CSSS508, Lecture 9

Mapping

Michael Pearce
(based on slides from Chuck Lanfear)

December 1, 2022

Topics

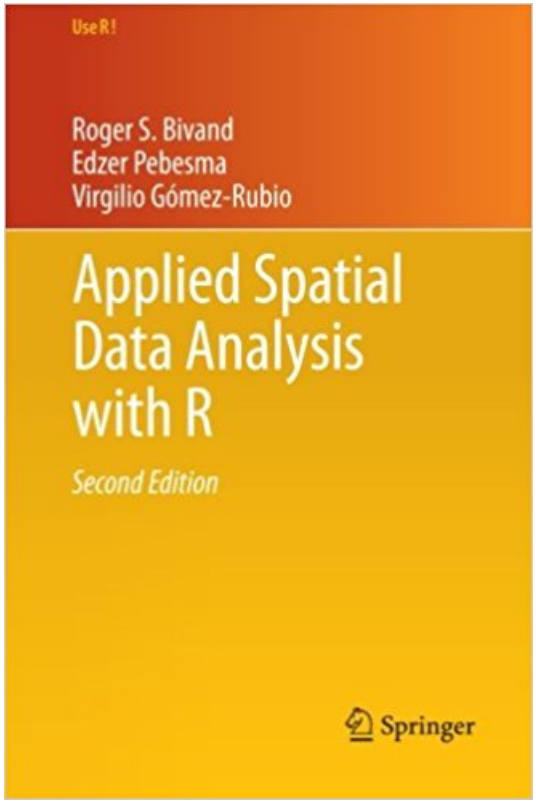
Last time, we learned about,

1. Basics of Strings
2. Strings in Base R
3. Strings in `stringr` (Tidyverse)

Today, we will cover,

1. `ggmap` for mapping in `ggplot2`
2. Density plots
3. Labeling points

Mapping in R: A quick plug



Mapping can be **complex!** If you are interested in digging deeper, here are some resources:

- If you are interested in mapping, GIS, and geospatial analysis in R, *acquire this book*.
- RSpatial.org is a great resource for working with spatial data.
- For more information on *spatial statistics*, consider taking Jon Wakefield's **CSSS 554: Statistical Methods for Spatial Data** this winter.

1. `ggmap`

One Day of SPD Incidents

Today, we'll study data from the Seattle Police Department regarding incidents on just one day: March 25, 2016.

The data can be downloaded from my predecessor's Github using the code below (code in the R companion file).

```
library(ggplot2)
library(readr)
library(dplyr)
```

```
spd_raw <- read_csv("https://clanfear.github.io/CSSS508/Seattle_Polic
```

Taking a `glimpse()`

```
glimpse(spd_raw)
```

```
## Rows: 706
## Columns: 19
## $ `CAD CDW ID`                  <dbl> 1701856, 1701857, 1701853, 170…
## $ `CAD Event Number`            <dbl> 16000104006, 16000103970, 1600…
## $ `General Offense Number`      <dbl> 2016104006, 2016103970, 201610…
## $ `Event Clearance Code`        <chr> "063", "064", "161", "245", "2…
## $ `Event Clearance Description` <chr> "THEFT - CAR PROWL", "SHOPLIFT…
## $ `Event Clearance SubGroup`    <chr> "CAR PROWL", "THEFT", "TRESPAS…
## $ `Event Clearance Group`       <chr> "CAR PROWL", "SHOPLIFTING", "T…
## $ `Event Clearance Date`        <chr> "03/25/2016 11:58:30 PM", "03/…
## $ `Hundred Block Location`      <chr> "S KING ST / 8 AV S", "92XX BL…
## $ `District/Sector`             <chr> "K", "S", "D", "M", "M", "B", …
## $ `Zone/Beat`                   <chr> "K3", "S3", "D2", "M1", "M3", …
## $ `Census Tract`                <dbl> 9100.102, 11800.602, 7200.106,…
## $ Longitude                     <dbl> -122.3225, -122.2680, -122.342…
## $ Latitude                      <dbl> 47.59835, 47.51985, 47.61422, …
## $ `Incident Location`           <chr> "(47.598347, -122.32245)", "(4…
## $ `Initial Type Description`    <chr> "THEFT (DOES NOT INCLUDE SHOPL…
## $ `Initial Type Subgroup`       <chr> "OTHER PROPERTY", "SHOPLIFTING…
## $ `Initial Type Group`          <chr> "THEFT", "THEFT", "TRESPASS", …
## $ `At Scene Time`               <chr> "03/25/2016 10:25:51 PM", "03/…
```

What does each **row represent?** What are the **variables and values?**

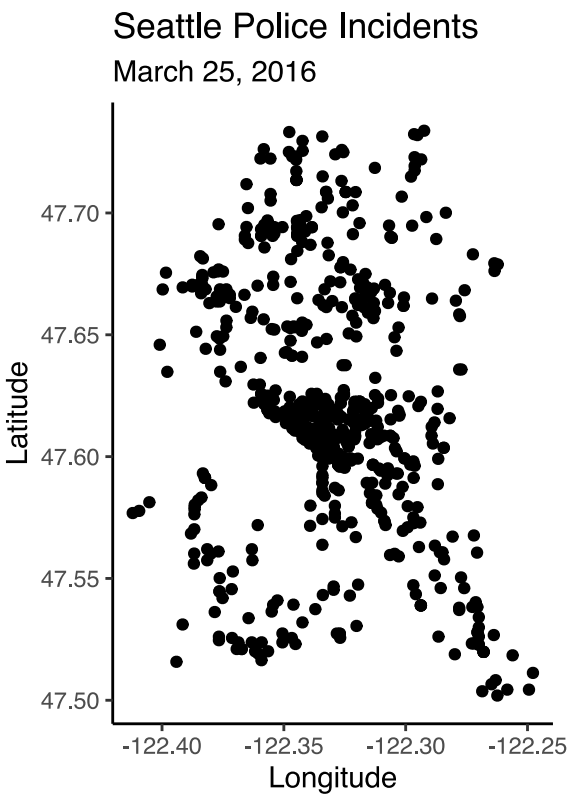
Simple Plotting: Regular `ggplots`

Coordinates, such as longitude and latitude, can be provided in `aes()` as `x` and `y` values.

This is ideal when you don't need to place points over some map for reference.

```
ggplot(spd_raw,
       aes(Longitude, Latitude)) +
  geom_point() +
  coord_fixed() + # evenly spaces x and y
  ggtitle("Seattle Police Incidents",
          subtitle="March 25, 2016") +
  theme_classic()
```

Sometimes, however, we want to plot these points over existing maps.



Better plots with `ggmap`

`ggmap` is a package that works with `ggplot2` to plot spatial data directly on map images.

What this package does for you:

1. Queries servers for a map at the location and scale you want
2. Plots the image as a `ggplot` object
3. Lets you add more `ggplot` layers like points, 2D density plots, text annotations
4. Additional functions for interacting with Google Maps (beyond this course)

Installation

We can install `ggmap` like other packages:

```
install.packages("ggmap")
```

Note: If prompted to "install from sources the package which needs compilation", I find that typing "no" works best!

Because the map APIs it uses change frequently, sometimes you may need to get a newer development version of `ggmap` from the author's GitHub. This can be done using the `remotes` package.

```
if(!requireNamespace("remotes")){install.packages("remotes")}
remotes::install_github("dkahle/ggmap", ref = "tidyup")
```

Note, this may require compilation on your computer.

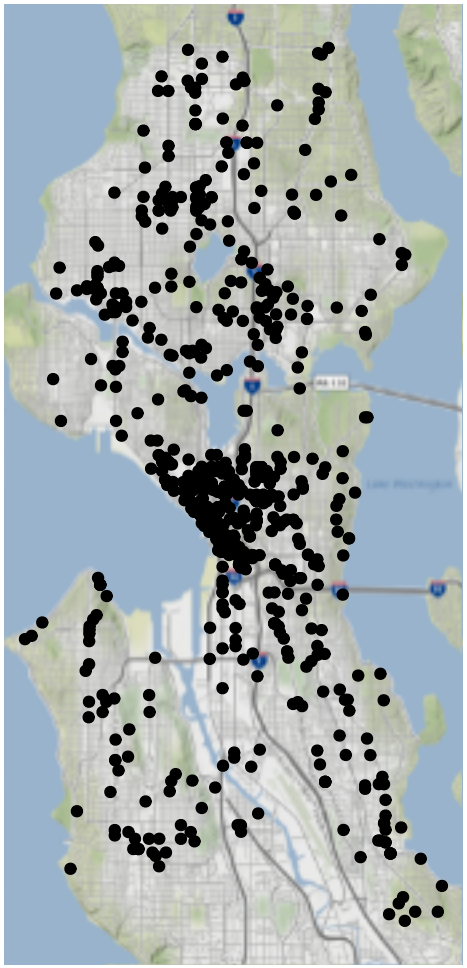
```
library(ggmap)
```

Quick Maps with `qmplot()`

`qmplot` will automatically set the map region based on your data:

```
qmplot(data = spd_raw,
       x = Longitude,
       y = Latitude)
```

All I provided was numeric latitude and longitude, and it placed the data points correctly on a raster map of Seattle.



`get_map()`

`qmplot()` internally uses the function `get_map()`, which retrieves a base map layer. Some options:

- `location=` search query or numeric vector of longitude and latitude
- `zoom=` a zoom level (3 = continent, 10 = city, 21 = building)
- `maptype=`: `"watercolor"`, `"toner"`, `"toner-background"`, `"toner-lite"`
- `color=`: `"color"` or `"bw"`

There are fancier options available, but many require a Google Maps API.

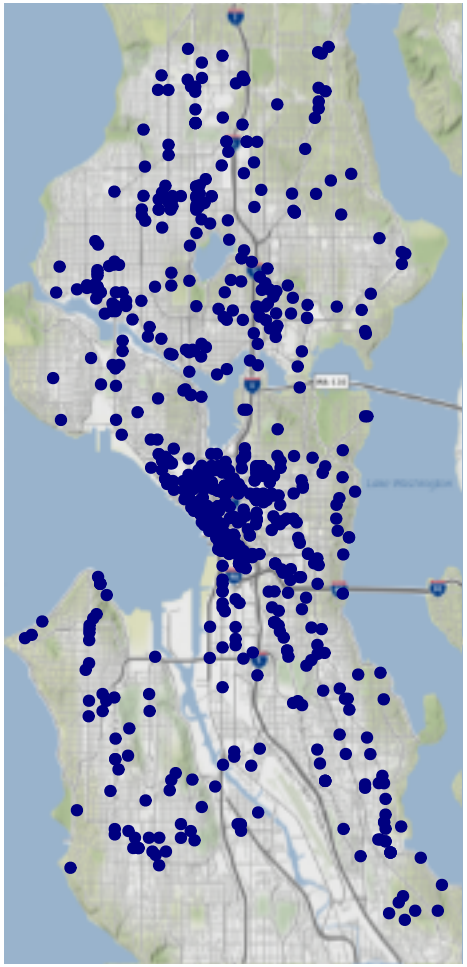
See the **help page** for `gmplot()` or `get_map` for more information!

Nicer Example

Let's add *color* to our plot from before

```
qmplot(data = spd_raw,
       x = Longitude,
       y = Latitude,
       color=I("navy")
)
```

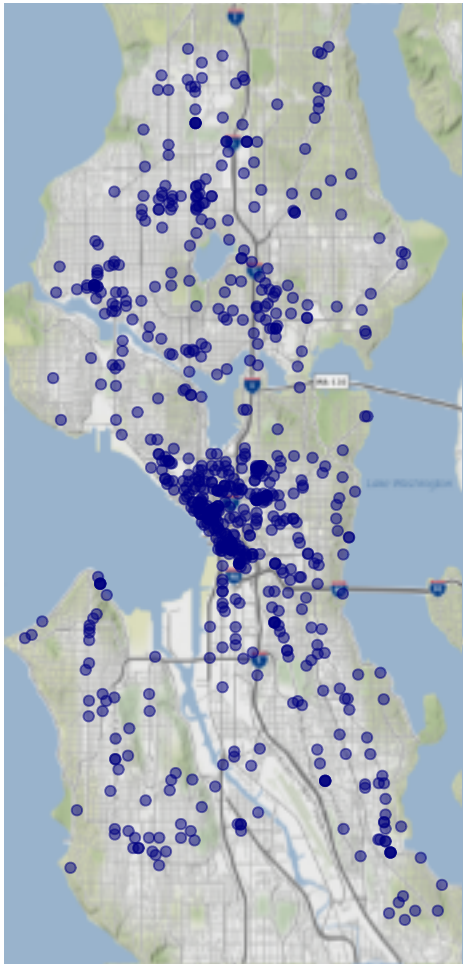
`I()` is used here to specify *set* (constant) rather than *mapped* values.



Nicer Example

Add *transparency*

```
qmplot(data = spd_raw,
       x = Longitude,
       y = Latitude,
       color=I("navy"),
       alpha=I(0.5)
)
```

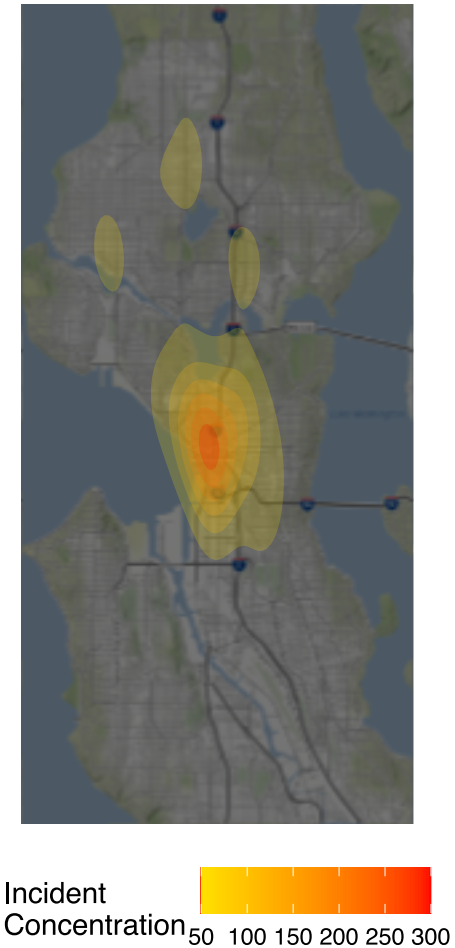


2. Density plots

Density Layers

We can create a **spatial density plot** using the layer function `stat_density_2d()`:

- What to use when creating a "density" plot (where the observations occur)
- Aesthetics of the density plot
- Additional layer functions for customization



Basic Map

Start with basic plot, remove points:

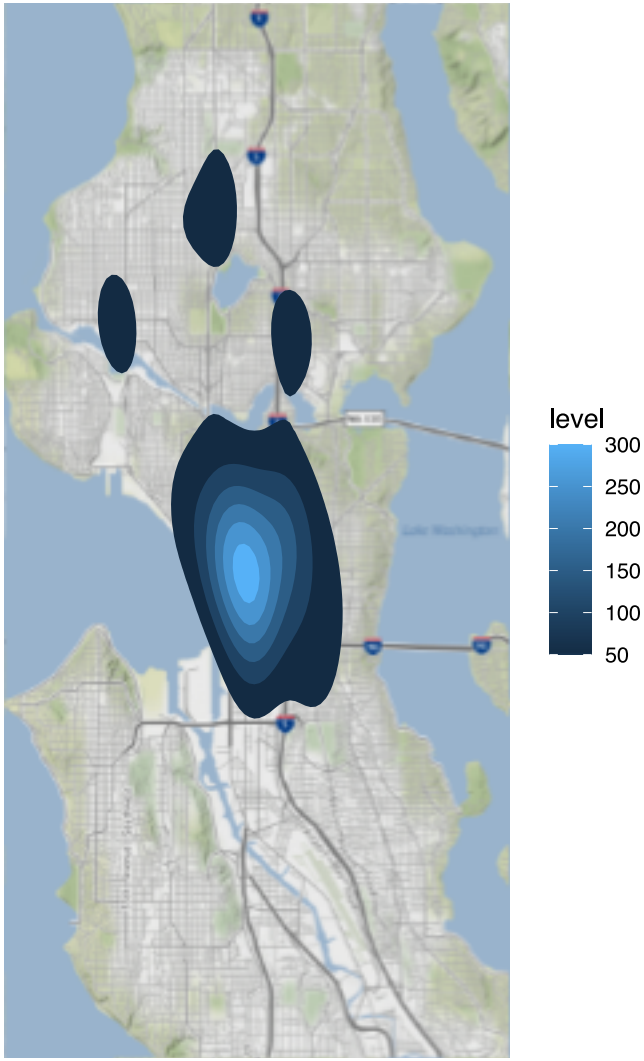
```
qmplot(data = spd_raw,
       geom = "blank",
       x = Longitude,
       y = Latitude)
```

Add Density Layer

Add `stat_density_2d` layer:

```
qmplot(data = spd_raw,
       geom = "blank",
       x = Longitude,
       y = Latitude)+
  stat_density_2d(
    aes(fill = stat(level)),
    geom = "polygon")
```

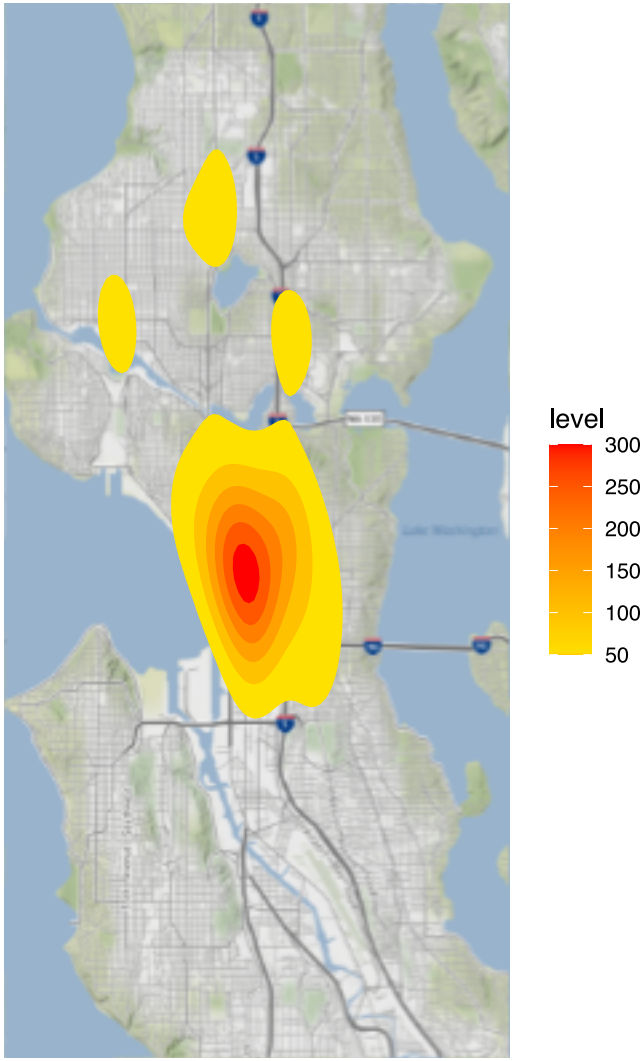
`stat(level)` indicates we want `fill=` to be based on `level` values calculated by the layer (i.e., number of observations).



Color Scale

Specify color gradient

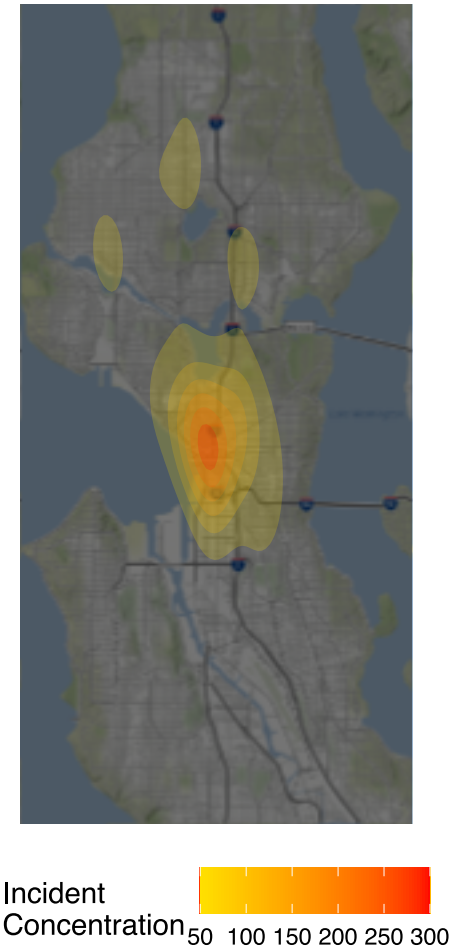
```
qmplot(data = spd_raw,
       geom = "blank",
       x = Longitude,
       y = Latitude)+
  stat_density_2d(
    aes(fill = stat(level)),
    geom = "polygon",
  )+
  scale_fill_gradient2(
    low = "white",
    mid = "yellow",
    high = "red")
```



Customize transparency and legend

```
qmplot(data = spd_raw,
       geom = "blank",
       x = Longitude,
       y = Latitude,
       darken = 0.5)+
  stat_density_2d(
    aes(fill = stat(level)),
    geom = "polygon",
    alpha = .2,
  )+
  scale_fill_gradient2(
    "Incident\nConcentration",
    low = "white",
    mid = "yellow",
    high = "red")+
  theme(legend.position = "bottom")
```

Done!



3. Labeling points

Focus in on Downtown Seattle

First, let's filter our data to downtown based on values "eyeballed" from our earlier map:

```
downtown <- spd_raw %>%
  filter(Latitude > 47.58, Latitude < 47.64,
         Longitude > -122.36, Longitude < -122.31)
```

We'll plot just these values from now on!

Let's also make a dataframe that includes just assaults and robberies downtown:

```
assaults <- downtown %>%
  filter(`Event Clearance Group` %in%
                  c("ASSAULTS", "ROBBERY")) %>%
  mutate(assault_label = `Event Clearance Description`)
```

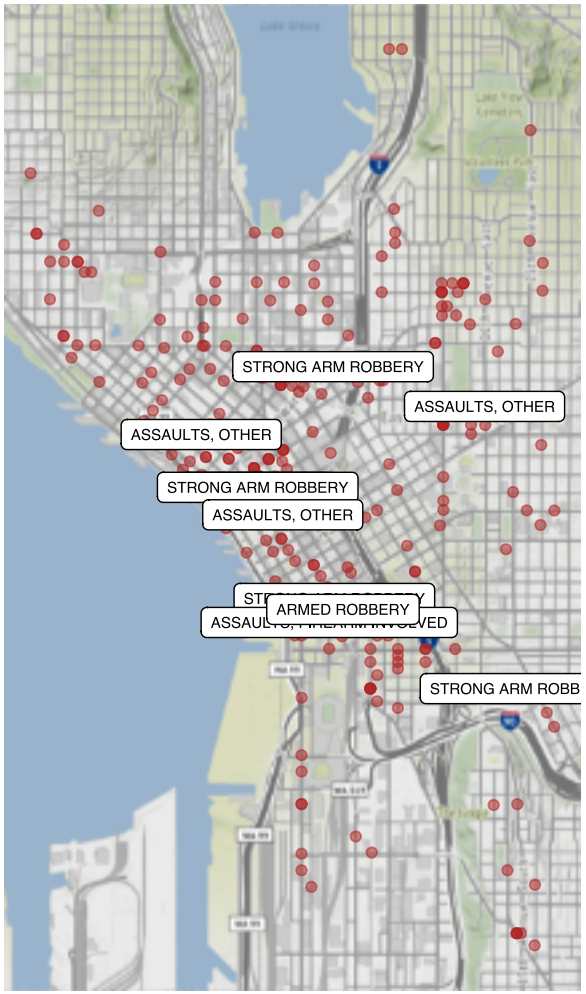
We'll **label** these observations!

Labels

Now let's plot the events and label them with `geom_label()`:

```
qmplot(data = downtown,
       x = Longitude,
       y = Latitude,
       maptype = "toner-lite",
       color = I("firebrick"),
       alpha = I(0.5)) +
  geom_label(data = assaults,
       aes(label = assault_label),
       size=2)
```

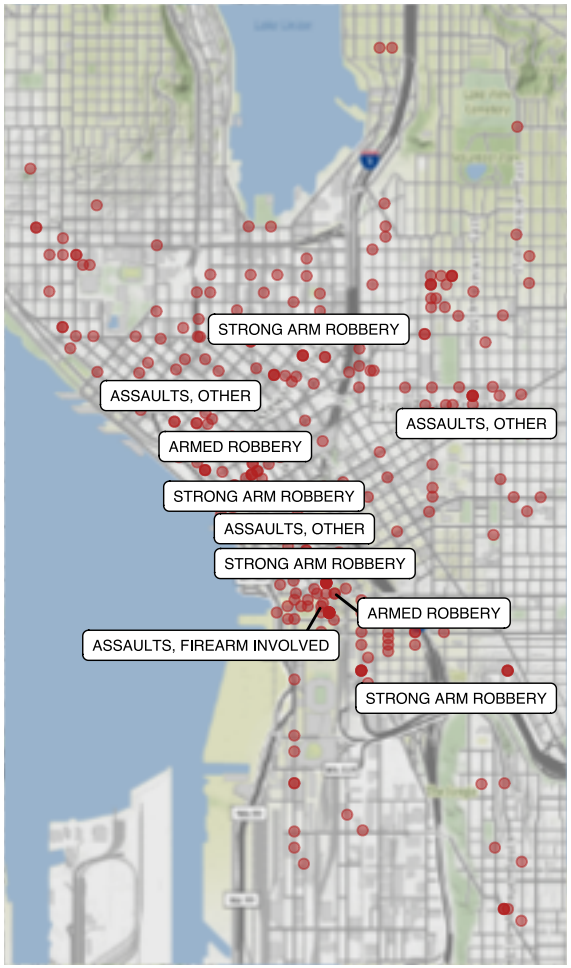
Note that one dataframe is used for points (`downtown`) and another for labels (`assaults`)!!



Fixing Overlapping Labels

The `ggrepel` package lets use fix or reduce overlapping labels using the function `geom_label_repel()`.

```
library(ggrepel)
qmplot(data =
    downtown,
    x = Longitude,
    y = Latitude,
    maptype = "toner-lite",
    color = I("firebrick"),
    alpha = I(0.5)) +
  geom_label_repel(
    data = assaults,
    aes(label = assault_label),
    size=2)
```



Lab/Homework

For the remainder of class, we'll use breakout rooms on Zoom to work on homework. Please *assign yourself* to a breakout room, then start working on Homework 8!

- The first part is based on Lecture 8 (strings)
  - This includes lab questions from last lecture!
- The second part is based on Lecture 9 (mapping)
  - This includes an analysis of the restaurant data from last week!

You can call me to your breakout room in Zoom! Otherwise, I will be moving between rooms.