```
library(sp)
library(rgdal)
library(maps)
```

Today's section will quickly debrief Problem Set 2 and introduce spatial data analysis in R.

Problem Set Debrief

- To be added.

Mapping

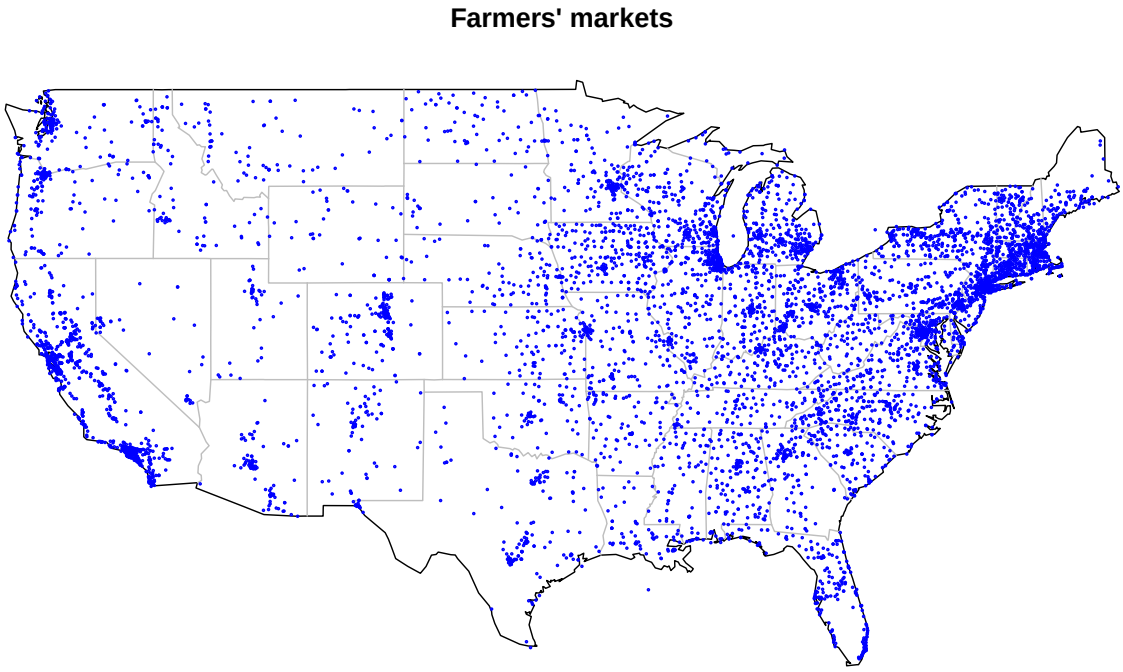
This is a fun section to show how to visualize spatial data in R. Thinking about the spatial dimension of data can be an interesting and useful addition to a standard econometrics analysis. Interesting because, hey, visualizations are cool, and useful because spatial patterns can sometimes provide sources of quasi-random variation. Today, we're just going to stick to visualizations.

Farmers' Markets Data

Data.gov is a data repository administered by the US government with over 445,000 geographic data sets. One data set is the coordinates and characteristics of 7,863 farmers markets in the United States[1]. Today, we're going to be making some maps and plotting these data using both base graphics and `ggplot2`. Patrick was nice enough to clean these data up for us and turn it into a `.csv` (which is now on bCourses). Let's read them in and plot them on a US map.

```
library(maps)
farmersMarkets <- read.csv("farmers-mkts.csv", header = TRUE)
map("state", interior = FALSE)
title("Farmers' markets")
map("state", boundary = FALSE, col = "gray", add = TRUE)
points(farmersMarkets$x, farmersMarkets$y, cex = 0.2, col = "blue")
```

[1] https://explore.data.gov/d/wfna-38ey



This introduces the `maps` package; this is an old one and provides `map` objects which, as far as I can tell, are only useful for visualization. The distribution of farmers markets across the US is neat to see, but there are so many points that it is difficult to visually glean any useful information. Let's cut it down to some of our favorite ZIPs. We'll also download a ZCTA (ZIP code tabulation area) polygon for the backing map.[2]

```
bayZips <- c("94706", "94707", "94708", "94710",
             "94702", "94720", "94703", "94709",
             "94704", "94705", "94618", "94611",
             "94607", "94612", "94610", "94602",
             "94606", "94609", "94608",
             "94102", "94103", "94104", "94105",
             "94107", "94108", "94109", "94110",
             "94111", "94112", "94114", "94115",
             "94116", "94117", "94118", "94121",
             "94122", "94123", "94124", "94127",
             "94129", "94130", "94131", "94132",
             "94133", "94134", "94158")

bayFMs <- farmersMarkets[farmersMarkets$Zip %in% bayZips,]
```

[2]Note that, because a ZIP code is simply a collection of addresses for postal delivery purposes, the concept of a ZIP polygon doesn't quite make sense. There are many blocks that have addresses from more than one ZIP, and the ZCTA allocates each census block to the ZIP that accounts for the most addresses on that block.

```
tmpFile <- tempfile()
download.file("http://www2.census.gov/geo/tiger/GENZ2013/cb_2013_us_zcta510_500k.zip",
              destfile = tmpFile)
unzipped <- unzip(zipfile = tmpFile, exdir = getwd())
zipPoly <- readOGR(dsn = getwd(), layer = "cb_2013_us_zcta510_500k")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:/Users/Kenny/Dropbox/PhD/ARE212GSI/Github/ARE212/section-12", layer: "cb_2013_us_z
## with 33144 features and 5 fields
## Feature type: wkbPolygon with 2 dimensions
```

```
# Remove all the files on disk
file.remove(tmpFile)
```

```
## [1] TRUE
```

```
file.remove(unzipped)
```

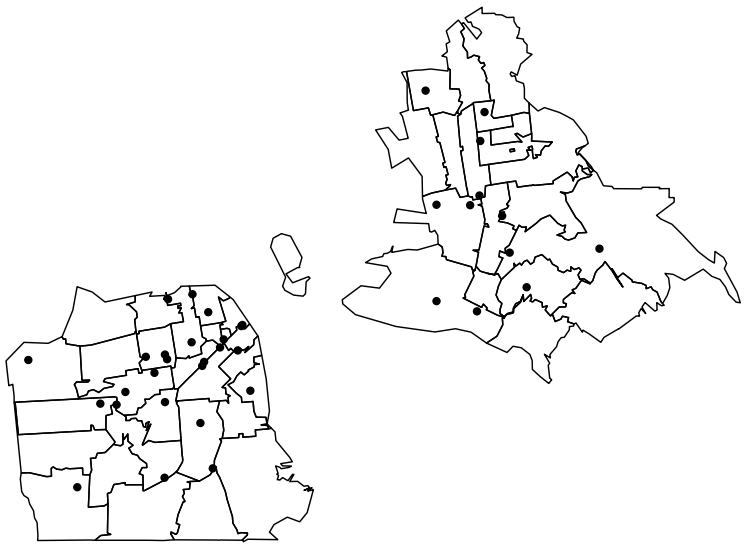
```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
bayPoly <- zipPoly[zipPoly$ZCTA5CE10 %in% bayZips,]
```

Base graphics

Let's take a look at what we've got so far:

```
plot(bayPoly)
# pch = 20 is a filled dot, cex is dot size
points(bayFMs$x, bayFMs$y, cex = 1, pch = 20)
```

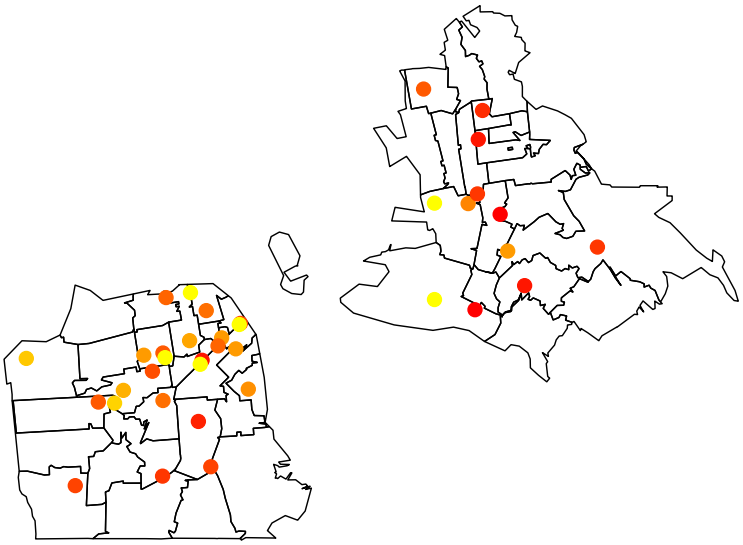


Great! We've generated a simple map without too much fuss. Seems like it could be better. Let's try and add something. In the `bayFMs` object, the last 24 columns are binary variables with entries `"Y"` or `"N"`, indicating whether the market sells cheese, for example, or accepts credit cards. These are the attributes which might indicate the variety or quality of the farmers market. Let's add a column that summarizes these attributes by simply adding up the `"Y"`'s:

```
bayFMs$qualityScore <- apply(bayFMs, 1, function(row) {
  sum(row == "Y")
})
```

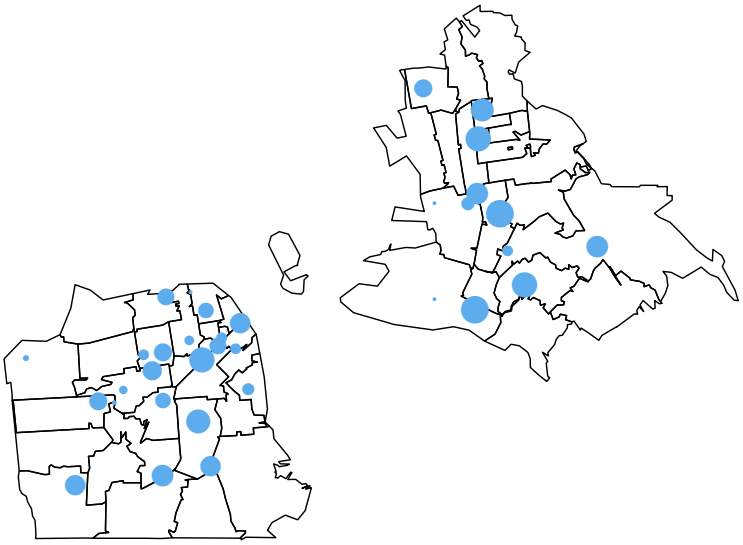
Now, let's visualize this quality score on the map! Remember, any plot can contain a third dimension using color or dot size (or a fourth using both!), not just maps. For the dot colors, we'll need another new package, `plotrix`.

```
plot(bayPoly)
library(plotrix)
points(bayFMs$x, bayFMs$y, cex = 2, pch = 20,
       col = color.scale(bayFMs$qualityScore, extremes=c("yellow","red")))
```



We can also alter the size of the dot to indicate the quality score:

```
plot(bayPoly)
symbols(bayFMs$x, bayFMs$y, cex = 2, pch = 20, circles = bayFMs$qualityScore,
        inches=1/10, ann=F, bg="steelblue2", fg=NULL, add=TRUE)
```



So, base graphics in R can do a lot, but building these things up is messy and you run into walls fast. Let's move to `ggplot2`. I like to use base graphics for exploration as they're a bit more efficient than `ggplot2`. For publication ready plots, `ggplot2` is it.

ggplot2

Before we get into plotting things, lets calculate the population density for each ZCTA that we have. Once we've got the right plot, this will allow us to visually see the correlation between population density and farmer's market location. Splitwise[3], for some reason, provides these.

```
populationByZCTA <- read.csv(paste0("http://s3.amazonaws.com/Splitwi",
                                    "seBlogJB/2010+Census+Population+",
                                    "By+Zipcode+(ZCTA).csv"))
names(populationByZCTA)[1] <- "ZCTA5CE10"
# Creates a new SpatialPolygonsDataFrame
# sort = FALSE as we need the polygons to match with their data
bayPoly@data <- merge(bayPoly@data, populationByZCTA,
                      all.x = TRUE, sort=FALSE)
bayPoly$popDens <- bayPoly$X2010.Census.Population /
  bayPoly@data$ALAND10
```

Whenever we plot a `SpatialPolygonsDataFrame` using `ggplot2`, we have to convert it into a `data.frame` using the `fortify` function. For very large sets of polygons this can take a long time, which is unfortunate and a flaw with this solution. You can simplify polygons using `gSimplify` from `rgeos` but this can sometimes result in ugly plots. I usually test plots with a highly simplified polygon object, and run the plot using the high quality polygon object when I know it's correct.

[3] A bill splitting app.

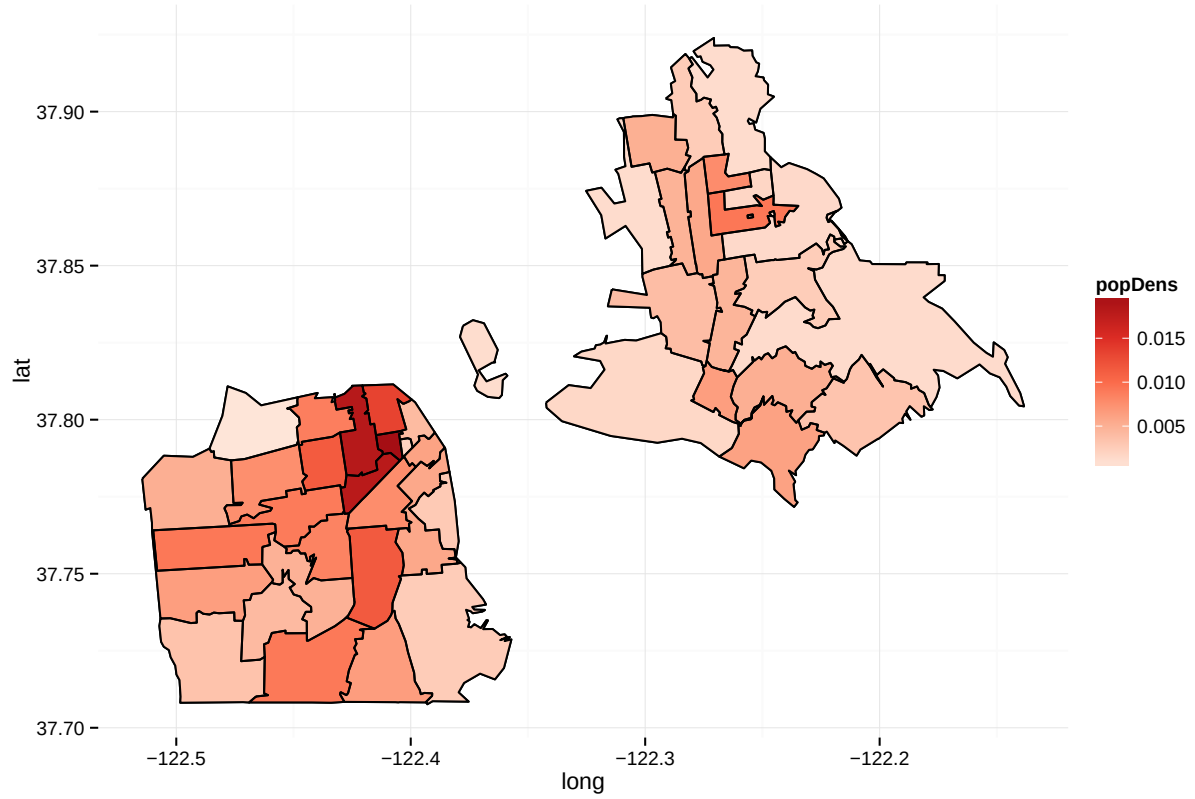
```
library(ggplot2)
# Adds a unique id in case it's not there already
bayPoly@data$id <- rownames(bayPoly@data)

# Turns the SpatialPolygons object into a data.frame - slow
# simplify with gSimplify if this takes forever. Or consider
# subsetting the polygon to what you actually want to plot.
bayPolypoints <- fortify(bayPoly, region = "id")
library(plyr) # plyr::join is a bit faster than merge
bayPolyDf <- join(bayPolypoints, bayPoly@data, by = "id")
```

Now, we have a large `data.frame` where each row represents a point on one of the polygons. Because we've just got a bunch of grouped points, `ggplot2` can work for us now, as long as we correctly choose our aesthetics and geom's.

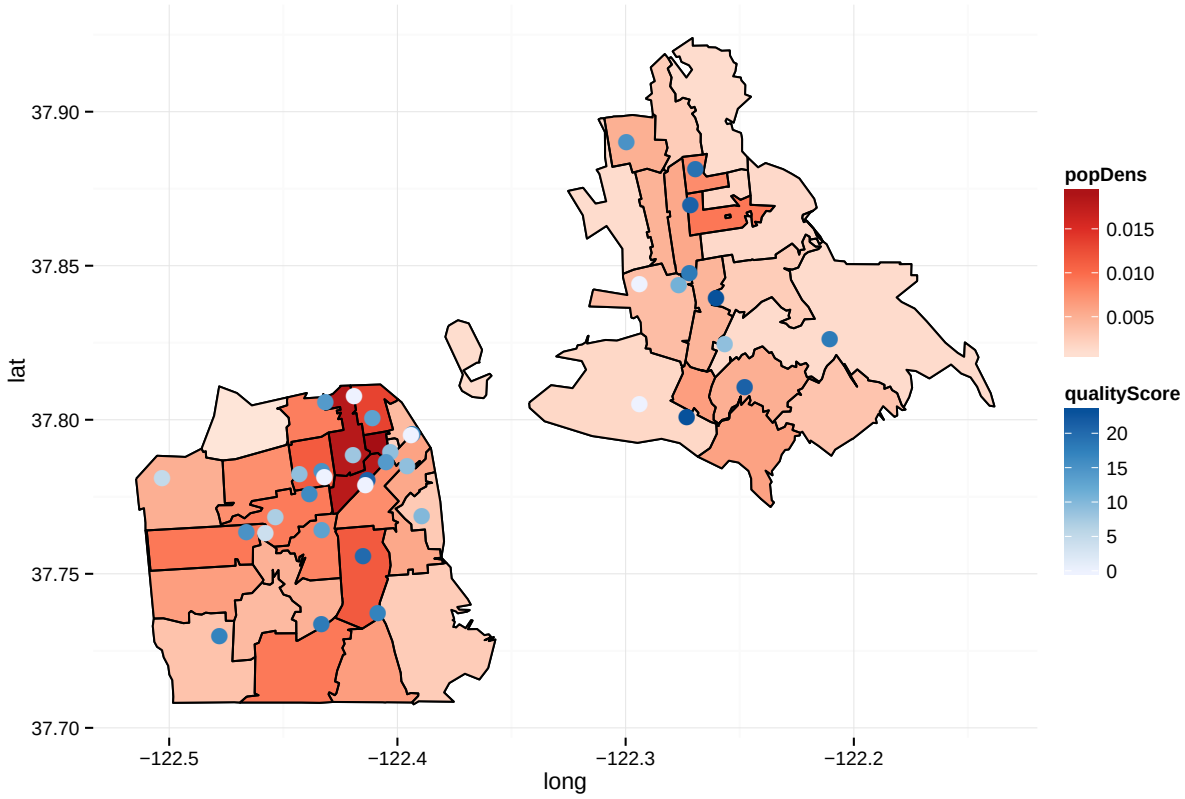
```
mapPlot <- ggplot(bayPolyDf, aes(long, lat)) +
  geom_path(aes(group=group)) + theme_minimal()

library(RColorBrewer)
mapPlot <- mapPlot + geom_polygon(aes(fill = popDens, group=group)) +
  scale_fill_gradientn(colours = brewer.pal(5, "Reds")) + geom_path(aes(group=group))
mapPlot
```



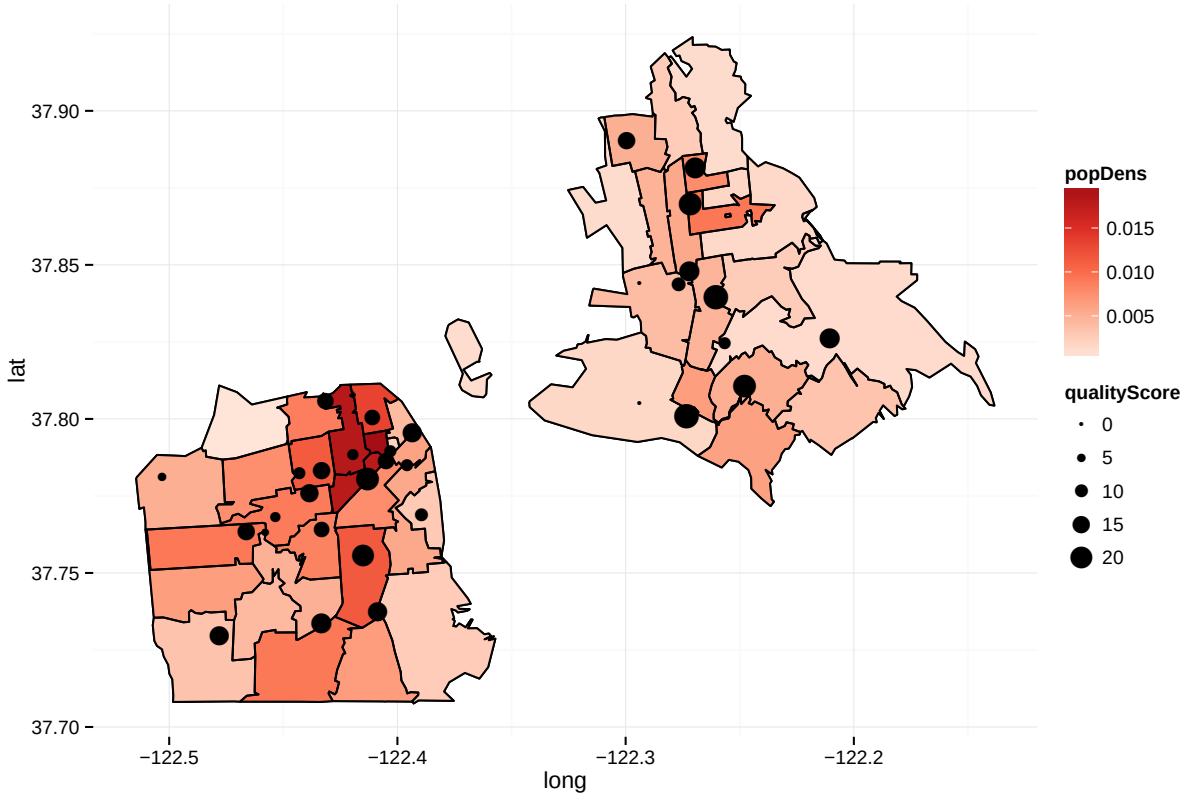
Now, let's add the farmers markets with the quality score as the color:

```
mapPlot + geom_point(data = bayFMs, aes(x = x, y = y, group = NULL,
                                        color = qualityScore), size = 4) +
  scale_colour_gradientn(colours = brewer.pal(5, "Blues"))
```



And the quality score as the size of the dot:

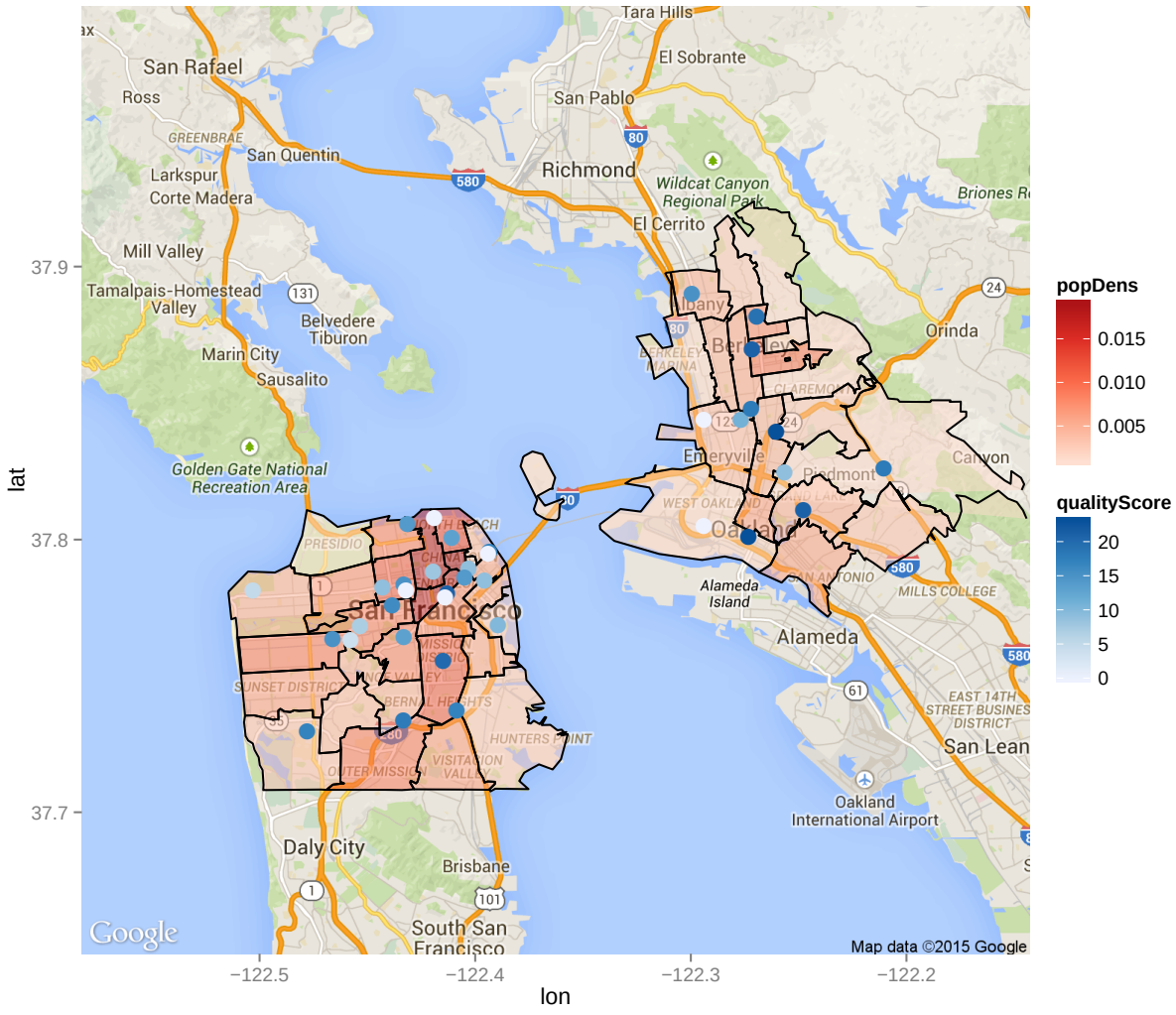
```
mapPlot + geom_point(data = bayFMs, aes(x = x, y = y, group = NULL,
                                        size = qualityScore))
```



We can even plot on top of static google maps!

```
library(ggmap)
ggMap <- ggmap(get_map(location = c(-122.363028, 37.822109),
                       zoom = 11, scale = 4, maptype = "roadmap"))
ggMap <- ggMap +
  geom_polygon(data = bayPolyDf, aes(long, lat, fill = popDens, group=group), alpha = 0.5) +
  scale_fill_gradientn(colours = brewer.pal(5, "Reds")) +
  geom_path(data = bayPolyDf, aes(long, lat, group=group))

ggMap + geom_point(data = bayFMs, aes(x = x, y = y, group = NULL,
                                      color = qualityScore), size = 4) +
  scale_colour_gradientn(colours = brewer.pal(5, "Blues"))
```


San Rafael
Ross
Larkspur
Corte Madera
Mill Valley
Tamalpais-Homestead Valley
Marin City
Sausalito
Belvedere
Tiburon
San Quentin
Golden Gate National Recreation Area
Tara Hills
El Sobrante
San Pablo
Richmond
Wildcat Canyon Regional Park
El Cerrito
Albany
Berkeley
Emeryville
Piedmont
Oakland
Alameda
Alameda Island
Orinda
Canyon
San Francisco
Daly City
Brisbane
South San Francisco
Oakland International Airport
San Lean
Google
Map data ©2015 Google
lat
lon
37.9
37.8
37.7
−122.5
−122.4
−122.3
−122.2
popDens
0.015
0.010
0.005
qualityScore
20
15
10
5
0