How to make a DEM in R

Emily Burchfield

2017-05-19

What is a DEM?

A DIGITAL ELEVATION MODEL (DEM) is a 3D model of Earth's surface. Most DEMs you'll work with today are built using remote sensing techniques (lidar, synthetic aperture radar), though some of the more old school DEMs are built using contour maps or land surveying techniques. There are plenty of interesting things you can do with a DEM. One of the most common applications in the environmental sciences is modeling the flow of water across a landscape.[1]

[1] Honestly, this is one of the things ArcGIS does quite well. Check out this example in Sri Lanka

The most commonly used DEM in the research community is the ASTER DEM[2]. In 2009, the kind folks at NASA and METI (Japan) released a global 30 meter DEM that provides terrain elevation estimates for nearly the entire globe. Take a second a marvel at that scientific accomplishment. There are plenty of other DEMs out there. The most commonly used are NASA's SRTM DEM which is an earlier global DEM at a 90 meter resolution, the GTOPO30 made by the USGS at a 1 kilometer resolution, and for US the National Elevation Dataset which has bands at multiple resolutions including some pretty hi-res stuff.[3] I also encourage you to explore the world of Lidar, which is often very high resolution and is being used on some extremely impressive mapping projects.[4]

[2] Advanced Spaceborne Thermal Emission and Reflection Radiometer... whew.

You can find lots of DEMs on the web.[5] Read a bit about HOW the data was collected and think deeply about potential data quality issues before you decide which DEM to use.

[3] If you want to go a bit rogue, you can also play with a DEM from Mars. Yay **SCIENCE**!

[4] Like mapping the tree canopy of the Amazon

[5] MapZen is a really cool open source service. You can also check out the OpenTopography website

How do I work with said DEM?

Let's start with a 30 meter DEM from the USGS that is easily accessed as a `R` dataset.[6] To access this dataset, you'll need to install the `landsat` package in `R`. This is probably a good package to install anyways! You can install the package by typing `install.packages("landsat")` in the Console. Ok, let's load the DEM and see what it looks like:

[6] Don't worry, we'll get to downloading and building your own DEMs later

```r
library(landsat)
library(sp)

data(dem)
n = 15
plot(dem, col = topo.colors(n))
```

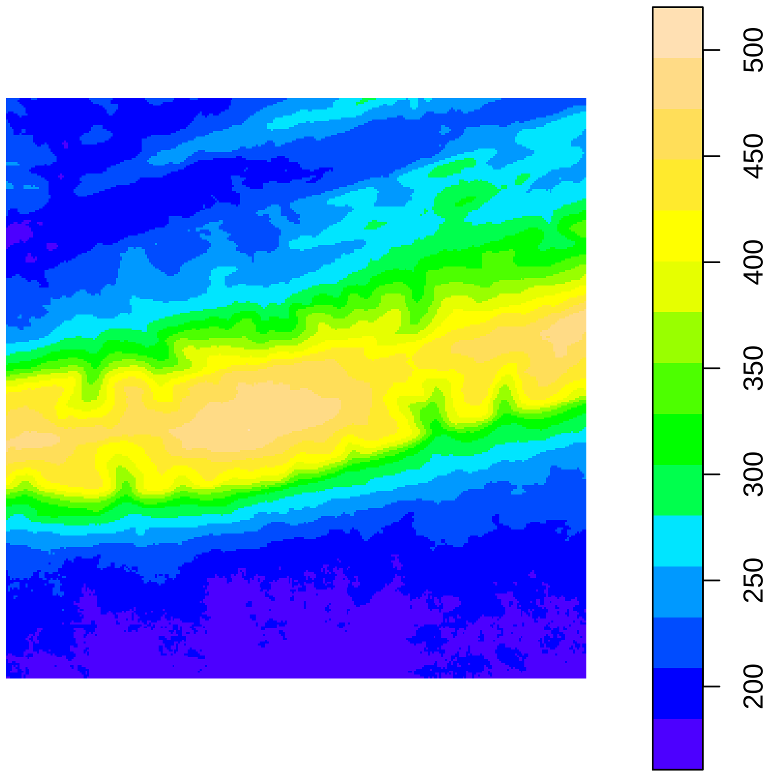


Figure 1: A sample DEM

```
# Change n and see what happens.
```

Ooooohhh, ahhhhhh, what a nice DEM.

How do I get my DEM into `R`*?*

Using `elevatr`

I poked around a bit on the InterWebs and found the following:

The `elevatr` project is a pretty cool package that standardizes access to elevation from web APIs.[7] Let's say you're sitting in Logan, Utah wanting to have a precise estimate of your elevation above sea-level. We'll say you're right around `41.7370° N, 111.8338° W`. First, install the `elevatr` package...

[7] Note, if you want to download LOTS of elevation data, you'll need to get an API Key... check out the documentation if you're going to be downloading crazy amounts of data.

```
library(elevatr)
library(sp)

lat <- 41.737
lon <- 111.8338
```

```
proj <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
# If you’re wondering what this proj thing is,
# see document on projection

sp <- SpatialPoints(cbind.data.frame(lon, lat),
    proj4string = CRS(proj))

get_elev_point(sp, src = "mapzen")
```

```
##          coordinates elevation elev_units
## 1 (111.8338, 41.737)      1483     meters
```

Read the `elevatr` documentation for more information on the `src`.[8]

Ok, well what if we want to actually use `elevatr` to build a raster dataset? What if we want to know the elevation across Cache County? Let’s install the `tigris` package which lets us download US Census shapefiles for states and counties right in `R`.

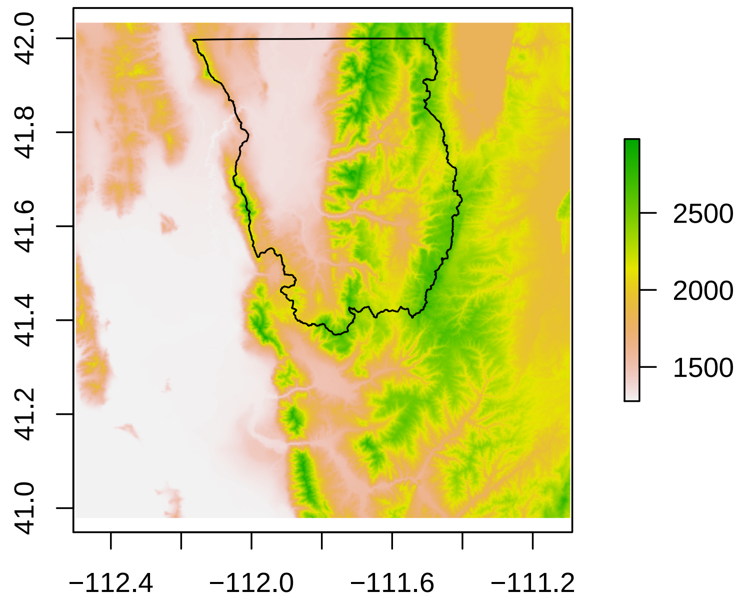
[8] `"mapzen"` pulls data from multiple sources while `"epqs"` pulls only from the USGS.

```
library(elevatr)
library(tigris)
library(maptools)
options(tigrus_use_cache = T)

ut <- counties(state = "UT", cb = T)
cache <- ut[ut$NAME == "Cache", ]

cache_elevation <- get_elev_raster(cache, z = 9,
    src = "aws")
# Amazon Web Service used as source here

plot(cache_elevation)
plot(cache, add = T)
```



The `z` variable reflects the resolution at which the download occurs.[9]

[9] More info found here

```
lat <- 41.737
zoom <- 5
ground_resolution = (cos(lat * pi/180) * 2 * pi *
    6378137)/(256 * 2^zoom)  #meters
```

Using `FedData`

`FedData` is a cool package that makes download of many federal climate and environmental datasets very easy.[10] The `get_ned` function pulls the National Elevation Dataset 30 meter or 10 meter data straight onto your computer. This is high-resolution data, so if you want to download for a large area, it may take awhile. Let's use `tigris` to download data for a sigle Census tract in Cache County:

[10] Go check out the full list of datasets: National Hydrography Dataset, SSURGO, Daymet and more!

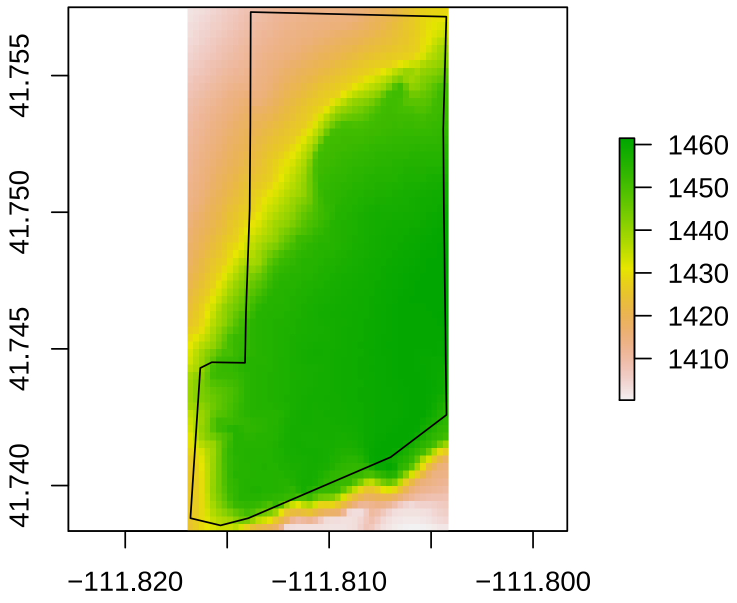
```
library(FedData)
library(tigris)
library(raster)

cache <- tracts(state = "UT", county = "Cache",
    cb = T)
```

```r
tract <- cache[cache$TRACTCE == "000702", ]

dem_directory = "./data"
ned <- get_ned(tract, label = "Cache County",
    res = "1", raw.dir = dem_directory, extraction.dir = dem_directory,
    force.redo = F)
# ned <- raster(paste0(dem_directory, '/Cache
# County_NED_1.tif'))

plot(ned)
plot(tract, add = T)
```


41.755
41.750
41.745
41.740
−111.820
−111.810
−111.800
1460
1450
1440
1430
1420
1410