Downloading and using SoilGrids

- *Edited by*: T. Hengl
- Part of: **ISRIC Spring School**

SoilGrid (available for download via www.soilgrids.org) contains predictions for a selection of soil properties (soil organic carbon content, soil pH, texture fractions, bulk density, coarse fragments and Cation Exchange Capacity), and soil classes (FAO's World Reference Base soil types and USDA's Soil Taxonomy suborders). This tutorial explains how to obtain SoilGrids GeoTiffs, how to load the data to an R session, plot maps and run some basic analysis.

Download the tutorial as **R script**.

Software in use

We start by loading all necessary packages used in the exercises. To find out how to install R, GDAL and similar, please refer to this tutorial:

```
> library(RCurl)
> library(rgdal)
> library(GSIF)
> library(raster)
> library(plotKML)
> library(XML)
> library(lattice)
> library(aqp)
> library(soiltexture)
> ## GDAL path:
> if(.Platform$OS.type == "windows"){
+   gdal.dir <- shortPathName("C:/Program files/GDAL")
+   gdal_translate <- paste0(gdal.dir, "/gdal_translate.exe")
+   gdalwarp <- paste0(gdal.dir, "/gdalwarp.exe")
+   gdalinfo <- paste0(gdal.dir, "/gdalinfo.exe")
+ } else {
+   gdal_translate = "gdal_translate"
+   gdalwarp = "gdalwarp"
+   gdalinfo = "gdalinfo"
+ }
```

FTP data access

SoilGrids (GeoTiffs) can be obtained either via the web-mapping interface at www.soilgrids.org or via FTP. To download layers from R, we can use the RCurl package:

```
> sg.ftp <- "ftp://ftp.soilgrids.org/data/recent/"
> filenames = getURL(sg.ftp, ftp.use.epsv = TRUE, dirlistonly = TRUE)
> filenames = strsplit(filenames, "\r*\n")[[1]]
> filenames[1:5]
[1] "ACDWRB_M_ss_250m_ll.tif"     "ACDWRB_M_ss_250m_ll.tif.xml"
"AWCh1_M_sl1_250m_ll.tif"
[4] "AWCh1_M_sl1_250m_ll.tif.xml" "AWCh1_M_sl2_250m_ll.tif"
```

These shows that there are >600 files in the `/recent/` directory. To download a single file we can run:

```
> ORC.name <- filenames[grep(filenames,
pattern=glob2rx("ORCDRC_M_sl1_250m_ll.tif$"))]
> ORC.name
```

```
[1] "ORCDRC_M_sl1_250m_ll.tif"
```

```
> try(download.file(paste(sg.ftp, ORC.name, sep=""), ORC.name))
```

```
trying URL 'ftp://ftp.soilgrids.org/data/recent/ORCDRC_M_sl1_250m_ll.tif'
ftp data connection made, file length ...
opened URL
downloaded 2.8 Gb
```

Note that the file is almost 3GB, hence it might take time to download. To check that they layer has been successfully downloaded we can use:

```
> GDALinfo(ORC.name)
```

```
rows        71698
columns     172800
bands       1
lower left origin.x        -180
lower left origin.y        -62.00081
res.x       0.002083333
res.y       0.002083333
ysign       -1
oblique.x
oblique.y
driver      GTiff
projection  +proj=longlat +datum=WGS84 +no_defs
file        ORCDRC_M_sl1_250m_ll.tif
apparent band summary:
  GDType hasNoDataValue NoDataValue blockSize1 blockSize2
```

```
1  Int16           TRUE\      -32768        512        512
apparent band statistics:
    Bmin  Bmax Bmean Bsd
1 -32768 32767    NA\  NA\
Metadata:
AREA_OR_POINT=Area
Warning message:
statistics not supported by this driver
```

Note that the compressed files are still large (from hundreds of MB to 4–5GB), so that download of all layers (ca 400GB) could take significant amount of time (depending on the network speed). Note also that the layer `ORCDRC_M_sl1_250m_ll.tif` (compressed) is almost 3GB in size. The whole stack of SoilGrids would require at least 400GB of hard disk space.

WCS data access

To obtain only a subset of SoilGrids (i.e. values for a single country), probably a better idea of using FTP is to use the Web Coverage Service provided by ISRIC. Most of the SoilGrids layers are available via a Geoserver installation, so that subsets of the maps can be obtained by using the `gdal_translate` command i.e. without a need to download the whole layer. Imagine if we want to obtain values of organic carbon content, bulk density and coarse fragments for Ghana. Obviously there is no need to download these layers from FTP but try to obtain just subsets that represent Ghana.

We start with determining the bounding box for Ghana. For this we use the administrative boundaries for Africa, available via this dokuwiki:

```
> wg.url <-
url("http://gsif.isric.org/lib/exe/fetch.php?media=admin.af.rda")
> load(wg.url)
> proj4string(admin.af) <- "+proj=longlat +datum=WGS84"
> country.af <- as(admin.af, "SpatialLines")
> ghana <- admin.af[admin.af$FORMAL_EN=="Republic of Ghana",]
```

In the last line of the code we have created a `SpatialPolygonDataFrame` that is a subset of a vector map showing country borders for the whole African continent. The bounding box of this object is the bounding box of interest:

```
> ghana@bbox
        min       max
x -3.262509  1.187968
y  4.737128 11.162937
```

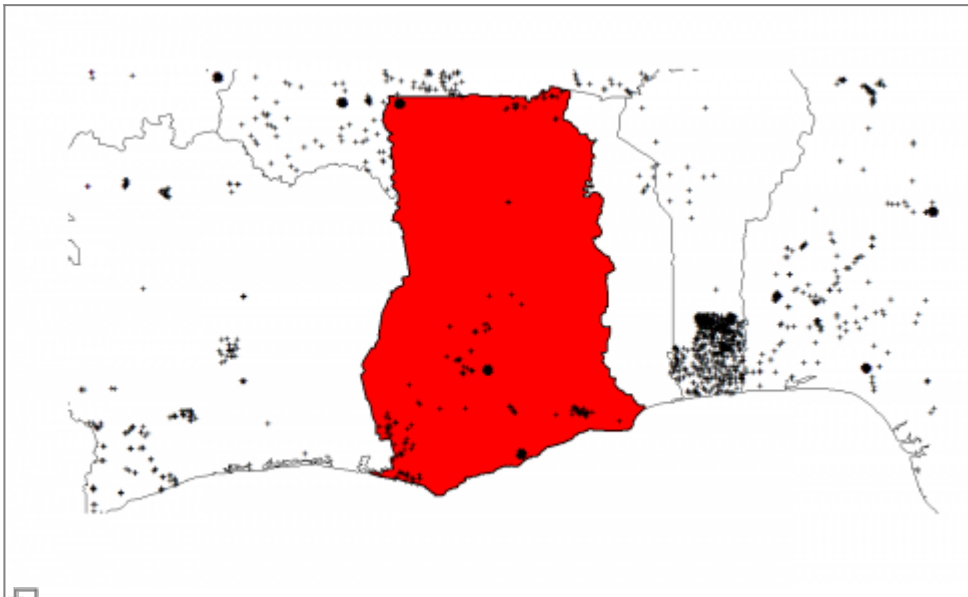
Next, we can also load the soil profile data from the Africa Soil Profile DB available via the GSIF package:

```
> library(GSIF)
> data(afsp)
> sites <- afsp$sites
```

```
> coordinates(sites) <- ~ LONWGS84 + LATWGS84
> proj4string(sites) <- "+proj=longlat +datum=WGS84"
```

so that we can now plot the country of interest and distribution of soil profiles:

```
> plot(ghana, col="red", lwd=2, asp=1)
> lines(country.af)
> points(sites, pch="+")
```



Country borders and distribution of soil profiles (afsp) for Ghana.

To subset Ghana from the whole global coverage GeoTiff, we can use the GDAL utilities software:

```
> te = as.vector(ghana@bbox)
> unlink("ORC_sl1_Ghana.tif")
> system(paste0(gdalwarp, ' ', ORC.name, ' ORC_sl1_Ghana.tif -te ',
paste(te, collapse=" ")))
```

```
0...10...20...30...40...50...60...70...80...90...100 - done.
```

```
> ORCDRC_sl1_ghana <- readGDAL("ORC_sl1_Ghana.tif")
```

```
ORC_sl1_Ghana.tif has GDAL driver GTiff
and has 3084 rows and 2136 columns
```

here the command `gdalwarp` resamples/subsets the original grid to the bounding box of interest i.e. the borders of Ghana. To run this operation (MS Windows users), you will need to install GDAL before you start the R session.

The size of the subset of `ORCDRC_sd1_M_02_apr_2013.tif` for Ghana is 3084 rows and 2136 columns (hence very small subset of the whole world), hence it seems rather inefficient to download the complete coverage GeoTiff and then work on a copy on your machine. A more efficient alternative to access SoilGrids raw data is to use the Web Coverage Service (WCS) driver available via GDAL, that will allow us to access the values for Ghana without a need to download the global GeoTiff. We start by defining the location of the WCS and the target layer. The Web Coverage Service requires that the

URL of the service and the name of the target layer are specified in a local service description xml file:

```
> wcs = "http://webservices.isric.org/geoserver/wcs?"
> l1 <- newXMLNode("WCS_GDAL")
> l1.s <- newXMLNode("ServiceURL", wcs, parent=l1)
> l1.l <- newXMLNode("CoverageName", "orcdrc_m_sl1_250m", parent=l1)
> l1
```

```
<WCS_GDAL>
  <ServiceURL>http://webservices.isric.org/geoserver/wcs?</ServiceURL>
  <CoverageName>orcdrc_m_sl1_250m</CoverageName>
</WCS_GDAL>
```

```
> xml.out = "ORCDRC_M_sl1.xml"
> saveXML(l1, file=xml.out)
```

```
[1] "ORCDRC_M_sl1.xml"
```

For GDAL it is enough it knows where to look and for which layer to look for. If everything is OK, we can test that GDAL can indeed access that layer:

```
> system(paste(gdalinfo, xml.out))
```

```
Driver: WCS/OGC Web Coverage Service
Files: ORCDRC_M_sl1.xml
Size is 172800, 71698
Coordinate System is:
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",],
    UNIT["degree",0.0174532925199433],
    AUTHORITY["EPSG","4326"]]
Origin = (-180.000000000000000,87.370000000000005)
Pixel Size = (0.002083333000000,-0.002083333000000)
Corner Coordinates:
Upper Left  (-180.0000000,  87.3700000) (180d ' 0.00"W, 87d22'12.00"N)
Lower Left  (-180.0000000, -62.0008094) (180d 0' 0.00"W, 62d ' 2.91"S)
Upper Right ( 179.9999424,  87.3700000) (179d59'59.79"E, 87d22'12.00"N)
Lower Right ( 179.9999424, -62.0008094) (179d59'59.79"E, 62d 0' 2.91"S)
Center      (  -0.0000288,  12.6845953) (  0d 0' 0.10"W, 12d41' 4.54"N)
Band 1 Block=1024x512 Type=Int16, ColorInterp=Undefined
  Overviews: 86400x35849, 43200x17924, 21600x8962, 10800x4481, 5400x2240,
2700x1120, 1350x560, 675x280
```

Next we want to obtain only the values of the layer for Ghana at 1 km resolution. First we need to estimate the source window i.e. the rows and columns that we would like to subset from the original

GeoTiff (which has 172800 rows and 71698 columns):

```
> extent(raster(ORC.name))
```

```
class       : Extent
xmin        : -180
xmax        : 179.9999
ymin        : -62.00081
ymax        : 87.37
```

```
> bb <- matrix(nrow=2, c(-180,-62.00081,179.9999,87.37))
> o.x = 172800 + round(172800*(te[1]-bb[1,2])/(bb[1,2]-bb[1,1]))
> o.y = round(71698*(bb[2,2]-te[4])/(bb[2,2]-bb[2,1]))
> d.y = round(71698*(te[4]-te[2])/(bb[2,2]-bb[2,1]))
> d.x = round(172800*(te[3]-te[1])/(bb[1,2]-bb[1,1]))
> o.x; o.y; d.x; d.y
```

```
[1] 84834
[1] 36579
[1] 2136
[1] 3084
```

so that now `gdal_translate` function (depends on GDAL >2.0) can be used to fetch only that subset for Ghana from the server:

```
> system(paste0(gdal_translate, ' ', xml.out, ' ORC_sl1_Ghana.tif -tr ',
1/120, ' ', 1/120, ' -co \"COMPRESS=DEFLATE\" -srcwin ', paste(c(o.x, o.y,
d.x, d.y), collapse=" ")))
```

```
Input file size is 172800, 71698
...10...20...30...40...50...60...70...80...90...100 - done.
```

```
> GDALinfo("ORC_sl1_Ghana.tif")
```

```
rows        771
columns     534
bands       1
lower left origin.x        -3.262528
lower left origin.y        4.738762
res.x       0.008333333
res.y       0.008333333
ysign       -1
oblique.x
oblique.y
driver      GTiff
projection  +proj=longlat +datum=WGS84 +no_defs
file        ORC_sl1_Ghana.tif
apparent band summary:
  GDType hasNoDataValue NoDataValue blockSize1 blockSize2
1  Int16          FALSE\                     7        534
```

```
apparent band statistics:
    Bmin  Bmax Bmean Bsd
1 -32768 32767    NA\  NA\
Metadata:
AREA_OR_POINT=Area
Warning message:
statistics not supported by this driver
```

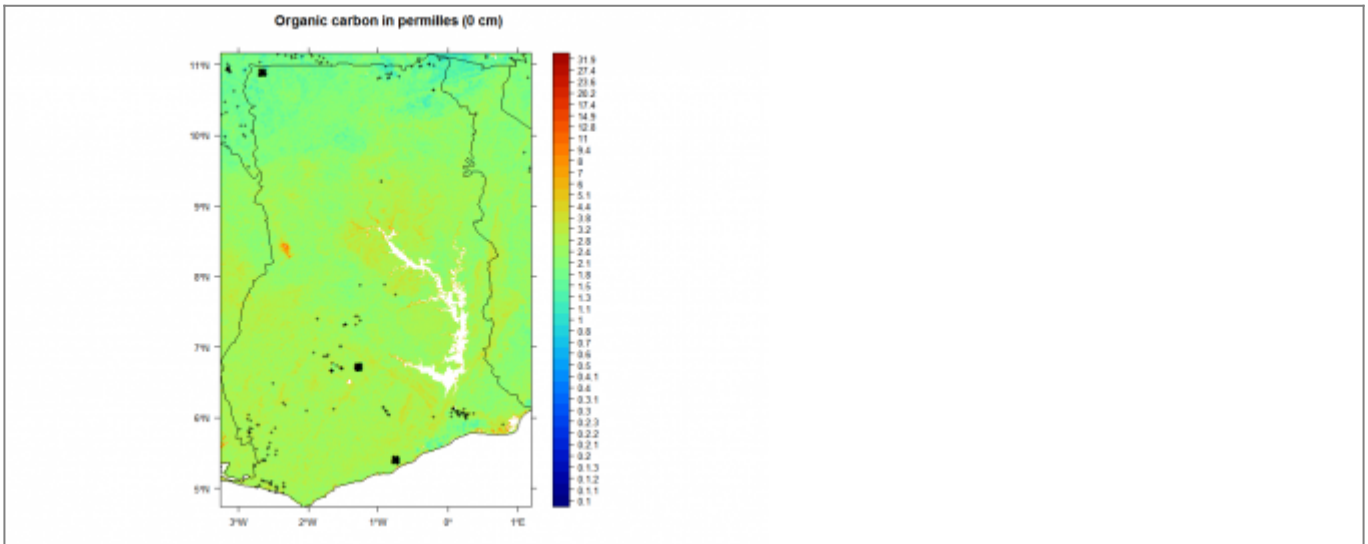
Note that this is now extremely efficient as fast, as we do not have to wait to download the image for whole world. In principle, the smaller the subset you need to obtain from SoilGrids, the more important is to use the SoilGrids WCS.

We can finally plot this map in R using the `spplot` command:

```
> ORCDRC_sl1_ghana <- readGDAL("ORC_sl1_Ghana.tif")
```

```
ORC_sl1_Ghana.tif has GDAL driver GTiff
and has 771 rows and 534 columns
```

```
> data(soil.legends)
> class.labels =
make.unique(as.character(round((soil.legends[["ORCDRC"]]$MAX-
soil.legends[["ORCDRC"]]$MIN)/2, 1)))
> ORCDRC_sl1_ghana$val <- cut(ORCDRC_sl1_ghana$band1,
breaks=c(soil.legends[["ORCDRC"]]$MIN[1], soil.legends[["ORCDRC"]]$MAX),
labels = class.labels)
> bnd <- list(list("sp.points", sites, pch="+", col="black"),
list("sp.lines", country.af))
> spplot(ORCDRC_sl1_ghana["val"],
col.regions=soil.legends[["ORCDRC"]]$COLOR, sp.layout=bnd,
scales=list(draw=TRUE), main="Organic carbon in permilles (0 cm)")
```

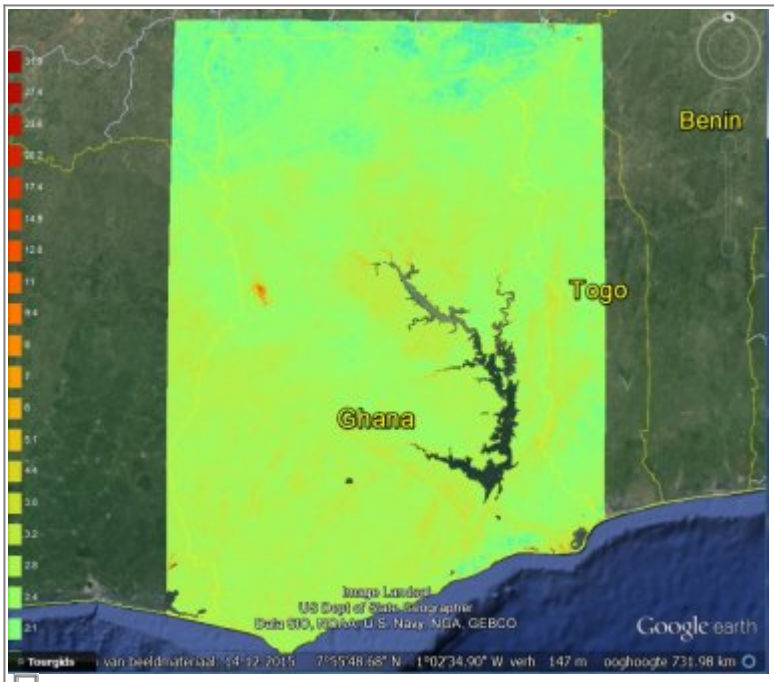


Soil organic carbon content for top soil (0 cm) for Ghana obtained from SoilGrids. Visualized using a global legend that can be loaded from the GSIF package.

or plot in Google Earth using the `plotKML` command:

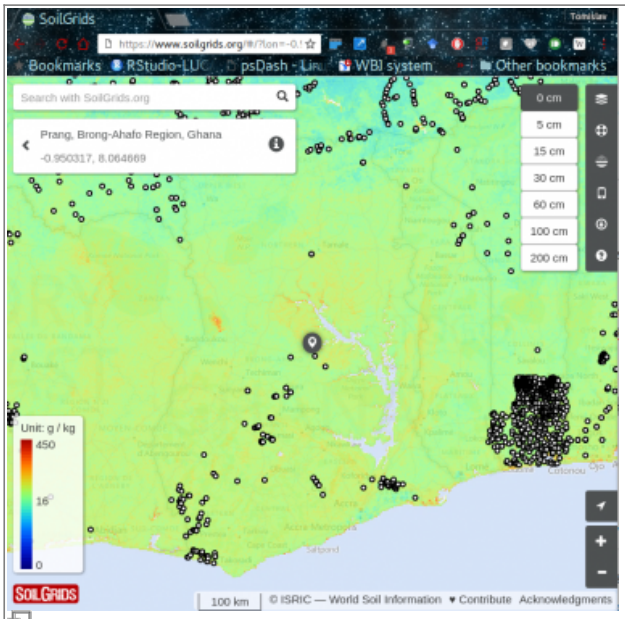
```
> library(plotKML)
> plotKML(ORCDRC_sl1_ghana["val"],
colour_scale=soil.legends[["ORCDRC"]]$COLOR)
```

```
Plotting the first variable on the list
KML file opened for writing...
Loading required package: animation
Writing to KML...
Closing  ORCDRC_sl1_ghana__val__.kml
```



Soil organic carbon content for Ghana visualized in Google Earth.

Note that both plots use standardized global color legends prepared specifically for displaying SoilGrids. The same way we can obtain any subset of SoilGrids and visualize it in R or in Google Earth. If the subset covers >40% of land surface, then it might be a better idea to obtain the whole layer from the FTP, as indicated in the previous section.



Soil organic carbon content for Ghana visualized in SoilGrids.

Deriving Soil Organic Carbon stock using SoilGrids

In the next exercise, we would like to derive total Soil Organic Carbon Stock (SOCS) using SoilGrids layers. We will now run processing in loops to avoid making long chunks of code. First, we need to define the layers of interest i.e. organic carbon content, bulk density and coarse fragments maps:

```
> var.name <- c(paste0("orcdrc_m_sl", c(1,2,3,4), "_250m"),
paste0("bldfie_m_sl", c(1,2,3,4), "_250m"), paste0("crfvol_m_sl",
c(1,2,3,4), "_250m"))
```

and now we can run the operations explained in the previous section in a loop:

```
> for(i in 1:length(var.name)){
+  t1 <- newXMLNode("WCS_GDAL")
+  t1.s <- newXMLNode("ServiceURL", wcs, parent=t1)
+  t1.l <- newXMLNode("CoverageName", var.name[i], parent=t1)
+  xml.out <- paste(var.name[i], ".xml", sep="")
+  saveXML(t1, file=xml.out)
+  f.name <- paste(var.name[i], "_Ghana.tif", sep="")
+  if(!file.exists(f.name)){
+    system(paste0(gdal_translate, ' ', xml.out, ' ', f.name, ' -tr ',
1/120, ' ', 1/120, ' -co \"COMPRESS=DEFLATE\" -srcwin ', paste(c(o.x, o.y,
d.x, d.y), collapse=" ")))
+  }
+ }
```

This will generate XML files and extract all layers of interest using the `gdal_translate` function. Note that this takes only few seconds because the geographical subset is relatively small. Downloading whole GeoTiffs would have taken maybe even hours otherwise.

Next we read all layers of interest in R via the raster package:

```
> ghana1km <- stack(paste(var.name, "_Ghana.tif", sep=""))
> ghana1km <- as(as(ghana1km, "SpatialGridDataFrame"),
"SpatialPixelsDataFrame")
> ## mask out missing pixels:
> sel.NA <- !ghana1km$orcdrc_m_sl1_250m_Ghana <  &
!ghana1km$crfvol_m_sl1_250m_Ghana <
> summary(sel.NA)
```

```
   Mode   FALSE\    TRUE\    NA\'s
logical   43566  368148       0
```

```
> ghana1km <- ghana1km[sel.NA,]
> str(ghana1km@data)
```

```
'data.frame':    368148 obs. of  12 variables:
 $ orcdrc_m_sl1_250m_Ghana: int  25 23 26 21 21 30 33 32 28 29 ...
 $ orcdrc_m_sl2_250m_Ghana: int  12 12 13 12 14 18 19 20 17 16 ...
```

```
 $ orcdrc_m_sl3_250m_Ghana: int  9 9 10 10 12 15 16 17 14 13 ...
 $ orcdrc_m_sl4_250m_Ghana: int  4 4 5 4 6 8 9 10 8 7 ...
 $ bldfie_m_sl1_250m_Ghana: int  1521 1505 1502 1506 1467 1460 1443 1451
1449 1452 ...
 $ bldfie_m_sl2_250m_Ghana: int  1457 1442 1442 1458 1444 1415 1385 1372
1399 1424 ...
 $ bldfie_m_sl3_250m_Ghana: int  1505 1486 1485 1486 1457 1445 1411 1402
1432 1440 ...
 $ bldfie_m_sl4_250m_Ghana: int  1533 1514 1511 1513 1475 1466 1432 1423
1447 1456 ...
 $ crfvol_m_sl1_250m_Ghana: int  11 15 18 14 18 21 21 23 20 20 ...
 $ crfvol_m_sl2_250m_Ghana: int  11 15 19 14 19 21 20 22 21 20 ...
 $ crfvol_m_sl3_250m_Ghana: int  14 18 21 16 21 23 22 24 23 23 ...
 $ crfvol_m_sl4_250m_Ghana: int  17 21 25 18 25 26 24 27 25 25 ...
```

This shows the values of organic carbon (ORCDRC), bulk density of fine earth (BLDFIE) and coarse fragments volumetric (CRFVOL) for Ghana for depths 0–5, 5–15 and 15–30 cm. From these four layers we can derive the total Soil Organic Carbon Stock (0–30 cm) by using the standard formula from the GSIF package. Assuming the organic carbon content in permilles of 50 (5%), bulk density of 1200 kg / per cubic meter, and 12% of coarse fragments for a volume of 30 by 100 by 100 cm, we would get:

```
> OCSKGM(50, 1200, 12, 30)
```

```
[1] 15.84
attr(,"measurementError")
[1] 3.55
attr(,"units")
[1] "kilograms per square-meter"
```

or about 160 tonnes per ha. We can apply the same formula to all SoilGrids and all standard thicknesses / depths by:

```
<code rsplus>
> std <- c(,5,15,30)
> n.lst=c("orcdrc","bldfie","crfvol")
> for(d in c(1,2,3)){
+   ## Upper/lower horizons:
+   U.lst <- paste0(n.lst, "_m_sl", d, "_250m_Ghana")
+   L.lst <- paste0(n.lst, "_m_sl", d+1, "_250m_Ghana")
+   ghana1km$ORC <- rowMeans(ghana1km@data[,sapply(c(U.lst[1],L.lst[1]),
function(x){grep(x, names(ghana1km))})], na.rm = TRUE)
+   ghana1km$BLD <- rowMeans(ghana1km@data[,sapply(c(U.lst[2],L.lst[2]),
function(x){grep(x, names(ghana1km))})], na.rm = TRUE)
+   ghana1km$CRF <- rowMeans(ghana1km@data[,sapply(c(U.lst[3],L.lst[3]),
function(x){grep(x, names(ghana1km))})], na.rm = TRUE)
+   ## Predict organic carbon stock (in tones / ha):
+   ghana1km@data[,paste0("OCS_sd", d)] <-
round(as.vector(OCSKGM(ORCDRC=ghana1km$ORC, BLD=ghana1km$BLD,
CRFVOL=ghana1km$CRF, HSIZE=get("stsize", envir = GSIF.opts)[d]*100,
ORCDRC.sd=8, BLD.sd=120, CRFVOL.sd=10)*10))
```

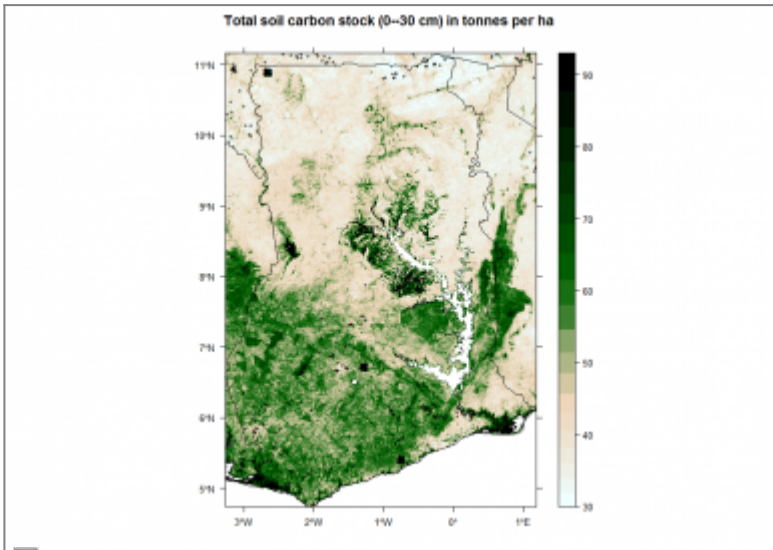
```
+ }
```

next, we can sum all values for each standard thicknesses:

```
> ghana1km$OCS_30cm <- rowSums(ghana1km@data[,paste0("OCS_sd", c(1,2,3))],
na.rm=TRUE)
```

so that the final estimate can be plotted by using:

```
> rg <- quantile(ghana1km$OCS_30cm, c(0.01, 0.99), na.rm=TRUE)
> at <- expm1(seq(log1p(rg[1]), log1p(rg[2]), length.out=20))
> ghana1km$OCS_30cmf <- ifelse(ghana1km$OCS_30cm<rg[1], rg[1],
ifelse(ghana1km$OCS_30cm>rg[2], rg[2], ghana1km$OCS_30cm))
> spplot(ghana1km["OCS_30cmf"], at=at, col.regions=R_pal[["soc_pal"]],
sp.layout=bnd, scales=list(draw=TRUE), main="Total soil carbon stock (0--30
cm) in tonnes per ha")
```



Example of the Soil Organic Carbon Stock map for Ghana.

Note that we use a log-transformed legend to emphasize lower values and adjust for the log-normal distribution of SOCS. Also note that the actual total SOCS for this area is probably two times higher than showed because we only used layers up to 30 cm depth. To derive actual SOCS for the whole depth of soil (0-200 cm), you will need to obtain all layers at all depths and then sum the numbers up (see also function OCSKGM). A more detailed soil organic carbon tutorial you can also find here.

REST service for SoilGrids

SoilGrids are also available through Representational State Transfer (REST) API. The SoilGrids REST API provides support for developers to access data, query and display it, without a need to download the complete grids. REST access to SoilGrids is available through the GSIF package. This allows users to query value of different soil properties for sets of points. Here is an example with two points:

```
> library(rjson)
> library(sp)
```

```
> pnts <- data.frame(lon=c(10.65,5.36), lat=c(51.81,51.48), id=c("p1","p2"))
> coordinates(pnts) <- ~lon+lat
> proj4string(pnts) <- CRS("+proj=longlat +datum=WGS84")
> pnts
class       : SpatialPointsDataFrame
features    : 2
extent      : 5.36, 10.65, 51.48, 51.81  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=,,
variables   : 1
names       : id
min values  : p1
max values  : p2
```

To obtain values of SoilGrids at those two locations we use the generic `over` (overlay) function exported from the sp package (please use package GSIF version ⇒0.5-3):

```
> soilgrids.r <- REST.SoilGrids(c("ORCDRC","PHIHOX"))
> ov <- over(soilgrids.r, pnts)
> str(ov)
```

```
'data.frame':    101 obs. of  35 variables:
 $ soilmask               : Factor w/ 1 level "soil": 1 1 1 1 1 1 1 1 1 1
...
 $ ORCDRC.units_of_measure: Factor w/ 1 level "g / kg": 1 1 1 1 1 1 1 1 1 1
...
 $ ORCDRC.M.sl1           : num  29 34 35 37 35 34 36 37 34 35 ...
 $ ORCDRC.M.sl2           : num  16 23 25 28 24 23 28 28 23 26 ...
 $ ORCDRC.M.sl3           : num  13 16 18 21 17 16 19 20 16 19 ...
 $ ORCDRC.M.sl4           : num  7 11 11 13 11 11 12 13 10 11 ...
 $ ORCDRC.M.sl5           : num  4 6 7 8 7 6 9 9 5 6 ...
 $ ORCDRC.M.sl6           : num  3 5 6 7 5 5 7 8 4 5 ...
 $ ORCDRC.M.sl7           : num  3 7 5 6 7 7 8 7 6 4 ...
 $ PHIHOX.units_of_measure: Factor w/ 1 level "index*10": 1 1 1 1 1 1 1 1 1
1 ...
 $ PHIHOX.M.sl1           : num  63 63 63 64 63 63 62 64 63 63 ...
 $ PHIHOX.M.sl2           : num  63 63 63 64 63 63 62 63 63 63 ...
 $ PHIHOX.M.sl3           : num  63 63 63 64 63 63 62 64 62 63 ...
 $ PHIHOX.M.sl4           : num  63 63 63 64 63 63 63 64 62 63 ...
 $ PHIHOX.M.sl5           : num  63 63 63 64 63 63 63 64 62 63 ...
 $ PHIHOX.M.sl6           : num  64 63 63 64 63 63 63 64 62 63 ...
 $ PHIHOX.M.sl7           : num  64 62 63 63 63 62 63 63 62 63 ...
 $ depthCodesMeters.sd1   : num  -0.025 -0.025 -0.025 -0.025 -0.025 -0.025
-0.025 -0.025 -0.025 -0.025 ...
 $ depthCodesMeters.sd2   : num  -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1
-0.1 -0.1 ...
 $ depthCodesMeters.sd3   : num  -0.225 -0.225 -0.225 -0.225 -0.225 -0.225
-0.225 -0.225 -0.225 -0.225 ...
 $ depthCodesMeters.sd4   : num  -0.45 -0.45 -0.45 -0.45 -0.45 -0.45 -0.45
-0.45 -0.45 -0.45 ...
 $ depthCodesMeters.sd5   : num  -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8 -0.8
-0.8 -0.8 ...
```

```
 $ depthCodesMeters.sd6   : num  -1.5 -1.5 -1.5 -1.5 -1.5 -1.5 -1.5 -1.5
-1.5 -1.5 ...
 $ depthCodesMeters.sl1   : num            ...
 $ depthCodesMeters.sl2   : num  -0.05 -0.05 -0.05 -0.05 -0.05 -0.05 -0.05
-0.05 -0.05 -0.05 ...
 $ depthCodesMeters.sl3   : num  -0.15 -0.15 -0.15 -0.15 -0.15 -0.15 -0.15
-0.15 -0.15 -0.15 ...
 $ depthCodesMeters.sl4   : num  -0.3 -0.3 -0.3 -0.3 -0.3 -0.3 -0.3 -0.3
-0.3 -0.3 ...
 $ depthCodesMeters.sl5   : num  -0.6 -0.6 -0.6 -0.6 -0.6 -0.6 -0.6 -0.6
-0.6 -0.6 ...
 $ depthCodesMeters.sl6   : num  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
 $ depthCodesMeters.sl7   : num  -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 ...
 $ depthCodesMeters.xd1   : num  -0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -0.2
-0.2 -0.2 ...
 $ depthCodesMeters.xd2   : num  -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5
-0.5 -0.5 ...
 $ publication_date       : Factor w/ 1 level "23-04-2016": 1 1 1 1 1 1 1 1
1 1 ...
 $ lon                    : num  -0.715 -0.705 -0.707 -0.709 -0.707 ...
 $ lat                    : num  5.37 5.39 5.39 5.4 5.39 ...
```

The same way we could overlay any set of points and fetch values of soil properties available in the SoilGrids, again far more effecient than downloading the complete GeoTiffs.