Overlapping Areas

Ari Anisfeld

12/18/2017

Introduction

This document has two main sections. First, we create a zip-code to senate district weight matrix and, second, we apply it to CWS data to interpolate senate districts data from the zipcode level. The two parts are nearly self-contained as the main output from the first part is saved in a csv that is accessible in the folder. Unless you are particularly intersted in playing with the geographic data / maps, I recommend reading through part 1 and then doing part 2 interactively.

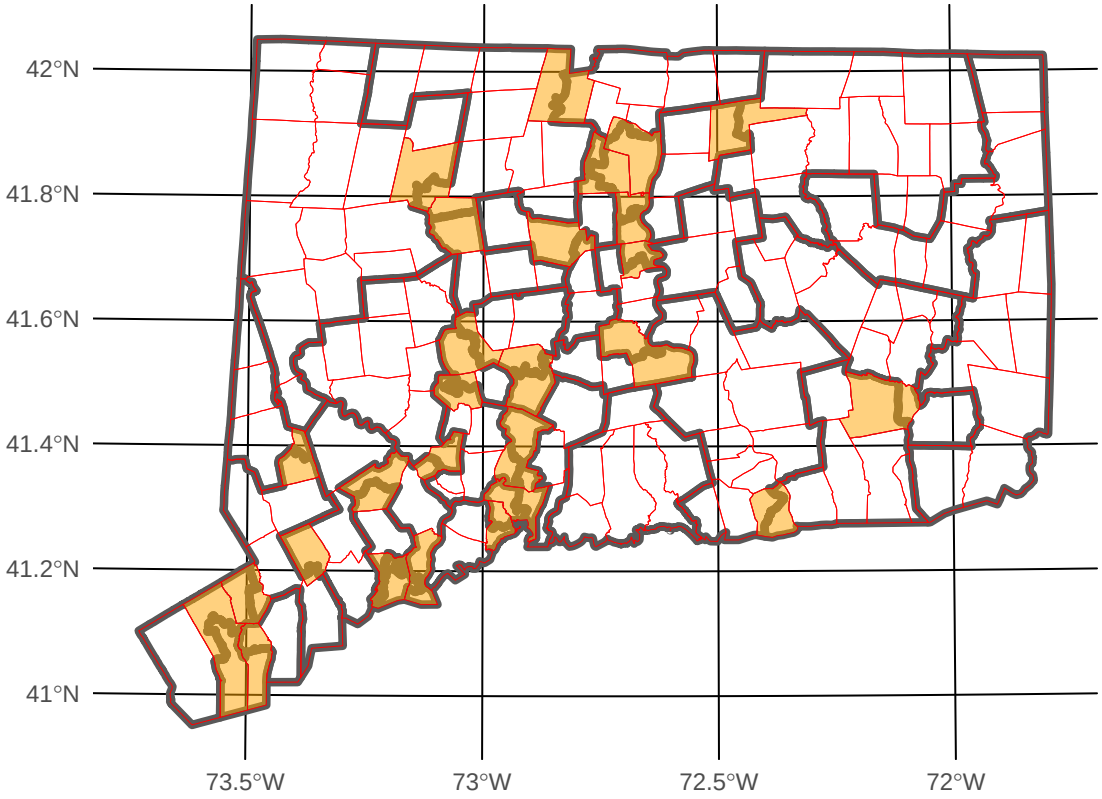
PART 1

Visualizing the problem

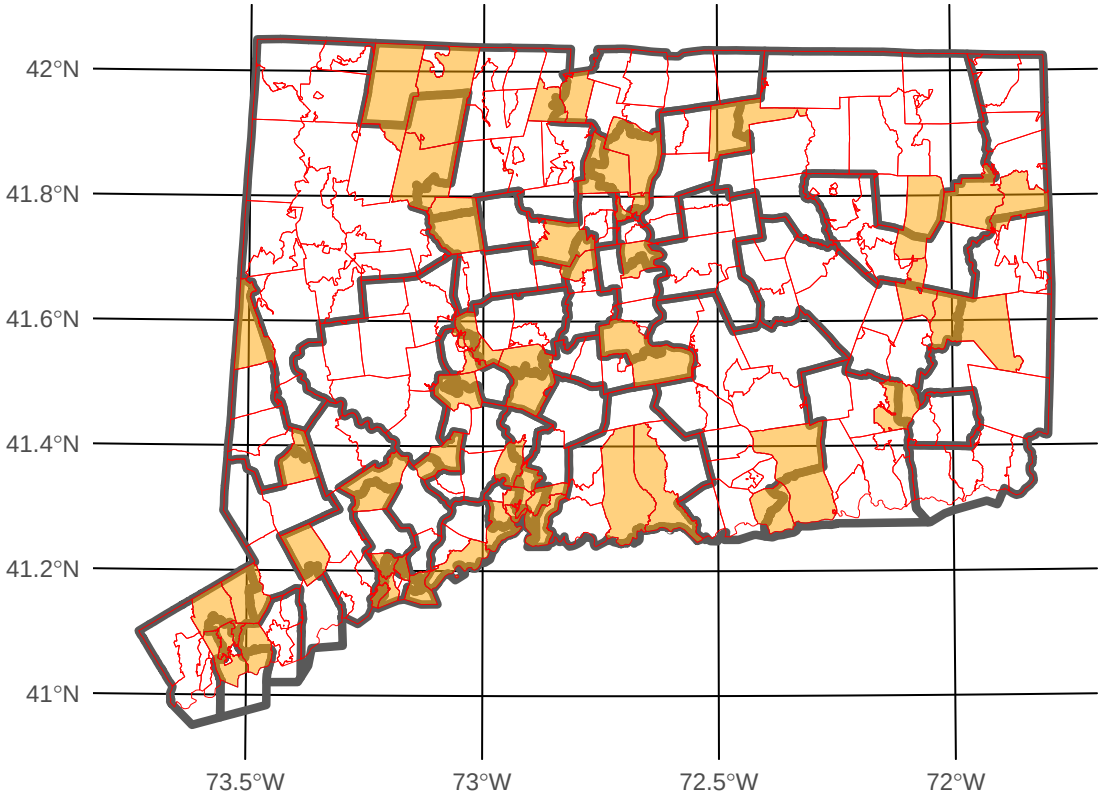
My initial approach tried to make use of geographic analysis packages. The main issue is the boundaries of senate districts are drawn agnostic to where zipcodes are. Small deviations between lines lead to a number of sliver polygons and make it analytically difficult to determine which polygons overlap. I hacked around this, but could not get it perfect as you will see in the maps below. Numerically, these overlaps amount to rounding errors, but a more precise future iteration would use additional geographic data.

```
## Reading layer `senatect_37800_0000_2010_s100_census_1_shp_nad83_feet' from data source `/Users/arian
## Simple feature collection with 37 features and 12 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 730515.2 ymin: 544042.3 xmax: 1263092 ymax: 944273.9
## epsg (SRID):    NA
## proj4string:    +proj=lcc +lat_1=41.2 +lat_2=41.86666666666667 +lat_0=40.83333333333334 +lon_0=-72.7
## Reading layer `townct_37800_0000_2010_s100_census_1_shp_nad83_feet' from data source `/Users/arianis
## Simple feature collection with 173 features and 20 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 730515.2 ymin: 544042.3 xmax: 1263092 ymax: 944273.9
## epsg (SRID):    NA
## proj4string:    +proj=lcc +lat_1=41.2 +lat_2=41.86666666666667 +lat_0=40.83333333333334 +lon_0=-72.7
## Reading layer `zipct_37800_0000_2010_s100_census_1_shp_nad83_feet' from data source `/Users/arianisf
## Simple feature collection with 282 features and 11 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 730515.2 ymin: 553460.1 xmax: 1263092 ymax: 944273.9
## epsg (SRID):    NA
## proj4string:    +proj=lcc +lat_1=41.2 +lat_2=41.86666666666667 +lat_0=40.83333333333334 +lon_0=-72.7
```

Towns not within a single senate district



zip codes not within a single senate district



Zipcodes and senate districts maps are not built to be overlaid, so there are a number of boundaries particularly along the coast that create sliver polygons that represent roughly 3% of the zipcodes area. Clean slivers removes most of these, but notice there's still a zipcode near Sherman (41.6 N, 73.5 W) that falls completely within a senate district, but is treated as if it were in two districts. Inspection of satellite images show that the overlapping area is a body of water.

Interpolation methods

The goal is to interpolate state senate estimates given zipcode and town level data. I will refer to the units with known data as the subgeography.

Any interpolation requires assumptions. Most basically we assume variable X is evenly distributed within a subgeography, after conditioning on information Z, which could include any covariate known at both the subgeography and interpolation level. For example, if a zipcode contained males and females and was divided between two senate districts that were unisex, we could easily assign X associated with males to the man-senate district. In this analysis, we will not use covariates for conditioning.

When no conditioning information is used, the interpolation simply requires identifying a weighting matrix W.

$$W * X = \tilde{X}$$

where W is $m \times n$ where m is the number of polygons in the interpolated geography and n is the number of polygons in the subgeography. w_{is} represents the fraction of subgeography unit s within interpolated geography unit i.

Area overlap

The simplest method of interpolatation is weighting based on area overlap. This method assumes all variables are evenly distributed over the given subgeography. Then, weight w_{is} is found by the fraction of the subgeography s that is contained within interpolated geography i. This is purely geometeric.

```
## # A tibble: 282 x 37
##    ZCTA5CE10 `001` `002` `003` `004` `005` `006` `007` `008` `009` `010`
##       <fctr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     06001     0  0.00     0     0  0.00     0     0     1     0     0
##  2     06002     0  0.55     0     0  0.45     0     0     0     0     0
##  3     06010     0  0.00     0     0  0.00     0     0     0     0     0
##  4     06013     0  0.00     0     0  1.00     0     0     0     0     0
##  5     06016     0  0.00     1     0  0.00     0     0     0     0     0
##  6     06018     0  0.00     0     0  0.00     0     0     0     0     0
##  7     06019     0  0.00     0     0  0.00     0     0     1     0     0
##  8     06020     0  0.00     0     0  0.00     0     0     1     0     0
##  9     06021     0  0.00     0     0  0.00     0     0     1     0     0
## 10     06022     0  0.00     0     0  0.00     0     0     1     0     0
## # ... with 272 more rows, and 26 more variables: `011` <dbl>, `012` <dbl>,
## #   `013` <dbl>, `014` <dbl>, `015` <dbl>, `016` <dbl>, `017` <dbl>,
## #   `018` <dbl>, `019` <dbl>, `020` <dbl>, `021` <dbl>, `022` <dbl>,
## #   `023` <dbl>, `024` <dbl>, `025` <dbl>, `026` <dbl>, `027` <dbl>,
## #   `028` <dbl>, `029` <dbl>, `030` <dbl>, `031` <dbl>, `032` <dbl>,
## #   `033` <dbl>, `034` <dbl>, `035` <dbl>, `036` <dbl>
```

Part 2

Working with CWS data

In the previous section, we created the transformation weights matrix. Now we pull in CWS data and use the matrix to interpolate data at the senate district level.

```
# example 1
q5_at_zipcode_level <- cws %>% get_cws(q5)
q5_at_zipcode_level
```

```
## # A tibble: 280 x 4
## # Groups:   ctzip [280]
##    ctzip          n         `1`        `2`
##  * <chr>      <dbl>       <dbl>      <dbl>
##  1 06001  77.073665  32.1077742  44.965891
##  2 06002  98.935900  35.1270284  63.808872
##  3 06010 276.340667 100.0037576 176.336909
##  4 06013  40.547512  19.9129951  20.634517
##  5 06016  28.309607   3.0817714  25.227836
##  6 06018  14.597100   0.0000000  14.597100
##  7 06019  37.504851  23.6081614  13.896690
##  8 06020   7.118936   0.8384671   6.280469
##  9 06021   5.579302   1.9271377   3.652164
## 10 06023   7.776399   1.8785445   5.897854
## # ... with 270 more rows
```

```
# example 2
rates_q5_at_zipcode_level <- cws %>% get_cws(q5, zip_rates = TRUE)
rates_q5_at_zipcode_level
```

```
## # A tibble: 280 x 4
## # Groups:   ctzip [280]
##    ctzip          n       `1`       `2`
##  * <chr>      <dbl>     <dbl>     <dbl>
##  1 06001  77.073665 0.4165855 0.5834145
##  2 06002  98.935900 0.3550484 0.6449516
##  3 06010 276.340667 0.3618858 0.6381142
##  4 06013  40.547512 0.4911028 0.5088972
##  5 06016  28.309607 0.1088596 0.8911404
##  6 06018  14.597100 0.0000000 1.0000000
##  7 06019  37.504851 0.6294695 0.3705305
##  8 06020   7.118936 0.1177798 0.8822202
##  9 06021   5.579302 0.3454084 0.6545916
## 10 06023   7.776399 0.2415700 0.7584300
## # ... with 270 more rows
```

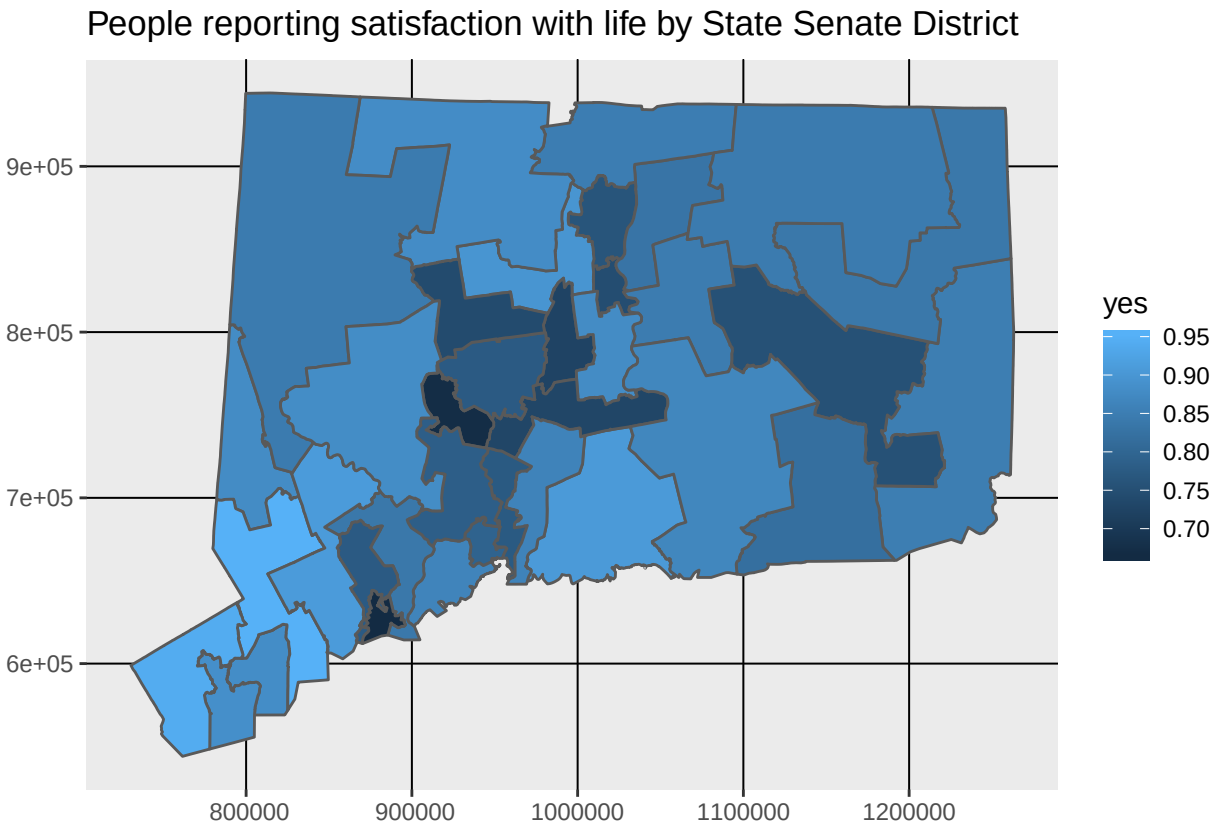
```
# example 1
# note when you make rates for likert questions, you want to pass the cws_senate() function
# data with count data
(q1_rates <- (cws %>% get_cws(q1) %>% cws_senate(rates = TRUE)))
```

```
## # A tibble: 36 x 4
##    district        n       `1`       `2`
##  *    <chr>    <dbl>     <dbl>     <dbl>
##  1      001 362.0729 0.7459315 0.2540685
```

```
##  2      002 423.6287 0.7625082 0.2374918
##  3      003 438.0646 0.8296178 0.1703822
##  4      004 463.2482 0.8461415 0.1538585
##  5      005 471.8245 0.8958544 0.1041456
##  6      006 416.3603 0.7231674 0.2768326
##  7      007 471.3509 0.8480130 0.1519870
##  8      008 381.1049 0.8783984 0.1216016
##  9      009 440.8934 0.8777862 0.1222138
## 10      010 440.3043 0.7926105 0.2073895
## # ... with 26 more rows
```

```
# this code returns garbage:
# cws %>% get_cws(q1, zip_rates=TRUE) %>% cws_senate(rates = TRUE)
```

```
# attach to geo data
q1_geom <- left_join(q1_rates, senate, by=c("district" = "SLDUST10")) %>% rename('yes' = '1')
ggplot() + geom_sf(data=q1_geom, aes(fill=yes)) + ggtitle("People reporting satisfaction with life by S
```




```
# example 2
(finsec <- cws %>% get_cws(FINSEC_POP, index=TRUE) %>% cws_senate(index=TRUE))
```

```
## # A tibble: 36 x 3
##    district        n     index
##       <chr>    <dbl>     <dbl>
##  1      001 365.7938 0.7106473
##  2      002 428.7935 0.7067493
##  3      003 440.1350 0.7779820
##  4      004 470.3510 0.7861347
##  5      005 475.8358 0.8132412
```

```
##  6      006 421.2896 0.7587259
##  7      007 474.2839 0.7911406
##  8      008 381.1049 0.7934615
##  9      009 443.9627 0.8129842
## 10      010 450.3776 0.7316991
## # ... with 26 more rows
```

```
# attach to geo data
finsec <- left_join(finsec, senate, by=c("district" = "SLDUST10"))
ggplot() + geom_sf(data=finsec, aes(fill=index)) + ggtitle("Financial Security Index by State Senate Dis
```