Herramientas Computacionales, Algoritmos y Machine Learning (HCML)

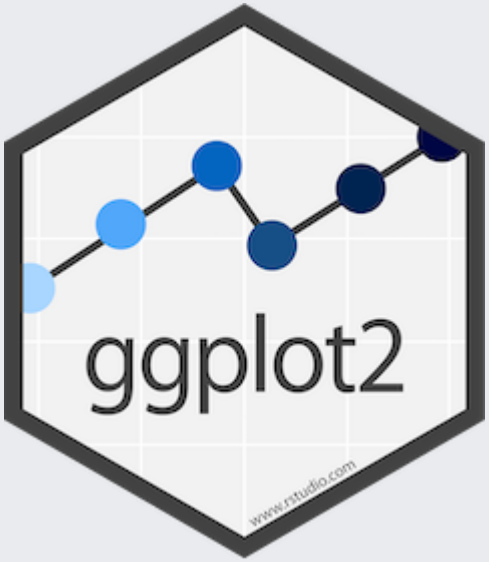
Clase 9: Visualización de datos con `ggplot2` - Parte II

Constanza Prado - Alex Antequeda - Diego Muñoz

Clase 9: Visualización de datos con `ggplot2` - Parte II

Gráficos con ggplot2 II

- Bases de datos a usar
- Graficos de barras y de tortas
- Añadir lineas a gráficos
- Agregar texto a gráficos
- Función `ggpairs`
- Graficos Interactivos (`plotly`)
- Gráficos Animados (`gganimate`)



Actividad

Material complementario

Bases de datos a usar

En esta clase, se usarán distintas bases de datos precargadas de la librería `datos`. Principalmente, se usarán las bases `paises` (conocida en inglés como `gapminder`) y `mtautos` (conocida en inglés como `mtcars`).

Además, se creará dos sub bases para trabajar de manera más acotada:

- **base_chile**: Base con los datos de Chile, sacada desde `paises`.
- **base_vecinos**: Base con los datos de Chile y sus países vecinos, sacada desde `paises`.
- **america07**: Base con la información del continente americano en el año 2007.

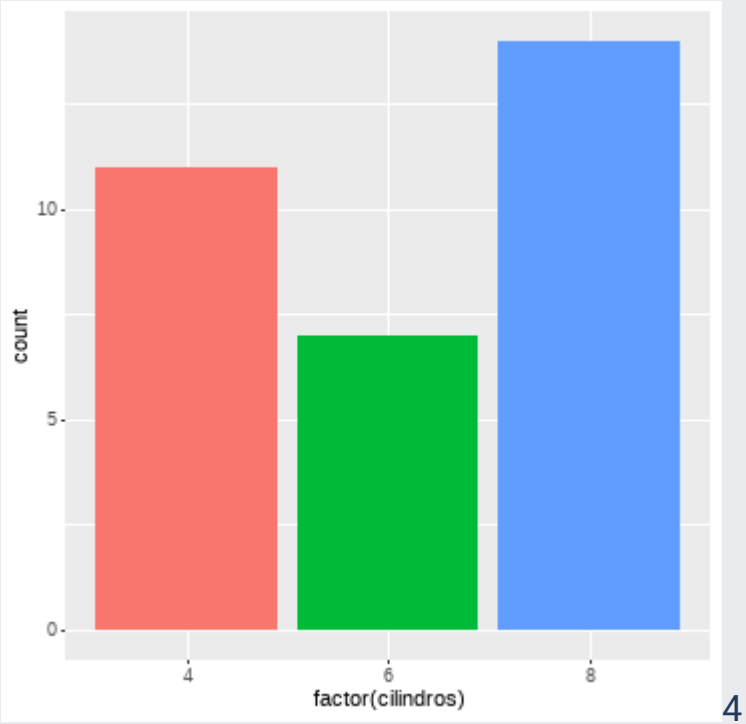
```
library(tidyverse)
library(datos)

base_chile = paises %>%
  filter(pais=="Chile")

base_vecinos = paises %>%
  filter(pais %in% c("Chile","Perú","Bolivia","Argentina"))

america07 = paises %>%
  filter(anio==2007, continente == "Américas")
```

Gráfico de barra (geom_bar())

En ggplot2 los gráficos de barras se realizan con el comando geom_bar(), indicando en el argumento aes() desde cual variable queremos discriminar. A continuación usaremos la base mtautos de la librería datos:

```
ggplot(data = mtautos, aes(x = factor(cilindros))) +
  geom_bar(aes(fill = factor(cilindros))) +
  theme(legend.position = 'none')
```

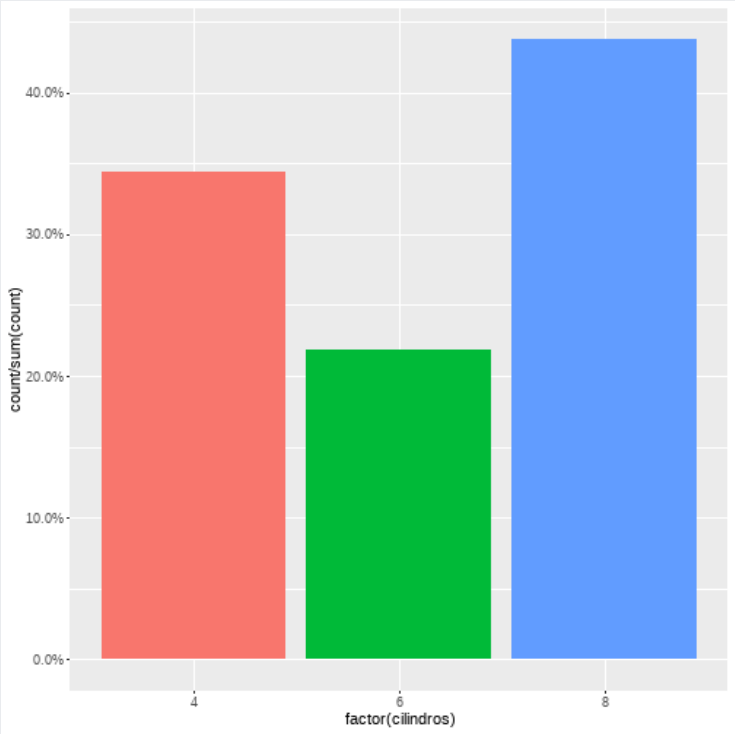
- Por defecto el gráfico de barras grafica en el eje y da cuenta de las observaciones en cada grupo. Esto puede modificarse dentro del comando aes() dentro del comando geom_bar().



```
ggplot(data = mtautos, aes(x = factor(cilindros) )) +
  geom_bar(aes(y = ..count../sum(..count..),
               fill = factor(cilindros))) +
  scale_y_continuous(labels = scales::percent) +
  theme(legend.position = 'none')
```

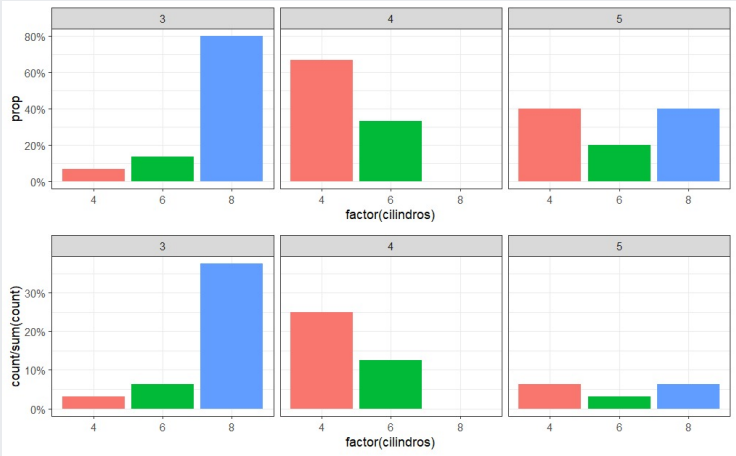
- `y=..count../sum(..count..)`dentro del argumento `aes()` permite graficar un gráfico de proporciones. La sentencia anterior calcula en cada grupo la cuenta y lo divide por el total acumulado.

- La función `scale_y_continuous` permite modificar el eje y del gráfico. En este caso el argumento interior transforma el eje en porcentajes, esto usando la función `percent` de la librería `scales`.



```
ggplot(data = mtautos, aes(x = factor(cilindros),
                           group = factor(cambios))) +
  geom_bar(aes(y = ..prop..,
               fill = factor(..x..))) +
  facet_grid(~cambios) +
  scale_y_continuous(labels = scales::percent)

ggplot(data = mtautos, aes(x = factor(cilindros))) +
  geom_bar(aes(y = ..count../sum(..count..),
               fill = factor(..x..))) +
  facet_grid(~cambios) +
  scale_y_continuous(labels = scales::percent)
```

Al realizar las interacciones entre los factores, se debe tener en consideración el objetivo de este, ya que se pueden obtener resultados diferentes, donde se considera para el cálculo de las métricas solo la información del nivel `gráfico 1`, así como, métricas considerando la información global `gráfico 2`.



Gráficos de torta

En `ggplot2` no hay una función **`geom_*`** que nos genere directamente un gráfico de torta (pie chart), pero existe una manera de generarlo a partir de un gráfico de barras (`geom_bar()`) de la siguiente forma:

1.- Generar una base de datos con las frecuencias a graficar.

Usaremos el comando `table()` para obtener las frecuencias de los cilindros de los automóviles de la base `mtautos` de la librería `datos`. Luego la tranformamos a una base de datos con el comando `data.frame()`.

```
df_cilindros = with(mtautos, table(cilindros)) %>%
  data.frame()

df_cilindros
```

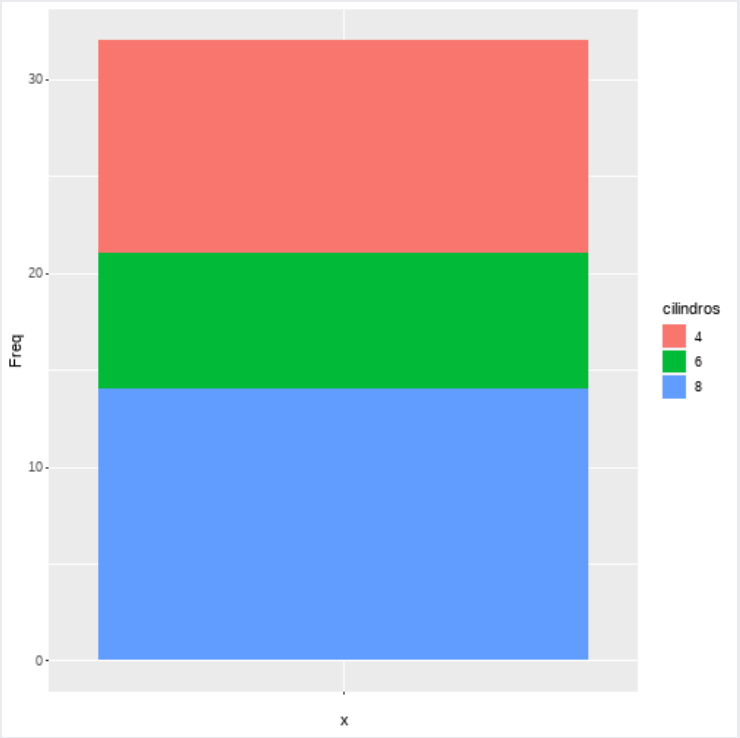
```
##   cilindros Freq
## 1         4   11
## 2         6    7
## 3         8   14
```

2.- Generamos un barplot de una barra.

Usamos la base creada para graficar un barplot. Usaremos la variable `cilindros` para pintar las barras y la variable `Freq` para determinar el tamaño de cada una.

```
plot_barra = ggplot(df_cilindros, aes(x = "", y = Freq,
                                      fill = cilindros)) +
  geom_bar(stat = 'identity', width = 1)
```

- `stat = 'identity'` permite graficar los datos como aparecen en la base de datos. En este caso es necesario dado que los datos ya están agrupados.
- `width = 1` permite graficar el barplot agrupado en una barra.

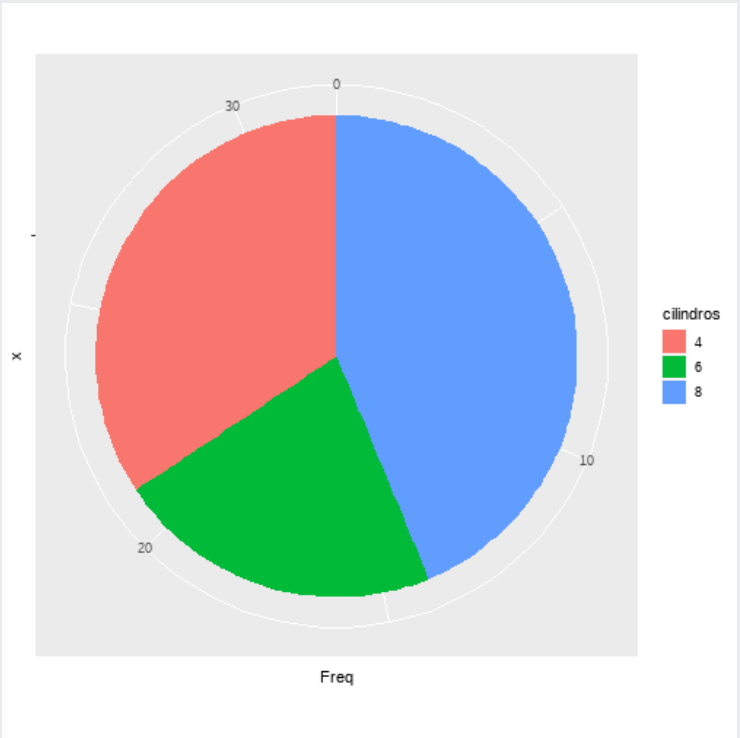


3.- Usamos el comando `coord_polar()` para definir un gráfico de tortas.

El comando `coord_polar()` nos permite tranformar un eje de un gráfico de `ggplot2` a coordenadas polares. En este caso lo usaremos para tranformar el gráfico `plot_barra` a un gráfico de torta:

```
plot_torta1 = plot_barra + coord_polar("y", start = 0)
```

- El argumento **"y"** indica cual eje queremos tomar como referencia. En este caso es el eje y, dado que es el eje por el cual se graficó la variable `Freq`.
- El argumento **start = 0** define el punto dentro de las coordenadas polares en donde inicia el gráfico.

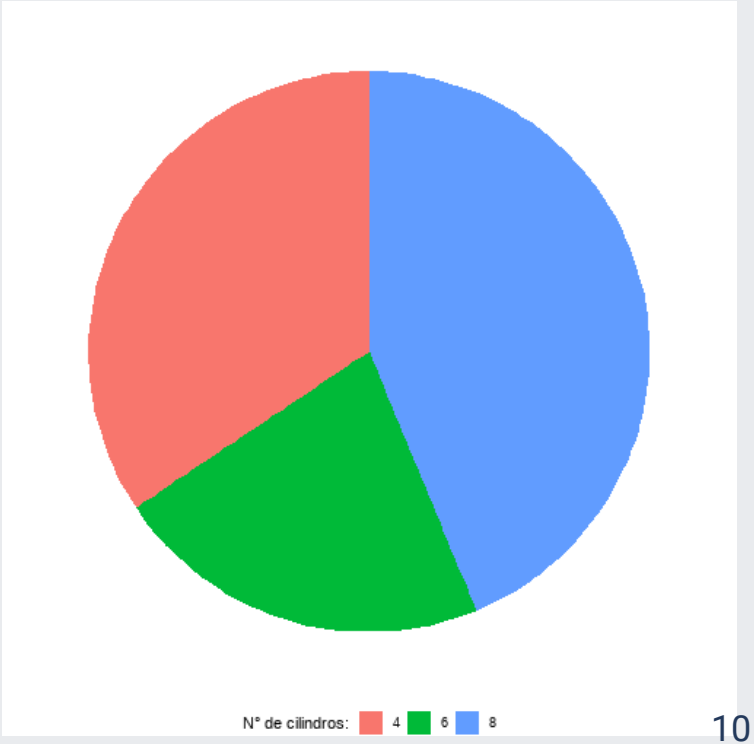


4.- Modificaciones visuales

En la siguiente slide realizamos algunas modificaciones visuales para mejorar el gráfico:

```
plot_torta2 = plot_torta1 +
  theme_void() +
  theme(legend.position = "bottom") +
  labs(fill = "N° de cilindros:")
```

- `theme_void` elimina la mayoría de los objetos visuales del gráfico, incluyendo fondo y ejes de gráficos.
- En `theme` modificamos la leyenda, ubicándola en la parte inferior del gráfico.
- En `labs` modificamos el título de la leyenda.



Líneas en gráficos

Existen distintos comandos que nos permiten añadir lineas o segmentos a nuestros gráficos de `ggplot2`. A continuación veremos con detención algunos de ellos:

Añadir líneas horizontales o verticales

El comando **`geom_hline()`** permite colocar una línea horizontal en el o los puntos indicados por el argumento `yintercept`.

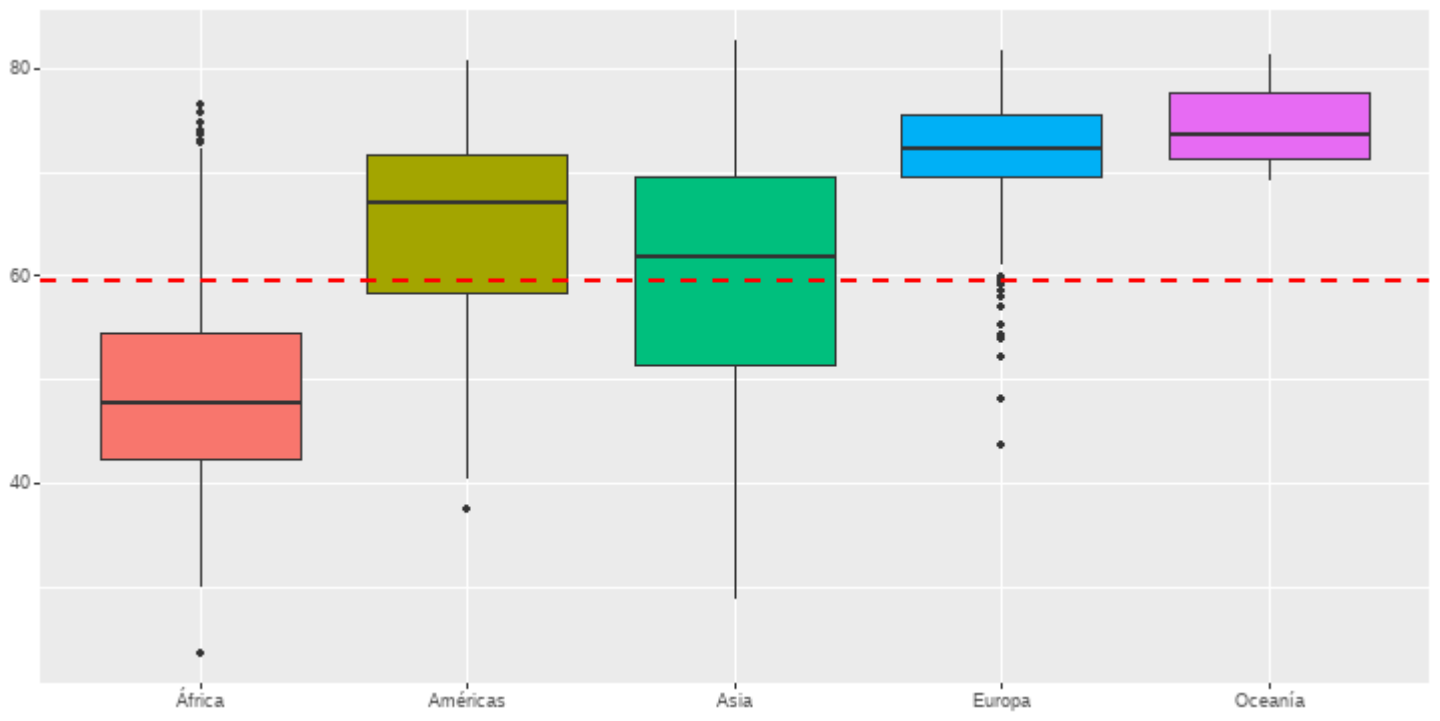
El comando **`geom_vline()`** permite colocar una línea vertical en el o los puntos indicados en el argumento `xintercept`.

Estas funciones tienen los siguientes argumentos en común:

- **`linetype`**: Tipo de línea (`'solid'`,`'dotted'`, `'dashed'`, `'longdash'`, etc).
- **`size`**: Grosor de la línea, se ingresa un número real.
- **`color`**: Color de la línea, puede ser un color de R o un código hexadecimal.

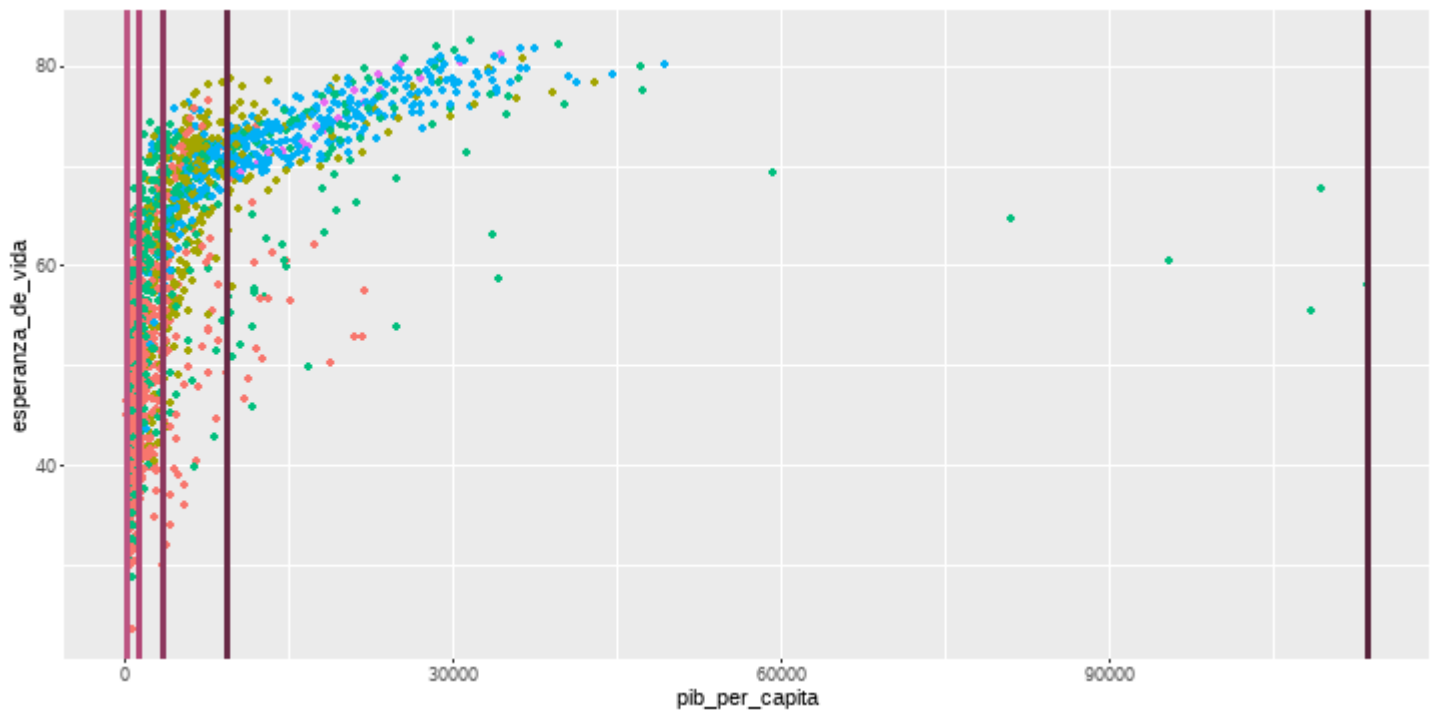
Ejemplo geom_hline()

```
ggplot(data=paises, aes(x=continente, y=esperanza_de_vida))+
  geom_boxplot(aes(fill = continente)) +
  theme(legend.position = 'none', axis.title = element_blank()) +
  geom_hline(yintercept = mean(paises$esperanza_de_vida),
             linetype = 'dashed',
             color = "red",
             size = 1)
```



Ejemplo geom_vline()

```
ggplot(data=paises, aes(x=pib_per_capita, y=esperanza_de_vida))+
  geom_point(aes(color = continente)) +
  theme(legend.position = 'none') +
  geom_vline(xintercept = quantile(paises$pib_per_capita),
             size = 1.2,
             color =
               c("#bc5482","#b14575","#8c375d","#672945","#552139"))
```



Añadir rectas

El comando `geom_abline()` permite añadir todo tipo de rectas en un gráfico de **ggplot2**. Este recibe dos argumentos:

- `intercept`: Número indicando la constante de la recta a graficar.
- `slope`: Número indicando la pendiente de la recta a graficar.

Es decir, con el comando `geom_abline(intercept = c, slope = m)` al gráfico se agregará la recta $y = m \cdot x + c$, donde x e y corresponden a los ejes del gráfico.

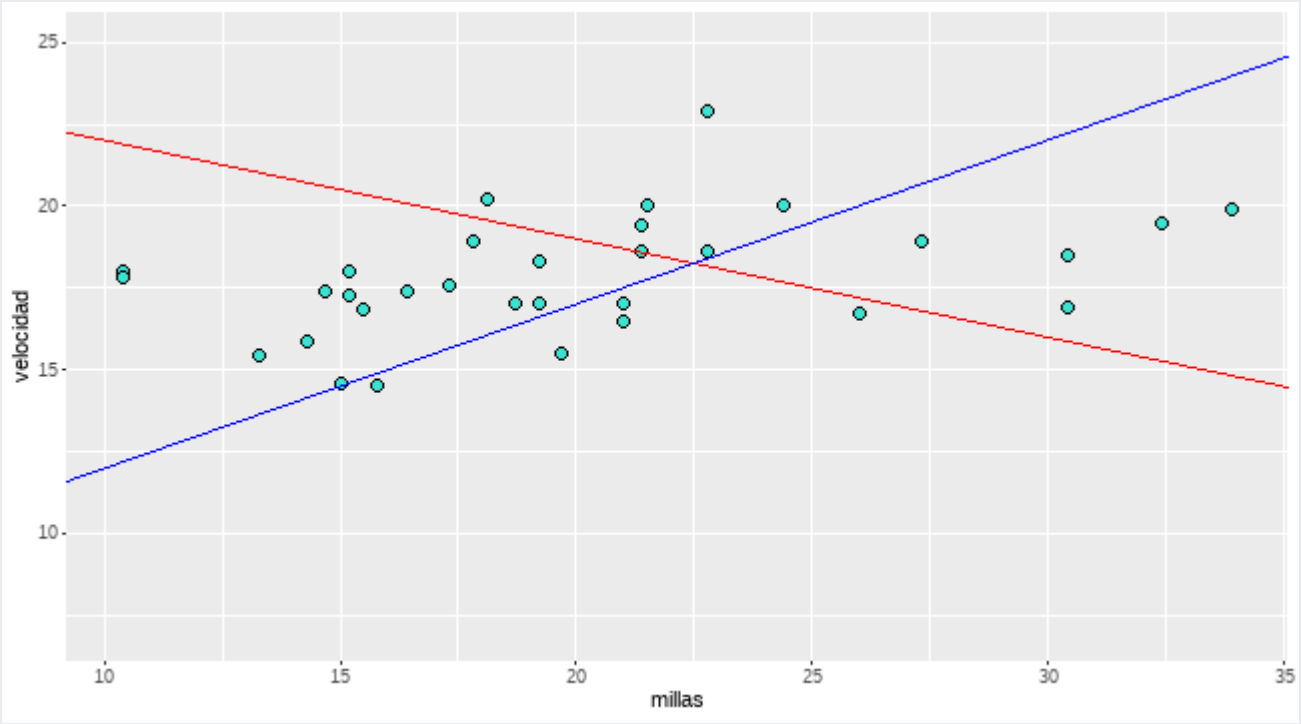
La función `geom_abline()` también contiene los argumentos `linetype`, `color` y `size`.

El comando **stat_smooth()** permite agregar ajustes de modelos a los datos. En este caso el modelo de Regresión Lineal nos entrega una recta de manera automática al añadir el argumento `method = "lm"`.

Por defecto este comando entrega bandas de confianza del ajuste de la regresión, dadas por el error estándar del ajuste. Estas pueden ser desactivadas con el argumento `se = FALSE`

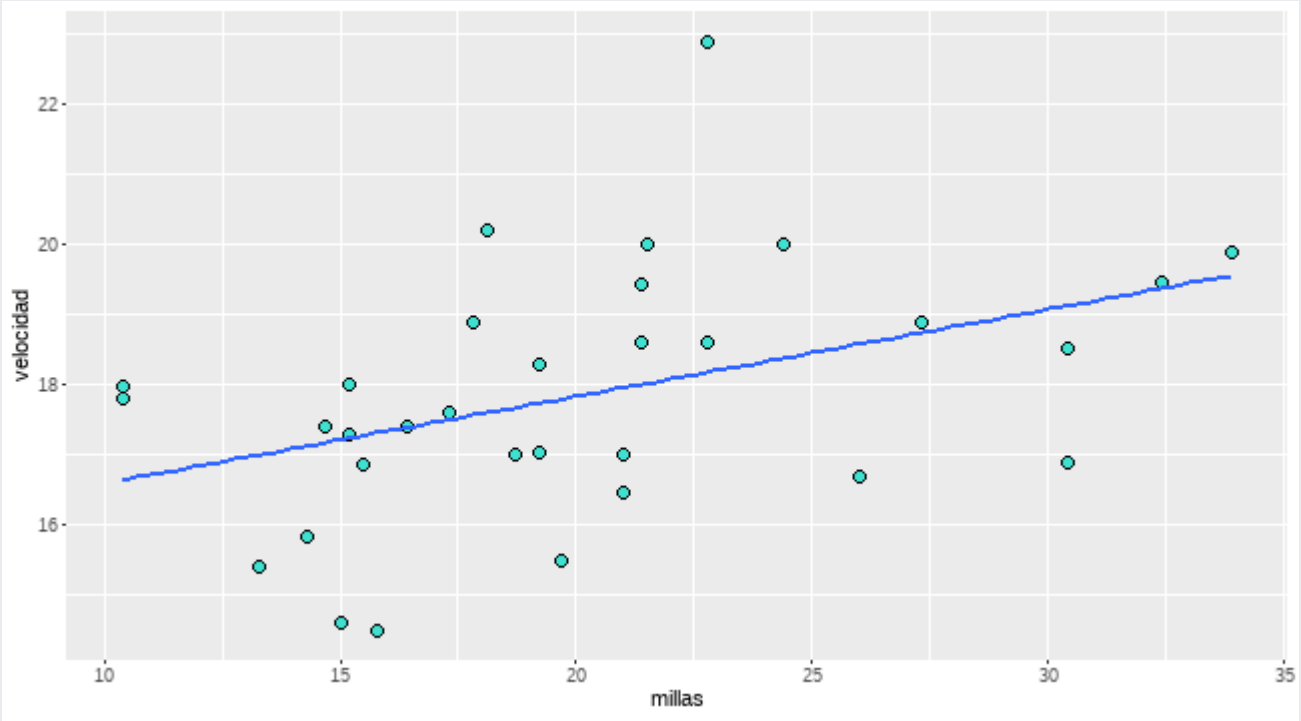
Rectas con geom_abline()

```
ggplot(data = mtautos, aes(x=millas, y=velocidad))+
  geom_point(size = 3, shape = 21, fill = "turquoise") +
  geom_abline(intercept = 25, slope = -0.3, color = "red") +
  geom_abline(intercept = 7, slope = 0.5, color = "blue")
```



Recta de regresión lineal con `stat_smooth()`

```
ggplot(data = mtautos, aes(x=millas, y=velocidad))+
  geom_point(size = 3, shape = 21, fill = "turquoise") +
  stat_smooth(method = "lm", se = F)
```



Función `ggpairs`

La función `ggpairs` de la libería `GGally` entrega un resumen gráfico completo de las variables de una base de datos, esto mediante una matriz de gráficos.

Por defecto los gráficos de `ggpairs` contiene:

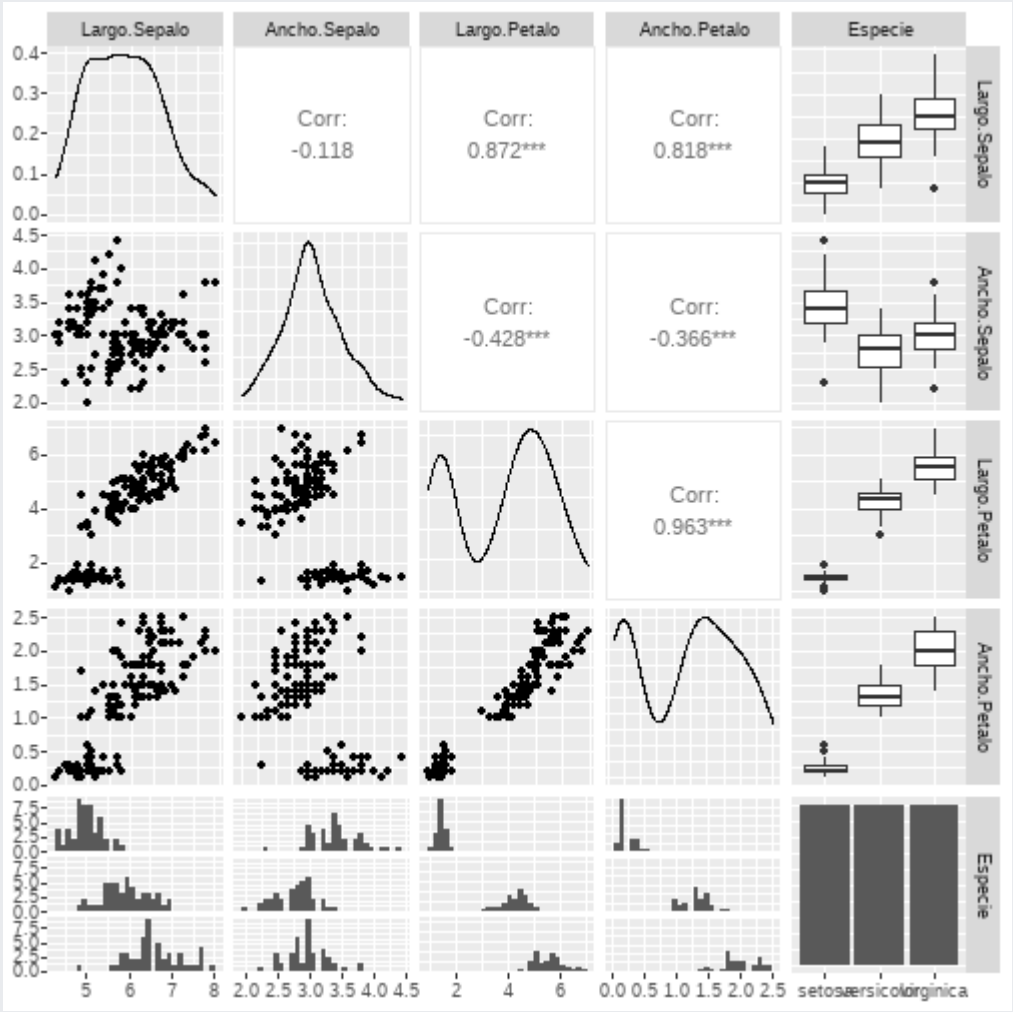
- **Gráfico de cada variable**: En la diagonal de la matriz se encuentra un gráfico de cada variable, este es un gráfico de densidad (variable continua) o un gráfico de barras (variable categórica).

- **Gráfico entre cada par de variables**: Fuera la diagonal se grafican todos los cruces de las variables de la base de datos. El gráfico en cuestión depende de los tipos de variables:

    - **Dos variables continuas**: Gráfico de dispersión y correlación.
    - **Categórica y continua**: Histograma y boxplot por grupo.
    - **Dos categóricas**: Gráficos de barra.

A continuación se encuentra un ejemplo de la función `ggpairs` usando la base de datos `flores` de la librería `datos`:

```
GGally::ggpairs(flores)
```



Añadir texto a los gráficos

Comando `geom_text`

El comando `geom_text` es la herramienta base utilizada para incluir texto en los gráficos de `ggplot2` esto mediante dos funciones:

- **`geom_text`**: Incluye texto a un gráfico de `ggplot2`.
- **`geom_label`**: Incluye un texto con un cuadro de fondo a un gráfico de `ggplot2`.

Librería `ggrepel`

La librería `ggrepel` contine funciones que permiten incluir texto a un gráfico de `ggplot2`. A diferencia de las funciones anteriores estas contienen funcionalidades que permiten que el texto no se superponga entre si o con elementos gráficos. Esta librería contiene principalmente las siguientes funciones:

- **`geom_text_repel`**: Incluye texto a un gráfico de `ggplot2`.
- **`geom_label_repel`**: Incluye un texto con un cuadro de fondo a un gráfico de `ggplot2`.

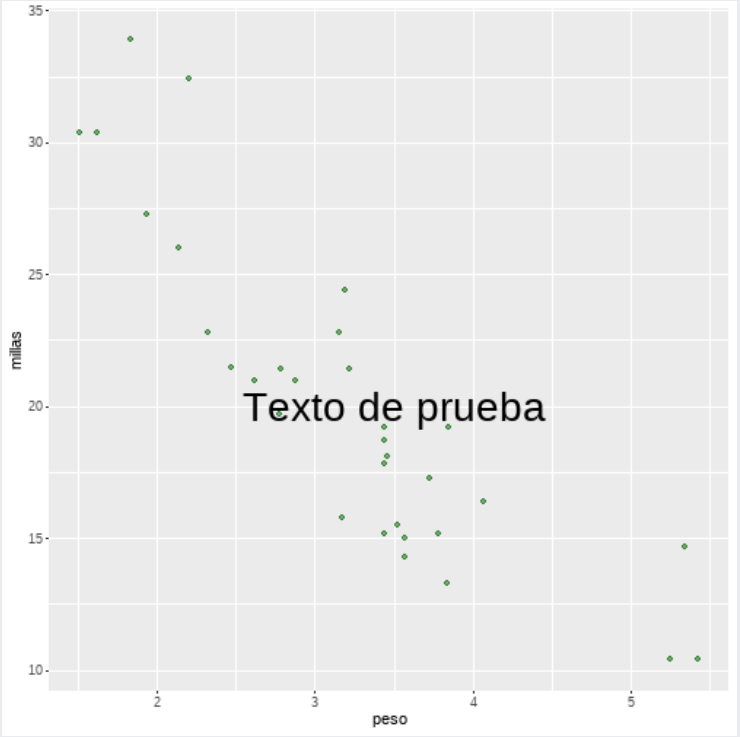
Tantos las funciones bases como las de `ggrepel` pueden agregar elementos de texto sueltos o variables de la base de datos dentro del comando `aes()`.

Ejemplo elementos únicos de texto

```
ggplot(data=mtautos, aes(x=peso, y=millas)) +
  geom_point(color = "darkgreen", alpha = 0.5) +
  geom_text(x = 3.5, y = 20, label = "Texto de prueba",
            color = "black",stat = "unique", size = 10)
```

Para ingresar texto de manera independiente en el gráfico se tienen que indicar los siguientes argumentos:

- **x,y**: Coordenadas del texto.
- **label**: Texto a ingresar.
- **color**: Color del texto.
- **size**: Tamaño del texto.
- **stat = "unique"**: Argumento para ingresar sólo una vez el texto.

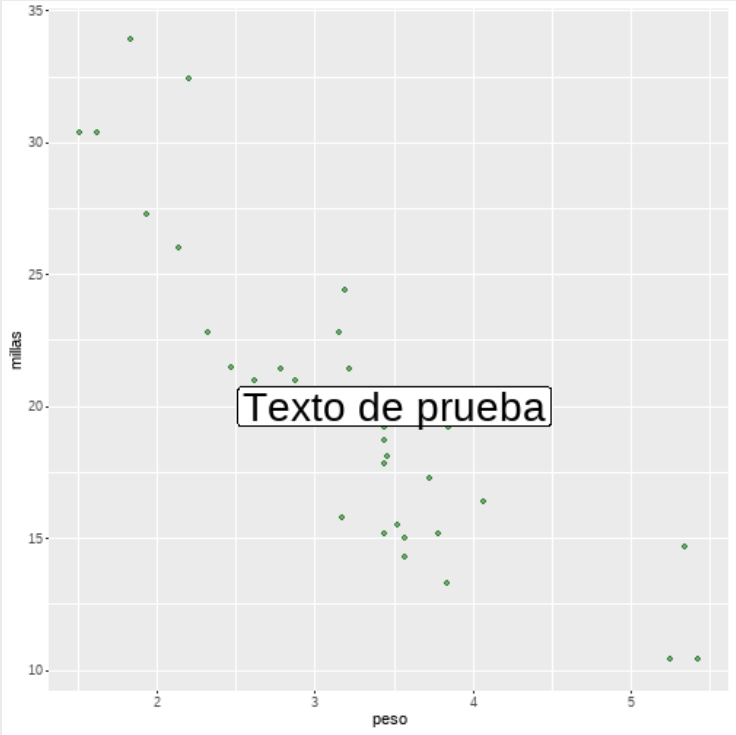


Ejemplo elementos únicos de texto

```
ggplot(data=mtautos, aes(x=peso, y=millas)) +
  geom_point(color = "darkgreen", alpha = 0.5) +
  geom_label(x = 3.5, y = 20, label = "Texto de prueba",
             color = "black",stat = "unique", size = 10)
```

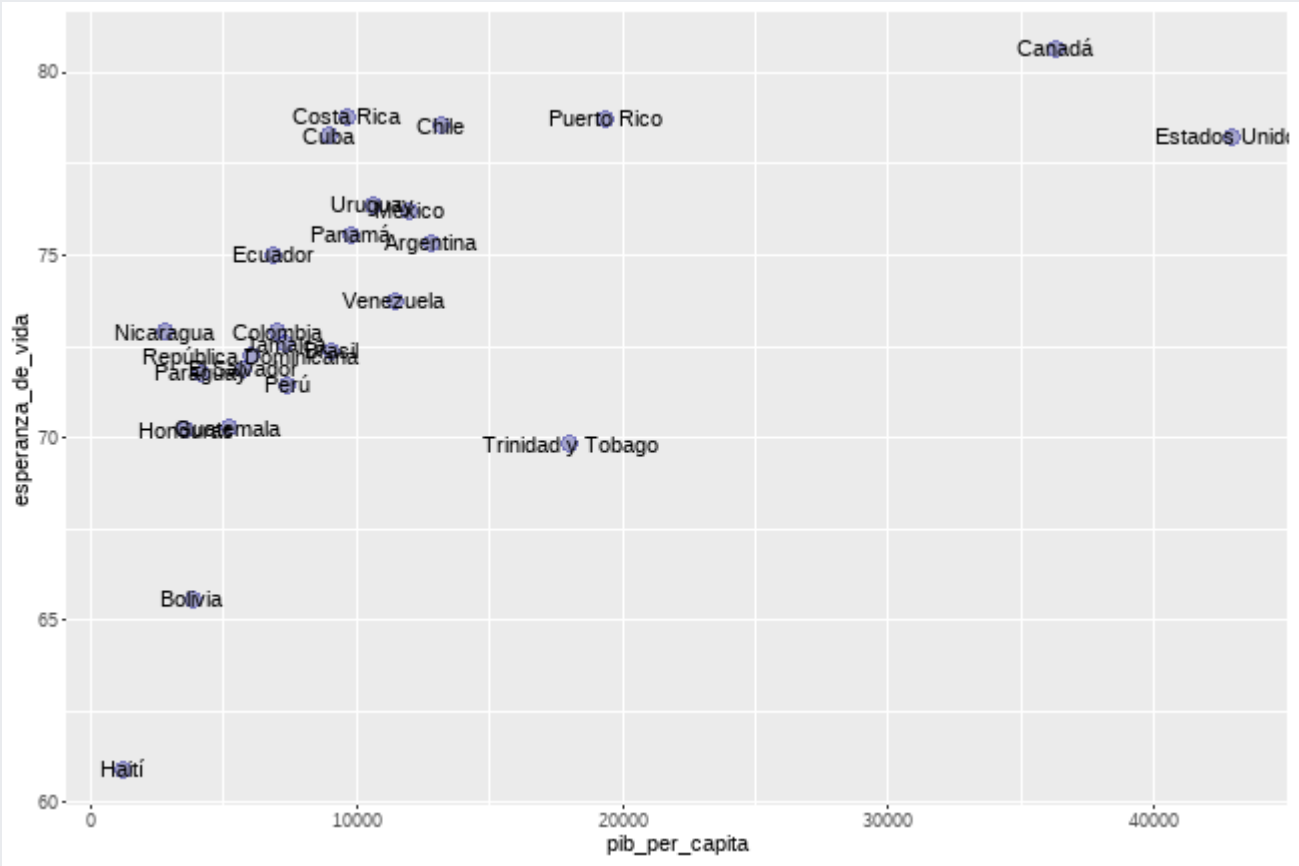
Para ingresar texto de manera independiente en el gráfico se tienen que indicar los siguientes argumentos:

- **x,y**: Coordenadas del texto.
- **label**: Texto a ingresar.
- **color**: Color del texto.
- **size**: Tamaño del texto.
- **stat = "unique"**: Argumento para ingresar sólo una vez el texto.



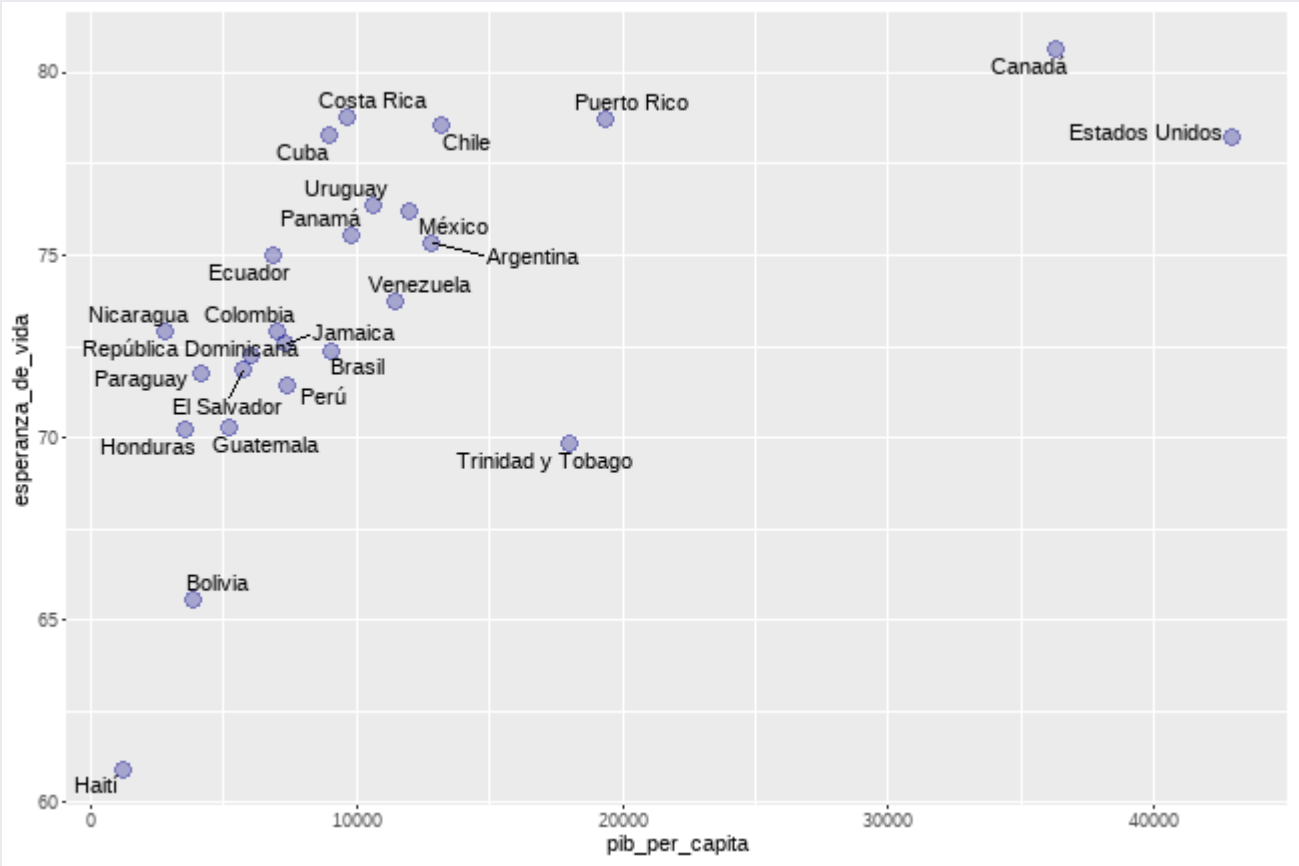
Ejemplo geom_text()

```
ggplot(data = america07, aes(x = pib_per_capita,
                             y = esperanza_de_vida)) +
  geom_point(size=4, alpha=0.3, col = "darkblue") +
  geom_text(aes(label = pais))
```



Ejemplo geom_text_repel()

```
ggplot(data = america07, aes(x = pib_per_capita,
                             y = esperanza_de_vida)) +
  geom_point(size=4, alpha=0.3, col = "darkblue") +
  ggrepel::geom_text_repel(aes(label = pais))
```



Librería `plotly`

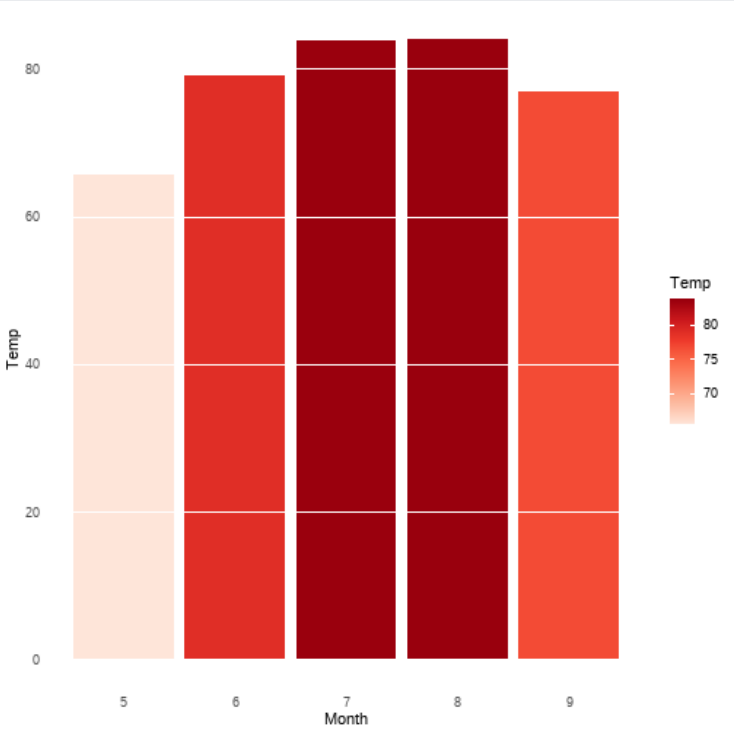
La librería `plotly` permite entregarle interactividad a los gráficos de `ggplot2`. Esto se hace con el comando `ggplotly`:

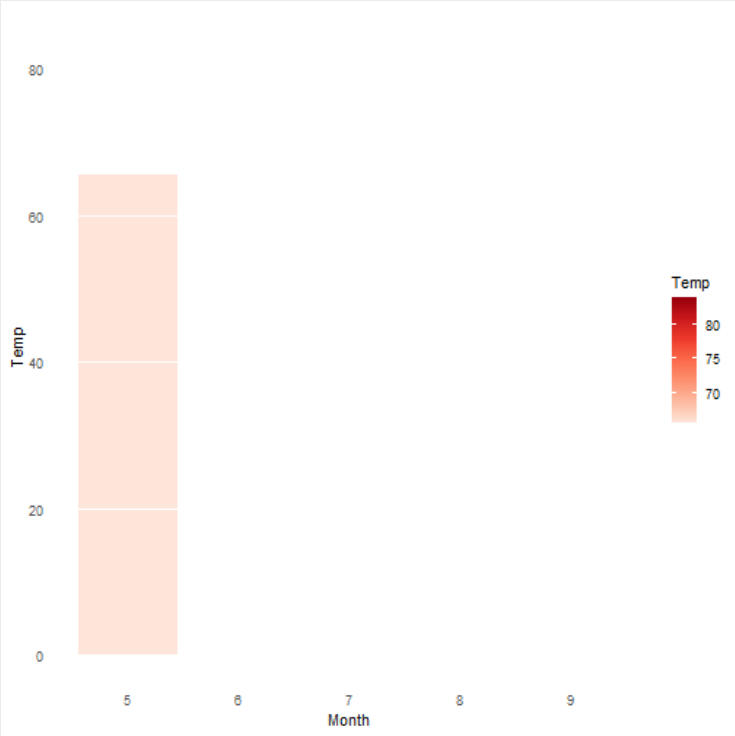
```
plotly::ggplotly( grafico_ggplot2 )
```

Por defecto el comando `ggplotly` recibe como argumento `ggplot2::last_plot()`, esto entrega sin necesidad de definir un objeto la versión interactiva del último gráfico de `ggplot2` ejecutado en la sesión actual.

Gráficos animados con `gganimate`

La librería `gganimate` permite animar nuestros gráficos de `ggplot2`, generando un archivo animado en formato `.gif`.





Ejemplo de https://www.datanovia.com/en/blog/gganimate-how-to-create-plots-with-beautiful-animation-in-r/

Gráficos animados con `gganimate`

Para que la animación funcione correactamente tenemos que tener las librerías `gganimate` y `gifski` instaladas en la sesión.

La animación se genera a través de comandos de **transición** que permiten determinar como y sobre que variable se animará el gráfico. Algunas de ellas son:

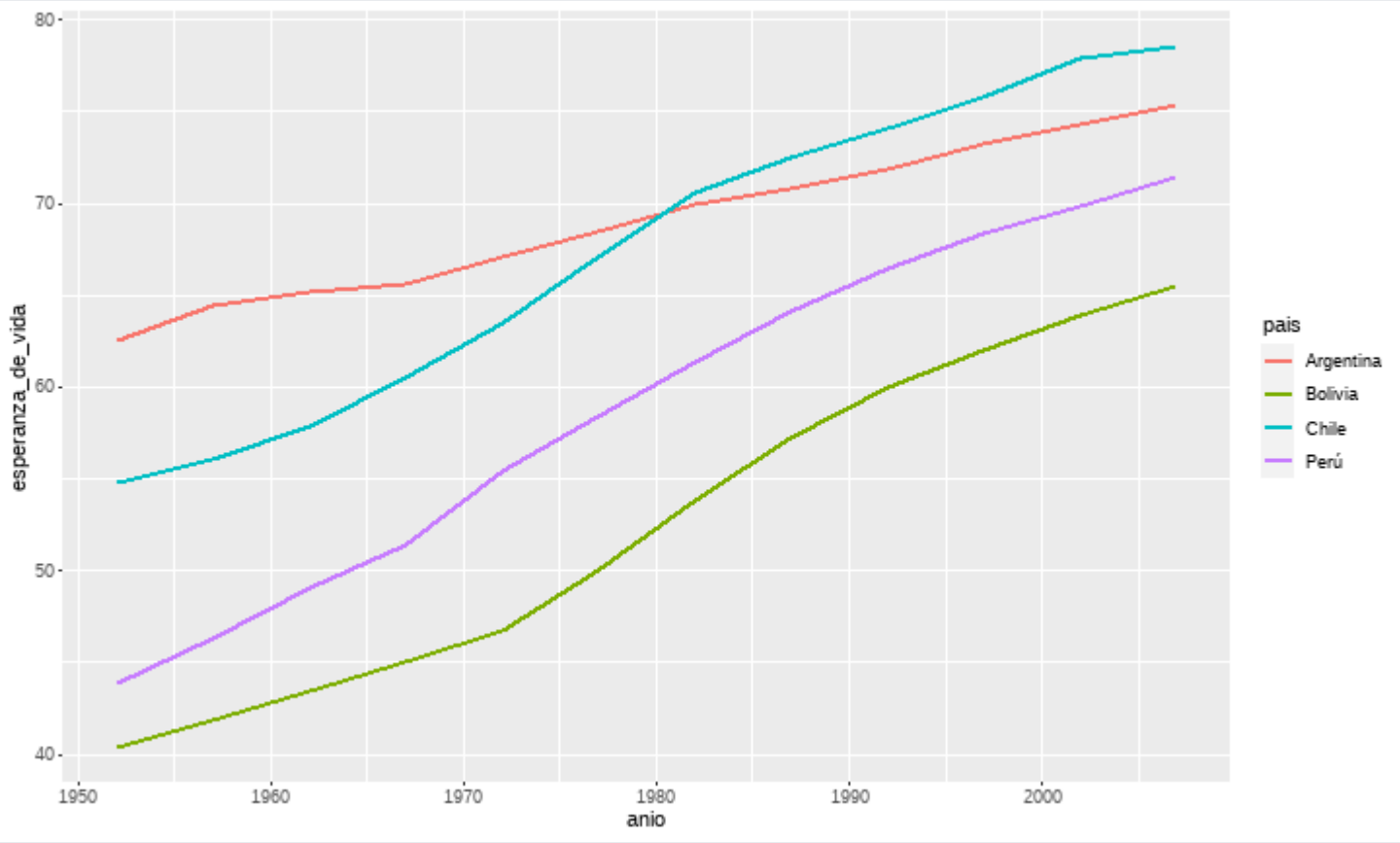
- `transition_reveal` : Los datos aparecen gradualmente basados en una variable de tiempo, este genera una transición suave al calcular los valores en los puntos intermedios.
- `transition_states` : Se dividen los datos en distintos estados, basándose en los niveles de una variable, despúes se genera un gráfico que transiciona entre estos niveles pausando un tiempo en cada estado.

La animación más simple se realiza con un gráfico de ggplot más un comando de transición dentro de la variable `animate()`, de la siguiente forma:

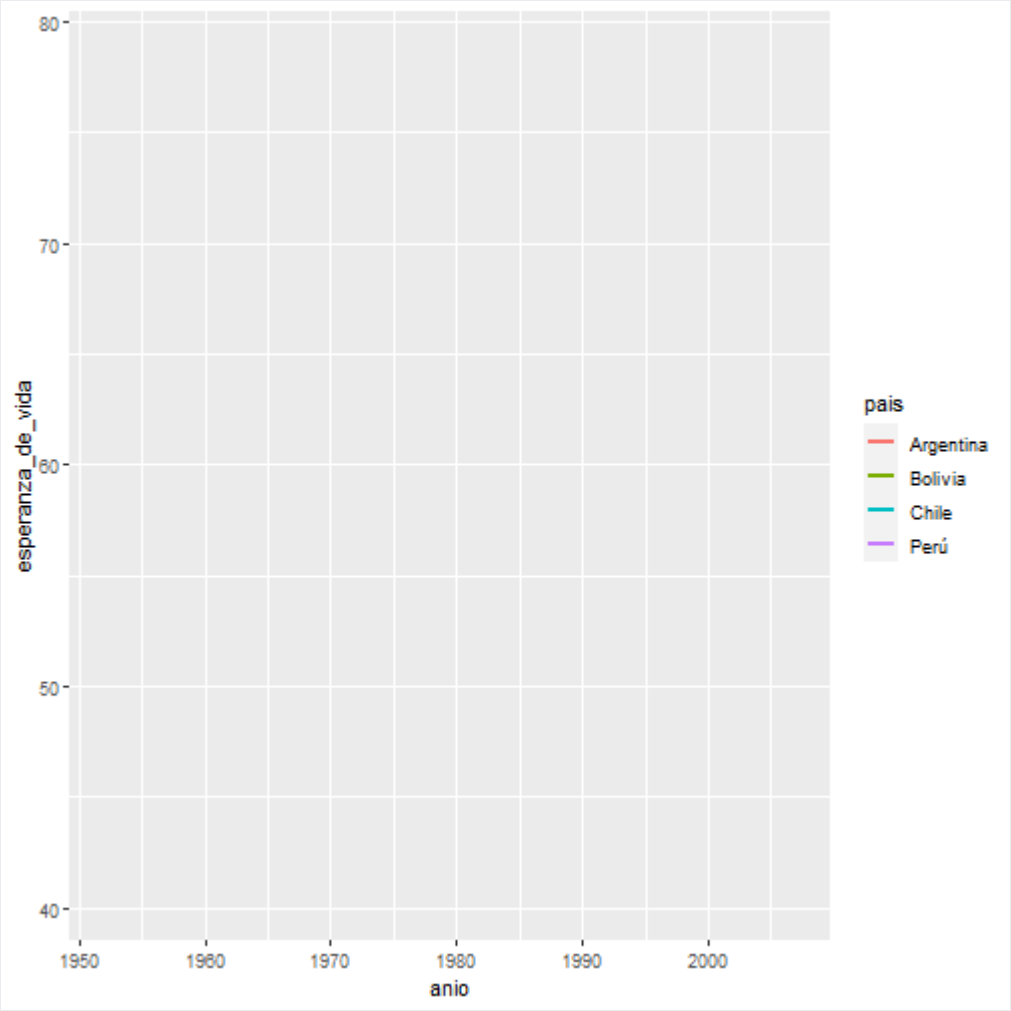
```
animate(grafico_ggplot + transition_*(variable))
```

Al igual que en las funciones de `ggplot2`, en `gganimate` existen funciones extras que permiten modificar la animación de las funciones.

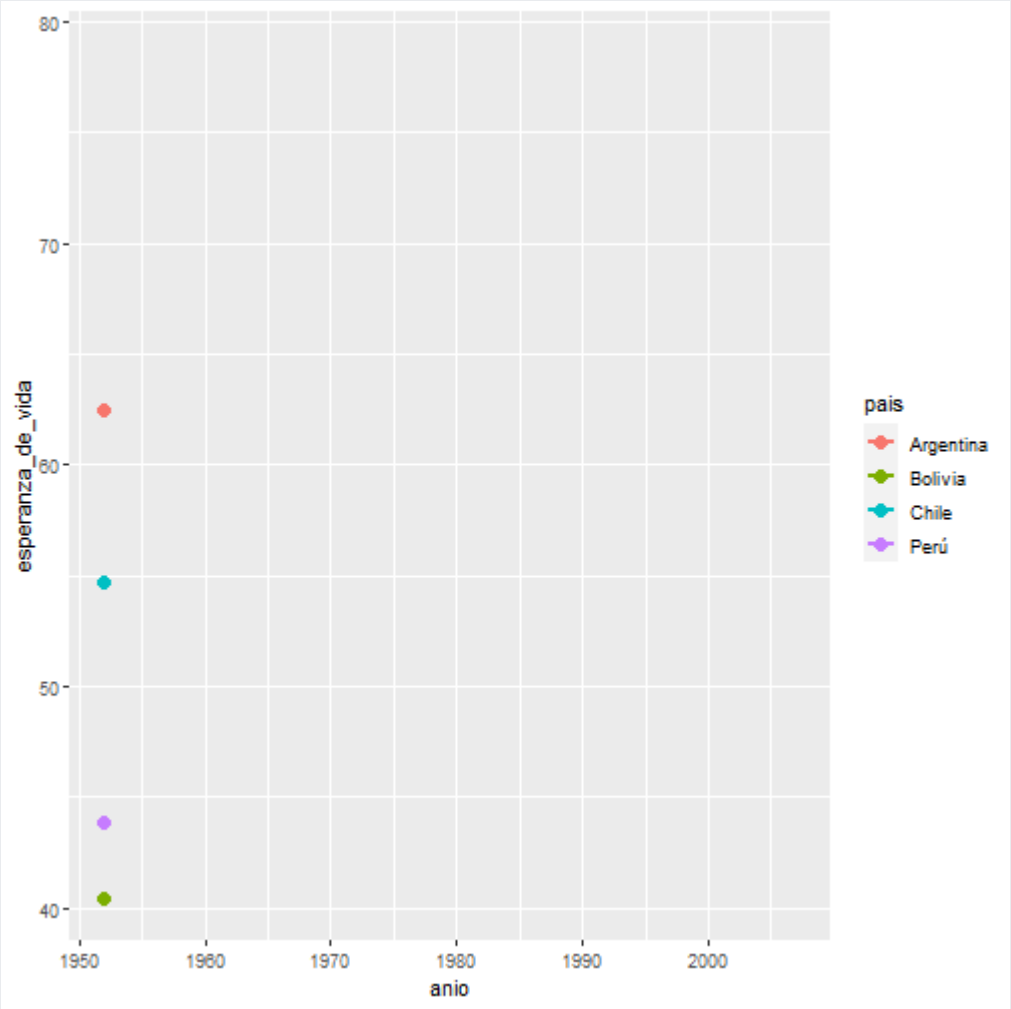
```
plot_1 = ggplot(data = base_vecinos,
                aes(x = anio, y = esperanza_de_vida, color = pais) ) +
   geom_line(size = 1)
plot_1
```



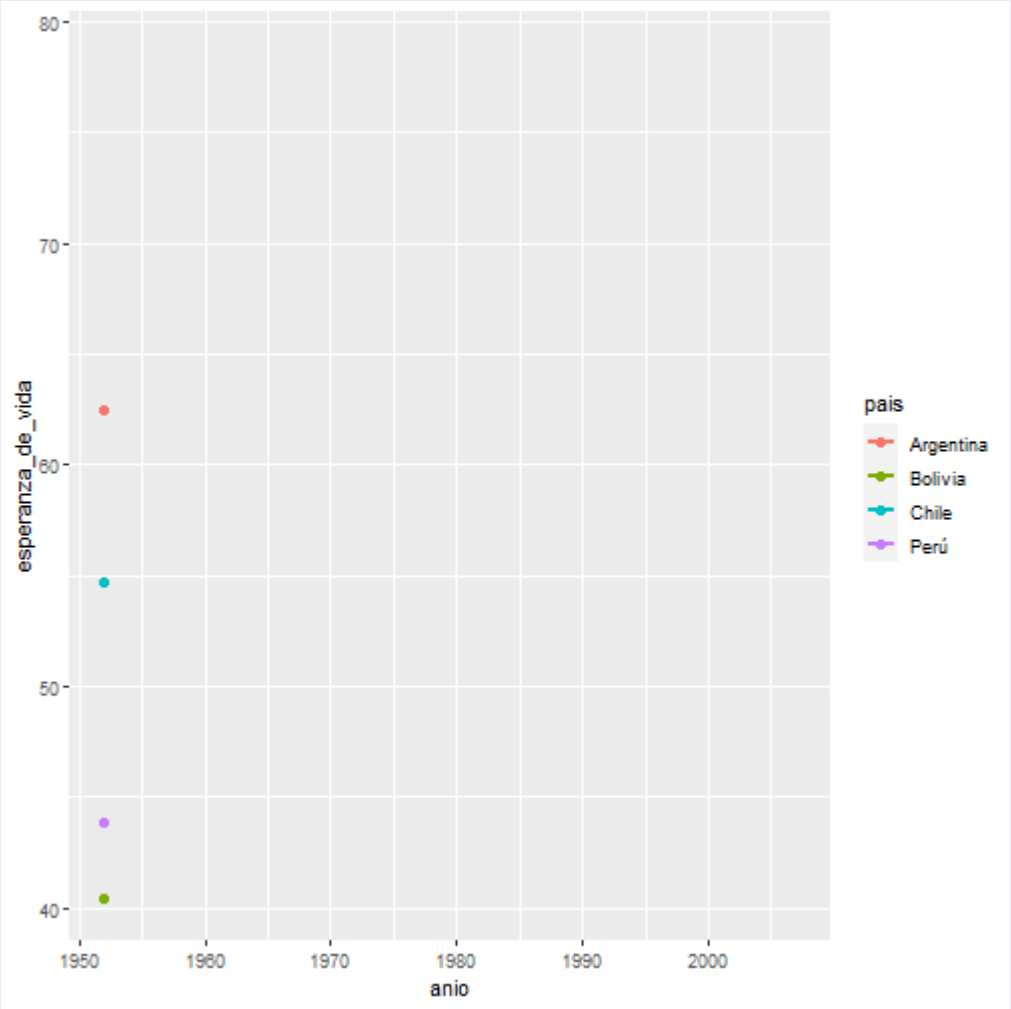
```
animate(plot_1 +
          transition_reveal(anio))
```



```
animate(plot_1 +
          geom_point(size=3) +
          transition_reveal(anio))
```



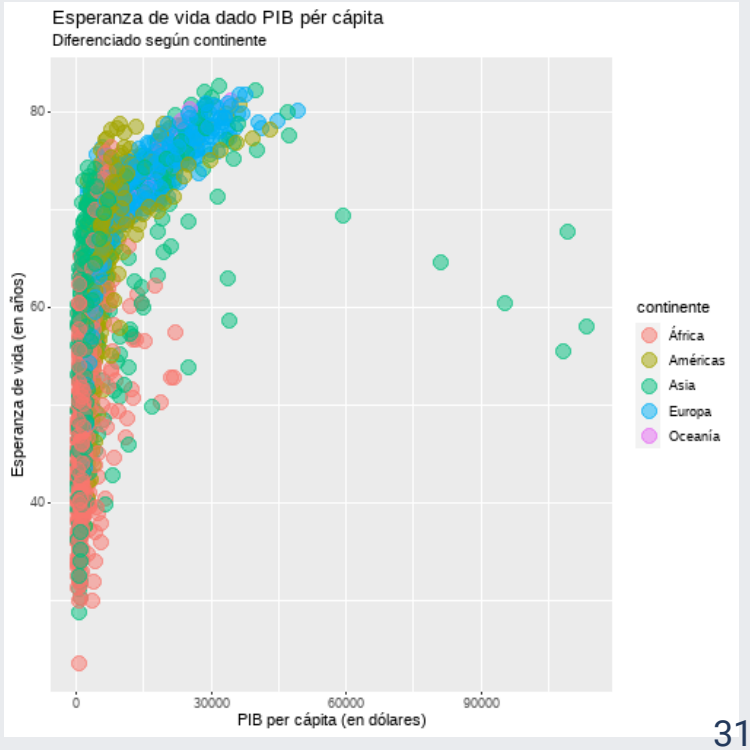
```
animate(plot_1 +
          geom_point(aes(group = seq_along(anio)), size = 2) +
          transition_reveal(anio) )
```



```
plot_3 = ggplot(paises, aes(x = pib_per_capita,
                            y = esperanza_de_vida,
                            color = continente)) +
  geom_point(size=5, alpha=0.5) +
  labs(x = "PIB per cápita (en dólares)",
       y = "Esperanza de vida (en años)",
       title = "Esperanza de vida dado PIB pér cápita",
       subtitle = "Diferenciado según continente")
plot_3
```

El estado actual de esta visualización no permite distinguir los puntos del continente de Oceanía.

Es de interés definir una animación en donde los puntos aparezcan agrupados por continente. Además, sería ideal que los puntos de los continentes anteriores no afecten la visibilidad de las nuevos puntos, esto permitiría ilustrar tanto los datos de los continentes y las diferencias entre estos.



```
animate(plot_3 +
          transition_states(continente,transition_length = 1,
                            state_length = 2) +
          shadow_mark(alpha=0.3, size=2) +
          enter_fade()+
          exit_shrink())
```

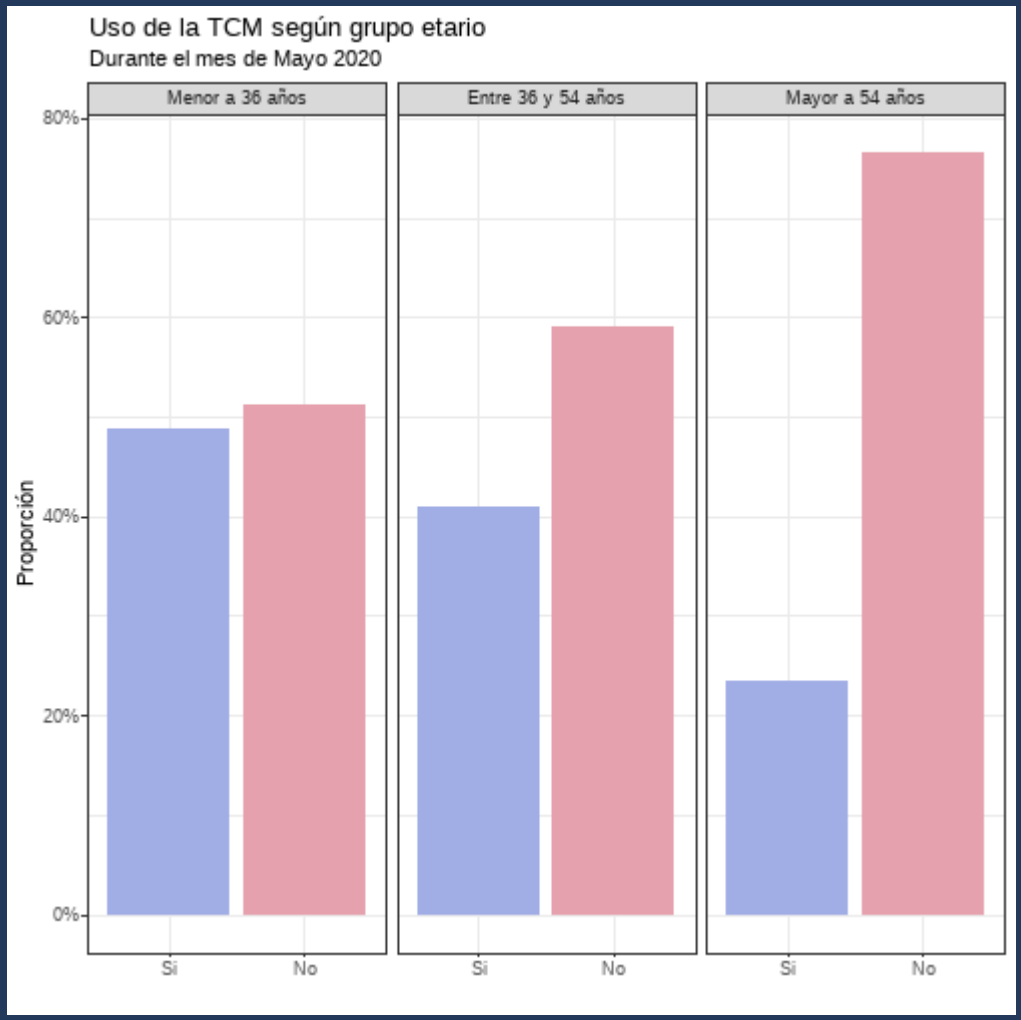
- En el comando **transition_states()** se define el tiempo de transición y el tiempo de permanencia en cada estado.
- El comando **shadow_mark()** deja un rastro de los estados anteriores, en él definimos el tamaño y la trasparencia con la que queremos dejar estos puntos.
- El comando **enter_fade()** entrega una animación de entrada a los nuevos estádos.
- El comando **exit_shrink()** entrega una animación de encogida de los puntos a la salida de los estados.

Actividad

1. La base de datos `TCM2020.xlxs` contiene datos recolectados en el año de Mayo 2020 sobre la Tarjeta Compra Más, una tarjeta de crédito de una casa comercial. Alguna de las variables contenidas en su base de datos son:

Nombre	Descripción
Cliente	Identificador del cliente
Edad	Edad en años
UsoMayo	Uso de la TCM en Mayo 2020. (1:Si, 0:No)
Uso2020	Uso de la TCM tarjeta en 2020 (1:Si, 0:No)

Considerando la base de datos anterior, replique el siguiente gráfico en R usando `ggplot2`:



Referencias y material complementario

- **Link**: Anotaciones de texto con `geom_text` (r-charts.com)
- **Link:** Ejemplos y funcionalidades `ggpairs` (r-graph-gallery.com)
- **Link:** Más funcionalidades ggploty.

Añadir lineas, segmentos, distribuciones

- **Link:** Líneas rectas, horizontales, verticales.
- **Link:** Líneas distribuciones.

¡Gracias!