Problem set 3: Worked solutions

Statistics and statistical programming
Northwestern University
MTS 525

Aaron Shaw

October 12, 2020

Contents

Programming challenges

PC1. Learn about the data.

The documentation makes it sound like the Illinois data is particularly messy/incomplete. It explains that the data is limited to 2012-2017 (instead of extending back to 2004) "due to format issues and relevance." The notes point out that data related to searches vary substantially year-to-year, suggesting that the measurement was inconsistent and that caution should be exercised in the analysis of this measure in particular. The `search_conducted` variable seems to include searches of any kind (vehicles, driver, passengers). I also notice that the codebook does not provide much insight into *how* `subject_race` and `subject_sex` information were collected, measured, and/or recoded. I think it's best to presume that the records reflect the judgments/notes/documents accessed by the officer(s) and/or police forces involved rather

than the perspectives of the subjects of the traffic stops. There are likely some inconsistencies in these measures as well.

PC2. Import, explore, clean

I'll use the tidyverse `read_csv()` function to do this. While I'm at it, I'll import the `lubridate` library because I already know this is time series data and there are a bunch of helpful functions in `lubridate` that I will probably want to use.

```
library(tidyverse)
library(lubridate)
data.dir <- "https://communitydata.science/~ads/teaching/2020/stats/data/week_05"
filename <- "il_statewide_sample-2020_04_01.csv"
ilstops <- read_csv(url(paste(data.dir, filename, sep = "/")))
```

```
## Warning: 30129 parsing failures.
##  row  col expected     actual         file
## 1153 beat a double east       <connection>
## 1155 beat a double Rt1/325N   <connection>
## 1158 beat a double village    <connection>
## 1160 beat a double PULASKI CO <connection>
## 1161 beat a double P21        <connection>
## .... .... ........ .......... ............
## See problems(...) for more details.
```

```
ilstops
```

```
## # A tibble: 127,064 x 29
##    raw_row_number date       time  location  beat subject_age subject_race
##             <dbl> <date>     <tim>    <dbl> <dbl> <lgl>       <chr>
##  1             71 2012-01-01 06:00    60160  1924 NA          white
##  2             81 2012-01-01 07:34       NA  1654 NA          asian/pacif~
##  3            170 2012-01-01 18:35       NA  1324 NA          hispanic
##  4            274 2012-01-02 00:18       NA  1633 NA          white
##  5            344 2012-01-02 09:08    60608  1232 NA          white
##  6            379 2012-01-02 11:00    60621   733 NA          black
##  7            410 2012-01-02 13:00       NA  1613 NA          hispanic
##  8            570 2012-01-02 21:01    60644  1512 NA          black
##  9            729 2012-01-03 09:40    60618  2131 NA          white
## 10            789 2012-01-03 13:29    60639  2534 NA          hispanic
## # ... with 127,054 more rows, and 22 more variables: subject_sex <chr>,
## #   department_id <dbl>, department_name <chr>, type <chr>, violation <chr>,
## #   citation_issued <lgl>, warning_issued <lgl>, outcome <chr>,
## #   contraband_found <lgl>, contraband_drugs <lgl>, contraband_weapons <lgl>,
## #   search_conducted <lgl>, search_person <lgl>, search_vehicle <lgl>,
## #   search_basis <chr>, reason_for_stop <chr>, vehicle_make <chr>,
## #   vehicle_year <dbl>, raw_DriverRace <dbl>, raw_ReasonForStop <dbl>,
## #   raw_TypeOfMovingViolation <dbl>, raw_ResultOfStop <dbl>
```

Recode and explore key variables

Right off the bat, I see a lot of `NA` values and some things likely to need recoding. Let's look a little more closely at the variables we are going to work with.

```
key_variables <- c("date", "vehicle_year", "subject_race", "subject_sex", "search_conducted")
```

```
lapply(ilstops[key_variables], class)
```

```
## $date
## [1] "Date"
##
## $vehicle_year
## [1] "numeric"
##
## $subject_race
## [1] "character"
##
## $subject_sex
## [1] "character"
##
## $search_conducted
## [1] "logical"
```

Indeed, I want to create some factors for the `subject_race` and `subject_sex` variables.

```
ilstops$subject_race <- factor(ilstops$subject_race)
ilstops$subject_sex <- factor(ilstops$subject_sex)
```

PC3 Summarize key variables

Now, let's look at summaries of the key variables. Keep in mind that these are all variables calculated over all traffic stops in the dataset.

```
lapply(ilstops[key_variables], summary)
```

```
## $date
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2012-01-01" "2013-06-29" "2015-01-22" "2015-01-09" "2016-07-17" "2017-12-31"
##
## $vehicle_year
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1900    2001    2006    2005    2010    2021     137
##
## $subject_race
## asian/pacific islander                 black              hispanic
##                   4053                 25627                 16940
##                  other                 white                  NA's
##                    335                 80105                     4
##
## $subject_sex
## female   male   NA's
##  47698  79364      2
##
## $search_conducted
##    Mode   FALSE    TRUE    NA's
## logical  120942    5947     175
```

Note that the number of missing (`NA`) values is quite different depending on the variable.

Recode nonsense `vehicle_year` values

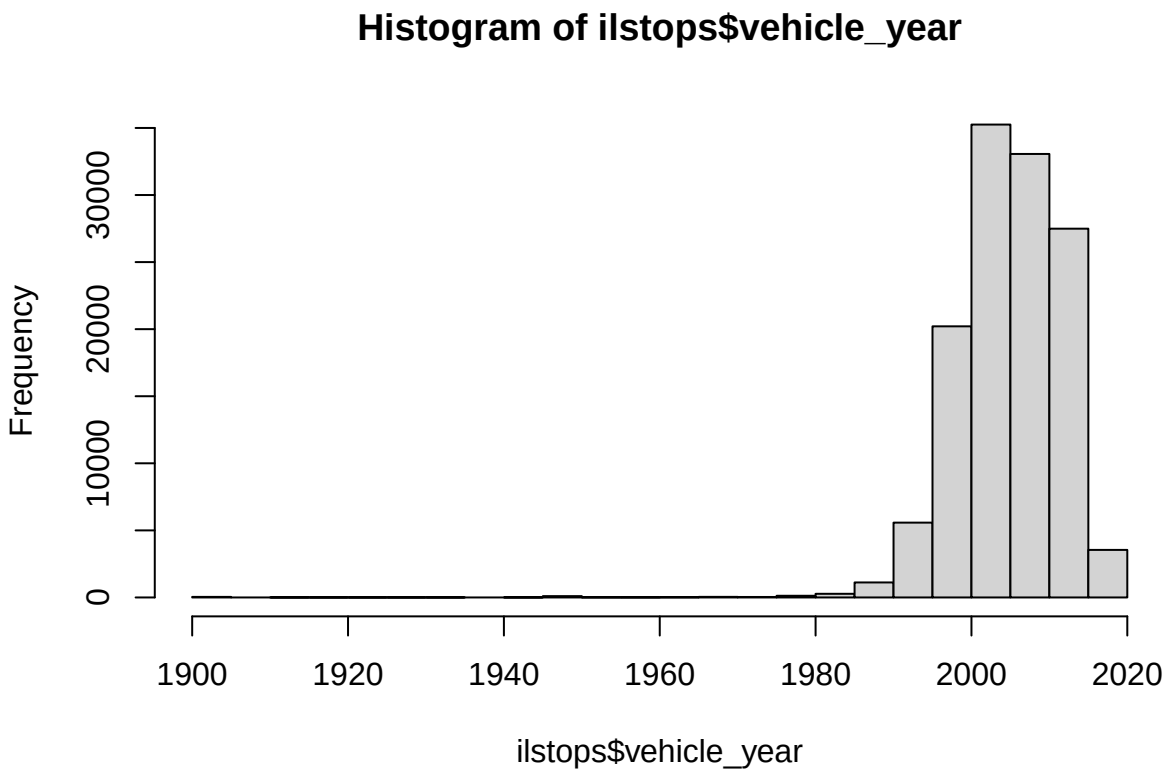
Review those summaries and you might notice something a little funny about the `vehicle_year` variable. Doesn't this data end in 2017? None of the `vehicle_year` values should be more recent than 2018 (if we're feeling generous since car manufacturers sometimes start selling "next year's car" at the end of a given year?). I'm also modestly skeptical that the police pulled over a car from 1900, but at least that's a plausible value! In the abence of additional information about the provenance of this variable or reasons why values larger than 2018 might be reasonable, I will go ahead and recode anything impossibly new as missing (`NA`).

```
ilstops$vehicle_year[ilstops$vehicle_year > 2018] <- NA

summary(ilstops$vehicle_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1900    2001    2006    2005    2010    2018     139
```

```
hist(ilstops$vehicle_year)
```



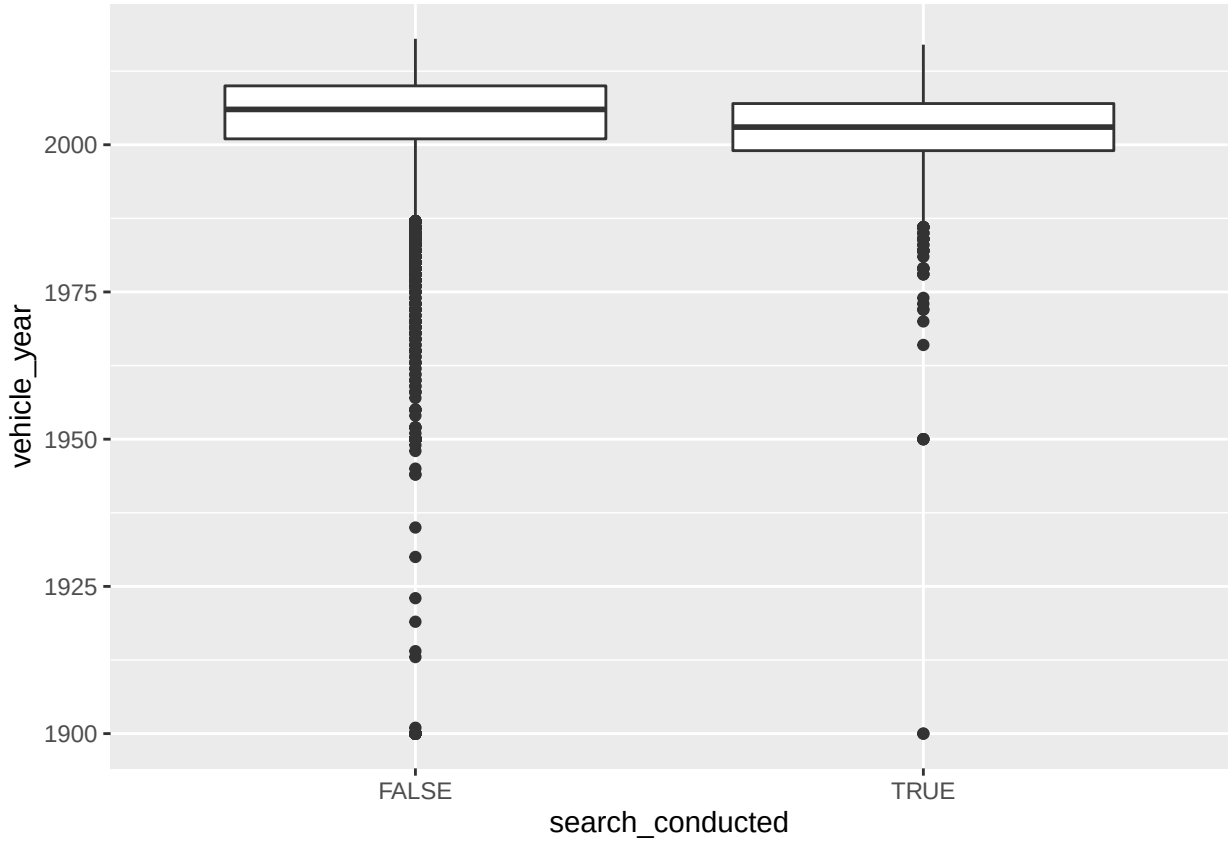
PC4. Summarize bivariate relationships

For bivariate summaries and comparisons, I'll walk through things variable by variable. In all of these cases except for the `date` variable, I'm considering the bivariate relationships in aggregate (rather than taking the "over time" character of the data into account). This is good to keep in mind since it might mean that the summaries mask temporal trends.

Searches by `vehicle year`

I'll start with a continuous predictor, `vehicle_year`. Since the outcome variable is a dichotomous category, I can look at this by calculating summary statistics within the two outcomes or create a pair of plots. I love visual summaries, so I'll start with side-by-side boxplots:

```
ilstops %>%
  filter(!is.na(vehicle_year) & !is.na(search_conducted)) %>%
  ggplot(aes(x = search_conducted, y = vehicle_year)) +
  geom_boxplot()
```



It looks like stops of older vehicles are, on average, a little bit more likely to result in a search, but stops involving the very oldest vehicles tend to result in fewer searches.

With that in mind, I'll calculate summary statistics across the two groups. One way to do this is with a call to the `tapply` function nested within a call to the `with()` command. I don't think we've used `with` yet, but it can allow me to identify the object I want to use (the first argument to `with()`) to run some command on (the second argument, in this case the call to `tapply()`).

```
with(
  ilstops,
  tapply(vehicle_year, search_conducted, summary)
)
```

```
## $`FALSE`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1900    2001    2006    2005    2010    2018     132
##
## $`TRUE`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1900    1999    2003    2003    2007    2017       7
```

Indeed, as I suspected based on the histograms, the center of the distribution for stops resulting in searches is a little lower (older vehicles) than among stops not resulting in searches. I'll calculate the standard deviations with a separate command:

```
with(ilstops, tapply(vehicle_year, search_conducted, sd, na.rm = TRUE))
```

```
##    FALSE     TRUE
## 6.748166 6.655972
```

The stops not resulting in searches (`FALSE`) have a larger/wider spread of vehicle ages than those that do result in searches (`TRUE`). This difference is likely driven by the larger number of old vehicle outliers inovlved in stops that do not result in searches. This is a nice example of how the visual and numerical summaries might help convey a somewhat subtle feature of my data.

Searches by `subject_sex`

Next, we'll focus on the `subject_sex` variable. Both the outcome and this predictor variable are categorical measures that take only two (non-missing) values. That means that I can focus on creating contingency tables to report the raw numbers of traffic stops in each of the four possible combinations of the two variables. I've provided a couple of methods here:

```
table(ilstops$subject_sex, ilstops$search_conducted)
```

```
##
##          FALSE  TRUE
##   female 46098  1536
##   male   74842  4411
```

```
## the xtabs() command produces nicely formatted output that includes both variable names
xtabs(~ subject_sex + search_conducted, ilstops)
```

```
##            search_conducted
## subject_sex FALSE  TRUE
##      female 46098  1536
##      male   74842  4411
```

One way to get the proportions in Base-R is to call the `proportions()` function on a contingency table.

```
proportions(
  xtabs(~ subject_sex + search_conducted, ilstops)
)
```

```
##            search_conducted
## subject_sex      FALSE       TRUE
##      female 0.36329963 0.01210526
##      male   0.58983190 0.03476321
```

That's a lot of significant digits. . . especially for proportions. I can clean it up a bit by nesting everything within a call to the `round()` function and restricting the output to just two significant digits.

```
round(
  proportions(
    xtabs(~ subject_sex + search_conducted, ilstops)
  ),
  digits = 2
)
```

```
##            search_conducted
## subject_sex FALSE TRUE
##      female  0.36 0.01
##      male    0.59 0.03
```

Much better! However, notice that these proportions are all calculated against the *total* number of non-missing values. I can specify row or column proportions with an additional argument to the `proportions` function to specify which margins to use. Following R conventions, `margin=1` calculates proportions row-wise and `margin=2` calculates them column-wise:

```
## Row-wise proportions
round(
  proportions(
    xtabs(~ subject_sex + search_conducted, ilstops), 1
  ),
  digits = 2
)
```

```
##            search_conducted
## subject_sex FALSE TRUE
##      female  0.97 0.03
##      male    0.94 0.06
```

```
## Column-wise proportions
round(
  proportions(
    xtabs(~ subject_sex + search_conducted, ilstops), 2
  ),
  digits = 2
)
```

```
##            search_conducted
## subject_sex FALSE TRUE
##      female  0.38 0.26
##      male    0.62 0.74
```

These different marginal proportions support distinct statements about the data. Here is an example for each:

- **Within the rows (`subject_sex`):** *the proportion of searches in stops of male drivers is twice as large as the proportion in stops of female drivers.*
- **Within the columns (`search_conducted`):** *stops that resulted in searches involved male drivers almost* 75% *of the time.*

Searches by `subject_race`

The `subject_race` variable is also categorical, but takes one of five non-missing values in this data. As a result, the basic structure/approach for my summaries is almost identical to that pursued with `subject_sex`.

First, tabular summaries of raw counts and proportions:

```
xtabs(~ subject_race + search_conducted, ilstops)
```

```
##                          search_conducted
## subject_race              FALSE  TRUE
##   asian/pacific islander  3981    68
##   black                  23742  1806
##   hispanic               15865  1049
##   other                    320    14
##   white                  77033  3010
```

In this case, I think the marginal proportions are far more interesting than the aggregate proportions. Here are proportions of searches within each category of `subject_race` (row-wise given the way I've structured the call to `xtabs`):

```
## Proportions of searches among stops within groups
round(
  proportions(
    xtabs(~ subject_race + search_conducted, ilstops), 1
  ),
  digits = 2
)
```

```
##                          search_conducted
## subject_race              FALSE TRUE
##   asian/pacific islander  0.98 0.02
##   black                   0.93 0.07
##   hispanic                0.94 0.06
##   other                   0.96 0.04
##   white                   0.96 0.04
```

Notice that the proportion of black and hispanic drivers with stops resulting in searches is higher than within other groups.

Now, I'll calculate the proportions of each `subject_race` category among stops resulting in searches and stops that do not result in searches:

```
## Proportions of groups within those stops (not) resulting in searches
round(
  proportions(
    xtabs(~ subject_race + search_conducted, ilstops), 2
  ),
  digits = 2
)
```

```
##                          search_conducted
## subject_race              FALSE TRUE
##   asian/pacific islander  0.03 0.01
##   black                   0.20 0.30
##   hispanic                0.13 0.18
##   other                   0.00 0.00
##   white                   0.64 0.51
```

Here the noteworthy comparisons again arise within the black and hispanic categories. Both groups account for a substantially larger proportion of stops resulting in searches vs. those that do not result in searches.

For the sake of completeness/comparison, here's a way to do similar cross-tabulations in chunks of tidyverse code. This first bit summarizes the number of stops and proportion of total stops accounted for within each of the categories of `subject_race`.

```
ilstops %>%
  group_by(subject_race) %>%
  filter(!is.na(subject_race)) %>%
  summarize(
    n_stops = n(),
    prop_total_stops = round(n() / nrow(ilstops), digits = 3),
  )
```

```
## # A tibble: 5 x 3
##   subject_race           n_stops prop_total_stops
##   <fct>                    <int>            <dbl>
## 1 asian/pacific islander    4053            0.032
## 2 black                   25627            0.202
```

```
## 3 hispanic                16940           0.133
## 4 other                     335           0.003
## 5 white                   80105           0.63
```

In that block I first make a call to `group_by()` to tell R that I want it to run subsequent commands on the data "grouped" within the categories of `subject_race`. Then I pipe the grouped data to `summarize()`, which I use to calculate the number of stops within each group (in this data that's just the number of observations within each group) as well as the proportion of total stops within each group.

What about counting up the number and proportion of searches within each group? One way to think about that is as another call to `summarize()` (since, after all, I want to calculate the summary information for searches within the same groups). Within the Tidyverse approach to things, this kind of summarizing within groups and within another variable (`search_conducted` in this case) can be accomplished with the `across()` function.

In general, the `across()` function seems to usually be made within a call to another verb like `summarize()` or `mutate()`. The syntax for `across()` is similar to these others. It requires two things: (1) at least one variable to summarize across (`search_conducted` here) and (2) the outputs I want.

In this particular case, I'll use it to calculate the within group sums of `search_conducted`. Notice that I also filter out the missing values from `search_conducted` before I call `summarize` here.

```
ilstops %>%
  group_by(subject_race) %>%
  filter(!is.na(subject_race), !is.na(search_conducted)) %>%
  summarize(
    across(search_conducted, sum)
  )
```

```
## # A tibble: 5 x 2
##   subject_race           search_conducted
##   <fct>                             <int>
## 1 asian/pacific islander               68
## 2 black                              1806
## 3 hispanic                           1049
## 4 other                                14
## 5 white                              3010
```

If I want `across()` to calculate more than one summary, I need to provide it a list of things (in a `name = value` format sort of similar to `summarize()` or `mutate()`).

```
ilstops %>%
  group_by(subject_race) %>%
  filter(!is.na(subject_race) & !is.na(search_conducted)) %>%
  summarize(
    across(
      search_conducted,
      list(
        sum = sum,
        over_n_stops = mean
      )
    )
  )
```

```
## # A tibble: 5 x 3
##   subject_race           search_conducted_sum search_conducted_over_n_stops
##   <fct>                                 <int>                         <dbl>
## 1 asian/pacific islander                   68                        0.0168
```

```
## 2 black                                1806                      0.0707
## 3 hispanic                             1049                      0.0620
## 4 other                                  14                      0.0419
## 5 white                                3010                      0.0376
```

I can clean this up a bit by using two functions to the output in descending order by one of the columns. I do this with a nested call to two functions `arrange()` and `desc()`. I can also insert my earlier summary statistics for the number and proportions of stops by group back into the table.

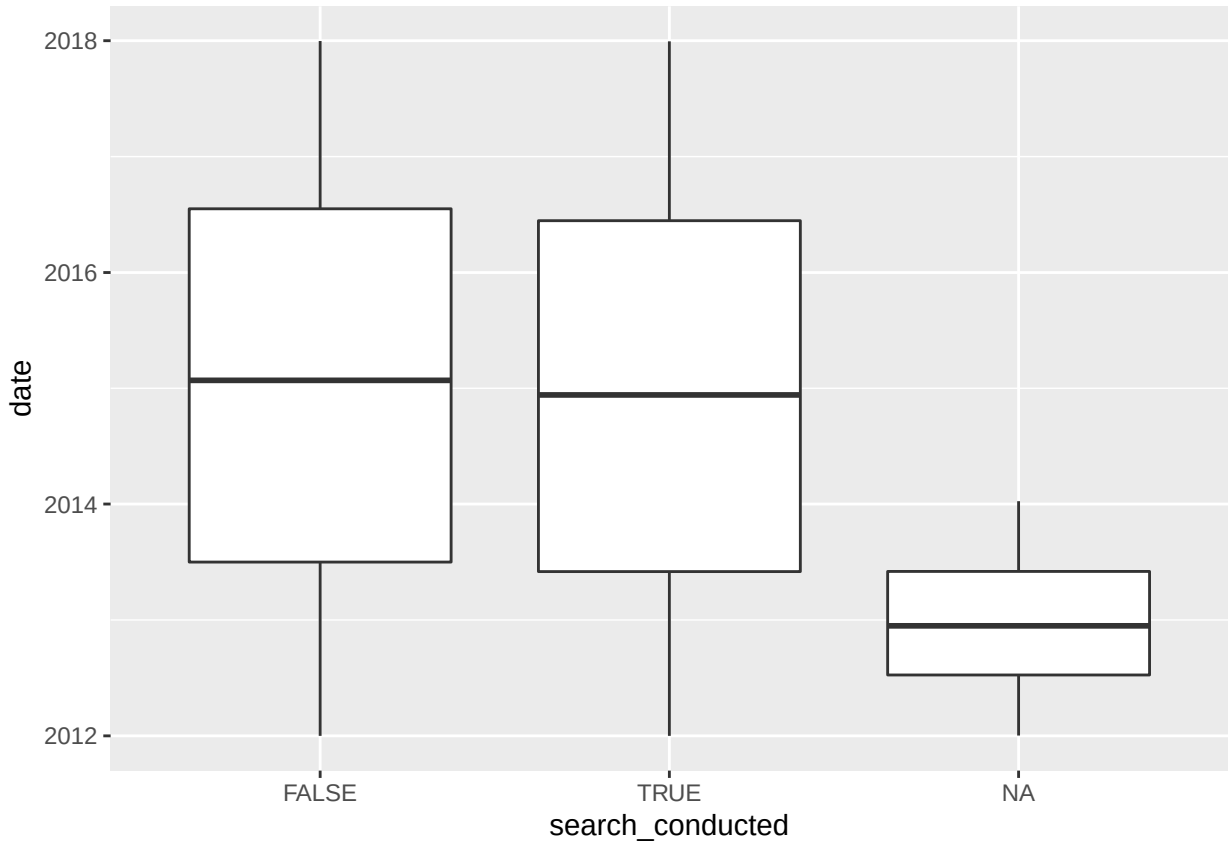
```
ilstops %>%
  group_by(subject_race) %>%
  filter(!is.na(subject_race) & !is.na(search_conducted)) %>%
  summarize(
    n_stops = n(),
    prop_total_stops = round(n() / nrow(ilstops), digits = 3),
    across(
      search_conducted,
      list(
        sum = sum,
        over_n_stops = mean
      )
    )
  ) %>%
  arrange(desc(n_stops))
```

```
## # A tibble: 5 x 5
##   subject_race    n_stops prop_total_stops search_conducted~ search_conducted_o~
##   <fct>             <int>            <dbl>             <int>                <dbl>
## 1 white             80043            0.63               3010               0.0376
## 2 black             25548            0.201              1806               0.0707
## 3 hispanic          16914            0.133              1049               0.0620
## 4 asian/pacific ~    4049            0.032                68               0.0168
## 5 other               334            0.003                14               0.0419
```

Searches by `date`

Last, but not least, I can inspect how the number of searches is distributed over time. I can do this a few ways, but once again will start with a visual summary because that is how I prefer to approach things. Remember that `search_conducted` is a dichotomous categorical measure, so a boxplot is a good starting point.

```
ilstops %>%
  ggplot(aes(x = search_conducted, y = date)) +
  geom_boxplot()
```



How considerate of ggplot to include the `NA` category by default! Notice how seemingly all of the missing values come from 2012-2014? There's some of that missing data the SOPP folks warned us about in their documentation. Let's inspect things a bit more closely:

```
ilstops %>%
  group_by(search_conducted) %>%
  summarize(
    min = min(date, na.rm = TRUE),
    max = max(date, na.rm = TRUE)
  )
```

```
## # A tibble: 3 x 3
##   search_conducted min        max
##   <lgl>            <date>     <date>
## 1 FALSE            2012-01-01 2017-12-31
## 2 TRUE             2012-01-01 2017-12-30
## 3 NA               2012-01-02 2014-01-10
```

All the missing data for `search_conducted` comes within a two-year range.

I can use `filter` to look at the `search_conducted` variable within that two year range alone:

```
ilstops %>%
  filter(date < as_date("2014-01-11")) %>%
  group_by(search_conducted) %>%
  summarize(
    n = n()
  )
```

```
## # A tibble: 3 x 2
```

```
##   search_conducted     n
##   <lgl>            <int>
## 1 FALSE            40076
## 2 TRUE              2044
## 3 NA                 175
```

Most of the `search_conducted` data between 2012-2014 isn't missing, even if that's the only period within which missing values for `search_conducted` appear. There's not much we can do about this, but it's worth taking into consideration as we go.

PC5 Searches within `subject_race` groups over time

The goal here is to synthesize pieces of the work we did in PC3 and PC4 to analyze subgroups of the stops and the stops resulting in searches over time. I'll start by reorganizing/tidying the data to support the creation of my time series plots.

5.1 Binning data within time periods

I want to plot this data as a time series of some numeric measure, but right now my outcome variable is a categorical indicator. How can I plot TRUE/FALSE values over time? One good way is to bin my data into meaningful time-periods (e.g., months or weeks) and count the number of TRUE values (and/or calculate proportions of TRUE values) within each bin.

Let's do that using some of the handy features of `lubridate` for working with date objects. I have demonstrated a few approaches to binning and grouping in the R tutorial materials, but here's yet another that takes advantage of some `lubridate` features (specifically the call to `format` within the `mutate` step below) quite effectively:

```
## library(lubridate)

ilstops_monthly <- ilstops %>%
  filter(!is.na(subject_race)) %>%
  mutate(
    yearmonth = format(date, "%Y-%m")
  ) %>%
  group_by(yearmonth) %>%
  summarize(
    stops = n(),
    searches = sum(as.numeric(search_conducted), na.rm = T),
    prop_searched = round(searches / stops, 3)
  )

ilstops_monthly
```

```
## # A tibble: 72 x 4
##    yearmonth stops searches prop_searched
##    <chr>     <int>    <dbl>         <dbl>
##  1 2012-01    1864       94         0.05
##  2 2012-02    1907      106         0.056
##  3 2012-03    1975       86         0.044
##  4 2012-04    1725       78         0.045
##  5 2012-05    1909       70         0.037
##  6 2012-06    1816       83         0.046
##  7 2012-07    1584       73         0.046
##  8 2012-08    1778       70         0.039
##  9 2012-09    1700       81         0.048
```

```
## 10 2012-10    1674       93         0.056
## # ... with 62 more rows
```

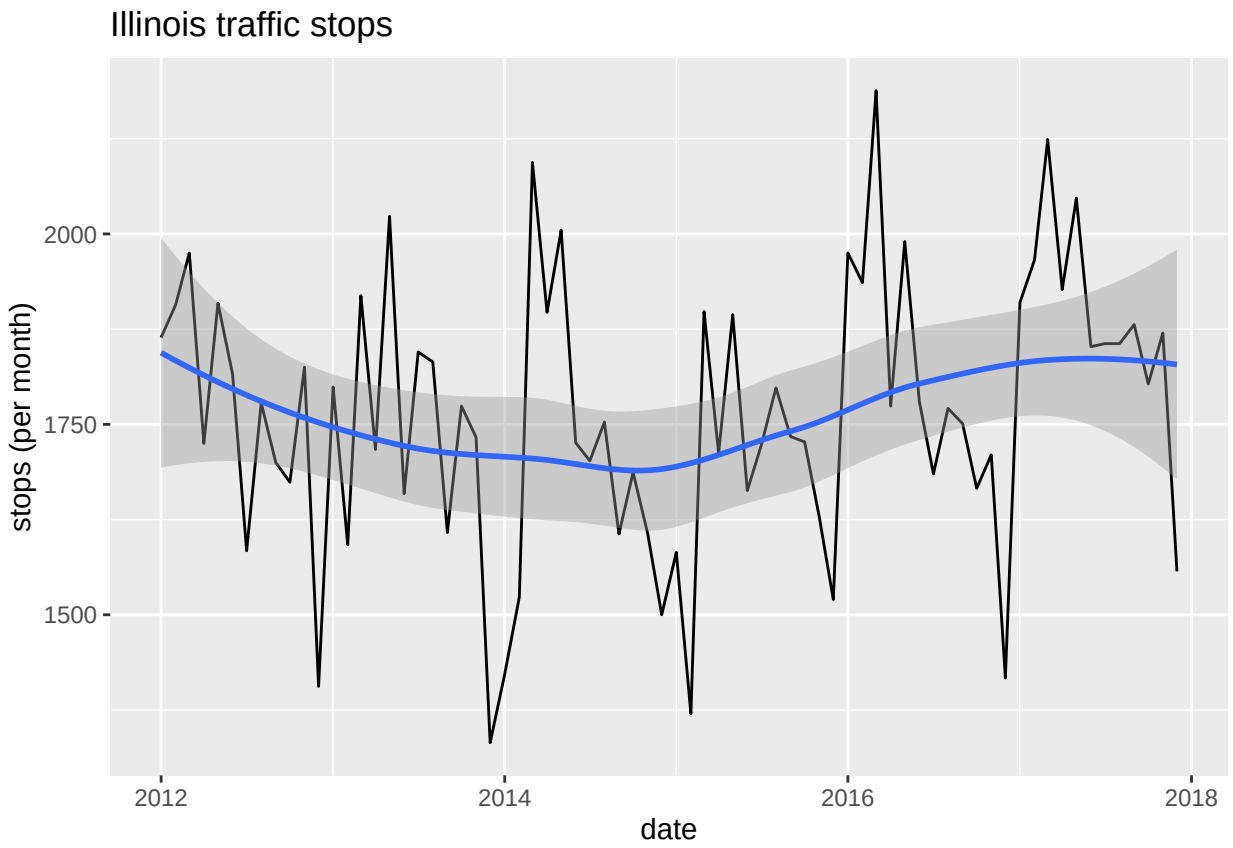
Looks great, except do you notice how my new `yearmonth` variable is no longer recognized as a date or date-time object? I think I can fix that with the handy `parse_date_time` function from `lubridate`:

```
ilstops_monthly$date <- parse_date_time(ilstops_monthly$yearmonth, "ym")
class(ilstops_monthly$date)
```

```
## [1] "POSIXct" "POSIXt"
```

Now I can plot it since `ggplot` knows what to do with objects in the `POSIXct` class. I'm also going to add a type of layer we haven't worked with yet (`geom_smooth`) which incorporates a smoothed conditional mean to the plot (appropriately enough in this dataset, in the form of a thin blue line).

```
ilstops_monthly %>%
  ggplot(aes(x = date, y = stops)) +
  geom_line() +
  geom_smooth() +
  labs(title = "Illinois traffic stops", y = "stops (per month)")
```

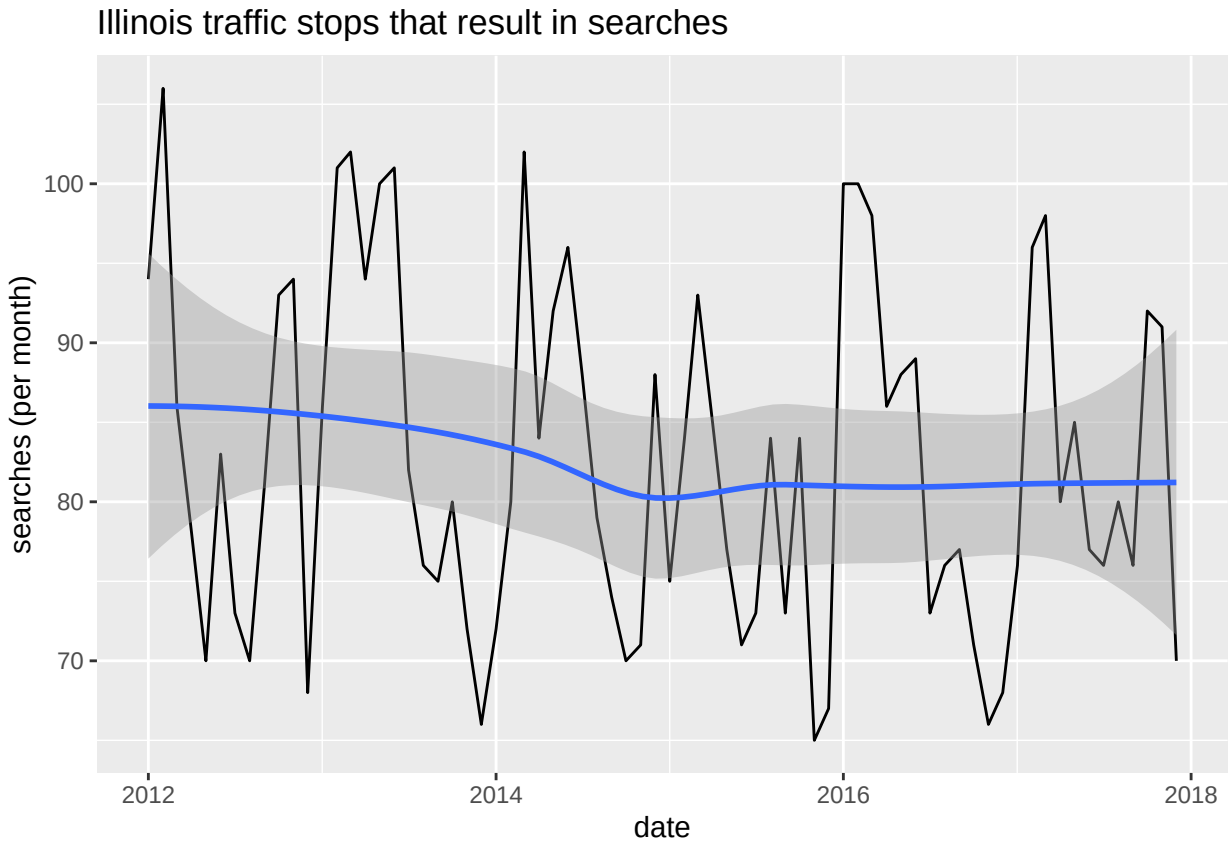


Turns out traffic stops are not distributed smoothly across the months of the year (I notice, in particular, the big dips and spikes around the end of each year). It also looks like there is a slight ∪-shaped trend overall, but the shifts do not look very large relative to either the month-to-month variations or the total number of stops per year.

Here's a similar plot for searches:

```
ilstops_monthly %>%
  ggplot(aes(x = date, y = searches)) +
  geom_line() +
```

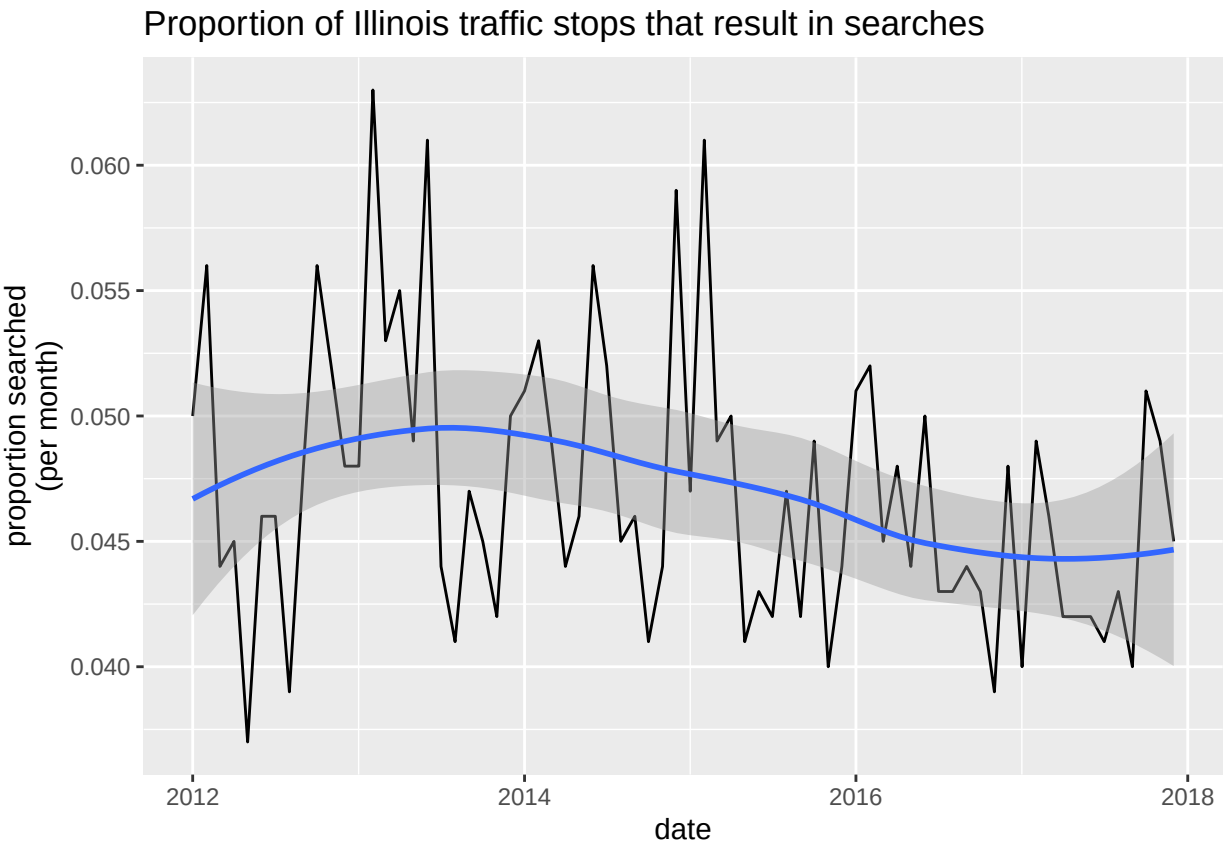
```
  geom_smooth() +
  labs(
    title = "Illinois traffic stops that result in searches",
    y = "searches (per month)"
  )
```



Turns out searches are also not distributed smoothly across the months of the year (again, the big dips and spikes around the end of the year). In this case, there seems to be a very slight downward trend on average, but this decrease mostly appears in the first couple of years of the data and does not look very large relative to the month-to-month variations or the total number of searches per year (maybe ten fewer searches per month on average during the latter half of the dataset?).

Let's try looking at the proportion of stops that result in searches.

```
ilstops_monthly %>%
  ggplot(aes(x = date, y = prop_searched)) +
  geom_line() +
  geom_smooth() +
  labs(title = "Proportion of Illinois traffic stops that result in searches", y = "proportion searched
```



Still quite spiky! Interestingly, we see an inverse-∩ trend in the average here (contrast with the raw counts of traffic stops above), but these fluctuations are, again, very small relative to the underlying variation or the range of measure. Also, notice the narrow y-axis scale sort of hides the fact that almost all the values fall between $4 - 6\%$. In other words, all of the variation here is within quite a narrow range.

5.2 Plot stops within `subject_race` categories

To do this, I'll need to go back to the code that created my `ilstops_monthly` tibble above and incorporate the conditional summary data. To do this, I'll "explode" the aggregated summaries from the old tibble by inserting an extra variable to `group_by()` before making my call to `summarize`. Notice that I go ahead and cleanup the dates again too (since I know to anticipate that issue)

```
ilstops_monthly_grouped <- ilstops %>%
  filter(!is.na(subject_race)) %>%
  mutate(
    yearmonth = format(date, "%Y-%m")
  ) %>%
  group_by(yearmonth, subject_race) %>%
  summarize(
    stops = n(),
    searches = sum(as.numeric(search_conducted), na.rm = T),
    prop_searched = round(searches / stops, 3)
  )

ilstops_monthly_grouped$date <- parse_date_time(ilstops_monthly_grouped$yearmonth, "ym")

ilstops_monthly_grouped
```

```
## # A tibble: 360 x 6
```

```
## # Groups:   yearmonth [72]
##    yearmonth subject_race       stops searches prop_searched date
##    <chr>     <fct>              <int>    <dbl>         <dbl> <dttm>
##  1 2012-01   asian/pacific isl~    56        1         0.018 2012-01-01 00:00:00
##  2 2012-01   black                350       33         0.094 2012-01-01 00:00:00
##  3 2012-01   hispanic             254       21         0.083 2012-01-01 00:00:00
##  4 2012-01   other                  7        0         0     2012-01-01 00:00:00
##  5 2012-01   white               1197       39         0.033 2012-01-01 00:00:00
##  6 2012-02   asian/pacific isl~    53        0         0     2012-02-01 00:00:00
##  7 2012-02   black                378       37         0.098 2012-02-01 00:00:00
##  8 2012-02   hispanic             223       26         0.117 2012-02-01 00:00:00
##  9 2012-02   other                  4        0         0     2012-02-01 00:00:00
## 10 2012-02   white               1249       43         0.034 2012-02-01 00:00:00
## # ... with 350 more rows
```

Great! Now, I need to add the values for total stops and total searches each month (note that this will be the same for all five groups/rows within any given value of `date` or `yearmonth`) so that I can calculate the proportion of total stops and total searches accounted for by each group. There are a bunch of ways we could go about this. Since I've made it this far with tidyverse code, I'll keep going in that direction and use the `mutate` verb to keep all my existing rows while also creating some new ones. Notice that I have to do this twice because my first call to `group_by` is only by `yearmonth` so that I can create the monthly totals, whereas my second call to `group_by` once again looks wihtin `yearmonth` and `subject_race` to calculate the proportions. In order to make sure that nothing weird happens along the way I insert a call to `ungroup` as well (which does exactly what you might hope following an operation performed on some data that's already grouped in some way).

```
ilstops_monthly_grouped <- ilstops_monthly_grouped %>%
  group_by(yearmonth) %>%
  mutate(
    total_stops = sum(stops),
    total_searches = sum(searches)
  ) %>%
  ungroup() %>%
  group_by(yearmonth, subject_race) %>%
  mutate(
    prop_total_stops = stops / total_stops,
    prop_total_searches = searches / total_searches
  )

ilstops_monthly_grouped
```

```
## # A tibble: 360 x 10
## # Groups:   yearmonth, subject_race [360]
##    yearmonth subject_race stops searches prop_searched date
##    <chr>     <fct>        <int>    <dbl>         <dbl> <dttm>
##  1 2012-01   asian/pacif~    56        1         0.018 2012-01-01 00:00:00
##  2 2012-01   black          350       33         0.094 2012-01-01 00:00:00
##  3 2012-01   hispanic       254       21         0.083 2012-01-01 00:00:00
##  4 2012-01   other            7        0         0     2012-01-01 00:00:00
##  5 2012-01   white         1197       39         0.033 2012-01-01 00:00:00
##  6 2012-02   asian/pacif~    53        0         0     2012-02-01 00:00:00
##  7 2012-02   black          378       37         0.098 2012-02-01 00:00:00
##  8 2012-02   hispanic       223       26         0.117 2012-02-01 00:00:00
##  9 2012-02   other            4        0         0     2012-02-01 00:00:00
## 10 2012-02   white         1249       43         0.034 2012-02-01 00:00:00
```

```
## # ... with 350 more rows, and 4 more variables: total_stops <int>,
## #   total_searches <dbl>, prop_total_stops <dbl>, prop_total_searches <dbl>
```

```
## I think the html drops a few of our new columns, so I'll inspect them manually here
head(ilstops_monthly_grouped[, c("yearmonth", "subject_race", "total_searches", "prop_total_stops", "pr
```

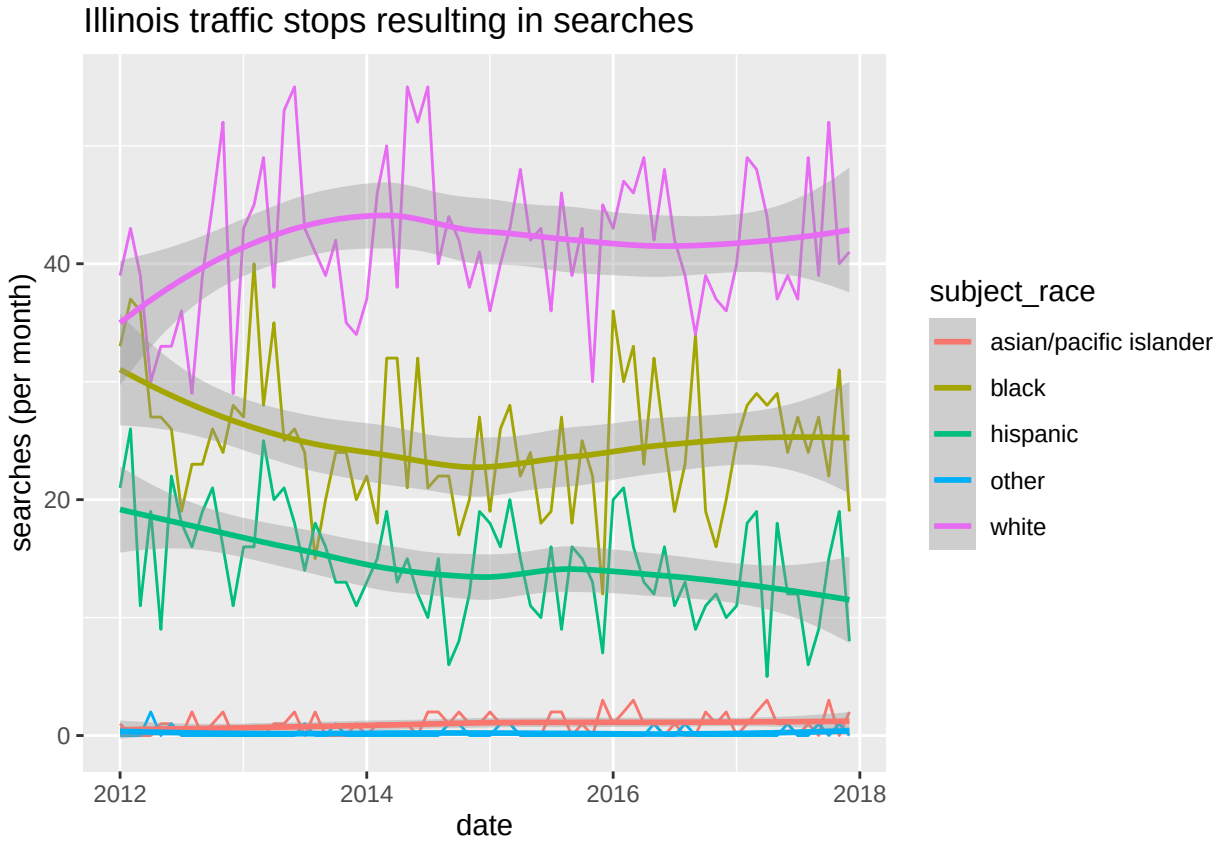
```
## # A tibble: 6 x 5
## # Groups:   yearmonth, subject_race [6]
##   yearmonth subject_race       total_searches prop_total_stops prop_total_searc~
##   <chr>     <fct>                       <dbl>            <dbl>             <dbl>
## 1 2012-01   asian/pacific isl~             94          0.0300            0.0106
## 2 2012-01   black                          94          0.188             0.351
## 3 2012-01   hispanic                       94          0.136             0.223
## 4 2012-01   other                          94          0.00376           0
## 5 2012-01   white                          94          0.642             0.415
## 6 2012-02   asian/pacific isl~            106          0.0278            0
```

Now my plotting code will look pretty similar to the univariate plots above, just with the addition of another aesthetic...

```
ilstops_monthly_grouped %>%
  ggplot(aes(x = date, y = stops, color = subject_race)) +
  geom_line() +
  geom_smooth() +
  labs(title = "Illinois traffic stops", y = "stops (per month)")
```

5.3 Plot searches within `subject_race` categories

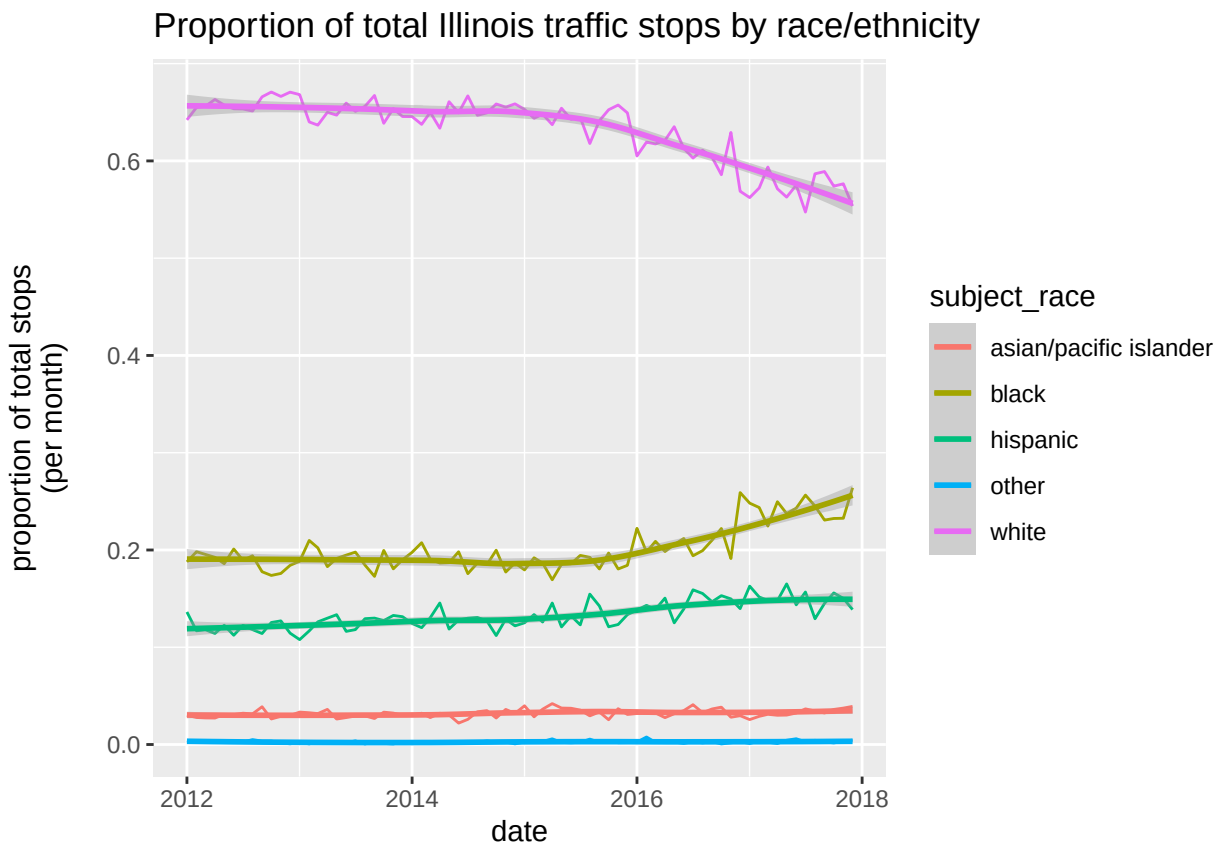
```
ilstops_monthly_grouped %>%
  ggplot(aes(x = date, y = searches, color = subject_race)) +
  geom_line() +
  geom_smooth() +
  labs(title = "Illinois traffic stops resulting in searches", y = "searches (per month)")
```



5.4 Plot proportion of searches within `subject_race` categories

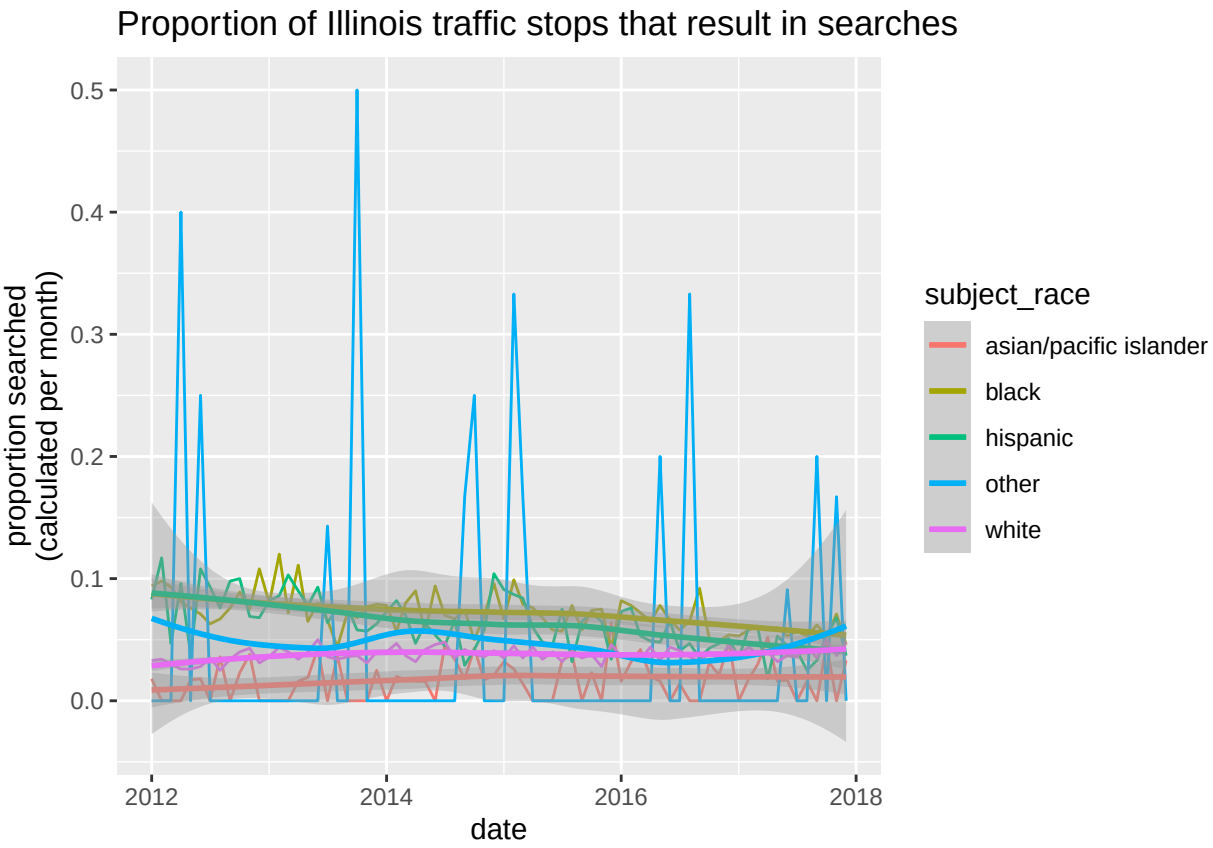
The question for this one only asked about proportions of total searches accounted for by each group of `subject_race`, when several other interesting baselines/comparisons are available. For example, the proportion of total stops across groups as well as stops resulting in searches *within* each group also strike me as potentially illustrative. I'll plot each of these proportions just for the sake of completeness. Let's start with the proportion of total stops plotted across groups.

```
ilstops_monthly_grouped %>%
  ggplot(aes(x = date, y = prop_total_stops, color = subject_race)) +
  geom_line() +
  geom_smooth() +
  labs(title = "Proportion of total Illinois traffic stops by race/ethnicity", y = "proportion of total
```



Onwards to the proportion of stops resulting in searches calculated *within* each group of `subject_race`:

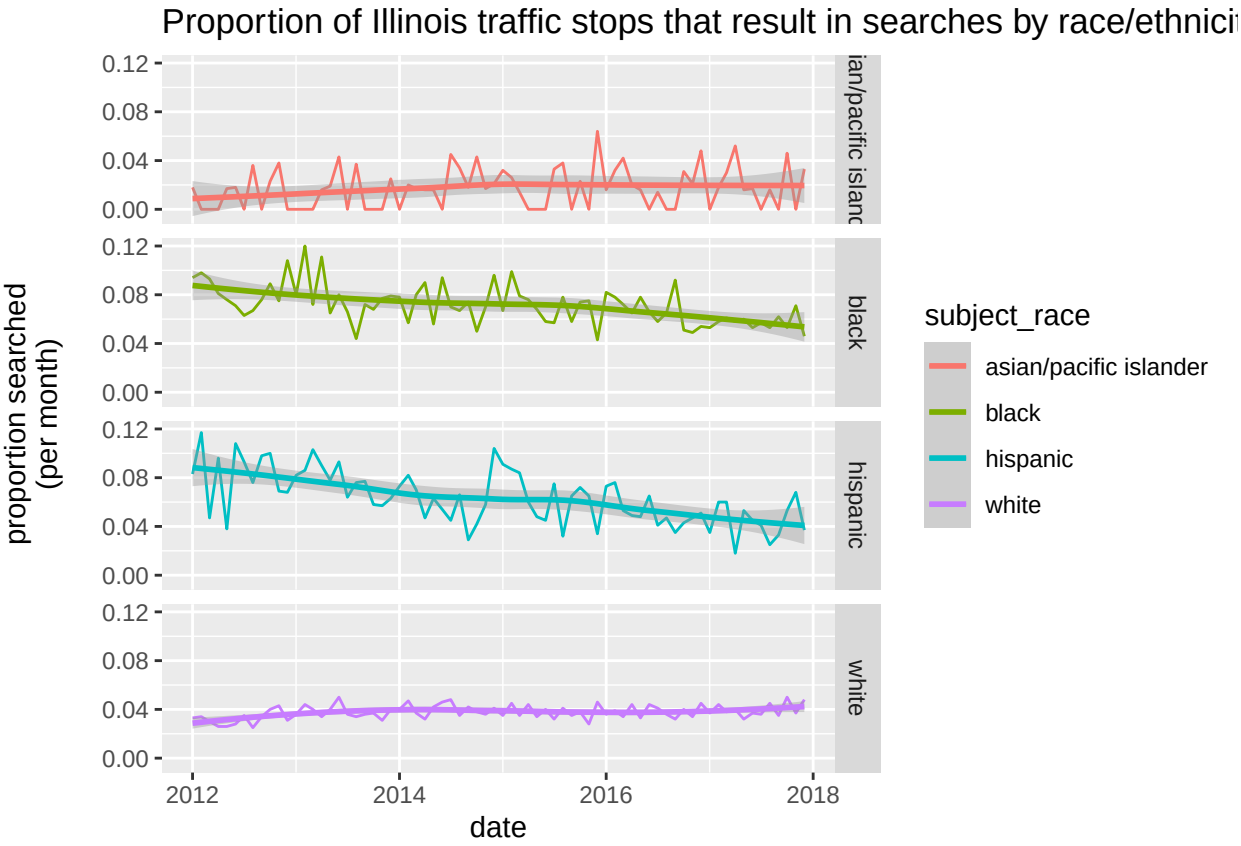
```
ilstops_monthly_grouped %>%
  ggplot(aes(x = date, y = prop_searched, color = subject_race)) +
  geom_line() +
  geom_smooth() +
  labs(title = "Proportion of Illinois traffic stops that result in searches", y = "proportion searched
```



Hmm, that looks pretty messy. The proportions compress everything into a narrow region of the y-axis. Also, the data for some of these groups (especially "other") fluctuates all over the place, which is a distration since that's the smallest category.

Instead of plotting all the proportions on the same figure, it might be easier to see/compare variations if I use facets and don't include the "other" category (because the y-axis scale is just so different).

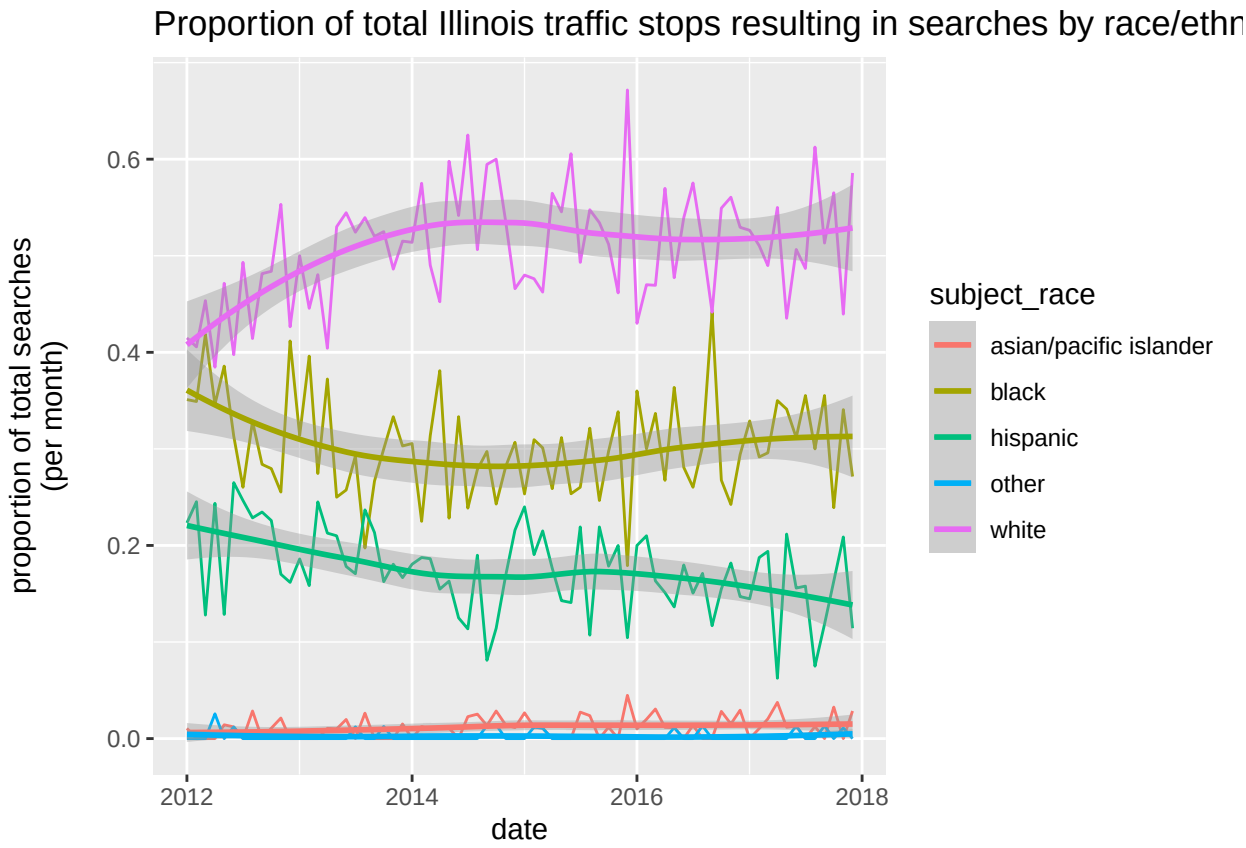
```
ilstops_monthly_grouped %>%
  filter(subject_race != "other") %>%
  ggplot(aes(x = date, y = prop_searched, color = subject_race)) +
  geom_line() +
  geom_smooth() +
  facet_grid(rows = vars(subject_race)) +
  labs(title = "Proportion of Illinois traffic stops that result in searches by race/ethnicity", y = "p
```



The title runs a little long and facet label for "asian/pacific islander" gets truncated. If I were preparing this for publication I'd fix those issues, but for now, I'm going to acknowledge them and move on.

Now, let's look at the proportion of total searches.

```
ilstops_monthly_grouped %>%
  ggplot(aes(x = date, y = prop_total_searches, color = subject_race)) +
  geom_line() +
  geom_smooth() +
  labs(title = "Proportion of total Illinois traffic stops resulting in searches by race/ethnicity", y =
```



PC6. Calculate baseline population proportions

As recommended, I'll calculate this using the `county_complete` provided as part of the `openintro` package to build the relevant variables for Illinois in 2010. You'll want to review the list of variables to figure out what you need to use here.

Note that what we're looking for are *statewide-proportions* of race/ethnic groups in 2010 within the same categories in the SOPP traffic stop data for Illinois. The `county_complete` dataset does not provide these values directly, but instead has county-level populations and county-level proportions of slightly different race/ethnic groups over multiple different years across all the counties in the United States. That means that we'll need to take some steps (like the following) to generate state-level proportion estimates:

- Restrict the dataset to Illinois counties only.
- Multiply county-level populations by county-level proportions for salient groups to create county-level population counts within groups.
- Sum county-level populations within groups to create state-level group populations.
- Divide state-level group populations by total state population (sum of all county populations) to calculate state-level proportions.

In the code chunk below, I'll tackle these steps in a very "verbose" and redundant way:

```
library(openintro)
data(county_complete)

il_county <- county_complete[county_complete$state == "Illinois", ]
```

```r
## total state population is easy
il_pop2010 <- sum(il_county$pop2010)

## Turn IL county-level proportions into counts
il_county$white2010 <- il_county$pop2010 * il_county$white_2010
il_county$black2010 <- il_county$pop2010 * il_county$black_2010
il_county$hispanic2010 <- il_county$pop2010 * il_county$hispanic_2010

## We'll have to sum these two groups to approximate "Asian / Pacific Islander" from SOPP data
il_county$asian2010 <- (il_county$pop2010 * il_county$asian_2010)
il_county$pi2010 <- (il_county$pop2010 * il_county$pac_isl_2010)

## Then go through summing and dividing by the total population
##
## prop.white
sum(il_county$white2010) / il_pop2010
```

```
## [1] 71.53672
```

```r
## prop.black
sum(il_county$black2010) / il_pop2010
```

```
## [1] 14.5509
```

```r
## prop.hispanic
sum(il_county$hispanic2010) / il_pop2010
```

```
## [1] 15.82155
```

```r
## prop.api
(sum(il_county$asian2010, na.rm = TRUE) + sum(il_county$pi2010, na.rm = TRUE)) / il_pop2010
```

```
## [1] 4.576022
```

That is all perfectly accurate; however, the code is verbose and redundant because it calculates each (intermediate and group) value with separate lines of code that are very repetitive. There are many more concise ways to go about this, all of which make it faster, less prone to typing mistakes, and easier to extend. How to proceed? Anytime I find myself repeating the same steps of analysis more than once, I consider creating a function and then calling that function within a call to an `*apply()` function. Here's what that might look like here:

```r
gen_subgroup_prop <- function(group_var, d = il_county) {
  total_pop2010 <- sum(d["pop2010"]) # state population total
  county_ests <- d["pop2010"] * d[group_var] # county-level counts within groups
  group_prop <- sum(county_ests, na.rm = TRUE) / total_pop2010 # state-level proportions within groups
  return(round(group_prop, 4)) # prettify output
}

## give it a try:
gen_subgroup_prop("hispanic_2010")
```

```
## [1] 15.8215
```

```r
## Now let's tie everything together
groups <- c("white_2010", "black_2010", "hispanic_2010", "asian_2010", "pac_isl_2010")

sapply(groups, gen_subgroup_prop)
```

```
##    white_2010    black_2010 hispanic_2010    asian_2010  pac_isl_2010
##       71.5367       14.5509       15.8215        4.5717        0.0043
```

I need to add those last two values together to reproduce the "asian/pacific islander" category from the SOPP data. Note also that there are slight differences between my results here and my results above. Any differences **should** be due entirely to rounding and, in a scenario where this analysis was being prepared for publication and/or informing a policy decision, I'd ansp;ite;y want to check that and think carefully how many significant digits to include in my reported estimates. The Illinois population is sufficiently large that even a few hundredths of a percent turn into very meaningful numbers of people!

Statistical questions

Note that my "solutions" below reflect possible interpretations, but are not intended to be exhaustive or exclusive of other possible interpretations. We'll discuss these in our class meeting.

SQ1. Interpret analysis from PCs3-5

Some interpretation appeared above alongside specific results. Here is a brief summary of several striking points.

- Overall, traffic stops and searches in Illinois between 2012-2018 were distributed unequally across different racial/ethnic groups.
  - The proportion of stops resulting in searches is higher among stops of black and hispanic drivers (7% and 6% respectively vs. 4% or less for all other groups).
  - Black and hispanic drivers account for a higher proportion of stops resulting in searches than stops not resulting in searches (30% and 18% vs. 20% and 13% respectively).
- Over time, the tpyical number of traffic stops, stops resulting in searches, and the proportion of stops resulting in searches fluctuate widely, but have fairly stationary central trends.
  - The typical number of monthly stops dipped a bit between 2014-2015.
  - The typical number of monthly stops resulting in searches may have decreased 2012-2015, but held flat 2015-2018.
  - The typical monthly proportion of stops resulting in searches increased a little 2012-2014, but then declined a little bit 2014-2017.
- Comparing across the racial/ethnic groups identified in the data, the *typical monthly numbers of stops and searches* vary substantially and exhibit distinct trends.
  - The typical number of black and hispanic drivers stopped per month has increased.
  - The typical number of white drivers stopped per month has decreased.
  - The typical number of asian/pacific islander and "other" category drivers per month has remained nearly flat.
  - The typical number of stops resulting in searches among black, white, and hispanic drivers per month have tended to be quite close to each other, and sometimes overlap,
  - The typical number of stops resulting in searches among white drivers per month has increased slightly.
  - The typical number of stops resulting in searches among black and hispanic drivers per month has decreased slightly.
- The typical *proportion of stops resulting in searches within racial/ethnic groups* have shifted somewhat over time.
  - The proportion has increased slightly among white and asian/pacific islander drivers.

  - The proportion has decreased slightly among black and hispanice drivers.
- The typical *proportion of total stops accounted for by each racial/ethnic group* has shifted somewhat over time.
  - The proportion has decreased substantially among white drivers.
  - The proportion has increased among black drivers (substantially) and among hispanic drivers (slightly).
- The typical *proportion of total searches accounted for by each racial/ethnic group* has also shifted over time.
  - As of 2012, the proportion accounted for by black and white drivers was nearly identical (close to 40% each)!
  - The proprtion has increased substantially among white drivers (up to .
  - The proportion has decreased substantially among black and hispanic drivers.

SQ2. Contextualize SQ1 interpretation in relation to PC6

Several noteworthy comparisons come looking across the different proportions for traffic stops and searches and comparing those against the baseline population proportions accounted for by each racial/ethnic group category. Here are several specific observations:

- The white proportion of the state population is consistently larger than either the proportion of white drivers involved in all traffic stops or the proportion of white drivers involved in all stops resulting in searches.

- The hispanic proportion of the state population is about the same as the proportion of hispanic drivers involved in all traffic stops and usually a little smaller than the proportion of hispanic drivers involved in all stops resulting in searches (although this latter difference shrank over the period of data collection).

- The black proportion of the state population is consistently and substantially smaller than the proportion of black drivers involved in all traffic stops and consistently and substantially smaller than the proportion of black drivers involved in all stops resulting in searches. This latter difference in particular stands out as it is almost more than double the baseline population proportion.

- Overall, these results suggest that race/ethnicity and traffic stops and searches are not indepdendent in a statistical sense. The nature of their relationship is complex and could be due to a number of factors including biased policing practices, socioeconomic inequalities, different levels of behaviors linked to traffic stops within specific sub-groups, regional variations masked by these state-level analyses, political/cultural factors shaping police/driver interactions, and more.

SQ3. Limitations and possible extensions

Again, many possible things worth mentioning here, so I'll provide a few that stand out to me.

- The generalizability of analysis focused on one state during one 6 year period is limited.
- Working with a random 1% sample of the full dataset means that our results here could diverge from those we would find in an analysis of the full population of traffic stops in unpredictable ways. That said, even the very small sample is quite big and once you've read *OpenIntro* §5 you'll have some tools to estimate standard errors and confidence intervals around the various results from this analysis.

- The data seem very prone to measurement errors of various kinds. In particular, I suspect the race/ethnicity classifications provided by officers are subject to some biases that are hard to identify and might also shift over time/region. The prevalence of missing values during the first two years of the dataset illustrate one aspect of this and may impact estimates of raw counts and proportions.

- While the comparisons across racial/ethnic groups and between the traffic stops/searches and baseline population proportions illustrates a number of suggestive patterns, conclusive interpretation or attribution of those patterns to any specific cause or causes is quite difficult in the absence of additional information or assumptions. For one example, see my comments regarding statistical independence and the possible explanations in SQ2 above.
- Extensions of this analysis might seek to investigate how some of the patterns identified in the aggregate sate-level data vary across sub-regions (e.g., counties or police districts) or even in comparison to other states.

- Another extension might investigate how specific policy and/or socioeconomic changes may or may not relate to shifting patterns in the data.
- Yet another extension could try to identify some factor that introduces some sudden, uncontrolled (by the drivers or the police) change into the process of traffic stops that could make it possible to identify and estimate the effects of specific explanatory factors. The original SOPP paper is an extraordinary example of this that we can discuss!