Sales location analysis

Ian Thulin

2023-05-11

```
if(!file.exists("Data")){dir.create("Data")}
```

We need to clean the address data in order to improve the geocoding function we will use later. To start I will parse the street number, name, type, and direction from the given data. Once the fields have been separated we can adjust any capitalization errors and combine the fields into a form usable by tidygeocoder.

```
if(!file.exists("Data/Raw data (pre-geocoding).csv")) {

# Load in the original data files
Raw_sales_data <- read_excel("Data/Sandia Sunrooms - Sales Data - Lead Perfection - 5.11.2023.xlsx")
Raw_prospect_data <- read_excel("Data/Sandia Sunrooms - Prospect Data - Lead Perfection - 5.11.2023.xlsx

# We need to remove several fields from the given data set to protect the anonymity of our clients
Raw_sales_data <- Raw_sales_data %>% select(-FirstName, -LastName, -Phone, -Phone2)
Raw_prospect_data <- Raw_prospect_data %>% rename("id" = "CustNumber",
                                                  "productid" = "Productid",
                                                  "SubSource" = "SourceSubDescr")
Raw_prospect_data$id <- as.numeric(as.character(Raw_prospect_data$id))

# Combine the two data sets into a single data frame consisting of all sales and prospect information
Raw_data <- dplyr::bind_rows(Raw_sales_data, Raw_prospect_data)
  # Convert the address column to upper case to help type and direction recognition
Raw_data$Address1 <- toupper(Raw_data$Address1)

# Regular expression for each component we need to strip
number_regex <- "^\\d+"
street_regex <- "\\b[A-Za-z]+(\\s+(?!ST|AVE|BLVD|DR|CT|LN|RD|WAY|CIR|TER|HWY|NE|SE|NW|SW)[A-Za-z]+)*\\b"
type_regex <- "\\b(?:ST|AVE|BLVD|DR|CT|LN|RD|WAY|CIR|PL|TER|HWY)\\b"
direction_regex <- "\\b(?:N|S|E|W|NE|SE|NW|SW)\\b"

# Extract each component and add the value as a new column
Raw_data['StreetNumber'] <- str_extract(Raw_data$Address1, number_regex)
Raw_data <- Raw_data %>% relocate(StreetNumber, .before = City)

Raw_data['StreetName'] <- str_extract(Raw_data$Address1, street_regex)
Raw_data <- Raw_data %>% relocate(StreetName, .before = City)

Raw_data['StreetType'] <- str_extract(Raw_data$Address1, type_regex)
Raw_data <- Raw_data %>% relocate(StreetType, .before = City)

Raw_data['StreetDirection'] <- str_extract(Raw_data$Address1, direction_regex)
```

```
Raw_data <- Raw_data %>% relocate(StreetDirection, .before = City)

# Convert StreetName and StreetType back to title case to correct grammar
Raw_data$StreetName <- str_to_title(Raw_data$StreetName)
Raw_data$StreetType <- str_to_title(Raw_data$StreetType)

# Set the value of Address2 equal to the form
# "StreetNumber" "StreetName" "StreetType" "Direction" "City", "State" "Zipcode"

# Create a new data frame to combine all columns while omitting NA values.
AddressPart1 <- data.frame(Raw_data$StreetNumber, Raw_data$StreetName, Raw_data$StreetType, Raw_data$St
AddressPart1$FullAddress <- apply(AddressPart1, 1, function(x) paste(x[!is.na(x)], collapse = " "))

# Reassign the combined address to Address2 in Raw_data and then past in the city, state and zip with c
Raw_data$Address2 <- AddressPart1$FullAddress
Raw_data$Address2 <- paste(Raw_data$Address2, ", ", Raw_data$City, ", ", Raw_data$State, Raw_data$Zip)
Raw_data <- Raw_data %>% rename('FullAddress' = 'Address2')
Raw_data$StreetName <- AddressPart1$FullAddress

# Create a new data frame to geocode the address information
Raw_Geo_data <- data.frame(AddressPart1$FullAddress, Raw_data$City, Raw_data$State, Raw_data$Zip)

# Remove all the redundant data fields created in cleaning this portion of the data
Raw_data <- Raw_data %>% select(-StreetNumber, -StreetType, -StreetDirection, -Address1)
rm(AddressPart1)

# Export the cleaned data to CSV files in the data folder
write.csv(Raw_Geo_data, "Data/Data for export to Geocodio.csv", row.names=FALSE)
write.csv(Raw_data, "Data/Raw data (pre-geocoding).csv", row.names=FALSE)
}
```

The next step in the data cleaning process is to create a new field for the latitude and longitude of each address in our data set. Using the tidygeocoder package in R we can input an address and receive the associated latitude and longitude for each data point. This information can then be passed into the Google maps API for further analysis.

```
if(!file.exists("Data/Geolocated Data - Sales & Prospect - 5.11.2023.csv")) {

# Load the previously created csv files in preparation for merging
Temp_data <- read.csv("Data/Geocoded Addresses.csv")
Pre_Geo <- read.csv("Data/Raw data (pre-geocoding).csv")

# Remove unneeded columns from geo-data
Temp_data <- Temp_data[-c(2:5,10:19)]

# Combine the two data sets and save to file
Comb_data <- cbind(Pre_Geo, Temp_data[c("Latitude", "Longitude", "Accuracy.Score", "Accuracy.Type")])
write.csv(Comb_data, "Data/Geolocated Data - Sales & Prospect - 5.11.2023.csv", row.names=FALSE)
}
```

```
Data <- read.csv("Data/Geolocated Data - Sales & Prospect - 5.11.2023.csv")

Sales <- subset(Data, (!is.na(Data$ContractDate)))
```

```
Prospect <- subset(Data, (is.na(Data$ContractDate)))

ABQ_sales <- subset(Sales, Sales$City == "Albuquerque")
RR_sales <- subset(Sales, Sales$City == "Rio Rancho")
SF_sales <- subset(Sales, Sales$City == "Santa Fe")

Metro_sales <- bind_rows(ABQ_sales, RR_sales)
```

```
earth.dist <- function (long1, lat1, long2, lat2) {
  rad <- pi/180
  a1 <- lat1 * rad
  a2 <- long2 * rad
  b1 <- lat2 * rad
  b2 <- long2 * rad
  dlon <- b2 - a2
  dlat <- b1 - a1

  a <- (sin(dlat/2))^2 + cos(a1) * cos(b1) * (sin(dlon/2))^2
  c <- 2 * atan2(sqrt(a), sqrt(1 - a))
  R <-6378.145
  d <- R * c
  return(d)
}
```

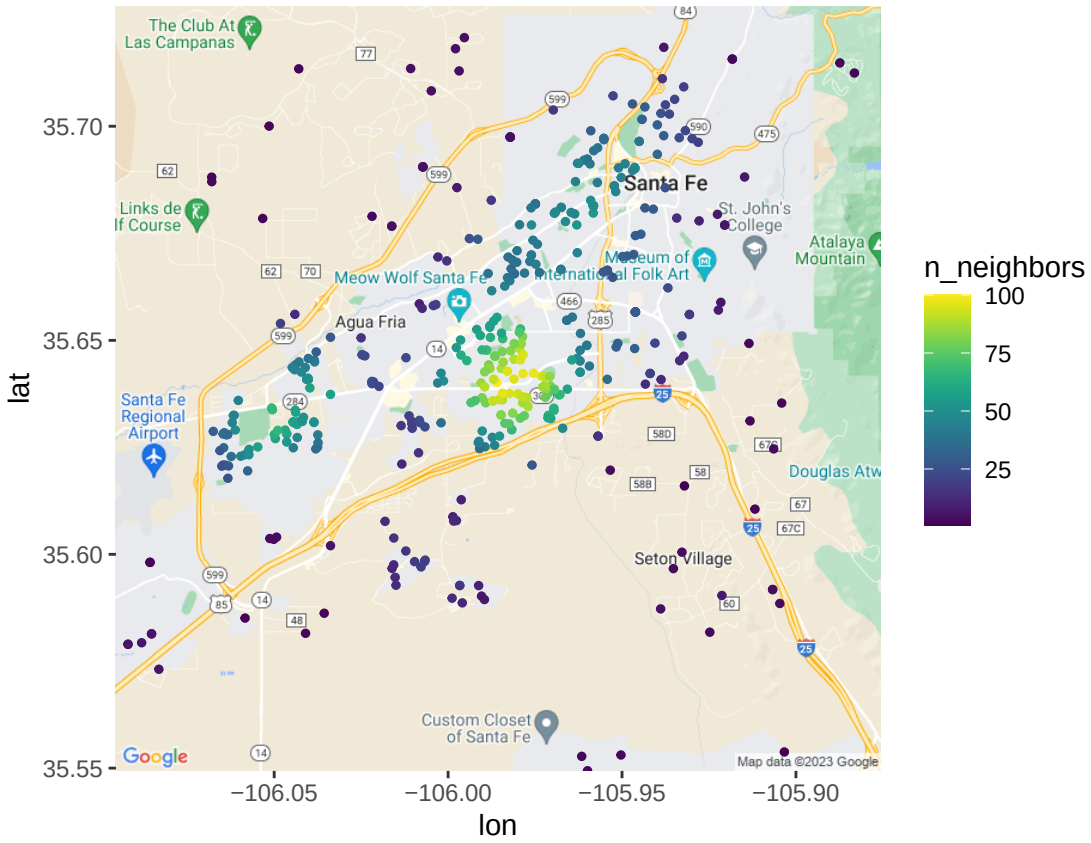
```
# Add empty column to each set of sales/prospect data
RR_sales['Distance'] <- NA
ABQ_sales['Distance'] <- NA
SF_sales['Distance'] <- NA

# Calculate distance from the calculated center of each area
RR_sales$Distance <- earth.dist(RR_sales$Longitude,
                                RR_sales$Latitude,
                                mean(RR_sales$Longitude),
                                mean(RR_sales$Latitude))
ABQ_sales$Distance <- earth.dist(ABQ_sales$Longitude,
                                ABQ_sales$Latitude,
                                mean(ABQ_sales$Longitude),
                                mean(ABQ_sales$Latitude))
SF_sales$Distance <- earth.dist(SF_sales$Longitude,
                                SF_sales$Latitude,
                                mean(SF_sales$Longitude),
                                mean(SF_sales$Latitude))

# Filter results farther than X miles from the center of the area
RR_sales <- RR_sales[RR_sales$Distance <= 10,]
ABQ_sales <- ABQ_sales[ABQ_sales$Distance <= 15,]
SF_sales <- SF_sales[SF_sales$Distance <= 20,]
```

```
if (!file.exists("Maps/NM_map.RData")){
NM <- ggmap(get_googlemap(center = c(lon = -106.018066, lat =34.307144),
                          zoom = 7, scale = 2,
                          maptype = 'roadmap',
                          color = 'color'))
```

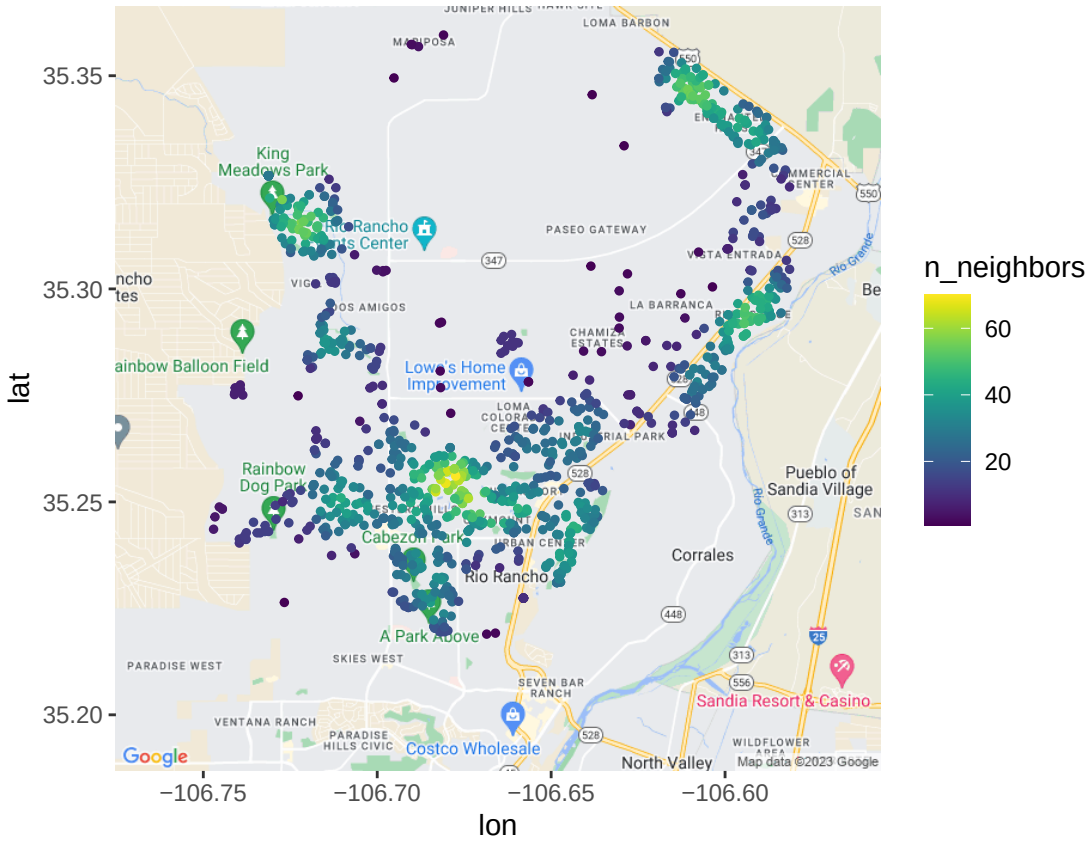
```
save(NM, file = "Maps/NM_map.RData")
}

if (!file.exists("Maps/Metro_map.RData")){
Metro <- ggmap(get_googlemap(center = c(lon = mean(Metro_sales$Longitude), mean(Metro_sales$Latitude)),
                          zoom = 11, scale = 1,
                          maptype = 'roadmap',
                          color = 'color'))
save(Metro, file = "Maps/Metro_map.RData")
}

if (!file.exists("Maps/RR_map.RData")){
RR <- ggmap(get_googlemap(center = c(lon = mean(RR_sales$Longitude), mean(RR_sales$Latitude)),
                          zoom = 12, scale = 1,
                          maptype = 'roadmap',
                          color = 'color'))
save(RR, file = "Maps/RR_map.RData")
}

if (!file.exists("Maps/SFcounty_map.RData")){
SFcounty <- ggmap(get_googlemap(center = c(lon = mean(SF_sales$Longitude), mean(SF_sales$Latitude)),
                          zoom = 11, scale = 1,
                          maptype = 'roadmap',
                          color = 'color'))
save(SFcounty, file = "Maps/SFcounty_map.RData")
}

if (!file.exists("Maps/SF_map.RData")){
SF <- ggmap(get_googlemap(center = c(lon = mean(SF_sales$Longitude), mean(SF_sales$Latitude)),
                          zoom = 12, scale = 1,
                          maptype = 'roadmap',
                          color = 'color'))
save(SF, file = "Maps/SF_map.RData")
}
```

```
load(file = "Maps/Metro_map.RData")
load(file = "Maps/NM_map.RData")
load(file = "Maps/RR_map.RData")
load(file = "Maps/SF_map.RData")
load(file = "Maps/SFcounty_map.RData")
```

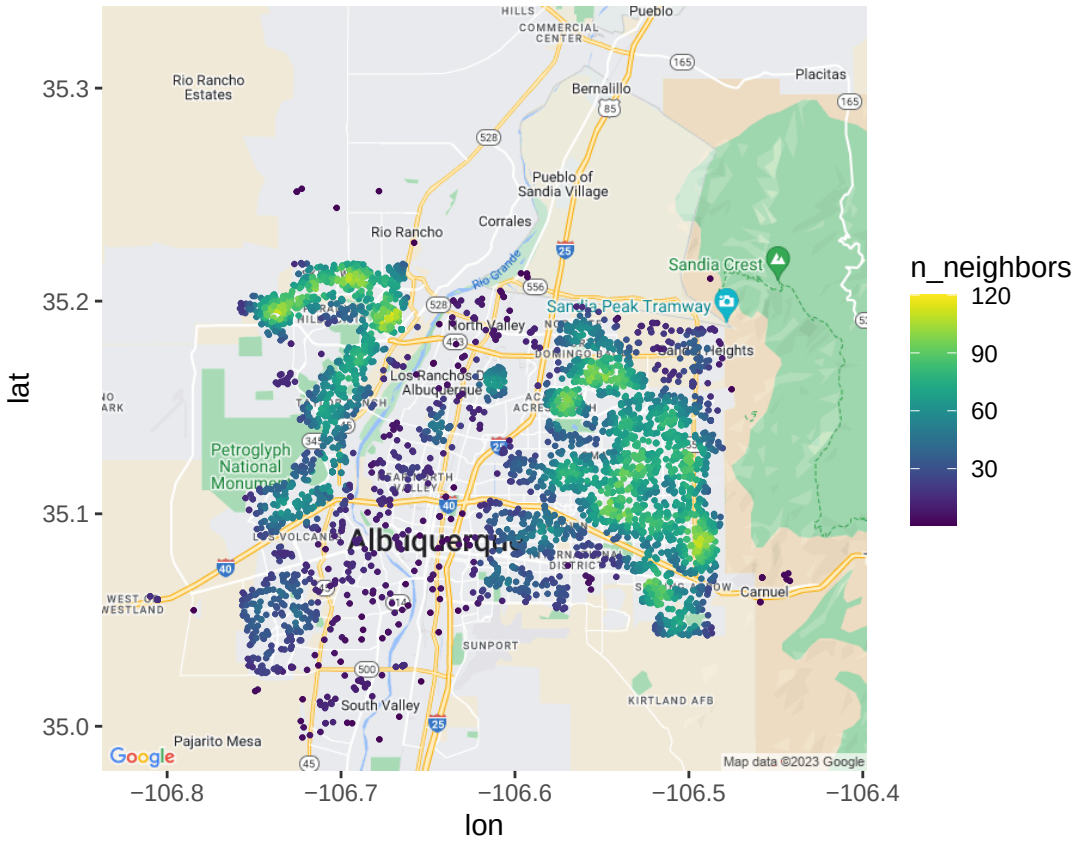
```
SF +
  geom_pointdensity(aes(x = Longitude,
                        y = Latitude),
                    data = SF_sales,
                    size = 1,
                    adjust = .05) +
  scale_color_viridis()
```



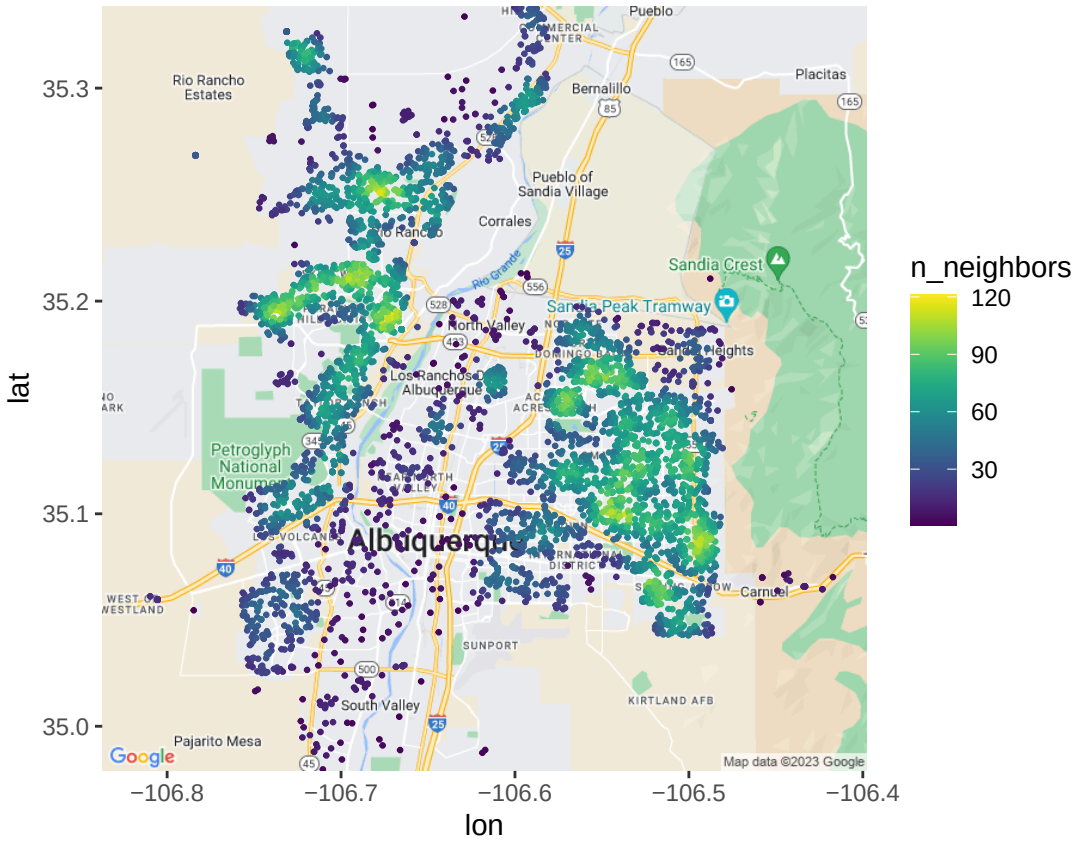

```
RR +
  geom_pointdensity(aes(x = Longitude, y = Latitude),
                    data = RR_sales,
                    size = 1,
                    adjust = .01) +
  scale_color_viridis()
```



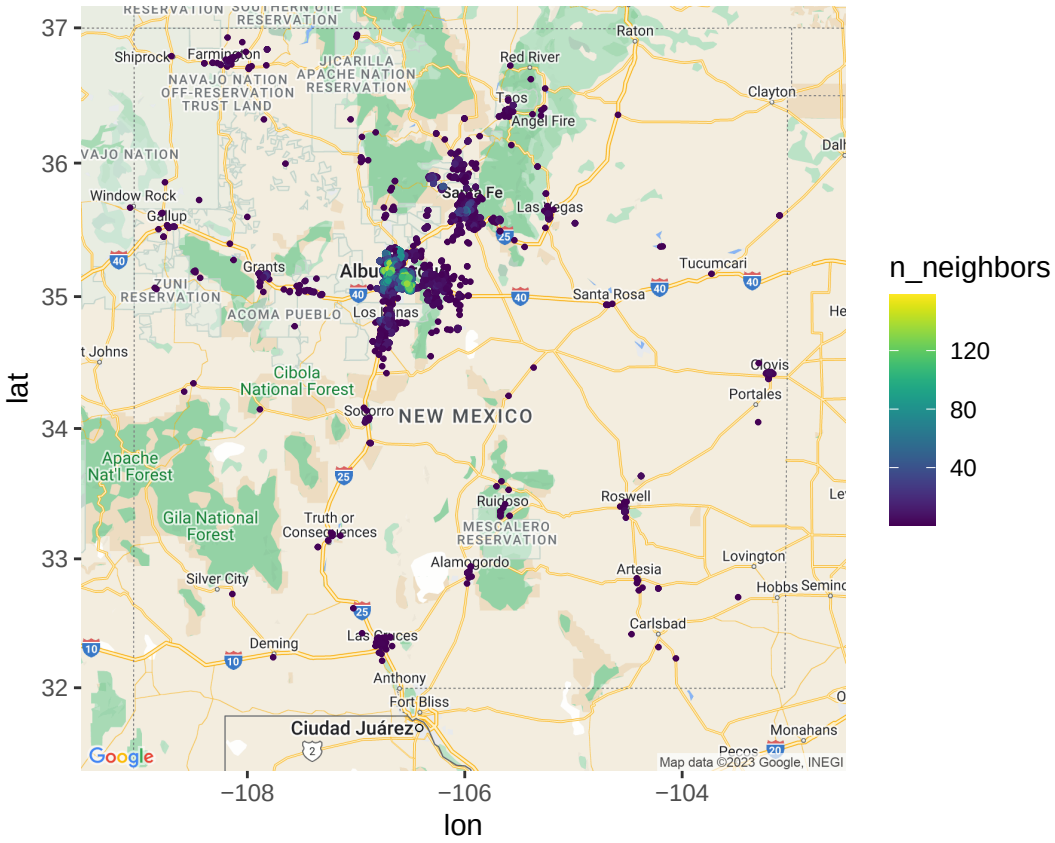

```
Metro +
  geom_pointdensity(aes(x = Longitude, y = Latitude),
                    data = ABQ_sales,
                    size = .5,
                    adjust = .01) +
  scale_color_viridis()
```




```
Metro +
  geom_pointdensity(aes(x = Longitude, y = Latitude),
                    data = Metro_sales,
                    size = .5,
                    adjust = .01)+
  scale_color_viridis()
```




```
NM +
  geom_pointdensity(aes(x = Longitude, y = Latitude),
                    data = Sales,
                    size = .5,
                    adjust = .001) +
  scale_color_viridis()
```




```
citation("ggmap")
sessionInfo()
```