

Introducción al análisis de datos con R

Dataviz e introducción a ggplot2

Manuel Mejías Leiva

Universidad de Valladolid | manuel.mejias@uva.es

5 - 9 junio de 2023

Primeros pasos con ggplot

Antes de comenzar: librerías y datos

El paquete **ggplot** viene integrado dentro del paquete **tidyverse**

```
library(tidyverse) #contiene ggplot
library(gapminder) #contiene la base de datos gapminder
```

Cargamos la base de datos de gapminder al entorno

```
gapminder
```

```
## # A tibble: 1,704 × 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>    <int> <dbl>    <int>    <dbl>
## 1 Afghanistan Asia      1952  28.8  8425333    779.
## 2 Afghanistan Asia      1957  30.3  9240934    821.
## 3 Afghanistan Asia      1962  32.0 10267083    853.
## 4 Afghanistan Asia      1967  34.0 11537966    836.
## 5 Afghanistan Asia      1972  36.1 13079460    740.
## 6 Afghanistan Asia      1977  38.4 14880372    786.
## 7 Afghanistan Asia      1982  39.9 12881816    978.
## 8 Afghanistan Asia      1987  40.8 13867957    852.
## 9 Afghanistan Asia      1992  41.7 16317921    649.
## 10 Afghanistan Asia      1997  41.8 22227415    635.
## # i 1,694 more rows
```

Antes de comenzar: librerías y datos

El **conjunto de datos** `gapminder`, del paquete homónimo, es un fichero con **datos de esperanzas de vida, poblaciones y renta per cápita** de distintos países en distintos momentos temporales.

```
glimpse(gapminder)
```

```
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ...
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ...
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ...
## $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8...
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12...
## $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ...
```

Características de una buena visualización

Según Alberto Cairo, una buena representación gráfica de los datos debe de tener las siguientes características:

- Contiene **información fiable**.
- El diseño se ha escogido para **destacar las pautas más relevantes**.
- La **presentación es atractiva**, pero las apariencias no pueden ser un sustituto de la **honestidad**, la **claridad** y la **profundidad**.
- Cuando resulta apropiado, está organizada de tal manera que **permita una cierta exploración de los datos**.

La gramática de los gráficos

Dataviz en R: **ggplot2**

La filosofía detrás de **{ggplot2}** es entender los **gráficos como parte del flujo** de trabajo, dotándoles de una **gramática**

El objetivo es empezar con un lienzo en blanco e ir **añadiendo capas a tu gráfico**. La ventaja de **{ggplot2}** es poder **mapear atributos estéticos** (color, forma, tamaño) de objetos geométricos (puntos, barras, líneas) en función de los datos.

La **documentación** del paquete puedes consultarla en <https://ggplot2-book.org/introduction.html>

Dataviz en R: **ggplot2**

Un gráfico se podrá componer de las siguientes **capas**

- **Datos (data)**
- **Mapeado (aesthetics)** de elementos estéticos: ejes, color, forma, tamaño, etc (en función de los datos)
- **Geometría (geom)**: puntos, líneas, barras, polígonos, etc.
- **Componer gráficas (facet)**: visualizar varias gráficas a la vez.
- **Transformaciones (stat)**: ordenar, resumir, agrupar, etc.
- **Coordenadas (coord)**: coordenadas cartesianas, polares, grids, etc.
- **Temas (theme)**: fuente, tamaño de letra, subtítulos, captions, leyenda, ejes, etc.

A continuación, se muestran muestra dos formas de definir un gráfico con ggplot2 en R:

```
#1ª forma sin usar el pipe  
ggplot(data = , aes(x =, y =, ...))+  
  geom_*()  
  
#2ª forma usando el pipe  
data %>%  
  ggplot(aes(x =, y =, ...))+  
  geom_*()
```


Primer intento: scatter plot

Imagina que queremos dibujar un **scatter plot** o **diagrama de (dispersión) de puntos**.

El fichero consta de 1704 registros y 6 variables: **country**, **continent**, **year**, **lifeExp** (esperanza de vida), **pop** (población) y **gdpPercap** (renta per cápita).

```
glimpse(gapminder)
```

```
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ...
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ...
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ...
## $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8...
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12...
## $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ...
```

Para empezar con algo sencillo **filtraremos solo los datos de 1997**

```
gapminder_1997 <- gapminder %>% filter(year == 1997)
gapminder_1997
```

```
## # A tibble: 142 × 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>      <int>  <dbl>    <int>    <dbl>
```

Primer intento: scatter plot

Vamos a realizar un **diagrama de puntos**:

- **Eje X**: renta per cápita (variable `gdpPercap`)
- **Eje Y**: población (variable `pop`)

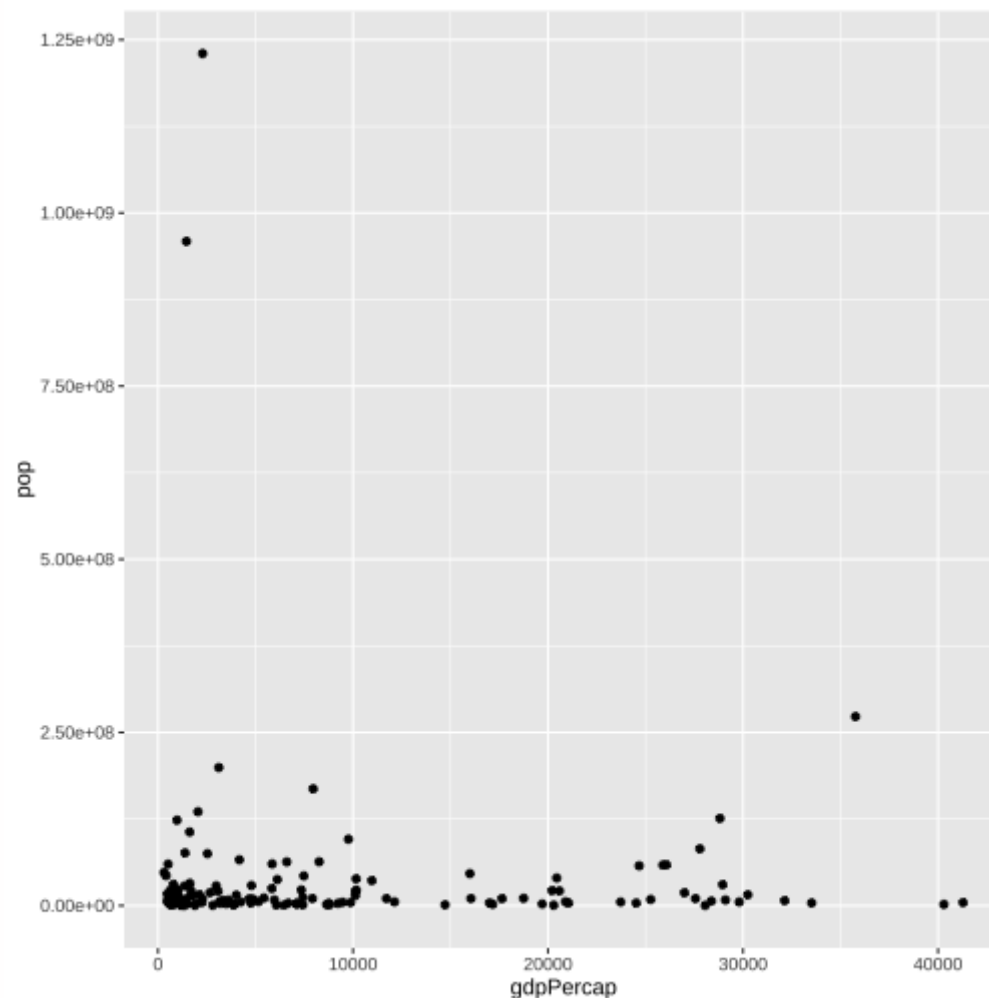
¿Qué necesitamos?

- **Datos**: el conjunto filtrado `gapminder_1997`.
- **Mapeado**: indicarle dentro de `aes()` (aesthetics) las variables a pintar en cada coordenada. Todo lo que esté **dentro de `aes()` dependerá de los datos** (en este caso `aes(x = gdpPercap, y = pop)`).

```
gapminder_1997 %>% ggplot(aes(x = gdpPercap, y = pop))
```

- **Elegir una geometría**: optaremos por **puntos** con `geom_point()`.

```
gapminder_1997 %>%  
  ggplot(aes(x = gdpPercap, y = pop)) +  
  geom_point() #<< Geometría
```

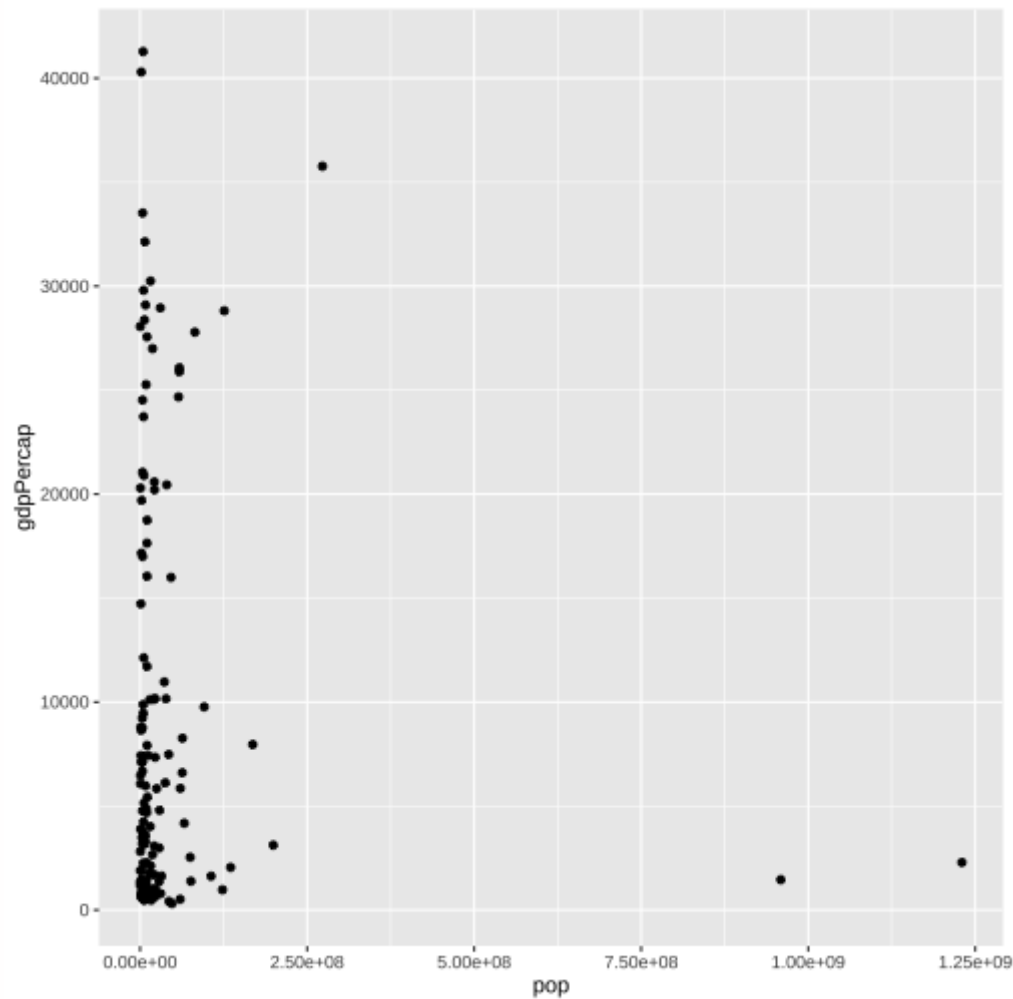


Primer intento: scatter plot

Vamos a profundizar en ese mapeado, cambiando el rol de los ejes:

- **Eje X:** población (variable `pop`)
- **Eje Y:** renta per cápita (variable `gdpPercap`)

```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = pop)) +  
  geom_point()
```

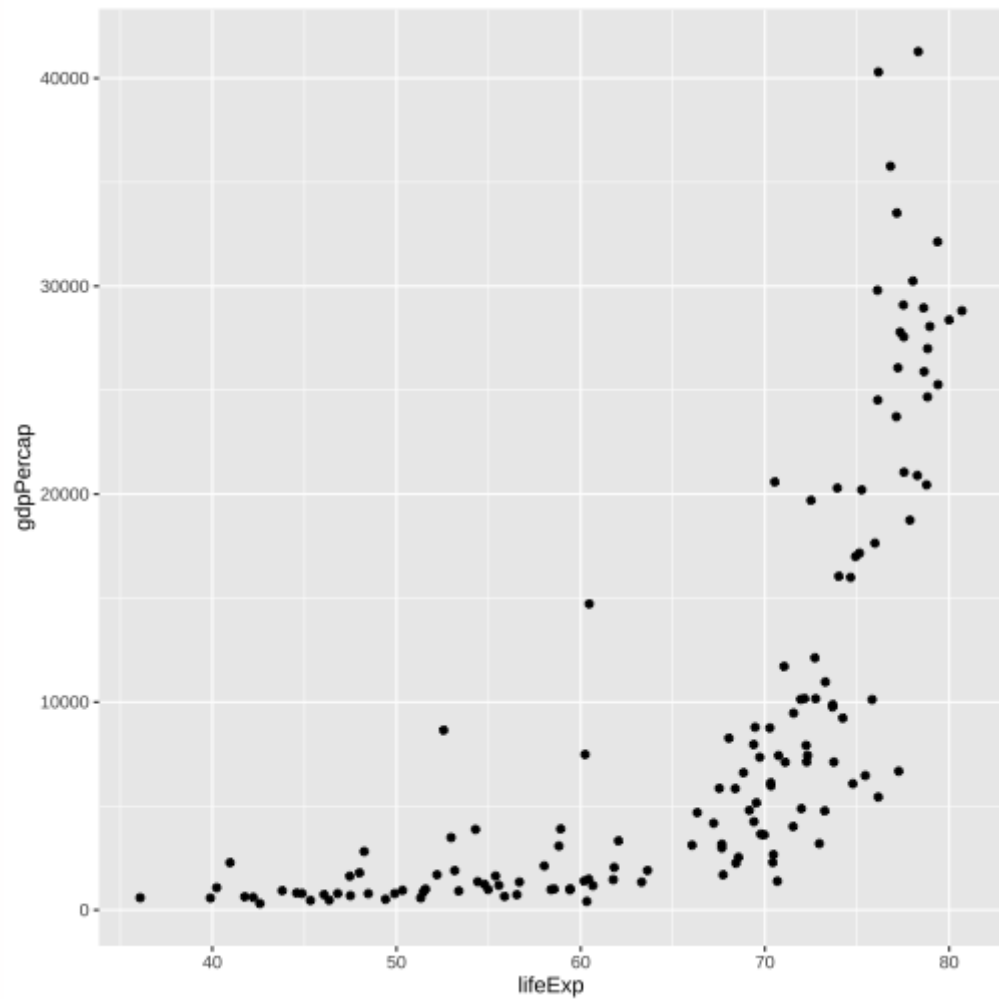


Primer intento: scatter plot

La idea podemos repetirla enfrentando ahora la **esperanza de vida** frente a **la renta per cápita**.

- **Eje X:** esperanza de vida (variable `lifeExp`)
- **Eje Y:** renta per cápita (variable `gdpPercap`)

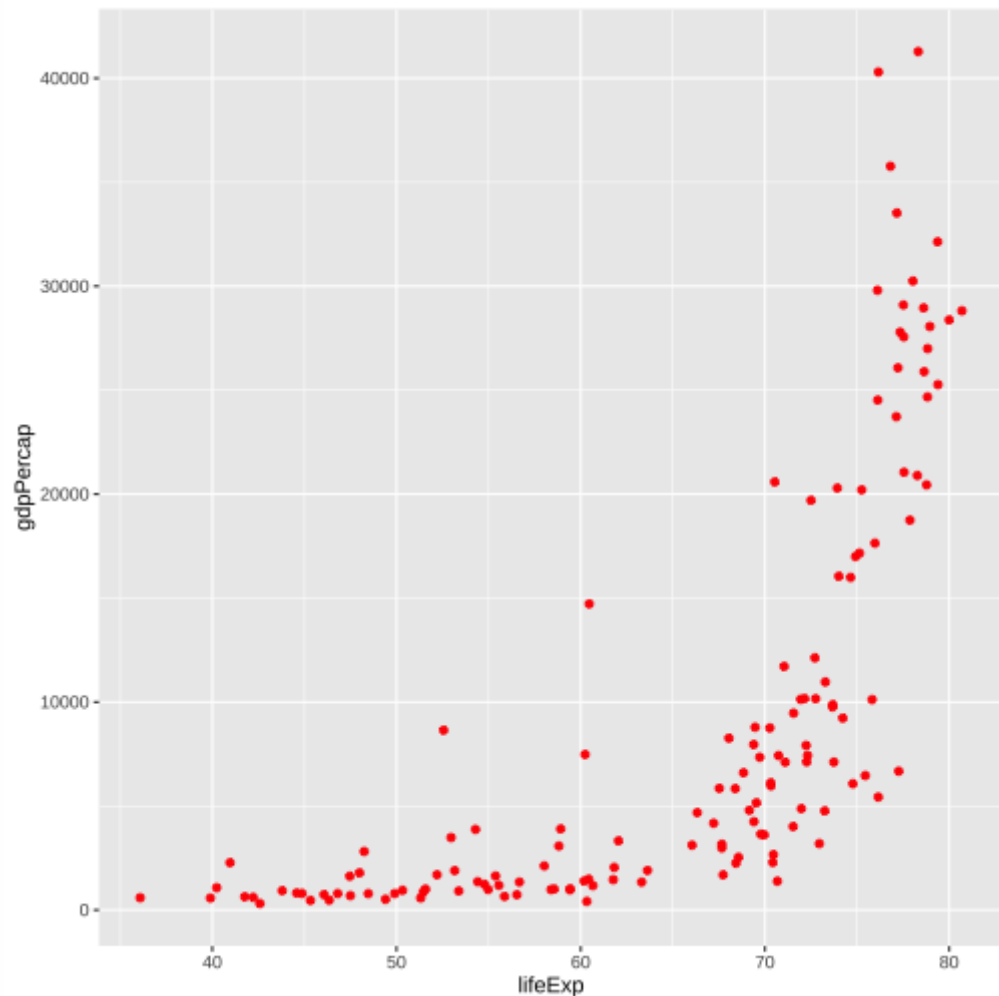
```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp)) +  
  geom_point()
```



Colores, tamaños y formas (constantes)

Para **cambiar el color de los puntos**, indicaremos dentro de `geom_point()` el color de la geometría con `color = ...` (en este caso, el color del punto). Empezaremos por un **color fijo**, por ejemplo `"red"` (existen otros como `"blue"`, `"black"`, `"yellow"`, etc)

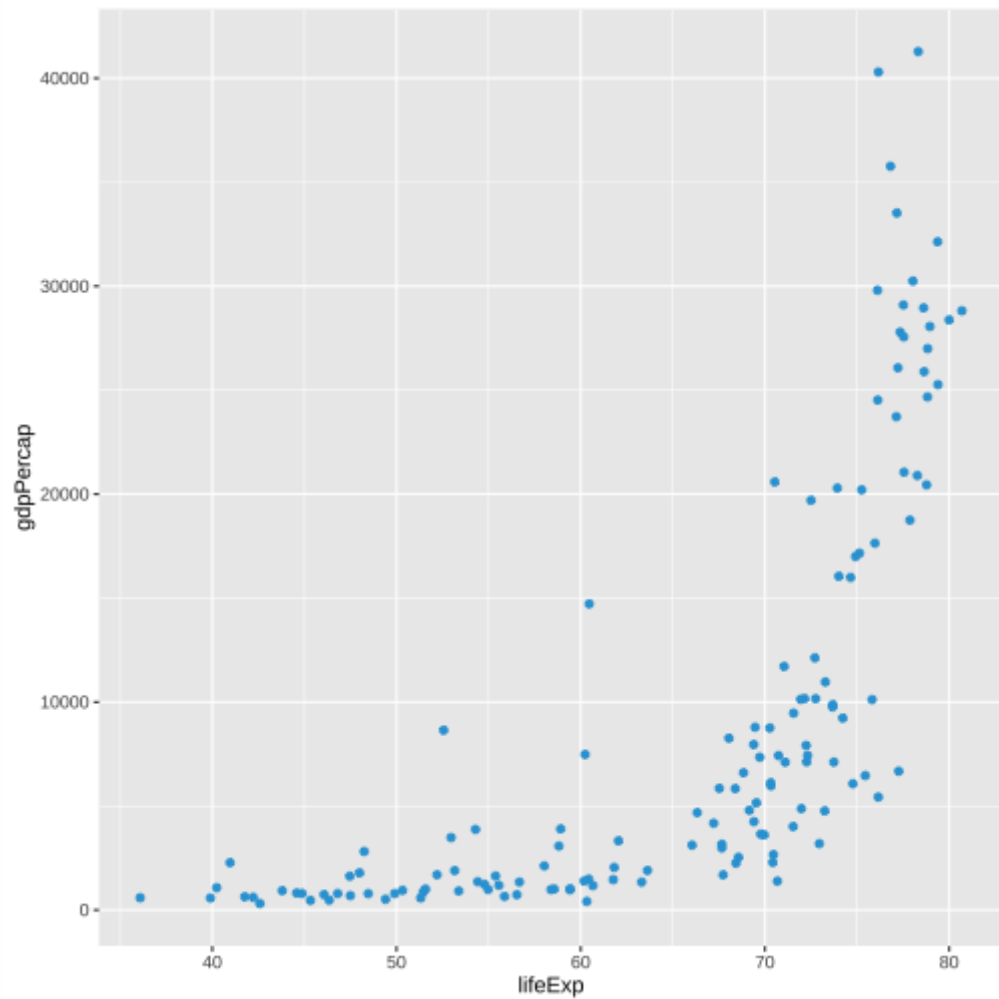
```
# Color con palabra reservada
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp)) +
  geom_point(color = "red")
```



Colores, tamaños y formas (constantes)

Los colores también podemos asignárselos por su **código hexadecimal**, consultando en la página <https://htmlcolorcodes.com/es/>, eligiendo el color que queramos. El código hexadecimal siempre comenzará con **#**

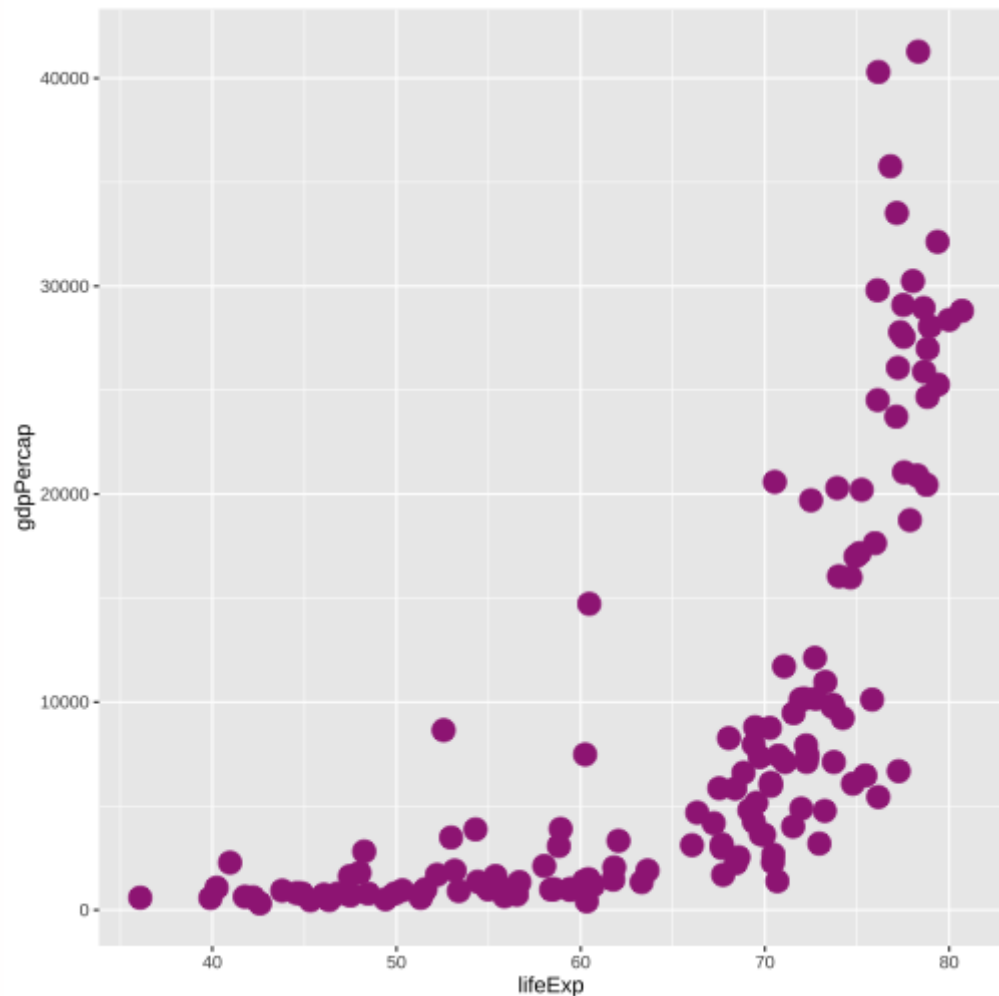
```
# Color en hexadecimal
# https://htmlcolorcodes.com/es/
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp)) +
  geom_point(color = "#2EA2D8")
```



Colores, tamaños y formas (constantes)

De la misma manera podemos **indicarle el tamaño de la geometría** (en este caso el **tamaño de los punto**) con `size = ...` (cuanto mayor sea el número, mayor será el tamaño de la geometría).

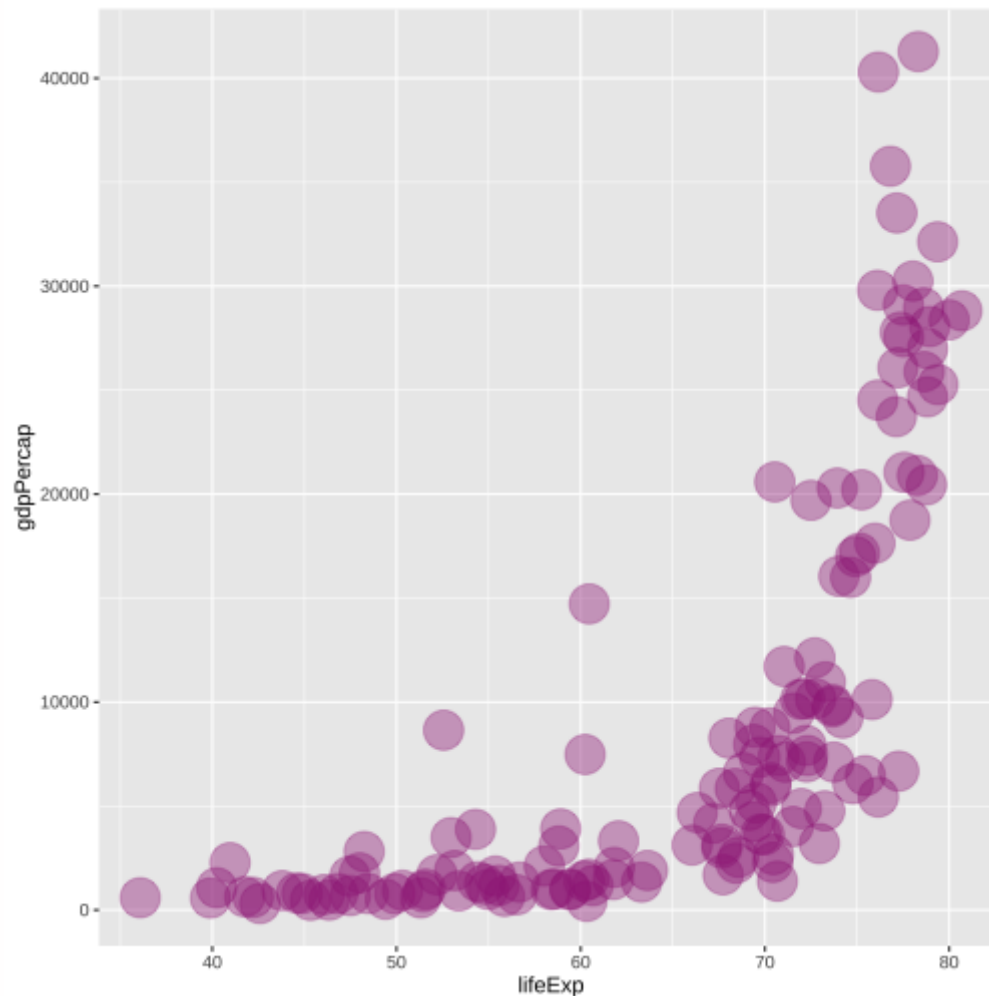
```
# Color y tamaño
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp)) +
  geom_point(color = "#A02B85", size = 5)
```



Colores, tamaños y formas (constantes)

También podemos jugar con la **transparencia del color** con `alpha = ...`: si `alpha = 1`, el color será totalmente opaco (por defecto); si `alpha = 0` será totalmente transparente.

```
# Color, tamaño y transparencia
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp)) +
  geom_point(color = "#A02B85", size = 9,
             alpha = 0.4)
```

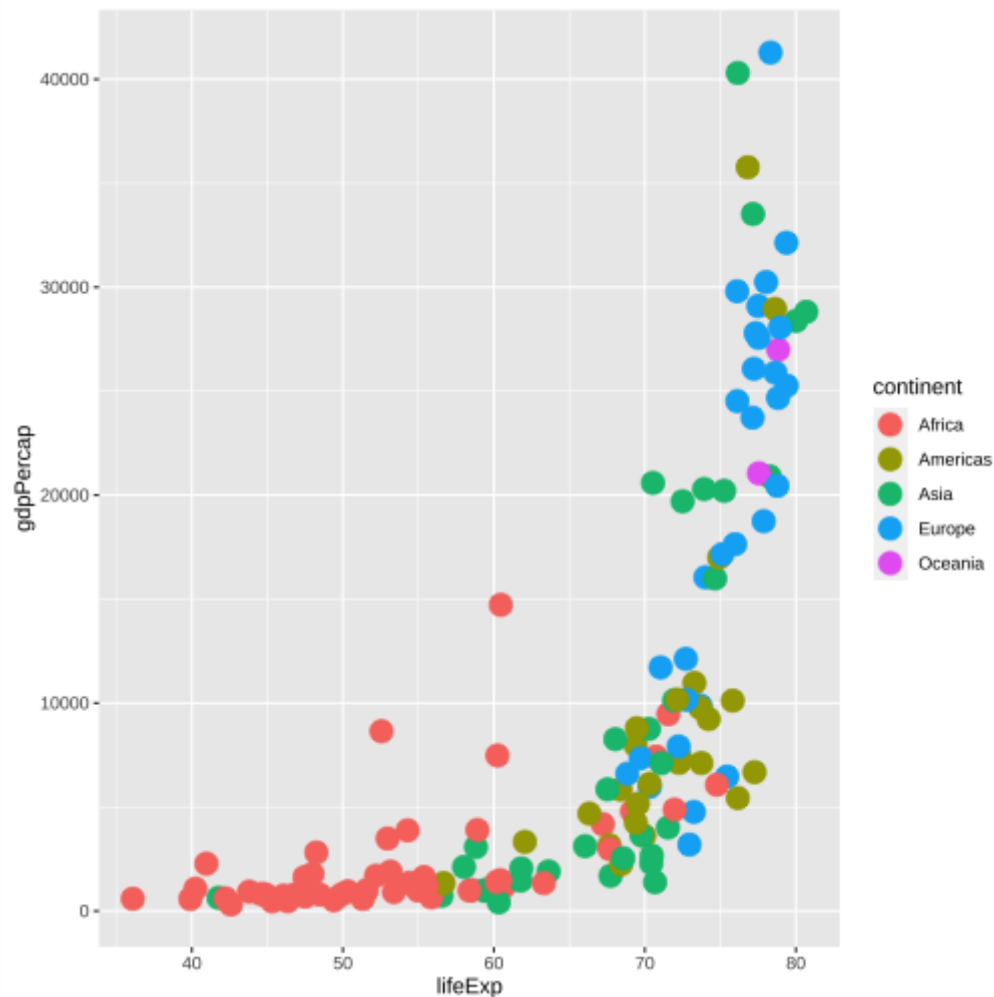


Mapear aesthetics

Hasta ahora los **parámetros estéticos** se los hemos pasado fijos y **constantes**. Pero la verdadera potencia y versatilidad de **ggplot** es entender todos esos parámetros como entendemos el mapeado coordenadas: podemos **mapear los atributos estéticos** en **aes()** para que dependan de variables de los datos

Por ejemplo, vamos a **asignar un color a cada dato en función de su continente**.

```
# Tamaño fijo
# Color por continentes
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp,
             color = continent)) +
  geom_point(size = 5)
```

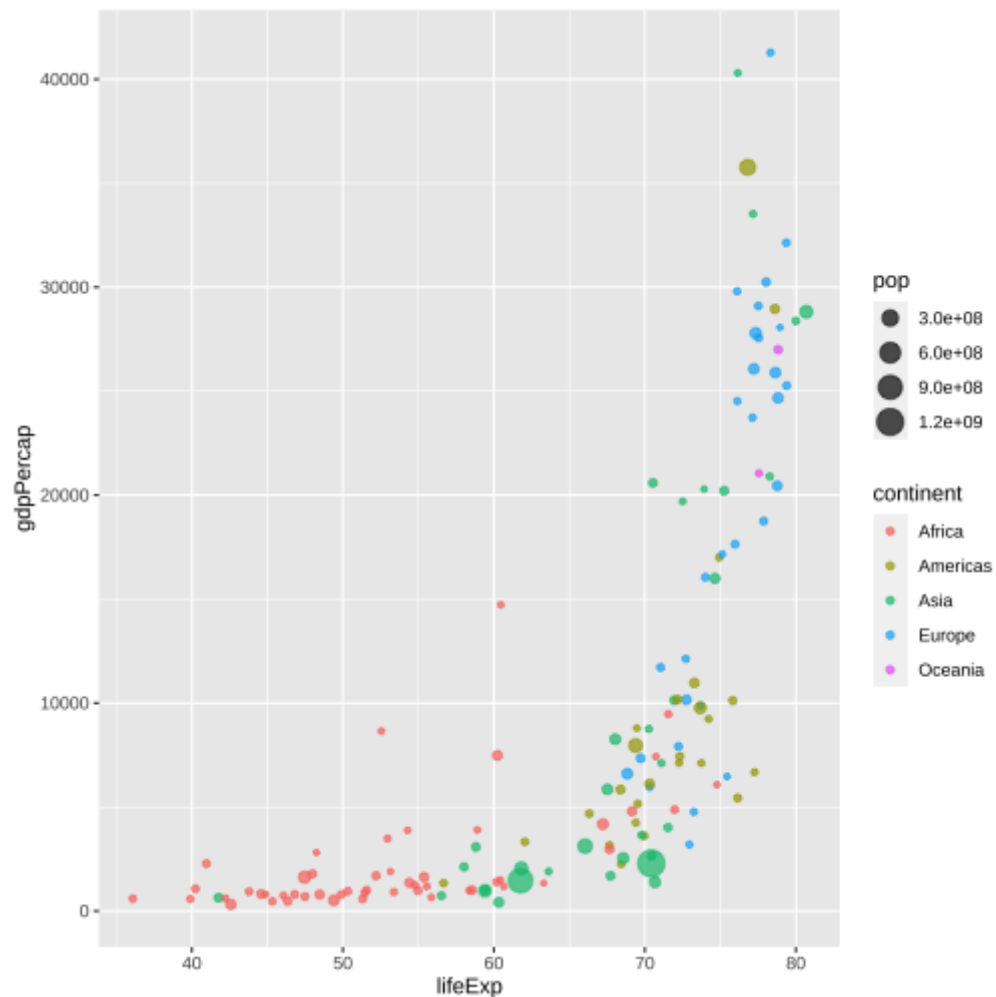


Mapear aesthetics

Podemos combinarlo con lo que hemos hecho anteriormente:

- **color** en función del **continente**.
- **tamaño** en función de la **población**
- **transparencia** fija del 70%

```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp,  
             color = continent, size = pop)) +  
  geom_point(alpha = 0.7)
```



Mapear aesthetics

En lugar de jugar con el color, también podríamos añadir las variables en función de la **forma de la geometría** (en este caso la forma de los «puntos») con `shape = ...`.

```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp,  
             shape = continent, size = pop)) +  
  geom_point(alpha = 0.7)
```

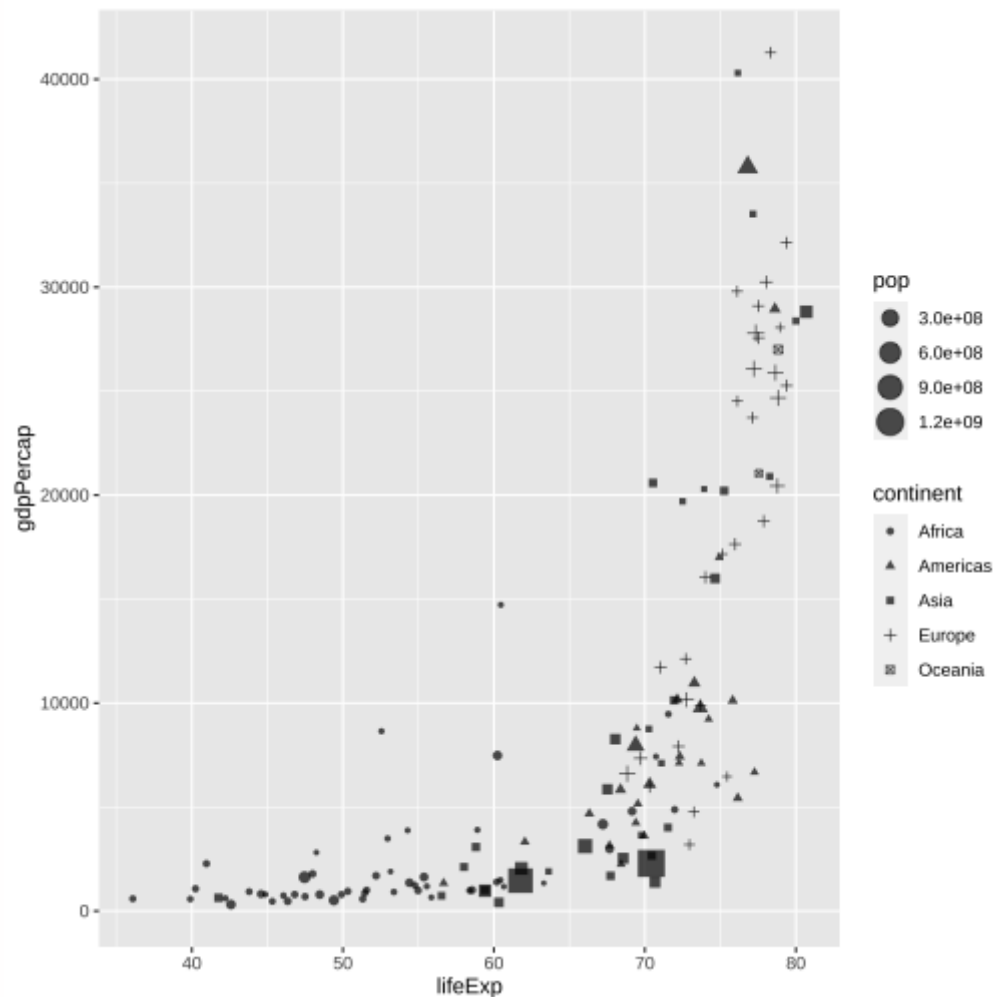


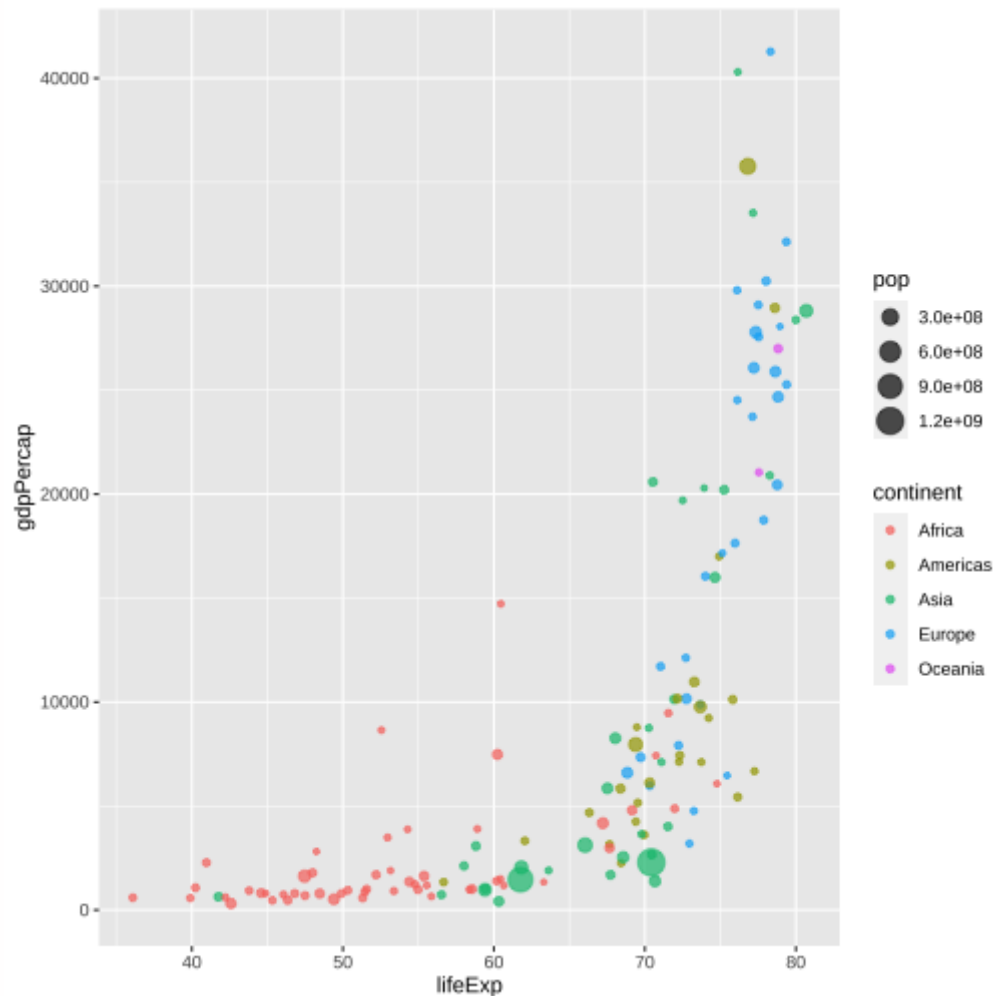
Gráfico multidimensional en 2D

Reflexionemos sobre el gráfico anterior:

- **color** en función del **continente**.
- **tamaño** en función de la **población**
- **transparencia** fija del 70%

Usando los datos hemos conseguido **dibujar en un gráfico bidimensional 4 variables** (**lifeExp** y **gdpPercap** en los ejes (X, Y)), **continent** como color y **pop** como tamaño de la geometría) con muy pocas líneas de código.

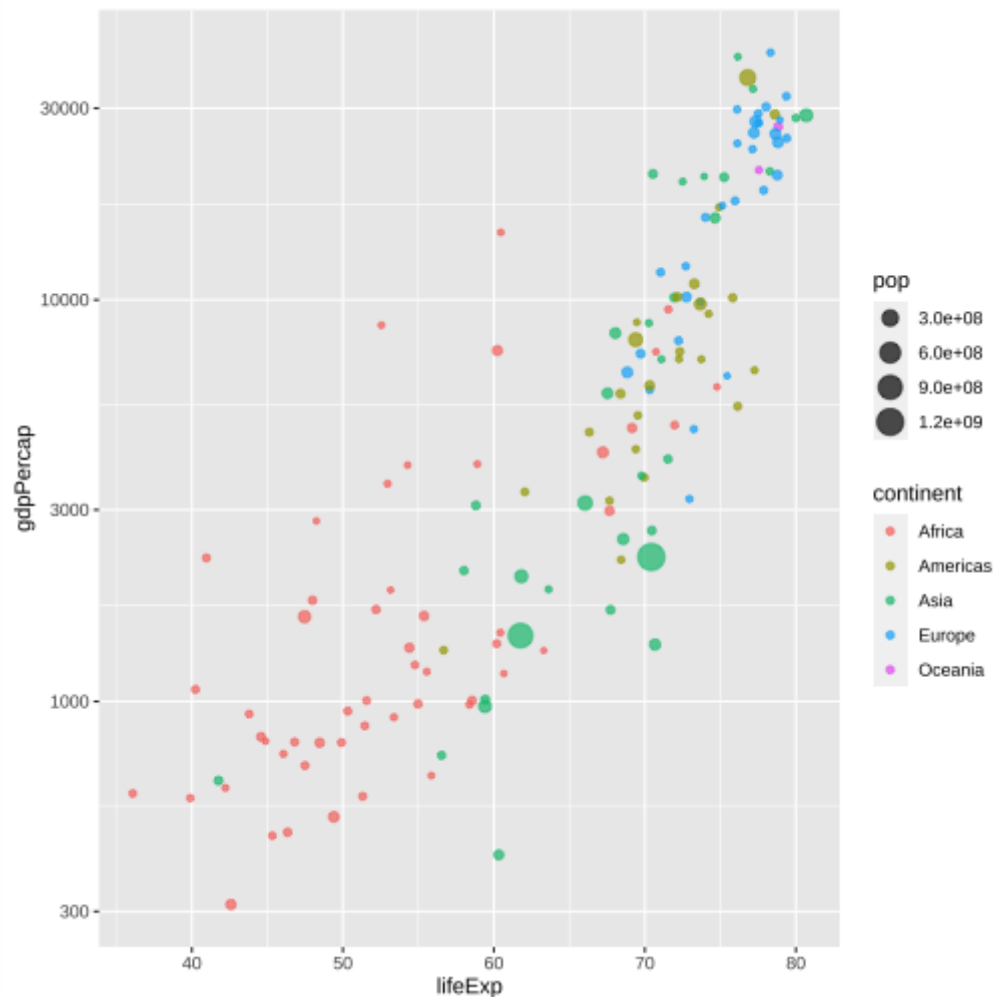
```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp,  
             color = continent, size = pop)) +  
  geom_point(alpha = 0.7)
```



Escala de los ejes

A veces nos puede ser más conveniente **representar alguna de las variables** en otras escalas, por ejemplo en **escala logarítmica** (importante indicarlo en el gráfico), lo que podemos hacer fácilmente con `scale_x_log10()` y/o `scale_y_log10()`.

```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp,  
             color = continent, size = pop)) +  
  geom_point(alpha = 0.7) +  
  # Eje Y con escala logarítmica  
  scale_y_log10()
```

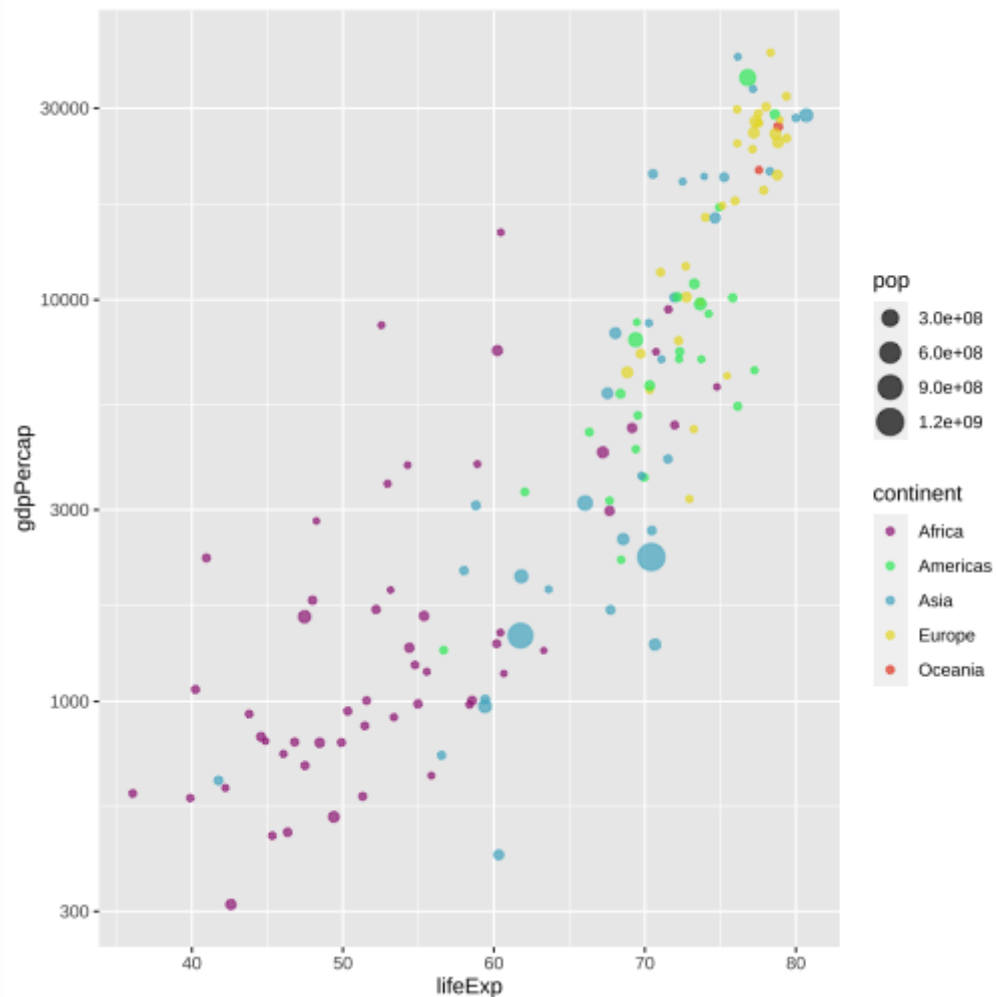


Paletas de colores

Si no indicamos nada, **R** selecciona automáticamente una **paleta de colores**, pero podemos indicarle alguna paleta concreta de varias maneras.

La primera y más inmediata es indicarle los **colores manualmente**: con **scale_color_manual** le podemos indicar un **vector de colores**.

```
pal <- c("#A02B85", "#2DE86B", "#4FB2CA",  
        "#E8DA2D", "#E84C2D")  
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp,  
            color = continent, size = pop)) +  
  geom_point(alpha = 0.7) +  
  scale_y_log10() +  
  # Escala manual de colores  
  scale_color_manual(values = pal)
```



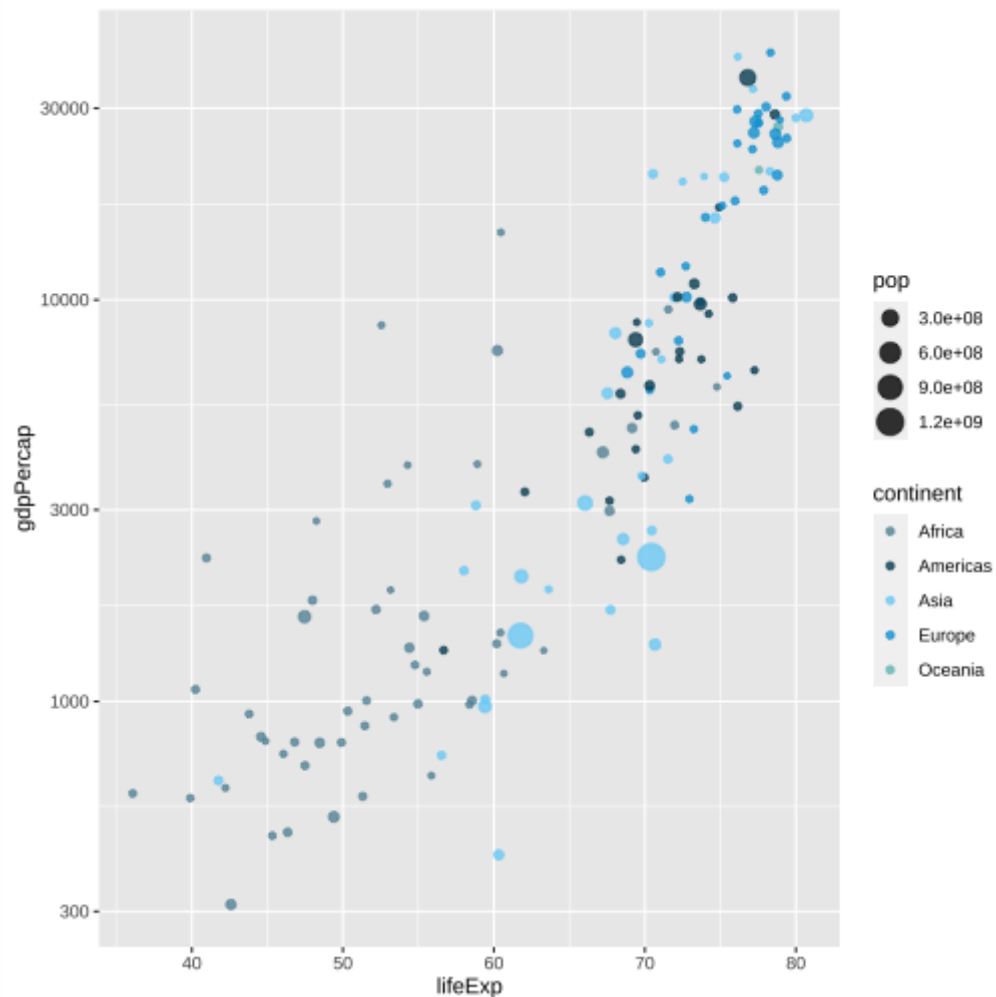
Paletas de colores

Otra opción es elegir alguna de las **paletas de colores disponibles** en el paquete `{ggthemes}`:

- `scale_color_economist()`: paleta de colores basada en los colores de The Economist.

```
library(ggthemes)

# scale_color_economist()
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp,
             color = continent, size = pop)) +
  geom_point(alpha = 0.8) +
  scale_y_log10() +
  scale_color_economist()
```

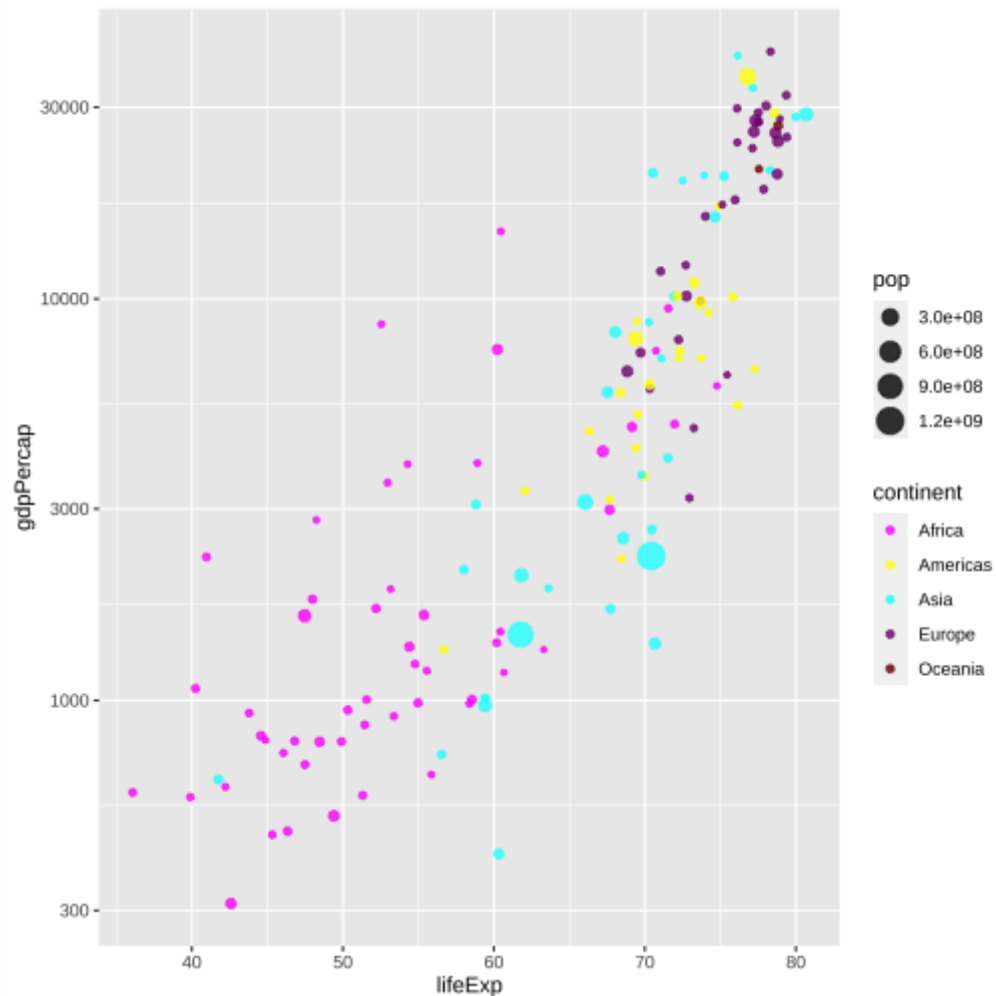


Paletas de colores

Otra opción es elegir alguna de las **paletas de colores disponibles** en el paquete `{ggthemes}`:

- `scale_color_excel()`: paleta de colores basada en los colores del Excel.

```
library(ggthemes)
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp,
             color = continent, size = pop)) +
  geom_point(alpha = 0.8) +
  scale_y_log10() +
  scale_color_excel()
```

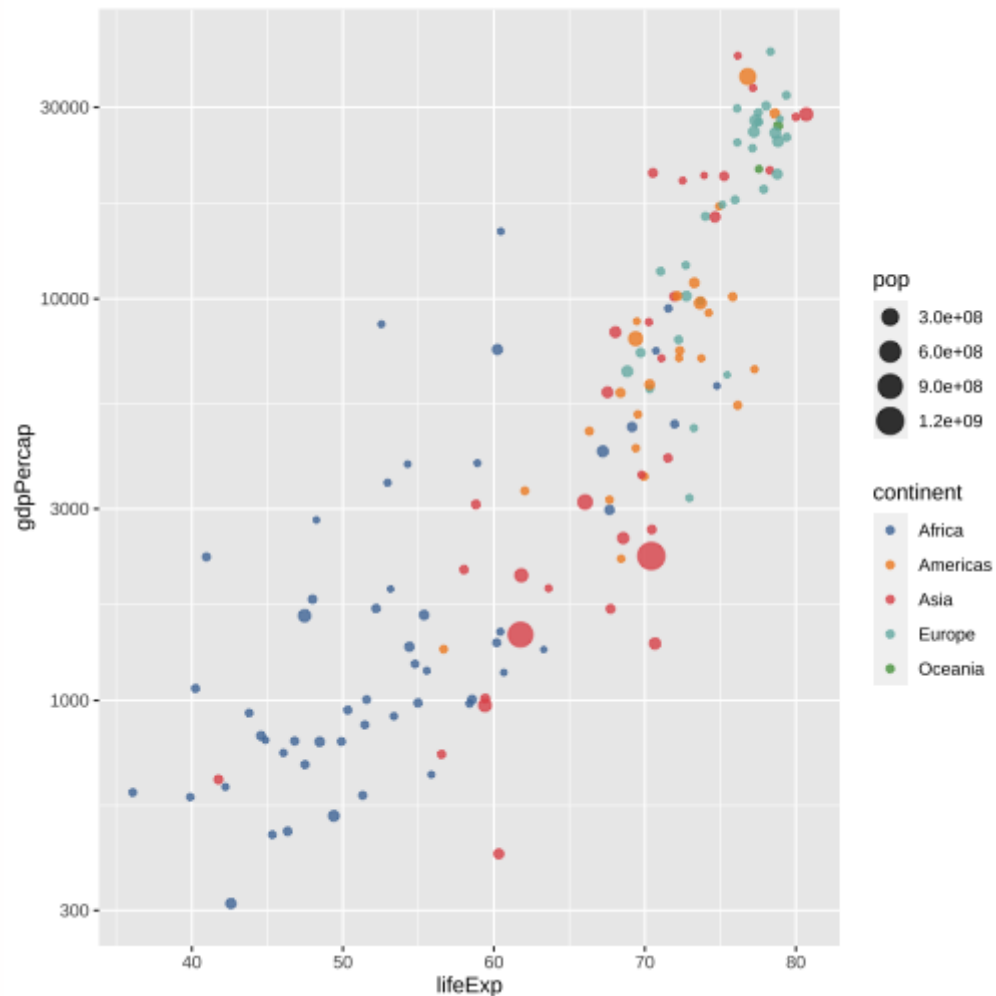


Paletas de colores

Otra opción es elegir alguna de las **paletas de colores disponibles** en el paquete `{ggthemes}`:

- `scale_color_tableau()`: paleta de colores basada en los colores de Tableau.

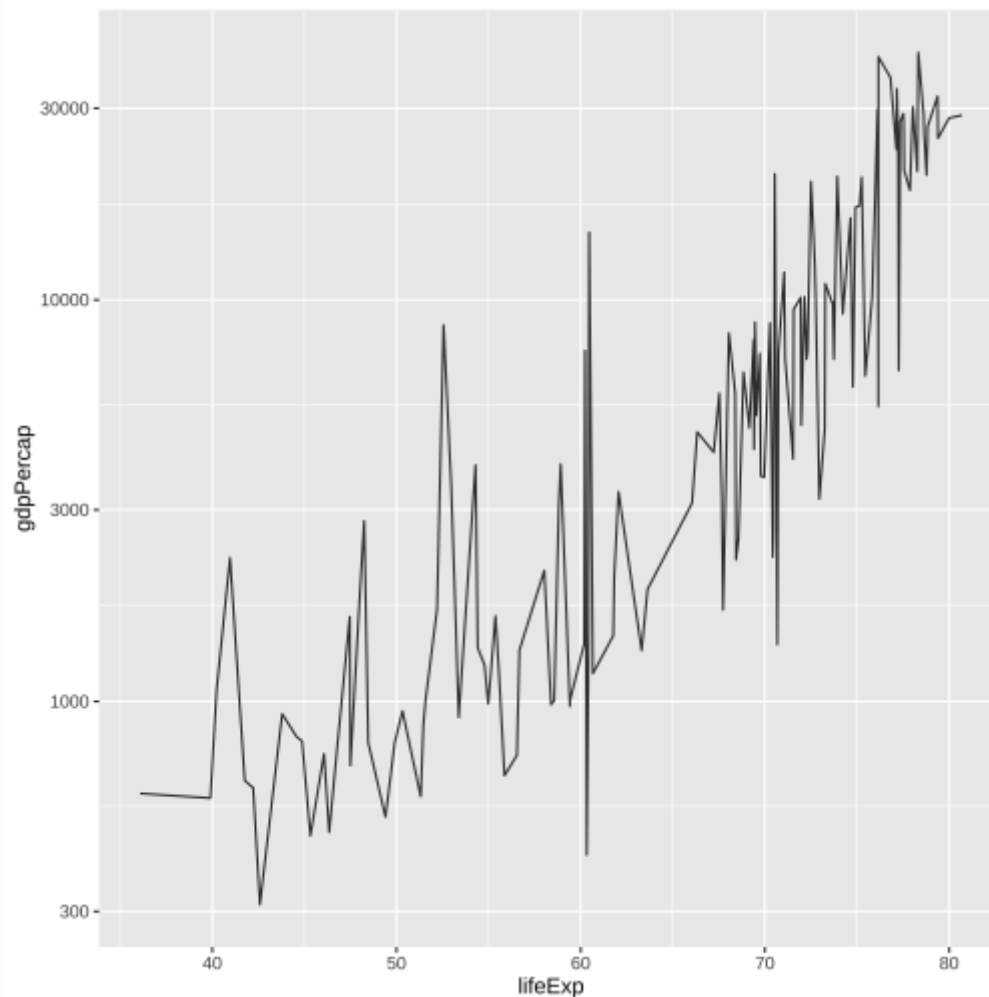
```
library(ggthemes)
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp,
             color = continent, size = pop)) +
  geom_point(alpha = 0.8) +
  scale_y_log10() +
  scale_color_tableau()
```



Geometrías (geom)

Hemos jugado un poco con las formas, tamaños y colores, pero siempre ha sido un diagrama de dispersión con puntos. Al igual que hemos usado `geom_point()`, podríamos usar otras geometrías como **líneas** con `geom_line()`.

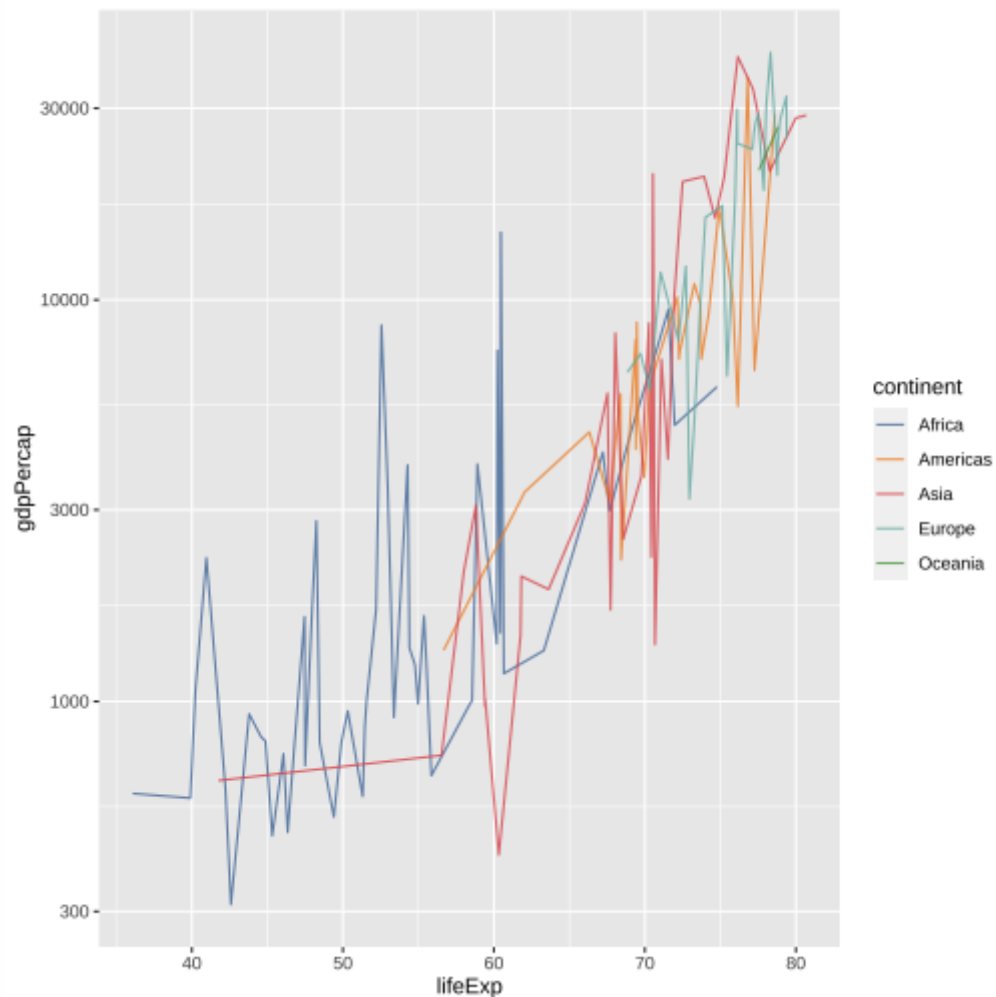
```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp)) +  
  geom_line(alpha = 0.8) +  
  scale_y_log10() +  
  scale_color_tableau()
```



Geometrías (geom)

Asignado los colores a la variable **continent**, automáticamente obtenemos cada curva separada por continente.

```
# Separando por continente
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp,
             color = continent)) +
  geom_line(alpha = 0.8) +
  scale_y_log10() +
  scale_color_tableau()
```



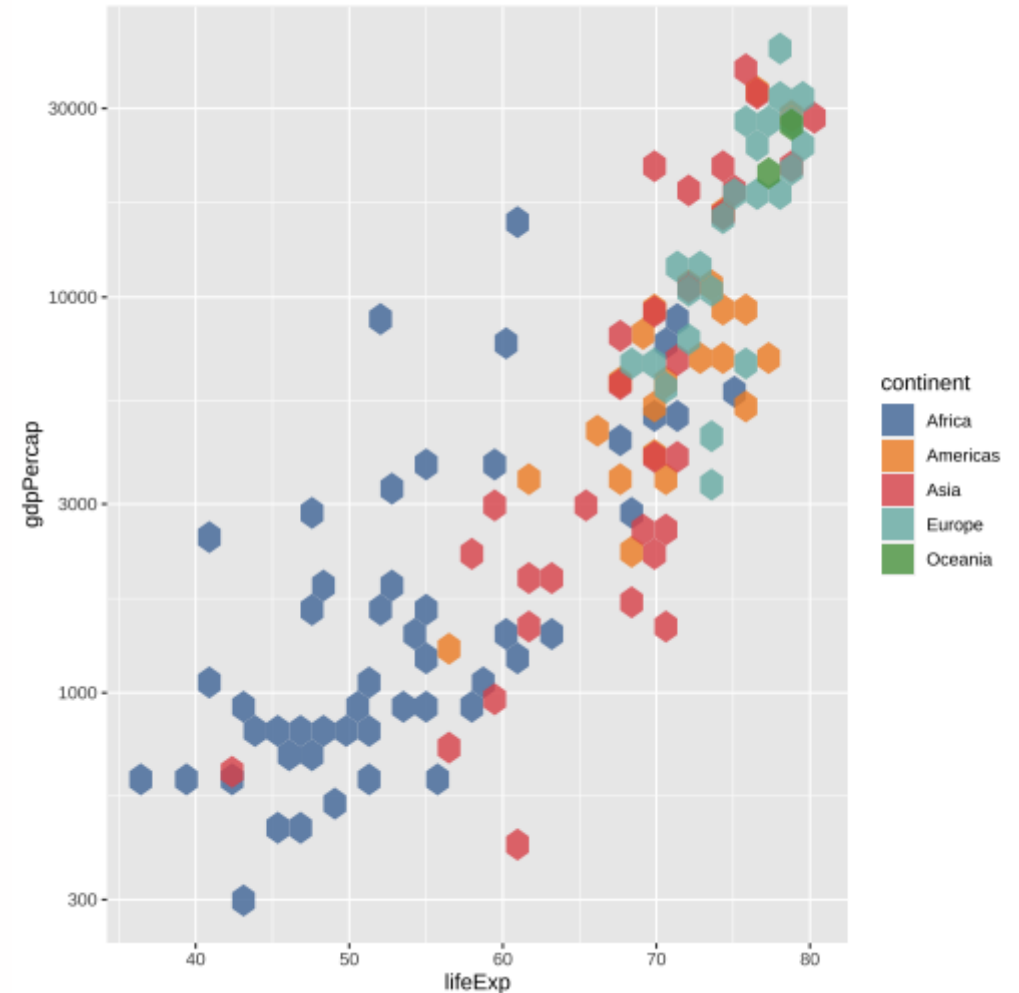
Geometrías (geom)

La **filosofía es siempre la misma**: dado que cada elemento lo podemos tratar de forma individual, pasar de un gráfico a otro es relativamente sencillo, sin más que cambiar `geom_point()` por `geom_line()`.

De la misma manera podemos dibujar un diagrama de dispersión con **formas hexagonales** con `geom_hex()`. Dado que ahora nuestra geometría **tiene volumen** tendremos dos parámetros: **color** para el contorno y **fill** para el **relleno** (fíjate que también cambiamos `scale_color_tableau()` por `scale_fill_tableau()`)

```
library(hexbin)

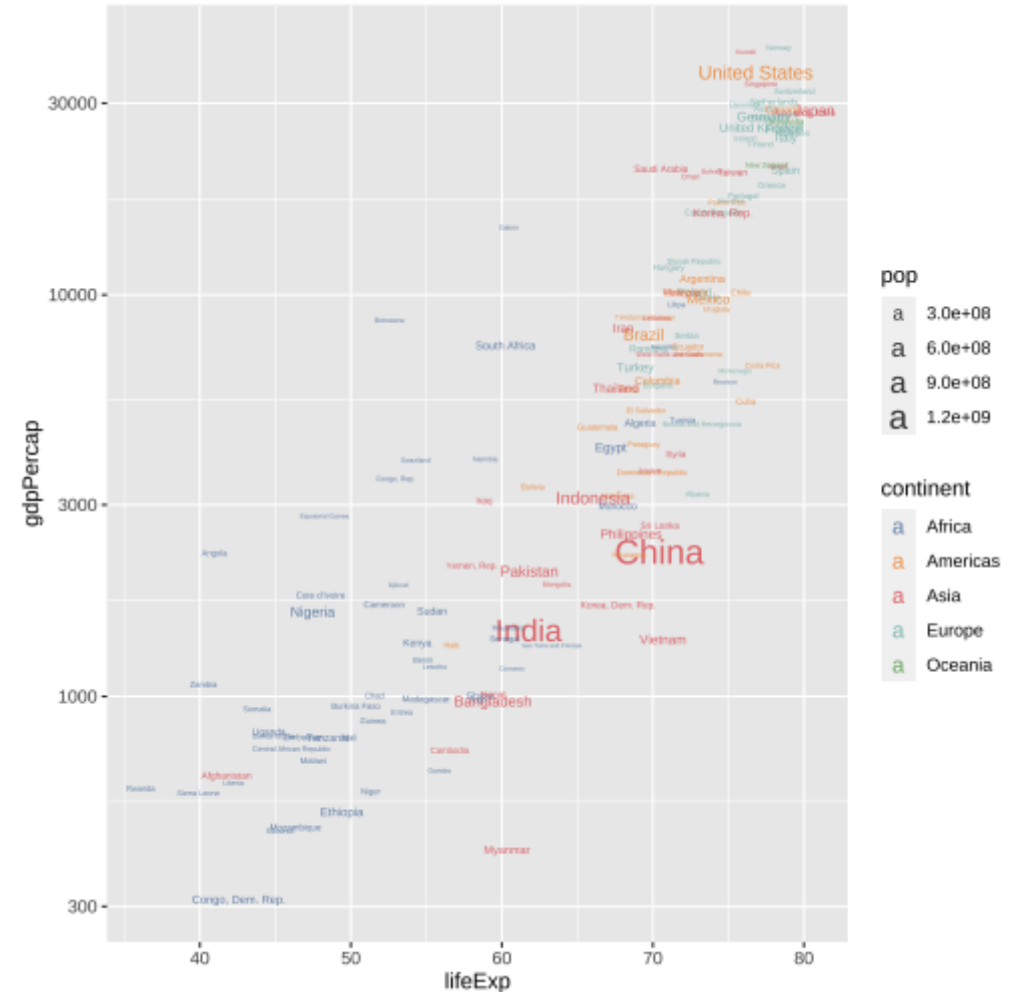
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp,
             fill = continent, size = pop)) +
  geom_hex(alpha = 0.8) +
  scale_y_log10() +
  scale_fill_tableau()
```



Geometrías (geom)

Tenemos varias funciones de este tipo, como `geom_tile()`, que nos visualiza los datos con «mosaicos» (como baldosas), o `geom_text()`, con la podemos hacer que en lugar de una forma geométrica aparezcan **textos que tengamos en alguna variable**, que la pasaremos en `aes()` por el **parámetro label** (en este caso, la variable de la que tomará los nombres será `country`).

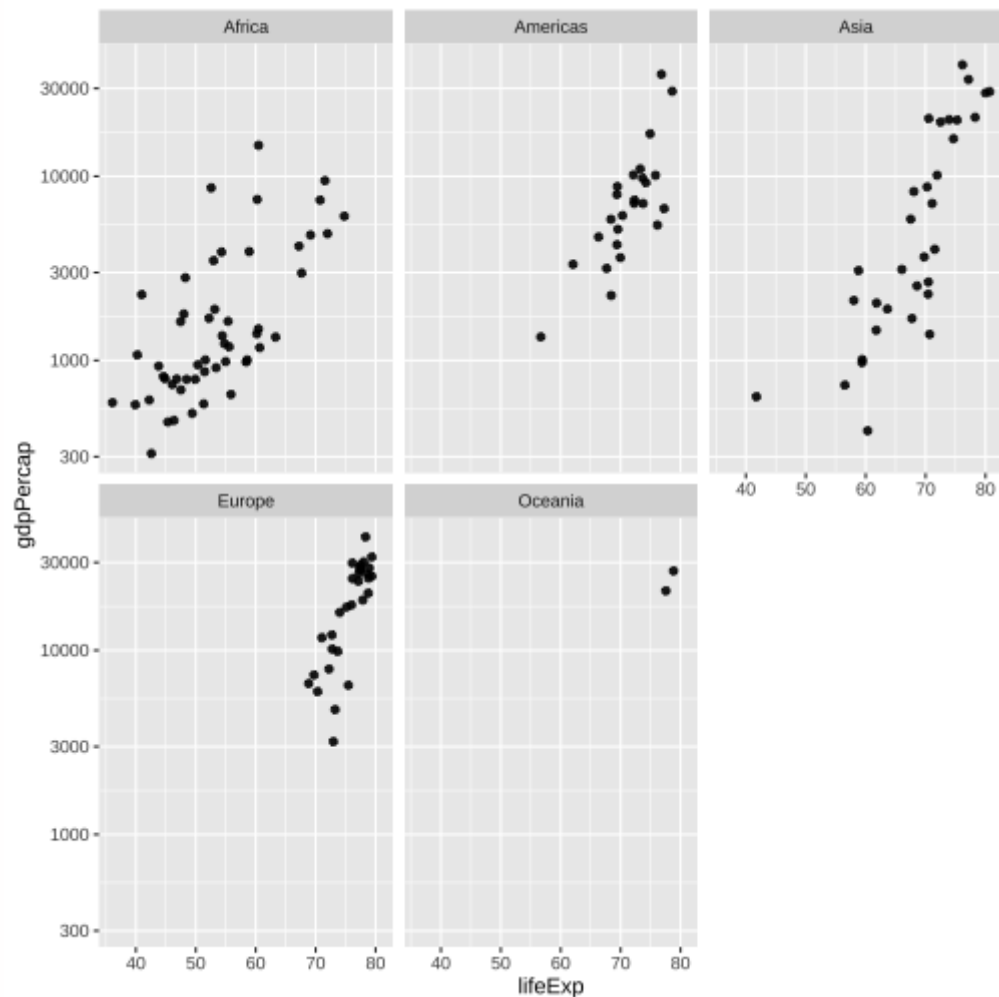
```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp,  
             color = continent, size = pop, label = country)) +  
  geom_text(alpha = 0.8) +  
  scale_y_log10() +  
  scale_color_tableau()
```



Componer (facet)

Hasta ahora hemos pintado una sola gráfica, **codificando información en colores y formas**. Pero también podemos **dividir/desagregar los gráficos (facetar) por variables**, pintando por ejemplo un **gráfico por continente**, mostrando todos los gráficos a la vez pero por separado, con `facet_wrap()`.

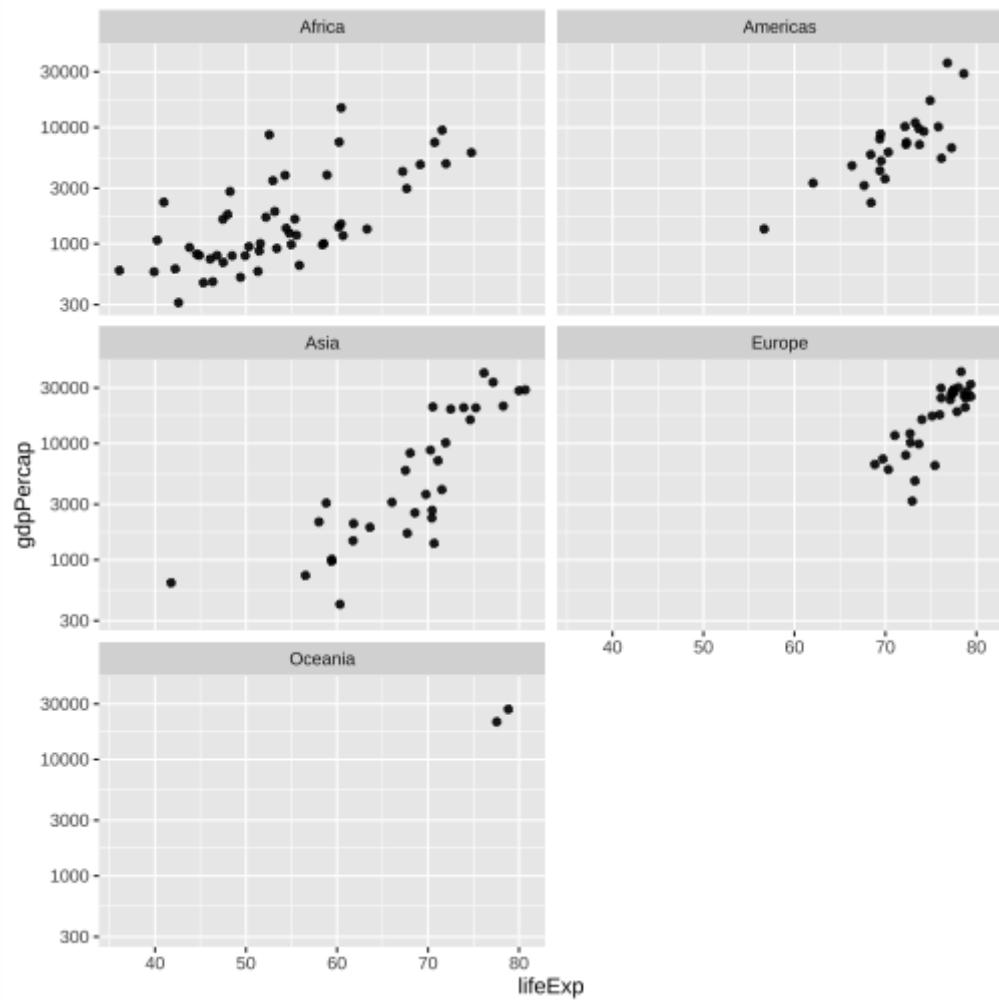
```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp)) +  
  geom_point(alpha = 0.9) +  
  scale_y_log10() +  
  facet_wrap(~ continent)
```



Componer (facet)

También le podemos pasar **argumentos opcionales** para indicarle el **número de columnas o de filas** que queremos.

```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp)) +  
  geom_point(alpha = 0.9) +  
  scale_y_log10() +  
  facet_wrap(~ continent, nrow = 3)
```

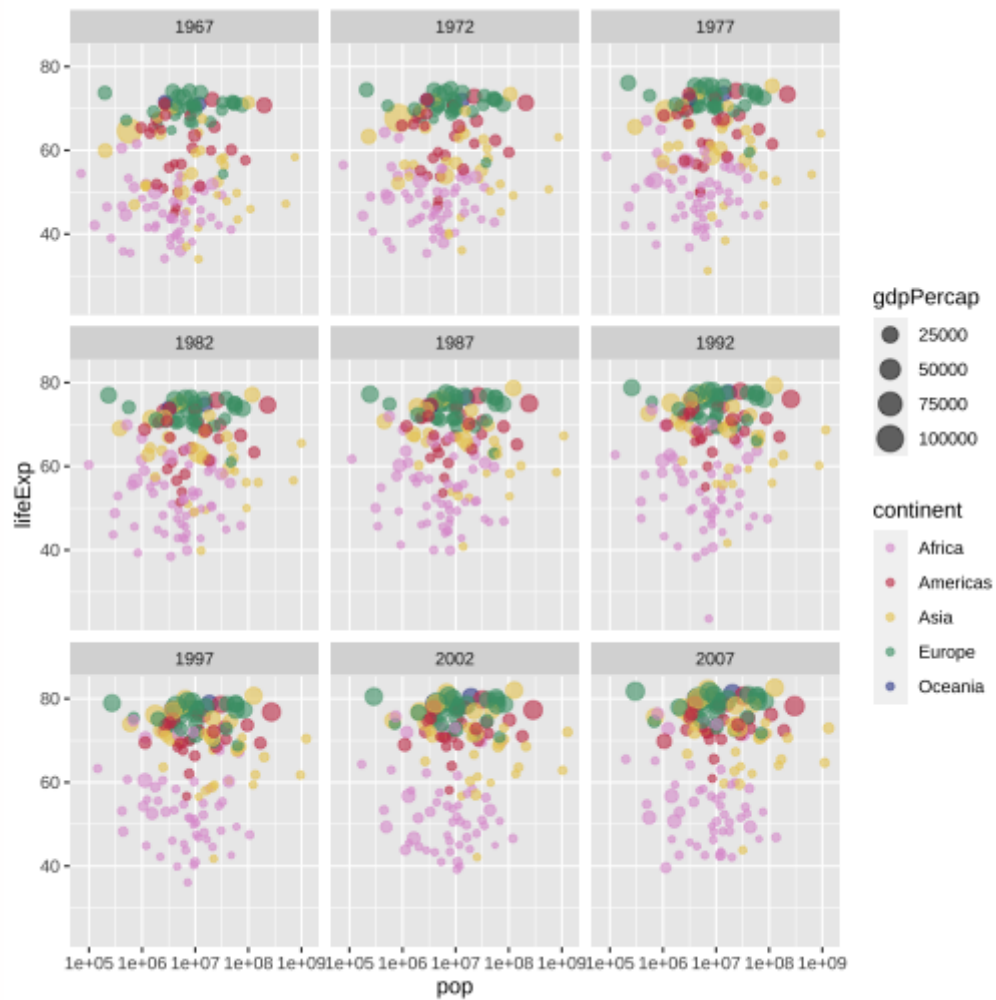


Componer (facet)

De esta manera podríamos incluso **visualizar el fichero de datos originales incluye hasta 5 variables** en un gráfico bidimensional:

- las variables **pop** y **lifeExp** en los **ejes**.
- la variable **gdpPercap** en el **tamaño**.
- la variable **continent** en el **color**.
- la variable **year** en la composición de **facet_wrap()**.

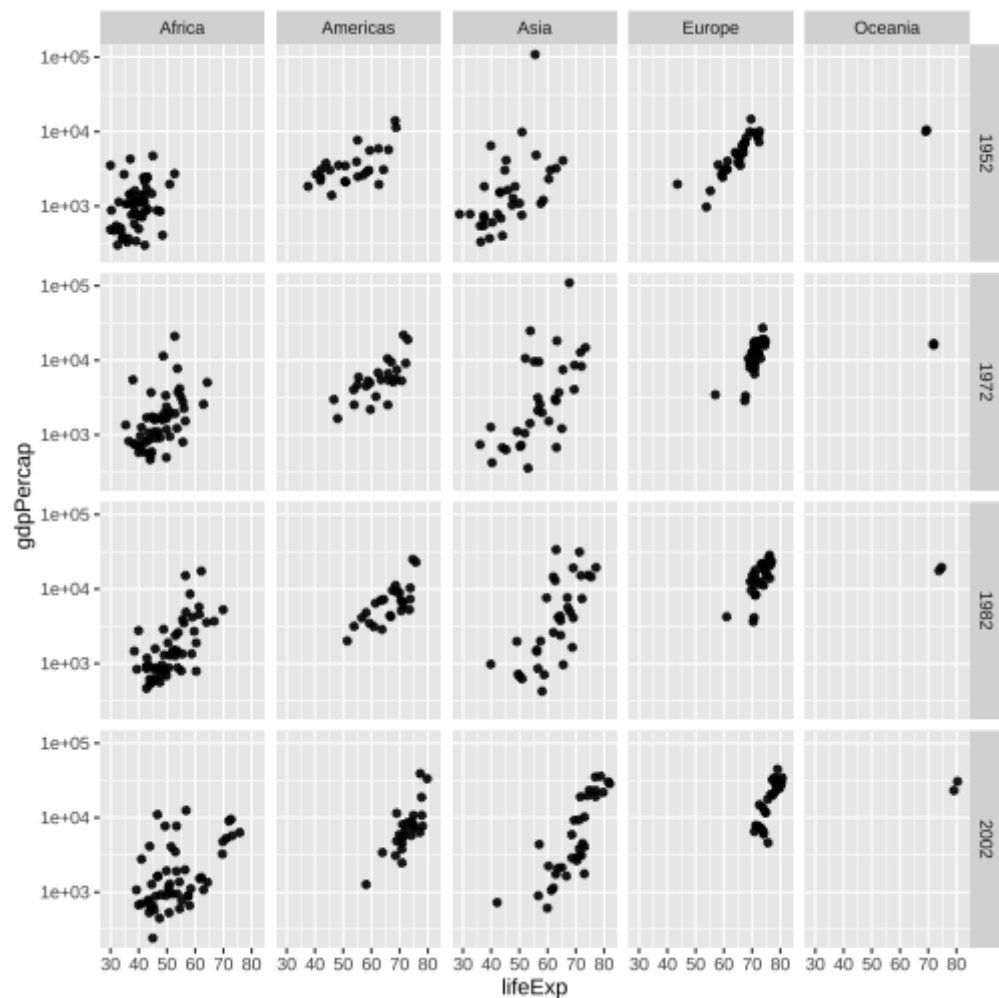
```
library(MetBrewer)
gapminder %>%
  filter(year > 1962) %>%
  ggplot(aes(y = lifeExp, x = pop,
             size = gdpPercap, color = continent)) +
  geom_point(alpha = 0.6) +
  scale_x_log10() +
  scale_colour_manual(values =
                      met.brewer("Klimt")) +
  facet_wrap(~ year)
```



Componer (facet)

Con `facet_grid()` podemos incluso **organizar una cuadrícula en base a dos variables**, por ejemplo que haya una **fila por año** (vamos a usar la tabla original en los **años 1952, 1972, 1982 y 2002**) y una **columna por continente**.

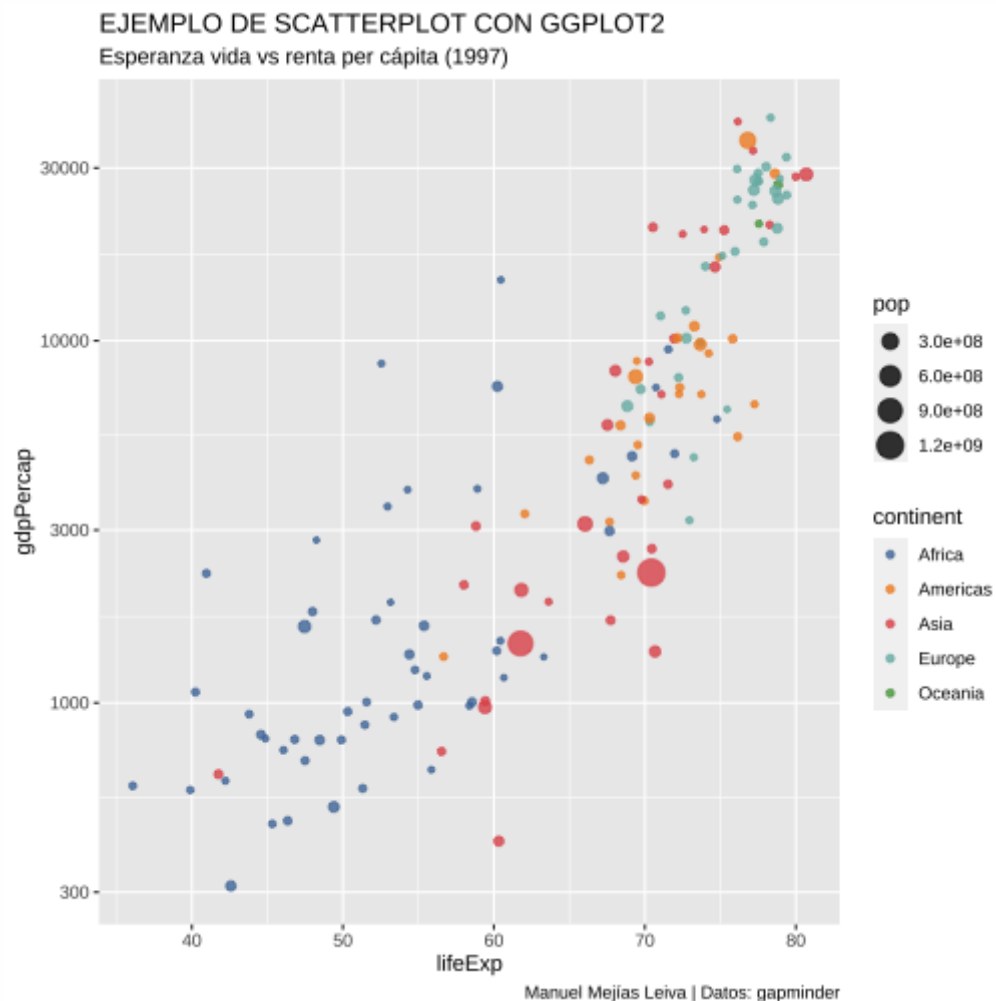
```
gapminder %>%  
  filter(year %in% c(1952,1972,1982,2002)) %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp)) +  
  geom_point(alpha = 0.9) +  
  scale_y_log10() +  
  facet_grid(year ~ continent)
```



Coordenadas y tema

Los gráficos pueden además **personalizarse añadiendo**, por ejemplo, *títulos y subtítulos** de la gráfica con `labs()`, asignando textos a `title`, `subtitle` y `caption`.

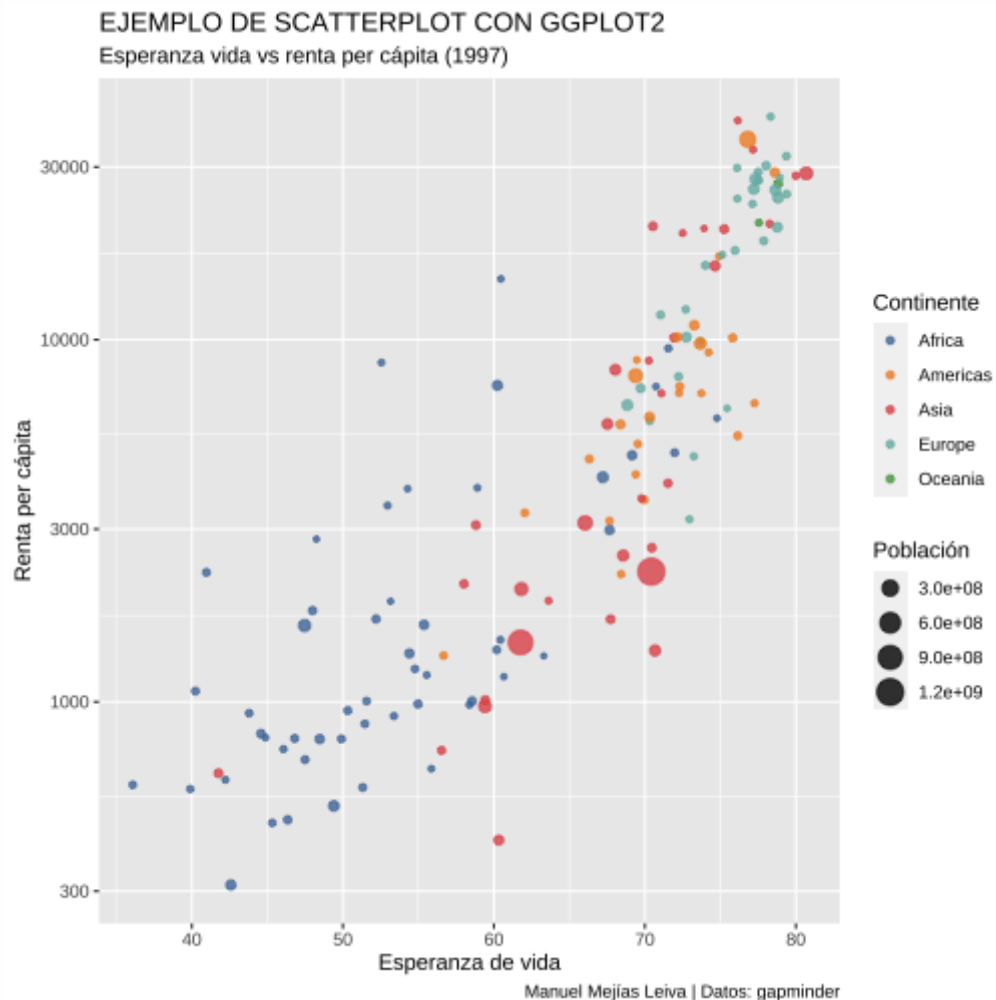
```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp,  
             color = continent, size = pop)) +  
  geom_point(alpha = 0.8) +  
  scale_y_log10() +  
  scale_color_tableau() +  
  labs(title = "EJEMPLO DE SCATTERPLOT CON GGLOT2",  
        subtitle =  
          "Esperanza vida vs renta per cápita (1997)",  
        caption = "Manuel Mejías Leiva | Datos: gapminder")
```



Coordenadas y tema

También podemos **personalizar algunos aspectos extras**, como el **título que vamos a dar a los ejes** o el **título de las leyendas**.

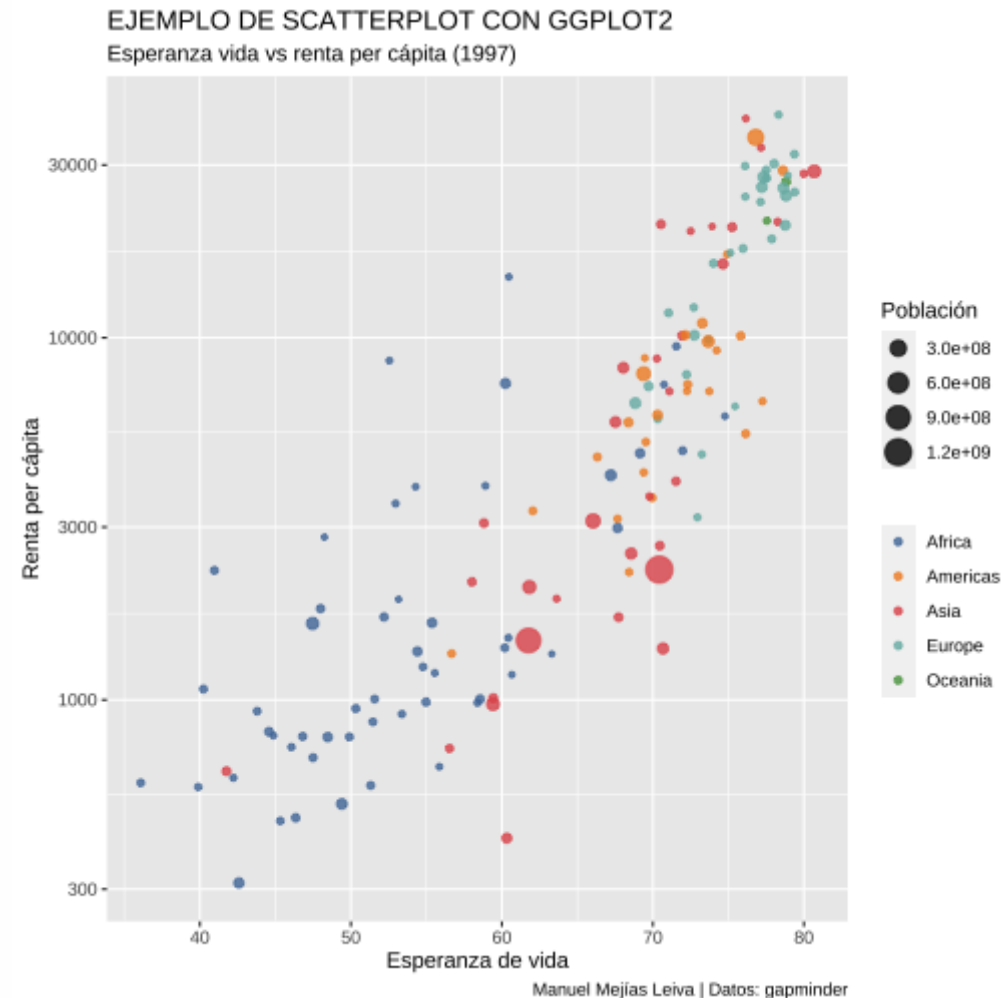
```
gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp,
             color = continent, size = pop)) +
  geom_point(alpha = 0.8) +
  # Eje Y con escala logarítmica
  scale_y_log10() +
  scale_color_tableau() +
  labs(x = "Esperanza de vida",
       y = "Renta per cápita",
       color = "Continente",
       size = "Población",
       title = "EJEMPLO DE SCATTERPLOT CON GGLOT2",
       subtitle = "Esperanza vida vs renta per cápita (1997)",
       caption = "Manuel Mejías Leiva | Datos: gapminder")
```



Coordenadas y tema

También podemos **ocultar algún nombre de las leyendas** (o ambos) si ya es explícito de lo que se está hablando. Por ejemplo, vamos a indicarle que no queremos el nombre de la leyenda en continentes, haciendo `color = NULL` (la variable que codifica los continentes a NULL).

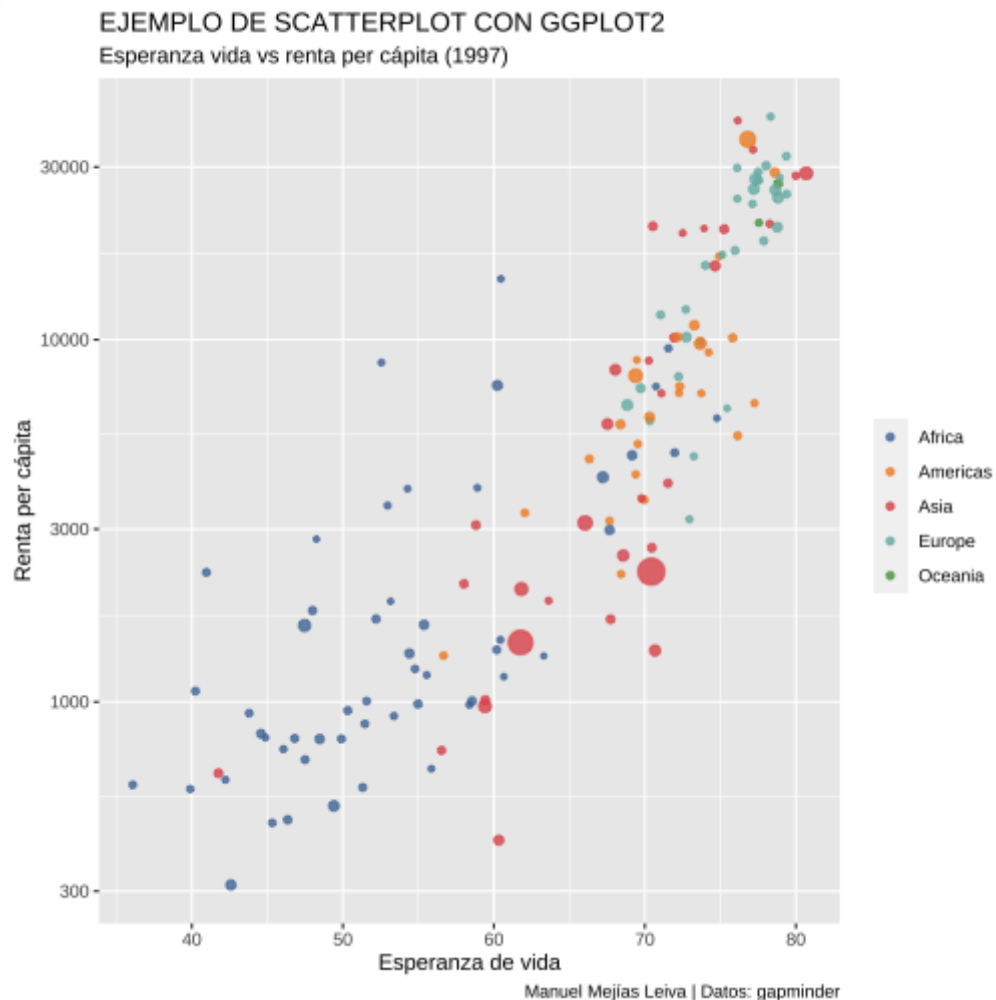
```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp,  
             color = continent, size = pop)) +  
  geom_point(alpha = 0.8) +  
  scale_y_log10() +  
  scale_color_tableau() +  
  labs(x = "Esperanza de vida",  
       y = "Renta per cápita",  
       color = NULL,  
       size = "Población",  
       title = "EJEMPLO DE SCATTERPLOT CON GGLOT2",  
       subtitle =  
         "Esperanza vida vs renta per cápita (1997)",  
       caption = "Manuel Mejías Leiva | Datos: gapminder")
```




Coordenadas y tema

Incluso podemos **ocultar la leyenda en sí de alguna de alguna de las variables** con `guides(size = "none")` (en este caso, `size = "none"` nos elimina la leyenda que codifica el tamaño de los puntos).


```
gapminder_1997 %>%  
  ggplot(aes(y = gdpPercap, x = lifeExp,  
             color = continent, size = pop)) +  
  geom_point(alpha = 0.8) +  
  scale_y_log10() +  
  scale_color_tableau() +  
  guides(size = "none") +  
  labs(x = "Esperanza de vida",  
       y = "Renta per cápita",  
       color = NULL, size = "Población",  
       title = "EJEMPLO DE SCATTERPLOT CON GGPLOT2",  
       subtitle =  
         "Esperanza vida vs renta per cápita (1997)",  
       caption = "Manuel Mejías Leiva | Datos: gapminder")
```



Ejercicios

-  **Ejercicio 1:** del conjunto `starwars` filtra solo los registros que no tenga ausente las columnas `mass`, `height`, `eye_color`. Dibuja un diagrama de puntos enfrentando `x = height` en el eje X e `y = mass` en el eje Y.

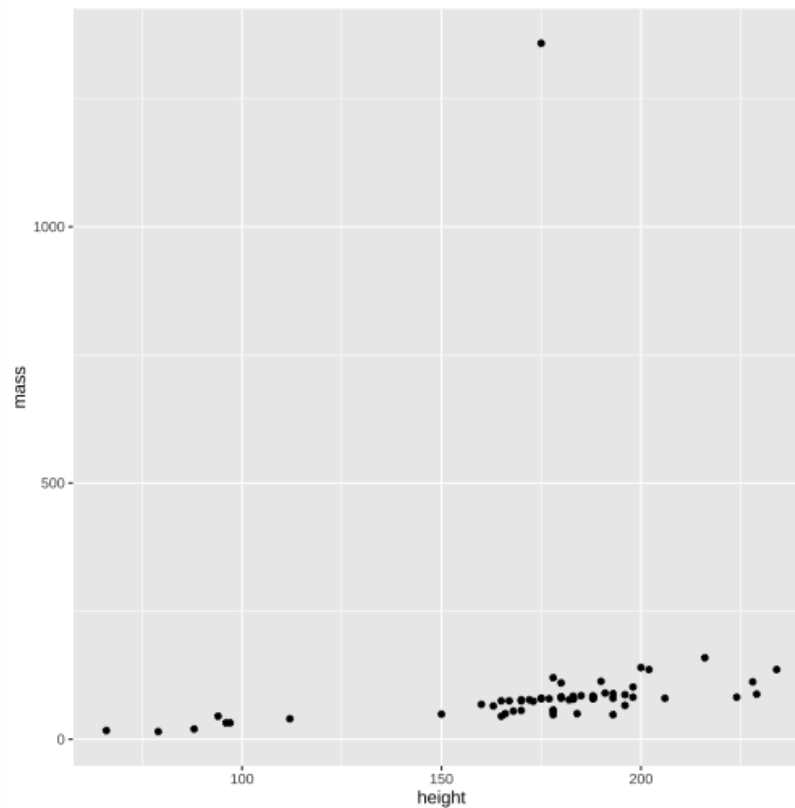
Ejercicios

-  **Ejercicio 1:** del conjunto **starwars** filtra solo los registros que no tenga ausente las columnas **mass**, **height**, **eye_color**. Dibuja un diagrama de puntos enfrentando **x = height** en el eje X e **y = mass** en el eje Y.

Sol. 1

```
# Eliminamos NA
starwars_filtro <- starwars %>%
  drop_na(c(mass, height, eye_color))

# Visualizamos
starwars_filtro %>%
  ggplot(aes(x = height, y = mass)) +
  geom_point()
```



Ejercicios

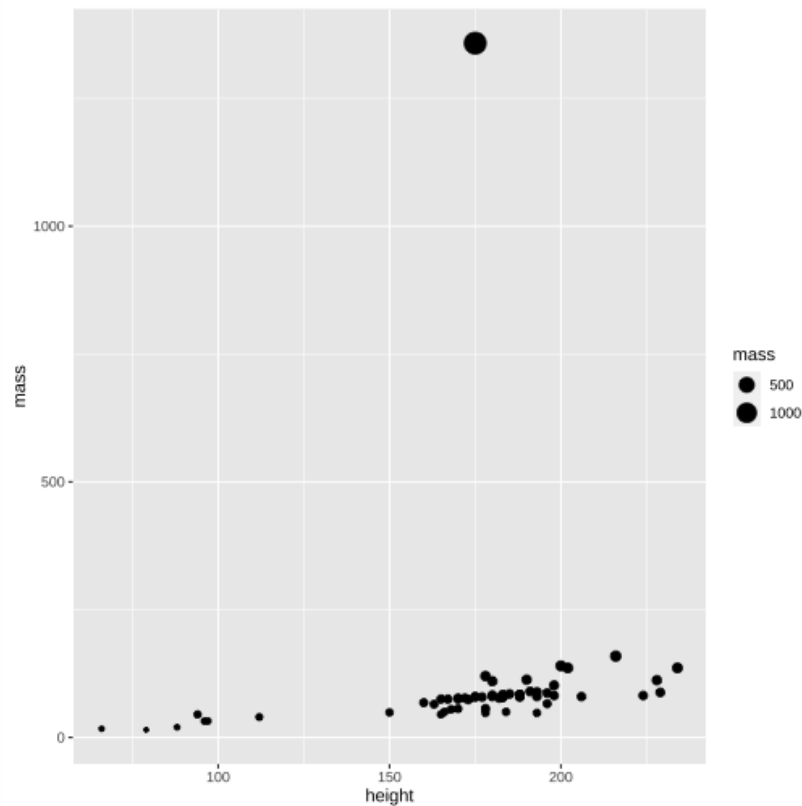
-  **Ejercicio 2:** modifica el código anterior para asignar el tamaño en función de `mass`.

Ejercicios

-  **Ejercicio 2:** modifica el código anterior para asignar el tamaño en función de **mass**.

Sol. 2

```
starwars_filtro %>%  
  ggplot(aes(x = height, y = mass, size = mass)) +  
  geom_point()
```



Ejercicios

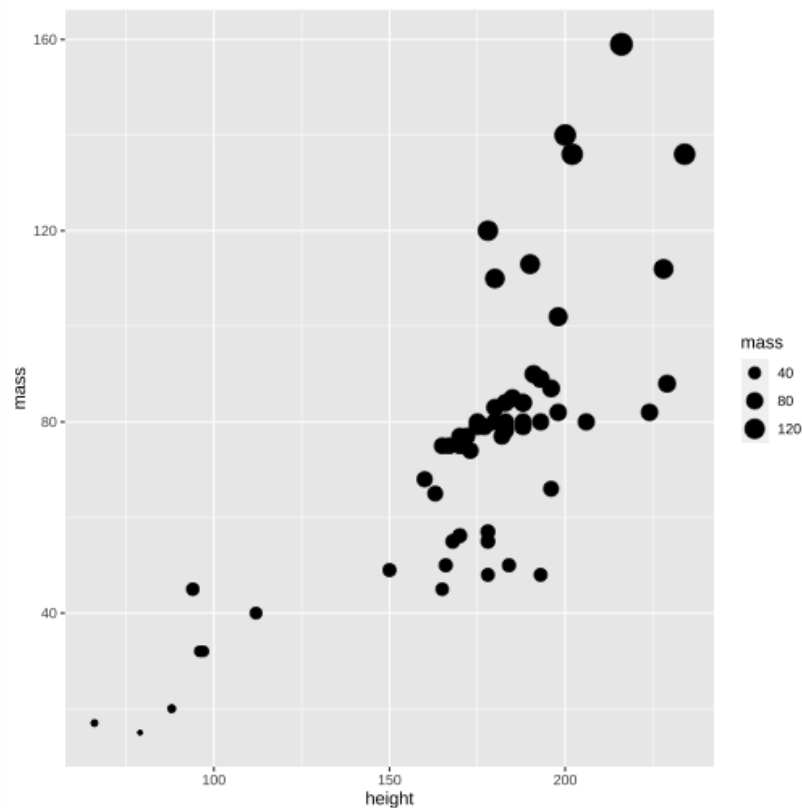
-  **Ejercicio 3:** repite el gráfico anterior localizando ese dato con un peso extremadamente elevado (outlier), elimínalo y vuelve a repetir la visualización.

Ejercicios


-  **Ejercicio 3:** repite el gráfico anterior localizando ese dato con un peso extremadamente elevado (outlier), elimínalo y vuelve a repetir la visualización.

Sol. 3


```
starwars_filtro %>%  
  filter(mass < 500)%>%  
  ggplot(aes(x = height, y = mass, size = mass)) +  
  geom_point()
```



Ejercicios

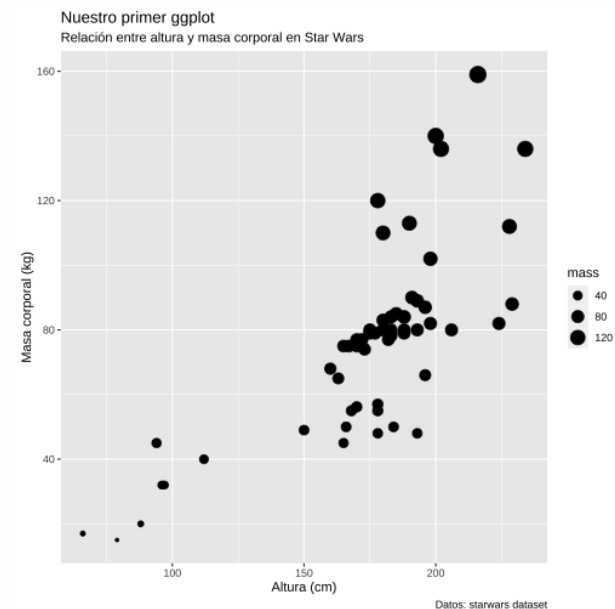
-  **Ejercicio 4:** repite el gráfico modificando títulos de ejes (a castellano) con título, subtítulo y caption. Por ejemplo:
 - Título: "Nuestro primer ggplot"
 - Subtítulo: "Relación entre altura y masa corporal en Star Wars"
 - Eje X: "Altura (cm)"
 - Eje Y: "Masa corporal (kg)"
 - Caption (fuente): "Datos: starwars dataset"

Ejercicios

-  **Ejercicio 4:** repite el gráfico modificando títulos de ejes (a castellano) con título, subtítulo y caption. Por ejemplo:
 - Título: "Nuestro primer ggplot"
 - Subtítulo: "Relación entre altura y masa corporal en Star Wars"
 - Eje X: "Altura (cm)"
 - Eje Y: "Masa corporal (kg)"
 - Caption (fuente): "Datos: starwars dataset"

Sol. 4

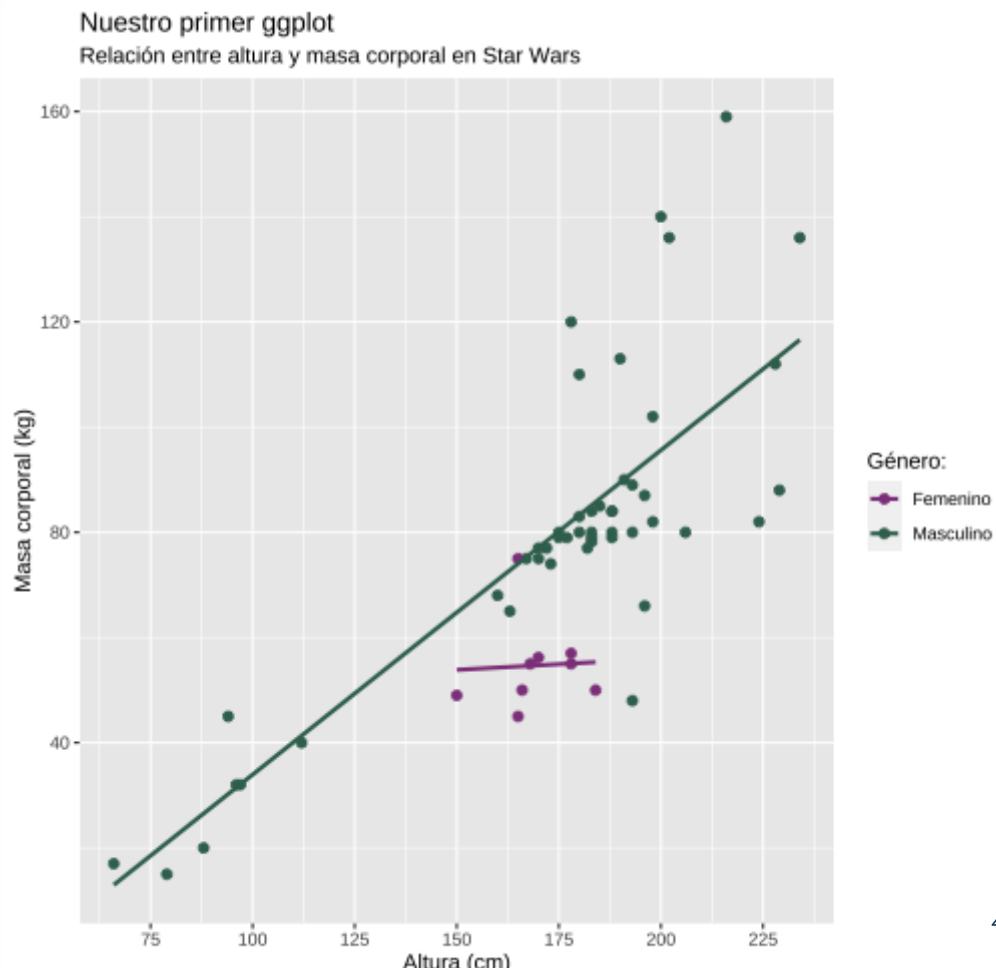
```
starwars_filtro %>%  
  filter(mass < 500)%>%  
  ggplot(aes(x = height, y = mass, size = mass)) +  
  geom_point() +  
  labs(x = "Altura (cm)",  
       y = "Masa corporal (kg)",  
       title = "Nuestro primer ggplot",  
       subtitle =  
"Relación entre altura y masa corporal en Star Wars",  
       caption = "Datos: starwars dataset")
```



Bonus track: geom_smooth()

La función `geom_smooth()` en `ggplot2` sirve para **agregar una línea suave (o de tendencia)** en un gráfico y mostrar la tendencia general de los datos.

```
starwars %>%  
  filter(mass < 500,  
         !is.na(gender))%>%  
  ggplot(aes(x = height, y = mass, color = gender))+  
  geom_point(size = 2)+  
  geom_smooth(se= FALSE, method = "lm")+  
  labs(x = "Altura (cm)",  
       y = "Masa corporal (kg)",  
       color = "Género:",  
       title = "Nuestro primer ggplot",  
       subtitle =  
         "Relación entre altura y masa corporal en Star Wars",  
       caption = "Datos: starwars dataset")+  
  scale_x_continuous(breaks = seq(50, 250, by = 25))+  
  scale_color_manual(labels = c("Femenino", "Masculino"),  
                    values = c("#8F448B", "#3A705D"))
```

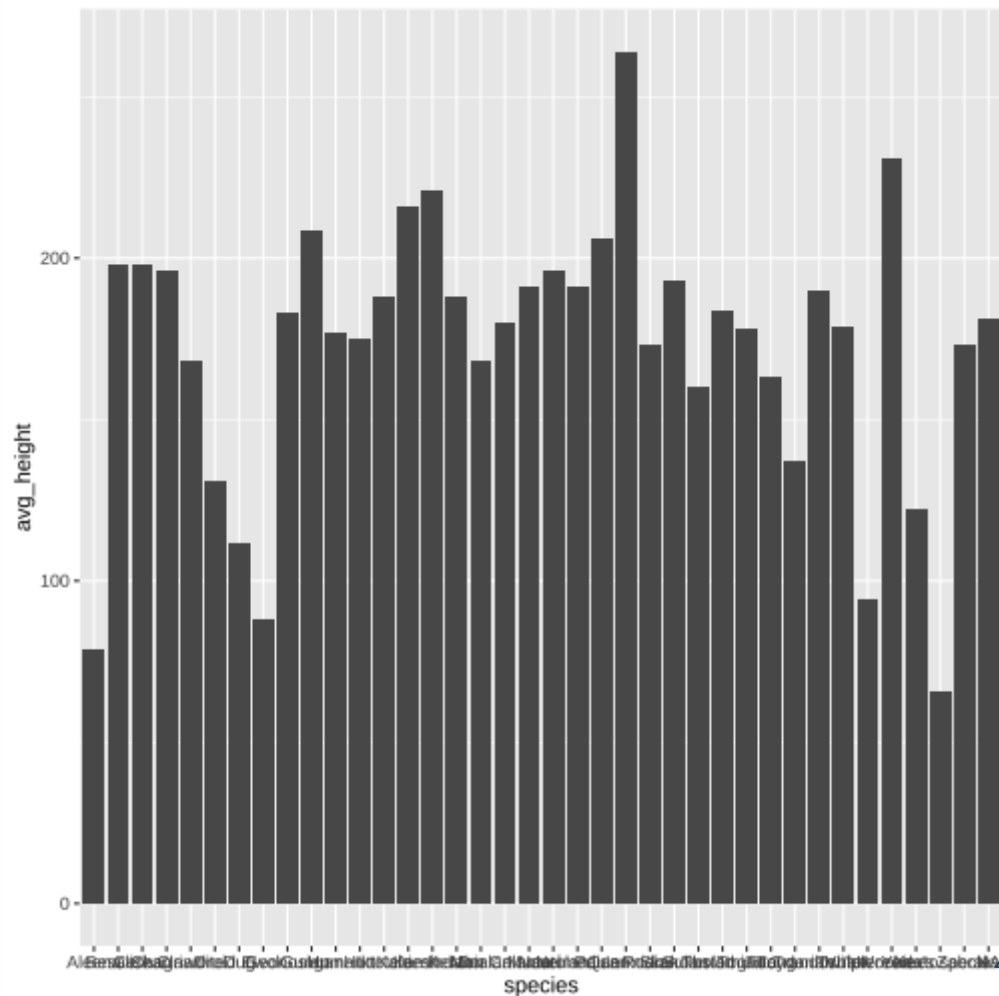


Más geom: gráfico de barras

Vamos a representar en un gráfico de barras la altura media por especie.

✗ No se ve NADA!

```
starwars %>%  
  group_by(species) %>%  
  summarize(avg_height = mean(height, na.rm = TRUE)) %>%  
  ggplot(aes(x= species, y= avg_height))+  
  geom_col()
```

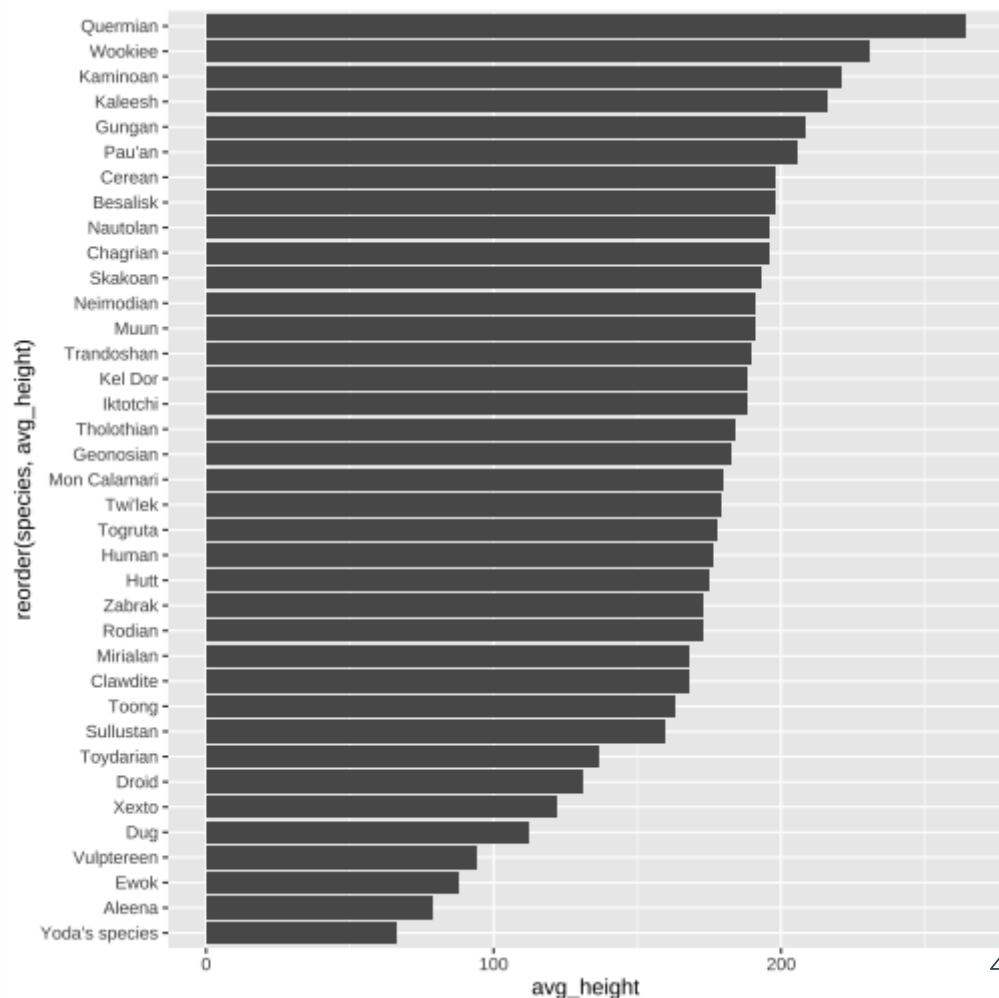


Más geom: gráfico de barras

¿Podemos hacer que esto sea legible?

- La función **reorder** se utiliza para cambiar el orden de las categorías en un gráfico de barras. Permite ordenar las categorías según una variable numérica o categórica, lo que facilita la interpretación de los datos en el gráfico.

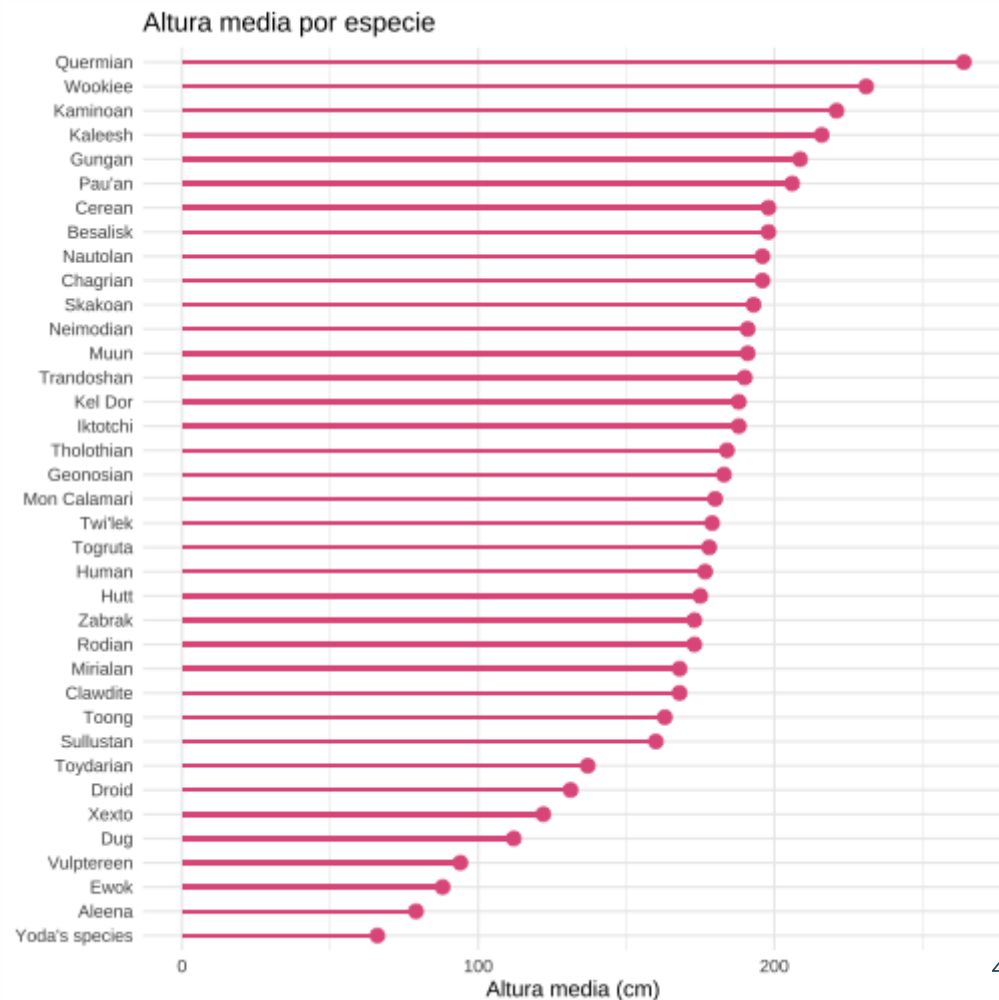
```
starwars %>%  
  drop_na(species)%>%  
  group_by(species)%>%  
  summarize(avg_height = mean(height, na.rm = TRUE))%>%  
  ggplot(aes(x= avg_height,  
             y= reorder(species, avg_height)))+  
  geom_col()
```



Más geom: gráfico de barras

¿Qué más podemos añadir?

```
starwars %>%  
  drop_na(species)%>%  
  group_by(species)%>%  
  summarize(avg_height = mean(height, na.rm = TRUE))%>%  
  ggplot(aes(x= avg_height,  
            y= reorder(species, avg_height)))+  
  geom_col(fill = "#E15D8A", width = .2)+  
  geom_point(size = 3, color = "#E15D8A")+  
  labs(y = NULL,  
       x = "Altura media (cm)",  
       title = "Altura media por especie")+  
  theme_minimal()
```



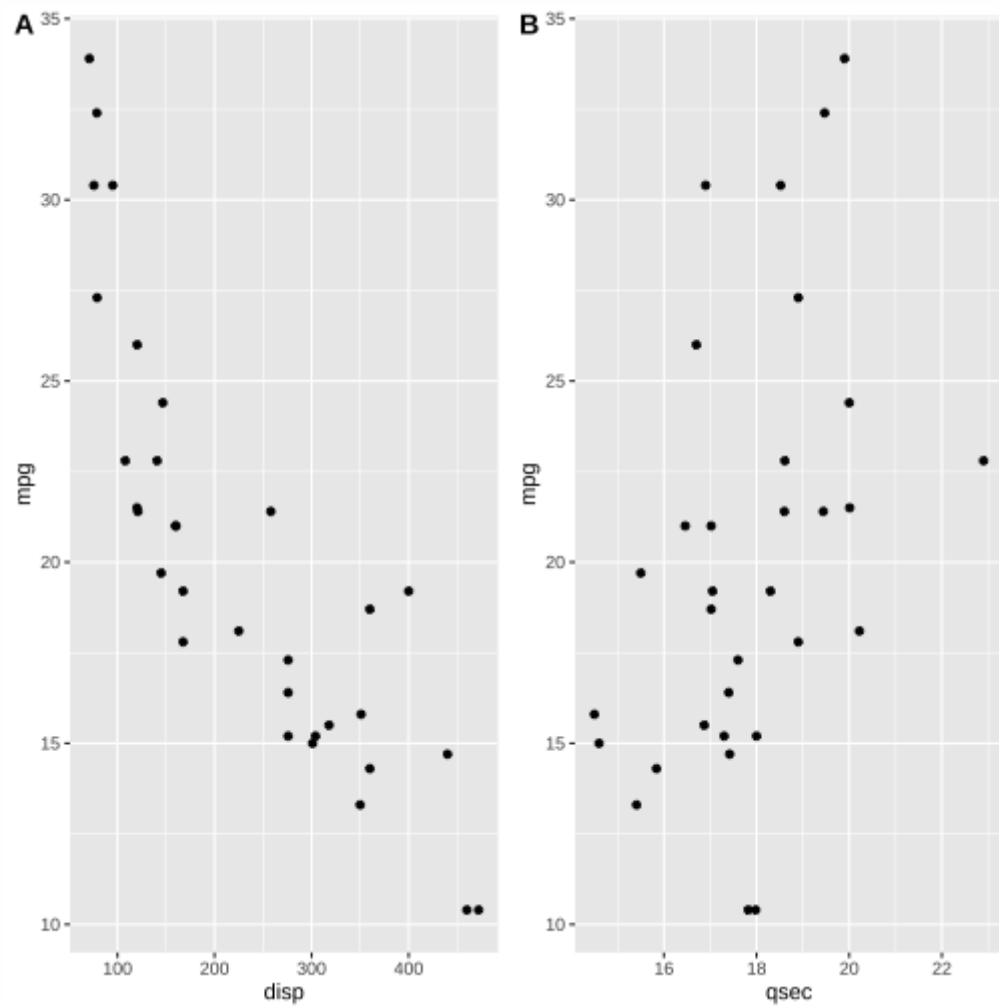
Combinando gráficos con `plot_grid`

La función **`plot_grid`** es utilizada para combinar y mostrar varios gráficos en una sola figura.

```
library(cowplot)

p1 <- ggplot(mtcars, aes(dis, mpg)) +
  geom_point()
p2 <- ggplot(mtcars, aes(qsec, mpg)) +
  geom_point()

plot_grid(p1, p2, labels = c('A', 'B'))
```

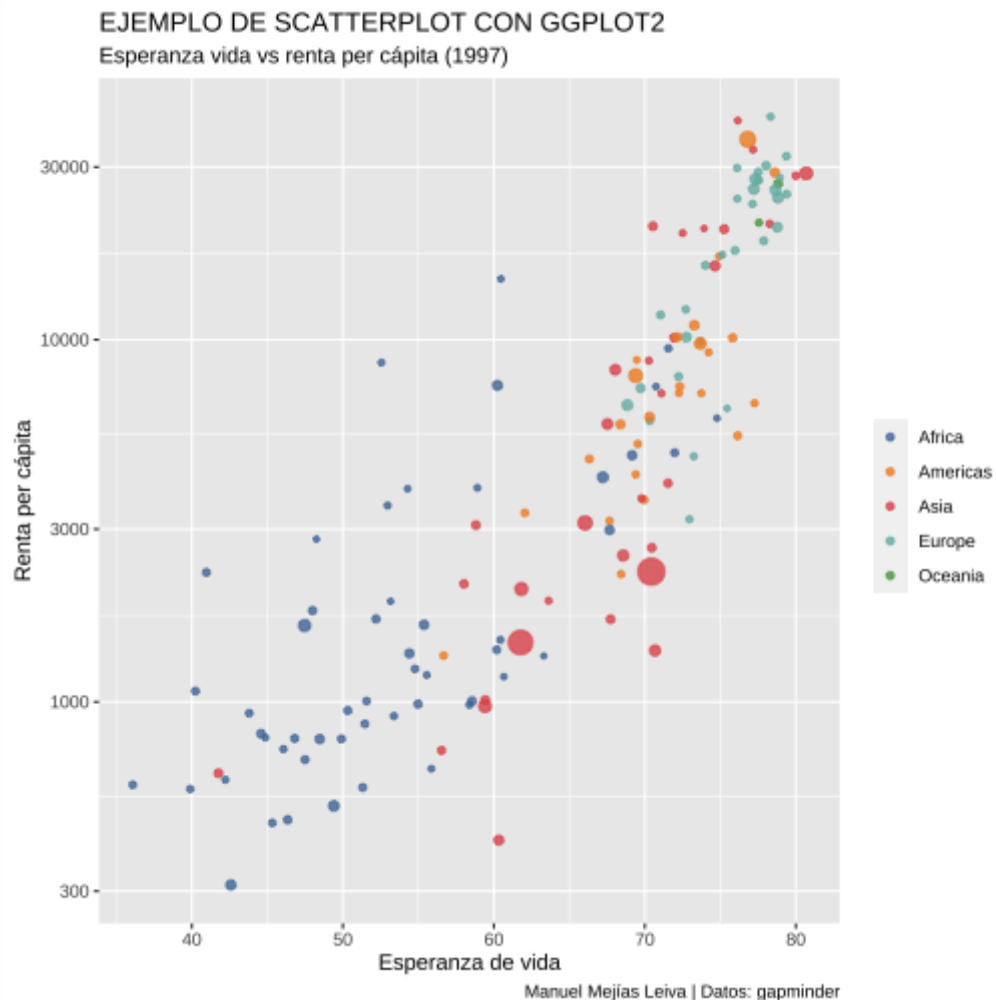


Exportar gráficos con ggsave

```
# 1. Crea un gráfico y asígnalo a un objeto
plot_ggsave <- gapminder_1997 %>%
  ggplot(aes(y = gdpPercap, x = lifeExp,
             color = continent, size = pop)) +
  geom_point(alpha = 0.8) +
  scale_y_log10() +
  scale_color_tableau() +
  guides(size = "none") +
  labs(x = "Esperanza de vida",
       y = "Renta per cápita",
       color = NULL, size = "Población",
       title = "EJEMPLO DE SCATTERPLOT CON GGLOT2",
       subtitle =
         "Esperanza vida vs renta per cápita (1997)",
       caption = "Manuel Mejías Leiva | Datos: gapminder")
```

Guarda el gráfico como un archivo PNG en la carpeta de trabajo

```
ggsave("primer_plot.png", plot_ggsave)
```



Recursos para Dataviz con ggplot

Recursos para Dataviz con ggplot

- Todas las geoms disponibles: <https://ggplot2.tidyverse.org/reference/>
- The R Graph Gallery: <https://r-graph-gallery.com/>