

CACCS: Secuencia de taller de RStudio – Parte 2

Rashid C.J. Marcano Rivera

12 de feb.^o de 2025

Contents

Visualización de datos: porqués	1
ggplot2	5
Componentes de un gráfico	5
Lienzo vacío y capas	6
El gráfico que buscábamos	16
Gráficos rápidos: <code>qplot()</code>	17
Alcanzando visualización de densidades y sus cambios a través del tiempo	25
Datos del Covid	37
Datos del censo	40
Recapitulando	50
Qué aprendimos en este taller inicial	50
Continuamos el próximo miércoles	51

Visualización de datos Este es el segundo de cuatro talleres secuenciales de R a través de RStudio. Este taller está basado en viñetas de RStudio con tidyverse (en específico `ggplot2`,) así como elementos y ejemplos del libro de Rafael Irizarry disponible aquí o de manera similar a los html producidos en este taller y más actualizada aquí. Finalmente la parte del censo proviene en parte de código ajustado del libro Public Policy Analytics: Code & Context for Data Science in Government, por Ken Steif.

Si aún no has instalado R, está aquí. Acto seguido, baja RStudio. Puedes también ir a la nube en Posit Cloud.

Visualización de datos: porqués

Ver números y cadenas de caracteres que forman un conjunto de datos puede ser interesante, o no, pero normalmente no tiene tanta utilidad. Por ejemplo:

```
library(wooldridge)  
  
data(wage1)  
head(wage1)
```

```

##   wage educ exper tenure nonwhite female married numdep smsa northcen south
## 1 3.10   11     2     0      0     1     0     2     1     0     0
## 2 3.24   12    22     2      0     1     1     3     1     0     0
## 3 3.00   11     2     0      0     0     0     2     0     0     0
## 4 6.00    8    44    28      0     0     1     0     1     0     0
## 5 5.30   12     7     2      0     0     1     1     0     0     0
## 6 8.75   16     9     8      0     0     1     0     1     0     0
##   west construc ndurman trcommpr trade services profserv profocc clerocc
## 1     1         0     0      0     0     0     0     0     0     0
## 2     1         0     0      0     0     1     0     0     0     0
## 3     1         0     0      0     1     0     0     0     0     0
## 4     1         0     0      0     0     0     0     0     0     1
## 5     1         0     0      0     0     0     0     0     0     0
## 6     1         0     0      0     0     0     0     1     1     0
##   servocc    lwage expersq tenursq
## 1     0 1.131402      4     0
## 2     1 1.175573    484     4
## 3     0 1.098612      4     0
## 4     0 1.791759   1936    784
## 5     0 1.667707      49     4
## 6     0 2.169054     81    64

```

¿Qué aprendemos de ver estos datos así? ¿Podemos rápidamente determinar si años de educación se traducen a mayores ingresos? ¿Podemos determinar si afecta en algo la relación marital? Para muchos humanos, es difícil extraer información con meramente mirar a números sin contexto adicional. Pero podríamos ver algo en este gráfico

Lo mismo podríamos hacer con los datos que trabajamos la semana pasada de homicidios con armas de fuego en EEUU:

```

library(dslabs)
library(tidyverse)

```

```

## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## vforcats   1.0.1     v stringr   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyv     1.3.1
## v purrr    1.1.0
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()   masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

```

```

data("murders")
tail(murders)

```

```

##           state abb       region population total
## 46      Vermont VT Northeast  625741     2
## 47    Virginia VA      South  8001024   250
## 48 Washington WA      West  6724540   93
## 49 West Virginia WV      South 1852994   27
## 50 Wisconsin WI North Central 5686986   97
## 51    Wyoming WY      West  563626     5

```

Relación entre educación y salario

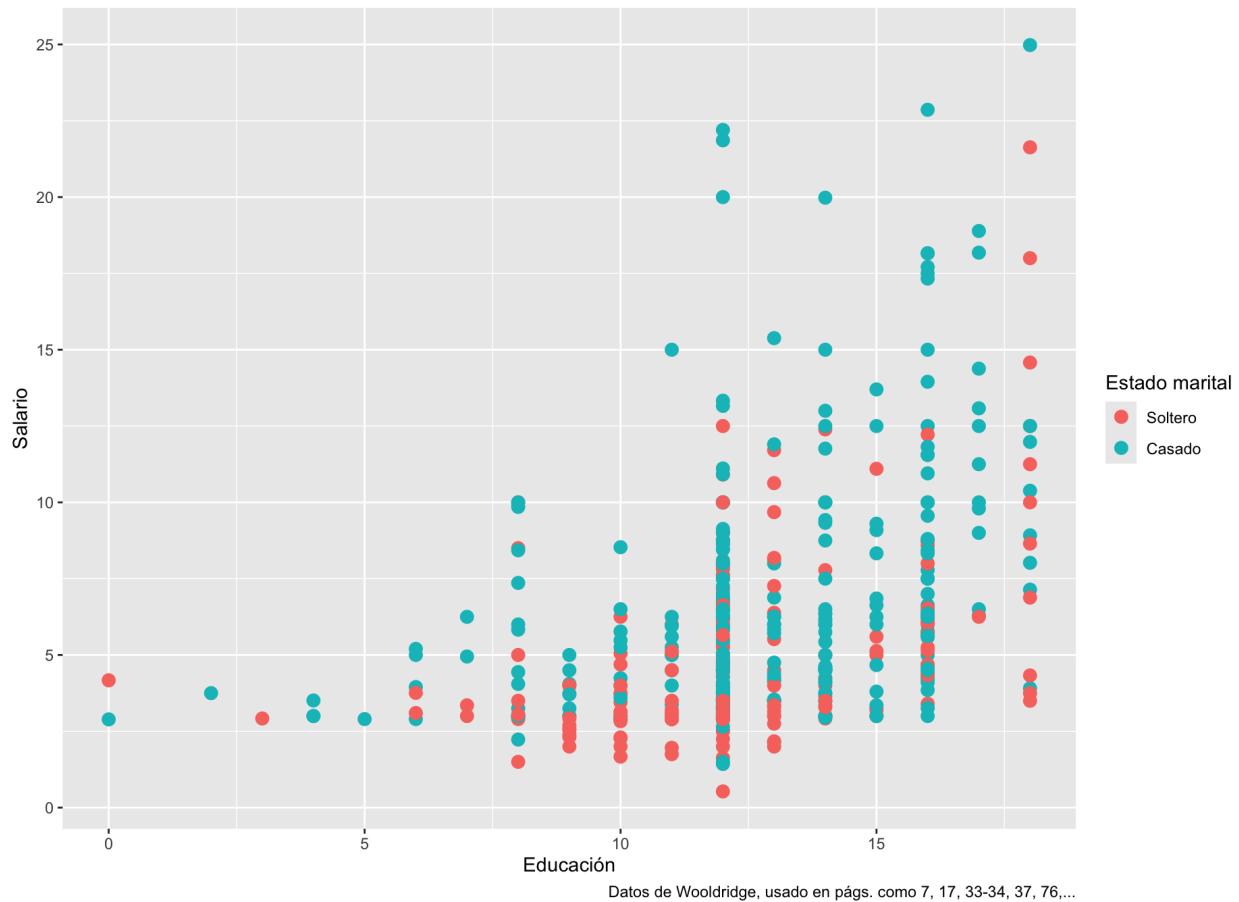


Figure 1: Meta 1.^a: datos de ingreso, que continuaremos al abundar en modelos inferenciales el miércoles que viene.

No podemos determinar con facilidad a qué estado le toca la población más grande o pequeña, y si existe alguna relación entre el tamaño de población y el total de asesinatos, o de cómo varían las tasas de asesinatos por regiones de estados en la federación estadounidense. Sin embargo, eso puede responderse sin muchas palabras con el próximo gráfico.

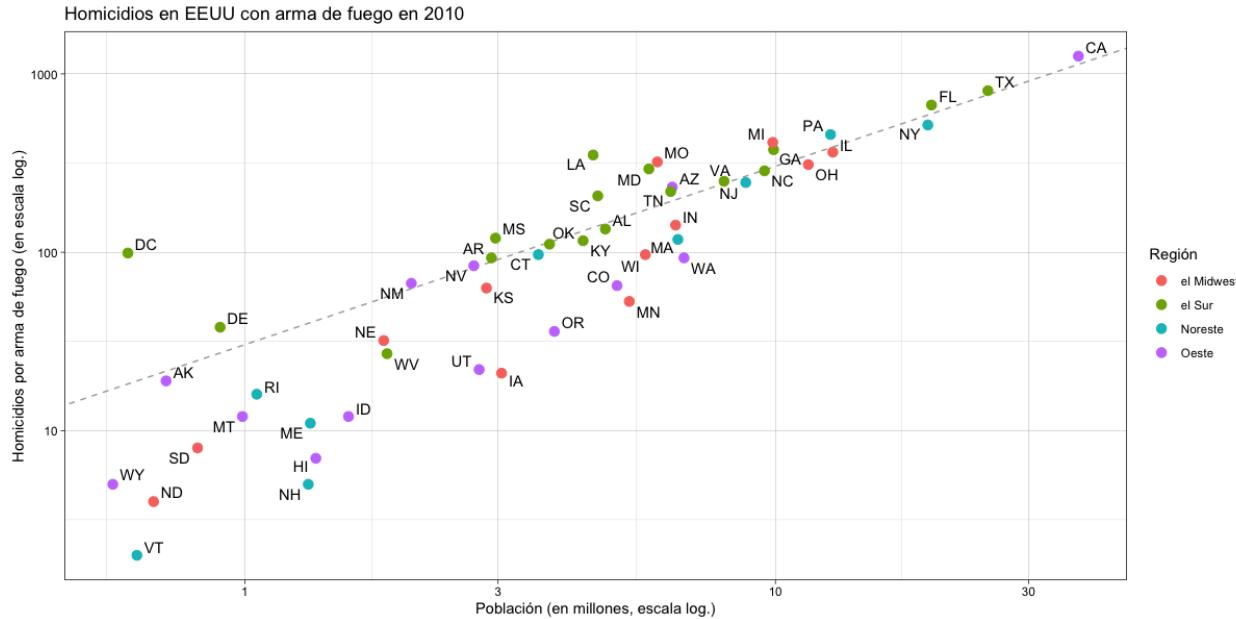


Figure 2: Gráfico de la meta 2.^a

Una imagen vale más que mil palabras dice el dicho. Sin adentrarnos a la inferencia estadística (que cubriremos en la tercera semana de esta secuencia de talleres), hemos podido comunicar relaciones y hallazgos en datos. A veces puede ser este ejercicio uno tan convincente que no requiera análisis subsiguientes.

Vivimos en una era de creciente disponibilidad de conjuntos de datos informativos y de herramientas de software, con lo cual el uso de visualizaciones ha aumentado en diversos espacios: académicos, gubernamentales, organizaciones sociales, prensa, e industrias varias.

En R, una de las principales maneras con la que trabajaremos estos análisis visuales es de la mano de `ggplot2`, así como con otros paquetes que ayudan a procesar esta información. Si bien existen otros métodos de graficar en R y otros programas, la preferencia de este taller es utilizar el sistema de procesamiento que ofrece `tidyverse`, con `ggplot2` incluido.

Esta es una secuencia de cuatro semanas, y en esta lección continuamos con lo aprendido la semana anterior, donde terminamos con visualizaciones sencillas y con el uso de `tidyverse` para manejar datos. Tenemos la meta hoy de que al culminar las segundas dos horas de esta secuencia podamos:

1. Entender cómo usar la gramática de gráficas
2. Entender cómo usar `ggplot2`, continuando con lo aprendido de `tidyverse` de la semana pasada.
3. Entender cómo utilizar en varias maneras `tidycensus` para generar mapas.

En la próxima sesión, del 19 de febrero, entraremos más en *wrangling* de datos, así como en la inferencia estadística, con introducciones en R sobre modelos lineales, jerárquicos y longitudinales, así como sus diagnósticos posteriores. En la última sesión, pautada para el 26 de este mes, evaluaremos análisis de redes, y de ser posible, otros métodos de análisis textual.

ggplot2

R ofrece varias opciones para graficar, siendo útiles las capacidades incluidas en su instalación básica. Además, existen paquetes como `grid`, que permite un control preciso de los elementos gráficos, y `lattice`, que facilita la creación de gráficos multivariados y en paneles o facetas. Sin embargo, en este taller (y en los libros de referencia usados y descritos arriba) se ha optado por usar `ggplot2`, ya que permite a los principiantes crear gráficos complejos y estéticos mediante una sintaxis intuitiva y fácil de recordar, dividiendo los gráficos en componentes básicos.

`ggplot2` destaca por su uso de una *gramática de gráficos* – de donde vienen las primeras dos g – que simplifica el proceso de creación de gráficos. Al aprender unos pocos componentes esenciales de esta gramática, los usuarios pueden generar una amplia variedad de gráficos con facilidad. Además, su comportamiento por defecto está diseñado para producir resultados agradables y funcionales con código conciso y legible, lo que facilita su uso por principiantes. Como factor limitante está el que está diseñado para trabajar con tablas de formato `tidy` (con filas con observaciones y columnas conteniendo variables), pero un número sustancial de conjuntos de datos se trabajan en ese formato, o pueden convertirse a ese formato.

Componentes de un gráfico

Hoy construiremos varios tipos de gráficos como los que vimos arriba, así como mapas informativos. Antes que todo eso, vale señalar que los gráficos se dividen en tres componentes principales. Usaré otro gráfico que creara en 2020 durante la pandemia del Coronavirus para ejemplificar estos componentes.

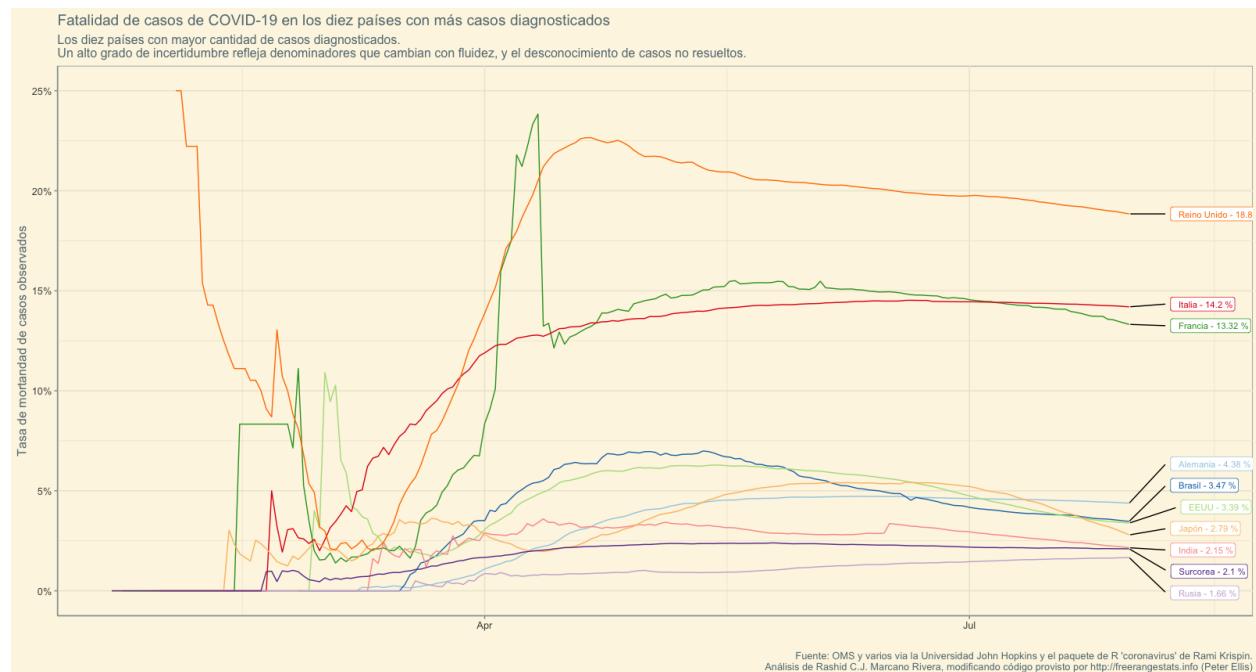


Figure 3: Datos para ejemplificar

- **Datos:**

- Estoy pasando al gráfico un conjunto de datos que corté sobre fatalidad de casos de COVID-19 hasta el 31 de julio de 2020 (los datos continúan hasta 2023).
- Este es el componente de *datos* del gráfico.

- **Geometrías:**

- El gráfico es uno de líneas, útil para varias series de tiempo. Este componente es una *geometría*. Otras geometrías posibles son gráficos de dispersión, histogramas, diagramas de barras, densidades suavizadas, y diagrama de cajas, entre otros.

- **Mapeo estético:**

- El gráfico usa señales visuales para representar en el lienzo vacío con capas distintos tipos de información provista en el conjunto de datos:
 - * Posiciones en el eje de x (tiempo)
 - * Posiciones en el eje de y (tasas de mortandad observadas)
 - * Color (asignado por país)
- Cada línea representa la información de un país para una serie de fechas. Estos se aclaran con una etiqueta para aclararnos esa relación de línea-color-país.
- El *mapeo estético* depende de la geometría utilizada.

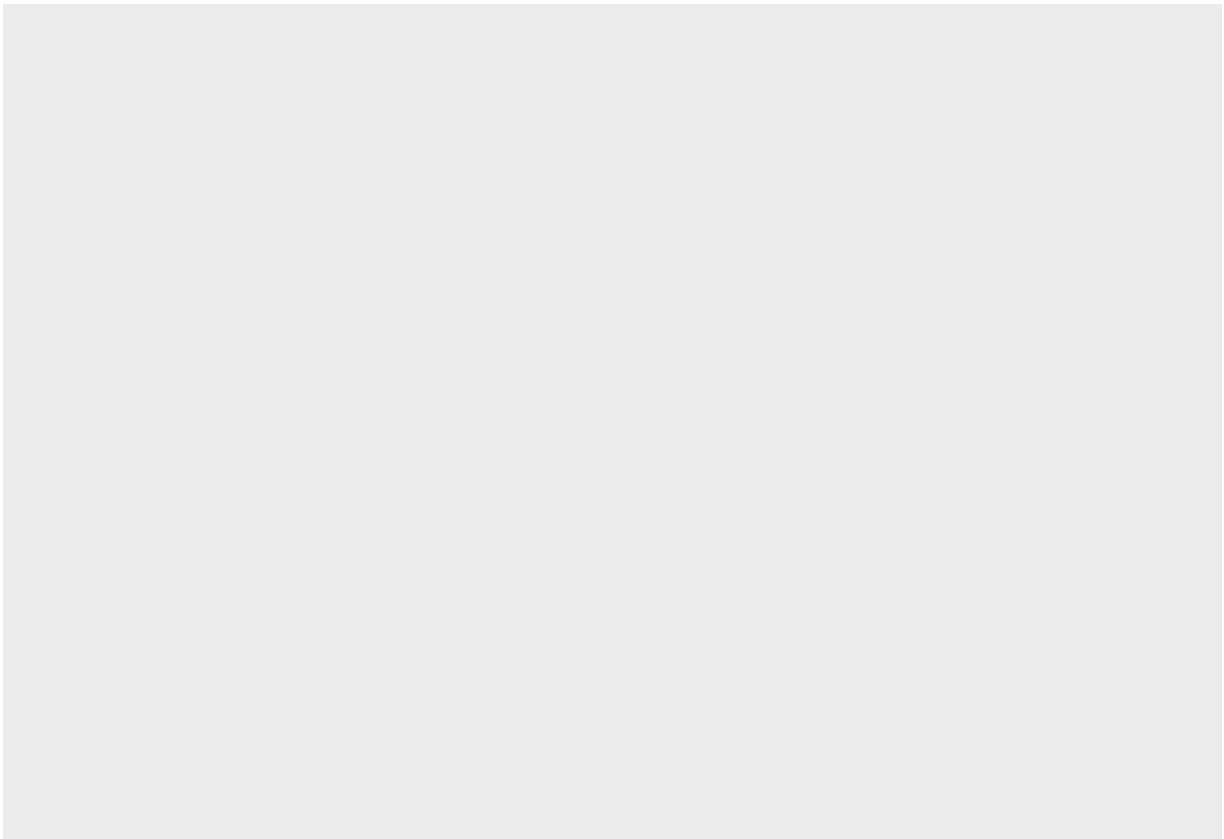
- **Observaciones adicionales:**

- Ejes *x* e *y* definidos por el rango de los datos y ambos en escalas logarítmicas.
- El gráfico incluye etiquetas, un título, etiquetas de variables, nota al calce, y el tema utilizado es uno solarizado que pareciera no muy distante al utilizado por el periódico especializado “The Financial Times”.
- Volveremos luego a estos datos para construir esta imagen si nos da tiempo en el taller, y si no, tendrán disponible el cómo hacerlo para referencia.

Lienzo vacío y capas

Manteniéndonos cerca de los datos utilizados en la semana pasada, construiremos por *capas* la información que va en el gráfico.

```
murders |> ggplot()
```



El primer paso de crear un gráfico de ggplot es asignar los datos a un lienzo vacío. Esto no significa poblar el lienzo con esos datos, sino pasarle al programa la información inicial, como un pintor que selecciona el tema que usará para la pintura que visualiza en su mente. Esto lo hicimos al pasar el *pipe* (`|>` o `|>`) los datos a ggplot, y lo que ocurrió fue que al no darle más información (capas, como la pintura en un lienzo), nos quedó un recuadro gris enmarcado por un borde blanco.

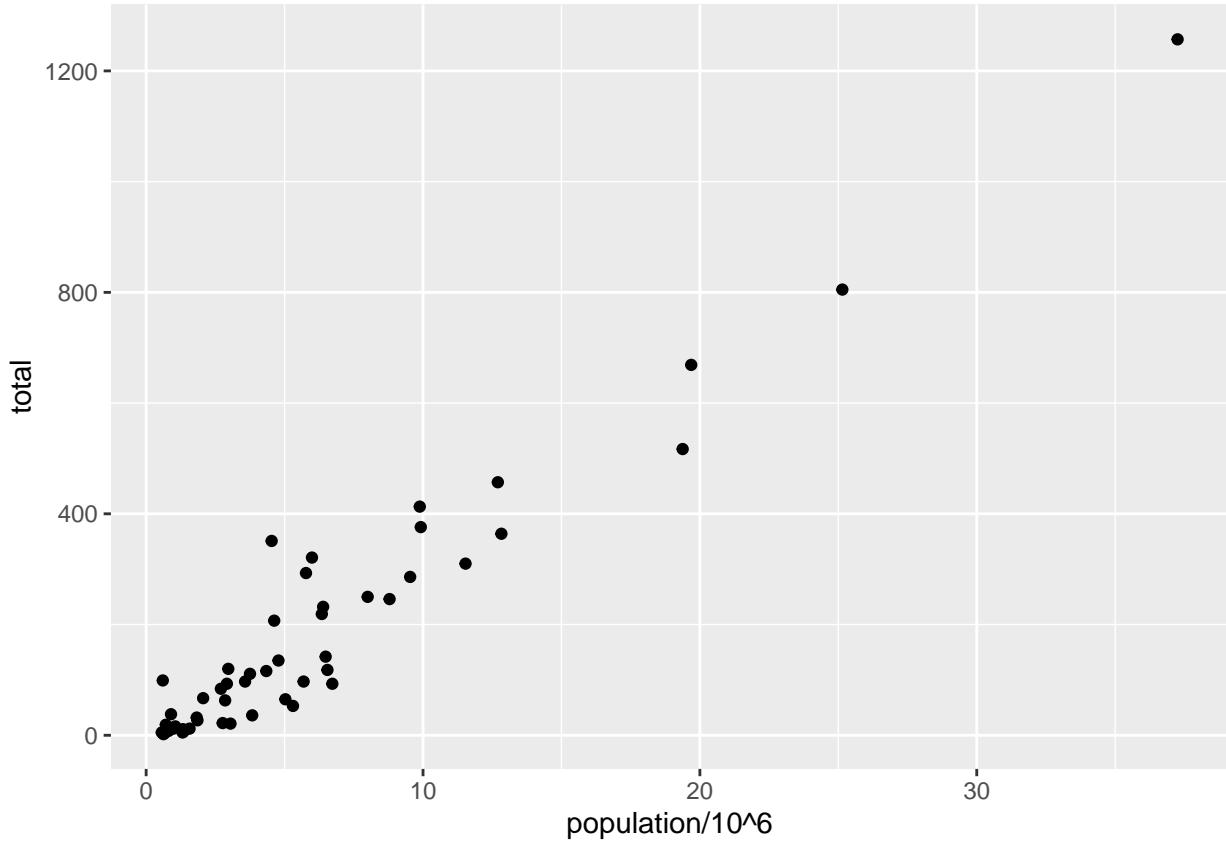
En ggplot, la información entonces se suministra al lienzo por capas, y se le pueden añadir adicionales. Esto tomará la forma de código siguiente:

```
DATOS |> ggplot() + CAPA 1 +
CAPA 2 + ... + CAPA N
```

Usualmente, la primera capa que añadimos define la geometría. Si queremos hacer un diagrama de dispersión, ¿qué geometría deberíamos utilizar?

Si vemos la hoja de referencia (en la carpeta del taller 2, o accesible en esta página: <https://github.com/rstudio/cheatsheets/blob/main/data-visualization.pdf>), vemos que la función utilizada para crear gráficos con esta geometría puntillista es `geom_point`.

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

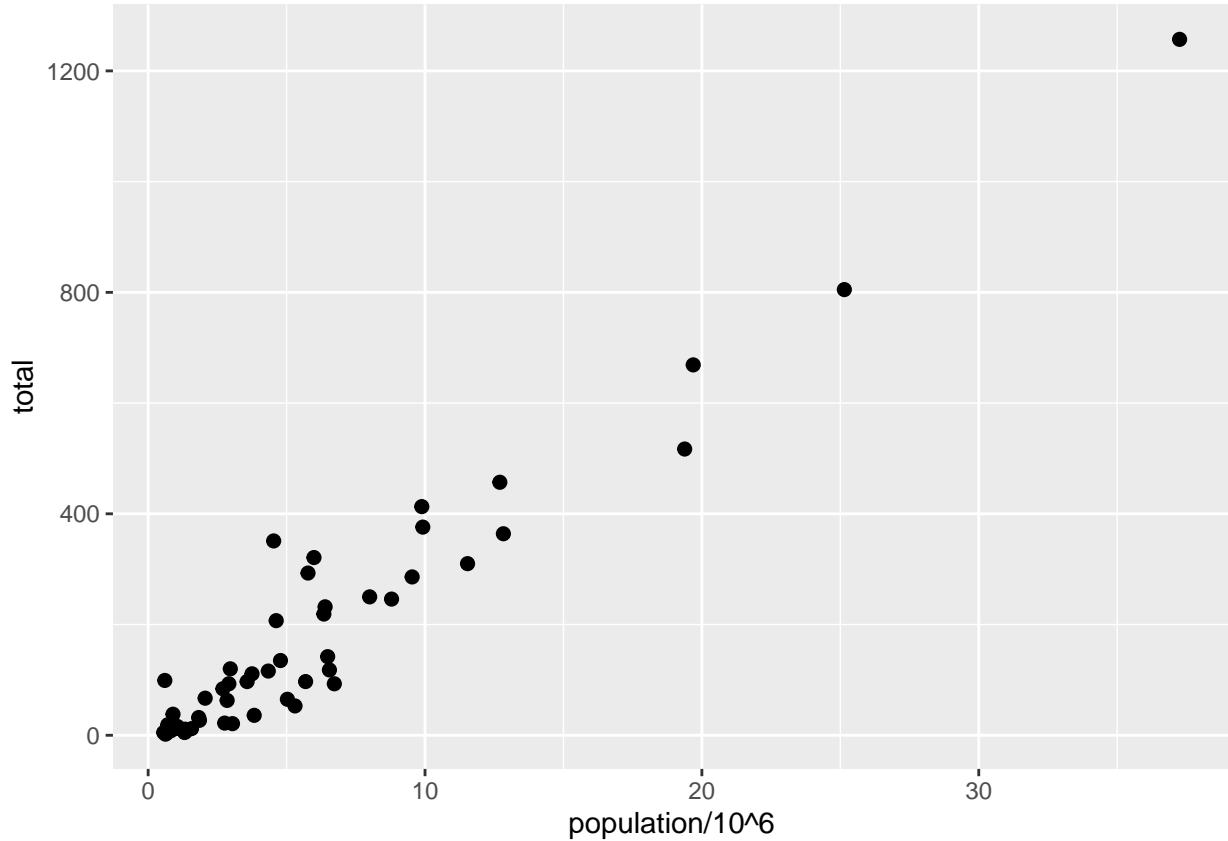


En este caso ya hemos dado un paso adicional y añadido una primera capa en esta obra: le indicamos que queremos una capa que tenga una geometría de puntos, y un mapeo estético que toma las coordenadas en un plano cartesiano donde el eje de x quedó definido como `population/10^6`, la población de los estados o Washington D.C., en millones; el eje de y quedó definido entonces como el total de asesinatos con armas de fuego. En el mapeo estético entonces los puntos quedan asignados a esas coordenadas. De esta manera, las distancias entre puntos, así como otras características que queramos añadir, quedan expresadas. Esto se da a través de la función `aes`. Esta será de las funciones que más usen al graficar.

Noten que hasta ahora hemos trabajado este lienzo sin guardarlo como objeto. Si bien esto puede funcionar bien, es posible que queramos guardar nuestro progreso y seguir añadiendo capas adicionales. En este caso, al ejecutar el comando y guardarlo en objeto, el programa no nos dará automáticamente una actualización del gráfico; tendremos que llamar al objeto para que aparezca:

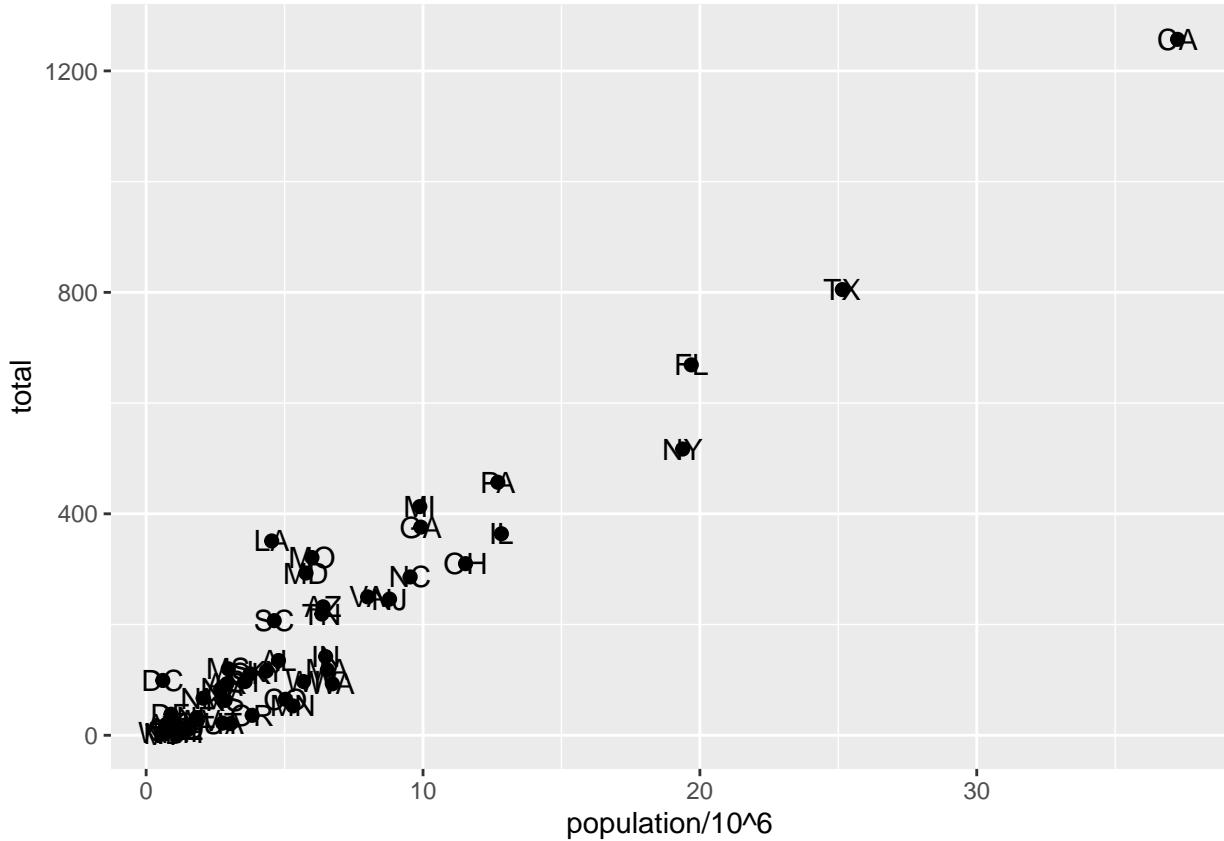
```
p<-murders |>
  ggplot() +
  geom_point(aes(x = population/10^6, y = total), size=2)

p
```



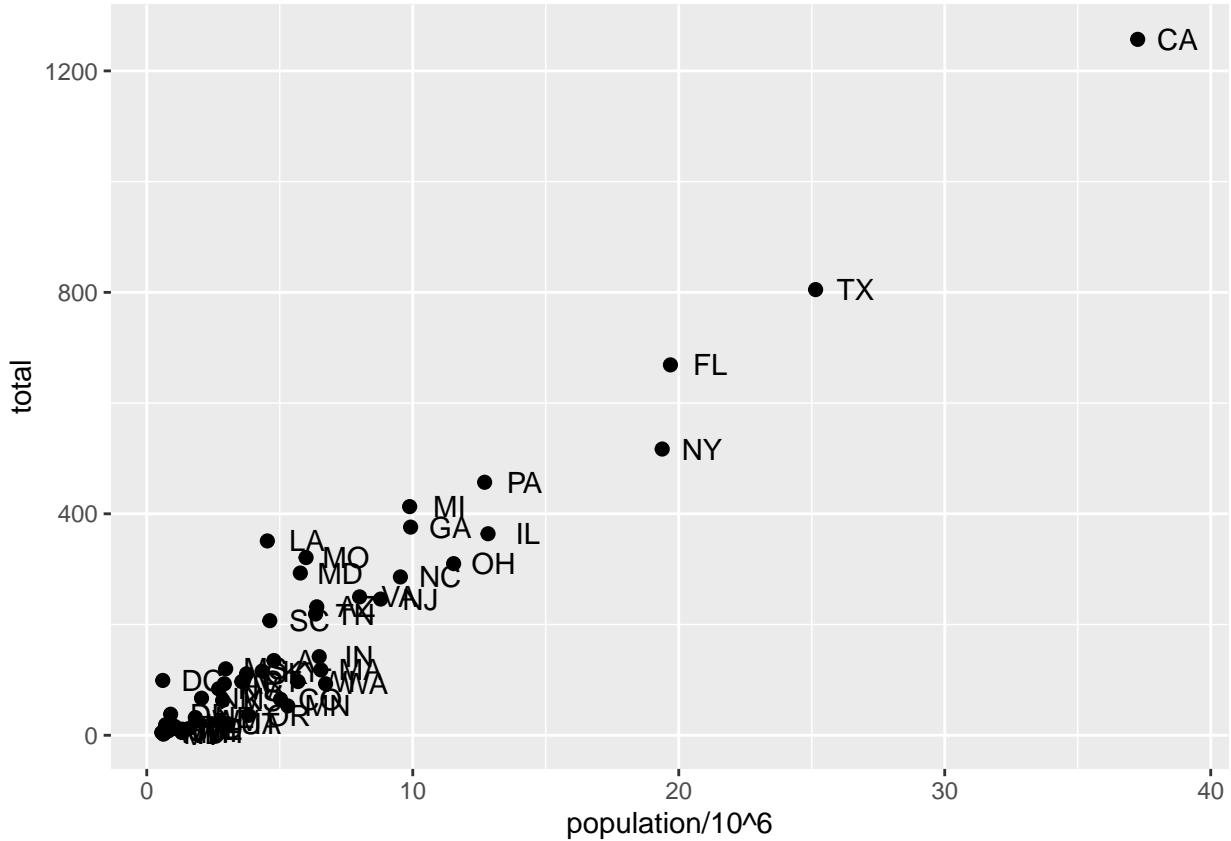
Tenemos entonces nuestro gráfico de dispersión inicial. Quizás no es tan informativo pero vemos una serie de puntos con el mapeo estético inicial. Noten que podríamos quitar la `x=` y la `y=` y no pasaría nada, ya que en ausencia de esta especificación, el programa entiende por defecto que lo primero que se le asigna es la información del eje horizontal, y en segundo orden el vertical:

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



Aquí he añadido una geometría nueva: texto. Hay dos geometrías para esto, `geom_label` y `geom_text`, uno con el texto enmarcado en un recuadro y el segundo sin ello. Ya que cada punto (cada jurisdicción que es realmente parte de los Estados Unidos de América en este caso) tiene una etiqueta, necesitamos un mapeo estético para hacer la conexión entre los puntos y las etiquetas, así que se le asignó el mismo tal que el texto cayera exactamente en la misma coordenada que el punto. Pero esto se puede corregir:

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



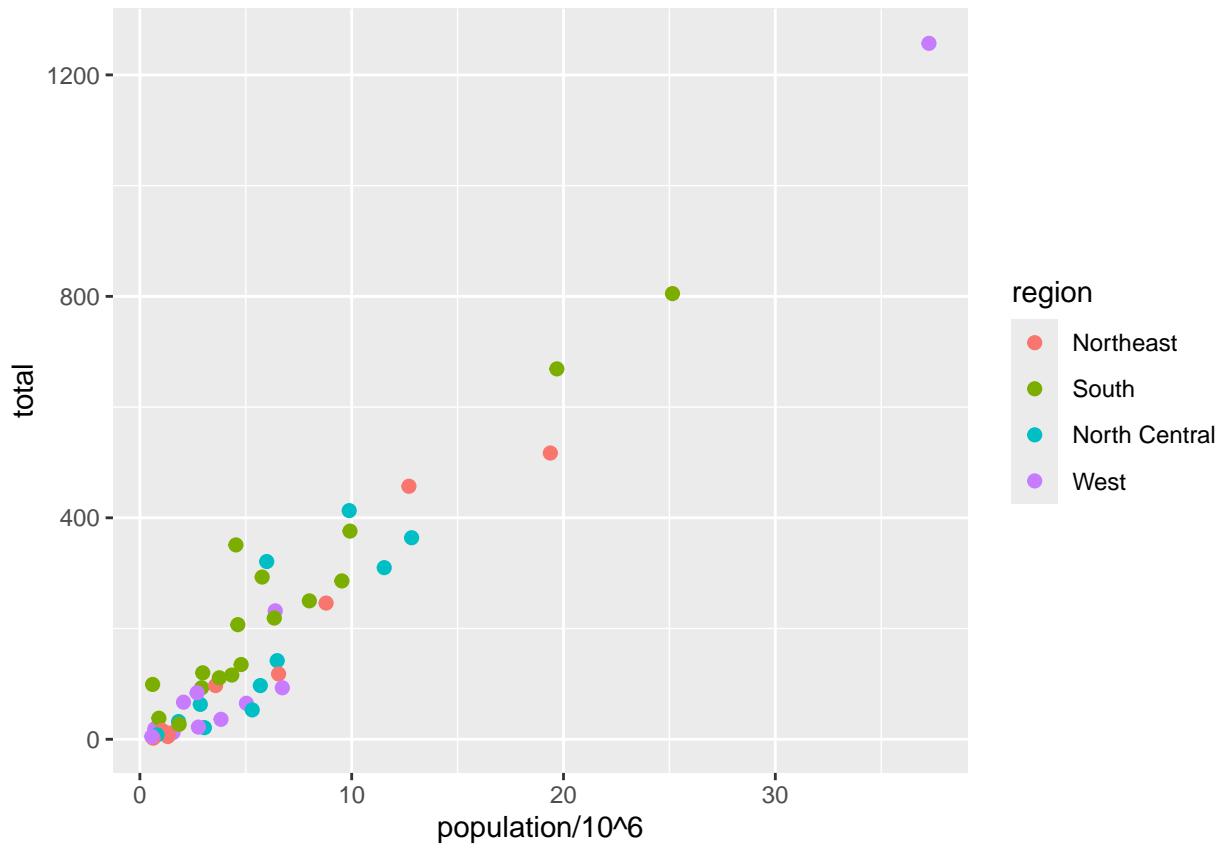
```
#nudge_y desplazaría el texto en el eje de y
```

En este caso hemos empujado a través del eje de equis la etiqueta de texto con un valor numérico de 1.5. Valores mayores aumentarían la distancia del texto y el punto, mientras que menores harían lo contrario.

Ahora podremos seguir con una complicación en el mapeo estético: queremos añadir una capa de color al lienzo que represente regiones de estas jurisdicciones. Esto se hace al añadir la opción de `colour` y dándole una variable, en este caso `region`.

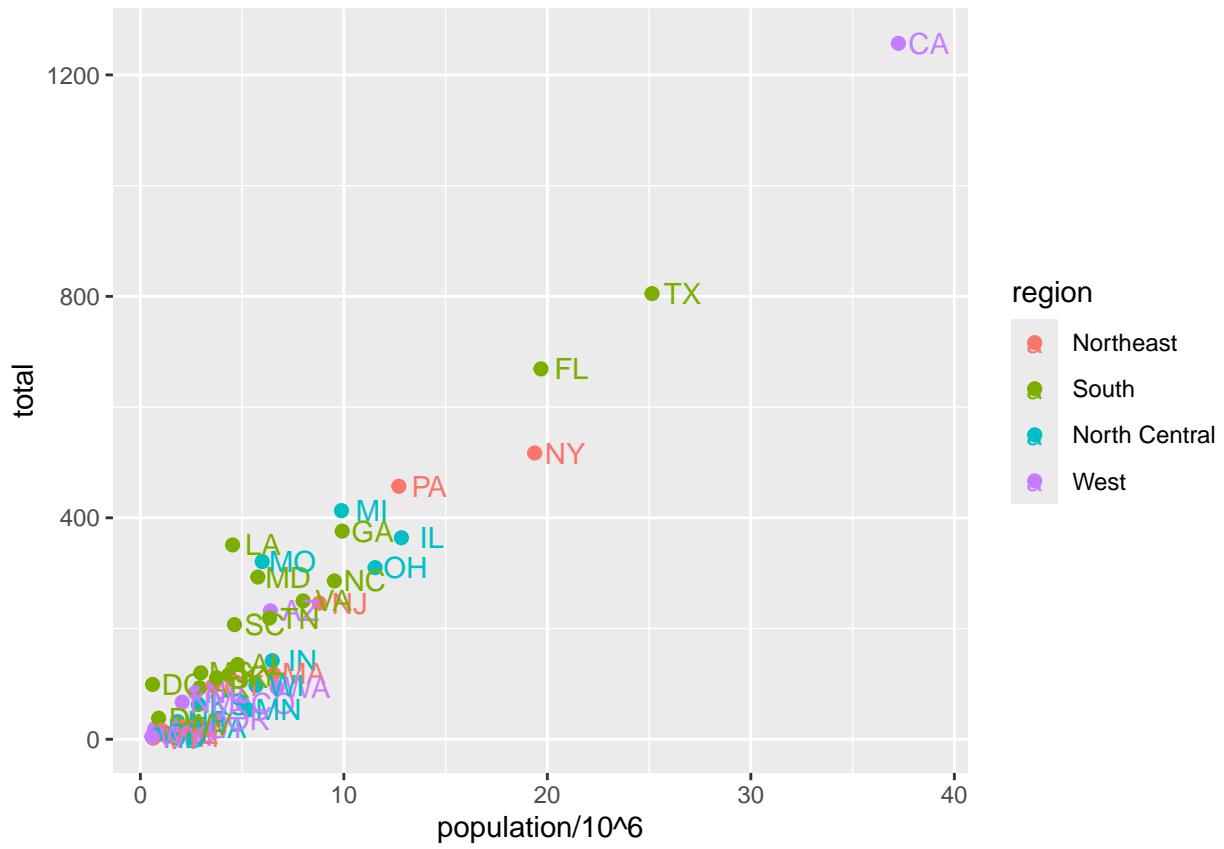
```
# Crear el gráfico base con puntos coloreados por región
p <- murders |>
  ggplot() +
  geom_point(aes(x = population/10^6, y = total, colour = region), size = 2)

# Mostrar el gráfico
p
```



Ahora paso nuevamente a añadir etiquetas abreviadas, con el color por región.

```
# Añadir etiquetas desplazadas en el eje x (con color por región)
p<-p +
  geom_text(aes(x = population/10^6, y = total, label = abb, colour = region), nudge_x = 1.5)
```

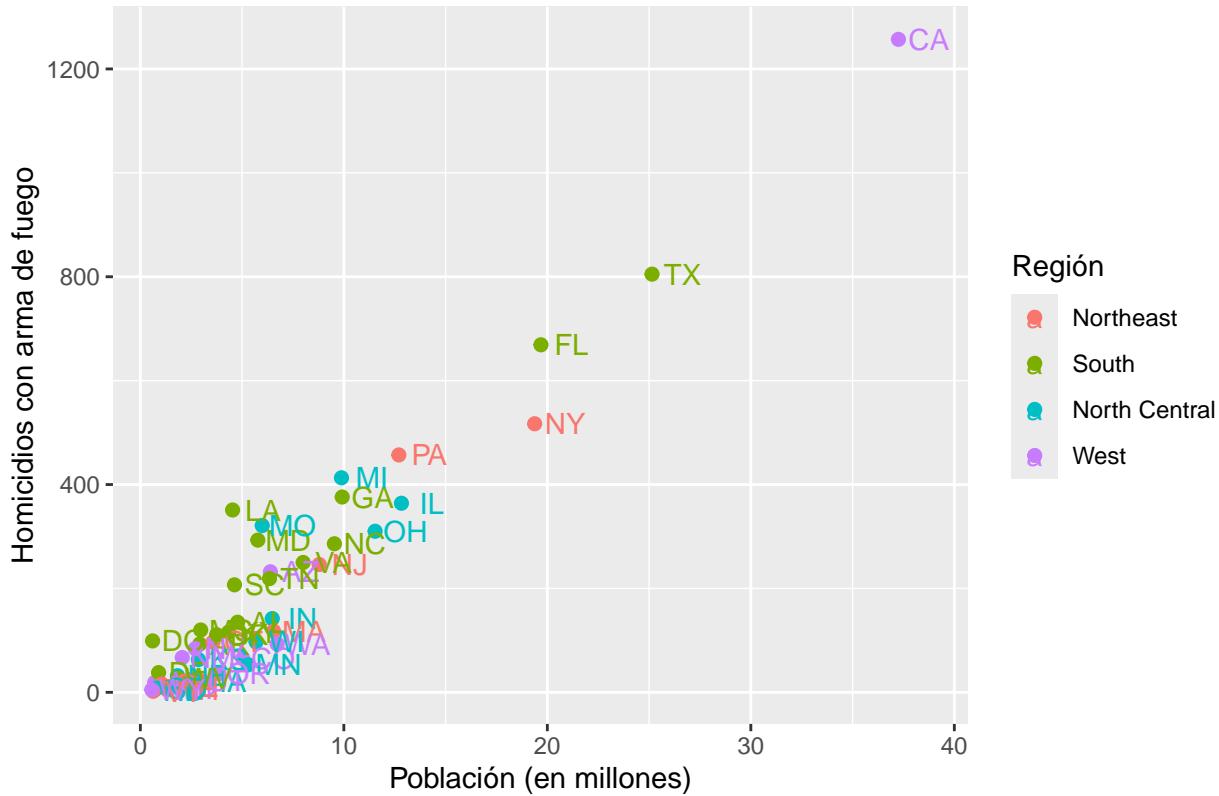


Y para darle más información al gráfico le añadimos etiquetas y títulos que sean más informativas que los nombres de variables.

```
#demosle más información a la gráfica de dispersión.
```

```
p+labs(
  x = "Población (en millones)",    # Cambia el nombre del eje x
  y = "Homicidios con arma de fuego",      # Cambia el nombre del eje y
  colour = "Región",                  # Cambia el nombre de la leyenda de color
  title = "Homicidios con arma de fuego en EEUU, 2010"  # Titulo del gráfico
)
```

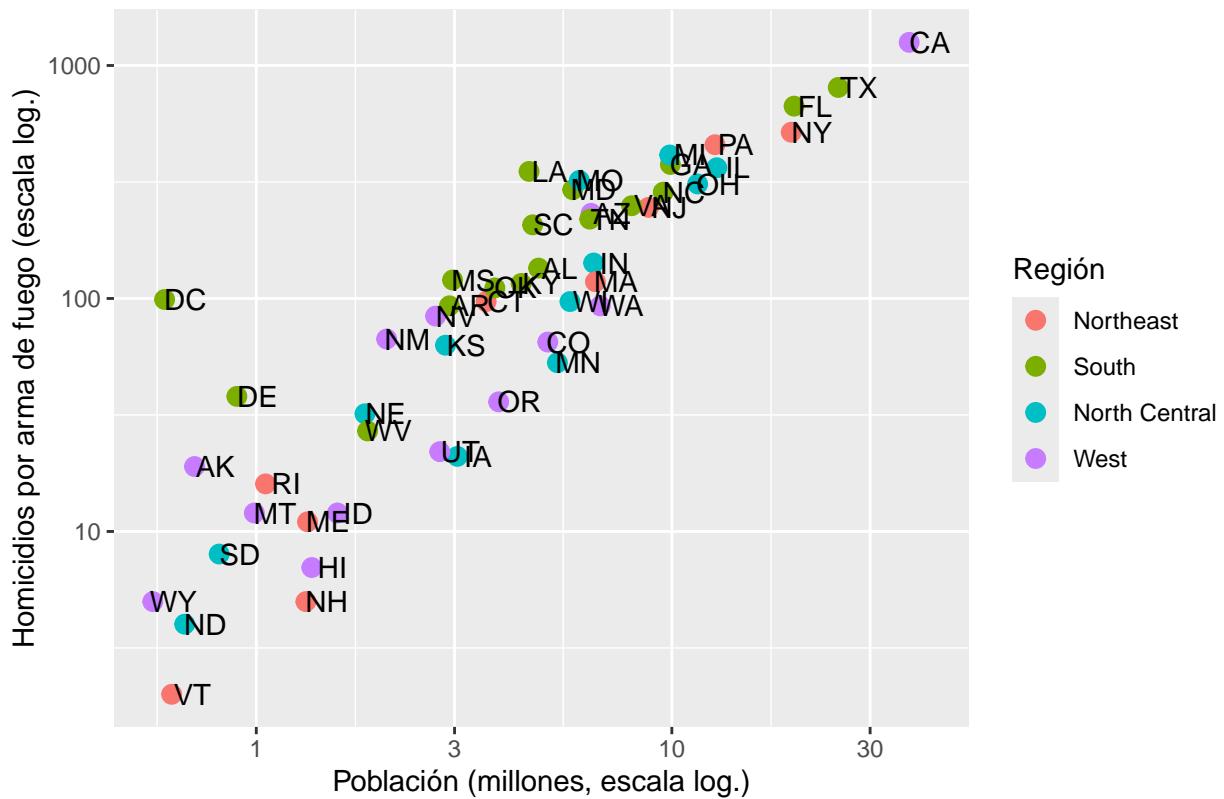
Homicidios con arma de fuego en EEUU, 2010



Hasta ahora hemos ido añadiendo capas pero notamos que tenemos una gran concentración de puntos en la parte inferior izquierda del lienzo: la mayoría de las jurisdicciones tienen menos de 10 millones de habitantes y menos de 400 homicidios. Esto hace leer e interpretar lo que sucede en para estos casos difícil. Podríamos entonces representar el gráfico con una transformación logarítmica al re-escalar con `scale_x_continuous` y `scale_y_continuous`:

```
p2<-murders |>
  ggplot() +
  geom_point(aes(x = population/10^6, y = total, colour = region), size = 3)
p2<- p2 + geom_text(aes(x = population/10^6, y = total, label = abb), nudge_x = 0.05)
p2<-p2+scale_x_continuous(trans = "log10")+
  scale_y_continuous(transform = "log10")
p2<-p2+
  labs(
    x = "Población (millones, escala log.)", # Cambia el nombre del eje x
    y = "Homicidios por arma de fuego (escala log.)", # Cambia el nombre del eje y
    colour = "Región", # Cambia el nombre de la leyenda de color
    title = "Homicidios por arma de fuego vs población, por región" # Título del gráfico
  )
p2
```

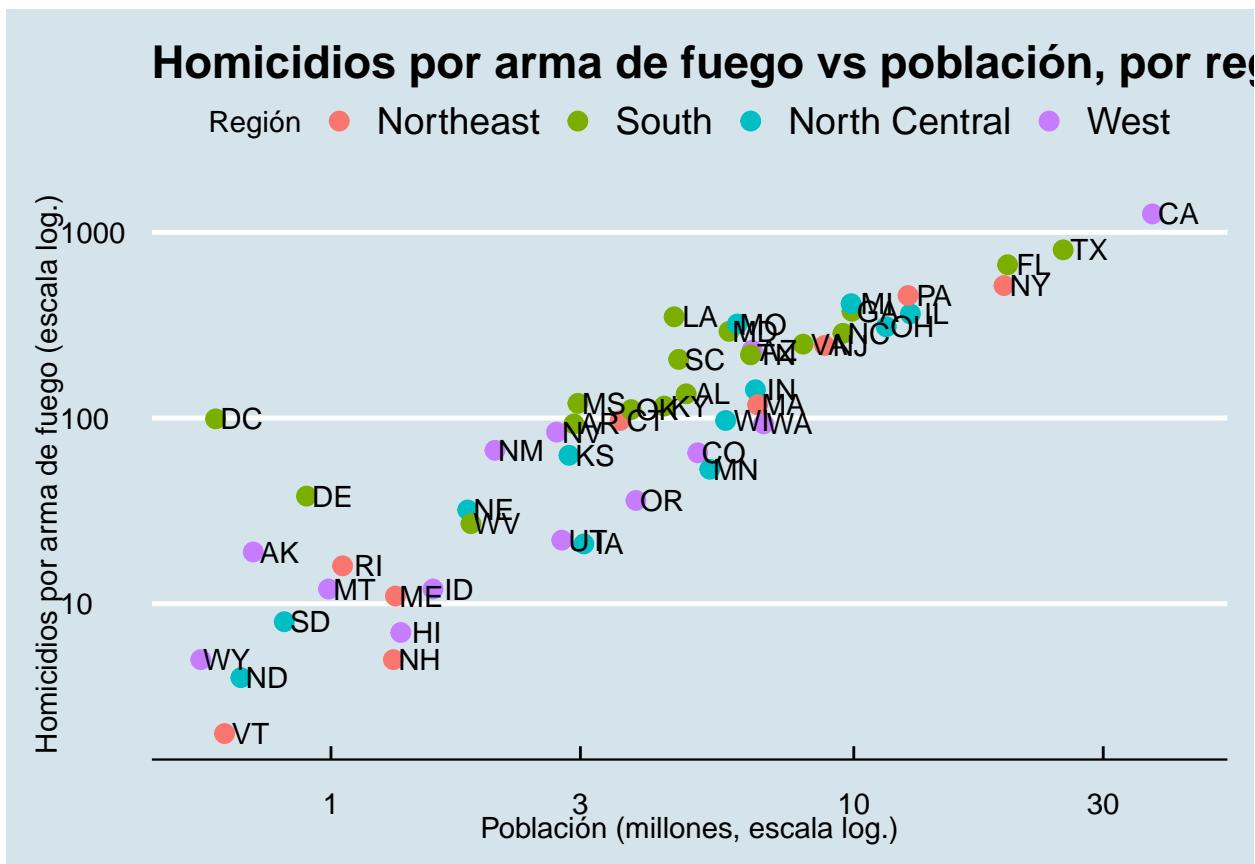
Homicidios por arma de fuego vs población, por región



Podríamos añadir una capa temática, usando el paquete `ggthemes`, que tiene un catálogo estético variado, que recomiendo verifiquen a través de <https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/>. Esto es en adición a los que ya ofrece el paquete de `ggplot2` <https://ggplot2.tidyverse.org/reference/ggtheme.html>.

En este caso le añadire una visualización al estilo del semanario británico *The Economist*.

```
library(ggthemes)
p2 + theme_economist()
```



El gráfico que buscábamos

Normalmente queremos añadir formas o anotaciones a las figuras que no se derivan directamente del mapeo estético; algunos ejemplos incluyen etiquetas, cuadros, áreas sombreadas y líneas. Si queremos añadir una línea que represente la tasa promedio de asesinatos en todos los Estados Unidos, tendremos que determinarlo aparte con la ayuda de `dplyr` (parte de `tidyverse`), y tendremos que hacer la transformación adecuada también (logarítmica):

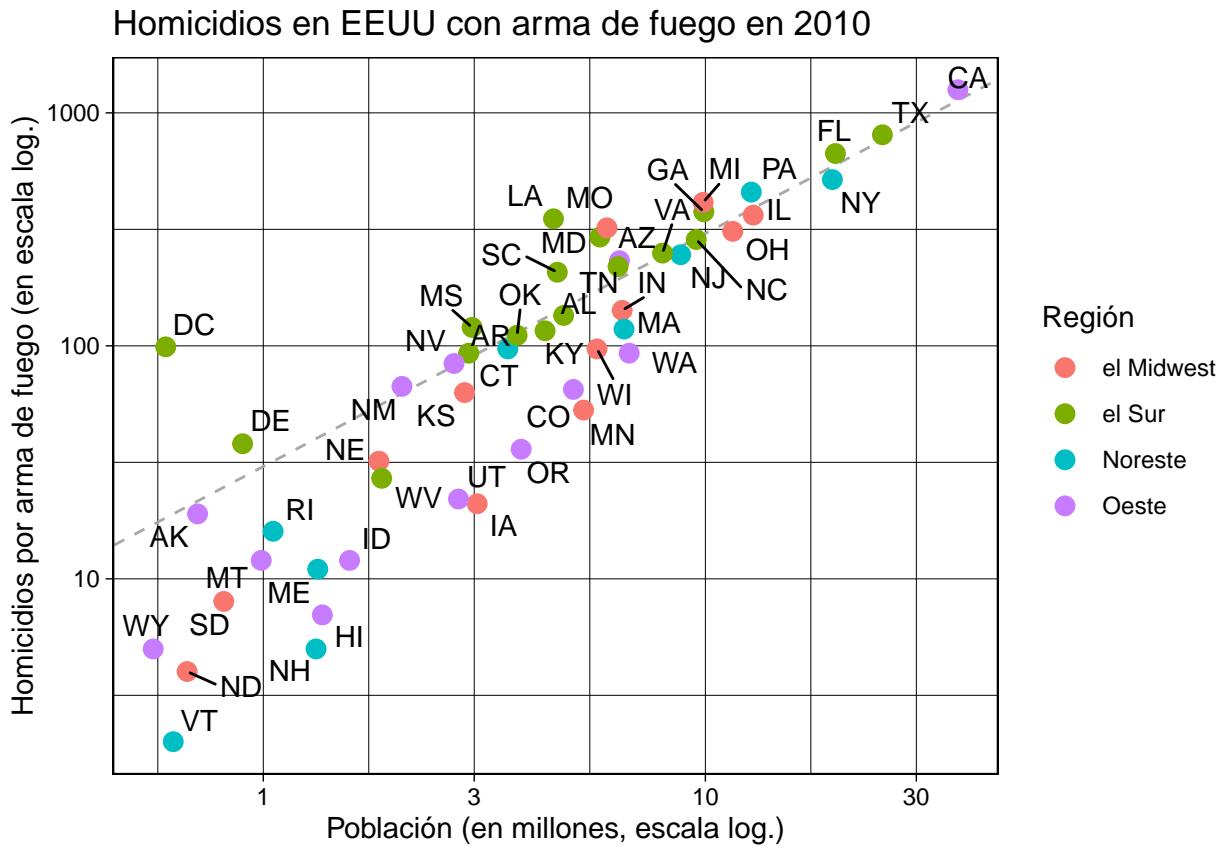
```
library(ggthemes)
library(ggrepel)
library(dslabs)
t <- murders |>
  summarise(tasa = sum(total) / sum(population) * 10^6) |>
  pull(tasa)

murders |>
  mutate(region=case_when(
    region == "Northeast" ~ "Noreste",
    region == "North Central" ~ "el Midwest",
    region == "West" ~ "Oeste",
    region == "South" ~ "el Sur")) |>
  ggplot(aes(population/10^6, total)) +
  geom_abline(intercept = log10(t), lty = 2, color = "darkgrey") +
  geom_point(aes(col = region), size = 3) +
  geom_text_repel(aes(label = abb)) +
```

```

scale_x_log10() +
scale_y_log10() +
labs(title = "Homicidios en EEUU con arma de fuego en 2010",
x = "Población (en millones, escala log.)",
y = "Homicidios por arma de fuego (en escala log.)",
color = "Región") +
theme_linedraw()

```



He aquí los pasos entonces que necesitábamos para recrear la imagen arriba.'

Gráficos rápidos: `qplot()`

Si bien esto ha sido útil y han podido ver cómo generar gráficas complejas siguiendo la gramática de gráficos, es probable que en algún momento quieran ver rápidamente unas relaciones visuales de manera instantánea, con la intención luego de volver y darle mejor forma y seguir las complicaciones de un gráfico de `ggplot2` con todas sus capas. La opción que da `ggplot2` a esto es el uso de `qplot()`.

Por ejemplo, aquí modifiqué los grupos de jurisdicciones de Estados Unidos a unos grupos específicos, y creo un gráfico para ver densidades con relleno (`fill=grupo`) y con líneas que también sirven para demarcar estos grupos de manera distinta (e.g. línea sólida, línea entrecortada). Noten que ha corrido todo el gráfico de la última línea.

```

murderssg<-murders |>
  mutate(tasa= total/population *10^5,
 ,grupo = case_when(

```

```

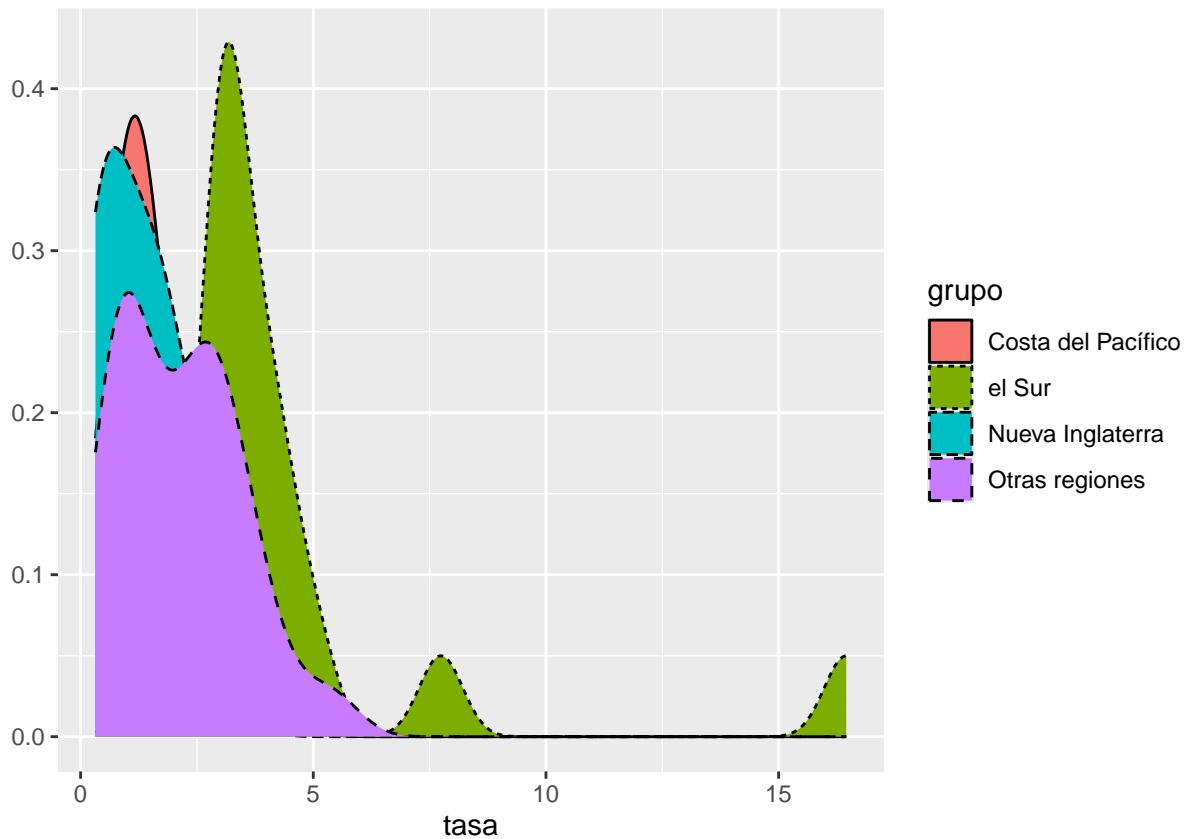
abb %in% c("ME", "NH", "VT", "MA", "RI", "CT") ~ "Nueva Inglaterra",
abb %in% c("WA", "OR", "CA") ~ "Costa del Pacífico",
region == "South" ~ "el Sur",
TRUE ~ "Otras regiones"))
qplot(tasa, data = murderssg, geom= "density", fill = grupo, linetype=grupo)

```

```

## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

```

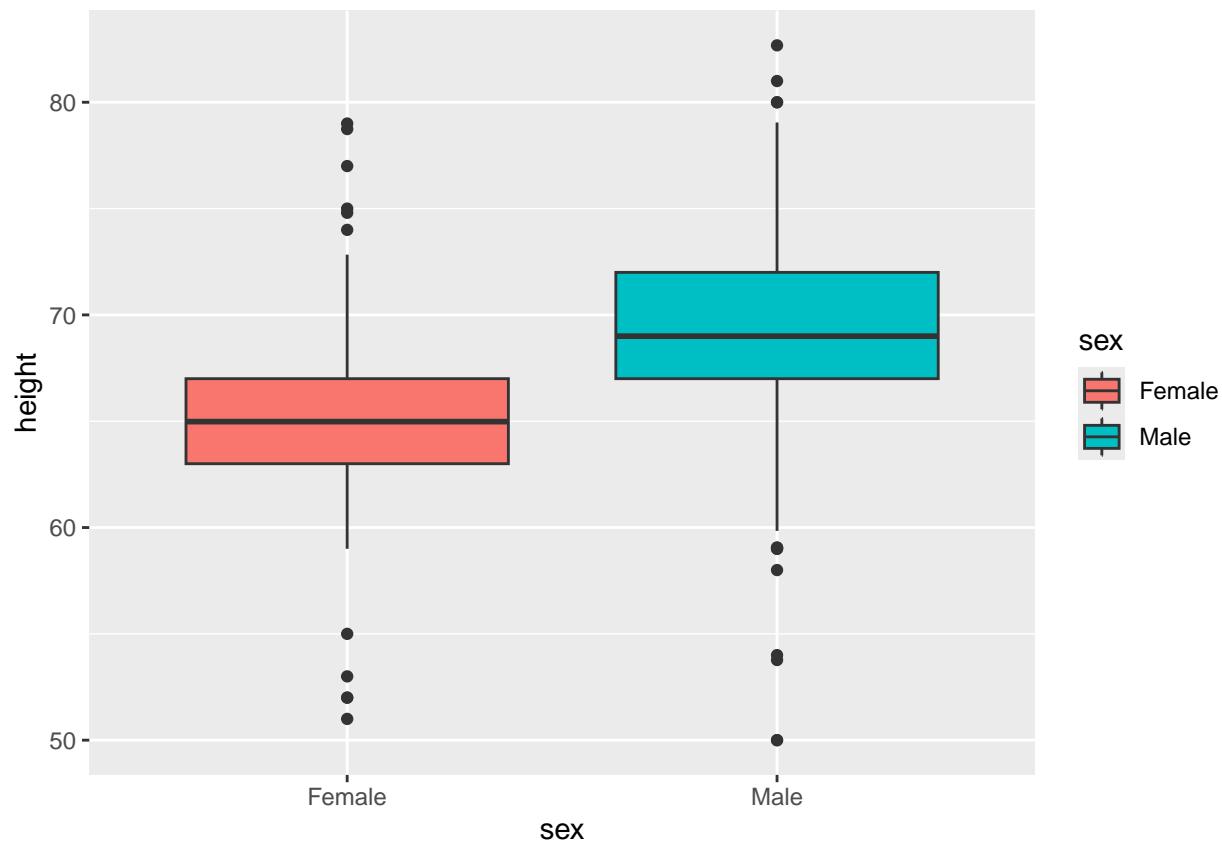


En las siguientes líneas pido y ejecuto variaciones de gráficos con qplot con los datos de altura del libro de Irizarry:

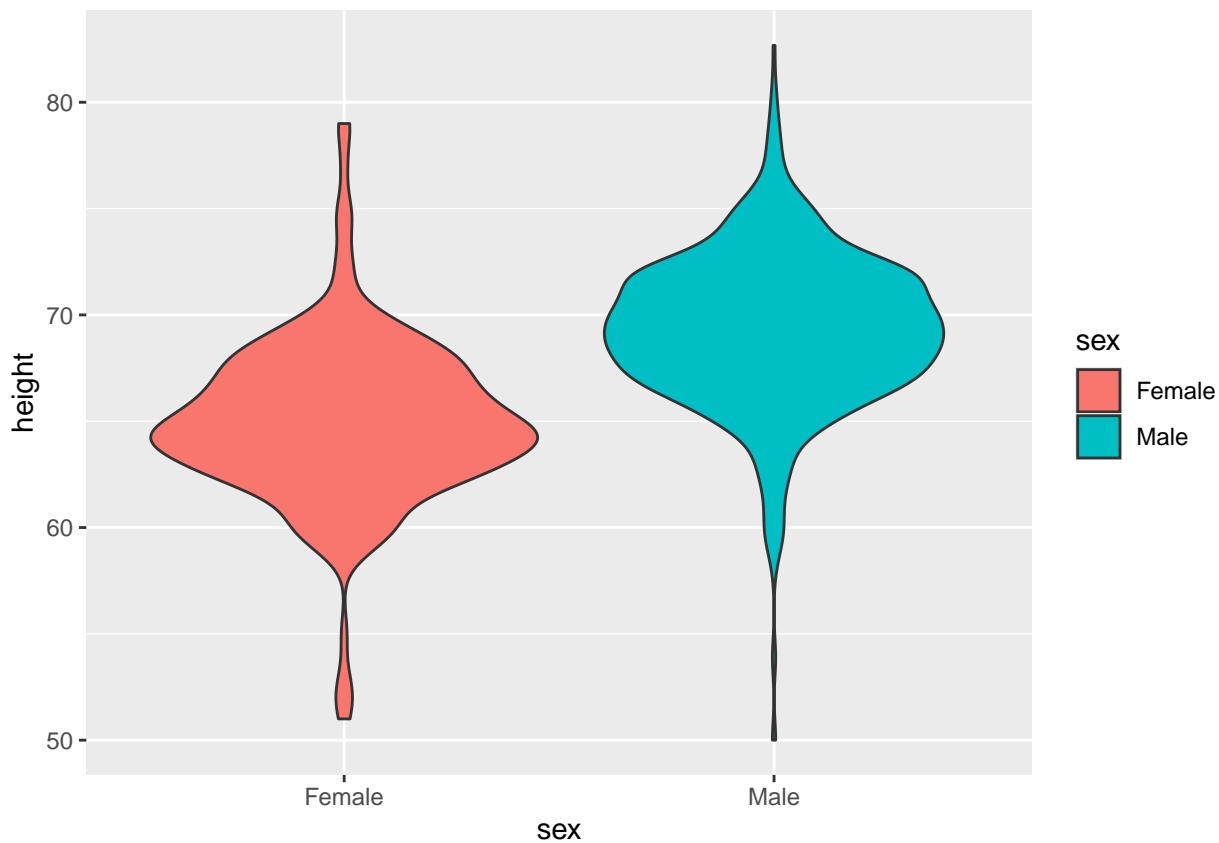
```

data(heights)
b<-heights|>ggplot()
qplot(sex, height, data = heights, geom= "boxplot", fill = sex)

```

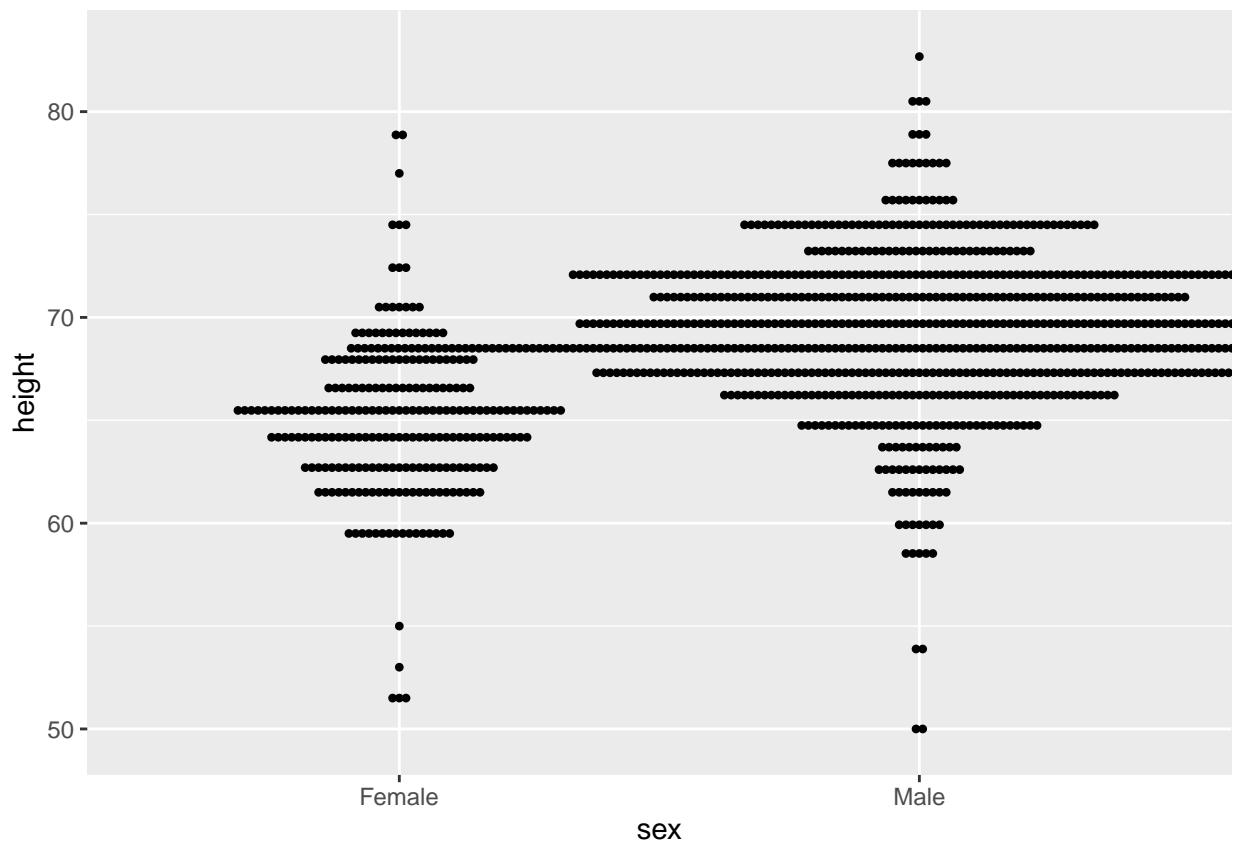


```
qplot(sex, height, data = heights, geom= "violin", fill = sex)
```

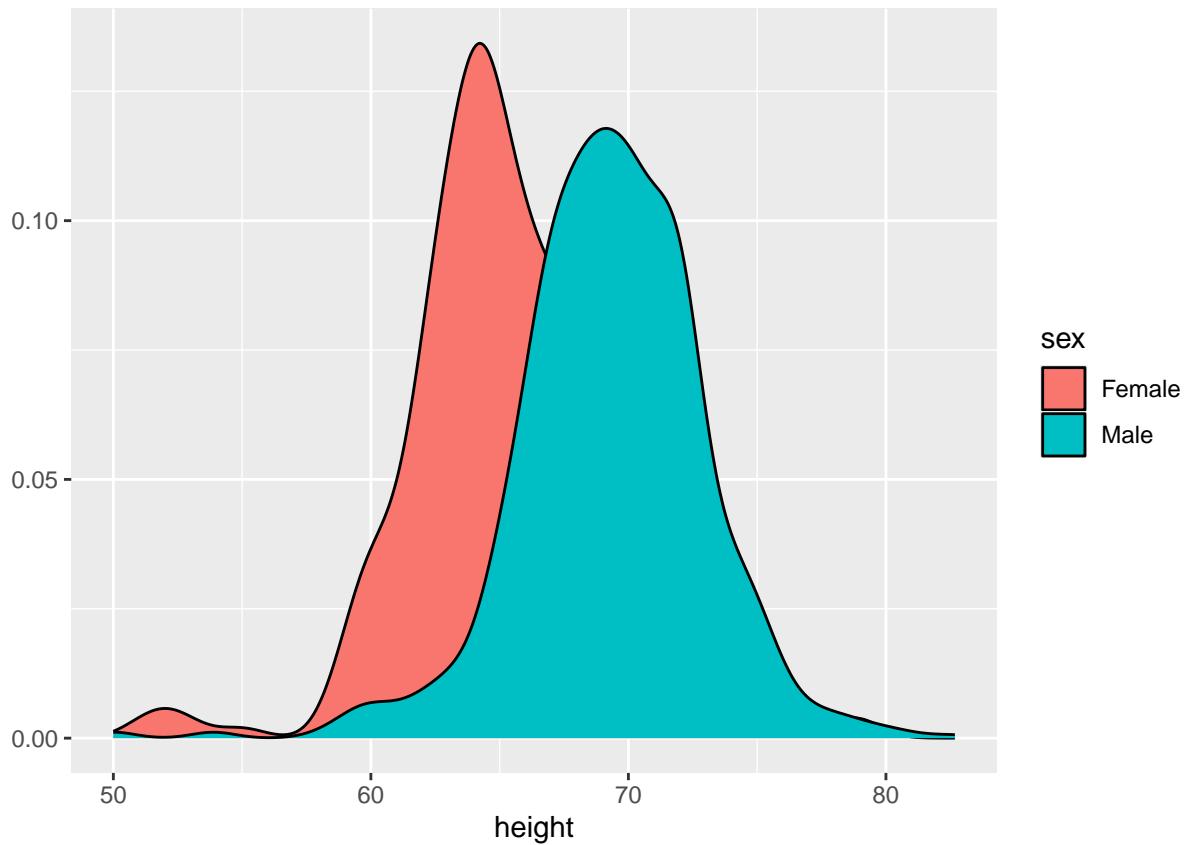


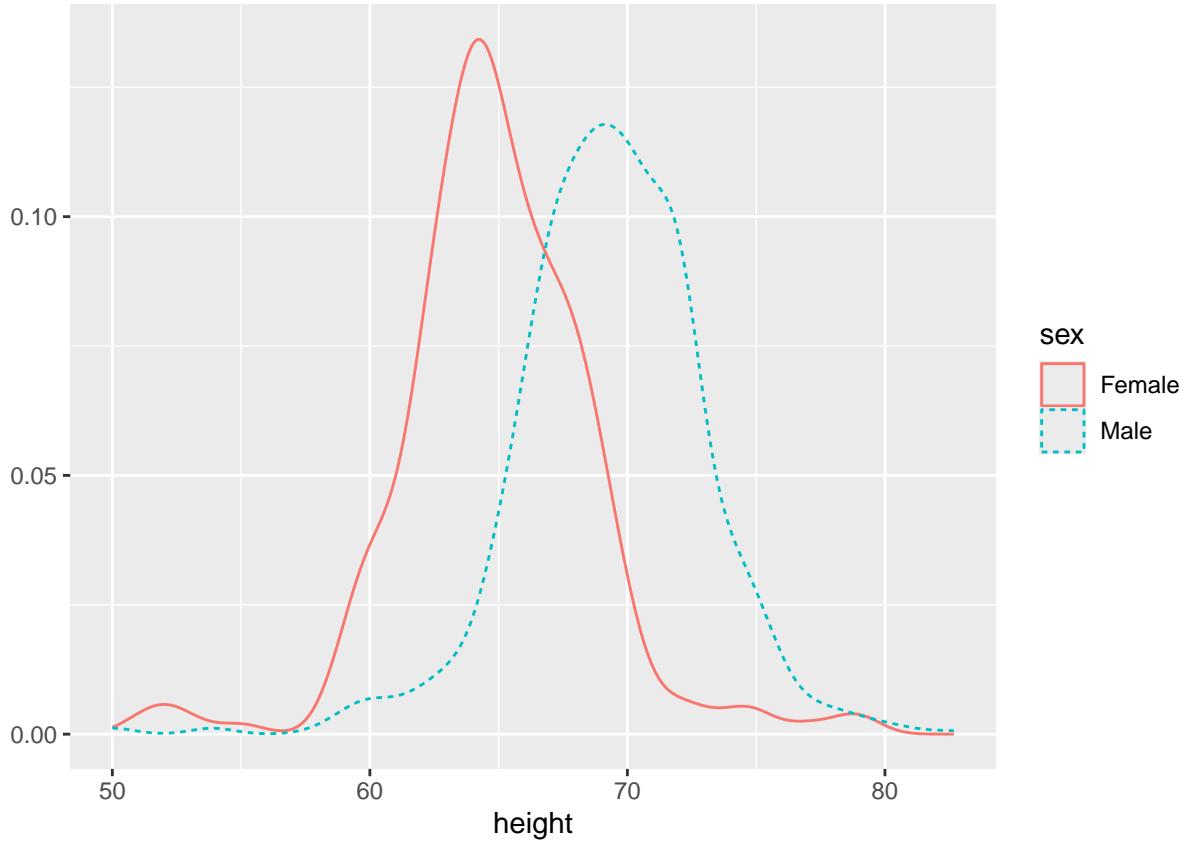
```
qplot(sex, height, data = heights, geom = "dotplot",
      stackdir = "center", binaxis = "y", dotsize = 0.3)
```

```
## Bin width defaults to 1/30 of the range of the data. Pick better value with
## 'binwidth'.
```



```
qplot(height, data = heights, geom = "density", fill = sex)
```

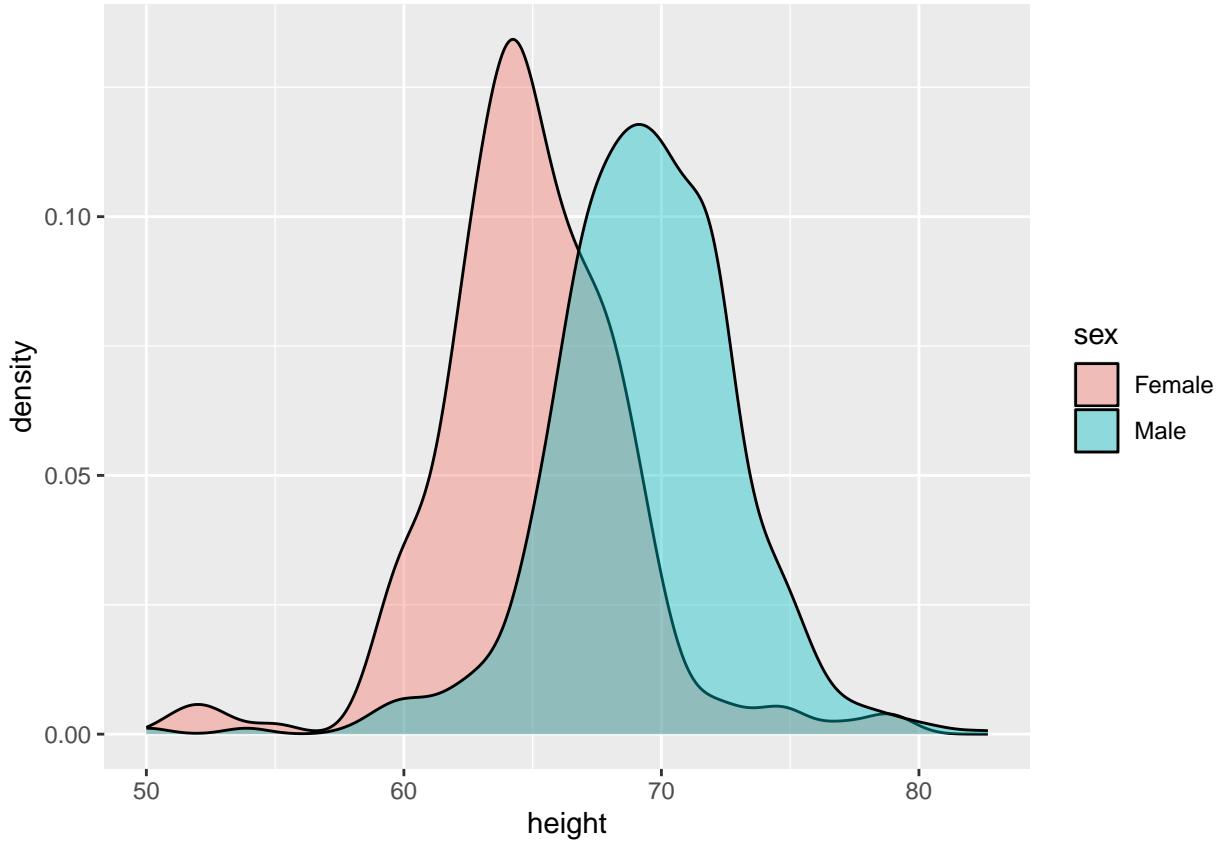




Nuevamente podríamos mostrar información adicional, como medias añadidas fuera de las estéticas definidas dentro del mapeo:

```
mu_alt<-heights |>
  group_by(sex) |>
  summarise(media=mean(height))

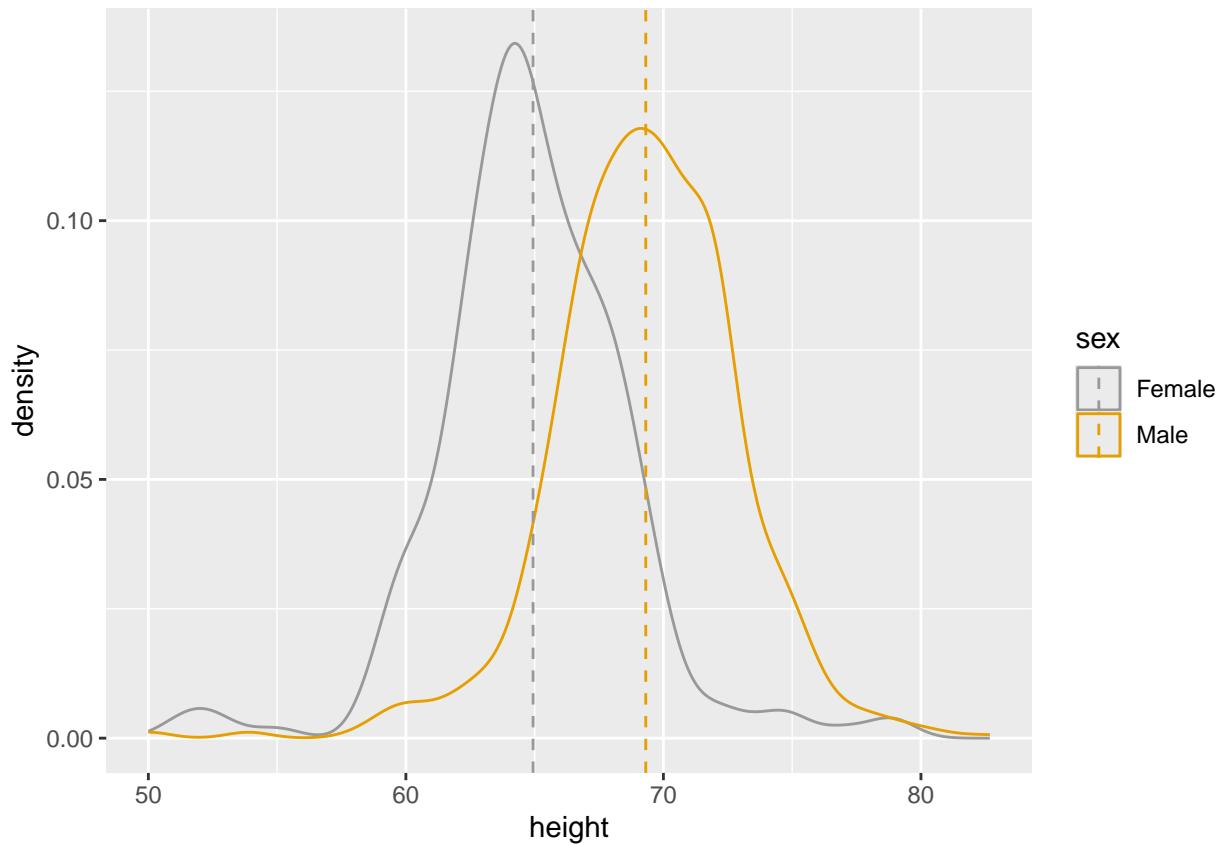
c<-heights|>ggplot(aes(x = height))
c+geom_density(aes(fill = sex), alpha=0.4) #el alpha le da un nivel de transparencia
```



#¿qué creen que pase si suben el valor a 0.9 o lo reducen a 0.1?

Noten que si tenemos dos categorías, al pedir color, tiende a establecer a uno con un rojo magenta y al segundo con un azul cian (aciano o ciano), en este caso suavizado y transparentado para poder ver las distribuciones. Sin embargo, podemos editar el color manualmente, a la vez que añadimos complicaciones como una media de alturas por grupo.

```
c+ geom_density(aes(color = sex)) +
  geom_vline(data=mu_alt, aes(xintercept=media, color=sex),
             linetype="dashed") +
  scale_color_manual(values=c("#999999", "#E69F00")) #noten que aquí le di los colores con código alfanumérico
```



Hay varios métodos para asignar colores al nombrarlos en R. Se puede hacer por nombre (en inglés), con códigos alfanuméricos, y se puede también importar paletas de colores con paquetes específicos. Considero útil revisar las opciones en este enlace: <https://r-graph-gallery.com/ggplot2-color.html>

Alcanzando visualización de densidades y sus cambios a través del tiempo

```
library(dslabs)
data(gapminder)
gapminder |> as_tibble()

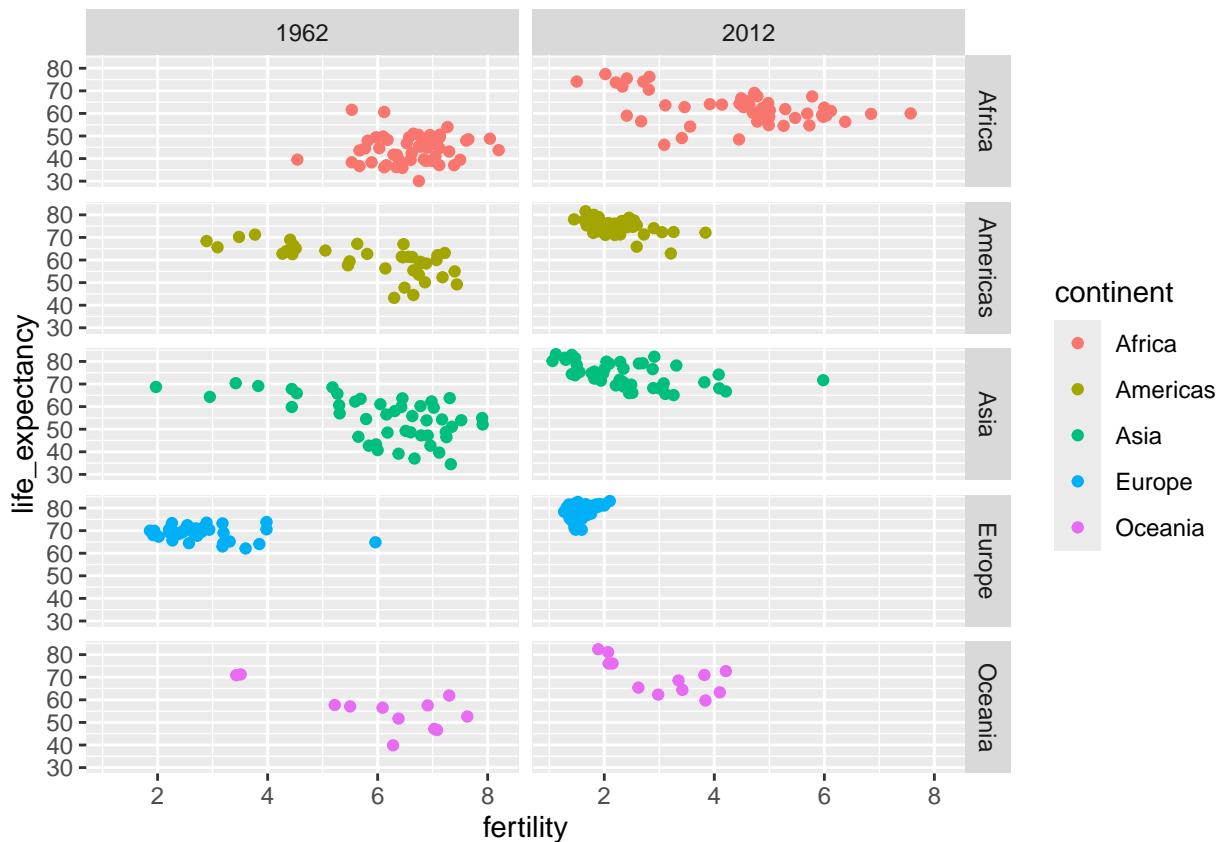
## # A tibble: 10,545 x 9
##   country  year infant_mortality life_expectancy fertility population      gdp
##   <fct>    <int>            <dbl>              <dbl>        <dbl>       <dbl>      <dbl>
## 1 Albania  1960             115.              62.9        6.19     1636054 NA
## 2 Algeria  1960             148.              47.5        7.65     11124892 1.38e10
## 3 Angola   1960             208               36.0        7.32     5270844 NA
## 4 Antigua~ 1960             NA                63.0        4.43     54681 NA
## 5 Argenti~ 1960             59.9               65.4        3.11     20619075 1.08e11
## 6 Armenia  1960             NA                66.9        4.55     1867396 NA
## 7 Aruba    1960             NA                65.7        4.82     54208 NA
## 8 Austral~ 1960             20.3               70.9        3.45     10292328 9.67e10
## 9 Austria  1960             37.3               68.8        2.7      7065525 5.24e10
## 10 Azerbai~ 1960             NA                61.3        5.57     3897889 NA
## # i 10,535 more rows
```

```
## # i 2 more variables: continent <fct>, region <fct>
```

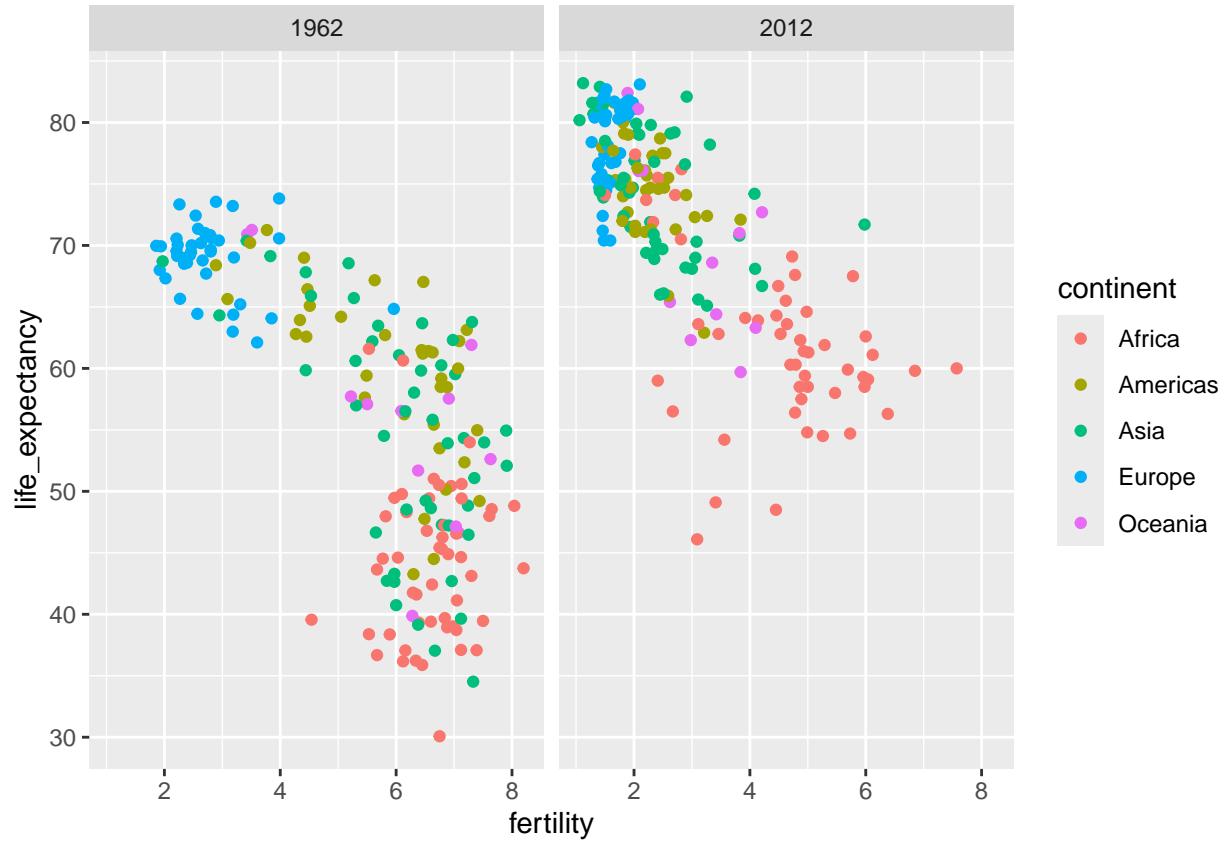
Vemos que estos datos incluyen una variedad de información a nivel de país, que incluye un número de años para ellos.

Podemos ver los grupos de países por continentes usando una matriz de gráficos, con cuadrículas paradas y acostadas:

```
filter(gapminder, year%in%c(1962, 2012)) |>
  ggplot(aes(fertility, life_expectancy, colour = continent)) +
  geom_point() +
  facet_grid(continent~year)
```

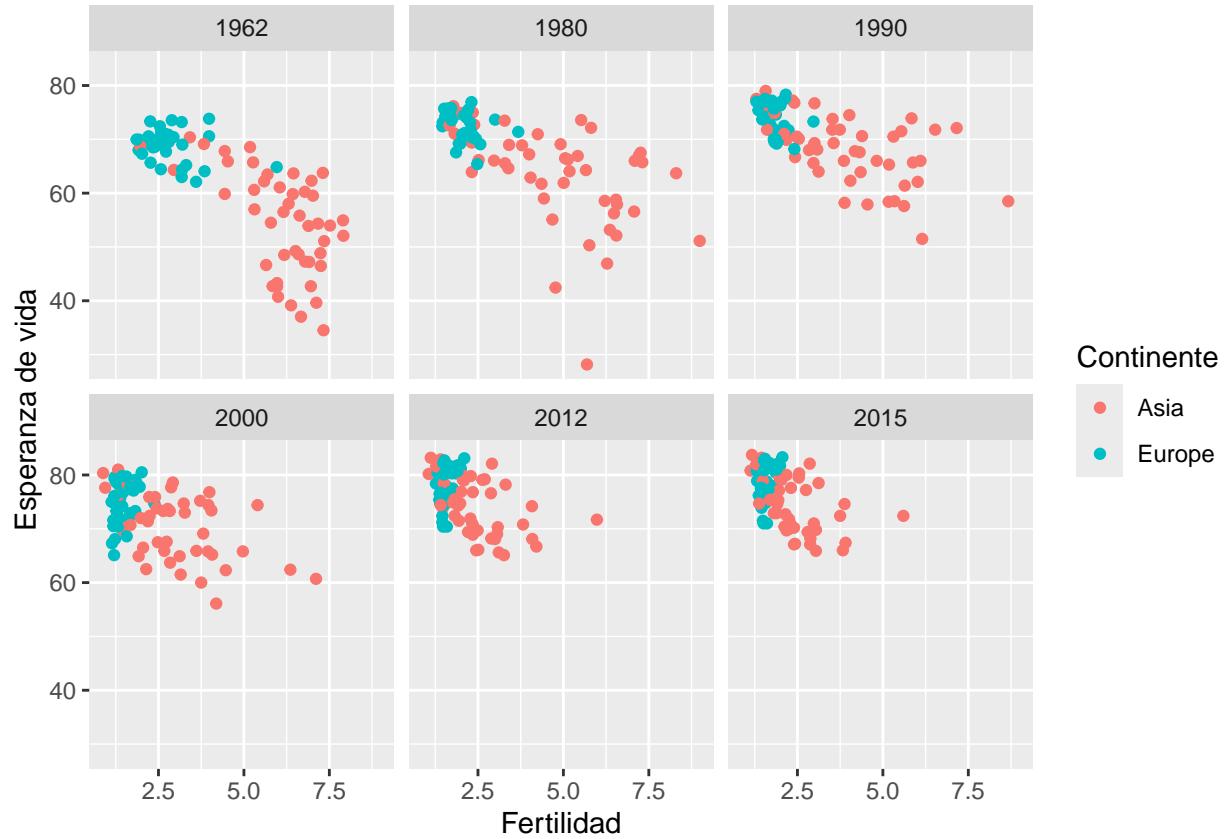


```
filter(gapminder, year%in%c(1962, 2012)) |>
  ggplot(aes(fertility, life_expectancy, col = continent)) +
  geom_point() +
  facet_grid(. ~ year)
```



```

years <- c(1962, 1980, 1990, 2000, 2012, 2015)
continents <- c("Europe", "Asia")
gapminder |>
  filter(year %in% years & continent %in% continents) |>
  ggplot( aes(fertility, life_expectancy, colour = continent)) +
  geom_point() +
  facet_wrap(~year)+labs(x="Fertilidad", y="Esperanza de vida",colour="Continente")
  
```

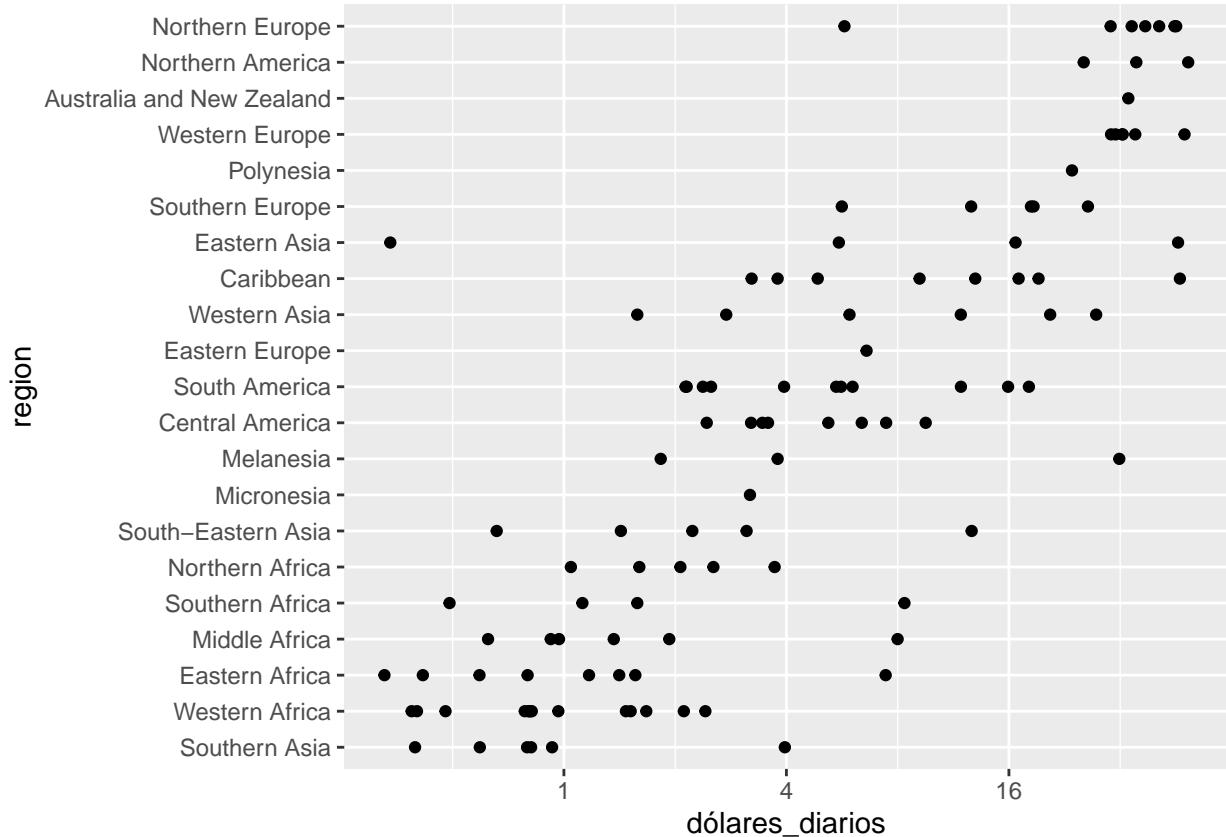


En este caso podemos apreciar mejor cómo la dispersión y distribución de países entre los años 1960s y el presente reciente ha ido encaminado a una convergencia en estándares de vida, dejando atrás los puntos donde originaron estereotipos que aún persisten sobre cómo son las vidas y experiencias de distintas regiones del planeta (sin desestimar diferencias e inequidades presentes).

```
gapminder <- gapminder |> mutate(dólares_diarios = gdp/population/365)

antaño <- 1970

gapminder |>
  filter(year == antaño & !is.na(gdp)) |>
  mutate(region = reorder(region, dólares_diarios, FUN = median)) |>
  ggplot(aes(dólares_diarios, region)) +
  geom_point() +
  scale_x_continuous(trans = "log2")
```

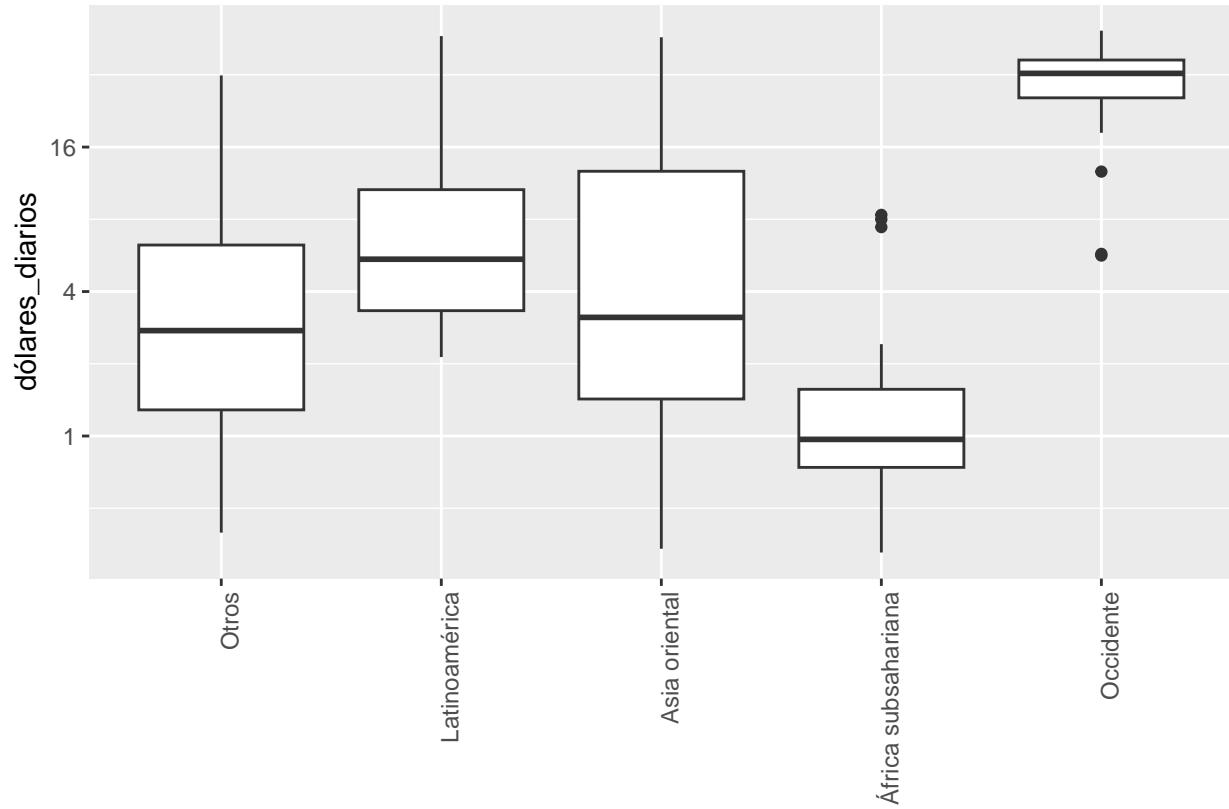


```

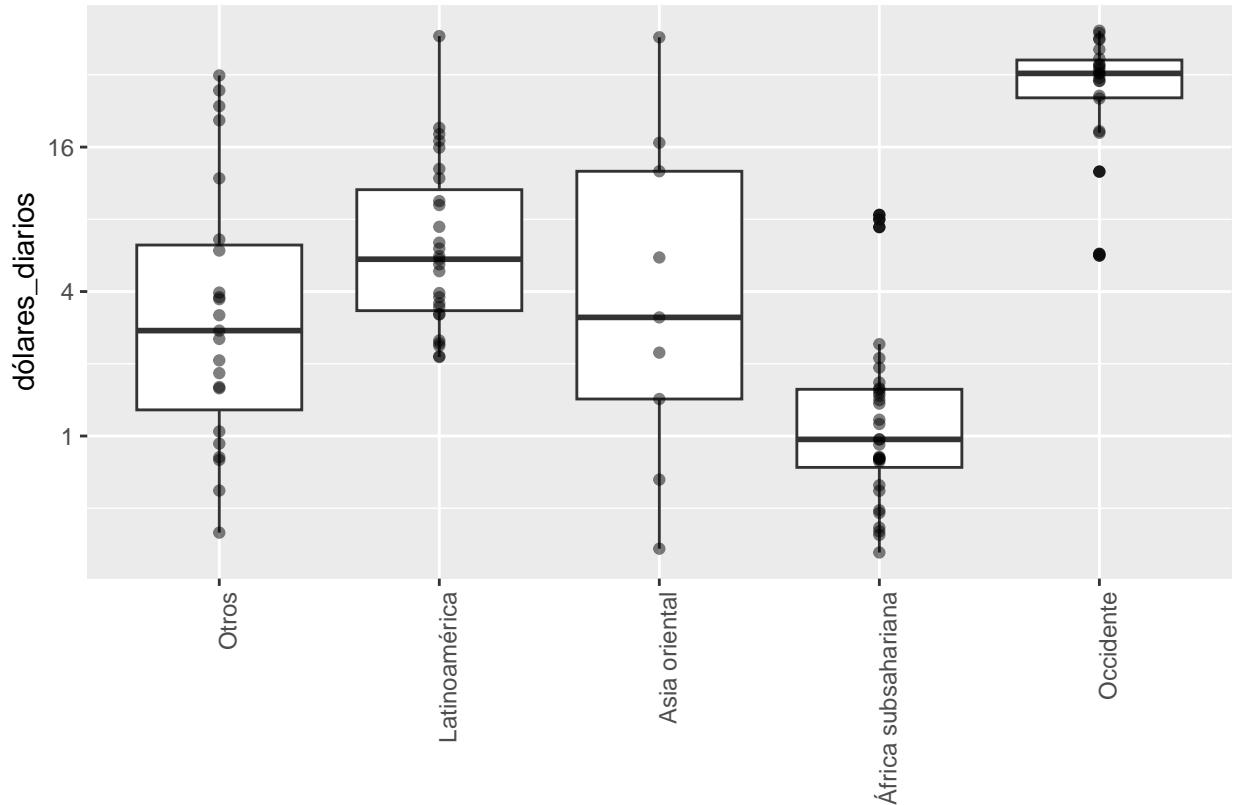
gapminder <- gapminder |>
  mutate(grupo = case_when(
    region %in% c("Western Europe", "Northern Europe", "Southern Europe",
                  "Northern America",
                  "Australia and New Zealand") ~ "Occidente",
    region %in% c("Eastern Asia", "South-Eastern Asia") ~ "Asia oriental",
    region %in% c("Caribbean", "Central America",
                  "South America") ~ "Latinoamérica",
    continent == "Africa" &
      region != "Northern Africa" ~ "África subsahariana",
      TRUE ~ "Otros"))
#le damos orden a los niveles
gapminder <- gapminder |>
  mutate(grupo = factor(grupo, levels = c("Otros", "Latinoamérica",
                                            "Asia oriental", "África subsahariana",
                                            "Occidente")))

p <- gapminder |>
  filter(year == antaño & !is.na(gdp)) |>
  ggplot(aes(grupo, dólares_diarios)) +
  geom_boxplot() +
  scale_y_continuous(trans = "log2") +
  xlab("") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
p

```

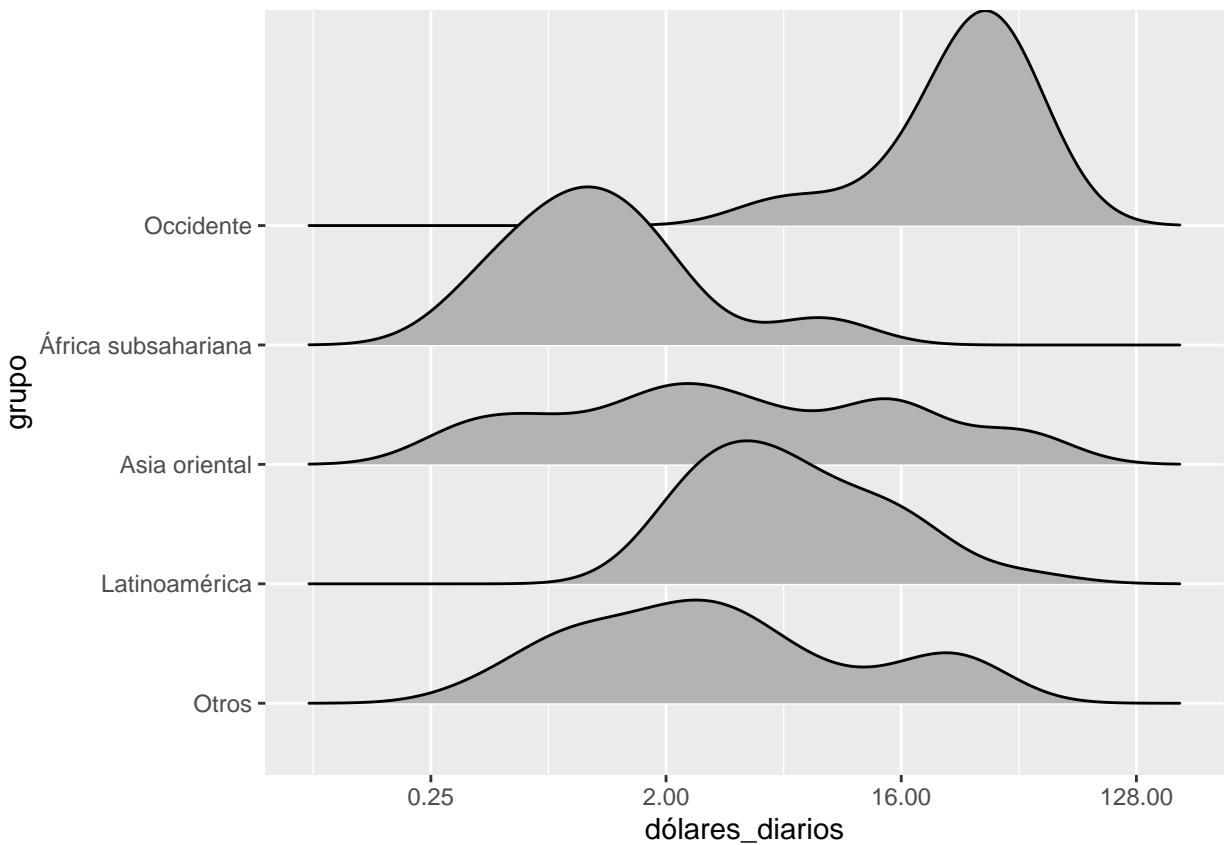


```
p + geom_point(alpha = 0.5) #añadiendo puntos para ver mejor distribuciones y cantidad de casos
```



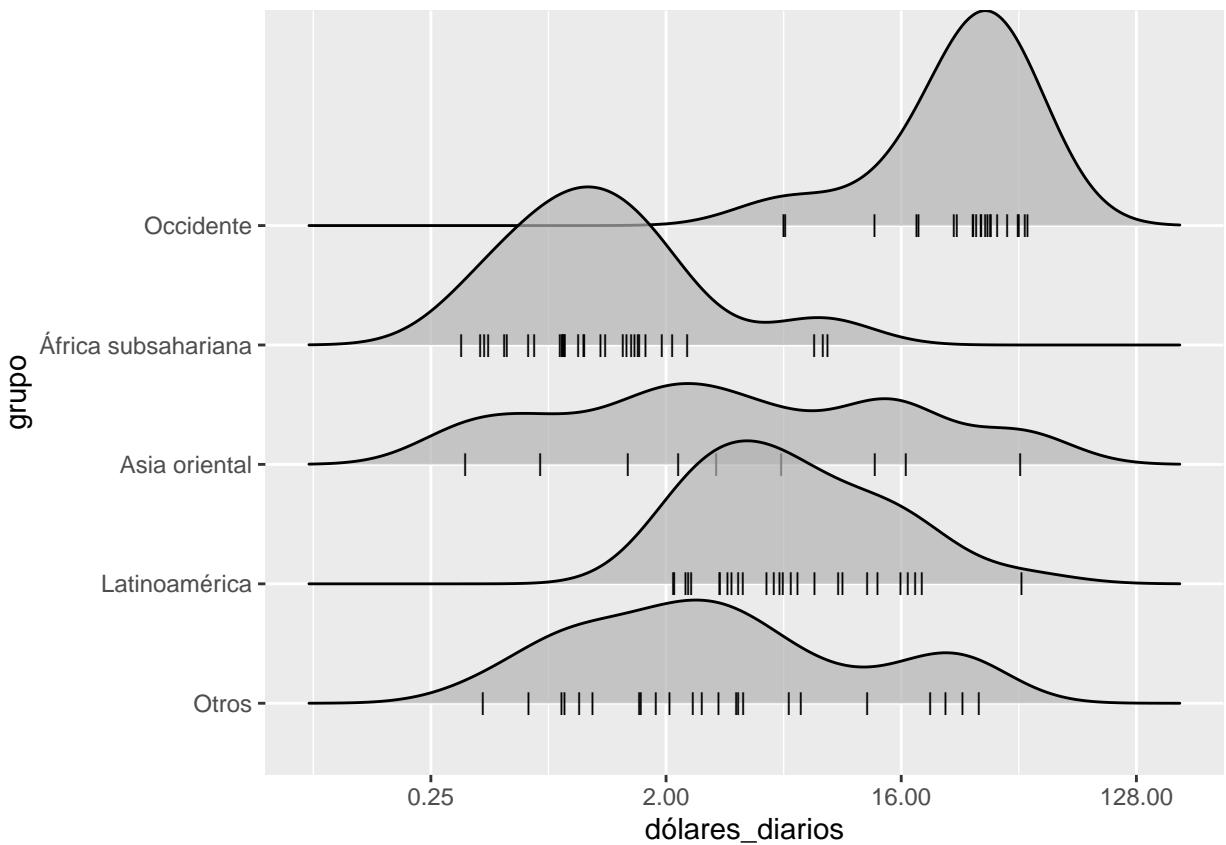
```
library(gggridges)
p <- gapminder |>
  filter(year == antaño & !is.na(dólares_diarios)) |>
  ggplot(aes(dólares_diarios, grupo)) +
  scale_x_continuous(trans = "log2")
p + geom_density_ridges()
```

Picking joint bandwidth of 0.648



```
p + geom_density_ridges(jittered_points = TRUE,
                        position = position_points_jitter(height = 0),
                        point_shape = '|', point_size = 3,
                        point_alpha = 1, alpha = 0.7)
```

```
## Picking joint bandwidth of 0.648
```

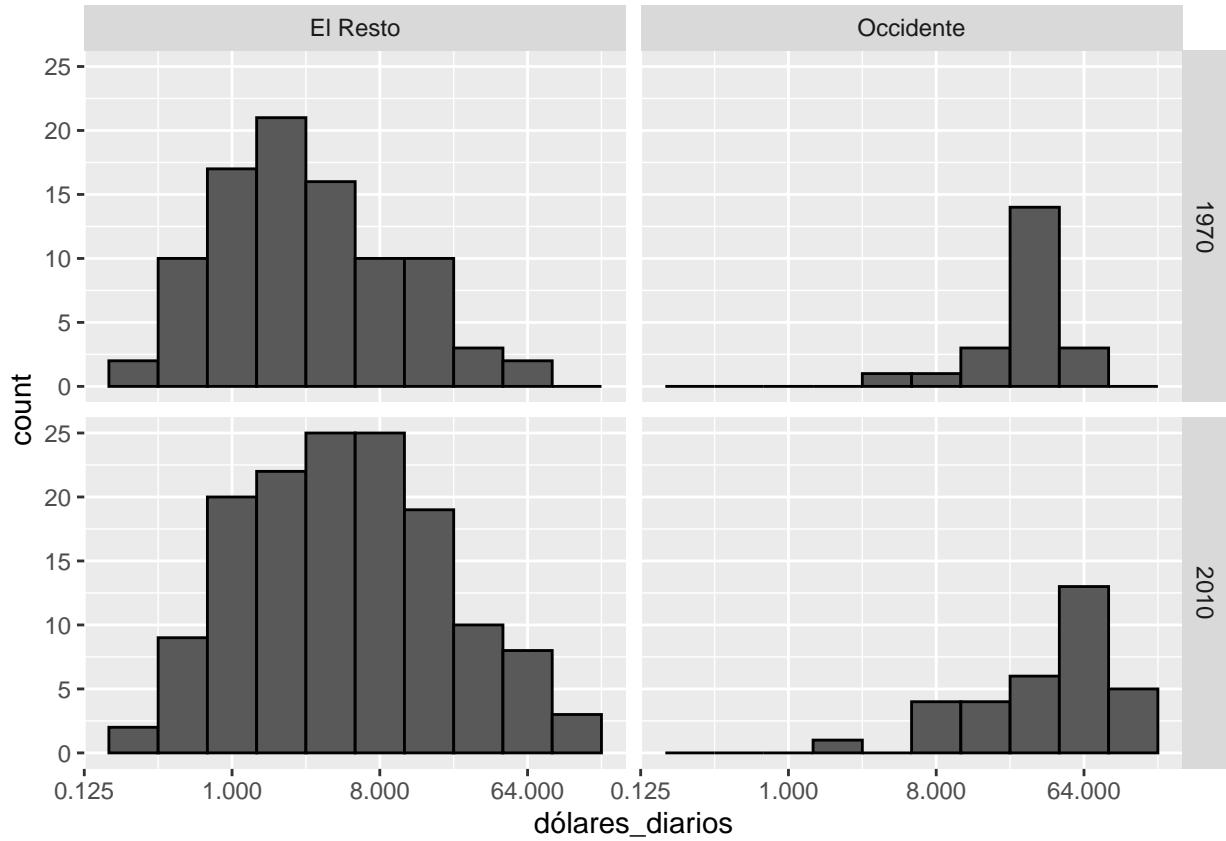


```

antaño <- 1970
año_presente <- 2010

years <- c(antaño, año_presente)
gapminder |>
  filter(year %in% years & !is.na(gdp)) |>
  mutate(west = ifelse(grupo == "Occidente", "Occidente", "El Resto")) |>
  ggplot(aes(dólares_diarios)) +
  geom_histogram(binwidth = 1, color = "black") +
  scale_x_continuous(trans = "log2") +
  facet_grid(year ~ west)

```



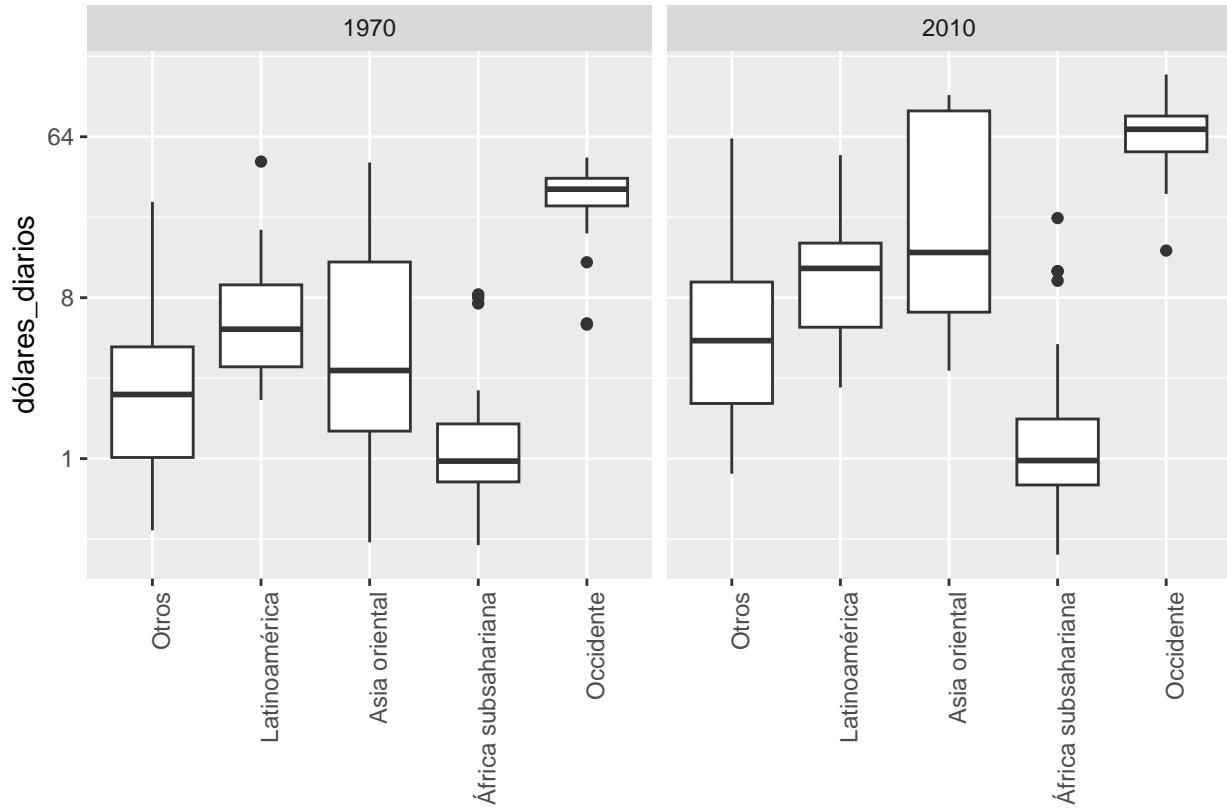
Sabemos que muchos países surgieron después de 1970 (razón por la cual el histograma se nutre para el grupo **El Resto**), así que para comparar los países que tienen toda la información y han existido consistentemente entre esos periodos y ver si hay cambios y de dónde surgen en distribuciones podemos hacer lo siguiente:

```
country_list_1 <- gapminder |>
  filter(year == antaño & !is.na(dólares_diarios)) |>
  pull(country)

country_list_2 <- gapminder |>
  filter(year == año_presente & !is.na(dólares_diarios)) |>
  pull(country)

country_list <- intersect(country_list_1, country_list_2)

gapminder |>
  filter(year %in% years & country %in% country_list) |>
  ggplot(aes(grupo, dólares_diarios)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  scale_y_continuous(trans = "log2") +
  xlab("") +
  facet_grid(. ~ year)
```

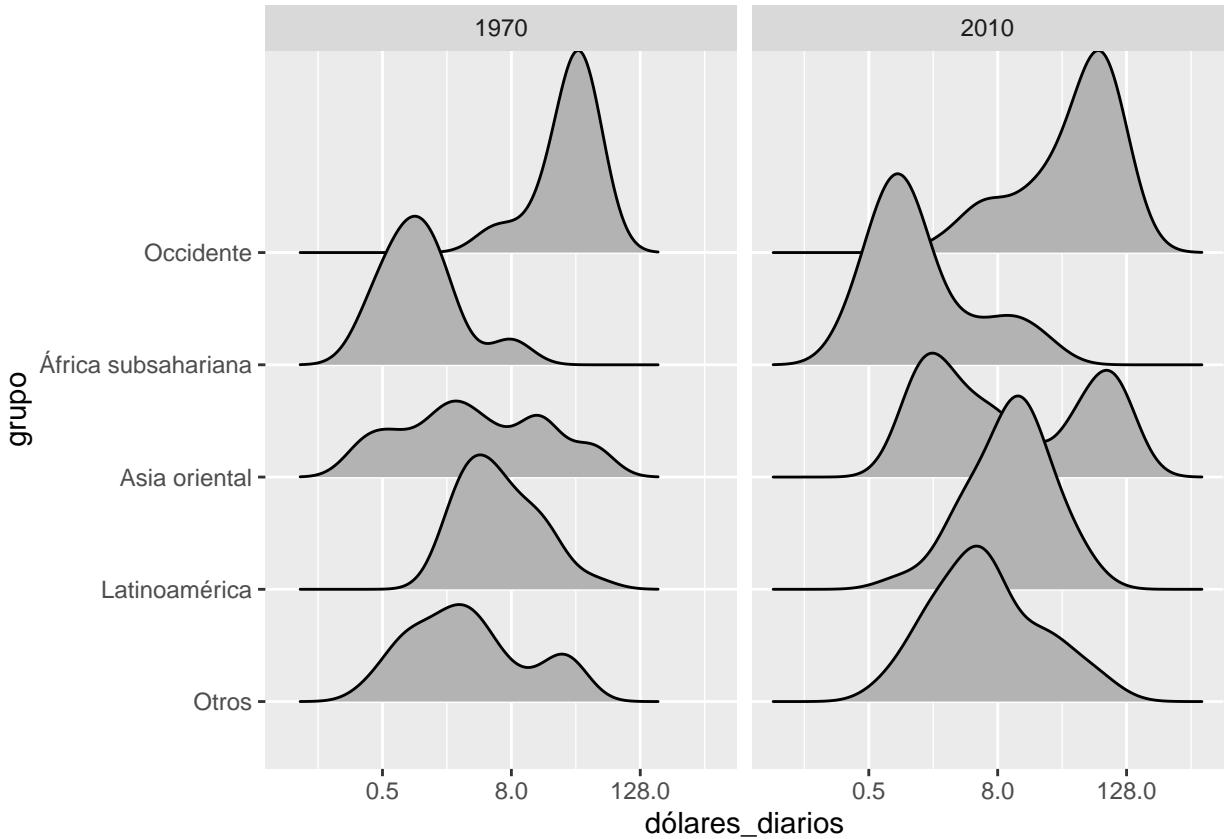


```
gapminder |>
  filter(year %in% years & !is.na(dólares_diarios)) |>
  ggplot(aes(dólares_diarios, grupo)) +
  scale_x_continuous(trans = "log2") +
  geom_density_ridges(adjust = 1.5) +
  facet_grid(. ~ year)
```

```
## Warning in geom_density_ridges(adjust = 1.5): Ignoring unknown parameters:
##   'adjust'
```

```
## Picking joint bandwidth of 0.648
```

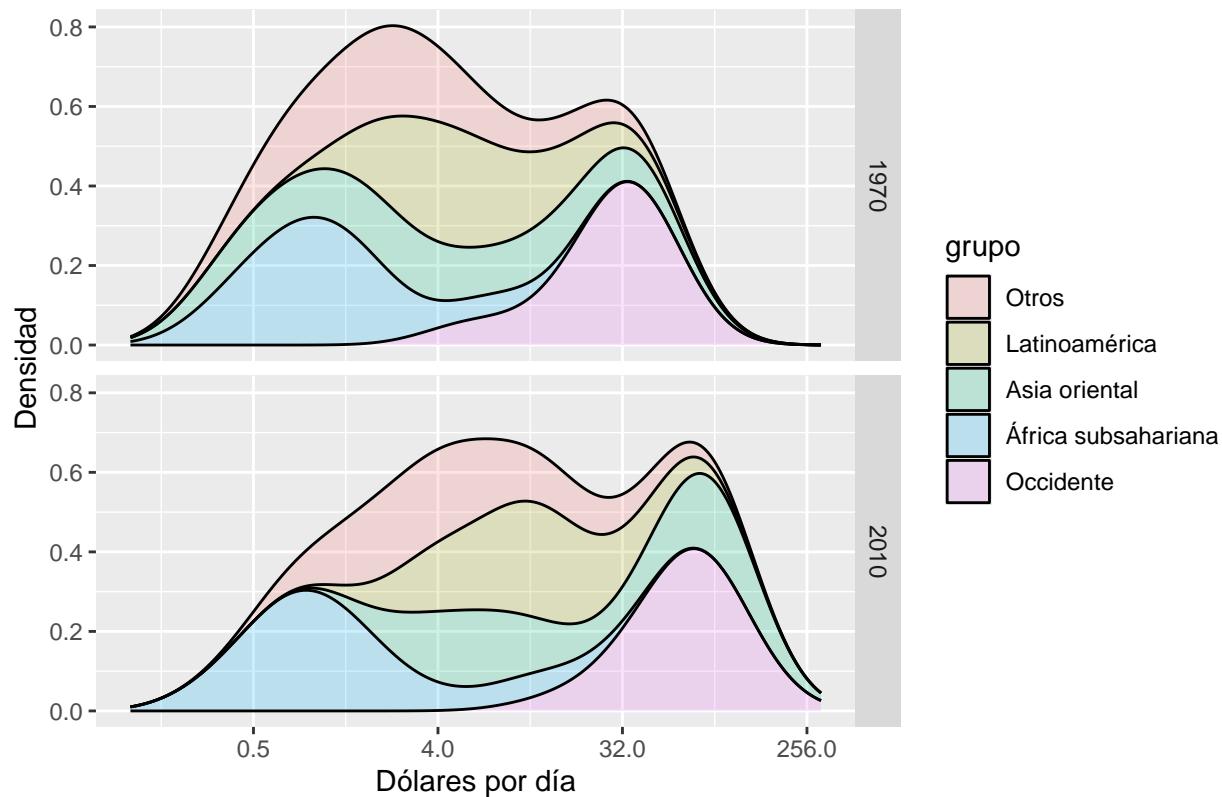
```
## Picking joint bandwidth of 0.726
```



```
gapminder |>
  filter(year %in% years & country %in% country_list) |>
  group_by(year) |>
  mutate(weight = population/sum(population)*2) |>
  ungroup() |>
  ggplot(aes(dólares_diarios, fill = grupo)) +
  scale_x_continuous(trans = "log2", limit = c(0.125, 300)) +
  geom_density(alpha = 0.2, bw = 0.75, position = "stack") +
  facet_grid(year ~ .)+labs(title = "Densidades apiladas: ingreso en dólares por día per cápita entre 1970 y 2010",
    x = "Dólares por día",
    y = "Densidad",
    color = "Regiones")

## Ignoring unknown labels:
## * colour : "Regiones"
```

Densidades apiladas: ingreso en dólares por día per cápita entre 1970 y 2010



Datos del Covid

Durante la pandemia del Coronavirus de 2019, todos pasamos por bastantes cosas, entre ellas, tratar de entender el fenómeno inaudito en nuestras vidas, que era una mortífera y peligrosa pandemia, de la cual desconocíamos en general muchas características. Esto llevó a los gobiernos del mundo a tomar distintos tipos de acciones y nos informábamos con gráficas así como números y tablas del progreso de la enfermedad y su avance a través de los países del planeta, así como la variable tasa de mortalidad que la acompañaba. Varios científicos lanzaron programas para recoger y expresar estos datos al público general para mantenernos todos bien informados.

En mi caso tomé código y datos que se recogían a menudo y produje por unos meses varios tipos de gráficos para representar el desarrollo de la pandemia (y tratar de entender qué ocurría de una manera que yo pudiera digerir e informar). Aquí doy una versión del código que utilicé para estos fines (modificando las fechas ya que se siguió recopilando esa información mucho después de cuando dejara de actualizar esos gráficos).

```
#devtools::install_github("RamiKrispin/coronavirus", force = TRUE)
library(coronavirus)
library(scales)

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard
```

```

## The following object is masked from 'package:readr':
##
##     col_factor

coronavirus<-coronavirus
the_caption = "Fuente: OMS y varios via la Universidad John Hopkins y el paquete de R 'coronavirus' de"
top_countries <- coronavirus |>
  filter(date <= as.Date("2020-07-31")) |>
  filter(type == "confirmed") |>
  group_by(country) |>
  summarise(cases = sum(cases)) |>
  top_n(10, wt = cases)

d2 <- coronavirus |>
  filter(date <= as.Date("2020-07-31")) |> # Filtro para datos dentro de 2020
  group_by(date, country, type) |>
  summarise(cases = sum(cases)) |>
  group_by(date, country) |>
  spread(type, cases) |>
  arrange(date) |>
  group_by(country) |>
  mutate(cfr_cumulative = cumsum(death) / cumsum(confirmed)) |>
  filter(!is.na(cfr_cumulative)) |>
  ungroup() |>
  inner_join(top_countries, by = "country")

```

‘summarise()’ has grouped output by ‘date’, ‘country’. You can override using
the ‘.groups’ argument.

```
summary(as.factor(d2$country))
```

	Brazil	Chile	India	Iran	Mexico
##	157	160	184	164	155
##	Peru	Russia	South Africa	United Kingdom	US
##	148	183	149	184	192

```
today<-as.Date("2020-08-06")
```

```
x_limits <- c(today, NA)
```

```
top_countries
```

```

## # A tibble: 10 x 2
##   country      cases
##   <chr>       <dbl>
## 1 Brazil    2670451
## 2 Chile     355667
## 3 India    1695988
## 4 Iran      304204
## 5 Mexico    424637
## 6 Peru      407492
## 7 Russia    838461
## 8 South Africa 493183

```

```
## 9 US          4548497
## 10 United Kingdom 304789
```

Vemos que los países están en inglés pero queremos que aparezcan en nuestro gráfico en español (ya que interesa informar en mi caso a un público hispanófono). Podemos modificar la información con los paquetes de `tidyverse`:

```
d2<- d2 |>
  mutate(country=recode(country, US = "EEUU", Russia ="Rusia", Mexico = "México", Brazil = "Brasil",
                        "United Kingdom" = "Reino Unido", Japan="Japón",
                        Italy = "Italia", Iran = "Irán", Peru = "Perú",
                        "South Africa"= "Sudáfrica"))
d2<- d2 |>
  mutate(cfr_cumulativeperc=round(cfr_cumulative*100,2))
```

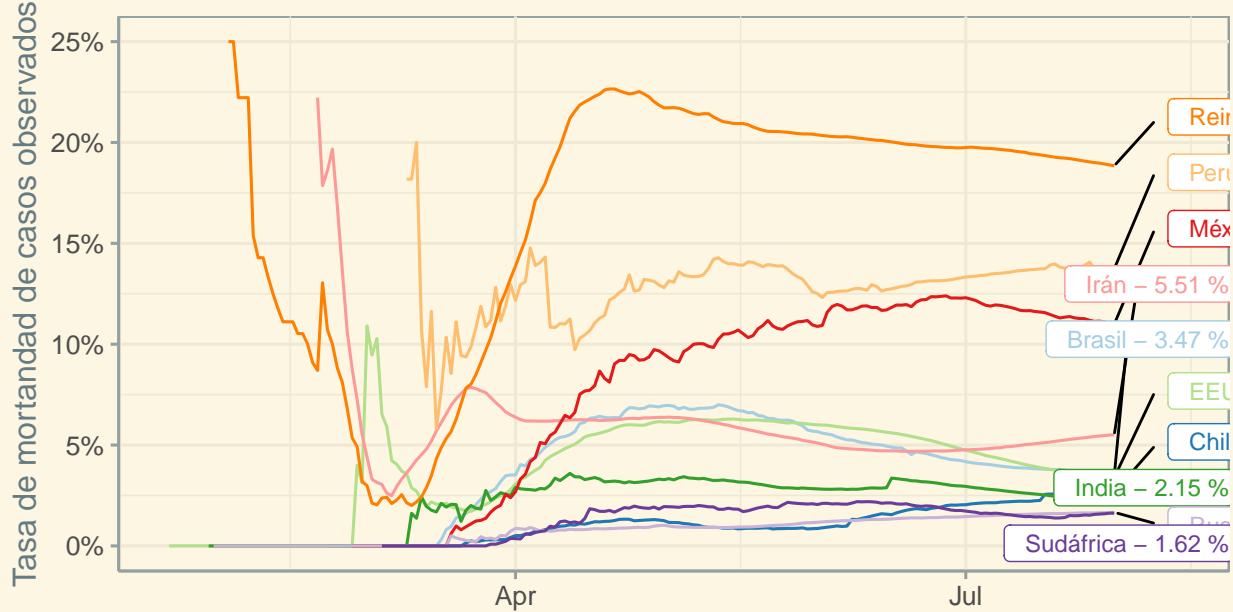
Finalmente, expresamos el gráfico buscado. Paso los datos de mapeo estético como opciones globales y le añado las capas: líneas para series de tiempo, etiquetas automáticamente posicionadas (pero editadas para especificar dónde las quiero y añadir información adicional), transformaciones numéricas, información adicional de selección colores, una extensión del marco para acomodar las etiquetas, así como los títulos que deseara utilizar:

```
d2 |>
  ggplot(aes(x = date, y = cfr_cumulative, colour = country)) +
  geom_line() +
  geom_label_repel(data = filter(d2, date == max(date)), aes(label = paste("",country,"-",cfr_cumulativeperc,
    hjust = 1, size = 3, xlim = x_limits, segment.color="black" ) +
  scale_y_continuous(label = percent_format(accuracy = 1), limits = c(0, .25)) +
  scale_colour_brewer(type = 'qual', palette = 'Paired', direction = 1) +
  expand_limits(x = max(d2$date) + 13) +
  labs(caption = the_caption,
       x = "",
       y = "Tasa de mortandad de casos observados",
       title = "Fatalidad de casos de COVID-19 en los diez países con más casos diagnosticados",
       subtitle = "Los diez países con mayor cantidad de casos diagnosticados. El caso de Irán ha sido omitido por razones de privacidad")
  theme_solarized()+
  theme(legend.position = "none")
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## ('geom_line()').
```

Fatalidad de casos de COVID-19 en los diez países con más casos

Los diez países con mayor cantidad de casos diagnosticados. El caso de Irán ha sido muy variable. Un alto grado de incertidumbre refleja denominadores que cambian con fluidez,



Fuente: OMS y varios via la Universidad John Hopkins y el paquete de R 'coronavirus' de Rami Krispin. Análisis de Rashid C.J. Marcano Rivera, modificando código provisto por <http://freerangestats.info> (Peter Ellis)

Datos del censo

En este bloque introductorio, utilizamos los datos del Censo de Estados Unidos para obtener variables demográficas y socioeconómicas de Puerto Rico. Los datos del Censo nos permiten comprender mejor las características de la población, como ingresos, educación, vivienda, y más, a lo largo del tiempo. Para este ejercicio, hacemos uso de `tidycensus`, una poderosa herramienta que facilita la descarga y manipulación de los datos censales dentro del entorno de R.

El primer paso es cargar las bibliotecas necesarias, como `tidyverse`, `tidycensus` y `sf`. La combinación de estas bibliotecas nos permitirá no solo acceder a los datos, sino también analizarlos y visualizarlos espacialmente en mapas. Es importante también tener configurada una clave API de la Oficina del Censo para poder acceder a los datos.

Luego, se configura el entorno con algunas opciones útiles:

- `scipen = 999`: Evita el uso de notación científica en los números, lo cual facilita la lectura de resultados.
- `tigris_class = "sf"`: Permite manejar las geometrías de manera eficiente mediante `sf`, una clase para manejar datos geoespaciales. A continuación, se usan las funciones de `tidycensus` para cargar variables específicas de diversos conjuntos de datos censales, como la Encuesta de la Comunidad Americana (ACS), la Encuesta de la Comunidad de Puerto Rico (PRCS, que se obtiene aquí vía la función de ACS) y los datos del Censo Decenal. Cada conjunto de datos proporciona información clave sobre diferentes períodos: anual, quinquenal y decenal. Finalmente, mostramos las variables disponibles que podemos utilizar para análisis posteriores.

```

library(tidyverse)
library(tidycensus)
library(sf)

## Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE

#census_api_key(INSERTE SU LLAVE DEL CENSO AQUÍ)
census_api_key("d4e473f5a898ce2107a3af89a7e63a7151c050a3", install= TRUE,overwrite=TRUE)

## Your original .Renvironment will be backed up and stored in your R HOME directory if needed.

## Your API key has been stored in your .Renvironment and can be accessed by Sys.getenv("CENSUS_API_KEY").
## To use now, restart R or run 'readRenvironment("~/Renvironment")'

## [1] "d4e473f5a898ce2107a3af89a7e63a7151c050a3"

options(scipen=999)
options(tigris_class = "sf")
https://api.census.gov/data.html
census_variables <- load_variables(year = 2020, dataset = "acs5", cache = TRUE)
#census_variables_sf1 <- load_variables(year = 2020, dataset = "sf1", cache = TRUE)
#census_variables_sf2 <- load_variables(year = 2020, dataset = "sf2", cache = TRUE)
#census_variables4Redist <- load_variables(year = 2020, dataset = "pl", cache = TRUE)
#v00 <- load_variables(2000, "sf3", cache = TRUE)
v18 <- load_variables(2018, "acs5", cache = TRUE)
v10 <- load_variables(2010, "sf1", cache = TRUE)
census_variables

## # A tibble: 27,850 x 4
##   name      label          concept      geography
##   <chr>     <chr>        <chr>        <chr>
## 1 B01001A_001 Estimate!!Total: SEX BY AGE (W~ tract
## 2 B01001A_002 Estimate!!Total:!!Male: SEX BY AGE (W~ tract
## 3 B01001A_003 Estimate!!Total:!!Male:!!Under 5 years SEX BY AGE (W~ tract
## 4 B01001A_004 Estimate!!Total:!!Male:!!5 to 9 years SEX BY AGE (W~ tract
## 5 B01001A_005 Estimate!!Total:!!Male:!!10 to 14 years SEX BY AGE (W~ tract
## 6 B01001A_006 Estimate!!Total:!!Male:!!15 to 17 years SEX BY AGE (W~ tract
## 7 B01001A_007 Estimate!!Total:!!Male:!!18 and 19 years SEX BY AGE (W~ tract
## 8 B01001A_008 Estimate!!Total:!!Male:!!20 to 24 years SEX BY AGE (W~ tract
## 9 B01001A_009 Estimate!!Total:!!Male:!!25 to 29 years SEX BY AGE (W~ tract
## 10 B01001A_010 Estimate!!Total:!!Male:!!30 to 34 years SEX BY AGE (W~ tract
## # i 27,840 more rows

v10

## # A tibble: 8,959 x 3
##   name      label          concept
##   <chr>     <chr>
## 1 H001001 Total:           HOUSING UNITS
## 2 H002001 Total:           URBAN AND RURAL

```

```

## 3 H002002 Total!!Urban URBAN AND RURAL
## 4 H002003 Total!!Urban!!Inside urbanized areas URBAN AND RURAL
## 5 H002004 Total!!Urban!!Inside urban clusters URBAN AND RURAL
## 6 H002005 Total!!Rural URBAN AND RURAL
## 7 H002006 Total!!Not defined for this file URBAN AND RURAL
## 8 H003001 Total OCCUPANCY STATUS
## 9 H003002 Total!!Occupied OCCUPANCY STATUS
## 10 H003003 Total!!Vacant OCCUPANCY STATUS
## # i 8,949 more rows

```

v18

```

## # A tibble: 26,997 x 4
##   name      label            concept      geography
##   <chr>     <chr>          <chr>        <chr>
## 1 B00001_001 Estimate!!Total UNWEIGHTED SAMP~ block gr-
## 2 B00002_001 Estimate!!Total UNWEIGHTED SAMP~ block gr-
## 3 B01001A_001 Estimate!!Total SEX BY AGE (WHI~ tract
## 4 B01001A_002 Estimate!!Total!!Male SEX BY AGE (WHI~ tract
## 5 B01001A_003 Estimate!!Total!!Male!!Under 5 years SEX BY AGE (WHI~ tract
## 6 B01001A_004 Estimate!!Total!!Male!!5 to 9 years SEX BY AGE (WHI~ tract
## 7 B01001A_005 Estimate!!Total!!Male!!10 to 14 years SEX BY AGE (WHI~ tract
## 8 B01001A_006 Estimate!!Total!!Male!!15 to 17 years SEX BY AGE (WHI~ tract
## 9 B01001A_007 Estimate!!Total!!Male!!18 and 19 years SEX BY AGE (WHI~ tract
## 10 B01001A_008 Estimate!!Total!!Male!!20 to 24 years SEX BY AGE (WHI~ tract
## # i 26,987 more rows

```

Podemos pasar por las variables en catálogo y seleccionar la que nos llame más la atención. Recomiendo verifiquen las variables a través de `View(variables)` ya que pueden usar una opción para filtrar e ir buscando entre las miles de opciones que tienen para uso. En este caso, quiero revisar para algún futuro trabajo de investigación datos sobre el ingreso mediano de los distritos representativos de Puerto Rico para los años 2018 y 2019 (que en el Censo aparecen como geografía tipo `state legislative district (lower chamber)`). Usaré la función `get_acs`, podemos ver qué hace:

```

?get_acs
help("get_acs")

```

Para efectos estadísticos Puerto Rico entra como estado en estos datos. Uso entonces la variable `B19013_001` que da los ingresos medianos por hogar.

```

prmedian_2019 <- get_acs(geography = "state legislative district (lower chamber)",
                           variables = c(medincome = "B19013_001"),
                           state = "PR",
                           year = 2019)

```

```

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

```

```

prmedian_2018 <- get_acs(geography = "state legislative district (lower chamber)",
                           variables = c(medincome = "B19013_001"),
                           state = "PR",
                           year = 2018)

```

```

## Getting data from the 2014-2018 5-year ACS

head(prmedian_2018)

## # A tibble: 6 x 5
##   GEOID NAME          variable estimate    moe
##   <chr> <chr>        <chr>      <dbl> <dbl>
## 1 72001 State House District 1 (2018), Puerto Rico medincome 18336 955
## 2 72002 State House District 2 (2018), Puerto Rico medincome 18343 884
## 3 72003 State House District 3 (2018), Puerto Rico medincome 20315 1394
## 4 72004 State House District 4 (2018), Puerto Rico medincome 33471 1586
## 5 72005 State House District 5 (2018), Puerto Rico medincome 23274 1619
## 6 72006 State House District 6 (2018), Puerto Rico medincome 32921 1581

pr_median_combined <- prmedian_2019 |>
  mutate(year = 2019) |>
  bind_rows(
    prmedian_2018 |> mutate(year = 2018)
  )
head(pr_median_combined)

## # A tibble: 6 x 6
##   GEOID NAME          variable estimate    moe year
##   <chr> <chr>        <chr>      <dbl> <dbl> <dbl>
## 1 72001 State House District 1 (2018), Puerto Rico medinco~ 18523 1142 2019
## 2 72002 State House District 2 (2018), Puerto Rico medinco~ 19004 995 2019
## 3 72003 State House District 3 (2018), Puerto Rico medinco~ 20655 1400 2019
## 4 72004 State House District 4 (2018), Puerto Rico medinco~ 33678 1519 2019
## 5 72005 State House District 5 (2018), Puerto Rico medinco~ 25912 1733 2019
## 6 72006 State House District 6 (2018), Puerto Rico medinco~ 33977 1491 2019

```

Habiendo obtenido las tablas, digamos que no quiero hacer sólo un análisis numérico a ojo. El ojo y mente humana rara vez puede determinar mucho al ver una colección grande de estos datos rápidamente.

Pero al visualizar, podemos mejorar algo la velocidad en lo que entendemos algo más de lo que está ante nos.

```

pr_median_combined <- pr_median_combined |>
  filter(!is.na(estimate)) # Filter out rows where the estimate is NA

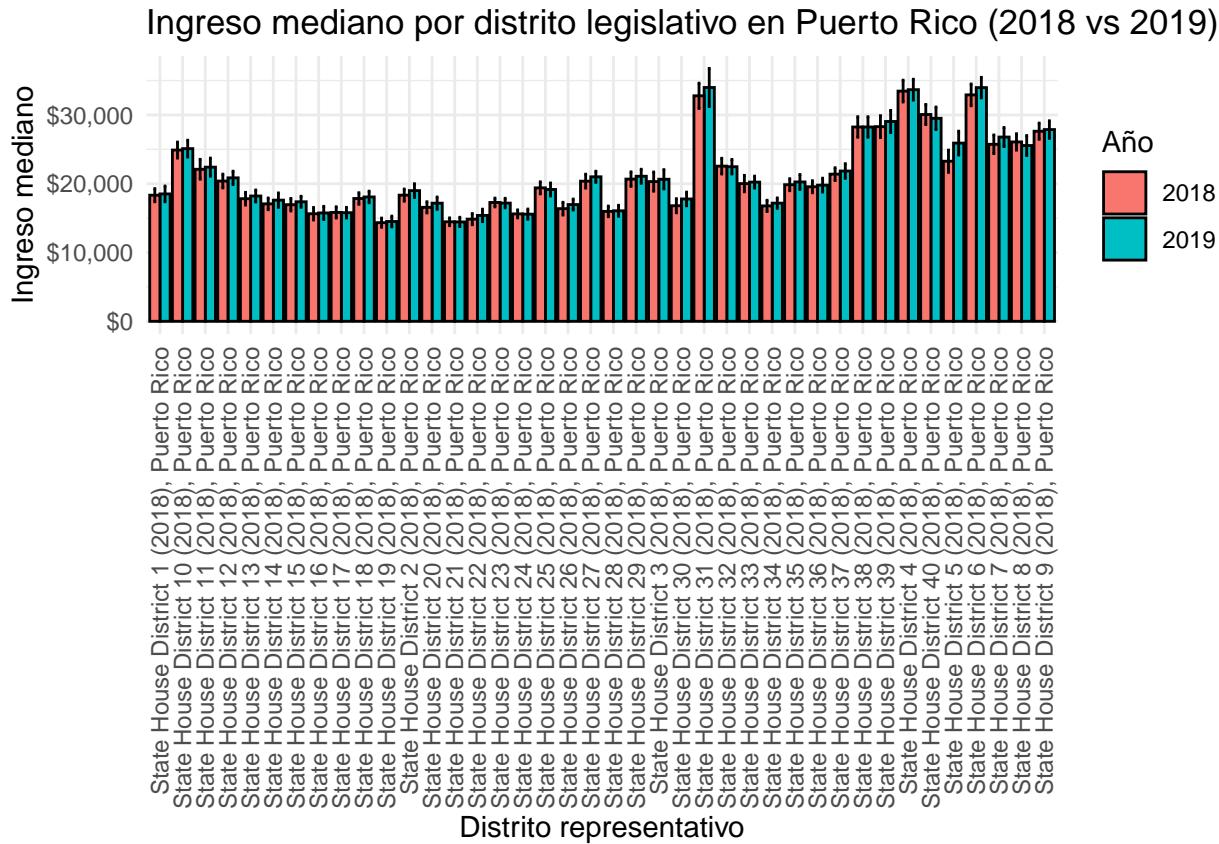
pr_median_combined |>
  ggplot(aes(x = NAME, y = estimate, fill = factor(year))) +
  geom_bar(stat = "identity", position = position_dodge(), color = "black") +
  geom_errorbar(aes(ymin = estimate - moe, ymax = estimate + moe),
                position = position_dodge(0.9), width = 0.25) +
  scale_y_continuous(labels = scales::dollar_format()) +
  labs(
    title = "Ingreso mediano por distrito legislativo en Puerto Rico (2018 vs 2019)",
    x = "Distrito representativo",
    y = "Ingreso mediano",
    fill = "Año"
  ) +
  theme_minimal() +

```

```

theme(
  axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1) # Rotar nombres de distritos 90 grados
)

```



Noten que ordenó los distritos de una manera extraña: el 4 aparece antes de 40 y todos del 5 al 9 se fueron como posteriores al 40. Voy a corregir y renombrar los distritos:

```

pr_median_combined <- pr_median_combined |>
  # Extraer el número del distrito desde el nombre original y convertirlo en numérico
  mutate(district_number = as.numeric(gsub(".*District ([0-9]+).*", "\\\\1", NAME))) |>
  # Renombrar el campo NAME para que tenga el formato "Distrito 01" hasta "Distrito 40"
  mutate(NAME = sprintf("Distrito %02d", district_number)) |>
  # Ordenar los datos según el número de distrito
  arrange(district_number)

# Crear el gráfico nuevamente
pr_median_combined |>
  ggplot(aes(x = NAME, y = estimate, fill = factor(year))) +
  geom_bar(stat = "identity", position = position_dodge(), color = "black") +
  geom_errorbar(aes(ymin = estimate - moe, ymax = estimate + moe),
                position = position_dodge(0.9), width = 0.25) +
  scale_y_continuous(labels = scales::dollar_format()) +
  labs(
    title = "Ingreso mediano por distrito representativo en Puerto Rico (2018 vs 2019)",
    x = "Distrito representativo",
    y = "Ingreso mediano",
  )

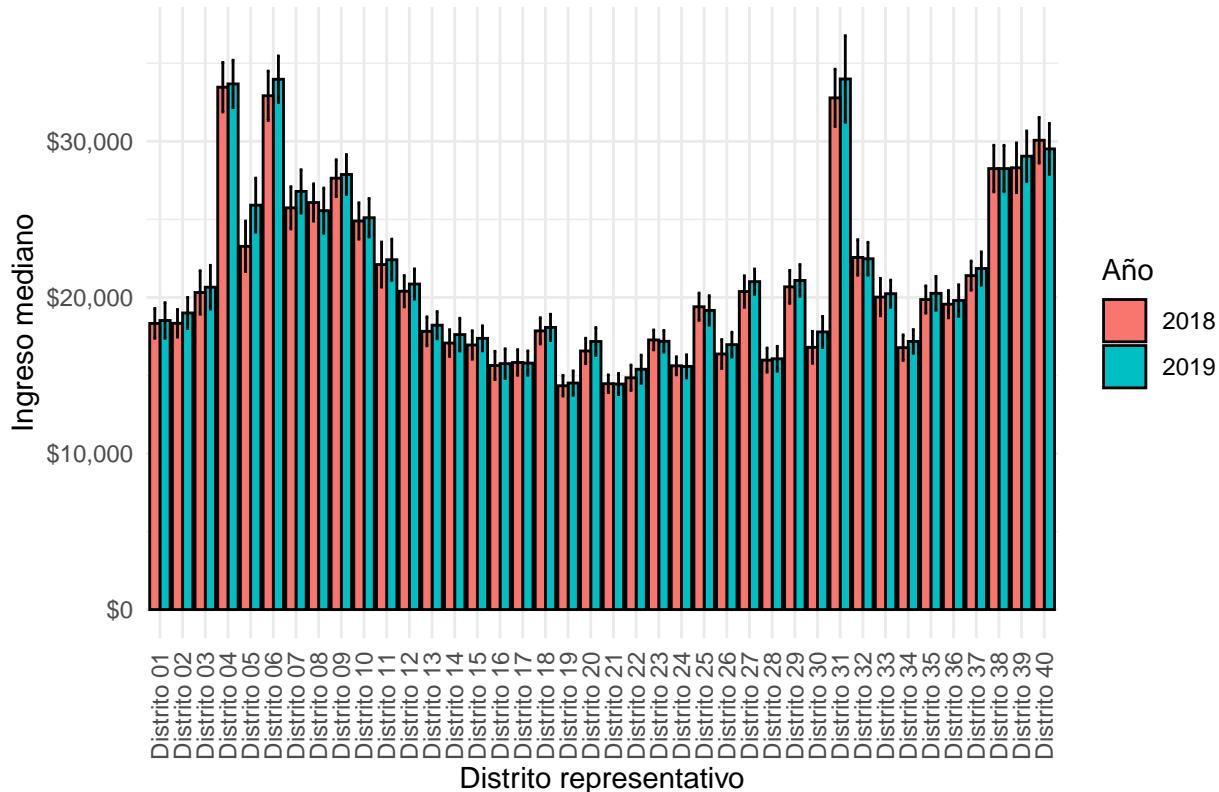
```

```

    fill = "Año"
) +
theme_minimal() +
theme(
  axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1) # Rotar nombres de distritos 90 grados
)

```

Ingreso mediano por distrito representativo en Puerto Rico (2018 vs 2019)



Finalmente, usaremos esta información para generar un mapa geospatial con ggplot. En este caso tomamos como parte de los pasos una transformación de coordenadas para que el mapeo entienda dónde posicionar las geometrías en el mapeo estético.

```

library(tigris)

## To enable caching of data, set 'options(tigris_use_cache = TRUE)'
## in your R script or .Rprofile.

pr_legislative_districts <- state_legislative_districts(state = "PR", house = "lower", year = 2018)

## | 

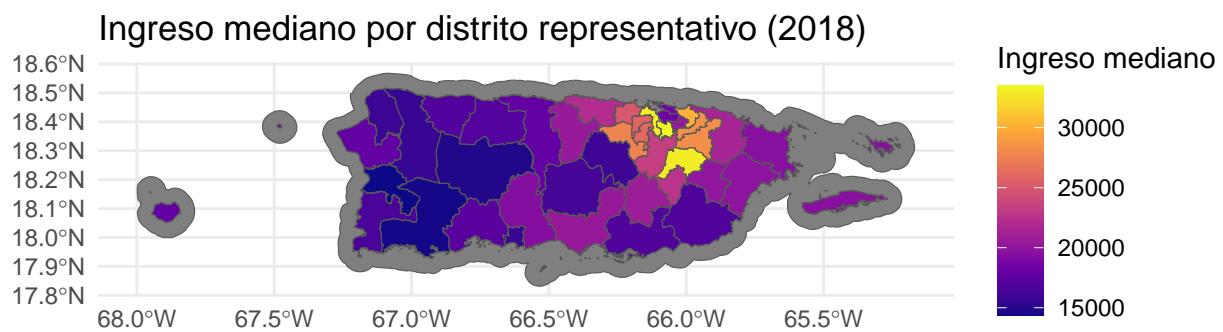
pr_legislative_districts <- st_transform(pr_legislative_districts, crs = "EPSG:4326") # ajustar proyección
pr_median_2018_geo <- pr_legislative_districts |>
  left_join(prmedian_2018, by = c("GEOID" = "GEOID"))
pr_median_2018_geo|>

```

```

ggplot() +
  geom_sf(aes(fill = estimate)) +
  scale_fill_viridis_c(option = "plasma", name = "Ingreso mediano") +
  theme_minimal() +
  labs(
    title = "Ingreso mediano por distrito representativo (2018)",
    fill = "Ingreso mediano"
)

```



En este bloque próximo, utilizamos datos espaciales y demográficos para mapear la población de Puerto Rico a nivel de tractos censales del censo de 2020. Combinamos dos fuentes de información: una capa espacial con los límites geográficos de los tractos (la forma de los polígonos en un *shapefile*: <https://electionspuertorico.org/datos/2020/#MAPA>) y los datos poblacionales del Censo Decenal de 2020.

1. El shapefile nos proporciona los límites de los tractos que necesitamos para asignar la información demográfica correctamente.
 - **st_read()**: Carga el shapefile, aplicando la codificación `latin1` para asegurarnos de que los caracteres especiales (acentos, diéresis, eñes) en los nombres se lean correctamente.
 - **st_set_crs() y st_transform()**: Aquí ajustamos el CRS (sistema de referencia de coordenadas) de la capa geográfica. Primero asignamos el CRS EPSG:4326 (coordenadas geográficas), y luego transformamos la capa a EPSG:3920 (un CRS proyectado que es común para Puerto Rico) para asegurar que todas las capas estén en el mismo sistema de referencia.
2. **Datos de población:** Usamos `tidycensus` para obtener los datos de población de Puerto Rico a nivel de tractos censales del censo de 2020.

- `get_decennial()`: Esta función descarga los datos de población (variable P1_001N, que representa el total de la población). La opción `geometry = TRUE` incluye la geometría de los trácticos, lo cual nos permite hacer un análisis espacial directamente.
- `st_transform()`: Aplicamos la misma transformación de CRS a los datos de población para que coincidan con la capa geográfica de los trácticos.

3. **Unión espacial:** Utilizamos `st_join()` para combinar los datos espaciales y de población, uniendo la capa de trácticos con los datos del censo según el campo `GEOID`. Esto nos permite representar los datos poblacionales en el mapa.

```
# Este archivo lo descargué el 9 de agosto de 2022 desde https://electionspuertorico.org/datos/2020/PR2020TV.shp
# folder<-("~/Library/CloudStorage/OneDrive-IndianaUniversity/Elecciones/Rashid and Brevin/")

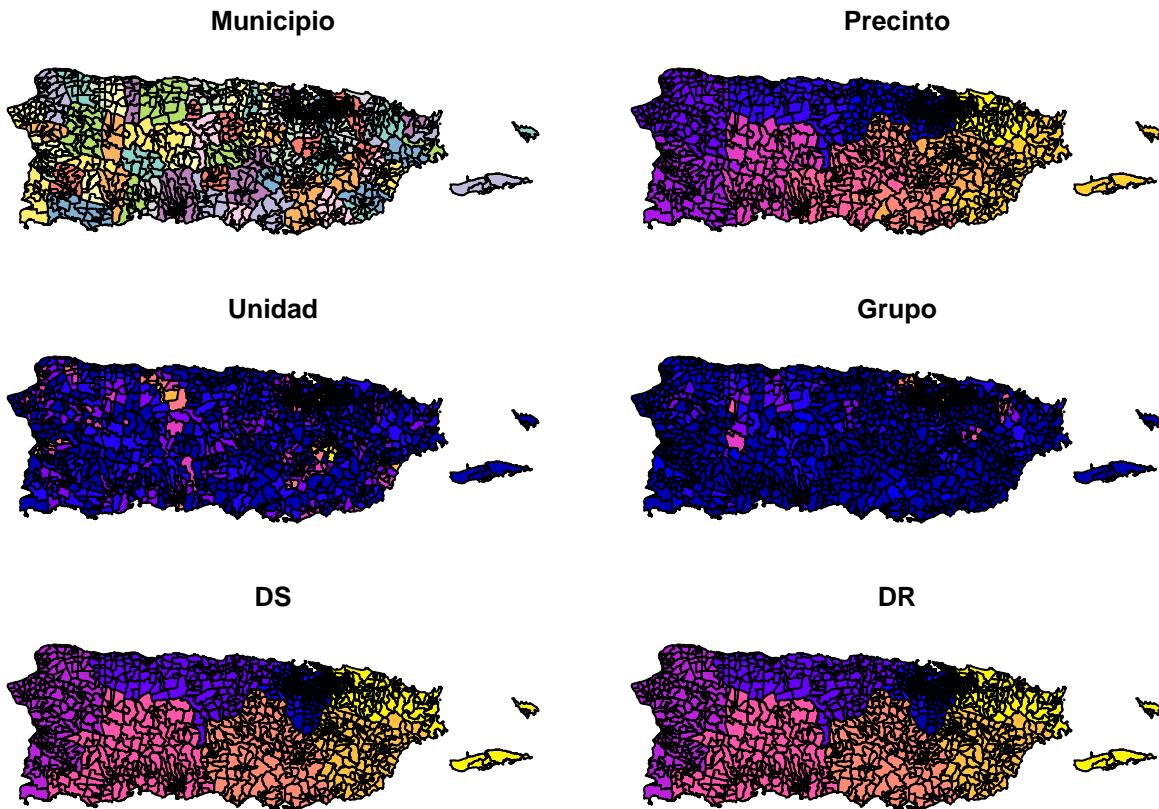
pr_unidad_shp_filename <- paste(folder, "PR2020Shp/PR2020TV.shp", sep= "")
pr_unidades <- st_read(pr_unidad_shp_filename, options = "ENCODING=latin1")

## options:           ENCODING=latin1
## Reading layer 'PR2020TV' from data source
##   '/Users/rashid/Library/CloudStorage/OneDrive-IndianaUniversity/Elecciones/Rashid and Brevin/PR2020TV.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 1365 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: -67.2713 ymin: 17.92684 xmax: -65.2442 ymax: 18.51609
## Geodetic CRS:  WGS 84

st_set_crs(pr_unidades, "EPSG:4326")

## Simple feature collection with 1365 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: -67.2713 ymin: 17.92684 xmax: -65.2442 ymax: 18.51609
## Geodetic CRS:  WGS 84
## First 10 features:
##   Municipio Precinto Unidad Grupo DS DR           geometry
## 1 Barranquitas     70      5    3  6 26 MULTIPOLYGON (((-66.30361 1...
## 2 Cabo Rojo        46      2    3  4 20 MULTIPOLYGON (((-67.14746 1...
## 3 Cabo Rojo        46      3    3  4 20 MULTIPOLYGON (((-67.16632 1...
## 4 Cabo Rojo        46      9    2  4 20 MULTIPOLYGON (((-67.1375 18...
## 5 Cabo Rojo        46      6    3  4 20 MULTIPOLYGON (((-67.16957 1...
## 6 San Germán       44      3    1  4 20 MULTIPOLYGON (((-67.08108 1...
## 7 San Germán       43      7    2  4 19 MULTIPOLYGON (((-67.02885 1...
## 8 San Germán       43      6    2  4 19 MULTIPOLYGON (((-67.04865 1...
## 9 San Germán       44      4    1  4 20 MULTIPOLYGON (((-67.06009 1...
## 10 San Germán      43      2    1  4 19 MULTIPOLYGON (((-67.08731 1...

pr_unidades <- pr_unidades |> st_transform(crs = "EPSG:3920")
plot(pr_unidades)
```



```
pr_population <- get_decennial(
  geography = "tract",
  variables = "P1_001N",
  state = "72",    # FIPS code for Puerto Rico
  year = 2020,
  geometry = TRUE
)
```

```
## Getting data from the 2020 decennial Census
```

```
## Downloading feature geometry from the Census website. To cache shapefiles for use in future sessions
```

```
## Using the PL 94-171 Redistricting Data Summary File
```

```
## |
```

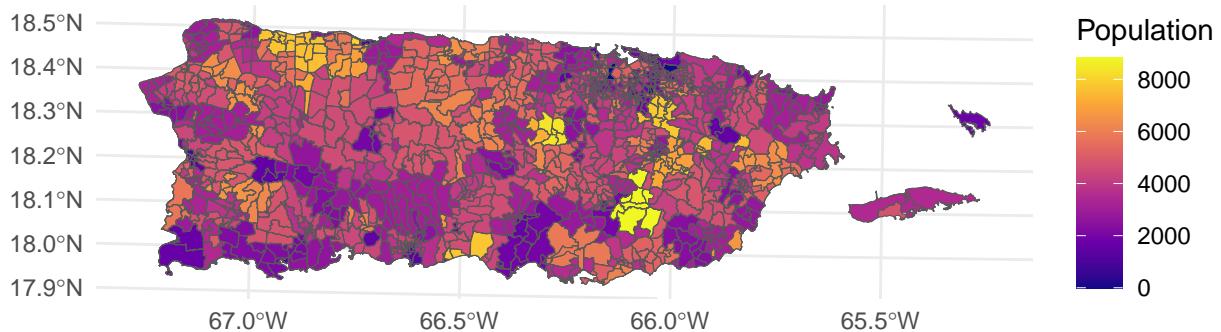
```
## Note: 2020 decennial Census data use differential privacy, a technique that
## introduces errors into data to preserve respondent confidentiality.
## i Small counts should be interpreted with caution.
## i See https://www.census.gov/library/fact-sheets/2021/protecting-the-confidentiality-of-the-2020-cen
## This message is displayed once per session.
```

```

pr_population <- st_transform(pr_population, crs = "EPSG:3920")
pr_unidades <- pr_unidades |>
  st_join(pr_population, by = c("GEOID" = "GEOID")) # Asumiendo que comparten GEOID en común
pr_unidades|>ggplot() +
  geom_sf(aes(fill = value)) + # `value` is the population count
  scale_fill_viridis_c(option = "plasma", name = "Population") +
  theme_minimal() +
  labs(title = "Población de Puerto Rico por tracto censal (2020)")

```

Población de Puerto Rico por tracto censal (2020)



```
pr_unidades
```

```

## Simple feature collection with 5669 features and 10 fields
## Geometry type: MULTIPOLYGON
## Dimension: XY
## Bounding box: xmin: 48384.55 ymin: 1985190 xmax: 262743.8 ymax: 2052288
## Projected CRS: Puerto Rico / UTM zone 20N
## First 10 features:
##           Municipio Precinto Unidad Grupo DS DR      GEOID
## 1   Barranquitas      70      5    3  6 26 72019952500
## 1.1 Barranquitas     70      5    3  6 26 72045952100
## 1.2 Barranquitas     70      5    3  6 26 72019952201
## 1.3 Barranquitas     70      5    3  6 26 72045951800
## 1.4 Barranquitas     70      5    3  6 26 72019952400
## 1.5 Barranquitas     70      5    3  6 26 72009250100
## 1.6 Barranquitas     70      5    3  6 26 72019952302

```

```

## 1.7 Barranquitas      70      5      3 6 26 72019952202
## 2        Cabo Rojo     46      2      3 4 20 72023830400
## 2.1      Cabo Rojo     46      2      3 4 20 72023830102
##
##                                     NAME variable value
## 1      Census Tract 9525, Barranquitas Municipio, Puerto Rico P1_001N 7271
## 1.1      Census Tract 9521, Comerío Municipio, Puerto Rico P1_001N 4726
## 1.2 Census Tract 9522.01, Barranquitas Municipio, Puerto Rico P1_001N 4215
## 1.3      Census Tract 9518, Comerío Municipio, Puerto Rico P1_001N 5211
## 1.4      Census Tract 9524, Barranquitas Municipio, Puerto Rico P1_001N 2038
## 1.5      Census Tract 2501, Aibonito Municipio, Puerto Rico P1_001N 5597
## 1.6 Census Tract 9523.02, Barranquitas Municipio, Puerto Rico P1_001N 6319
## 1.7 Census Tract 9522.02, Barranquitas Municipio, Puerto Rico P1_001N 5369
## 2      Census Tract 8304, Cabo Rojo Municipio, Puerto Rico P1_001N 4268
## 2.1      Census Tract 8301.02, Cabo Rojo Municipio, Puerto Rico P1_001N 4709
##
##           geometry
## 1  MULTIPOLYGON (((150446.9 20...
## 1.1 MULTIPOLYGON (((150446.9 20...
## 1.2 MULTIPOLYGON (((150446.9 20...
## 1.3 MULTIPOLYGON (((150446.9 20...
## 1.4 MULTIPOLYGON (((150446.9 20...
## 1.5 MULTIPOLYGON (((150446.9 20...
## 1.6 MULTIPOLYGON (((150446.9 20...
## 1.7 MULTIPOLYGON (((150446.9 20...
## 2  MULTIPOLYGON (((60772.66 20...
## 2.1 MULTIPOLYGON (((60772.66 20...

```

Podré llevarme estos datos para otros análisis en el futuro (o con datos más útiles).

En este bloque, generamos un mapa que muestra la **población por tracto censal** en Puerto Rico. Cada trácto está coloreado según su población, lo que nos permite visualizar cómo se distribuye la población a lo largo de la isla.

- **geom_sf(aes(fill = value))**: Aquí, visualizamos los tráctos censales y asignamos un color a cada uno de acuerdo con su población (**value**).
- **scale_fill_viridis_c()**: Utilizamos una escala de color continua, con la opción **plasma**, para mejorar la legibilidad de los datos.
- **labs()**: Añadimos un título al gráfico y una leyenda clara para que el lector entienda fácilmente la visualización.

Recapitulando

Qué aprendimos en este taller inicial

Hoy aprendimos varias cosas en R:

- Aprendimos principios de visualización
- Lenguaje de gramática de gráficos
- Uso de capas para añadir complejidad visual a los datos
- Gráficas con datos complejos
- Uso de **tidycensus** para descargar datos censales
- Gráficos con mapas

Continuamos el próximo miércoles

En los talleres que vienen continuaremos ahondando en operaciones estadísticas, así como estadística inferencial, y finalmente gráficos que ayuden a entender y diagnosticar los modelos estadísticos que usaremos. Finalmente entraremos en análisis de redes y si el tiempo permite, el último miércoles, también trabajaremos análisis de datos textual. Gracias por asistir hoy.