Making a Hotspot Map

Greg Ridgeway (gridge@upenn.edu)
Ruth Moyer (moyruth@upenn.edu)

July 28, 2018

Introduction

Now that we've learned how to work with SQL, we can make a hotspot map with a large dataset. The concept of determining the geographic locations where crime is most intense - crime hotspots - is very important to criminological theory as well as to the very practical question of where police officers should be deployed.

Here, we will make hotspot maps with our Chicago crime data (which requires use of SQL). Of course, if you had a small dataset, you could use these same methods to make a hotspot map using only R.

Setting up the data

Let's first load up the `sqldf` library and reconnect to our Chicago crime database.

```
library(sqldf)
```

```
Loading required package: gsubfn
```

```
Loading required package: proto
```

```
Loading required package: RSQLite
```

```
con <- dbConnect(SQLite(), dbname="chicagocrime.db")
```

Let's run a SQL query to extract the two columns, `Latitude` and `Longitude` from our crime database, creating a datafame with one row for each crime incident in the dataset.

```
res <- dbSendQuery(con, "
                   SELECT Latitude, Longitude
                   FROM crime
                   WHERE Latitude  IS NOT NULL AND
                         Longitude IS NOT NULL")
a <- fetch(res, n = -1)
dbClearResult(res)
```

How would your query differ if you wanted to do a hotspot map of only assaults?

```
res <- dbSendQuery(con, "
                   SELECT Latitude,Longitude
                   FROM crime
                   WHERE [Primary.Type]='ASSAULT' AND
```

```
                          Latitude  IS NOT NULL AND
                          Longitude IS NOT NULL")
b <- fetch(res, n = -1)
dbClearResult(res)
```

Creating the maps

We'll be using two packages to make our maps, `ggmap` and `hexbin`, so be sure you have those installed.

```
library(ggmap)
```

```
Loading required package: ggplot2
```

```
library(hexbin)
```

Next, we'll need to get a map of Chicago. The `ggmap` package enables us to obtain a Google Maps map. Try changing the zoom level. Experiment with moving the legend.

If you type `?get_map` or type `?ggmap`, you'll see a range of different options for your map. Instead of doing `maptype="satellite"` try `maptype="hybrid"`.

```
chicago.map <- ggmap(get_map("Chicago",
                             zoom = 11,
                             maptype = "satellite"),
                     extent="device",
                     legend="topright")
```

```
Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Chicago&zoom=11&size=640x640
```

```
Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Chicago&sensor=f
```

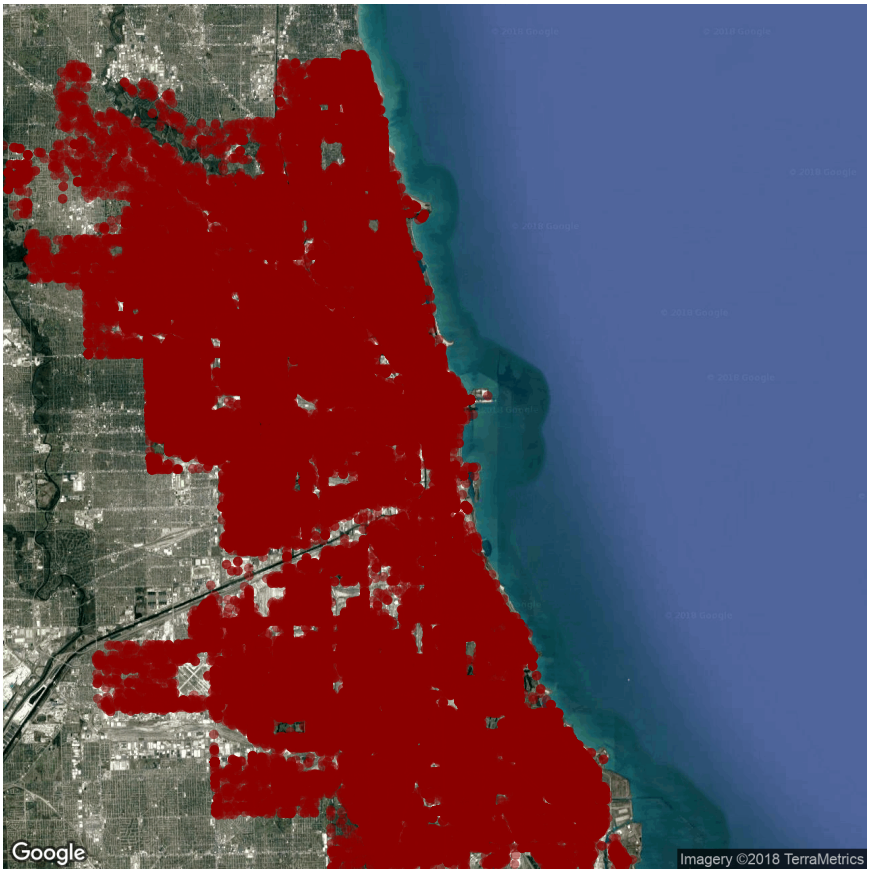
```
Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
instead
```

```
chicago.map
```

Let's plot as points on the Chicago map, a random sample of 100,000 rows from our dataset.

```
i <- sample(1:nrow(a),100000)
chicago.map +
  geom_point(aes(x=Longitude,y=Latitude),
             data=a[i,],
             alpha=0.5,
             color="darkred",
             size = 1)
```

```
Warning: Removed 7429 rows containing missing values (geom_point).
```



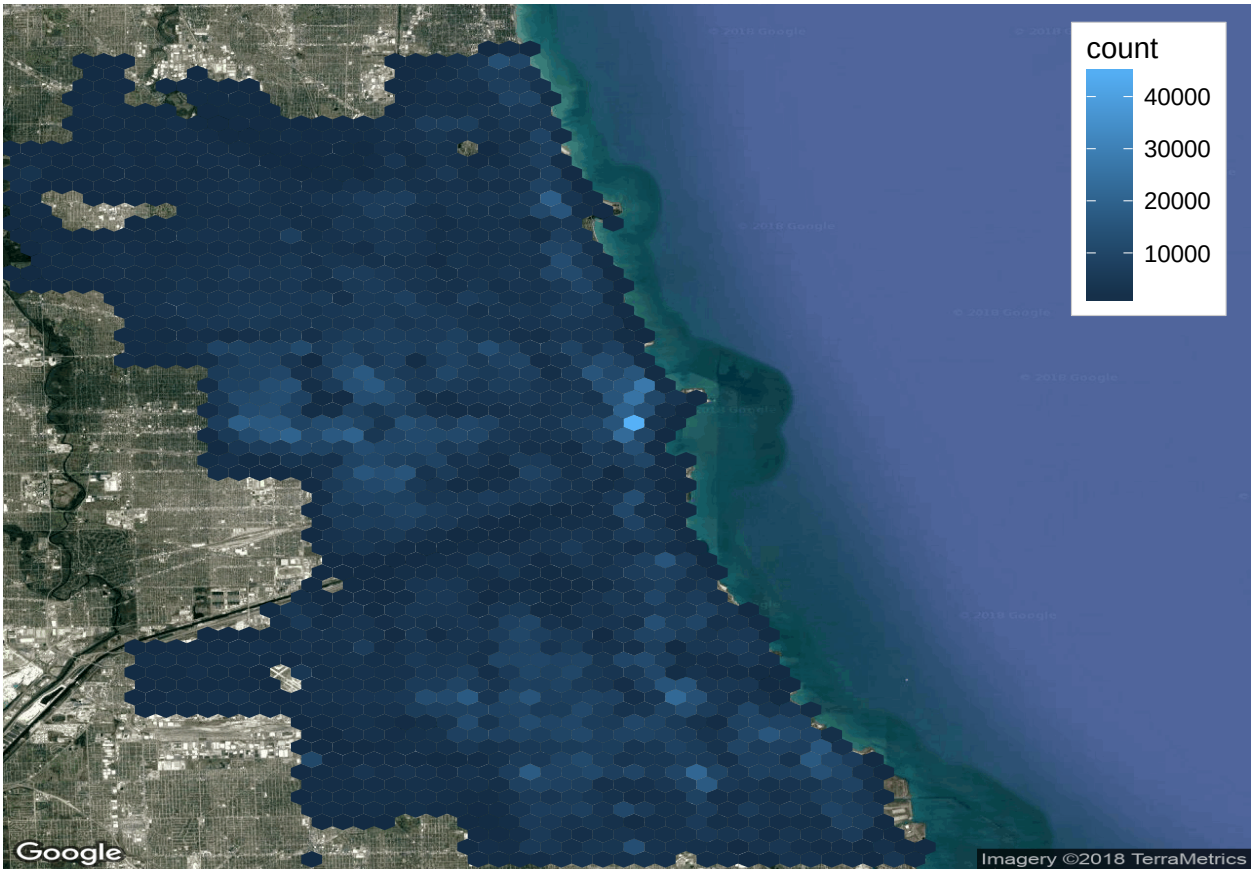
It doesn't look terribly useful because it is just a giant red splotch. These are all the points where a crime occurred. There's no density to show us the high crime or low crime areas. But, we're making process with getting points onto a map!

Let's instead try plotting hexagonal bins. Why hexagons? It turns out that squares tend to have problems in the corners. The precision is low there and our eye tends to get drawn to the parallel lines the square grid makes. Ideally, we want to use a shape that has a small perimeter-to-area ratio. Circles have the smallest perimeter-to-area ratio, but we can't use circles to tile over the map. Hexagons fall in between squares and circles.

```
chicago.map +
  coord_cartesian() +
  stat_binhex(aes(x=Longitude, y=Latitude),
              bins=60,
              data = a)
```

```
Warning: Removed 487065 rows containing non-finite values (stat_binhex).
```

```
Warning: Removed 21 rows containing missing values (geom_hex).
```

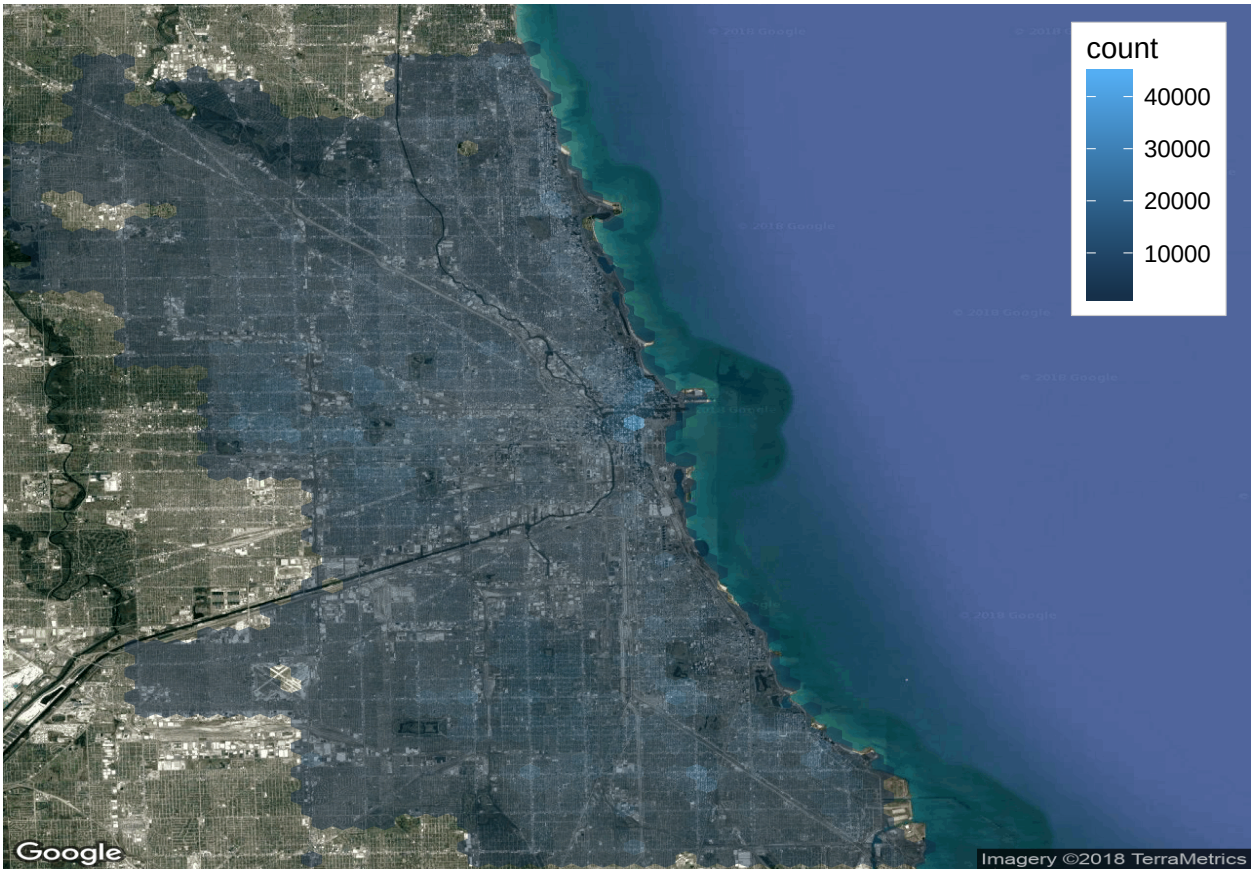


Or try the following. Setting `alpha=2/4` makes the bins a little more transparent. Try comparing with `alpha=1/4` or `alpha=3/4`.

```
chicago.map +
  coord_cartesian() +
  stat_binhex(aes(x=Longitude, y=Latitude),
              bins=60,
              alpha = 2/4,
              data = a)
```

```
Warning: Removed 487065 rows containing non-finite values (stat_binhex).
```

```
Warning: Removed 21 rows containing missing values (geom_hex).
```

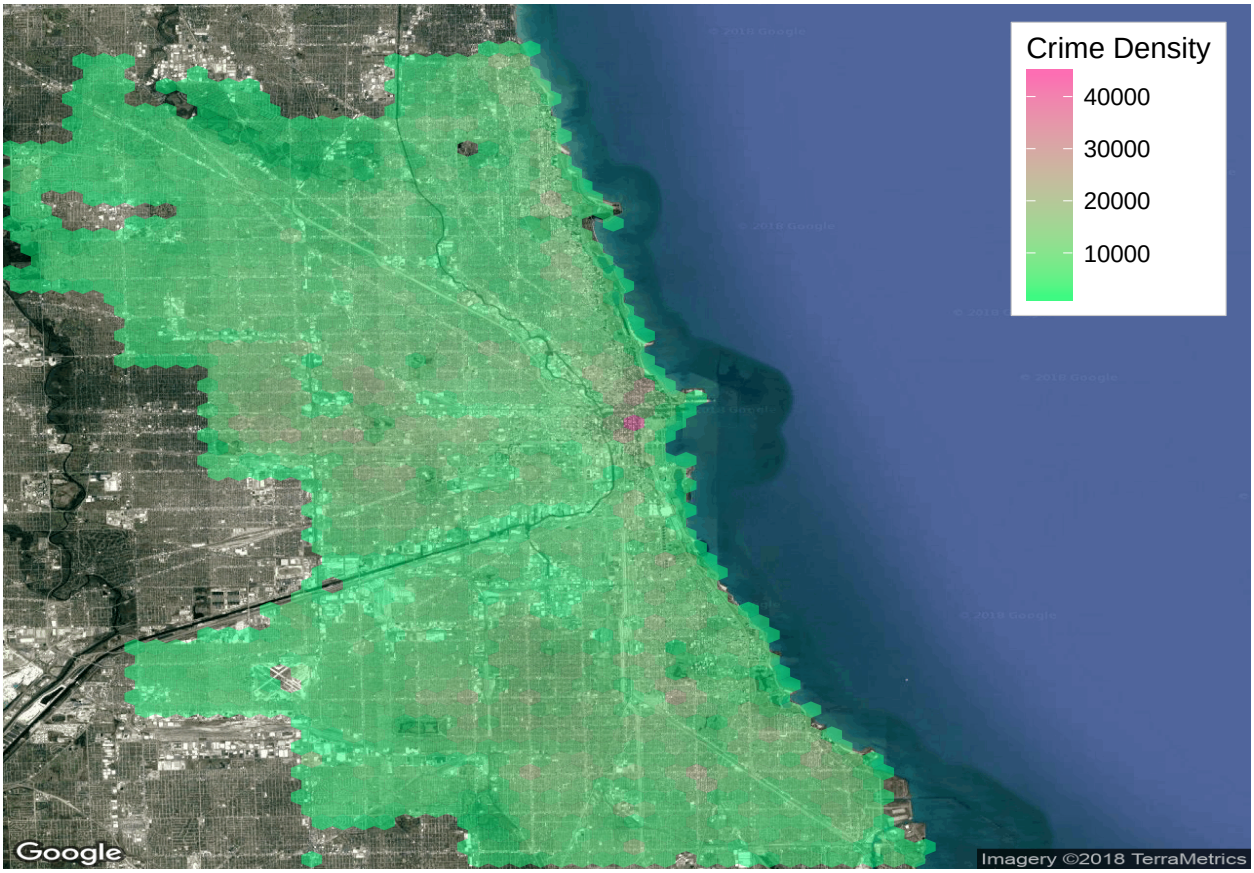


The previous map was a default monochromatic color (blue). You can change the color gradient. To see what colors are available, type `colors()`.

```
chicago.map +
  coord_cartesian() +
  stat_binhex(aes(x=Longitude, y=Latitude),
              bins=60,
              alpha = 2/4,
              data = a) +
  scale_fill_gradient('Crime Density',
                      low="springgreen",
                      high="hotpink")
```

```
Warning: Removed 487065 rows containing non-finite values (stat_binhex).
```

```
Warning: Removed 21 rows containing missing values (geom_hex).
```



You can also do a map that shows the high crime areas in a way that is akin to elevation on a topographic map. The more concentrated the concentric areas, the higher the crime (or, in the case of a topographic map, the higher the terrain). Note that a plot using `stat_density2d()` requires a lot of memory. That's why it's a good idea to use a sample of cases.

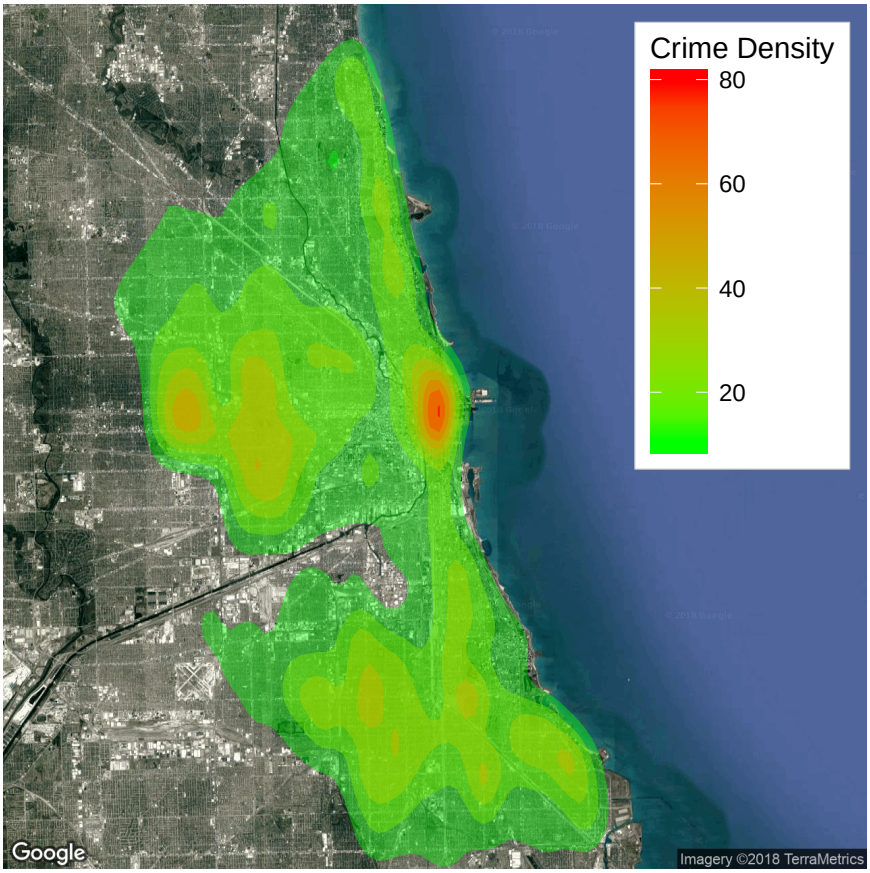
```
i <- sample(1:nrow(a),100000)
chicago.map +
  stat_density2d(aes(x=Longitude,y=Latitude),
                 bins = 5,
                 data = a[i,],
                 geom = 'density2d',
                 col='white')
```

```
Warning: Removed 7363 rows containing non-finite values (stat_density2d).
```

Let's make a hotspot map with a gradient, red for high crime and green for low crime.

```
chicago.map +
  stat_density2d(aes(x=Longitude, y=Latitude,
                     fill=..level..,
                     alpha=..level..),
                 data = a[i,],
                 geom = 'polygon') +
  scale_fill_gradient('Crime Density',
                      low="green",
                      high="red") +
  scale_alpha(range = c(.4, .75),
              guide = FALSE) +
  guides(fill = guide_colorbar(barwidth = 1.5, barheight = 10))
```

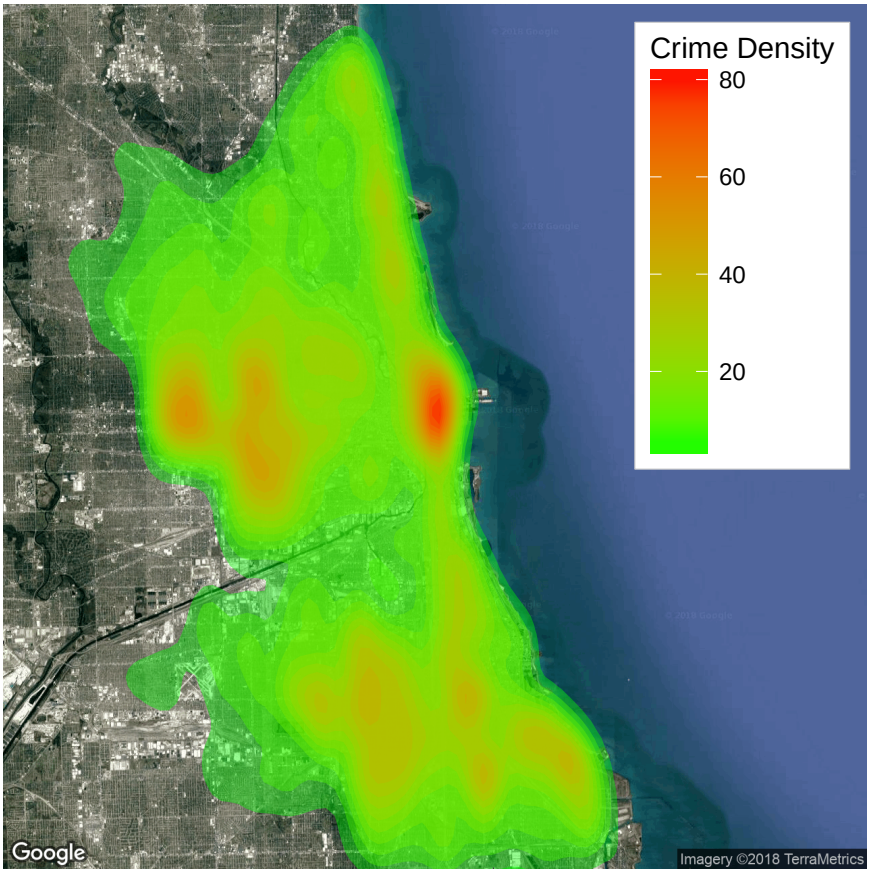
```
Warning: Removed 7363 rows containing non-finite values (stat_density2d).
```



We can use `bins` to control the number of levels of colors plotted.

```
chicago.map +
  stat_density2d(aes(x=Longitude, y=Latitude,
                     fill=..level..,
                     alpha=..level..),
                 bins = 20,
                 data = a[i,],
                 geom = 'polygon') +
  scale_fill_gradient('Crime Density',
                      low="green",
                      high="red") +
  scale_alpha(range = c(.4, .75),
              guide = FALSE) +
  guides(fill = guide_colorbar(barwidth = 1.5, barheight = 10))
```

```
Warning: Removed 7363 rows containing non-finite values (stat_density2d).
```



Let's do hotspot maps with just motor vehicle theft, and let's make a map for each year between 2005 and 2014. Note the line with the SQL function `SUBSTR()`. The `SUBSTR()` command helps us grab a section of a string. Here's an example. Take a look at the `Date` column in our initial dataset.

```
res <- dbSendQuery(con, "SELECT Date
                         FROM crime
                         LIMIT 5")
fetch(res, n = -1)
dbClearResult(res)
```

```
                   Date
1 03/18/2015 07:44:00 PM
2 03/18/2015 11:00:00 PM
3 03/18/2015 10:45:00 PM
4 03/18/2015 10:30:00 PM
5 03/18/2015 09:00:00 PM
```

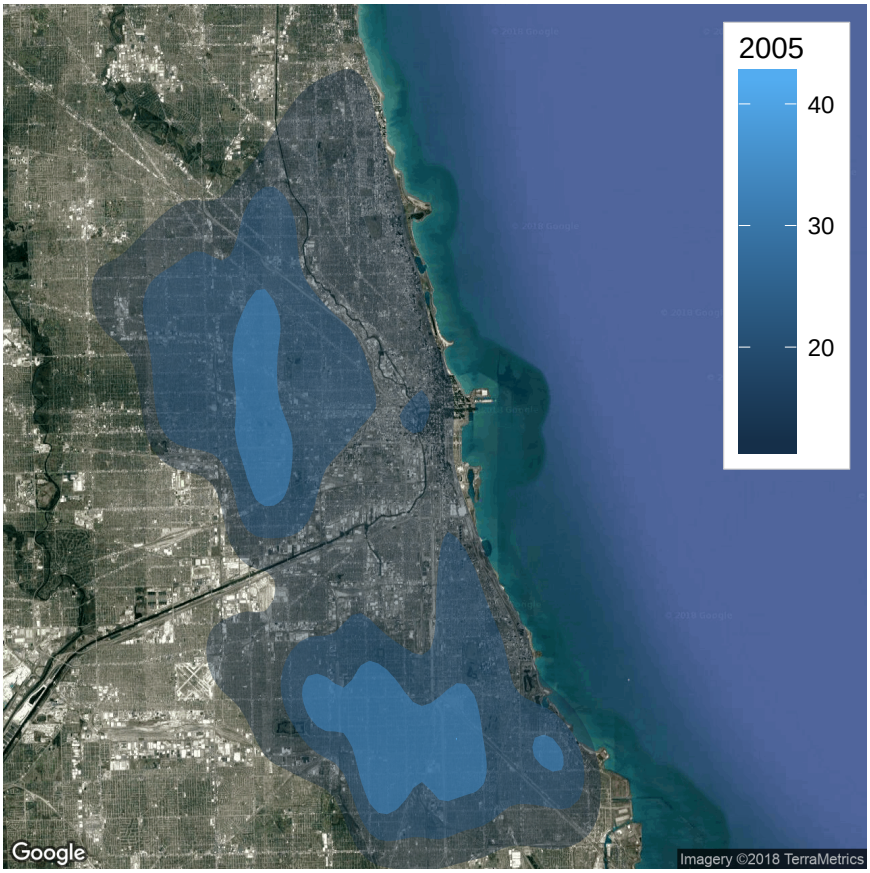
We want only the year part of these dates. The year starts with the 7th character and has a length of 4 characters. So to extract just the year we select `SUBSTR(Date, 7, 4)`.

```
res <- dbSendQuery(con, "
                    SELECT Latitude,
                           Longitude,
                           SUBSTR(Date,7,4) AS year
                    FROM crime
```

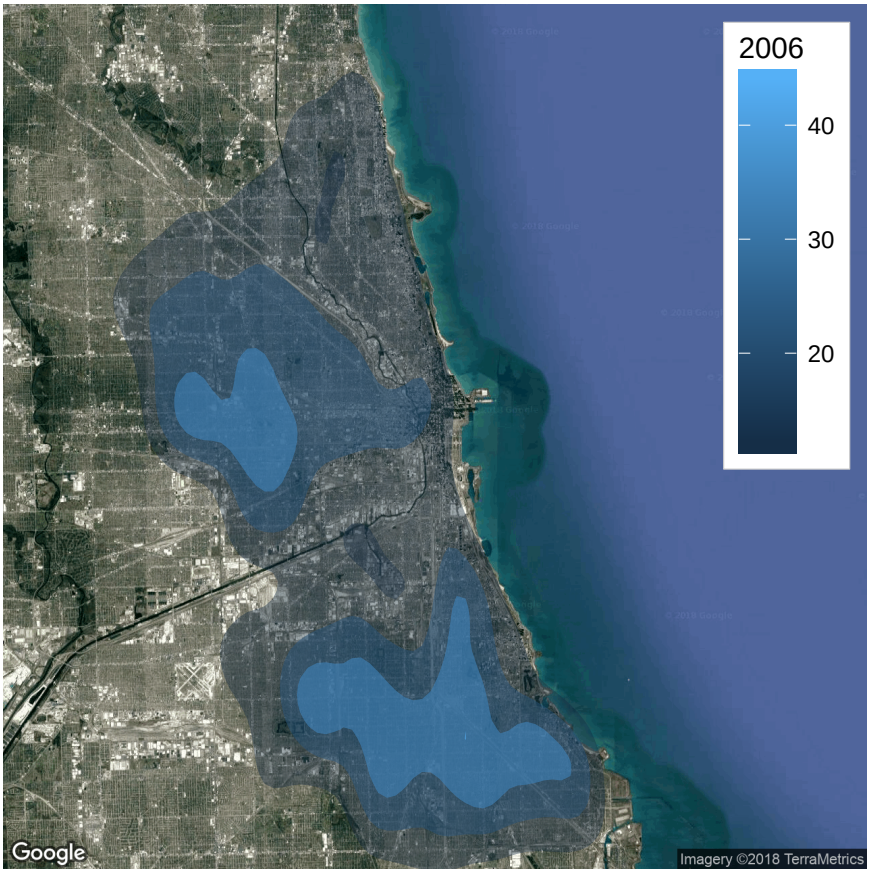
```
                    WHERE [Primary.Type]='MOTOR VEHICLE THEFT' AND
                          Latitude  IS NOT NULL AND
                          Longitude IS NOT NULL")
b <- fetch(res, n = -1)
dbClearResult(res)

for(year0 in 2005:2017)
{
  print(
    chicago.map +
       stat_density2d(aes(x=Longitude, y=Latitude,
                          fill=..level..,
                          alpha=..level..),
                      size = 2,
                      bins = 4,
                      geom = 'polygon',
                      data = subset(b,year==year0)) +
       scale_fill_gradient(year0) +
       scale_alpha(range = c(.4, .75), guide = FALSE) +
       guides(fill = guide_colorbar(barwidth = 1.5, barheight = 10))
  )
  # Sys.sleep(5) # uncomment this if you want a 5 second pause between plots
}
```

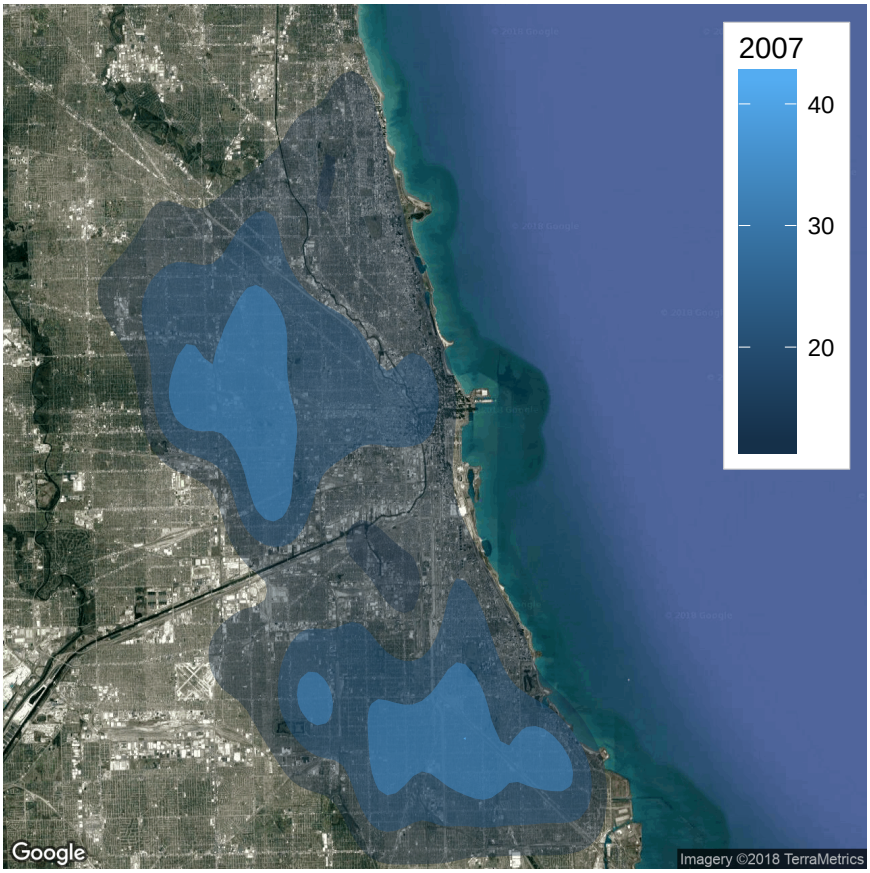
```
Warning: Removed 1650 rows containing non-finite values (stat_density2d).
```



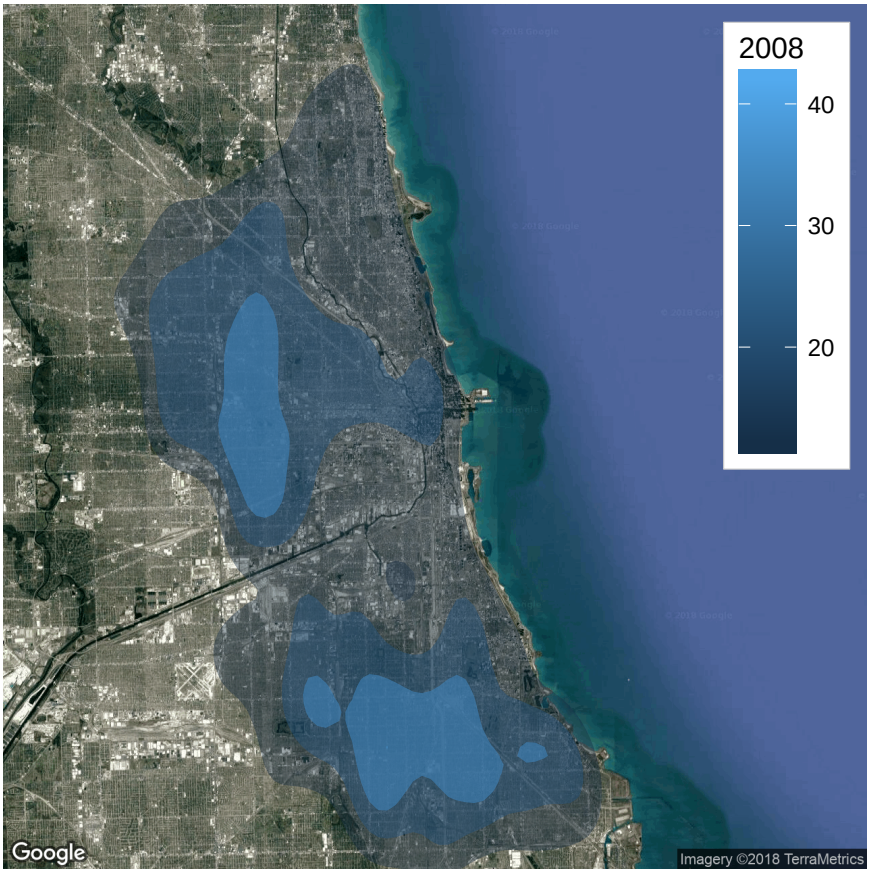

```
Warning: Removed 1589 rows containing non-finite values (stat_density2d).
```



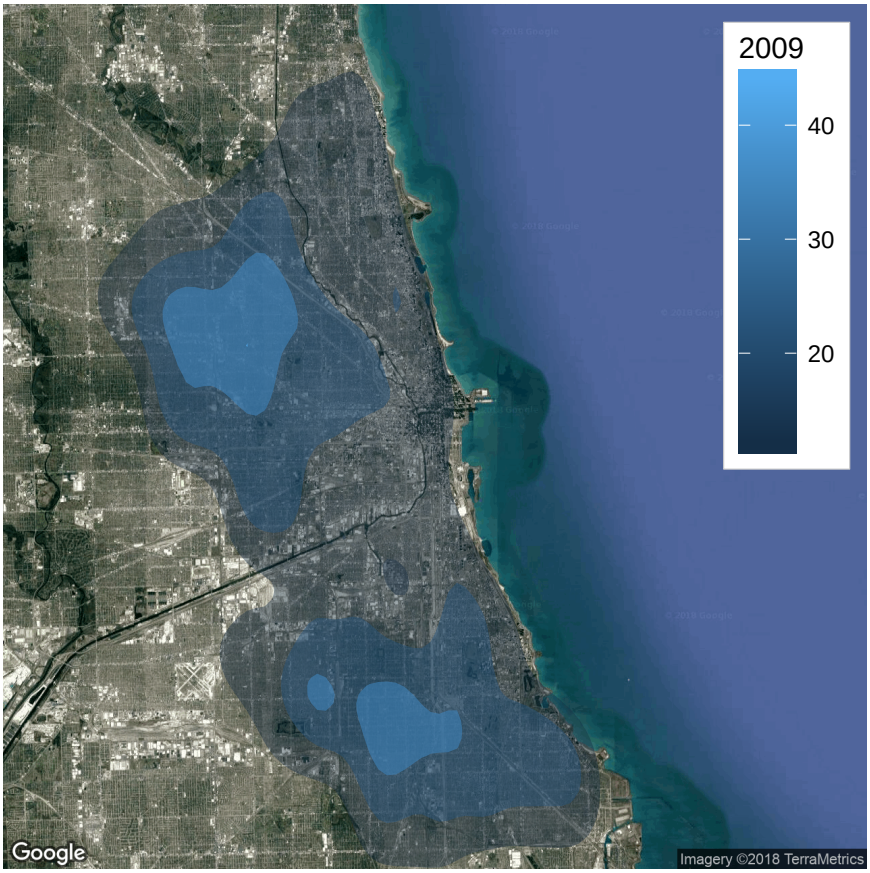

```
Warning: Removed 1480 rows containing non-finite values (stat_density2d).
```



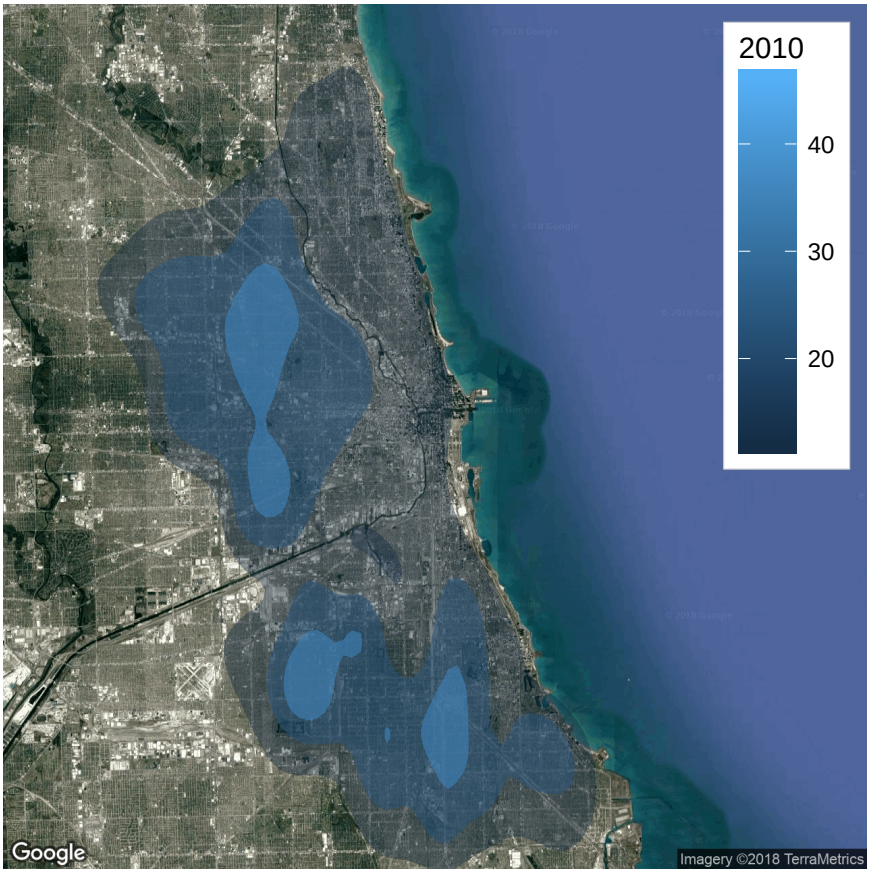

```
Warning: Removed 1359 rows containing non-finite values (stat_density2d).
```



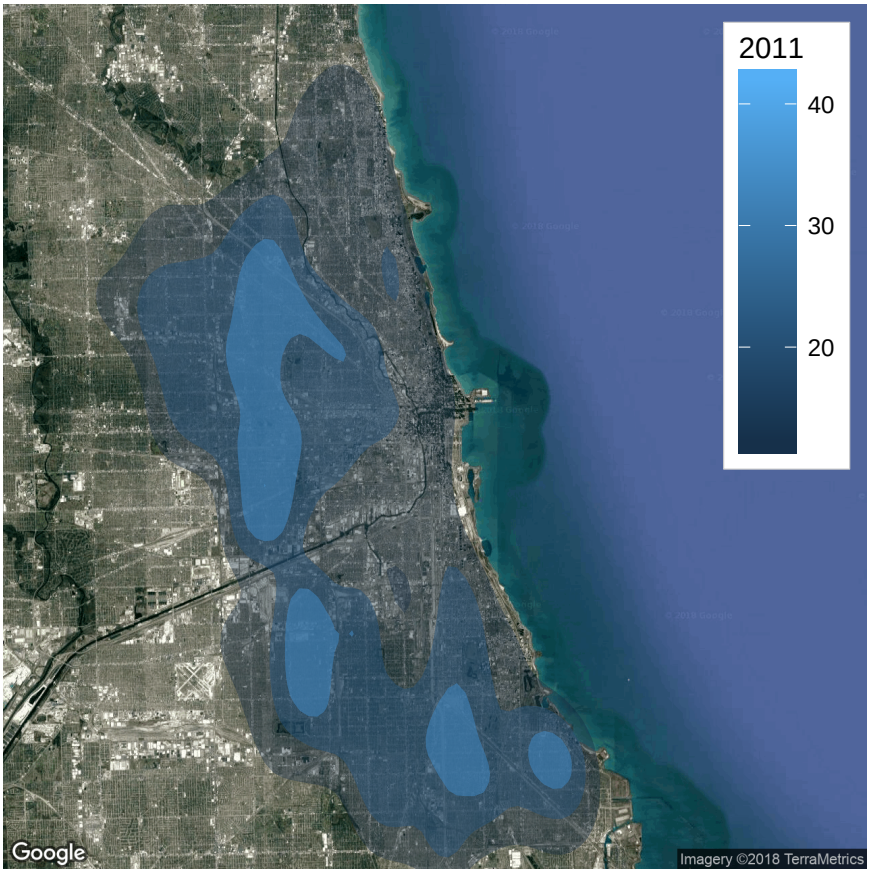

```
Warning: Removed 1080 rows containing non-finite values (stat_density2d).
```



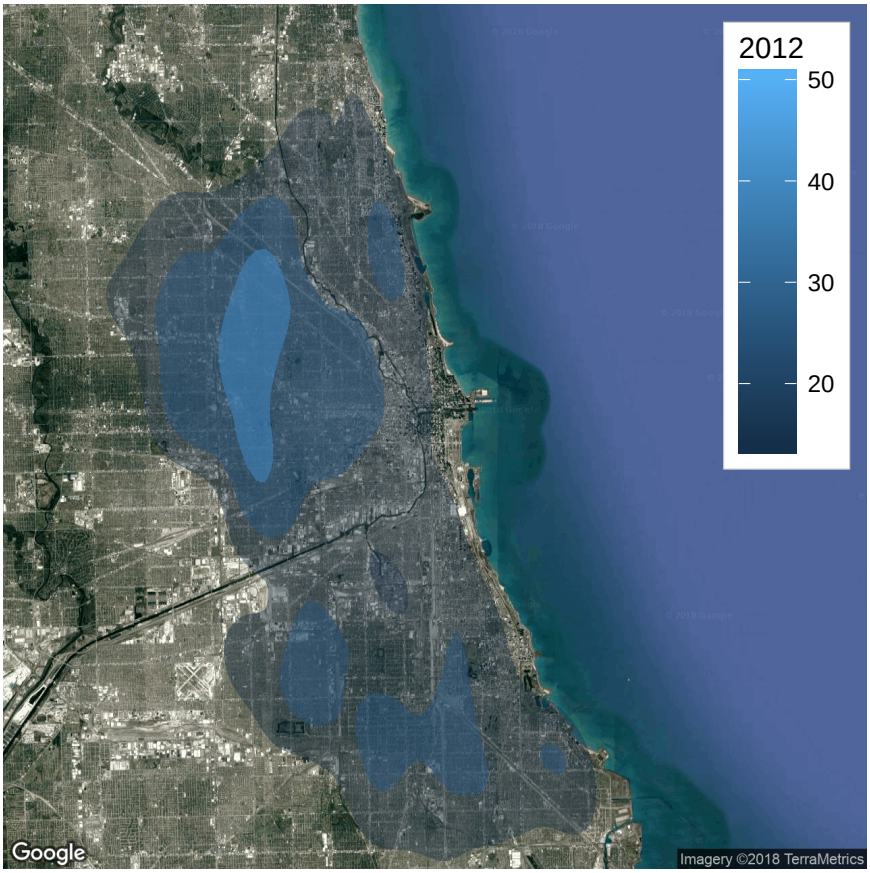

```
Warning: Removed 1291 rows containing non-finite values (stat_density2d).
```



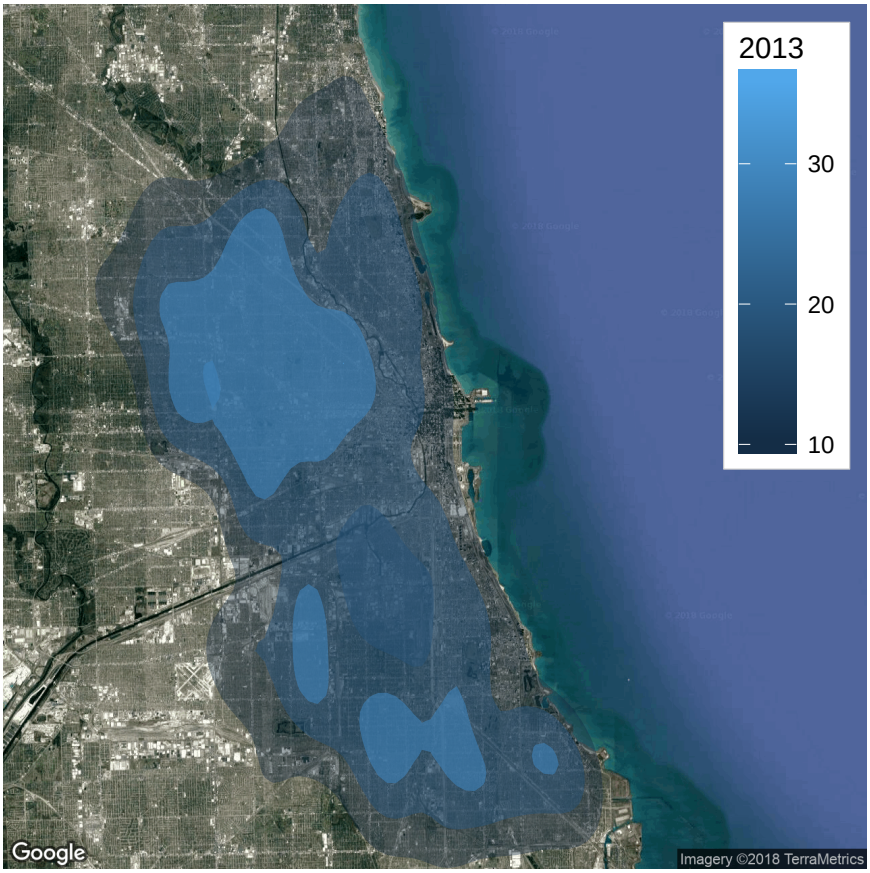

```
Warning: Removed 1375 rows containing non-finite values (stat_density2d).
```



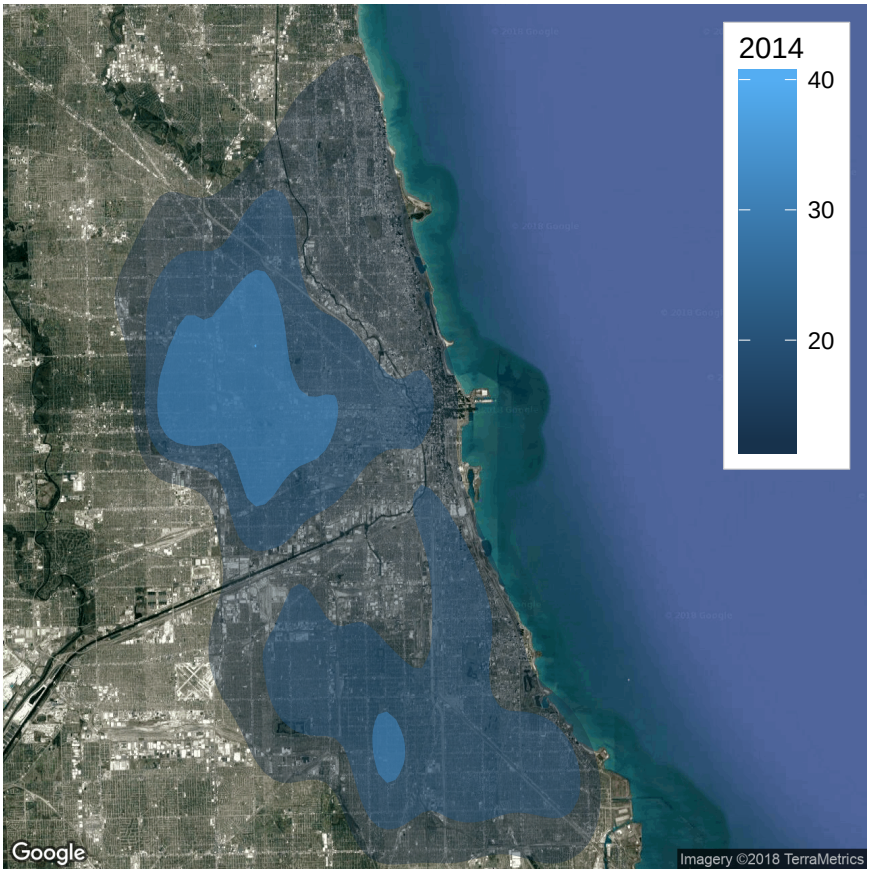

```
Warning: Removed 1064 rows containing non-finite values (stat_density2d).
```



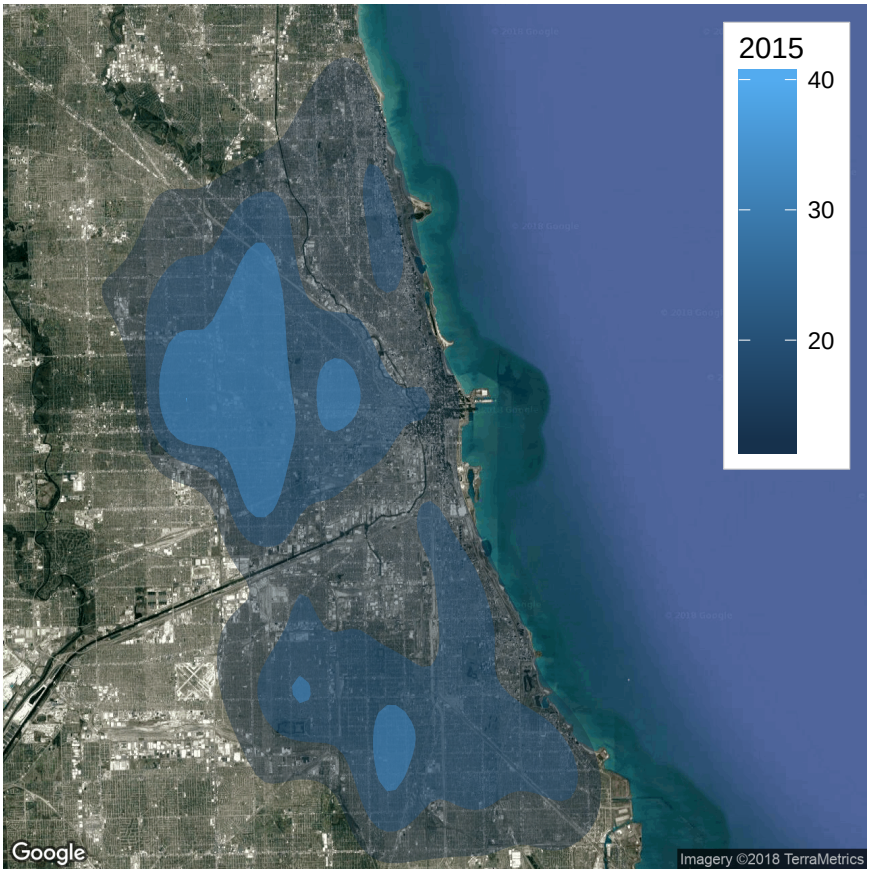

```
Warning: Removed 822 rows containing non-finite values (stat_density2d).
```



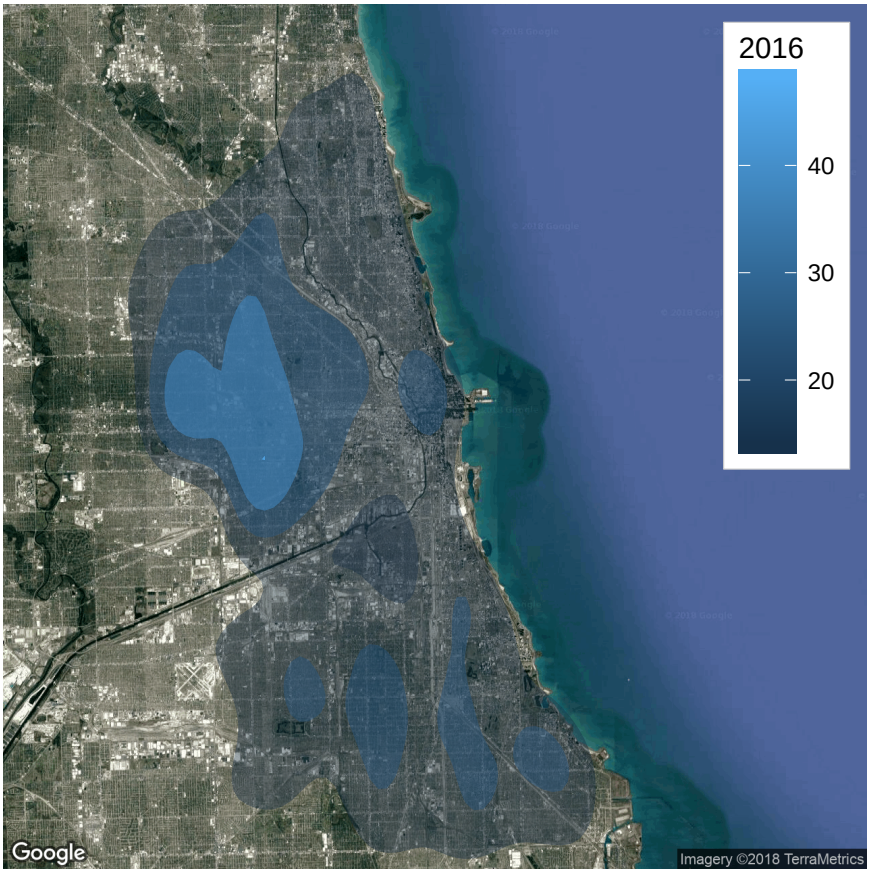

```
Warning: Removed 672 rows containing non-finite values (stat_density2d).
```



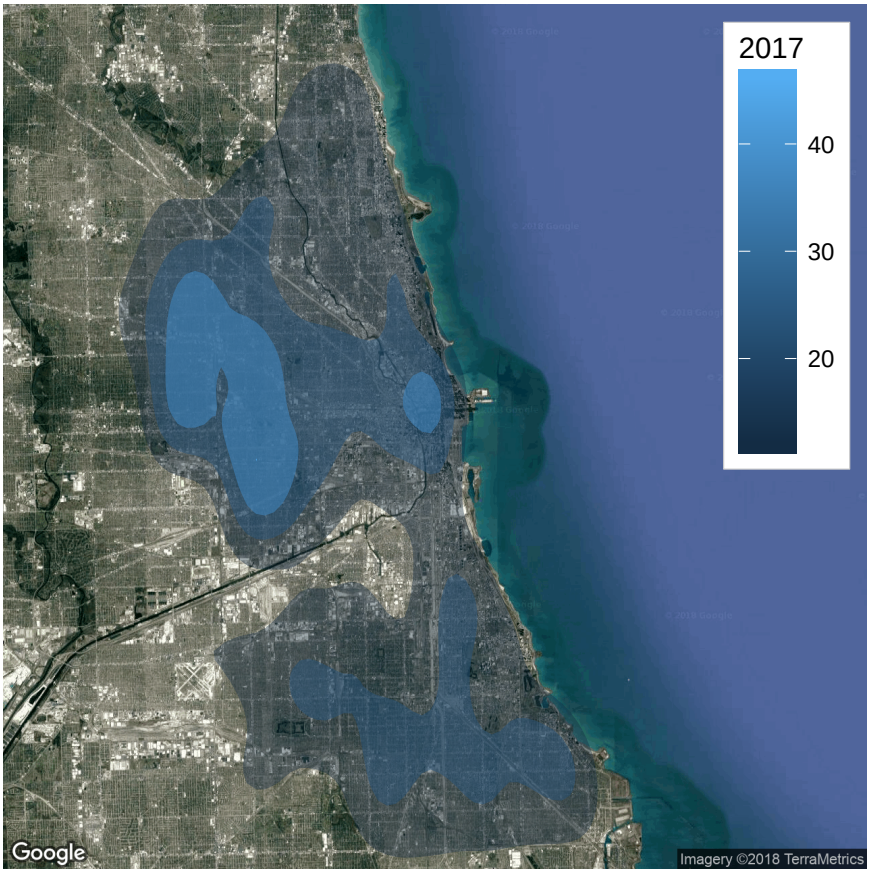

```
Warning: Removed 615 rows containing non-finite values (stat_density2d).
```




```
Warning: Removed 752 rows containing non-finite values (stat_density2d).
```




```
Warning: Removed 722 rows containing non-finite values (stat_density2d).
```



Many cities other than Chicago have accessible incident-level data so that you can try to make a hotspot map. Cities include Jersey City, Philadelphia, Los Angeles, Seattle, San Francisco, Baltimore, and Washington, DC. To work with the data from some of these cities, it's necessary to learn some tricks to clean up the dataset.