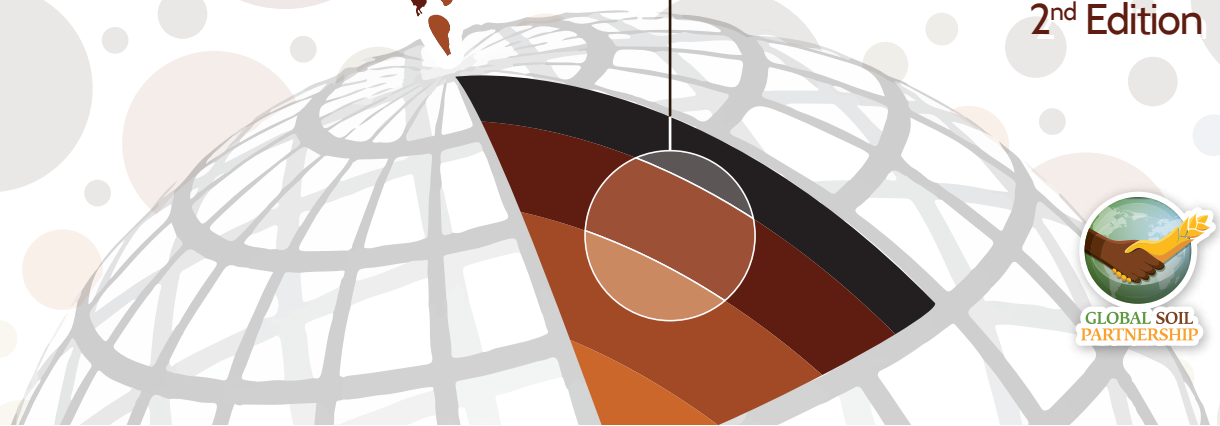
Food and Agriculture Organization of the United Nations

EDITION 2nd

SOIL ORGANIC CARBON MAPPING Cookbook

2nd Edition



GLOBAL SOIL PARTNERSHIP

Soil Organic Carbon Mapping Cookbook

2nd edition

Food and Agriculture Organization of the United Nations, Rome, 2018

ISBN 978-92-5-130440-2

Contents

Figures

Tables

Acronyms

ASCII American Standard Code for Information Interchange

AVE amount of variance explained

BD bulk density

CDF cumulative distribution function

CGMS Coordination Group for Meteorological Satellites

CI confidence interval

COE coefficient of efficiency

CRAN Comprehensive R Archive Network

CRS coordinate reference system

CSV comma-separated values

CV cross-validation

DB database

DEM digital elevation model

DSM digital soil mapping

EEA European Environment Agency

EPSG European Petroleum Survey Group

ESA European Space Agency

FAC fraction of prediction

FAO Food and Agriculture Organization of the United Nations

FYROM Former Yugoslav Republic of Macedonia

GAUL Global Administrative Unit Layers

GDAL Geospatial Data Abstraction Library

GDEM Global Digital Elevation Model

GIS Geographic Information System

GLOSIS Global Soil Information System

GM Geo-matching

GPS Global Positioning System

GSIF Global Soil Information Facility

GSOC global soil organic carbon

GSP Global Soil Partnership

GUI graphical user interface

HWSD Harmonized World Soil Database

IIASA International Institute for Applied Systems Analysis

INSII International Network of Soil Information Institutions

IOA index of agreement

ISRIC International Soil Reference and Information Centre

ISS CAS Institute of Soil Science, Chinese Academy of Sciences

ITPS Intergovernmental Technical Panel on Soils

JRC Joint Research Center European Commision

KML Keyhole Markup Language

MAE mean absolute error

MASIS Macedonian Soil Information System Database

MB mean bias

ME mean error

MGE mean gross error

MSDR mean squared deviation ratio

MSE mean squared error

NA not available, missing values

NMB normalized mean bias

NMGE normalized mean gross error

OCS organic carbon stock

PTF pedo-transfer functions

RF random forest

RK regression-kriging

RMSE	root-mean-square error
SIS	Soil Information System
SMLR	stepwise multiple linear regression
SOC	soil organic carbon
SOM	soil organic matter
SPC	Soil Profile Collection
SRTM	Shuttle Radar Topographic Mission
SVM	support vector machines
TSS	total sum of squares
UNEP	United Nations Environment Programme
UTM	Universal Transverse Mercator
VIF	Variance Inflation Factor
WCS	Web coverage service
WGS	World Geodetic System
WRB	World Reference Base

List of contributors

Editorial board

Yusuf Yigini, Guillermo Federico Olmedo, Stephanie Reiter, Rainer Baritz, Kostiantyn Viatkin, Ronald R. Vargas Global Soil Partnership, Food and Agriculture Organization of the United Nations

Cover design: Matteo Sala

Contributing authors

Rainer Baritz - European Environment Agency

Dick Brus - Department of Plant Sciences, Wageningen University, the Netherlands

Mario Guevara - University of Delaware, USA

Tomislav Hengl - ISRIC - World Soil Information, Wageningen, the Netherlands

Gerard Heuvelink - World Soil Information, Wageningen, the Netherlands

Bas Kempen - ISRIC, World Soil Information, Wageningen, the Netherlands

Titia V.L. Mulder - Wageningen University, Department of Environmental Sciences, the Netherlands

Guillermo Federico Olmedo - Instituto Nacional de Tecnología Agropecuaria, Mendoza, Argentina

Laura Poggio - The James Hutton Institute, Craigiebuckler Aberdeen, Scotland UK

Eloi Ribeiro - ISRIC, World Soil Information, Wageningen, the Netherlands

Christian Thine Omuto - Department of Environmental and Biosystems Engineering, University of Nairobi, Kenya

Yusuf Yigini - Food and Agriculture Organization of the United Nations

Ronald R. Vargas - Food and Agriculture Organization of the United Nations

Recommended citation:

FAO. 2018. Soil Organic Carbon Mapping Cookbook. Y. Yigini, G.F. Olmedo, S. Reiter, R. Baritz, K. Viatkin, and R.R. Vargas, (Eds). 2nd Edition, Rome.

Chapter 1

Presentation

Soils provide ecosystem services critical to life on Earth. The Food and Agricultural Organization of the United Nations (FAO) recognizes the need to preserve soil resources from degradation and restore them. In 2012, the Global Soil Partnership (GSP) was established to improve soil governance and promote sustainable soil management .

The GSP aims to promote sustainable soil management at all levels globally through normative tools relying on evidence-based science. Understanding the status of a given soil in different land uses, including its properties and functions, and relating this information to the various ecosystem services provided by soils, becomes mandatory for sustainable soil management decisions. As the availability and use of soil data and information is fundamental to underpin these decisions, members of the GSP decided to establish a Global Soil Information System (GLOSIS) based on the development of national soil information systems .

In the process of establishing GLOSIS, a number of tools and networks are being created, including the International Network of Soil Information Institutions (INSII), a *GSP Soil Data Policy* (FAO and GSP, 2017b) and more. Taking advantage of this process and responding to a request for support in addressing the Sustainable Development Goal Indicators, especially indicator 15.3 which includes the restoration of degraded soils, the GSP Plenary Assembly in 2016 instructed the Intergovernmental Technical Panel on Soils (ITPS) and the GSP Secretariat to develop the first-ever Global Soil Organic Carbon map (GSOCmap) following the same bottom-up approach as GLOSIS. To this end, members under the INSII umbrella developed guidelines and technical specifications for the preparation of the GSOCmap and countries were invited to prepare their national soil organic carbon maps according to these specifications.

Given the scientific advances in tools for mapping soil organic carbon (SOC), many countries requested the GSP Secretariat to support them in the process of preparing national maps, hence an intensive capacity development programme on SOC mapping has been implemented to support countries in this process. Various re-

gional and national training sessions were organized using an on-the-job-training modality to ensure national experts were trained to utilize their own datasets to produce reliable SOC maps. The GSP Secretariat invited a group of experts to prepare the first edition of a *Soil Organic Carbon Mapping Cookbook* (FAO, 2017) as a comprehensible reference knowledge source to support the capacity development process.

The second edition of the cookbook provides generic methodologies and technical steps to produce SOC maps. This edition has been updated with knowledge and practical experiences gained during the implementation process of GSOCmap V1.0 throughout 2017. The cookbook includes step-by-step guidance for developing 1 km grids for SOC stocks, as well as for the preparation of local soil data, the compilation and preprocessing of ancillary spatial data sets, upscaling methodologies, and uncertainty assessment methods. Guidance is mainly specific to SOC data, but as this cookbook contains generic Sections on soil grid development it can be applicable to map various soil properties.

The guidance is focusing on the upscaling of national SOC stocks in order to produce the GSOCmap . Therefore, the cookbook supplements the *GSP Guidelines for Sharing National Data/Information to Compile a Global Soil Organic Carbon (GSOC) Map* (FAO and GSP, 2017a), providing technical guidelines to prepare and evaluate spatial soil data sets to:

- Determine SOC stocks from local samples to a target depth of 30 cm;
- Prepare spatial covariates for upscaling; and
- Select and apply the best suitable upscaling methodology.

In terms of statistical upscaling methods, the use of conventional upscaling methods using soil maps and soil profiles is still very common, although this approach is mostly considered empirical by soil mappers. Even though evaluations are based on polygon soil maps, the resulting SOC maps can be rasterized to any target grid. However, a spatially-explicit assessment of uncertainties is impossible. The use of digital soil mapping to upscale local soil information is increasingly applied and recommended.

This cookbook presents two approaches in detail, namely spatial modelling using either regression or data mining analysis , combined with geostatistics such as regression-kriging . The second edition includes updates on uncertainty assessment of the presented methods.

It is our hope that this cookbook will fulfill its mandate of easily enabling any user to produce a digital SOC or other soil property map using soil legacy data and modern methods of digital soil mapping with the overall aim for improved decision making on soil management.

1.1 How to use this book

In the second edition of this cookbook, we want to introduce five different approaches for obtaining a SOC map, using sample data set from the Former Yugoslav Republic of Macedonia (FYROM).

The first two mapping methods presented are classified as conventional upscaling using soil maps. The first one is class-matching . In this approach, we derive average SOC stocks per class from the soil type for which a national map exists, or through combination with other spatial covariates (e.g. land use category, climate type, biome, etc.) . This approach is used in the absence of spatial coordinates of the source data. The second one is geo-matching , were upscaling is based on averaged SOC values per mapping unit.

In addition to that, we present three digital soil mapping methods . Regression-kriging is a hybrid model with both, a deterministic and a stochastic component (Hengl et al., 2007). Next method is called random forest . Random forest is an ensemble of regression trees based on bagging. This machine learning algorithm uses a different combination of prediction factors to train multiple regression trees (Breiman, 1996). The last method is called support vector machines . This method applies a simple linear method to the data but in a high-dimensional feature space non-linearly related to the input space (Karatzoglou et al., 2006).

We present this diversity of methods because there is no *best* mapping method for digital soil mapping and testing, and selection has to be done for every data scenario (Guevara et al., 2018).

This cookbook is a step-by-step manual starting with the soil data preparation, producing the map, and finishing with map validation and data sharing (see Fig. 1.1). In the different chapters, we present a continuing example using the FYROM soil database and some environmental covariates. In every chapter, we start with the input data and produce the proposed output, which is usually used as input for the following chapter. The soil data needed can be downloaded from the data repository, and the same applies to the environmental covariates. The results obtained in every chapter can be also downloaded for comparison purposes from the result folder. Finally, all the needed code is summarized in Chapter 12 and can be downloaded from the code folder.

The authors use the Github development environment to build this document. We produced also an online version of the cookbook to be able to implement minor changes/improvements without reprinting. We encourage users to follow the online version which is provided on the GSP Website.

1.2 The FYROM soil database

The sample data set used for the spline function and the regression-kriging for digital soil mapping in this cookbook has been extracted from the Macedonian Soil Information System Database (MASIS). The database contains around 4,000

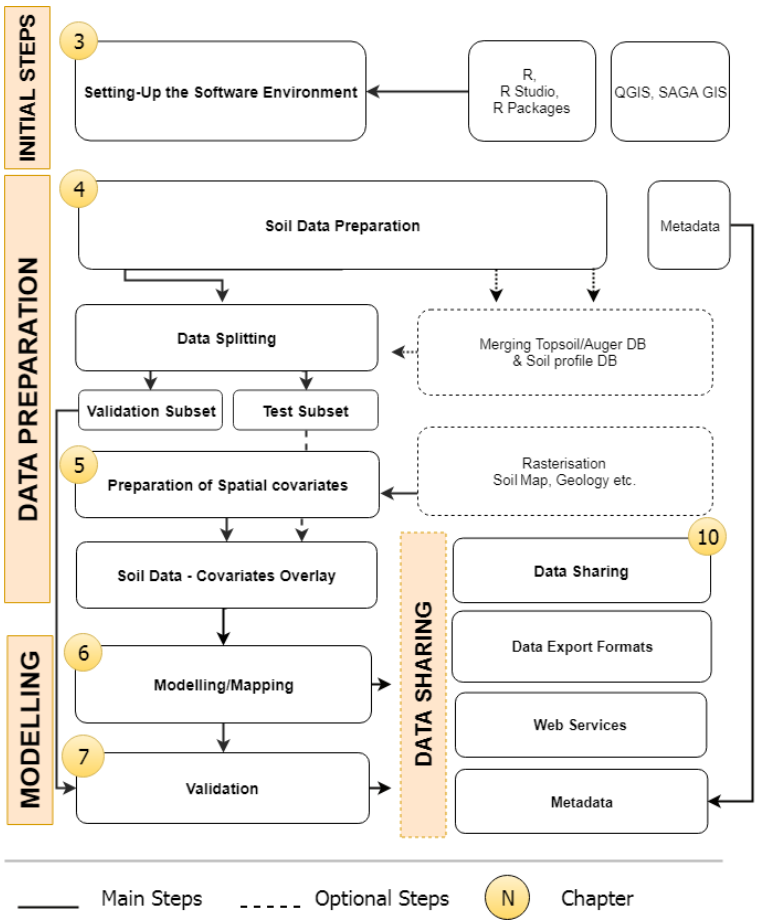


Figure 1.1: Proposed graphical workflow for producing a SOC map from data preparation to final result publishing

soil profiles with 11,000 horizons. The MASIS was developed with support from the FAO and the GSP. Decades of archived soil research have now been transformed into an accurate, up-to-date, and fully functioning information system, based on state-of-the-art digital soil mapping techniques.

Soils are a precious resource in this landlocked country, but climate change and other threats, including heavy human activities, are degrading the soils at an alarming rate. Without a proper evidence base such as a soil information system (SIS) , the FYROM had no systematic way of making decisions to promote sustainable soil management, or of monitoring soil conditions and functions over time. Since soil information is needed to guide sustainable soil and land management, this has contributed to stagnating agricultural development.

Bringing MASIS to life took just over two years and was completed in four phases. The first phase involved compiling, evaluating and systemizing more than 100 existing hardcopy soil maps that contained information on over 80% of the country, and soil legacy data from about 15,000 soil profiles. The missing 20% was complemented with an advanced soil survey.

The data were then used to develop a spatial geo-database within MASIS. This led to the production of a national soil map compliant with both, European and global standards, in phase two of the project.

The MASIS is available at the following link: http://www.maksoil.ukim.mk/.

1.3 Foreword to the first edition

The first edition of the cookbook provides step-by-step guidance for developing 1 km grids for the mapping of soil carbon stocks. It includes the preparation of local soil data, the compilation and pre-processing of ancillary spatial data sets, upscaling methodologies, and uncertainty assessments. Guidance is mainly specific to soil carbon data, but also contains many generic sections on soil grid development, as it is relevant for other soil properties.

Therefore, this first edition is the beginning of a series of updates and extensions, necessary to cover a larger variety of upscaling approaches. Experiences gained throughout 2017 during the GSOCmap programme, through applications at country scale and various trainings scheduled for 2017, shall be considered in the next editions. Also, the section on uncertainties will be adjusted to more practical implementation steps.

1.4 Foreword to the second edition

This second edition has been updated with knowledge and practical experiences gained during the implementation process of GSOCmap V1.0 throughout 2017. The cookbook includes step-by-step guidance for developing 1 km grids for SOC

stocks, as well as for the preparation of local soil data, the compilation and pre-processing of ancillary spatial data sets, upscaling methodologies, and uncertainty assessment methods. Guidance is mainly specific to soil organic carbon data, but as this cookbook contains generic sections on soil grid development, it can be applicable to map various soil properties.

Chapter 2

Soil property maps

R. Baritz

2.1 Definitions and objectives

Soil property maps represent spatial information about soil properties to a certain depth or for soil horizons. Conventionally, soil property maps are generated as polygon maps, with properties from typical soil profiles representing soil mapping units. Digital Soil Mapping (DSM) allows more accurate spatial mapping of soil properties, including the spatial quantification of the prediction error. The quality of such predictions improves with increasing number of local observations (e.g. soil profiles) available to build the prediction model. Whenever possible, DSM is recommended.

The development of soil property maps via DSM is spatially flexible. For different soil properties (e.g. concentration and stocks of nutrients in the soil, carbon, heavy metals, pH, cation exchange capacity, physical soil properties such as particle sizes and bulk density, etc.), various depth classes and spatial resolution can be modeled depending on project and mapping objectives and available input data. For GSOCmap, a 1 km grid is pursued. The same methodology and input data can also be used to produce higher resolution soil grids.

The mapping of global soil organic carbon (GSOC) stocks will be the first implementation of a series of other soil property grids to be developed for GLOSIS, based on the typical GSP country-driven system. GSOCmap will demonstrate the capacity of countries all around the globe to compile and manage national soil information systems and to utilize and evaluate these data following agreed international specifications. The GSP Secretariat, FAO, and its regional offices, as well as the Regional Soil Partnerships, are challenged together with the GSP members, especially the members of INSII, to establish national capacity and soil data infrastructures to enable soil property mapping.

2.2 Generic mapping of soil grids: upscaling of plot-Level measurements and estimates

The following table presents an overview of different geographic upscaling approaches, recommended to produce soil property maps, in particular, GSOCmap.

Conventional upscaling (Lettens et al., 2004)	Class-matching	Derive average SOC stocks per class: soil type for which a national map exists, or combination with other spatial covariates (e.g. land use category, climate type, biome, etc.). This approach is used in the absence of spatial coordinates of the source data.
	Geomatching	Point locations with spatial referencing are overlaid with geographic information system (GIS) layers of important covariates (e.g. a soil map). Upscaling is based on averaged SOC values per mapping unit.
Digital soil mapping (Dobos, 2006)	Data mining and geostatistics	Multiple regression, classification tree, random forests, regression-kriging, kriging with external drift.

Digital soil mapping is based on the development of functions for upscaling point data (with soil measurements) to a full spatial extent using correlated environmental covariates, for which spatial data are available.

> **DSM**: Concept of environmental correlation that explores the quantitative relationship between environmental variables and soil properties and could be used to predict the latter with multivariate prediction techniques.

The approaches outlined and demonstrated in this technical manual consider the reference framework of the SCORPAN model for digital soil mapping (DSM; McBratney et al. (2003)). 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 from the Dokuchaev hypothesis and Jenny's soil forming equation based on climate, organisms, relief, parent material and elapsed time of soil formation (Florinsky, 2012). The SCORPAN (Soils, Climate, Organisms, Parent material, Age and (N) space or spatial position, see McBratney et al. (2003)) reference framework is a empirical approach that can be expressed as in Eq. (1):

$$Sa_{[x;y\ t]} = f(S_{[x;y\ t]}, C_{[x;y\ t]}, O_{[x;y\ t]}, R_{[x;y\ t]}, P_{[x;y\ t]}, A_{[x;y\ t]}) \quad (2.1)$$

where Sa is the soil attribute of interest at a specific location N (represented by the spatial coordinates of field observations x; y) and representative for a specific time frame (t); S is the soil or other soil properties that are correlated with Sa; C is the climate or climatic properties of the environment; O are the organisms, vegetation, fauna or human activity; R is topography or landscape attributes; P is parent material or lithology; and A is the substrate age or the time factor. To generate predictions of Sa across places where no soil data is available, N should be explicit for the information layers representing the soil forming factors. These predictions will be representative of the time period (t) when soil available data was collected. Therefore, the prediction factors ideally should represent, the conditions of the soil forming environment for the same period of time (as much as possible) when soil available data was collected. In Eq. (1) the left side is usually represented by the available geo-spatial soil observational data (e.g., from legacy soil profile collections) and the right side of the equation is represented by the soil prediction factors. These prediction factors are normally derived from four main sources of information: a) thematic maps (i.e., soil type, rock type, land use type); b) remote sensing (i.e., active and passive); c) climate surfaces and meteorological data; and d) digital terrain analysis or geomorphometry. The SCORPAN reference framework is widely used, but one critical challenge is to quantify the relative importance of the soil forming factors (i.e., prediction factors) that could explain the underlying soil processes controlling the spatial variability of a specific soil attribute (i.e., SOC).

Chapter 3

Setting-up the software environment

Y. Yigini

This cookbook focuses on SOC modeling using open source digital mapping tools. The instructions in this chapter will guide user through installing and manually configuring the software to be used for DSM procedures for Microsoft Windows desktop platform. Instructions for the other platforms (e.g. Linux Flavours, MacOS) can be found through free online resources.

3.1 Use of R, RStudio and R Packages

R is a language and environment for statistical computing. It provides a wide variety of statistical (e.g. linear modeling, statistical tests, time-series, classification, clustering, etc.) and graphical methods, and is highly extensible.

3.1.1 Obtaining and installing R

Installation files and instructions can be downloaded from the Comprehensive R Archive Network (CRAN).

1. Go to the following link https://cloud.r-project.org/index.html to download and install **R**.
2. Pick an installation file for your platform.

3.1.2 Obtaining and installing RStudio

Beginners will find it very hard to start using **R** because it has no Graphical User Interface (GUI). There are some GUIs which offer some of the functionality of **R**. **RStudio** makes **R** easier to use. It includes a code editor, debugging and visualization tools. Similar steps need to be followed to install **RStudio**.

1. Go to https://www.rstudio.com/products/rstudio/download/ to download and install **RStudio**'s open source edition.
2. On the download page, *RStudio Desktop, Open Source License* option should be selected.
3. Pick an installation file for your platform.

3.1.3 Getting started with R

- **R** manuals: http://cran.r-project.org/manuals.html
- Contributed documentation: http://cran.r-project.org/other-docs.html
- Quick-**R**: http://www.statmethods.net/index.html
- Stackoverflow **R** community: https://stackoverflow.com/questions/tagged/r

3.2 R packages

When you download **R**, you get the basic **R** system which implements the **R** language. **R** becomes more useful with the large collection of packages that extend the basic functionality of it. **R** packages are developed by the **R** community.

3.2.1 Finding R packages

The primary source for **R** packages is CRAN's official website, where currently about 12,000 available packages are listed. For spatial applications, various packages are available. You can obtain information about the available packages directly on CRAN with the `available.packages()` function. The function returns a matrix of details corresponding to packages currently available at one or more repositories. An easier way to browse the list of packages is using the *Task Views* link, which groups together packages related to a given topic.

3.2.2 Most used R packages for digital soil mapping

As was previously mentioned, **R** is extensible through packages. **R** packages are collections of **R** functions, data, documentation and compiled code easy to share with others. In the following Subsections, we are going to present the most used packages related to digital soil property mapping.

3.2.2.1 Soil science and pedometrics

aqp: Algorithms for quantitative pedology. A collection of algorithms related to modeling of soil resources, soil classification, soil profile aggregation, and visualization.

GSIF: Global Soil Information Facility (GSIF). Tools, functions and sample datasets for digital soil mapping. GSIF tools (standards and functions) and sample datasets for global soil mapping.

ithir: A collection of functions and algorithms specific to pedometrics. The package was developed by Brendan Malone at the University of Sydney.

soiltexture: The *Soil Texture Wizard* is a set of **R** functions designed to produce texture triangles (also called texture plots, texture diagrams, texture ternary plots), classify and transform soil textures data. These functions virtually allow to plot any soil texture triangle (classification) into any triangle geometry (isosceles, right-angled triangles, etc.). The set of functions is expected to be useful to people using soil textures data from different soil texture classification or different particle size systems. Many (> 15) texture triangles from all around the world are predefined in the package. A simple text-based GUI is provided: `soiltexture_gui()`.

3.2.2.2 Spatial analysis

raster: Reading, writing, manipulating, analyzing and modeling of gridded spatial data. The package implements basic and high-level functions, processing of very large files is supported.

rgdal: Provides bindings to Frank Warmerdam's Geospatial Data Abstraction Library (GDAL).

RSAGA: The package provides access to geocomputing and terrain analysis functions of SAGA GIS from within **R** by running the command line version of System for Automated Geoscientific Analyses (SAGA).

sf: The package provides an improvement on **sp** and **rgdal** for spatial datasets. It is using the newly Simple Feratures (SF) standard which is widely implemented in spatial databases (PostGIS, ESRI ArcGIS), and forms the vector data basis for libraries such as GDAL and web standards such as GeoJSON (http://geojson.org/).

sp: The package provides classes and methods for spatial data. The classes document where the spatial location information resides, for 2D or 3D data.

3.2.2.3 Modelling

automap: This package performs an automatic interpolation by automatically estimating the variogram and then calling **gstat**.

caret: Extensive range of functions for training and plotting classification and regression models.

Cubist: Regression modeling using rules with added instance-based corrections. Cubist models were developed by Ross Quinlan.

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.

gstat: Variogram modeling with simple, ordinary and universal point or block (co)kriging, sequential Gaussian or indicator (co)simulation. The package includes variogram and variogram map plotting utility functions.

nnet: Software for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models.

3.2.2.4 Mapping and plotting

Both **raster** and **sp** have handy functions for plotting spatial data. The following packages can be used as functional extentions.

ggplot2: Besides using the base plotting functionality, this is another useful plotting package.

leaflet: Create and customize interactive maps using the Leaflet JavaScript library and the **htmlwidgets** package. These maps can be used directly from the **R** console, from **RStudio**, in **Shiny** apps and **RMarkdown** documents.

plotKML: Writes sp-class, spacetime-class, raster-class and similar spatial and spatiotemporal objects to KML following some basic cartographic rules.

3.2.3 Packages used in this cookbook

The authors of this cookbook used a number of different **R** packages. All required packages used in the cookbook can be installed using the following code and the `install.packages()` function when starting a new SOC mapping project. Alternatively, the code for the installation of the needed packages is included at the beginning of each Chapter.

```
# Install all required R packages used in the cookbook
install.packages(c("aqp", "automap", "car", "caret", "e1071",
                   "GSIF", "htmlwidgets", "leaflet", "mapview",
                   "Metrics", "openair", "plotKML", "psych",
                   "quantregForest", "randomForest", "raster",
                   "rasterVis", "reshape", "rgdal", "RSQLite",
                   "snow", "soiltexture", "sf", "sp"))
```

3.3 R and spatial data

R has a large and growing number of spatial data packages. We recommend taking a quick browse on **R**'s official website to see the spatial packages available: http://cran.r-project.org/web/views/Spatial.html.

3.3.1 Reading shapefiles

The Environmental Systems Research Institute (ESRI) provides a shapefile format SHP which is widely used for storing vector-based spatial data (i.e., points, lines, polygons). This example demonstrates use of **raster** package that provides functions for reading and/or writing shapefiles.

```
library(raster)
```

```
## Loading required package: sp
```

```
# Load the soil map from a shapefile *.shp file
soilmap <- shapefile("MK_soilmap_simple.shp")
```

We may want to use these data in other GIS environments such as ArcGIS, QGIS, SAGA GIS, etc. This means we need to export the `SpatialPointsDataFrame` to an appropriate spatial data format such as a shapefile `*.shp`.

```
# For example, we can select the soil units classified as
# fluvisols according to WRB
Fluvisols <- soilmap[soilmap$WRB == "Fluvisol",]

# Save this as a new shapefile
shapefile(Fluvisols, filename = 'results/fluvisols.shp',
          overwrite = TRUE)
```

3.3.2 Coordinate reference systems in R

We need to define the Coordinate Reference System (CRS) to be able to perform any sort of spatial analysis in **R**. To clearly tell **R** this information we define the CRS which describes a reference system in a way understood by the PROJ.4 projection library. Find more information at this link: http://trac.osgeo.org/proj/.

An interface to the PROJ.4 library is available in the **rgdal** package. An alternative to using PROJ.4 character strings, we can use the corresponding yet simpler EPSG (European Petroleum Survey Group) code. **rgdal** also recognizes these codes. If you are unsure of the PROJ.4 or EPSG code for the spatial data that you have but know the CRS, you should consult http://spatialreference.org/ for assistance.

```
# Print the CRS for the object soilmap
soilmap@proj4string
```

```
## CRS arguments:
##  +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84
## +towgs84=0,0,0
```

The following example shows how a spatial object from a (comma-separated values) CSV file (`*.csv`) can be created. We can use the `coordinates()` function from the **sp** package to define which columns in the data frame refer to actual spatial coordinates. Here, the coordinates are listed in columns X and Y.

```
# Load the table with the soil observations site information
dat_sites <- read.csv(file = "data/site-level.csv")

# Convert from table to spatial points object
coordinates(dat_sites) <- ~ X + Y

# Check the coordinate system
dat_sites@proj4string
```

```
## CRS arguments: NA
```

```
# As the CRS is not defined, we can assign the correct CRS as we
# have information about it. In this case, it should be EPSG:4326
dat_sites@proj4string <- CRS("+init=epsg:4326")

# Check the CRS again
dat_sites@proj4string
```

```
## CRS arguments:
##  +init=epsg:4326 +proj=longlat +datum=WGS84 +no_defs
## +ellps=WGS84 +towgs84=0,0,0
```

3.3.3 Working with rasters

Most of the functions for handling raster data are available in the **raster** package. There are functions for reading and writing raster files from and to different formats. In DSM, we mostly work with data in table format and then rasterize this data so that we can produce a continuous map. For doing this in **R** environment, we will load raster data in a data frame. This data is a digital elevation model (DEM) provided by the International Soil Reference and Information Centre (ISRIC) for FYROM.

```
# For handling raster data, we load raster package
library(raster)

# Load DEM from the raster *.tif files
DEM <- raster("covs/DEMENV5.tif")
```

We may want to export this raster to a suitable format to work in a standard GIS environment. See the help file for writing a raster `?writeRaster` to get information

regarding the supported grid types that data can be exported into. Here, we will export our raster to ESRI ASCII (American Standard Code for Information Interchange), as it is a common and universal raster format.

We may also want to export our DEM to KML (Keyhole Markup Language) file (`*.kml`) using the `KML()` function. `KML()` is a handy function from the **raster** package for exporting grids to KML format. Note that we need to re-project the data to the World Geodetic System (WGS) WGS84 geographic. The raster re-projection is performed using the `projectRaster()` function. Look at the help file `?projectRaster` for more information.

3.4 Other DSM software and tools

- **GRASS GIS**: Available at https://grass.osgeo.org/ (free and open source).
- **SAGA GIS**: Available at https://sourceforge.net/projects/saga-gis/files/ (free and open source).
- **QGIS**: Available at http://www.qgis.org/en/site/forusers/download.html (free and open source).

Chapter 4

Preparation of local soil data

G.F. Olmedo & R. Baritz

The authors of this Chapter used **R** packages. To run the code provided in this Chapter, the following packages need to be installed in the **R** user library. If the packages are not yet installed, the `install.packages()` function can be used.

```
# Installing packages for Chapter 'Preparation of Local Soil Data'
install.packages(c("aqp", "GSIF"))
```

4.1 Soil profiles and soil augers

Soil profiles are complex real-world entities. They are composed of soil layers which form soil horizons; the soil layers have different properties and these properties are evaluated with different methods. As we know, soil and vertical soil properties are landscape elements and part of matter dynamics (water, nutrients, gases, habitat, etc.). Local soil samples or soil profiles add a third dimension into the spatial assessment of soil properties in the landscape.

Most commonly, soils are described as vertical profiles using soil pits (sometimes also augerings, but this is less accurate). Soil profiles are described using macro-morphological properties. These properties can be assessed in the field without analysis by making a field inventory or land evaluation. For additional quantitative analysis, soils are then sampled by genetic horizons or by depth class.

The sampling of soils is the basis to obtain quantitative information. Depending on the goal of a project, sampling can be quite diverse. Sampling can follow the description of the soil or can be conducted without, for example using a spade or auger to generate a composite sample for a certain depth independent of the morphological features such as soil horizons. Sampling locations can be representative of a certain location, project, field, or mapped object, such as a soil type.

Table 4.1: Example for site-level data table

ProfID	X_coord	Y_coord	Year	Soil_Type
P1276	7591265	4632108	2012	Complex of Chernozem ...
P1277	7592027	4631664	2012	Complex of Chernozem ...
P1278	7592704	4631941	2012	Complex of Chernozem ...
P1279	7590817	4633115	2013	Complex of Chernozem ...

4.2 Soil database

4.2.1 Type of soil database

In order to process and evaluate soil information from field assessments, soil profile data and analytical information need to be stored in a database. This can be a set of simple Microsoft Office *Excel* spreadsheets, or a relational or object-oriented database management system (Baritz et al., 2008). When working in **R**, `SoilProfileCollections` (SPC) from the **R aqp** package are a useful tool. Tables 4.1 and 4.2 are examples of how soil information can be stored. The advantage of such organization is the possibility to develop relational databases which can be easily queried. Such a systematic approach will support the organization of national soil information and will reduce errors in future modeling exercises (Baritz et al., 2008).

Table 4.1 stores site-level data, which describe the location of the soil description and/or sampling site: spatial coordinates, landscape attributes such as slope gradient and slope form, soil class, land cover type, rock type, etc. In this table, every row should hold a single soil profile. One column, usually the first one, should be the soil profile's unique identifier. Using the latter, soil information can be easily linked from one table to another.

Table 4.2 stores information from the soil description, such as horizon name, horizon thickness, organic matter content, carbonate content, soil color, laboratory soil analysis, etc. The first column contains the soil profile's unique identifier. It is important to include the upper and lower limits for each soil layer; in case the sampling strategy deviates from soil layers/soil horizons, the upper and lower depth of the sampling locations should be specified if possible. This information is needed for modeling soil properties over the soil profile.

4.2.2 Technical steps – Loading soil data from tables in R

This Chapter includes two examples for soil data preparation. As each step for the soil data preparation is explained in detail, the given code is mixed with text. A copy of the bare code is presented in Section 12.1 for using soil profiles data, and for using topsoil or auger data the code can be found in Section 12.2.

Table 4.2: Example for profile-description table

ProfID	HorID	top	bottom	SOC	BLD	CRF	Sand	Silt	Clay
P1276	P1276H01	0	50	2.78	1.05	11	52	39	9
P1276	P1276H02	50	76	1.75	1.45	4	56	31	14
P1276	P1276H03	76	100	1.19	1.22	2	43	35	22
P1277	P1277H01	0	28	1.93	1.36	8	59	22	18
P1277	P1277H02	28	48	1.60	1.43	9	69	15	16
P1277	P1277H03	48	63	1.26	NA	25	65	21	13
P1277	P1277H04	63	120	0.86	NA	54	63	23	14
P1278	P1278H01	0	40	2.32	1.27	0	50	39	12
P1278	P1278H02	40	68	1.80	1.48	1	46	39	16
P1278	P1278H03	68	120	0.89	1.18	0	47	39	14

The following code shows the necessary steps for loading soil profiles data.

Step 1 - Loading soil horizons data

```
dat <- read.csv(file = "data/horizons.csv")

# Explore the data
str(dat)
summary(dat)
```

```
## 'data.frame':    10292 obs. of  10 variables:
##  $ ProfID: Factor w/ 4118 levels "P0000","P0001",..: 1 1 1 2 2 ..
##  $ HorID : Factor w/ 9914 levels "P0000H01","P0000H02",..: 1 2 ..
##  $ top   : int  4 23 46 2 11 0 22 63 0 3 ...
##  $ bottom: int  23 46 59 11 31 22 63 90 19 10 ...
##  $ SOC   : num  NA NA NA NA NA ...
##  $ BLD   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ CRF   : num  54 62 47 66 70 57 77 87 8 4 ...
##  $ SAND  : int  52 59 67 45 40 52 48 43 50 48 ...
##  $ SILT  : num  34 31 24 39 31 33 36 42 16 35 ...
##  $ CLAY  : num  14 11 8 16 28 15 16 16 34 17 ...
##      ProfID           HorID            top
##  P2881  :   64   P2881H01:   64   Min.   :  0.00
##  P1481  :   32   P0434H02:    8   1st Qu.:  0.00
##  P2096  :   32   P1286H01:    8   Median : 20.00
##  P3623  :   32   P2056H01:    8   Mean   : 27.48
##  P2056  :   24   P2056H02:    8   3rd Qu.: 47.00
##  P2142  :   24   P2056H03:    8   Max.   :285.00
##  (Other):10084   (Other) :10188
##      bottom            SOC              BLD
##  Min.   :  1.00   Min.   : 0.000   Min.   :0.00
##  1st Qu.: 25.00   1st Qu.: 1.090   1st Qu.:1.40
```

```
##  Median : 45.00   Median : 1.800   Median :1.54
##  Mean   : 55.82   Mean   : 2.603   Mean   :1.55
##  3rd Qu.: 80.00   3rd Qu.: 2.940   3rd Qu.:1.66
##  Max.   :295.00   Max.   :83.820   Max.   :2.93
##                   NA's   :831      NA's   :7845
##       CRF             SAND             SILT
##  Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.:  2.0   1st Qu.: 44.00   1st Qu.:16.00
##  Median :  8.0   Median : 58.00   Median :24.00
##  Mean   : 13.5   Mean   : 57.67   Mean   :25.64
##  3rd Qu.: 21.0   3rd Qu.: 72.00   3rd Qu.:33.00
##  Max.   :104.0   Max.   :100.00   Max.   :80.00
##  NA's   :3360    NA's   :1049     NA's   :920
##       CLAY
##  Min.   : 0.00
##  1st Qu.: 7.00
##  Median :14.00
##  Mean   :17.08
##  3rd Qu.:24.00
##  Max.   :83.00
##  NA's   :927
```

Step 2 - Loading site-level data

```
dat_sites <- read.csv(file = "data/site-level.csv")

# Explore the data
str(dat_sites)
```

```
## 'data.frame':    4118 obs. of  6 variables:
##  $ X.1       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ProfID    : Factor w/ 4118 levels "P0000","P0001",..: 1 2 3 ..
##  $ soiltype  : Factor w/ 58 levels "Albic Luvisol",..: 3 3 3 57..
##  $ Land.Cover: int  25 24 25 26 26 27 27 27 22 23 ...
##  $ X         : num  20.8 20.8 20.8 20.8 20.8 ...
##  $ Y         : num  42 42 42 42 42 ...
```

4.3 Completeness of measurements and estimates

The *GSP Guidelines for Sharing National Data/Information to Compile a Global Soil Organic Carbon (GSOC) Map* (FAO and GSP, 2017a) specify which soil parameters are needed to produce a GSOCmap. Of course, other soil properties can be evaluated and modeled using this cookbook as well.

SOC stocks for soil horizons or targeted soil depths can be calculated using the equations in Section 8.4.3 of the *GSP Guidelines for Sharing National Data/Information to Compile a Global Soil Organic Carbon (GSOC) Map*.

Carbon concentration, bulk density and stone content for a certain depth or genetic horizon are needed to calculate the amount of carbon stored in that depth interval/soil horizon. In many countries, legacy data from former surveys and projects, as well as from various owners and data sources are compiled. Often, measured bulk densities are either missing, only available for few soil profiles, or are estimated. Stones in the soil profile are usually only estimated, and if augers are used for sampling, stone content is not assessed at all. Pedo-transfer functions (PTFs) can be used to fill data gaps (e.g. bulk density), and interpolation approaches can be used to infer from measured depths to target depths.

4.3.1 Stones

The estimation of stoniness is difficult and time-consuming, and therefore not carried out in many national soil inventories, or only estimated visually in the profile. Unfortunately, if soil inventories and sampling are done with simple pits or augers rather than standard soil pits, stones are very often not assessed.

As a proxy, it is recommended to derive national default values from well-described soil profile pits by soil type.

```
# Summary of column CRF (coarse fragments) in the example data base
summary(dat$CRF)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     0.0     2.0     8.0    13.5    21.0   104.0    3360
```

```
# Convert NA's to 0
dat$CRF[is.na(dat$CRF)] <- 0

hist(dat$CRF, col = "light gray")
```

4.3.2 Bulk density

The amount of fine earth is one of the basic estimation parameters to estimate SOC stocks in the mineral soil as well as in peat layers. It depends on the volume of soil considered (depth × reference area) and the bulk density (BD). BD expresses the soil weight per unit volume. When determining the BD, it is important to subtract stones, if any, from the cylinder samples; if this is not done, BD is underestimated, and the resulting SOC stocks are overestimated. Stones in the cylinders are added to the total stone content in order to correct for the total amount of fine earth per volume of soil in a given area.

Most of the soil profiles in national databases come from agricultural land. Very often, BD estimates do not consider fine stones because top soils (e.g. plough layers) seem to be free of visible stones.

Mineral soil: Default values from the *General Guide for Estimating Moist Bulk Density* given by United States Department of Agriculture. Soil Conservation

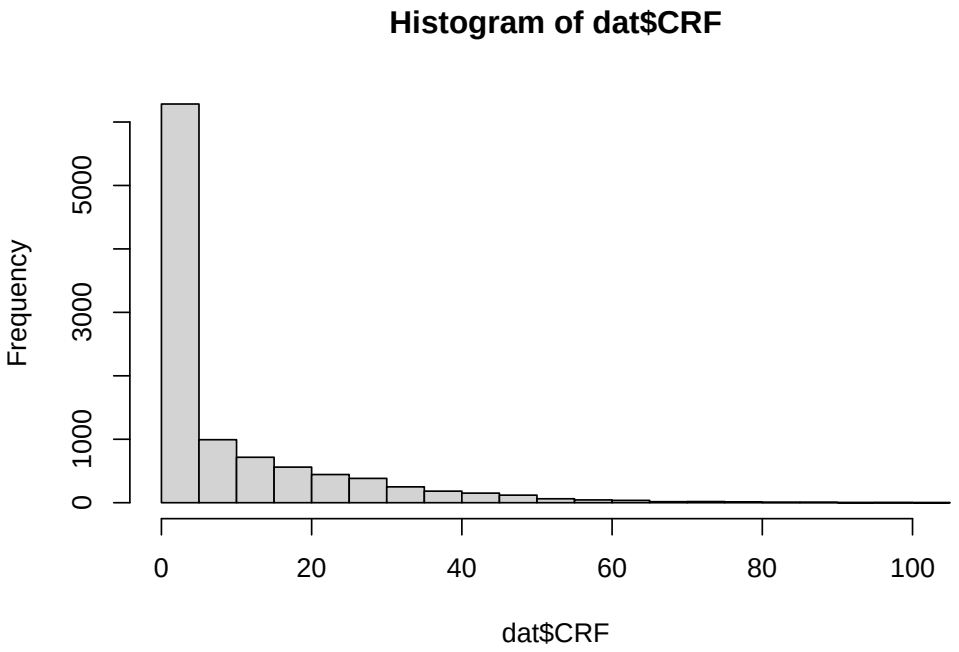


Figure 4.1: Histogram of coarse fragments values

Service (2018). If analytical BD is missing, BD can be estimated using pedotransfer functions (see examples listed below).

For organic soil material, BD_x can be estimated as follows, considering existing litter layers L (or Oi horizon); organic or duff layers, partially decomposed material above the mineral soil and beneath the litter layer; fermentation horizons F;humus horizons H (or Oe and Oa horizons); and peat layers P, as described in the *U.S. Soil Taxonomy* (United States Department of Agriculture. Soil Conservation Service, 1975).

Forest floor: Default values from Barney et al. (1981) and Ottmar and Andreu (2007):

- Pine: $BD_L = 0.018g \cdot cm^{-3}$; $BD_{F,H} = 0.057g \cdot cm^{-3}$
- Hardwood: $BD_L = 0.012g \cdot cm^{-3}$
- Birch: $BD_{F,H} = 0.17g \cdot cm^{-3}$
- Spruce: $BD_L = 0.051g \cdot cm^{-3}$; $BD_H = 0.13g \cdot cm^{-3}$

Peat: The range of peat BD is generally from 0.02 to $0.3t \cdot m^{-3}$ depending on maturity and compaction, as well as the ash content (Agus et al., 2011). total and Agus et al. (2011) distinguish different peat decomposition types with different C content:

- Sapric: $BD_{P,\text{sapric}} = 0.174g \cdot cm^{-3}$ (48.90% C)
- Hemic: $BD_{P,\text{hemic}} = 0.117g \cdot cm^{-3}$ (52.27% C)
- Fibric: $BD_{P,\text{fibric}} = 0.089g \cdot cm^{-3}$ (53.56% C)

Example equations for PTFs to estimate BD, based on the soil organic matter (SOM) (OM) content in percent (%):

Saini (1966):

$$BD = 1.62 - 0.06 * OM \tag{4.1}$$

Drew (1973):

$$BD = 1/(0.6268 + 0.0361 * OM) \tag{4.2}$$

Jeffrey (1970):

$$BD = 1.482 - 0.6786 * (logOM) \tag{4.3}$$

Grigal et al. (1989):

$$BD = 0.669 + 0.941 * e^{(-0{,}06*OM)} \tag{4.4}$$

Adams (1973):

$$BD = 100/(OM/0.244 + (100 - OM))/MBD \tag{4.5}$$

Honeysett and Ratkowsky (1989):

$$BD = 1/(0.564 + 0.0556 * OM) \tag{4.6}$$

where MDB is the mineral particle density, assumed to be the specific gravity of quartz, $2.65Mg \cdot m^{-3}$. And OM is the SOM content, estimated as $OM = SOC \cdot 1.724$, with SOC content in percent (%).

Each method to estimate the BD presented in is derived from a specific set of regional soils that is regionally adapted. Selection of the proper method for a given country shall be based on existing reviews and comparisons.

```
# Creating a function in R to estimate BLD using the SOC
# SOC is the soil organic carbon content in percent %
estimateBD <- function(SOC, method="Saini1996"){
  OM <- SOC * 1.724
  if(method=="Saini1996"){BD <- 1.62 - 0.06 * OM}
  if(method=="Drew1973"){BD <- 1 / (0.6268 + 0.0361 * OM)}
```

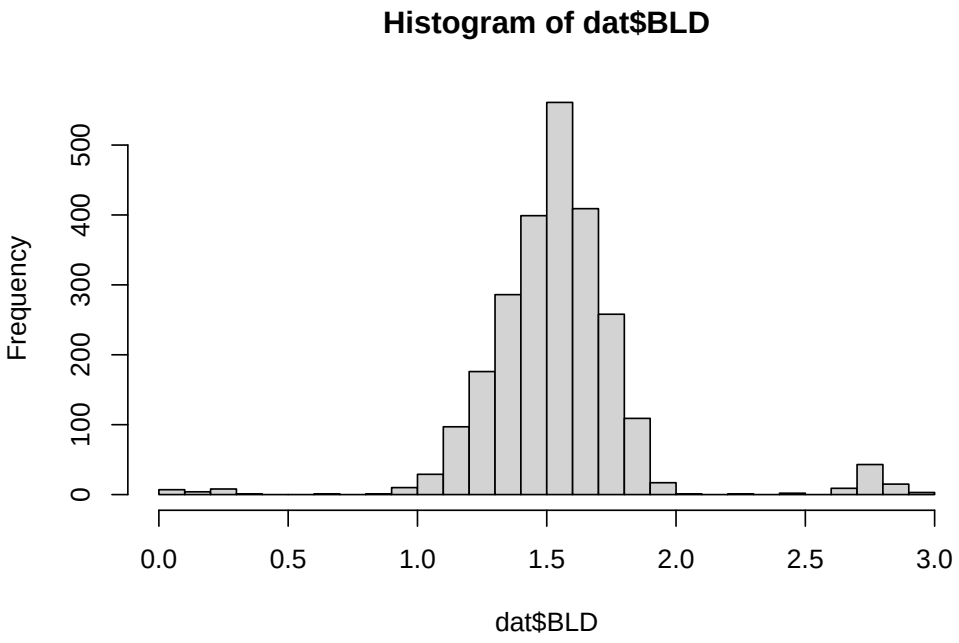


Figure 4.2: Histogram of bulk density values

```
  if(method=="Jeffrey1979"){BD <- 1.482 - 0.6786 * (log(OM))}
  if(method=="Grigal1989"){BD <- 0.669 + 0.941 * exp(1)^(-0.06 * OM)}
  if(method=="Adams1973"){BD <- 100 / (OM /0.244 + (100 - OM)/2.65)}
  if(method=="Honeyset_Ratkowsky1989"){BD <- 1/(0.564 + 0.0556 * OM)}
  return(BD)
}
```

```
# Summary of BLD (bulk density) in the example data base
summary(dat$BLD)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00    1.40    1.54    1.55    1.66    2.93    7845
```

```
hist(dat$BLD, col = 'light gray', breaks = 32)
```

Exploring SOC and bulk density values in the database is an important step for ensuring quality of the map. Digital soil mapping framework is a data-driven approach to modelling spatial distribution of soil properties, therefore any irregularities or errors in the input data may significantly influence modelling results. In the current example database we can observe bulk density values as low as 0 kg/m3 and as high as 2.93 g/cm3. Having 0 g/ m3 bulk density values is physically impossible, and the values higher than 2 g/ m3 are not typical for fine earth (**?**) and most likely correspond to coarse fragments. The irregularities in the data

may occur due to errors in field measurement procedures, wrong unit conversion or simply typing mistakes. In case there is doubt in the quality of the input data, the questionable values should be thoroughly checked and, if found erroneous, removed from the database.

```
# remove bad values
dat$BLD[dat$BLD == 0] <- NA
dat$BLD[dat$BLD > 2] <- NA
summary(dat$BLD)
```

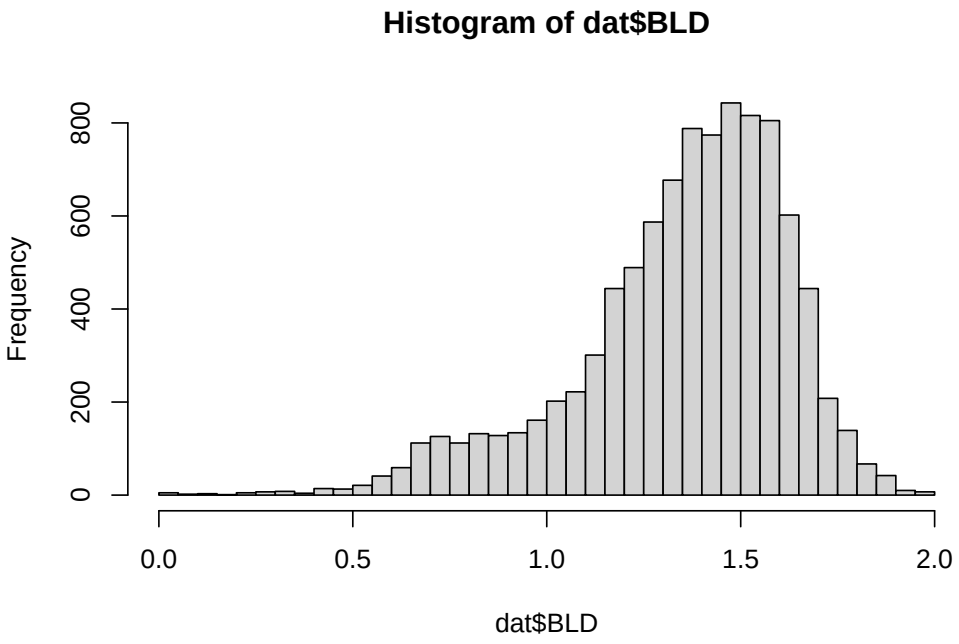
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.080   1.400   1.540   1.516   1.650   1.990    7924
```

```
# See the summary of values produced using the pedo-transfer
# function with one of the proposed methods
summary(estimateBD(dat$SOC[is.na(dat$BLD)],
                   method="Honeyset_Ratkowsky1989"))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.2019  1.1625  1.3507  1.2987  1.4981  1.7730     737
```

```
# Fill NA's using the pedo-transfer function
dat$BLD[is.na(dat$BLD)] <- estimateBD(dat$SOC[is.na(dat$BLD)],
                                      method="Honeyset_Ratkowsky1989")

# Explore the results
hist(dat$BLD, col = 'light gray', breaks = 32)
```



4.3.3 Soil carbon analysis

Rosell et al. (2001) have closely reviewed the different SOC and SOM estimation procedures, and have also drawn some conclusions about the sources of errors. Determination of SOC from dry combustion methods is least susceptible to errors.

- **Dry combustion by Loss on Ignition (LOI)**: SOC is re-calculated applying a conversion factor. It is commonly assumed, that organic matter contains an average of 58% organic carbon (so-called Van Bemmelen factor 1.724; for non-organic horizons: $SOC = SOM/1.724$). For organic horizons, conversion factor ranges from 1.9 to 2.5 (Nelson and Sommers, 1982). The inorganic carbon is not resolved, since typically, temperatures between 400°C and 550°C are used.
- **Wet oxidation**: Since wet oxidation is applied without additional (external) heating, low temperatures of around 120°C (internal heat) are typical. Thus, the oxidation of carbon is incomplete, and a so-called oxidation factor needs to be applied. With external heating, the C-recovery of the method becomes improved, up to complete recovery. No correction of the mineral carbon is needed. Wet oxidation should typically only be applied to samples with <5% organic matter.

Usually, an average of 76% organic carbon is recovered, leading to a standard oxidation factor of 1.33 (Lettens et al., 2005).

4.3.4 Carbonates

In case the total organic carbon is determined with temperatures higher than 600°C to 800°C, the proportion of mineral soil in $CaCO_3$ has to be subtracted in order to derive the amount of organic carbon (inorganic carbon is also oxidized). The pH value gives the first indication whether the sample has to be analyzed for inorganic carbon or not.

It is crucial to report in the metadata whether national SOC values refer to total C or if the inorganic component has been considered.

4.3.5 Depth

The standard depth for GSOCmap is **0 cm to 30 cm** (FAO and GSP, 2017a). Subdivisions are possible depending on the available data, by genetic horizons or depth classes. The following depths are additionally considered for GSOCmap (optional):

- **Forest floor**: Thickness (cm) subdivision in horizons depending on national soil inventory method (e.g. for forest floors organic layers L, F, H).
- **Peat**: From 30cm to 100cm, depending on national data.

4.4 Soil depth estimate

4.4.1 Completeness of depth estimate

Soil properties are commonly collected from field inventories (see Table 4.2) or from sampling and analyzing horizons and/or fixed depths. Since a fixed target depth of 30 cm is required for GSOC (other depth classes will be recommended in the future, following the *GSP Guidelines for Sharing National Data/Information to Compile a Global Soil Organic Carbon (GSOC) Map*, data holders are confronted with the following options:

- **Option 1**: Soil sampling has already considered this depth, data can be directly used for upscaling (see Chapter 6).
- **Option 2**: Horizons or layers/depth classes are sampled but aggregation is needed over the 0cm to 30cm.
- **Option 3**: The target depth (0cm to 30cm) was not completely covered by sampling, e.g. only the A horizon or a topsoil layer (e.g. 0cm to 20cm) has been sampled.

For both **Options 2** and **Option 3**, the transformation is needed, using e.g. equal-area splines. In the case of **Option 2**, the use of equal-area splines was first proposed by Ponce Hernandes et al. (1986), and later tested against real data (Bishop et al., 1999). This technique is based on fitting continuous depth functions for modeling the variability of soil properties with depth. Thus, it is possible to convert soil profiles to standard depths, but also to fill gaps. The equal-area spline function consists of a series of local quadratic polynomials that join at so-called knots, located at the horizon boundaries, whereby the mean value of each horizon is maintained by the spline fit. They are called equal-area splines because the area to the left of the fitted spline curve is equal to the area to the right of the curve.

In case of **Option 3**, additional information on the vertical distribution of carbon in the soils is required for accurate recalculation from the sampling depth to target depth, e.g. as was shown by Bernoux et al. (1998).

4.4.2 Technical steps - Equal-area splines using R

In **R** environment, the easiest way to apply equal-area splines is using the function `GSIF::mpspline` from the **R** package **GSIF** ((Hengl, 2016), see Section 3.2.2.1). For illustration, a sample dataset has been used (see Chapter 5). This function requires data stored as `SoilProfileCollection` (SPC) using package **aqp**. Nevertheless, data in any local soil database or in tables like the ones proposed before (see Tables 4.1, 4.2) can be transformed to an SPC.

The function `GSIF::mpspline` has several arguments. One of the arguments is the lambda value mentioned before. The proposed default value is 0.1. Another argument for this function is the target standard depths. The function produces spline-estimated values at these depths. However, this function also produces spline-estimated values at 1cm increments.

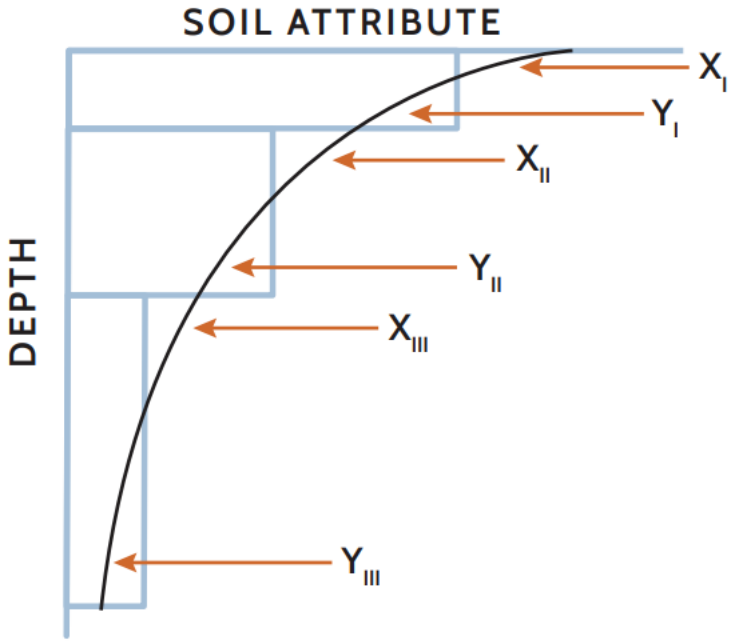


Figure 4.3: An equal-area quadratic spline from Ponce-Hernandez et al. (1986) (Cited by Bishop et al., 1999)

The following technical steps require **R** and the named packages.

Step 1 - Promote data table to SPC

```
# Load aqp package
library(aqp)

# Promote to SoilProfileCollection
# The SoilProfileCollection is a object class in R designed to
# handle soil profiles
depths(dat) <- ProfID ~ top + bottom
```

```
## Warning: converting IDs from factor to character
```

Step 2 - Add site-level data and coordinates

```
# Merge the soil horizons information with the site-level
# information from dat_sites
site(dat) <- dat_sites

# Set spatial coordinates
coordinates(dat) <- ~ X + Y

# A summary of our SoilProfileCollection
```

```
dat
```

```
## Object of class SoilProfileCollection
## Number of profiles: 4118
## Depth range: 5-295 cm
##
## Horizon attributes:
##   ProfID    HorID top bottom SOC BLD CRF SAND SILT CLAY
## 1  P0000 P0000H01   4     23  NA  NA  54   52   34   14
## 2  P0000 P0000H02  23     46  NA  NA  62   59   31   11
## 3  P0000 P0000H03  46     59  NA  NA  47   67   24    8
## 4  P0001 P0001H01   2     11  NA  NA  66   45   39   16
## 5  P0001 P0001H02  11     31  NA  NA  70   40   31   28
## 6  P0002 P0002H01   0     22  NA  NA  57   52   33   15
##
## Sampling site attributes:
##   ProfID X.1                                  soiltype Land.Cover
## 1  P0000   1                                  Cambisol         25
## 2  P0001   2                                  Cambisol         24
## 3  P0002   3                                  Cambisol         25
## 4  P0003   4                       Rendzic Leptosols           26
## 5  P0004   5                       Rendzic Leptosols           26
## 6  P0005   6 Complex of Rendzic Leptosol and Leptosol           27
##
## Spatial Data:
##        min      max
## X 20.46434 23.01039
## Y 40.68543 42.35932
## [1] NA
```

Step 3 - Run mass preserving splines for all the needed properties

```
library(GSIF)

# Estimate 0 cm - 30 cm standard horizons
# using mass preserving splines
try(SOC <- mpspline(dat, 'SOC', d = t(c(0,30))))
try(BLD <- mpspline(dat, 'BLD', d = t(c(0,30))))
try(CRFVOL <- mpspline(dat, 'CRF', d = t(c(0,30))))
```

Step 4 - Convert back to table

```
# Prepare final data frame
dat <- data.frame(id = dat@site$ProfID,
                  Y = dat@sp@coords[,2],
                  X = dat@sp@coords[,1],
                  SOC = SOC$var.std[,1],
                  BLD = BLD$var.std[,1],
                  CRFVOL = CRFVOL$var.std[,1])
```

```
dat <- dat[complete.cases(dat),]

# Take a look at the results
head(dat)
```

```
##       id        Y        X       SOC       BLD    CRFVOL
## 4  P0003 42.02828 20.81819 26.380000 0.7304483  8.000000
## 5  P0004 42.02747 20.81464 24.561667 0.8955320  6.305316
## 7  P0006 42.02885 20.82757 19.940000 0.7886235 14.000000
## 8  P0007 42.02279 20.83165  6.149114 1.1653355 18.633590
## 9  P0008 42.05014 20.82036  3.940352 1.2962582 31.875748
## 10 P0009 42.02047 20.93529  3.258545 1.3446297 21.714059
```

Step 5 - Estimate the soil organic carbon stock using the virtual horizons

Finally, the estimation of the soil organic carbon stock (OCS) can be done using the **GSIF** package.

```
library(GSIF)

# Estimate organic carbon stock (OCS)
# SOC must be in g/kg
# BLD in kg/m3
# CRF in percentage %
OCSKGM <- OCSKGM(ORCDRC = dat$SOC, BLD = dat$BLD*1000,
                 CRFVOL = dat$CRFVOL, HSIZE = 30)

dat$OCSKGM <- OCSKGM
dat$meaERROR <- attr(OCSKGM,"measurementError")
dat <- dat[dat$OCSKGM>0,]

summary(dat)
```

```
##        id             Y               X              SOC
##  P0003  :   1   Min.   :40.69   Min.   :20.46   Min.   : 0.080
##  P0004  :   1   1st Qu.:41.19   1st Qu.:21.35   1st Qu.: 1.750
##  P0006  :   1   Median :41.42   Median :21.49   Median : 2.603
##  P0007  :   1   Mean   :41.49   Mean   :21.66   Mean   : 3.539
##  P0008  :   1   3rd Qu.:41.83   3rd Qu.:22.14   3rd Qu.: 4.089
##  P0009  :   1   Max.   :42.36   Max.   :23.01   Max.   :86.510
##  (Other):3880
##       BLD              CRFVOL             OCSKGM
##  Min.   :0.05353   Min.   : 0.00000   Min.   :0.0111
##  1st Qu.:1.28089   1st Qu.: 0.00516   1st Qu.:0.6951
##  Median :1.39549   Median : 5.01397   Median :0.9866
##  Mean   :1.38073   Mean   :11.02257   Mean   :1.1399
##  3rd Qu.:1.47612   3rd Qu.:17.69065   3rd Qu.:1.3997
##  Max.   :2.92191   Max.   :93.95013   Max.   :7.4302
```

```
##
##     meaERROR
##  Min.   :0.172
##  1st Qu.:3.160
##  Median :3.840
##  Mean   :3.715
##  3rd Qu.:4.260
##  Max.   :8.770
##
```

Soil organic carbon tends to have a log-normal distribution with a right-skew, and transforming the original values to its natural logarithm would generate a normal distribution of SOC values. Here we will test:

1. If the log-transformation of the response variable (SOC) tends to normality; and
2. If this transformation increases the simple correlation of SOC and its prediction factors.

```
# Generate a new column with the transformed OCSKGM to its natural
# logarithm
dat$OCSKGMlog <- log(dat$OCSKGM)

# Plot the next two plots as one
par(mfrow=c(1,2))
plot(density(dat$OCSKGM),
     main='Original values')

plot(density(dat$OCSKGMlog),
     main='Log-transformed values')
```

```
par(mfrow=c(1,1))

# We can save our processed data as a *.csv table
write.csv(dat, "data/dataproc.csv", row.names = FALSE)
```

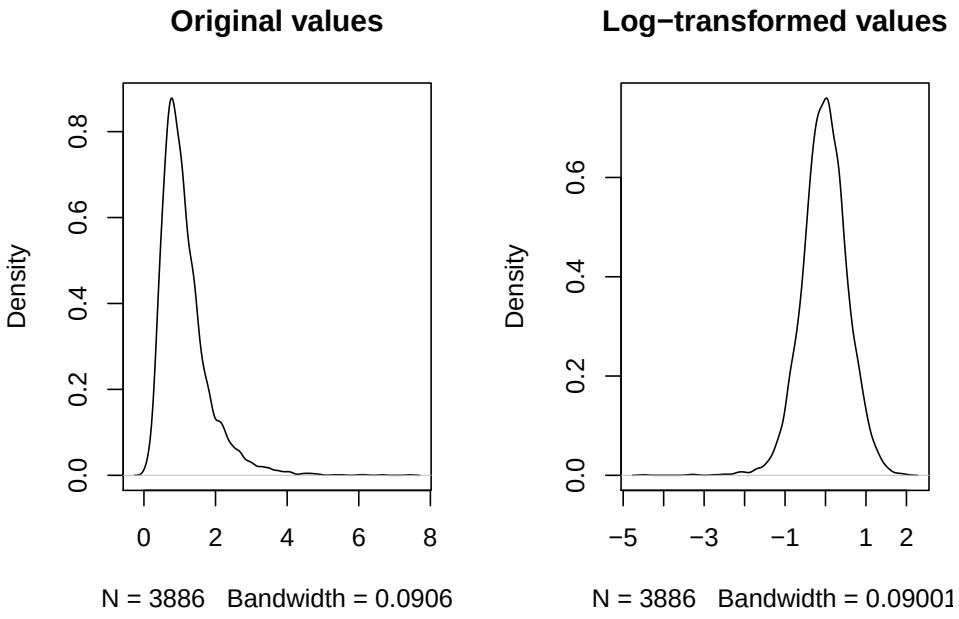


Figure 4.4: Statistical distribution of original values vs. log-transformed values for OCS

Chapter 5

Preparation of spatial covariates

R. Baritz & Y. Yigini

The authors of this Chapter used **R** packages. To run the code provided in this Chapter, the following package need to be installed in the **R** user library. If the package is not yet installed, the `install.packages()` function can be used.

```
# Installing package for Chapter 'Preparation of Spatial Covariates'
install.packages("raster")
```

The example covariates from this Chapter were prepared by ISRIC. The access and use limitations are presented in Section 5.6. A small subset of these covariates, comprising most of the soil forming factors, will be used for the code examples. This subset is presented in the following table.

5.1 DEM-derived covariates

This Section gives a short overview on available DEM source data sets. 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 at the global scale, which are freely available for users. Both DEMs were compared by Wong et al. (2014). Comparison against high-resolution topographic data of Light Detection and Ranging (LiDAR) in a mountainous tropical montane landscape showed that the SRTM (90 m) produced better topographic data in comparison with ASTER GDEM.

- Recommended for **national level** applications: ASTER GDEM/SRTM with 30 m resolution.

Table 5.1: Code and name of example covariates provided with the cookbook

CODE	ATTRIBUTE_TITLE
DEMENV5	Land surface elevation
SLPMRG5	Terrain slope
VBFMRG5	Multiresolution Index of Valley Bottom Flatness (MRVBF)
VDPMRG5	Valley depth
TWIMRG5	SAGA Wetness Index
TMDMOD3	Mean annual LST (daytime) MODIS
TMNMOD3	Mean annual LST (nighttime) MODIS
PRSCHE3	Total annual precipitation at 1 km
B04CHE3	Temperature seasonality at 1 km
B07CHE3	Temperature Annual Range at 1 km
B13CHE3	Precipitation of wettest month [mm]
B14CHE3	Precipitation of driest month [mm] at 1 km
LCEE10	ESA land cover map 2010

- Recommended for **global level** applications: SRTM with 90 m resolution, resampled to 1 km.

In both cases, noise and artefacts need to be filtered out. ASTER GDEM seems to contain more large artifacts (e.g. peaks), particularly in flat terrain, which are very difficult to remove through filtering.

```
library(raster)

# Load DEM from raster *.tif file
DEM <- raster("covs/DEMENV5.tif")
plot(DEM)
```

> Tip for GRASS GIS or GDAL: Use *mdenoise* module/utility to remove noise while preserving sharp features like ridges, lines, and valleys.

SRTM contains many gaps (pixels with no data). These gaps could be filled using splines. SAGA GIS has a module called *Close Gaps with Splines* and other similar tools for doing this.

5.2 Parent material

Parent material has a crucial impact on soil formation, soil geochemistry, and soil physics. Parent material, if not specifically mapped by soil mappers and included in soil maps, is usually available from geological maps. These maps focus on rock formation, mineral components, and age, and often lack younger surface sediments (even in quaternary maps). Parent material/rock types classified

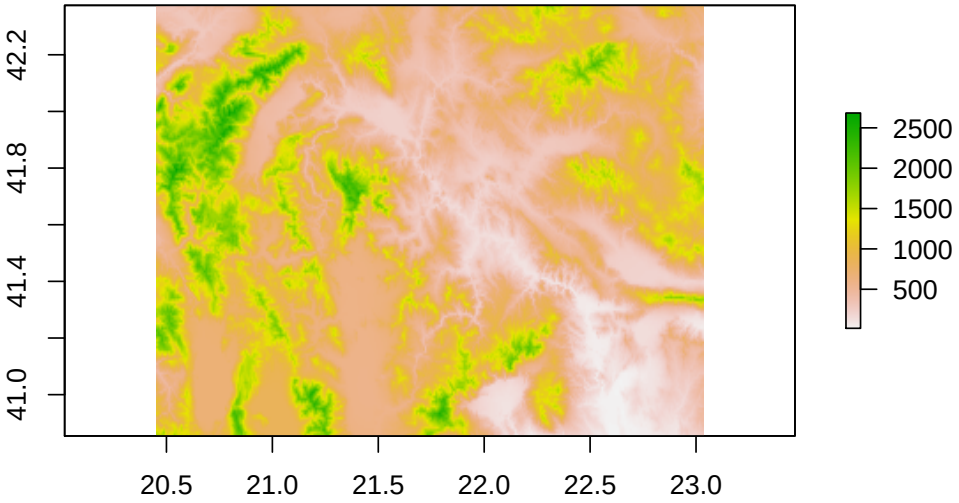


Figure 5.1: SRTM 90 m resampled to 1 km for FYROM

by soil mappers considers more strongly geochemistry and rock structure. Its geochemistry has an essential impact on the soil chemistry, e.g. cation exchange capacity, base saturation, and nutrient stock. The rock structure determines the ability to disintegrate, which has an impact on soil physical properties, like texture, skeleton content, permeability, and soil thickness.

National parent material and geological maps may be used. Other available datasets and data portals are given on the ISRIC WorldGrids website that can be accessed at this link: http://worldgrids.org/doku.php.

- **OneGeology**: The world geological maps are now being integrated via the OneGeology project which aims at producing a consistent geological map of the world in approximate scale 1:1M (Jackson, 2007); link: http://www.onegeology.org/.
- **USGS**: United States Geological Survey (USGS) as several data portals, e.g. that allow browsing of the International Surface Geology (split into South Asia, South America, Iran, Gulf of Mexico, Former Soviet Union, Europe, Caribbean, Bangladesh, Asia Pacific, Arctic, Arabian Peninsula, Africa and Afghanistan); link: https://mrdata.usgs.gov/geology/world/.
- **GLiM**: Hartmann and Moosdorf (2012) have assembled a global, purely lithological database called GLiM (Global Lithological Map). GLiM consists of over 1.25 million digital polygons that are classified into three levels (a total of 42 rock-type classes); link: https://www.geo.uni-hamburg.de/en/geologie/forschung/geochemie/glim.html).
- **USGS** and **ESRI**: Both jointly released in 2014 a Global Ecological Land Units map at 250 m resolution. This also includes world layer of rock types.

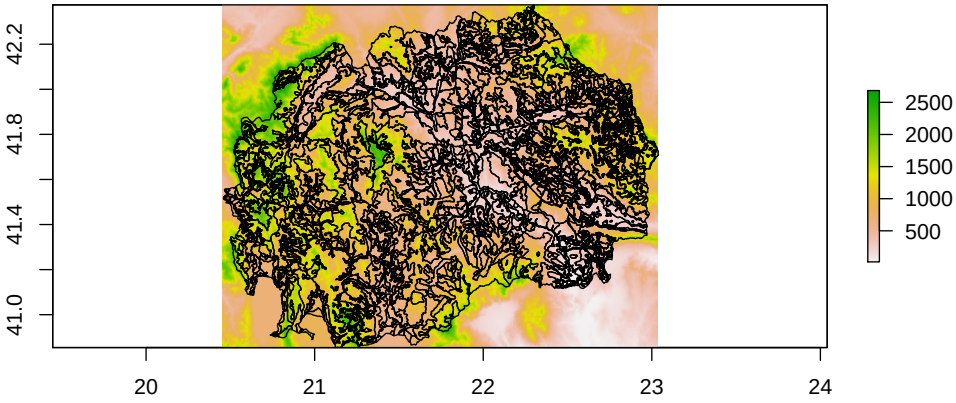


Figure 5.2: Soil map of FYROM

This data can be downloaded from the USGS site; link: (http://rmgsc.cr.usgs.gov/outgoing/ecosystems/Global/).

5.3 Soil maps

Soil maps play a crucial role for upscaling soil property data from point locations. They can be the spatial layer for conventional upscaling, they can also serve as a covariate in DSM. Predicted soil property maps have lower quality in areas where the covariates such as relief, geology, and climate do not correlate well with the dependent variable, here SOC stocks. This is especially true for soils under groundwater or stagnic water influence. This information is well-represented in soil maps.

5.3.1 Global HWSD soil property maps

FAO, ISRIC, the International Institute for Applied Systems Analysis (IIASA), the Institute of Soil Science, Chinese Academy of Sciences (ISS CAS), and the Joint Research Center of the European Commision (JRC) together produced a gridded 1 km soil class map (HWSD). Global HWSD-derived soil property maps can be downloaded as GeoTIFF files at the following link http://worldgrids.org/doku.php/wiki:layers#harmonized_world_soil_database_images_5_km (see Section 5.6).

```
# Load the soil map from a shapefile *.shp file
soilmap <- shapefile("MK_soilmap_simple.shp")

# Plot the DEM together with the soil types
plot(DEM)
lines(soilmap)
```

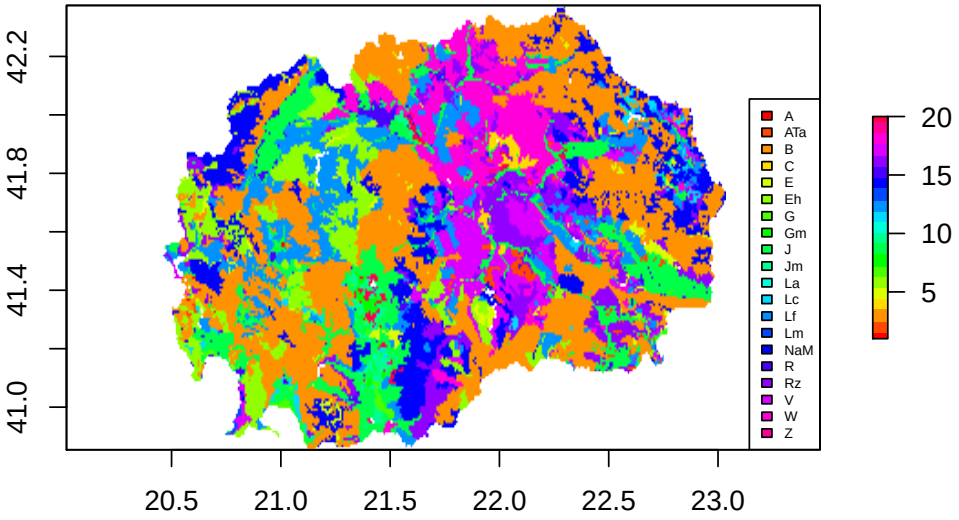


Figure 5.3: Rasterized soil map of FYROM from DEM raster layer

Digitized small-scale national soil maps are the most important spatial layer for soil property mapping. The higher their resolution, the better soil maps contribute to high-quality soil property maps - considering that the map should cover the target area/full country coverage.

5.3.2 Technical steps - Rasterizing a vector layer in R

```
# The Symbol attribute from the vector layer will be used for the
# rasterization process. It has to be a factor
soilmap@data$Symbol <- as.factor(soilmap@data$Symbol)

# Save the levels names in a character vector
Symbol.levels <- levels(soilmap$Symbol)

# The rasterization process needs a layer with the target grid
# system: spatial extent and cell size
# The DEM raster layer could be used for this
soilmap.r <- rasterize(x = soilmap, y = DEM, field = "Symbol")

plot(soilmap.r, col=rainbow(21))
legend("bottomright",legend = Symbol.levels, fill=rainbow(21),
       cex=0.5)
```

5.4 Land cover and land use

Besides soil, geology, and climate, land use and/or land cover data are unarguably vital data for any statistical effort to map soil properties. There are many of various sources of data on the land cover including global and continental products, such as GlobCover, GeoCover, GlobeLand30, CORINE Land Cover.

For further reading and other global data sources not listed below, see the information following this link: http://worldgrids.org/doku.php/wiki:land_cover_and_land_use.

```
# Load the landcover from the raster *.tif file
landcover <- raster("covs/LCEE10.tif")

# Land cover is a categorical covariate, this has to be made
# explicit using function as.factor()
landcover <- as.factor(landcover)
```

5.4.1 GlobCover (Global)

GlobCover is a European Space Agency (ESA) initiative which began in 2005 in partnership with JRC, FAO, the European Environmental Agency (EEA), the United Nations Environment Programme (UNEP), the Global Observation for Forest Cover and Land Dynamics (GOFC-GOLD), and the International Geosphere-Biosphere Programme (IGBP).

The aim of the project was to develop a service capable of delivering global composites and land cover maps using as input observations from the 300 m MERIS sensor onboard the ENVISAT satellite mission. ESA makes available the land cover maps, which cover 2 periods: December 2004 - June 2006 and January - December 2009.

The classification module of the GlobCover processing chain consists in transforming the MERIS-FR multispectral mosaics produced by the pre-processing modules into a meaningful global land cover map. The global land cover map has been produced in an automatic and global way and is associated with a legend defined and documented using the UN Land Cover Classification System (LCCS). The GlobCover 2009 land cover map is delivered as one global land cover map covering the entire Earth. Its legend, which counts 22 land cover classes, has been designed to be consistent at the global scale and therefore, it is determined by the level of information that is available and that makes sense at this scale (Bontemps et al., 2011).

The GlobCover data are available here http://due.esrin.esa.int/page_globcover.php.

5.4.2 Landsat GeoCover (Global)

The Landsat GeoCover collection of global imagery was merged into mosaics by the Earth Satellite Company (now MDA Federal). The result was a series of tiled imagery that is easier to wield than individual scenes, especially since they cover larger areas than the originals. The great detail in these mosaic scenes, however, makes them large in storage size, so the Mr.Sid file format, which includes compression operations, was chosen for output. While GeoCover itself is available in three epochs of 1975, 1990 and 2000, only the latter two epochs were made into mosaics.

The GeoCover Landsat mosaics are delivered in a Universal Transverse Mercator (UTM) / World Geodetic System 1984 (WGS84) projection. The mosaics extend north-south over 5 degrees of latitude and span east-west for the full width of the UTM zone. For mosaics below 60 degrees north latitude, the width of the mosaic is the standard UTM zone width of 6 degrees of longitude. For mosaics above 60 degrees of latitude, the UTM zone is widened to 12 degrees, centered on the standard even-numbered UTM meridians. To insure overlap between adjacent UTM zones, each mosaic extends for at least 50 kilometers to the east and west, and 1 kilometer to the north and south. The pixel size is 14.25 meters (V 2000).

The Landsat GeoCover data are available here ftp://ftp.glcf.umd.edu/glcf/Mosaic_Landsat/ (FTP Access).

5.4.3 GlobeLand30 (Global)

GlobeLand30, the Earth's first global land cover dataset at 30 m resolution for the years 2000 and 2010, was recently released and made publicly available by China. The National Geomatics Center of China under the *Global Land Cover Mapping at Finer Resolution* project has recently generated a global land cover map named GlobeLand30. The dataset covers two timestamps of 2000 and 2010, primarily acquired from Landsat TM and ETM+ sensors, which were then coupled/checked with some local products.

The GlobaLand30 data are publicly available for non-commercial purposes here http://www.globallandcover.com/GLC30Download/index.aspx.

5.4.4 CORINE land cover (Europe only)

The pan-European component is coordinated by EEA and produces satellite image mosaics, land cover/land use information in the CORINE Land Cover data, and the high-resolution layers.

The CORINE Land Cover is provided for the years 1990, 2000, 2006 and 2012. This vector-based dataset includes 44 land cover and land use classes. The time-series also includes a land-change layer, highlighting changes in land cover and land-use. The high-resolution layers are raster-based datasets (100 m, 250 m) which provide

information about different land cover characteristics and is complementary to land-cover mapping (e.g. CORINE) datasets.

The CORINE Land Cover data are available here http://www.eea.europa.eu/data-and-maps/data.

5.5 Climate

5.5.1 WorldClim V1.4 and V2 (Global)

WorldClim is a set of global climate layers (gridded climate data) with a spatial resolution of about 1 km^2 (10 minutes, 5 minutes, 2.5 minutes are also available). These data can be used for mapping and spatial modeling. The current version is Version 1.4. and a preview of Version 2 is available for testing at http://worldclim.org/. The data can be downloaded as generic grids or in ESRI grid format.

The WorldClim data layers were generated by interpolation of average monthly climate data from weather stations on a 30 arc-second resolution grid. In V1.4, variables included are monthly total precipitation, and monthly mean, minimum and maximum temperatures, and 19 derived bioclimatic variables. The WorldClim precipitation data were obtained from a network of 1,473 stations, mean temperature from 24,542 stations, and minimum and maximum temperatures from 14,835 stations (Hijmans et al., 2005).

The Bioclimatic parameters are: annual mean temperature, mean diurnal range, iso-thermality, temperature seasonality, max temperature of warmest month, minimum temperature of coldest month, temperature annual range , mean temperature of wettest quarter, mean temperature of driest quarter, mean temperature of warmest quarter, mean temperature of coldest quarter, annual precipitation, precipitation of wettest month, precipitation of driest month, precipitation seasonality (coefficient of variation), precipitation of wettest quarter, precipitation of driest quarter, precipitation of warmest quarter, precipitation of coldest quarter.

WorldClim Climate Data are available at: www.worldclim.org (WorldClim 1.4 (current conditions) by www.worldclim.org; Hijmans et al. (2005). Is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License).

```
# Load the climate covariates from the raster *.tif files
files <- list.files(path = "covs/", pattern = "CHE3.tif",
                    full.names = TRUE)

# Stack all the files in one RasterStack
climate <- stack(files)
```

```
# Plot the first two layers
plot(climate[[1:2]])
```

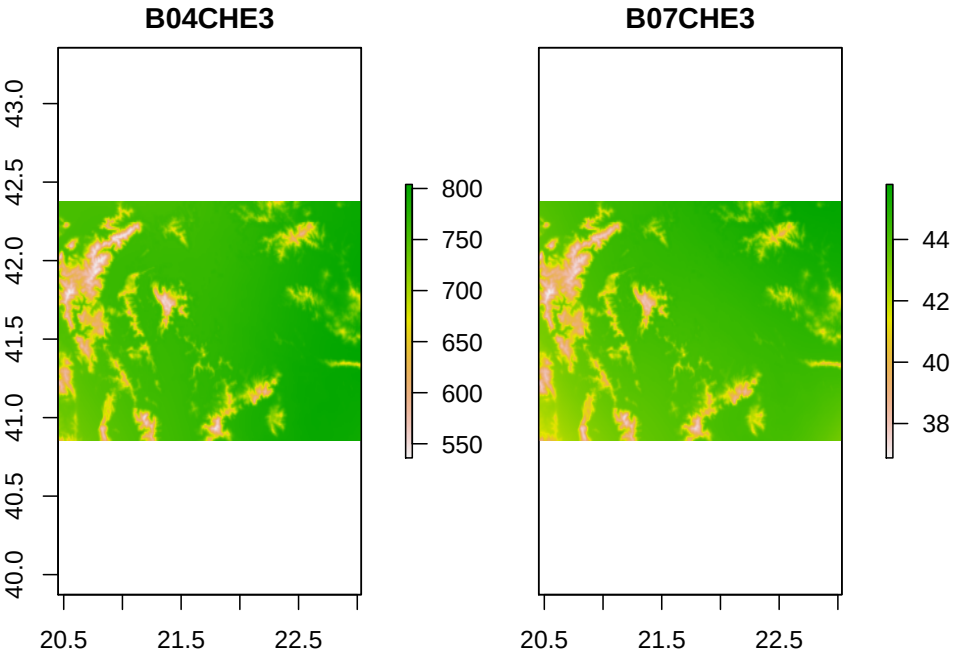


Figure 5.4: Two different temperature layers included in the climate covariates

5.5.2 Gridded agro-meteorological data in Europe (Europe)

CGMS database contains meteorological parameters from weather stations interpolated on a 25×25 km grid. Meteorological data are available on a daily basis from 1975 to the last calendar year completed, covering the EU member states, and neighboring European countries.

The following parameters are available at 1-day time resolution:

- Maximum air temperature (°C)
- Minimum air temperature (°C)
- Mean air temperature (°C)
- Mean daily wind speed at 10m (m/s)
- Mean daily vapor pressure (hPa)
- Sum of precipitation (mm/day)
- Potential evaporation from a free water surface (mm/day)
- Potential evapotranspiration from a crop canopy (mm/day)
- Potential evaporation from a moist bare soil surface (mm/day)
- Total global radiation (KJ/m2/day)
- Snow depth

Data is accessible at the following link: http://agri4cast.jrc.ec.europa.eu/DataPortal/Index.aspx.

5.6 GSOCMap - Data repository (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. These layers can be used as covariates in any DSM exercise.

5.6.1 Covariates and empty mask

The territories and their boundaries are obtained from the Global Administrative Unit Layers (GAUL) dataset. Each folder contains three subfolders:

- **Covs**: GIS layers of various biophysical earth surface properties.
- **Mask**: One *empty* grid file of the territory with territory boundary according to GAUL This grid to be used for the final delivery.
- **Soilgrids**: All `SoilGrids250m` soil class and property layers as available through www.soilgrids.org. Layers are aggregated to 1 km.

5.6.2 Data specifications

The data is provided as follows:

- **File format**: GeoTIFF
- **Coordinate system**: WGS84, latitude-longitude in decimal degrees
- **Spatial resolution**: 1 km

5.6.3 Data access

The data can be accessed at the following links:

ftp.isric.org/ (username: gsp, password: gspisric) or ftp://85.214.253.67/ (username: gsp, password: gspisric)

LICENCE and ACKNOWLEDGEMENT

The GIS layers can be freely used under the condition that proper credit should be given to the original data source in each publication or product derived from these layers. Licences, data sources, data citations are indicated the data description table.

5.7 Extending the soil property table for spatial statistics

The upscaling procedures (see Chapter 6) depend on the rationale that the accumulation of local soil carbon stocks (and also other properties) depend on parameters

for which spatial data are available, such as climate, soil type, parent material, slope, management. This information (Covariates) must be collected first. Details are provided above. The properties contained in the covariates can be extracted for each georeferenced sample site and added to the soil property table (Tab. 4.1). This table is used for training and validation of the statistical model for predicting the SOC stocks which subsequently can be applied to the full spatial extent.

5.8 Preparation of a soil property table for spatial statistics

The upscaling procedures (see Chapter 6) depend on the rationale, that the accumulation of local soil carbon concentrations and stocks (and also other properties) depends on influential parameters for which spatial data are available, such as climate, soil type, parent material, slope, management. Any parameter in the table of local soil properties, for which a spatial layer is available, may be included in the final table. Other covariates will be added in Section 5.9.

> In case this table is prepared for different depths, 0 cm - 10 cm, 10 cm - 30 cm, and if the host institution intends to develop different spatial models for different depths (e.g. separate spatial prediction model for litter and mineral soil 0 cm - 30cm), then the separate grids have to be added.

5.9 Technical steps - Overlay covariates and soil points data

Step 1 - Load soil sample data and covariates

```
# Load the processed data
# This table was prepared in the previous Chapter
dat <- read.csv("data/dat_train.csv")

# Read covariates from raster *.tif files
files <- list.files(path = "covs", pattern = "tif$",
                    full.names = TRUE)

covs <- stack(files)
```

Step 2 - Adding raster soilmap to the raster stack

```
# soilmap.r is the rasterization of the soil map and was obtained
# in a previous step
covs <- stack(covs, soilmap.r)
```

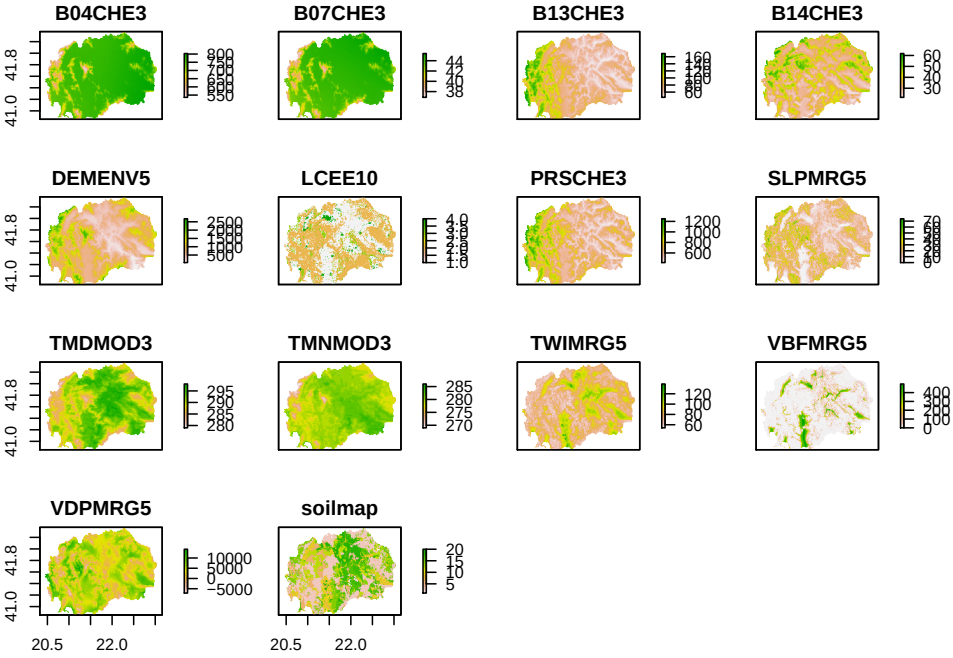


Figure 5.5: FYROM soil covariates

```
# Correct the name for layer 14
names(covs)[14] <- "soilmap"
```

Finally, we will mask the covariates with a mask developed using the country limits. Next, we will export all the covariates as `*.RData` file. This will allow us to load this file in the following Chapters of this cookbook.

```
# Mask the covariates with the country mask from the data repository
mask <- raster("data/mask.tif")
covs <- mask(x = covs, mask = mask)

# Export all the covariates
save(covs, file = "covariates.RData")

plot(covs)
```

Step 3 - Overlay covariates and point data

In order to carry out DSM in terms of examining the statistical significance of environmental predictors for explaining the spatial variation of SOC, we should link both sets of data together and extract the values of the covariates at the locations of the soil point data.

Note that the stacking of rasters can only be possible if they are in the same resolution and extent. If they are not, **raster** package resample and `projectRaster()` functions are for harmonizing all your different raster layers. With the stacked rasters of environmental covariates, we can now perform the intersection and extraction.

```
# Upgrade points data frame to SpatialPointsDataFrame
coordinates(dat) <- ~ X + Y

# Extract values from covariates to the soil points
dat <- extract(x = covs, y = dat, sp = TRUE)

# LCEE10 and soilmap are categorical variables
dat@data$LCEE10 <- as.factor(dat@data$LCEE10)
dat@data$soilmap <- as.factor(dat@data$soilmap)

levels(soilmap) <- Symbol.levels

summary(dat@data)
```

```
##        id             SOC              BLD
##  P0003  :   1   Min.   : 0.080   Min.   :0.05353
##  P0007  :   1   1st Qu.: 1.772   1st Qu.:1.27881
##  P0008  :   1   Median : 2.620   Median :1.39605
##  P0009  :   1   Mean   : 3.579   Mean   :1.37990
##  P0010  :   1   3rd Qu.: 4.090   3rd Qu.:1.47619
##  P0011  :   1   Max.   :86.510   Max.   :2.89888
##  (Other):2910
##      CRFVOL             OCSKGM           meaERROR
##  Min.   : 0.00000   Min.   :0.03524   Min.   :0.172
##  1st Qu.: 0.01195   1st Qu.:0.70410   1st Qu.:3.160
##  Median : 5.00000   Median :0.99674   Median :3.830
##  Mean   :10.91665   Mean   :1.15062   Mean   :3.717
##  3rd Qu.:17.19974   3rd Qu.:1.40756   3rd Qu.:4.260
##  Max.   :93.95013   Max.   :7.43023   Max.   :8.620
##
##    OCSKGMlog            B04CHE3          B07CHE3
##  Min.   :-3.345652   Min.   :547.2   Min.   :37.60
##  1st Qu.:-0.350842   1st Qu.:759.8   1st Qu.:43.74
##  Median :-0.003263   Median :763.9   Median :44.06
##  Mean   :-0.008980   Mean   :759.3   Mean   :43.88
##  3rd Qu.: 0.341860   3rd Qu.:774.8   3rd Qu.:44.54
##  Max.   : 2.005557   Max.   :802.9   Max.   :45.39
##
##     B13CHE3          B14CHE3          DEMENV5        LCEE10
##  Min.   : 45.71   Min.   :22.26   Min.   :  45.0   1   :1967
##  1st Qu.: 60.01   1st Qu.:27.70   1st Qu.: 395.5   2   : 698
##  Median : 71.19   Median :30.11   Median : 594.0   3   :  94
```

```
##  Mean   : 74.26   Mean   :32.31   Mean   : 655.8   4   : 156
##  3rd Qu.: 78.79   3rd Qu.:35.61   3rd Qu.: 814.2   NA's:   1
##  Max.   :167.31   Max.   :60.32   Max.   :2375.0
##
##     PRSCHE3          SLPMRG5          TMDMOD3
##  Min.   : 430.4   Min.   : 0.000   Min.   :280.0
##  1st Qu.: 529.9   1st Qu.: 0.000   1st Qu.:291.0
##  Median : 565.4   Median : 4.000   Median :293.0
##  Mean   : 604.6   Mean   : 8.553   Mean   :292.3
##  3rd Qu.: 651.3   3rd Qu.:13.000   3rd Qu.:294.0
##  Max.   :1211.9   Max.   :61.000   Max.   :297.0
##
##     TMNMOD3          TWIMRG5          VBFMRG5
##  Min.   :270.0   Min.   : 56.00   Min.   :  0.0
##  1st Qu.:279.0   1st Qu.: 79.00   1st Qu.:  0.0
##  Median :280.0   Median : 94.00   Median : 70.5
##  Mean   :279.7   Mean   : 91.12   Mean   :173.6
##  3rd Qu.:281.0   3rd Qu.:102.00   3rd Qu.:390.0
##  Max.   :284.0   Max.   :139.00   Max.   :498.0
##
##     VDPMRG5         soilmap
##  Min.   :-5869   9      :857
##  1st Qu.: 3571   3      :425
##  Median : 5702   12     :217
##  Mean   : 5370   10     :211
##  3rd Qu.: 7024   14     :205
##  Max.   :14104   (Other):982
##                  NA's   : 19
```

After the extraction, it is useful to check if there are not available/missing values, so-called `NA` values, both in the target variable and covariates. In these cases, these data should be excluded. A quick way to assess if there are missing or `NA` values in the data is to use the `complete.cases()` function.

After removing the `NA` values, now there do not appear to be any missing data as indicated by the `integer(0)` output above. It means we have zero rows with missing information.

The last step involves exporting a table which is our regression matrix including the soil data and the values of the environmental covariates in the position of the point samples.

The summary of the `dat data.frame` shows one point with `NA` values for most of the covariates, and 22 points with `NA` values in the soilmap layer. The regression matrix should not contain `NA` values. There are two options to proceed:

- **Option 1**: In some cases, these `NA` values are from points with bad position data. Therefore, the points area outside the study area. In this case, the solution is to correct the coordinates or eliminate the points. This is the case

for the two points with NA values for most of the covariates.

- **Option 2**: Another case is when a covariate is incomplete and does not cover all the area. This could produce many NA values. There are two different solutions, either to eliminate the covariate or to eliminate the point data. This is the case for the soilmap layer and the 30 points.

Step 4 - Convert result to data.frame and save as a *.csv table

```
dat <- as.data.frame(dat)

# The points with NA values have to be removed
dat <- dat[complete.cases(dat),]

# Export as a *.csv table
write.csv(dat, "data/MKD_RegMatrix.csv", row.names = FALSE)
```

Chapter 6

Mapping methods

R. Baritz, M. Guevara, V.L. Mulder, G.F. Olmedo, C. Thine, R.R. Vargas, Y. Yigini

In this Chapter, we want to introduce five different approaches for obtaining the SOC map for FYROM. The first two methods presented in Section 6.1 are classified as conventional upscaling and the presented methods in the Sections 6.2, 6.3, and 6.4 are approaches from DSM.

The first method is class-matching. In this approach, we derive average SOC stocks per class either from the soil type for which a national map exists, or through the combination with other spatial covariates such as land use category, climate type, biome, etc. This approach is used in the absence of spatial coordinates of the source data. The second conventional upscaling method is geo-matching, were upscaling is based on averaged SOC values per mapping unit.

Furthermore, we present three methods from DSM. Regression-kriging (RK) is a hybrid model with both, a deterministic and a stochastic component (Hengl et al., 2007). Next method is called random forest (RF). This one is an ensemble of regression trees based on bagging. This machine learning algorithm uses a different combination of prediction factors to train multiple regression trees (Breiman, 1996). The last method is called support vector machines (SVM). This method applies a simple linear method to the data but in a high-dimensional feature space, non-linearly related to the input space (Karatzoglou et al., 2006). We present this diversity of methods because there is no *best* mapping method for DSM, and testing and selection has to be done for every data scenario (Guevara et al., 2018).

The authors of this Chapter used **R** packages. To run the code provided in this Chapter, the following packages need to be installed in the **R** user library. If the packages are not yet installed, the `install.packages()` function can be used.

```
# Installing packages for Chapter 'Mapping Methods'
install.packages(c("sp", "car", "automap",
                   "randomForest", "caret", "Metrics",
```

```
                    "quantregForest", "snow", "reshape",
                    "e1071"))
```

6.1 Conventional upscaling using soil maps

R. Baritz, V.L. Mulder

6.1.1 Overview

The two conventional upscaling methods in the context of SOC mapping, are described by Lettens et al. (2004). Details about weighted averaging can be found in Hiederer (2013). Different conventional upscaling approaches were applied in many countries e.g. by Krasilnikov et al. (2013) (Mexico), Greve et al. (2007) (Denmark), Kolli et al. (2009) (Estonia), Arrouays et al. (2001) (France), and Bhatti et al. (2002) (Canada). Because the structure of soil map databases differs between countries (e.g. definition of the soil mapping unit, stratification, soil associations, dominating and co-dominating soils, typical and estimate soil properties for different depths), it is difficult to define a generic methodology for the use of these maps for mapping soil property information.

However, the essential principle which is commonly used, is to combine soil property data from local observations with soil maps via class-matching and geo-matching.

6.1.2 Diversity of national soil legacy data sets

In order to develop a representative and large national soil database, very often, data from different sources (e.g. soil surveys or projects in different parts of the country at different times) are combined. The following case of Belgium demonstrates, how available legacy databases could be combined. Three different sources are used to compile an overview of national SOC stocks:

- **Data source 1**: Soil profile database with 13,000 points of genetic horizons; for each site, there is information about the soil series, map coordinates, and land use class; for each horizon, there is information about depth and thickness, textural fractions and class, volume percentage of rock fragments; analytically, there is the organic carbon content and inorganic carbon content.
- **Data source 2**: Forest soil data base which includes ectorganic horizons. According to their national definition, the term *ectorganic* designates the surface horizons with an organic matter content of at least 30%, thus, it includes both the litter layer and the organic soil layers. For the calculation of SOC stocks for the ectorganic layer, no fixed-depth was used, instead, the measured thickness of the organic layers and litter layers was applied.

- **Data source 3**: 15,000 soil surface samples were used (upper 20 cm of mineral soil); carbon measurements are available per depth class.

From all data sources, SOC stocks for peat soils were calculated separately.

6.1.3 Technical steps - Class-matching

Step 1 - Data preparation

1. Separate the database for forests, peat, and other land uses. If only horizons are provided, derive or estimate average depth of horizons per soil type, and add upper and lower depth.
2. Check the completeness of parameters per depth using the solum depth to code empty cells.
3. Correct the organic carbon in case total carbon was determined (total carbon minus inorganic carbon concentration).
4. Correct the Walkley and Black method for incomplete oxidation (using the correction factor of 1.32 for the quantification of the organic carbon content in the soil samples) (Walkley and Black, 1934).
5. If BD measured is lacking, select proper PTFs and estimate BD. Publications about the choice of the best suited PTF for specific physio-geographic conditions are available.
6. If the stone content is missing, investigate using other data sources or literature, to which a correction for stones should be applied.
7. If possible, derive the standard average stone content for different soils/horizons/depths, or used published soil profiles, as a simple correction factor.
8. Calculate SOC stocks for all mineral and peat soils over 0 cm - 30 cm, and optionally for forest organic layers, and peat between >30 cm and <100 cm.

Step 2 - Preparatory GIS operations

1. Prepare the covariates.
2. Identify the properties of covariates for each point observation using geo-matching.
3. For mapping using **geo-matching of all points**: Extract the covariate information to all georeferenced sample sites. The SOC values from all points within the unit are then averaged. It is assumed that the points represent the real variability of soil types within the units.

Step 3 - Mapping using class-matching of points in agreement with classes

Through class-matching, only those points or profiles are attributed to a soil or landscape unit if both the soil and the land use class are the same. Class-matching thus can be performed regardless of the profile location. Before averaging, a weighing factor can be introduced according to the area proportions of dominant, co-dominant and associated soils. Each profile needs to be matched to its soil type/landscape type, and the SOC value averaged.

1. Determine a soil or landscape unit (e.g. national soil legend stratified by climate area and mainland cover type such as forest, grassland, cropland).
2. Calculate average SOC stocks from all soils which match the soil/landscape unit.
3. Present the soil/landscape map with SOC stocks, do not classify SOC stocks (tons/hectare) into groups (e.g. < 50, 50 - 100, > 100).

   Note: Pre-classified SOC maps cannot be integrated into a global GSOCmap legend.

6.1.4 Technical steps - Geo-matching

Because of its importance, geo-matching as a mapping method is described in more detail. It is neccessary to first prepare the working environment with the pre-processed input data. The following Section presents different geo-matching procedures.

1. Setting up software and working environment.
2. Geo-matching SOC with WRB soil map (step-by-step, using the soil Map of FYROM and the demonstration data presented above).
3. Geo-matching SOC with other environmental variables: land use.
4. Finally, the development of landscape Units (Lettens et al., 2004) is outlined.

This example was developed for QGIS and focusses on SOC mapping using vector data. QGIS 2.18 with GRASS 7.05 will be used. For more information, see also the following links:

- https://gis.stackexchange.com
- http://www.qgis.org/
- http://www.qgisforum.org/

Step 1 - Setting up a QGIS project

1. Install QGIS and supporting software; download the software at http://www.qgis.org/en/site/forusers/download.html (select correct version for Windows, MacOS or Linux, 32 or 64 bit).
2. Create a work folder, e.g. D:\GSOC\practical_matching. Copy the folder with the FYROM demonstration data into this folder.
3. Start QGIS Desktop with GRASS. Figure 6.1 shows the start screen of QGIS Desktop. In the upper left panel, there is the **Browser Panel**, which lists the geodata used for this example. In the bottom left, the layer information is given in the **Layers Panel** for the layers displayed on the right.
4. Load the FYROM soil map. Right-click the file in the **Browser panel** and add the map to your project.
5. Display the soil classes. Right-click on the file in the **Layers Panel** and select *Properties*. Go to the *Style* and change from *Single Symbol* to *Categorized* as shown in Figure 6.2. Select the column WRB and press the icon *Classify* and change the colors if you want. Next, apply the change and finish by clicking the **OK-Button**.

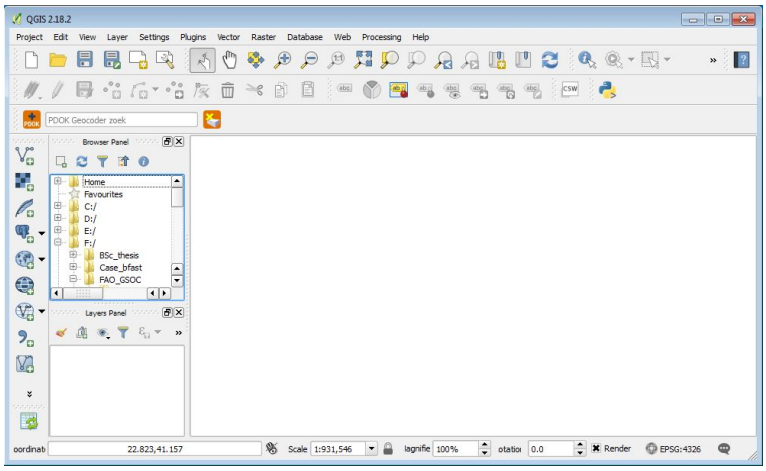


Figure 6.1: QGIS Desktop with the Browser Panel, the Layers Panel and the display of layers

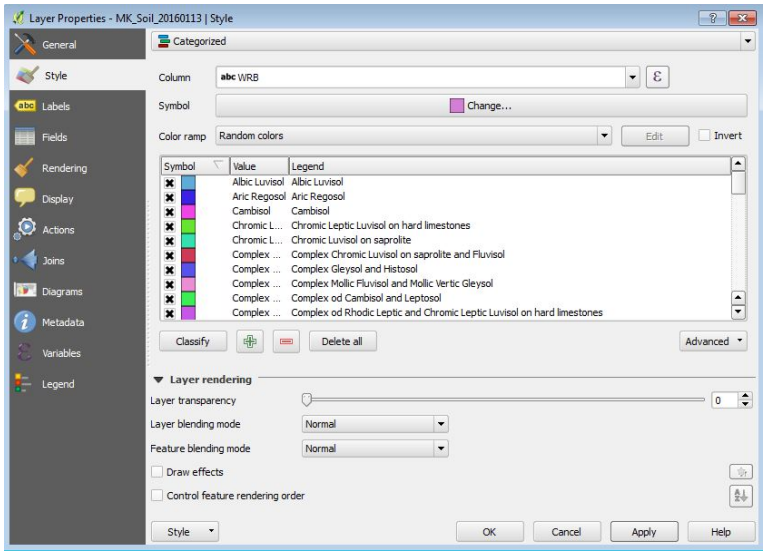


Figure 6.2: Changing layer properties for the FYROM soil map

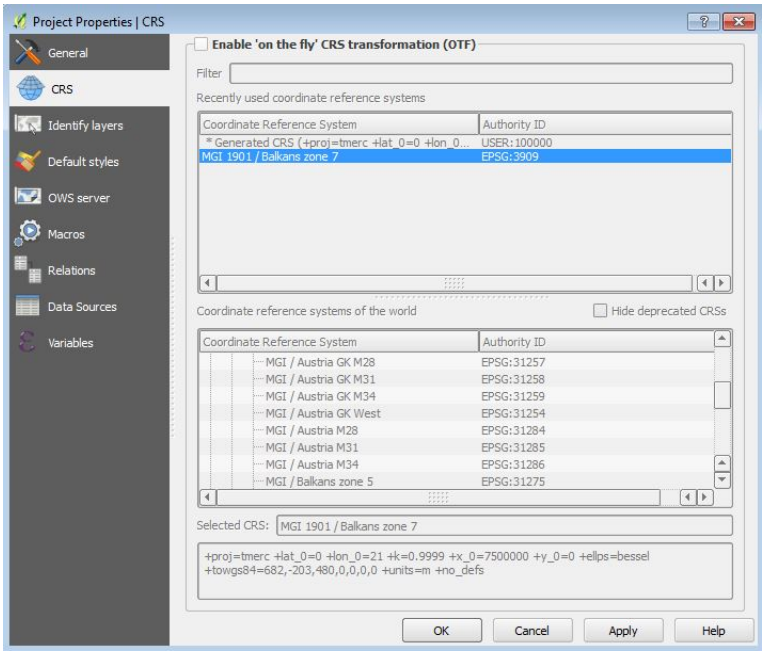


Figure 6.3: Project properties and projection settings

6. Ensure the correct projection for this project. Go to select *Project*, *Project Properties*, *CRS*. In this case, you automatically use the local projection for FYROM. The EPSG code is 3909 which corresponds to MGI 1901/Balkans Zone 7 as shown in Figure 6.3).

7. Save the project in the created folder. Load and display the pre-processed SOC point data. If a shapefile already exists, this is done the same way as described in Step 4. If you have the data as a text file, you need to create a vector layer out of that file. Go to *Layer*, *Add Layer*, *Add Delimited Text Layer*. Select the correct file and proper CRS projection. The layer should be added to your **Layers Panel** and displayed on top of the soil map.

Step 2A - Geo-matching SOC with WRB soil map

In **Step 2** you will make a SOC map, based on the FYROM soil map and the SOC values at the sampled points, following three steps. First you extract the soil map information for the point data, then you obtain the mean and standard deviation of the SOC stocks per soil class, based on the point data, and last you assign these values to the corresponding soil map units. The steps are described in detail below.

1. Extract the soil map information to the soil profile data by *Join Attributes by Location*. Select *Vector*, *Data Management Tools*, and *Join Attributes by Location*. Here, the target vector layers are the soil point data, and the join vector layer is the FYROM soil map. The geometric predicate is *intersects*.

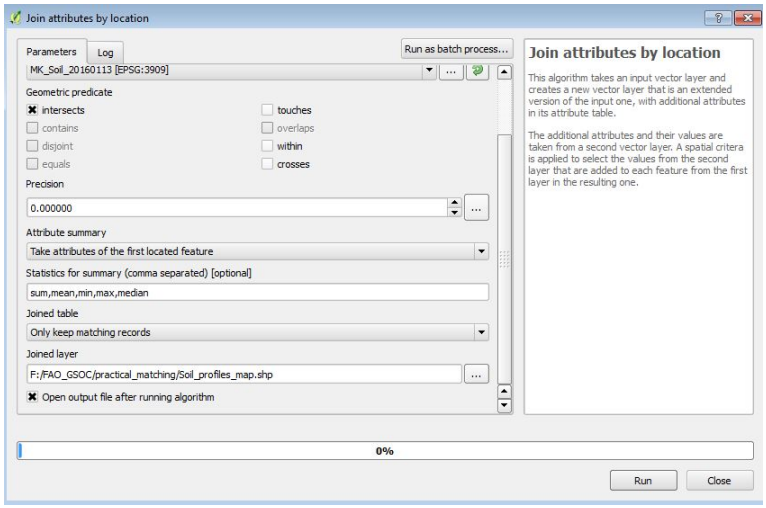


Figure 6.4: Join attributes by location

Figure 6.4 shows how to specify within the joined table to keep only matching records. Save the joined layer as a new file.

2. Check the newly generated file, open the attribute table. The new file is added to the **Layers Panel**. Right-click on the file and open the attribute table. The information from the FYROM soil map is now added to the soil point data.
3. Most likely, the SOC values in the table are not numeric and thus statistics cannot be calculated. Check the data format, right-click on the file in the **Layers Panel** and check the type name of the SOC field under the tab *Fields*. If they are not integer then change the format.
4. Change of the data format: Open the attribute table and start editing (the pencil symbol in the upper left corner of your table). Open the field calculator and follow these instructions as shown in Figure 6.5):

a. Checkbox: Create a new field
b. Output field name: Specify the name of your field
c. Output field type: Decimal Number (real)
d. Output field length: 10, precision: 3
e. Expression: to_real("SOC"), the to_real function can be found under *Conversions* and the SOC field is found under *Fields and Values*

5. After calculating the field, save the edits and leave the editing mode prior to closing the table. If changes are not saved, the added field will be lost.
6. Calculate the median SOC stock per soil type. Go to the tab *Vector* and select *Group Stats*. Select the layer from the spatial join you made in Step 2. Add the field SOC and median to the box with *Values* and the field WRB to the *Rows*. Make sure the box with *Use only selected features* is **not** checked. Now calculate the statistics. A table will be given in the left pane (see Figure

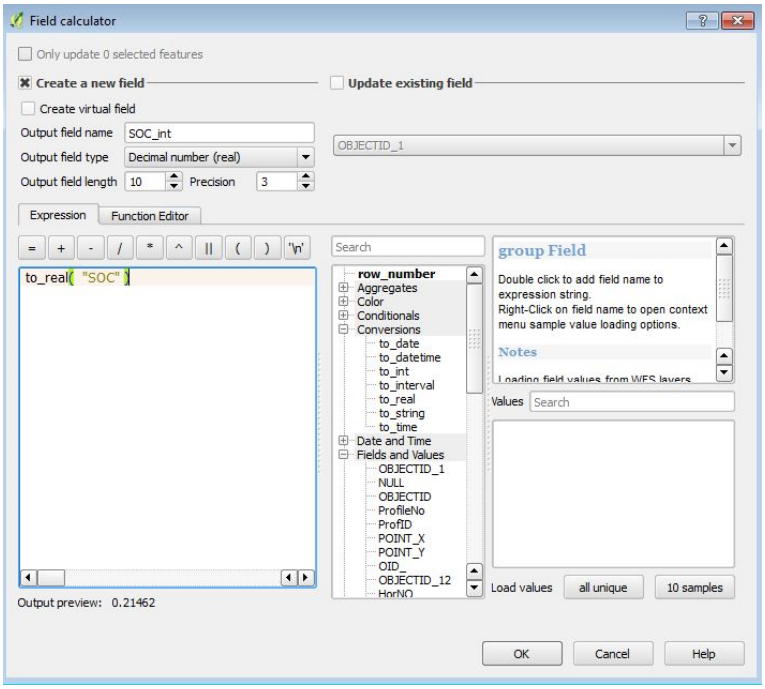


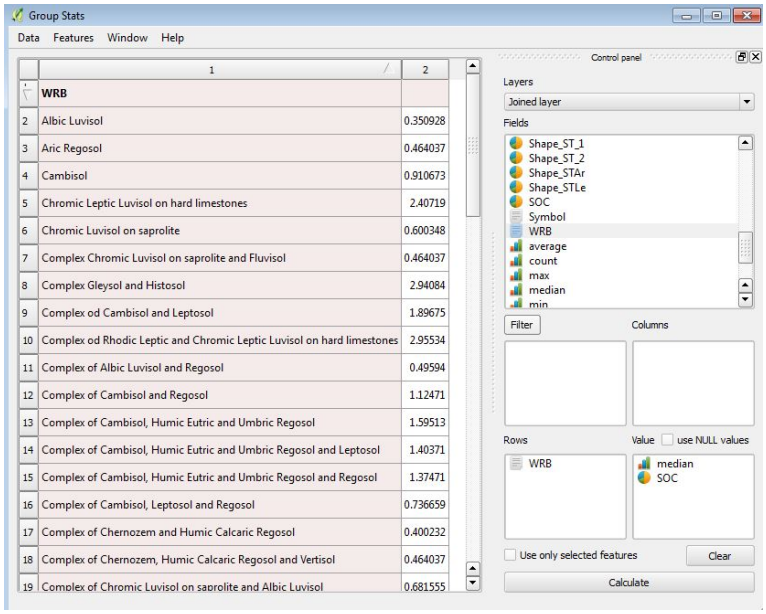
Figure 6.5: Example field calculator

6.6). Save this file as `*.csv` and repeat the same for the standard deviation.

7. Join the mean and standard deviation of SOC to the soil map. First, add the files generated during Step 6 to the **Layers Panel**. In the **Layers Panel**, right-click on the FYROM soil map. Go to *Properties*, select *Joins* and add a new join for both the median and standard deviation of SOC. The Join and Target Field are both WRB.
8. Display the SOC maps. Go to the layer properties of the FYROM soil map. Go to *Style* and change the legend to a graduated legend. In the column, you indicate the assigned SOC values. Probably this is not a integer number and so you have to convert this number again to a numeric values. You can do this with the box next to the box as depicted in Figure 6.7. Change the number of classes to e.g. 10 classes, change the mode of the legend and change the color scheme if you want and apply the settings. Now you have a map with the median SOC stocks per WRB soil class.

9. In order to generate a proper layout, go to *Project* and select *New Print Composer* (see Figure 6.8):

a. Add map using *Layout* and *Add Map*. Define a square on the canvas and the selected map will be displayed.
b. Similarly, title, scale bar, legend and a north arrow can be added. Specific properties can be changed in the box *Item properties*.



	1	2
	WRB	
2	Albic Luvisol	0.350928
3	Aric Regosol	0.464037
4	Cambisol	0.910673
5	Chromic Leptic Luvisol on hard limestones	2.40719
6	Chromic Luvisol on saprolite	0.600348
7	Complex Chromic Luvisol on saprolite and Fluvisol	0.464037
8	Complex Gleysol and Histosol	2.94084
9	Complex od Cambisol and Leptosol	1.89675
10	Complex od Rhodic Leptic and Chromic Leptic Luvisol on hard limestones	2.95534
11	Complex of Albic Luvisol and Regosol	0.49594
12	Complex of Cambisol and Regosol	1.12471
13	Complex of Cambisol, Humic Eutric and Umbric Regosol	1.59513
14	Complex of Cambisol, Humic Eutric and Umbric Regosol and Leptosol	1.40371
15	Complex of Cambisol, Humic Eutric and Umbric Regosol and Regosol	1.37471
16	Complex of Cambisol, Leptosol and Regosol	0.736659
17	Complex of Chernozem and Humic Calcaric Regosol	0.400232
18	Complex of Chernozem, Humic Calcaric Regosol and Vertisol	0.464037
19	Complex of Chromic Luvisol on saprolite and Albic Luvisol	0.681555

Figure 6.6: Calculate group statistics

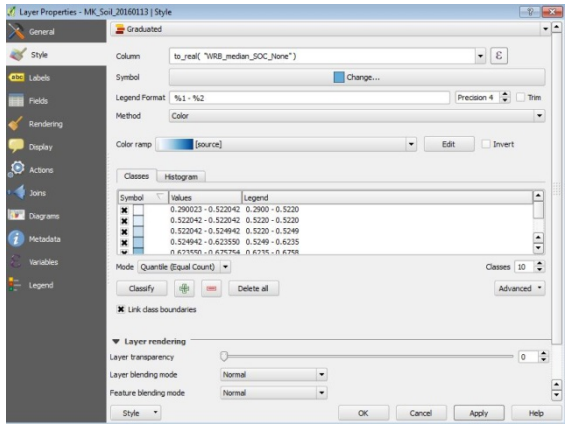


Figure 6.7: Change the legend style to display the SOC values

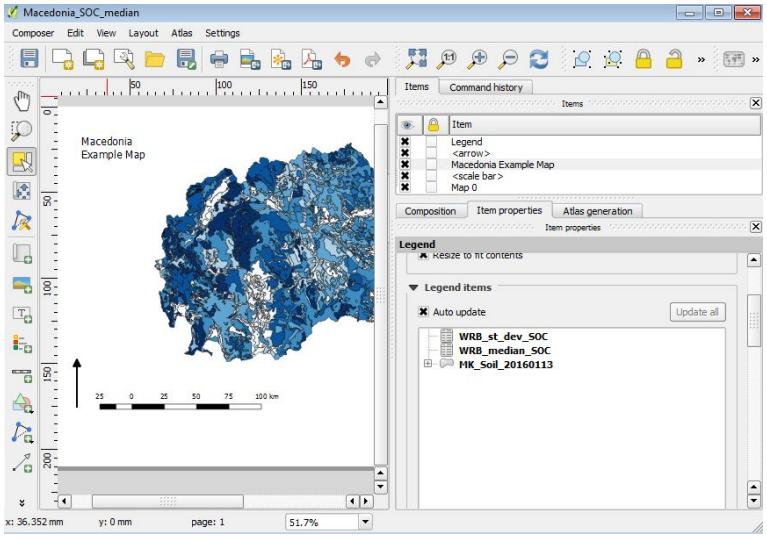


Figure 6.8: Example of the Map composer

c. When the map is finished, it can be exported as an image or `*.pdf` file.

10. Repeat the Steps 2 to 8 but now for the standard deviation of the SOC stocks.
11. Save the file as a new shapefile `*.shp`. Go to **Layers Panel**, *Save As* and select ESRI SHP format and make sure that you define the symbology export *Feature Symbology*. Now, a shapefile is generated, with both the median and standard deviation SOC stock per soil type. Redundant fields can be removed after the new file is created.

Step 2B - Geo-matching SOC with other environmental variables: Land use

1. Start a new project and add the soil point data and FYROM soil map layers from the **Browser Panel**.
2. Add the land use raster file to the **Layers Panel**. This is a raster file with 1 km resolution and projected in lat-long degrees (WGS84). For more information about this product see the online information from worldgrids: http://worldgrids.org/doku.php/wiki:glcesa3.
3. Change the projection to the MGI 1901/Balkans Zone 7. Go to *Raster*, *Projections*, *Warp* and select the proper projection and a suitable file name, e.g. `LU_projected_1km`. Tick the checkbox for the resampling method and choose *Near*. This is the nearest neighbor and most suitable for a transformation of categorical data, such as land use (see Figure 6.9).
4. In order to geo-match the soil point data with land use data, the raster file needs to be converted into a vector file. Go to *Raster*, *Conversions*, and select *Polygonize*. Set a proper output filename, e.g. `LU_polygon_1km`, and check the tickbox for *Fieldname*.

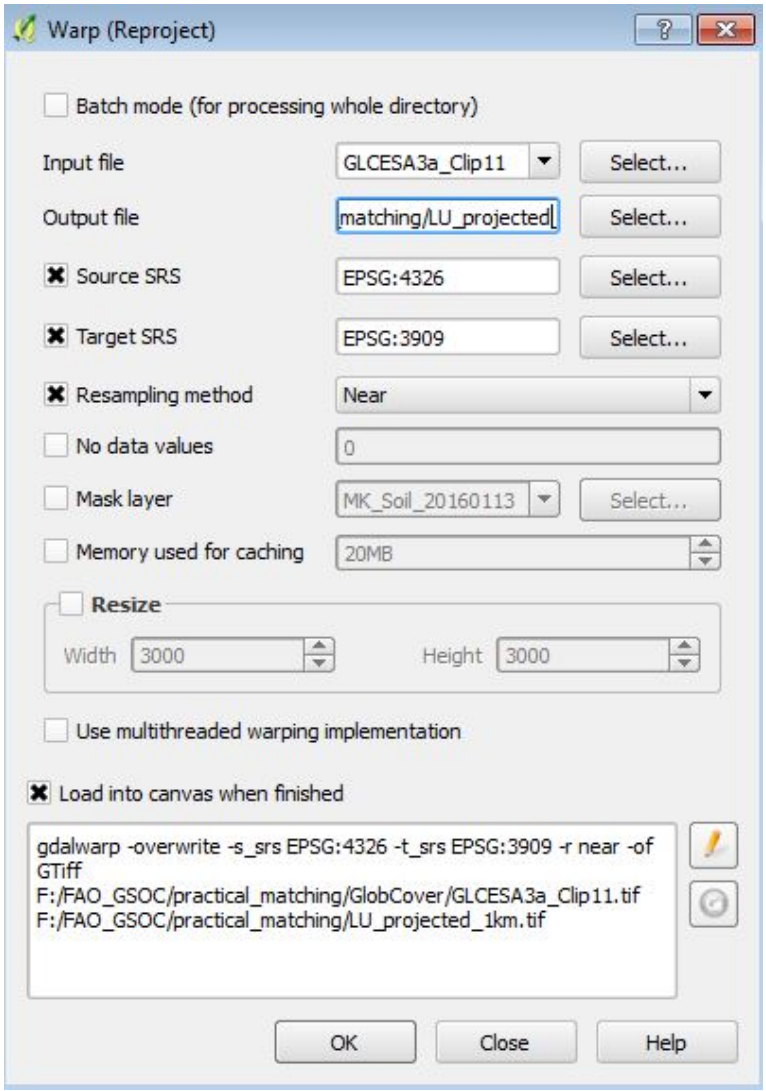


Figure 6.9: Change the projection of a raster file

5. Change the legend style into categories (Step 1-5). Now, the steps from the previous Section need to be repeated, using the land use polygon map instead of using the FYROM soil map.
6. Join attributes by location using the soil point data and the polygon land use map.
7. Calculate the median and standard deviation of SOC by using the Group Statistics for SOC and the land use classes and save the files as CSV file.
8. Add the generated CSV files to the **Layers Panel**.
9. Join the files with the land use polygon map, generated in Steps 3 and 4.
10. Change the classes in the legend and inspect the histogram with the median SOC values. Try to find a proper definition of the class boundaries (Steps 2 to 8).

Step 2C - Joining landscape units and soil mapping units to support class- and geo-matching (optional)

In this Section, it is outlined how SOC stocks can be mapped following the method outlined by Lettens et al. (2004). The general idea is that the landscape is stratified into more or less homogenous units and subsequently, the SOC stocks are obtained following the procedure outlined earlier in this practical. Lettens et al. (2004) outlines a method to stratify the landscape into homogeneous strata with respect to land use and soil type, as was explained earlier. In order to obtain such strata, the soil map and the land use map need to be combined. This can be done using various types of software, e.g. ArcMap, GRASS, QGIS or **R**.

When using the GIS software, the only thing that needs to be done, is intersecting the vector files and dissolving the newly created polygon features. Depending on the software and the quality of your shapefile, you may experience problems with the geometry of your shapefile. Generally, ArcMap and GRASS correct the geometry when the shapefile is loaded, while QGIS does not do this automatically. There are various ways to correct the geometry, however, correcting the geometry falls outside the scope of this training. Therefore, we give some hints on how to correct your geometry prior to using the functions *Intersect* and *Dissolve*.

1. Change the land use raster map to 5 km resolution: Right-click the `Lu_project_1km` file, select *Save as*. Change the resolution to 5000 meters. Scroll down, check the *Pyramids* box, and change the resampling method to *Nearest Neighbour*.
2. Convert the raster map to a polygon map and add the file to the **Layers Panel**.
3. Check the validity of the soil map and land use map. Go to *Vector*, *Geometry Tools*, and select *Check Validity*. Below you find the instructions in case you have no problems with your geometry.
4. Intersect the soil map and the land use map. In ArcGIS and QGIS you can use *Intersection* function following *Vector*, *Geoprocessing Tools* (in GRASS you have to use the function *Overlay* from the *Vector* menu).
5. Dissolve the newly generated polygons vie *Vector*, *Geoprocessing Tools*, select *Dissolve*.
6. Next, this layer can be used to continue with the class-matching or geo-

matching procedures.

How to correct your geometry When encountering problems?

- Run the *v_clean* tool from GRASS within QGIS. Open the *Processing Toolbox*, *GRASS GIS 5 Commands*, *Vector*, and select *v.clean*.
- Install the plugin **Processing LWGEOM Provider**. Go to the plugins menu and search for the plugin and install it. You can find the newly installed tool in the *Processing Toolbox* by typing the name in the *Search* function.
- Manually correct the error nodes of the vector features.

6.1.5 Technical steps - Geo-matching SOC with WRB soil map in R

G.F. Olmedo

Geo-matching can also be done in **R**. We will provide a example code following the steps from Section 6.1.4 - Step 2A, using the WRB soil map.

Step 1 - Load the soil data and soil map

Load the shapefile including the soil map. Then load the points, and promote the points to `spatialPointsDataFrame` using the `coordinates()` function. Make sure both are in the same coordinate system. In case they are not, you can use `spTransform()` to reproject one of the layers.

```
library(raster)
# Load the soil map from a shapefile *.shp file
soilmap <- shapefile("MK_soilmap_simple.shp")

library(sp)
# Load soil data
dat <- read.csv("data/dataproc.csv")

# Promote to SpatialPointsDataFrame
coordinates(dat) <- ~ X + Y

# Set the CRS for the soil samples. As if the same we will copy the
# CRS from the soilmap object
dat@proj4string <- soilmap@proj4string

class(dat)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
# We can plot the soil map units with ths soil samples data over
plot(soilmap, lwd=0.3)
points(dat, col="red", cex=0.2)
```

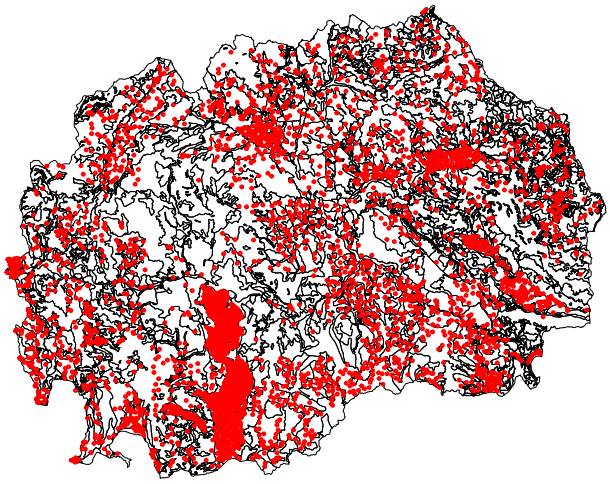


Figure 6.10: Soil map units and soil samples (in red) for the geo-matching exercise

Step 2 - Geo-matching

Now, use the `over()` function to estimate the mean value of the *OCSKGM* field for every map unit.

```
soilmap$OCSKGM <- over(soilmap, dat[,"OCSKGM"], fun=mean)[,1]
```

Step 3 - Rasterize and export the results as geotiff

Finally, convert the layer to raster, to comply with the deliverables specifications.

```
pred <- rasterize(soilmap, DEM, "OCSKGM")

plot(pred)
```

```
writeRaster(pred, "results/MKD_OCSKGM_GM.tif",
            overwrite=TRUE)
```

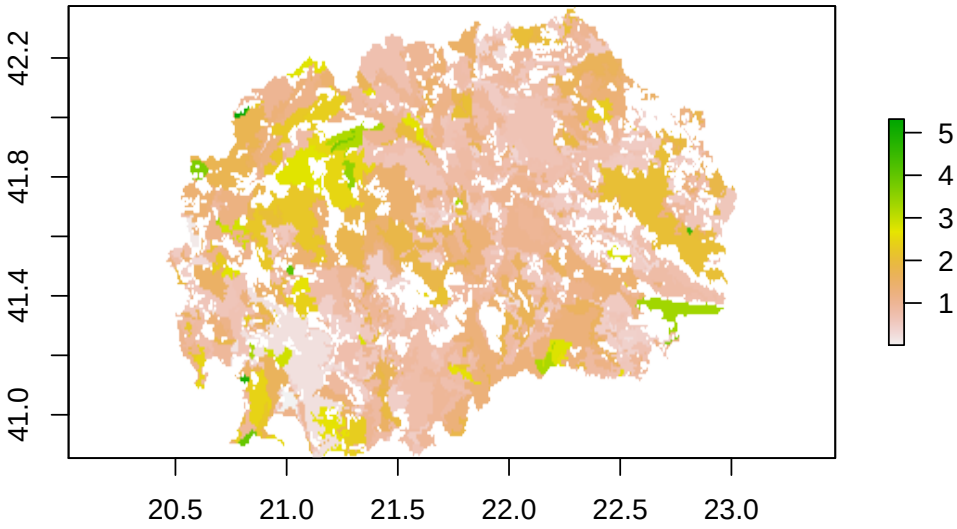


Figure 6.11: SOC prediction map for FYROM using geo-matching

6.2 Regression-Kriging

G.F. Olmedo & Y. Yigini

6.2.1 Overview

Regression-kriging is a spatial interpolation technique that combines a regression of the dependent variable (target variable) on predictors (i.e. the environmental covariates) with kriging of the prediction residuals. In other words, RK is a hybrid method that combines either a simple or a multiple-linear regression model with ordinary kriging of the prediction residuals.

A multiple regression analysis models the relationship of multiple predictor variables and one dependent variable, i.e. it models the deterministic trend between the target variable and environmental covariates. The modeled relationship between predictors and target are summarized in the regression equation, which can then be applied to a different data set in which the target values are unknown but the predictor variables are known. The regression equation predicts the value of the dependent variable using a linear function of the independent variables.

In this Section, we review the RK method for DSM. First, the deterministic part of the trend is modeled using a regression model. Next, the prediction residuals are kriged. In the regression phase of a RK technique, there is a continuous random variable called the dependent variable (target) Y, which is in our case SOC, and a number of independent variables $x_1, x_2, \ldots, x_p$ which are selected covariates. Our purpose is to predict the value of the dependent variable using a linear function of the independent variables. The values of the independent variables (environmental covariates) are known quantities to be used for prediction.

6.2.2 Assumptions

Standard linear regression models with standard estimation techniques make a number of assumptions about the predictor variables, the response variables, and their relationship. One must review the assumptions made when using the model.

- **Linearity**: The mean value of Y for each specific combination of the X's is a linear function of the X's. In practice this assumption can virtually never be confirmed; fortunately, multiple regression procedures are not greatly affected by minor deviations from this assumption. If curvature in the relationships is evident, one may consider either transforming the variables or explicitly allowing for nonlinear components.

- **Normality Assumption**: It is assumed in multiple regression that the residuals (predicted minus observed values) are distributed normally (i.e., follow the normal distribution). Again, even though most tests (specifically the F-test) are quite robust with regard to violations of this assumption,

it is always a good idea, before drawing final conclusions, to review the distributions of the major variables of interest. You can produce histograms of the residuals as well as normal probability plots, in order to inspect the distribution of the residual values.

- **Collinearity**: There is not perfect collinearity in any combination of the X's. A higher degree of collinearity, or overlap, among independent variables, can cause problems in multiple linear regression models. Collinearity (also multicollinearity) is a phenomenon in which two or more predictors in a multiple regression models are highly correlated. Collinearity causes increase in variances and relatedly increases inaccuracy.

- **Distribution of the Errors**: The error term is normally distributed with a mean of zero and constant variance.\
- **Homoscedasticity**: The variance of the error term is constant for all combinations of X's. The term homoscedasticity means *same scatter*. Its antonym is heteroscedasticity which means *different scatter*.

6.2.3 Pre-processing of covariates

Before using the selected predictors, multicollinearity assumption must be reviewed. As an assumption, there is not perfect collinearity in any combination of the X's. A higher degree of collinearity, or overlap, among independent variables, can cause problems in multiple linear regression models. The multicollinearity of a number of variables can be assessed using Variance Inflation Factor (VIF).

In **R**, the function `vif()` from **caret** package can estimate the VIF . There are several rules of thumb to establish when there is a serious multi-collinearity (e.g. when the VIF square root is over 2). The principal component analysis (PCA) can be used to overcome multicollinearity issues.

Principal components analysis can cope with data containing large numbers of covariates that are highly collinear which is the common case in environmental predictors. Often the principal components with higher variances are selected as regressors. However, for the purpose of predicting the outcome, the principal components with low variances may also be important, in some cases even more important.

6.2.4 The terminology

- **Dependent variable** Y: which is to be predicted from a given set of predictors (e.g. SOC content).
- **Independent variables X's (Predictors)**: which influence or explain the dependent variable (Covariates: environmental covariates, DEM derived

Figure 6.12: Linear regression model

covariates, soil maps, land cover maps, climate maps). The data sources for the environmental predictors are provided in Chapter 5.

- **Coefficients** β: Values, computed by the multiple regression tool, reflect the relationship and strength of each independent variable to the dependent variable.
- **Residuals** ε: The portion of the dependent variable that cannot be explained by the model; the model under/over predictions.

Before we proceed with the regression analysis, it is advisable to inspect the histogram of the dependent/target variable, in order to see if it needs to be transformed before fitting the regression model. The data for the selected soil property is normal when the frequency distribution of the values follow a bell-shaped curve (Gaussian Distribution) which is symmetric around its mean. Normality tests may be used to assess normality. If a normality test indicates that data are not normally distributed, it may be necessary to transform the data to meet the normality assumption.

Both, the normality tests and data transformations can be easily performed using any commercial or open source statistical tool (**R**, SPSS, MINITAB…).

The main steps for regression kriging (RK) are shown in the Figure 6.13.

1. The first step is to prepare a map showing the spatial distribution of the sample locations and the corresponding soil property information, e.g. soil organic matter and environmental properties. The first can be achieved as outlined in Section [Overlay covariates and spatial data]. The overlaying operation can be performed in **R**, ArcGIS, SAGA GIS or QGIS.
2. The essential part of multiple regression analysis is to build a regression model by using the environmental predictors. After extracting the values of explanatory maps and target variables into the single table, we can now start fitting multiple regression model using the table that contains data from dependent variable and predictors.
3. In particular cases, stepwise multiple linear regression (MLR) can be used to eliminate insignificant predictors. The stepwise MLR usually selects predictors that have the strongest linear correlations with the target variable, which reflect the highest predictive capacity.
4. Kriging of the residuals (prediction errors): In RK, the regression model detrends the data, produces the residuals which we need to krige and to be added to the regression model predictions.

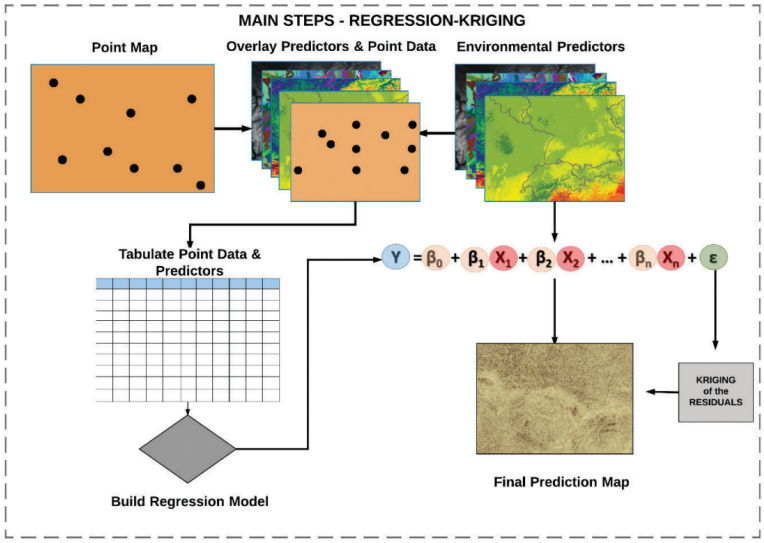


Figure 6.13: Workflow for regression-kriging

6.2.5 Interpret the Key Results of Multiple Regression

Regression analysis generates an equation to describe the statistical relationship between one or more predictor variables and the response variable. The R-squared, p-values and coefficients that appear in the output for linear regression analysis must also be reviewed. Before accepting the result of a linear regression it is important to evaluate its suitability at explaining the data. One of the many ways to do this is to visually examine the residuals. If the model is appropriate, then the residual errors should be random and normally distributed.

R-squared

R-sqared (R^2) is the percentage of variation in the response that is explained by the model. The higher the R^2 value is, the better the model fits your data. R^2 ranges between 0% and 100%. R^2 usually increases when additional predictors are added in the model.

p-value

To determine whether the association between the dependent and each predictor in the model is statistically significant, compare the p-value for the term to your significance level to assess the null hypothesis. Usually, a significance level of 0.05 works well.

- **p-value $\leq$ significance level**: The relationship is statistically significant. If the p-value is less than or equal to the significance level, we can conclude that there is a statistically significant relationship between the dependent variable and the predictor.

- p**-value > significance level**: The relationship is not statistically significant. If the p-value is greater than the significance level, you cannot conclude that there is a statistically significant relationship between the dependent variable and the predictor. You may want to refit the model without the predictor.

Residuals

We can plot the residuals which can help us determine whether the model is adequate and meets the assumptions of the analysis. If the model is appropriate, then the residual errors should be random and normally distributed. We can plot residuals versus fits to verify the assumption that the residuals are randomly distributed and have constant variance. Ideally, the points should fall randomly around zero, with no recognizable patterns in the points.

The diagnostic plots for the model should be evaluated to confirm if all the assumptions of linear regression are met. After the abovementioned assumptions are validated, we can proceed with making the prediction map using the model with significant predictors.

6.2.6 Using the results of a regression analysis to make predictions

The purpose of a regression analysis, of course, is to develop a model that can be used to make the prediction of a dependent variable. The derived regression equation is to be used to create the prediction map for the dependent variable.

> Raster calculation can be easily performed using **raster** package in **R** or ArcGIS using the **Raster Calculator** tool (is called **Map Algebra** in the prior versions).

6.2.7 Technical steps - Regression-kriging

Requirements

The following computational steps from the previous Chapters are required to implement RK in **R**:

1. Setting-up the software environment.
2. Obtaining and installing **RStudio**.
3. Installing and loading the required **R** packages.
4. Preparation of local soil property data.
5. Preparation of spatial covariates (DEM-derived covariates, land cover/land use, climate, parent material).

Step 1 - Setting working space and initial steps

One of the first steps should be setting our working directory. If you read/write files from/to disk, this takes place in the working directory. If we do not set the

working directory, we could easily write files to an undesirable file location. The following example shows how to set the working directory in **R** to our folder which contains data for the study area (point data, covariates).

Note that we must use the forward slash / or double backslash \\ in **R**! Single backslash \ will not work. Now we can check if the working directory has been correctly set by using the `getwd()` function:

```
getwd()
```

Step 2 - Data preparation

Point dataset

We previously prepared the point dataset in 4 chapter. We will be using this data file (.csv) in this section.

Environmental predictors (covariates)

In the Chapter 5, we presented and prepared several global and continental datasets. In addition to these datasets, numerous covariate layers have been prepared by ISRIC for the GSOCmap project. These are GIS raster layers of various biophysical earth surface properties for each country in the world. Some of these layers will be used as predictors in this Section. Please download the covariates for your own study area from GSOCmap Data Repository as explained in Section 5.6.

In Section 5.9, a table with the points values after data preparation and the values of our spatial predictors was prepared. This step involves loading this table.

Now we will import our point dataset using `read.csv()` function. The easiest way to create a data frame is to read in data from a file. This is done using the function `read.csv()`, which works with comma delimited files. Data can be read in from other file formats as well, using different functions, but `read.csv()` is the most commonly used approach. **R** is very flexible in how it reads in data from text files (`read.table()`, `read.csv()`, `read.csv2()`, `read.delim()`, `read.delim2()`). Please type `?read.table()` for help.

```
# Load data
dat <- read.csv("data/MKD_RegMatrix.csv")

dat$LCEE10 <- as.factor(dat$LCEE10)
dat$soilmap <- as.factor(dat$soilmap)

# Explore the data structure
str(dat)
```

```
## 'data.frame':    2897 obs. of  23 variables:
##  $ id       : Factor w/ 2897 levels "P0003","P0007",..: 1 2 3 4..
##  $ Y        : num  42 42 42.1 42 42 ...
##  $ X        : num  20.8 20.8 20.8 20.9 20.9 ...
##  $ SOC      : num  26.38 6.15 3.94 3.26 2.29 ...
```

```
##  $ BLD       : num  0.73 1.17 1.3 1.34 1.41 ...
##  $ CRFVOL    : num  8 18.6 31.9 21.7 14.5 ...
##  $ OCSKGM    : num  5.32 1.75 1.04 1.03 0.83 ...
##  $ meaERROR  : num  2.16 2.85 2.65 3.16 3.63 2.83 2.94 2.49 2.77..
##  $ OCSKGMlog: num  1.6712 0.5591 0.0429 0.0286 -0.1862 ...
##  $ B04CHE3   : num  574 553 693 743 744 ...
##  $ B07CHE3   : num  38.5 37.8 42.1 43.7 43.7 ...
##  $ B13CHE3   : num  111.6 125 99.8 118.1 121 ...
##  $ B14CHE3   : num  59.2 60.3 42.4 39.9 38.7 ...
##  $ DEMENV5   : int  2327 2207 1243 1120 1098 1492 1413 1809 1731..
##  $ LCEE10    : Factor w/ 4 levels "1","2","3","4": 1 3 2 1 2 2 2..
##  $ PRSCHE3   : num  998 1053 780 839 844 ...
##  $ SLPMRG5   : int  13 36 6 25 30 24 15 17 20 43 ...
##  $ TMDMOD3   : int  282 280 285 288 289 287 286 286 287 286 ...
##  $ TMNMOD3   : int  272 270 277 279 279 277 277 273 274 273 ...
##  $ TWIMRG5   : int  61 62 81 66 65 72 68 67 65 59 ...
##  $ VBFMRG5   : int  0 0 14 0 0 0 0 0 0 0 ...
##  $ VDPMRG5   : int  311 823 10048 1963 -173 -400 -9 -692 -1139 2..
##  $ soilmap   : Factor w/ 20 levels "1","2","3","4",..: 6 14 14 3..
```

Since we will be working with spatial data we need to define the coordinates for the imported data. Using the `coordinates()` function from the **sp** package we can define the columns in the data frame to refer to spatial coordinates. Here the coordinates are listed in columns `X` and `Y`.

```
library(sp)

# Promote to SpatialPointsDataFrame
coordinates(dat) <- ~ X + Y

class(dat)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

`SpatialPointsDataFrame` structure is essentially the same as a data frame, except that additional *spatial* elements have been added or partitioned into slots. Some important ones being the bounding box (sort of like the spatial extent of the data), and the coordinate reference system `proj4string()`, which we need to define for the sample dataset. To define the CRS, we must know where our data are from, and what was the corresponding CRS used when recording the spatial information in the field. For this data set, the CRS used was WGS84 (EPSG:4326).

To clearly tell **R** this information we define the CRS which describes a reference system in a way understood by the PROJ.4 projection library. An interface to the PROJ.4 library is available in the **rgdal** package. As an alternative to using PROJ.4 character strings, we can use the corresponding yet simpler EPSG code. **rgdal** also recognizes these codes. If you are unsure of the PROJ.4 or EPSG

code for the spatial data that you have but know the CRS, you should consult http://spatialreference.org/ for assistance.

> **CRS**: Please note that, when working with spatial data, it is very important that the CRS of the point data and covariates are the same.

```
dat@proj4string <- CRS(projargs = "+init=epsg:4326")

dat@proj4string
```

```
## CRS arguments:
##  +init=epsg:4326 +proj=longlat +datum=WGS84 +no_defs
## +ellps=WGS84 +towgs84=0,0,0
```

Now we will import the covariates. When the covariate layers are in common resolution and extent, rather than working with individual rasters it is better to stack them all into a single **R** object. In this example, we use 13 covariates from the GSOCmap Data Repository and a rasterized version of the soil type map. The rasterization of vectorial data was covered in Technical Steps - Rasterizing a vector layer in R. The file containing all the covariates was prepared at the end of Chapter 5.

```
load(file = "covariates.RData")

names(covs)
```

```
##  [1] "B04CHE3" "B07CHE3" "B13CHE3" "B14CHE3" "DEMENV5" "LCEE10"
##  [7] "PRSCHE3" "SLPMRG5" "TMDMOD3" "TMNMOD3" "TWIMRG5" "VBFMRG5"
## [13] "VDPMRG5" "soilmap"
```

Step 3 - Fitting the MLR model

It would be better to progress with a data frame of just the data and covariates required for the modeling. In this case, we will subset the columns SOC, the covariates and the spatial coordinates (`X` and `Y`).

```
datdf <- dat@data

datdf <- datdf[, c("OCSKGM", names(covs))]
```

Let's fit a linear model with all available covariates.

```
# Fit a multiple linear regression model between the log-transformed
# values of OCS and the top 20 covariates
model.MLR <- lm(log(OCSKGM) ~ ., data = datdf)
```

From the summary of our fitted model (`model.MLR`) above, it seems only a few of the covariates are significant in describing the spatial variation of the target variable. To determine the most predictive model we can run a stepwise regression using the `step()` function. With this function, we can also specify the mode of stepwise search, can be one of *both*, *backward*, or *foreward*.

```
# Stepwise variable selection
model.MLR.step <- step(model.MLR, direction="both")
```

Comparing the summary of both the full and stepwise linear models, there is very little difference between the models such as the R^2. Both models explain about 23% of variation of the target variable. Obviously, the full model is more complex as it has more parameters than the step model.

```
# Summary and ANOVA of the new model using stepwise covariates
# selection
summary(model.MLR.step)
anova(model.MLR.step)
```

```
##
## Call:
## lm(formula = log(OCSKGM) ~ B04CHE3 + B07CHE3 + B13CHE3 + DEMENV5 +
##     LCEE10 + PRSCHE3 + SLPMRG5 + TMDMOD3 + TMNMOD3 + VBFMRG5 +
##     VDPMRG5 + soilmap, data = datdf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3625 -0.2637  0.0368  0.3111  1.8859
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.166e+00  4.343e+00   2.111  0.03489 *
## B04CHE3     -5.877e-03  1.099e-03  -5.347 9.64e-08 ***
## B07CHE3      1.110e-01  3.460e-02   3.209  0.00135 **
## B13CHE3     -4.361e-03  1.640e-03  -2.659  0.00788 **
## DEMENV5     -1.882e-04  8.926e-05  -2.108  0.03508 *
## LCEE102      9.745e-02  3.369e-02   2.893  0.00385 **
## LCEE103      1.399e-01  5.490e-02   2.548  0.01088 *
## LCEE104     -3.612e-02  4.360e-02  -0.829  0.40741
## PRSCHE3      9.174e-04  3.139e-04   2.923  0.00350 **
## SLPMRG5     -2.440e-03  1.508e-03  -1.619  0.10559
## TMDMOD3     -5.584e-02  7.612e-03  -7.336 2.86e-13 ***
## TMNMOD3      2.467e-02  1.291e-02   1.911  0.05611 .
## VBFMRG5      4.941e-04  9.867e-05   5.008 5.84e-07 ***
## VDPMRG5     -2.696e-05  4.360e-06  -6.185 7.11e-10 ***
## soilmap2    -3.848e-01  4.885e-01  -0.788  0.43090
## soilmap3    -2.094e-01  4.825e-01  -0.434  0.66429
## soilmap4    -1.955e-01  4.886e-01  -0.400  0.68919
## soilmap5    -6.323e-02  5.202e-01  -0.122  0.90327
## soilmap6     2.087e-01  4.841e-01   0.431  0.66649
## soilmap7     1.459e-01  4.857e-01   0.300  0.76398
## soilmap8    -1.875e-01  4.848e-01  -0.387  0.69904
## soilmap9    -3.278e-01  4.817e-01  -0.681  0.49617
```

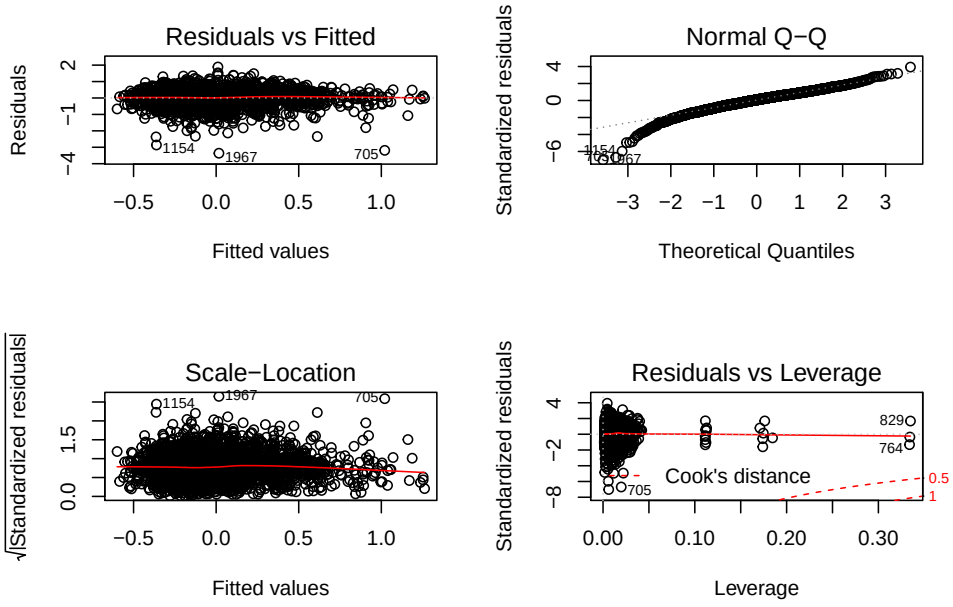
```
## soilmap10   -1.001e-01  4.833e-01  -0.207  0.83590
## soilmap11   -3.587e-01  4.874e-01  -0.736  0.46188
## soilmap12   -2.135e-01  4.822e-01  -0.443  0.65797
## soilmap13    3.091e-01  5.563e-01   0.556  0.57855
## soilmap14   -2.224e-01  4.828e-01  -0.461  0.64506
## soilmap15   -1.905e-01  4.876e-01  -0.391  0.69600
## soilmap16   -3.482e-01  4.831e-01  -0.721  0.47112
## soilmap17   -1.478e-01  4.838e-01  -0.306  0.75996
## soilmap18   -8.714e-02  4.830e-01  -0.180  0.85684
## soilmap19   -5.491e-01  5.082e-01  -1.080  0.28002
## soilmap20   -3.123e-01  4.846e-01  -0.644  0.51934
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4806 on 2864 degrees of freedom
## Multiple R-squared:  0.2472, Adjusted R-squared:  0.2388
## F-statistic: 29.38 on 32 and 2864 DF,  p-value: < 2.2e-16
##
## Analysis of Variance Table
##
## Response: log(OCSKGM)
##             Df Sum Sq Mean Sq  F value    Pr(>F)
## B04CHE3      1 111.35 111.347 482.0082 < 2.2e-16 ***
## B07CHE3      1   2.33   2.335  10.1060  0.001494 **
## B13CHE3      1   1.64   1.642   7.1059  0.007726 **
## DEMENV5      1   5.07   5.067  21.9339 2.953e-06 ***
## LCEE10       3  17.22   5.740  24.8497 7.201e-16 ***
## PRSCHE3      1   4.91   4.910  21.2530 4.201e-06 ***
## SLPMRG5      1   1.97   1.971   8.5305  0.003520 **
## TMDMOD3      1   3.75   3.749  16.2300 5.756e-05 ***
## TMNMOD3      1   0.60   0.602   2.6081  0.106428
## VBFMRG5      1  12.75  12.750  55.1943 1.433e-13 ***
## VDPMRG5      1  10.80  10.797  46.7375 9.880e-12 ***
## soilmap     19  44.83   2.359  10.2135 < 2.2e-16 ***
## Residuals 2864 661.60   0.231
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In those two models above, we used all available points. It is important to test the performance of a model based upon an external validation.

```
# Graphical diagnosis of the regression analysis
par(mfrow=c(2,2))
plot(model.MLR.step)
```

```
## Warning: not plotting observations with leverage one:
##   340
```

```
## Warning: not plotting observations with leverage one:
##   340
```




```
par(mfrow=c(1,1))
```

```
# Collinearity test using variance inflation factors
library(car)
vif(model.MLR.step)
```

```
##              GVIF Df GVIF^(1/(2*Df))
## B04CHE3 17.352988  1        4.165692
## B07CHE3 17.580993  1        4.192969
## B13CHE3 14.463943  1        3.803149
## DEMENV5 14.325207  1        3.784866
## LCEE10   3.349841  3        1.223219
## PRSCHE3 20.323563  1        4.508166
## SLPMRG5  3.201225  1        1.789197
## TMDMOD3  6.724708  1        2.593204
## TMNMOD3  6.172634  1        2.484479
## VBFMRG5  4.182605  1        2.045142
## VDPMRG5  1.933334  1        1.390444
## soilmap 16.784795 19        1.077047
```

```
# Problematic covariates should have sqrt (VIF) > 2
sqrt(vif(model.MLR.step))
```

```
##              GVIF       Df GVIF^(1/(2*Df))
## B04CHE3 4.165692 1.000000        2.041003
## B07CHE3 4.192969 1.000000        2.047674
```

```
## B13CHE3 3.803149 1.000000        1.950166
## DEMENV5 3.784866 1.000000        1.945473
## LCEE10  1.830257 1.732051        1.105992
## PRSCHE3 4.508166 1.000000        2.123244
## SLPMRG5 1.789197 1.000000        1.337609
## TMDMOD3 2.593204 1.000000        1.610343
## TMNMOD3 2.484479 1.000000        1.576223
## VBFMRG5 2.045142 1.000000        1.430085
## VDPMRG5 1.390444 1.000000        1.179171
## soilmap 4.096925 4.358899        1.037809
```

Colinear: Temperature seasonality at 1 km (B04CHE3) and Temperature Annual Range [°C] at 1 km (B07CHE3).

```
# Removing B07CHE3 from the stepwise model
model.MLR.step <- update(model.MLR.step, . ~ . - B07CHE3)

# Test the vif again
sqrt(vif(model.MLR.step))
```

```
##             GVIF       Df GVIF^(1/(2*Df))
## B04CHE3 2.268418 1.000000        1.506127
## B13CHE3 3.624615 1.000000        1.903842
## DEMENV5 3.645817 1.000000        1.909402
## LCEE10  1.818077 1.732051        1.104762
## PRSCHE3 4.460815 1.000000        2.112064
## SLPMRG5 1.783090 1.000000        1.335324
## TMDMOD3 2.572322 1.000000        1.603846
## TMNMOD3 2.299838 1.000000        1.516522
## VBFMRG5 2.015123 1.000000        1.419550
## VDPMRG5 1.387165 1.000000        1.177780
## soilmap 3.840284 4.358899        1.036043
```

```
# Summary of the new model using stepwise covariates selection
summary(model.MLR.step)
```

```
##
## Call:
## lm(formula = log(OCSKGM) ~ B04CHE3 + B13CHE3 + DEMENV5 + LCEE10 +
##     PRSCHE3 + SLPMRG5 + TMDMOD3 + TMNMOD3 + VBFMRG5 + VDPMRG5 +
##     soilmap, data = datdf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3857 -0.2662  0.0390  0.3096  1.9418
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.536e+01  3.896e+00   3.943 8.23e-05 ***
```

```
## B04CHE3     -2.919e-03  5.994e-04  -4.869 1.18e-06 ***
## B13CHE3     -5.955e-03  1.566e-03  -3.804 0.000146 ***
## DEMENV5     -2.651e-04  8.612e-05  -3.079 0.002099 **
## LCEE102      1.006e-01  3.373e-02   2.983 0.002879 **
## LCEE103      1.250e-01  5.479e-02   2.281 0.022598 *
## LCEE104     -2.756e-02  4.358e-02  -0.632 0.527224
## PRSCHE3      1.063e-03  3.111e-04   3.417 0.000642 ***
## SLPMRG5     -2.041e-03  1.505e-03  -1.356 0.175085
## TMDMOD3     -5.275e-02  7.563e-03  -6.974 3.80e-12 ***
## TMNMOD3      8.998e-03  1.197e-02   0.752 0.452305
## VBFMRG5      4.401e-04  9.738e-05   4.519 6.47e-06 ***
## VDPMRG5     -2.792e-05  4.357e-06  -6.410 1.70e-10 ***
## soilmap2    -4.166e-01  4.892e-01  -0.852 0.394494
## soilmap3    -1.925e-01  4.832e-01  -0.398 0.690372
## soilmap4    -1.959e-01  4.894e-01  -0.400 0.688957
## soilmap5    -6.440e-02  5.211e-01  -0.124 0.901641
## soilmap6     2.098e-01  4.849e-01   0.433 0.665224
## soilmap7     1.337e-01  4.865e-01   0.275 0.783490
## soilmap8    -1.838e-01  4.856e-01  -0.379 0.705074
## soilmap9    -3.376e-01  4.825e-01  -0.700 0.484097
## soilmap10   -9.951e-02  4.841e-01  -0.206 0.837162
## soilmap11   -3.570e-01  4.882e-01  -0.731 0.464661
## soilmap12   -2.104e-01  4.830e-01  -0.436 0.663172
## soilmap13    2.675e-01  5.570e-01   0.480 0.631148
## soilmap14   -2.312e-01  4.836e-01  -0.478 0.632615
## soilmap15   -1.733e-01  4.883e-01  -0.355 0.722668
## soilmap16   -3.620e-01  4.839e-01  -0.748 0.454464
## soilmap17   -1.564e-01  4.846e-01  -0.323 0.746854
## soilmap18   -6.759e-02  4.837e-01  -0.140 0.888886
## soilmap19   -5.479e-01  5.090e-01  -1.076 0.281873
## soilmap20   -3.061e-01  4.853e-01  -0.631 0.528351
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4814 on 2865 degrees of freedom
## Multiple R-squared:  0.2445, Adjusted R-squared:  0.2363
## F-statistic:  29.9 on 31 and 2865 DF,  p-value: < 2.2e-16
```

```
# Outlier test using the Bonferroni test
outlierTest(model.MLR.step)
```

```
##       rstudent unadjusted p-value Bonferonni p
## 1967 -7.114604         1.4122e-12   4.0897e-09
## 705  -6.795188         1.3108e-11   3.7959e-08
## 1154 -6.045856         1.6789e-09   4.8621e-06
## 1395 -5.112379         3.3907e-07   9.8194e-04
## 709  -5.078342         4.0515e-07   1.1733e-03
## 1229 -4.842237         1.3520e-06   3.9155e-03
```

```
## 1136 -4.468410          8.1860e-06   2.3707e-02
```

Step 4 - Prediction and residual kriging

Now we can make the predictions and plot the map. We can use either our DSM data table for covariate values or `covs` object for making our prediction. Using stack avoids the step of arranging all covariates into a table format. If multiple rasters are being used, it is necessary to have them arranged as a rasterStack object. This is useful as it also ensures all the rasters are of the same extent and resolution. Here we can use the raster predict function such as below using the covStack raster stack as we created in the Step 3.

```
# Project point data
dat <- spTransform(dat, CRS("+init=epsg:6204"))

# Project covariates to VN-2000 UTM 48N
covs <- projectRaster(covs, crs = CRS("+init=epsg:6204"),
                      method='ngb')

covs$LCEE10 <- as.factor(covs$LCEE10)
covs$soilmap <- as.factor(covs$soilmap)
```

```
# Promote covariates to spatial grid dataframe.
# Takes some time and a lot of memory!
covs.sp <- as(covs, "SpatialGridDataFrame")
covs.sp$LCEE10 <- as.factor(covs.sp$LCEE10)
covs.sp$soilmap <- as.factor(covs.sp$soilmap)
```

```
## Checking if any bins have less than 5 points, merging bins when necessary...
##
## Selected:
##   model      psill    range
## 1   Nug 0.16735323    0.000
## 2   Sph 0.06746394 6646.127
##
## Tested models, best first:
##    Tested.models kappa      SSerror
## 1            Sph     0 3.964233e-07
## 25           Ste    10 4.182590e-07
## 24           Ste     5 4.266696e-07
## 23           Ste     2 4.501717e-07
## 22           Ste   1.9 4.519816e-07
## 21           Ste   1.8 4.539405e-07
## 20           Ste   1.7 4.560649e-07
## 19           Ste   1.6 4.583753e-07
## 18           Ste   1.5 4.608930e-07
## 17           Ste   1.4 4.636455e-07
## 16           Ste   1.3 4.666616e-07
## 3            Gau     0 4.680499e-07
```

```
## 15           Ste   1.2 4.699776e-07
## 14           Ste   1.1 4.736337e-07
## 13           Ste     1 4.776785e-07
## 12           Ste   0.9 4.821684e-07
## 11           Ste   0.8 4.871689e-07
## 10           Ste   0.7 4.927610e-07
## 9            Ste   0.6 4.990387e-07
## 2            Exp     0 5.061150e-07
## 8            Ste   0.5 5.061153e-07
## 7            Ste   0.4 5.141265e-07
## 6            Ste   0.3 5.232363e-07
## 5            Ste   0.2 5.336428e-07
## 4            Ste  0.05 1.148336e-06
## [using universal kriging]
```

```
# RK model
library(automap)

# Run regression-kriging prediction.
# This step can take hours!
OCS.krige <- autoKrige(formula =
                         as.formula(model.MLR.step$call$formula),
                       input_data = dat,
                       new_data = covs.sp,
                       verbose = TRUE,
                       block = c(1000, 1000))

OCS.krige
```

```
# Convert prediction and standard deviation to rasters
# and back-tansform the vlaues
RKprediction <- exp(raster(OCS.krige$krige_output[1]))
RKpredsd <- exp(raster(OCS.krige$krige_output[3]))
```

```
plot(RKprediction)
```

```
plot(RKpredsd)
```

```
# Save results as *.tif files
writeRaster(RKprediction, filename = "results/MKD_OCSKGM_RK.tif",
            overwrite = TRUE)

writeRaster(RKpredsd, filename = "results/MKD_OCSKGM_RKpredsd.tif",
            overwrite = TRUE)
```

```
# Save the model
saveRDS(model.MLR.step, file="results/RKmodel.Rds")
```

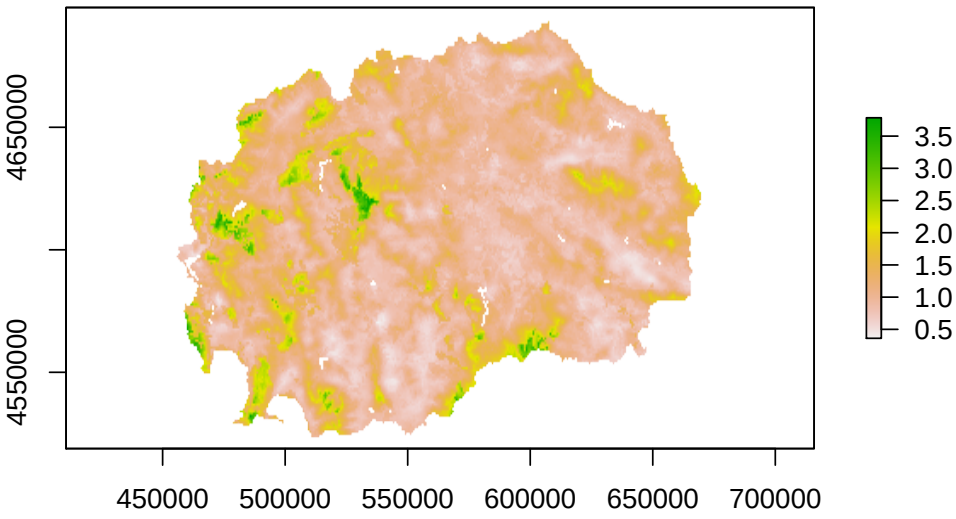


Figure 6.14: SOC prediction of FYROM using a regression kriging model

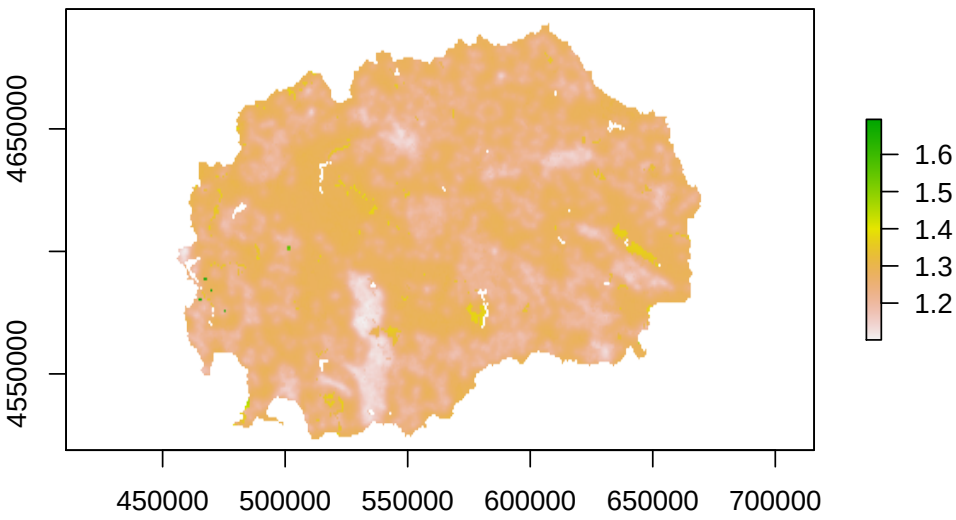


Figure 6.15: Standard deviation map of the regression-kriging model

6.2.8 Technical steps - Cross-validation of regression-kriging models

Cross-validation is introduced in Section 7.4.3. In RK models, n-fold cross-validation and leave-one-out cross-validation can be run using the `krige.cv()` included in **gstat R** package. In this example, we will apply 10 fold cross-validation to our RK model.

```
# autoKrige.cv() does not removes the duplicated points
# We have to do it manually before running the cross-validation
dat = dat[which(!duplicated(dat@coords)), ]

OCS.krige.cv <- autoKrige.cv(formula =
                             as.formula(model.MLR.step$call$formula),
                             input_data = dat, nfold = 10)
```

```
summary(OCS.krige.cv)
```

```
##              [,1]
## mean_error  0.003371
## me_mean     -0.2795
## MAE         0.3351
## MSE         0.2079
## MSNE        NaN
## cor_obspred 0.5624
## cor_predres -0.04639
## RMSE        0.4559
## RMSE_sd     0.8276
## URMSE       0.4559
## iqr         0.5299
```

6.3 Data mining: random forest

M. Guevara, C. Thine, G.F. Olmedo & R.R. Vargas

6.3.1 Overview

Data mining uses different forms of statistics, such as machine learning, to explore data matrices for a particular situation, from specific information sources, and with a specific objective. Data mining is used in DSM frameworks to generate spatial and temporal predictions of soil properties or classes in places where no information is available.

Under a data mining-based DSM framework, the exploration of statistical relationships (linear and non-linear) between soil observational data and soil environmental predictors is generally performed by the means of machine learning. Machine learning methods represent a branch of statistics that can be used to automatically extract information from available data, including the non-linear and hidden relationships of high dimensional spaces or hyper-volumes of information when high performance or distributed computing resources are available. Machine learning methods do not rely on statistical assumptions about the spatial structure of soil variability or the empirical relationship of soil available data and its environmental predictors. Therefore machine learning methods are also suitable for digital soil mapping under limited and sparse scenarios of data availability, although in practice the statistical performance of machine learning (or any statistical method) is reduced by a low representativeness of a soil property or class in the statistical space given available data. Machine learning methods can be used for (supervised and unsupervised) regression (e.g., predicting soil organic carbon) or classification (e.g., predicting soil type classes) on digital soil mapping. Machine learning methods can be roughly divided into four main groups: linear-based (e.g., MLR), kernel-based (e.g., kernel weighted nearest neighbors or SVM), probabilistic-based (e.g., Bayesian statistics) and tree-based (e.g., classification and regression trees).

Random forest is a tree-based machine learning algorithm that is popular in DSM because it has proven to be efficient mapping soil properties across a wide range of data scenarios and scales of soil variability. RF can be implemented using open source platforms and this Chapter is devoted to provide a reproducible example of this machine learning algorithm applied to SOC mapping across FYROM.

6.3.2 Random forest

Random forest is a decision-tree-based machine learning method used in DSM for uncovering the statistical relationship between a dependent variable (e.g., soil property) and its predictors. Decision-tree-based models (also known as classifiers) are literally like trees (e.g., with stem, many branches, and leaves). The leaves are the prediction outcomes (final decisions) that flow from higher levels based on decision rules through the stem and the branches (Breiman et al., 1984). The

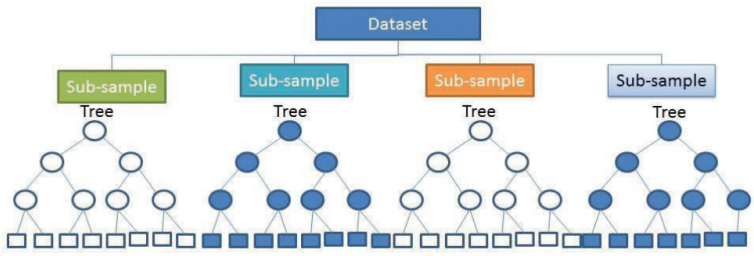


Figure 6.16: Schematical representation of data splitting to generate the random subsets used to train regression trees within a random forest model (ensemble of regression trees)

decision tree model recursively splits the data into final uniform groups (classes) or unique values based on a set of rules (e.g., based on probability values and hypothesis testing). In random forest, there are many decision trees and each tree recursively splits randomly selected sub-samples from the data (see Figure 6.16). The name for RF originates from the fact that the original data is first randomly split into sub-samples, and many decision trees (or forest) are used to model the sub-samples.

Random forest has been tested by many researchers on DSM (Poggio et al. (2013); Rad et al. (2014), and references therein). For SOC mapping (which for large areas usually rely on sparse datasets), it holds a lot of promises when compared to other prediction models, because it is practically free of assumptions. RF has shown accuracy of spatial predictions and, given the random selection of subsets and prediction factors, reduce potential over-fitting and data noise (Wiesmeier et al., 2011). Over-fitting and data noise are important uncertainty sources across high dimensionally spaces used to represent the soil forming environment. Thus, the advent of open-source platforms and freely downloadable ancillary data (e.g. http://worldgrids.org/) to represent the soil forming environment makes of RF and other such models increasingly appealing for DSM.

To predict continuous data (such as carbon density), RF generates an averaged ensemble of regression trees based on bagging, which is the statistical term for the random selection of subsets and predictors to generate each regression tree. Bagging is a bootstrapping aggregation technique where each sample is different from the original data set but resembles it in distribution and variability (Breiman, 2001, 1996). Each tree contributes to weight the statistical relationship between a dependent variable (e.g. soil property) and its prediction factors (e.g., terrain attributes, remote sensing, climate layers and/or legacy maps). Each tree (generated using a different subset of available data and random combinations of the prediction factors) is internally evaluated by an out-of-bag cross validation form which allows assessing the relative importance of the available prediction factors. Thus, higher weight is given to the most accurate trees (which use the most informative prediction factors). The final prediction of new data is the weighted average of all generated trees.

This method has been used to generate accurate predictions of SOC from the plot to the global scale and also in a country-specific basis (Hengl et al., 2014; Hengl T., 2017; Bonfatti et al., 2016; Guevara et al., 2018). RF can be implemented for DSM using open source code (e.g., **R** package **randomForest**, see Breiman, 2017) and public sources of environmental information.

The objective of this Chapter is to demonstrate a reproducible framework for the implementation of the RF algorithm applied for SOC predictive mapping, including the uncertainty of model estimates and using open source platforms for statistical computing.

6.3.3 Conceptual model and data preparation

To use RF for digital soil organic carbon mapping the SCORPAN (Soils, Climate, Organisms, Relief, Parent material, Age and N space) conceptual model (McBratney et al., 2003; Florinsky, 2012) will have take the following form:

$$\mathrm{SOC}_{x,y\ t} \sim \mathrm{randomForest}(E_{x,y\ t})$$

where soil organic carbon estimates (SOC) for a specific site (x, y) and for a specific period of time (t) can be modeled as a Random forest (randomForest) function of the soil forming environment ($Ex, y\ t$), which is represented by the SOC prediction factors (e.g., terrain attributes, remote sensing, climate layers and/or legacy maps).

To feed the right side of the equation, $SOC_{x,y\ t}$ is usually represented in a tabular form or a geospatial object (e.g., shapefile) with three fundamental columns. Two columns represent the spatial coordinates x and y (e.g. latitude and longitude) that are used to extract the values of the prediction factors for the representative locations of the SOC estimates. SOC estimates are represented in a third column (see previous Chapters of this book dealing with the transformation of soil carbon density to mass units). The left side of the equation is generally represented by gridded (raster) files, so all available sources of information should be first harmonized into a common pixel size and coordinate reference system.

6.3.4 Software

For the RF implementation, we will use the platform for statistical computing **R**. This open source object-oriented software relies on specific-contributor libraries. There are several libraries for the implementation of the RF algorithm in **R** as well as several variants of the method that can be used to solve DSM problems.

In this Section, we will show the use of Random forest using the **randomForest**, the **quantregForest**, the **raster** and the **caret R** packages. The quantile regression forest (**quantregForest**; Meinshausen, 2006) has two main advantages. First, it can be used to extract the variance of all the trees generated by RF, not just the mean (as in the original **randomForest** package), and therefore we can calculate the dispersion of the full conditional distribution of SOC as a function of the prediction factors, which given available data, represent the RF model uncertainty.

Second, the quantile regression forest approach can also run in parallel using all available computational resources, in a way that we can predict and estimate the uncertainty of predictions at reasonable time frames whit large datasets.

6.3.5 Tunning random forest model parameters

Two important parameters of RF are *mtry* and *ntree*. The *mtry* parameter controls the number of prediction factors that are randomly used on each tree, while the *ntree* parameter controls the number of trees generated by RF. These two parameters can be selected by the means of cross-validation to maximize the prediction capacity of RF.

We will use the **caret** package to select the most appropriate values for these parameters using 10-fold cross-validation (Kuhn et al., 2017). Tunning the main parameters of RF (or any other model) can be time-consuming in computational terms because implies the need to run and internally validate an independent model for each possible combination of parameter values. Thus, tunning the RF parameters would be relevant, given available data, to achieve the best possible accuracy of predictions.

6.3.6 Technical steps - Random forest

We will use the FYSOM dataset for this exercise (see previous Chapters of this book dealing with data preparation). The first dataset contains in a tabular form the *OCSKGM* values and the values of the prediction factors for the same locations (e.g., x, y, *OCSKGM*, covariate$_1$, covariate$_2$ etc.) while the second database is represented by a stack of raster files containing prediction factors across all the area of interest at the spatial resolution of 0.0083º (approx. 1 km). Import the datasets and load in **R** all our libraries of interest.

Step 1 - Data preparation

In the Chapter 5, we presented and prepared several global and continental datasets. In addition to these datasets, numerous covariate layers have been prepared by ISRIC for the GSOCmap project. These are GIS raster layers of various biophysical earth surface properties for each country in the world. Some of these layers will be used as predictors in this Section. Please download the covariates for your own study area from GSOCmap Data Repository as explained in Section 5.6.

In Section 5.9, a table with the points values after data preparation and the values of our spatial predictors was prepared. This step involves loading this table.

Now we will import our point dataset using `read.csv()` function. The easiest way to create a data frame is to read in data from a file. This is done using the function `read.csv()`, which works with comma delimited files. Data can be read in from other file formats as well, using different functions, but `read.csv()` is the most commonly used approach. **R** is very flexible in how it reads in data from text

files (`read.table()`, `read.csv()`, `read.csv2()`, `read.delim()`, `read.delim2()`). Please type `?read.table()` for help.

```
# Load data
dat <- read.csv("data/MKD_RegMatrix.csv")

dat$LCEE10 <- as.factor(dat$LCEE10)
dat$soilmap <- as.factor(dat$soilmap)

# Explore the data structure
str(dat)
```

```
## 'data.frame':    2897 obs. of  23 variables:
##  $ id       : Factor w/ 2897 levels "P0003","P0007",..: 1 2 3 4..
##  $ Y        : num  42 42 42.1 42 42 ...
##  $ X        : num  20.8 20.8 20.8 20.9 20.9 ...
##  $ SOC      : num  26.38 6.15 3.94 3.26 2.29 ...
##  $ BLD      : num  0.73 1.17 1.3 1.34 1.41 ...
##  $ CRFVOL   : num  8 18.6 31.9 21.7 14.5 ...
##  $ OCSKGM   : num  5.32 1.75 1.04 1.03 0.83 ...
##  $ meaERROR : num  2.16 2.85 2.65 3.16 3.63 2.83 2.94 2.49 2.77..
##  $ OCSKGMlog: num  1.6712 0.5591 0.0429 0.0286 -0.1862 ...
##  $ B04CHE3  : num  574 553 693 743 744 ...
##  $ B07CHE3  : num  38.5 37.8 42.1 43.7 43.7 ...
##  $ B13CHE3  : num  111.6 125 99.8 118.1 121 ...
##  $ B14CHE3  : num  59.2 60.3 42.4 39.9 38.7 ...
##  $ DEMENV5  : int  2327 2207 1243 1120 1098 1492 1413 1809 1731..
##  $ LCEE10   : Factor w/ 4 levels "1","2","3","4": 1 3 2 1 2 2 2..
##  $ PRSCHE3  : num  998 1053 780 839 844 ...
##  $ SLPMRG5  : int  13 36 6 25 30 24 15 17 20 43 ...
##  $ TMDMOD3  : int  282 280 285 288 289 287 286 286 287 286 ...
##  $ TMNMOD3  : int  272 270 277 279 279 277 277 273 274 273 ...
##  $ TWIMRG5  : int  61 62 81 66 65 72 68 67 65 59 ...
##  $ VBFMRG5  : int  0 0 14 0 0 0 0 0 0 0 ...
##  $ VDPMRG5  : int  311 823 10048 1963 -173 -400 -9 -692 -1139 2..
##  $ soilmap  : Factor w/ 20 levels "1","2","3","4",..: 6 14 14 3..
```

Since we will be working with spatial data we need to define the coordinates for the imported data. Using the `coordinates()` function from the **sp** package we can define the columns in the data frame to refer to spatial coordinates. Here the coordinates are listed in columns `X` and `Y`.

```
library(sp)

# Promote to spatialPointsDataFrame
coordinates(dat) <- ~ X + Y

class(dat)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

`SpatialPointsDataFrame` structure is essentially the same as a data frame, except that additional *spatial* elements have been added or partitioned into slots. Some important ones being the bounding box (sort of like the spatial extent of the data), and the coordinate reference system `proj4string()`, which we need to define for the sample dataset. To define the CRS, we must know where our data are from, and what was the corresponding CRS used when recording the spatial information in the field. For this data set, the CRS used was WGS84 (EPSG:4326).

To clearly tell **R** this information we define the CRS which describes a reference system in a way understood by the PROJ.4 projection library. An interface to the PROJ.4 library is available in the **rgdal** package. As an alternative to using PROJ.4 character strings, we can use the corresponding yet simpler EPSG code. **rgdal** also recognizes these codes. If you are unsure of the PROJ.4 or EPSG code for the spatial data that you have but know the CRS, you should consult http://spatialreference.org/ for assistance.

> **CRS**: Please note that, when working with spatial data, it is very important that the CRS of the point data and covariates are the same.

```
dat@proj4string <- CRS(projargs = "+init=epsg:4326")

dat@proj4string
```

```
## CRS arguments:
##  +init=epsg:4326 +proj=longlat +datum=WGS84 +no_defs
## +ellps=WGS84 +towgs84=0,0,0
```

Now we will import the covariates. When the covariate layers are in common resolution and extent, rather than working with individual rasters it is better to stack them all into a single **R** object. In this example, we use 13 covariates from the GSOCmap Data Repository and a rasterized version of the soil type map. The rasterization of vectorial data was covered in Technical Steps - Rasterizing a vector layer in R. The file containing all the covariates was prepared at the end of Chapter 5.

```
load(file = "covariates.RData")

names(covs)
```

```
##  [1] "B04CHE3" "B07CHE3" "B13CHE3" "B14CHE3" "DEMENV5" "LCEE10"
##  [7] "PRSCHE3" "SLPMRG5" "TMDMOD3" "TMNMOD3" "TWIMRG5" "VBFMRG5"
## [13] "VDPMRG5" "soilmap"
```

Random forest does not have assumptions about the statistical distribution of the response variable, but it is a good practice prior to model building to analyze the statistical distribution of the response variable (e.g., if is normal or not) and its relationships with the prediction factors. Soil organic carbon tends to have a

log-normal distribution with a right-skew, and transforming the original values to its natural logarithm would generate a normal distribution of soil organic carbon values.

For further analysis, we will use the dataset transformed to its natural logarithm (*OCSKGMlog*) because this transformation, given this dataset, increases the correlation of the response variable and the covariate space.

Keep in mind that selecting the most appropriate prediction factors is required to generate an interpretable model and high accuracy of prediction in places where no information is available. Variable selection ideally should incorporate expert soil knowledge about the study area and statistical criteria (e.g., just to use the best-correlated predictors). Multivariate analysis (e.g., principal component analysis) is a widely used approach to identify informative predictors. Here we use this combination of prediction factors to be consistent with other Chapters of this book and because they were previously selected for this exercise using expert knowledge about the spatial variability of soil organic carbon.

Now, we will build a working hypothesis from our conceptual model, using all the continuous prediction factors for *OCSKGMlog*:

OCSKGMlog ~ randomForest *B04CHE3 + B07CHE3 + B13CHE3 + B14CHE3 + DEMENV5 + LCEE10 + PRSCHE3 + SLPMRG5 + TMDMOD3 + TMNMOD3 + TWIMRG5 + VBFMRG5 + VDPMRG5*

```
# For its use on R we need to define a model formula
fm = as.formula(paste("log(OCSKGM) ~", paste0(names(covs[[-14]]),
                                              collapse = "+")))
```

This is the **R** syntax to define a model formula required for the model structure, where soil organic carbon transformed to its natural logarithm (*OCSKGMlog*) can be predicted as a function of the available prediction factors, each explained in Chapter 5 of this book, e.g., B04CHE3, B07CHE3, B13CHE3, B14CHE3, DEMENV5 , LCEE10, PRSCHE3, SLPMRG5, TMDMOD3, TMNMOD3, TWIMRG5, VBFMRG5, VDPMRG5.

Note that the variable soil map is categorical, so is not included in the correlation analysis. In fact, although soil type polygon maps are in theory powerful predictors for *OCSKGM* we will not use this map for this exercise, because not all categories in the map are represented by available *OCSKGM* estimates, therefore this map requires a generalization of soil type units in function of the classes represented by the sites of *OCSKGM* estimates, which is beyond the scope of this Chapter.

Ideally, the number of observations across all the categories of soil type or any other factorial variable should be balanced. Another alternative to using an unbalanced categorical map is by generating dummy variables, where each category in the map becomes an independent binomial predictor variable (e.g., only 0 and 1 values) as is explained in the following Chapter. The risk of doing so rely upon the potential underestimation of the spatial variability of the target variable under each category with a low density of available data.

Step 2 - Tuning parameters

Now we will use the cross-validation strategy implemented in the train function of the **caret** package (Kuhn et al., 2017), which default is 10-fold. The result of this function includes information to select the best *mtry* parameter and to decide the appropriate number of trees. The out-of-bag RMSE will be used to select the optimal *mtry* model. To analyze the *ntree* parameter we will plot the number of trees against the out-of-bag RMSE, an optimal *ntree* can be selected with the number of trees when these relationships stabilizes at the minimum possible RMSE (in the y axis). Reducing the number of trees will reduce the computational demand, which is specially important when dealing with large databases. In the presence of multidimensional and highly correlated prediction factors, avoiding an excessive number of trees will also reduce the risk of model overfitting.

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
# Default 10-fold cross-validation
ctrl <- trainControl(method = "cv", savePred=T)

# Search for the best mtry parameter
rfmodel <- train(fm, data=dat@data, method = "rf", trControl = ctrl,
              importance=TRUE)

# This is a very useful function to compare and test different
# prediction algorithms.
# Type names(getModelInfo()) to see all the
# possibilitites implemented on this function.
```

The object derived from the train function can be used to generate predictions of *OCSKGMlog* at the spatial resolution of the prediction factors. Before generating predictions, we will plot the most important predictors sorted in decreasing order of importance. From the variable importance plot, MSE represent an informative measure for variable selection. It is the increase in error (mean squared error, MSE) of predictions which was estimated with out-of-bag-cross validation as a result of

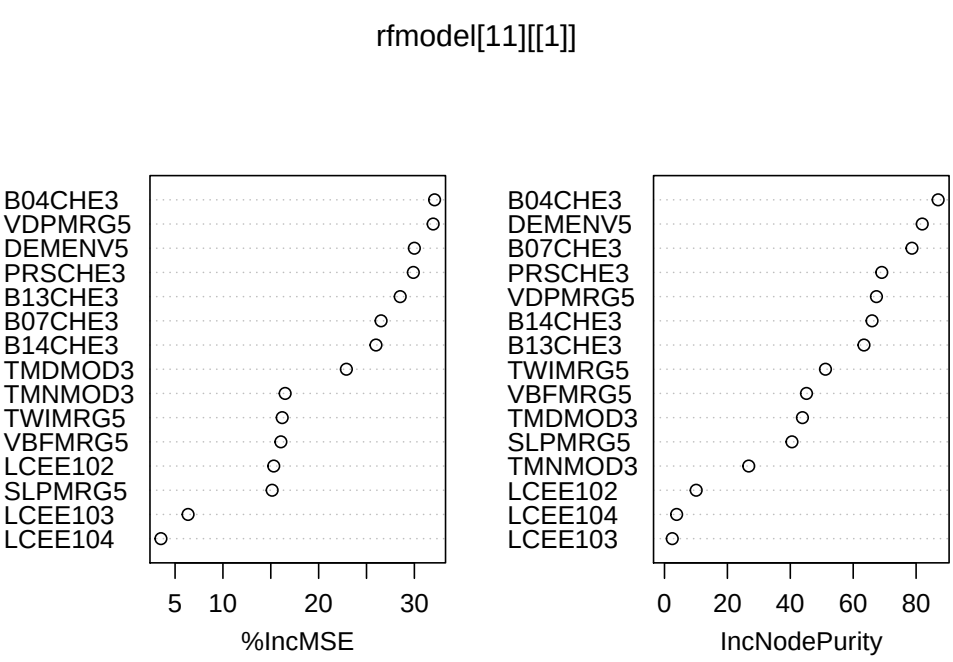


Figure 6.17: Model Decreasing Error and Node Purity for the RF model

prediction factor being permuted with values randomly shuffled. This is one of the strategies that RF uses to reduce overfitting.

```
# Variable importance plot, compare with the correlation matrix
# Select the best prediction factors and repeat
varImpPlot(rfmodel[11][[1]])
```

```
# Check if the error stabilizes
plot(rfmodel[11][[1]])
```

Random forest users are encouraged to compare and test the prediction capacity of different combinations of prediction factors in order to reduce the complexity of the model and the statistical redundancy of environmental information on further applications of predicted OCSKGM maps (e.g., quantifying the carbon dynamics). The resulting map of our RF model needs to be validated using the independent dataset to complement the results of the cross-validation (e.g., RMSE and explained variance) derived using the train function and to have a more comprehensive interpretation of accuracy and bias. Note how the RMSE and the explained variance derived from the independent validations are slightly lower than the values obtained using cross-validation.

```
# Make a prediction across all FYROM
# Note that the units are still in log
pred <- predict(covs, rfmodel)
```

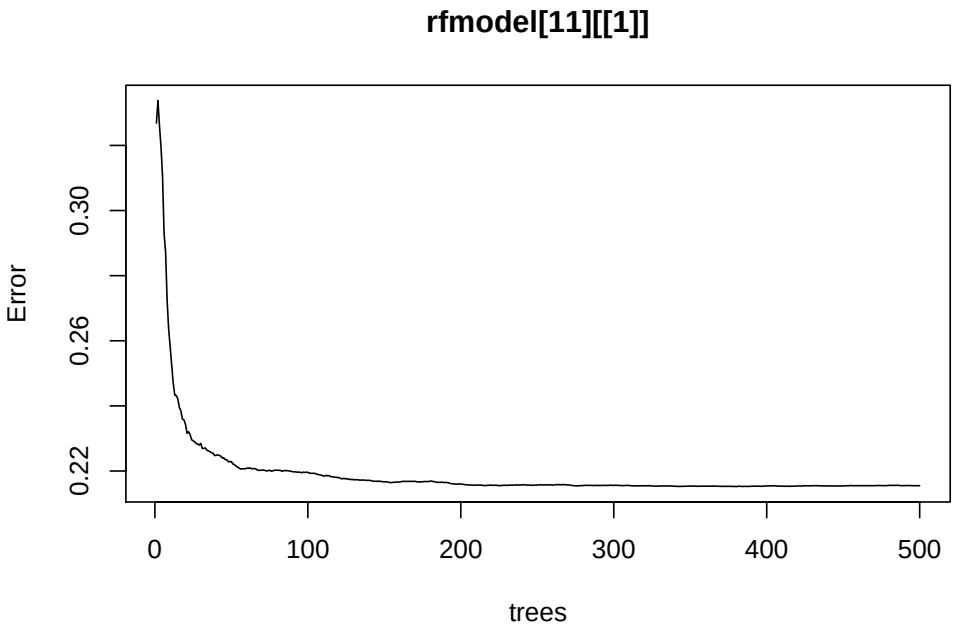


Figure 6.18: Select ntree

```
## Warning in .local(object, ...): not sure if the correct factor
## levels are used here
```

```
# Back transform predictions log transformed
pred <- exp(pred)

# Save the result as a tiff file
writeRaster(pred, filename = "results/MKD_OCSKGM_rf.tif",
            overwrite=TRUE)

plot(pred)
```

6.3.7 Technical steps - Using quantile regression forest to estimate uncertainty

Ideally, a digital soil map should include a spatial explicit metric of uncertainty. The uncertainty can be roughly divided into four main components, uncertainty in soil data, uncertainty in soil covariates, uncertainty in the model and uncertainty in variations of available data. Here, we show an approach to estimate the sensitivity of the model to available data and the uncertainty of the model. The first two are beyond of the aim of this Chapter. For the third and fourth we will generate a reproducible example.

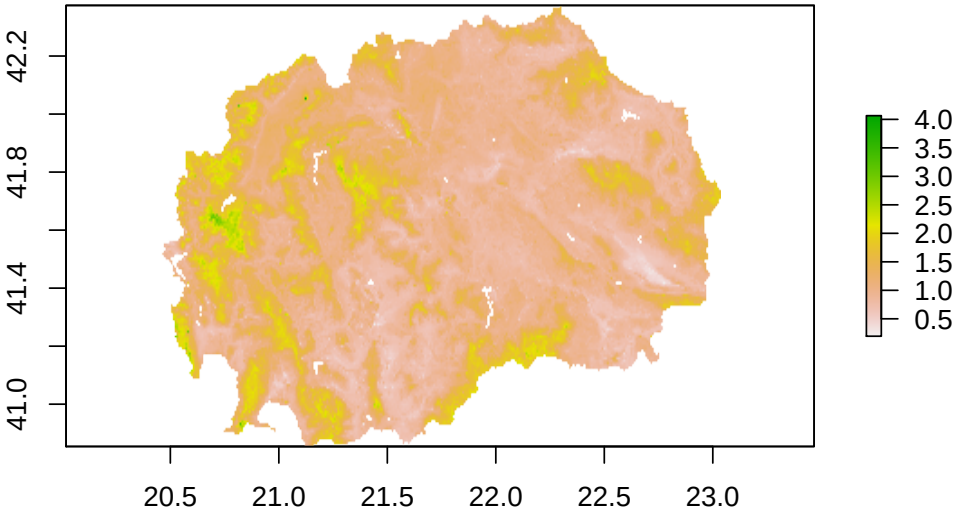


Figure 6.19: SOC prediction map for FYROM using a random forest model

Step 1 - Split the data in into training and testing subsets

To analyze the sensitivity of the model to available data we need to randomly split the data several times (e.g., 10 or more, is possible until the variance stabilizes) in training and testing subsets. A model generation and prediction are made on each split, in a way that the dispersion of the predicted values at the pixel level will represent the uncertainty and the sensitivity of the model to variations in available data.

This process increases computational demand and memory since it will repeat n times (10 in this example) the model and the prediction using each time a different random combination of data for training and testing the models. As larger the sample and the number of realizations the more robust our validation strategy. For this example, we will use only 10 realizations and random splits of 25% of available data.

```
library(Metrics)

# Generate an empty dataframe
validation <- data.frame(rmse=numeric(), r2=numeric())

# Sensitivity to the dataset
# Start a loop with 10 model realizations
for (i in 1:10){
  # We will build 10 models using random samples of 25%
  smp_size <- floor(0.25 * nrow(dat))
  train_ind <- sample(seq_len(nrow(dat)), size = smp_size)
  train <- dat[train_ind, ]
```

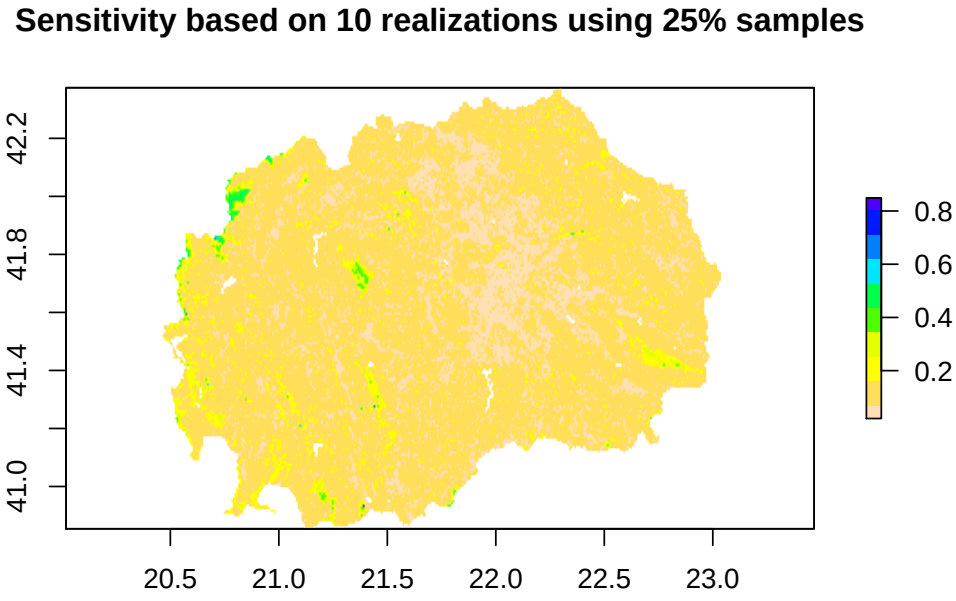


Figure 6.20: Sensitivity based on 10 realizations using 25% samples

```
  test <- dat[-train_ind, ]
  modn <- train(fm, data=train@data, method = "rf",
                trControl = ctrl)
  pred <- stack(pred, predict(covs, modn))
  test$pred <- extract(pred[[i+1]], test)
  # Store the results in a dataframe
  validation[i, 1] <- rmse(test$OCSKGMlog, test$pred)
  validation[i, 2] <- cor(test$OCSKGMlog, test$pred)^2
}
```

Step 2 - Build a sensitivity map

```
# The sensitivity map is the dispersion of all individual models
sensitivity <- calc(pred[[-1]], sd)

plot(sensitivity, col=rev(topo.colors(10)),
     main='Sensitivity based on 10 realizations using 25% samples')
```

The RMSE and the explained variance of the models is stored in the object validation (type `summary(validation)`). The standard deviation of all the ten predictions allows generating a map of model sensitivity to available data.

```
# Sensitivity of validation metrics
summary(validation)
```

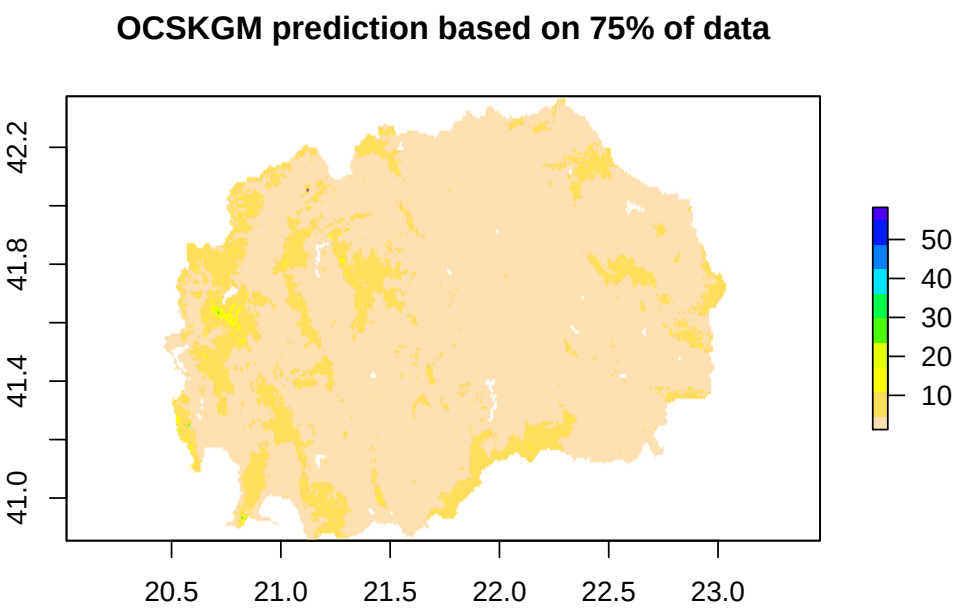


Figure 6.21: OCSKGM prediction based on 75% of data

```
##       rmse              r2
##  Min.   :0.4736   Min.   :0.2317
##  1st Qu.:0.4761   1st Qu.:0.2385
##  Median :0.4787   Median :0.2435
##  Mean   :0.4790   Mean   :0.2454
##  3rd Qu.:0.4804   3rd Qu.:0.2535
##  Max.   :0.4910   Max.   :0.2611
```

Step 3 - Analyze the sensitivity of the map

```
# Plot of the map based on 75% of data and the sensitivity to data
# variations
prediction75 <- exp(pred[[1]])

plot(prediction75, main='OCSKGM prediction based on 75% of data',
     col=rev(topo.colors(10)))
```

Step 4 - Estimate the full conditional distribution of OCSKGMlog

Finally, we will estimate the model uncertainty, represented by the full conditional distribution of the response variable (*OCSKGMlog*) as a function of the selected prediction factors using the quantile regression forest package **quantregForest** of **R**. This approach has proven to be efficient for DSM across large areas (Vaysse and Lagacherie, 2017).

```
# Use quantile regression forest to estimate the full conditional
# distribution of OCSKGMlog, note that we are using the mtry
# parameter that was selected by the train function of the caret
# package, assuming that the 75% of data previously used well
# resembles the statistical distribution of the entire data
# population. Otherwise, repeat the train function with all
# available data (using the object dat that instead of train)
# to select mtry.
library(quantregForest)
```

```
## Loading required package: RColorBrewer
```

```
model <- quantregForest(y=dat@data$OCSKGMlog, x=dat@data[,8:20],
                        ntree=500, keep.inbag=TRUE,
                        mtry = as.numeric(rfmodel$bestTune))
```

Step 5 - Estimate the probability distribution function for each pixel

This method will calculate a probability distribution function for each pixel and therefore can be time-consuming. Therefore we will run it using parallel computing. Note that the code to run in parallel this analysis can also be passed to the previous predictions (predict function). The result will be a map of the standard deviation of the distribution calculated for each pixel, which represents the extreme values that a prediction can take for a specific site (e.g., pixel) given available data and predictors. Note that this analysis is performed using all available data and a second map of OCSKGM is created.

```
library(snow)

# Estimate model uncertainty at the pixel level using parallel
# computing
# Define number of cores to use
beginCluster()
```

```
## 8 cores detected, using 7
```

```
# Estimate model uncertainty
unc <- clusterR(covs, predict, args=list(model=model,what=sd))

# OCSKGMlog prediction based in all available data
mean <- clusterR(covs, predict,
                 args=list(model=model, what=mean))

# The total uncertainty is the sum of sensitivity and model
# uncertainty
unc <- unc + sensitivity

# Express the uncertainty in percent % (divide by the mean)
Total_unc_Percent <- exp(unc)/exp(mean)
```

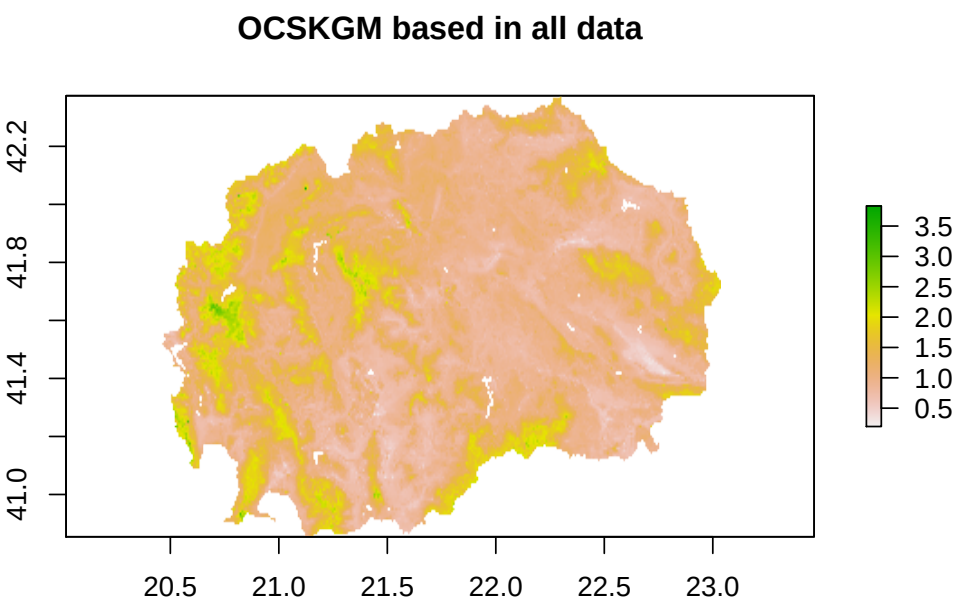


Figure 6.22: OCSKGM quantregForest prediction

```
endCluster()
```

Our final prediction uses all available data, while the total uncertainty (in percent) is represented by the sum of the quantile regression forest standard deviation and the sensitivity map from the previous Section. The total uncertainty is then divided by the prediction to obtain a percent map, which is easier to interpret.

```
# Plot both maps (the predicted OCSKGM and associated uncertainty)
plot(exp(mean), main='OCSKGM based in all data')
```

```
plot(Total_unc_Percent, zlim=c(0, 5), main='Total uncertainty')
```

Step 6 - Save the results as raster format

Finally, the predicted OCSKGM and the total uncertainty can be saved in the working directory in a generic (*.tif) raster format.

```
# Save the resulting maps in separated *.tif files
writeRaster(exp(mean), file='results/MKD_OCSKGM_quantrf.tif',
            overwrite=TRUE)
writeRaster(Total_unc_Percent, file='results/MKD_OCSKGM_rf_unc.tif',
            overwrite=TRUE)
```

We have created two maps in the working directory, one represents the predicted OCSKGM and the second one its uncertainty, which is the sum of the model sensitivity to data variations and the full conditional distribution of the response

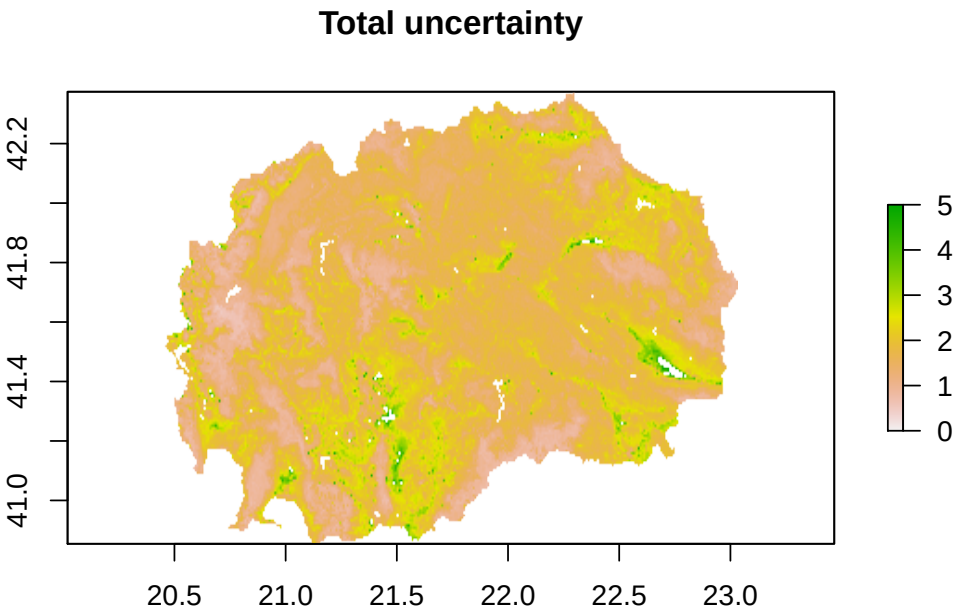


Figure 6.23: Total uncertainty

variable as a function of available prediction factors. The following Chapters of this book will show you how to prepare a stock report based on this soil carbon digital soil maps.

6.4 Data mining: support vector machines

G.F. Olmedo & M. Guevara

6.4.1 Overview

Support vector machines is a kernel-based machine learning technique suitable for mapping SOC. SVM use decision surfaces (defined by a kernel function) to map non-linear relationships across a high-dimension induced feature space (Cortes and Vapnik, 1995). SVM is widely used to perform classification and regression analysis on DSM.

According to Pedregosa et al. (2011) the advantages of SVM are:

- Effective in high dimensional spaces.
- Still effective in cases where the number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

And the disadvantages of SVM include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVM do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

In DSM, the problems usually involve working in high dimensional spaces (were the dimensions are the covariates) with a limited number of samples. SVM is a technique mostly used in classification problems, but it can be used to solve regression problems, such as modeling the continuous variability of SOC using environmental covariates. When SVM is used to solve a regression problem, it is called support vector regression.

Support vector regression applies a simple linear method to the data but in a high-dimensional feature space non-linearly related to the input space. It creates n hyperplanes through the n-dimensional spectral-space and each hyperplanes separates numerical data based on a Kernel function (e.g., Gaussian). SVM uses parameters such as *gamma*, *cost* and *epsilon*. These parameters are used to define the shape of the hyperplane, including the margin from the closest point to the hyperplane that divides data with the largest possible margin and defines the tolerance to errors on each single training. Linear models are fitted to the support vectors and used for prediction purposes. The support vectors are the points which fall within each hyperplane (Guevara et al., 2018).

In the example below, we will use the implementation of SVM in the R package `e1071` (Meyer et al., 2017). The package `e1071` offers an interface to the award-winning C++ implementation by Chih-Chung Chang and Chih-Jen Lin, libsvm (current version: 2.6). For further implementation details on libsvm, see Chang and Lin (2001).

> **SVM**: This approach is a broad research area and for a better understanding of the mathematical background we can recommend the following books: Vapnik (2013), Friedman et al. (2001), and James et al. (2013).

6.4.2 Technical steps - Fitting an SVM model to predict the SOC

Step 1 - Setting working space and initial steps

One of the first steps should be setting our working directory. If you read/write files from/to disk, this takes place in the working directory. If we do not set the working directory, we could easily write files to an undesirable file location. The following example shows how to set the working directory in **R** to our folder which contains data for the study area (point data, covariates).

Note that we must use the forward slash / or double backslash \\ in **R**! Single backslash \ will not work. Now we can check if the working directory has been correctly set by using the `getwd()` function:

```
getwd()
```

Step 2 - Data preparation

In the Chapter 5, we presented and prepared several global and continental datasets. In addition to these datasets, numerous covariate layers have been prepared by ISRIC for the GSOCmap project. These are GIS raster layers of various biophysical earth surface properties for each country in the world. Some of these layers will be used as predictors in this Section. Please download the covariates for your own study area from GSOCmap Data Repository as explained in Section 5.6.

In Section 5.9, a table with the points values after data preparation and the values of our spatial predictors was prepared. This step involves loading this table.

Now we will import our point dataset using `read.csv()` function. The easiest way to create a data frame is to read in data from a file. This is done using the function `read.csv()`, which works with comma delimited files. Data can be read in from other file formats as well, using different functions, but `read.csv()` is the most commonly used approach. **R** is very flexible in how it reads in data from text files (`read.table()`, `read.csv()`, `read.csv2()`, `read.delim()`, `read.delim2()`). Please type `?read.table()` for help.

```
# Load data
dat <- read.csv("data/MKD_RegMatrix.csv")

dat$LCEE10 <- as.factor(dat$LCEE10)
dat$soilmap <- as.factor(dat$soilmap)

# Explore the data structure
str(dat)
```

```
## 'data.frame':    2897 obs. of  23 variables:
##  $ id       : Factor w/ 2897 levels "P0003","P0007",..: 1 2 3 4..
##  $ Y        : num  42 42 42.1 42 42 ...
##  $ X        : num  20.8 20.8 20.8 20.9 20.9 ...
##  $ SOC      : num  26.38 6.15 3.94 3.26 2.29 ...
##  $ BLD      : num  0.73 1.17 1.3 1.34 1.41 ...
##  $ CRFVOL   : num  8 18.6 31.9 21.7 14.5 ...
##  $ OCSKGM   : num  5.32 1.75 1.04 1.03 0.83 ...
##  $ meaERROR : num  2.16 2.85 2.65 3.16 3.63 2.83 2.94 2.49 2.77..
##  $ OCSKGMlog: num  1.6712 0.5591 0.0429 0.0286 -0.1862 ...
##  $ B04CHE3  : num  574 553 693 743 744 ...
##  $ B07CHE3  : num  38.5 37.8 42.1 43.7 43.7 ...
##  $ B13CHE3  : num  111.6 125 99.8 118.1 121 ...
##  $ B14CHE3  : num  59.2 60.3 42.4 39.9 38.7 ...
##  $ DEMENV5  : int  2327 2207 1243 1120 1098 1492 1413 1809 1731..
##  $ LCEE10   : Factor w/ 4 levels "1","2","3","4": 1 3 2 1 2 2 2..
##  $ PRSCHE3  : num  998 1053 780 839 844 ...
##  $ SLPMRG5  : int  13 36 6 25 30 24 15 17 20 43 ...
##  $ TMDMOD3  : int  282 280 285 288 289 287 286 286 287 286 ...
##  $ TMNMOD3  : int  272 270 277 279 279 277 277 273 274 273 ...
##  $ TWIMRG5  : int  61 62 81 66 65 72 68 67 65 59 ...
##  $ VBFMRG5  : int  0 0 14 0 0 0 0 0 0 0 ...
##  $ VDPMRG5  : int  311 823 10048 1963 -173 -400 -9 -692 -1139 2..
##  $ soilmap  : Factor w/ 20 levels "1","2","3","4",..: 6 14 14 3..
```

Since we will be working with spatial data we need to define the coordinates for the imported data. Using the `coordinates()` function from the **sp** package we can define the columns in the data frame to refer to spatial coordinates. Here the coordinates are listed in columns `X` and `Y`.

```
library(sp)

# Promote to spatialPointsDataFrame
coordinates(dat) <- ~ X + Y

class(dat)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
```

```
## [1] "sp"
```

`SpatialPointsDataFrame` structure is essentially the same as a data frame, except that additional *spatial* elements have been added or partitioned into slots. Some important ones being the bounding box (sort of like the spatial extent of the data), and the coordinate reference system `proj4string()`, which we need to define for the sample dataset. To define the CRS, we must know where our data are from, and what was the corresponding CRS used when recording the spatial information in the field. For this data set, the CRS used was WGS84 (EPSG:4326).

To clearly tell **R** this information we define the CRS which describes a reference system in a way understood by the PROJ.4 projection library. An interface to the PROJ.4 library is available in the **rgdal** package. As an alternative to using PROJ.4 character strings, we can use the corresponding yet simpler EPSG code. **rgdal** also recognizes these codes. If you are unsure of the PROJ.4 or EPSG code for the spatial data that you have but know the CRS, you should consult http://spatialreference.org/ for assistance.

> **CRS**: Please note that, when working with spatial data, it is very important that the CRS of the point data and covariates are the same.

```
# Now, we will define our CRS
dat@proj4string <- CRS(projargs = "+init=epsg:4326")

dat@proj4string
```

```
## CRS arguments:
##  +init=epsg:4326 +proj=longlat +datum=WGS84 +no_defs
## +ellps=WGS84 +towgs84=0,0,0
```

Now we will import the covariates. When the covariate layers are in common resolution and extent, rather than working with individual rasters it is better to stack them all into a single **R** object. In this example, we use 13 covariates from the GSOCmap Data Repository and a rasterized version of the soil type map. The rasterization of vectorial data was covered in Technical Steps - Rasterizing a vector layer in R. The file containing all the covariates was prepared at the end of Chapter 5.

```
load(file = "covariates.RData")

names(covs)
```

```
##  [1] "B04CHE3" "B07CHE3" "B13CHE3" "B14CHE3" "DEMENV5" "LCEE10"
##  [7] "PRSCHE3" "SLPMRG5" "TMDMOD3" "TMNMOD3" "TWIMRG5" "VBFMRG5"
## [13] "VDPMRG5" "soilmap"
```

Step 3 - Variable selection using correlation analysis

```
# Plot the names of the covariates
names(dat@data)
```

```
##  [1] "id"        "SOC"       "BLD"       "CRFVOL"    "OCSKGM"
```

```
##  [6] "meaERROR"  "OCSKGMlog" "B04CHE3"   "B07CHE3"   "B13CHE3"
## [11] "B14CHE3"   "DEMENV5"   "LCEE10"    "PRSCHE3"   "SLPMRG5"
## [16] "TMDMOD3"   "TMNMOD3"   "TWIMRG5"   "VBFMRG5"   "VDPMRG5"
## [21] "soilmap"
```

For the variable selection we will use `cor()` function. `x` must be a table including only the column with the response variable, and `y` must be a table including **only** the covariates. Besides, remember `dat@data` in the `data.frame` included in the `SpatialPointsDataFrame`. For `y`, columns 1 to 7 are out, because they are not covariates. At the same time, correlation analysis cannot be applied to categorical covariates, this means that columns 13 and 21 have to be removed too.

```
selectedCovs <- cor(x = as.matrix(dat@data[,5]),
           y = as.matrix(dat@data[,-c(1:7,13,21)]))

# Print correlation results
selectedCovs
```

```
##          B04CHE3    B07CHE3  B13CHE3   B14CHE3   DEMENV5
## [1,] -0.4199537 -0.3926615 0.330696 0.3481847 0.3926275
##         PRSCHE3   SLPMRG5    TMDMOD3    TMNMOD3    TWIMRG5
## [1,] 0.3948779 0.2593964 -0.4077552 -0.2963631 -0.2525764
##          VBFMRG5    VDPMRG5
## [1,] -0.1156285 -0.3001934
```

Now we used the correlation results to select the top five covariates.

```
library(reshape)

x <- subset(melt(selectedCovs), value != 1 | value != NA)
x <- x[with(x, order(-abs(x$value))),]

idx <- as.character(x$X2[1:5])

dat2 <- dat[c('OCSKGM', idx)]
names(dat2)
```

```
## [1] "OCSKGM"  "B04CHE3" "TMDMOD3" "PRSCHE3" "B07CHE3" "DEMENV5"
```

```
COV <- covs[[idx]]

# Selected covariates
names(COV)
```

```
## [1] "B04CHE3" "TMDMOD3" "PRSCHE3" "B07CHE3" "DEMENV5"
```

Step 4 - Categorical variables in svm models

According to Hsu et al. (2003), SVM requires each variable to be represented by a vector of real numbers. This means that factor variables, like `covs$LCEE10` and `covs$soilmaphas` to be converted into numeric data. In statistics, this kind of variables are called Boolean indicators or *dummy variables*.

Dummy variables take a value of 0 or 1 indicating the presence or absence of a specific value/category in our factor covariate, i.e. if we have five categories like in `covs$LCEE10`, we will have five dummy variables indicating the presence/absence of every category. For converting our covariates to dummies we will have to create a new function that returns the dummy raster stack `dummyRaster` from the factor version of the raster layer.

```
dummyRaster <- function(rast){
  rast <- as.factor(rast)
  result <- list()
  for(i in 1:length(levels(rast)[[1]][[1]])){
    result[[i]] <- rast == levels(rast)[[1]][[1]][i]
    names(result[[i]]) <- paste0(names(rast),
                                 levels(rast)[[1]][[1]][i])
  }
  return(stack(result))
}
```

We can use the function we just created to convert our categorical covariates to dummies and then stack all the layers together.

```
# Convert soilmap from factor to dummy
soilmap_dummy <- dummyRaster(covs$soilmap)

# Convert LCEE10 from factor to dummy
LCEE10_dummy <- dummyRaster(covs$LCEE10)

# Stack the 5 COV layers with the 2 dummies
COV <- stack(COV, soilmap_dummy, LCEE10_dummy)

# Print the final layer names
names(COV)
```

```
##  [1] "B04CHE3"   "TMDMOD3"   "PRSCHE3"   "B07CHE3"   "DEMENV5"
##  [6] "soilmap1"  "soilmap2"  "soilmap3"  "soilmap4"  "soilmap5"
## [11] "soilmap6"  "soilmap7"  "soilmap8"  "soilmap9"  "soilmap10"
## [16] "soilmap11" "soilmap12" "soilmap13" "soilmap14" "soilmap15"
## [21] "soilmap16" "soilmap17" "soilmap18" "soilmap19" "soilmap20"
## [26] "LCEE101"   "LCEE102"   "LCEE103"   "LCEE104"
```

We have to convert the columns with categorical variables in the soil samples `data.frame` to dummies as well. For doing this we can use function `model.matrix()`. After this, we use `cbind()` to merge the resulting `data.frame`.

```
# Convert soilmap column to dummy, the result is a matrix
# To have one column per category we have to add -1 to the formula
dat_soilmap_dummy <- model.matrix(~soilmap -1, data = dat@data)

# Convert the matrix to a data.frame
dat_soilmap_dummy <- as.data.frame(dat_soilmap_dummy)
```

```
# Convert LCEE10 column to dummy, the result is a matrix
# To have one column per category we have to add -1 to the formula
dat_LCEE10_dummy <- model.matrix(~LCEE10 -1, data = dat@data)

# Convert the matrix to a data.frame
dat_LCEE10_dummy <- as.data.frame(dat_LCEE10_dummy)

dat@data <- cbind(dat@data, dat_LCEE10_dummy, dat_soilmap_dummy)

names(dat@data)
```

```
##  [1] "id"         "SOC"        "BLD"        "CRFVOL"     "OCSKGM"
##  [6] "meaERROR"   "OCSKGMlog" "B04CHE3"    "B07CHE3"    "B13CHE3"
## [11] "B14CHE3"    "DEMENV5"    "LCEE10"     "PRSCHE3"    "SLPMRG5"
## [16] "TMDMOD3"    "TMNMOD3"    "TWIMRG5"    "VBFMRG5"    "VDPMRG5"
## [21] "soilmap"    "LCEE101"    "LCEE102"    "LCEE103"    "LCEE104"
## [26] "soilmap1"   "soilmap2"   "soilmap3"   "soilmap4"   "soilmap5"
## [31] "soilmap6"   "soilmap7"   "soilmap8"   "soilmap9"   "soilmap10"
## [36] "soilmap11" "soilmap12" "soilmap13" "soilmap14" "soilmap15"
## [41] "soilmap16" "soilmap17" "soilmap18" "soilmap19" "soilmap20"
```

Step 5 - Fitting a SVM model

To improve the model performance, the parameters of the SVM can be tuned. In this example, we will show how to tune two parameters using a grid search for hyperparameter optimization using the function `tune()`.

The first parameter is *epsilon* which is the insensitive-loss function. The larger *epsilon* is, the larger errors in the solution are not penalized. The default value for *epsilon* is 0.1, and we will try 11 different values from 0.05 to 0.12 in 0.1 increments. The second parameter is the cost which is the cost of constraints violation – it is the 'C'-constant of the regularization term in the Lagrange formulation. The default value for this parameter is 1, and we will try values from 1 to 20 in 5 increments. The value of cost helps us to avoid overfitting. This is a heavy and time consuming computational step since we will try a extensive number of different models in order to find the best parameters for our svm model.

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:raster':
##
##     interpolate
```

```
library(e1071)
library(caret)

# Test different values of epsilon and cost
  tuneResult <- tune(svm, OCSKGM ~.,  data = dat@data[,c("OCSKGM",
```

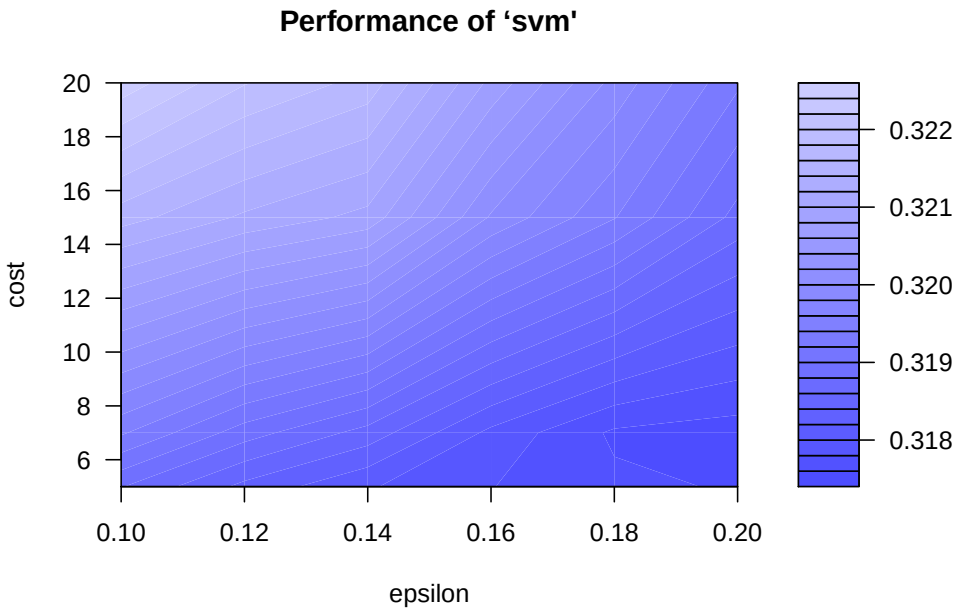


Figure 6.24: Performance of the different SVM models in the parameter tuning procedure

```
                                                    names(COV))],
                    ranges = list(epsilon = seq(0.1,0.2,0.02),
                                  cost = c(5,7,15,20)))
```

We can plot the performance of the different models. When the region is darker, the RMSE is closer to zero.

```
plot(tuneResult)
```

Step 6 - Select the model with the best combination of epsilon and cost

The best model is the one with the lowest mean squared error derived by cross-validation. The parameters for the cross-validation can be defined in the `tune.control()` function. By default, it uses cross-validation using 10 folds.

```
# Choose the model with the best combination of epsilon and cost
tunedModel <- tuneResult$best.model

print(tunedModel)
```

```
##
## Call:
## best.tune(method = svm, train.x = OCSKGM ~ ., data = dat@data[,
##     c("OCSKGM", names(COV))], ranges = list(epsilon = seq(0.1,
```

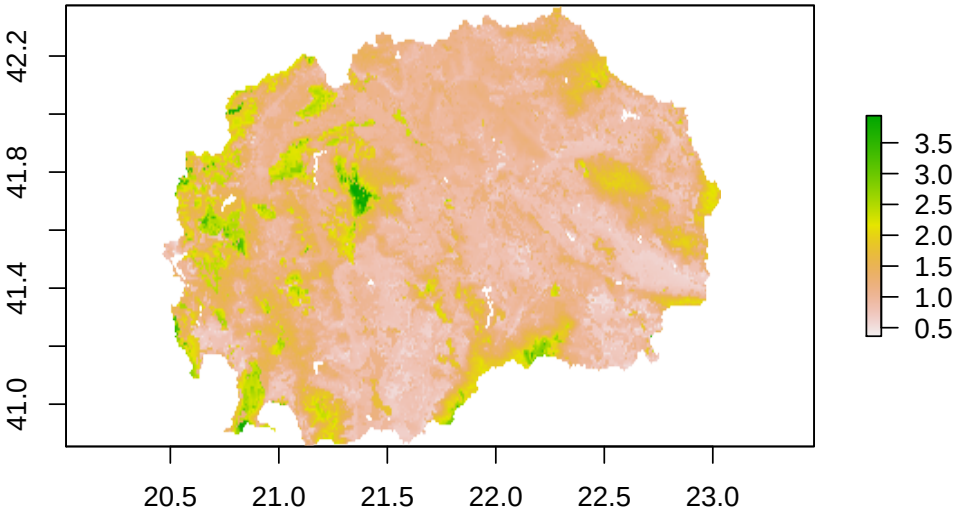


Figure 6.25: SOC prediction map for FYROM using a support vector machines model

```
##     0.2, 0.02), cost = c(5, 7, 15, 20)))
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  7
##       gamma:  0.03448276
##     epsilon:  0.2
##
##
## Number of Support Vectors:  2189
```

Step 7 - Predict the OCS using the model

```
# Use the model to predict the SOC in the covariates space
OCSsvm <- predict(COV, tunedModel)

# Save the result
writeRaster(OCSsvm, filename = "results/MKD_OCSKGM_svm.tif",
            overwrite=TRUE)

plot(OCSsvm)
```

Finally, we can evaluate the contribution of each covariate to the model (Guyon and Elisseeff, 2003).

```
# Variable importance in svm
# Code by: stackoverflow.com/questions/34781495
# Weight vectors
w <- t(tunedModel$coefs) %*% tunedModel$SV

# Weight
w <- apply(w, 2, function(v){sqrt(sum(v^2))})

w <- sort(w, decreasing = T)

print(w)
```

```
##      B04CHE3     soilmap6      TMDMOD3     soilmap1     soilmap7
## 7.562482e+01 4.741345e+01 4.438758e+01 4.097177e+01 3.770267e+01
##      DEMENV5      B07CHE3      PRSCHE3      LCEE103     soilmap2
## 3.442923e+01 3.092630e+01 2.902640e+01 2.711066e+01 2.575601e+01
##    soilmap19     soilmap5      LCEE104    soilmap16      LCEE101
## 2.496913e+01 1.733616e+01 1.664618e+01 1.590055e+01 1.231509e+01
##    soilmap15      LCEE102     soilmap4    soilmap20    soilmap17
## 1.204309e+01 1.118658e+01 9.247627e+00 9.229459e+00 7.526749e+00
##     soilmap3    soilmap10     soilmap9    soilmap12     soilmap8
## 7.487471e+00 6.294465e+00 6.017019e+00 3.637515e+00 2.564390e+00
##    soilmap18    soilmap14    soilmap11    soilmap13
## 1.262072e+00 1.046765e+00 7.097243e-01 2.398082e-14
```

SVM is a powerful technique which represent another welcome possibility to generate reliable and interpretable SOC predictions across different scales of data availability, including country-specific SOC maps.

Chapter 7

Validation

B. Kempen, D.J. Brus & G.B.M. Heuvelink

The authors of this Chapter used **R** packages. To run the code provided in this Chapter, the following packages need to be installed in the **R** user library. If the packages are not yet installed, the `install.packages()` function can be used.

```
# Installing packages for Chapter 'Validation'
install.packages(c("sp", "raster", "caret"))
```

7.1 What is validation?

No map is perfect. All maps, including soil maps, are representations of reality that are often based on an underlying model. This means that there will always be a deviation between the phenomenon depicted on the map and the phenomenon observed in the real world, i.e. each map will contain errors. The magnitude of the errors determines the quality of the map. If a map matches reality well (the error is small), the quality or accuracy of the map is high. On the other hand, if a map does not match reality well, map accuracy is low.

Soil maps are used for many purposes. For example to report on (changes in) SOC stocks, as input in agro-environmental models, to determine land use suitability or for decision- and policy-making. It is therefore, important that the quality of a map is determined and quantified. This is achieved through (statistical) validation.

Validation is defined here as an activity in which the soil map predictions are compared with observed values. From this comparison, the map quality can be quantified and summarized using map quality measures. These measures indicate how accurate the map is on average for the mapping area, i.e. what is the expected error at a randomly selected location in the mapping area. This means that map quality measures obtained through validation are global measures: each quality

measure gives one value for the entire map. Note that this is different from results obtained through uncertainty assessment. Such assessment provides local, location-specific (i.e. for each individual grid cell) estimates of map quality as we saw in the previous Sections. Another important difference between validation and uncertainty assessment is that validation can be done using a model-free approach. Uncertainty assessment takes a model-based approach by defining a geostatistical model of the soil property of interest and deriving an interpolated map and the associated uncertainty from that, or by constructing a geostatistical model of the error in an existing map. The approach yields a complete probabilistic characterization of the map uncertainty, but such characterization is only valid under the assumptions made; for instance, the stationarity assumptions required for kriging. Validation, when done properly as explained hereafter, does not assume a geostatistical model of the error, and hence is model- or assumption-free. This is an important property of validation since we do not want to question the objectivity and validity of the validation results.

We distinguish internal and external map accuracy. Statistical methods typically produce direct estimates of map quality, for instance, the kriging variance or the coefficient of determination R^2 of a linear regression model. These we refer to as internal accuracy measures since these rely on model assumptions and are computed from data that are used for model calibration. Preferably, validation is done with an independent dataset not used in map making. Using such dataset gives the external map accuracy. One will often see that the external accuracy is poorer than the internal accuracy.

In Section 7.2 we will present the most common accuracy measures used to quantify map quality of quantitative (continuous) soil maps and qualitative (categorical) soil maps. In Section 7.4 we will introduce three commonly used validation methods and show how to estimate the map quality measures from a sample. This Chapter is largely based on Brus et al. (2011). For details, please refer to this paper.

7.2 Map quality measures

7.2.1 Quality measures for quantitative soil maps

All map quality measures considered here are computed from the prediction error. For quantitative soil maps of continuous soil properties (e.g. organic carbon content, pH, clay content) the prediction error is defined as the difference between the predicted value at a location and the true value at that location (which is the value that would be observed or measured by a preferably errorless measurement instrument) (Brus et al., 2011):

$$e(s) = \hat{Z}(s) - Z(s) \tag{7.1}$$

where $\hat{Z}(s)$ is the predicted soil property at validation location s, and $Z(s)$ is the true value of the soil property at that location. We consider six map quality

measures that are computed from the prediction error here: the mean error, the mean absolute error, the mean squared error and root mean squared error, the model efficiency, and the mean squared deviation ratio.

Before we introduce the map quality measures and show how to estimate these, it is important to understand the difference between the population and a sample taken from the population. The population is the set of all locations in a mapping area. For digital soil maps, this is the set of all pixels or grid cells of a map. A sample is a subset of locations, selected in some way from the set of all locations in the mapping area. With validation we want to assess the map accuracy for the entire population, i.e. for the map as a whole; we are not interested in the accuracy at the sample of locations only. For instance, we would like to know the prediction error averaged over all locations of a map and not merely the average prediction error at a sample of locations. Map quality measures are, therefore, defined as population means. Because we cannot afford to determine the prediction error at each location (grid cell) of the mapping area to calculate the population means, we have to take a sample of a limited number of locations in the mapping area. This sample is then used to estimate the population means. It is important to realize that we are uncertain about the population means because we estimate it from a sample. Ideally, this uncertainty is quantified and reported together with the estimated map quality measures.

In this Section, we will introduce the definitions of the map quality measures. In the next Section, we show how we can estimate these measures from a sample.

Mean error

The mean error (ME) measures bias in the predictions. The ME is defined as the population mean (spatial mean) of the prediction errors:

$$ME = e = \frac{1}{N}\sum_{i=1}^{N} e(s_i) \tag{7.2}$$

where i indicates the location, $i = 1, 2, \ldots N$, and N is the total number of locations or grid cells/pixels in the mapping area. The mean error should be (close to) zero, which means that predictions are unbiased meaning that there is no systematic over- or under-prediction of the soil property of interest.

Mean absolute error and (root) mean squared error

The mean absolute error (MAE) and mean squared error (MSE) are measures of map accuracy and indicate the magnitude of error we make on average. The MAE is defined by the population mean of the absolute errors:

$$MAE = |\underline{e}| = \frac{1}{N}\sum_{i=1}^{N} \underline{e}(s_i) \tag{7.3}$$

and the MSE by the population mean of the squared errors:

$$MSE = \underline{e}^2 = \frac{1}{N}\sum_{i=1}^{N}\underline{e}^2(s_i) \tag{7.4}$$

Many authors report the root mean squared error ($RMSE$) instead of the MSE, which is computed by taking the square root of the MSE. The $RMSE$ can be a more appealing quality measure since it has the same unit of measurement as the mapped property and can, therefore, more easily be compared to it. If the squared error distribution is strongly skewed, for instance when several very large errors are present, then this can severely inflate the $(R)MSE$. In such case, the (root) median squared error is a more robust statistic for the *average* error (Kempen et al., 2012).

Brus et al. (2011) argue that instead of using a single summary statistic (the mean) to quantify map quality measures, one should preferably express quality measures for quantitative soil maps through cumulative distribution functions (CDFs). Such functions provide a full description of the quality measures from which various parameters can be reported, such as the mean, median or percentiles. Furthermore, they argue that it can be of interest to define CDFs or its parameters for sub-areas, for instance, geomorphic units, soil or land cover classes. Brus et al. (2011) give examples of estimating CDFs for validation of digital soil maps.

Amount of variance explained

The model efficiency, or Amount of Variance Explained (AVE) (Angelini et al., 2016, Samuel-Rosa et al. (2015)), quantifies the fraction of the variation in the data that is explained by the prediction model. It measures the improvement of the model prediction over using the mean of the data set as predictor and is defined as follows (Krause et al., 2005):

$$AVE = 1 - \frac{\sum_{i=1}^{N}(\hat{Z}(s_i) - Z(s_i))^2}{\sum_{i=1}^{N}(Z(s_i) - \underline{Z})^2} \tag{7.5}$$

where $\underline{Z}$ is the population mean of soil property Z. The quantity in the numerator is the sum of the squared prediction errors (for each location the prediction error is computed and squared; the squared prediction errors are summed over all locations in the area). In linear regression, this quantity is known as the residual sum of squares (RSS). The quantity in the denominator is also a sum of squared prediction errors, but here the mean of the area is used as a predictor. In linear regression, this quantity is known as the total sum of squares (TSS). Note that if we would divide the quantity in the denominator by the number of locations in the mapping area N we would obtain the population variance (spatial variance) of the soil property Z.

If the numerator and denominator are equal, meaning the AVE is zero, then the model predictions are no improvement over using the mean of the data set as a predictor for any location in the mapping area. An AVE value larger than zero (RSS smaller than TSS) means that the model predictions are an improvement

over using the mean as a predictor (this is what we hope for). In case the AVE is negative, then the mean of the data set is a better predictor than the prediction model.

Mean squared deviation ratio

Finally, we introduce the mean squared deviation ratio ($MSDR$) as a map quality measure (Kempen et al., 2010; Lark, 2000; Voltz and Webster, 1990; Webster and Oliver, 2007). Contrary to the quality measures discussed so far, the $MSDR$ assesses how well the prediction model estimates the prediction uncertainty (expressed as the prediction error variance). The $MSDR$ is defined as:

$$MSDR = \frac{1}{N}\sum_{i=1}^{N}\frac{(\hat{Z}(s_i) - Z(s_i))^2}{\sigma^2(s_i)} \tag{7.6}$$

where $\sigma^2(s_i)$ is the prediction error variance at location $s_i, i = 1, 2, \ldots N$. The numerator is the squared error at location s_i. The fraction represents the squared Z_{score}. In case of kriging, the prediction error variance is the kriging variance. In case of linear regression, the prediction error variance is the prediction variance of the linear regression predictions that can be obtained by the statistical software R by running the predict function with argument $se.fit = TRUE$. This function returns for each prediction location the standard error of the predicted value as well as the residual standard deviation (the residual.scale value). By squaring both values and then summing these, the prediction error variance is obtained. If the prediction model estimates the error variance well, then the $MSDR$ should be close to one. A value smaller than one suggests that the prediction error variance overestimates the variance; a value larger than one suggests that the prediction error variance underestimates the variance.

Lark (2000) notes that outliers in the prediction data will influence the squared Z_{score} and suggests to use the median squared Z_{score} instead of the mean since it is a more robust estimator. A median squared Z_{score} equal to 0.455 suggests that the prediction model estimates the prediction uncertainty well.

7.2.2 Quality measures for qualitative soil maps

Like the quality measures for quantitative soil maps, the quality measures for qualitative or categorical soil maps (e.g. soil classes) are defined for the population, i.e. all locations in the mapping area. The basis for map quality assessment of qualitative maps is the error matrix (Brus et al., 2011; Lark, 1995). This matrix is constructed by tabulating the observed and predicted class for all locations in the mapping area in a two-way contingency table. The population error matrix is a square matrix of order U, with U being the number of soil classes observed and mapped. The columns of the matrix correspond to observed soil classes and the rows of predicted soil classes (the map units).N is the total number of locations of the mapping area. Elements N_{ij} are the number of locations mapped as class i with observed class j. The row margins N_{i+} are the locations mapped as class i,

and column margins N_{+j} the locations for which the observed soil class is j. Note that the elements of the population error matrix can also be interpreted as surface areas. In that case element N_{ij} is the surface area mapped as class i with observed class j.

From the population error matrix, several quality measures can be summarized, though it is strongly recommended that the error matrix is included in a validation assessment. Brus et al. (2011) follow the suggestion by Stehman (1997) that quality measures for categorical maps should be directly interpretable in terms of the probability of a misclassification and therefore recommend the use of three map quality measures: the overall purity, the map unit purity, and class representation. We follow this recommendation here. Note that the map unit purity often is referred to as user's accuracy, and class representation as producer's accuracy (Stehman, 1997; Adhikari et al., 2014). Lark (1995) however, questions the appropriateness of these terms since both quality measures can be important for users as well as producers. He proposes to use map unit purity and class representation instead, which is adopted by Brus et al. (2011) and followed here.

A fourth frequently used group of quality measures are Kappa indices, which adjust the overall purity measure for hypothetical chance agreement (Stehman, 1997). How this chance agreement is defined differs between the various indices. Some authors, however, conclude that Kappa indices are difficult to interpret, not informative, misleading and/or flawed and suggest to abandon their use (Pontius Jr and Millones, 2011). These authors argue that Kappa indices attempt to compare accuracy to a baseline of randomness, but randomness is not a reasonable alternative for map construction. We, therefore, do not consider kappa here.

The overall purity is the fraction of locations for which the mapped soil class equals the observed soil class and is defined as (Brus et al., 2011):

$$\rho = \sum_{i=1}^{U} N_{uu}/N \tag{7.7}$$

which is the sum of the principal diagonal of the error matrix divided by the total number of locations in the mapping area. The overall purity can be interpreted as the areal proportion of the mapping area that is correctly classified.

Alternatively, an indicator approach can be used to compute the overall purity. A validation site gets a **1** if the observed soil class is correctly predicted and a **0** otherwise. The overall purity is then computed by taking the average of the indicators.

Map unit purity

The map unit purity is calculated from the row marginals of the error matrix. It is the fraction of validation locations with mapped class u for which the observed class is also u. The map unit purity for class u is defined as (Brus et al., 2011):

$$\rho_u = \frac{N_{uu}}{N_{u+}} \tag{7.8}$$

The map unit purity can be interpreted as the proportion of the area of the map unit that is correctly classified. The complement of ρ_u, $1 - \rho_u$, is referred to as the error of commission for mapped class u.

Class representation

The class representation is calculated from the column marginals of the error matrix. It is the fraction of validation locations with observed class u for which the mapped class is u. The class representation for class u is defined as (Brus et al., 2011):

$$r_u = \frac{N_{uu}}{N_{+u}} \tag{7.9}$$

The class representation can be interpreted as the proportion of the area where in reality class u occurs that is also mapped as class u. The complement of r_u, $1-r_u$, is referred to as the error of omission for mapped class u.

7.2.3 Estimating the map quality measures and associated uncertainty

In validation, we estimate the population means of the map quality measures from a sample taken from a limited number of locations in the mapping area. After all, we cannot afford to sample all locations, i.e. each grid cell of our soil map. Because the map quality measures are estimates, we are uncertain about these: we infer the quality measures from only a limited number of observations taken from the population. We do not know the true population means. The estimation uncertainty can be quantified with the sampling variance. From the variance, the lower and upper boundary of a confidence interval, typically the 95%, can be computed using basic statistical theory:

$$CI = (\underline{\hat{x}} - 1.96x\frac{\sigma}{\sqrt{n}}; \underline{\hat{x}} + 1.96x\frac{\sigma}{\sqrt{n}}) \tag{7.10}$$

where x is the estimated map quality measure, for instance, the ME, MSE or overall purity, σ is the estimated standard deviation of the map quality measure and n is the validation sample size.

Quantified information about the uncertainty associated to map quality measures is useful and required for statistical testing. For instance, if one wants to test if one mapping method performs better than the other method one needs quantified information about uncertainty. Because we are uncertain about the estimated quality measures, an observed difference in map quality between two methods does not necessarily mean that one method is better than the others, even when

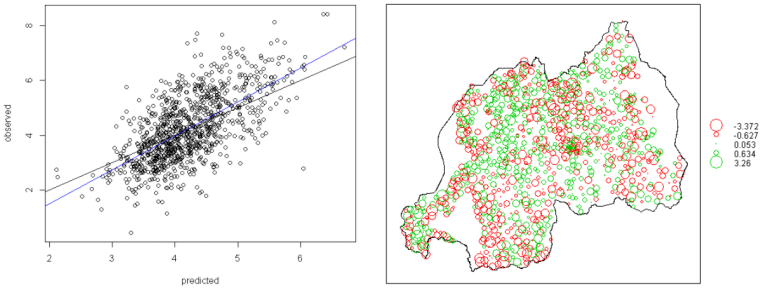


Figure 7.1: Scatterplot of predicted versus observed SOM content for Rwanda (left) and spatial bubble plot of cross-validation error for SOM (right) (Kempen et al., 2015). The black line in the scatter plot represents the 1:1 line of prediction versus observed, the blue line represents the regression between observed and predicted values

there is a substantial difference. The difference might be attributed to chance because we infer the quality measures from a limited sample from the population. With statistical hypothesis testing, we can calculate how large the probability is that observed difference is caused by chance. Based on the outcome we can accept or reject the hypothesis that there is no difference between the performance of two mapping methods (this would be the null hypothesis for statistical testing) for a given significance level, usually 0.05.

7.3 Graphical map quality measures

In addition to quantifying map accuracy statistically, one can also present validation results obtained from a sample graphically. This can be done by creating scatter plots of predicted against observed values and spatial bubble plots of validation errors.

Figure 7.1 shows an example of a scatterplot and bubble plot. Both plots can be easily made with **R** (R Core Team, 2017). Use the function `plot(x,y)` to generate a scatter plot. The 1:1 line (black line in Figure 7.1) can be added to the plot with the command `abline(0, 1)`. The spatial bubble plot can be generated with the `bubble()` function of the **sp** package (Pebesma and Bivand, 2005).

7.4 Validation methods and statistical inference

Following Brus et al. (2011), we introduce and discuss three common validation methods: additional probability sampling, data-splitting and cross-validation, and show how to estimate the map quality measures introduced in the previous Section from a sample.

With additional probability sampling, an independent dataset is collected from the sampling population (all grid cells of a digital soil map) for the purpose of validation. This dataset is used in addition to a dataset that is used to calibrate a prediction model. Such dataset is often a legacy dataset collected with a purposive sampling design.

Data-splitting and cross-validation are applied in situations where one has only one data set available for prediction model calibration and validation. This can be a dataset collected with probability sampling, but in practice this typically is a legacy dataset collected with some purposive sampling design.

We warn here that if one uses data-splitting or cross-validation with a dataset collected with purposive sampling, then this has severe implications on the validity and interpretation of the estimated map quality measures as we will explain below

7.4.1 Additional probability sampling

The most appropriate approach for validation is by additional probability sampling. This means that an independent validation dataset is collected in the field on basis of a probability sampling design. Validation based on probability sampling ensures one obtains unbiased and valid estimates of the map quality measures (Brus et al., 2011; Stehman, 1999).

Additional probability sampling has several **advantages** compared to data-splitting and cross-validation using non-probability sample data. These are:

- No model is needed for estimating map quality estimates. We can apply design-based estimation, meaning that model-free unbiased and valid estimates of the map quality measures can be obtained;
- Discussions on the validity of the estimated map quality are avoided;
- Model-free, valid estimates of the variance of the map quality measures can be obtained that allows for hypothesis testing, e.g. for comparison of model performance.

Disadvantages can be extra costs involved in collecting an additional sample or terrain conditions that make it difficult to access all locations in the mapping area.

Probability sampling is random sampling such that:

- All locations in the mapping area have a probability larger than 0 of being selected;
- The inclusion probabilities are known but need not be equal.

It should be noted that random sampling is often used for arbitrary or haphazard sampling. Such sampling is not probability sampling because the inclusion probabilities are not known. Design-based, model-free estimation of map quality measures is not possible in this case. All probability samples are random samples

but not all random samples are probability samples. The term probability sampling should therefore only be used for random sampling with known inclusion probabilities.

There are many different probability sampling designs: simple, stratified, systematic, two-stage, clustered random sampling. We will not give an exhaustive overview here of all these designs. A good resource is De Gruijter et al. (2006). For reasons of simplicity, we focus here on simple random sampling.

Simple random sampling: In simple random sampling, no restrictions are imposed on the random selection of sampling sites except that the sample size is fixed and chosen prior to sampling (De Gruijter et al., 2006). All sampling locations are selected with equal probability and independently from each other.

This can, for instance, be done as follows (De Gruijter et al., 2006):

1. Determine the minimum and maximum X and Y coordinates of the mapping area (the bounding box).
2. Generate two independent random coordinates X and Y from a uniform probability distribution on the interval (x_{min}, x_{max}) and (y_{min}, y_{max})
3. Check if the selected sampling site falls within the mapping area. Accept the sampling site if it does; discard the sampling site if it does not.
4. Repeat steps 2 and 3 until the n locations have been selected.

If a sampling location cannot be visited because of inaccessibility for instance, then this location should be discarded and be replaced by a location chosen from a reserve list. Always the location at the top of the list should be selected for this purpose; not an arbitrarily chosen location from the list such as the closest one. It is not allowed to shift an inaccessible sampling location to a location nearby that can be accessed. Irregularity, clustering and open spaces characterize the simple random sampling design (De Gruijter et al., 2006).

Estimation of quantitative map quality measures: For each validation location we compute the error, $e(s_i)$, the absolute error, $|e|(s_i)$, or squared error, $e^2(s_i)$. The spatial mean of the mapping area for map quality measure x is then estimated by:

$$e(s_i) = \hat{Z}(s_i) - Z(s_i) \tag{7.11}$$

$$|e|(s_i) = |\hat{Z}(s_i) - Z(s_i)| \tag{7.12}$$

$$e^2(s_i) = (\hat{Z}(s_i) - Z(s_i))^2 \tag{7.13}$$

$$\hat{\underline{x}} = \frac{1}{N}\sum_{i=1}^{N} x(s_i) \tag{7.14}$$

where i indicates the validation location, $i = 1, 2, \ldots, n$, n the validation sample size, and x_{si} the estimated population mean of map quality measure x at location s_i. x is the prediction error in case of the ME, absolute error in case of the MAE, squared prediction error in case of the MSE. Note that the estimator is the unweighted sample mean. This unweighted mean is an unbiased estimator because all sampling locations were selected with equal probability.

The $MSDR$ is estimated by:

$$\widehat{MSDR} = \frac{1}{N} \sum_{i=1}^{n} \frac{(\hat{Z}(s_i) - Z(s_i))^2}{\sigma^2(s_i)} \tag{7.15}$$

and the AVE by:

$$\widehat{AVE} = 1 - \frac{\sum_{i=1}^{n} (\hat{Z}(s_i) - Z(s_i))^2}{\sum_{i=1}^{n} (Z(s_i) - \underline{\hat{Z}})^2} \tag{7.16}$$

where $\underline{\hat{Z}}$ is the mean of the target soil property estimated from the validation sample.

One should be careful when assessing the proportion of variance explained by computing the R^2 from a linear regression of the predicted value on the observed value (Krause et al., 2005), as is often done in practice. The R^2 quantifies the dispersion around the regression line; not around the 1:1 line in which we are interested in validation. So it does not directly compare the predicted with observed value as does the AVE; i.e. it is not based on the prediction error. A high R^2-value, therefore, does not automatically mean a high AVE. For instance, in case of strongly biased predictions the R^2 can be high but the AVE will be low. The blue line in Figure 7.1 is the regression line that one obtains when regression the observed value of the predicted value. This line slightly differs from the 1:1 line. In this example, the R^2 of the regression is 0.42 while the AVE is 0.40. The uncertainty associated to the estimated map quality measures is quantified with the sampling variance, which for the ME, MAE and MSE is estimated by:

$$Var(\underline{\hat{x}}) = \frac{1}{n(n-1)} \sum_{i=1}^{n} (x(s_i) - \underline{\hat{x}}) \tag{7.17}$$

and the 95% confidence interval (CI) of x is given by:

$$CI_{95} = \underline{\hat{x}} \pm 1.96x\sqrt{Var(\underline{\hat{x}})} \tag{7.18}$$

We should warn here that the calculation of the CI is based on the assumption that the estimated map quality measure means have a normal distribution (the central limit theorem). For the squared errors this assumption can be unrealistic, especially for small sample sizes.

		Observed					
		1	2	.	.	U	Σ
Mapped	1	n11	n12	.	.	n1U	n1+
	2	n21	n22	.	.	n2U	n2+
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	U	nU1	nU2	.	.	nUU	nU+
	Σ	n+1	n+2	0	0	n+U	n

Figure 7.2: Sample error matrix

Estimation of qualitative map quality measures: For validation of qualitative soil maps, a sample error matrix is constructed from the validation data (Figure 7.2). n is the total number of validation locations in the sample. Element n_{ij} of the matrix corresponds to the number of validation locations that have been predicted as class i, $i = 1, 2, \ldots U$ and belong to class j, $j = 1, 2, \ldots U$ (Lark, 1995). The matrix summarizes correct predictions and incorrect predictions within the validation data.

From the sample error matrix the overall purity, map unit purity and class representation are estimated by:

$$\hat{\rho} = \sum_{i=1}^{U} n_{uu}/n \tag{7.19}$$

$$\hat{\rho}_u = \frac{n_{uu}}{n_{u+}} \tag{7.20}$$

$$\hat{r_u} = \frac{n_{uu}}{n_{+u}} \tag{7.21}$$

Alternatively, the overall purity can be estimated by defining a purity indicator variable for each validation location that takes value **1** if the mapped soil class equals the observed soil class at that location, and **0** else. The overall purity is then estimated by:

$$\hat{\rho} = \frac{1}{n} \sum_{i=1}^{n} \partial(s_i) \tag{7.22}$$

where $\delta(s_i)$ is the indicator variable at validation location s_i. The variance of the estimated overall purity is estimated by:

$$Var(\hat{\rho}) = \frac{1}{n(n-1)} \sum_{i=1}^{n} (\partial(s_i) - \hat{\rho})^2 \tag{7.23}$$

Alternatively, the variance is estimated by:

		Observed					
		Anthrosol	Cambisol	Gleysol	Luvisol	Podzol	Σ
Mapped	Anthrosol	19	5	3	0	1	28
	Cambisol	5	33	9	13	5	65
	Gleysol	2	8	25	3	5	43
	Luvisol	3	15	9	42	2	71
	Podzol	1	3	8	2	19	33
	Σ	30	64	54	60	32	240

Figure 7.3: Sample error matrix for a hypothetical soil class map

$$Var(\hat{\rho}) = \frac{\hat{\rho}(1-\hat{\rho})}{n-1)} \tag{7.24}$$

which is the variance of a binomial probability distribution. The 95% confidence interval of $\hat{\rho}$ is given by:

$$CI_{95} = \hat{\rho} \pm 1.96x\sqrt{Var(\hat{\rho})} \tag{7.25}$$

We warn that the CI as calculated here is a rough approximation which only holds when $n \times \hat{\rho}$ and $n \times (1-\hat{\rho})$ are large (5 as a rule of thumb). Otherwise, the binomial distribution should be used to compute the CI. Figure 7.3 shows a hypothetical example of a sample error matrix for soil class map. For this example, the overall purity is estimated by:

$$\hat{\rho} = \frac{(19+33+25+42+19)}{240} = 0.575 \tag{7.26}$$

meaning that for an estimated 57.5% of the mapping area the mapped soil class is equal to the true soil class.

Table 7.1 gives the map unit purities and class representations for this example. The map unit purity of the Gleysol is 0.581, meaning that at 58.1% of the validation locations for which a Gleysol is predicted, a Gleysol is observed. Assuming the validation data were collected by simple random sampling, we could conclude that for 58.1% of the area mapped as Gleysol we would find a Gleysol in the field. The class representation of the Gleysol is 0.463, meaning that for 46.3% of the validation locations classified as Gleysol, we map a Gleysol. The majority of the Gleysol locations is thus mapped as a different soil class. Again, assuming the validation data were collected by probability sampling, we would estimate that 22.5% ($\frac{54}{240} \times 100\%$) of our mapping area is covered by Gleysols. We map Gleysols for 17.9% of the area ($\frac{43}{240} \times 100\%$). It can happen that a soil class has a high map unit purity and a low-class representation. This means that if we map a Gleysol we will likely find a Gleysol there, but that a large extent of the true Gleysol area is not mapped as such.

Table 7.1: Map unit purity and class representation statistics for an hypothetical example

unit	map.unit.purity	class.representation
Anthrosol	0.679	0.633
Cambisol	0.508	0.516
Gleysol	0.508	0.463
Luvisol	0.592	0.700
Podzol	0.576	0.594

7.4.2 Data-splitting

In data-splitting, the sample data set is split into two subsets. One subset is used to calibrate the prediction model. The other subset is used for validation. A frequently used splitting criterion is 70 - 30, where 70% of the sample data are used for calibration and 30% for validation. The choice of a splitting criterion, however, is arbitrary and it is not evident how to split a data set in such a way that unbiased and valid estimates of the map accuracy can be obtained. For sparse data sets, data-splitting can be inefficient since the information in the data set is not fully exploited for both calibration and validation.

It is important to note here that a random subsample of (legacy) data that are collected with a purposive (non-probability) design, is not a probability sample of the study area. This means that design-based estimation of map quality measures is not possible.

Thus, we will not obtain model-free, unbiased and valid estimates of the quality measures from non-probability sample validation data. In a case study, Knotters and Brus (2013) showed that model-based predictions of producer's accuracies from two models differed strongly, indicating that with the model-based approach the validation results strongly depend on model assumptions.

In most studies, however, spatial correlation is not accounted for when estimating map quality measures using the estimators presented above in Section 7.4.1 as simple random sampling from non-probability sample data. In such case, the quality measures cannot be considered unbiased and valid estimates of the population means of the map quality measures. In addition, the estimated variance of the map quality measures is not valid and statistical testing of mapping methods to assess which method gives the most accurate predictions cannot be done.

In other words, if the simple random sampling estimators are used to estimate map quality measures then these are only valid for the validation data points. The map quality measures do not give a valid estimate of the quality of the map as a whole (the population). For instance, the overall purity cannot be interpreted as an areal proportion of correctly mapped soil classes, only as the proportion of the validation data points for which the soil class is correctly predicted. 70 - 30, where 70% of the sample data are used for calibration and 30% for validation.

The choice of a splitting criterion, however, is arbitrary and it is not evident how to split a data set in such a way that unbiased and valid estimates of the map accuracy can be obtained. For sparse data sets, data-splitting can be inefficient since the information in the data set is not fully exploited for both calibration and validation.

It is important to note here that a random subsample of (legacy) data that are collected with a purposive (non-probability) design, is not a probability sample of the study area. This means that design-based estimation of map quality measures is not possible.

If a validation (sub)sample is a non-probability sample of the mapping area, then we must account for possible spatial autocorrelation of the prediction errors when estimating the map quality measures. One can imagine that when two validation locations are close together and the prediction errors are correlated that there is less information in these two locations (there is information redundancy because of autocorrelation) than in two isolated locations. This information redundancy has to be accounted for when estimating map quality measures and implies that we have to rely on model-based estimation: a model for the spatially autocorrelated prediction error has to be assumed. Thus, we will not obtain model-free, unbiased and valid estimates of the quality measures from non-probability sample validation data. In a case study, Knotters and Brus (2013) showed that model-based predictions of producer's accuracies from two models differed strongly, indicating that with the model-based approach the validation results strongly depend on model assumptions.

7.4.3 Cross-validation

In K-fold cross-validation, the dataset is split into K roughly equal sets. One of these sets is set aside for validation. The model is then calibrated using the data from the K-1 sets and used to predict the target variable for the data points set aside. From this prediction, the prediction error is calculated. This procedure is repeated K times, each time setting a different set aside for validation. In this way, we obtain K estimates of the prediction error: one for each validation sample site. In this way, all data are used for validation and model calibration. It is thus much more efficient than data-splitting.

K is typically chosen as 5 or 10 or as N the number of data points. The latter is referred to as leave-one-out cross-validation in which only one validation site is set aside in each iteration. The model is then calibrated with N-1 observations. Some repeat K-fold cross-validation a number of times and average the results to obtain a more robust estimate of the map quality measures.

Note that the problem of spatially correlated errors remains when data are non-probability sample data. Cross-validation using a non-probability sampling dataset suffers from the same drawbacks with respect to unbiasedness and validity of the estimates of the map quality measures as data-splitting. The estimates cannot

be interpreted as being valid for the mapping area, but only for the validation locations.

In **R**, the **caret** package (Kuhn, 2016) offers functionality for data-splitting and cross-validation.

7.5 Technical steps - Validation

G.F. Olmedo

Step 1 - Prediction error

First, we will load the validation dataset. This dataset was measured at the same time as the modeling dataset. After data preparation in Chapter 4, we used the code in Section 7.6 to split the database: 75% of the points were used in the models from Chapter 6 and now we will use the remaining 25% of the points to test those results.

```
library(sp)

dat <- read.csv("data/dat_test.csv")

# Promote to spatialPointsDataFrame
coordinates(dat) <- ~ X + Y

dat@proj4string <- CRS(projargs = "+init=epsg:4326")
```

Now, we will load the predicted layers from Chapter 6, extract the predicted value for every point and then, estimate the prediction error.

```
library(raster)

OCSKGM_GM <- raster("results/MKD_OCSKGM_GM.tif")
OCSKGM_RK <- raster("results/MKD_OCSKGM_RK.tif")
OCSKGM_rf <- raster("results/MKD_OCSKGM_rf.tif")
OCSKGM_svm <- raster("results/MKD_OCSKGM_svm.tif")

dat <- extract(x = OCSKGM_GM, y = dat, sp = TRUE)
dat <- extract(x = OCSKGM_RK, y = dat, sp = TRUE)
dat <- extract(x = OCSKGM_rf, y = dat, sp = TRUE)
dat <- extract(x = OCSKGM_svm, y = dat, sp = TRUE)

dat$PE_GM <- dat$MKD_OCSKGM_GM - dat$OCSKGM
dat$PE_RK <- dat$MKD_OCSKGM_RK - dat$OCSKGM
dat$PE_rf <- dat$MKD_OCSKGM_rf - dat$OCSKGM
dat$PE_svm <- dat$MKD_OCSKGM_svm - dat$OCSKGM
```

Table 7.2: Summary of prediction errors for three different mapping methods

	GM	RK	randomForest	svm
Min.	-4.03	-4.09	-4.25	-4.75
1st Qu.	-0.24	-0.31	-0.29	-0.26
Median	0.00	0.00	0.02	0.04
Mean	0.01	-0.08	-0.08	-0.03
3rd Qu.	0.29	0.23	0.24	0.33
Max.	2.81	2.01	2.22	1.72
NA's	40.00	10.00	12.00	12.00

```
# Save the validation results
write.csv(dat, "results/validation.csv", row.names = F)
```

Step 2 - Estimating the map quality measures

In this Section, we present the code needed to estimate the map quality measures for quantitative soil maps.

```
# Mean Error
ME_GM <- mean(dat$PE_GM, na.rm=TRUE)
ME_RK <- mean(dat$PE_RK, na.rm=TRUE)
ME_rf <- mean(dat$PE_rf, na.rm=TRUE)
ME_svm <- mean(dat$PE_svm, na.rm=TRUE)

# Mean Absolute Error (MAE)
MAE_GM <- mean(abs(dat$PE_GM), na.rm=TRUE)
MAE_RK <- mean(abs(dat$PE_RK), na.rm=TRUE)
MAE_rf <- mean(abs(dat$PE_rf), na.rm=TRUE)
MAE_svm <- mean(abs(dat$PE_svm), na.rm=TRUE)

# Mean Squared Error (MSE)
MSE_GM <- mean(dat$PE_GM^2, na.rm=TRUE)
MSE_RK <- mean(dat$PE_RK^2, na.rm=TRUE)
MSE_rf <- mean(dat$PE_rf^2, na.rm=TRUE)
MSE_svm <- mean(dat$PE_svm^2, na.rm=TRUE)

# Root Mean Squared Error (RMSE)
RMSE_GM <- sqrt(sum(dat$PE_GM^2, na.rm=TRUE) / length(dat$PE_GM))
RMSE_RK <- sqrt(sum(dat$PE_RK^2, na.rm=TRUE) / length(dat$PE_RK))
RMSE_rf <- sqrt(sum(dat$PE_rf^2, na.rm=TRUE) / length(dat$PE_rf))
RMSE_svm <- sqrt(sum(dat$PE_svm^2, na.rm=TRUE) / length(dat$PE_svm))

# Amount of Variance Explained (AVE)
AVE_GM <- 1 - sum(dat$PE_GM^2, na.rm=TRUE) /
```

Table 7.3: Summary of map quality measures for three different mapping methods

	GM	RK	randomForest	svm
ME	0.014	-0.077	-0.081	-0.026
MAE	0.409	0.359	0.368	0.391
MSE	0.395	0.293	0.312	0.332
RMSE	0.615	0.538	0.555	0.573
AVE	-0.033	-1.054	-1.572	-1.360

```
  sum( (dat$OCSKGM - mean(dat$OCSKGM, na.rm = TRUE))^2,
       na.rm = TRUE)

AVE_RK <- 1 - sum(dat$PE_RK^2, na.rm=TRUE) /
  sum( (dat$OCSKGM - mean(dat$OCSKGM, na.rm = TRUE))^2,
       na.rm = TRUE)

AVE_rf <- 1 - sum(dat$PE_rf^2, na.rm=TRUE) /
  sum( (dat$OCSKGM - mean(dat$OCSKGM, na.rm = TRUE))^2,
       na.rm = TRUE)

AVE_svm <- 1 - sum(dat$PE_svm^2, na.rm=TRUE) /
  sum( (dat$OCSKGM - mean(dat$OCSKGM, na.rm = TRUE))^2,
       na.rm = TRUE)
```

Step 3 - Graphical map quality measures

In this Section, we will apply the proposed graphical quality measures to the results of Chapter 6.

```
# Two plots in one plot
par(mfrow=c(2,2))

# Scatter plot
plot(dat$MKD_OCSKGM_GM, dat$OCSKGM, main="GM", xlab="predicted",
     ylab='observed')

# 1:1 line in black
abline(0,1, lty=2, col='black')

# Regression line between predicted and observed in blue
abline(lm(dat$OCSKGM ~ dat$MKD_OCSKGM_GM), col = 'blue', lty=2)

# Scatter plot
plot(dat$MKD_OCSKGM_RK, dat$OCSKGM, main="RK", xlab="predicted",
     ylab='observed')
```

```
# 1:1 line in black
abline(0,1, lty=2, col='black')

# Regression line between predicted and observed in blue
abline(lm(dat$OCSKGM ~ dat$MKD_OCSKGM_RK), col = 'blue', lty=2)

# Scatter plot
plot(dat$MKD_OCSKGM_rf, dat$OCSKGM, main="rf", xlab="predicted",
     ylab='observed')

# 1:1 line in black
abline(0,1, lty=2, col='black')

# Regression line between predicted and observed in blue
abline(lm(dat$OCSKGM ~ dat$MKD_OCSKGM_rf), col = 'blue', lty=2)

# Scatter plot
plot(dat$MKD_OCSKGM_svm, dat$OCSKGM, main="svm", xlab="predicted",
     ylab='observed')

# 1:1 line in black
abline(0,1, lty=2, col='black')

# Regression line between predicted and observed in blue
abline(lm(dat$OCSKGM ~ dat$MKD_OCSKGM_svm), col = 'blue', lty=2)
```

```
par(mfrow=c(1,1))
```

```
# Spatial bubbles for prediction errors
bubble(dat[!is.na(dat$PE_GM),], "PE_GM", pch = 21,
       col=c('red', 'green'))
```

```
# Spatial bubbles for prediction errors
bubble(dat[!is.na(dat$PE_RK),], "PE_RK", pch = 21,
       col=c('red', 'green'))
```

```
# Spatial bubbles for prediction errors
bubble(dat[!is.na(dat$PE_rf),], "PE_rf", pch = 21,
       col=c('red', 'green'))
```

```
# Spatial bubbles for prediction errors
bubble(dat[!is.na(dat$PE_svm),], "PE_svm", pch = 21,
       col=c('red', 'green'))
```

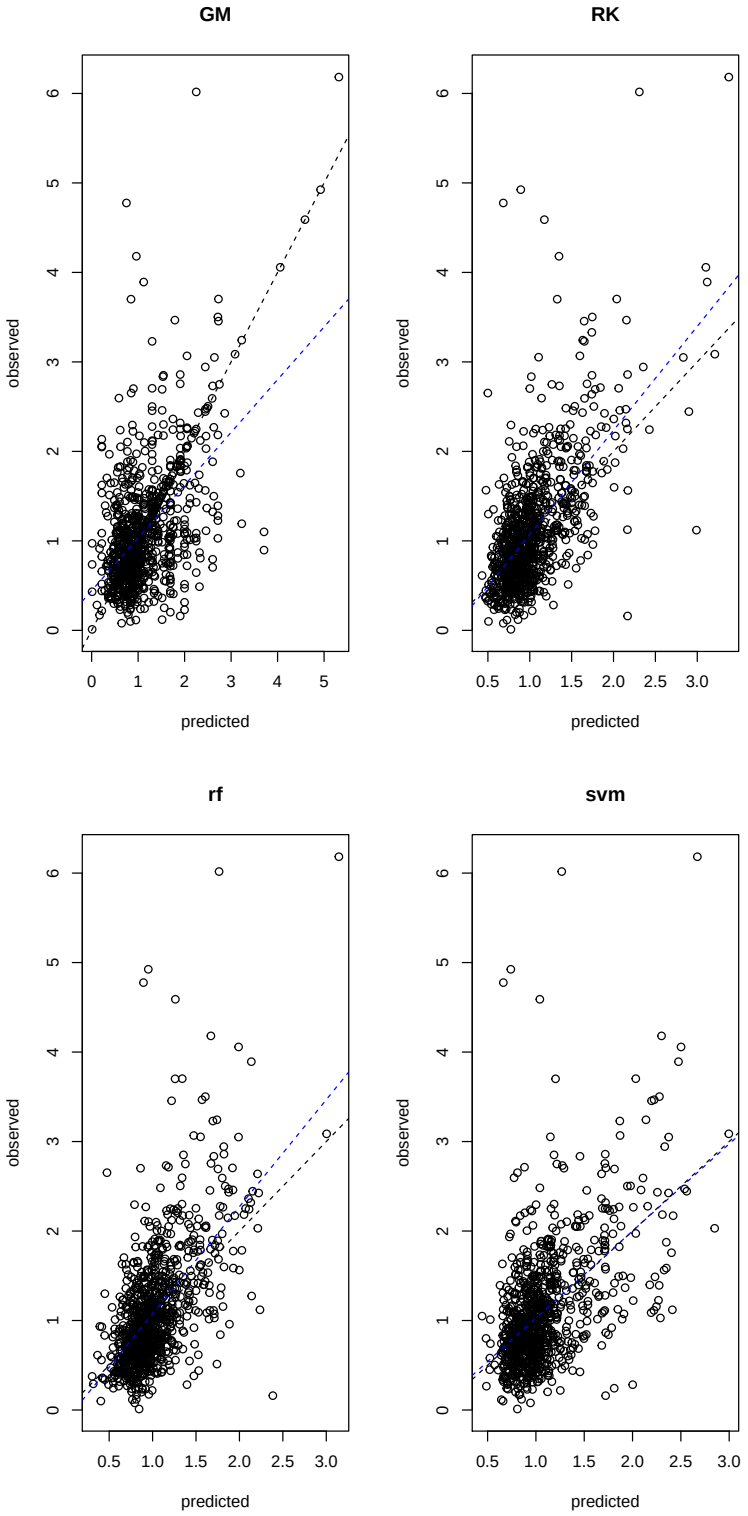


Figure 7.4: Scatter plots of predicted against observed values

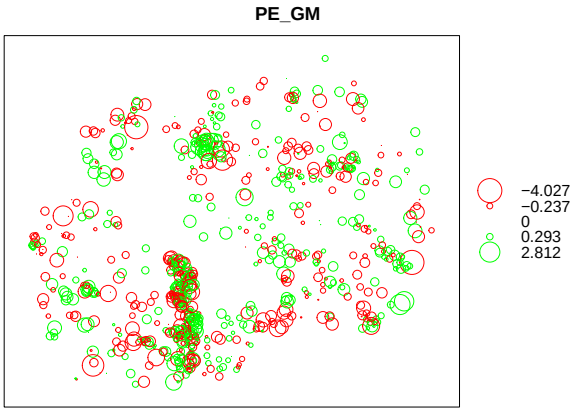


Figure 7.5: Spatial bubble of the prediction errors for GM

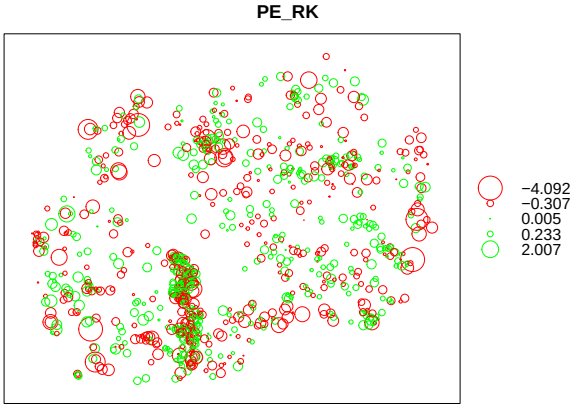


Figure 7.6: Spatial bubble of the prediction errors for RK

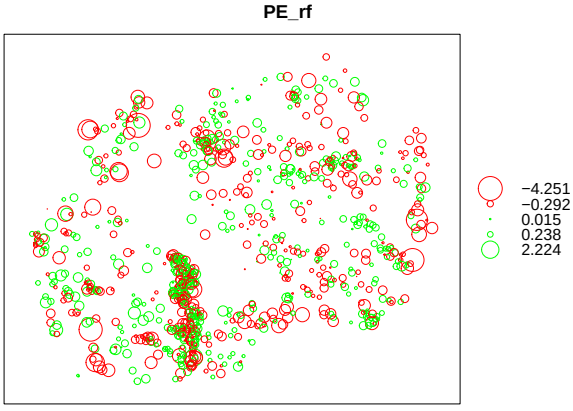


Figure 7.7: Spatial bubble of the prediction errors for RF

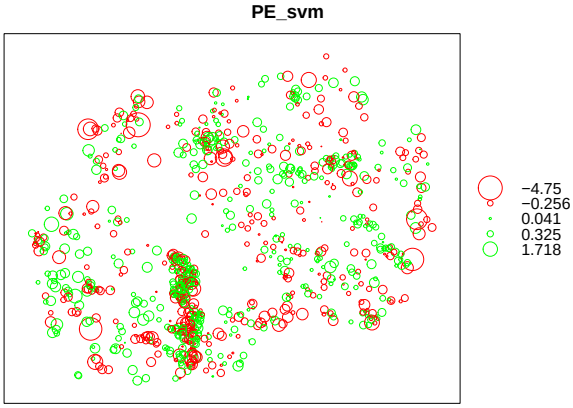


Figure 7.8: Spatial bubble of the prediction errors for SVM

7.6 Technical steps - Data-splitting

As explained before, many times for running validation analysis, we split the data before fitting the models. In this Section, we present the code needed to achieve this using **caret** package.

```
library(caret)

dat <- read.csv("data/dataproc.csv")

train.ind <- createDataPartition(1:nrow(dat), p = .75, list = FALSE)
train <- dat[ train.ind,]
test  <- dat[-train.ind,]

plot(density (log(train$OCSKGM)), col='red', main="")
lines(density(log(test$OCSKGM)), col='blue')
legend('topright', legend=c("train", "test"),
       col=c("red", "blue"), lty=1, cex=1.5)
```

```
write.csv(train, file="data/dat_train.csv", row.names = FALSE)
write.csv(test, file="data/dat_test.csv", row.names = FALSE)
```

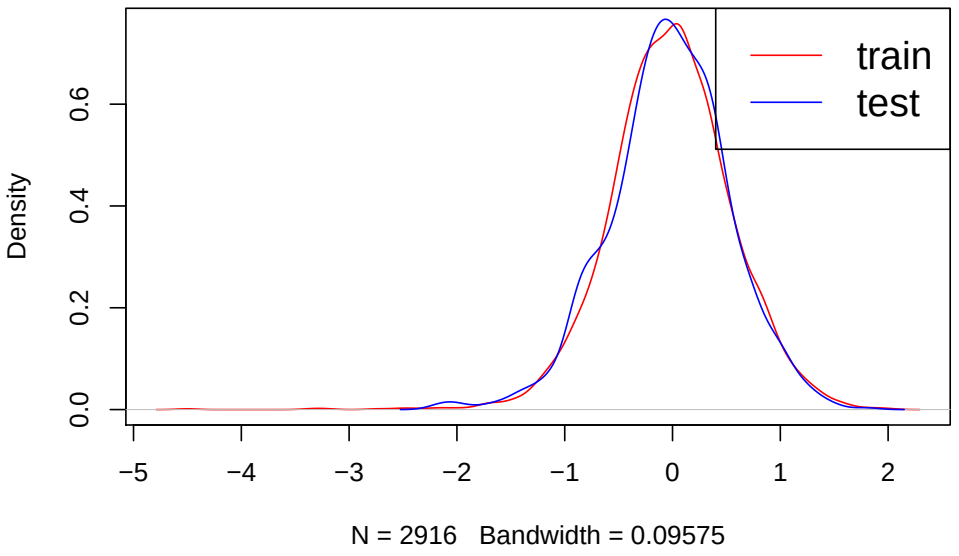


Figure 7.9: Statistical distribution of train and test datasets

Chapter 8

Model evaluation in digital soil mapping

M. Guevara & G.F. Olmedo

There are no best methods for statistical modeling and different evaluation strategies should be considered in order to identify, realistically, the overall modeling accuracy (Ho and Pepyne, 2002; Qiao et al., 2015; Guevara et al., 2018; Nussbaum et al., 2018). This Section is devoted to describe quantitative methods for model evaluation applied to SOC mapping across FYROM.

Our objective is to provide a model evaluation example based on a vector of observed SOC and a vector of modeled SOC estimates derived from three different statistical methods describes in Chapter 6: multiple linear regression-kriging (RK) (see Section 6.2), random forests (RF) (see Section 6.3), and support vector machines (SVM) (see Section 6.4)). The model evaluation methods presented here were adapted from the original work of Carslaw and Ropkins (2012) for air quality assessments and their **R** package **openair**.

We found in this package a very useful set of functions for model evaluation metrics that are suitable (and we highly recommend) for comparing digital soil maps derived from different prediction algorithms. We will first analyze the simple correlation and major differences of generated SOC maps by the three different methods. Then for further analysis we will prepare a data frame containing the observed and modeled vectors as well as the method column. Ideally the observed vector should be derived from a completely independent SOC dataset, as explained in the previous Chapter. The cross-validation strategy and the repeated random split for training and testing the models are other two alternatives when no independent dataset is available for validation purposes. However, we do not recommend to use the same training dataset for performing the following analysis, since the resulting *best method* could be the one that overfits the most.

The authors of this Chapter used **R** packages. To run the code provided in this

Chapter, the following packages need to be installed in the **R** user library. If the packages are not yet installed, the `install.packages()` function can be used.

```
# Installing packages for Chapter 'Model Evaluation in
# Digital Soil Mapping'
install.packages(c("raster", "psych", "rasterVis",
                   "mapview", "openair"))
```

8.1 Technical steps - Model correlations and spatial differences

Step 1 - Harmonize the predicted maps to the same spatial resolution

We will import the predicted maps and harmonize them in to the same regular grid (1×1 km of spatial resolution).

```
library(raster)

GM<-raster('results/MKD_OCSKGM_GM.tif')
RK<-raster('results/MKD_OCSKGM_RK.tif')
RF<-raster('results/MKD_OCSKGM_rf.tif')
SVM<-raster('results/MKD_OCSKGM_svm.tif')

# Note that RK has a different reference system
RK <- projectRaster(RK, SVM)
models <- stack(GM, RK, RF, SVM)
```

Step 2 - Compare the statistical distribution and correlation between the models

Then we will plot the statistical distribution and the correlation between the three different methods (RK, RF, SVM).

```
library(psych)

pairs.panels(na.omit(as.data.frame(models)),
             # Correlation method
             method = "pearson",
             hist.col = "#00AFBB",
             # Show density plots
             density = TRUE,
             # Show correlation ellipses
             ellipses = TRUE
             )
```

Here we found that the higher Pearson correlation coefficient (r value) between predicted values was between RK and SVM (0.86). We also found that the statistical distribution of predicted values is quite similar between the three methods

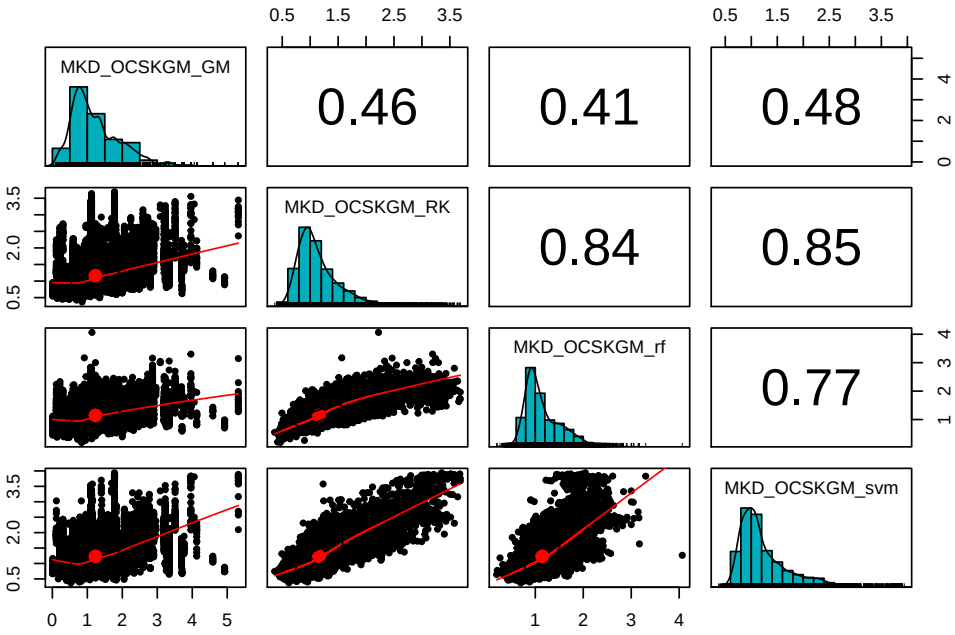


Figure 8.1: Comparision of DSM model correlations (GM, RK, RF, SVM) and statical distributions

and that the lowest correlation of predictions was found between the two machine learning approaches with r 0.79 for RF and SVM.

We can in addition overlap the probability distribution functions for the three different methods to verify that their predictions are similar across the full data distribution of values.

```
library(rasterVis)

densityplot(models)
```

Step 3 - Identify spatial differences between the models using the standard deviation

Lets now take a look at the spatial differences. This step will allow to identify the geographical areas within the prediction domain where model predictions more agreee and disagree. To spatially compare model predictions we will estimate the standard deviation and the differences between the three SOC maps.

```
library(mapview)

SD <- calc(models , sd)

mapview(SD)
```

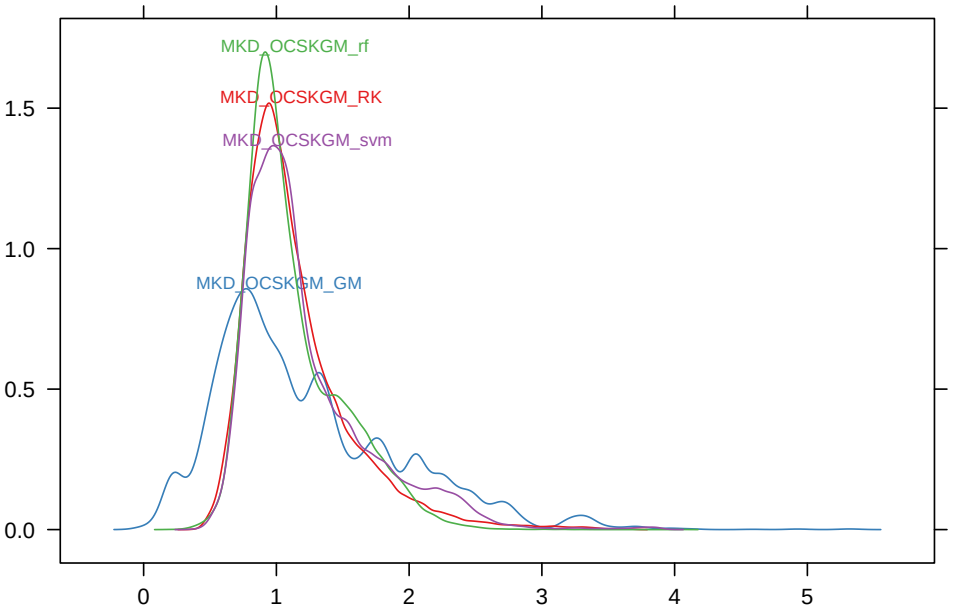


Figure 8.2: Density plot of the prediction for three DSM models

The mapview command will plot the standard deviation map in a `.html` file. Note roughly how the hotspots (yellow to red colors) of higher variance of predictions tend to be higher towards the west of the country, whereas models tend to agree in their predictions across the east side of the country. Note also that the variability although noisy, it shows a general pattern, (e.g., from east to west), suggesting that model agreement could be associated with specific land surface characteristics.

Now we will analyze specific differences between the three models (RK - RF, RK - SVM, RF - SVM).

```
library(raster)

RKRF  <- calc(models[[c(1,2)]], diff)
RKSVM <- calc(models[[c(1,3)]], diff)
RFSVM <- calc(models[[c(2,3)]], diff)

preds <- stack(RKRF, RKSVM, RFSVM)
names(preds) <- c('RKvsRF','RKvsSVM','RFvsSVM')

X <- raster::cellStats(preds, mean)
levelplot(preds - X, at=seq(-0.5,0.5, length.out=10),
          par.settings = RdBuTheme)
```

Note how the spatial differences of the predicted SOC values have similar patterns, but the difference between RK and RF seems to be less sharp than the differences

of SVM with the other two methods. Note that we use the `levelplot()` function to generate a better visualization (from red-to-white-to-blue) of the main effects of differences (e.g., if they are positive or negative), but we could also use the `mapview()` function to analyze these maps in a more interactive fashion. The variance of predictions derived from different models can be used as a proxy of model uncertainty and provides valuable information to consider in further applications of SOC maps (e.g., modeling crop production or quantifying SOC stocks).

8.2 Technical steps - Model evaluation

Step 1 - Data preparation

To compare the performance of the four models, we will compare the observed values used and the predicted values for the the validation points. We have to load the validation dataset and the prediction result of the four models. The table containing these values was prepared in Section 7.5.

```
dat <- read.csv("results/validation.csv")
```

We will prepare a new table from this data that we are going to use for model evaluation purposes. The new table should have the observed value, the predicted value and the model.

```
# Prepare 4 new data.frame with the observed, predicted and the model
modGM <- data.frame(obs = dat$OCSKGM, mod = dat$MKD_OCSKGM_GM,
                    model = "GM")

modRK <- data.frame(obs = dat$OCSKGM, mod = dat$MKD_OCSKGM_RK,
                    model = "RK")

modRF <- data.frame(obs = dat$OCSKGM, mod = dat$MKD_OCSKGM_rf,
                    model = "RF")

modSVM <- data.frame(obs = dat$OCSKGM, mod = dat$MKD_OCSKGM_svm,
                     model = "SVM")

# Merge the 3 data.frames into one
modData <- rbind(modGM, modRK, modRF, modSVM)

summary(modData)
```

```
##       obs               mod          model
##  Min.   :0.0111   Min.   :0.0111   GM :970
##  1st Qu.:0.6681   1st Qu.:0.8012   RK :970
##  Median :0.9518   Median :0.9612   RF :970
##  Mean   :1.1076   Mean   :1.0605   SVM:970
##  3rd Qu.:1.3744   3rd Qu.:1.1894
##  Max.   :6.1827   Max.   :5.3183
```

Table 8.1: Summary of different model evaluation statistics for the three models compared

model	FAC2	MB	MGE	NMB	NMGE	RMSE	r	COE	IOA
GM	0.80	0.01	0.41	0.01	0.37	0.63	0.54	0.15	0.57
RK	0.89	-0.08	0.36	-0.07	0.32	0.54	0.63	0.25	0.63
RF	0.88	-0.08	0.37	-0.07	0.33	0.56	0.59	0.23	0.62
SVM	0.86	-0.03	0.39	-0.02	0.35	0.58	0.53	0.18	0.59

```
##                    NA's   :74
```

Step 2 - Calculate statistical evaluation metrics

Now we will use the `modStats()` function to calculate common numerical model evaluation statistics which are described and mathematically defined in the **openair** package manual (Carslaw, 2015, Ch. 27, pp. 231-233). These include:

- n, the number of complete pairs of data;
- $FAC2$, fraction of predictions within a factor of two;
- MB, the mean bias;
- MGE, the mean gross error;
- NMB, the normalized mean bias;
- $NMGE$, the normalized mean gross error;
- $RMSE$, the root mean squared error;
- r, the Pearson correlation coefficient;
- COE, the Coefficient of Efficiency based on Legates and McCabe (1999), Legates and McCabe (2013). A perfect model has a $COE = 1$. A value of $COE = 0$ or negative implies no prediction capacity;
- IOA, the Index of Agreement based on Willmott et al. (2012), which spans between -1 and 1, with values approaching +1 representing better model performance.

A perfect model would have a $FAC2$, r, COE and $IOA \sim 1.0$, while all the others ~ 0.

However, digital soil mappers should have in mind that there is no such thing as a perfect model on digital SOC mapping for large areas, especially if we deal with sparse data from legacy soil profile or pit observations usually collected over long periods of time. Depending on the situation, some performance measures might be more appropriate than others. Hence, there is not a single best measure, and it is necessary to use a combination of the performance measures (Chang and Hanna, 2004).

```
library(openair)

modsts <- modStats(modData,obs = "obs", mod = "mod", type = "model")
```

From our SOC mapping example across FYROM, the three models generate similar

results. The $FAC2$ is over to 0.8 in all cases, being RK the one closer to 1. MB and NMB suggest that all the models tent to underestimate SOC because they are negative. SVM tend to underestimate SOC values less than RK and RF. The MGE, $NMGE$, and $RMSE$ suggest however that SVM is generating the larger error rate and by the values of r, COE and IOA, we could say that given available SOC data across FYROM, the RK method improves the predictive capacity of RF and SVM.

These conclusion can be verified by plotting a Taylor Diagram (see Figure 8.3), which summarizes multiple aspects of model performance, such as the agreement and variance between observed and predicted values (Taylor, 2001). Recent reports show that the integration of simple validation metrics (e.g., the RMSE correlation ratio) allows to extract information about modeling performance that could not be obtained by analyzing the validation metrics independently, such as the agreement between explained variance and bias (Guevara et al., 2018; Nussbaum et al., 2018). Taylor Diagrams interpteration rely on the relationships between explained variance and bias (from observed and modeled data). Note from our Taylor Diagram that the RK method is closer to the observed value, followed by RF. Although, no significant difference was evident between the three implemented algorithms.

```
TaylorDiagram(modData, obs = "obs", mod = "mod", group = "model",
              cols = c("green", "yellow", "red","blue"),
              cor.col='brown', rms.col='black')
```

However, we need also to check that the effectiveness of RK remains across all the full distribution of SOC observed values. For doing this, we can plot the conditional quantiles to verify the higher prediction capacity of RK (see Figure 8.4).

```
conditionalQuantile(modData,obs = "obs", mod = "mod", type = "model")
```

The blue line shows the results for a perfect model. In this case, the observations cover a range from 0 to $6kg \cdot m^{-2}$. Interestingly, the maximum predicted value is in all cases less than $3kg \cdot m^{-2}$, which is consistent with the MB and NMB previous results. The red line shows the median value of the predictions. Note that the median of predictions across the larger SOC values seems to have more variability across SVM and RK compared with RF, which tends to be conservative across the third quantile of data distribution. The shading shows the predicted quantile intervals (i.e., the 25/75th and the 10/90th). A perfect model (blue line) would have a very narrow spread (Carslaw and Ropkins, 2012). The histograms show the counts of observed (gray) and predicted values (blue). While RF shows more conservative results across the higher SOC values (e.g., over the 4th), SVM had the lowest model performance based on these metrics. SVM was also the method showing higher spatial differences compared to RF and RK.

We conclude, for this specific example, that RK showed the best performance based on the implemented evaluation metrics. Therefore RK is a suitable method predicting SOC values across FYROM, although other methods generate similar results, we propose that, given available data, RK could be also easier to interpret,

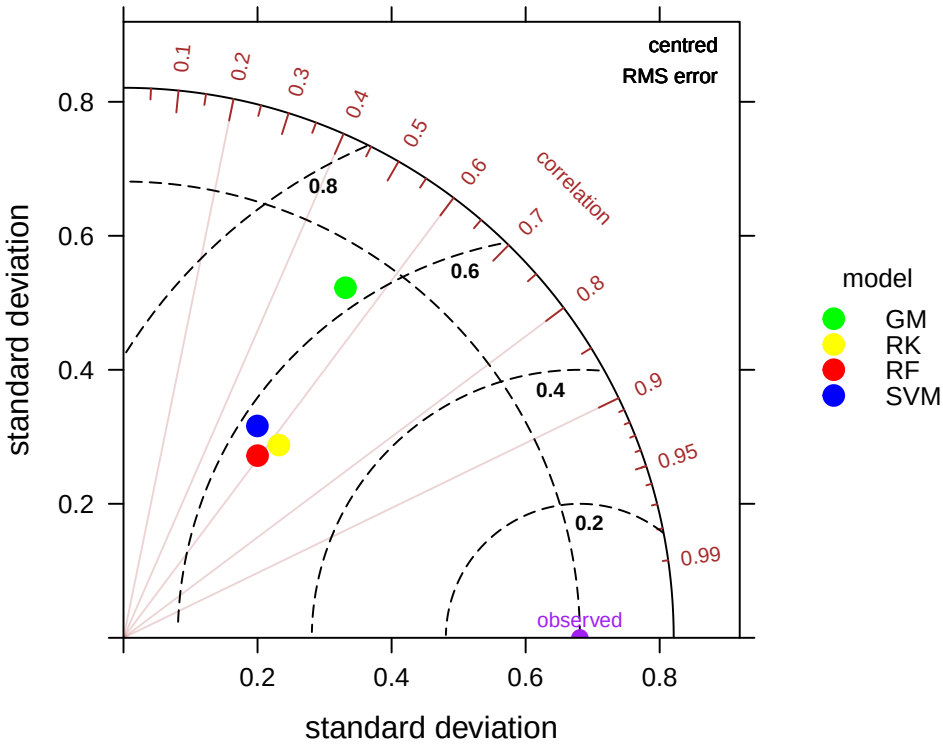


Figure 8.3: Taylor diagram used in the evaluation of the three selected DSM models

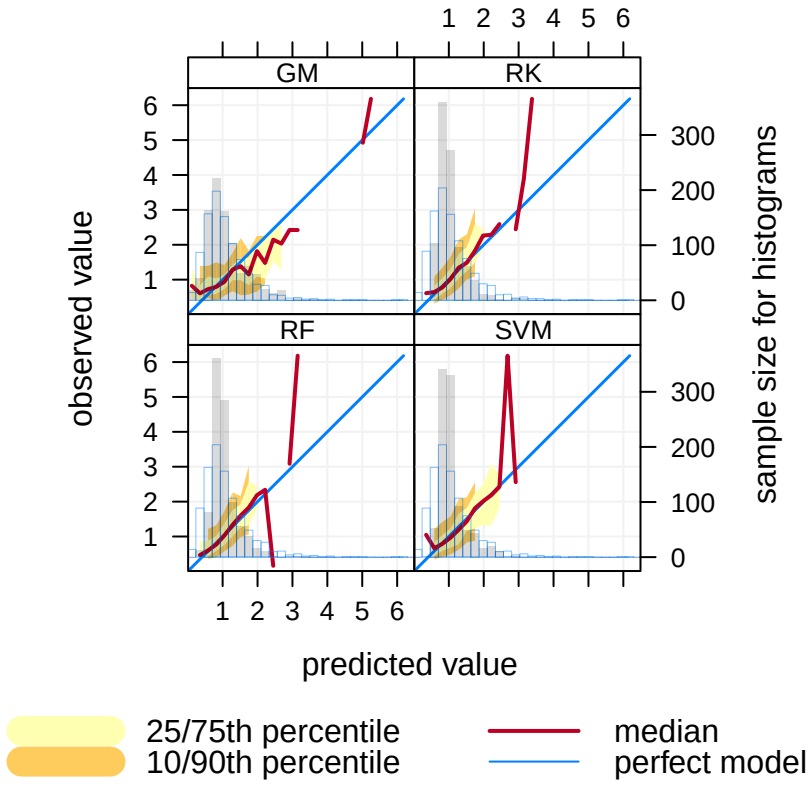


Figure 8.4: Effectiveness of the different DSM models across the full distribution of SOC observed values

because it provides the means to identify the main effects and coefficients of the relationships across the covariate space and the response variable. Beyond model evaluation, the spatial combination of different modeling approaches (e.g., by the means of the ensemble learning theory) should also be considered in order to maximize the accuracy of results.

Finally, we want to highlight that integrating different statistical methods and visualization tools, such as those provided by the **openair** package in **R** (Carslaw and Ropkins, 2012) will enhance our capacity to identify the best modeling approaches and a combination of SOC prediction factors given a specific dataset. Model evaluation benefits our understanding of the circumstances why each modeling approach will generate different results, which is a first step towards reducing uncertainty while increasing the spatial resolution and accuracy of predictions, the eternal problem for DSM.

Chapter 9

Uncertainty

G.B.M. Heuvelink

Soil mapping involves making predictions at locations where no soil measurements were taken. This inevitably leads to prediction errors because soil spatial variation is complex and cannot be modeled perfectly. It also implies that we are uncertain about the true soil class or true soil property at prediction locations. We only have the predictions, which differ from the true values in an unpredictable way, and hence we are uncertain about the true value. In fact, we may even be uncertain about the soil at the measurement locations because no measurement method is perfect and uncertainty also arises from measurement errors.

This Chapter describes how uncertainty may be characterized by probability distributions. It also explains how the parameters of these distributions may be derived, leading to quantification of uncertainty. We will see that this can become quite complex, because soil properties vary in space and are often cross-correlated, which the uncertainty model must take into account. A further complication is that there are many different sources of uncertainty. In some cases, it may be too difficult to arrive at a spatially explicit characterization of uncertainty, and in such case, statistical validation may be used to derive summary measures of the accuracy of soil maps. We begin this Chapter with a description of uncertainty sources.

9.1 Sources of uncertainty

Consider a case in which soil samples were taken from a large number of measurement locations in a study area, taken to the laboratory and analyzed for various soil properties. Let us further assume that the measurement locations were indicated on a topographic map and that the soil was also classified at each measurement location. Next, the soil property and soil type observations were used to create maps of soil properties and soil type using DSM techniques. These techniques not only make use of the soil observations but also benefit from maps of environmental

variables that are correlated with the soil, and hence help explain the soil spatial variation. Which sources of uncertainty contribute to uncertainty about the final soil maps? We distinguish four main categories.

9.1.1 Attribute uncertainty of soil measurements

Soil measurements suffer from measurement errors in the field and laboratory. Perhaps the soil was not sampled at the right depth, perhaps the organic layer was not removed completely before collecting soil material, or perhaps by accident bags were interchanged or numbered wrongly. Field estimates of soil type and soil properties are also not error-free, especially when estimation is difficult, such as estimation of SOC content or texture. Field estimates may also be subjective because soil scientists may be trained differently and so there may be systematic differences between their field estimates of soil properties. Similarly, it is also not uncommon for soil scientists to disagree about the soil type when classifying a soil in the field.

Laboratory analysis adds error too. Soil samples may not be perfectly mixed prior to taking a much smaller subsample that is actually measured; instruments have limited precision and may have systematic errors, climate conditions in the lab vary, and there can be differences between procedures used by laboratory personnel. Differences between laboratories are even bigger and may be of the same order of magnitude as the soil variation itself. It is strongly advised to always take sufficient duplicates and randomize the order in which soil samples are analyzed in the laboratory. This allows quantifying the combined field and laboratory measurement errors.

9.1.2 Positional uncertainty of soil measurements

When collecting soil data in the field we would generally note the geographic coordinates of the measurement locations. Nowadays this is easy with GPS instruments and depending on the device, modest to high positional accuracy can be achieved. But it may still be too large to be negligible. For instance, consider the case where the soil data are used to train a DSM model that predicts soil properties from covariates. Let these covariates be available at high spatial resolution and have substantial fine-scale spatial variation. Then it is clear that positional uncertainty in the soil measurements may link these measurements to the wrong covariates, which will weaken the strength of the relationship between the soil variable and covariates and deteriorate the quality of the final soil map.

Many soil legacy data suffer from large positional uncertainty. Locations may only be traced from vague descriptions such as *near village A* or *east of the road from B to C*. In such case, researchers should consider whether using such data for calibration of a DSM model and for spatial prediction using the calibrated DSM model is wise. It may do more harm than good. This depends on the specific DSM model used and the degree of spatial variation of the covariates. It also depends

on the degree of spatial variation of the soil property itself. If it has negligible fine-scale spatial variation and hence has similar values at the registered and actual geographic location, then little harm is done. For instance, in the Sahara desert many soil properties will show little spatial variation over distances of hundreds or perhaps thousands of meters, so in such case, poor geographic positional accuracy will not seriously affect DSM predictions.

9.1.3 Uncertainty in covariates

Maps of covariates that are used in DSM can also suffer from errors and uncertainties. For instance, a DEM is a major source of geomorphological covariates but DEMs are only approximations of the real elevation. DEM errors will propagate and cause uncertainty in geomorphological properties such as slope, aspect and topographic wetness index. As a result, the DSM model must be trained on covariate data that are merely approximations of the intended covariates, which will generally lead to weakened relationships and larger DSM prediction errors. Land cover is another example; soil properties may be strongly influenced by land cover, but such relationship may come out quite weak if the DSM model is trained with a land cover map that represents land cover wrongly for a large part of the study area.

Covariates also come in a specific spatial resolution which may be quite coarse in specific cases. In order to use the covariate in a fine-scale DSM model, the coarse-scale grid cell value will be copied to all fine-scale grid cells contained in it, but clearly, fine-scale spatial variation implies that uncertainties will be introduced. A possible solution might be to smooth the coarse-scale covariate prior to entering it to DSM calibration but clearly, this will not remedy all problems.

Uncertainty in covariates leads to weaker DSM models, but this weakening is not hidden to the developer because the deterioration of predictive power is implicitly included in the DSM model. For instance, the amount of variance explained by a DSM model that uses the true land cover as measured on sampling sites may be much higher than that of a model that uses a land cover map. Users may then be tempted to calibrate the DSM model with the true land cover data, but if they next apply that model using the land cover map to predict the soil at non-measurement locations they would systematically underestimate the uncertainty of the resulting map.

9.1.4 Uncertainty in models predicting soil properties from covariates and soil point data

Even if the soil point data and covariate data were error-free, the resulting DSM predictions would still deviate from the true soil properties. This is because the DSM model itself also introduces uncertainties. Models are merely simplified representations of the real world. The real world is too complex and approximations

are needed. For instance, even though we know that physical, chemical and biological processes determine the soil as given by the state equation of soil formation $soil = f(cl, o, r, p, t)$, the function f is too complex to be fully understood and implemented in a computer model. Instead, we use crude approximations such as multiple linear regression and machine-learning algorithms. These empirical models have the additional burden that extrapolation beyond conditions represented by the calibration data is difficult and risky. For extrapolation purposes, it is advised to use DSM models that better represent the mechanisms behind soil formation, but again it is practically impossible to build mechanistic models that represent the real world perfectly. This is not only because we may not understand all processes and their interactions well, but also because dynamic mechanistic models need much information, such as the initial state, boundary conditions, and driving forces. Such detailed information is generally lacking.

Model uncertainty is generally subdivided into model parameter uncertainty and model structural uncertainty. The first can be reduced by using models with fewer parameters or by using a larger calibration data set. The latter can be reduced by using a more complex model, but this will only work if there are enough data to calibrate such model. Thus, in general, a compromise has to be sought by choosing a level of model complexity that matches the amount of information available.

9.2 Uncertainty and spatial data quality

Research into spatial accuracy in geographic information science has listed five main elements of spatial data quality:

- Lineage;
- Positional accuracy;
- Attribute accuracy;
- Logical consistency; and
- Completeness.

We have already discussed positional and attribute accuracy. Lineage refers to documenting the original sources for the data and the processing steps. This is strongly related to the principle of reproducible research. Logical consistency addresses whether there are any contradictory relationships in the database. For instance, it checks whether all data have the same geographic projection and that measurement units are consistent. Completeness refers to whether there are any missing data. For instance, covariate maps must cover the entire study area if they are to be used as explanatory variables in a DSM model. Soil profile data need not capture all relevant soil properties and tend to have fewer soil measurements at greater depths.

In summary, there are many sources of uncertainty that affect the quality of DSM products. This Section has reviewed these sources but was purposely descriptive. The next Section selects a few major uncertainty sources and works out quantitatively how these cause uncertainty in the resulting soil map. Perhaps it is useful to

mention that focussing attention on errors and uncertainties may give the wrong impression that soil maps are generally inaccurate and of poor quality. This is not the message that we wish to convey here. But producers and users of soil maps should be aware of the sources of uncertainty and should ideally identify how these uncertainties affect the final product. Thus, quantification of the uncertainty in DSM maps, be it through explicit modeling or independent validation is important.

9.3 Quantifying prediction uncertainty

Uncertainties in soil measurements, covariates and DSM models propagate to resulting soil maps. The uncertainty propagation can fairly easily be traced provided that the uncertainty sources are characterized adequately. The most appropriate way of doing that is by making use of statistics and probability distributions. This Section also takes that approach and starts by providing a brief overview of probability distributions and how these may be used to represent uncertainty. Next, it analyses how the four sources of uncertainty distinguished in Section 9.1 lead to uncertainty in soil maps produced using DSM.

9.3.1 Uncertainty characterised by probability distributions

If we are uncertain about the value of a soil property at some location and depth this means that we cannot identify one single, true value for that soil property (Goovaerts, 2001; Heuvelink, 2014). Instead, we may be able to provide a list of all possible values for it and attach a probability to each. In other words, we represent the true but unknown soil property by a probability distribution.

For instance, suppose that we estimate the sand content of a soil sample in the field as 35%, while recognizing that a field estimate is quite crude and that the true sand content may very well be less or more than the estimated 35%. We might be confident that the estimation error is unlikely to be greater than 8%, and hence it would be reasonable to represent the sand content by a normal distribution with a mean of 35% and a standard deviation of 4%. For the normal distribution, 95% of the probability mass lies within two standard deviations from the mean, so we would claim that there is a 5% probability that the sand content is smaller than 27% or greater than 43%.

In the example above we had chosen the normal distribution because it is the most common probability distribution but we might as well have used a different distribution, such as the uniform or lognormal distribution. Indeed many soil properties, such as soil nutrient concentrations are better described by lognormal distributions, because values below zero cannot occur and because very high positive values (i.e., outliers) are not unlikely. For instance, we may estimate the organic carbon concentration (OC) of a soil sample as 1.2% and identify with it an asymmetric 95% credibility interval ranging from 0.8% to 2.5%. In general,

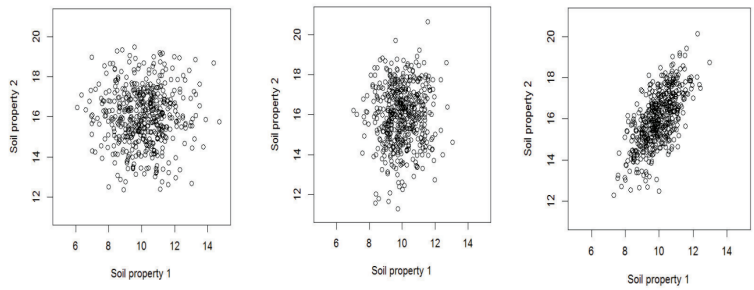


Figure 9.1: Scatter plots of 500 paired soil property values drawn from a two-dimensional normal distribution

statistical modeling is easier if the variables under study can be described by normal distributions. This explains why we usually apply a transformation to skewed variables prior to statistical modeling. For instance, when building a DSM model of OC, it may be wise to develop such model for the logarithm of OC and do a back-transform on the DSM predictions.

There are many different soil properties that in addition vary in space and possibly time. Thus, the characterization of uncertainty about soil properties needs to be extended and include cross- and space-time correlations. It is beyond the scope of this Chapter to explain this in detail, for this we refer to standard textbooks such as Goovaerts (1997) and Webster and Oliver (2007). If we assume a joint normal distribution, then a vector of soil properties (be it different soil properties or the same soil property at multiple locations, depths or times) Z is fully characterized by the vector of means m and variance-covariance matrix C.

Figure 9.1 shows three examples of 500 paired soil property values that were simulated from different bivariate normal distributions. The left panel shows an uncorrelated case with equal standard deviations for both properties. The centre and right panels show a case where soil property 2 has a greater standard deviation than soil property 1. The difference between these two cases is that the centre panel has a zero correlation between the two soil properties while it is positive in the right panel.

9.3.2 Propagation of model uncertainty

Now that we have clarified how uncertainty in soil properties may be characterized by probability distributions, let us consider what these distributions look like in DSM and how these are influenced by the uncertainty sources described in Section 9.1. We begin with uncertainty source 4, uncertainty in DSM models. We noted before that uncertainty in DSM models may be separated in model parameter and model structural uncertainty. A typical example of this is a multiple linear regression model:

$$Z(s) = \beta_0 + \beta_1 \cdot X_1(s) + \beta_2 \cdot X_2(s) + \varepsilon(s) \tag{9.1}$$

Note that here for simplicity we assumed two environmental covariates X_1 and X_2 while in practice we are likely to use many more. Parameter uncertainty of this model occurs because the parameters β_0 , β_1 and β_2 are merely estimated using calibration data. Under the assumptions made by the linear regression model, these estimation errors are normally distributed and have zero mean, while their standard deviations and cross-correlations can also be computed (Snedecor and Cochran, 1989, Section 17.5). The standard deviations become smaller as the size of the calibration dataset increases. Both the standard deviations and cross-correlations are standard output of statistical software packages. Thus, we could sample from the joint distribution of the parameter estimation errors in a similar way as displayed in Figure 9.1.

The model structural uncertainty associated with the multiple linear regression model Eq. 9.1 is represented by the stochastic residual ε. It too is normally distributed and has zero mean, while its standard deviation depends on the (spatial) variation of the soil property Z and the strength of the relationship between Z and the covariates X_1 and X_2. If the covariates explain a great deal of the variation of the soil property then the standard deviation of the residual will be much smaller than that of the soil property, as expressed by the goodness-of-fit characteristic R^2, also termed Amount of Variance Explained (AVE) (see Section 7.4.1). It will be close to 1 in case of a strong linear relationship between soil property and covariates. In that case, the standard deviation of the stochastic residual will be much smaller than that of the soil property, because a large part of the variation is explained by the model. If the covariates bear no linear relationship with the soil property (i.e., $R^2 = 0$), the stochastic residual will have the same standard deviation as the soil property.

Since the joint probability distributions of the parameter estimation errors and the stochastic residual can analytically be computed and are routinely provided by statistical software, it is not difficult to analyse how these uncertainties propagate through the DSM model Eq. 9.1. This can be done analytically, because Eq. 9.1 is linear in the stochastic arguments (note that the covariates are treated known and deterministic). If we predict the soil property Z at a prediction location s_0 using the calibrated regression model as:

$$\hat{Z}(s_0) = \hat{\beta}_0 + \hat{\beta}_1 \cdot X_1(s_0) + \hat{\beta}_2 \cdot X_2(s_0) \tag{9.2}$$

then the prediction error will be normally distributed with zero mean and variance (i.e., the square of the standard deviation) given by:

$$Var(\hat{Z}(s_0)) - Z(s_0)) = Var(\hat{\beta}_0) + Var(\hat{\beta}_1) \cdot X_1(s_0)^2 + Var(\hat{\beta}_2) \cdot X_2(s_0)^2 + 2Cov(\hat{\beta}_0, \hat{\beta}_1) \cdot X_1(s_0) + 2Cov(\hat{\beta}_0, \hat{\beta}_2) \cdot X_2(s_0) + 2Cov(\hat{\beta}_1, \hat{\beta}_2) \cdot X_1(s_0) \cdot X_2(s_0) + Var(\varepsilon(s_0)) \tag{9.3}$$

This is a complicated expression but all entries are known and hence it can be easily calculated.

In many DSM applications, an additional step will be included that makes use of the fact that the stochastic residual ε in Eq. 9.1 is spatially autocorrelated, as characterized by a semivariogram. If this is the case the residual spatial correlation can be exploited by incorporating a kriging step (Hengl et al., 2004). Kriging has been explained in Chapter 6, where it was also explained that the uncertainty in the predictions is quantified by the kriging variance. We will not repeat the theory here, but simply note that the kriging variance computes the prediction error variance just as was done in Eq. 9.3, but that in case of kriging the $Var(\varepsilon(s_0)))$ term in Eq. 9.3 is replaced by a smaller term, because kriging benefits from residual spatial correlation. In fact, in case of a pure nugget variogram, the kriging variance would be identical to Eq. 9.3, because in such case there is no spatial autocorrelation that one can benefit from. Note also that here we refer to kriging with external drift because we included a non-constant mean (i.e., covariates $X1$ and X_2). If no covariates were included Eq. 9.3 would simplify dramatically leaving only uncertainty in the estimated (constant) mean and the stochastic residual. This might then be compared with the ordinary kriging variance.

So far we considered uncertainty in DSM models that are linear in the covariates and that represent the model structural uncertainty by an additive stochastic term. This was relatively easy because tracing how uncertainty in model parameters and model structure propagate to the model output could be done analytically. However, using linear models also poses serious restrictions. The relationship between soil properties and covariates are typically not linear but much more complex. This has led to the development and use of complex non-linear DSM models, such as regression trees, artificial neural networks, support vector machines and random forests approaches, all summarised under the term *machine learning* (e.g., Hengl et al., 2015a). These more complex models typically yield more accurate soil predictions but quantification of the associated uncertainty is more difficult. In most cases, one resorts to validation and cross-validation statistics that summarise the prediction accuracy over the entire study area. How this is done will be explained in detail in Section 9.3.3. Such summary validation measures are very valuable but are no substitute for spatially explicit uncertainties such as the kriging variance and the prediction error variance presented in Eq. 9.3. Research into quantification of location-specific uncertainties when using machine learning algorithms is therefore important. However, it is beyond the scope of this Chapter to review this area of ongoing research. One particular approach makes use of quantile regression forests. We refer to Meinshausen (2006) for a general text and to Vaysse and Lagacherie (2017) for a DSM application of this promising, albeit computationally

challenging approach.

9.3.3 Propagation of attribute, positional and covariate uncertainty

In Section 9.1 we noted that next to uncertainties in model parameters and model structure there may also be uncertainties in the attribute values and positions of the soil point data and in the covariates. These sources of uncertainty will also affect the outcome of DSM model predictions.

Uncertainties in soil attribute values effectively mean that the DSM model is calibrated with error-contaminated observations of the dependent variable. Let us consider the multiple linear regression model Eq. 9.1 again. True values of the dependent variable Z (i.e., the target soil property, such as pH, clay content or total nitrogen concentration) are no longer for calibration of this model. Instead, we must make do with measurements Y of Z:

$$Y(s_i) = Z(s_i) + \delta(s_i), \quad i = 1 \ldots n \tag{9.4}$$

where n is the number of measurement locations and $\delta(s_i)$ is a random variable representing measurement error. It is custom to assume that all $\delta(s_i)$ are normally distributed, have zero mean and are mutually independent, although these assumptions are not strictly necessary. Their standard deviations may vary between cases and depend on the accuracy and precision of the measurement method. For instance, field estimates tend to be more uncertain than laboratory measurements and so the corresponding measurement errors will have a larger standard deviation. The consequence of the presence of measurement errors is that the estimates of the model parameters will be more uncertain. This is no surprise because the calibration data are of poorer quality. The prediction error variance will be greater too, for the same reason. If spatial correlation of the model residual ε is included and an extension to kriging with external drift is made, uncertainty due to measurement errors is further increased because the conditioning of predictions to observations cannot benefit as much as when the observations were error-free. For mathematical details we refer to Cressie (1993). Finally, we should also note that if different observations have different degrees of measurement error, then this will influence the weights that each measurement gets in calibration and prediction. Measurements with larger measurement errors get smaller weights. This is automatically incorporated in multiple linear regression and kriging with external drift, but how this can be incorporated in machine-learning approaches is less clear.

Positional uncertainty of soil point observations will also deteriorate the quality of the predictions of calibrated DSM models. However, it is difficult to predict how much the prediction accuracy is affected. It largely depends on the degree of fine-scale spatial variation of the soil property and covariates. For instance, if both the soil property of interest and the covariates are spatially smooth and hardly change over distances within the range of spatial displacement due to positional uncertainty, then little damage is afflicted by positional uncertainty. But

otherwise much harm can be done because the soil observations will be paired with covariate values from displaced locations that can be very different. So far, this interesting and important topic has received only little attention in the DSM literature. Grimm and Behrens (2010) and Nelson et al. (2011) are two examples of studies that assessed the effect of positional error on the accuracy of digital soil maps.

Finally, there are also uncertainties in covariates that affect the accuracy of DSM predictions. In fact, these uncertainties are already incorporated in the model structural uncertainty discussed before, because offering covariates that are poor approximations of the true soil forming factors will explain little of the spatial variation and lead to low goodness-of-fit statistics. From a statistical point of view, the covariates used in Eq. 9.1 need not be the *true* soil forming factors but could as well be proxies of those. This does not harm the theory and quantification of the prediction error variance such as through Eq. 9.3 in the multiple linear regression case or using the kriging variance in a KED approach remain perfectly valid. This does not mean that digital soil mappers should not look for the most accurate and informative covariates because clearly weak covariates lead to poor predictions of the soil (Samuel-Rosa et al., 2015).

Chapter 10

Data sharing

T. Hengl, L. Poggio, E. Ribeiro & B. Kempen

This Chapter reviews possibilities and *good practices* of exchanging produced soil data. Once the analysis, spatial prediction and quality control have been all completed, it is useful to follow some minimum steps and export and prepare the data for distribution so that its potential users can easily access it, use it, and make a correct interpretation of data. We consider geo-publishing options for soil data either based on using third-party web services or by using one's own installation of the software. We put a clear focus on using the Open Source software solutions: GDAL, GeoServer, Leaflet, OpenLayers and **R**, and public domain data and metadata standards.

The authors have more than 15 years of experience in producing, publishing and sharing soil maps and have been involved in large soil mapping projects where data volumes often exceed standard desktop GIS capacities. For information on specific software please refer to the provided links. Even more information on using GDAL and similar GIS tools through a command line can be found via the Global Soil Information Facilities (GSIF) tutorials of ISRIC at http://gsif.isric.org. The text is illustrated with example scripts of the statistical software **R** in combination with GDAL.

The authors of this Chapter used **R** packages. To run the code provided in this Chapter, the following packages need to be installed in the **R** user library. If the packages are not yet installed, the `install.packages()` function can be used.

```
# Installing packages for Chapter 'Data Sharing'
install.packages(c("soiltexture", "rgdal", "plotKML",
                   "sf", "RSQLite", "leaflet",
                   "htmlwidgets", "GSIF", "raster"))
```

10.1 Export formats

10.1.1 Type of soil data and their formatting

Before we start reviewing soil data formats, it is useful to understand which types of soil variables, soil maps, and soil DBs are most commonly generated and used, and what are their specific advantages and limitations. Soil science works with many variables common to ecology and/or physical geography (e.g. soil temperature), but it also works with several variables specific to soil science only. Some soil factor-type variables specific to soil science only are for example:

- Soil taxa or soil classes (this includes taxonomic systems and connected diagnostic soil properties and horizons);
- Soil texture-class systems;
- Soil color classification systems e.g. Munsell color codes;
- Soil drainage classes (hydrological classifications); and
- Soil diagnostic horizons.

Consider for example the following soil texture data:

```
library(soiltexture)

tex <- data.frame(
  CLAY = c(05,60,15,05,25,05,25,45,65,75,13,47),
  SILT = c(05,08,15,25,55,85,65,45,15,15,17,43),
  SAND = c(90,32,70,70,20,10,10,10,20,10,70,10)
 )

TT.plot(class.sys = "USDA.TT", tri.data = tex, main = "",
        cex.axis=.7, cex.lab=.7)
```

The way soil texture data is displayed and texture classes (SaLo, Lo, Sa, etc.) used in a texture triangle is specific to soil science. The way this data is formatted and presented can be, likewise, specific to soil science only.

Most of the soil data is in fact spatial. *Spatial* implies that spatial (and temporal) reference is attached to each measured/estimated value, i.e. it is location specific. Spatio-temporal references typically include for example:

- Geographic location in local or geographic coordinates (ideally longitude and latitude in the WGS84 coordinate system);
- Depth interval expressed in cm from land surface (upper and lower depth);
- Support size or referent soil volume (or voxel) i.e. the horizontal sampling area multiplied by the thickness of the sampling block;
- Temporal reference i.e. begin and end date/time of the period of measurements/estimations.

Spatial data formats are used to represent spatial objects. This can be (Bivand et al., 2013; Neteler and Mitasova, 2013):

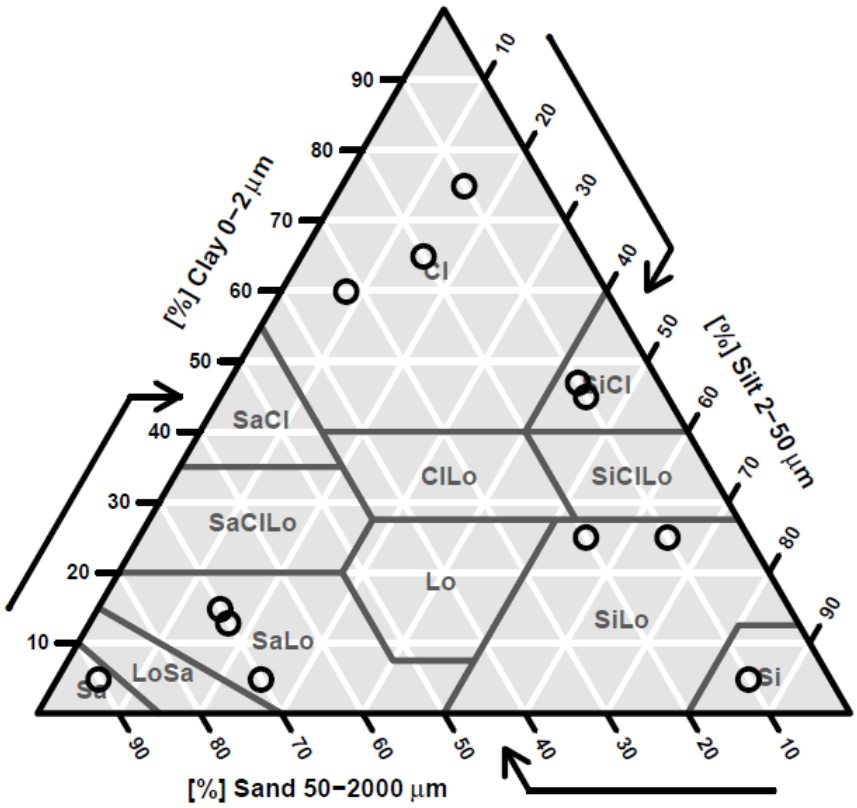


Figure 10.1: Soil texture triangle plot as an example of soil science specific data

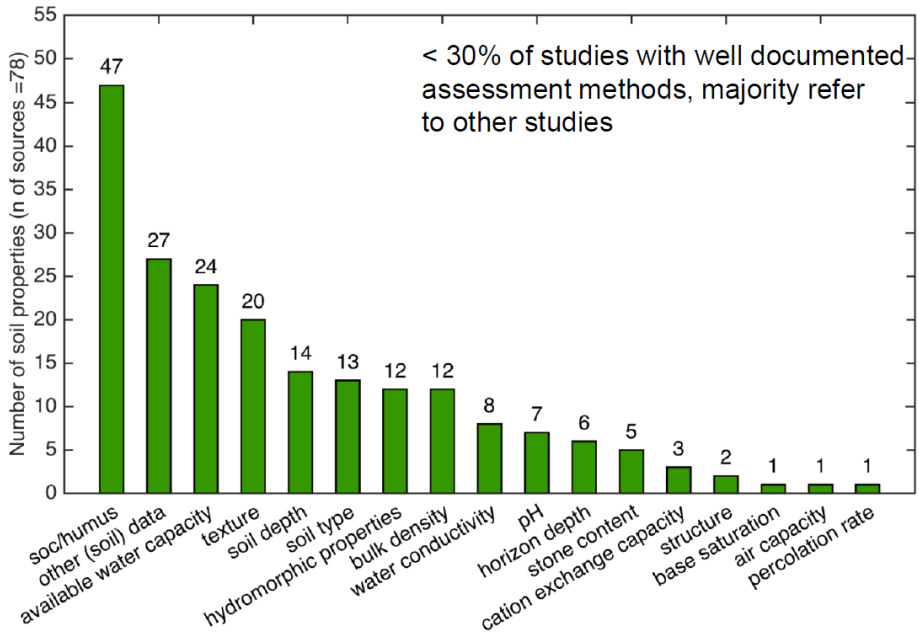


Figure 10.2: Some frequently required soil variables (sorted by number of studies) based on the study by Keller et al. (2014). This list is probably country/project specific but illustrates the differences considering the interest in soil data

- **Points** (2D or 3D): Used to represent sampling locations, soil horizons, soil profiles, etc.
- **Lines** (2D): Used to represent soil transects, streams, administrative boundaries, etc.
- **Polygons** (2D): Used to represent soil mapping units and/or geomorphological units, landforms, administrative areas, farms, plot trials, etc.
- **Grids or rasters** (2D or 2.5D): Used to represent soil spatial predictions (spatially complete) of soil properties and classes, etc.
- **3D grids or Voxels**: Used to represent soil spatial predictions (spatially complete) of soil properties in 3D.

It is also important to be able to distinguish between sampled or predicted soil data:

- **Sampled soil Data**: Soil samples (usually points or transects) are spatially incomplete. They are used to generate spatial predictions.
- **Predicted soil data**: Spatial predictions of soil variables (soil maps) are spatially complete. They are used for decision making and further modeling, i.e. they are used to construct a Soil Information System (SIS).

A collection of spatially exhaustive soil grids of various soil properties (physical and chemical soil properties, soil water, soil classification, etc.) make a SIS. SIS

are often complemented with soil sample data and serve both data formats. A SIS should preferably be a database (DB), so that users can access and query data using some standard DB languages (e.g. SQL). Steps to export soil data into a DB format are explained in later Sections.

10.1.2 General GIS data formats: vector, raster, table

All soil data we produce through soil mapping can be in principle distributed using one of the two basic GIS formats of data:

- **Vector format**: This format is often more suitable for exporting point, line, and polygon (areal) data.
- **Raster or gridded format**: This format is often more suitable for exporting spatial predictions of soil variables.

Data in vector format can be converted to raster (see e.g. `rasterize()` function in the **raster R** package) and vice versa, raster data can be converted to vector formats. For example, rasters can be converted to polygons (see e.g. `rast2vect()` function in the **plotKML R** package). If the conversion is done carefully and if all the relations between scale and pixel size have been considered (see Hengl, 2006, for more details), then information loss due to conversion from raster to vector and vice versa should be minimal.

Both vector and raster GIS data can also be converted to tabular data formats. By converting a GIS layer to a table, spatial geometry and spatial relations will be *stripped off*, so that only limited spatial analysis operations can be applied. To convert raster layers to tabular data, consider using the `SpatialPixelsDataFrame` class in the **sp** package and/or the `RasterLayer` class from the **raster** package in combination with the **rgdal** package (Bivand et al., 2013).

```
library(rgdal)
library(plotKML)

?readGDAL
spnad83 <- readGDAL(system.file("pictures/erdas_spnad83.tif",
                                package = "rgdal")[1])
spnad83.tbl <- as.data.frame(spnad83)
str(spnad83.tbl)
```

In this example, `as.data.frame()` is a function converting a raster object to a table. Note that the output table now contains coordinates for each cell (center of the grid node), which is in fact memory inefficient as coordinates are provided for each row in the table.

Likewise, to convert a vector layer to tabular formats one can use the Simple Features (SF) functionality of the **sf** package . The SF standard is widely implemented in spatial databases (PostGIS, ESRI ArcGIS) and forms the vector data basis for libraries such as GDAL and web standards such as GeoJSON (http://geojson.org/). To convert for example spatial polygons layer to a tabular format we would use:

```
library(sf)
library(plotKML)

data(eberg_zones)
class(eberg_zones)
eberg_zones.tbl <- as(eberg_zones, "sf")
str(eberg_zones.tbl)
```

Note that using spatial layers in simple tabular formats can be cumbersome because many spatial relationships and properties are likely lost (although these can be assigned reversibly). In addition, the size of tabular objects is much bigger than if we use data in the original GIS data formats, especially if those formats support compression. On the other hand, having data in tabular format can be often the only way to exchange the data from spatial to non-spatial databases or from software without any data communication bridge. Also, tabular data is human-readable which means that it can be opened in text editors, spreadsheet programs or similar.

10.1.3 Recommended GIS data exchange formats

As a general recommendation producers of soil data should primarily look at using the following data formats for exchanging soil data (points, polygons, and rasters):

- **GPKG**: An open format (`*.gpkg` file) for geospatial information. Platform-independent, portable, self-describing, compact format for transferring geospatial information.
- **GeoTIFF** (for rasters): A `*.tiff` or `*.tif` (image) file that allows embedding spatial reference information, metadata and color legends. It also supports internal compression algorithms and hierarchical indexing.

Both formats can be read easily in **R** or similar data processing software. Vectors are also commonly exported and shared in ESRI shapefile format (as `*.shp` file). The advantage of GPKG format versus somewhat more common ESRI shapefile format is that `*.gpkg` files are basically a portable database (SQLite container) so that the user does not have to import the whole data into a program but also fetch parts of data by using SQL queries and it can handle vector and raster data in it. The following example demonstrates how to create a `*.gpkg` file and how to query it:

```
library(RSQLite)

data(eberg)
coordinates(eberg) <- ~X+Y
proj4string(eberg) <- CRS("+init=epsg:31467")
writeOGR(eberg, "eberg.gpkg", "eberg", "GPKG")

con <- dbConnect(RSQLite::SQLite(), dbname = "eberg.gpkg")
```

```
df <- dbGetQuery(con, 'select "soiltype" from eberg')

summary(as.factor(df$soiltype))

dbGetQuery(con,
           'select * from gpkg_spatial_ref_sys')[3,"description"]
```

Note that the **RSQLite** package is a generic package for connecting to SQLite DBs. This means that `*.gpkg` files can be accessed and updated in its native storage format without intermediate format translations. Just putting a `*.gpkg` file on a server with read and execute access allows users to connect and fetch data.

Alternatively, it is also a good idea to store point data in a non-spatial format such as simple tables. For example, in comma-separated file format (CSV) as a `*.csv` file. A fast way to publish and share tabular data is to use Google Fusion Tables™. Google Fusion Tables™ have an API that allows accessing and using tabular data through various programming platforms. The limitation of using Google Fusion tables™ is, however, data size (currently about 1 GB per user) and similar data volume limits, so this platform should be only used as an intermediate solution for smaller data sets.

GeoTIFF format is highly recommended for sharing raster data for the following reasons:

- It is GDAL's default data format and much functionality for subsetting, reprojecting, reading and writing GeoTIFFs already exists (see GDAL utils).
- It supports internal compression via creation options (e.g. COMPRESS=DEFLATE).
- Extensive overlay, subset, index, translate functionality is available via GDAL and other open source software. Basically, GeoTIFF functions as a raster DB.

Consider for, example the `gdallocationinfo()` function which allows spatial queries following some indexing system such as row and column number:

```
spnad83.file = system.file("pictures/erdas_spnad83.tif",
                           package = "rgdal")[1]
system(paste0('gdallocationinfo ', spnad83.file, ' 100 100'))
```

Such type of overlay operations, thanks to GDAL (Warmerdam, 2008), are extremely fast and efficient. Likewise, `gdalwarp()` function can be used subset rasters based on spatial extent or grid index. Rasters can be imported to GeoServer and shared through Web Coverage Service (see Section 10.2) or similar likewise function as a spatial raster DB.

As a general recommendation, and to avoid large file sizes, we recommend, however, that you always use integers inside GeoTIFF formats because floating point formats can lead to up to more than four times larger sizes (without any gains in accuracy). This might mean you have to multiply the values of the soil property

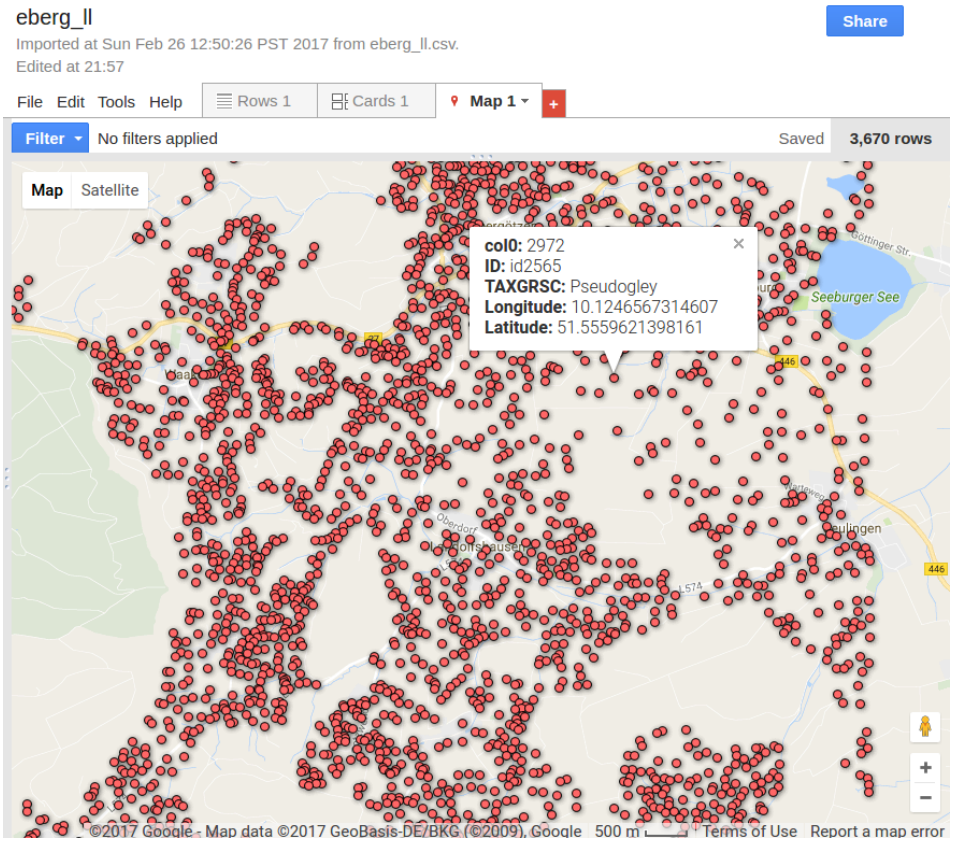


Figure 10.3: Displaying point dataset eberg (used in the previous example) in Google Fusion Tables.

of interest by 10 or 100, in order not to lose accuracy (e.g. multiply pH values by 10 before exporting your raster in GeoTIFF format).

10.2 Web services: serving soil data using web technology

10.2.1 Third-party services

If you are a data producer but with limited technical capacity and/or financial resources, then publishing geo-data through a third-party service could be very well that the easiest and most professional solution for you. Some commonly used commercial web services to share geo-data are:

- Google MyMaps (https://www.google.com/mymaps)
- ArcGIS Online (https://www.arcgis.com/home/)
- MapBox (https://www.mapbox.com/)
- CARTO (https://carto.com/)

All these have limitations and primarily suitable for sharing vector type data only. Their free functionality is very limited so before you start uploading any larger data sets, please check the size limits based on your account. Upgrading your license will allow you to increase storage and functionality so that even with few hundred dollars per year you could have a robust solution for sharing your data with thousands of users.

Soil data producers can also contact ISRIC, as World Data Centre for Soils, to request support for hosting and/or distributing their soil data in case they lack the technical capacity to do so themselves while adhering to the data sharing agreement and license set by the data producer.

10.2.2 GeoServer (web serving and web processing)

GeoServer (http://geoserver.org/) is an open source software solution for serving raster or vector data. It includes the majority of the Open Geospatial Consortium (OGC) service standards: the Web Map Service, Web Coverage Service and Web Processing Service (Youngblood, 2013). Installation and maintenance of GeoServer is however not trivial and requires specialized technical staff. Web services can also entail significant costs depending on the amount of web-processing and web-traffic. For every medium to large size organization, it is probably a better idea to use the out-of-box solution for GeoServer which is the GeoNode (http://geonode.org/). GeoNode also includes a web-interface and user-management system so that new layers can be uploaded to GeoServer through a web-form type interface.

The very important functionality of GeoServer are the OGC standard services such as the Web Coverage Service (WCS) and the Web Feature Service (WFS). WCS means that not only data views are available to users, but WCS can also do

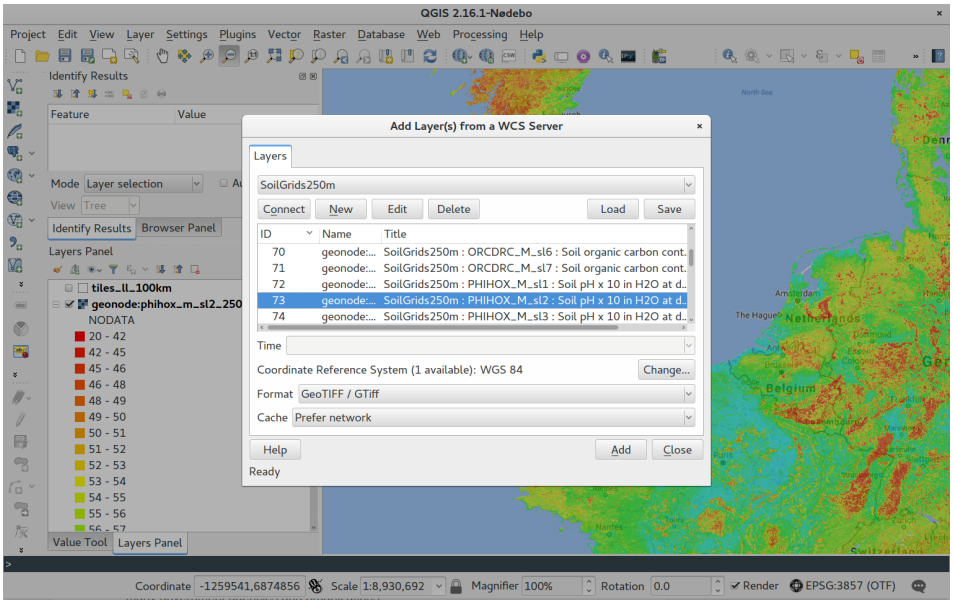


Figure 10.4: SoilGrids (Hengl et al. 2017) WCS opened in QGIS

data translation, aggregation or resampling, overlay etc. Consider for example the SoilGrids WCS (which can also be opened in QGIS or similar software supporting WCS). Users can direct this WCS and request only a subset of data for some area, aggregated to some preferred resolution/pixel size by using *gdal_translate* or similar. This means that, in few steps, users can download a subset of data on GeoServer in a preferred format without a need to download the whole dataset.

10.2.3 Visualizing data using Leaflet and/or Google Earth

A quick way to visualize produced soil maps and then share them to users without GIS capacities is to use **leaflet** package in **R**. Leaflet is basically a stand-alone web-page that contains all information (including some popular Web Mapping Services) so that users can visually explore patterns without having to install and use any desktop GIS. Consider the following example:

```
library(leaflet)
library(htmlwidgets)
library(GSIF)
library(raster)

demo(meuse, echo=FALSE)
omm <- autopredict(meuse["om"], meuse.grid[c("dist","soil","ffreq")],
                   method="ranger", auto.plot=FALSE, rvgm=NULL)
```

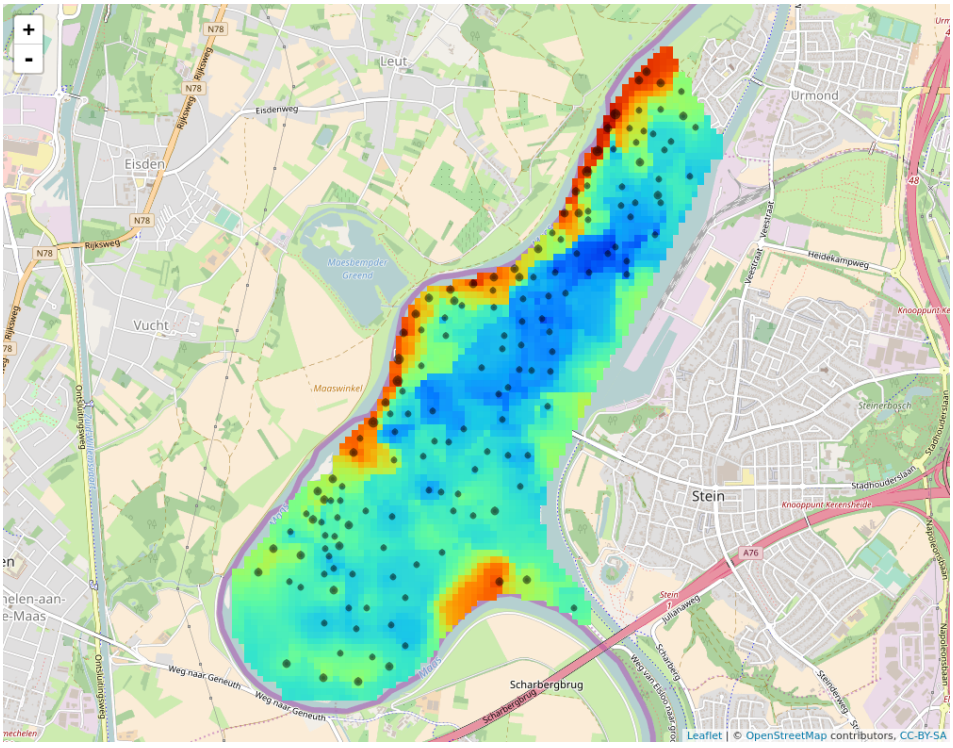


Figure 10.5: Sampled locations and produced predictions visualized using Leaflet package

```
meuse.ll <- reproject(meuse["om"])

# Reprojecting to +proj=longlat +datum=WGS84
m = leaflet() %>% addTiles() %>%
    addRasterImage(raster(omm$predicted["om"]),
    colors = SAGA_pal[[1]][4:20]) %>%
    addCircles(lng = meuse.ll@coords[,1],
    lat = meuse.ll@coords[,2],
    color = c('black'), radius=meuse.ll$om)

saveWidget(m, file="organicmater_predicted.html")
```

Note that the whole data set including styling and legends is basically available through a single `*.html` file (`organicmater_predicted.html`). Anyone opening that HTML format file in their browsers will get an interactive web-map that contains both samples and spatial predictions.

An alternative to using Leaflet is to put all data, including documents and mul-

timedia, about your project in a Keyhole Markup Language (KML) format. As `*.kml` file the data is available for viewing in Google Earth (Hengl et al., 2015b). KML is very rich in what it can incorporate: textual data, photographs, documents, animations, videos, etc. In fact, probably whole projects can be put into a single `*.kml` file so that the users only need to open it in Google Earth and then explore interactively. Note that `*.kml` files with ground overlays will be generated by GeoServer by default, although further customization is up to the data producer.

10.3 Preparing soil data for distribution

10.3.1 Metadata

One important thing to consider prior to data distribution is construction of metadata (explanation of data, how was it produced and what are the exact technical specifications). There are several metadata standards that can be used to prepare metadata. More recently, complete and consistent metadata is a requirement by many government agencies and organizations. There are now several public metadata validators that run all possible consistency and completeness checks before the metadata (and data) can be accepted.

Typical metadata should (at least) contain:

- Detailed description of the variables available in the data;
- Data license and terms of use (URL);
- Explanation of measurement methods and units used;
- Mention of the reference support size including referent depth intervals to which the soil data refers to (e.g. 0 cm – 30 cm depth interval);
- Mention of the referent time period in which the calibration data was collected;
- Link to literature (report, book or scientific article) where the data production is explained in detail. Using a published and peer-reviewed scientific article as the main reference for data is a good practice since it also shows that the data production process has been evaluated by independent researchers; and
- Project homepage, i.e. URL containing more information and especially up-to-date contacts where users can find original data producers and request support.

Metadata (including color legends) can be also directly embedded into the GeoTIFF `*.tif` file by using the *gdal_edit* command available in GDAL. The following example shows how to add a simple explanation of the data and a URL to find more info about the GeoTIFF in **R**:

```
data("eberg_grid")

gridded(eberg_grid) = ~ x+y
```

```
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
writeGDAL(eberg_grid["DEMSRT6"], "eberg_DEM.tif",
            options="COMPRESS=DEFLATE")
?eberg

system(paste0('gdal_edit.py
-mo \"DESCRIPTION=elevation values from the SRTM DEM\"
-mo \"DOWNLOAD_URL=http://geomorphometry.org/content/ebergotzen\"
eberg_DEM.tif'))
system('gdalinfo eberg_DEM.tif')
```

Similarly, all necessary metadata can be added into GeoTIFF so that future users have all information at one place, i.e. inside the data file.

10.3.2 Exporting data: final tips

As we have shown previously, if you export soil data into either GPKG and/or GeoTIFF format, these data can be accessed using DB operations. In fact, by exporting the data to GPKG and GeoTIFFs, you have created a soil spatial DB or a soil information system. This does not necessarily mean that its targeted users will be able to find all information without problems and/or questions. How usable and how popular a data set is, is a function of many aspects, not only data quality. You could create maps of perfect quality but have no users at all. |The following instructions should be definitively considered, as a way to boost the usability of your data are.

1. Make a landing page for your data that includes:

a. Simple access/download instructions;
b. Screenshots of your data in action (people prefer visual explanations with examples);
c. Links to key documents explaining how the data was produced; and
d. Workflows explaining how to request support (who to contact and how).

2. Make data accessible from multiple systems, e.g, both via WCS, FTP and through a mirror site. This might be inefficient considering there will be multiple copies of the same data, but sometimes it quadruples data usage.
3. Explain the data formats used to share data, and point to tutorials that explain how to access and use data to both beginners and advanced users.
4. Consider installing and using a version control system (or simply use GitHub or similar repository) so that the users can track back versions of data.
5. Consider closely following principles of reproducible research (all processing steps, inputs, and outputs accessible). This tutorial comes with **R** code that is available via GitHub so that everyone should be able to reproduce the examples shown in the text.

10.4 Export formats

The produced results need to be exported in formats that can be easily read by a variety of GIS software. Two widely used formats for raster data are GeoTIFF and KML.

- **GeoTIFF**: A public domain metadata standard which allows georeferencing information to be embedded within a `*.tiff` file. The potential additional information includes map projection, coordinate systems, ellipsoids, datums, and everything else necessary to establish the exact spatial reference for the file.
- **KML**: Keyhole Markup Language is an XML notation for expressing geographic annotation and visualization within internet-based, two-dimensional maps and three-dimensional Earth browsers. KML became an international standard of the OGC in 2008.

Raster data in GeoTIFF format need a defined geographic projection. Each country has its own national system (or systems). In order to construct a mosaic of national datasets, a common projection has to be defined and national data need to be re-projected in the common projection. Data projections can be managed using open source tools such as GIS software, GDAL tools suite (http://www.gdal.org/) and various packages of the **R** software (https://www.r-project.org/). Projections can be defined according to different standards. A common way is the EPSG database. EPSG Geodetic Parameter Dataset is a collection of definitions of coordinate reference systems and coordinate transformations which may be global, regional, national or local application. A numeric code is assigned to each of the most common projections, making easier to refer to them and to switch between them. One of the most common global projections is WGS84 (EPSG 4336), used for maps and by the GPS satellite navigation system.

Each file should be accompanied by a set list of metadata. Geospatial metadata is a type of metadata that is applicable to objects that have an explicit or implicit geographic extent. While using GeoTIFF files it is possible to define a metadata field in which information can be recorded. Metadata can be edited with most GIS software and directly with GDAL tools (http://www.gdal.org/gdal_edit.html).

> Tips for **GDAL**: *gdalwarp -t_srs* 'xxx' `input.tif output.tif` where *t_srs* is the target spatial reference set, i.e. the coordinate systems that can be passed are anything supported by the `GRSpatialReference.SetFromUserInput()` call, which includes EPSG PCS and GCSes (i.e. EPSG:4326), PROJ.4 declarations or the name of a `*.prj` file containing well known text. For further information see the link: http://www.gdal.org/gdalwarp.html.

Chapter 11

Technical overview and the checklist

V.L. Mulder

11.1 Point dataset

- Did you remove non-georeferenced observations from your dataset?
- Did you check for outliers or any unusual values for the measured SOC, pH, BD, stoniness/gravel content, and min/max definitions of your soil horizons?
- Is there spatial correlation in your SOC values, as observed from the variogram?
- Did you check the probability distribution and applied a transformation in case the samples were not normally distributed?
- Be aware whether you are going to predict SOC or SOM values!

11.2 Covariates

- Did you choose and apply the proper projection, one that is suitable for your country and is suitable for spatial statistics?
- Do all the covariates have a resolution of 1 km and did you use either nearest neighbor resampling for the categorical variables and IDW/cubic spline for continuous data?
- Did you correctly set the `NoData` values as not available data, i.e. not a standard assigned values such as -9999, 256, etc. or `NA`value.
- Did you check for outliers or any unusual values, especially in your DEM layer(s)?

- Did you set any categorical dataset as `factor` instead of being `numeric` or `integer`?

11.3 Statistical inference

- Did you choose a proper model which is capable to model the variability in your SOC point data best (multiple regression or data mining with or without interpolation of the residuals using kriging)?
- Did you make sure that the random forest did not over fit your data?
- Did you apply a validation scheme, e.g. k-fold cross-validation or an independent validation if so report the R^2 and RMSE as accuracy measures?
- Do the model summaries make sense, i.e. did most important predictor variables and model fit?
- Is there spatial structure left in your residuals, if so make sure you interpolate the model residuals using kriging?

11.4 Spatial interpolation

- Did you obtain an exhaustive map or are there still gaps? If so, check if your raster has the correct `factor` values, if the model was calibrated on fewer classes than the extrapolation may not work for those pixels?
- Do the patterns make sense or is there a covariate that causes an unrealistic pattern, based on expert judgment. If so, consider removing this covariate?
- In case you did kriging, do not forget to look at the kriging variance. This is a very important indicator of the accuracy. Otherwise, consider modeling the 90% confidence intervals of the predictions!
- Do not forget to back-transform your predicted values!

11.5 Calculation of stocks

- If you might want to calculate stocks per land use type, management type or by municipality, make sure you give an informed number, i.e. an estimate plus an estimate of the uncertainty!

11.6 Evaluation of output and quality assessment

- Report the model calibration and validation statistics!
- Report some map quality measures!
- Evaluate to which extent the model and map is either underestimating or overestimating SOC/SOM!

- Describe the spatial patterns and relate them to the landscape characteristics!

Chapter 12

Compendium of the code examples

G.F. Olmedo

This technical manual gives relatively simple step-by-step guidance on DSM. Guidance is mainly on mapping and modelling of soil organic carbon, but also contains generic sections on soil grid development that can be used to map various soil properties. The document follows the typical DSM work-flow and provides users a complete soil property mapping solution from the ground up (see Fig. 1.1).

The authors have demonstrated the methods in the previous Chapters using a dataset extracted from the Macedonian Soil Information System (MASIS). In this Chapter we present a compendium of the technical steps presented in the practical chapters.

With the help of these examples, the user should be able to produce a soil property map starting from ground data to the validation of the map. There are also optional work-flow steps depending on the nature of the input data. The **framework** presented in this manual is aiming to help users preparing a soil property prediction map includes soil data preparation, preparation of environmental covariates, overlaying soil data and covariates, fitting a model for the spatial interpolation and validation:

1. Soil data preparation

- **Option A**: Data preparation - Soil profiles (Code 12.1)
- **Option B**: Data preparation - Topsoil or auger samples (Code 12.2)
- *[optional]* Data preparation - Merging topsoil or auger samples and soil profiles (Code 12.3)
- *[optional]* Split the soil data in test and validations datasets (Code 12.4)

2. Covariates preparation

- *[optional]* Rasterizing a vector layer in **R** (Code 12.5)

- Overlay covariates and soil points data (Code 12.6)

3. Spatial interpolation model

- **Option A**: Fitting a regression-kriging model to predict the SOC stock (Code 12.7)
- **Option B**: Fitting a random forest model to predict the SOC stock (Code 12.9)
- **Option C**: Fitting a support vector machines model to predict the SOC stock (Code 12.11)

4. Validation

- Quality measures for quantitative data (Code 12.12)
- *[optional]* Graphical quality measures for quantitative data (Code 12.13)
- *[optional]* Cross-validation for regression-kriging models (Code 12.8)
- *[optional]* Validation of random forest using quantile regression trees (Code 12.10)

5. Model evaluation

- Model evaluation (Code 12.14)

12.1 Data preparation for soil profiles

The extended and discussed version of the following code is presented in Chapter 4, by G.F. Olmedo & R. Baritz.

```
dat <- read.csv(file = "data/horizons.csv")

# Explore the data
str(dat)
summary(dat)

dat_sites <- read.csv(file = "data/site-level.csv")

# Explore the data
str(dat_sites)

# summary of column CRF (Coarse Fragments) in the example data base
summary(dat$CRF)

# Convert NA's to 0
dat$CRF[is.na(dat$CRF)] <- 0

hist(dat$CRF)

# Creating a function in R to estimate BLD using the SOC
# SOC is the soil organic carbon content in \%
estimateBD <- function(SOC, method="Saini1996"){
  OM <- SOC * 1.724
  if(method=="Saini1996"){BD <- 1.62 - 0.06 * OM}
  if(method=="Drew1973"){BD <- 1 / (0.6268 + 0.0361 * OM)}
  if(method=="Jeffrey1979"){BD <- 1.482 - 0.6786 * (log(OM))}
  if(method=="Grigal1989"){BD <- 0.669 + 0.941 * exp(1)^(-0.06 * OM)}
  if(method=="Adams1973"){BD <- 100 / (OM /0.244 + (100 - OM)/2.65)}
  if(method=="Honeyset_Ratkowsky1989"){BD <- 1/(0.564 + 0.0556 * OM)}
  return(BD)
}

# summary of BLD (bulk density) in the example data base
summary(dat$BLD)

# See the summary of values produced using the pedo-transfer
# function with one of the proposed methods.
summary(estimateBD(dat$SOC[is.na(dat$BLD)], m
                   ethod="Honeyset_Ratkowsky1989"))

# Fill NA's using the pedotransfer function:
dat$BLD[is.na(dat$BLD)] <- estimateBD(dat$SOC[is.na(dat$BLD)],
```

```
                                        method="Grigal1989")

# explore the results
boxplot(dat$BLD)

# Load aqp package
library(aqp)

# Promote to SoilProfileCollection
# The SoilProfileCollection is a object class in R designed to
# handle soil profiles
depths(dat) <- ProfID ~ top + bottom

# Merge the soil horizons information with the site-level
# information from dat_sites
site(dat) <- dat_sites

# Set spatial coordinates
coordinates(dat) <- ~ X + Y

# A summary of our SoilProfileCollection
dat

library(GSIF)

## Estimate 0-30 standard horizon usin mass preserving splines
try(SOC <- mpspline(dat, 'SOC', d = t(c(0,30))))
try(BLD <- mpspline(dat, 'BLD', d = t(c(0,30))))
try(CRFVOL <- mpspline(dat, 'CRF', d = t(c(0,30))))

## Prepare final data frame
dat <- data.frame(id = dat@site$ProfID,
                  Y = dat@sp@coords[,2],
                  X = dat@sp@coords[,1],
                  SOC = SOC$var.std[,1],
                  BLD = BLD$var.std[,1],
                  CRFVOL = CRFVOL$var.std[,1])

dat <- dat[complete.cases(dat),]

## Take a look to the results
head(dat)

# Estimate Organic Carbon Stock
# SOC must be in g/kg
# BLD in kg/m3
```

```r
# CRF in percentage
OCSKGM <- OCSKGM(ORCDRC = dat$SOC, BLD = dat$BLD*1000,
                 CRFVOL = dat$CRFVOL, HSIZE = 30)

dat$OCSKGM <- OCSKGM
dat$meaERROR <- attr(OCSKGM,"measurementError")
dat <- dat[dat$OCSKGM>0,]
summary(dat)

## We can save our processed data as a table
write.csv(dat, "data/dataproc.csv")
```

12.2 Data preparation for topsoil or auger samples

The extended and discussed version of the following code is presented in Chapter 4, by G.F. Olmedo & R. Baritz.

```
dat <- read.csv(file = "data/auger.csv")

# Explore the data
str(dat)
summary(dat)

# Creating a function in R to estimate BLD using the SOC
# SOC is the soil organic carbon content in \%
estimateBD <- function(SOC, method="Saini1996"){
  OM <- SOC * 1.724
  if(method=="Saini1996"){BD <- 1.62 - 0.06 * OM}
  if(method=="Drew1973"){BD <- 1 / (0.6268 + 0.0361 * OM)}
  if(method=="Jeffrey1979"){BD <- 1.482 - 0.6786 * (log(OM))}
  if(method=="Grigal1989"){BD <- 0.669 + 0.941 * exp(1)^(-0.06 * OM)}
  if(method=="Adams1973"){BD <- 100 / (OM /0.244 + (100 - OM)/2.65)}
  if(method=="Honeyset_Ratkowsky1989"){BD <- 1/(0.564 + 0.0556 * OM)}
  return(BD)
}

# See the summary of values produced using the pedo-transfer
# function with one of the proposed methods.
summary(estimateBD(dat$SOC, method="Honeyset_Ratkowsky1989"))

# Estimate BLD using the pedotransfer function:
dat$BLD <- estimateBD(dat$SOC, method="Grigal1989")

# explore the results
boxplot(dat$BLD)

# Remove points with NA's values
dat <- dat[complete.cases(dat),]

## Take a look to the results
head(dat)

library(GSIF)
# Estimate Organic Carbon Stock
# SOC must be in g/kg
# BLD in kg/m3
# CRF in percentage
OCSKGM <- OCSKGM(ORCDRC = dat$SOC, BLD = dat$BLD*1000, CRFVOL = 0,
                 HSIZE = 30)
```

```
dat$OCSKGM <- OCSKGM
dat$meaERROR <- attr(OCSKGM,"measurementError")
dat <- dat[dat$OCSKGM>0,]
summary(dat)

## We can save our processed data as a table
write.csv(dat, "data/dataproc.csv")
```

12.3 Merging topsoil or auger samples and soil profiles databases

```
profiles <- read.csv("data/dataproc_profiles.csv")
topsoils <- read.csv("data/dataproc.csv")


# column names could be different, but the units and order has
# to be the same! Because we are going to add the rows from 1
# table to the other table.

topsoils <- topsoils[, c("ID", "X", "Y", "SOC", "BLD",
                         "OCSKGM", "meaERROR")]

profiles <- profiles[, c("id", "X", "Y", "SOC", "BLD",
                         "OCSKGM", "meaERROR")]

names(profiles) <- names(topsoils)
dat <- rbind(topsoils, profiles)

write.csv(dat, "data/dataproc_all.csv", row.names = F)
```

12.4 Data-splitting

The extended and discussed version of the following code is presented in Chapter 7, by B. Kempen, D.J. Brus & G.B.M. Heuvelink, with code contributions from G.F. Olmedo.

```
library(caret)

dat <- read.csv("data/dataproc.csv")

train.ind <- createDataPartition(1:nrow(dat), p = .75, list = FALSE)
train <- dat[ train.ind,]
test  <- dat[-train.ind,]

plot(density (log(train$OCSKGM)), col='red',
     main='Statistical distribution of train and test datasets')
lines(density(log(test$OCSKGM)), col='blue')
legend('topright', legend=c("train", "test"),
       col=c("red", "blue"), lty=1, cex=1.5)

write.csv(train, file="data/dat_train.csv", row.names = FALSE)
write.csv(test, file="data/dat_test.csv", row.names = FALSE)
```

12.5 Rasterizing a vector layer in R

The extended and discussed version of the following code is presented in Chapter 5, by R. Baritz & Y. Yigini.

```
library (raster)
soilmap <- shapefile("MK_soilmap_simple.shp")
DEM <- raster("covs/DEMENV5.tif")

# the "Symbol" attribute from the vector layer will be used for the
# rasterization process. It has to be a factor
soilmap@data$Symbol <- as.factor(soilmap@data$Symbol)

#save the levels names in a character vector
Symbol.levels <- levels(soilmap$Symbol)

# The rasterization process needs a layer with the target grd
# system: spatial extent and cell size.
soilmap.r <- rasterize(x = soilmap, y = DEM, field = "Symbol")
# The DEM raster layer could be used for this.

plot(soilmap.r, col=rainbow(21))
legend("bottomright",legend = Symbol.levels, fill=rainbow(21),
       cex=0.5)
```

12.6 Overlay covariates and soil points data

The extended and discussed version of the following code is presented in Chapter 5, by R. Baritz & Y. Yigini.

```
# Load the processed data. This table was prepared in the previous
# chapter.
dat <- read.csv("data/dataproc.csv")

files <- list.files(path = "covs", pattern = "tif$",
                    full.names = TRUE)

covs <- stack(files)

#covs <- stack(covs, soilmap.r)

# correct the name for layer 14
#names(covs)[14] <- "soilmap"

library (raster)

#mask the covariates with the country mask from the data repository
mask <- raster("data/mask.tif")

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

# export all the covariates
save(covs, file = "covariates.RData")

plot(covs)

#upgrade points data frame to SpatialPointsDataFrame
coordinates(dat) <- ~ X + Y

# extract values from covariates to the soil points
dat <- extract(x = covs, y = dat, sp = TRUE)

# LCEE10 and soilmap are categorical variables
dat@data$LCEE10 <- as.factor(dat@data$LCEE10)

#dat@data$soilmap <- as.factor(dat@data$soilmap)
#levels(soilmap) <- Symbol.levels

summary(dat@data)

dat <- as.data.frame(dat)
```

```
# The points with NA values has to be removed
dat <- dat[complete.cases(dat),]

# export as a csv table
write.csv(dat, "data/MKD_RegMatrix.csv", row.names = FALSE)
```

12.7 Fitting a regression-kriging model to predict the SOC stock

The extended and discussed version of the following code is presented in Section 6.2, by G.F. Olmedo & Y. Yigini.

```
# load data
dat <- read.csv("data/MKD_RegMatrix.csv")

# explore the data structure
str(dat)

library(sp)

# Promote to spatialPointsDataFrame
coordinates(dat) <- ~ X + Y

class(dat)

dat@proj4string <- CRS(projargs = "+init=epsg:4326")

dat@proj4string

library(raster)

load(file = "covariates.RData")

names(covs)

# print the names of the 14 layers:
names(covs)

datdf <- dat@data

datdf <- datdf[, c("OCSKGM", names(covs))]

# Fit a multiple linear regression model between the log transformed
# values of OCS and the top 20 covariates
model.MLR <- lm(log(OCSKGM) ~ ., data = datdf)

# stepwise variable selection
model.MLR.step <- step(model.MLR, direction="both")

# summary and anova of the new model using stepwise covariates
# selection
summary(model.MLR.step)
anova(model.MLR.step)
```

```
# graphical diagnosis of the regression analysis
par(mfrow=c(2,2))
plot(model.MLR.step)
par(mfrow=c(1,1))

# collinearity test using variance inflation factors
library(car)
vif(model.MLR.step)

# problematic covariates should have sqrt(VIF) > 2
sqrt(vif(model.MLR.step))

# Removing B07CHE3 from the stepwise model:
model.MLR.step <- update(model.MLR.step, . ~ . - DEMSRE3a - PX3WCL3a)

# Test the vif again:
sqrt(vif(model.MLR.step))

## summary  of the new model using stepwise covariates selection
summary(model.MLR.step)

# outlier test using the Bonferroni test
outlierTest(model.MLR.step)

# Project point data.
dat <- spTransform(dat, CRS("+init=epsg:32648"))

# project covariates to VN-2000 UTM 48N
covs <- projectRaster(covs, crs = CRS("+init=epsg:32648"),
                      method='ngb')


# Promote covariates to spatial grid dataframe. Takes some time and
# a lot of memory!
covs.sp <- as(covs, "SpatialGridDataFrame")

# RK model
library(automap)


# Run regression kriging prediction. This step can take hours...!
OCS.krige <- autoKrige(formula =
                         as.formula(model.MLR.step$call$formula),
                       input_data = dat,
                       new_data = covs.sp,
                       verbose = TRUE,
```

```
                        block = c(1000, 1000))

OCS.krige

# Convert prediction and standard deviation to rasters
# And back-tansform the vlaues
RKprediction <- exp(raster(OCS.krige$krige_output[1]))
RKpredsd <- exp(raster(OCS.krige$krige_output[3]))


plot(RKprediction)

## Save results as tif files
writeRaster(RKprediction, filename = "results/MKD_OCSKGM_RK.tif",
            overwrite = TRUE)

writeRaster(RKpredsd, filename = "results/MKD_OCSKGM_RKpredsd.tif",
            overwrite = TRUE)

# save the model
saveRDS(model.MLR.step, file="results/RKmodel.Rds")
```

12.8 Cross-validation of regression-kriging models

The extended and discussed version of the following code is presented in Section 6.2, by G.F. Olmedo & Y. Yigini.

```
# autoKrige.cv() does not removes the duplicated points
# we have to do it manually before running the cross-validation
dat = dat[which(!duplicated(dat@coords)), ]

OCS.krige.cv <- autoKrige.cv(formula =
                               as.formula(model.MLR.step$call$formula),
                             input_data = dat, nfold = 10)

summary(OCS.krige.cv)
```

12.9 Fitting a random forest model to predict the SOC stock

The extended and discussed version of the following code is presented in Section 6.3, by M. Guevara, C. Thine, G.F. Olmedo & R.R. Vargas.

```
dat <- read.csv("data/MKD_RegMatrix.csv")

dat$LCEE10 <- as.factor(dat$LCEE10)
#dat$soilmap <- as.factor(dat$soilmap)

# explore the data structure
str(dat)

library(sp)

# Promote to spatialPointsDataFrame
coordinates(dat) <- ~ X + Y

class(dat)

dat@proj4string <- CRS(projargs = "+init=epsg:4326")

dat@proj4string

load(file = "covariates.RData")

names(covs)

# For its use on R we need to define a model formula

fm = as.formula(paste("log(OCSKGM) ~", paste0(names(covs[[-14]]),
                                              collapse = "+")))

library(randomForest)
library(caret)

# Default 10-fold cross-validation
ctrl <- trainControl(method = "cv", savePred=T)
# Search for the best mtry parameter
rfmodel <- train(fm, data=dat@data, method = "rf", trControl = ctrl,
                 importance=TRUE)
# This is a very useful function to compare and test different
# prediction algorithms type names(getModelInfo()) to see all the
# possibilitites implemented on this function
```

```
# Variable importance plot, compare with the correlation matrix
# Select the best prediction factors and repeat
varImpPlot(rfmodel[11][[1]])

# Check if the error stabilizes
plot(rfmodel[11][[1]])

#Make a prediction across all Macedonia
#Note that the units are still in log
pred <- predict(covs, rfmodel)

# Back transform predictions log transformed
pred <- exp(pred)

# Save the result as a tiff file
writeRaster(pred, filename = "results/MKD_OCSKGM_rf.tif",
            overwrite=TRUE)


plot(pred)
```

12.10 Using quantile regression forest to estimate uncertainty

The extended and discussed version of the following code is presented in Section 6.3, by M. Guevara, C. Thine, G.F. Olmedo & R.R. Vargas.

```
library(Metrics)

#Generate an empty dataframe
validation <- data.frame(rmse=numeric(), r2=numeric())
#Sensitivity to the dataset
#Start a loop with 10 model realizations
for (i in 1:10){
  # We will build 10 models using random samples of 25%
  smp_size <- floor(0.25 * nrow(dat))
  train_ind <- sample(seq_len(nrow(dat)), size = smp_size)
  train <- dat[train_ind, ]
  test <- dat[-train_ind, ]
  modn <- train(fm, data=train@data, method = "rf", trControl = ctrl)
  pred <- stack(pred, predict(covs, modn))
  test$pred <- extract(pred[[i+1]], test)
  # Store the results in a dataframe
  validation[i, 1] <- rmse(log(test$OCSKGM), test$pred)
  validation[i, 2] <- cor(log(test$OCSKGM), test$pred)^2
}

#The sensitivity map is the dispersion of all individual models
sensitivity <- calc(pred[[-1]], sd)

plot(sensitivity, col=rev(topo.colors(10)),
     main='Sensitivity based on 10 realizations using 25% samples')

#Sensitivity of validation metrics
summary(validation)

# Plot of the map based on 75% of data and the sensitivity to data
# variations
prediction75 <- exp(pred[[1]])

plot(prediction75, main='OCSKGM prediction based on 75% of data',
     col=rev(topo.colors(10)))

# Use quantile regression forest to estimate the full conditional
# distribution of OCSKGMlog, note that we are using the mtry
# parameter that was selected by the train funtion of the caret
# package, assuming that the 75% of data previously used well
```

```
# resembles the statistical distribution of the entire data
# population. Otherwise repeat the train function with all available
#data (using the object dat that instead of train) to select mtry.

library(quantregForest)

model <- quantregForest(y=log(dat@data$OCSKGM), x=dat@data[,8:20], ntree=500,
                        keep.inbag=TRUE,
                        mtry = as.numeric(modn$bestTune))

library(snow)
# Estimate model uncertainty at the pixel level using parallel
# computing
beginCluster() #define number of cores to use
# Estimate model uncertainty
unc <- clusterR(covs, predict, args=list(model=model,what=sd))
# OCSKGMlog prediction based in all available data
mean <- clusterR(covs, predict,
                 args=list(model=model, what=mean))
# The total uncertainty is the sum of sensitivity and model
# uncertainty
unc <- unc + sensitivity
# Express the uncertainty in percent (divide by the mean)
Total_unc_Percent <- exp(unc)/exp(mean)
endCluster()

# Plot both maps (the predicted OCSKGM + its associated uncertainty)
plot(exp(mean), main='OCSKGM based in all data',
     col=rev(topo.colors(10)))

plot(Total_unc_Percent, col=rev(heat.colors(100)), zlim=c(0, 5),
     main='Total uncertainty')

#Save the resulting maps in separated *.tif files
writeRaster(exp(mean), file='rfOCSKGMprediction.tif',
            overwrite=TRUE)
writeRaster(Total_unc_Percent, file='rfOCSKGMtotalUncertPercent.tif',
            overwrite=TRUE)
```

12.11 Fitting a support vector machines model to predict the SOC stock

The extended and discussed version of the following code is presented in Section 6.4, by G.F. Olmedo & M. Guevara.

```
# load data
dat <- read.csv("data/MKD_RegMatrix.csv")

dat$LCEE10 <- as.factor(dat$LCEE10)
# dat$soilmap <- as.factor(dat$soilmap)

# explore the data structure
str(dat)

library(sp)

# Promote to spatialPointsDataFrame
coordinates(dat) <- ~ X + Y

class(dat)

dat@proj4string <- CRS(projargs = "+init=epsg:4326")

dat@proj4string

load(file = "covariates.RData")

names(covs)

# plot the names of the covariates
names(dat@data)

# variable selection using correlation analysis
selectedCovs <- cor(x = as.matrix(dat@data[,5]),
                    y = as.matrix(dat@data[,-c(1:7,13,21)]))

# print correlation results
selectedCovs

library(reshape)
x <- subset(melt(selectedCovs), value != 1 | value != NA)
x <- x[with(x, order(-abs(x$value))),]

idx <- as.character(x$X2[1:5])

dat2 <- dat[c('OCSKGM', idx)]
```

```
names(dat2)

COV <- covs[[idx]]

# Selected covariates
names(COV)


# Categorical variables in svm models
dummyRaster <- function(rast){
  rast <- as.factor(rast)
  result <- list()
  for(i in 1:length(levels(rast)[[1]][[1]])){
    result[[i]] <- rast == levels(rast)[[1]][[1]][i]
    names(result[[i]]) <- paste0(names(rast),
                                 levels(rast)[[1]][[1]][i])
  }
  return(stack(result))
}

# convert soilmap from factor to dummy
# soilmap_dummy <- dummyRaster(covs$soilmap)

# convert LCEE10 from factor to dummy
LCEE10_dummy <- dummyRaster(covs$LCEE10)

# Stack the 5 COV layers with the 2 dummies
COV <- stack(COV, LCEE10_dummy)
# COV <- stack(COV, soilmap_dummy, LCEE10_dummy)

# print the final layer names
names(COV)

# convert soilmap column to dummy, the result is a matrix
# to have one column per category we had to add -1 to the formula
# dat_soilmap_dummy <- model.matrix(~soilmap -1, data = dat@data)
# convert the matrix to a data.frame
# dat_soilmap_dummy <- as.data.frame(dat_soilmap_dummy)


# convert LCEE10 column to dummy, the result is a matrix
# to have one column per category we had to add -1 to the formula
dat_LCEE10_dummy <- model.matrix(~LCEE10 -1, data = dat@data)
# convert the matrix to a data.frame
dat_LCEE10_dummy <- as.data.frame(dat_LCEE10_dummy)
```

```
dat@data <- cbind(dat@data, dat_LCEE10_dummy)
# dat@data <- cbind(dat@data, dat_LCEE10_dummy, dat_soilmap_dummy)

names(dat@data)

# Fitting a svm model and parameter tuning
library(e1071)
library(caret)

#  Test different values of epsilon and cost
tuneResult <- tune(svm, OCSKGM ~.,  data = dat@data[,c("OCSKGM",
                                                       names(COV))],
                   ranges = list(epsilon = seq(0.1,0.2,0.02),
                                 cost = c(5,7,15,20)))


plot(tuneResult)

# Choose the model with the best combination of epsilon and cost
tunedModel <- tuneResult$best.model

print(tunedModel)


# Use the model to predict the SOC in the covariates space
OCSsvm <- predict(COV, tunedModel)

# Save the result
writeRaster(OCSsvm, filename = "results/MKD_OCSKGM_svm.tif",
            overwrite=TRUE)

plot(OCSsvm)

# Variable importance in svm. Code by:
# stackoverflow.com/questions/34781495

w <- t(tunedModel$coefs) %*% tunedModel$SV      # weight vectors
w <- apply(w, 2, function(v){sqrt(sum(v^2))})  # weight

w <- sort(w, decreasing = T)
print(w)
```

12.12 Validation

The extended and discussed version of the following code is presented in Chapter 7, by B. Kempen, D.J. Brus & G.B.M. Heuvelink, with code contributions from G.F. Olmedo.

```
dat <- read.csv("data/dat_test.csv")

# Promote to spatialPointsDataFrame
coordinates(dat) <- ~ X + Y

dat@proj4string <- CRS(projargs = "+init=epsg:4326")

library(raster)

OCSKGM_rf <- raster("results/MKD_OCSKGM_rf.tif")

dat <- extract(x = OCSKGM_rf, y = dat, sp = TRUE)

# prediction error
dat$PE_rf <- dat$MKD_OCSKGM_rf - dat$OCSKGM

# Mean Error
ME_rf <- mean(dat$PE_rf, na.rm=TRUE)

# Mean Absolute Error (MAE)
MAE_rf <- mean(abs(dat$PE_rf), na.rm=TRUE)

# Mean Squared Error (MSE)
MSE_rf <- mean(dat$PE_rf^2, na.rm=TRUE)

# Root Mean Squared Error (RMSE)
RMSE_rf <- sqrt(sum(dat$PE_rf^2, na.rm=TRUE) / length(dat$PE_rf))

# Amount of Variance Explained (AVE)
AVE_rf <- 1 - sum(dat$PE_rf^2, na.rm=TRUE) /
  sum( (dat$OCSKGM - mean(dat$OCSKGM, na.rm = TRUE))^2,
       na.rm = TRUE)
```

12.13 Graphical map quality measures

The extended and discussed version of the following code is presented in Chapter 7, by B. Kempen, D.J. Brus & G.B.M. Heuvelink, with code contributions from G.F. Olmedo.

```
# scatter plot
plot(dat$MKD_OCSKGM_rf, dat$OCSKGM, main="rf", xlab="predicted",
     ylab='observed')
# 1:1 line in black
abline(0,1, lty=2, col='black')
# regression line between predicted and observed in blue
abline(lm(dat$OCSKGM ~ dat$MKD_OCSKGM_rf), col = 'blue', lty=2)

# spatial bubbles for prediction errors
bubble(dat[!is.na(dat$PE_rf),], "PE_rf", pch = 21,
       col=c('red', 'green'))
```

12.14 Model evaluation

The extended and discussed version of the following code is presented in Chapter 8, by M. Guevara and G.F. Olmedo.

```
library(raster)
RF<-raster('results/MKD_OCSKGM_rf.tif')
RK<-raster('results/MKD_OCSKGM_RK.tif')
SVM<-raster('results/MKD_OCSKGM_svm.tif')
#Note that RK has a different reference system
RK <- projectRaster(RK, SVM)
models <- stack(RK, RF, SVM)

library(psych)
pairs.panels(na.omit(as.data.frame(models)),
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
)


library(rasterVis)
densityplot(models)

SD <- calc(models , sd)
library(mapview)
mapview(SD)

RKRF  <- calc(models[[c(1,2)]], diff)
RKSVM <- calc(models[[c(1,3)]], diff)
RFSVM <- calc(models[[c(2,3)]], diff)
preds <- stack(RKRF, RKSVM, RFSVM)
names(preds) <- c('RKvsRF','RKvsSVM','RFvsSVM')
X <- cellStats(preds, mean)
levelplot(preds - X, at=seq(-0.5,0.5, length.out=10),
          par.settings = RdBuTheme)

dat <- read.csv("results/validation.csv")

# prepare 3 new data.frame with the observed, predicted and the model
modRK <- data.frame(obs = dat$OCSKGM, mod = dat$MKD_OCSKGM_RK,
                    model = "RK")

modRF <- data.frame(obs = dat$OCSKGM, mod = dat$MKD_OCSKGM_rf,
                    model = "RF")
```

```
modSVM <- data.frame(obs = dat$OCSKGM, mod = dat$MKD_OCSKGM_svm,
                     model = "SVM")

# merge the 3 data.frames into one
modData <- rbind(modRK, modRF, modSVM)

summary(modData)

#Load the openair library
library(openair)

modsts <- modStats(modData,obs = "obs", mod = "mod", type = "model")

modsts

TaylorDiagram(modData, obs = "obs", mod = "mod", group = "model",
              cols = c("orange", "red","blue"), cor.col='brown',
              rms.col='black')

conditionalQuantile(modData,obs = "obs", mod = "mod", type = "model")
```

Bibliography

Adams, W. (1973). The effect of organic matter on the bulk and true densities of some uncultivated podzolic soils. *European Journal of Soil Science*, 24(1):10–17.

Adhikari, K., Minasny, B., Greve, M. B., and Greve, M. H. (2014). Constructing a soil class map of denmark based on the fao legend using digital techniques. *Geoderma*, 214:101–113.

Agus, F., Hairiah, K., and Mulyani, A. (2011). Measuring carbon stock in peat soils: practical guidelines | world agroforestry centre.

Angelini, M. E., Heuvelink, G. B., Kempen, B., and Morrás, H. J. (2016). Mapping the soils of an argentine pampas region using structural equation modelling. *Geoderma*, 281:102–118.

Arrouays, D., Deslais, W., and Badeau, V. (2001). The carbon content of topsoil and its geographical distribution in france. *Soil use and Management*, 17(1):7–11.

Baritz, R., Eberhardt, E., Van Liedekerke, M., and Panagos, P. (2008). Environmental assessment of soil for monitoring: Volume iii database design and selection. *EUR*, 23490:125.

Barney, R. J., Bevins, C. D., Bradshaw, L. S., Forestry Sciences Laboratory (Missoula, M. ., United States. Forest Service, and Intermountain Forest and Range Experiment Station (Ogden, U. (1981). *Forest floor fuel loads, depths and bulk densities in four interior Alaskan cover types*. Ogden, Utah : U.S. Dept. of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station.

Bernoux, M., Arrouays, D., Cerri, C. C., and Bourennane, H. (1998). Modeling vertical distribution of carbon in oxisols of the western brazilian amazon (rondonia). *Soil Science*, 163(12):941–951.

Bhatti, J. S., Apps, M. J., and Tarnocai, C. (2002). Estimates of soil organic carbon stocks in central canada using three different approaches. *Canadian Journal of Forest Research*, 32(5):805–812.

Bishop, T., McBratney, A., and Laslett, G. (1999). Modelling soil attribute depth functions with equal-area quadratic smoothing splines. *Geoderma*, 91(1):27–45.

Bivand, R. S., Pebesma, E., and Gómez-Rubio, V. (2013). Classes for spatial data in r. In *Applied Spatial Data Analysis with R*, pages 21–57. Springer.

Bonfatti, B. R., Hartemink, A. E., Giasson, E., Tornquist, C. G., and Adhikari, K. (2016). Digital mapping of soil carbon in a viticultural region of Southern Brazil. *Geoderma*, 261:204–221.

Bontemps, S., Defourny, P., Bogaert, E. V., Arino, O., Kalogirou, V., and Perez, J. R. (2011). Globcover 2009-products description and validation report.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine learning*, pages 5–32.

Breiman, L. (2017). Cutler's random forests for classification and regression. 2015. r package version 4.6. 12.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Brus, D., Kempen, B., and Heuvelink, G. (2011). Sampling for validation of digital soil maps. *European Journal of Soil Science*, 62(3):394–407.

Carslaw, D. (2015). The openair manual–open-source tools for analysing air pollution data. manual for version 1.1–4, king's college london.

Carslaw, D. C. and Ropkins, K. (2012). openair — an r package for air quality data analysis. *Environmental Modelling & Software*, 27–28(0):52–61.

Chang, C.-C. and Lin, C.-J. (2001). Libsvm: A library for support vector machines [eb/ol].

Chang, J. and Hanna, S. (2004). Air quality model performance evaluation. *Meteorology and Atmospheric Physics*, 87(1-3):167–196.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cressie, N. A. (1993). Statistics for spatial data: Wiley series in probability and mathematical statistics. *Find this article online*.

De Gruijter, J., Brus, D. J., Bierkens, M. F., and Knotters, M. (2006). *Sampling for natural resource monitoring*. Springer Science & Business Media.

Dobos, E. (2006). *Digital Soil Mapping: As a Support to Production of Functional Maps*. Office for Official Publication of the European Communities.

Drew, L. A. (1973). Bulk density estimation based on organic matter content of some minnesota soils.

FAO (2017). *Soil Organic Carbon Mapping Cookbook*, volume 1st edition.

FAO and GSP (2017a). Gsp guidelines for sharing national data/information to compile a global soil organic carbon (gsoc) map - pillar 4 working group.

FAO and GSP (2017b). Gsp soil data policy.

Florinsky, I. V. (2012). The dokuchaev hypothesis as a basis for predictive digital soil mapping (on the 125th anniversary of its publication). *Eurasian Soil Science*, 45(4):445–451.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford University Press on Demand.

Goovaerts, P. (2001). Geostatistical modelling of uncertainty in soil science. *Geoderma*, 103(1):3–26.

Greve, M. H., Greve, M. B., Bøcher, P. K., Balstrøm, T., Breuning-Madsen, H., and Krogh, L. (2007). Generating a danish raster-based topsoil property map combining choropleth maps and point information. *Geografisk Tidsskrift-Danish Journal of Geography*, 107(2):1–12.

Grigal, D., Brovold, S., Nord, W., and Ohmann, L. (1989). Bulk density of surface soils and peat in the north central united states. *Canadian Journal of Soil Science*, 69(4):895–900.

Grimm, R. and Behrens, T. (2010). Uncertainty analysis of sample locations within digital soil mapping approaches. *Geoderma*, 155(3):154–163.

Guevara, M., Olmedo, G. F., Stell, E., Yigini, Y., Aguilar Duarte, Y., Arellano Hernández, C., Arévalo, G. E., Arroyo-Cruz, C. E., Bolivar, A., Bunning, S., Bustamante Cañas, N., Cruz-Gaistardo, C. O., Davila, F., Dell Acqua, M., Encina, A., Figueredo Tacona, H., Fontes, F., Hernández Herrera, J. A., Ibelles Navarro, A. R., Loayza, V., Manueles, A. M., Mendoza Jara, F., Olivera, C., Osorio Hermosilla, R., Pereira, G., Prieto, P., Alexis Ramos, I., Rey Brina, J. C., Rivera, R., Rodríguez-Rodríguez, J., Roopnarine, R., Rosales Ibarra, A., Rosales Riveiro, K. A., Schulz, G. A., Spence, A., Vasques, G. M., Vargas, R. R., and Vargas, R. (2018). No silver bullet for digital soil mapping: Country-specific soil organic carbon estimates across latin america. *SOIL Discussions*, 2018:1–20.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

Hartmann, J. and Moosdorf, N. (2012). The new global lithological map database glim: A representation of rock properties at the earth surface. *Geochemistry, Geophysics, Geosystems*, 13(12).

Hengl, T. (2006). Finding the right pixel size. *Computers & Geosciences*, 32(9):1283–1298.

Hengl, T. (2016). Gsif: Global soil information facilities. r package version 0.5-3.

Hengl, T., de Jesus, J. M., MacMillan, R. A., Batjes, N. H., Heuvelink, G. B., Ribeiro, E., Samuel-Rosa, A., Kempen, B., Leenaars, J. G., Walsh, M. G., et al. (2014). Soilgrids1km—global soil information based on automated mapping. *PLoS One*, 9(8):e105992.

Hengl, T., Heuvelink, G. B., Kempen, B., Leenaars, J. G., Walsh, M. G., Shepherd, K. D., Sila, A., MacMillan, R. A., de Jesus, J. M., Tamene, L., et al. (2015a). Mapping soil properties of africa at 250 m resolution: random forests significantly improve current predictions. *PloS one*, 10(6):e0125814.

Hengl, T., Heuvelink, G. B., and Rossiter, D. G. (2007). About regression-kriging: from equations to case studies. *Computers & geosciences*, 33(10):1301–1315.

Hengl, T., Heuvelink, G. B., and Stein, A. (2004). A generic framework for spatial prediction of soil variables based on regression-kriging. *Geoderma*, 120(1):75–93.

Hengl, T., Roudier, P., Beaudette, D., Pebesma, E., et al. (2015b). plotkml: scientific visualization of spatio-temporal data. *Journal of Statistical Software*, 63(5):1–25.

Hengl T., Heuvelink G.B.M., K. B. (2017). Methods to fit a regression-kriging model.

Heuvelink, G. (2014). Uncertainty quantification of globalsoilmap products. in 'globalsoilmap, basis of the global spatial soil information system'.(eds d arrouays, nj mckenzie, jw hempel, ar de forges, ab mcbratney) pp. 327–332.

Hiederer, R. (2013). Mapping soil properties for europe—spatial representation of soil database attributes. *Publications Office of the European Union, EUR26082EN Scientific and Technical Research series. Luxembourg*.

Hijmans, R. J., Cameron, S. E., Parra, J. L., Jones, P. G., and Jarvis, A. (2005). Very high resolution interpolated climate surfaces for global land areas. *International journal of climatology*, 25(15):1965–1978.

Ho, Y.-C. and Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115(3):549–570.

Honeysett, J. and Ratkowsky, D. (1989). The use of ignition loss to estimate bulk density of forest soils. *European Journal of Soil Science*, 40(2):299–308.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification.

Jackson, I. (2007). Onegeology: Making geological map data for the earth accessible. *Episodes*, 30(1):60–61.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Jeffrey, D. (1970). A note on the use of ignition loss as a means for the approximate estimation of soil bulk density. *The Journal of Ecology*, pages 297–299.

Karatzoglou, A., Meyer, D., and Hornik, K. (2006). Support Vector Algorithm in R. *Journal of Statistical Software*, 15(9):1–28.

Kempen, B., Brus, D. J., Stoorvogel, J. J., Heuvelink, G., and de Vries, F. (2012). Efficiency comparison of conventional and digital soil mapping for updating soil maps. *Soil Science Society of America Journal*, 76(6):2097–2115.

Kempen, B., Heuvelink, G., Brus, D., and Stoorvogel, J. (2010). Pedometric mapping of soil organic matter using a soil map with quantified uncertainty. *European journal of soil science*, 61(3):333–347.

Knotters, M. and Brus, D. (2013). Purposive versus random sampling for map validation: a case study on ecotope maps of floodplains in the netherlands. *Ecohydrology*, 6(3):425–434.

Kolli, R., Ellermäe, O., Köster, T., Lemetti, I., Asi, E., and Kauer, K. (2009). Stocks of organic carbon in estonian soils. *Estonian Journal of Earth Sciences*, 58(2).

Krasilnikov, P., del Carmen Gutiérrez-Castorena, M., Ahrens, R. J., Cruz-Gaistardo, C. O., Sedov, S., and Solleiro-Rebolledo, E. (2013). *The soils of Mexico*. Springer Science & Business Media.

Krause, P., Boyle, D., and Bäse, F. (2005). Comparison of different efficiency criteria for hydrological model assessment. *Advances in geosciences*, 5:89–97.

Kuhn, M. (2016). A short introduction to the caret package. *URL: https://cran. r-project. org/web/packages/caret/vignettes/caret. pdf*.

Kuhn, M., Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., et al. (2017). Caret: classification and regression training. 2016. *R package version*, 4.

Lark, R. (1995). Components of accuracy of maps with special reference to discriminant analysis on remote sensor data. *International Journal of Remote Sensing*, 16(8):1461–1480.

Lark, R. (2000). A comparison of some robust estimators of the variogram for use in soil survey. *European Journal of Soil Science*, 51(1):137–157.

Legates, D. R. and McCabe, G. J. (1999). Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation. *Water resources research*, 35(1):233–241.

Legates, D. R. and McCabe, G. J. (2013). A refined index of model performance: a rejoinder. *International Journal of Climatology*, 33(4):1053–1056.

Lettens, S., Orshoven, J., Wesemael, B., Muys, B., and Perrin, D. (2005). Soil organic carbon changes in landscape units of belgium between 1960 and 2000 with reference to 1990. *Global Change Biology*, 11(12):2128–2140.

Lettens, S., Orshoven, J. v., Wesemael, B., , and Muys, B. (2004). Soil organic and inorganic carbon contents of landscape units in belgium derived using data from 1950 to 1970. *Soil Use and Management*, 20(1):40–47.

McBratney, A. B., Santos, M. M., and Minasny, B. (2003). On digital soil mapping. *Geoderma*, 117(1):3–52.

Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun):983–999.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2017). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.6-8.

Nelson, D. and Sommers, L. (1982). Total carbon, organic carbon, and organic matter. In *Methods of Soil Analysis. Part 2. Chemical and Microbiological Properties*, pages 539–579. American Society of Agronomy, Soil Science Society of America.

Nelson, M., Bishop, T., Triantafilis, J., and Odeh, I. (2011). An error budget for different sources of error in digital soil mapping. *European Journal of Soil Science*, 62(3):417–430.

Neteler, M. and Mitasova, H. (2013). *Open source GIS: a GRASS GIS approach*, volume 689. Springer Science & Business Media.

Nussbaum, M., Spiess, K., Baltensweiler, A., Grob, U., Keller, A., Greiner, L., Schaepman, M. E., and Papritz, A. (2018). Evaluation of digital soil mapping approaches with large sets of environmental covariates. *Soil*, 4(1):1.

Ottmar, R. and Andreu, A. (2007). Litter and duff bulk densities in the southern united states: final report for joint fire sciences program, project 04-2-1-49.

Pebesma, E. J. and Bivand, R. S. (2005). Classes and methods for spatial data in r. *R news*, 5(2):9–13.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Poggio, L., Gimona, A., and Brewer, M. J. (2013). Regional scale mapping of soil properties and their uncertainty with a large number of satellite-derived covariates. *Geoderma*, 209:1–14.

Ponce Hernandes, R., Marriott, F., and Beckett, P. (1986). An improved method for reconstructing a soil profile from analyses of a small number of samples. *European Journal of Soil Science*, 37(3):455–467.

Pontius Jr, R. G. and Millones, M. (2011). Death to kappa: birth of quantity disagreement and allocation disagreement for accuracy assessment. *International Journal of Remote Sensing*, 32(15):4407–4429.

Qiao, H., Soberón, J., and Peterson, A. T. (2015). No silver bullets in correlative ecological niche modelling: insights from testing among many potential algorithms for niche estimation. *Methods in Ecology and Evolution*, 6(10):1126–1136.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rad, M. R. P., Toomanian, N., Khormali, F., Brungard, C. W., Komaki, C. B., and Bogaert, P. (2014). Updating soil survey maps using random forest and conditioned latin hypercube sampling in the loess derived soils of northern iran. *Geoderma*, 232:97–106.

Rosell, R., Gasparoni, J., and Galantini, J. (2001). Soil organic matter evaluation. In *Assessment methods for soil carbon*, pages 311–322. Lewis Publishers Boca Raton.

Saini, G. (1966). Organic matter as a measure of bulk density of soil. *Nature*, 210(5042):1295–1296.

Samuel-Rosa, A., Heuvelink, G., Vasques, G., and Anjos, L. (2015). Do more detailed environmental covariates deliver more accurate soil maps? *Geoderma*, 243:214–227.

Snedecor, G. and Cochran, W. (1989). Stadistical methods. eight ed. ames.

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89.

Stehman, S. V. (1999). Basic probability sampling designs for thematic map accuracy assessment. *International Journal of remote sensing*, 20(12):2423–2441.

Taylor, K. E. (2001). Summarizing multiple aspects of model performance in a single diagram. *Journal of Geophysical Research: Atmospheres*, 106(D7):7183–7192.

United States Department of Agriculture. Soil Conservation Service (1975). *Soil taxonomy: a basic system of soil classification for making and interpreting soil surveys*. Number 436. US Department of Agriculture.

United States Department of Agriculture. Soil Conservation Service (2018). Estimating moist bulk density by texture. general guide for estimating moist bulk density (one-tenth and one-third bar).

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vaysse, K. and Lagacherie, P. (2017). Using quantile regression forest to estimate uncertainty of digital soil mapping products. *Geoderma*, 291:55–64.

Voltz, M. and Webster, R. (1990). A comparison of kriging, cubic splines and classification for predicting soil properties from sample information. *European Journal of Soil Science*, 41(3):473–490.

Walkley, A. and Black, I. A. (1934). An examination of the degtjareff method for determining soil organic matter, and a proposed modification of the chromic acid titration method. *Soil science*, 37(1):29–38.

Warmerdam, F. (2008). The geospatial data abstraction library. In Hall, G. B. and Leahy, M. G., editors, *Open Source Approaches in Spatial Data Handling*, pages 87–104. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-74831-1_5.

Webster, R. and Oliver, M. A. (2007). *Geostatistics for environmental scientists. Statistics in practice. 2nd Edition.* John Wiley & Sons,.

Wiesmeier, M., Barthold, F., Blank, B., and Kögel-Knabner, I. (2011). Digital mapping of soil organic matter stocks using random forest modeling in a semi-arid steppe ecosystem. *Plant and soil*, 340(1-2):7–24.

Willmott, C. J., Robeson, S. M., and Matsuura, K. (2012). A refined index of model performance. *International Journal of Climatology*, 32(13):2088–2094.

Wong, W. V. C., Tsuyuki, S., Ioki, K., and Phua, M.-H. (2014). Accuracy Assessment of Global Topographic Data (SRTM & ASTER GDEM) in Comparison With LIDAR for Tropical Montane Forest.

Youngblood, B. (2013). *GeoServer Beginner's Guide*. Packt Publishing Ltd.

This cookbook was prepared thanks to the financial support of
The Kingdom of the Netherlands and the Swiss Confederation

Ministry of Foreign Affairs of the Netherlands

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

ISBN 978-92-5-130440-2

9 789251 304402

I8895EN/1/03.18