Lecture 9:
Maps 2 of 3

April 2, 2018

Overview

Course Administration

Good, Bad and Ugly

Spatial Analysis

Maps in R

Course Administration

1. Make sure you anticipate problems in preparing policy brief
2. Project has very draconian late policies
3. To help you, next week is our in-class workshop
   - handout, slightly modified from last week's posting
   - brief summary of your own thoughts and comments on others' work
   - summary is due to google drive
4. Missing anything else from me?

Next Next Week's Good Bad and Ugly

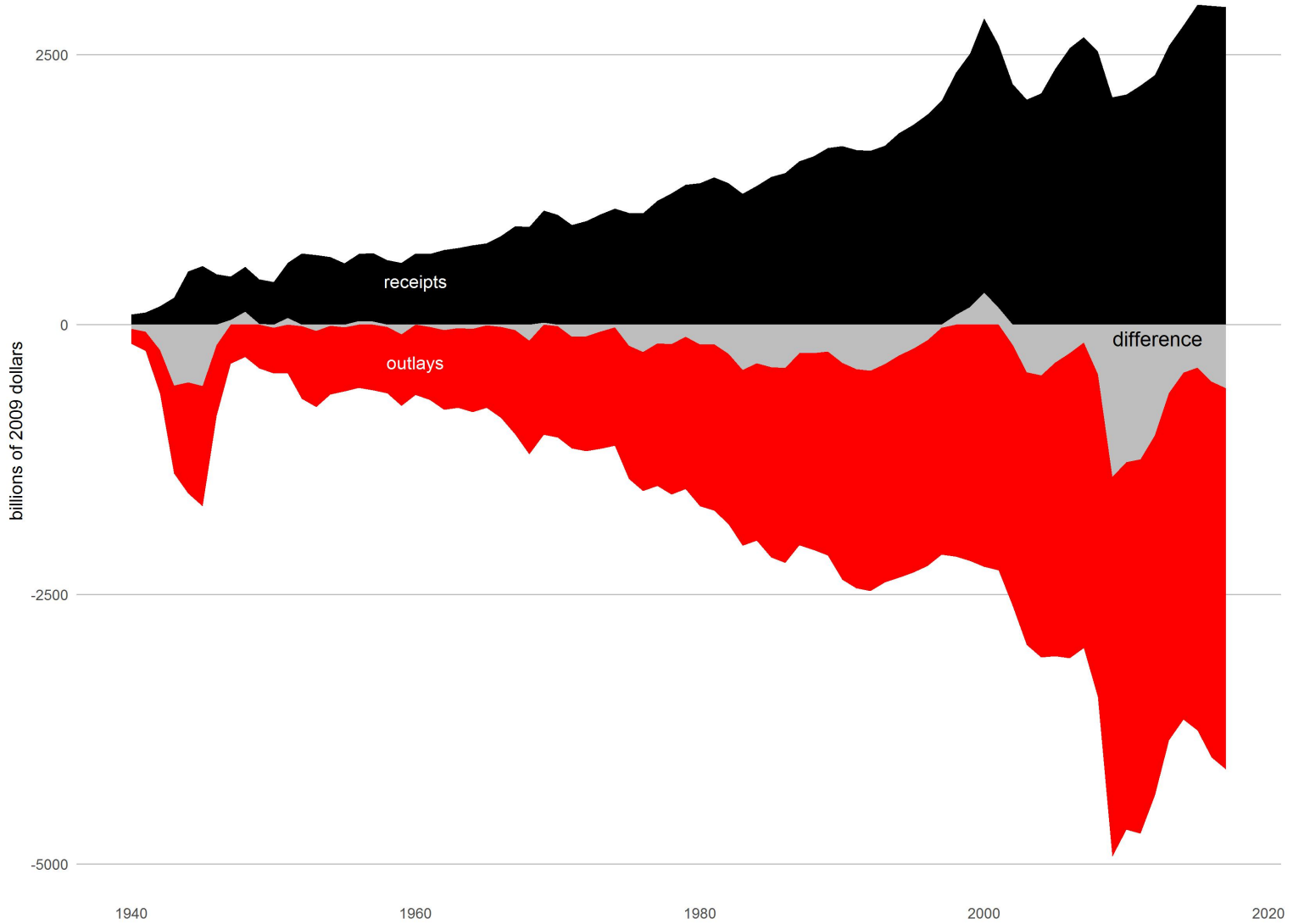
Two weeks from today (Monday), by 9 am. Earlier is ok.

- Julie Edmonds
- Amber Carter
- Azwa Saleh

Next Week's Good Bad and Ugly

- Sophie Godfrey-McKee
- Colleen McBride

My Surplus Chart



Maps: Why and How

Today

1. What you can do with digital maps
2. Save for next time: Choropleth maps

What Digital Maps Can Do For You

- Find distance from a set of points to another set of points
- Calculate area or length
- Things I've done
  - Which census tracts are in which police district?
  - How far apart are parcels of land?
  - Which parcels of land are contiguous?
  - How far is a census tract from an extinct streetcar?
  - How far is a county from the coast?

What You Can Pull From Google

- directions for a route
- by different modes of transit
- time a route takes
- location in space of address (geocoding)
- but you are limited in number of queries unless you pay!

Maps and Intersections in R

Today

A. `ggmap` package

B. Vector vs. Raster maps

C. Intersect Example

A. ggmap package

- this package pulls in "static" maps from the web
- on top of which you can put other spatial data
- can also "geocode" for you
  - geocode means find the lat and long of a point
  - but number of queries is limited
- can compute travel time distances (not as-the-crow-files distances, which are easy)
- grab a "route" from google maps

get_map() command

```
new.object <- get_map(location = address /
                        c(lon = num, lat = num) /
                        name of location,
                      source = c("google", "osm", "stamen"),
                      zoom = some number)
```

- more options than those listed, of course

Pulling in a map

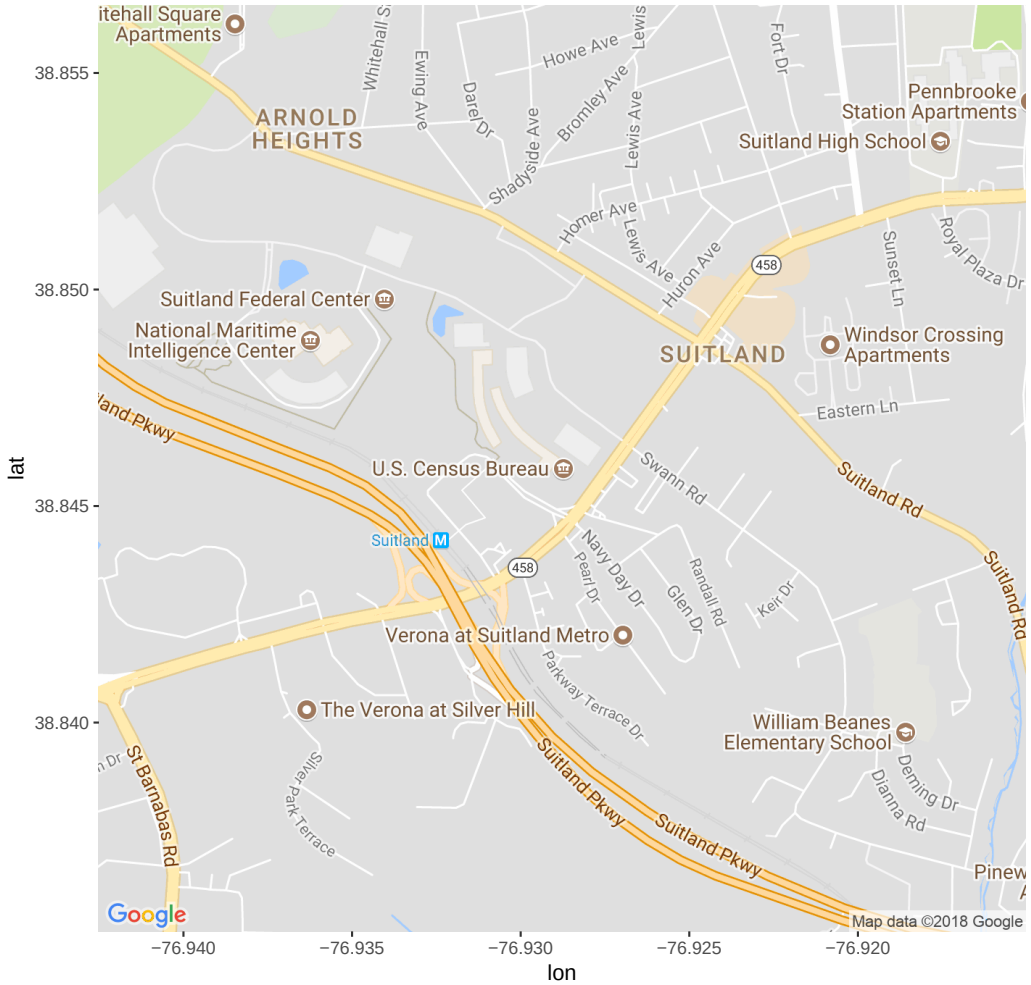
```
census.map <- get_map(location =
                  "4600 Silver Hill Road Suitland MD 20746",
                        maptype = "roadmap",
                        source = "google",
                        zoom = 15)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/stat
```

```
## Information from URL : http://maps.googleapis.com/maps/a
```

ggmap: display map from get_map()

```
ggmap(census.map)
```



ggmap can do more

- it can put ggmap plot together with other geographic data

```
new.map <- ggmap(new.object) + geom_point(some.data)
new.map <- ggmap(new.object) + stat_polygon(polygons)
new.map <- ggmap(new.object) + geom_text(some.data)
```

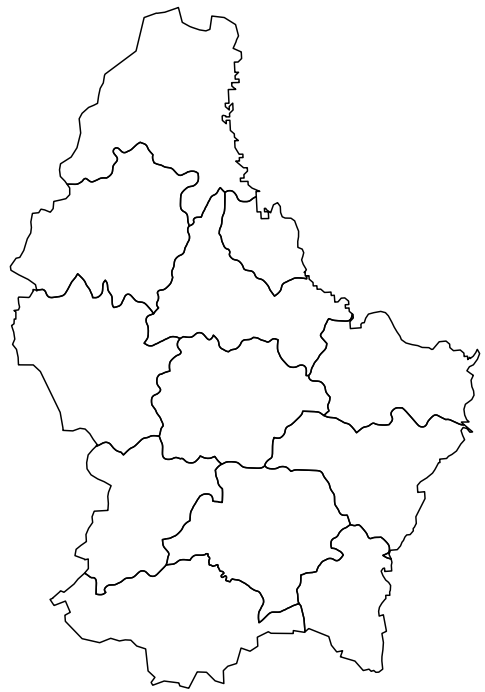
B. Vector vs Raster Maps

1. Vector maps
   - so far, we've used these types of maps
   - they are points, lines or polygons
   - things defined by points in space
   - or points in space that are connected
2. Raster maps
   - a set of colored pixels
   - `get_map()` makes these maps
   - can't move names
   - or zoom in too much
   - instead, pull a different raster image
   - can use to calculate how many pixels are in a polygon

C. Intersect

- Suppose you want to know how much one geography overlaps with another one
- For example, we will look at Luxembourg and some squares
- Example taken from here

Luxembourg



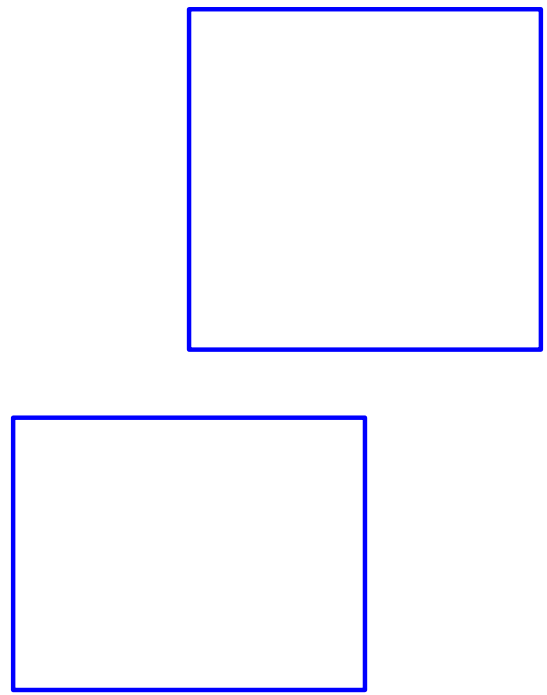
Squares

```
# make some squares
p2 <- union(as(extent(6, 6.4, 49.75, 50),
                 'SpatialPolygons'),
             as(extent(5.8, 6.2, 49.5, 49.7),
                 'SpatialPolygons'))
squares <- SpatialPolygonsDataFrame(p2,
              data.frame(field=c('x','y')),
              match.ID=F)
projection(squares) <- projection(p1)
squares@data
```

```
##   field
## 1     x
## 2     y
```

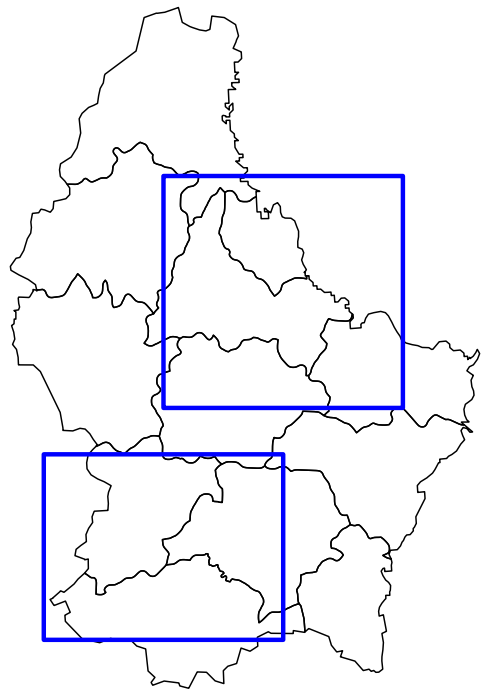
Squares in a map

```
plot(squares, border = "blue", lwd = 3)
```



Luxembourg + Squares

```
# plot the two on top of each other
plot(p1);
plot(squares, add = TRUE, border = "blue", lwd = 3)
```



Why would you want an intersection?

- Suppose you want to know how many people live in each square
- You can't do that with these data!
- But you can know the number of people in each canton that overlaps with a square
- Let's add population to the shapefile
- Thanks, Wikipedia (and you can make the map on this page)

```
pop.frame <- data.frame( p1$NAME_2,
                pop = c("17126","31819","17609","4951",
                        "15680","18007","20985","28492",
                        "45276","167955","178000","30382"))

pop.shape <- merge(p1, pop.frame,
                       by.x = c("NAME_2"),
                       by.y = c("p1.NAME_2"),
                  all = TRUE)
```

Make sure I added population correctly

```
# from the merged file
lister <- data.frame(pop.shape$NAME_2, pop.shape$pop)
lister
```

```
##    pop.shape.NAME_2 pop.shape.pop
## 1           Clervaux         17126
## 2           Diekirch         31819
## 3            Redange         17609
## 4            Vianden          4951
## 5              Wiltz         15680
## 6         Echternach         18007
## 7              Remich         20985
## 8       Grevenmacher         28492
## 9            Capellen         45276
## 10 Esch-sur-Alzette        167955
## 11       Luxembourg        178000
## 12            Mersch         30382
```

Now intersect

```
# intersect them
sq.int <- raster::intersect(pop.shape, squares)
int.names <- names(sq.int)
int.names
```

```
## [1] "NAME_2" "ID_1"   "NAME_1" "ID_2"   "AREA"   "pop"
```

```
look.at.it <- data.frame(NAME2 = sq.int$NAME_2,
                         ID_2 = sq.int$ID_2, pop = sq.int$p
                         field = sq.int$field)
```

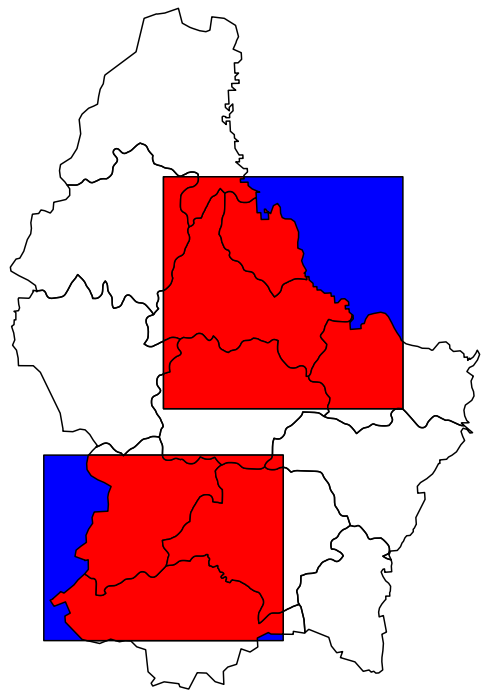
Look at the data: Why 14 obs?

```
look.at.it
```

```
##                NAME2 ID_2    pop field
## 1           Clervaux    1  17126     x
## 2           Diekirch    2  31819     x
## 3            Redange    3  17609     x
## 4            Redange    3  17609     y
## 5            Vianden    4   4951     x
## 6              Wiltz    5  15680     x
## 7         Echternach    6  18007     x
## 8       Grevenmacher   12  28492     x
## 9       Grevenmacher   12  28492     y
## 10          Capellen    8  45276     y
## 11 Esch-sur-Alzette    9 167955     y
## 12       Luxembourg   10 178000     y
## 13            Mersch   11  30382     x
## 14            Mersch   11  30382     y
```

See what the intersection looks like

```
plot(pop.shape); plot(squares, add = TRUE, col = "blue");
plot(sq.int, add = TRUE, col = "red")
```



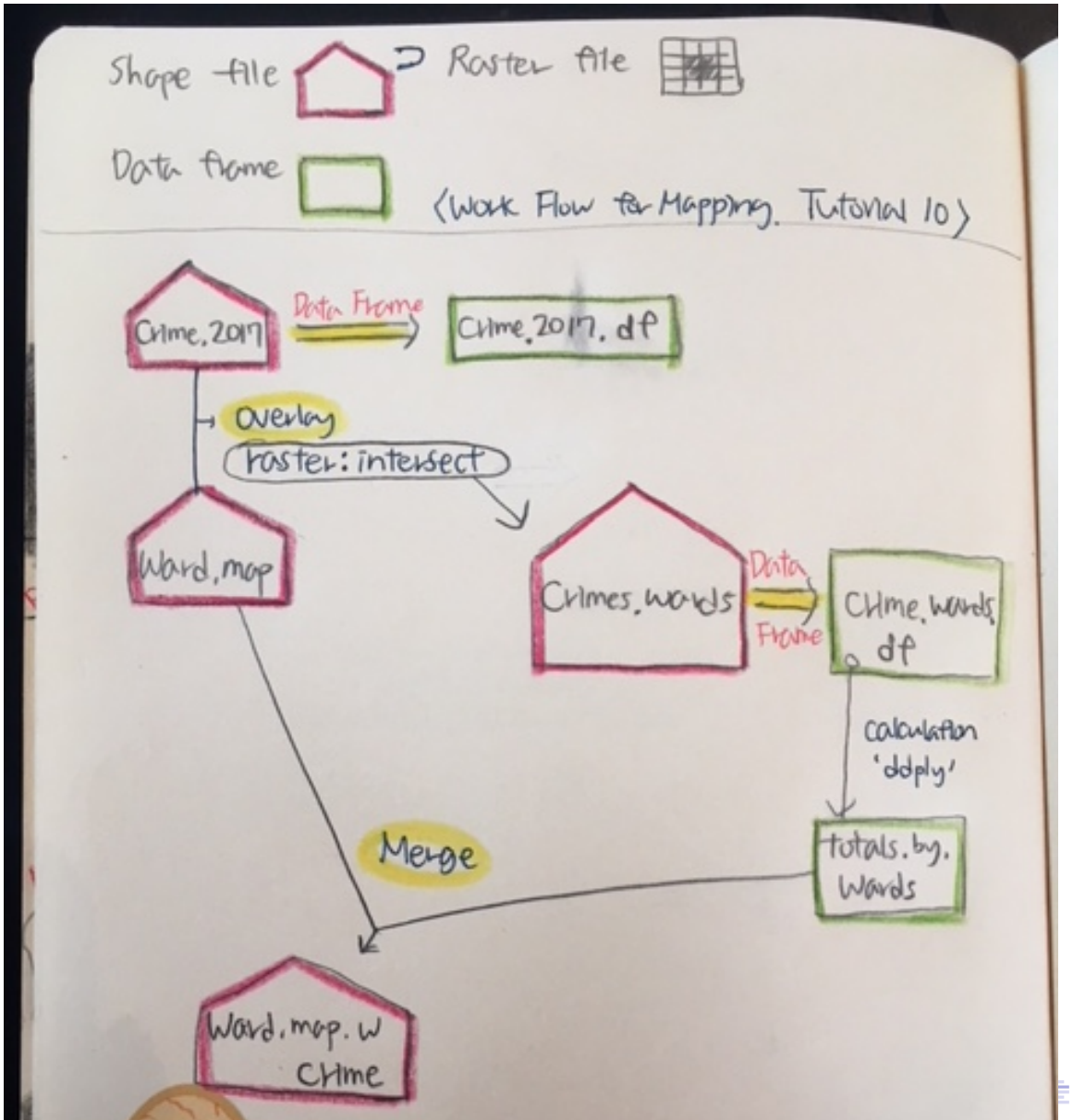
How should area of new polygons relate to old ones?

```
sq.int$int.area <- area(sq.int) / 1000000
```

```
outer
```

```
##          int.area AREA           NAME_2 field
## 1   24.572263541  312          Clervaux     x
## 2  209.565929415  218          Diekirch     x
## 3    5.714943365  259           Redange     x
## 4    0.005311882  259           Redange     y
## 5   76.200409156   76           Vianden     x
## 6   31.015468891  263             Wiltz     x
## 7  101.945521274  188       Echternach     x
## 8    0.007106824  210     Grevenmacher     x
## 9    2.973231980  210     Grevenmacher     y
## 10 175.270207897  185          Capellen     y
## 11 188.656204146  251 Esch-sur-Alzette     y
## 12 153.822938405  237       Luxembourg     y
## 13 132.174792299  233            Mersch     x
```

Workflow for Tutorial: Thank you, Rosa!



Try Today's Tutorial

- Ask questions if the command doesn't make sense
- Go forth!

Next Lecture

- Next week: in-class workshop
- Following week: maps 3 of 3
- Lectures 13 and 14 are presentations