Introducción a Tidyverse

Rubén Sierra Serrano

2024-03-06

Índice

Manipulación de datos con dplyr

Tidyverse es una colección de herramientas destinadas para el análisis de datos, dichas herramientas permiten la manipulación y la visualización de los datos.

Se va a trabajar con el DataFrame **gapminder** presente en la librería homónima. Dicho DataFrame recoge indicadores sociales y económicos de los países como su esperanza de vida o su PIB a lo largo de varios años.

```
library(tidyverse)
library(gapminder)
gapminder
```

```
## # A tibble: 1,704 x 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
```

La función filter()

La función **filter()** permite observar un subconjunto del conjunto de datos que cumpla con una condición particular. Dicha función suele estar acompañada del operador pipe %>% que toma el resultado de una expresión y lo pasa como primer argumento a la siguiente expresión, facilitando la composición de operaciones al permitir encadenar funciones de manera más clara y concisa.

Sin operador pipe:

```
filter(gapminder, year == 2007)
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp       pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Afghanistan Asia       2007    43.8  31889923      975.
##  2 Albania     Europe     2007    76.4   3600523     5937.
##  3 Algeria     Africa     2007    72.3  33333216     6223.
##  4 Angola      Africa     2007    42.7  12420476     4797.
##  5 Argentina   Americas   2007    75.3  40301927    12779.
##  6 Australia   Oceania    2007    81.2  20434176    34435.
##  7 Austria     Europe     2007    79.8   8199783    36126.
##  8 Bahrain     Asia       2007    75.6    708573    29796.
##  9 Bangladesh  Asia       2007    64.1 150448339     1391.
## 10 Belgium     Europe     2007    79.4  10392226    33693.
## # i 132 more rows
```

Con operador pipe:

```
gapminder %>%
  filter(year == 2007)
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp       pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Afghanistan Asia       2007    43.8  31889923      975.
##  2 Albania     Europe     2007    76.4   3600523     5937.
##  3 Algeria     Africa     2007    72.3  33333216     6223.
##  4 Angola      Africa     2007    42.7  12420476     4797.
##  5 Argentina   Americas   2007    75.3  40301927    12779.
##  6 Australia   Oceania    2007    81.2  20434176    34435.
##  7 Austria     Europe     2007    79.8   8199783    36126.
##  8 Bahrain     Asia       2007    75.6    708573    29796.
##  9 Bangladesh  Asia       2007    64.1 150448339     1391.
## 10 Belgium     Europe     2007    79.4  10392226    33693.
## # i 132 more rows
```

No se esta eliminando ningun dato del resto del DataFrame, solo selecciona un conjunto que cumple con una condición, en este caso, son los datos de todos los países en el año 2007.

La función **filter()** permite discriminar bajo varias condiciones

Sin operador pipe:

```
filter(gapminder, year == 2007, country == "China")
```

```
## # A tibble: 1 x 6
##   country continent  year lifeExp        pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>      <int>     <dbl>
## 1 China   Asia       2007    73.0 1318683096     4959.
```

Con operador pipe:

```
gapminder %>%
  filter(year == 2007, country == "China")
```

```
## # A tibble: 1 x 6
##   country continent  year lifeExp        pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>      <int>     <dbl>
## 1 China   Asia       2007    73.0 1318683096     4959.
```

La función arrange()

La función **arrange()** ordena las observaciones de un dataset en orden ascendente o descendente según una de sus variables. Puede también ser empleada con el operador pipe %>%.

Para ordenar de forma descendiente el dataset según la variable `gdpPercap`:

```
gapminder %>%
  arrange(desc(gdpPercap))
```

```
## # A tibble: 1,704 x 6
##    country   continent  year lifeExp     pop gdpPercap
##    <fct>     <fct>     <int>   <dbl>   <int>     <dbl>
##  1 Kuwait    Asia       1957    58.0  212846   113523.
##  2 Kuwait    Asia       1972    67.7  841934   109348.
##  3 Kuwait    Asia       1952    55.6  160000   108382.
##  4 Kuwait    Asia       1962    60.5  358266    95458.
##  5 Kuwait    Asia       1967    64.6  575003    80895.
##  6 Kuwait    Asia       1977    69.3 1140357    59265.
##  7 Norway    Europe     2007    80.2 4627926    49357.
##  8 Kuwait    Asia       2007    77.6 2505559    47307.
##  9 Singapore Asia       2007    80.0 4553009    47143.
## 10 Norway    Europe     2002    79.0 4535591    44684.
## # i 1,694 more rows
```

El operador pipe permite combinar dos funciones, de tal forma que permite primero aplicar la función **filter()** y luego la función **arrange()**, de tal forma que primero filtra y luego organiza:

```
gapminder %>%
  filter(country == "Spain") %>%
  arrange(desc(gdpPercap))
```

```
## # A tibble: 12 x 6
##    country continent  year lifeExp      pop gdpPercap
##    <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Spain   Europe     2007    80.9 40448191    28821.
##  2 Spain   Europe     2002    79.8 40152517    24835.
##  3 Spain   Europe     1997    78.8 39855442    20445.
##  4 Spain   Europe     1992    77.6 39549438    18603.
##  5 Spain   Europe     1987    76.9 38880702    15765.
##  6 Spain   Europe     1982    76.3 37983310    13926.
##  7 Spain   Europe     1977    74.4 36439000    13237.
##  8 Spain   Europe     1972    73.1 34513161    10639.
##  9 Spain   Europe     1967    71.4 32850275     7994.
## 10 Spain   Europe     1962    69.7 31158061     5694.
## 11 Spain   Europe     1957    66.7 29841614     4565.
## 12 Spain   Europe     1952    64.9 28549870     3834.
```

Y en sentido ascendente sería:

```
gapminder %>%
  filter(country == "Spain") %>%
  arrange(gdpPercap)
```

```
## # A tibble: 12 x 6
##    country continent  year lifeExp      pop gdpPercap
##    <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Spain   Europe     1952    64.9 28549870     3834.
##  2 Spain   Europe     1957    66.7 29841614     4565.
##  3 Spain   Europe     1962    69.7 31158061     5694.
##  4 Spain   Europe     1967    71.4 32850275     7994.
##  5 Spain   Europe     1972    73.1 34513161    10639.
##  6 Spain   Europe     1977    74.4 36439000    13237.
##  7 Spain   Europe     1982    76.3 37983310    13926.
##  8 Spain   Europe     1987    76.9 38880702    15765.
##  9 Spain   Europe     1992    77.6 39549438    18603.
## 10 Spain   Europe     1997    78.8 39855442    20445.
## 11 Spain   Europe     2002    79.8 40152517    24835.
## 12 Spain   Europe     2007    80.9 40448191    28821.
```

La función mutate()

La función **mutate()** permite la modificación de variables existentes así como la eliminación y creación (a partir de otras) de variables.

Uso de **mutate()** para modificar una variable existente; dividir la variable `pop` (población) entre un millón:

```
gapminder %>%
  mutate(pop = pop / 1e6)
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp   pop gdpPercap
##    <fct>       <fct>     <int>   <dbl> <dbl>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8.43      779.
##  2 Afghanistan Asia       1957    30.3  9.24      821.
##  3 Afghanistan Asia       1962    32.0 10.3       853.
##  4 Afghanistan Asia       1967    34.0 11.5       836.
##  5 Afghanistan Asia       1972    36.1 13.1       740.
##  6 Afghanistan Asia       1977    38.4 14.9       786.
##  7 Afghanistan Asia       1982    39.9 12.9       978.
##  8 Afghanistan Asia       1987    40.8 13.9       852.
##  9 Afghanistan Asia       1992    41.7 16.3       649.
## 10 Afghanistan Asia       1997    41.8 22.2       635.
## # i 1,694 more rows
```

Nótese que no se esta modificando al dataset original, solo esta cambiando el valor en el DataFrame que esta devolviendo.

```
head(gapminder)
```

```
## # A tibble: 6 x 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.
```

Uso de **mutate()** para crear una nueva variable a partir de otras; el PIB per cápita viene determinado por la fórmula PIB per cápita $= \frac{\text{PIB}}{\text{Población}}$, por tanto, para introducir la variable `gdp` en el DataFrame:

```
gapminder %>%
  mutate(gdp = gdpPercap * pop)
```

```
## # A tibble: 1,704 x 7
##    country     continent  year lifeExp      pop gdpPercap          gdp
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>        <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.  6567086330.
##  2 Afghanistan Asia       1957    30.3  9240934      821.  7585448670.
##  3 Afghanistan Asia       1962    32.0 10267083      853.  8758855797.
##  4 Afghanistan Asia       1967    34.0 11537966      836.  9648014150.
##  5 Afghanistan Asia       1972    36.1 13079460      740.  9678553274.
##  6 Afghanistan Asia       1977    38.4 14880372      786. 11697659231.
##  7 Afghanistan Asia       1982    39.9 12881816      978. 12598563401.
##  8 Afghanistan Asia       1987    40.8 13867957      852. 11820990309.
##  9 Afghanistan Asia       1992    41.7 16317921      649. 10595901589.
## 10 Afghanistan Asia       1997    41.8 22227415      635. 14121995875.
## # i 1,694 more rows
```

De nuevo, el operador pipe permite emplear varias funciones en la misma expresión; lista descendente con los países con mayor PIB total:

```
gapminder %>%
  mutate(gdp = gdpPercap * pop) %>%
  filter(year == 2007) %>%
  arrange(desc(gdp))
```

```
## # A tibble: 142 x 7
##    country        continent  year lifeExp        pop gdpPercap     gdp
##    <fct>          <fct>     <int>   <dbl>      <int>     <dbl>   <dbl>
##  1 United States  Americas   2007    78.2  301139947    42952. 1.29e13
##  2 China          Asia       2007    73.0 1318683096     4959. 6.54e12
##  3 Japan          Asia       2007    82.6  127467972    31656. 4.04e12
##  4 India          Asia       2007    64.7 1110396331     2452. 2.72e12
##  5 Germany        Europe     2007    79.4   82400996    32170. 2.65e12
##  6 United Kingdom Europe     2007    79.4   60776238    33203. 2.02e12
##  7 France         Europe     2007    80.7   61083916    30470. 1.86e12
##  8 Brazil         Americas   2007    72.4  190010647     9066. 1.72e12
##  9 Italy          Europe     2007    80.5   58147733    28570. 1.66e12
## 10 Mexico         Americas   2007    76.2  108700891    11978. 1.30e12
## # i 132 more rows
```

Visualización con ggplot2

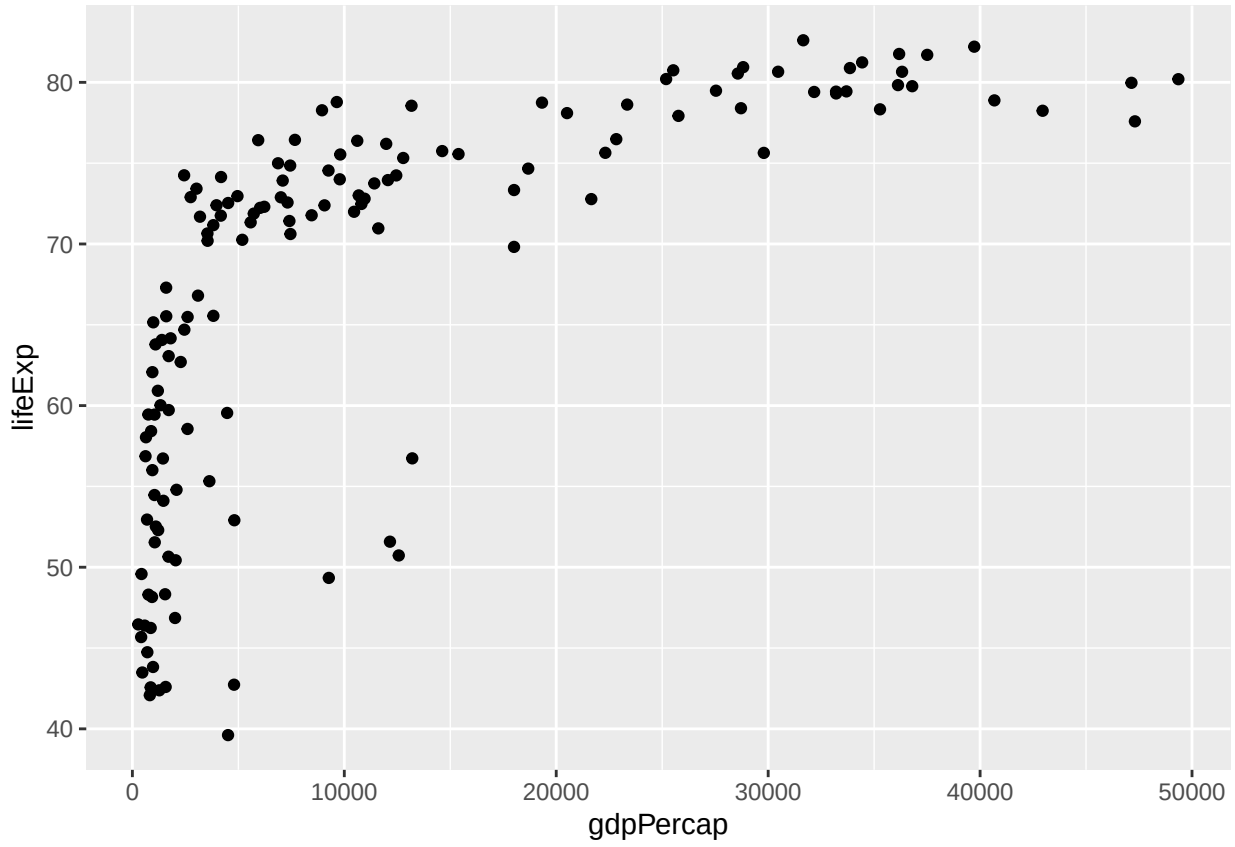
Diagramas de dispersión

Cuando se trabaja con un subconjunto de datos con el objetivo de visualizarlo, suele ser útil guardar este subconjunto en una nueva variable. Esto facilita las operaciones posteriores sobre los datos.

```
gapminder_2007 <- gapminder %>% filter(year == 2007)
```

El siguiente gráfico permite el estudio de la relación entre la esperanza de vida y el PIB de un país empleando los datos del año 2007 asignados en la variable `gapminder_2007` declarada anteriormente.

```
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```

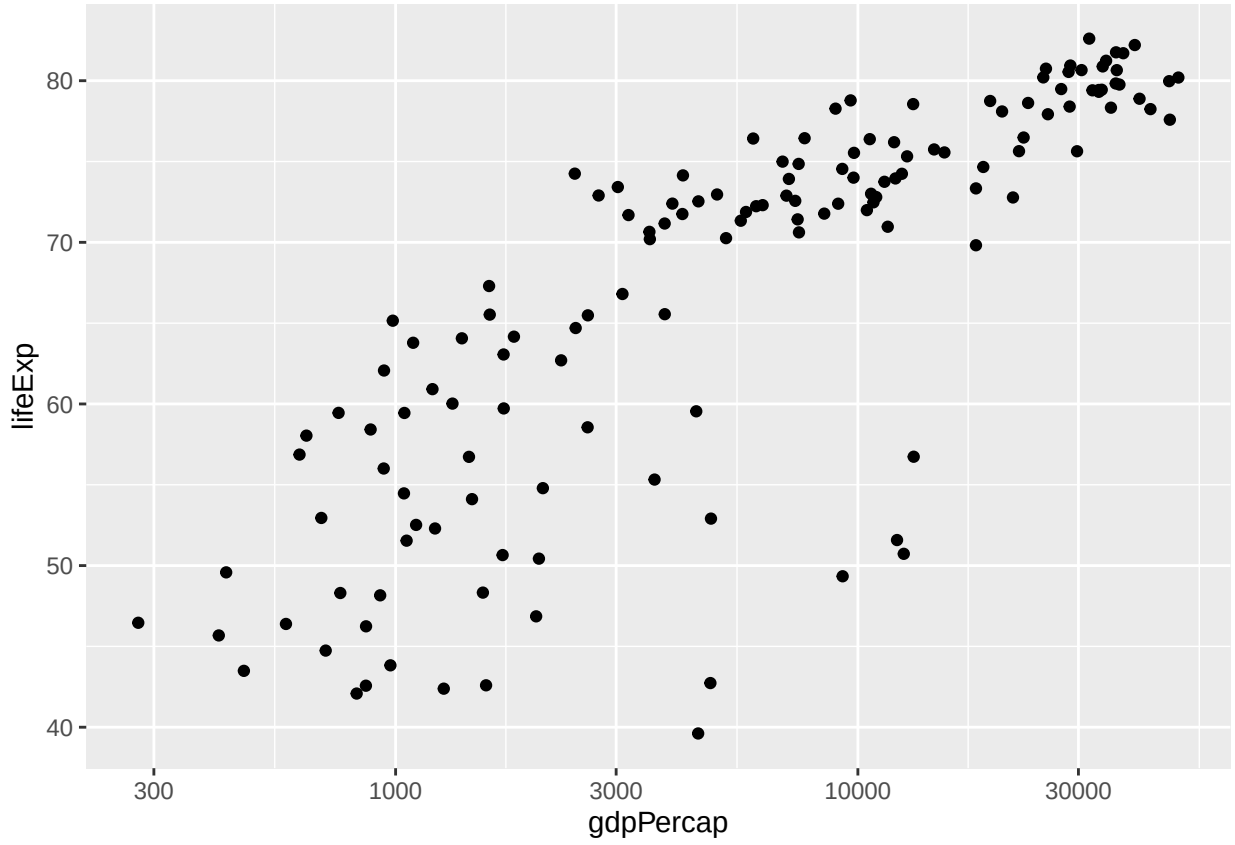


- **ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp))}**: Este segmento inicia la construcción del gráfico utilizando la función **ggplot()**. Se especifica el conjunto de datos a utilizar (`gapminder_2007`) y se establece la asignación estética con **aes()** de las variables a los ejes del gráfico. En este caso, el eje x se asigna al GDP per capita (`gdpPercap`) y el eje y se asigna a la esperanza de vida (`lifeExp`).
- **geom_point()**: Este comando agrega una capa geométrica al gráfico generado por **ggplot()**. En particular, **geom_point()** indica que se deben representar los datos como puntos en el gráfico, lo que crea un gráfico de dispersión donde cada observación se representa como un punto en el plano cartesiano definido por las variables especificadas anteriormente.

Escalas logarítmicas

Podemos observar en la gráfica anterior que los países con niveles más altos de ingresos exhiben una tendencia hacia una mayor esperanza de vida. Sin embargo, se observa que numerosos países se encuentran agrupados en la porción más izquierda del eje horizontal debido a la amplia variabilidad en los valores del Producto Interno Bruto per cápita, los cuales abarcan múltiples órdenes de magnitud. Con el propósito de superar esta dificultad, se recomienda la adopción de una escala logarítmica, en la cual cada intervalo fijo representa una multiplicación del valor correspondiente. Al implementar esta escala logarítmica sobre el eje horizontal, se evidencia que cada unidad en dicho eje implica un incremento de diez veces en el valor del PIB. Esta medida facilita una representación más clara de la relación existente entre el PIB per cápita y la esperanza de vida, lo que a su vez permite una distinción más nítida entre los países situados en el extremo inferior del espectro. Para la creación de este tipo de gráficos se emplea la instrucción **scale_x_log10()** (en el eje Y sería **scale_y_log10()**).

```
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  scale_x_log10()
```



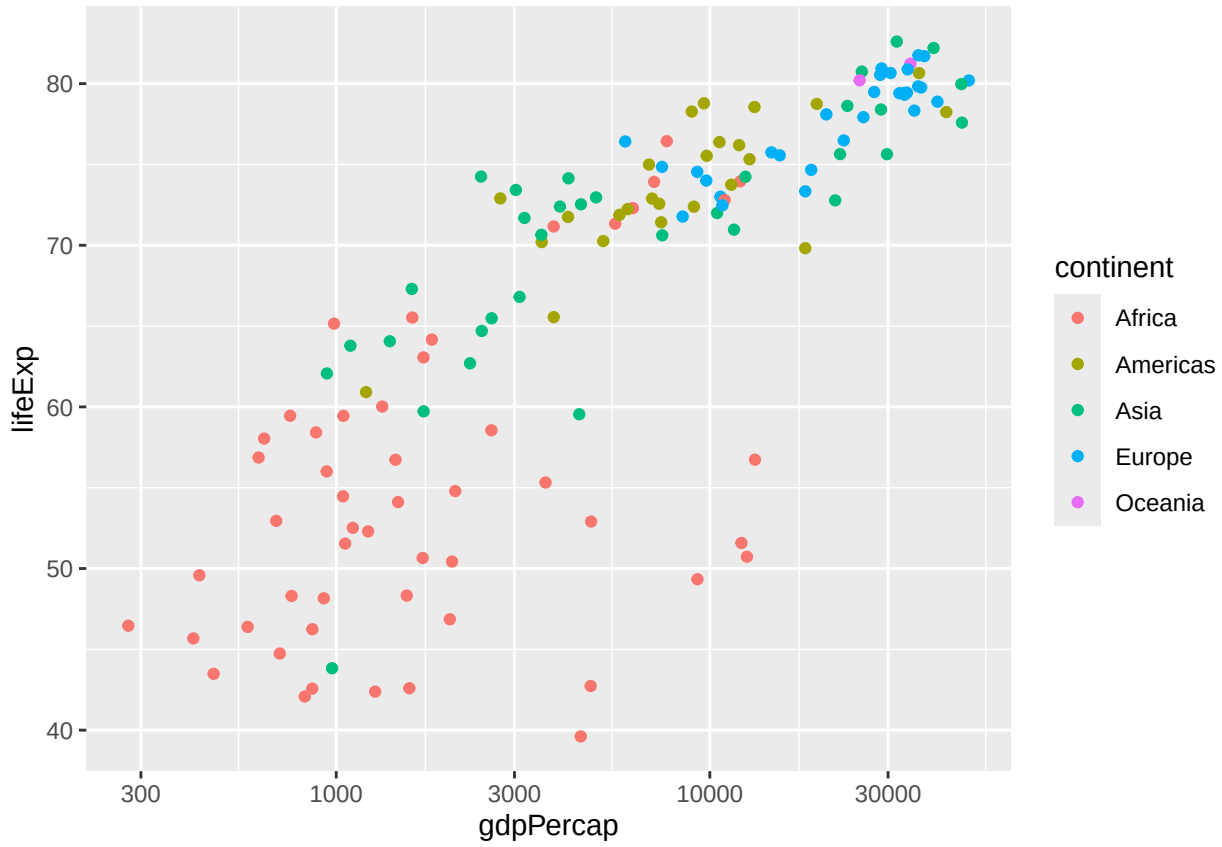
Configuraciones adicionales

Para representar nuevas variables en los gráficos se puede emplear el color y el tamaño.

Color

Dentro de la función **ggplot()** se puede especificar el parámetro `color` de manera que los puntos adquieran color. En el gráfico siguiente, se indica el continente al que pertenece cada país en función del color de su punto:

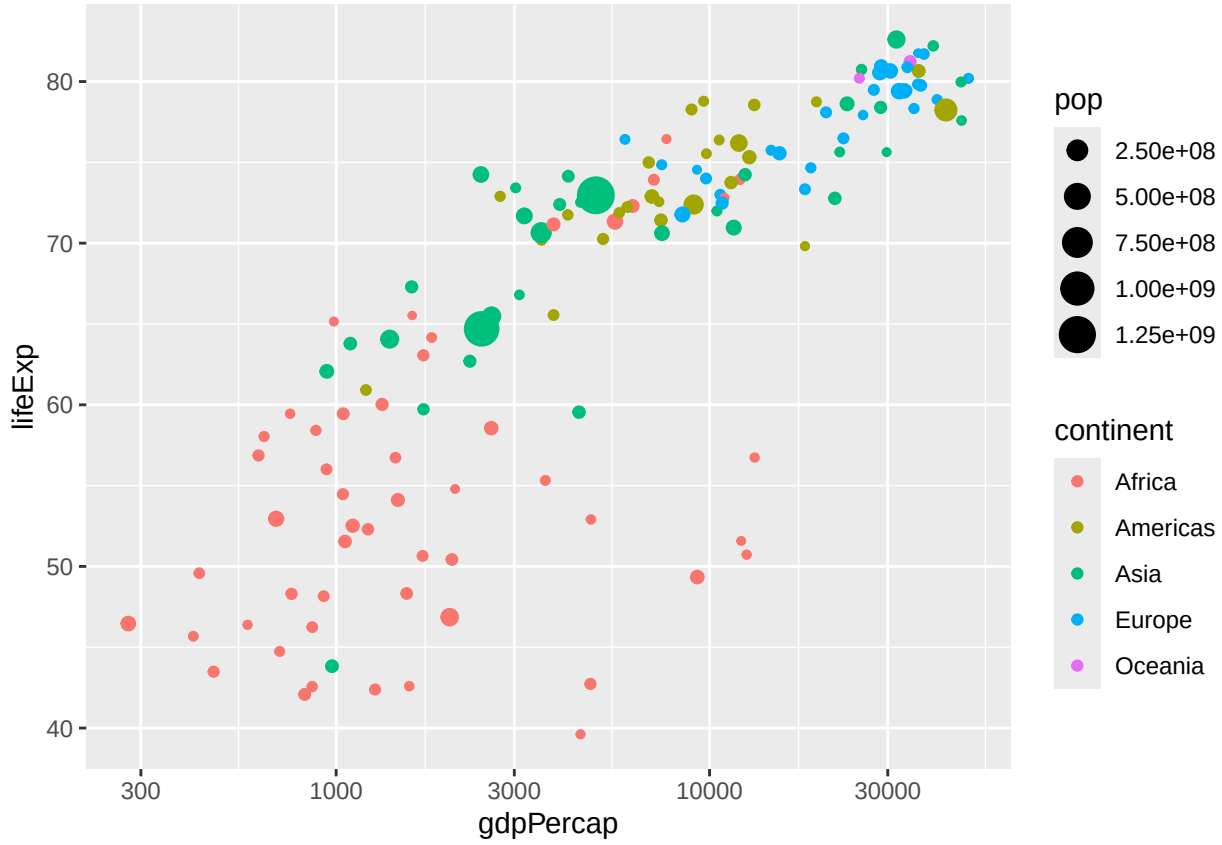
```
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point() +
  scale_x_log10()
```



Tamaño

El gráfico anterior aún permite representar más variables; por ejemplo, la población puede ser representada mediante el tamaño de los puntos en el gráfico de dispersión. Para lograrlo, la función ggplot() cuenta con el parámetro `size`:

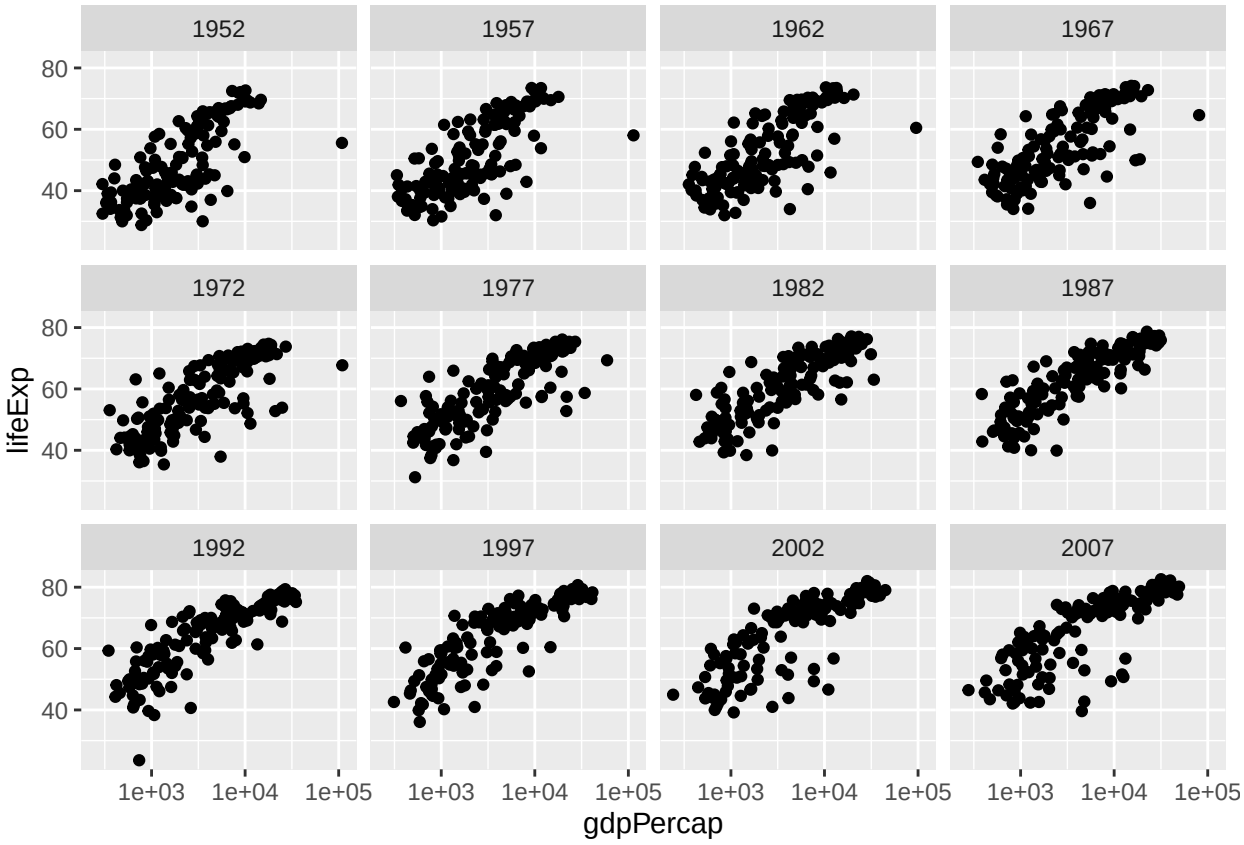
```
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp, color = continent,
                           size = pop)) +
  geom_point() +
  scale_x_log10()
```



Facetas

ggplot2 permite la división de un gráfico en subgráficos más pequeños según una variable categórica haciendo uso de la instrucción **facet_wrap()**:

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  scale_x_log10() +
  facet_wrap(~ year)
```



Agrupar y resumir

La función summarize()

La función **summarize()** se utiliza específicamente para generar resúmenes estadísticos a partir de los datos agrupados. La agrupación de datos es un proceso mediante el cual se dividen los datos en grupos basados en una o varias variables clave. Algunas de las funciones más empleadas con **summarize()** son:

- **mean()**: calcula la media
- **sum()**: calcula la suma de todos los valores
- **median()**: calcula la mediana
- **min()** y **max()**: obtienen los valores mínimos y máximos, respectivamente.

```
gapminder %>%
    filter(year == 2007) %>%
    summarize(medianLifeExp = median(lifeExp),
              maxGdpPercap = max(gdpPercap))
```

```
## # A tibble: 1 x 2
##   medianLifeExp maxGdpPercap
##           <dbl>        <dbl>
## 1          71.9       49357.
```

El código anterior crea un DataFrame con la mediana de la variable `lifeExp` y el valor máximo de `gdpPercap` de todos los países del DataFrame originial del año 2007.

La función group_by()

La función **group_by()** se utiliza para agrupar un conjunto de datos según una o varias variables específicas. Este proceso es fundamental en análisis de datos, ya que permite segmentar los datos en grupos más pequeños con base en características comunes. Una vez que se han agrupado los datos, se pueden realizar operaciones de resumen, filtrado o transformación dentro de cada grupo.

```
gapminder %>%
  group_by(year) %>%
  summarize(medianLifeExp = median(lifeExp),
            maxGdpPercap = max(gdpPercap))
```

```
## # A tibble: 12 x 3
##     year medianLifeExp maxGdpPercap
##    <int>         <dbl>        <dbl>
##  1  1952          45.1      108382.
##  2  1957          48.4      113523.
##  3  1962          50.9       95458.
##  4  1967          53.8       80895.
##  5  1972          56.5      109348.
##  6  1977          59.7       59265.
##  7  1982          62.4       33693.
##  8  1987          65.8       31541.
##  9  1992          67.7       34933.
## 10  1997          69.4       41283.
## 11  2002          70.8       44684.
## 12  2007          71.9       49357.
```

El código anterior crea un DataFrame con la mediana de la variable `lifeExp` y el valor máximo de `gdpPercap` de todos los países del DataFrame originial para cada año de la variable `year`.

group_by() permite agrupar en función de dos o más variables

```
gapminder %>%
  group_by(year, continent) %>%
  summarize(medianLifeExp = median(lifeExp),
            maxGdpPercap = max(gdpPercap))
```

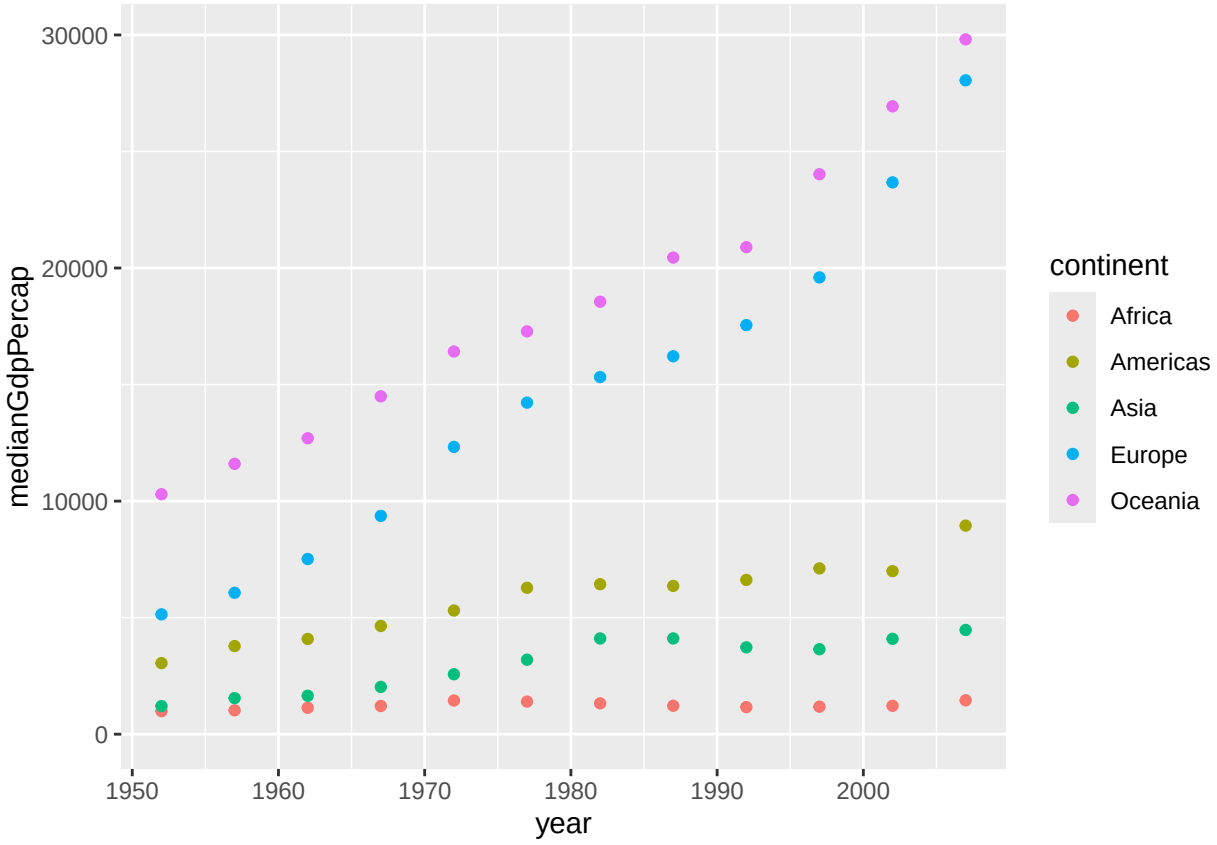
```
## # A tibble: 60 x 4
## # Groups:   year [12]
##     year continent medianLifeExp maxGdpPercap
##    <int> <fct>             <dbl>        <dbl>
##  1  1952 Africa             38.8        4725.
##  2  1952 Americas           54.7       13990.
##  3  1952 Asia               44.9      108382.
##  4  1952 Europe             65.9       14734.
##  5  1952 Oceania            69.3       10557.
##  6  1957 Africa             40.6        5487.
##  7  1957 Americas           56.1       14847.
##  8  1957 Asia               48.3      113523.
##  9  1957 Europe             67.6       17909.
## 10  1957 Oceania            70.3       12247.
## # i 50 more rows
```

El código anterior crea un DataFrame con la mediana de la variable `lifeExp` y el valor máximo de `gdpPercap` de todos los países del DataFrame originial para cada año y continente de las variable `year` y `continent` respectivamente.

Visualización

```
by_year_continent <- gapminder %>%
                        group_by(continent, year) %>%
                        summarize(medianGdpPercap = median(gdpPercap))

ggplot(by_year_continent, aes(x = year, y = medianGdpPercap, color = continent)) +
    geom_point() +
    expand_limits(y = 0)
```



La instrucción **expand_limits(y = 0)** configura el eje Y del gráfico para que empiece en 0.

Tipos de gráficos

Diagramas de líneas

Los diagramas de líneas resultan útiles para estudiar la variación de una variable a lo largo del tiempo. Se construyen de manera similar a un gráfico de dispersión, con la diferencia de que se utiliza **geom_line()** en lugar de **geom_point()**.

```
ggplot(by_year_continent, aes(x = year, y = medianGdpPercap, color = continent)) +
    geom_line() +
    expand_limits(y = 0)
```

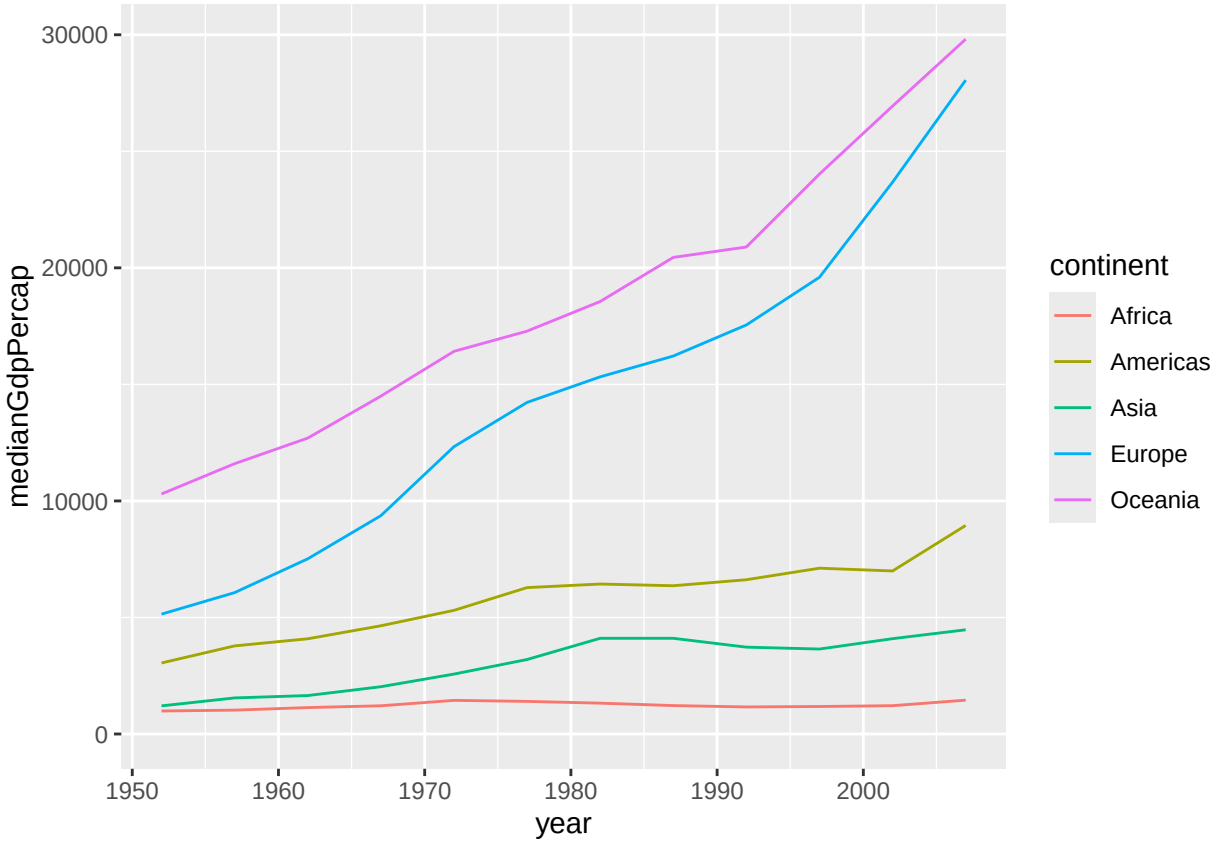
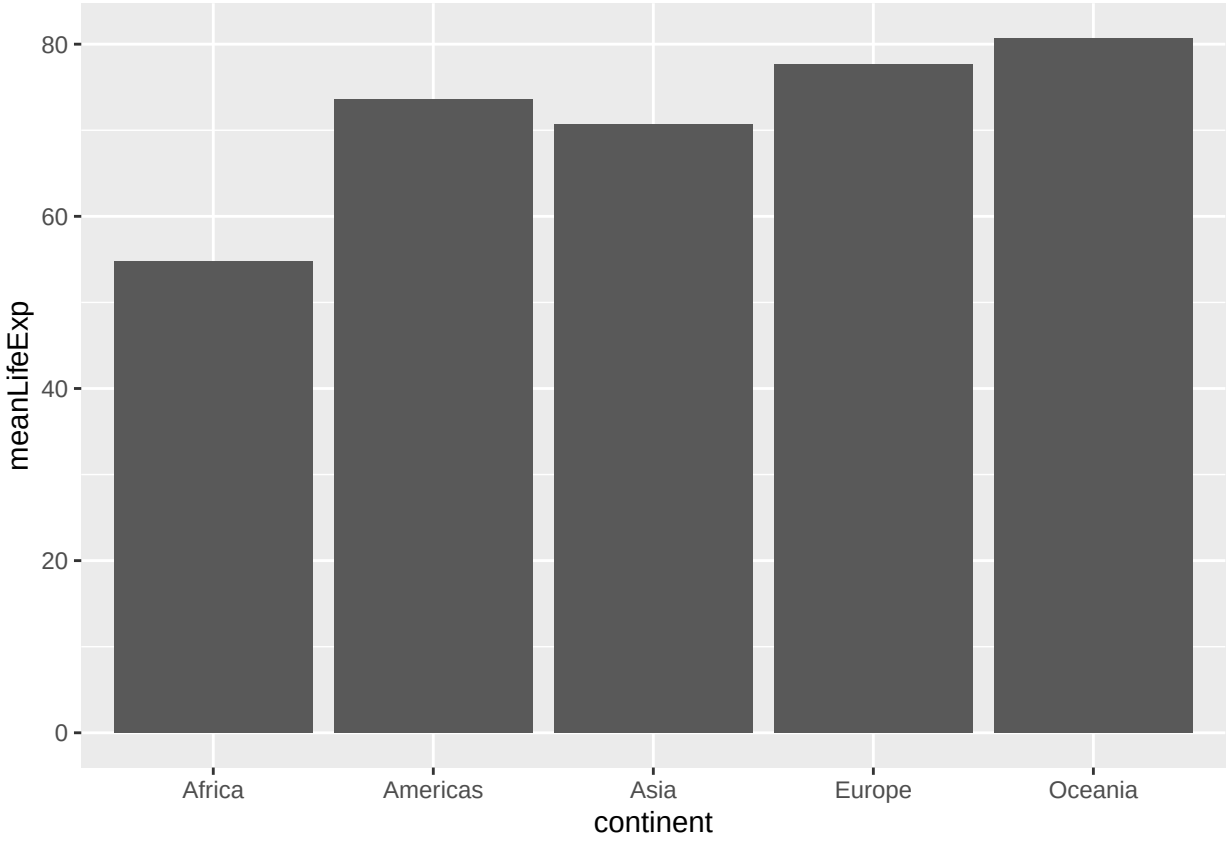


Gráfico de barras

Los gráficos de barras resultan útiles para comparar valores entre categorías discretas. Para crear un gráfico de barras se emplea la instrucción **geom_col()**

```
by_continent <- gapminder%>%
                  filter(year == 2007) %>%
                  group_by(continent) %>%
                  summarize(meanLifeExp = mean(lifeExp))

ggplot(by_continent, aes(x = continent, y = meanLifeExp)) +
  geom_col()
```

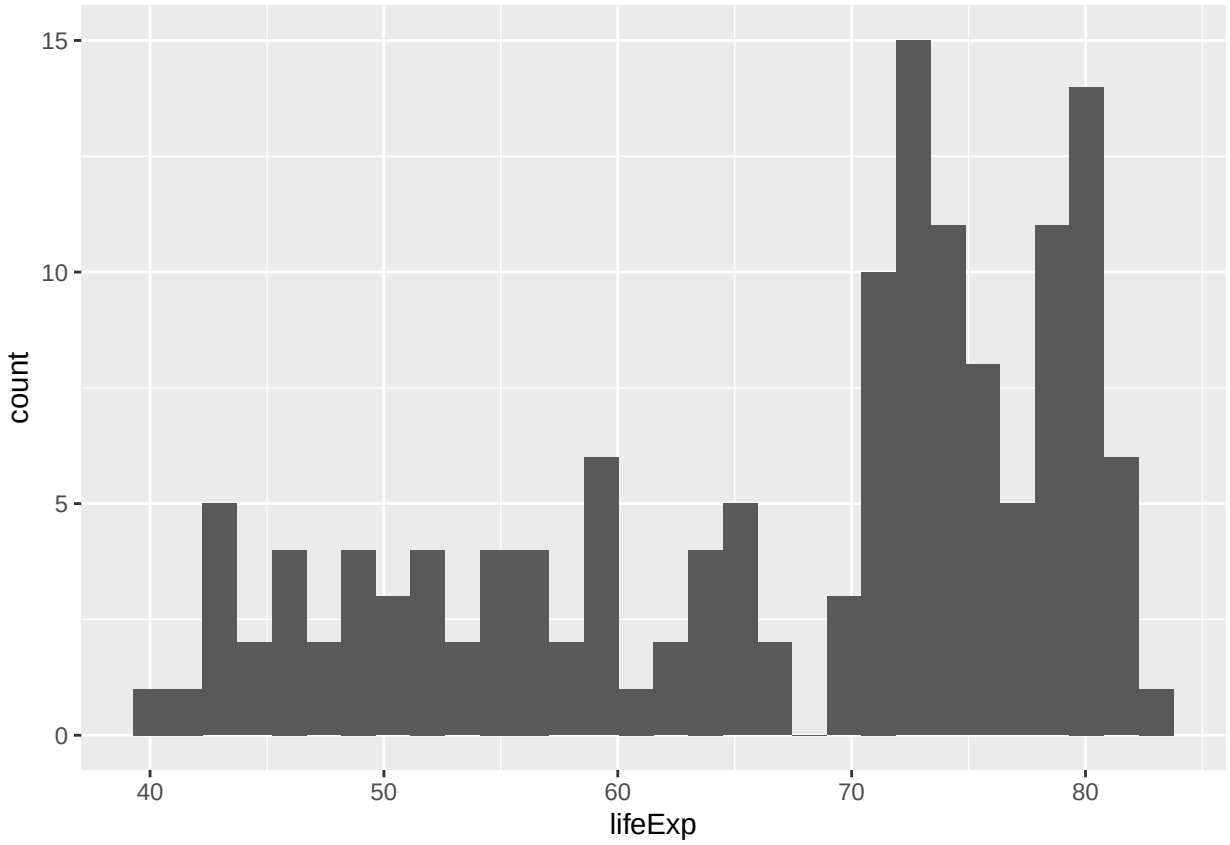


Histogramas

Los histogramas permiten la visualización de la distribución de una variable. Se declaran con la instrucción **geom_histogram()**.

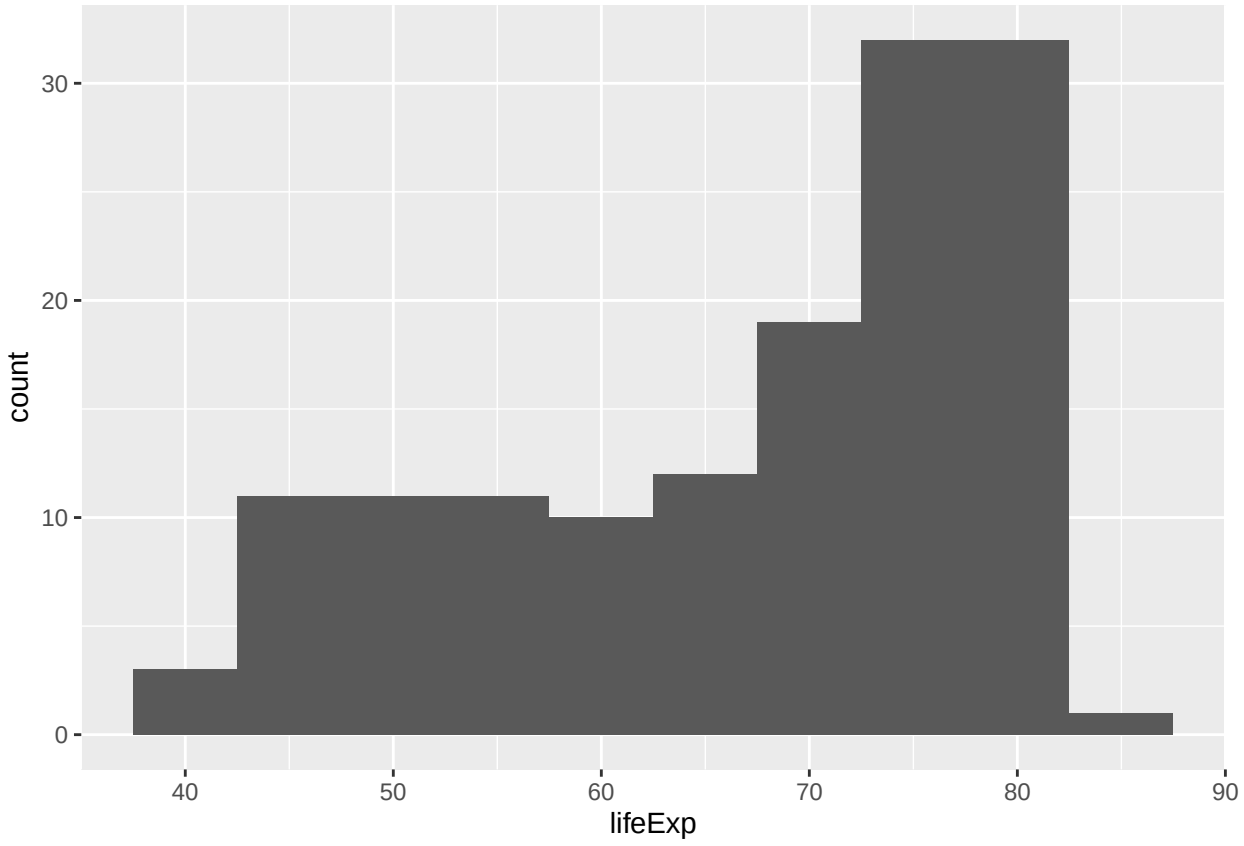
```
ggplot(gapminder_2007, aes(x = lifeExp)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



La anchura de cada barra se ajusta automáticamente; para establecer un valor específico, se utiliza el parámetro `binwidth` dentro de **geom_histogram()**.

```
ggplot(gapminder_2007, aes(x = lifeExp)) +
  geom_histogram(binwidth = 5)
```



También se puede modificar el número de barras con el parámetro `bins` dentro de **geom_histogram()**.

```
ggplot(gapminder_2007, aes(x = lifeExp)) +
  geom_histogram(bins = 5)
```

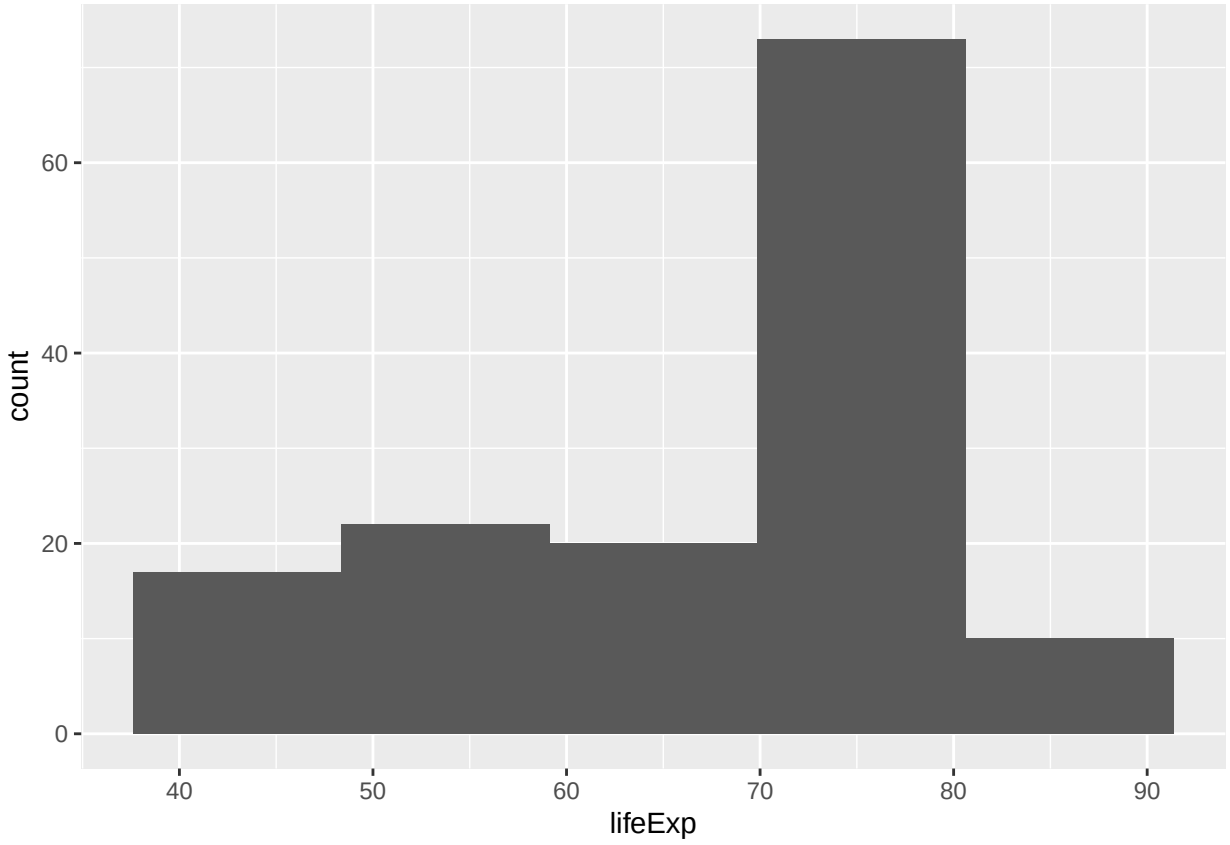


Gráfico de bigotes

Un gráfico de bigotes proporciona información sobre la distribución de un conjunto de datos y muestra de manera resumida cinco estadísticas clave: el valor mínimo, el percentil 25 (Q1), la mediana (Q2), el percentil 75 (Q3) y el valor máximo, además de visualizar los valores atípicos si los hubiera. Se declara con la instrucción **geom_boxplot()**.

```
ggplot(gapminder_2007, aes(x = continent, y = lifeExp)) +
  geom_boxplot()
```