SoilFER

Soil mapping
for resilient agrifood systems

Global Soil Nutrient and Nutrient Budgets maps (GSNmap) Phase I Technical Manual

Food and Agriculture Organization of the United Nations
Rome, 2022

Required citation:
FAO. 2022. *Global Soil Organic Carbon Map* – GSOCmap v.1.6. *Technical report.* Rome. https://doi.org/10.4060/cb9015en

ISBN 978-92-5-135899-3
© FAO, 2022

Cover design: ©FAO/Matteo Sala

Licence

Placeholder

Presentation and basics

Placeholder

How to use this book

0.1 Training material

0.2 Setting up the software environment

0.2.1 Digital Soil Mapping

0.2.2 Main concepts: SCORPAN model

Chapter 1

Data preparation for digital soil mapping in R

1.1 Introduction

Data preparation is one of the most critical and time-consuming steps in Digital Soil Mapping (DSM). Raw soil laboratory data typically contains numerous inconsistencies, errors, and redundancies that must be systematically identified and resolved before use in modeling. Poor data quality at this stage directly translates to poor prediction models and unreliable soil property maps.

This section demonstrates systematic approaches to soil data validation, cleaning, and harmonization using the **Kansas KSSL dataset** as a practical example. The goal is to transform raw soil measurements into clean, consistent data ready for Digital Soil Mapping analysis and modeling.

Raw soil laboratory data commonly contains the following types of quality issues:

- **Geographic errors**: Missing or out-of-bounds coordinates, coordinate swaps, and unit inconsistencies
- **Depth inconsistencies**: Missing soil depths, zero-thickness horizons ((bottom = top), invalid depth logic (bottom < top), overlapping intervals, and duplicate measurements

- **Laboratory data problems**: Missing values, out-of-range soil properties, texture validation failures (clay+silt+sand $\neq$ 100%)
- **Duplicate profiles**: Multiple measurement sequences at the same location with conflicting depth intervals
- **Logical inconsistencies**: Values that are physically or chemically impossible given soil science constraints

The Kansas KSSL dataset used throughout this section provides real-world examples of these issues, allowing you to practice identifying and correcting them systematically using reproducible R workflows.

Considerations

- Every soil dataset is unique and may have specific data-quality issues.
- Proper identification of quality problems requires prior understanding of the dataset's structure.
- When using a different database, the code shown here must be adapted to your actual database column names and data types.

1.2 Loading and exploring raw soil data

Before performing any data cleaning or validation, you must first understand the structure and content of your raw data. This exploratory phase reveals data types, identifies potential problems, and informs your validation strategy.

1.2.1 Basic data loading and structure

```r
# Load libraries
library(tidyverse)        # Data manipulation and visualization
library(readxl)           # Read Excel files
library(knitr)            # For formatted tables

# Read Excel file containing raw soil data
raw_data <- read_excel("../../01_data/module1/KSSL_data.xlsx", sheet = 1)
```

```r
# Define the folder to store the results of the exercise
output_dir <-"../../03_outputs/module1/"

# Create the output directory if not existing
if (!file.exists(output_dir)){
  # create a new sub directory inside the main path
  dir.create(output_dir)
}

str(raw_data) # Examine the structure of the data
head(raw_data, 10) # Show the first 10 rows
summary(raw_data[,1:25]) # Summarize the data in 25 first columns
```

This dataset contains four types of information: *site* data (location and depth), *laboratory* data (measured soil properties), and *spectral* data (mid-infrared, MIR, reflectance). In addition, each row includes identifiers for the sample (`smp_id`), and other keys related to the spectral analyses (`join_key`, `scan_path_name`, `file_name`). In many cases, soil datasets do not include spectral data. Understanding these components will help you organize the cleaning process systematically.

1.2.2 Data components and organization

Typically, a raw soil dataset includes several types of information:

- **Site information** defines where and at what depth a soil sample was collected:
  - Geographic coordinates (longitude and latitude, often in WGS84 or X, Y in Cartesian projected systems)
  - Unique identifiers (profile ID, horizon ID, sample ID)
  - Depth boundaries (top and bottom depth of the analyzed horizons in each soil profile)
- **Laboratory data** contains measured soil properties:
  - Physical properties (texture: clay, silt, sand percentages)
  - Chemical properties (pH, organic carbon, cation exchange capacity)
  - Other analyzed parameters (salinity, nutrients, etc.)
- **Spectral data** (optional) includes:
  - Spectral reflectance values at multiple wavelengths (MIR, VIS-NIR)

– Used for predicting soil properties through spectroscopy

1.3 Preparing site data for analysis

Data cleaning involves extracting relevant information from raw data, standardizing column names, assigning unique identifiers, and validating data quality. To track changes through this process, we add a unique row identifier to the raw dataset before making any modifications.

We operationally define a **site** as the set of horizons or layers that share the same geographic location (i.e., identical coordinates at the chosen precision). The site has information on location (lat and long coordinates), depth (upper and lower depth boundaries) and metadata-related information (horizon ID).

1.3.1 Adding a unique row identifier

During data cleaning, some records may be removed, merged, or modified. Good practice is to add a unique sequential `rowID` to the raw (unchanged) dataset so you can track each original record throughout the entire workflow. This identifier allows you to trace any result back to its source data and understand which raw records were retained or excluded.

```r
# Add unique row identifier to track individual records through processing
raw_data <- raw_data %>%
  mutate(rowID = row_number(), .before = 1)
```

The `rowID` column preserves the link between processed data and raw data, enabling full transparency and reproducibility in your cleaning workflow.

1.3.2 Extracting and standardizing column names for sites

Raw soil data often uses inconsistent column naming conventions that vary between data sources, laboratories, and surveys. Standardizing these names prevents errors and makes your code more readable and reusable across different projects

The R object `site` will store the location, depth, and metadata from the full set of observations, including all sites and their associated horizons or layers in the database.

```
# Select only the columns needed for site data preparation
site <- raw_data %>%
  select(
    rowID,
    Long_Site.x,              # Raw column name for longitude
    Lat_Site.x,               # Raw column name for latitude
    smp_id,                   # Sample/horizon identifier
    Top_depth_cm.x,           # Top depth in centimeters
    Bottom_depth_cm.x         # Bottom depth in centimeters
  )

  # Rename columns to standard, consistent names
site <- site %>%
  rename(
    lon = Long_Site.x,
    lat = Lat_Site.x,
    HorID = smp_id,
    top = Top_depth_cm.x,
    bottom = Bottom_depth_cm.x
  )
```

1.3.3 Creating unique profile identifiers

Each unique geographic location represents a single soil profile. A profile may contain multiple horizons or layers sampled at different depths. We create a unique identifier `ProfID` for each location based on coordinates so that all horizons from the same location can be grouped together in the same profile.

```
site <- site %>%
  # Group all horizons at the same location
  group_by(lon, lat) %>%
  # Assign sequential ID to each unique location (cur_group_id() returns group nu
  mutate(ProfID = cur_group_id()) %>%
  ungroup() %>%
  # Format as standardized IDs: PROF0001, PROF0002, etc. with 4 digit resolution
```

```r
  mutate(ProfID = sprintf("PROF%04d", ProfID))

  # Reorder columns for clarity
site <- site %>%
  select(rowID, ProfID, HorID, lon, lat, top, bottom)
```

After this step, horizons from the same location share a common `ProfID`, while those from different locations receive distinct `ProfID` values. When unique profile IDs are not available in the raw data, this method reconstructs profile identity using the geographic position of the samples.

1.3.4 Remove exact duplicate site records

Multiple site records may correspond to identical observations (i.e., the same location and depth). In this dataset, spectroscopic measurements were performed four times per sample, resulting in duplicate entries. Remove duplicates and retain a single record to prevent redundancy and double counting in subsequent analyses.

```r
# Remove exact duplicate rows
site <- site %>%
  distinct(across(-rowID), .keep_all = TRUE)
```

Standard column naming convention ensures consistency across all your code and projects:

- `ProfID`: Profile identifier
- `HorID`: Horizon or sample identifier
- `lon`, `lat`: Geographic coordinates (decimal degrees, WGS84)
- `top`,`bottom`: Depth boundaries in centimeters

Adjust `ProfID` for locations with multiple profile descriptions

- Horizons or layers measured at different times or for different purposes may occur at the same location. If the `ProfID` is based only on spatial position, these observations may share the same identifier.
- Identify these profiles and assign a unique `ProfID` to each one (see the procedure in Section 6 of this chapter).

- Proper identification of unique profiles is necessary to ensure consistent data management and reliable Digital Soil Mapping results.

1.4 Coordinate validation and correction

Spatial coordinates form the basis of all spatial analyses. Invalid or inaccurate coordinates result in erroneous maps and unreliable spatial predictions. Systematic validation is therefore essential to identify and correct coordinate errors before they propagate through the analytical workflow.

Coordinates can be expressed in different coordinate reference systems (CRS):

- **Geographic coordinate**s (longitude and latitude in decimal degrees, typically WGS84)
- **Projected coordinates** (X and Y in a projected system, such as UTM or a local projection)

Any dataset should explicitly document which CRS is used. For Digital Soil Mapping workflows, it is recommended to standardize all coordinates to WGS84 (EPSG:4326) geographic coordinates (longitude and latitude) to ensure interoperability and consistency across projects.

1.4.1 Check 1: Missing coordinates

Records lacking valid spatial coordinates cannot be georeferenced and must be excluded from spatial analyses. Check for missing or null values in both coordinate dimensions (e.g., latitude/longitude or projected X/Y) before proceeding.

```r
# Remove records with missing coordinates
site <- site %>%
  dplyr::filter(!is.na(lon) & !is.na(lat))
```

1.4.2 Check 2: Valid coordinate ranges

For geographic coordinates (longitude and latitude expressed in decimal degrees), values must fall within the following absolute ranges:

Longitude: -180° to +180° (negative = West, positive = East) **Latitude**: -90° to +90° (negative = South, positive = North)

Values outside these limits indicate invalid or incorrectly formatted coordinates.

The following validation routine applies only to geographic coordinates stored in decimal degrees.

```r
# Keep only rows with valid lon/lat geographic coordinates inside valid ranges
site <- site %>%
  dplyr::filter(
    lon >= -180, lon <= 180,
    lat >=  -90, lat <=  90
  )
```

If projected coordinates are used, valid ranges depend on the specific projection and the spatial extent of the study area. In such cases, define the expected bounds for the coordinate reference system (CRS) before performing range checks.

Coordinate system considerations:

- Geographic coordinates (longitude/latitude in decimal degrees) have fixed global bounds (–180° to +180°, –90° to +90°) that must not be exceeded.
- Projected coordinates (e.g., X/Y in meters, such as UTM) require CRS-specific limits.
- To minimize CRS-related errors, standardize coordinates to WGS84 (EPSG:4326) early in the workflow. Document the original CRS and any transformations applied to ensure reproducibility and transparency.

1.5 Soil depth validation and correction

Depth intervals define the soil layer represented by each observation. Missing or inconsistent depth information prevents harmonization across profiles and can lead to incorrect interpretation of depth-dependent patterns and model outputs. This section describes a set of quality-control checks to validate and, where possible, correct depth interval data. The checks are applied sequentially, removing structurally invalid records before attempting logical corrections.

Understanding depth conventions

Soil horizons (or layers) are represented as depth intervals with explicit upper and lower boundaries, typically recorded as top depth and bottom depth (e.g., "0–15 cm" indicates measurements from 0 to 15 cm below the soil surface). The following rules apply:

- Depth boundaries define a continuous interval in the soil profile.
- Bottom depth must be strictly greater than top depth.
- Depths are measured downward from the soil surface (0 cm).
- Horizon thickness is computed as bottom – top and must be positive.

1.5.1 Check 1: Missing depth boundaries

Records with missing top or bottom depths cannot be assigned to a valid interval and should be excluded unless depth information can be recovered from the source.

```
# Keep records where `top` or `bottom` are not NA
site <- site %>%
  dplyr::filter(!is.na(top) & !is.na(bottom))
```

1.5.2 Check 2: Negative depth values

Depth values must be non-negative. Negative depths indicate invalid input and should be removed unless the error can be corrected using metadata or original field notes.

```
# Keep records where `top` or `bottom` are positive
  site <- site %>%
    filter(!(top < 0 | bottom < 0))
```

1.5.3 Check 3: Zero-thickness intervals

Horizons with top = bottom have zero thickness and do not represent a measurable soil layer. These records should be excluded from analysis unless the error can be corrected.

```
# Remove zero-thickness horizons
  site <- site %>%
    filter(!(bottom - top == 0))
```

1.5.4 Check 4: Invalid depth logic

For a valid depth interval, bottom > top must hold. Violations typically indicate data entry errors (e.g., swapped boundaries or incorrect units).

```
# Remove invalid depth logic
site <- site %>%
  filter(bottom > top)
```

1.5.5 Check 5: Profiles without a surface horizon (top > 0) · · · · todo

Each profile should represent the complete soil column starting at surface. This is essential for depth harmonization in a later step for Digital Soil Mapping.

```
## Keep only profiles that start at the surface (min top == 0)
site <- site %>%
  dplyr::group_by(ProfID) %>%
  dplyr::filter(!is.na(top) & min(top, na.rm = TRUE) == 0) %>%
  dplyr::ungroup()
```

Further checks for depth integrity required

After applying the basic depth validation steps above, some profiles may still contain duplicated or overlapping horizons, or multiple depth sequences for the same `ProfID`. These situations often arise from repeated sampling or laboratory analyses conducted at different times.

Do not remove these records prematurely. Duplicate horizons may contain complementary analytical measurements that are not consistently available across all repetitions. Eliminating them at this stage may result in unintended data loss.

Depth-sequence conflicts should therefore be resolved only after laboratory data have been cleaned, validated, and consolidated at the horizon level. Once analytical completeness has been ensured, apply the dedicated profile-level procedure described in **'Detecting and resolving duplicate soil profiles'** to identify and retain a single representative depth sequence per profile.

This staged approach prevents loss of valid measurements and ensures consistent profile harmonization.

1.6 Preparing lab data: harmonization and validation

Laboratory analyses provide the soil property measurements used as soil property inputs for Digital Soil Mapping. Laboratory data must be quality-controlled to ensure completeness, plausible value ranges, and internal consistency (e.g., particle-size fractions summing to ~100%). This section describes standard extraction, harmonization, and validation checks.

Laboratory analyses provide the soil property measurements used as inputs for Digital Soil Mapping. Prior to modelling, laboratory data must be quality-controlled to ensure:

(i) completeness of critical variables,
(ii) numeric data types for analytical parameters,
(iii) plausibility of measured values based on feasible analytical thresholds, and
(iv) traceable handling of potential errors (flagging, correction, or exclusion).

1.6.1 Prepare and standardize laboratory columns

In this step, you will extract horizon-level laboratory results from the raw dataset, convert relevant fields to numeric, and retain only the records associated with the profiles preserved in the cleaned site dataset. Finally, you will merge the laboratory data back into the site dataset using `rowID` as the unique record identifier.

```
# Extract laboratory columns with standardized names
lab <- raw_data %>%
  select(
    rowID,
    SOC, Carbon_Total,                               # Soil Organic Carbon and T
    Bulk.Density_1_3.BAR, Bulk.Density_ovendry,      # Bulk density at 1.3 bar a
    Sand, Silt, Clay,                                # Texture (%)
    pH,                                              # pH H2O
    CEC,                                             # CEC in cmol(+)/kg
    Nitrogen_Total,                                  # Total nitrogen (%),
    Phosphorus_Mehlich3, Phosphorus_Olsen, Potassium, # Available P (mg/kg), Exch
    Calcium_Carbonate_equivalent                     # CaCO3 equivalent (%)
  )

  # Ensure numeric type for all analytical parameters (prevents issues if stored
  lab <- lab %>%
  mutate(across(-rowID, as.numeric))

  # Keep only lab records that are present in the cleaned site dataset
  lab <- lab %>%
    filter(rowID %in% site$rowID)

  # Join both site and lab data by the common identifier 'rowID'
  site_lab <- site %>%
    left_join(lab, by = "rowID")
```

After preparing the laboratory dataset, the analytical results are validated through three quality-control checks:

- Check 1: Identify out-of-bounds values — flag measurements that fall outside plausible or admissible ranges.
- Check 2: Texture validation — verify that sand, silt, and clay values are internally consistent (e.g., within expected limits and summing appropriately when expressed as percentages).
- Check 3: Correction of out-of-bounds laboratory values — apply a proper correction strategy (targeted fixes where justified, or replacing suspect values with `NA`).

1.6.2 Check 1: Check each property against feasible analytical thresholds

Soil properties have physically and analytically plausible bounds. Values outside these bounds typically indicate measurement issues, unit inconsistencies, or transcription/data entry errors. Thresholds should be defined a priori and documented (e.g., based on existing soil datasets or peer-reviewed literature), then adjusted as needed for the target region and soil types.

1.6.2.1 Load property thresholds

Store thresholds in a configuration file (e.g., `property_thresholds.csv`) to ensure transparency and reproducibility.

```r
# -----
# Load valid ranges for soil properties
# NOTE: These thresholds are based on global soil datasets and/or literature.
#       Adjust for your specific region and soil types.
property_thresholds <- read_csv("property_thresholds.csv")
# Display thresholds
print(property_thresholds)
```

1.6.2.2 Identify out-of-bounds values

Each soil analytical property is evaluated against its min/max thresholds. Out-of-bounds values are compiled into a structured issue log (`out_of_bounds_issues`) to support inspection, reporting, and correction.

```r
# Identify out-of-bounds values
out_of_bounds_issues <- list()

for (i in seq_len(nrow(property_thresholds))) {
  prop <- property_thresholds$property[i]
  prop_desc <- property_thresholds$description[i]
  min_val <- property_thresholds$min_valid[i]
  max_val <- property_thresholds$max_valid[i]

  # Check property exists in the dataset
  if (prop %in% names(site_lab)) {
    x <- site_lab[[prop]]

    # Detect out-of-bounds: non-missing values outside [min_val, max_val]
    idx <- which(!is.na(x) & (x < min_val | x > max_val))

    if (length(idx) > 0) {
      out_of_bounds_issues[[prop]] <- tibble(
        rowID = site_lab$rowID[idx],
        property = prop,
        description = prop_desc,
        value = x[idx],
        min_valid = min_val,
        max_valid = max_val,
        issue = ifelse(
          x[idx] < min_val,
          paste0("Below minimum: ", round(x[idx], 2), " < ", min_val),
          paste0("Above maximum: ", round(x[idx], 2), " > ", max_val)
        )
      )
    }
  }
}
# Remove temporary objects
rm(i,idx,max_val,min_val, prop,prop_desc,x)
```

1.6.2.3 Reporting out-of-bounds for properties and audit trail

Generate a summary by property, identify records with issues (often indicative of systematic errors), and export a full QC report for review and documentation.

```r
# Report out-of-bounds if present
if (length(out_of_bounds_issues) > 0) {
  all_issues <- bind_rows(out_of_bounds_issues)
  cat("\n Out-of-bounds properties found\n")

  # Summary by property
  issue_summary <- all_issues %>%
    group_by(property, description) %>%
    summarise(
      count = n(),
      min_value_found = min(value, na.rm = TRUE),
      max_value_found = max(value, na.rm = TRUE),
      min_valid = first(min_valid),
      max_valid = first(max_valid),
      .groups = "drop"
    ) %>%
    arrange(desc(count))

  cat("Issues by property:\n")
  print(issue_summary)

  # Rows with multiple issues
  rows_with_multiple_issues <- all_issues %>%
    group_by(rowID) %>%
    summarise(
      n_issues = n(),
      properties = paste(property, collapse = ", "),
      .groups = "drop"
    ) %>%
    filter(n_issues > 1) %>%
    arrange(desc(n_issues))

  if (nrow(rows_with_multiple_issues) > 0) {
    cat("\n Records with MULTIPLE property issues:\n")
    print(head(rows_with_multiple_issues, 10))
    cat("\nThese records likely have data entry errors and should be reviewed.\n"
  }

  # Export QC report
  write_xlsx(
    list(
      Summary = issue_summary,
      Issues_by_record = rows_with_multiple_issues,
      All_issues = all_issues
    ),
    paste0(output_dir,"soil_property_validation_report.xlsx")
  )

  cat("\n Detailed report saved to: soil_property_validation_report.xlsx\n")

  rm(all_issues, issue_summary, rows_with_multiple_issues)

} else {
  cat("\n All soil properties within valid ranges!\n")
}
```

1.6.3 Check 2: Texture validation

Particle-size fractions should sum to approximately 100%. This check is used to flag potential inconsistencies (rounding, unit conversion issues, or data errors). The following code just flag inconsistencies in Particle-size fractions. Values failing this check should be reviewed rather than automatically removed.

```
# Print rows with texture validation incosistencies
texture_problems <- site_lab %>%
  mutate(
    texture_sum = Clay + Silt + Sand,
    texture_valid = abs(texture_sum - 100) < 2
  )

texture_problems <- texture_problems %>%
  filter(!texture_valid)

if (nrow(texture_problems) > 0) {
  cat(" Found", nrow(texture_problems),
      "records with invalid texture sums\n\n")
  print(texture_problems %>%
          select(rowID, ProfID, Clay, Silt, Sand, texture_sum))
  # Flag for review (do not automatically remove)
}
```

1.6.4 Check 3: Correction of out-of-bounds laboratory values

Out-of-bounds values identified during validation should be handled systematically and transparently. Corrections must follow clearly defined rules to avoid introducing subjective bias or undocumented changes.

Two complementary approaches are recommended:

Option 1 (preferred): targeted correction, when the error mechanism is known and the true value can be reasonably inferred (e.g., sign errors or unit-scaling mistakes).

Option 2: replacement with `NA`, when the true value cannot be reliably reconstructed. This preserves the observation while preventing propagation of erroneous measurements into subsequent analyses.

- **Option 1** requires a clear understanding of the relevant properties and thresholds to correctly identify errors, which can then be fixed with dedicated code.
- **Option 2** is a more drastic approach, as it will automatically replace all potential mistakes with `NA`. This can result in a critical decrease in records for some measured properties.
- Choosing between these options is a decision for the data manager. Handle both with care.

1.6.4.1 Option 1: Targeted corrections (when error mechanism is known)

Apply deterministic corrections only when the source of error is clearly understood and scientifically justified.

The summary of the critical values provides information on the nature of each issue.

```
for (property in names(out_of_bounds_issues)){
  cat("Total errors in",property, ":",n_distinct(out_of_bounds_issues[[property]]
  print(summary(data.frame(out_of_bounds_issues[property])[4]))
}
```

In the KSSL dataset provided, `SOC` is negative in 43 rows while `Phosphorus_Mehlich3` has wrong values in 1 row.

Example corrections shown below assume:
- Negative `SOC` values arise from sign errors, and
- Extremely high `Phosphorus_Mehlich3` values arise from unit scaling (e.g., ppb recorded instead of mg/kg).

```
# Correction: Negative SOC values
idx <- !is.na(site_lab$SOC) & site_lab$SOC < 0
if (any(idx)) site_lab$SOC[idx] <- abs(site_lab$SOC[idx])
# Remove temporary objects
rm(idx)

# Correction: Phosphorus Mehlich 3 > 2000 mg/kg (likely 1000× error - ppb instead
idx <- !is.na(site_lab$Phosphorus_Mehlich3) & site_lab$Phosphorus_Mehlich3 > 2000
n_idx <- sum(idx)
if (n_idx > 0) site_lab$Phosphorus_Mehlich3[idx] <- site_lab$Phosphorus_Mehlich3[
# Remove temporary objects
rm(idx, n_idx)
```

1.6.4.2 Option 2: Replace out-of-bounds values with `NA`

When values cannot be corrected with confidence, replace only the problematic measurements with `NA` while retaining the rest of the record.

```
# Loop through each property in the out_of_bounds_issues list
for (property in names(out_of_bounds_issues)) {
  # Get the rowIDs with issues for this property
  rowIDs_with_issues <- unique(out_of_bounds_issues[[property]]$rowID)
  # Change the values of the property in those rows to NA
  site_lab <- site_lab %>%
    dplyr::mutate(
      "{property}" := dplyr::if_else(rowID %in% rowIDs_with_issues,
                                     as.numeric(NA), .data[[property]])
    )
}
```

1.7 Resolving duplicated data in soil profiles

Duplicate or repeated soil profile descriptions may occur when the same location is sampled or analysed multiple times. These situations can produce multiple horizon sequences under the same `ProfID`, resulting in inconsistent depth intervals, overlapping layers, or conflicting analytical values.

Such inconsistencies must be resolved before depth harmonization and Digital Soil Mapping, as it requires one coherent and unique vertical profile per location.

The objective of this section is to:

-Detect duplicated or overlapping depth descriptions

-Merge replicated analytical measurements when appropriate

- Separate or select among competing depth sequences

-Retain only complete and internally consistent profiles

1.7.1 Why duplicates occur

Duplicates typically arise from:

- Re-sampling of the same location in later campaigns (temporal duplicates)
- Multiple laboratory analyses of the same sample (analytical replicates)
- Re-use of identifiers across merged surveys

These situations may produce:

- Multiple horizon sequences per `ProfID`
- Overlapping or conflicting depth intervals
- Repeated horizons with different values
- Ambiguity about which sequence should be used for modelling

1.7.2 Types of duplicate situations

It is important to distinguish between two fundamentally different cases:

- Case A — Replicated analyses of the same horizons (same depths)

```
- Identical top-bottom intervals in the same profile

- Multiple measurements of the same soil layer

- Depth integrity is preserved

- Typical in monitoring databases
```

- *Action*: If used for monitoring, use the data for the corresponding period; otherwise, merge measurements (e.g., average values)

- Case B — Multiple depth sequences (different depths)

```
- Different top-bottom structures within the same ProfID

- Represents independent profile descriptions
```

- *Action*: select one representative sequence

1.7.3 Check 1: Detect potential horizon duplicates within profiles

Profile identifiers `ProfID` have been previously constructed by grouping horizons with identical coordinates into the same profile. However, if multiple profile descriptions have been created at the same location, the `ProfID` will be the same. The objective now is to flag `ProfID` values that appear to contain more than one distinct depth sequence (i.e., multiple sets of horizon boundaries).

```r
### Detect potential horizon duplicates within profiles
profile_analysis <- site_lab %>%
  group_by(ProfID) %>%
  summarise(
    n_horizons = n(),
    n_unique_tops = n_distinct(top),
    n_unique_bottoms = n_distinct(bottom),
    max_depth = max(bottom, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  mutate(
    # If all horizons have unique top/bottom values,
    # depths are consistent (no duplicates)
    consistent = (n_unique_tops == n_horizons & n_unique_bottoms == n_horizons),
    likely_duplicates = !consistent
  )

# Find profiles with likely duplicates
duplicates <- profile_analysis %>%
  filter(likely_duplicates)

if (nrow(duplicates) > 0) {
  cat(" Found", nrow(duplicates),
      "profiles with likely duplicates measurement sequences\n\n")
  print(duplicates)
}

# Select all profiles presenting duplicate horizons
duplicates <- site_lab %>%
  filter(ProfID %in% duplicates$ProfID)
```

1.7.4 Resolving Profile Duplicates

Duplicate handling should follow the order below:

1 → merge duplicated horizons
2 → resolve competing depth sequences
3 → remove incomplete profiles

This order prevents premature data loss and preserves maximum analytical information.

1.7.4.1 Check 1: Average duplicated horizons (same depth intervals)

When multiple records share identical `ProfID`, `top`, and `bottom`, they represent repeated measurements of the same soil layer. These should be consolidated into a single horizon. Numeric properties are averaged and common identifiers are retained from the first occurrence.

```r
# -----
# Correction 1: Summarize property in duplicated horizons my mean
# (e.g. PROF0237, PROF0262, PROF0271, PROF0284, PROF0368)
# -----
site_lab <- site_lab %>%
  group_by(ProfID, top, bottom) %>%
  summarise(
    # keep identifiers as the first value in each group
    across(c(rowID, HorID, lon, lat), ~ first(.x)),

    # compute mean for all other numeric columns (NA-safe)
    across(
      where(is.numeric) & !any_of(c("rowID","HorID","lon","lat","top","bottom")),
      ~ if (all(is.na(.x))) NA_real_ else mean(.x, na.rm = TRUE)
    ),
    .groups = "drop"
  ) %>%
  select(names(site_lab))   # <- restores original column order
```

- Replicates increase measurement reliability.
- Averaging avoids discarding valid analytical information.

1.7.4.2 Check 2: Resolve ProfID for multiple depth sequences

Multiple independent profile descriptions likely exist in a single location. This data is valid and must remain in the dataset, but it has to be differentiated by its `ProfID`.

A `chain_horizons` function has been created to identify different sequences in horizons within each profile.

```r
# Detect ProfID series with different top-bottom depth sequences
# 1. For each profile, check if all rows form ONE continuous depth sequence
# 2. If YES → Single profile (done)
# 3. If NO → Find the consecutive horizons that ARE continuous
# 4. If we find blocks with no gaps → Split the series into subprofiles
# Create a function to identify sequences of horizons for each profile
chain_horizons <- function(top, bottom) {
  n <- length(top)
  remaining <- seq_len(n)
  chain_id <- integer(n)
  cid <- 1

  while(length(remaining) > 0) {
    # start new chain at smallest top
    cur <- remaining[which.min(top[remaining])]
    repeat {
      chain_id[cur] <- cid
      remaining <- setdiff(remaining, cur)
      nxt <- remaining[top[remaining] == bottom[cur]]
      if(length(nxt) == 0) break
      cur <- nxt[1]
    }
    cid <- cid + 1
  }
  chain_id
}

site_lab <- site_lab %>%
  group_by(lon, lat, ProfID) %>%
  mutate(chain = chain_horizons(top, bottom)) %>%   # detect sequences
  arrange(chain, top, .by_group = TRUE) %>%         # sort within each chain
  mutate(
    ProfID = paste0(ProfID, "_", chain)             # add a numeric suffix
  ) %>%
  ungroup()


if (max(site_lab$chain, na.rm = TRUE) > 1) {
  corrected_profiles <- unique(site_lab$ProfID[site_lab$chain >= 2])
  cat("→ Corrected depth continuity in", length(corrected_profiles), "profiles\n"
  cat("  Corrected Profiles:", paste(sub("_2$", "", corrected_profiles), collapse
} else {
  cat("→ No depth continuity corrections were needed\n")
}

# Delete the chain column
site_lab <- site_lab %>%
  select(-chain)

# Delete temporary objects
rm(corrected_profiles,chain_horizons)
```

1.7.5 Remove profiles not starting at the surface

Profiles whose shallowest horizon does not begin at 0 cm represent mistakes or incomplete descriptions and may bias depth harmonization and DSM modelling.

Only profiles whose first horizon begins at the surface (top = 0 cm) should be retained for analyses requiring complete vertical representation.

1.7.5.1 Check 1: Remove profiles with incomplete surface coverage

As a result of the previous steps, some horizons may remain without forming a continuous sequence within their profile. The objective here is to retain only those profiles whose shallowest recorded depth starts at 0 cm.

```
site_lab <- site_lab %>%
  group_by(ProfID) %>%
  filter(min(top, na.rm = TRUE) == 0) %>%
  arrange(ProfID, top, bottom, HorID) %>%
  ungroup()
```

1.7.6 Result

After this procedure, we obtain a clean horizon-level dataset containing validated site, analytical, and metadata information. The dataset contains:

- One consistent depth sequence per `ProfID`

- No duplicated horizons

- Consolidated analytical measurements
- Alternative profiles at the same location can coexist

This database can be exported to `csv` and `.xlsx` files.

```r
# Save to CSV
output <- paste0(output_dir,"KSSL_cleaned.csv")
write.csv(site_lab, output, row.names = FALSE)

# Save to Excel
output <- paste0(output_dir,"KSSL_cleaned.xlsx")
write_xlsx(site_lab, output)
```

1.8 Harmonizing data for DSM

As shown in the previous step, the database produced can still present several alternative profiles coexisting at the same location. These profiles present different top-depth valid horizon sequences that must be retained in the soil database. When multiple valid profiles exist at the same coordinates, DSM requires a single profile description at each single location. There are different profile selection criteria based in the data available and/or modeling purpose. Select the option using one of these criteria: :

```
    -   **Most complete**: Keep sequence with most horizons (more depth detail)
    -   **Best coverage**: Keep sequence extending deepest (most information)
    -   **Best quality** : Keep sequence with fewest missing values
    -   **Monitoring**   : Keep sequence analyzed at the period of interest (re
```

In general, for monitoring activities, profiles can be selected upon a time key (e.g., `Date`, `campaign_id`) and treat profiles as separate observations information. This will allow to model temporal trends, or time-specific DSM surfaces. When sampling dates or campaign metadata are unavailable, the **most complete** (more depth detail) sequence is typically safest.

```r
# Create a new object to store DSM harmonized data
# Keep most complete profiles at each location to avoid duplicated profiles
horizons <- site_lab %>%
  group_by(lon, lat, ProfID) %>%
  summarise(n_hz = n_distinct(paste(top, bottom)), .groups = "drop") %>%
  group_by(lon, lat) %>%
  dplyr::slice_max(n_hz, n = 1, with_ties = FALSE) %>%
  select(lon, lat, ProfID) %>%
  inner_join(site_lab, by = c("lon", "lat", "ProfID")) %>%
  ungroup()
```

DSM requires one profile per location

- If no `Date/campaign` metadata exists → keep the most complete profile per location.
- If `Date/campaign` exists and the purpose is monitoring → stratify by `Date/campaign` and keep one profile per location per `Date/campaign`

1.8.1 Depth standardization

Finally, DSM requires analytical data calculated at standardized depth intervals (the same depths for every profile). The standard depths typically used are 0–30 cm (topsoil), 30–60 cm (subsoil), and 60–100 cm (deep subsoil). These represent meaningful soil zones in terms of soil fertility, root penetration, weathering, and soil formation.

Since the cleaned dataset contains properties at variable-depth horizons for each profile, depth harmonization is needed.

The Algorithms for Quantitative Pedology (`aqp`) package provides the `slab()` function for standardizing variable-depth soil data using weighted averaging.

```
library(aqp)

# Define standard depth intervals
standard_depths <- c(0, 30, 60)  # 0-30, 30-60 cm

# Select properties to standardize
properties_to_standardize <- names(site_lab)[!names(site_lab) %in% c("rowID","Pro

# Prepare data for aqp

# Create SoilProfileCollection Object
# aqp needs profiles + depth structure for proper interpolation
```

```
depths(horizons) <- ProfID ~ top + bottom

# Add Spatial Information to SoilProfileCollection
#   Links geographic location to soil profiles
initSpatial(horizons, crs = "EPSG:4326") <- ~ lon + lat

# Build the standardization formula
fml <- as.formula(
  paste("ProfID ~", paste(properties_to_standardize, collapse = " + "))
)

# Apply slab() to interpolate to standard depths
KSSL_standardized <- slab(
  horizons,
  fml,
  slab.structure = standard_depths,  # Target standard depths
  na.rm = TRUE                        # Ignore NA values in calculations
)
```

The `slab()` function produces output with: - `p.q5`: 5th percentile (lower confidence bound) - `p.q50`: Median estimate (best estimate) - `p.q95`: 95th percentile (upper confidence bound)

The 5th to 95th percentile range provides a 90% confidence interval, quantifying uncertainty in the standardized estimates.

```
# Create Confidence Interval of the estimations (CI column)
# Shows range of uncertainty (p.q5 to p.q95)

KSSL_standardized <- KSSL_standardized %>%
  mutate(
    # Create 90% confidence interval string (p.q5 to p.q95)
    CI = paste0(
      round(p.q5, 3),                 # Lower bound (5th percentile)
      "-",
      round(p.q95, 3)                 # Upper bound (95th percentile)
    )
  )
```

1.8.2 Processing standardized data

The output of the `slab()` function is a data frame in long format, where soil properties are stored as rows and the estimated percentile values are stored as columns. Each record is identified by `ProfID`, `top`, `and bottom`. For most downstream analyses, the data must be reshaped to wide format (i.e., one row per depth interval, with properties as columns).

```
# Convert from long to wide format
KSSL_standardized <- KSSL_standardized %>%
  pivot_wider(
    id_cols = c(ProfID, top, bottom), # Keep these as-is
    names_from = variable,             # Property names become column names
    values_from = c(p.q50, CI),        # Both point estimate and CI
    names_glue = "{variable}_{.value}" # Create names like "SOC_p.q50", "SOC_CI"
  )

# Add geographic coordinates back
KSSL_standardized <- KSSL_standardized %>%
  # Get coordinates from original data (one per profile)
  left_join(
    site_lab %>%
      distinct(ProfID, .keep_all = TRUE) %>%  # One row per profile
      select(ProfID, lon, lat),
    by = "ProfID"
  ) %>%
  # Move coordinates to front for readability
  relocate(lon, lat, .after = ProfID)

# Since ProfIDs are now unique at each location, remove tailings ProfID values
KSSL_standardized$ProfID <- sub("_[12]$", "", KSSL_standardized$ProfID)

# Result: One row per profile-depth, with standardized soil properties
head(KSSL_standardized)
```

The final step is to save this dataset with standardized soil properties at two depths.

```
# Save to CSV
output <- paste0(output_dir,"KSSL_standardized.csv")
```

```r
write.csv(KSSL_standardized, output, row.names = FALSE)

# Save to Excel
output <- paste0(output_dir,"KSSL_standardized.xlsx")
write_xlsx(KSSL_standardized, output)
```

1.8.3 Exporting standardized data for Digital Soil Mapping

At this stage, the dataset is standardized to one row per profile and standard depth interval. Soil properties at these depths are derived from the original horizon measurements using depth-weighted averaging, and the output includes the percentiles of the weighted estimates (e.g., p05, p50, p95).

For the Digital Soil Modelling exercise in this tutorial, we will work with a subset focused on the topsoil (0–30 cm) and use the median (p50) estimates for the following properties: clay, silt, sand, SOC, and pH.

```r
# Keep only 0-30 cm depth and select relevant columns (Clay, Silt, Sand, SOC & pH
subset_data <- data %>%
  filter(top == 0 & bottom == 30) %>%
  select(
    ProfID,
    lon,
    lat,
    top,
    bottom,
    Clay = Clay_p.q50,
    Silt = Silt_p.q50,
    Sand = Sand_p.q50,
    SOC = SOC_p.q50,
    pH = pH_p.q50
  )

# Save to CSV
output_csv <- paste0(output_dir,"KSSL_DSM_0-301.csv")
write.csv(subset_data, output_csv, row.names = FALSE)
cat(" Saved to:", output_csv, "\n")
```

```
# Save to Excel
output_xlsx <- paste0(output_dir,"KSSL_DSM_0-30.xlsx")
write_xlsx(subset_data, output_xlsx)
cat(" Saved to:", output_xlsx, "\n")

cat(" Subset data ready for Digital Soil Mapping\n")
cat("  Output file: KSSL_DSM\n")
```

1.9 Preparing data for spectroscopy analyses

The original KSSL dataset includes visible–near infrared (vis–NIR) spectral observations associated with each soil horizon. To support spectroscopy-based estimation of soil properties, spectral observations must be integrated with the cleaned horizon dataset (`site_lab`) that contains unique and consistent soil profiles (`ProfID`), validated and harmonized horizon depths (`top`, `bottom`), corrected laboratory measurements.

Depth-consistent and quality-controlled reference data are essential for building robust spectral calibration models and avoiding bias introduced by duplicated horizons or invalid analytical values.

In this dataset, each soil sample/horizon was measured four times by spectroscopy. Therefore, after merging spectra to the cleaned horizon dataset, the resulting dataset is expected to contain approximately 4× more rows than the resulting cleaned dataset (subject to missing spectra or incomplete records).

Use a `left join` to preserve the cleaned horizon dataset as the reference. This ensures that every cleaned horizon remains in the merged dataset (even if spectra are missing), and no spectral-only records are introduced without corresponding cleaned horizon metadata. The join key for this operation is `HorID` in the cleaned dataset and `smp_id` in the spectral dataset. Then save the results as `-csv`and `.xlsx` files.

```
# Read and subset spectral data from the original dataset
raw_data <- read_excel("../../01_data/module1/MIR_KANSAS_data.xlsx", sheet = 1)
spec <- raw_data[,-c(1,2,4:22)]

# Merge site_lab data to the original Spectral data by their common IDs
```

```
site_lab_spec <- left_join (site_lab,spec, by=c("HorID"="smp_id") )

# Save to CSV
output <- paste0(output_dir,"KSSL_spectral_cleaned.csv")
write.csv(site_lab_spec, output, row.names = FALSE)

# Save to Excel
output <- paste0(output_dir,"KSSL_spectral_cleaned.xlsx")
write_xlsx(site_lab_spec, output)

# Remove spectral data object
rm(spec)
```

- Best practices and recommendations

- **Trust but verify**: Never assume data is correct. Always validate systematically.
- **Document everything**: Record what was removed, why it was removed, and how many records were affected. This documentation is essential for transparency and reproducibility.
- **Preserve data lineage**: Use unique row identifiers to track which raw records became which cleaned records. This allows you to trace any result back to its source data.
- **Be conservative with removal**: Only remove records if certain they are wrong. When uncertain, flag records for manual review rather than automatically excluding them.
- **Automate, don't manually edit**: Write code that performs all cleaning steps, rather than manually editing spreadsheets. Code-based approaches are reproducible, transparent, and less prone to error.
- **Save intermediate steps**: Keep clean versions after each major processing step. This allows you to backtrack if a decision doesn't work out.

- Common pitfalls to avoid

Pitfall	Problem	Prevention
Removing too much data	Biased results from non-random loss	Document removal rate; flag >30%
Skipping validation	Problems propagate to analysis	Use systematic checklists
Manual edits	Not reproducible, hard to verify	Everything in code
Ignoring depth issues	Impossible harmonization	Verify bottom > top for all
No documentation	Can't explain analysis later	Keep detailed notes
Overconfident correction	Guessing wrong fixes errors	Only correct if confident

1.10 Summary of exported files

File	Description
KSSL_cleaned	Clean horizon-level dataset with validated analytical data
KSSL_spectral_cleaned	Clean horizon-level dataset with validated analytical and spectroscopic data
KSSL_standardized	Depth-harmonized dataset (0–30 cm; 30–60 cm) of all soil properties
KSSL_DSM	Depth-harmonized dataset (0–30 cm) for Digital Soil Mapping
soil_property_validation_report	Detailed report of analytical properties outside valid ranges

References

Chapter 2

Sampling design for soil surveys

Placeholder

2.1 Introduction

2.2 Sampling methodologies for soil spatial survey

2.2.1 Selection of the sampling methodology

2.2.2 Inference Methods

2.2.3 Purpose of Sampling

2.2.4 Sampling Design Types

2.2.4.1 Example: Soils4Africa Sampling Design

2.3 SoilFER Sampling Design

2.3.1 Soil Properties

2.3.2 Environmental Covariates

2.3.3 Understanding the Methodology

2.3.3.1 PSU Selection Process

2.3.3.2 SSU and TSU Selection Process

2.3.3.3 Site Identification System

2.4 Tutorial Using R

2.4.1 Setting Up the Environment

2.4.1.1 Install Required Libraries

2.4.2 Define Variables and Parameters

2.4.3 Country ISO Code

2.4.3.1 Land Use Type

2.4.3.2 File Paths

2.4.3.3 Coordinate Reference System

2.4.3.4 Sample Size Calculation

2.4.3.5 Sampling Unit Definitions

2.4.3.6 Algorithm Parameters

2.4.4 Custom Functions

2.4.4.1 Covariate Space Coverage Function (CSIS)

2.4.4.2 TSU Generation Function

2.4.5 Load Country Boundaries and Legacy Data

2.4.5.1 Visualize Boundaries and Legacy Data

2.4.6 Load Environmental Covariates for PSUs

2.4.6.1 Optional Auxiliary Data

2.4.6.2 Load and Process Auxiliary Data

2.4.7 Load Main Environmental Covariates

2.4.7.1 Load and Process Soil Climate Data (Newhall)

2.4.7.2 Convert Categorical Variables to Dummy Variables

2.4.7.3 Merge All Covariate Layers

2.4.7.4 Crop and Save Covariates

2.5 Principal Component Analysis (PCA)

2.6 Organize the Sampling Universe

2.6.1 Load and Process Land Use Data

2.6.2 Exclude Protected Areas

2.6.3 Restrict to Accessible Slopes

2.6.4 Aggregate to 100m Resolution

2.6.5 Save Processed Land Use

2.6.6 Filter Legacy Data

2.7 Create Primary Sampling Units (PSU Grid)

2.7.1 Generate 2km $\times$ 2km Grid

2.7.2 Trim to Country Boundaries

2.7.3 Save PSU Grid

2.8 Select PSUs with Sufficient Land Use Coverage

2.8.1 Calculate Land Use Percentage

2.8.2 Filter PSUs

2.9 Rasterize PSUs for Covariate Space Coverage

2.10 Compute Optimal Sample Size

2.10.1 Run Optimization

2.10.2 Extract Optimal Sample Size

2.11 Covariate Space Coverage - Computing PSUs

2.11.1 Prepare Function Parameters

2.11.2 Perform K-means Clustering

2.11.3 Assign Clusters and Calculate Distances

2.11.4 Extract Selected PSUs

2.11.5 Visualize in Covariate Space

2.12 Compute SSUs and TSUs

2.12.1 Load High-Resolution Covariates

2.12.2 Process Each PSU

2.12.3 Combine SSUs and TSUs

2.13 Export Sampling Units

2.13.1 Create Cluster Raster

2.13.2 Join Cluster Info to TSUs

2.13.3 Create Site IDs

2.13.4 Export Shapefiles

2.14 Compute Alternative PSUs

2.15 Assign Country-Wide Unique IDs

2.16 Export Unified Datasets

2.17 Summary Statistics

2.18 Create Distribution Maps

2.19 GPS Device Preparation

2.19.1 Export to GPX Format

2.19.2 Load onto GPS Device

2.20 Sampling Procedure

2.21 Data Recording

2.22 Quality Control

Appendix A: Software and Data Sources

2.22.1 Software Requirements

2.22.2 Data Sources

Appendix B: Troubleshooting Common Issues

2.22.3 Memory Issues

2.22.4 CRS Misalignment

2.22.5 Empty Geometry

Appendix C: Acronyms and Abbreviations

Chapter 3

TBD

Chapter 4

Soil Data Preparation

4.1 Introduction

4.2 Data cleaning

4.3 Soil Data Harmonization

4.4 Concepts on digital exchange of soil-related data (ISO 28258)

4.5 Plot data

4.6 Profile data

4.7 Element data

4.8 Specimen data

4.9 Definition of laboratory procedures

4.10 Data transformation

4.11 Organizing soil data according to the GloSIS database

4.12 Tools for harmonization of soil databases within GloSIS

Chapter 5

Digital Soil Mapping and Modeling

Placeholder

5.1 Introduction to Digital Soil Mapping

5.2 The SCORPAN Framework

5.3 Statistical Theory for Predictive Soil Mapping

5.3.1 Mechanistic versus Empirical Approaches

5.3.2 Sources of Residual Variance

5.3.3 Covariates as Proxies of Soil-Forming Factors

5.3.4 The Universal Model of Soil Variation

5.3.5 Extending the Model: Space, Depth, and Time (3D+T)

5.3.6 Types of Soil Variables

5.3.7 Prediction Methods

5.3.8 Nonlinearity and Uncertainty

Chapter 6

Soil Data Sharing

6.1 Data sharing and export formats

6.2 Metadata

6.3 Web Services

References

The Global Soil Partnership (GSP) is a globally recognized mechanism established in 2012. Our mission is to position soils in the Global Agenda through collective action. Our key objectives are to promote Sustainable Soil Management (SSM) and improve soil governance to guarantee healthy and productive soils, and support the provision of essential ecosystem services towards food security and improved nutrition, climate change adaptation and mitigation, and sustainable development.

Australian Government
Department of Agriculture, Water and the Environment

AFACI
Asian Food & Agriculture Cooperation Initiative

Ministry of Finance of the Russian Federation

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Rural Development Administration