Data Science Research Circle

Pomona College Data Science Research Circle

2020-05-29

Contents

Chapter 1

Preface

Welcome! This page is the manifestation of a semester-long investigative project conducted by a Data Science Research Circle at Pomona College in Claremont, CA. The research circle consisted of seven students including: Amber Lee (‘22), Arm Wonghirundacha (‘22), Emma Godfrey (‘21), Ethan Ong (‘21), Ivy Yuan (‘21), Oliver Chang (‘22), and Will Gray (‘22). Additionally the research circle was advised by Professors Jo Hardin and Ghassan Sarkis from the Department of Mathematics. The work is motivated by recent literature and investigative analysis on policing practices around the country, particularly from the Stanford Open Policing Project.

The Stanford Open Policing Project is an ongoing research effort to gather and publish police stop data from around the United States. The project is a collaboration between the Stanford Computational Journalism Lab and the Stanford School of Engineering led by professor Cheryl Phillips and Sharad Goel. Beginning in 2015, the Stanford Open Policing project has requested both statewide and city-wide public information about police stops; requests were filed for all 50 states and over 100 cities. On June 19th 2017, the project published the collected data on their website openpolicing.stanford.edu and as of 2020, the website holds data from 42 states totalling 255 million data points. 221 million of these stops were from 33 state patrol datasets collected and the remaining 34 million were from 56 city-wide datasets.

One of the primary focuses of the Stanford Open Policing Project is traffic stop data. More than 50,000 traffic stops occur every day — 20 million in a year — yet prior to the debut of the Open Policing Project database, there was a lack of reliable, comprehensive, and standardized national datasets on these traffic stop data to analyze. The Open Policing Project provides a unique opportunity to investigate these traffic stops, with particular intention to work towards a definitive test of racial profiling. The interdisciplinary work towards developing a definitive test of racial profiling is grounded in social and political

relevance – the true extent of racial profiling would be crucial tool for informing police department trainings, protecting Fourth Amendment rights of minority defendants, and validating the qualitative experiences of historically-oppressed groups (Goel 2017).

The motivation for this project is to continue the analysis of the Stanford Open Policing Project dataset. This includes reproducing previous findings and applying methodologies used on smaller datasets to the Stanford nation-wide database. The three topics of interest before beginning EDA for this project were the following: the enactment of seatbelt laws and its effect on traffic stops, night versus day stops, and discrepancies of search rates. Not all were pursued, but each provided a unique direction for data analysis. The report is organized into several parts: I. Preface, II. Literature Review, III. Description of Data, IV. Databases/Using SQL, V. Exploratory Data Analysis, and VI. Limitations and Discussion.

Chapter 2

Functions

This section explains the various functions used throughout our research.

2.1 Data Uploading Functions

The functions here serve as cleaning functions such that a ready to use dataset gets uploaded to the database as a data table.

convertLogicalToInt function: input: dataset output: dataset with logical values set to binary

```
convertLogicalToInt <- function(input_df){
  vars <- sapply(input_df, class) %>%
  list.which(.=='logical')

  input_df[,vars] <- sapply(input_df[,vars], as.numeric)
  input_df
  return(input_df)
}
```

getDFName: input: name of dataset (this could also be name of link from web scrape) output: clean dataset name that includes state and city

```
getDFName <- function(input_string){
  tmp <- str_split(input_string, '_', simplify = TRUE)
  tmp

  state <- str_to_upper(tmp[,2])

  if(tmp[,4] != "2019"){
      cityName <- paste(tmp[,3], tmp[,4], sep="")
```

```
      name <- paste(state, cityName, sep="")
  } else {
      cityName <- tmp[,3]
      name <- paste(state, cityName, sep="")
  }
  name
  return(name)
}
```

uploadLinksToDatabase: input: string link from web scrape outout: 0 - indicating that the code finished executing This function will take in a RDS link and write the dataset to the database

```
uploadLinksToDatabase <- function(input_link){
  tmpDF <- readRDS(gzcon(url(input_link)))
  tmpDF <- convertLogicalToInt(tmpDF)
  name<- getDFName(input_link)

  dbWriteTable(con2, name, tmpDF, overwrite = TRUE)
  return(0)
}
```

2.2 Veil of Darkness Functions

Veil of Darkness functions work with lutz and lubridate to determine whether a stop took place in the dark.

The first step in getting the sunset and sunrise times is to get the coordinates for the city. To do this, we can Google search the city name and webscrape the Google search results.

The two code chunk below has functions that will perform this process. get_cityNames takes in a datatable's name and cleans it so that the google search engine result will return the desired page that has the coordinates.

```
get_cityNames <- function(name){
  check <- str_extract(name, "[a-z]+")
  if(check == "statewide"){
    return(state.name[grep(str_sub(name, 1,2), state.abb)])
  } else {
    return(check)
  }
}
```

get_coordinates takes in said clean city names and scrape the google search web result. The function will return a vector of doubles. The first index is for latitude and second index has longitude.

```
get_coordinates <- function(city){
  url <-
    paste("https://www.google.com/search?q=",
          city,
          "+lat+long&oq=sandiego+lat+long&aqs=chrome..69i57.2463j0j7&sourceid=chrome&ie=UTF-8")
  doc <- htmlParse(readLines(url), asText=TRUE)
  links <- xpathSApply(doc, "//div[@class='kCrYT']", xmlValue)
  clean_coor <- as.list(str_split(links[2], ","))
  lat <- as.numeric(str_extract(clean_coor[[1]][1], "\\d+\\.*\\d*"))
  long <- -1*as.numeric(str_extract(clean_coor[[1]][2], "\\d+\\.*\\d*"))
  x <- c(lat,long)
  return(x)
}

clean_names <- list()
clean_names <- lapply(datasets_of_interest, get_cityNames)
coordinates <- lapply(clean_names,get_coordinates)
```

Next, we will define two helper functions in getting that call the lutz package and retrieve the times.

outsunriseset input: latitude (dbl), longitude(dbl), date(Date or Posix), timezime (tz), direction(string) output: Date with time of the desired sun direction

```
oursunriseset <- function(latitude, longitude, date, timezone, direction) {
  date.lat.long <- data.frame(date = date, lat = latitude, lon = longitude)
  if(direction == "sunset"){
    # call getSunlightTimes from the lutz package
    getSunlightTimes(data = date.lat.long, keep=direction, tz=timezone)$sunset
  } else if(direction == "sunrise"){
    getSunlightTimes(data = date.lat.long, keep=direction, tz=timezone)$sunrise
  } else if (direction == "dusk"){
    getSunlightTimes(data = date.lat.long, keep=direction, tz=timezone)$dusk
  } else if (direction == "dawn"){
    getSunlightTimes(data = date.lat.long, keep=direction, tz=timezone)$dawn
  }
}
```

time_to_minute is the helper function the Stanford Open Policing Project uses in their tutorial to help convert times into a numeric values that's easier to manipulate - this will be useful when splicing out times between sunset and dusk to remove ambiguouity in the intertwilight zone. input: time (character) output: minutes (double)

```
time_to_minute <- function(time) {
  hour(hms(time)) * 60 + minute(hms(time))
}
```

add_night_day function utilizes oursunriseset function to mutate sunrise and sunset times and a binary variables to see if the stop happened in the night of day. In addition, this function takes out the intertwilightzone i.e. stops between sunset and dusk and dawn and sunrise.

```
add_night_day <- function(city_df, time_zone, lat, long){

    sunset_times <- city_df %>% distinct(date) %>% mutate(date = as.Date(ymd(date, tz =
  mutate(sunset = oursunriseset(lat, long, date, time_zone, direction="sunset"),
         dusk = oursunriseset(lat, long, date, time_zone, direction="dusk"),
         dawn = oursunriseset(lat, long, date, time_zone, direction="dawn"),
         sunrise = oursunriseset(lat, long, date, time_zone, direction="sunrise"),
         sunset = format(sunset, "%H:%M:%S"),
         dusk = format(dusk, "%H:%M:%S"),
         dawn = format(dawn, "%H:%M:%S"),
         sunrise = format(sunrise, "%H:%M:%S"),
         sunset_min=time_to_minute(sunset),
         dusk_min=time_to_minute(dusk),
         dawn_min=time_to_minute(dawn),
         sunrise_min=time_to_minute(sunrise)) %>% drop_na()

  city_df <- city_df %>% drop_na() %>% mutate(date=as.Date(ymd(date, tz = time_zone)))
    left_join(sunset_times, by="date") %>% drop_na() %>%
    mutate(minute = time_to_minute(time),
           minutes_after_dark = minute - dusk_min,
           is_dark = as.factor(minute > dusk_min | minute < dawn_min)) %>%
        # filter out amiguous time between sunset and dusk and dawn and sunrise
  filter(!(minute > sunset_min & minute < dusk_min),
         !(minute < sunrise_min & minute > dawn_min)) %>% select(subject_race, is_dark)

  # select(subject_race, search_conducted, subject_age, is_dark)
  return(city_df)
}
```

2.3 Nationwide Functions

These are functions used when analyzing multiple datatables from the database.

relevant_dataests: input: - all dataset names which can be acquired through SHOW TABLES sql command - A vector containing string of variables names we wish to disect output: a character string of dataset names

```
relevant_datasets <- function(all_dataset_names, variables_of_interest){

  # create empty vector
  datasets_of_interest <- c()
```

```
  for(city in all_dataset_names){

    # run only if table is not empty
    check_command = paste("SELECT 1 FROM", city, sep=" ", "LIMIT 1")
    count <- DBI::dbGetQuery(con, check_command)

    if(nrow(count) != 0){
      # cancenate SQL query string
      command <- paste("EXPLAIN", city, sep = " ")
      field_vector <- unlist(as.list(DBI::dbGetQuery(con, command))$Field,
                             use.names = FALSE)

      # of_interest_book is TRUE iff field_vector contains all the variables of interest
      of_interest_bool <- setequal(intersect(field_vector, variables_of_interest),
                                   variables_of_interest)

      # add dataset name to vector if of_interest_bool
      if(of_interest_bool){
        datasets_of_interest <- c(datasets_of_interest, city)
      }
    }
  }

  return(datasets_of_interest)
}
```

query_data: input: name (character) output: dataframe

logistic_regression input: - city dataset (dataframe) - name of city (character) output: a dataframe where the columns are the coefficients (only returns one row of coefficient matrix; this function is insteaded to be used in a for loop or mapply so that we run the logistic regression on every dataset)

```
coefficient_matrix <- data.frame("intercept" = numeric(), "subject_age" = numeric(), "subject_rac

logistic_regression <- function(city_dataset, name){
  # run logistic regression
  fitlog <- glm(formula = search_conducted ~ subject_race*subject_age, data = city_dataset, famil
  # record logistic regression coefficients
  coefficient_row_vector = t(fitlog$coefficients)
  # row bind each coefficient and dataset tibble with the coefficent_matrix
  coefficient_matrix <<- rbind(coefficient_matrix, cbind.data.frame(as.data.frame(coefficient_row
}
```

Chapter 3

Description of Data

The data was extracted from the Stanford Open Policing Project data hub. The Open Policing Project's website contains ninety city and state-wide standardized stop datasets in both RDS and CSV format. From there, we uploaded six datasets to a mySQL database and locally queried the data on our personal laptops via RMySQL package within R.

The possible variables within each dataset are listed below with description and corresponding units.

Variable	Description	Units
Stop Date	States the year, month, day	XXXX-XX-X
Stop Time	States the time of the stop	XX:XX:XX
Stop Location	States the street intersection of the stop	Names two st
Driver Race	Describes the race of driver as given on the driver's license	String represe
Driver Sex	Describes the sex orientation of driver	Binary Femal
Driver Age	Describes the age of the driver	Integer repres
Search Conducted	Describes whether there was a search conducted or not	True (search c
Contraband Found	Describes whether there was contraband found after a search was conducted	True (contrab
Citation Issued	Describes whether there was a citation issued to the driver	True (citation
Warning Issued	Describes whether there was a warning issued to the driver	True (warning
Frisk Performed	Describes whether there was a frisk performed on the driver	True (frisk pe
Arrest Made	Describes whether the driver was arrested	True (arrest m
Reason for Stop	Describes why the driver was stopped	String describ
Violation	Describes the violation when a citation or warning was issued	String represe

We focused on the datasets for San Francisco, San Diego, Oakland, Raleigh, Charlotte, and Durham, and San Antonio. Below, the "X" represents whether the dataset for that particular city includes the specified variable.

City	Number.of.Observations	Stop.Date	Stop.Time	Stop.Location	Driver.Race	Se
San Francisco	905,070	X	X	X	X	X
Oakland	133,405	X	X		X	X
San Diego	382,844	X	X	X	X	X
Charlotte	1,598,453	X	X		X	X
Durham	326,024	X	X		X	X
Raleigh	856,400	X	X		X	X
San Antonio	1,040,428	X	X	X	X	X

While many of the datasets included a multitude of the variables above, there were clear limitations. First, while the California datasets were rich in variables, the "reason for stop" was dubious at best. Not only did many reasons make no logical sense, but also up to four reasons were listed for a single observation. Moreover, there is no uniformity in the "reason for stop" for all datasets in California. In contrast, the datasets in North Carolina had a clean "reason for stop"; however, these datasets do not have latitude/longitude which limits the geographical comparisons we can make. Additionally, only a small fraction of the city and state datasets contained subject age; the scarcity of information regarding subject age makes it difficult to compare nationally. Lastly, only a few datasets, such as Chicago, contained the police officer's identification number. While we did not investigate the effect of the age, race, or other factors of individual police officers, it would be meaningful for more datasets to contain that information to draw national correlations.

3.1 Querying Datasets with RMySQL

First import the necesarry packages to work with SQL in R

```
library(XML)
library(RMySQL)
```

```
## Loading required package: DBI
```

```
library(tidyverse)
```

```
## -- Attaching packages -------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
## -- Conflicts ----------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(knitr)
library(rlist)
```

```
## Warning: package 'rlist' was built under R version 3.6.3
```

Next, setup a connection to the database

```
# Connect to database
con <- dbConnect(
  MySQL(), host = "traffic.st47s.com", user = "loading",
  password = "dataupload", dbname = "traffic")
```

dbGetQuery returns a dataframe, so storing it in a variable adds that dataframe to the global environment

```
raleigh_df <- DBI::dbGetQuery(con, "SELECT * FROM NCraleigh LIMIT 15")
```

3.1.1 Querying Subsets of dataset

Certain scenarios such as running a logistic regression need not every variable from a dataset - only the variables of interest. Moreover, running a logistic regression on every city with all columns is a taxing operation. This is where querying can be useful in selecting portions of the datasets deemed essential.

The logistic regression called for the race, age, sex, and data variables. The query_data function will takes in a database table name (city_name) and will select the desired variables from that table.

```
query_data <- function(city_name){
  # cancenate SQL query string
  command <-
    paste("SELECT subject_age, subject_race, subject_sex, date, search_conducted FROM",
          city_name, sep = " ")
  return(DBI::dbGetQuery(con, command))
}

# Make a list of datasets
datasets <- list()
datasets <- lapply(datasets_of_interest, query_data)
```

3.2 Web Scrapping and Uploading Datasets

This section will show the dataset upload process. The first step is to acquire all the RDS weblinks from the Stanford Policing Project Website.

The code here is written by Professor Sarkis and it will get all the rds files from the stanford policing project website. Then, it stores all the file names into a character string

```
url <- "https://openpolicing.stanford.edu/data/"
doc <- htmlParse(readLines(url), asText=TRUE)
links <- xpathSApply(doc, "//a/@href")
free(doc)

all_links <- as.character(links[grep('_.*rds', links)])
```

One could also get all the dataset file names for a specific state by adding the state's two character abbreviation e.g.

```
california_links <- as.character(links[grep('ca_.*rds', links)])
```

Another optional step is to splice certain links using base R indexing

```
# subset list to get desired links
all_links <- all_links[43:48]
all_links <- all_links[-c(1, 2, 5)]
```

Since the datasets are written to the database using the dbWriteTable function from RMySQL library, logical datatypes get interpreted as character, which will be tedious to deal with in future analysis. Therefore, the convertLogicalToInt function will convert columns that have logical datatypes into doubles (0 and 1).

convertLogicalToInt function: input: dataset output: dataset with logical values set to binary

```
convertLogicalToInt <- function(input_df){
  vars <- sapply(input_df, class) %>%
  list.which(.=='logical')

  input_df[,vars] <- sapply(input_df[,vars], as.numeric)
  input_df
  return(input_df)
}
```

Subsequently, dbWriteTable also needs a name for the table, which can parsed out of the weblinks. The getDFName function returns a suitable datatable name.

getDFName: input: name of dataset (this could also be name of link from web scrape) output: clean dataset name that includes state and city

```
getDFName <- function(input_string){
  tmp <- str_split(input_string, '_', simplify = TRUE)
tmp

state <- str_to_upper(tmp[,2])
```

```
if(tmp[,4] != "2019"){
    cityName <- paste(tmp[,3], tmp[,4], sep="")
    name <- paste(state, cityName, sep="")
} else {
    cityName <- tmp[,3]
    name <- paste(state, cityName, sep="")
}
name
return(name)
}
```

The penultimate step is to read the RDS file and write that to the database using the dbWriteTable function. The results from the two previous functions will serve as arguments for the dbWriteTable function.

uploadLinksToDatabase: input: string link from web scrape outout: 0 - indicating that the code finished executing This function will take in a RDS link and write the dataset to the database

```
uploadLinksToDatabase <- function(input_link){
  tmpDF <- readRDS(gzcon(url(input_link)))
  tmpDF <- convertLogicalToInt(tmpDF)
  name<- getDFName(input_link)

  dbWriteTable(con2, name, tmpDF, overwrite = TRUE)
  return(0)
}
```

lapply serves as a 'oneline' for loop. The argument all_links is a character string of links, which lapply will iterate through and call the function, uploadLinksToDatabase, on each link

```
lapply(all_links, uploadLinksToDatabase)
```

Chapter 4

Exploratory Data Analysis

4.1 Stop outcomes, day-to-night, in Oakland, CA

This section of data analysis incorporates our day variable with stop outcomes in the Oakland data set.

4.1.0.1 Set up

I use the following packages for this analysis. Suncalc, lutz (a cute acronym for "look up time zone"), and lubridate are necessary for calculating sunset and sunrise times. I also connect to our SQL server, but that code is excluded here because of privacy.

```
library(tidyverse)
library(RMySQL)
library(suncalc)
library(lutz)
```

```
## Warning: package 'lutz' was built under R version 3.6.3
```

```
library(lubridate)
```

Next, I query the entire Oakland data set to be available locally. I also clean the data by removing the entries that do not record data, parsing my date and time variables with lubridate, and creating a POSIX type time variable that will serve as an input for suncalc functions.

```
CAoak <- DBI::dbGetQuery(con, "SELECT * FROM CAoakland")

# Lubridate
CAoak <- CAoak %>%
```

```
  filter(!is.na(date)) %>%
  mutate(nice_date = ymd(date),
         nice_year = year(nice_date),
         nice_month = month(nice_date),
         nice_day = day(nice_date),
         nice_time = hms(time),
         nice_day_of_year = yday(date),

         #for sunset sunrise:
         posix_date_time = as.POSIXct(paste(nice_date, time), tz = "America/Los_Angeles
```

```
## Warning in .parse_hms(..., order = "HMS", quiet = quiet): Some strings failed to
## parse, or all strings are NAs
```

4.1.0.2 Set up sunset/sunrise times in CA Oakland

Next, I use the function we wrote, oursunriseset, and dplyr functions to create a new set of variables in the Oakland data. For each stop, we analyze the time of day and day of the year to categorize the stop as either occurring during the day (for which the new variable light would be "day") or during the night (for which light would be "night."

When I use the suncalc function getSunlightTimes, I let the input timezone be the Oakland timezone.

```
OaklandTZ <- lutz::tz_lookup_coords(37.75241810000001, -122.18087990000001, warn = F)

oursunriseset <- function(latitude, longitude, date, direction = c("sunrise", "sunset"

  date.lat.long <- data.frame(date = date, lat = latitude, lon = longitude)

  if(direction == "sunrise"){

    getSunlightTimes(data = date.lat.long, keep=direction, tz = OaklandTZ)$sunrise }els
      getSunlightTimes(data = date.lat.long, keep=direction, tz = OaklandTZ)$sunset } )
```

Next, I run oursunriseset on the Oakland data set, using mutate to create the intermediate variables, sunrise and sunset, and the light variable. I remove the traffic stop observations for which time was not recorded.

```
# add light variable
CAoak <- CAoak %>%

  # use oursunriseset function to return posixct format sunrise and sunset times
  mutate(sunrise = oursunriseset(lat, lng, nice_date, direction = "sunrise"),
         sunset = oursunriseset(lat, lng, nice_date, direction = "sunset")) %>%
```

```
  mutate(light = ifelse(posix_date_time > sunrise & posix_date_time < sunset, "day", "night")) %>

  # about 100 NA's to filter out

  filter(!is.na(light))

# knitr::kable((CAoak %>% select(posix_date_time, subject_race, light) %>% head()))
```

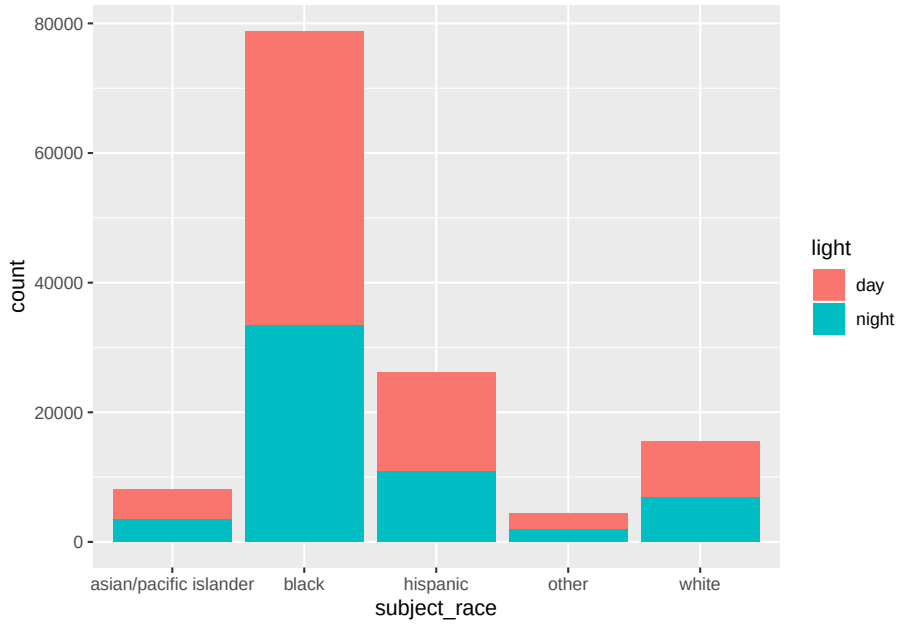
This short excerpt of the transformed Oakland data set shows how the time of stop and light variable correspond to one another.

4.1.0.3 Stop outcomes, by race and day/night

With the set up done, we can begin plotting stop outcomes related to day and night!

4.1.0.3.1 Counts, by race The bar chart shows the number of stops per race occurring during the day and night.

```
# see how drivers are stopped by race and light in absolute counts
ggplot(data = CAoak) +
  geom_bar(mapping = aes(x = subject_race, fill = light))
```



The bar chart relays effects of a slew of variables, such as driving behavior of a time of day, time of year, and racial group; traffic and patrol behavior for a

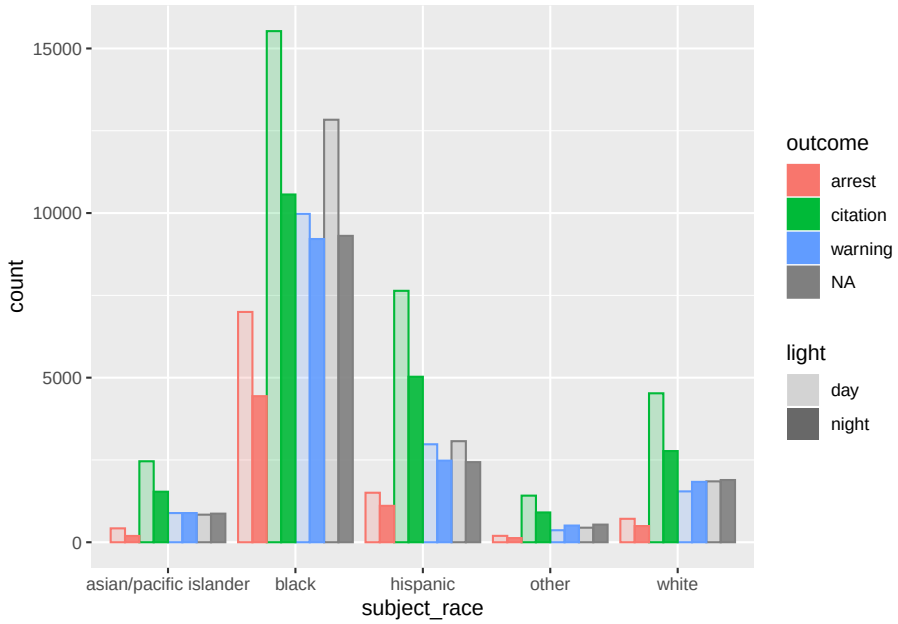
given time and location is also relayed. Thus, again, we cannot make definitive statements about racial profiling.

As in the veil of darkness assumption, the race of a motorist is harder to discern during the night than during the day. However, just because the day-night bars in each racial group looks to be about equal neither corroborate nor denies racial profiling. We are without information on the true traffic infringement rate per racial group in different times of the day.

4.1.0.4 Day/night outcomes of stop by race

Next, I examine how outcome of stop, race, and day/night interact. I pipe the Oakland data set into ggplot, allowing the color of each bar to represent a traffic stop outcome and the transparency of the bars to denote day and night stops.

```
# how are outcomes of stops affected by race and time of day?

CAoak %>%
  ggplot(aes(x = subject_race, fill = outcome, color = outcome, alpha = light)) +
  geom_bar(position = "dodge") +
  scale_alpha_manual(values = c(.2, .9))
```



Observations:

- For each type of outcome and race, stops during the day outnumber stops during the night. This may result from the Oakland police department's patrol patterns, driver behavior during the day, and/or another factor.

- Missing data regarding traffic outcome does not look to be evenly distributed among stopped motorists – missing data during the day and night look to be about the same for all racial groups except for Black.

- Citations look to be the most frequent outcome of traffic stops for all racial groups, followed by warning, then arrest (not including missing data.)

- Future directions could look more deeply into the probabilities of each of these outcomes for different racial groups, as the nature of a stricter outcome (being issued a citation rather than a warning) may not be uniformly distributed among racial groups. Furthermore, incorporating the light variable in this direction could be illuminating: are traffic patrols more lenient during the day or during the night once a motorist is stopped? Does the extent of this leniency differ for different racial groups?

4.2 Dataset-specific Exploration

4.2.1 [illegible]

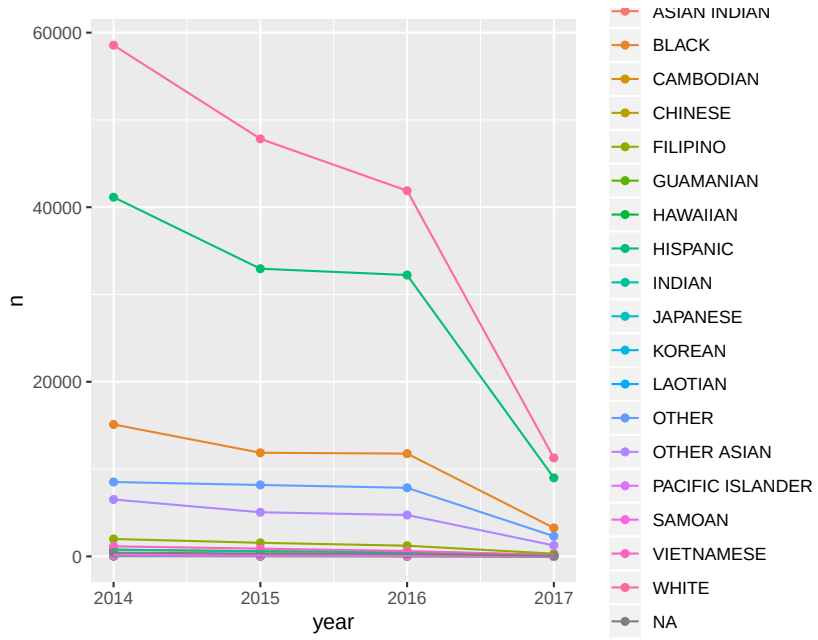
```
CAsd1 <- DBI::dbGetQuery(con1, "SELECT raw_row_number, arrest_made, citation_issued, warning_issu
CAsd2 <- DBI::dbGetQuery(con2, "SELECT raw_row_number, date, time, service_area, subject_age, sub
```

4.2.1.1 [illegible]

```
goodSD %>%
count(year = year(date), raw_subject_race_description) %>%
ggplot(aes(x = year, y = n, color = raw_subject_race_description)) +
geom_point() +
geom_line()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```




```
# in SD, subject race was very detailed and consistant unlike other datasets
# thus, recoded raw_subject_race_description for 'ASIAN/PACIFIC ISLANDER'
goodSD$raw_subject_race_description <-recode(goodSD$raw_subject_race_description, "VIE
                                    "SAMOAN"="ASIAN/PACIFIC ISLANDER",
                                    "LAOTIAN"="ASIAN/PACIFIC ISLANDER",
                                    "KOREAN"="ASIAN/PACIFIC ISLANDER",
                                    "JAPANESE"="ASIAN/PACIFIC ISLANDER",
                                    "INDIAN"="ASIAN/PACIFIC ISLANDER",
                                    "HAWAIIAN"="ASIAN/PACIFIC ISLANDER",
                                    "GUAMANIAN"="ASIAN/PACIFIC ISLANDER",
                                   "FILIPINO"="ASIAN/PACIFIC ISLANDER",
                                    "CHINESE"="ASIAN/PACIFIC ISLANDER",
                                    "CAMBODIAN"="ASIAN/PACIFIC ISLANDER",
                                   "ASIAN INDIAN"="ASIAN/PACIFIC ISLANDER",
                                   "OTHER ASIAN"="ASIAN/PACIFIC ISLANDER",
                                   "PACIFIC ISLANDER"="ASIAN/PACIFIC ISLANDER")
```

4.2.1.1.1 Recoding Race column

4.2.1.2 Plotting Search Conducted by Race

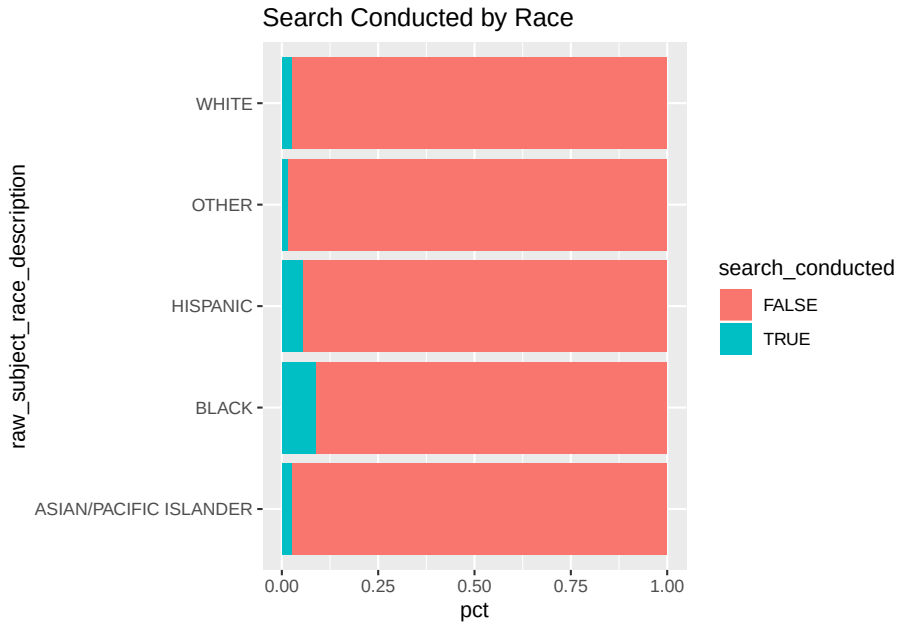
It is helpful to look at the distribution of stop outcomes because it could suggest underlying intent from the officer or commonalities in policing different races (i.e. prevalence of profiled searches, warrant for arrest). The following two visu-

alizations display the distribution of searches and arrests made on traffic stops in San Diego.

```
# create a new dataset
SearchesByRaceCOL <- goodSD %>%

  # select race and search_conducted columns
  group_by(raw_subject_race_description, search_conducted) %>%

  # calculate count for search_conducted by race
  summarize(Total =n()) %>%
  group_by(raw_subject_race_description) %>%
  mutate(pct=Total/sum(Total))
  data.table(SearchesByRaceCOL)
```

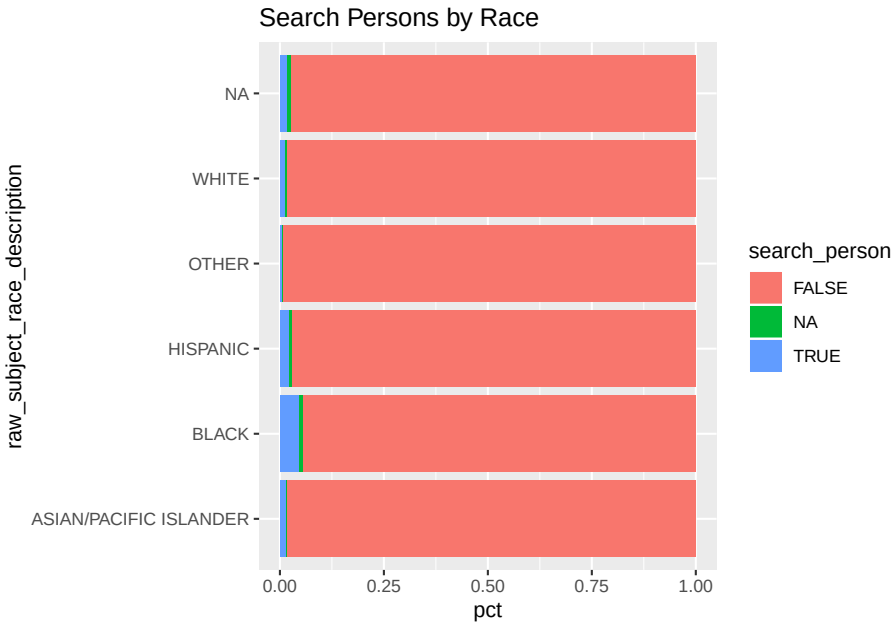
```
##     raw_subject_race_description search_conducted  Total        pct
##  1:       ASIAN/PACIFIC ISLANDER            FALSE  31147 0.97203757
##  2:       ASIAN/PACIFIC ISLANDER             TRUE    896 0.02796243
##  3:                      BLACK            FALSE  38217 0.90949548
##  4:                      BLACK             TRUE   3803 0.09050452
##  5:                   HISPANIC            FALSE 108932 0.94456536
##  6:                   HISPANIC             TRUE   6393 0.05543464
##  7:                      OTHER            FALSE  26420 0.98350892
##  8:                      OTHER             TRUE    443 0.01649108
##  9:                      WHITE            FALSE 155130 0.97223006
## 10:                      WHITE             TRUE   4431 0.02776994
## 11:                       <NA>            FALSE   6893 0.95537076
## 12:                       <NA>             TRUE    322 0.04462924
```

```
# omit NAs for search_conducted
SearchesByRaceCOL <- na.omit(SearchesByRaceCOL)
# visualize proportion of searches to non-search stops
ggplot(SearchesByRaceCOL, aes(x=raw_subject_race_description, y=pct, fill=search_conducted))+ geo
```



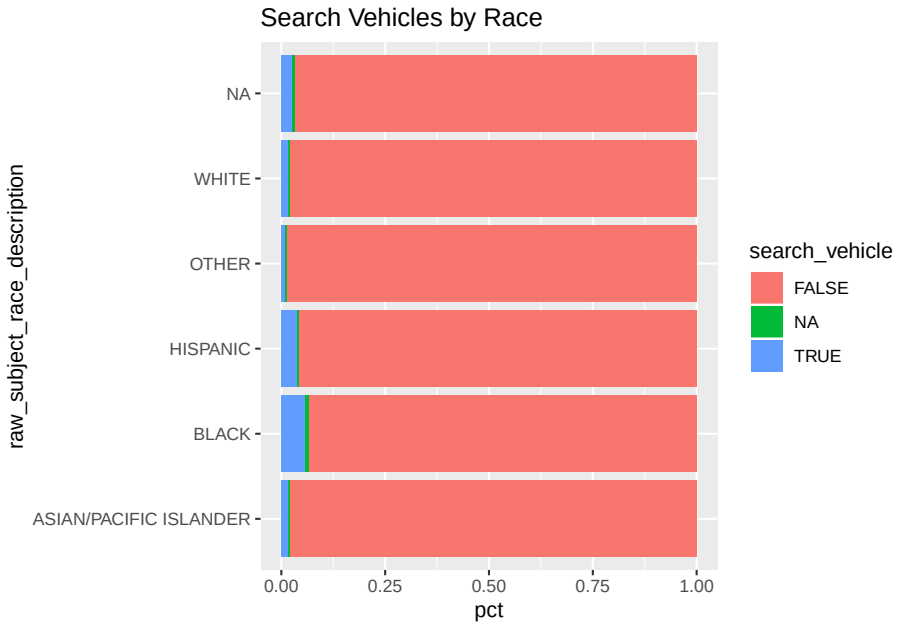
4.2.1.3 Plotting Search Person by Race

```
##     raw_subject_race_description search_person  Total         pct
##  1:       ASIAN/PACIFIC ISLANDER         FALSE  31470 0.982117779
##  2:       ASIAN/PACIFIC ISLANDER            NA    116 0.003620135
##  3:       ASIAN/PACIFIC ISLANDER          TRUE    457 0.014262085
##  4:                       BLACK         FALSE  39630 0.943122323
##  5:                       BLACK            NA    417 0.009923846
##  6:                       BLACK          TRUE   1973 0.046953832
##  7:                    HISPANIC         FALSE 111919 0.970466074
##  8:                    HISPANIC            NA    762 0.006607414
##  9:                    HISPANIC          TRUE   2644 0.022926512
## 10:                       OTHER         FALSE  26630 0.991326360
## 11:                       OTHER            NA     76 0.002829170
## 12:                       OTHER          TRUE    157 0.005844470
## 13:                       WHITE         FALSE 156744 0.982345310
## 14:                       WHITE            NA    755 0.004731733
## 15:                       WHITE          TRUE   2062 0.012922957
## 16:                        <NA>         FALSE   7020 0.972972973
## 17:                        <NA>            NA     64 0.008870409
## 18:                        <NA>          TRUE    131 0.018156618
```



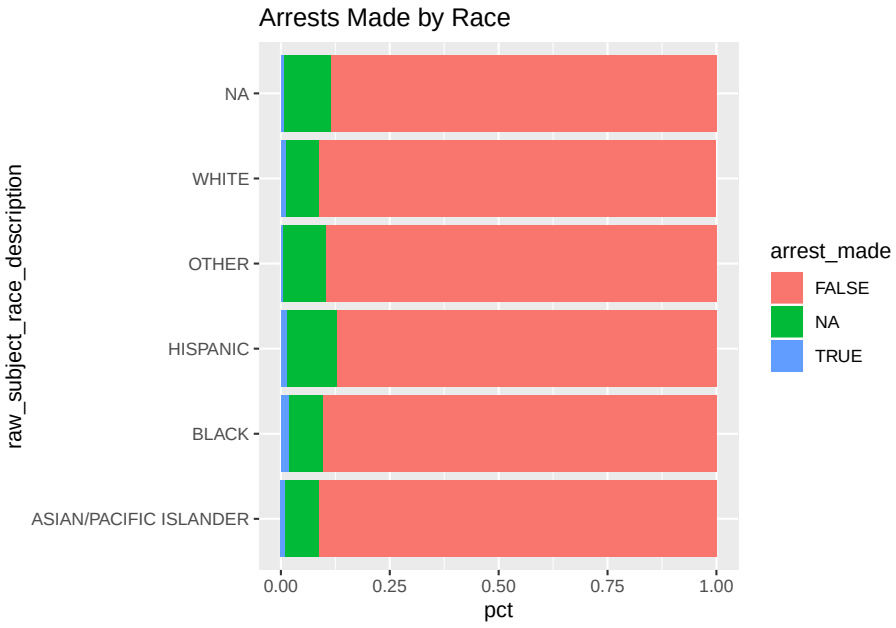
4.2.1.4 Plotting Search Vehicle by Race

```
##     raw_subject_race_description search_vehicle  Total         pct
##  1:       ASIAN/PACIFIC ISLANDER          FALSE  31400 0.979933215
##  2:       ASIAN/PACIFIC ISLANDER             NA    116 0.003620135
##  3:       ASIAN/PACIFIC ISLANDER           TRUE    527 0.016446650
##  4:                        BLACK          FALSE  39200 0.932889100
##  5:                        BLACK             NA    417 0.009923846
##  6:                        BLACK           TRUE   2403 0.057187054
##  7:                     HISPANIC          FALSE 110313 0.956540212
##  8:                     HISPANIC             NA    762 0.006607414
##  9:                     HISPANIC           TRUE   4250 0.036852374
## 10:                        OTHER          FALSE  26513 0.986970927
## 11:                        OTHER             NA     76 0.002829170
## 12:                        OTHER           TRUE    274 0.010199903
## 13:                        WHITE          FALSE 156344 0.979838432
## 14:                        WHITE             NA    755 0.004731733
## 15:                        WHITE           TRUE   2462 0.015429836
## 16:                         <NA>          FALSE   6968 0.965765766
## 17:                         <NA>             NA     64 0.008870409
## 18:                         <NA>           TRUE    183 0.025363825
```



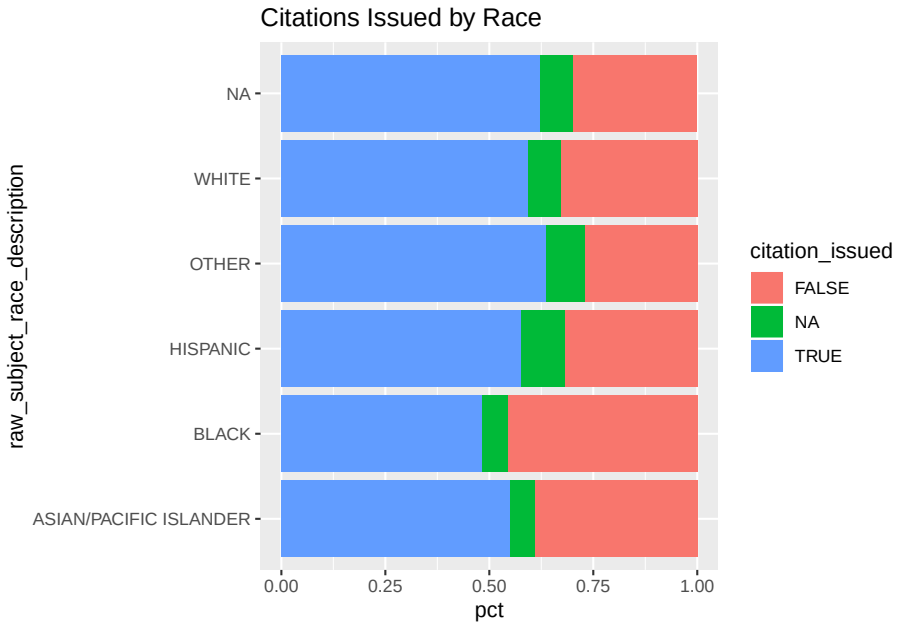
4.2.1.5 Plotting Arrest Made by Race

```
##     raw_subject_race_description arrest_made  Total         pct
##  1:       ASIAN/PACIFIC ISLANDER       FALSE  29262 0.913210374
##  2:       ASIAN/PACIFIC ISLANDER          NA   2476 0.077271167
##  3:       ASIAN/PACIFIC ISLANDER        TRUE    305 0.009518460
##  4:                        BLACK       FALSE  37953 0.903212756
##  5:                        BLACK          NA   3215 0.076511185
##  6:                        BLACK        TRUE    852 0.020276059
##  7:                     HISPANIC       FALSE 100404 0.870617819
##  8:                     HISPANIC          NA  13224 0.114667245
##  9:                     HISPANIC        TRUE   1697 0.014714936
## 10:                        OTHER       FALSE  24066 0.895879090
## 11:                        OTHER          NA   2638 0.098201988
## 12:                        OTHER        TRUE    159 0.005918922
## 13:                        WHITE       FALSE 145399 0.911243976
## 14:                        WHITE          NA  12416 0.077813501
## 15:                        WHITE        TRUE   1746 0.010942524
## 16:                         <NA>       FALSE   6385 0.884961885
## 17:                         <NA>          NA    774 0.107276507
## 18:                         <NA>        TRUE     56 0.007761608
```



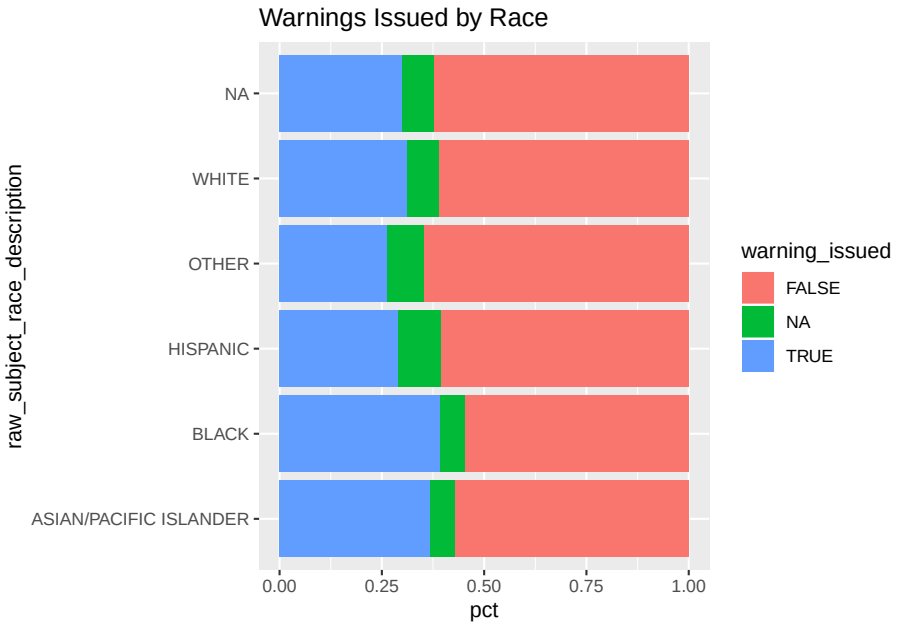
4.2.1.6 Plotting Citations Issued by Race

```
##     raw_subject_race_description citation_issued Total        pct
##  1:       ASIAN/PACIFIC ISLANDER           FALSE 12496 0.38997597
##  2:       ASIAN/PACIFIC ISLANDER              NA  1937 0.06045002
##  3:       ASIAN/PACIFIC ISLANDER            TRUE 17610 0.54957401
##  4:                       BLACK           FALSE 19105 0.45466445
##  5:                       BLACK              NA  2566 0.06106616
##  6:                       BLACK            TRUE 20349 0.48426940
##  7:                    HISPANIC           FALSE 36667 0.31794494
##  8:                    HISPANIC              NA 12041 0.10440928
##  9:                    HISPANIC            TRUE 66617 0.57764578
## 10:                       OTHER           FALSE  7263 0.27037189
## 11:                       OTHER              NA  2475 0.09213416
## 12:                       OTHER            TRUE 17125 0.63749395
## 13:                       WHITE           FALSE 52240 0.32739830
## 14:                       WHITE              NA 12377 0.07756908
## 15:                       WHITE            TRUE 94944 0.59503262
## 16:                        <NA>           FALSE  2148 0.29771310
## 17:                        <NA>              NA   575 0.07969508
## 18:                        <NA>            TRUE  4492 0.62259182
```



4.2.1.7 Plotting Warnings Issued by Race

```
##     raw_subject_race_description warning_issued Total        pct
##  1:       ASIAN/PACIFIC ISLANDER          FALSE 18313 0.57151328
##  2:       ASIAN/PACIFIC ISLANDER             NA  1937 0.06045002
##  3:       ASIAN/PACIFIC ISLANDER           TRUE 11793 0.36803670
##  4:                       BLACK          FALSE 22953 0.54623989
##  5:                       BLACK             NA  2566 0.06106616
##  6:                       BLACK           TRUE 16501 0.39269396
##  7:                    HISPANIC          FALSE 69890 0.60602645
##  8:                    HISPANIC             NA 12041 0.10440928
##  9:                    HISPANIC           TRUE 33394 0.28956427
## 10:                       OTHER          FALSE 17356 0.64609314
## 11:                       OTHER             NA  2475 0.09213416
## 12:                       OTHER           TRUE  7032 0.26177270
## 13:                       WHITE          FALSE 97389 0.61035591
## 14:                       WHITE             NA 12377 0.07756908
## 15:                       WHITE           TRUE 49795 0.31207501
## 16:                        <NA>          FALSE  4485 0.62162162
## 17:                        <NA>             NA   575 0.07969508
## 18:                        <NA>           TRUE  2155 0.29868330
```



4.2.1.8 Distribution of NAs in outcome variables

```
#search_conducted vs. arrest_made, citation_issued, warning_issued
table(goodSD$search_conducted,goodSD$arrest_made)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 331180  34642    917
##   TRUE   12289    101   3898
```

```
table(goodSD$search_conducted,goodSD$citation_issued)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 122819  31235 212685
##   TRUE    7100    736   8452
```

```
table(goodSD$search_conducted,goodSD$warning_issued)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 217781  31235 117723
##   TRUE   12605    736   2947
```

```
#search_person vs. arrest_made, citation_issued, warning_issued
table(goodSD$search_person,goodSD$arrest_made)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 337245  34666   1502
##   NA      1483     51    656
##   TRUE    4741     26   2657
```

```
table(goodSD$search_person,goodSD$citation_issued)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 124552  31364 217497
##   NA       756    252   1182
##   TRUE    4611    355   2458
```

```
table(goodSD$search_person,goodSD$warning_issued)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 223434  31364 118615
##   NA      1526    252    412
##   TRUE    5426    355   1643
```

```
#search_vehicle vs. arrest_made, citation_issued, warning_issued
table(goodSD$search_vehicle,goodSD$arrest_made)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 333675  34662   2401
##   NA      1483     51    656
##   TRUE    8311     30   1758
```

```
table(goodSD$search_vehicle,goodSD$citation_issued)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 125189  31467 214082
##   NA       756    252   1182
##   TRUE    3974    252   5873
```

```
table(goodSD$search_vehicle,goodSD$warning_issued)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 220741  31467 118530
##   NA      1526    252    412
##   TRUE    8119    252   1728
```

```
#What does a conducted search result in?
table(goodSD$search_conducted,goodSD$search_vehicle)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 366739      0      0
##   TRUE    3999   2190  10099
```

```
table(goodSD$search_conducted,goodSD$search_person)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 366739      0      0
##   TRUE    6674   2190   7424
```

```
table(goodSD$search_person,goodSD$search_vehicle)
```

```
##
##           FALSE     NA   TRUE
##   FALSE 366739      0   6674
##   NA         0   2190      0
##   TRUE    3999      0   3425
```

```
#NAs in search variables appear to be indicators that search was not designated as a person or ve
#three variable plots
#looking at traffic stops in relation to income level in the area
#dimension reduction: finding what we wanted to predict/model
#chugging away at the Shiny app
```

4.2.2 Weather Overlay in Raleigh

This section of exploratory data analysis sees if precipitation shows correlation with the amount of traffic stops per day. The expected correlation is that if it rains more there will be an increase in traffic stops because wet roads could lead to more reckless driving.

The Rnoaa package requires an API key, which can be acquired from the Rnoaa documentation

4.2.2.1 Visualizing Weather in Raleigh, North Carolina

In this example, we will overlay precipitation over traffic stops in Raleigh February.

```
raleigh <- DBI::dbGetQuery(con, "SELECT * FROM NCraleigh")
```

In order to get weather data we must call the ncdc function. The key argument to keep in mind is the stationid argument. In order to find a city's station ID you must go to this NOAA's website https://www.ncdc.noaa.gov/cdo-web/search. Moreover, the type of data such as precipation, temperature, and snow etc is contingent on the station ID. For this example, we will look at precipitation only, but there is room to explore more variables to see in any affect traffic.

```
out <- ncdc(datasetid='GHCND', stationid='GHCND:USC00317079', datatypeid='PRCP', starto
```

Now that we have the two datasets we must clean them because we will be joining them.

Here we use the lubridate package to make year, month, and day their own columns in the weather dataframe.

```
weather_df <- data.frame(out$data)

# fix weather_df dates
weather_df <- weather_df %>% mutate(clean_date = ymd_hms(date))

# make individual columns with year, month, & day
weather_df <- weather_df %>%
  mutate(year = year(clean_date),
         month = month(clean_date),
         day = day(clean_date))
```

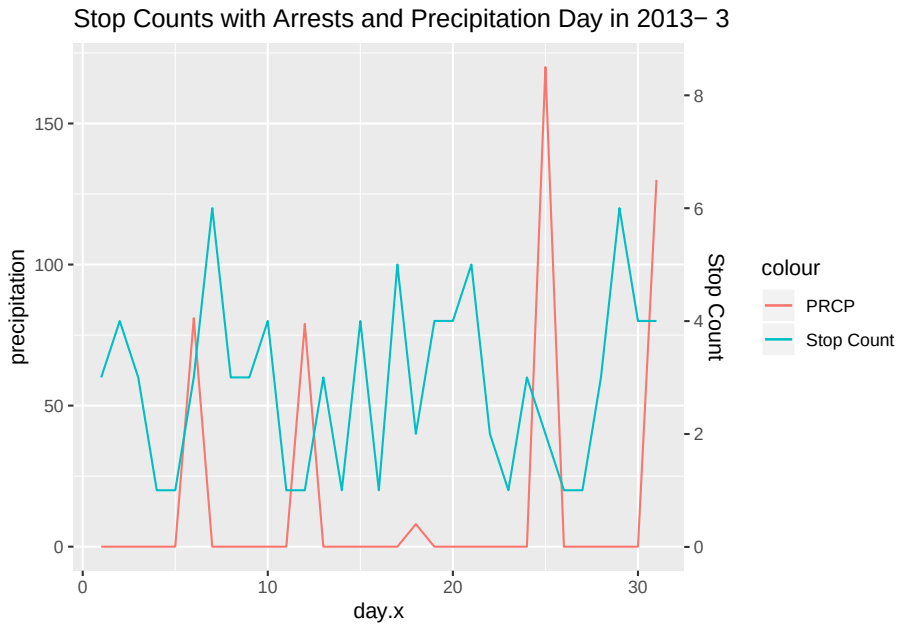
Similarly, the raleigh dataset must also have year, month, and day in separate columns.

```
# use libridate to add clean date column for lubridate to interpret
raleigh <- raleigh %>% mutate(clean_date = ymd(date, tz = 'UTC'))

# use lubridate to add year, month, day columns
raleigh <- raleigh %>%
  mutate(year = year(clean_date),
         month = month(clean_date),
         day = day(clean_date))
```

Now that we have the two dataframes cleaned up, we can follow a sequence of piping commands to join and plot them.

```
# define month variable so we can easily plot different months
month_var = 3
title = paste("Stop Counts with Arrests and Precipitation Day in 2013-", paste(month_va

# the pipe sequence below will examine a month in a year such that we count all the tr
# stops that happened that day. Then, we plot the stop counts for a specific day and m

raleigh %>%
  # filter out raleigh entries in year 2013 and month variable
  filter(year == 2013, month == month_var, arrest_made=="TRUE") %>%
  # left_join by 'clean_date'
  left_join(weather_df, by = c('clean_date'))%>%
  # Group the data by day as this will be our shared x-axis
  group_by(day.x) %>%
```

```
  # count the number of stops per day and the precipitation value of that day
  summarize(count = n(), precipitation=max(value)) %>%
  ggplot() +
  geom_line(aes(x=day.x, y= precipitation, color="PRCP")) +
  geom_line(aes(x=day.x, y= count/.05, color = "Stop Count")) +
  scale_y_continuous(sec.axis = sec_axis(~.*.05, name = "Stop Count")) +
  ggtitle(title)
```



Based on the plot, it seems like precipitation in day does not impract the amount of stops. For example, the precipitation day 15 and 25 is relatively but stop count still fluctuates. Future analysis could look at the hourly level of stop counts and precipation or a year long look of precipiation and stop counts per day.

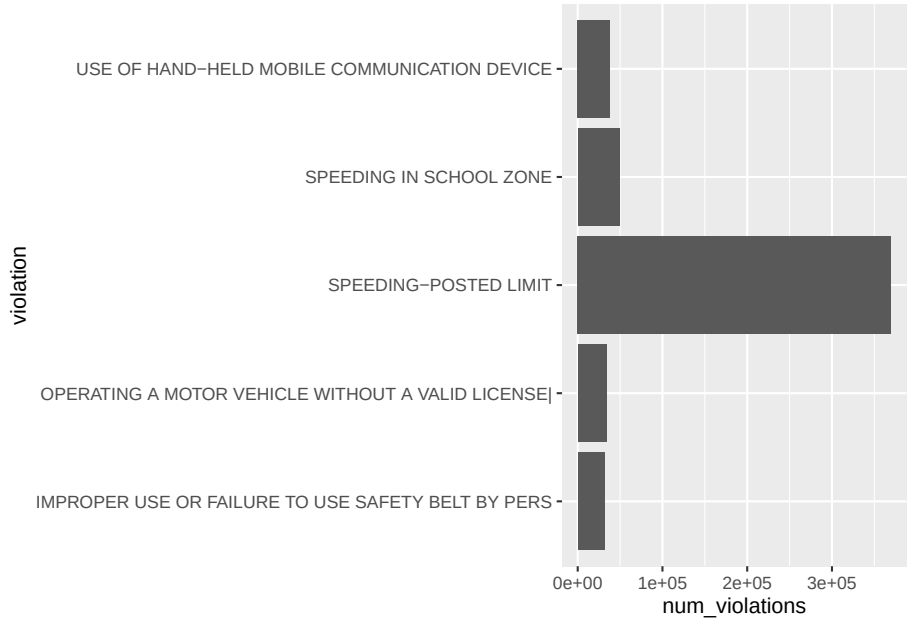
4.2.3 Common Violations and Map Visualisations in San Antonio, TX

4.2.3.1 Common Violations

Here I graphed the 5 violations that occured most frequently. It is important to notice that this would only include those violations that are worded exactly the same. More work would need to be done in order to find which violations mean the same thing but some had typos or minor differences.

```
VioData <- DBI::dbGetQuery(con,
  "SELECT violation, COUNT(violation) AS 'num_violations' FROM TXsanantonio
```

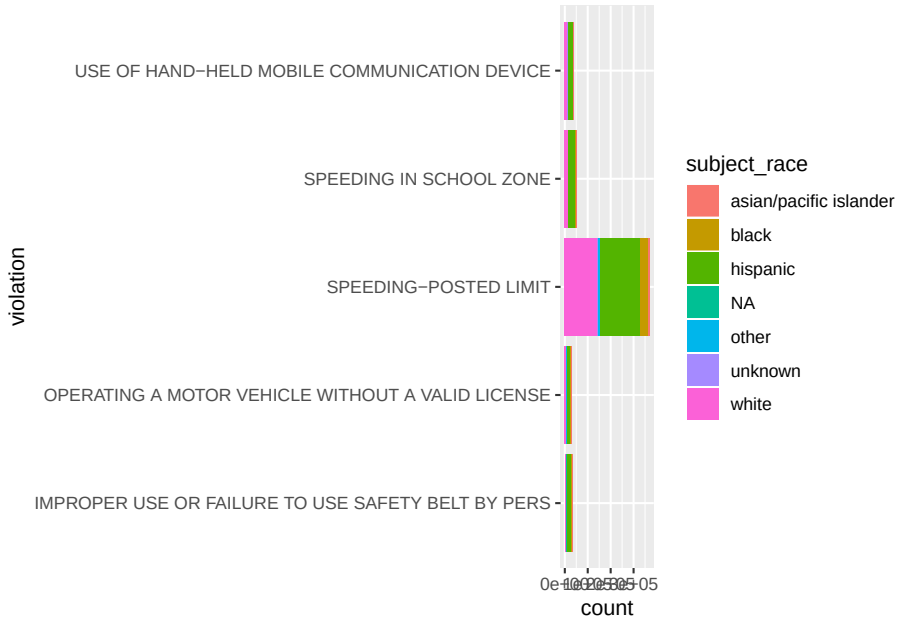
```
  GROUP BY violation
  ORDER BY `num_violations` DESC
  LIMIT 5")

ggplot(data = VioData) + geom_bar(mapping = aes(x = violation, y = num_violations),
                                  stat = 'identity') + coord_flip()
```



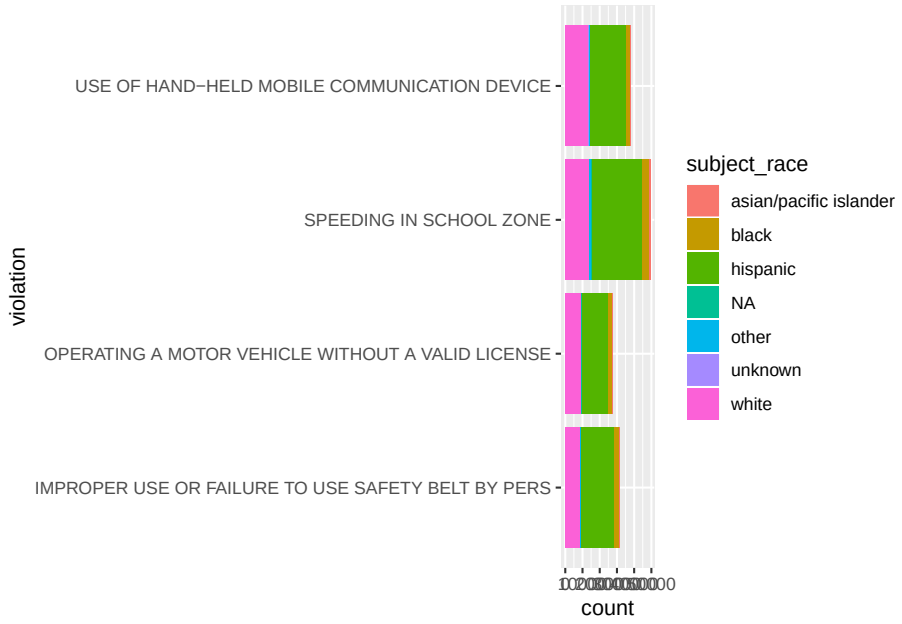
I then took these top 5 violations and split them up by sex and race. I also included additional graphs of the 2-5 violations for clarity because of the overwhelming violation of speeding.

```
top5_vio <- DBI::dbGetQuery(con,
  "SELECT subject_sex, subject_race, violation FROM TXsanantonio
  WHERE violation = 'USE OF HAND-HELD MOBILE COMMUNICATION DEVICE'
  OR violation = 'SPEEDING-POSTED LIMIT'
  OR violation = 'SPEEDING IN SCHOOL ZONE'
  OR violation = 'OPERATING A MOTOR VEHICLE WITHOUT A VALID LICENSE'
  OR violation = 'IMPROPER USE OR FAILURE TO USE SAFETY BELT BY PERS'")

top4_vio <- DBI::dbGetQuery(con,
  "SELECT subject_sex, subject_race, violation FROM TXsanantonio
  WHERE violation = 'USE OF HAND-HELD MOBILE COMMUNICATION DEVICE'
  OR violation = 'SPEEDING IN SCHOOL ZONE'
  OR violation = 'OPERATING A MOTOR VEHICLE WITHOUT A VALID LICENSE'
  OR violation = 'IMPROPER USE OR FAILURE TO USE SAFETY BELT BY PERS'")
```

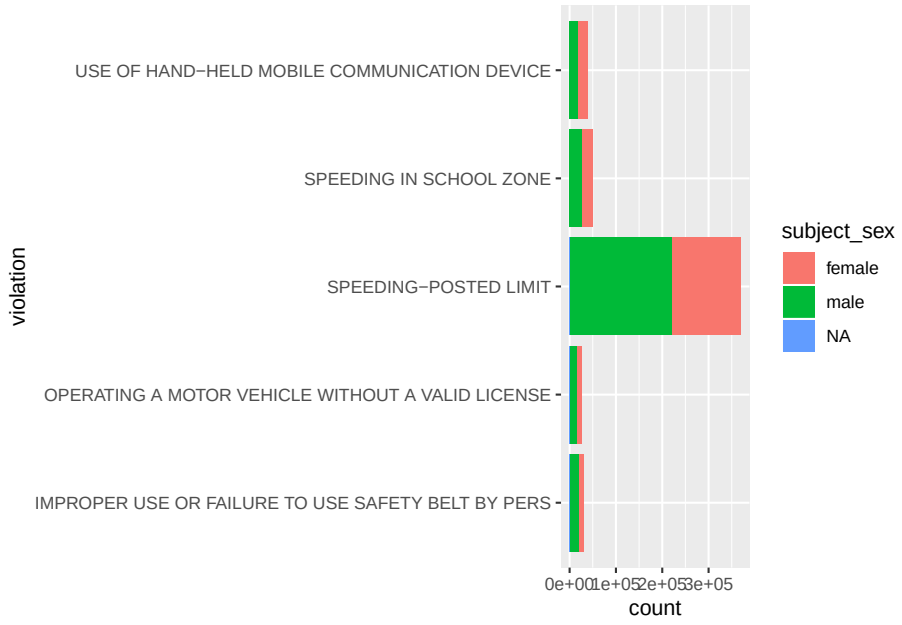
```
ggplot(data = top5_vio) + geom_bar(mapping = aes(x = violation, fill = subject_race)) +
  coord_flip()
```




```
#created this to see the 2-5 violations more clearly
ggplot(data = top4_vio) + geom_bar(mapping = aes(x = violation, fill = subject_race)) +
  coord_flip()
```

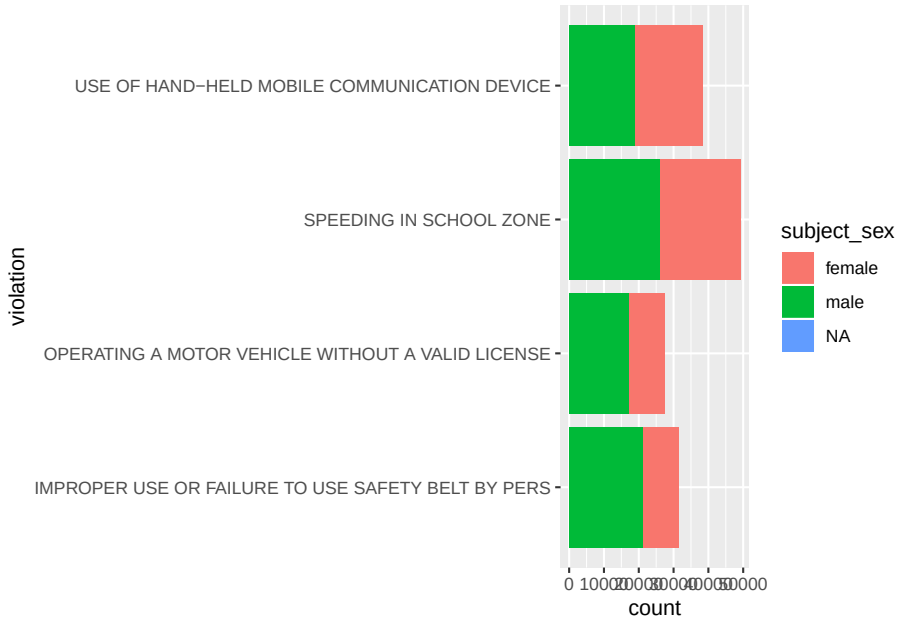



```
ggplot(data = top5_vio) + geom_bar(mapping = aes(x = violation, fill = subject_sex)) +
  coord_flip()
```




```
#created this to see the 2-5 violations more clearly
ggplot(data = top4_vio) + geom_bar(mapping = aes(x = violation, fill = subject_sex)) +
```

```
  coord_flip()
```

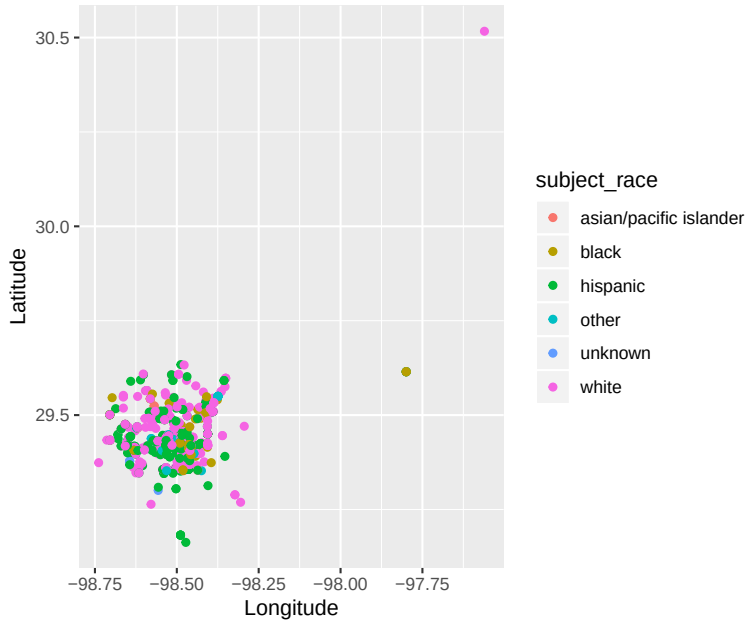


4.2.3.2 Map Overlay

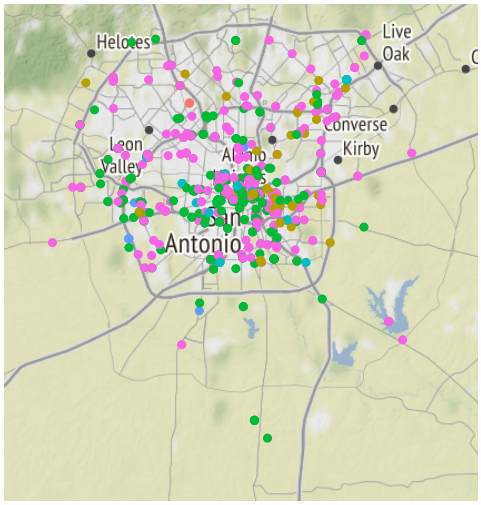
During the EDA phase, geography became of interest using the latitude and longitude variables from the dataset. This image displays the plot of the San Antonio dataset using ggplot with the dots grouped by the race of the victim of the stop. This raised a few questions. For example, we questioned how highway stops may differ from non highway stops given a potential difference in demographics for those who drive on the highways. We also recognized the possibility of controlling for demographics of certain neighborhoods and comparing stop rates to those demographics. Both questions raise interesting questions for further investigation.

```
Coordinates <- DBI::dbGetQuery(con,
  "SELECT * FROM TXsanantonio LIMIT 1000")

coor <- DBI::dbGetQuery(con,
  "SELECT lng, lat, subject_race FROM TXsanantonio LIMIT 1000")

coor$lng <- with(coor, as.numeric(lng))
coor$lat <- with(coor, as.numeric(lat))

race_plot <- ggplot(Coordinates, aes(x = as.numeric(lng),
                                     y = as.numeric(lat), xaxt = 'n', yaxt = 'n')) +
```

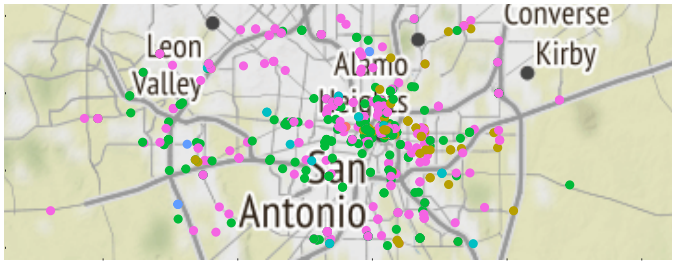
```
  geom_point(aes(color = subject_race)) +
  xlab("Longitude") + ylab("Latitude") + coord_quickmap()
race_plot
```




```
coord_plot <- qmplot(lng, lat, data = coor, maptype = "toner-background",
                     color = subject_race, xlim = c(-99, -98.25), ylim = c(29, 29.6))
coord_plot
```




```
qmplot(lng, lat, data = coor, maptype = "toner-background",
       color = subject_race, xlim = c(-98.75, -98.3), ylim = c(29.35, 29.5))
```

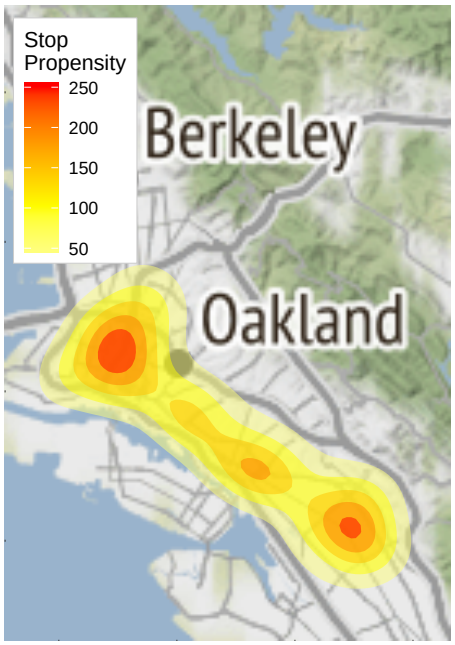


This is the same coordinate plot for the Oakland Dataset.

This is a density plot of stops for the oaklans dataset. This was in an attempt

to see if there were any particular areas of high concentration.

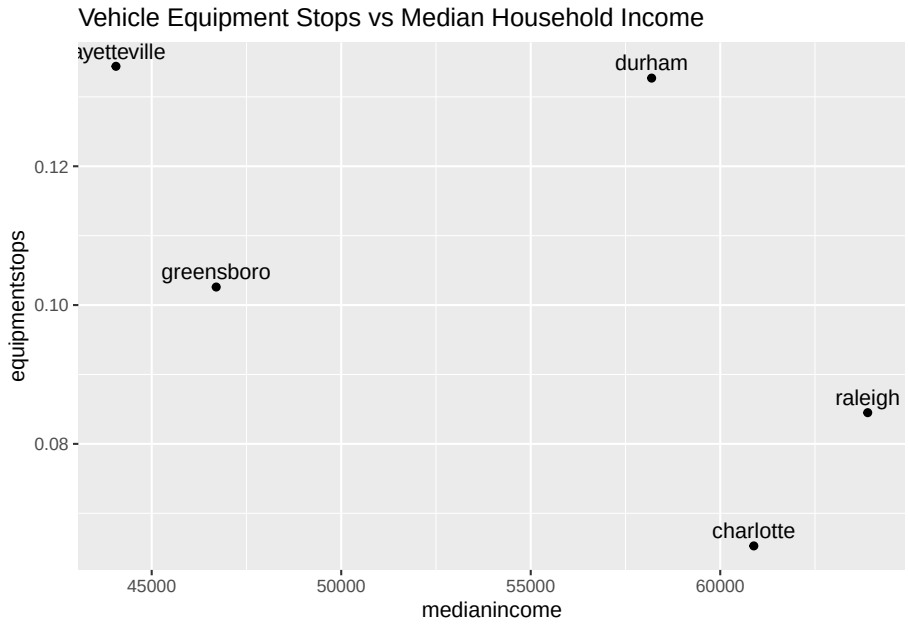
```
qmplot(lng, lat, data = coor2, geom = "blank",
  zoom = 10, maptype = "toner-background", darken = 0, legend = "topleft") +
  stat_density_2d(aes(fill = ..level..), geom = "polygon", alpha = .5, color = NA) +
  scale_fill_gradient2("Stop\nPropensity", low = "white", mid = "yellow", high = "red"
                       midpoint = 100)
```



4.2.4 Socioeconomic Factors in Durham

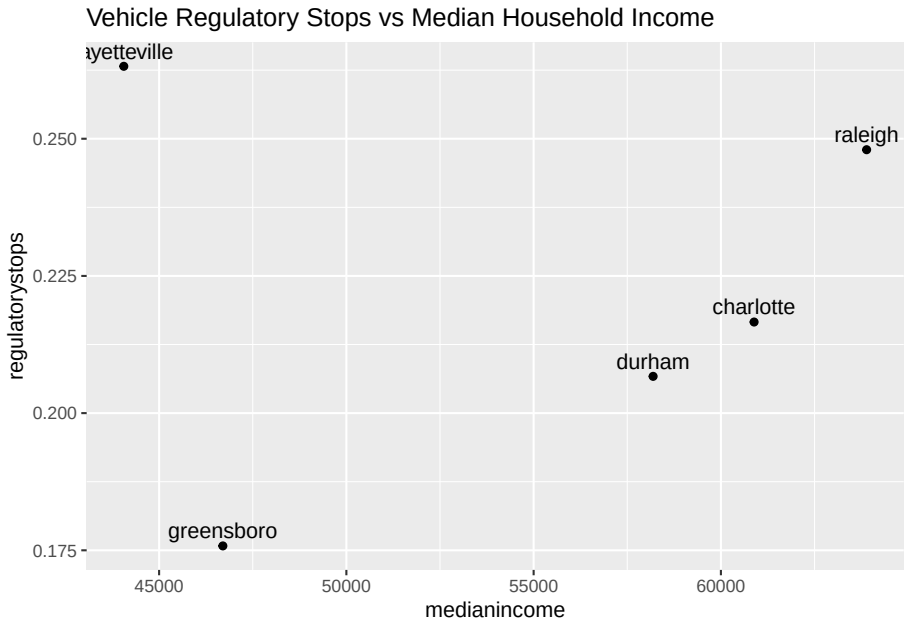
The reason behind a vehicle equipment stop seems to be an economical one, as drivers who have funds to fix their vehicles in the case of a broken tail light for example would be less likely to get stopped for an equipment violation. The plot is used to investigate this hypothesis, where, as median income increases, we look for the percentage of equipment violations to drop.

```
#manual data entry
incomeNC <- data.frame("city" = c("durham", "raleigh", "charlotte", "fayetteville", "gr

#plot for percent of equipment stops
equipment <- incomeNC %>%
  ggplot() +
  geom_point(aes(x = medianincome, y=equipmentstops)) +
  geom_text(aes(x = medianincome, y=equipmentstops, label=city),hjust=0.5, vjust=-0.5)
  labs(title="Vehicle Equipment Stops vs Median Household Income")
equipment
```



This is a scatter plot of percentage of vehicle equipment stops against median household income (obtained from census.gov). The scatter plot reflects this relationship, however since there are so few observations, maybe more data is needed for a more comprehensive conclusion.

```
#plot for percent of regulatory stops
regulatory <- incomeNC %>%
  ggplot() +
  geom_point(aes(x = medianincome, y=regulatorystops)) +
  geom_text(aes(x = medianincome, y=regulatorystops, label=city),hjust=0.5, vjust=-0.5) +
  labs(title="Vehicle Regulatory Stops vs Median Household Income")
regulatory
```



We also look at vehicle regulatory stops against median household income, which also seems to be related to economically implications. Here we see a more clear trend excluding Fayetteville, that there is an increase in percentage stops with the increase of median income.

4.3 Modeling the Probability of Seach Given Stop

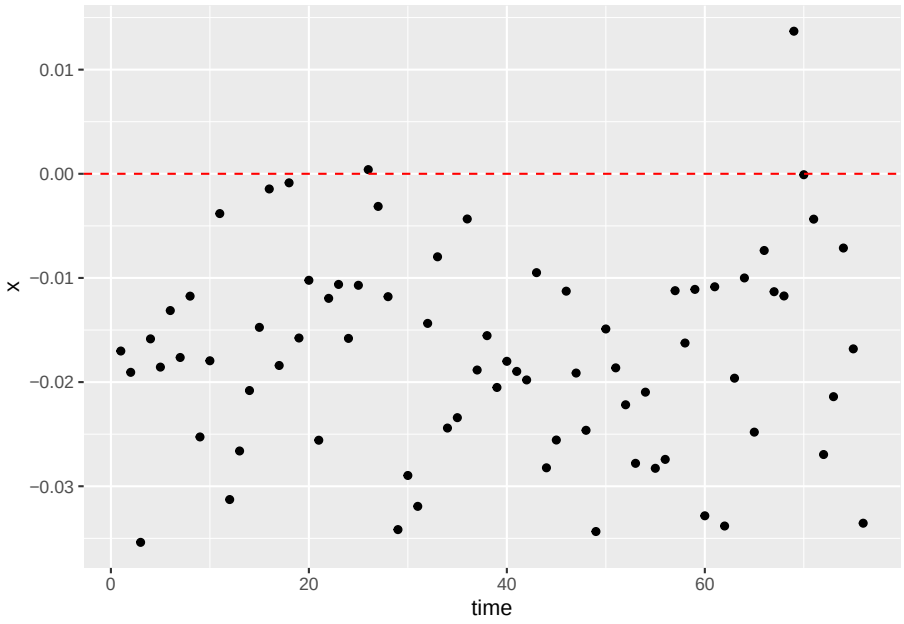
During night times when the sun is set, researchers speculate that stops are made by police officers without the acknowledgment of the racial identity of the driver. This should be the exact opposite for police officers during the day times. With the hope of identifying possible underlying racial discrimination with the available stop data, it has therefore been determined that for a specific region, the differences between the percentage of minority drivers within day times and night times could be used to detect such discrimination. The central idea is that, by observing the distribution of the difference values through the progression of time, we can gain a better understanding of the change of police behaviors with the transformation from day times to night times.

The calculation of the difference for this research project is set to be the daytime percentage of minority drivers (black) being stopped deducted by that of the night time percentage. A positive difference value indicates that if the police officer is able to identify the identity of the driver (i.e., under the day times), she or he has a higher percentage chance of stopping the minority driver. Whereas during the night times with a lack of sufficient light source, we assumed that the

police officer was unable to identify the driver's race. Therefore, if there exists racial discrimination against minority drivers, we should expect to observe that a significant proportion of percentage differences should be positive. However, if the visualization shows an opposite distribution, then we have to conclude that the division of day time and night time has failed to capture the existence of racial discrimination.

```
SAN <- SAN %>%
  dplyr::mutate(date2 = lubridate::as_date(lubridate::ymd(as.character(date))))

SAN <- SAN %>% mutate(date_time = as.POSIXct(paste(date, time), tz = "America/Chicago", format =

oursunriseset <- function(latitude, longitude, date, direction = c("sunrise", "sunset")) {
  date.lat.long <- data.frame(date = date, lat = latitude, lon = longitude)
  if(direction == "sunrise"){
    x <- getSunlightTimes(data = date.lat.long, keep=direction, tz="America/Chicago")$sunrise }el
    x <- getSunlightTimes(data = date.lat.long, keep=direction, tz="America/Chicago")$sunset  }
  return(x)
}

sunrise <- oursunriseset(29.4241, -98.4936, SAN$date2, direction = "sunrise")
sunset <- oursunriseset(29.4241, -98.4936, SAN$date2, direction = "sunset")

SAN <- cbind(SAN, sunrise, sunset)

SAN <- SAN %>%
mutate(light = ifelse(date_time < sunrise, "night", ifelse(date_time > sunset, "night", "day")))

racial_perc <- function(mon, yr, race){
  x1 <- filter(SAN, light == "day", month == mon, year == yr)
  x1 <- prop.table(table(x1$subject_race))
  x1 <- as.data.frame(x1)
  x1 <- filter(x1, Var1 == race)$Freq

  x2 <- filter(SAN,light == "night", month == mon, year == yr)
  x2 <- prop.table(table(x2$subject_race))
  x2 <- as.data.frame(x2)
  x2 <- filter(x2, Var1 == race)$Freq

  return(x1-x2)
}

x <- c()
y <- c()
z <- c()
for (i in seq(2012,2017)){
```

```
  for (j in seq(1,12)){
    diff <- racial_perc(j,i,"black")
    x <- c(x, diff)
    y <- c(y,i)
    z <- c(z,j)
  }
}

diff_data_SAN <- as.data.frame(cbind(x,y,z))
diff_data_SAN <- diff_data_SAN %>% mutate(time = 1:nrow(diff_data_SAN))

x <- c()
y <- c()
z <- c()
for (i in seq(1,4)){
  diff <- racial_perc(i,'2018',"black")
  x <- c(x, diff)
  z <- c(1,2,3,4)
  y <- c(2018, 2018, 2018, 2018)
}

n <- nrow(diff_data_SAN)
diff_data_SAN2 <- as.data.frame(cbind(x,y,z))
diff_data_SAN2 <- diff_data_SAN2 %>% mutate(time = (n+1):(n+4))
diff_data_SAN <- rbind(diff_data_SAN,diff_data_SAN2)


ggplot(data = diff_data_SAN) +
  geom_point(mapping = aes(x = time, y = x)) +
  geom_hline(yintercept=0, linetype="dashed", color = "red")
```



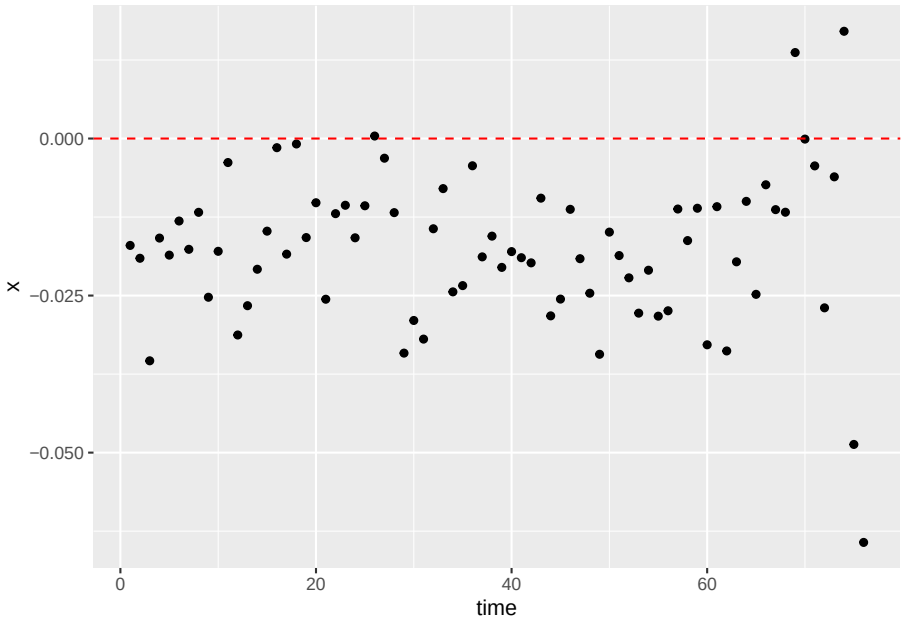
The policing data used for the visualization is from Sanantonio, Texas, and the date is from January 2012 to April 2018. This data recorded a total number of 1,040,428 observations. Key variables are the date of the stop made and the race of the driver, and the corresponding sunrise and sunset time were calculated with each of the given dates. If the stop took place after the sunrise time and before the sunset time, it was set to be a stop made during the day time. Stops that took place outside of this range were set to be night time stops. Then for each day, the percentage of black drivers stopped during day and night was separately calculated and their difference was documented. Finally, the visualization of the key variables was a scatter plot with the horizontal axis as the progression of time and the vertical axis as the day time percentage deducted by night time percentage. For us to claim that the plot has successfully captured the possible existence of racial discrimination, we should expect to observe the majority of the points to be above the horizontal line at zero. However, the scatter plot demonstrates a trend contrary to our expectations. Most of the points fall under the horizontal line, which indicates that for the region of Sanantonio, black drivers on average were stopped at a lower percentage during day time than night time. Using this method and the given data, we have failed the attempt to prove that there has been racial discrimination against black drivers within the data.

One possible downfall of the aforementioned method is that it failed to acknowledge the possible difference in the distribution of driver populations between day time and night time. More specifically speaking, it is likely that for the region of Sanantonio, drivers who are on the road during night time and day time

are two completely different population groups, and therefore, comparing the percentages of drivers stopped between these two possibly distinct population might be less meaningful. One possible improvement in the hope of alleviating population distinctions is the "veil of darkness" test, which limits to the stops made only one hour before and after the sunset time. This approach helps to reduce the possible distinction in the racial distribution of the night time and day time drivers. A similar set of procedures to the previous method was conducted, and a second scatter plot was produced. For the second time, the majority of the points are still below the horizontal line, which indicates that even if we have attempted to reduce the possible transformation of racial distribution, black drivers were mostly stopped at a higher percentage during night time relative to that of the day time. Therefore, the second plot has also failed to offer evidence for the existence of racial discrimination in the data for Sanantonio.

```
SAN2 <- SAN %>%
  filter(sunset-3600 < date_time & sunset + 3600 > date_time)

racial_perc2 <- function(mon, yr, race){
  x1 <- filter(SAN2, light == "day", month == mon, year == yr)
  x1 <- prop.table(table(x1$subject_race))
  x1 <- as.data.frame(x1)
  x1 <- filter(x1, Var1 == race)$Freq

  x2 <- filter(SAN2,light == "night", month == mon, year == yr)
  x2 <- prop.table(table(x2$subject_race))
  x2 <- as.data.frame(x2)
  x2 <- filter(x2, Var1 == race)$Freq

  return(x1-x2)
}

x <- c()
y <- c()
z <- c()
for (i in seq(2012,2017)){
  for (j in seq(1,12)){
    diff <- racial_perc(j,i,"black")
    x <- c(x, diff)
    y <- c(y,i)
    z <- c(z,j)
  }
}

diff_data_SAN <- as.data.frame(cbind(x,y,z))
diff_data_SAN <- diff_data_SAN %>% mutate(time = 1:nrow(diff_data_SAN))
```

```
x <- c()
y <- c()
z <- c()
for (i in seq(1,4)){
  diff <- racial_perc2(i,'2018',"black")
  x <- c(x, diff)
  z <- c(1,2,3,4)
  y <- c(2018, 2018, 2018, 2018)
}

n <- nrow(diff_data_SAN)
diff_data_SAN2 <- as.data.frame(cbind(x,y,z))
diff_data_SAN2 <- diff_data_SAN2 %>% mutate(time = (n+1):(n+4))
diff_data_SAN <- rbind(diff_data_SAN,diff_data_SAN2)

ggplot(data = diff_data_SAN) +
  geom_point(mapping = aes(x = time, y = x)) +
  geom_hline(yintercept=0, linetype="dashed", color = "red")
```



Chapter 5

Nationwide Logistic Regression

Here, we apply our logistic regression test to multiple cities or statewide datasets. The logistic regression in Charlotte, NC reveals distinct probabilities for black and white drivers as a function of age. The purposes behind applying the logistic regression to multiple cities is to observe differences in search conducted probabilities based on age and race and see if there is any interaction with age and race variables.

The logistic_regression function takes in a dataset and name inputs. This function will run a logistic regression on the given datasets. Then, it takes the coefficients and added to a row of a data frame called, coefficient_matrix. Each row coefficient_matrix represents a different cities with their respective coefficients from the logistic regression.

```
coefficient_matrix <- data.frame("intercept" = numeric(), "subject_age" = numeric(), "subject_rac

logistic_regression <- function(city_dataset, name){

  # run logistic regression
  fitlog <- glm(formula = search_binary ~ subject_age*subject_race, data = city_dataset, family =

  # record logistic regression coefficients
  coefficient_row_vector = t(fitlog$coefficients)

  # row bind each coefficient and dataset tibble with the coefficent_matrix
  coefficient_matrix <<- rbind(coefficient_matrix, cbind.data.frame(as.data.frame(coefficient_row

}
```

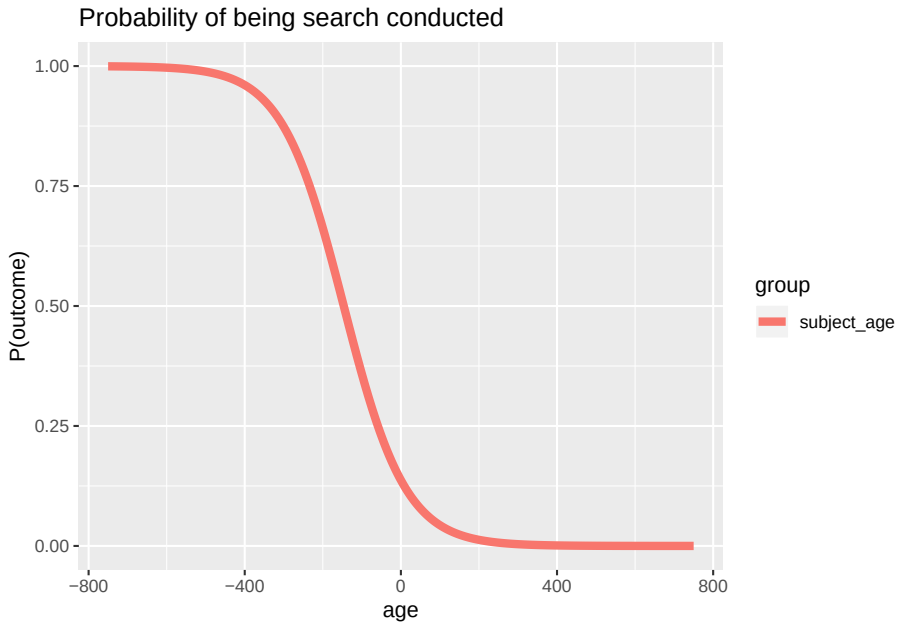
```
mapply(logistic_regression, datasets, datasets_of_interest)

coefficient_matrix
colnames(coefficient_matrix) <- c("intercept", "subject_age", "subject_raceW", "subject
```

Running the code above will output a dataframe in your local environment called coefficient matrix. We decided to write the dataframe as a csv file ("coeff_matrix_raw.csv") so that we could resuse the coefficients.

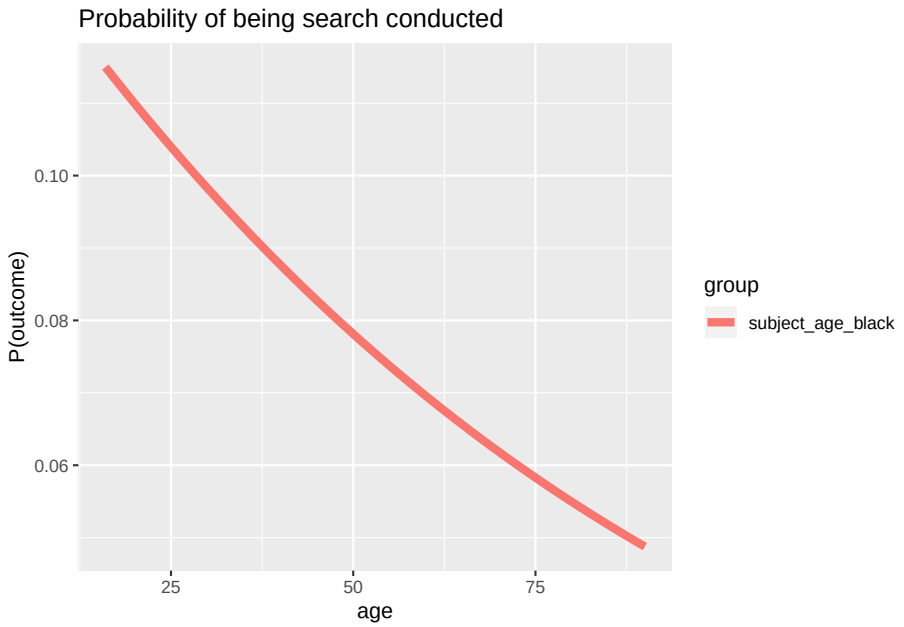
5.1 Plotting S-curves

Plotting an S curve for 1 row of coefficients using the coefficient equation for black drivers Ordering by all 50 states (found at https://wallethub.com/edu/most-least-diverse-states-in-america/38262/#methodology)

```
coeff_matrix <- read.csv("coeff_matrix_raw.csv")
matrix <- coeff_matrix[1,]
ages <- seq(-750,750,1)
coeff <- (matrix$intercept) + (matrix$subject_age)*ages
scurve <- exp(coeff)/(1+exp(coeff))
plot.data <- data.frame(subject_age=scurve, age=ages)
plot.data <- gather(plot.data, key=group, value=prob, subject_age)
ggplot(plot.data, aes(x=age, y=prob, color=group)) + geom_line(lwd=2) +
labs(x="age", y="P(outcome)", title="Probability of being search conducted")
```

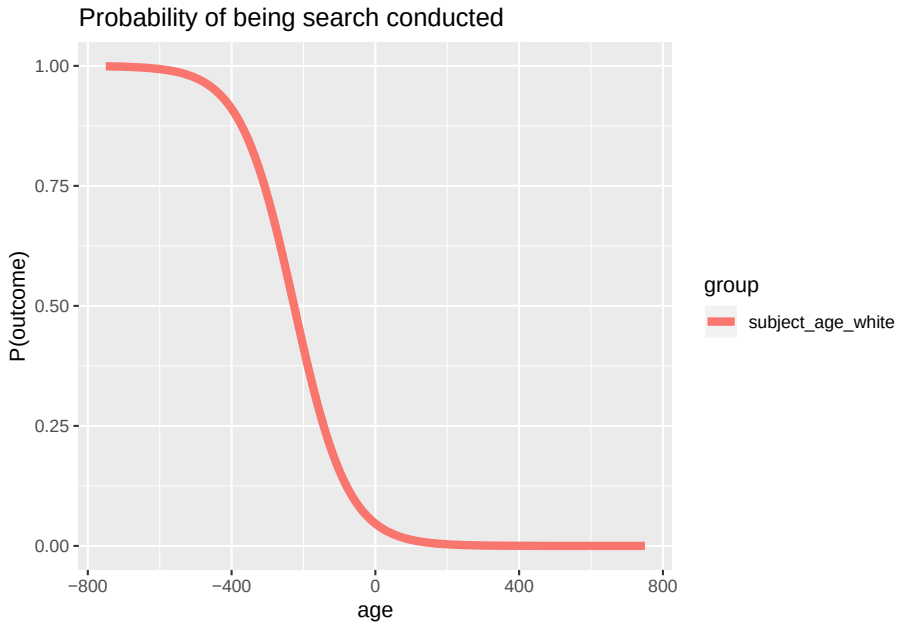


```
# looking at an appropriate age range of 16-90
ages <- seq(16,90,1)
coeff <- (matrix$intercept) + (matrix$subject_age)*ages
scurve <- exp(coeff)/(1+exp(coeff))
plot.data <- data.frame(subject_age_black=scurve, age=ages)
plot.data <- gather(plot.data, key=group, value=prob, subject_age_black)
ggplot(plot.data, aes(x=age, y=prob, color=group)) +
  geom_line(lwd=2) +
  labs(x="age", y="P(outcome)", title="Probability of being search conducted")
```

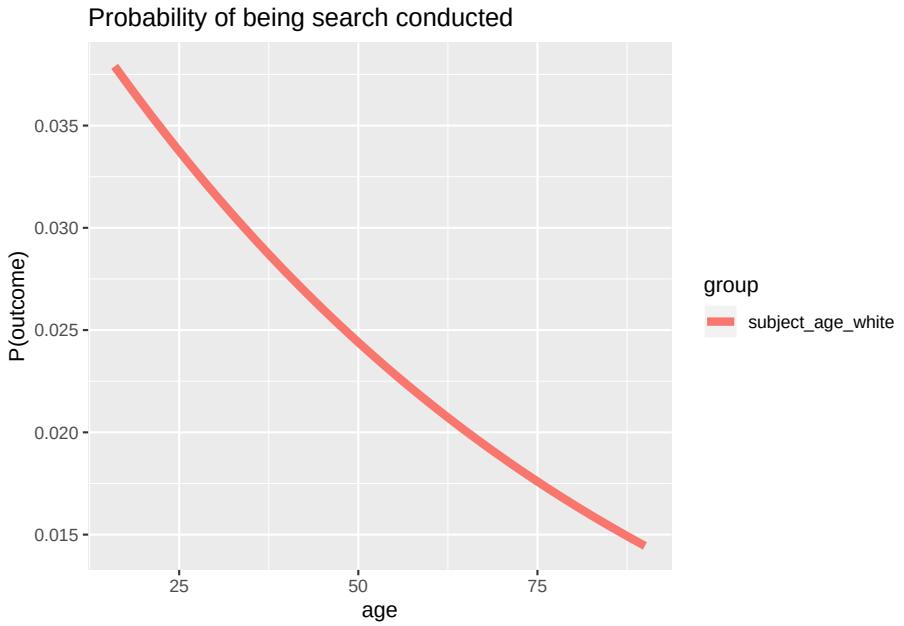


This displays the same equation but for white drivers.

```
ages <- seq(-750,750,1)
coeff <- ((matrix$intercept) + (matrix$subject_race)) +
  (matrix$subject_age + matrix$subject_age.subject_race)*ages
scurve <- exp(coeff)/(1+exp(coeff))
plot.data <- data.frame(subject_age_white=scurve, age=ages)
plot.data <- gather(plot.data, key=group, value=prob, subject_age_white)
ggplot(plot.data, aes(x=age, y=prob, color=group)) +
  geom_line(lwd=2) +
  labs(x="age", y="P(outcome)", title="Probability of being search conducted")
```




```
ages <- seq(16,90,1)
coeff <- ((matrix$intercept) + (matrix$subject_race)) +
  (matrix$subject_age + matrix$subject_age.subject_race)*ages
scurve <- exp(coeff)/(1+exp(coeff))
plot.data <- data.frame(subject_age_white=scurve, age=ages)
plot.data <- gather(plot.data, key=group, value=prob, subject_age_white)
ggplot(plot.data, aes(x=age, y=prob, color=group)) +
  geom_line(lwd=2) +
  labs(x="age", y="P(outcome)", title="Probability of being search conducted")
```



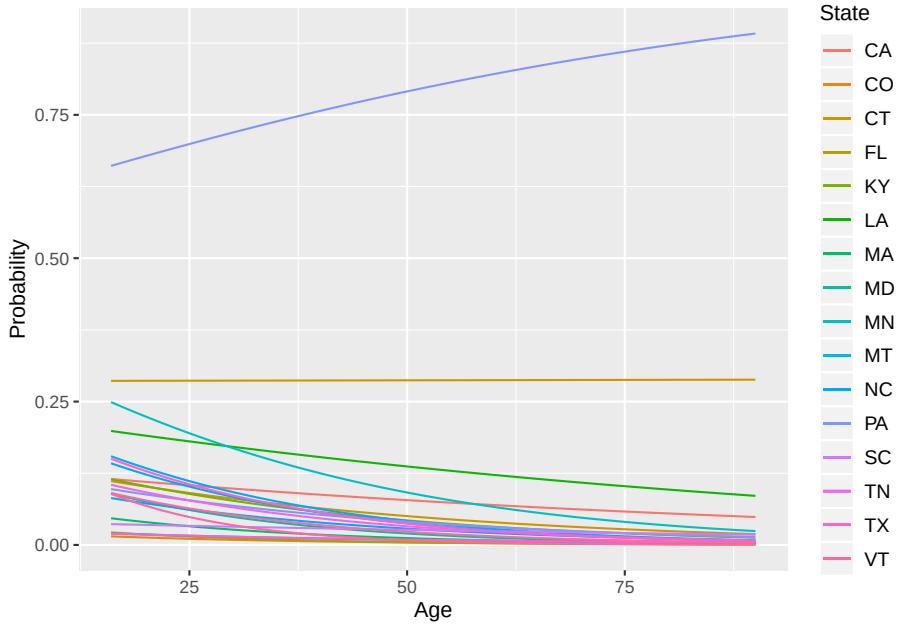
Plotting both curves (black and white drivers) for each state

```
for (num in 1:22){
matrix <- coeff_matrix[num,]
ages <- seq(16,90,1)
coeff_b <- (matrix$intercept) + (matrix$subject_age)*ages
coeff_w <- ((matrix$intercept) + (matrix$subject_race)) +
  (matrix$subject_age + matrix$subject_age.subject_race)*ages
scurve_b <- exp(coeff_b)/(1+exp(coeff_b))
scurve_w <- exp(coeff_w)/(1+exp(coeff_w))
plot_b <- data.frame(prob=scurve_b, age=ages, state = matrix$state_abbreviation)
plot_w <- data.frame(prob=scurve_w, age=ages, state = matrix$state_abbreviation)
print(ggplot(plot_b, aes(x=age, y=prob, color="blue")) +
geom_line(lwd=1.5) + geom_line(plot_w,
                               mapping = aes(x=age, y=prob, color="red"), lwd=1.5) +
  ggtitle(matrix$state_abbreviation))}
```

Compiling all states onto one graph for black drivers. The results show that generally, the probably of being searched is lower as an individual gets older for both races. Notice the one positive trend (maybe a mistake in the dataset that we could share with Stanford?)

```
my_plot <- ggplot()
for (num in 1:22){
matrix <- coeff_matrix[num,]
ages <- seq(16,90,1)
coeff <- (matrix$intercept) + (matrix$subject_age)*ages
```

```
scurve <- exp(coeff)/(1+exp(coeff))
plot <- data.frame(Probability=scurve, Age=ages, State = matrix$state_abbreviation)
my_plot <- my_plot + geom_line(plot, mapping = aes(x=Age, y=Probability, color=State))
my_plot + theme(legend.text = element_text(size = 10))
```



We then subtracted the s-curve results for black and white drivers with each curve being grouped by the political party the state voted for in the 2016 election. The results indicate that in the majority of states, the probability of being searched is greater for black individuals than for white individuals of every age.

```
my_plot <- ggplot()
for (num in 1:22){
matrix <- coeff_matrix[num,]
ages <- seq(16,90,1)
coeff <- (matrix$intercept) + (matrix$subject_age)*ages
coeff2 <- ((matrix$intercept) + (matrix$subject_race)) +
  (matrix$subject_age + matrix$subject_age.subject_race)*ages
scurve1 <- exp(coeff)/(1+exp(coeff))
scurve2 <- exp(coeff2)/(1+exp(coeff2))
scurve <- scurve1 - scurve2
plot <- data.frame(Probability=scurve, Age=ages,
                   State = matrix$state_abbreviation, makeup = matrix$state_politics)
my_plot <- my_plot + geom_line(plot, mapping = aes(x=Age, y=Probability, color=makeup))
my_plot + scale_color_manual(values = c("blue", "red")) +
  theme(legend.text = element_text(size = 10)) +
```

```
  ggtitle("Subtracted S-curves by blue or red state")
```

We then continued by grouping by various forms of diversity found at https://wallethub.com/edu/most-least-diverse-states-in-america/38262/#methodology). This was done on a sliding scale for the ranks.

```
my_plot <- ggplot()
for (num in 1:22){
matrix <- coeff_matrix[num,]
ages <- seq(16,90,1)
coeff <- (matrix$intercept) + (matrix$subject_age)*ages
coeff2 <- ((matrix$intercept) + (matrix$subject_race)) +
  (matrix$subject_age + matrix$subject_age.subject_race)*ages
scurve1 <- exp(coeff)/(1+exp(coeff))
scurve2 <- exp(coeff2)/(1+exp(coeff2))
scurve <- scurve1 - scurve2
plot <- data.frame(Probability=scurve, Age=ages,
                   State = matrix$state_abbreviation, makeup = matrix$state_politics,
                   diversity = matrix$racial_and_ethnic)
my_plot <- my_plot + geom_line(plot, mapping = aes(x=Age, y=Probability, color=diversity))}
my_plot + scale_color_gradient(low = "green", high = "red") +
  theme(legend.text = element_text(size = 10)) +
  ggtitle("Subtracted S-curves by racial and ethnic diversity")
```

Notice that none seem to have any trend. Further work can be done to analyze other variables to group over.

5.2 Veil of Darkness Nationwide

Running a veil of darkness logistic regression follows a similar process as using race and age as variables to predict a search being conducted. This time, we add a light binary variable

First we want to get the relevant datasets. This time, we splice out different datasets as those are empty Here we apply the relevant_datasets function to get the character string. Then, we can use base R splicing to remove empty datasets.

This code block defines the function, query_data, which calls the SQL command to retrieve the datasets with their speficied variables. Lastly, it will append each dataset to a list called datasets.

At last, the previous four functions will be called in the clean_data function. Clean_data does two main things: 1) make search conducted into a binary datatype 2) add night and day variables to our dataset Notice how this is very similar to the previous clean_data function only this time we add a add_night_day function

```
clean_data <- function(i){
  city_dataset <- datasets[[i]] %>% drop_na()

  # first add light and day variables
  tmp_lat <- coordinates[[i]][1]
  tmp_long <- coordinates[[i]][2]
  time_zone <- lutz::tz_lookup_coords(tmp_lat, tmp_long, warn = F)
  city_dataset <- add_night_day(city_dataset, time_zone, tmp_lat, tmp_long)

  # clean data
  if (typeof(city_dataset$search_conducted) == "character"){

    city_dataset <- city_dataset %>%
      filter(subject_race == "black" | subject_race == "white") %>%
      mutate(search_conducted = case_when(search_conducted == "TRUE" ~ 1,
                                          search_conducted == "FALSE" ~ 0))
  } else {
    # some datasets have search_conducted as already a dbl
    city_dataset <- city_dataset %>%
    filter(subject_race == "black" | subject_race == "white") %>%
    mutate(search_conducted = search_conducted)
  }

  city_dataset <- city_dataset %>%
    mutate(subject_race = as.factor(case_when(subject_race == "white"~"W", subject_race


  return(city_dataset)
}

datasets <- lapply(seq(datasets), clean_data)
```

fix_ages quickly sets any ages to a dbl data type

```
fix_ages <- function(city_dataset){
  city_dataset <- city_dataset %>% mutate(subject_age = as.numeric(subject_age))
  return(city_dataset)
}

datasets<- lapply(datasets, fix_ages)
```

Since we are analyzing multiple outputs from a logistic regression, we need a way to store those outputs. We decided to takes the coefficients from each logistic regression and make a dataframe out of them.

```
coefficient_matrix <- data.frame("intercept" = numeric(), "subject_raceW" = as.numeric(), "subjec

mapply(logistic_regression, datasets, datasets_of_interest)

colnames(coefficient_matrix) <- c("intercept", "subject_raceW", "subject_age", "is_dark", "subjec
```

Now that we have the coefficient_matrix we can select certain coefficients to plot the s-curve for white and black people along with day and night. There are four s-curve we want to plot: Black driver in the day, Black and night, white and day, and White and night. Thus, we'll make a dataframe for each scenario and then row bind all of them.

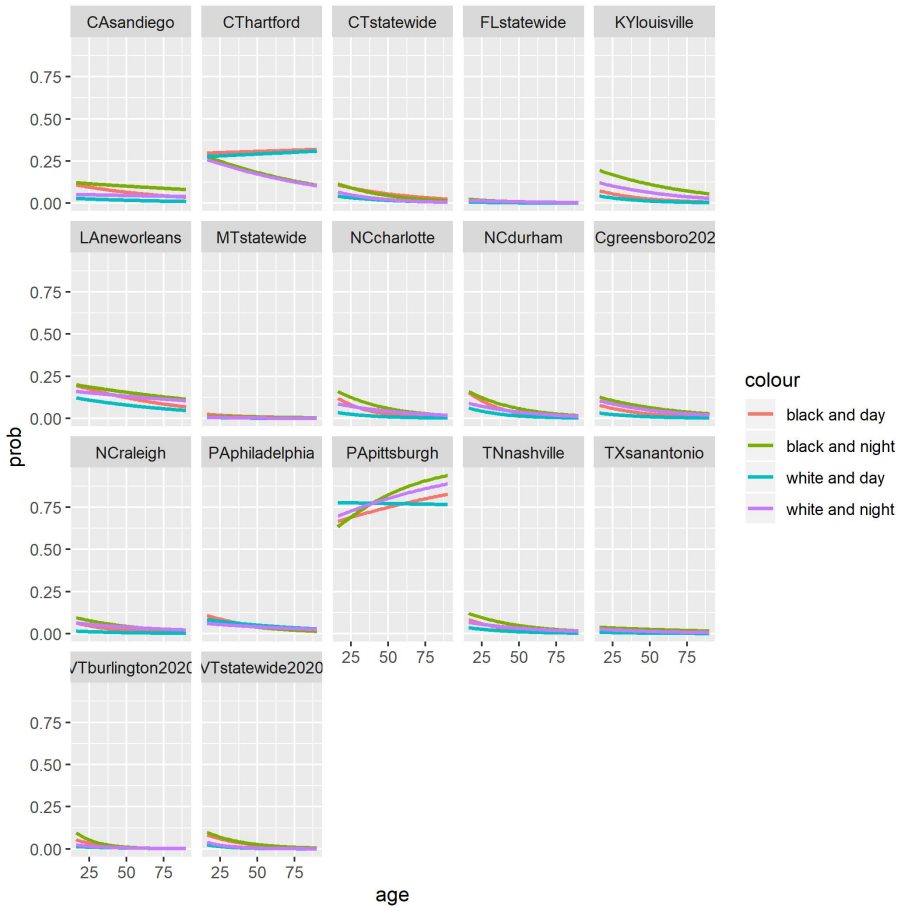
```
plot_all <- function(i){
  matrix <- coefficient_matrix[i,]
  ages <- seq(16,90,1)

  # black and day coefficients
  coeff_black <- (matrix$intercept) + (matrix$subject_age)*ages
  scurve <- exp(coeff_black)/(1+exp(coeff_black))
  plot.data <- data.frame(prob=scurve, age=ages,"state" = as.character(matrix$datasetnames), "rac

  # black and night
  coeff_black_night <- (matrix$intercept) + (matrix$is_dark) + (matrix$subject_age + matrix$`subj
  scurve_night <- exp(coeff_black_night)/(1+exp(coeff_black_night))
  plot.data_night <- data.frame(prob=scurve_night, age=ages, "state" = as.character(matrix$datase

  # white and day
  coeff_white_day <- matrix$intercept + matrix$subject_raceW + (matrix$subject_age + matrix$`subj
  scurve_white_day <- exp(coeff_white_day)/(1+exp(coeff_white_day))
  plot.data_white_day <- data.frame(prob=scurve_white_day, age=ages, "state" = as.character(matri

  # white and night
  coeff_white_night <- matrix$intercept + matrix$subject_raceW + matrix$is_dark + matrix$`subject
    (matrix$subject_age + matrix$`subject_age:is_dark` + matrix$`subject_raceW:subject_age`)*ages
  scurve_white_night <- exp(coeff_white_night)/(1+exp(coeff_white_night))
  plot.data_white_night <- data.frame(prob=scurve_white_night, age=ages, "state" = as.character(m

  full_plot_data <- bind_rows(plot.data_white_day, plot.data_white_night, plot.data, plot.data_ni

}

datum <- lapply(seq(datasets), plot_all)
all_cities <- do.call("rbind", datum)
```

Let's just plot the empirical values for the sake of sanity

Lastly, after getting the predicted probabilities from the different coefficients it

is time to plot them.

```
black_day_data <- all_cities %>% filter(race=="black" & is_dark=="FALSE")
black_night_data <- all_cities %>% filter(race=="black" & is_dark=="TRUE")
white_day_data <- all_cities %>% filter(race=="white" & is_dark=="FALSE")
white_night_data <- all_cities %>% filter(race=="white" & is_dark=="TRUE")

p <- ggplot() +
  geom_line(aes(x=age,y=prob, color="black and day"), data=black_day_data, lwd=1) + geo
  geom_line(aes(x=age,y=prob, color="white and day"), data=white_day_data, lwd=1) +
  geom_line(aes(x=age,y=prob, color="white and night"), data=white_night_data, lwd=1) +
facet_wrap(~state)
ggsave("cities_night_day_log_reg.pdf", p,scale=15)
```



Chapter 6

Final Words

We have finished a nice book.