Using `ggmap` to Plot San Francisco Housing Prices

Nate Magee

October 31, 2017

The aim of this post is to give a brief introduction to how the `ggmap` library can be used to supplement data visualization for geographic regions. `ggmap` can be easily used to make calls to the Google Maps API, a vast resource of geographic data — such as street names, elevation features, and landmarks — to aid in plotting detailed maps in line with the standard Grammar of Graphics. This comes in particularly handy when working with spatial data, which is omnipresent in the world of data journalism and reporting. Visualization of crime, weather patterns, cost of living information: these are all applications to which a tool such as `ggmap` can be crucial. The easy-to-use `ggmap` workflow first calls `get_map()` to make a call to the Google Maps API, taking in location values and with a zoom level, alongside several other optional parameters. After the map has been retrieved and generated, the user can call `ggmap()` on the object to actually visualize the region in question.

> Conveniently, `ggmap` objects follow the standard Grammar of Graphics used in `ggplot`, making it possible to learn how to use the tool simply and quickly if you already have knowledge of the latter.

To begin working with the API, we'll first gather some housing data for the San Francisco Bay Area. This data will include columns for longitude and latitude, property price, and other useful values that will benefit from the `ggmap` capabilities. The ultimate goal of the following tutorial will be to visualize how San Francisco property prices change in different neighborhoods of the city. Let's first collect our data.

Data Collection and Initial Analysis

The dataset I'll be working with is aggregated from Zillow, and includes homes purchased in San Francisco from 2013-2015. I'll use `download.file()` to directly pull it off of github, and then store it as a dataframe called `homes` in my R environment. I chose this dataset particularly because of it detailed spatial data — most housing data I was able to find included only an address, which would have created additional steps in the visualization process, as I likely would have had to figure out a way to convert addresses into coordinates or place each individual property into an approximate bin, which would have made my final analysis less precise.

```
download.file("https://raw.githubusercontent.com/RuiChang123/Regression_for_house_price_estimation/master/f
              destfile="housing-data.csv")

homes <- data.frame(read.csv('housing-data.csv'))
```

Examining some of the column names in `homes` should give you a good idea of what type of data it includes. Though we will only be using a small portion of these attributes, it's always good to know what your dataset includes for later analysis. Also note the sufficient size of the dataset: over 11000 homes.

```
# retrieve the column names in homes
attributes(homes)$names
```

```
##  [1] "X"             "address"      "info"         "z_address"
##  [5] "bathrooms"     "bedrooms"     "finishedsqft" "lastsolddate"
##  [9] "lastsoldprice" "latitude"     "longitude"    "neighborhood"
## [13] "totalrooms"    "usecode"      "yearbuilt"    "zestimate"
## [17] "zindexvalue"   "zipcode"      "zpid"
```

```
# check how many rows there are
nrow(homes)
```

```
## [1] 11330
```

Now, for the purpose of exploring how location affects price, we'll cut our columns down considerably to include just longitude, latitude, and the purchase price at the time the house was last sold.

```
# cut homes down to include just location and price
home_prices <- homes[, c("longitude", "latitude", "lastsoldprice")]

head(home_prices, 5)
```

```
##   longitude latitude lastsoldprice
## 1 -122.4126 37.77871       1300000
## 2 -122.3934 37.77764        750000
## 3 -122.3965 37.75920       1495000
## 4 -122.3968 37.76189       2700000
## 5 -122.4135 37.74079       1530000
```

Gauging a relationship without `ggmap`

Before we get into the spatial plotting, we'll first try to see if there's a relationship between location and price by just plotting the data regularly. To accomplish this, we'll need the standard `ggplot2` package.

```
library("ggplot2")
```

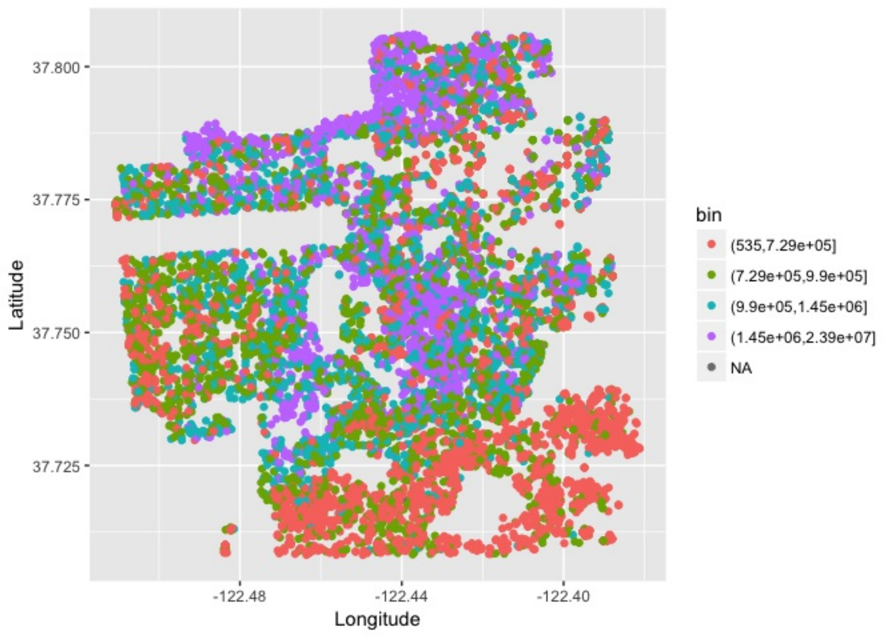
We'll use the base R `cut` function to separate the homes into bins based on quantile values. These bin values can then be used to roughly judge how expensive a property is compared to other properties. Using this scale, a home that lands in the highest-valued bin is in the top 25% of all homes by price point, whereas a property in the lowest bin finds itself among the cheapest quarter.

```
# cut the price vector into bins based on quantile
bins <- cut(home_prices$lastsoldprice, quantile(home_prices$lastsoldprice))

# create a new data frame with `bins` as a column
home_prices_bins <- data.frame(home_prices, "bin" = bins)

head(home_prices_bins, 5)
```

```
##   longitude latitude lastsoldprice                 bin
## 1 -122.4126 37.77871       1300000  (9.9e+05,1.45e+06]
## 2 -122.3934 37.77764        750000  (7.29e+05,9.9e+05]
## 3 -122.3965 37.75920       1495000 (1.45e+06,2.39e+07]
## 4 -122.3968 37.76189       2700000 (1.45e+06,2.39e+07]
## 5 -122.4135 37.74079       1530000 (1.45e+06,2.39e+07]
```

Finally, we can plot every home sold in San Francisco over the two-year period from January 2013 to December 2015 by longitude and latitude, color-coded based on price. Hopefully, we'll be able to see a pattern emerging as to where the pricey houses are located.

```
# use ggplot to render a scatterplot of homes, colored by price bin
ggplot(data=home_prices_bins) + geom_point(aes(x=longitude, y=latitude, color=bin)) +
  xlab("Longitude") + ylab("Latitude")
```



Looking at the plot, we can see that red dots signify less expensive houses whereas purple dots denote the most expensive. There's clearly a relationship, as expected, between where a house is located and where its price falls relative to other San

Francisco homes. But without a ridiculously firm grasp on every single aspect of the city's geography, no one would be able to look at this plot and easily derive any sort of insight. That is where `ggmap` can be used to overlay this otherwise ineffectual, if not aesthetically appealling, scatterplot onto a useful map.

Using `ggmap`

First things first: we'll import the `ggmap` library and make a call to the Google Maps API.

```
library("ggmap")
```

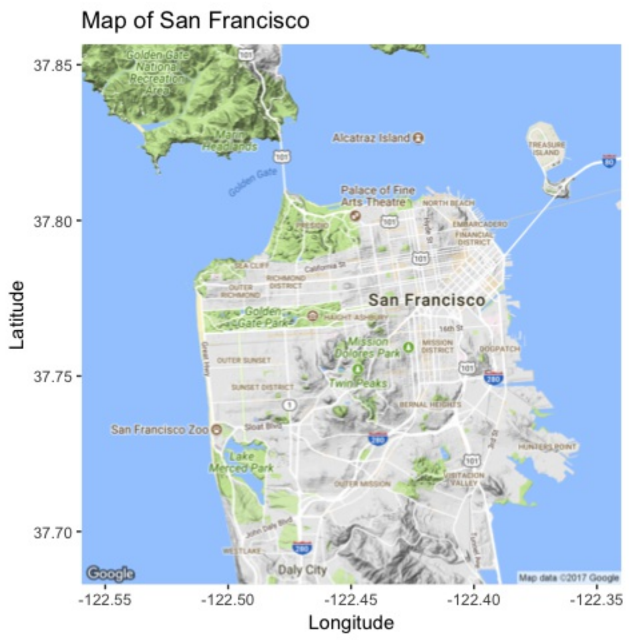
San Francisco sits at a latitude of 37.77 degrees, and a longitude -122.45 degrees. I'll first make a vector called `san_francisco` of these two values, then use `get_map()` to create a `ggmap` object. This object will be a color map of San Francisco, centered at the latitude and longitude which I specify in my vector.

```
san_francisco <- c(lon = -122.45, lat = 37.77)

# the get_map() function is what makes the call to the Google API
sf_map <- get_map(location = san_francisco, zoom = 12, color = "color")
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=37.77,-122.45&zoom=12&size=640x640&s
```

Now that I have my map object stored as `sf_map`, I can easily call `ggmap()` and render a detailed map of the city.

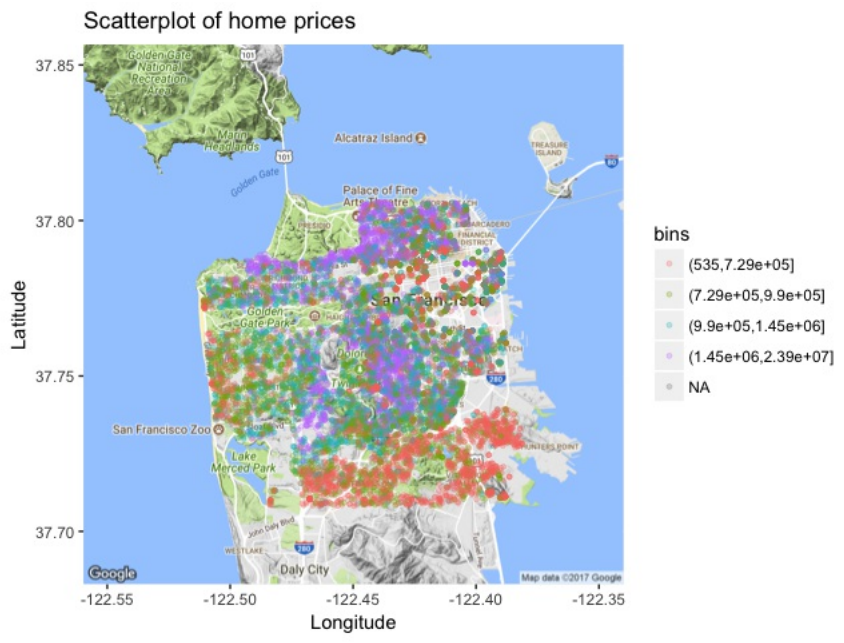
```
# establish a base map of San Francisco
base_map <- ggmap(sf_map) + xlab("Longitude") + ylab("Latitude")

base_map + ggtitle("Map of San Francisco")
```



Overlaying my housing data

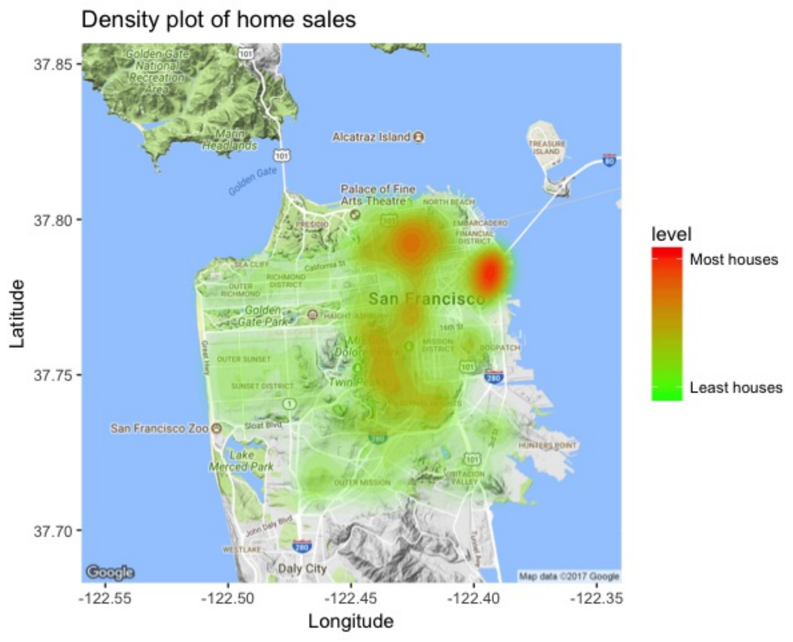
With a map of the city easily rendered, we can now use `geom_point` to overlay the scatterplot from before onto map. With this, we'll be able to get a better picture of the most expensive areas in San Francisco. As with before, those areas that are stacked with purple and green will tend to have the pricier homes, while the red areas will have the lowest home prices.

```
# use ggmap to overlay a scatterplot of homes, colored by price bin, over the base map
base_map + geom_point(data = home_prices_bins, aes(x=longitude, y=latitude, color=bins),
                      alpha=.25, cex=1) +
  ggtitle("Scatterplot of home prices")
```



This is a nice visualization, and is much more useful than the simple scatterplot used before. Now we can actually identify the long strip of empty points in the northwest corner as Golden Gate Park, rather than a confusingly geometric hole in our data. Say we also want to take note of the most densely populated areas of the city — at least in terms of number of properties sold. This can be accomplished with a density plot, which we will set using `stat_density_2d`. We will specify the lowest-density areas to be green, and the highest to be red.

```
# overlay a density plot to show areas of highest and lowest population density
base_map +
  stat_density2d(data = home_prices,
                 aes(x = longitude, y = latitude, fill = ..level.., alpha = ..level..),
                 size = .0001, bins = 16, geom = "polygon") +
  scale_fill_gradient(low = "green", high = "red", labels=c("Least houses", "Most houses"), breaks=c(40, 28
  scale_alpha(range = c(0.05, 0.75), guide = FALSE) +
  ggtitle("Density plot of home sales")
```



As one could expect, the most densely populated area, that at the base of the Bay Bridge, is also one of the areas with the most diverse range of property prices. As this neighborhood thereby caters to people of all socioeconomic classes, it comes as no surprise that it sees the highest density of properties being purchased.

To Conclude

When using spatial data, visualization is among the most significant, if not *the* most significant, aspect of reporting. Without an effective way of conveying to your audience where exactly your data exists, the purpose of geographic data becomes lost. As a simple-to-learn tool that can drastically improve the quality of one's visualizations, `ggmap` stands among the best plotting packages R has to offer. Mastery of its various inputs and control parameters can be achieved in only a few hours with prior knowledge of `ggplot`, and is a vital part of any data analyst's toolkit.

References and Further Reading

https://datascienceplus.com/linear-regression-in-python-predict-the-bay-areas-home-prices/

https://raw.githubusercontent.com/RuiChang123/Regression_for_house_price_estimation/master/final_data.csv

"Introduction to visualising spatial data in R" by Robin Lovelace, James Cheshire, Rachel Oldroyd and others

http://www.exegetic.biz/blog/2013/12/contour-and-density-layers-with-ggmap/

https://www.r-bloggers.com/google-maps-and-ggmap/

https://blog.dominodatalab.com/geographic-visualization-with-rs-ggmaps/

https://cran.r-project.org/web/packages/ggmap/ggmap.pdf