Introducción a la Estadística en R

Rubén Sierra Serrano

2024-03-28

Índice

Estadística Básica

A lo largo del documento se va a trabajar con el DataFrame `food_consumption`, `sales` y `world_happiness`.

Cuartiles y cuantiles

Los cuartiles son puntos que dividen una distribución de datos en cuatro partes iguales, representando así el 25% de los datos en cada una de ellas. Hay tres cuartiles principales que dividen los datos en cuatro partes:

- El primer cuartil (Q1) es el valor que separa el 25% inferior de los datos del 75% superior.
- El segundo cuartil (Q2) es el mismo que la mediana, que separa los datos en dos partes iguales, dejando el 50% de los datos a cada lado.
- El tercer cuartil (Q3) es el valor que separa el 75% inferior de los datos del 25% superior.

Los cuantiles son puntos que dividen una distribución de datos en partes iguales, representando así un porcentaje específico de los datos en cada una de ellas. Los cuartiles son un tipo específico de cuantiles (dividen los datos en cuatro partes iguales), pero los cuantiles pueden dividir los datos en cualquier número de partes iguales. Por ejemplo, los percentiles son cuantiles que dividen los datos en cien partes iguales y los deciles dividen los datos en diez partes iguales.

Los quintiles son un tipo específico de cuantiles que dividen una distribución de datos en cinco partes iguales, representando así el 20% de los datos en cada una de ellas.

La función **quantile()** del paquete `dplyr` permite calcular los cuantiles. Por defecto, calcula los cuartiles. Al especificar el argumento `probs`, se pueden definir cualquier cantidad de valores entre 0 y 1 para obtener los cuantiles correspondientes.

```
quantile(food_consumption$co2_emission)
```

```
##       0%      25%      50%      75%     100%
##   0.0000   5.2100  16.5300  62.5975 1712.0000
```

```
quantile(food_consumption$co2_emission, probs = c(0,0.2,0.4,0.6,0.8,1))
```

```
##       0%      20%      40%      60%      80%     100%
##    0.000    3.540   11.026   25.590   99.978 1712.000
```

```
quantile(food_consumption$co2_emission, probs = seq(0,1,0.1))
```

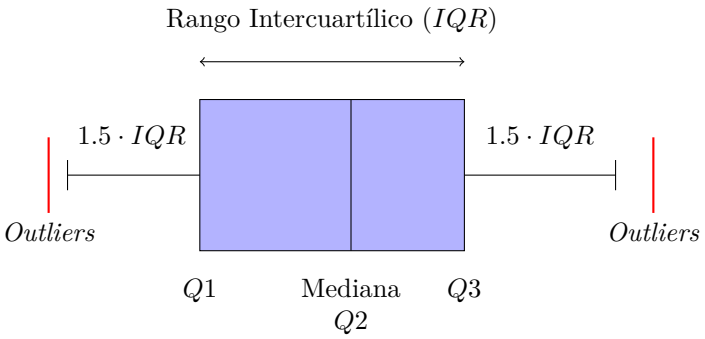
```
##       0%      10%      20%      30%      40%      50%      60%      70%
##    0.000    0.668    3.540    7.040   11.026   16.530   25.590   44.271
##      80%      90%     100%
##   99.978  203.629 1712.000
```

Rango intercuartílico y outliers

El rango intercuartílico (IQR) es una medida de dispersión que se calcula restando el tercer cuartil ($Q3$) del primer cuartil ($Q1$) de un conjunto de datos, es decir, $IQR = Q3 - Q1$.

Los *outliers* (valores atípicos) son puntos de datos que se encuentran significativamente alejados del resto de los datos en un conjunto y que no se relacionan con otros. Estos valores pueden deberse a errores de medición, variabilidad natural en los datos o a circunstancias especiales.

Un criterio para identificar valores atípicos como aquellos que están por debajo de $Q1 - 1.5 \cdot IQR$ o por encima de $Q3 + 1.5 \cdot IQR$.




```
emissions_by_country <- food_consumption %>%
  group_by(country) %>%
  summarize(total_emission = sum(co2_emission))

q1 <- quantile(emissions_by_country$total_emission, 0.25)
q3 <- quantile(emissions_by_country$total_emission, 0.75)
iqr <- q3 - q1

lower <- q1 - 1.5 * iqr
upper <- q3 + 1.5 * iqr

emissions_by_country %>%
  filter(total_emission < lower | total_emission > upper)
```

```
## # A tibble: 1 x 2
##   country   total_emission
##   <chr>              <dbl>
## 1 Argentina          2172.
```

Varianza y desviación estándar

La varianza y la desviación estándar son medidas de dispersión que se utilizan en estadística para describir la distribución de un conjunto de datos.

1. Varianza: La varianza es una medida de la dispersión de los datos respecto a la media. Se calcula como la media de los cuadrados de las diferencias entre cada dato y la media del conjunto. En R se calcula con la función **var()**. Formalmente, si tenemos un conjunto de datos $X = \{x_1, x_2, ..., x_n\}$ con media $\bar{x}$, la varianza (σ^2) se calcula mediante la fórmula:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

   Donde:

   - x_i son los elementos del conjunto de datos.
   - $\bar{x}$ es la media del conjunto de datos.
   - n es el número de elementos en el conjunto de datos.

2. Desviación estándar: La desviación estándar es la raíz cuadrada de la varianza. Se utiliza comúnmente porque está en las mismas unidades que los datos originales y resulta más sencillo de interpretar. En R se calcula con la función **sd()**. Se denota comúnmente por σ o s, dependiendo de si se refiere a la población o a la muestra, respectivamente. La fórmula para la desviación estándar (σ o s) es:

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$$

   Donde:

   - σ^2 es la varianza.
   - σ es la desviación estándar.

En resumen, la varianza mide la dispersión total de los datos, mientras que la desviación estándar mide la dispersión promedio de los datos respecto a la media.

```
food_consumption %>%
  group_by(food_category) %>%
  summarize(var_co2 = var(co2_emission),
            sd_co2 = sd(co2_emission))
```

```
## # A tibble: 11 x 3
##    food_category   var_co2  sd_co2
##    <fct>             <dbl>   <dbl>
##  1 beef           88748.    298.
##  2 eggs              21.4     4.62
##  3 fish             922.     30.4
##  4 lamb_goat      16476.    128.
##  5 dairy          17672.    133.
##  6 nuts              35.6     5.97
##  7 pork            3095.     55.6
##  8 poultry          245.     15.7
##  9 rice            2281.     47.8
## 10 soybeans           0.880   0.938
## 11 wheat             71.0     8.43
```

Probabilidad y distribuciones

Probabilidad

La probabilidad de un evento se puede calcular tomando el número de veces en que el evento puede ocurrir y dividiéndolo por el número total de resultados posibles. Es decir, la probabilidad de un evento es una medida numérica que indica la posibilidad de que dicho evento ocurra en relación con el conjunto completo de resultados posibles.

$$P(\text{Evento}) = \frac{\text{Número de veces en que el evento puede ocurrir}}{\text{Total de resultados posibles}}$$

El DataFrame `sales` tiene la siguiente forma:

```
glimpse(sales)
```

```
## Rows: 178
## Columns: 5
## $ product   <fct> Product F, Product C, Product B, Product I, Product E, Produ~
## $ client    <fct> Current, New, New, Current, Current, New, Current, Current, ~
## $ status    <fct> Won, Won, Won, Won, Won, Won, Won, Won, Won, Won, Lost,~
## $ amount    <dbl> 7389.52, 4493.01, 5738.09, 2591.24, 6622.97, 5496.27, 3043.1~
## $ num_users <dbl> 19, 43, 87, 83, 17, 2, 29, 13, 80, 23, 26, 6, 15, 43, 41, 12~
```

La función **sample_n()** sirve para escoger x elementos del DataFrame de forma pseudo-aleatoria.

```
sales %>%
  sample_n(5)
```

```
##     product  client status  amount num_users
## 1 Product D Current   Lost 4274.80         9
## 2 Product D Current    Won 6733.62        27
## 3 Product J     New   Lost 3182.09         2
## 4 Product C Current    Won 4796.13        44
## 5 Product B Current    Won 5237.24        23
```

Al realizar otra vez el experimento obtendremos distintos resultados:

```
sales %>%
  sample_n(5)
```

```
##     product  client status  amount num_users
## 1 Product B     New    Won 5665.33        68
## 2 Product D Current    Won 6755.66        59
## 3 Product F Current    Won 3357.24        53
## 4 Product B Current    Won 2812.45        37
## 5 Product B Current    Won 7219.99        16
```

La función **set.seed()** sirve para establecer una semilla para generar números pseudoaleatorios. Al establecer una semilla, se garantiza que se obtendrán los mismos resultados "aleatorios" cada vez que se ejecute el código.

```
set.seed(42)
sales %>%
  sample_n(1)
```

```
##     product  client status  amount num_users
## 1 Product C Current   Lost 3727.66        19
```

```
set.seed(42)
sales %>%
  sample_n(1)
```

```
##     product  client status  amount num_users
## 1 Product C Current   Lost 3727.66        19
```

La función **sample_n()** posee el argumento `replace` que de manera predeterminada adopta el valor `FALSE`. Sin embargo, al proporcionarle el valor `TRUE`, la función tomará las muestras con reemplazo.

```
sales %>%
  sample_n(5, replace = TRUE)
```

```
##     product  client status  amount num_users
## 1 Product B Current    Won 1640.06        59
## 2 Product A Current   Lost 9027.17        71
## 3 Product D     New   Lost 2723.39        42
## 4 Product A Current    Won 4682.94        63
## 5 Product B Current    Won 4831.73        12
```

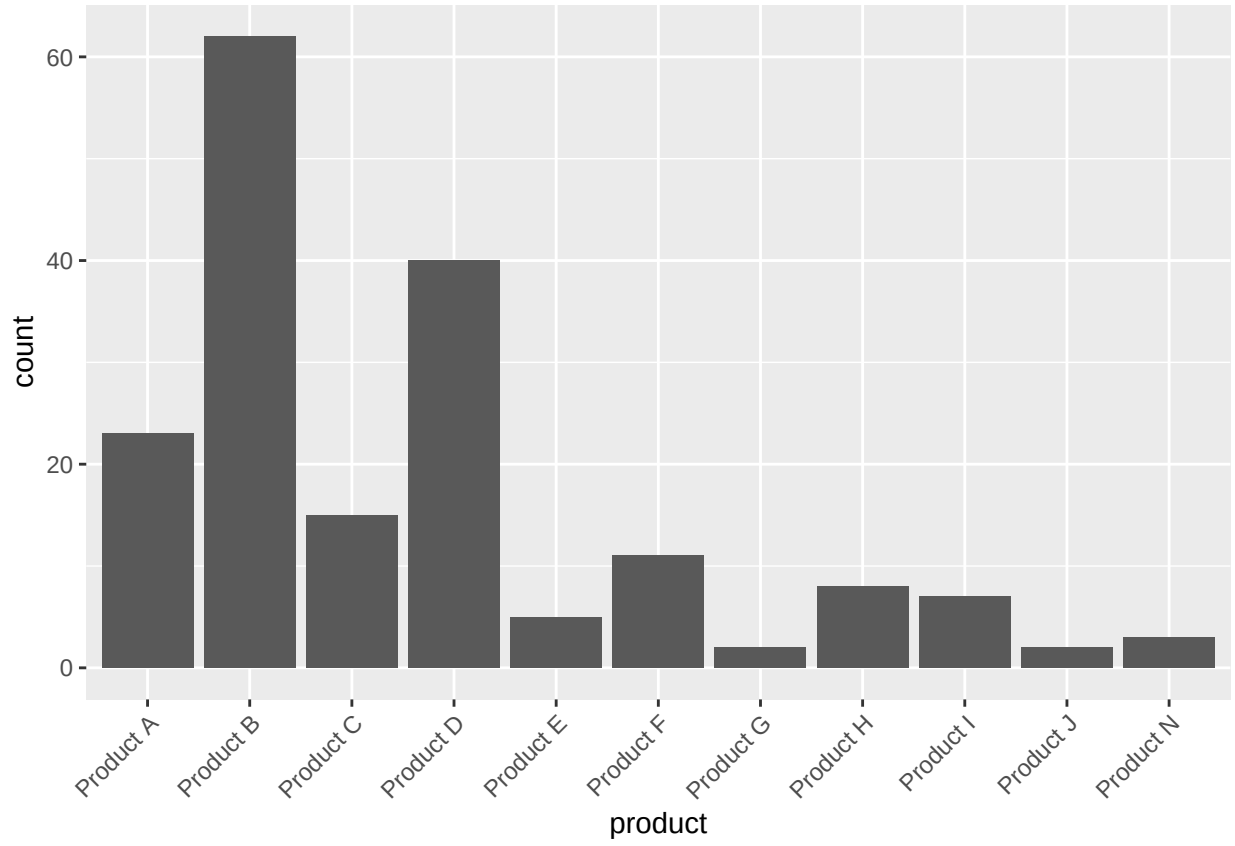
Distribuciones discretas

Una distribución de probabilidad muestra cómo se distribuyen las probabilidades entre los diferentes resultados posibles de un experimento aleatorio. El área bajo la curva de una distribución de probabilidad representa la probabilidad de que ocurra un evento particular o una combinación de eventos. En otras palabras, la integral (o suma, en el caso discreto) de la función de densidad de probabilidad sobre un rango específico de valores corresponde a la probabilidad de que el resultado del experimento aleatorio esté dentro de ese rango.

En el caso de una distribución de probabilidad discreta, se refiere a una distribución en la que las posibles ocurrencias son contables o discretas, es decir, hay un número finito o infinito numerable de resultados posibles.

Se puede visualizar una distribución de probabilidad discreta con un gráfico de barras; para variables continuas se emplearía un histograma:

```
ggplot(sales, aes(x = product)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```



Las probabilidades de la distribución son:

```
distribucion <- sales %>%
  count(product) %>%
  mutate(probabilidad = n / sum(n))

distribucion
```

```
##      product  n probabilidad
## 1  Product A 23   0.12921348
## 2  Product B 62   0.34831461
## 3  Product C 15   0.08426966
## 4  Product D 40   0.22471910
## 5  Product E  5   0.02808989
## 6  Product F 11   0.06179775
## 7  Product G  2   0.01123596
## 8  Product H  8   0.04494382
## 9  Product I  7   0.03932584
## 10 Product J  2   0.01123596
## 11 Product N  3   0.01685393
```

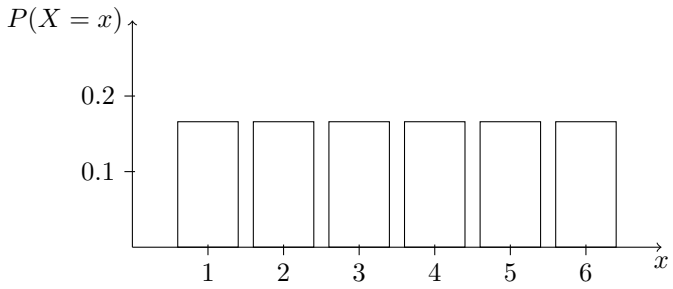
Nótese que al calcular la suma, podemos obtener la probabilidad de obtener un producto dentro de una categoría específica.

```
distribucion %>%
  filter(product %in% c("Product A","Product B","Product C")) %>%
  summarize(prob_A_B_C = sum(probabilidad))
```

```
##   prob_A_B_C
## 1  0.5617978
```

En el ejemplo anterior, la probabilidad de que el producto elegido sea uno de los productos A, B o C es de aproximadamente el 56%.

Una distribución de probabilidad uniforme discreta es un tipo de distribución de probabilidad que modela un conjunto finito de posibles resultados, donde cada resultado tiene la misma probabilidad de ocurrir.



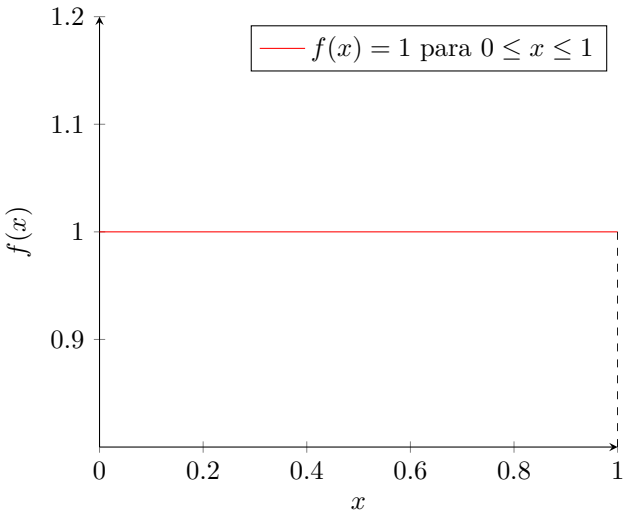
Distribuciones continuas

En el caso de una distribución de probabilidad continua, se refiere a una distribución en la que las posibles ocurrencias son incontables o continuas, es decir, hay un número infinito no numerable de resultados posibles.

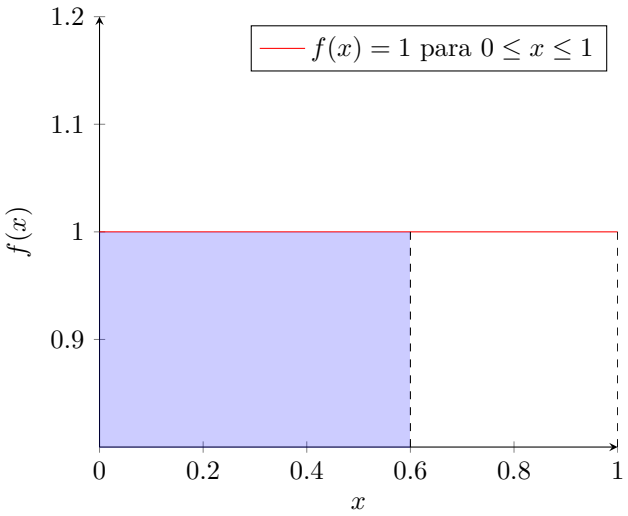
Existe el equivalente a las distribuciones de probabilidad uniformes discretas en el caso continuo; al igual que en su análogo discreto, cada resultado tiene la misma probabilidad de ocurrir. El área bajo la curva de dicha distribución es la probabilidad de que dicho evento ocurra.

$$f(x) = \begin{cases} \dfrac{1}{b-a} & \text{si } a \leq x \leq b \\ 0 & \text{si } x < a \text{ o } x > b \end{cases}$$

Un ejemplo de distribución uniforma continua con $a = 0$ y $b = 1$



Si se quiere conocer el valor de $P(X \leq x)$ para un $x \in [a, b]$ habría que calcular el área $\int_a^x \frac{1}{b-a} dx$. En el siguiente gráfico se muestra $P(X \leq 0.6)$ dada la función uniforme previamente declarada:

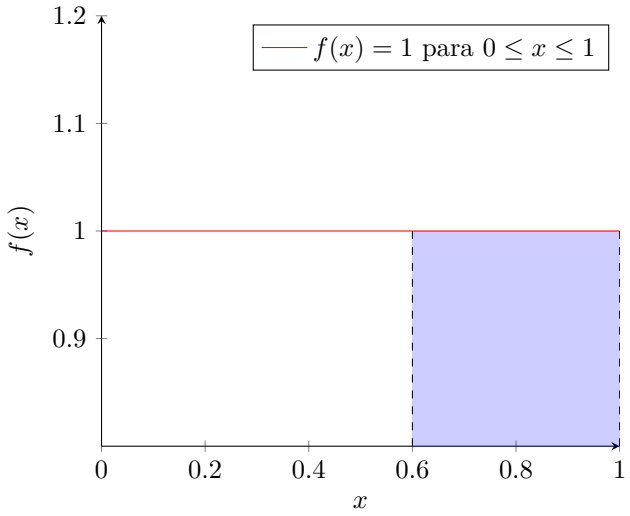


La función **punif()** calcula la probabilidad de que una variable aleatoria con distribución uniforme en un intervalo sea menor o igual que x.

```
min <- 0
max <- 1
punif(0.6, min = min, max = max)
```

```
## [1] 0.6
```

En el siguiente gráfico se muestra $P(X > 0.6)$ dada la función uniforme previamente declarada:



El argumento `lower.tail` de **punif()** es un parámetro booleano opcional que especifica si se debe calcular la probabilidad acumulativa en la "cola inferior" de la distribución o en la "cola superior". Por defecto, tiene el valor `TRUE`, lo que significa que devuelve la probabilidad de que la variable aleatoria sea menor o igual a x. Si se especifica con el valor `FALSE`, devuelve la probabilidad acumulativa de que la variable aleatoria sea mayor que x.

```
punif(0.6, min = min, max = max, lower.tail = FALSE)
```

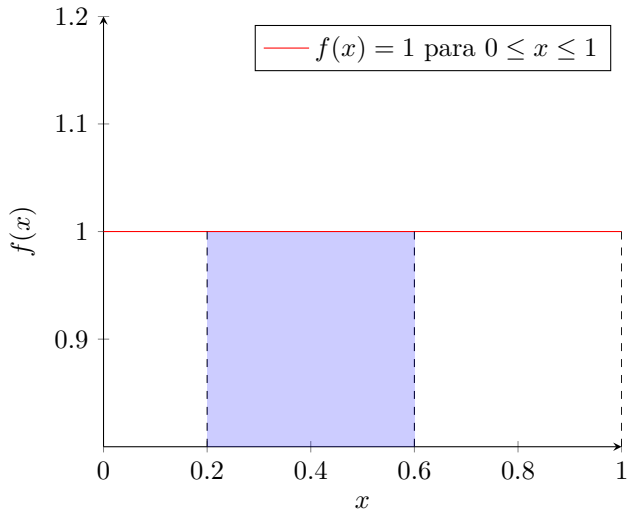
```
## [1] 0.4
```

Nótese que $P(X > x) = 1 - P(X \leq x)$

```
1 - punif(0.6, min = min, max = max)
```

```
## [1] 0.4
```

Para calcular la probabilidad acumulativa para un subintervalo tal que $a, b \notin [c, d]$ dentro del intervalo original $[a, b]$ se emplearía la expresión $P(c \leq X \leq d) = P(X \leq d) - P(X \leq c)$. En el siguiente gráfico se muestra $P(0.2 \leq X \leq 0.6)$ dada la función uniforme previamente declarada:




```
punif(0.6, min = min, max = max) - punif(0.2, min = min, max = max)
```

```
## [1] 0.4
```

La función **runif()** se utiliza para generar números aleatorios que siguen una distribución uniforme continua en un intervalo específico.

```
runif(n = 5, min = min, max = max)
```

```
## [1] 0.6569923 0.7050648 0.4577418 0.7191123 0.9346722
```

Donde:

- `n` es el número de valores aleatorios a generar.
- `min` es el límite inferior del intervalo de la distribución uniforme.
- `max` es el límite superior del intervalo de la distribución uniforme.

Distribución binomial

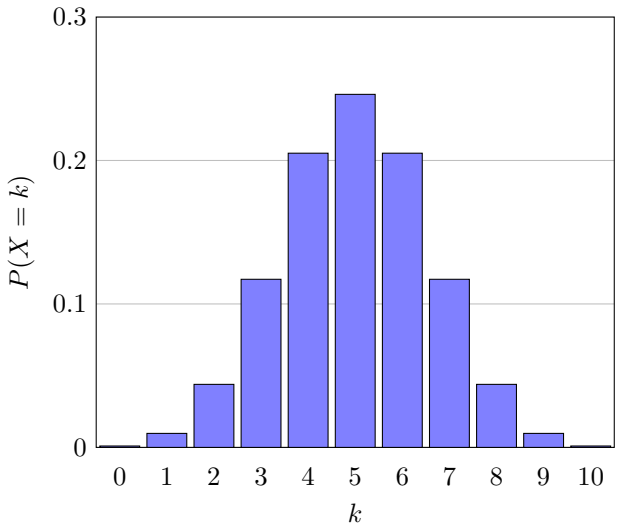
La distribución binomial describe la probabilidad de obtener un número específico de éxitos en un número fijo de n ensayos independientes, donde cada ensayo tiene dos resultados posibles: éxito o fracaso. Es un tipo de distribución discreta.

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Donde:

- n: Es el número total de ensayos o experimentos.
- k: Es el número de éxitos que se desean obtener en los n ensayos.
- p: Es la probabilidad de éxito en un solo ensayo.
- $(1-p)$ (o q): Es la probabilidad de fracaso en un solo ensayo.
- $\binom{n}{k}$: Representa el coeficiente binomial, que se calcula como $\frac{n!}{k!(n-k)!}$.

Ejemplo de gráfica de distribución binomial con $n = 10$ y $p = 0.5$:



La función **pbinom()** se utiliza para calcular la probabilidad acumulativa de una distribución binomial en un cierto punto $P(X \leq x)$.

```
pbinom(q = 3, size = 10, prob = 0.5, lower.tail = FALSE)
```

```
## [1] 0.828125
```

Donde:

- `q` es el valor en el que se evalúa la probabilidad acumulativa.
- `size` es el número total de ensayos.
- `prob` es la probabilidad de éxito en cada ensayo.
- `lower.tail` es un parámetro booleano opcional que especifica si se calcula la probabilidad acumulativa en la "cola inferior" de la distribución o en la "cola superior". Por defecto, `lower.tail` = `TRUE` calcula la probabilidad de que el número de éxitos sea menor o igual a `q`. Funciona igual que con **punif()**.

La función **dbinom()** permite conocer la probabilidad de que la variable aleatoria tome un valor concreto dado un número de ensayos y la probabilidad de éxito $P(X = x)$.

```
pbinom(q = 3, size = 10, prob = 0.5)
```

```
## [1] 0.171875
```

Donde:

- `q` es el valor en el que se evalúa la probabilidad.
- `size` es el número total de ensayos.
- `prob` es la probabilidad de éxito en cada ensayo.

La función **rbinom()** se utiliza para generar números aleatorios que siguen una distribución binomial.

```
rbinom(n = 5, size = 10, prob = 0.5)
```

```
## [1] 4 5 7 8 3
```

Donde:

- `n` es el número de valores aleatorios a generar.
- `size` es el número total de ensayos.
- `prob` es la probabilidad de éxito de cada ensayo.

Distribución normal

La distribución normal es una de las distribuciones estadísticas más importantes debido a su capacidad para modelar una amplia variedad de fenómenos naturales y sociales que exhiben comportamientos aleatorios.Se trata de una distribución de variable continua y se define completamente por dos parámetros: la media, que determina la ubicación del centro de la campana, y la desviación estándar, que describe la dispersión de los datos alrededor de la media.

Su función de densidad sigue la forma:

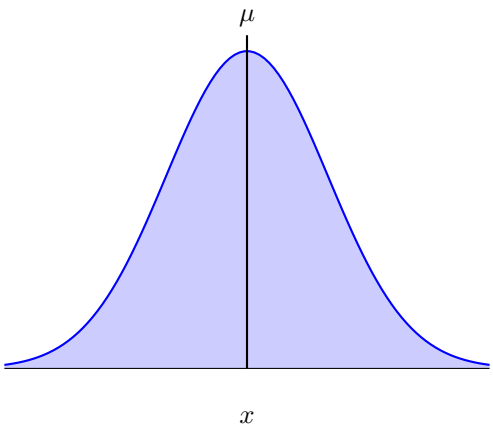
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

donde:

- σ es la desviación típica.
- σ^2 es la varianza.
- μ es la media.

La distribución normal tiene las siguientes propiedades importantes:

- Simetría: La distribución normal es simétrica alrededor de su media, lo que significa que la mitad de la distribución está a la izquierda de la media y la otra mitad está a la derecha.
- Media, mediana y moda iguales: En una distribución normal, la media, la mediana y la moda son todas iguales y están ubicadas en el mismo punto, que es la media de la distribución.
- La forma de campana: La gráfica de la distribución normal tiene una forma de campana, con la densidad de probabilidad más alta en la media y disminuye simétricamente a medida que nos alejamos de la media, es decir, tiene una distribución unimodal.
- Comportamiento asintótico en las colas: En las colas de la distribución normal, las curvas tienden asintóticamente hacia el eje x. Esto significa que a medida que nos alejamos infinitamente de la media, la probabilidad de que un valor caiga fuera de un rango dado se acerca a cero, pero nunca alcanza realmente cero.



La función **pnorm()** calcula la función de distribución acumulativa de la distribución normal, es decir, calcula $P(X \leq x)$.

```
pnorm(q = 62, mean = 70, sd = 7, lower.tail = TRUE)
```

```
## [1] 0.126549
```

Donde:

- `q` es el valor en el que se evalúa la probabilidad acumulativa.
- `mean` es la media de la distribución normal.
- `sd` es la desviación estándar de la distribución normal.
- `lower.tail` es un parámetro booleano opcional que especifica si se calcula la probabilidad acumulativa en la "cola inferior" de la distribución o en la "cola superior". Por defecto, `lower.tail` = `TRUE` calcula la probabilidad de que el número de éxitos sea menor o igual a `q`. Funciona igual que con **punif()** y **pbinom()**.

La función **dnorm()** permite conocer la probabilidad de que la variable aleatoria tome un valor concreto dado un número de ensayos y la probabilidad de éxito $P(X = x)$.

```
dnorm(x = 60, mean = 70, sd = 7)
```

```
## [1] 0.02054255
```

Donde:

- `x` es el valor en el que se evalúa la probabilidad.
- `mean` es la media de la distribución normal.
- `sd` es la desviación estándar de la distribución normal.

La función **rnorm()** se utiliza para generar números aleatorios que siguen una distribución binomial.

```
rnorm(n = 5, mean = 70, sd = 7)
```

```
## [1] 69.56100 79.13409 86.00652 60.27798 68.04848
```

Teorema central del límite

Una muestra aleatoria de tamaño n proveniente de una población con una media μ y una desviación estándar σ.Entonces, cuando n se vuelve suficientemente grande, la distribución de la media muestral $\bar{X}$ se aproxima a una distribución normal con una media μ y una desviación estándar σ.

$$Z = \lim_{n\to\infty} \left(\frac{\bar{X}_n - \mu}{\sigma_{\bar{X}}} \right), \text{ con } \sigma_{\bar{X}} = \sigma/\sqrt{n}$$

Para estudiar dicho teorema en R, resulta de gran utilidad la función **replicate()**. Esta función repite un número n de veces una expresión especificada en ella.

En el siguiente ejemplo, se muestra la aplicación del teorema central del límite al experimento aleatorio de calcular la media de cinco tiradas de un dado sin trucar de seis caras; se puede observar la similitud cada vez más palpable de la distribución con una normal con el aumento de muestras:

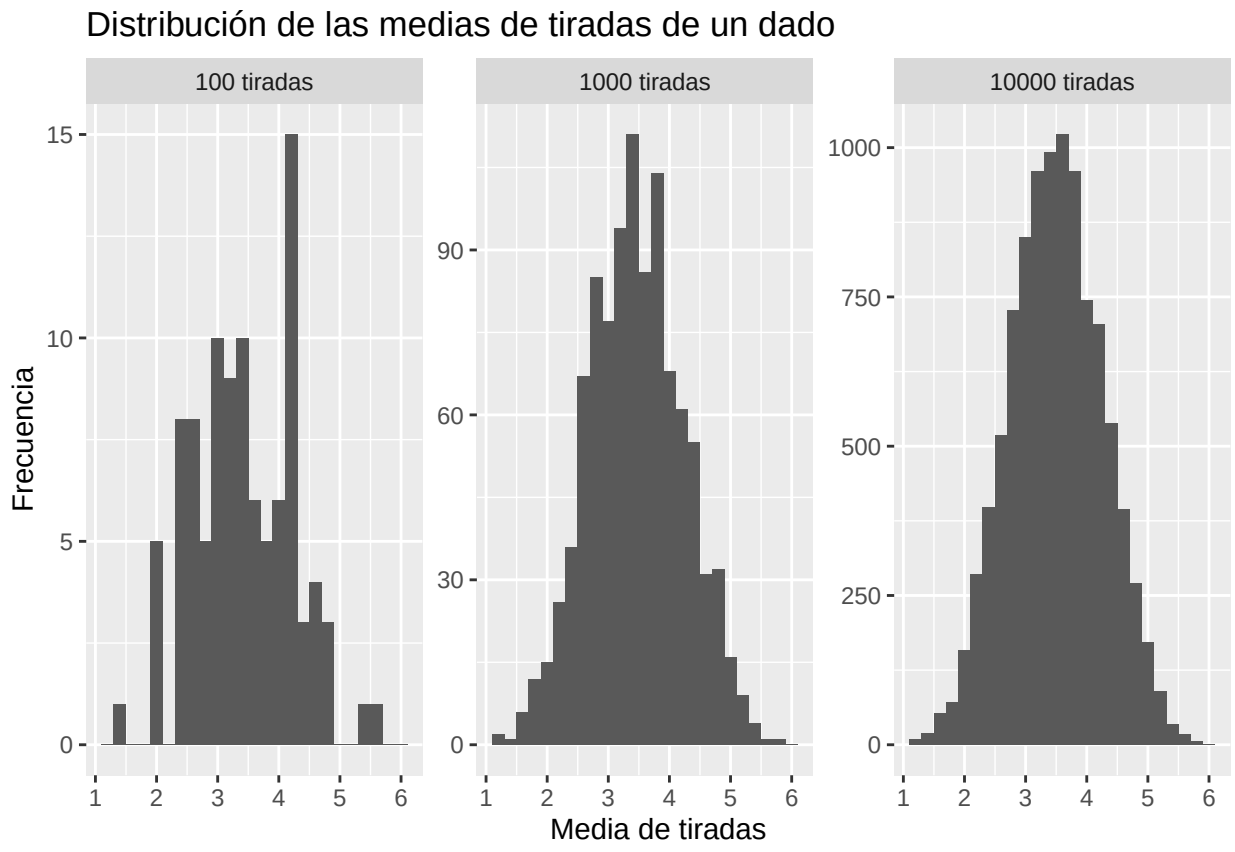
```r
set.seed(42)
dado <- 1:6

generar_datos <- function(n) {
  replicate(n, sample(dado, 5, replace = TRUE) %>% mean())
}

datos_100 <- data.frame(tlm = generar_datos(100), grupo = '100 tiradas')
datos_1000 <- data.frame(tlm = generar_datos(1000), grupo = '1000 tiradas')
datos_10000 <- data.frame(tlm = generar_datos(10000), grupo = '10000 tiradas')

datos <- rbind(datos_100, datos_1000, datos_10000)

ggplot(datos, aes(x = tlm)) +
  geom_histogram(bins = 25) +
  facet_wrap(~ grupo, scales = "free_y") +
  ggtitle("Distribución de las medias de tiradas de un dado") +
  xlab("Media de tiradas") +
  ylab("Frecuencia")
```



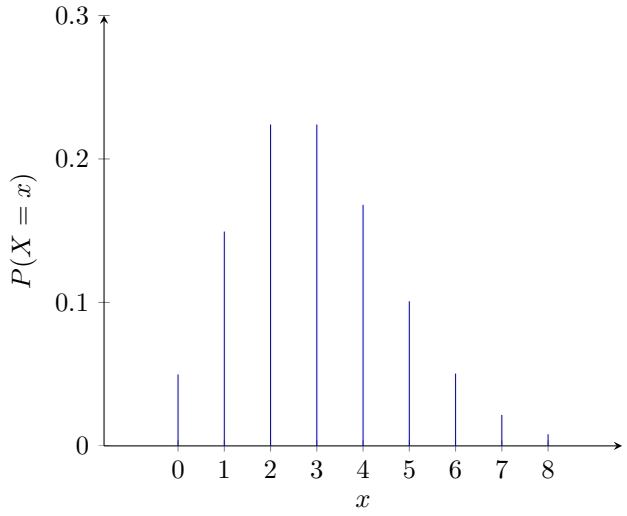
Distribución de Poisson

La distribución de Poisson es una distribución de probabilidad discreta que expresa, a partir de una frecuencia de ocurrencia media, la probabilidad de que ocurra un determinado número de eventos durante cierto período de tiempo. Su función de probabilidad viene definida por la expresión:

$$f(k;\lambda) = \Pr(X{=}k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Donde:

- λ es la media del número de sucesos en el intervalo que estemos tomando.
- k es el número de ocurrencias del suceso.



La función **ppois()** calcula la función de distribución acumulativa de la distribución normal, es decir, calcula $P(X \leq x)$.

```
ppois(q = 62, lambda = 70, lower.tail = TRUE)
```

```
## [1] 0.1859671
```

Donde:

- `q` es el valor en el que se evalúa la probabilidad acumulativa.
- `lambda` es la media de la frecuencia de las ocurriencias.
- `lower.tail` es un parámetro booleano opcional que especifica si se calcula la probabilidad acumulativa en la "cola inferior" de la distribución o en la "cola superior". Por defecto, `lower.tail` = `TRUE` calcula la probabilidad de que el número de éxitos sea menor o igual a `q`. Funciona igual que en las anteriores distribuciones.

La función **dpois()** permite conocer la probabilidad de que la variable aleatoria tome un valor concreto dado un número de ensayos y la probabilidad de éxito $P(X = k)$.

```
dpois(x = 60, lambda = 70)
```

```
## [1] 0.02427134
```

Donde:

- `x` es el valor en el que se evalúa la probabilidad.
- `lambda` es la media de la frecuencia de las ocurriencias.

La función **rpois()** se utiliza para generar números aleatorios que siguen una distribución de Poisson.

```
rpois(n = 5, lambda = 70)
```

```
## [1] 83 53 60 58 67
```

Correlación

La correlación de Pearson mide la fuerza y dirección de una relación lineal entre dos variables continuas. Esta correlación varía entre -1 y 1:

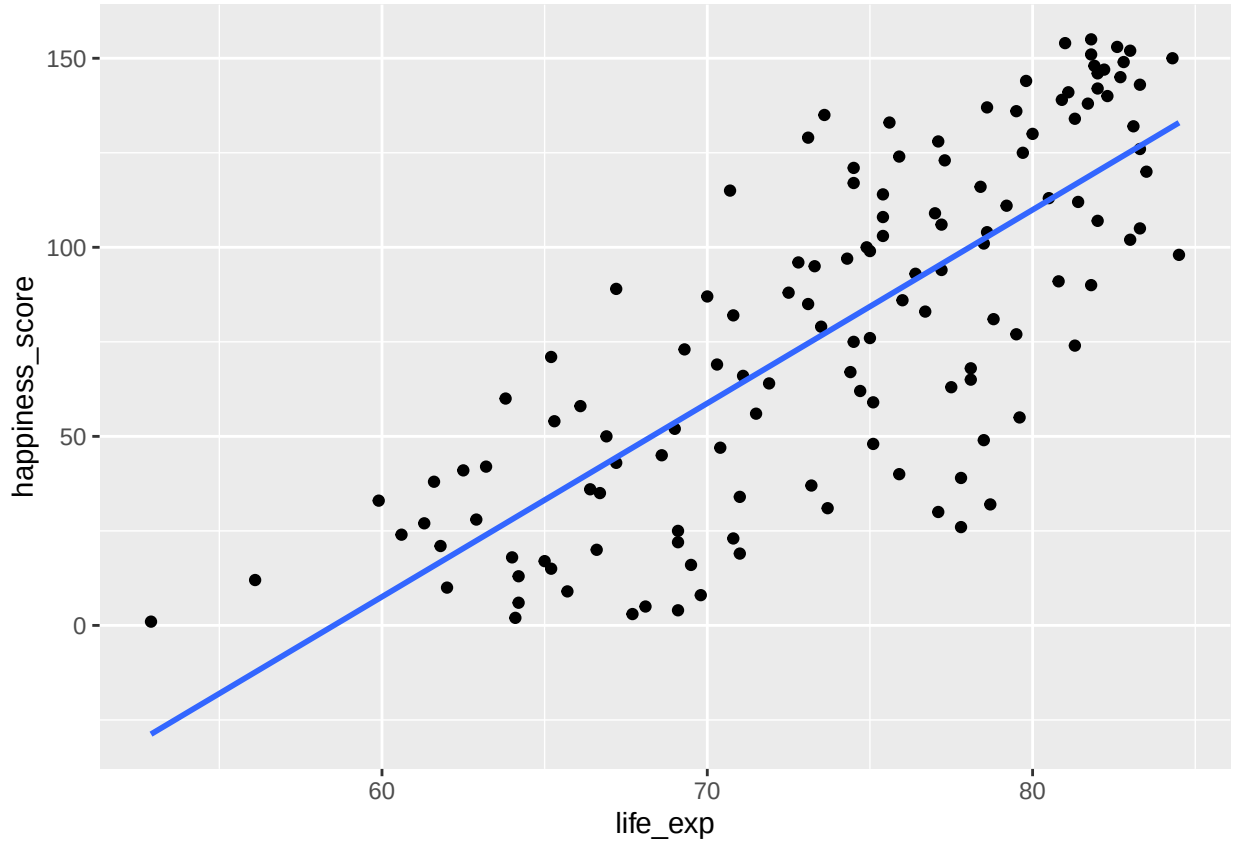
- Una correlación de 1 indica una correlación positiva perfecta: cuando una variable aumenta, la otra también lo hace de manera proporcional.
- Una correlación de -1 indica una correlación negativa perfecta: cuando una variable aumenta, la otra disminuye de manera proporcional.
- Una correlación de 0 indica ausencia de correlación lineal.

Una manera de visualizar la correlación entre dos variables es graficando la nube de puntos. Además, al aplicar la función geom_smooth() se grafica una línea de tendencia suavizada en el gráfico de dispersión. Dicha función tiene dos argumentos de suma importancia:

- `method`: especifica el método que se emplea para ajustar la línea de tendencia. El valor `"lm"` indica que se utiliza el método de mínimos cuadrados lineales.
- `se`: argumento que toma valores booleanos y controla si se muestra o no el intervalo de confianza alrededor de la línea de tendencia.

```
ggplot(world_happiness, aes(life_exp, happiness_score)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

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



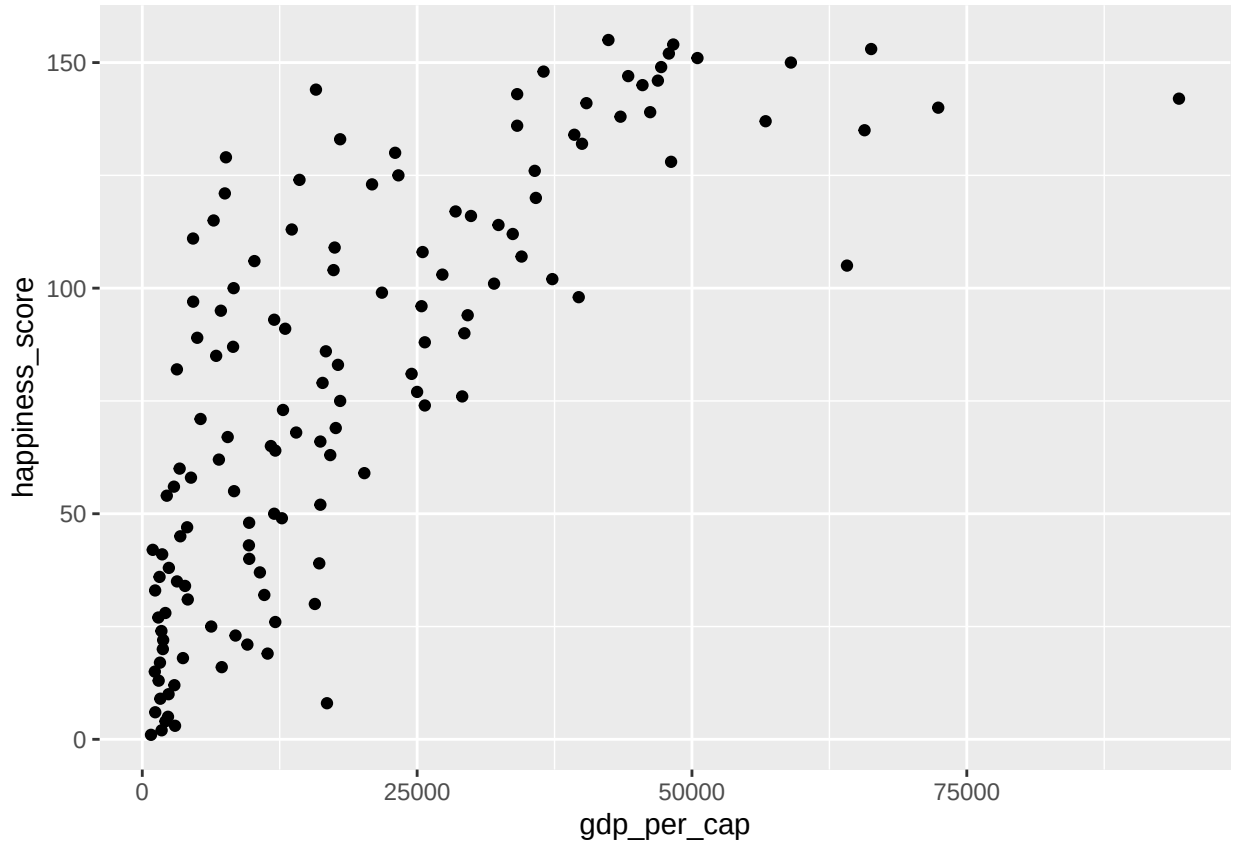
Y la función **cor()** calcula el valor numérico de dicha correlación:

```
cor(world_happiness$life_exp, world_happiness$happiness_score)
```

```
## [1] 0.7737615
```

Es importante destacar que el índice de correlación de Pearson solo mide la correlación lineal entre dos variables. Por lo tanto, puede ser necesario aplicar técnicas de transformación de datos, como pasarlos a escala logarítmica con **log()**, aplicar la raíz cuadrada con **sqrt()**, o la transformación recíproca $\frac{1}{x}$ con $\mathbf{1/x}$.

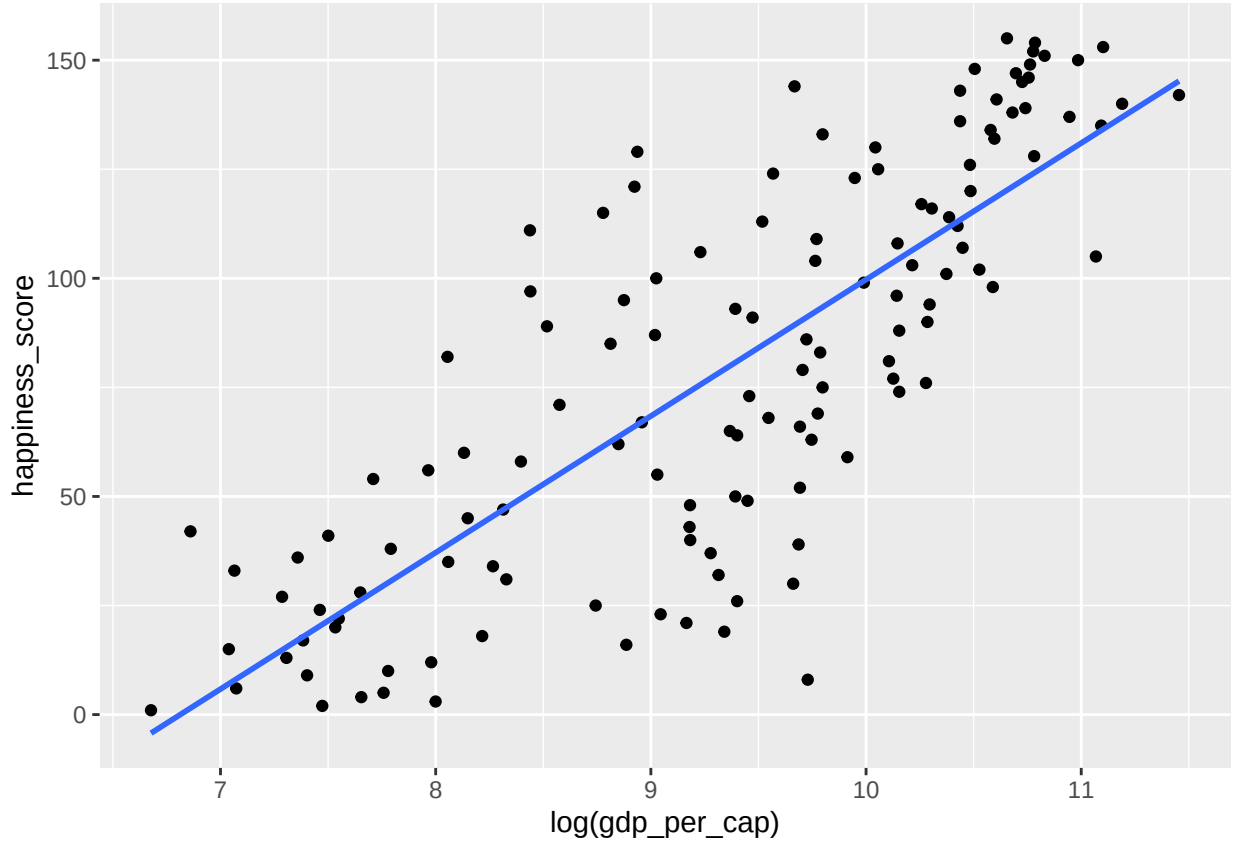
```
ggplot(world_happiness, aes(x = gdp_per_cap, y = happiness_score)) +
    geom_point()
```



La gráfica anterior no muestra una relación lineal; debido a su forma que se asemeja a una función logarítmica, resulta interesante considerar una transformación logarítmica.

```
ggplot(world_happiness, aes(x = log(gdp_per_cap), y = happiness_score)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE)
```

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




```
cor(world_happiness$happiness_score, log(world_happiness$gdp_per_cap))
```

```
## [1] 0.7965484
```