Intermediate Tidyverse

Keaton Wilson

8/7/2019

Introduction

The goal of this workshop is to use a real-world data set (police stops in Minneapolis in 2017) to work through a bunch of the Tidyverse's more advanced functions. The workshop is catered to users who have some familiarity with the Tidyverse's main principles (tidy data, the pipe, ggplot2) but want to take their knowledge a step further to do more complex data manipulation, summaries and graphing. It's basically an overview of tools that data scientists end up using frequently.

Learning Objectives

Learners in this workshop will work to:
1. Quickly summarize and gain knowledge about the overview of a new data set
2. Filter their data by columns and by certain conditions
3. Learn how to transform and add new columns to existing data sets
4. Understand powerful grouping techniques to apply transformations and summaries
5. Summarize their data multiple times to gain critical insights
6. Join relational data together
7. Generate plots by piping data directly into ggplot2

Generally, the format will be for me to live code and for all of you to follow along and work on your individual scripts, but there will be a few times during the workshop that I'll pose a challenge. This is the time for you and your table to get together and try and solve a problem.

Data read-in and exploration

The first step is to snag the data we're going to use for this workshop. It's an interesting real-life dataset: a large data set of stop-records for the Minneapolis police department in 2017. Let's get it.

```
#Reading in from my github repo
library(tidyverse)

police_data = read_csv("https://tinyurl.com/y55y2hjk")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Warning: 19110 parsing failures.
##   row            col           expected actual                           file
## 28573 citationIssued 1/0/T/F/TRUE/FALSE     NO 'https://tinyurl.com/y55y2hjk'
## 28577 citationIssued 1/0/T/F/TRUE/FALSE     NO 'https://tinyurl.com/y55y2hjk'
## 28589 citationIssued 1/0/T/F/TRUE/FALSE     NO 'https://tinyurl.com/y55y2hjk'
## 28597 citationIssued 1/0/T/F/TRUE/FALSE     NO 'https://tinyurl.com/y55y2hjk'
## 28599 citationIssued 1/0/T/F/TRUE/FALSE     NO 'https://tinyurl.com/y55y2hjk'
## ..... ............. .................. ...... ..............................
## See problems(...) for more details.
```

```
#Get some parsing failures, but we can scope that out in a sec

#Using glimpse (a superior version of str())
glimpse(police_data)
```

```
## Observations: 51,920
## Variables: 15
## $ X1             <dbl> 6823, 6824, 6825, 6826, 6827, 6828, 6829, 6830,...
## $ idNum          <chr> "17-000003", "17-000007", "17-000073", "17-0000...
## $ date           <dttm> 2017-01-01 00:00:42, 2017-01-01 00:03:07, 2017...
## $ problem        <chr> "suspicious", "suspicious", "traffic", "suspici...
## $ MDC            <chr> "MDC", "MDC", "MDC", "MDC", "MDC", "MDC", "MDC"...
## $ citationIssued <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ personSearch   <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO",...
## $ vehicleSearch  <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO",...
## $ preRace        <chr> "Unknown", "Unknown", "Unknown", "Unknown", "Un...
## $ race           <chr> "Unknown", "Unknown", "White", "East African", ...
## $ gender         <chr> "Unknown", "Male", "Female", "Male", "Female", ...
## $ lat            <dbl> 44.96662, 44.98045, 44.94835, 44.94836, 44.9790...
## $ long           <dbl> -93.24646, -93.27134, -93.27538, -93.28135, -93...
## $ policePrecinct <dbl> 1, 1, 5, 5, 1, 1, 1, 2, 2, 4, 5, 1, 2, 1, 1, 1,...
## $ neighborhood   <chr> "Cedar Riverside", "Downtown West", "Whittier",...
```

```
#Overall this looks good - we can see we've got ~51k observations with 15 variables
#Let's scope out what's going on with those parsing failures....
problems(police_data)
```

```
## # A tibble: 19,110 x 5
##      row col           expected          actual file
##    <int> <chr>         <chr>             <chr>  <chr>
##  1 28573 citationIssu~ 1/0/T/F/TRUE/FAL~ NO     'https://tinyurl.com/y55y2~
##  2 28577 citationIssu~ 1/0/T/F/TRUE/FAL~ NO     'https://tinyurl.com/y55y2~
##  3 28589 citationIssu~ 1/0/T/F/TRUE/FAL~ NO     'https://tinyurl.com/y55y2~
##  4 28597 citationIssu~ 1/0/T/F/TRUE/FAL~ NO     'https://tinyurl.com/y55y2~
##  5 28599 citationIssu~ 1/0/T/F/TRUE/FAL~ NO     'https://tinyurl.com/y55y2~
##  6 28601 citationIssu~ 1/0/T/F/TRUE/FAL~ NO     'https://tinyurl.com/y55y2~
##  7 28604 citationIssu~ 1/0/T/F/TRUE/FAL~ NO     'https://tinyurl.com/y55y2~
##  8 28605 citationIssu~ 1/0/T/F/TRUE/FAL~ NO     'https://tinyurl.com/y55y2~
##  9 28606 citationIssu~ 1/0/T/F/TRUE/FAL~ NO     'https://tinyurl.com/y55y2~
## 10 28607 citationIssu~ 1/0/T/F/TRUE/FAL~ YES    'https://tinyurl.com/y55y2~
## # ... with 19,100 more rows
```

```
#Looks like the culprit is the citation issued column - let's just reimport with
#an increased guess amount to include this
address = "https://tinyurl.com/y55y2hjk"
police_data = read_csv(address, guess_max = 29000)
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
glimpse(police_data)
```

```
## Observations: 51,920
## Variables: 15
## $ X1             <dbl> 6823, 6824, 6825, 6826, 6827, 6828, 6829, 6830,...
## $ idNum          <chr> "17-000003", "17-000007", "17-000073", "17-0000...
## $ date           <dttm> 2017-01-01 00:00:42, 2017-01-01 00:03:07, 2017...
```

```
## $ problem        <chr> "suspicious", "suspicious", "traffic", "suspici...
## $ MDC            <chr> "MDC", "MDC", "MDC", "MDC", "MDC", "MDC", "MDC"...
## $ citationIssued <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ personSearch   <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO",...
## $ vehicleSearch  <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO",...
## $ preRace        <chr> "Unknown", "Unknown", "Unknown", "Unknown", "Un...
## $ race           <chr> "Unknown", "Unknown", "White", "East African", ...
## $ gender         <chr> "Unknown", "Male", "Female", "Male", "Female", ...
## $ lat            <dbl> 44.96662, 44.98045, 44.94835, 44.94836, 44.9790...
## $ long           <dbl> -93.24646, -93.27134, -93.27538, -93.28135, -93...
## $ policePrecinct <dbl> 1, 1, 5, 5, 1, 1, 1, 2, 2, 4, 5, 1, 2, 1, 1, 1,...
## $ neighborhood   <chr> "Cedar Riverside", "Downtown West", "Whittier",...
```

```
#Let's check out that column
unique(police_data$citationIssued)
```

```
## [1] NA    "NO"  "YES"
```

Filtering (and selecting) Data

Ok, so now that we have our data and have given it a cursory glance, I'm sure you can think of a million questions to ask about it, some of which we will get to in a bit. But first, let's go over one of the fundamental tools of tidyverse: filtering your data via the `select()` and `filter()` functions.

`select()` allows you to pull out certain columns of data, while `filter()` allows you select only certain rows (usually based on some set of conditions). Let's start with `select()`.

The first thing we might want to do is clean up our dataframe - it's probably not absolutely necessary to have the X1 column (which is just an ID number and is replicated by the idNum column), also... MDC is not a useful addition.

```
#I usually start non-destructively - note lack of quotations and negative signs
police_data %>%
  select(-MDC, -X1)

#We can also use select to change the order and rename
police_data %>%
  select(citation_issued = citationIssued, lat, lon = long, date)

#And we also have helper functions if you a boatload of columns
police_data %>%
  select(contains("Search"), lat:neighborhood)
```

You can already see the power over base R - I'm not even sure how I would go about coding that same functionality with bracket or dollar-sign indexing... this is easy, fast and powerful.

Let's do some filtering.

Often, you may want to subsections of a dataframe for graphing, analysis or summarization. For instance, it might be useful to only have a subset of the data where be useful to have a portion of the data where citations were issued. This is easy using the `filter()` function.

```
police_data %>%
  filter(citationIssued == "YES")
```

```
## # A tibble: 3,211 x 15
##       X1 idNum date                problem MDC   citationIssued
##    <dbl> <chr> <dttm>              <chr>   <chr> <chr>
```

```
##  1 36603 17-2~ 2017-07-06 13:27:32 suspic~ MDC   YES
##  2 36615 17-2~ 2017-07-06 14:34:37 traffic MDC   YES
##  3 36620 17-2~ 2017-07-06 15:09:29 traffic MDC   YES
##  4 36626 17-2~ 2017-07-06 15:58:31 traffic MDC   YES
##  5 36636 17-2~ 2017-07-06 17:20:22 suspic~ MDC   YES
##  6 36638 17-2~ 2017-07-06 17:27:32 suspic~ MDC   YES
##  7 36659 17-2~ 2017-07-06 18:45:15 traffic MDC   YES
##  8 36665 17-2~ 2017-07-06 19:47:12 traffic MDC   YES
##  9 36677 17-2~ 2017-07-06 21:12:10 suspic~ MDC   YES
## 10 36707 17-2~ 2017-07-06 23:45:35 suspic~ MDC   YES
## # ... with 3,201 more rows, and 9 more variables: personSearch <chr>,
## #   vehicleSearch <chr>, preRace <chr>, race <chr>, gender <chr>,
## #   lat <dbl>, long <dbl>, policePrecinct <dbl>, neighborhood <chr>
```

You can also combine filters using & or |.

```
police_data %>%
  filter(citationIssued == "YES" & personSearch == "YES")
```

```
## # A tibble: 459 x 15
##       X1 idNum date                problem MDC   citationIssued
##    <dbl> <chr> <dttm>              <chr>   <chr> <chr>
##  1 36636 17-2~ 2017-07-06 17:20:22 suspic~ MDC   YES
##  2 36665 17-2~ 2017-07-06 19:47:12 traffic MDC   YES
##  3 36773 17-2~ 2017-07-07 10:55:30 traffic MDC   YES
##  4 36860 17-2~ 2017-07-07 20:17:01 suspic~ MDC   YES
##  5 36927 17-2~ 2017-07-08 02:49:26 traffic MDC   YES
##  6 36956 17-2~ 2017-07-08 14:42:05 suspic~ MDC   YES
##  7 36972 17-2~ 2017-07-08 19:04:43 suspic~ MDC   YES
##  8 37019 17-2~ 2017-07-09 01:06:12 traffic MDC   YES
##  9 37085 17-2~ 2017-07-09 21:22:49 suspic~ MDC   YES
## 10 37168 17-2~ 2017-07-10 12:45:39 traffic MDC   YES
## # ... with 449 more rows, and 9 more variables: personSearch <chr>,
## #   vehicleSearch <chr>, preRace <chr>, race <chr>, gender <chr>,
## #   lat <dbl>, long <dbl>, policePrecinct <dbl>, neighborhood <chr>
```

We can also do more complicated time-based filters with `dplyr` and `lubridate`.
For instance, maybe we just want to filter our data to look at records where citations were issued in Novemebr of 2017.

```
#loading lubridate
library(lubridate)
police_data %>%
  filter(citationIssued == "YES" & lubridate::month(date) == 11)
```

```
## # A tibble: 537 x 15
##       X1 idNum date                problem MDC   citationIssued
##    <dbl> <chr> <dttm>              <chr>   <chr> <chr>
##  1 53626 17-4~ 2017-11-01 09:45:06 suspic~ MDC   YES
##  2 53633 17-4~ 2017-11-01 11:24:15 traffic MDC   YES
##  3 53658 17-4~ 2017-11-01 13:54:31 traffic MDC   YES
##  4 53661 17-4~ 2017-11-01 14:16:20 traffic MDC   YES
##  5 53662 17-4~ 2017-11-01 14:25:11 traffic MDC   YES
##  6 53663 17-4~ 2017-11-01 14:30:05 traffic MDC   YES
##  7 53667 17-4~ 2017-11-01 15:22:02 suspic~ MDC   YES
##  8 53668 17-4~ 2017-11-01 15:40:35 suspic~ MDC   YES
##  9 53669 17-4~ 2017-11-01 15:43:18 traffic MDC   YES
```

```
## 10 53677 17-4~ 2017-11-01 16:12:16 suspic~ MDC   YES
## # ... with 527 more rows, and 9 more variables: personSearch <chr>,
## #   vehicleSearch <chr>, preRace <chr>, race <chr>, gender <chr>,
## #   lat <dbl>, long <dbl>, policePrecinct <dbl>, neighborhood <chr>
```

You can also filter by group_by (which we'll get to in a sec). Some useful functions include:
1. ==, >, >=
2. is.na()
3. !
4. between(), near()

Challenge

Using the police data above work with your group/partner to create a subset of the larger data set of traffic violation stops on black men between 1 and 4 am where a citation was issued. Include the date, problem, citationIssued, race, gender, lat and long columns.

The time problem is tricky if you're not familiar with lubridate (and even if you are) - check out the cheat sheet and google!

```
police_data %>%
  select(date, problem, citationIssued, race, gender, lat, long) %>%
  filter(race == "Black" & gender == "Male" & citationIssued == "YES" & problem == "traffic") %>%
  filter(between(hour(date), 1, 4))
```

```
## # A tibble: 82 x 7
##    date                problem citationIssued race  gender   lat  long
##    <dttm>              <chr>   <chr>          <chr> <chr>  <dbl> <dbl>
##  1 2017-07-08 02:49:26 traffic YES            Black Male    44.9 -93.3
##  2 2017-07-11 02:04:15 traffic YES            Black Male    45.0 -93.3
##  3 2017-07-11 02:46:18 traffic YES            Black Male    45.0 -93.3
##  4 2017-07-14 02:31:29 traffic YES            Black Male    45.0 -93.3
##  5 2017-07-15 01:31:15 traffic YES            Black Male    45.0 -93.3
##  6 2017-07-17 01:34:50 traffic YES            Black Male    45.0 -93.3
##  7 2017-07-17 02:44:55 traffic YES            Black Male    45.0 -93.3
##  8 2017-07-22 01:06:41 traffic YES            Black Male    45.0 -93.3
##  9 2017-07-23 01:25:27 traffic YES            Black Male    45.0 -93.3
## 10 2017-07-27 01:24:26 traffic YES            Black Male    45.0 -93.3
## # ... with 72 more rows
```

Mutations

Mutations are really just mildly-confusing term to describe creating columsn from row-by-row calcuations (either making new columns or replacing old columns with new ones). This is maybe my most-used function from the tidyverse - lots of utility!

I'd say it's generally more applicable with continuous data, of which this data set is lacking, but there is still utility here. Let's say you wanted to add a column that indicates whether or not the cop for a given record was successful at determining the race of the person being stopped. We can add a new column to the data easily!

```
police_data %>%
  select(date, problem, preRace, race, gender, lat, long) %>% #slimming down the data a bit first so we
  mutate(race_correct = ifelse(preRace == race, TRUE, FALSE))
```

```
## # A tibble: 51,920 x 8
##    date                problem preRace race  gender   lat  long
##    <dttm>              <chr>   <chr>   <chr> <chr>  <dbl> <dbl>
##  1 2017-01-01 00:00:42 suspic~ Unknown Unkn~ Unkno~  45.0 -93.2
##  2 2017-01-01 00:03:07 suspic~ Unknown Unkn~ Male    45.0 -93.3
##  3 2017-01-01 00:23:15 traffic Unknown White Female  44.9 -93.3
##  4 2017-01-01 00:33:48 suspic~ Unknown East~ Male    44.9 -93.3
##  5 2017-01-01 00:37:58 traffic Unknown White Female  45.0 -93.3
##  6 2017-01-01 00:46:48 traffic Unknown East~ Male    45.0 -93.3
##  7 2017-01-01 00:48:46 suspic~ Unknown Black Male    45.0 -93.3
##  8 2017-01-01 00:50:55 traffic Unknown Other Female  45.0 -93.2
##  9 2017-01-01 00:57:10 traffic Unknown White Male    45.0 -93.3
## 10 2017-01-01 01:05:50 traffic Unknown Black Male    45.0 -93.3
## # ... with 51,910 more rows, and 1 more variable: race_correct <lgl>
```

And here's another more complicated example that adds an evening, morning or night designator.

```
police_data %>%
  select(date, problem, preRace, race, gender, lat, long, policePrecinct) %>%
  mutate(time_of_day = ifelse(between(hour(date), 5, 12), "Morning",
                              ifelse(between(hour(date), 13, 19), "Afternoon", "Night")))
```

```
## # A tibble: 51,920 x 9
##    date                problem preRace race  gender   lat  long
##    <dttm>              <chr>   <chr>   <chr> <chr>  <dbl> <dbl>
##  1 2017-01-01 00:00:42 suspic~ Unknown Unkn~ Unkno~  45.0 -93.2
##  2 2017-01-01 00:03:07 suspic~ Unknown Unkn~ Male    45.0 -93.3
##  3 2017-01-01 00:23:15 traffic Unknown White Female  44.9 -93.3
##  4 2017-01-01 00:33:48 suspic~ Unknown East~ Male    44.9 -93.3
##  5 2017-01-01 00:37:58 traffic Unknown White Female  45.0 -93.3
##  6 2017-01-01 00:46:48 traffic Unknown East~ Male    45.0 -93.3
##  7 2017-01-01 00:48:46 suspic~ Unknown Black Male    45.0 -93.3
##  8 2017-01-01 00:50:55 traffic Unknown Other Female  45.0 -93.2
##  9 2017-01-01 00:57:10 traffic Unknown White Male    45.0 -93.3
## 10 2017-01-01 01:05:50 traffic Unknown Black Male    45.0 -93.3
## # ... with 51,910 more rows, and 2 more variables: policePrecinct <dbl>,
## #   time_of_day <chr>
```

Lots of options for continuous data, think about the functions:
1. +, -, log() 2. lead(), lag() 3. percent_rank(), ntile() 4. cummean() 5. na_if()

Group Challenge

Replace the race column in the dataset with a column of the same name except that the value "Unknown" is replaced with NA.

```
police_data %>%
  mutate(race = na_if(race, "Unknown"))
```

```
## # A tibble: 51,920 x 15
##       X1 idNum date                problem MDC   citationIssued
##    <dbl> <chr> <dttm>              <chr>   <chr> <chr>
##  1  6823 17-0~ 2017-01-01 00:00:42 suspic~ MDC   <NA>
##  2  6824 17-0~ 2017-01-01 00:03:07 suspic~ MDC   <NA>
##  3  6825 17-0~ 2017-01-01 00:23:15 traffic MDC   <NA>
##  4  6826 17-0~ 2017-01-01 00:33:48 suspic~ MDC   <NA>
```

```
##  5  6827 17-0~ 2017-01-01 00:37:58 traffic MDC   <NA>
##  6  6828 17-0~ 2017-01-01 00:46:48 traffic MDC   <NA>
##  7  6829 17-0~ 2017-01-01 00:48:46 suspic~ MDC   <NA>
##  8  6830 17-0~ 2017-01-01 00:50:55 traffic MDC   <NA>
##  9  6831 17-0~ 2017-01-01 00:57:10 traffic MDC   <NA>
## 10  6832 17-0~ 2017-01-01 01:05:50 traffic MDC   <NA>
## # ... with 51,910 more rows, and 9 more variables: personSearch <chr>,
## #   vehicleSearch <chr>, preRace <chr>, race <chr>, gender <chr>,
## #   lat <dbl>, long <dbl>, policePrecinct <dbl>, neighborhood <chr>
```

Group Operations and Summaries

One of the most powerful features of dplyr is the ability to do stuff on groups or chunks of your data. Sometimes you'll want to leave the dataset intact, and other times you'll want to summarize the data after you do these groupings. Let's look at some examples of both.

First, let's look at some relatively complicated manipulations that leave all the data in place... we're going to calculate the average hour at which stops happen for each month, and then filter the data frame for only incidents that happen after that average.

```
police_data %>%
  group_by(month(date)) %>%
  mutate(mean_hour = mean(hour(date))) %>%
  filter(hour(date) > mean_hour) %>%
  tail()
```

```
## # A tibble: 6 x 17
## # Groups:   month(date) [1]
##      X1 idNum date                problem MDC   citationIssued personSearch
##   <dbl> <chr> <dttm>              <chr>   <chr> <chr>          <chr>
## 1 60833 17-4~ 2017-12-31 23:11:15 suspic~ MDC   NO             NO
## 2 60834 17-4~ 2017-12-31 23:15:50 traffic MDC   YES            NO
## 3 60835 17-4~ 2017-12-31 23:18:32 suspic~ MDC   NO             NO
## 4 60836 17-4~ 2017-12-31 23:31:57 traffic MDC   NO             NO
## 5 60837 17-4~ 2017-12-31 23:48:22 traffic MDC   NO             YES
## 6 60838 17-4~ 2017-12-31 23:52:35 traffic MDC   NO             NO
## # ... with 10 more variables: vehicleSearch <chr>, preRace <chr>,
## #   race <chr>, gender <chr>, lat <dbl>, long <dbl>, policePrecinct <dbl>,
## #   neighborhood <chr>, `month(date)` <dbl>, mean_hour <dbl>
```

Now let's look at a summary - this is where stuff get's really useful. Let's say that you wanted to compare the total number of incidents among white and black men and women.

```
police_data %>%
  filter(race == "Black" | race == "White") %>%
  group_by(race, gender) %>%
  summarize(n = n())
```

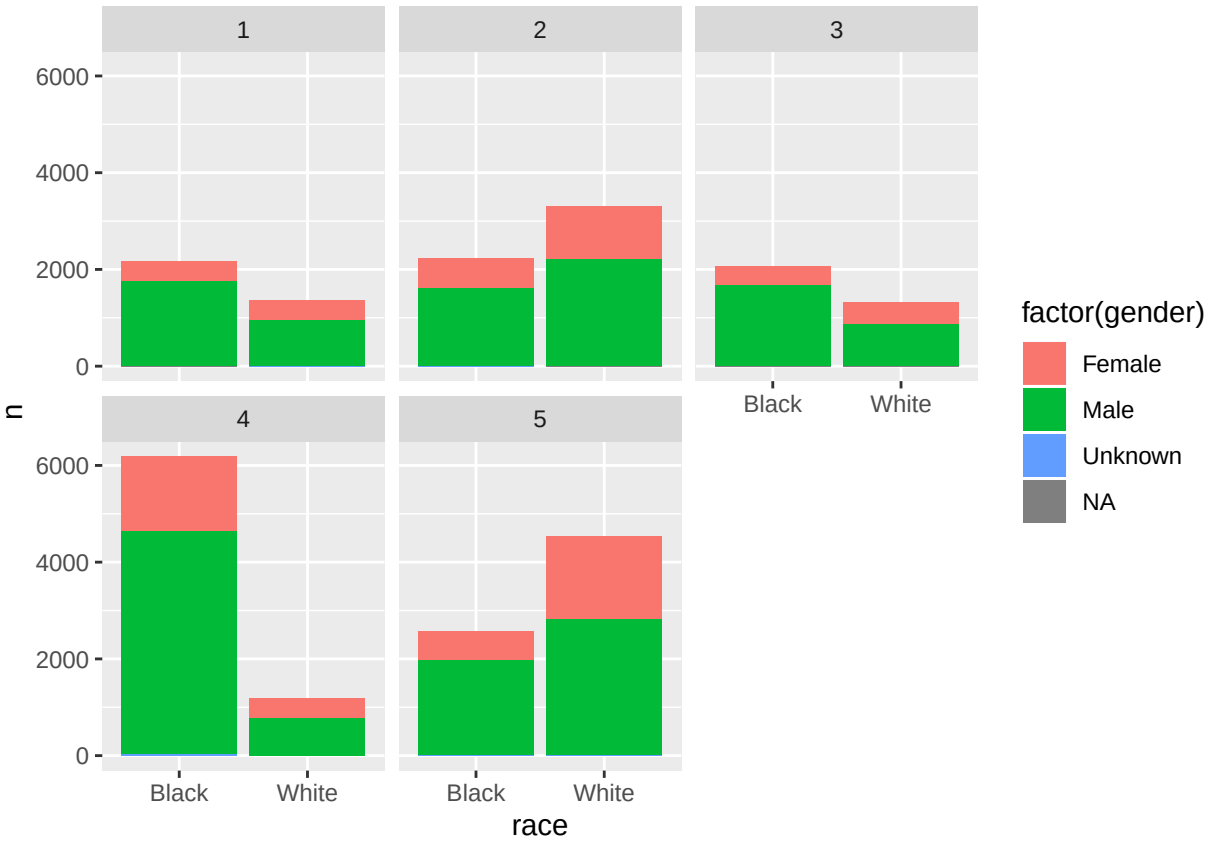
```
## # A tibble: 8 x 3
## # Groups:   race [2]
##   race  gender      n
##   <chr> <chr>   <int>
## 1 Black Female   3510
## 2 Black Male    11630
## 3 Black Unknown    64
## 4 Black <NA>       16
```

```
## 5 White Female   4036
## 6 White Male     7635
## 7 White Unknown    20
## 8 White <NA>       12
```

```
#We could also do the same thing to include precinct to see if there are biases by gender or race by pr
police_data %>%
  filter(race == "Black" | race == "White") %>%
  group_by(race, gender, policePrecinct) %>%
  summarize(n = n())
```

```
## # A tibble: 38 x 4
## # Groups:   race, gender [8]
##    race  gender policePrecinct     n
##    <chr> <chr>           <dbl> <int>
##  1 Black Female              1   396
##  2 Black Female              2   602
##  3 Black Female              3   379
##  4 Black Female              4  1542
##  5 Black Female              5   591
##  6 Black Male                1  1769
##  7 Black Male                2  1619
##  8 Black Male                3  1660
##  9 Black Male                4  4605
## 10 Black Male                5  1977
## # ... with 28 more rows
```

```
#A lot of data, probably better to visualize... we'll get to this later, but all of this can be piped d
police_data %>%
  filter(race == "Black" | race == "White") %>%
  group_by(race, gender, policePrecinct) %>%
  summarize(n = n()) %>%
  ggplot(aes(x = race, fill = factor(gender), y = n)) +
  geom_col() +
  facet_wrap(~ policePrecinct)
```



Let's address another question - which are the top 5 neighborhoods with the most number of incidents?

```
police_data %>%
  group_by(neighborhood) %>%
  summarize(n = n()) %>%
  top_n(n = 5, wt = n) %>%
  arrange(desc(n))
```

```
## # A tibble: 5 x 2
##   neighborhood      n
##   <chr>         <int>
## 1 Downtown West  4409
## 2 Whittier       3328
## 3 Near - North   2256
## 4 Lyndale        2154
## 5 Jordan         2075
```

Summarize is really powerful, and you can do it multiple times (depending on your grouping structure) and use functions like:

1. mean() and median()
2. sd()
3. quantile(), min(), max()
4. first(), last()
5. n_distinct()

Group challenge

For this next challenge, develop code that shows the number of stops for each police precinct for white men and women (sort by precinct).

Bonus challenge - which precinct has the most equal number of arrests between men and women? What is the percentage?

```
#Part 1
police_data %>%
  filter(race == "White") %>%
  filter(gender == "Male" | gender == "Female") %>%
  group_by(policePrecinct, gender) %>%
  summarize(n = n()) %>%
  arrange((policePrecinct))
```

```
## # A tibble: 10 x 3
## # Groups:   policePrecinct [5]
##    policePrecinct gender     n
##             <dbl> <chr>  <int>
##  1              1 Female   404
##  2              1 Male     962
##  3              2 Female  1083
##  4              2 Male    2212
##  5              3 Female   443
##  6              3 Male     863
##  7              4 Female   400
##  8              4 Male     790
##  9              5 Female  1706
## 10              5 Male    2808
```

```
#Part 2
police_data %>%
  filter(race == "White") %>%
  filter(gender == "Male" | gender == "Female") %>%
  group_by(policePrecinct, gender) %>%
  summarize(n = n()) %>%
  arrange((policePrecinct)) %>%
  ungroup() %>%
  group_by(policePrecinct) %>%
  mutate(percent_of_total = n/sum(n)) %>%
  mutate(percent_diff = max(percent_of_total)-min(percent_of_total))
```

```
## # A tibble: 10 x 5
## # Groups:   policePrecinct [5]
##    policePrecinct gender     n percent_of_total percent_diff
##             <dbl> <chr>  <int>            <dbl>        <dbl>
##  1              1 Female   404            0.296        0.408
##  2              1 Male     962            0.704        0.408
##  3              2 Female  1083            0.329        0.343
##  4              2 Male    2212            0.671        0.343
##  5              3 Female   443            0.339        0.322
##  6              3 Male     863            0.661        0.322
##  7              4 Female   400            0.336        0.328
##  8              4 Male     790            0.664        0.328
##  9              5 Female  1706            0.378        0.244
```

```
## 10              5 Male    2808            0.622        0.244
```

Joins

A lot of dplyr syntax comes from SQL - a language designed to interact with databases, which are often just tables that are linked to each other through **key** columns. Imagine a scneario where you have a set of information about users - one table describes how much they've been on a certain website (number of hours total, number of days logged in, etc.), and another describes some of their personal attributes (gender, date of birth, login, username, etc.). You can imagine wanting to join the information from these two tables together to do certain analyses. Being able to join together different tables of data is a frequent occurence, and really, really easy with dplyr syntax. You may be asking yourself why it isn't easier just to have everything in one giant table.... giant tables are slow and bulky. A linked database with many smaller tables is more agile. Most folks don't want all the information all the time! Forwhat it's worth - dplyr can also play nice with databases.... but that's for another lesson. :)

For these examples and exercises, I've split the data into three different tables that are each unified by a different theme: incident information, demographic data and geographic data. This is supposed to mimic something you might see in a real database (or get from one of your colleagues). Let's load and look at them.

```
incident_info = read_csv("./data/police_incident_info.csv", guess_max = 29000)
```

```
## Parsed with column specification:
## cols(
##   idNum = col_character(),
##   date = col_datetime(format = ""),
##   problem = col_character(),
##   citationIssued = col_character(),
##   personSearch = col_character(),
##   vehicleSearch = col_character()
## )
```

```
geography = read_csv("./data/police_geography.csv")
```

```
## Parsed with column specification:
## cols(
##   idNum = col_character(),
##   lat = col_double(),
##   long = col_double(),
##   policePrecinct = col_double(),
##   neighborhood = col_character()
## )
```

```
demographics = read_csv("./data/police_demographics.csv")
```

```
## Parsed with column specification:
## cols(
##   idNum = col_character(),
##   preRace = col_character(),
##   race = col_character(),
##   gender = col_character()
## )
```

```
summary_data = read_csv("./data/police_data_summary.csv")
```

```
## Parsed with column specification:
## cols(
```

```
##   policePrecinct = col_double(),
##   neighborhood = col_character(),
##   n_incidents = col_double()
## )
```

```
#glimpse(incident_info)
#Others too...
```

There are couple main types of joins, depending on which combinations of tables you want:
Mutating Joins

1. Inner join - only takes the matching things from each table (if multiple rows match, returns one row)
2. Left or Right Join - one-sided - keeps stuff from one side

3. Full Join (keep everything!)

Filtering Joins

4. Anti-join - differences between two tables

5. Semi-join - keeps all observations in x that have a match in y (returns only rows from the left table)

```
#Let's do this with pipe syntax - simple join where we're keeping everything that is common between the
new = incident_info %>%
  inner_join(geography, by = "idNum")

glimpse(new)
```

```
## Observations: 51,920
## Variables: 10
## $ idNum          <chr> "17-000003", "17-000007", "17-000073", "17-0000...
## $ date           <dttm> 2017-01-01 00:00:42, 2017-01-01 00:03:07, 2017...
## $ problem        <chr> "suspicious", "suspicious", "traffic", "suspici...
## $ citationIssued <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ personSearch   <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO",...
## $ vehicleSearch  <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO",...
## $ lat            <dbl> 44.96662, 44.98045, 44.94835, 44.94836, 44.9790...
## $ long           <dbl> -93.24646, -93.27134, -93.27538, -93.28135, -93...
## $ policePrecinct <dbl> 1, 1, 5, 5, 1, 1, 1, 2, 2, 4, 5, 1, 2, 1, 1, 1,...
## $ neighborhood   <chr> "Cedar Riverside", "Downtown West", "Whittier",...
```

```
#What happens when we want to join a smaller set of data (some summary data) onto a larger dataframe?
summary =  new %>%
  inner_join(summary_data, by = c("neighborhood", "policePrecinct"))
```

Fair warning - joins can get a lot more complicated than this when you have data that is represented in one table and not another, and it's easy to get confused and lose data. This is just a primer! Check out the great book R for Data Science for more on joins!

Group Challenge

Create a two dataframes using joins. First, one that only contains records for stops during the top 3 busiest hours of stops for each neighborhood. Second, one that contains only records for outside of the 3 busiest hours of stops for each neighborhood.

Hint: look up the `hour()` function from lubridate, and think about how to use `semi_join()` and `anti_join()`.

```r
top_hours = police_data %>%
  mutate(hour = hour(date)) %>%
  group_by(neighborhood, hour) %>%
  summarize(n = n()) %>%
  ungroup() %>%
  group_by(neighborhood) %>%
  top_n(n = 3, wt = n)

#Crimes in top 3 hours by neighborhood
police_data %>%
  mutate(hour = hour(date)) %>%
  semi_join(top_hours)
```

```
## Joining, by = c("neighborhood", "hour")
```

```
## # A tibble: 14,006 x 16
##       X1 idNum date                problem MDC   citationIssued
##    <dbl> <chr> <dttm>              <chr>   <chr> <chr>
##  1  6824 17-0~ 2017-01-01 00:03:07 suspic~ MDC   <NA>
##  2  6827 17-0~ 2017-01-01 00:37:58 traffic MDC   <NA>
##  3  6828 17-0~ 2017-01-01 00:46:48 traffic MDC   <NA>
##  4  6829 17-0~ 2017-01-01 00:48:46 suspic~ MDC   <NA>
##  5  6830 17-0~ 2017-01-01 00:50:55 traffic MDC   <NA>
##  6  6831 17-0~ 2017-01-01 00:57:10 traffic MDC   <NA>
##  7  6835 17-0~ 2017-01-01 01:19:59 traffic MDC   <NA>
##  8  6836 17-0~ 2017-01-01 01:22:01 traffic MDC   <NA>
##  9  6837 17-0~ 2017-01-01 01:41:24 suspic~ MDC   <NA>
## 10  6838 17-0~ 2017-01-01 01:45:09 suspic~ MDC   <NA>
## # ... with 13,996 more rows, and 10 more variables: personSearch <chr>,
## #   vehicleSearch <chr>, preRace <chr>, race <chr>, gender <chr>,
## #   lat <dbl>, long <dbl>, policePrecinct <dbl>, neighborhood <chr>,
## #   hour <int>
```

```r
#crimes NOT in top 3 hours by neighborhood
police_data %>%
  mutate(hour = hour(date)) %>%
  anti_join(top_hours)
```

```
## Joining, by = c("neighborhood", "hour")
```

```
## # A tibble: 37,914 x 16
##       X1 idNum date                problem MDC   citationIssued
##    <dbl> <chr> <dttm>              <chr>   <chr> <chr>
##  1  6823 17-0~ 2017-01-01 00:00:42 suspic~ MDC   <NA>
##  2  6825 17-0~ 2017-01-01 00:23:15 traffic MDC   <NA>
##  3  6826 17-0~ 2017-01-01 00:33:48 suspic~ MDC   <NA>
##  4  6832 17-0~ 2017-01-01 01:05:50 traffic MDC   <NA>
##  5  6833 17-0~ 2017-01-01 01:09:13 suspic~ MDC   <NA>
##  6  6834 17-0~ 2017-01-01 01:15:01 traffic other <NA>
##  7  6839 17-0~ 2017-01-01 01:51:02 suspic~ MDC   <NA>
##  8  6841 17-0~ 2017-01-01 01:53:42 traffic MDC   <NA>
##  9  6842 17-0~ 2017-01-01 02:04:13 suspic~ MDC   <NA>
## 10  6843 17-0~ 2017-01-01 02:09:42 traffic MDC   <NA>
## # ... with 37,904 more rows, and 10 more variables: personSearch <chr>,
## #   vehicleSearch <chr>, preRace <chr>, race <chr>, gender <chr>,
```

```
## #   lat <dbl>, long <dbl>, policePrecinct <dbl>, neighborhood <chr>,
## #   hour <int>
```

Piping into ggplot (and some fun plotting things)

The other great thing about dplyr and tidyverse as a whole is that you can pipe modified dataframes directly into ggplot2, and you can also do piped dplyr operations **within** ggplot2.

For example, let's say that we wanted to do some mapping (we have lat lon info for all the stops, so this makes sense!).

```
library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

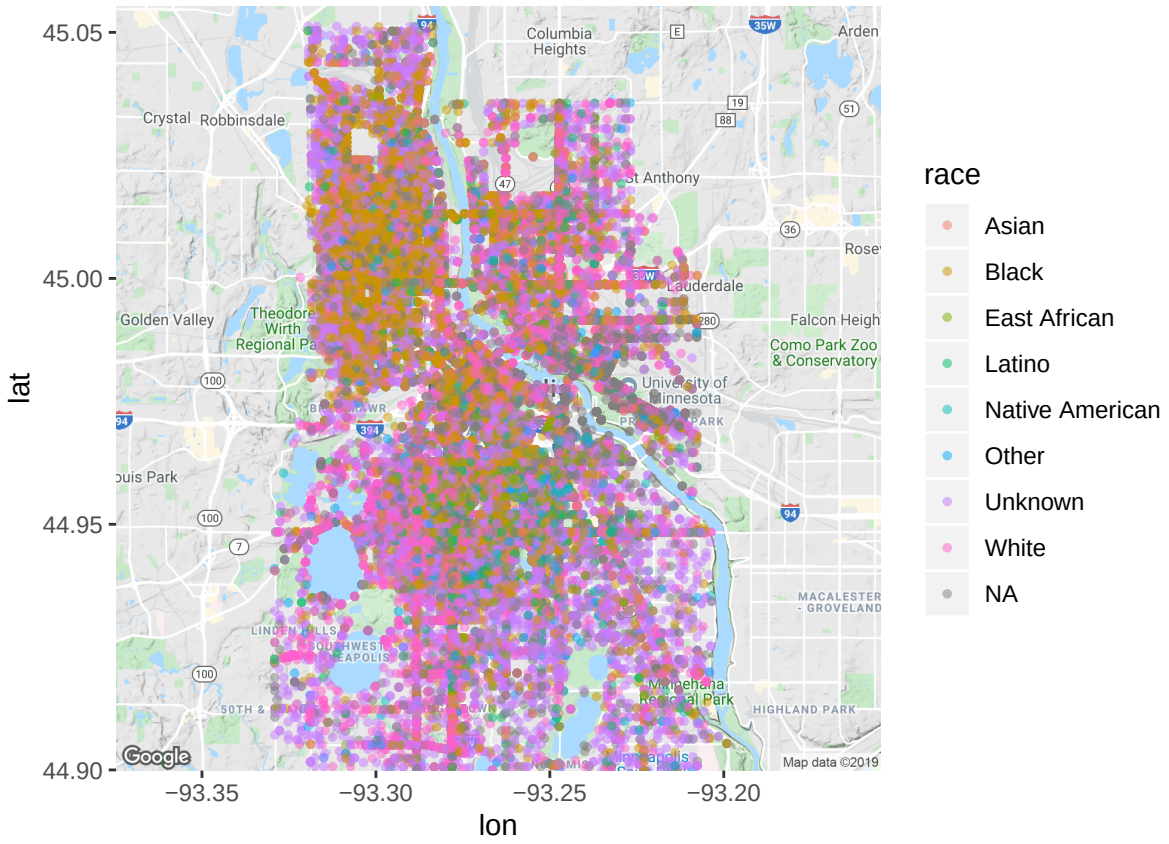
```
register_google(key = "AIzaSyDyAqUc4o9p_DOBSF_JOXH5c_JXPqoU4Yw")
loc = c(police_data$lat[1], police_data$long[1])
our_map = get_map(location = "Minneapolis", zoom = 12)
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=Minneapolis&zoom=12&size=640x640&scal
```

```
## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Minneapolis&key=xxx
```

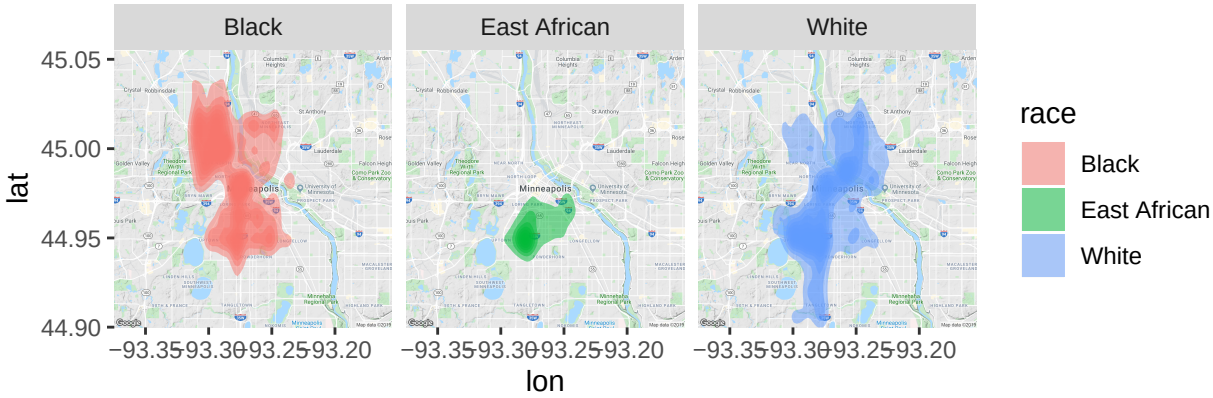
```
#Tooooo much data
ggmap(our_map) +
  geom_point(data = police_data, aes(x = long, y = lat, color = race),
             size = 1, alpha = 0.5)
```

```
## Warning: Removed 560 rows containing missing values (geom_point).
```




```
#Let's do filtering within ggplot
ggmap(our_map) +
  stat_density_2d(data = police_data %>%
                    filter(race == "Black" | race == "White" | race == "East African"),
                  aes(x = long, y = lat, color = NULL, fill = race),
                  geom = "polygon",
                  alpha = 0.5) +
  facet_wrap(~ race)
```

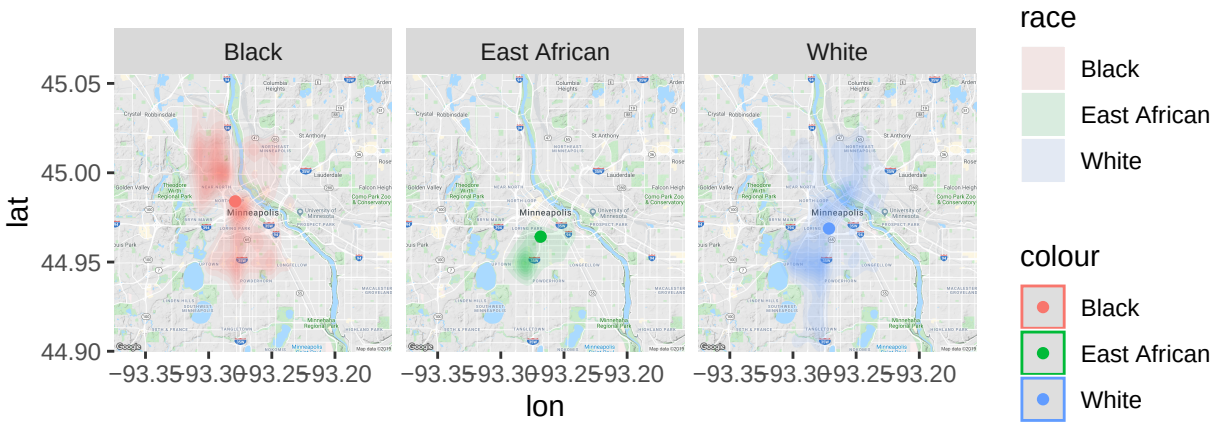
```
## Warning: Removed 233 rows containing non-finite values (stat_density2d).
```




```
#Could also add centroids
ggmap(our_map) +
  stat_density_2d(data = police_data %>%
                    filter(race == "Black" | race == "White" | race == "East African"),
```

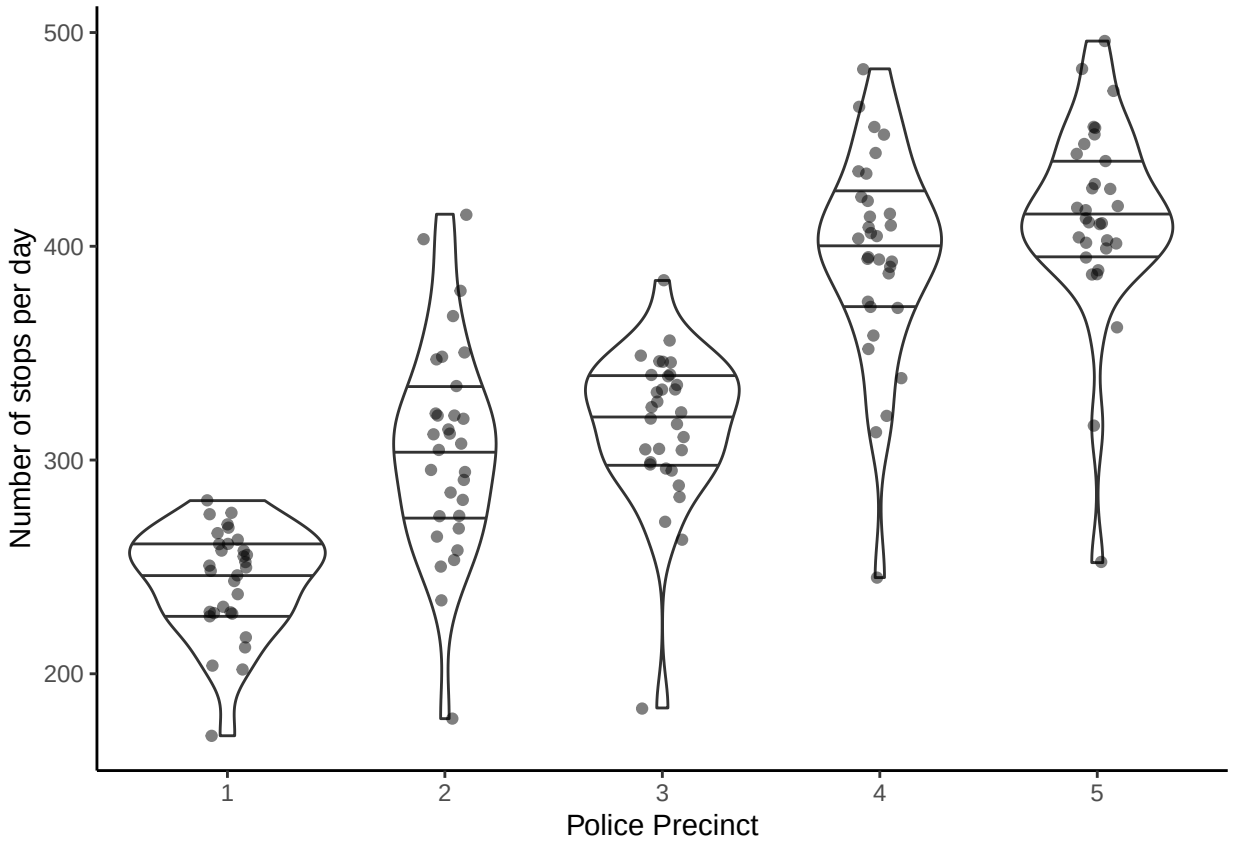
```
            aes(x = long, y = lat, color = NULL, fill = race),
            geom = "polygon",
            alpha = 0.1) +
  geom_point(data = police_data %>%
                    filter(race == "Black" | race == "White" | race == "East African") %>%
                    group_by(race) %>%
                    summarize(avg_lat = mean(lat),
                              avg_lon = mean(long)),
            aes(x = avg_lon, y = avg_lat, color = race)) +
  facet_wrap(~ race)
```

```
## Warning: Removed 233 rows containing non-finite values (stat_density2d).
```




```
#And we can also pipe directly into ggplot (though we can't do this for ggmap)

police_data %>%
  mutate(day = day(date)) %>%
  group_by(policePrecinct, day) %>%
  summarize(n = n()) %>%
  ggplot(aes(x = factor(policePrecinct), y = n)) +
    geom_violin(draw_quantiles = c(0.25, 0.5, 0.75)) +
    geom_jitter(width = 0.1, alpha = 0.5) +
    # geom_dotplot(binaxis = "y", stackdir = 'center', dotsize = 0.5) +
  theme_classic() +
  xlab("Police Precinct") +
  ylab("Number of stops per day")
```



Final group challenge

Put it all together now! Determine which three precincts had the worst racial mis-match (when preRace didn't line up with actual race). Plot a map of all the stops that involved racial mis-matches for the worst precinct, colored by the actual race of the individual stopped.

```
police_data %>%
  filter(preRace != "Unknown" & race != "Unknown") %>%
  mutate(mismatch = ifelse(preRace == race, FALSE, TRUE)) %>%
  group_by(policePrecinct) %>%
  count(mismatch) %>%
  mutate(perc = n/sum(n)) %>%
  filter(mismatch == TRUE) %>%
  arrange(desc(perc))
```

```
## # A tibble: 5 x 4
## # Groups:   policePrecinct [5]
##   policePrecinct mismatch     n   perc
##            <dbl> <lgl>    <int>  <dbl>
## 1              5 TRUE       338 0.0801
## 2              2 TRUE       106 0.0484
## 3              4 TRUE        78 0.0308
## 4              3 TRUE        61 0.0283
## 5              1 TRUE        36 0.0172
```

```
to_map = police_data %>%
  filter(policePrecinct %in% c(5)) %>%
```

```
  mutate(mismatch = ifelse(preRace == race, FALSE, TRUE)) %>%
  filter(mismatch == TRUE)


mismatch_map = get_map(location = "Minneapolis", zoom = 11)
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=Minneapolis&zoom=11&size=640x640&scal
```

```
## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Minneapolis&key=xxx
```

```
ggmap(mismatch_map) +
  geom_point(data = to_map, aes(x = long, y = lat, color = race), alpha = 0.5, size = 1) +
  facet_wrap(~ race)
```