ESM 263 HW2

Linus Blomqvist

2021-01-26

Loading and exploring data

Check what layers are in the basemap:

```
st_layers("HW2/data_hw2/basemap.gpkg")$name
```

```
## [1] "California" "Cities"     "County"     "ROI"        "Streets"
```

Read them in:

```
california <- read_sf("HW2/data_hw2/basemap.gpkg", layer = "California")
cities <- read_sf("HW2/data_hw2/basemap.gpkg", layer = "Cities")
county <- read_sf("HW2/data_hw2/basemap.gpkg", layer = "County")
ROI <- read_sf("HW2/data_hw2/basemap.gpkg", layer = "ROI")
streets <- read_sf("HW2/data_hw2/basemap.gpkg", layer = "streets")
```

For this and any other layers we can do a little exploring. For example, we can check the variable names:

```
names(california)
```

```
##  [1] "OBJECTID"     "NAME_PCASE"   "NAME_UCASE"   "FMNAME_PC"    "FMNAME_UC"
##  [6] "ABBREV"       "NUM"          "ABCODE"       "FIPS"         "ANSI"
## [11] "ISLAND"       "Shape_Leng"   "Shape_Length" "Shape_Area"   "geom"
```

Note that in an `sf` object, there's always a column at the end called `geom`; this contains all the spatial information.

We can also do a quick plot to see what we have in the `california` shape. Looks like counties.

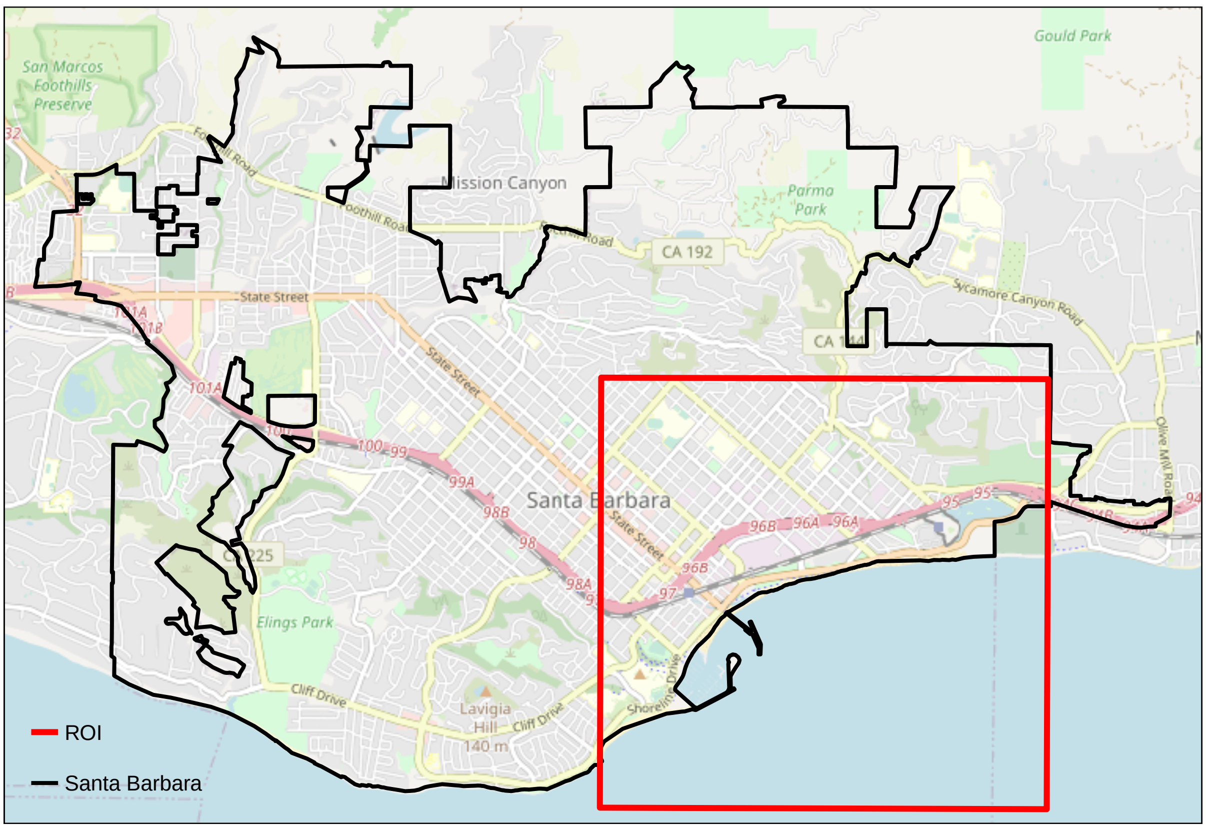
```r
tm_shape(st_geometry(california)) +
  tm_polygons()
```

I'm also interested to see the ROI (region of interest). In the html version of this file, you'll be able to see this against a base layer and zoom around etc, as the map view mode is interactive, but in the pdf version, it's a static map. We can see that we're only interested in what looks like the downtown area including the harbor.

For the static map, I use the `read_osm` function, where osm stands for Open Street Map, to create a base layer. This requires a bounding box which is represented by the Santa Barbara city feature. This is a multipolygon with two parts, the actual city, and the airport, so I turn it into a polygon with two separate features, and select the second one, Santa Barbara proper.

```r
# Create feature for bounding box and read Open Street Map layer
sb <- st_geometry(cities[cities$CITY == "Santa Barbara",])
sb_polygon <- st_cast(sb, to = "POLYGON")
osm <- read_osm(st_buffer(sb_polygon[2], 300), type = "osm")  # use buffer to
# have a bit of margin around the map
```

```r
# Map
tm_shape(osm) +
  tm_rgb(alpha = 0.7) +
  tm_shape(sb_polygon[2]) +
  tm_borders(col = "black", lwd = 2) +
  tm_shape(ROI) +
  tm_borders(col = "red", lwd = 3) +
  tm_add_legend(type = "line", lwd = 3, col = "red", labels = "ROI", alpha = 1) +
  tm_add_legend(type = "line", lwd = 2, col = "black", labels = "Santa Barbara", alpha = 1) +
  tm_layout(legend.position = c("left", "bottom"))
```

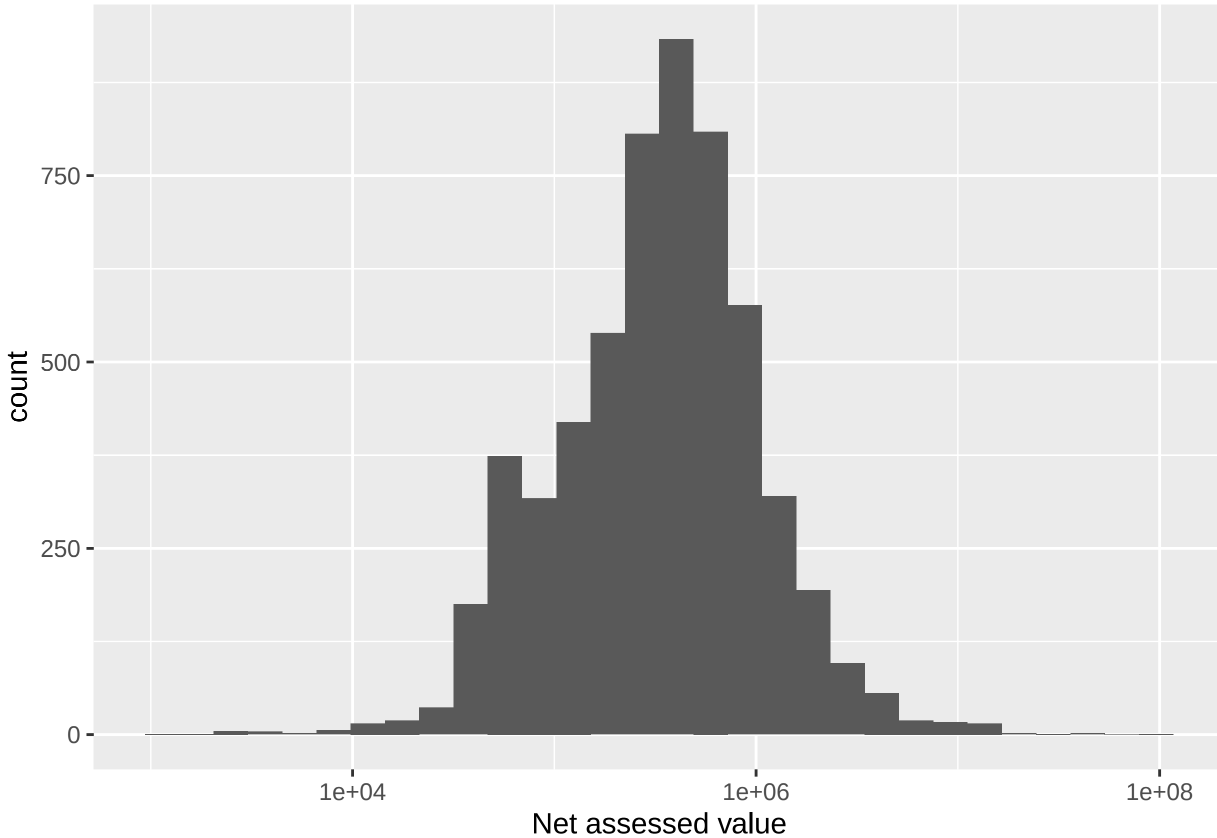


There's only one layer, "parcels", in the parcels file so we'll read that in.

```r
parcels <- read_sf("HW2/data_hw2/parcels.gpkg")
```

The variable we're interested in is `NET_AV`, so we can check what that looks like (on a log 10 scale to make it easier to read).

```r
ggplot(parcels) +
  geom_histogram(aes(x = NET_AV)) +
  scale_x_log10() +
  xlab("Net assessed value")
```



Seems like most parcels are valued at just under a million dollars, but some are worth tens of millions of dollars.

For the inundation scenarios, I combine all the layers into a single `sf` object.

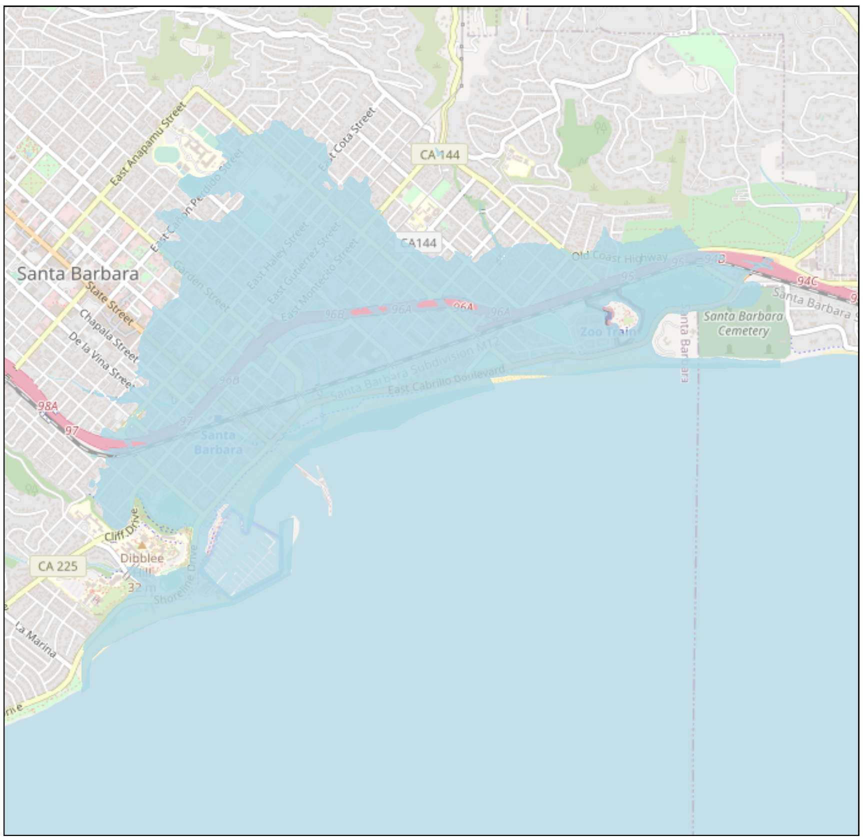
```r
# Get layer names
inund_layers <- st_layers("HW2/data_hw2/inundation_scenarios.gpkg")$name

# Start with one layer and then row bind the others onto it with a loop
scenarios <- st_read("HW2/data_hw2/inundation_scenarios.gpkg", layer = inund_layers[1], quiet = TRUE)
for(i in 2:length(inund_layers)) {
  scenarios <- rbind(scenarios, read_sf("HW2/data_hw2/inundation_scenarios.gpkg", layer = inund_layers[
}
```

Let’s look at one of these scenarios:

```r
# New bounding box for the base layer
osm_ROI <- read_osm(st_buffer(ROI, 500), type = "osm")


# Map
tm_shape(osm_ROI) +
  tm_rgb(alpha = 0.7) +
  tm_shape(st_geometry(filter(scenarios, GRIDCODE == 10))) +
  tm_fill(col = "lightblue", alpha = 0.8)
```



Seems like this represents current land area that would be inundated under the scenario in question.

Spatial join

What we want here is the total value of all parcels that fall within the inundated area for each scenario.

```r
# Calculate the parcel areas
parcels$area <- st_area(parcels) # calculate area
units(parcels$area) <- with(ud_units, ha) # convert from m^2 to ha
```

```r
parcels$area <- drop_units(parcels$area) # roundabout way of doing this,
# but it is less prone to human error to use the units package for conversions

# Do the join and summarize for each of the three variables
scenarios <- scenarios %>%
  st_join(parcels, join = st_intersects) %>%
  group_by(GRIDCODE) %>% # this is the ID of each scenario
  summarize(parcel_count = n(), net_value = round(sum(NET_AV)/1e6, 0), area = round(sum(area), 0))

# Rename GRIDCODE column
names(scenarios)[1] <- "scenario"
```

Results: table

Now we can turn this into a table.

```r
st_drop_geometry(scenarios) %>%
  kbl(col.names = c("Sea-level rise (m)", "Parcel count", "Net loss ($m)", "Area flooded (ha)")) %>%
  kable_material(c("striped", "hover"))
```

Sea-level rise (m)	Parcel count	Net loss ($m)	Area flooded (ha)
1	60	51	43
2	89	88	108
3	227	195	189
4	620	566	301
5	1275	909	354
6	1863	1215	392
7	2196	1372	436
8	2614	1555	465
9	2958	1712	486
10	3287	1881	512

Results: map

In the map, the numbers 1 through 10 represent the number of meters of sea-level rise associated with each scenario and the intensity of the color represents the amount of property value lost.

```
tmap_mode("plot")
tm_shape(scenarios) +
  tm_polygons("net_value", title = "Net loss ($m)") +
  tm_facets(by = "scenario", nrow = 5, ncol = 2) +
  tm_layout(main.title = "Inundation scenarios for downtown Santa Barbara",
            legend.position = c("right", "bottom"),
            main.title.size = 0.8)
```