# Tarea 09: Máxima Verosimilitud

Rodrigo Alan García Pérez

2024-11-06

## Estimador de Maximam Verosimilitud

### Ejercicio 1: Crear función de verosimilitud con el ejemplo de número de errores

```
crear_log_verosim <- function(n, n_err){
  # n es tamaño de muestra
  # n_err el número de errores detectados (datos)
  n_corr <- n - n_err
  log_verosim <- function(p){
    n_err * log(p) + n_corr * log(1-p)
  }
}
```

**Pregunta: ¿Qué devuelve esta función?**

```
crear_log_verosim(100, 50)
```

La función crear_log_verosim devuelve otra función llamada log_verosim, la cual toma un valor p (probabilidad del error) y calcula el logaritmo de verosimilitud. Esta función evalúa que tan probable es observar el número de errores y aciertos dado el valor de p.

**Utiliza esta función para construir la función de log verosimilitud cuando los datos observados son n = 500, y el número de errores observados es 121**

```
# Crear la función de log-verosimilitud
log_verosim_func <- crear_log_verosim(n = 500, n_err = 121)
```

**Pregunta: de qué parámetros depende esta ultima función**

La ultima función llamada log_verosim_func depende únicamente del parámetro p, que representa la probabilidad de error.

**¿Dónde quedaron los datos observados?**

Los datos observados de n = 500 y n_err = 121 fueron pasados a la primer funcion crear_log_verosim. Ahora son parte de esta funcion como variables internas y estaran fijas al llamar a log_verosim_func y lo unico variable sera ahora el valor p.

**Usa max verosimilitud para estimar el porcentaje de errores en la tabla de datos de donde se sacó la muestra usa la función optimize**

```
# Maximizar la log-verosimilitud para encontrar la estimación de p
resultado <- optimize(log_verosim_func, c(0, 1), maximum = TRUE)

resultado$maximum  # Este es el valor estimado de p que maximiza la verosimilitud
```

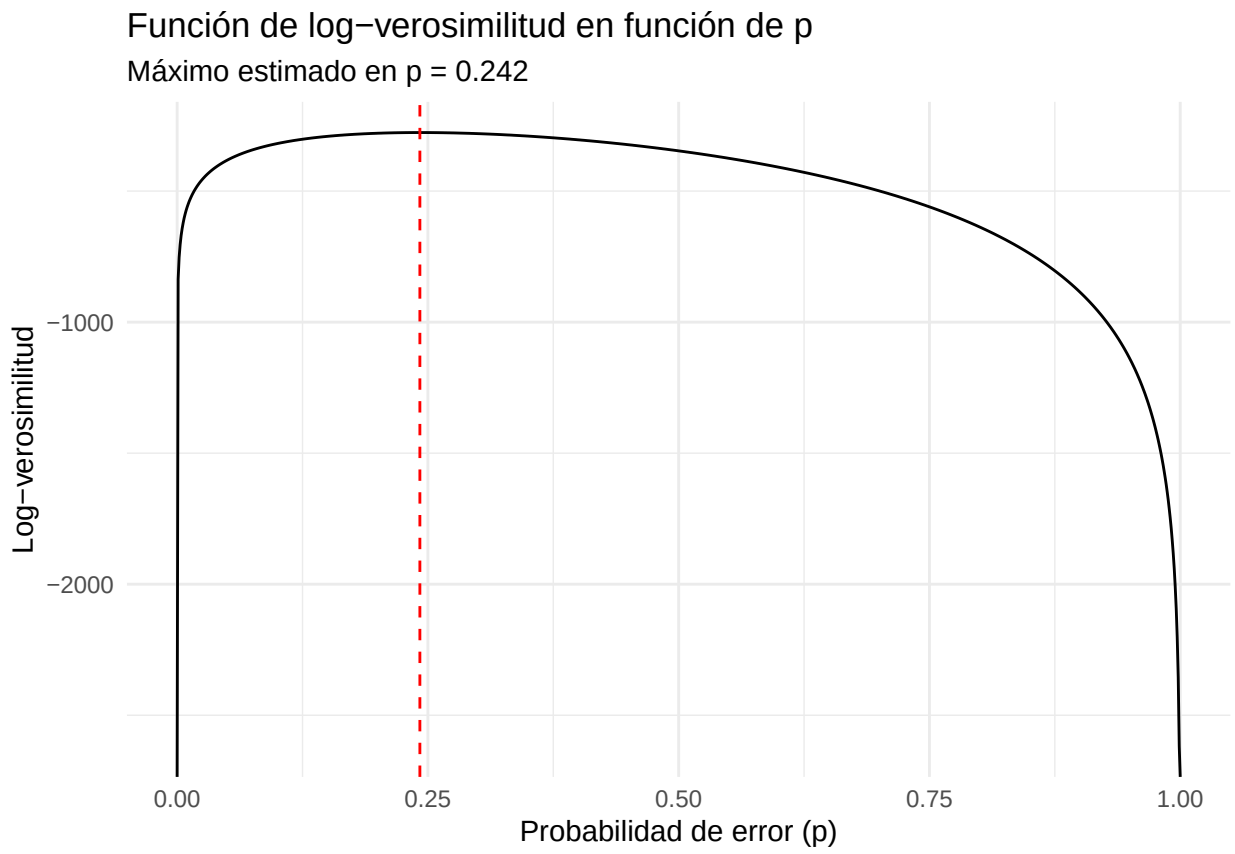
```
## [1] 0.2420091
```

**Grafica la funcion de verosimilitud para p entre cero y uno**

```
library(ggplot2)
library(tibble)
library(dplyr)

# Generar una tabla con valores de p y la verosimilitud correspondiente
grafica_tbl <- tibble(p = seq(0, 1, 0.001)) %>%
  mutate(log_verosimilitud = sapply(p, log_verosim_func))

# Graficar la función de log-verosimilitud
ggplot(grafica_tbl, aes(x = p, y = log_verosimilitud)) +
  geom_line() +
  geom_vline(xintercept = resultado$maximum, color = "red", linetype = "dashed") +
  labs(
    title = "Función de log-verosimilitud en función de p",
    subtitle = paste("Máximo estimado en p =", round(resultado$maximum, 4)),
    x = "Probabilidad de error (p)",
    y = "Log-verosimilitud"
  ) +
  theme_minimal()
```



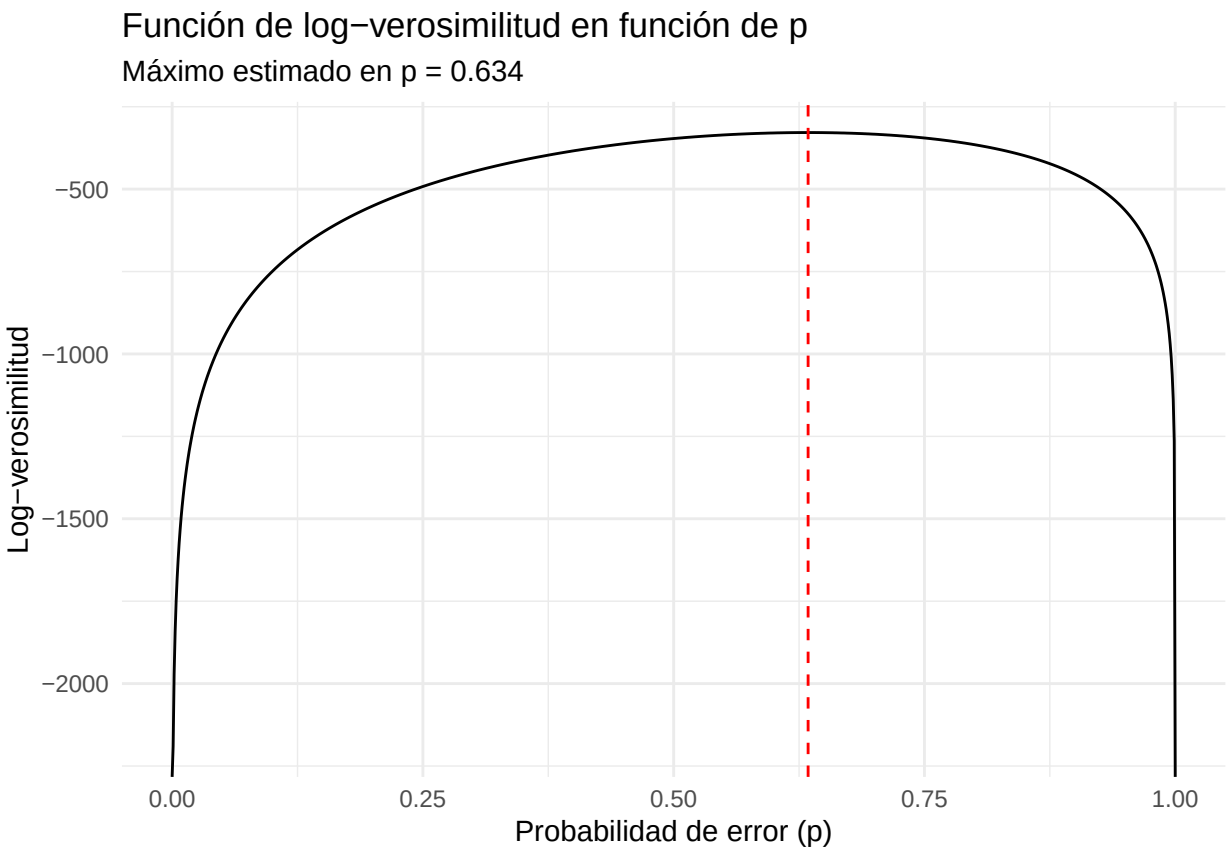

**Repite el ejercicio anterior si observas 317 errores**

```r
# Crear la función de log-verosimilitud específica para los datos observados con 317 errores
log_verosim_func <- crear_log_verosim(n = 500, n_err = 317)

# Generar una tabla con valores de p y la verosimilitud correspondiente
grafica_tbl <- tibble(p = seq(0, 1, 0.001)) %>%
  mutate(log_verosimilitud = sapply(p, log_verosim_func))

# Encontrar el valor de p que maximiza la verosimilitud
resultado <- optimize(log_verosim_func, c(0, 1), maximum = TRUE)
p_max <- resultado$maximum

# Graficar la función de log-verosimilitud con la línea en el valor que maximiza
ggplot(grafica_tbl, aes(x = p, y = log_verosimilitud)) +
  geom_line() +
  geom_vline(xintercept = p_max, color = "red", linetype = "dashed") +
  labs(
    title = "Función de log-verosimilitud en función de p",
    subtitle = paste("Máximo estimado en p =", round(p_max, 4)),
    x = "Probabilidad de error (p)",
    y = "Log-verosimilitud"
  ) +
  theme_minimal()
```




## Ejercicio 2: normal

**Estimación de parámetros de una distribución normal**

1. Considera una muestra de variables aleatorias Gaussianas. Escribe la verosimilitud para una muestra de tamaño n y después escribe la función de log-verosimilitud.

2. Generamos una mu y una sigma al azar para que no sepamos cuáles son

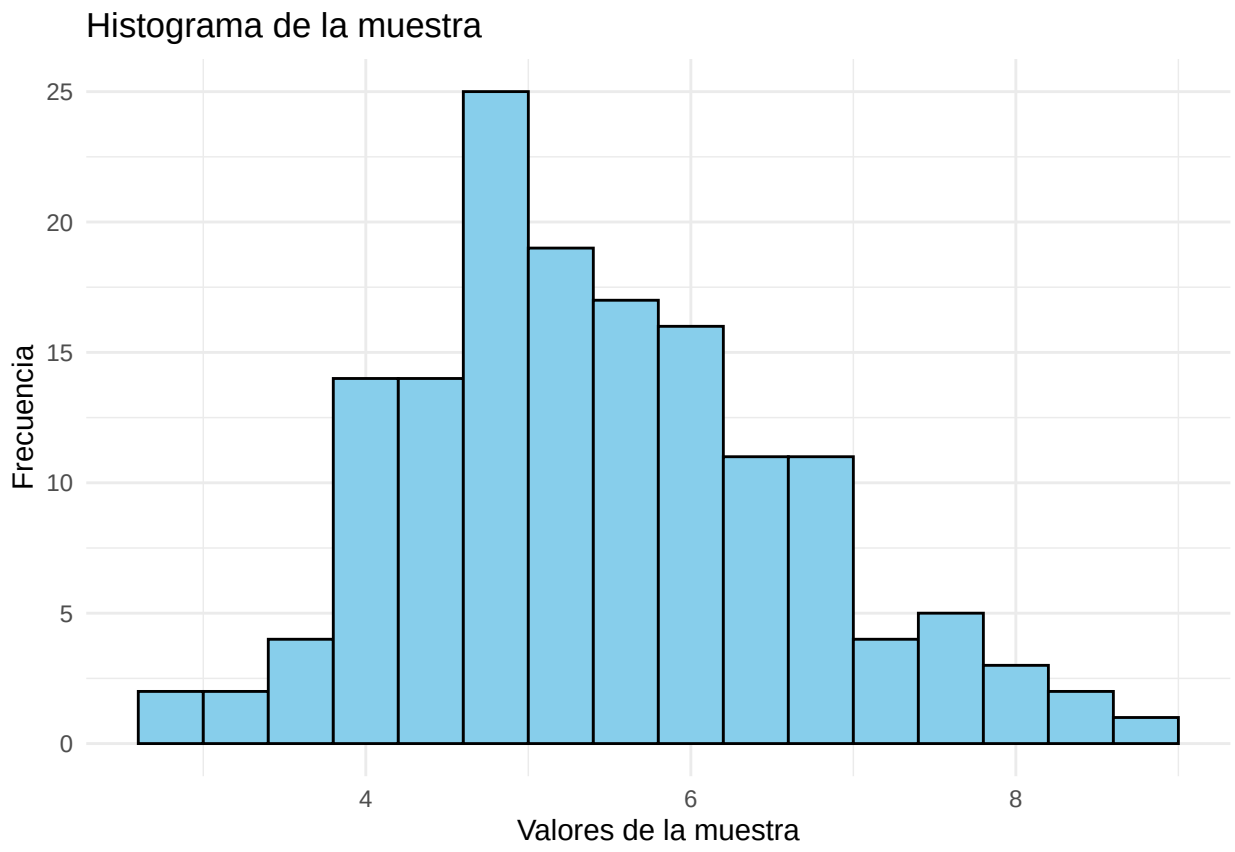
```
#fijamos semilla
set.seed(1234)
m <- runif(1, 5, 10) # media entre 5 y 10
desv_est <- runif(1, 0, 2) # desviación estándar entre 0 y 2

#simulamos una muestra con la que trabajaremos.
x <- rnorm(150, mean = m, sd = desv_est)
```

## 2. Checa tus datos usando un histograma

```
# Graficar el histograma de la muestra x
ggplot(data = data.frame(x = x), aes(x = x)) +
  geom_histogram(binwidth = 0.4, color = "black", fill = "skyblue") +
  labs(
```

```
    title = "Histograma de la muestra",
    x = "Valores de la muestra",
    y = "Frecuencia"
  ) +
  theme_minimal()
```




**Primero estima a ojo por dónde creen que esté la media y la desviación estándar**

Los datos parecen centrados en 5, lo cual seria un valor aproximado para la media y parecen concentrados alrededor de 4 y 6 lo cual indica una desviación estándar de 1.

**3. Estima la media y la desviación estándar usando estimador plug-in**

```
# Estimación de la media y la desviación estándar usando el estimador plug-in
media_est <- mean(x)
de_est <- sd(x)
```

**4. Escribe una función que regrese la función de verosimilitud dado una muestra (revisa las notas en la sección de más de un parámetro)**

```r
# Definimos la función generadora de la log-verosimilitud para una muestra
crear_log_p <- function(x) {
  log_p <- function(pars) {
    media <- pars[1]
    desv_est <- pars[2]

    n <- length(x)
    z <- (x - media) / desv_est
    log_verosim <- -n * log(desv_est) - 0.5 * mean(z^2)
    return(log_verosim)
  }

  return(log_p)
}
```

**5. Usa la función del paso anterior para crear una función de verosimilitud que dependerá de mu y sigma**

```r
# Creamos la función de log-verosimilitud específica para nuestra muestra
log_p <- crear_log_p(x)
```

**6.Optimiza y compara tus estimaciones con las del paso 3**

```r
# Realizamos la optimización para encontrar los valores que maximizan la verosimilitud
# Usamos como valores iniciales media = 0 y sigma = 0.5, y aplicamos optimización numérica
res <- optim(c(0, 0.5), log_p, control = list(fnscale = -1, maxit = 1000), method = "Nelder-Mead")
```

```
## Warning in log(desv_est): NaNs produced

## Warning in log(desv_est): NaNs produced
```

```r
# Mostrar los resultados
res$convergence  # 0 indica que la optimización fue exitosa
```

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

```r
# Resultados de la optimización
mu_optimizado <- res$par[1]
sigma_optimizado <- res$par[2]

# Estimaciones muestrales directas (paso 3)
media_est <- mean(x)
de_est <- sd(x)

# Crear un tibble para comparar los resultados
comparacion <- tibble(
  parametro = c("media", "sigma"),
  estimador_muestral = c(media_est, de_est),
```

```
  estimador_maxima_verosimilitud = c(mu_optimizado, sigma_optimizado)
)

# Mostrar la comparación
comparacion
```

```
## # A tibble: 2 x 3
##   parametro estimador_muestral estimador_maxima_verosimilitud
##   <chr>                  <dbl>                           <dbl>
## 1 media                   5.45                            5.45
## 2 sigma                   1.19                            0.0968
```

# Boostrap paramétrico

## Ejercicio 2: Ahora calculamos error estándar para el ejercicio de número de errores usando bootstrap paramétrico

### Paso 1 y 2: argumenta que el paso 1 y 2 ya lo hicimos arriba

### Pregunta: ¿cuál es tu estimador puntual?

El estimador puntual de la probabilidad de error es el valor de p que maximiza la función de log-verosimilitud, es decir, p_max. Este valor representa la estimación de la probabilidad de error en la población y se basa en los datos de la muestra (317 errores en 500 observaciones).

### Paso 3: la siguiente función simula bajo nuestro modelo teórico

```
sim_modelo <- function(p, n){
  # rellena: qué es p y n
  muestra <- rbinom(n, 1, p)
  n_errores <- sum(muestra)
  n_errores
}
```

### investiga la función rbinom, ¿qué hace?

La función rbinom en R genera números aleatorios a partir de una distribución binomial, donde el primer parametro es el numero de observaciones a generar, el segundo es el número de ensayos en cada observación y el tercero es la probabilidad de éxito en cada ensayo.

### calcula una simulación de estos datos, con alguna p y n fijas

```
# Definimos los parámetros para la simulación
p <- 0.3    # Probabilidad de error
n <- 500    # Tamaño de la muestra
```

```r
# Calculamos una simulación del número de errores
n_errores_simulados <- sim_modelo(p, n)

# Mostramos el resultado
n_errores_simulados
```

```
## [1] 162
```

**Paso 4: Enchufa tu estimador puntual de max verosimilitud y simula 3 mil observaciones**

```r
# Estimador puntual de máxima verosimilitud para p (calculado previamente)
p_max <- resultado$maximum  # Este es el valor de p que maximiza la verosimilitud

# Tamaño de cada muestra
n <- 500

# Número de simulaciones
n_simulaciones <- 3000

# Simulamos 3,000 observaciones del número de errores usando el estimador puntual p_max
set.seed(123)
simulaciones <- replicate(n_simulaciones, sim_modelo(p_max, n))

# Mostrar las primeras 10 simulaciones como ejemplo
head(simulaciones, 10)
```

```
##  [1] 322 315 327 324 315 302 323 313 330 319
```

**Paso 5: calcula el estimador de max verosimilitud para cada simulación**

‘## tu codi

encontrar_mv <- function(n, n_err){ # log_ver <- ######### # res <- optimize( , c(0, 1) , maximum = TRUE)

}

```r
encontrar_mv <- function(n, n_err) {
  # Crear la función de log-verosimilitud para una cantidad fija de errores
  log_ver <- function(p) {
    n_err * log(p) + (n - n_err) * log(1 - p)
  }

  # Maximizar la log-verosimilitud para encontrar el mejor valor de p
  res <- optimize(log_ver, c(0, 1), maximum = TRUE)

  # Devolver el valor de p que maximiza la verosimilitud
  res$maximum
}

# Aplicar la función a cada simulación para calcular el estimador de máxima verosimilitud de p
```

```r
# Usamos el vector `simulaciones` que contiene el número de errores en cada simulación

estimadores_mv <- sapply(simulaciones, function(n_err) encontrar_mv(n = 500, n_err))

# Mostrar los primeros 10 estimadores de máxima verosimilitud como ejemplo
head(estimadores_mv, 10)
```

```
##  [1] 0.6440065 0.6300042 0.6540077 0.6480071 0.6300042 0.6040038 0.6460068
##  [8] 0.6259901 0.6600081 0.6380056
```

**verifica que los valores que generaste son proporciones entre0 y 1**

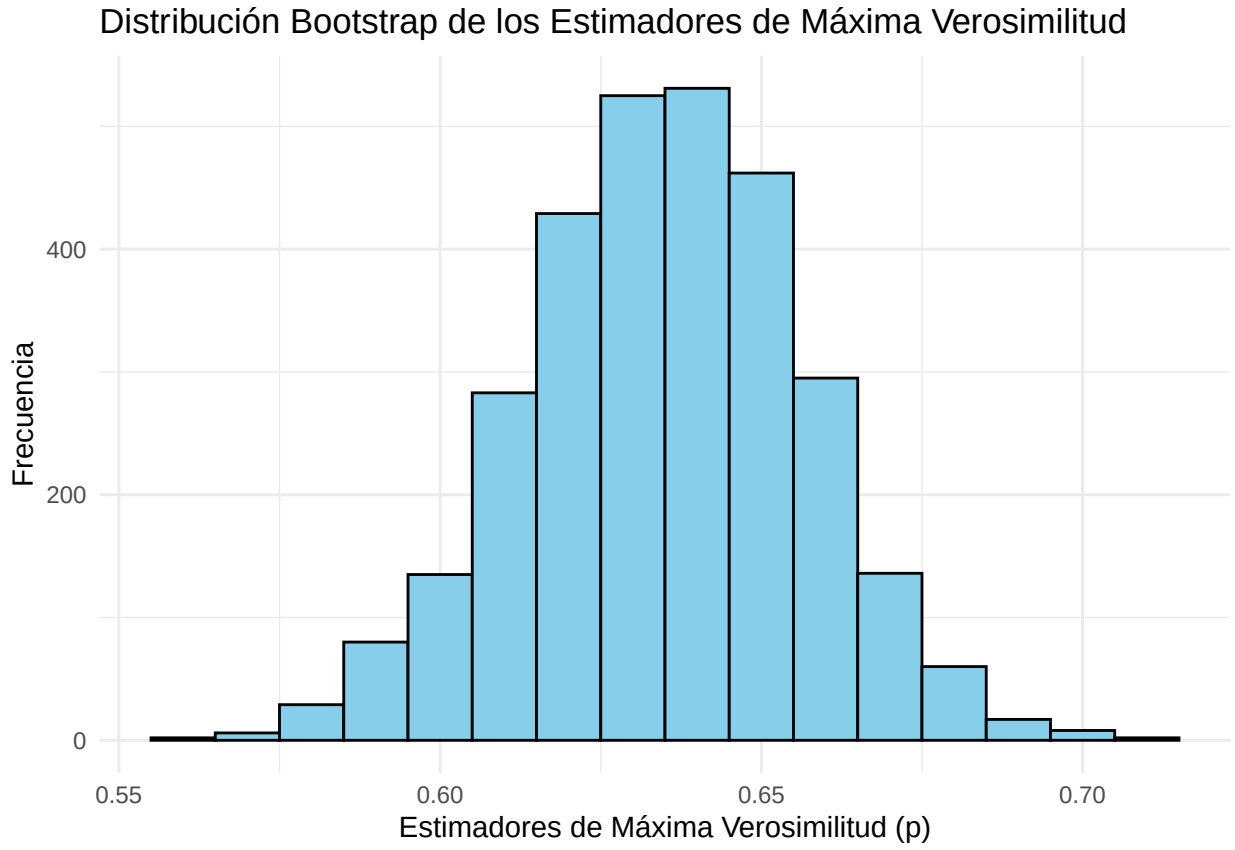
```r
summary(estimadores_mv)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5620  0.6200  0.6360  0.6348  0.6500  0.7120
```

**Paso 6: Resumen. Grafica un histograma de la distribución bootstrap y calcula el error estándar de tu estimacion de max verosimilitud**

```r
library(ggplot2)

# Graficar el histograma de la distribución bootstrap de los estimadores de máxima verosimilitud
ggplot(data = data.frame(estimadores_mv = estimadores_mv), aes(x = estimadores_mv)) +
  geom_histogram(binwidth = 0.01, color = "black", fill = "skyblue") +
  labs(
    title = "Distribución Bootstrap de los Estimadores de Máxima Verosimilitud",
    x = "Estimadores de Máxima Verosimilitud (p)",
    y = "Frecuencia"
  ) +
  theme_minimal()
```




```
# Calcular el error estándar de la estimación de máxima verosimilitud
error_estandar <- sd(estimadores_mv)
error_estandar
```

```
## [1] 0.02157807
```