Jacob Kaplan

Crime by the Numbers: A Criminologist's Guide to R

To my love Kristina, and our puppies Peanut and Moose.

Contents

(PART) Introduction

Preface

This book introduces the programming language R and is meant for undergrads or graduate students studying criminology. R is a programming language that is well-suited to the type of work frequently done in criminology - taking messy data and turning it into useful information. While R is a useful tool for many fields of study, this book focuses on the skills criminologists should know and uses crime data for the example data sets.

For this book you should have the latest version of R[1] installed and be running it through RStudio Desktop (The free version)[2]. We'll get into detail on what R and RStudio are soon but please have them both installed to be able to follow along with each chapter. While you must install both, you only ever need to open RStudio. While R is the actual programming language, RStudio is program that makes it a lot easier to interact with R than opening up the R application itself.[3] I highly recommend following along with the code for each lesson and then trying to use the lessons learned on a data set that you are interested in.

Why learn to program?

With the exception of some more advanced techniques like scraping data from websites or from PDFs, nearly everything we do here can be done through Excel, a software you're probably more familiar with. The basic steps for research projects are generally:

1. Open up a data set - which frequently comes as an Excel file!
2. Change some values - misspellings or too specific categories for our purposes are very common in crime data
3. Delete some values - such as states you won't be studying
4. Make some graphs
5. Calculate some values - such as number of crimes per year

[1] https://cloud.r-project.org/
[2] https://www.rstudio.com/products/rstudio/download/
[3] This is formally known as an "integrated development environment" or an IDE.

6. Sometimes do a statistical analysis depending on the type of project
7. Write up what you find

R can do all of this but why should you want (or have) to learn an entirely new skill just to do something you can already do? R is useful for two main reasons: scale and reproducibility.

Scale

If you do a one-off project in your career such as downloading some data and making a graph out of it, it makes sense to stick with software like Excel. The cost (in time and effort) of learning R is certainly not worth it for a single (or even several) project - even one perfectly suited for using R. R (and many programming languages more generally, such as Python) has its strength in doing something fairly simple many times. For example, it may be quicker to download one file yourself than it is to write the code in R to download that file. But when it comes to downloading hundreds of files, writing the R code becomes very quickly the better option than doing it by hand.

For most tasks you do in research when dealing with data, you will end up doing them many times (including doing the same task in future projects). So R offers the trade-off of spending time upfront by learning the code with the benefit of that code being able to do work at a large scale with little extra work from you. Please keep in mind this trade-off - you need to front-load the costs of learning R for the rewards of making your life easier when dealing with data - when feeling discouraged about the small returns you get early in learning R.

Reproducibility

The second major benefit of using R over something like Excel is that R is reproducible. Every action you take is written down. This is useful when collaborating with others (including your future self) as they can look at your code and follow along what you did without you having to show them every click you made as you frequently would on Excel. Your collaborator can look at your code to help you figure out a bug in the code or add their own code to yours.

In the research context specifically, you want to have code to give to people to ensure that your research was done correctly and there aren't bugs in the code. Additionally, if you build a tool to, for example, interpret raw crime data from an agency and turn it into a map, being able to share the code so others can modify it for their own city saves these people a lot of time and effort.

While not required (yet) in criminology, some academic journals (such as in economics) even require that you submit your data and code if your paper is accepted. If criminology follows in this trend, or if you submit to journals that require code submissions, you'll need to be able to write code and not rely on software that doesn't track your steps (such as Excel and SPSS).

What you will learn

For many of the lessons we will be working through real research questions and working from start to finish as you would on your own project. This involves thinking about what you want to accomplish from the data you have and what steps you need to take to reach that goal. This involves more than just knowing what code to write - it includes figuring out what your data has, whether it can answer the question you're asking, and planning out (without writing any code yet) what you need to do when you start coding. For most lessons we'll be using actual crime data that is commonly used in research so you'll become acquainted to a number of important data sets.

Skills

There is a large range of skills in criminology research - far too large to cover in a single book. Here we will attempt to teach fundamental skills to build a solid foundation for future work. We'll be focusing on the following skills and trying to reinforce our skills with each lesson.

- Subsetting - Taking only certain rows or columns from a data set
- Graphing
- Regular expressions - Essentially R's "Find and Replace" function for text
- Getting data from websites (webscraping)
- Getting data from PDFs (PDF scraping)
- Mapping
- Writing documents through R

What you won't learn

This book is not a statistics book so we will not be covering any statistical techniques. Though some data sets we handle are fairly large, this book does not discuss how to deal with Big Data. While the lessons you learn in this book can apply to larger data sets, Big Data (which I tend to define loosely as data that are too large for my computer to handle) requires special skills that are outside the realm of this book. If you do intend to deal with huge data sets I recommend you look at the R package data.table[4] which is an excellent resource for it. While we briefly cover mapping, this book will not cover working with geographic data in detail. For a comprehensive look at geographic data please see this book[5]. This book also will not cover any qualitative data or analysis. While qualitative research is an important part of criminology, this book only focuses on working with quantitative data. Some parts of this book may apply to dealing with qualitative data, such as PDF scraping and regular expressions, but the examples I use in those chapters still deal with quantitative data.

Simple vs Easy

In the course of this book we will cover things that are very simple. For example, we'll take a data set (think of it like an Excel file) with crime for nearly every police agency in the United States and keep only data from Colorado for a small number of years. We'll then find out how many murders happened in Colorado each year. This is a fairly simple task - it can be expressed in two sentences. You'll find that most of what you do is simple like this - it is quick to talk about what you are doing and the concepts are not complicated. What it isn't is easy. To actually write the R code to do this takes knowing a number of interrelated concepts in R and several lines of code to implement each step.

While this distinction may seem minor, I think it is important for newer programmers to understand that what they are doing may be simple to talk about but hard to implement. It is easy to feel like a bad programmer because something that can be articulated in 10 seconds may take hours to do. So during times when you are working with R try to keep in mind that even though a project may be simple to articulate, it may be hard to code and that there is often very little correlation between the two.

[4]https://github.com/Rdatatable/data.table/wiki
[5]https://geocompr.robinlovelace.net/

How to read this book

This book is written so a person who has no programming experience can start with this chapter and by the end of the book be able to do a data project from start to finish. Each chapter introduces a new skill and builds on the skills introduced in previous chapters. So if you skip ahead you may miss important skills taught in the chapters you didn't read. For someone who has no - or minimal - programming experience, I recommend reading each chapter in order. If you have more programming experience and just want to learn how to do a specific thing, feel free to skip directly to that chapter.

Citing this book

If this book was useful in your research, please cite it. To cite this book, please use the below citation:

Kaplan J (2021). *Crime by the Numbers: A Criminologist's Guide to R.* `https://crimebythenumbers.com/`.

BibTeX format:

```
@Manual{crimebythenumbers,
  title = {Crime by the Numbers: A Criminologist's Guide to R},
  author = {Jacob Kaplan},
  year = {2021},
  url = {https://crimebythenumbers.com/},
}
```

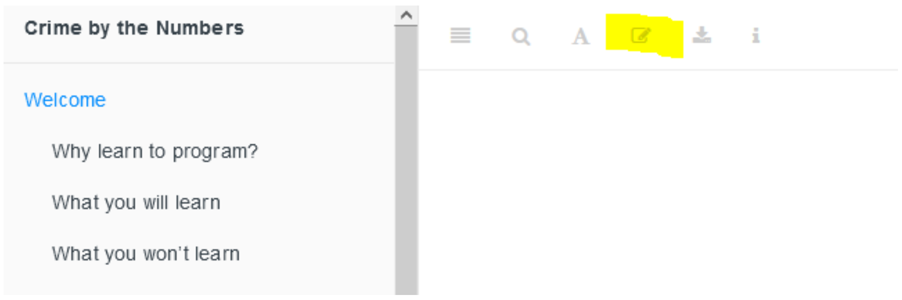
How to contribute to this book

If you have any questions, suggestions (such as a topic to cover), or find any issues, please make a post on the Issues page[6] for this book on GitHub. On this page you can create a new issue (which is basically just a post on this

[6]`https://github.com/jacobkap/crimebythenumbers/issues`

forum) with a title and a longer description of your issue. You'll need a GitHub account to make a post. Posting here lets me track issues and respond to your message or alert you when the issue is closed (i.e. I've finished or denied the request). Issues are also public so you can see if someone has already posted something similar.

For more minor issues like typos or grammar mistakes, you can edit the book directly through its GitHub page. That'll make an update for me to accept, which will change the book to include your edit. To do that, click the edit button at the top of the site - the button is highlighted in the below figure. You will need to make a GitHub account to make edits. When you click on that button you'll be taken to a page that looks like a Word Doc where you can make edits. Make any edits you want and then scroll to the bottom of the page. There you can write a short (please, no more than a sentence or two) description of what you've done and then submit the changes for me to review.



Please only use the above two methods to contribute or make suggestions about the book. Don't email me. While it's a bit more work for you to do it this way, since you'll need to make a GitHub account if you don't already have one, it helps me. I wrote this book, in part, to help my career so having evidence that people read it and are contributing to it is important to me. It's a way to publicly measure the book's impact.

Where to find data included in this book

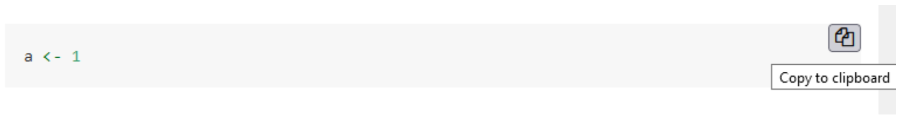
To download the data used in this book please see here[7]. Each of the files that are used in this book are available to download at that link. At the top of

[7] https://github.com/jacobkap/r4crimz/tree/master/data

every chapter that uses one of these files I'll say exactly which file(s) you need to download. The best way to use this book is to follow along by downloading the data and running the code that I include in each chapter.

Where to find code included in this book

If you're reading this book through its website[8] you can easily copy the code by clicking on the "Copy to clipboard" option on the top right of every chunk of code. This button, shown in the image below, will copy all of the code in the chunk and you can then paste (through Control/Command+V) into R.



```r
a <- 1
```

I've also made each chapter available to download as an R file that has every line of code used in each chapter available to you to run. To download the files, please go to the book's GitHub page here[9]. I've saved each chapter twice - once where it only includes the code used (in the "just_code" folder) and once where it includes the code and all of the text in the chapter (in the "code_and_text" folder). So download whichever one you want to use. The code is identical in each.

[8] https://crimebythenumbers.com
[9] https://github.com/jacobkap/crimebythenumbers/tree/master/code_repository

0 Interactive maps

For this chapter you'll need the following files, which are available for download here[10]: san_francisco_marijuana_geocoded.csv and sf_neighborhoods_suicide.rda.

While maps of data are useful, their ability to show incident-level information is quite limited. They tend to show broad trends - where crime happened in a city - rather than provide information about specific crime incidents. While broad trends are important, there are significant drawbacks about being unable to get important information about an incident without having to check the data. An interactive map bridges this gap by showing trends while allowing you to zoom into individual incidents and see information about each incident.

For this lesson we will be using data on every marijuana dispensary in San Francisco that has an active dispensary license as of late September 2019. The file is called "san_francisco_marijuana_geocoded.csv".

When downloaded from California's Bureau of Cannabis Control (here[11] if you're interested) the data contains the address of each dispensary but does not have coordinates. Without coordinates we are unable to map points, meaning we need to geocode them. Geocoding is the process of taking an address and getting the longitude and latitude of that address for mapping. For this lesson I've already geocoded the data and we'll learn how to do so in Chapter @ref(geocoding).

```r
library(readr)
marijuana <- read_csv("data/san_francisco_marijuana_geocoded.csv")
marijuana <- as.data.frame(marijuana)
```

[10] https://github.com/jacobkap/r4crimz/tree/master/data
[11] https://aca5.accela.com/bcc/customization/bcc/cap/licenseSearch.aspx

0.1 Why do interactive graphs matter?

0.1.1 Understanding your data

The most important thing to learn from this book is that understanding your data is crucial to good research. Making interactive maps is a very useful way to better understand your data as you can immediately see geographic patterns and quickly look at characteristics of those incidents to understand them.

In this lesson we will make a map of each marijuana dispensary in San Francisco that lets you click on the dispensary and see some information about it. If we see a cluster of dispensaries, we can click on each one to see if they are similar - for example if owned by the same person. Though it is possible to find these patterns just looking at the data, it is easier to be able to see a geographic pattern and immediately look at information about each incident.

0.1.2 Police departments use them

Interactive maps are popular in large police departments such as Philadelphia and New York City. They allow easy understanding of geographic patterns in the data and, importantly, allow such access to people who do not have the technical skills necessary to interact with the data itself. If nothing else, learning interactive maps may help you with a future job.

0.2 Making the interactive map

As usual, let's take a look at the top 6 rows of the data.

```
head(marijuana)
#>    License_Number                License_Type
#> 1 C10-0000614-LIC Cannabis - Retailer License
#> 2 C10-0000586-LIC Cannabis - Retailer License
#> 3 C10-0000587-LIC Cannabis - Retailer License
#> 4 C10-0000539-LIC Cannabis - Retailer License
#> 5 C10-0000522-LIC Cannabis - Retailer License
#> 6 C10-0000523-LIC Cannabis - Retailer License
#>      Business_Owner        Business_Structure
#> 1      Terry Muller Limited Liability Company
#> 2     Jeremy Goodin               Corporation
#> 3      Justin Jarin               Corporation
#> 4 Ondyn Herschelle               Corporation
#> 5       Ryan Hudson Limited Liability Company
#> 6       Ryan Hudson Limited Liability Company
#>                           Premise_Address Status Issue_Date
#> 1  2165 IRVING ST san francisco, CA 94122 Active  9/13/2019
#> 2 122 10TH ST SAN FRANCISCO, CA 941032605 Active  8/26/2019
#> 3   843 Howard ST SAN FRANCISCO, CA 94103 Active  8/26/2019
#> 4    70 SECOND ST SAN FRANCISCO, CA 94105 Active   8/5/2019
#> 5   527 Howard ST San Francisco, CA 94105 Active  7/29/2019
#> 6 2414 Lombard ST San Francisco, CA 94123 Active  7/29/2019
#>   Expiration_Date                Activities
#> 1       9/12/2020 N/A for this license type
#> 2       8/25/2020 N/A for this license type
#> 3       8/25/2020 N/A for this license type
#> 4        8/4/2020 N/A for this license type
#> 5       7/28/2020 N/A for this license type
#> 6       7/28/2020 N/A for this license type
#>   Adult-Use/Medicinal      lat      long
#> 1                BOTH 37.76318 -122.4811
#> 2                BOTH 37.77480 -122.4157
#> 3                BOTH 37.78228 -122.4035
#> 4                BOTH 37.78823 -122.4004
#> 5                BOTH 37.78783 -122.3965
#> 6                BOTH 37.79944 -122.4414
```

This data has information about the type of license, who the owner is, and where the dispensary is (as an address and as coordinates). We'll be making a map showing every dispensary in the city and make it so when you click a dot it'll make a popup showing information about that dispensary.

We will use the package `leaflet` for our interactive map. `leaflet` produces maps similar to Google Maps with circles (or any icon we choose) for each value we add to the map. It allows you to zoom in, scroll around, and provides context to each incident that isn't available on a static map.

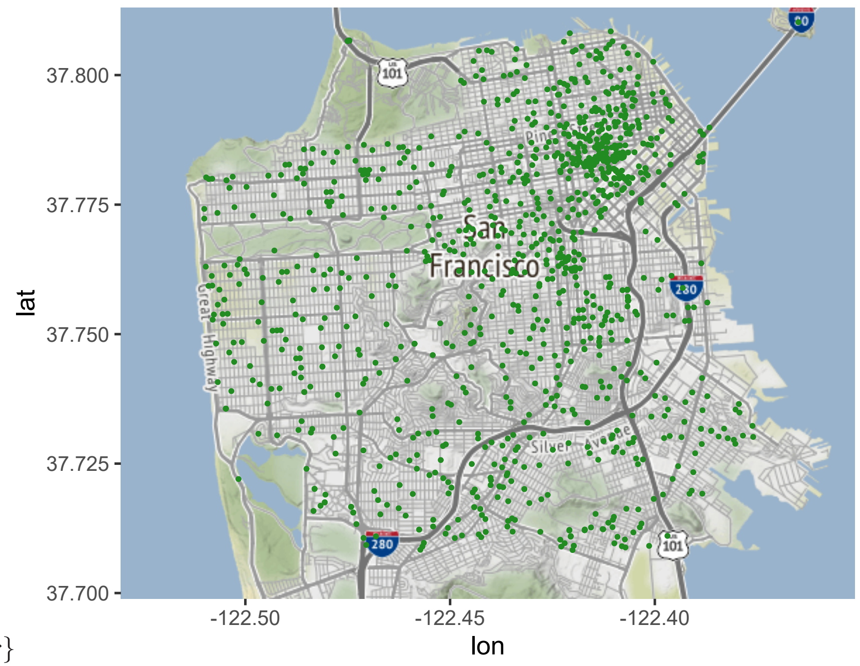
```
install.packages("leaflet")
```

```
library(leaflet)
```

To make a `leaflet` map we need to run the function `leaflet()` and add a tile to the map. A tile is simply the background of the map. This website[12] provides a large number of potential tiles to use, though many are not relevant to our purposes of crime mapping.

We will use a standard tile from Open Street Maps. This tile gives street names and highlights important features such as parks and large stores which provides useful contexts for looking at the data. The `attribution` parameter isn't strictly necessary but it is good form to say where your tile is from.

```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                  OpenStreetMap</a> contributors'
  )
```



\begin{center}

When you run the above code it shows a world map (copied several times).

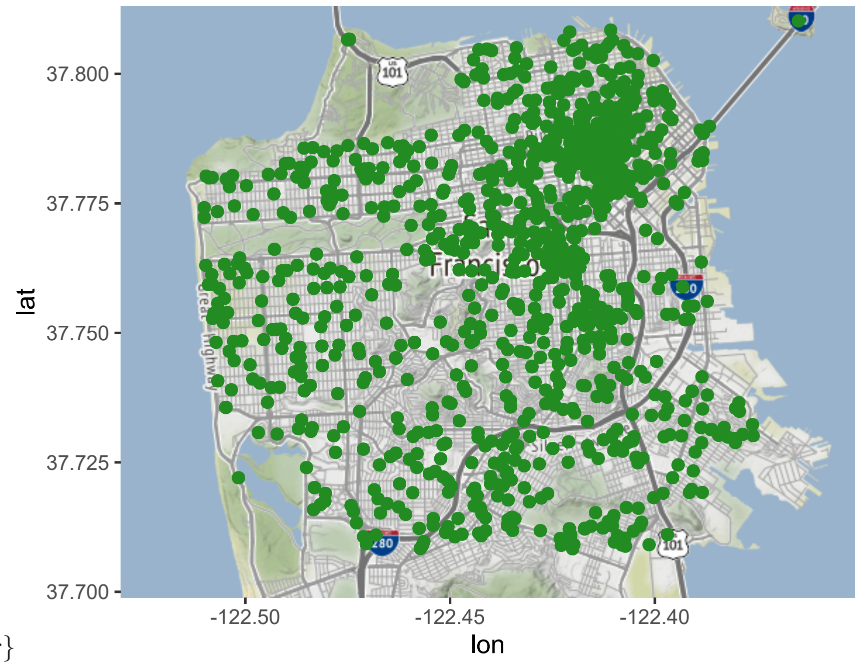
[12] https://leaflet-extras.github.io/leaflet-providers/preview/

Zoom into it and it'll start showing relevant features of wherever you're looking.

Note the `%>%` between the `leaflet()` function and the `addTiles()` function. `leaflet` is one of the packages in R where we can use pipes.

To add the points to the graph we use the function `addMarkers()` which has two parameters, `lng` and `lat`. For both parameters we put the column in which the longitude and latitude are, respectively.

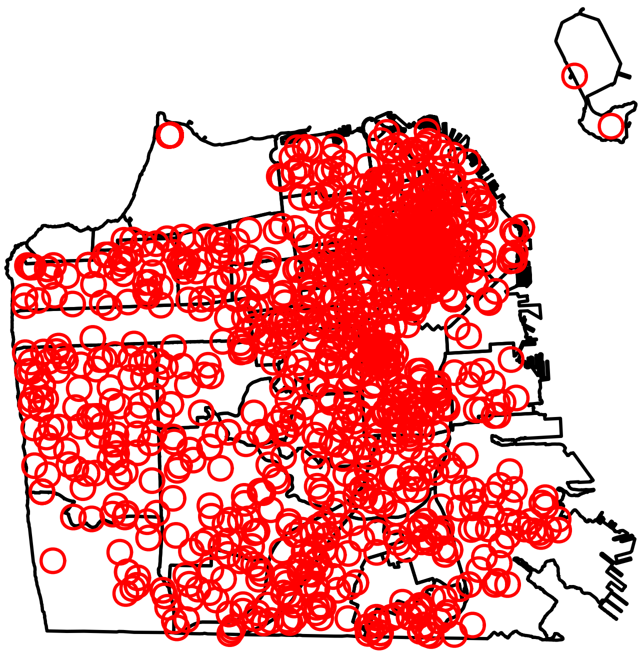
```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                  OpenStreetMap</a> contributors'
  ) %>%
  addMarkers(
    lng = marijuana$long,
    lat = marijuana$lat
  )
```



\begin{center}

It now adds an icon indicating where every dispensary in our data is. You can zoom in and scroll around to see more about where the dispensaries are. There are only a few dozen locations in the data so the popups overlapping a bit doesn't affect our map too much. If we had more - such as crime data with millions of offenses - it would make it very hard to read. To change the icons to circles we can change the function `addMarkers()` to `addCircleMarkers()`, keeping the rest of the code the same.

```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addCircleMarkers(
    lng = marijuana$long,
    lat = marijuana$lat
  )
```

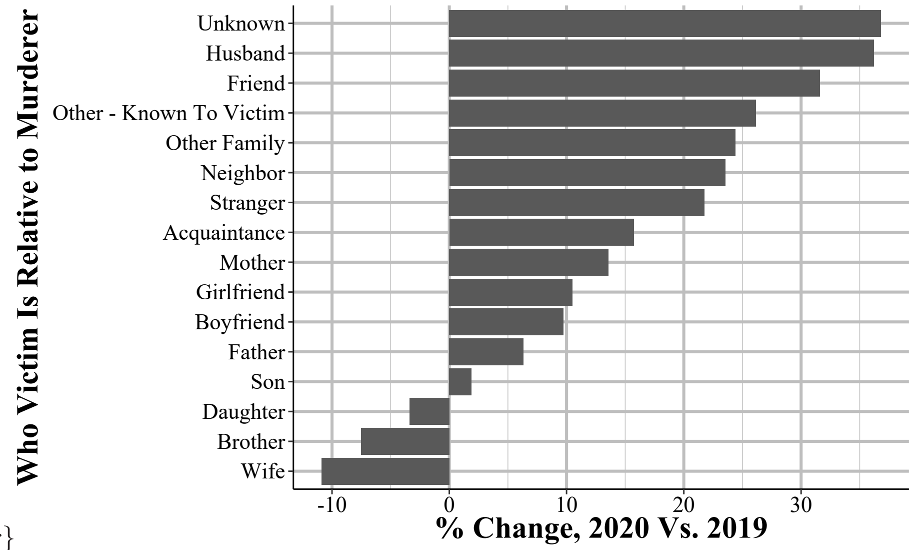


\begin{center}

This makes the icon into circles which take up less space than icons. To adjust the size of our icons we use the `radius` parameter in `addMarkers()` or `addCircleMarkers()`. The larger the radius, the larger the icons.

```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
```

```
                OpenStreetMap</a> contributors'
  ) %>%
  addCircleMarkers(
    lng = marijuana$long,
    lat = marijuana$lat,
    radius = 5
  )
```



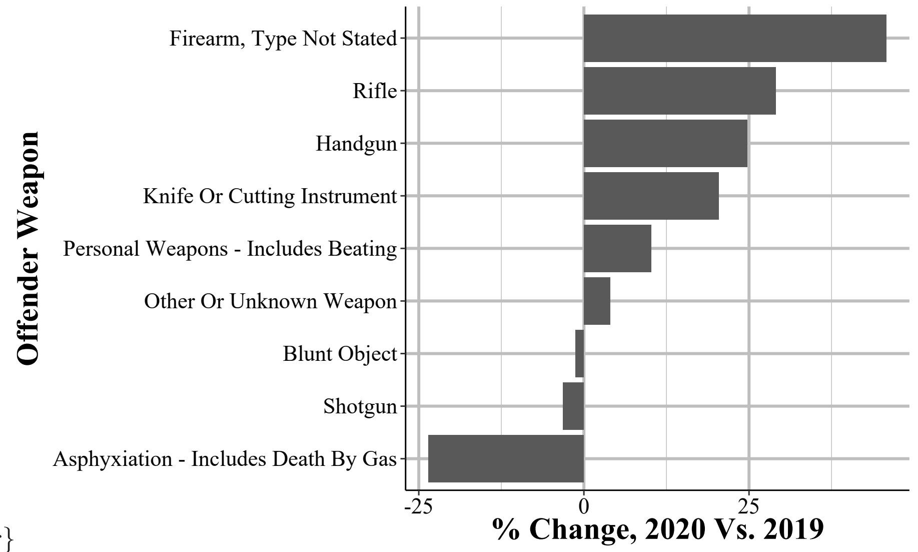
\begin{center}

Setting the `radius` option to 5 shrinks the size of the icon a lot. In your own maps you'll have to fiddle with this option to get it to look the way you want. Let's move on to adding information about each icon when clicked upon.

0.3 Adding popup information

The parameter `popup` in the `addMarkers()` or `addCircleMarkers()` functions lets you input a character value (if not already a character value it will convert it to one) and that will be shown as a popup when you click on the icon. Let's start simple here by inputting the business owner column in our data and then build it up to a more complicated popup.

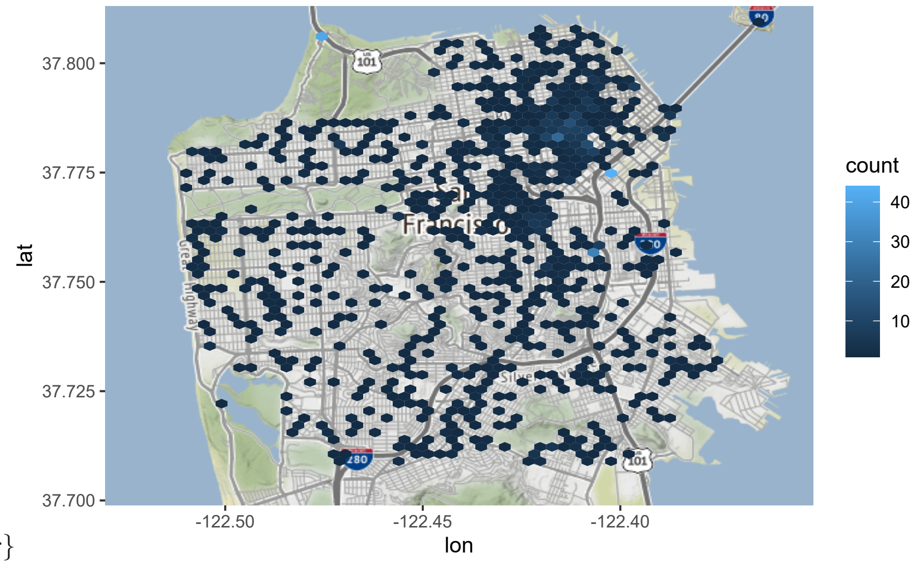
```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addCircleMarkers(
    lng = marijuana$long,
    lat = marijuana$lat,
    radius = 5,
    popup = marijuana$Business_Owner
  )
```



\begin{center}

Try clicking around and you'll see that the owner of the dispensary you clicked on appears over the dot. We usually want to have a title indicating what the value in the popup means. We can do this by using the `paste()` function to combine text explaining the value with the value itself. Let's add the words "Business Owner:" before the business owner column.

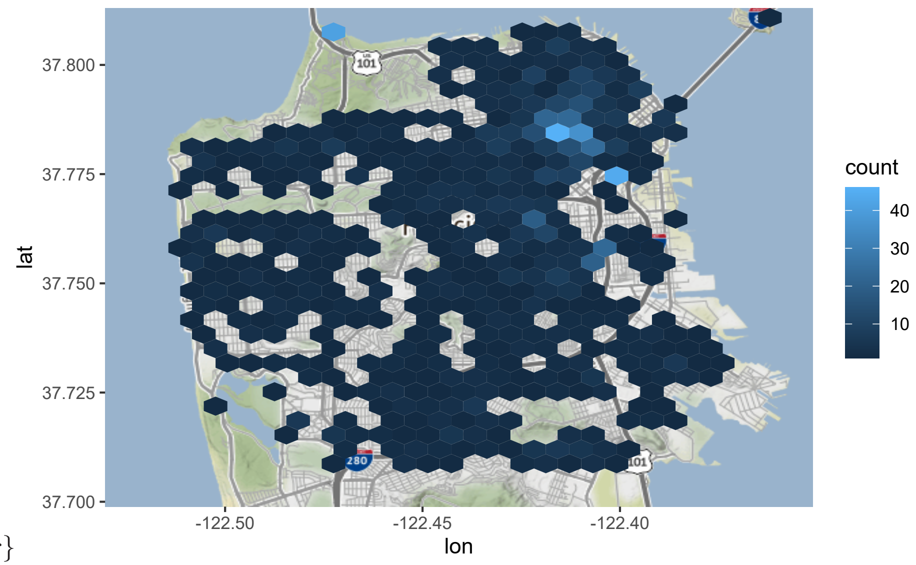
```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addCircleMarkers(
    lng = marijuana$long,
    lat = marijuana$lat,
    radius = 5,
    popup = paste(
      "Business Owner:",
      marijuana$Business_Owner
    )
  )
```



\begin{center}

We don't have too much information in the data but we let's add the address and license number to the popup by adding them to the `paste()` function we're using.

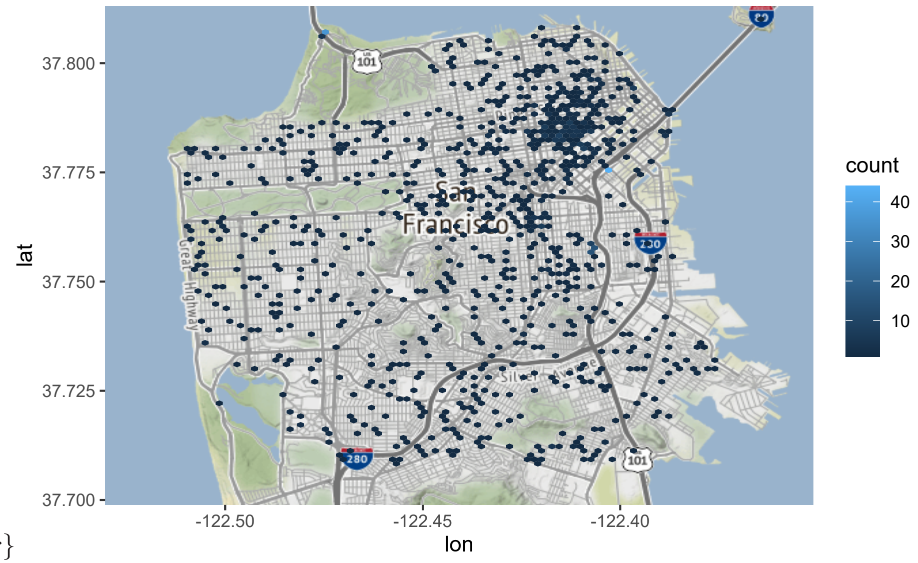
```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addCircleMarkers(
    lng = marijuana$long,
    lat = marijuana$lat,
    radius = 5,
    popup = paste(
      "Business Owner:",
      marijuana$Business_Owner,
      "Address:",
      marijuana$Premise_Address,
      "License:",
      marijuana$License_Number
    )
  )
```



\begin{center}

Just adding the location text makes it try to print out everything on one line which is hard to read. If we add the text `<br>` where we want a line break it will make one. `<br>` is the HTML tag for line-break which is why it works making a new line in this case.

```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                  OpenStreetMap</a> contributors'
  ) %>%
  addCircleMarkers(
    lng = marijuana$long,
    lat = marijuana$lat,
    radius = 5,
    popup = paste(
      "Business Owner:",
      marijuana$Business_Owner,
      "<br>",
      "Address:",
      marijuana$Premise_Address,
      "<br>",
      "License:",
      marijuana$License_Number
    )
  )
```



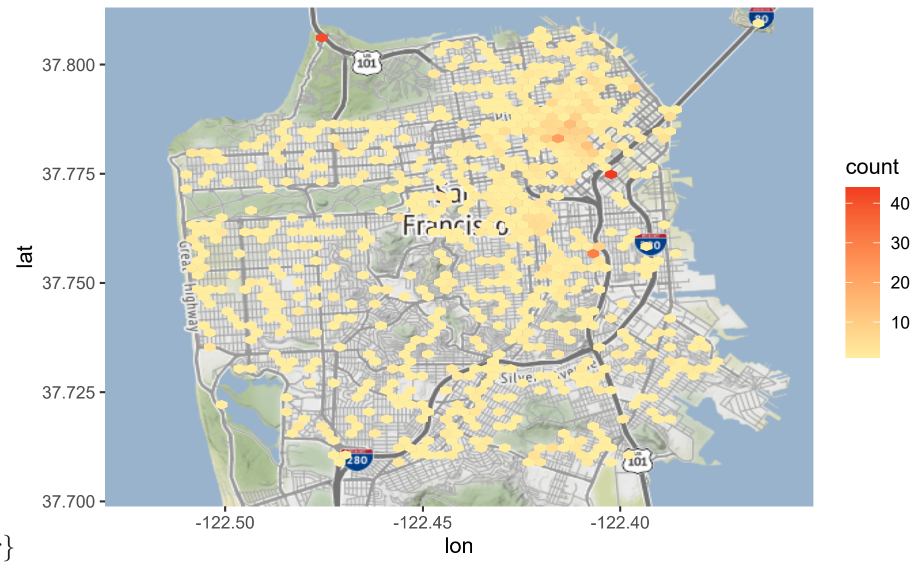
\begin{center}

0.4 Dealing with too many markers

In our case with only 33 rows of data, turning the markers to circles solves our visibility issue. In cases with many more rows of data, this doesn't always work. A solution for this is to cluster the data into groups where the dots only show if you zoom in.

If we add the code `clusterOptions = markerClusterOptions()` to our `addCircleMarkers()` it will cluster for us.

```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addCircleMarkers(
    lng = marijuana$long,
    lat = marijuana$lat,
    radius = 5,
    popup = paste(
      "Business Owner:",
      marijuana$Business_Owner,
      "<br>",
      "Address:",
      marijuana$Premise_Address,
      "<br>",
      "License:",
      marijuana$License_Number
    ),
    clusterOptions = markerClusterOptions()
  )
```



\begin{center}

Locations close to each other are grouped together in fairly arbitrary groupings and we can see how large each grouping is by moving our cursor over the circle. Click on a circle or zoom in and it will show smaller groupings at lower levels of aggregation. Keep clicking or zooming in and it will eventually show each location as its own circle.

This method is very useful for dealing with huge amounts of data as it avoids overflowing the map with too many icons at one time. A downside, however, is that the clusters are created arbitrarily meaning that important context, such as neighborhood, can be lost.

0.5 Interactive choropleth maps

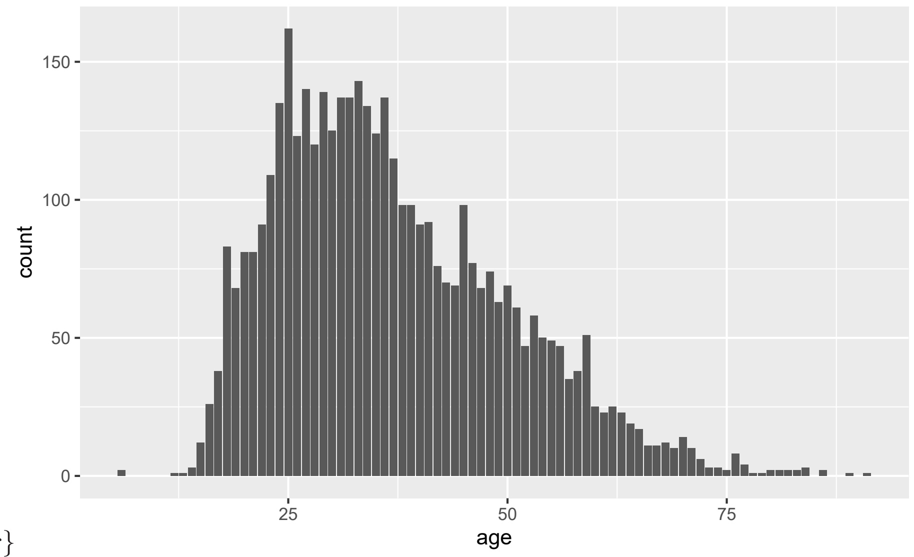
In Chapter @ref(choropleth-maps) we worked on choropleth maps which are maps with shaded regions, such as states colored by which political party won them in an election. Here we will make interactive choropleth maps where you can click on a shaded region and see information about that region. We'll make the same map as before - neighborhoods shaded by the number of suicides.

Let's load the San Francisco suicides-by-neighborhood data that we made earlier. We'll also want to project it to the standard longitude and latitude projection.

```r
library(sf)
load("data/sf_neighborhoods_suicide.rda")
sf_neighborhoods_suicide <- st_transform(
  sf_neighborhoods_suicide,
  "+proj=longlat +datum=WGS84"
)
```

We'll begin the `leaflet` map similar to before but use the function `addPolygons()` and our input here is the geometry column of *sf_neighborhoods_suicide*.

```r
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addPolygons(data = sf_neighborhoods_suicide$geometry)
```



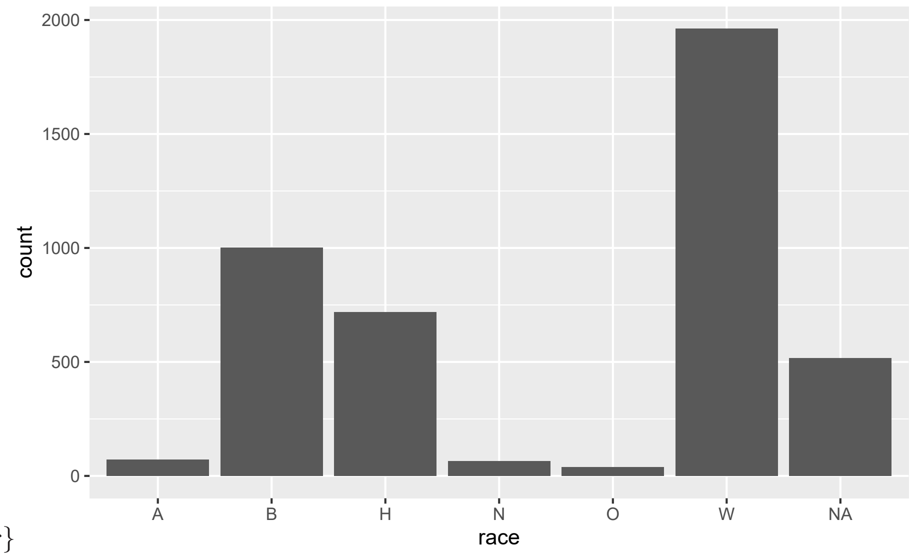
\begin{center}

It gives us a blank map because our polygons are projected to San Francisco's projection while the `leaflet` map expects the standard CRS, WGS84 which uses longitude and latitude. So we need to change our projection to that using the `st_transform()` function from the `sf` package.

```
library(sf)
sf_neighborhoods_suicide <- st_transform(sf_neighborhoods_suicide,
  crs = "+proj=longlat +datum=WGS84"
)
```

Now let's try again.

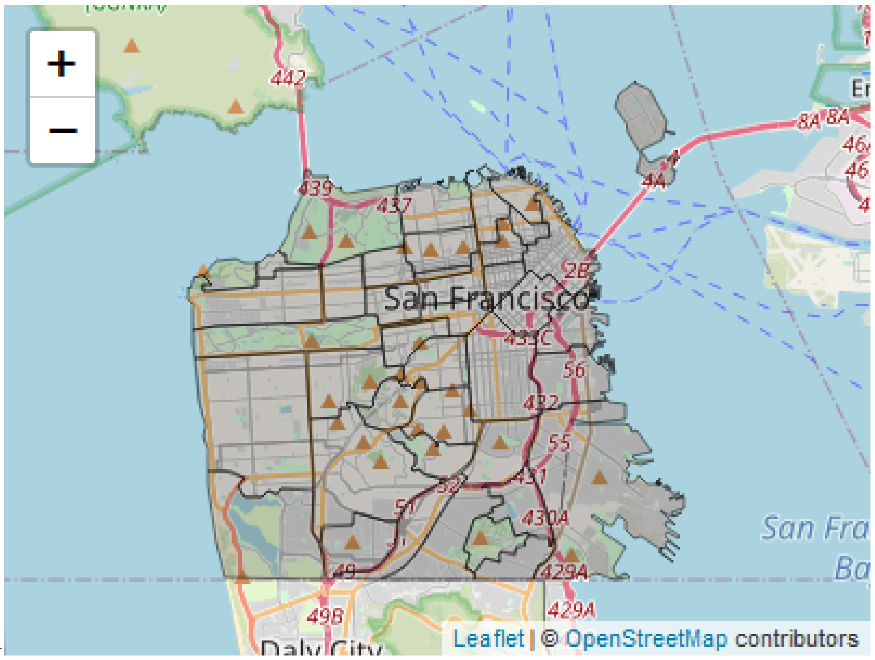
```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addPolygons(data = sf_neighborhoods_suicide$geometry)
```



\begin{center}

It made a map with thick blue lines indicating each neighborhood. Let's change the appearance of the graph a bit before making a popup or shading the neighborhoods The parameter `color` in `addPolygons()` changes the color of the lines - let's change it to black. The lines are also very thick, blurring into each other and making the neighborhoods hard to see. We can change the `weight` parameter to alter the size of these lines - smaller values are thinner lines. Let's try setting this to 1.

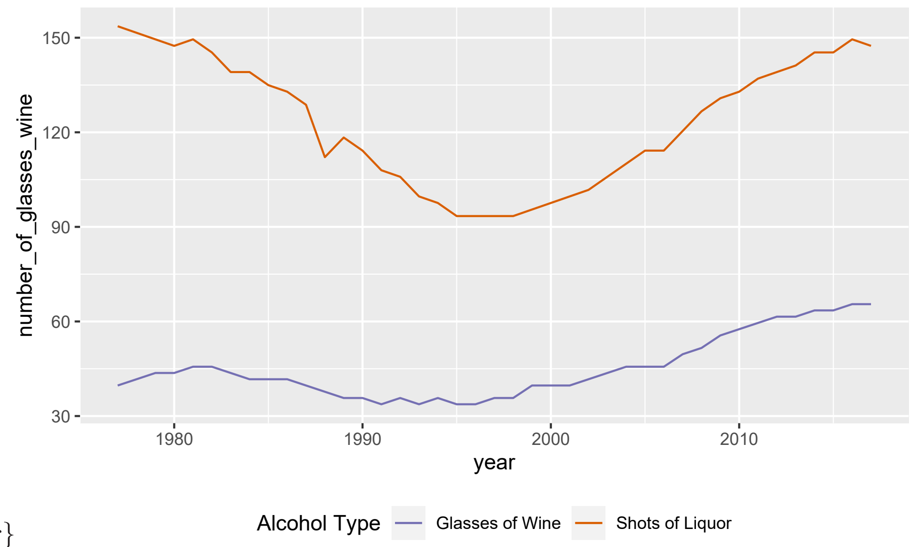
```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                  OpenStreetMap</a> contributors'
  ) %>%
  addPolygons(
    data = sf_neighborhoods_suicide$geometry,
    color = "black",
    weight = 1
  )
```

\begin{center}

That looks better and we can clearly distinguish each neighborhood now.

As we did earlier, we can add the popup text directly to the function which makes the geographic shapes, in this case `addPolygons()`. Let's add the *nhood* column value - the name of that neighborhood - and the number of suicides that occurred in that neighborhood. As before, when we click on a neighborhood a popup appears with the output we specified.

```
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                  OpenStreetMap</a> contributors'
  ) %>%
  addPolygons(
    data = sf_neighborhoods_suicide$geometry,
    col = "black",
    weight = 1,
    popup = paste0(
      "Neighborhood: ",
      sf_neighborhoods_suicide$nhood,
      "<br>",
      "Number of Suicides: ",
      sf_neighborhoods_suicide$number_suicides
  )
)
```



\begin{center}

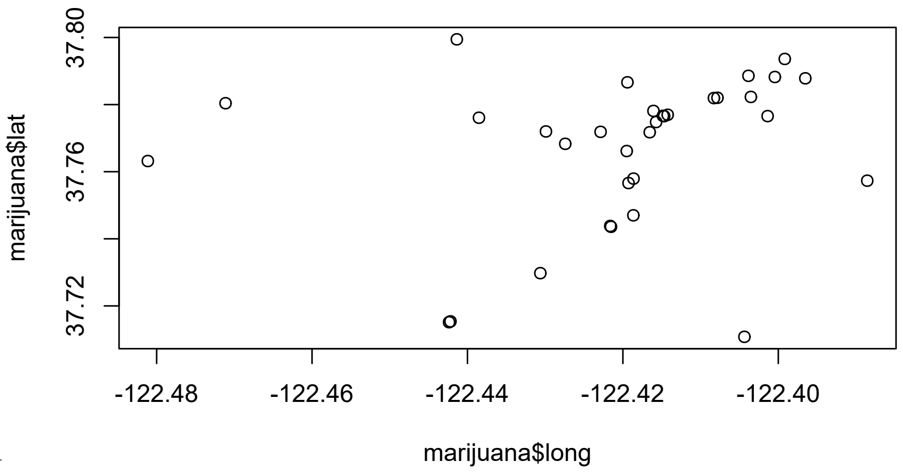
For these types of maps we generally want to shade each polygon to indicate how frequently the event occurred in the polygon. We'll use the function `colorNumeric()` which takes a lot of the work out of the process of coloring in the map. This function takes two inputs, first a color palette which we can get from the site Color Brewer[13]. Let's use the fourth bar in the Sequential page, which is light orange to red. If you look in the section with each HEX value it says that the palette is "3-class OrRd". The "3-class" just means we selected 3 colors, the "OrRd" is the part we want. That will tell `colorNumeric()` to make the palette using these colors. The second parameter is the column for our numeric variable, *number_suicides*.

We will save the output of `colorNumeric("OrRd", sf_neighborhoods_suicide$number_suicides)` as a new object which we'll call *pal* for convenience since it is a palette of colors. Then inside of `addPolygons()` we'll set the parameter `fillColor` to `pal(sf_neighborhoods_suicide$number_suicides)`, running this function on the column. What this really does is determine which color every neighborhood should be based on the value in the *number_suicides* column.

```
pal <- colorNumeric("OrRd", sf_neighborhoods_suicide$number_suicides)
leaflet() %>%
```

[13] http://colorbrewer2.org/#type=sequential&scheme=OrRd&n=3

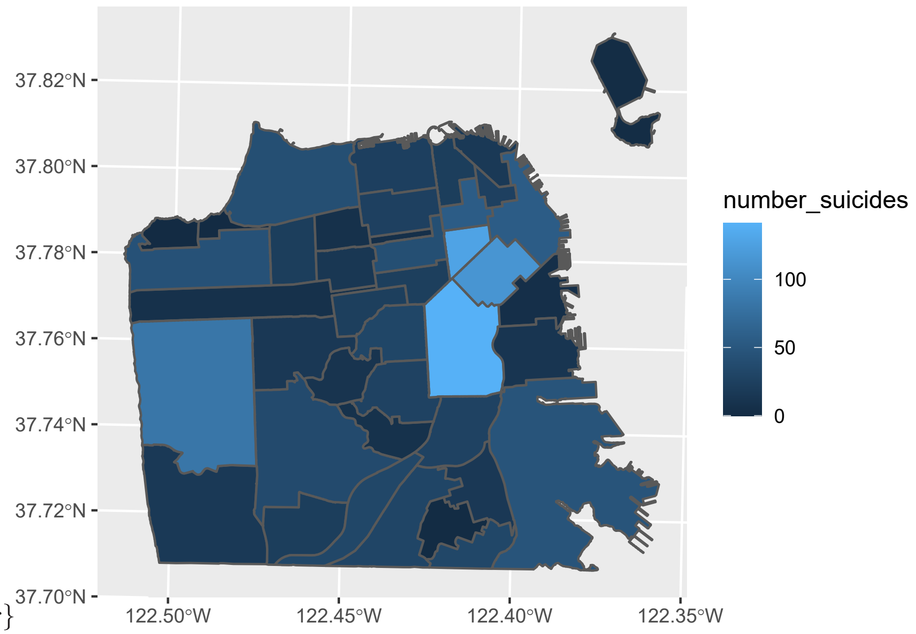
```
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addPolygons(
    data = sf_neighborhoods_suicide$geometry,
    col = "black",
    weight = 1,
    popup = paste0(
      "Neighborhood: ",
      sf_neighborhoods_suicide$nhood,
      "<br>",
      "Number of Suicides: ",
      sf_neighborhoods_suicide$number_suicides
    ),
    fillColor = pal(sf_neighborhoods_suicide$number_suicides)
  )
```



\begin{center}

Since the neighborhoods are transparent, it is hard to distinguish which color is shown. We can make each neighborhood a solid color by setting the parameter `fillOpacity` inside of `addPolygons()` to 1.

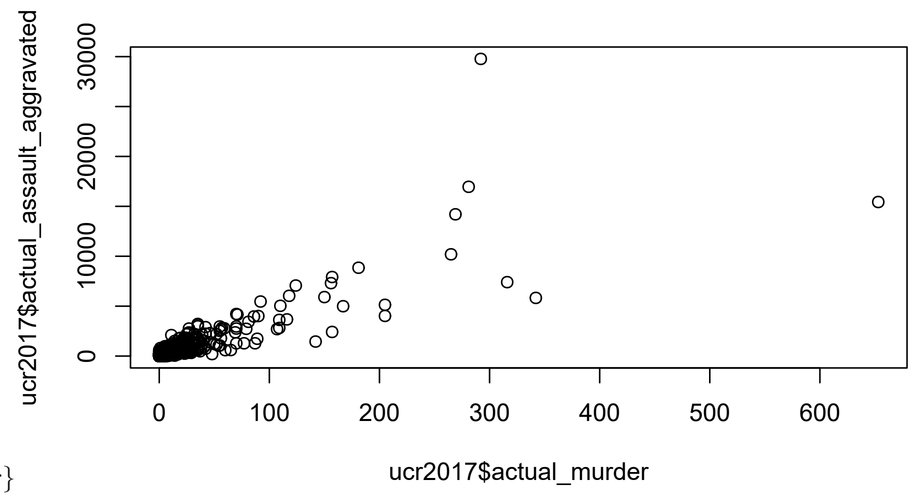
```
pal <- colorNumeric("OrRd", sf_neighborhoods_suicide$number_suicides)
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addPolygons(
    data = sf_neighborhoods_suicide$geometry,
    col = "black",
    weight = 1,
    popup = paste0(
      "Neighborhood: ",
      sf_neighborhoods_suicide$nhood,
      "<br>",
      "Number of Suicides: ",
      sf_neighborhoods_suicide$number_suicides
    ),
    fillColor = pal(sf_neighborhoods_suicide$number_suicides),
    fillOpacity = 1
  )
```



\begin{center}

To add a legend to this we use the function `addLegend()` which takes three parameters. `pal` asks which color palette we are using - we want it to be the exact same as we use to color the neighborhoods so we'll use the *pal* object we made. The `values` parameter is used for which column our numeric values are from, in our case the *number_suicides* column so we'll input that. Finally `opacity` determines how transparent the legend will be. As each neighborhood is set to not be transparent at all, we'll also set this to 1 to be consistent.

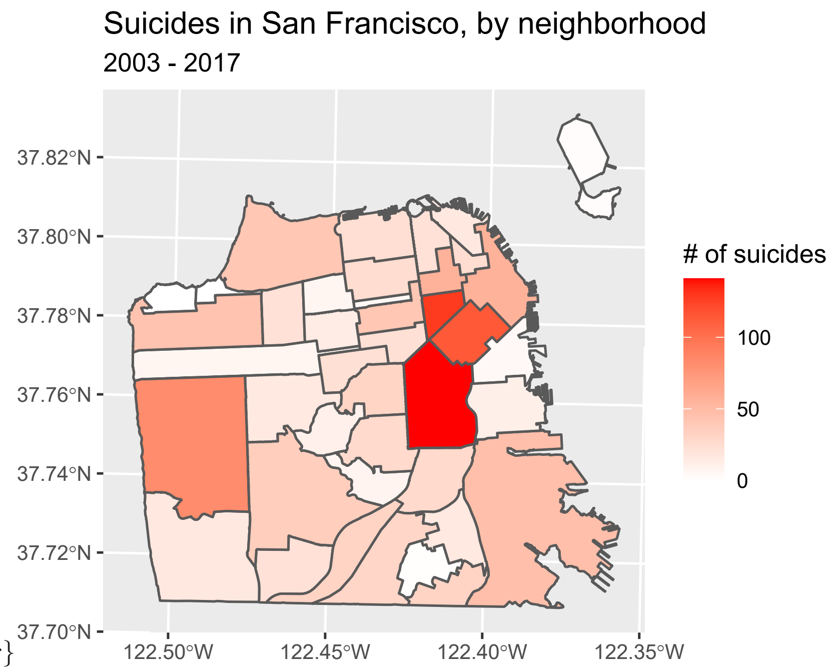
```
pal <- colorNumeric("OrRd", sf_neighborhoods_suicide$number_suicides)
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                  OpenStreetMap</a> contributors'
  ) %>%
  addPolygons(
    data = sf_neighborhoods_suicide$geometry,
    col = "black",
    weight = 1,
    popup = paste0(
      "Neighborhood: ",
      sf_neighborhoods_suicide$nhood,
      "<br>",
      "Number of Suicides: ",
      sf_neighborhoods_suicide$number_suicides
    ),
    fillColor = pal(sf_neighborhoods_suicide$number_suicides),
    fillOpacity = 1
  ) %>%
  addLegend(
    pal = pal,
    values = sf_neighborhoods_suicide$number_suicides,
    opacity = 1
  )
```



\begin{center}

Finally, we can add a title to the legend using the `title` parameter inside of `addLegend()`.

```r
pal <- colorNumeric("OrRd", sf_neighborhoods_suicide$number_suicides)
leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    attribution = '&copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors'
  ) %>%
  addPolygons(
    data = sf_neighborhoods_suicide$geometry,
    col = "black",
    weight = 1,
    popup = paste0(
      "Neighborhood: ",
      sf_neighborhoods_suicide$nhood,
      "<br>",
      "Number of Suicides: ",
      sf_neighborhoods_suicide$number_suicides
    ),
    fillColor = pal(sf_neighborhoods_suicide$number_suicides),
    fillOpacity = 1
  ) %>%
  addLegend(
    pal = pal,
    values = sf_neighborhoods_suicide$number_suicides,
    opacity = 1,
    title = "Suicides"
  )
```



\begin{center}

0

Scraping tables from PDFs

For this chapter you'll need the following file, which is available for download here[14]: usbp_stats_fy2017_sector_profile.pdf.

Government agencies in particular like to release their data in long PDFs which often have the data we want in a table on one of the pages. To use this data we need to scrape it from the PDF into R. In the majority of cases when you want data from a PDF it will be in a table. Essentially the data will be an Excel file inside of a PDF. This format is not altogether different than what we've done before.

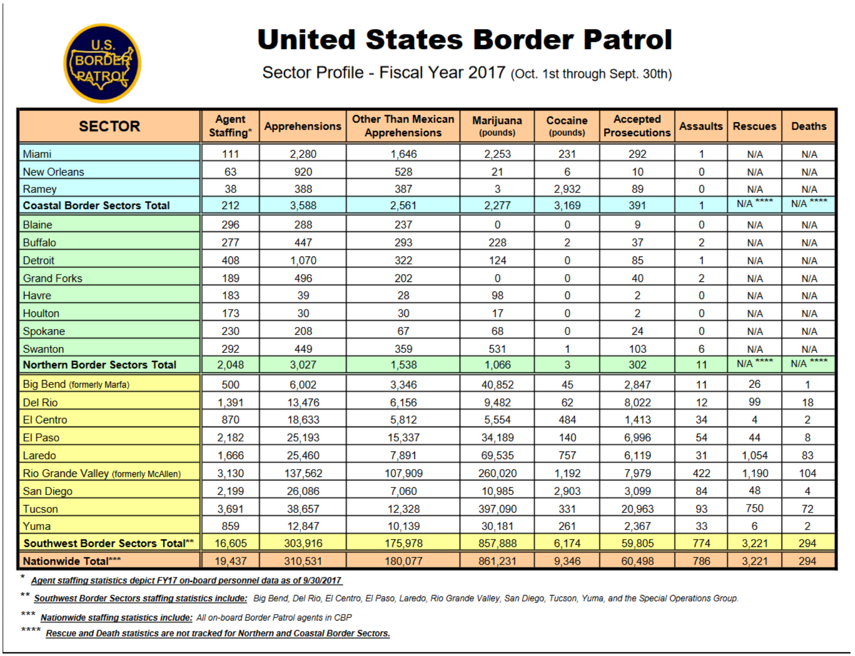
Let's first take a look at the data we will be scraping. The first step in any PDF scraping should be to look at the PDF and try to think about the best way to approach this particular problem. While all PDF scraping follows a general format, you cannot necessarily reuse your old code as each situation is likely slightly different. Our data is from the US Customs and Border Protection (CBP) and contains a wealth of information about apprehensions and contraband seizures in border sectors.

We will be using the Sector Profile 2017 PDF which has information in four tables, three of which we'll scrape and then combine together. The data was downloaded from the US Customs and Border Protection "Stats and Summaries" page here[15]. If you're interested in using more of their data, some of it has been cleaned and made available here[16].

The file we want to use is called "usbp_stats_fy2017_sector_profile.pdf" and has four tables in the PDF. Let's take a look at them one at a time, understanding what variables are available, and what units each row is in. Then we'll start scraping the tables.

The first table is "Sector Profile - Fiscal Year 2017 (Oct. 1st through Sept. 30th)". Before we even look down more at the table, the title is important. It is for fiscal year 2017, not calendar year 2017 which is more common in the data we usually use. This is important if we ever want to merge this data with other data sets. If possible, we would have to get data that is monthly so we can just use October 2016 through September 2017 to match up properly.

[14] https://github.com/jacobkap/r4crimz/tree/master/data
[15] https://www.cbp.gov/newsroom/media-resources/stats
[16] https://www.openicpsr.org/openicpsr/project/109522/version/V2/view



United States Border Patrol

Sector Profile - Fiscal Year 2017 (Oct. 1st through Sept. 30th)

SECTOR	Agent Staffing*	Apprehensions	Other Than Mexican Apprehensions	Marijuana (pounds)	Cocaine (pounds)	Accepted Prosecutions	Assaults	Rescues	Deaths
Miami	111	2,280	1,646	2,253	231	292	1	N/A	N/A
New Orleans	63	920	528	21	6	10	0	N/A	N/A
Ramey	38	388	387	3	2,932	89	0	N/A	N/A
Coastal Border Sectors Total	212	3,588	2,561	2,277	3,169	391	1	N/A ****	N/A ****
Blaine	296	288	237	0	0	9	0	N/A	N/A
Buffalo	277	447	293	228	2	37	2	N/A	N/A
Detroit	408	1,070	322	124	0	85	1	N/A	N/A
Grand Forks	189	496	202	0	0	40	2	N/A	N/A
Havre	183	39	28	98	0	2	0	N/A	N/A
Houlton	173	30	30	17	0	2	0	N/A	N/A
Spokane	230	208	67	68	0	24	0	N/A	N/A
Swanton	292	449	359	531	1	103	6	N/A	N/A
Northern Border Sectors Total	2,048	3,027	1,538	1,066	3	302	11	N/A ****	N/A ****
Big Bend (formerly Marfa)	500	6,002	3,346	40,852	45	2,847	11	26	1
Del Rio	1,391	13,476	6,156	9,482	62	8,022	12	99	18
El Centro	870	18,633	5,812	5,554	484	1,413	34	4	2
El Paso	2,182	25,193	15,337	34,189	140	6,996	54	44	8
Laredo	1,666	25,460	7,891	69,535	757	6,119	31	1,054	83
Rio Grande Valley (formerly McAllen)	3,130	137,562	107,909	260,020	1,192	7,979	422	1,190	104
San Diego	2,199	26,086	7,060	10,985	2,903	3,099	84	48	4
Tucson	3,691	38,657	12,328	397,090	331	20,963	93	750	72
Yuma	859	12,847	10,139	30,181	261	2,367	33	6	2
Southwest Border Sectors Total**	16,605	303,916	175,978	857,888	6,174	59,805	774	3,221	294
Nationwide Total***	19,437	310,531	180,077	861,231	9,346	60,498	786	3,221	294

* *Agent staffing statistics depict FY17 on-board personnel data as of 9/30/2017*

** *Southwest Border Sectors staffing statistics include:* *Big Bend, Del Rio, El Centro, El Paso, Laredo, Rio Grande Valley, San Diego, Tucson, Yuma, and the Special Operations Group.*

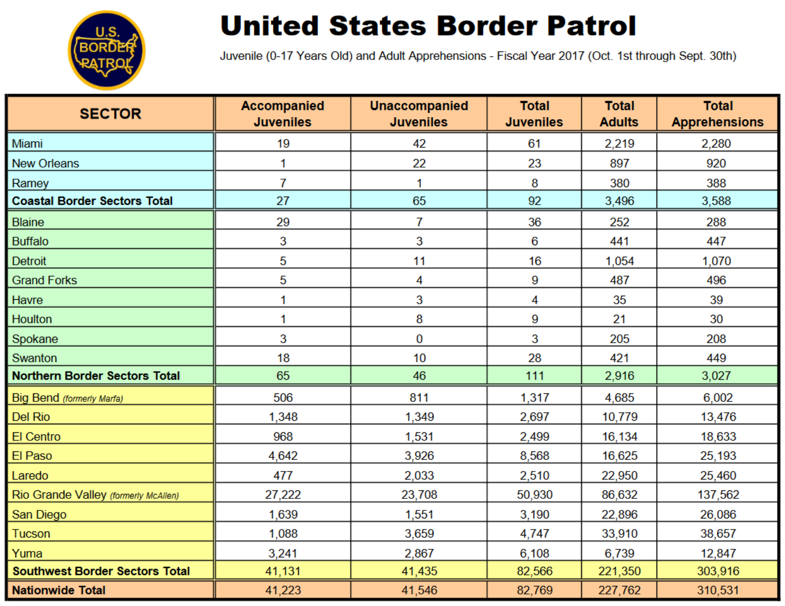
*** *Nationwide staffing statistics include:* *All on-board Border Patrol agents in CBP*

**** *Rescue and Death statistics are not tracked for Northern and Coastal Border Sectors.*

Now if we look more at the table, we can see that each row is a section of the US border. There are three main sections - Coastal, Northern, and Southwest, with subsections of each also included. The bottom row is the sum of all these sections and gives us nationwide data. Many government data sets will be like this form with sections and subsections in the same table. Watch out when doing mathematical operations! Just summing any of these columns will give you triple the true value due to the presence of nationwide, sectional, and subsectional data.

There are 9 columns in the data other than the border section identifier. We have total apprehensions, apprehensions for people who are not Mexican citizens, marijuana and cocaine seizures (in pounds), the number of accepted prosecutions (presumably of those apprehended), and the number of CBP agents assaulted. The last two columns have the number of people rescued by CBP and the number of people who died (it is unclear from this data alone if this is solely people in custody or deaths during crossing the border). These two columns are also special as they only have data for the Southwest border.

Table 2 has a similar format with each row being a section or subsection. The columns now have the number of juveniles apprehended, subdivided by if they were accompanied by an adult or not, and the number of adults apprehended. The last column is total apprehensions which is also in Table 1.

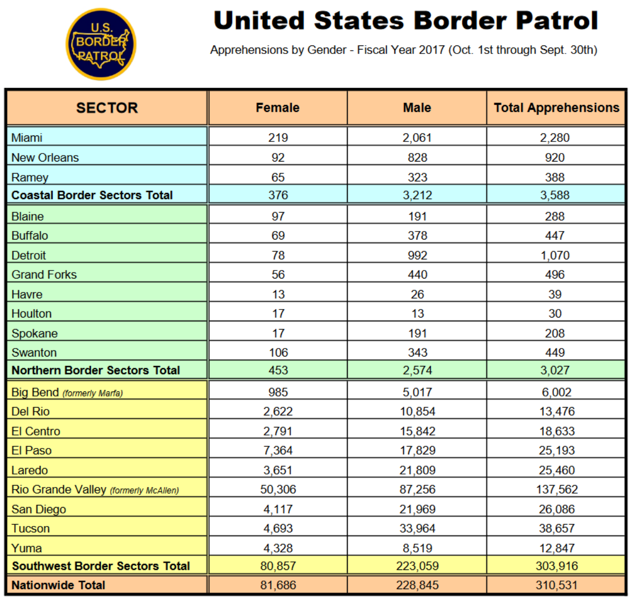


United States Border Patrol

Juvenile (0-17 Years Old) and Adult Apprehensions - Fiscal Year 2017 (Oct. 1st through Sept. 30th)

SECTOR	Accompanied Juveniles	Unaccompanied Juveniles	Total Juveniles	Total Adults	Total Apprehensions
Miami	19	42	61	2,219	2,280
New Orleans	1	22	23	897	920
Ramey	7	1	8	380	388
Coastal Border Sectors Total	27	65	92	3,496	3,588
Blaine	29	7	36	252	288
Buffalo	3	3	6	441	447
Detroit	5	11	16	1,054	1,070
Grand Forks	5	4	9	487	496
Havre	1	3	4	35	39
Houlton	1	8	9	21	30
Spokane	3	0	3	205	208
Swanton	18	10	28	421	449
Northern Border Sectors Total	65	46	111	2,916	3,027
Big Bend *(formerly Marfa)*	506	811	1,317	4,685	6,002
Del Rio	1,348	1,349	2,697	10,779	13,476
El Centro	968	1,531	2,499	16,134	18,633
El Paso	4,642	3,926	8,568	16,625	25,193
Laredo	477	2,033	2,510	22,950	25,460
Rio Grande Valley *(formerly McAllen)*	27,222	23,708	50,930	86,632	137,562
San Diego	1,639	1,551	3,190	22,896	26,086
Tucson	1,088	3,659	4,747	33,910	38,657
Yuma	3,241	2,867	6,108	6,739	12,847
Southwest Border Sectors Total	41,131	41,435	82,566	221,350	303,916
Nationwide Total	41,223	41,546	82,769	227,762	310,531

Table 3 follows the same format and the new columns are number of apprehensions by gender.

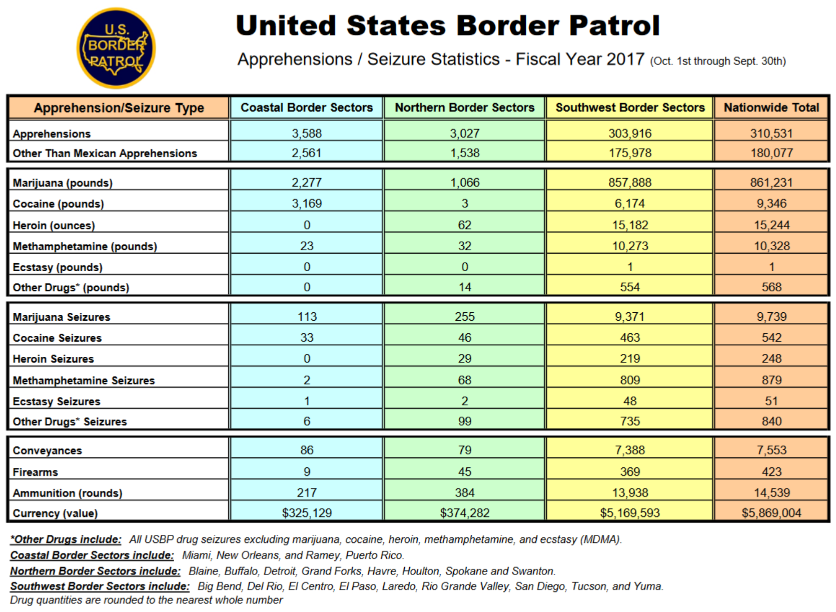


United States Border Patrol

Apprehensions by Gender - Fiscal Year 2017 (Oct. 1st through Sept. 30th)

SECTOR	Female	Male	Total Apprehensions
Miami	219	2,061	2,280
New Orleans	92	828	920
Ramey	65	323	388
Coastal Border Sectors Total	376	3,212	3,588
Blaine	97	191	288
Buffalo	69	378	447
Detroit	78	992	1,070
Grand Forks	56	440	496
Havre	13	26	39
Houlton	17	13	30
Spokane	17	191	208
Swanton	106	343	449
Northern Border Sectors Total	453	2,574	3,027
Big Bend *(formerly Marfa)*	985	5,017	6,002
Del Rio	2,622	10,854	13,476
El Centro	2,791	15,842	18,633
El Paso	7,364	17,829	25,193
Laredo	3,651	21,809	25,460
Rio Grande Valley *(formerly McAllen)*	50,306	87,256	137,562
San Diego	4,117	21,969	26,086
Tucson	4,693	33,964	38,657
Yuma	4,328	8,519	12,847
Southwest Border Sectors Total	80,857	223,059	303,916
Nationwide Total	81,686	228,845	310,531

Finally, Table 4 is a bit different in its format. The rows are now variables and the columns are the locations. In this table it doesn't include subsections, only border sections and the nationwide total. The data it has available are partially a repeat of Table 1 but with more drug types and the addition of the number of drug seizures and some firearm seizure information. As this table is formatted differently than the others, we won't scrape it in this lesson - but you can use the skills you'll learn to do so yourself.



United States Border Patrol

Apprehensions / Seizure Statistics - Fiscal Year 2017 (Oct. 1st through Sept. 30th)

Apprehension/Seizure Type	Coastal Border Sectors	Northern Border Sectors	Southwest Border Sectors	Nationwide Total
Apprehensions	3,588	3,027	303,916	310,531
Other Than Mexican Apprehensions	2,561	1,538	175,978	180,077
Marijuana (pounds)	2,277	1,066	857,888	861,231
Cocaine (pounds)	3,169	3	6,174	9,346
Heroin (ounces)	0	62	15,182	15,244
Methamphetamine (pounds)	23	32	10,273	10,328
Ecstasy (pounds)	0	0	1	1
Other Drugs* (pounds)	0	14	554	568
Marijuana Seizures	113	255	9,371	9,739
Cocaine Seizures	33	46	463	542
Heroin Seizures	0	29	219	248
Methamphetamine Seizures	2	68	809	879
Ecstasy Seizures	1	2	48	51
Other Drugs* Seizures	6	99	735	840
Conveyances	86	79	7,388	7,553
Firearms	9	45	369	423
Ammunition (rounds)	217	384	13,938	14,539
Currency (value)	$325,129	$374,282	$5,169,593	$5,869,004

Other Drugs include: *All USBP drug seizures excluding marijuana, cocaine, heroin, methamphetamine, and ecstasy (MDMA).*
Coastal Border Sectors include: *Miami, New Orleans, and Ramey, Puerto Rico.*
Northern Border Sectors include: *Blaine, Buffalo, Detroit, Grand Forks, Havre, Houlton, Spokane and Swanton.*
Southwest Border Sectors include: *Big Bend, Del Rio, El Centro, El Paso, Laredo, Rio Grande Valley, San Diego, Tucson, and Yuma.*
Drug quantities are rounded to the nearest whole number

0.6 Scraping the first table

We've now seen all three of the tables that we want to scrape so we can begin the process of actually scraping them. Note that each table is very similar meaning that we can reuse some code to scrape as well as to clean the data. That means that we will want to write some functions to make our work easier and avoid copy and pasting code.

We will start by using the `pdf_text()` function from the `pdftools` package to read the PDFs into R.