Brandon Tran
MATH 154

Work in Progress for Final Project

Generating Test Data

For testing purposes, we will utilize the dataset for San Francisco. Note that we will have to omit any observations with NA entries for any one of `lat`, `lng`, `date`, `time`, and `subject_race`.

```
sf <- readRDS(file = url('https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_san_francis
sf.api <- sf %>%
  select(c(date, time, lat, lng, subject_race)) %>%
  drop_na()
```

Functions to Classify Times as Day or Night

First, we'll write a helper function that utilizes `POSIXct`, `hms`, and the data itself in order to first determine the sunrise and sunset times for a given day and coordinates, then return a boolean evaluating to `True` if the observed time was during the day and `False` if it occurred at night.

```
classify <- function(lat, long, date, time) {
  #Takes lat, long, time, and date as input and classifies TRUE if the time is at day and FALSE if at n

  sunrise <- sunrise.set(lat, long, date)$sunrise
  attributes(sunrise)$tzone <- 'America/Los_Angeles'
  sunrise <- as_hms(sunrise)

  sunset <- sunrise.set(lat, long, date)$sunset
  attributes(sunset)$tzone <- 'America/Los_Angeles'
  sunset <- as_hms(sunset)

  if (time < sunset && time >= sunrise)
    return(TRUE)

  else
    return(FALSE)
}
```

Next, we'll write a second function that will apply our helper function to any given dataframe, appending the booleans as a new column variable.

```
mutateClass <- function(df) {
  #Takes a dataframe and adds the boolean day/night classification found via `classify` as a new column

  classifications <- c()

  for (i in 1:nrow(df)) {
    classifications[i] <- classify(df$lat[i], df$lng[i], df$date[i], df$time[i])
  }

  df <- df %>%
```

```
    mutate(daytime = classifications)
}
```

This function can now be used to generate `True` and `False` classifications for our new variable, `daytime`.

Putting it All Together

We now have some data, and we have functions that will classify an observation (really, a traffic stop) as during or not during the daytime. We now want to parse data together to form an appropriate sample of the State of California, which will put us in a good place for both our Shiny visualization and the statistical testing for signficance. Note that for some datasets (namely, Los Angeles, San Diego, Oakland, and San Jose), `lat` and `long` are not included. Luckily, we only need the location in order to calculate the exact sunrise/sunset times, and this can be generalized to each city. Our function will include functionality to create the necessary `lat` and `long` variables for datasets without such information.

```
getCityCoords <- function(link) {
  #Takes a link to data and assigns appropriate lat and lng coordinates.

  if (grepl('los_angeles', link))
    return(c(34.052235, -118.243683))

  if (grepl('san_diego', link))
    return(c(32.715736, -117.161087))

  if (grepl('oakland', link))
    return(c(37.804363, -122.271111))

  if (grepl('san_jose', link))
    return(c(37.335480, -121.893028))
}

getData <- function(link) {
  #Takes a link to data and configures it as necessary.  Samples n/10000 traffic stops.  Returns a muta

  set.seed(47)

  raw <- readRDS(file = url(link))

  if (! 'lat' %in% colnames(raw)) {
    raw <- raw %>%
      mutate(lat = getCityCoords(link)[1]) %>%
      mutate(lng = getCityCoords(link)[2])
  }

  clean <- raw %>%
    select(c(date, time, lat, lng, subject_race)) %>%
    drop_na() %>%
    sample_n(nrow(raw)/100)

  return(mutateClass(clean))
}
```

We now have a function, `getData`, that will effectively scrape all of the necessary datasets and generate the samples and daytime booleans that will be used to conduct our statistical analyses and visualizations. The

following function calls will generate individual datasets for each city.

```
sf <- getData('https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_san_francisco_2019_08_
ok <- getData('https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_oakland_2019_08_13.rds
sj <- getData('https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_san_jose_2019_08_13.rds
bf <- getData('https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_bakersfield_2019_08_13
la <- getData('https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_los_angeles_2019_08_13
sd <- getData('https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_san_diego_2019_08_13.r
```

Our final goal is to create one large dataset containing all of the data samples.

```
parseData <- function(links, city.codes) {
  #Takes in a list of URLs and city codes and merges the datasets that are scraped from the sources.

  master.frame <- getData(links[1])
  master.frame <- master.frame %>% mutate(city = city.codes[1])

  for (i in 2:length(links)) {
    data <- getData(links[i])
    data <- data %>% mutate(city = city.codes[i])
    master.frame <- full_join(master.frame, data)
  }

  return(master.frame)
}
```

And, we're done! Running the following code will generate the dataset that we need:

```
links <- c('https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_san_francisco_2019_08_13.
           'https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_oakland_2019_08_13.rds',
           'https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_san_jose_2019_08_13.rds',
           'https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_bakersfield_2019_08_13.rds
           'https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_los_angeles_2019_08_13.rds
           'https://stacks.stanford.edu/file/druid:hp256wp2687/hp256wp2687_ca_san_diego_2019_08_13.rds')
city.codes <- c('sf', 'ok', 'sj', 'bf', 'la', 'sd')

ca.df <- parseData(links, city.codes)
```

To save a local copy of this dataframe, run the following code chunk. A copy has been saved to the repo in the `Data` folder.

```
save(ca.df, file='cadata.Rda')
```