SoilGrids250 — Import via WCS for DSM comparisons

D G Rossiter david.rossiter@wur.nl

08-January-2024

Contents

Introduction

SoilGrids is a system for global digital soil mapping that makes use of global soil profile information and covariate data to model the spatial distribution of soil properties across the globe. SoilGrids250 is a collections of soil property maps at six standard depths at 250 m grid resolution.

SoilGrids250 filenames, procedures etc. are explained in a FAQ. The choice of the Goode Homolosine projection is explained in Moreira de Sousa, L., L. Poggio, and B. Kempen. 2019. Comparison of FOSS4G Supported Equal-Area Projections Using Discrete Distortion Indicatrices. ISPRS International Journal of Geo-Information 8(8): 351. https://doi.org/10.3390/ijgi8080351.

Download of SoilGrids layers is most convenient by Web Coverage Service (WCS), see this description of procedures from ISRIC.

This script creates a tile for a property and depth slice, over a Area of Interest delimited by geographic coordinates. Tiles are created in EPSG 4326 (WGS84 long/lat) with SoilGrid's nominal 250 m grid resolution.

This tile can then be compared with other PSM products.

To use this script:

1. Adjust the directory structure to your system

Steps 2–3 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("SoilGrids250_WCS_import.Rmd", output_format = "html_document",
                  params = "ask")
```

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

```
rmarkdown::render("SoilGrids250_WCS_import.Rmd", output_format = "html_document",
                    params = list(lrc_long = -76, lrc_lat = 42,
                                  size = 1, quantile.n = 2, voi.n = 3))
```

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

2. Select a property and quantile 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. Either compile to HTML or PDF ("knit"), or "Run All" within R Markdown, or call `rmarkdown::render` from the R command line, as explained above.
5. The processed tile will be in the directory structure, in a subdirectory named for the AOI.

Directories

Set base directories, specific to the local file system.

1. `base.dir.import`: This is where downloaded large GeoTIFF are located. Because of their size they may be on a separate file system, e.g., removable or networked drive.
2. `base.dir`: This is where the processed SoilGrids250 maps are stored.

```
(base.dir.sg <- path.expand("~/ds_reference/Compare_DSM/SoilGrids250/"))
```

```
[1] "/Users/rossiter/ds_reference/Compare_DSM/SoilGrids250/"
```

```
(base.dir.sg.import <- path.expand("~/tmp/Compare_DSM/SoilGrids250/"))
```

```
[1] "/Users/rossiter/tmp/Compare_DSM/SoilGrids250/"
```

These are the base of destination directories built below

Packages

```
library(sf)              # spatial data types
library(terra)           # raster data, replaces `raster`
require(rgdal)           # directly access GDAL functions -- deprecated but still works
require(gdalUtilities)
require(XML)
```

GDAL is used for spatial data import/export, coordinate systems etc. Check the GDAL installation, and whether the WCS service is available.

```
gdal() # version
gdal.drivers <- terra::gdal(drivers = TRUE)
ix <- which(gdal.drivers$name  == "WCS")
gdal.drivers[ix, ]
```

CRS

We want to use geographic coordinates for the tile. But the ISRIC WCS does not seem to serve this – or at least, I can not figure out how. So, we must download SoilGrids250 in the native Homolosine CRS. For this we need to know the bounding box in that CRS.

This CRS with pseudo-EPSG code 152160 should be added to to the `epsg` file of the PROJ database [1] as a final line, as explained here. But for now do not do this, just specify the projection directly.

```
crs.igh <- '+proj=igh +lat_0=0 +lon_0=0 +datum=WGS84 +units=m +no_defs'
```

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$voi.n, "; depth.n:", params$depth.n))
```

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

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

```
[1] "quantile.n: 4"
```

Variable and quantile

Define the variables for the soil property and layer of interest. See here for the naming conventions

`Q0.05` - 5% quantile from the Quantile Random Forest (QRF); `Q0.5` - median of the distribution from the QRF – note *not* `Q0.50`; `Q0.95` - 95% quantile from the QRF; `mean` - mean of the distribution.

```
quantile.list <- c("Q0.05", "Q0.5", "Q0.95", "mean")
```

Here are the properties predicted by SoilGrids250:

```
voi.list.sg <- c("clay", "silt", "sand", "phh2o", "cec", "soc", "bdod", "cfvo", "nitrogen", "ocd")
```

Depth slice

Depth slices: 3

```
depth.list <- paste0(c("0-5", "5-15", "15-30", "30-60", "60-100", "100-200"),"cm")
```

Set the property, depth and quantile from the YAML or rendering parameters:

```
voi <- voi.list.sg[params$voi.n]
depth <- depth.list[params$depth.n]
quantile.sg <- quantile.list[params$quantile.n]
(voi_layer <- paste(voi, depth, quantile.sg, sep="_"))
```

```
[1] "phh2o_0-5cm_mean"
```

Area of Interest (AOI)

The AOI is a tile using WGS84 geographic coordinates. A $1 \times 1°$ allows comparison with POLARIS, but here other sizes can be specified.

[1] for example `/Library/Frameworks/PROJ.framework/Versions/6/Resources/proj/epsg`

Specify the *lower-right corner* and *tile size* from the YAML or rendering parameters:

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

Compute the four corner and the bounding box. Note because of the projection this is somewhat larger than a $1 \times 1°$ tile.

```
tile.ulc <- c(tile.lrc[1]-size, tile.lrc[2]+size)   # lower-right corner
m <- matrix(c(tile.lrc[1]-size, tile.lrc[2]+size,  #ulc
              tile.lrc[1], tile.lrc[2]+size,  #urc
              tile.lrc[1], tile.lrc[2],       #lrc
              tile.lrc[1]-size, tile.lrc[2]  #lcc
              ),
            nrow=4, byrow = TRUE)
m <- rbind(m, m[1,]) # close the polygon
bb.ll <- st_sfc(st_polygon(list(m)))
st_crs(bb.ll) <- 4326
print(bb.ll)
```

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

```
st_boundary(bb.ll)
```

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

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]))
```

```
[1] "lat4243_lon-77-76"
```

Destination directories

Set destination directories, adding to the base directories the variable of interest, quantile, depth. Make sure the directory exists.

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

```
[1] "/Users/rossiter/tmp/Compare_DSM/SoilGrids250/lat4243_lon-77-76"
```

```
if (!dir.exists(dest.dir.sg.import)) {
   dir.create(dest.dir.sg.import, recursive = TRUE)
}
(dest.dir.sg <-  paste0(base.dir.sg,
                        AOI.dir.prefix))
```

```
[1] "/Users/rossiter/ds_reference/Compare_DSM/SoilGrids250/lat4243_lon-77-76"
```

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

Convert the long/lat bounding box to the SoilGrids250 projection. We want the extreme values in both X and Y, to ensure we cover the whole tile. If we just use the corners we will cut off some parts at the upper-right an lower-left.

```
(bb.igh <- st_transform(bb.ll, crs.igh))         # reproject the polygon
```

```
Geometry set for 1 feature
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -9234503 ymin: 4674913 xmax: -9132054 ymax: 4785101
Projected CRS: +proj=igh +lat_0=0 +lon_0=0 +datum=WGS84 +units=m +no_defs
```

```
(bb.igh.coords <-  st_coordinates(bb.igh)[,1:2]) # convert to coordinates, we only need 2D
```

```
           X       Y
[1,] -9234503 4785101
[2,] -9152005 4785101
[3,] -9132054 4674913
[4,] -9215383 4674913
[5,] -9234503 4785101
```

```
#  convert to a bounding box, must order these as c(ulx, uly, lrx, lry)
(bb.sg <- as.vector(c(min(bb.igh.coords[,"X"]),
                      max(bb.igh.coords[,"Y"]),
                      max(bb.igh.coords[,"X"]),
                      min(bb.igh.coords[,"Y"]))))
```

```
[1] -9234503  4785101 -9132054  4674913
```

Accessing SoilGrids250 WCS with GDAL

Adapted from instrutions by Luis de Souza (ISRIC), see https://git.wur.nl/isric/soilgrids/soilgrids.notebooks/-/blob/master/markdown/wcs_from_R.md.

```
wcs_request <- "DescribeCoverage"
wcs_path <- paste0("https://maps.isric.org/mapserv?map=/map/",voi,".map") # Path to the WCS. See maps.i
wcs_service = "SERVICE=WCS"
wcs_version = "VERSION=2.0.1"
(wcs <- paste(wcs_path, wcs_service, wcs_version, wcs_request, sep="&"))
```

```
[1] "https://maps.isric.org/mapserv?map=/map/phh2o.map&SERVICE=WCS&VERSION=2.0.1&DescribeCoverage"
```

```
l1 <- newXMLNode("WCS_GDAL")
l1.s <- newXMLNode("ServiceURL", wcs, parent=l1)
l1.l <- newXMLNode("CoverageName", voi_layer, parent=l1)
# Save to local disk
xml.out <- "./sg.xml"
saveXML(l1, file = xml.out)
```

```
[1] "./sg.xml"
```

```
gdalinfo("./sg.xml")
```

```
Driver: WCS/OGC Web Coverage Service
```

```
Files: ./sg.xml
Size is 159246, 58034
Coordinate System is:
PROJCRS["Interrupted_Goode_Homolosine",
    BASEGEOGCRS["WGS 84",
        DATUM["World Geodetic System 1984",
            ELLIPSOID["WGS 84",6378137,298.257223563,
                LENGTHUNIT["metre",1]],
            ID["EPSG",6326]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["Degree",0.0174532925199433]]],
    CONVERSION["unnamed",
        METHOD["Interrupted Goode Homolosine"]],
    CS[Cartesian,2],
        AXIS["easting",east,
            ORDER[1],
            LENGTHUNIT["metre",1,
                ID["EPSG",9001]]],
        AXIS["northing",north,
            ORDER[2],
            LENGTHUNIT["metre",1,
                ID["EPSG",9001]]]]
Data axis to CRS axis mapping: 1,2
Origin = (-19949875.000000000000000,8361125.000000000000000)
Pixel Size = (250.000000000000000,-250.000000000000000)
Corner Coordinates:
Upper Left  (-19949875.000, 8361125.000)
Lower Left  (-19949875.000,-6147375.000)
Upper Right (19861625.000, 8361125.000)
Lower Right (19861625.000,-6147375.000)
Center      (  -44125.000, 1106875.000) (  0d51'35.71"W,  9d56'35.62"N)
Band 1 Block=1024x512 Type=Int16, ColorInterp=Undefined
  Overviews: 79623x29017, 39811x14508, 19905x7254, 9952x3627, 4976x1813, 2488x906, 1244x453, 622x226
```

Get the coverage:

```
bb.sg
```

```
[1] -9234503  4785101 -9132054  4674913
```

```
crs.igh
```

```
[1] "+proj=igh +lat_0=0 +lon_0=0 +datum=WGS84 +units=m +no_defs"
```

```
(wcs <- paste(wcs_path,wcs_service,wcs_version,sep="&"))
```

```
[1] "https://maps.isric.org/mapserv?map=/map/phh2o.map&SERVICE=WCS&VERSION=2.0.1"
```

```
l1 <- newXMLNode("WCS_GDAL")
l1.s <- newXMLNode("ServiceURL", wcs, parent=l1)
l1.l <- newXMLNode("CoverageName", voi_layer, parent=l1)
xml.out = "./sg.xml"
saveXML(l1, file = xml.out)
```

```
[1] "./sg.xml"
```

```
(file.out <- paste0(dest.dir.sg.import, "/", voi_layer, '.tif'))
```

```
[1] "/Users/rossiter/tmp/Compare_DSM/SoilGrids250/lat4243_lon-77-76/phh2o_0-5cm_mean.tif"
```

```
gdal_translate(xml.out, file.out,
    tr=c(250,250),
    projwin = bb.sg, projwin_srs = crs.igh, # corners in this CRS
    co=c("TILED=YES","COMPRESS=DEFLATE","PREDICTOR=2","BIGTIFF=YES")
    )
```

Check the result by reading into R, summarizing, and plotting.

```
r.sg <- terra::rast(file.out)
print(r.sg)
```

```
class       : SpatRaster
dimensions  : 441, 410, 1  (nrow, ncol, nlyr)
resolution  : 250, 250  (x, y)
extent      : -9234625, -9132125, 4674875, 4785125  (xmin, xmax, ymin, ymax)
coord. ref. : Interrupted_Goode_Homolosine
source      : phh2o_0-5cm_mean.tif
name        : phh2o_0-5cm_mean
```

```
summary(values(r.sg))
```

```
 phh2o_0-5cm_mean
 Min.   : 0.00
 1st Qu.:52.00
 Median :55.00
 Mean   :52.74
 3rd Qu.:60.00
 Max.   :73.00
```

```
terra::plot(r.sg, col=(sp::bpy.colors(50)))
```

The 0 values in SoilGrids are in fact un-mapped areas, so convert these to `NA`.

```
quantile(values(r.sg), seq(0,1,by=0.05))
```

```
  0%   5%  10%  15%  20%  25%  30%  35%  40%  45%  50%  55%  60%  65%  70%  75%
   0    0   47   49   51   52   53   53   54   55   55   56   57   58   59   60
 80%  85%  90%  95% 100%
  61   62   63   65   73
```

```
values(r.sg) <- ifelse(values(r.sg) < 1, NA, values(r.sg))
quantile(values(r.sg), seq(0,1,by=0.05), na.rm = TRUE)
```

```
  0%   5%  10%  15%  20%  25%  30%  35%  40%  45%  50%  55%  60%  65%  70%  75%
  43   47   50   51   52   52   53   54   54   55   56   56   57   58   59   60
 80%  85%  90%  95% 100%
  61   62   63   65   73
```

```
terra::plot(r.sg, col=(sp::bpy.colors(50)))
```

Resample to WGS84 geographic coordinates

Set the CRS and resample to WGS84 geographic coordinates. Use bilinear interpolation – notice that the values are now fractional, rather than integers as in the original SoilGrids layer.

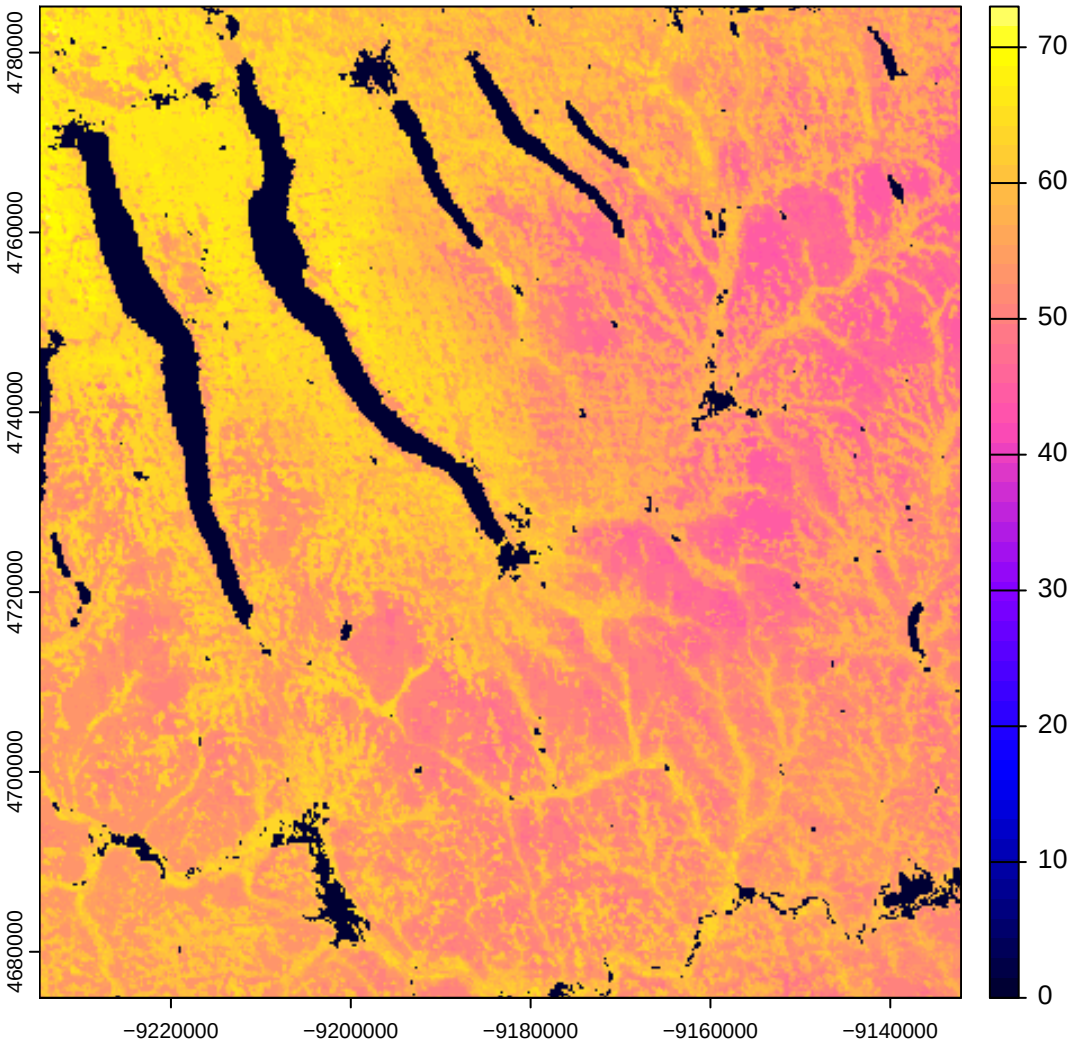

Figure 1: SoilGrids IGH tile

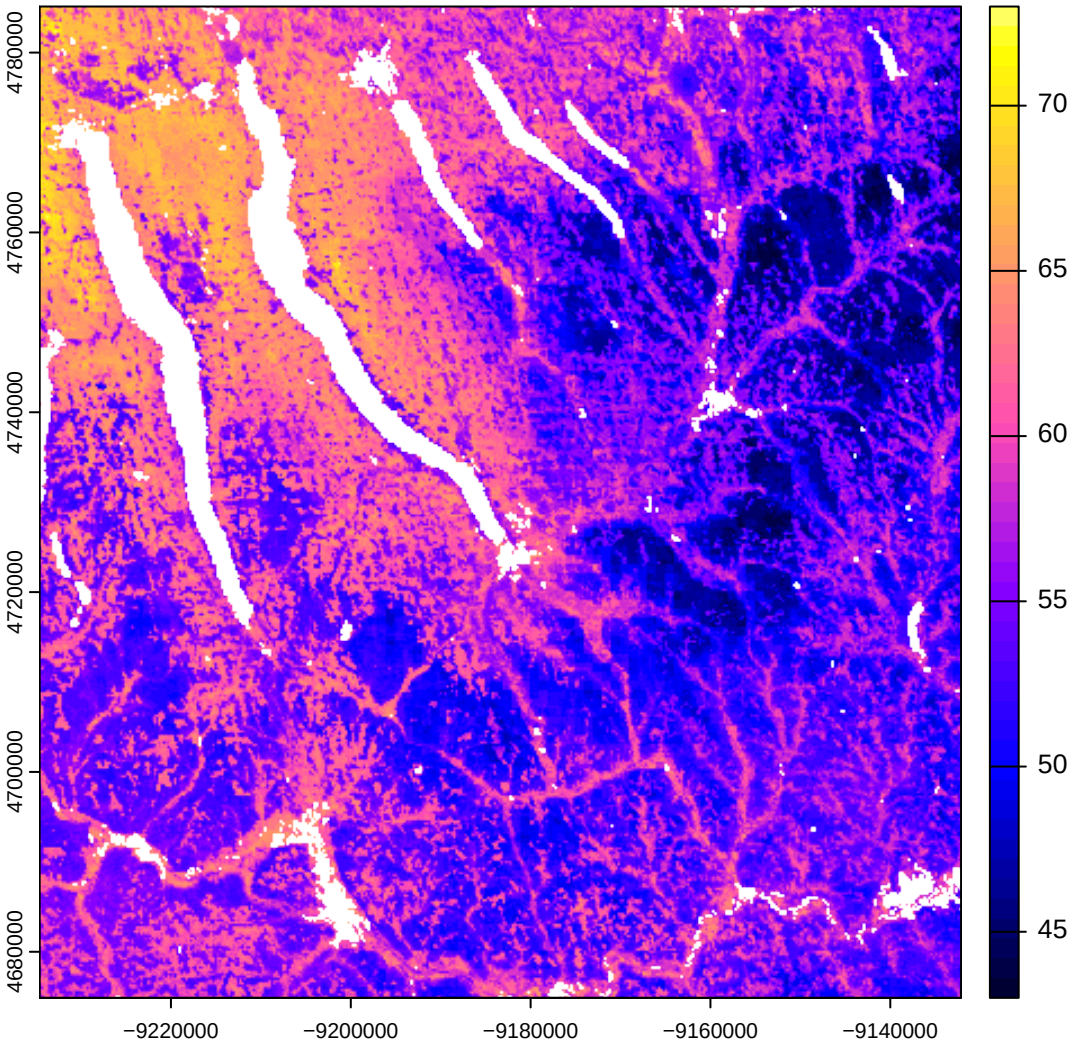

Figure 2: SoilGrids IGH tile, with NA

```
st_bbox(r.sg)
```

```
    xmin     ymin     xmax     ymax
-9234625  4674875 -9132125  4785125
```

```
st_crs(r.sg)$proj4string
```

```
[1] "+proj=igh +lon_0=0 +x_0=0 +y_0=0 +datum=WGS84 +units=m +no_defs"
```

```
st_crs(r.sg)$epsg       # not in the EPSG database
```

```
[1] NA
```

```
r.sg.84 <- terra::project(r.sg, "epsg:4326")
quantile(values(r.sg.84), seq(0,1,by=.1), na.rm = TRUE)
```

```
      0%      10%      20%      30%      40%      50%      60%      70%
43.11129 49.83549 51.98556 53.32446 54.60259 55.86326 57.14711 58.72944
     80%      90%     100%
60.49945 63.00000 72.64450
```

```
st_crs(r.sg.84)$proj4string
```

```
[1] "+proj=longlat +datum=WGS84 +no_defs"
```

```
st_crs(r.sg.84)$epsg
```

```
[1] 4326
```

Crop this to the long/lat bounding box – recall, the Homolosine bounding box is larger:

```
r.sg.84.crop <- crop(r.sg.84, bb.ll)
quantile(values(r.sg.84.crop), seq(0,1,by=0.05), na.rm = TRUE)
```

```
      0%       5%      10%      15%      20%      25%      30%      35%
43.11129 48.23084 50.10423 51.22021 52.06516 52.87885 53.55260 54.22994
     40%      45%      50%      55%      60%      65%      70%      75%
54.94522 55.58445 56.21183 56.89588 57.58313 58.30275 59.07038 59.93369
     80%      85%      90%      95%     100%
60.82733 61.90358 63.15700 64.94463 71.58004
```

```
terra::plot(r.sg.84.crop, col=(sp::bpy.colors(50)))
```

Save tile

Save this map for further processing, e.g., comparing with 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 750 kB.

```
print(paste("file name:",
      file.out <- paste0(dest.dir.sg, "/", voi_layer, '.tif')))
```

```
[1] "file name: /Users/rossiter/ds_reference/Compare_DSM/SoilGrids250/lat4243_lon-77-76/phh2o_0-5cm_mea
```

```
writeRaster(r.sg.84.crop, file = file.out,
                        overwrite=TRUE, datatype="FLT4S",
                        gdal ="TFW=YES")
```

This can now be imported into analysis scripts.

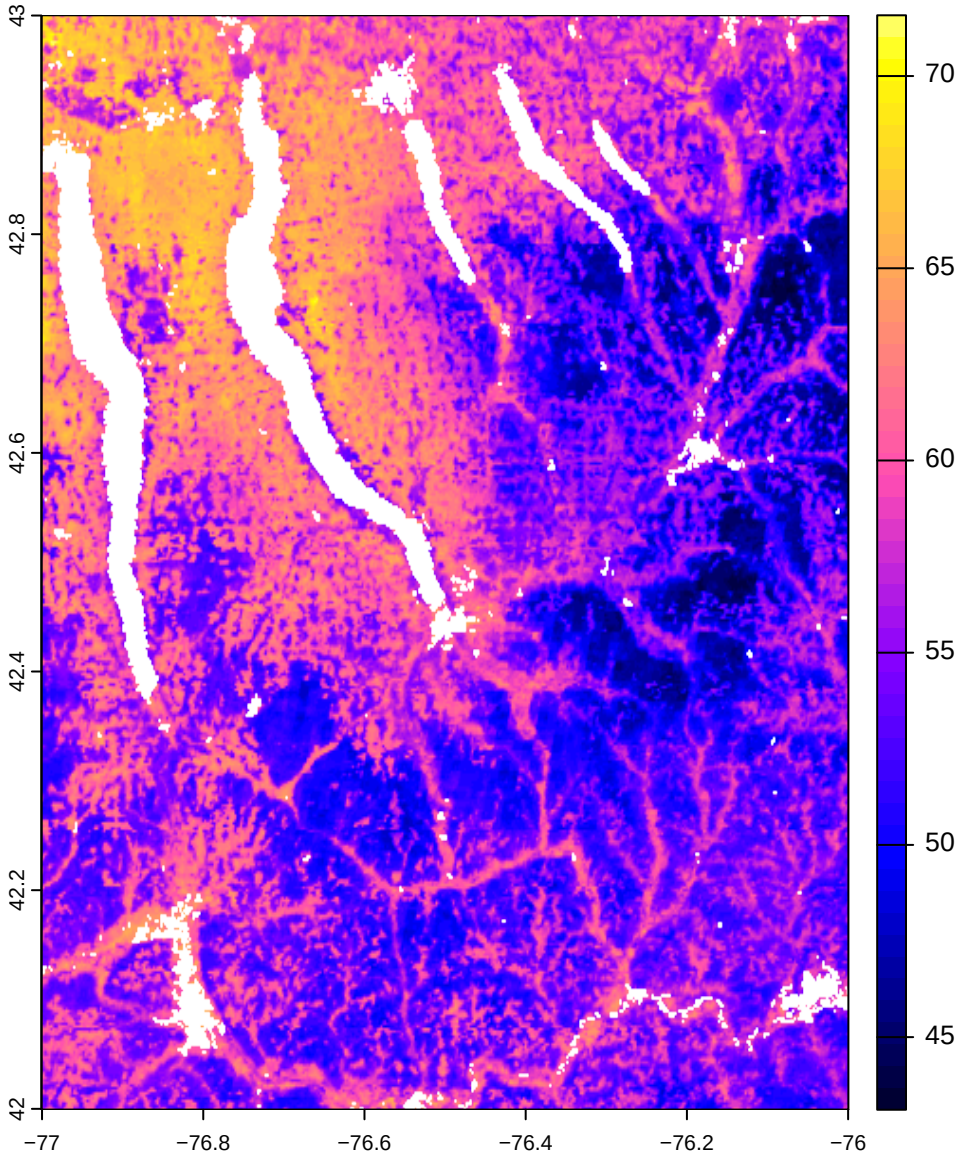

Figure 3: SoilGrids WGS84 tile