Supplementary Material S1. Table of input covariates for the quantile random forest models built for surface (0-5 cm) and rootzone (0-1 m) soil moisture (volumetric water content).

Surface soil moisture	Satellite or model products	Variables	Abbreviation	Original spatial resolution	Original temporal resolution	Version	Reference
	Sentinel-1	backscatter at VV and VH modes and incidence angle	vv, vh, angle	10-20 m	6-12 days	Ground Range Detected (GRD) scenes	https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S1_GRD
	SMAP	Surface soil moisture	ssm	10 km	3 days	NASA-USDA Enhanced SMAP Global Soil Moisture Data	https://developers.google.com/earth-engine/datasets/catalog/NASA_USDA_HSL_SMAP10KM_soil_moisture
	MODIS	Land surface temperature	LST	1 km	Daily	MOD11A1.061 Terra	https://developers.google.com/earth-engine/datasets/catalog/MODIS_061_MOD11A1
	Landsat 8	Surface reflectance of bands 5 (near-infrared), 6 (shortwave infrared-1), 7 (shortwave infrared-2), 10 (thermal), NDVI	B5, B6, B7, B10, NDVI, NDWI	30 m or 100 m	16 days	Level 2, Collection 2, Tier 1	https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_L2

		(Normalized Difference Vegetation Index), NDWI (Normalized Difference Water Index)					
	USGS DEM	Elevation, slope, aspect, hillshade	elevation, slope, aspect, hillshade	10 m	Constant	USGS 3DEP National Map Spatial Metadata 1/3 Arc-Second	https://developers.google.com/earth-engine/datasets/catalog/USGS_3DEP_10m
	Polaris	Clay and sand content, bulk density at 0-5 cm	clay_5, sand_5, bd_5	30 m	Constant	1	Chaney et al. (2019)
	NLCD	Land cover type	landcover	30 m	Constant	USGS National Land Cover Database, 2016 release	https://developers.google.com/earth-engine/datasets/catalog/USGS_NLCD_RELEASES_2016_REL
Rootzone soil moisture	Sentinel-1	backscatter at VV and VH modes and incidence angle	vv, vh, angle	–	–	–	–
	SMAP	subsurface soil moisture	susm	10 km	3 days	NASA-USDA Enhanced SMAP Global Soil Moisture Data	https://developers.google.com/earth-engine/datasets/catalog/NASA_USDA_HSL_SMAP10KM_soil_moisture

MODIS	Land surface temperature	LST	–	–	–	–
Landsat 8	Surface reflectance of bands 5, 6, 7, 10, NDVI, NDWI	B5, B6, B7, B10, NDVI, NDWI	–	–	–	–
USGS DEM	Elevation, slope, aspect, hillshade	elevation, slope, aspect, hillshade	–	–	–	–
Polaris	Clay and sand content, bulk density at 0-1 m	clay_100, sand_100, bd_100	30 m	Constant	1	Chaney et al. (2019)
NLCD	Land cover type	landcover	–	–	–	–

Supplementary Material S2. Package installation instructions.

1. Install the latest version of RTools (e.g., RTools 4.3 or the version that is compatible with the user's R console, https://cran.r-project.org/bin/windows/Rtools/).

> install.packages(c('raster', 'rgee', 'sf', 'tidyverse', 'viridis', 'FedData', 'RColorBrewer', 'caret', 'chillR', 'leaflet', 'hydroGOF', 'quantregForest', 'randomForest', 'reshape2', 'rgdal', 'sp', 'lubridate', 'geojsonio', 'stars'))

2. Install the following dependency R packages.
3. Install R package mlhrsm. The users can install it from GitHub.

> install.packages(c("devtools","R.rsp"))
> devtools::install_github("soilsensingmonitoring/mlhrsm_1.0", build_vignettes = T)

4. Set up a Google Earth Engine account, project, and API. First, all users need to create a free Google Earth Engine account (https://earthengine.google.com/signup/). Second, install gcloud CLI before downloading maps from Google Earth Engine (https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe). Third, create a project on the Google Earth Account for future use. After installing the gcloud CLI, if a CMD window pops out (when the user enables configuration of gcloud) to ask the user to connect gcloud CLI, select "Y" to log in. Then a web page will appear with a message saying, "Google Cloud SDK wants to access your Google Account"; select Allow and return to the CMD, where the system asks the user to "Pick cloud project to use." Select the project the user wants to use, and close CMD. Lastly, relaunch R software and install Google Earth Engine API in the R environment.

> library(mlhrsm)
> ee_Initialize("Your email address", drive = T) # insert your email address

If it is the first time the user uses ee_Initialize() on the computer, R will print downloading and installation messages when preparing for the initialization. Select "Y" when R asks to install Miniconda. If the

computer does not have the Python package "earthengine-api" installed, an error message will appear, and the user should run the following command line to install it.

> .rs.restartR() # If this does not work, please restart the R session manually
> ee_install()

Then R will ask the user to store environment variables EARTHENGINE_PYTHON and EARTHENGINE_ENV in the .Renviron file to use the Python path in future sessions. Type "Y" to continue and restart the R session when prompted to do so after installation is completed. Run ee_Initialize("Your email address", drive = T) again. A new window will pop up in the browser saying, "Google Earth Engine Authenticator wants to access your Google Account"; then, select Allow to allow the local R environment to connect to the user's Google Earth Engine. If successful, the user will see the following messages in the R console. The user can now access the maps in Earth Engine from the local R environment and download them to the user's Google Drive.

Fetching credentials using gcloud
Successfully saved authorization token

Supplementary Material S3. Instructions for accessing the default ML models and refitting the models.

S3.1 Viewing the default ML models

The input training and testing data and the quantile random forest models for surface and rootzone soil moisture are saved in "all.rda", "model_surface.rda", and "model_rz.rda". These files can be accessed from the folder "mlhrsm_1.0/data/" under the R library directory. The user can also download them from the GitHub site (https://github.com/soilsensingmonitoring/mlhrsm_1.0/tree/main/data). Once downloaded, the user can input them into the R environment.

Load packages
> library_names <- c('raster', 'rgee', 'sf', 'tidyverse', 'viridis', 'FedData', 'RColorBrewer', 'caret', 'chillR', 'leaflet', 'hydroGOF', 'quantregForest', 'randomForest', 'reshape2', 'rgdal', 'sp', 'lubridate', 'geojsonio', 'stars')
> lapply(library_names, require, character.only = TRUE)
> *setwd("directory where the installed mlhrsm package is located")*
> *load("all.rda")*

```
> load("model_surface.rda")
> load("model_rz.rda")
> ## Split the data into training (calibration) and testing (validation) datasets
> all_surface <-
all[!is.na(all$VWC_5)&!is.na(all$elevation)&!is.na(all$clay_5)
    &!is.na(all$VWC_5)&!is.na(all$vv)&!is.na(all$vh)&!is.na(all$LST)
    &!is.na(all$LS_B4)&!is.na(all$LS_B10)&!is.na(all$landcover),]
> all_rz <-
all[!is.na(all$VWC_100)&!is.na(all$elevation)&!is.na(all$clay_100)
    &!is.na(all$VWC_100)&!is.na(all$vv)&!is.na(all$vh)&!is.na(all$LST)
    &!is.na(all$LS_B4)&!is.na(all$LS_B10)&!is.na(all$landcover),]
> cali_surface <- all_surface[all_surface$Validation = =0,]
> vali_surface <- all_surface[all_surface$Validation = =1,]
> cali_rz <- cali_surface[!is.na(cali_surface$VWC_100),]
> vali_rz <- vali_surface[!is.na(vali_surface$VWC_100),]
```

To display the variable/feature importance of the surface and rootzone models, run the following command lines:

```
> varImpPlot(model_surface)
> varImpPlot(model_rz)
```

To display the model performance of the surface and rootzone models at the training and testing dataset, run the following command lines:

```
> index_surface <- names(cali_surface) %in% c("landcover",
"elevation", "slope", "aspect", "hillshade",
"clay_5", "sand_5", "bd_5",
"ssm",
"vv", "vh", "angle",
"LST",
"LS_B5", "LS_B6", "LS_B7", "LS_B10", "LS_NDVI", "LS_NDWI")
> index_rz <- names(cali_rz) %in% c("landcover",
"elevation", "slope", "aspect", "hillshade",
"clay_100", "sand_100", "bd_100",
"susm",
"vv", "vh", "angle",
"LST",
"LS_B5", "LS_B6", "LS_B7", "LS_B10", "LS_NDVI", "LS_NDWI")
```

> soil_cali_mean = predict(model_surface, newdata = cali_surface[, index_surface], what = mean) ## Training (5-fold cross-validation)

> soil_vali_mean = predict(model_surface, newdata = vali_surface[, index_surface], what = mean) ## Testing

> soil_cali_rz_mean = predict(model_rz, newdata = cali_rz[, index_rz], what = mean) ## Training (5-fold cross-validation)

> soil_vali_rz_mean = predict(model_rz, newdata = vali_rz[, index_rz], what = mean) ## Testing

> ## Training cross-validation results for Surface Soil Moisture

> sqrt(mean((model_surface$predicted - cali_surface$VWC_5)^2)) # RMSE training

> mean(model_surface$predicted - cali_surface$VWC_5) # bias training

> cor(model_surface$predicted, cali_surface$VWC_5)^2 # r2 training

> NSE(model_surface$predicted, cali_surface$VWC_5) # NSE training

> KGE(model_surface$predicted, cali_surface$VWC_5) # KGE training

> ## Testing for Surface Soil Moisture

> sqrt(mean((vali_surface$VWC_5 - soil_vali_mean)^2, na.rm = T)) # RMSE Testing

> mean(vali_surface$VWC_5 - soil_vali_mean) # bias Testing

> cor(vali_surface$VWC_5 , soil_vali_mean)^2 # r2 Testing

> NSE(vali_surface$VWC_5 , soil_vali_mean) # NSE Testing

> KGE(vali_surface$VWC_5 , soil_vali_mean) # KGE Testing

Training cross-validation results for Rootzone soil moisture

> sqrt(mean((model_rz$predicted - cali_rz$VWC_100)^2)) # RMSE training

> mean(model_rz$predicted - cali_rz$VWC_100) # bias training

> cor(model_rz$predicted, cali_rz$VWC_100)^2 # r2 training

> NSE(model_rz$predicted, cali_rz$VWC_100) # NSE training

> KGE(model_rz$predicted, cali_rz$VWC_100) # KGE training

> ## Testing for Rootzone soil moisture

> sqrt(mean((vali_rz$VWC_100 - soil_vali_rz_mean)^2, na.rm = T)) # RMSE Testing

> mean(vali_rz$VWC_100 - soil_vali_rz_mean) # bias Testing

> cor(vali_rz$VWC_100 , soil_vali_rz_mean)^2 # r2 Testing

> NSE(vali_rz$VWC_100 , soil_vali_rz_mean) # NSE Testing

> KGE(vali_rz$VWC_100 , soil_vali_rz_mean) # KGE Testing

*S*3.2 Refitting the ML models

The default models were fitted using 5-fold cross-validation and 40 trees using the selected covariates in **Supplementary Material S1**. Note that the example code below only fits models using the training dataset and then applies them to the testing data to obtain the results in **Table 1**. The selection of covariates for surface and subsurface soil moisture models was determined based on the optimal testing results of the models, and the number of trees was set so that the models were not over-fitted (a similar performance was observed between the training and testing datasets). Note that the default ML models used in the **mlhrsm** package were fitted using all the data points from both training and testing stations to reach a good coverage across the contiguous USA. To build models using all available datasets, the user can simply replace the *"cali_surface"* and *"cali_rz"* in the "train" functions with "all_surface" and "all_rz", respectively.

In case the user would like to fit the models using their own dataset, please refer to the following command lines to train the ML models using the "train" function from the "caret" package. By changing *method = "qrf"* in the **caret::train** function, the users can switch to other ML algorithms. A detailed list of available ML models and their usage in the **train** function can be found here (https://topepo.github.io/caret/train-models-by-tag.html). In addition, the user can include additional in situ soil moisture datasets along with all the predictors in the same format of "all_surface" and "all_rz" to build these localized models. Once the models are built for soil moisture mapping, the user should save the models and the input data using the same names and overwrite the existing rdata ("all.rda", "model_surface.rda", and "model_rz.rda") under the installation directory of the *mlhrsm* package. After this is completed, calling the functions to map soil moisture will automatically use the updated/newly saved ML models.

```
> ## Quantile random forest parameters
> fitControl = trainControl(## 5-fold CV
method = "cv",
number = 5)
> ntree = 40
> ## Surface model
> qrf_surface = caret::train(x = cali_surface[, index_surface],
y = cali_surface$VWC_5,
na.action = na.omit,
trControl = fitControl,
```

metric = "RMSE",

ntree = ntree,

nodesize = 10,

method = "qrf")

> model_surface <- qrf_surface$finalModel

> ## Rootzone model

> qrf_rz = caret::train(x = cali_rz[,index_rz],

y = cali_rz$VWC_100,

na.action = na.omit,

trControl = fitControl,

metric = "RMSE",

ntree = ntree,

nodesize = 10,

method = "qrf")

> model_rz <- qrf_rz$finalModel

3.3 Mapping soil moisture outside the US

The users can also modify the functions to map soil moisture outside the USA and use the remaining functions to conduct spatial and temporal analysis on the generated soil moisture maps. To achieve this, two main modifications are needed.

First, because all the dynamic variables (e.g., SMAP, Sentinel-1, and MODIS) are globally available, the users can replace the existing static variables with the ones that are globally available and accessible on Google Earth Engine. For example, USGS DEM can be replaced with NASA SRTM Digital Elevation 30m product (ee.Image("USGS/SRTMGL1_003")), Polaris soil maps replaced with SoilGrids 250m map product (ee.Image("OpenLandMap/SOL")), and NLCD land cover maps replaced with MODIS landcover dataset (ee.ImageCollection("MODIS/061/MCD12Q1"), select a specific year and product). To ensure the spatial and temporal analysis functions can still be used outside the US, please modify all three data downloading functions ("large_region_prediction.R", "small_region_prediction.R", "point_prediction.R") in the source code package or directly download them from GitHub (https://github.com/soilsensingmonitoring/mlhrsm_1.0/tree/main/R).

Second, rename the variables for the newly modified static covariates so that they match with the variable names previously used in the machine learning models. It is strongly recommended that the users provide additional in situ soil moisture data to refit the models so that the models are locally trained to represent the landscape features in the study sites outside the USA. Please refer to the

previous section (**Supplementary Material 3.2**) for details on retraining the models with new datasets.