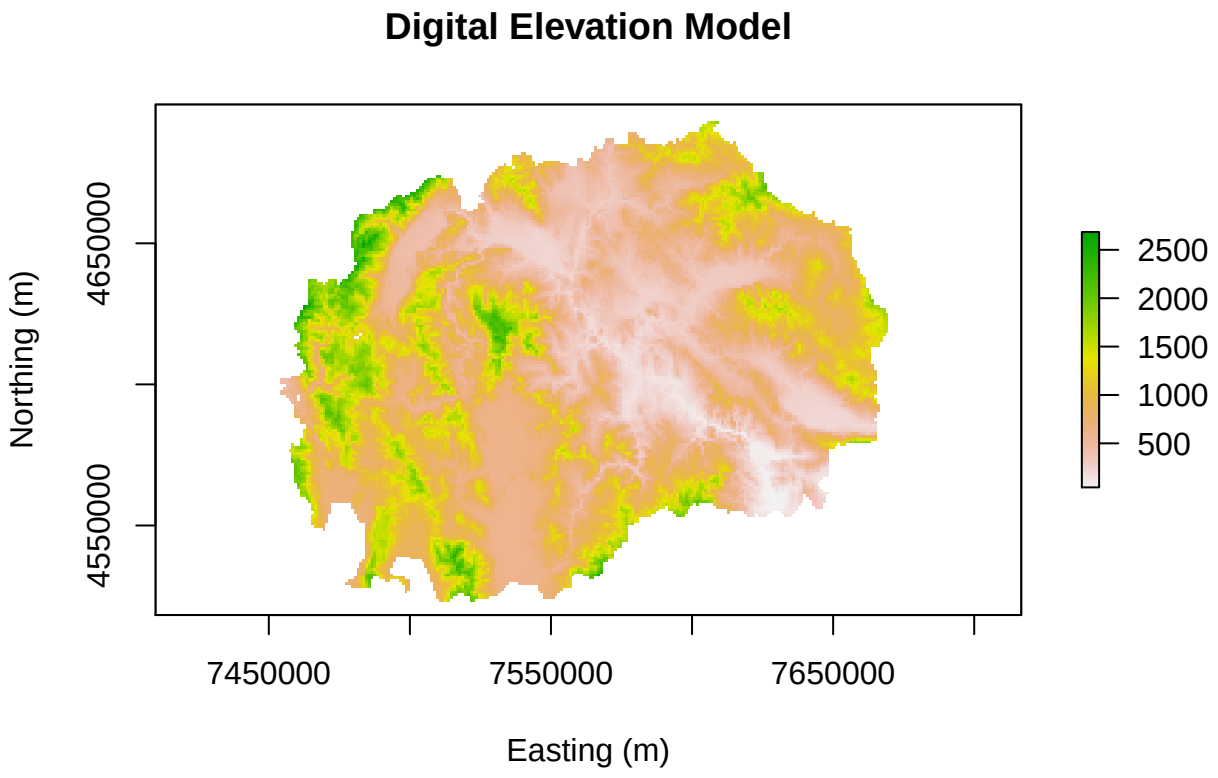
You might consider to make two different stacks with reprojected data. One for numeric covariates using a bilinear interpolation and another for the categorical covariates using a neirest neighbour interpolation. See **?projectRaster** for more information. Here we use nearest neighbor interpolation for resampling.

```
# reproject the stack
r3 <- projectRaster(r2, crs = CRS("+init=epsg:6316"), res = 1000, method = 'ngb')

# inspect the output
r3
```

```
## class       : RasterBrick
```

```
## dimensions  : 181, 228, 41268, 59  (nrow, ncol, ncell, nlayers)
## resolution  : 1000, 1000  (x, y)
## extent      : 7449266, 7677266, 4518255, 4699255  (xmin, xmax, ymin, ymax)
## coord. ref. : +init=epsg:6316 +proj=tmerc +lat_0=0 +lon_0=21 +k=0.9999 +x_0=7500000 +y_0=0 +ellps=bes
## data source : in memory
## names       :     B02CHE3,     B04CHE3,     B07CHE3,     B13CHE3,     B14CHE3,      BARL10,      CHAGS
## min values  :    20.25891,   536.16620,    36.87946,    44.83688,    21.44841,     0.00000,     0.000
## max values  :    21.94757,   803.28107,    45.40100,   169.11386,    61.47525,   100.00000,   255.000
```

```
# plot one covariate layer
plot(r3[["DEMENV5"]], main = "Digital Elevation Model", xlab = "Easting (m)", ylab = "Northing (m)")
```



2.3 Processing covariate layers

Since processing and working with categorical layers with the `raster` package is somewhat cumbersome, we do the processing on a `data.frame` object. Converting converting the `rasterBrick` object to a `SpatialGridDataFrame` object. Then we further convert this object to a `data.frame`.

```
# convert to SpatialGridDataFrame
r4 <- as(r3, "SpatialGridDataFrame")

# further convert to data.frame
r4 <- as(r4, Class = "data.frame")
```

2.3.1 Converting to factor

For modelling it is important that the data is correctly defined as either numeric (*num*), integer (*int*) or categorical (*factor*). There is one categorical covarite in the stack (**LCEE10**), this is a land cover class layer.

The classes of a categorical covariate are represented with numbers in the GeoTiff layer. So when such layer is read in R, the data values are stored as numeric data. Before modelling we should convert the land cover covariate to factor. First, we list the categorical covariates (here we have only one) in one object. Then we use a for-loop which converts each of the listed covariates from *numeric* to *factor* data using the function **as.factor()**.

```
# list categorical covariates
cat.list <- c("LCEE10")

# check data type
class(r4[,cat.list])
summary(r4[,cat.list])

# convert to factor in a loop
for(i in 1:length(cat.list)){
  r4[,cat.list[i]] <- as.factor(r4[,cat.list[i]])
}

# check again; note the variables have been converted to factor
class(r4[,cat.list])
summary(r4[,cat.list])
```

2.3.2 Creating binary variables

Using categorical covariates in a statistical model requires that one has an observation (sampling site) located in each class (e.g. soil, land cover, parent material) of the covariate. When classes cover only a very small part of the mapping area it might happen that this will not be the case. The **summary** function showed us that several classes cover only one or only a handful of pixels. To avoid problems with modelling and prediction later on, we want to get rid of these classes. We can either do this by combining classes that cover only a small area with other classes in a sensible manner, or ddrop the class from the layer through converting the land cover layer to n binary layers [0,1 - presence/absence] with n equal to the number of land cover classes.

In this section we follow the latter approach since this is more straightforward. In the next section we show how we can drop layers that show very little or no variation in data values.

(**Note**: the land cover covariate is derived from the ESA land cover map 2010. Documentation of this product (including a legend of the classes) can be found here. Should one wish to reclassify classes (using for instance the **ifelse()** function) then one could use this resource to decide on how to combine classess (e.g. 61 with 60).)

Conversion from a multiclass variable to a set of binary variables is done in a few steps. We start with applying the **model.matrix()** function to the categorical variable (here land cover). This function creates a design matrix by expanding a multiclass factor variable to a set of binary variables. The outcome is a `matrix` that we convert to a `data.frame` in a next step. Then we need to convert the numeric [0,1] output values to factor. To do this for all in one go we use the **lapply** fundtion. The output is a `list` object that we again convert to a `data.frame` object. Finally we append the set of binary variables to the covariate stack.

We implement these processing steps in a loop so that we can convert multiple categorical variables in one go (note that in the example here we have only one `LCEE10` so the loop runs once).

```
# convert categorical covariates to binary layers
for(i in 1:length(cat.list)){

  # create design matrix
  dum <- model.matrix(as.formula(paste0("~",cat.list[i],"+0")), r4)

  # convert to data.frame
  dum <- as.data.frame(dum)

  # convert to factor
  dum <- lapply(dum, FUN = factor)

  # convert to data.frame
  dum <- as.data.frame(dum)

  # append to covariate stack
  r5 <- cbind(r4, dum)
}
```

Now we remove the original categorical covariate(s).

```
# exclude original categorical covariates
r5 <- r5[,!names(r5) %in% cat.list]
```

2.3.3 (Near-)zero variance analysis

The next step in the covariate processing workflow is to remove covariates for which the values show very little or no variation. Such covariates cannot be used for modelling. Instead of visually inspecting the individual layers we can use the **nearZeroVar()** function of the **caret** package that can detect which covariate layer have zero or near-zero variance.

```
# run nzv function
nzv <- nearZeroVar(r5, saveMetrics = TRUE)

# inspect the output
nzv
```

```
##               freqRatio percentUnique zeroVar   nzv
## B02CHE3        1.000000  89.792619461   FALSE FALSE
## B04CHE3        1.000000  97.969563160   FALSE FALSE
## B07CHE3        1.000000  94.957419965   FALSE FALSE
## B13CHE3        1.500000  99.794985018   FALSE FALSE
## B14CHE3        1.000000  99.467749566   FALSE FALSE
## BARL10         5.715235   0.283866898   FALSE FALSE
## CHAGSW7      354.873239   0.185302003   FALSE  TRUE
## CMCF5avg       1.333333  69.314776849   FALSE FALSE
## CRDMRG5        1.082707   8.090206592   FALSE FALSE
## CRUMRG5        1.058333   8.563318089   FALSE FALSE
## CRVMRG5        1.000000   7.057246491   FALSE FALSE
## DEMENV5        1.000000   8.646112601   FALSE FALSE
## DV2MRG5        1.010471   1.734742154   FALSE FALSE
## DVMMRG5        1.041451   1.604636493   FALSE FALSE
## ENTENV3        1.000000  19.117647059   FALSE FALSE
## ESMOD5avg      1.052632  19.681438259   FALSE FALSE
## EVEENV3        1.023810   4.415707302   FALSE FALSE
```

```
## EXMOD5avg      1.222222  46.995741997   FALSE FALSE
## EXTGSW7      182.695652   0.011827787   FALSE  TRUE
## F01USG5        2.710900   0.007885192   FALSE FALSE
## F02USG5        8.474785   0.007885192   FALSE FALSE
## F03USG5     1584.250000   0.007885192   FALSE  TRUE
## F04USG5        3.492384   0.007885192   FALSE FALSE
## F05USG5        7.686301   0.007885192   FALSE FALSE
## F06USG5        3.711871   0.007885192   FALSE FALSE
## F07USG5       13.273495   0.007885192   FALSE FALSE
## IMOD4avg       1.000000  40.249960574   FALSE FALSE
## MANMCF5        1.040000  13.728118593   FALSE FALSE
## MAXENV3        1.027027   5.472322977   FALSE FALSE
## MMOD4avg       1.100000  50.208957578   FALSE FALSE
## MRNMRG5        3.182584   0.658413499   FALSE FALSE
## NEGMRG5        1.012915   1.470588235   FALSE FALSE
## NIRL00         1.034048   0.437628134   FALSE FALSE
## NIRL14         1.036517   0.512537455   FALSE FALSE
## NMOD3avg       1.048443   0.768806182   FALSE FALSE
## NMSD3avg       1.025692   1.009304526   FALSE FALSE
## OCCGSW7     1481.000000   0.264153919   FALSE  TRUE
## PCHE3avg       1.000000  99.617568207   FALSE FALSE
## POSMRG5        1.003289   1.348367765   FALSE FALSE
## PRSCHE3        1.500000  99.613625611   FALSE FALSE
## QUAUEA3        0.000000   0.003942596    TRUE  TRUE
## RANENV3        1.000000   8.031067655   FALSE FALSE
## REDL00         1.038820   0.445513326   FALSE FALSE
## REDL14         1.105815   0.413972560   FALSE FALSE
## SESA4avg       1.024755   1.127582400   FALSE FALSE
## SLPMRG5        1.055556   0.279924302   FALSE FALSE
## SW1L00         1.042510   0.551963413   FALSE FALSE
## SW1L14         1.001490   0.571676392   FALSE FALSE
## SW2L00         1.142261   0.311465069   FALSE FALSE
## SW2L14         1.089316   0.268096515   FALSE FALSE
## TMDMOD3        1.121790   0.078851916   FALSE FALSE
## TMNMOD3        1.404630   0.070966724   FALSE FALSE
## TPIMRG5        1.050314   7.494874625   FALSE FALSE
## TREL10        21.444089   0.354833622   FALSE  TRUE
## TWIMRG5        1.041667   0.327235452   FALSE FALSE
## VBFMRG5        9.012483   1.845134837   FALSE FALSE
## VDPMRG5        5.200000  41.740261788   FALSE FALSE
## VW1MOD1avg     1.029101   0.658413499   FALSE FALSE
## s1             1.000000   0.847658098   FALSE FALSE
## s2             1.000000   0.670241287   FALSE FALSE
## LCEE1010       2.066248   0.007885192   FALSE FALSE
## LCEE1011     873.620690   0.007885192   FALSE  TRUE
## LCEE1012     100.863454   0.007885192   FALSE  TRUE
## LCEE1020      27.954338   0.007885192   FALSE  TRUE
## LCEE1030      23.697176   0.007885192   FALSE  TRUE
## LCEE1040      20.048963   0.007885192   FALSE  TRUE
## LCEE1060       1.734368   0.007885192   FALSE FALSE
## LCEE1061   25363.000000   0.007885192   FALSE  TRUE
## LCEE1070      55.489978   0.007885192   FALSE  TRUE
## LCEE1090      64.371134   0.007885192   FALSE  TRUE
## LCEE10100     19.554295   0.007885192   FALSE  TRUE
```

```
## LCEE10110  12681.000000   0.007885192   FALSE  TRUE
## LCEE10120     99.252964   0.007885192   FALSE  TRUE
## LCEE10130     19.892916   0.007885192   FALSE  TRUE
## LCEE10150    421.733333   0.007885192   FALSE  TRUE
## LCEE10180   5071.800000   0.007885192   FALSE  TRUE
## LCEE10190     34.081604   0.007885192   FALSE  TRUE
## LCEE10200   4226.333333   0.007885192   FALSE  TRUE
## LCEE10210    265.989474   0.007885192   FALSE  TRUE
```

```
# zero variance covariates
summary(nzv$zeroVar)
```

```
##    Mode   FALSE    TRUE    NA's
## logical      78       1       0
```

```
# near-zero variance covariates
summary(nzv$nzv)
```

```
##    Mode   FALSE    TRUE    NA's
## logical      56      23       0
```

The analysis shows that there are 23 near-zero variance covariate layers and 1 zero variance. We will drop these from our stack (the output of the near-zero variance assessment is stored in the fourth column).

```
# drop nzv layers
r6 <- r5[,!nzv[,4]]
```

2.3.4 Removing incomplete data pixels

In the last covariate stack processing step we will generate a statistical summary of each covariate layer to check if there are any unrealistic or suspicious values.

```
# summary statistics
summary(r6)
```

```
##     B02CHE3         B04CHE3         B07CHE3         B13CHE3
##  Min.   :20.26   Min.   :536.2   Min.   :36.88   Min.   : 44.84
##  1st Qu.:20.99   1st Qu.:745.6   1st Qu.:43.36   1st Qu.: 61.58
##  Median :21.26   Median :764.4   Median :44.20   Median : 72.75
##  Mean   :21.22   Mean   :750.4   Mean   :43.67   Mean   : 79.06
##  3rd Qu.:21.46   3rd Qu.:775.5   3rd Qu.:44.55   3rd Qu.: 89.85
##  Max.   :21.95   Max.   :803.3   Max.   :45.40   Max.   :169.11
##
##     B14CHE3          BARL10          CMCF5avg       CRDMRG5
##  Min.   :21.45   Min.   :  0.000   Min.   :4493   Min.   :-2412.0
##  1st Qu.:29.88   1st Qu.:  0.000   1st Qu.:5226   1st Qu.: -416.2
##  Median :34.59   Median :  1.000   Median :5831   Median : -171.0
##  Mean   :35.54   Mean   :  4.915   Mean   :5869   Mean   : -268.7
##  3rd Qu.:40.23   3rd Qu.:  7.000   3rd Qu.:6360   3rd Qu.:  -32.0
##  Max.   :61.48   Max.   :100.000   Max.   :8794   Max.   : 1112.0
##
##     CRUMRG5          CRVMRG5          DEMENV5          DV2MRG5
##  Min.   :-1091.0   Min.   :-2363.0   Min.   :  45.0   Min.   :-200.000
##  1st Qu.:  -39.0   1st Qu.: -396.0   1st Qu.: 495.0   1st Qu.: -52.000
##  Median :   79.0   Median : -174.0   Median : 758.0   Median : -14.000
##  Mean   :  190.2   Mean   : -261.5   Mean   : 835.9   Mean   :  -4.475
##  3rd Qu.:  370.0   3rd Qu.:  -45.0   3rd Qu.:1109.0   3rd Qu.:  38.000
```

```
##  Max.   : 3109.0   Max.   :  852.0   Max.   :2684.0   Max.   : 371.000
##
##     DVMMRG5            ENTENV3        ESMOD5avg         EVEENV3
##  Min.   :-185.000   Min.   :18999   Min.   : 230.0   Min.   :8132
##  1st Qu.: -48.000   1st Qu.:27170   1st Qu.: 582.8   1st Qu.:9303
##  Median : -12.000   Median :27956   Median : 688.0   Median :9428
##  Mean   :  -4.306   Mean   :27745   Mean   : 706.0   Mean   :9401
##  3rd Qu.:  36.000   3rd Qu.:28552   3rd Qu.: 809.2   3rd Qu.:9529
##  Max.   : 309.000   Max.   :29984   Max.   :1951.2   Max.   :9947
##                                     NA's   :65
##    EXMOD5avg         F01USG5          F02USG5          F04USG5
##  Min.   : 172.8   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00
##  1st Qu.:2680.8   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00
##  Median :3133.5   Median :  0.00   Median :  0.00   Median :  0.00
##  Mean   :3123.8   Mean   : 26.95   Mean   : 10.55   Mean   : 22.26
##  3rd Qu.:3576.5   3rd Qu.:100.00   3rd Qu.:  0.00   3rd Qu.:  0.00
##  Max.   :5046.0   Max.   :100.00   Max.   :100.00   Max.   :100.00
##  NA's   :65
##     F05USG5          F06USG5          F07USG5           IMOD4avg
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.000   Min.   : 331.8
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.000   1st Qu.: 647.1
##  Median :  0.00   Median :  0.00   Median :  0.000   Median : 797.7
##  Mean   : 11.51   Mean   : 21.22   Mean   :  7.006   Mean   : 902.6
##  3rd Qu.:  0.00   3rd Qu.:  0.00   3rd Qu.:  0.000   3rd Qu.: 967.2
##  Max.   :100.00   Max.   :100.00   Max.   :100.000   Max.   :4565.5
##                                                      NA's   :81
##     MANMCF5         MAXENV3        MMOD4avg         MRNMRG5
##  Min.   :4493   Min.   : 636   Min.   : 249.3   Min.   :  0.00
##  1st Qu.:5226   1st Qu.: 977   1st Qu.: 930.9   1st Qu.:  4.00
##  Median :5831   Median :1093   Median :1179.8   Median : 16.00
##  Mean   :5869   Mean   :1139   Mean   :1195.9   Mean   : 23.53
##  3rd Qu.:6360   3rd Qu.:1250   3rd Qu.:1451.8   3rd Qu.: 35.00
##  Max.   :8794   Max.   :3364   Max.   :2604.9   Max.   :194.00
##                                NA's   :81
##     NEGMRG5          NIRL00           NIRL14          NMOD3avg
##  Min.   :1142   Min.   :  0.00   Min.   :  0.00   Min.   :269.0
##  1st Qu.:1469   1st Qu.: 68.00   1st Qu.: 66.00   1st Qu.:278.7
##  Median :1522   Median : 76.00   Median : 75.00   Median :279.8
##  Mean   :1505   Mean   : 77.74   Mean   : 76.72   Mean   :279.5
##  3rd Qu.:1558   3rd Qu.: 86.00   3rd Qu.: 86.00   3rd Qu.:280.8
##  Max.   :1590   Max.   :145.00   Max.   :147.00   Max.   :285.8
##
##     NMSD3avg         PCHE3avg         POSMRG5         PRSCHE3
##  Min.   :16.50   Min.   : 34.94   Min.   :1189   Min.   : 419.2
##  1st Qu.:30.75   1st Qu.: 45.13   1st Qu.:1478   1st Qu.: 541.6
##  Median :31.92   Median : 52.02   Median :1517   Median : 624.2
##  Mean   :31.89   Mean   : 54.47   Mean   :1504   Mean   : 653.6
##  3rd Qu.:33.17   3rd Qu.: 61.05   3rd Qu.:1542   3rd Qu.: 732.6
##  Max.   :45.25   Max.   :104.18   Max.   :1596   Max.   :1250.2
##
##     RANENV3          REDL00           REDL14          SESA4avg
##  Min.   : 369   Min.   :  0.00   Min.   :  0.00   Min.   :  0.000
##  1st Qu.: 988   1st Qu.: 15.00   1st Qu.: 19.00   1st Qu.:  0.500
##  Median :1204   Median : 29.00   Median : 28.00   Median :  1.125
```

```
##  Mean   :1260   Mean   : 31.08   Mean   : 30.17   Mean   :  4.006
##  3rd Qu.:1472   3rd Qu.: 45.00   3rd Qu.: 39.00   3rd Qu.:  3.125
##  Max.   :3350   Max.   :161.00   Max.   :171.00   Max.   :254.000
##
##     SLPMRG5          SW1L00           SW1L14           SW2L00
##  Min.   : 0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 7.00   1st Qu.: 68.00   1st Qu.: 71.00   1st Qu.: 4.00
##  Median :14.00   Median : 82.00   Median : 81.00   Median : 9.00
##  Mean   :16.52   Mean   : 82.92   Mean   : 81.87   Mean   :11.43
##  3rd Qu.:25.00   3rd Qu.: 97.00   3rd Qu.: 93.00   3rd Qu.:17.00
##  Max.   :74.00   Max.   :170.00   Max.   :187.00   Max.   :89.00
##
##      SW2L14          TMDMOD3          TMNMOD3          TPIMRG5
##  Min.   :  0.0   Min.   :279.0   Min.   :269.0   Min.   :-1276.000
##  1st Qu.:  5.0   1st Qu.:288.0   1st Qu.:279.0   1st Qu.: -141.250
##  Median :  8.0   Median :291.0   Median :280.0   Median :  -16.000
##  Mean   : 10.5   Mean   :290.9   Mean   :279.5   Mean   :    3.588
##  3rd Qu.: 14.0   3rd Qu.:294.0   3rd Qu.:281.0   3rd Qu.:  133.250
##  Max.   :116.0   Max.   :298.0   Max.   :286.0   Max.   : 1737.000
##
##     TWIMRG5          VBFMRG5          VDPMRG5        VW1MOD1avg
##  Min.   : 53.00   Min.   :  0.00   Min.   :-7127   Min.   :123.5
##  1st Qu.: 70.00   1st Qu.:  0.00   1st Qu.: 2177   1st Qu.:136.8
##  Median : 79.00   Median :  0.00   Median : 4266   Median :148.0
##  Mean   : 80.42   Mean   : 49.77   Mean   : 4276   Mean   :146.6
##  3rd Qu.: 90.00   3rd Qu.: 36.00   3rd Qu.: 6376   3rd Qu.:155.7
##  Max.   :135.00   Max.   :498.00   Max.   :14544   Max.   :169.7
##
##        s1               s2          LCEE1010  LCEE1060
##  Min.   :7454766   Min.   :4523755   0:17092   0:16088
##  1st Qu.:7515766   1st Qu.:4576755   1: 8272   1: 9276
##  Median :7558766   Median :4607755
##  Mean   :7559925   Mean   :4607627
##  3rd Qu.:7603766   3rd Qu.:4638755
##  Max.   :7668766   Max.   :4692755
##
```

Next we exclude all pixels that do not have complete covariate data (we cannot predict for pixels with missing covariate values) with the **complete.cases()** function.

```
# exclude pixels that do not have complete covariate data
r <- r6[complete.cases(r6),]
```

Finally we reorganize the data order in the data.frame (first the coordinate columns, then the covariate data), clean-up the R environment and save the covariate stack.

```
# check the covariate names
names(r6)
```

```
##  [1] "B02CHE3"    "B04CHE3"    "B07CHE3"    "B13CHE3"    "B14CHE3"
##  [6] "BARL10"     "CMCF5avg"   "CRDMRG5"    "CRUMRG5"    "CRVMRG5"
## [11] "DEMENV5"    "DV2MRG5"    "DVMMRG5"    "ENTENV3"    "ESMOD5avg"
## [16] "EVEENV3"    "EXMOD5avg"  "F01USG5"    "F02USG5"    "F04USG5"
## [21] "F05USG5"    "F06USG5"    "F07USG5"    "IMOD4avg"   "MANMCF5"
## [26] "MAXENV3"    "MMOD4avg"   "MRNMRG5"    "NEGMRG5"    "NIRL00"
## [31] "NIRL14"     "NMOD3avg"   "NMSD3avg"   "PCHE3avg"   "POSMRG5"
```

```
## [36] "PRSCHE3"    "RANENV3"    "REDL00"     "REDL14"     "SESA4avg"
## [41] "SLPMRG5"    "SW1L00"     "SW1L14"     "SW2L00"     "SW2L14"
## [46] "TMDMOD3"    "TMNMOD3"    "TPIMRG5"    "TWIMRG5"    "VBFMRG5"
## [51] "VDPMRG5"    "VW1MOD1avg" "s1"         "s2"         "LCEE1010"
## [56] "LCEE1060"
```

```
# re-order; names of coordinate columns are "s1","s2"
r <- cbind(r6[,c("s1","s2")], r6[,!names(r6) %in% c("s1","s2")])

# clean-up
rm(dum, i)

# save covariate stack
save(r, file='covariateStack.rda')
```

Note that we have made several copies of the covariate stack objects (r1 - r6) with intermediate results. We have done this on purpose here so it is easy to go back one or two processing steps without having to rerun all previous steps. Of course this replication of data is memory inefficient and is not required. One could create one covariate stack object, e.g. `r` and then use this object for all processing steps, hereby overwriting previous versions of the object.

3. Creating a regression matrix

A digital soil mapping model tries to find predictive relationships between soil observation and covariates through empirical correlation. In order to calibrate a DSM model, we must obtain the covariate values for each sampling site. This can be done through a spatial overlay that results in an object called a **regression matrix**.

We start by converting the soil sample dataset to a spatial object: a **SpatialPointsDataFrame**, which is a spatial data class of the **sp** package, and defining the projection with the **CRS()** function. The sample data must have the same projection as the covariate stack to do an overlay. We have found out earlier that the EPSG code for the local Macedonian coordinate system is 6316.

```
# get site locations from the data_stock file
coordinates(d) <- ~X_coord+Y_coord

# set projection
proj4string(d) <- CRS("+init=epsg:6316")

# show the data structure
str(d)
```

We proceed with converting the covariate stack from a `data.frame` to a `SpatialPixelsDataFrame` object. We also have to set the projection of the spatial stack.

```
# get site locations from the data_stock file
gridded(r) <- ~s1+s2

# set projection
proj4string(r) <- CRS("+init=epsg:6316")
```

To extract the covariate values at the sampling sites from the covariate layers we use the **over** function of the `sp` package. The spatial overlay results in a `data.frame` (not spatial!) that we can then simply append to the `data` slot of the spatial object that stores the sampling data.

```
# spatial overlay
dum <- over(d, r)

# append covariate data to soil data
rm <- cbind(d@data, d@coords, dum)

# save
save(rm, file="regressionMatrix.rda")
```

Alternatively, the overlay can be done with the functionality of the **raster** package. The **rasterFromXYZ()** function converts a `data.frame` to a `RasterLayer` or `RasterBrick` object. Note that this function requires that the first column in the `data.frame` stores the X coordinate (longitude) and the second the Y coordinate (latitude). The **extract()** function extracts the covariate values at the sampling sites from the stack. This will give the same output as using the functions of the **sp** package.

```
# convert stack to data.frame and reorder
r <- as.data.frame(r)
r <- cbind(r6[,c("s1","s2")], r6[,!names(r6) %in% c("s1","s2")])

# create rasterBrick object
r7 <- rasterFromXYZ(r, crs="+init=epsg:6316")

# overlay
dum <- extract(r7, d)

# convert to data.frame
dum <- as.data.frame(dum)

# convert categorical covariates to factor
dum <- cbind(r6[,c("s1","s2")], dum[,names(dum) %in% c("LCEE")])

# convert the binary land cover layers to factor
# use patter recognition with the grep function to identify columns
dum[,grep("LCEE", names(dum), value = TRUE)] <- lapply(dum[,grep("LCEE", names(dum), value = TRUE)], FU

# append covariate data to soil data
rm <- cbind(d@data, d@coords, dum)

# clean-up
rm(dum)
```

Save the working environment.

```
# save
save.image("DataPreparation.rda")
```

Acknowledgements

This tutorial is developed from an earlier version prepared by Dr. Bas Kempen (ISRIC - World Soil Information) and Dr. Titia Mulder (Wageningen University).

The soil data used here is from the Macedonian Soil Information System (Vrscaj et al. 2017), and provided by the Secretariat of the Global Soil Partnership. The author is grateful to Dr. Mulder for processing the raw soil point data into the two data tables that are provided with this document.

The covariate layers were all derived from freely available global biophysical data and have been made available by ISRIC for all territories in the world at 1km resolution from ftp://isric.org (user: gsp, password: gspisric).

References

Vrscaj, B., L. Poggio, D. Muaketov, and R. Vargas. 2017. "Utilizing the Legacy Soil Data of Macedonia: The Creation of the Macedonian Soil Information System and its use for digital soil mapping and assessment applications. Abstract Book of Pedometrics 2017, Wageningen, 26 June - 1 July 2017." http://www.pedometrics2017.org/s/Abstract-Book-Pedometrics-2017.pdf.