Introduction to Geographic Information Systems

2022-02-17

# 1 Introduction

## 1.1 Workshop Objectives

In this workshop, we will introduce you to spatial data analysis and visualization using R Studio. We will do so through a substantive exploration of systemic racism in traffic police stops. In particular, we will take a large (over 3,000,000 observations) dataset of Colorado traffic stops released by the [Stanford Open Policing Project](https://openpolicing.stanford.edu/), and develop a simple county-level measure of the magnitude of anti-Black racial bias in traffic police stops during the year 2010. We will then display county-level variation in this bias index on a map of Colorado counties.

The workshop has several learning objectives. Among other things, you will learn how to:

* Read in datasets (both tabular and spatial) into R Studio
* Clean and process data in R Studio using *tidyverse* functions (i.e. subset data, reshape data, summarize datasets, join different datasets together, and define new variables)
* Make static maps and interactive web maps in R Studio using the *tmap* package
* Export your maps from R Studio so they can be embedded in your reports and websites

Most broadly, and most importantly, you will gain an appreciation for how to use real-world social scientific data to better understand important social problems, and use this understanding to reflect on possible ways to address these problems.

## 1.2 Data

The workshop makes use of the following three datasets:

* Colorado State Patrol traffic stops data, available through the Stanford Open Policing Project (mentioned above). The data was collected by Pierson et al. (2020). The data is available at the project website [here](https://openpolicing.stanford.edu/data/), under the “CO” heading; if you’re downloading the data straight from the website, be sure to use the “State Patrol” data.
* A dataset of county-level demographic data (based on the 2010 decennial census), which your instructors generated prior to the workshop. See the Appendix to this tutorial to see how this census data was extracted.
* A spatial dataset of Colorado counties, available from the US census via the [Colorado GeoLibrary](https://geo.colorado.edu/), a curated online archive of Colorado-related spatial data. The specific dataset we will use is available [here](https://geo.colorado.edu/catalog/47540-5e712aeda3d91e0009f59fc7). Note that this spatial dataset is stored as a shapefile, which is a common file format used to store spatial data.

For convenience, all of this data can be downloaded from this [folder](https://www.dropbox.com/sh/sd7wwi5b0r9s9mn/AABscRHrEvXRoX26e_yqEiTFa?dl=0).

## 1.3 R and R Studio Installation

In order to follow along with the lesson, you must install both R and R Studio. Please see the installation instructions at this [page](https://datacarpentry.org/r-socialsci/setup.html), from [Data Carpentry](https://datacarpentry.org/).

## 1.4 The R Studio Interface

If you haven’t used R Studio before, you can get a quick tour of the R Studio interface and get oriented by reading the brief **Introduction to RStudio** section [here](https://datacarpentry.org/r-intro-geospatial/01-rstudio-intro/index.html).

## 1.5 Installing and loading packages

In order to carry out the analysis in the workshop, you must install the *tidyverse* (a suite of R packages that facilitate data science and analysis), *sf* (a package used to load and process spatial datasets in R), and *tmap* (a package that allows one to create maps in R). If you haven’t already installed one or more of these packages, you can do so by placing the package name in quotes as an argument to the install.packages function.

For example, if you wanted to install *tmap*, you would print the following in your R Studio console, or run it from a script (which you can open by clicking File, selecting New File, and then clicking R Script):

install.packages("tmap")

After installing the required passage, you must load the packages into memory, which you can do with the following:

# Loads libraries  
library(tidyverse)  
library(tmap)  
library(sf)

Note that you only need to install packages once on a given computer, and (usually) do not need to reinstall packages after quitting an R session and opening up a new one. However, you *do* need to reload any necessary libraries each time you open a new R session.

If you would like additional information about how R packages work and the installation process, please see the **Installing Packages** section in this Data Carpentry tutorial: <https://datacarpentry.org/r-intro-geospatial/01-rstudio-intro/index.html>.

## 1.6 Setting your working directory

Before we can bring our data into R Studio and begin the tutorial, we have to specify the location of the relevant data on our computer. This step is known as setting one’s working directory. Before setting the working directory, make sure that you’ve downloaded the data provided at the [folder](https://www.dropbox.com/sh/sd7wwi5b0r9s9mn/AABscRHrEvXRoX26e_yqEiTFa?dl=0) mentioned above to a directory on your computer.

If you’re unfamiliar with the concept of file paths, the easiest way to set your working directory is through the R Studio menu. To do so, follow these steps:

* First, Click on the “Session” menu on the R Studio menu bar at the top of your screen, and then scroll down to the “Set Working Directory” button in the menu that opens up.
* When you hover over the “Set Working Directory” button, a subsidiary menu that contains a button that says “Choose Directory” will open; click this “Choose Directory” button.
* In the dialog box that opens up, navigate to the directory that contains the downloaded workshop data, select it, and click “Open”. At this point, your working directory should be set!

**Alternatively**, if you are familiar with the concept of file paths, and know the file path to the folder containing the downloaded datasets, you can set the working directly using the setwd() function, where the argument to the function is the relevant file path enclosed in quotation marks. For example:

# Setting the working directory programmatically  
setwd("<FILE PATH TO DATA DIRECTORY HERE>")

At this point, we’re ready to begin the main part of our lesson!

# 2 Loading data

Let’s begin by reading in the dataset on State Patrol traffic stops released by the Stanford Open Policing Project. The data can be downloaded from this [website](https://openpolicing.stanford.edu/data/), but has also been made available to you as part of the workshop materials as a CSV file.

Once the traffic patrol data has been downloaded into your working directory, pass the name of the file to the read\_csv function, and assign it to an object. Here, we’ll assign the traffic patrol data to an object named co\_traffic\_stops. Note that the name of an object is arbitrary, but ideally should meaningfully describe the data that has been assigned to it.

# Read in Stanford police data for Colorado and assign to object named "co\_traffic\_stops"  
co\_traffic\_stops<-read\_csv("co\_statewide\_2020\_04\_01.csv")

##   
## ── Column specification ───────────────────────────────────────────────────────────────────────  
## cols(  
## .default = col\_character(),  
## date = col\_date(format = ""),  
## time = col\_logical(),  
## subject\_age = col\_double(),  
## arrest\_made = col\_logical(),  
## citation\_issued = col\_logical(),  
## warning\_issued = col\_logical(),  
## contraband\_found = col\_logical(),  
## search\_conducted = col\_logical()  
## )  
## ℹ Use `spec()` for the full column specifications.

Once the traffic patrol data has been read into R studio and assigned to an object, we can print the contents of the dataset to the console by typing the name of that object into the R Studio console (note that only the first few records will be printed)

# Print the contents of "co\_traffic\_stops" (i.e. the CO traffic patrol data) to the console; the first few records of the dataset will print  
co\_traffic\_stops

## # A tibble: 3,112,853 × 20  
## raw\_row\_number date time location county\_name subject\_age subject\_race subject\_sex  
## <chr> <date> <lgl> <chr> <chr> <dbl> <chr> <chr>   
## 1 1947986|1947987 2013-06-19 NA 19, I70… Mesa County 26 hispanic male   
## 2 1537576 2012-08-24 NA 254, H2… Jefferson … NA <NA> <NA>   
## 3 1581594 2012-09-23 NA 115, I7… Logan Coun… 52 white male   
## 4 1009205 2011-08-25 NA 197, H8… Douglas Co… 32 white female   
## 5 1932619 2013-06-08 NA 107, H2… Kiowa Coun… 33 hispanic male   
## 6 1179436 2011-12-23 NA 48, 384… Boulder Co… NA <NA> <NA>   
## 7 1326795 2012-04-07 NA 0, R250… Boulder Co… 39 white male   
## 8 1786795 2013-03-03 NA 19, E47… Arapahoe C… 44 white female   
## 9 1552164 2012-09-02 NA 224, H2… Park County NA <NA> <NA>   
## 10 1004281|1004282|… 2011-08-21 NA R2000, … Adams Coun… 32 hispanic male   
## # … with 3,112,843 more rows, and 12 more variables: officer\_id\_hash <chr>, officer\_sex <chr>,  
## # type <chr>, violation <chr>, arrest\_made <lgl>, citation\_issued <lgl>,  
## # warning\_issued <lgl>, outcome <chr>, contraband\_found <lgl>, search\_conducted <lgl>,  
## # search\_basis <chr>, raw\_Ethnicity <chr>

We can also view the co\_traffic\_stops object (or, for that matter, any dataset in R Studio) within the R Studio data viewer by passing the name of the relevant object to the View function:

# Inspect ```co\_traffic\_stops``` in the R Studio data viewer  
View(co\_traffic\_stops)

# 3 Cleaning and filtering data

To make things tractable (after all, the entire dataset has more than 3,000,000 observations!), let’s focus our attention on Colorado traffic patrol from the year 2010; this also has the advantage of allowing us to eventually use 2010 census data in crafting our measure of racial bias in traffic stops. Note that currently, there isn’t a separate field which contains the year in which a particular stop took place; rather, there is a “date” field, in which the date is stored in YYYY-MM-DD format. The easiest way to extract observations from the year 2010 is therefore to first extract the YYYY information from the “date” field, and use that information to make a new “Year” field, which only contains the year of a given stop; we can then extract the 2010 observations based on this newly generated “Year” field.

## 3.1 Create a “Year” field

To create this new “Year” field within co\_traffic\_stops, we can use the mutate function, which is a tidyverse function that allows us to define new variables within a dataset, and the substr function, which allows us to extract a subset of a given string.

The code below takes the data in the co\_traffic\_stops object, and then (%>%) creates a new field named “Year” using the mutate function; this field is set equal to the the first four digits of the existing “date” field by passing co\_traffic\_stops$data, 1, 4 to the substr function. It then uses the assignment operator (<-) to assign this change back to the co\_traffic\_stops object, which permanently updates the dataset with the addition of the new “Year” field:

# Creates "Year" field, that contains the year of a given stop, in "co\_traffic\_stops"  
co\_traffic\_stops<-co\_traffic\_stops %>%   
 mutate(Year=substr(co\_traffic\_stops$date, 1,4))

Let’s check the updated co\_traffice\_stops object and make sure that the new field has been successfully created:

# prints contents of "co\_traffic\_stops"  
co\_traffic\_stops

## # A tibble: 3,112,853 × 21  
## county\_name date Year raw\_row\_number time location subject\_age subject\_race  
## <chr> <date> <chr> <chr> <lgl> <chr> <dbl> <chr>   
## 1 Mesa County 2013-06-19 2013 1947986|1947987 NA 19, I70… 26 hispanic   
## 2 Jefferson County 2012-08-24 2012 1537576 NA 254, H2… NA <NA>   
## 3 Logan County 2012-09-23 2012 1581594 NA 115, I7… 52 white   
## 4 Douglas County 2011-08-25 2011 1009205 NA 197, H8… 32 white   
## 5 Kiowa County 2013-06-08 2013 1932619 NA 107, H2… 33 hispanic   
## 6 Boulder County 2011-12-23 2011 1179436 NA 48, 384… NA <NA>   
## 7 Boulder County 2012-04-07 2012 1326795 NA 0, R250… 39 white   
## 8 Arapahoe County 2013-03-03 2013 1786795 NA 19, E47… 44 white   
## 9 Park County 2012-09-02 2012 1552164 NA 224, H2… NA <NA>   
## 10 Adams County 2011-08-21 2011 1004281|1004282|1… NA R2000, … 32 hispanic   
## # … with 3,112,843 more rows, and 13 more variables: subject\_sex <chr>, officer\_id\_hash <chr>,  
## # officer\_sex <chr>, type <chr>, violation <chr>, arrest\_made <lgl>, citation\_issued <lgl>,  
## # warning\_issued <lgl>, outcome <chr>, contraband\_found <lgl>, search\_conducted <lgl>,  
## # search\_basis <chr>, raw\_Ethnicity <chr>

Note the newly created Year field above. We can also check to make sure that the “Year” field has been successfully created by viewing co\_traffic\_stops in the R Studio data viewer with View(co\_traffic\_stops).

## 3.2 Filter by year

Now that we have created the “Year” field, we can use it to extract the 2010 observations using the filter function , and assign the new dataset of 2010 stops to a new object named co\_traffic\_stops\_2010:

# Extract 2010 observations and assign to a new object named "co\_traffic\_stops\_2010"  
co\_traffic\_stops\_2010<-co\_traffic\_stops %>% filter(Year==2010)

When we print the contents of the newly created co\_traffic\_stops\_2010 object, note that the observations are now only from 2010.

# Print contents of "co\_traffic\_stops\_2010" object  
co\_traffic\_stops\_2010

## # A tibble: 470,284 × 21  
## date Year county\_name subject\_race raw\_row\_number time location subject\_age  
## <date> <chr> <chr> <chr> <chr> <lgl> <chr> <dbl>  
## 1 2010-04-17 2010 Montezuma County white 188721|188722 NA 2, 989,… 16  
## 2 2010-04-17 2010 Montezuma County white 187958 NA 991, 32 54  
## 3 2010-04-17 2010 Montezuma County hispanic 188451 NA 9, 280,… 49  
## 4 2010-04-17 2010 Montezuma County white 186989|186990|186… NA 3, 277,… 16  
## 5 2010-04-17 2010 Montezuma County white 186997|186998|186… NA 3, 277,… 37  
## 6 2010-04-17 2010 Montezuma County white 186993|186994|186… NA 3, 277,… 39  
## 7 2010-12-21 2010 Mineral County <NA> 600865 NA 164.5, … 110  
## 8 2010-12-21 2010 Mineral County <NA> 600477 NA 163, 29… 110  
## 9 2010-01-20 2010 Pueblo County hispanic 36625|36626 NA 312, H5… 45  
## 10 2010-01-01 2010 Chaffee County white 275 NA 127, H2… 17  
## # … with 470,274 more rows, and 13 more variables: subject\_sex <chr>, officer\_id\_hash <chr>,  
## # officer\_sex <chr>, type <chr>, violation <chr>, arrest\_made <lgl>, citation\_issued <lgl>,  
## # warning\_issued <lgl>, outcome <chr>, contraband\_found <lgl>, search\_conducted <lgl>,  
## # search\_basis <chr>, raw\_Ethnicity <chr>

# 4 Transforming Data

Now that we have created a new data object that contains information on traffic stops from our year of interest (2010), let’s do a bit of work with the data so that we can find out the total number of traffic stops within each county, and the number of stops within each county in which the person stopped by the traffic police was Black.

## 4.1 Tabulate county-level count of traffic stops by race

First, let’s find out the racial breakdown of stops for each county, using the data in the “subject\_race” field of co\_traffice\_stops\_2010.

Below, we take the co\_traffic\_stops\_2010 object, declare “county\_name” as a grouping variable using group\_by(county\_name), and then count up the number of stops associated with each racial category within each county using count(subject\_race). The dataset thats results from these operations is assigned to a new object named co\_county\_summary:

# Compute county-level count of traffic stops by race  
co\_county\_summary<-co\_traffic\_stops\_2010 %>%   
 group\_by(county\_name) %>%   
 count(subject\_race)

Let’s print the first few rows from the dataset and observe its structure:

# Prints contents of "co\_county\_summary"  
co\_county\_summary

## # A tibble: 439 × 3  
## # Groups: county\_name [65]  
## county\_name subject\_race n  
## <chr> <chr> <int>  
## 1 Adams County asian/pacific islander 582  
## 2 Adams County black 1208  
## 3 Adams County hispanic 8012  
## 4 Adams County other 36  
## 5 Adams County unknown 462  
## 6 Adams County white 20225  
## 7 Adams County <NA> 3825  
## 8 Alamosa County asian/pacific islander 18  
## 9 Alamosa County black 43  
## 10 Alamosa County hispanic 1537  
## # … with 429 more rows

Note that each row contains information on the number of times a person from a given racial category was stopped by the traffic police in a given county. For example, in Adams County, CO, Asian/Pacific Islander motorists were stopped 582 times, Black motorists were stopped 1208 times, and white motorists were stopped 20225 times. The dataset contains information on the racial and ethnic breakdown of police stops for each county in Colorado, in the year 2010.

## 4.2 Reshape the data

Note that co\_county\_summary is currently a “long” dataset, in which there are multiple rows associated with each county, with each row corresponding to a distinct county/racial category combination, and the n column providing information on the number of stops associated with that county/racial category pairing.

It will be easier to instead work with a “wide” dataset, in which each county is associated with a single row, and each racial category is assigned to its own column. Each cell in this “wide” dataset would correspond to the number of stops associated with a given county (defined by the row) for a given racial category (defined by the column).

To reshape the dataset from its current “long” format into a “wide” format, we can use the pivot\_wider function. The code below takes the current co\_county\_summary data object (in “long” format), and then transforms it into a “wide” format with pivot\_wider(names\_from=subject\_race, values\_from=n). The names\_from argument specifies the name of the current column which contains the categories which we want to transform into columns (here, subject\_race), while the values\_from argument specifies the name of the current column which contains the values that will be associated with each county/racial category combination (here, n). Finally, the code below assigns the transformed dataset to a new object named co\_county\_summary\_wide:

# Transforms "co\_county\_summary" from long format to wide, and assigns the reshaped dataset to a new object named "co\_county\_summary\_wide"  
co\_county\_summary\_wide<-co\_county\_summary %>%   
 pivot\_wider(names\_from=subject\_race, values\_from=n)

Let’s print the contents of co\_county\_summary\_wide to ensure that the data has indeed been transformed into a “wide” format:

# prints contents of "co\_county\_summary\_wide"  
co\_county\_summary\_wide

## # A tibble: 65 × 8  
## # Groups: county\_name [65]  
## county\_name `asian/pacific islander` black hispanic other unknown white `NA`  
## <chr> <int> <int> <int> <int> <int> <int> <int>  
## 1 Adams County 582 1208 8012 36 462 20225 3825  
## 2 Alamosa County 18 43 1537 9 30 2427 414  
## 3 Arapahoe County 540 1819 1862 12 300 11089 1898  
## 4 Archuleta County 17 28 392 71 41 4125 417  
## 5 Baca County 11 61 288 NA 6 971 174  
## 6 Bent County 8 46 314 1 6 1155 278  
## 7 Boulder County 345 192 1050 10 180 9682 1594  
## 8 Broomfield County 32 22 104 3 18 690 226  
## 9 Chaffee County 43 37 361 9 71 4806 1194  
## 10 Cheyenne County 10 38 147 3 2 821 85  
## # … with 55 more rows

Indeed, we can see that each row is now associated with a single county, and each column is now associated with a given racial category.

## 4.3 Calculate total stops for each county in co\_county\_summary\_wide

Now that we have our dataset in “wide” format, let’s create a new column, named “total\_stops” that contains information on the total number of traffic stops for each county. We can create this column by calculating the sum total of stops across all of the racial categories for each county, which yields the total number of stops for each county.

The code below takes the co\_county\_summary\_wide object, and then calls the rowise function, which allows us to make calculations across the rows of a data frame. It then creates a new column, called “total\_stops” using the now-familiar mutate function; this “total\_stops” column is populated by taking the sum of traffic stops across racial categories for each county. This is accomplished with sum(c\_across(where(is.integer)), na.rm=TRUE). This expression can be translated as follows: “for each row in the dataset, calculate the sum across the columns whenever the value in a column is an integer; whenever a cell value is ‘NA’, simply ignore it in the calculation.” We’ll assign the dataset, with the newly added “total\_stops” column, back to the same co\_county\_summary\_wide object; this effectively overwrites the contents of the current co\_county\_summary\_wide object (a dataset without the “total\_stops” column), with the new dataset (which does have the “total\_stops” column).

# Takes the existing "co\_county\_summary\_wide" dataset, and creates a new column called "total\_stops" that sums the values across columns for each row; the revised dataset is assigned back to "co\_county\_summary\_wide", which overwrites the object's previous contents with the revised dataset  
  
co\_county\_summary\_wide<-co\_county\_summary\_wide %>%   
 rowwise() %>%   
 mutate(total\_stops=sum(c\_across(where(is.integer)), na.rm=TRUE))

Let’s print the contents of the co\_county\_summary\_wide object and confirm that the new column has been successfully created:

# Prints updated contents of "co\_county\_summary\_wide"  
co\_county\_summary\_wide

## # A tibble: 65 × 9  
## # Rowwise: county\_name  
## county\_name total\_stops `asian/pacific isla…` black hispanic other unknown white `NA`  
## <chr> <int> <int> <int> <int> <int> <int> <int> <int>  
## 1 Adams County 34350 582 1208 8012 36 462 20225 3825  
## 2 Alamosa County 4478 18 43 1537 9 30 2427 414  
## 3 Arapahoe County 17520 540 1819 1862 12 300 11089 1898  
## 4 Archuleta County 5091 17 28 392 71 41 4125 417  
## 5 Baca County 1511 11 61 288 NA 6 971 174  
## 6 Bent County 1808 8 46 314 1 6 1155 278  
## 7 Boulder County 13053 345 192 1050 10 180 9682 1594  
## 8 Broomfield County 1095 32 22 104 3 18 690 226  
## 9 Chaffee County 6521 43 37 361 9 71 4806 1194  
## 10 Cheyenne County 1106 10 38 147 3 2 821 85  
## # … with 55 more rows

## 4.4 Clean co\_county\_summary\_wide and assign to new object

Let’s clean up the co\_county\_summmary\_wide dataset a bit more to make it easier to work with. First, because our interest in this workshop is in exploring the possibility that Black motorists suffer disproportionately high traffic stop rates (relative to their share of the overall adult population), let’s create a new object that only contains the data that is essential for the subsequent analysis: the county’s name (“county\_name”), the number of Black motorists that were stopped (“black”), and the total number of stops in the county across all racial categories ("total\_stops).

The code below takes the existing co\_county\_summary\_wide object, and then uses the select function to select the columns we want to keep. It then assigns this selection to a new object named “co\_county\_black\_stops”:

# Selects the "county\_name", "black", and "total\_stops" columns from the "co\_county\_summary\_wide" object, and assigns the selection to a new object named "co\_county\_black\_stops"  
  
co\_county\_black\_stops<-co\_county\_summary\_wide %>%  
 select(county\_name, black, total\_stops)

Let’s open up the newly created co\_county\_black\_stops object to ensure that these changes have been successfully implemented:

# Prints contents of "co\_county\_black\_stops"  
co\_county\_black\_stops

## # A tibble: 65 × 3  
## # Rowwise: county\_name  
## county\_name black total\_stops  
## <chr> <int> <int>  
## 1 Adams County 1208 34350  
## 2 Alamosa County 43 4478  
## 3 Arapahoe County 1819 17520  
## 4 Archuleta County 28 5091  
## 5 Baca County 61 1511  
## 6 Bent County 46 1808  
## 7 Boulder County 192 13053  
## 8 Broomfield County 22 1095  
## 9 Chaffee County 37 6521  
## 10 Cheyenne County 38 1106  
## # … with 55 more rows

As expected, we now have a dataset that contains information on the total number of traffic stops for each Colorado county in 2010 (“total\_stops”), and the number of traffic stops that involved Black motorists (“black”).

The name of the column containing information on the number of Black motorists stopped by the traffic patrol is simply “black”, which was inherited from the initial dataset from the Stanford Policing project. However, this column name is somewhat vague, and may cause confusion down the road when we work with census demographic data. Therefore, let’s rename that column name to “black\_stops” using the rename function, and assign that change back to co\_county\_black\_stops:

# Takes the existing "co\_county\_black\_stops" object and renames the column named "black" to "black\_stops"; assigns the modified dataset back to "co\_county\_black\_stops", which overwrites the existing contents of "co\_county\_black\_stops"  
co\_county\_black\_stops<-co\_county\_black\_stops %>%  
 rename(black\_stops=black)

Let’s now print the modified co\_county\_black\_stops object:

# Prints contents of "co\_county\_black\_stops"  
co\_county\_black\_stops

## # A tibble: 65 × 3  
## # Rowwise: county\_name  
## county\_name black\_stops total\_stops  
## <chr> <int> <int>  
## 1 Adams County 1208 34350  
## 2 Alamosa County 43 4478  
## 3 Arapahoe County 1819 17520  
## 4 Archuleta County 28 5091  
## 5 Baca County 61 1511  
## 6 Bent County 46 1808  
## 7 Boulder County 192 13053  
## 8 Broomfield County 22 1095  
## 9 Chaffee County 37 6521  
## 10 Cheyenne County 38 1106  
## # … with 55 more rows

Note that the name of the column has successfully been changed, and is now more descriptive.

Finally, if we view co\_county\_black\_stops within the R Studio data viewer (View(co\_county\_black\_stops)), we’ll note a small quirk in the dataset. In particular, while Colorado has 64 counties, the dataset has 65 rows; one of the rows is an extra row, with an “NA” value associated with the “county\_name” field, and 20 total traffic stops associated with it. Because we’re interested in a county-level analysis, and these stops are not associated with an actual county, we’ll go ahead and delete that row with the following code, which takes the current co\_county\_black\_stops object, and then uses the filter function to select only those rows for which “county\_name” is NOT EQUAl (!=) to “NA”; after effectively deleting the row where “county\_name” is set to “NA”, the code assigns the modified dataset back to co\_county\_black\_stops:

# Takes the "co\_county\_black\_stops" object and removes the row for which "county\_name" is "NA"; assigns the modified dataset back to "co\_county\_black\_stops", which overwrites the dataset that is currently assigned to that object  
co\_county\_black\_stops<-co\_county\_black\_stops %>%   
 filter(county\_name!="NA")

Note that co\_county\_black\_stops now contains 64 rows:

# Prints contents of "co\_county\_black\_stops"  
co\_county\_black\_stops

## # A tibble: 64 × 3  
## # Rowwise: county\_name  
## county\_name black\_stops total\_stops  
## <chr> <int> <int>  
## 1 Adams County 1208 34350  
## 2 Alamosa County 43 4478  
## 3 Arapahoe County 1819 17520  
## 4 Archuleta County 28 5091  
## 5 Baca County 61 1511  
## 6 Bent County 46 1808  
## 7 Boulder County 192 13053  
## 8 Broomfield County 22 1095  
## 9 Chaffee County 37 6521  
## 10 Cheyenne County 38 1106  
## # … with 54 more rows

# 5 Defining an index of racial bias in traffic stops

Let’s briefly take stock of where we are. We started with a massive dataset (over 3,000,000 observations) of traffic patrol stops in the state of Colorado over the course of almost a decade. Over several steps, we have worked our way to a county-level dataset containing information on the total number of traffoc stops, and the number of those stops involving a Black motorist, over the course of the year 2010. That dataset looks something like this:

# Prints contents of "co\_county\_black\_stops"  
co\_county\_black\_stops

## # A tibble: 64 × 3  
## # Rowwise: county\_name  
## county\_name black\_stops total\_stops  
## <chr> <int> <int>  
## 1 Adams County 1208 34350  
## 2 Alamosa County 43 4478  
## 3 Arapahoe County 1819 17520  
## 4 Archuleta County 28 5091  
## 5 Baca County 61 1511  
## 6 Bent County 46 1808  
## 7 Boulder County 192 13053  
## 8 Broomfield County 22 1095  
## 9 Chaffee County 37 6521  
## 10 Cheyenne County 38 1106  
## # … with 54 more rows

Now, let’s think about how to use this information to develop a measure of the extent to which traffic police stops may have been driven by racial bias (whether conscious or unconscious) against Black motorists. We might expect that in a discrimination-free world, the share of traffic stops for a given racial group will reflect their share of the overall adult (over-17) population (we’ll consider the over-17 population as our baseline, since that is the demographic eligible to drive). For example, if the Black percentage of a given county’s adult population is 5%, we could form a simple baseline expectation that in a discrimination-free world where “driving while Black” is not effectively criminalized, the percentage of traffic stops involving Black motorists would not exceed 5%. To the extent that the percentage of traffic stops involving Black motorists **did** exceed 5%, we might assume a statistical pattern consistent with racial discrimination.

Based on this logic, a simple county-level indicator of anti-Black bias in traffic stops would simply be the difference between the percentage of traffic stops in a given county involving Black motorists, and the percentage of the county’s overall population that is Black:

**County-Level Traffic Stop Bias Index**= *(Percentage of County Traffic Stops Involving Black Motorists)*-*(Percentage of County’s Adult Population that is Black)*

As the value of this difference rises above 0, and we see higher positive values for the bias index, we might infer higher degrees of discrimination in traffic stops.

Of course, this is a very simple index, and sets aside many complexities (for example, commuting and driving patterns, among other things). However, despite its simplicity, the intuition behind the index is often used in the social science literature on the topic (Stelter et al., 2021); at least to a first approximation, therefore, this index will give us a meaningful way to identify counties in which the behavior of Traffic Patrol might merit greater public scrutiny, on account of disproportionately high stop-rates for the Black population.

Let’s create this index for Colorado counties in the year 2010. Note that we already have the data to compute the first part of the bias index (i.e. the percentage of traffic stops involving Black motorists); this is contained in co\_county\_black\_stops:

# prints contents of "co\_county\_black\_stops"  
co\_county\_black\_stops

## # A tibble: 64 × 3  
## # Rowwise: county\_name  
## county\_name black\_stops total\_stops  
## <chr> <int> <int>  
## 1 Adams County 1208 34350  
## 2 Alamosa County 43 4478  
## 3 Arapahoe County 1819 17520  
## 4 Archuleta County 28 5091  
## 5 Baca County 61 1511  
## 6 Bent County 46 1808  
## 7 Boulder County 192 13053  
## 8 Broomfield County 22 1095  
## 9 Chaffee County 37 6521  
## 10 Cheyenne County 38 1106  
## # … with 54 more rows

Let’s therefore turn to calculating the second part of the index: the percentage of each county’s population that is Black. We’ll carry out this calculation using demographic data from the 2010 decennial census.

## 5.1 Read in and join 2010 census data to co\_county\_black\_stops

First, let’s read in the census dataset that was provided to you at the start of workshop (please see the Appendix to see how the data was extracted) using the read\_csv function, and assign it to an object named co\_counties\_census\_2010:

# Reads in census data and assigns to object named "co\_counties\_census\_2010"  
co\_counties\_census\_2010<-read\_csv("co\_county\_decennial\_census.csv")

##   
## ── Column specification ───────────────────────────────────────────────────────────────────────  
## cols(  
## GEOID = col\_character(),  
## County = col\_character(),  
## total\_pop = col\_double(),  
## total\_black\_pop\_over17 = col\_double(),  
## total\_pop\_over17 = col\_double()  
## )

Let’s print the contents of this census dataset:

# Prints contents of "co\_counties\_census\_2010"  
co\_counties\_census\_2010

## # A tibble: 64 × 5  
## GEOID County total\_pop total\_black\_pop\_over17 total\_pop\_over17  
## <chr> <chr> <dbl> <dbl> <dbl>  
## 1 08023 Costilla County 3524 18 2788  
## 2 08025 Crowley County 5823 556 5034  
## 3 08027 Custer County 4255 37 3525  
## 4 08029 Delta County 30952 139 24101  
## 5 08031 Denver County 600158 45338 471392  
## 6 08035 Douglas County 285465 2447 198453  
## 7 08033 Dolores County 2064 4 1602  
## 8 08049 Grand County 14843 43 11825  
## 9 08039 Elbert County 23086 122 17232  
## 10 08041 El Paso County 622263 27280 459587  
## # … with 54 more rows

Note that it contains county-level information on the overall population (i.e. all age groups and all racial categories), the adult (over-17) population (for all racial groups), and the adult (over-17) Black population. We can use the latter two variables to compute the second part of our bias index.

## 5.2 Join census data to co\_county\_black\_stops

Before doing so, however, let’s join the census dataset we just viewed (co\_counties\_census\_2010) to the dataset containing information on county-level police stops (co\_county\_black\_stops); this will allow us to calculate the bias index using the information from the single dataset that results from this join.

The code below uses the full\_join function to join co\_counties\_census\_2010 (the second argument), to co\_county\_black\_stops (the first argument), using the “county\_name” column (from co\_county\_black\_stops) and the “County” column (from co\_counties\_census\_2010) as the join fields; it assigns the dataset which results from the join to a new object named co\_counties\_census\_trafficstops:

# Joins "co\_counties\_census\_2010" to "co\_counties\_black\_stops" using the "county\_name" and "County" fields as the join fields; assigns the product of the join to an object named "co\_counties\_census\_trafficstops"  
  
co\_counties\_census\_trafficstops<-full\_join(co\_county\_black\_stops, co\_counties\_census\_2010,  
 by=c("county\_name"="County"))

When we open co\_counties\_census\_trafficstops, we should see information from both of the constituent datasets (co\_county\_black\_stops and co\_counties\_census\_2010):

# Prints contents of "co\_counties\_census\_trafficstops"  
co\_counties\_census\_trafficstops

## # A tibble: 64 × 7  
## # Rowwise: county\_name  
## county\_name black\_stops total\_stops GEOID total\_pop total\_black\_pop\_… total\_pop\_over17  
## <chr> <int> <int> <chr> <dbl> <dbl> <dbl>  
## 1 Adams County 1208 34350 08001 441603 9396 315480  
## 2 Alamosa County 43 4478 08003 15445 142 11617  
## 3 Arapahoe County 1819 17520 08005 572003 40558 424679  
## 4 Archuleta County 28 5091 08007 12084 19 9676  
## 5 Baca County 61 1511 08009 3788 15 2974  
## 6 Bent County 46 1808 08011 6499 486 5403  
## 7 Boulder County 192 13053 08013 294567 1961 231813  
## 8 Broomfield County 22 1095 08014 55889 415 41237  
## 9 Chaffee County 37 6521 08015 17809 264 14821  
## 10 Cheyenne County 38 1106 08017 1836 4 1386  
## # … with 54 more rows

## 5.3 Define the variables that will be used in the bias index

Now that we have all the information in one dataset, let’s calculate the constituent parts of the bias index. The code below takes the current co\_counties\_census\_trafficstops object that we created above, and then uses the mutate function to create two new variables. The first variable, named “black\_stop\_pct”, is defined as the percentage of traffic stops that involved a Black motorist. The second variable, named “black\_pop\_pct”, is defined as the percentage of a given county’s adult population that is Black. The dataset that includes these two new variables is assigned back to co\_counties\_census\_trafficstops:

# Takes "co\_counties\_census\_trafficstops" and then creates two new variables; one new variable is named "black\_stop\_pct" (the Black percentage of traffic stops) and the other is "black\_pop\_pct" (the Black percentage of the over-17 population; assigns the updated dataset back to "co\_counties\_census\_trafficstops"  
  
co\_counties\_census\_trafficstops<-  
 co\_counties\_census\_trafficstops %>%   
 mutate(black\_stop\_pct=((black\_stops/total\_stops)\*100),  
 black\_pop\_pct=((total\_black\_pop\_over17/total\_pop\_over17)\*100))

Having created these two new variables (i.e. the components of the bias index we’ll calculate below) in co\_counties\_census\_trafficstops, let’s view the contents below:

# Prints contents of "co\_counties\_census\_trafficstops" object  
co\_counties\_census\_trafficstops

## # A tibble: 64 × 9  
## # Rowwise: county\_name  
## county\_name black\_stop\_pct black\_pop\_pct black\_stops total\_stops GEOID total\_pop  
## <chr> <dbl> <dbl> <int> <int> <chr> <dbl>  
## 1 Adams County 3.52 2.98 1208 34350 08001 441603  
## 2 Alamosa County 0.960 1.22 43 4478 08003 15445  
## 3 Arapahoe County 10.4 9.55 1819 17520 08005 572003  
## 4 Archuleta County 0.550 0.196 28 5091 08007 12084  
## 5 Baca County 4.04 0.504 61 1511 08009 3788  
## 6 Bent County 2.54 9.00 46 1808 08011 6499  
## 7 Boulder County 1.47 0.846 192 13053 08013 294567  
## 8 Broomfield County 2.01 1.01 22 1095 08014 55889  
## 9 Chaffee County 0.567 1.78 37 6521 08015 17809  
## 10 Cheyenne County 3.44 0.289 38 1106 08017 1836  
## # … with 54 more rows, and 2 more variables: total\_black\_pop\_over17 <dbl>,  
## # total\_pop\_over17 <dbl>

## 5.4 Calculate the bias index

Now that we have the two components of the traffic stop bias index in co\_counties\_census\_trafficstops (“black\_stop\_pct”, and “black\_pop\_pct”), all that is left for us to do is create a new variable, which we’ll call “bias\_index”, that is calculated by taking the difference between “black\_stop\_pct” and “black\_pop\_pct.” The code below creates this new “bias\_index” variable in co\_counties\_census\_trafficstops, and assigns the dataset with this new variable back to co\_counties\_census\_trafficstops:

# Creates new variable in "co\_counties\_census\_trafficstops" named "bias\_index" that is defined as the difference between the existing "black\_stop\_pct" and "black\_pop\_pct" variables  
co\_counties\_census\_trafficstops<-co\_counties\_census\_trafficstops %>%   
 mutate(bias\_index=black\_stop\_pct-black\_pop\_pct)

Note that the “bias\_index” variable is now in the dataset assigned to the co\_counties\_census\_trafficstops object:

# prints contents of "co\_counties\_census\_trafficstops"  
co\_counties\_census\_trafficstops

## # A tibble: 64 × 10  
## # Rowwise: county\_name  
## county\_name bias\_index black\_stop\_pct black\_pop\_pct black\_stops total\_stops GEOID total\_pop  
## <chr> <dbl> <dbl> <dbl> <int> <int> <chr> <dbl>  
## 1 Adams County 0.538 3.52 2.98 1208 34350 08001 441603  
## 2 Alamosa Cou… -0.262 0.960 1.22 43 4478 08003 15445  
## 3 Arapahoe Co… 0.832 10.4 9.55 1819 17520 08005 572003  
## 4 Archuleta C… 0.354 0.550 0.196 28 5091 08007 12084  
## 5 Baca County 3.53 4.04 0.504 61 1511 08009 3788  
## 6 Bent County -6.45 2.54 9.00 46 1808 08011 6499  
## 7 Boulder Cou… 0.625 1.47 0.846 192 13053 08013 294567  
## 8 Broomfield … 1.00 2.01 1.01 22 1095 08014 55889  
## 9 Chaffee Cou… -1.21 0.567 1.78 37 6521 08015 17809  
## 10 Cheyenne Co… 3.15 3.44 0.289 38 1106 08017 1836  
## # … with 54 more rows, and 2 more variables: total\_black\_pop\_over17 <dbl>,  
## # total\_pop\_over17 <dbl>

We can use this information to calculate an aggregated bias index for the state of Colorado as a whole in the year 2010:

# Calculates the total number of Black traffic stops in Colorado and assigns the value to an object named "colorado\_total\_black\_stops"  
colorado\_total\_black\_stops<-sum(co\_counties\_census\_trafficstops$black\_stops, na.rm=T)  
  
# Caclulates the total number of traffic stops in Colorado and assigns the value to an object named "colorado\_total\_stops"  
colorado\_total\_stops<-sum(co\_counties\_census\_trafficstops$total\_stops, na.rm=T)  
  
# Calculates the total adult Black population in CO and assigns the value to an object named "colorado\_17plus\_blackpopulation"  
colorado\_adult\_blackpopulation<-sum(co\_counties\_census\_trafficstops$total\_black\_pop\_over17, na.rm=T)  
  
# Calculates the total 17+ population in CO and assigns the value to an object named "colorado\_17\_plus\_overall"  
colorado\_adult\_overall<-sum(co\_counties\_census\_trafficstops$total\_pop\_over17, na.rm=T)  
  
# Calculates the percentage of CO traffic patrol stops and assigns the value to an object named "black\_stop\_pct\_CO"  
black\_stop\_pct\_CO<-(colorado\_total\_black\_stops/colorado\_total\_stops)\*100  
  
# Calculates the percentage of the Colorado 17+ population that is Black and assigns the value to an object named "black\_over17\_population\_pct\_CO"  
black\_over17\_population\_pct\_CO<-(colorado\_adult\_blackpopulation/colorado\_adult\_overall)\*100  
  
# Calculates the bias index for the state of Colorado as a whole, by computing the difference beween "black\_stop\_pct\_CO" and "black\_over17\_population\_pct\_CO"; assigns the result to an object named "CO\_bias\_index"  
CO\_bias\_index<-(black\_stop\_pct\_CO)-(black\_over17\_population\_pct\_CO)

Now, let’s print the value of CO\_bias\_index:

CO\_bias\_index

## [1] -1.112616

Interestingly, the value of the aggregate state-level bias index is negative; specifically, this result suggests that in 2010, the Black share of traffic stops in Colorado was about 1.11 percentage points less than the Black share of Colorado’s adult population. At first glance, this might suggest that “driving while black” was not criminalized in Colorado( during the year under consideration), according to our simple bias indicator. However, it might still be the case that there are specific areas in the state where racial bias in traffic stops **is** a problem, and focusing on an aggregated state-level measure of the bias index obscures possible micro-level variation in patterns of anti-Black bias with respect to traffic stops. Documenting this micro-level variation is an important task, since it would allow us to identify “problem areas” that might be excessively punitive towards Black drivers. The remainder of the lesson attempts to document this micro-geography of systemic bias in traffic stops (as measured by the index we’ve defined)

## 5.5 Compute summary statistics for the bias index

Recall that we have a measure of the bias index for each county in Colorado; a simple way to document the spatial distribution in the bias index would be to simply compute some basic summary statistics for the “bias\_index” variable in co\_counties\_census\_trafficstops, which will give us a sense of how the bias index varies across counties. We can generate these summary statistics using the describe function.

Below, the argument to the describe function, which reads co\_counties\_census\_trafficstops$bias\_index simply specifies that we want summary statistics for the “bias\_index” variable that is in the dataset assigned to the co\_counties\_census\_trafficstops data object:

# Produces summary statistics for "bias\_index" variable calculated above  
describe(co\_counties\_census\_trafficstops$bias\_index)

## vars n mean sd median trimmed mad min max range skew kurtosis se  
## X1 1 62 0.53 2.04 0.47 0.65 0.59 -9.84 4.26 14.1 -2.45 10.67 0.26

This table of summary statistics shows that the mean bias index across counties was 0.53 and the median value of the bias index was 0.47; this clearly suggests that even though the bias index for the state as a whole was negative, there are many counties with a ***positive*** bias index. In other words, many Colorado counties appear to have been stopping black motorists at disproportionately high rates relative to their share of the county population. Moreover, there is a fairly large range in the bias index with respect to counties; the county with the largest value for “bias\_index” had an index of 4.26 (i.e. the share of Black motorists stopped by the police was 4.26 percentage points higher than the share of Black county residents in the over-17 age demographic), while the county with the smallest value for the index had an index value of -9.84 (i.e. the share of Black motorists stopped by police was 9.84 percentage points lower than the share of Black county residents in the 17+ age demographic). These basic results suggest that a few outlier parts of the state where racial bias (as we are measuring it here) does not appear to be a problem are driving down the state’s overall bias index, despite the fact that in many areas of the state, Black motorists are indeed subject to disproportionately aggressive policing. One way to quickly explore this possibility further might be to create a quick visualization that conveys county-level variation in the bias index. In Section 5.6, we’ll visualize this county-level variation in the bias index on a simple graph.

## 5.6 Visualize county-level variation in the bias index using ggplot

There are many possible ways to visualize county-level variation in the “bias\_index” variable. Here, we’ll make a simple bar graph using ggplot, a popular visualization package that is part of the *tidyverse* suite of packages.

Before making this graph, let’s quickly make a new column in co\_counties\_census\_trafficstops, named “County”, that takes the information in the existing “county\_name” field (which takes the form of “ County”) and removes the part of the string that reads “County”. This will result in a cleaner-looking graph, since we can convey that our geographic units are counties in the graph’s main title.

The following code takes the dataset currently assigned to theco\_counties\_census\_trafficstops object, and then uses the mutate function to create a new field named “County”, which is populated by taking the existing “county\_name” column and using the str\_remove function to remove the part of the “county\_name” string that reads “County”. It then assigns the modified dataset back to the co\_counties\_census\_trafficstops object, which overwrites the previous version of the dataset:

co\_counties\_census\_trafficstops<-co\_counties\_census\_trafficstops %>%   
 mutate(County=str\_remove(county\_name, " County"))

Let’s confirm that the new “County” field has been successfully created:

# prints "co\_counties\_census\_trafficstops"  
co\_counties\_census\_trafficstops

## # A tibble: 64 × 11  
## # Rowwise: county\_name  
## county\_name County bias\_index black\_stop\_pct black\_pop\_pct black\_stops total\_stops GEOID  
## <chr> <chr> <dbl> <dbl> <dbl> <int> <int> <chr>  
## 1 Adams County Adams 0.538 3.52 2.98 1208 34350 08001  
## 2 Alamosa County Alamo… -0.262 0.960 1.22 43 4478 08003  
## 3 Arapahoe County Arapa… 0.832 10.4 9.55 1819 17520 08005  
## 4 Archuleta Coun… Archu… 0.354 0.550 0.196 28 5091 08007  
## 5 Baca County Baca 3.53 4.04 0.504 61 1511 08009  
## 6 Bent County Bent -6.45 2.54 9.00 46 1808 08011  
## 7 Boulder County Bould… 0.625 1.47 0.846 192 13053 08013  
## 8 Broomfield Cou… Broom… 1.00 2.01 1.01 22 1095 08014  
## 9 Chaffee County Chaff… -1.21 0.567 1.78 37 6521 08015  
## 10 Cheyenne County Cheye… 3.15 3.44 0.289 38 1106 08017  
## # … with 54 more rows, and 3 more variables: total\_pop <dbl>, total\_black\_pop\_over17 <dbl>,  
## # total\_pop\_over17 <dbl>

Now, we’re ready to use ggplot to make our bar graph.

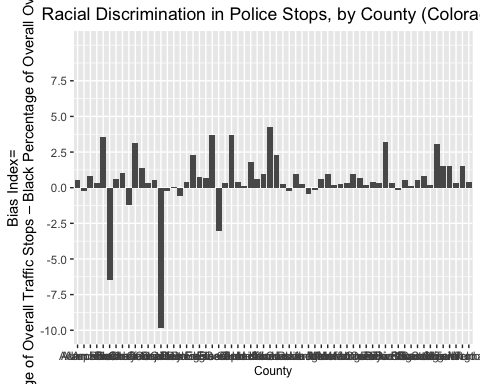
The following code takes co\_counties\_trafficstops, and then:

* Uses the drop\_na function to remove counties for which the “bias\_index” value has an “NA” value (there are two such counties), so that they do not show up on the graph.
* It then enters the ggplot environment by calling the ggplot function.
* It then uses the geom\_col function to indicate that the desired output is a bar graph. Within the geom\_col function, the expression that reads aes(x=County, y-bias\_index) specifies which variables we want to represent on the graph, and how we want to represent those variables (i.e. which variable do we want on the x- axis, and which variable on the y-axis?). The instructions used to translate the variables in a tabular dataset into a visual representation are known as an “aesthetic mapping”, which is abbreviated within the grammar of ggplot to aes.
* After specifiying that we want a bar graph based on the co\_counties\_census\_trafficstops object (with the “County” column mapped to the x-axis and the “bias\_index” column mapped to the y-axis), we use the labs function (short for “labels”) to designate the graph’s main title, and the labels for x and y axis.
* In ggplot, the theme function is a versatile function that helps to customize the appearance of a ggplot object; here, we set the axis.title.x argument equal to element\_text(size=9), which effectively sets the size of the x axis labels. In addition, we set the plot.title argument within the theme function equal to element\_text(hjust=0.5), which effectively center-justifies the plot’s main title. Next, we use the scale\_y\_continuous function to set the interval breaks of the y-axis; this range defined as a vector passed to the breaks argument, where each numeric element of the vector denotes a desired interval break.
* The expand\_limits function allows us to expand the range of either axis beyond the range established by the axis interval markers; here, by specifying y=c(-10,10) as an argument to the expand\_limits function, we effectively stretch out the y-axis range from -10 to 10, even though the highest Y-axis tick-mark is set at 7.5.
* Finally, we assign the plot that is created with this code to a new object named bias\_graph:

# Uses ggplot to create bar graph of "bias\_index" variable, with "bias\_index" on Y-axis and counties on X-axis  
bias\_graph<-  
 co\_counties\_census\_trafficstops %>%   
 drop\_na(bias\_index) %>%   
 ggplot()+  
 geom\_col(aes(x=County, y=bias\_index))+  
 labs(title="Racial Discrimination in Police Stops, by County (Colorado)", x="County", y="Bias Index=\nBlack Percentage of Overall Traffic Stops – Black Percentage of Overall Over-17 Population")+  
 theme(axis.title.x = element\_text(size = 9), plot.title=element\_text(hjust=0.5))+   
 scale\_y\_continuous(breaks=c(-10, -7.5, -5, -2.5, 0, 2.5, 5, 7.5))+  
 expand\_limits(y=c(-10,10))

Let’s print the contents of bias\_graph and see what the plot looks like; it will appear in the “plots” tab on the bottom-right of your R Studio interface.

# Prints contents of "bias\_graph"  
bias\_graph



As we can see, the graph does show that certain counties have values around zero, or well below zero on the bias index, which drives down the overall value for the state; but in many other counties, the value of the “bias\_index” variable is positive and substantively large.

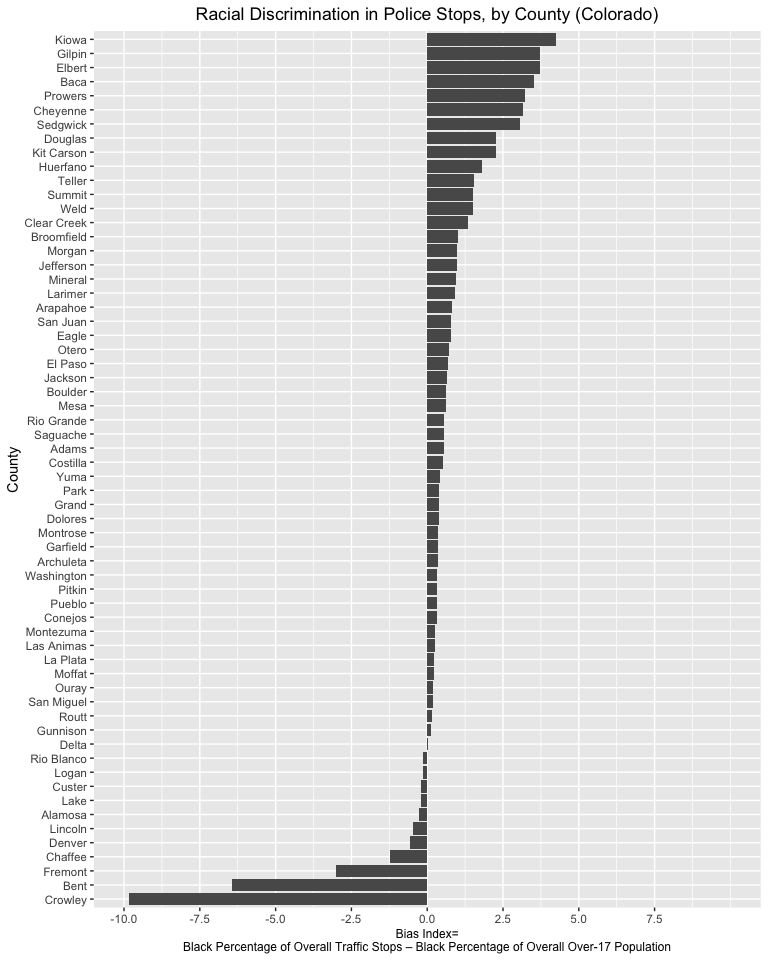
While the graph does convey valuable information, it’s a little bit confusing to look at; for example, the county names are squished together in a way that makes them unreadable.

A quick way to make the plot more readable is to invert the axes (such that counties are on the y-axis, and the bias index is arrayed along the x-axis). The code below is largely the same as the code in the previous code block. The differences are twofold, and contribute to a change in the map’s appearance. First, the argument to geom\_col looks a bit different. Instead of simply specifying (aes(x=County, y=bias\_index), we instead specify (aes(x=reorder(County, bias\_index), y=bias\_index). The reorder function is essentially specifying that we want the x-axis variable (“County”), to be arrayed in descending order with respect to “bias\_index”. Arraying the values in descending order makes for a graph that is easier to read. Second, we call the coord\_flip() function, which inverts the axes such that the x-axis and y-axis (specified as arguments to geom\_col) are inverted. We’ll assign this new plot to a new object named bias\_graph\_inverted:

# Uses ggplot to create bar graph of "bias\_index" variable, with "bias\_index" on X-axis and counties on y-axis; assigned to object named "bias\_graph\_inverted"  
bias\_graph\_inverted<-  
 co\_counties\_census\_trafficstops %>%   
 drop\_na(bias\_index) %>%   
 ggplot()+  
 geom\_col(aes(x=reorder(County, bias\_index), y=bias\_index))+  
 coord\_flip()+  
 labs(title="Racial Discrimination in Police Stops, by County (Colorado)", x="County", y="Bias Index=\nBlack Percentage of Overall Traffic Stops – Black Percentage of Overall Over-17 Population")+  
 theme(axis.title.x = element\_text(size = 9), plot.title=element\_text(hjust=0.5))+   
 scale\_y\_continuous(breaks=c(-10, -7.5, -5, -2.5, 0, 2.5, 5, 7.5))+  
 expand\_limits(y=c(-10,10))

Let’s view the revised plot:

# prints contents of "bias\_graph\_inverted"  
bias\_graph\_inverted



We can see that those relatively small changes produced a more readable and informative plot. We can get a clear sense of how the traffic stops bias index varies across Colorado counties, and identify the counties where policing practices appear excessively punitive towards Black drivers. The plot clearly conveys why focusing solely on aggregated statistics might be problematic: such measures have the potential to obscure more granular patterns of bias within jurisdictions.

# 6 Mapping the bias index

However, while the plot above gives us a sense of counties where the State Patrol practices may deserve greater scrutiny, it is difficult to contextualize that information without a clear sense of where these counties are located. For example, we might want to know whether there are clusters of counties with particularly high or low values for the bias index. And if so, what might explain such patterns?

In short, counties are spatial units, and creating a visualization where we can explicitly situate those those spatial units in their geographic context might prove even more informative than a plot such as the one developed in the previous section (where counties are not situated in their spatial context). In short, it would be useful to display the bias index on county-level map of Colorado, which will allow us to get a clearer sense of the granular spatial distribution of the bias index across the state. This section will walk through the process of creating such a map.

## 6.1 Read in and view the shapefile of CO counties

In order to visualize data on the bias index we created on a map of Colorado counties, we need to first load a spatial dataset of Colorado counties into R Studio. A spatial dataset is simply a dataset that has geographic information attached to it; this geographic information can be used to render the data as a map.

Let’s load in the spatial dataset of Colorado counties that was provided to you at the start of the workshop. In particular, the spatial dataset that was provided to you is stored as a shapefile, which is a commonly used file format to store spatial datasets. We can load a shapefile into R Studio using the st\_read function from the *sf* package. Below, we’ll load in our shapefile, and assign it to a new object named co\_counties\_shapefile. Note that a given shapefile is comprised of several different files with various extensions; make sure that all of these files are in your working directory. However, the file name passed to the st\_read function must have an “.shp” extension, as below:

# Reads in shapefile and assigns to object named "co\_counties\_shapefile"  
co\_counties\_shapefile<-st\_read("tl\_2019\_08\_county.shp")

## Reading layer `tl\_2019\_08\_county' from data source `/Users/adra7980/Documents/CU\_workshops/gis/data/tl\_2019\_08\_county/tl\_2019\_08\_county.shp' using driver `ESRI Shapefile'  
## Simple feature collection with 64 features and 17 fields  
## geometry type: MULTIPOLYGON  
## dimension: XY  
## bbox: xmin: -109.0602 ymin: 36.99245 xmax: -102.0415 ymax: 41.00344  
## geographic CRS: NAD83

Upon reading in the shapefile, you’ll notice that some metadata about the shapefile is printed to the console. We see that the shapefile has a geometry type of “multipolygon” (other possible geometry types are points and lines), that there are 64 rows and 17 columns in the dataset, and that its coordinate reference system is NAD83. Coordinate reference systems are important to consider if you plan to do spatial analysis that involves calculations with spatially defined attributes; since we don’t plan to do analysis of this nature (we’re only interested in using the shapefile as a way to visualize data), we can set this concept aside for the purpose of our workshop.

Now, let’s print the contents of co\_counties\_shapefile:

# Prints contents of "co\_counties\_shapefile"  
co\_counties\_shapefile

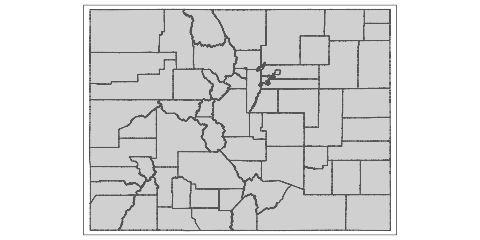
## Simple feature collection with 64 features and 17 fields  
## geometry type: MULTIPOLYGON  
## dimension: XY  
## bbox: xmin: -109.0602 ymin: 36.99245 xmax: -102.0415 ymax: 41.00344  
## geographic CRS: NAD83  
## First 10 features:  
## geometry STATEFP COUNTYFP COUNTYNS GEOID NAME NAMELSAD  
## 1 MULTIPOLYGON (((-106.8714 3... 08 109 00198170 08109 Saguache Saguache County  
## 2 MULTIPOLYGON (((-102.6521 4... 08 115 00198173 08115 Sedgwick Sedgwick County  
## 3 MULTIPOLYGON (((-102.5769 3... 08 017 00198124 08017 Cheyenne Cheyenne County  
## 4 MULTIPOLYGON (((-105.7969 3... 08 027 00198129 08027 Custer Custer County  
## 5 MULTIPOLYGON (((-108.2952 3... 08 067 00198148 08067 La Plata La Plata County  
## 6 MULTIPOLYGON (((-107.9751 3... 08 111 00198171 08111 San Juan San Juan County  
## 7 MULTIPOLYGON (((-106.9154 3... 08 097 00198164 08097 Pitkin Pitkin County  
## 8 MULTIPOLYGON (((-105.9751 3... 08 093 00198162 08093 Park Park County  
## 9 MULTIPOLYGON (((-106.0393 3... 08 003 00198117 08003 Alamosa Alamosa County  
## 10 MULTIPOLYGON (((-102.2111 3... 08 099 00198165 08099 Prowers Prowers County  
## LSAD CLASSFP MTFCC CSAFP CBSAFP METDIVFP FUNCSTAT ALAND AWATER INTPTLAT  
## 1 06 H1 G4020 <NA> <NA> <NA> A 8206547705 4454510 +38.0316514  
## 2 06 H1 G4020 <NA> <NA> <NA> A 1419419016 3530746 +40.8715679  
## 3 06 H1 G4020 <NA> <NA> <NA> A 4605713960 8166129 +38.8356456  
## 4 06 H1 G4020 <NA> <NA> <NA> A 1913031921 3364150 +38.1019955  
## 5 06 H1 G4020 <NA> 20420 <NA> A 4376255148 25642578 +37.2873673  
## 6 06 H1 G4020 <NA> <NA> <NA> A 1003660672 2035929 +37.7810492  
## 7 06 H1 G4020 233 24060 <NA> A 2514104907 6472577 +39.2175376  
## 8 06 H1 G4020 216 19740 <NA> A 5682182508 43519840 +39.1189141  
## 9 06 H1 G4020 <NA> <NA> <NA> A 1871465874 1847610 +37.5684423  
## 10 06 H1 G4020 <NA> <NA> <NA> A 4243429484 15345176 +37.9581814  
## INTPTLON  
## 1 -106.2346662  
## 2 -102.3553579  
## 3 -102.6017914  
## 4 -105.3735123  
## 5 -107.8397178  
## 6 -107.6702567  
## 7 -106.9161587  
## 8 -105.7176479  
## 9 -105.7880414  
## 10 -102.3921613

Notice that this looks very much like a typical dataset, one that we can also view in R Studio’s data viewer with View(co\_counties\_shapefile). The key element that makes this dataset different from a conventional dataset is contained in the “geometry” column. For each county, the geometry column contains a series of latitude/longitude pairs corresponding to that county’s borders in the “real world”. These lat/long pairs are processed by GIS software, and used to render a visual representation of a county’s geographic borders that reflects its “real-world” shape and location. When each row is rendered simultaneously, the result is a map of Colorado counties that can be used as a springboard for data visualization and analysis.

Within R Studio, we can leverage the geometry information in a shapefile to visually render its geographic attributes using functions from the *tmap* package. The code below uses the geometry information in co\_counties\_shapefile to render the Colorado county polygons as a map. First, we pass the name of the spatial object we’d like to map (co\_counties\_shapefile) to the tm\_shape function, and then use the tm\_polygons function to instruct the mapping utility that the spatial data in the “geometry” column is to be rendered as polygons:

## tmap mode set to plotting

# Uses "geometry" information in "co\_counties\_shapefile" to create map of CO counties  
tm\_shape(co\_counties\_shapefile)+  
 tm\_polygons()



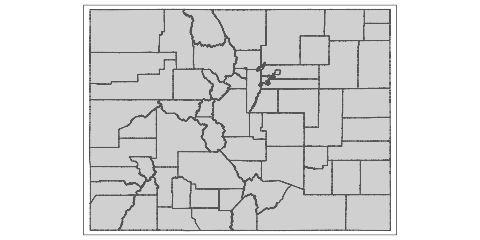
You should see a map that looks like this in the “Plots” tab on the bottom-right of your R Studio interface.

Below, we’ll assign this basic map, which was rendered from co\_counties\_shapefile ,to an object named co\_counties\_map:

# assigns map of CO polygons to new object named "co\_counties\_map"  
co\_counties\_map<-tm\_shape(co\_counties\_shapefile)+  
 tm\_polygons()

Now, whenever we want to retrieve the map, we can simply print the name of the object to which it has been assigned:

co\_counties\_map



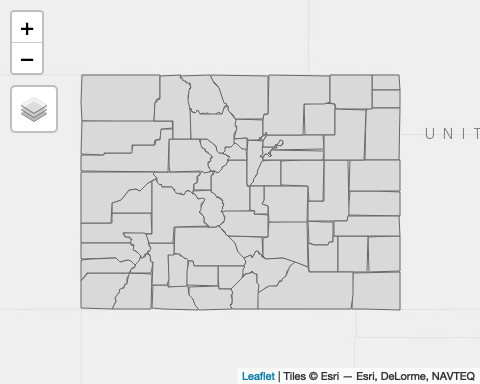
In using *tmap* to translate the geographic information in the “geometry” column of our spatial dataset of Colorado counties into a visual representation, we produced a static map; in other words, we cannot do things like pan around the map, or zoom in and out. However, if we want to render a dynamic map where such things are possible, we can simply use the tmap\_mode function to change the setting of the *tmap* environment to “view”, as below:

# Sets tmap mode to "view"  
tmap\_mode("view")

## tmap mode set to interactive viewing

Now, if we open the co\_counties\_map object that we created earlier, *tmap* will render a dynamic and interactive map:

# Prints "co\_counties\_map" as a web map  
co\_counties\_map



You will be able to view this interactive map in the “Viewer” tab on the bottom right of your R Studio interface. This interactive map is essentially a web map, and can easily be exported as an html file and embedded on a website (if we so choose).

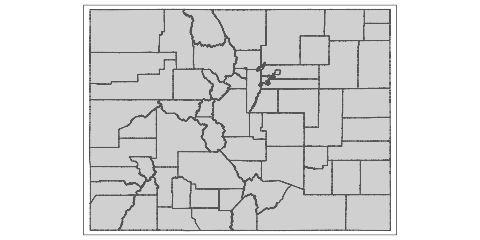
If we want to switch back to making static maps, we can switch back to “plot” mode using the same tmap\_mode function:

# Switches tmap mode back to "plot"  
tmap\_mode("plot")

## tmap mode set to plotting

Once we’re back in “plot” mode, *tmap* will render spatial objects as static maps. For instance, if we open co\_counties\_map again, it will render as a static map:

# prints "co\_counties\_map" as a static plot  
co\_counties\_map



As we work with spatial data and maps in R Studio using *tmap*, we can easily toggle back and forth between “view” and “plot” modes, depending on our desired outputs.

## 6.2 Join co\_counties\_census\_trafficstops to the shapefile of Colorado counties

Now that we’ve loaded and explored our spatial dataset of Colorado counties, let’s turn to the process of displaying the bias index on the map that we just rendered from our spatial dataset of CO counties.

In order to display our data of interest on the map of Colorado counties that we generated above, we must first join the data on the bias index to the spatial dataset; once the bias index data is in our spatial dataset, we can use *tmap* functions to render a map that displays this data on the county polygons.

We can implement the join between a spatial dataset and a tabular dataset in much the same was as we implemented a join between two tabular datasets above in Section 6.2 (between the traffic stops dataset and the census dataset).

Below, the first argument to the full\_join function is the name of our spatial object of Colorado counties; the second argument is the name of the object which contains the “bias\_index” data (which we want to merge into the shapefile). Finally, the third argument indicates that we want to join these datasets together using the field named “GEOID” (which is the same in both datasets), as the join field. We’ll assign the expanded spatial dataset that results from the join to a new object named county\_shapefile\_biasIndex:

# Joins "co\_counties\_census\_trafficstops" into "co\_counties\_shapefile" using GEOID as the join field; assigns the new joined dataset to a new object named "county\_shapefile\_biasIndex"  
county\_shapefile\_biasIndex<-  
 full\_join(co\_counties\_shapefile,   
 co\_counties\_census\_trafficstops, by="GEOID")

Now, when we open county\_shapefile\_biasIndex, we should see the “bias\_index” variable in the dataset, along with the “geometry” information needed to render a map of Colorado counties:

# prints contents of "county\_shapefile\_biasIndex"  
county\_shapefile\_biasIndex

## Simple feature collection with 64 features and 27 fields  
## geometry type: MULTIPOLYGON  
## dimension: XY  
## bbox: xmin: -109.0602 ymin: 36.99245 xmax: -102.0415 ymax: 41.00344  
## geographic CRS: NAD83  
## First 10 features:  
## NAME geometry bias\_index STATEFP COUNTYFP COUNTYNS GEOID  
## 1 Saguache MULTIPOLYGON (((-106.8714 3... 0.5591577 08 109 00198170 08109  
## 2 Sedgwick MULTIPOLYGON (((-102.6521 4... 3.0473002 08 115 00198173 08115  
## 3 Cheyenne MULTIPOLYGON (((-102.5769 3... 3.1472044 08 017 00198124 08017  
## 4 Custer MULTIPOLYGON (((-105.7969 3... -0.2021878 08 027 00198129 08027  
## 5 La Plata MULTIPOLYGON (((-108.2952 3... 0.2272495 08 067 00198148 08067  
## 6 San Juan MULTIPOLYGON (((-107.9751 3... 0.8000000 08 111 00198171 08111  
## 7 Pitkin MULTIPOLYGON (((-106.9154 3... 0.3336883 08 097 00198164 08097  
## 8 Park MULTIPOLYGON (((-105.9751 3... 0.3973331 08 093 00198162 08093  
## 9 Alamosa MULTIPOLYGON (((-106.0393 3... -0.2620964 08 003 00198117 08003  
## 10 Prowers MULTIPOLYGON (((-102.2111 3... 3.2319999 08 099 00198165 08099  
## NAMELSAD LSAD CLASSFP MTFCC CSAFP CBSAFP METDIVFP FUNCSTAT ALAND AWATER  
## 1 Saguache County 06 H1 G4020 <NA> <NA> <NA> A 8206547705 4454510  
## 2 Sedgwick County 06 H1 G4020 <NA> <NA> <NA> A 1419419016 3530746  
## 3 Cheyenne County 06 H1 G4020 <NA> <NA> <NA> A 4605713960 8166129  
## 4 Custer County 06 H1 G4020 <NA> <NA> <NA> A 1913031921 3364150  
## 5 La Plata County 06 H1 G4020 <NA> 20420 <NA> A 4376255148 25642578  
## 6 San Juan County 06 H1 G4020 <NA> <NA> <NA> A 1003660672 2035929  
## 7 Pitkin County 06 H1 G4020 233 24060 <NA> A 2514104907 6472577  
## 8 Park County 06 H1 G4020 216 19740 <NA> A 5682182508 43519840  
## 9 Alamosa County 06 H1 G4020 <NA> <NA> <NA> A 1871465874 1847610  
## 10 Prowers County 06 H1 G4020 <NA> <NA> <NA> A 4243429484 15345176  
## INTPTLAT INTPTLON county\_name County black\_stop\_pct black\_pop\_pct black\_stops  
## 1 +38.0316514 -106.2346662 Saguache County Saguache 0.7296607 0.1705030 20  
## 2 +40.8715679 -102.3553579 Sedgwick County Sedgwick 3.4120735 0.3647733 26  
## 3 +38.8356456 -102.6017914 Cheyenne County Cheyenne 3.4358047 0.2886003 38  
## 4 +38.1019955 -105.3735123 Custer County Custer 0.8474576 1.0496454 1  
## 5 +37.2873673 -107.8397178 La Plata County La Plata 0.6191950 0.3919455 70  
## 6 +37.7810492 -107.6702567 San Juan County San Juan 0.8000000 0.0000000 1  
## 7 +39.2175376 -106.9161587 Pitkin County Pitkin 0.8213552 0.4876670 4  
## 8 +39.1189141 -105.7176479 Park County Park 0.7943403 0.3970072 64  
## 9 +37.5684423 -105.7880414 Alamosa County Alamosa 0.9602501 1.2223466 43  
## 10 +37.9581814 -102.3921613 Prowers County Prowers 3.7458295 0.5138297 247  
## total\_stops total\_pop total\_black\_pop\_over17 total\_pop\_over17  
## 1 2741 6108 8 4692  
## 2 762 2379 7 1919  
## 3 1106 1836 4 1386  
## 4 118 4255 37 3525  
## 5 11305 51334 160 40822  
## 6 125 699 0 571  
## 7 487 17148 69 14149  
## 8 8057 16206 52 13098  
## 9 4478 15445 142 11617  
## 10 6594 12551 47 9147

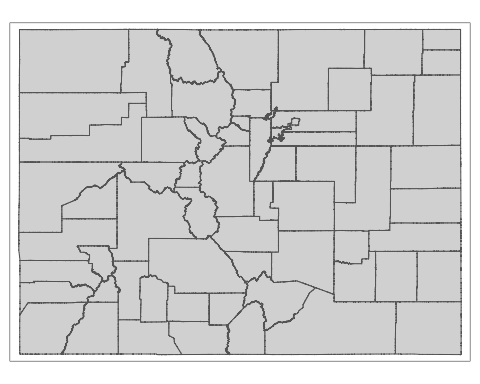
## 6.3 Display the bias index on a map of Colorado counties

At this point, with our “bias\_index” variable included in a spatially explicit dataset, we are ready to visualize this data on a map, and observe how our measure of bias in police stops varies geographically across counties.

### 6.3.1 Make a rough draft of a map

Let’s start by making the simplest possible map of “bias\_index”. Recall that earlier, we drew the Colorado county polygons with the following:

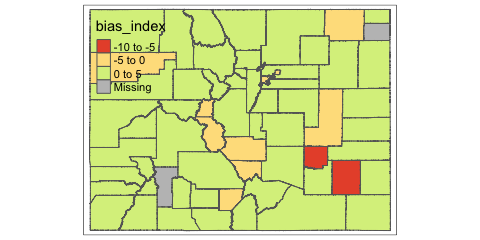
# Uses tmap to render Colorado polygons based on geometry information in "county\_shapefile\_biasIndex" object  
tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons()



Now, to display actual data from the dataset on the Colorado county polygons, we can simply pass an argument to the tm\_polygons function. In particular, we can specify the column that contains the data we want to represent on the map, using the col argument to the tm\_polygons function. In our case, the name of the column that contains the data we want to display on the map is “bias\_index”, so we will specify col="bias\_index" within the tm\_polygons function:

# Creates map of "bias\_index" variable from "county\_shapefile\_biasIndex" spatial object  
tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons(col="bias\_index")

## Variable(s) "bias\_index" contains positive and negative values, so midpoint is set to 0. Set midpoint = NA to show the full spectrum of the color palette.



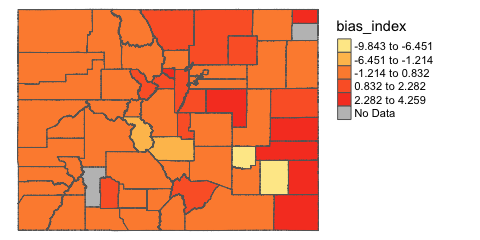
As you can see, this allowed us to display the “bias\_index” data on the map of Colorado counties. Admittedly, this map is very rough; important elements of the map, such as the legend interval breaks and the color scheme, were chosen arbitrarily by *tmap*, and these arbitrary settings hinder the ability of the map to effectively convey information about the spatial distribution of “bias\_index” across Colorado counties.

However, *tmap* allows us to customize our maps, and explicitly specify parameters that shape the map’s appearance. Let’s begin exploring some of these customization possibilities. In the code below, we still start out by passing county\_shapefile\_biasIndex to the tm\_shape function (so as to specify the spatial object we’d like to map), and passing col="bias\_index to the tm\_polygons function (as before).

However, we’ll begin customizing the map by passing additional arguments to the tm\_polygons function:

* With respect to the color scheme, we’ll designate a yellow-to-red color palette with palette="YlOrRd", and set 0 as the the point in the data distribution that corresponds to the midpoint of the palette (in other words, the the hypothetical county where “col\_bias” is exactly zero will be displayed in the color palette’s median color). For more information on color options in R (including color and palette codes), see the useful [R Color cheatsheet](https://www.nceas.ucsb.edu/sites/default/files/2020-04/colorPaletteCheatsheet.pdf).
* In the next three arguments to tm\_polygons, we’ll begin customizing the map’s legend. Setting textNA="No Data" specifies that the legend should label the category for counties with NA values on the bias index as “No Data”, rather than the default label, which is “Missing”. Setting n=5 establishes that we want to group our data into five distinct intervals, which means our legend will have five breaks. Finally, setting style="jenks" specifies that we want to use the jenks algorithm to decide where to place those legend breaks (or in other words, how to break up the data into five distinct intervals). For more For more information on the Jenks Natural Breaks Classification, as well as other data partition algorithms, see [here](https://pro.arcgis.com/en/pro-app/2.7/help/mapping/layer-properties/data-classification-methods.htm).
* Finally, we call the tm\_layout function, which allows us to customize the map’s layout; within this function, we set frame=FALSE (which removes the map’s frame, or bounding box) and legend.outside=TRUE (which places the legend outside the map’s domain, so as to not interfere with the display of counties).

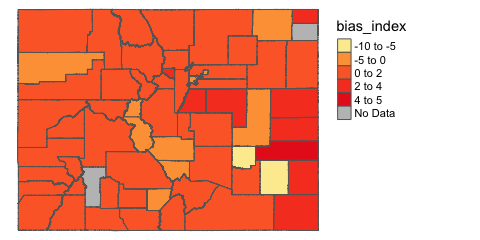
tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons(col="bias\_index",   
 palette="YlOrRd",   
 midpoint=0,  
 textNA="No Data",   
 n=5,  
 style="jenks")+  
 tm\_layout(frame=FALSE,   
 legend.outside=TRUE)

 This is starting to look better; in the next two subsections, we’ll explore how to further control the map’s appearance by setting custom breaks and custom color schemes.

### 6.3.2 Make a map with custom breaks

Let’s say that instead of using the Jenks (or some other algorithm) to partition our data into intervals, we want to set our own data intervals. Doing so might make sense here, especially since we want to clearly differentiate counties with a bias index less than or equal to zero from those that are above zero. We can specify our legend breaks in a numeric vector (i.e. a sequence of numbers) passed to the breaks argument within the tm\_polygons function. Below, we set breaks=c(-10,-5,0,2,4,5), which indicates that we want our intervals to be from -10 to -5; -5 to 0; 0 to 2; 2 to 4; and 4 to 5. Note that the “c” before the sequence of numbers bounded by parentheses is actually a function, which is used to indicate that the sequence of elements that follows must be interpreted as a vector. Also, note that we’ve removed the style="jenks" argument that we used above, since we’re using custom legend breaks (rather than breaks implemented by the jenks algorithm). Other than those changes, other elements of the code below are the same as that used in the previous code block.

tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons(col="bias\_index",   
 palette="YlOrRd",   
 midpoint=0,  
 textNA="No Data",  
 breaks=c(-10,-5, 0, 2, 4, 5))+  
 tm\_layout(frame=FALSE,   
 legend.outside=TRUE)



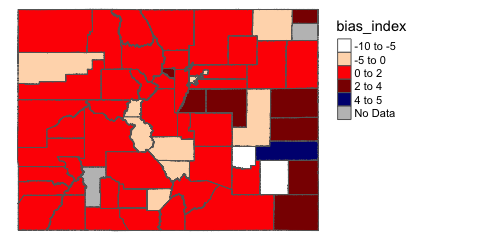
### 6.3.3 Make a map with custom colors

So far, we’ve been using a predefined color template (“YlOrRd”) to display the range of values on the map. While this color scheme might be a good start, it can sometimes be difficult to distinguish the colors on the map. One possible way to fix this might be to explore other possible predefined color schemes, and find one that makes colors easier to distinguish given the data we have. Another possibility is to specify our own colors for the intervals we want to map. In order to do this, let’s first first define a character vector, assigned to an object named my\_colors, that contains the colors we want to use (once again, a reminder color names are available on the R [Color Cheatsheet](https://www.nceas.ucsb.edu/sites/default/files/2020-04/colorPaletteCheatsheet.pdf)):

# defines vector of colors and assigns vector to an object named "my\_colors"  
my\_colors<-c("white", "peachpuff", "red1", "red4", "navy")

Now, we can pass this vector as an argument to the tm\_polygons function. Instead of setting palette=YlOrRd (as above), we instead set palette=my\_colors. The colors in the my\_colors vector are assigned to the numeric intervals in order; that is, the interval from -10 to -5 is assigned the color “white” (the first color in the vector), the interval from -5 to 0 is assigned the color “peachpuff” (the second color in the vector), the interval from 0 to 2 is assigned the color “red1” (the third color in the vector), and so on. Everything else in the code remains the same as in the previous section:

tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons(col="bias\_index",   
 palette=my\_colors,   
 textNA="No Data",  
 breaks=c(-10,-5, 0, 2, 4, 5))+  
 tm\_layout(frame=FALSE,   
 legend.outside=TRUE)



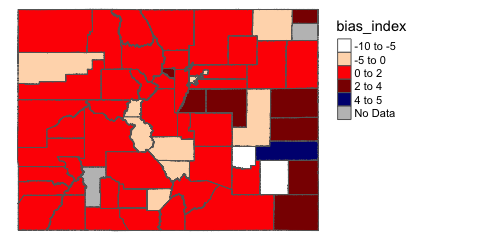
We can see that these colors are easier to distinguish, which makes it easier to quickly scan the map for relevant patterns.

It is also worth reminding ourselves that it’s possible to assign the maps we create using *tmap* to objects. For example, let’s assign the last map we created to a new object named traffic\_bias\_map\_continuous:

# Creates map showing variation in "bias\_index" across CO counties, and assigns the map to object named "traffic\_stop\_map\_continuous"  
traffic\_bias\_map\_continuous<-  
tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons(col="bias\_index",   
 palette=my\_colors,   
 textNA="No Data",   
 breaks=c(-10,-5, 0, 2, 4, 5))+  
 tm\_layout(frame=FALSE,   
 legend.outside=TRUE)

Now, we can bring up this map whenever we need it:

# Prints contents of "traffic\_stop\_map\_continuous"  
traffic\_bias\_map\_continuous



### 6.3.4 Make a categorical map

So far, we have been mapping the the bias\_index variable, which is a continuous variable. This has the advantage of allowing us to visualize the full extent of variation in the bias\_index variable. However, there are also other ways we might visualize the data. For example, we could transform bias\_index from a continuous numeric variable into a categorical variable, and visualize this categorical variable on a map.

More specifically, let’s say we want to use a map to clearly distinguish the counties where racial bias in traffic stops appears to be a problem (where “bias\_index”>0) and those counties in which it does NOT appear to be a problem (where “bias\_index”<=0). On the one hand, this would throw out useful information on the variation of “bias\_index”, but on the other hand, it could yield a more stark and focused map.

To build such a map, the first step is to create a new categorical variable, based on the continuous bias\_index variable, within county\_shapefile\_biasIndex. In the code below, we take the existing spatial dataset assigned to the county\_shapefile\_biasIndex object, and then use the mutate function to create a new variable named “apparent\_bias.” This new “apparent\_bias” variable is set to “Apparent Bias” for counties where “bias\_index”>0, and set to “No Apparent Bias” for all other counties (i.e. where the bias index is less than or equal to zero). This is accomplished using the ifelse function. The first argument to the ifelse function is a given condition (here bias\_index>0). The second argument specifies the value that the new “apparent\_bias” variable should take when that condition is true; the third argument specifies the value that the “apparent\_bias” variable should take when that condition is false. After creating and defining this new variable, we assign the changes back to the county\_shapefile\_biasIndex object, which overwrites the previous version of the dataset.

# Takes the existing dataset assigned to the "county\_shapefile\_biasIndex" object, and creates a new variable named "apparent\_bias"; this variable takes on the value "Apparent Bias" where the "bias\_index" variable is >0 and "No Apparent Bias" where it is less than or equal to zero; these changes are then assigned back to the "county\_shapefile\_biasIndex" object  
county\_shapefile\_biasIndex<-  
 county\_shapefile\_biasIndex %>%   
 mutate(apparent\_bias=ifelse(bias\_index>0,   
 "Apparent Bias",   
 "No Apparent Bias"))

Let’s take a look at what the new variable looks like:

# prints contents of "county\_shapefile\_biasIndex"  
county\_shapefile\_biasIndex

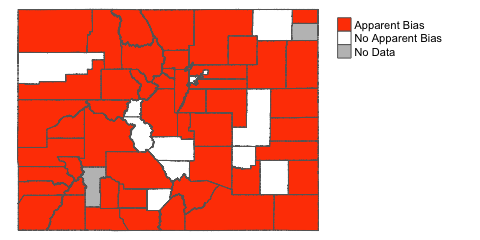
## Simple feature collection with 64 features and 28 fields  
## geometry type: MULTIPOLYGON  
## dimension: XY  
## bbox: xmin: -109.0602 ymin: 36.99245 xmax: -102.0415 ymax: 41.00344  
## geographic CRS: NAD83  
## First 10 features:  
## NAME geometry bias\_index apparent\_bias STATEFP COUNTYFP  
## 1 Saguache MULTIPOLYGON (((-106.8714 3... 0.5591577 Apparent Bias 08 109  
## 2 Sedgwick MULTIPOLYGON (((-102.6521 4... 3.0473002 Apparent Bias 08 115  
## 3 Cheyenne MULTIPOLYGON (((-102.5769 3... 3.1472044 Apparent Bias 08 017  
## 4 Custer MULTIPOLYGON (((-105.7969 3... -0.2021878 No Apparent Bias 08 027  
## 5 La Plata MULTIPOLYGON (((-108.2952 3... 0.2272495 Apparent Bias 08 067  
## 6 San Juan MULTIPOLYGON (((-107.9751 3... 0.8000000 Apparent Bias 08 111  
## 7 Pitkin MULTIPOLYGON (((-106.9154 3... 0.3336883 Apparent Bias 08 097  
## 8 Park MULTIPOLYGON (((-105.9751 3... 0.3973331 Apparent Bias 08 093  
## 9 Alamosa MULTIPOLYGON (((-106.0393 3... -0.2620964 No Apparent Bias 08 003  
## 10 Prowers MULTIPOLYGON (((-102.2111 3... 3.2319999 Apparent Bias 08 099  
## COUNTYNS GEOID NAMELSAD LSAD CLASSFP MTFCC CSAFP CBSAFP METDIVFP FUNCSTAT ALAND  
## 1 00198170 08109 Saguache County 06 H1 G4020 <NA> <NA> <NA> A 8206547705  
## 2 00198173 08115 Sedgwick County 06 H1 G4020 <NA> <NA> <NA> A 1419419016  
## 3 00198124 08017 Cheyenne County 06 H1 G4020 <NA> <NA> <NA> A 4605713960  
## 4 00198129 08027 Custer County 06 H1 G4020 <NA> <NA> <NA> A 1913031921  
## 5 00198148 08067 La Plata County 06 H1 G4020 <NA> 20420 <NA> A 4376255148  
## 6 00198171 08111 San Juan County 06 H1 G4020 <NA> <NA> <NA> A 1003660672  
## 7 00198164 08097 Pitkin County 06 H1 G4020 233 24060 <NA> A 2514104907  
## 8 00198162 08093 Park County 06 H1 G4020 216 19740 <NA> A 5682182508  
## 9 00198117 08003 Alamosa County 06 H1 G4020 <NA> <NA> <NA> A 1871465874  
## 10 00198165 08099 Prowers County 06 H1 G4020 <NA> <NA> <NA> A 4243429484  
## AWATER INTPTLAT INTPTLON county\_name County black\_stop\_pct black\_pop\_pct  
## 1 4454510 +38.0316514 -106.2346662 Saguache County Saguache 0.7296607 0.1705030  
## 2 3530746 +40.8715679 -102.3553579 Sedgwick County Sedgwick 3.4120735 0.3647733  
## 3 8166129 +38.8356456 -102.6017914 Cheyenne County Cheyenne 3.4358047 0.2886003  
## 4 3364150 +38.1019955 -105.3735123 Custer County Custer 0.8474576 1.0496454  
## 5 25642578 +37.2873673 -107.8397178 La Plata County La Plata 0.6191950 0.3919455  
## 6 2035929 +37.7810492 -107.6702567 San Juan County San Juan 0.8000000 0.0000000  
## 7 6472577 +39.2175376 -106.9161587 Pitkin County Pitkin 0.8213552 0.4876670  
## 8 43519840 +39.1189141 -105.7176479 Park County Park 0.7943403 0.3970072  
## 9 1847610 +37.5684423 -105.7880414 Alamosa County Alamosa 0.9602501 1.2223466  
## 10 15345176 +37.9581814 -102.3921613 Prowers County Prowers 3.7458295 0.5138297  
## black\_stops total\_stops total\_pop total\_black\_pop\_over17 total\_pop\_over17  
## 1 20 2741 6108 8 4692  
## 2 26 762 2379 7 1919  
## 3 38 1106 1836 4 1386  
## 4 1 118 4255 37 3525  
## 5 70 11305 51334 160 40822  
## 6 1 125 699 0 571  
## 7 4 487 17148 69 14149  
## 8 64 8057 16206 52 13098  
## 9 43 4478 15445 142 11617  
## 10 247 6594 12551 47 9147

Now that we have created this new categorical variable, let’s go ahead and create a map that displays it on our map of Colorado counties. The code below looks very similar to the code used to map the original “bias\_index” variable. The main difference is that instead of setting col=bias\_index, we set the col argument equal to "apparent\_bias (i.e. col="apparent\_bias"). Another difference worth pointing out is that we use a different, bipartite color scheme (since there are now only two categories to map); this color scheme is defined by the vector c("orangered1", "white"). Finally, in the previous map, the legend’s title was taken from the name of the column that was mapped; here, having this title would be redundant, so we can remove the legend title using title="". We’ll assign this map to a new object named traffic\_bias\_map\_categorical:

traffic\_bias\_map\_categorical<-  
 tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons(col="apparent\_bias",   
 title="",   
 palette=c("orangered1", "white"),   
 textNA="No Data")+  
 tm\_layout(frame=FALSE,   
 legend.outside=TRUE)

If we print the contents of traffic\_bias\_map\_categorical, we open a map that looks something like this:

traffic\_bias\_map\_categorical

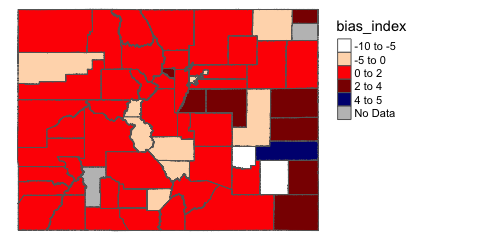


# 7 Refining and formatting the maps of the Colorado traffic-stop bias index

In the previous section, we created and refined two maps based on the index of racial bias in traffic stops we created earlier in the tutorial. The first map shows the spatial distribution of the continuous “bias\_index” variable across counties.

It looked like this:

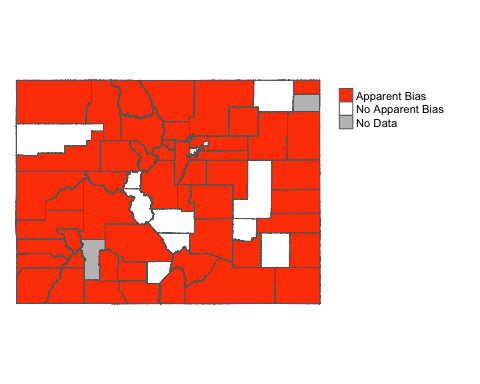
# prints "traffic\_bias\_map\_continuous"  
traffic\_bias\_map\_continuous



The second map created a new categorical variable based on “bias\_index”, and displayed these categories on a map.

It looked like this:

# prints "traffic\_bias\_map\_categorical"  
traffic\_bias\_map\_categorical



In this section, we’ll continue to refine and customize these maps (for example, by adding titles, map credits, county labels, and labels and titles for the legend).

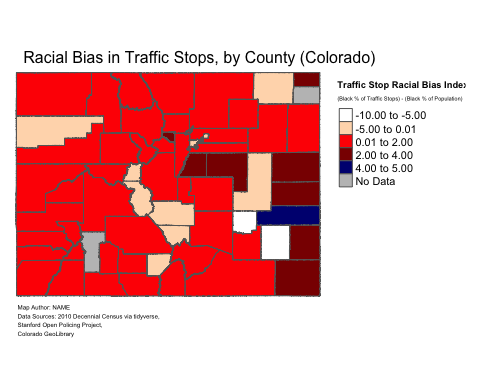
## 7.1 Refining the map of the continuous “bias\_index” variable

Let’s start by fine-tuning the map of the continuous “bias\_index” variable. As an exercise, read through the code below (without scrolling down to the map), and see if you can decipher what each line is doing, forming a mental picture of the in-progress map as you go. If you’re having trouble understanding something, look up the function’s documentation, which yoi can access by typing the name of the function, preceded by a question mark. For example, if we want to access the documentation for the tm\_polygons function, we can type ?tm\_polygons, which will bring up the function’s documentation (which provides a description of the various arguments to the function), in the “Help” tab on the bottom-right of the R Studio interface.

my\_colors<-c("white", "peachpuff", "red1", "red4", "navy")  
  
traffic\_bias\_map\_continuous<-  
tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons(col="bias\_index",   
 palette=my\_colors,  
 title="(Black % of Traffic Stops) - (Black % of Population)",  
 textNA="No Data",   
 breaks=c(-10,-5, 0.01, 2, 4, 5))+  
 tm\_layout(frame=FALSE,   
 legend.outside=TRUE,  
 legend.text.size=0.68,  
 legend.title.size=0.75,  
 title.size=0.75,  
 title.fontface = 2,  
 title="Traffic Stop Racial Bias Index",  
 main.title="Racial Bias in Traffic Stops, by County (Colorado)",  
 main.title.position=0.03,  
 main.title.size=1,  
 attr.outside=TRUE)+  
 tm\_credits("Map Author: NAME\nData Sources: 2010 Decennial Census via tidyverse,\nStanford Open Policing Project,\nColorado GeoLibrary ",   
 position=c(0.02,0.01),  
 size=0.38)

Having read through this code, let’s see what that the resulting map looks like:

# prints updated map assigned to "traffic\_stop\_map\_continuous" object  
traffic\_bias\_map\_continuous



Now, let’s unpack the various components of the code.

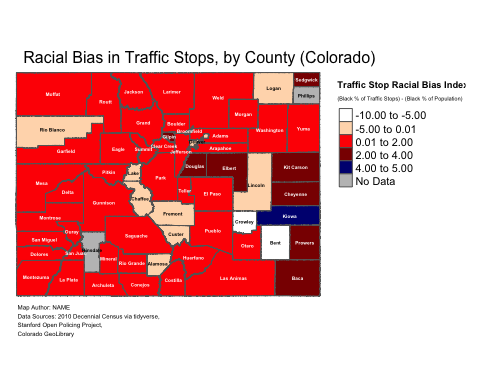
* The code begins by defining the my\_colors vector; this is the same color vector we used in Section 6.3.3, and is reproduced for convenience here.
* traffic\_bias\_map\_continuous<- indicates that the map created through the code on the right side of the assignment operator will be assigned to the object named traffic\_bias\_map\_continuous. This overwrites the previous version of the map assigned to this object.
* The first function that is called is tm\_shape; we pass the name of the object that contains the data we want to map to this function, which yields tm\_shape(county\_shapefile\_biasIndex).
* After specifying our dataset with tm\_shape, we call the tm\_polygons function. The tm\_polygons function takes several arguments, beginning with col="bias\_index (which specifies the column within county\_shapefile\_biasIndex that we want to map), and palette=my\_colors (which specifies that we want to use the colors from the my\_colors vector to represent different interval ranges for the “bias\_index” variable). Next, in title="(Black % of Traffic Stops) - (Black % of Population), we define a subtitle for the legend, so that viewers of the map can discern the intuition behind the bias index that is being mapped. Finally, textNA="No Data" defines the legend label for “NA” values, and breaks=c(-10,-5, 0.01, 2, 4, 5) sets the data intervals, which are communicated in the map’s legend. Notice that instead of setting one of the break points at 0, this version of the map uses 0.01; that is because the lower bound on data intervals is inclusive, while the upper bound is exclusive (i.e. a hypothetical county with a bias index of exactly 4.00 would be assigned the color that corresponds with the 4.00 to 5.00 legend interval, not the one corresponding to 2.00 to 4.00). As a result, a county with a bias index value of exactly zero would be grouped with counties with a positive value on the index, when it would make more sense to group such a county with counties with negative values on the index. Setting the break point slightly above 0 avoids a scenario where a county with an index value of exactly 0 is grouped together with counties with an index value greater than zero.
* The next *tmap* function that is called is the tm\_layout function, which allows us to specify various parameters that shape the map’s layout. Recall from before that frame=FALSE removes the map’s bounding box (which surrounds the area displayed on the map), while legend.outside=TRUE places the legend outside the map’s domain, so that it doesn’t overlap with any of the counties that are displayed. Next, legend.text.size=0.68 sets the size for legend text elements (that are not part of the legend’s title or subtitle), while legend.title.size=0.75 sets the size of the legend’s subtitle, and title.size=0.75 sets the size of the text in the legend’s main title. Note that when setting the size of text elements on your map, the best approach to use is usually trial and error; start with an arbitrary size, and then iterate from there. title="Traffic Stop Racial Bias Index" sets the text for legend’s main title, and title.fontface = 2 sets the legend’s main title text in boldface. main.title="Racial Bias in Traffic Stops, by County (Colorado)" sets the map’s main title, main.title.position=0.03 sets the position/justification of the title, and main.title.size=1 sets the size of the main title’s text. Finally, attr.outside=TRUE specifies that any map elements other than the legend (such as the map credits) are to be placed outside the map boundary.
* The tm\_credits function is a *tmap* function which allows us to add a credits section to the map, to convey information about things such as the name of the map author and the sources of the data used to create the map. In the code above, the first argument to the tm\_credits function is simply the text we would like to include in the credits; new lines are indicated by text that reads \n. Second, position=c(0.02,0.01) specifies the position of the credits section with respect to the map (the first element of the vector, 0.02, can be thought of as the x-coordinate, while the second element, 0.01, can be thought of as a y-coordinate); these coordinates place the credits below the map on the left-hand side. Finally, size=0.38 specifies the size of the text used in the credits section.

Notice that our map currently does not have labels for county names; these can easily be added using the tm\_text function that is part of the *tmap* package. Instead of rewriting all of the code from above, we can simply append the tm\_text function and its arguments to the map object we already created above (traffic\_bias\_map\_continuous). The code below takes the existing traffic\_bias\_map\_continous object, which we updated above, and adds a line of code that reads tm\_text("NAME", size=0.30, fontface=2). This labels the map’s polygons using information from the underlying dataset’s “NAME” field (which contains county names), using a text size of 0.3; the argument fontface=2 specifies that the county name labels are to be printed in boldface. No other changes or additions are made to traffic\_bias\_map\_continuous. The version of the map that is labelled is assigned to a new object, named traffic\_bias\_map\_continuous\_labeled.

traffic\_bias\_map\_continuous\_labeled<-  
 traffic\_bias\_map\_continuous+  
 tm\_text("NAME", size=0.30, fontface=2)

When we open traffic\_bias\_map\_continuous\_labeled, we can see the map with labels:

# prints contents of "traffic\_bias\_map\_continuous\_labeled"  
traffic\_bias\_map\_continuous\_labeled



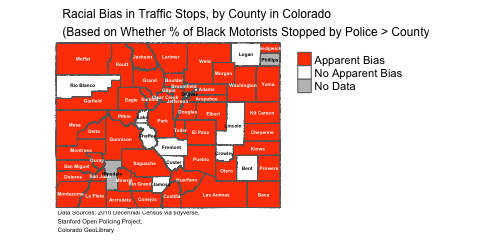
## 7.2 Refining the categorical map

The code below refines the categorical map we created in Section 6.3.4. The map is largely the same as the one created in that section, but adds and customizes the appearance of the map’s title using arguments to the tm\_layout function that we have already discussed in Section 7.1, and also adds a map credits section that is identical to the one we used in 7.1. The map is assigned to the traffic\_bias\_map\_categorical object, which overwrites the map we previously assigned to this object in Section 6.3.4.

traffic\_bias\_map\_categorical<-  
 tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons(col="apparent\_bias",   
 title="",   
 palette=c("orangered1", "white"),   
 textNA="No Data")+  
 tm\_layout(frame=FALSE,   
 legend.outside=TRUE,  
 main.title="Racial Bias in Traffic Stops, by County in Colorado\n(Based on Whether % of Black Motorists Stopped by Police > County's Adult Black Population %)",  
 main.title.position=0.03,  
 main.title.size=0.75,  
 attr.outside=TRUE)+  
 tm\_credits("Map Author: NAME\nData Sources: 2010 Decennial Census via tidyverse,\nStanford Open Policing Project,\nColorado GeoLibrary ", # Sets text for map credits  
 position=c(0.02,0.01), # Specifies location of map credits  
 size=0.38)+  
 tm\_text("NAME", size=0.30, fontface=2)

We can print the contents of traffic\_bias\_map\_categorical to view the updated map in the “Plots” window:

# prints contents of "traffic\_bias\_map\_categorical"  
traffic\_bias\_map\_categorical



## 7.3 Making a dynamic map of the bias index

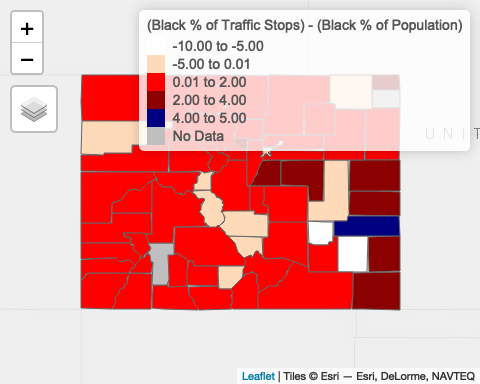
Recall from Section 6.1 that it is possible to render both static maps and interactive maps using the *tmap* package. Thus far, our focus has been on creating static maps to display county-level variation in the “bias\_index” variable (or the derivative “apparent\_bias” variable). It’s worth briefly reminding ourselves that we can easily turn these static maps into dynamic maps by changing the *tmap* mode to “view.” For example, let’s say we want to display the map assigned to traffic\_bias\_map\_continuous in a dynamic setting:

# Changes tmap mode to "view"  
tmap\_mode("view")

## tmap mode set to interactive viewing

# Prints "traffic\_bias\_map\_continuous" in View mode  
traffic\_bias\_map\_continuous

## Credits not supported in view mode.



# 8 Exporting maps

Once you have made a map and are satisfied with its appearance, it is straightforward to export the map from R Studio to a local directory on your computer, where you can then embed your map in papers, projects, presentations, or websites.

The easiest way to export maps is by using the tmap\_save function. This function allows users to specify the desired dimensions and resolution of the exported map, among other things. For our purposes, we’ll simply export the map assigned to traffic\_bias\_map\_continuous\_labeled using the default settings. Below, the first argument is the name of the map object we’d like to export; here, that is traffic\_bias\_map\_continuous\_labeled. The second argument is the desired file name and extension; it is possible to pick any number of extensions, such as PDF, jpeg, and png. Here, we’ll export the map as a png file.

tmap\_save(traffic\_bias\_map\_continuous\_labeled, "traffic\_bias\_map\_continuous\_labeled.png")

## Map saved to /Users/adra7980/Documents/git\_repositories/intro\_GIS/traffic\_bias\_map\_continuous\_labeled.png

## Resolution: 2448.943 by 1800.777 pixels

## Size: 8.163142 by 6.002591 inches (300 dpi)

Because we didn’t specify the path to a directory in the second argument, the map will be exported to our working directory.

Note that if we’d like to export a dynamic/interactive map, we can use the same code as above; the only difference is that we would use an “html” extension. The interactive map would then be exported to our working directory as an html file, which can then easily be embedded on a website.

# 9 Summary scripts

This section summarizes the code we have written over the course of the tutorial to map county-level variation in possible anti-Black bias in Colorado’s 2010 traffic patrol stops. Section 9.1 provides the script to clean and process the original dataset published by the Stanford Open Policing Project, and get that data into a form that is suitable for mapping. Section Section 9.2 provides the script to create a map of the continuous “bias\_index” variable, with counties labeled. Section 9.3 provides the script to create a map of a categorical variable that indicates whether a county’s value for “bias\_index” is greater than zero, or less than/equal to zero, and then make a map of this categorical variable (with counties labeled).

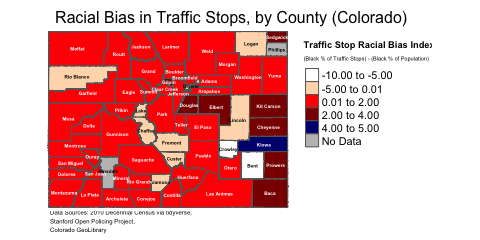
## 9.1 Summary script to prepare, clean, and process data for mapping

# Read in Stanford police data for Colorado and assign to object named "co\_traffic\_stops"   
co\_traffic\_stops<-read\_csv("co\_statewide\_2020\_04\_01.csv")  
  
# Create "Year" field based on existing "date" field  
co\_traffic\_stops<-co\_traffic\_stops %>%   
 mutate(Year=substr(co\_traffic\_stops$date, 1,4))# Filter 2010 observations and assign to a #   
# Filter 2010 observations and assign to a new object named "co\_traffic\_stops\_2010"  
co\_traffic\_stops\_2010<-co\_traffic\_stops %>% filter(Year==2010)  
  
# Compute county-level count of traffic stops by race and assign to object named "co\_county\_summary"  
co\_county\_summary<-co\_traffic\_stops\_2010 %>%   
 group\_by(county\_name) %>%   
 count(subject\_race)   
  
# Reshape the data so that the racial categories are transposed  
# from rows into columns and assign the result to an object named  
# "co\_county\_summary\_wide"  
co\_county\_summary\_wide<-co\_county\_summary %>%   
 pivot\_wider(names\_from=subject\_race, values\_from=n)  
  
  
# Creates a new column named "total\_stops" in "co\_county\_summary\_wide" that  
# contains information on the total number of stops for each county (across all racial categories)  
co\_county\_summary\_wide<-co\_county\_summary\_wide %>%   
 rowwise() %>%   
 mutate(total\_stops=sum(c\_across(where(is.integer)), na.rm=TRUE))  
  
# Selects "county\_name", "black", and "total\_stops" variables from "co\_county\_summary\_wide";  
# then renames the "black" variable to "black\_stops" for clarity; then removes counties that  
# are named "NA" due to an error in the dataset  
co\_county\_black\_stops<-co\_county\_summary\_wide %>%  
 select(county\_name, black, total\_stops) %>%   
 rename(black\_stops=black) %>%   
 filter(county\_name!="NA")  
  
# Read in the pre-prepared demographic data from the 2010 decennial census and assign  
# to an object named "co\_counties\_census\_2010"  
co\_counties\_census\_2010<-read\_csv("co\_county\_decennial\_census.csv")  
  
# Join "co\_counties\_census\_2010" to "co\_county\_black\_stops" and assign the result  
# to an object named "co\_counties\_census\_trafficstops"  
co\_counties\_census\_trafficstops<-full\_join(co\_county\_black\_stops, co\_counties\_census\_2010,  
 by=c("county\_name"="County"))  
  
# Use the information in "co\_counties\_census\_trafficstops" to define new variables that will be used  
# to compute the racial bias index: "black\_stop\_pct" (the black percentage of overall traffic stops within  
# a county) and "black\_pop\_pct" (the black percentage of the county's over-17 population)  
  
co\_counties\_census\_trafficstops<-  
 co\_counties\_census\_trafficstops %>%   
 mutate(black\_stop\_pct=((black\_stops/total\_stops)\*100),  
 black\_pop\_pct=((total\_black\_pop\_over17/total\_pop\_over17)\*100))  
  
# Calculate the bias index and include it as a new variable in "co\_counties\_census\_trafficstops"  
co\_counties\_census\_trafficstops<-co\_counties\_census\_trafficstops %>%   
 mutate(excess\_stops\_index=black\_stop\_pct-black\_pop\_pct)  
  
# Reads in Colorado county shapefile and assigns the shapefile to a new object named   
# "co\_counties\_shapefile"  
co\_counties\_shapefile<-st\_read("tl\_2019\_08\_county.shp")  
  
# Join "co\_counties\_census\_trafficstops" to "co\_counties\_shapefile" using "GEOID" as the join field; assign the result to a new object named "county\_shapefile\_biasIndex"  
county\_shapefile\_biasIndex<-full\_join(co\_counties\_shapefile, co\_counties\_census\_trafficstops, by="GEOID")

## 9.2 Summary script for map of continuous “bias\_index” variable

# make a map of the continuous "bias\_index" variable  
my\_colors<-c("white", "peachpuff", "red1", "red4", "navy") # create color vector  
traffic\_bias\_map\_continuous\_labeled<- # object assignment; assigns map to object named "traffic\_stop\_bias\_map"  
 tm\_shape(county\_shapefile\_biasIndex)+ # declares the spatial object that is the basis for the map  
 tm\_polygons(col="bias\_index", # declares variable containing data to be mapped  
 palette=my\_colors, # sets color scheme (based on "my\_colors" vector)  
 title="(Black % of Traffic Stops) - (Black % of Population)", # sets legend subtitle  
 textNA="No Data", # Sets the name of the legend label for "NA" values  
 n=5, # defines the number of intervals in the legend  
 breaks=c(-10,-5, 0.01, 2, 4, 5))+ # sets custom legend breaks  
 tm\_layout(frame=FALSE, # removes bounding box  
 legend.outside=TRUE, # sets legend outside (invisible) bounding box  
 legend.text.size=0.68, # sets size of legend text elements  
 legend.title.size=0.75, # sets size of legend main title  
 title="Traffic Stop Racial Bias Index", # sets legend's main title  
 title.size=0.75, # sets relative size of legend's subtitle  
 title.fontface = 2, # Makes legend title bold  
 main.title="Racial Bias in Traffic Stops, by County (Colorado)", # specifies main title of map  
 main.title.position=0.03, # specifies position of main title  
 main.title.size=1, # specifies size of main title  
 attr.outside=TRUE)+ # specifies that map credits should be placed outside bounding box  
 tm\_credits("Map Author: NAME\nData Sources: 2010 Decennial Census via tidyverse,\nStanford Open Policing Project,\nColorado GeoLibrary ", # Sets text for map credits  
 position=c(0.02,0.01), # Specifies location of map credits  
 size=0.38)+ # sets title of credits  
 tm\_text("NAME", size=0.30, fontface=2)

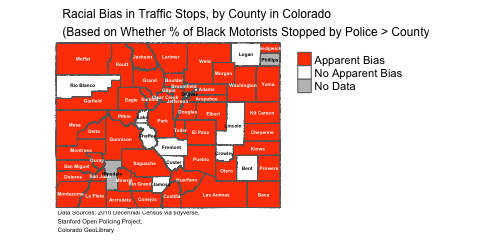
# Prints map  
traffic\_bias\_map\_continuous\_labeled



## 9.3 Summary script for categorical map

# Makes categorical variable  
county\_shapefile\_biasIndex<-  
 county\_shapefile\_biasIndex %>%   
 mutate(apparent\_bias=ifelse(bias\_index>0, "Apparent Bias", "No Apparent Bias"))  
  
# Makes categorical map and assigns to object named "traffic\_bias\_map\_categorical"  
traffic\_bias\_map\_categorical<-  
 tm\_shape(county\_shapefile\_biasIndex)+  
 tm\_polygons(col="apparent\_bias",   
 title="",   
 palette=c("orangered1", "white"),   
 textNA="No Data")+  
 tm\_layout(frame=FALSE,   
 legend.outside=TRUE,  
 main.title="Racial Bias in Traffic Stops, by County in Colorado\n(Based on Whether % of Black Motorists Stopped by Police > County's Adult Black Population %)",  
 main.title.position=0.03,  
 main.title.size=0.75,  
 attr.outside=TRUE)+  
 tm\_credits("Map Author: NAME\nData Sources: 2010 Decennial Census via tidyverse,\nStanford Open Policing Project,\nColorado GeoLibrary ", # Sets text for map credits  
 position=c(0.02,0.01), # Specifies location of map credits  
 size=0.38)+  
 tm\_text("NAME", size=0.30, fontface=2)

# prints categorical map  
traffic\_bias\_map\_categorical



# 10 Reflections and Discussion Questions

Now that we’ve created our maps, it’s worthwhile to reflect on the process and its outcomes. Consider the following:

* How might you change the maps we made? For example, would you change the color schemes, number of intervals, title, or any other features of their appearance? If so, why? See if you can change the code we developed to implement those changes.
* What do you think are the possible shortcomings of the “bias\_index” variable we created? Can you think of an alternative measure of racial bias in traffic stops that addresses these shortcoming? Could you create it with the data we already have? If not, what data would you need to collect?
* What spatial patterns (if any) do you notice in how the “bias\_index” is distributed across counties? Are there regions of the state where several counties with high values for the bias index are clustered? What might explain these patterns?
* What implications might these maps have for public policy?

# 11 Conclusion and Further Reading

If you’re interested in exploring projects resources that lie at the intersection of GIS, Black history, and the study of structural racism, ESRI’s [Racial Equity Hub](https://gis-for-racialequity.hub.arcgis.com/) is a good place to start.

If you’re interested in exploring a specific example of a large-scale interdisciplinary project on systemic racism that incorporates spatial analysis, check out the [Mapping Prejudice](https://mappingprejudice.umn.edu/) project from the University of Minnesota (Ehrman-Solberg et al, 2020). The data for that project is available [here](https://conservancy.umn.edu/handle/11299/217209).

If you’d like to read the paper by Pierson et al (2020) for which the police stop data was originally collected, it is available [here](https://5harad.com/papers/100M-stops.pdf). The Stanford project site also has a useful [tutorial](https://openpolicing.stanford.edu/tutorials/) that introduces the data and provides some guidance in analyzing it. The tutorial includes a discussion of more advanced ways to measure racial bias in stops than the one we used here, and a useful way to extend your knowledge might be to think about how to create and map those measures.

If you would like to further develop your spatial visualization and mapping skills, an excellent place to start is with the free and open-source book by Lovelace, Nowosad, and Muenchow (2021), entitled [*Geocomputation with R*](https://geocompr.robinlovelace.net/index.html). The book is more than an introduction to making maps; it’s a comprehensive guide to spatial analysis and Geographic Information Systems more generally.

# 12 References

Ehrman-Solberg, K. P. Petersen, M. Mills, K. Delegard, R. Mattke. 2020. Racial Covenants in Hennepin County. *University of Minnesota Data Repository.* <https://doi.org/10.13020/a88t-yb14>

Esri. n.d. GIS for Racial Equity: Racial Equity Hub. Accessed February 15, 2022. <https://gis-for-racialequity.hub.arcgis.com/>

Esri. n.d. Data Classification Methods. Accessed February 14, 2021. <https://pro.arcgis.com/en/pro-app/2.7/help/mapping/layer-properties/data-classification-methods.htm>

Frazier, M. “R Color Cheatsheet.” Accessed February 10, 2022. <https://www.nceas.ucsb.edu/sites/default/files/2020-04/colorPaletteCheatsheet.pdf>.

Lovelace, R, J. Nowosad, and J. Muenchow. 2021. *Geocomputation With R*. <https://geocompr.robinlovelace.net/>.

Pebesma, E. 2018. Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal 10*(1): 439-446. <https://doi.org/10.32614/RJ-2018-009>.

Pierson, E., C. Simoiu, J. Overgoor, S. Corbett-Davies, D.Jenson, A. Shoemaker, V. Ramachandran, P. Barghouty, C. Phillips, R. Shroff, and S. Goel. 2020. A large-scale analysis of racial disparities in police stops across the United States. *Nature Human Behaviour 4*: 736-745.

R For Social Scientists: Setup. 2018-2021. Data Carpentry. <https://datacarpentry.org/r-socialsci/setup.html>.

Stanford Open Policing Project. 2019. Open Policing Project Tutorial. Accessed February 15, 2022. <https://openpolicing.stanford.edu/tutorials/>.

Stelter, M., I. Essien, C. Sander, and J. Degner. 2021. Racial bias in police traffic stops: White residents’ county-level prejudice and Sterotypes are related to disproportionate stopping of Black drivers." *PsyArXiv Preprints.* <https://psyarxiv.com/djp8g/>.

Tennekes, M. 2018. tmap: Thematic Maps in R. *Journal of Statistical Software 84*(6): 1-39. <https://doi.org/10.18637/jss.v084.i06>.

U.S. Census Bureau. 2019. Colorado Counties Shapefile. <https://geo.colorado.edu/catalog/47540-5e712aeda3d91e0009f59fc7>.

Walker, K. and M. Herman. 2021. “tidycensus: Load US Census Boundary and Attribute Data as tidyverse and sf-Read Data Frames.” R Package version 1.1. <https://CRAN.R-project.org/package=tidycensus>

Wickham, H., M. Averick, J. Bryan, W. Chang, L.D. McGowan, R. Francois, G. Grolemund et al. 2019. "Welcome to the tidycerse. *Journal of Open Source Software 4*(43): 1686. <https://doi.org/10.21105/joss.01686>.

# (APPENDIX) Appendix

# 13 Using tidycensus to extract relevant census data

This section provides a script used to extract the census dataset that was read into R Studio in Section 5.1. To save time during a workshop, it is recommended to prepare the census dataset required to create the relevant index beforehand, and simply provide students with the relevant dataset. However, if you are looking for a way to extract the census dataset within R, the following script can be used as a guide.

To extract the census data required to create the bias index, load the tidyverse (which was used in the workshop tutorial) and the tidycensus package, which is an R package that allows users to pull Census Bureau data using the Census API (if tidycensus is not already installed, please install it using install.packages("tidycensus").

# Loads libraries required to extract census data  
library(tidycensus)  
library(tidyverse)

Before extracting data with tidycensus, you must acquire a Census API key from the Census Bureau website; once you apply for a key on the website, your key will be immediately emailed to the address you provide. Enter your census API key in R Studio with the following code, replacing the with your key:

census\_api\_key("INSERT HERE")

In the workshop, we were working with 2010 police stops data, so it made sense to pull demographic data from the 2010 decennial census (which would have been collected in 2009). The discussion below is therefore framed with respect to the 2010 decennial census; if you choose to use another census dataset to create the index, your code may look slightly different.

## 13.1 Step 1: Define your variables

First, we can generate a table that contains the various variables (and associated variable codes) for the 2010 decennial census by using the load\_variables function. The arguments to the function below (2010, "sf1") indicate that we want to extract variable names and codes for the 2010 decennial census. We assign the table to an object named decennial\_2010\_variables, which allows us to easily view the table by using the View function, and refer back to it whenever needed.

# Variable list for 2010 Decennial  
decennial\_2010\_variables<-load\_variables(2010, "sf1")

Based on the information in decennial\_2010\_variables, we can identify the variable codes for the variables we want to extract. Then, we define a vector, assigned to an object named my\_vars that assigns the variable codes to descriptive names; these descriptive names will be used as column names in the dataset returned by the census API call, while the variable codes will be used by tidycensus to populate the respective fields with the desired data.

For the purpose of defining the “bias\_index” variable, recall that the two key variables we need are the over-17 total population, and the over-17 Black population (counted at the county level). There is no separate category in the census dataset for these measures, so we must derive them based on the data that is available.

Given the available data, to calculate the total over-17 population, we must extract data for the male under-5 population, the male 5 to 9 population, the male 10 to 14 population, and the male 15 to 17 population, and analogous measures for the female population. Subtracting these values from the total overall population (among all age groups) will yield a value for the total over-17 population.

To calculate the Black over-17 population, we will extract a variable that defines the total Black population, and a series of variables that measure the Black population for different demographic (sex/age) combinations under 17 years old; subtracting the sum of the latter variables from the total Black population yields a measure of the Black over-17 population.

The code below extracts all the variables needed to carry out these calculation:

# Define and name variables for census API call  
  
my\_vars<-c(total\_pop="P001001",  
 totalpop\_men\_u5="P012003",  
 totalpop\_men\_5to9="P012004",  
 totalpop\_men\_10to14="P012005",  
 totalpop\_men\_15to17="P012006",  
 totalpop\_women\_u5="P012027",  
 totalpop\_women\_5to9="P012028",  
 totalpop\_women\_10to14="P012029",  
 totalpop\_women\_15to17="P012030",  
 black\_totalpop="PCT012B001",  
 black\_men\_u1="PCT012B003",  
 black\_men\_1="PCT012B004",  
 black\_men\_2="PCT012B005",  
 black\_men\_3="PCT012B006",  
 black\_men\_4="PCT012B007",  
 black\_men\_5="PCT012B008",  
 black\_men\_6="PCT012B009",  
 black\_men\_7="PCT012B010",  
 black\_men\_8="PCT012B011",  
 black\_men\_9="PCT012B012",  
 black\_men\_10="PCT012B013",  
 black\_men\_11="PCT012B014",  
 black\_men\_12="PCT012B015",  
 black\_men\_13="PCT012B016",  
 black\_men\_14="PCT012B017",  
 black\_men\_15="PCT012B018",  
 black\_men\_16="PCT012B019",  
 black\_men\_17="PCT012B020",  
 black\_women\_u1="PCT012B107",  
 black\_women\_1="PCT012B108",  
 black\_women\_2="PCT012B109",  
 black\_women\_3="PCT012B110",  
 black\_women\_4="PCT012B111",  
 black\_women\_5="PCT012B112",  
 black\_women\_6="PCT012B113",  
 black\_women\_7="PCT012B114",  
 black\_women\_8="PCT012B115",  
 black\_women\_9="PCT012B116",  
 black\_women\_10="PCT012B117",  
 black\_women\_11="PCT012B118",  
 black\_women\_12="PCT012B119",  
 black\_women\_13="PCT012B120",  
 black\_women\_14="PCT012B121",  
 black\_women\_15="PCT012B122",  
 black\_women\_16="PCT012B123",  
 black\_women\_17="PCT012B124")

## 13.2 Step 2: Extract the variables using tidycensus

Now that we have a vector of the variables we want to extract (along with descriptive names for those variables), we will use the get\_decennial function from *tidycensus* to extract these variables from the 2010 decennial census. Several arguments are passed to the get\_decennial function below:

* geography="county" specifies that we want the census data to be provided at the county level
* variables=my\_vars specifies the variables we want to extract, and the names they are to be given in the dataset; this information is contained in the my\_vars vector defined above
* state=CO specifies the state for which we want to extract the data; this argument, together with the geography="county" argument, means that *tidycensus* will extract the specified data in my\_vars at the county level for the state of Colorado.
* survey="sf1" indicates which census dataset we would like to query; here sf1 (short for Summary File 1), indicates we are referring to the decennial census (as opposed, for example, to the American Community Survey)
* output=wide indicates that we want the dataset with the extracted variables to be in “wide” format, wherein each variable is assigned to its own column.
* year=2010 indicates that we are interested in data from 2010. Combined with survey="sf1", this will extract the 2010 decennial census data.

Finally, we’ll assign the extracted dataset to an object named co\_counties\_race:

# Issue call to Census API  
co\_counties\_race<-  
get\_decennial(  
 geography="county",   
 variables=my\_vars,  
 state="CO",  
 survey="sf1",  
 output="wide",  
 year=2010)

## Getting data from the 2010 decennial Census

## Using FIPS code '08' for state 'CO'

## Using Census Summary File 1

We can print the first few lines of the dataset to the console to view its structure, and ensure that everything looks in order:

# prints contents of "co\_counties\_race"  
co\_counties\_race

## # A tibble: 64 × 48  
## GEOID NAME total\_pop totalpop\_men\_u5 totalpop\_men\_5t… totalpop\_men\_10… totalpop\_men\_15…  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 08023 Costilla… 3524 99 102 107 95  
## 2 08025 Crowley … 5823 88 110 136 81  
## 3 08027 Custer C… 4255 62 95 113 85  
## 4 08029 Delta Co… 30952 887 913 1042 680  
## 5 08031 Denver C… 600158 22252 18894 15319 8920  
## 6 08035 Douglas … 285465 11278 13587 12664 6913  
## 7 08033 Dolores … 2064 58 63 71 40  
## 8 08049 Grand Co… 14843 416 421 431 261  
## 9 08039 Elbert C… 23086 594 790 976 611  
## 10 08041 El Paso … 622263 23152 23050 23252 14097  
## # … with 54 more rows, and 41 more variables: totalpop\_women\_u5 <dbl>,  
## # totalpop\_women\_5to9 <dbl>, totalpop\_women\_10to14 <dbl>, totalpop\_women\_15to17 <dbl>,  
## # black\_totalpop <dbl>, black\_men\_u1 <dbl>, black\_men\_1 <dbl>, black\_men\_2 <dbl>,  
## # black\_men\_3 <dbl>, black\_men\_4 <dbl>, black\_men\_5 <dbl>, black\_men\_6 <dbl>,  
## # black\_men\_7 <dbl>, black\_men\_8 <dbl>, black\_men\_9 <dbl>, black\_men\_10 <dbl>,  
## # black\_men\_11 <dbl>, black\_men\_12 <dbl>, black\_men\_13 <dbl>, black\_men\_14 <dbl>,  
## # black\_men\_15 <dbl>, black\_men\_16 <dbl>, black\_men\_17 <dbl>, black\_women\_u1 <dbl>, …

As always, it is also possible to view the dataset in the R Studio data viewer by running View(co\_counties\_race).

## 13.3 Step 3: Clean the tidycensus dataset

Having extracted the dataset, you may want to tidy it up depending on your needs and preferences. For example, the “NAME” field includes the name of the state, which is not really necessary here since there are only observations from Colorado in the dataset. The code below removes the state name from the NAME field, and updates the co\_counties\_race object with this change:

# Remove state name from name field  
co\_counties\_race<-co\_counties\_race %>%   
 separate(col=NAME, c("County", "x"), sep=",") %>%   
 select(-x)

# Prints contents of "co\_counties\_race"  
co\_counties\_race

## # A tibble: 64 × 48  
## GEOID County total\_pop totalpop\_men\_u5 totalpop\_men\_5t… totalpop\_men\_10… totalpop\_men\_15…  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 08023 Costilla… 3524 99 102 107 95  
## 2 08025 Crowley … 5823 88 110 136 81  
## 3 08027 Custer C… 4255 62 95 113 85  
## 4 08029 Delta Co… 30952 887 913 1042 680  
## 5 08031 Denver C… 600158 22252 18894 15319 8920  
## 6 08035 Douglas … 285465 11278 13587 12664 6913  
## 7 08033 Dolores … 2064 58 63 71 40  
## 8 08049 Grand Co… 14843 416 421 431 261  
## 9 08039 Elbert C… 23086 594 790 976 611  
## 10 08041 El Paso … 622263 23152 23050 23252 14097  
## # … with 54 more rows, and 41 more variables: totalpop\_women\_u5 <dbl>,  
## # totalpop\_women\_5to9 <dbl>, totalpop\_women\_10to14 <dbl>, totalpop\_women\_15to17 <dbl>,  
## # black\_totalpop <dbl>, black\_men\_u1 <dbl>, black\_men\_1 <dbl>, black\_men\_2 <dbl>,  
## # black\_men\_3 <dbl>, black\_men\_4 <dbl>, black\_men\_5 <dbl>, black\_men\_6 <dbl>,  
## # black\_men\_7 <dbl>, black\_men\_8 <dbl>, black\_men\_9 <dbl>, black\_men\_10 <dbl>,  
## # black\_men\_11 <dbl>, black\_men\_12 <dbl>, black\_men\_13 <dbl>, black\_men\_14 <dbl>,  
## # black\_men\_15 <dbl>, black\_men\_16 <dbl>, black\_men\_17 <dbl>, black\_women\_u1 <dbl>, …

## 13.4 Step 4: Define new variables

Now that we have a cleaned dataset with all our necessary variables, we can use these variables to generate the demographic variables needed to calculate the bias index. First, the code below defines a new variable, called “total\_pop\_over17”, that is calculated by subtracting the total population that is 17 and under from the total overall population:

# Create variable for total over-17 population  
co\_counties\_race<-  
 co\_counties\_race %>%   
 mutate(total\_pop\_over17=total\_pop-totalpop\_men\_u5-totalpop\_men\_5to9-  
 totalpop\_men\_10to14-totalpop\_men\_15to17-totalpop\_women\_u5-  
 totalpop\_women\_5to9-totalpop\_women\_10to14-totalpop\_women\_15to17)

Then, we create a new variable named “total\_black\_pop\_over17”, which is defined by subtracting the total Black population that is 17 and under from the total Black population:

# Create variable for total over--17 black population  
co\_counties\_race<-  
 co\_counties\_race %>%   
 mutate(total\_black\_pop\_over17=  
 black\_totalpop-black\_men\_u1-black\_men\_1-  
 black\_men\_2-black\_men\_3-black\_men\_4-black\_men\_5-black\_men\_6-  
 black\_men\_7-black\_men\_8-black\_men\_9-black\_men\_10-black\_men\_11-  
 black\_men\_12-black\_men\_13-black\_men\_14-black\_men\_15-black\_men\_16-  
 black\_men\_17-black\_women\_u1-black\_women\_1-black\_women\_2-black\_women\_3-  
 black\_women\_4-black\_women\_5-black\_women\_6-black\_women\_7-black\_women\_8-  
 black\_women\_9-black\_women\_10-black\_women\_11-black\_women\_12-  
 black\_women\_13-black\_women\_14-black\_women\_15-black\_women\_16-  
 black\_women\_17)

## 13.5 Step 5: Finalize and export the dataset

Now that we have our two key variables defined, let’s clean up the dataset by removing the variables we no longer need, and only keeping the variables necessary to create the bias index. Below, we take the existing dataset assigned to co\_counties\_race, and select the “GEOID” and “County” variables (which serve as ID variables), “total\_black\_pop\_over17” and “total\_pop\_over17” (which are used to compute the bias index), and “total\_pop” (which is not necessary to create “bias\_index”, but which could prove useful in exploring alternate ways of defining a bias index than the one implemented in the tutorial). The new dataset is assigned to an object named co\_counties\_census\_2010:

#Clean data by select relevant variables for analysis, and assign selection to new object named "co\_counties\_census\_2010"  
co\_counties\_census\_2010<-  
 co\_counties\_race %>%   
 select(GEOID, County, total\_pop, total\_black\_pop\_over17, total\_pop\_over17)

Let’s view the dataset’s contents:

# prints contents of "co\_counties\_census\_2010"  
co\_counties\_census\_2010

## # A tibble: 64 × 5  
## GEOID County total\_pop total\_black\_pop\_over17 total\_pop\_over17  
## <chr> <chr> <dbl> <dbl> <dbl>  
## 1 08023 Costilla County 3524 18 2788  
## 2 08025 Crowley County 5823 556 5034  
## 3 08027 Custer County 4255 37 3525  
## 4 08029 Delta County 30952 139 24101  
## 5 08031 Denver County 600158 45338 471392  
## 6 08035 Douglas County 285465 2447 198453  
## 7 08033 Dolores County 2064 4 1602  
## 8 08049 Grand County 14843 43 11825  
## 9 08039 Elbert County 23086 122 17232  
## 10 08041 El Paso County 622263 27280 459587  
## # … with 54 more rows

At this point, we now have the census data used in the tutorial. This data was exported from R Studio, and provided to workshop participants as a CSV file that was included in the workshop materials.

To export the data, use the write\_csv function; below, the first argument is the name of the object which contains the dataset to be exported, and the second argument is the desired file name. The data is exported to the current working directory, and can subsequently be opened on your spreadsheet software of choice as a CSV file.

# Exports the data  
write\_csv(co\_counties\_census\_2010, "co\_counties\_census\_2010.csv")