gNATSGO – Import via WCS for DSM comparisons

D G Rossiter david.rossiter@wur.nl

08-January-2024

Contents

Introduction

gNATSGO "is a USDA-NRCS Soil & Plant Science Division (SPSD) composite database that provides complete coverage of the best available soils information for all areas of the United States and Island Territories. It was created by combining data from the Soil Survey Geographic Database (SSURGO), State Soil Geographic Database (STATSGO2), and Raster Soil Survey Databases (RSS) offsite link image into a single seamless ESRI file geodatabase."

It is thus the most authoritative digital product, representing many decades of field work and subsequent compilation.

Web Coverage Service (WCS) access is now provided by NRCS; we use that in this script.

This script creates a tile for a property and depth slice, over a Area of Interest delimited by geographic coordinates, in the WGS84 geographic coordinate reference system (EPSG:4326).

This tile can then be compared with other DSM products.

To use this script:

1. Adjust the directory structure to your system

Steps 2–4 refer to the YAML headers of the R Markdown source document, or external calls with `knitr::render`. You can ask to be prompted for the parameters with the R command at the console:

```
rmarkdown::render("gNATSGO_WCS_import.Rmd", output_format = "html_document",
                  params = "ask")
```

or you can specify them directly, e.g.,

```
rmarkdown::render("gNATSGO_WCS_import.Rmd", output_format = "html_document",
                    params = list(lrc_long = -76, lrc_lat = 42,
                                  size = 1, quantile.n = 2, voi.n = 3, res = 90))
```

See the respective sections of the code for the parameter definitions.

2. Select a property and its value (representative value, lower or upper estimated limits) and select a depth slice, using the YAML header or by knitting with parameters.
3. Select an Area of Interest, typically a $1 \times 1°$ tile, using the YAML header or by knitting with parameters.
4. Select a grid resolution, using the YAML header or by knitting with parameters.
5. Either compile to HTML or PDF ("knit") from the R Markdown toolbar, or "Run All" within R Markdown, or call `rmarkdown::render` from the R command line, as explained above.
6. The processed tile will be in the directory structure, in a subdirectory named for the AOI.

Packages and Drivers

```
library(sf)
library(terra)
library(aqp)
library(soilDB)
library(tidyverse)
library(rasterVis)
```

Directories

One directory is used for the original files (map unit grids), and another for the attribute tiles. The latter will be used in the DSM comparison.

Set these to areas on your own system.

```
base.dir.gnatsgo <- path.expand("~/ds_reference/Compare_DSM/gNATSGO/")
base.dir.gnatsgo.import <- path.expand("~/tmp/Compare_DSM/gNATSGO/")
```

Parameters

Parameters for this run:

```
print(paste("lrc_long:", params$lrc_long, "; lrc_lat:",
            params$lrc_lat, "; size:", params$size))
```

```
[1] "lrc_long: -76 ; lrc_lat: 42 ; size: 1"
```

```
print(paste("voi.n:", params$voi.n, "; quantile.n:",
            params$quantile.n, "; depth.n:", params$depth.n))
```

```
[1] "voi.n: 4 ; quantile.n: 2 ; depth.n: 1"
```

Property of interest and its value

The following properties can be compared to SoilGrids250 and other DSM products.

Convert the "quantile" corresponding to 5%, 50%, 95%, mean in SoilGrids and POLARIS to: `_l` (low), `_r` (representative value), `_h` (high) and `_r` (again) as a suffix to the property name. These are all estimates by expert opinion.

```
# which property?
voi.list.sg <- c("clay", "silt", "sand", "phh2o", "cec", "soc", "bdod", "cfvo")
# which value? l, r, h, r
voi.val <- c("l", "r", "h", "r")[match(params$quantile.n,
                                       c("1", "2", "3", "4"))]
# which variable of interest?
voi.list.gnatsgo <- paste(c("claytotal", "silttotal", "sandtotal",
                            # note SOM not SOC
                            "ph1to1h2o", "cec7", "om",
                            # which value? l, r, h, r
                            "dbthirdbar", "sieveno10"),
                          voi.val, sep="_")
```

Select a property by its position in the list, and make a full name from it:

```
(voi.name <- voi.list.gnatsgo[params$voi.n])
```

```
[1] "ph1to1h2o_r"
```

Depth of interest

Depth slices as specified by SoilGrids and gNATSGO.

```
depth.list.sg <- c("0-5", "5-15", "15-30", "30-60", "60-100", "100-200")
depth.list.gnatsgo <- c("05", "515", "1530", "3060", "60100", "100200")
```

Select a depth slice by its position in the list, based on the YAML or run-time parameter, and make a full name from the property of interest and the selected depth slice:

```
depth.gnatsgo <- depth.list.gnatsgo[params$depth.n]
(voi.depth.name <- paste0(voi.name, "_", depth.gnatsgo))  # , "cm"
```

```
[1] "ph1to1h2o_r_05"
```

Area of Interest (AOI)

Specify the lower-right corner from the YAML or rendering parameters:

```
# lower-right corner
(tile.lrc <- c(params$lrc_long, params$lrc_lat))
```

```
[1] -76  42
```

Compute the upper-left corner:

```
# Tile size, in integer degrees
size.long <- params$size; size.lat <- params$size
tile.ulc <- c(tile.lrc[1]-size.long, tile.lrc[2]+size.lat) # upper-left corner
m <- matrix(c(tile.ulc[1],tile.lrc[1],  #ulc
              tile.ulc[2], tile.lrc[2]  #lrc
              ),
            nrow=2)
bb.ll <- st_sfc(st_multipoint(m))
st_crs(bb.ll) <- 4326
print(bb.ll)
```

```
Geometry set for 1 feature
Geometry type: MULTIPOINT
Dimension:     XY
Bounding box:  xmin: -77 ymin: 42 xmax: -76 ymax: 43
Geodetic CRS:  WGS 84
```

AOI in form needed for gNATSGO WCS import:

```
wcs.aoi <- list(
  aoi = c(tile.ulc[1],tile.lrc[2], tile.lrc[1], tile.ulc[2]),
  crs = '+init=EPSG:4326')
```

A prefix for directories, to keep AOI results separate.

```
AOI.dir.prefix <- paste0("lat", tile.lrc[2], tile.ulc[2],
                         "_lon", tile.ulc[1], tile.lrc[1])
```

A directory to store the map unit tile and its linked database on import:

```
(dest.dir.gnatsgo.import <-  paste0(base.dir.gnatsgo.import,
                           AOI.dir.prefix))
```

```
[1] "/Users/rossiter/tmp/Compare_DSM/gNATSGO/lat4243_lon-77-76"
```

```
if (!dir.exists(dest.dir.gnatsgo.import)) {
   dir.create(dest.dir.gnatsgo.import, recursive = TRUE)
}
```

A directory to save the processed tile:

```
base.dir.gnatsgo.import
```

```
[1] "/Users/rossiter/tmp/Compare_DSM/gNATSGO/"
```

```
(dest.dir <-  paste0(base.dir.gnatsgo, AOI.dir.prefix))
```

```
[1] "/Users/rossiter/ds_reference/Compare_DSM/gNATSGO/lat4243_lon-77-76"
```

```
if (!dir.exists(dest.dir)) {
   dir.create(dest.dir, recursive = TRUE)
}
```

Grid resolution

```
(grid.res <- params$res)
```

```
[1] 250
```

WCS access

The 30~m map unit raster takes about 16Mb per tile. The default EPSG code is 6350. This CRS is an Albers Equal Area with parameters suitable for the CONUS.

Do not repeat the WCS call if we already have the tile; if you want to make sure to have the most recent, delete any stored tile before calling.

```
spc.name <- "mukey"
(spc.file <-  paste0(dest.dir.gnatsgo.import, "/", spc.name, "_", grid.res, ".tif"))
```

```
[1] "/Users/rossiter/tmp/Compare_DSM/gNATSGO/lat4243_lon-77-76/mukey_250.tif"
```

```
if (file.exists(spc.file)) {
  gn.m <- rast(spc.file)
} else {
  system.time(
    gn.m <- soilDB::mukey.wcs(db = 'gnatsgo',
                              aoi = wcs.aoi, res = grid.res) # crs = "EPSG:6350"
  )
  names(gn.m) <- "mukey"
  class(gn.m)
#  gn.m <- terra::rast(gn.m)
  terra::writeRaster(gn.m, spc.file, overwrite = TRUE)  # extension  is .tif
}
st_crs(gn.m)$proj4string
```

```
[1] "+proj=aea +lat_0=23 +lon_0=-96 +lat_1=29.5 +lat_2=45.5 +x_0=0 +y_0=0 +datum=NAD83 +units=m +no_defs
```

```
st_bbox(gn.m)
```

```
   xmin    ymin    xmax    ymax
1530465 2266635 1633215 2392635
```

```
rasterVis::levelplot(gn.m, att = 'ID', margin = FALSE,
                     colorkey = FALSE, ask=FALSE)
```

The colours are from the map unit ID, they have no other meaning.

Map unit IDs:

```
mu.list <- levels(gn.m)[[1]]
dim(mu.list)
```

```
[1] 1509    2
```

```
head(mu.list)
```

```
      ID  mukey
1 289163 289163
2 289164 289164
3 289165 289165
4 289166 289166
5 289167 289167
6 289168 289168
```

There are 1509 unique map unit IDs in this window. This is the basis of the RAT for eventual map reclassification; we will add the attribute values as a second field.

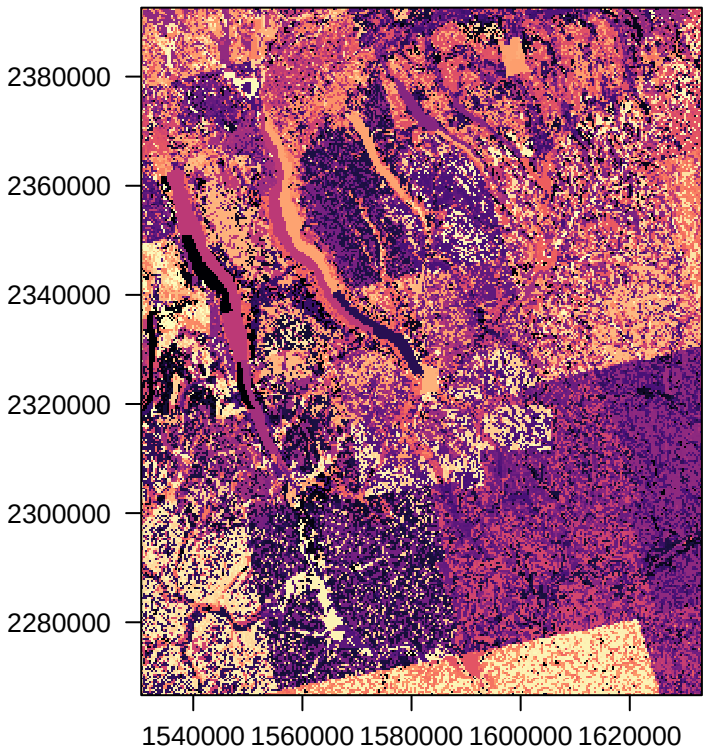


Figure 1: Map units at 30m

Resample to WGS84 geographic coordinates

For comparison with other products, we want this in WGS84 geographic coordinates.

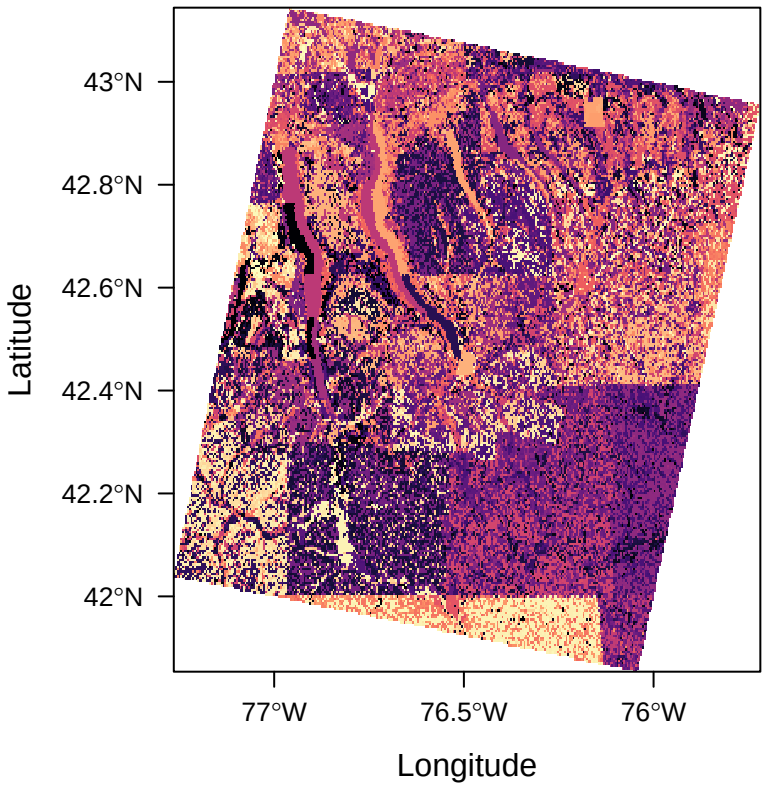
```
gn.84 <- terra::project(gn.m, "epsg:4326", method = "near",
                                align = TRUE)
print(gn.84)
```

```
class       : SpatRaster
dimensions  : 530, 635, 1  (nrow, ncol, nlyr)
resolution  : 0.002434898, 0.002434898  (x, y)
extent      : -77.26467, -75.71851, 41.85364, 43.14414  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source(s)   : memory
categories  : mukey
name        :   mukey
min value   :  289163
max value   : 3264877
```

```
st_bbox(gn.84)
```

```
     xmin      ymin      xmax      ymax
-77.26467  41.85364 -75.71851  43.14414
```

```
rasterVis::levelplot(gn.84, att = 'ID', margin = FALSE,
                     colorkey = FALSE, ask=FALSE)
```

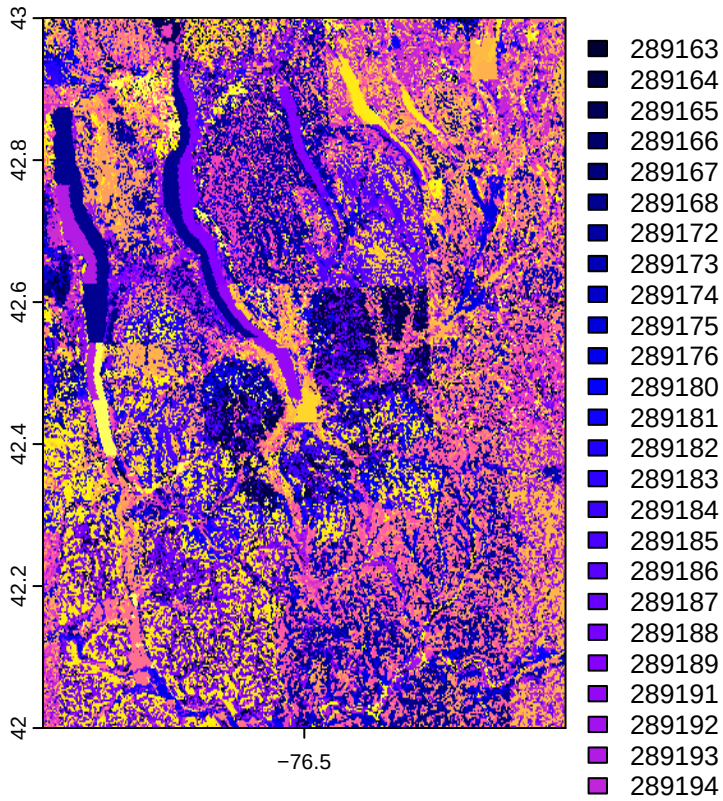



```
st_bbox(gn.84)
```

```
     xmin      ymin      xmax      ymax
-77.26467  41.85364 -75.71851  43.14414
```

Note that the bounding box larger than the originally requested box, so that the original tile can be all represented in geographic coordinates. So, crop to the original tile specification.

```
gn.84.crop <- crop(gn.84, bb.ll)
terra::plot(gn.84.crop, col=(sp::bpy.colors(50)))
```



Attributes database

The Soil Data Access (SDA) web service has the information for each map unit. SDA from R is explained in this tutorial.

We have the map unit key, so get their information.

Query SDA by `mukey` for the map units in this tile.

This will bring down most of the interesting site / horizon level attributes from SSURGO/STATSGO, including the variable of interest.

Do not repeat the `fetchSDA` call if we already have the attributes for this tile; if you want to make sure to have the most recent, delete the stored `.rds` file before calling.

```
spc.name <- "muinfo"
(spc.file <-  paste0(dest.dir.gnatsgo.import, "/", spc.name, "_", grid.res, ".rds"))
```

```
[1] "/Users/rossiter/tmp/Compare_DSM/gNATSGO/lat4243_lon-77-76/muinfo_250.rds"
```

```
if (file.exists(spc.file)) {
  mu.info <- readRDS(spc.file)
} else {
  # Format vector of values into a string suitable for an SQL `IN` statement
  IS <- soilDB::format_SQL_in_statement(mu.list$ID)
  # query string -- all components
  ws <- sprintf("mukey IN %s", IS)
  system.time(
    mu.info <- suppressMessages(
      soilDB::fetchSDA(WHERE = ws, duplicates = TRUE,
                       droplevels = TRUE, stringsAsFactors = FALSE,
```

```
                        childs = FALSE)
    )
  )
  saveRDS(mu.info, spc.file)
}
class(mu.info)
```

```
[1] "SoilProfileCollection"
attr(,"package")
[1] "aqp"
```

```
head(mu.info)
```

```
SoilProfileCollection with 6 profiles and 12 horizons
profile ID: cokey  |  horizon ID: hzID
Depth range: 15 - 152 cm

----- Horizons (6 / 12 rows  |  10 / 63 columns) -----
    cokey hzID hzdept_r hzdepb_r hzname      texture    chkey           texcl
 14218282    1        0       23     H1          fsl 40905279 Fine sandy loam
 14218283    2        0       15     H1        grx-s 40905283            Sand
 14218284    3        0       46     H1            l 40905285            Loam
 14218284    4       76      152     H3 sr- grv-s sl 40905287      Sandy loam
 14218284    5       76      152     H3 sr- grv-s sl 40905287            Sand
 14218285    6        0       18     H1         gr-l 40905288            Loam
 lieutex fragvol_l
    <NA>         0
    <NA>        32
    <NA>         0
    <NA>         8
    <NA>         8
    <NA>        11
[... more horizons ...]

----- Sites (6 / 6 rows  |  10 / 61 columns) -----
  mukey nationalmusym    cokey compname comppct_r           compkind
 666285          qcb2 14218282  Arkport         3             Series
 666285          qcb2 14218283     Pits         2 Miscellaneous area
 666285          qcb2 14218284   Halsey         1             Series
 666285          qcb2 14218285   Fredon         2             Series
 666285          qcb2 14218286    Alton        10             Series
 666285          qcb2 14218287   Phelps         3             Series
 majcompflag localphase              drainagecl hydricrating
        No        <NA>            Well drained           No
        No        <NA>                    <NA>           No
        No        <NA>     Very poorly drained          Yes
        No        <NA> Somewhat poorly drained           No
        No        <NA>            Well drained           No
        No        <NA> Moderately well drained           No

Spatial Data:
[EMPTY]
```

Link to attribute of interest

Aggregate at component level for variable and depth interval of interest. For this we use the `aqp::slab()` function, "Aggregate soil properties along user-defined 'slabs', and optionally within groups".

Set up the depths and formula and then call the function:

```
# the SoilGrids depths have the correct format
(slab.depths <- as.numeric(strsplit(depth.list.sg[params$depth.n],"-")[[1]]))
```

```
[1] 0 5
```

```
slab.fm <- formula(paste0("cokey ~ ", voi.name))
mu.attr <- aqp::slab(mu.info, slab.fm,
            slab.structure = c(slab.depths[1], slab.depths[2]),
            slab.fun = mean, na.rm = TRUE)
str(mu.attr)
```

```
'data.frame':   4051 obs. of  6 variables:
 $ variable             : Factor w/ 1 level "ph1to1h2o_r": 1 1 1 1 1 1 1 1 1 1 ...
 $ cokey                : chr  "14218282" "14218283" "14218284" "14218285" ...
 $ value                : num  5.9 NaN 6.5 6.5 5 6.5 6.2 6.5 6.2 6.5 ...
 $ contributing_fraction: num  1 0 1 1 1 1 1 1 1 1 ...
 $ top                  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ bottom               : int  5 5 5 5 5 5 5 5 5 5 ...
```

```
warnings()[1]
```

```
NULL
```

This is a list of components, each with its attribute value for the depth slice.

For some tiles, some of the map units have incorrect horizonation.

Make an ID field for reshaping; this is the same for all components:

```
mu.attr$variable.id <- sprintf("%s%s%s%s",
                               mu.attr$variable, "_",
                               mu.attr$top,
                               mu.attr$bottom)
```

Long → wide format as a dataframe with two columns: the component key and the attribute value in the depth slice.

```
mu.attr.w <- reshape2::dcast(mu.attr, cokey ~ variable.id, value.var = 'value')
str(mu.attr.w)
```

```
'data.frame':   4051 obs. of  2 variables:
 $ cokey         : chr  "14218282" "14218283" "14218284" "14218285" ...
 $ ph1to1h2o_r_05: num  5.9 NaN 6.5 6.5 5 6.5 6.2 6.5 6.2 6.5 ...
```

Get the components of each map unit from the site information, via `aqp::site`, and then add the map unit key and proportions to the data frame:

```
mu.site <- aqp::site(mu.info)[, c('mukey', 'cokey', 'comppct_r')]
mu.site <- base::merge(mu.site, mu.attr.w, by = 'cokey', sort = FALSE)
```

So now we have the map unit, its components, their percentages of the map unit, and each component's attribute value averaged over the depth slice.

Split this into separate data frames for each map unit:

```
mu.site.split <- base::split(mu.site, as.factor(mu.site$mukey),
                             lex.order = TRUE)
```

Note that the list of data frames is in lexical order, i.e., the map unit code.

Look at the composition of the first map unit:

```
(tmp <- mu.site.split[[1]])
```

```
        cokey  mukey comppct_r ph1to1h2o_r_05
2392 24272115 289163        40            5.4
2393 24272116 289163         5            6.0
2394 24272117 289163        45            6.2
2395 24272118 289163        10            5.4
```

```
sum(tmp$comppct_r)
```

```
[1] 100
```

This has `dim(tmp)[1]` components; their proportion adds to `round(sum(tmp$comppct_r),1`%, as we expect (or hope).

Now we have two ways to get properties from the map unit: weighted proportion or dominant component.

Functions

Weight the property value by the component proportions

Define a function to weight the property by the component proportions.

Arguments:

- `i`: map unit sequence in `mu.site.split` – this will be called for all of them
- `var.name`: the name of variable to weighted
- `wt.name`: the name of the field containing the component proportions

Implicit argument (in scope):

- `mu.site.split`: a separate data frame for each site

```
wt.mean.component <- function(i = 1, var.name, wt.name = 'comppct_r') {
  # make a local copy of this map unit's information
  mu.info.one <- mu.site.split[[i]]

  # get map unit ID, the list of component values and their weights
  mu.id <- as.character(mu.info.one[1, "mukey"])
  vals <- mu.info.one[,var.name]
  wts <- mu.info.one[,wt.name]

  # remove any list entries with NA in the values list or component proportions
  idx <- which(is.na(vals) | is.na(wts))
  if(length(idx) > 0) { mu.info.one <- mu.info.one[-idx, ] }

  # rebuild values and weights list w/o the components with missing values
  vals <- mu.info.one[,var.name]
  wts <- mu.info.one[,wt.name]

  # weighted mean -- note wts should sum to 100 but we don't assume that, because of possibl NA's
  mean.w <- sum(vals * wts) / sum(wts)
```

```
    # pack results into a one-line data frame
  result <- data.frame(
    mukey = mu.id,
    var = mean.w,
    stringsAsFactors = FALSE
  )
  # name  the variable field with the variable name.
  names(result)[2] <- var.name
  return(result)
}
```

Reclassify raster map

Call the weight function for each map unit and add the result to the data frame of map unit IDs. We have to match the map unit ID of the result with that of the map unit list in the RAT.

```
result.field <- "mean.val.aggr"
mu.list[ , result.field] <- as.numeric(NA)
for (i in 1:length(mu.site.split)) {
  mu.id <- as.character(mu.site.split[[i]][1, "mukey"])
  mean.wt <- wt.mean.component(i, voi.depth.name, "comppct_r")[, voi.depth.name]
  ix <- which(mu.list$ID == mu.id)
  mu.list[ix, result.field] <- mean.wt
}
mu.list$mukey <- as.numeric(mu.list$mukey)
mu.list <- mu.list[, 2:3]
head(mu.list)
```

```
   mukey mean.val.aggr
1 289163         5.790
2 289164         5.900
3 289165         5.425
4 289166         5.300
5 289167         5.300
6 289168         5.300
```

This is now a `data.frame` that can be used as a RAT (Raster Attribute Table).

Match each grid cell map unit ID with its value, using the RAT:

```
r.attr <- classify(gn.84.crop, mu.list)
summary(r.attr)
```

```
     mukey
 Min.   :4.600
 1st Qu.:5.675
 Median :6.012
 Mean   :6.026
 3rd Qu.:6.360
 Max.   :7.620
 NA's   :6080
```

```
class(r.attr)
```

```
[1] "SpatRaster"
attr(,"package")
```

```
[1] "terra"
```

Rename the attribute:

```
names(r.attr)
```

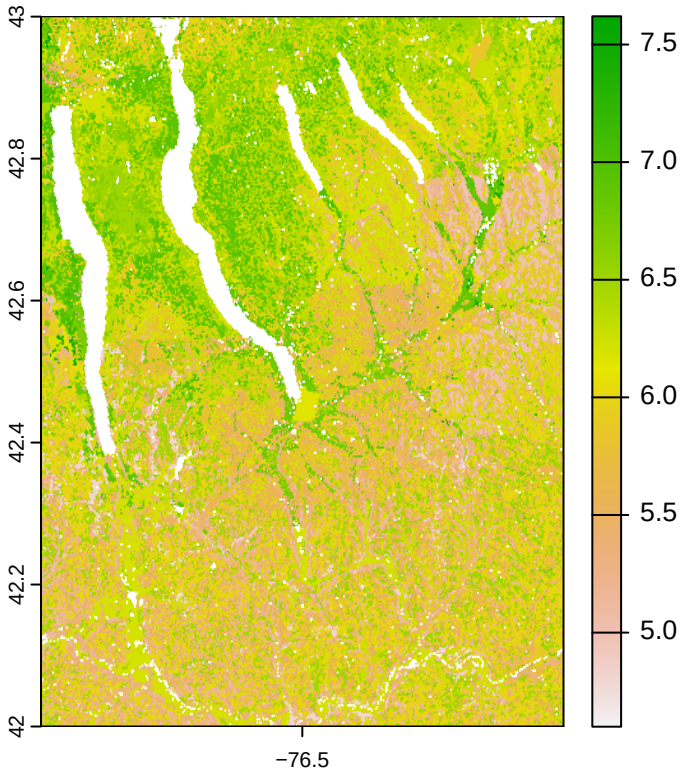
```
[1] "mukey"
```

```
names(r.attr) <- voi.name
names(r.attr)
```

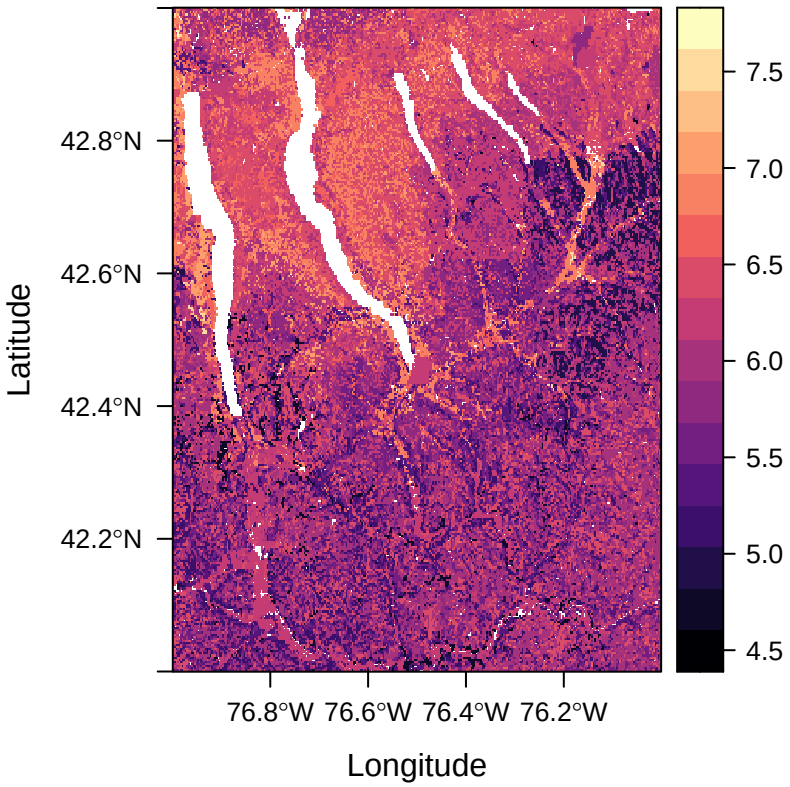
```
[1] "ph1to1h2o_r"
```

Let's see how this looks:

```
plot(r.attr)
```




```
rasterVis::levelplot(r.attr, layers = 1, margin = FALSE, colorkey = TRUE, ask=FALSE)
```



Save tile

Save this map for further processing, e.g., comparing with SoilGrids250 or other DSM products.

Save the tile. Note that the file name includes the property name and depth slice. Specify the float-4 bit datatype and a GeoTIFF "world" file. Each tile is about 14 Mb at 30~m resolution and 300 Kb at 250~m resolution.

```
print(paste("file name:",
            (fn <- paste0(dest.dir, "/", voi.depth.name , "_", grid.res, ".tif"))))
```

```
[1] "file name: /Users/rossiter/ds_reference/Compare_DSM/gNATSGO/lat4243_lon-77-76/ph1to1h2o_r_05_250.t
```

```
f <- terra::writeRaster(r.attr, filename=fn,
                        overwrite=TRUE, datatype="FLT4S",
                        gdal ="TFW=YES")
print(f)
```

```
class       : SpatRaster
dimensions  : 411, 410, 1  (nrow, ncol, nlyr)
resolution  : 0.002434898, 0.002434898  (x, y)
extent      : -76.99926, -76.00095, 41.99973, 43.00048  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : ph1to1h2o_r_05_250.tif
name        : ph1to1h2o_r
min value   :        4.60
max value   :        7.62
```