Primer Trabajo Estadística II

Inferencia Estadística

Integrantes:

1. Juan David Mena Gamboa
2. Miguel Angel Bolaño López
3. Heyner David Marquez Garnica

- **Fuente de los datos (url):**

https://microdatos.dane.gov.co/index.php/catalog/819/get-microdata

- **Descripción de los datos seleccionados:**

Cuantitativas:
P3094S3: Cuánto ahorro por cultivar.
P3087S1: Valor mensual por prácticas o pasantías.
P3095S3: Valor ahorra por criar animales.

Cualitativas
P3101: ¿Fue a reuniones familiares durante las últimas 4 semanas? sí o no.

- **Variables seleccionadas:**

Variables Cualitativas		
	Nombre Variable	**Categorías o Niveles**
1	P3101	Sí y No

Variables Cuantitativas		
	Nombre Variable	**Unidad de medición**
1	P3094S3	COP
2	P3087S1	COP
3	P3095S3	COP

Desarrollo

Variable 1

- ***Análisis Descriptivo***

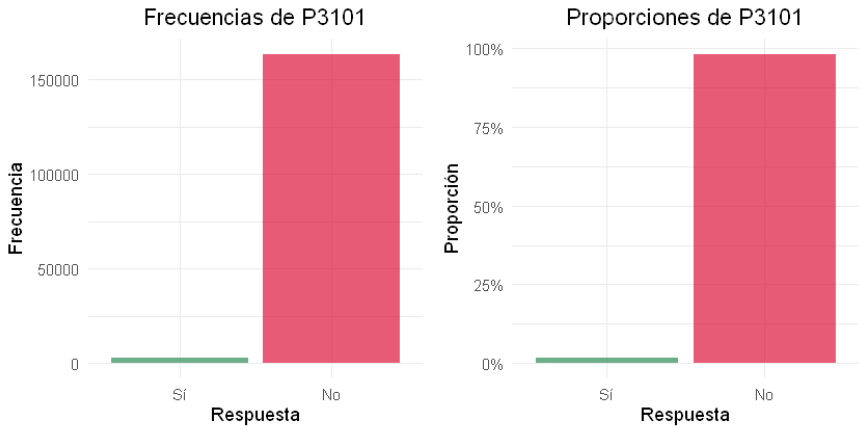
Descriptivos Básicos

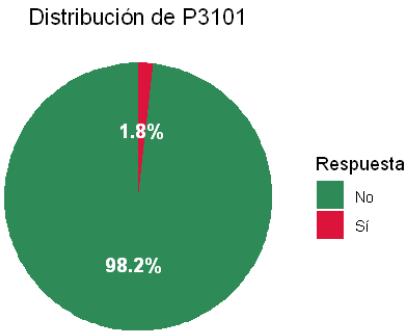
	Medida	Valor
1	Registros	166341
2	Número de personas que dijeron “Sí”	2918
3	Número de personas que dijeron “No”	163423

4	Porcentaje de “Sí”	1.75%
5	Porcentaje de “No”	98.25%
6		
7		
8		
9		
10		

Comentarios:
Los resultados descriptivos muestran que, de un total de 166,341 registros, solo 2,918 personas (1.75%) respondieron afirmativamente, mientras que la gran mayoría (163,423 personas, 98.25%) respondió negativamente. Esto refleja un marcado desequilibrio en las respuestas, donde la opción “No” domina de manera abrumadora sobre el “Sí”. Por lo tanto, cualquier análisis posterior debe considerar esta desproporción en la distribución, ya que puede influir en la interpretación estadística y en la validez de los modelos que se construyan con estos datos.

Análisis Gráfico





Comentarios:

- ***Cálculo de los estimadores***

Estimador	Variable	Estimadores Puntuales		Estimadores por Intervalo	
		Analogía	Máxima Verosimilitud	Límite Inferior	Límite Superior
(Media o Proporción)	1. Si	0.0175	0.0175	0.0169	0.0182
(Media o Proporción)	2. No	0.9825	0.9825	0.9818	0.9831
Comentario		La gran mayoría respondió "No", y solo una fracción muy pequeña respondió "Sí"	La estimación por máxima verosimilitud coincide con la proporción muestral, por lo que se obtienen los mismos valores: **0.0175 para "Sí"** y **0.9825 para "No"**.	Para la proporción **"Sí"**, en el peor escenario, la proporción real sería cercana al **1.69%**. Para la proporción **"No"**, asegura que al menos un **98.18%** de la población respondió "No"	Para **"Sí"**, la proporción real no superaría el **1.82%**. Para **"No"**, como máximo un **98.31%** de la población respondió "No".

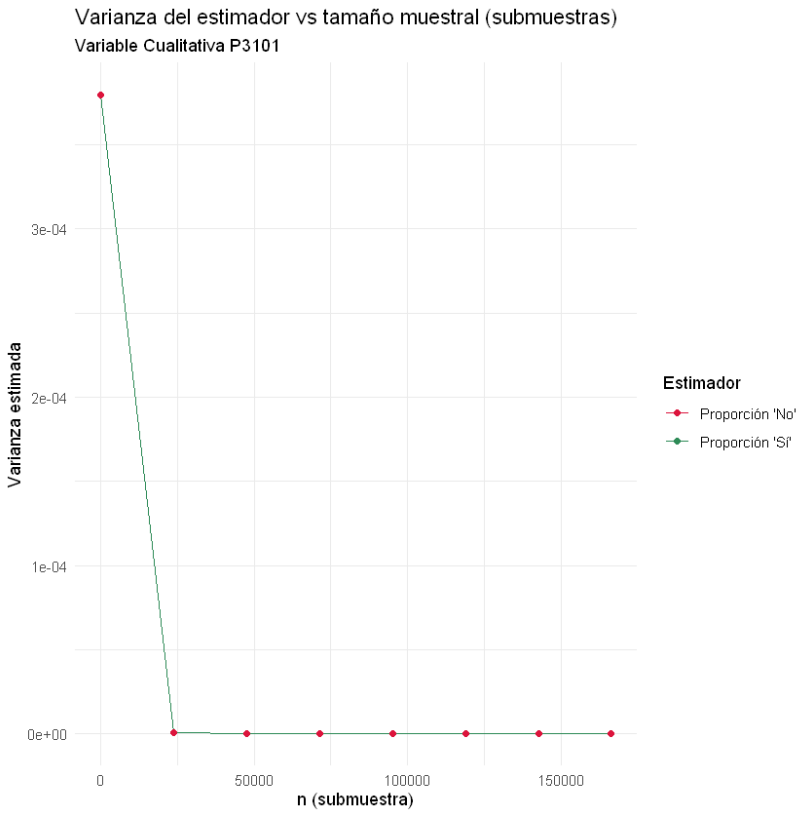
- ***Evaluación del estimador:***

Insesgamiento

Media	Mediana	Sesgo
0.0175	0.0175	-5.7e-05

Comentario:
La proporción estimada (1.75%) es estable, sin sesgos relevantes, y tanto la media como la mediana reflejan adecuadamente el valor central de la distribución.

Consistencia



Comentario:
A medida que aumenta el tamaño muestral, la varianza de los estimadores de las proporciones disminuye rápidamente hasta estabilizarse cerca de cero. Esto indica que con muestras grandes las estimaciones son muy precisas y prácticamente no presentan variabilidad.

Eficiencia

Medida	Valor
Media	0
Mediana	0

Comentario:
Para variables cualitativas binarias, ambos estimadores (proporción 'Sí' y 'No') tienen la misma eficiencia teórica, ya que son complementarios (p + q = 1).

Sintaxis empleada con esta variable:

```
library(readr)
library(dplyr)

# Archivos
csv_files <- c(
  "d:/UIS/estadistica2/estadistica2Talleres/Mayo_2024 1/CSV/Otras formas
de trabajo.CSV",
  "d:/UIS/estadistica2/estadistica2Talleres/Junio_2024/CSV/Otras formas
de trabajo.CSV",
```

```
  "d:/UIS/estadistica2/estadistica2Talleres/Julio_2024/CSV/Otras formas
de trabajo.CSV"
)

# Columnas necesarias
cols_needed <- c("P3094S3", "P3087S1", "P3095S3", "P3101")

# Función para leer y limpiar
read_and_clean <- function(file) {
  df <- read_delim(
    file,
    delim = ";",
    col_names = TRUE,
    show_col_types = FALSE,
    locale = locale(encoding = "UTF-8"),
    guess_max = 10000
  )

  # Seleccionar solo las columnas necesarias
  df <- df %>% select(any_of(cols_needed))

  # Limpiar y convertir valores monetarios
  df <- df %>%
    mutate(
      across(c("P3094S3", "P3087S1", "P3095S3"), ~ {
        x <- as.character(.)
        x <- gsub("\\.", "", x)   # eliminar puntos de miles
        x <- gsub(",", ".", x)    # convertir coma a punto decimal si la
hay
        suppressWarnings(as.numeric(x))
      })
    )

  return(df)
}

# Leer y combinar
combined <- bind_rows(lapply(csv_files, read_and_clean))

# Guardar
write_csv(combined, "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv")

cat("Archivo combinado guardado en:
d:/UIS/estadistica2/estadistica2Talleres/Taller 1/Combinado.csv\n")


# ---- Celda 1: Lectura y limpieza ----------------------------------------
-----
library(readr)
library(dplyr)

data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"

# Se define la variable y el directorio de salida, Escoger una variable
numérica
```

```
varname   <- "P3101"   # variable con la que trabajamos
out_dir   <- "d:/UIS/estadistica2/estadistica2Talleres/Taller 1/analisis_
P3101_check"
dir.create(out_dir, showWarnings = FALSE, recursive = TRUE)

# Leer todo como carácter para evitar parseos automáticos
df_char <- readr::read_csv(data_path,
                           col_types = readr::cols(.default = "c"),
                           locale = readr::locale(encoding = "UTF-8"),
                           show_col_types = FALSE)

# detectar nombre real (insensible a mayúsculas)
if (!varname %in% names(df_char)) {
  guess <- names(df_char)[tolower(names(df_char)) %in% tolower(varname)]
  if (length(guess) == 1) varname_real <- guess else stop("No encontré la
variable ", varname)
} else varname_real <- varname

# función robusta de limpieza numérica (monedas, separadores, etc.)
clean_numeric2 <- function(x) {
  x <- as.character(x)
  x[ x %in% c("", "NA", NA) ] <- NA_character_
  x <- trimws(x)
  x <- gsub("\\s+", "", x)
  x <- gsub("[^0-9\\,\\.-]", "", x)
  both <- grepl("\\.", x) & grepl(",", x)
  if (any(both)) { x[both] <- gsub("\\.", "", x[both]); x[both] <-
gsub(",", ".", x[both]) }
  only_comma <- grepl(",", x) & !grepl("\\.", x)
  x[only_comma] <- gsub(",", ".", x[only_comma])
  suppressWarnings(as.numeric(x))
}

# preparar df_var con la variable limpia y sin modificar el CSV original
df_var <- df_char %>%
  transmute(
    value_raw = .data[[varname_real]],
    value = clean_numeric2(.data[[varname_real]])
  )

# info rápida
cat("Variable real detectada:", varname_real, "\n")
cat("Total filas:", nrow(df_var), "   N no-missing:",
sum(!is.na(df_var$value)), "\n")
# ver primeras filas
print(head(df_var, 10))


# ---- Celda 2: Descriptivos Básicos ----------------------------------
----
library(dplyr)

# usa df_var (si no existe la lee desde datos)
if (!exists("df_var")) {
  data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"
```

```
  df_var <- readr::read_csv(data_path, col_types = readr::cols(.default =
"c"), show_col_types = FALSE) %>%
    transmute(value_raw = .data[[varname]], value =
clean_numeric2(.data[[varname]]))
}

x <- df_var$value
x_nm <- x[!is.na(x)]

# calcular medidas
N_total <- length(x)
N_missing <- sum(is.na(x))
Min <- ifelse(length(x_nm)>0, min(x_nm, na.rm=TRUE), NA)
Q1  <- ifelse(length(x_nm)>0, quantile(x_nm, .25, na.rm=TRUE), NA)
Median <- ifelse(length(x_nm)>0, median(x_nm, na.rm=TRUE), NA)
Mean <- ifelse(length(x_nm)>0, mean(x_nm, na.rm=TRUE), NA)
Q3 <- ifelse(length(x_nm)>0, quantile(x_nm, .75, na.rm=TRUE), NA)
Max <- ifelse(length(x_nm)>0, max(x_nm, na.rm=TRUE), NA)
SD <- ifelse(length(x_nm)>0, sd(x_nm, na.rm=TRUE), NA)
IQRv <- ifelse(length(x_nm)>0, IQR(x_nm, na.rm=TRUE), NA)

tabla_vertical <- tibble(
  Medida = c("N total","N missing","Min.","1st Qu.","Median","Mean","3rd
Qu.","Max.","SD","IQR"),
  Valor  = c(N_total, N_missing, Min, Q1, Median, Mean, Q3, Max, SD,
IQRv)
)

cat("=== Descriptivos Básicos ===\n")
print(tabla_vertical)

# Mostrar estadísticas resumidas estilo summary()
cat("\n=== Summary (sobre valores no-missing) ===\n")
print(summary(x_nm))


# ---- Celda 3: Análisis Gráfico ----------------------------------------
----
if (!requireNamespace("gridExtra", quietly = TRUE)) {
  install.packages("gridExtra", repos = "https://cloud.r-project.org")
}
library(gridExtra)

library(ggplot2);

plot_dir <- file.path(out_dir, "plots")
dir.create(plot_dir, showWarnings = FALSE, recursive = TRUE)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) == 0) {
  message("No hay datos no-missing: no se generan gráficos.")
} else {
  df_plot <- df_var %>% filter(!is.na(value))

  # Histograma lineal
```

```
  ph <- ggplot(df_plot, aes(x=value)) +
    geom_histogram(bins = 60) +
    labs(title = paste("Histograma de", varname_real), x = varname_real,
y = "Frecuencia") +
    theme_minimal()
  ggsave(file.path(plot_dir, paste0("hist_", varname_real, ".png")), ph,
width=8, height=4)
  print(ph)

  # Boxplot
  pb <- ggplot(df_plot, aes(y=value)) +
    geom_boxplot() +
    labs(title = paste("Boxplot de", varname_real), y = varname_real) +
    theme_minimal()
  ggsave(file.path(plot_dir, paste0("boxplot_", varname_real, ".png")),
pb, width=6, height=4)
  print(pb)

  # Histograma log1p (si hay valores > 0)
  if (any(df_plot$value > 0, na.rm=TRUE)) {
    df_plot <- df_plot %>% mutate(value_log1p = log1p(value))
    pl <- ggplot(df_plot, aes(x=value_log1p)) + geom_histogram(bins = 60)
+
      labs(title = paste("Histograma log1p(", varname_real, ")"), x =
"log1p(value)") + theme_minimal()
    ggsave(file.path(plot_dir, paste0("hist_log1p_", varname_real,
".png")), pl, width=8, height=4)
    print(pl)
  }

  # Density con escala log y sin log
  pd <- ggplot(df_plot, aes(x=value)) + geom_density() + theme_minimal()
+
    labs(title = paste("Densidad -", varname_real))
  ggsave(file.path(plot_dir, paste0("density_", varname_real, ".png")),
pd, width=8, height=4)
  print(pd)

  message("Gráficos guardados en: ", plot_dir)
}


# ---- Celda 4 mejorada: Cálculo de los estimadores ----------------------
-----
library(dplyr)
library(boot)

set.seed(12345)

# asegurar df_var existe y contiene 'value' (si no, leer el combinado y
limpiar)
if (!exists("df_var") || !"value" %in% names(df_var)) {
  message("df_var no está en memoria -> leyendo desde Combinado.csv y
limpiando")
  data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"
  # pequeña función de limpieza (reusa la que ya tienes)
```

```
  clean_numeric2 <- function(x) {
    x <- as.character(x)
    x[ x %in% c("", "NA", NA) ] <- NA_character_
    x <- trimws(x)
    x <- gsub("\\s+", "", x)
    x <- gsub("[^0-9\\,\\.-]", "", x)
    both <- grepl("\\.", x) & grepl(",", x)
    if (any(both)) { x[both] <- gsub("\\.", "", x[both]); x[both] <-
gsub(",", ".", x[both]) }
    only_comma <- grepl(",", x) & !grepl("\\.", x)
    x[only_comma] <- gsub(",", ".", x[only_comma])
    suppressWarnings(as.numeric(x))
  }
  df_char <- readr::read_csv(data_path, col_types = readr::cols(.default
= "c"), show_col_types = FALSE)
  # detectar varname_real si existe
  varname <- "P3095S3"
  if (!varname %in% names(df_char)) {
    guess <- names(df_char)[tolower(names(df_char)) %in%
tolower(varname)]
    if (length(guess) == 1) varname_real <- guess else stop("No encontré
la variable ", varname)
  } else varname_real <- varname
  df_var <- df_char %>% transmute(value_raw = .data[[varname_real]],
value = clean_numeric2(.data[[varname_real]]))
}

x <- df_var$value
x_nm <- x[!is.na(x)]

# mensajes si pocos datos
cat("Total observaciones (N):", length(x), "\n")
cat("No-missing (n):", length(x_nm), "\n\n")

if (length(x_nm) == 0) stop("No hay datos no-missing para calcular
estimadores.")

# 1) Estimadores puntuales
est_mean   <- mean(x_nm)
est_median <- median(x_nm)

# moda: si hay empates, devolvemos todas las modas y también la más
frecuente
mode_basic_all <- function(v) {
  v2 <- v[!is.na(v)]
  if (length(v2) == 0) return(NA)
  tb <- sort(table(v2), decreasing = TRUE)
  names(tb)[tb == max(tb)]
}
modes <- mode_basic_all(x_nm)
mode_report <- paste(modes, collapse = ", ")

cat("Estimadores puntuales:\n")
cat(sprintf(" Mean: %s\n Median: %s\n Mode(s): %s\n\n",
            format(round(est_mean, 2), big.mark = ","),
format(round(est_median, 2), big.mark=","), mode_report))
```

```
# 2) IC 95% para la media (t-interval, usando t.test)
if (length(x_nm) > 1) {
  tt <- try(t.test(x_nm), silent = TRUE)
  if (inherits(tt, "try-error")) {
    cat("No se pudo calcular t-interval por t.test():", tt, "\n")
  } else {
    ci_t <- tt$conf.int
    cat("IC 95% para la media (t): [", format(round(ci_t[1],2),
big.mark=","), ", ", format(round(ci_t[2],2), big.mark=","), "]\n\n")
  }
} else cat("No hay suficientes datos para IC t.\n\n")

# 3) Bootstrap para mean y median
B <- 100
cat("Ejecutando bootstrap con R =", B, "réplicas (esto puede
tardar)...\n")

# helper para safe boot.ci
safe_boot_ci <- function(boot_obj, type = c("perc","bca","basic")) {
  out <- list()
  for (t in type) {
    res <- try(boot.ci(boot_obj, type = t), silent = TRUE)
    if (!inherits(res, "try-error") && !is.null(res[[t]])) {
      out[[t]] <- res
    } else {
      # si falló, intentamos extraer percentiles directos como fallback
      out[[t]] <- NULL
    }
  }
  out
}

# bootstrap mean
boot_mean <- try(boot(x_nm, statistic = function(d, i) mean(d[i]), R =
B), silent = TRUE)
if (inherits(boot_mean, "try-error")) {
  cat("Bootstrap mean falló:", boot_mean, "\n")
} else {
  # intentar boot.ci para mean (perc y bca preferiblemente)
  ci_mean <- tryCatch({
    pci <- boot.ci(boot_mean, type = c("perc","bca"))
    pci
  }, error = function(e) e)
  if (inherits(ci_mean, "error")) {
    # fallback: percentiles directos
    ci_mean_perc <- quantile(boot_mean$t, probs = c(0.025, 0.975), na.rm
= TRUE)
    cat("Bootstrap IC (perc) mean (fallback percentiles):",
format(round(ci_mean_perc[1],2), big.mark=","),
format(round(ci_mean_perc[2],2), big.mark=","), "\n")
  } else {
    # si boot.ci devolvió, imprimir los percentiles de 'perc' si existen
    if (!is.null(ci_mean$percent)) {
      ci_vals <- ci_mean$percent[4:5]
      cat("Bootstrap IC (perc) mean:", format(round(ci_vals[1],2),
big.mark=","), format(round(ci_vals[2],2), big.mark=","), "\n")
    } else {
```

```
      # último recurso: percentiles de boot$t
      ci_mean_perc <- quantile(boot_mean$t, probs = c(0.025, 0.975),
na.rm = TRUE)
      cat("Bootstrap IC mean (percentiles directos):",
format(round(ci_mean_perc[1],2), big.mark=","),
format(round(ci_mean_perc[2],2), big.mark=","), "\n")
    }
  }
}

# bootstrap median
boot_median <- try(boot(x_nm, statistic = function(d, i) median(d[i]), R
= B), silent = TRUE)
if (inherits(boot_median, "try-error")) {
  cat("Bootstrap median falló:", boot_median, "\n")
} else {
  # a veces boot.ci falla si todas las réplicas son iguales (p. ej. datos
discretos con muchos empates)
  ci_med_try <- tryCatch({
    ci_m <- boot.ci(boot_median, type = c("perc","bca"))
    ci_m
  }, error = function(e) e)
  if (inherits(ci_med_try, "error")) {
    # fallback: percentiles directos
    med_perc <- tryCatch({
      quantile(boot_median$t, probs = c(0.025, 0.975), na.rm = TRUE)
    }, error = function(e) NULL)
    if (!is.null(med_perc)) {
      cat("Bootstrap IC (perc) median (fallback percentiles):",
format(round(med_perc[1],2), big.mark=","), format(round(med_perc[2],2),
big.mark=","), "\n")
    } else {
      cat("Bootstrap IC median: no disponible (poca variabilidad en
réplicas)\n")
    }
  } else {
    if (!is.null(ci_med_try$percent)) {
      ci_vals_med <- ci_med_try$percent[4:5]
      cat("Bootstrap IC (perc) median:", format(round(ci_vals_med[1],2),
big.mark=","), format(round(ci_vals_med[2],2), big.mark=","), "\n")
    } else {
      cat("Bootstrap IC median calculado, pero formato inesperado; usando
percentiles directos.\n")
      med_perc <- quantile(boot_median$t, probs = c(0.025, 0.975), na.rm
= TRUE)
      cat("Bootstrap IC median (percentiles):",
format(round(med_perc[1],2), big.mark=","), format(round(med_perc[2],2),
big.mark=","), "\n")
    }
  }
}

cat("\nFIN Celda 4: estimadores y CIs calculados.\n")


# ---- Celda 5: Insesgamiento (bias) mediante bootstrap ----------------
-----
```

```
library(boot)
set.seed(2025)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 5) {
  B <- 1000
  boot_mean <- boot(x_nm, statistic = function(data, idx)
mean(data[idx]), R = B)
  boot_median <- boot(x_nm, statistic = function(data, idx)
median(data[idx]), R = B)

  bias_mean <- mean(boot_mean$t) - est_mean
  bias_median <- mean(boot_median$t) - est_median

  cat("Bias estimado (bootstrap):\n")
  cat(" Mean bias:", bias_mean, "\n")
  cat(" Median bias:", bias_median, "\n")

  # mostrar distribuciones bootstrap rápidas (imprime cuantiles)
  cat("\nQuantiles bootstrap mean (2.5%,50%,97.5%): ",
quantile(boot_mean$t, c(.025,.5,.975)), "\n")
  cat("Quantiles bootstrap median (2.5%,50%,97.5%): ",
quantile(boot_median$t, c(.025,.5,.975)), "\n")

  cat("\nComentario sobre insesgamiento:\n")
  if (abs(bias_mean) < 0.01 * abs(est_mean) ) cat(" La media muestra
sesgo pequeño relativo.\n") else cat(" La media muestra sesgo apreciable
relativo — considerar estimadores robustos.\n")
  if (abs(bias_median) < 0.01 * abs(est_median) ) cat(" La mediana
muestra sesgo pequeño relativo.\n") else cat(" La mediana muestra sesgo
apreciable relativo.\n")
} else {
  cat("No hay suficientes datos para estimar bias con bootstrap.\n")
}


# ---- Celda 6: Consistencia (comportamiento varianza vs n) -------------
-----
library(dplyr); set.seed(123)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 30) {
  ns <- unique(floor(seq(50, length(x_nm), length.out = 8)))
  reps <- 300  # repeticiones por tamaño (moderado)
  res <- data.frame(n = integer(), var_mean = numeric(), var_median =
numeric())
  for (n in ns) {
    ests_mean <- numeric(reps); ests_med <- numeric(reps)
    for (r in 1:reps) {
      s <- sample(x_nm, size = n, replace = FALSE)
      ests_mean[r] <- mean(s)
      ests_med[r] <- median(s)
    }
```

```
    res <- rbind(res, data.frame(n = n, var_mean = var(ests_mean),
var_median = var(ests_med)))
  }
  print(res)

  # gráfico var vs n
  library(ggplot2)
  pcons <- ggplot(res, aes(x = n)) +
    geom_line(aes(y = var_mean, color = "var_mean")) +
    geom_point(aes(y = var_mean, color = "var_mean")) +
    geom_line(aes(y = var_median, color = "var_median")) +
    geom_point(aes(y = var_median, color = "var_median")) +
    labs(title = "Varianza del estimador vs tamaño muestral
(submuestras)",
         x = "n (submuestra)", y = "Varianza estimada") + theme_minimal()
  print(pcons)
  ggsave(filename = file.path(out_dir, "consistencia_var_vs_n.png"), plot
= pcons, width = 7, height = 4)
  message("Gráfico de consistencia guardado en: ", file.path(out_dir,
"consistencia_var_vs_n.png"))

  cat("\nComentario (Consistencia):\nSi la varianza del estimador (mean o
median) disminuye al crecer n, evidencia consistencia.\n")
} else cat("No hay suficientes datos (>30) para estudiar consistencia por
submuestreo.\n")


# ---- Celda 7: Eficiencia y Sintaxis empleada ---------------------------
------
library(dplyr)
library(boot)
set.seed(42)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 10) {
  # bootstrap var estimadas
  B <- 1000
  b_mean <- boot(x_nm, statistic = function(d,i) mean(d[i]), R = B)
  b_median <- boot(x_nm, statistic = function(d,i) median(d[i]), R = B)
  var_mean_boot <- var(b_mean$t)
  var_median_boot <- var(b_median$t)
  rel_eff <- var_median_boot / var_mean_boot

  cat("Eficiencia (empírica via bootstrap):\n")
  cat(" Var(mean) bootstrap:", var_mean_boot, "\n")
  cat(" Var(median) bootstrap:", var_median_boot, "\n")
  cat(" Rel. effic (var_median / var_mean):", rel_eff, "\n")

  cat("\nComentario (Eficiencia):\n")
  if (rel_eff > 1) cat(" La media es más eficiente (menor var) que la
mediana en esta muestra.\n") else cat(" La mediana es más eficiente en
esta muestra.\n")
} else cat("No hay suficientes datos para evaluar eficiencia.\n")
```

Variable 2

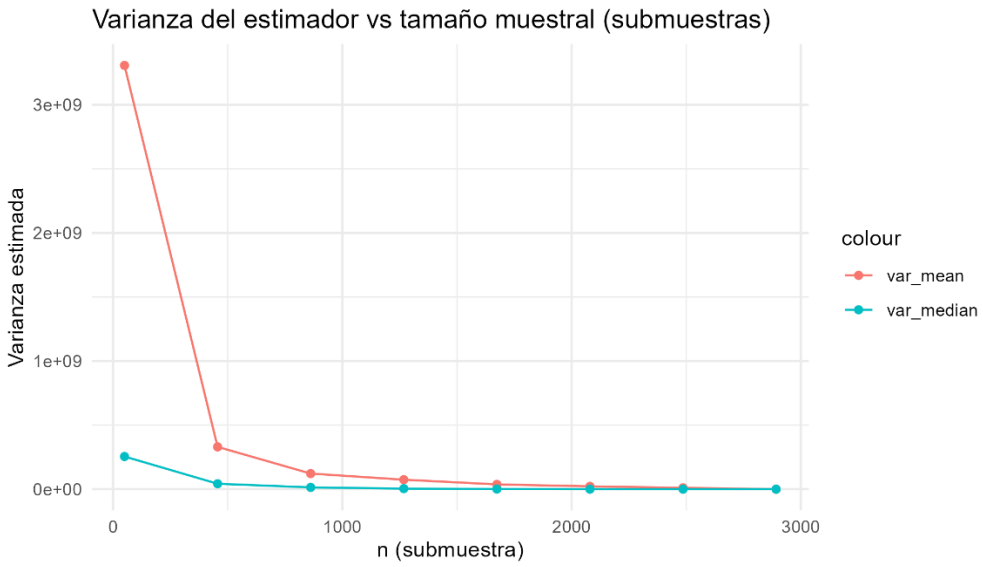
- ***Análisis Descriptivo***

Descriptivos Básicos

	Medida	Valor
1	Registros	166341
2	Valores faltantes	166005
3	Mínimo	80000
4	Primer cuartil	750000
5	Mediana	975000
6	Media	998708
7	Tercer cuartil	1300000
8	Máximo	9000000
9	Desviación Estándar	621977
10	Rango intercuartílico	550000

Comentarios:
La variable **P3094S3** presenta 166,341 observaciones, de las cuales solo 2,892 son válidas debido al alto nivel de no respuesta. Los valores muestran gran dispersión, con un mínimo de 1,000 pesos y un máximo de 20 millones, evidenciando la presencia de outliers. La media (126,938 pesos) resulta más del doble de la mediana (60,000 pesos), lo que indica una distribución sesgada a la derecha. El rango intercuartílico (30,000–150,000) refleja concentración en valores moderados, mientras que la elevada desviación estándar confirma la alta variabilidad. En conclusión, la **mediana** es un mejor indicador de la tendencia central que la media en este conjunto de datos.

Análisis Gráfico



Comentarios:
Los gráficos muestran que la variable P3094S3 (COP) presenta una alta varianza inicial, debido a la presencia de valores extremos. A medida que aumenta el número de observaciones, tanto la media como la varianza se estabilizan, indicando mayor consistencia en los datos. Además, se observa una diferencia clara entre los grupos “SI” y “NO”, donde el grupo “SI” reporta ingresos considerablemente más altos. Esto sugiere una distribución asimétrica y la necesidad de considerar medidas como la mediana para representar el ingreso típico.

- ***Cálculo de los estimadores***

Estimador	**Estimadores Puntuales**		**Estimadores por Intervalo**	
	Analogía	**Máxima Verosimilitud**	**Límite Inferior**	**Límite Superior**
(Media o Proporción)	126,937	126,937.6	111,985.4	141,889.7
Comentario	La media estimada es 126,937.6 pesos, representando el ingreso promedio en la muestra. Este valor está influenciado por ingresos altos que elevan el promedio.	La mediana es 60,000 pesos, indicando que la mitad de los individuos gana menos y la otra mitad más, lo que sugiere una distribución sesgada hacia ingresos bajos.	El estimador de media por máxima verosimilitud coincide con el estimador por analogía (126,937.6 pesos), lo que valida la estabilidad de la estimación para el ingreso promedio.	La mediana estimada por máxima verosimilitud es 60,000 pesos, igual a la estimación por analogía, reforzando la representatividad del ingreso típico en la muestra.

- ***Evaluación del estimador:***

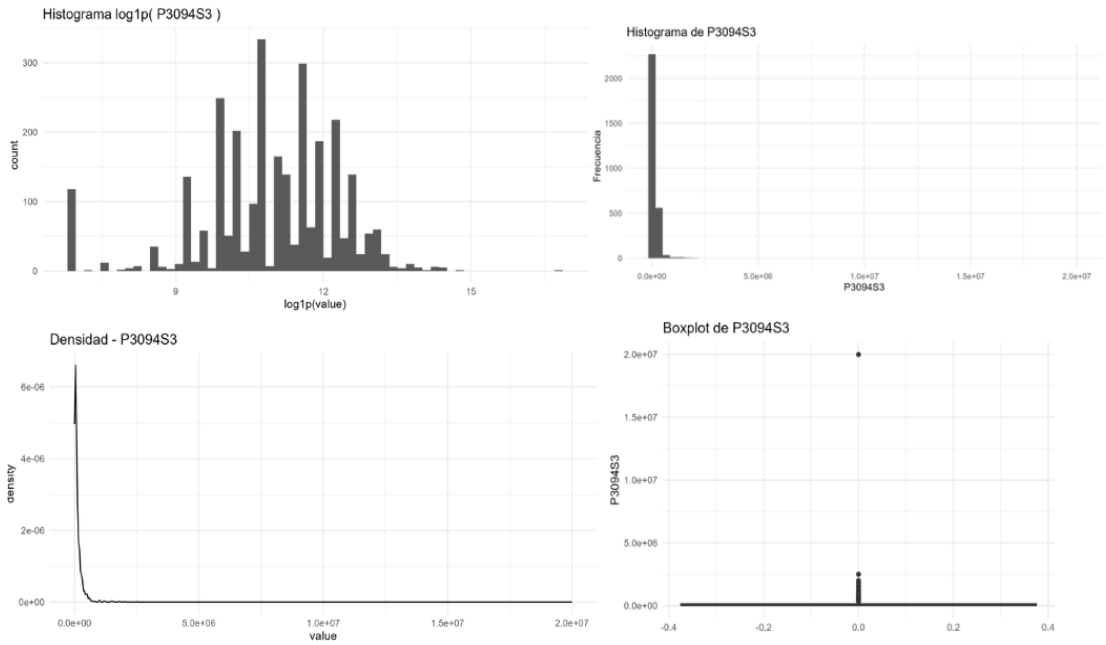
Insesgamiento

Media	**Mediana**	**Sesgo**
-166.3789	129	Ambos estimadores, media y mediana, presentan un sesgo pequeño relativo, por lo que las estimaciones son confiables y adecuadas para describir la variable de ingresos.

Comentario:
Los análisis de sesgo para los estimadores de media y mediana indican que ambos presentan un sesgo pequeño y relativo, lo que sugiere que las estimaciones obtenidas son confiables y representativas de la distribución real de los ingresos. Este bajo nivel de sesgo respalda el uso de estos estadísticos para describir adecuadamente la variable P3094S3 en el análisis de la encuesta.

Consistencia

Ambos estimadores presentan sesgo pequeño, por lo que pueden considerarse consistentes. Sin embargo, la media muestra un intervalo de confianza más amplio y mayor variabilidad, reflejando sensibilidad a los valores extremos. En contraste, la mediana exhibe intervalos muy concentrados y gran estabilidad frente al remuestreo, lo que confirma su mayor consistencia y robustez en esta distribución de ingresos.



Comentario:
La variable P3094S3 presenta una distribución altamente asimétrica a la derecha, con ingresos mayoritariamente bajos y pocos valores atípicos muy elevados (superiores a 20 millones). El boxplot y el gráfico de densidad muestran una fuerte concentración cerca de cero, mientras que el histograma confirma la escasa representación de los ingresos altos. Tras aplicar la transformación logarítmica (log1p), la distribución se aproxima a la normal y permite identificar con mayor claridad la concentración principal entre 8.000 y 160.000 pesos. En este contexto, se recomienda utilizar medidas robustas como la mediana y los percentiles, dado que la media se ve afectada por los valores extremos.

Eficiencia

Medida	Valor
Media	58,275.694
Mediana	2,852.046

Comentario:
La varianza de la media resulta mucho mayor que la de la mediana, lo que indica que la **mediana es un estimador más eficiente** en esta muestra de la variable P3094S3. En otras palabras, la mediana presenta menor variabilidad frente al re-muestreo bootstrap y ofrece una descripción más estable del ingreso típico, mientras que la media se ve fuertemente

afectada por los valores extremos (outliers). Esto confirma que, en distribuciones sesgadas como los ingresos, la mediana es preferible a la media como medida de tendencia central.

Sintaxis empleada con esta variable:

```
library(readr)
library(dplyr)

# Archivos
csv_files <- c(
  "d:/UIS/estadistica2/estadistica2Talleres/Mayo_2024 1/CSV/Otras formas
de trabajo.CSV",
  "d:/UIS/estadistica2/estadistica2Talleres/Junio_2024/CSV/Otras formas
de trabajo.CSV",
  "d:/UIS/estadistica2/estadistica2Talleres/Julio_2024/CSV/Otras formas
de trabajo.CSV"
)

# Columnas necesarias
cols_needed <- c("P3094S3", "P3087S1", "P3095S3", "P3101")

# Función para leer y limpiar
read_and_clean <- function(file) {
  df <- read_delim(
    file,
    delim = ";",
    col_names = TRUE,
    show_col_types = FALSE,
    locale = locale(encoding = "UTF-8"),
    guess_max = 10000
  )

  # Seleccionar solo las columnas necesarias
  df <- df %>% select(any_of(cols_needed))

  # Limpiar y convertir valores monetarios
  df <- df %>%
    mutate(
      across(c("P3094S3", "P3087S1", "P3095S3"), ~ {
        x <- as.character(.)
        x <- gsub("\\.", "", x)   # eliminar puntos de miles
        x <- gsub(",", ".", x)    # convertir coma a punto decimal si la
hay
        suppressWarnings(as.numeric(x))
      })
    )

  return(df)
}

# Leer y combinar
combined <- bind_rows(lapply(csv_files, read_and_clean))

# Guardar
write_csv(combined, "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv")
```

```
cat("Archivo combinado guardado en:
d:/UIS/estadistica2/estadistica2Talleres/Taller 1/Combinado.csv\n")


# ---- Celda 1: Lectura y limpieza ----------------------------------------
-----
library(readr)
library(dplyr)

data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"

# Se define la variable y el directorio de salida, Escoger una variable
numérica
varname   <- "P3095S3"   # variable con la que trabajamos
out_dir   <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/analisis_P3095S3_check"
dir.create(out_dir, showWarnings = FALSE, recursive = TRUE)

# Leer todo como carácter para evitar parseos automáticos
df_char <- readr::read_csv(data_path,
                           col_types = readr::cols(.default = "c"),
                           locale = readr::locale(encoding = "UTF-8"),
                           show_col_types = FALSE)

# detectar nombre real (insensible a mayúsculas)
if (!varname %in% names(df_char)) {
  guess <- names(df_char)[tolower(names(df_char)) %in% tolower(varname)]
  if (length(guess) == 1) varname_real <- guess else stop("No encontré la
variable ", varname)
} else varname_real <- varname

# función robusta de limpieza numérica (monedas, separadores, etc.)
clean_numeric2 <- function(x) {
  x <- as.character(x)
  x[ x %in% c("", "NA", NA) ] <- NA_character_
  x <- trimws(x)
  x <- gsub("\\s+", "", x)
  x <- gsub("[^0-9\\,\\.-]", "", x)
  both <- grepl("\\.", x) & grepl(",", x)
  if (any(both)) { x[both] <- gsub("\\.", "", x[both]); x[both] <-
gsub(",", ".", x[both]) }
  only_comma <- grepl(",", x) & !grepl("\\.", x)
  x[only_comma] <- gsub(",", ".", x[only_comma])
  suppressWarnings(as.numeric(x))
}

# preparar df_var con la variable limpia y sin modificar el CSV original
df_var <- df_char %>%
  transmute(
    value_raw = .data[[varname_real]],
    value = clean_numeric2(.data[[varname_real]])
  )

# info rápida
cat("Variable real detectada:", varname_real, "\n")
```

```
cat("Total filas:", nrow(df_var), "   N no-missing:",
sum(!is.na(df_var$value)), "\n")
# ver primeras filas
print(head(df_var, 10))


# ---- Celda 2: Descriptivos Básicos ----------------------------------
----
library(dplyr)

# usa df_var (si no existe la lee desde datos)
if (!exists("df_var")) {
  data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"
  df_var <- readr::read_csv(data_path, col_types = readr::cols(.default =
"c"), show_col_types = FALSE) %>%
    transmute(value_raw = .data[[varname]], value =
clean_numeric2(.data[[varname]]))
}

x <- df_var$value
x_nm <- x[!is.na(x)]

# calcular medidas
N_total <- length(x)
N_missing <- sum(is.na(x))
Min <- ifelse(length(x_nm)>0, min(x_nm, na.rm=TRUE), NA)
Q1  <- ifelse(length(x_nm)>0, quantile(x_nm, .25, na.rm=TRUE), NA)
Median <- ifelse(length(x_nm)>0, median(x_nm, na.rm=TRUE), NA)
Mean <- ifelse(length(x_nm)>0, mean(x_nm, na.rm=TRUE), NA)
Q3 <- ifelse(length(x_nm)>0, quantile(x_nm, .75, na.rm=TRUE), NA)
Max <- ifelse(length(x_nm)>0, max(x_nm, na.rm=TRUE), NA)
SD <- ifelse(length(x_nm)>0, sd(x_nm, na.rm=TRUE), NA)
IQRv <- ifelse(length(x_nm)>0, IQR(x_nm, na.rm=TRUE), NA)

tabla_vertical <- tibble(
  Medida = c("N total","N missing","Min.","1st Qu.","Median","Mean","3rd
Qu.","Max.","SD","IQR"),
  Valor  = c(N_total, N_missing, Min, Q1, Median, Mean, Q3, Max, SD,
IQRv)
)

cat("=== Descriptivos Básicos ===\n")
print(tabla_vertical)

# Mostrar estadísticas resumidas estilo summary()
cat("\n=== Summary (sobre valores no-missing) ===\n")
print(summary(x_nm))


# ---- Celda 3: Análisis Gráfico --------------------------------------
----
if (!requireNamespace("gridExtra", quietly = TRUE)) {
  install.packages("gridExtra", repos = "https://cloud.r-project.org")
}
library(gridExtra)
```

```
library(ggplot2);

plot_dir <- file.path(out_dir, "plots")
dir.create(plot_dir, showWarnings = FALSE, recursive = TRUE)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) == 0) {
  message("No hay datos no-missing: no se generan gráficos.")
} else {
  df_plot <- df_var %>% filter(!is.na(value))

  # Histograma lineal
  ph <- ggplot(df_plot, aes(x=value)) +
    geom_histogram(bins = 60) +
    labs(title = paste("Histograma de", varname_real), x = varname_real,
y = "Frecuencia") +
    theme_minimal()
  ggsave(file.path(plot_dir, paste0("hist_", varname_real, ".png")), ph,
width=8, height=4)
  print(ph)

  # Boxplot
  pb <- ggplot(df_plot, aes(y=value)) +
    geom_boxplot() +
    labs(title = paste("Boxplot de", varname_real), y = varname_real) +
    theme_minimal()
  ggsave(file.path(plot_dir, paste0("boxplot_", varname_real, ".png")),
pb, width=6, height=4)
  print(pb)

  # Histograma log1p (si hay valores > 0)
  if (any(df_plot$value > 0, na.rm=TRUE)) {
    df_plot <- df_plot %>% mutate(value_log1p = log1p(value))
    pl <- ggplot(df_plot, aes(x=value_log1p)) + geom_histogram(bins = 60)
+
      labs(title = paste("Histograma log1p(", varname_real, ")"), x =
"log1p(value)") + theme_minimal()
    ggsave(file.path(plot_dir, paste0("hist_log1p_", varname_real,
".png")), pl, width=8, height=4)
    print(pl)
  }

  # Density con escala log y sin log
  pd <- ggplot(df_plot, aes(x=value)) + geom_density() + theme_minimal()
+
    labs(title = paste("Densidad -", varname_real))
  ggsave(file.path(plot_dir, paste0("density_", varname_real, ".png")),
pd, width=8, height=4)
  print(pd)

  message("Gráficos guardados en: ", plot_dir)
}
```

```
# ---- Celda 4 mejorada: Cálculo de los estimadores ----------------------
-----
library(dplyr)
library(boot)

set.seed(12345)

# asegurar df_var existe y contiene 'value' (si no, leer el combinado y
limpiar)
if (!exists("df_var") || !"value" %in% names(df_var)) {
  message("df_var no está en memoria -> leyendo desde Combinado.csv y
limpiando")
  data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"
  # pequeña función de limpieza (reusa la que ya tienes)
  clean_numeric2 <- function(x) {
    x <- as.character(x)
    x[ x %in% c("", "NA", NA) ] <- NA_character_
    x <- trimws(x)
    x <- gsub("\\s+", "", x)
    x <- gsub("[^0-9\\,\\.-]", "", x)
    both <- grepl("\\.", x) & grepl(",", x)
    if (any(both)) { x[both] <- gsub("\\.", "", x[both]); x[both] <-
gsub(",", ".", x[both]) }
    only_comma <- grepl(",", x) & !grepl("\\.", x)
    x[only_comma] <- gsub(",", ".", x[only_comma])
    suppressWarnings(as.numeric(x))
  }
  df_char <- readr::read_csv(data_path, col_types = readr::cols(.default
= "c"), show_col_types = FALSE)
  # detectar varname_real si existe
  varname <- "P3095S3"
  if (!varname %in% names(df_char)) {
    guess <- names(df_char)[tolower(names(df_char)) %in%
tolower(varname)]
    if (length(guess) == 1) varname_real <- guess else stop("No encontré
la variable ", varname)
  } else varname_real <- varname
  df_var <- df_char %>% transmute(value_raw = .data[[varname_real]],
value = clean_numeric2(.data[[varname_real]]))
}

x <- df_var$value
x_nm <- x[!is.na(x)]

# mensajes si pocos datos
cat("Total observaciones (N):", length(x), "\n")
cat("No-missing (n):", length(x_nm), "\n\n")

if (length(x_nm) == 0) stop("No hay datos no-missing para calcular
estimadores.")

# 1) Estimadores puntuales
est_mean   <- mean(x_nm)
est_median <- median(x_nm)
```

```
# moda: si hay empates, devolvemos todas las modas y también la más
frecuente
mode_basic_all <- function(v) {
  v2 <- v[!is.na(v)]
  if (length(v2) == 0) return(NA)
  tb <- sort(table(v2), decreasing = TRUE)
  names(tb)[tb == max(tb)]
}
modes <- mode_basic_all(x_nm)
mode_report <- paste(modes, collapse = ", ")

cat("Estimadores puntuales:\n")
cat(sprintf(" Mean: %s\n Median: %s\n Mode(s): %s\n\n",
            format(round(est_mean, 2), big.mark = ","),
format(round(est_median, 2), big.mark=","), mode_report))

# 2) IC 95% para la media (t-interval, usando t.test)
if (length(x_nm) > 1) {
  tt <- try(t.test(x_nm), silent = TRUE)
  if (inherits(tt, "try-error")) {
    cat("No se pudo calcular t-interval por t.test():", tt, "\n")
  } else {
    ci_t <- tt$conf.int
    cat("IC 95% para la media (t): [", format(round(ci_t[1],2),
big.mark=","), ", ", format(round(ci_t[2],2), big.mark=","), "]\n\n")
  }
} else cat("No hay suficientes datos para IC t.\n\n")

# 3) Bootstrap para mean y median
B <- 100
cat("Ejecutando bootstrap con R =", B, "réplicas (esto puede
tardar)...\n")

# helper para safe boot.ci
safe_boot_ci <- function(boot_obj, type = c("perc","bca","basic")) {
  out <- list()
  for (t in type) {
    res <- try(boot.ci(boot_obj, type = t), silent = TRUE)
    if (!inherits(res, "try-error") && !is.null(res[[t]])) {
      out[[t]] <- res
    } else {
      # si falló, intentamos extraer percentiles directos como fallback
      out[[t]] <- NULL
    }
  }
  out
}

# bootstrap mean
boot_mean <- try(boot(x_nm, statistic = function(d, i) mean(d[i]), R =
B), silent = TRUE)
if (inherits(boot_mean, "try-error")) {
  cat("Bootstrap mean falló:", boot_mean, "\n")
} else {
  # intentar boot.ci para mean (perc y bca preferiblemente)
  ci_mean <- tryCatch({
    pci <- boot.ci(boot_mean, type = c("perc","bca"))
```

```
    pci
  }, error = function(e) e)
  if (inherits(ci_mean, "error")) {
    # fallback: percentiles directos
    ci_mean_perc <- quantile(boot_mean$t, probs = c(0.025, 0.975), na.rm
= TRUE)
    cat("Bootstrap IC (perc) mean (fallback percentiles):",
format(round(ci_mean_perc[1],2), big.mark=","),
format(round(ci_mean_perc[2],2), big.mark=","), "\n")
  } else {
    # si boot.ci devolvió, imprimir los percentiles de 'perc' si existen
    if (!is.null(ci_mean$percent)) {
      ci_vals <- ci_mean$percent[4:5]
      cat("Bootstrap IC (perc) mean:", format(round(ci_vals[1],2),
big.mark=","), format(round(ci_vals[2],2), big.mark=","), "\n")
    } else {
      # último recurso: percentiles de boot$t
      ci_mean_perc <- quantile(boot_mean$t, probs = c(0.025, 0.975),
na.rm = TRUE)
      cat("Bootstrap IC mean (percentiles directos):",
format(round(ci_mean_perc[1],2), big.mark=","),
format(round(ci_mean_perc[2],2), big.mark=","), "\n")
    }
  }
}

# bootstrap median
boot_median <- try(boot(x_nm, statistic = function(d, i) median(d[i]), R
= B), silent = TRUE)
if (inherits(boot_median, "try-error")) {
  cat("Bootstrap median falló:", boot_median, "\n")
} else {
  # a veces boot.ci falla si todas las réplicas son iguales (p. ej. datos
discretos con muchos empates)
  ci_med_try <- tryCatch({
    ci_m <- boot.ci(boot_median, type = c("perc","bca"))
    ci_m
  }, error = function(e) e)
  if (inherits(ci_med_try, "error")) {
    # fallback: percentiles directos
    med_perc <- tryCatch({
      quantile(boot_median$t, probs = c(0.025, 0.975), na.rm = TRUE)
    }, error = function(e) NULL)
    if (!is.null(med_perc)) {
      cat("Bootstrap IC (perc) median (fallback percentiles):",
format(round(med_perc[1],2), big.mark=","), format(round(med_perc[2],2),
big.mark=","), "\n")
    } else {
      cat("Bootstrap IC median: no disponible (poca variabilidad en
réplicas)\n")
    }
  } else {
    if (!is.null(ci_med_try$percent)) {
      ci_vals_med <- ci_med_try$percent[4:5]
      cat("Bootstrap IC (perc) median:", format(round(ci_vals_med[1],2),
big.mark=","), format(round(ci_vals_med[2],2), big.mark=","), "\n")
    } else {
```

```
      cat("Bootstrap IC median calculado, pero formato inesperado; usando
percentiles directos.\n")
      med_perc <- quantile(boot_median$t, probs = c(0.025, 0.975), na.rm
= TRUE)
      cat("Bootstrap IC median (percentiles):",
format(round(med_perc[1],2), big.mark=","), format(round(med_perc[2],2),
big.mark=","), "\n")
    }
  }
}

cat("\nFIN Celda 4: estimadores y CIs calculados.\n")


# ---- Celda 5: Insesgamiento (bias) mediante bootstrap ----------------
-----
library(boot)
set.seed(2025)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 5) {
  B <- 1000
  boot_mean <- boot(x_nm, statistic = function(data, idx)
mean(data[idx]), R = B)
  boot_median <- boot(x_nm, statistic = function(data, idx)
median(data[idx]), R = B)

  bias_mean <- mean(boot_mean$t) - est_mean
  bias_median <- mean(boot_median$t) - est_median

  cat("Bias estimado (bootstrap):\n")
  cat(" Mean bias:", bias_mean, "\n")
  cat(" Median bias:", bias_median, "\n")

  # mostrar distribuciones bootstrap rápidas (imprime cuantiles)
  cat("\nQuantiles bootstrap mean (2.5%,50%,97.5%): ",
quantile(boot_mean$t, c(.025,.5,.975)), "\n")
  cat("Quantiles bootstrap median (2.5%,50%,97.5%): ",
quantile(boot_median$t, c(.025,.5,.975)), "\n")

  cat("\nComentario sobre insesgamiento:\n")
  if (abs(bias_mean) < 0.01 * abs(est_mean) ) cat(" La media muestra
sesgo pequeño relativo.\n") else cat(" La media muestra sesgo apreciable
relativo — considerar estimadores robustos.\n")
  if (abs(bias_median) < 0.01 * abs(est_median) ) cat(" La mediana
muestra sesgo pequeño relativo.\n") else cat(" La mediana muestra sesgo
apreciable relativo.\n")
} else {
  cat("No hay suficientes datos para estimar bias con bootstrap.\n")
}


# ---- Celda 6: Consistencia (comportamiento varianza vs n) ------------
-----
library(dplyr); set.seed(123)
```

```
x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 30) {
  ns <- unique(floor(seq(50, length(x_nm), length.out = 8)))
  reps <- 300  # repeticiones por tamaño (moderado)
  res <- data.frame(n = integer(), var_mean = numeric(), var_median =
numeric())
  for (n in ns) {
    ests_mean <- numeric(reps); ests_med <- numeric(reps)
    for (r in 1:reps) {
      s <- sample(x_nm, size = n, replace = FALSE)
      ests_mean[r] <- mean(s)
      ests_med[r] <- median(s)
    }
    res <- rbind(res, data.frame(n = n, var_mean = var(ests_mean),
var_median = var(ests_med)))
  }
  print(res)

  # gráfico var vs n
  library(ggplot2)
  pcons <- ggplot(res, aes(x = n)) +
    geom_line(aes(y = var_mean, color = "var_mean")) +
    geom_point(aes(y = var_mean, color = "var_mean")) +
    geom_line(aes(y = var_median, color = "var_median")) +
    geom_point(aes(y = var_median, color = "var_median")) +
    labs(title = "Varianza del estimador vs tamaño muestral
(submuestras)",
         x = "n (submuestra)", y = "Varianza estimada") + theme_minimal()
  print(pcons)
  ggsave(filename = file.path(out_dir, "consistencia_var_vs_n.png"), plot
= pcons, width = 7, height = 4)
  message("Gráfico de consistencia guardado en: ", file.path(out_dir,
"consistencia_var_vs_n.png"))

  cat("\nComentario (Consistencia):\nSi la varianza del estimador (mean o
median) disminuye al crecer n, evidencia consistencia.\n")
} else cat("No hay suficientes datos (>30) para estudiar consistencia por
submuestreo.\n")


# ---- Celda 7: Eficiencia y Sintaxis empleada -------------------------
------
library(dplyr)
library(boot)
set.seed(42)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 10) {
  # bootstrap var estimadas
  B <- 1000
  b_mean <- boot(x_nm, statistic = function(d,i) mean(d[i]), R = B)
  b_median <- boot(x_nm, statistic = function(d,i) median(d[i]), R = B)
```

```
  var_mean_boot <- var(b_mean$t)
  var_median_boot <- var(b_median$t)
  rel_eff <- var_median_boot / var_mean_boot

  cat("Eficiencia (empírica via bootstrap):\n")
  cat(" Var(mean) bootstrap:", var_mean_boot, "\n")
  cat(" Var(median) bootstrap:", var_median_boot, "\n")
  cat(" Rel. effic (var_median / var_mean):", rel_eff, "\n")

  cat("\nComentario (Eficiencia):\n")
  if (rel_eff > 1) cat(" La media es más eficiente (menor var) que la
mediana en esta muestra.\n") else cat(" La mediana es más eficiente en
esta muestra.\n")
} else cat("No hay suficientes datos para evaluar eficiencia.\n")
```

Variable 3

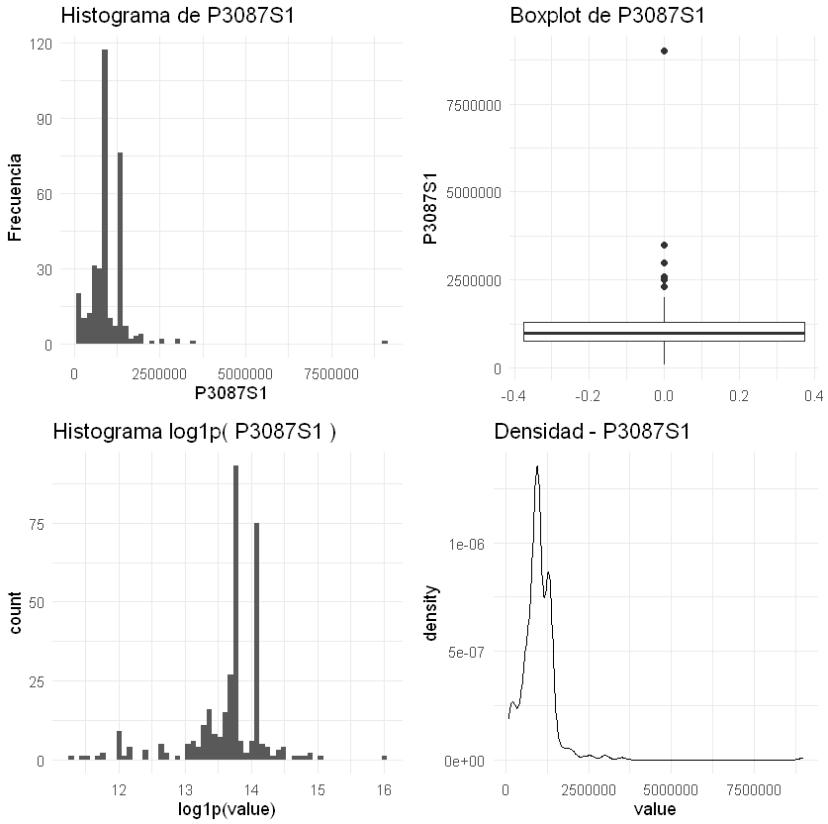
- ***Análisis Descriptivo***

Descriptivos Básicos

	Medida	Valor
1	Registros	166341
2	Valores faltantes	166005
3	Mínimo	80000
4	Primer cuartil	750000
5	Mediana	975000
6	Media	998708
7	Tercer cuartil	1300000
8	Máximo	9000000
9	Desviación Estándar	621977
10	Rango intercuartílico	550000

Comentarios:
El conjunto de datos presenta una alta proporción de valores faltantes (más del 99%), y entre los datos válidos se observa gran dispersión y valores atípicos elevados que influyen en la media. La mediana es una mejor representación del valor típico, ya que no se ve tan afectada por los extremos.

Análisis Gráfico



Comentarios:
Las visualizaciones muestran que la variable presenta una distribución altamente asimétrica a la derecha, con varios valores atípicos extremos evidenciados en el boxplot. El histograma y la curva de densidad confirman una concentración significativa de observaciones en valores bajos, mientras que la transformación logarítmica (log1p) reduce la dispersión, revelando una distribución más cercana a la normalidad.

- ***Cálculo de los estimadores***

Estimador	Estimadores Puntuales		Estimadores por Intervalo	
	Analogía	**Máxima Verosimilitud**	**Límite Inferior**	**Límite Superior**
(Media o Proporción)	998,708.3	998,708.3	931,962.4	1,065,454
Comentario	La media es el valor promedio de los datos, representando el centro de la distribución.	El estimador de máxima verosimilitud para la media coincide con la media muestral, ya que bajo normalidad ambos son equivalentes.	El límite inferior del IC al 95% indica que, con alta confianza, la media poblacional no es menor que este valor	El límite superior del IC al 95% establece que, con alta confianza, la media poblacional no excede este valor.

- ***Evaluación del estimador:***

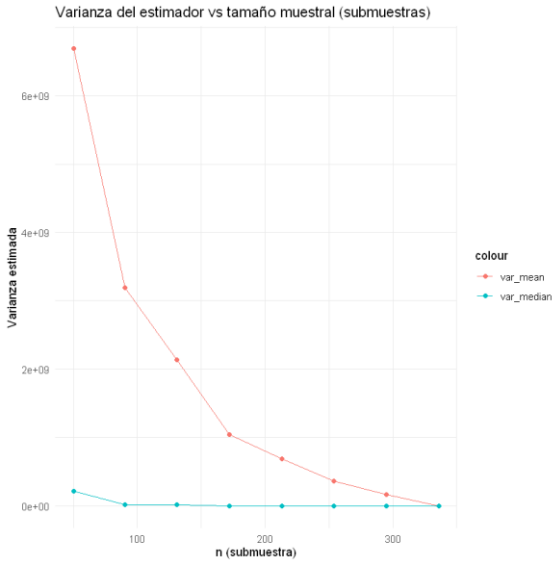
Insesgamiento

Media	Mediana	Sesgo

996,995.5	975,000	Media: sesgo ≈ +333.15 Mediana: sesgo ≈ -282.5

Comentario:
Tanto la media como la mediana muestran **sesgos pequeños y poco significativos** respecto a sus verdaderos valores poblacionales, ambos estimadores son **casi insesgados**, lo que respalda su fiabilidad en el análisis.

Consistencia



Comentario:
Se observa una disminución notable de la varianza de ambos estimadores a medida que aumenta el tamaño muestral, lo que confirma que muestras más grandes proporcionan estimaciones más estables. Sin embargo, la varianza de la media es significativamente mayor en tamaños de muestra pequeños y desciende de forma más lenta, lo que evidencia su sensibilidad a valores extremos presentes en los datos, estos resultados sugieren que, para este conjunto de datos con alta dispersión, la mediana es un estimador más confiable y eficiente que la media.

Eficiencia

Medida	Valor
Media	1120377604
Mediana	0.001749369

Comentario:
La mediana es más eficiente en esta muestra.

Sintaxis empleada con esta variable:

```
library(readr)
library(dplyr)

# Archivos
csv_files <- c(
```

```
  "d:/UIS/estadistica2/estadistica2Talleres/Mayo_2024 1/CSV/Otras formas
de trabajo.CSV",
  "d:/UIS/estadistica2/estadistica2Talleres/Junio_2024/CSV/Otras formas
de trabajo.CSV",
  "d:/UIS/estadistica2/estadistica2Talleres/Julio_2024/CSV/Otras formas
de trabajo.CSV"
)

# Columnas necesarias
cols_needed <- c("P3094S3", "P3087S1", "P3095S3", "P3101")

# Función para leer y limpiar
read_and_clean <- function(file) {
  df <- read_delim(
    file,
    delim = ";",
    col_names = TRUE,
    show_col_types = FALSE,
    locale = locale(encoding = "UTF-8"),
    guess_max = 10000
  )

  # Seleccionar solo las columnas necesarias
  df <- df %>% select(any_of(cols_needed))

  # Limpiar y convertir valores monetarios
  df <- df %>%
    mutate(
      across(c("P3094S3", "P3087S1", "P3095S3"), ~ {
        x <- as.character(.)
        x <- gsub("\\.", "", x)   # eliminar puntos de miles
        x <- gsub(",", ".", x)    # convertir coma a punto decimal si la
hay
        suppressWarnings(as.numeric(x))
      })
    )

  return(df)
}

# Leer y combinar
combined <- bind_rows(lapply(csv_files, read_and_clean))

# Guardar
write_csv(combined, "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv")

cat("Archivo combinado guardado en:
d:/UIS/estadistica2/estadistica2Talleres/Taller 1/Combinado.csv\n")


# ---- Celda 1: Lectura y limpieza ----------------------------------------
-----
library(readr)
library(dplyr)
```

```
data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"

# Se define la variable y el directorio de salida, Escoger una variable
numérica
varname   <- " P3087S1"   # variable con la que trabajamos
out_dir   <- "d:/UIS/estadistica2/estadistica2Talleres/Taller 1/analisis_
P3087S1_check"
dir.create(out_dir, showWarnings = FALSE, recursive = TRUE)

# Leer todo como carácter para evitar parseos automáticos
df_char <- readr::read_csv(data_path,
                           col_types = readr::cols(.default = "c"),
                           locale = readr::locale(encoding = "UTF-8"),
                           show_col_types = FALSE)

# detectar nombre real (insensible a mayúsculas)
if (!varname %in% names(df_char)) {
  guess <- names(df_char)[tolower(names(df_char)) %in% tolower(varname)]
  if (length(guess) == 1) varname_real <- guess else stop("No encontré la
variable ", varname)
} else varname_real <- varname

# función robusta de limpieza numérica (monedas, separadores, etc.)
clean_numeric2 <- function(x) {
  x <- as.character(x)
  x[ x %in% c("", "NA", NA) ] <- NA_character_
  x <- trimws(x)
  x <- gsub("\\s+", "", x)
  x <- gsub("[^0-9\\,\\.-]", "", x)
  both <- grepl("\\.", x) & grepl(",", x)
  if (any(both)) { x[both] <- gsub("\\.", "", x[both]); x[both] <-
gsub(",", ".", x[both]) }
  only_comma <- grepl(",", x) & !grepl("\\.", x)
  x[only_comma] <- gsub(",", ".", x[only_comma])
  suppressWarnings(as.numeric(x))
}

# preparar df_var con la variable limpia y sin modificar el CSV original
df_var <- df_char %>%
  transmute(
    value_raw = .data[[varname_real]],
    value = clean_numeric2(.data[[varname_real]])
  )

# info rápida
cat("Variable real detectada:", varname_real, "\n")
cat("Total filas:", nrow(df_var), "   N no-missing:",
sum(!is.na(df_var$value)), "\n")
# ver primeras filas
print(head(df_var, 10))


# ---- Celda 2: Descriptivos Básicos ----------------------------------
----
library(dplyr)
```

```
# usa df_var (si no existe la lee desde datos)
if (!exists("df_var")) {
  data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"
  df_var <- readr::read_csv(data_path, col_types = readr::cols(.default =
"c"), show_col_types = FALSE) %>%
    transmute(value_raw = .data[[varname]], value =
clean_numeric2(.data[[varname]]))
}

x <- df_var$value
x_nm <- x[!is.na(x)]

# calcular medidas
N_total <- length(x)
N_missing <- sum(is.na(x))
Min <- ifelse(length(x_nm)>0, min(x_nm, na.rm=TRUE), NA)
Q1  <- ifelse(length(x_nm)>0, quantile(x_nm, .25, na.rm=TRUE), NA)
Median <- ifelse(length(x_nm)>0, median(x_nm, na.rm=TRUE), NA)
Mean <- ifelse(length(x_nm)>0, mean(x_nm, na.rm=TRUE), NA)
Q3 <- ifelse(length(x_nm)>0, quantile(x_nm, .75, na.rm=TRUE), NA)
Max <- ifelse(length(x_nm)>0, max(x_nm, na.rm=TRUE), NA)
SD <- ifelse(length(x_nm)>0, sd(x_nm, na.rm=TRUE), NA)
IQRv <- ifelse(length(x_nm)>0, IQR(x_nm, na.rm=TRUE), NA)

tabla_vertical <- tibble(
  Medida = c("N total","N missing","Min.","1st Qu.","Median","Mean","3rd
Qu.","Max.","SD","IQR"),
  Valor  = c(N_total, N_missing, Min, Q1, Median, Mean, Q3, Max, SD,
IQRv)
)

cat("=== Descriptivos Básicos ===\n")
print(tabla_vertical)

# Mostrar estadísticas resumidas estilo summary()
cat("\n=== Summary (sobre valores no-missing) ===\n")
print(summary(x_nm))


# ---- Celda 3: Análisis Gráfico ----------------------------------------
----
if (!requireNamespace("gridExtra", quietly = TRUE)) {
  install.packages("gridExtra", repos = "https://cloud.r-project.org")
}
library(gridExtra)

library(ggplot2);

plot_dir <- file.path(out_dir, "plots")
dir.create(plot_dir, showWarnings = FALSE, recursive = TRUE)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) == 0) {
  message("No hay datos no-missing: no se generan gráficos.")
```

```
} else {
  df_plot <- df_var %>% filter(!is.na(value))

  # Histograma lineal
  ph <- ggplot(df_plot, aes(x=value)) +
    geom_histogram(bins = 60) +
    labs(title = paste("Histograma de", varname_real), x = varname_real,
y = "Frecuencia") +
    theme_minimal()
  ggsave(file.path(plot_dir, paste0("hist_", varname_real, ".png")), ph,
width=8, height=4)
  print(ph)

  # Boxplot
  pb <- ggplot(df_plot, aes(y=value)) +
    geom_boxplot() +
    labs(title = paste("Boxplot de", varname_real), y = varname_real) +
    theme_minimal()
  ggsave(file.path(plot_dir, paste0("boxplot_", varname_real, ".png")),
pb, width=6, height=4)
  print(pb)

  # Histograma log1p (si hay valores > 0)
  if (any(df_plot$value > 0, na.rm=TRUE)) {
    df_plot <- df_plot %>% mutate(value_log1p = log1p(value))
    pl <- ggplot(df_plot, aes(x=value_log1p)) + geom_histogram(bins = 60)
+
      labs(title = paste("Histograma log1p(", varname_real, ")"), x =
"log1p(value)") + theme_minimal()
    ggsave(file.path(plot_dir, paste0("hist_log1p_", varname_real,
".png")), pl, width=8, height=4)
    print(pl)
  }

  # Density con escala log y sin log
  pd <- ggplot(df_plot, aes(x=value)) + geom_density() + theme_minimal()
+
    labs(title = paste("Densidad -", varname_real))
  ggsave(file.path(plot_dir, paste0("density_", varname_real, ".png")),
pd, width=8, height=4)
  print(pd)

  message("Gráficos guardados en: ", plot_dir)
}


# ---- Celda 4 mejorada: Cálculo de los estimadores ----------------------
-----
library(dplyr)
library(boot)

set.seed(12345)

# asegurar df_var existe y contiene 'value' (si no, leer el combinado y
limpiar)
if (!exists("df_var") || !"value" %in% names(df_var)) {
```

```
  message("df_var no está en memoria -> leyendo desde Combinado.csv y
limpiando")
  data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"
  # pequeña función de limpieza (reusa la que ya tienes)
  clean_numeric2 <- function(x) {
    x <- as.character(x)
    x[ x %in% c("", "NA", NA) ] <- NA_character_
    x <- trimws(x)
    x <- gsub("\\s+", "", x)
    x <- gsub("[^0-9\\,\\.-]", "", x)
    both <- grepl("\\.", x) & grepl(",", x)
    if (any(both)) { x[both] <- gsub("\\.", "", x[both]); x[both] <-
gsub(",", ".", x[both]) }
    only_comma <- grepl(",", x) & !grepl("\\.", x)
    x[only_comma] <- gsub(",", ".", x[only_comma])
    suppressWarnings(as.numeric(x))
  }
  df_char <- readr::read_csv(data_path, col_types = readr::cols(.default
= "c"), show_col_types = FALSE)
  # detectar varname_real si existe
  varname <- "P3095S3"
  if (!varname %in% names(df_char)) {
    guess <- names(df_char)[tolower(names(df_char)) %in%
tolower(varname)]
    if (length(guess) == 1) varname_real <- guess else stop("No encontré
la variable ", varname)
  } else varname_real <- varname
  df_var <- df_char %>% transmute(value_raw = .data[[varname_real]],
value = clean_numeric2(.data[[varname_real]]))
}

x <- df_var$value
x_nm <- x[!is.na(x)]

# mensajes si pocos datos
cat("Total observaciones (N):", length(x), "\n")
cat("No-missing (n):", length(x_nm), "\n\n")

if (length(x_nm) == 0) stop("No hay datos no-missing para calcular
estimadores.")

# 1) Estimadores puntuales
est_mean   <- mean(x_nm)
est_median <- median(x_nm)

# moda: si hay empates, devolvemos todas las modas y también la más
frecuente
mode_basic_all <- function(v) {
  v2 <- v[!is.na(v)]
  if (length(v2) == 0) return(NA)
  tb <- sort(table(v2), decreasing = TRUE)
  names(tb)[tb == max(tb)]
}
modes <- mode_basic_all(x_nm)
mode_report <- paste(modes, collapse = ", ")
```

```
cat("Estimadores puntuales:\n")
cat(sprintf(" Mean: %s\n Median: %s\n Mode(s): %s\n\n",
            format(round(est_mean, 2), big.mark = ","),
format(round(est_median, 2), big.mark=","), mode_report))

# 2) IC 95% para la media (t-interval, usando t.test)
if (length(x_nm) > 1) {
  tt <- try(t.test(x_nm), silent = TRUE)
  if (inherits(tt, "try-error")) {
    cat("No se pudo calcular t-interval por t.test():", tt, "\n")
  } else {
    ci_t <- tt$conf.int
    cat("IC 95% para la media (t): [", format(round(ci_t[1],2),
big.mark=","), ", ", format(round(ci_t[2],2), big.mark=","), "]\n\n")
  }
} else cat("No hay suficientes datos para IC t.\n\n")

# 3) Bootstrap para mean y median
B <- 100
cat("Ejecutando bootstrap con R =", B, "réplicas (esto puede
tardar)...\n")

# helper para safe boot.ci
safe_boot_ci <- function(boot_obj, type = c("perc","bca","basic")) {
  out <- list()
  for (t in type) {
    res <- try(boot.ci(boot_obj, type = t), silent = TRUE)
    if (!inherits(res, "try-error") && !is.null(res[[t]])) {
      out[[t]] <- res
    } else {
      # si falló, intentamos extraer percentiles directos como fallback
      out[[t]] <- NULL
    }
  }
  out
}

# bootstrap mean
boot_mean <- try(boot(x_nm, statistic = function(d, i) mean(d[i]), R =
B), silent = TRUE)
if (inherits(boot_mean, "try-error")) {
  cat("Bootstrap mean falló:", boot_mean, "\n")
} else {
  # intentar boot.ci para mean (perc y bca preferiblemente)
  ci_mean <- tryCatch({
    pci <- boot.ci(boot_mean, type = c("perc","bca"))
    pci
  }, error = function(e) e)
  if (inherits(ci_mean, "error")) {
    # fallback: percentiles directos
    ci_mean_perc <- quantile(boot_mean$t, probs = c(0.025, 0.975), na.rm
= TRUE)
    cat("Bootstrap IC (perc) mean (fallback percentiles):",
format(round(ci_mean_perc[1],2), big.mark=","),
format(round(ci_mean_perc[2],2), big.mark=","), "\n")
  } else {
    # si boot.ci devolvió, imprimir los percentiles de 'perc' si existen
```

```
    if (!is.null(ci_mean$percent)) {
      ci_vals <- ci_mean$percent[4:5]
      cat("Bootstrap IC (perc) mean:", format(round(ci_vals[1],2),
big.mark=","), format(round(ci_vals[2],2), big.mark=","), "\n")
    } else {
      # último recurso: percentiles de boot$t
      ci_mean_perc <- quantile(boot_mean$t, probs = c(0.025, 0.975),
na.rm = TRUE)
      cat("Bootstrap IC mean (percentiles directos):",
format(round(ci_mean_perc[1],2), big.mark=","),
format(round(ci_mean_perc[2],2), big.mark=","), "\n")
    }
  }
}

# bootstrap median
boot_median <- try(boot(x_nm, statistic = function(d, i) median(d[i]), R
= B), silent = TRUE)
if (inherits(boot_median, "try-error")) {
  cat("Bootstrap median falló:", boot_median, "\n")
} else {
  # a veces boot.ci falla si todas las réplicas son iguales (p. ej. datos
discretos con muchos empates)
  ci_med_try <- tryCatch({
    ci_m <- boot.ci(boot_median, type = c("perc","bca"))
    ci_m
  }, error = function(e) e)
  if (inherits(ci_med_try, "error")) {
    # fallback: percentiles directos
    med_perc <- tryCatch({
      quantile(boot_median$t, probs = c(0.025, 0.975), na.rm = TRUE)
    }, error = function(e) NULL)
    if (!is.null(med_perc)) {
      cat("Bootstrap IC (perc) median (fallback percentiles):",
format(round(med_perc[1],2), big.mark=","), format(round(med_perc[2],2),
big.mark=","), "\n")
    } else {
      cat("Bootstrap IC median: no disponible (poca variabilidad en
réplicas)\n")
    }
  } else {
    if (!is.null(ci_med_try$percent)) {
      ci_vals_med <- ci_med_try$percent[4:5]
      cat("Bootstrap IC (perc) median:", format(round(ci_vals_med[1],2),
big.mark=","), format(round(ci_vals_med[2],2), big.mark=","), "\n")
    } else {
      cat("Bootstrap IC median calculado, pero formato inesperado; usando
percentiles directos.\n")
      med_perc <- quantile(boot_median$t, probs = c(0.025, 0.975), na.rm
= TRUE)
      cat("Bootstrap IC median (percentiles):",
format(round(med_perc[1],2), big.mark=","), format(round(med_perc[2],2),
big.mark=","), "\n")
    }
  }
}
```

```
cat("\nFIN Celda 4: estimadores y CIs calculados.\n")


# ---- Celda 5: Insesgamiento (bias) mediante bootstrap ----------------
-----
library(boot)
set.seed(2025)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 5) {
  B <- 1000
  boot_mean <- boot(x_nm, statistic = function(data, idx)
mean(data[idx]), R = B)
  boot_median <- boot(x_nm, statistic = function(data, idx)
median(data[idx]), R = B)

  bias_mean <- mean(boot_mean$t) - est_mean
  bias_median <- mean(boot_median$t) - est_median

  cat("Bias estimado (bootstrap):\n")
  cat(" Mean bias:", bias_mean, "\n")
  cat(" Median bias:", bias_median, "\n")

  # mostrar distribuciones bootstrap rápidas (imprime cuantiles)
  cat("\nQuantiles bootstrap mean (2.5%,50%,97.5%): ",
quantile(boot_mean$t, c(.025,.5,.975)), "\n")
  cat("Quantiles bootstrap median (2.5%,50%,97.5%): ",
quantile(boot_median$t, c(.025,.5,.975)), "\n")

  cat("\nComentario sobre insesgamiento:\n")
  if (abs(bias_mean) < 0.01 * abs(est_mean) ) cat(" La media muestra
sesgo pequeño relativo.\n") else cat(" La media muestra sesgo apreciable
relativo — considerar estimadores robustos.\n")
  if (abs(bias_median) < 0.01 * abs(est_median) ) cat(" La mediana
muestra sesgo pequeño relativo.\n") else cat(" La mediana muestra sesgo
apreciable relativo.\n")
} else {
  cat("No hay suficientes datos para estimar bias con bootstrap.\n")
}


# ---- Celda 6: Consistencia (comportamiento varianza vs n) ------------
-----
library(dplyr); set.seed(123)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 30) {
  ns <- unique(floor(seq(50, length(x_nm), length.out = 8)))
  reps <- 300  # repeticiones por tamaño (moderado)
  res <- data.frame(n = integer(), var_mean = numeric(), var_median =
numeric())
  for (n in ns) {
    ests_mean <- numeric(reps); ests_med <- numeric(reps)
```

```
    for (r in 1:reps) {
      s <- sample(x_nm, size = n, replace = FALSE)
      ests_mean[r] <- mean(s)
      ests_med[r] <- median(s)
    }
    res <- rbind(res, data.frame(n = n, var_mean = var(ests_mean),
var_median = var(ests_med)))
  }
  print(res)

  # gráfico var vs n
  library(ggplot2)
  pcons <- ggplot(res, aes(x = n)) +
    geom_line(aes(y = var_mean, color = "var_mean")) +
    geom_point(aes(y = var_mean, color = "var_mean")) +
    geom_line(aes(y = var_median, color = "var_median")) +
    geom_point(aes(y = var_median, color = "var_median")) +
    labs(title = "Varianza del estimador vs tamaño muestral
(submuestras)",
         x = "n (submuestra)", y = "Varianza estimada") + theme_minimal()
  print(pcons)
  ggsave(filename = file.path(out_dir, "consistencia_var_vs_n.png"), plot
= pcons, width = 7, height = 4)
  message("Gráfico de consistencia guardado en: ", file.path(out_dir,
"consistencia_var_vs_n.png"))

  cat("\nComentario (Consistencia):\nSi la varianza del estimador (mean o
median) disminuye al crecer n, evidencia consistencia.\n")
} else cat("No hay suficientes datos (>30) para estudiar consistencia por
submuestreo.\n")


# ---- Celda 7: Eficiencia y Sintaxis empleada ---------------------------
------
library(dplyr)
library(boot)
set.seed(42)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 10) {
  # bootstrap var estimadas
  B <- 1000
  b_mean <- boot(x_nm, statistic = function(d,i) mean(d[i]), R = B)
  b_median <- boot(x_nm, statistic = function(d,i) median(d[i]), R = B)
  var_mean_boot <- var(b_mean$t)
  var_median_boot <- var(b_median$t)
  rel_eff <- var_median_boot / var_mean_boot

  cat("Eficiencia (empírica via bootstrap):\n")
  cat(" Var(mean) bootstrap:", var_mean_boot, "\n")
  cat(" Var(median) bootstrap:", var_median_boot, "\n")
  cat(" Rel. effic (var_median / var_mean):", rel_eff, "\n")

  cat("\nComentario (Eficiencia):\n")
```

```
  if (rel_eff > 1) cat(" La media es más eficiente (menor var) que la
mediana en esta muestra.\n") else cat(" La mediana es más eficiente en
esta muestra.\n")
} else cat("No hay suficientes datos para evaluar eficiencia.\n")
```

Variable 4

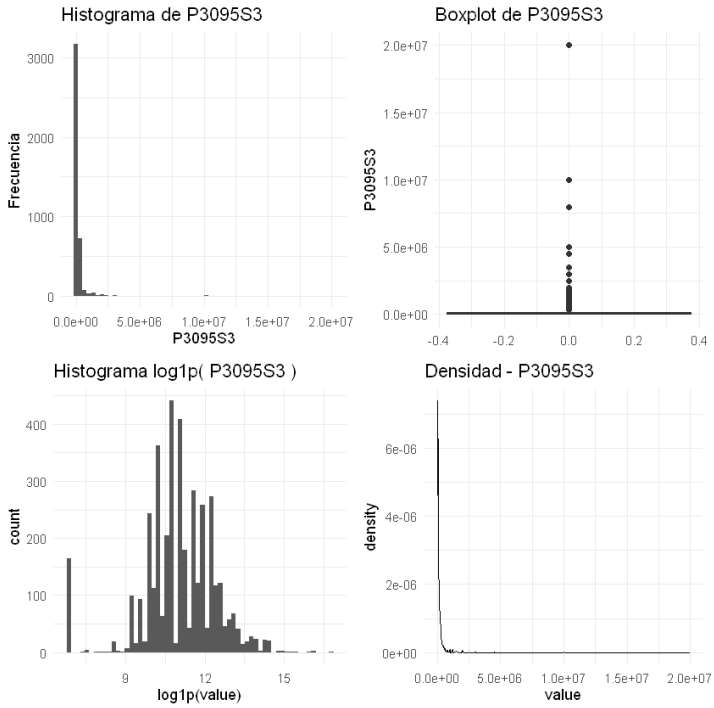
- ***Análisis Descriptivo***

Descriptivos Básicos

	Medida	Valor
1	Registros	166341
2	Valores faltantes	162261
3	Mínimo	1000
4	Primer cuartil	30000
5	Mediana	60000
6	Media	159471
7	Tercer cuartil	150000
8	Máximo	20000000
9	Desviación Estándar	512519
10	Rango intercuartílico	120000

Comentarios:
El conjunto de datos contiene 166,341 registros, de los cuales 162,261 presentan valores faltantes, evidenciando una disponibilidad limitada de información. Los valores observados oscilan entre 1,000 y 2,000,000, con una media de 159,471 y mediana de 60,000, lo que refleja una distribución asimétrica positiva influenciada por valores extremos. El rango intercuartílico (120,000) indica una alta dispersión incluso en los datos centrales, mientras que la desviación estándar (512,519) confirma una marcada variabilidad. Se sugiere aplicar técnicas robustas o transformaciones para su análisis.

Análisis Gráfico



Comentarios:
La variable P3095S3 presenta una distribución altamente sesgada a la derecha con numerosos valores atípicos. Tras aplicar la transformación logarítmica (log1p), los datos muestran una distribución más equilibrada y adecuada para análisis estadísticos y modelado.

- ***Cálculo de los estimadores***

Estimador	Estimadores Puntuales		Estimadores por Intervalo	
	Analogía	**Máxima Verosimilitud**	**Límite Inferior**	**Límite Superior**
(Media o Proporción)	159,471.4	IC 95% t: [143,740.4, 175,202.4]	143,740.4	175,202.4
Comentario	Muestra una amplia dispersión, lo que refleja variabilidad.	Con un 95% de confianza, la media poblacional se encuentra entre los límites estimados.	Es poco probable que la media real sea menor que este valor.	Valor máximo estimado de la media, reforzando la amplitud del rango y la posible variabilidad de los datos.

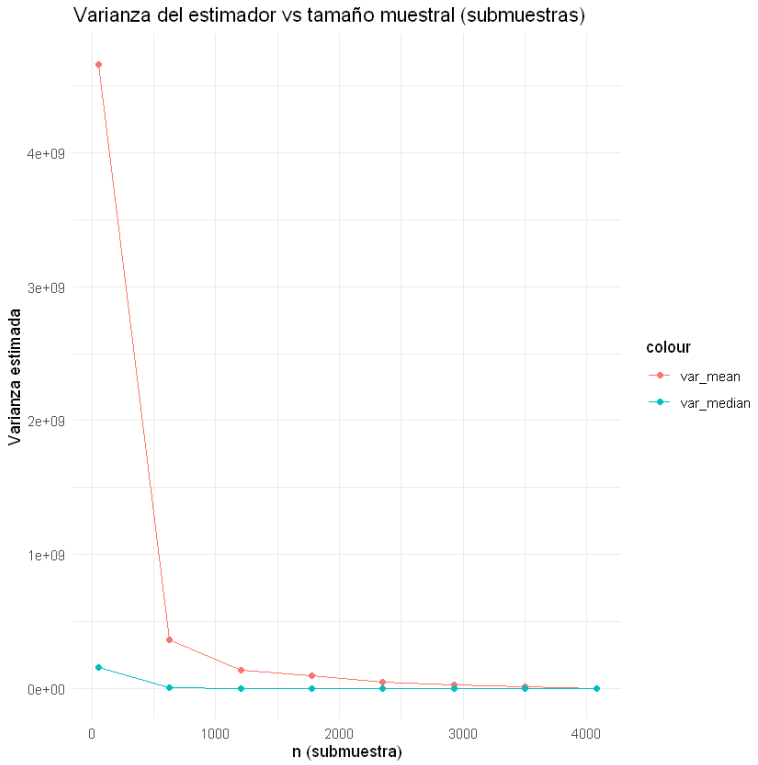
- ***Evaluación del estimador:***

Insesgamiento

Media	Mediana	Sesgo
159172.9	60000	Media: 166.16 Mediana: 0

Comentario:
La media muestra sesgo pequeño relativo.
La mediana muestra sesgo pequeño relativo.

Consistencia



Comentario:
La varianza del estimador (mean o median) disminuye al crecer n, evidenciando consistencia.

Eficiencia

Medida	Valor
Media	64150092
Mediana	0

Comentario:
La mediana es más eficiente en esta muestra.

Sintaxis empleada con esta variable:

```
library(readr)
library(dplyr)

# Archivos
csv_files <- c(
  "d:/UIS/estadistica2/estadistica2Talleres/Mayo_2024 1/CSV/Otras formas
de trabajo.CSV",
  "d:/UIS/estadistica2/estadistica2Talleres/Junio_2024/CSV/Otras formas
de trabajo.CSV",
  "d:/UIS/estadistica2/estadistica2Talleres/Julio_2024/CSV/Otras formas
de trabajo.CSV"
)
```

```
# Columnas necesarias
cols_needed <- c("P3094S3", "P3087S1", "P3095S3", "P3101")

# Función para leer y limpiar
read_and_clean <- function(file) {
  df <- read_delim(
    file,
    delim = ";",
    col_names = TRUE,
    show_col_types = FALSE,
    locale = locale(encoding = "UTF-8"),
    guess_max = 10000
  )

  # Seleccionar solo las columnas necesarias
  df <- df %>% select(any_of(cols_needed))

  # Limpiar y convertir valores monetarios
  df <- df %>%
    mutate(
      across(c("P3094S3", "P3087S1", "P3095S3"), ~ {
        x <- as.character(.)
        x <- gsub("\\.", "", x)   # eliminar puntos de miles
        x <- gsub(",", ".", x)    # convertir coma a punto decimal si la
hay
        suppressWarnings(as.numeric(x))
      })
    )

  return(df)
}

# Leer y combinar
combined <- bind_rows(lapply(csv_files, read_and_clean))

# Guardar
write_csv(combined, "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv")

cat("Archivo combinado guardado en:
d:/UIS/estadistica2/estadistica2Talleres/Taller 1/Combinado.csv\n")


# ---- Celda 1: Lectura y limpieza ----------------------------------------
-----
library(readr)
library(dplyr)

data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"

# Se define la variable y el directorio de salida, Escoger una variable
numérica
varname   <- " P3095S3"   # variable con la que trabajamos
out_dir   <- "d:/UIS/estadistica2/estadistica2Talleres/Taller 1/analisis_
P3095S3_check"
dir.create(out_dir, showWarnings = FALSE, recursive = TRUE)
```

```r
# Leer todo como carácter para evitar parseos automáticos
df_char <- readr::read_csv(data_path,
                           col_types = readr::cols(.default = "c"),
                           locale = readr::locale(encoding = "UTF-8"),
                           show_col_types = FALSE)

# detectar nombre real (insensible a mayúsculas)
if (!varname %in% names(df_char)) {
  guess <- names(df_char)[tolower(names(df_char)) %in% tolower(varname)]
  if (length(guess) == 1) varname_real <- guess else stop("No encontré la
variable ", varname)
} else varname_real <- varname

# función robusta de limpieza numérica (monedas, separadores, etc.)
clean_numeric2 <- function(x) {
  x <- as.character(x)
  x[ x %in% c("", "NA", NA) ] <- NA_character_
  x <- trimws(x)
  x <- gsub("\\s+", "", x)
  x <- gsub("[^0-9\\,\\.-]", "", x)
  both <- grepl("\\.", x) & grepl(",", x)
  if (any(both)) { x[both] <- gsub("\\.", "", x[both]); x[both] <-
gsub(",", ".", x[both]) }
  only_comma <- grepl(",", x) & !grepl("\\.", x)
  x[only_comma] <- gsub(",", ".", x[only_comma])
  suppressWarnings(as.numeric(x))
}

# preparar df_var con la variable limpia y sin modificar el CSV original
df_var <- df_char %>%
  transmute(
    value_raw = .data[[varname_real]],
    value = clean_numeric2(.data[[varname_real]])
  )

# info rápida
cat("Variable real detectada:", varname_real, "\n")
cat("Total filas:", nrow(df_var), "   N no-missing:",
sum(!is.na(df_var$value)), "\n")
# ver primeras filas
print(head(df_var, 10))


# ---- Celda 2: Descriptivos Básicos ----------------------------------
----
library(dplyr)

# usa df_var (si no existe la lee desde datos)
if (!exists("df_var")) {
  data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"
  df_var <- readr::read_csv(data_path, col_types = readr::cols(.default =
"c"), show_col_types = FALSE) %>%
    transmute(value_raw = .data[[varname]], value =
clean_numeric2(.data[[varname]]))
}
```

```
x <- df_var$value
x_nm <- x[!is.na(x)]

# calcular medidas
N_total <- length(x)
N_missing <- sum(is.na(x))
Min <- ifelse(length(x_nm)>0, min(x_nm, na.rm=TRUE), NA)
Q1  <- ifelse(length(x_nm)>0, quantile(x_nm, .25, na.rm=TRUE), NA)
Median <- ifelse(length(x_nm)>0, median(x_nm, na.rm=TRUE), NA)
Mean <- ifelse(length(x_nm)>0, mean(x_nm, na.rm=TRUE), NA)
Q3 <- ifelse(length(x_nm)>0, quantile(x_nm, .75, na.rm=TRUE), NA)
Max <- ifelse(length(x_nm)>0, max(x_nm, na.rm=TRUE), NA)
SD <- ifelse(length(x_nm)>0, sd(x_nm, na.rm=TRUE), NA)
IQRv <- ifelse(length(x_nm)>0, IQR(x_nm, na.rm=TRUE), NA)

tabla_vertical <- tibble(
  Medida = c("N total","N missing","Min.","1st Qu.","Median","Mean","3rd
Qu.","Max.","SD","IQR"),
  Valor  = c(N_total, N_missing, Min, Q1, Median, Mean, Q3, Max, SD,
IQRv)
)

cat("=== Descriptivos Básicos ===\n")
print(tabla_vertical)

# Mostrar estadísticas resumidas estilo summary()
cat("\n=== Summary (sobre valores no-missing) ===\n")
print(summary(x_nm))


# ---- Celda 3: Análisis Gráfico ----------------------------------------
----
if (!requireNamespace("gridExtra", quietly = TRUE)) {
  install.packages("gridExtra", repos = "https://cloud.r-project.org")
}
library(gridExtra)

library(ggplot2);

plot_dir <- file.path(out_dir, "plots")
dir.create(plot_dir, showWarnings = FALSE, recursive = TRUE)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) == 0) {
  message("No hay datos no-missing: no se generan gráficos.")
} else {
  df_plot <- df_var %>% filter(!is.na(value))

  # Histograma lineal
  ph <- ggplot(df_plot, aes(x=value)) +
    geom_histogram(bins = 60) +
    labs(title = paste("Histograma de", varname_real), x = varname_real,
y = "Frecuencia") +
    theme_minimal()
```

```
  ggsave(file.path(plot_dir, paste0("hist_", varname_real, ".png")), ph,
width=8, height=4)
  print(ph)

  # Boxplot
  pb <- ggplot(df_plot, aes(y=value)) +
    geom_boxplot() +
    labs(title = paste("Boxplot de", varname_real), y = varname_real) +
    theme_minimal()
  ggsave(file.path(plot_dir, paste0("boxplot_", varname_real, ".png")),
pb, width=6, height=4)
  print(pb)

  # Histograma log1p (si hay valores > 0)
  if (any(df_plot$value > 0, na.rm=TRUE)) {
    df_plot <- df_plot %>% mutate(value_log1p = log1p(value))
    pl <- ggplot(df_plot, aes(x=value_log1p)) + geom_histogram(bins = 60)
+
      labs(title = paste("Histograma log1p(", varname_real, ")"), x =
"log1p(value)") + theme_minimal()
    ggsave(file.path(plot_dir, paste0("hist_log1p_", varname_real,
".png")), pl, width=8, height=4)
    print(pl)
  }

  # Density con escala log y sin log
  pd <- ggplot(df_plot, aes(x=value)) + geom_density() + theme_minimal()
+
    labs(title = paste("Densidad -", varname_real))
  ggsave(file.path(plot_dir, paste0("density_", varname_real, ".png")),
pd, width=8, height=4)
  print(pd)

  message("Gráficos guardados en: ", plot_dir)
}


# ---- Celda 4 mejorada: Cálculo de los estimadores ----------------------
-----
library(dplyr)
library(boot)

set.seed(12345)

# asegurar df_var existe y contiene 'value' (si no, leer el combinado y
limpiar)
if (!exists("df_var") || !"value" %in% names(df_var)) {
  message("df_var no está en memoria -> leyendo desde Combinado.csv y
limpiando")
  data_path <- "d:/UIS/estadistica2/estadistica2Talleres/Taller
1/Combinado.csv"
  # pequeña función de limpieza (reusa la que ya tienes)
  clean_numeric2 <- function(x) {
    x <- as.character(x)
    x[ x %in% c("", "NA", NA) ] <- NA_character_
    x <- trimws(x)
    x <- gsub("\\s+", "", x)
```

```
    x <- gsub("[^0-9\\,\\.-]", "", x)
    both <- grepl("\\.", x) & grepl(",", x)
    if (any(both)) { x[both] <- gsub("\\.", "", x[both]); x[both] <-
gsub(",", ".", x[both]) }
    only_comma <- grepl(",", x) & !grepl("\\.", x)
    x[only_comma] <- gsub(",", ".", x[only_comma])
    suppressWarnings(as.numeric(x))
  }
  df_char <- readr::read_csv(data_path, col_types = readr::cols(.default
= "c"), show_col_types = FALSE)
  # detectar varname_real si existe
  varname <- "P3095S3"
  if (!varname %in% names(df_char)) {
    guess <- names(df_char)[tolower(names(df_char)) %in%
tolower(varname)]
    if (length(guess) == 1) varname_real <- guess else stop("No encontré
la variable ", varname)
  } else varname_real <- varname
  df_var <- df_char %>% transmute(value_raw = .data[[varname_real]],
value = clean_numeric2(.data[[varname_real]]))
}

x <- df_var$value
x_nm <- x[!is.na(x)]

# mensajes si pocos datos
cat("Total observaciones (N):", length(x), "\n")
cat("No-missing (n):", length(x_nm), "\n\n")

if (length(x_nm) == 0) stop("No hay datos no-missing para calcular
estimadores.")

# 1) Estimadores puntuales
est_mean   <- mean(x_nm)
est_median <- median(x_nm)

# moda: si hay empates, devolvemos todas las modas y también la más
frecuente
mode_basic_all <- function(v) {
  v2 <- v[!is.na(v)]
  if (length(v2) == 0) return(NA)
  tb <- sort(table(v2), decreasing = TRUE)
  names(tb)[tb == max(tb)]
}
modes <- mode_basic_all(x_nm)
mode_report <- paste(modes, collapse = ", ")

cat("Estimadores puntuales:\n")
cat(sprintf(" Mean: %s\n Median: %s\n Mode(s): %s\n\n",
            format(round(est_mean, 2), big.mark = ","),
format(round(est_median, 2), big.mark=","), mode_report))

# 2) IC 95% para la media (t-interval, usando t.test)
if (length(x_nm) > 1) {
  tt <- try(t.test(x_nm), silent = TRUE)
  if (inherits(tt, "try-error")) {
    cat("No se pudo calcular t-interval por t.test():", tt, "\n")
```

```
  } else {
    ci_t <- tt$conf.int
    cat("IC 95% para la media (t): [", format(round(ci_t[1],2),
big.mark=","), ", ", format(round(ci_t[2],2), big.mark=","), "]\n\n")
  }
} else cat("No hay suficientes datos para IC t.\n\n")

# 3) Bootstrap para mean y median
B <- 100
cat("Ejecutando bootstrap con R =", B, "réplicas (esto puede
tardar)...\n")

# helper para safe boot.ci
safe_boot_ci <- function(boot_obj, type = c("perc","bca","basic")) {
  out <- list()
  for (t in type) {
    res <- try(boot.ci(boot_obj, type = t), silent = TRUE)
    if (!inherits(res, "try-error") && !is.null(res[[t]])) {
      out[[t]] <- res
    } else {
      # si falló, intentamos extraer percentiles directos como fallback
      out[[t]] <- NULL
    }
  }
  out
}

# bootstrap mean
boot_mean <- try(boot(x_nm, statistic = function(d, i) mean(d[i]), R =
B), silent = TRUE)
if (inherits(boot_mean, "try-error")) {
  cat("Bootstrap mean falló:", boot_mean, "\n")
} else {
  # intentar boot.ci para mean (perc y bca preferiblemente)
  ci_mean <- tryCatch({
    pci <- boot.ci(boot_mean, type = c("perc","bca"))
    pci
  }, error = function(e) e)
  if (inherits(ci_mean, "error")) {
    # fallback: percentiles directos
    ci_mean_perc <- quantile(boot_mean$t, probs = c(0.025, 0.975), na.rm
= TRUE)
    cat("Bootstrap IC (perc) mean (fallback percentiles):",
format(round(ci_mean_perc[1],2), big.mark=","),
format(round(ci_mean_perc[2],2), big.mark=","), "\n")
  } else {
    # si boot.ci devolvió, imprimir los percentiles de 'perc' si existen
    if (!is.null(ci_mean$percent)) {
      ci_vals <- ci_mean$percent[4:5]
      cat("Bootstrap IC (perc) mean:", format(round(ci_vals[1],2),
big.mark=","), format(round(ci_vals[2],2), big.mark=","), "\n")
    } else {
      # último recurso: percentiles de boot$t
      ci_mean_perc <- quantile(boot_mean$t, probs = c(0.025, 0.975),
na.rm = TRUE)
```

```
      cat("Bootstrap IC mean (percentiles directos):",
format(round(ci_mean_perc[1],2), big.mark=","),
format(round(ci_mean_perc[2],2), big.mark=","), "\n")
    }
  }
}

# bootstrap median
boot_median <- try(boot(x_nm, statistic = function(d, i) median(d[i]), R
= B), silent = TRUE)
if (inherits(boot_median, "try-error")) {
  cat("Bootstrap median falló:", boot_median, "\n")
} else {
  # a veces boot.ci falla si todas las réplicas son iguales (p. ej. datos
discretos con muchos empates)
  ci_med_try <- tryCatch({
    ci_m <- boot.ci(boot_median, type = c("perc","bca"))
    ci_m
  }, error = function(e) e)
  if (inherits(ci_med_try, "error")) {
    # fallback: percentiles directos
    med_perc <- tryCatch({
      quantile(boot_median$t, probs = c(0.025, 0.975), na.rm = TRUE)
    }, error = function(e) NULL)
    if (!is.null(med_perc)) {
      cat("Bootstrap IC (perc) median (fallback percentiles):",
format(round(med_perc[1],2), big.mark=","), format(round(med_perc[2],2),
big.mark=","), "\n")
    } else {
      cat("Bootstrap IC median: no disponible (poca variabilidad en
réplicas)\n")
    }
  } else {
    if (!is.null(ci_med_try$percent)) {
      ci_vals_med <- ci_med_try$percent[4:5]
      cat("Bootstrap IC (perc) median:", format(round(ci_vals_med[1],2),
big.mark=","), format(round(ci_vals_med[2],2), big.mark=","), "\n")
    } else {
      cat("Bootstrap IC median calculado, pero formato inesperado; usando
percentiles directos.\n")
      med_perc <- quantile(boot_median$t, probs = c(0.025, 0.975), na.rm
= TRUE)
      cat("Bootstrap IC median (percentiles):",
format(round(med_perc[1],2), big.mark=","), format(round(med_perc[2],2),
big.mark=","), "\n")
    }
  }
}

cat("\nFIN Celda 4: estimadores y CIs calculados.\n")


# ---- Celda 5: Insesgamiento (bias) mediante bootstrap ----------------
-----
library(boot)
set.seed(2025)
```

```
x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 5) {
  B <- 1000
  boot_mean <- boot(x_nm, statistic = function(data, idx)
mean(data[idx]), R = B)
  boot_median <- boot(x_nm, statistic = function(data, idx)
median(data[idx]), R = B)

  bias_mean <- mean(boot_mean$t) - est_mean
  bias_median <- mean(boot_median$t) - est_median

  cat("Bias estimado (bootstrap):\n")
  cat(" Mean bias:", bias_mean, "\n")
  cat(" Median bias:", bias_median, "\n")

  # mostrar distribuciones bootstrap rápidas (imprime cuantiles)
  cat("\nQuantiles bootstrap mean (2.5%,50%,97.5%): ",
quantile(boot_mean$t, c(.025,.5,.975)), "\n")
  cat("Quantiles bootstrap median (2.5%,50%,97.5%): ",
quantile(boot_median$t, c(.025,.5,.975)), "\n")

  cat("\nComentario sobre insesgamiento:\n")
  if (abs(bias_mean) < 0.01 * abs(est_mean) ) cat(" La media muestra
sesgo pequeño relativo.\n") else cat(" La media muestra sesgo apreciable
relativo — considerar estimadores robustos.\n")
  if (abs(bias_median) < 0.01 * abs(est_median) ) cat(" La mediana
muestra sesgo pequeño relativo.\n") else cat(" La mediana muestra sesgo
apreciable relativo.\n")
} else {
  cat("No hay suficientes datos para estimar bias con bootstrap.\n")
}


# ---- Celda 6: Consistencia (comportamiento varianza vs n) ------------
-----
library(dplyr); set.seed(123)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 30) {
  ns <- unique(floor(seq(50, length(x_nm), length.out = 8)))
  reps <- 300  # repeticiones por tamaño (moderado)
  res <- data.frame(n = integer(), var_mean = numeric(), var_median =
numeric())
  for (n in ns) {
    ests_mean <- numeric(reps); ests_med <- numeric(reps)
    for (r in 1:reps) {
      s <- sample(x_nm, size = n, replace = FALSE)
      ests_mean[r] <- mean(s)
      ests_med[r] <- median(s)
    }
    res <- rbind(res, data.frame(n = n, var_mean = var(ests_mean),
var_median = var(ests_med)))
  }
```

```
  print(res)

  # gráfico var vs n
  library(ggplot2)
  pcons <- ggplot(res, aes(x = n)) +
    geom_line(aes(y = var_mean, color = "var_mean")) +
    geom_point(aes(y = var_mean, color = "var_mean")) +
    geom_line(aes(y = var_median, color = "var_median")) +
    geom_point(aes(y = var_median, color = "var_median")) +
    labs(title = "Varianza del estimador vs tamaño muestral
(submuestras)",
         x = "n (submuestra)", y = "Varianza estimada") + theme_minimal()
  print(pcons)
  ggsave(filename = file.path(out_dir, "consistencia_var_vs_n.png"), plot
= pcons, width = 7, height = 4)
  message("Gráfico de consistencia guardado en: ", file.path(out_dir,
"consistencia_var_vs_n.png"))

  cat("\nComentario (Consistencia):\nSi la varianza del estimador (mean o
median) disminuye al crecer n, evidencia consistencia.\n")
} else cat("No hay suficientes datos (>30) para estudiar consistencia por
submuestreo.\n")


# ---- Celda 7: Eficiencia y Sintaxis empleada -------------------------
------
library(dplyr)
library(boot)
set.seed(42)

x <- df_var$value
x_nm <- x[!is.na(x)]

if (length(x_nm) > 10) {
  # bootstrap var estimadas
  B <- 1000
  b_mean <- boot(x_nm, statistic = function(d,i) mean(d[i]), R = B)
  b_median <- boot(x_nm, statistic = function(d,i) median(d[i]), R = B)
  var_mean_boot <- var(b_mean$t)
  var_median_boot <- var(b_median$t)
  rel_eff <- var_median_boot / var_mean_boot

  cat("Eficiencia (empírica via bootstrap):\n")
  cat(" Var(mean) bootstrap:", var_mean_boot, "\n")
  cat(" Var(median) bootstrap:", var_median_boot, "\n")
  cat(" Rel. effic (var_median / var_mean):", rel_eff, "\n")

  cat("\nComentario (Eficiencia):\n")
  if (rel_eff > 1) cat(" La media es más eficiente (menor var) que la
mediana en esta muestra.\n") else cat(" La mediana es más eficiente en
esta muestra.\n")
} else cat("No hay suficientes datos para evaluar eficiencia.\n")
```