Tarea 07 Cobertura

Rodrigo Alan Garcia Perez

2024-10-02

Intervalos bootstrap de percentiles

Una alternativa a los intervalos bootstrap normales son los intervalos bootstrap de percentiles. Para construir un intervalo del 95% de confianza usamos los percentiles correspondientes a las probabilidades 0.025 y 0.975 de la distribución bootstrap.

En este ejemplo calculamos intervalos de percentiles para el precio promedio de las casas, utilizando una muestra de tamaño 150.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# muestra original
set.seed(121)

poblacion_casas <- read_csv("data/casas.csv")
```

```
## Rows: 1144 Columns: 46
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (22): tipo_zona, calle, forma_lote, nombre_zona, tipo_edificio, estilo, ...
## dbl (24): id, frente_lote, calidad_gral, condicion_gral, año_construccion, b...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# muestra
muestra_casas <- slice_sample(poblacion_casas, n = 150)

# replicaciones bootstrap
medias_boot <- map_dbl(1:5000,  ~sample(muestra_casas$precio_miles, replace = TRUE) %>% mean())

# intervalo de percentiles
quantile(medias_boot, probs = c(0.025, 0.975))
```

```
##     2.5%    97.5%
```

```
## 178.5773 203.2153
```

Ejercicio

Iniciamos la tarea estableciendo una semilla para el generador de números aleatorios.

```
set.seed(38972938)
```

Cobertura

Supón que estamos interesados en entender como es el tráfico de llamadas a un conmutador, antes de realizar el estudio hacemos un análisis de simulación para estudiar el comportamiento de nuestro estimador bajo el supuesto que las llamadas al conmutador siguen un proceso $Poisson(\lambda)$.

En este ejemplo la cantidad de interés es $P(X = 0)^2 = e^{-2\lambda}$ que corresponde a la probabilidad de que no se reciban llamadas en 2 minutos.

Entonces el parámetro de interés $\theta = e^{-2\lambda}$ que estimaremos con el estimador plug-in $\hat{\theta} = e^{-2\bar{X}}$.

Sigue el siguiente proceso:

i) Genera una muestra aleatoria de tamaño n = 50 de una distribución Poisson con parámetro $\lambda = 3$ (revisa la función rpois).

ii) Genera 1000 muestras bootstrap y calcula intervalos de confianza del 95% para $\hat{\theta}$ usando 1) el método normal y 2) percentiles.

iii) Revisa si el intervalo de confianza contiene el verdadero valor del parámetro, en caso de que no lo contenga registra si falló por la izquierda o falló por la derecha, así como la longitud del intervalo.

```
# Parámetro lambda de la distribución de Poisson
lambda <- 3

# Parámetro de interés verdadero
theta_true <- exp(-2 * lambda)

# Tamaño de la muestra
n <- 50

# Número de muestras bootstrap
n_bootstrap <- 1000

# ----- Inciso (1) -----
set.seed(123)
muestra <- rpois(n, lambda)

# Estimador plug-in para theta
X_bar <- mean(muestra)  # Media muestral
theta_hat <- exp(-2 * X_bar)

# ----- Inciso (2) -----
# Función para generar muestras bootstrap y calcular theta
bootstrap_theta <- function(data, n_bootstrap, n) {
  thetas <- numeric(n_bootstrap)
  for (i in 1:n_bootstrap) {
    sample_boot <- sample(data, size = n, replace = TRUE)
    X_bar_boot <- mean(sample_boot)
    thetas[i] <- exp(-2 * X_bar_boot)
```

```r
  }
  return(thetas)
}

# Generar las muestras bootstrap y calcular theta para cada una
thetas_bootstrap <- bootstrap_theta(muestra, n_bootstrap, n)

# Método 1: Intervalo de confianza por el método normal
sd_bootstrap <- sd(thetas_bootstrap)
# Intervalo de confianza al 95% (método normal)
IC_normal <- round(c(theta_hat - 2 * sd_bootstrap, theta_hat + 2 * sd_bootstrap), 5)

# Método 2: Intervalo de confianza por percentiles
IC_percentiles <- round(quantile(thetas_bootstrap, c(0.025, 0.975)), 5)

# ----- Inciso (3) -----
# Revisar si los intervalos contienen el valor verdadero de theta
contiene_normal <- (IC_normal[1] <= theta_true) && (IC_normal[2] >= theta_true)
contiene_percentiles <- (IC_percentiles[1] <= theta_true) && (IC_percentiles[2] >= theta_true)

# Resultados
cat("Estimador plug-in de theta: ", theta_hat, "\n")
```

```
## Estimador plug-in de theta:  0.002029431
```

```r
cat("Intervalo de confianza normal (95%): ", IC_normal, "\n")
```

```
## Intervalo de confianza normal (95%):  -0.00038 0.00444
```

```r
cat("¿Contiene el valor verdadero theta? ", contiene_normal, "\n")
```

```
## ¿Contiene el valor verdadero theta?  TRUE
```

```r
cat("Intervalo de confianza por percentiles (95%): ", IC_percentiles, "\n")
```

```
## Intervalo de confianza por percentiles (95%):  0.00072 0.0053
```

```r
cat("¿Contiene el valor verdadero theta? ", contiene_percentiles, "\n")
```

```
## ¿Contiene el valor verdadero theta?  TRUE
```

```r
# Analizar si falla por la izquierda o por la derecha
if (!contiene_normal) {
  if (IC_normal[2] < theta_true) {
    cat("El intervalo normal falló por la izquierda.\n")
  } else if (IC_normal[1] > theta_true) {
    cat("El intervalo normal falló por la derecha.\n")
  }
}

if (!contiene_percentiles) {
  if (IC_percentiles[2] < theta_true) {
    cat("El intervalo por percentiles falló por la izquierda.\n")
  } else if (IC_percentiles[1] > theta_true) {
    cat("El intervalo por percentiles falló por la derecha.\n")
  }
}
```

```r
# Longitud de los intervalos redondeada
longitud_normal <- round(IC_normal[2] - IC_normal[1], 5)
longitud_percentiles <- round(IC_percentiles[2] - IC_percentiles[1], 5)

cat("Longitud del intervalo normal: ", longitud_normal, "\n")
```

```
## Longitud del intervalo normal:  0.00482
```

```r
cat("Longitud del intervalo por percentiles: ", longitud_percentiles, "\n")
```

```
## Longitud del intervalo por percentiles:  0.00458
```

a) Repite el proceso descrito 500 veces y llena la siguiente tabla:

Método	% fallo izquierda	% fallo derecha	cobertura
Normal			.
Percentiles			

La columna cobertura es una estimación de la cobertura del intervalo basada en las simulaciones, para calcularla simplemente escribe el porcentaje de los intervalos que incluyeron el verdadero valor del parámetro. Recuerda usar la semilla.

```r
# Optimizando el codigo para ejecutarlo 500 veces
lambda <- 3
theta_true <- exp(-2 * lambda)
n <- 50
n_bootstrap <- 1000
n_repeticiones <- 500

fallo_izquierda_normal <- 0
fallo_derecha_normal <- 0
cobertura_normal <- 0
fallo_izquierda_percentiles <- 0
fallo_derecha_percentiles <- 0
cobertura_percentiles <- 0

set.seed(123)
for (rep in 1:n_repeticiones) {
  muestra <- rpois(n, lambda)
  theta_hat <- exp(-2 * mean(muestra))

  thetas_bootstrap <- replicate(n_bootstrap, exp(-2 * mean(sample(muestra, size = n, replace = TRUE))))

  sd_bootstrap <- sd(thetas_bootstrap)
  IC_normal <- c(theta_hat - 2 * sd_bootstrap, theta_hat + 2 * sd_bootstrap)
  IC_percentiles <- quantile(thetas_bootstrap, c(0.025, 0.975))

  if (IC_normal[1] > theta_true) {
    fallo_derecha_normal <- fallo_derecha_normal + 1
  } else if (IC_normal[2] < theta_true) {
    fallo_izquierda_normal <- fallo_izquierda_normal + 1
  } else {
    cobertura_normal <- cobertura_normal + 1
  }
```

```
  if (IC_percentiles[1] > theta_true) {
    fallo_derecha_percentiles <- fallo_derecha_percentiles + 1
  } else if (IC_percentiles[2] < theta_true) {
    fallo_izquierda_percentiles <- fallo_izquierda_percentiles + 1
  } else {
    cobertura_percentiles <- cobertura_percentiles + 1
  }
}

porcentaje_fallo_izquierda_normal <- round((fallo_izquierda_normal / n_repeticiones) * 100, 5)
porcentaje_fallo_derecha_normal <- round((fallo_derecha_normal / n_repeticiones) * 100, 5)
porcentaje_cobertura_normal <- round((cobertura_normal / n_repeticiones) * 100, 5)

porcentaje_fallo_izquierda_percentiles <- round((fallo_izquierda_percentiles / n_repeticiones) * 100, 5)
porcentaje_fallo_derecha_percentiles <- round((fallo_derecha_percentiles / n_repeticiones) * 100, 5)
porcentaje_cobertura_percentiles <- round((cobertura_percentiles / n_repeticiones) * 100, 5)
```

Método	% fallo izquierda	% fallo derecha	cobertura
Normal	4.8	0	95.2
Percentiles	2	2	96

b) Realiza una gráfica de páneles, en cada panel mostrarás los resultados de uno de los métodos (normal, percentiles), el eje x corresponderá al número de intervalo de confianza $(1, ..., 500)$ y en ele eje vertical grafica el intervalo, usando color para indicar si cubrió al verdadero valor $exp(-2\lambda)$.

```
library(ggplot2)
library(tidyr)

# Parámetro lambda de la distribución de Poisson
lambda <- 3
theta_true <- exp(-2 * lambda)
n <- 50
n_bootstrap <- 1000
n_repeticiones <- 500

# Listas para almacenar los intervalos
intervalos_normal <- data.frame(rep = integer(), lower = numeric(), upper = numeric(), cubre = logical()
intervalos_percentiles <- data.frame(rep = integer(), lower = numeric(), upper = numeric(), cubre = logi

# Repetir el proceso 500 veces
set.seed(123)
for (rep in 1:n_repeticiones) {
  muestra <- rpois(n, lambda)
  theta_hat <- exp(-2 * mean(muestra))

  thetas_bootstrap <- replicate(n_bootstrap, exp(-2 * mean(sample(muestra, size = n, replace = TRUE))))

  sd_bootstrap <- sd(thetas_bootstrap)
  IC_normal <- c(theta_hat - 2 * sd_bootstrap, theta_hat + 2 * sd_bootstrap)
  IC_percentiles <- quantile(thetas_bootstrap, c(0.025, 0.975))

  # Almacenar resultados para el método normal
```

```r
  cubre_normal <- IC_normal[1] <= theta_true && IC_normal[2] >= theta_true
  intervalos_normal <- rbind(intervalos_normal, data.frame(rep = rep, lower = IC_normal[1], upper = IC_n

  # Almacenar resultados para el método percentiles
  cubre_percentiles <- IC_percentiles[1] <= theta_true && IC_percentiles[2] >= theta_true
  intervalos_percentiles <- rbind(intervalos_percentiles, data.frame(rep = rep, lower = IC_percentiles[1
}

# Calcular porcentaje de cubrimiento y no cubrimiento para el método normal
porcentaje_cubre_normal <- sum(intervalos_normal$cubre) / n_repeticiones * 100
porcentaje_no_cubre_normal <- 100 - porcentaje_cubre_normal

# Calcular porcentaje de cubrimiento y no cubrimiento para el método de percentiles
porcentaje_cubre_percentiles <- sum(intervalos_percentiles$cubre) / n_repeticiones * 100
porcentaje_no_cubre_percentiles <- 100 - porcentaje_cubre_percentiles

# Imprimir resultados
cat("Resultados para el método normal:\n")
```

```
## Resultados para el método normal:
```

```r
cat("Porcentaje de intervalos que cubrieron el valor verdadero:", round(porcentaje_cubre_normal, 2), "%"
```

```
## Porcentaje de intervalos que cubrieron el valor verdadero: 95.2 %
```

```r
cat("Porcentaje de intervalos que NO cubrieron el valor verdadero:", round(porcentaje_no_cubre_normal, 2
```

```
## Porcentaje de intervalos que NO cubrieron el valor verdadero: 4.8 %
```

```r
cat("Resultados para el método de percentiles:\n")
```

```
## Resultados para el método de percentiles:
```

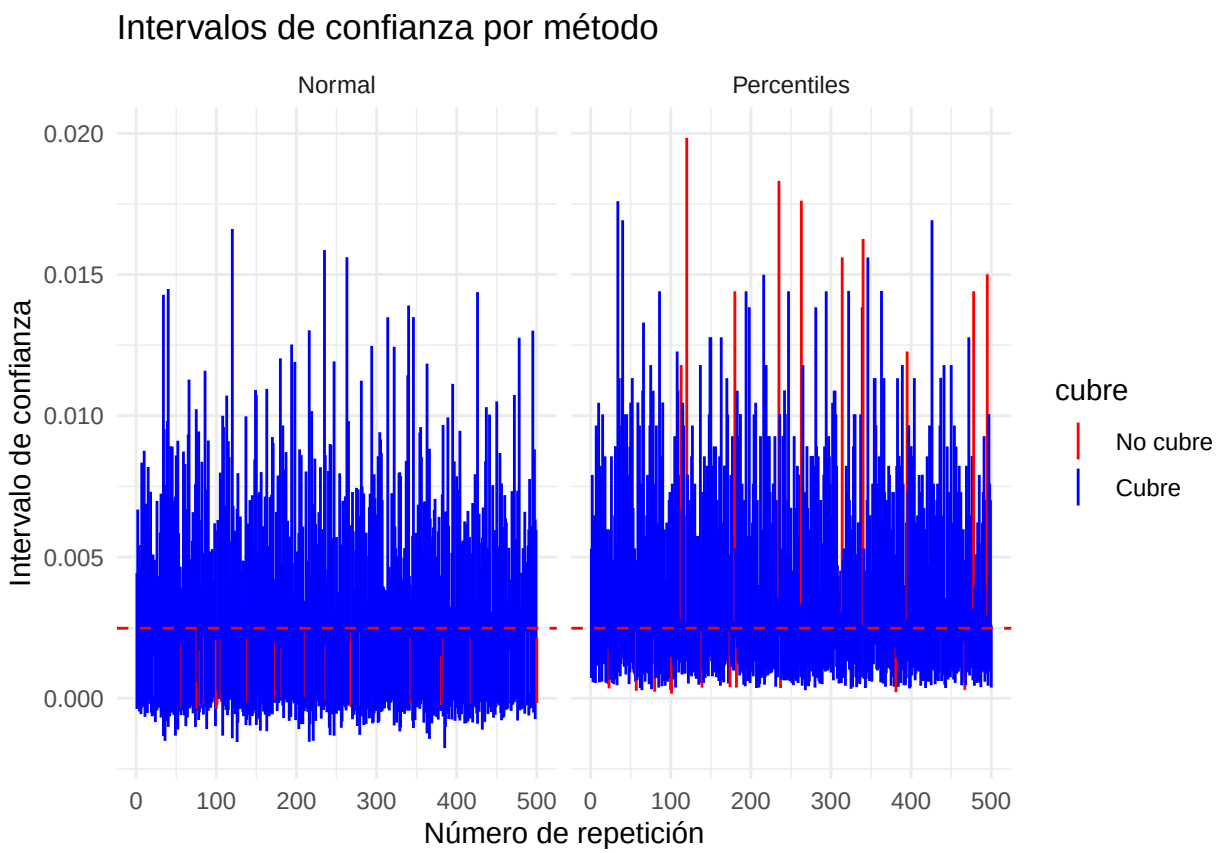
```r
cat("Porcentaje de intervalos que cubrieron el valor verdadero:", round(porcentaje_cubre_percentiles, 2)
```

```
## Porcentaje de intervalos que cubrieron el valor verdadero: 96 %
```

```r
cat("Porcentaje de intervalos que NO cubrieron el valor verdadero:", round(porcentaje_no_cubre_percentil
```

```
## Porcentaje de intervalos que NO cubrieron el valor verdadero: 4 %
```

```r
# Agregar una columna con el método para poder graficar en paneles
intervalos_normal$metodo <- "Normal"
intervalos_percentiles$metodo <- "Percentiles"

# Combinar los dos dataframes
intervalos <- rbind(intervalos_normal, intervalos_percentiles)

# Realizar la gráfica
ggplot(intervalos, aes(x = rep, y = lower, ymax = upper, ymin = lower, color = cubre)) +
  geom_linerange() +
  geom_hline(yintercept = theta_true, linetype = "dashed", color = "red") +  # Valor verdadero de theta
  facet_wrap(~ metodo) +  # Crear un panel por cada método
  labs(title = "Intervalos de confianza por método", x = "Número de repetición", y = "Intervalo de confi
  scale_color_manual(values = c("red", "blue"), labels = c("No cubre", "Cubre")) +
  theme_minimal()
```



c) Repite considerando que se selecciona una muestra de tamaño n = 100.

```r
library(ggplot2)
library(tidyr)

# Parámetro lambda de la distribución de Poisson
lambda <- 3
theta_true <- exp(-2 * lambda)
n <- 100  # Cambiamos el tamaño de la muestra a 100
n_bootstrap <- 1000
n_repeticiones <- 500

# Listas para almacenar los intervalos
intervalos_normal <- data.frame(rep = integer(), lower = numeric(), upper = numeric(), cubre = logical()
intervalos_percentiles <- data.frame(rep = integer(), lower = numeric(), upper = numeric(), cubre = log

# Repetir el proceso 500 veces
set.seed(123)
for (rep in 1:n_repeticiones) {
  muestra <- rpois(n, lambda)
  theta_hat <- exp(-2 * mean(muestra))

  thetas_bootstrap <- replicate(n_bootstrap, exp(-2 * mean(sample(muestra, size = n, replace = TRUE))))

  sd_bootstrap <- sd(thetas_bootstrap)
  IC_normal <- c(theta_hat - 2 * sd_bootstrap, theta_hat + 2 * sd_bootstrap)
  IC_percentiles <- quantile(thetas_bootstrap, c(0.025, 0.975))
```

```r
  # Almacenar resultados para el método normal
  cubre_normal <- IC_normal[1] <= theta_true && IC_normal[2] >= theta_true
  intervalos_normal <- rbind(intervalos_normal, data.frame(rep = rep, lower = IC_normal[1], upper = IC_

  # Almacenar resultados para el método percentiles
  cubre_percentiles <- IC_percentiles[1] <= theta_true && IC_percentiles[2] >= theta_true
  intervalos_percentiles <- rbind(intervalos_percentiles, data.frame(rep = rep, lower = IC_percentiles[
}

# Calcular porcentaje de cubrimiento y no cubrimiento para el método normal
porcentaje_cubre_normal <- sum(intervalos_normal$cubre) / n_repeticiones * 100
porcentaje_no_cubre_normal <- 100 - porcentaje_cubre_normal

# Calcular porcentaje de cubrimiento y no cubrimiento para el método de percentiles
porcentaje_cubre_percentiles <- sum(intervalos_percentiles$cubre) / n_repeticiones * 100
porcentaje_no_cubre_percentiles <- 100 - porcentaje_cubre_percentiles

# Imprimir resultados
cat("Resultados para el método normal:\n")
```

```
## Resultados para el método normal:
```

```r
cat("Porcentaje de intervalos que cubrieron el valor verdadero:", round(porcentaje_cubre_normal, 2), "%
```

```
## Porcentaje de intervalos que cubrieron el valor verdadero: 95.4 %
```

```r
cat("Porcentaje de intervalos que NO cubrieron el valor verdadero:", round(porcentaje_no_cubre_normal, 
```

```
## Porcentaje de intervalos que NO cubrieron el valor verdadero: 4.6 %
```

```r
cat("Resultados para el método de percentiles:\n")
```

```
## Resultados para el método de percentiles:
```

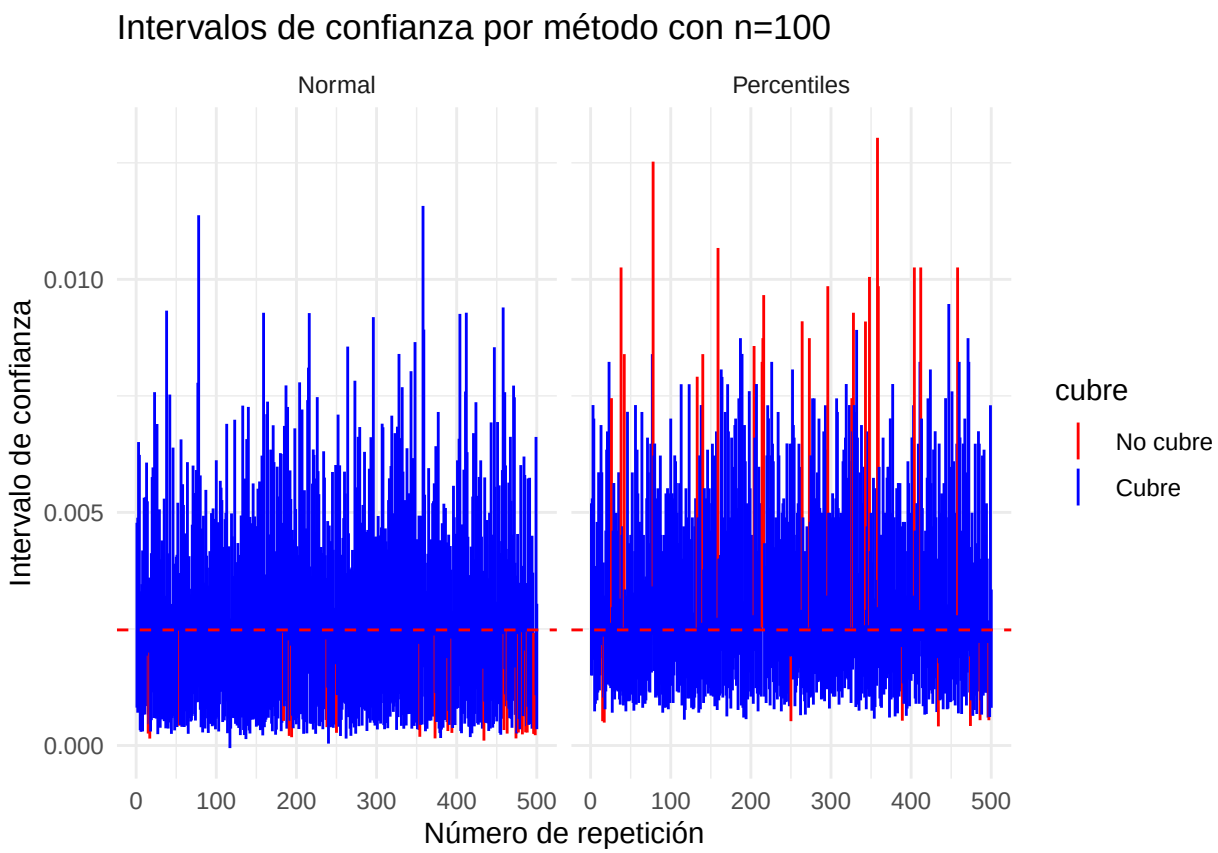
```r
cat("Porcentaje de intervalos que cubrieron el valor verdadero:", round(porcentaje_cubre_percentiles, 2
```

```
## Porcentaje de intervalos que cubrieron el valor verdadero: 93.8 %
```

```r
cat("Porcentaje de intervalos que NO cubrieron el valor verdadero:", round(porcentaje_no_cubre_percenti
```

```
## Porcentaje de intervalos que NO cubrieron el valor verdadero: 6.2 %
```

```r
# Agregar una columna con el método para poder graficar en paneles
intervalos_normal$metodo <- "Normal"
intervalos_percentiles$metodo <- "Percentiles"

# Combinar los dos dataframes
intervalos <- rbind(intervalos_normal, intervalos_percentiles)

# Realizar la gráfica
ggplot(intervalos, aes(x = rep, y = lower, ymax = upper, ymin = lower, color = cubre)) +
  geom_linerange() +
  geom_hline(yintercept = theta_true, linetype = "dashed", color = "red") +  # Valor verdadero de theta
  facet_wrap(~ metodo) +  # Crear un panel por cada método
  labs(title = "Intervalos de confianza por método con n=100", x = "Número de repetición", y = "Interval
  scale_color_manual(values = c("red", "blue"), labels = c("No cubre", "Cubre")) +
  theme_minimal()
```

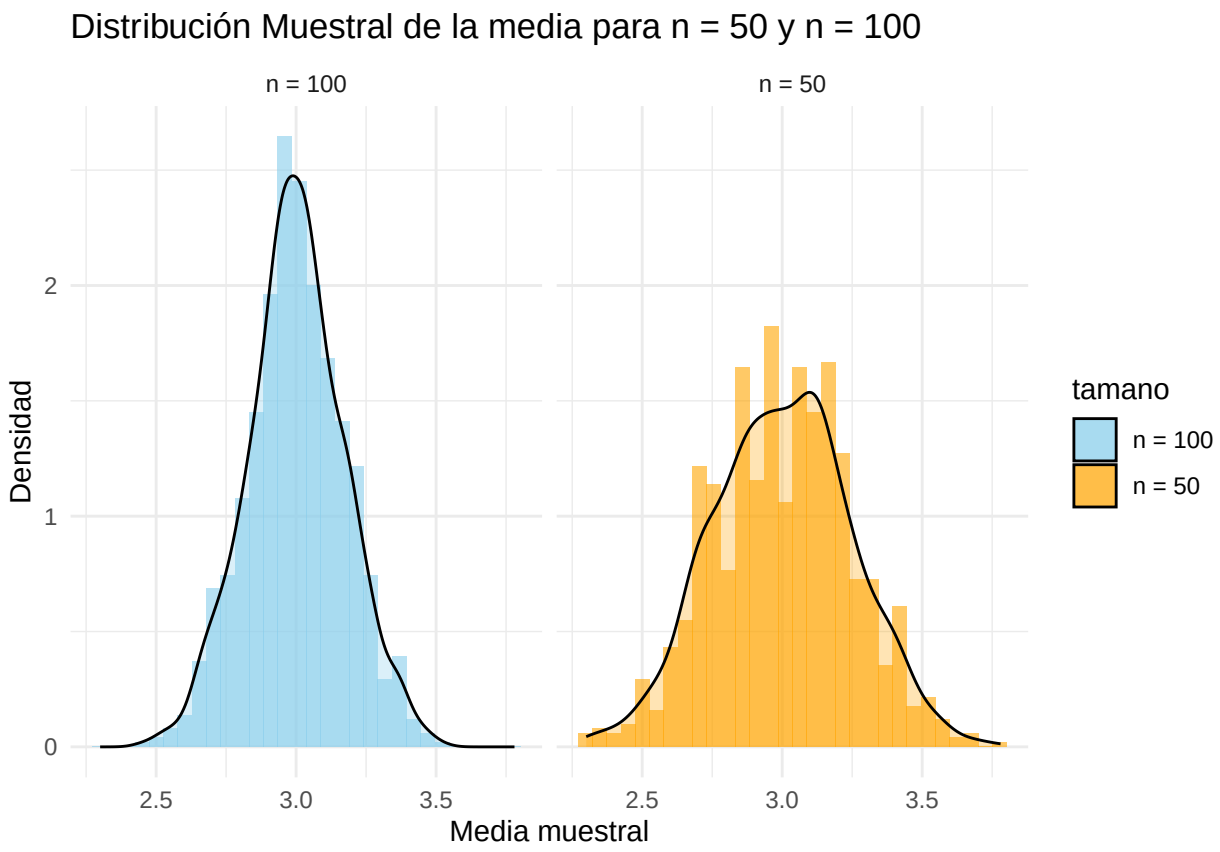


d) Grafica la distribución muestral para una simulación de cada caso de tamaño de muestra, es decir, dos gráficas únicamente.

```r
library(ggplot2)

# Parámetros
lambda <- 3

# Tamaños de muestra
n_50 <- 50
n_100 <- 100
n_simulaciones <- 1000

# Simulaciones para n = 50
muestras_50 <- replicate(n_simulaciones, mean(rpois(n_50, lambda)))

# Simulaciones para n = 100
muestras_100 <- replicate(n_simulaciones, mean(rpois(n_100, lambda)))

# Crear dataframes para las muestras simuladas
df_50 <- data.frame(muestras = muestras_50, tamano = "n = 50")
df_100 <- data.frame(muestras = muestras_100, tamano = "n = 100")

# Combinar los dataframes
df <- rbind(df_50, df_100)

# Graficar las distribuciones muestrales
```

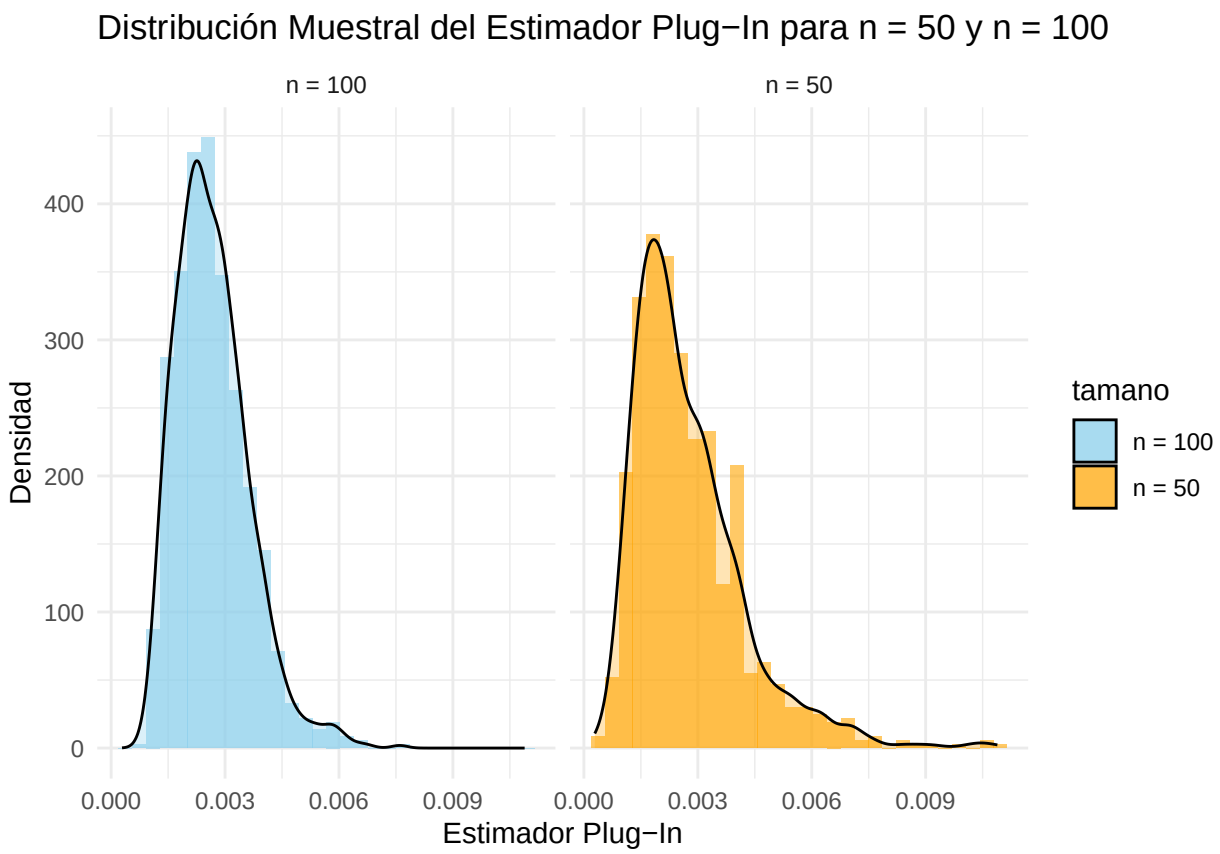
```r
ggplot(df, aes(x = muestras, fill = tamano)) +
  geom_histogram(aes(y = ..density..), bins = 30, alpha = 0.6, position = "identity") +
  geom_density(alpha = 0.3) +
  facet_wrap(~ tamano) +
  labs(title = "Distribución Muestral de la media para n = 50 y n = 100",
       x = "Media muestral",
       y = "Densidad") +
  theme_minimal() +
  scale_fill_manual(values = c("skyblue", "orange"))
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```r
library(ggplot2)

# Parámetros
lambda <- 3

# Tamaños de muestra
n_50 <- 50
n_100 <- 100
n_simulaciones <- 1000

# Simulaciones para n = 50 usando el estimador plug-in
```

```r
plugin_50 <- replicate(n_simulaciones, exp(-2 * mean(rpois(n_50, lambda))))

# Simulaciones para n = 100 usando el estimador plug-in
plugin_100 <- replicate(n_simulaciones, exp(-2 * mean(rpois(n_100, lambda))))

# Crear dataframes para las muestras simuladas
df_50 <- data.frame(plugin = plugin_50, tamano = "n = 50")
df_100 <- data.frame(plugin = plugin_100, tamano = "n = 100")

# Combinar los dataframes
df <- rbind(df_50, df_100)

# Graficar las distribuciones muestrales del estimador plug-in
ggplot(df, aes(x = plugin, fill = tamano)) +
  geom_histogram(aes(y = ..density..), bins = 30, alpha = 0.6, position = "identity") +
  geom_density(alpha = 0.3) +
  facet_wrap(~ tamano) +
  labs(title = "Distribución Muestral del Estimador Plug-In para n = 50 y n = 100",
       x = "Estimador Plug-In",
       y = "Densidad") +
  theme_minimal() +
  scale_fill_manual(values = c("skyblue", "orange"))
```



Conclusiones

En n= 100 el método de percentiles tiene peor cobertura (93.8%) en comparación con el método normal (95.4%), lo cual es sorprendente ya que esperariamos que al aumentar el tamaño de la muestra, ambos métodos

se acerquen al nivel de confianza esperado del 95%. Investigando entre las diferencias de ambos métodos, encontre que el de percentiles aunque es un enfoque más flexible, puede ser más sensible a la asimetría o distribuciones no normales, lo que puede hacer que el intervalo de confianza no sea tan preciso como en el método normal, especialmente en situaciones donde la asimetría juega un rol importante.

Como vimos en la gráfica del estimador plug-in, hay una clara asimetría hacia la derecha. Esta asimetría puede hacer que los percentiles 2.5 y 97.5 no representen correctamente la "cola" de la distribución, lo que hace que el método de percentiles no capture correctamente el valor verdadero de theta.

El método de percentiles puede empeorar a medida que el tamaño de la muestra aumenta si el estimador plug-in genera distribuciones bootstrap que son asimétricas o tienen colas pesadas, lo cual parece ser el caso en este escenario.

Por otro lado, el método normal, que asume que las distribuciones son aproximadamente normales a medida que crece, parece adaptarse mejor en este caso, proporcionando una cobertura más consistente.