Intro to Raster Data in R

Data Carpentry

Started 2020-03-26

This document originated at datacarpentry.org.

Introduction

Questions

- What is a raster dataset?
- How do I work with and plot raster data in R?
- How can I handle missing or bad data values for a raster?

Objectives

- Describe the fundamental attributes of a raster dataset.
- Explore raster attributes and metadata using R.
- Import rasters into R using the raster package.
- Plot a raster file in R using the `ggplot2` package.
- Describe the difference between single- and multi-band rasters.

Things needed to complete this task

See the lesson homepage for detailed information about the software, data, and other prerequisites you will need to work through the examples in this episode.

In this episode, we will introduce the fundamental principles, packages and metadata/raster attributes that are needed to work with raster data in R. We will discuss some of the core metadata elements that we need to understand to work with rasters in R, including CRS and resolution. We will also explore missing and bad data values as stored in a raster and how R handles these elements.

We will continue to work with the `dplyr` and `ggplot2` packages that were introduced in the Introduction to R for Geospatial Data lesson. We will use two additional packages in this episode to work with raster data - the `raster` and `rgdal` packages. Make sure that you have these packages loaded.

```
library(dplyr)
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(raster)
```

```
Loading required package: sp


Attaching package: 'raster'

The following object is masked from 'package:dplyr':

    select
```

```
library(rgdal)
```

```
rgdal: version: 1.4-8, (SVN revision 845)
 Geospatial Data Abstraction Library extensions to R successfully loaded
 Loaded GDAL runtime: GDAL 2.4.2, released 2019/06/28
 Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.6/Resources/library/rgdal/gdal
 GDAL binary built with GEOS: FALSE
 Loaded PROJ.4 runtime: Rel. 5.2.0, September 15th, 2018, [PJ_VERSION: 520]
 Path to PROJ.4 shared files: /Library/Frameworks/R.framework/Versions/3.6/Resources/library/rgdal/proj
 Linking to sp version: 1.3-2
```

```
library(here)
```

```
here() starts at /Users/jrminter/Documents/git/earth-science-maps
```

```
library(ggplot2)
```

Introduce the Data

If not already discussed, introduce the datasets that will be used in this lesson. A brief introduction to the datasets can be found on the Geospatial workshop homepage.

For more detailed information about the datasets, check out the Geospatial workshop data page.

View Raster File Attributes

We will be working with a series of GeoTIFF files in this lesson. The GeoTIFF format contains a set of embedded tags with metadata about the raster data. We can use the function `GDALinfo()` to get information about our raster data before we read that data into R. It is ideal to do this before importing your data.

```
fi <- paste0(here(),"/data/tif/HARV_dsmCrop.tif")
GDALinfo(fi)
```

```
rows        1367
columns     1697
bands       1
lower left origin.x        731453
lower left origin.y        4712471
res.x       1
res.y       1
ysign       -1
oblique.x   0
oblique.y   0
```

```
driver      GTiff
projection  +proj=utm +zone=18 +datum=WGS84 +units=m +no_defs
file        /Users/jrminter/Documents/git/earth-science-maps/data/tif/HARV_dsmCrop.tif
apparent band summary:
   GDType hasNoDataValue NoDataValue blockSize1 blockSize2
1 Float64           TRUE       -9999          1       1697
apparent band statistics:
    Bmin   Bmax    Bmean      Bsd
1 305.07 416.07 359.8531 17.83169
Metadata:
AREA_OR_POINT=Area
```

```
HARV_dsmCrop_info <- capture.output(GDALinfo(fi))
```

Open a raster in R

```
DSM_HARV <- raster(fi)
DSM_HARV
```

```
class      : RasterLayer
dimensions : 1367, 1697, 2319799  (nrow, ncol, ncell)
resolution : 1, 1  (x, y)
extent     : 731453, 733150, 4712471, 4713838  (xmin, xmax, ymin, ymax)
crs        : +proj=utm +zone=18 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
source     : /Users/jrminter/Documents/git/earth-science-maps/data/tif/HARV_dsmCrop.tif
names      : HARV_dsmCrop
values     : 305.07, 416.07  (min, max)
```

The information above includes a report of min and max values, but no other data range statistics. Similar to other R data structures like vectors and data frame columns, descriptive statistics for raster data can be retrieved like

```
summary(DSM_HARV)
```

```
Warning in .local(object, ...): summary is an estimate based on a sample of 1e+05 cells (4.31% of all c
```

```
        HARV_dsmCrop
Min.        305.7200
1st Qu.     345.5775
Median      359.5300
3rd Qu.     374.1100
Max.        413.9000
NA's          0.0000
```

Note the warning - unless you force R to calculate these statistics using every cell in the raster, it will take a random sample of 100,000 cells and calculate from that instead. To force calculation on more, or even all values, you can use the parameter `maxsamp`:

```
summary(DSM_HARV, maxsamp = ncell(DSM_HARV))
```

```
        HARV_dsmCrop
Min.          305.07
1st Qu.       345.59
Median        359.67
3rd Qu.       374.28
Max.          416.07
```

```
NA's            0.00
```

You may not see major differences in summary stats as `maxsamp` increases, except with very large rasters.

To visualize this data in R using `ggplot2`, we need to convert it to a `dataframe`. We learned about dataframes in an earlier lesson. The `raster` package has an built-in function for conversion to a plotable dataframe.

```
DSM_HARV_df <- as.data.frame(DSM_HARV, xy = TRUE)
```

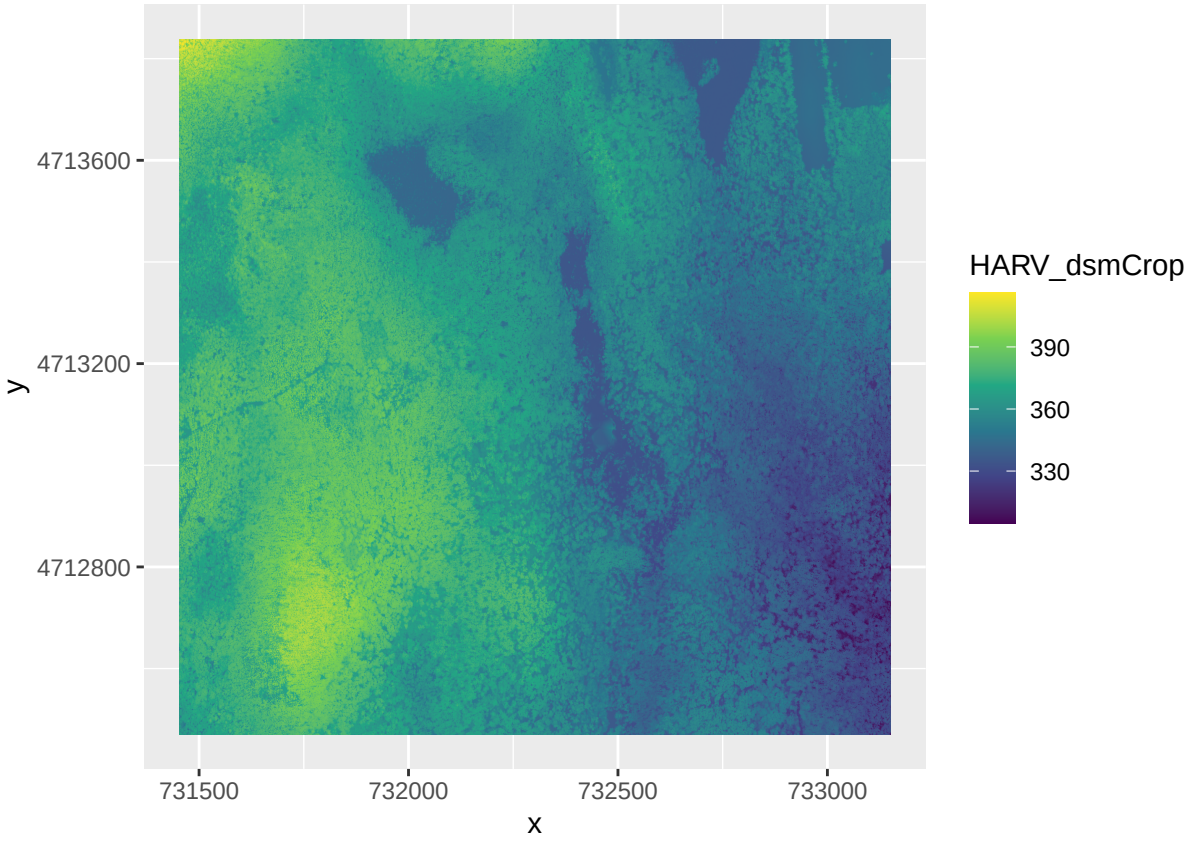
Now when we view the structure of our data, we will see a standard `dataframe` format.

```
str(DSM_HARV_df)
```

```
'data.frame':   2319799 obs. of  3 variables:
 $ x           : num  731454 731454 731456 731456 731458 ...
 $ y           : num  4713838 4713838 4713838 4713838 4713838 ...
 $ HARV_dsmCrop: num  409 408 407 407 409 ...
```

We can use `ggplot()` to plot this data. We will set the color scale to `scale_fill_viridis_c` which is a color-blindness friendly color scale. We will also use the `coord_quickmap()` function to use an approximate Mercator projection for our plots. This approximation is suitable for small areas that are not too close to the poles. Other coordinate systems are available in `ggplot2` if needed, you can learn about them at their help page `?coord_map`.

```
ggplot() +
    geom_raster(data = DSM_HARV_df ,
                aes(x = x, y = y,
                fill = HARV_dsmCrop)) +
    scale_fill_viridis_c() +
    coord_quickmap()
```



More Info

- More information about the Viridis palette used above at R Viridis package documentation.
- For faster, simpler plots, you can use the `plot` function from the `raster` package.