Code

STAT 209: Lab 14

Working With Geospatial Data

The `sf` package

We will attempt to take advantage of the bleeding edge tidyverse friendly R package for geospatial data, called `sf`. Even though our textbook is less than three years old, this R package had not been released in its final form at the time of publication. However, compared to the previous state of the art, it simplifies the interface for interacting with spatial layers substantially and makes it play nicely with other tidyverse packages, most especially `dplyr` and `ggplot`. What a time to be alive!

Installing `sf` can be a bit of a hassle, so I suggest working on the `360pc` server for this lab.

Before you go further, make sure you are able to load the `sf` package, with `library()`. If you get an error, it may be because you have installed packages to your user account that have different versions than the system-wide ones. If this comes up let me know and I will show everyone how to get around it.

Opening a dataset

The `sf` package comes with some example datasets. One of these is about cases of Sudden Infant Death Syndrome in North Carolina. The dataset is described at this link (https://cran.r-project.org/web/packages/spdep/vignettes/sids.pdf) (key excerpt quoted below)

“This data set was presented first in Symons et al. (1983), analysed with reference to the spatial nature of the data in Cressie and Read (1985), expanded in Cressie and Chan (1989), and used in detail in Cressie (1991). It is for the 100 counties of North Carolina, and includes counts of numbers of live births (also non-white live births) and numbers of sudden infant deaths, for the July 1, 1974 to June 30, 1978 and July 1, 1979 to June 30, 1984 periods. In Cressie and Read (1985), a listing of county neighbours based on shared boundaries (contiguity) is given, and in Cressie and Chan (1989), and in Cressie (1991, pp. 386–389), a different listing based on the criterion of distance between county seats, with a cutoff at 30 miles. The county seat location coordinates are given in miles in a local (unknown) coordinate reference system. The data are also used to exemplify a range of functions in the S-PLUS spatial statistics module user’s manual (Kaluzny et al., 1996)”

Read in the data as follows:

Code:

Code

```
## Reading layer `nc' from data source `/usr/local/lib/R/site-library/sf/shape/nc.shp'
using driver `ESRI Shapefile'
## Simple feature collection with 100 features and 14 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
```

The main advantage of `sf` over older packages is that we can interact with the `nc` object as a regular `data.frame`.

For example, we can view the `AREA`, `NAME` and `geometry` variables with `select()` as we normally would.

Code:

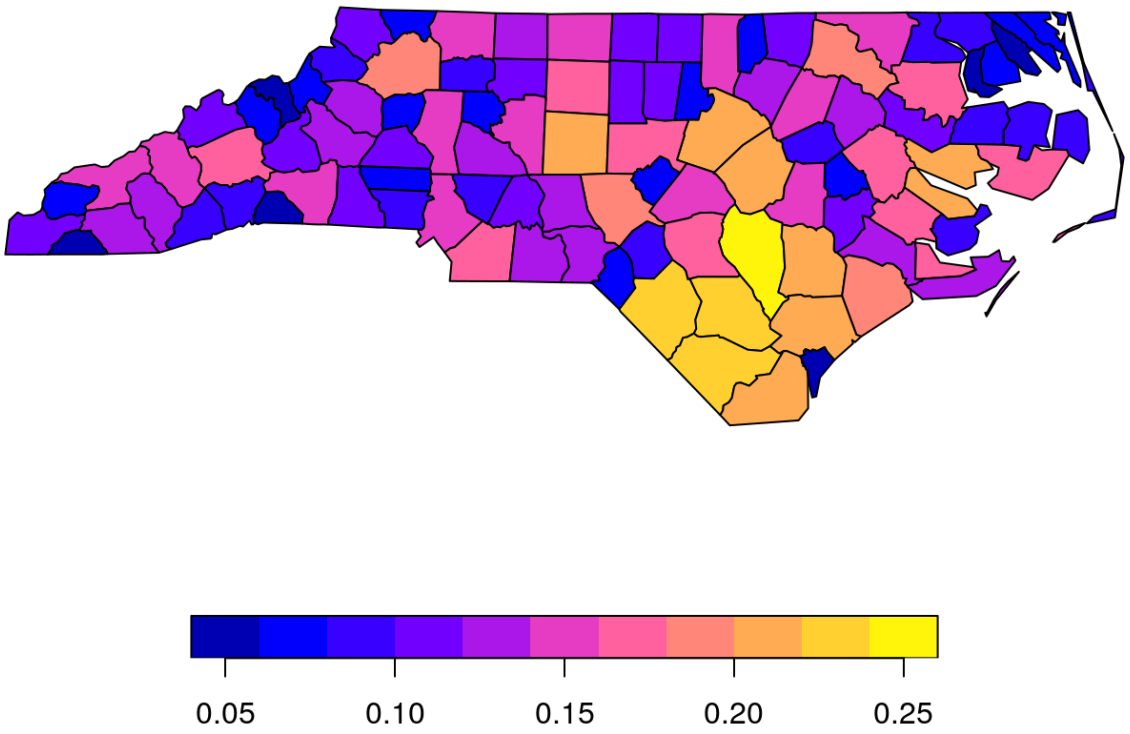
Code

```
## Simple feature collection with 100 features and 2 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
## First 10 features:
##     AREA        NAME                       geometry
## 1  0.114        Ashe MULTIPOLYGON (((-81.47276 3...
## 2  0.061   Alleghany MULTIPOLYGON (((-81.23989 3...
## 3  0.143       Surry MULTIPOLYGON (((-80.45634 3...
## 4  0.070   Currituck MULTIPOLYGON (((-76.00897 3...
## 5  0.153 Northampton MULTIPOLYGON (((-77.21767 3...
## 6  0.097    Hertford MULTIPOLYGON (((-76.74506 3...
## 7  0.062      Camden MULTIPOLYGON (((-76.00897 3...
## 8  0.091       Gates MULTIPOLYGON (((-76.56251 3...
## 9  0.118      Warren MULTIPOLYGON (((-78.30876 3...
## 10 0.124      Stokes MULTIPOLYGON (((-80.02567 3...
```

There is also a plot method for this type of object that makes it very easy to see the data in map form.

The following code will produce a map of the counties in North Carolina, colored by their size in land area. **Code:**

Code



If we use the latest version of `ggplot2`, we can also use the `geom_sf()` function to create a similar plot.

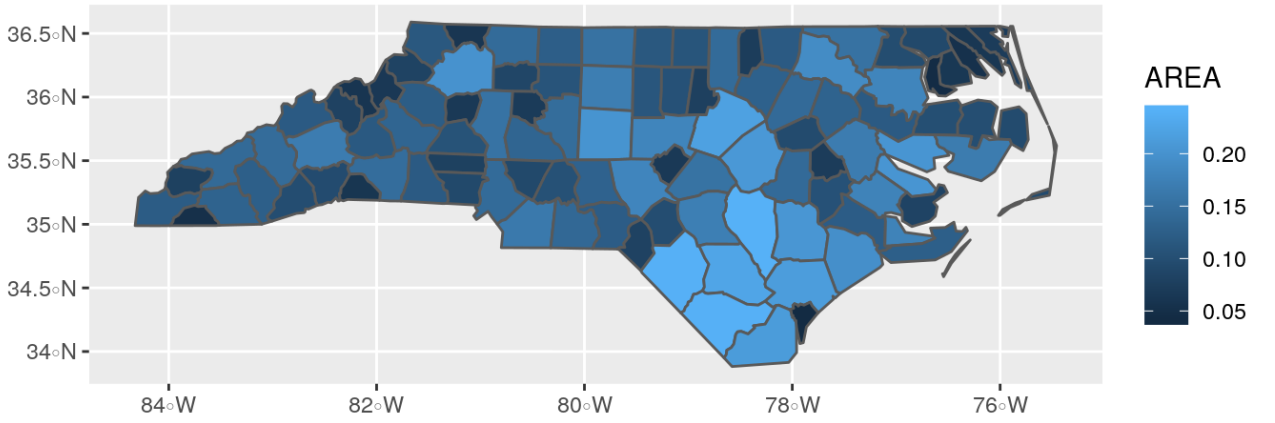
Code

```
##  [1] "AREA"      "PERIMETER" "CNTY_"     "CNTY_ID"   "NAME"
##  [6] "FIPS"      "FIPSNO"    "CRESS_ID"  "BIR74"     "SID74"
## [11] "NWBIR74"   "BIR79"     "SID79"     "NWBIR79"   "geometry"
```

Code

```
## Simple feature collection with 6 features and 14 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -81.74107 ymin: 36.07282 xmax: -75.77316 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
##    AREA PERIMETER CNTY_ CNTY_ID        NAME  FIPS FIPSNO CRESS_ID BIR74
## 1 0.114     1.442  1825    1825        Ashe 37009  37009        5  1091
## 2 0.061     1.231  1827    1827   Alleghany 37005  37005        3   487
## 3 0.143     1.630  1828    1828       Surry 37171  37171       86  3188
## 4 0.070     2.968  1831    1831   Currituck 37053  37053       27   508
## 5 0.153     2.206  1832    1832 Northampton 37131  37131       66  1421
## 6 0.097     1.670  1833    1833    Hertford 37091  37091       46  1452
##   SID74 NWBIR74 BIR79 SID79 NWBIR79                       geometry
## 1     1      10  1364     0      19 MULTIPOLYGON (((-81.47276 3...
## 2     0      10   542     3      12 MULTIPOLYGON (((-81.23989 3...
## 3     5     208  3616     6     260 MULTIPOLYGON (((-80.45634 3...
## 4     1     123   830     2     145 MULTIPOLYGON (((-76.00897 3...
## 5     9    1066  1606     3    1197 MULTIPOLYGON (((-77.21767 3...
## 6     7     954  1838     5    1237 MULTIPOLYGON (((-76.74506 3...
```

Code



Exercise 1 Using the NC dataset and your wrangling and visualization skillz, create a plot showing the geographic distribution of the number of births in each of two years: 1974 (in the `BIR74` variable) and 1979 (in the `BIR790`

variable), where year is a faceting variable. (Hint: You will need a `gather()` for the faceting part)

Exercise 2 Create a similar plot, this time showing the number of SIDS cases in these two years, **normalized by the number of births**.

Data with multiple layers

A more complex dataset comes from the `macleish` package, which has spatial data coupled with weather data recorded from two monitor stations in Massachussetts.

To get the data, first load the `macleish` package with `library()`.

The data consists of multiple *layers*, each in the form of an `sf` data frame, and collected into an R list called `macleish_layers`. See the names of the layers with `names(macleish_layers)`.

Code:

Code

```
##  [1] "landmarks"         "forests"           "streams"
##  [4] "challenge_courses" "buildings"         "wetlands"
##  [7] "boundary"          "research"          "soil"
## [10] "trails"            "contours_30ft"     "contours_3m"
```

To access individual layers in this list, you can use the `extract2()` function from the `magrittr` package. This is a “pipe-friendly” version of using double square brackets.

In a nutshell, when working with a proper list object, single square brackets or `extract()` (`macleish_layers %>% extract(3)` or `macleish_layers %>% extract(3:4)`) return a sub-list, whereas double brackets or `extract2()` actually return the entries in the list. We can refer to the entries by name using quotes.

For example, the buildings layer: **Code:**

Code

```
## Simple feature collection with 27 features and 1 field
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: -72.68251 ymin: 42.44104 xmax: -72.67923 ymax: 42.44919
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## First 10 features:
##    name                       geometry
## 1     0 POLYGON ((-72.68056 42.4484...
## 2     0 POLYGON ((-72.68051 42.4483...
## 3     0 POLYGON ((-72.68123 42.4464...
## 4     0 POLYGON ((-72.68116 42.4463...
## 5     0 POLYGON ((-72.68134 42.4461...
## 6     0 POLYGON ((-72.68135 42.4451...
## 7     0 POLYGON ((-72.68137 42.4450...
## 8     0 POLYGON ((-72.68102 42.4445...
## 9     0 POLYGON ((-72.68098 42.4444...
## 10    0 POLYGON ((-72.68143 42.4443...
```

We can get a quick plot of the buildings in this layer by simply piping the layer into `plot()`.

Code

There is a tutorial, or “vignette” in R parlance, that gives you some context, examples, and analysis of this data, available by typing `vignette("macleish")`.

Exercise 3 Plot the “streams” layer of the `macleish` data.

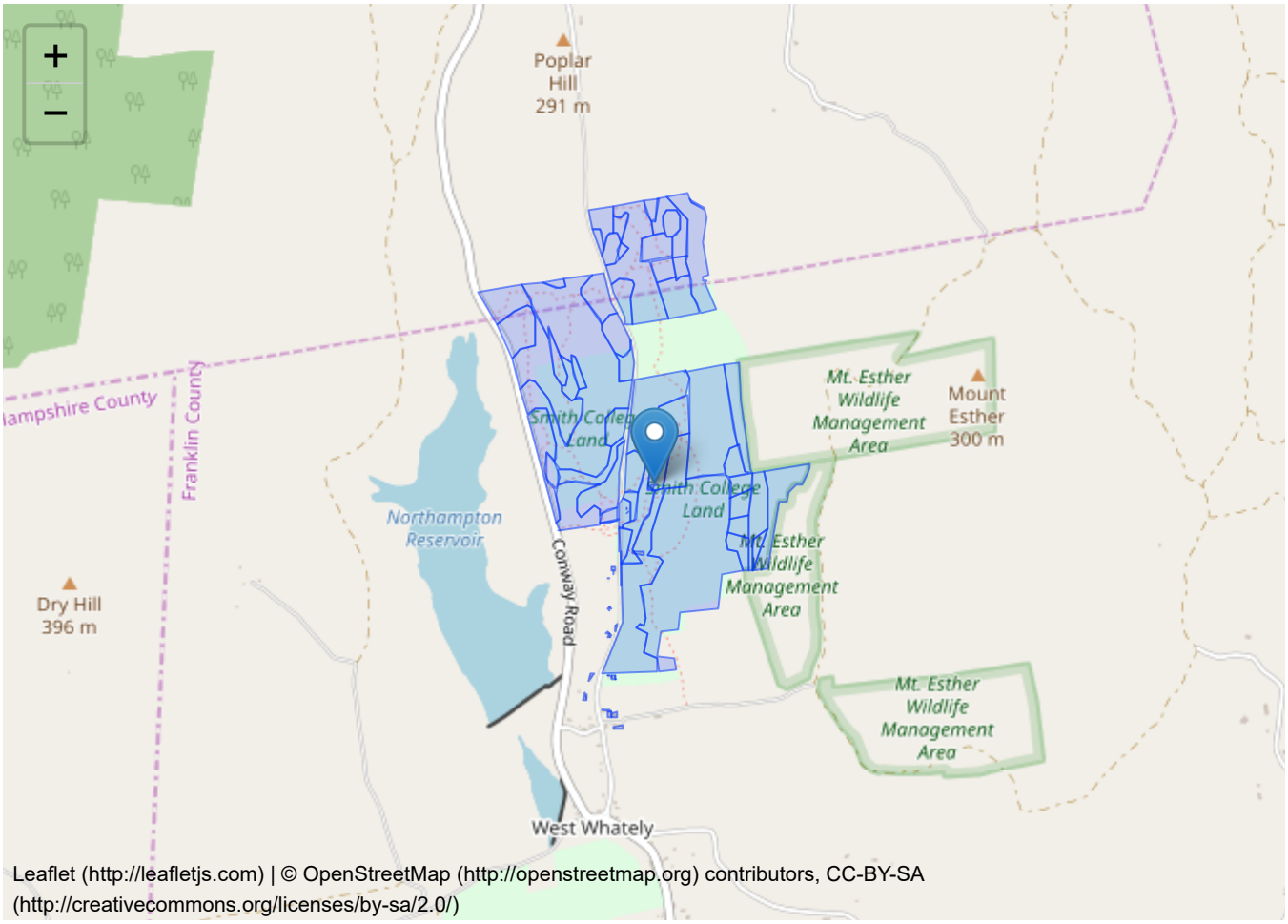
Interactive plots with `leaflet`

When working with multiple layers, it can be nicer to create an interactive plot so the user can get context about the data by hovering.

We can use yet another plotting library well suited for spatial data, called `leaflet`. Here is an example, showing a particular building in the context of buildings and forests.

Code:

Code



Exercise 4 Using the above example as a guide, make an interactive visualization of the MacLeish data, including layers for buildings, streams, and trails.

Getting credit

Upload your plot for Exercise 4 to #lab15 on Slack by Thursday’s class.