Applied Economics Research using R: Session 2

Geospatial Data for Applied Economics

Seunghyun Lee (arslee@ucdavis.edu)

10/25/2021

Intro

- Fine scale geospatial data has become popular among applied economists.
- It has opened up new research opportunities.

Issue

- We need to pay some fixed costs.
- There are too much stuff out there.
- To beginners, things may look overwhelming.

Where should we start?

- Should I take a GIS course?
- Should I learn Python, R, ArcGIS, Google Earth Engine?

Goal

- To show you geospatial data analysis as an economist is a doable thing
- To lower the entry barrier to geospatial data

Scoping

- Things you shouldn't do as an economist
  - Image processing using raw images like from unprocessed satellite images
    - predicting wildfire burned areas
    - creating cropland data layers
- Things you should do as an economist
  - Find a research question that may require spatial data
  - Do some geocomputation
    - creating gridded data by interpolating point data
    - constructing field-level crop choice data
    - constructing county-level weather data

Why I use R as a GIS tool

- Script based
- Flexible and rich geoprocessing packages which keep improving
- Nice integration with non geospatial workflow
  - Downloading data
  - Visualization
  - Converting geospatial data to data frame
  - Running regressions
  - R markdown

Why I don't like R as a GIS tool

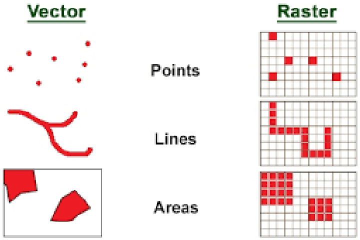
- Again, there are often multiple packages that do similar things
- Different packages may require different syntax
- Checking compatibility between packages is sometimes annoying

Data types

It is *really* important to understand data types and what you can do and can't with them. ~~Otherwise, you may end up doing some silly things like past me. Looking back, it was like trying to use your hands when you are playing soccer.~~

Raster vs Vector Data

- **Raster data** is a representation of the world as a surface divided into a regular grid of cells. Raster data are useful for storing data that varies continuously, as in an aerial photograph, a satellite image, a surface of chemical concentrations, or an elevation surface.
- **Vector data** is a representation of the world using points, lines, and polygons. Vector models are useful for storing data that has discrete boundaries, such as country borders, land parcels, and streets.



Raster vs Vector Data to me

- **Raster data** is matrix (2D) or array (3D) with geographic information
- **Vector data** is data frame with geographic information
- *Note:* Pixel-level data, gridded data and raster data refer to the same thing. I have seen people outside our displine interchangeably using field level and pixel level.

Raster in R

Consider a 5x5 matrix

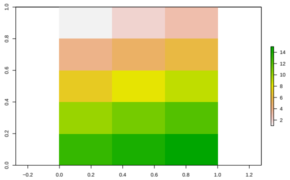
```
M <- matrix(1:15, nrow=5, byrow = T)
M
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
## [5,]   13   14   15
```

Raster in R

You can convert it to raster and plot it

```
library(raster)
R <- raster(M)
plot(R)
```



Raster in R

But we didn't put any geographic information in this case

```
R
```

```
## class       : RasterLayer
## dimensions  : 5, 3, 15  (nrow, ncol, ncell)
## resolution  : 0.3333333, 0.2  (x, y)
## extent      : 0, 1, 0, 1  (xmin, xmax, ymin, ymax)
## crs         : NA
## source      : memory
## names       : layer
## values      : 1, 15  (min, max)
```

Raster in R

Let's consider another matrix.

```
dim(M)
```

```
## [1] 40345 23648
```

```
M[10000:10005,10000:10005]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]  152  152  152  152  152  152
## [2,]  152  142  152  152  152  152
## [3,]  152  152  152  152  152  152
## [4,]  152  152  152  152  152  152
## [5,]  152  142  152  152  152  152
## [6,]  152  152  152  152  152  152
```

Raster in R

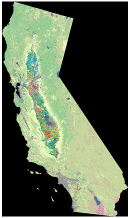
Let's put some geographic information. (Don't worry about the details for now)

```
R <- raster(M)

crs(R) <- crsR
attributes(R)$legend <- legendR
attributes(R)$extent <- extentR
```

Raster in R

```
plot(R)
```



This is CDL for California in 2020 at 30mx30m resolution.

Raster in R

You can stack $2D$ raster layers to create a $3D$ raster. This is called raster stack. What Google Earth Engine is specialized in is pixel-level processing.

Vector data in R

Consider a data frame.

```
class(V)
```

```
## [1] "data.frame"
```

```
head(V)
```

```
##   GEOID          NAME       ALAND
## 1 06091        Sierra  2468694583
## 2 06067    Sacramento  2499983887
## 3 06083 Santa Barbara  7084063392
## 4 06009     Calaveras  2641784992
## 5 06111       Ventura  4771987962
## 6 06037   Los Angeles 10511861492
```

Vector data in R

Also consider a vector of geometry information.

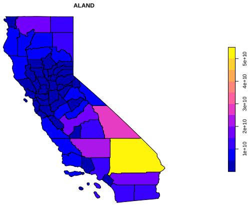
```
geometry
```

```
## Simple feature collection with 58 features and 0 fie
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -124.482 ymin: 32.52883 xmax: -
## Geodetic CRS:  NAD83
## First 10 features:
##                          geometry
## 1  MULTIPOLYGON (((-120.6556 3...
## 2  MULTIPOLYGON (((-121.1886 3...
## 3  MULTIPOLYGON (((-120.7343 3...
## 4  MULTIPOLYGON (((-120.6309 3...
## 5  MULTIPOLYGON (((-119.3292 3...
## 6  MULTIPOLYGON (((-118.7034 3...
```

Vector data in R

Let's bind two set of information and convert to *sf* (simple feature) class.

```
V <- cbind(V,geometry) %>%
  st_as_sf()
plot(V[,"ALAND"])
```



Vector data in R

sf package supports data wrangling in *tidyverse* syntax.
geometry is sticky.

```
V %>%
  filter(NAME == "Yolo") %>%
  mutate(NAME = toupper(NAME))
```

```
## Simple feature collection with 1 feature and 3 fiel
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -122.422 ymin: 38.31305 xmax: -
## Geodetic CRS:  NAD83
##   GEOID NAME      ALAND                          geome
## 1 06113 YOLO 2628144759 MULTIPOLYGON (((-122.1649 3.
```

Vector data in R

You can drop it to make a normal data frame

```
V %>%
  st_drop_geometry() %>%
  head()
```

```
##   GEOID          NAME       ALAND
## 1 06091        Sierra  2468694583
## 2 06067    Sacramento  2499983887
## 3 06083 Santa Barbara  7084063392
## 4 06009     Calaveras  2641784992
## 5 06111       Ventura  4771987962
## 6 06037   Los Angeles 10511861492
```

Some vector data processing

Calculating pair-wise distance

```
V[1:3,]$NAME
```

```
## [1] "Sierra"        "Sacramento"    "Santa Barbara"
```

```
st_distance(V[1:3,1:3])
```

```
## Units: [m]
##           [,1]      [,2]     [,3]
## [1,]      0.00  75378.01 481520.0
## [2,]  75378.01      0.00 349011.8
## [3,] 481520.02 349011.83      0.0
```

Some vector data processing

Extracting neighboring features

```
# select Sacramento
V[2, ]$NAME
```

```
## [1] "Sacramento"
```

```
# select neighboring counties
sac_nb <- st_touches(V)[2]
sac_nb[[1]]
```

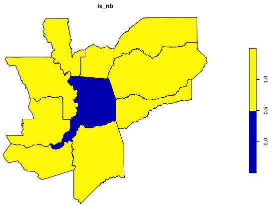
```
## [1] 10 23 25 27 28 41 50 51
```

```
V <- V[c(2, sac_nb[[1]]), ]
```

Some vector data processing

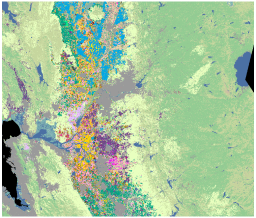
```r
# check with plot

Sac_nb <- V %>%
  mutate(is_nb = NAME !="Sacramento") %>%
  select(is_nb)
plot(Sac_nb[, "is_nb"])
```



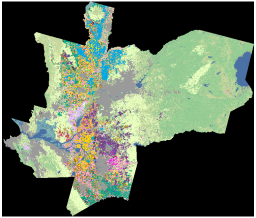
Cropping and masking

```
V <- st_transform(V, crs(R))
crop(R,V) %>% plot()
```



Cropping and masking

```
mask(crop(R,V),V) %>% plot()
```



Cropping and masking

What if you want to do it for every county?

```
library(purrr)
V <- V %>%
  mutate(maskedR = map(GEOID, function(id) {
    county <- V %>% filter(GEOID == id)
    croppedR <- crop(R, county)
    mask(croppedR, county)
  })) %>%
  select(GEOID, NAME, maskedR)
```

Extracting

Masking does not include pixels if pixel centroids fall outside the boundary. You can also extract all raster values as data frame for each boundary including any intersecting pixels simply by

```
library(exactextractr)
V <- V %>% mutate(allR = exact_extract(R, V))
```

```
##    |
```

Extracting

```
V$NAME[[1]]
```

```
## [1] "Sacramento"
```

```
V$allR[[1]] %>% dim()
```

```
## [1] 2868186       2
```

```
V$allR[[1]][1:5,1:2]
```

```
##   value coverage_fraction
## 1   111        0.07551883
## 2   190        0.01096589
## 3   111        0.01818634
## 4   190        0.84255177
## 5    76        0.93342447
```

Extracting (mode)

```
cropcode <- readRDS("Data/Raw/cropcode.rds")
V <- V %>%
  mutate(mode = exact_extract(R, V, "mode")) %>%
  left_join(cropcode, by = c("mode" = "MasterCat")) %>%
  st_drop_geometry()
```

```
##   |
```

```
V[,c("NAME","Crop")]
```

```
##             NAME               Crop
## 1     Sacramento Grassland/Pasture
## 2         Placer   Evergreen_Forest
## 3         Solano Grassland/Pasture
## 4   Contra Costa Grassland/Pasture
## 5      El Dorado   Evergreen_Forest
## 6           Yolo          Shrubland
```

Exercise

Construct county-level data for degree days by using :

- county boundary
- (4km resolution) PRISM temperature grid data
- (30m resolution) National Land Cover Database