Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Machine Learning Based High-resolution Soil Moisture Mapping and Spatial-temporal Analysis across the Contiguous USA: The mlhrsm Package

Yuliang Peng
Department of Statistics, University of Wisconsin-Madison, Madison, WI 53706, USA

Jingyi Huang*
Department of Soil Science, University of Wisconsin-Madison, Madison, WI 53706, USA

Zhengwei Yang
National Agricultural Statistics Service, U.S. Department of Agriculture, Washington, DC, 20250, USA

Zhou Zhang
Department of Biological Systems Engineering, University of Wisconsin-Madison, WI 53706, USA

Abstract

Soil moisture is a key variable for a variety of applications including agricultural management, ecological modeling, weather forecasting, and environmental monitoring. It varies from field to global scales and from seconds to decades as a function of meteorological forcing, vegetation, soil texture, topography, and water resources management. Combinations of observations from ground-based sensors, remote sensing platforms, and empirical or mechanistic models have been used to retrieve and map soil moisture variations across scales. This article presents the new R package **mlhrsm**, which allows the user to generate high-resolution (30 to 500 m, daily to monthly) soil moisture maps and uncertainty estimates across the contiguous USA (CONUS) at the soil surface (0-5 cm) and rootzone (0-1 m) using a machine learning based model (ML-HRSM). The model is based on the quantile random forest algorithm and integrates in situ soil sensors collected across the CONUS, satellite-derived land surface parameters (vegetation, terrain, and soil) and satellite based models of surface and rootzone soil moisture. Compared to existing data downloading and pre-processing tools, the package also provides functions for spatial and temporal analysis of the retrieved soil moisture maps for scientific research and water resources management across scales. A case study is provided to demonstrate the functionality of the **mlhrsm** package to generate 30-m daily to weekly soil moisture maps across an 70-ha cropland, followed by a spatial-temporal analysis to understand the variations of soil moisture for field-level water resources management.

Keywords: remote sensing, quantile random forest, visualization, leaflet, spatial-temporal analysis, water resources management, R.

1. Introduction

Soil moisture is a key variable for a variety of applications including agricultural management Mladenova, Bolten, Crow, Anderson, Hain, Johnson, and Mueller (2017), ecological modeling Robinson, Campbell, Hopmans, Hornbuckle, Jones, Knight, Ogden, Selker, and Wendroth (2008), weather forecasting Dirmeyer and Halder (2016), and environmental monitoring (Korošak, Suhadolnik, and Pleteršek 2019). It varies from field to global scales and from seconds to decades as controlled by meteorological forcing (e.g., precipitation, air temperature), vegetation (e.g., evapotranspiration), soil texture, topography, and water resources management (Vereecken, Huisman, Bogena, Vanderborght, Vrugt, and Hopmans 2008). Due to the heterogeneity of soil moisture in space and time, combinations of observations from remote sensing platforms and process-based mechanistic models have been used to model and map soil moisture information across large areas and at short time intervals (Ochsner, Cosh, Cuenca, Dorigo, Draper, Hagimoto, Kerr, Larson, Njoku, Small *et al.* 2013; Babaeian, Sadeghi, Jones, Montzka, Vereecken, and Tuller 2019; Vergopolan, Chaney, Pan, Sheffield, Beck, Ferguson, Torres-Rojas, Sadri, and Wood 2021). Current soil moisture products with a global coverage include surface soil moisture retrieved directly from NASA Soil Moisture Active Passive (SMAP) mission (Entekhabi, Njoku, O'Neill, Kellogg, Crow, Edelstein, Entin, Goodman, Jackson, Johnson *et al.* 2010) and ESA Soil Moisture and Ocean Salinity (SMOS) mission (Kerr, Waldteufel, Richaume, Wigneron, Ferrazzoli, Mahmoodi, Al Bitar, Cabot, Gruhier, Juglea *et al.* 2012), as well as model-based surface and rootzone soil moisture estimated by assimilating SMAP product with water balance or land surface models (Mladenova, Bolten, Crow, Sazib, and Reynolds 2020; Reichle, De Lannoy, Koster, Crow, Kimball, and Liu 2020; Vergopolan *et al.* 2021).

With the recent advances in machine learning (ML) algorithms, researchers have also developed data-driven models for mapping soil moisture combining ground-based (*in situ*) sensors with remote sensing observations or models. Assisted with the cloud-based storage and computation infrastructure (e.g., Google Earth Engine - GEE), these data-driven models have the potential to be applied to map soil moisture at regional to global scales. Recently, Huang, Desai, Zhu, Hartemink, Stoy, Loheide, Bogena, Zhang, Zhang, and Arriaga (2020) used a quantile random forest model to map surface soil moisture (0-5 cm) globally on a 12-day basis at 100-m. The model integrated global *in situ* soil moisture sensors with satellite observations (e.g., SMAP and Sentinel-1) and remotely sensed land surface parameters (terrain and soil properties). Similarly, Greifeneder, Notarnicola, and Wagner (2021) developed a Python package to generate global surface soil moisture maps at a 50-m resolution using ML models by combining in situ soil moisture observations with satellite data. Despite success, compared to process-based models (Vergopolan *et al.* 2021), current data-driven ML models neither provide soil moisture dynamics at a high temporal resolution (currently 12-day interval limited by the revisiting time of Sentinel-1) nor did they provide estimates of soil moisture at subsurface/rootzone given most satellite platforms (e.g., optical and microwave sensors) only measure top few centimeters of soil. Obtaining field-level ($<$ 500 m spatial resolution) soil moisture information at a short time interval (e.g., subweekly) and at depths (e.g., within the rootzone) is particularly important and necessary for agricultural condition monitoring, hydrological modeling and water resources management.

Surface soil moisture	Training (5-fold Cross-validation)	Testing
Bias (m^3m^{-3})	0.000	0.005
RMSE (m^3m^{-3})	0.030	0.076
Correlation coefficient (squared, r^2)	0.942	0.636
Kling-Gupta Efficiency	0.942	0.658
Rootzone soil moisture		
Bias (m^3m^{-3})	0.000	0.008
RMSE (m^3m^{-3})	0.031	0.075
Correlation coefficient (squared, r^2)	0.938	0.650
Kling-Gupta Efficiency	0.941	0.687

Table 1: Summary statistics of model performance at the training (70 percent) and testing (30 percent) stations as shown in Figure 1.

2. Machine learning based soil moisture models used in this package

To address the problems of the existing soil moisture models, this article presents a new R package **mlhrsm**, which allows the user to generate high-resolution (30 m to 500 m) soil moisture (volumetric water content - VWC) maps and uncertainty estimates across the contiguous United States for both soil surface (0-5 cm) and rootzone (0-1 m) on a daily basis using a newly developed ML based high-resolution soil moisture (ML-HRSM) model. The model was established based on the quantile random forest algorithm (Meinshausen and Ridgeway, 2006) combining nationwide *in situ* soil moisture measurements averaged at two depths (0-5 cm and 0-1 m) with spatiotemporal resampled satellite imagery (e.g., Sentinel-1 Synthetic Aperture Radar backscatter, Landsat-8 visible, near-infrared, and thermal bands and indices), satellite-based models of soil surface and subsurface moisture (NASA-USDA Enhanced SMAP soil moisture), land surface temperature (MODIS), and ancillary land surface parameters (NLCD land cover, USGS digital elevation model and terrain attributes, Polaris soil properties at 0-5 cm and 0-1 m). The selected covariates for modeling surface and rootzone soil moisture are summarized in Appendix A.

The quantile random forest model algorithm was selected because 1) it has an overall good performance in handling covariates that have non-linear relationship with model responses (e.g., soil moisture) and are inter-correlated with each other (Chatterjee, Huang, and Hartemink 2020; Huang *et al.* 2020); 2) it directly provides model uncertainty estimates (e.g., standard deviation and percentiles), which are useful for hydrological modeling and risk assessment in decision-making. As shown in Table 1, the models trained from the randomly selected stations (70 percent, Figure 1) were applied to the randomly hold-out testing stations and have overall r^2 (Pearson's correlation coefficient squared) of 0.636 and 0.650, bias of 0.005, 0.008 m^3m^{-3}, and RMSE of 0.075 and 0.076 m^3m^{-3}, for modeling the surface (0-5 cm) and rootzone (0-1 m) soil VWC within the CONUS, respectively. Note that the random split of *in situ* soil moisture stations are only for evaluating the ML models and the final models used both training and testing stations for fitting the models to make predictions of soil VWC across the CONUS.

In addition to produce application ready high spatial and temporal resolution surface and rootzone soil moisture data, the **mlhrsm** package also provides functions for spatial and temporal analysis of the retrieved soil moisture maps for scientific research and water resources

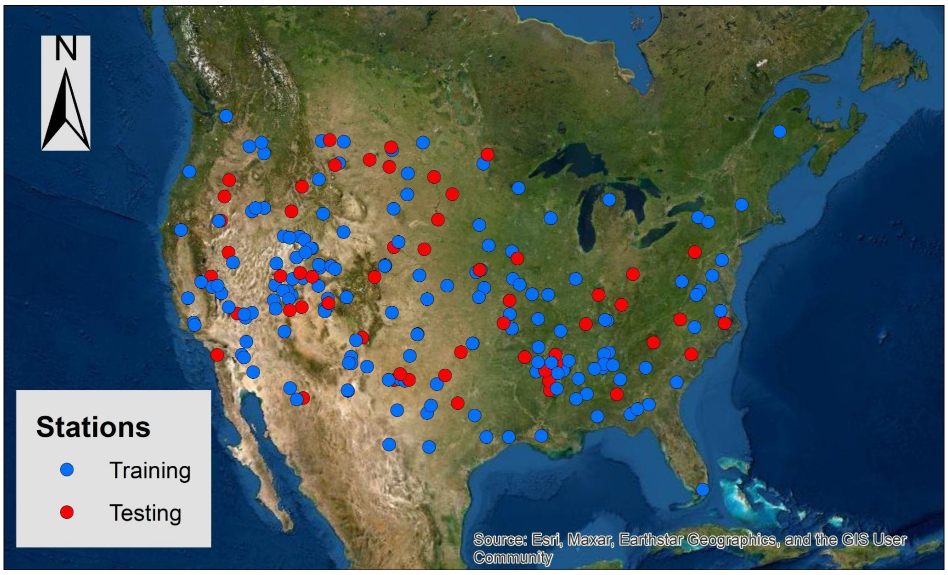
N
Stations
● Training
● Testing
Source: Esri, Maxar, Earthstar Geographics, and the GIS User Community

Figure 1: Locations of training (70 percent) and testing (30 percent) stations randomly selected across the CONUS for evaluating the ML model performance.

management across scales. A case study is provided to demonstrate the functionality of the **mlhrsm** package to obtain 30-m soil moisture maps across an 80-ha cropland field and at selected sites, Wisconsin, USA, followed by spatial-temporal analysis to understand the variations of soil moisture for field-level water resources management.

3. Overview of the functionality of mlhrsm

The R package has been developed and tested on the Windows operation system with R software (version 4.2.1). To use the **mlhrsm** package (version 1.0), the user should first create a Google Earth Engine account (`https://earthengine.google.com/signup/`), and then install the gcloud CLI software (`https://cloud.google.com/sdk/docs/install#windows/`). A number of dependency R packages are needed for the **mlhrsm** package, including **raster**, **rgee**, **sf**, **tidyverse**, **viridis**, **FedData**, **RColorBrewer**, **caret**, **chillR**, **leaflet**, **hydroGOF**, **quantregForest**, **randomForest**, **reshape2**, **rgdal**, **sp**, **lubridate**, **geojsonio**, and **stars**. After loading the **mlhrsm** package, packages **sf**, **raster**, **tidyverse**, and **rgee** are automatically loaded as well. For detailed instructions on package installation, the user can refer to Appendix B. The main functionality of **mlhrsm** is described in the following diagram of data flows and function usage provided as shown in Figure 2.

3.1. Main functions

The main functions provided by the package are used to download satellite data and apply the ML models to the selected areas or sites. To use these functions, the user is expected to provide an input file of the region of interest (ROI, in the shapefile format) or GPS coordinate locations (in the csv format) for soil moisture retrieval. It is recommended that the input files be stored in the root directory of the **mlhrsm** package.

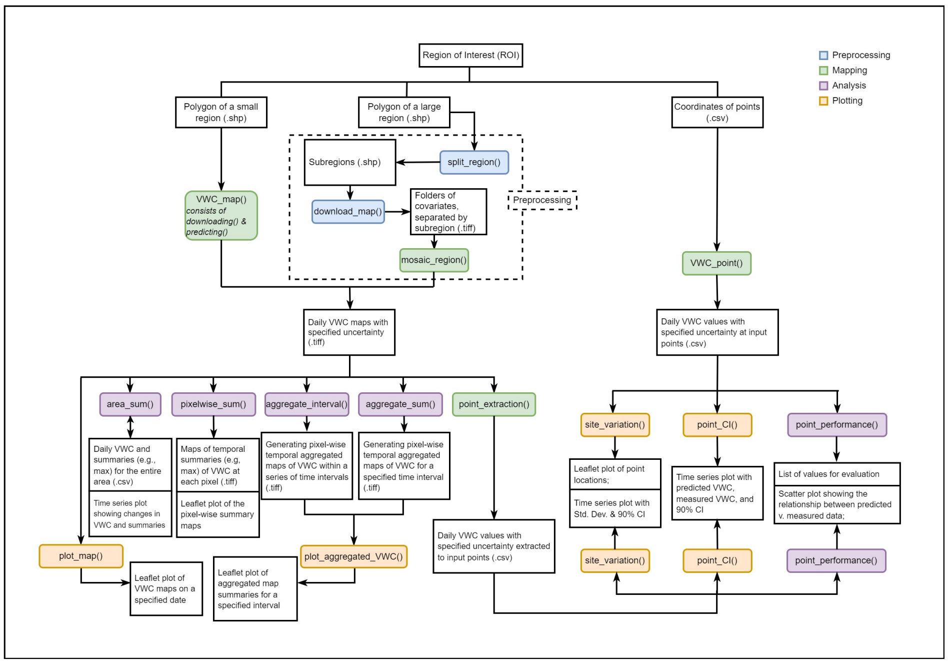
Region of Interest (ROI)
Polygon of a small region (.shp)
Polygon of a large region (.shp)
Coordinates of points (.csv)
Preprocessing
Mapping
Analysis
Plotting
Subregions (.shp)
split_region()
download_map()
Folders of covariates, separated by subregion (.tiff)
Preprocessing
VWC_map() consists of downloading() & predicting()
mosaic_region()
VWC_point()
Daily VWC maps with specified uncertainty (.tiff)
Daily VWC values with specified uncertainty at input points (.csv)
area_sum()
pixelwise_sum()
aggregate_interval()
aggregate_sum()
point_extraction()
site_variation()
point_CI()
point_performance()
Daily VWC and summaries (e.g., max) for the entire area (.csv)
Time series plot showing changes in VWC and summaries
Maps of temporal summaries (e.g, max) of VWC at each pixel (.tiff)
Leaflet plot of the pixel-wise summary maps
Generating pixel-wise temporal aggregated maps of VWC within a series of time intervals (.tiff)
Generating pixel-wise temporal aggregated maps of VWC for a specified time interval (.tiff)
Leaflet plot of point locations;
Time series plot with Std. Dev. & 90% CI
Time series plot with predicted VWC, measured VWC, and 90% CI
List of values for evaluation
Scatter plot showing the relationship between predicted v. measured data;
plot_map()
plot_aggregated_VWC()
Daily VWC values with specified uncertainty extracted to input points (.csv)
site_variation()
point_CI()
point_performance()
Leaflet plot of VWC maps on a specified date
Leaflet plot of aggregated map summaries for a specified interval

Figure 2: Flowchart of the functions of the mlhrsm package.

Note that the established (default) ML models and the input *in situ* soil moisture datasets (Figure 1) used for training and evaluating the models can also be accessed from the R package. Previous studies have suggested that the inclusion of local training datasets in a regional or global soil moisture model can significantly improve the model performance in regions with sparse training datasets (Huang *et al.* 2020). The user can also choose to modify the default ML models by incorporating local training dataset (*in situ* soil moisture measurements) collected from their study fields. Details about building the models for the **mlhrsm** package are provided in Appendix C.

3.1.1 `VWC_map`

The **mlhrsm** package provides three ways to apply the established ML models to generate VWC maps. The basic function `VWC_map` allows the user to download input covariates and map VWC across a small ROI in one step. To use the `VWC_map` function, the user is expected to provide the name of the shapefile of the ROI (preferably in the WGS84 system, “EPSG:4326”), start and end dates specifying the period of daily soil moisture maps to be retrieved (in the format of “yyyy-mm-dd”), and the spatial resolution of the output VWC maps (in the unit of meter). If indicated by the user (percentile=TRUE), the model will also calculate the upper and lower bounds of the 90 percent Confidence Interval (CI) of the VWCs predicted from the quantile random forest models. The outputs of this function are raster files of covariates and predicted VWCs (daily mean and standard deviation (sd) of VWC at 0-5 cm and 0-100 cm depths, with or without CI of VWC) in GEOTIFF format. The output maps will be saved in the NAD83 system and USA Contiguous Albers Equal Area Conic, USGS (“EPSG:5070”) for water resources management in the USA. A project name is needed in the function so that all the downloaded covariates and VWC maps will be saved in a folder under the working directory named after the project name. This project name will also be used later for spatial-temporal analysis and visualization with other functions.

With the limitation of the availability of several input satellite covariates (e.g., Sentinel-1 backscatter data), the earliest date of the available VWC map is January 1st, 2016 (no sufficient Sentinel-1 data prior to that date for soil moisture estimation), and the end date lasts to the present (ongoing until the end of the mission of Sentinel-1, SMAP, MODIS, or Landsat 8). The finest spatial resolution is 30 m and the user can specify other resolutions up to 500 m. Any mapping attempt beyond the ranges of the input arguments will result in errors.

The `VWC_map` function involves three main steps as briefly described below (refer to Appendix A for details). First, the function will download various input covariates data from the GEE within the ROI, including;

a) constant land surface parameters: 30-m National Land Cover Dataset (NLCD) land cover maps in 2016, 10-m elevation data from the USGS 10-m digital elevation model and derived slope, aspect, and hillshade, 30-m Polaris soil clay and sand content and bulk density maps at 0-5 cm and 0-1 m, and

b) dynamic variables spanning the input period of VWC maps (with a buffer period of 6-64 days for temporal interpolation depending on the available satellite data): 30-m 12-day Sentinel-1 backscatter data measured at VV and VH polarizations and incidence angle (masked for outliers and despeckling following Huang *et al.* (2020), 1-km daily SMAP land surface temperature, Landsat-8 bands 5, 6, 7, and 10, and NDVI and NDWI indices, and NASA-USDA Enhanced SMAP 10-km surface and subsurface soil moisture storage maps.

Second, the constant covariates will be resampled across the ROI to the spatial resolution defined by the user using bilinear interpolation and the dynamic covariates will be resampled spatially to the input resolution and temporally to a daily basis using bilinear interpolation.

Lastly, the pre-established ML models will be applied to the processed satellite covariates within the ROI to generate VWC maps at the surface and rootzone at the specified spatial resolution on a daily basis.

3.1.2 `VWC_point`

In addition to generating soil moisture maps across a certain area of interest using `VWC_map`, the `VWC_point` function is created for predicting VWC at individual points for the user interested in exploring the VWC dynamics at specific locations, especially with points located far from each other (e.g., in different US states). The `VWC_point` is similar to `VWC_map`, except that the former generates a csv file containing the input points' IDs and coordinates along with the VWC values extracted at 30 m (with the points located at the center of the 30-m pixels) while the latter produces VWC maps across the input ROI in the Geotiff format.

3.1.3 `split_region, download_map, and mosaic_region for large ROIs`

For large-size satellite imagery, GEE often tends to split the file into multiple Geotiff files, which can lead to memory errors when these files are post-processed locally in the R envi-

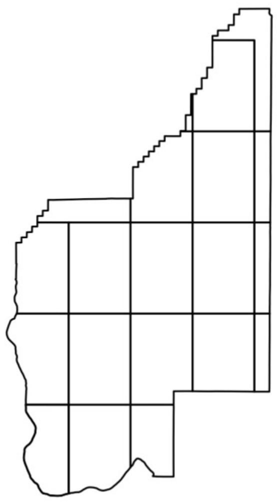

Figure 3: Plot of the subregions after splitting an approximately 7,000 km^2 ROI of Grant County, WA, USA.

ronment (e.g., mosaicing, masking, resampling). In this case, `VWC_map` will return a message directing the user to split the ROI and mapping task into subregions using `split_region`, `download_map`, and `mosaic_region`. Although the user can still map soil moisture across large regions with `VWC_map`, it is suggested that these alternate functions be used to save computation time when the input ROI is large ($>=$ 1,000 km^2) or a high input image resolution ($<$ 50 m) is specified within an ROI covering a catchment or larger ($>=$ 500 km^2).

The `split_region` requires the input of an ROI polygon (and project name if the user wants to save it in a new folder rather than the working directory). It will split the large polygon stored in the root directory into several small subregions with a cell size of 0.25 arc-degrees (approximately 500 km^2) and save the split subregions as a new shapefile with a default name of `sub_regions.shp`. If there are multiple subregions with an area smaller than 250 m^2, the `split_region` function will first merge them into one subregion to decrease the total number of subregions to reduce the data processing and downloading time in the GEE. The total number of subregions will be printed after splitting is finished. An example of the splitting process is provided below with the subregions shown in Figure 3.

```
R> library("\pkg{mlhrsm}")
R> split_region("Grant.shp", "Grant2021")
R> sub_grant <- read_sf("Grant2021/sub_regions.shp")$geometry
R> plot(sub_grant)
```

After splitting the large ROI into smaller subregions, the user can choose to download the input covariates using the `download_map` function and produce VWC maps in all subregions sequentially on one computer. Alternatively, the user could use multiple computers to download covariates and map VWC in different subregions in parallel to reduce the processing time. Note that the latter requires multiple GEE accounts to be activated on different computers to prevent all submitted tasks from being assigned to the same GEE account and held on a long waiting line. The `download_map` function requires similar user inputs as the `VWC_point`, except for an additional `sub_area` argument that allows the user to specify the subregion IDs in which the function will download the satellite covariates. This argument can either be assigned to a single number (subregion ID), indicating a specific subregion, or a vector of IDs for multiple subregions.

After the covariate maps are downloaded for each subregion, the `mosaic_region` function needs to be run to generate soil VWC maps in each subregion and merge the VWC maps into one master map once the mapping is completed. If one or several subregions do not have any Sentinel-1 (due to its flight path) or Landsat (due to cloud contamination) imagery on specified dates, the mean values of the corresponding Sentinel-1 or Landsat imagery across the entire ROI (covering all subregions) on those dates are used to fill these NA values on a daily basis. This gap-filling is used to avoid sharp artifacts at the boundaries of the neighboring subregions. Afterward, the ML models for surface (0-5 cm) and rootzone (0-1 m) VWC will be applied to the subregions to generate maps of VWC and CIs (if selected). Lastly, VWC maps from all subregions will be automatically mosaiced into master maps for the original large ROI on a daily basis, similar to those VWC maps produced from the `VWC_map` function.

3.2. Functions for spatial and temporal analysis

Apart from the main mapping functions, **mlhrsm** also provides spatial-temporal analysis and plotting functions to assist with the interpretation and visualization of the output maps. The plotting functions extract and plot the VWC maps and their spatial statistics within different time intervals for a specific ROI or locations. The spatial-temporal analysis functions calculate the VWC summary statistics, including mean, sd, min, and max in space or time, and round the summary statistics to the third decimal place.

`3.2.1 Area-based/zonal functions`

`plot_VWC` plots the statistical values (mean, sd, and CI if percentile=TRUE) of VWC maps for one specified date ("yyyy-mm-dd") and depth (0-5 cm or 0-1 m) with the leaflet package. It returns an interactive leaflet map showing the soil VWC values of the study area at the defined spatial resolution on a selected date. The user can choose to display the mean, sd, or CI values of the VWC maps estimated from the quantile random forest models. The user can also plot VWC maps on multiple days and compare the change in soil moisture (see case study below).

`area_sum` summarizes the area-based/zonal statistical values (mean, min, max, and sd) of the daily VWC at a specified depth (0-5 cm or 0-1 m) across the entire ROI. The output of this function includes the calculated statistical values on a daily basis as a csv file (`VWC_depth_ts_data.csv`) in the specific project folder for further analysis and a time series plot showing the changes in these summary statistics of VWC over the entire study period on a daily basis during the study period. If the ROI represents a specific field, farm, or catchment, then the time series plot will indicate the field-, farm-, or catchment- averaged VWC statistics during the study period.

Unlike `area_sum`, `pixelwise_sum` returns the summary statistics of VWC at a specified depth calculated pixel-wise across the ROI area over the study period, displayed as area maps of these statistics in leaflet. The maps are also saved as Geotiff raster files in a folder called `temporal_VWC` under the project folder. These area maps delineate subregions within the

ROI area that has wetter/drier soils over a long period than neighboring places.

To facilitate the interpretation of VWC dynamics at different temporal scales (e.g., daily vs. weekly vs. seasonally), `aggregate_sum` and `aggregate_interval` provide methods to aggregate the daily VWC maps according to specified time intervals. `aggregate_sum` has similar functionality as `pixelwise_sum`, except that the user can specify a different start and end date (needs to be within the initial study period provided in the `VWC_map function`. `aggregate_sum` then summarizes the VWC maps at a specified depth pixel-wise over the newly defined period and outputs the summary statistics (pixel-wise mean, median, min, max, sd across the ROI) as Geotiff maps in the `VWC_aggregation` folder. It also saves the summary statistics as separate csv files along with each grid pixel's coordinates. Different from `aggregate_sum`, `aggregate_interval` allows the user to define the time intervals/temporal scale for the aggregation. Besides a start and end date, the user will provide a scale/frequency argument (number of days) representing the aggregation interval (e.g., 7 means converts daily VWC maps to weekly averaged VWC maps). The `aggregate_interval` function will return a list of dates indicating the starting dates of each interval, and save the pixel-wise temporally aggregated maps named after the list of the dates in the `VWC_aggregation` folder, along with the csv files of the grid coordinates and aggregated VWC values on the new temporal scale/intervals. If the total number of days (between the newly defined start and end dates) is not a multiple of the frequency, several days (less than the length of frequency) will be left behind, and the summaries of VWC maps over these days will still be calculated and saved as output data (unless there is only one day left).

Once the temporal aggregation is done, `plot_aggregated_VWC` serves as a plotting function specifically for the aggregated maps. It needs inputs of a start and end date and returns a similar output plot in leaflet as the `pixelwise_sum`. If the user wants to visualize the map saved by `aggregate_interval` (e.g., a weekly VWC map), they need to provide a specific date (starting date of one of the day intervals) and the frequency for the function to locate and plot the processed files.

3.2.2 Point-based functions

For users who want to investigate VWC dynamics at specific locations within the produced VWC maps, `point_extraction` can be used. The user needs to specify the site locations by assigning the name of the shapefile (saved in the same project folder) as the function argument. `point_extraction` will then extract the predicted VWC values (mean, sd, and lower and upper bounds of 90 percent CI if such files exist in the project's VWC folder) to the points and save the resulting site coordinates and daily VWCs as a csv file with a default name of `VWC_point_data.csv`. The user should note that if no specific filename is identified, the saved csv file will overwrite the older version in the project folder, which may result in loss of data. If a specific filename is given by the user, the VWC values will be saved according to the specified filename. The user is then able to run the plotting and evaluation functions for point VWCs at a certain depth from the csv file produced.

`site_variation` visualizes the change in VWC values at specified point locations over the study period defined when running the `VWC_map` and `point_extraction` or `VWC_point` function. The output of this function includes a leaflet plot showing the locations of the selected points (specified in the `site_variation` function) and time series plots of the predicted VWC at different point locations over time, with mean +/- sd highlighted in shaded regions and dashed lines for the upper and lower bounds of the 90 percent CI (when percentile=TRUE). If no site location is specified, the function will automatically plot VWC values at all the points. When the total site number exceeds 12, a random selection of 12 points will be included in the output plot. The function inputs data from the `VWC_point_data.csv` file by default but can be applied to another extracted point dataset by specifying it in the function.

`point_CI` also returns a time series plot. It provides an evaluation of the model performance at the selected point by plotting the 90 percent CI of the predicted VWC values along with the measured VWC values from an *in situ* soil moisture sensor at a specified depth (the measured data need to be provided in the input csv file). The plot follows a similar algorithm as `site_variation`. The user can choose the sites to visualize, and if no site is specified, a total of 12 randomly chosen points will be presented. In the visualization plot, the shaded region represents the 90 percent CI of daily VWC values, with solid lines representing predicted mean VWC values from the machine learning models and dashed lines as measured values from *in situ* sensors.

To fully evaluate the model performance at selected sites with *in situ* VWC measurements, `point_performance` can be used to calculate a number of performance metrics between the sensor-measured VWC and model-derived VWC values, including coefficient of determination (R^2), root-mean-square-error (RMSE), Nash–Sutcliffe efficiency (NSE) (McCuen et al., 2006), Kling-Gupta efficiency (KGE) (Gupta et al., 2009), Bias, ratio of performance to deviation (RPD), and ratio of performance to inter-quartile (RPIQ) (Wijewardane et al., 2016). The evaluation metrics will be returned in a list together with a scatter plot showing predicted VWC against measured VWC at the selected sites. A diagonal (1:1) line is also included in the plot. If indicated (stats=TRUE), the validation values of R^2, RMSE, Bias, and KGE will also be presented on the scatter plot.

To use the `point_CI` and `point_performance` functions, the user should provide a csv file of the measured VWC (from *in situ* soil sensors), which includes site ID, coordinates (Longitude, Latitude), Date (in "yyyy-mm-dd" format), and depth interval (`VWC_5` or `VWC_100` for averaged VWC at 0-5 cm or 0-1 m, respectively). The evaluation csv file can be stored outside the working directory and passed to the function with the full directory when needed.

4. Case study

4.1. Study area

A 70-ha cropland field in Rock County, Wisconsin, USA is used here as an example to demonstrate the functionalities of the **mlhrsm** package. In total, 12 *in situ* soil moisture sensors (TEROS 12, Meter Group, Inc.) were installed across the field and soil moisture data were collected at 5 cm depth from June to September 2020.

4.2. Generating soil moisture maps at different spatial resolutions

Since the ROI has a small area of 0.7 km^2, after the **mlhrsm** package is installed and loaded into the R environment, and a GEE account is activated (refer to Appendix 2), the user can use the `VWC_map` to generate soil VWC maps directly for the study field and plot the results using `plot_map`.

```
R> VWC_map("WI.shp", "2020-06-15", "2020-09-15", 30, TRUE, "WI_region")
R> plot_map("2020-08-01", 5, TRUE, project="WI_region")
```

The estimated soil VWC values at the soil surface (0-5 cm) across the study field on 2020-08-01 are displayed as a leaflet map object using the `plot_map` function (Figure 4). The default layer is set at the mean. The user can select different layers of standard deviation (SD) and upper (0.95) and lower (0.05) bounds of the 90 percent confidence interval produced by the quantile random forest model, if computed by `VWC_map`. The map has a spatial resolution of 30 m, pre-determined in the `VWC_map` function and can no longer be changed during plotting (unless the user resamples the Geotiffs using other R functions). As shown in Figure 4, pixels highlighted in red indicate dry regions of the field (VWC of 0.15-0.16 m^3 m^{-3}), while pixels in blue indicate wet regions of the field (VWC of 0.21-0.23 m^3 m^{-3}). In summary, the field was dry on the selected day during the crop growing season.

4.3. Spatial and temporal analysis

Compared to other existing packages that generate soil moisture maps (Greifeneder *et al.* 2021), one major contribution of the **mlhrsm** package is its spatial-temporal analysis tools. For example, the user can use the `function area_sum` to calculate and save the statistical summaries (mean, sd, min, and max) at a certain depth of the daily average VWC across the entire ROI as shown below:

```
R> area_sum(5, project="WI_region")
R> head(read.csv("WI_region/VWC_5_ts_data.csv"))
```

The function `area_sum` saves zonal statistics of soil VWC values across the entire ROI and returns a time series plot of its temporal variations. As shown in Table 2, the Date variable indicates the dates for which the summaries are calculated, and Summary defines the type of summary for each value. The changes in the VWC summary characteristics are presented in different color for different statistical values and visualized on a daily basis as in Figure 5. It is noted that the predicted VWC for the study area (ROI) increases to a peak around June 22, 2020 (due to a major rainfall event), then decreases gradually until the end of August (due to water uptake by soil and plant evapotranspiration), where the VWC values increased again to another peak in mid-September (due to another major rainfall event).

The user can also calculate and save the maps of the summary statistics of each pixel at a certain depth by calling `pixelwise_sum` as shown below:

```
R> pixelwise_sum(100, project="WI_region")
```

After running the `pixelwise_sum`, six summary maps of pixel-wise Mean, minimum (Min), Median, maximum (Max), standard deviation (SD) and Range of the VWC values at the

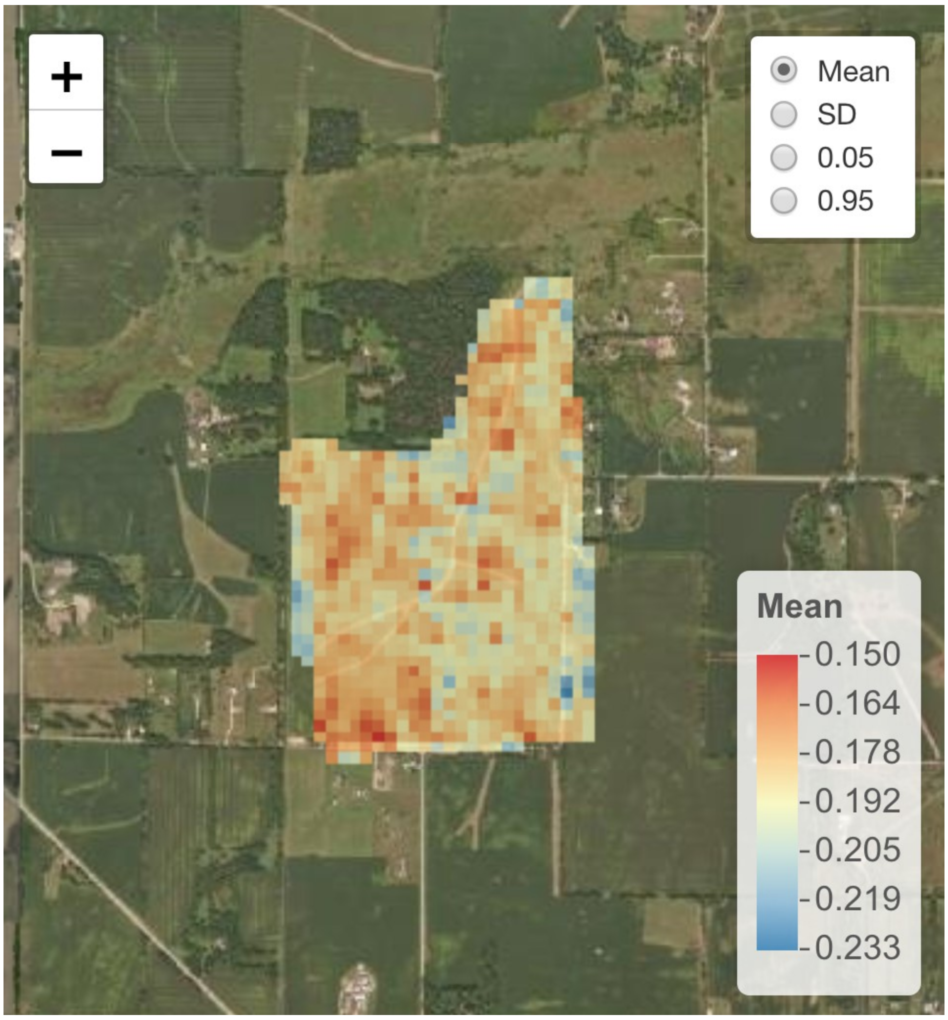

Figure 4: A leaflet visualization of the mapped VWC of a specific date and depth is returned upon calling plot map.

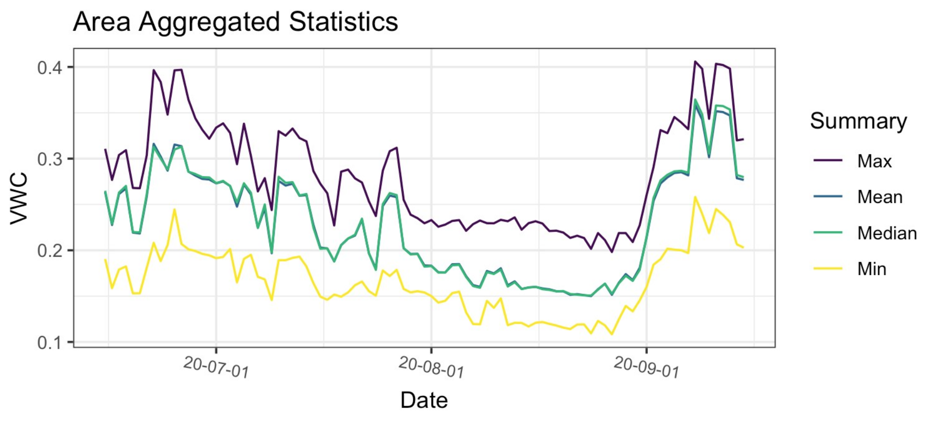

Figure 5: Time series plot output of area sum.

	Date	Summary	Value
1	2020-06-15	Mean	0.264
2	2020-06-15	Median	0.265
3	2020-06-15	Min	0.190
4	2020-06-15	Max	0.311
5	2020-06-16	Mean	0.228
6	2020-06-16	Median	0.229

Table 2: Subset of VWC 5 ts data.csv, produced by area sum.

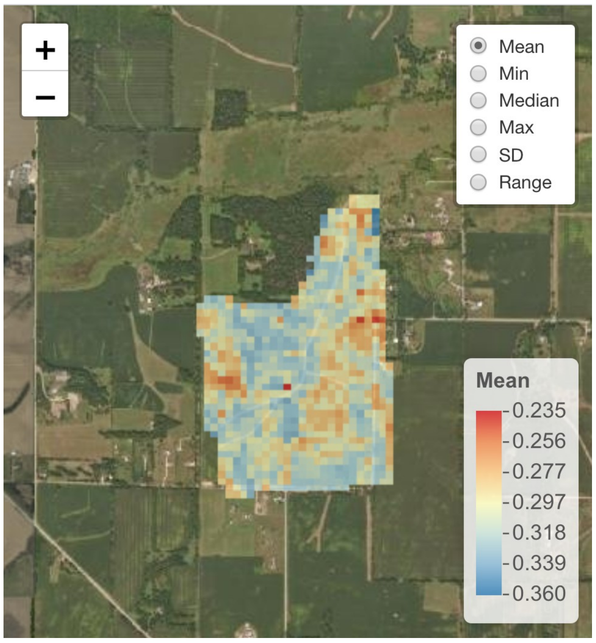

Figure 6: Leaflet plot output of pixelwise sum.

rootzone (0-1 m, 100 means 100 cm) across the entire ROI area from 2020-06-15 to 2020-09-15, produced by the `VWC_map`, will be automatically saved and visualized. As shown in Figure 6, the plots consist of the six map layers, with a similar color scale as the `plot_VWC` function output. The pixel-wise mean VWC values across the ROI from 2020-06-15 to 2020-09-15 ranged between 0.23 and 0.36 m^3 m^{-3}. The uneven distribution of soil VWC is mostly caused by the spatial variations of soil texture (e.g., clay and sand content) given the field has a relatively uniform elevation (refer to Chatterjee, Hartemink, Triantafilis, Desai, Soldat, Zhu, Townsend, Zhang, and Huang (2021) for details about this field). Note that if the user sets a large ROI and/or with higher spatial resolution, the calculation time for the `pixelwise_sum` will increase accordingly.

The **mlhrsm** package also allows to calculate the statistical summaries for the pre-mapped VWCs across multiple intervals of time. To do this, the user can apply `aggregate_interval` on a subset of entire range of time defined by `VWC_map` as below:

```
R> aggregate_interval(5, "2020-06-15", "2020-08-15", frequency=7,
+ project="WI_region")
```

	x	y	layer	date
1	561512.8	2196288	0.228	2020-06-15
2	561542.8	2196288	0.248	2020-06-15
3	561572.8	2196288	0.221	2020-06-15
4	561602.8	2196288	0.236	2020-06-15
5	561512.8	2196258	0.227	2020-06-15
6	561542.8	2196258	0.246	2020-06-15

Table 3: A subset of 7 day mean.csv, saved by aggregate interval.

```
R> head(read.csv("WI_region/VWC_aggregation/2020-06-15_2020-08-15
  + /7days_5cm/7_day_mean.csv"))
```

The pixel-wise summary statistics for the 7-day intervals are saved in designated directories as csv files and Geotiff maps. Table 3 shows the first six rows of the saved csv file on average pixel-wise VWC values. x and y represent the map pixels' coordinates (in NAD83 projection system, "EPSG:5070"); layer stores the pixels' mean values, and date shows the starting date of each time interval. The user can use `plot_aggregated_VWC` to visualize the maps saved by `aggregate_interval` with the same inputs and a date parameter specifying the starting date of the time intervals to plot. `aggregate_sum` can be run following a similar logic, except that no starting date needs to be defined when plotting.

```
R> plot_aggregated_VWC(5, "2020-06-15", "2020-08-15", date="2020-06-29",
  + frequency=7, project="WI_region")
```

`plot_aggregated_VWC` returns a similar leaflet plot as `pixelwise_sum` with four layers of Mean, Min, Median, Max, and SD of the pixels' VWC values during the 7-day interval starting from 2020-06-29 (Figure 7). From the plot, it was noted that the study area had an average surface (0-5 cm) VWC ranging from 0.19-0.32 m3 m-3 during the week of June 29, 2020, and some of the highest mean VWC values were located at the southwestern corner of the ROI.

The user can also compare the VWCs on a daily v.s. weekly basis using the results saved by `VWC_map` and `aggregate_interval`, as demonstrated below:

```
R> # define the dates to plot
R> dates <- c("2020-07-06", "2020-07-13", "2020-07-20", "2020-07-27")

R> # define the files (with specific VWC statistical characteristics) to plot
R> pattern <- paste(paste0("VWC_5_mean_", dates), sep="", collapse="|")
R> maps <- stack(list.files("WI_region/VWC", pattern, full.names=T))

R> # set at same scale
R> plot(maps, zlim=c(0.15, 0.35))
```

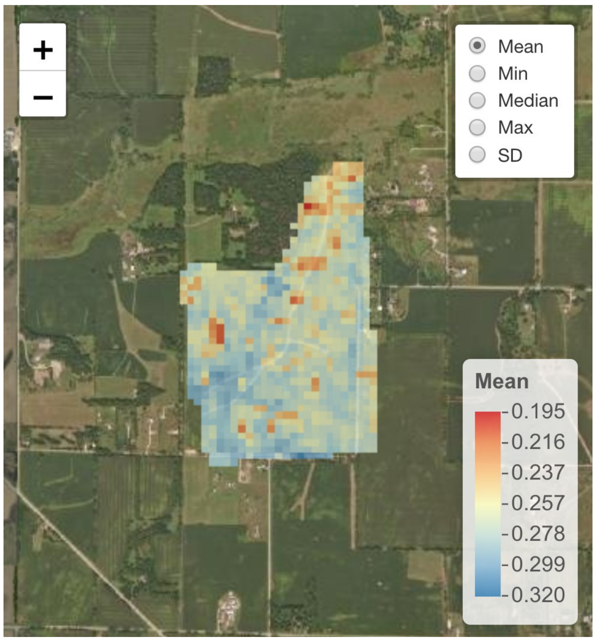

Figure 7: Leaflet plot of the 7-day VWC average starting from 2020-06-29, as calculated by aggregate interval.

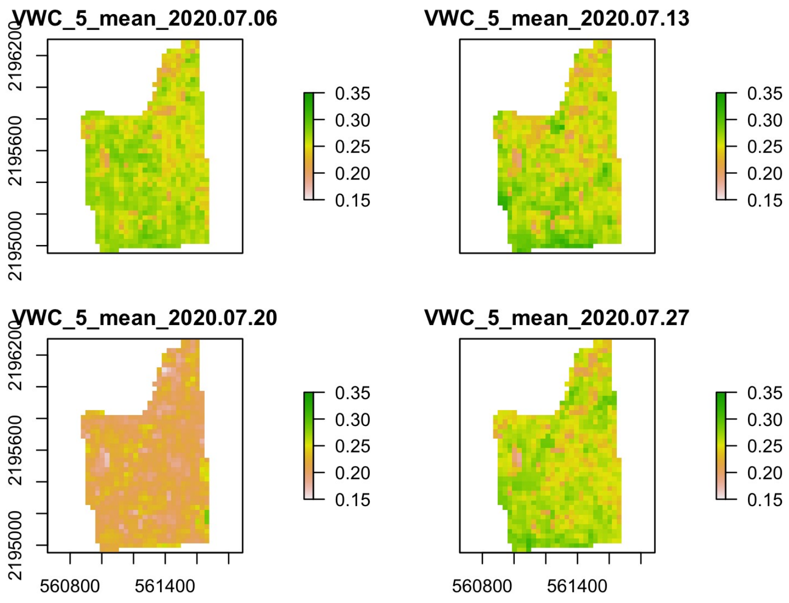

Figure 8: Four daily VWC maps, extracted from outputs saved by VWC map.

The daily VWC maps (Figure 8) show that the predicted surface soil moisture of the study area was similarly wet on the four days except for 2020-07-20, which was drier than the other three days. The trend of VWC maps was consistent with the time series plot (Figure 5), produced from the `area_sum`, where the average VWC values for July 6, 13, and 27 were approximately at the same level while the mean VWC on July 20 was lower than the other

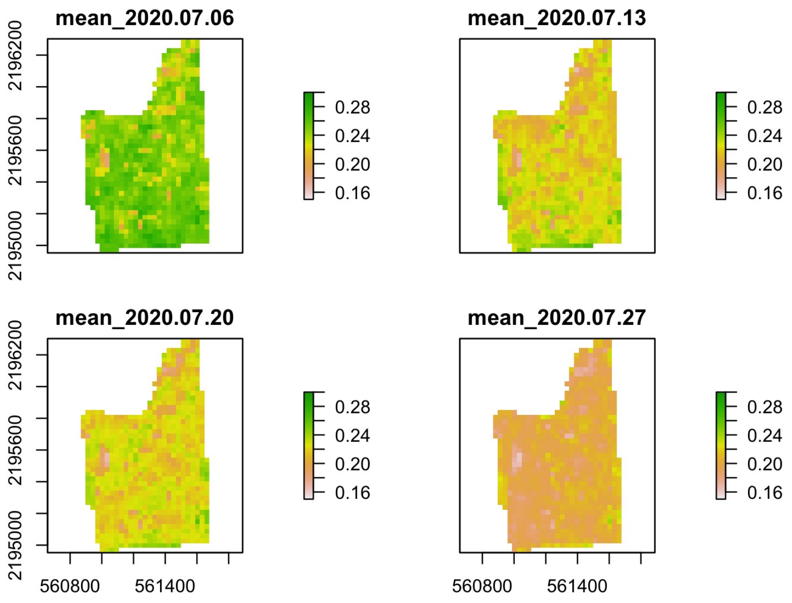

Figure 9: Four weekly VWC maps, extracted from outputs saved by aggregate interval.

three. On the other hand, when plotted using the weekly VWC maps (Figure 9), the week of 2020-07-27 experienced the lowest surface VWC value, while the week of 2020-07-06 had the wettest, and the weeks of July 13 and 20 had similar intermediate VWC values:

```
R> WD <- "WI_region/VWC_aggregation/2020-06-15_2020-08-15/7days_5cm"
R> pattern <- paste(paste0("mean_", dates), sep="", collapse="|")
R> maps <-stack(list.files(paste0(WD, "/aggregated_VWC", pattern,
  +full.names=T)))
R> plot(maps, zlim=c(0.15, 0.3))
```

Both daily and weekly maps were consistent with the time series plot shown in Figure 5. It should be also noted that although daily VWC on 2020-07-20 was the driest among the four, seeing from a larger temporal scale (weekly), the week of 2020-07-27 became drier compared with the other three weeks. This suggests that the `Aggregate_interval` function is useful for inspecting the intermediate- to long- term trend in VWC while daily VWC dynamics produced by the `VWC_map` capture short-term variations in soil moisture. A combination of these two functions will help the user to understand the time-dependent variations/memories in VWC for a specific area (Koster and Suarez 2001).

4.4. Extract soil moisture data at individual sites

To extract soil VWC values from the generated maps to the points where measurements are available (saved in the `WI_SM.shp` file), the user can apply `point_extraction`:

```
R> point_extraction("WI_SM.shp", TRUE, project="WI_region")
R> head(read.csv("WI_region/VWC_point_data.csv"))
```

```
        ID        Longitude         Latitude          Date         VWC_5_mean_pts
1        S2         -89.11825         42.57247          2020-06-15         0.268
2        S2         -89.11825         42.57247          2020-06-16         0.212
3        S2         -89.11825         42.57247          2020-06-17         0.249
4        S2         -89.11825         42.57247          2020-06-18         0.255
5        S2         -89.11825         42.57247          2020-06-19         0.212
6        S2         -89.11825         42.57247          2020-06-20         0.210


Continued
VWC_5_sd_pts          VWC_100_mean_pts          VWC_100_sd_pts          VWC_5_lower_pts
0.082          0.356          0.097          0.153
0.074          0.363          0.101          0.106
0.087          0.358          0.098          0.092
0.073          0.358          0.097          0.148
0.078          0.348          0.103          0.112
0.081          0.348          0.101          0.103


Continued
VWC_5_upper_pts          VWC_100_lower_pts          VWC_100_upper_pts
0.391          0.224          0.471
0.331          0.199          0.476
0.367          0.206          0.476
0.374          0.205          0.476
0.358          0.188          0.468
0.354          0.188          0.466
```

The output above shows the VWC values extracted using `point_extraction`. The ID represents site identification, Longitude and Latitude are the coordinates of the sites. Date corresponds to the date on which the VWC values are extracted, which should have the same range as defined in `VWC_map` (start date to end date). Columns `VWC_5_mean_pts` through `VWC_100_upper_pts` are the extracted VWC values (mean, sd, lower and upper bounds of 90 percent CI) at the specific depth (5 or 100 cm). The user should be aware that `point_extraction` does not calculate VWC CIs for the points. Therefore, if the user did not save CI values in the mapping process (via `VWC_map` or `mosaic_region`), the 90 percent CI option in `point_extraction` (percentile=TRUE) will not be applicable when extracting values for the points. After obtaining the point VWC values, the user can plot the dynamics of VWC at a chosen depth for different sites using `site_vairation`. This function can also be used on data generated from `VWC_point`, whose output has the same format as that of `point_extraction`.

```
R> plots <- site_variation(5, TRUE, project="WI_point")
R> plots[[1]] # returns leaflet plot showing site locations
R> plots[[2]] # returns time series plot showing point variation over-time
```

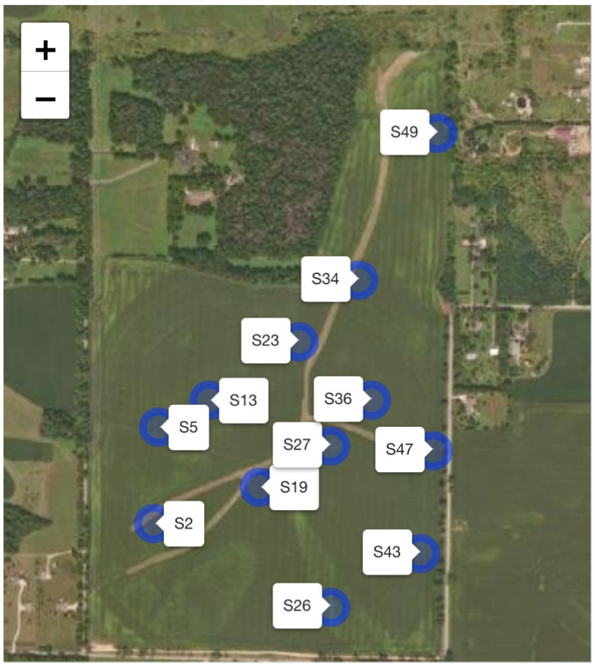

Figure 10: Leaflet plot marking the point locations.

The first plot returned from `site_variation` is a leaflet plot indicating the locations of the study sites, as shown in Figure 10. The sites are highlighted using fixed circle makers and ID labels for easier identification. Since no sites are specified in the code, the second plot returned by `site_variation`, as shown in Figure 11, is a time series plot visualizing the changes in VWC of 12 sites, faceted by site IDs, and the dates on the x-axis are labeled in the format of "yy-mm-dd." In the time series plot, solid lines represent the mean, and gray bands represent the standard deviation; 90 percent CI will be added in the form of dashed lines (if available). The time series plot showed that all 12 sites experienced similar trends in the change in surface-level soil moisture, as controlled by precipitation and crop evapotranspiration, with low VWC values during August 2020 and an increase in the VWC values at the beginning of September. The magnitude of the VWC changes differed with several sites having a steeper increase than others (S2, S13, etc.), as controlled by environmental factors such as soil texture and topography (both used as covariates in modeling soil moisture variations at the field level).

4.5. Evaluation of model estimation with ground truth data

After VWC information is extracted to the points by `point_extraction`, `point_CI` and `point_performance` can be used to evaluate the performance of the models by comparing the predicted values to the measured data (saved in `"Huges_VWC.csv"`). The two evaluation functions can also be applied to the point-based VWC datasets obtained from `VWC_point` with the same input parameters.

```
R> point_CI("Huges_VWC.csv", 5, project="WI_region")
```

`plot_CI` returns a similar time series plot as `site_variation`. As shown in Figure 12, most

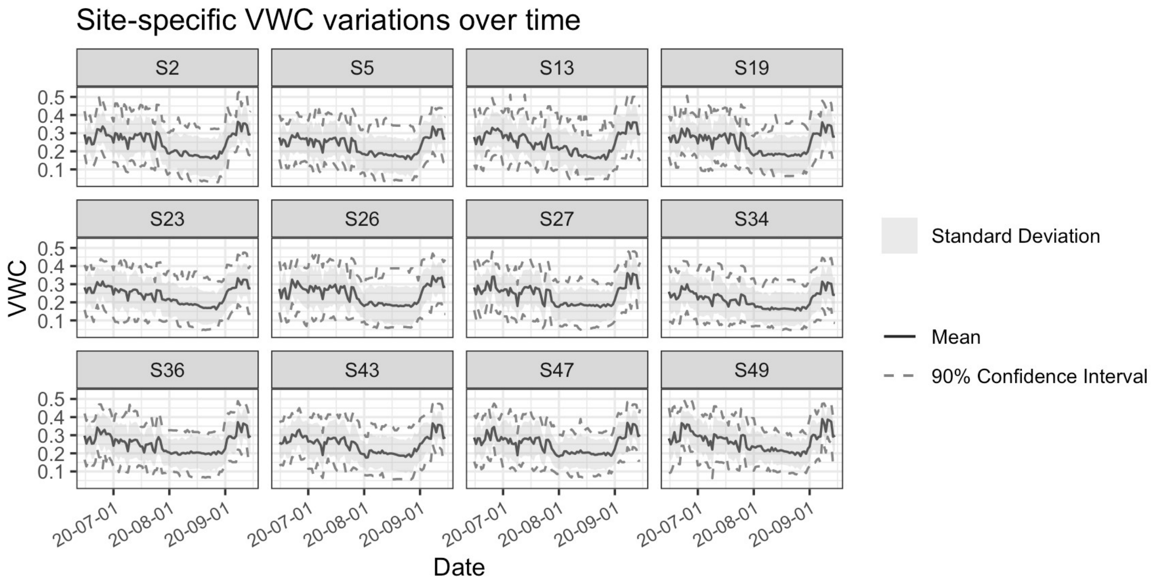

Figure 11: Time series plot showing the trends in change of VWC at 12 sites.

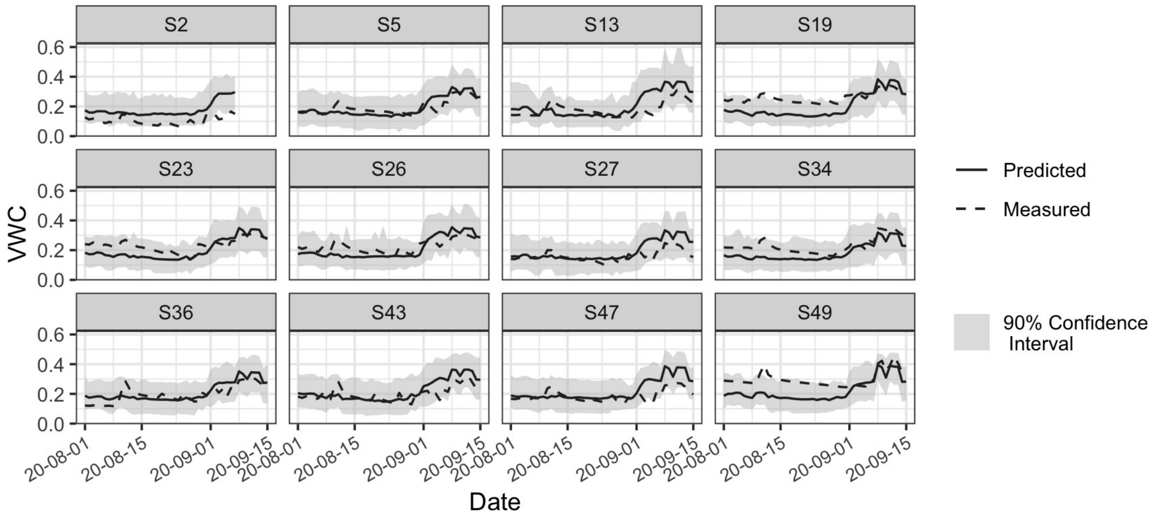

Figure 12: Time series plot as a result of calling point CI.

of the extracted VWC values on the points have a similar trend as the measured VWC values, with most of the measured values within the 90 percent confidence interval of the predicted values. To further evaluate the model performance at the selected fields, the user can calculate the performance statistics (R^2, RMSE, NSE, KGE, etc.) by running `point_performance`:

```
R> point_performance("Huges_VWC.csv", 5, TRUE, project="WI_region")
```

Figure 13 illustrates the scatter plot of the predicted VWC against the field-measured VWC with the metrics R^2, RMSE, Bias, and KGE. The predictions were smoothed as compared to

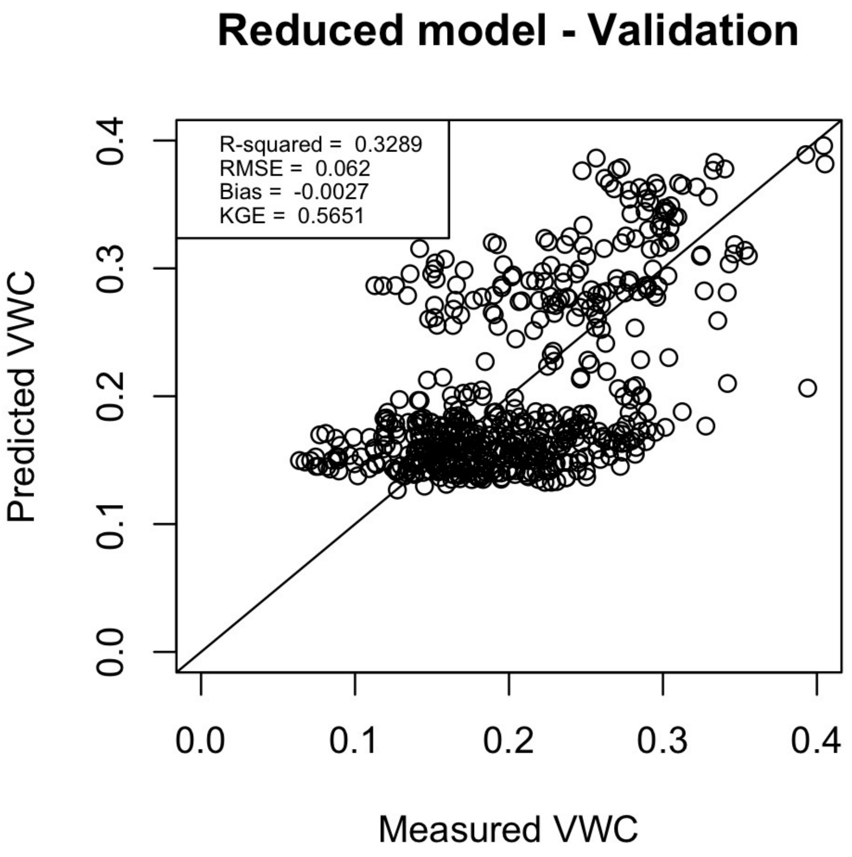

Figure 13: Evaluation plot returned by point performance.

the actual values (due to the use of quantile random forest models), which yielded a relatively low R^2 of 0.329 (Pearson's $r = 0.57$). However, the small RMSE and bias (0.06 and -0.00 m^3 m^{-3}) and an intermediate KGE of 0.565 indicate the modeling result is reasonable at such a high spatial resolution given the soil texture is extremely heterogeneous across the study field (formed from glacial outwash, see data from (Chatterjee *et al.* 2021). The mediocre model performance at this site is also consistent with other studies that attempted to delineate soil moisture at the subfield level (e.g., 30-m resolution). For instance, when assimilating a land surface model with SMAP satellite data, due to the field-scale soil heterogeneity, the modeled and measured soil VWC at the soil surface had a median temporal correlation coefficient of 0.73±0.13 and a median KGE of 0.52±0.20 across the CONUS (Vergopolan *et al.* 2021).

4.6. Usability evaluation

Careful selection of input arguments of the package's various functions is needed for mapping soil moisture of any study area or at multiple locations given the tradeoff between spatial resolution/area size/number of study sites and computation time. Here, we provide some basic computational performance statistics for different sizes of mapping areas in Table 4

Function	No. dates	Size	Resolution	CI	Computing time	Output size
VWC map						
Small region	45	0.703 km^2	100 m	T	33 min	1.93 MB
Large region	30	7229.816 km^2	500 m	F	110 min	82.1 MB
VWC point						
	30	10 points		T	260 min	1.4 MB
	60	20 points		T	792 min	5.1 MB
	90	30 points		T	764 min	4.1 MB

Table 4: Computation time of the VWC map and VWC point functions for different tasks.

for the user, which can help the user to select suitable parameters for different tasks. The computer used for testing the parameters has a 64-bit Windows 10 operation system, with a 12th Gen Intel(R) Core(TM) i5-12500 at 3.00 GHz, and 128 GB of RAM. Based on the results shown in Table 4 and Section 4.5, when the soil in the study area is strongly heterogeneous, the user should set the map resolution to a lower value (e.g., 100 m) to achieve a balance between the accuracy of the models as well as computation time.

4.7. Sustainability plan

The current ML-HRSM product relies on several satellite data as input covariates and produces surface and subsurface soil moisture estimates from 2016-01-01 to the present (approximately end of 2022). In case of failure, decommissioning, or adding certain satellite products in the future, an updated version of the ML-HRSM product and the **mlhrsm** package will be produced using replacement satellite products, including VIIRS/NPP VNP21A1D for MODIS land surface temperature, Landsat 9 and Sentinel-2 for Landsat 8 bands and indices, Global Land Data Assimilation System (GLDAS) or North American Land Data Assimilation System (NLDAS) Noah Land Surface Model for precipitation and ET and SMAP-derived surface and subsurface soil moisture.

4.8. Code availability

The source code of the **mlhrsm** package is available on GitHub (`https://github.com/soilsensingmonitoring/ml-hrsm_1.0`). The datasets for the case study are also available for downloading (`https://github.com/soilsensingmonitoring/ml-hrsm_1.0/tree/main/data`). The documentation and "Help Pages" for the **mlhrsm** package and all the functions can be accessed by searching the package name "**mlhrsm**" under the "Packages" tab within the RStudio after the package is installed. Descriptions of the input and output of the all the functions from the **mlhrsm** package can be found after the user click on the **mlhrsm** package name under the "Packages" tab. We have also created a Google Group email list (`ml-hrsm@googlegroups.com`) for continuous user support on applications of the soil moisture maps for scientific research and water resources management and will collect feedback from users for future improvement of the product.

5. Conclusions

This paper describes a novel R package, **mlhrsm**, developed for generating Machine Learning-based High-Resolution Soil Moisture (ML-HRSM) maps for soil surface (0-5 cm) and rootzone (0-1 m) at 30 to 500 m from daily to seasonally or time-series data across the continental United States. The user can choose the spatial and temporal resolutions of the soil moisture maps based on the knowledge of soil variability of the study site and management needs. It has a number of built-in functions for spatial and temporal analysis of the produced soil moisture maps or time series data. It is envisioned that a combination of the mapping functions and spatial-temporal analysis tools will help researchers to better understand the water cycles across scales and inform land managers with field-level soil moisture information for water resources management.

6. Acknowledgments and Disclaimer

Authors would like to thank Randy Hughes and Wilden Hughes (Hughes Farms, Wisconsin, USA) for allowing us to access their field to collect *in situ* soil moisture measurements used in the demonstration example and for Sumanta Chatterjee and Jie Hu for assistance in the field data collection. Support for this research was provided by the University of Wisconsin–Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation. The findings and conclusions in this publication are those of the authors and should not be construed to represent any official USDA or U.S. Government determination or policy.

Computational details

The results in this paper were obtained using R 4.2.1 with the **mlhrsm** 1.0, **raster** 3.5-29, **sf** 1.0-8, **tidyverse** 1.3.2, **viridis** 0.6.2, **FedData** 2.5.7, **RColorBrewer** 1.1-3, **caret** 6.0-93, **chillR** 0.72.8, **leaflet** 2.1.1, **hydroGOF** 0.4-0, **quantregForest** 1.3-7, **randomForest** 4.7-1.1, **reshape2** 1.4.4, **rgdal** 1.5-32, **sp** 1.5-0, **lubridate** 1.8.0, **geojsonio** 0.9.4, **stars** 0.5-6, **devtools** 2.4.4, and **R.rsp** 0.45.0 packages. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/`.

References

Babaeian E, Sadeghi M, Jones SB, Montzka C, Vereecken H, Tuller M (2019). "Ground, proximal, and satellite remote sensing of soil moisture." *Reviews of Geophysics*, **57**(2), 530–616. `doi:10.1029/2018RG000618`.

Chatterjee S, Hartemink AE, Triantafilis J, Desai AR, Soldat D, Zhu J, Townsend PA, Zhang Y, Huang J (2021). "Characterization of field-scale soil variation using a stepwise multi-sensor fusion approach and a cost-benefit analysis." *Catena*, **201**, 105190. `doi:10.1016/j.catena.2021.105190`.

Chatterjee S, Huang J, Hartemink AE (2020). "Establishing an empirical model for surface soil moisture retrieval at the US Climate Reference Network using Sentinel-1 Backscatter and Ancillary Data." *Remote Sensing*, **12**(8), 1242. `doi:10.3390/rs12081242`.

Dirmeyer PA, Halder S (2016). "Sensitivity of numerical weather forecasts to initial soil moisture variations in CFSv2." *Weather and Forecasting*, **31**(6), 1973–1983. doi:10.1175/WAF-D-16-0049.1.

Entekhabi D, Njoku EG, O'Neill PE, Kellogg KH, Crow WT, Edelstein WN, Entin JK, Goodman SD, Jackson TJ, Johnson J, *et al.* (2010). "The soil moisture active passive (SMAP) mission." *Proceedings of the IEEE*, **98**(5), 704–716. doi:10.1109/JPROC.2010.2043918.

Greifeneder F, Notarnicola C, Wagner W (2021). "A machine learning-based approach for surface soil moisture estimations with google earth engine." *Remote Sensing*, **13**(11), 2099. doi:10.3390/rs13112099.

Huang J, Desai AR, Zhu J, Hartemink AE, Stoy PC, Loheide SP, Bogena HR, Zhang Y, Zhang Z, Arriaga F (2020). "Retrieving Heterogeneous Surface Soil Moisture at 100 m Across the Globe via Fusion of Remote Sensing and Land Surface Parameters." *Frontiers in Water*, **2**, 578367. doi:10.3389/frwa.2020.578367.

Kerr YH, Waldteufel P, Richaume P, Wigneron JP, Ferrazzoli P, Mahmoodi A, Al Bitar A, Cabot F, Gruhier C, Juglea SE, *et al.* (2012). "The SMOS soil moisture retrieval algorithm." *IEEE transactions on geoscience and remote sensing*, **50**(5), 1384–1403. doi:10.1109/TGRS.2012.2184548.

Korošak Ž, Suhadolnik N, Pleteršek A (2019). "The implementation of a low power environmental monitoring and soil moisture measurement system based on UHF RFID." *Sensors*, **19**(24), 5527. doi:10.3390/s19245527.

Koster RD, Suarez MJ (2001). "Soil moisture memory in climate models." *Journal of hydrometeorology*, **2**(6), 558–570. doi:10.1175/1525-7541(2001)002<0558:SMMICM>2.0.CO;2.

Mladenova IE, Bolten JD, Crow W, Sazib N, Reynolds C (2020). "Agricultural drought monitoring via the assimilation of SMAP soil moisture retrievals into a global soil water balance model." *Frontiers in big Data*, **3**, 10. doi:10.3389/fdata.2020.00010.

Mladenova IE, Bolten JD, Crow WT, Anderson MC, Hain CR, Johnson DM, Mueller R (2017). "Intercomparison of soil moisture, evaporative stress, and vegetation indices for estimating corn and soybean yields over the US." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **10**(4), 1328–1343. doi:10.1109/JSTARS.2016.2639338.

Ochsner TE, Cosh MH, Cuenca RH, Dorigo WA, Draper CS, Hagimoto Y, Kerr YH, Larson KM, Njoku EG, Small EE, *et al.* (2013). "State of the art in large-scale soil moisture monitoring." *Soil Science Society of America Journal*, **77**(6), 1888–1919. doi:10.2136/sssaj2013.03.0093.

Reichle R, De Lannoy G, Koster RD, Crow WT, Kimball JS, Liu Q (2020). "SMAP L4 Global 3-hourly 9 km EASE-Grid Surface and Root Zone Soil Moisture Geophysical Data, Version 4." doi:10.5067/KPJNN2GI1DQR. URL https://nsidc.org/data/SPL4SMGP/versions/4.

Robinson DA, Campbell CS, Hopmans JW, Hornbuckle BK, Jones SB, Knight R, Ogden F, Selker J, Wendroth O (2008). "Soil moisture measurement for ecological and hydrological watershed-scale observatories: A review." *Vadose Zone Journal*, **7**(1), 358–389. doi:10.2136/vzj2007.0143.

Vereecken H, Huisman JA, Bogena H, Vanderborght J, Vrugt JA, Hopmans JW (2008). "On the value of soil moisture measurements in vadose zone hydrology: A review." *Water resources research*, **44**(4). doi:10.1029/2008WR006829.

Vergopolan N, Chaney NW, Pan M, Sheffield J, Beck HE, Ferguson CR, Torres-Rojas L, Sadri S, Wood EF (2021). "SMAP-HydroBlocks, a 30-m satellite-based soil moisture dataset for the conterminous US." *Scientific data*, **8**(1), 1–11. doi:10.1038/s41597-021-01050-2.

Surface	Satellite	Variables	Abbreviation
	Sentinel-1	backscatter (VV,VH), angle	vv, vh, angle
	SMAP	Surface soil moisture	ssm
	MODIS	Land surface temperature	LST
	Landsat 8	Bands 5-7,10, NDVI, NDWI	B5-B7, B10, NDVI, NDWI
	USGS DEM	Elevation, slope, aspect, hillshade	–
	Polaris	Clay, sand, bulk density 0-5 cm	clay 5, sand 5, bd 5
	NLCD	Land cover type	landcover
Rootzone	Sentinel-1	–	vv, vh, angle
	SMAP	subsurface soil moisture	susm
	MODIS	–	LST
	Landsat 8	–	–
	USGS DEM	–	–
	Polaris	Clay, sand, bulk density 0-1 m	clay 100, sand 100, bd 100
	NLCD	–	–

Table 5: 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). – means the same as surface soil moisture model.

Spatial resolution	Frequency	Version
10-20 m	6-12 days	Ground Range Detected (GRD) scenes
10 km	3 days	NASA-USDA Enhanced SMAP Global Soil Moisture Data
1 km	Daily	MOD11A1.061 Terra
30 m or 100 m	16 days	Landsat 8 surface reflectance Level 2, Collection 2, Tier 1
10 m	Constant	USGS 3DEP National Map
30 m	Constant	1, Chaney et al. (2019)
30 m	Constant	USGS National Land Cover Database, 2016 release
–	–	–
–	–	–
–	–	–
–	–	–
–	–	–
–	–	–
–	–	–

Table 6: Table of input covariates, continued. – means the same as surface soil moisture model.

A. Appendix Table of input covariates for the ML models.

B. Appendix Package installation instructions

1. Install the latest version RTools (RTools 4.2 or the version that is compatible to user's R console, https://cran.r-project.org/bin/windows/Rtools/).

2. Install the following dependency R packages.

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

3. Install R package mlhrsm. The user can install it from GitHub.

```
R> install.packages(c("devtools","R.rsp"))
R> devtools::install_github("soilsensingmonitoring/ml-hrsm_1.0", build_vignettes=T)
```

4. Set up 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 map 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 go back 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.

```
R> ee_Initialize("Your email address", drive=T) # insert your email address
```

If it's 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.

```
R> .rs.restartR()
R> ee_install()
```

Then R will ask the user to store environment variables `EARTHENGINE_PYTHON` and `EARTHENGINE_ENV` in the .Renviron file to use Python path in future sessions. Type "Y" to continue, and restart 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
```

C. Appendix Instructions for accessing and updating the 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 `"ml-hrsm_1.0/data/"` under the R library directory. The user can also download them from the GitHub site (`https://github.com/soilsensingmonitoring/ml-hrsm_1.0/tree/main/data`). Once downloaded, the user can input them to the R environment.

```
R> setwd("directory where the installed mlhrsm package is located")
R> load("all.rda")
R> load("model_surface.rda")
R> load("model_rz.rda")
```

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

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

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

```
R> ## Training (5-fold cross-validation)
R> soil_cali_mean  = predict(model_surface, newdata = cali, what = mean)
R> ## Testing
R> soil_vali_mean  = predict(model_surface, newdata = vali, what = mean)
```

```
R> ## Training cross-validation results
R> sqrt(mean((model_full$predicted[!is.na(model_full$predicted)] -
+ cali$VWC_5[!is.na(model_full$predicted)])^2))  # RMSE training
R> mean(model_full$predicted[!is.na(model_full$predicted)] -
+  cali$VWC_5[!is.na(model_full$predicted)])   # bias training
R> cor(model_full$predicted[!is.na(model_full$predicted)],
+ cali$VWC_5[!is.na(model_full$predicted)])^2   # r2 training
R> NSE(model_full$predicted[!is.na(model_full$predicted)],
+ cali$VWC_5[!is.na(model_full$predicted)]) # NSE training
R> KGE(model_full$predicted[!is.na(model_full$predicted)],
+ cali$VWC_5[!is.na(model_full$predicted)]) # KGE training
```

```
R> ## Testing
R> sqrt(mean((vali[!is.na(soil_vali_mean),]$VWC_5 -
+ soil_vali_mean[!is.na(soil_vali_mean)])^2, na.rm = T))   # RMSE Testing
```

```
R> mean(vali$VWC_5 - soil_vali_mean, na.rm = T)    # bias Testing
R>  cor(vali[!is.na(soil_vali_mean),]$VWC_5 ,
+ soil_vali_mean[!is.na(soil_vali_mean)])^2   # r2 Testing
R>  NSE(vali[!is.na(soil_vali_mean),]$VWC_5 ,
+ soil_vali_mean[!is.na(soil_vali_mean)])  # NSE Testing
R>  KGE(vali[!is.na(soil_vali_mean),]$VWC_5 ,
+ soil_vali_mean[!is.na(soil_vali_mean)])  # KGE Testing
```

In case the user would like to fit the models using their own dataset, please refer to the following commend lines to train the quantile random forest models using the "train" function from "caret" package. The default models were fitted using 5-fold cross-validation and 40 trees using the selected covariates in Appendix Table 1. Note that the example code below only fit models using the training dataset and then apply 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 observed between the training and testing datasets).

```
R> ## Split the data into training and testing
R> cali_surface <- all[all$Validation==0,]
R> vali_surface <- all[all$Validation==1,]
R>  cali_rz <- cali_surface[!is.na(cali_surface$VWC_100),]
R>  vali_rz <- vali_surface[!is.na(vali_surface$VWC_100),]
R> all_rz <- all[!is.na(all$VWC_100),]


R> ## Quantile random forest parameters
R> fitControl = trainControl(## 5-fold CV
    method = "cv",
    number = 5)
R> ntree = 40


R> ## Surface model
R> index_surface <- names(cali) %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")

R> qrf_surface = caret::train(x = cali[, index_surface],
                               y = cali$VWC_5,
                               na.action = na.omit,
                                trControl = fitControl,
                                metric = "RMSE",
                                ntree = ntree,
                                nodesize = 10,
                                method="qrf")
R> model_surface <- qrf_surface$finalModel
```

```
R> ## Rootzone model
R> 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")
 R> qrf_rz = caret::train(x = cali_rz[,index_rz],
                             y = cali_rz$VWC_100,
                             na.action = na.omit,
                             trControl = fitControl,
                             metric = "RMSE",
                             ntree = ntree,
                             nodesize = 10,
                             method="qrf")
 R> model_rz <- qrf_rz$finalModel
```

The default ML models used in the mlhrsm package were fitted using all the data points from training and testing stations. To build such models, the user can simply replace the `"cali_surface"` and `"cali_rz"` in the “train” functions with `"all"` and `"all_rz"`, respectively. Similarly, the user can include additional datasets in the same format of `"all"` and `"all_rz"` and build their localized models. To apply the updated models 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.

Affiliation:

Jingyi Huang
Department of Soil Science
University of Wisconsin-Madison
1525 Observatory Drive, Madison, WI, USA
E-mail: jhuang426@wisc.edu
URL: https://soilsensingmonitoring.soils.wisc.edu/
GitHub: https://github.com/soilsensingmonitoring/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd