Introduction to Statistics with R

Luis M.

5/7/2021

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
food_consumption <- readRDS("/Users/luismoreno/Documents/R/datacamp/Career Tracks/Statistician with R/da

amir_deals <- readRDS("/Users/luismoreno/Documents/R/datacamp/Career Tracks/Statistician with R/datasets

world_happiness <- readRDS("/Users/luismoreno/Documents/R/datacamp/Career Tracks/Statistician with R/da
```

Summary Statistics

- La estadística es la práctica y el estudio de la recolección y análisis de datos.
  - La **estadística descriptiva** describe y resume los datos.
  - La **estadística inferencial** utiliza una muestra de datos para inferir sobre una mayor población.
- Tipos de datos:
  - Numéricos (**cuantitativos**):
    - Continuos (medidas)
    - Discretos (conteos)
  - Categóricos (**cualitativos**):
    - Nominales (sin orden):
      - Pueden representarse numéricamente con binarios (0 y 1) para indicar ausencia o presencia y dígitos (0 a 9) para enumerar casos o eventos.
  - Ordinales (ordenados):
    - Pueden representarse numéricamente (0 a 9) para representar escalas de orden (mayor o menor).

Medidas de Tendencia Central

- **Media**: es el valor comprendido entre el mínimo y el máximo del conjunto de los datos. Su uso es preferible cuando existen datos simétricos,
- **Mediana**: es el valor donde el 50% de los datos es menor a este y el 50% restante de los datos es mayor. Su uso es preferible cuando existen datos sesgados o no simétricos.
- **Moda**: es el valor más frecuente dentro del conjunto de datos, en otras palabras, es el valor que más se repite.

```
# Filter for Belgium
belgium_consumption <- food_consumption %>%
  filter(country == "Belgium")

head(belgium_consumption)
```

```
## # A tibble: 6 x 4
##   country food_category consumption co2_emission
##   <chr>   <fct>               <dbl>        <dbl>
## 1 Belgium pork                38.6         137.
## 2 Belgium poultry             12.2          13.1
## 3 Belgium beef                15.6         482.
## 4 Belgium lamb_goat            1.32         46.2
## 5 Belgium fish                19.0          30.3
## 6 Belgium eggs                12.6          11.6
```

```
# Filter for USA
usa_consumption <- food_consumption %>%
  filter(country == "USA")

head(usa_consumption)
```

```
## # A tibble: 6 x 4
##   country food_category consumption co2_emission
##   <chr>   <fct>               <dbl>        <dbl>
## 1 USA     pork                27.6          97.8
## 2 USA     poultry             50.0          53.7
## 3 USA     beef                36.2        1118.
## 4 USA     lamb_goat            0.43         15.1
## 5 USA     fish                12.4          19.7
## 6 USA     eggs                14.6          13.4
```

```
# Calculate mean and median consumption in Belgium
paste("The average consumption in Belgium is",
      round(mean(belgium_consumption$consumption),2))
```

```
## [1] "The average consumption in Belgium is 42.13"
```

```
paste("The median consumption in Belgium is",
      round(median(belgium_consumption$consumption),2))
```

```
## [1] "The median consumption in Belgium is 12.59"
```

```
# Calculate mean and median consumption in USA
mean(usa_consumption$consumption)
```

```
## [1] 44.65
```

```
median(usa_consumption$consumption)
```

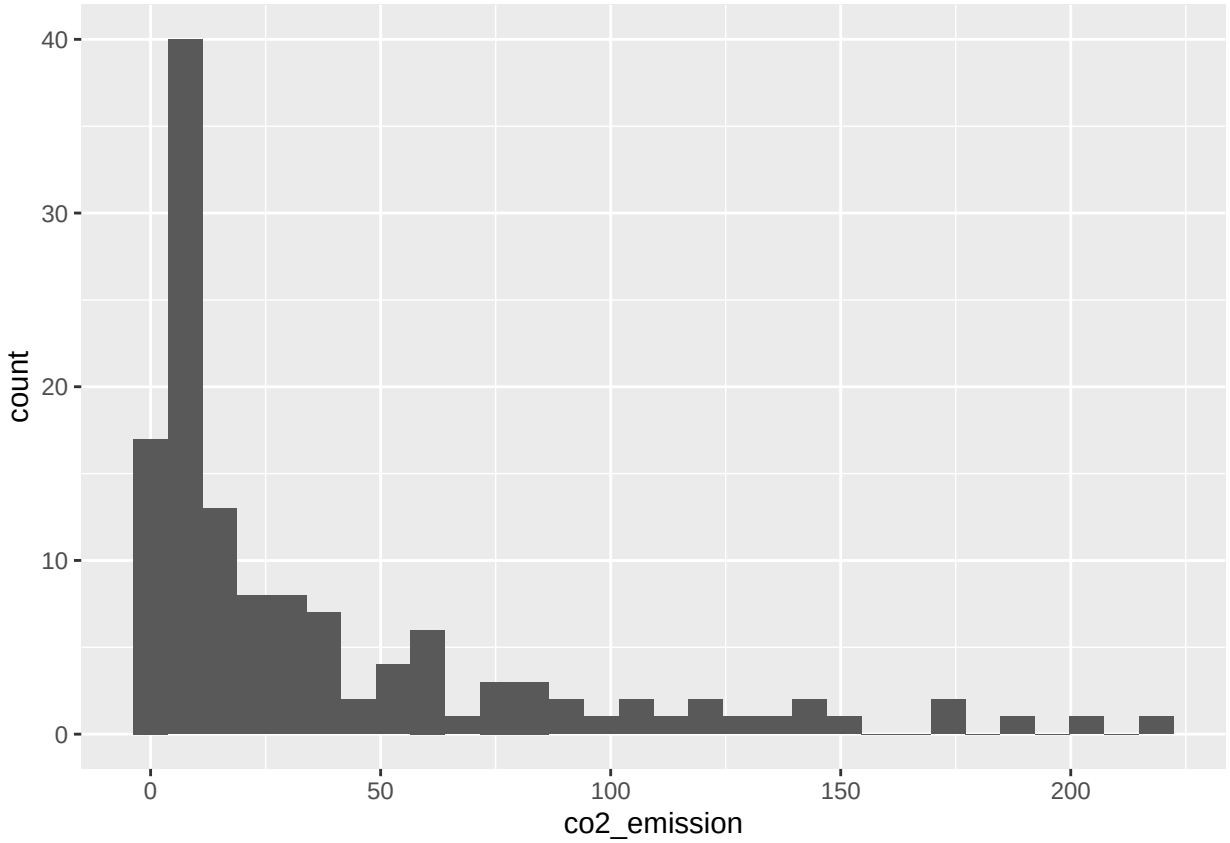
```
## [1] 14.58
```

```
food_consumption %>%
  # Filter for Belgium and USA
  filter(country %in% c("Belgium", "USA")) %>%
  # Group by country
  group_by(country) %>%
  # Get mean_consumption and median_consumption
  summarize(mean_consumption = mean(consumption),
      median_consumption = median(consumption))
```

```
## # A tibble: 2 x 3
##   country mean_consumption median_consumption
##   <chr>              <dbl>              <dbl>
## 1 Belgium             42.1               12.6
## 2 USA                 44.6               14.6
```

```
food_consumption %>%
  # Filter for rice food category
  filter(food_category == "rice") %>%
  # Create histogram of co2_emission
  ggplot(aes(co2_emission)) +
    geom_histogram()
```

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




```
food_consumption %>%
  # Filter for rice food category
  filter(food_category == "rice") %>%
  # Get mean_co2 and median_co2
  summarize(mean_co2 = mean(co2_emission),
            median_co2 = median(co2_emission))
```

```
## # A tibble: 1 x 2
##   mean_co2 median_co2
##      <dbl>      <dbl>
## 1     37.6       15.2
```

Medidas de Dispersión

Las medidas de dispersión ayudan a describir qué tan cerca o dispersos están los puntos de datos.

- **Varianza**: mide la distancia promedio entre cada dato y su valor promedio. Mientras más grande sea la varianza, más dispersos son los datos.
- **Desviación estándar**: se cálcula aplicando la raíz cuadrada de la varianza.
- **Desviación media absoluta**: toma el valor absoluto de las distancias entre cada valor del conjunto de datos y el valor promedio, luego se calcula el promedio de dichas diferencias.
- **Cuartiles**: los cuartiles dividen los datos en cuatro partes iguales.

  - El primer cuartil abarca de 0% a 25% de los datos,
  - El segundo cuartil abarca de 25% a 50% de los datos,
  - El tercer cuartil abarca de 50% a 75% de los datos,
  - El cuarto cuartil abarca del 75% al 100% de los datos.

- **Cuantiles**: también llamados percentiles, son una versión generalizada y dividen los datos en cinco o diez partes iguales.
- **Rango intercuartil**: es la distancia entre el percentil 25 y 75, que también es la altura de la caja de un diagrama de caja.
- **Valores atípicos**: son datos extremos en el conjunto de datos. Como regla general se pueden clasificar como cualquier dato menor que el primer cuartil menos 1.5 veces el rango intercuartil, como también cualquier dato mayor que el tercer cuartil más 1.5 veces el rango intercuartil. Aquellos estadísticos que dependen del promedio son afectados por la presencia de valores atípicos.

```
# Calculate the quartiles of co2_emission
quantile(food_consumption$co2_emission)
```

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

```
# Calculate the quintiles of co2_emission
quantile(food_consumption$co2_emission,
         probs = seq(from = 0,
                     to = 1,
                     by = 0.2))
```

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

```
# Calculate the deciles of co2_emission
quantile(food_consumption$co2_emission,
         probs = seq(from = 0,
                     to =1,
                     by = 0.1))
```

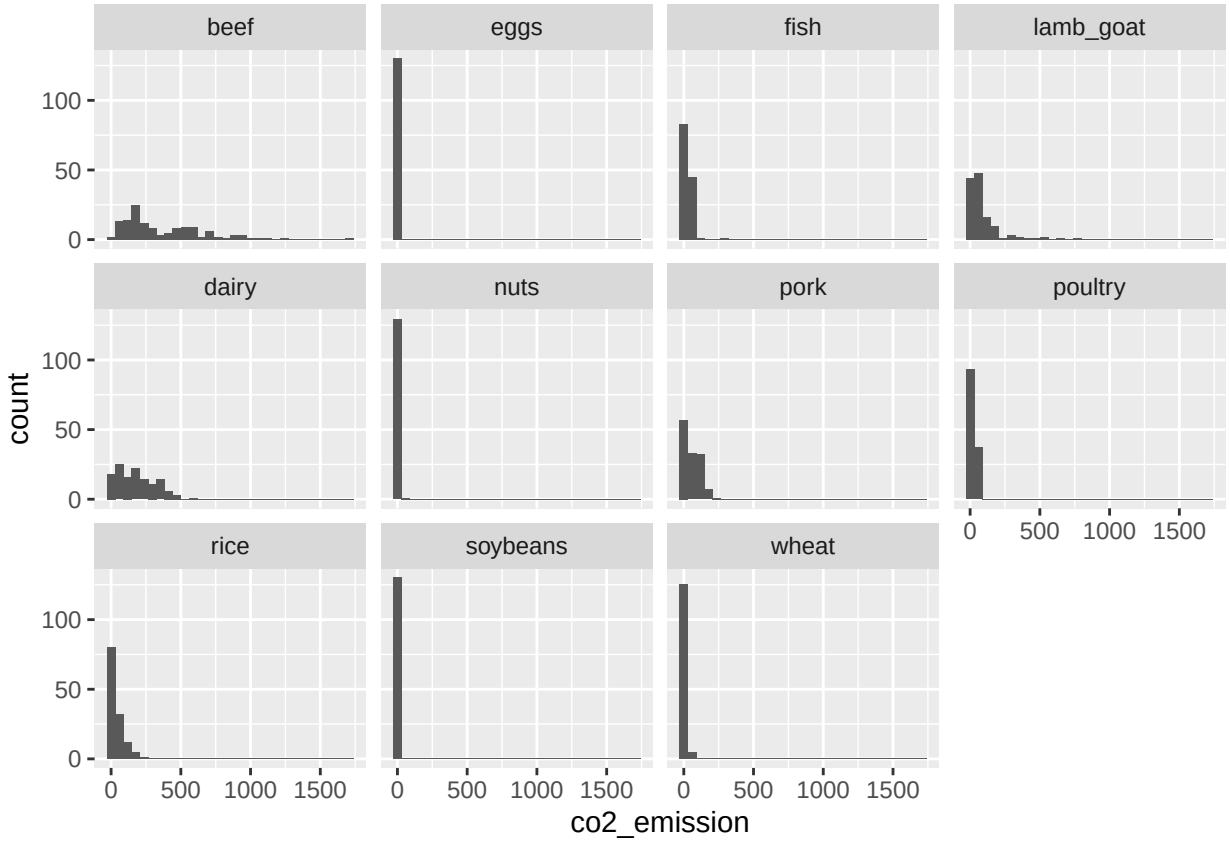
```
##       0%      10%      20%      30%      40%      50%      60%      70%
##    0.000    0.668    3.540    7.040   11.026   16.530   25.590   44.271
##      80%      90%     100%
##   99.978  203.629 1712.000
```

```
# Calculate variance and sd of co2_emission for each food_category
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
```

```
# Plot food_consumption with co2_emission on x-axis
food_consumption %>%
  ggplot(mapping = aes(x = co2_emission)) +
  # Create a histogram
  geom_histogram() +
  # Create a separate sub-graph for each food_category
  facet_wrap(~ food_category)
```

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



```r
# Calculate total co2_emission per country: emissions_by_country
emissions_by_country <- food_consumption %>%
  group_by(country) %>%
  summarize(total_emission = sum(co2_emission))

# Compute the first and third quantiles and IQR of total_emission
q1 <- quantile(emissions_by_country$total_emission, 0.25)
q3 <- quantile(emissions_by_country$total_emission, 0.75)
iqr <- q3 - q1

# Calculate the lower and upper cutoffs for outliers
lower <- q1 - 1.5 * iqr
upper <- q3 + 1.5 * iqr

# Filter emissions_by_country to find outliers
emissions_by_country %>%
  filter(total_emission < lower | total_emission > upper)
```

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

Random Numbers and Probability

- We can calculate the probability of some event by taking the number of ways the event can happen and dividing it by the total number of possible outcomes.
- Probability is always between zero and 100 percent. If the probability of something is zero, it's impossible, and if the probability of something is 100%, it will certainly happen.
- `sample()` function can recreate scenarios where a process of sampling randomly is involved. Sampling can be done with or without replacement, this is done by setting the `replace` argument as `TRUE` OR `FALSE`.
- To ensure same results in R when running random experiments, a random seed must be set, using `set.seed()` function. The seed is a number that R's random number generator uses as a starting point, so if we orient it with a seed number, it will generate the same random value each time.
- Two events are **independent** if the probability of the second event isn't affected by the outcome of the first event.
- Events are considered **dependent** when the outcome of the first changes the probability of the second.
- Because of different ways of taking **samples**, it is not easy to tell which **best fits** various situations, it's important to correctly identify this so that any probabilities you report are **accurate** you report are accurate.

Calculating Simple Probabilities

```
amir_deals %>%
  # Count the number of deals by each product
  count(product) %>%
  # Create a new column that displays the probability of picking a deal with each product
  mutate(prob = n / sum(n))
```

```
##      product  n       prob
## 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
```

$$P(\text{event}) = \frac{\#\text{ ways event can happen}}{\text{total }\#\text{ of possible outcomes}}$$

Figure 1: General Probability Formula

Sampling

```
# Set random seed to 31
set.seed(31)

# Sample 5 deals without replacement
amir_deals %>%
  sample_n(size = 5,
           replace = FALSE)
```

```
##     product  client status  amount num_users
## 1 Product D Current   Lost 3086.88        55
## 2 Product C Current   Lost 3727.66        19
## 3 Product D Current   Lost 4274.80         9
## 4 Product B Current    Won 4965.08         9
## 5 Product A Current    Won 5827.35        50
```

```
# Sample 5 deals with replacement
amir_deals %>%
  sample_n(size = 5,
           replace = TRUE)
```

```
##     product  client status  amount num_users
## 1 Product A Current    Won 6010.04        24
## 2 Product B Current   Lost 5701.70        53
## 3 Product D Current    Won 6733.62        27
## 4 Product F Current    Won 6780.85        80
## 5 Product C Current    Won -539.23        11
```

Distributions and Central Limit Theorem

Discrete Distributions

- A probability distribution describes the probability of each possible outcome in a scenario.
- The expected value of a distribution is the mean of a distribution.
- Probabilities of different outcomes can be calculated by taking areas of the probability distribution.

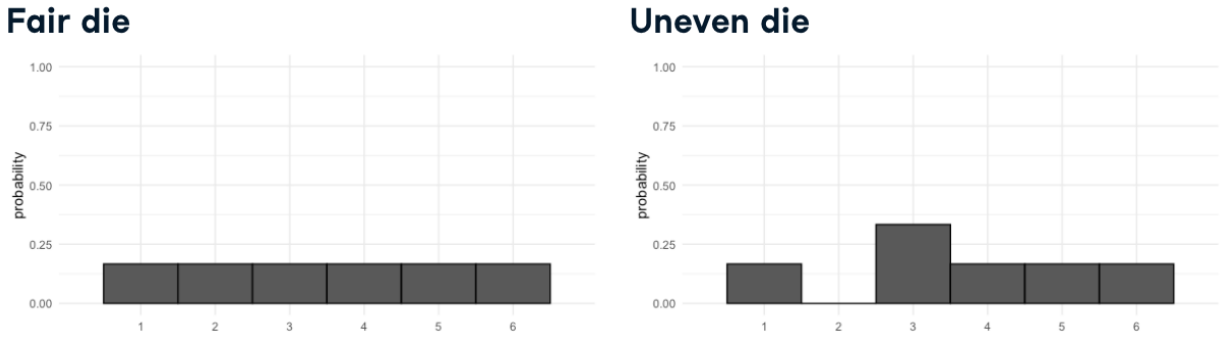


Figure 2: General Probability Formula

- Discrete probability distributions represent situations with discrete outcomes.
- When all outcomes have the same probability, like a fair die, this is a special distribution called a **discrete uniform distribution**.
- Using a **histogram** is a way of visualizing the outcomes of an event.
- The `bins` parameter of the `geom_histogram()` is used to specify the number of outcomes for each event.
- The **law of large numbers** is the idea that as the size of your sample increases, the sample mean will approach the **theoretical mean**.

Probability Distributions

```
# Create probability distribution
size_distribution <- restaurant_groups %>%
  # Count the number of each group size
  count(group_size) %>%
  # Calculate probability
  mutate(probability = n / sum(n))
```

```
# Calculate probability of picking group of 4 or more
size_distribution %>%
  # Filter for groups of 4 or larger
  filter(group_size >= 4) %>%
  # Calculate prob_4_or_more by taking sum of probabilities
  summarize(prob_4_or_more = sum(probability))
```

There are two different variants of the uniform distribution: the **discrete uniform distribution**, and the **continuous uniform distribution**.

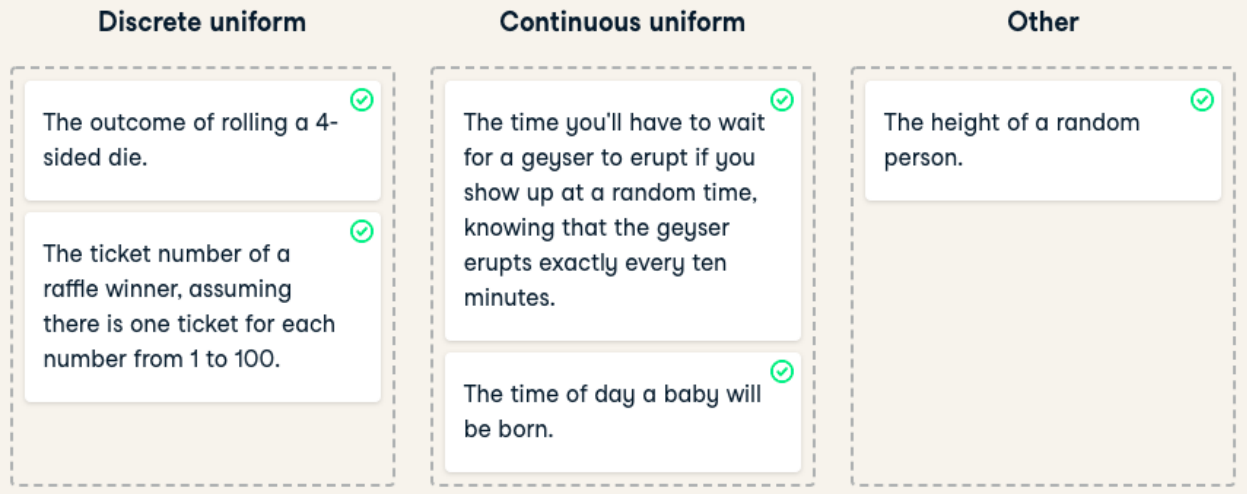


Figure 3: Examples of Distributions

Uniform Distribution

A sales software is set to automatically back itself up. Back-ups happen exactly every 30 minutes. At random times the data is updated, so how long you'll have to wait for the newly-entered data to get backed up?

```
# Min and max wait times for back-up that happens every 30 min
min <- 0
max <- 30

# Calculate probability of waiting less than 5 mins
prob_less_than_5 <- punif(5, min, max)
prob_less_than_5
```

```
## [1] 0.1666667
```

```
# Calculate probability of waiting more than 5 mins
prob_greater_than_5 <- punif(5, min, max, lower.tail = FALSE)
prob_greater_than_5
```

```
## [1] 0.8333333
```

```r
# Calculate probability of waiting 10-20 mins
prob_between_10_and_20 <- punif(20, min, max) - punif(10, min, max)
prob_between_10_and_20
```

```
## [1] 0.3333333
```

- `punif()`: gives the distribution function about a uniform distribution on the interval from `min` to `max`.
  - `lower.tail` if `TRUE` the probability is **less than**, if `FALSE` the probability is **greater than**.

```r
# Set random seed to 334
set.seed(334)

# Generate 1000 wait times between 0 and 30 mins, save in time column
wait_times %>%
  mutate(time = runif(1000, min = 0, max = 30)) %>%
  # Create a histogram of simulated times
  ggplot(aes(x = time)) +
  geom_histogram()
```

Binomial Distribution

The binomial distribution is a probability distribution of the number of successes in a sequence of **independent** trials.

- `rbinom()` is a random generator for the binomial distribution.
  - `n` argument is the number of observations,
  - `size` argument is the number of trials per observation,
  - `prob` argument is the probability of success on each trial.

`rbinom(x = 8, size 1, prob = 0.5` can be read as "8 flips of 1 coin with 50% chance of success".

- `dbinom()` function calculates probabilities for binomial distributions.
  - `x` argument is the value we want to calculate its probability,
  - `size` argument is the number of trials,
  - `prob` argument is the probability of success on each trial.
- `pbinom()` function is used to calculate the probability **less than or equal** to any discrete value. It uses the **same arguments** as `dbinom()`.
  - `lower.tail = FALSE` for **greater than** probabilities.

Example: A seller works 3 deals per week, and overall, he wins 30% of deals he works on. Each deal can be either lost or won.

```r
# Set random seed to 10
set.seed(10)

# Simulate a single deal
rbinom(1, 1, 0.3)
```

```
## [1] 0
```

```r
# Simulate 1 week of 3 deals
rbinom(1, 3, 0.3)
```

```
## [1] 0
```

```r
# Simulate 52 weeks of 3 deals
deals <- rbinom(52, 3, 0.3)

# Calculate mean deals won per week
mean(deals)
```

```
## [1] 0.8076923
```

```r
# Probability of closing 3 out of 3 deals
dbinom(3, 3, 0.3)
```

```
## [1] 0.027
```

```r
# Probability of closing <= 1 deal out of 3 deals
pbinom(1, 3, 0.3)
```

```
## [1] 0.784
```

```r
# Probability of closing > 1 deal out of 3 deals
pbinom(1, 3, 0.3, lower.tail = FALSE)
```

```
## [1] 0.216
```

```r
# Expected number won with 30% win rate
won_30pct <- 3 * 0.3
won_30pct
```

```
## [1] 0.9
```

```r
# Expected number won with 25% win rate
won_25pct <- 3 * 0.25
won_25pct
```

```
## [1] 0.75
```

```r
# Expected number won with 35% win rate
won_35pct <- 3 * 0.35
won_35pct
```

```
## [1] 1.05
```

Normal Distribution

- It is symmetrical, so the left side is a mirror image of the right,
- The area beneath the curve is 1,
- The probability never reaches 0,
- Only 0.006% of its area is contained beyond the edges of the curve,
- It is described by its **mean** and **standard deviation**,
- When the `mean = 0` and `standard deviation = 1` it is known as **standard normal distribution**,
- 68% of the area is within 1 standard deviation of the mean,
- 95% of the area is within 2 standard deviations of the mean,
- 99.7% of the area falls within 3 standard deviations of the mean.
- `pnorm()` function is the **distribution function** for **normal distributions**. It returns the percentage of the data that is **less than** the given value `q`.
  - `q` argument is the value we are trying to calculate its probability,
  - `mean` argument is the mean of the dataset,
  - `sd` argument is the standard deviation of the dataset,
  - `lower.tail = FALSE` argument is used for calculating probabilities **greater than** a value.
  - To calculate the probability or percentage of the data between a range, we subtract the probability of each value of the range.
  - `pnorm(154, mean = 161, sd = 7)` returns "the percentage of data that is less than 154, given a mean of 161 and standard deviation of 7".
- `qnorm()` function is the quantile function for normal distributions. It returns the value that is **less than** the specified quantile or percentage `q`.
  - `p` argument is the quantile or percentage where the value is at,
  - `mean` argument is the mean of the dataset,
  - `sd` is the standard deviation of the dataset,
  - `lower.tail = FALSE` argument is used for calculating values **greater than** a specified quantile or percentage.
- `rnom()` functions generates `n` number of observations with a specific mean and standard deviation.
  - `n` argument is the number of random numbers we want to generate,
  - `mean` is the mean of the normal distribution,
  - `sd` is the standard deviation of the normal distribution.

Example 1: The amount of each deal Amir worked on had a different amount of money. Considering a normal distribution with a mean of 5,000 and standard deviation of 2,000 dollars, calculate the probabilities of Amir closing a deal worth various amounts.

```r
# Probability of deal < 7500
pnorm(7500, mean = 5000, sd = 2000)
```

```
## [1] 0.8943502
```

```r
# Probability of deal > 1000
pnorm(1000, mean = 5000, sd = 2000, lower.tail = FALSE)
```

```
## [1] 0.9772499
```

```r
# Probability of deal between 3000 and 7000
pnorm(7000, mean = 5000, sd = 2000) - pnorm(3000, mean = 5000, sd = 2000)
```

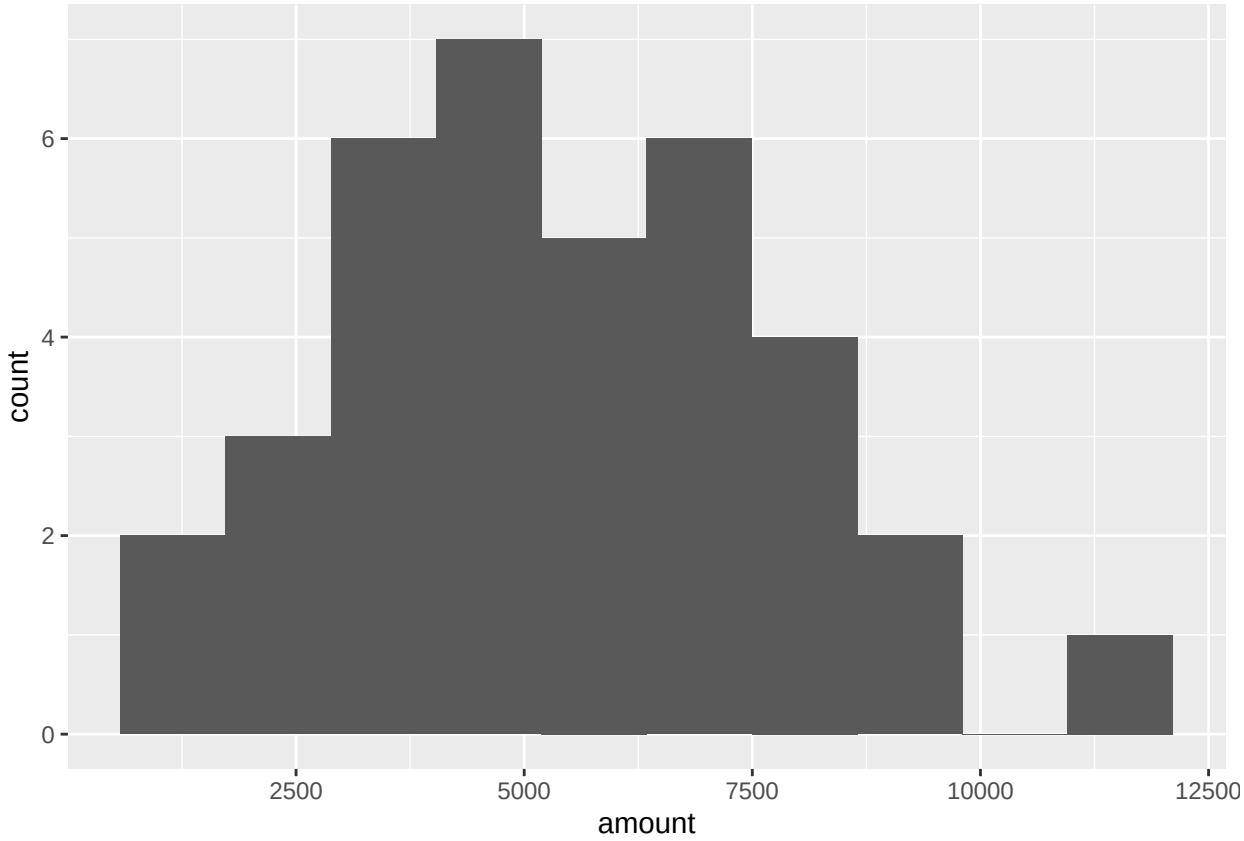
```
## [1] 0.6826895
```

```r
# Calculate amount that 75% of deals will be more than
qnorm(0.75, mean = 5000, sd = 2000, lower.tail = FALSE)
```

```
## [1] 3651.02
```

Example 2: The worth of each sale will increase by 20% and the volatility, or standard deviation, of each sale's worth will increase by 30%. Simulate new sales amounts.

```r
# Calculate new average amount
new_mean <- 5000 * (1 + 0.2)

# Calculate new standard deviation
new_sd <- 2000 * (1 + 0.3)

# Create a sales dataframe
new_sales <- data.frame(sale_nums = 1:36)

# Simulate 36 sales by using average and standard deviation
new_sales <- new_sales %>%
  mutate(amount = rnorm(36, mean = new_mean, sd = new_sd))

# Create histogram with 10 bins
new_sales %>%
  ggplot(aes(x = amount)) +
  geom_histogram(bins = 10)
```



Example 3: The key metric that the company uses to evaluate salespeople is the percent of sales they make over $1000 since the time put into each sale is usually worth a bit more than that.

Amir's current sales amounts have a mean of 5000 and a standard deviation of 2000, and Amir's predicted amounts in next quarter's market have a mean of 6000 and a standard deviation of 2600.

Based only on the metric of percent of sales over $1000, does Amir perform better in the current market or the predicted market?

```
# Calculate the probability of current sales being worth more than 1,000
current_probability <- pnorm(1000, mean = 5000, sd = 2000, lower.tail = FALSE)

# Calculate the probability of next estimated sales being worth more than 1,000
next_probability <- pnorm(1000, mean = 6000, sd = 2600, lower.tail = FALSE)

# Is current sales probability greater than next estimated sales?
current_probability > next_probability
```

```
## [1] TRUE
```

Central Limit Theorem

- In many situations, when **independent random variables** are added, their properly normalized sum tends toward a **normal distribution** even if the original variables themselves are not normally distributed.

- It implies that **probabilistic and statistical methods** that work for **normal distributions** can be **applicable** to many problems involving **other types of distributions**.
- Regardless of the shape of the distribution you're taking sample means from, the central limit theorem will apply if the sampling distribution contains enough sample means.

```
# Set seed to 104
set.seed(104)

# Sample 20 num_users from amir_deals and take mean
sample(amir_deals$num_users,
       size = 20,
       replace = TRUE) %>%
  mean()

# Repeat the above 100 times
sample_means <- replicate(100, sample(amir_deals$num_users,
                                      size = 20,
                                      replace = TRUE) %>%
                            mean())

# Create data frame for plotting
samples <- data.frame(mean = sample_means)

# Histogram of sample means
samples %>%
  ggplot(aes(x = mean)) +
  geom_histogram(bins = 10)
```

```
# Set seed to 321
set.seed(321)

# Take 30 samples of 20 values of num_users, take mean of each sample
sample_means <- replicate(30, sample(all_deals$num_users, 20) %>% mean())

# Calculate mean of sample_means
sample_means %>% mean()

# Calculate mean of num_users in amir_deals
mean(amir_deals$num_users)
```

Poisson Distribution

- A **Poisson process** is a process where events appear to happen at a certain rate.
- The **Poisson distribution** describes the probability of some number of events happening over a fixed period of time.
- The Poisson distribution is described by **lambda (λ)**, which represents the **average number of events per time period**.

- Lambda changes the shape of the distribution, no matter what, the distribution's peak is always at its lambda value.
- `dpois()` function is the Poisson **density function**, which is used to calculate the **probability** of a **single value**.
  - `x` argument is the value that we want to calculate its probability,
  - `lambda` argument is the vector of (non-negative) means.
- `ppois()` function is the Poisson **distribution function**, which by default calculates the probability of **less than or equal** a given value.
  - `lower.tail` argument is used to calculate probabilities **greater than** a given value.
- `rpois()` function is used to create random numbers following a Poisson distribution.
  - `n` argument is the number of random values to return,
  - `lambda` argument is the vector of non-negative means.

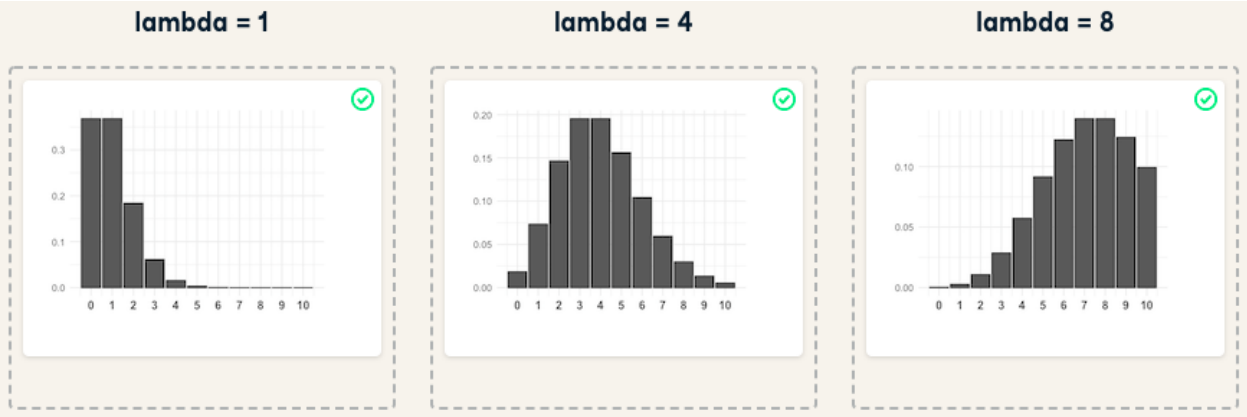


Figure 4: Poisson Comparison by Lambda

Example: the sales software in your company keeps track of new sales leads. Assuming that Amir responds on average to 4 leads each day, calculate the following probabilities.

```
# Probability that Amir responds to 5 leads in a day.
dpois(x = 5, lambda = 4)
```

```
## [1] 0.1562935
```

```
# Probability of Amir responding to 2 or fewer leads in a day.
ppois(q = 2, lambda = 4)
```

```
## [1] 0.2381033
```

```
# Probability of Amir responding to more than 10 leads in a day.
ppois(q = 10, lambda = 4, lower.tail = FALSE)
```

```
## [1] 0.002839766
```

Exponential Distribution

- Represents certain **time** passing between Poisson events. Some examples include:
  - Probability of fewer than 10 minutes between restaurant arrivals,
  - Probability of 6-8 months passing between earthquakes.
- Uses **lambda** as the rate or mean.
- It is **continuous**, since it represents time.
- It measures frequency in terms of time between events.
- The **expected value** is calculated by taking $1/\lambda$.
- `pexp()` distribution function calculates the probability **less than** the given value and its rate.
  - `q` argument is the value which we want to calculate its probability.
  - `rate` argument is the mean or rate of time.
  - `lower.tail = FALSE` argument is used when trying to calculate a probability **greater than** a given value.

Example: you want to know how much time it takes Amir to respond to a lead after he opens it. On average, it takes 2.5 hours for him to respond.

```
# Probability of Amir responding in less than 1 hour
pexp(q = 1, rate = 1/2.5)
```

```
## [1] 0.32968
```

```
# Probability of Amir responding in more than 4 hours
pexp(q = 4, rate = 1/2.5, lower.tail = FALSE)
```

```
## [1] 0.2018965
```

```
# Probability of Amir responding between 3-4 hours
pexp(q = 4, rate = 1/2.5) - pexp(q = 3, rate = 1/2.5)
```

```
## [1] 0.09929769
```

T-Distribution

- Observations are more likely to fall further from the mean.
- **Degrees of freedom** affects the thickness of the distribution tails.
  - Lower degrees of freedom = thicker tails, higher standard deviation.
- The t-distribution is not skewed, just like the normal distribution, but it does have thicker tails and higher variance than the normal distribution.

Log-Normal Distribution

- This distribution describes variables whose logarithm is normally distributed.
  - Examples include: length of chess games, adult blood pressure, number of hospitalizations during the 2003 SARS outbreak.

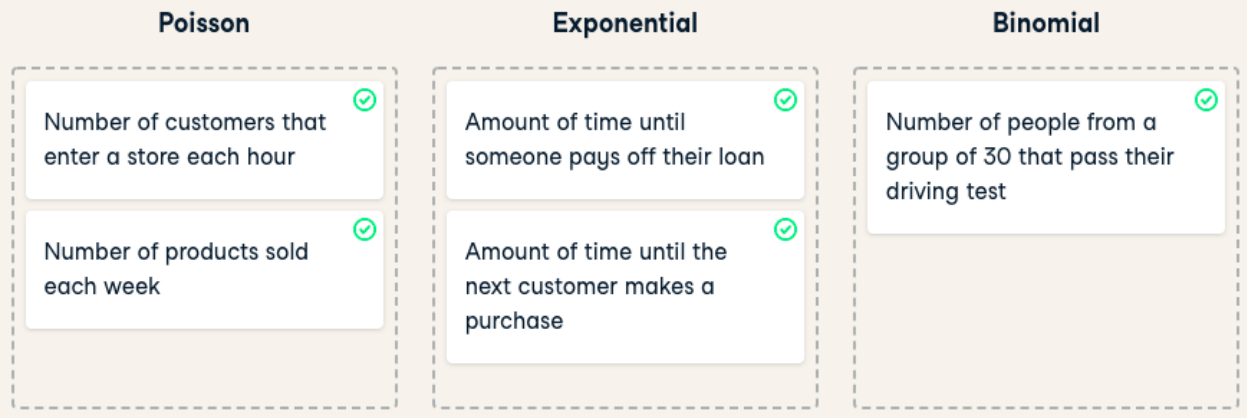


Figure 5: Events and their type of distribution

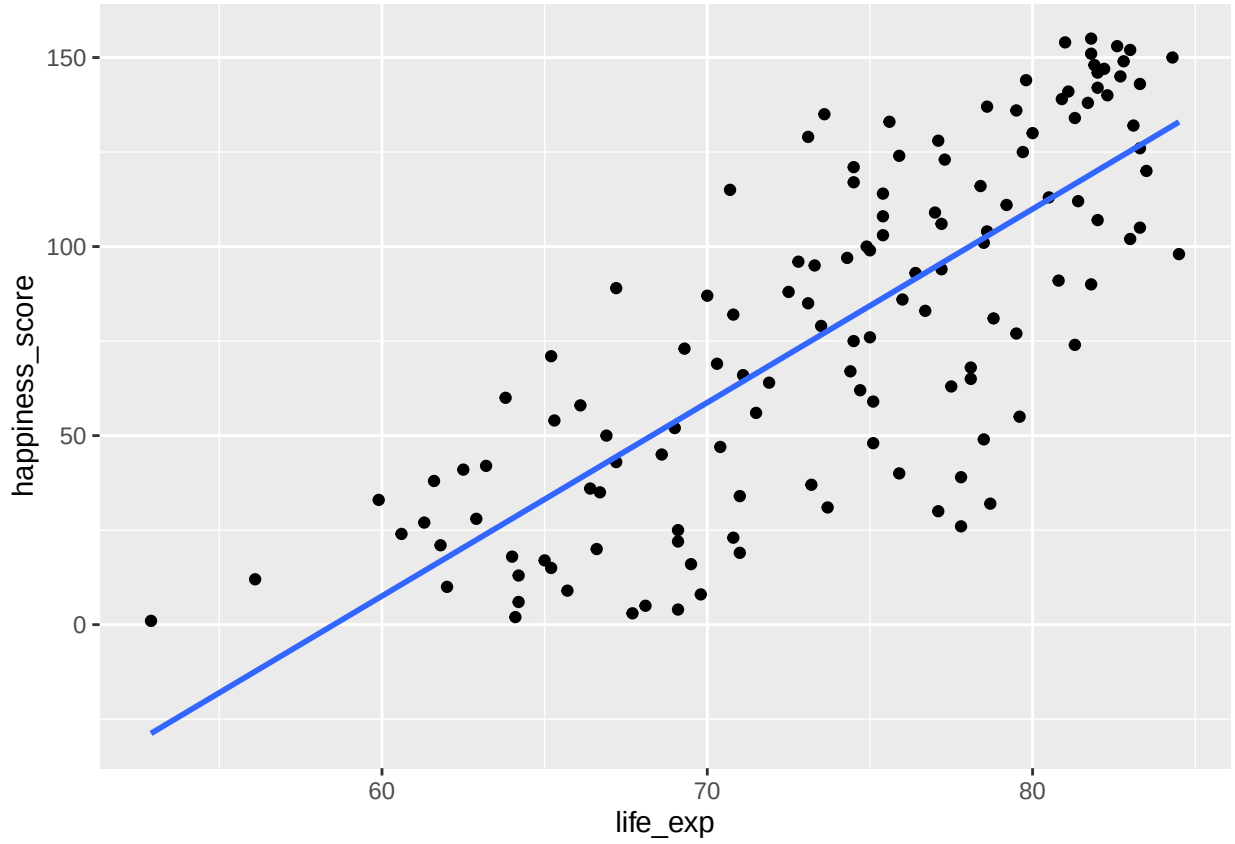
Correlation and Experimental Design

Correlation

- Quantifies the **linear relationship** between two variables:
  - `x` = **explanatory/independent variable**,
  - `y` = **response/dependent variable**.
- **Coefficient** ranges from `-1` to `1`, where `+` is **positive relationship** and `-` **negative relationship**.
- `geom_smooth()`: aids the eye in seeing patterns by drawing a trendline in a scatterplot.
  - `method = "lm"` argument uses a linear model to draw the trendline.
  - `se = FALSE` argument deactivates the standard error.
- `cor()` function is used to calculate correlations.
  - `use = "pairwise.complet.obs` argument is used when there is `NA` values in one of the two datasets.

```
# Create a scatterplot to visualize data distribution
world_happiness %>%
  ggplot(aes(x = life_exp,
             y = happiness_score)) +
  geom_point() +
  # Add a linear trendline to scatterplot
  geom_smooth(method = "lm",
              se = FALSE)
```

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



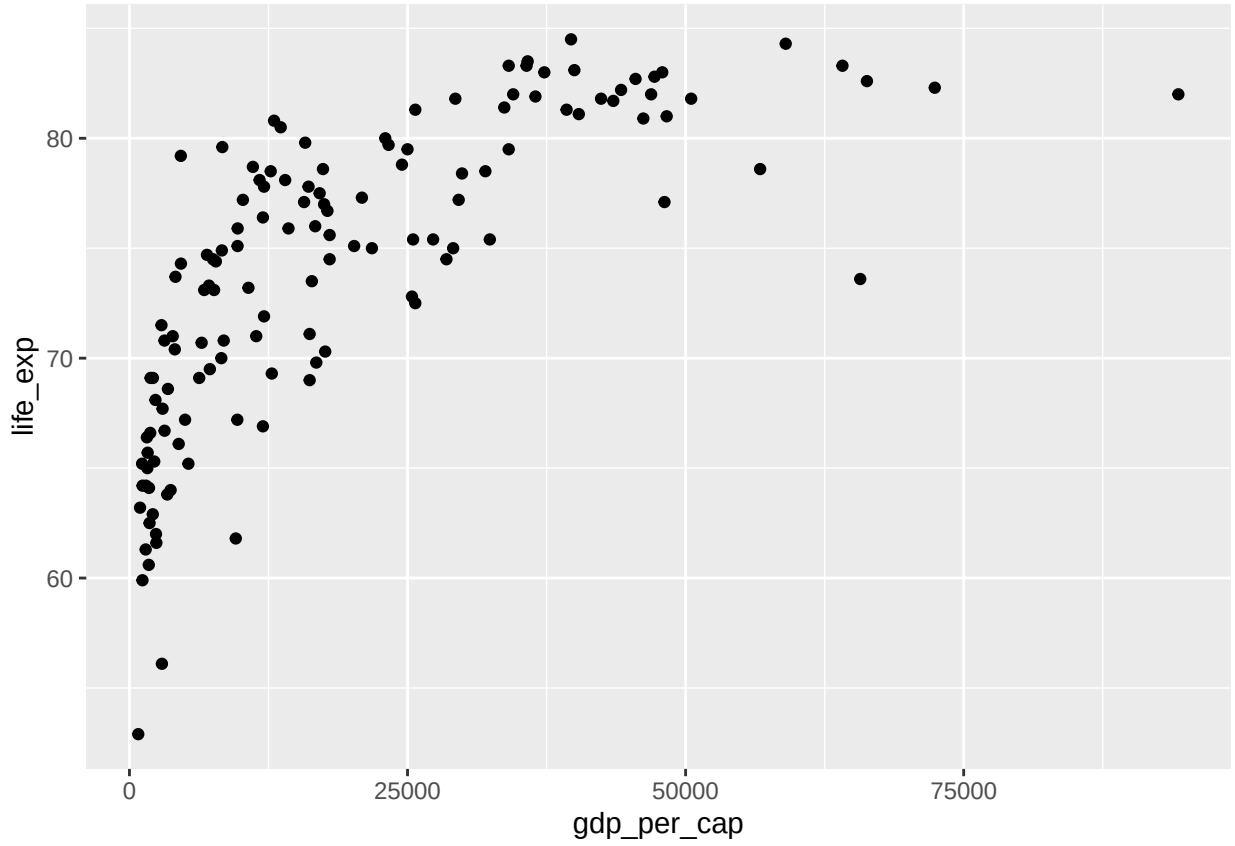

```
# Calculate the correlation between life_exp and happiness_score
cor(world_happiness$life_exp,
    world_happiness$happiness_score)
```

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

Correlation Caveats

- When data is **skewed** transformations are useful to better assess the relationship between variables.
  - Log transformation can be done using `log(x)`,
  - Square root transformation can be done using `sqrt(x)`,
  - Reciprocal transformation can be done using `(1/x)`.
- The choice of transformation will **depend** on the **data** and how **skewed** it is.
- **Correlation does not imply causation**.
- **Spurious correlation**: no causation but high correlation.
- **Confounding** is a phenomenon that lead to spurious correlations.

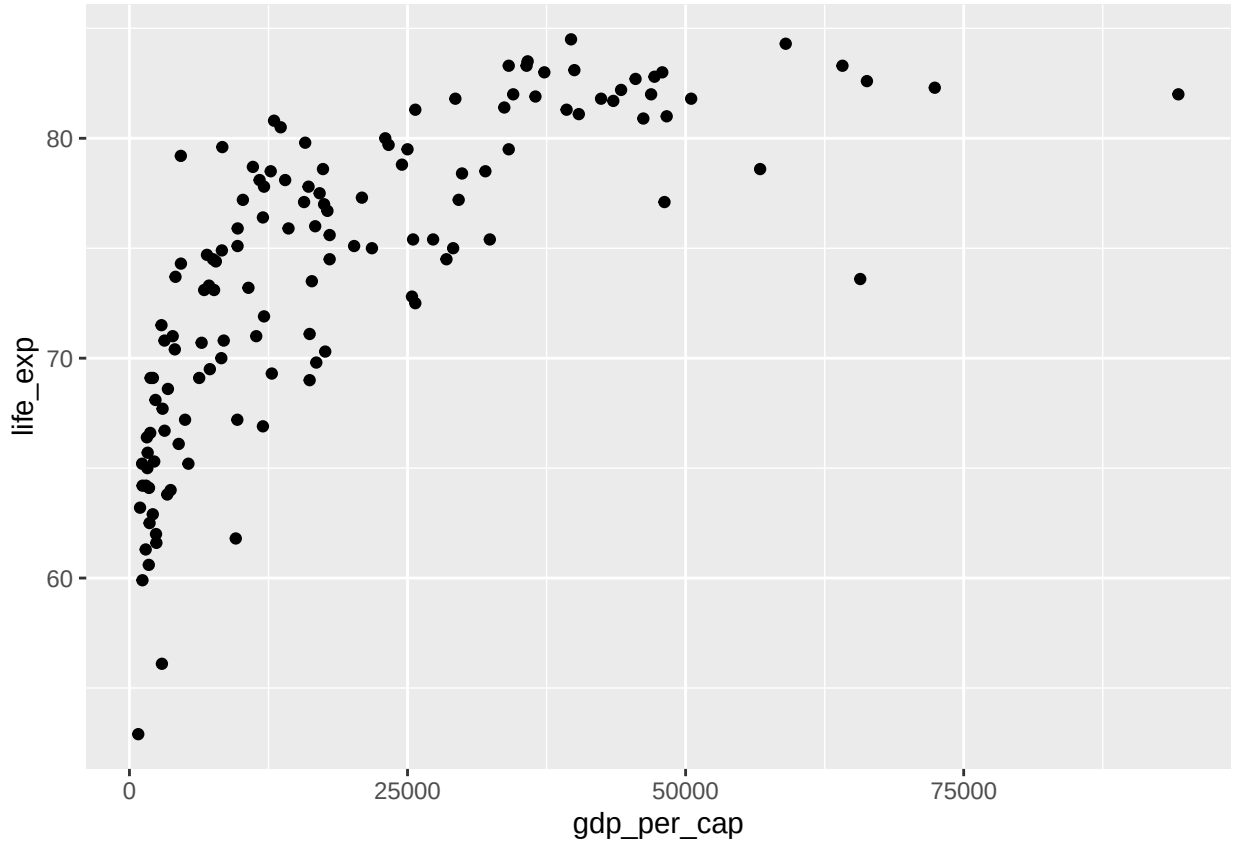
```
# Scatterplot of gdp_per_cap and life_exp
ggplot(world_happiness, aes(gdp_per_cap, life_exp)) +
  geom_point()
```



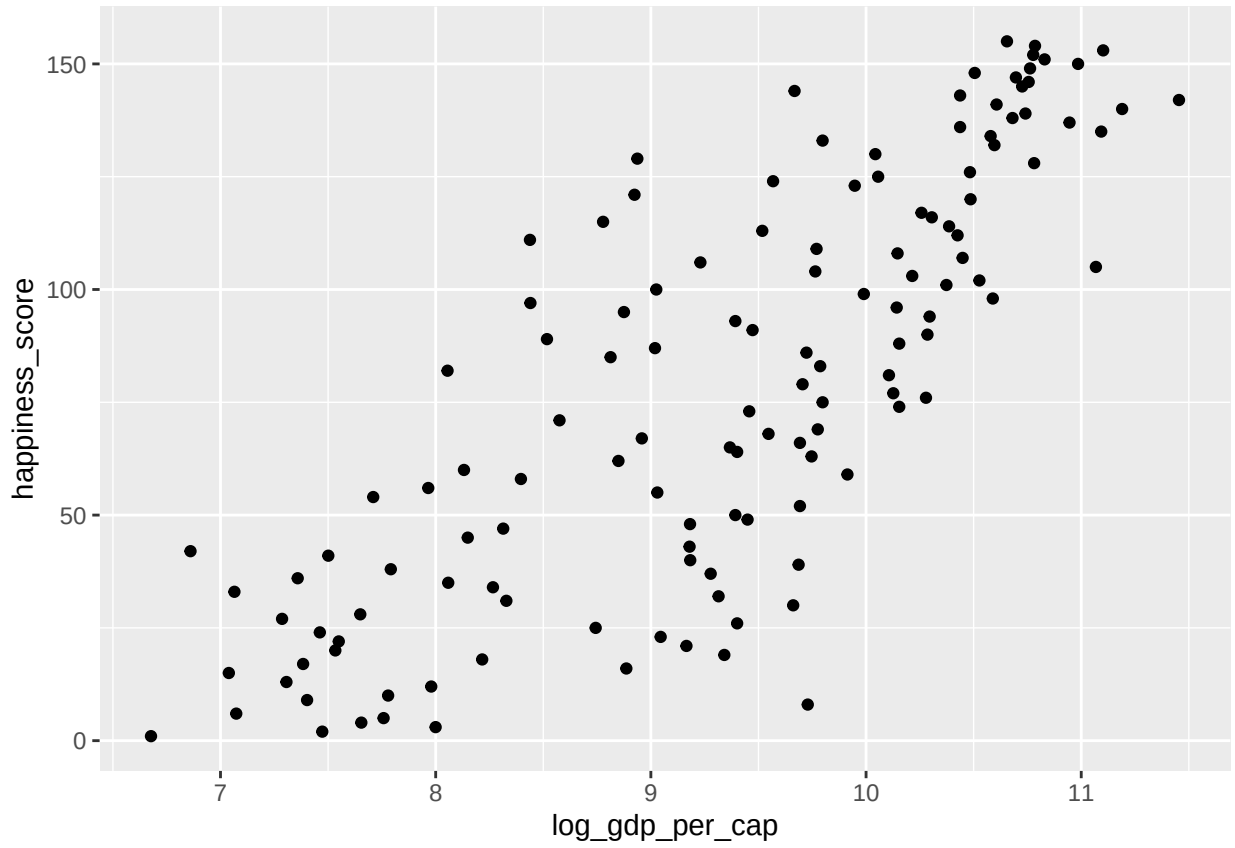

```
# Correlation between gdp_per_cap and life_exp
cor(world_happiness$gdp_per_cap, world_happiness$life_exp)
```

```
## [1] 0.7235027
```

```
# Scatterplot of gdp_per_cap and life_exp
world_happiness %>%
    ggplot(aes(x = gdp_per_cap,
               y = life_exp)) +
    geom_point()
```



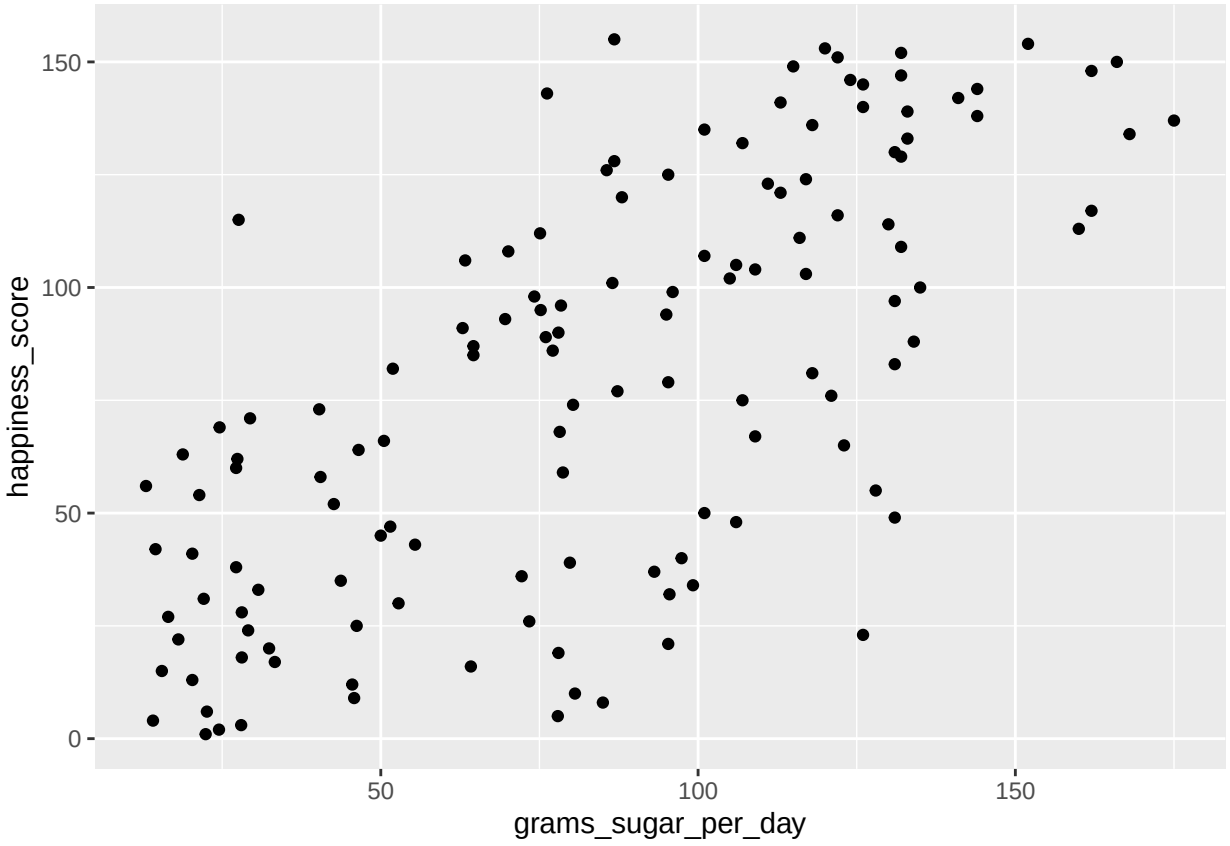

```
# Create log_gdp_per_cap column
world_happiness <- world_happiness %>%
  mutate(log_gdp_per_cap = log(gdp_per_cap))

# Scatterplot of log_gdp_per_cap vs. happiness_score
ggplot(world_happiness,
       aes(x = log_gdp_per_cap,
           y = happiness_score)) +
  geom_point()
```




```
# Calculate correlation
cor(world_happiness$log_gdp_per_cap,
    world_happiness$happiness_score)
```

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

```
# Scatterplot of grams_sugar_per_day and happiness_score
ggplot(world_happiness,
       aes(x = grams_sugar_per_day,
           y = happiness_score)) +
  geom_point()
```




```
# Correlation between grams_sugar_per_day and happiness_score
cor(world_happiness$grams_sugar_per_day,
    world_happiness$happiness_score)
```

```
## [1] 0.69391
```

Experimental Design

- Experiments aim to answer a question "what is the effect of the independent or explanatory variable on the dependent or response variable?"

Controlled Experiments

- In **controlled experiments**, participants are **randomly assigned** to either the **treatment group** or **control group**.
- **Causation** can likely be inferred if the control and test groups have similar characteristics and don't have any systematic difference between them.
- Ideal controlled experiments use the following tools to **reduce bias**:
  - **Randomized controlled trials**, were participants are randomly assigned to the treatment or control group.
  - Use **placebo**, by using something that resembles the treatment but has no effect on the user.

- Double-blind experiments, where the administer does not know whether they are administering the actual treatment or placebo. This helps prevent bias in the response as well as the analysis of the results.

Observational Studies

- Participants are **not randomly assigned** to groups. Instead, **participants assign themselves** usually based on **pre-existing characteristics**.

- Because assignment is not random, there is no guarantee that the groups will be comparable in every aspect, and therefore **causation can't be established only association**.

Longitudinal and Cross-sectional Studies

- In a **longitudinal study**, the same participants are **followed over a period of time** to examine the effect of treatment on the response.

- In a **cross-sectional study**, data is collected from **a single snapshot in time**.