Desafio1

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE

02/22/2025 CALI - COLOMBIA

MAESTRIA EN INTELIGENCIA ARTIFICIAL Y CIENCIA DE DATOS

INFERENCIA ESTADISTICA

ENTREGA: DESAFÍO 1

- **Profesor:** Cristian E García
- **Alumno:** Yoniliman Galvis Aguirre
- **Código:** 22500214
- **Repositorio:** https://github.com/ygalves/Master-inferencia-estadistica.git (https://github.com/ygalves/Master-inferencia-estadistica.git)

Preparar el Sistema

Es necesario actualizar ó instalar Algunas librerias en RStudio, debido a que teste trabajo fúe realizado usando Linux Ubuntu 22.04, se presentaron algunos Issues que se solucionaron borrando algunos archivos de librerías, reinstalandolos y tomado desiciones al instalar el paquete dplyr el cual es un compendio de librerías y algunas de ellas estan usando funciones que comparten un nombre comun y con lo cual se genera fallas al tratar de cargarlas en el sistema.

```
# Variable de control para habilitar o deshabilitar la eliminación de archivos, si tiene problemas para instalar un paqute puede que ayude borrar estos archivos , primero ejecuta este Chunk en FALSE, si hay problemas llevalo a TRUE y ejecuta este Chunk de nuevo. ES probable que tenga que instalar las siguientes dependencias uasndo los siguientes comandos en uan terminal:
# sudo apt-get update
# sudo apt-get install libharfbuzz-dev libfribidi-dev libcurl4-openssl-dev libssl-dev libxml2-dev libfontconfig1-dev libfreetype6-dev libpng-dev libtiff5-dev libjpeg-dev

eliminar_archivos_habilitado <- FALSE

# Definir los archivos a eliminar
archivos <- c(".Rhistory", ".RData", ".Rprofile")

# Función para eliminar los archivos si existen
eliminar_archivos <- function(archivos) {
  for (archivo in archivos) {
    if (file.exists(archivo)) {
      file.remove(archivo)
      cat("Archivo eliminado:", archivo, "\n")
    } else {
      cat("Archivo no encontrado:", archivo, "\n")
    }
  }
}

# Eliminar los archivos si se habilitó la opción
if (eliminar_archivos_habilitado) {
  eliminar_archivos(archivos)
} else {
  cat("La eliminación de archivos está deshabilitada.\n")
}
```

```
## La eliminación de archivos está deshabilitada.
```

```
# Instalar y cargar el paquete para manejo de conflictos
install.packages("conflicted")
```

```
## Installing package into '/home/ygalvis/R/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
library(conflicted)

# Preferir funciones específicas de paquetes que estan en conflicto en la librería tidyverse purrr y tidyr las cu
ales tienen funciones con nombres iguales "stats" y "caret". así que vamos adefinir cual de estas funciones vamos
a preferir

conflict_prefer("filter", "dplyr")
```

```
## [conflicted] Will prefer dplyr::filter over any other package.
```

```
conflict_prefer("lag", "dplyr")
```

```
## [conflicted] Will prefer dplyr::lag over any other package.
```

```
conflict_prefer("lift", "purrr")
```

```
## [conflicted] Will prefer purrr::lift over any other package.
```

```
# Para hacer isntalacion de varios paquetes creamos un vector que contenga los nombre de los paquetes que queremo
s instalar
paquetes <- c("dplyr", "ggplot2", "caret", "ModelMetrics", "stats4", "tidyverse","rlang","tidyr","gridExtra","pro
gress","stats4","knitr")

# hacemos una Función que nos permite instalar paquetes si no están ya instalados en el sistema
instalar_paquetes <- function(paquetes) {
  paquetes_instalados <- paquetes[paquetes %in% installed.packages()[,"Package"]]
  nuevos_paquetes <- paquetes[!(paquetes %in% installed.packages()[,"Package"])]

  if(length(nuevos_paquetes)) {
    install.packages(nuevos_paquetes, quiet = TRUE)
    cat("Se instalaron los siguientes paquetes:", nuevos_paquetes, "\n")
  } else {
    cat("Todos los paquetes ya están instalados.\n")
  }

  if(length(paquetes_instalados)) {
    cat("Los siguientes paquetes ya estaban instalados:", paquetes_instalados, "\n")
  }
}

# Instala los paquetes que son necesarios
instalar_paquetes(paquetes)
```

```
## Todos los paquetes ya están instalados.
## Los siguientes paquetes ya estaban instalados: dplyr ggplot2 caret ModelMetrics stats4 tidyverse rlang tidyr g
ridExtra progress stats4 knitr
```

```
# Cargar los paquetes
library(dplyr)
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(ModelMetrics)
library(stats4)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ lubridate 1.9.4     ✔ tibble    3.2.1
## ✔ purrr     1.0.4     ✔ tidyr     1.3.1
## ✔ readr     2.1.5
```

```
library(rlang)
library(progress)
### grid Extra esta deshabilitada ya que causa problemas con pivot_longer
### library(gridExtra)
library(tidyr)
library(stats4)
library(knitr)
```

SITUACIÓN 1

Para estimar la proporción desconocida $\pi(0 < \pi \leq 0,50)$ de una determinada especie de peces que habita en el océano Pacífico, se aplicará el siguiente plan de muestreo. Cada uno de $n(> 1)$ barcos pesqueros capturará peces hasta capturar exactamente $k(> 1)$ peces de la especie particular de interés, y se registrará el número total de peces capturados por cada barco pesquero. Todos los peces capturados se devolverán ilesos al océano Pacífico. Para resolver dicho problema se le pide investigar 4 formas diferentes de estimar dicha proporción y comparar el desempeño de los resultados.

Plantee tres escenarios diferentes para llevar a cabo dichas estimaciones, compare para diferentes tamaños de muestra y concluya teniendo en cuenta las medidas de desempeño como el ECM

Entendiendo el Problema

- En el plan de muestreo se capturará peces de forma aleatoria, cada captura tiene sólo dos posibles resultados, el pez capturado puede ser o no de la especie de interés, por tanto se obseva que el muestreo tiene un comportamiento binomial, el éxito de cada captura tiene una probabilidad π y el fracaso $(1 - \pi)$. En estadística se llama a este tipo de muestreos como Ensayo de Bernoulli[1].
- Cada barco en el plan de muestreo va ha capturar peces hasta que se completen $k(> 1)$ éxitos o sea numero de peces de la especie que nos interesa, mientras no se alcance ese numero de éxitos, cada barco debe repetir el ensayo de bernoulli hasta completar la meta establecida.
- El número total de muestras ó capturas necesarias para cumplir con los exitos solicitados es una variable aleatoria desconocida X.
- Según la **teoría de la probabilidad**[2], el número de ensayos necesarios para conseguír k exitos en ensayos independientes de Bernoulli y con una probabilidad de éxito que permanece constantesigue una **Distribución negativa Binomial**[3]

La funcion de probilidad pra la distibución negativa binomial es:

$$P(X = x) = \binom{k-1}{x-1}\pi^{k}(1-\pi)^{x-k},\ x = k, k+1, k+2, \ldots$$

Ya que para el $x - ésima$ captura sea el especimen $k - ésimo$ correctO, teniendo que $k - 1$ éxitos en los $x - 1$ capturas previas y luego una captura correcta en el $x - ésima$ captura

Dónde:

$$Esperanza = \mathbb{E}[X] = \frac{k}{\pi}$$

$$Varianza = Var(X) = \frac{k(1-\pi)}{\pi^{2}}$$

- Es necesario definir el valor real de la proporción de peces en el oceano de la especie buscada (π) y a este se le denominará como $\pi_{verdadero}$, ese valor que ya se conoce como correcto nos va ha permitir comparar los estimadores en la simulaciones ó en los diferentes escenarios propuestos y luego comparar los resultados con el valor real para saber la precisión y consistencia de los diferentes estimadores.
- Si no conocemos a $\pi_{verdadero}$ y es lo que realmente sucede en la vida real, para las simulaciones establecemos el valor a un estimado para poder realizar los análisis necesarios a los estimadores o en su defecto se evaluan los estimadores usando tecnicas diferentes a ECM tal como usar la varianza, usar los intervalos de confianza, estudios de remuestreo, etc.
- Los estudios de simulacion donde se estima a$\pi_{verdadero}$ nos permiten evaluar el comportamiento de los estimadores bajo condiciones controladas para verificar la fiabilidad y robustez para poder elegir un método apropiado para aplicarlo despues a los datos reales y evitar fallas posteriores.
- Debemos usar un prior que es básicamente la distribucion de probabilidad que tenemos antes de realizar la observacion de datos, o sea es básicamente una estiamción de la probailidad que esperamos del muestreo antes de que se realice y se analisen los datos.

- El Prior es fundamental para el teorema de bayes donde se combina con la función de verosimilitud para obtener la distribucion posterior:

$$Teorema\ de\ Bayes = Posterior\ \alpha\ Verosimilitud\ x\ Prior$$

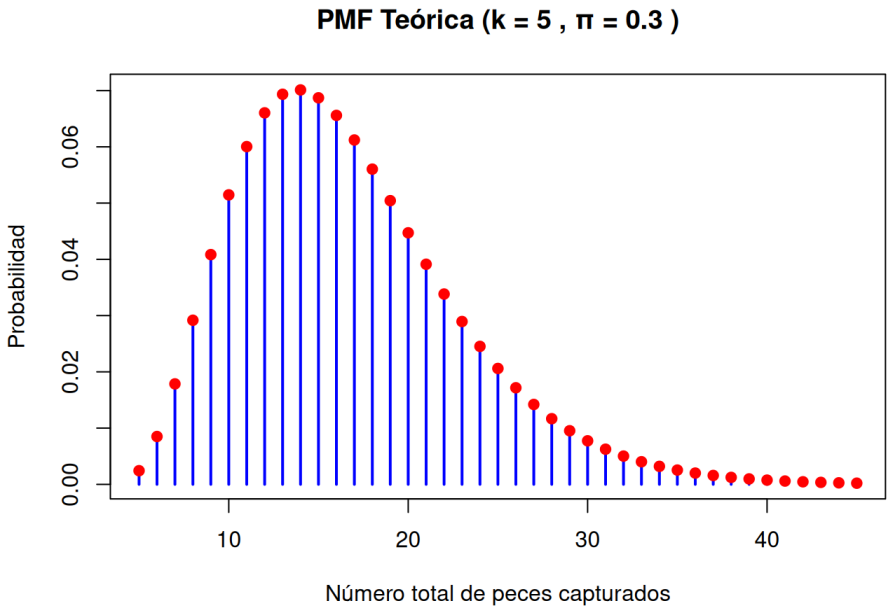
En este ejercicio, creemos que la proporción está entre 0 y 0.5 y no se inclina a favor de ningun valor particular y podemos utilizar un prior uniforme en intervalo $(0, 0.5)$ y de esta forma inicialmente consideramos que todos los valores en ese rango son probables.

Solución

Propongo cuatro estimadores para π:

1. Estimador MLE, de la mediana muestral ó Momentos: $\hat{\pi}_{MLE} = \frac{k}{\bar{X}}$
2. Estimador basado en transformación logarítmica: $\hat{\pi}_{log} = exp\left\{ \frac{1}{n} \sum_{i=1}^{n} ln\frac{k}{X_i} \right\}$
3. Estimador de Bayes ó bayesiano: asumiremos que es probable cualquier valor de π
4. Estimador insesgado de varianza mínima - (Uniformly Minimum-Variance Unbiased Estimator): $\hat{\pi}_{UMVUE} = \frac{nk-1}{\sum_{i=1}^{n} X_i - 1}$

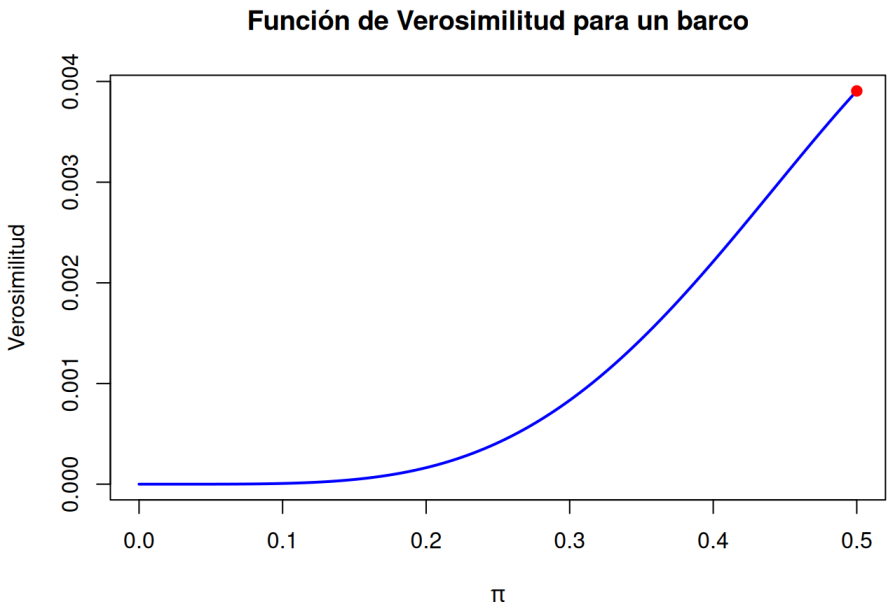
```
################################################################################
# Clase: Estimadores y Distribuciones Muestrales para la estimación de π
################################################################################

### 1. SIMULACIÓN DEL PROCESO DE CAPTURA DE PECES
# Cada barco captura peces hasta lograr k éxitos (peces de la especie de interés).
# El total de peces capturados sigue una distribución negativa binomial:
# total = k (éxitos) + número de fracasos.
# vamos a simular la distribución de los peces capturados por un barco

pi_verdadero <- 0.3     # Valor real de π (0 < π ≤ 0.5)
set.seed(1234)

# Para ver el comportamiento , usamos un primer escenario con k = 5 capturas requeridas.
k <- 5  # Número de éxitos requeridos por barco

# Definirmos el rango de totales (mínimo: k, máximo: k+40)
total_capturas <- k:(k + 40)

# Calculamos la PMF teórica:
# dnbinom(x, size = k, prob = pi_verdadero) calcula la probabilidad de x fracasos.
pmf_teorica <- dnbinom(total_capturas - k, size = k, prob = pi_verdadero)

# Graficamos la PMF teórica para este escenario
plot(total_capturas, pmf_teorica, type = "h", lwd = 2, col = "blue",
     main = paste("PMF Teórica (k =", k, ", π =", pi_verdadero, ")"),
     xlab = "Número total de peces capturados", ylab = "Probabilidad")
points(total_capturas, pmf_teorica, pch = 19, col = "red")
```



```
################################################################################
# 2. ESTIMACIÓN DE π POR MÁXIMA VEROSIMILITUD (MLE) CON stats4
################################################################################

# Simulamos los datos para un escenario concreto:
# n barcos capturan peces hasta lograr k éxitos.
n <- 20  # Número de barcos
datos <- rnbinom(n, size = k, prob = pi_verdadero) + k

# Función de log-verosimilitud (ignorando constantes):
# log L(π) = n*k*log(π) + (sum(datos) - n*k)*log(1-π)

negLogLik <- function(pi) {
  - ( n * k * log(pi) + (sum(datos) - n * k) * log(1 - pi) )
}

# Valor inicial: π0 = k/mean(datos)
pi_inicial <- k / mean(datos)

# Estimación por MLE imponiendo límites (0, 0.5]
resultado_mle <- mle(negLogLik, start = list(pi = pi_inicial),
                     method = "L-BFGS-B", lower = 0.0001, upper = 0.5)
summary(resultado_mle)
```

```
## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = negLogLik, start = list(pi = pi_inicial), method = "L-BFGS-B",
##     lower = 1e-04, upper = 0.5)
##
## Coefficients:
##     Estimate Std. Error
## pi 0.3378378 0.02749082
##
## -2 log L: 378.6378
```

```
############################################
#   1. Función de Verosimilitud (Ejemplo)    #
############################################

# Para una única observación (un barco)
k <- 5                  # Número de capturas requeridas
x_obs <- 8              # Total de peces capturados (incluye k éxitos)
p <- seq(0, 0.5, length.out = 1000)  # π varía entre 0 y 0.5
# La verosimilitud (sin incluir el coeficiente combinatorio)
verosimilitud <- p^k * (1 - p)^(x_obs - k)
plot(p, verosimilitud, type = "l", lwd = 2, col = "blue",
     main = "Función de Verosimilitud para un barco",
     xlab = expression(pi), ylab = "Verosimilitud")
pi_est <- p[which.max(verosimilitud)]
points(pi_est, max(verosimilitud), col = "red", pch = 19)
```



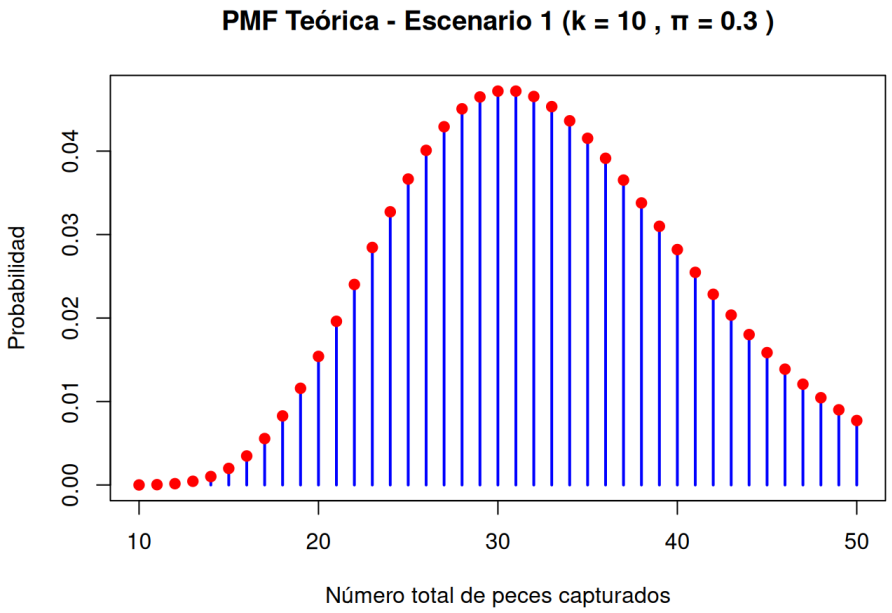

```
cat("Estimación MLE (gráfica) para un barco:", pi_est, "\n\n")
```

```
## Estimación MLE (gráfica) para un barco: 0.5
```

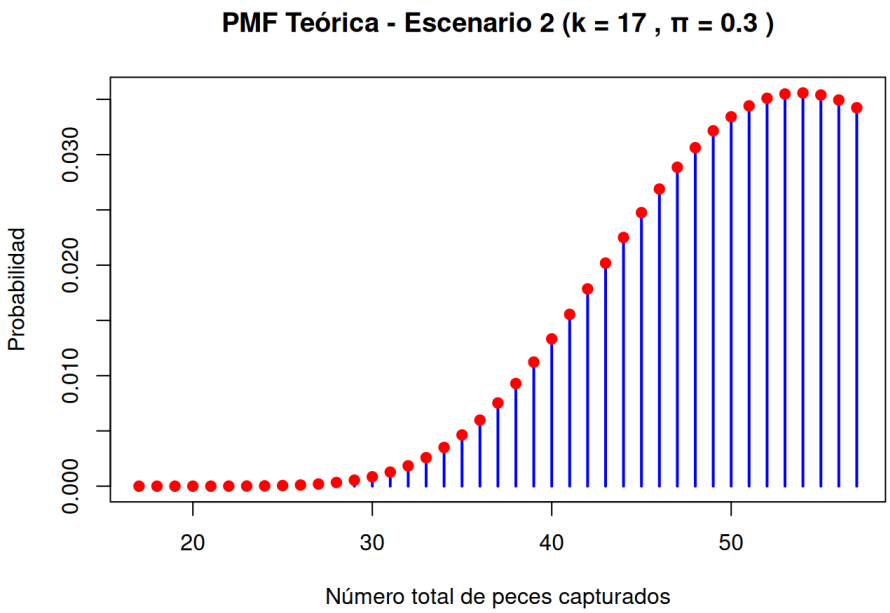
```
################################################################################
# 3. COMPARACIÓN DE ESTIMADORES MEDIANTE USO DE ESCENARIOS DE SIMULACIÓN
################################################################################

# Se consideran 4 estimadores para π:
# T_MLE:       π^ = k / media de capturas
# T_Log:       Estimador basado en transformación logarítmica
# T_Bayes:     Estimador Bayesiano (media del posterior con prior uniforme en (0,0.5))
# T_UMVUE:     Estimador insesgado (UMVUE)

# Definir las funciones estimadoras:
T_MLE <- function(x, k) {
  k / mean(x)
}

T_Log <- function(x, k) {
  exp(mean(log(k / x)))
}

# Calculo de la media del posterior para un prior uniforme en (0,0.5)
media_posterior <- function(sum_x, n, k) {
  post_density <- function(pi) { 2 * pi^(n * k) * (1 - pi)^(sum_x - n * k) }
  norm_const <- integrate(post_density, lower = 0, upper = 0.5)$value
  num_int <- integrate(function(pi) pi * post_density(pi), lower = 0, upper = 0.5)$value
  num_int / norm_const
}

T_Bayes <- function(x, k) {
  media_posterior(sum(x), length(x), k)
}

T_UMVUE <- function(x, k) {
  (length(x) * k - 1) / (sum(x) - 1)
}

# Crear los tres escenarios con diferentes valores de n (barcos) y k (capturas requeridas):
escenarios <- list(
  "Escenario 1" = list(n = 15, k = 10),
  "Escenario 2" = list(n = 30, k = 17),
  "Escenario 3" = list(n = 75, k = 30)
)

B <- 5000  # Número de simulaciones que usaremos para evaluar el ECM
resultados_ECM <- list()   # Lo usaremos Para guardar el ECM de cada estimador en cada escenario

# Data frame para guardar todas las estimaciones individuales y usarlas en graficas o tablas finales
estimaciones_total <- data.frame()

# Loop para la simulacion de los escenarios
for (esc in names(escenarios)) {
  n <- escenarios[[esc]]$n
  k <- escenarios[[esc]]$k

  # Guardamos las estimaciones de cada simulación
  simulaciones <- matrix(NA, nrow = B, ncol = 4)
  colnames(simulaciones) <- c("MLE", "Logaritmo", "Bayes", "UMVUE")

  for (i in 1:B) {
    # Ahora para cada barco simulamos el proceso: rnbinom(n, size=k, prob=π) y luego retornamos los fracasos,
    # Y sumando k obtenemos el total de peces capturados.
    datos_sim <- rnbinom(n, size = k, prob = pi_verdadero) + k
    simulaciones[i, "MLE"]       <- T_MLE(datos_sim, k)
    simulaciones[i, "Logaritmo"] <- T_Log(datos_sim, k)
    simulaciones[i, "Bayes"]     <- T_Bayes(datos_sim, k)
    simulaciones[i, "UMVUE"]     <- T_UMVUE(datos_sim, k)
  }

  # Obtener el ECM para cada estimador en el escenario actual (i)
  ECM <- colMeans((simulaciones - pi_verdadero)^2)
  resultados_ECM[[esc]] <- ECM
  cat("\n", esc, "\n")
  print(ECM)

  # Para la grafica final organizamos las simulaciones
```

```
  simulaciones_df <- as.data.frame(simulaciones)
  simulaciones_df$Simulacion <- 1:B
  simulaciones_df$Escenario <- esc
  simulaciones_long <- pivot_longer(simulaciones_df,
                                    cols = c("MLE", "Logaritmo", "Bayes", "UMVUE"),
                                    names_to = "Estimador", values_to = "Valor")
  estimaciones_total <- rbind(estimaciones_total, simulaciones_long)

  # Graficamos la PMF teórica para el escenario (i)
  totales <- k:(k + 40)
  pmf_teorica <- dnbinom(totales - k, size = k, prob = pi_verdadero)
  plot(totales, pmf_teorica, type = "h", lwd = 2, col = "blue",
       main = paste("PMF Teórica -", esc, "(k =", k, ", π =", pi_verdadero, ")"),
       xlab = "Número total de peces capturados", ylab = "Probabilidad")
  points(totales, pmf_teorica, pch = 19, col = "red")
}
```

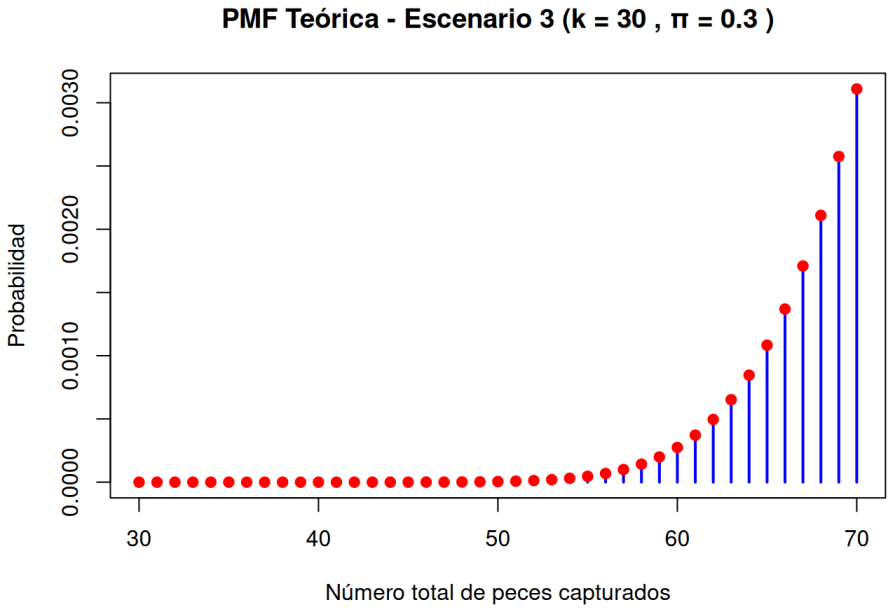
```
##
##  Escenario 1
##          MLE    Logaritmo        Bayes        UMVUE
## 0.0004095252 0.0005640463 0.0004115672 0.0004054495
```




```
##
##  Escenario 2
##          MLE    Logaritmo        Bayes        UMVUE
## 0.0001243965 0.0001704630          NaN 0.0001240126
```




```
##
##  Escenario 3
##          MLE    Logaritmo        Bayes        UMVUE
## 2.922231e-05 4.175261e-05          NaN 2.922930e-05
```



```
##############################################################################
# 4. TABLA RESUMEN DEL ECM (ERROR CUADRÁTICO MEDIO)
##############################################################################

ECM_tabla <- data.frame(
  Escenario   = names(resultados_ECM),
  n           = sapply(names(resultados_ECM), function(x) escenarios[[x]]$n),
  k           = sapply(names(resultados_ECM), function(x) escenarios[[x]]$k),
  MLE         = sapply(resultados_ECM, function(x) { if("MLE" %in% names(x)) as.numeric(x["MLE"]) else NA }),
  Logaritmo   = sapply(resultados_ECM, function(x) { if("Logaritmo" %in% names(x)) as.numeric(x["Logaritmo"]) els
e NA }),
  Bayes       = sapply(resultados_ECM, function(x) { if("Bayes" %in% names(x)) as.numeric(x["Bayes"]) else NA }),
  UMVUE       = sapply(resultados_ECM, function(x) { if("UMVUE" %in% names(x)) as.numeric(x["UMVUE"]) else NA }),
  stringsAsFactors = FALSE
)

if(requireNamespace("knitr", quietly = TRUE)){
  knitr::kable(ECM_tabla,
               caption = "ECM de cada estimador por escenario (n es el núm. de barcos y k es el núm. de capturas
requeridas).")
} else {
  print(ECM_tabla)
}
```

ECM de cada estimador por escenario (n es el núm. de barcos y k es el núm. de capturas requeridas).

	Escenario	n	k	MLE	Logaritmo	Bayes	UMVUE
Escenario 1	Escenario 1	15	10	0.0004095	0.0005640	0.0004116	0.0004054
Escenario 2	Escenario 2	30	17	0.0001244	0.0001705	NaN	0.0001240
Escenario 3	Escenario 3	75	30	0.0000292	0.0000418	NaN	0.0000292

Conclusiones

En el ejercicio propuse 4 etimadores: el de mediana muestral o momentos (MLE) y su version insesgada UMVUE o de varianza mínima, el de transformación Logaritmica y el Bayesiano.

Se plantearon 3 escenarios diferentes y se realizáron las simulaciones respectivas. Obsevamos que segun el ECM:

- Los estimadores basados en la media muestral (MLE y UMVUE) son los de mejor desempeño.
- Los estimadores basados en la media muestral (MLE y UMVUE) se comportan de manera muy similar.
- Los estimadores basados en la media muestral (MLE y UMVUE) mejoran en consistencia a medida que las muesras k aumentan.
- El estimador basado en transformación logarítmica mejora igual a medida que las muestras sonmas grandes pero mantiene un ECM superior y eso significa menor eficiencia.
- El estimador bayesiano no presenta resultados en todos los escenarios y puede signiicar que necesita mayor verificacion o una seleccion del pior nueva o que este estimador no es suficientemente robusto para este muestreo en particular.
- La verosimilitud teórica produce un estimador $\hat{\pi}$ y es equivalente al MCE del estimador MLE ya que $MLE = \frac{k}{\bar{x}} = \hat{\pi}$

SITUACIÓN 2

Suponga que se tiene una muestra aleatoria de tamaño $2n$ tomada de una población X, con $E(X) = \mu$ y $Var(X) = \sigma^2$, Sean:

$$\overline{X}_1 = \frac{1}{2n}\sum_{i=1}^{2n} x_i \quad \text{y} \quad \overline{X}_2 = \frac{1}{n}\sum_{i=1}^{n} x_i$$

dos estimadores de μ.

- ¿Cuál es le mejor estimador de μ?
- Simule una situación con 1000 muestras de tal forma que se pueda evidenciar de manera gráfica cual de los dos estimados es mejor.

Entendiendo el Problema

- Tenemos una poblacion X la cual tiene una media dad por la funcion $E(X) = \mu$ y una varianza $Var(X) = \sigma^2$.
- Como el ejercicio no hace mencion a la distribución de esta población X, asumimos una distribucion normal[4].

- De dicha población vamos a extraer una muestra aleatória cuyo tamaño es $2n$.
- De esta muestra consideramos dos estimadores para calcular la media μ:
    1. El primer estimador de $\overline{X}_1$ el cual va ha utilizar todos los datos de la muestra ($2n$ datos):

$$\overline{X}_1 = \frac{1}{2n}\sum_{i=1}^{2n} x_i$$

    2. El segundo estimador de $\overline{X}_2$ el cual va ha utilizar sólo los datos de la primera mitad de la muestra (n datos):

$$\overline{X}_2 = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- Tenemos que determinar cual es el mejor estimador de μ
    1. Ambos estimadores son insesgados ya que: $E(\overline{X}_1) = E(\overline{X}_2) = \mu$.
    2. Vamos a medir el desepeño usando la varanza de cada uno de los dos estimadores, donde la Varianza de $\overline{X}_1$ es: $Var(\overline{X}_1) = \frac{\sigma^2}{2n}$ y la Varianza de $\overline{X}_2$ es: $Var(\overline{X}_2) = \frac{\sigma^2}{n}$
    3. Podemos estimar entonces por observación del divisor que el estimador de $Var(\overline{X}_1)$ es menor y entonces debe ser mas preciso, y por tanto debe de tener menor varianza, entonces **el estimador $\overline{X}_1$ es el mejor estimador de μ.**

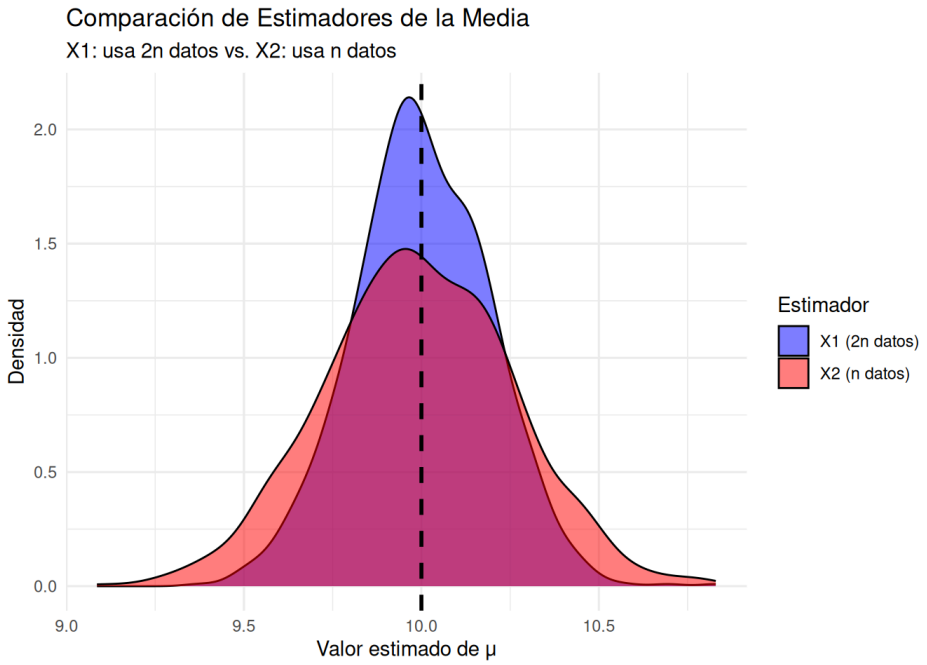
```
################################################################################
# Estimamos de la Media de una Población X con tamaño 2n
# Y Comparamos los dos estimadores:
#   X1 = (1/(2n)) * Σ x_i (usa 2n datos)
#   X2 = (1/n)  * Σ x_i (usa solo los primeros n datos)
################################################################################

# Parámetros de la simulación
set.seed(123)               # La semilla para grantizar la reproducibilidad
n <- 50                     # tamaño de la muestra
N <- 2 * n                  # Tamaño total de la muestra
mu <- 10                    # Media de la poblacion
sigma <- 2                  # Desviación estándarde la poblacion
replicaciones <- 1000       # Número de muestras para la simulacion

# Creamos los vectores para almacenar los estimadores
X1 <- numeric(replicaciones) # Estimador que usa 2n datos (todos)
X2 <- X1  # Estimador que solo usa el primer set de datos

# Simulamos y en cada réplica extraemos una muestra de tamaño 2n de la población
for(i in 1:replicaciones){
  sample_data <- rnorm(N, mean = mu, sd = sigma)
  X1[i] <- mean(sample_data)        # Usa todos los 2n datos
  X2[i] <- mean(sample_data[1:n])     # Usa solo los primeros n datos
}

# Calculamos las varianzas empíricas (ECM, pues son insesgados)
ecm_X1 <- var(X1)   # Teóricamente: sigma^2/(2n) = (4)/(100) = 0.04
ecm_X2 <- var(X2)   # Teóricamente: sigma^2/n   = (4)/(50)  = 0.08

cat("Varianza (ECM) de X1 (2n datos):", ecm_X1, "\n")
```

```
## Varianza (ECM) de X1 (2n datos): 0.03616556
```

```
cat("Varianza (ECM) de X2 (n datos):", ecm_X2, "\n")
```

```
## Varianza (ECM) de X2 (n datos): 0.07072529
```

```
# Creamos un data frame para graficar las densidades de los dos estimadores
df <- data.frame(
  Estimador = factor(c(rep("X1 (2n datos)", length(X1)), rep("X2 (n datos)", length(X2)))),
  Valor = c(X1, X2)
)

# Gráficamos la comparativa entre las densidades de los estimadores y la línea vertical en mu
g <- ggplot(df, aes(x = Valor, fill = Estimador)) +
  geom_density(alpha = 0.5) +
  geom_vline(xintercept = mu, linetype = "dashed", color = "black", linewidth = 1) +
  labs(title = "Comparación de Estimadores de la Media",
       subtitle = "X1: usa 2n datos vs. X2: usa n datos",
       x = "Valor estimado de μ", y = "Densidad") +
  scale_fill_manual(values = c("blue", "red")) +
  theme_minimal()

print(g)
```




```
# Creamos una tabla de resumen mejorada usando knitr::kable
ECM_tabla <- data.frame(
  Estimador = c("X1 (2n datos)", "X2 (n datos)"),
  ECM = c(ecm_X1, ecm_X2)
)

# Mostramos la tabla
kable(ECM_tabla, caption = "ECM (Varianza empírica) de cada estimador")
```

ECM (Varianza empírica) de cada estimador

Estimador	ECM
X1 (2n datos)	0.0361656
X2 (n datos)	0.0707253

Conclusiones

- El análisis en la gráfica y la teoría nos demuestra que el mejor estimador de μ es $\overline{X}_1$ ya que este utiliza los $2n$ datos y presenta una varianza $\frac{\sigma^2}{(2n)}$ y esa es menor que la varianza de $\overline{X}_2 = \frac{\sigma^2}{(n)}$, porque esta solo utiliza la primera mitad de lso datos n.
- Se demuestra entonces de forma teórica y de forma gráfica que ql estimador de mejor desempeño será siempre aquel que utiliza toda la muestra de datos y no solo una parte de ella.

SITUACIÓN 3

En una población hay un número θ de vehículos informales (llamados “piratas”), que es desconocido. Supongamos que los piratas, están numerados visiblemente en forma consecutiva: $1, 2, 3, \ldots, \theta$. Con el propósito de estimar θ , usted registra una muestra aleatoria de n piratas y anotando cada vez el número X correspondiente. Así dispone de una muestra aleatoria: $X1, X2 \ldots, Xn$. Existen varias propuestas razonables de estimadores para el número total θ de taxis, como las que se describen a continuación y que surgen de la consideración de que la distribución de la variable aleatoria X (número del carro), es uniforme discreta. Los estimadores propuestos son los siguientes:

$$\hat{\theta}_{(1)} = 2\overline{X}_n - 1$$

$$\hat{\theta}_{(2)} = X_{(n)} + X_{(1)} - 1$$

$$\hat{\theta}_{(3)} = X_{max} + \bar{d}$$

Donde $\bar{d}$ es la media de los saltos consecutivos de los números de la muestra al ser ordenados en forma creciente. (Desarrolle $\bar{d}$ y se simplificará la expresión)

$$\hat{\theta}_{(4)} = 2Me(X) - 1$$

$$\hat{\theta}_{(5)} = \overline{X} + 3S$$

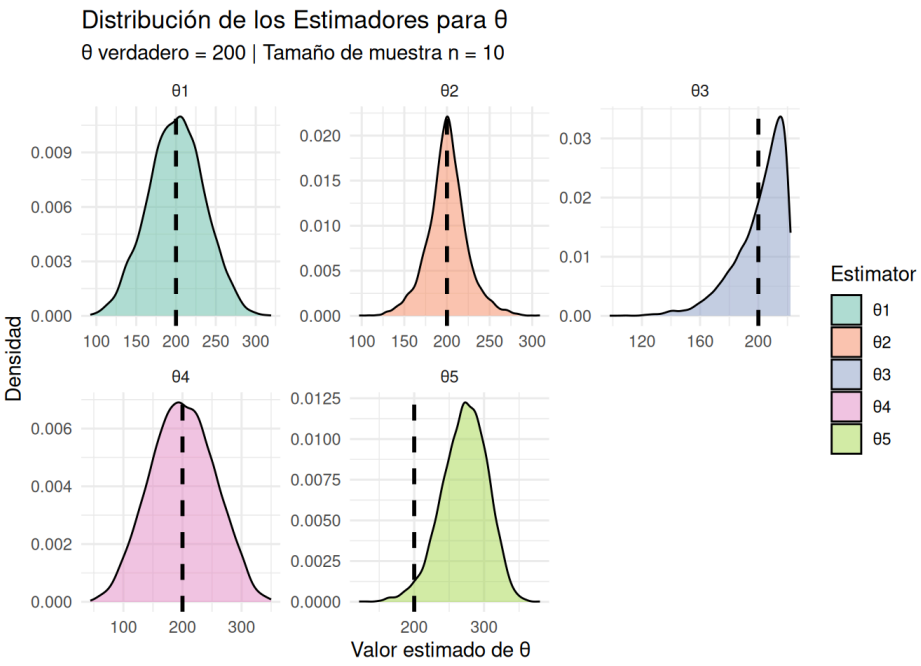
Simule el comportamiento de cada uno de estos estimadores, R o cualquier otra opción tecnológica, variando el valor del parámetro y el tamaño de muestra n . Nos interesa compararlos especialmente en cuanto al Sesgo, la Varianza de los estimadores y también el Error Cuadrático Medio (ECM) y cualquier otro indicador que le parezca razonable.

Entendiendo el Problema

- Dada una población de vehiculos tenemos una cantidad desconocida θ de vehiculos que hacen acarreo de personas de forma informal (“piratas”).
- Asumimos que los vehiculos estan marcados con numeros consecutivos así: $1, 2, 3, \ldots, \theta$.
- Asumimos que si extraemos una muestra aleatória de θ seguirá una **distribución uniforme discreta**[5] sobre el conjunto $\{1, 2 \ldots, \theta\}$. Esto porque:
    1. Los vehículos estan numerados de forma consecutiva $1, 2, 3, \ldots, \theta$. Con lo cual cada número entre 1 y θ aparece solo una vez.
    2. Todos los vehículos en la población de interés tiene la misma probabilidad de ser seleccionado al tomar una muestra aleatória, como la numeración es consecutiva y cada número es único, la probabilidad de tomar cualquier número x con $(1 \leq x \leq \theta)$ igual para todos los sujetos y sin favorecer a ninguno, o sea $P(X = x) = \frac{1}{\theta} \quad para\ todo\ x\ en \{1, 2, \ldots, \theta\}$
    3. **distribución uniforme discreta -** $U(a, b)$ es aquella en la que cada uno de los valores posibles tienen la misma probabilodad de ocurrir.
    4. Tomando las estimaciones dadas en el ejercicio podemos derivar fórmulas basadas en las propiedades de la distribución uniforme directa, por ejemplo la media teórica de la distribución uniforme discreta es: $E(X) = \frac{\theta+1}{2}$, con esta funcion podemos fromular el estimador $\hat{\theta}_{(1)} = 2\overline{X}_n - 1$.
- Como θ es desconocido vamos a estimarlo tomando una muestra aleatória de n vehículos y se llevan apuntes de sus números $X_1, X_2, \ldots, X_n$.
- Hay que tener en cuenta que θ es el valor límite superior de la población de vehiculos informales, que es desconocido, la probabilidad de encontrarlo ó acercarnos a este, tomando sólo una muestra de esta población es baja si $n << \theta$
- Para el estimador $\hat{\theta}_{(1)}$ usamos la media muestral $\overline{X}_n$ ya que la media de una distribución uniforme discreta en $1, \ldots, \theta$ es $\frac{\theta+1}{2}$ y se puede despejar para θ.

```
################################################################################
# Estimación de θ (número total de vehículos piratas)
#
# La población consiste en vehículos numerados consecutivamente: 1,2,...,θ.
# Se extrae una muestra aleatoria de tamaño n (sin reemplazo) y se registran los
# números observados: X1, X2, …, Xn.
#
# Se proponen los siguientes estimadores:
#
#   (1)   θ^(1) = 2·X¯ - 1
#          (porque E(X) = (θ+1)/2 para X ~ Uniforme{1, …, θ})
#
#   (2)   θ^(2) = X_(n) + X_(1) - 1
#
#   (3)   θ^(3) = X_max + d¯, donde d¯ es la media de los saltos consecutivos.
#          Al ordenar la muestra: d¯ = (X_(n) - X_(1))/(n-1), de modo que
#          θ^(3) = X_(n) + (X_(n) - X_(1))/(n-1)
#
#   (4)   θ^(4) = 2·Med(X) - 1
#
#   (5)   θ^(5) = X¯ + 3·S
#
# Se simula el comportamiento de estos estimadores para diferentes escenarios
# variando el valor real de θ y el tamaño de muestra n.
################################################################################

#Instalamos la libreria que necesitamos pero no al principio porque causa fallas con tidyr y pivot_longer
library(gridExtra)

# Parámetros de simulación
set.seed(1234)
theta_true <- 200    # Valor real de θ (número total de vehículos)
n <- 10              # Tamaño de la muestra (n vehículos observados)
B <- 5000            # Número de réplicas de la simulación

# Inicializamos vectores para almacenar los valores de cada estimador
est1 <- numeric(B)   # Estimador (1): 2*mean - 1
est2 <- numeric(B)   # Estimador (2): min + max - 1
est3 <- numeric(B)   # Estimador (3): max + (max - min)/(n-1)
est4 <- numeric(B)   # Estimador (4): 2*median - 1
est5 <- numeric(B)   # Estimador (5): mean + 3*sd

# Simulamos para cada réplica y extraemos una muestra aleatoria de tamaño n de {1,2,...,θ}
for (i in 1:B) {
  muestra <- sample(1:theta_true, size = n, replace = FALSE)

  # Calculamos cada uno de los estimadores:
  # Estimador ajustado de la media, es una forma de ajustar la media muestral.
  est1[i] <- 2 * mean(muestra) - 1
  # Estimador basado en el rango. Utiliza los valores extremos de la muestra para hacer una estimación
  est2[i] <- min(muestra) + max(muestra) - 1
  # estimador ajustado del máximo. Ajusta el valor máximo de la muestra utilizando el rango y el tamaño de la mue
stra.
  est3[i] <- max(muestra) + (max(muestra) - min(muestra)) / (n - 1)
  # estimador ajustado de la mediana. Similar al primer estimador, ajusta la mediana muestral.
  est4[i] <- 2 * median(muestra) - 1
  # Estimador de la media ajustado por la desviación estándar. Utiliza tanto la media como la desviación estándar
de la muestra para hacer una estimación.
  est5[i] <- mean(muestra) + 3 * sd(muestra)
}

# Crear un data frame con los resultados de las simulaciones y convertir a formato largo
df <- data.frame(
  Replication = 1:B,
  `θ1` = est1,
  `θ2` = est2,
  `θ3` = est3,
  `θ4` = est4,
  `θ5` = est5
)

df_long <- pivot_longer(df, cols = -Replication,
                        names_to = "Estimator", values_to = "Value")
```

```
# Convertimos la variable Estimator a factor (usando los valores observados)
df_long$Estimator <- as.factor(df_long$Estimator)

# Graficamos las densidades de cada estimador (facetas) con la línea vertical en θ_true
p_dens <- ggplot(df_long, aes(x = Value, fill = Estimator)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ Estimator, scales = "free") +
  geom_vline(xintercept = theta_true, linetype = "dashed", color = "black", linewidth = 1) +
  labs(title = "Distribución de los Estimadores para θ",
       subtitle = paste("θ verdadero =", theta_true, "| Tamaño de muestra n =", n),
       x = "Valor estimado de θ", y = "Densidad") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2")
print(p_dens)
```

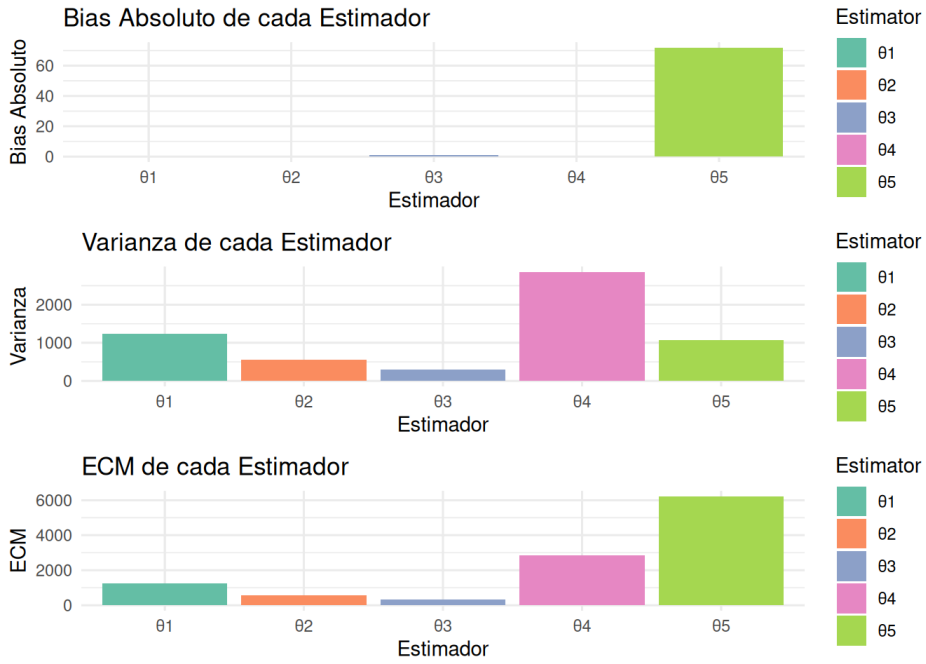



```
# Calculamos ahora el sesgo, varianza y ECM para cada estimador y construimos una tabla de resumen
summary_table <- df_long %>%
  group_by(Estimator) %>%
  summarise(
    Bias = round(mean(Value) - theta_true, 3),
    Variance = round(var(Value), 3),
    ECM = round((mean(Value) - theta_true)^2 + var(Value), 3)
  )

# Imprimimos la tabla resumen con knitr::kable
kable(summary_table, digits = 3, caption = "Resumen de los Estimadores: Sesgo, Varianza y ECM")
```

Resumen de los Estimadores: Sesgo, Varianza y ECM

Estimator	Bias	Variance	ECM
θ1	0.216	1234.534	1234.581
θ2	0.234	562.229	562.284
θ3	1.079	306.327	307.491
θ4	0.041	2856.683	2856.685
θ5	71.792	1064.247	6218.319

```
# Graficamos las barras comparativas para Bias (valor absoluto), Varianza y ECM
p_bias <- ggplot(summary_table, aes(x = Estimator, y = abs(Bias), fill = Estimator)) +
  geom_bar(stat = "identity") +
  labs(title = "Bias Absoluto de cada Estimador", y = "Bias Absoluto", x = "Estimador") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2")

p_variance <- ggplot(summary_table, aes(x = Estimator, y = Variance, fill = Estimator)) +
  geom_bar(stat = "identity") +
  labs(title = "Varianza de cada Estimador", y = "Varianza", x = "Estimador") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2")

p_ecm <- ggplot(summary_table, aes(x = Estimator, y = ECM, fill = Estimator)) +
  geom_bar(stat = "identity") +
  labs(title = "ECM de cada Estimador", y = "ECM", x = "Estimador") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2")

# Combinamos las tres gráficas en un panel vertical
grid.arrange(p_bias, p_variance, p_ecm, ncol = 1)
```



Conclusiones

- El indicador θ_4 presenta un sesgo cercano a cero (0.04) lo cual es el ideal pero la varianza es la mas alta de todos (2856.68) y nos indica que los valores estimados tienen variaciones altas entre una muestra y la otra.
- El indicador θ_5 presenta el sesgo mas elevado (71.79) y una varianza alta (1064.25) aunque la varianza está entre la media de todos el alto sesgo nos indica que no es un estimador muy adecuado para la implementacion.
- Los indicadores θ_1 y θ_2 presentan los sesgos mas bájos (0.22) y (0.23) respectivamente, pero la varianza de θ_1 es alta (1234.53) mientras la varianza de θ_2 es moderada pero no es la mas baja de entre todos los indicadores (562.23)
- El indicador θ_3 presenta un sesgo bajo y aunque no es el menor de todos los estimadores si es cercano a 0: (1.08) además si tiene el valor de varianza mas bajo de todo el set (306.33).
- El ECM es la sumatoria del cuadrado del sesgo y la varianza y refleja el desempeño o presicion global del estimador, tomando en cuenta esto el ECM más bajo es de θ_3 con (307.49) lo cual nos indica que a pesar de que el sesgo no es el mas bájo la combinacion de este y la poca varianza hacen este indicador como el mejor en precision global.
- Los estimadores $\theta_1, \theta_4 y \theta_5$ tiene ECMs muy altos $(1234.58, 2856.68, 6212.32)$ respectivamente y esto indica que no sin importar el bajo sesgo, la alta varianza los convierte en inadecuados.
- El estimador θ_2 tiene igualmente un ECM bajo (562.28), tiene tambien bajo sesgo (0.23) y una varianza moderada (562.23) pero el ECM es mayor que el ECM de θ_3 con (307.45), con lo cual concluimos que θ_2 es un buen indicador.

La mejor estimación de θ se logra usando el estimador $\hat{\theta_3} = X_{max} + \frac{X_{max} - X_{min}}{n-1}$, esto a que este presenta la mejor precision global (307.49) manteniendo un sesgo cercano a cero (1.08) y la varianza mas baja (306.33), ahora bien este desafio necesita estimar un valor desconocido de θ por tanto una baja varianza nos ayuda a estimar un valor que sea mas cercano al valor real de θ.

SITUACIÓN 4

Tiempos de atención entre llamadas de reclamaciones por seguros.

Sean $(X_1, \ldots, X_n)$ con densidad $\lambda e^{\lambda x}, (x \geq 0), (n \leq 2)$. Sea $S_n = \sum_{i=1}^{n} X_i)$. Es bien conocido que $Z = \lambda S_n$ tiene densidad:

$$f_z(z) = \frac{z^{n-1}e^{-z}}{(n-1)!},\ z \geq 0$$

* Utilice esto para calcular el sesgo y el ECM de $\hat{\lambda} = \frac{n-1}{S_n}$.

- Calcule el estimados de momentos y máximo verosímil para la función de densidad.

Entendiendo el Problema

- En un callcenter de una empresa de seguros, se necesita evaluar las propiedades de los estimadores de los tiempos de atención entre llamadas de reclamaciones para la densidad $f_z(z) = \frac{z^{n-1}e^{-z}}{(n-1)!},\ z \geq 0$
- Utilizando la transformación $Z = \lambda S_n$ se necesita calcular teoricamente el sesgo y el ECM del estimador dado: $\hat{\lambda} = \frac{n-1}{S_n}$.
- Se necesita calcular el estimador de momentos y el estimador de máxima verosimilitud para el ejercicio.
- El ejercicio indica que cada X_i tiene una densidad $f(x) = \lambda e^{\lambda x}, (x \geq 0)$, y dado que el estimador propuesto $\hat{\lambda} = \frac{n-1}{S_n}$ y ya que el objetivo es estimar los tiempos muertos entre las llamadas al call center, podemos deducir que en este ejercicio los datos a evaluar se distinguen como una **Distribución exponencial**[6]
- Para este análisis de este ejercicio vamos a utilizar las propiedades de la esperanza porque estamos iniciando desde que S_n tiene una distribución Gamma y por eso es que

  $$E\left(\frac{1}{S_n}\right) = \frac{\lambda}{n-1},$$

  Cuando despejemos esperamos demostrar que:

  $$E\left(\frac{n-1}{S_n}\right) = \lambda.$$

- Al despejar usando la esperanza podremos comprobar que el estimador $\hat{\lambda} = \frac{n-1}{S_n}$ es insesgado y que permite derivar la varianza y el ECM usando las propiedades de la **distribucion Gamma**

Propiedades de la distribucion Gamma (Γ)[7]

Las propiedades que voy a colocar acá permiten de forma general utilizar la distribución Gamma en diferentes aplicaciones, por ejemplo en el modelo de este ejercicio sobre tiempos de atención, análisis de superviciencia y en la suma de tiempos entre eventos en procesos $Poisson$. La relación $E\left[\frac{1}{S_n}\right] = \frac{\lambda}{(n-1)}$ que es muy usado en los problemas de estimación se fundamenta en las propiedades de Gamma y en especial en el cálculo de momentos de la distribucion Gamma:

1. **Función de Densidad de Probabilidad ó PDF** Si X sigue una distribucion gamma con parámertrós forma α y tasa β (a veces se usa la escala $\theta = \frac{1}{\beta}$ y se escribe:

   $$f(x, \alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x},\ x > 0,$$

   Donde $\Gamma(\alpha)$ es la función Gamma definida por:

   $$\Gamma(\alpha) = \int_0^{\infty} t^{\alpha-1} e^{-t}\, dt$$

2. **Función de Distribución acumulada ó CDF** la CDF se expresa en términos de la funcion de Gamma incompleta:

   $$F(x; \alpha, \beta) = \frac{1}{\Gamma(\alpha)} \gamma(\alpha, \beta x),$$

   dónde $\gamma(\alpha, \beta x)$ es la Gamma incompleta inferior.
3. **Cálculo de Momentos** a. La esperanza de X es:

   $$E[X] = \frac{\alpha}{\beta}.$$

   b. La varianza es:

   $$Var(X) = \frac{\alpha}{\beta^2}.$$

   c. Momento inverso ó expectativa del inverso, esperanza inversa ó propiedad clásica, esta es el resultado derivado de la propiedad de los

momentos y es la expecatativa del inverso de una variable Gamma, si $S \sim \Gamma(\alpha, \beta)\ y\ \alpha > 1$, se tiene que

$$E\left(\frac{1}{S}\right) = \frac{\beta}{\alpha - 1}.$$

4. **Función Generadora de Momentos o MGF** La función generadora de momentos de X está dad por:

$$M_x(t) = \left(1 - \frac{t}{\beta}\right)^{-\alpha},\ para\ t < \beta.$$

5. **Suma de Variables Gamma** Una propiedad muy útil es la suma de variables Gamma independientes, estas omparten el mismo parámetro de tasa de β, también tiene una distribución Gamma, así:

$$X_1 \sim\ \Gamma(\alpha_1, \beta) \quad y \quad X_2 \sim\ Gamma(\alpha_2, \beta),$$

entonces, pues tenemos que:

$$X_1 + X_2\ \sim\ \Gamma(\alpha_1 + \alpha_2, \beta).$$

6. **Relación con Otras Distribuciones** * **Distribución Exponencial:** Si $\alpha = 1$, la distribución Gamma se reduce a la distribución exponencial con parámetro β. * La distribución X^2 con ν grados de libertad es un caso especial de la Gamma porque:

$$\chi^2_\nu\ \sim\ \Gamma\left(\frac{\nu}{2}, \frac{1}{2}\right).$$

7. **Propiedad de Escalamiento** Si $X\,\Gamma(\alpha, \beta)\ y\ c > 0$, entonces la variable escalada cX tiene distribución:

$$cX\,\nu\,\Gamma\left(\alpha, \frac{\beta}{c}\right).$$

Solución

1. **Para calcular el sesgo del estimador** $\hat{\lambda} = \frac{n-1}{S_n}$

Ya que la propiedad de el momento inverso (ver punto 3.c de las propiedades Gamma): $S \sim \Gamma(\alpha, \beta)\ y\ \alpha > 1$, y asumimos que $\alpha = n\ y\ \beta = \lambda)$, deducimos:

Bueno conociendo que para $S_n\ \sim\ \Gamma(n, \lambda)$, tenemos que la propiedad es:

$$E(\frac{1}{S_n}) = \frac{\lambda}{n-1}$$

, asi que el valor esperado de el estimador es:

$$E(\hat{\lambda}) = E\left(\frac{n-1}{S_n}\right) = (n-1)E\left(\frac{1}{S_n}\right) = (n-1)\left(\frac{\lambda}{n-1}\right) = \lambda.$$

Por tanto, como $E(\hat{\lambda}) = \lambda$ podemos decir que el estimador $\hat{\lambda}$ es **insesgado**, porque el sesgo es la diferencia entre el valor esperado del estiamdor y el parámetro verdadero o sea:

$$Sesgo = E(\hat{\lambda}) - \lambda = 0$$

y como el sesgo en este caso es 0, entonces se puede decir que no existe ó es insesgado.

2. **Eror cuadrático medio ó ECM** Teniendo en cuenta que el $\hat{\lambda}$ es **insesgado**, entonces esperamos que $ECM = varianza$ y para obtener $Var(\hat{\lambda})$ tenemos que hallar la varianza de $\frac{1}{S_n}$ y sabemos que para $S_n \sim \Gamma(n, \lambda)\ \ n > 2$ y usamos 2 para exista una varianza, entonces:

$$* \quad E\left(\frac{1}{S_n}\right) = \frac{\lambda}{n-1'}$$

$$* \quad E\left(\frac{1}{S_n^2}\right) = \frac{\lambda^2}{(n-1)(n-2)}$$

Ahora :

$$Var\left(\frac{1}{S_n}\right) = E\left(\frac{1}{S_n^2}\right) - \left[E\left(\frac{1}{S_n}\right)\right]^2 = \frac{\lambda^2}{(n-1)(n-2)} - \left(\frac{\lambda}{n-1}\right)^2$$

$$Var\left(\frac{1}{S_n}\right) = \frac{\lambda^2}{(n-1)^2}\left(\frac{n-1}{n-2} - 1\right) = \frac{\lambda^2}{(n-1)^2}\left(\frac{(n-1)-(n-2)}{n-2}\right) = \frac{\lambda^2}{(n-1)^2(n-2)}.$$

Ahora como $\hat{\lambda} = (n-1)(\frac{1}{S_n})$, entonces tenemos:

$$Var(\hat{\lambda}) = (n-1)^2 Var\left(\frac{1}{S_n}\right) = (n-1)^2 . \frac{\lambda^2}{(n-1)^2(n-2)} = \frac{\lambda^2}{n-2}$$

Entonces como ya sabemos que ECM = Var tenemos que:

$$ECM(\lambda^2) = \frac{\lambda^2}{n-2}$$

3. **Estimador por el Momento y máxima verosimilitud** Ahora como el ejercicio es una distribución exponencial y $X \sim Exp(\lambda)$, tenemos que: $E[X] = \frac{1}{\lambda}$

- Y usando el metodo de momentos igualamos el primer momento teórico com el momengto muestral:

$$\overline{X} = \frac{1}{\lambda} \rightarrow \hat{\lambda}_{MM} = \frac{1}{\overline{X}}$$

- Aplicando la función de verosimilitud ó MLE para la muestra de $\{x_1, \ldots, x_n\}$ es:

$$L(\lambda) = \prod_{i=1}^{n} \lambda e^{-\lambda x_i} = \lambda^n \exp\left(-\lambda \sum_{i=1}^{n} x_i\right)$$

Y conociendo que el **logaritmo de la verosimilitud**[8] es:

$$\ell(\lambda) = n \ln \lambda - \lambda \sum_{i=1}^{n} x_i$$

Ahora tomamos el logaritmo de la verosimilitud y lo derivamos respetco a λ e igualando a cero:

$$\frac{d\ell}{d\lambda} = \frac{n}{\lambda} - \sum_{i=1}^{n} x_i = 0 \implies \hat{\lambda}_{MLE} = \frac{n}{\sum_{i=1}^{n} x_i} = \frac{1}{\bar{X}}$$

```
################################################################################
# Situación 4: Estimación de λ en tiempos entre llamadas de seguros
################################################################################

# Parámetro del estimado del valor verdadero
lambda_real <- 2

# Tamaños de muestra que vamos a evaluar
n_values <- c(2, 5, 10, 30, 100)

# Número de repeticiones en la simulación
B <- 10000

# Función para poder generar datos y estimar λ con diferentes métodos
simular_estimadores <- function(n, B, lambda_real) {
  estimadores_lambda <- matrix(NA, nrow = B, ncol = 3)
  colnames(estimadores_lambda) <- c("MLE", "Momentos", "Sesgo_ECM")

  for (i in 1:B) {
    # Generamos una muestra de tamaño n con distribución exponencial
    X <- rexp(n, rate = lambda_real)
    S_n <- sum(X)  # Suma muestral

    # Estimadores:
    lambda_MLE <- n / S_n               # Máxima Verosimilitud (MLE)
    lambda_MOM <- 1 / mean(X)           # Momentos: Se usa E(X) = 1/λ → λ_MOM = 1/X¯
    lambda_sesgado <- (n - 1) / S_n     # Estimador sesgado del problema: (n-1)/S_n

    estimadores_lambda[i, ] <- c(lambda_MLE, lambda_MOM, lambda_sesgado)
  }

  # Calculamos el sesgo y ECM de cada uno de los estimadores
  sesgo_MLE <- mean(estimadores_lambda[, "MLE"]) - lambda_real
  sesgo_MOM <- mean(estimadores_lambda[, "Momentos"]) - lambda_real
  sesgo_Sesgado <- mean(estimadores_lambda[, "Sesgo_ECM"]) - lambda_real

  ECM_MLE <- mean((estimadores_lambda[, "MLE"] - lambda_real)^2)
  ECM_MOM <- mean((estimadores_lambda[, "Momentos"] - lambda_real)^2)
  ECM_Sesgado <- mean((estimadores_lambda[, "Sesgo_ECM"] - lambda_real)^2)

  return(c(Sesgo_MLE = sesgo_MLE, ECM_MLE = ECM_MLE,
           Sesgo_MOM = sesgo_MOM, ECM_MOM = ECM_MOM,
           Sesgo_Sesgado = sesgo_Sesgado, ECM_Sesgado = ECM_Sesgado))
}

# Simulamos diferentes tamaños de muestra
resultados_ECM <- lapply(n_values, function(n) simular_estimadores(n, B, lambda_real))
names(resultados_ECM) <- paste0("n=", n_values)

################################################################################
# Creación de la tabla ECM
################################################################################

ECM_tabla <- data.frame(
  n             = n_values,
  Sesgo_MLE     = sapply(resultados_ECM, function(x) x["Sesgo_MLE"]),
  ECM_MLE       = sapply(resultados_ECM, function(x) x["ECM_MLE"]),
  Sesgo_MOM     = sapply(resultados_ECM, function(x) x["Sesgo_MOM"]),
  ECM_MOM       = sapply(resultados_ECM, function(x) x["ECM_MOM"]),
  Sesgo_Sesgado = sapply(resultados_ECM, function(x) x["Sesgo_Sesgado"]),
  ECM_Sesgado   = sapply(resultados_ECM, function(x) x["ECM_Sesgado"])
)

# Imprimir tabla usando knitr si está disponible
if(requireNamespace("knitr", quietly = TRUE)){
  knitr::kable(ECM_tabla, caption = "Sesgo y ECM de los estimadores de λ para diferentes tamaños de muestra")
} else {
  print(ECM_tabla)
}
```

Sesgo y ECM de los estimadores de λ para diferentes tamaños de muestra

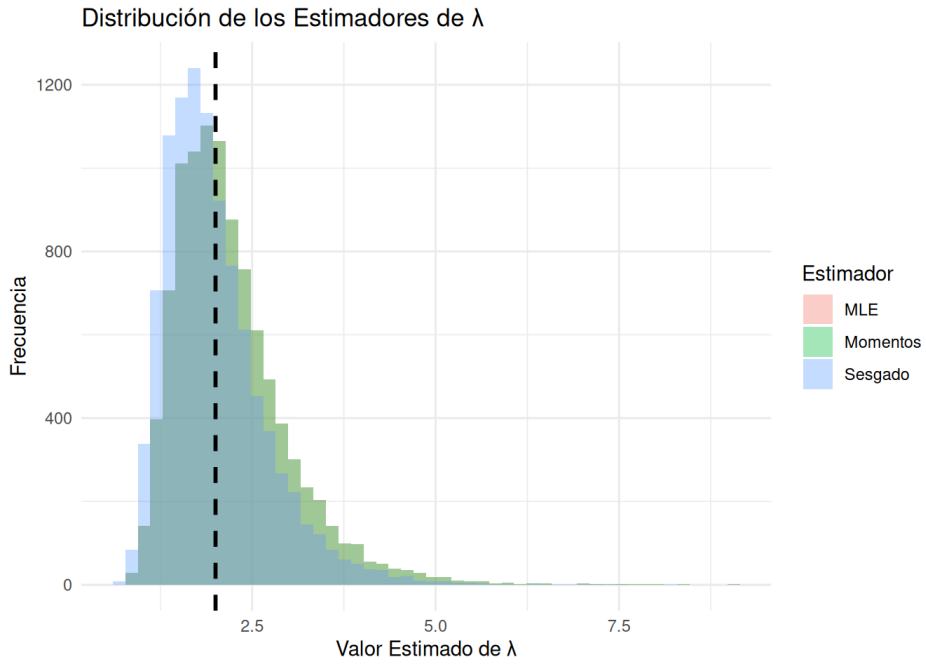
	n	Sesgo_MLE	ECM_MLE	Sesgo_MOM	ECM_MOM	Sesgo_Sesgado	ECM_Sesgado

	n	Sesgo_MLE	ECM_MLE	Sesgo_MOM	ECM_MOM	Sesgo_Sesgado	ECM_Sesgado
n=2.Sesgo_MLE	2	2.0191035	52.2745686	2.0191035	52.2745686	0.0095518	12.0495386
n=5.Sesgo_MLE	5	0.4760143	2.5577136	0.4760143	2.5577136	-0.0191885	1.4922876
n=10.Sesgo_MLE	10	0.2152827	0.6467647	0.2152827	0.6467647	-0.0062456	0.4863777
n=30.Sesgo_MLE	30	0.0696835	0.1587926	0.0696835	0.1587926	0.0006940	0.1438458
n=100.Sesgo_MLE	100	0.0193922	0.0417581	0.0193922	0.0417581	-0.0008018	0.0405592

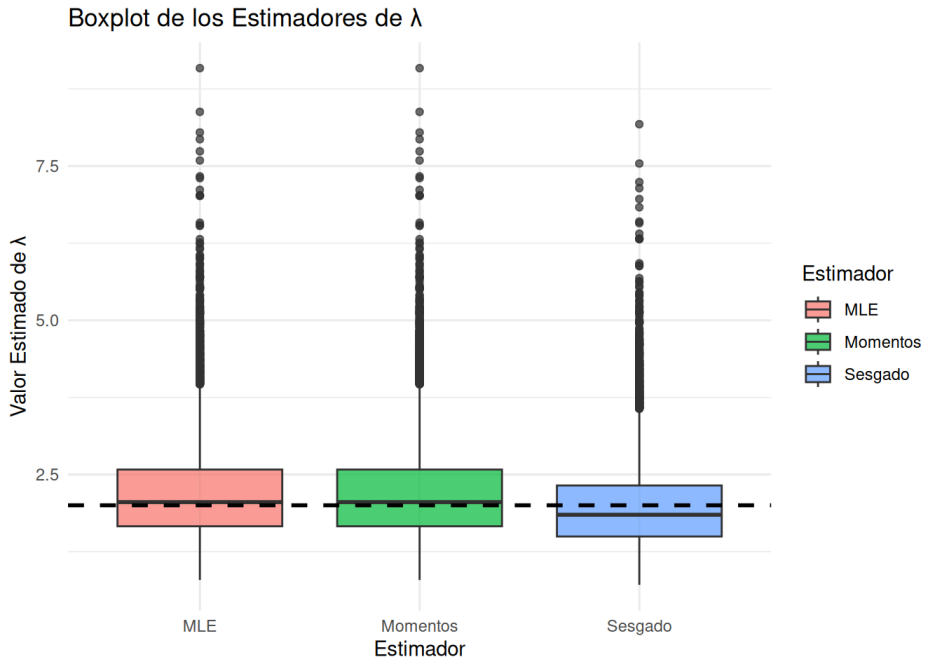
```
################################################################################
# Visualización: Comparación de los estimadores
################################################################################

par(mfrow=c(1,2))

# Parámetro real y tamaño de muestra para la simulación de gráficos
lambda_true <- 2     # Valor real de λ
n_ejemplo <- 10      # Tamaño de muestra (n)
B_plot <- 10000      # Número de simulaciones para la visualización

# Creamos una matriz para almacenar los estimadores en cada simulación
estimadores <- matrix(NA, nrow = B_plot, ncol = 3)
colnames(estimadores) <- c("MLE", "Momentos", "Sesgado")

# Simulamos B_plot repeticiones
for (i in 1:B_plot) {
  X <- rexp(n_ejemplo, rate = lambda_true)  # Muestra de tamaño n_ejemplo
  S_n <- sum(X)                             # Suma de la muestra

  # Estimadores
  estimadores[i, "MLE"] <- n_ejemplo / S_n                 # MLE: λˆ = n / S_n
  estimadores[i, "Momentos"] <- 1 / mean(X)                #  Momentos: λˆ = 1 / mean(X)
  estimadores[i, "Sesgado"] <- (n_ejemplo - 1) / S_n       # Estimador sesgado dado en el problema: λˆ = (n-1) /
S_n
}

# Usamos un data frame y organizamos los datos para pasarle a ggplot
df_estim <- as.data.frame(estimadores)
df_melt <- pivot_longer(df_estim, cols = c("MLE", "Momentos", "Sesgado"),
                        names_to = "Estimador", values_to = "Valor")

# Creamos un Histograma de los estimadores y los superponemos para observarlos
p1 <- ggplot(df_melt, aes(x = Valor, fill = Estimador)) +
  geom_histogram(alpha = 0.35, bins = 50, position = "identity") +
  geom_vline(xintercept = lambda_true, linetype = "dashed",
             color = "black", linewidth = 1) +
  labs(title = "Distribución de los Estimadores de λ",
       x = "Valor Estimado de λ", y = "Frecuencia") +
  theme_minimal()

# ahora creamos un Boxplot comparativo de los estimadores
p2 <- ggplot(df_melt, aes(x = Estimador, y = Valor, fill = Estimador)) +
  geom_boxplot(alpha = 0.7) +
  geom_hline(yintercept = lambda_true, linetype = "dashed",
             color = "black", size = 1) +
  labs(title = "Boxplot de los Estimadores de λ",
       x = "Estimador", y = "Valor Estimado de λ") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ¡ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# Mostramos los gráficos
print(p1)
```




```
print(p2)
```



Conclusiones * Tanto en el soporte matemático como el programa en R podemos observar que el estimador $\hat{\lambda} = \frac{n-1}{S_n}$ es **insesgado**.

- El ECM calculado es $ECM(\hat{\lambda}) = \frac{\lambda^2}{n-2}$.
- Los estimadores calculados de momentos y máxima verosimilitud coinciden ya que $\hat{\lambda} = \frac{1}{\bar{X}} = \frac{n}{S_n}$, con lo cual podemos definir que no importa cual usamos vamos a obtener la misma estimación, en términos generales podemos decir que aplicar el método de momentos es mas fácil de aplicar en situaciones que necesiten rapidez y facilidad, mientras, aplicar el método de máxima verosimilitud cuando se necesite un resultado mas robusto.

NOTAS

Reglas básicas de Logaritmos:

- Producto: $log(ab) = log(a) + log(b)$

- Cociente: $log(\frac{a}{b}) = log(a) - log(b)$
- Potencia: $log(a^r) = rlog(a)$
- Logaritmo de 1: $log(1) = 0$
- Cambio de base: $log_b(a) = \frac{log_c(a)}{log_c(b)}$

---

1. https://es.wikipedia.org/wiki/Ensayo_de_Bernoulli (https://es.wikipedia.org/wiki/Ensayo_de_Bernoulli)↩
2. https://www.eurekando.org/biografias/biografia-de-daniel-bernoulli-teoria-de-la-probabilidad/ (https://www.eurekando.org/biografias/biografia-de-daniel-bernoulli-teoria-de-la-probabilidad/)↩
3. https://es.statisticseasily.com/glosario/%C2%BFQu%C3%A9-es-la-distribuci%C3%B3n-binomial-negativa%3F/# (https://es.statisticseasily.com/glosario/%C2%BFQu%C3%A9-es-la-distribuci%C3%B3n-binomial-negativa%3F/#)↩
4. https://economipedia.com/definiciones/teorema-central-del-limite.html (https://economipedia.com/definiciones/teorema-central-del-limite.html)↩
5. https://es.wikipedia.org/wiki/Distribuci%C3%B3n_uniforme_discreta (https://es.wikipedia.org/wiki/Distribuci%C3%B3n_uniforme_discreta)↩
6. https://mundoteca.org/distribuciones-exponenciales-y-sus-caracteristicas/ (https://mundoteca.org/distribuciones-exponenciales-y-sus-caracteristicas/)↩
7. https://es.wikipedia.org/wiki/Distribuci%C3%B3n_gamma (https://es.wikipedia.org/wiki/Distribuci%C3%B3n_gamma)↩
8. https://halweb.uc3m.es/esp/Personal/personas/aarribas/esp/docs/estI_tema3.pdf (https://halweb.uc3m.es/esp/Personal/personas/aarribas/esp/docs/estI_tema3.pdf)↩