What Does Data 'See'? Public Data, Bias, and Surveillance in NYC

Hannah Pullen-Blasnik

QUESTIONS

This workshop will teach you how to access and think through the biases in public data. How do we use data critically to understand what it can show, where its limitations are, and how it may not necessarily be as neutral or objective as it is often perceived to be? You will:

1. Understand how social scientists might use quantitative data
2. Gain familiarity with public data on New York City using R
3. Think about data's potential biases
4. Explore how researchers and organizations push back against urban surveillance with data

For the coding portion of this session, we will walk through downloading and exploring data on the demographics of NYC, policing stops, and the location of surveillance cameras as provided by the Decode Surveillance project at Amnesty International: https://banthescan.amnesty.org/decode/

The coding portion assumes some prior familiarity with coding in R. If you do not have experience in R, you may want to work with others that do for the coding portion of this presentation.

This session is led by Hannah Pullen-Blasnik, PhD Candidate in Sociology at Columbia University.

Please send any questions or comments to Hannah Pullen-Blasnik at hannah.pullen-blasnik@columbia.edu.

Coding Activity: Census Data, Policing Data, and Surveillance

Importing Libraries We will use packages like tidyverse for organizing our data, ggplot2 for creating charts, and sf for mapping.

```
# install.packages("tidyverse", "ggplot2")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.2     v purrr   1.0.2
## v tibble  3.2.1     v dplyr   1.1.4
## v tidyr   1.3.1     v stringr 1.5.1
## v readr   2.1.3     v forcats 0.5.2
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
## Warning: package 'stringr' was built under R version 4.2.3
```

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

```
library(ggplot2)
library(sf)
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```
library(RSocrata)
library(tidycensus)
library(units)
```

```
## udunits database from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/units/shar
```

```
options(scipen=999)
```

Census Data - City Demographics

Data Download - Using APIs

We'll download data from the Census Bureau to get demographic information about New York City.

To do so, we'll use the package tidycensus and a Census API key. APIs are a common way to access publicly available data. If you don't have a Census API key, you'll have to create one: https://api.census.gov/data/key_signup.html

If you cannot or do not want to create an API key, you can instead skip down to the visualizing census data section and load the acs_clean.csv, skipping the API call and cleaning steps below.

```
### Uncomment and run this line if you just got a new API key:
# census_api_key("YOUR API KEY HERE", install = T)
```

```
# If you already have an API key installed, all you need to run is:
census_api_key(Sys.getenv("CENSUS_API_KEY"))
```

```
## To install your API key for use in future sessions, run this function with `install = TRUE`.
```

Now that we have our credentials we can look at what variables are available. There are many more variables available from the census website, but not all are made available through this R package. For today, we'll stick to a few that we have easy access to. For more exploration of Census data, visit their website: https://data.census.gov/

We're going to download 5-year ACS (American Community Survey) data for 2019 (2015-2019) at the tract level. To view what variables are available, use the load_variables() function from tidycensus.

```
v19 <- load_variables(2019, "acs5", cache = TRUE)
```

We're interested in ACS data from NY state for the 5 counties in NYC ("061", "047", "081", "005", "085"). We want:

- Total population
- Racial composition (for white, Black, Latino, Asian)
  - Note that there are many different racial groupings provided by the census. We need the one that includes whether they are Hispanic or Latino
- Gender composition (male, female)
- Median household income

```
acs <- get_acs(geography="tract",
               state="NY",
               county=c("061", "047", "081", "005", "085"),
               year=2019,
               variables=c(med_inc="B19013_001",
                           total_pop='B01003_001',
                           white="B03002_003",
                           black="B03002_004",
                           asian="B03002_006",
                           latino="B03002_012",
                           male='B01001_002',
                           female='B01001_026'))
```

```
## Getting data from the 2015-2019 5-year ACS
```

Note: How do the categories provided by the census constrain our analyses? Who might be getting left out or miscategorized?

If you did not get an API key, you can instead load the data from the following CSV:

```
acs <- read_csv("data/acs.csv") %>% mutate(GEOID=as.character(GEOID))
```

```
## Rows: 17336 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): NAME, variable
## dbl (3): GEOID, estimate, moe
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Data Exploration

Once we have the data downloaded, we can begin to explore its contents. Start by viewing the table contents with head()

```
head(acs)
```

```
## # A tibble: 6 x 5
##   GEOID       NAME                                      variable  estimate   moe
##   <chr>       <chr>                                     <chr>        <dbl> <dbl>
## 1 36005000100 Census Tract 1, Bronx County, New York male          6345   322
## 2 36005000100 Census Tract 1, Bronx County, New York female         519   182
## 3 36005000100 Census Tract 1, Bronx County, New York total_pop     6864   331
```

```
## 4 36005000100 Census Tract 1, Bronx County, New York white          603   172
## 5 36005000100 Census Tract 1, Bronx County, New York black         3601   296
## 6 36005000100 Census Tract 1, Bronx County, New York asian          181    70
```

We can see that what we get from this API lists all the variables in one column, "variable", and then for each provides the estimated value (estimate) and the margin of error for that esimate (moe).

For our purposes, we'll be working with the estimated value column, but it's good to note that these are estimated values and so have some uncertainty in the value. We can recognize this from the name – the American Community Survey – and know that this data is survey responses that represent the total population. This is one of the most comprehensive surveys in the country.

This format is not the best for what we want to do with the data. We'd like to pivot the table so that each row is one census tract, and there are columns for all the values. Use pivot_wider() with GEOID and NAME as id_cols, taking the names from the variable column and the values from the estimate column. Since we're not going to look at the margin of error in this analysis, we can ignore it.

```
acs_clean <- acs %>%
  pivot_wider(id_cols=c("GEOID", "NAME"), names_from="variable", values_from="estimate")
```

Bonus: We can also use mutate() to create some new columns:

- estimate the population that would fall into an "other race" category by subtracting the sum of the race columns (black, white, latino, and asian) from total_pop
- pull out the county/borough from NAME using str_detect() and a case_when() statement. We want to look for where NAME contains "Bronx", "Kings", "Queens", "New York County", and "Richmond" and code them to "Bronx", "Brooklyn", "Queens", "Manhattan", and "Staten Island", respectively
- determine the most prevalent racial group for each census tract. This one is a little tricky and there are a few ways to do this. One is to identify which of the race columns has the largest value using pmax() and then use a case_when() to identify which of the columns is equal to the maximum value for that row

```
acs_clean <- acs_clean %>%
  mutate(
    otherrace=total_pop-(black+white+asian+latino),
    county=case_when(
      str_detect(NAME, "Bronx") ~ "Bronx",
      str_detect(NAME, "Kings") ~ "Brooklyn",
      str_detect(NAME, "Queens") ~ "Queens",
      str_detect(NAME, "New York County") ~ "Manhattan",
      str_detect(NAME, "Richmond") ~ "Staten Island",
      T ~ "Error"
    ),
    max_race=pmax(black, latino, white, asian, otherrace),
    tract_race=case_when(
           max_race==black ~ "Black",
           max_race==white ~ "White",
           max_race==latino ~ "Latino",
           max_race==asian ~ "Asian",
           T ~ "Other")
  ) %>%
  select(-max_race)

head(acs_clean)
```

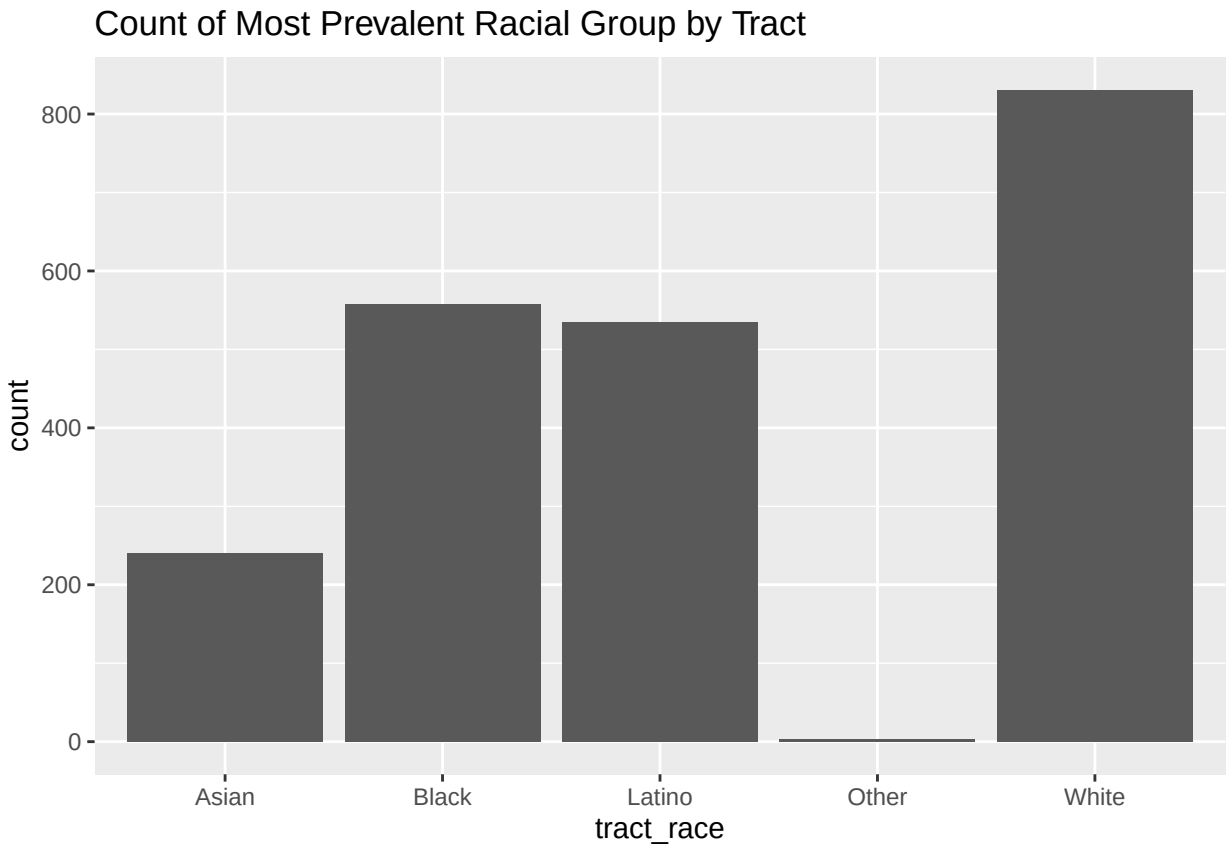
```
## # A tibble: 6 x 13
##   GEOID  NAME   male female total_pop white black asian latino med_inc otherrace
##   <chr>  <chr> <dbl>  <dbl>     <dbl> <dbl> <dbl> <dbl>  <dbl>   <dbl>     <dbl>
## 1 36005~ Cens~  6345    519      6864   603  3601   181   2407      NA        72
## 2 36005~ Cens~  2193   2339      4532    33   920    75   3444   51100        60
## 3 36005~ Cens~  2810   2706      5516   387  1264   121   3661   78409        83
## 4 36005~ Cens~  2422   3403      5825   213  2021    64   3452   34093        75
## 5 36005~ Cens~  1583   1558      3141   340   937    72   1714   45156        78
## 6 36005~ Cens~  4128   4912      9040    21  3031     0   5912   20592        76
## # i 2 more variables: county <chr>, tract_race <chr>
```

Demographic Visuals

Now that we've got our demographic data into a more usable format, let's create some visuals.

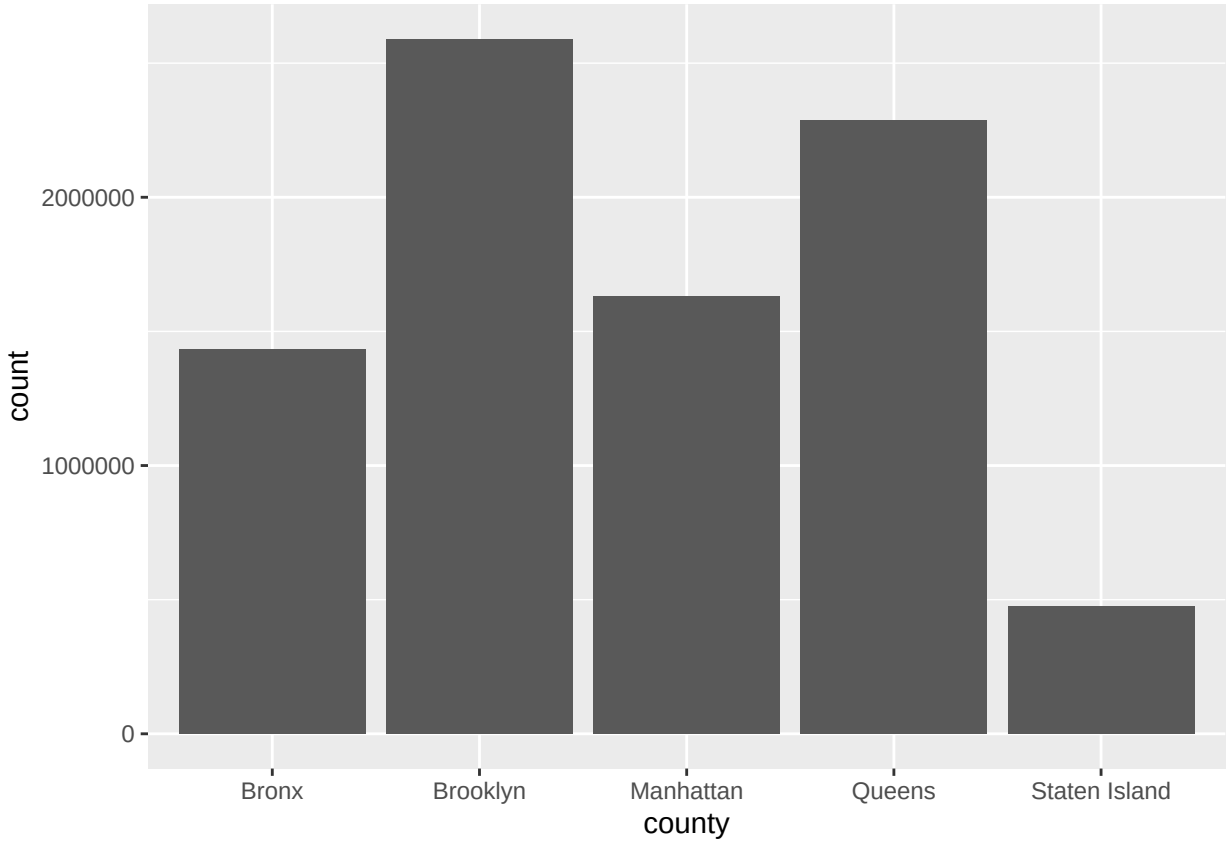
Using ggplot() on our dataframe, create a bar graph (geom_bar()) of the most prevalent racial groups

```
ggplot(acs_clean) +
  geom_bar(aes(x=tract_race)) +
  ggtitle("Count of Most Prevalent Racial Group by Tract")
```



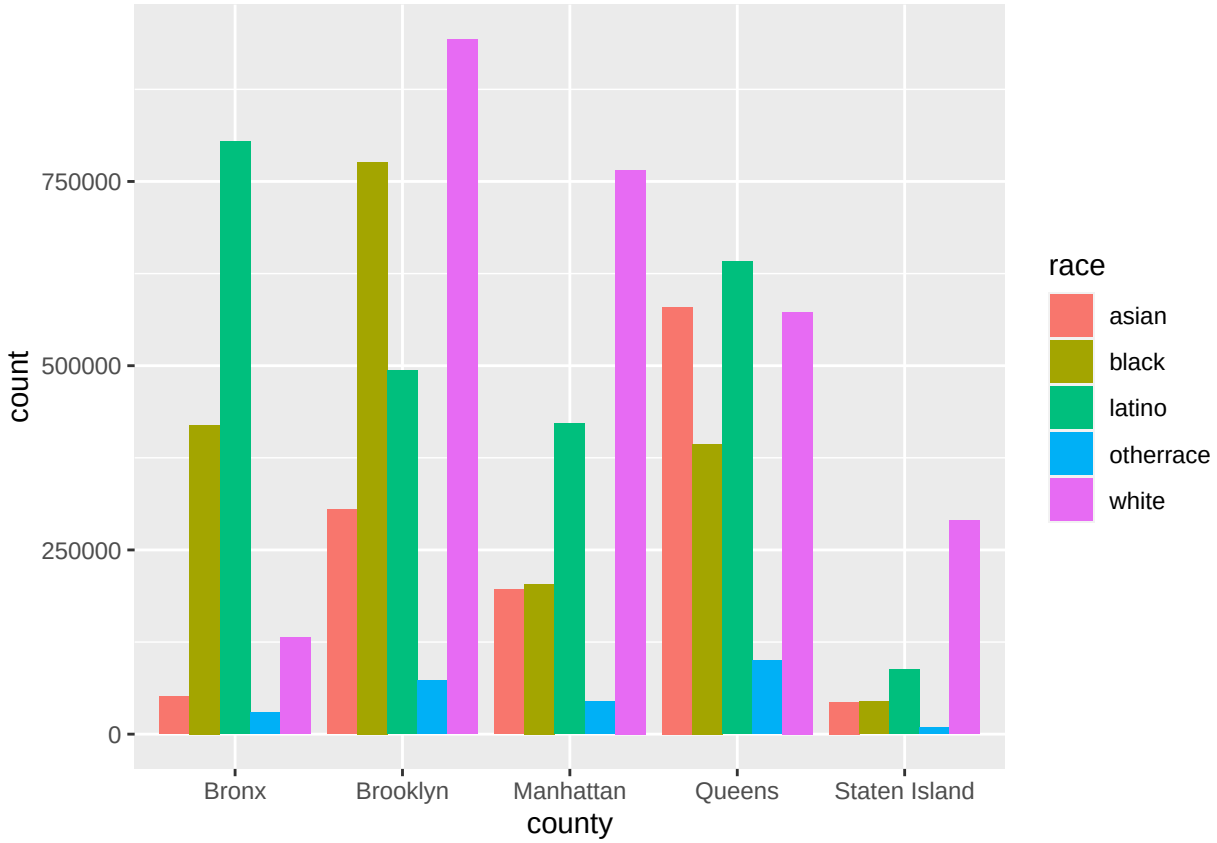
We can also make a bar graph of the population in each county. Since we want to count the people this time, instead of the census tracts, we need to set weight=total_pop

```r
ggplot(acs_clean) +
  geom_bar(aes(x=county, weight=total_pop))
```



Bonus: Graph the population by race by county

```r
acs_clean %>%
  pivot_longer(c("white", "black", "asian", "latino", "otherrace"), names_to="race", values_to="race_pop
  ggplot() +
  geom_bar(aes(x=county, fill=race, weight=race_pop), position="dodge")
```



Geographic Data

Another way we might want to visualize our data is on a map of the city. In order to do so, we need to download a file that tells R where the census tracts are located geographically. You can download this data from the Census website, and often you've been able to use the tigris package in R to download similarly to how we did for the ACS data. However, recently that API has gone down. I've provided the relevant tract shapefile in the data/ folder, so we have load it in using read_sf()

```
# Tigris is currently not working, but usually this would also pull in the tracts:
# library(tigris)
# options(tigris_use_cache = TRUE)

# nyc_tracts <- tracts(state="NY", county=c("061", "047", "081", "005", "085"), year=2019)
```

```
nyc_tracts <- read_sf( "data/tracts.shp")

head(nyc_tracts)
```

```
## Simple feature collection with 6 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 1007285 ymin: 238802.2 xmax: 1034600 ymax: 270146.9
## Projected CRS: NAD83(HARN) / New York Long Island (ftUS)
## # A tibble: 6 x 15
```

```
##   fips  STATEFP COUNTYFP TRACTCE GEOID      NAME  NAMELSAD MTFCC FUNCSTAT  ALAND
##   <chr> <chr>   <chr>    <chr>   <chr>      <chr> <chr>    <chr> <chr>     <dbl>
## 1 36005 36      005      034500  360050345~ 345   Census ~ G5020 S        274627
## 2 36005 36      005      030701  360050307~ 307.~ Census ~ G5020 S        396736
## 3 36005 36      005      031000  360050310~ 310   Census ~ G5020 S        634013
## 4 36005 36      005      014100  360050141~ 141   Census ~ G5020 S        177165
## 5 36005 36      005      027401  360050274~ 274.~ Census ~ G5020 S        701116
## 6 36005 36      005      035000  360050350~ 350   Census ~ G5020 S        191626
## # i 5 more variables: AWATER <dbl>, INTPTLAT <chr>, INTPTLON <chr>,
## #   borough <chr>, geometry <MULTIPOLYGON [US_survey_foot]>
```

Once we have our spatial data, we can use a left_join to add our ACS demographic data based on the GEOID.

```
acs_tracts <- nyc_tracts %>%
  left_join(acs_clean, by="GEOID")
```

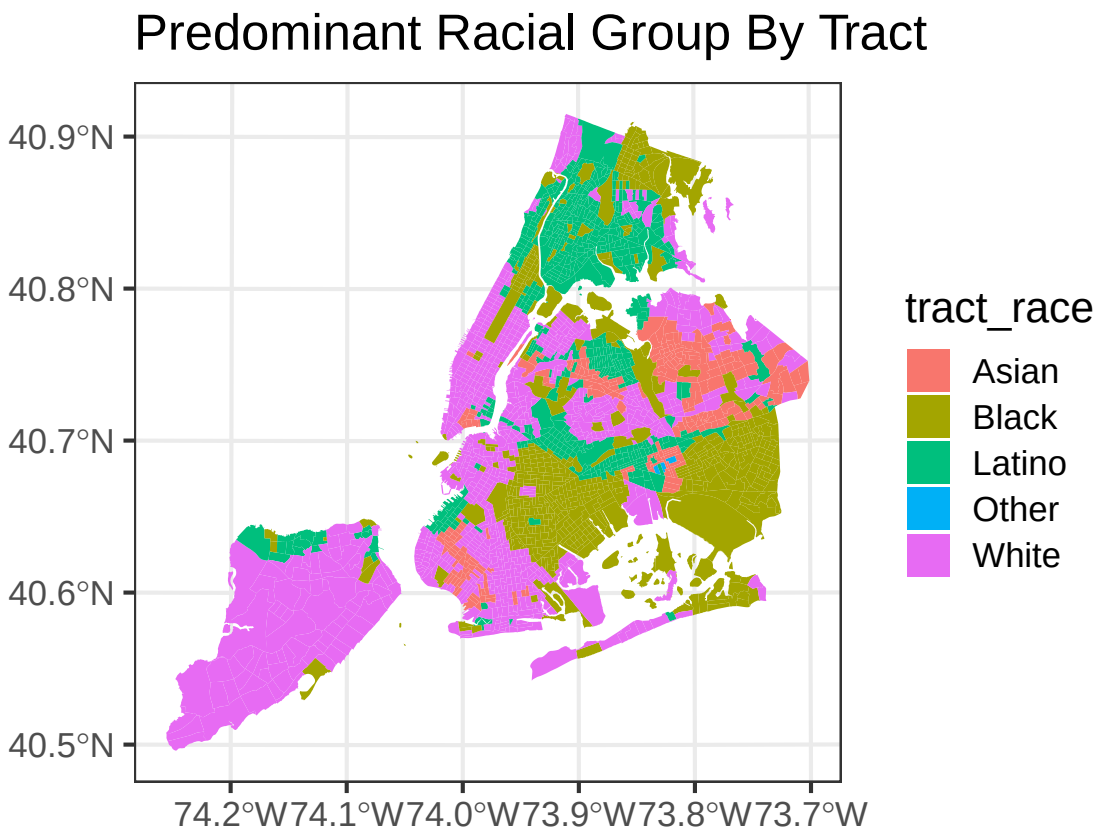
Mapping Demographics

We're going to map our data using ggplot() and geom_sf().

Let's look at racial composition of the city. We'll use that most prevalent race column that we created earlier to color the tracts based on which racial group is the most prevalent in that area.

Bonus: The tract boundaries are outlined by default, but it can look a little murky. Set color=NA in geom_sf() to remove them.

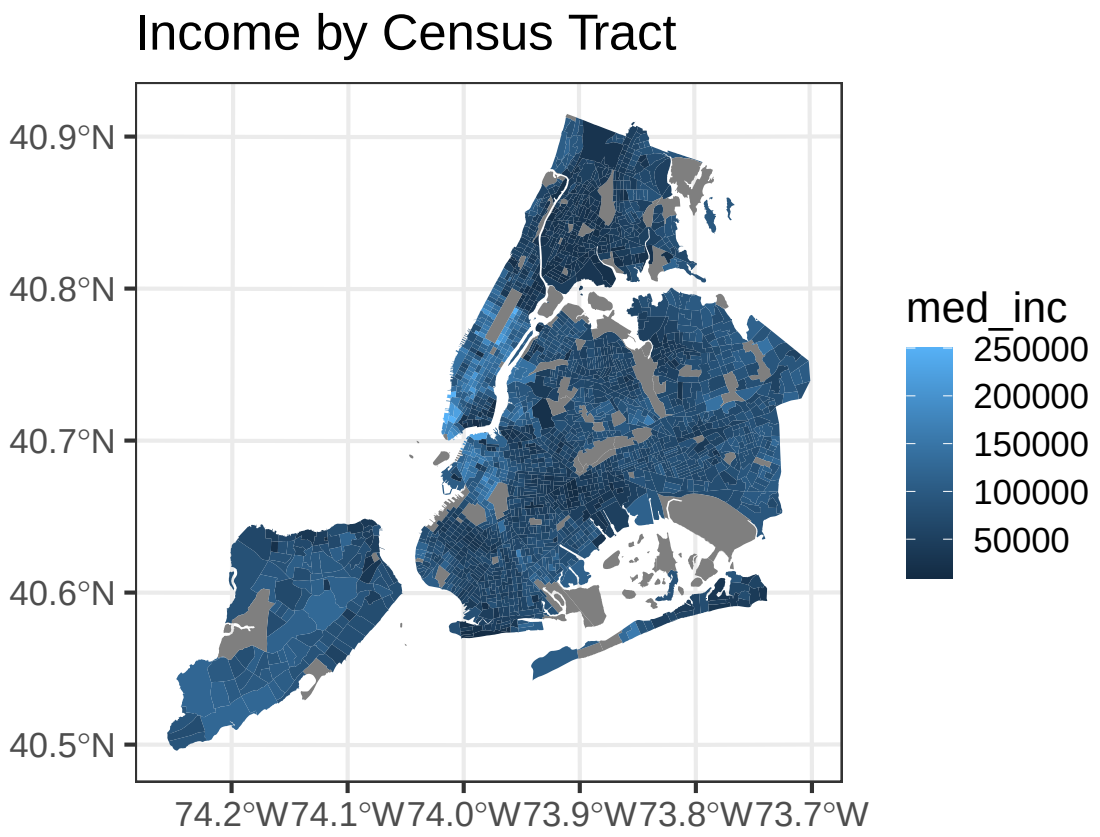
```
ggplot(acs_tracts) +
  geom_sf(aes(fill=tract_race), color=NA) +
  ggtitle('Predominant Racial Group By Tract') +
  theme_bw(base_size=16)
```



This map shows very clear racial boundaries across the city.

Bonus: Map med_income

```
ggplot(acs_tracts) +
  geom_sf(aes(fill=med_inc), color=NA) +
  theme_bw(base_size=16) +
  ggtitle("Income by Census Tract")
```



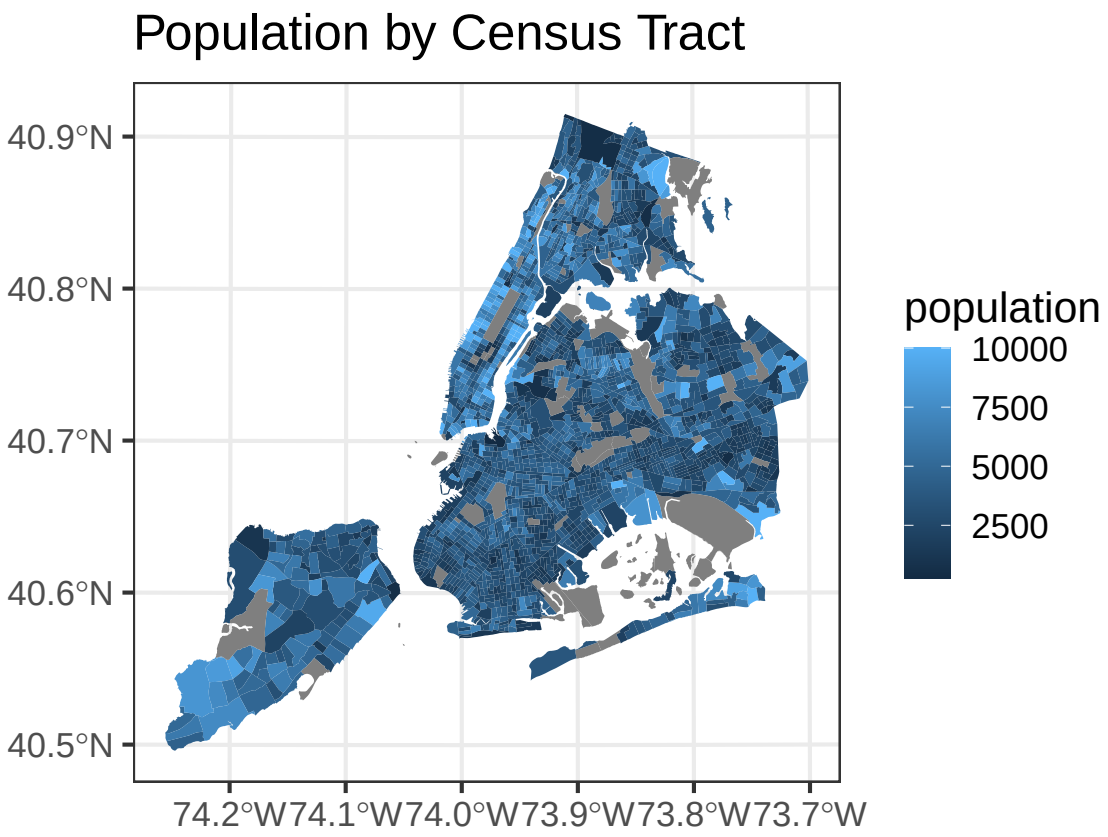
Bonus: We can also look at the population to see where people live in NYC. While most census tracts aim to have around 4k residents per tract, some are extreme outliers. We can see that by looking at a summary of the total_pop column

```
summary(acs_tracts$total_pop)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    2350    3538    3889    4937   28109
```

For our map, we'll cap the population at a maximum of 10k in the color scale so that we still see some variation across the city. We can do that using pmin(total_pop, 10000) to get the smaller value. We might also want to set very low population tracts (like parks) to NA. Together, we can accomplish both by setting fill=ifelse(total_pop > 250, pmin(total_pop, 10000), NA)).

```
ggplot(acs_tracts) +
  geom_sf(aes(fill=ifelse(total_pop > 250, pmin(total_pop, 10000), NA)), color=NA) +
  ggtitle("Population by Census Tract") +
  guides(fill=guide_colourbar(title='population')) +
  theme_bw(base_size=16)
```



NYPD Stops

Now that we've explored the demographic data about New York City, let's compare this to some data on policing patterns.

The following data contains records for every stop conducted by the NYPD during 2019 and 2020. These data can be found from the NYC Open Data Portal: https://data.cityofnewyork.us/Public-Safety/The-Stop-Question-and-Frisk-Data/ftxv-d5ix/about_data

Unfortunately, it is not in a database format and so needs to be downloaded manually by year. I have already done this for you and provided a CSV.

```
sqf <- read_csv("data/sqf.csv")
```

```
## Rows: 22999 Columns: 85
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (71): MONTH2, DAY2, STOP_WAS_INITIATED, RECORD_STATUS_CODE, ISSUING_OFF...
## dbl  (12): STOP_ID, YEAR2, ISSUING_OFFICER_COMMAND_CODE, SUPERVISING_OFFICER...
## dttm  (2): STOP_FRISK_DATE, STOP_FRISK_TIME
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(sqf)
```

```
## # A tibble: 6 x 85
##   STOP_ID STOP_FRISK_DATE     STOP_FRISK_TIME     YEAR2 MONTH2  DAY2
##     <dbl> <dttm>              <dttm>              <dbl> <chr>   <chr>
## 1   2019. 2019-01-02 00:00:00 1899-12-31 14:30:00  2019 January Wednesday
## 2   2019. 2019-01-08 00:00:00 1899-12-31 02:30:00  2019 January Tuesday
## 3   2019. 2019-01-12 00:00:00 1899-12-31 16:54:00  2019 January Saturday
## 4   2019. 2019-01-14 00:00:00 1899-12-31 21:21:00  2019 January Monday
## 5   2020. 2019-01-15 00:00:00 1899-12-31 18:50:00  2019 January Tuesday
## 6   2020. 2019-01-23 00:00:00 1899-12-31 06:16:00  2019 January Wednesday
## # i 79 more variables: STOP_WAS_INITIATED <chr>, RECORD_STATUS_CODE <chr>,
## #   ISSUING_OFFICER_RANK <chr>, ISSUING_OFFICER_COMMAND_CODE <dbl>,
## #   SUPERVISING_OFFICER_RANK <chr>, SUPERVISING_OFFICER_COMMAND_CODE <dbl>,
## #   LOCATION_IN_OUT_CODE <chr>, JURISDICTION_CODE <chr>,
## #   JURISDICTION_DESCRIPTION <chr>, OBSERVED_DURATION_MINUTES <dbl>,
## #   SUSPECTED_CRIME_DESCRIPTION <chr>, STOP_DURATION_MINUTES <dbl>,
## #   OFFICER_EXPLAINED_STOP_FLAG <chr>, ...
```

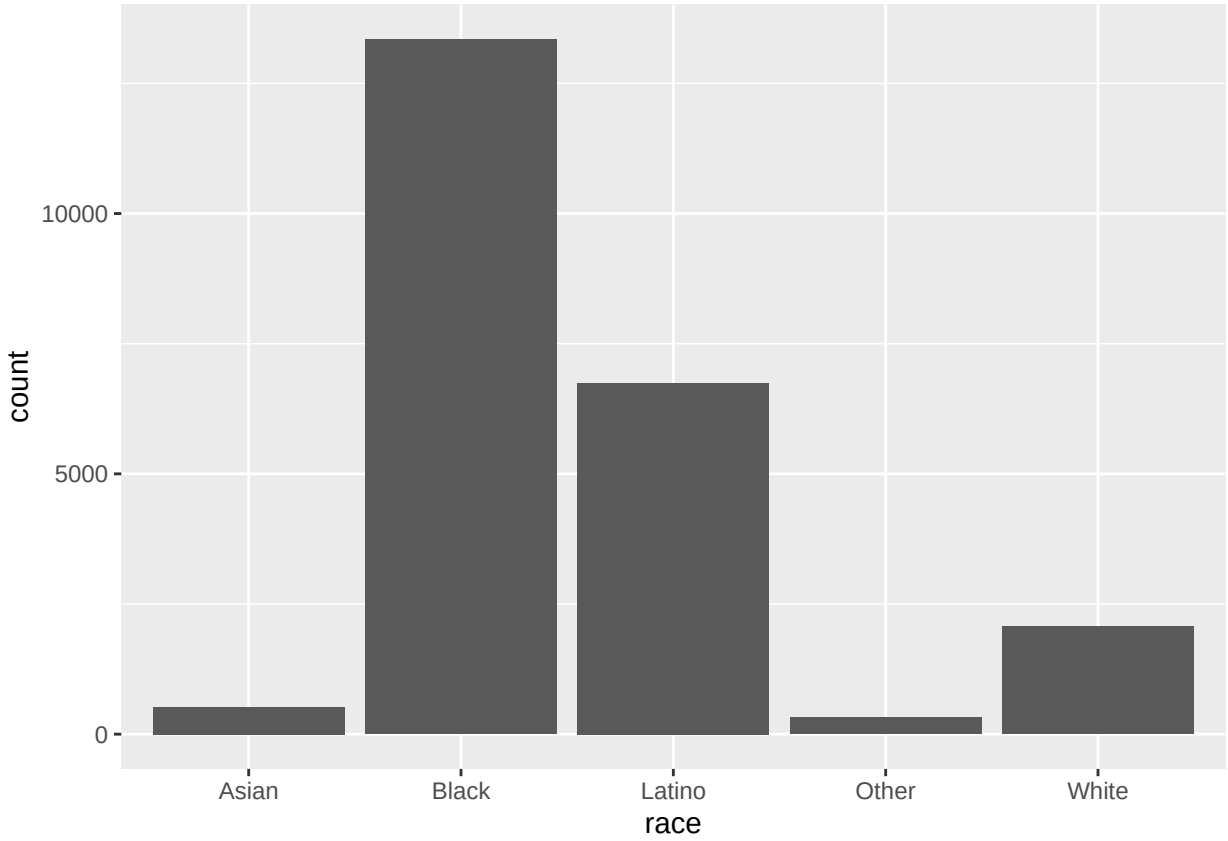
Consider: - What biases might be present in this data? - Why was this data collected?

Let's make the racial groups here align with the ones we're using for the Census

```r
sqf <- sqf %>%
  mutate(
    race=case_when(
      SUSPECT_RACE_DESCRIPTION=="WHITE" ~ "White",
      SUSPECT_RACE_DESCRIPTION=="BLACK" ~ "Black",
      SUSPECT_RACE_DESCRIPTION %in% c("BLACK HISPANIC", "WHITE HISPANIC") ~ "Latino",
      SUSPECT_RACE_DESCRIPTION=="ASIAN / PACIFIC ISLANDER" ~ "Asian",
      T ~ "Other"
    )
  )
```

How does a bar graph of the racial groups look for this data? How does that compare to the ones we made earlier of the city's demographics?

```r
ggplot(sqf) +
  geom_bar(aes(x=race))
```



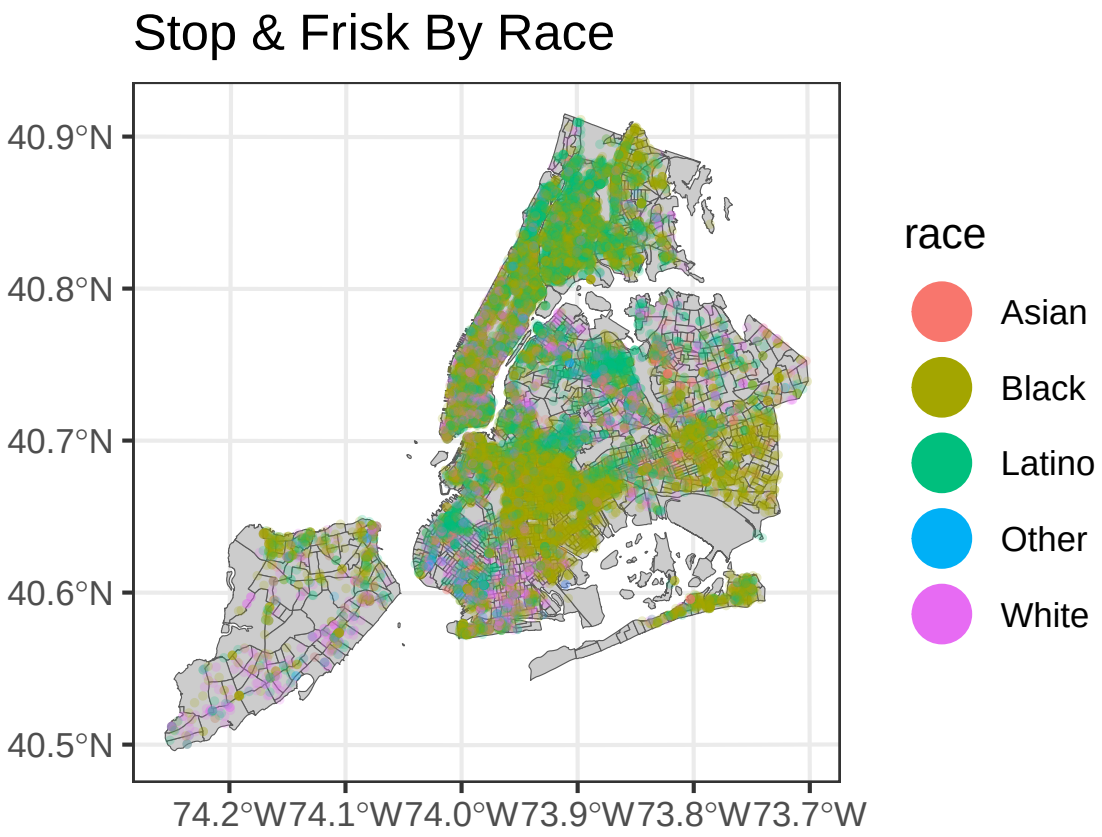
We can see this pattern even more clearly through mapping the stop locations.

First we need to provide information about which columns tell us the stop locations.

```
sqf_geo <- st_as_sf(sqf, coords=c('STOP_LOCATION_X','STOP_LOCATION_Y'), crs='epsg:2908', remove=FALSE)
```

Then, we can map the stops by the race of the person stopped. We'll map the census tracts file as a base layer, and then plot points for each stop on top of the tracts.

```
ggplot() +
  geom_sf(data=acs_tracts, fill="gray80") +
  geom_sf(data=sqf_geo, aes(col=race), alpha=.2, size=1) +
  guides(colour = guide_legend(override.aes=list(alpha=1, size=10))) +
  ggtitle('Stop & Frisk By Race') +
  theme_bw(base_size=16)
```



How can we see a racial bias in the people stopped by police compared to the city's population?

Surveillance - Camera Locations

We'll also load in Camera locations data collected by Amnesty International. Here we only need it at the census tract level, not at the individual camera level, so we'll load in their aggregated file that counts how many cameras are in each census tract.

This analysis only looks at public cameras.

The code to generate this data and analysis can be found at Amnesty's github (https://github.com/amnesty-crisis-evidence-lab/decode-surveillance-nyc), but it's included here for convenience. The columns here have been calculated by the Decode Surveillance team, and are already aggregated to the census tract level. They include: - cameras: the count of public cameras in the census tract - cameras_within_200m: the count of public cameras within the tract or within 200m of the tract's boundaries (as they may also capture images within the tract) - eff_cameras: Count of "effective cameras," or the estimated coverage of public cameras accounting for the fact that some cameras cover overlapping areas and so do not contribute new information/surveillance to those areas - eff_cameras_within_200m: "Effective cameras" in and within 200m of the census tract's boundaries

```
cameras <- read_csv("data/camera_count.csv")
```

```
## Rows: 2165 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (5): GEOID, eff_cameras_within_200m, eff_cameras, cameras_within_200m, c...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(cameras)
```

```
## # A tibble: 6 x 5
##         GEOID eff_cameras_within_200m eff_cameras cameras_within_200m cameras
##         <dbl>                   <dbl>       <dbl>               <dbl>   <dbl>
## 1 36005000200                   0.966      0.0462                   1       0
## 2 36005000400                   1.29       0.953                    1       1
## 3 36005001600                   1.20       0.146                    1       0
## 4 36005001900                   6.77       4.77                    10       9
## 5 36005002000                   3.10       1.36                     3       2
## 6 36005002300                   1.64       0.146                    5       0
```

And add it onto our main dataframe by GEOID (we may need to change GEOID in the camera dataset to a character/string using as.character())

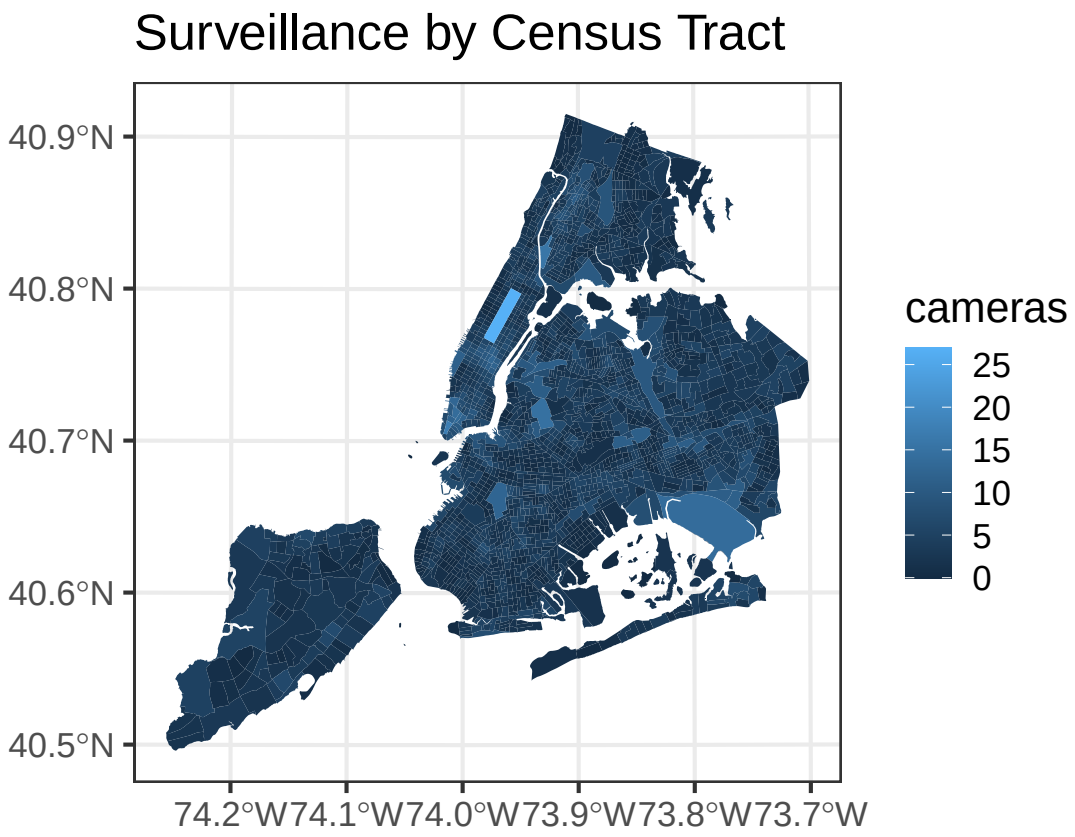
```
acs_tracts <- acs_tracts %>%
  left_join(cameras %>% mutate(GEOID=as.character(GEOID)), by="GEOID")
```

```
head(acs_tracts)
```

```
## Simple feature collection with 6 features and 30 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 1007285 ymin: 238802.2 xmax: 1034600 ymax: 270146.9
## Projected CRS: NAD83(HARN) / New York Long Island (ftUS)
## # A tibble: 6 x 31
##   fips  STATEFP COUNTYFP TRACTCE GEOID     NAME.x NAMELSAD MTFCC FUNCSTAT  ALAND
##   <chr> <chr>   <chr>    <chr>   <chr>     <chr>  <chr>    <chr> <chr>     <dbl>
## 1 36005 36      005      034500  36005034~ 345    Census ~ G5020 S        274627
## 2 36005 36      005      030701  36005030~ 307.01 Census ~ G5020 S        396736
## 3 36005 36      005      031000  36005031~ 310    Census ~ G5020 S        634013
## 4 36005 36      005      014100  36005014~ 141    Census ~ G5020 S        177165
## 5 36005 36      005      027401  36005027~ 274.01 Census ~ G5020 S        701116
## 6 36005 36      005      035000  36005035~ 350    Census ~ G5020 S        191626
## # i 21 more variables: AWATER <dbl>, INTPTLAT <chr>, INTPTLON <chr>,
## #   borough <chr>, geometry <MULTIPOLYGON [US_survey_foot]>, NAME.y <chr>,
## #   male <dbl>, female <dbl>, total_pop <dbl>, white <dbl>, black <dbl>,
## #   asian <dbl>, latino <dbl>, med_inc <dbl>, otherrace <dbl>, county <chr>,
## #   tract_race <chr>, eff_cameras_within_200m <dbl>, eff_cameras <dbl>,
## #   cameras_within_200m <dbl>, cameras <dbl>
```

Finally, let's map effective cameras (eff_cameras) across the city:

```
ggplot(acs_tracts) +
  geom_sf(aes(fill=eff_cameras_within_200m), color=NA) +
  guides(fill=guide_colourbar(title='cameras')) +
  theme_bw(base_size=16) +
  ggtitle("Surveillance by Census Tract")
```



- What patterns do you see?
- What future analyses might you want to do with these data sources?

Additional: Getting to a finalized dataset We'd need to aggregate our stops data to the census tract level and add it onto our main dataframe.

This data is currently at the incident level, but we may need it at the tract level if we want to analyze it. Let's group by GEOID and count the number of stops and the number of black people stopped.

```
sqf_tracts <- sqf %>%
  group_by(GEOID) %>%
  summarize(
    n_stops=n(),
    n_2019=sum(YEAR2=="2019"),
    n_2020=sum(YEAR2=="2020"),
    n_black=sum(race=="Black")
  )

head(sqf_tracts)
```

```
## # A tibble: 6 x 5
##         GEOID n_stops n_2019 n_2020 n_black
##         <dbl>   <int>  <int>  <int>   <int>
## 1 36005000200       5      1      4       0
## 2 36005000400       9      3      6       6
```

```
## 3 36005001600       7      3      4       5
## 4 36005001900      11      5      6       5
## 5 36005002000      52     26     26      34
## 6 36005002300      21      6     15      16
```

We can then add it onto our dataframe

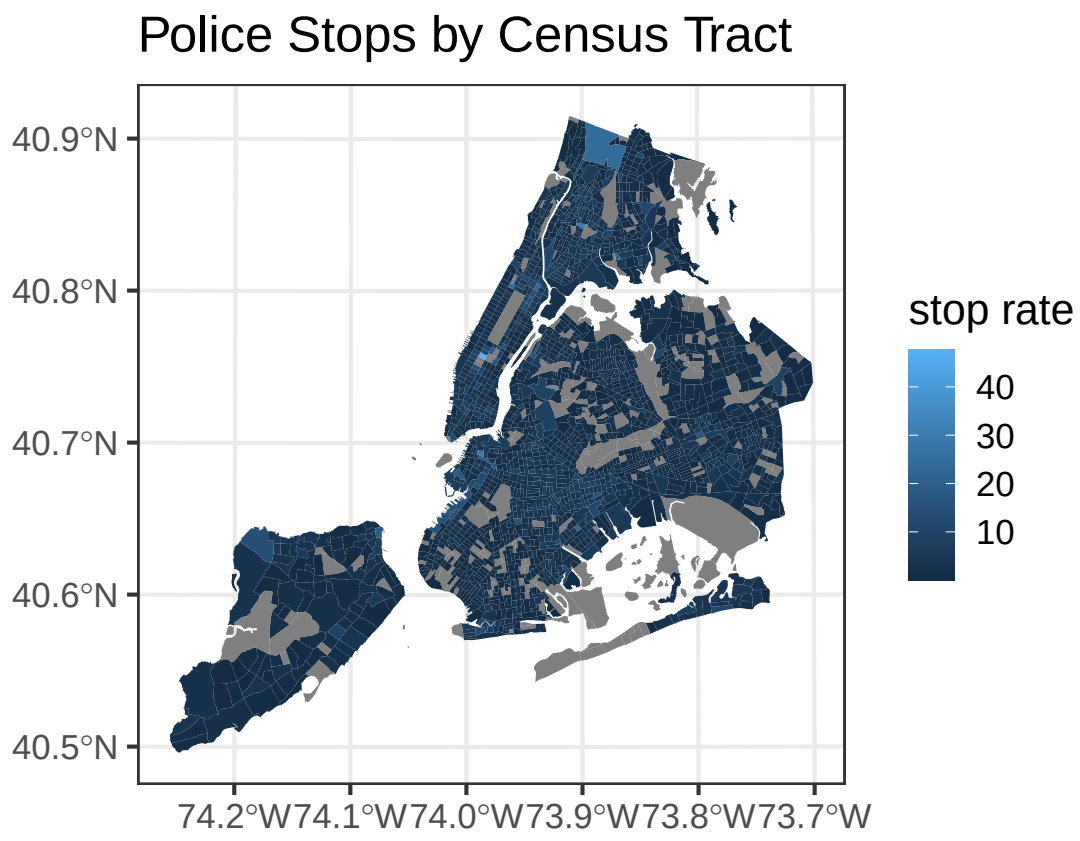
```
acs_tracts <- acs_tracts %>%
  left_join(sqf_tracts %>% mutate(GEOID=as.character(GEOID)), by="GEOID")
```

Calculate stop rates and surveillance rates per 1000 residents

```
acs_tracts <- acs_tracts %>%
  mutate(stop_rate=if_else(total_pop > 250, (n_stops/total_pop)*1000, NA),
         surv_rate=if_else(total_pop > 250, (eff_cameras_within_200m/total_pop)*1000, NA),
         surv_rank=rank(-surv_rate),
         surv_class=if_else(total_pop > 250, if_else(surv_rank < 50, "top 50", "other"), NA))
```

We can now map the number of stops (or number of stops for black people) in a tract

```
ggplot(acs_tracts) +
  geom_sf(aes(fill=stop_rate), color=NA) +
  guides(fill=guide_colourbar(title='stop rate')) +
  theme_bw(base_size=16) +
  ggtitle("Police Stops by Census Tract")
```



```
ggplot(acs_tracts) +
  geom_sf(aes(fill=factor(surv_class)), color=NA) +
  theme_bw(base_size=16) +
  scale_fill_manual(values=c('top 50'='red', 'other'='bisque2'), na.value='antiquewhite') +
  ggtitle("Surveillance by Census Tract")
```