DSM Workflow Documentation

ISRIC - World Soil Information

2026-05-26

Table of contents

Welcome

This is a Digital Soil Mapping (DSM) workflow developed at ISRIC - World Soil Information.

This workflow starts from standardised inputs, it organises data for modelling, performs modelling with Random Forest and creates soil maps as final outputs, including pixel based uncertainty with Quantile Regression Forests.

It is a file-based workflow with each step needing specific files as inputs and creating specific files as outputs. Files are named and organised according to the main variables/options chosen by the user. A configuration file allows the user to keep track of the chosen options.

Start with Chapter 1 and follow the instructions:

- In 1.1 we give **installation** instructions for the workflow and required dependencies.
- In 1.2 we explain how to **prepare the input data**.
- In 1.3 we explain how to **run the workflow**. This process requires two files: 1) the configuration file a `YAML` script, that we will simply call *config file*, and 2) an `R` script, to which we will refer as *workflow script*.

In Chapter 2 we introduce the config file.

- In 2.1 we present the template config file.
- In 2.2, the options that can be set in the config file for running the seedlings are explained.

The workflow script in Chapter 3 compiles the commands that trigger the workflow to run.

Chapter 4 describes main conceptual aspects that are important for understanding the model.

Chapter 5 mentions key points about how data and functions are handled in the Seedling workflow.

Note that Chapter 4 and Chapter 5 provide a deeper look at the options in the config file.

Frequently Asked Questions (FAQ) can be found in Chapter 6.

The **licence** for this software is in Chapter 7:

> The code in this repository is released under the GPL3 license according to the ISRIC data policy.

Disclaimer

The code is provided without warranty. ISRIC is not obliged to provide user support, updates or "bug fixes" of any kind.

Feedback, bugs and questions

Feedback and bug reports are welcome. Please contact us at dsm at isric dot org

Citation

Genova, G., Poggio, L., Kempen, B., & Colman, B. (2024). DSM Workflow Seedling. ISRIC - World Soil Information. https://doi.org/10.17027/ISRIC-FSX2-2691

1 Getting started

1.1 Installation

Install the `isric.dsm.base` R package. This package and its system dependencies are the only requirement for the workflow to run. The installation steps described below were tested on versions **R.4.5.3** and **R.4.6.0**.

Follow these steps to install the package:

- Install the latest R version.
- Within an R session use the following options to install the package.

First, since `tdigest` is no longer available on CRAN, install it manually with:

```
if (!require('pak')) install.packages('pak')
pak::pak('hrbrmstr/tdigest')
```

Second, download `isric.dsm.base` either:

---

a. Directly from the git repository link

```
if (!require('remotes')) install.packages('remotes')
remotes::install_git('https://git.wur.nl/isric/dsm-general/dsm.workflows/seedling.git',
 ↪  subdir='isric.dsm.base', ref='0.4.2.1')
```

---

b. Or by first cloning the repository with

```
git clone --depth 1 --branch 0.4.2.1
 ↪  https://git.wur.nl/isric/dsm-general/dsm.workflows/seedling.git
```

and then running

```
if (!require('devtools')) install.packages('devtools')
devtools::install('./isric.dsm.base')
```

---

c. Or by using the tar.gz file contained in the repository

```
if (!require('remotes')) install.packages('remotes')
remotes::install_local('isric.dsm.base_0.4.2.1.tar.gz')
```

1.2 Inputs

This workflow takes the following "standard" inputs. *It is crucial to ensure the data is prepared correctly before trying to run the workflow.*

- Soil **profiles** (`.csv` file): Table relating each unique pair of coordinates to a unique identifier.
- Soil **layers** (`.csv` file): Table that relates each identifier to the upper and lower depth of the layer, and the value of the measured property.
- **covariates** directory (directory containing `.tif` files): Spatially continuous environmental rasters representing the soil forming factors.
- **mask** file (`.tif` file): Raster covering the area of interest for which we want to predict.

Profiles and layers are separated for database efficiency utilising the concepts of keys and relationships in database management.

1.2.1 Profiles

The soil profiles identify a sampling location, the table (stored in a `.csv` file) follows a standard structure where at least three columns must be present. These columns are: an identifier column (uniquely identifying a profile), and two columns with the coordinates. See Table 1.1 for an example. The user is free to choose the column name, since each column is explicitly assigned to the corresponding variable in the config file (see how to do this in Chapter 2). Listed below are the variables in the config file required for the workflow to read the profiles.

- `pid_name` uniquely identifies the profile and allows the connection with the **layers** file. In the example table below the column is named "pid" so in the config file there will be a line with `pid_name: "pid"`
- `easting_name` defines the easting coordinate (usually called x or longitude). In the example table below the column is named "x" so in the config file there will be a line with `easting_name: "x"`
- `northing_name` defines the northing coordinate (usually called y or latitude). In the example table below the column is named "y" so in the config file there will be a line with `northing_name: "y"`

Note that for the easting and northing coordinates *the Coordinate Reference System (CRS), `crs_profiles` in the config file, needs to be specified.*

Table 1.1: Example of "standardised" **profiles** input file

pid	x	y
ARKH10001	102.377027777778	48.0066388888889
ARKH10003	102.516527777778	48.1358055555556
ARKH10004	102.6765	48.0079166666667

1.2.2 Layers

The soil layers also adhere to a standard convention. See Table 1.2 for an example. The user is free to choose the column name, since each column is explicitly assigned to the corresponding variable in the config file (see how to do this in Chapter 2). Listed below are the variables in the config file required for the workflow to read the layers.

- `pid_name` links the layers file with the **profiles** file. In the example table below the column is named "pid" so in the config file there will be a line with `pid_name: "pid"`
- `lyrid_name` uniquely identifies the soil layer. In the example table below the column is named "lyrid" so in the config file there will be a line with `lyrid_name: "lyrid"`
- `top_layer_name` specifies the depths of the upper bounds of the soil layers. In the example table below the column is named "top" so in the config file there will be a line with `top_layer_name: "top"`
- `bottom_layer_name` specifies the depths of the lower bounds of the soil layers. In the example table below the column is named "bottom" so in the config file there will be a line with `bottom_layer_name: "bottom"`
- In addition to the previous columns, multiple `voi` (variable of interest) columns can be present in the layers file. In the config file, it is specified which voi to use for each run, since the **seedling only produces results for one voi at a time**. See the Table 1.2 as an example.

Table 1.2: Example of "standardised" **layers** input file

pid	lyrid	top	bottom	pH	Salt	Density	SOC
ARKH10001	1	0	5	5.09	0.03	0.85	2.1
ARKH10001	2	5	10	5.09	0.02	0.91	1.92
ARKH10001	3	10	30	5.32	0.01	0.74	1.75

1.2.3 Covariates

The covariates are a set of georeferenced images (rasters) in GeoTIFF (`.tif`) format. It is best (but not necessary) that these spatial layers have the same **CRS**, the same **resolution** and the same **extent** as the mask. If these are different, the workflow will automatically convert them to the mask's attributes.

Note if you intend to run the Seedling for more than one soil property then we recommend to preprocess the covariates to the same resolution and extent (as the mask), otherwise this is done in the Seedling each time, becoming computationally expensive.

1.2.4 Mask

To run the workflow, a **mandatory mask** raster layer in GeoTiff format is **needed**. As a design choice, the mask defines the area of interest for mapping (i.e. for which the soil variable of interest will be predicted). It also defines the CRS and spatial resolution of the predictive soil maps that are generated with the workflow.

The mask layer should provide **no data (NA)** in the areas where no prediction is needed such as built-up areas, water bodies, rocky areas or glaciers. In the config file, the parameter `NA_mask_flag` can be used to specify which value of the mask should be considered *no data*.

It is best if the mask has the same **CRS**, the same **resolution** and the same **extent** as the **covariates**. If this is not the case, then all the covariates are going to be reprojected and clipped to the mask (this will slow down the process). Note that if one or more covariates have missing data for a pixel, the predictions will be missing for that pixel as well.

If no mask is available, one of the covariates can be used.

1.3 Running the workflow

To run the workflow you will need the following two files:

1. A configuration file (.yaml)

2. A workflow script file (.R)

Currently, there are the following workflow variants:

- Numerical: A single numerical property per run, e.g. pH, OrgC.
  - config.yaml
  - run_workflow.R
- Compositional: Compositional variables where the sum of their components equals 100%, e.g. Sand, Silt and Clay, also known as Texture.
  - config_compositional.yaml
  - run_workflow_compositional.R

1.3.1 Configuration file

The configuration file file contains:

- Files and directories paths (input, output).
- Workflow parameters (variable names, model options).

All these values need to be **modified** and **checked** by the user in order to ensure that the workflow can run **without errors**.

1.3.2 Running the workflow

The workflow can be executed once the R package `isric.dsm.base` is installed and the config file has been filled-in, **fully revised** and saved.

To execute the steps of the workflow, run the `run_workflow.R` script. Make sure that the variable "config_location" is set to the correct location of the configuration file. The script can be run interactively by using an IDE (e.g. **RStudio** or **VSCode**), or it can also be called from the command line (running this command from the folder where `run_workflow.R` is located):

- on Windows (note you might need to modify the path to `Rscript.exe` to reflect the R version and location on your specific computer)

```
"C:\Program Files\R\R-3.4.3\bin\Rscript.exe" run_workflow.R
```

- on Linux

```
Rscript run_workflow.R`
```

You can also give the configuration file location as a command line argument as follows:

```
Rscript run_workflow.R --config_file path/to/config_file
```

1.3.3 Optional steps

There are optional steps in this workflow: **run_rfe**, **model_tune** and **model_cv** (see Chapter 4). If these steps are not needed, the variables `dorfe` and `dotune` in the config file (see Chapter 2) need to be set as **norfe** and **notune** respectively.

2 Configuration file

The configuration file is the heart of the workflow. Here inputs, outputs, parameters and options are specified. Some parameters could be specific to one of the modelling modalities (numerical, compositional).

There are various config file parameters that will need adjusting, the most important config file parameters to double check and adjust if needed are:

- Input data:
  - `profilesStandardFile`: File path to the profiles `CSV` table.
  - `layersStandardFile`: File path to the layers `CSV` table.
  - `covarsDir`: File path(s) to the directory where the covariates files are stored. The workflow will read the files within the directory with the extension specified in the `covariate_pattern` argument.
  - `maskFile`: File path to the mask raster layer.
- `voi`: variable of interest.
- `pid_name`: profile identifier column name.
- `lyrid_name`: layer identifier column name.
- `crs_profiles`: CRS of the profiles.
- `NA_mask_flag`: No data value for the mask.
- `easting_name` & `northing_name`: easting (X) & northing (Y) column names.
- `top_layer_name` & `bottom_layer_name` Top & Bottom layer column names.

Have a look at Section 2.2 for more details of the options of the config file.

2.1 Get the configuration file

There are **two ways** you can get the configuration file:

2.1.1 Option A

Go to one of the following following links and download one (or more) of the files:

- `template.config.yaml` https://git.wur.nl/isric/dsm-general/dsm.workflows/seedling/-/blob/main/template.config.yaml
- `template.config_compositional.yaml` https://git.wur.nl/isric/dsm-general/dsm.workflows/seedling/-/blob/main/template.config_compositional.yaml

Then save it as `config.yaml` (or any other name, but it must be reflected in the `run_workflow.R` file)

2.1.2 Option B

Copy the following text in a file and save it as `config.yaml`

Numerical

```
# Version 0.4.2

# Input/Output files definition
outputDir: "./output/"
profilesStandardFile: "./input_example/points/geul_profiles_4326.csv"
layersStandardFile: "./input_example/points/geul_layers.csv"
covarsDir: "./input_example/covariates"
maskFile: "./input_example/other/geul_mask.tif"

# The variable of interest
voi: "pb"

voi_parameters:
  voi_min_value: 0
  voi_max_value: Inf
  transformation:  "notransform"  # "log"

# Dataset variables
pid_name: "pid"
lyrid_name: "lyrid"
easting_name: "x"
northing_name: "y"
top_layer_name: "top"
bottom_layer_name: "bottom"
flag_validation_name: "flag_validation" # "NULL" #
flag_tuning_name: "NULL"
flag_rfe_name: "NULL"
```

```yaml
flag_external_validation_name: "flag_external_validation" # "NULL"
depth_column_name: "depth"
fold_name: "fold"

# Model parameters
depth_name: "0-20" # "3D" #
depths_3D:
  - "0,5,15,30,60,100,200"
use_rfe: "dorfe"
do_tune: "tune"
model_name: "rangerquantreg" # "xgboostreg" #
n_folds: 10
tune_metric: "mec"

# Covariates information. Coordinate reference system definition
crs_profiles: "EPSG:4326"
crs_out: null
NA_mask_flag: null
crop_profiles_to_mask: true
covariate_pattern: ".tif$"

# Output options
multiply: 10
datatype_geotiff: "INT2U" #  For the output maps: values accepted are
 ↪  "INT1U", "INT2U", "INT2S", "INT4U", "INT4S", "FLT4S", "FLT8S". With GDAL
 ↪  >= 3.5 you can also use "INT8U" and "INT8S". And with GDAL >= 3.7 you can
 ↪  use also use "INT1S". See gdal to discover the GDAL version you are
 ↪  using. The first three letters indicate whether the datatype is an
 ↪  integer (whole numbers) of a real number ("float", decimal numbers), the
 ↪  fourth character indicates the number of bytes used for each number.
 ↪  Higher values allow for storing larger numbers and/or more precision; but
 ↪  create larger files. The "S" or "U" indicate whether the values are
 ↪  signed (both negative and positive) or unsigned (zero and positive values
 ↪  only).
gdal_options:
  - "COMPRESS=DEFLATE"
  - "PREDICTOR=2"
gdal_output_driver: "COG" # "GTiff" # "COG" is availbale with gdal >= 3.1
overwrite: true

# Tiling and cores to parallelize
tileside: 200
```

```yaml
cores: 3

# Advanced options. Only modify if you know what you are doing

# Other advanced options
covariatesListFile: null # "config/covs_list_corr_0.85_all" # Either a file
 ↪  path or NULL. A file with the names of the covariates to use (file name
 ↪  without extention, one per line)
thick_cond: 0 # Numeric from 0 to 1. 1 selects profiles that have a maximum
 ↪  depth equal or greater than the max modeled depth, 0 retains all the
 ↪  profiles. Only for 2D modelling (e.g. depth_name : "0-20")
use_dggridR: true
dggrid_resolution: 6
#' Seed for random processes
rndseed: 982374923
terra_memfrac: 0.3 # (from 0.1 to 0.9) fraction of memory that terra can use
# terra's default is 0.6
multiband: false
rfe_number: 5
rfe_repeats: 10
rfe_selection: "best" # "tolerance" #
min_rfe_cvrts: 0 # integer. The minimum number of covariates to be used in
 ↪  the RFE process. Default to 0. If the number of covariates is less than
 ↪  this value, the RFE process will not be performed.


# Plotting options
plot_border: false
legend_title: ""
add_north: true
add_sbar: true
save_to_disk: true
plot_range: NULL # [50, 1000]
plot_col: NULL # ["#00A600", "#2DB600", "#63C600", "#A0D600", "#E6E600",
 ↪  "#E8C32E", "#EBB25E", "#EDB48E", "#F0C9C0", "#F2F2F2"]

model_options:
  rangerquantreg:
    parsnip_engine: "ranger"
    parsnip_importance: "impurity"
    parsnip_quantreg: true
    parsnip_mode: "regression"
```

```
    parsnip_mtry_range_min_multiplier: 0.5
    parsnip_mtry_range_max_multiplier: 3
    parsnip_trees_range: [500, 1000]
    ntree_default: 500
    stats: ["mean", "Q0.5", "Q0.05", "Q0.95"]
    stats_to_log_backtansform: ["mean"]
    stats_to_exp: ["Q0.5", "Q0.05", "Q0.95"]
    search_type: "regular" # "random" #
    size_random_search: 50
    levels_regular_search: 7

# Automatically generated file paths
profilesOverlayFile: "{outputDir}/points/profiles_overlay.csv"
profilesFoldsFile: "{outputDir}/points/profiles_folds.csv"
layersDepthProcessedFile:
 ↪  "{outputDir}/points/layers_depth_processed_{depth_name}.csv"
regressionMatrixFile:
 ↪  "{outputDir}/model/{voi}/regression_matrix_{depth_name}.csv"
rfeCovariatesFile:
 ↪  "{outputDir}/model/{voi}/rfe_{depth_name}_{transformation}.csv"
rfeMetricsProfileFile:
 ↪  "{outputDir}/model/{voi}/rfe_profile_{depth_name}_{transformation}.RDS"
tuningGridFile:
 ↪  "{outputDir}/model/{voi}/tunegrid_{model_name}_{depth_name}_{transformation}_{use_rfe}.RD
bestParamFile:
 ↪  "{outputDir}/model/{voi}/bestparam_{model_name}_{depth_name}_{transformation}_{use_rfe}.c
cvPredictObserveFile:
 ↪  "{outputDir}/model/{voi}/cv-predict-observe_{model_name}_{depth_name}_{transformation}_{u
externalPredictObserveFile:
 ↪  "{outputDir}/model/{voi}/external-predict-observe_{model_name}_{depth_name}_{transformati
cvMetricsFile:
 ↪  "{outputDir}/model/{voi}/cv-metrics_{model_name}_{depth_name}_{transformation}_{use_rfe}_
internalMetricsFile:
 ↪  "{outputDir}/model/{voi}/internal-metrics_{model_name}_{depth_name}_{transformation}_{use
externalMetricsFile:
 ↪  "{outputDir}/model/{voi}/external-metrics_{model_name}_{depth_name}_{transformation}_{use
modelFittedFile:
 ↪  "{outputDir}/model/{voi}/model-fitted_{model_name}_{depth_name}_{transformation}_{use_rfe
tilesFile: "{outputDir}/other/tiles.csv"
tilesFileGpkg: "{outputDir}/other/tiles.gpkg"
tilesDir:
 ↪  "{outputDir}/maps/{voi}/{model_name}_{depth_name}_{transformation}_{use_rfe}_{do_tune}"
```

```
predictedMapVrt:
 ↪  "{outputDir}/maps/{voi}/{model_name}_{depth_name}_{transformation}_{use_rfe}_{do_tune}/{
plotDir:
 ↪  "{outputDir}/plots/{voi}/{model_name}_{depth_name}_{transformation}_{use_rfe}_{do_tune}"
```

Compositional

```
# Version 0.4.2

# Input/Output files definition
outputDir: "./output/"
profilesStandardFile: "./input_example/points/geul_profiles_4326.csv"
layersStandardFile: "./input_example/points/geul_layers.csv"
covarsDir: "./input_example/covariates"
maskFile: "./input_example/other/geul_mask.tif"

# The variable of interest
voi: ["fsand","fsilt","fclay"]

voi_parameters:
  voi_min_value: 0
  voi_max_value: Inf
  transformation:  "notransform"  # "log"

alr_prefix: "alr"
alrinv_scale_factor: 100 # Use 100 if your data ranges from 0 to 100, use 1
 ↪  if your data ranges from 0 to 1.

# Dataset variables
pid_name: "pid"
lyrid_name: "lyrid"
easting_name: "x"
northing_name: "y"
top_layer_name: "top"
bottom_layer_name: "bottom"
flag_validation_name: "NULL" # "flag_validation"
flag_tuning_name: "NULL"
flag_rfe_name: "NULL"
depth_column_name: "depth"
fold_name: "fold"
```

```yaml
# Model parameters
depth_name: "0-20" # "3D" #
depths_3D:
  - "0,5,15,30,60,100,200"
use_rfe: "dorfe"
do_tune: "tune"
model_name: "comprangerquantreg" # "xgboostreg" #
n_folds: 10
tune_metric: "mec"

# Covariates information. Coordinate reference system definition
crs_profiles:  "EPSG:4326"
crs_out: null
NA_mask_flag: 0
crop_profiles_to_mask: true
covariate_pattern: ".tif$"

# Output options
multiply: 100
datatype_geotiff: "INT2S" # "INT2U" #  For the output maps: values accepted
 ↪  are "INT1U", "INT2U", "INT2S", "INT4U", "INT4S", "FLT4S", "FLT8S". With
 ↪  GDAL >= 3.5 you can also use "INT8U" and "INT8S". And with GDAL >= 3.7
 ↪  you can use also use "INT1S". See gdal to discover the GDAL version you
 ↪  are using. The first three letters indicate whether the datatype is an
 ↪  integer (whole numbers) of a real number ("float", decimal numbers), the
 ↪  fourth character indicates the number of bytes used for each number.
 ↪  Higher values allow for storing larger numbers and/or more precision; but
 ↪  create larger files. The "S" or "U" indicate whether the values are
 ↪  signed (both negative and positive) or unsigned (zero and positive values
 ↪  only).
gdal_options:
  - "COMPRESS=DEFLATE"
  - "PREDICTOR=2"
gdal_output_driver: "COG" # "GTiff" # "COG" is availbale with gdal >= 3.1
overwrite: true

# Tiling and cores to parallelize
tileside: 200
cores: 3

# Advanced options. Only modify if you know what you are doing
```

```yaml
# Other advanced options
covariatesListFile: null # "config/covs_list_corr_0.85_all" # Either a file
 ↪  path or NULL. A file with the names of the covariates to use (file name
 ↪  without extention, one per line)
thick_cond: 0 # Numeric from 0 to 1. 1 selects profiles that have a maximum
 ↪  depth equal or greater than the max modeled depth, 0 retains all the
 ↪  profiles. Only for 2D modelling (e.g. depth_name : "0-20")
use_dggridR: true
dggrid_resolution: 6
#' Seed for random processes
rndseed: 982374923
terra_memfrac: 0.3 # (from 0.1 to 0.9) fraction of memory that terra can use
# terra's default is 0.6
multiband: false
rfe_number: 5
rfe_repeats: 10
rfe_selection: "best" # "tolerance" #
min_rfe_cvrts: 0 # integer. The minimum number of covariates to be used in
 ↪  the RFE process. Default to 0. If the number of covariates is less than
 ↪  this value, the RFE process will not be performed.


# Plotting options
plot_border: false
legend_title: ""
add_north: true
add_sbar: true
save_to_disk: true
plot_range: NULL # [50, 1000]
plot_col: NULL # ["#00A600", "#2DB600", "#63C600", "#A0D600", "#E6E600",
 ↪  "#E8C32E", "#EBB25E", "#EDB48E", "#F0C9C0", "#F2F2F2"]

model_options:
  comprangerquantreg:
    parsnip_engine: "ranger"
    parsnip_importance: "impurity"
    parsnip_quantreg: true
    parsnip_mode: "regression"
    parsnip_mtry_range_min_multiplier: 0.5
    parsnip_mtry_range_max_multiplier: 3
    parsnip_trees_range: [500, 1000]
```

```
    parsnip_minn_range: [2, 10]
    ntree_default: 500
    stats: ["mean", "Q0.5", "Q0.05", "Q0.95"]
    #stats :
 ↪  [134,3525,24,46346,246,746783,36356,3357,357,2446,2346,36,13,264,724]
    pred_num.trees: 100
    stats_to_log_backtansform: ["mean"]
    stats_to_exp: ["Q0.5", "Q0.05", "Q0.95"]
    search_type: "regular" # "random" #
    size_random_search: 50
    levels_regular_search: 7

# Automatically generated file paths
profilesOverlayFile: "{outputDir}/points/profiles_overlay.csv"
profilesFoldsFile: "{outputDir}/points/profiles_folds.csv"
layersDepthProcessedFile:
 ↪  "{outputDir}/points/layers_depth_processed_{depth_name}.csv"
regressionMatrixFile:
 ↪  "{outputDir}/model/{voi}/regression_matrix_{depth_name}.csv"
rfeCovariatesFile:
 ↪  "{outputDir}/model/{voi}/rfe_{depth_name}_{transformation}.csv"
rfeMetricsProfileFile:
 ↪  "{outputDir}/model/{voi}/rfe_profile_{depth_name}_{transformation}.RDS"
tuningGridFile:
 ↪  "{outputDir}/model/{voi}/tunegrid_{model_name}_{depth_name}_{transformation}_{use_rfe}.RD
bestParamFile:
 ↪  "{outputDir}/model/{voi}/bestparam_{model_name}_{depth_name}_{transformation}_{use_rfe}.c
cvPredictObserveFile:
 ↪  "{outputDir}/model/{voi}/cv-predict-observe_{model_name}_{depth_name}_{transformation}_{u
cvMetricsFile:
 ↪  "{outputDir}/model/{voi}/cv-metrics_{model_name}_{depth_name}_{transformation}_{use_rfe}_
internalMetricsFile:
 ↪  "{outputDir}/model/{voi}/internal-metrics_{model_name}_{depth_name}_{transformation}_{use
modelFittedFile:
 ↪  "{outputDir}/model/{voi}/model-fitted_{model_name}_{depth_name}_{transformation}_{use_rfe
tilesFile: "{outputDir}/other/tiles.csv"
tilesFileGpkg: "{outputDir}/other/tiles.gpkg"
tilesDir:
 ↪  "{outputDir}/maps/{voi}/{model_name}_{depth_name}_{transformation}_{use_rfe}_{do_tune}"
predictedMapVrt:
 ↪  "{outputDir}/maps/{voi}/{model_name}_{depth_name}_{transformation}_{use_rfe}_{do_tune}/{v
plotDir:
 ↪  "{outputDir}/plots/{voi}/{model_name}_{depth_name}_{transformation}_{use_rfe}_{do_tune}"
```

2.2 Config file options

Here follows the list of available configuration options and their description.

Option	Default	Description
`profilesStandardFile`		Location of the profiles table.
`outputDir`		The output directory, where the results will be saved.
`layersStandardFile`		Location of the layers table.
`covarsDir`		Location of the covariates directory. If you want to use several folders you can use `["./input_example/covariates1", "./input_example/covariates2"]`.
`maskFile`		Location of the mask file.
`voi`		The variable of interest. A list in the form [“variable_name_1”, “variable_name_2”, variable_name_3”]
`alr_prefix`	“alr”	Prefix to be given to additive log transformation variables
`alr_scale_factor`	100	Use 100 if your data ranges from 0 to 100, use 1 if your data ranges from 0 to 1
VOI parameters		
`voi_min_value`	0	Minimum value allowed. Values below this one will be discarded.
`voi_max_value`	Inf	maximum value allowed. Values above this one will be discarded.
`transformation`	“notransform”	Apply transformation of values to the voi. Only “notransform” available. Check Section 4.6
Dataset variables		
`pid_name`	“pid”	Profile ID column name.
`lyrid_name`	“lyrid”	Layer ID column name.
`easting_name`	“x”	In the profiles table, the name of the “easting” coordinate.
`northing_name`	“y”	In the profiles table, the name of the “northing” coordinate.

Option	Default	Description
`top_layer_name`	"top"	In the layers table, the name of the top layer depth column.
`bottom_layer_name`	"bottom"	In the layers table the name of the bottom layer depth column.
`flag_validation_name`	null	In the profiles table, the name of the column that identifies which profiles (1) are used for validation and which ones are not (0).
`flag_tuning_name`	null	In the profiles table, the name of the column that identifies which profiles (1) are used for hyperparameter tuning and which ones are not (0).
`flag_rfe_name`	null	In the profiles table, the name of the column that identifies which profiles (1) are used for the recursive feature elimination and which ones are not (0).
`flag_external_validation_name`	null	In the profiles table, the name of the column that identifies which profiles (1) are used for validation and which ones are not (0).
`depth_column_name`	"depth"	Name of the column name used to (internally) identify depth. This depth column will be created by the model, so it must have a different name than those already existing in the layers table.

Option	Default	Description
`fold_name`	"fold"	Name of the column name used to (internally) identify folds for cross validation. This fold column will be created by the model, so it must have a different name than those already existing in the input tables.
Model parameters		
`depth_name`	"0-20"	A string specifying the depth range to be extracted, e.g., "10-20" or "50-90", or "3D" for 3D models.
`depths_3D`	"0,5,15,30, 60,100,200"	Vector of depths for 3D models. This will produce the following depths: 0-5, 5-15,, 15-30, 30-60, 60-100, 100-200. Check Section 4.2 for further details.
`use_rfe`	"dorfe"	Either "dorfe" to use recursive feature elimination or "norfe" to not use it in model tuning. See Section 4.4.
`do_tune`	"tune"	Either "tune" or "notune" to tune model or not. See Section 4.5.
`model_name`	"rangerquantreg"	Name of model used. See Section 4.3.
`n_folds`	10	Number of folds for the cross validation. See Section 4.6
`tune_metric`	"mec"	Metric used to base the model tuning on. Default to "mec" model efficiency coeff (Janssen and Heuberger (1995)). Other options are "rmse" and "rsq"
Covariates information		
`crs_profiles`	"EPSG:4326"	Coordinate reference system of the profiles

Option	Default	Description
`crs_out`	null	The desired coordinate reference system of the output maps.
`NA_mask_flag`	null	The value of NA (*no data*) in the mask. If NULL, the *no data* value from the raster will be used. See Section 5.2.
`crop_profiles_to_mask`	true	Logical. Should the profiles be cropped using the mask. It uses `NA_mask_flag` to determine the areas to exclude the profiles. Default to TRUE.
`covariate_pattern`	".tif$"	The file format of the covariates.
Output options		
`multiply`	10	Multiplication factor, used to multiply the final predicted maps. This often allows to change the datatype to integer and save space.
`datatype_geotiff`	"INT2U"	Output datatype. Check `?terra::writeRaster` to see the accepted datatypes.
`gdal_options`	["COMPRESS=DEFLATE", "PREDICTOR=2"]	GDAL output options. DEFLATE is used for compression. PREDICTOR: modify or remove if FLT4S is used. See gdal documentation.
`gdal_output_driver`	"COG"	GDAL output driver options determines the file type of output raster
`overwrite`	true	If TRUE, overwrite outputs. If FALSE, function exits if all outputs are present. Default to TRUE
Tiling and cores to parallelise		

Option	Default	Description
`tileside`	200	Number of pixels n to divide the predictions into tiles of size n x n. Reduce this number to decrease the amount of RAM used. With a tileside of 200, the prediction step uses approx. 2.2 GB of RAM per core.
`cores`	3	Number of cores to parallelise predictions. Reduce this number to decrease the amount of RAM used.
Advanced options		
`covariatesListFile`	null	Either a file path or NULL. A file with the names of the covariates to use (file name without extension, one per line).
`thick_cond`	0	Numeric. From 0.1 to 1. Minimum proportion of total considered thickness (maxdepth - mindepth) that the sum of the available layers have. See Section 4.1.
`use_dggridR`	true	Logical. Use `dggridR` or not for cross validation folds creation. It is used to randomly select the folds within cells instead of selecting throughout the entire area. See Section 4.8.
`dggrid_resolution`	6	the resolution of the dggridR grid. Find the list here.
`rndseed`	982374923	Seed for random processes, so that these can be reproducible.
`terra_memfrac`	0.6	Fraction of memory that terra can use, from 0.1 to 0.9, terra's default is 0.6.

Option	Default	Description
`multiband`	false	Logical. Generate one multi-band file (TRUE) or several single-band files (FALSE).
`rfe_number`	3	Numeric. Number of resampling iterations for recursive feature elimination. Default to 3. Check `?caret::rfeControl`.
`rfe_repeats`	10	The number of complete sets of folds to compute. Check `?caret::rfeControl`.
`rfe_selection`	"best"	Criterion for selecting the number of covariables in the rfe subset.
`min_rfe_cvrts`	0	integer. The minimum number of covariates to be used in the RFE process. Default to 0. If the number of covariates is less than this value, the RFE process will not be performed
Plotting options		
`plot_border`	false	Logical - plot border around predictions
`legend_title`	" "	String for legend title.
`add_north`	true	Logical - add north bar to map.
`add_sbar`	true	Logical - add sclae bar to map.
`save_to_disk`	true	Logical - save to disk
`plot_range`	null	Range [min, max] of the voi in the plot. Note that voi values are multiplied by the `multiply` factor. Automatic when NULL.
`plot_col`	null	Custom colors for the plot. Automatic when NULL.

Option	Default	Description
Model options		"rangerquantreg" or "xgboostreg", indicated in `model_name`. Only need to modify options of the selected model.
`parsnip_engine`	"ranger"	Parsnip engine. Either "ranger" or "xgboost".
`parsnip_importance`	"impurity"	Index to use for variable importance in rangerquantreg.
`parsnip_quantreg`	true	fit model using "quantreg" option for rangerquantreg.
`parsnip_mode`	"regression"	Parsnip mode for rangerquantreg. Only "regression" is allowed.
`parsnip_mtry_range_min_multiplier`	0.5	Since the default value for mtry is the sqrt(Number of covariates), to find the minimum value for a tuning range we use '`parsnip_mtry_range_min()`'. This function takes the number of covariates and the multiplier and applies `round(sqrt(n_of_covariates) * multiplier)`. See Section 4.5.
`parsnip_mtry_range_max_multiplier`	3	Since the default value for mtry is the sqrt(Number of covariates) to find the maximum value for a tuning range we use '`parsnip_mtry_range_max()`'. This function takes the number of covariates and the multiplier and applies `round(sqrt(n_of_covariates) * multiplier)`. See Section 4.5.

Option	Default	Description
`parsnip_trees_range`	[500, 1000]	Range of number of trees for random forest model tuning. See Section 4.5.
`ntree_default`	500	Default number of trees for random forest. This is used if model tuning is not applied. See Section 4.5.
`stats`	[“mean”, “Q0.5”, “Q0.05”, “Q0.95”]	Predictions will be calculated for these quantiles. Values allowed: mean and quantiles in the form Q[quatile].
`stats_to_log_backtansform`	[“mean”]	When log-transforming the voi: which stats to log back-transform as `exp(value + 0.5 * variance_values)`. values allowed: mean and quantiles in the form Q[quantile]
`stats_to_exp`	[“Q0.5”, “Q0.05”, “Q0.95”]	When exp-transforming the voi: which stats to exp back-transform as `exp(value)`. Values allowed: mean and quantiles in the form Q[quantile]
`search_type`	“regular”	Which search type for model tuning. Either “regular” or “random”.
`size_random_search`	50	For random search in model tuning: how many combinations of the tuning parameters to use. When 50, for example, it finds 50 random number of trees.

Option	Default	Description
`levels_regular_search`	7	For random search in model tuning: how many levels to use, that is, how many unique values per tuning parameter to use. When it is 7, for example, it finds 5 more trees between the min and max number of trees in the range.

In addition to the previous options the configuration file also automatically generates input and output files needed by the workflow

Option	Default
profilesOverlayFile	"{outputDir}/points/profiles_overlay.csv"
profilesFoldsFile	"{outputDir}/points/profiles_folds.csv"
layersDepthProcessedFile	"{outputDir}/points/layers_depth_processed_{depth_name}
regressionMatrixFile	"{outputDir}/model/{voi}/regression_matrix_ {depth_name}.csv"
rfeCovariatesFile	"{outputDir}/model/{voi}/rfe_{depth_name}_ {transformation}.csv"
rfeMetricsProfileFile	"{outputDir}/model/{voi}/rfe_profile_{depth_name}_ {transformation}.RDS"
tuningGridFile	"{outputDir}/model/{voi}/tunegrid_{model_name}_ {depth_name}_{transformation}_{use_rfe}.RDS"
bestParamFile	"{outputDir}/model/{voi}/bestparam_{model_name}_ {depth_name}_{transformation}_{use_rfe}.csv"
cvPredictObserveFile	"{outputDir}/model/{voi}/cv-predict-observe_{model_name}_ {depth_name}_{transformation}_{use_rfe}_{do_tune}.csv"
cvMetricsFile	"{outputDir}/model/{voi}/cv-metrics_{model_name}_ {depth_name}_{transformation}_{use_rfe}_{do_tune}.csv"
internalMetricsFile	"{outputDir}/model/{voi}/internal-metrics_{model_name}_ {depth_name}_{transformation}_{use_rfe}_{do_tune}.csv"
modelFittedFile	"{outputDir}/model/{voi}/model-fitted_{model_name}_ {depth_name}_{transformation}_{use_rfe}_{do_tune}.RDS
tilesFile	"{outputDir}/other/tiles.csv"
tilesFileGpkg	"{outputDir}/other/tiles.gpkg"
tilesDir	"{outputDir}/maps/{voi}/{model_name}_{depth_name}_ {transformation}_{use_rfe}_{do_tune}"
predictedMapVrt	"{outputDir}/maps/{voi}/{model_name}_{depth_name}_ {transformation}_{use_rfe}_{do_tune}/{voi}.vrt"
plotDir	"{outputDir}/plots/{voi}/{model_name}_{depth_name}_ {transformation}_{use_rfe}_{do_tune}"

3 Workflow script

The script required to run the workflow depends on whether it is numerical or compositional modelling. Both can be obtained in the same way.

> When running the run_workflow.R script make sure that your **Working Directory** is set to the directory where the config file lives.
> Alternatively change the `config_location` to the absolute path as follows: config_location <- "path/to/file/config.yaml"

There are **two ways** to obtain the workflow script file:

3.0.1 Option A

Go to one of the following following links and download one (or more) of the files:

- `run_workflow.R` https://git.wur.nl/isric/dsm-general/dsm.workflows/seedling/-/blob/main/run_workflow.R
- `run_workflow_compositional.R` https://git.wur.nl/isric/dsm-general/dsm.workflows/seedling/-/blob/main/run_workflow_compositional.R

3.0.2 Option B

Copy the following R code and save it as `run_workflow.R`

Numerical

```
# Version 0.4.2

library(isric.dsm.base)
library(argparser)

config_location <- "template.config.yaml"
```

```r
p <- arg_parser("Run the entire workflow")
p <- add_argument(
     parser = p, arg = "--config_file",
     help = "configuration file location", default = config_location
)
config <- yaml::read_yaml(parse_args(p)$config_file, eval.expr = TRUE)

# ## Prepare

profiles_overlay(
     profilesStandardFile = config$profilesStandardFile,
     covarsDir = config$covarsDir,
     profilesOverlayFile = config$profilesOverlayFile,
     covariatesListFile = config$covariatesListFile,
     outputDir = config$outputDir,
     easting_name = config$easting_name,
     northing_name = config$northing_name,
     crs_profiles = config$crs_profiles,
     pid_name = config$pid_name,
     covariate_pattern = config$covariate_pattern,
     maskFile = config$maskFile,
     NA_mask_flag = config$NA_mask_flag,
     crop_profiles_to_mask = config$crop_profiles_to_mask,
     overwrite = config$overwrite
)

layers_depth(
     layersStandardFile = config$layersStandardFile,
     layersDepthProcessedFile = config$layersDepthProcessedFile,
     outputDir = config$outputDir,
     pid_name = config$pid_name,
     lyrid_name = config$lyrid_name,
     top_layer_name = config$top_layer_name,
     bottom_layer_name = config$bottom_layer_name,
     depth_name = config$depth_name,
     depths_3D = config$depths_3D,
     depth_column_name = config$depth_column_name,
     thick_cond = config$thick_cond, overwrite = config$overwrite
)

profiles_folds(
     profilesStandardFile = config$profilesStandardFile,
     profilesFoldsFile = config$profilesFoldsFile,
```

```
     outputDir = config$outputDir,
     pid_name = config$pid_name,
     easting_name = config$easting_name, northing_name =
      ↪  config$northing_name,
     fold_name = config$fold_name,
     n_folds = config$n_folds,
     rndseed = config$rndseed, use_dggridR = config$use_dggridR,
     dggrid_resolution = config$dggrid_resolution,
     flag_validation_name = config$flag_validation_name,
     flag_rfe_name = config$flag_rfe_name,
     flag_tuning_name = config$flag_tuning_name, overwrite = config$overwrite
)

regression_matrix(
     voi = config$voi,
     profilesOverlayFile = config$profilesOverlayFile,
     layersDepthProcessedFile = config$layersDepthProcessedFile,
     profilesFoldsFile = config$profilesFoldsFile,
     covariatesListFile = config$covariatesListFile,
     profilesStandardFile = config$profilesStandardFile,
     regressionMatrixFile = config$regressionMatrixFile,
     outputDir = config$outputDir,
     pid_name = config$pid_name,
     lyrid_name = config$lyrid_name,
     fold_name = config$fold_name,
     easting_name = config$easting_name,
     northing_name = config$northing_name,
     depth_name = config$depth_name,
     depth_column_name = config$depth_column_name,
     voi_min_value = config$voi_parameters$voi_min_value,
     voi_max_value = config$voi_parameters$voi_max_value,
     flag_validation_name = config$flag_validation_name,
     flag_tuning_name = config$flag_tuning_name,
     flag_rfe_name = config$flag_rfe_name,
     flag_external_validation_name = config$flag_external_validation_name,
     overwrite = config$overwrite
)


# ## Model
if (config$use_rfe == "dorfe") {
     run_rfe(
          voi = config$voi,
```

```
          regressionMatrixFile = config$regressionMatrixFile,
          rfeMetricsProfileFile = config$rfeMetricsProfileFile,
          rfeCovariatesFile = config$rfeCovariatesFile,
          outputDir = config$outputDir,
          transformation = config$voi_parameters$transformation,
          rfe_number = config$rfe_number,
          rfe_repeats = config$rfe_repeats,
          cores = config$cores,
          rndseed = config$rndseed,
          pid_name = config$pid_name,
          lyrid_name = config$lyrid_name,
          fold_name = config$fold_name,
          depth_name = config$depth_name,
          depth_column_name = config$depth_column_name,
          flag_validation_name = config$flag_validation_name,
          flag_tuning_name = config$flag_tuning_name,
          flag_rfe_name = config$flag_rfe_name, overwrite = config$overwrite,
          rfe_selection = config$rfe_selection,
          min_rfe_cvrts = config$min_rfe_cvrts
     )
}

if (config$do_tune == "tune") {
     model_tune(
          voi = config$voi,
          regressionMatrixFile = config$regressionMatrixFile,
          tuningGridFile = config$tuningGridFile,
          bestParamFile = config$bestParamFile,
          rfeCovariatesFile = config$rfeCovariatesFile,
          outputDir = config$outputDir, model_name = config$model_name,
          pid_name = config$pid_name, lyrid_name = config$lyrid_name,
          depth_name = config$depth_name,
          transformation = config$voi_parameters$transformation,
          use_rfe = config$use_rfe,
          fold_name = config$fold_name, tune_metric = config$tune_metric,
          depth_column_name = config$depth_column_name,
          rndseed = config$rndseed,
          cores = config$cores,
          flag_tuning_name = config$flag_tuning_name,
          flag_validation_name = config$flag_validation_name,
          flag_rfe_name = config$flag_rfe_name,
          model_options = config$model_options[[config$model_name]],
          overwrite = config$overwrite
```

```
    )
}

model_cv(
    voi = config$voi,
    regressionMatrixFile = config$regressionMatrixFile,
    rfeCovariatesFile = config$rfeCovariatesFile,
    bestParamFile = config$bestParamFile,
    cvMetricsFile = config$cvMetricsFile,
    cvPredictObserveFile = config$cvPredictObserveFile,
    outputDir = config$outputDir,
    model_name = config$model_name,
    do_tune = config$do_tune,
    transformation = config$voi_parameters$transformation,
    use_rfe = config$use_rfe, pid_name = config$pid_name,
    lyrid_name = config$lyrid_name, fold_name = config$fold_name,
    depth_name = config$depth_name,
    depth_column_name = config$depth_column_name,
    cores = config$cores,
    flag_validation_name = config$flag_validation_name,
    flag_tuning_name = config$flag_tuning_name,
    flag_rfe_name = config$flag_rfe_name,
    model_options = config$model_options[[config$model_name]],
    rndseed = config$rndseed,
    overwrite = config$overwrite
)

model_fit(
    voi = config$voi, regressionMatrixFile = config$regressionMatrixFile,
    bestParamFile = config$bestParamFile,
    rfeCovariatesFile = config$rfeCovariatesFile,
    modelFittedFile = config$modelFittedFile,
    internalMetricsFile = config$internalMetricsFile,
    outputDir = config$outputDir,
    use_rfe = config$use_rfe, do_tune = config$do_tune,
    transformation = config$voi_parameters$transformation,
    rndseed = config$rndseed,
    depth_name = config$depth_name, pid_name = config$pid_name,
    fold_name = config$fold_name, lyrid_name = config$lyrid_name,
    depth_column_name = config$depth_column_name,
    flag_validation_name = config$flag_validation_name,
    flag_tuning_name = config$flag_tuning_name,
    flag_rfe_name = config$flag_rfe_name,
```

```
     model_name = config$model_name,
     model_options = config$model_options[[config$model_name]],
     overwrite = config$overwrite
)


model_external_validation(
    voi = config$voi,
    profilesStandardFile = config$profilesStandardFile,
    profilesOverlayFile = config$profilesOverlayFile,
    layersDepthProcessedFile = config$layersDepthProcessedFile,
    profilesFoldsFile = config$profilesFoldsFile,
    modelFittedFile = config$modelFittedFile,
    externalMetricsFile = config$externalMetricsFile,
    externalPredictObserveFile = config$externalPredictObserveFile,
    outputDir = config$outputDir,
    rndseed = config$rndseed,
    pid_name = config$pid_name,
    lyrid_name = config$lyrid_name,
    fold_name = config$fold_name,
    depth_name = config$depth_name,
    depth_column_name = config$depth_column_name,
    use_rfe = config$use_rfe,
    do_tune = config$do_tune,
    transformation = config$voi_parameters$transformation,
    model_name = config$model_name,
    flag_external_validation_name = config$flag_external_validation_name,
    overwrite = config$overwrite
)


# ## Predict
make_tiles(
     maskFile = config$maskFile, tileside = config$tileside, tilesFile =
      ↪  config$tilesFile,
     tilesFileGpkg = config$tilesFileGpkg, outputDir = config$outputDir,
     NA_mask_flag = config$NA_mask_flag, overwrite = config$overwrite
)

predict_tiles(
     voi = config$voi, tilesFile = config$tilesFile,
     covarsDir = config$covarsDir,
     rfeCovariatesFile = config$rfeCovariatesFile, maskFile =
      ↪  config$maskFile,
```

```
     modelFittedFile = config$modelFittedFile, tilesDir = config$tilesDir,
     outputDir = config$outputDir, depth_name = config$depth_name,
     transformation = config$voi_parameters$transformation,
     use_rfe = config$use_rfe,
     do_tune = config$do_tune, multiply = config$multiply,
     gdal_options = config$gdal_options,
     datatype_geotiff = config$datatype_geotiff, depths_3D =
      ↪  config$depths_3D,
     covariate_pattern = config$covariate_pattern,
     NA_mask_flag = config$NA_mask_flag, cores = config$cores,
     rndseed = config$rndseed,
     model_name = config$model_name,
     model_options = config$model_options[[config$model_name]],
     overwrite = config$overwrite
)

mosaic_tiles(
     tilesDir = config$tilesDir, predictedMapVrt = config$predictedMapVrt,
     outputDir = config$outputDir, voi = config$voi,
     model_name = config$model_name, depth_name = config$depth_name,
     transformation = config$voi_parameters$transformation,
     use_rfe = config$use_rfe,
     do_tune = config$do_tune, gdal_output_driver =
      ↪  config$gdal_output_driver,
     gdal_options = config$gdal_options,
     datatype_geotiff = config$datatype_geotiff, multiband =
      ↪  config$multiband,
     crs_out = config$crs_out, extra_metadata =
          c(
               config[c(
                    "voi", "depth_name", "use_rfe", "do_tune", "model_name"
               )],
               config$voi_parameters,
               model_options = config$model_options[[config$model_name]],
               variable_importance =
                ↪  list(sort(ranger::importance(readRDS(glue::glue(config$modelFittedFile,
                    outputDir = config$outputDir, voi = config$voi,
                    model_name = config$model_name, depth_name =
                     ↪  config$depth_name,
                    transformation = config$voi_parameters$transformation,
                    use_rfe = config$use_rfe, do_tune = config$do_tune
               ))),decreasing = TRUE)),
```

```
                accuracy =
                 ↪  tibble::deframe(read.csv(glue::glue(config$cvMetricsFile,
                     outputDir = config$outputDir, voi = config$voi,
                     model_name = config$model_name, depth_name =
                      ↪  config$depth_name,
                     transformation = config$voi_parameters$transformation,
                     use_rfe = config$use_rfe, do_tune = config$do_tune
                ))[, c("metric", "value")]),
                nrow_regression_matrix =
                 ↪  nrow(read.csv(glue::glue(config$regressionMatrixFile,
                     outputDir = config$outputDir,
                     voi = config$voi, depth_name = config$depth_name
                )))
            ),
        maskFile = config$maskFile
)

plot_maps_terra(
        tilesDir = config$tilesDir, plotDir = config$plotDir,
        outputDir = config$outputDir, voi = config$voi,
        model_name = config$model_name, depth_name = config$depth_name,
        transformation = config$voi_parameters$transformation,
        use_rfe = config$use_rfe,
        do_tune = config$do_tune, plot_border = config$plot_border,
        divideBy = config$multiply, legend_title = config$legend_title,
        add_north = config$add_north, add_sbar = config$add_sbar,
        save_to_disk = config$save_to_disk,
        range = config$plot_range, col = config$plot_col
)
```

Compositional

```
# Version 0.4.2

library(isric.dsm.base)
library(argparser)

config_location <- "template.config_compositional.yaml"

p <- arg_parser("Run the entire workflow")
```

```
p <- add_argument(
     parser = p, arg = "--config_file",
     help = "configuration file location", default = config_location
)
config <- yaml::read_yaml(parse_args(p)$config_file, eval.expr = TRUE)

# ## Prepare

profiles_overlay(
     profilesStandardFile = config$profilesStandardFile,
     covarsDir = config$covarsDir,
     profilesOverlayFile = config$profilesOverlayFile,
     covariatesListFile = config$covariatesListFile,
     outputDir = config$outputDir,
     easting_name = config$easting_name,
     northing_name = config$northing_name,
     crs_profiles = config$crs_profiles,
     pid_name = config$pid_name,
     covariate_pattern = config$covariate_pattern,
     maskFile = config$maskFile,
     NA_mask_flag = config$NA_mask_flag,
     crop_profiles_to_mask = config$crop_profiles_to_mask,
     overwrite = config$overwrite
)

layers_depth(
     layersStandardFile = config$layersStandardFile,
     layersDepthProcessedFile = config$layersDepthProcessedFile,
     outputDir = config$outputDir,
     pid_name = config$pid_name,
     lyrid_name = config$lyrid_name,
     top_layer_name = config$top_layer_name,
     bottom_layer_name = config$bottom_layer_name,
     depth_name = config$depth_name,
     depths_3D = config$depths_3D,
     thick_cond = config$thick_cond, overwrite = config$overwrite
)

profiles_folds(
     profilesStandardFile = config$profilesStandardFile,
     profilesFoldsFile = config$profilesFoldsFile,
     outputDir = config$outputDir,
     pid_name = config$pid_name,
```

```
    easting_name = config$easting_name, northing_name =
     ↪  config$northing_name,
    fold_name = config$fold_name,
    n_folds = config$n_folds,
    rndseed = config$rndseed, use_dggridR = config$use_dggridR,
    dggrid_resolution = config$dggrid_resolution,
    flag_validation_name = config$flag_validation_name,
    flag_rfe_name = config$flag_rfe_name,
    flag_tuning_name = config$flag_tuning_name, overwrite = config$overwrite
)

regression_matrix(
    voi = config$voi,
    profilesOverlayFile = config$profilesOverlayFile,
    layersDepthProcessedFile = config$layersDepthProcessedFile,
    profilesFoldsFile = config$profilesFoldsFile,
    covariatesListFile = config$covariatesListFile,
    regressionMatrixFile = config$regressionMatrixFile,
    outputDir = config$outputDir,
    pid_name = config$pid_name,
    lyrid_name = config$lyrid_name,
    fold_name = config$fold_name,
    easting_name = config$easting_name,
    northing_name = config$northing_name,
    depth_name = config$depth_name,
    depth_column_name = config$depth_column_name,
    voi_min_value = config$voi_parameters$voi_min_value,
    voi_max_value = config$voi_parameters$voi_max_value,
    flag_validation_name = config$flag_validation_name,
    flag_tuning_name = config$flag_tuning_name,
    flag_rfe_name = config$flag_rfe_name, overwrite = config$overwrite
)

regression_matrix_alr(
    voi = config$voi, alr_prefix = config$alr_prefix,
    pid_name = config$pid_name, lyrid_name = config$lyrid_name,
    regressionMatrixFile = config$regressionMatrixFile,
    profilesOverlayFile = config$profilesOverlayFile,
    outputDir = config$outputDir,
    depth_name = config$depth_name
)

# ## Model
```

```
if (config$use_rfe == "dorfe") {
     . <- lapply(paste0(config$alr_prefix, seq(length(config$voi) - 1)),
 ↪  function(voi) {
          run_rfe(
               voi = voi,
               regressionMatrixFile = config$regressionMatrixFile,
               rfeMetricsProfileFile = config$rfeMetricsProfileFile,
               rfeCovariatesFile = config$rfeCovariatesFile,
               outputDir = config$outputDir,
               transformation = config$voi_parameters$transformation,
               rfe_number = config$rfe_number,
               rfe_repeats = config$rfe_repeats,
               cores = config$cores,
               rndseed = config$rndseed,
               pid_name = config$pid_name,
               lyrid_name = config$lyrid_name,
               fold_name = config$fold_name,
               depth_name = config$depth_name,
               depth_column_name = config$depth_column_name,
               flag_validation_name = config$flag_validation_name,
               flag_tuning_name = config$flag_tuning_name,
               flag_rfe_name = config$flag_rfe_name,
               overwrite = config$overwrite,
               rfe_selection = config$rfe_selection,
               min_rfe_cvrts = config$min_rfe_cvrts
          )
     })
}

if (config$do_tune == "tune") {
     . <- lapply(paste0(config$alr_prefix, seq(length(config$voi) - 1)),
 ↪  function(voi) {
          model_tune(
               voi = voi,
               regressionMatrixFile = config$regressionMatrixFile,
               tuningGridFile = config$tuningGridFile,
               bestParamFile = config$bestParamFile,
               rfeCovariatesFile = config$rfeCovariatesFile,
               outputDir = config$outputDir, model_name = config$model_name,
               pid_name = config$pid_name, lyrid_name = config$lyrid_name,
               depth_name = config$depth_name,
               transformation = config$voi_parameters$transformation,
               use_rfe = config$use_rfe,
```

```
                fold_name = config$fold_name, tune_metric =
                 ↪  config$tune_metric,
                depth_column_name = config$depth_column_name,
                rndseed = config$rndseed,
                cores = config$cores,
                flag_tuning_name = config$flag_tuning_name,
                flag_validation_name = config$flag_validation_name,
                flag_rfe_name = config$flag_rfe_name,
                model_options = config$model_options[[config$model_name]],
                overwrite = config$overwrite
            )
        })
}

. <- lapply(paste0(config$alr_prefix, seq(length(config$voi) - 1)),
 ↪  function(voi) {
        model_cv(
            voi = voi,
            regressionMatrixFile = config$regressionMatrixFile,
            rfeCovariatesFile = config$rfeCovariatesFile,
            bestParamFile = config$bestParamFile,
            cvMetricsFile = config$cvMetricsFile,
            cvPredictObserveFile = config$cvPredictObserveFile,
            outputDir = config$outputDir,
            model_name = config$model_name,
            do_tune = config$do_tune,
            transformation = config$voi_parameters$transformation,
            use_rfe = config$use_rfe, pid_name = config$pid_name,
            lyrid_name = config$lyrid_name, fold_name = config$fold_name,
            depth_name = config$depth_name,
            depth_column_name = config$depth_column_name,
            cores = config$cores,
            flag_validation_name = config$flag_validation_name,
            flag_tuning_name = config$flag_tuning_name,
            flag_rfe_name = config$flag_rfe_name,
            model_options = config$model_options[[config$model_name]],
            rndseed = config$rndseed, overwrite = config$overwrite
        )
})

model_cv_compositional(
        alr_names = paste0(config$alr_prefix, seq(length(config$voi) - 1)),
        alrinv_scale_factor = config$alrinv_scale_factor,
```

```
     voi = config$voi,
     pid_name = config$pid_name, fold_name = config$fold_name,
     lyrid_name = config$lyrid_name,
     outputDir = config$outputDir,
     model_name = config$model_name,
     depth_name = config$depth_name,
     use_rfe = config$use_rfe,
     do_tune = config$do_tune,
     cvPredictObserveFile = config$cvPredictObserveFile,
     layersDepthProcessedFile = config$layersDepthProcessedFile,
     cvMetricsFile = config$cvMetricsFile, overwrite = config$overwrite
)

. <- lapply(paste0(config$alr_prefix, seq(length(config$voi) - 1)),
 ↪  function(voi) {
     model_fit(
          voi = voi, regressionMatrixFile = config$regressionMatrixFile,
          bestParamFile = config$bestParamFile,
          rfeCovariatesFile = config$rfeCovariatesFile,
          modelFittedFile = config$modelFittedFile,
          internalMetricsFile = config$internalMetricsFile,
          outputDir = config$outputDir,
          use_rfe = config$use_rfe, do_tune = config$do_tune,
          transformation = config$voi_parameters$transformation,
          rndseed = config$rndseed,
          depth_name = config$depth_name, pid_name = config$pid_name,
          fold_name = config$fold_name, lyrid_name = config$lyrid_name,
          depth_column_name = config$depth_column_name,
          flag_validation_name = config$flag_validation_name,
          flag_tuning_name = config$flag_tuning_name,
          flag_rfe_name = config$flag_rfe_name,
          model_name = config$model_name,
          model_options = config$model_options[[config$model_name]],
          overwrite = config$overwrite
     )
})

# ## Predict
make_tiles(
     maskFile = config$maskFile, tileside = config$tileside,
     tilesFile = config$tilesFile, tilesFileGpkg = config$tilesFileGpkg,
     outputDir = config$outputDir, NA_mask_flag = config$NA_mask_flag,
     overwrite = config$overwrite
```

```
)

. <- lapply(paste0(config$alr_prefix, seq(length(config$voi) - 1)),
 ↪  function(voi) {
     predict_tiles(
          voi = voi, tilesFile = config$tilesFile,
          covarsDir = config$covarsDir,
          rfeCovariatesFile = config$rfeCovariatesFile, maskFile =
           ↪  config$maskFile,
          modelFittedFile = config$modelFittedFile, tilesDir =
           ↪  config$tilesDir,
          outputDir = config$outputDir, depth_name = config$depth_name,
          transformation = config$voi_parameters$transformation,
          use_rfe = config$use_rfe,
          do_tune = config$do_tune, multiply = 1,
          gdal_options = c("COMPRESS=DEFLATE"),
          datatype_geotiff = "FLT8S",
          depths_3D = config$depths_3D,
          covariate_pattern = config$covariate_pattern,
          NA_mask_flag = config$NA_mask_flag, cores = config$cores,
          rndseed = config$rndseed,
          model_name = config$model_name,
          model_options = config$model_options[[config$model_name]],
          overwrite = config$overwrite
     )
})

predict_tiles_compositional(
     voi = config$voi,
     tilesFile = config$tilesFile,
     alr_names = paste0(config$alr_prefix, seq(length(config$voi) - 1)),
     outputDir = config$outputDir, depth_name = config$depth_name,
     tilesDir = config$tilesDir,
     transformation = config$voi_parameters$transformation,
     use_rfe = config$use_rfe,
     do_tune = config$do_tune,
     alrinv_scale_factor = config$alrinv_scale_factor,
     multiply = config$multiply,
     gdal_options = config$gdal_options,
     datatype_geotiff = config$datatype_geotiff, depths_3D =
      ↪  config$depths_3D,
     covariate_pattern = config$covariate_pattern,
```

```
    NA_mask_flag = config$NA_mask_flag, cores = round(config$cores * (2 /
     ↪  3)),
    model_name = config$model_name,
    model_options = config$model_options[[config$model_name]],
    overwrite = config$overwrite
)

. <- lapply(config$voi, function(voi) {
    mosaic_tiles(
        tilesDir = config$tilesDir, predictedMapVrt =
         ↪  config$predictedMapVrt,
        outputDir = config$outputDir, voi = voi,
        model_name = config$model_name, depth_name = config$depth_name,
        transformation = config$voi_parameters$transformation,
        use_rfe = config$use_rfe, do_tune = config$do_tune,
        gdal_output_driver = config$gdal_output_driver,
        gdal_options = config$gdal_options,
        datatype_geotiff = config$datatype_geotiff,
        multiband = config$multiband,
        crs_out = config$crs_out, extra_metadata =
            c(
                voi = voi,
                config[c(
                    "depth_name", "use_rfe", "do_tune", "model_name"
                )],
                config$voi_parameters,
                model_options =
                 ↪  config$model_options[[config$model_name]],
                variable_importance = lapply(
                    paste0(
                        config$alr_prefix,
                        seq(length(config$voi) - 1)
                    ),
                    function(voi) {

                        ↪  return(sort(ranger::importance(readRDS(glue::glue(config$mo
                            outputDir = config$outputDir, voi = voi,
                            model_name = config$model_name, depth_name
                             ↪  = config$depth_name,
                            transformation =
                             ↪  config$voi_parameters$transformation,
                            use_rfe = config$use_rfe, do_tune =
                             ↪  config$do_tune
```

```
                              )), decreasing = TRUE)))
                        }
                    ),
                    accuracy =
                     ↪  tibble::deframe(read.csv(glue::glue(config$cvMetricsFile,
                         outputDir = config$outputDir, voi = voi,
                         model_name = config$model_name, depth_name =
                          ↪  config$depth_name,
                         transformation =
                          ↪  config$voi_parameters$transformation,
                         use_rfe = config$use_rfe, do_tune = config$do_tune
                    ))[, c("metric", "value")]),
                    nrow_regression_matrix =
                     ↪  nrow(read.csv(glue::glue(config$regressionMatrixFile,
                         outputDir = config$outputDir,
                         voi = voi, depth_name = config$depth_name
                    )))
                ),
    maskFile = config$maskFile
    )
})


. <- lapply(config$voi, function(voi) {
    plot_maps_terra(
        tilesDir = config$tilesDir, plotDir = config$plotDir,
        outputDir = config$outputDir, voi = voi,
        model_name = config$model_name, depth_name = config$depth_name,
        transformation = config$voi_parameters$transformation,
        use_rfe = config$use_rfe,
        do_tune = config$do_tune, plot_border = config$plot_border,
        divideBy = config$multiply, legend_title = config$legend_title,
        add_north = config$add_north, add_sbar = config$add_sbar,
        save_to_disk = config$save_to_disk,
        range = config$plot_range, col = config$plot_col
    )
})
```

4 Modeling Choices

4.1 Depth in a 2D model

Sampling depths can differ between sampling locations. This is for instance the case when soil horizons are sampled instead of fixed depth layers. In DSM, however, we typically predict for a fixed depth (e.g. 0-20 cm, 20-50 cm) at each prediction location. In a two-dimensional model this raises the question on how to standardise the depths of sampled layers, in case samples are not taken for fixed depth intervals, to the target depth interval of the model.

This workflow automatically treats profiles that have different depth layers by performing the weighted average of the layers for each profile for the desired depth interval. The weight is equal to the proportion that a layer contributes to the target depth interval. For instance, if a profile has one sampled layer extending from 0 to 15 cm and one from 15 to 45 cm and the target depth for modelling is 0-20 cm, then weight for the first layer equals 0.75 (15/20) and that of the second layer 0.25 (5/20). The weighted average is the simply computed as the weight of layer 1 multiplied by the value of the soil variable of interest for layer 1 plus the weight of layer 2 multiplied by the value of the soil variable of interest for layer 2.

For example, if the variable of interest is pH. For the layers described above, the pH value for the 0-15cm layer is 6.4 and the pH value for 15-45cm layer is 6, then the modeled pH for the 0–20 cm depth interval is calculated as:

$$\mathrm{pH}_{0\text{–}20} = (\frac{15}{20} * 6.4) + (\frac{5}{20} * 6)$$

The option **thick_cond** (thickness condition, from 0 to 1) identifies the proportion of depth interval that the layers in a profile need to cover in order for that profile to be used for modelling. Suppose we wish to model a soil variable of interest for the 0-20 cm depth interval. Ideally we use only profiles that have data about the soil variable of interest for a depth up to at least 20 cm. But this might not always be the case. With the **thick_cond** option we can relax this requirement and include profiles that might not have data for the complete target depth interval. The **thick_cond** defines the proportion of the target depth interval that requires data. For instance, if we set the **thick_cond** to 0.50 then profiles that have data for at least 10 cm (50% of the 20 cm interval) will be used for modelling. The larger the **thick_cond** the more stringent the requirement on the coverage of the target depth interval. If **thick_cond** equals 1 than only profiles that have data for the full target depth will be used.

4.2 Depth in a 3D model

The workflow can perform 3D modeling where depth is the third dimension. This is achieved by using the depth, defined as midpoint of the top and bottom depth (see Section 1.2.2), as a covariate. Since the depth is an independent variable of the model it can be specified at prediction time to predict for different soil depths. By default the following standard depths are predicted "0-5,5-15,15-30,30-60,60-100,100-200", these are the Global Soil Map (Arrouays et al. 2014) standard depths.

4.3 Quantile Regression Forests (QRF) implementation

This workflow can potentially be used with several modelling algorithms. To date the following algorithms/models are implemented in the Seedling:

- **rangerquantreg** (numerical) which follows the procedure used in SoilGrids 2.0 (Poggio et al. 2021). Models were obtained with the ranger package (Wright and Ziegler 2017), with the option quantreg to build quantile random forests p@meinshausen2006quantile]. With this option, the prediction is not a single value, e.g. the average of predictions from the group of decision trees in the random forest, but rather a cumulative probability distribution of the soil property at each location and depth.
- **comprangerquantreg** (compositional) in this workflow, compositional variables (typically sand, silt, and clay for soil) are handled using the additive log-ratio (ALR) transformation to account for their constrained nature, where the sum of components equals 100% or to 1. The denominator for the ALR transformation is user-defined: it corresponds to the first variable listed in the compositional variables section of the configuration file. Following the methodology used in SoilGrids 2.0 (Poggio et al. 2021), the ALR-transformed variables are modeled separately using the **rangerquantreg** model. A number of simulations are generated (as specified by the `pred_num.trees` parameter in the config file), and these are back-transformed to the original compositional space. From the resulting distribution, the mean and specified quantiles (by default 0.05, 0.5, and 0.95) are computed to produce the final predictions. This approach preserves the spatial correlation and compositional integrity of the original variables, as demonstrated in prior applications of ALR to soil texture data.

4.4 Recursive Feature Elimination (RFE)

This workflow performs a Recursive Feature Elimination (RFE) implemented in the caret package.

RFE is a feature selection technique used to identify the most relevant features (predictors; here the covariates) in a dataset for building a predictive model. When using the `caret` package in R, RFE can be applied in combination with the Random Forest algorithm to select the optimal subset of features (Guyon et al. 2002).

Here's an overview of how Recursive Feature Elimination works with the `caret` package and Random Forest:

1. **Define the Control Parameters**: Set the control parameters for the RFE process. In this case, you will specify the number of features to select (`size`) and the resampling method (`method`), which is typically cross-validation.
2. **Configure the RFE Algorithm**: Create a control object using the `rfeControl` function from the `caret` package. Set the control parameters defined in the previous step.
3. **Run the RFE Algorithm**: Apply the RFE algorithm by using the `rfe` function from the `caret` package. Specify the Random Forest model as the underlying algorithm and provide the training dataset along with the response variable.
4. **Evaluate Feature Importance**: The RFE algorithm iteratively builds the Random Forest model on different subsets of features, ranking them based on their importance. The importance of each feature is calculated using metrics such as mean decrease in Gini impurity or mean decrease in accuracy.
5. **Select Optimal Subset of Features**: The RFE algorithm selects the optimal subset of features based on their importance rankings. The number of features to select is determined by the `size` parameter defined in step 2.
6. **Model Training and Evaluation**: Build a Random Forest model using the selected subset of features on the training dataset. Evaluate the model's performance on the validation dataset using appropriate evaluation metrics.

By applying RFE with Random Forest in the `caret` package, the workflow can systematically identify and select the most informative features for the predictive modelling task. This helps to improve model performance, reduce overfitting, and potentially enhance interpretability by focusing on the most relevant features.

4.5 Model tuning

The workflow incorporates hyperparameter tuning by exploring a range of hyperparameters using either a random or regular grid approach. It systematically evaluates different combinations of these hyperparameters to identify the optimal configuration that produces the best results. By iteratively fitting models with various hyperparameter values, the workflow seeks to find the set of hyperparameters that maximizes performance.

`mtry` and `ntree` are used in model tuning the models: **rangerquantreg** and **comprangerquantreg**. In Random Forest, `mtry` and `ntree` are two important parameters that influence the behavior and performance of the algorithm:

1. **mtry**: The `mtry` parameter determines the number of randomly selected predictor variables considered at each split of a decision tree within the Random Forest algorithm. In other words, it controls the number of features available for splitting at each node of the tree. By default, `mtry` is set to the square root of the total number of predictors in the dataset. A higher **`mtry`** value can lead to more diverse trees and potentially increased model complexity, while a lower value may result in less randomness and potentially reduced model variance.
2. **ntree**: The `ntree` parameter defines the number of trees to be grown in the Random Forest ensemble. Each tree in the forest is constructed using a random subset of the training data and random feature selection. Increasing the number of trees can improve the stability and robustness of the model's predictions but also increases the computational time and memory requirements. It is generally recommended to set a higher `ntree` value to capture more complex patterns and reduce the impact of noise in the data, but there is a trade-off between model performance and computational cost.

Note that changing **mtry** and **ntree** settings can affect the stability of the random forest predictions. We recommend to use the default settings and only tune these two parameters if you are an experienced user who understands the possible consequences of parameter tuning on the results.

4.6 Cross-validation

An **n-fold** cross validation is performed using either the best hyperparameters (if model tuning is selected) or the default ones.

All the values predicted in the different folds are collected and the metrics are performed against the measured values. When applying a **transformation** to the input soil variable the model will be trained on the transformed values. The cross-validation metrics will be then calculated both on the transformed values as well as on the back-transformed ones.

When no transformation is applied the cross-validation metrics table will look like this:

metric	value	transformation
rmse	0.33	notransform
mae	0.21	notransform
rsq	0.47	notransform
ccc	0.62	notransform
mec	0.48	notransform

metric	value	transformation
me	0.08	notransform

The validation metrics computed are defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$\text{MEC} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

where $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

$$\text{CCC} = \frac{2\rho\,\sigma_y\,\sigma_{\hat{y}}}{\sigma_y^2 + \sigma_{\hat{y}}^2 + \left(\overline{y} - \overline{\hat{y}}\right)^2}$$

$$\text{RSQ} = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \overline{\hat{y}}_i\right)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

$$\text{ME} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$$

Where MAE is Mean Absolute Error, RMSE is Root Mean Squared Error, MEC is Model Efficiency Coefficient (Janssen and Heuberger (1995)), CCC is Concordance Correlation Coefficient, RSQ is the coefficient of determination, ME is the mean error, n is the number of observations, y_i is the i-th SOM observation, $\hat{y}_i$ the i-th SOM prediction, ρ is the Pearson correlation between predictions and observations, $\overline{\hat{y}} = \frac{1}{n}\sum_{i=1}^{n}\hat{y}_i$ and σ_y, and $\sigma_{\hat{y}}$ are the standard deviations of the observations and predictions.

4.7 Out-of-bag assessment

The final model fit is performed using 100% of the training data. Therefore, it is important to note that the accuracy metrics of the final model may not be directly known. Instead, one can consider utilising the out-of-bag (OOB) metrics from the model object. It is worth mentioning that the OOB metrics provide an estimate of the model's performance using an internally validated subset of the training data. While the accuracy metrics from cross-validation could be similar or potentially worse than those of the final model, the OOB metrics offer an alternative assessment of the model's performance.

4.8 Spatial folds

This workflow allows the user to perform a validation of the model performance using n-fold cross-validation. Cross-validation needs the samples to be in different groups called `folds`. These folds can either be generated randomly or picked based on their geographical location.

The package dggridR provides a spatial layer that enables the ramdom selection in a non spatially distorted way. Many different grid resolutions are available for many different grids. The chart here shows the number of cells, their area, and statistics regarding the spacing of their center nodes for the ISEA3H grid type.

5 Design Choices

5.1 File based

This workflow is file based. This means that each step writes one or more files to disk. These files are then the input for the next steps until the final results are created. Writing files to disk allows for memory efficiency and is in line with a modular structure where components can be used both singularly or in combinations with others.

5.2 Mask map as a reference

To run the workflow, a **mandatory mask** raster layer in GeoTiff format is **needed**. As a design choice, the mask defines the area of interest for mapping (i.e. for which the soil variable of interest will be predicted). It also defines the CRS and spatial resolution of the predictive soil maps that are generated with the workflow.

The mask layer should provide **no data (NA)** in the areas where no prediction is needed such as built-up areas, water bodies, rocky areas or glaciers. In the config file, the parameter `NA_mask_flag` can be used to specify which value of the mask should be considered *no data*.

It is best if the mask has the same **CRS**, the same **resolution** and the same **extent** as the **covariates**. If this is not the case, then all the covariates are going to be reprojected and clipped to the mask (this will slow down the process). Note that if one or more covariates have missing data for a pixel, the predictions will be missing for that pixel as well.

If no mask is available, one of the covariates can be used.

5.3 Tidymodels

This workflow makes use of the Tidymodels ecosystem. Tidymodels is a collection of R packages that provides a consistent and tidy interface for modeling and machine learning tasks in R. It aims to streamline the process of developing, evaluating, and deploying machine learning models by promoting a tidy and modular workflow.

Tidymodels packages are built on top of the popular tidyverse ecosystem, which includes packages like dplyr, ggplot2, and tidyr. The key packages within the tidymodels framework include:

- **parsnip**: Provides a consistent syntax for specifying models. It allows you to define models using a unified interface, regardless of the underlying modeling engine.
- **dials**: Offers a way to define and tune hyperparameters for models. It allows you to create sets of possible hyperparameter values and perform model tuning using techniques like grid search or random search.
- **recipes**: Provides a framework for preprocessing and feature engineering. It allows you to define a set of preprocessing steps, such as data transformations, imputation, and feature selection, and apply them consistently to different datasets.
- **workflows**: Allows you to create a workflow that combines model specification, preprocessing, and model evaluation steps into a single object. This facilitates the reproducibility and organisation of the modeling process.
- **yardstick**: Provides functions for evaluating model performance using common metrics like accuracy, precision, recall, and area under the curve (AUC).
- **tune**: Implements functionality for hyperparameter tuning using resampling and various search strategies.

By using tidymodels, you can leverage the power of the tidyverse to build end-to-end machine learning pipelines in a consistent and organised manner. It promotes a modular approach to model building, making it easier to iterate, experiment, and collaborate on machine learning projects in R.

6 FAQ

6.1 Can I have missing values in covariates?

When a covariate has `NA`s (missing values) at a given location then predictions at that location will not be possible and the resulting output map will have missing values.

Note that the NoData value should be correctly stated in the **metadata** of the raster. You can check this in R by doing:

```
terra::describe(rast("your/path/to/a/example_covariate.tif"))
```

And search for "NoData Value" in the output that you get. For example:

```
"Type=Int16"
"NoData Value=-32768"
```

If you see a value that looks like a NoData placeholder but was not declared as such in the metadata, investigate further. For instance, if the example covariate has values of -9999, which is common convention for designating `NA` in float types, you would start to suspect that some pixels that were meant to be assigned `NA` are being interpreted as a "real values" and would need to be checked.

As an alternative:

```
library(terra)
rst <- terra::rast("your/path/to/a/covariate.tif")
plot(rst)
```

And check if `NA` areas are actually transparent and no numeric values are assigned to them.

Also, we generally mask out water bodies and built-up areas, meaning that in the mask these areas are `NA`s. Following the same logic, the `NA` pixels in the mask will be reflected as `NA` pixels in the predictions.

6.2 Integer overflow

If a raster with values of −100000 is saved with the Int16 data type, which has a range of -32768 to 32767, those values will overflow and be corrupted into an arbitrary wrong number due to the datatype's limited range. They will not automatically become `NA`/NoData. To avoid this, you should either choose a datatype with a sufficient range (e.g. Int32), or scale/offset the values to fit within Int16's range before writing.

6.3 What does the "Tile n is empty" warning means?

This warning may appear while executing the `predict_tiles()` function. As mentioned in the previous question, when at least one of the pixels in the covariates used by the model is `NA`, that pixel will be `NA` in the predicted tile. It may happen that all pixels end up `NA` in a tile, for example, if the tile mostly covers a water surface. Therefore, there is no tile to write, and it does not appear in the tile list.

6.4 Using Categorical Rasters as Covariates

Yes, you can use categorical rasters as covariates. However, you must first convert them into binary rasters. For each category, create a separate raster where pixels corresponding to that category are marked with a value of 1, and all other pixels are marked with a value of 0.

Categorical data are stored as factors. For Example, R does not take 0 and 1 as 0 being lower than 1, but as 0 being a different category than 1.

6.5 How can legacy data be important in my analysis?

If you have legacy data, it is important to check positional uncertainty, as it is related to mapping resolution. If the uncertainty is high, mapping at a fine resolution may not be advisable. If the DEM has a fine resolution and the measurement uncertainty extends over a larger area, the model cannot find correlations between the covariates and the variable, since their value does not change at the same scale as the variable.

6.6 Flag columns?

Flag columns are used in this workflow to subset the points dataset. There we indicate whether we want to include (1) or not (0) certain points in a subset. Subsets are for testing (fitting models) and validating (calculating accuracy metrics). Ideally, test and validation datasets have similar basic statistics. Always try to perform some sort of stratified selection of points, rather than purely random selection.

6.7 How could I improve the model with my own scientific knowledge?

To try to improve the model, check your data, remove covariates, remove outliers, run again. As most models, the quality of the inputs is reflected in the quality of the outputs.

Always verify which covariates you use, regardless of whether the set is the result of RFE or not. As a soil scientist, does it make sense to use the selected covariates? Are these related to soil processes? You can run the seedling multiple times with different settings, evaluate the results (not only accuracy but also patterns), and, as a soil scientist, assess which results are most consistent.

6.8 MEC and bias

We can think of the Model Efficiency Coefficient (MEC), suggested `tune_metric` in Section 2.2, as the accuracy of our model compared to the average. For soil mapping, a value between 30% and 60% is already acceptable, and if it exceeds 95%, there is a high probability that the model is incorrect.

Bias can be the distance the model has from the 1:1 line in a predicted vs. observed scatter plot. If the MEC is low but the R-squared (how well a model fits the data) is high, it means the prediction is wrong, but patterns can still be observed where the actual values are low, and vice-versa.

6.9 How can I plot a scatter plot of predicted vs. observed?

One of the outputs is the `cvPredictObserveFile`, check Section 2.2. It contains the observed and predicted values of the voi. Use the tool of your choice to generate the scatter plot.

6.10 How can I quantify uncertainty?

There are two options to quantify the uncertainty, you can calculate the prediction interval (Q0.95 - Q0.05; Malone, McBratney, and Minasny (2011)) or you can calculate the uncertainty index (Q0.95 - Q0.05/Q0.50; (Poggio et al. 2021)).

Fore example you can create a map with the 90% prediction interval by computing the Q0.95 - Q0.05, which will show the uncertainty of the predictions, i.e. how much the value could vary. Make sure you include Q0.05 and Q0.95 in `stats` (see Section 2.2), and with the output GeoTIFFs, perform a pixel-by-pixel subtraction operation with your preferred tool.

6.11 UNRELIABLE VALUE: One of the foreach() iterations ('doFuture-1') unexpectedly generated random numbers without declaring so

Warning message:

```
UNRELIABLE VALUE: One of the foreach() iterations ('doFuture-1') unexpectedly generated rando
```

Upon reviewing the relevant discussions in the following GitHub threads:

- Seurat Issue #3622
- Tune Issue #377

It appears that this concern may be not pertinent or could potentially be a false positive. We will continue to conduct a thorough investigation to ensure the integrity of our project. However, based on the current evidence, it appears safe to disregard this specific message at this time.

Please let me know if further clarification is needed, or if there are any other concerns.

You can ignore the message by running

```
options(doFuture.rng.onMisuse = "ignore")
```

7 LICENCE

R-based Digital Soil Mapping workflow developed at ISRIC

Seedling is developed and maintained by ISRIC - World Soil Information.

The code in this repository is released under the GPL3 license according to the ISRIC data policy.

GNU GPL v3.0 is a strong copyleft license. This means that you may use the code and change/modify the code. If you distribute copies or modifications of the code, you are required to release these updates under the GPL v3 license.

The code is provided without warranty. ISRIC is not obliged to provide user support, updates or "bug fixes" of any kind.

For more information or collaboration opportunities please use the `dsm@isric.org` email address.

List of abbreviations

- CRS: Coordinate Reference System
- DSM: Digital Soil Mapping
- FAQ: Frequently Asked Questions
- IDE: Integrated Development Environment
- MEC: Model Efficiency Coefficient
- NA: No Data
- RFE: Recursive Feature Elimination
- RMSE: Root Mean Squared Error
- RSQ: R-squared - coefficient of determination
- VOI: Variable Of Interest

References

Arrouays, Dominique, Alex B Mcbratney, B Minasny, Jon W Hempel, GBM Heuvelink, RA MacMillan, AE Hartemink, Philippe Lagacherie, Neil J Mckenzie, et al. 2014. "The GlobalSoilMap Project Specifications." *GlobalSoilMap: Basis of the Global Spatial Soil Information System*.

Guyon, Isabelle, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. "Gene Selection for Cancer Classification Using Support Vector Machines." *Machine Learning* 46: 389–422.

Janssen, P. H. M., and P. S. C. Heuberger. 1995. "Calibration of Process-Oriented Models." *Ecological Modelling* 83 (1): 55–66. https://doi.org/https://doi.org/10.1016/0304-3800(95)00084-9.

Malone, BP, AB McBratney, and B Minasny. 2011. "Empirical Estimates of Uncertainty for Mapping Continuous Depth Functions of Soil Attributes." *Geoderma* 160 (3-4): 614–26.

Poggio, L., L. M. de Sousa, N. H. Batjes, G. B. M. Heuvelink, B. Kempen, E. Ribeiro, and D. Rossiter. 2021. "SoilGrids 2.0: Producing Soil Information for the Globe with Quantified Spatial Uncertainty." *SOIL* 7 (1): 217–40. https://doi.org/10.5194/soil-7-217-2021.

Shrestha, Durga L, and Dimitri P Solomatine. 2006. "Machine Learning Approaches for Estimation of Prediction Interval for the Model Output." *Neural Networks* 19 (2): 225–35.

Wright, Marvin N., and Andreas Ziegler. 2017. "Ranger: A Fast Implementation of Random Forests for High Dimensional Data in c++ and r." *Journal of Statistical Software* 77 (1): 1–17. https://doi.org/10.18637/jss.v077.i01.