



Food and Agriculture
Organization of the
United Nations



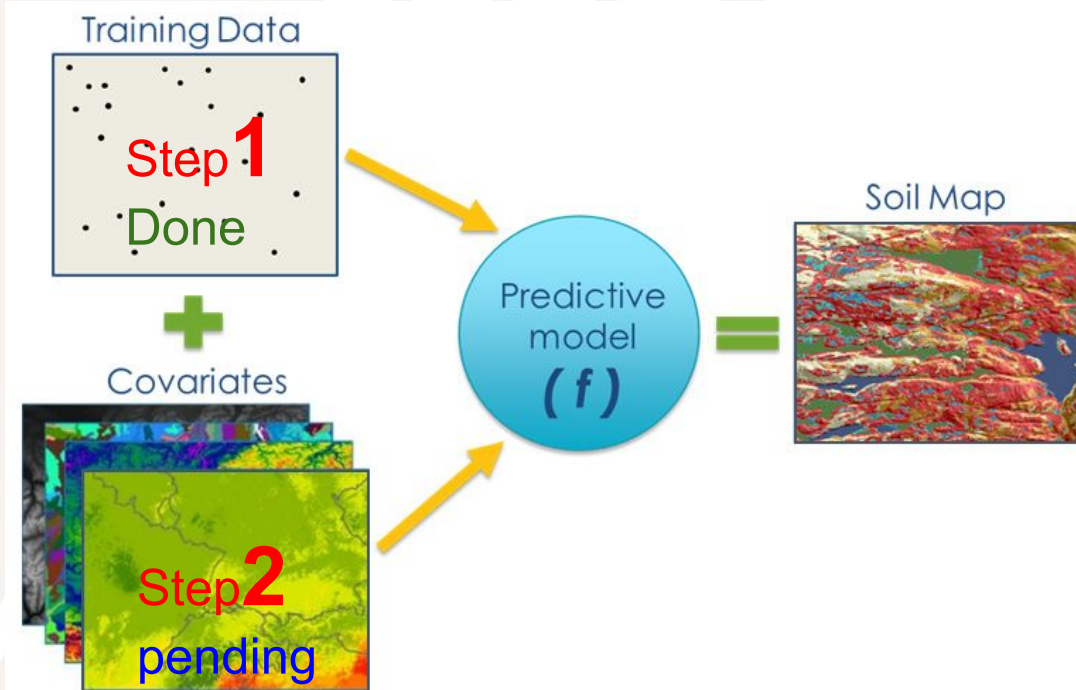
Rural Development
Administration



Preparation of spatial covariates for digital soil mapping

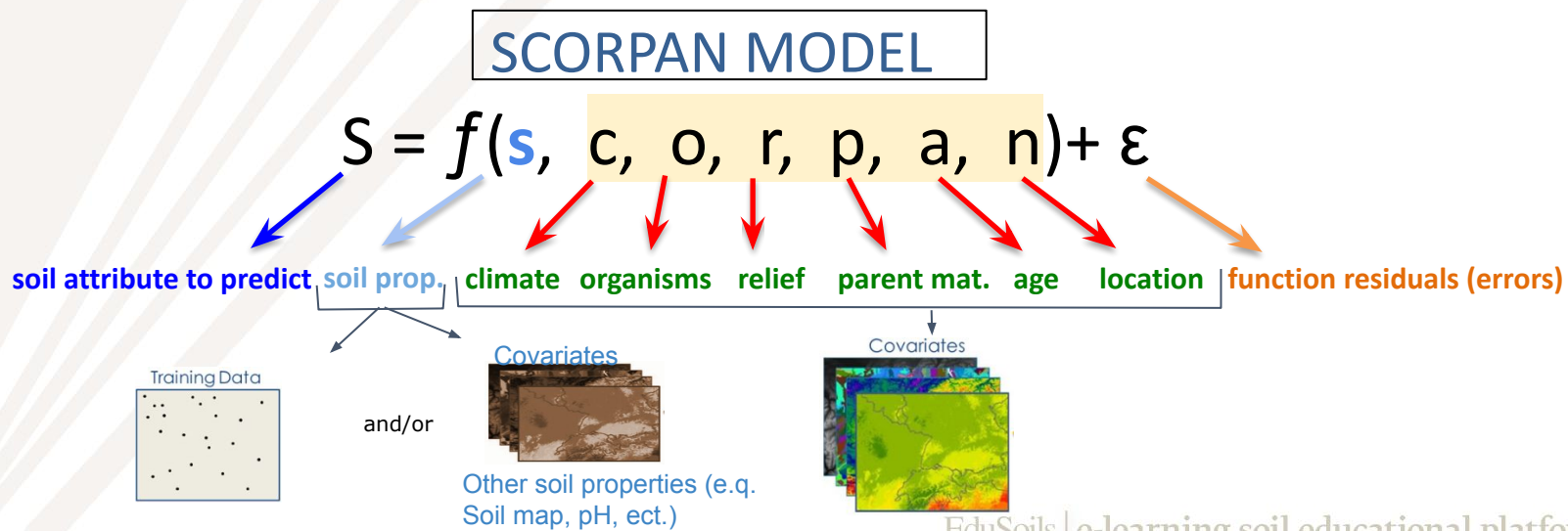


Digital Soil Mapping (DSM)



What are covariates?

In the SCORPAN reference framework a soil attribute (e.g., SOC) can be predicted as a function of the soil forming environment, in correspondence with soil forming factors based on climate, organisms, relief, parent material and elapsed time of soil formation (Florinsky, 2012)



Soil Properties

- Soil data:
 - Soil Samples
 - Profiles
 - Soil maps (**soil type**, soil property maps)
- It can be the soil property itself or a soil property that is correlated with the soil property we're trying to predict

Why do we need SOC data if we're trying to predict SOC?

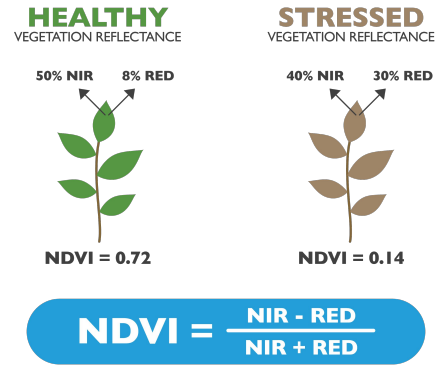
- Model calibration → using soil data to improve your models (if you want to create a model to identify cookies it's useful to make it first taste cookies)
- Validation → comparing predicted vs actual soil properties

Climate

- Some common climatic variables that are regularly observed and mapped over countries are:
 - minimum and maximum temperature
 - cumulated temperature and mean temperature
 - cumulated precipitation and mean precipitation
 - potential evapotranspiration
 - climatic water balance
 - radiation
 - snow cover, etc.
- WorldClim V1.4 and V2 is a set of global climate layers (gridded climate data) with a spatial resolution of about 1 km²
- WorldClim data layers were generated by interpolation of average monthly climate data from weather stations

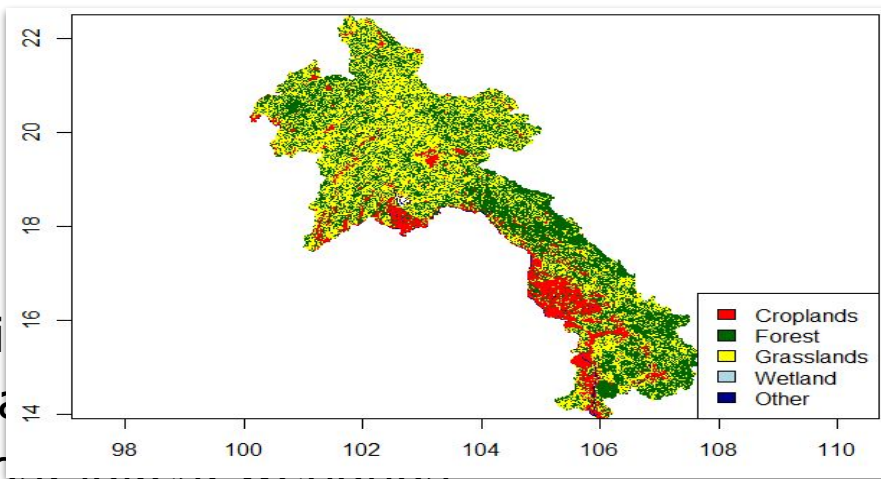
Organisms

- A **vegetation index** is an indicator that describes the greenness, the relative density and health of **vegetation** for each pixel, in a satellite image
- Although there are several **vegetation indices**, one of the most widely used is the Normalized Difference **Vegetation Index** (NDVI)
- Normalized Difference Vegetation Index (NDVI) quantifies vegetation by measuring the difference between near-infrared (which vegetation strongly reflects) and red light (which vegetation absorbs)
- The enhanced vegetation index (EVI) was developed as an alternative vegetation index to address some of the limitations of the NDVI



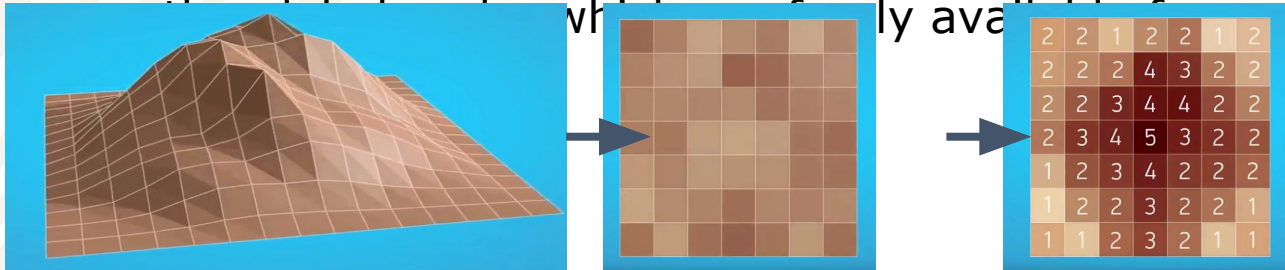
Organisms

- Land cover maps represent spatial types (classes) of physical cover, e.g. forests, grasslands, croplands, lakes, wetlands
- Land cover maps are a great asset for predicting soil properties, since the type of vegetation greatly affects them
- There are several free global and continental datasets of land cover maps: i.e. GlobCover, GeoCover, Globeland30, CORINE Land Cover.



Relief

- Digital Elevation Models (DEM) translate elevation values into grid cells
- Several indices (e.g. terrain slope, valley depth)
- Currently, two global level 30m DEMs are freely available; the Shuttle Radar Topographic Mission (**SRTM**) and the ASTER Global Digital Elevation Model (**GDEM**). They provide topographic data



Video explaining DEM: <https://www.youtube.com/watch?v=0UwgPOAkx-c>

Parent Material

- National parent material and geology maps may be used. Other available datasets and data portals are given on the Isric WorldGrids website (worldgrids.org)
- Data sources for parent material maps are:
 - OneGeology
 - USGS
 - GLiM

SCORPAN MODEL

$$S = f(s, c, o, r, p, a, n) + \epsilon$$

soil attribute to predict soil prop. climate organisms relief parent mat. age location function residuals (errors)

Selecting covariates

- The example covariates from this chapter were prepared by ISRIC (ISRIC, 2017)
- ISRIC World Soil Information has established a data repository which contains raster layers of various biophysical earth surface properties for each territory in the world
- The 171 raster layers are divided per country and can be retrieved here:
<https://files.isric.org/projects/gsp/>

User name: gsp

Password: gspisric

Index of /projects/gsp

- [Parent Directory](#)
- [Afghanistan/](#)
- [Albania/](#)
- [Algeria/](#)
- [American_Samoa/](#)
- [Andorra/](#)
- [Angola/](#)
- [Anguilla/](#)
- [Antigua_and_Barbuda/](#)
- [Argentina/](#)
- [Armenia/](#)
- [Aruba/](#)
- [Ashmore_and_Cartier_Islands/](#)
- [Australia/](#)
- [Austria/](#)
- [Azerbaijan/](#)

Selecting Covariates

- Adding relevant covariates that can help explain the distribution of soil properties increases the accuracy of spatial predictions
- Following the SCORPAN model is a good strategy, but one must keep in mind that certain factors are more important e.g. in the tropics than in temperate regions
- During the workshop we will work with a few covariates from ISRIC, a histosol layer retrieved from the Harmonized World Soil Database (HWSD) and a land cover map derived from CORINE

List of Covariates I

B04CHE3	Temperature seasonality at 1 km (based on CHELSA climate surfaces).
B07CHE3	Temperature Annual Range [°C] at 1 km (based on CHELSA climate surfaces).
B13CHE3	Precipitation of wettest month [mm] at 1 km (based on CHELSA climate surfaces).
B14CHE3	Precipitation of driest month [mm] at 1 km (based on CHELSA climate surfaces).
BARL10	Global 30m Bare Ground (circa 2010)
DEMENV5	Land surface elevation
ENTENV3	Attribute derived from EVI
EVEENV3	Attribute derived from EVI
Soil types	Soil types layer from the Harmonized World Soil Database (HWSD)
MAXENV3	EVI
NIRL00	Mean NIR reflectance for the year 2000
PRSCHE3	Total annual precipitation at 1 km

List of Covariates II

RANENV3	Attribute derived from EVI
REDL00	Landsat Band 3 (red) for year 2000
SLPMRG5	Terrain slope
SW1L00	Landsat Band 5 (SWIR) for year 2000
SW2L00	Landsat Band 7 (SWIR) for year 2000
TMDMOD3	Mean annual LST (daytime) MODIS
TMNMOD3	Mean annual LST (nighttime) MODIS
TPIMRG5	Topographic Position Index
TREL10	Global 30m Tree Cover
TWIMRG5	SAGA Wetness Index
VBFMRG5	Multiresolution Index of Valley Bottom Flatness (MRVBF)
VDPMRG5	Valley depth
Hist	Histosol layer from ISRIC

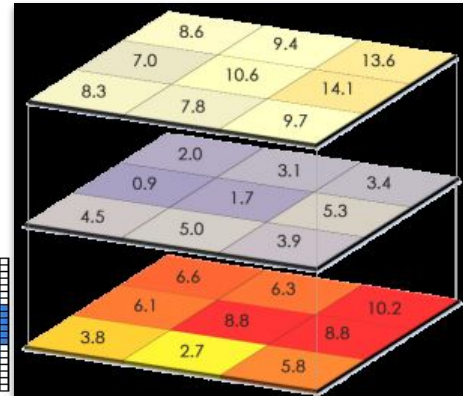
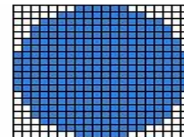
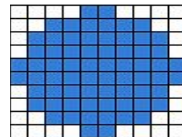
Preparation of Spatial Covariates

In this exercise we're going to focus on some routine steps required to prepare the covariates before modelling:

1. [Prepare a RasterStack](#)
2. [Create a SpatialPointsDataFrame](#)
3. [Extract data from the Rasterstack and merge it with our data](#)
4. [Check which covariates are correlated with the target variable](#)
5. [Subset the Rasterstack](#)
6. [Reproject a land cover map](#)
7. [Rasterize a soil type map](#)
8. [Mask our stacked covariates](#)
9. [Save the rasterStack as .RData](#)

The RasterStack

- Most of the functions for handling raster data are available in the **raster** package
- Multiple layers can be combined into the object class RasterStack, which allows you to perform things on multiple rasters at once
- Rasters can be *stacked* only, if they have the same:
 - Extent (or in other words they cover the same area),
 - The same file extension
 - Same projection
 - Pixel resolution (cell size)



RasterStack

- As a first step we're going to create a character vector containing all the names of the covariates with the `list.files()` function
- Covariate names will be selected based if the end with `.tif`
- This will tell R which covariates to stack when using the `stack()` function

```
# Set a working directory

setwd("C:\\Macedonia")
library(raster)
files <- list.files(path = "01-Data/covs", pattern = "tif*$",
                   full.names = TRUE)
covs <- stack(files)
```


RasterStack

- We can treat the rasterstack similarly to a dataframe
- Just like with the a dataframe we can use \$ to select which covariate we want to explore

```
# Explore the data  
names(covs)  
plot(covs$DEMENV5)
```

Question 1: What's the highest total annual precipitation for Macedonia?

SpatialPointsDataFrame

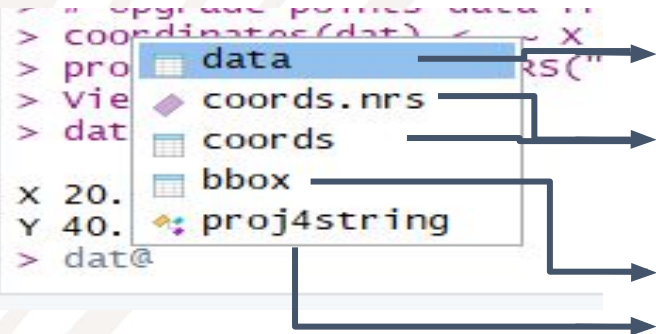
- `SpatialPointsDataFrame` structure is essentially like a data frame, except that additional "spatial" elements have been added
- To define the **CRS**, we must know where our data is from, and what was the corresponding **CRS** used when recording the spatial information **in the field**
- First we're going to import a dataframe

```
# Load the processed data for digital soil mapping. This table was #prepared in the
'data_preparation_profiles' script
dat <- read.csv("02-Outputs/dat_train.csv")
# Upgrade points data frame to SpatialPointsDataFrame and define their coordinate system
coordinates(dat) <- ~ X + Y
proj4string(dat) = CRS("+init=epsg:4326") # WGS84
```

SpatialPointsDataFrame

- **SpatialPointsDataFrame** structure is essentially like a data frame, except that additional "spatial" elements have been added
- The spatial components that were added to the dataframe can be explored with the @ symbol

```
> COORDINATOR(dat) <- CRS("
> proj4string(dat) <- CRS("
> view(dat)
> dat@
x 20.
y 40.
> dat@
```



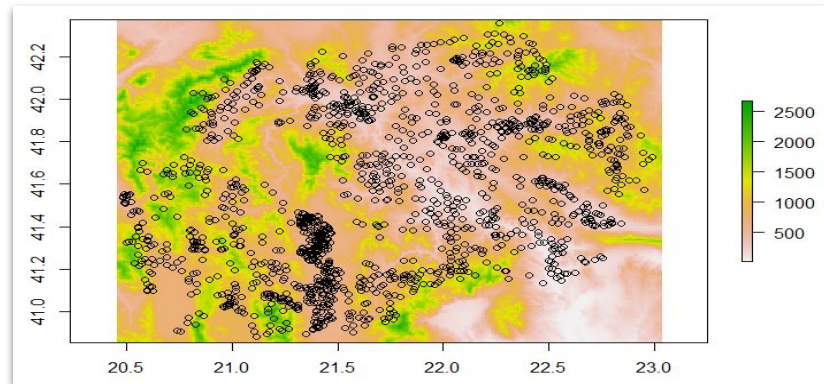
- In GIS terminology this is now the attribute table
- And index indicating which columns have XY coordinates and the matrix containing the XY columns
- The Bounding Box
- Object of class "CRS" projection string

```
#To explore single columns use @ and then $
dat@data$SOC
```

Extract data from the Rasterstack

- With the extract function we can now add the values of the Rasterstack to the SpatialPointsDataFrame
- Y in our argument corresponds to our point data
- The extract function will extract a value only where the points overlay a raster cell

```
# Check that the points overlay with the  
#rasters  
plot(covs$DEMENV5)  
points(dat)
```

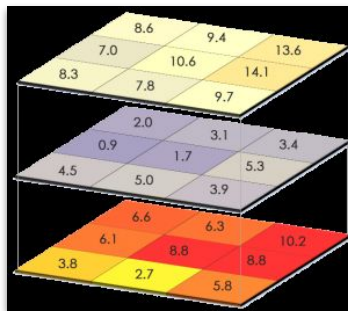


Extract data from the Rasterstack

- In the extract() function we're defining which object contains rasters (=x) and which object is made out of discrete locations (=y)
- By putting sp= TRUE we're telling are that the y object has a spatial component

```
# Extract values
```

```
dat <- extract(x = covs, y = dat,  
sp = TRUE)  
summary(dat)
```



	M	BLD	CRFVOL	OCSKGM	meaERROR	OCSKGMlog	
1	P0004	14.2546404	0.6892215	6.3053161	276.15398	4.69	6.620959
2	P0011	2.2877943	1.2781158	27.3020950	63.80581	2.87	4.155844
3	P0012	3.6345497	1.2519741	22.9949428	76.19777	3.00	4.333332
4	P0013	4.2923954	1.0361627	26.0004816	98.73652	2.58	4.590455
5	P0015	4.4892420	1.0157821	16.3997185	114.36749	2.87	4.739417
6	P0016	4.4945244	1.0155001	12.4031778	119.94257	3.00	4.787013
7	P0018	4.0471519	1.0694152	38.8981681	79.33619	2.19	4.373694
8	P0019	4.7951981	0.9769693	34.1128101	92.59973	2.26	4.528286
9	P0020	4.118295	1.0430139	44.0000000	72.16398	1.99	4.278941
10	P0024	1.3142620	1.4517808	33.0047946	38.34846	2.94	3.646714
11	P0027	1.6063333	1.3930520	1.1328188	66.57070	4.17	4.195256
12	P0028	3.9197415	1.0645912	1.9656346	122.70190	3.39	4.809758
13	P0030	1.7007044	1.3757932	7.9473790	64.61590	3.84	4.168461
14	P0033	1.8825410	1.3472143	19.9394573	60.91453	3.29	4.109472
15	P0035	2.4672431	1.2514827	28.1527537	65.63852	2.75	4.184178

Check for correlation

- The object class limits which functions we can use, in this case: `complete.cases()` and `cor()`
- Therefore we have to back-transform our `SpatialPointDataFrame` into a `dataframe` with the `as.dataframe()` function

```
# Remove NA values
dat<-as.data.frame(dat)
dat <- dat[complete.cases(dat),]
# Test correlation between each covariate and the 'OCSKGMlog' and #the other covariates, select with
[]
names(dat)
test_covs <- cor(x = as.matrix(dat[,8]),
                 y = as.matrix(dat[,c(9:33)]))
test_covs
```

Check for correlation

- Let's Select only the covariates that have correlation higher than 0.3

```
library(reshape)
#Let's get rid of NAs and correlation = 1 and reshape test_covs
x <- subset(melt(test_covs), value != 1 | value != NA)
test_covs

#Arrange the the values in descending order
x <- x[with(x, order(-abs(x$value))),]
```

ID	Product1	Product2	Product3	Product4
1	1	NA	1	1
2	1	1	NA	1
3	1	1	NA	NA
4	1	1	1	1

wide-format

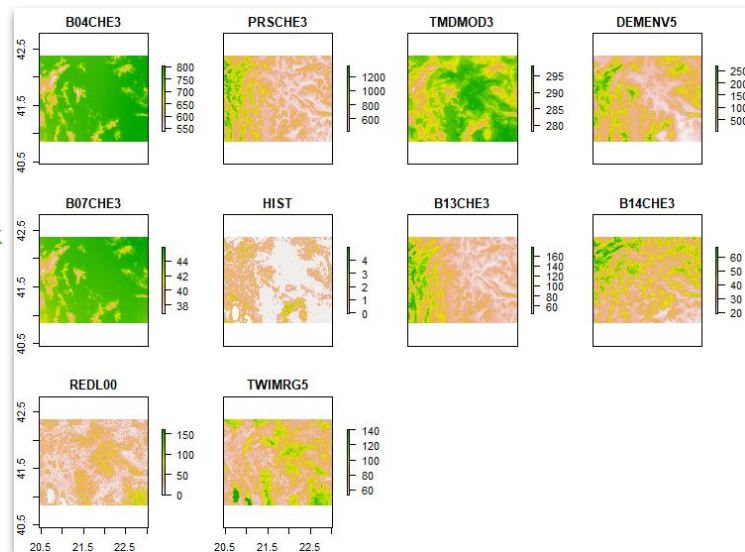
ID	Product	value
1	Product1	1
1	Product3	1
1	Product4	1
2	Product1	1
2	Product2	1
2	Product4	1
3	Product1	1
3	Product2	1
4	Product1	1
4	Product2	1
4	Product3	1
4	Product4	1

long-format

Subset the RasterStack

```
#Subset for values greater than 0.3
(x <- subset(x,abs(x$value)>0.3))
selection <- as.character(x$X2) #This creates an index

# Leave only selected covariates in the 'covs' raster stack
covs <- covs[[selection]]
names(covs)
plot(covs)
```



Reproject a land cover map

- . We want to add a new covariate from a different dataset
- . Rasters can be *stacked* only if they have the same:
 - Extent (or in other words they cover the same area),
 - The same file extension
 - Same projection
 - Pixel resolution (cell size)
- . In this example we're loading a categorical land cover map into R

```
# Import land cover layer
landcover <- raster ('01-Data/land cover/LandCover.tif')
plot(landcover)
# Try to stack LandCover raster with the rest
covs <- stack(covs, landcover)
```

Reproject a land cover map

```
# Error indicates different extent.  
#Check coordinate system (crs) of covs and LandCover  
covs@crs  
landcover@crs
```

```
> # Try to stack LandCover raster with the rest  
> covs <- stack(covs, landcover)  
Error in compareRaster(x) : different extent  
>  
> # Error indicates different extent. Check coordinate system (crs) of covs and LandCover  
> (covs@crs)  
CRS arguments:  
+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0  
> (landcover@crs)  
CRS arguments:  
+proj=utm +zone=34 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0  
>
```

- 'covs' has crs WGS 84, while landcover has crs UTM Zone 34

Reproject a land cover map

```
# 'covs' has crs WGS 84, while landcover has crs UTM Zone 34
# We need to reproject the 'Lancover' raster using one of the #'covs' as a template
landcover <- projectRaster(from = landcover, to = covs$DEMENV5, method = "ngb")

# Save the reprojected raster
writeRaster(landcover, '02-Outputs/Landcover_WGS84.tif', overwrite=TRUE)

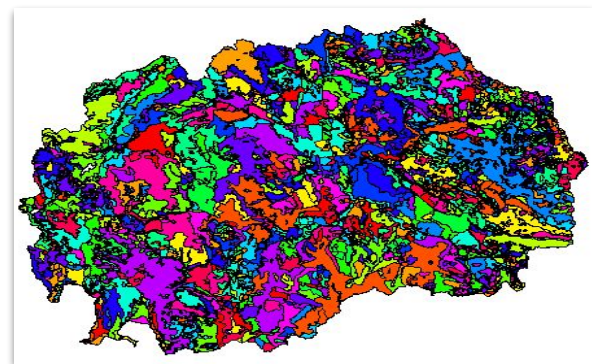
# Now we can stack it with the other rasters
covs <- stack(covs, landcover)
```

Rasterize a soil type map

- Reading in a vector in shapefile (.shp) format is analogous to loading a raster into r: shapefile()
- The function shapefile() is part of the package called raster

```
#Import and explore the soil map (vector polygon data)
soilmap<-shapefile("01-Data/Soil map/SoilTypes.shp")
plot(soilmap, col= rainbow(19))
summary(soilmap)
```

```
> summary(soilmap)
Object of class SpatialPolygonsDataFrame
Coordinates:
      min      max
x 20.45862 23.03978
y 40.85309 42.37356
Is projected: FALSE
proj4string :
[+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0]
Data attributes:
      FAO          WRB          Symbol
Length:2516      Length:2516      Length:2516
Class :character  Class :character  Class :character
Mode :character  Mode :character  Mode :character
```



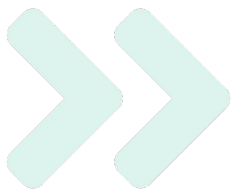
Rasterize a soil type map

- Factors (categories) in R are stored as a vector of integer values with a corresponding set of character values

```
soilmap@data$Symbol <- as.factor(soilmap@data$Symbol)
str(soilmap@data)
```

```
> str(soilmap@data)
'data.frame': 2516 obs. of 3 variables:
 $ FAO : chr "Fluvisol" "Fluvisol" "Fluvisol" "Regosol" ...
 $ WRB : chr "Fluvisol" "Fluvisol" "Fluvisol" "Regosol" ...
 $ symbol: Factor w/ 19 levels "ATa","B","C",...: 8 8 8 15 4 13 13 4 15 2 ...
```

Symbol
ATa
B
C
J



1	1	19	19
1	1	19	18
2	2	3	15
2	2	3	3
3	4	4	4

EduSoils | e-learning soil educational platform



Rasterize a soil type map

- The function rasterize() allows you to rasterize a shapefile according to another raster → the output will have the same spatial extent and resolution

```
# Rasterize the soil map using one of the 'covs' as a template and #'Symbol'  
as value field
```

```
soilmap.r <- rasterize(x = soilmap, fun='first', y = covs$DEMENV5, field =  
"Symbol")
```

Rasterize a soil type map

- Let's see the difference: the borders are gone and now we have a raster with pixels containing numbers corresponding to each soil type symbol

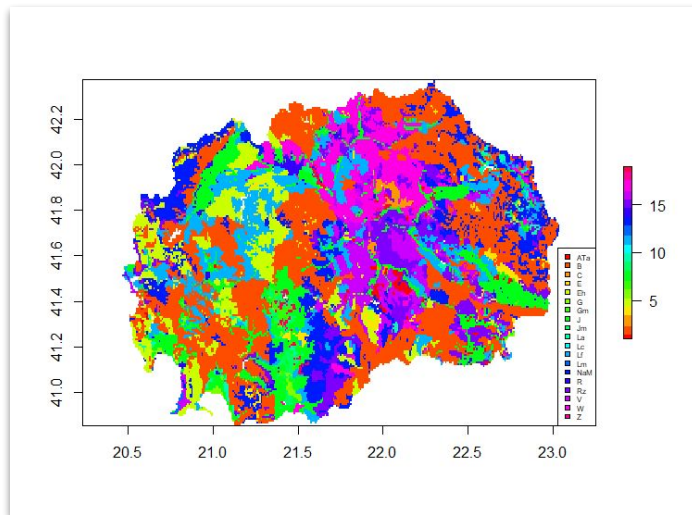
```
# Explore the result
```

```
plot(soilmap.r, col=rainbow(20))
```

```
legend("bottomright", legend =
```

```
levels(soilmap$Symbol),
```

```
fill=rainbow(20), cex=0.5)
```



EduSoils | e-learning soil educational platform



Rasterize a soil type map

```
# Save the rasterized map
writeRaster(soilmap.r, '02-Outputs/Soil_map.tif', overwrite=TRUE)

# Now we can stack it with the other rasters
covs <- stack(covs, soilmap.r)
names(covs)
# correct the name
names(covs)[names(covs)=='layer'] <- "soilmap"
```

Question 2: What would be another way to change the name of the layer?

Mask values in a Raster object

- Create a new raster that has the same values as x, except for the cells that are NA in the mask
- This makes sense when mapping SOC and wanting to get rid of cells that are located on cities or water bodies

```
# Mask the covariates with the country mask
```

```
plot(covs$DEMENV5)
```

```
mask <- raster("01-Data/mask.tif")
```

```
plot(mask)
```

```
covs <- mask(x = covs, mask = mask)
```

```
plot(covs$DEMENV5)
```

1	1	19	19
1	1	19	18
2	2	3	15
2	2	3	3
3	4	4	4

1	1	1	NA
1	1	1	1
1	1	1	1
1	1	1	NA
1	1	1	1

1	1	19	
1	1	19	18
2	2	3	15
2	2	3	
3	4	4	4

Export the RasterStack as .RData

- .RData → File created by R, saves a workspace, which includes the function and objects created during an open session in R; used for storing a working environment that can be extended or modified later.

```
# export all the covariates in the R-object  
save(covs, file = "02-Outputs/covariates.RData")
```



Food and Agriculture
Organization of the
United Nations



Thank you for your attention !

:)

