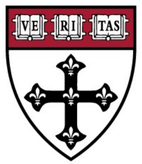
ggplot2

Summer Institute in Data Science

Rolando J. Acosta



HARVARD
SCHOOL OF PUBLIC HEALTH

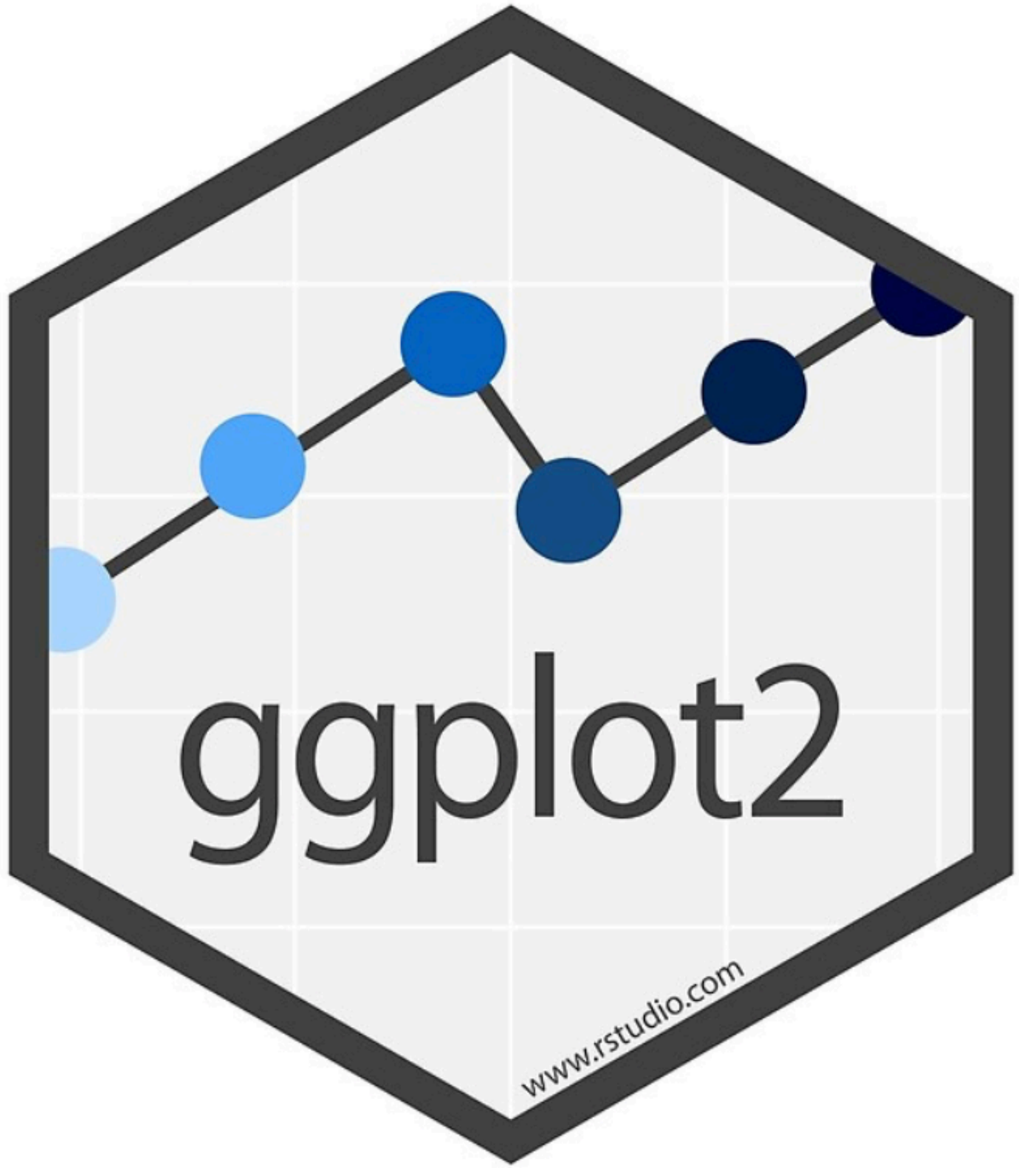
June 23, 2020

@RJANunez

What is ggplot2?

- "ggplot2 is an R package for producing statistical, or data, graphics, but it is unlike most other graphics packages because it has a deep underlying grammar. This grammar, based on the Grammar of Graphics (Wilkinson 2005), is made up of a set of independent components that can be composed in many different ways. This makes ggplot2 very powerful because you are not limited to a set of pre-specified graphics, but you can create new graphics that are precisely tailored for your problem."

https://ggplot2-book.org/index.html



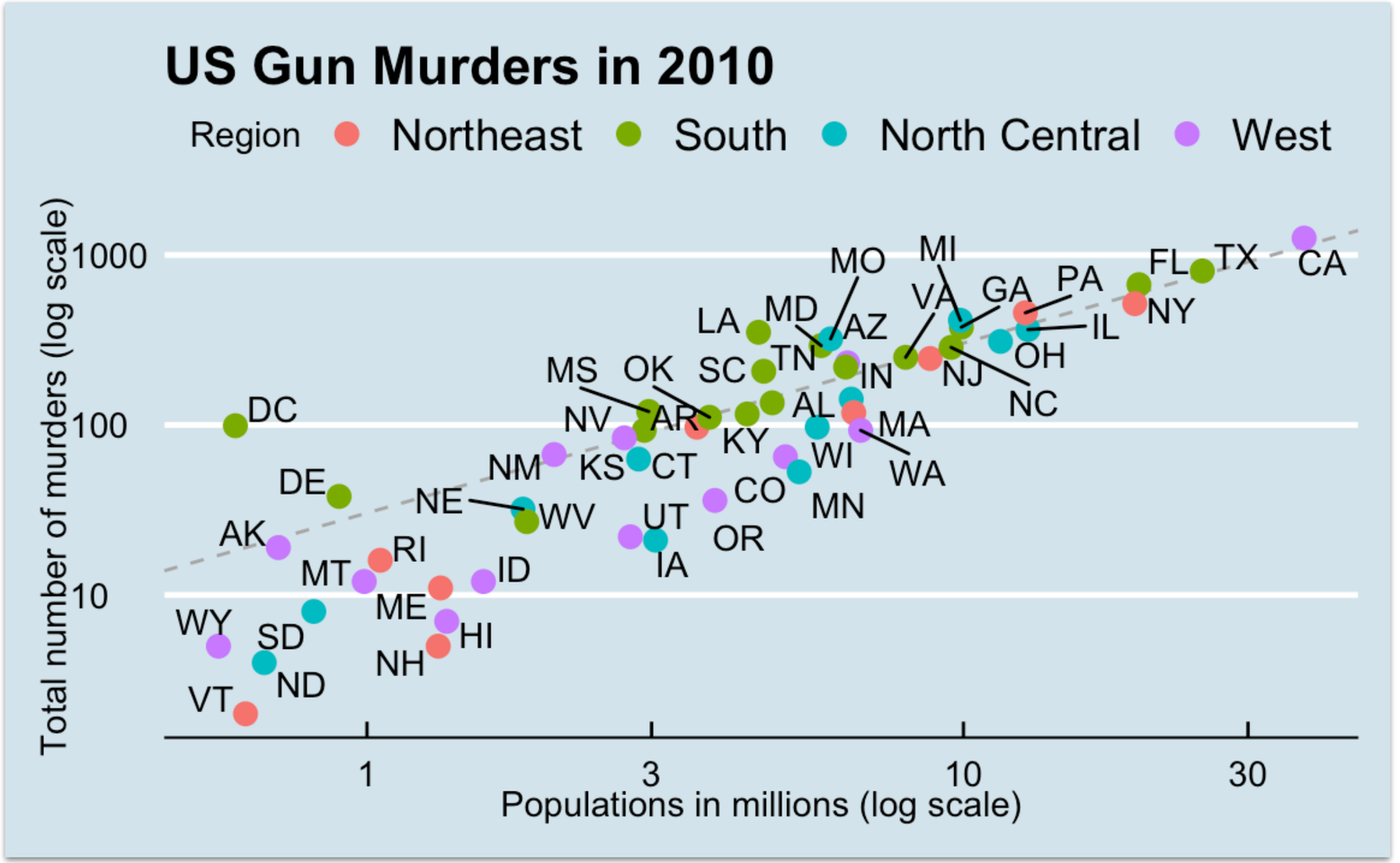
Why `ggplot2`?

- Many other approaches are available for creating plots in R:
  - `grid`
  - `lattice`
- The plotting options within base R are powerful yet many times unintuitive (in my opinion).
- Excel may be easier than R for some plots, but it is nowhere near as flexible. D3 may be more flexible and powerful than R, but it takes much longer to generate a plot.
- `ggplot2` breaks the components of a plot in a way that is intuitive and easier to understand for beginners
- We will learn how to construct plots the same way we learned how to construct sentences.

Limitations

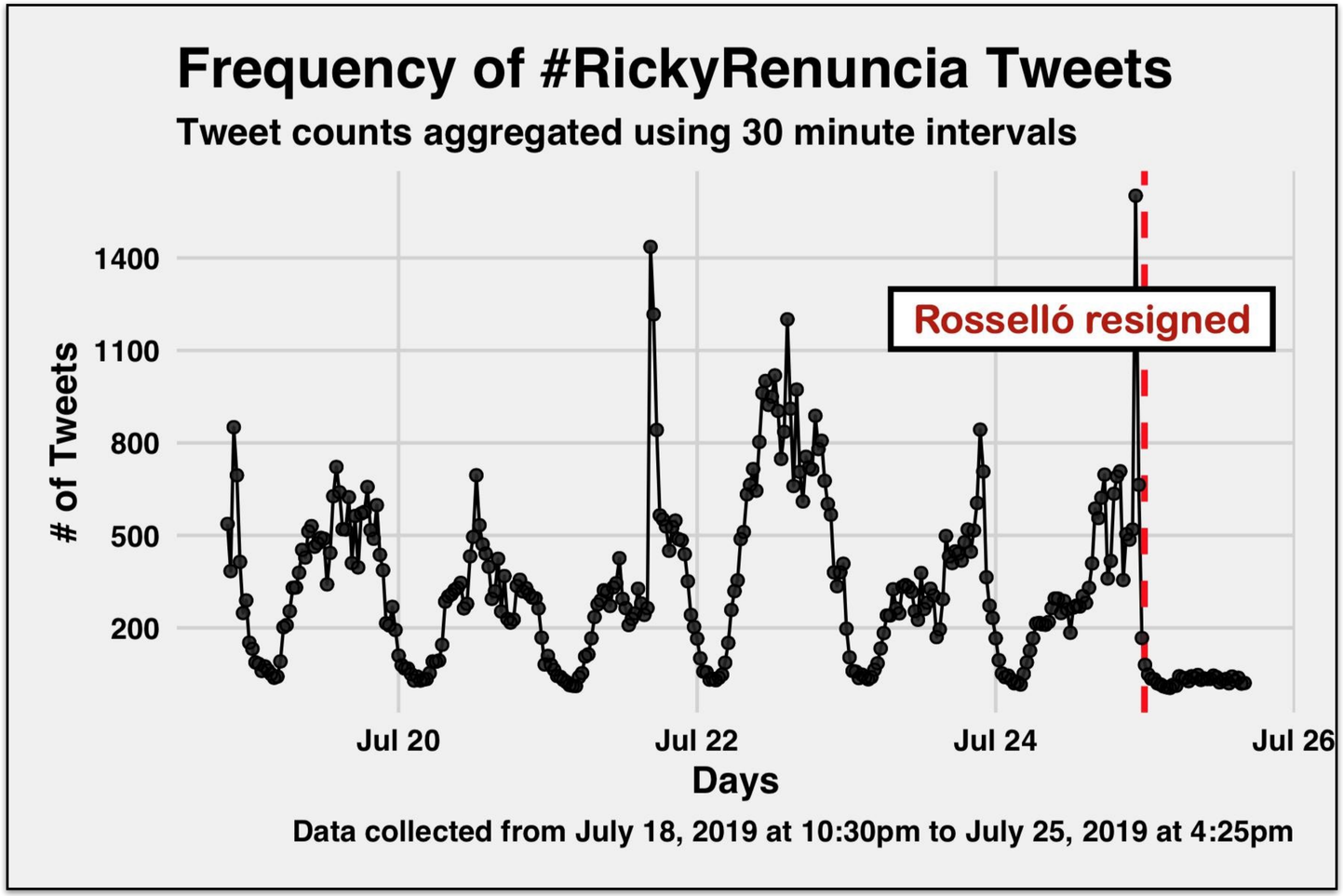
- ggplot2 is designed to work exclusively with data frames in tidy format
- There are a myriad of functions and arguments that are hard to memorize
- Luckily there is a cheat sheet! Find it here
- If you don't know how to do something, google it.
  - Google will be your best friend!

Motivation

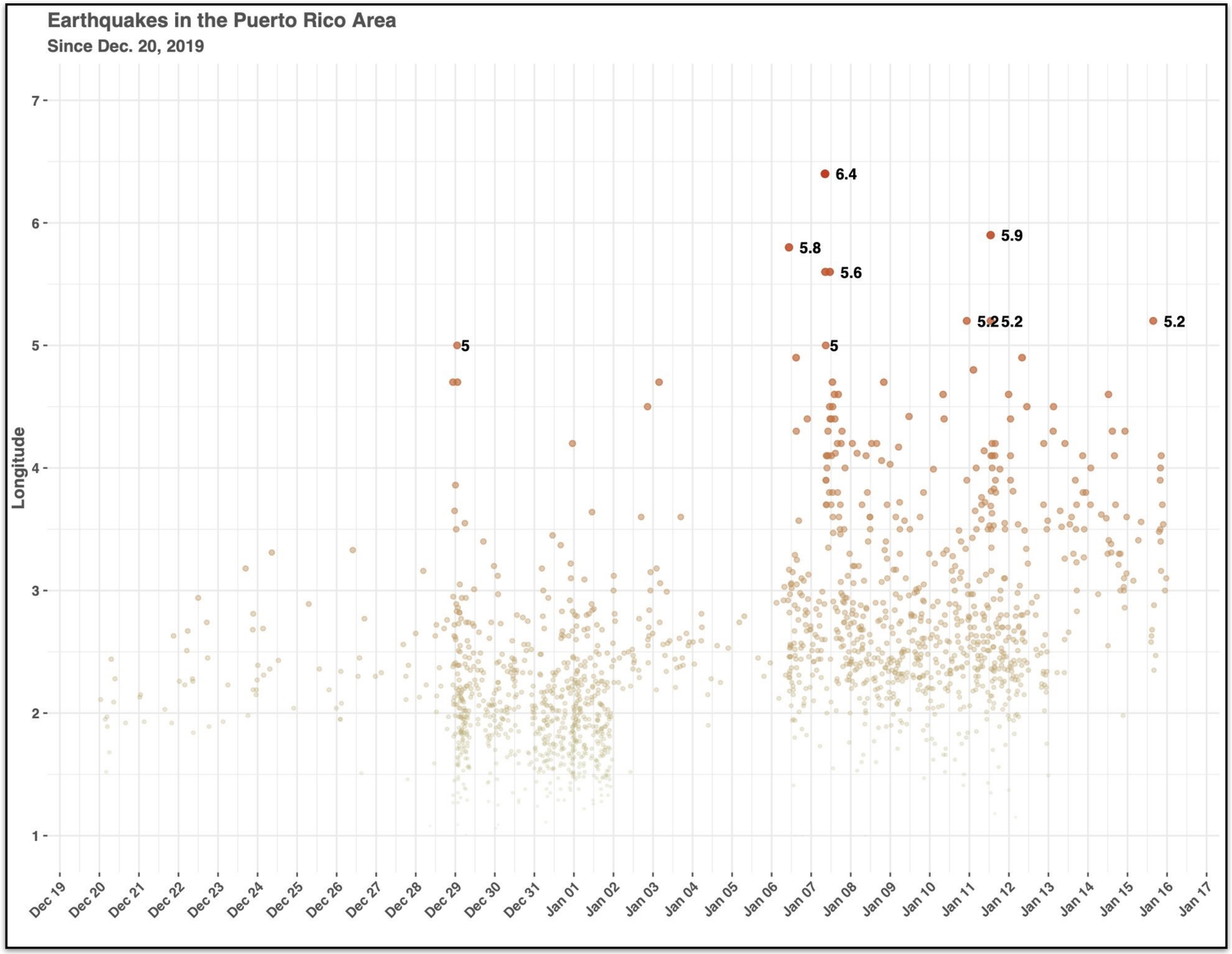


- We will create this plot from scratch

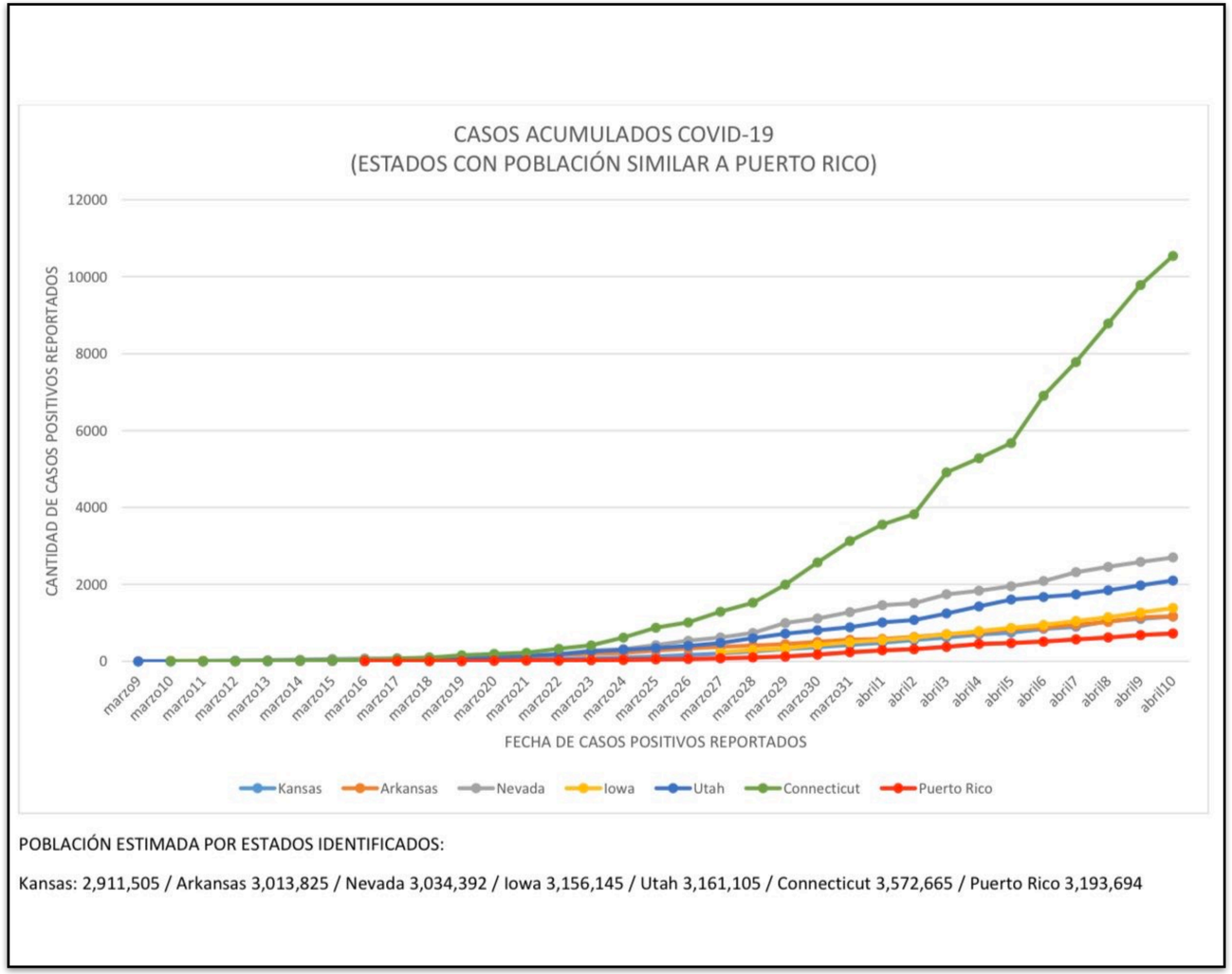
Motivation: Closer to home



Motivation: Closer to home



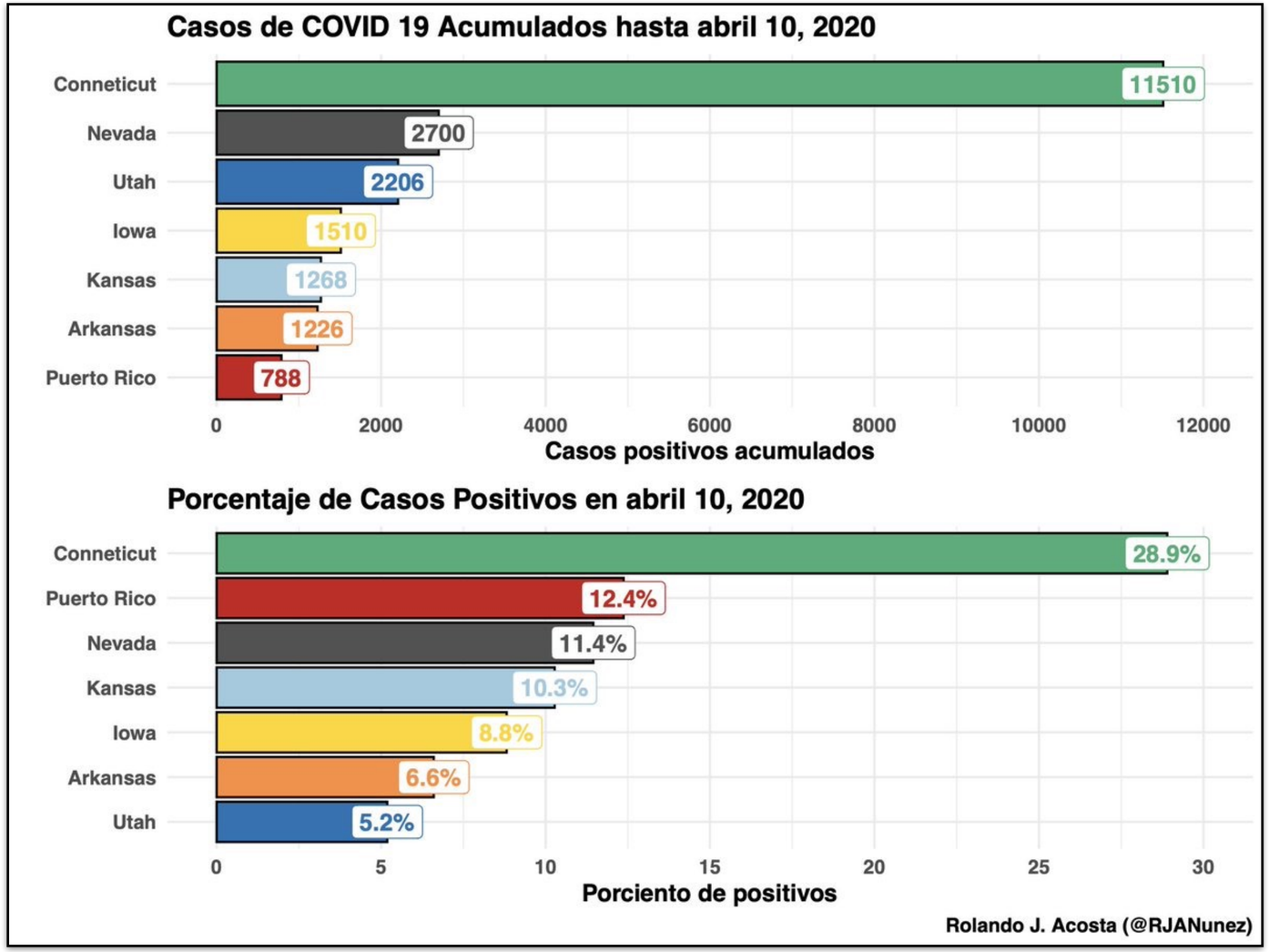
Motivation: Closer to home



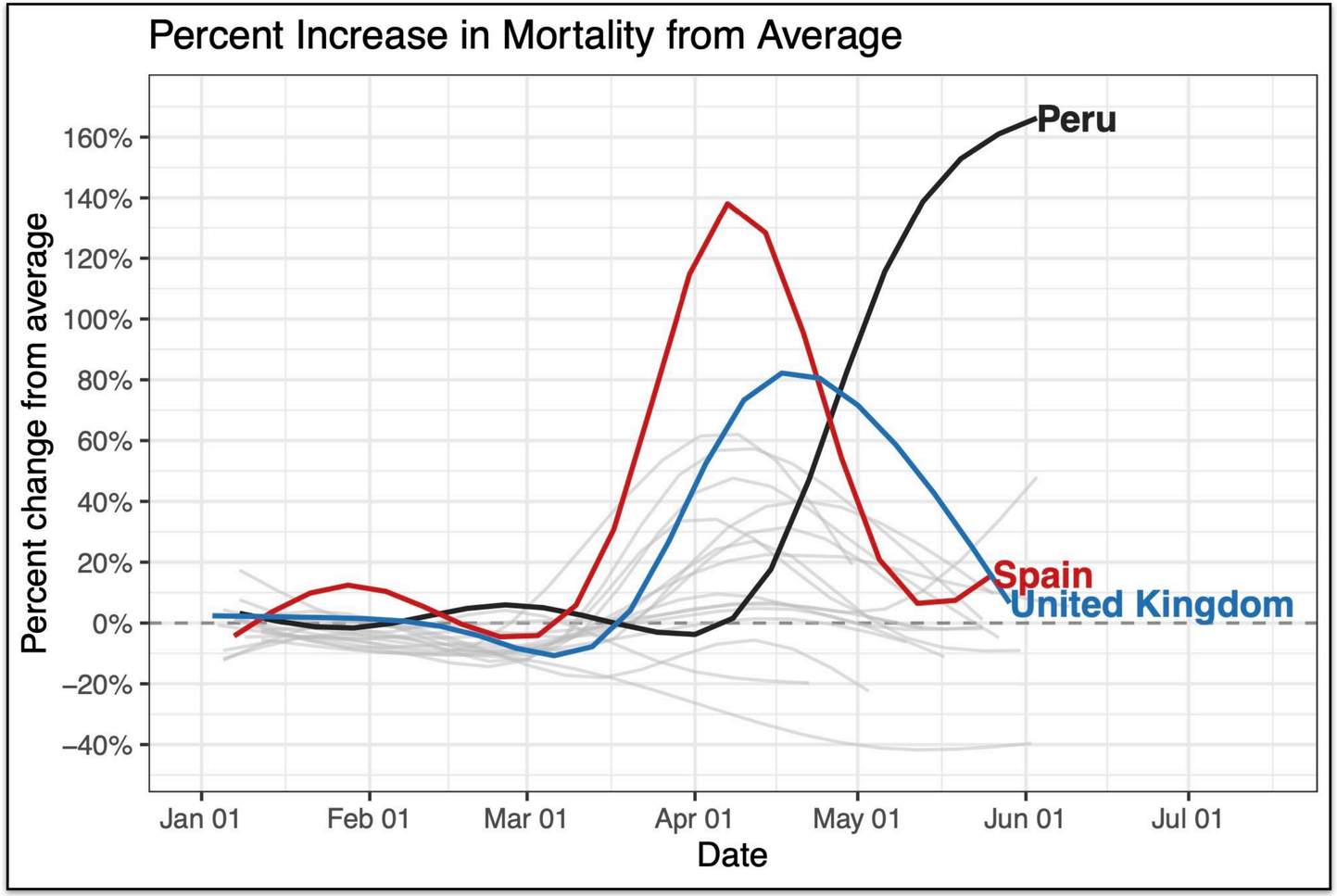
POBLACIÓN ESTIMADA POR ESTADOS IDENTIFICADOS:

Kansas: 2,911,505 / Arkansas 3,013,825 / Nevada 3,034,392 / Iowa 3,156,145 / Utah 3,161,105 / Connecticut 3,572,665 / Puerto Rico 3,193,694

Motivation: Closer to home

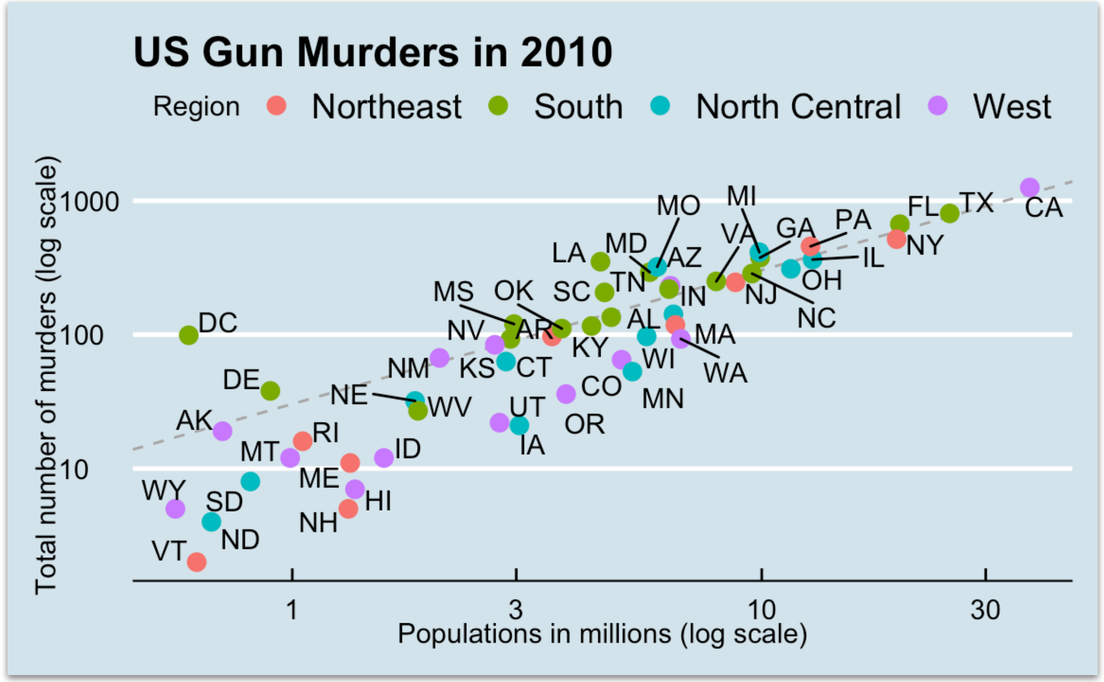


Motivation: Recent example



Components of a figure

- **Data:** The US murders dataset
- **Geometry:** This plot is a *scatterplot*. Other possible geometries are *barplot*, *histogram*, *boxplot*, and others
- **Aesthetic mapping:** Set of instructions that allow us to *map* data to the figure. The two main instructions are the *x* and *y* coordinates that correspond to total number of murders and population size, respectively. Other instructions are color, which is mapped to region, and text. These mappings will depend on the chosen geometry.



ggplot2 objects

- The first step in creating a ggplot2 graph is to create a ggplot2 object

```
ggplot(data = murders)
```

ggplot2 objects

- We can also use the pipe operator

```
murders %>% ggplot()
```

`ggplot2` objects

- This creates a blank plot since no geometry was defined.

```
murders %>% ggplot()
```

`ggplot2` objects

- We can also assign the plot to an object

```
p <- murders %>% ggplot()
```

- The we can render the plot with either of these two statements:

```
print(p)
p
```

ggplot2 syntax

- In ggplot2 we create graphs by adding layers:
  - geometries
  - summary statistics
  - scales
  - styles
- To add layers, we use the symbol +
- A line code will generally look like this:

Image taken from google

```
data %>%
   ggplot() +
   LAYER 1 +
   LAYER 2
```

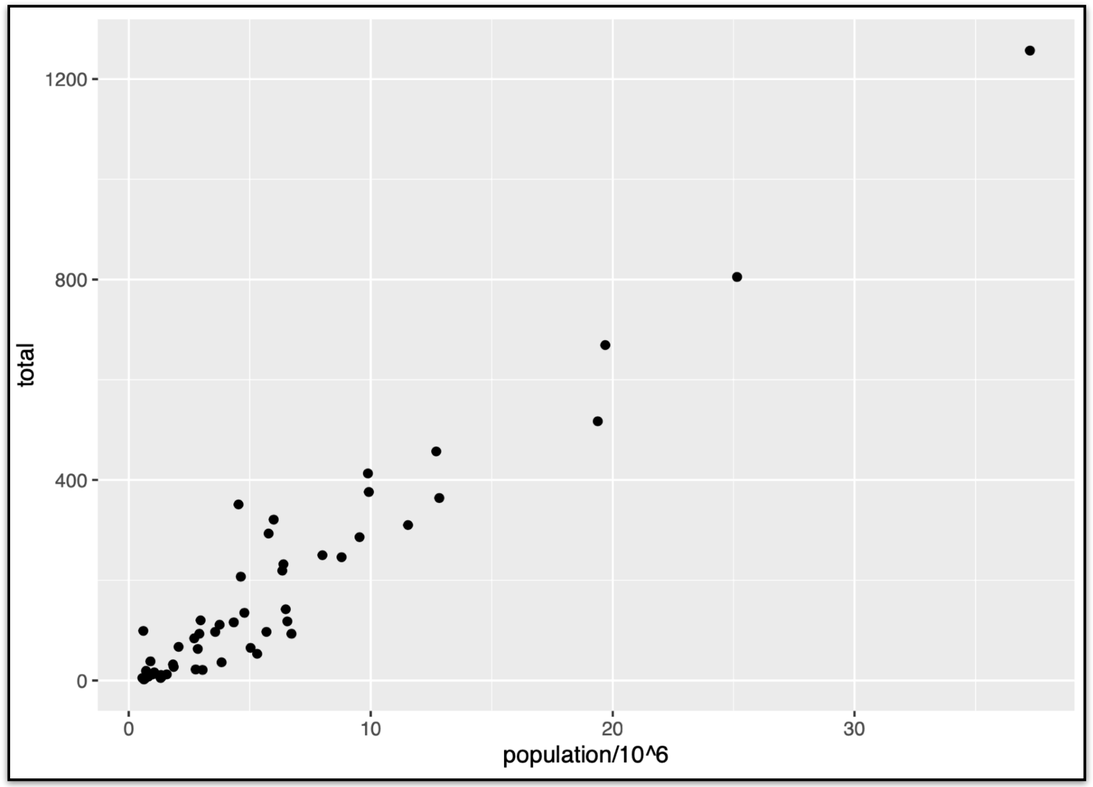
Aesthetic mapping

- Aesthetic mappings describe how properties of the data connect with features of the graph. We do this with the `aes` function

```
murders %>%
   ggplot(aes(x = population/10^6, y = total)) +
   geom_point()
```

- We can drop `x =` and `y =` and get the same the same thing

```
murders %>%
   ggplot(aes(population/10^6, total)) +
   geom_point()
```



Aesthetic mapping

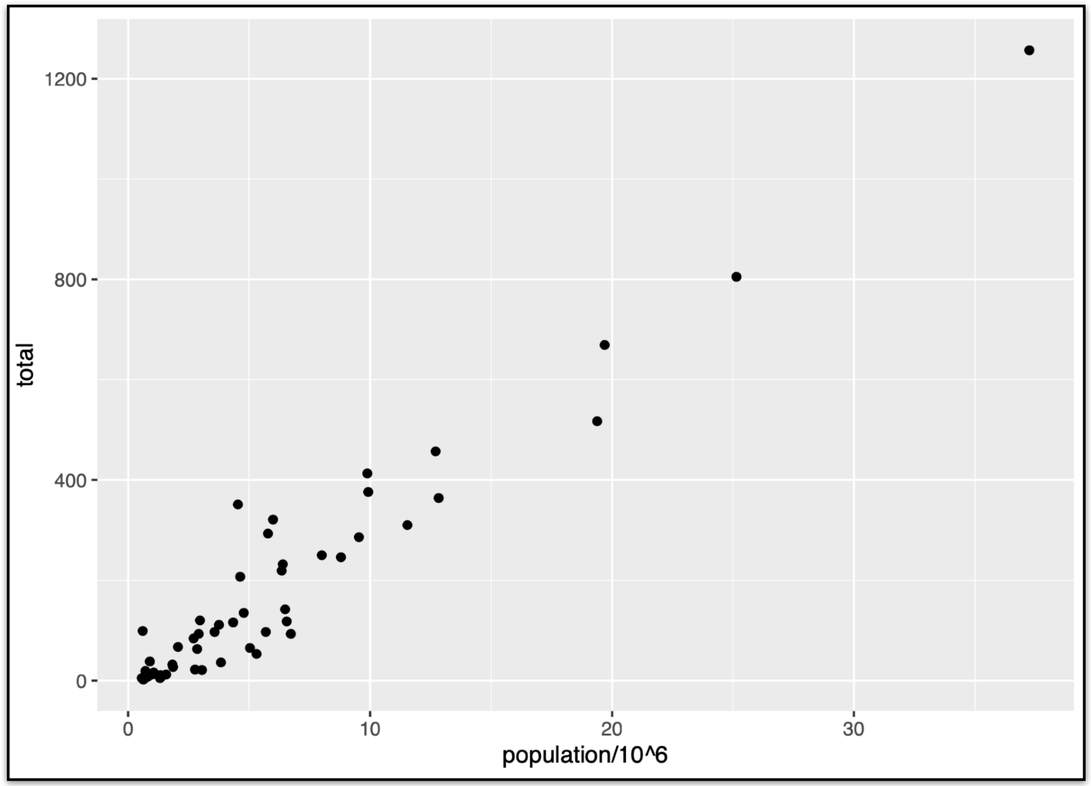
- Alternatively, we can use the object p that we defined earlier:

```
p <- murders %>% ggplot()
```

- Then

```
p +
  geom_point(aes(population/10^6, total))
```

- Note that the labels are defined by default



Adding text

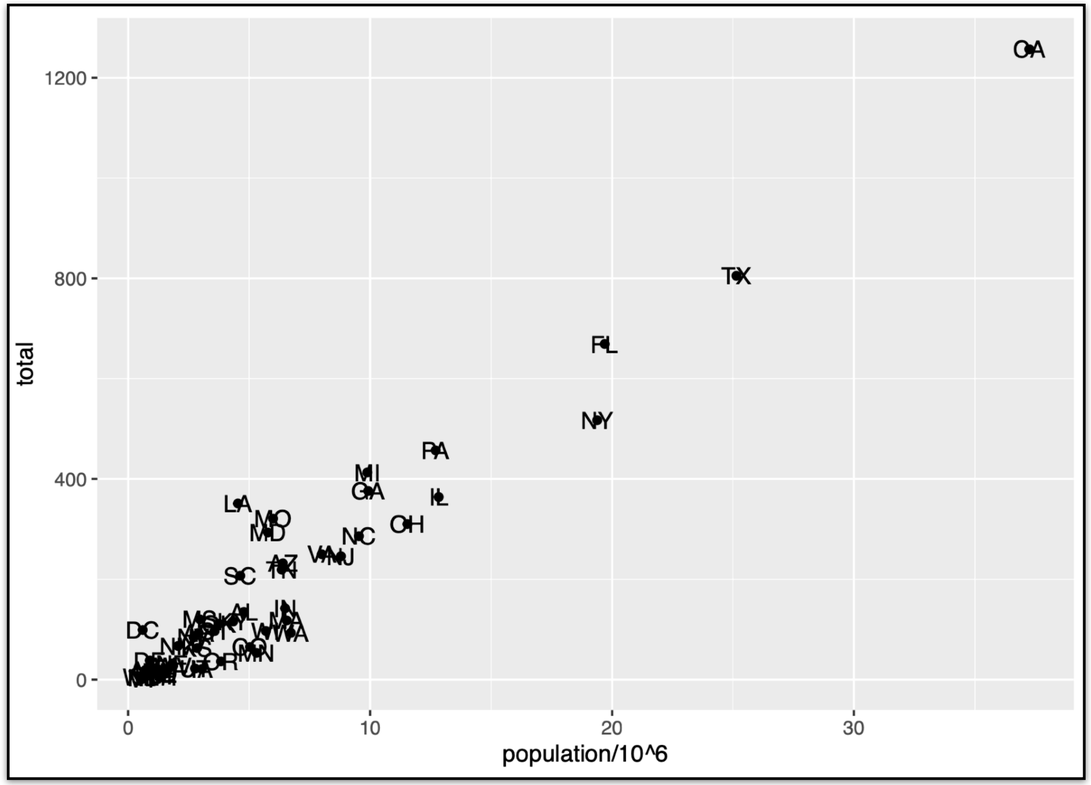
- Now we want to add a second layer with state abbreviations

```
p +
   geom_point(aes(population/10^6, total)) +
   geom_text(aes(population/10^6, total, label=abb))
```

- However, the following code gives an error

```
p +
   geom_point(aes(population/10^6, total)) +
   geom_text(aes(population/10^6, total), label=abb)
```

- because `label` was defined outside of the `aes` function

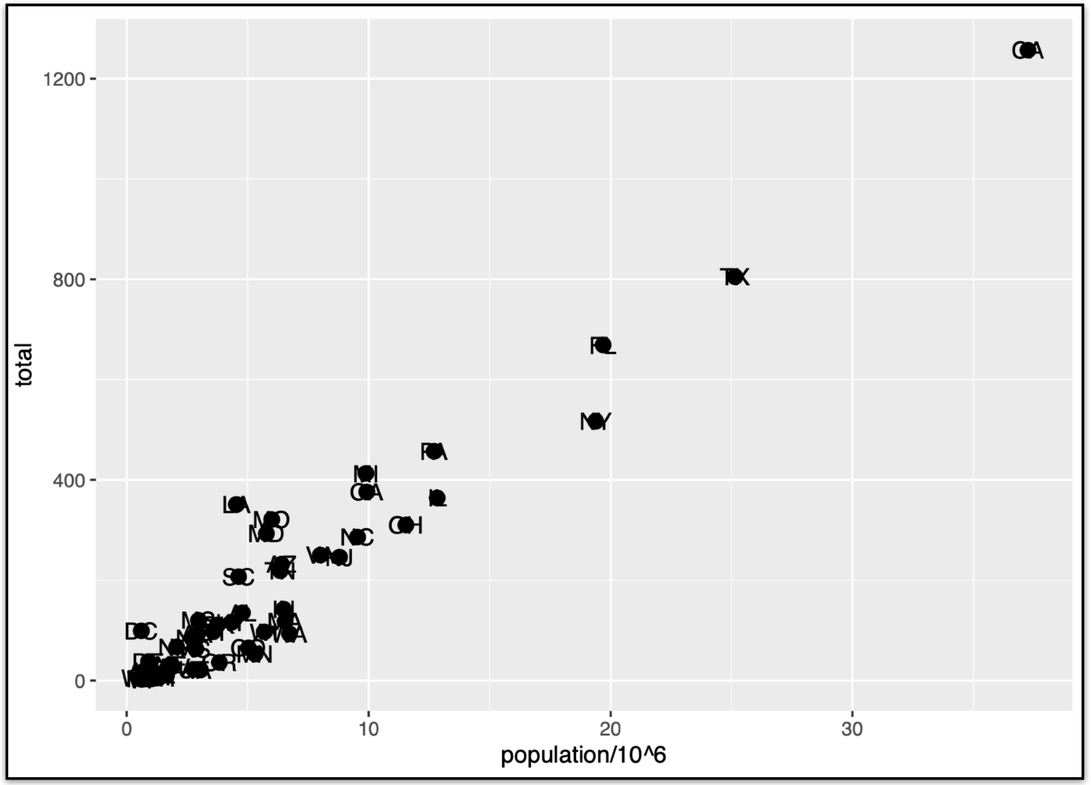


Tinkering with arguments

- Each geometry has many geometry-specific arguments

```
p + geom_point(aes(population/10^6, total), size=3) +
   geom_text(aes(population/10^6, total, label=abb))
```

- Here we make the points bigger

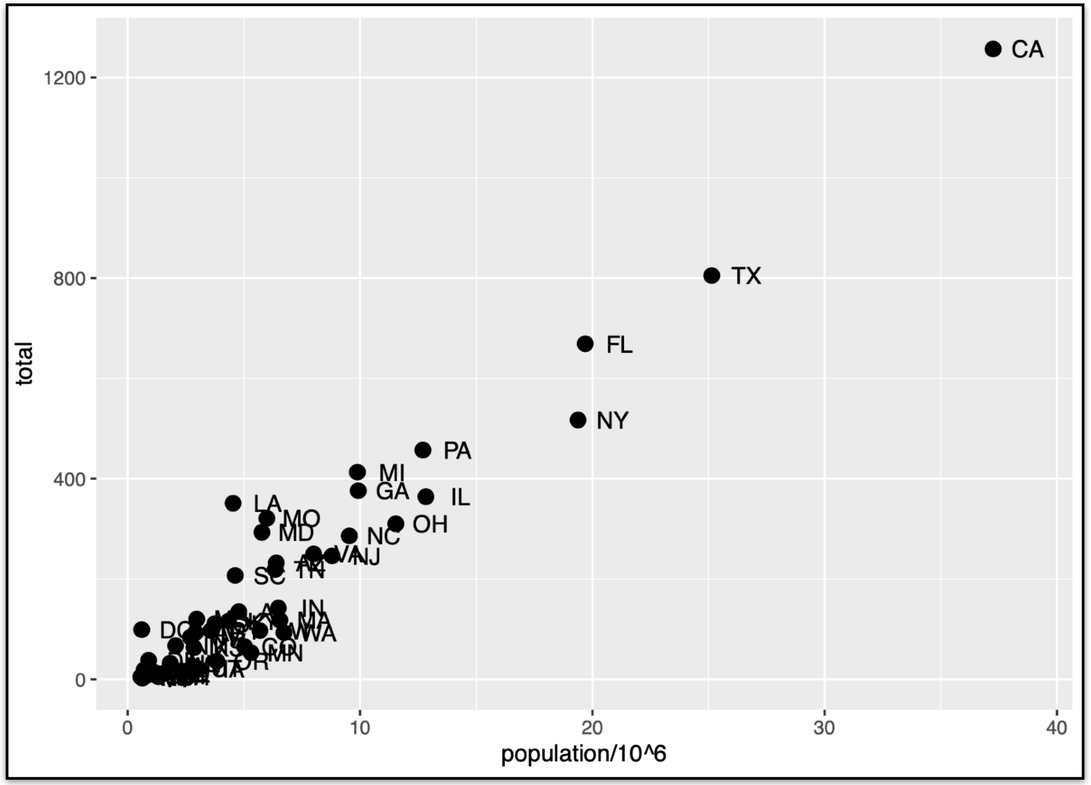


Tinkering with arguments

- Each geometry has many geometry-specific arguments

```
p + geom_point(aes(population/10^6, total), size=3) +
   geom_text(aes(population/10^6, total, label=abb), nudge_x = 1.5)
```

- and here we adjust the text position
- Note the repetition of code



Global vs local aesthetic mappings

- In the previous slide defined the following mapping twice, one for each geometry:

```
aes(population/10^6, total)
```

- These are examples of local aesthetic mappings
- Instead, we can define these mappings globally:

```
p <- murders %>% ggplot(aes(population/10^6, total, label=abb))
```

- Then we can create the plot with the following code:

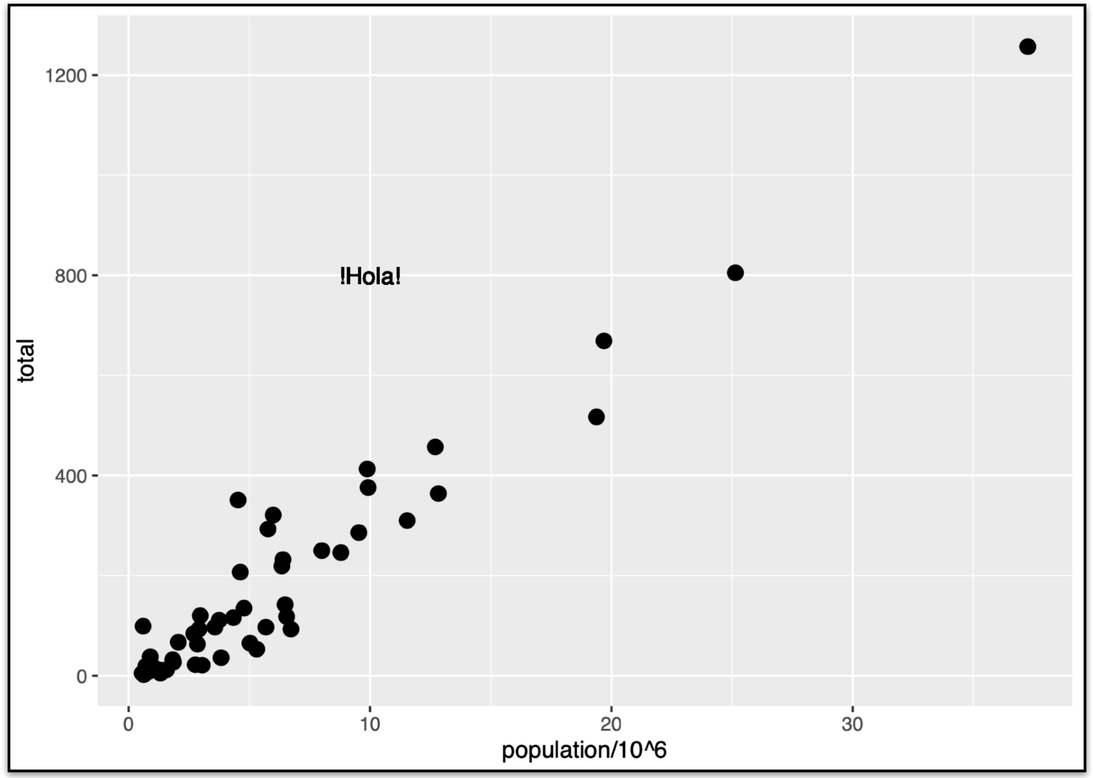
```
p + geom_point(size=3) + geom_text(nudge_x = 1.5)
```

- We can also overwrite global mappings by defining new mappings within layers:

```
p + geom_point(size=3) + geom_text(aes(x=10, y=800, label="!Hola!"))
```

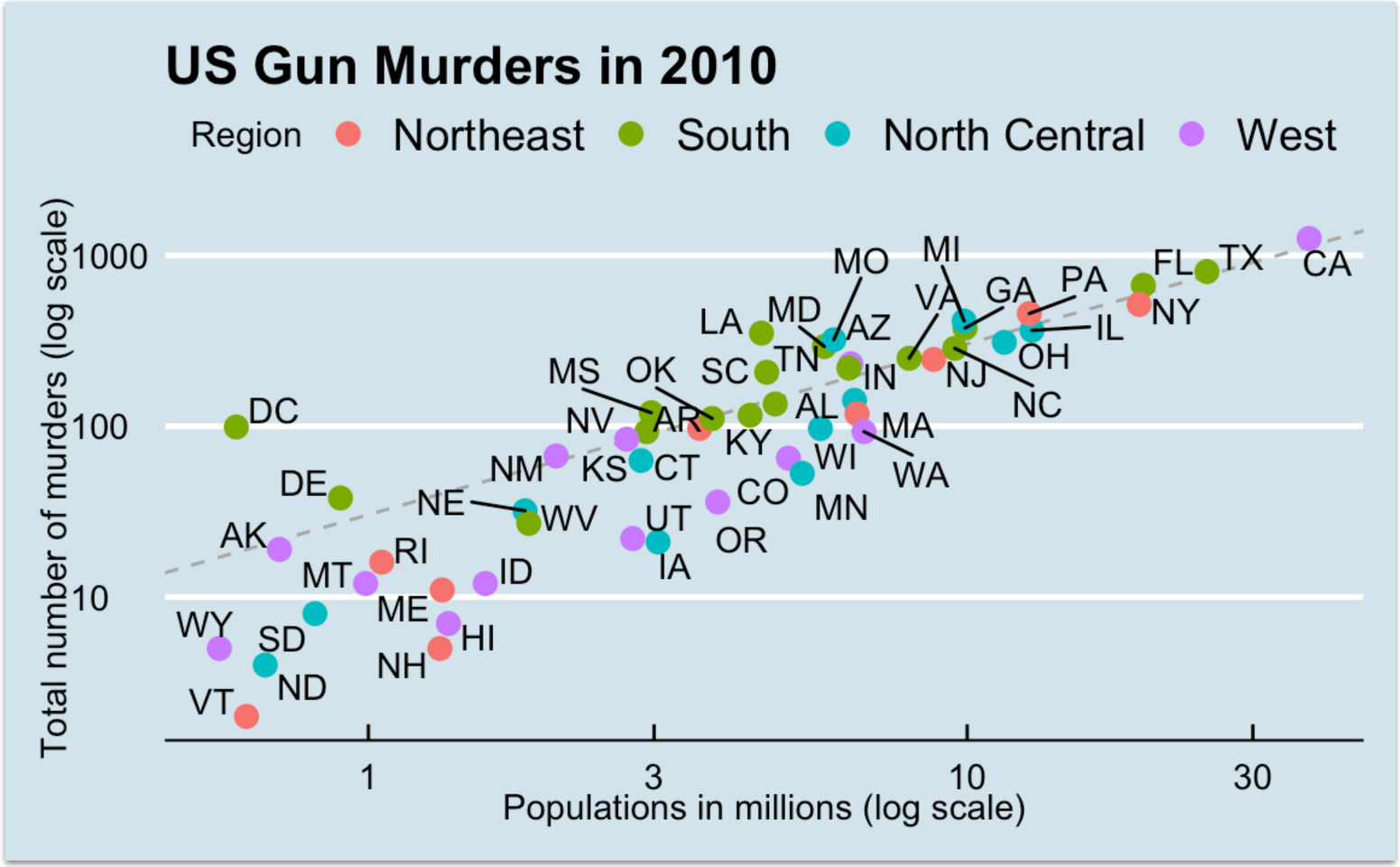
Global vs local aesthetic mappings

```
p + geom_point(size=3) + geom_text(aes(x=10, y=800, label="!Hola!"))
```



Scales

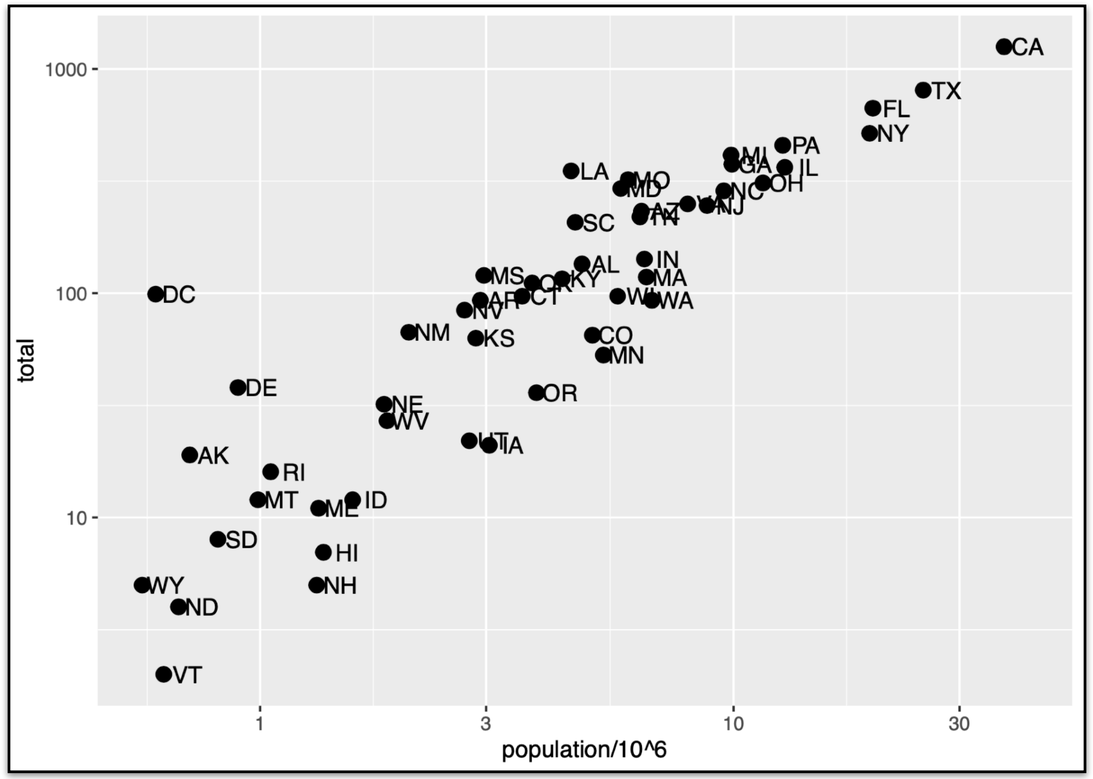
- Recall the figure we want to recreate:



Scales

- Let's transform the *x*- and *y-axis*:

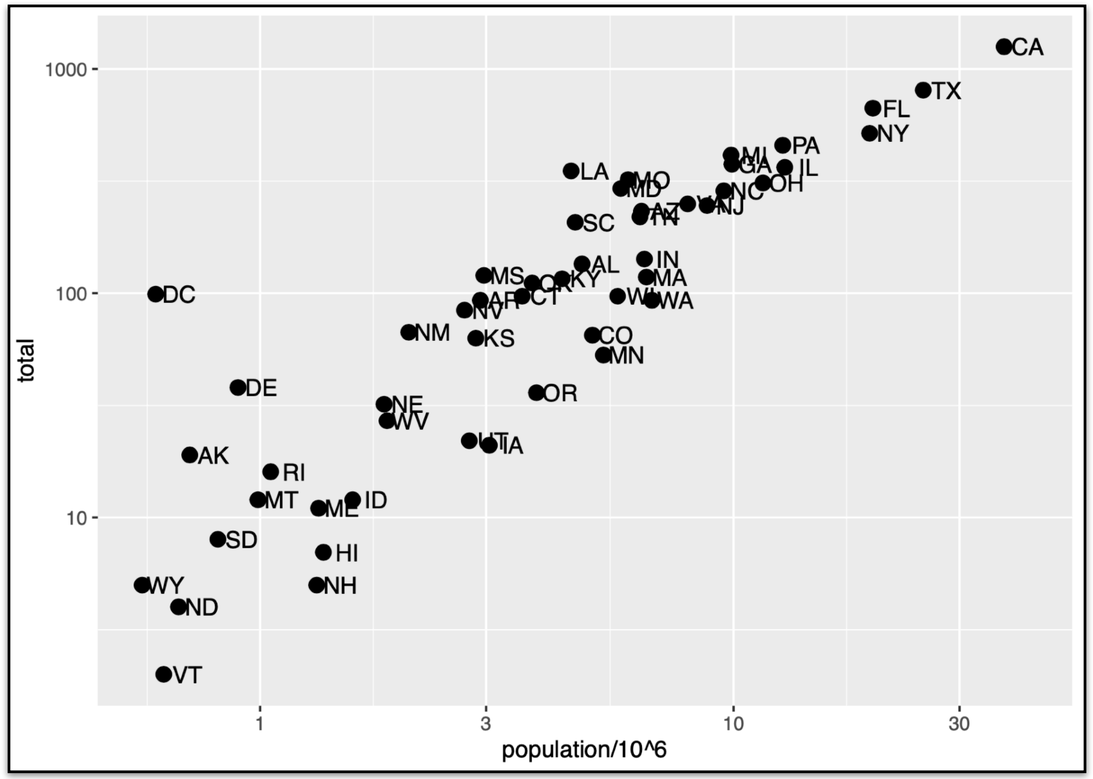
```
p + geom_point(size=3) + geom_text(nudge_x = 0.05) +
   scale_x_continuous(trans = “log10") + scale_y_continuous(trans = "log10")
```



Scales

- Alternatively, we can use the following code:

```
p + geom_point(size=3) + geom_text(nudge_x = 0.05) +
   scale_x_log10() + scale_y_log10()
```



Labels and titles

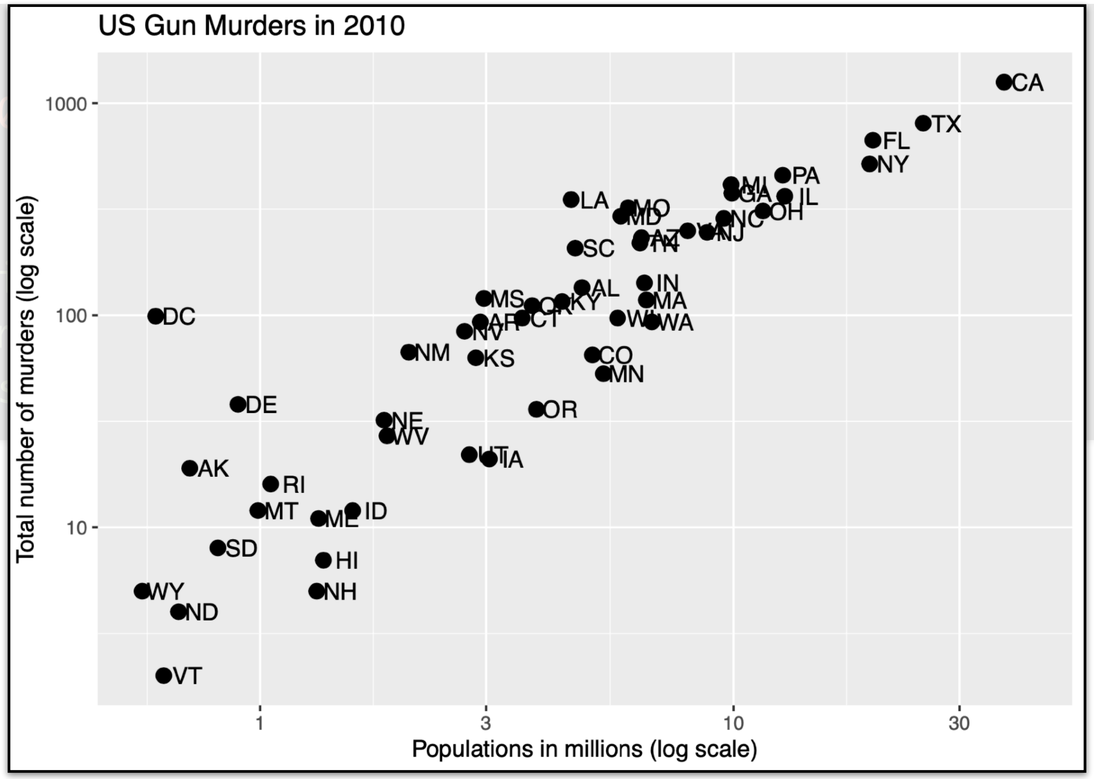
- Alternatively, we can use the following code:

```
p + geom_point(size=3) +
   geom_text(nudge_x = 0.05) +
   scale_x_log10() +
   scale_y_log10() +
   xlab("Populations in millions (log scale)") +
   ylab("Total number of murders (log scale)") +
   ggtitle("US Gun Murders in 2010")
```

Labels and titles

- Alternatively, we can use the following code:

```
p + geom_point(size=3) +
  geom_text(nudge_x = 0.
  scale_x_log10() +
  scale_y_log10() +
  xlab("Populations in m
  ylab("Total number of
  ggtitle("US Gun Murder
```



- We are close!

Adding color

- Let's start by redefining the object `p`

```
p <- murders %>%
   ggplot(aes(population/10^6, total, label=abb)) +
   geom_text(nudge_x = 0.05) +
   scale_x_log10() +
   scale_y_log10() +
   xlab("Populations in millions (log scale)") +
   ylab("Total number of murders (log scale)") +
   ggtitle("US Gun Murders in 2010")
```

- We can use the argument `color` to change the color of points:

```
p + geom_point(size = 3, color = “blue”)
```

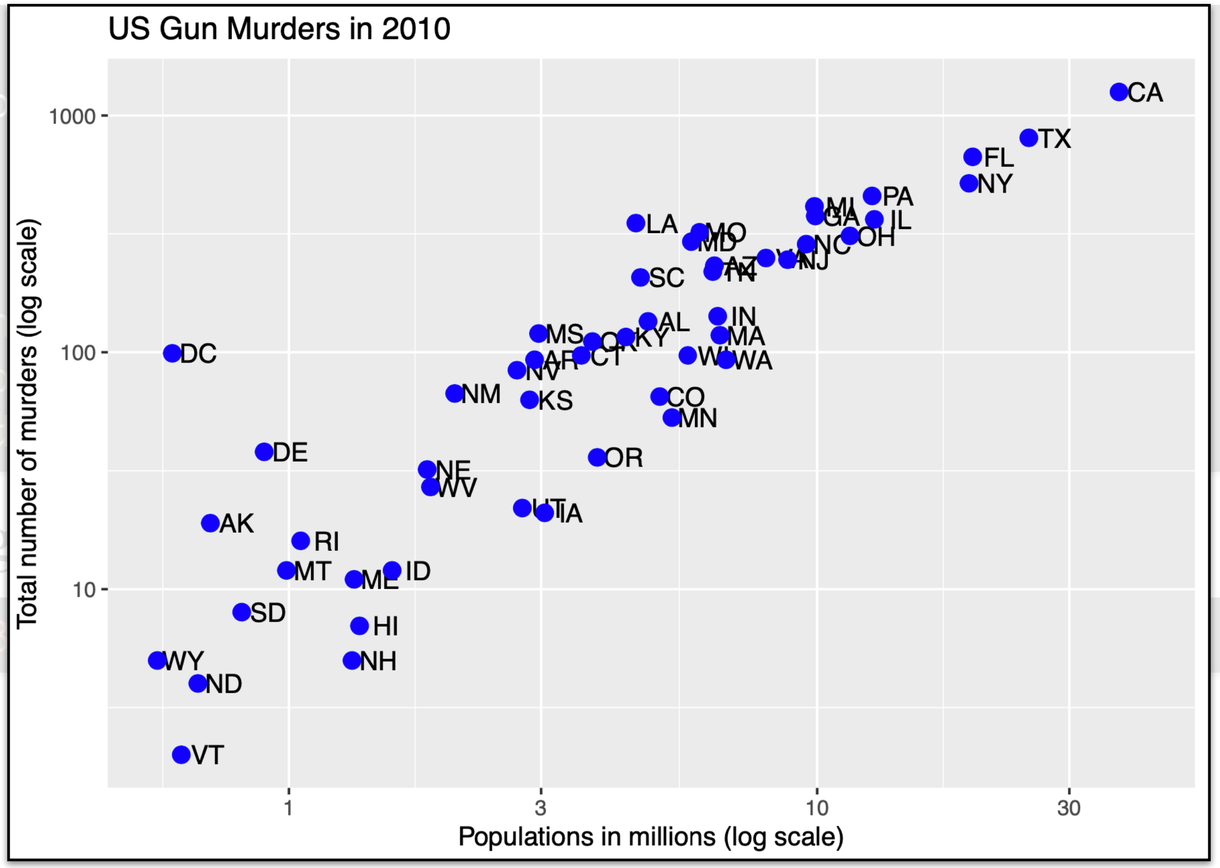
Adding color

- Let's start by redefining the object p

```
p <- murders %>%
  ggplot(aes(populatio
  geom_text(nudge_x =
  scale_x_log10() +
  scale_y_log10() +
  xlab("Populations in
  ylab("Total number o
  ggtitle("US Gun Murd
```

- We can use the arg

```
p + geom_point(size = 3
```



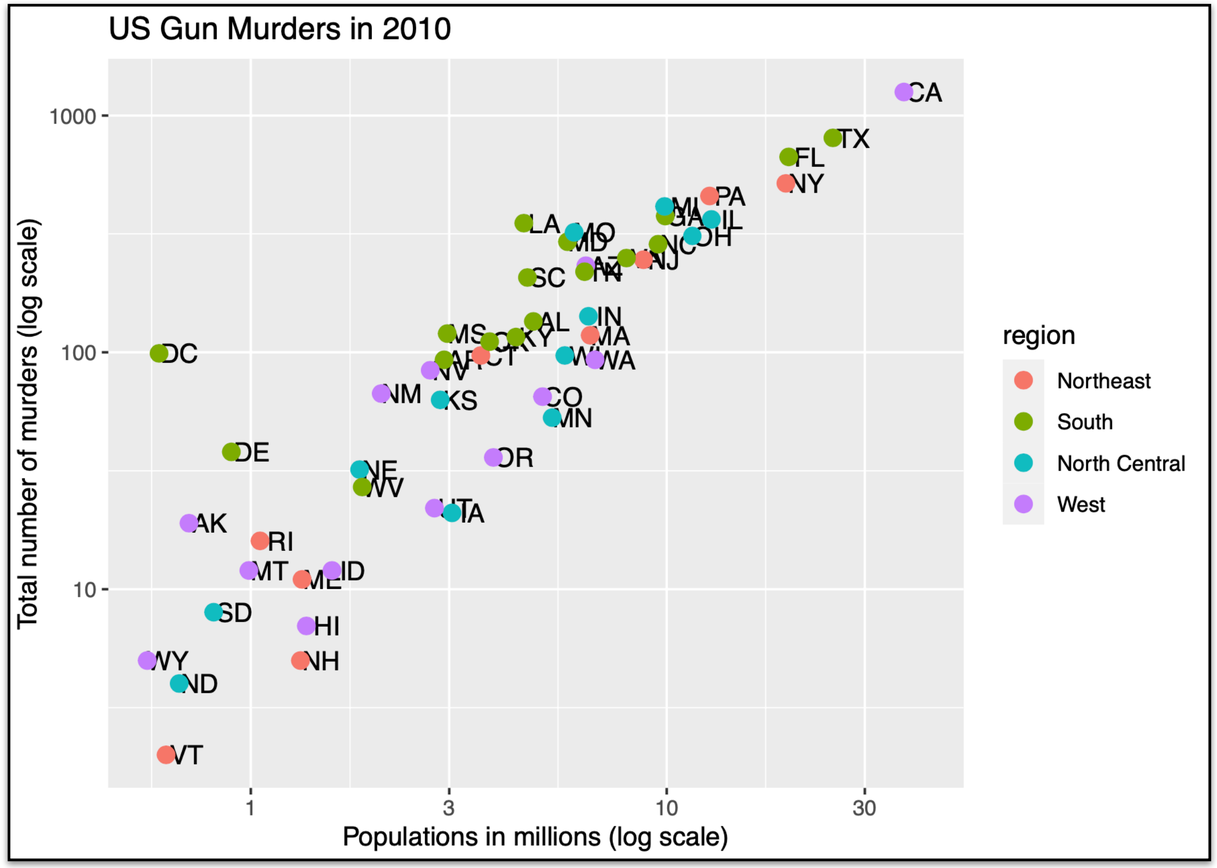
- However, we want to add color by region

Adding color

- Since the choice of color depends on a feature of the observation we have to use `aes`

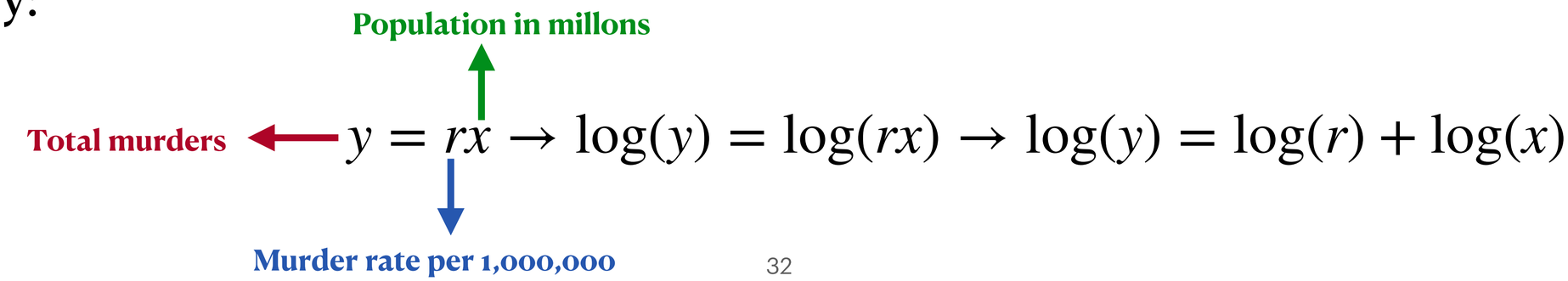
```
p + geom_point(aes(color = region), size = 3)
```

- Note that `ggplot2` automatically adds a legend to the plot



Annotations, shapes, and adjustments

- Often we want to add annotations to figures that are derive from the aesthetic mapping:
  - Lines
  - Labels
  - Boxes
  - Shaded areas
- Here we want to add a line that represents the average murder rate for the entire country:



Annotations, shapes, and adjustments

- Let's start by computing `r` (the average murder rate per million for the US)

```
r <- murders %>%
   summarize(rate = sum(total) /  sum(population) * 10^6) %>%
   pull(rate)
```

- Now we can add a line with `geom_abline`:

```
p + geom_point(aes(col=region), size = 3) +
   geom_abline(intercept = log10(r))
```

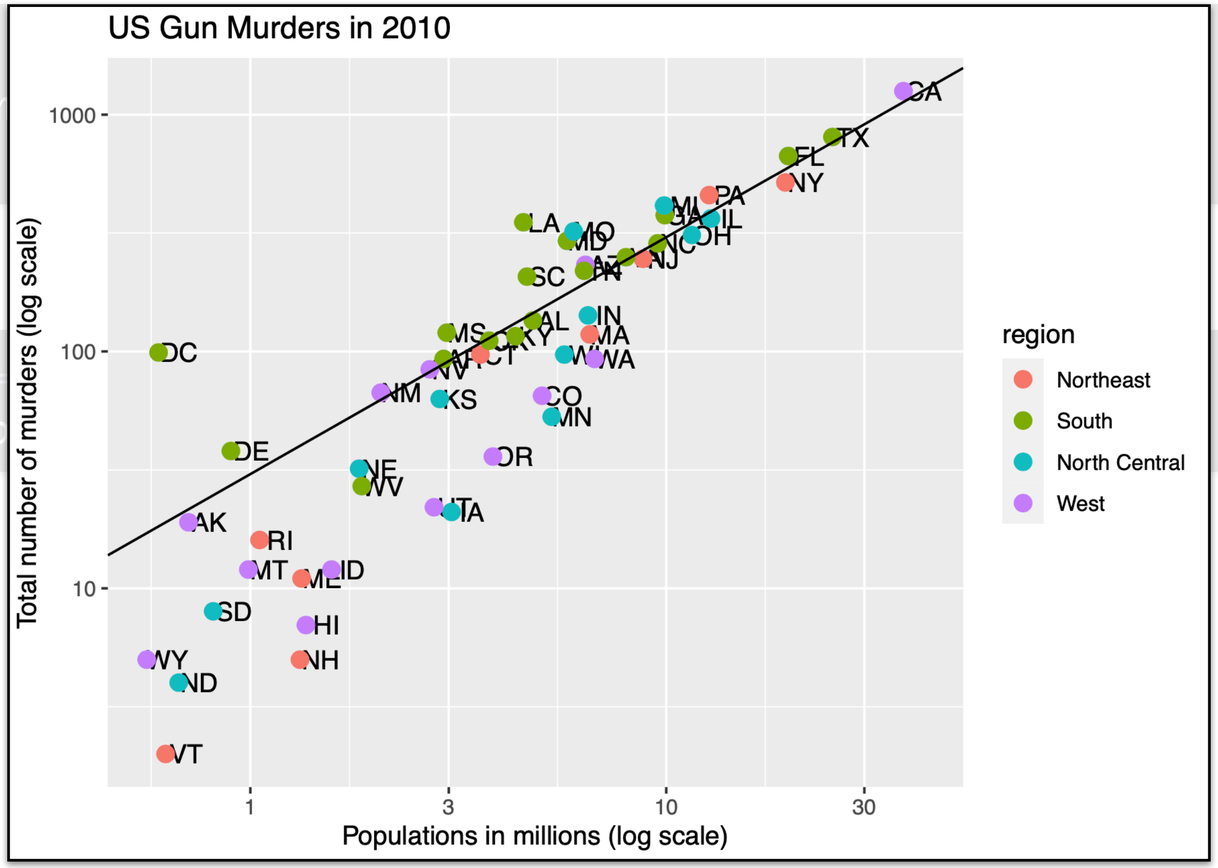
Annotations, shapes, and adjustments

- Let's start by computing `r` (the average murder rate per million for the US)

```
r <- murders %>%
  summarize(rate = sum
  pull(rate)
```

- Now we can add a

```
p + geom_point(aes(col=
  geom_abline(intercep
```



Annotations, shapes, and adjustments

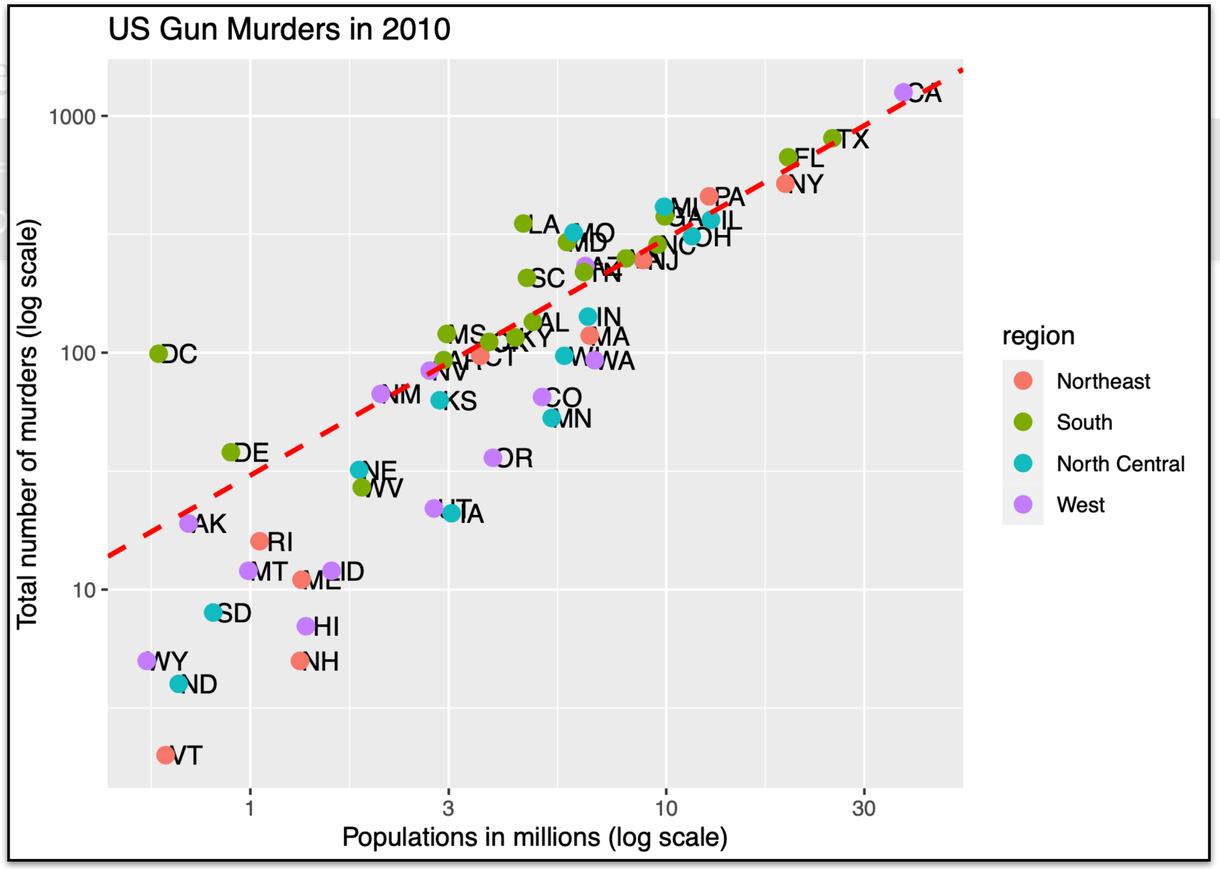
- We can change the color, shape, and size of the line by changing the arguments
- Here is an example:

```
p + geom_point(aes(col=region), size = 3) +
   geom_abline(intercept = log10(r), color="red", size=1, linetype=2)
```

Annotations, shapes, and adjustments

- We can change the color, shape, and size of the line by changing the arguments
- Here is an example

```
p + geom_point(aes(col=
  geom_abline(intercep
```



- Note that the line goes over the data points
- We want the line to lie on the background

Annotations, shapes, and adjustments

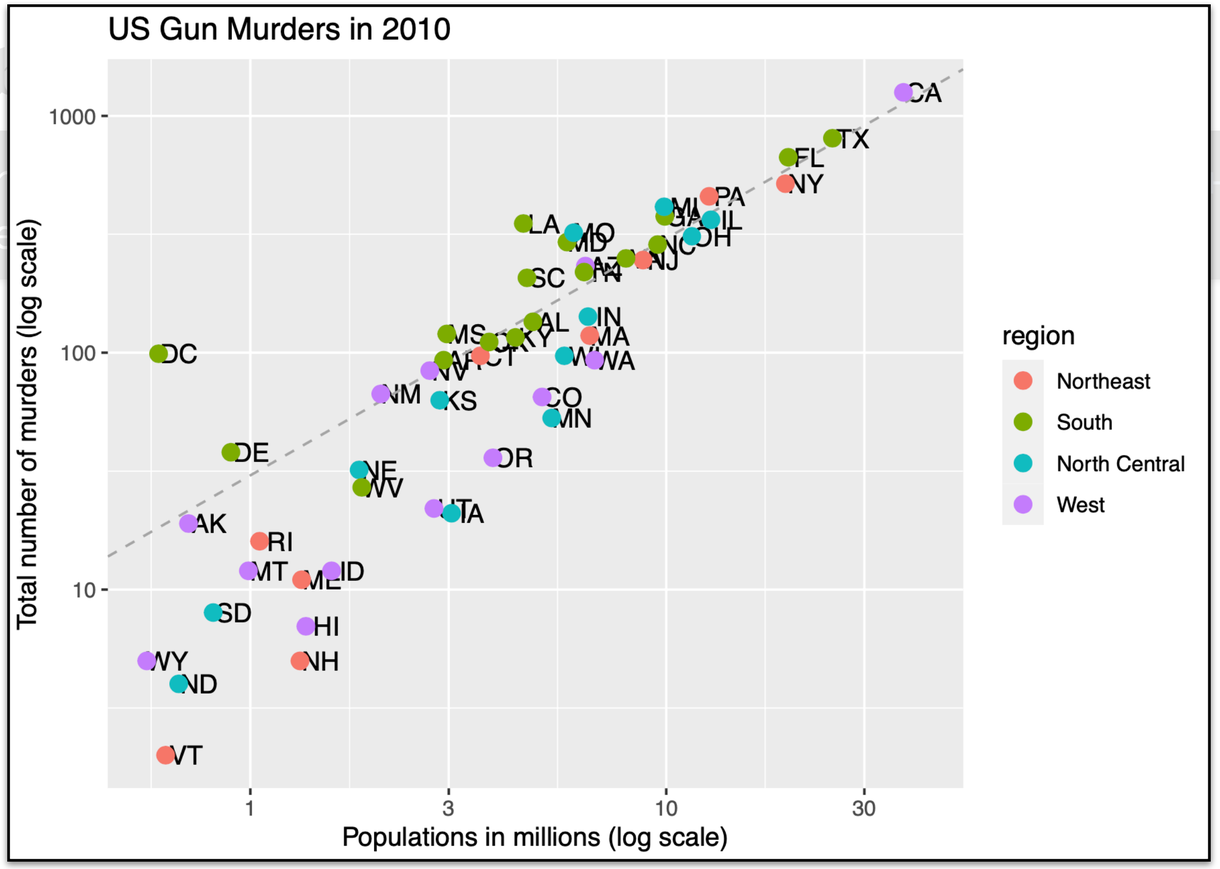
- We can fix this by defining the line first.
- Let's update the object `p`:

```
p <- p + geom_abline(intercept = log10(r), lty = 2, color = "darkgrey") +
  geom_point(aes(col=region), size = 3)
```

Annotations, shapes, and adjustments

- We can fix this by defining the line first.
- Let's update the ob

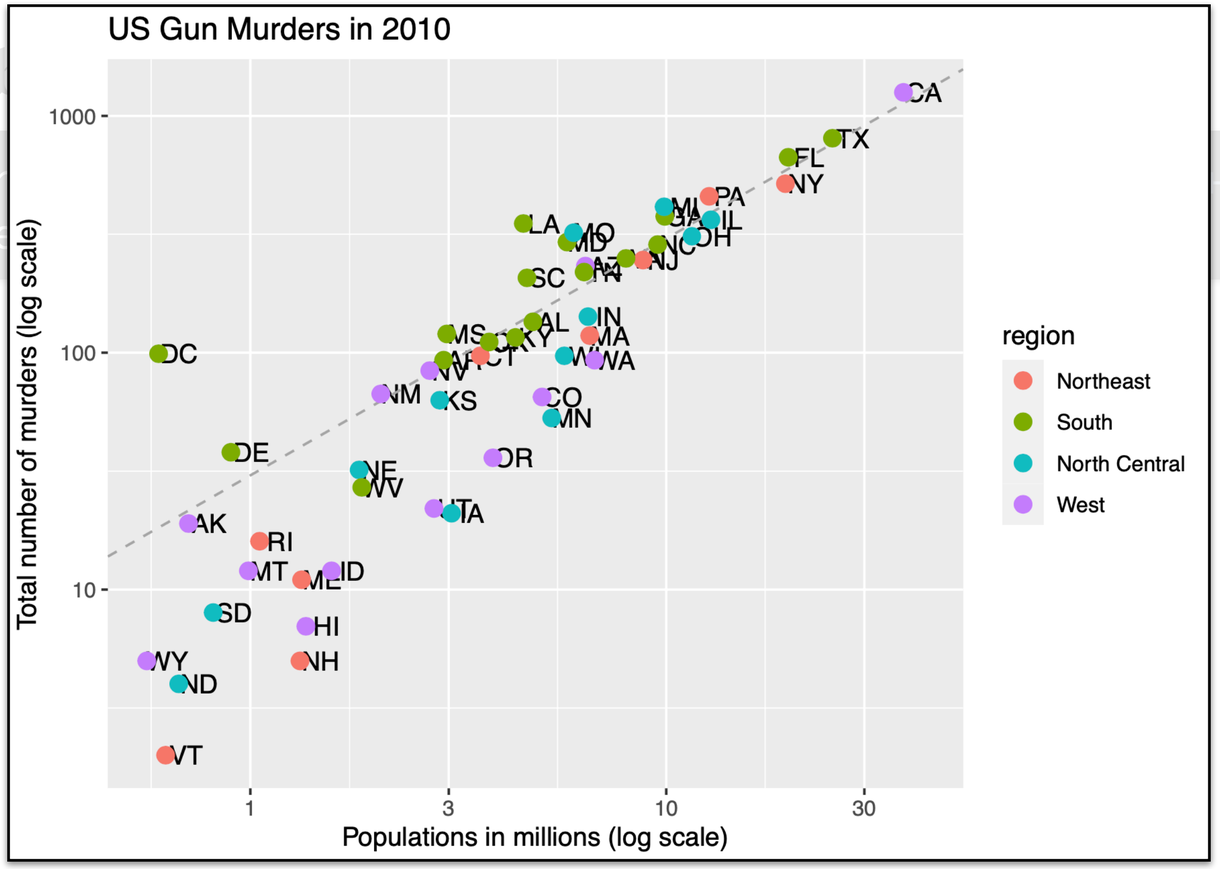
```
p <- p + geom_abline(in
  geom_point(aes(col=re
```



Annotations, shapes, and adjustments

- We can fix this by defining the line first.
- Let's update the ol

```
p <- p + geom_abline(ir
  geom_point(aes(col=re
```



- One last thing, let's capitalize the legend title

Annotations, shapes, and adjustments

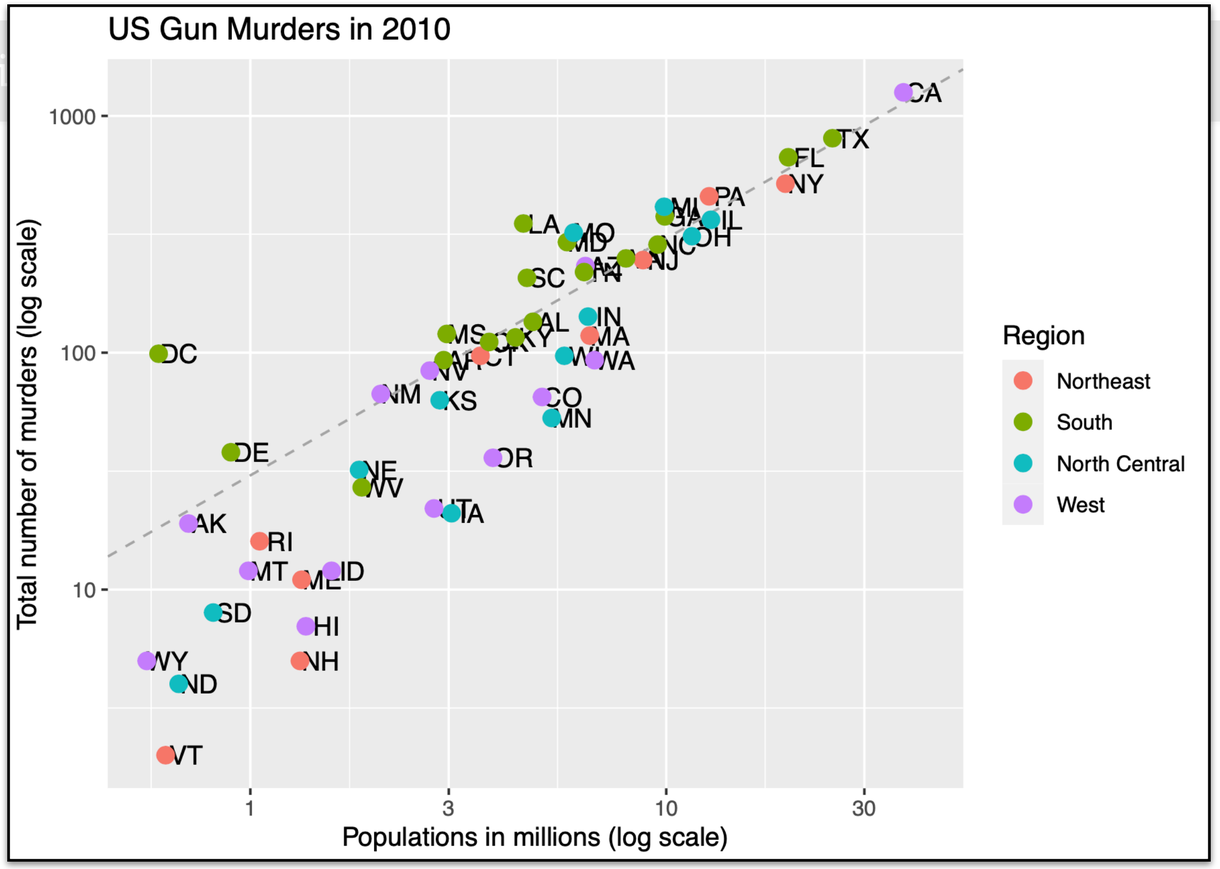
- We can capitalize the legend title with the following code:

```
p <- p + scale_color_discrete(name = "Region")
```

Annotations, shapes, and adjustments

- We can capitalize the legend title with the following code:

```
p <- p + scale_color_di
```



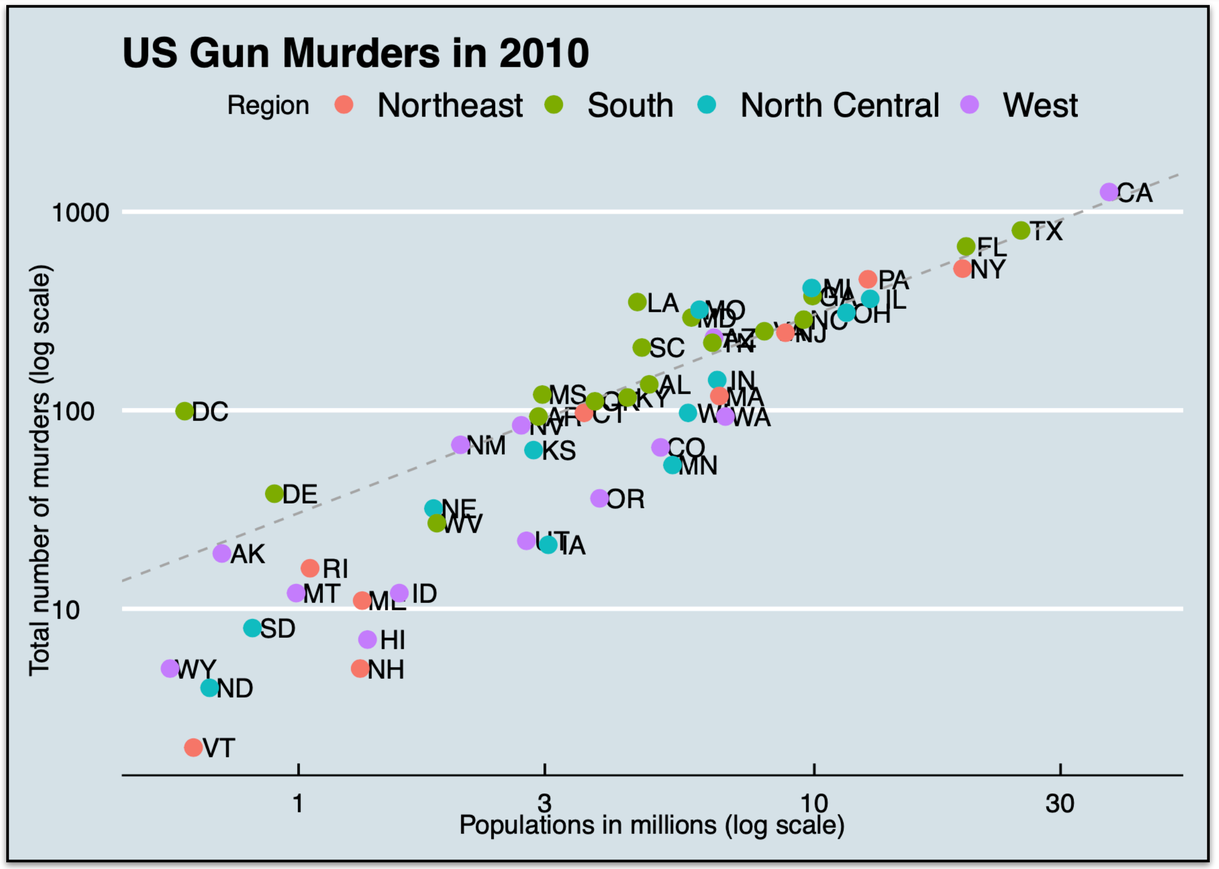
Add-on packages

- `ggplot2` is augmented by other packages that expand its capabilities
- The remaining details for the desire figure require two packages
  - `ggthemes`
  - `ggrepel`
- The style of `ggplot2` graph can be change with the `theme` functions
- `ggplot2` already includes a lot of themes
- The package `ggthemes` provides much more

Add-on packages

- The theme we want in `theme_economist` from `ggthemes`

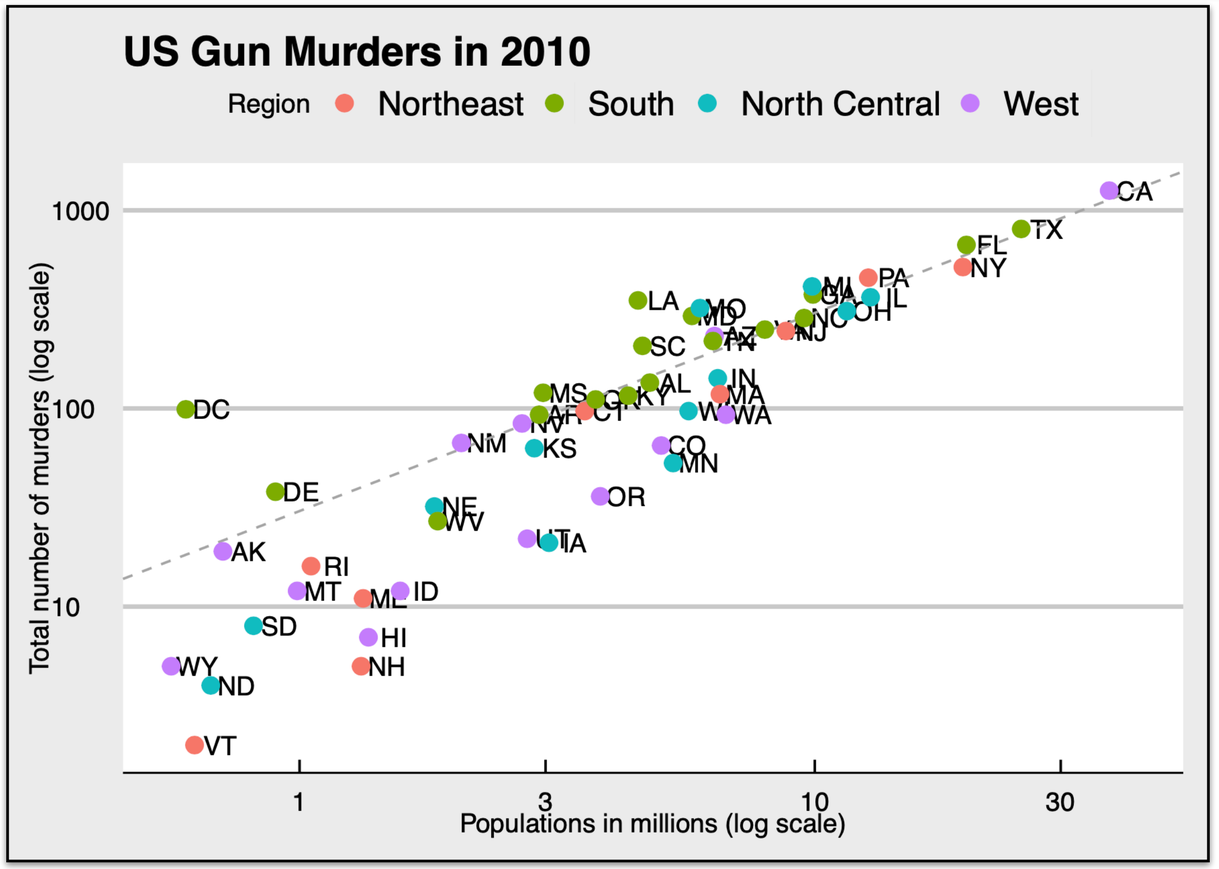
```
library(ggthemes)
p + theme_economist()
```



Add-on packages

- Other themes

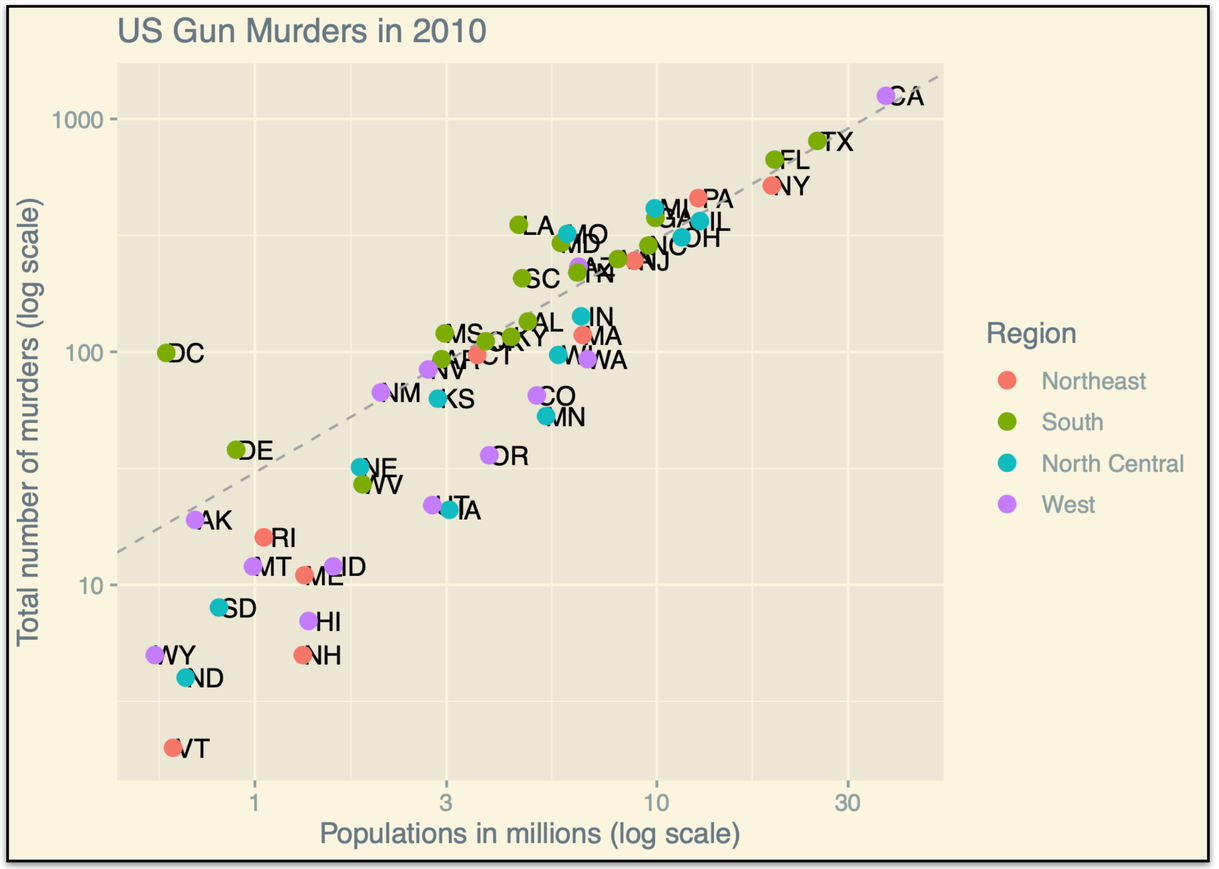
```
library(ggthemes)
p + theme_economist_white()
```



Add-on packages

- Other themes

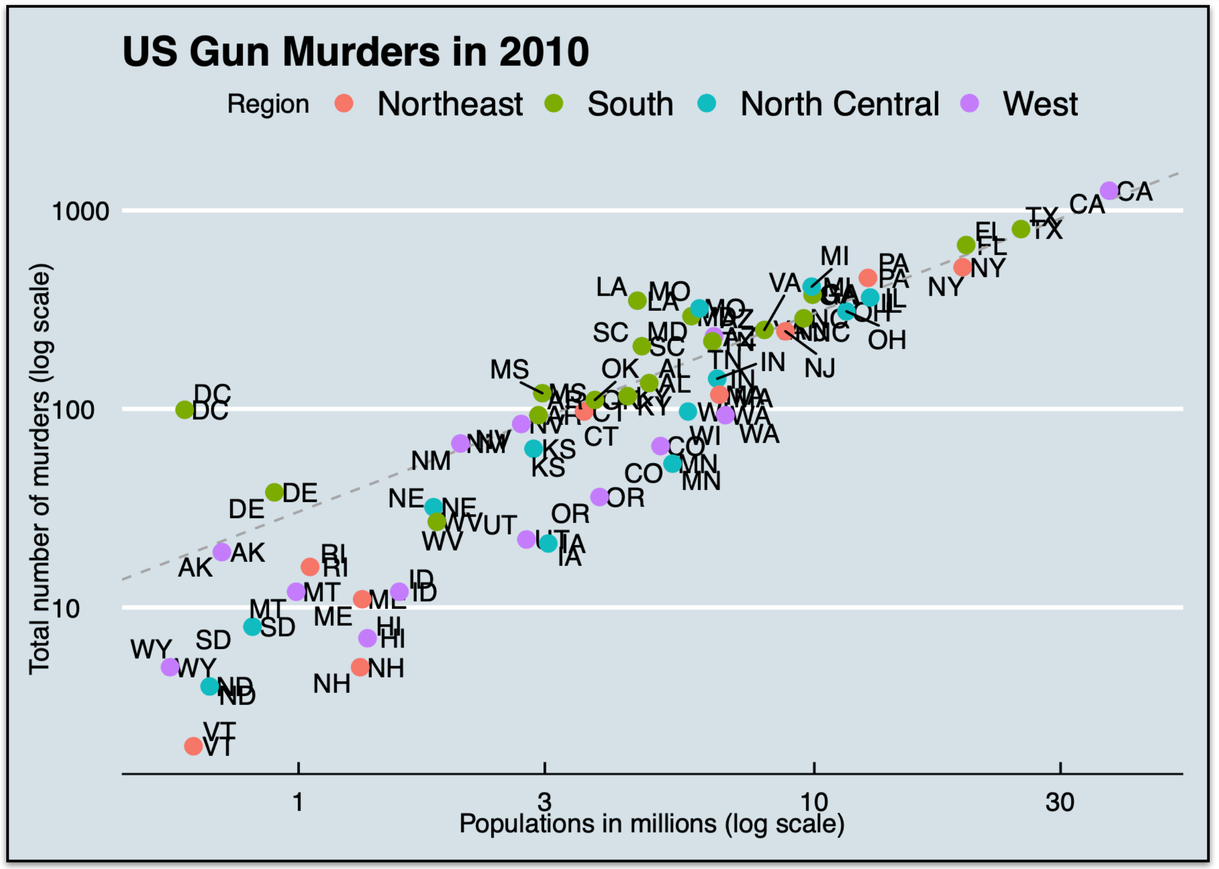
```
library(ggthemes)
p + theme_solarized_2()
```



Add-on packages

- Finally, let's use the `ggrepel` package to adjust the text

```
library(ggrepel)
p + theme_economist() + geom_text_repel()
```



Add-on packages

- Finally, let's use the `ggrepel` package to adjust the text

```
library(ggrepel)
p + theme_economist() + geom_text_repel()
```

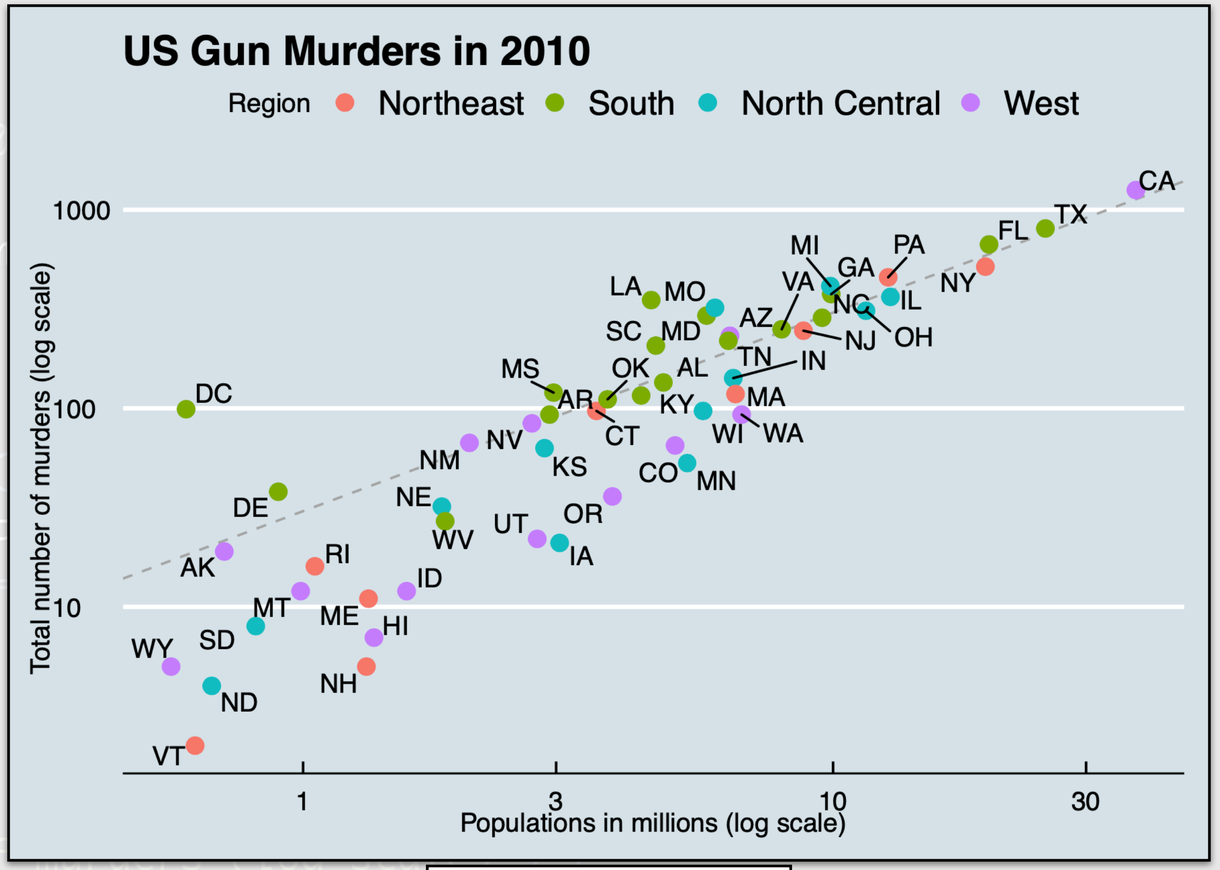
Note the repeated text. Any ideas on why this is happening?

Putting it all together

```
# Libraries
library(ggthemes)
library(ggrepel)
library(dslabs)
data("murders")

# Murder rate per 1,000,000
r <- murders %>%
   summarize(rate = sum(total) /  sum(population) * 10^6) %>%
   pull(rate)

# Code for plot
murders %>% ggplot(aes(population/10^6, total, label = abb)) +
  geom_abline(intercept = log10(r), lty = 2, color = "darkgrey") +
  geom_point(aes(color=region), size = 3) +
  geom_text_repel() +
  scale_x_log10() +
  scale_y_log10() +
  xlab("Populations in millions (log scale)") +
  ylab("Total number of murders (log scale)") +
  ggtitle("US Gun Murders in 2010") +
  scale_color_discrete(name = "Region") +
  theme_economist()
```

Putting it all together

```
# Libraries
library(ggthemes)
library(ggrepel)

# Murder rate per 1,000
r <- murders %>%
   summarize(rate = sum
   pull(rate)

# Code for plot
murders %>% ggplot(aes(
  geom_abline(intercept
  geom_point(aes(color=
  geom_text_repel() +
  scale_x_log10() +
  scale_y_log10() +
  xlab("Populations in
  ylab("Total number of
  ggtitle("US Gun Murders in 2010") +
  scale_color_discrete(name = "Region") +
  theme_economist()
```

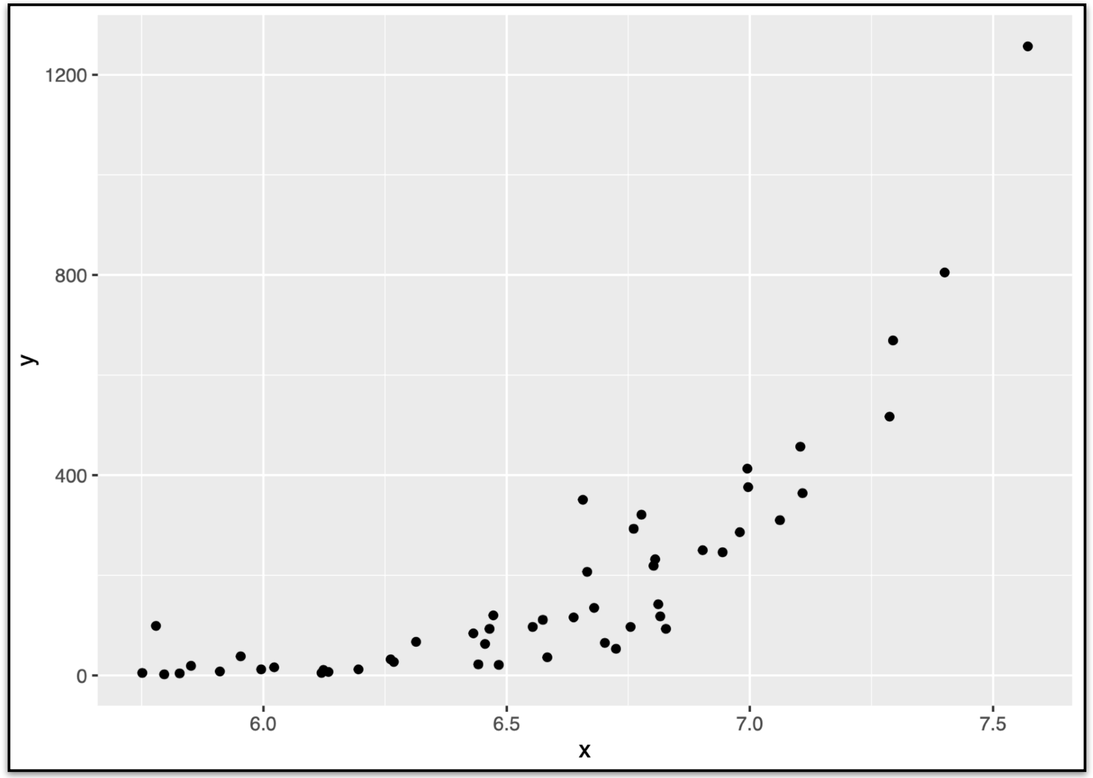


We are done!

Ñapa!

- The function `qplot` allows us to quickly generate a plot:

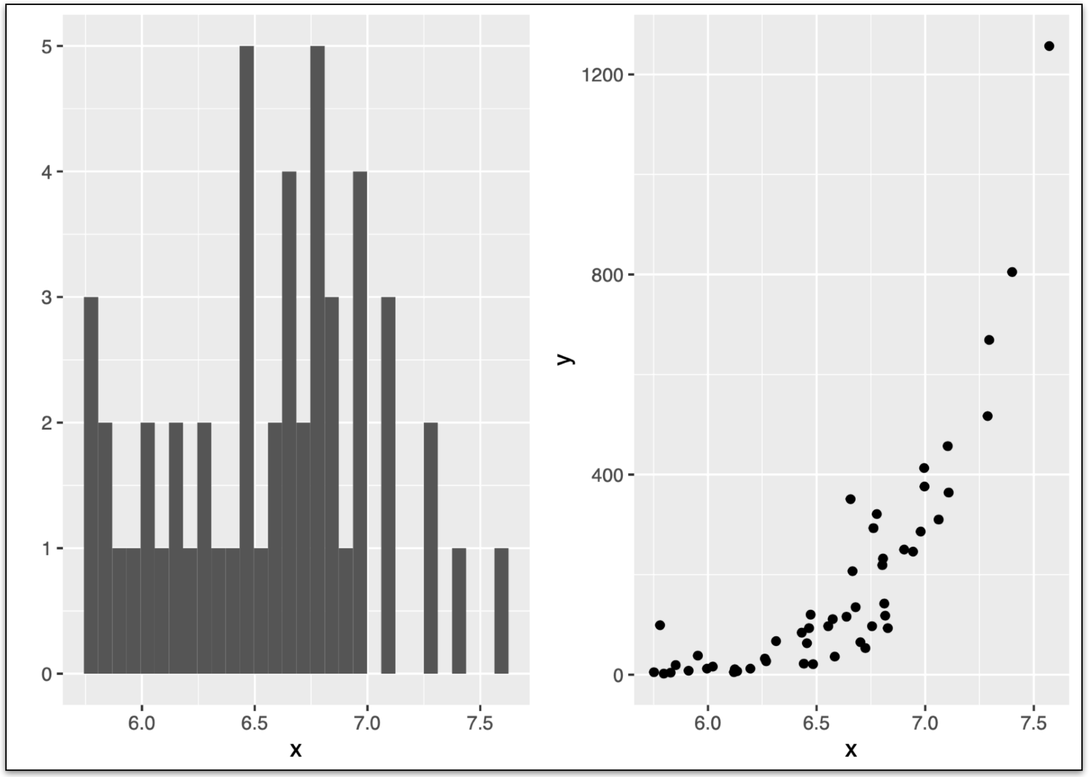
```
x <- log10(murders$population)
y <- murders$total
qplot(x, y)
```



Ñapa!

- The package gridExtra allow us to include plots next to each other

```
library(gridExtra)
p1 <- qplot(x)
p2 <- qplot(x,y)
grid.arrange(p1, p2, ncol = 2)
```



References

1. Introduction to Data Science: Data analysis and prediction algorithms with R by Rafael A. Irizarry, Chapter 7. https://rafalab.github.io/dsbook/
2. R for Data Science by Grolemund & Wickham, Chapter 3. https://r4ds.had.co.nz/index.html
3. ggplot2: Elegant graphics for data analysis: Wickham. https://ggplot2-book.org

Referencias en español:

1. Introducción a la Ciencia de Datos: Análisis de datos y algoritmos de predicción con R por Rafael A. Irizarry, Capítulo 7. https://rafalab.github.io/dslibro/
2. R para Ciencia de Datos por Grolemund & Wickham, Capítulo 3. https://es.r4ds.hadley.nz