Combining Raster and Vector Data

HES 505 Fall 2024: Session 16

Carolyn Koehn

Today's Plan

Objectives

- By the end of today, you should be able to:
  - Convert between raster and vector datasets
  - Generate new rasters describing the spatial arrangement of vector data
  - Extract raster values as attributes of vector data

Converting Between Formats

Converting Between Formats

- Using coercion (`as`, `rast`, `vect`) can change `class`, but not data model
- Sometimes we need to actually change the data model

Converting Vectors to Rasters Using `rasterize`

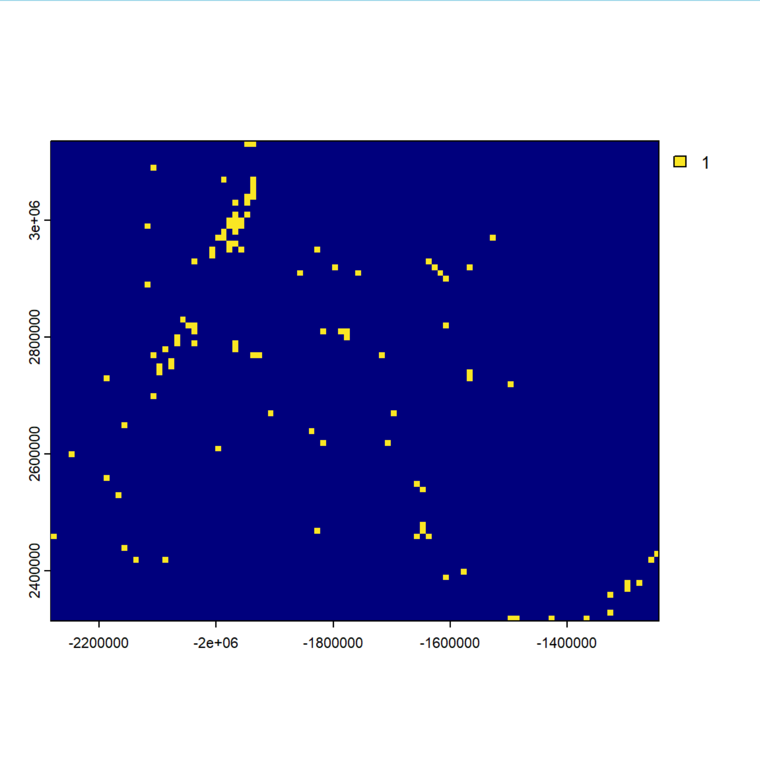
- A special kind of data aggregation
- `x` is your `SpatVector` object
- `y` is a template raster with the appropriate CRS, resolution, and extent
- `fun` allows you to specify the value of the resulting raster

Using `rasterize`

- Presence/Absence
- `field` specifies which value should be returned to non-empty cells

```
hospitals_pnw <- read_csv("/opt/data/data/assignment06/landmar
  filter(., MTFCC == "K2543") %>%
  st_as_sf(., coords = c("longitude", "latitude"), crs=4269) %
  st_transform(crs = 5070)
```

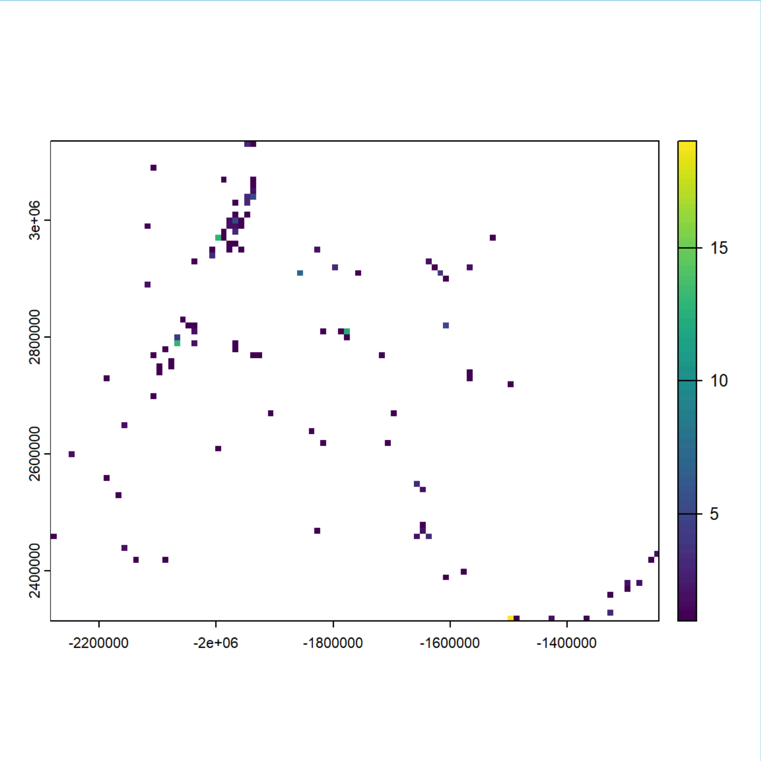
```
raster_template = rast(ext(hospitals_pnw), resolution = 10000,
                       crs = st_crs(hospitals_pnw)$wkt)

hosp_raster1 = rasterize(hospitals_pnw, raster_template,
                         field = 1)
```



Using `rasterize`

- The `fun` argument specifies how we aggregate the data
- Useful for counting occurrences (using `length`)

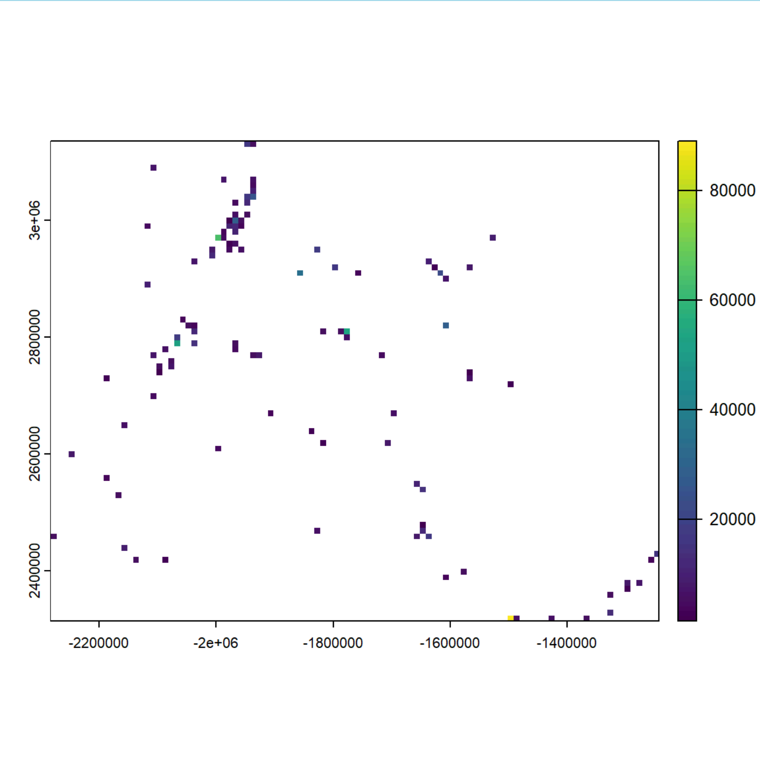
```
hosp_raster2 = rasterize(hospitals_pnw, raster_template,
                         fun = "length")
```



Using `rasterize`

- The `fun` argument specifies how we aggregate the data
- Can use a variety of functions

```r
hospitals_pnw$rand_capacity <- rnorm(n = nrow(hospitals_pnw),
                                     mean = 5000,
                                     sd = 2000)

hosp_raster3 = rasterize(hospitals_pnw, raster_template,
                         field = "rand_capacity", fun = sum)
```



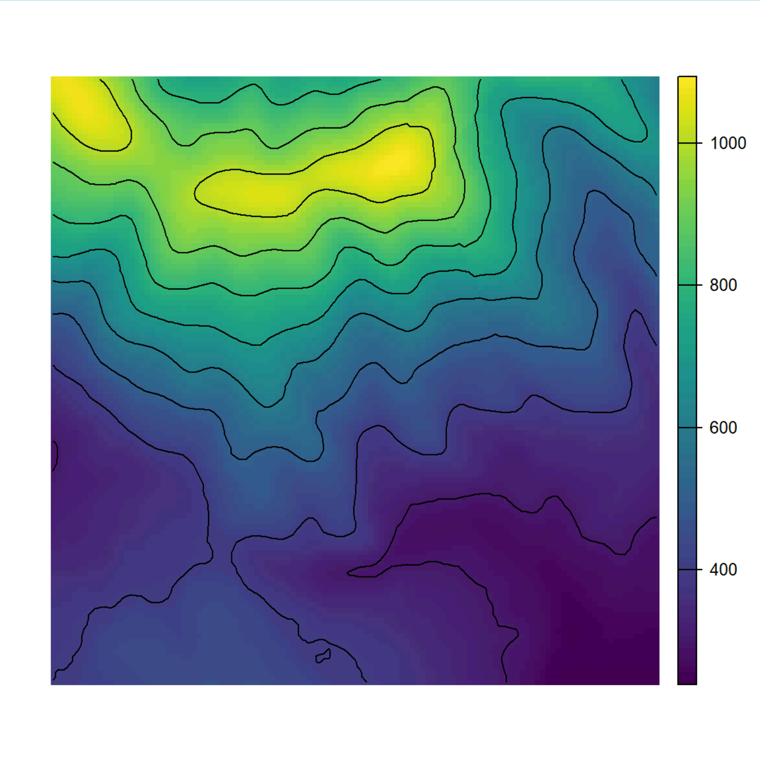
Lines and Polygons

- Can use `rasterize` or `stars::st_rasterize`
- Result depends on the `touches` argument

Converting rasters to vectors

- Less common, but can convert to vector data
- `as.points`, `as.countour`, and `polygonize`

```
dem = rast(system.file("raster/dem.tif", package = "spDataLarg
cl = as.contour(dem)
```



Generating New Data

Generating New Data

- Sometimes we want a raster describing the spatial context of vector data
- `distance` is a simple method
- We'll use interpolation in the next few weeks

Generating Distance Rasters

- returns a distance matrix or `SpatRaster`

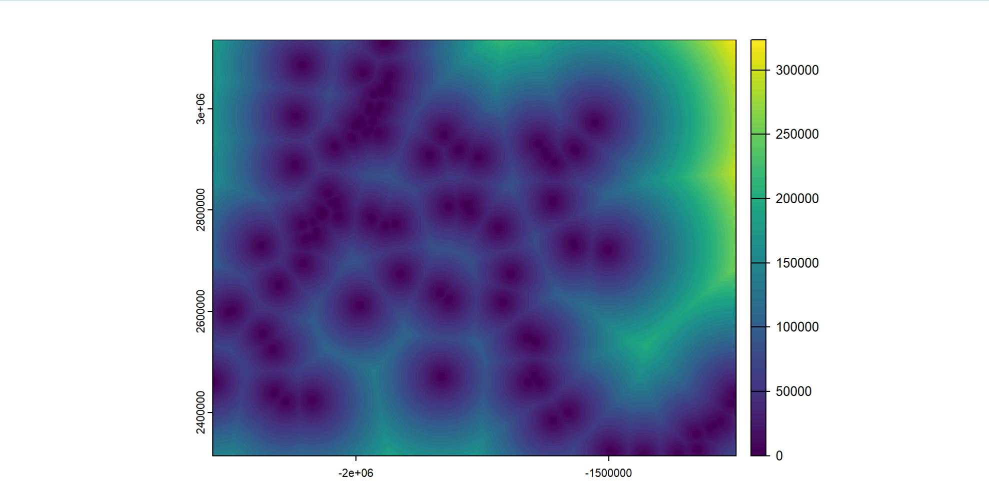
```r
hosp_dist <- distance(vect(hospitals_pnw))
head(as.matrix(hosp_dist))[1:5, 1:5]
```

```
           1         2         3          4        5
1      0.000 209100.3 474603.9   5731.844 422252.6
2 209100.275       0.0 401284.9 204972.864 281571.2
3 474603.876 401284.9       0.0 469036.193 171252.0
4   5731.844 204972.9 469036.2      0.000 416568.2
5 422252.623 281571.2 171252.0 416568.171      0.0
```

Generating Distance Rasters

- returns a distance matrix or `SpatRaster`

```r
raster_template = rast(ext(hospitals_pnw), resolution = 1000,
                       crs = st_crs(hospitals_pnw)$wkt)
hosp_raster1 = rasterize(hospitals_pnw, raster_template,
                       field = 1)

hosp_dist_rast <- distance(hosp_raster1)
plot(hosp_dist_rast)
```



Creating Vector Data by Extraction

- Sometimes we want to use rasters to create new attributes
- fun controls how the cells are aggregated

```
wildfire_haz <- rast("/opt/data/data/assignment07/wildfire_hazard_agg.tif")
```

```
hospitals_pnw_proj <- st_transform(hospitals_pnw, crs(wildfire_haz))

hosp_fire_haz <- terra::extract(wildfire_haz, hospitals_pnw_proj)
head(hosp_fire_haz)
```

```
  ID    WHP_ID
1  1 1952.8750
2  2    0.0000
3  3  741.4531
4  4  200.2812
5  5    0.0000
6  6  150.5938
```

Creating Vector Data by Extraction

- Can use `zonal` for one summary statistic for polygons

```r
cejst <- st_read("/opt/data/data/assignment06/cejst_pnw.shp") %>%
  st_transform(crs = crs(wildfire_haz)) %>%
  filter(!st_is_empty(.))
```

```r
wildfire.zones <- terra::zonal(wildfire_haz, vect(cejst), fun="mean", na.rm

head(wildfire.zones)
```

```
       WHP_ID
1    3.053172
2 2997.795051
3    6.647930
4   85.971309
5   34.706535
6   17.306250
```

3 ways to extract raster data for polygons

```r
system.time(wildfire.zones <- terra::zonal(wildfire_haz, vect(cejst), fun="
```

```
  user  system elapsed 
 31.66    1.36   33.12 
```

```r
system.time(wildfire.zones2 <- terra::extract(wildfire_haz, vect(cejst), fu
```

```
  user  system elapsed 
 31.63    1.06   32.91 
```

```r
system.time(wildfire.zones3 <- exactextractr::exact_extract(wildfire_haz, c
```

```
  user  system elapsed 
  2.94    0.17    3.10 
```

```
       WHP_ID                ID      WHP_ID          [1]     3.230088
1    3.053172              1  1    3.053172          2997.102783
2 2997.795051              2  2 2997.795051          6.464695   86.015327
3    6.647930              3  3    6.647930          34.672573   16.559727
4   85.971309              4  4   85.971309
5   34.706535              5  5   34.706535
6   17.306250              6  6   17.306250
```

Motivating Question

> How do Collaborative Forest Landscape Restoration projects compare to other National Forest lands with respect to social and wildfire risks?

Thinking about the data

- **Datasets** - Forest Service Boundaries, CFLRP Boundaries, Wildfire Risk Raster, CEJST shapefile
- **Dependent Variable** - CFLRP (T or F)
- **Independent Variables** - Wildfire hazard, income, education, housing burden

Building some Pseudocode

```
1. Load libraries
2. Load data
3. Check validity and alignment
4. Subset to relevant geographies
5. Select relevant attributes
6. Extract wildfire risk
7. CFLRP T or F
8. Compare risks
```

Load libraries

```r
library(sf)
library(terra)
library(tidyverse)
library(tmap)
```

Load the data

- Downloading USFS data using the function in the code folder

```r
download_unzip_read <- function(link){
  tmp <- tempfile()
  download.file(link, tmp)
  tmp2 <- tempfile()
  unzip(zipfile=tmp, exdir=tmp2)
  shapefile.sf <- read_sf(tmp2)
}

### FS Boundaries
fs.url <- "https://data.fs.usda.gov/geodata/edw/edw_resources/shp/S_USA.Adm
fs.bdry <- download_unzip_read(link = fs.url)

### CFLRP Data
cflrp.url <- "https://data.fs.usda.gov/geodata/edw/edw_resources/shp/S_USA.
cflrp.bdry <- download_unzip_read(link = cflrp.url)
```