Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA INFORMÁTICA

Asignatura: INFERENCIA ESTADÍSTICA
Grado en Ciencia e Ingeniería de Datos

Laboratorios de la asignatura

(Fecha del material: Noviembre 2024)

Curso académico 2024-2025

Material docente en abierto de la Universidad Rey Juan Carlos

Autores: Víctor Aceña, Isaac Martín de Diego y Carmen Lancho

Índice de los laboratorios

Además del presente documento, se ponen a disposición del lector los códigos de R que generan los laboratorios. Se pueden encontrar en el siguiente repositorio:

https://github.com/URJCDSLab/LaboratoriosRInferenciaEstadistica

Laboratorio de Probabilidad y Variables Aleatorias

Carmen Lancho - Isaac Martín - Víctor Aceña

Grado en Ciencia e Ingeniería de Datos - Inferencia Estadística - Curso 2024/2025

Índice

Objetivo

El objetivo principal de este documento es aprender a calcular probabilidades de distintas variables aleatorias. Como objetivos secundarios tenemos:

1. Identificar modelos de distribución de probabilidad a partir de unos datos
2. Saber manejar las funciones de R para calcular probabilidades
3. Saber interpretar los cálculos

1. Cálculos en variables aleatorias genéricas

La teoría de variables aleatorias ofrece un marco robusto para entender y modelizar fenómenos aleatorios en diversas disciplinas. En esta sección, abordaremos cómo realizar cálculos prácticos con variables aleatorias, tanto discretas como continuas, utilizando herramientas de R. Comenzaremos con las variables aleatorias discretas, donde la probabilidad de cada posible resultado se puede calcular directamente o acumularse. Luego, avanzaremos hacia las variables aleatorias continuas, cuya comprensión requiere integrar su función de densidad sobre un intervalo de interés.

1.1. Variables aleatorias discretas

Si tenemos la función de masa de probabilidad de una variable aleatoria discreta en forma de tabla, podemos guardar los valores x_i en un vector y los valores p_i en otro vector, y a partir de ahí realizar operaciones vectoriales para multiplicar y/o sumar, de forma que se obtengan los valores pedidos.

> Supongamos que tenemos la variable aleatoria "número de caras en el lanzamiento de 3 monedas". Los posibles valores son $\{0, 1, 2, 3\}$ y sus probabilidades $\{\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}\}$. Queremos evaluar las siguientes cuestiones:
>
> 1. **Verificación de la Suma de Probabilidades:** Es fundamental asegurar que las probabilidades sumen 1 para confirmar que están bien definidas.
> 2. **Probabilidad de un Evento:** Analizar la probabilidad de obtener menos de 2 caras nos permite entender mejor la distribución de la variable.
> 3. **Media de la Variable Aleatoria:** La media nos proporciona una medida central de la distribución.
> 4. **Varianza de la Variable Aleatoria:** La varianza nos da una idea de la dispersión de los valores alrededor de la media.

Para analizar esta variable aleatoria, definimos matemáticamente los posibles valores que puede tomar la variable $X = x_i$ y las correspondientes probabilidades $P(X = x_i) = p_i$, donde $i \in \{1, 2, 3, 4\}$. En R, almacenamos los posibles valores x_i en el vector `x` y las probabilidades asociadas p_i en el vector `p`, como se muestra a continuación:

$$X = \{x_i\} = \{0, 1, 2, 3\}$$

$$P = \{p_i\} = \left\{\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}\right\}$$

Estos vectores se utilizan luego para realizar cálculos estadísticos sobre la variable aleatoria:

```r
# Definición de los vectores de valores y probabilidades
x <- c(0, 1, 2, 3)
p <- c(1/8, 3/8, 3/8, 1/8)
```

Para confirmar que las probabilidades están correctamente normalizadas, verificamos que su suma sea igual a 1, lo cual se representa matemáticamente como:

$$\sum_{i=1}^{n} p_i = 1$$

Implementamos esta verificación en R de la siguiente manera:

```r
# Suma de probabilidades
suma_probabilidades <- sum(p)
cat(sprintf("La suma de las probabilidades es %.2f.\n", suma_probabilidades))
```

```
## La suma de las probabilidades es 1.00.
```

Calculamos la probabilidad de obtener menos de 2 caras. Matemáticamente, esta probabilidad se calcula sumando las probabilidades de los eventos donde el número de caras es menor que 2:

$$P(X < 2) = \sum_{x_i < 2} p_i = \sum_{i=1}^{2} p_i$$

Implementamos esta probabilidad en R de la siguiente manera:

```
# Probabilidad de obtener menos de 2 caras
probabilidad_menos_2 <- sum(p[x < 2])
cat(sprintf("La probabilidad de obtener menos de 2 caras es %.2f.\n", probabilidad_menos_2))
```

```
## La probabilidad de obtener menos de 2 caras es 0.50.
```

La media o esperanza matemática se calcula utilizando la expresión:

$$E(X) = \sum_{i=1}^{n} x_i p_i$$

Para determinar la media de la variable aleatoria:

```
# Media de la variable aleatoria
media <- sum(x * p)
cat(sprintf("La media de la variable aleatoria es %.2f.\n", media))
```

```
## La media de la variable aleatoria es 1.50.
```

La varianza se calcula con la expresión:

$$Var(X) = E(X^2) - [E(X)]^2 = \sum_{i=1}^{n} (x_i^2 p_i) - \left(\sum_{i=1}^{n} x_i p_i \right)^2$$

En R se puede implementar como:

```
# Varianza de la variable aleatoria
varianza <- sum(x^2 * p) - media^2
cat(sprintf("La varianza de la variable aleatoria es %.2f.\n", varianza))
```

```
## La varianza de la variable aleatoria es 0.75.
```

1.2. Variables aleatorias continuas

Los cálculos de variables aleatorias continuas se realizan a través de la función de densidad (con integrales) o a través de la función de distribución (sustituyendo valores en la función). En ambos casos, lo más cómodo es crear una función en R para esas funciones matemáticas.

Supongamos que tenemos una variable aleatoria definida por sus funciones de densidad y de distribución, definidas de la siguiente manera:

$$f(x) = \begin{cases} \frac{1}{8}x & \text{si} \quad 0 < x < 4 \\ \\ 0 & \text{resto} \end{cases} \quad ; \quad F(x) = \begin{cases} 0 & \text{si} \quad x \leq 0 \\ \frac{x^2}{16} & \text{si} \quad 0 < x < 4 \\ 1 & \text{si} \quad x \geq 4 \end{cases}$$

Sobre esta variable aleatoria se plantean las siguientes cuestiones:

1. **Verificación de la integral de la función de densidad:** Es fundamental asegurar que la función $f(x)$ es una función de densidad, para ello se verifica que la integral total es 1.
2. **Probabilidad de un Evento:** La probabilidad de que la variable aleatoria sea menor de 3.
3. **Probabilidad de un Evento:** La probabilidad de que la variable aleatoria esté entre 1.5 y 2.
4. **Media de la Variable Aleatoria:** La media nos proporciona una medida central de la distribución.
5. **Varianza de la Variable Aleatoria:** La varianza nos da una idea de la dispersión de los valores alrededor de la media.

A continuación, vamos a implementar estas funciones en R. La función `f` representa la función de densidad, mientras que `Fdist` representa la función de distribución.

```
f <- function(x) (1/8)*x
Fdist <- function(x) x^2/16
```

La sencillez de las expresiones nos sirven para esta práctica, poniendo cuidado en elegir los valores de `x` entre 0 y 4.

Si quisiéramos algo más elaborado podemos introducir condiciones, por ejemplo así:

```
Fdist2 <- function(x){
  if(x <= 0){
    0
  } else if(x < 4){
    x^2/16
  }
  else{
    1
  }
}
```

Las funciones creadas se pueden utilizar ahora para obtener valores sustituyendo la `x` por cualquier valor. Por ejemplo, para obtener valores de la función de distribución, que son probabilidades directamente, $F(1)$ sería `Fdist(1)`. Para obtener probabilidades con la función de densidad, utilizamos la función `integrate`, introduciendo como primer argumento la función, y después los límites de la integral. Admite el valor `inf`.

Para verificar que $f(x)$ es una función de densidad se evalúa que su integral total es 1:

$$\int_{-\infty}^{\infty} f(x)\,dx = \int_{0}^{4} \frac{1}{8}x\,dx = 1$$

En R, realizamos el cálculo mediante función `integrate` de R para evaluar la integral:

```r
integral_total <- integrate(f, 0, 4)

cat(sprintf("La integral de f(x) es %.2f.\n", integral_total$value))
```

```
## La integral de f(x) es 1.00.
```

> Fíjate que la salida de la integral incluye el error cometido. Esto es porque el ordenador utiliza métodos numéricos (no exactos) para obtener la integral. por eso después para usar el valor lo extraemos con `$value`. Por otra parte, podemos meter como argumento de la función `integrate` funciones creadas "al vuelo".

Para calcular la probabilidad de que la variable aleatoria sea menor que 3, es decir $P\{X \leq 3\}$, evaluamos $F(x)$ en $x = 3$:

```r
cat(sprintf("La probabilidad de que X sea menor o igual que 3 es %.2f.\n", Fdist(3)))
```

```
## La probabilidad de que X sea menor o igual que 3 es 0.56.
```

Para calcular la probabilidad de que la variable aleatoria esté entre 1.5 y 2, necesitamos integrar la función de densidad en ese intervalo. Matemáticamente, esto se expresa como:

$$P(1{,}5 \leq X \leq 2) = \int_{1{,}5}^{2} f(x)\,dx$$

En R, realizamos el cálculo mediante función `integrate` de R para evaluar la integral entre 1.5 y 2.

```r
cat(sprintf("La probabilidad de que X esté entre 1.5 y 2 es %.2f.\n", integrate(f, 1.5, 2)$value))
```

```
## La probabilidad de que X esté entre 1.5 y 2 es 0.11.
```

Podemos comprobar que este resultado es equivalente si empleamos la función de distribución:

$$P(1{,}5 \leq X \leq 2) = F(2) - F(1{,}5)$$

Donde $F(2)$ es la probabilidad de que la variable aleatoria sea menor o igual que 2, y $F(1{,}5)$ es la probabilidad de que la variable aleatoria sea menor o igual que 1.5. Restar estas dos probabilidades nos da la probabilidad de que la variable aleatoria esté entre 1.5 y 2.

```r
diferencia <- Fdist(2) - Fdist(1.5)

cat(sprintf("La probabilidad de que X esté entre 1.5 y 2 es %.2f.\n", diferencia))
```

```
## La probabilidad de que X esté entre 1.5 y 2 es 0.11.
```

En el contexto de la variable aleatoria definida por la función de densidad $f(x)$, la esperanza se calcula como:

$$E(X) = \int_{-\infty}^{\infty} x \cdot f(x)\,dx$$

Podemos calcular la esperanza de la variable aleatoria utilizando la función `integrate` de R para evaluar la integral $\int_0^4 x \cdot f(x)\,dx$:

```r
Ex <- integrate(function(x) x*f(x), 0, 4)$value

cat(sprintf("La media de la variable aleatoria es %.2f.\n", Ex))
```

```
## La media de la variable aleatoria es 2.67.
```

En el contexto de la variable aleatoria definida por la función de densidad $f(x)$, la varianza se calcula como:

$$Var(X) = E(X^2) - [E(X)]^2$$

Aplicando este concepto, primero calculamos $E(X^2)$ utilizando la función de densidad y luego calculamos la varianza utilizando la expresión anterior:

```r
Ex2 <- integrate(function(x) x^2*f(x), 0, 4)$value
Vx <- Ex2-Ex^2

cat(sprintf("La varianza de la variable aleatoria es %.2f.\n", Vx))
```

```
## La varianza de la variable aleatoria es 0.89.
```

2. Cálculos en modelos de distribución de probabilidad

En R, para cada modelo de distribución de probabilidad tenemos una función que empieza por `d` y devuelve la "densidad" (masa de probabilidad en el caso de discretas) y otra que empieza por `p` y devuelve la "probabilidad (acumulada)", es decir, la función de distribución (o su complementario, añadiendo el argumento `lower.tail = FALSE`). Después de la `d` o la `p` vendrá el nombre (o abreviatura) del modelo de probabilidad, por ejemplo para la distribución normal `norm`.

Para cada modelo de distribución de probabilidad tenemos otras dos funciones, una que empieza por `q`, que calcula el cuantil dada una probabilidad acumulada (es decir, es la función inversa de la función de distribución) y otra que empieza por `r`, con la que podemos obtener valores aleatorios (*random*) o simulaciones de una variable aleatoria.

2.1. Distribuciones discretas

Las distribuciones discretas juegan un papel crucial cuando tratamos con variables aleatorias que toman valores específicos o cuentas. En contextos donde los resultados son contables y no continuos, aplicamos modelos de distribución discreta para describir la probabilidad asociada a cada posible valor de la variable. En R, utilizamos funciones que comienzan con `d` para calcular la masa de probabilidad, que nos da la probabilidad de cada valor específico de la variable. Similarmente, las funciones que comienzan con `p` nos ayudan a obtener la función de distribución acumulada para evaluar la probabilidad de que la variable tome un valor menor o igual a un cierto límite.

En esta sección, abordaremos modelos específicos como la distribución binomial, la distribución geométrica, la distribución binomial negativa, la distribución hipergeométrica y la distribución de Poisson. A través de ejemplos prácticos, exploraremos cómo aplicar las funciones `dxxx`, `pxxx`, `qxxx`, y `rxxx` para comprender y analizar estas distribuciones en diversos contextos.

2.1.1. Funciones disponibles

En R, cada distribución discreta cuenta con cuatro funciones asociadas, que permiten realizar distintos cálculos estadísticos:

- `dxxx(x, ...)` calcula la función de masa de probabilidad, $f(x)$, para un valor dado de x.
- `pxxx(q, ...)` obtiene la función de distribución acumulada, $F(x)$, hasta un punto q.
- `qxxx(p, ...)` determina el cuantil para el cual la probabilidad $P(X \leq q)$ es igual a p.
- `rxxx(n, ...)` genera n números aleatorios siguiendo la distribución especificada.

Reemplace `xxx` por el identificador correspondiente de la distribución que desea utilizar. A continuación, se presentan los identificadores para las seis distribuciones discretas básicas en R:

- `binom`: Distribución Binomial.
- `geo`: Distribución Geométrica.
- `nbinom`: Distribución Binomial Negativa.
- `hyper`: Distribución Hipergeométrica.
- `pois`: Distribución de Poisson.
- `multinom`: Distribución Multinomial (Nota: `multinom` no sigue la misma nomenclatura de `dxxx`, `pxxx`, `qxxx`, `rxxx` ya que se utiliza principalmente para modelos multinomiales).

Las funciones en R están meticulosamente diseñadas para facilitar el análisis y la modelización de variables aleatorias. La consistencia en la nomenclatura de estas funciones ayuda a identificar rápidamente la herramienta adecuada para cada tipo de cálculo relacionado con distintas distribuciones estadísticas.

2.1.2. Distribución Binomial

La distribución binomial se utiliza para modelar el número de éxitos en una secuencia de n ensayos independientes entre sí, con una probabilidad fija p de ocurrencia del éxito en cada ensayo. La función `dbinom(x, size, prob)` en R proporciona la probabilidad de obtener exactamente x éxitos en n ensayos.

Para utilizar `dbinom`, necesitamos especificar:

- x: el número de éxitos que estamos investigando.
- `size` (n): el número total de ensayos.
- `prob` (p): la probabilidad de éxito en cada ensayo.

La probabilidad de obtener exactamente x éxitos se calcula como:

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

> Supongamos que la probabilidad de que un estudiante acabe un grado en Ciencias es de 0,4. Tomamos al azar un grupo de 5 estudiantes. ¿Cuál es la probabilidad de que ninguno obtenga el grado? ¿Y la probabilidad de que al menos dos lo obtengan?

Para calcular la probabilidad de un número específico de éxitos en una distribución binomial, utilizamos la función `dbinom` en R. Esta función devuelve la probabilidad de obtener un número determinado de éxitos (`x`) en un número fijo de ensayos independientes (`size`), cada uno con la misma probabilidad de éxito (`prob`).

En el contexto de nuestro ejemplo, si queremos calcular la probabilidad de que ninguno de los 5 estudiantes termine su grado en Ciencias (considerando que la probabilidad de que un estudiante termine es de 0.4),

podemos configurar `dbinom` con los siguientes parámetros: `x = 0` (ningún estudiante termina), `size = 5` (cinco ensayos o estudiantes) y `prob = 0.4` (la probabilidad de éxito, es decir, que un estudiante termine).

La fórmula matemática que `dbinom` utiliza para calcular esta probabilidad es:

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

Donde:

- X es la variable aleatoria que representa el número de éxitos (estudiantes que terminan el grado).
- $n = 5$ es el número total de ensayos (estudiantes).
- $x = 0$ es el número de éxitos para los que queremos calcular la probabilidad.
- $p = 0{,}4$ es la probabilidad de éxito en cada ensayo.

Al aplicar estos valores en la fórmula, calculamos la probabilidad específica para nuestro escenario:

```
prob_no_grado <- dbinom(x = 0, size = 5, prob = 0.4)
cat(sprintf("La probabilidad de que ninguno obtenga el grado es%.4f.\n", prob_no_grado))
```

```
## La probabilidad de que ninguno obtenga el grado es 0.0778.
```

Para calcular la probabilidad de que el número de éxitos sea mayor a un valor específico en una distribución binomial, podemos utilizar la función `pbinom` de R. Esta función devuelve la probabilidad acumulada de obtener un número de éxitos menor o igual a `q` en una serie de ensayos.

En nuestro ejemplo, si queremos saber la probabilidad de que más de un estudiante (es decir, al menos dos) termine su grado en Ciencias, podemos calcular la complementaria de la probabilidad de que uno o ninguno lo logre. Esto se puede hacer utilizando `1 - pbinom(q = 1, size = 5, prob = 0.4)`, donde `q = 1` representa el umbral máximo de estudiantes (uno) que no queremos superar para nuestro cálculo complementario.

La función `pbinom` utiliza la siguiente fórmula para calcular la probabilidad acumulada:

$$P(X \leq q) = \sum_{k=0}^{q} \binom{n}{k} p^k (1-p)^{n-k}$$

Al restar este resultado de 1, obtenemos la probabilidad de que más de un estudiante (al menos dos) termine el grado:

$$P(X > q) = 1 - P(X \leq q)$$

Por lo tanto, el siguiente bloque de código calcula la probabilidad de que al menos dos estudiantes terminen el grado:

```
prob_dos_grado <- 1 - pbinom(q = 1, size = 5, prob = 0.4)
cat(sprintf("La probabilidad de que al menos dos estudiantes terminen el grado es%.4f.\n", prob_dos_gra
```

```
## La probabilidad de que al menos dos estudiantes terminen el grado es 0.6630.
```

Otra forma de calcular la probabilidad de que más de un estudiante termine el grado es utilizar directamente la función `pbinom` con el argumento `lower.tail = FALSE`. Esto nos permite calcular la probabilidad de que el número de éxitos sea mayor que `q` directamente, sin necesidad de calcular la complementaria.

El argumento `lower.tail = FALSE` indica que estamos interesados en la probabilidad de que el número de éxitos en nuestra distribución binomial sea mayor que el valor `q` proporcionado. En este caso, queremos la probabilidad de que más de un estudiante, es decir, dos o más, termine el grado.

Por lo tanto, el código:

```
prob_dos_grado_v2 <- pbinom(q = 1, size = 5, prob = 0.4, lower.tail = FALSE)

cat(sprintf("La probabilidad de que al menos dos estudiantes terminen el grado es%.4f.\n", prob_dos_gra
```

```
## La probabilidad de que al menos dos estudiantes terminen el grado es 0.6630.
```

realiza directamente el cálculo de $P(X > 1)$ para nuestra distribución binomial, donde X es el número de estudiantes que terminan el grado, `size = 5` es el número total de estudiantes considerados, y `prob = 0.4` es la probabilidad de que un estudiante dado termine el grado.

Finalmente, otra manera de abordar el cálculo de la probabilidad de que al menos dos estudiantes terminen su grado es sumar directamente las probabilidades de obtener 2, 3, 4 o 5 estudiantes que terminan el grado. Esto se puede hacer sumando las probabilidades individuales para cada uno de estos casos.

Utilizamos `dbinom` para calcular la probabilidad de cada número específico de éxitos (en este caso, de 2 a 5 éxitos) y luego sumamos estos valores. Esto nos proporciona la probabilidad total de que al menos dos estudiantes, entre los cinco considerados, terminen el grado en Ciencias. El parámetro `size = 5` define el número total de ensayos (estudiantes), y `prob = 0.4` es la probabilidad de éxito individual (un estudiante terminando el grado).

El código siguiente ejecuta este cálculo:

```
prob_dos_grado_v3 <- sum(dbinom(x = 2:5, size = 5, prob = 0.4))

cat(sprintf("La probabilidad de que al menos dos estudiantes terminen el grado es%.4f.\n", prob_dos_gra
```

```
## La probabilidad de que al menos dos estudiantes terminen el grado es 0.6630.
```

Esta expresión suma las probabilidades calculadas por `dbinom` para 2, 3, 4 y 5 éxitos (estudiantes que terminan el grado) en 5 ensayos, ofreciendo otra vía para determinar la probabilidad de que al menos dos estudiantes logren terminar su grado.

2.1.3. Distribución de Bernoulli

La distribución de Bernoulli es la distribución más simple, modelando un experimento que tiene solo dos posibles resultados: éxito o fracaso (normalmente codificados como 1 y 0, respectivamente). La función `dbinom(x, size, prob)` en R puede utilizarse para una variable aleatoria de Bernoulli, especificando `size = 1` (un solo ensayo).

En la distribución de Bernoulli:

- x: el valor de éxito o fracaso (0 o 1).
- `size = 1`: ya que se trata de un solo ensayo.
- `prob` (p): la probabilidad de éxito en ese ensayo.

La probabilidad de obtener exactamente x en un experimento de Bernoulli se calcula como:

$$P(X = x) = p^x(1-p)^{1-x}$$

> Supongamos que lanzamos una moneda con una probabilidad de 0.6 de obtener cara (éxito). ¿Cuál es la probabilidad de que salga cara (éxito)? ¿Cuál es la probabilidad de que salga cruz (fracaso)?

Podemos utilizar la función `dbinom` en R para calcular las probabilidades de los dos resultados posibles (0 o 1) de una variable aleatoria de Bernoulli.

```
p <- 0.6  # Probabilidad de éxito

# Probabilidades para éxito (1) y fracaso (0)
prob_1 <- dbinom(x = 1, size = 1, prob = p)
prob_0 <- dbinom(x = 0, size = 1, prob = p)

cat(sprintf("La probabilidad de éxito (cara) es %.4f.\n", prob_1))
```

```
## La probabilidad de éxito (cara) es 0.6000.
```

```
cat(sprintf("La probabilidad de fracaso (cruz) es %.4f.\n", prob_0))
```

```
## La probabilidad de fracaso (cruz) es 0.4000.
```

2.1.4. Distribución de Poisson

La distribución de Poisson se utiliza para modelar el número de eventos que ocurren en un intervalo de tiempo fijo cuando estos eventos ocurren con una tasa media constante y de manera independiente entre sí. Esta distribución se define completamente por su parámetro λ (lambda), que representa la tasa media de ocurrencia de los eventos por intervalo.

Para un valor dado k, la probabilidad de observar exactamente k eventos se calcula como:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

Donde:

- X es la variable aleatoria que representa el número de eventos,
- λ es la tasa media de eventos por intervalo,
- k es el número de ocurrencias del evento.

Las funciones principales en R para trabajar con esta distribución son `dpois` para la función de probabilidad y `ppois` para la función de distribución acumulada.

> En una parada de autobús llegan de media cuatro autobuses cada hora. ¿Cuál es la probabilidad de que en una hora pasen más de 8?

En el contexto de una parada de autobús donde llegan de media cuatro autobuses cada hora, podemos usar la distribución de Poisson para modelar la probabilidad de diferentes números de llegadas. Para calcular la probabilidad de que lleguen más de 8 autobuses en una hora, podemos usar la función `ppois` con `lower.tail = FALSE`, que nos da la probabilidad de que el número de llegadas sea mayor que un valor dado (en este caso, 8 autobuses).

La función `ppois(8, lambda = 4, lower.tail = FALSE)` calculará la probabilidad de que el número de autobuses que llegan en una hora sea mayor que 8, dado que la tasa media (λ) es de 4 autobuses por hora.

```
prob_8buses <- ppois(8, lambda = 4, lower.tail = FALSE)
cat(sprintf("La probabilidad de que el número de autobuses que llegan en una hora sea mayor que 8 es %.4
```

```
## La probabilidad de que el número de autobuses que llegan en una hora sea mayor que 8 es 0.0214.
```

Otro escenario común para aplicar la distribución de Poisson es el modelado del número de errores tipográficos que ocurren en una página de texto.

> Supongamos que, en promedio, se cometen 2 errores tipográficos por página en un libro. ¿Cuál es la probabilidad de que en una página no haya errores?

En este ejemplo, queremos calcular la probabilidad de que en una página específica no haya errores tipográficos, sabiendo que la tasa media (λ) es de 2 errores por página.

La función `dpois` nos permitirá calcular la probabilidad de observar exactamente $k = 0$ errores en una página, dado que la tasa media de errores es $\lambda = 2$.

```
prob_no_errors <- dpois(0, lambda = 2)
cat(sprintf("La probabilidad de que no haya errores en una página es %.4f.\n", prob_no_errors))
```

```
## La probabilidad de que no haya errores en una página es 0.1353.
```

2.1.5. Distribución Binomial Negativa

La distribución binomial negativa modela el número de fracasos que ocurren antes de que se alcancen un número fijo de éxitos en una secuencia de ensayos independientes con probabilidad constante de éxito. La función `dnbinom(x, size, prob)` en R devuelve la probabilidad de observar un número específico de fracasos antes de alcanzar un número determinado de éxitos.

Para utilizar `dnbinom`, necesitamos especificar:

- x: el número de fracasos antes de alcanzar los éxitos deseados.
- `size` (k): el número de éxitos que estamos esperando alcanzar.
- `prob` (p): la probabilidad de éxito en cada ensayo.

La probabilidad se calcula como:

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

> Supongamos que lanzamos una moneda con probabilidad de 0.4 de éxito (cara). Queremos saber la probabilidad de que ocurran 3 fracasos antes de que consigamos 2 éxitos.

Donde:

- X es la variable que representa el número de fracasos.
- $k = 2$ es el número de éxitos que se desean.
- $p = 0{,}4$ es la probabilidad de éxito en cada ensayo.

```
p <- 0.4  # Probabilidad de éxito
size <- 2  # Número de éxitos deseados

# Probabilidad de tener exactamente 3 fracasos antes de 2 éxitos
prob_3_failures <- dnbinom(x = 3, size = size, prob = p)
cat(sprintf("La probabilidad de que ocurran exactamente 3 fracasos antes de 2 éxitos es%.4f.\n", prob_3
```

```
## La probabilidad de que ocurran exactamente 3 fracasos antes de 2 éxitos es 0.1382.
```

2.1.6. Distribución Geométrica

La distribución geométrica modela el número de fracasos que ocurren antes del primer éxito en una secuencia de ensayos independientes con probabilidad constante de éxito. La función `dgeom(x, prob)` en R calcula la probabilidad de obtener exactamente x fracasos antes de obtener el primer éxito.

Para utilizar `dgeom`, necesitamos especificar:

- x: el número de fracasos antes del primer éxito.
- `prob` (p): la probabilidad de éxito en cada ensayo.

La probabilidad de obtener exactamente x fracasos antes del primer éxito se calcula como:

$$P(X = x) = (1 - p)^x p$$

Donde p es la probabilidad de éxito en cada ensayo y x es el número de fracasos.

> Supongamos que lanzamos una moneda con una probabilidad de 0.7 de obtener cara (éxito). Queremos calcular la probabilidad de que haya exactamente 2 fracasos antes de obtener la primera cara.

```
p <- 0.7  # Probabilidad de éxito

# Probabilidad de obtener 2 fracasos antes del primer éxito
prob_2_failures <- dgeom(x = 2, prob = p)
cat(sprintf("La probabilidad de que ocurran exactamente 2 fracasos antes del primer éxito es%.4f.\n", p
```

```
## La probabilidad de que ocurran exactamente 2 fracasos antes del primer éxito es 0.0630.
```

Otro caso en el que se puede aplicar la distribución geométrica es en la detección de defectos en un proceso de producción. Supongamos que en una fábrica, cada artículo producido tiene una probabilidad de 0.1 de ser defectuoso. Queremos calcular la probabilidad de que haya exactamente 3 productos defectuosos antes de encontrar el primer producto no defectuoso (éxito).

> En una fábrica, cada producto tiene una probabilidad de 0.1 de ser defectuoso. Calcula la probabilidad de que haya exactamente 3 productos defectuosos antes de encontrar el primer producto no defectuoso.

En este ejemplo, podemos usar la función `dgeom` para calcular la probabilidad de tener exactamente 3 fracasos (productos defectuosos) antes de encontrar el primer éxito (producto no defectuoso).

```
p <- 0.9  # Probabilidad de éxito (producto no defectuoso)

# Probabilidad de obtener 3 productos defectuosos antes del primer no defectuoso
prob_3_failures <- dgeom(x = 3, prob = p)
cat(sprintf("La probabilidad de que haya exactamente 3 productos defectuosos antes del primer no defectu
```

```
## La probabilidad de que haya exactamente 3 productos defectuosos antes del primer no defectuoso es 0.
```

2.1.7. Distribución Hipergeométrica

Para la distribución hipergeométrica, utilizamos `dhyper` en R, que requiere parámetros específicos para describir el escenario: el total de éxitos en la población (`m`), el total de no éxitos en la población (`n`), y el número de extracciones (`k`). La función devuelve la probabilidad de obtener un número dado de éxitos (`x`) en las extracciones realizadas.

> En un comité de dirección de una empresa medioambiental con 50 miembros, 30 están de acuerdo en crear una línea de negocio de vehículo eléctrico, y el resto no. En el descanso, cinco directivos (al azar) se salen a la máquina de café. ¿Cuál es la probabilidad de que de esos cinco solo uno esté de acuerdo en crear la línea de negocio?

Los parámetros para `dhyper` en este caso son: `x = 1` (un directivo a favor entre los seleccionados), `m = 30` (total de directivos a favor en el comité), `n = 20` (total de directivos en contra), y `k = 5` (número de directivos seleccionados para ir al café).

La probabilidad se calcula como:

$$P(X = x) = \frac{\binom{m}{x}\binom{N-m}{k-x}}{\binom{N}{k}}$$

Donde:

- $N = m + n$ es el total de la población o el tamaño del comité.
- X es la variable aleatoria que representa el número de directivos a favor seleccionados.

Aplicamos la función `dhyper` para obtener la probabilidad:

```
prob_no_acuerdo <- dhyper(x = 1, m = 30, n = 20, k = 5)

cat(sprintf("La probabilidad de que de esos cinco solo uno esté de acuerdo en crear la línea de negocio
```

```
## La probabilidad de que de esos cinco solo uno esté de acuerdo en crear la línea de negocio 0.0686.
```

2.2. Distribuciones continuas

Las distribuciones continuas son esenciales cuando abordamos variables aleatorias que pueden tomar cualquier valor dentro de un intervalo. Estas distribuciones nos permiten entender y modelar fenómenos donde

la precisión y la continuidad son claves, como tiempos, distancias o temperaturas. En este caso, las funciones que empiezan por `d` nos proporcionan la densidad de probabilidad, que, a diferencia de la masa de probabilidad para las discretas, nos ofrece la densidad en un punto específico o intervalo. Las funciones que inician con `p` siguen siendo cruciales para determinar la probabilidad acumulada, permitiéndonos calcular la probabilidad de que la variable aleatoria caiga por debajo de un cierto valor.

En esta parte, nos centraremos en distribuciones continuas claves como la distribución normal, la distribución exponencial, y otras distribuciones relevantes. Exploraremos el uso de `dxxx`, `pxxx`, `qxxx`, y `rxxx` para efectuar análisis detallados y aplicaciones prácticas de estas distribuciones, proporcionando así una base sólida para el modelado y la interpretación de datos continuos.

2.2.1. Funciones disponibles

En R, cada distribución continua tiene asociadas cuatro funciones principales que facilitan la realización de cálculos estadísticos importantes:

- `dxxx(x, ...)` calcula la función de densidad de probabilidad, $f(x)$, para un valor dado de x.
- `pxxx(q, ...)` obtiene la función de distribución acumulada, $F(x)$, hasta un punto q.
- `qxxx(p, ...)` determina el cuantil para el cual la probabilidad $P(X \leq q)$ es igual a p.
- `rxxx(n, ...)` genera n números aleatorios que siguen la distribución especificada.

Reemplace `xxx` por el identificador correspondiente de la distribución que desea utilizar. A continuación, se presentan los identificadores para algunas de las distribuciones continuas comunes en R:

- `beta`: Distribución Beta.
- `cauchy`: Distribución Cauchy.
- `chisq`: Distribución Chi-cuadrado.
- `exp`: Distribución Exponencial.
- `f`: Distribución F.
- `gamma`: Distribución Gamma.
- `lnorm`: Distribución Log-normal.
- `norm`: Distribución Normal.
- `t`: Distribución t-Student.
- `unif`: Distribución Uniforme.
- `weibull`: Distribución Weibull.

Las funciones en R para distribuciones continuas permiten calcular la densidad de probabilidad (`dxxx`), la probabilidad acumulada (`pxxx`), cuantiles (`qxxx`), y generar muestras aleatorias (`rxxx`). A diferencia de las distribuciones discretas, donde `dxxx` devuelve una probabilidad puntual, en las continuas proporciona la densidad en un punto, reflejando la concentración de probabilidad, no una probabilidad exacta. Este diseño consistente facilita la identificación y aplicación correcta de cada función para análisis y modelización estadística eficaz.

Para la distribución uniforme utilizamos las funciones `dunif` y `punif`, con los argumentos `min` y `max` para los parámetros a y b respectivamente. `dunif(x, min, max)` ofrece el valor de la densidad de probabilidad en el punto `x`, mientras que `punif(q, min, max)` calcula la probabilidad de que una variable aleatoria uniforme sea menor o igual a `q`.

2.2.2. Distribución Uniforme

La distribución uniforme continua se utiliza para modelar situaciones en las que todos los valores dentro de un intervalo están igualmente distribuidos, es decir, tienen la misma probabilidad de ocurrencia. La función

`dunif(x, min, max)` en R calcula la densidad de probabilidad para un valor específico dentro de un intervalo definido por `min` y `max`.

Para utilizar `dunif`, necesitamos especificar:

- x: el valor para el cual estamos calculando la densidad de probabilidad.
- `min`: el límite inferior del intervalo.
- `max`: el límite superior del intervalo.

La función de densidad de la distribución uniforme continua se define como:

$$f(x) = \frac{1}{b-a} \quad \text{para } a \leq x \leq b$$

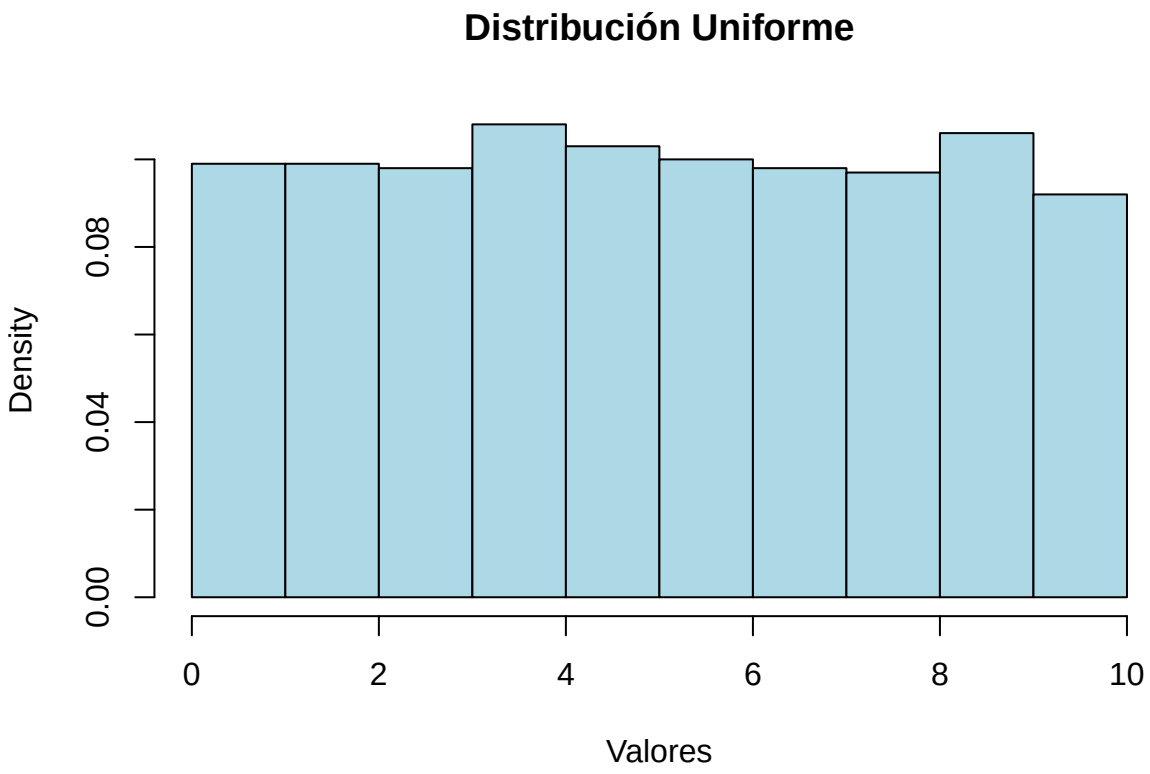
Donde a es el límite inferior y b es el límite superior.

> Supongamos que seleccionamos al azar un número entre 0 y 10. Queremos calcular la probabilidad de que ese número caiga exactamente en 5, y luego queremos generar y visualizar 1000 números aleatorios que sigan una distribución uniforme entre 0 y 10.

```r
# Parámetros de la distribución uniforme
min_val <- 0
max_val <- 10

# Probabilidad de que el número sea exactamente 5
# Nota: en la distribución continua, la probabilidad de un solo punto es 0, pero mostramos cómo usar la
prob_x_equals_5 <- dunif(5, min = min_val, max = max_val)
cat(sprintf("La densidad en x = 5 es%.4f.\n", prob_x_equals_5))
```

```
## La densidad en x = 5 es 0.1000.
```

```r
# Generar 1000 números aleatorios con distribución uniforme entre 0 y 10
set.seed(123)
random_uniform <- runif(1000, min = min_val, max = max_val)

# Visualizar histograma de los números aleatorios generados
hist(random_uniform, probability = TRUE, main = "Distribución Uniforme",
     xlab = "Valores", xlim = c(min_val, max_val), col = "lightblue", border = "black")
```



2.2.3. Distribución exponencial

La distribución exponencial es utilizada comúnmente para modelar el tiempo que transcurre entre eventos sucesivos en un proceso con tasa constante. En R, las funciones `dexp` para la densidad y `pexp` para la distribución acumulativa manejan estas consultas, utilizando el parámetro de tasa `rate`, que corresponde a $1/\beta$ en la formulación matemática de la distribución.

> El tiempo en horas que se tarda en llegar desde la empresa de mantenimiento de aerogeneradores hasta una determinada instalación sigue una distribución exponencial de parámetro $\beta = 2$. ¿Cuál es la probabilidad de que un operario tarde más de tres horas en llegar a la instalación?

La función `pexp(3, rate = 2, lower.tail = FALSE)` calculará esta probabilidad, donde `3` es el tiempo en horas que queremos evaluar, `rate = 2` es el parámetro de la distribución exponencial, y `lower.tail = FALSE` indica que buscamos la probabilidad de que el tiempo sea mayor que 3 horas.

La fórmula correspondiente para la función de distribución acumulativa de la exponencial en este contexto es:

$$P(T > t) = 1 - F(t) = e^{-\beta t}$$

Aplicando nuestros valores:

```
prob_mas_3h <- pexp(3, rate = 2, lower.tail = FALSE)

cat(sprintf("La probabilidad de que un operario tarde más de tres horas en llegar a la instalación%.4f.
```

```
## La probabilidad de que un operario tarde más de tres horas en llegar a la instalación 0.0025.
```

Este comando proporciona la probabilidad de que el tiempo hasta el próximo evento (en este caso, la llegada del operario) sea mayor que 3 horas.

Supongamos que el tiempo de respuesta de un servidor web sigue una distribución exponencial con una tasa de 0.5 respuestas por segundo, es decir, el tiempo medio de respuesta es de 2 segundos ($\beta = 2$). Queremos calcular la probabilidad de que el servidor responda en menos de 1 segundo.

> El tiempo de respuesta de un servidor web sigue una distribución exponencial con parámetro $\beta = 2$. ¿Cuál es la probabilidad de que el servidor responda en menos de 1 segundo?

Podemos usar la función `pexp` con el parámetro `lower.tail = TRUE`, que calcula la probabilidad acumulada de que el tiempo de respuesta sea **menor** que 1 segundo.

```r
rate <- 1/2  # Tasa inversa del tiempo medio de respuesta
time <- 1  # Tiempo que queremos evaluar

# Probabilidad de que el servidor responda en menos de 1 segundo
prob_menos_1s <- pexp(time, rate = rate, lower.tail = TRUE)

cat(sprintf("La probabilidad de que el servidor responda en menos de 1 segundo es%.4f.\n", prob_menos_1
```

```
## La probabilidad de que el servidor responda en menos de 1 segundo es 0.3935.
```

La distribución exponencial también se utiliza para modelar el tiempo hasta la falla de dispositivos, como bombillas. Supongamos que el tiempo de vida de una bombilla sigue una distribución exponencial con un promedio de vida útil de 1000 horas (lo que implica que el parámetro $\beta = 1000$). Queremos calcular la probabilidad de que una bombilla dure más de 1500 horas.

> El tiempo de vida útil de una bombilla sigue una distribución exponencial con parámetro $\beta = 1000$. ¿Cuál es la probabilidad de que una bombilla dure más de 1500 horas?

En este caso, podemos usar la función `pexp` para calcular la probabilidad de que la vida útil de una bombilla sea mayor a 1500 horas. Dado que el parámetro de tasa `rate` en R es la inversa de β, el valor de `rate` será 1/1000.

```r
rate <- 1/1000  # Tasa inversa del tiempo medio de vida
time <- 1500  # Tiempo que queremos evaluar

# Probabilidad de que una bombilla dure más de 1500 horas
prob_mas_1500h <- pexp(time, rate = rate, lower.tail = FALSE)
```

2.2.4. Distribución normal

La distribución normal es una de las distribuciones más importantes en estadística, utilizada para modelar fenómenos naturales, sociales y de investigación científica cuando los datos se distribuyen en forma de campana alrededor de un promedio. En R, las funciones `dnorm` y `pnorm` nos permiten trabajar con la densidad y la distribución acumulativa de la normal, respectivamente, sin necesidad de tipificar manualmente las variables, ya que podemos especificar directamente la media (`mean`) y la desviación estándar (`sd`). Su función de densidad de probabilidad es:

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

Para calcular probabilidades acumuladas y encontrar cuantiles en una distribución normal, podemos usar las funciones `pnorm` y `qnorm` en R, que incorporan estos cálculos matemáticos internamente:

- `pnorm(x, mean, sd)` calcula la probabilidad $P(X \leq x)$ para una variable normal X con media `mean` y desviación estándar `sd`.
- `qnorm(p, mean, sd)` encuentra el valor x tal que $P(X \leq x) = p$.

> El peso de los paquetes que contienen los pedidos que recibe un laboratorio se distribuye según una distribución normal de media 1,8 y desviación típica 0,5 kg. ¿Cuál es la probabilidad de que un paquete esté entre 1 y 2 kilos?¿Por debajo de qué peso estarán probablemente al menos el 95 % de los paquetes? Por último, realiza una simulación de 100 paquetes y haz un histograma.

Para calcular la probabilidad de que el peso de un paquete se encuentre en un intervalo específico, utilizamos la función `pnorm` para la distribución normal. En este caso, queremos saber la probabilidad de que un paquete pese entre 1 y 2 kilos, dados una media de 1.8 kg y una desviación estándar de 0.5 kg. La función `pnorm` nos da la probabilidad acumulada hasta un punto dado, por lo que calcularemos la probabilidad de que un paquete pese menos o igual a 2 kilos y restaremos la probabilidad de que pese menos o igual a 1 kilo:

```
prob_paquete_12 <- pnorm(2, 1.8, 0.5) - pnorm(1, 1.8, 0.5)

cat(sprintf("La probabilidad de que un paquete esté entre 1 y 2 kilos%.4f.\n", prob_paquete_12))
```

```
## La probabilidad de que un paquete esté entre 1 y 2 kilos 0.6006.
```

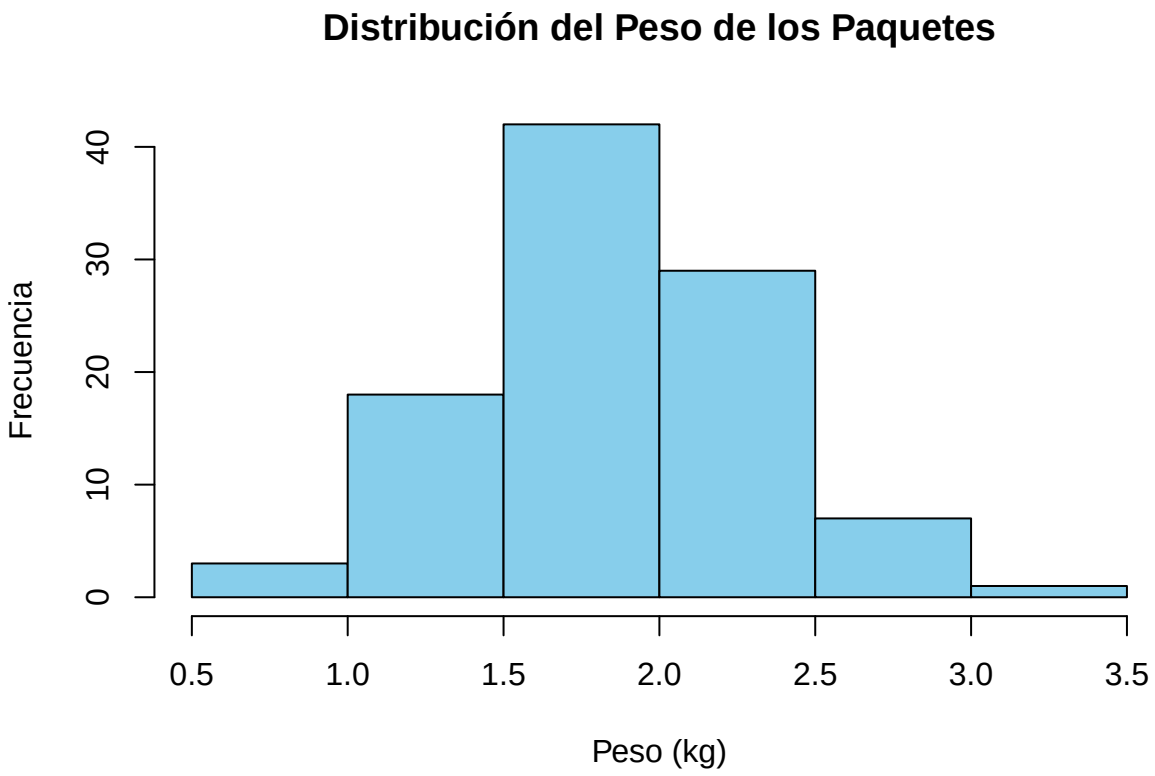
En cuanto al segundo cálculo, si deseamos encontrar un umbral de peso por debajo del cual se encuentre al menos el 95 % de los paquetes, utilizaremos la función `qnorm`. Esta función devuelve el cuantil o valor crítico asociado a una probabilidad acumulada dada. Para obtener el peso que no supera el 95 % de los paquetes, establecemos `p = 0.95`, junto con la media (`mean = 1.8`) y desviación estándar (`sd = 0.5`) especificadas:

```
menos_95_kg <- qnorm(p = 0.95, 1.8, 0.5)

cat(sprintf("El 95%% de los paquetes se encuentran por debajo de%.2fkg.\n", menos_95_kg))
```

```
## El 95% de los paquetes se encuentran por debajo de 2.62kg.
```

Para visualizar cómo se distribuye el peso de los paquetes basándonos en nuestros parámetros especificados de la distribución normal, podemos realizar una simulación. Utilizamos la función `rnorm` para generar 100 valores aleatorios que siguen una distribución normal con una media de 1.8 kg y una desviación estándar de 0.5 kg. Para garantizar que la simulación sea reproducible, establecemos una semilla inicial con `set.seed(1)`. Después de generar los datos simulados, creamos un histograma para visualizar la distribución del peso de los 100 paquetes simulados:

```
set.seed(1)
simu <- rnorm(n = 100, mean = 1.8, sd = 0.5)
hist(simu, main = "Distribución del Peso de los Paquetes", xlab = "Peso (kg)", ylab = "Frecuencia", col
```



Este histograma nos ofrece una representación gráfica de la distribución esperada del peso de los paquetes, permitiéndonos observar la variabilidad y la tendencia central de los datos simulados basados en la distribución normal.

Laboratorio de Muestreo y Estimadores

Carmen Lancho - Isaac Martín - Víctor Aceña

Grado en Ciencia e Ingeniería de Datos - Inferencia Estadística - Curso 2024/2025

Índice

Objetivo

El objetivo principal de este laboratorio es aprender a aplicar técnicas de **muestreo estadístico** y a calcular **estimaciones puntuales** de parámetros poblacionales a partir de una muestra. Como objetivos secundarios tenemos:

1. Comprender los distintos tipos de **muestreo probabilístico** y su aplicación en diferentes situaciones.
2. Saber calcular **estimaciones puntuales** y **estadísticos muestrales** en R.
3. Identificar y entender las **propiedades** de los estimadores, como **insesgadez**, **eficiencia**, **consistencia** y **suficiencia**, mediante ejemplos y simulaciones.
4. Saber interpretar los resultados obtenidos y su implicación en la inferencia sobre la población.

1. Introducción

En esta sección, introduciremos los conceptos básicos de **muestreo estadístico** y **estimación**. El muestreo es el proceso mediante el cual seleccionamos una parte de la población para obtener información sobre el todo. Los **estimadores** son las herramientas que utilizamos para aproximar los parámetros poblacionales (como la media o la proporción) a partir de la muestra obtenida.

Veremos los siguientes conceptos:

1. Tipos de muestreo: Muestreo aleatorio simple, estratificado, conglomerados y sistemático.
2. Definición de estimadores y su diferencia con los parámetros poblacionales.
3. Propiedades deseables de los estimadores: insesgadez, eficiencia, consistencia.

2. Tipos de Muestreo

Existen diferentes métodos de muestreo que podemos utilizar dependiendo de la situación y la naturaleza de la población. A continuación, se describen brevemente los métodos más comunes:

2.1. Muestreo Aleatorio Simple (MAS)

El **muestreo aleatorio simple** es el método más sencillo. Cada elemento de la población tiene la misma probabilidad de ser seleccionado. Es ideal para poblaciones pequeñas y homogéneas.

> **Ejemplo**: Si tenemos una lista de 100 estudiantes y queremos seleccionar una muestra aleatoria de 20, cada estudiante tiene una probabilidad igual de ser seleccionado.

```r
set.seed(123)
poblacion <- 1:100  # Población de 100 estudiantes
muestra <- sample(poblacion, size = 20, replace = FALSE)  # Seleccionar una muestra aleatoria de 20
```

En este ejemplo, realizamos un **muestreo aleatorio simple** en una población de 100 estudiantes para seleccionar una muestra de 20 estudiantes.

1. **Establecer la semilla para la reproducibilidad**:

   La función `set.seed(123)` fija la semilla de los números aleatorios en R. Esto asegura que cada vez que se ejecute el código, se obtenga la misma muestra aleatoria, permitiendo la **reproducibilidad** de los resultados.

2. **Definir la población**:

   Creamos un vector llamado `poblacion` que contiene los números del 1 al 100. Cada número representa a un estudiante único en la población total de 100 estudiantes.

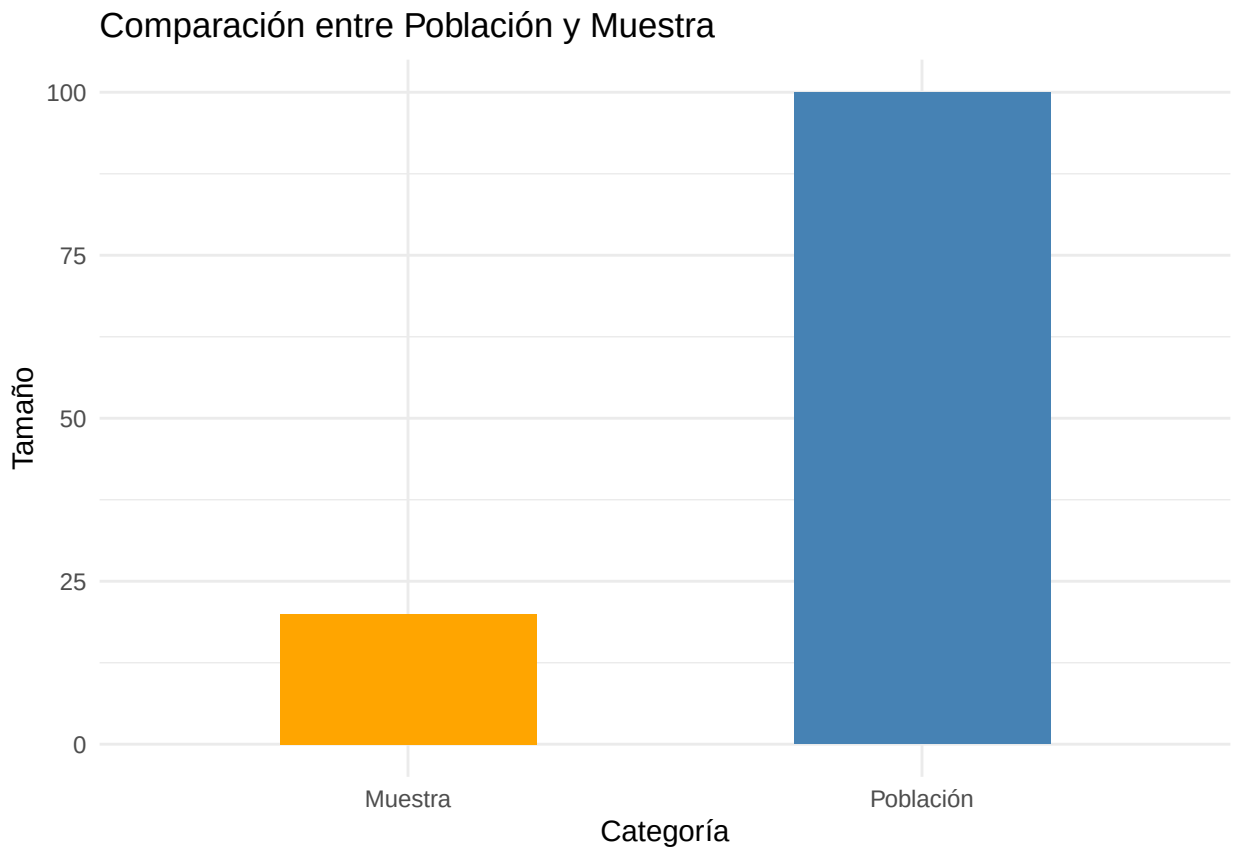
3. **Seleccionar la muestra aleatoria**:

   Utilizamos la función `sample()` para seleccionar una muestra aleatoria de 20 estudiantes de la población definida. El parámetro `size = 20` indica el número de estudiantes a seleccionar, y `replace = FALSE` asegura que la selección sea **sin reemplazo**, es decir, que cada estudiante solo pueda ser seleccionado una vez.

   > Si `replace = TRUE`, los elementos seleccionados podrían repetirse en la muestra, lo que sería un **muestreo con reemplazamiento**.

Este método garantiza que cada estudiante en la población tenga la misma probabilidad de ser seleccionado para la muestra, asegurando una **representación equitativa** y **aleatoria** de la población en la muestra.

```
# Crear un dataframe con la información de la población y la muestra
data_proporcion <- data.frame(
  Categoria = c("Población", "Muestra"),
  Tamano = c(length(poblacion), length(muestra))
)

# Crear el gráfico de barras
ggplot(data_proporcion, aes(x = Categoria, y = Tamano, fill = Categoria)) +
  geom_bar(stat = "identity", width = 0.5) +
  scale_fill_manual(values = c("Población" = "steelblue", "Muestra" = "orange")) +
  labs(title = "Comparación entre Población y Muestra",
       x = "Categoría",
       y = "Tamaño") +
  theme_minimal() +
  theme(legend.position = "none")
```



Este gráfico de barras destaca visualmente la diferencia entre el número total de estudiantes en la población y los que fueron seleccionados para formar parte de la muestra.

¿Cuándo usar el Muestreo Aleatorio Simple?

Este tipo de muestreo es ideal cuando:

- La población es **pequeña** y homogénea.

- Todos los elementos de la población tienen la **misma probabilidad** de ser seleccionados.
- No existen diferencias importantes entre subgrupos dentro de la población.

Sin embargo, puede no ser el método adecuado para poblaciones grandes o cuando existen diferencias significativas entre los subgrupos, en cuyo caso se podrían utilizar otros métodos como el **muestreo estratificado**.

2.2. Muestreo Estratificado

El **muestreo estratificado** se utiliza cuando la población no es homogénea, pero sabemos que podemos dividirla en grupos (estratos) homogéneos. De cada estrato, se selecciona una muestra aleatoria. Este método es útil cuando los subgrupos dentro de la población tienen características que afectan el estudio y queremos asegurarnos de que estén representados adecuadamente.

> **Ejemplo**: Si queremos muestrear a estudiantes de diferentes carreras (estratos), podemos dividir la población de estudiantes en función de la carrera y seleccionar una muestra aleatoria de cada carrera para asegurar que todas las carreras estén representadas en nuestra muestra.

```r
# Simulación de una población de estudiantes clasificados por carrera
set.seed(123)
poblacion <- data.frame(
  id = 1:100,  # Identificación de estudiantes
  carrera = sample(c("Ingeniería", "Medicina", "Derecho", "Ciencias Sociales"), 100, replace = TRUE)
)

# Tamaño de la muestra total
n_muestra <- 20

# Muestreo estratificado: número de estudiantes seleccionados por carrera
muestra_estratificada <- poblacion |>
  group_by(carrera) |>
  sample_n(size = round(n_muestra * n() / nrow(poblacion)), replace = FALSE)
```

En este ejemplo, realizamos un **muestreo estratificado** en una población de 100 estudiantes distribuidos en diferentes carreras.

1. **Crear la población**:

   Creamos un dataframe llamado `poblacion` que contiene 100 estudiantes. Cada estudiante tiene un `id` único del 1 al 100 y está asignado aleatoriamente a una de las 4 carreras (**Ingeniería**, **Medicina**, **Derecho**, **Ciencias Sociales**) utilizando la función `sample()`. Esto simula una población diversificada donde cada carrera puede tener una representación diferente.

2. **Determinar el tamaño de la muestra**:

   Definimos que queremos una muestra total de 20 estudiantes. Este número determina cuántos estudiantes serán seleccionados en total a partir de la población.

3. **Calcular el tamaño de la muestra por estrato**:

   Calculamos cuántos estudiantes se deben seleccionar de cada carrera (estrato) proporcionalmente al tamaño de cada carrera en la población total. Utilizamos la fórmula `round(n_muestra * n() / nrow(poblacion))` para asegurarnos de que la cantidad de estudiantes seleccionados de cada carrera refleje su proporción en la población. Esto garantiza que cada estrato esté representado adecuadamente en la muestra final.

4. **Seleccionar la muestra estratificada**:

   Utilizamos la función `group_by()` para agrupar los estudiantes por su carrera. Luego, aplicamos `sample_n()` para seleccionar una muestra aleatoria dentro de cada grupo estratificado, según el tamaño calculado previamente. Este proceso asegura que la muestra obtenida sea representativa de la diversidad de carreras presentes en la población.

5. **Obtener la muestra final**:

   La muestra estratificada resultante contiene una representación proporcional de cada carrera, de acuerdo con su presencia en la población total. Esto mejora la precisión y representatividad de las estimaciones realizadas a partir de la muestra, ya que cada subgrupo relevante está adecuadamente representado.

Este método asegura que cada carrera esté representada adecuadamente en la muestra final, mejorando la **representatividad** y **precisión** de las estimaciones obtenidas a partir de la muestra.

Detalle de las Funciones Utilizadas

1. `sample_n()`:

   La función `sample_n()` se utiliza para seleccionar una muestra aleatoria dentro de cada estrato. En este caso, garantiza que se seleccione un número de estudiantes proporcional al tamaño de cada grupo en la población total. Esto asegura que las muestras de cada carrera sean representativas y reflejen la distribución real en la población.

2. `group_by()`:

   `group_by()` agrupa a los estudiantes por la variable `carrera`. De esta manera, se pueden aplicar operaciones (como `sample_n()`) a cada grupo por separado. En este ejemplo, primero agrupamos los estudiantes según la carrera y luego aplicamos el muestreo dentro de cada grupo, lo que facilita una selección proporcional y representativa.

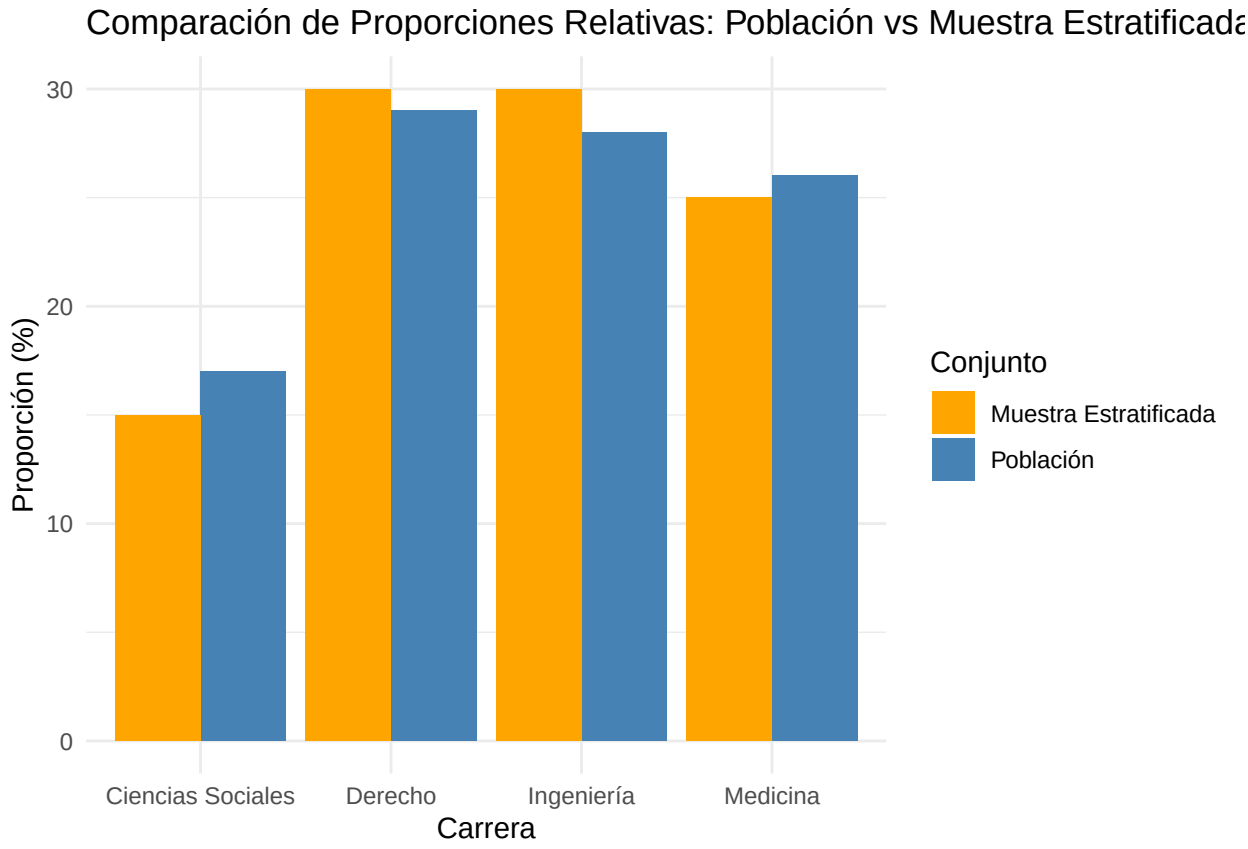
3. `round(n_muestra * n() / nrow(poblacion))`:

   Esta fórmula calcula el número de estudiantes que se deben seleccionar de cada grupo (estrato) en función del tamaño total de la muestra. El uso de `round()` asegura que el número de estudiantes seleccionados sea un número entero, manteniendo la proporcionalidad y evitando fracciones de estudiantes.

Este enfoque de muestreo estratificado mejora la **exactitud** y **eficiencia** de las estimaciones estadísticas al asegurar que todos los subgrupos importantes de la población estén adecuadamente representados en la muestra.

```r
# Calcular la proporción de cada carrera en la población
proporcion_poblacion <- poblacion |>
  count(carrera) |>
  mutate(prop_poblacion = n / sum(n)) |>
  select(carrera, prop_poblacion)

# Calcular la proporción de cada carrera en la muestra, pero con respecto al total de la muestra (n_mue
proporcion_muestra <- muestra_estratificada |>
  count(carrera) |>
  mutate(prop_muestra = n / sum(n_muestra)) |>
  select(carrera, prop_muestra)

# Unir las proporciones de población y muestra
proporciones_comparadas <- left_join(proporcion_poblacion, proporcion_muestra, by = "carrera")
```

```r
# Convertir a formato largo para ggplot
proporciones_long <- proporciones_comparadas |>
  pivot_longer(cols = starts_with("prop"), names_to = "Conjunto", values_to = "Proporción")

# Renombrar las categorías para hacerlas más claras en el gráfico
proporciones_long$Conjunto <- recode(proporciones_long$Conjunto,
                                     "prop_poblacion" = "Población",
                                     "prop_muestra" = "Muestra Estratificada")

# Crear el gráfico de barras comparando las proporciones
ggplot(proporciones_long, aes(x = carrera, y = Proporción * 100, fill = Conjunto)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Comparación de Proporciones Relativas: Población vs Muestra Estratificada",
       x = "Carrera",
       y = "Proporción (%)") +
  scale_fill_manual(values = c("Población" = "steelblue", "Muestra Estratificada" = "orange")) +
  theme_minimal()
```



Este gráfico de barras compara las proporciones relativas de cada carrera en la **población total** con las proporciones en la **muestra estratificada**. En el eje **X** se representan las diferentes carreras (**Ingeniería**, **Medicina**, **Derecho**, **Ciencias Sociales**), mientras que en el eje **Y** se muestra la proporción porcentual de estudiantes en cada carrera.

- **Población**: Las barras azules indican la proporción de cada carrera dentro de la población completa de 100 estudiantes. Esto refleja la distribución original de las carreras en la población.

- **Muestra Estratificada**: Las barras naranjas representan la proporción de cada carrera en la muestra estratificada de 20 estudiantes. Dado que se ha aplicado un muestreo estratificado proporcional, las proporciones en la muestra reflejan de manera fiel las proporciones observadas en la población.

Este gráfico demuestra que el **muestreo estratificado** ha logrado mantener la representatividad de cada subgrupo (carrera) en la muestra. Al comparar las barras azules y naranjas, se observa que las proporciones en la muestra son consistentes con las de la población, lo que indica una selección equilibrada y proporcional de cada carrera.

Esta representatividad es crucial para asegurar que las estimaciones y conclusiones derivadas de la muestra sean precisas y reflejen adecuadamente las características de la población total. En contraste, métodos de muestreo no estratificados podrían resultar en desequilibrios y sesgos, afectando la validez de los análisis estadísticos posteriores.

¿Cuándo usar el Muestreo Estratificado?

Este tipo de muestreo es ideal cuando:

- La población es **heterogénea** y se puede dividir en **subgrupos (estratos)** que son internamente homogéneos pero diferentes entre sí.
- Queremos asegurar que cada estrato esté **adecuadamente representado** en la muestra.
- Buscamos **mejorar la precisión** de las estimaciones, especialmente cuando los subgrupos tienen características que afectan el estudio.

Sin embargo, el muestreo estratificado requiere **conocer previamente la estructura** de la población para identificar los estratos, lo que puede no ser viable si no tenemos esta información disponible. En poblaciones pequeñas o homogéneas, puede ser más adecuado el **muestreo aleatorio simple**.

2.3. Comparación entre MAS y Muestreo Estratificado

Vamos a realizar una comparación entre el Muestreo Aleatorio Simple (MAS) y el Muestreo Estratificado. Para ello, tomaremos muestras repetidamente de la población y compararemos la proporción de estudiantes de "Ingeniería" en cada muestra utilizando ambos métodos.

2.3.1. Simulación de Muestras Repetidas

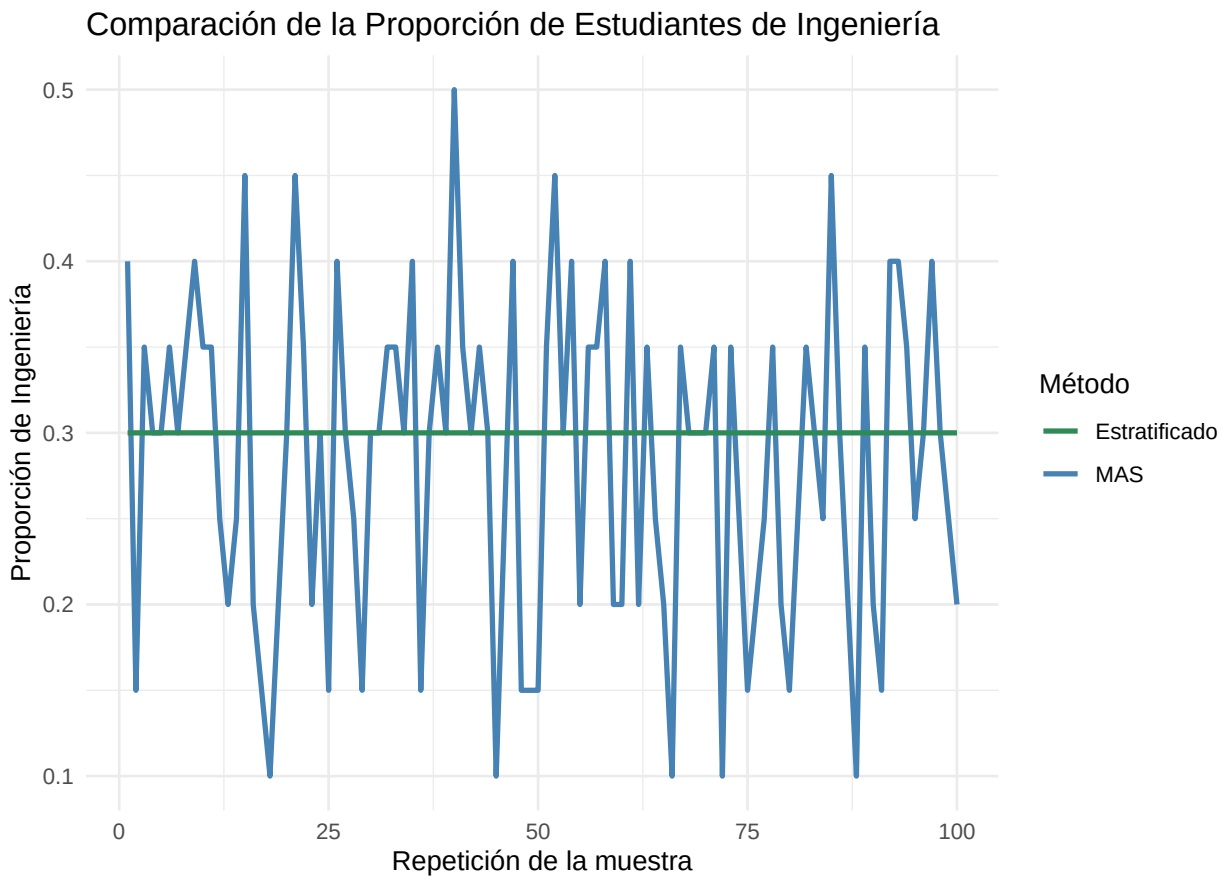
Para entender mejor las diferencias entre MAS y el Muestreo Estratificado, realizaremos una **simulación** en la que extraeremos **100 muestras** utilizando cada método. En cada muestra, calcularemos la **proporción de estudiantes de "Ingeniería"**. Este enfoque nos permitirá observar cómo varían las estimaciones según el método de muestreo utilizado.

Procedimiento de la Simulación

1. **Configuración Inicial**:
   - **Número de Repeticiones** `n_reps`: 100
   - **Tamaño de la Muestran_muestra**: 20 estudiantes
2. **Definición de Funciones de Muestreo**:
   - **MAS**: Selecciona 20 estudiantes al azar de toda la población sin considerar la estructura de la misma.
   - **Estratificado**: Divide la población en estratos (en este caso, carreras) y selecciona una muestra proporcional de cada estrato.

3. **Ejecución de la Simulación**:
   - Utilizamos la función `replicate()` para repetir el proceso de muestreo 100 veces para cada método.
   - Calculamos la proporción de estudiantes de "Ingeniería" en cada muestra obtenida.
4. **Almacenamiento de Resultados**:
   - Creamos un dataframe que contiene las proporciones obtenidas en cada repetición para ambos métodos, lo que nos permitirá compararlos fácilmente.

```r
# Parámetros de la simulación
n_reps <- 100  # Número de repeticiones
n_muestra <- 20  # Tamaño de la muestra total

# Función para obtener la proporción de "Ingeniería" en MAS
mas_sim <- function() {
  muestra_mas <- sample(poblacion$id, size = n_muestra, replace = FALSE)
  prop_ing_mas <- mean(poblacion$carrera[muestra_mas] == "Ingeniería")
  return(prop_ing_mas)
}

# Función para obtener la proporción de "Ingeniería" en Muestreo Estratificado
estrat_sim <- function() {
  muestra_estrat <- poblacion |>
    group_by(carrera) |>
    sample_n(size = round(n_muestra * n() / nrow(poblacion)), replace = FALSE)
  prop_ing_estrat <- mean(muestra_estrat$carrera == "Ingeniería")
  return(prop_ing_estrat)
}

# Simulación
set.seed(123)
resultados_mas <- replicate(n_reps, mas_sim())
resultados_estrat <- replicate(n_reps, estrat_sim())

# Crear un data frame con los resultados
resultados <- data.frame(
  rep = 1:n_reps,
  MAS = resultados_mas,
  Estratificado = resultados_estrat
)

# Graficar los resultados de MAS y Estratificado
ggplot(resultados, aes(x = rep)) +
  geom_line(aes(y = MAS, color = "MAS"), size = 1) +
  geom_line(aes(y = Estratificado, color = "Estratificado"), size = 1) +
  labs(title = "Comparación de la Proporción de Estudiantes de Ingeniería",
       x = "Repetición de la muestra",
       y = "Proporción de Ingeniería",
       color = "Método") +
  scale_color_manual(values = c("MAS" = "steelblue", "Estratificado" = "seagreen")) +
  theme_minimal()
```



Este gráfico de líneas compara la proporción de estudiantes de **Ingeniería** obtenida mediante dos métodos de muestreo: **Muestreo Aleatorio Simple (MAS)** y **Muestreo Estratificado**, a lo largo de 100 repeticiones de muestras.

- **Eje X**: Representa el número de repeticiones de la muestra (de 1 a 100).
- **Eje Y**: Representa la proporción de estudiantes de Ingeniería en cada muestra.

Elementos del Gráfico

- **Línea Azul (MAS)**: Muestra la proporción de Ingeniería obtenida a través del Muestreo Aleatorio Simple en cada una de las 100 repeticiones.
- **Línea Verde (Estratificado)**: Muestra la proporción de Ingeniería obtenida a través del Muestreo Estratificado en cada una de las 100 repeticiones.

Observaciones

1. **Variabilidad**:
   - **MAS**: La línea azul presenta una mayor variabilidad en la proporción de Ingeniería entre las diferentes muestras. Esto indica que las proporciones obtenidas pueden fluctuar significativamente alrededor de la proporción real en la población.
   - **Estratificado**: La línea verde muestra una menor variabilidad, con proporciones de Ingeniería más consistentes y cercanas a la proporción real de la población.
2. **Consistencia**:

- **MAS**: Al ser un método completamente aleatorio, las muestras pueden no reflejar de manera precisa la estructura de la población, lo que resulta en fluctuaciones más amplias en las proporciones.
- **Estratificado**: Al garantizar una representación proporcional de cada estrato (en este caso, cada carrera), el muestreo estratificado logra una mayor consistencia en las proporciones obtenidas en cada repetición.

Implicaciones

- **Precisión y Representatividad**:
  - El **muestreo estratificado** proporciona una representación más precisa y consistente de subgrupos específicos dentro de la población, reduciendo la variabilidad de las estimaciones en comparación con el Muestreo Aleatorio Simple.
- **Elección del Método de Muestreo**:
  - Cuando es importante asegurar la representatividad de subgrupos específicos (como una carrera particular), el muestreo estratificado es preferible debido a su mayor consistencia y menor variabilidad en las estimaciones.
  - Por otro lado, el **MAS** puede ser adecuado en situaciones donde la población es homogénea o cuando se busca simplicidad, aunque a costa de una mayor variabilidad en las estimaciones.

Conclusión

Este gráfico demuestra que el **muestreo estratificado** es más efectivo para obtener proporciones consistentes y representativas de subgrupos específicos dentro de una población. Al reducir la variabilidad en las estimaciones, este método mejora la precisión de las inferencias estadísticas realizadas a partir de la muestra, lo que es fundamental para obtener conclusiones fiables en estudios estadísticos.

2.4. Muestreo por Conglomerados

El **muestreo por conglomerados** se utiliza cuando la población está naturalmente dividida en grupos o **conglomerados** (por ejemplo, aulas, escuelas, barrios). En lugar de muestrear individuos de toda la población, se seleccionan aleatoriamente algunos conglomerados y se muestrean todos (o una muestra) los individuos dentro de esos conglomerados seleccionados. Este método es útil cuando es difícil o costoso enumerar a todos los individuos de la población, pero es más fácil listar los conglomerados.

> **Ejemplo**: Supongamos que queremos muestrear estudiantes de diferentes aulas en una escuela. En lugar de seleccionar estudiantes al azar de toda la escuela, primero seleccionamos algunas aulas al azar y luego muestreamos a todos los estudiantes dentro de esas aulas seleccionadas.

```r
# Simulación de una población de estudiantes clasificados por aula
set.seed(123)
poblacion <- data.frame(
  id = 1:100,  # Identificación de estudiantes
  aula = sample(paste("Aula", 1:10), 100, replace = TRUE)
)

# Tamaño de la muestra total
n_muestra <- 20

# Número de conglomerados a seleccionar (por ejemplo, 2 aulas)
```

```r
n_conglomerados <- 2

# Seleccionar aleatoriamente los conglomerados
conglomerados_seleccionados <- sample(unique(poblacion$aula), size = n_conglomerados)

# Seleccionar todos los estudiantes dentro de los conglomerados seleccionados
muestra_conglomerados <- poblacion |>
  filter(aula %in% conglomerados_seleccionados)
```

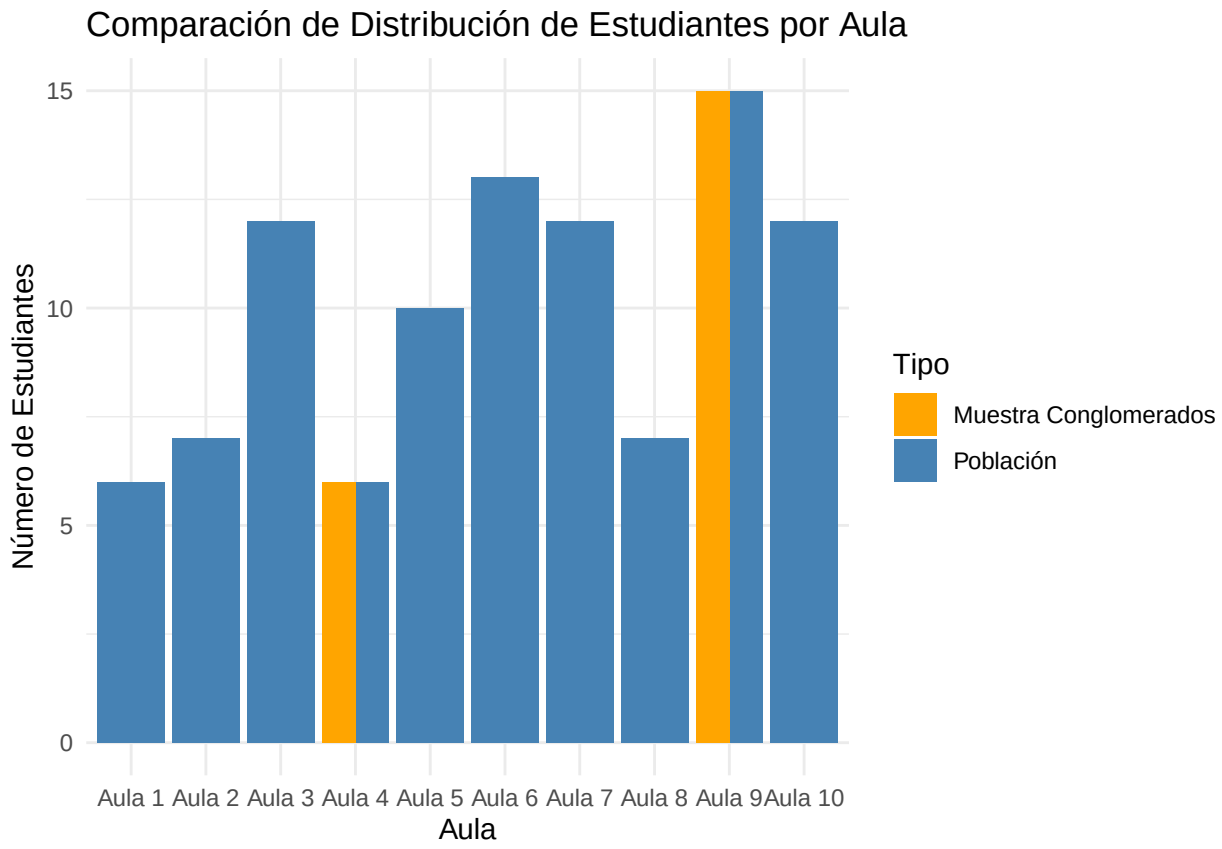
En este ejemplo, realizamos un **muestreo por conglomerados** en una población de 100 estudiantes distribuidos en 10 aulas.

1. **Crear la población**: Creamos un dataframe llamado `poblacion` que contiene 100 estudiantes, cada uno con un `id` del 1 al 100 y asignándolos aleatoriamente a una de las 10 aulas usando la función `sample()`.
2. **Determinar el tamaño de la muestra**: Definimos que queremos una muestra total de 20 estudiantes.
3. **Determinar el número de conglomerados a seleccionar**: Decidimos seleccionar 2 aulas al azar de las 10 disponibles.
4. **Seleccionar los conglomerados**: Usamos la función `sample()` para seleccionar aleatoriamente 2 aulas de las 10.
5. **Seleccionar todos los estudiantes dentro de los conglomerados seleccionados**: Filtramos el dataframe `poblacion` para incluir solo a los estudiantes que están en las aulas seleccionadas, formando así la muestra por conglomerados.

Este método asegura que la muestra esté concentrada en los conglomerados seleccionados, lo que puede ser eficiente en términos de costos y logística cuando los conglomerados son internamente heterogéneos pero homogéneos entre sí.

```r
# Contar el número de estudiantes por aula en la población
poblacion_aulas <- poblacion |>
  count(aula) |>
  mutate(Tipo = "Población")

# Contar el número de estudiantes por aula en la muestra por conglomerados
muestra_aulas <- muestra_conglomerados |>
  count(aula) |>
  mutate(Tipo = "Muestra Conglomerados")

# Unir los datos para el gráfico
comparacion_aulas <- bind_rows(poblacion_aulas, muestra_aulas)

# Definir el orden de las aulas correctamente, como "Aula 1", "Aula 2", etc.
orden_aulas <- paste("Aula", 1:10)

# Convertir la variable 'aula' en factor con el orden deseado
comparacion_aulas$aula <- factor(comparacion_aulas$aula, levels = orden_aulas)

# Crear el gráfico de barras
ggplot(comparacion_aulas, aes(x = aula, y = n, fill = Tipo)) +
  geom_bar(stat = "identity", position = "dodge") +
```

```
  labs(title = "Comparación de Distribución de Estudiantes por Aula",
       x = "Aula",
       y = "Número de Estudiantes",
       fill = "Tipo") +
  scale_fill_manual(values = c("Población" = "steelblue", "Muestra Conglomerados" = "orange")) +
  theme_minimal()
```



Este gráfico de barras compara la distribución de estudiantes por aula en la población total y en la muestra seleccionada por conglomerados. Podemos ver cómo los estudiantes se agrupan en los conglomerados seleccionados, lo que es característico del muestreo por conglomerados.

¿Cuándo usar el Muestreo por Conglomerados?

Este tipo de muestreo es ideal cuando:

- La población está naturalmente dividida en **grupos o conglomerados**.
- Es **costoso o difícil enumerar** a todos los individuos de la población.
- Los conglomerados son **internamente heterogéneos** pero **homogéneos entre sí**.

Sin embargo, puede no ser el método adecuado si los conglomerados son **internamente homogéneos**, ya que podría aumentar la **variabilidad de las estimaciones**.

2.5. Muestreo Sistemático

El **muestreo sistemático** se utiliza cuando la población está **ordenada** de cierta manera (por ejemplo, lista alfabética, numérica). En lugar de seleccionar elementos al azar de toda la población, se elige un punto

de inicio aleatorio y luego se seleccionan elementos a intervalos regulares a partir de ese punto. Este método es útil cuando se desea una **distribución uniforme** de la muestra a lo largo de la población y cuando es más sencillo aplicar una regla sistemática de selección que un muestreo completamente aleatorio.

Para implementar el muestreo sistemático, se utiliza el parámetro k, conocido como el **intervalo de muestreo**, que se calcula dividiendo el tamaño de la población N entre el tamaño de la muestra deseada n:

$$k = \frac{N}{n}$$

Donde N es el tamaño total de la población y n el de la muestra.

Pasos para el Muestreo Sistemático

1. **Calcular el intervalo k**:

$$k = \frac{N}{n}$$

2. **Seleccionar un punto de inicio aleatorio**:
   Se elige un número aleatorio entre 1 y k para determinar el primer elemento de la muestra.

3. **Seleccionar cada k-ésimo elemento**:
   A partir del punto de inicio, se selecciona cada k-ésimo elemento para formar la muestra.

> **Ejemplo**: Supongamos que queremos seleccionar cada 5º estudiante de una lista ordenada de 100 estudiantes. Primero, calculamos el intervalo `k = 5` y luego seleccionamos un punto de inicio aleatorio entre 1 y 5. A partir de ese punto, tomamos cada 5º estudiante para formar la muestra.

```r
# Simulación de una población ordenada de estudiantes
set.seed(123)
poblacion <- data.frame(
  id = 1:100,  # Identificación de estudiantes
  nombre = paste("Estudiante", 1:100)
)

# Tamaño de la muestra total
n_muestra <- 20

# Calcular el intervalo k
k <- floor(nrow(poblacion) / n_muestra)

# Seleccionar un punto de inicio aleatorio entre 1 y k
inicio <- sample(1:k, 1)

# Seleccionar cada k-ésimo estudiante a partir del punto de inicio
indices <- seq(from = inicio, to = nrow(poblacion), by = k)
muestra_sistematico <- poblacion[indices, ]
```

En este ejemplo, realizamos un **muestreo sistemático** de una población ordenada de 100 estudiantes para seleccionar una muestra de 20 estudiantes.

1. **Crear la población**: Creamos un dataframe llamado `poblacion` que contiene 100 estudiantes, cada uno con un `id` del 1 al 100 y un `nombre` correspondiente.

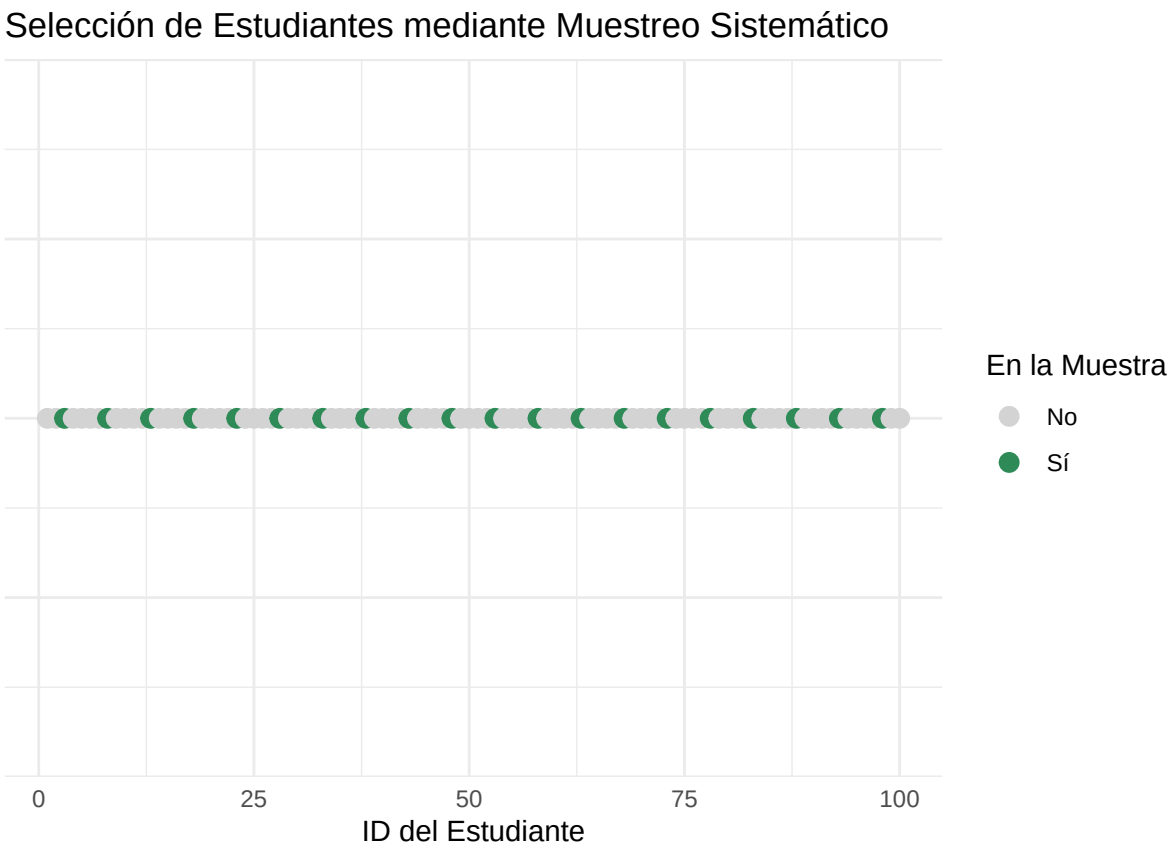
2. **Determinar el tamaño de la muestra**: Definimos que queremos una muestra de 20 estudiantes.

3. **Calcular el intervalo `k`**: Calculamos el intervalo `k` dividiendo el tamaño de la población entre el tamaño de la muestra, es decir, `k = floor(100 / 20) = 5`. Esto significa que seleccionaremos cada 5^{o} estudiante.

4. **Seleccionar el punto de inicio**: Elegimos aleatoriamente un punto de inicio entre 1 y `k` (1 y 5). Por ejemplo, si el punto de inicio es 3, seleccionaremos estudiantes con IDs 3, 8, 13, . . . , 98.

5. **Seleccionar cada `k`-ésimo estudiante**: Usamos la función `seq()` para generar una secuencia de índices desde el punto de inicio hasta el tamaño de la población, con un intervalo de `k`. Luego, seleccionamos esos estudiantes para formar la muestra sistemática.

```r
# Añadir una columna para indicar si el estudiante está en la muestra
poblacion$Muestra_Sistematico <- ifelse(poblacion$id %in% muestra_sistematico$id, "Sí", "No")

ggplot(poblacion, aes(x = id, y = 1)) +
  geom_point(aes(color = Muestra_Sistematico), size = 3) +
  scale_color_manual(values = c("Sí" = "seagreen", "No" = "lightgray")) +
  labs(title = "Selección de Estudiantes mediante Muestreo Sistemático",
       x = "ID del Estudiante",
       y = "",
       color = "En la Muestra") +
  theme_minimal() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```



Este gráfico muestra la selección de estudiantes mediante **muestreo sistemático** en una población ordenada. Podemos ver cómo los estudiantes son seleccionados a intervalos regulares `k`, lo que garantiza una **distribución uniforme** de la muestra a lo largo de la población. Esto es característico del muestreo sistemático, que busca una representación equitativa mediante una selección sistemática de elementos.

¿Cuándo usar el Muestreo Sistemático?

Este tipo de muestreo es ideal cuando:

- La población está **ordenada** de alguna manera (por ejemplo, lista alfabética, numérica).
- Se desea una **distribución uniforme** de la muestra a lo largo de la población.
- Es más **sencillo y rápido** que el muestreo aleatorio simple, especialmente para poblaciones grandes.

Sin embargo, puede no ser el método adecuado si existe un **patrón periódico** en la población que coincide con el intervalo de muestreo, lo que podría sesgar la muestra.

3. Estimadores

En esta sección, profundizaremos en los conceptos fundamentales relacionados con los **estimadores** y sus **propiedades** en el contexto del muestreo estadístico. Comprender estos conceptos es esencial para realizar inferencias precisas sobre una población a partir de una muestra.

3.1. ¿Qué es un Estimador?

Un **estimador** es una regla, fórmula o función utilizada para calcular una **estimación** de un **parámetro poblacional** basándose en los datos de una **muestra**. Los estimadores son funciones de los datos muestrales

y permiten inferir características de la población completa sin necesidad de examinar todos sus elementos.

3.1.1. Características de los Estimadores

- **Función de la Muestra**: Un estimador es una función matemática que toma los datos de la muestra como entrada y produce una estimación del parámetro poblacional. Por ejemplo, la media muestral $\bar{x}$ se calcula sumando todos los valores de la muestra y dividiendo entre el tamaño de la muestra.
- **Objetivo**: El objetivo principal de un estimador es proporcionar una estimación lo más cercana posible al verdadero parámetro poblacional. Para lograr esto, los estimadores deben poseer ciertas propiedades que garantizan su calidad y precisión.
- **Estimación vs Parámetro**:
  - **Estimación**: Es el valor calculado a partir de la muestra, como la media muestral $\bar{x}$ o la proporción muestral $\hat{p}$.
  - **Parámetro**: Es el valor real en la población, como la media poblacional μ o la proporción poblacional p.

3.2. Estimadores Comunes

A continuación, se presentan algunos de los estimadores más utilizados en estadística, junto con sus fórmulas y descripciones:

Media Muestral $\bar{x}$

La **media muestral** es uno de los estimadores más comunes para la **media poblacional** μ. Representa el promedio de los valores en la muestra.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- n: Tamaño de la muestra.
- x_i: Valores individuales en la muestra.

Interpretación: La media muestral proporciona una estimación central de los datos de la muestra y, bajo ciertas condiciones, es un buen estimador de la media poblacional.

Proporción Muestral $\hat{p}$

La **proporción muestral** estima la **proporción poblacional** p. Es especialmente útil en estudios de encuestas y análisis de características categóricas.

$$\hat{p} = \frac{\text{Número de éxitos en la muestra}}{n}$$

- **Número de éxitos**: Cantidad de observaciones que cumplen con la característica de interés.
- n: Tamaño de la muestra.

Interpretación: La proporción muestral indica la fracción de la muestra que posee una determinada característica, proporcionando una estimación directa de la proporción en la población.

Cuasivarianza s^2

La **cuasivarianza** estima la **varianza poblacional** σ^2. Mide la dispersión de los datos alrededor de la media muestral.

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

- n: Tamaño de la muestra.
- x_i: Valores individuales en la muestra.
- $\bar{x}$: Media muestral.

Interpretación: La cuasivarianza cuantifica la variabilidad de los datos en la muestra, proporcionando información sobre la dispersión de los valores individuales respecto a la media.

3.3. Propiedades de los Estimadores

Los estimadores poseen ciertas **propiedades** que determinan su calidad y adecuación para inferir parámetros poblacionales. Las propiedades más importantes son:

3.3.1. Insesgadez

Un estimador es **insesgado** si su valor esperado es igual al parámetro que estima. Es decir, en promedio, el estimador no sobreestima ni subestima el parámetro poblacional.

$$E(\hat{\theta}) = \theta$$

> **Ejemplo**: La media muestral $\bar{x}$ es un estimador insesgado de la media poblacional μ. Esto implica que, en promedio, la media muestral coincide con la media poblacional, garantizando que $\bar{x}$ no tiende a sobrestimar ni subestimar el valor real de μ. Para ilustrar la **insesgadez** de la media muestral, realizaremos una simulación en R con 1000 repeticiones de una distribución normal $N(50, 10)$, donde compararemos la media de las medias muestrales con la media poblacional para verificar que ambas son prácticamente iguales.

A continuación, presentamos una simulación que verifica esta propiedad.

```r
# Simulación para demostrar la insesgadez de la media muestral
set.seed(123)        # Fijar la semilla para reproducibilidad
n <- 30              # Tamaño de la muestra
mu <- 50             # Media poblacional
sigma <- 10          # Desviación estándar poblacional
reps <- 1000         # Número de repeticiones

# Generar múltiples muestras y calcular sus medias
medias_muestrales <- replicate(reps, {
  muestra <- rnorm(n, mean = mu, sd = sigma)  # Generar una muestra de tamaño n
  mean(muestra)                               # Calcular la media muestral
})

# Calcular la media de las medias muestrales
media_de_medias <- mean(medias_muestrales)
```

Configuración Inicial:

- `set.seed(123)`: Fija la semilla de los números aleatorios para asegurar que los resultados sean reproducibles.
- `n <- 30`: Define el tamaño de cada muestra.
- `mu <- 50`: Establece la media poblacional.
- `sigma <- 10`: Define la desviación estándar poblacional.
- `reps <- 1000`: Determina el número de repeticiones de la simulación.

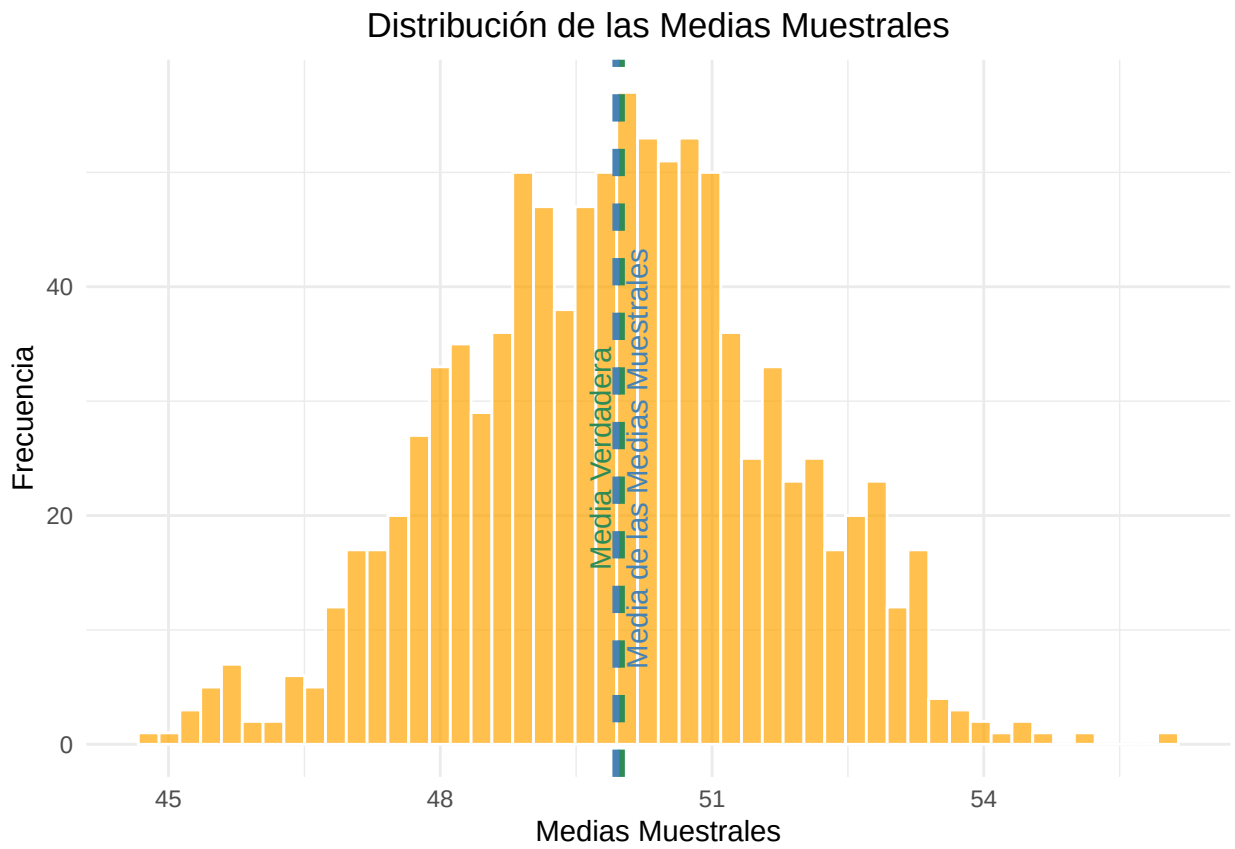
Generación de Muestras y Cálculo de Medias:

- `replicate(reps, { ... })`: Repite el proceso de generación de muestras y cálculo de sus medias 1000 veces.
  - Dentro de `replicate`:
    - `rnorm(n, mean = mu, sd = sigma)`: Genera una muestra aleatoria de tamaño `n` de una distribución normal con media `mu` y desviación estándar `sigma`.
    - `mean(muestra)`: Calcula la media de la muestra generada.

Cálculo de la Media de las Medias Muestrales:

- `media_de_medias <- mean(medias_muestrales)`: Calcula la media de todas las medias muestrales obtenidas en las 1000 repeticiones.

```
# Crear un data frame para ggplot
datos <- data.frame(medias_muestrales)

# Calcular la media de las medias muestrales
media_de_medias <- mean(datos$medias_muestrales)

# Definir la media poblacional
media_poblacional <- mu

# Graficar las medias muestrales usando ggplot2
ggplot(datos, aes(x = medias_muestrales)) +
  geom_histogram(bins = 50, fill = "orange", color = "white", alpha = 0.7) +
  geom_vline(aes(xintercept = media_poblacional), color = "seagreen", linetype = "dashed", size = 1.2) +
  geom_vline(aes(xintercept = media_de_medias), color = "steelblue", linetype = "dashed", size = 1.2) +
  labs(title = "Distribución de las Medias Muestrales",
       x = "Medias Muestrales",
       y = "Frecuencia") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  annotate("text", x = media_poblacional, y = max(table(datos$medias_muestrales)) * 25,
           label = "Media Verdadera", color = "seagreen", angle = 90, vjust = -0.5) +
  annotate("text", x = media_de_medias, y = max(table(datos$medias_muestrales)) * 25,
           label = "Media de las Medias Muestrales", color = "steelblue", angle = 90, vjust = 1.5)
```



El gráfico resultante muestra la distribución de las medias muestrales obtenidas a lo largo de las 1000 repeticiones. Observamos lo siguiente:

- **Histograma**: Representa la frecuencia de las diferentes medias muestrales. La mayoría de las medias muestrales se concentran alrededor de la media poblacional $\mu = 50$, formando una distribución aproximadamente normal debido al **Teorema del Límite Central**.
- **Línea Verde (Media Verdadera)**: Indica la media poblacional $\mu = 50$. Es el valor al cual esperamos que las medias muestrales se acerquen en promedio.
- **Línea Azul (Media de las Medias Muestrales)**: Representa la media de todas las medias muestrales calculadas en la simulación. Debería estar muy cercana a la media poblacional, confirmando que $\bar{x}$ es un estimador insesgado de μ.

Este gráfico visualiza cómo, a pesar de la variabilidad inherente en cada muestra individual, la media de las medias muestrales converge hacia la media poblacional, demostrando la **insesgadez** de la media muestral.

3.3.2. Eficiencia

Entre los estimadores insesgados, el **eficiente** es aquel que tiene la menor varianza posible. Un estimador eficiente proporciona estimaciones más precisas al reducir la dispersión en torno al valor del parámetro poblacional.

$$\text{Var}(\hat{\theta}_1) \leq \text{Var}(\hat{\theta}_2)$$

Ejemplo: Dentro de los estimadores insesgados de la media, la **media muestral** $\bar{x}$ es más eficiente que la **mediana muestral**. Para demostrar la **eficiencia** de la media muestral, compararemos la varianza de ambos estimadores utilizando una simulación de 1000 repeticiones de una distribución normal $N(50, 10)$.

A continuación, presentamos una simulación en R para comparar la varianza de la media y la mediana muestral.

```
# Simulación para comparar la eficiencia de la media y la mediana muestral
set.seed(123)
n <- 30  # Tamaño de la muestra
mu <- 50  # Media poblacional
sigma <- 10  # Desviación estándar poblacional
reps <- 1000  # Número de repeticiones

# Generar muestras y calcular la media y mediana muestral
resultados <- replicate(reps, {
  muestra <- rnorm(n, mean = mu, sd = sigma)
  c(mean = mean(muestra), median = median(muestra))
})

# Extraer las medias y medianas muestrales
medias_muestrales <- resultados[1, ]  # Primer fila: medias
medianas_muestrales <- resultados[2, ]  # Segunda fila: medianas

# Calcular las varianzas
var_media <- var(medias_muestrales)
var_mediana <- var(medianas_muestrales)

# Imprimir las varianzas
var_media
```

```
## [1] 3.15235
```

```
var_mediana
```

```
## [1] 4.75556
```

Configuración Inicial:

- `set.seed(123)`: Fija la semilla de los números aleatorios para asegurar que los resultados sean reproducibles.
- `n <- 30`: Define el tamaño de cada muestra.
- `mu <- 50`: Establece la media poblacional.
- `sigma <- 10`: Define la desviación estándar poblacional.
- `reps <- 1000`: Determina el número de repeticiones de la simulación.

Generación de Muestras y Cálculo de Medias:

- `replicate(reps, { ... })`: Repite el proceso de generación de muestras y cálculo de sus medias 1000 veces.

- Dentro de `replicate`:
  - `rnorm(n, mean = mu, sd = sigma)`: Genera una muestra aleatoria de tamaño `n` de una distribución normal con media `mu` y desviación estándar `sigma`.
  - `mean(muestra)`: Calcula la media de la muestra generada.

Generación de Muestras y Cálculo de Medianas:

- Similar al proceso anterior, pero en lugar de calcular la media, se calcula la mediana de cada muestra.

Cálculo de las Varianzas de los Estimadores:

- `var_media <- var(medias_muestrales)`: Calcula la varianza de las medias muestrales obtenidas en las 1000 repeticiones.
- `var_mediana <- var(medianas_muestrales)`: Calcula la varianza de las medianas muestrales obtenidas en las 1000 repeticiones.

Mostrar las Varianzas:

- Al imprimir `var_media` y `var_mediana`, podemos comparar las varianzas de ambos estimadores.

Para una comparación más detallada de la eficiencia entre la media muestral y la mediana muestral, realizaremos una simulación de bootstrap.

El **bootstrap** es un método estadístico que permite estimar la distribución de un estadístico (como la media o la varianza) a partir de muestras repetidas generadas de la misma muestra original. En lugar de depender de suposiciones teóricas sobre la distribución de los datos, el bootstrap genera muchas réplicas simuladas mediante el muestreo con reemplazo de la muestra original. Esto nos permite construir una estimación empírica de la distribución del estadístico que estamos evaluando.

En este caso, el bootstrap nos permitirá estimar la distribución de las diferencias de varianza entre la media muestral y la mediana muestral, lo que facilitará una comparación más detallada de la eficiencia entre ambos estimadores.

```r
# Configuración inicial
set.seed(123)
n_muestra <- 100  # Tamaño de los grupos de muestreo
reps_bootstrap <- 1000  # Número de repeticiones para el bootstrap

# Inicializar un vector para almacenar las diferencias de varianzas
diferencias_varianzas <- numeric(reps_bootstrap)

# Realizar el muestreo con reemplazo sobre medias_muestrales y medianas_muestrales
for (i in 1:reps_bootstrap) {
  # Generar índices de muestreo con reemplazamiento
  indices <- sample(1:reps, size = n_muestra, replace = TRUE)

  # Muestrear las medias y las medianas usando los mismos índices
  muestra_media <- medias_muestrales[indices]
  muestra_mediana <- medianas_muestrales[indices]

  # Calcular las varianzas de las muestras
  var_media <- var(muestra_media)
  var_mediana <- var(muestra_mediana)
```

```r
  # Almacenar la diferencia de varianzas entre media y mediana
  diferencias_varianzas[i] <- var_media - var_mediana
}

# Crear un dataframe para ggplot
datos_varianzas <- data.frame(diferencias_varianzas)
```

Configuración Inicial:

- `set.seed(123)`: Fija la semilla de los números aleatorios para asegurar la reproducibilidad de la simulación.
- `n_muestra <- 100`: Define el tamaño de cada grupo de muestreo en el bootstrap.
- `reps_bootstrap <- 1000`: Establece el número de repeticiones para el proceso de bootstrap.

Inicialización del Vector de Diferencias de Varianzas:

- `diferencias_varianzas <- numeric(reps_bootstrap)`: Crea un vector vacío para almacenar las diferencias de varianza en cada repetición del bootstrap.

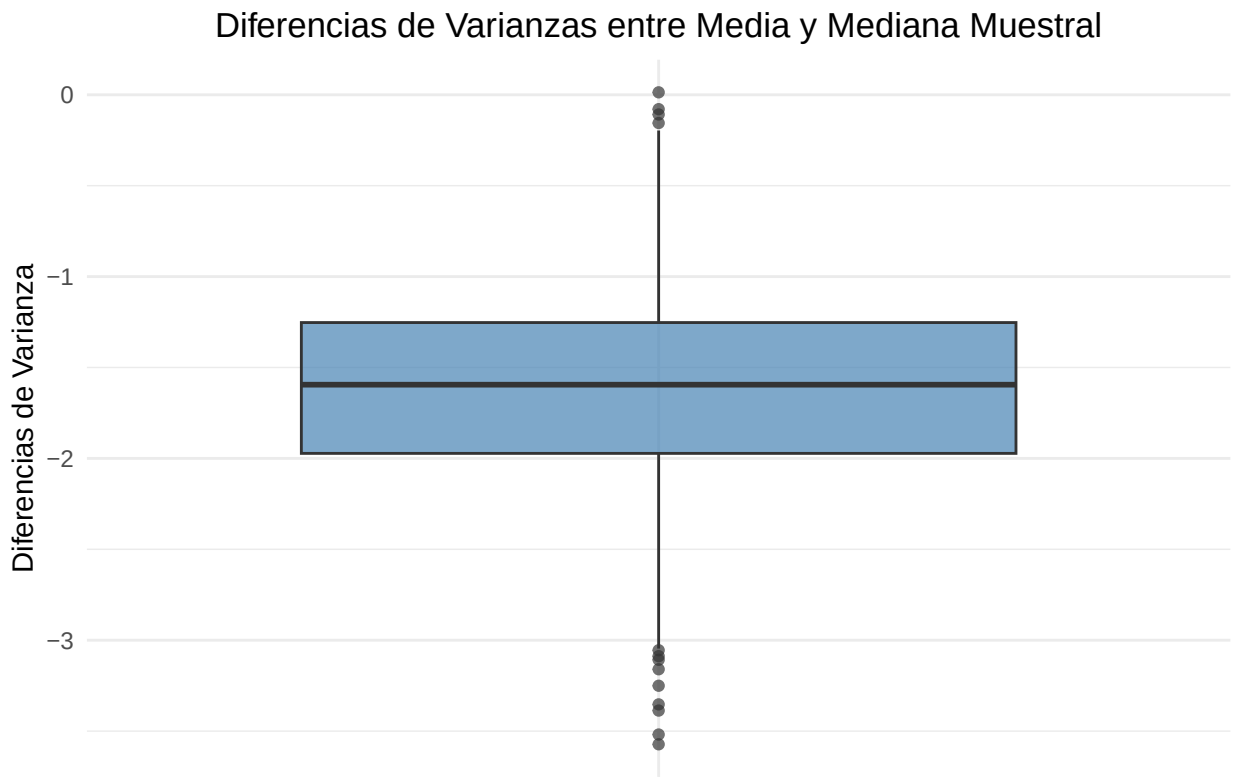
Proceso de Bootstrap:

- **Muestreo con Reemplazo:**
  - `indices <- sample(1:reps, size = n_muestra, replace = TRUE)`: Genera 100 índices aleatorios con reemplazo para muestrear de las `medias_muestrales` y `medianas_muestrales`.
- **Muestreo de Medias y Medianas:**
  - `muestra_media <- medias_muestrales[indices]`: Selecciona las medias muestrales correspondientes a los índices seleccionados.
  - `muestra_mediana <- medianas_muestrales[indices]`: Selecciona las medianas muestrales correspondientes a los mismos índices, asegurando que ambas muestras sean comparables.
- **Cálculo de Varianzas:**
  - `var_media <- var(muestra_media)`: Calcula la varianza de la muestra de medias muestrales.
  - `var_mediana <- var(muestra_mediana)`: Calcula la varianza de la muestra de medianas muestrales.
- **Almacenamiento de Diferencias de Varianzas:**
  - `diferencias_varianzas[i] <- var_media - var_mediana`: Calcula y almacena la diferencia de varianzas entre la media y la mediana para cada repetición.

Creación del Dataframe para la Visualización:

- `datos_varianzas <- data.frame(diferencias_varianzas)`: Convierte el vector de diferencias de varianzas en un dataframe para facilitar su manejo con `ggplot2`.

```r
# Graficar las diferencias de varianzas entre media y mediana con un boxplot
ggplot(datos_varianzas, aes(x = "", y = diferencias_varianzas)) +
  geom_boxplot(fill = "steelblue", alpha = 0.7) +
  labs(title = "Diferencias de Varianzas entre Media y Mediana Muestral",
       y = "Diferencias de Varianza",
       x = "") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



El gráfico resultante muestra la **distribución de las diferencias de varianza** entre la media y la mediana muestral a lo largo de las 1000 repeticiones. Se destacan los siguientes puntos:

- **Boxplot**: Representa la dispersión de estas diferencias. Cada punto en el gráfico corresponde a una diferencia de varianza calculada a partir de un grupo de muestras. El boxplot también muestra la mediana y los valores atípicos, proporcionando una visión clara de la dispersión.
- **Mediana Negativa**: Sugiere que, en la mayoría de las simulaciones, la varianza de la media muestral es menor que la de la mediana, lo que confirma la mayor eficiencia de la media.
- **Outliers**: Señalan casos donde la diferencia de varianza fue significativamente mayor o menor de lo esperado, posiblemente debido a la variabilidad en muestras pequeñas.

En resumen, el gráfico demuestra que la varianza de la media muestral tiende a ser menor que la de la mediana, confirmando la **eficiencia** de la media como estimador.

3.3.3. Consistencia

Un estimador es **consistente** si, a medida que el tamaño de la muestra aumenta, el estimador converge en probabilidad al verdadero valor del parámetro poblacional.

$$\hat{\theta}_n \xrightarrow{P} \theta \quad \text{cuando} \quad n \to \infty$$

Esto implica que, con muestras suficientemente grandes, el estimador proporcionará una estimación arbitrariamente cercana al parámetro que se desea estimar

Ejemplo: La **media muestral** $\bar{x}$ es un estimador consistente de la **media poblacional** μ. Para demostrar la **consistencia** de $\bar{x}$, realizaremos una simulación en R donde generaremos múltiples muestras de una distribución normal $N(50, 10)$ con diferentes tamaños de muestra, y observaremos cómo la media muestral se aproxima cada vez más a la media poblacional μ conforme aumenta el tamaño de la muestra n. Específicamente, utilizaremos tamaños de muestra que van desde 10 hasta 1000 en incrementos de 10, y realizaremos 1000 repeticiones para cada tamaño de muestra.

```r
# Simulación para demostrar la consistencia de la media muestral
set.seed(123)          # Fijar la semilla para reproducibilidad
mu <- 50               # Media poblacional
sigma <- 10            # Desviación estándar poblacional

# Definir una secuencia de tamaños de muestra
n_values <- seq(10, 10000, by = 10)

# Calcular la media muestral para cada tamaño de muestra utilizando sapply
media_convergente <- sapply(n_values, function(n) {
  # Generar una muestra de tamaño n de una distribución normal N(50, 10)
  muestra <- rnorm(n, mean = mu, sd = sigma)
  # Calcular la media muestral
  mean(muestra)
})

# Crear un data frame para graficar
datos_convergencia <- data.frame(
  Tamaño_Muestra = n_values,
  Media_Muestral = media_convergente
)
```

Configuración Inicial:

- `set.seed(123)`: Fija la semilla de los números aleatorios para asegurar la reproducibilidad de la simulación.
- `mu <- 50`: Establece la media poblacional.
- `sigma <- 10`: Define la desviación estándar poblacional.
- `n_values <- seq(10, 10000, by = 10)`: Crea una secuencia de tamaños de muestra desde 10 hasta 10000, incrementando de 10 en 10.

Proceso de Simulación:

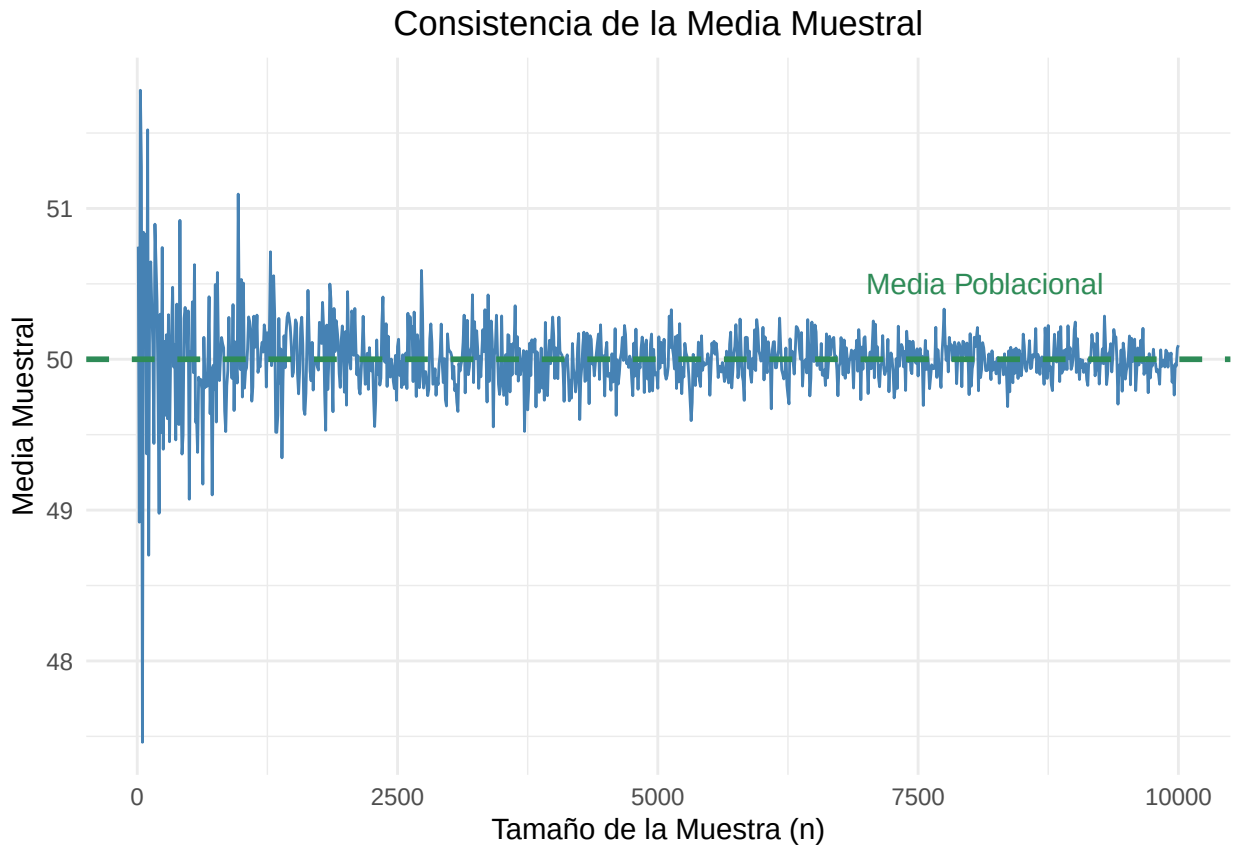
1. Generación de Muestras y Cálculo de Medias Muestrales:

- `media_convergente <- sapply(n_values, function(n) { ... })`: Para cada tamaño de muestra n en `n_values`, se genera una muestra aleatoria de tamaño n de una distribución normal $N(\mu, \sigma)$ y se calcula su media muestral.

2. Almacenamiento de Resultados:

- Se crea un data frame `datos_convergencia` que contiene los tamaños de muestra y las correspondientes medias muestrales.

```r
# Graficar la convergencia de la media muestral hacia la media poblacional

ggplot(datos_convergencia, aes(x = Tamaño_Muestra, y = Media_Muestral)) +
  geom_line(color = "steelblue") +
  geom_hline(aes(yintercept = mu), color = "seagreen", linetype = "dashed", size = 1) +
  labs(title = "Consistencia de la Media Muestral",
       x = "Tamaño de la Muestra (n)",
       y = "Media Muestral") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  annotate("text", x = max(n_values) * 0.7, y = mu + 0.5,
           label = "Media Poblacional", color = "seagreen", hjust = 0)
```



El gráfico resultante muestra cómo la media muestral se comporta a medida que aumenta el tamaño de la muestra. Observamos lo siguiente:

- **Línea Azul (Media Muestral):** Representa la media muestral obtenida para cada tamaño de muestra n. A medida que n incrementa, las fluctuaciones en las medias muestrales tienden a reducirse, acercándose más al valor verdadero de la media poblacional $\mu = 50$.
- **Línea Verde Discontinua (Media Poblacional):** Indica el valor verdadero de la media poblacional. Es el punto al cual esperamos que la media muestral converja conforme n aumenta.

La simulación demuestra que, al incrementar el tamaño de la muestra, la **media muestral** $\bar{x}$ se aproxima cada vez más a la **media poblacional** μ. Esto confirma la propiedad de **consistencia** de la media muestral, ya que $\bar{x}$ converge en probabilidad a μ cuando $n \to \infty$.

Además, esta propiedad asegura que, con muestras suficientemente grandes, podemos confiar en que la media muestral proporcionará una estimación precisa del parámetro poblacional, reduciendo la incertidumbre asociada a la variabilidad muestral.

3.3.4. Suficiencia

Un estimador suficiente es aquel que garantiza que dicho estimador utiliza toda la información relevante contenida en la muestra respecto al parámetro que se está estimando. Un estimador suficiente no pierde información útil para la estimación del parámetro poblacional. Formalmente, un estimador $T(X)$ es **suficiente** para un parámetro θ si la distribución condicional de la muestra X dado el estimador $T(X)$ no depende de θ. En otras palabras, $T(X)$ captura toda la información relevante de la muestra sobre θ.

Además, según el **Teorema de Factorización de Fisher**, un estadístico $T(X)$ es suficiente para θ si y solo si la **función de densidad** $f(x|\theta)$ puede factorizarse de la siguiente manera:

$$f(x|\theta) = g(T(x), \theta) \cdot h(x)$$

donde $g(T(x), \theta)$ es una función que depende de la muestra x únicamente a través de $T(x)$ y del parámetro θ, mientras que $h(x)$ es una función que depende de la muestra x pero no del parámetro θ.

Propiedades de Estimadores Suficientes

- **Propiedad Teórica**: La suficiencia se demuestra mediante la estructura de la función de densidad o masa de probabilidad y su factorización, no a través de cálculos empíricos.
- **Limitaciones**: Un estimador suficiente no es necesariamente el mejor en términos de otras propiedades como la varianza mínima. Sin embargo, dentro de la clase de estimadores suficientes, es posible identificar estimadores que son óptimos bajo ciertos criterios.
- **Reducción de Datos**: La suficiencia permite reducir los datos de la muestra a un estadístico suficiente sin pérdida de información relevante para la estimación del parámetro. Esto simplifica el análisis al trabajar con un resumen conciso de los datos originales.

Relación con Otras Propiedades de los Estimadores

- **Insesgadez**: Un estimador suficiente puede ser insesgado, pero la suficiencia no implica necesariamente insesgadez.
- **Eficiencia**: Dentro de la clase de estimadores suficientes, se puede buscar el estimador que minimiza la varianza (estimador eficiente).
- **Consistencia**: La suficiencia no garantiza la consistencia, aunque muchos estimadores consistentes también son suficientes.

Importancia de la Suficiencia

- **Optimización de Datos**: Permite reducir la muestra a un estadístico suficiente sin perder información relevante, facilitando el análisis.
- **Base para Inferencia**: Muchos métodos de inferencia estadística, como los intervalos de confianza y las pruebas de hipótesis, se basan en estimadores suficientes para garantizar la validez de los resultados.

- **Teorema de Basu**: Un teorema importante que establece que cualquier estadístico independiente de un estimador suficiente también es independiente de cualquier función de ese estimador. Esto ayuda a separar la información relevante de la irrelevante en la muestra.

La propiedad de **suficiencia** asegura que un estimador captura toda la información necesaria de la muestra respecto al parámetro de interés, evitando la pérdida de información y optimizando el proceso de estimación. Comprender y identificar estimadores suficientes es esencial para desarrollar inferencias estadísticas sólidas y eficientes.

> **Ejemplo**: La **media muestral** $\bar{x}$ es un estimador suficiente de la **media poblacional** μ en una Distribución Normal $N(\mu, \sigma)$ con σ^2 conocida.

Utilizamos el **Teorema de Factorización de Fisher** para demostrar que $\bar{x}$ es un estadístico suficiente para μ.

Función de Densidad de la Distribución Normal:

La función de densidad para una distribución normal $N(\mu, \sigma^2)$ es:

$$f(x|\mu) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

Factorización de la Función de Densidad:

Podemos factorizar la función de densidad de la siguiente manera:

$$f(x|\mu) = \left(\frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{n(\bar{x} - \mu)^2}{2\sigma^2}\right)\right) \cdot \exp\left(-\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{2\sigma^2}\right)$$

Donde:

- $g(\bar{x}, \mu) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{n(\bar{x}-\mu)^2}{2\sigma^2}\right)$ depende de los datos solo a través de $\bar{x}$ y del parámetro μ.
- $h(x) = \exp\left(-\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{2\sigma^2}\right)$ depende de los datos x pero no de μ.

Según el **Teorema de Factorización de Fisher**, esto implica que $\bar{x}$ es un estadístico suficiente para μ, ya que la función de densidad se factoriza en una función que depende de los datos solo a través de $\bar{x}$ y otra que no depende de μ.

3.4. Conclusión

Este laboratorio nos ha permitido comprender y aplicar de manera práctica los conceptos fundamentales de **muestreo** y **estimación estadística**, prestando especial atención a las propiedades de los estimadores y su suficiencia.

A través de ejemplos y simulaciones realizadas en R, abordamos diversos tipos de muestreo, como el **muestreo aleatorio simple**, **estratificado**, **por conglomerados** y **sistemático**, destacando cuándo es conveniente aplicar cada método y cómo influyen en la representatividad de las muestras obtenidas.

Adicionalmente, exploramos las propiedades de los estimadores más comunes:

- **Insesgadez**: Se verificó que la media muestral $\bar{x}$ es un estimador insesgado de la media poblacional μ. En promedio, la media muestral converge al valor real del parámetro, lo cual se demostró mediante simulaciones.

- **Eficiencia**: Comparando la media y la mediana muestral, observamos que la media tiene una menor varianza, lo que la convierte en un estimador más eficiente en nuestra muestra simulada.
- **Consistencia**: Simulaciones con tamaños de muestra crecientes mostraron que la media muestral se aproxima cada vez más a la media poblacional, confirmando su consistencia a medida que el tamaño de la muestra aumenta.
- **Suficiencia**: Aplicando el **Teorema de Factorización de Fisher**, demostramos que la media muestral es un estimador suficiente para la media de una población normal. Este concepto asegura que $\bar{x}$ utiliza toda la información relevante de la muestra sin perder ningún detalle necesario para estimar con precisión el parámetro poblacional.

Implicaciones Prácticas:

- **Selección de métodos de muestreo**: La elección correcta del método de muestreo, dependiendo de la estructura de la población, es crucial para obtener muestras representativas y realizar inferencias precisas.
- **Uso de estimadores eficientes y suficientes**: La eficiencia y suficiencia de los estimadores permiten realizar inferencias estadísticamente óptimas, reduciendo errores y maximizando la información obtenida de los datos muestrales.
- **Decisiones informadas basadas en datos**: Contar con estimadores consistentes, insesgados y eficientes es esencial para garantizar la confiabilidad de los análisis estadísticos, lo cual es fundamental en la toma de decisiones en diversos campos, desde la investigación científica hasta la industria.

En resumen, este laboratorio ha integrado teoría y práctica para mejorar nuestra comprensión del **muestreo** y las **estimaciones**. Al dominar tanto los conceptos teóricos como las simulaciones en R, los estudiantes están mejor preparados para aplicar estas técnicas en contextos reales, asegurando la precisión y validez de los resultados obtenidos en sus análisis estadísticos.

Laboratorio de Estimación y Contrastes de Hipótesis

Carmen Lancho - Isaac Martín - Víctor Aceña

Grado en Ciencia e Ingeniería de Datos - Inferencia Estadística - Curso 2024/2025

Índice

Objetivo

El objetivo principal de este laboratorio es aplicar técnicas de **estimación estadística** y realizar **contrastes de hipótesis** utilizando R. Al finalizar este laboratorio, deberías ser capaz de:

1. Calcular estimaciones puntuales e intervalos de confianza para medias y proporciones.
2. Realizar contrastes de hipótesis para medias y proporciones.
3. Interpretar los resultados estadísticos obtenidos.
4. Visualizar datos y resultados para apoyar tus conclusiones.

1. Introducción

En esta sección, introduciremos los conceptos básicos de **estimación** y **contraste de hipótesis**.

- **Estimación Estadística**: Proceso de inferir el valor de un parámetro poblacional a partir de datos muestrales.

- **Estimación Puntual**: Un único valor estimado del parámetro.
- **Estimación por Intervalo**: Un rango de valores dentro del cual se espera que se encuentre el parámetro con cierto nivel de confianza.

- **Contraste de Hipótesis**: Procedimiento para decidir si los datos muestrales proporcionan suficiente evidencia para rechazar una hipótesis sobre un parámetro poblacional.

2. Estimación Puntual y por Intervalos

Existen diferentes métodos de estimación que podemos utilizar dependiendo de la situación y la naturaleza de los datos. A continuación, se describen brevemente los métodos más comunes:

2.1. Estimación de la Media

El objetivo es estimar la media poblacional μ utilizando la media muestral $\bar{X}$. La **media muestral** es un estimador puntual de la media poblacional y se calcula como:

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

2.1.0.1. Intervalo de Confianza para la Media

- **Cuando σ es conocido**:

$$IC = \bar{X} \pm Z_{\frac{\alpha}{2}} \left(\frac{\sigma}{\sqrt{n}} \right)$$

- **Cuando σ es desconocido** (más común):

$$IC = \bar{X} \pm t_{\frac{\alpha}{2}, n-1} \left(\frac{s}{\sqrt{n}} \right)$$

Donde:

- $\bar{X}$: Media muestral.
- s: Desviación estándar muestral.
- n: Tamaño de la muestra.
- $Z_{\frac{\alpha}{2}}$: Valor crítico de la distribución normal.
- $t_{\frac{\alpha}{2}, n-1}$: Valor crítico de la distribución t de Student con $n-1$ grados de libertad.

Estimación de la Media de la Estatura de una Población

Ejemplo: Supongamos que deseamos conocer la estatura media de una población de adultos, pero no podemos medir a cada individuo en la población debido a limitaciones de tiempo y recursos. En su lugar, tomamos una muestra aleatoria de 30 personas y registramos sus estaturas en centímetros. Utilizaremos esta muestra para hacer una estimación puntual de la media poblacional y construir un intervalo de confianza que nos permita inferir, con cierto nivel de confianza, el rango en el que se encuentra la verdadera media de estatura en la población.

```r
set.seed(123)
estaturas <- rnorm(30, mean = 170, sd = 10)
```

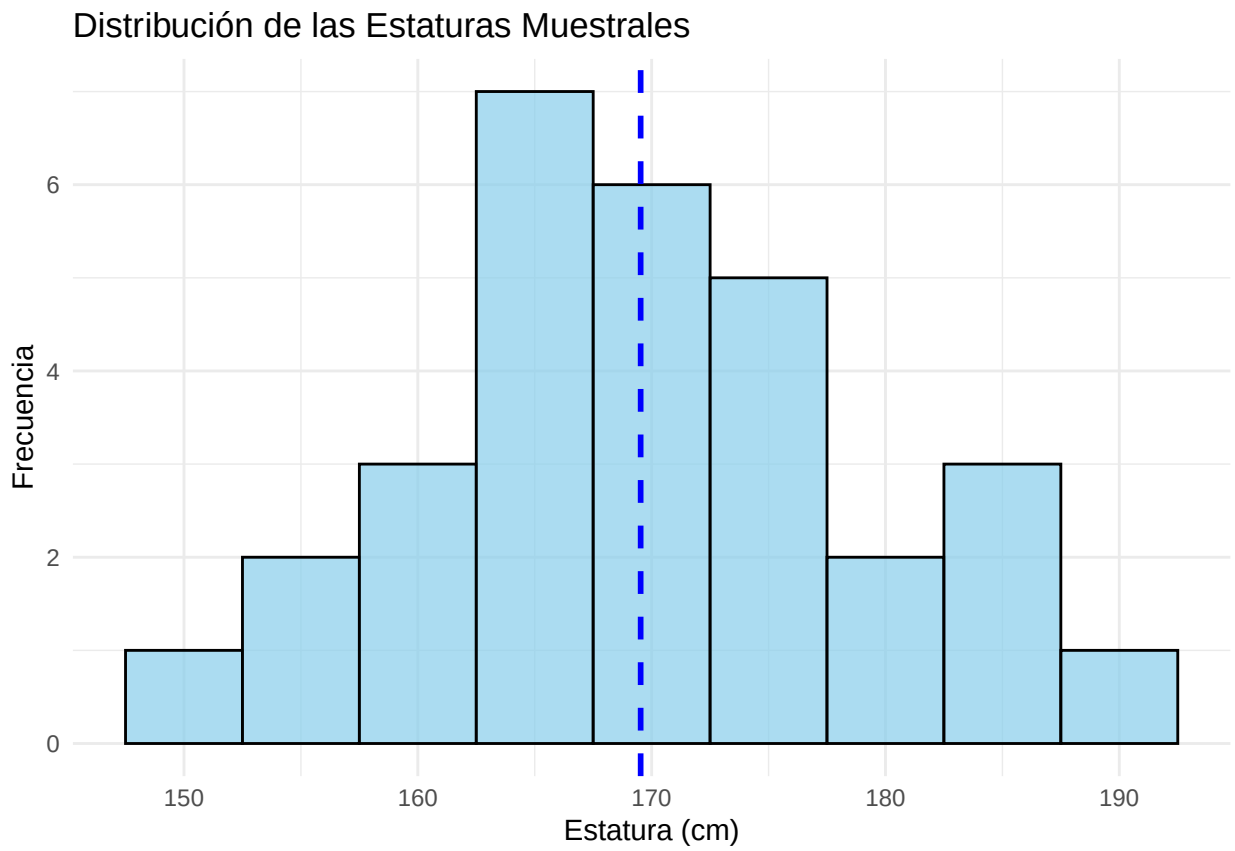
Primero, calculamos la media muestral como estimación puntual de la media poblacional. Este valor representa la mejor estimación del promedio de la población, basada en nuestra muestra.

```r
media_muestral <- mean(estaturas)
cat("La media muestral es:", round(media_muestral, 2), "cm")
```

```
## La media muestral es: 169.53 cm
```

A continuación, visualizamos la distribución de las estaturas muestrales para observar su forma y el estimador de la media que acabamos de calcular.

```r
ggplot(data.frame(estaturas), aes(x = estaturas)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black", alpha = 0.7) +
  geom_vline(aes(xintercept = media_muestral), color = "blue", linetype = "dashed", size = 1) +
  labs(title = "Distribución de las Estaturas Muestrales",
       x = "Estatura (cm)",
       y = "Frecuencia") +
  theme_minimal()
```



Como no conocemos la desviación estándar de la población, utilizamos la distribución t de Student para construir un intervalo de confianza. Este intervalo nos proporciona un rango de valores dentro del cual esperamos que se ubique la media poblacional, con un nivel de confianza específico.

```r
# Definimos una función llamada `calcular_ic` que calcula el intervalo de confianza
# para un conjunto de datos (en este caso, las estaturas) y un nivel de confianza específico.
# Por defecto, usa un nivel de confianza del 95% (alpha = 0.05).
calcular_ic <- function(datos, alpha = 0.05) {

  # Tamaño de la muestra: `n` representa el número de datos en la muestra.
  # Esto es importante para calcular el error estándar.
  n <- length(datos)

  # Calculamos la media muestral de `datos`, que representa la estimación puntual
  # de la media poblacional.
  media <- mean(datos)

  # Calculamos la desviación estándar muestral `s`, que mide la dispersión de los datos.
  # Como no conocemos la desviación estándar poblacional, utilizamos la de la muestra.
  s <- sd(datos)

  # Calculamos el valor crítico de la t de Student `t_critico`.
  # Este valor depende del nivel de confianza (`alpha`) y de los grados de libertad (`df = n - 1`).
  # Nos indica cuántas desviaciones estándar debemos alejarnos de la media para cubrir
  # el intervalo de confianza deseado.
  t_critico <- qt(1 - alpha/2, df = n - 1)

  # Calculamos el error estándar de la media `error_estandar`, que nos dice
  # cuánto varía la media muestral en relación con la media poblacional.
  # Se calcula dividiendo la desviación estándar muestral `s` entre la raíz cuadrada del tamaño de la m
  error_estandar <- s / sqrt(n)

  # Calculamos el margen de error `ME`, que determina cuánto sumamos y restamos
  # a la media para crear el intervalo de confianza. Es el producto del valor crítico
  # y el error estándar.
  ME <- t_critico * error_estandar

  # Retornamos el intervalo de confianza, que va desde `media - ME` hasta `media + ME`.
  # Esto nos da un rango de valores dentro del cual, con un 95% de confianza, se encuentra la media pob
  c(media - ME, media + ME)
}

# Llamamos a la función `calcular_ic` con nuestros datos de estatura y obtenemos el intervalo de confia
IC <- calcular_ic(estaturas)

# Mostramos el resultado en pantalla, redondeando cada límite del intervalo a dos decimales.
cat("Intervalo de confianza al 95%: [", round(IC[1], 2), ",", round(IC[2], 2), "] cm")
```

```
## Intervalo de confianza al 95%: [ 165.87 , 173.19 ] cm
```

Interpretación

Con un 95 % de confianza, estimamos que la estatura media de la población se encuentra dentro del intervalo calculado. Esto significa que, si repitiéramos este procedimiento muchas veces con diferentes muestras, aproximadamente el 95 % de las veces el intervalo de confianza incluiría la verdadera media poblacional.

2.2. Cálculo del Tamaño de Muestra Necesario para una Estimación

Cuando queremos obtener un intervalo de confianza con un margen de error específico, es importante determinar de antemano cuántas observaciones necesitamos. Esto nos ayuda a optimizar la recolección de datos y asegurar que nuestra muestra sea representativa y precisa.

2.2.1. Teoría

Para calcular el tamaño de muestra necesario, utilizamos la fórmula:

$$n = \left(\frac{Z_{\alpha/2} \cdot \sigma}{ME} \right)^2$$

Donde:

- `n`: Tamaño de la muestra necesario.
- `Z_{\alpha/2}`: Valor crítico de la distribución normal para el nivel de confianza deseado.
- `\sigma`: Desviación estándar poblacional (o una estimación).
- `ME`: Margen de error que deseamos obtener en el intervalo de confianza.

> **Ejemplo**: Supongamos que queremos estimar la estatura media de una población de adultos con un intervalo de confianza del 95 % y un margen de error de 2 cm. Sabemos, por estudios previos, que la desviación estándar de estatura en esta población es de aproximadamente 10 cm.

Podemos definir una función para calcular el tamaño de muestra necesario dado un margen de error específico:

```
calcular_tamaño_muestra <- function(sigma, ME, alpha = 0.05) {
  # Valor crítico Z para el nivel de confianza deseado
  Z_critico <- qnorm(1 - alpha / 2)

  # Cálculo del tamaño de muestra necesario
  n <- (Z_critico * sigma / ME)^2

  # Redondeamos al número entero superior
  ceiling(n)
}

# Parámetros de ejemplo
sigma <- 10     # Desviación estándar de la población
ME <- 2         # Margen de error deseado

# Llamada a la función
tamaño_muestra <- calcular_tamaño_muestra(sigma, ME)
cat("Tamaño de muestra necesario:", tamaño_muestra)
```

```
## Tamaño de muestra necesario: 97
```

Interpretación

Con estos parámetros, necesitamos al menos 97 observaciones para asegurarnos de que nuestro intervalo de confianza tenga un margen de error de 2 cm con un nivel de confianza del 95 %. Esta información es útil para planificar la recolección de datos de forma efectiva y asegurar la precisión de nuestra estimación.

2.3. Estimación de una Proporción

A veces, estamos interesados en conocer la proporción de individuos en una población que posee cierta característica. En este caso, utilizaremos la **proporción muestral** como una estimación de la **proporción poblacional**.

La proporción muestral, denotada como $\hat{p}$, es un estimador puntual de la proporción poblacional p. Para calcular un **intervalo de confianza** para la proporción, utilizamos la siguiente fórmula:

$$IC = \hat{p} \pm Z_{\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

Donde:

- $\hat{p}$: Proporción muestral, calculada como el número de éxitos (o individuos con la característica de interés) dividido por el tamaño de la muestra n.
- $Z_{\frac{\alpha}{2}}$: Valor crítico de la distribución normal para el nivel de confianza deseado.
- n: Tamaño de la muestra.

> **Ejemplo**: Supongamos que queremos estimar la proporción de personas en una ciudad que practica regularmente algún deporte. Tomamos una muestra de 200 personas y encontramos que 120 de ellas practican deporte con regularidad. Calcularemos la proporción muestral y construimos un intervalo de confianza del 95 % para esta proporción.

Empezaremos definiendo los datos de nuestro ejemplo y luego calcularemos la proporción muestral.

```r
# Definimos los datos iniciales del ejemplo
datos_deporte <- data.frame(
  muestra = 1:200,
  practica_deporte = c(rep(1, 120), rep(0, 80)) # 1 si practica deporte, 0 si no
)

# Cálculo de la proporción muestral
p_muestral <- mean(datos_deporte$practica_deporte)
cat("La proporción muestral es:", round(p_muestral, 3))
```

```
## La proporción muestral es: 0.6
```

A continuación, utilizaremos la proporción muestral calculada para construir un intervalo de confianza para la proporción poblacional, lo que nos permitirá estimar el rango en el que se encuentra la verdadera proporción de personas que practican deporte regularmente en la población.

2.3.1. Intervalo de Confianza para una Proporción

El intervalo de confianza para una proporción se calcula utilizando la fórmula:

$$IC = \hat{p} \pm Z_{\frac{\alpha}{2}} \cdot \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

Donde:

- $\hat{p}$: Proporción muestral.
- $Z_{\frac{\alpha}{2}}$: Valor crítico de la distribución normal para el nivel de confianza deseado.
- n: Tamaño de la muestra.

En este ejemplo, utilizaremos un nivel de confianza del 95 %, por lo que el valor crítico $Z_{\frac{\alpha}{2}}$ será aproximadamente 1.96.

```
# Parámetros iniciales
p_muestral <- mean(datos_deporte$practica_deporte)  # Proporción muestral de personas que practican dep
n <- length(datos_deporte$practica_deporte)         # Tamaño de la muestra
alpha <- 0.05                                       # Nivel de significancia para un IC del 95%
z_critico <- qnorm(1 - alpha / 2)                   # Valor crítico Z para un IC del 95%

# Cálculo del error estándar
error_estandar <- sqrt((p_muestral * (1 - p_muestral)) / n)

# Cálculo del margen de error
ME <- z_critico * error_estandar

# Cálculo del intervalo de confianza
IC_inferior <- p_muestral - ME
IC_superior <- p_muestral + ME

# Mostrar el intervalo de confianza
cat("Intervalo de confianza al 95% para la proporción poblacional: [", round(IC_inferior, 3), ",", roun
```

```
## Intervalo de confianza al 95% para la proporción poblacional: [ 0.532 , 0.668 ]
```

Interpretación

Con un 95 % de confianza, estimamos que la proporción de personas en la población que practican deporte regularmente se encuentra dentro del intervalo calculado. Esto significa que, si tomáramos muchas muestras y construyéramos un intervalo de confianza para cada una, aproximadamente el 95 % de esos intervalos contendrían la verdadera proporción poblacional.

Este enfoque es particularmente útil en estudios donde queremos hacer inferencias sobre un comportamiento o característica en una población, como en este caso la práctica regular de deporte. A partir de este intervalo, podemos tener una idea más precisa y confiable del porcentaje de personas que practican deporte en la ciudad sin haber necesitado encuestar a toda la población.

2.4. Diferencia de Proporciones

En este ejemplo, queremos comparar la proporción de personas que practican deporte regularmente en dos ciudades: **Ciudad A** y **Ciudad B**. Tomamos muestras independientes en cada ciudad y calculamos la proporción de personas que practican deporte. Utilizaremos estas proporciones muestrales para construir un intervalo de confianza que nos permita inferir si existe una diferencia significativa entre las dos proporciones poblacionales.

Para construir un intervalo de confianza para la diferencia entre dos proporciones poblacionales, utilizamos la siguiente fórmula:

$$IC = (p_1 - p_2) \pm Z_{\frac{\alpha}{2}} \cdot \sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}$$

donde:

- p_1 y p_2: Proporciones muestrales de personas que practican deporte en Ciudad A y Ciudad B, respectivamente.
- n_1 y n_2: Tamaños de las muestras en Ciudad A y Ciudad B, respectivamente.
- $Z_{\frac{\alpha}{2}}$: Valor crítico de la distribución normal para el nivel de confianza deseado (por ejemplo, 1.96 para un IC del 95 %).

Esta fórmula permite estimar el rango en el que se encuentra la diferencia real entre las dos proporciones poblacionales con un nivel de confianza especificado.

> **Ejemplo**: Supongamos que en una muestra de 300 personas en **Ciudad A**, 180 personas practican deporte regularmente, mientras que en una muestra de 250 personas en **Ciudad B**, 130 personas practican deporte regularmente. Queremos calcular el intervalo de confianza al 95 % para la diferencia de proporciones entre las dos ciudades.

```
# Datos del ejemplo
n1 <- 300          # Tamaño de la muestra en Ciudad A
exitos1 <- 180     # Número de personas que practican deporte en Ciudad A
n2 <- 250          # Tamaño de la muestra en Ciudad B
exitos2 <- 130     # Número de personas que practican deporte en Ciudad B

# Cálculo de las proporciones muestrales
p1_muestral <- exitos1 / n1
p2_muestral <- exitos2 / n2

# Mostrar las proporciones muestrales
cat("Proporción muestral en Ciudad A:", round(p1_muestral, 3), "\n")
```

```
## Proporción muestral en Ciudad A: 0.6
```

```
cat("Proporción muestral en Ciudad B:", round(p2_muestral, 3), "\n")
```

```
## Proporción muestral en Ciudad B: 0.52
```

2.4.1. Intervalo de Confianza para la Diferencia de Proporciones

En este ejemplo, queremos estimar la diferencia en la proporción de personas que practican deporte regularmente entre dos ciudades, Ciudad A y Ciudad B. Con los datos obtenidos de cada muestra, calcularemos un intervalo de confianza para esta diferencia de proporciones con un nivel de confianza del 95 %.

Para ello, utilizaremos la siguiente fórmula de intervalo de confianza para la diferencia de proporciones:

$$IC = (p_1 - p_2) \pm Z_{\frac{\alpha}{2}} \cdot \sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}$$

Donde:

- p_1 y p_2 son las proporciones muestrales de Ciudad A y Ciudad B, respectivamente.
- n_1 y n_2 son los tamaños de las muestras en Ciudad A y Ciudad B.
- $Z_{\frac{\alpha}{2}}$ es el valor crítico de la distribución normal para el nivel de confianza deseado.

```r
# Nivel de significancia y valor crítico Z para un IC del 95%
alpha <- 0.05
z_critico <- qnorm(1 - alpha / 2)

# Cálculo del error estándar para la diferencia de proporciones
error_estandar <- sqrt((p1_muestral * (1 - p1_muestral) / n1) + (p2_muestral * (1 - p2_muestral) / n2))

# Cálculo del margen de error
ME <- z_critico * error_estandar

# Cálculo del intervalo de confianza para la diferencia de proporciones
IC_inferior <- (p1_muestral - p2_muestral) - ME
IC_superior <- (p1_muestral - p2_muestral) + ME

# Mostrar el intervalo de confianza
cat("Intervalo de confianza al 95% para la diferencia de proporciones: [",
    round(IC_inferior, 3), ",", round(IC_superior, 3), "]")
```

```
## Intervalo de confianza al 95% para la diferencia de proporciones: [ -0.003 , 0.163 ]
```

Interpretación

El intervalo de confianza calculado para la diferencia de proporciones nos indica que, con un 95 % de confianza, la diferencia en la proporción de personas que practican deporte regularmente entre Ciudad A y Ciudad B se encuentra dentro del rango obtenido. Si el intervalo incluye el cero, esto sugiere que no hay una diferencia significativa entre las proporciones de las dos ciudades en cuanto a la práctica deportiva. Si el intervalo no contiene el cero, podemos inferir que existe una diferencia significativa en la proporción de personas que practican deporte regularmente entre ambas ciudades.

Este análisis es útil para comparar la prevalencia de una característica o comportamiento entre dos grupos o poblaciones y es ampliamente aplicable en estudios de encuestas, marketing, salud pública, y otros campos de investigación.

2.5. Contraste de Igualdad de Medias

Cuando queremos comparar las medias de dos grupos diferentes para ver si existe una diferencia significativa entre ellos, utilizamos un **contraste de igualdad de medias**. Este contraste de hipótesis nos permite evaluar si la diferencia observada entre las medias muestrales es suficientemente grande como para inferir que existe una diferencia real en las medias poblacionales.

En un contraste de igualdad de medias, las hipótesis se plantean de la siguiente manera:

- **Hipótesis Nula (H_0)**: No hay diferencia en las medias poblacionales, es decir, $\mu_1 = \mu_2$.
- **Hipótesis Alternativa (H_1)**: Existe una diferencia en las medias poblacionales, es decir, $\mu_1 \neq \mu_2$.

Para realizar este contraste, utilizamos el **test t de Student** para dos muestras independientes. Los resultados del contraste incluyen:

- **p-valor**: Nos indica si la diferencia es estadísticamente significativa. Un p-valor bajo (típicamente menor que 0.05) sugiere que podemos rechazar la hipótesis nula y concluir que existe una diferencia significativa entre las medias.

- **Intervalo de Confianza (IC)**: Nos permite observar si el intervalo incluye el valor cero. Si el intervalo de confianza de la diferencia de medias no contiene el cero, esto refuerza la conclusión de que hay una diferencia significativa entre las medias de los grupos.

Un **intervalo de confianza que no incluye el cero** y un **p-valor menor que 0.05** sugieren una diferencia significativa entre las medias de los dos grupos.

> **Ejemplo**: Supongamos que estamos interesados en comparar los niveles de pH de agua de dos fuentes diferentes (orígenes) en un laboratorio. Para este propósito, utilizamos una muestra de 100 mediciones de pH para cada origen y aplicamos un contraste de igualdad de medias para determinar si existe una diferencia significativa en el nivel de pH entre ambas fuentes.

En primer lugar, cargamos y preparamos los datos:

```
# Cargamos y preparamos los datos del laboratorio
set.seed(1)
library(tidyverse)
library(readxl)

lab <- read_excel("../../data/lab.xlsx") |>
  mutate(fecha = as.Date(fecha)) |>
  slice_sample(n = 100)
```

```
## Error: ‘path‘ does not exist: ’../../data/lab.xlsx’
```

A continuación, observamos un resumen descriptivo de las mediciones de pH por grupo de origen. Esta descripción nos permite tener una visión general de los datos y prepararnos para la interpretación de los resultados en el contraste de hipótesis.

```
# Generar un resumen estadístico de la variable ph según el grupo de origen
with(lab, stby(ph, origen, descr, stats = "common", transpose = TRUE))
```

```
## Error in eval(expr, envir, enclos): objeto ’lab’ no encontrado
```

Para confirmar si existe una diferencia significativa en los niveles de pH entre los grupos, realizaremos un **contraste de hipótesis para la igualdad de medias**. Este contraste es útil cuando queremos determinar si la media de una variable continua difiere significativamente entre dos grupos.

2.5.1. Hipótesis del contraste

En este caso, formulamos las siguientes hipótesis:

- **Hipótesis nula** (H_0): La media de pH es igual en ambos grupos de origen, es decir, $\mu_1 = \mu_2$.
- **Hipótesis alternativa** (H_a): La media de pH es diferente entre los grupos de origen, es decir, $\mu_1 \neq \mu_2$.

2.5.2. Procedimiento del contraste

Utilizaremos el **t-test** de Student para comparar las medias entre ambos grupos. Este método es adecuado cuando la variable sigue una distribución normal y las varianzas de ambos grupos son iguales (supuesto de homogeneidad de varianzas).

Implementación

En este análisis, utilizamos la función `t.test()` de R para realizar un **contraste de hipótesis de igualdad de medias** entre los grupos "Norte" y "Sur". A continuación, desglosamos su propósito y uso:

- `ph ~ origen` define la variable dependiente (`ph`, que representa el nivel de pH) y la variable independiente (`origen`, que indica el grupo, "Norte" o "Sur"). Esto indica a R que compare las medias de `ph` entre los dos grupos definidos en `origen`.
- `data = lab` indica que los datos de pH y de origen se encuentran en el conjunto de datos `lab`.
- `var.equal = TRUE` asume que las varianzas de ambos grupos son iguales, lo cual permite realizar un test t de Student para muestras independientes, adecuado cuando los grupos tienen varianzas similares.

El uso de `t.test()` es fundamental para **determinar si existe una diferencia significativa entre las medias** de dos grupos en una variable cuantitativa (en este caso, el nivel de pH entre dos orígenes de agua).

Realizaremos este contraste con un nivel de significancia del 5 % ($\alpha = 0{,}05$), y evaluaremos el **p-valor** y el **intervalo de confianza** para interpretar los resultados.

```
# Realizamos el t-test para comparar las medias de pH entre los grupos de origen
t.test(ph ~ origen, data = lab, var.equal = TRUE)
```

```
## Error in eval(m$data, parent.frame()): objeto 'lab' no encontrado
```

Interpretación de los Resultados

Tras ejecutar el test de hipótesis, interpretamos los resultados obtenidos:

1. **Valor p**: El valor p reportado es **3.862e-06**, que es mucho menor al nivel de significancia comúnmente utilizado (0.05). Esto sugiere que existe suficiente evidencia para rechazar la hipótesis nula de que las medias de los grupos "Norte" y "Sur" son iguales. En otras palabras, hay una diferencia estadísticamente significativa en los niveles de pH entre los dos grupos de origen.
2. **Intervalo de Confianza (IC)**: El intervalo de confianza al 95 % para la diferencia en las medias de pH entre el grupo "Norte" y el grupo "Sur" es **[0.0341, 0.0805]**. Dado que este intervalo no incluye el valor cero, esto respalda la conclusión de que existe una diferencia significativa entre las medias de los dos grupos. La diferencia de medias es positiva, lo que indica que, en promedio, el pH del grupo "Norte" es ligeramente mayor que el del grupo "Sur".
3. **Estimaciones de las Medias**: La media de pH en el grupo "Norte" es **6.6614** y en el grupo "Sur" es **6.6041**. Esto indica que, aunque la diferencia es pequeña, el pH promedio en el grupo "Norte" es mayor.

Con un 95 % de confianza, podemos afirmar que existe una diferencia significativa en el nivel de pH entre el grupo "Norte" y el grupo "Sur". Esta diferencia, aunque pequeña, es estadísticamente significativa, y el intervalo de confianza nos indica que la verdadera diferencia en las medias de los niveles de pH entre los dos grupos probablemente se encuentra entre 0.034 y 0.080 unidades de pH.

Para visualizar esta diferencia y entender mejor la distribución de los datos entre los grupos de origen, generaremos un gráfico comparativo utilizando la librería `ggstatsplot`, que mostrará las medias de cada grupo y sus respectivas distribuciones.

```r
# Gráfico de comparación de medias entre los grupos de origen
lab |> ggbetweenstats(x = origen, y = ph)
```

```
## Error in eval(expr, envir, enclos): objeto 'lab' no encontrado
```

Este gráfico visualiza las diferencias entre los grupos en términos de sus distribuciones y medias, permitiendo una interpretación más intuitiva de la posible diferencia en pH entre los orígenes.

Interpretación del Gráfico

El gráfico muestra visualmente si hay una superposición significativa entre las distribuciones de los grupos, junto con las medias y los intervalos de confianza. Si las distribuciones no se superponen o hay poca superposición, y las medias son claramente distintas, esto refuerza la conclusión del test de hipótesis de que existe una diferencia significativa entre los grupos en el pH medido.

2.6. Potencia Estadística

La **potencia estadística** es la probabilidad de rechazar correctamente la hipótesis nula (H_0) cuando es falsa. Matemáticamente, se define como:

$$\text{Potencia} = P(\text{Rechazar } H_0 \mid H_0 \text{ es falsa})$$

Una potencia alta (generalmente considerada como 0.8 o más) significa que el test es sensible para detectar diferencias reales. El cálculo de la potencia es fundamental en el diseño experimental, ya que ayuda a determinar el tamaño de muestra necesario y a minimizar los errores de Tipo II (β), que ocurren cuando se falla en rechazar la hipótesis nula cuando en realidad es falsa.

2.6.1. Factores que Afectan la Potencia del Test

La potencia de un test depende de varios factores, que se pueden expresar de la siguiente manera:

1. **Nivel de significancia (α)**:
   - Es la probabilidad de cometer un error de Tipo I, que se define como la probabilidad de rechazar la hipótesis nula cuando es verdadera. En términos matemáticos:

$$\alpha = P(\text{Rechazar } H_0 \mid H_0 \text{ es verdadera})$$

   - Un nivel de significancia más alto (como 0.10 en lugar de 0.05) incrementa la potencia, ya que aumenta la probabilidad de rechazar H_0. Sin embargo, esto también aumenta el riesgo de cometer un error Tipo I.
2. **Tamaño del efecto (d)**:
   - Representa la magnitud de la diferencia que queremos detectar, calculada como la diferencia entre las medias poblacionales en unidades de desviación estándar:

$$d = \frac{\mu_1 - \mu_2}{\sigma}$$

   - Un tamaño del efecto más grande resulta en una mayor potencia, ya que es más fácil detectar diferencias significativas cuando las diferencias entre los grupos son grandes.
3. **Tamaño de la muestra (n)**:

- El tamaño de la muestra se refiere al número de observaciones en el estudio. A medida que aumentamos n, la potencia del test también aumenta. Esto se debe a que muestras más grandes reducen la variabilidad de la estimación de la media y proporcionan una estimación más precisa de la diferencia entre las poblaciones:

$$\text{Error estándar} = \frac{\sigma}{\sqrt{n}}$$

- Con un error estándar menor, es más probable que una diferencia observada sea considerada significativa.

4. **Variabilidad de los datos (σ)**:
   - Representa la desviación estándar de la población. Una mayor variabilidad en los datos reduce la potencia, ya que las diferencias son más difíciles de detectar en datos más dispersos. La relación se puede expresar como:

$$\text{Potencia} \propto \frac{1}{\sigma}$$

   - Esto significa que si aumentamos la variabilidad de los datos (un σ más grande), la potencia disminuye, haciendo más difícil detectar diferencias significativas.

La relación entre estos factores y la potencia estadística es esencial para diseñar experimentos eficaces. Es crucial encontrar un equilibrio entre el tamaño de la muestra, el nivel de significancia, el tamaño del efecto y la variabilidad de los datos para asegurar que el estudio tenga una potencia adecuada para detectar diferencias significativas, minimizando así el riesgo de cometer errores Tipo II.

Ejemplo:Vamos a utilizar el paquete `pwr` para calcular la potencia de un test t para dos muestras independientes. Supongamos que queremos detectar una diferencia en la media de dos grupos con un tamaño del efecto medio.

1. Definir los Parámetros del Estudio

- **Nivel de significancia (α)**: 0.05
- **Tamaño del efecto (d)**: Utilizamos el estadístico de Cohen para el tamaño del efecto.
  - Pequeño: 0.2
  - Medio: 0.5
  - Grande: 0.8
- **Tamaño de la muestra (n)**: Número de observaciones en cada grupo.

2. Calcular la Potencia

Supongamos que tenemos 50 observaciones en cada grupo y queremos detectar un tamaño del efecto medio.

```r
# Parámetros
n <- 50            # Tamaño de muestra por grupo
d <- 0.5           # Tamaño del efecto medio
alpha <- 0.05      # Nivel de significancia

# Cálculo de la potencia
resultado_potencia <- pwr.t.test(n = n, d = d, sig.level = alpha,
                                 type = "two.sample", alternative = "two.sided")

# Mostrar los resultados
print(resultado_potencia)
```

```
##
##      Two-sample t test power calculation
##
##               n = 50
##               d = 0.5
##       sig.level = 0.05
##           power = 0.6968934
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Interpretación

La potencia del test realizado es de aproximadamente **69.7 %**, lo que indica una **probabilidad moderada** de detectar una diferencia real si esta existe. Para aumentar la confiabilidad de los resultados y reducir la posibilidad de no detectar diferencias significativas, se recomienda aumentar el tamaño de la muestra.

> **Ejemplo**: Ahora, supongamos que queremos determinar el tamaño de muestra necesario para alcanzar una potencia del 80 %.

```r
# Parámetros
d <- 0.5           # Tamaño del efecto medio
power <- 0.8       # Potencia deseada
alpha <- 0.05      # Nivel de significancia

# Cálculo del tamaño de muestra
resultado_tamano <- pwr.t.test(power = power, d = d, sig.level = alpha,
                               type = "two.sample", alternative = "two.sided")

# Mostrar los resultados
print(resultado_tamano)
```

```
##
##      Two-sample t test power calculation
##
##               n = 63.76561
##               d = 0.5
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Interpretación

- **n = 63.76561**: Se requieren aproximadamente **64 observaciones por grupo** para alcanzar una potencia del **80 %** con un tamaño de efecto medio.
- **d = 0.5**: Tamaño del efecto medio.
- **sig.level = 0.05**: Nivel de significancia del **5 %**.
- **power = 0.8**: Potencia del **80 %**.
- **alternative = two.sided**: Contraste bilateral.

En este caso, determinar que se necesitan aproximadamente **64 observaciones por grupo** garantiza que el test t de Student tendrá una potencia del **80 %** para detectar una diferencia media con un nivel de significancia del **5 %**. Esto asegura que el estudio es suficientemente sensible para identificar diferencias reales, minimizando la probabilidad de errores Tipo II y aumentando la confiabilidad de los resultados obtenidos.

2.7. Medidas del Tamaño del Efecto

Además de la significancia estadística proporcionada por el p-valor, es importante evaluar la **magnitud del efecto** para entender la relevancia práctica de los resultados. El **d de Cohen** es una medida del tamaño del efecto que expresa la diferencia entre dos medias en términos de desviaciones estándar, permitiendo comparar la magnitud de la diferencia independientemente de las unidades de medida.

2.7.1. Cálculo del d de Cohen

El **d de Cohen** se calcula utilizando la siguiente fórmula:

$$d = \frac{\bar{X}_1 - \bar{X}_2}{s_{\text{pooled}}}$$

Donde:

- $\bar{X}_1$ y $\bar{X}_2$ son las medias muestrales de los grupos 1 y 2, respectivamente.
- s_{pooled} es la desviación estándar combinada (desviación estándar agrupada), calculada como:

$$s_{\text{pooled}} = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

Donde:

- s_1^2 y s_2^2 son las varianzas muestrales de los grupos 1 y 2, respectivamente.
- n_1 y n_2 son los tamaños de muestra de los grupos 1 y 2.

Esta fórmula combina las desviaciones estándar de ambos grupos, asumiendo que las varianzas son homogéneas.

2.7.2. Interpretación del d de Cohen

Los valores del **d de Cohen** se interpretan generalmente de la siguiente manera:

- **Pequeño**: $d = 0{,}2$
- **Medio**: $d = 0{,}5$
- **Grande**: $d = 0{,}8$ o mayor

Un valor mayor de d indica una diferencia más grande entre los grupos en términos de desviaciones estándar.

2.7.3. Implementación en R

Para calcular el **d de Cohen** en R, utilizamos el paquete `effsize`.

- `library(effsize)`:
  - **Descripción**: Carga el paquete `effsize` para poder utilizar la función `cohen.d()`.
- `cohen.d(ph ~ origen, data = lab, pooled = TRUE, hedges.correction = FALSE)`: Calcula el **d de Cohen** para comparar las medias de `ph` entre los niveles de `origen` en el conjunto de datos `lab`.
  - **Argumentos**:
    - `ph ~ origen`: Fórmula que indica que queremos comparar la variable `ph` entre los niveles de `origen`.
    - `data = lab`: Especifica el conjunto de datos que contiene las variables.
    - `pooled = TRUE`: Utiliza la desviación estándar combinada de ambos grupos, asumiendo homogeneidad de varianzas.
    - `hedges.correction = FALSE`: Indica que no se aplicará la corrección de Hedges; si se desea una estimación más precisa en muestras pequeñas, se puede establecer como `TRUE`.

Ejemplo: Vamos a calcular el tamaño del efecto utilizando el **d de Cohen** para comparar los niveles de pH entre los grupos "Norte" y "Sur" en nuestro conjunto de datos `lab`. Esto nos permitirá cuantificar la magnitud de la diferencia observada entre los dos grupos y entender su relevancia práctica.

```
# Calcular el d de Cohen para la variable ph entre los grupos de origen
cohen_d <- cohen.d(ph ~ origen, data = lab, pooled = TRUE, hedges.correction = FALSE)
```

```
## Error in eval(expr, envir, enclos): objeto 'lab' no encontrado
```

```
print(cohen_d)
```

```
## Error in eval(expr, envir, enclos): objeto 'cohen_d' no encontrado
```

Interpretación

- **Valor del d de Cohen (`d estimate`):** El valor obtenido es **1.078852**, lo que se considera un **tamaño del efecto grande** según las convenciones establecidas. Este valor indica que la diferencia entre las medias de los grupos "Norte" y "Sur" es de aproximadamente **1.08 desviaciones estándar**, lo que representa una diferencia sustancial entre los grupos.
- **Intervalo de Confianza del 95 %:** Con un 95 % de confianza, el verdadero tamaño del efecto en la población se encuentra entre **0.616** y **1.542**. Dado que todo el intervalo está por encima de 0, esto refuerza la conclusión de que existe una diferencia significativa y positiva entre los grupos.

Relevancia Práctica:

- Un **tamaño del efecto grande** sugiere que la diferencia observada no solo es estadísticamente significativa, sino también relevante en la práctica.
- Esto implica que el origen del agua ("Norte" vs. "Sur") tiene un impacto considerable en los niveles de pH medidos.

Conclusión:

El **d de Cohen** de **1.078852** indica una diferencia grande y significativa entre los grupos "Norte" y "Sur" en términos de pH. Este hallazgo tiene implicaciones prácticas importantes, ya que sugiere que el origen del agua afecta significativamente su nivel de pH. Los resultados respaldan la necesidad de considerar el origen del agua en análisis futuros y posibles intervenciones o regulaciones relacionadas con la calidad del agua.

3. Conclusiones

A lo largo de este laboratorio, hemos aplicado técnicas de **estimación estadística** y **contraste de hipótesis** para analizar diferencias en **medias** y **proporciones**. En particular, se destacan los siguientes puntos:

- **Visualización de Datos**:
  - Utilizamos herramientas gráficas para **visualizar las distribuciones** de los datos y las diferencias entre grupos, lo que facilita una interpretación más intuitiva y efectiva de los resultados.
  - Las visualizaciones permiten identificar patrones, tendencias y posibles anomalías en los datos que podrían influir en el análisis estadístico.
- **Significancia Estadística**:
  - Aprendimos a determinar si las diferencias observadas entre grupos son estadísticamente significativas utilizando **p-valores** e **intervalos de confianza**.
  - La interpretación correcta de estos resultados nos permite evaluar la evidencia contra la hipótesis nula de manera objetiva.
- **Verificación de Supuestos Estadísticos**:
  - Exploramos la importancia de verificar los **supuestos necesarios** para la aplicación de pruebas paramétricas, como la **normalidad de los datos** y la **homogeneidad de varianzas**.
  - La correcta verificación de estos supuestos asegura la validez y confiabilidad de los análisis realizados.
- **Cálculo de la Potencia Estadística**:
  - Entendimos cómo la **potencia estadística** influye en la capacidad de un test para detectar diferencias reales cuando existen.
  - Aprendimos a calcular el **tamaño de muestra necesario** para alcanzar una potencia deseada, optimizando así el diseño experimental y reduciendo la probabilidad de errores de Tipo II.
- **Tamaño del Efecto**:
  - Incorporamos el cálculo y la interpretación del **d de Cohen**, una medida que cuantifica la magnitud de la diferencia entre grupos en términos de desviaciones estándar.
  - Esta medida complementa la significancia estadística al proporcionar información sobre la relevancia práctica de los resultados obtenidos.

Nota: Algunos de los ejemplos de este laboratorio utilizan datos del libro *Introducción al software estadístico R* de López Cano E. (2024), disponible en https://www.lcano.com/b/iser/_book/.