Short Training Initiative on Digital Soil Mapping in R Environment

Practical Workbook

May 2024

Title:	Digital Soil Mapping in R Environment: Practical Workbook
Author:	RCMRD
Last update:	29-05-2024
Printed:	31-05-2024

Preface

Soils provide multiple ecosystem services, which support life on Earth. Thus, the need for their protection, conservation, and sustainable management cannot be overstated. Sustainable soil management calls for spatially-explicit information, depicting the spatial variations in soil types and properties to support data-driven and targeted (site-specific) interventions, decision-making and policy formulation. Such soil information can be easily generated through a combination of field observations, remote sensing, geographic information systems (GIS), geostatistical and machine learning (ML) tools, techniques and data.

This practical workbook is for those want to leverage the power of computers, free and open source software for GIS (FOSS4G), freely accessible GIS and remote sensing data to model and map the spatial variations of different soil properties. This is commonly known as predictive soil mapping (PSM), or digital soil mapping (DSM). The goal of the training is to build capacity for supporting spatially-targeted and sustainable management of agricultural soils in Africa.

Structurally, the practical workbook is divided into two (2) parts. Part I is about definition of the term Digital Soil Mapping (DSM), description of how to obtain and install the requisite software for DSM, and introduction of the spatial data to be used to complete the practical steps. Part II, on the other hand, demonstrates in a logical sequence the fundamental steps of executing a DSM task using reproducible codes, simple data, and geostatistical and machine learning methods. Therefore, the workbook can also serve as a DSM workflow, which can be used at any GIS laboratory for soil mapping.

The DSM approach followed in this training is based on R, the cheapest and most professional statistical programming language and computing environment with powerful data processing, visualization and geospatial capabilities. Staunch users of the software often say that all things are possible in R thanks to the unrestricted creativity and flexibility it allows.

Upon completion of the practical exercises in this Lab, users will be equipped with profound appreciation of R's outstanding geographic capabilities, new spatial skills and the confidence required to solve a range of soil health constraints with geographic data, as well as with the ability to communicate results of soil health investigations with maps and reproducible codes. The spatial skills to be gained will range from importing, processing, exploring and manipulating geographic data, to fitting models and predicting the distribution of geographic phenomena, as well as making thematic maps. So, fold up your sleeves and let the exercises begin!

PART I

This Section is about definition of the term Digital Soil Mapping (DSM), description of how to obtain and install the requisite software for DSM, and introduction of the datasets to be used to complete the practical steps in this workbook.

1.0 Defining DSM

DSM is an approach that *creates spatially-referenced soil maps and databases at a given resolution by quantitatively expressing the relationships between point observations of specific soil properties (or classes) and environmental data, representing soil-forming factors*. DSM has become a major framework in the generation of new soil information grids.

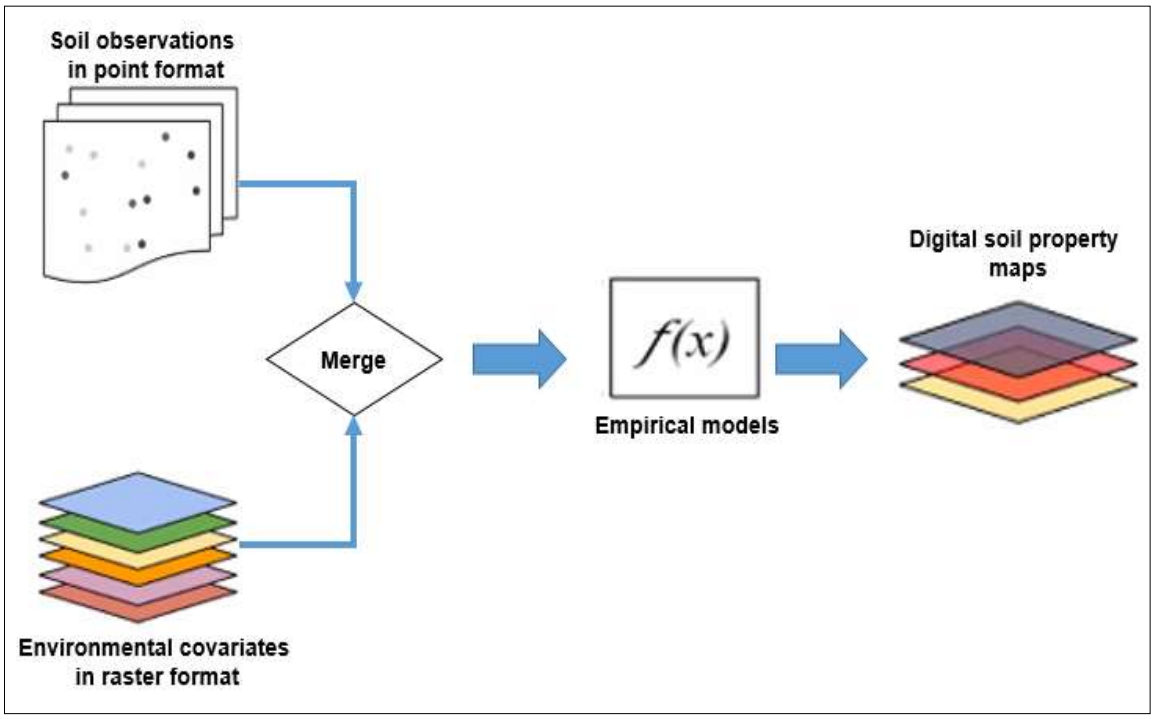


Figure 1: The basis of DSM

Based on the *scorpan* conceptual model of DSM (Eq. (1)) that was formalized by Jenny (1941) and later updated by McBratney et al. (2003), the spatial variability of a target soil property is explained and mapped by its quantitative relationship with the environmental covariates. To achieve this, a wide spectrum of spatially-explicit and readily available auxiliary remote sensing- and GIS-based environmental data, representing the different elements of the *scorpan* model are retrieved from credible open-access geodata sources; for example, the Google Earth Engine (GEE) data archive.

$$S = f\,(s, c, o, r, p, a, n) \qquad \text{Eq. (1)}$$

Where: *S* is the target soil attribute (or class) at the sampled location, which is a function of *s* (other soil properties), *o* (organisms, including land cover), *r* (relief, or terrain attributes), *p* (parent material), *a* (age or time), and *n* (geographic space, or position).

Some of the commonly used environmental covariates include land cover, soils, geology, satellite data (e.g., Landsat and Sentinel spectral reflectance bands), satellite-derived vegetation indices (e.g., Normalized Difference Vegetation Index (NDVI), Enhanced Vegetation Index (EVI), Soil-Adjusted Vegetation Index (SAVI), Modified Soil-Adjusted Vegetation Index and (MSAVI)), climate (e.g., rainfall and temperature), Digital Elevation Model (DEM), and DEM-derivatives (e.g., slope gradient, slope aspect, curvature, topographic wetness index, LS factor and topographic position index).

The foregoing environmental covariates can either be processed in GEE environment or using any other appropriate geo-computing environments, such as ArcGIS, QGIS and R. The main processing operations include resampling, sub-setting and transforming the environmental covariates to a common spatial resolution, extent and referencing framework. After processing, the environmental covariates are stacked together and then intersected with the soil data points to extract the values of the covariates to the points. Next, the functional relationships between the environmental covariates and soil observations (i.e., *f* in Eq. (1)) are modelled, and the spatial patterns predicted and mapped, using appropriate statistical, geostatistical, or machine learning algorithms (e.g., multiple linear regression, kriging and random forests). Lastly, the uncertainties associated with the resultant soil maps are assessed and quantified.

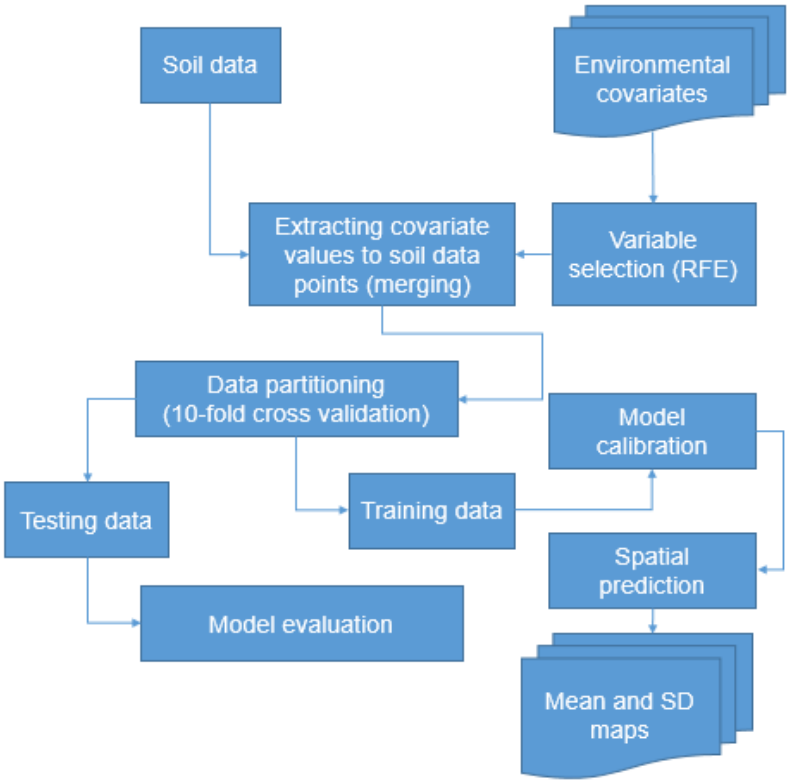


Figure 2: Schematic diagram of the DSM framework

1.1 Obtaining and installing software

This DSM practical workbook comes with some software requirements. Specifically, you are required to install R, R Studio and the associated packages on your computer in order to complete the exercises.

1.1.1 R

R is an open-source programming language and environment for statistical computing and graphics with a unique combination of package ecosystem, powerful data processing, visualization options, geospatial capabilities, and Integrated Development Environments (IDEs). It is well-suited to the interactive use required in many geographic data analysis operations and enables reproducible workflows, or sharing of scripts, which allows others to build-on one's work.

R installation files are freely available for Windows, Mac, and Linux operating systems from the Comprehensive R Archive Network (CRAN) at: http://cran.r-project.org. Run the executable file to install the base R product on your computer.

Prior experience with geographic data and R software is assumed here; so, if you are a beginner, we highly recommend that you first learn the basics of R before attempting to do these exercises. Fortunately for R beginners, there is a supportive community that has developed a wealth of resources that can help.

1.1.2 R Studio

R Studio is an integrated development environment (IDE) for R that runs on Linux, Windows and Mac OS X. You will mostly use this IDE because it has been very well designed (advanced interface), is intuitively organized, and quite stable. The installation files and instructions can be obtained at: http://www.rstudio.com/.

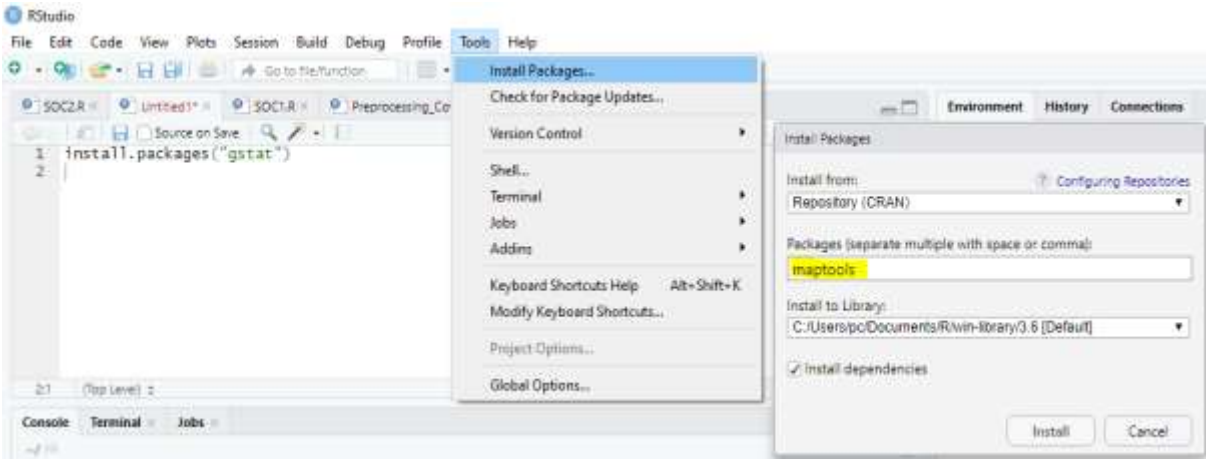
1.1.3 R Packages

The base R environment, which comes with about 25 in-built (standard) packages, has been extended by thousands of contributed packages (~ 15,000 as of October 2019), written by many different authors. Some of the packages implement specialized statistical methods, and others give access to data, or are designed to complement textbooks. Most are available for download through CRAN, its mirrors, and family of internet sites. To perform the practical exercises, you will need additional packages (e.g., sp, raster, terra, and caret), of which each can be installed with the *install.packages ()* function as follows:

- *install.packages ("package name")*
- *install.packages(c("raster", "terra", "caret", etc.))*

Alternatively, you can go straight to the *Tools menu* > *Install Packages* on the graphical user interface (GUI). Then specify your package of interest on the dialogue box that appears, and

hit the *Install* button. If the package is already installed, running *install.packages* overwrites the old installation.



After installation, you can load the packages you want to use into R workspace with the *library ()* function as follows:

- *library (package name)*

1.2 Introducing the datasets

1.2.1 Soil data

In this Lab, you will work with geo-referenced legacy soil profile data of Sudan compiled by ISRIC World Soil Information as an activity of the globally integrated, Africa Soil Information Service (AfSIS) project. The dataset can be freely accessed via ISRIC website (https://files.isric.org/public/afsp/AF-AfSP1.2.zip).

1.2.2 Environmental data

In terms of environmental covariates, you will use 10 gridded geospatial raster datasets, which were freely downloaded from the Google Earth Engine platform. These are:

Data	Spatial resolution	Year
1. SRTM DEM	30 m	
2. Landsat 8 OLI	30 m	2017

PART II

After defining DSM, describing how to obtain and install the necessary software, and introducing the datasets, this Section demonstrates in a logical sequence the core and practical steps of executing a DSM task using reproducible codes and simple data.

2.0 Getting started

For starters, launch R Studio, set the working directory, and load the necessary libraries.

2.1 Setting the working directory and freeing up the memory

```
setwd("C:/Users/pc/Downloads/SUDAN")
gc() # free up memory and report memory usage
rm(list = ls(all.names = TRUE)) # clear all the objects in the global environment
```

2.2 Loading the required libraries

If you have not installed the requisite packages, do so using the instructions given in sub-section 1.1.3. After installation, load each of the packages as follows:

```
library (raster)    # for the manipulation of raster data
library (caret)     # for classification and regression training
library(doParallel) # for parallel processing
library(terra)      # for the manipulation of raster data
install.packages("devtools") # needed for installing package ithir
devtools::install_bitbucket("brendo1001/ithir/pkg") # install package ithir
library(ithir)      # for model evaluation
```

3.0 Pre-processing and exploring soil data

The soil data that you will be using have already been cleaned; that is, the data have been arranged in the required format, missing values have been checked, and outliers, unusual values and duplicates have been removed.

3.1 Loading and viewing soil data

```
soil.data <- read.csv("soilDataSDN.csv", header=T, sep=",") # or use read.csv(choose.files())
View(soil.data) # viewing the data in tabular form
str(soil.data)  # viewing the structure of the data
```

3.2 Exploring the distribution and normality of soil data

```
summary(soil.data$C) # descriptive statistics
boxplot (soil.data$C) # plotting a box plot
hist(soil.data$C, freq=F, xlim=c(0,30), ylim=c(0,0.1), col = "light blue",
     main ="Histogram of SOC", xlab ="SOC") # plotting a histogram
x <- seq(min(soil.data$C, na.rm = TRUE), max(soil.data$C, na.rm = TRUE),length=40)
curve(dnorm(x, mean(soil.data$C, na.rm = TRUE), sd(soil.data$C, na.rm = TRUE)),
      col=2, lwd=2, add=TRUE) # plotting a normal curve
legend("topright",
       legend=c("Density curve", "Normal curve"),
       lty=c("dotted", "solid"),
       bty = "n",
       col=c("dark blue", "red"))
box() # adding a bounding box
qqnorm(soil.data$C) # generating a normal QQ plot to check normality
qqline(soil.data$C, col=2) # generating a line for the QQ plot
shapiro.test(soil.data$C) # Shapiro-wilk statistical test for normality
```

3.3 Defining the CRS and plotting soil data

```
soil.data <- vect(soil.data, geom=c("LON", "LAT"), crs = "+init=epsg:4326") # defining the CRS
soil.data <- project(soil.data, "+init=epsg:32636") # transforming the CRS to UTM Zone 36N
plot(soil.data, pch=19, cex = 0.9, col="blue") # plotting the soil data points
study.area <- vect("Study_area36N.shp") # loading the shapefile of the region of interest (ROI)
study.area
plot(study.area, add=TRUE) # overlaying the ROI on the soil layer
```

3.4 Splitting the soil data (Data partitioning)

Randomly split the soil data into two: (i) a *training* set, and (ii) a *testing* set. You will use the former dataset to calibrate a model and the latter to validate it.

```
set.seed(1234) # for reproducibility of samples and results
train.index <- sample(nrow(soil.data), 0.8 * nrow(soil.data)) # returns indices of the selected samples
soil.train.data <- soil.data[train.index,]
soil.test.data <- soil.data[-train.index,]
```

4.0 Preparing and selecting environmental covariates

Environmental covariates are mostly procured from various freely available sources; hence, they are available in different spatial resolutions, reference systems and extents. Therefore, covariates must be brought to a common spatial resolution, reference system and extent. In this Lab, all covariates have already been pre-processed for you (i.e., resampled to 1000 m resolution, projected to WGS 84 UTM Zone 36 N and clipped to the area of interest).

4.1 Loading the raster files

```
encovs <- list.files(path = ".", pattern = "\\.tif$", full.names = T) # Loads all the raster files at
once using list.files() function
encovs # what files are there?
```

4.2 Stacking the raster files

```
encovs_stack <- rast() # creating an empty rasterStack to store the output
for (i in 1:length(encovs)){
  encovs_stack <- c(encovs_stack, rast(encovs[i]))
}
names(encovs_stack) # what files are there?
encovs_names <- names(encovs_stack)
```

4.3 Extracting raster values to the soil data

Now, intersect the soil data points and environmental covariates to form a *matrix* where each row is a soil observation at a given spatial location and the columns are the corresponding values of the covariates. This is why it was important to stack the environmental covariates. Be sure that the CRS of the covariates and soil data match.

4.3.1 Soil data matrix (Training)

```
encovs_stack <- project(encovs_stack, crs(study.area))## Project rasterstack, if projection is missing
soil.train.data.ov <- extract(x = encovs_stack, y = soil.train.data)
soil.train.data.ov <- cbind(soil.train.data, soil.train.data.ov)
soil.train.data.ov <- as.data.frame(soil.train.data.ov) # coercing into a data frame
which(!complete.cases(soil.train.data.ov)) # checking for missing values
soil.train.data.ov <- soil.train.data.ov[complete.cases(soil.train.data.ov),]
str(soil.train.data.ov)
write.table(soil.train.data.ov, "soil.train.data.ov.csv", sep=",", col.names=TRUE) # Saving to .csv file
```

4.3.2 Soil data matrix (Testing)

```
soil.test.data.ov <- extract(x = encovs_stack, y = soil.test.data, xy=T)
soil.test.data.ov <- cbind(soil.test.data, soil.test.data.ov)
soil.test.data.ov <- as.data.frame(soil.test.data.ov)
which(!complete.cases(soil.test.data.ov))
soil.test.data.ov <- soil.test.data.ov[complete.cases(soil.test.data.ov),]
str(soil.test.data.ov)
write.table(soil.test.data.ov, "soil.test.data.ov.csv", sep=",", col.names=TRUE)
```

4.4 Exploring and selecting the optimal environmental covariates

Availability of multiple, freely available spatial datasets explains the high dimensionality of environmental covariates in DSM models. However, it is always advisable to select just a few important (i.e., optimal) covariates for model development to increase the interpretability and computational efficiency of the models. Covariate selection can be accomplished using several methods. Here, recursive feature elimination (RFE) method is applied. Like backward regression, RFE begins with the maximum number of covariates and iteratively eliminates the least significant variable until a predefined number of covariates is attained.

4.4.1 Setting the RFE parameters

```
set.seed(1235)
rfe.ctrl <- rfeControl(functions = rfFuncs,
                       method = "repeatedcv",
                       number = 10,          ## 10 -fold CV
                       repeats = 3,         ## repeated 3 times
                       verbose = TRUE,
                       saveDetails = TRUE,
                       returnResamp = "all")
```

4.4.2 Calibrating the RFE model

```
## Parallel processing (using multiple cores)
cl <- makeCluster(detectCores()-1)
registerDoParallel(cl)

## Set the regression function/ formula with all covariates
fm <- as.formula(paste("C"," ~", paste0(encovs_names, collapse = "+")))

covsel <- rfe(fm,
              data=soil.train.data.ov,
              sizes= c(1:30),
              rfeControl = rfe.ctrl,
              verbose = TRUE,
              keep.inbag = T)

stopCluster(cl)

print(covsel) # summarize the results
plot(covsel, type = c("g", "o")) # plot the results
predictors(covsel) # list the selected features
opt_encovs <- predictors(covsel) # extract the selected features
```

5.0 Spatial modelling, prediction and mapping

5.1 Model development

This step involves quantification of the functional relationships between the environmental covariates and soil attributes using the training data and appropriate statistical, geostatistical, or machine learning algorithms. Random forest algorithm, which creates an ensemble of decision trees using a random selection of covariates is used in this case.

```
## Parallel processing (using multiple cores)
cl <- makeCluster(detectCores()-1)
registerDoParallel(cl)

## Set training parameters
trn.ctrl <- trainControl(method="repeatedcv",
                         number=10, ## 10 -fold CV
                         repeats = 3, ## repeated 3 times
                         search="grid",
                         savePredictions = TRUE)

## Tune the hyperparameters
rfGrid <-  expand.grid(mtry=c(1:11))
```

```
## Update the regression function/ formula with the selected covariates
rfm <- as.formula(paste("C"," ~", paste0(opt_encovs, collapse = "+")))

## Fit the RF model
rf.C.fit <- train(rfm,
                  data=soil.train.data.ov,
                  method="rf",
                  trControl=trn.ctrl,
                  verbose = TRUE,
                  tuneGrid=rfGrid,
                  keep.inbag = T,
                  importance = TRUE)
stopCluster(cl)

## Model summary
print(rf.C.fit) # or rf.C.fit$results
plot(rf.C.fit)
```

5.2 Relative importance of the input covariates

```
importance <- varImp(rf.C.fit, scale=FALSE, main="Soil Organic Carbon")
print(importance)
plot(importance)
```

5.3 Model evaluation

This step entails computation of accuracy statistics to measure the model performance. The RF model calibrated using the training data is applied to predict the observed values of the target soil attributes in the testing data, and the resultant residuals used to estimate various performance metrics. In this case, you will compute *root mean squared error* (RMSE), *mean error* (ME, bias), *coefficient of determination* (R^2) and *concordance* as the measures of model performance.

```
## Internal evaluation (using training data)
  ## Extract observed and predicted values
  o <- rf.C.fit$pred$obs
  p <- rf.C.fit$pred$pred
  df.op <- data.frame(o,p)

  ## Plot scatter plot
  (g1 <- ggplot(df.op, aes(x = o, y = p)) +
      geom_point() +
      geom_abline(color = "red", linetype = 2)+
      geom_smooth(method = lm, se = FALSE)+
      ylim(c(min(o), max(o))) + theme(aspect.ratio=1)+
      labs(title = "Soil organic carbon content") +
      xlab("Observed") + ylab("Predicted"))

## External evaluation (using testing data)
  ## Predict onto the testing soil data points
  rf.C.test.predictions <- predict(rf.C.fit, soil.test.data.ov)

  ## Compare the observations with the predictions
  goof(soil.test.data.ov$C, rf.C.test.predictions)
```

5.4 Model application (spatial mapping)

Here, you will predict onto the grid of covariates to generate a prediction surface and a digital soil organic carbon content map.

```
rf_C_map <- predict(encovs_stack, rf.C.fit)
plot (rf_C_map, main = "Spatial Distribution of SOC content (%)")
grid() # add a grid
```

5.5 Saving the output prediction surface (digital map)

```
writeRaster(rf_C_map, "rf.C.prediction.tif", overwrite=TRUE)
```

6.0 Summary and concluding remarks

This Lab has demonstrated how to apply geospatial and machine learning techniques to model, predict and map the spatial variations of SOC using a free (geo)computing software and spatial data. You can load the saved prediction surface in a GIS environment, such as QGIS and ArcGIS, to design better maps for presentations and publications. The wealth of spatial analytical and mapping skills and knowledge gained in the process can also be applied to investigate and map other related and spatially-continuous environmental phenomena.