US19-FOR-875-730 - R Programming for Data Science YC

exercise_sp

FOR/STT 875, Spatial Data Exercise

Learning objectives

- practice loading and reprojecting spatial data
- analyze spatial data
- create `leaflet` maps to convey analysis results
- interpret analysis results

Overview and deliverables

Create a dynamic web map to convey information in spatial data.

Submission instructions

Upload your exercise_sp.Rmd and exercise_sp.html files to the Spatial Data Exercise D2L dropbox.

Grading

You will receive full credit if your R Markdown document: 1) compiles without error; and 2) creates a map that resembles and behaves like the Map 1 in the Colorado wells Section. Also, please, fill in the feedback Questions at the end of the exercise.

Getting started

This exercise builds on Chapter 8 by illustrating some R `leaflet` 's functionality for creating dynamic web maps. Your task is to recreate the look and behavior of the map in the Colorado wells Section. To do this you will need to become familiar with the data described below and work through the tutorials on RStudio's leaflet site https://rstudio.github.io/leaflet/ (https://rstudio.github.io/leaflet/). Please review the following topics in the left menu on the R leaflet site: Introduction, Map Widget, Basemaps, Popups and Labels, Lines and Shapes, Colors, Legends, Show/Hide Layers, and Choropleths. These are short tutorials and quite fun, so this should be fairly painless.

Motivating data

All data for this exercise is available at http://blue.for.msu.edu/FOR875/data/CO-wells.zip (http://blue.for.msu.edu/FOR875/data/CO-wells.zip). Let's use the `downloader` package `download` function to grab the data, then unzip it into your working directory (note, you should set your working directory to the directory dedicated to this exercise prior to running the code chunk below), and take a look at the content.

```
library(downloader)

download("http://blue.for.msu.edu/FOR875/data/CO-wells.zip",
         destfile="./CO-wells.zip", mode="wb")

unzip("CO-wells.zip", exdir = ".")

list.files("CO-wells")
```

```
## [1] "CO-HUC.cpg"   "CO-HUC.dbf"   "CO-HUC.prj"   "CO-HUC.qpj"
## [5] "CO-HUC.shp"   "CO-HUC.shx"   "CO-wells.csv"
```

For this exercise we'll use some data that was recently highlighted in an April 30, 2017, Denver Post piece entitled Home explosion reignites debate over drilling setbacks, but with a twist (http://www.denverpost.com/2017/04/30/firestone-home-explosion-drilling-setbacks/) by Christopher Osher and Bruce Finley (no relation). This article pulled data from the Colorado Oil & Gas Conservation Commission (COGCC) (http://cogcc.state.co.us/data2.html#/downloads) to generate this map (http://www.denverpost.com/2017/05/01/oil-gas-wells-colorado-map/) of oil and gas wells across Colorado (CO).

In preparation for this exercise, I downloaded and cleaned up the COGCC data (e.g., removed some variables and corrected `Facil_Stat` variable data entry errors). The resulting csv file is called "CO-wells.csv" and is included in "CO-wells.zip". The file contains the following variables (columns):

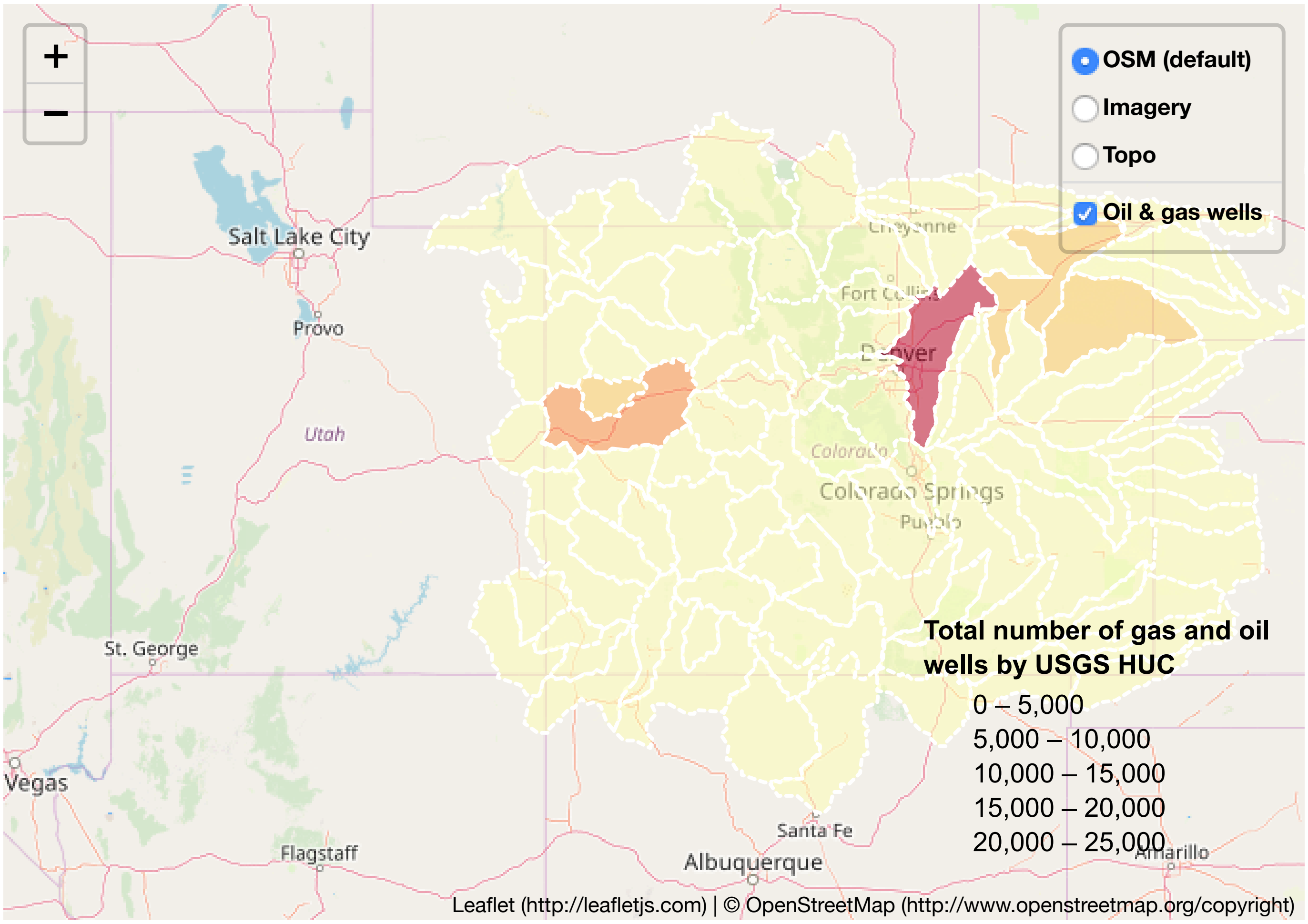
1. `Latitude` geographic coordinate system.
2. `Longitude` geographic coordinate system.
3. `Spud_Date` spud is the process of beginning to drill a well in the oil and gas industry. This is the date (year-month-day) when drilling was started.
4. `Spud_Year` The year when drilling was started.
5. `Facil_Stat` Facility status in COGIS database (AB-Abandoned, AC-Active, AL-Abandoned Location, CL-Closed, CM-Commingled, DA-Dry and Abandoned, DG-Drilling, DM-Domestic Well, IJ-Injecting, PA-Plugged and Abandoned, PR-Producing, RC-Recompleted, SI-Shut In, TA-Temporarily Abandoned, WO-Waiting on Completion, XX-Permitted Location).
6. `Max_TVD` Maximum total vertical well depth in feet.

The "CO-HUC" shapefile in the "CO-wells.zip" comprises non-overlapping polygons that delineate hydrologic units (i.e., watersheds) each with a unique code. These hydrologic unit codes (HUC) (https://water.usgs.gov/GIS/huc.html) and associated shapefiles are maintained and distributed by the United States Geological Survey (USGS) for the entire US. I downloaded and extracted only those HUC that fell within Colorado. The "CO-HUC" shapefile is in Albers equal-area conic projection. The column in "CO-HUC"'s data frame are:

1. `HUC_ID` a unique integer code for each HUC polygon.
2. `HUC_NAME` the name of each HUC polygon (these may or may not be unique).

Colorado wells

Your goal for this exercise is to produce a leaflet map that looks and behaves like Map 1. You can take some artistic (or should I say cartographic) liberties and change the color scheme and other style elements.



Map 1: Total, active, and inactive oil and gas wells by USGS hydrologic units across Colorado. You can see a full browser window version of this map here (http://blue.for.msu.edu/FOR875/data/exercise_sp_map1.html).

Here are the steps you'll need to follow to create the map above.

1. Read in "CO-wells.csv" and promote it into a `SpatialPointsDataFrame` . As noted above `Latitude` and `Longitude` are in proj.4 string "+proj=longlat +datum=WGS84" geographic coordinate system, so this will need to be set using the `proj4string` function. I called the resulting object `wells` .
2. Read in "CO-HUC" shapefile and reproject it to "+proj=longlat +datum=WGS84". Note it is in some other projection when you first read it in. I called the resulting `SpatialPolygonsDataFrame` object `huc`
3. Create a new column in `wells` called `HUC_ID` which records the HUC ID from the spatially coinciding `HUC` polygons. This can be done using the `over` function.
4. For each HUC ID we want to count the: a) number of wells; b) number of active wells; c) number of inactive wells. Active wells are defined by `Facil_Stat` values "AC" and "PR". Inactive wells are all values of `Facil_Stat` other than "AC" and "PR". Use a sequence of `dplyr` functions on `wells@data` to calculate these three quantities for each `HUC_ID` . The function sequence should use `group_by()` and `summarize()` . Within `summarize()` you can use the `length` function to get total number of wells and the `sum` function with logical statements to get the number of active and inactive wells. Here is a description of my resulting object called `huc.wells` (yours should be the same):

```
huc.wells
```

```
## # A tibble: 91 x 4
##    HUC_ID total active inactive
##     <int> <int>  <int>    <int>
##  1    873     1      0        1
##  2    880     5      0        5
##  3    899   773    179      594
##  4    903   313    139      174
##  5    932   203     31      172
##  6    936     2      0        2
##  7    947  3095   1404     1691
##  8    951  4308   1945     2363
##  9    963  1231    571      660
## 10    964     4      0        4
## # ... with 81 more rows
```

5. Now use the `left_join` function to join `huc@data` to `huc.wells` by `HUC_ID` and assign the result back to `huc@data` . My `huc@data` now looks like this:

```
dim(huc@data)
```

```
## [1] 93  5
```

```
head(huc@data)
```

```
##   HUC_ID                            HUC_NAME total active inactive
## 1    846                  Upper North Platte    NA     NA       NA
## 2    873 Upper Green-Flaming Gorge Reservoir     1      0        1
## 3    880                       Upper Laramie     5      0        5
## 4    899                        Little Snake   773    179      594
## 5    903                           Vermilion   313    139      174
## 6    932                     Lower Lodgepole   203     31      172
```

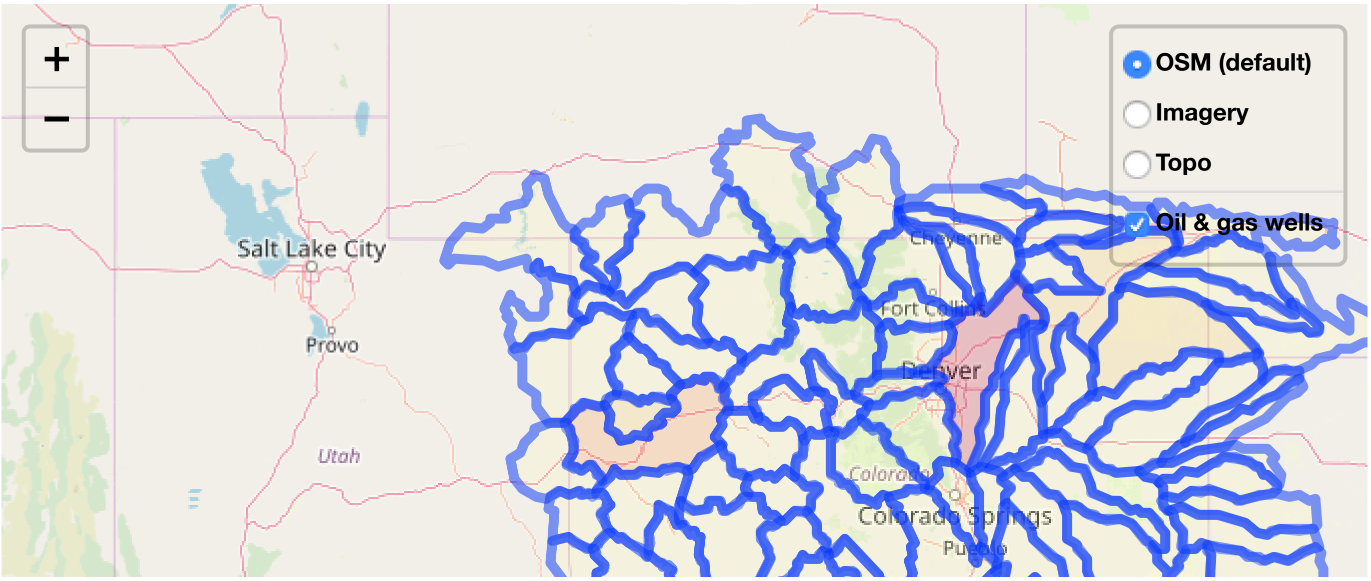
6. Notice above that `huc@data` has 93 rows, i.e., 93 polygons. However, `huc.wells` has only 91 rows. This is because `over` only returns `HUC_ID` 's that contain at least one well. The left join of `huc@data` to `huc.wells` in Step 5 maintains the 93 rows in `huc@data` but fills the two missing `huc.wells` with `NA` s, see, e.g., the `NA` for `total` , `active` , and `inactive` in `huc@data` 's `HUC_ID` 846 above. Our map should show zeros for these `NA` , because there are zero wells in these HUC's (at least those portions of the HUC that are in Colorado). I used the following code to replace the offending `NA` s with zeros (take your time working through this code chunk, notice how the `ifelse` functions work, they can be very handy. Double check your results to be sure the `NA` values were converted to zeros.

```
huc@data <- huc@data %>%
  mutate(total = ifelse(is.na(total), 0, total),
         active = ifelse(is.na(active), 0, active),
         inactive = ifelse(is.na(inactive), 0, inactive))
head(huc@data)
```

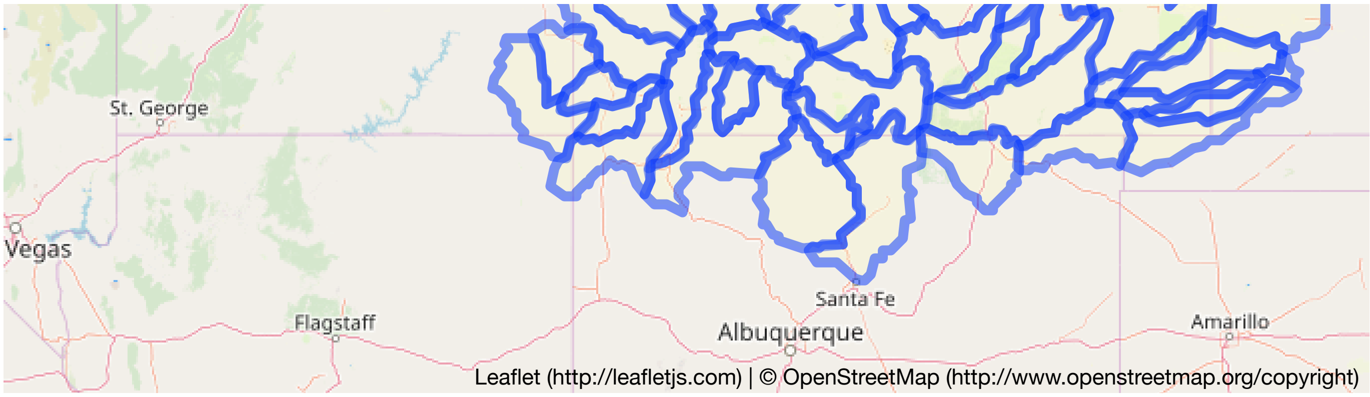
```
##   HUC_ID                            HUC_NAME total active inactive
## 1    846                  Upper North Platte     0      0        0
## 2    873 Upper Green-Flaming Gorge Reservoir     1      0        1
## 3    880                       Upper Laramie     5      0        5
## 4    899                        Little Snake   773    179      594
## 5    903                           Vermilion   313    139      174
## 6    932                     Lower Lodgepole   203     31      172
```

7. Now we're ready to make our first `leaflet` map. Here's a simple one that lacks the bells and whistles that your final map should include. I start by defining a `leaflet` color palette (see the Colors tutorial on RStudio's leaflet site referenced above). Then I construct the leaflet map. A few things to note. First the `width = "100%"` in the initial `leaflet()` call is only necessary when compiling your leaflet map in R Markdown (it tells it to span the width of the webpage). Second, the `huc` polygons are added to the map using the `addPolygons` function with `fillColor` argument defined using `pal`. Third I added several basemap options (see other options here (http://leaflet-extras.github.io/leaflet-providers/preview/index.html)). Finally, the layer controls that appear in the top right corner of the map use the `group` argument in `addPolygons` and `addTiles` to identify the on/off toggles for each layer.

```
pal <- colorBin("YlOrRd", domain = huc$total, bins = 6, pretty = TRUE)

leaflet(width = "100%") %>%

  # Overlay groups
  addPolygons(data = huc, fillColor = ~pal(total), group = "Oil & gas wells") %>%

  # Base groups
  addProviderTiles(providers$OpenStreetMap.Mapnik, group = "OSM (default)") %>%
  addProviderTiles(providers$Esri.WorldImagery, group = "Imagery") %>%
  addProviderTiles(providers$Esri.WorldTopoMap, group = "Topo")  %>%

  # Layers control
  addLayersControl(
    overlayGroups = c("Oil & gas wells"),
    baseGroups = c("OSM (default)", "Imagery", "Topo"),
    options = layersControlOptions(collapsed = FALSE)
  )
```



● OSM (default)
○ Imagery
○ Topo
☑ Oil & gas wells


St. George
Vegas
Flagstaff
Santa Fe
Albuquerque
Amarillo
Leaflet (http://leafletjs.com) | © OpenStreetMap (http://www.openstreetmap.org/copyright)

8. Work through the tutorials on RStudio's leaflet site https://rstudio.github.io/leaflet/ (https://rstudio.github.io/leaflet/) to add the remaining functionality, including mouse over popups/labels, polygon outlines, and legend. One last piece of guidance. Construction of the labels that appear when mousing over the HUCs can be a bit tricky. There are several ways to get this `label` behavior; however, I used the approach detailed in the "Custom info" Section toward the bottom of the RStudio's leaflet site Choropleths (https://rstudio.github.io/leaflet/choropleths.html) page. In fact working the Choropleths tutorial will get you most of the way through this exercise

Reflect in ePortfolio Download Print ‹ ›

Activity Details

Task: View this topic