Final-2023

Mariano Villafuerte -156057 Mario Medina - 156940

01 diciembre 2023

Contents

1 Relación bootstrap e inferencia bayesiana

1.1 Contexto

Consideremos el caso en que tenemos una única observación x proveniente de una distribución normal

$$x \sim N(\theta, 1)$$

Supongamos ahora que elegimos una distribución inicial Normal para el parámetro θ.

$$\theta \sim N(0, \tau)$$

dando lugar a la distribución posterior (como vimos en la tarea)

$$\theta|x \sim N\left(\frac{x}{1 + 1/\tau}, \frac{1}{1 + 1/\tau}\right)$$

Ahora, entre mayor τ, más se concentra la posterior en el estimador de máxima verosimilitud $\hat{\theta} = x$. En el límite, cuando $\tau \to \infty$ obtenemos una inicial no-informativa (constante) y la distribución posterior

$$\theta|x \sim N(x, 1)$$

Esta posterior coincide con la distribución de bootstrap paramétrico en que generamos valores x^* de $N(x, 1)$, donde x es el estimador de máxima verosimilitud.

Lo anterior se cumple debido a que utilizamos un ejemplo Normal pero también se cumple aproximadamente en otros casos, lo que conlleva a una correspondencia entre el bootstrap paramétrico y la inferencia bayesiana. En este caso, la distribución bootstrap representa (aproximadamente) una distribución posterior no-informartiva del parámetro de interés. Mediante la perturbación en los datos el bootstrap aproxima el efecto bayesiano de perturbar los parámetros con la ventaja de ser más simple de implementar (en muchos casos). *Los detalles se pueden leer en *The Elements of Statistical Learning* de Hastie y Tibshirani.

Comparemos los métodos en otro problema con el fin de apreciar la similitud en los procedimientos:

Supongamos $x_1, ..., x_n \sim N(0, \sigma^2)$, es decir, los datos provienen de una distribución con media cero y varianza desconocida.

En los puntos 2.1 y 2.2 buscamos hacer inferencia del parámetro σ^2.

1.2 Bootstrap paramétrico

1.2.1 Log-verosimilitud y estimador de mv

Sabemos que $\mu = 0$, entonces nos quedamos con la parte de la distribución que considera el término de σ^2. Tenemos 150 datos obtenidos del archivo `x.RData`

```r
load(paste0(wd, "/data/x.RData"))
```

$$\begin{aligned}
f(x|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\
f(x|0, \sigma^2) &= \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \\
L(0, \sigma^2|x) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} (\sigma^2)^{\frac{-1}{2}} e^{-\frac{x_i^2}{2\sigma^2}} \propto (\sigma^2)^{\frac{-n}{2}} e^{-\frac{\sum^n x_i^2}{2\sigma^2}} \\
l(0, \sigma^2|x) &= \frac{-n}{2} log(\sigma^2) - \frac{\sum^n x_i^2}{2} (\sigma^2)^{-1}
\end{aligned}$$

Tenemos $n = 150$ y $\sum^{150} x^2 = 19693.64$ podemos hacerlo de forma manual y calcular la derivada e igualar a 0.

$$\frac{\partial l}{\partial \sigma^2} = \frac{-150}{2} \frac{1}{\sigma^2} + \frac{19693.64}{2} \frac{1}{(\sigma^2)^2} = 0$$

Se puede resolver asumiendo $x = \frac{1}{\sigma^2}$ y resolvemos por la "chicharronera"

```r
set.seed(156057)
# Coefficients
a <- 9846.821
b <- -75
```

```r
c <- 0

# Quadratic formula
discriminant <- b^2 - 4 * a * c
x1 <- (-b + sqrt(discriminant)) / (2 * a)
x2 <- (-b - sqrt(discriminant)) / (2 * a)

# Reciprocal to get sigma^2
sigma_squared_1 <- 1 / x1
sigma_squared_2 <- 1 / x2
```

Lo que nos da un valor **para $\sigma^2_{mv} = 131.291$. Podemos verlo graficamente y replicarlo con métodos de optimización numérica**

```r
set.seed(156057)
log_p <- function(pars){
    (-150/2)*log(pars[1]) - (19693.64/2)*((1)/pars[1])
}
solucion <- optim(c(0.5), log_p,
                  control = list(fnscale = -1, maxit = 10000),
                  method = "Nelder-Mead")

print(paste0("Comprobamos convergencia: ",solucion$convergence))
```

```
## [1] "Comprobamos convergencia: 0"
```

```r
est_mv <- tibble(parametro = c("varianza"), estimador = solucion$par) %>%
  column_to_rownames(var = "parametro")
est_mv
```

```
##          estimador
## varianza    131.65
```

```r
dat_verosim <- tibble(x = seq(5,300, 0.01)) %>% mutate(log_prob = map_dbl(x, log_p))
ggplot(dat_verosim, aes(x = x, y = log_prob)) + geom_line() +
  geom_vline(xintercept = 131.291, color = "red") +
  labs(title = "Optimización") +
  xlab("Varianza (sigma^2)") + ylab("Log verosimilitud") +
  theme_pubclean(base_size = 12)
```

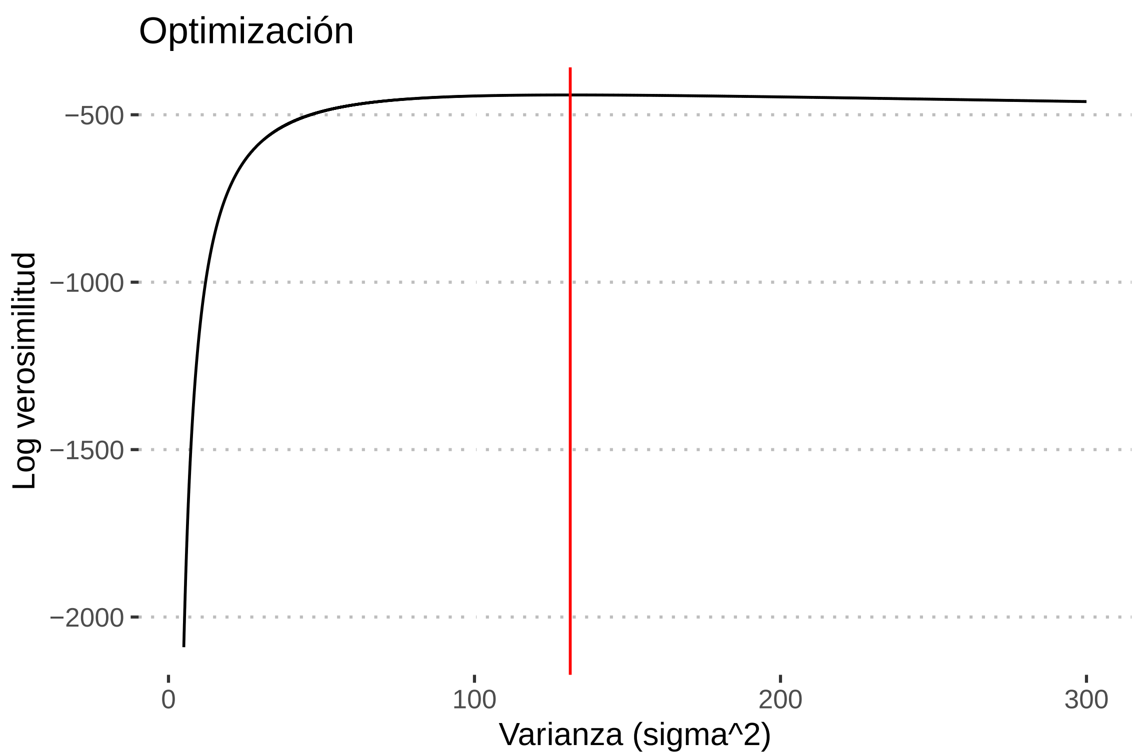

Figure 1: Optimización varianza

1.2.2 Estimación de ee - Boostrap paramétrico

Creamos nuestro flujo de generador de muestra (utilizando el parámetro de máxima verosimilitud), cálculamos la log-verosimilitud y optimizamos

```
set.seed(156057)

est_mle <- 131.291
n <- 150

rep_boot <- function(rep, log_p, n, est_mle){
  muestra_bootstrap <- rnorm(n, 0, sqrt(est_mle))
  log_p <- function(pars){
    (-n/2)*log(pars[1]) - (sum(muestra_bootstrap^2)/2)*((1)/pars[1])
  }
  solucion <- optim(c(0.5), log_p,
                    control = list(fnscale = -1, maxit = 10000),
                    method = "Nelder-Mead")
  try(if(solucion$convergence != 0) stop("No se alcanzó convergencia."))
  tibble(parametro = c("varianza"), estimador_boot = solucion$par)
}

reps_boot <- map_dfr(1:15000, ~ rep_boot(.x, log_p, n = length(x),
                                         est_mle), rep = ".id")

ggplot(reps_boot, aes(x = estimador_boot)) +
```

```
  geom_histogram(aes(x = estimador_boot, y = ..density..), bins = 30,
                 fill = "#F8766D",
                 alpha=0.5) +
  geom_vline(xintercept = est_mle, color = "red", linetype = "dashed") +
  labs(title = "Replicaciones bootstrap", subtitle = "MLE: Varianza") +
  xlab("Varianza (Sigma^2)") + ylab("Densidad") +
  ggpubr::theme_pubclean(base_size = 12)
```

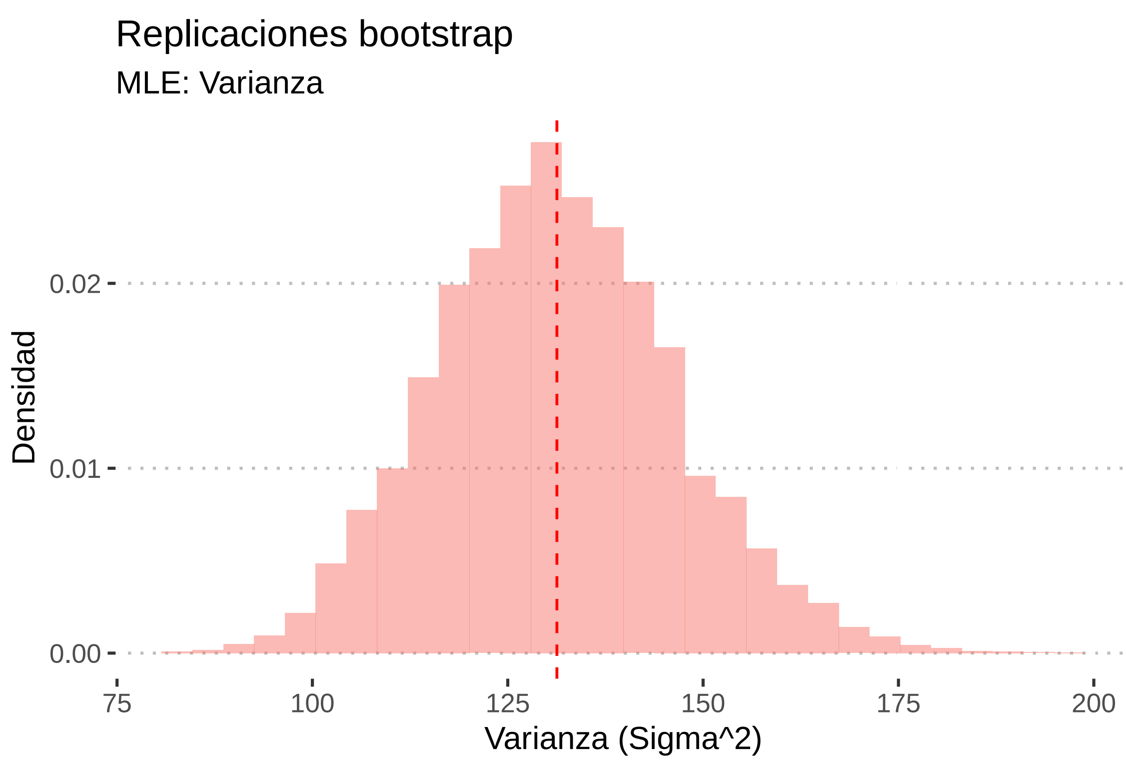

Figure 2: Histograma replicaciones bootstrap

Ahora podemos **calcular el error estándar de nuestra estimación**

```
set.seed(156057)
error_est <- reps_boot %>% group_by(parametro) %>%
  summarise(ee_boot = sd(estimador_boot))
error_est
```

```
## # A tibble: 1 x 2
##   parametro ee_boot
##   <chr>       <dbl>
## 1 varianza     15.3
```

Resumiendo. Nuestro $\sigma^2_{MLE} = 131.291$ y su $\hat{ee} = 15.186$

1.3 Análisis Bayesiano

1.3.1 Gamma Inversa

Empezamos definiendo una Gamma Inversa, **los parámetros al no tener mayor contexto del problema serán de un valor bajo mostrando que es una a priori con poca información. Asimismo buscando aprovechar las colas pesadas de la distribución ya que no tenemos certeza de la cantidad de varianza del problema** e.g $\alpha = 0.05, \beta = 2$

$$f(\sigma^2) : \frac{\beta^\alpha}{\Gamma(\alpha)} * \frac{1}{(\sigma^2)^{\alpha+1}} * e^{\frac{\beta}{\sigma^2}}$$

```r
set.seed(156057)
# Calculate the corresponding alpha and beta
alpha <- 0.05  # Adjust based on your preference
beta <- 2

# Generate a range of sigma^2 values
sigma2_values <- seq(0.01, 150, by = 0.01)

# Calculate the probability densities using the Inverse Gamma density function
density_values <- actuar::dinvgamma(sigma2_values, shape = alpha, rate = beta)

# Create a data frame for ggplot2
df <- data.frame(sigma2 = sigma2_values, density = density_values)

ggplot(df, aes(x = sigma2, y = density)) +
  geom_line() +
  labs(title = "Función de densidad para la gamma inversa",
       subtitle = "InvGamma(0.05,2)",
       x = expression(sigma^2),
       y = "Densidad") +
  ggpubr::theme_pubclean(base_size = 12)
```

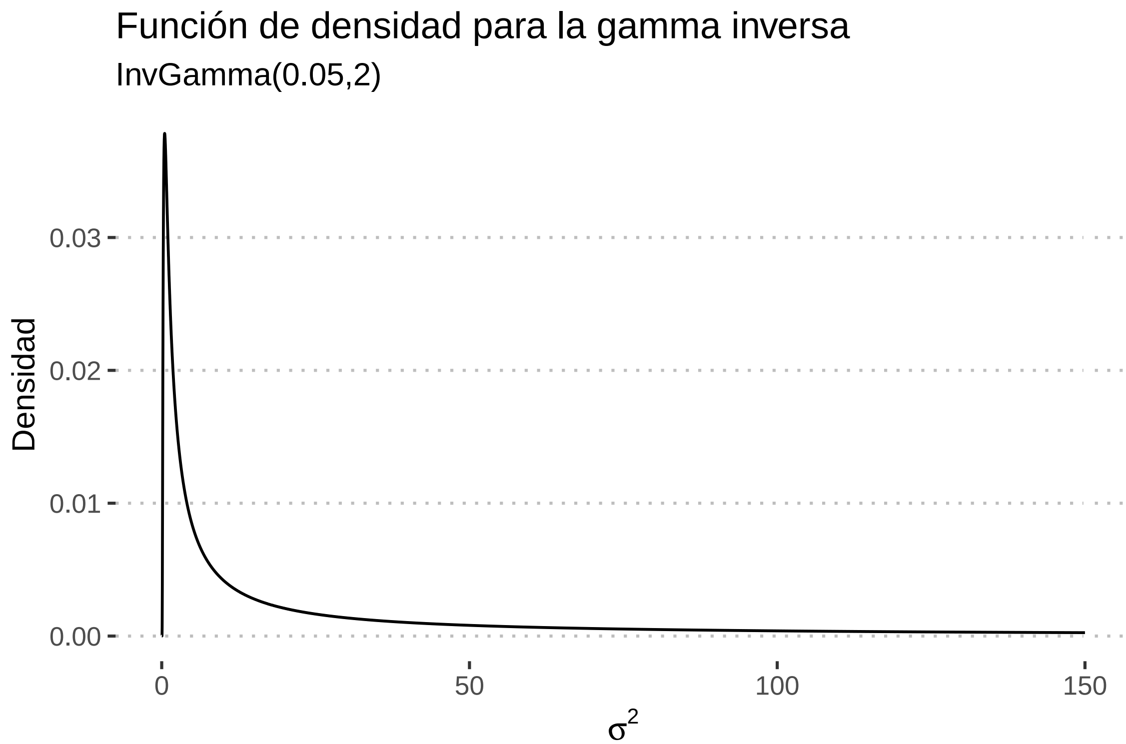

Figure 3: Distribución de inversa gamma

1.3.2 Distribución posterior.

Sabemos que la posterior es el producto de los núcleos de la verosimilitud y de la apriori por lo que tenemos lo siguiente.

- Conocemos "n" y la suma de x^2
- Conocemos α, β

$$
\begin{aligned}
&P(\sigma^2|x) = P(x|\sigma^2)P(\sigma^2)\\
&P(\sigma^2|x) \propto \left((\sigma^2)^{\frac{-n}{2}} e^{-\frac{\sum^n x_i^2}{2\sigma^2}}\right)\left((\sigma^2)^{-(\alpha+1)} e^{\frac{\beta}{\sigma^2}}\right)\\
&P(\sigma^2|x) \propto (\sigma^2)^{-76.05} e^{\frac{9848.82}{\sigma^2}}\\
&P(\sigma^2|x) \sim InvGamma(75.05, 9848.82)
\end{aligned}
$$

1.3.3 Simulaciones de posterior y ee

```
set.seed(156057)
alpha_post <- 75.05
beta_post <- 9848.82

post_samples <- 1 / rgamma(15000, shape = alpha_post, rate = beta_post)
df_posterior <- data.frame(post_samples)
```

```r
a <- ggplot(df_posterior, aes(x = post_samples)) +
  geom_histogram(aes(x = post_samples, y = ..density..), bins = 30,
                 fill = "#F8766D",
                 alpha=0.5) +
  geom_vline(xintercept = (9848.82/74.05), color = "red", linetype = "dashed") +
  annotate("text", x = (9848.82/74.05), y = Inf, label = "E[sigma^2]: 133.00",
           vjust = 1, hjust = 0.5, colour = "red")+
  labs(title = "Simulaciones de posterior",
       subtitle = "InvGamma(75.05,9848.82)") +
  xlab("Varianza (Sigma^2)") + ylab("Densidad") +
  xlim(80,250)+
  ggpubr::theme_pubclean(base_size = 8)

b <- ggplot(reps_boot, aes(x = estimador_boot)) +
  geom_histogram(aes(x = estimador_boot, y = ..density..), bins = 30,
                 fill = "#F8766D",
                 alpha=0.5) +
  geom_vline(xintercept = est_mle, color = "red", linetype = "dashed") +
  annotate("text", x = est_mle, y = Inf, label = "MLE: 131.29",
           vjust = 1, hjust = 0.5,colour = "red")+
  labs(title = "Replicaciones bootstrap", subtitle = "MLE: Varianza") +
  xlab("Varianza (Sigma^2)") + ylab("Densidad") +
  xlim(80,250)+
  ggpubr::theme_pubclean(base_size = 8)

a+b
```

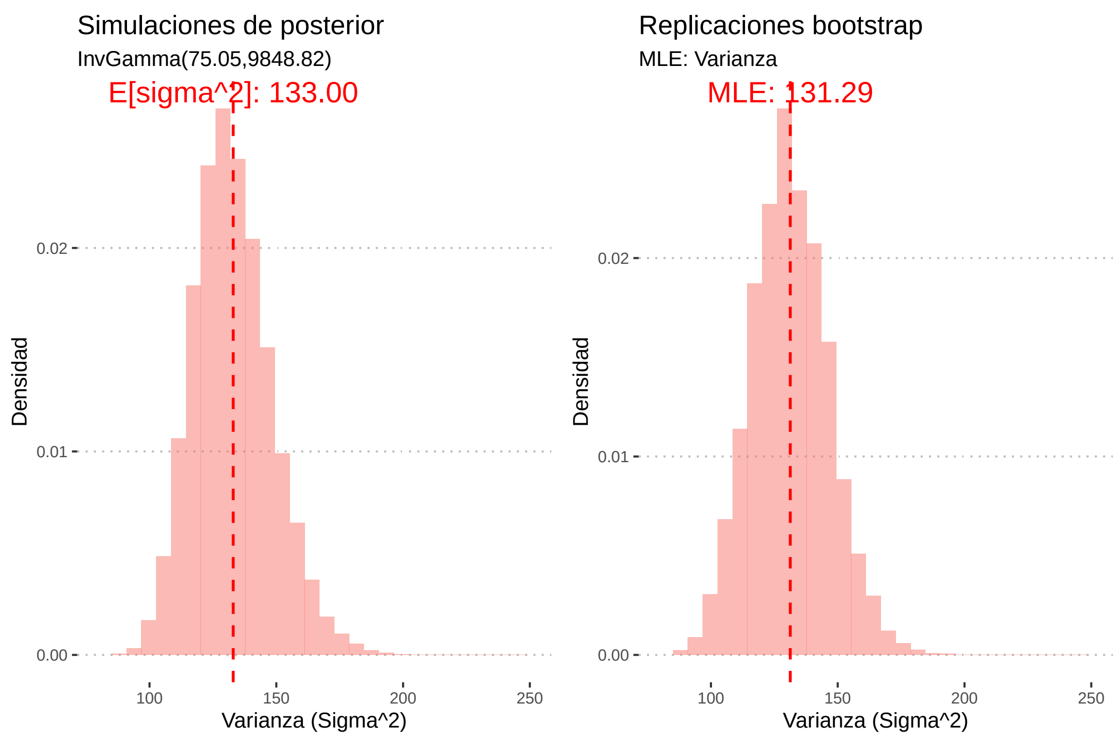

Figure 4: Comparativo Bayesiana vs. MLE

Calculamos el error estándar

```
set.seed(156057)
error_est_bayes <- df_posterior %>%
  dplyr::mutate(parametro = "varianza") %>%
  dplyr::group_by(parametro) %>%
  summarise(ee_bayes = sd(post_samples))
error_est_bayes
```

```
## # A tibble: 1 x 2
##   parametro ee_bayes
##   <chr>        <dbl>
## 1 varianza      15.5
```

```
error_est_bayes %>% left_join(error_est)
```

```
## # A tibble: 1 x 3
##   parametro ee_bayes ee_boot
##   <chr>        <dbl>   <dbl>
## 1 varianza      15.5    15.3
```

1.3.4 Comparativo Bayesiana vs. Bootstrap paramétrico

En la gráfica de arriba como en la tabla viene el comparativo de los estimadores. Ponemos de nuevo el resumen

- Bayesiana: Calculamos el valor esperado ($E[\sigma^2] = \frac{\beta}{\alpha-1} = 133.00$)
- Bootstrap paramétrico: Calculamos el estimador por medio de máxima verosimilitud (i.e derivando igualando a 0) $\sigma^2_{MV} = 131.29$

Y los errores estándar obtenido por medio de simulaciones.

- Bayesiana: Distribución posterior $InvGamma(75.05, 9848.82)$ $\hat{ee} = 15.459$
- Bootstrap paramétrico: Distribución $Normal(0, \sigma^2_{MV})$ $\hat{ee} = 15.302$

Corroboramos la correspondencia

1.4 Parámetro $=$ log( )

1.4.1 Máxima verosmilitud

Podemos argumentar al tratarse de una transformación logaritmica bien definida (sobre valores estricatamente positivos) que por la propiedad de **Equivarianza de MLE** que…

$$\hat{\tau} = g(\widehat{\sigma^2}) = log(\sqrt{131.291}) = 2.4387$$

será el **MLE** de τ

```r
reps_boot <-
  reps_boot %>% dplyr::mutate(tau_boot = log(sqrt(estimador_boot)))
ggplot(reps_boot, aes(x = tau_boot)) +
  geom_histogram(aes(x = tau_boot, y = ..density..), bins = 30,
                 fill = "#F8766D",
                 alpha=0.5) +
  geom_vline(xintercept = 2.4387, color = "red", linetype = "dashed") +
  labs(title = "Replicaciones bootstrap", subtitle = "MLE: Tau") +
  xlab("Tau") + ylab("Densidad") +
  ggpubr::theme_pubclean(base_size = 12)
```

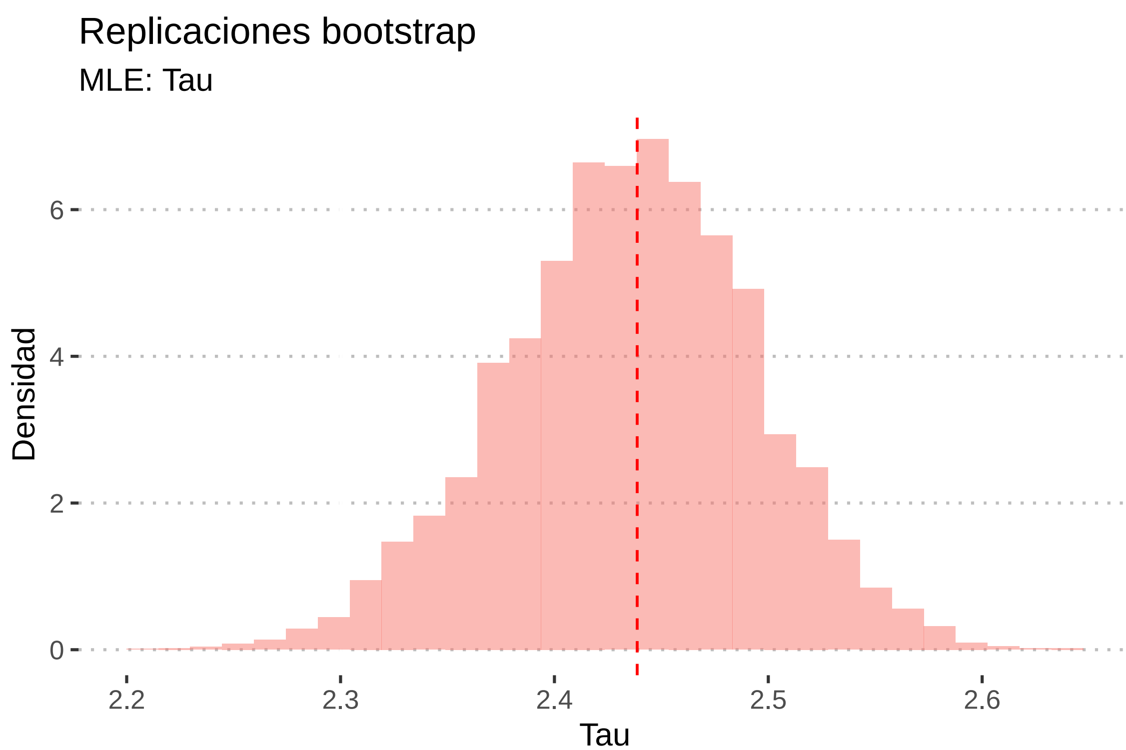

Figure 5: Histograma bootstrap: Tau

Podemos utilizar **intervalos de cuantiles para reportar un intervalo al 95%**

```r
quantil_95_izq <- quantile(reps_boot$tau_boot,.025)
quantil_95_der <- quantile(reps_boot$tau_boot,.975)
print(paste0("Intervalo de confianza al 95% es : (",
             round(quantil_95_izq,3), ", ",round(quantil_95_der,3),")"))
```

```
## [1] "Intervalo de confianza al 95% es : (2.317, 2.546)"
```

1.4.2 Enfoque bayesiano

Dado que en Baysesiana trabajamos una vez con los datos dados es más fácil agarrar la info y hacer la transformación.

```r
df_posterior <-
  df_posterior %>% dplyr::mutate(tau_bayes = log(sqrt(post_samples)))

a <- ggplot(df_posterior, aes(x = tau_bayes)) +
  geom_histogram(aes(x = tau_bayes, y = ..density..), bins = 30,
                 fill = "#F8766D",
                 alpha=0.5) +
  geom_vline(xintercept = log(sqrt((9848.82/74.05))), color = "red", linetype = "dashed") +
  annotate("text", x = log(sqrt((9848.82/74.05))), y = Inf, label = "2.4452",
           vjust = 1, hjust = 0.5, colour = "red")+
  labs(title = "Simulaciones de posterior",
       subtitle = "InvGamma(75.05,9848.82)") +
  xlab("Tau") + ylab("Densidad") +
  xlim(0,4)+
  ggpubr::theme_pubclean(base_size = 8)

b <- ggplot(reps_boot, aes(x = tau_boot)) +
  geom_histogram(aes(x = tau_boot, y = ..density..), bins = 30,
                 fill = "#F8766D",
                 alpha=0.5) +
  geom_vline(xintercept = 2.4387, color = "red", linetype = "dashed") +
  annotate("text", x = 2.4387, y = Inf, label = "2.4387",
           vjust = 1, hjust = 0.5,colour = "red")+
  labs(title = "Replicaciones bootstrap", subtitle = "MLE: Tau") +
  xlab("Tau") + ylab("Densidad") +
  xlim(0,4)+
  ggpubr::theme_pubclean(base_size = 8)

a+b
```

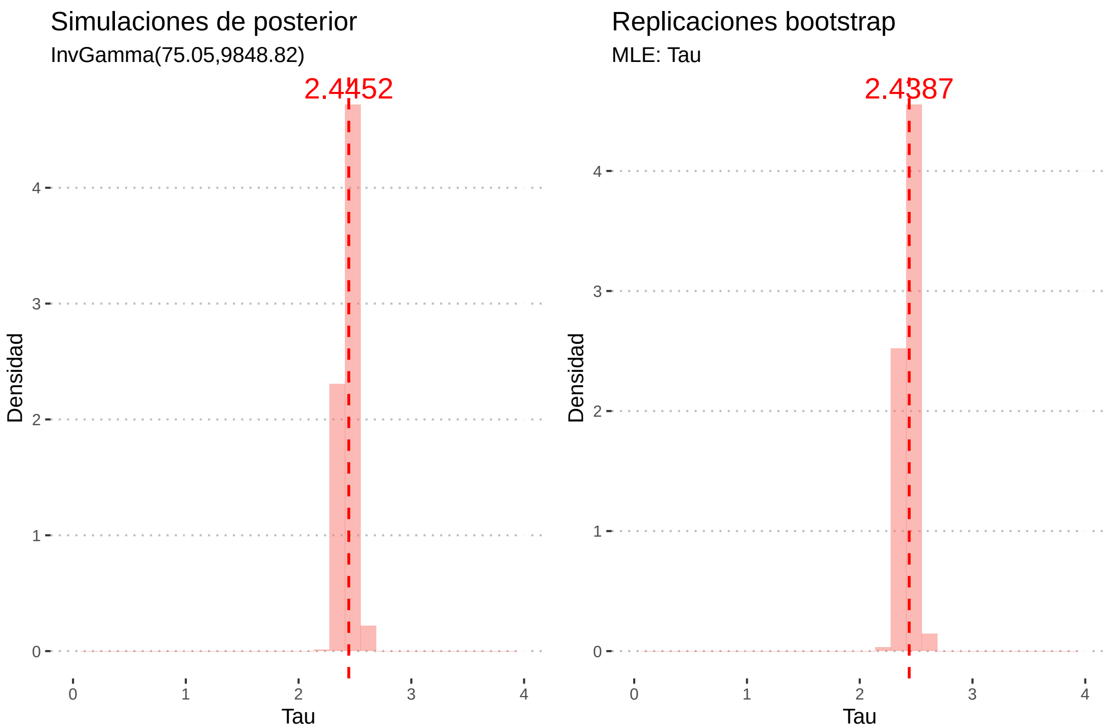

Figure 6: Comparativo Bayesiano vs. Bootstrap (Tau)

El intervalo de credibilidad para *tau* es:

```
paste0("Intervalo dist. posterior: (",
       round(log(sqrt(1/qgamma(0.975, alpha_post, beta_post))),3),",",
       round(log(sqrt(1/qgamma(0.025, alpha_post, beta_post))),3),")")
```

```
## [1] "Intervalo dist. posterior: (2.331,2.558)"
```