1. The power of data science

I believe a certain uncle once said to his benevolent nephew

> *With great power comes great responsibility*

This, of course, is an oft-quoted Spider-man line. I don't think any individual need bear the responsibility of a superhero, but it is important to understand the value of technical analysis skills in the context of modern social issues. Within the workplace, data science skills can greatly increase your utility as an employee or prospective applicant. What is not always mentioned is that these same skills can be applied to positively impact your community. The concept of **open science** has led to new standards in **open data**, and there is an exciting plethora of raw information ready to be probed for insights. Organizations focused on *data science for social good* are rapidly growing, and the volunteer soup kitchen seems to have a 21st-century rendition.

Many local and federal governments support access to interesting datasets; as this trend grows, the utility of open information becomes more robust. Data is begging for audacious volunteers to poke and prod it, and make use of the raw input in an impactful way. We can now fight crime as a vigilante - all from behind a computer.

In this notebook, we will explore San Francisco crime data in order to understand the relationship between civilian-reported incidents of crime and police-reported incidents of crime. Along the way we will use table intersection methods to subset our data, aggregation methods to calculate important statistics, and simple visualizations to understand crime trends.

In [15]:
```
# Load required packages
library(tidyverse)
library(lubridate)

# Read in incidents dataset
incidents <- read_csv("datasets/downsample_police-department-incidents.csv")

# Read in calls dataset
calls <- read_csv("datasets/downsample_police-department-calls-for-service.cs
v")

print('Done!')
```

```
Parsed with column specification:
cols(
  IncidntNum = col_double(),
  Category = col_character(),
  Descript = col_character(),
  DayOfWeek = col_character(),
  Date = col_datetime(format = ""),
  Time = col_time(format = ""),
  PdDistrict = col_character(),
  Resolution = col_character(),
  Address = col_character(),
  X = col_double(),
  Y = col_double(),
  Location = col_character(),
  PdId = col_double()
)
Parsed with column specification:
cols(
  `Crime Id` = col_double(),
  Descript = col_character(),
  `Report Date` = col_datetime(format = ""),
  Date = col_datetime(format = ""),
  `Offense Date` = col_datetime(format = ""),
  `Call Time` = col_time(format = ""),
  `Call Date Time` = col_datetime(format = ""),
  Disposition = col_character(),
  Address = col_character(),
  City = col_character(),
  State = col_character(),
  `Agency Id` = col_double(),
  `Address Type` = col_character(),
  `Common Location` = col_character()
)

[1] "Done!"
```

In [16]:
```
# Everything looks ready. Let's beggin.
```

2. First poke and prod

First things first: we need to wrap our heads around the data in order to understand *what* we have. Let's `glimpse()` the data to see if there are any variables in the two datasets that are the same or similar. Then we can ask an investigative question about these variables, and return a simple statistic such as a frequency count.

In [17]:
```
# Glimpse the structure of both datasets
glimpse(incidents)
glimpse(calls)

# Aggregate the number of reported incidents by Date
daily_incidents <- incidents %>%
    count(Date, sort = TRUE) %>%
    rename(n_incidents = n)

# Aggregate the number of calls for police service by Date
daily_calls <- calls %>%
    count(Date, sort = TRUE) %>%
    rename(n_calls = n)
```

```
Observations: 84,000
Variables: 13
$ IncidntNum <dbl> 176122807, 160569314, 160362475, 160435298, 90543656, 1
8...
$ Category   <chr> "LARCENY/THEFT", "ASSAULT", "ROBBERY", "KIDNAPPING", "M
I...
$ Descript   <chr> "GRAND THEFT FROM UNLOCKED AUTO", "BATTERY", "ROBBERY,
B...
$ DayOfWeek  <chr> "Saturday", "Thursday", "Tuesday", "Friday", "Tuesday",
...
$ Date       <dttm> 2017-05-13, 2016-07-14, 2016-05-03, 2016-05-27, 2009-0
5...
$ Time       <time> 10:20:00, 16:00:00, 14:19:00, 23:57:00, 07:40:00, 18:0
0...
$ PdDistrict <chr> "SOUTHERN", "MISSION", "NORTHERN", "SOUTHERN", "TARAVA
L"...
$ Resolution <chr> "NONE", "NONE", "ARREST, BOOKED", "ARREST, BOOKED", "LO
C...
$ Address    <chr> "800 Block of BRYANT ST", "MISSION ST / CESAR CHAVEZ S
T"...
$ X          <dbl> -122.4034, -122.4182, -122.4299, -122.4050, -122.4612,
-...
$ Y          <dbl> 37.77542, 37.74817, 37.77744, 37.78512, 37.71912, 37.80
6...
$ Location   <chr> "{'latitude': '37.775420706711', 'human_address': '{\"a
d...
$ PdId       <dbl> 1.761228e+13, 1.605693e+13, 1.603625e+13, 1.604353e+13,
...
Observations: 100,000
Variables: 14
$ `Crime Id`        <dbl> 163003307, 180870423, 173510362, 163272811, 1728
1...
$ Descript          <chr> "Bicyclist", "586", "Suspicious Person", "911 Dr
o...
$ `Report Date`     <dttm> 2016-10-26, 2018-03-28, 2017-12-17, 2016-11-22,
...
$ Date              <dttm> 2016-10-26, 2018-03-28, 2017-12-17, 2016-11-22,
...
$ `Offense Date`    <dttm> 2016-10-26, 2018-03-28, 2017-12-17, 2016-11-22,
...
$ `Call Time`       <time> 17:47:00, 05:49:00, 03:00:00, 17:39:00, 08:54:0
0...
$ `Call Date Time`  <dttm> 2016-10-26 17:47:00, 2018-03-28 05:49:00, 2017-
1...
$ Disposition       <chr> "GOA", "HAN", "ADV", "NOM", "GOA", "ADV", "REP",
...
$ Address           <chr> "The Embarcadero Nor/kearny St", "Ingalls St/van
...
$ City              <chr> "San Francisco", "San Francisco", "San Francisc
o"...
$ State             <chr> "CA", "CA", "CA", "CA", "CA", "CA", "CA", "CA",
"...
$ `Agency Id`       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1...
$ `Address Type`    <chr> "Intersection", "Intersection", "Intersection",
"...
```

```
$ `Common Location` <chr> NA, NA, NA, NA, NA, NA, "Midori Hotel Sro #612,
S...
```

In [18]:
```
head(daily_calls)
```

Date	n_calls
2016-09-21	165
2017-09-14	165
2017-06-01	162
2016-06-24	161
2016-06-25	160
2016-09-06	160

In [19]:
```
# Let's make sure it looks like it's supposed to look.
```

3. Mutating join

Now that we have a better understanding of what variables are present in our information set we can see there are shared variables that will allow us to ask a wider variety of questions. We can inquire about the relationship between civilian-reported incidents and police-reported incidents by the date on which the incidents were documented. To combine this information we will perform a type of mutating join between the data frames. The new dataset structure preserves only days on which both civilians reported incidents and police encountered incidents.

In [20]:
```
# Join data frames to create a new "mutated" set of information
shared_dates <- inner_join(daily_calls, daily_incidents)

# Take a glimpse of this new data frame
glimpse(shared_dates)
```

```
Joining, by = "Date"

Observations: 776
Variables: 3
$ Date        <dttm> 2016-09-21, 2017-09-14, 2017-06-01, 2016-06-24, 2016-
0...
$ n_calls     <int> 165, 165, 162, 161, 160, 160, 159, 158, 158, 157, 156,
...
$ n_incidents <int> 60, 97, 100, 105, 100, 89, 109, 97, 93, 72, 73, 68, 8
0,...
```

In [21]:
```
head(shared_dates)
```

Date	n_calls	n_incidents
2016-09-21	165	60
2017-09-14	165	97
2017-06-01	162	100
2016-06-24	161	105
2016-06-25	160	100
2016-09-06	160	89

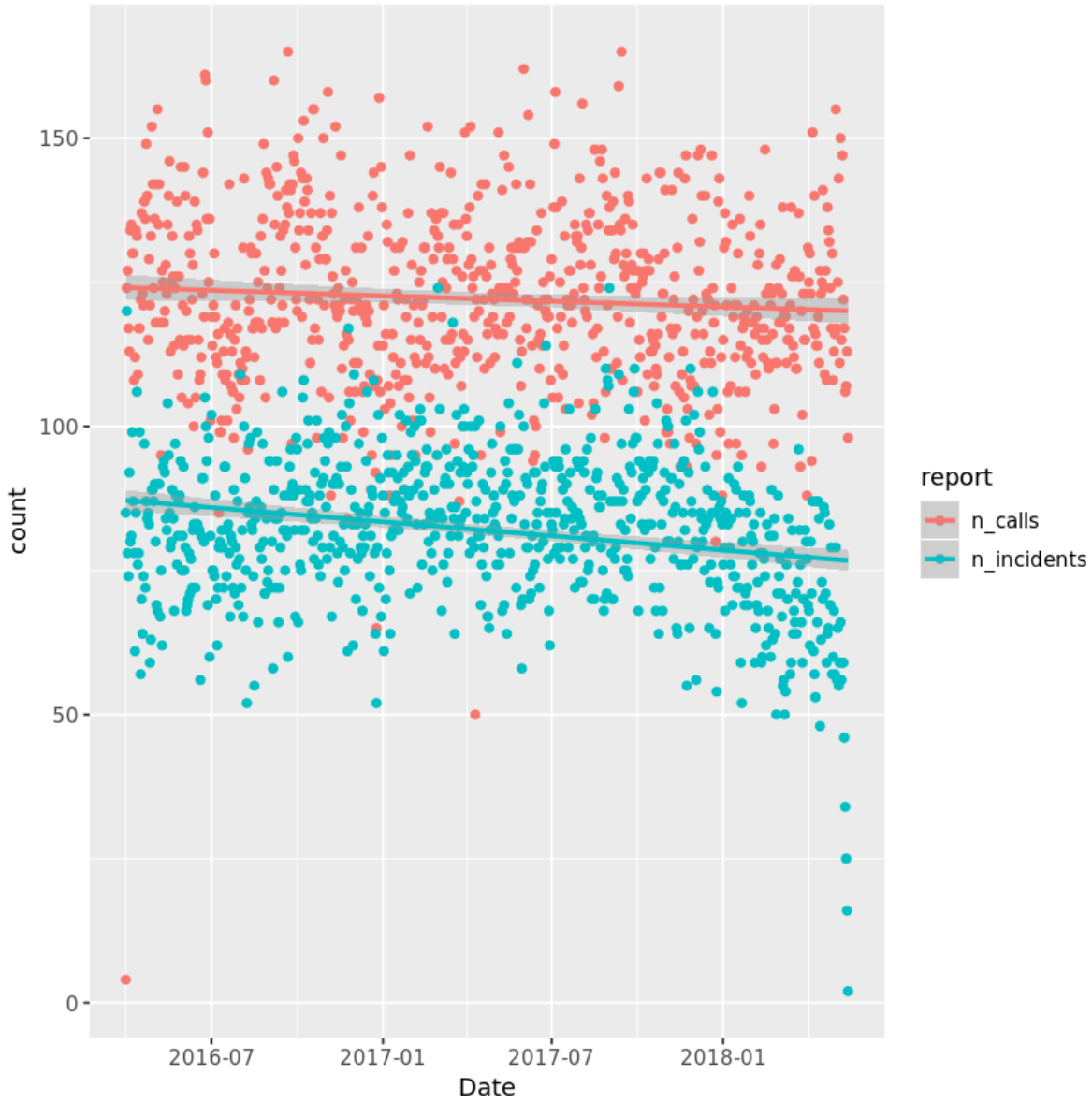
4. Inspect frequency trends

We now have a data frame that contains new information generated by combining datasets. In order to understand this new information we must visualize it. And I don't mean just giving the data a `glimpse()` - a table of raw information limits our comprehension of crime patterns. A picture is worth a thousand words, right? So let's try to represent the information in a concise way that will lead to other questions. We will look at the frequency of calls and incidents across time to help discern if there is a relationship between these variables.

Often times restructuring of data is required to perform new operations like plotting. `ggplot2` is amenable to "long format" data rather than "wide format" data. To plot time series data in `ggplot2` we would like one column to represent the dates, one column to represent the counts, and one column to map each observation to either `n_calls` or `n_incidents`. This allows us to pass a single column to each `x`, `y`, and `color` argument in `ggplot()`. We want the `key` column in `gather()` to define the columns `n_incidents` and `n_calls`. The `value` column will define each of these variable's corresponding counts. By leaving the `Date` column out of `key`, the result is a long and narrow data frame with multiple rows for each date observation.

In [22]:
```
# Gather into long format using the "Date" column to define observations
plot_shared_dates <- shared_dates %>%
  gather(key = report, value = count, -Date)

# Plot points and regression trend lines
ggplot(plot_shared_dates, aes(x = Date, y = count, color = report)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x)
```



5. Correlation between trends

A more quantitive way to discern the relationship between 2 variables is to calculate a correlation coefficient between vectors of data. This statistic is represented between a range from -1 to +1; specifically it represents the linear dependence between two sets of data. The correlation coefficient can be interpreted as perfect negative correlation when -1, no correlation whatsoever when 0, and perfect positive correlation when +1. We will look at 2 different but related correlation coefficients in order to understand how our interpretation of statistics can greatly influence the conclusions we come to.

We will first check if there is a correlation between the frequency of incidents and calls on a day-to-day basis. However, this may be too granular of a statistic - year over year daily correlations are probably only likely on big events (New Year's Eve, Halloween, Bay to Breakers). It may be helpful to take a broader view of inherent trends by summarising the data into monthly counts and calculating a correlation coefficient.

In [23]:
```
# Calculate correlation coefficient between daily frequencies
daily_cor <- cor(shared_dates$n_calls, shared_dates$n_incidents)
daily_cor

# Summarize frequencies by month
correlation_df <- shared_dates %>%
  mutate(month = month(Date)) %>%
  group_by(month) %>%
  summarize(n_incidents = sum(n_incidents),
            n_calls = sum(n_calls))

# Calculate correlation coefficient between monthly frequencies
monthly_cor <- cor(correlation_df$n_incidents, correlation_df$n_calls)
monthly_cor
```

0.146968821239074

0.970682977463344

6. Filtering joins

When working with relational datasets there are situations in which it is helpful to subset information based on another set of values. Remember mutating joins? Filtering joins are a complementary type of join which allows us to keep all specific cases within a data frame while preserving the structure of the data frame itself. It will be helpful to have all the information from each police reported incident and each civilian call on their shared dates so we can calculate similar statistics from each dataset and compare results. In this case we will use `shared_dates` to subset down both the full `calls` and `incidents` data frames.

In [24]:
```
# Subset calls to police by shared_dates
calls_shared_dates <- semi_join(calls, shared_dates, , by = c("Date" = "Date"
))
glimpse(calls_shared_dates)
```

```
Observations: 94,720
Variables: 14
$ `Crime Id`        <dbl> 163003307, 180870423, 173510362, 163272811, 1728
1...
$ Descript          <chr> "Bicyclist", "586", "Suspicious Person", "911 Dr
o...
$ `Report Date`     <dttm> 2016-10-26, 2018-03-28, 2017-12-17, 2016-11-22,
...
$ Date              <dttm> 2016-10-26, 2018-03-28, 2017-12-17, 2016-11-22,
...
$ `Offense Date`    <dttm> 2016-10-26, 2018-03-28, 2017-12-17, 2016-11-22,
...
$ `Call Time`       <time> 17:47:00, 05:49:00, 03:00:00, 17:39:00, 08:54:0
0...
$ `Call Date Time`  <dttm> 2016-10-26 17:47:00, 2018-03-28 05:49:00, 2017-
1...
$ Disposition       <chr> "GOA", "HAN", "ADV", "NOM", "GOA", "REP", "GOA",
...
$ Address           <chr> "The Embarcadero Nor/kearny St", "Ingalls St/van
...
$ City              <chr> "San Francisco", "San Francisco", "San Francisc
o"...
$ State             <chr> "CA", "CA", "CA", "CA", "CA", "CA", "CA", "CA",
"...
$ `Agency Id`       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1...
$ `Address Type`    <chr> "Intersection", "Intersection", "Intersection",
"...
$ `Common Location` <chr> NA, NA, NA, NA, NA, "Midori Hotel Sro #612, Sf",
...
```

In [25]:
```
# Subset calls to police by shared_dates
calls_shared_dates <- semi_join(calls, shared_dates, , by = c("Date" = "Date"
))

# Perform a sanity check that we are using this filtering join function approp
riately
identical(sort(unique(calls_shared_dates$Date)), sort(unique(shared_dates$Date
)))

# Filter recorded incidents by shared_dates
incidents_shared_dates <- semi_join(incidents, shared_dates, , by = c("Date" =
"Date"))
```

```
TRUE
```

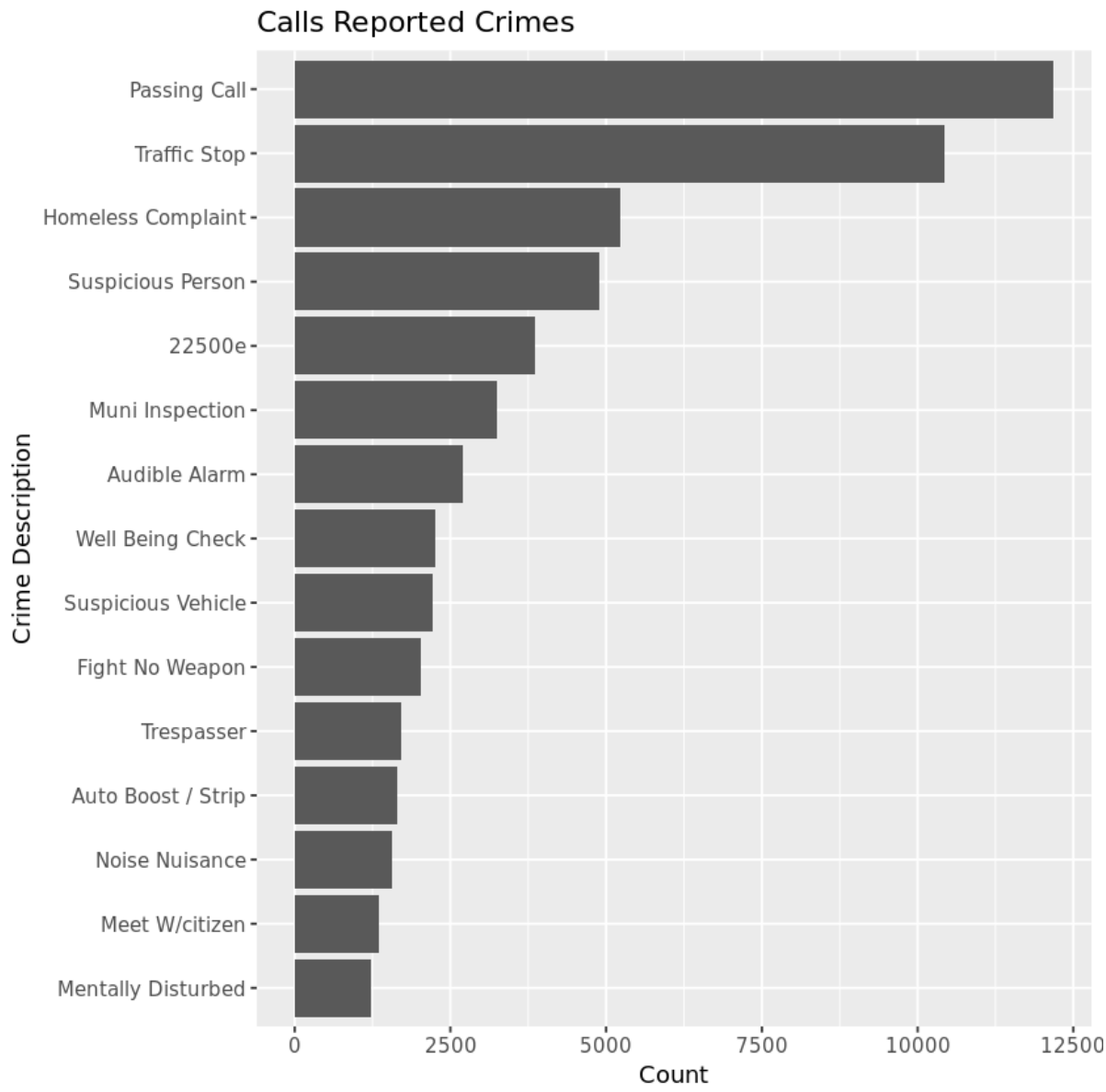
7. True crime

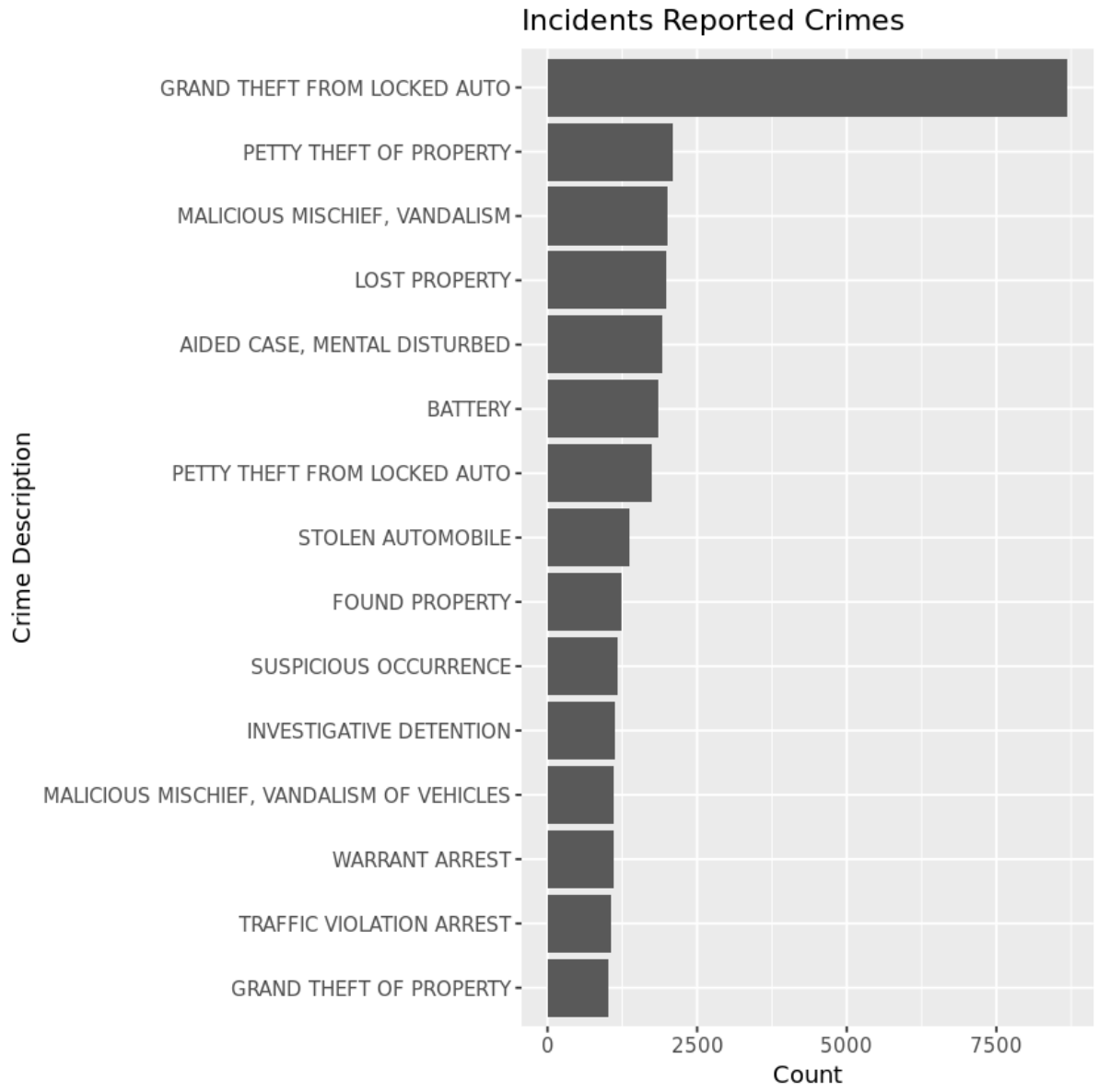
Back to some data viz! Now we need to see what the data look like after joining the datasets. Previously we made a scatterplot and fit a linear model to the data to see if there was a trend in the frequency of calls and the frequency of reported incidents over time. Scatterplots are a great tool to look at overall trends of continuous data. However, to see trends in categorical data, we need to visualize the ranked order of the variables to understand their levels of importance.

In [26]:
```
# Create a bar chart of the number of calls for each crime
plot_calls_freq <- calls_shared_dates %>%
  count(Descript) %>%
  top_n(15, n) %>%
  ggplot(aes(x = reorder(Descript, n), y = n)) +
      geom_bar(stat = 'identity') +
      ylab("Count") +
      xlab("Crime Description") +
      ggtitle("Calls Reported Crimes") +
      coord_flip()


# Create a bar chart of the number of reported incidents for each crime
plot_incidents_freq <- incidents_shared_dates %>%
  count(Descript) %>%
  top_n(15, n) %>%
  ggplot(aes(x = reorder(Descript, n), y = n)) +
      geom_bar(stat = 'identity') +
      ylab("Count") +
      xlab("Crime Description") +
      ggtitle("Incidents Reported Crimes") +
      coord_flip()

# Output the plots
plot_calls_freq
plot_incidents_freq
```


Calls Reported Crimes
Crime Description
Passing Call
Traffic Stop
Homeless Complaint
Suspicious Person
22500e
Muni Inspection
Audible Alarm
Well Being Check
Suspicious Vehicle
Fight No Weapon
Trespasser
Auto Boost / Strip
Noise Nuisance
Meet W/citizen
Mentally Disturbed
0
2500
5000
7500
10000
12500
Count



8. Grand theft auto

Interesting - far and away the crime of highest incidence is "GRAND THEFT FROM LOCKED AUTO". This category probably captures many crimes of opportunity where unsupervised vehicles are broken into. However, there **are** vigilantes out there trying to prevent crime! The 12th most civilian reported crime is "Auto Boost / Strip"! Maybe these civilians are truly helping to prevent crime. Yet, this is probably only the case where the location of a called-in-crime is similar to the location of crime incidence. Let's check to see if the locations of the most frequent civilian reported crime and police reported crime are similar.

In [27]:
```
"Auto Boost / Strip" %in% (calls_shared_dates$Descript)
names(calls_shared_dates)
```

TRUE

'Crime Id' 'Descript' 'Report Date' 'Date' 'Offense Date' 'Call Time' 'Call Date Time' 'Disposition' 'Address' 'City' 'State' 'Agency Id' 'Address Type' 'Common Location'

In [28]:
```
# Arrange the top 10 locations of called in crimes in a new variable
location_calls <- calls_shared_dates %>%
  filter(Descript == "Auto Boost / Strip") %>%
  count(Address) %>%
  arrange(desc(n))%>%
  top_n(10, n)

# Arrange the top 10 locations of reported incidents in a new variable
location_incidents <- incidents_shared_dates %>%
  filter(Descript == "GRAND THEFT FROM LOCKED AUTO") %>%
  count(Address) %>%
  arrange(desc(n))%>%
  top_n(10, n)

# Print the top locations of each dataset for comparison
location_calls
location_incidents
```

Address	n
1100 Block Of Point Lobos Av	21
3600 Block Of Lyon St	20
100 Block Of Christmas Tree Point Rd	18
1300 Block Of Webster St	12
500 Block Of 6th Av	12
800 Block Of Vallejo St	10
1000 Block Of Great Hy	9
100 Block Of Hagiwara Tea Garden Dr	7
1100 Block Of Fillmore St	7
3300 Block Of 20th Av	7
800 Block Of Mission St	7

Address	n
800 Block of BRYANT ST	441
500 Block of JOHNFKENNEDY DR	89
1000 Block of POINTLOBOS AV	84
800 Block of MISSION ST	61
2600 Block of GEARY BL	38
3600 Block of LYON ST	36
1300 Block of WEBSTER ST	35
1100 Block of FILLMORE ST	34
22ND ST / ILLINOIS ST	33
400 Block of 6TH AV	30

9. Density map

It appears the datasets share locations where auto crimes occur and are reported most frequently - such as on Point Lobos Avenue, Lyon Street, and Mission Street. It would be great to plot co-occurrence of these locations to visualize overlap, however we only have longitude and latitude data for police reported `incidents`. No matter, it will still be very valuable to inspect the frequency of auto crime occurrence on a map of San Francisco. This will give us immediate insight as to where auto crimes occur. Most importantly, this visualization will provide a powerful means of communication.

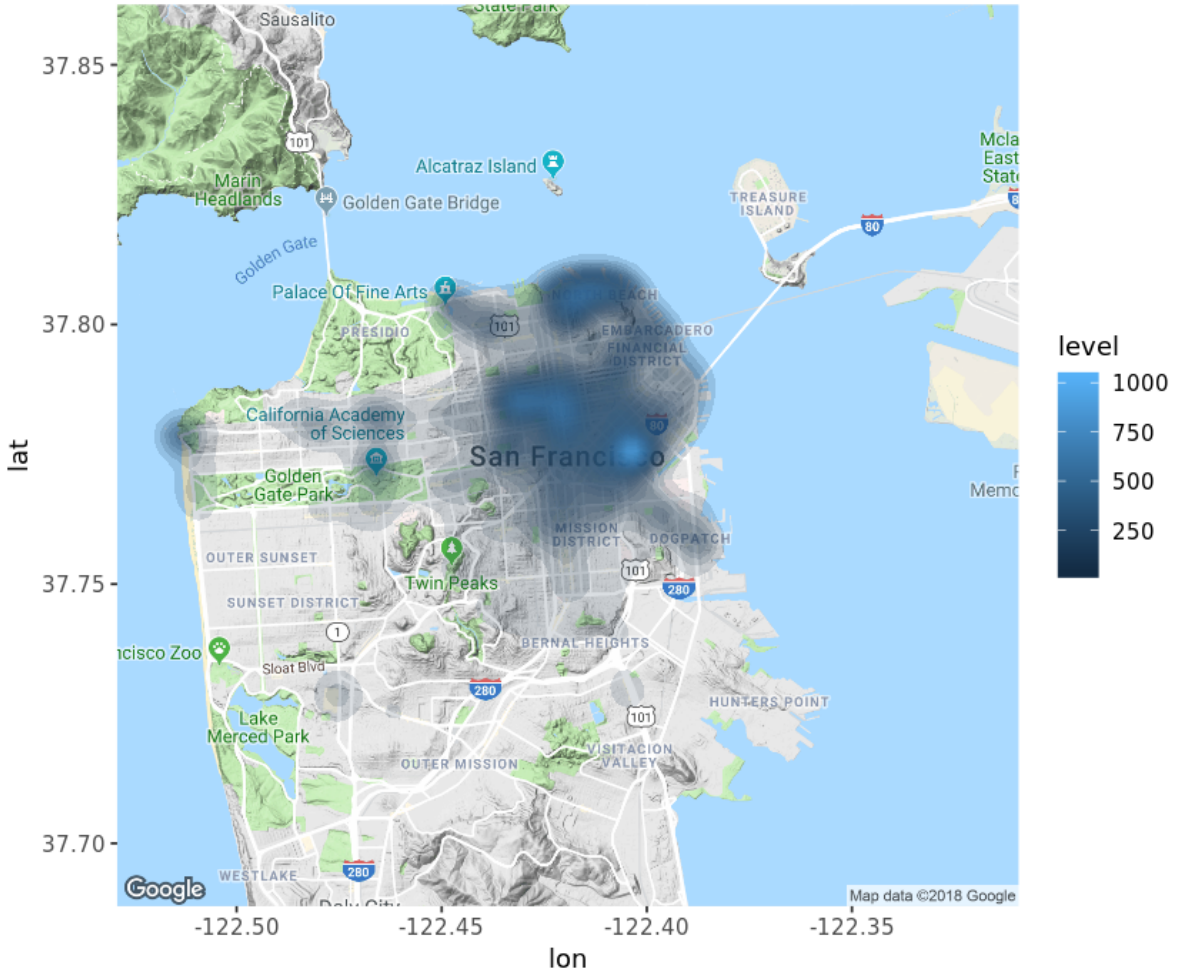
As we ask deeper questions it becomes obvious that many details of each dataset are not standardized (such as the `Address` variable in each data frame and the lack of exact location data in `calls`) and thus require more advanced analysis. Now this is the fun part - applying your technical creativity to difficult questions. Go forth from here, check out the original dataset (https://www.kaggle.com/san-francisco/sf-police-calls-for-service-and-incidents), and ask some new questions!

In [29]:
```
# Load ggmap
library(ggmap)

# Read in a static map of San Francisco
sf_map <- readRDS("datasets/sf_map.RDS")

# Filter grand theft auto incidents
auto_incidents <- incidents_shared_dates %>%
    filter(Descript == "GRAND THEFT FROM LOCKED AUTO")

# Overlay a density plot of auto incidents on the map
ggmap(sf_map) +
  stat_density_2d(
    aes(x = X, y = Y, fill = ..level..), alpha = 0.15,
    size = 0.01, bins = 30, data = auto_incidents,
    geom = "polygon")
```



In [30]:
```
# Alright. So that's why GTA looks so much like San Francisco.
```