Sampling theory for estimating soil organic carbon change - a tutorial

Gerard Heuvelink
Diana Collazos Cortes
Giulio Genova

2025-09-17

Funded by the European Union

Table of contents

Preface

Citation Heuvelink, G., Collazos Cortes, D. F., & Genova, G. (2025). Sampling theory for estimating soil organic carbon change - a tutorial. ISRIC - World Soil Information. https://doi.org/10.17027/AWW3-F404

Acknowledgements This tutorial was developed within the **ORCaSa** project, which received funding from the European Union's Horizon Europe research programme under grant agreement n°101059863.

1 Introduction

This tutorial shows how sampling theory from statistics can be used to estimate the total Soil Organic Carbon (SOC) stock for an area, or to estimate the change in total SOC stock for an area between two points in time. Sampling theory requires a probability sample from the area of interest and uses so-called design-based statistical inference. This has the important advantages that the estimates are unbiased, model-free (i.e., makes no model assumptions), and that the estimation accuracy can be quantified. The latter is for instance important to derive confidence intervals and evaluate whether an estimated increase or decrease of the SOC stock over time is 'real' and statistically significant.

The tutorial explains the underlying theory but the emphasis is on practical application using a concrete example and the R language for statistical computing. The tutorial provides all R scripts and datasets for the example area.

We developed this tutorial such that it should not be difficult to adapt it to other case studies, thus supporting the verification step of Monitoring, Reporting and Verification (MRV) projects. More information on MRV for SOC and other aspects of sampling for SOC stock changes is available in Ceschia et al. (2025).

We assume that users of the tutorial have basic knowledge of statistics, geo-information science and basic experience with R. Note that we also assume that R and required packages have been installed (version 4.4.3 or higher).

2 Soil organic carbon stock of an example area

In this chapter we introduce the example area that is used throughout the tutorial. We present a map of the soil organic carbon (SOC) stock of the area, and explore how the total SOC stock of this area can be estimated with simple random sampling.

All calculations are shown in code chunks, which should be run in the order as presented. We explain the most relevant aspects of the functions used, but we encourage the user to consult the documentation and experiment with the script by making small modifications or extensions to support a better understanding.

The setup lines below are necessary for the code chunks in this chapter to run. Therefore, we define them at the beginning of the script.

Setup

In this chapter we will only use functions from the `terra` package.

```
# load library
library(terra)
```

R automatically prints large numbers in scientific notation. We prefer, however, to have numbers printed in standard notation. To avoid scientific notation, we adjust the `scipen` ("scientific penalty") option. By setting it to `999`, we essentially force standard notation.

```
# avoid scientific notation
options(scipen=999)
```

The tutorial makes use of a pseudo-random number generator. To encourage reproducible research we set the seed of this generator, so that anyone running this script gets the same results.

```
# set a seed (for reproducible research)
set.seed(12345)
```

Numeric results of calculations are often rounded, but since R does not print trailing zeros by default, we wrap the rounding in a number formatting function.

```r
# x is the number, d the intended number of decimals
# p is the precision to round the number to the nearest factor
# for example, when x=87.8904, d=1 and p=1 returns "87.9"
# for example, when x=33745.5, d=0 and p=100 returns "33700"
round2 <- function(x, d, p = 1){
  p <- p*(10^-d)
  n <- round(x/p)*p
  format(n, nsmall = d)
  }
```

We selected a rectangular study area in the north-east of the Netherlands, shown in Figure 2.1. The rectangle measures 4.8 *km* by 4.3 *km*. The main land use is agriculture (cropland and grassland) on clay soils.

Figure 2.1: Study area (red rectangle).

> The bounding box was defined in *EPSG:28992 (Amersfoort / RD New)* to match the coordinate system of the input data. When reprojected to *WGS84 (EPSG:4326)* for visualization in Leaflet, the rectangle may appear rotated or skewed. This is a harmless visual effect caused by the transformation from a planar to a geographic coordinate system. All calculations in this tutorial are performed in *EPSG:28992*, so the distortion does not affect the analysis.

In order to take a random sample from the example area and use this sample of SOC stock

point observations to estimate the 'true' SOC stock of the area, as well as evaluate how close this estimate is to the 'true' SOC stock of the area, we need to know the SOC stock everywhere in the area. For the aims of this tutorial we therefore created a SOC stock synthetic reality raster. It is important to be aware that in reality we of course would not know the SOC stock at each and every location in a study area. While we need a map of the 'true' SOC stock for the purposes of this tutorial, obviously sampling theory works well and is in fact intended for cases where such information is not available.

The primary data source for simulation of a synthetic SOC stock raster is Helfenstein et al. (2024), which provides high-resolution soil property grid maps, including soil organic matter (SOM) and bulk density (BD) maps. We first combined the SOM and BD maps to derive SOC stock maps for the $0-5$ cm , $5-15$ cm and $15-30$ cm depth layers, summed these to obtain SOC stock maps for the $0-30$ cm topsoil, and finally added zero-mean spatially-correlated residuals using geostatistical simulation. Residuals were added to acknowledge that the $0-30$ cm SOC stock map derived from the maps provided in Helfenstein et al. (2024) is only a smoothed representation of the true SOC stock.

We load the SOC stock map from the ISRIC WebDAV.

Load data

```
# load raster from ISRIC WebDAV
url_folder <- "https://files.isric.org/public/tutorials/soc_stock_change/input/"
raster_fn <- "soc_stock_2020.tif"
soc <- rast(paste0(url_folder, raster_fn))
```

Plot raster

```
# plot soc stock raster
plot(soc, main = "topsoil SOC stock",
     plg = list(title = "ton/ha"))
```

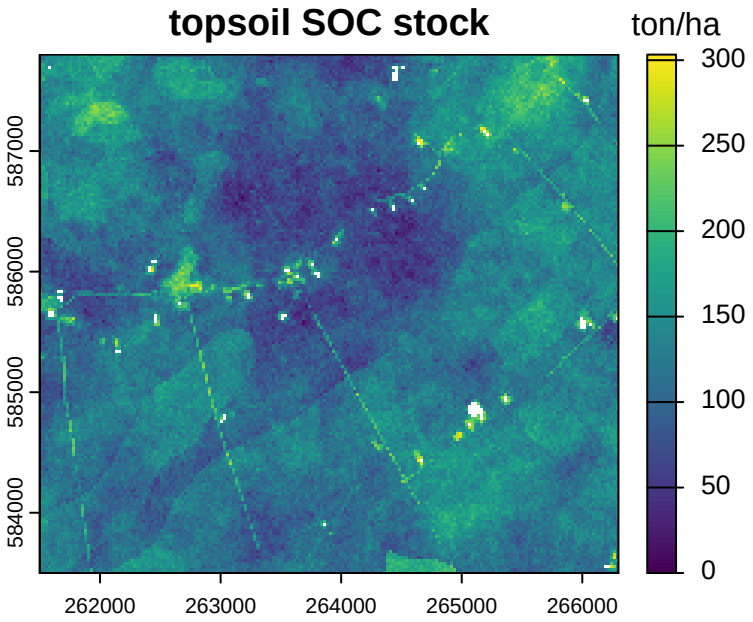


Figure 2.2: Topsoil SOC stock map of the example area, at 25m x 25m resolution. This map defines the 'reality' from which we will sample.

To calculate the SOC stock for the example area we average all grid cell values and multiply by the size of the area. Note that there are `NA` values in the rectangle shown in Figure 2.2 (i.e., houses and farm buildings). These cells are not part of the target area and must therefore be removed from the calculation.

Total area

```
# terra::expanse() returns the area covered by all non-NA raster cells
area_ha <- expanse(soc, unit = "ha")[1, "area"]

# print
paste0("Area: ", round2(area_ha, 2), " ha")
```

```
[1] "Area: 2057.93 ha"
```

'True' SOC stock

```
# calculate the mean soc stock across the non-NA pixels (ton/ha)
mean_stock <- global(soc, fun = "mean", na.rm=TRUE)[1, "mean"]

# multiply the mean SOC stock (ton/ha) by area (ha) to get the total SOC stock (ton)
tot_stock <- mean_stock * area_ha

# print
paste0("'True' SOC stock: ", round2(tot_stock, 0, 10), " ton")
```

```
[1] "'True' SOC stock: 248390 ton"
```

This is 248.4 Gg (Gigagram). Feel free to check in R that since the size of the area without `NA` cells is 2057.9 ha, the mean SOC stock is 120.7 ton/ha.

In reality, we do not know the SOC stock at all locations in the area and so we cannot calculate it. At best we can make an estimate of it, by measuring the SOC stock at a limited number of locations and multiplying the mean of these SOC stock measurements by the size of the study area. For instance, we might randomly select 50 locations and estimate the total SOC stock, as follows:

Case 1

```
# sample 50 non-NA locations as spatial points. In spatSample(),
# when as.points = TRUE, it converts the ampled cells into
# spatial points using xyFromCell(), which by default returns the
# cell centroid. When values = TRUE, raster cell values are returned
random_sample <- spatSample(soc, size = 50, method = "random",
                            as.points = TRUE, values = TRUE,
                            na.rm = TRUE)
# plot
plot(soc, main = "Random sample of 50 locations, case 1", cex.main = 1,
     plg = list(title = "ton/ha", title.cex = 0.8, cex = 0.8))
plot(random_sample, pch = 21, cex = 1,
     col = "black", bg = "orange", add = TRUE)
```

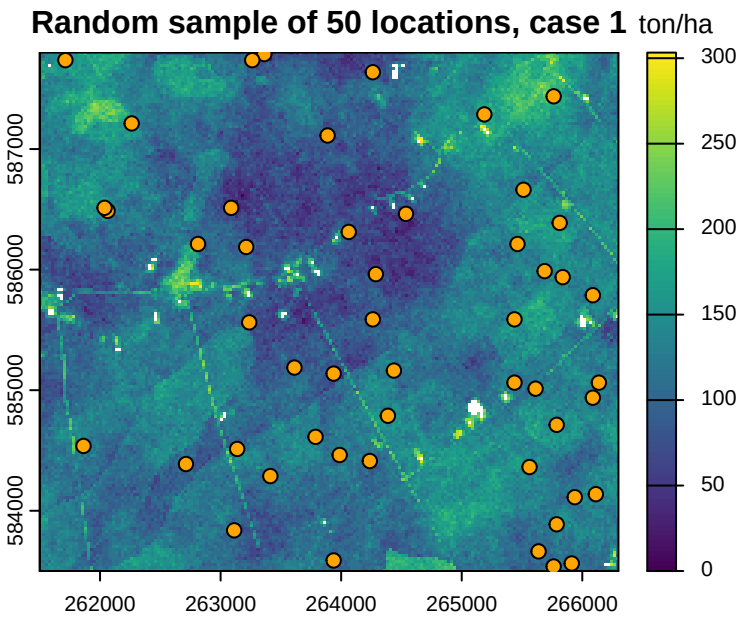


Figure 2.3: Random sample of 50 locations in the example area (case 1).

```
# select the first column, which contains the extracted soc stock values
random_sample <- as.data.frame(random_sample)[[1]]

# compute the mean soc stock for the sampling locations (ton/ha)
sample_mean <- mean(random_sample)

# total soc stock in the study area computed from the sample mean
tot_stock_estimate_1 <- sample_mean * area_ha

# print
paste0("Total SOC stock estimate 1: ", round2(tot_stock_estimate_1, 0, 10), " ton")
```

```
[1] "Total SOC stock estimate 1: 245860 ton"
```

This is only an estimate of the total SOC stock based on 50 observations. There is an **estimation error**, which we can quantify because we know the true SOC stock of the area:

Case 1

```r
# calculate the estimation error
error_1 <- tot_stock_estimate_1 - tot_stock

# print
paste0("Estimation error 1: ", round2(error_1, 0, 10), " ton")
```

```
[1] "Estimation error 1: -2530 ton"
```

This shows that we underestimated the total SOC stock in the area by 2528.0 *ton*. Clearly, if we had sampled 50 other locations we would get a different estimate and estimation error. Let us now repeat the estimation procedure above four times to get a feel for the magnitude and sign of the estimation error:

Case 2

```r
# sample 50 non-NA locations as spatial points
random_sample <- spatSample(soc, size = 50, method = "random",
                            as.points = TRUE, values = TRUE,
                            na.rm = TRUE)

# plot
plot(soc, main = "Random sample of 50 locations, case 2", cex.main = 1,
     plg = list(title = "ton/ha", title.cex = 0.8, cex = 0.8))
plot(random_sample, pch = 21, cex = 1,
     col = "black", bg = "orange", add = TRUE)
```

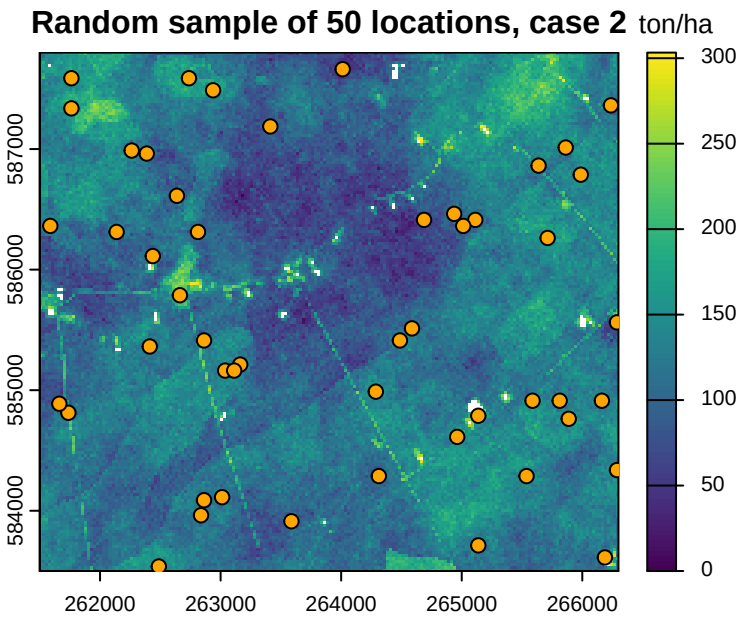


Figure 2.4: Random sample of 50 locations in the example area (case 2).

```
# select the first column, which contains the extracted soc stock values
random_sample <- as.data.frame(random_sample)[[1]]

# compute the mean soc stock for the sampling locations (ton/ha)
sample_mean <- mean(random_sample)

# total soc stock in the study area computed from the sample mean
tot_stock_estimate_2 <- sample_mean * area_ha

# check
paste0("Total SOC stock 2: ", round2(tot_stock_estimate_2, 0, 10), " ton")

# calculate the estimation error
error_2 <- tot_stock_estimate_2 - tot_stock

# print
paste0("Estimation error 2: ", round2(error_2, 0, 10), " ton")
```

```
[1] "Total SOC stock 2: 276870 ton"
[1] "Estimation error 2: 28490 ton"
```

Case 3

```
# sample 50 non-NA locations
random_sample <- spatSample(soc, size = 50, method = "random",
                            as.points = FALSE, values = TRUE,
                            na.rm = TRUE)
# when we set as.points = FALSE, spatSample() returns a data frame

# select the first column, which contains the extracted soc stock values
random_sample <- random_sample[[1]]

# compute the mean soc stock for the sampling locations (ton/ha)
sample_mean <- mean(random_sample)

# total soc stock in the study area computed from the sample mean
tot_stock_estimate_3 <- sample_mean * area_ha

# check
paste0("Total SOC stock 3: ", round2(tot_stock_estimate_3, 0, 10), " ton")

# calculate the estimation error
error_3 <- tot_stock_estimate_3 - tot_stock

# print
paste0("Estimation error 3: ", round2(error_3, 0, 10), " ton")
```

```
[1] "Total SOC stock 3: 241330 ton"
[1] "Estimation error 3: -7060 ton"
```

Case 4

```r
# sample 50 non-NA locations
random_sample <- spatSample(soc, size = 50, method = "random",
                            as.points = FALSE, values = TRUE,
                            na.rm = TRUE)
# when we set as.points = FALSE, spatSample() returns a data frame

# select the first column, which contains the extracted soc stock values
random_sample <- random_sample[[1]]

# compute the mean soc stock for the sampling locations (ton/ha)
sample_mean <- mean(random_sample)

# total soc stock in the study area computed from the sample mean
tot_stock_estimate_4 <- sample_mean * area_ha

# print
paste0("Total SOC stock 4: ", round2(tot_stock_estimate_4, 0, 10), " ton")

# calculate the estimation error
error_4 <- tot_stock_estimate_4 - tot_stock

# print
paste0("Estimation error 4: ", round2(error_4, 0, 10), " ton")
```

```
[1] "Total SOC stock 4: 240150 ton"
[1] "Estimation error 4: -8230 ton"
```

Case 5

```r
# sample 50 non-NA locations
random_sample <- spatSample(soc, size = 50, method = "random",
                            as.points = FALSE, values = TRUE,
                            na.rm = TRUE)
# when we set as.points = FALSE, spatSample() returns a data frame

# select the first column, which contains the extracted soc stock values
random_sample <- random_sample[[1]]

# compute the mean soc stock for the sampling locations (ton/ha)
sample_mean <- mean(random_sample)

# total soc stock in the study area computed from the sample mean
tot_stock_estimate_5 <- sample_mean * area_ha

# print
paste0("Total SOC stock 5: ", round2(tot_stock_estimate_5, 0, 10), " ton")

# calculate the estimation error
error_5 <- round((tot_stock_estimate_5 - tot_stock), 1)

# print
paste0("Estimation error 5: ", round2(error_5, 0, 10), " ton")
```

```
[1] "Total SOC stock 5: 235000 ton"
[1] "Estimation error 5: -13380 ton"
```

It appears that the estimation error can be positive as well as negative and that its magnitude varies between cases, and is far from negligible. The absolute value of the five estimation errors had a maximum of 28490 *ton*, which is 11.5 % of the total SOC stock. Ideally we would like to know how large the estimation error is expected to be for a given sample size. Can anything be said about this, even if we have only a single sample of 50 observations? And how much smaller would the estimation error on average be if we had taken a larger sample (for instance 100 instead of 50 random locations)? If we could answer these questions we would be able to determine how large a sample we must take to achieve a pre-defined required estimation accuracy.

It is important to note that above we could only compute the estimation error for each case because we 'knew' the true SOC stock of the area. In reality we do not know it, and cannot calculate the actual estimation error. In fact, all we have in reality is a single estimate of the true SOC stock, based on the one random sample that we took. Can we tell how accurate the estimate is, even if we do not know the true SOC stock and have a single random sample? Can we derive a confidence interval for the true SOC stock based on this sample? Can we quantify how estimation accuracy depends on sample size? These are important questions that can all

be answered affirmatively, with the help of sampling theory from statistics.

The next chapters provide a concise explanation of the basics of sampling theory, and applies this theory to our example case.

3 Sampling theory

In statistics, a **population** is defined as the complete set of elements or objects in which a researcher has an interest. In this tutorial, the population is the set of all locations, that is all 25 $m \times 25$ m grid cells that are agricultural land, in the example area. The number of objects in the population is called the **population size**, and often denoted with the mathematical symbol N. For our study area, introduced in Chapter 2, we can calculate the number of non-`NA` grid cells with `terra`.

First, we run the same setup code as in Chapter 2, load the data, and calculate the number of objects in the population.

Setup

```
# load library
library(terra)
```

```
# avoid scientific notation
options(scipen=999)
```

```
# rounding function
round2 <- function(x, d, p = 1){
  p <- p*(10^-d)
  n <- round(x/p)*p
  format(n, nsmall = d)
  }
```

Load data

```
# load raster from ISRIC WebDAV
url_folder <- "https://files.isric.org/public/tutorials/soc_stock_change/input/"
raster_fn <- "soc_stock_2020.tif"
soc <- rast(paste0(url_folder, raster_fn))
```

Number of non-NA cells

```
# number of non-NA cells
ncells <- global(soc, fun="notNA")

# global() returns a data frame, but we only need a number
ncells <- ncells$notNA

# print
paste0("Number of non-NA grid cells: ", ncells)
```

```
[1] "Number of non-NA grid cells: 32932"
```

Therefore, in this tutorial, $N = 32932$.

The elements in a population usually have multiple measurable attributes, which are called **population characteristics**. Example characteristics of our population are elevation, annual precipitation and groundwater level, but in this tutorial we are only concerned with one specific characteristic; that is the SOC stock of the $0-30$ cm topsoil, denoted by SOC. Each object in the population has a SOC value, and we define the **population mean** (μ_{SOC}) as the average of the characteristic over all objects in the population:

$$\mu_{SOC} = \frac{1}{N}\sum_{i=1}^{N} SOC_i$$

where SOC_i is the SOC stock of the i-th object in the population.

The SOC stock is usually not constant in space so not all N objects have the same SOC value. We can quantify the variability by the **population variance**, defined as:

$$\sigma^2_{SOC} = \frac{1}{N}\sum_{i=1}^{N}(SOC_i - \mu_{SOC})^2$$

If SOC is measured in $ton\ ha^{-1}$ then the population mean μ_{SOC} will also be in $ton\ ha^{-1}$, but the population variance σ^2_{SOC} will be in $(ton\ ha^{-1})^2$. To facilitate interpretation, we often take its square root to obtain σ_{SOC}, which is known as the **population standard deviation**, measured in $ton\ ha^{-1}$.

For our population, we can calculate the population mean and standard deviation in R as follows:

Population mean

```
# calculate the mean of the soc stock across the example area (ton/ha)
mean_stock <- global(soc, fun = "mean", na.rm=TRUE)

# global() returns a data frame, but we need a number
mean_stock <- mean_stock$mean

# print
paste0("Population mean: ", round2(mean_stock, 1), " ton/ha")
```

```
[1] "Population mean: 120.7 ton/ha"
```

Population standard deviation

```
# calculate the population standard deviation of
# soc stock across the example area (ton/ha)
std_stock <- global(soc, fun = "std", na.rm=TRUE)

# global() returns a data frame, but we need a number
std_stock <- std_stock$std

# print
paste0("Population standard deviation: ", round2(std_stock, 1), " ton/ha")
```

```
[1] "Population standard deviation: 36.1 ton/ha"
```

Coefficient of variation

```
# coefficient of variation
cv_stock <- std_stock/mean_stock

# print
paste0("Coefficient of variation: ", round2(cv_stock * 100, 1), " %")
```

```
[1] "Coefficient of variation: 29.9 %"
```

Note that we had already calculated the population mean in Chapter 2. Note also that SOC has substantial spatial variation, since σ_{SOC} is 0.299 times μ_{SOC}, that is the **coefficient of variation** is 29.9%.

3.1 Unbiased estimation of the population mean

In practice, we do not know the population mean μ_{SOC} because we cannot afford to measure the SOC stock for each and every grid cell in the study area. Instead, we measure it for a subset of the population, that is a **sample**. The number of observations in the sample is denoted by n, the **sample size**. Typically, the sample size is much smaller than the population size, that is $n \ll N$. In Chapter 2 we used $n = 50$. We also chose the sample of 50 objects randomly from the population, where each object of the population had equal probability of being chosen and where a next object was chosen independently from objects already included in the sample. In other words, we took a **simple random sample** from the population.

We can now estimate the population mean with the **sample mean**:

$$\hat{\mu}_{SOC} = \overline{SOC} = \frac{1}{n}\sum_{i=1}^{n} SOC_i$$

where in this case the SOC_i, $(i = 1 \ldots n)$ are the SOC observations at the n sampling locations (i.e., grid cells).

Sampling theory now tells us that the sample mean $\overline{SOC}$ is an **unbiased** estimate of the population mean μ_{SOC}. Note that this does not mean that the estimation error will be zero (as we had already noticed in Chapter 2). It only means that on average, if we would repeat the random sampling many times, the estimation error will be zero. We will verify this in Chapter 4. In mathematical terms, sampling theory tells us that the **expected value** of the estimation error is zero.

3.2 Standard error

While using an unbiased estimation method is important and attractive, the estimate we get from just one sample will still have an error; the sample mean will likely differ from the population mean. Fortunately, sampling theory also tells us how large the estimation error is likely to be. It turns out that in case of simple random sampling, the variance of the estimation error satisfies:

$$var(\hat{\mu}_{SOC} - \mu_{SOC}) = \frac{\sigma^2_{SOC}}{n}$$

Recall that the variance is a measure of the variability. Here, it means that if we would take a simple random sample of n objects from the population many times, each time calculating the sample mean and use it as an estimate of the population mean, then the collection of the associated estimation errors would have a variance equal to the result of the equation above. Note also that the mean of that same collection of estimation errors would be zero, since the sample mean is an unbiased estimate of the population mean.

From the equation above follows that the standard deviation of the estimation error, denoted by the **standard error**, is equal to $\frac{\sigma_{SOC}}{\sqrt{n}}$. Note that, as expected, the standard error becomes smaller as the sample sizes increases. However, to double the estimation accuracy (i.e., to halve the standard error) one needs to quadruple the sample size. The standard error is also proportional to the population standard deviation. This means that it is more difficult to accurately estimate the population mean of populations that are highly variable, while a small sample size will be sufficient to obtain an accurate estimate if the objects in the population have similar values for the characteristic of interest. In the extreme case of no variability (i.e., all objects in the population have the same value), one needs to measure only one object ($n = 1$) to obtain a perfect, error-free estimate of the population mean.

3.3 Confidence interval

Finally, sampling theory also tells us that the frequency distribution of a large collection of estimation errors will approximate a normal distribution (if at least n is sufficiently large, say $n > 30$). This is thanks to the Central Limit Theorem from statistics. See Figure 3.1 for a graphical illustration. Note that the distribution is centered on zero, since the sample mean is an unbiased estimate of the population mean.

In practice, we have only one sample from the population and hence only one estimate of the population mean. Figure 3.1 shows the **probability distribution** of the estimation error associated with that estimate. Using properties of the normal distribution, we learn that in about 95 out of a 100 cases, the estimation error is bigger than $-1.96 \times \frac{\sigma_{SOC}}{\sqrt{n}}$ and smaller than $1.96 \times \frac{\sigma_{SOC}}{\sqrt{n}}$. With this we can derive a **95% confidence interval** of the population mean as:

$$\langle \hat{\mu}_{SOC} - 1.96 \times \frac{\sigma_{SOC}}{\sqrt{n}}, \hat{\mu}_{SOC} + 1.96 \times \frac{\sigma_{SOC}}{\sqrt{n}} \rangle$$

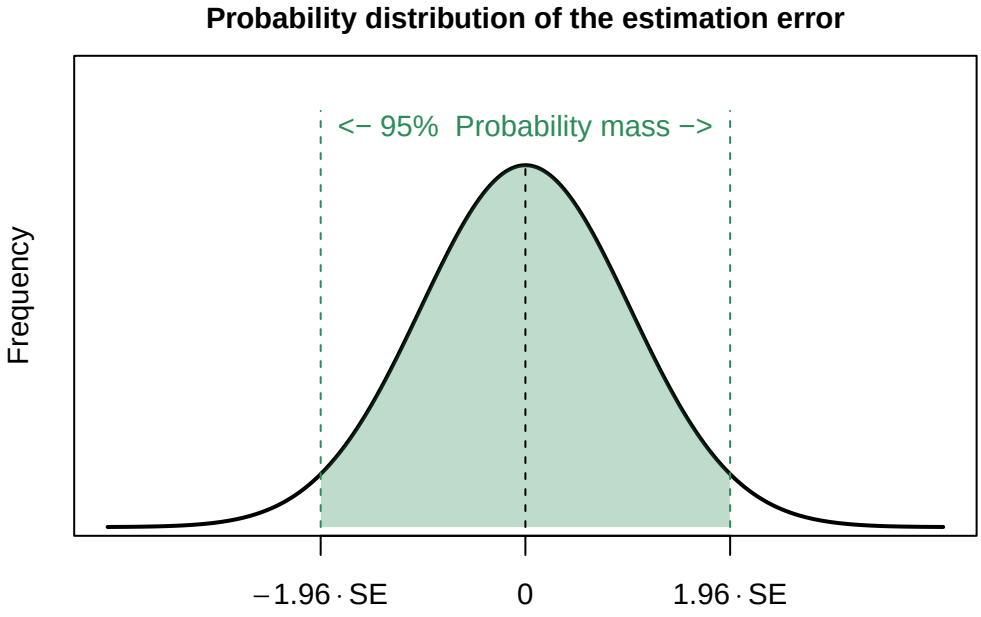


Figure 3.1: The estimation error associated with a simple random sample is approximately normally distributed. The probability of an estimation error bigger than $-1.96\ SE$ and smaller than $+1.96\ SE$ is 95%. SE: Standard Error.

In Chapter 4 we will repeatedly calculate the 95% confidence interval, each time for a different simple random sample from the SOC population, and verify that indeed in about 95 out of 100 cases we find that the population mean is inside the lower and upper limits of the confidence interval.

The attentive reader might have noticed that we used the population variance σ^2_{SOC} to calculate the standard error and confidence interval limits. In practice we do not know the population and would likely replace it by the **sample variance** s^2_{SOC}:

$$s^2_{SOC} = \frac{1}{n}\sum_{i=1}^{n-1}(SOC_i - \hat{\mu}_{SOC})^2$$

This has some implications that will lead to wider confidence intervals if n is small, but this is beyond the scope of this tutorial.

It is important to note that all results presented in this chapter are valid because we took a **simple random sample** from the population. If we had taken the sample in another way, for instance through **preferential sampling** or **convenience sampling**, then we could still use the sample to estimate the population mean, but we could not claim that the estimate is

unbiased, we could not quantify the estimation error with a standard error, and we could not derive a confidence interval.

4 Application to example area

In Chapter 3 we presented the basics of sampling theory. In this chapter we will verify the theoretical results of Chapter 3 with the SOC stock data from the example area.

To prepare for the calculations in this chapter, we first run the following setup lines.

Setup

```
# load library
library(terra)
```

```
# avoid scientific notation
options(scipen=999)
```

```
# set a seed
set.seed(987)
```

```
# rounding function
round2 <- function(x, d, p = 1){
  p <- p*(10^-d)
  n <- round(x/p)*p
  format(n, nsmall = d)
  }
```

Load data

```
# load raster from ISRIC WebDAV
url_folder <- "https://files.isric.org/public/tutorials/soc_stock_change/input/"
raster_fn <- "soc_stock_2020.tif"
soc <- rast(paste0(url_folder, raster_fn))
```

4.1 Demonstrating unbiasedness

We first verify that the sample mean is an unbiased estimate of the population mean by running a for loop that repeatedly takes a random sample of 50 observations from the study area (i.e., the population), each time computing the estimated SOC stock, and checking how close the average of those estimates is to the true total SOC stock:

Population mean and 'True' SOC stock

```
# compute the area (ha) (excluding NAs)
area_ha <- expanse(soc, unit = "ha")[1, "area"]

# calculate the mean SOC stock across the non-NA pixels (ton/ha)
mean_stock <- global(soc, fun = "mean", na.rm=TRUE)[1, "mean"]

# multiply mean SOC stock (ton/ha) by area (ha) to get the total SOC stock
tot_stock <- mean_stock * area_ha
```

Estimation error of random samples

```
# indicate number of loops and sample size
loops <- 1000
sample_size <- 50
```

```r
# create a vector to store the estimation errors
est_errors <- c()

# warning: it may take a while to run
for (i in seq_len(loops)){
  # sample n non-NA locations as spatial points
  random_sample <- spatSample(soc, size = sample_size,
                              method = "random", as.points = FALSE,
                              values = TRUE, na.rm = TRUE)
  # when we set as.points = FALSE, spatSample() returns a data frame

  # select the first column, which contains the extracted soc stock values
  random_sample <- random_sample[[1]]

  # compute the average SOC stock for this sample (ton/ha)
  sample_mean <- mean(random_sample)

  # calculate associated estimate of the total stock
  tot_stock_est <- sample_mean * area_ha

  # calculate the estimation error
  est_error <- tot_stock_est - tot_stock

  # add to the estimation error vector
  est_errors <- c(est_errors, est_error)
}
```

```r
# print summary statistics of the 1000 estimation errors
summary(est_errors)
```

```
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-33858.073  -6985.637     -8.042    -84.803   7094.286  34178.303
```

Estimation errors distribution

```r
# plot a frequency distribution of the 1000 estimation errors
hist(est_errors,
     main = paste0("Distribution of ", loops,
                   " estimation errors with sample size n = ",
                   sample_size),
     xlab = "Estimation error (ton)",
     freq = TRUE,
     breaks = 24,
     cex.main = 0.8, cex.lab = 0.8, cex.axis = 0.8)
```

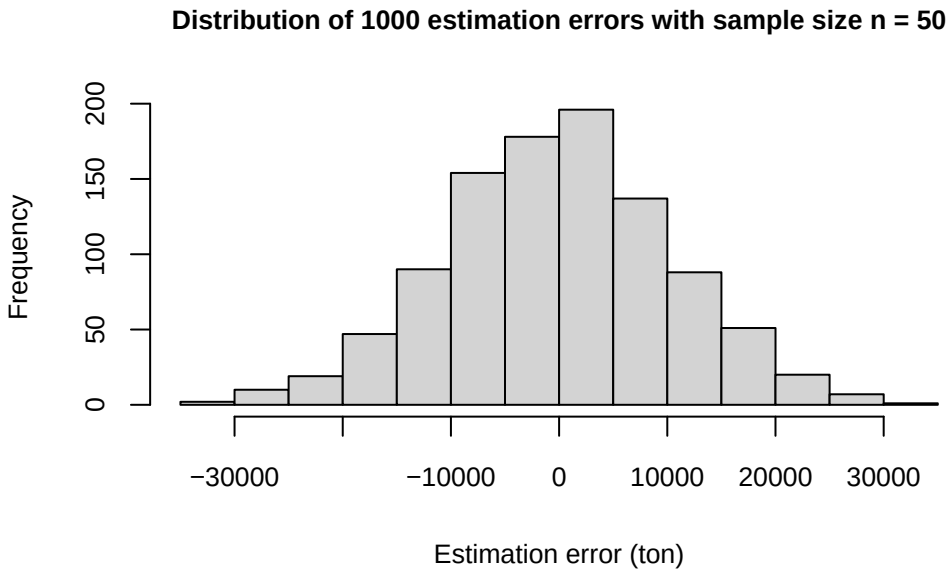


Figure 4.1: Frequency distribution of 1000 SOC stock estimation errors, each of which associated with an estimate of the SOC stock using a simple random sample of size 50.

Note that the mean of the 1000 estimation errors is very small (it is -84.8 ton), compared to the SOC stock (248387.7 ton), and that it will get closer and closer to zero if we increase the number of repeats. Feel free to check this yourself by changing $n = 1000$ to $n = 100000$ or $n = 1000000$, if this is computationally feasible.

4.2 Verifying the normal distribution and confidence intervals

Chapter 3 not only stated that the expected value of the estimation error is zero but also that it is normally distributed and has a spread equal to the standard error. This is also confirmed by the 1000 estimation errors, because the frequency distribution shown in Figure 4.1 matches the normal distribution very well and has a spread that equals the standard error. This is verified in the R code below.

Estimation errors distribution vs normal distribution

```
# calculate the standard error (see Chapter 3)
SE <- global(soc, fun = "std", na.rm=TRUE)$std *area_ha / sqrt(sample_size)

# create a sequence of x values (range covers the histogram)
x <- seq(-4*SE, 4*SE, length.out = loops)
# width of histogram bins
bin_width <- diff(hist(est_errors, plot = FALSE)$breaks)[1]
# calculate the normal density values at x
# scaled by histogram bin width and count
y <- dnorm(x, mean = 0, sd = SE)*loops*bin_width

# plot histogram with normal distribution on top
hist(est_errors,
     main = paste0("Distribution of ", loops,
                   " estimation errors with sample size n = ",
                   sample_size, "\n",
                   "with theoretical normal distribution plotted on top"),
     xlab = "Estimation error (ton)",
     freq = TRUE,
     breaks = 24,
     cex.main = 0.8, cex.lab = 0.8, cex.axis = 0.8)

lines(x, y, lwd = 2, col = "black")
```

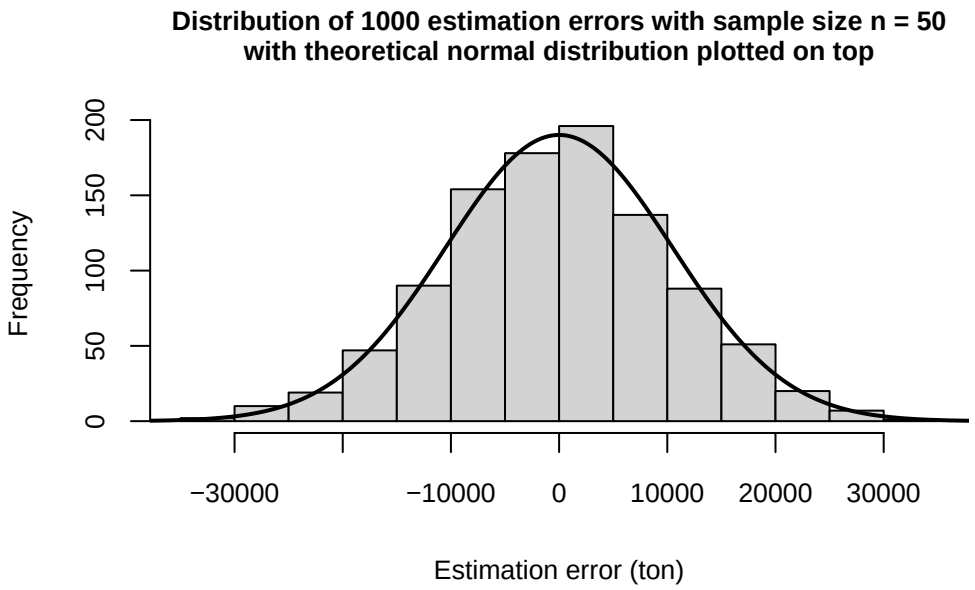


Figure 4.2: Distribution of 1000 SOC stock estimation errors based on random samples of size 50. The black line represents a zero-mean normal distribution with a spread equal to the standard error.

Feel free to run the code above with a larger number of repeats. The more repeats we do, the closer the frequency distribution of the estimation errors will approximate the theoretical probability distribution.

Section 3.3 showed how a 95% confidence interval for the population mean is derived. It was explained that, in 95 out of a 100 cases, the true population mean is between the lower and upper limits of the 95% confidence interval. We can verify this by counting how often this occurred with the 1000 repeats.

Verify theoretical confidence interval

```r
# indicate number of loops and sample size
loops <- 1000
sample_size <- 50

# create a vector to store TRUE or FALSE if the
# true population mean is within the confidence interval
within_confintv <- c()

# warning: it may take a while to run
for (i in seq(1, loops, 1)){
  # sample n non-NA locations as spatial points
  random_sample <- spatSample(soc, size = sample_size,
                              method = "random", as.points = FALSE,
                              values = TRUE, na.rm = TRUE)
  # when we set as.points = FALSE, spatSample() returns a data frame

  # select the first column, which contains the extracted soc stock values
  random_sample <- random_sample[[1]]

  # compute the average SOC stock for this sample (ton/ha)
  sample_mean <- mean(random_sample)

  # compute the standard error
  SE <- sd(random_sample)/sqrt(sample_size)

  # compute the lower and upper limit of the confidence interval
  lower_limit <- sample_mean - 1.96*SE
  upper_limit <- sample_mean + 1.96*SE

  # is the true population mean within the confidence interval?
  check_intv <- lower_limit <= mean_stock & mean_stock <= upper_limit

  # add iteration check to vector
  within_confintv <- c(within_confintv, check_intv)
}
```

```r
# how many times did the true mean fall within the confidence interval?
within_confintv <- sum(within_confintv)

# print
paste0("The true mean fell within the confidence interval ",
       within_confintv, " out of 1000 times")
```

```
[1] "The true mean fell within the confidence interval 941 out of 1000 times"
```

We find that in 941 out of 1000 cases the population mean is inside the 95% confidence interval, which is close to the expected value of 950.

4.3 Analysing the effect of sample size

Let us now repeat the analysis above for different sample sizes, that is for $n = 200$ and $n = 800$, and compare the average width of the 95% confidence interval with that of the $n = 50$ case.

Calculate confidence interval widths for different sample sizes

```
# indicate number of loops and sample size
loops <- 1000
sample_size <- c(50, 200, 800)
```

```r
# warning: it may take a while to run
for (n in sample_size) {
  # initialize a vector to store the width of the confidence interval
  width_confintv <- c()

  for (i in seq_len(loops)) {
    # sample n non-NA locations as spatial points
    random_sample <- spatSample(soc, size = n, method = "random",
                                as.points = FALSE, values = TRUE,
                                na.rm = TRUE)
    # when we set as.points = FALSE, spatSample() returns a data frame

    # select the first column, which contains the extracted soc values
    random_sample <- random_sample[[1]]

    # compute the average SOC stock for this sample (ton/ha)
    sample_mean <- mean(random_sample)

    # compute the standard error
    SE <- sd(random_sample) / sqrt(n)

    # compute the lower and upper limit of the confidence interval
    lower_limit <- sample_mean - 1.96 * SE
    upper_limit <- sample_mean + 1.96 * SE

    # add iteration width to vector
    width_confintv <- c(width_confintv, upper_limit - lower_limit)
  }

  # compute the mean of the confidence interval widths
  width_confintv_mean <- mean(width_confintv)

  # convert to SOC stock
  width_confintv_mean <-  width_confintv_mean * area_ha

  # print
  print(paste0("Confidence interval width mean when n=", n, ": ",
               round2(width_confintv_mean, 0, 10)," ton"))
}
```

```
[1] "Confidence interval width mean when n=50: 40880 ton"
[1] "Confidence interval width mean when n=200: 20550 ton"
[1] "Confidence interval width mean when n=800: 10270 ton"
```

This shows, as expected, that the confidence interval width is smaller if the sample size is

larger. It also confirms that to double the estimation accuracy (to obtain a twice as narrow confidence interval) we need to quadruple the sample size. Figure 4.3 shows the relation between the confidence interval width and sample size for sample sizes varying from $n = 25$ to $n = 1600$. You may verify that the confidence interval width is inversely proportional to the square root of the sample size. Note also that for small sample sizes the confidence interval width is quite variable and varies from case to case (i.e., it is highly influenced by the random sample that one happens to take), as evidenced by the wider box plots.

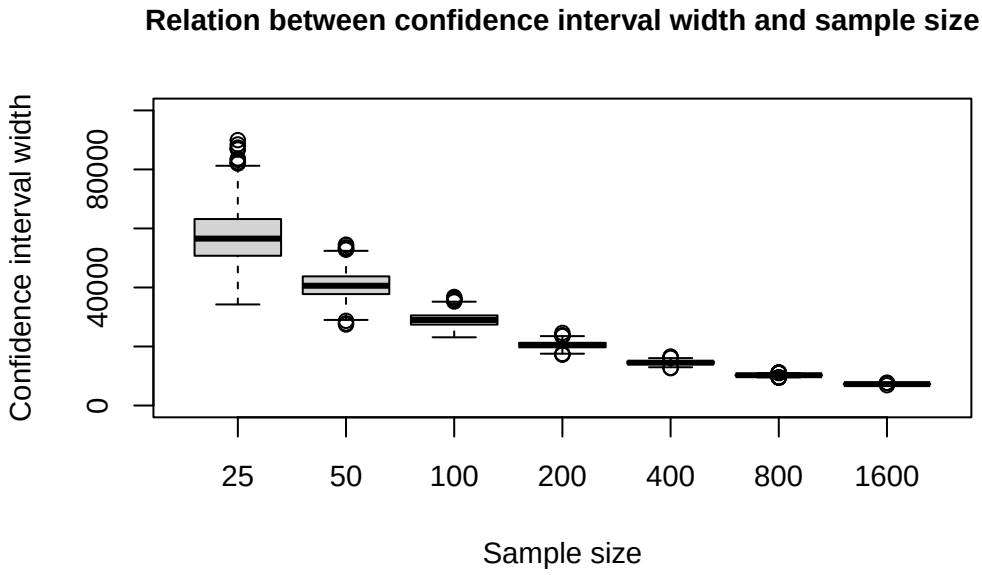


Figure 4.3: The confidence interval width becomes smaller as the sample size increases. Results for the 1000 repeats for each sample size are summarised in boxplots.

This chapter confirmed the theoretical results of Chapter 3. It showed that simple random sampling yields an unbiased estimate of the total SOC stock in an area of interest. It also showed that the estimation accuracy is adequately assessed with a confidence interval, and that the width of the confidence interval decreases as the sample size increases.

It is important to be aware that in 1 out of 20 cases the true SOC stock is outside the 95% confidence interval. In statistics we can never be 100% certain, unless we measure the entire population.

We could of course reduce the chance of the true population mean being outside the confidence interval by replacing the 95% confidence interval with a 98% or 99% confidence interval (feel free to try this out yourself), but the price paid is an increase of the confidence interval width.

We can decrease the confidence interval width by increasing the sample size, but this will

increase fieldwork and laboratory costs. In Chapter 5 we will explore how **stratified random sampling** can improve estimation accuracy without requiring a larger sample size.

5 Stratified random sampling

We start this chapter with running a few setup lines and loading the data.

Setup

```
# load libraries
library(terra)
library(sf)
library(dplyr)
```

```
# set a seed
set.seed(987)
```

```
# rounding function
round2 <- function(x, d, p = 1){
  p <- p*(10^-d)
  n <- round(x/p)*p
  format(n, nsmall = d)
  }
```

Load data

```
# load raster from ISRIC WebDAV
url_folder <- "https://files.isric.org/public/tutorials/soc_stock_change/input/"
raster_fn <- "soc_stock_2020.tif"
soc <- rast(paste0(url_folder, raster_fn))
```

Prepare data

```r
# modify soc raster metadata to make it easier to call
soc_values <- "soc_values"
names(soc) <- soc_values
```

Total area

```r
# terra::expanse() returns the area covered by all non-NA raster cells
area_ha <- expanse(soc, unit = "ha")[1, "area"]
```

In Chapter 4 we learnt that with simple random sampling we obtain an unbiased estimate of the population mean, in our case the total topsoil SOC stock in the example area. The estimate is simply the unweighted average of the sample of observations that we took, that is the **sample mean**. We also learnt that we can quantify the accuracy of this estimate with a confidence interval.

Confidence interval of simple random sampling

```r
# indicate the sample size
sample_size <- 200

# sample n non-NA locations
random_sample <- spatSample(soc, size = sample_size, method = "random",
                            as.points = FALSE, values = TRUE,
                            na.rm = TRUE)

# select the first column, which contains the extracted soc values
random_sample <- random_sample[[1]]

# compute the average SOC stock for this sample (ton)
sample_mean <- mean(random_sample)*area_ha

# compute the standard error
SE <- sd(random_sample)/sqrt(sample_size)*area_ha

# compute the lower and upper limit of the confidence interval
lower_limit <- sample_mean - 1.96*SE
upper_limit <- sample_mean + 1.96*SE
width_confintv <- upper_limit - lower_limit

# print
paste0("Estimated mean (ton): ", round2(sample_mean, 0, 10))
paste0("95% confidence interval: [", round2(lower_limit, 0, 10), ", ",
       round2(upper_limit, 0, 10), "]")
paste0("Confidence interval width: ", round2(width_confintv, 0, 10))
```

```
[1] "Estimated mean (ton): 241090"
[1] "95% confidence interval: [230530, 251640]"
[1] "Confidence interval width: 21110"
```

The above shows that, using a simple random sample of size 200, we are 95% confident that the true SOC stock of the example area is between 230530 and 251640 ton.

Simple random sampling is just one of many probability sampling designs, and usually not the most efficient. There are other designs that can produce a more accurate estimate (i.e., a narrower confidence interval) with the same sample size. In this chapter we will consider one such design, that is **stratified random sampling**.

Stratified random sampling divides the population of interest into sub-populations, also called *strata*. In our case, we subdivide the example area into subareas that do not overlap and that together cover the entire area. Stratified random sampling then continues by taking a simple random sample from each stratum, and deriving the estimate of the population mean and the associated standard error and confidence interval from the observations in all strata.

Stratified random sampling will produce a more accurate estimate of the population mean than simple random sampling for the same sample size if the strata are homogeneous, that is, if the variance of the population characteristic is smaller within strata than between strata.

One way of defining the strata in our example area would therefore be to delineate subareas that have the same landuse, soil type or land management history, because the SOC stock depends on these factors. However, in this chapter we will use strata that are geographically compact. This should also work because the SOC stock will tend to have similar values for locations that are not that far apart. We will derive the strata first, next briefly explain the statistical inference associated with stratified random sampling, and finally apply it to our example area to verify that the confidence interval indeed narrows.

5.1 Compact geographical strata

Compact geographical strata can be calculated for any area of interest using the k-means algorithm. We adapted code from the "Spatial Sampling with R" book by Brus (2022). We use 40 strata and ensure that all have approximately the same size (number of pixels) and are as geographically compact as possible.

Compute geographical strata using the k-means algorithm

```
# extract cell coordinates of non-NA cells
cells <- which(!is.na(values(soc))) # indices of valid cells
coords <- xyFromCell(soc, cells)    # centroid coordinates matrix (x,y)

# define number of strata
k <- 40

# run kmeans on the coordinates
# iter.max is 10 by default
km <- kmeans(coords, centers = k)

# create an empty raster to store strata IDs
strata_raster <- rast(soc)
values(strata_raster) <- NA

# assign strata IDs back to raster cells
values(strata_raster)[cells] <- km$cluster

# modify metadata
varnames(strata_raster) <- "kmeans_stratified_raster"
names(strata_raster) <- "strata"
```

```
# print
strata_raster
```

```
class       : SpatRaster
size        : 172, 192, 1  (nrow, ncol, nlyr)
resolution  : 25, 25  (x, y)
extent      : 261500, 266300, 583500, 587800  (xmin, xmax, ymin, ymax)
coord. ref. : Amersfoort / RD New (EPSG:28992)
source(s)   : memory
varname     : kmeans_stratified_raster
name        : strata
min value   :      1
max value   :     40
```

We could already run `terra::plot(strata_raster)` to view the strata, but let us first add some elements to refine the map. We want to apply a discrete colour legend in which each stratum has a different colour than its neighbours.

Plot k-means geographic strata with discrete colour legend

```
# convert raster to polygons, dissolve to merge cells of the same stratum
polys <- as.polygons(strata_raster, dissolve = TRUE) # SpatVector polygon
```

```
Warning in x@pntr$as_polygons(round[1], aggregate[1], values[1], na.rm[1], :
GDAL Message 1: DeprecationWarning: 'Memory' driver is deprecated since GDAL
3.11. Use 'MEM' onwards. Further messages of this type will be suppressed.
```

```r
polys_sf <- st_as_sf(polys) # sf polygon

# adjacency is a list of integer vectors indicating
# which polygons touch which other polygons
adjacency <- st_touches(polys_sf)

# initialize vector of colours (by index)
color_ids <- rep(0, length(adjacency))

# function to assign the smallest colour index not used by neighbours
for (i in seq_along(color_ids)) {
  neighbor_colors <- color_ids[adjacency[[i]]]
  used <- unique(neighbor_colors[neighbor_colors > 0])
  color_ids[i] <- min(setdiff(1:5, used))
}

# choose a discrete colour palette
palette <- c("#8DD3C7", "#FFFFB3", "#BEBADA", "#FB8072", "#80B1D3")

# assign these colours to the strata IDs
strata_colors <- palette[color_ids]
```

We can also add a label that specifies the stratum ID and the number of pixels it contains.

```r
# find the number of pixels per strata
fc <- freq(strata_raster, bylayer = FALSE)
```

```r
# get centroids from the spatVect polygon
centroids <- centroids(polys)
```

```r
# plot to check
plot(strata_raster, col = strata_colors, legend = FALSE,
     main = paste0(k, " geographic strata (k-means on terra)"))

# add labels
text(centroids, labels = paste(fc$value, fc$count, sep = "\n"),
     cex = 0.7, col = "black")
```

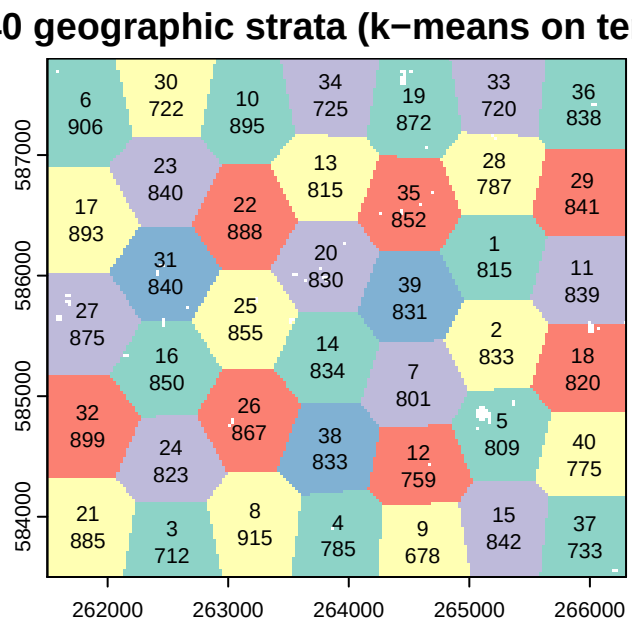


Figure 5.1: Compact geographic stratification of the study area generated by the k-means algorithm with `terra`. The labels inside the strata indicate the stratum ID and its number of pixels.

5.2 Statistical inference of stratified random sampling

In stratified random sampling, we divide the population into K strata, take a simple random sample of size n_k $(k = 1 \dots K)$ from each stratum, and estimate the population mean μ_{SOC} by:

$$\hat{\mu}_{SOC,Str} = \sum_{k=1}^{K} a_k \cdot \overline{SOC}_k$$

where a_k represents the size of stratum k relative to that of the whole area. It is calculated as:

$$a_k = \frac{N_k}{N}$$

where N_k is the number of grid cells in stratum k. $\overline{SOC}_k$ is the average of the n_k SOC stock observations in stratum k:

$$\overline{SOC}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} SOC_{k,i}$$

with $SOC_{k,i}$ the i-th SOC stock observation in stratum k.

Note that in addition to choosing the number of strata K and defining the strata, one must also define the sample size n_k of each stratum. Obviously the sum of these must equal the total sample size, that is $\sum_{k=1}^{K} n_k = n$, but different divisions over the strata can be used. One option is to allocate larger sample sizes to strata that are heterogeneous (if one has prior information about that), or to allocate smaller sample sizes to strata that are poorly accessible. Everything is allowed, as long as one takes a simple random sample from within the stratum and all stratum sample sizes are 2 or bigger. It is also common to use **proportional allocation**, that is to choose the stratum sample size proportional to the size of the stratum, meaning that a stratum that is twice as large an another stratum has a twice as large sample size. In this chapter we will use a constant sample size per stratum, which is close to proportional allocation since all strata in Figure 5.1 have a similar same size.

In stratified random sampling, the variance of the estimation error is given by:

$$var(\hat{\mu}_{SOC,Str} - \mu_{SOC}) = \sum_{k=1}^{K} a_k^2 \cdot \frac{s_k^2}{n_k}$$

where s_k^2 is the sample variance within stratum k and given by:

$$s_k^2 = \frac{1}{n_k - 1} \sum_{i=1}^{n_k} (SOC_{k,i} - \overline{SOC}_k)^2$$

As you can see the calculations are a bit more involved than for simple random sampling, but it is only a matter of working meticulously and making sure to use the inference that goes with the sampling design. We can compute the standard error from the variance of the estimation error in the usual way (by taking the square root) and use that to derive a confidence interval.

5.3 Application to the example area

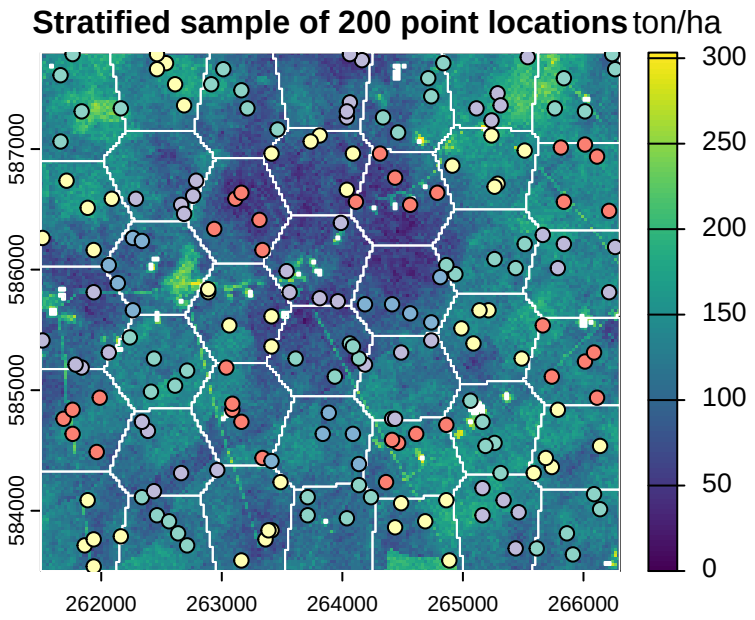
Let us use $K = 40$ and $n = 200$ to estimate the SOC stock of our example area using stratified random sampling. Note that this implies that we use $n_k = 5$.

Generate a stratified random sample

```
# we want the total sample size to be n=200
# since we have 40 strata, we need 5 sampling
# locations in each stratum
n_k <- 5

# convert strata_raster to data frame with cell index
strata_df <- as.data.frame(strata_raster, xy = TRUE,
                           cells = TRUE, na.rm = TRUE)

# sample 5 points per stratum
stratified_df <- strata_df %>%
  group_by(strata) %>%
  slice_sample(n = n_k)

# convert to SpatVector
stratified_points <- vect(stratified_df, geom = c("x", "y"),
                          crs = crs(strata_raster))

# extract soc stock values at stratified point locations
stratified_sample <- extract(soc, stratified_points, bind = TRUE)
```

Plot stratified random sample

```
# previously we had calculated the strata
# we show stratum boundaries and the 200 point locations
# so that we can verify we have exactly 5 points per stratum
strata_colors <- strata_colors[stratified_sample$strata]

# plot
plot(soc, main = "Stratified sample of 200 point locations",
     plg = list(title = "ton/ha"), cex.main = 1)
plot(polys, col = NA, border = "white", add = TRUE)
plot(stratified_sample["strata"], pch = 21, cex = 1,
     col = "black", bg = strata_colors,
     add = TRUE)
```



Check that the total number of grid cells over all strata is correct

```
# number of non-NA cells
N <- global(soc, fun="notNA")
# global() returns a data frame, but we only need a number
N <- N$notNA

# find the number of pixels per stratum
N_k <- freq(strata_raster, bylayer = FALSE)
# make sure it is sorted by stratum
N_k <- N_k[order(N_k$value), ]

# in N_k, the number of pixels per stratum is stored in the "count" column
# the sum of the counts should be equal to the total number of non-NA pixels
sum(N_k$count) == N
```

```
[1] TRUE
```

Stratified sample mean

The `group_by()` function in the `dplyr` library will help us group the data by stratum to calculate the sample mean of each stratum.

```
# a_k represents the size of a stratum relative to that of the whole area
a_k <- N_k[["count"]]/N

# calculate the sample mean per stratum
soc_k <- stratified_sample %>%
  as.data.frame() %>%
  group_by(strata) %>%
  summarise(soc_k = mean(soc_values)) %>%
  arrange(strata) # make sure it is sorted by strata

# convert to vector
soc_k <- soc_k[["soc_k"]]

# compute the stratified mean and
# multiply by size of area to get total SOC stock (ton instead of ton/ha)
stratified_mean <- sum(a_k * soc_k) * area_ha
```

Stratified variance

```
# compute the sample variance within strata s_k^2
s_k2 <- stratified_sample %>%
  as.data.frame() %>%
  group_by(strata) %>%
  summarise(s_k2 = var(soc_values)) %>%
  arrange(strata)

# convert to vector
s_k2 <- s_k2[["s_k2"]]

# compute variance of the estimation error
stratified_variance <- sum((a_k^2) * (s_k2/n_k))
```

Confidence interval of stratified random sampling

```r
# compute standard error
SE <- sqrt(stratified_variance) * area_ha

# compute confidence interval
lower_limit <- stratified_mean - 1.96 * SE
upper_limit <- stratified_mean + 1.96 * SE
width_confintv_str <- upper_limit - lower_limit

# print
paste0("Estimated mean (stratified): ",
       round2(stratified_mean, 0, 10), " ton")
paste0("95% confidence interval: [",
       round2(lower_limit, 0, 10), ", ",
       round2(upper_limit, 0, 10), "]")
paste0("Confidence interval width: ",
       round2(width_confintv_str, 0, 10), " ton")
```

```
[1] "Estimated mean (stratified): 248190 ton"
[1] "95% confidence interval: [239580, 256810]"
[1] "Confidence interval width: 17230 ton"
```

As expected, the confidence interval width of stratified random sampling is smaller than that of simple random sampling. The width decreased from 21110 ton to 17230 ton, thus a decrease of 18.4%. It pays off to use stratified random sampling, even with using compact geographical strata.

6 Estimating SOC stock change

So far we applied statistical sampling theory to estimate the SOC stock for an area, but it can also be used to estimate the SOC stock **change** in an area between two points in time. This is most interesting for MRV projects, since these wish to establish whether the SOC stock of an area has changed between the start and end of a project. Much of the methodology used so far applies to estimating change as well, but there are some extensions that must be addressed. The most important of these is a decision on the **space-time sampling design**. In this chapter we distinguish two cases:

1) **Static sampling**, where we use the same sampling locations at the start and end of the project;
2) **Synchronous sampling**, also referred to as **dynamic sampling**, where we use a different set of sampling locations at the start and end of the project.

We will see that static sampling is more efficient, meaning that, with the same sample size, it yields a more accurate estimate of the SOC stock change than synchronous sampling. The disadvantage, however, is that it is susceptible to manipulation, since projects might pay more effort to increasing the SOC stock at the sampling locations and ignore other parts of the area, since these will not be sampled anyway. It is also important to be aware that in practice one will never be able to return exactly to the same location and measure the SOC stock of the same soil sample at the start and end of a project, if only because soil sampling is destructive.

We will illustrate static and synchronous sampling using the same example area that we used in previous chapters. In Chapter 2 we presented a map of the SOC stock in Figure 2.2, but in fact, this was a map of the SOC stock for the year 2020. Using space-time geostatistics, SOC stock maps for multiple years were generated, including the years 1990 and 2020. In this chapter we sample from the simulated SOC stock maps of these two years to estimate the SOC stock change during the 30-year period from 1990 to 2020.

As usual we start with running some setup lines.

Setup

```
# load library
library(terra)
```

```
# avoid scientific notation
options(scipen=999)
```

```
# set a seed (for reproducible research)
set.seed(987)
```

```
# rounding function
round2 <- function(x, d, p = 1){
  p <- p*(10^-d)
  n <- round(x/p)*p
  format(n, nsmall = d)
  }
```

To ensure colour scale comparability when plotting multiple rasters, the `range` argument can be set in the `terra::plot` function. The function below saves us a few lines of code when we need to find the minimum and maximum values for one or more rasters.

```
# function to compute a raster min max values
get_minmax <- function(c_rasters){
  mimax_rasters <- global(c_rasters, fun = "range", na.rm = TRUE)
  min_range <- floor(min(mimax_rasters["min"]))
  max_range <- ceiling(max(mimax_rasters["max"]))
  rasters_range <- c("min" = min_range,
                     "max" = max_range)
  return(rasters_range)
}
```

Load data

```
# load raster from ISRIC WebDAV
url_folder <- "https://files.isric.org/public/tutorials/soc_stock_change/input/"
soc_1990 <- rast(file.path(url_folder, "soc_stock_1990.tif"))
soc_2020 <- rast(file.path(url_folder, "soc_stock_2020.tif"))
```

Prepare data

```
# rename to make it easier to call
names(soc_1990) <- "soc_1990"
names(soc_2020) <- "soc_2020"
```

Figure 6.1 and Figure 6.2 show the SOC stock maps for 1990 and 2020, and Figure 6.3, shows a map of the SOC stock change from 1990 to 2020. As noted before, in practice we would never have these maps but only the samples that we took in 1990 and 2020. However, for the purposes of this tutorial we need these maps, so that we can sample from them and assess how close the estimated SOC stock change is to the 'true' SOC stock change.

Derive the 'true' SOC stock change map

```
# calculate the soc stock change
soc_change <- soc_2020 - soc_1990

# fix metadata
names(soc_change) <- "soc_stock_change_ton_ha_1990-2020"
varnames(soc_change) <- names(soc_change)
```

Prepare elements for plotting

```
# find the min-max across the 1990 and 2020 rasters
soc_range <- get_minmax(c(soc_1990, soc_2020))
```

```r
# min-max values of the soc stock change raster
diff_range <- get_minmax(soc_change)

# get the absolute maximum to mirror the colour palette around it
max_abs <- max(abs(diff_range))

# define the palette colours
colour_palette <- colorRampPalette(c("red", "white", "blue"))

# define number of colour codes to generate from the colour palette
n_colours <- 100

# get the colour codes
colours <- colour_palette(n_colours)
```

Plot SOC stock in 1990, 2020, and its change

```r
# plot soc stock 1990
plot(soc_1990, main = "topsoil SOC stock 1990",
     plg = list(title = "ton/ha"), range = soc_range)
```

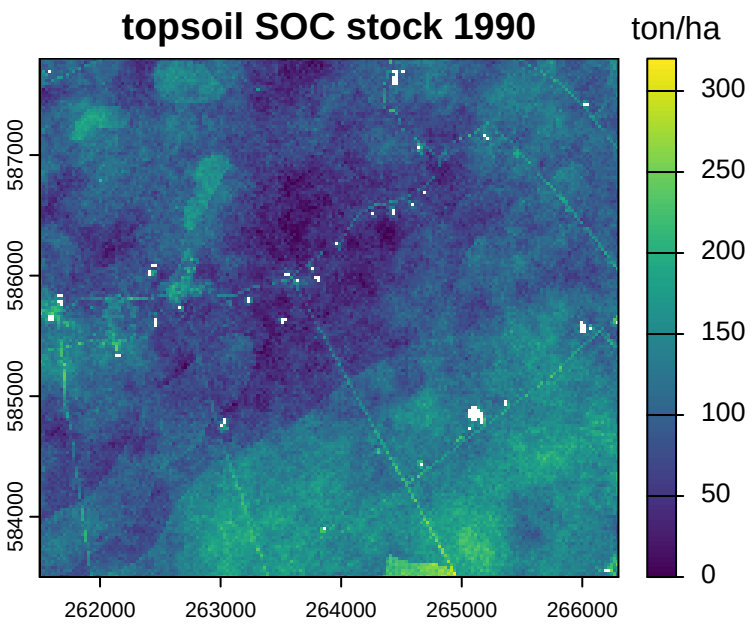


Figure 6.1: Topsoil SOC stock in the year 1990.

```
# plot soc stock 2020
plot(soc_2020, main = "topsoil SOC stock 2020",
     plg = list(title = "ton/ha"), range = soc_range)
```

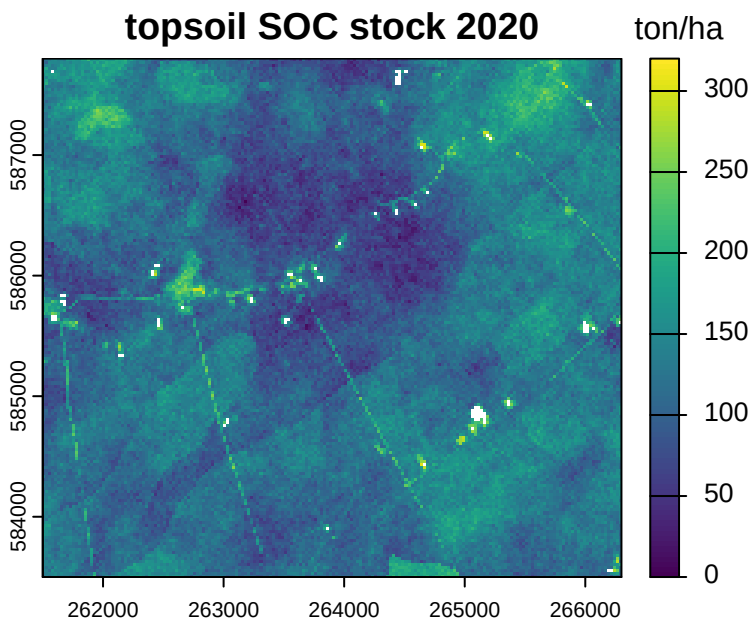


Figure 6.2: Topsoil SOC stock in the year 2020.

```
# plot soc change
plot(soc_change,
     main = paste0("topsoil SOC stock change"),
     plg = list(title = "ton/ha"), col = colours,
     range = c(-max_abs, max_abs))
```

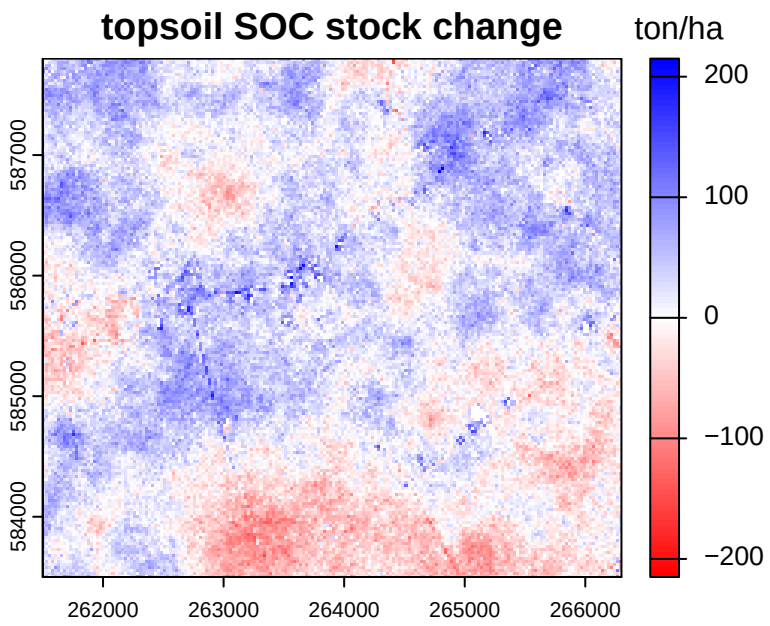


Figure 6.3: SOC stock change between 1990 and 2020.

Save SOC stock change raster (optional)

```
# provide an output path
outputDir <- "/home/tutorial/example/output"
```

```
# use writeRaster() to save a raster output
writeRaster(soc_change,
            filename = file.path(outputDir, "soc_stock_change.tif"),
            gdal=c(
              "of=COG",           # Output format COG (Cloud Optimized GeoTIFF)
              "COMPRESS=DEFLATE"  # Compression method
              ),
            overwrite=TRUE)
```

Total area

```
# terra::expanse() returns the area covered by all not-NA raster cells
# since the area is the same for both rasters, we can use either
area_ha <- expanse(soc_2020, unit = "ha")[1, "area"]
```

'True' SOC stock change

```
# calculate the mean soc stock change across the non-NA pixels (ton/ha)
mean_stock_change <- global(soc_change, fun = "mean", na.rm=TRUE)[1, "mean"]

# multiply SOC stock change (ton/ha) by area (ha) to get the
# total SOC stock change (ton) between 1990 and 2020
tot_stock_change <- mean_stock_change*area_ha

# print
cat("'True' SOC stock change: ", round2(tot_stock_change, 0, 10), " ton\n")

# divide by the number of years to get the soc change per yer
stock_change_per_year <- tot_stock_change/(30 * area_ha)

# print
cat("'True' SOC stock change per unit area and year:\n",
       round2(stock_change_per_year, 3), " ton per ha per year\n")
```

```
'True' SOC stock change:  33750  ton
'True' SOC stock change per unit area and year:
 0.547  ton per ha per year
```

The 'true' SOC stock change for the example area during the 30-year period is 33750 ton, which corresponds to 0.547 $ton\ ha^{-1}\ yr^{-1}$.

Overall, the SOC stock in the example area has increased between 1990 and 2020. But Figure 6.3 shows that the SOC stock change is highly spatially variable with values between -7 and $+7$ $ton\ ha^{-1} yr^{-1}$. Given this large spatial variability it might be difficult to prove that the SOC stock has truly increased, if all that we have is a sample of n SOC stock observations from the area. Let us explore this in the next sections.

6.1 Static sampling

In static sampling we visit the same locations in 1990 and 2020. This means that we can calculate the SOC stock difference at the n sampling locations and estimate the population mean (SOC stock difference for the example area) with the sample mean. In other words, we can apply exactly the same methods as used in Chapter 4 and Chapter 5, but now for a different population characteristic (SOC stock change instead of SOC stock). This is done in the code chunk below for the case of a simple random sample of size 400. Note that the variance of the estimation error is calculated in the same way as in Chapter 3, only with SOC replaced by ΔSOC (i.e., the SOC stock change from 1990 to 2020):

$$var(\hat{\mu}_{\Delta SOC,static} - \mu_{\Delta SOC}) = \frac{\sigma^2_{\Delta SOC}}{n}$$

Generate a random sample

```
# define the sample size
sample_size <- 400

# sample n non-NA locations from soc_2020
sample_2020 <- spatSample(soc_2020, size = sample_size,
                          method = "random", as.points = TRUE,
                          values = TRUE, na.rm = TRUE)

# use the same locations to sample from soc_1990
# when bind = TRUE, it retains the values extracted from soc_2020
sample_extract <- as.data.frame(
  extract(soc_1990, sample_2020,
          ID = FALSE, bind = TRUE))

# compute the difference
random_sample <-
  sample_extract[, "soc_2020"] - sample_extract[, "soc_1990"]
```

Calculate estimate and confidence interval

```r
# compute the average SOC stock change for this sample (ton/ha)
sample_mean <- mean(random_sample) * area_ha

# compute the standard error
std_change <- global(soc_change, fun = "std", na.rm=TRUE)
SE <- std_change/sqrt(sample_size) * area_ha

# compute the lower and upper limit of the confidence interval
lower_limit <- sample_mean - 1.96*SE
upper_limit <- sample_mean + 1.96*SE

# is zero inside the confidence interval?
check_zero_intv <- lower_limit < 0 & 0 < upper_limit

# print - units in tons
paste0("Estimated SOC stock change: ", round2(sample_mean, 0, 10)," ton")
paste0("95% confidence interval: [", round2(lower_limit, 0, 10), ", ",
      round2(upper_limit, 0, 10), "]")
paste0("Confidence interval width: ",
       round2(upper_limit - lower_limit, 0, 10)," ton")
paste0("Is zero SOC stock change inside the confidence interval: ",
       check_zero_intv)
```

```
[1] "Estimated SOC stock change: 34970 ton"
[1] "95% confidence interval: [26420, 43510]"
[1] "Confidence interval width: 17080 ton"
[1] "Is zero SOC stock change inside the confidence interval: FALSE"
```

The estimated SOC stock change is quite close to the true SOC stock change, so it appears that a sample size of 400 is sufficient to obtain a fairly accurate estimate. This is confirmed by the confidence interval, which is not that wide. Note also that the 95% confidence interval does not include zero, which means that we can state with 95% confidence that the true SOC stock of the example area has indeed increased from 1990 to 2020. In other words, the estimated increase is **statistically significant**.

6.2 Synchronous sampling

In synchronous sampling a different set of sampling locations is selected for the two sampling years. Here, we take a simple random sample of size $n = 400$ in 1990 and another simple random sample of size $n = 400$ in 2020. The estimate of the SOC stock change from 1990 to 2020 is then obtained by subtracting the first estimate from the second:

Generate synchronous random samples

```
# define the sample size
sample_size <- 400

# sample n non-NA locations
random_sample_1990 <- spatSample(soc_1990, size = sample_size,
                                 method = "random", as.points = TRUE,
                                 values = TRUE, na.rm = TRUE)

random_sample_2020 <- spatSample(soc_2020, size = sample_size,
                                 method = "random", as.points = TRUE,
                                 values = TRUE, na.rm = TRUE)
```

Calculate synchronous sample mean

```
# select the first column, which contains the extracted soc values
random_sample_1990 <- as.data.frame(random_sample_1990)[[1]]
random_sample_2020 <- as.data.frame(random_sample_2020)[[1]]

# compute the average SOC stock for this sample (ton/ha)
sample_mean_1990 <- mean(random_sample_1990)
sample_mean_2020 <- mean(random_sample_2020)

# sample mean difference
sample_mean_change <- sample_mean_2020 - sample_mean_1990

# multiply by size of area to covert to total SOC stock change
sample_mean_change <- sample_mean_change * area_ha

# print
round2(sample_mean_change, 0, 10)
```

```
[1] "36120"
```

As usual we also want to quantify the uncertainty of this estimate with a standard error and derive the associated confidence interval. For this we need to calculate the variance of the estimation error, which is given by:

$$var(\hat{\mu}_{\Delta SOC,syn} - \mu_{\Delta SOC}) = \frac{1}{n}(\sigma^2_{SOC,2020} + \sigma^2_{SOC,1990})$$

This equation holds because in statistics we have the equality

$$var(X - Y) = var(X) + var(Y) - 2\,cov(X, Y)$$

where in our case the covariance term is zero because the sampling locations in 1990 were selected independently from those in 2020. Using the equation above we can now calculate the standard error and confidence interval associated with synchronous sampling:

Calculate confidence interval of synchronous sampling

```
# terra::global() fun does not include variance,
# so let's use the standard deviation instead
std_1990 <- global(soc_1990, fun = "std", na.rm=TRUE)
std_2020 <- global(soc_2020, fun = "std", na.rm=TRUE)

# compute the standard error
SE <- sqrt((std_1990^2 + std_2020^2) / sample_size) * area_ha

# compute the lower and upper limit of the confidence interval
lower_limit <- sample_mean_change - 1.96*SE
upper_limit <- sample_mean_change + 1.96*SE

# is zero inside the confidence interval?
check_zero_intv <- lower_limit < 0 & 0 < upper_limit

# print - units in tons
paste0("Estimated SOC stock change: ", round2(sample_mean_change, 0, 10))
paste0("95% confidence interval: [", round2(lower_limit, 0, 10), ", ",
      round2(upper_limit, 0, 10), "]")
paste0("Confidence interval width: ",
       round2(upper_limit - lower_limit, 0, 10)," ton")
paste0("Is zero SOC stock change inside the confidence interval: ",
       check_zero_intv)
```

```
[1] "Estimated SOC stock change: 36120"
[1] "95% confidence interval: [24860, 47380]"
[1] "Confidence interval width: 22520 ton"
[1] "Is zero SOC stock change inside the confidence interval: FALSE"
```

Also in the synchronous sampling case a sample size of $n = 400$ is sufficient to show a statistically significant SOC stock increase from 1990 to 2020. But note that the estimation accuracy is somewhat lower than that of static sampling, as shown by the wider confidence interval.

7 Composite soil sampling

Previous chapters in this tutorial have shown that sampling theory from statistics is a useful technique to estimate the total or average SOC stock in an area as well as the total or average SOC stock change of that area between two points in time. The advantages of sampling theory are that the estimates are unbiased, model-free and that the estimation accuracy can be quantified by a standard error or confidence interval. But an important disadvantage is that to reach a desired accuracy a large sample size may be required, which is costly. We learnt that stratified random sampling is more efficient than simple random sampling, and for estimating changes a static design outperforms a synchronous design, but in practice one might still need to collect and analyse a large number of soil samples to reach a required accuracy level. In this chapter we discuss one more technique to reduce field and laboratory costs, by using **composite soil sampling**.

In composite soil sampling we merge soil samples taken from multiple locations, and mix them well before we analyse the composite soil sample in the laboratory on soil properties of interest, such as SOC. Usually, the merged soil samples are taken from nearby locations, thus eliminating short-distance spatial variability and effectively reducing the variance and standard deviation of the data. Recall from Chapter 3 that this will decrease the standard error, since the standard error is proportional to the population standard deviation. And since a smaller standard error leads to a smaller confidence interval we get that with the same sample size our estimate of the total SOC stock in an area is more accurate than if we had not used composite soil sampling. Let us verify this now and quantify the uncertainty reduction for estimation of the SOC stock change between 1990 and 2020 in the example area.

We will 'collect' composite soil samples from the example area by merging 9 soil samples in a 3 by 3 grid with grid distance 25 m. We will use a static design and so visit the same locations in 1990 and 2020. In other words, we calculate the average of the 9 SOC stock change values in a 3×3 window of the ΔSOC raster map and otherwise do exactly the same as we did in Chapter 6. We will use a sample size of $n = 100$, which is much smaller than the sample size ($n = 400$) used in Chapter 6. We are curious about the confidence interval that we will get: will it be wider than that in Chapter 6, similar in size, or perhaps even narrower?

As usual we start with running some setup lines. After that we randomly select the centre locations of 3×3 windows, assemble all cells in the window and calculate their average SOC stock change (while making sure that we do not go outside the study area and that we exclude NA cells).

Setup

```
# load library
library(terra)
```

```
# avoid scientific notation
options(scipen=999)
```

```
# set a seed
set.seed(2025)
```

```
# rounding function
round2 <- function(x, d, p = 1){
  p <- p*(10^-d)
  n <- round(x/p)*p
  format(n, nsmall = d)
  }
```

```
# function to compute raster min max values
get_minmax <- function(c_rasters){
  mimax_rasters <- global(c_rasters, fun = "range", na.rm = TRUE)
  min_range <- floor(min(mimax_rasters["min"]))
  max_range <- ceiling(max(mimax_rasters["max"]))
  rasters_range <- c("min" = min_range,
                     "max" = max_range)
  return(rasters_range)
}
```

Load data

```
# load raster from ISRIC WebDAV
url_folder <- "https://files.isric.org/public/tutorials/soc_stock_change/output/"
raster_fn <- "soc_stock_change.tif"
soc_change <- rast(paste0(url_folder, raster_fn))
```

Prepare data

```
# rename to make it easier to call
layer_name <- "soc_change"
names(soc_change) <- layer_name
```

Total area

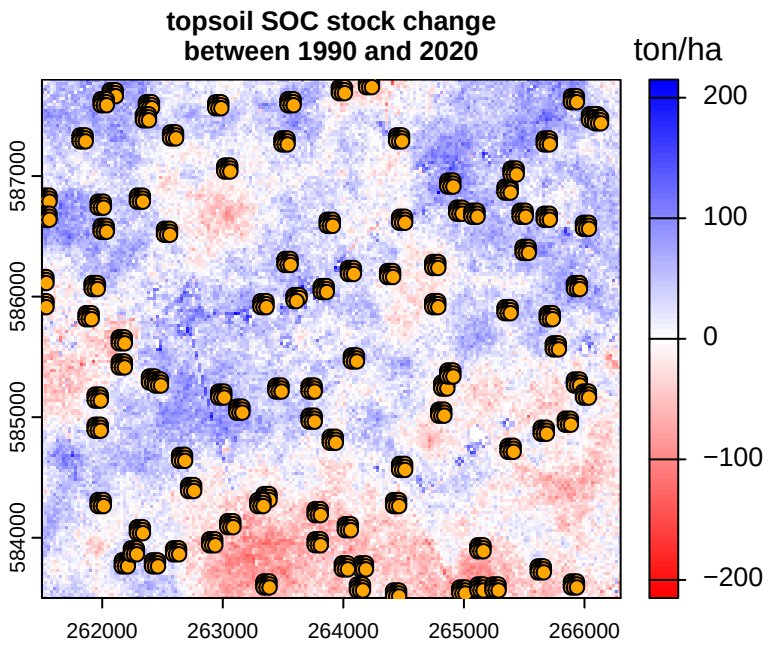
```
# terra::expanse() returns the area covered by all not-NA raster cells
area_ha <- expanse(soc_change, unit = "ha")[1, "area"]
```

Generate a random sample and its average in a 3 x 3 cell window

```r
# define the sample size
sample_size <- 100

# sample n non-NA locations
random_sample <- spatSample(soc_change, size = sample_size, method = "random",
                            as.points = TRUE, values = TRUE, na.rm = TRUE)

# get the centre cell numbers of each point
center_cells <- cellFromXY(soc_change, crds(random_sample))

# function to compute the 3 x 3 window cell indices around each sampled cell
# "r" refers to the raster
get_window_cells <- function(cell, winside = 3, r) {

  # check that winside is an odd number  1
  if (winside %% 2 != 1 || winside < 1) {
    stop("winside must be an odd number  1")
  }

  # compute how many rows/cols to expand in each direction
  half_win <- floor(winside / 2)

  # get the row/column index of the central cell
  rc <- rowColFromCell(r, cell)

  # build the surrounding rows and columns
  rows <- (rc[1] - half_win):(rc[1] + half_win)
  cols <- (rc[2] - half_win):(rc[2] + half_win)

  # keep only valid indices
  rows <- rows[rows >= 1 & rows <= nrow(r)]
  cols <- cols[cols >= 1 & cols <= ncol(r)]

  # get cell numbers for the window
  cells <- cellFromRowColCombine(r, rows, cols)

  # remove cells that are NA in the raster
  values_in_cells <- r[cells]
  valid_cells <- cells[!is.na(values_in_cells)]

  return(valid_cells)
}

# compute the window of each sampled cell
window_cells_list <- lapply(center_cells, get_window_cells,
                            winside = 3, r = soc_change)
```

Plot random sample and its 3 x 3 cell window

```
# convert each set of cell indices to XY coordinates
xy_list <- lapply(window_cells_list, function(cells) {
  xyFromCell(soc_change, cells)
})

# convert xy_list to a SpatVector
window_points <- vect(xy_list, crs = crs(soc_change))

# find the min-max values in the soc stock change raster
diff_range <- get_minmax(soc_change)

# get the absolute maximum to mirror the colour palette around it
max_abs <- max(abs(diff_range))

# indicate the palette colours
colour_palette <- colorRampPalette(c("red", "white", "blue"))

# define number of colour codes to generate from the colour palette
n_colours <- 100

# get the colour codes
colours <- colour_palette(n_colours)

# plot soc change
plot(soc_change,
     main = paste0("topsoil SOC stock change", "\n",
                   "between 1990 and 2020"), cex.main = 0.8,
     plg = list(title = "ton/ha"), col = colours,
     range = c(-max_abs, max_abs))
# add 3x3 window random sample points
plot(window_points, pch = 21, cex = 1,
     col = "black", bg = "orange", add = TRUE)
```



Compute the mean of each sampled 3 x 3 cell window and from it the SOC stock change estimate

```
# extract values and compute mean for each window
mean_3x3 <- sapply(window_cells_list, function(x) {
  vals <- soc_change[x][[layer_name]]
  mean(vals, na.rm = TRUE)
})

# calculate and print the estimated SOC stock change
comp_est <- mean(mean_3x3) * area_ha
paste0("Estimated SOC stock change: ", round2(comp_est, 0, 10), " ton")
```

```
[1] "Estimated SOC stock change: 29190 ton"
```

Note that for some windows we use averages of fewer than nine point values because some grid cells, such as those at the border of the study area, have NA cells around them. This implies that cells with NA cells around them have a smaller **inclusion probability**, that is they have a smaller probability to be included in the sample, because these cells are less often a neighbour of a non-NA cell. But note also that when they are included they have a

bigger weight, because their SOC change value is averaged with the SOC change of fewer other cells. Strictly speaking, the statistical inference should account for these effects, but since the window used here is relatively small the effects will not be large and we ignore them here.

To quantify the accuracy of the estimated SOC stock change above with a standard error and confidence interval we first need to calculate the population variance. Note that this is the same population variance as in Chapter 6. In Chapter 6 the population characteristic of interest was the SOC stock change at point locations, but in this chapter it is the SOC stock change of 3×3 windows. This characteristic has a different (i.e., smaller) population variance. The code below calculates the population standard deviation of this characteristic.

Compute population standard deviation

```
# define a nxn moving window (matrix of weights)
winside <- 3
w <- matrix(1, nrow=winside, ncol=winside)

# compute window means
window_means <- focal(soc_change, fun=mean, w = w,
                  na.policy='omit', # omit computing when a focal value is NA
                  na.rm=TRUE # exclude NA around focal cells
)

# focal() returns a raster, we compute its standard deviation
pop_sd <- global(window_means, fun = "std", na.rm=TRUE)[[1]] * area_ha
```

Compute standard error and confidence interval

```
# compute the standard error
SE <- pop_sd/sqrt(sample_size)

# compute the lower and upper limit of the confidence interval
lower_limit <- comp_est - 1.96*SE
upper_limit <- comp_est + 1.96*SE
width_confintv <- upper_limit - lower_limit

# print
paste0("Estimated mean (composite): ", round2(comp_est, 0, 10), " ton")
paste0("95% confidence interval: [", round2(lower_limit, 0, 10), ", ",
       round2(upper_limit, 0, 10), "]")
paste0("Confidence interval width: ", round2(width_confintv, 0, 10))
```

```
[1] "Estimated mean (composite): 29190 ton"
[1] "95% confidence interval: [13930, 44440]"
[1] "Confidence interval width: 30510"
```

Recall from Chapter 6 that with $n = 400$ we obtained a much smaller 95% confidence interval width of 17080 ton. This tells us that removing short-distance spatial variability by merging soil samples within a 3×3 window did not help that much. It was certainly not enough to justify a reduction of the sample size from $n = 400$ to $n = 100$, because we get a much wider confidence interval. Apparently, the short-distance spatial variability of the SOC stock change is not that large (see also Figure 6.3). We can verify this by comparing the population standard deviation of the SOC stock change at poin locations with that of the SOC stock change for 3×3 windows:

Compare population standard deviations

```
# population standard deviation of point values
point_sd <- global(soc_change, fun = "std", na.rm=TRUE)[[1]] * area_ha
# population standard deviation of window averages
window_sd <- global(window_means, fun = "std", na.rm=TRUE)[[1]] * area_ha

# print
paste0("Point value population standard deviation: ",
       round2(point_sd, 0, 10), " ton")
paste0("Window value population standard deviation: ",
       round2(window_sd, 0, 10), " ton")
```

```
[1] "Point value population standard deviation: 87160 ton"
[1] "Window value population standard deviation: 77830 ton"
```

The small reduction in the standard deviation tells us that merging soil samples in small spatial windows does not really pay off. But there is nothing that stops us from merging soil samples from far away locations. The R code below does this by randomly sampling 9 grid cells in the example area, averaging the SOC stock change of these 9 grid cells, repeating this procedure $n = 100$ times, and estimating the SOC stock change and the associated confidence interval from the sample of 100 averages of 9 point values each. We expect a narrower confidence interval than before, since we are not restricted to only removing short-distance spatial variation.

All calculations are done in the code chunk below. Note that the code below calculates the standard error using the **sample variance** instead of the **population variance**, since calcu-

lation of the population variance would be very cumbersome in this case. As mentioned in Chapter 3, in practice one would always use the sample variance instead of the population variance, because the latter is unknown. The two variances will be very similar in case of large sample sizes, but for smaller sample sizes a correction has to be made: the normal distribution must be replaced by a **Student t-distribution** with $n - 1$ degrees of freedom. This will slightly widen the confidence interval, as illustrated in the code below.

Random composite sampling

```
# indicate the composite size
composite_size <- 9

# indicate how many times to sample
n_sample <- 100

# initialize vector to store the mean of each iteration
sample_means <- c()

for (i in seq_len(n_sample)){
  # sample 9 non-NA locations
  random_sample <- spatSample(soc_change, size = composite_size,
                              method = "random", as.points = FALSE,
                              values = TRUE, na.rm = TRUE)

  # select the first column, which contains the extracted soc values
  random_sample <- random_sample[[1]]

  # compute the average SOC stock for this sample
  sample_mean <- mean(random_sample)

  # store in vector
  sample_means <- c(sample_means, sample_mean)
}

# compute sample mean and multiply to convert to ton
rdm_comp_est <- mean(sample_means) * area_ha
paste0("Estimated SOC stock change: ",
       round2(rdm_comp_est, 0, 10), " ton")
```

```
[1] "Estimated SOC stock change: 39520 ton"
```

Random composite standard error and confidence interval

```
# estimate standard deviation
rdm_est_sd <- sd(sample_means) * area_ha

# compute the standard error
SE <- rdm_est_sd/sqrt(n_sample)

# compute the lower and upper limit of the confidence interval
# we use the t-distribution instead of the normal distribution
# check yourself that qt(0.975, df = n_sample-1) is bigger than 1.96
lower_limit <- rdm_comp_est - qt(0.975, df = n_sample-1)*SE
upper_limit <- rdm_comp_est + qt(0.975, df = n_sample-1)*SE
width_confintv <- upper_limit - lower_limit

# print
paste0("Estimated mean (random composite): ", round2(rdm_comp_est, 0, 10), " ton")
paste0("95% confidence interval: [", round2(lower_limit, 0, 10), ", ",
       round2(upper_limit, 0, 10), "]")
paste0("Confidence interval width: ", round2(width_confintv, 0, 10))
```

```
[1] "Estimated mean (random composite): 39520 ton"
[1] "95% confidence interval: [34120, 44920]"
[1] "Confidence interval width: 10800"
```

We now get a confidence interval that is narrower than the one obtained in Chapter 6, even though we used a sample size of $n = 100$ instead of $n = 400$. In fact we could have anticipated this result, because what we effectively did is collect soil samples from $100 \cdot 9 = 900$ randomly selected locations. Thus, the confidence interval width that we obtained should be about $\sqrt{9/4} = 1.5$ times smaller than that obtained in Chapter 6, which is indeed the case (you may check this yourself).

This chapter showed that much can be saved on laboratory costs by taking composite soil samples, except when merging is confined to small spatial windows and there is little short-distance spatial variability.

Further savings could be obtained by using spectral SOC measurements, since these are much cheaper than wet-chemistry measurements. But spectral observations tend to have larger measurement errors than wet chemistry observations, and ideally these errors should be included in the statistical inference. In fact, wet chemistry measurements are also not free of errors. It was beyond the scope of this tutorial to account for field and laboratory measurement errors, and so we assumed that all SOC stock observations were error-free. But we advise that projects always check on the quality of the laboratory measurements, by letting laboratories analyse

a few anonymised duplicate soil samples and calculating summary statistics of the differences between duplicate observations.

8 Conclusion

This tutorial explained the use of sampling theory from statistics for estimation of the SOC stock and SOC stock change in an area. We covered simple random sampling and stratified random sampling, and also compared static and synchronous sampling for estimation of SOC stock change over time. We explained the theory but the emphasis was on practical application using the R language for statistical computing. Using synthetic data from an example area we could demonstrate how the theory is applied in practice and could verify the theoretical findings with numerical experiments.

Obviously there is much more to sampling theory than what was covered in this tutorial. Readers who wish to dig deeper into the topic are advised to read De Gruijter (2006) and Brus (2022). The first of these books also addresses model-based estimation of spatio-temporal averages and totals (including geostatistical modelling), while the second provides R scripts to all methods used in the book. Both books are at a higher level than this tutorial, but we hope that with this tutorial readers have become familiar with the topic and are well-prepared and eager to make a next step.

Readers who would like to learn more about the measurement and modelling of SOC stock changes in the context of Monitoring, Reporting and Verification (MRV) are invited to read Ceschia et al. (2025), Batjes et al. (2024), Smith et al. (2020) and visit irc-orcasa.eu.

References

Batjes, Niels H., Eric Ceschia, Gerard B. M. Heuvelink, Julien Demenois, Guerric le Maire, Rémi Cardinael, Cristina Arias-Navarro, and Fenny van Egmond. 2024. "Towards a Modular, Multi-Ecosystem Monitoring, Reporting and Verification (MRV) Framework for Soil Organic Carbon Stock Change Assessment." *Carbon Management* 15 (1): 2410812. https://doi.org/10.1080/17583004.2024.2410812.

Brus, D. J. 2022. *Spatial Sampling with r*. CRC Press. https://doi.org/10.1201/9781003258940.

Ceschia, E, A Ihasusta, A A Bitar, N Batjes, F van Egmond, G Heuvelink, C Paul-Victor, et al. 2025. "Monitoring, Reporting and Verification of Soil Organic Carbon Stock Changes in Arable Land: Cookbook for Assessment in Different MRV Contexts and Proposition of a Harmonised Approach." Deliverable D4.2 (EU ORCaSa project). INRAE, CNRS (CESBIO), ISRIC, CIRAD, Agrosolution, CSIRO, FMI.

De Gruijter, D. J. & Bierkens, J. J. & Brus. 2006. *Sampling for Natural Resource Monitoring*. Springer. https://doi.org/10.1007/3-540-33161-1.

Helfenstein, A., V. L. Mulder, M. J. D. Hack-ten Broeke, M. van Doorn, K. Teuling, D. J. J. Walvoort, and G. B. M. Heuvelink. 2024. "BIS-4D: Mapping Soil Properties and Their Uncertainties at 25 m Resolution in the Netherlands." *Earth System Science Data* 16 (6): 2941–70. https://doi.org/10.5194/essd-16-2941-2024.

Smith, Pete, Jean-Francois Soussana, Denis Angers, Louis Schipper, Claire Chenu, Daniel P. Rasse, Niels H. Batjes, et al. 2020. "How to Measure, Report and Verify Soil Carbon Change to Realize the Potential of Soil Carbon Sequestration for Atmospheric Greenhouse Gas Removal." *Global Change Biology* 26 (1): 219–41. https://doi.org/10.1111/gcb.14815.