Visualización de datos con {ggplot2} I

Visualizar datos es útil para identificar a relación entre distintas variables pero también para comunicar el análisis de los datos y resultados. El paquete `ggplot2` permite generar gráficos de gran calidad en pocos pasos. Cualquier gráfico de ggplot tendrá como mínimo 3 componentes: los **datos**, un **sistema de coordenadas** y una **geometría** (la representación visual de los datos) y se irá construyendo por capas.

Primera capa: el área del gráfico

Cómo siempre será necesario cargar los paquetes que vamos a usar y ya que estamos los datos con los que venimos trabajando:

```
library(readr)
library(ggplot2)
paises <- read_csv(here::here("datos", "paises.csv"))
```

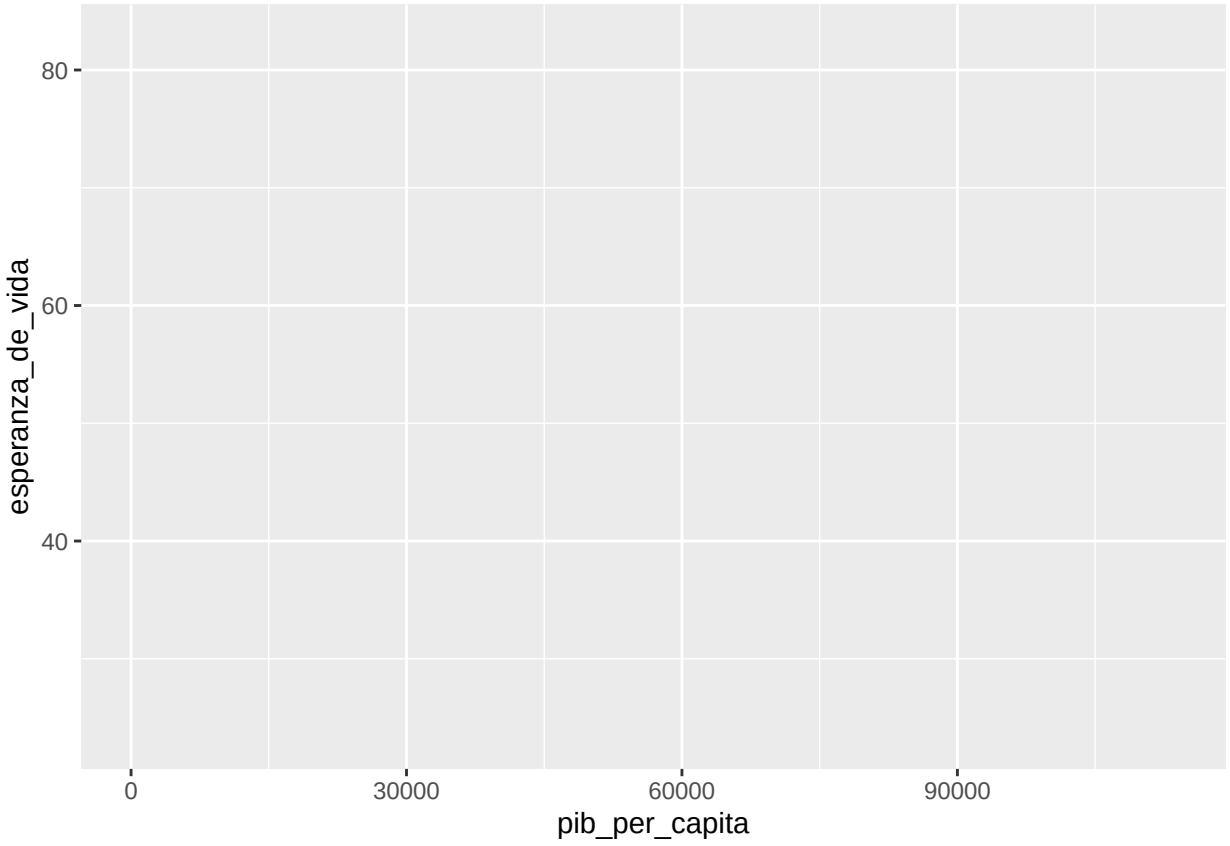
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   pais = col_character(),
##   continente = col_character(),
##   anio = col_double(),
##   esperanza_de_vida = col_double(),
##   poblacion = col_double(),
##   pib_per_capita = col_double()
## )
```

La función principal de ggplot2 es justamente `ggplot()` que nos permite *iniciar* el gráfico y además definir las características *globales*. El primer argumento de esta función serán los datos que queremos visualizar, siempre en un data.frame. En este caso usamos `paises`.

El segundo argumento se llama mapping justamente porque *mapea* o *dibuja* los ejes del gráfico y **siempre** va acompañado de la función `aes()`. La función `aes()` recibe las propiedades estéticas del gráfico (o *aesthetic* en inglés) a partir de las variables (o columnas) del data.frame estamos usando. En este caso le indicamos que en el eje **x** querremos graficar la variable `pib_per_capita` y en eje **y** la variable `esperanza_de_vida`.

Pero esta sola función no es suficiente, solo genera la primera capa: el área del gráfico.

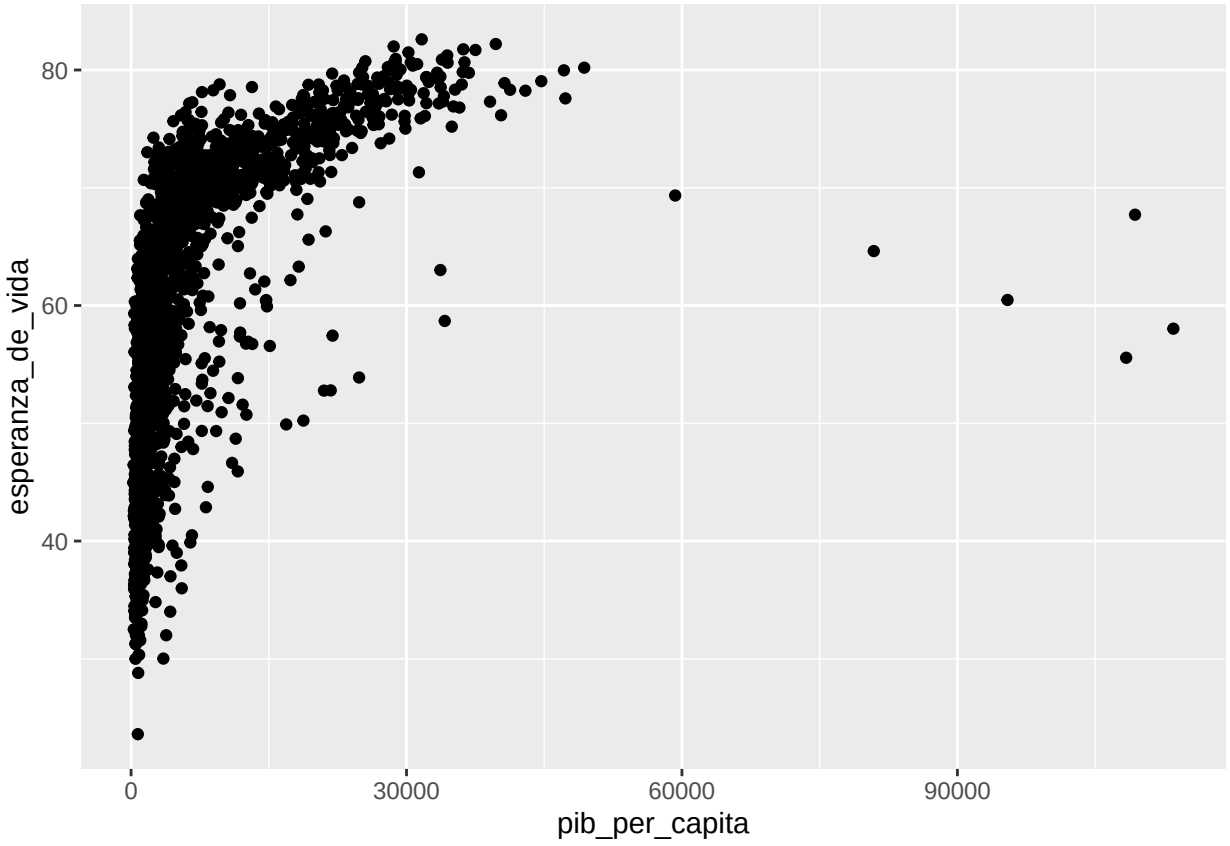
```
ggplot(data = paises, mapping = aes(x = pib_per_capita, y = esperanza_de_vida))
```



Segunda capa: geometrías

Necesitamos agregar una nueva capa a nuestro gráfico, los elementos geométricos o *geoms* que representaran los datos. Para esto *sumamos* una función geom, por ejemplo si queremos representar los datos con puntos usaremos `geom_point()`

```
ggplot(data = paises, mapping = aes(x = pib_per_capita, y = esperanza_de_vida)) +
  geom_point()
```



¡Nuestro primer gráfico!

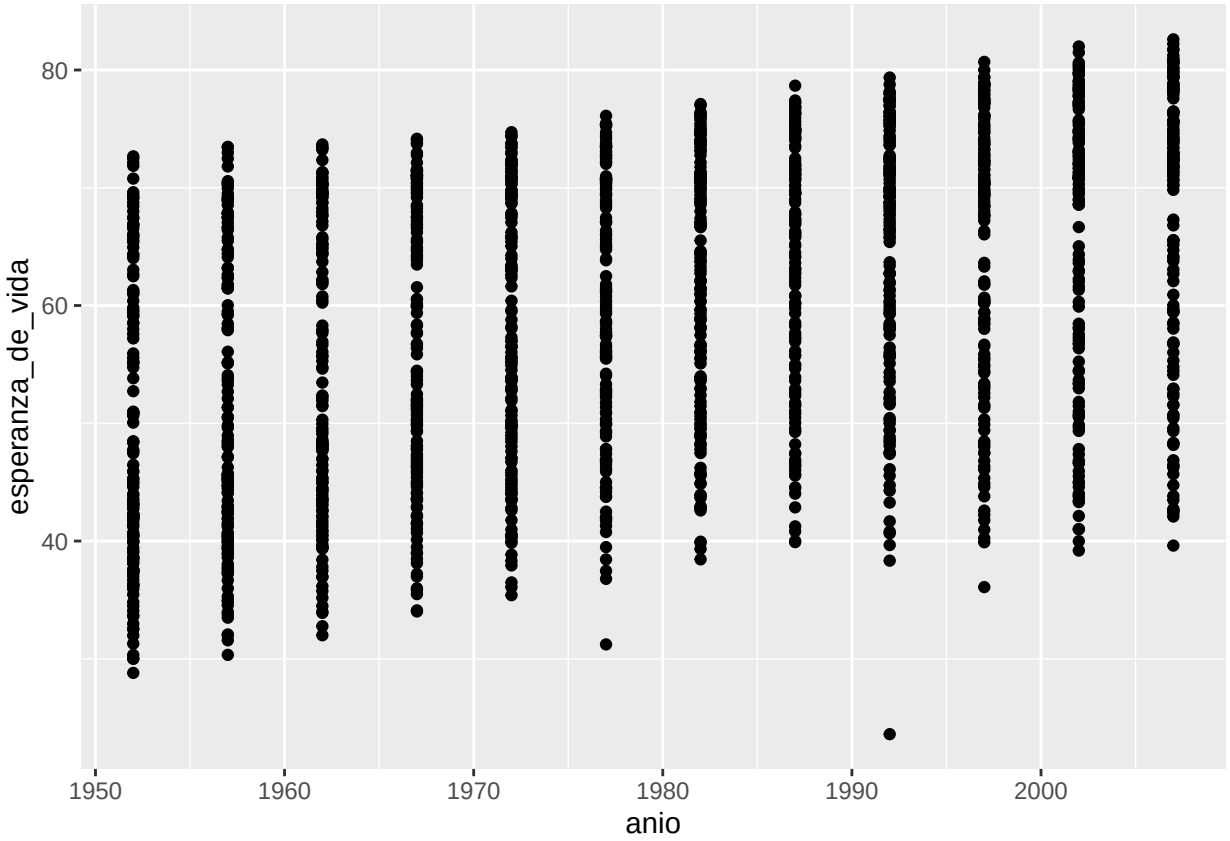
Primer desafío

Ahora es tu turno. Modifica el gráfico anterior para visualizar > cómo cambia la esperanza de vida a lo largo de los años.

¿Te parece útil este gráfico? ***

Este gráfico tiene muchísima información porque tiene un punto por cada país para cada año para visualizar la esperanza de vida. Pero por ahora no podemos identificar esos países, necesitamos agregar información al gráfico.

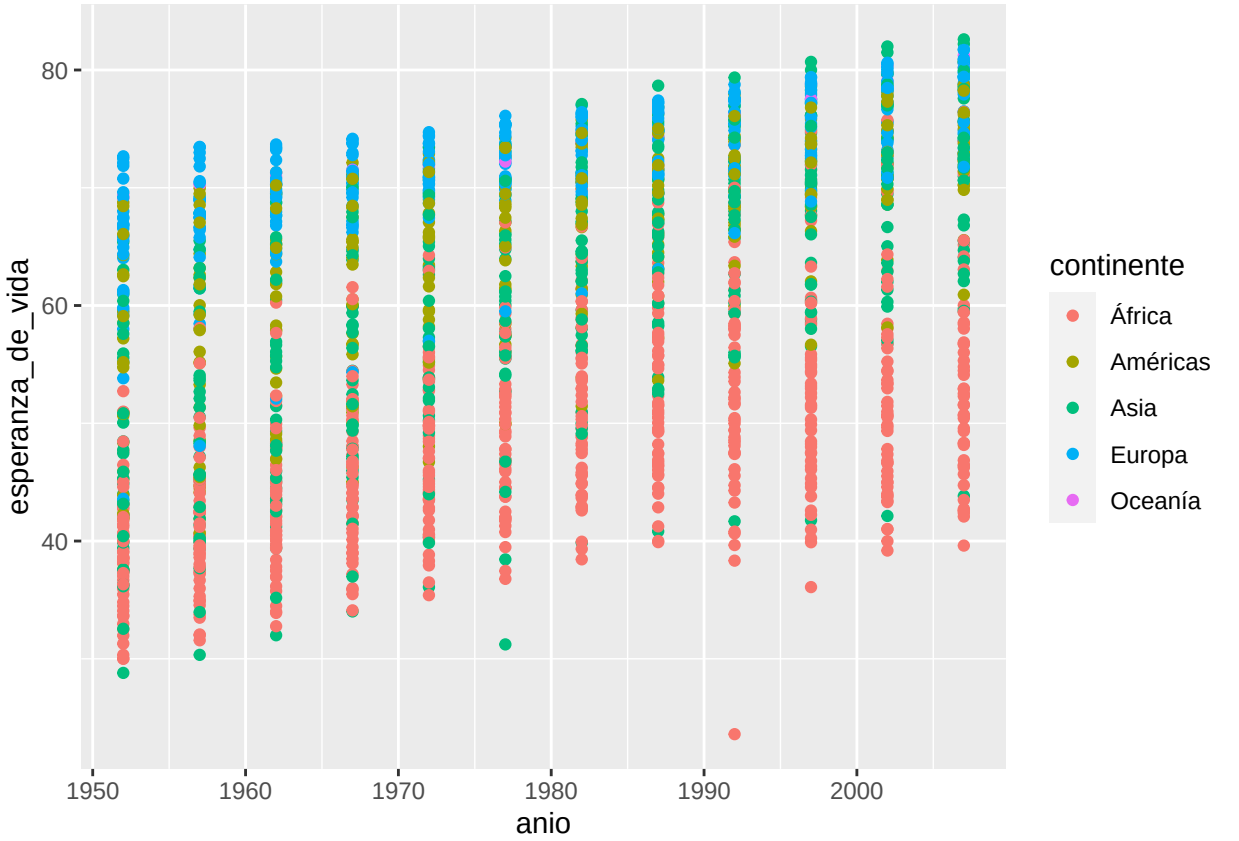
```
ggplot(data = paises, mapping = aes(x = anio, y = esperanza_de_vida)) +
  geom_point()
```



Mapear variables a elementos

Una posible solución sería utilizar otras variables de nuestros datos, por ejemplo `continente` y *mapear* el color de los puntos de a cuerdo al continente que pertenecen.

```
ggplot(data = paises, mapping = aes(x = anio, y = esperanza_de_vida)) +
  geom_point(aes(color = continente))
```



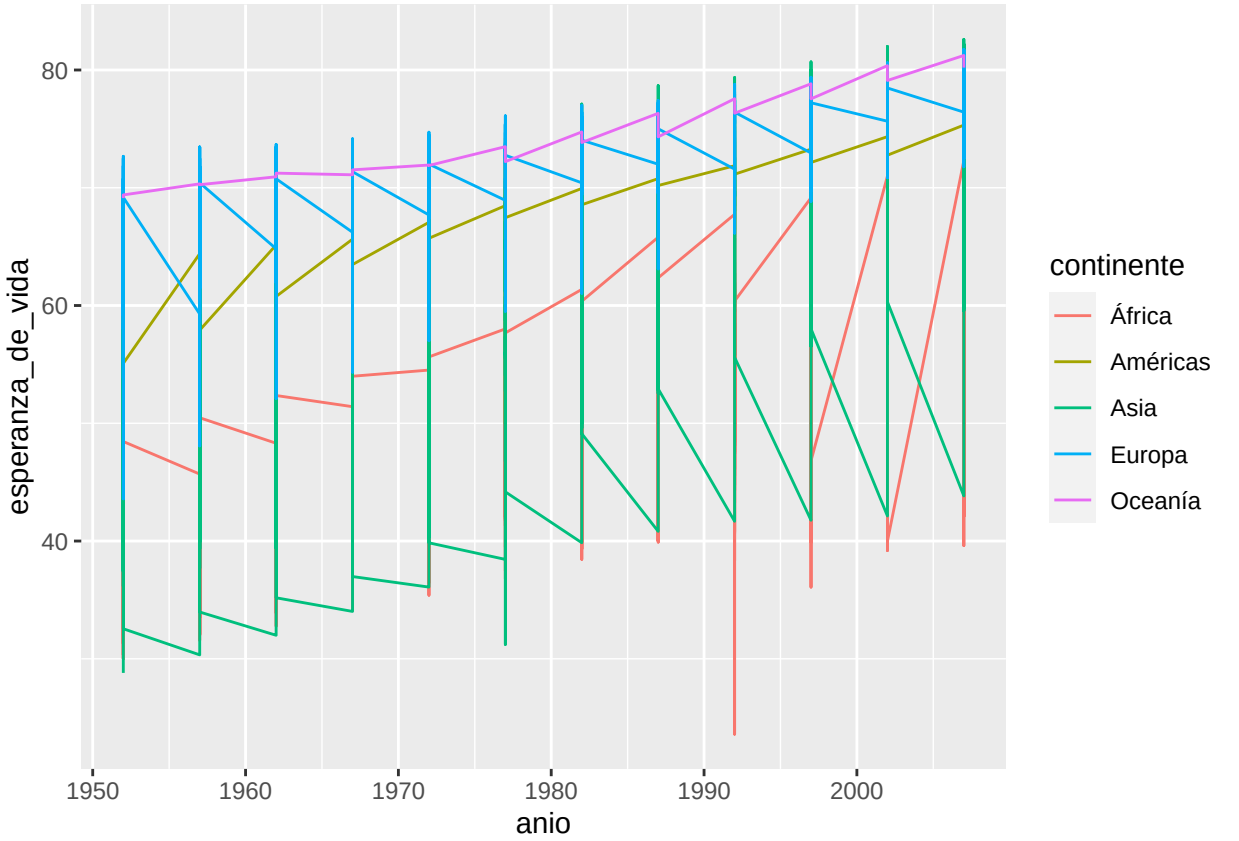
Ahh, ahora está un poco mejor. Por ejemplo ya podemos ver que muchos países de Europa (los puntos celestes) tienen en promedio mayor esperanza de vida a lo largo de los años que muchos países de África (los puntos rojos). Aún no podemos identificar los países individualmente pero podemos sacar algo más de información de nuestros datos.

Algo muy importante a tener en cuenta: **los puntos toman un color de acuerdo a una variable de los datos**, y para que ggplot2 identifique esa variable (en este caso `continente`) es necesario incluirla dentro de una función `aes()`.

Otras geometrías

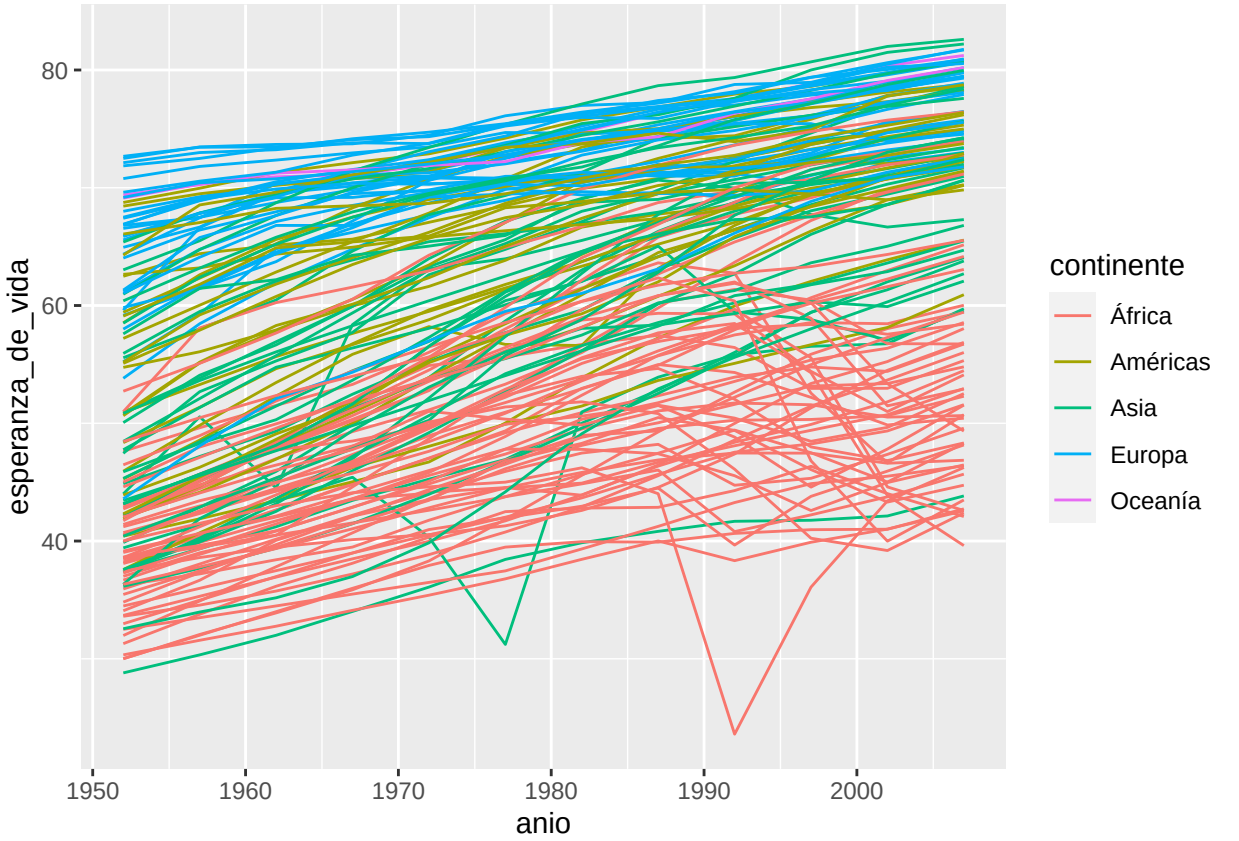
Este gráfico posiblemente no sea muy adecuado si queremos visualizar la *evolución* de una variable a lo largo del tiempo, necesitamos cambiar la geometría a lineas usando `geom_line()`

```
ggplot(data = paises, mapping = aes(x = anio, y = esperanza_de_vida)) +
  geom_line(aes(color = continente))
```



Por suerte las funciones `geom_*()` tienen más o menos nombres amigables. Pero el gráfico sigue teniendo problemas, al parecer dibujó una línea por continente. Si estuviéramos dibujando este gráfico con lápiz y papel muy posiblemente hubiéramos identificado los puntos que corresponden a cada país y los hubiéramos "unido con líneas", necesitamos que ggplot2 haga esto. ¿Cómo le indicamos que puntos corresponde a cada país? Necesitamos que los *agrupe* por la variable `pais` (¡qué bueno que tenemos toda esa información en nuestra base de datos!).

```
ggplot(data = paises, mapping = aes(x = anio, y = esperanza_de_vida)) +
  geom_line(aes(color = continente, group = pais))
```



Usamos el argumento `group =` y de nuevo, lo incluimos dentro de la función `aes()` para indicarle a ggplot2 que busque la variable `pais` dentro del data.frame que estamos usando.

Y ahora si, conseguimos el gráfico que estamos buscando.

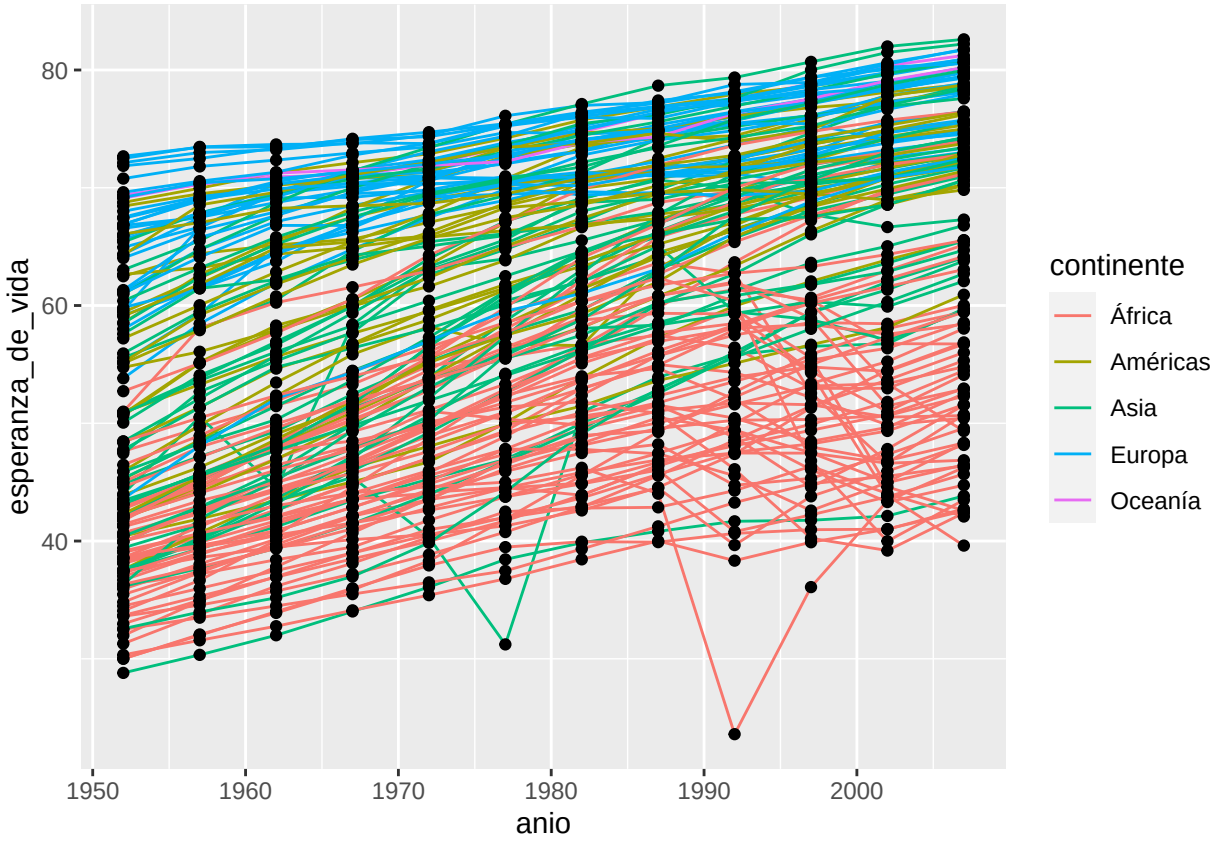
Segundo desafío

Cuando mencionamos que ggplot2 construye gráficos por capas, lo decíamos en serio! Hasta ahora tenemos dos capas: el área del gráfico y una geometría (las líneas).

1. Sumá una tercera capa para visualizar puntos además de las líneas.
2. ¿Porqué los puntos ahora no siguen los colores de los continentes?
3. ¿Qué cambio podrías hacer para que los puntos también tengan color según el continente?
   ***

Acá surge una característica importante de las capas: pueden tener apariencia independiente si solo *mapeamos* el color en la capa de las líneas y no en la capa de los puntos. Al mismo tiempo, si quisiéramos que todas las capas tenga la misma apariencia podemos incluir el argumento `color =`en la función global `ggpplot()` o repetirlo en cada capa.

```
ggplot(paises, aes(anio, esperanza_de_vida)) +
  geom_line(aes(color = continente, group = pais)) +
  geom_point()
```



Si te preguntabas a donde fueron a parar el `data =`, el `mapping =` y los nombres de los argumentos adentro de la función `aes()`, `x =` e `y =`, resulta que estamos aprovechando que tanto ggplot2 como nosotros ahora sabemos en que orden recibe la información cada función. Siempre el primer elemento que le *pasemos* o indiquemos a la función `ggplot()` será el data.frame.

Algunos argumentos para cambiar la apariencia de las geometrías son:

- `color` o `colour` modifica el color de líneas y puntos
- `fill`modifica el color del área de un elemento, por ejemplo el relleno de un punto
- `linetype` modifica el tipo de línea (punteada, continua, con guiones, etc.)
- `pch` modifica el tamaño del punto
- `size` modifica el tamaño de los elementos (por ejemplo el tamaño de puntos o el grosor de líneas)
- `alpha` modifica la transparencia de los elementos (1 = opaco, 0 = transparente)
- `shape` modifica el tipo de punto (círculos, cuadrados, triángulos, etc.)

El *mapeo* entre una variable y un parámetro de geometría se hace a través de una **escala**. La escala de colores es lo que define, por ejemplo, que los puntos donde la variable `continente` toma el valor `"África"` van a tener el color rosa, donde toma el valor `"Américas"`, mostaza, etc...

Modificar elementos utilizando un valor único

Es posible que en algún momento necesites cambiar la apariencia de los elementos o geometrías independientemente de las variables de tu data.frame. Por ejemplo podrías querer que todos los puntos sean de un único color: rojos. En este caso

`geom_point(aes(color = "red"))` no va a funcionar -ojo que los colores van en inglés-. Lo que ese código dice es que mapee el parámetro geométrico "color" a una variable que contiene el valor `"red"` para todas las filas. El mapeo se hace a través de la escala, que va a asignarle un valor rosa/rojo a los puntos correspondientes al valor `"red"`.

Ahora que no nos interesa *mapear* el color a una variable, podemos mover ese argumento **afuera** de la función `aes()`: `geom_point(color = "red")`.

Relación entre variables

Muchas veces no es suficiente con mirar los datos crudos para identificar la relación entre las variables; es necesario usar alguna transformación estadística que resalte esas relaciones, ya sea ajustando una recta o calculando promedios.

Para alguna transformaciones estadísticas comunes, {ggplot2} tiene geoms ya programados, pero muchas veces es posible que necesitemos manipular los datos antes de poder hacer un gráfico. A veces esa manipulación será compleja y entonces para no repetir el cálculo muchas veces, guardaremos los datos modificados en una nueva variable. Pero también podemos *encadenar* la manipulación de los datos y el gráfico resultante.

Por ejemplo, calculemos la esperanza de vida media por continente y para cada año usando `dplyr` y luego grafiquemos la `esperanza_de_vida_media` a los largo de los `anios`:

```
library(dplyr)
```

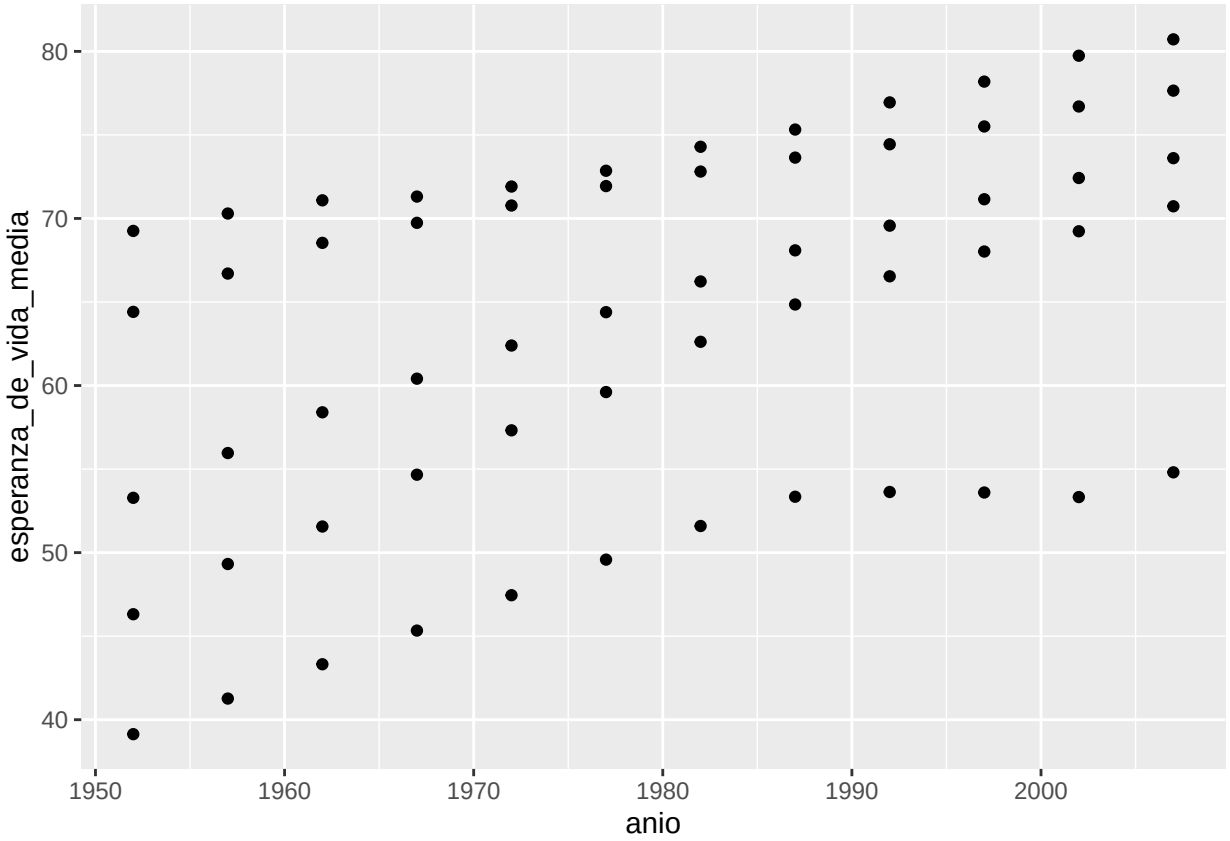
```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
paises %>%
group_by(continente, anio) %>%
summarise(esperanza_de_vida_media = mean(esperanza_de_vida)) %>%
ggplot(aes(anio, esperanza_de_vida_media)) +  # Acá se acaban los %>% y comienzan los "+"
geom_point()
```

```
## `summarise()` regrouping output by 'continente' (override with `.groups` argument)
```



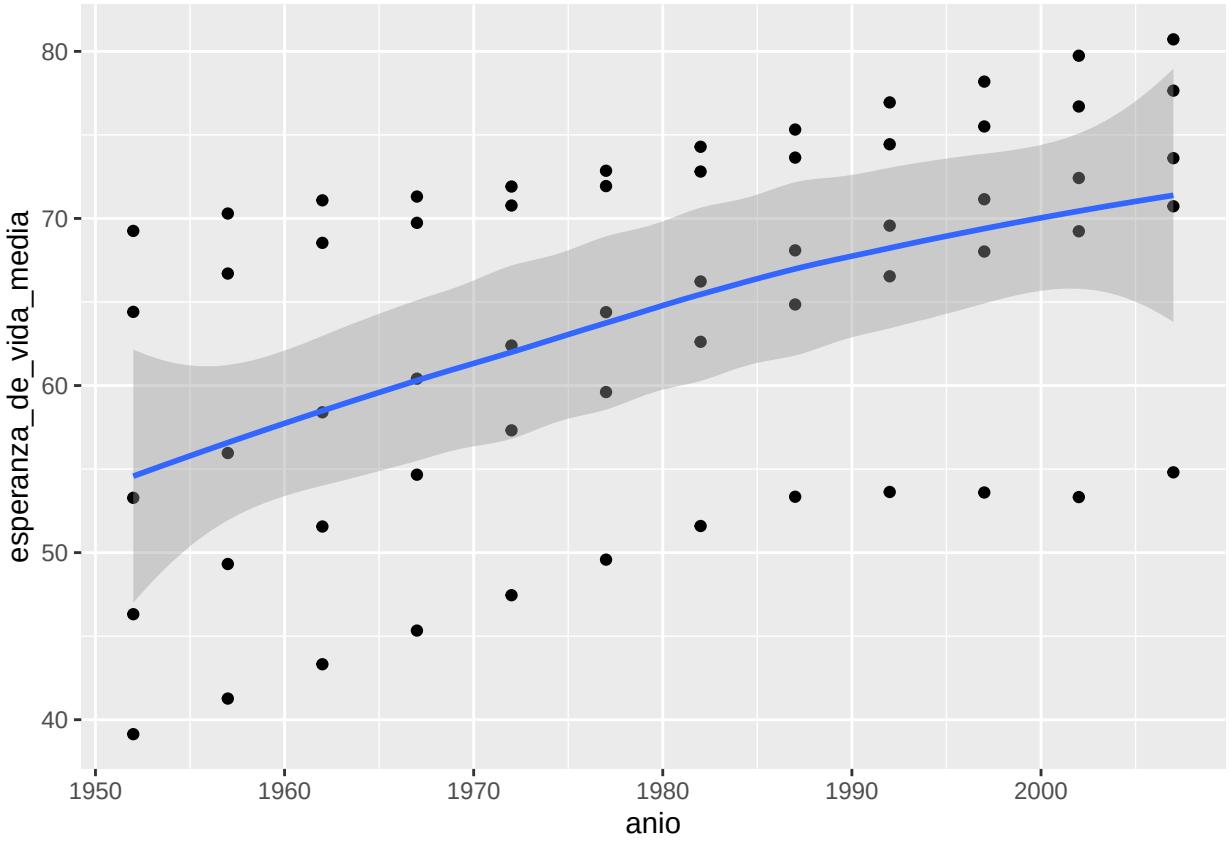
Esto es posible gracias al operador `%>%` que le *pasa* el resultado de `summarise()` a la función `ggplot()`. Y este resultado no es ni más ni menos que el data.frame que necesitamos para hacer nuestro gráfico. Es importante notar que una vez que comenzamos el gráfico ya **no** se puede usar el operador `%>%` y las capas del gráfico se *suman* como siempre con `+`.

Este gráfico entonces parece mostrar que la esperanza de vida fue aumentado a lo largo de los años pero sería interesante ver esa relación más explícitamente agregando una nueva capa con `geom_smooth()`.

```
paises %>%
group_by(continente, anio) %>%
summarise(esperanza_de_vida_media = mean(esperanza_de_vida)) %>%
ggplot(aes(anio, esperanza_de_vida_media)) +
geom_point() +
geom_smooth()
```

```
## `summarise()` regrouping output by 'continente' (override with `.groups` argument)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

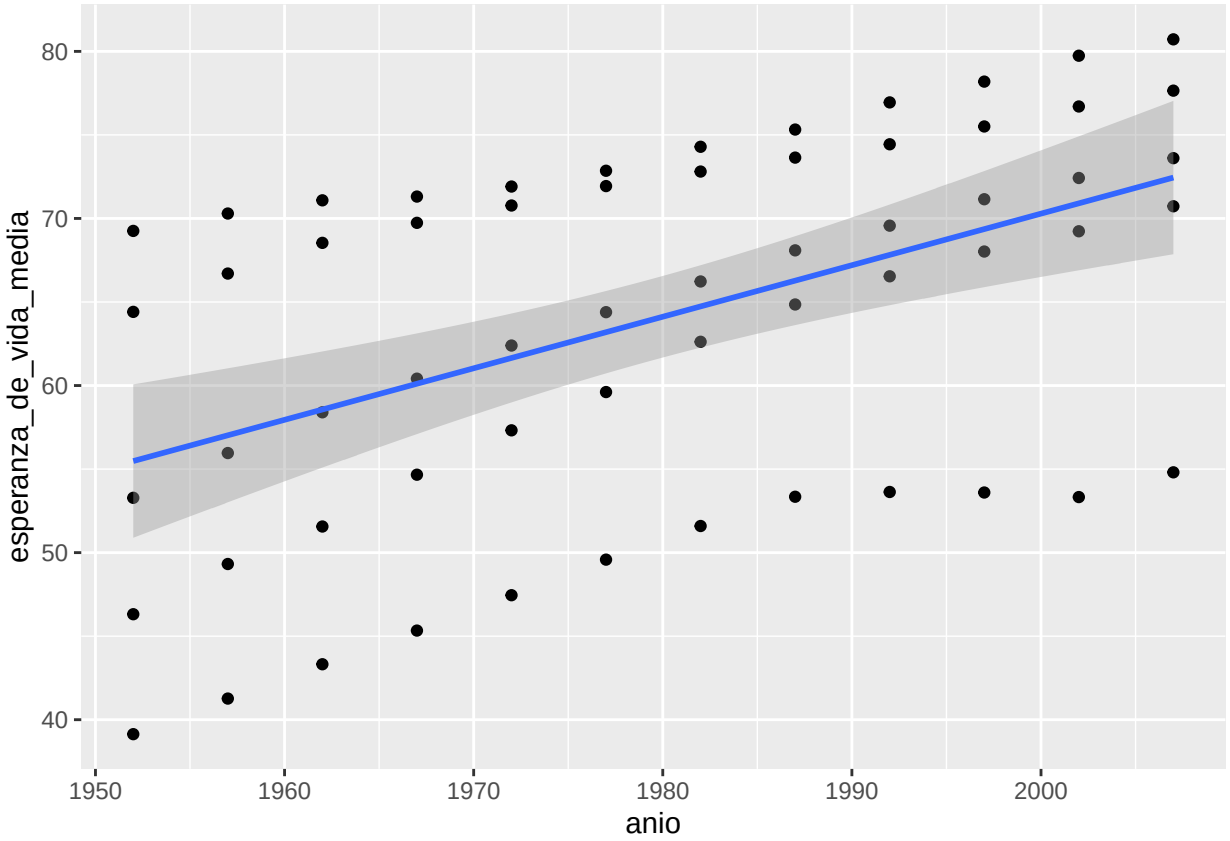


Como dice en el mensaje, por defecto `geom_smooth()` suaviza los datos usando el método *loess* (regresión lineal local) cuando hay menos de 1000 datos. Seguramente va a ser muy común que quieras ajustar una regresión lineal global. En ese caso, hay que poner `method = "lm"`:

```
paises %>%
group_by(continente, anio) %>%
summarise(esperanza_de_vida_media = mean(esperanza_de_vida)) %>%
ggplot(aes(anio, esperanza_de_vida_media)) +
geom_point() +
geom_smooth(method = "lm")
```

```
## `summarise()` regrouping output by 'continente' (override with `.groups` argument)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



En gris nos muestra el intervalo de confianza al rededor de este suavizado que en este caso es bastante grande porque tenemos pocos datos!

Cómo cualquier geom, podemos modificar el color, el grosor de la línea y casi cualquier cosa que se te ocurra.

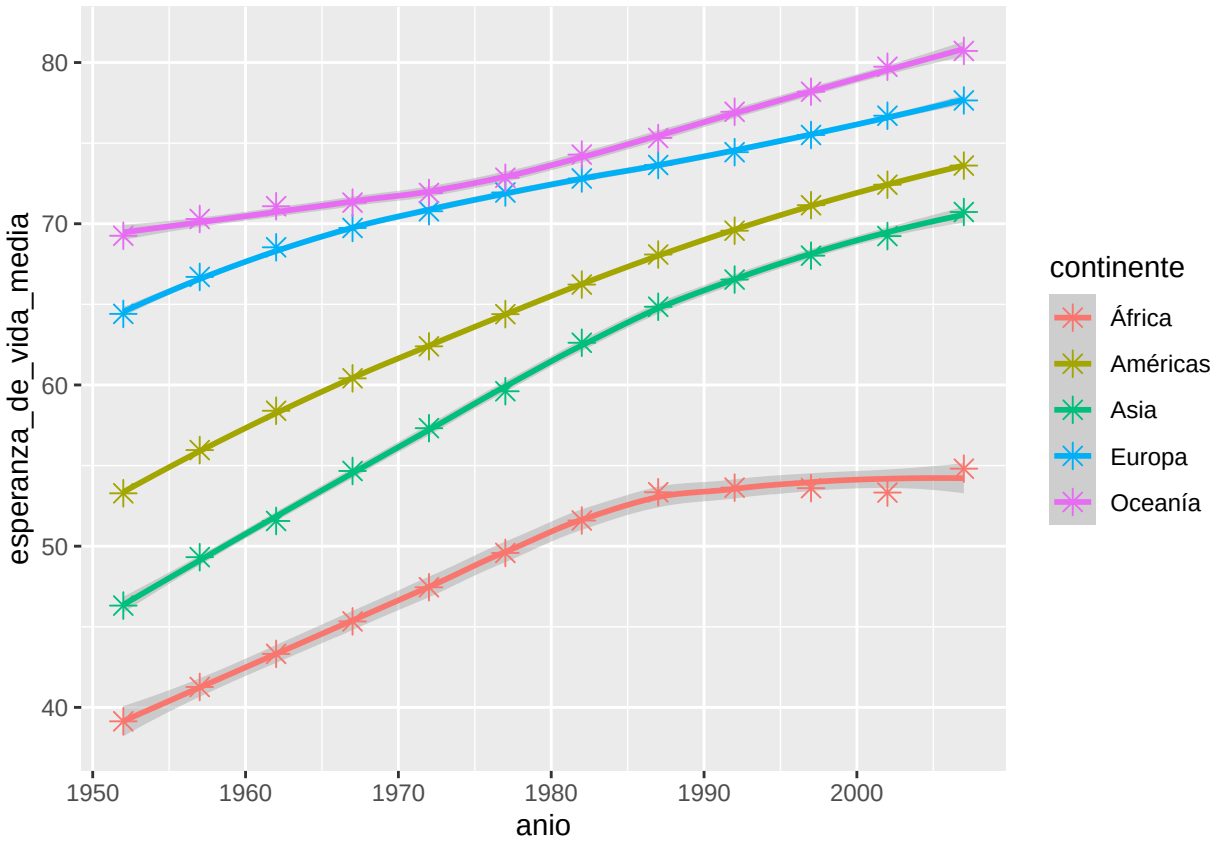
Tercer desafío

Modificá el siguiente código para obtener el gráfico que se muestra más abajo.

```
paises %>%
group_by(continente, _____) %>%
summarise(esperanza_de_vida_media = mean(esperanza_de_vida)) %>%
ggplot(aes(anio, _________________)) +
geom_point(aes(color = continente), size = 3, shape = _____) +
geom_smooth(color = continente)
```

```
## `summarise()` regrouping output by 'continente' (override with `.groups` argument)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



***

Fuente: Estos apuntes forman parte del curso de Excel a R paocorrales.github.io/deExcelaR de Elio Campitelli y Paola Corrales con licencia Creative Commons Attribution-ShareAlike 4.0.