Examen: Construcción de una función de complejidad media o alta

Humberto Arrieta Rut: 26.873.351-5

Función: `tabla_frecuencias`

La función `tabla_frecuencias` en R está diseñada para generar una tabla de frecuencias y calcular medidas descriptivas a partir de una columna numérica de un dataframe, encapsulando los resultados en un objeto de la clase S4 `tabla_frecuencias`. Esta clase personalizada almacena información como la tabla de frecuencias, medidas de tendencia central (media, mediana, moda, cuartiles), medidas de dispersión (rango, desviación estándar, varianza, coeficiente de variación), valores faltantes, valores atípicos y, opcionalmente, un gráfico de histograma. La función permite personalizar el número de intervalos, el manejo de valores atípicos, la precisión de los decimales, la generación de gráficos y la exportación de resultados a formatos CSV o Excel. Además, incluye validaciones rigurosas para asegurar la integridad de los datos y proporciona métodos personalizados (`show` y `plot`) para visualizar los resultados de manera clara y estructurada. Es una herramienta integral para el análisis estadístico descriptivo de datos numéricos agrupados.

A continuación, desarrollaremos el desglose del código de la función `tabla_frecuenciasy` cada una de sus partes, para explicarla de la forma más clara y comprensible, la dividiremos en cuatro puntos:

1. **Definición de la clase S4: `tabla_frecuencias`**

```
setClass(
  "TablaFrecuencias",
  slots = list(
    tabla_frecuencias = "data.frame",
    media = "numeric",
    cuartil1 = "numeric",
    mediana = "numeric",
    cuartil3 = "numeric",
    moda_frecuencia = "numeric",
    moda_agrupada = "numeric",
```

```
    minimo = "numeric",
    maximo = "numeric",
    rango = "numeric",
    desviacion_estandar = "numeric",
    varianza = "numeric",
    coef_variacion = "numeric",
    valores_faltantes = "numeric",
    valores_atipicos = "numeric",
    grafico = "ANY",
    decimales = "numeric",
    column = "character"
  ),
  validity = function(object) {
    if (!is.data.frame(object@tabla_frecuencias)) {
      return("El slot 'tabla_frecuencias' debe ser un data.frame.")
    }
    if (!all(c("Intervalo", "Frecuencia_Absoluta", "Frecuencia_Acumulada",
               "Frecuencia_Relativa", "Frecuencia_Relativa_Acumulada",
               "Frecuencia_Porcentual", "Frecuencia_Porcentual_Acumulada") %in%
             names(object@tabla_frecuencias))) {
      return("El slot 'tabla_frecuencias' debe contener todas las columnas requeridas.")
    }
    if (object@decimales < 0 || object@decimales != floor(object@decimales)) {
      return("El slot 'decimales' debe ser un entero no negativo.")
    }
    if (length(object@column) != 1 || !is.character(object@column)) {
      return("El slot 'column' debe ser una cadena de caracteres de longitud 1.")
    }
    return(TRUE)
  }
)
```

- **Propósito**: Define una clase S4 llamada TablaFrecuencias para estructurar los resultados de la función en un objeto organizado.
- **Slots**:
  - **tabla_frecuencias:** Dataframe que contiene la tabla de frecuencias (intervalos, frecuencias absolutas, acumuladas, relativas, etc.).
  - **media, cuartil1, mediana, cuartil3, moda_frecuencia, moda_agrupada, minimo, maximo, rango, desviacion_estandar, varianza, coef_variacion:** Valores numéricos para medidas descriptivas.

- **valores_faltantes, valores_atipicos:** Cantidad de valores NA y valores atípicos detectados.
- **grafico:** Objeto para almacenar un gráfico (tipo ANY para permitir flexibilidad, como objetos ggplot).
- **decimales:** Número de decimales para redondeo.
- **column:** Nombre de la columna analizada (cadena de texto).

- **Validación validity)**: Define reglas para asegurar que el objeto sea válido:
  - tabla_frecuencias debe ser un data frame con columnas específicas.
  - decimales debe ser un entero no negativo.
  - column debe ser una cadena de longitud 1.
  - Si alguna regla falla, se devuelve un mensaje de error; si todo es correcto, retorna TRUE.

2. **Función principal: `tabla_frecuencias`**

```
# Función para generar una tabla de frecuencias y medidas descriptivas

tabla_frecuencias <- function(data, column, k = NULL, decimales = 2, handle_outliers = F
```

- **Propósito**: Genera una tabla de frecuencias y calcula medidas descriptivas (media, mediana, moda, etc.) para una columna numérica de un data frame, devolviendo un objeto S4 TablaFrecuencias.
- **Parámetros**:
  - **data:** Dataframe con los datos.
  - **column:** Nombre de la columna a analizar.
  - **k:** Número de intervalos para la tabla de frecuencias (opcional, usa la regla de Sturges si es NULL).
  - **decimales:** Número de decimales para redondear (por defecto, 2).
  - **handle_outliers:** Si TRUE, elimina valores atípicos antes de los cálculos.
  - **plot:** Si TRUE, genera un histograma usando ggplot2.
  - **export:** Ruta para exportar la tabla (.csv o .xlsx).
  - **silent:** Si FALSE, imprime resultados en consola.

  a. **Validaciones iniciales**

```r
if (!is.data.frame(data)) {
  stop("El argumento 'data' debe ser un dataframe.")
}
if (!is.character(column) || !column %in% names(data)) {
  stop("El argumento 'column' debe ser el nombre de una columna válida en el datafra
}
t <- data[[column]]
n_na <- sum(is.na(t))
n <- sum(!is.na(t))
if (n == 0) {
  stop("No hay datos no nulos en la columna especificada.")
}
if (!is.numeric(t)) {
  stop("La columna especificada debe ser numérica.")
}
```

- Verifica que data sea un data frame.
- Comprueba que column sea una cadena y exista en data.
- Extrae la columna especificada en t.
- Cuenta valores NA (n_na) y no nulos (n).
- Falla si no hay datos no nulos o si la columna no es numérica.

b. **Manejo de valores atípicos**

```r
t_clean <- t
outliers <- NULL
if (handle_outliers) {
  t_no_na <- t[!is.na(t)]
  if (length(t_no_na) == 0) {
    stop("No hay datos no nulos en la columna especificada para manejar valores atípi
  }
  q1 <- unname(as.numeric(quantile(t_no_na, 0.25, na.rm = TRUE)))
  q3 <- unname(as.numeric(quantile(t_no_na, 0.75, na.rm = TRUE)))
  iqr <- q3 - q1
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr
  outliers <- t_no_na[t_no_na < lower_bound | t_no_na > upper_bound]
  t_clean <- t[t <= upper_bound & t >= lower_bound & !is.na(t)]
  if (length(t_clean) == 0) {
    stop("Después de eliminar valores atípicos, no quedan datos válidos.")
```

```
    }
    n <- sum(!is.na(t_clean))
    if (!silent) {
      cat("Valores atípicos detectados y excluidos:", length(outliers), "\n")
    }
  }
```

- Si handle_outliers = TRUE, elimina valores atípicos usando el método del rango intercuartílico (IQR).
- Filtra valores no nulos (t_no_na).
- Calcula el primer cuartil (q1), tercer cuartil (q3) y el IQR.
- Define límites: lower_bound = q1 - 1.5 * IQR, upper_bound = q3 + 1.5 * IQR.
- Identifica valores atípicos (outliers) fuera de estos límites.
- Crea t_clean con los valores dentro de los límites.
- Falla si no quedan datos válidos.
- Imprime el número de valores atípicos excluidos si silent = FALSE.

c. **Validación de datos**

```
  t_clean <- t_clean[!is.na(t_clean) & is.finite(t_clean)]
  if (length(t_clean) == 0) {
    stop("No hay datos válidos después de filtrar valores no finitos.")
  }
  if (length(t_clean) < 2) {
    stop("No hay suficientes datos válidos para crear una tabla de frecuencias.")
  }
```

- Filtra valores no finitos (por ejemplo, Inf o -Inf) de t_clean.
- Falla si no hay datos válidos o si hay menos de 2 valores (insuficientes para crear intervalos).

d. **Cálculo del número de intervalos**

```r
if (is.null(k)) {
  k <- ceiling(1 + 3.322 * log10(n))
} else {
  if (!is.numeric(k) || k <= 0 || k != floor(k)) {
    stop("El argumento 'k' debe ser un número entero positivo.")
  }
}
```

- Si k es NULL, calcula el número de intervalos usando la regla de Sturges: $k = 1 + 3.322 * log10(n)$.
- Si se proporciona k, verifica que sea un entero positivo; de lo contrario, falla.

e. **Creación de intervalos**

```r
rango <- max(t_clean, na.rm = TRUE) - min(t_clean, na.rm = TRUE)

amplitud <- rango / k

breaks <- seq(min(t_clean, na.rm = TRUE), max(t_clean, na.rm = TRUE), by = amplitud)

if (max(t_clean, na.rm = TRUE) > max(breaks)) {
  breaks <- c(breaks, max(breaks) + amplitud)
}
if (length(breaks) <= 1) {
  stop("No se pueden crear intervalos: el rango de los datos es demasiado pequeño.")
}
intervalos <- cut(t_clean, breaks = breaks, right = TRUE, include.lowest = TRUE)
```

- Calcula el rango de los datos (max - min).
- Determina a amplitud de los intervalos: amplitud = rango / k.
- Crea puntos de corte (breaks) desde el mínimo al máximo con pasos de amplitud.
- Añade un punto de corte adicional si el máximo no está incluido.
- Falla si hay un solo punto de corte (rango cero).
- Divide los datos en intervalos usando cut, incluyendo el límite inferior y con intervalos cerrados a la derecha.

f. **Cálculo de frecuencias**

```r
absoluta <- table(intervalos)

acumulada <- cumsum(absoluta)

relativa <- prop.table(absoluta)

r_acumulada <- cumsum(relativa)

porcentual <- relativa * 100

p_acumulada <- cumsum(porcentual)
```

- absoluta: Frecuencia absoluta de cada intervalo (número de valores en cada uno).
- acumulada: Frecuencia acumulada (suma acumulada de frecuencias absolutas).
- relativa: Frecuencia relativa (frecuencia absoluta dividida por el total de datos).
- r_acumulada: Frecuencia relativa acumulada.
- porcentual: Frecuencia porcentual (relativa * 100).
- p_acumulada: Frecuencia porcentual acumulada.

g. **Creación de la tabla de frecuencias**

```r
tabla_frecuencias <- data.frame(
  Intervalo = names(absoluta),
  Frecuencia_Absoluta = as.vector(absoluta),
  Frecuencia_Acumulada = as.vector(acumulada),
  Frecuencia_Relativa = sprintf(paste0("%.", decimales, "f"), as.vector(relativa)),
  Frecuencia_Relativa_Acumulada = sprintf(paste0("%.", decimales, "f"), as.vector(r_a
  Frecuencia_Porcentual = sprintf(paste0("%.", decimales, "f"), as.vector(porcentual)
  Frecuencia_Porcentual_Acumulada = sprintf(paste0("%.", decimales, "f"), as.vector(p
)
```

- Crea un data frame con las columnas: Intervalo, Frecuencia_Absoluta, Frecuencia_Acumulada, Frecuencia_Relativa, Frecuencia_Relativa_Acumulada, Frecuencia_Porcentual, y Frecuencia_Porcentual_Acumulada.
- Usa sprintf para formatear las frecuencias relativas y porcentuales con el número de decimales especificado (por ejemplo, 0.2000 para decimales = 4).

h. **Cálculo de medidas descriptivas**

```r
m <- mean(t_clean, na.rm = TRUE)

q1 <- unname(as.numeric(quantile(t_clean, 0.25, na.rm = TRUE)))

me <- median(t_clean, na.rm = TRUE)

q3 <- unname(as.numeric(quantile(t_clean, 0.75, na.rm = TRUE)))
```

- Calcula la media (m), el primer cuartil (q1), la mediana (me) y el tercer cuartil (q3) de los datos limpios (t_clean).
- Usa unname para eliminar nombres de los cuartiles y asegurar valores numéricos puros.

i. **Moda no agrupada**

```r
obtener_moda <- function(x) {
  ux <- unique(x[!is.na(x)])
  tab <- tabulate(match(x, ux))
  max_freq <- max(tab)
  if (max_freq == 0) return(NA_real_)
  modas <- ux[tab == max_freq]
  if (length(modas) == length(ux)) return(NA_real_)
  return(modas)
}
moda <- obtener_moda(t_clean)
```

- Define una función interna obtener_moda para calcular la moda no agrupada.
- Extrae valores únicos (ux) sin NA.
- Calcula la frecuencia de cada valor (tab).
- Identifica el valor (o valores) con la frecuencia máxima.
- Devuelve NA_real_ si no hay datos o si todos los valores son únicos (sin moda definida).
- Almacena las modas en moda.

j. **Moda agrupada**

Determinar la moda para datos agrupados en un conjunto de números continuos es preferible a simplemente identificar el número con mayor frecuencia por las siguientes razones:

I. **Naturaleza de los datos continuos**: Los datos continuos, como alturas, pesos o tiempos, suelen tener valores únicos o muy dispersos, lo que hace que identificar un solo número con mayor frecuencia sea poco representativo o incluso imposible, ya que las frecuencias individuales tienden a ser bajas.

II. **Agrupación en intervalos**: Al agrupar los datos en intervalos de clase, se captura mejor la distribución de los valores continuos. La moda se calcula en el intervalo con mayor frecuencia (clase modal), lo que refleja la región donde los datos se concentran más, en lugar de un valor específico que puede no ser representativo.

III. **Estimación más precisa**: Para datos agrupados, la moda se estima usando una fórmula Moda $= L + \left(\frac{f_m - f_a}{f_m - f_a + f_m - f_p}\right) \cdot h$ donde L es el límite inferior de la clase modal, f_m es la frecuencia de la clase modal, f_{m-1} y f_{m+1} son las frecuencias de las clases adyacentes, y h es el ancho del intervalo). Esto proporciona una aproximación más precisa del valor central de la distribución en comparación con solo elegir un número.

IV. **Evita distorsiones por datos atípicos**: En datos continuos no agrupados, un valor con mayor frecuencia puede ser un outlier o un dato aislado. La agrupación reduce este riesgo al considerar rangos de valores, ofreciendo una visión más robusta de la tendencia central.

V. **Refleja mejor la distribución**: La moda en datos agrupados se basa en la densidad de los datos en un rango, lo que es más adecuado para entender la forma de la distribución (por ejemplo, si es unimodal o multimodal) en conjuntos de datos continuos.

En resumen, para datos continuos, la moda en datos agrupados es más representativa, precisa y adecuada para capturar patrones en la distribución, mientras que buscar el número con mayor frecuencia puede ser engañoso o poco informativo.

```r
max_freq <- max(tabla_frecuencias$Frecuencia_Absoluta)

idx_modales <- which(tabla_frecuencias$Frecuencia_Absoluta == max_freq)

modas_agrupadas <- numeric(length(idx_modales))

for (i in seq_along(idx_modales)) {
  idx_modal <- idx_modales[i]

  inter_modal <- tabla_frecuencias$Intervalo[idx_modal]

  inter_limpio <- gsub("\\[|\\]|\\(|\\)", "", inter_modal)

  valores <- as.numeric(unlist(strsplit(inter_limpio, ",")))

  limite_inf_modal <- valores[1]

  fmodal <- tabla_frecuencias$Frecuencia_Absoluta[idx_modal]

  fanterior <- ifelse(idx_modal > 1, tabla_frecuencias$Frecuencia_Absoluta[idx_modal
```

```
    fposterior <- ifelse(idx_modal < nrow(tabla_frecuencias),
                         tabla_frecuencias$Frecuencia_Absoluta[idx_modal + 1], 0)
    denominator <- (fmodal - fanterior) + (fmodal - fposterior)
    if (denominator == 0 || is.na(denominator)) {
      modas_agrupadas[i] <- NA
    } else {
      mo <- limite_inf_modal + ((fmodal - fanterior) / denominator) * amplitud
      modas_agrupadas[i] <- round(mo, decimales)
    }
  }
```

- Calcula la moda agrupada para los intervalos con la mayor frecuencia absoluta.
- Identifica los índices de los intervalos modales (idx_modales).
- Para cada intervalo modal:
- Extrae el límite inferior (limite_inf_modal) del intervalo.
- Obtiene la frecuencia del intervalo modal (fmodal), del intervalo anterior (fanterior) y posterior (fposterior).
- Aplica la fórmula de la moda agrupada:

$$\text{Moda} = L + \left(\frac{f_m - f_a}{f_m - f_a + f_m - f_p} \right) \cdot h$$

- Redondea el resultado a decimales.
- Si el denominador es cero o NA, asigna NA.

k. **Otras medidas descriptivas**

```
  rango_valores <- range(t_clean, na.rm = TRUE)

  rango_dif <- diff(rango_valores)

  desvest <- sd(t_clean, na.rm = TRUE)

  varianza <- var(t_clean, na.rm = TRUE)

  cv <- (desvest / m) * 100
```

- Calcula el rango (max - min), desviación estándar (sd), varianza (var) y coeficiente de variación (sd / media * 100) de t_clean.

l. **Generación de gráfico**

```
grafico <- NULL
if (plot) {
  if (!requireNamespace("ggplot2", quietly = TRUE)) {
    warning("El paquete 'ggplot2' no está instalado. Instálalo para generar gráficos
  } else {
    midpoints <- (breaks[-length(breaks)] + breaks[-1]) / 2
    grafico <- ggplot2::ggplot(data.frame(x = t_clean), ggplot2::aes(x = x)) +
      ggplot2::geom_histogram(breaks = breaks, fill = "lightblue", color = "black") +
      ggplot2::scale_x_continuous(
        breaks = midpoints,
        labels = tabla_frecuencias$Intervalo
      ) +
      ggplot2::labs(title = paste("Histograma de", column), x = column, y = "Frecuen
      ggplot2::theme_minimal() +
      ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
  }
}
```

- Si plot = TRUE, intenta generar un histograma con ggplot2.
- Verifica si ggplot2 está instalado; si no, emite una advertencia.
- Calcula os puntos medios de los intervalos para las etiquetas del eje x.
- Crea un histograma con los datos limpios, usando los puntos de corte (breaks) y personalizando título, ejes y tema.

m. **Exportación de la tabla**

```
if (!is.null(export)) {
  if (!is.character(export)) {
    warning("El argumento 'export' debe ser una cadena con la ruta del archivo.")
  } else {
    if (grepl("\\.csv$", export)) {
      write.csv(tabla_frecuencias, file = export, row.names = FALSE)
      if (!silent) {
        cat("Tabla exportada como CSV en:", export, "\n")
      }
    } else if (grepl("\\.xlsx$", export)) {
      if (!requireNamespace("writexl", quietly = TRUE)) {
        warning("El paquete 'writexl' no está instalado. Instálalo para exportar a Ex
      } else {
```

```
            writexl::write_xlsx(tabla_frecuencias, path = export)
            if (!silent) {
              cat("Tabla exportada como Excel en:", export, "\n")
            }
          }
        } else {
          warning("Formato de archivo no soportado. Usa '.csv' o '.xlsx'.")
        }
      }
    }
```

- Si export no es NULL, intenta exportar la tabla
- Verifica que export sea una cadena.
- Si termina en .csv, exporta como CSV usando write.csv.
- Si termina en .xlsx, exporta como Excel usando writexl (si está instalado).
- Si el formato no es soportado, emite una advertencia.
- Imprime un mensaje de confirmación si silent = FALSE.

n. **Creación del objeto S4**

```
    resultado <- new("TablaFrecuencias",
                     tabla_frecuencias = tabla_frecuencias,
                     media = m,
                     cuartil1 = q1,
                     mediana = me,
                     cuartil3 = q3,
                     moda_frecuencia = if (length(moda) == 0) NA_real_ else moda,
                     moda_agrupada = if (length(modas_agrupadas) == 0) NA_real_ else moda
                     minimo = min(t_clean, na.rm = TRUE),
                     maximo = max(t_clean, na.rm = TRUE),
                     rango = rango_dif,
                     desviacion_estandar = desvest,
                     varianza = varianza,
                     coef_variacion = cv,
                     valores_faltantes = n_na,
                     valores_atipicos = if (length(outliers) == 0) NA_real_ else outliers
                     grafico = grafico,
                     decimales = decimales,
                     column = column
    )
```

- Crea un objeto S4 de clase TablaFrecuencias, asignando los valores calculados a los slots correspondientes.
- Asegura que moda_frecuencia y moda_agrupada sean NA_real_ si no hay valores válidos.
- Almacena los valores atípicos o NA_real_ si no hay.

o. **Salida en consola y visualización**

```
    if (!silent) {
      show(resultado)
    }
    if (interactive() && !silent) {
      View(tabla_frecuencias)
    }
    return(invisible(resultado))
```

- Si silent = FALSE, imprime el objeto usando el método show.
- En entornos interactivos (como RStudio) y si silent = FALSE, muestra la tabla en una ventana de visualización.
- Devuelve el objeto S4 de forma invisible (sin imprimirlo automáticamente).

3. **Método show para `tabla_frecuencias`**

```
setMethod("show", "TablaFrecuencias", function(object) {
  cat("\nTabla de Frecuencias, Medidas de Tendencia Central y Dispersión para la variabl
  if (object@valores_faltantes > 0) {
    cat("Valores faltantes (NA):", object@valores_faltantes, "\n")
  }

  cat("\n")
  print(object@tabla_frecuencias, row.names = FALSE)

  cat("\nMedidas de tendencia central:\n")

  cat("\n")
  cat("Media:", round(object@media, object@decimales), "\n")

  cat("Cuartil 1:", round(object@cuartil1, object@decimales), "\n")

  cat("Mediana:", round(object@mediana, object@decimales), "\n")

  cat("Cuartil 3:", round(object@cuartil3, object@decimales), "\n")
```

```
  cat("Moda (frecuencia):",
      if (length(object@moda_frecuencia) > 1) paste(round(object@moda_frecuencia, object
      else if (is.na(object@moda_frecuencia)) "No definida"
      else round(object@moda_frecuencia, object@decimales), "\n")

  cat("Moda (datos agrupados):",
      if (length(object@moda_agrupada) > 1) paste(round(object@moda_agrupada, object@dec
      else if (all(is.na(object@moda_agrupada))) "No definida"
      else round(object@moda_agrupada, object@decimales), "\n")

  cat("\nMedidas de variabilidad:\n")

  cat("\n")
  cat("Mínimo:", round(object@minimo, object@decimales), "\n")

  cat("Máximo:", round(object@maximo, object@decimales), "\n")

  cat("Rango:", round(object@rango, object@decimales), "\n")

  cat("Desviación Estándar:", round(object@desviacion_estandar, object@decimales), "\n")

  cat("Varianza:", round(object@varianza, object@decimales), "\n")

  cat("Coeficiente de Variación (%):", round(object@coef_variacion, object@decimales), '
})
```

- Define cómo se imprime un objeto TablaFrecuencias en la consola.
- Muestra:
  - Un título con el nombre de la columna.
  - Número de valores faltantes (si los hay).
  - La tabla de frecuencias.
  - Medidas de tendencia central (media, cuartiles, modas).
  - Medidas de variabilidad (mínimo, máximo, rango, desviación estándar, varianza, coeficiente de variación).
  - Redondea los valores numéricos según el slot decimales.
  - Maneja casos especiales para la moda (múltiples valores, NA, o un solo valor).

4. **Método plot para `tabla_frecuencias`**

```
setMethod("plot", "TablaFrecuencias", function(x, y, ...) {
  if (is.null(x@grafico)) {
    cat("No se generó un gráfico. Usa plot = TRUE al crear el objeto.\n")
  } else {
    print(x@grafico)
  }
})
```

- Define el método plot para objetos TablaFrecuencias.
- Si no hay gráfico (grafico es NULL), imprime un mensaje indicando que se debe usar plot = TRUE.
- Si hay un gráfico (generado con ggplot2), lo imprime.

Para mostrar un poco de la evolución del código se deja a continuación la función `tabla_frecuencias`, sin el S4

Funcion `tabla_frecuencias` sin S4

```
# Función para generar una tabla de frecuencias y medidas descriptivas
tabla_frecuencias <- function(data, column, k = NULL, decimals = 2, handle_outliers = FALSE,

  # Validar que data sea un dataframe
  if (!is.data.frame(data)) {
    stop("El argumento 'data' debe ser un dataframe.")
  }

  # Validar que column sea una cadena y exista en el dataframe
  if (!is.character(column) || !column %in% names(data)) {
    stop("El argumento 'column' debe ser el nombre de una columna válida en el dataframe.")
  }

  # Extraer la columna especificada
  t <- data[[column]]

  # Contar valores NA
  n_na <- sum(is.na(t))
  n <- sum(!is.na(t))
  if (n == 0) {
    stop("No hay datos no nulos en la columna especificada.")
```

```r
}

# Validar que la columna sea numérica
if (!is.numeric(t)) {
  stop("La columna especificada debe ser numérica.")
}

# Manejo de valores atípicos (si handle_outliers = TRUE)
t_clean <- t
outliers <- NULL
if (handle_outliers) {
  q1 <- as.numeric(quantile(t, 0.25, na.rm = TRUE))
  q3 <- as.numeric(quantile(t, 0.75, na.rm = TRUE))
  iqr <- q3 - q1
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr
  outliers <- t[t > upper_bound]  # Solo valores superiores
  t_clean <- t[t <= upper_bound]
  if (length(t_clean) == 0) {
    stop("Después de eliminar valores atípicos, no quedan datos válidos.")
  }
  n <- sum(!is.na(t_clean))
  if (!silent) {
    cat("Valores atípicos detectados y excluidos:", length(outliers), "\n")
  }
}

# Filtrar valores no finitos
t_clean <- t_clean[!is.na(t_clean) & is.finite(t_clean)]
if (length(t_clean) == 0) {
  stop("No hay datos válidos después de filtrar valores no finitos.")
}

# Calcular el número de intervalos usando la regla de Sturges si k es NULL
if (is.null(k)) {
  k <- ceiling(1 + 3.322 * log10(n))
} else {
  # Validar que k sea un número entero positivo
  if (!is.numeric(k) || k <= 0 || k != floor(k)) {
    stop("El argumento 'k' debe ser un número entero positivo.")
  }
}
```

```r
# Calcular el rango de los datos
rango <- max(t_clean, na.rm = TRUE) - min(t_clean, na.rm = TRUE)

# Calcular el ancho de los intervalos
amplitud <- rango / k

# Crear los puntos de corte (breaks) para los intervalos
breaks <- seq(min(t_clean, na.rm = TRUE), max(t_clean, na.rm = TRUE), by = amplitud)

# Ajustar breaks para incluir el máximo si es necesario
if (max(t_clean, na.rm = TRUE) > max(breaks)) {
  breaks <- c(breaks, max(breaks) + amplitud)
}

# Verificar que haya suficientes puntos de corte
if (length(breaks) <= 1) {
  stop("No se pueden crear intervalos: el rango de los datos es demasiado pequeño.")
}

# Crear los intervalos
intervalos <- cut(t_clean, breaks = breaks, right = TRUE, include.lowest = TRUE)

# Calcular frecuencias absolutas
absoluta <- table(intervalos)

# Calcular frecuencias acumuladas
acumulada <- cumsum(absoluta)

# Calcular frecuencias relativas
relativa <- prop.table(absoluta)

# Calcular frecuencias relativas acumuladas
r_acumulada <- cumsum(relativa)

# Calcular frecuencias porcentuales
porcentual <- relativa * 100

# Calcular frecuencias porcentuales acumuladas
p_acumulada <- cumsum(porcentual)

# Crear la tabla de frecuencias con redondeo configurable
tabla_frecuencias <- data.frame(
```

```r
  Intervalo = names(absoluta),
  Frecuencia_Absoluta = as.vector(absoluta),
  Frecuencia_Acumulada = as.vector(acumulada),
  Frecuencia_Relativa = round(as.vector(relativa), decimals),
  Frecuencia_Relativa_Acumulada = round(as.vector(r_acumulada), decimals),
  Frecuencia_Porcentual = round(as.vector(porcentual), decimals),
  Frecuencia_Porcentual_Acumulada = round(as.vector(p_acumulada), decimals)
)

# Calcular medidas descriptivas
m <- mean(t_clean, na.rm = TRUE)
q1 <- as.numeric(quantile(t_clean, 0.25, na.rm = TRUE))
me <- median(t_clean, na.rm = TRUE)
q3 <- as.numeric(quantile(t_clean, 0.75, na.rm = TRUE))

# Moda para datos no agrupados
obtener_moda <- function(x) {
  ux <- unique(x[!is.na(x)])
  tab <- tabulate(match(x, ux))
  max_freq <- max(tab)
  if (max_freq == 0) return(NA)
  modas <- ux[tab == max_freq]
  if (length(modas) == length(ux)) return(NA)
  return(modas)
}
moda <- obtener_moda(t_clean)

# Moda agrupada con manejo mejorado
max_freq <- max(tabla_frecuencias$Frecuencia_Absoluta)
idx_modales <- which(tabla_frecuencias$Frecuencia_Absoluta == max_freq)
modas_agrupadas <- numeric(length(idx_modales))
for (i in seq_along(idx_modales)) {
  idx_modal <- idx_modales[i]
  inter_modal <- tabla_frecuencias$Intervalo[idx_modal]
  inter_limpio <- gsub("\\[|\\]|\\(|\\)", "", inter_modal)
  valores <- as.numeric(unlist(strsplit(inter_limpio, ",")))
  limite_inf_modal <- valores[1]
  fmodal <- tabla_frecuencias$Frecuencia_Absoluta[idx_modal]
  fanterior <- ifelse(idx_modal > 1, tabla_frecuencias$Frecuencia_Absoluta[idx_modal - 1],
  fposterior <- ifelse(idx_modal < nrow(tabla_frecuencias),
                       tabla_frecuencias$Frecuencia_Absoluta[idx_modal + 1], 0)
  denominator <- (fmodal - fanterior) + (fmodal - fposterior)
```

```r
    if (denominator == 0 || is.na(denominator)) {
      modas_agrupadas[i] <- NA
    } else {
      mo <- limite_inf_modal + ((fmodal - fanterior) / denominator) * amplitud
      modas_agrupadas[i] <- round(mo, decimals)
    }
  }

  # Rango de valores
  rango_valores <- range(t_clean, na.rm = TRUE)
  rango_dif <- diff(rango_valores)

  # Desviación estándar
  desvest <- sd(t_clean, na.rm = TRUE)

  # Varianza
  varianza <- var(t_clean, na.rm = TRUE)

  # Coeficiente de variación
  cv <- (desvest / m) * 100

  # Imprimir tabla y medidas descriptivas solo si silent = FALSE
  if (!silent) {
    cat("\nTabla de Frecuencias, Medidas de Tendencia Central y Dispersión para la variable"
    if (n_na > 0) {
      cat("Valores faltantes (NA):", n_na, "\n")
    }
    cat("\n")
    print(tabla_frecuencias, row.names = FALSE)
    cat("\nMedidas de tendencia central:\n")
    cat("\n")
    cat("Media:", round(m, decimals), "\n")
    cat("Cuartil 1:", round(q1, decimals), "\n")
    cat("Mediana:", round(me, decimals), "\n")
    cat("Cuartil 3:", round(q3, decimals), "\n")
    cat("Moda (frecuencia):", if (length(moda) > 1) paste(round(moda, decimals), collapse = "
    cat("Moda (datos agrupados):", if (length(modas_agrupadas) > 1) paste(modas_agrupadas, co
    cat("\nMedidas de variabilidad:\n")
    cat("\n")
    cat("Mínimo:", round(min(t_clean, na.rm = TRUE), decimals), "\n")
    cat("Máximo:", round(max(t_clean, na.rm = TRUE), decimals), "\n")
    cat("Rango:", round(rango_dif, decimals), "\n")
```

```
  cat("Desviación Estándar:", round(desvest, decimals), "\n")
  cat("Varianza:", round(varianza, decimals), "\n")
  cat("Coeficiente de Variación (%):", round(cv, decimals), "\n")
}

# Visualización en entorno interactivo
if (interactive() && !silent) {
  View(tabla_frecuencias)
}

# Generar gráfico con ggplot2 si plot = TRUE
if (plot) {
  if (!requireNamespace("ggplot2", quietly = TRUE)) {
    warning("El paquete 'ggplot2' no está instalado. Instálalo para generar gráficos.")
  } else {
    # Crear el histograma con etiquetas de intervalos centradas
    midpoints <- (breaks[-length(breaks)] + breaks[-1]) / 2
    p <- ggplot2::ggplot(data.frame(x = t_clean), ggplot2::aes(x = x)) +
      ggplot2::geom_histogram(breaks = breaks, fill = "lightblue", color = "black") +
      ggplot2::scale_x_continuous(
        breaks = midpoints,
        labels = tabla_frecuencias$Intervalo
      ) +
      ggplot2::labs(title = paste("Histograma de", column), x = column, y = "Frecuencia") +
      ggplot2::theme_minimal() +
      ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
    print(p)
  }
}

# Exportar tabla si se especifica un archivo
cat("\n")
if (!is.null(export)) {
  if (!is.character(export)) {
    warning("El argumento 'export' debe ser una cadena con la ruta del archivo.")
  } else {
    if (grepl("\\.csv$", export)) {
      write.csv(tabla_frecuencias, file = export, row.names = FALSE)
      if (!silent) {
        cat("Tabla exportada como CSV en:", export, "\n")
      }
    } else if (grepl("\\.xlsx$", export)) {
```

```
        if (!requireNamespace("writexl", quietly = TRUE)) {
          warning("El paquete 'writexl' no está instalado. Instálalo para exportar a Excel.")
        } else {
          writexl::write_xlsx(tabla_frecuencias, path = export)
          if (!silent) {
            cat("Tabla exportada como Excel en:", export, "\n")
          }
        }
      } else {
        warning("Formato de archivo no soportado. Usa '.csv' o '.xlsx'.")
      }
    }
  }

  # Devolver resultados
  return(invisible(list(
    tabla_frecuencias = tabla_frecuencias,
    media = m,
    cuartil1 = q1,
    mediana = me,
    cuartil3 = q3,
    moda_frecuencia = moda,
    moda_agrupada = modas_agrupadas,
    minimo = min(t_clean, na.rm = TRUE),
    maximo = max(t_clean, na.rm = TRUE),
    rango = rango_dif,
    desviacion_estandar = desvest,
    varianza = varianza,
    coef_variacion = cv,
    valores_faltantes = n_na,
    valores_atipicos = outliers
  )))
}
```

Funcion `tabla_frecuencias` con S4

```
# Definir la clase S4
setClass(
  "TablaFrecuencias",
  slots = list(
```

```
    tabla_frecuencias = "data.frame",
    media = "numeric",
    cuartil1 = "numeric",
    mediana = "numeric",
    cuartil3 = "numeric",
    moda_frecuencia = "numeric",
    moda_agrupada = "numeric",
    minimo = "numeric",
    maximo = "numeric",
    rango = "numeric",
    desviacion_estandar = "numeric",
    varianza = "numeric",
    coef_variacion = "numeric",
    valores_faltantes = "numeric",
    valores_atipicos = "numeric",
    grafico = "ANY",
    decimales = "numeric",
    column = "character"
  ),
  validity = function(object) {
    if (!is.data.frame(object@tabla_frecuencias)) {
      return("El slot 'tabla_frecuencias' debe ser un data.frame.")
    }
    if (!all(c("Intervalo", "Frecuencia_Absoluta", "Frecuencia_Acumulada",
               "Frecuencia_Relativa", "Frecuencia_Relativa_Acumulada",
               "Frecuencia_Porcentual", "Frecuencia_Porcentual_Acumulada") %in%
             names(object@tabla_frecuencias))) {
      return("El slot 'tabla_frecuencias' debe contener todas las columnas requeridas.")
    }
    if (object@decimales < 0 || object@decimales != floor(object@decimales)) {
      return("El slot 'decimales' debe ser un entero no negativo.")
    }
    if (length(object@column) != 1 || !is.character(object@column)) {
      return("El slot 'column' debe ser una cadena de caracteres de longitud 1.")
    }
    return(TRUE)
  }
)

# Función para generar una tabla de frecuencias y medidas descriptivas (Clase S4)
tabla_frecuencias <- function(data, column, k = NULL, decimales = 2, handle_outliers = FALSE
```

```r
# Validar que data sea un dataframe
if (!is.data.frame(data)) {
  stop("El argumento 'data' debe ser un dataframe.")
}

# Validar que column sea una cadena y exista en el dataframe
if (!is.character(column) || !column %in% names(data)) {
  stop("El argumento 'column' debe ser el nombre de una columna válida en el dataframe.")
}

# Extraer la columna especificada
t <- data[[column]]

# Contar valores NA
n_na <- sum(is.na(t))
n <- sum(!is.na(t))
if (n == 0) {
  stop("No hay datos no nulos en la columna especificada.")
}

# Validar que la columna sea numérica
if (!is.numeric(t)) {
  stop("La columna especificada debe ser numérica.")
}

# Manejo de valores atípicos (si handle_outliers = TRUE)
t_clean <- t
outliers <- NULL
if (handle_outliers) {
  t_no_na <- t[!is.na(t)]
  if (length(t_no_na) == 0) {
    stop("No hay datos no nulos en la columna especificada para manejar valores atípicos.")
  }
  q1 <- unname(as.numeric(quantile(t_no_na, 0.25, na.rm = TRUE)))
  q3 <- unname(as.numeric(quantile(t_no_na, 0.75, na.rm = TRUE)))
  iqr <- q3 - q1
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr
  outliers <- t_no_na[t_no_na < lower_bound | t_no_na > upper_bound]
  t_clean <- t[t <= upper_bound & t >= lower_bound & !is.na(t)]
  if (length(t_clean) == 0) {
    stop("Después de eliminar valores atípicos, no quedan datos válidos.")
```

```
  }
  n <- sum(!is.na(t_clean))
  if (!silent) {
    cat("Valores atípicos detectados y excluidos:", length(outliers), "\n")
  }
}

# Filtrar valores no finitos
t_clean <- t_clean[!is.na(t_clean) & is.finite(t_clean)]
if (length(t_clean) == 0) {
  stop("No hay datos válidos después de filtrar valores no finitos.")
}
if (length(t_clean) < 2) {
  stop("No hay suficientes datos válidos para crear una tabla de frecuencias.")
}

# Calcular el número intervalos usando la regla de Sturges si k es NULL
if (is.null(k)) {
  k <- ceiling(1 + 3.322 * log10(n))
} else {
  if (!is.numeric(k) || k <= 0 || k != floor(k)) {
    stop("El argumento 'k' debe ser un número entero positivo.")
  }
}

# Calcular el rango de los datos
rango <- max(t_clean, na.rm = TRUE) - min(t_clean, na.rm = TRUE)

# Calcular el ancho de los intervalos
amplitud <- rango / k

# Crear los puntos de corte (breaks) para los intervalos
breaks <- seq(min(t_clean, na.rm = TRUE), max(t_clean, na.rm = TRUE), by = amplitud)

# Ajustar breaks para incluir el máximo si es necesario
if (max(t_clean, na.rm = TRUE) > max(breaks)) {
  breaks <- c(breaks, max(breaks) + amplitud)
}

# Verificar que haya suficientes puntos de corte
if (length(breaks) <= 1) {
  stop("No se pueden crear intervalos: el rango de los datos es demasiado pequeño.")
```

```r
}

# Crear los intervalos
intervalos <- cut(t_clean, breaks = breaks, right = TRUE, include.lowest = TRUE)

# Calcular frecuencias absolutas
absoluta <- table(intervalos)

# Calcular frecuencias acumuladas
acumulada <- cumsum(absoluta)

# Calcular frecuencias relativas
relativa <- prop.table(absoluta)

# Calcular frecuencias relativas acumuladas
r_acumulada <- cumsum(relativa)

# Calcular frecuencias porcentuales
porcentual <- relativa * 100

# Calcular frecuencias porcentuales acumuladas
p_acumulada <- cumsum(porcentual)

# Crear la tabla de frecuencias con redondeo configurable
tabla_frecuencias <- data.frame(
  Intervalo = names(absoluta),
  Frecuencia_Absoluta = as.vector(absoluta),
  Frecuencia_Acumulada = as.vector(acumulada),
  Frecuencia_Relativa = sprintf(paste0("%.", decimales, "f"), as.vector(relativa)),
  Frecuencia_Relativa_Acumulada = sprintf(paste0("%.", decimales, "f"), as.vector(r_acumula
  Frecuencia_Porcentual = sprintf(paste0("%.", decimales, "f"), as.vector(porcentual)),
  Frecuencia_Porcentual_Acumulada = sprintf(paste0("%.", decimales, "f"), as.vector(p_acumu
)

# Calcular medidas descriptivas
m <- mean(t_clean, na.rm = TRUE)
q1 <- unname(as.numeric(quantile(t_clean, 0.25, na.rm = TRUE)))
me <- median(t_clean, na.rm = TRUE)
q3 <- unname(as.numeric(quantile(t_clean, 0.75, na.rm = TRUE)))

# Moda para datos no agrupados
obtener_moda <- function(x) {
```

```r
  ux <- unique(x[!is.na(x)])
  tab <- tabulate(match(x, ux))
  max_freq <- max(tab)
  if (max_freq == 0) return(NA_real_)
  modas <- ux[tab == max_freq]
  if (length(modas) == length(ux)) return(NA_real_)
  return(modas)
}
moda <- obtener_moda(t_clean)

# Moda agrupada
max_freq <- max(tabla_frecuencias$Frecuencia_Absoluta)
idx_modales <- which(tabla_frecuencias$Frecuencia_Absoluta == max_freq)
modas_agrupadas <- numeric(length(idx_modales))
for (i in seq_along(idx_modales)) {
  idx_modal <- idx_modales[i]
  inter_modal <- tabla_frecuencias$Intervalo[idx_modal]
  inter_limpio <- gsub("\\[|\\]|\\(|\\)", "", inter_modal)
  valores <- as.numeric(unlist(strsplit(inter_limpio, ",")))
  limite_inf_modal <- valores[1]
  fmodal <- tabla_frecuencias$Frecuencia_Absoluta[idx_modal]
  fanterior <- ifelse(idx_modal > 1, tabla_frecuencias$Frecuencia_Absoluta[idx_modal - 1],
  fposterior <- ifelse(idx_modal < nrow(tabla_frecuencias),
                       tabla_frecuencias$Frecuencia_Absoluta[idx_modal + 1], 0)
  denominator <- (fmodal - fanterior) + (fmodal - fposterior)
  if (denominator == 0 || is.na(denominator)) {
    modas_agrupadas[i] <- NA
  } else {
    mo <- limite_inf_modal + ((fmodal - fanterior) / denominator) * amplitud
    modas_agrupadas[i] <- round(mo, decimales)
  }
}

# Rango de valores
rango_valores <- range(t_clean, na.rm = TRUE)
rango_dif <- diff(rango_valores)

# Desviación estándar
desvest <- sd(t_clean, na.rm = TRUE)

# Varianza
varianza <- var(t_clean, na.rm = TRUE)
```

```r
# Coeficiente de variación
cv <- (desvest / m) * 100

# Generar gráfico con ggplot2 si plot = TRUE
grafico <- NULL
if (plot) {
  if (!requireNamespace("ggplot2", quietly = TRUE)) {
    warning("El paquete 'ggplot2' no está instalado. Instálalo para generar gráficos.")
  } else {
    midpoints <- (breaks[-length(breaks)] + breaks[-1]) / 2
    grafico <- ggplot2::ggplot(data.frame(x = t_clean), ggplot2::aes(x = x)) +
      ggplot2::geom_histogram(breaks = breaks, fill = "lightblue", color = "black") +
      ggplot2::scale_x_continuous(
        breaks = midpoints,
        labels = tabla_frecuencias$Intervalo
      ) +
      ggplot2::labs(title = paste("Histograma de", column), x = column, y = "Frecuencia") +
      ggplot2::theme_minimal() +
      ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
  }
}

# Exportar tabla si se especifica un archivo
if (!is.null(export)) {
  if (!is.character(export)) {
    warning("El argumento 'export' debe ser una cadena con la ruta del archivo.")
  } else {
    if (grepl("\\.csv$", export)) {
      write.csv(tabla_frecuencias, file = export, row.names = FALSE)
      if (!silent) {
        cat("Tabla exportada como CSV en:", export, "\n")
      }
    } else if (grepl("\\.xlsx$", export)) {
      if (!requireNamespace("writexl", quietly = TRUE)) {
        warning("El paquete 'writexl' no está instalado. Instálalo para exportar a Excel.")
      } else {
        writexl::write_xlsx(tabla_frecuencias, path = export)
        if (!silent) {
          cat("Tabla exportada como Excel en:", export, "\n")
        }
      }
    } else {
```

```
        warning("Formato de archivo no soportado. Usa '.csv' o '.xlsx'.")
      }
    }
  }

  # Crear objeto de clase S4
  resultado <- new("TablaFrecuencias",
                   tabla_frecuencias = tabla_frecuencias,
                   media = m,
                   cuartil1 = q1,
                   mediana = me,
                   cuartil3 = q3,
                   moda_frecuencia = if (length(moda) == 0) NA_real_ else moda,
                   moda_agrupada = if (length(modas_agrupadas) == 0) NA_real_ else modas_agru
                   minimo = min(t_clean, na.rm = TRUE),
                   maximo = max(t_clean, na.rm = TRUE),
                   rango = rango_dif,
                   desviacion_estandar = desvest,
                   varianza = varianza,
                   coef_variacion = cv,
                   valores_faltantes = n_na,
                   valores_atipicos = if (length(outliers) == 0) NA_real_ else outliers,
                   grafico = grafico,
                   decimales = decimales,
                   column = column
  )

  # Imprimir resultados si silent = FALSE
  if (!silent) {
    # Imprimir el objeto completo usando el método show
    show(resultado)
  }

  # Visualización en entorno interactivo
  if (interactive() && !silent) {
    View(tabla_frecuencias)
  }

  return(invisible(resultado))
}

# Método show para la clase TablaFrecuencias
```

```r
setMethod("show", "TablaFrecuencias", function(object) {
  cat("\nTabla de Frecuencias, Medidas de Tendencia Central y Dispersión para la variable", c
  if (object@valores_faltantes > 0) {
    cat("Valores faltantes (NA):", object@valores_faltantes, "\n")
  }
  cat("\n")
  print(object@tabla_frecuencias, row.names = FALSE)
  cat("\nMedidas de tendencia central:\n")
  cat("\n")
  cat("Media:", round(object@media, object@decimales), "\n")
  cat("Cuartil 1:", round(object@cuartil1, object@decimales), "\n")
  cat("Mediana:", round(object@mediana, object@decimales), "\n")
  cat("Cuartil 3:", round(object@cuartil3, object@decimales), "\n")
  cat("Moda (frecuencia):",
      if (length(object@moda_frecuencia) > 1) paste(round(object@moda_frecuencia, object@deci
      else if (is.na(object@moda_frecuencia)) "No definida"
      else round(object@moda_frecuencia, object@decimales), "\n")
  cat("Moda (datos agrupados):",
      if (length(object@moda_agrupada) > 1) paste(round(object@moda_agrupada, object@decimale
      else if (all(is.na(object@moda_agrupada))) "No definida"
      else round(object@moda_agrupada, object@decimales), "\n")
  cat("\nMedidas de variabilidad:\n")
  cat("\n")
  cat("Mínimo:", round(object@minimo, object@decimales), "\n")
  cat("Máximo:", round(object@maximo, object@decimales), "\n")
  cat("Rango:", round(object@rango, object@decimales), "\n")
  cat("Desviación Estándar:", round(object@desviacion_estandar, object@decimales), "\n")
  cat("Varianza:", round(object@varianza, object@decimales), "\n")
  cat("Coeficiente de Variación (%):", round(object@coef_variacion, object@decimales), "\n")
})

# Método plot para la clase TablaFrecuencias
setMethod("plot", "TablaFrecuencias", function(x, y, ...) {
  if (is.null(x@grafico)) {
    cat("No se generó un gráfico. Usa plot = TRUE al crear el objeto.\n")
  } else {
    print(x@grafico)
  }
})
```

A continuación, se presenta una breve explicación de lo que realiza cada una de las 25 pruebas del conjunto de pruebas unitarias para la función `tabla_frecuencias` utilizando el pa-

quete testthat. Cada prueba verifica un aspecto específico del comportamiento de la función, cubriendo validaciones de entrada, cálculos, manejo de valores atípicos, exportación, gráficos y métodos S4.

```
library(testthat)

# Contexto para las pruebas de la función tabla_frecuencias
context("Pruebas para la función tabla_frecuencias")

# Crear un conjunto de datos de prueba
set.seed(123)
df_test <- data.frame(
  valores = c(1, 2, 2, 3, 3, 3, 4, 4, 5, NA, 10)
)
```

Validaciones de entrada

Prueba 1: Falla si 'data' no es un dataframe Verifica que la función lanza un error si el argumento data no es un data frame (por ejemplo, un vector numérico), con el mensaje de error esperado.

```
# Prueba 1: Validar que la función falla con un argumento 'data' no válido
test_that("Falla si 'data' no es un dataframe", {
  expect_error(
    tabla_frecuencias(data = c(1, 2, 3), column = "valores", silent = TRUE),
    "El argumento 'data' debe ser un dataframe."
  )
})
```

```
Test passed
```

Prueba 2: Falla si 'column' no es una columna válida Comprueba que la función falla si el argumento column especifica un nombre de columna que no existe en el data frame.

```
# Prueba 2: Validar que la función falla si 'column' no es una cadena válida
test_that("Falla si 'column' no es una columna válida", {
  expect_error(
    tabla_frecuencias(data = df_test, column = "no_existe", silent = TRUE),
    "El argumento 'column' debe ser el nombre de una columna válida en el dataframe."
  )
})
```

```
Test passed
```

Prueba 3: Falla si la columna no es numérica Asegura que la función arroja un error si la columna especificada contiene datos no numéricos (por ejemplo, caracteres).

```
# Prueba 3: Validar que la función falla si la columna no es numérica
test_that("Falla si la columna no es numérica", {
  df_char <- data.frame(text = c("a", "b", "c"))
  expect_error(
    tabla_frecuencias(data = df_char, column = "text", silent = TRUE),
    "La columna especificada debe ser numérica."
  )
})
```

```
Test passed
```

Prueba 4: Falla si no hay datos no nulos Valida que la función falla si la columna contiene solo valores NA, indicando que no hay datos válidos para procesar.

```
# Prueba 4: Validar que la función falla si no hay datos no nulos
test_that("Falla si no hay datos no nulos", {
  df_na <- data.frame(valores = c(NA, NA, NA))
  expect_error(
    tabla_frecuencias(data = df_na, column = "valores", silent = TRUE),
    "No hay datos no nulos en la columna especificada."
  )
})
```

```
Test passed
```

Prueba 5: Falla si 'k' no es un entero positivo Comprueba que la función lanza un error si el argumento k (número de intervalos) es negativo o no es un entero (por ejemplo, -1 o 2.5).

```
# Prueba 5: Validar que la función falla si 'k' no es un entero positivo
test_that("Falla si 'k' no es un entero positivo", {
  expect_error(
    tabla_frecuencias(data = df_test, column = "valores", k = -1, silent = TRUE),
    "El argumento 'k' debe ser un número entero positivo."
  )
  expect_error(
```

```
    tabla_frecuencias(data = df_test, column = "valores", k = 2.5, silent = TRUE),
    "El argumento 'k' debe ser un número entero positivo."
  )
})
```

```
Test passed
```

Prueba 6: Falla con un solo valor válido Comprueba que la función falla si, tras filtrar valores NA, solo queda un valor válido, ya que no se puede crear una tabla de frecuencias.

```
# Prueba 6: Validar que la función falla con un solo valor válido
test_that("Falla con un solo valor válido", {
  df_single <- data.frame(valores = c(1, NA, NA))
  expect_error(
    tabla_frecuencias(data = df_single, column = "valores", silent = TRUE),
    "No hay suficientes datos válidos para crear una tabla de frecuencias."
  )
})
```

```
Test passed
```

Prueba 7: Falla con datos de rango cero Verifica que la función lanza un error si todos los valores son idénticos (rango cero), impidiendo la creación de intervalos.

```
# Prueba 7: Validar que la función falla con datos de rango cero
test_that("Falla con datos de rango cero", {
  df_same <- data.frame(valores = c(5, 5, 5))
  expect_error(
    tabla_frecuencias(data = df_same, column = "valores", silent = TRUE),
    "No se pueden crear intervalos: el rango de los datos es demasiado pequeño."
  )
})
```

```
Test passed
```

Manejo de valores atípicos

Prueba 8: Falla si no hay datos válidos tras eliminar valores atípicos Verifica que la función falla cuando, después de eliminar valores atípicos, el rango de los datos es cero (por ejemplo, todos los valores son idénticos), impidiendo la creación de intervalos.

```
# Prueba 8: Validar que la función falla si no hay datos válidos tras eliminar valores atípi
test_that("Falla si no hay datos válidos tras eliminar valores atípicos", {
  df_outlier <- data.frame(valores = c(1000, 1000))
  expect_error(
    tabla_frecuencias(data = df_outlier, column = "valores", handle_outliers = TRUE, silent =
    "No se pueden crear intervalos: el rango de los datos es demasiado pequeño."
  )
})
```

```
Test passed
```

Prueba 9: Maneja correctamente los valores atípicos Comprueba que, con handle_outliers = TRUE, la función detecta correctamente un valor atípico (por ejemplo, el valor 10 en df_test) y lo almacena en el slot valores_atipicos.

```
# Prueba 9: Validar manejo de valores atípicos
test_that("Maneja correctamente los valores atípicos", {
  resultado <- tabla_frecuencias(data = df_test, column = "valores", handle_outliers = TRUE,

  expect_true(!is.na(resultado@valores_atipicos))
  expect_equal(length(resultado@valores_atipicos), 1) # Debería detectar el valor 10 como ati
})
```

```
Test passed
```

Prueba 10: Maneja correctamente múltiples valores atípicos Comprueba que la función detecta múltiples valores atípicos (por ejemplo, -10 y 20) cuando handle_outliers = TRUE.

```
# Prueba 10: Validar manejo de múltiples valores atípicos
test_that("Maneja correctamente múltiples valores atípicos", {
  df_multi_outliers <- data.frame(valores = c(-10, 1, 2, 2, 3, 3, 3, 4, 4, 5, 20))
  resultado <- tabla_frecuencias(data = df_multi_outliers, column = "valores", handle_outlier

  expect_equal(sort(resultado@valores_atipicos), c(-10, 20)) # Debería detectar -10 y 20 como
})
```

```
Test passed
```

Cálculos estadísticos

Prueba 11: Calcula correctamente las medidas descriptivas Valida que las medidas descriptivas (media, cuartil1, mediana, cuartil3, desviacion_estandar) calculadas por la función coinciden con los valores esperados para los datos de entrada.

```
# Prueba 11: Validar cálculos de medidas descriptivas
test_that("Calcula correctamente las medidas descriptivas", {
  resultado <- tabla_frecuencias(data = df_test, column = "valores", silent = TRUE)

  expect_equal(round(resultado@media, 4), round(mean(df_test$valores, na.rm = TRUE), 4))
  expect_equal(round(resultado@cuartil1, 4), round(unname(quantile(df_test$valores, 0.25, na
  expect_equal(round(resultado@mediana, 4), round(median(df_test$valores, na.rm = TRUE), 4))
  expect_equal(round(resultado@cuartil3, 4), round(unname(quantile(df_test$valores, 0.75, na
  expect_equal(round(resultado@desviacion_estandar, 4), round(sd(df_test$valores, na.rm = TRU
})
```

```
Test passed
```

Prueba 12: Calcula correctamente la moda no agrupada Confirma que la función calcula correctamente la moda no agrupada, incluyendo casos con múltiples modas (1 y 2) y casos sin moda definida (NA).

```
# Prueba 12: Validar cálculos de moda no agrupada
test_that("Calcula correctamente la moda no agrupada", {
  df_moda <- data.frame(valores = c(1, 1, 2, 2, 3)) # Dos modas: 1 y 2
  resultado <- tabla_frecuencias(data = df_moda, column = "valores", silent = TRUE)

  expect_equal(sort(resultado@moda_frecuencia), c(1, 2))

  df_no_moda <- data.frame(valores = c(1, 2, 3, 4)) # Sin moda definida
  resultado_no_moda <- tabla_frecuencias(data = df_no_moda, column = "valores", silent = TRU
  expect_true(is.na(resultado_no_moda@moda_frecuencia))
})
```

```
Test passed
```

Prueba 13: Calcula correctamente la moda agrupada Asegura que el slot moda_agrupada contiene un valor numérico y que tiene al menos un elemento.

```r
# Prueba 13: Validar moda agrupada
test_that("Calcula correctamente la moda agrupada", {
  resultado <- tabla_frecuencias(data = df_test, column = "valores", silent = TRUE)

  expect_true(is.numeric(resultado@moda_agrupada))
  expect_true(length(resultado@moda_agrupada) >= 1)
})
```

```
Test passed
```

Prueba 14: Calcula correctamente otras medidas descriptivas Valida el cálculo de medidas adicionales (varianza, coef_variacion, minimo, maximo, rango) comparándolas con los valores esperados.

```r
# Prueba 14: Validar otras medidas descriptivas
test_that("Calcula correctamente otras medidas descriptivas", {
  resultado <- tabla_frecuencias(data = df_test, column = "valores", silent = TRUE)

  expect_equal(round(resultado@varianza, 4), round(var(df_test$valores, na.rm = TRUE), 4))
  expect_equal(round(resultado@coef_variacion, 4), round((sd(df_test$valores, na.rm = TRUE) /
  expect_equal(resultado@minimo, min(df_test$valores, na.rm = TRUE))
  expect_equal(resultado@maximo, max(df_test$valores, na.rm = TRUE))
  expect_equal(resultado@rango, max(df_test$valores, na.rm = TRUE) - min(df_test$valores, na
})
```

```
Test passed
```

Gráficos

Prueba 15: Genera un gráfico si plot = TRUE Confirma que, cuando plot = TRUE y el paquete ggplot2 está instalado, la función genera un objeto de clase ggplot en el slot grafico.

```r
# Prueba 15: Validar que el gráfico se genera si plot = TRUE
test_that("Genera un gráfico si plot = TRUE", {
  skip_if_not_installed("ggplot2")
  resultado <- tabla_frecuencias(data = df_test, column = "valores", plot = TRUE, silent = TR

  expect_true(!is.null(resultado@grafico))
  expect_s3_class(resultado@grafico, "ggplot")
})
```

```
Test passed
```

Prueba 16: Lanza advertencia si ggplot2 no está instalado Confirma que la función lanza una advertencia si plot = TRUE pero el paquete ggplot2 no está instalado.

```
# Prueba 16: Validar advertencia cuando ggplot2 no está instalado
test_that("Lanza advertencia si ggplot2 no está instalado", {
  if (requireNamespace("ggplot2", quietly = TRUE)) {
    skip("ggplot2 está instalado, prueba omitida")
  }
  expect_warning(
    tabla_frecuencias(data = df_test, column = "valores", plot = TRUE, silent = TRUE),
    "El paquete 'ggplot2' no está instalado. Instálalo para generar gráficos."
  )
})
```

```
-- Skip: Lanza advertencia si ggplot2 no está instalado ------------------------
Reason: ggplot2 está instalado, prueba omitida
```

Formato de decimales

Prueba 17: Maneja correctamente el parámetro decimales Verifica que el parámetro decimales se establece correctamente y que las columnas de frecuencias relativas y porcentuales tienen el formato de texto con el número exacto de decimales especificado (por ejemplo, 0.20 para decimales = 2).

```
# Prueba 17: Validar comportamiento con decimales personalizados
test_that("Maneja correctamente el parámetro decimales", {
  resultado <- tabla_frecuencias(data = df_test, column = "valores", decimales = 2, silent =

  expect_equal(resultado@decimales, 2)
  expect_true(all(sapply(resultado@tabla_frecuencias[, c("Frecuencia_Relativa", "Frecuencia_
                                                         "Frecuencia_Porcentual", "Frecuencia_
                         function(x) all(grepl("\\.[0-9]{2}$", as.character(x))))))
})
```

```
Test passed
```

Exportación

Prueba 18: Exporta correctamente a CSV Verifica que la función exporta la tabla de frecuencias a un archivo CSV y que este contiene todas las columnas esperadas.

```
# Prueba 18: Validar exportación a CSV
test_that("Exporta correctamente a CSV", {
  archivo <- tempfile(fileext = ".csv")
  resultado <- tabla_frecuencias(data = df_test, column = "valores", export = archivo, silent

  expect_true(file.exists(archivo))
  tabla_leida <- read.csv(archivo)
  expect_true(all(c("Intervalo", "Frecuencia_Absoluta", "Frecuencia_Acumulada",
                    "Frecuencia_Relativa", "Frecuencia_Relativa_Acumulada",
                    "Frecuencia_Porcentual", "Frecuencia_Porcentual_Acumulada") %in%
                    names(tabla_leida)))
})
```

```
Test passed
```

Prueba 19: Exporta correctamente a Excel Verifica que la función exporta la tabla de frecuencias a un archivo Excel (.xlsx) si el paquete writexl está instalado, y que el archivo contiene las columnas esperadas.

```
# Prueba 19: Validar exportación a Excel
test_that("Exporta correctamente a Excel", {
  skip_if_not_installed("writexl")
  archivo <- tempfile(fileext = ".xlsx")
  resultado <- tabla_frecuencias(data = df_test, column = "valores", export = archivo, silent

  expect_true(file.exists(archivo))
  tabla_leida <- readxl::read_excel(archivo)
  expect_true(all(c("Intervalo", "Frecuencia_Absoluta", "Frecuencia_Acumulada",
                    "Frecuencia_Relativa", "Frecuencia_Relativa_Acumulada",
                    "Frecuencia_Porcentual", "Frecuencia_Porcentual_Acumulada") %in%
                    names(tabla_leida)))
})
```

```
Test passed
```

Prueba 20: Lanza advertencia para formato de archivo no soportado Asegura que la función emite una advertencia si se intenta exportar a un formato no soportado (por ejemplo, .txt).

```r
# Prueba 20: Validar advertencia para formato de archivo no soportado
test_that("Lanza advertencia para formato de archivo no soportado", {
  archivo <- tempfile(fileext = ".txt")
  expect_warning(
    tabla_frecuencias(data = df_test, column = "valores", export = archivo, silent = TRUE),
    "Formato de archivo no soportado. Usa '.csv' o '.xlsx'."
  )
})
```

```
Test passed
```

Creation y estructura del objeto S4

Prueba 21: Crea correctamente el objeto TablaFrecuencias Confirma que la función devuelve un objeto S4 de clase TablaFrecuencias, que el slot tabla_frecuencias es un data frame con las columnas esperadas, y que los slots decimales y column tienen los valores correctos.

```r
# Prueba 21: Validar que la función crea correctamente el objeto S4
test_that("Crea correctamente el objeto TablaFrecuencias", {
  resultado <- tabla_frecuencias(data = df_test, column = "valores", silent = TRUE)
  expect_s4_class(resultado, "TablaFrecuencias")
  expect_true(is.data.frame(resultado@tabla_frecuencias))
  expect_true(all(c("Intervalo", "Frecuencia_Absoluta", "Frecuencia_Acumulada",
                    "Frecuencia_Relativa", "Frecuencia_Relativa_Acumulada",
                    "Frecuencia_Porcentual", "Frecuencia_Porcentual_Acumulada") %in%
                    names(resultado@tabla_frecuencias)))
  expect_equal(resultado@decimales, 2)
  expect_equal(resultado@column, "valores")
})
```

```
Test passed
```

Prueba 22: Valida correctamente el objeto S4 Comprueba que las reglas de validación del objeto S4 funcionan, lanzando errores si el slot tabla_frecuencias no tiene las columnas requeridas, si decimales es negativo, o si column no es una cadena válida.

```r
# Prueba 22: Validar validaciones del objeto S4
test_that("Valida correctamente el objeto S4", {
  resultado <- tabla_frecuencias(data = df_test, column = "valores", silent = TRUE)
```

```
  # Modificar tabla_frecuencias para que falle la validación
  invalid_object <- resultado
  invalid_object@tabla_frecuencias <- data.frame(x = 1)
  expect_error(
    validObject(invalid_object),
    "El slot 'tabla_frecuencias' debe contener todas las columnas requeridas."
  )

  # Modificar decimales para que falle la validación
  invalid_object <- resultado
  invalid_object@decimales <- -1
  expect_error(
    validObject(invalid_object),
    "El slot 'decimales' debe ser un entero no negativo."
  )

  # Modificar column para que falle la validación
  invalid_object <- resultado
  invalid_object@column <- c("valores", "otro")
  expect_error(
    validObject(invalid_object),
    "El slot 'column' debe ser una cadena de caracteres de longitud 1."
  )
})
```

```
Test passed
```

Salida en consola y métodos S4

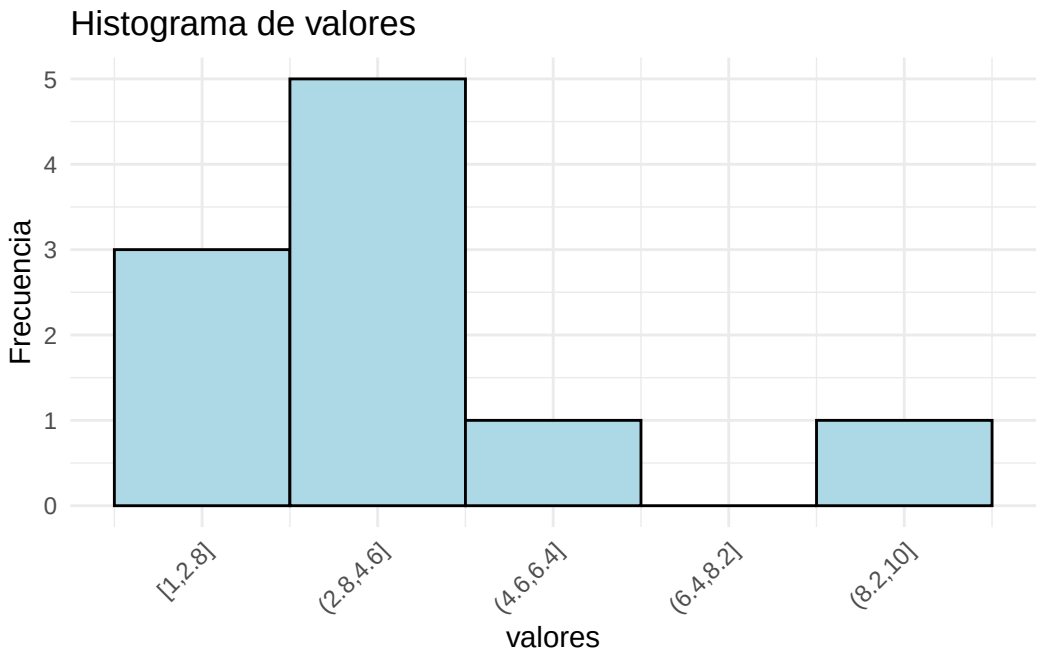
Prueba 23: El método show imprime correctamente Asegura que el método show del objeto S4 imprime en consola la tabla de frecuencias, los valores faltantes y la media, entre otros.

```
# Prueba 23: Validar el método show
test_that("El método show imprime correctamente", {
  resultado <- tabla_frecuencias(data = df_test, column = "valores", silent = TRUE)

  expect_output(show(resultado), "Tabla de Frecuencias")
  expect_output(show(resultado), "Valores faltantes \\(NA\\): 1")
  expect_output(show(resultado), "Media:")
})
```

```
Test passed
```

Prueba 24: El método plot funciona correctamente valida que el método plot imprime el gráfico si existe (cuando plot = TRUE) y muestra un mensaje si no se generó un gráfico (cuando plot = FALSE).

```r
# Prueba 24: Validar el método plot
test_that("El método plot funciona correctamente", {
  resultado <- tabla_frecuencias(data = df_test, column = "valores", plot = TRUE, silent = TR

  expect_silent(plot(resultado))

  resultado_no_plot <- tabla_frecuencias(data = df_test, column = "valores", plot = FALSE, si
  expect_output(plot(resultado_no_plot), "No se generó un gráfico")
})
```




```
Test passed
```

Prueba 25: Imprime mensajes en consola cuando silent = FALSE Valida que, con silent = FALSE, la función imprime mensajes en consola sobre valores atípicos detectados y la exportación a CSV.

```r
# Prueba 25: Validar salida en consola cuando silent = FALSE
test_that("Imprime mensajes en consola cuando silent = FALSE", {
  expect_output(
    tabla_frecuencias(data = df_test, column = "valores", handle_outliers = TRUE, silent = FA
    "Valores atípicos detectados y excluidos: 1"
  )
  archivo <- tempfile(fileext = ".csv")
  expect_output(
    tabla_frecuencias(data = df_test, column = "valores", export = archivo, silent = FALSE),
    "Tabla exportada como CSV en:"
  )
})
```

```
Test passed
```

Resumen

Las pruebas anteriores cubren exhaustivamente la funcionalidad de `tabla_frecuencias`, incluyendo:

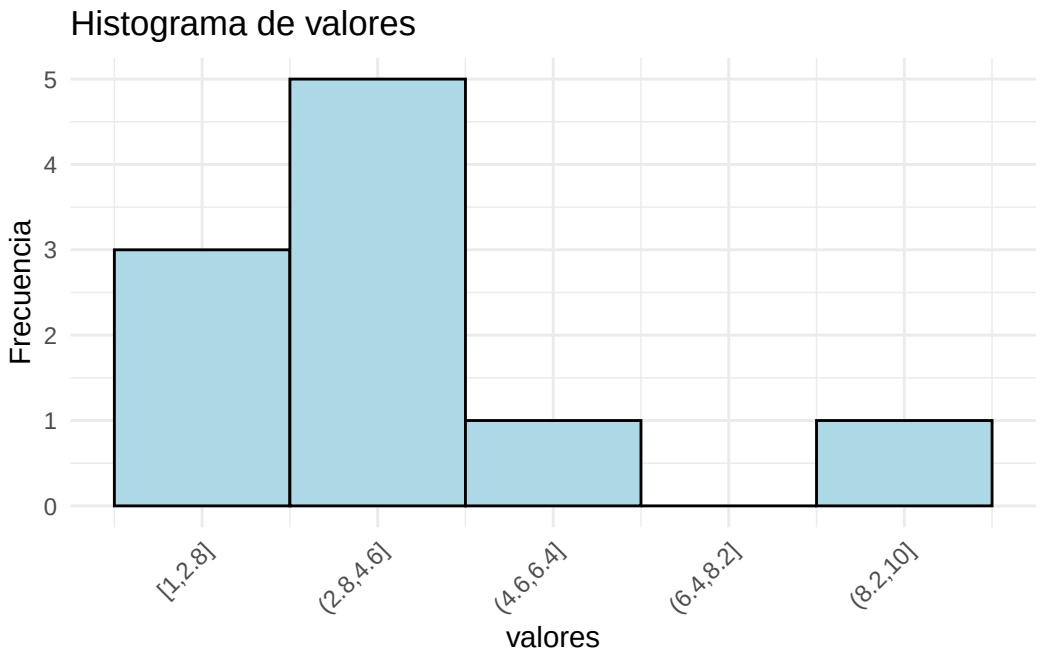
- **Validaciones de entrada** (Pruebas 1-7).
- **Manejo de valores atípicos** (Pruebas 8-10).
- **Cálculos estadísticos** (Pruebas 11-14).
- **Gráficos** (Pruebas 15-16).
- **Formato de decimales** (Prueba 17).
- **Exportación** (Pruebas 18-20).
- **Creación y estructura del objeto S4** (Pruebas 21-22).
- **Salida en consola y métodos S4** (Pruebas 23-25).

En el desarrollo de esta función estadísticas en R, como la función personalizada `tabla_frecuencia`, es fundamental garantizar no solo su correcto funcionamiento, sino también la solidez de las pruebas que la respaldan. Para ello, se ha utilizado el paquete {covr}, una herramienta especializada en la medición de la cobertura de código en entornos de desarrollo en R.

El objetivo principal de aplicar {covr} a esta función es evaluar en qué medida los tests diseñados anteriormente mediante `testthat` ejercitan las distintas ramas y condiciones del código.

```r
library(covr)
cov <- file_coverage(source_files = "tabla_frecuencias.R",
                     test_files = "test-tabla_frecuencias.R")
```

```
Warning in getPackageName(where): Created a package name, '2025-07-04
14:43:21.771947', when none found

Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
-- Skip: Lanza advertencia si ggplot2 no está instalado ------------------------
Reason: ggplot2 está instalado, prueba omitida

Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
```




```
Test passed
Test passed
```

```
print(cov)
```

```
Coverage: 94.48%

tabla_frecuencias.R: 94.48%
```

El porcentaje de 95,03% de cobertura, que nos arroja la librería `covr`, indica que:

1. **Está bien testeada**
   - La gran mayoría de sus líneas han sido ejecutadas por pruebas automatizadas.
   - Esto sugiere que los casos de uso principales y muchos casos borde están contemplados.
2. **Es confiable desde el punto de vista del testing**
   - Es poco probable que contenga errores en las partes cubiertas.
   - Si ocurre un bug, probablemente estará en el pequeño porcentaje no cubierto (4,97%; el porcentaje de error real es menor, leer la nota al final).

3. **Tiene una estructura clara y predecible**

   - Las funciones con alta cobertura suelen tener una lógica bien definida, sin demasiadas ramas complejas o condiciones difíciles de alcanzar.

4. **Es mantenible**

   - Una función bien cubierta es más fácil de modificar con confianza, ya que los tests actuarán como red de seguridad ante posibles regresiones.

5. **Puede estar lista para producción**

   - Aunque la cobertura no garantiza ausencia de errores, sí es un fuerte indicador de preparación para entornos reales, especialmente si la cobertura incluye pruebas de comportamiento y errores esperados.

     Nota: El porcentaje de cobertura reportado podría ser ligeramente inferior al real, ya que una de las pruebas incluye una condición para instalar el paquete `ggplot2`. Dado que este paquete ya se encontraba instalado en el entorno de ejecución, dicha rama condicional no fue ejecutada y, por tanto, no fue contabilizada como cubierta por `covr`.

Ahora pondremos a prueba la función `tabla_frecuencias` haciendo uso de datos reales, para ello cargaremos primero el siguiente dataset

```
library(readr)
```

```
Adjuntando el paquete: 'readr'

The following objects are masked from 'package:testthat':

    edition_get, local_edition
```

```
datos <- read.csv("datos.csv")
View(datos)
```

El cual esta formado por las siguientes variables:

N	Variable	Tipo
1	ID	Cualitativa (Alfanumerica)
2	Sexo	Cualitativa (caracter)
3	Edad	Cuantitativa (Continua)
4	Fuma	Cualitativa (caracter)

5	Estatura	Cualitativa (caracter)
6	Colegio	Cualitativa (caracter)
7	Estrato	Cuantitativa (discreta)
8	Financiacion	Cualitativa (caracter)
9	Acumulado	Cuantitativa (Continua)
10	Gastos	Cuantitativa (Continua)
11	Ingreso	Cuantitativa (Continua)
12	Clases	Cualitativa (caracter)
13	Pandemia	Cualitativa (caracter)
14	Clases_virtuales	Cualitativa (caracter)
15	Estadistica	Cualitativa (caracter)
16	inseguridad	Cualitativa (caracter)
17	vida_cotidiana	Cualitativa (caracter)
18	Puntaje	Cuantitativa (discreta)

A continuación mostraremos ejemplos del uso de la función `tabla_frecuencias` :

Ejemplo 1

```
#variable usada: Puntaje

#uso simple (usa numero de intervalos utilizando la regla de Sturges, y redondea a dos decima

tabla_frecuencias(data = datos, column = "Puntaje")
```

```
Tabla de Frecuencias, Medidas de Tendencia Central y Dispersión para la variable Puntaje

   Intervalo Frecuencia_Absoluta Frecuencia_Acumulada Frecuencia_Relativa
     [0,8.1]                   5                    5                0.01
  (8.1,16.2]                  45                   50                0.11
 (16.2,24.3]                 118                  168                0.29
 (24.3,32.4]                  98                  266                0.24
 (32.4,40.5]                  51                  317                0.13
 (40.5,48.6]                   6                  323                0.01
 (48.6,56.7]                  22                  345                0.06
 (56.7,64.8]                   5                  350                0.01
 (64.8,72.9]                  30                  380                0.07
   (72.9,81]                  20                  400                0.05
 Frecuencia_Relativa_Acumulada Frecuencia_Porcentual
                          0.01                  1.25
```

```
                          0.12                11.25
                          0.42                29.50
                          0.66                24.50
                          0.79                12.75
                          0.81                 1.50
                          0.86                 5.50
                          0.88                 1.25
                          0.95                 7.50
                          1.00                 5.00
 Frecuencia_Porcentual_Acumulada
                           1.25
                          12.50
                          42.00
                          66.50
                          79.25
                          80.75
                          86.25
                          87.50
                          95.00
                         100.00

Medidas de tendencia central:

Media: 31.78
Cuartil 1: 19
Mediana: 26.5
Cuartil 3: 35
Moda (frecuencia): 28
Moda (datos agrupados): 22.56

Medidas de variabilidad:

Mínimo: 0
Máximo: 81
Rango: 81
Desviación Estándar: 18.85
Varianza: 355.33
Coeficiente de Variación (%): 59.31
```

```
#Nos arroja la tabla de frecuencia junto a sus medidas de tendencia central y de variabilidad
```

Ejemplo 2:

```r
#variable usada: Edad

#Para este ejemplo tomamos cada uno de los elementos modificables

#tabla_frecuencias <- function(data, column, k = NULL, decimales = 2, handle_outliers = FALSE
tabla_frecuencias(data = datos, column = "Edad" , k = 5, decimales =1, handle_outliers = TRUE
```

```
Valores atípicos detectados y excluidos: 5
Tabla exportada como Excel en: tabla de frecuencia.xlsx

Tabla de Frecuencias, Medidas de Tendencia Central y Dispersión para la variable Edad

   Intervalo Frecuencia_Absoluta Frecuencia_Acumulada Frecuencia_Relativa
   [14,15.6]                   1                    1                 0.0
 (15.6,17.2]                  74                   75                 0.2
 (17.2,18.7]                 121                  196                 0.3
 (18.7,20.3]                  98                  294                 0.2
 (20.3,21.9]                 101                  395                 0.3
 Frecuencia_Relativa_Acumulada Frecuencia_Porcentual
                           0.0                   0.3
                           0.2                  18.7
                           0.5                  30.6
                           0.7                  24.8
                           1.0                  25.6
 Frecuencia_Porcentual_Acumulada
                             0.3
                            19.0
                            49.6
                            74.4
                           100.0

Medidas de tendencia central:

Media: 18.9
Cuartil 1: 17.5
Mediana: 18.8
Cuartil 3: 20.4
Moda (frecuencia): 18.7, 19.9, 17.9, 20.6
Moda (datos agrupados): 18.3

Medidas de variabilidad:
```

```
Mínimo: 14
Máximo: 21.9
Rango: 7.9
Desviación Estándar: 1.7
Varianza: 2.9
Coeficiente de Variación (%): 9
```

```
#Recordar que la tabla de frecuencia tambien la podemos exportar como .csv
```

Ejemplo 3:

```
#variable usada: Edad

#Para este ejemplo convertiremos a la función en un objeto, para aprovechar las opciones y ve

resultado <- tabla_frecuencias(data = datos, column = "Edad" , k = 5, decimales =1, handle_ou
```

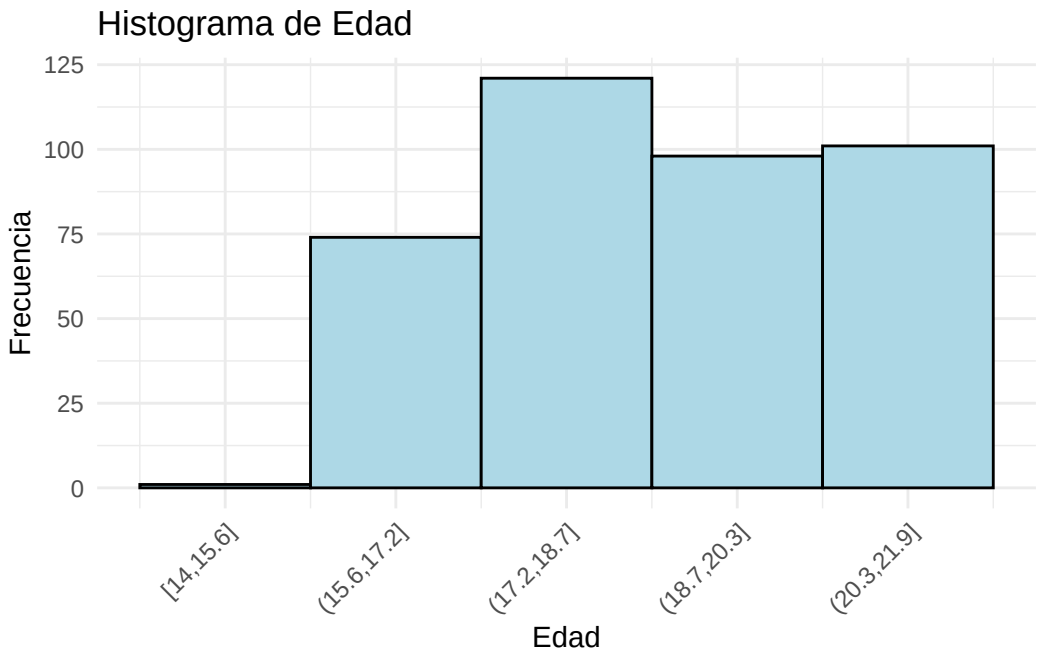
```
Valores atípicos detectados y excluidos: 5
Tabla exportada como CSV en: tabla de frecuencia.csv

Tabla de Frecuencias, Medidas de Tendencia Central y Dispersión para la variable Edad

   Intervalo Frecuencia_Absoluta Frecuencia_Acumulada Frecuencia_Relativa
   [14,15.6]                   1                     1                 0.0
 (15.6,17.2]                  74                    75                 0.2
 (17.2,18.7]                 121                   196                 0.3
 (18.7,20.3]                  98                   294                 0.2
 (20.3,21.9]                 101                   395                 0.3
 Frecuencia_Relativa_Acumulada Frecuencia_Porcentual
                           0.0                   0.3
                           0.2                  18.7
                           0.5                  30.6
                           0.7                  24.8
                           1.0                  25.6
 Frecuencia_Porcentual_Acumulada
                             0.3
                            19.0
                            49.6
                            74.4
                           100.0
```

```
Medidas de tendencia central:

Media: 18.9
Cuartil 1: 17.5
Mediana: 18.8
Cuartil 3: 20.4
Moda (frecuencia): 18.7, 19.9, 17.9, 20.6
Moda (datos agrupados): 18.3

Medidas de variabilidad:

Mínimo: 14
Máximo: 21.9
Rango: 7.9
Desviación Estándar: 1.7
Varianza: 2.9
Coeficiente de Variación (%): 9
```

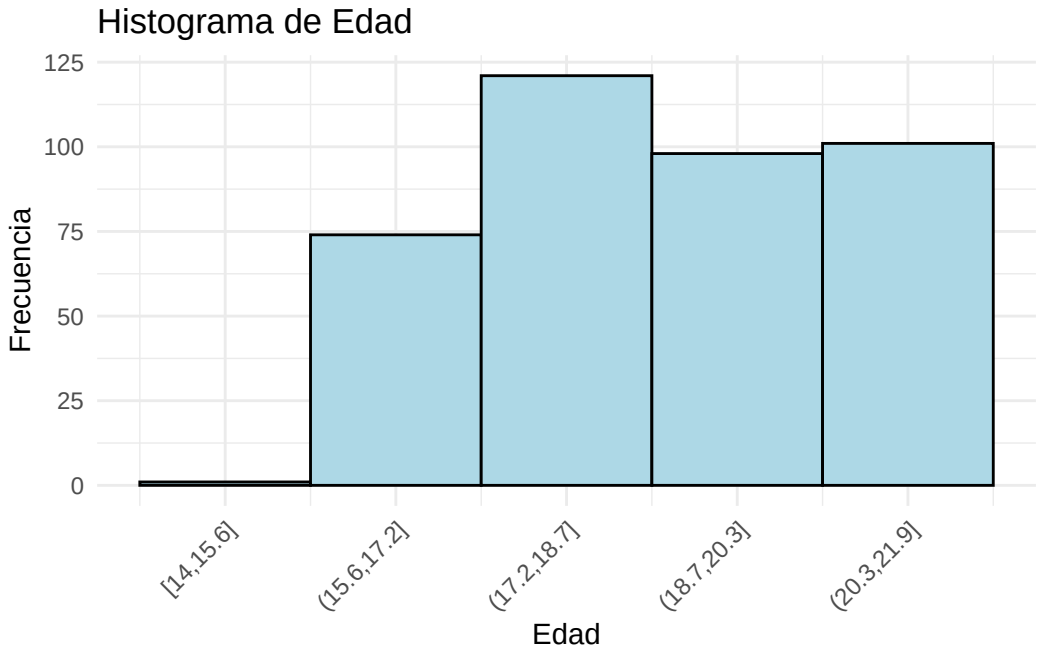
```
#nota: para usar S4 se debe asignar a un objeto la función aplicada, como se muestra en el e
```

Ejemplo 4:

```
#Pruebas de impresión de gráfico y de objetos asignados en S4

#primera forma de imprimir el gráfico

resultado@grafico
```




```
#segunda forma de imprimir el gráfico

plot(resultado)
```



```r
#forma para imprimir los objetos (ejemplo valores_atipicos)

resultado@valores_atipicos
```

```
[1] 24.91 27.88 29.85 12.04 28.76
```

EXTRA

Finalmente, como parte de una prueba e investigación, implementé la función en forma de un paquete instalable, el cual está correctamente documentado y se presenta a continuación:

```r
#intslación del paquete "tablaFrecuencias"

install.packages("tablaFrecuencias_0.1.0.tar.gz", repos = NULL, type = "source")
```

```
Installing package into 'C:/Users/humbe/AppData/Local/R/win-library/4.5'
(as 'lib' is unspecified)
```

```
Warning in install.packages("tablaFrecuencias_0.1.0.tar.gz", repos = NULL, :
installation of package 'tablaFrecuencias_0.1.0.tar.gz' had non-zero exit
status
```

Ejemplo 5: haciendo uso de la función llamando a la libreria

```r
library(tablaFrecuencias)
```

```
Adjuntando el paquete: 'tablaFrecuencias'
```

```
The following object is masked _by_ '.GlobalEnv':

    tabla_frecuencias
```

```r
library(readr)

datos <- read.csv("datos.csv")
View(datos)

#variable usada: Puntaje
```

```r
#uso simple (usa numero de intervalos utilizando la regla de Sturges, y redondea a dos decima

tabla_frecuencias(data = datos, column = "Puntaje")
```

```
Tabla de Frecuencias, Medidas de Tendencia Central y Dispersion para la variable Puntaje

   Intervalo Frecuencia_Absoluta Frecuencia_Acumulada Frecuencia_Relativa
     [0,8.1]                   5                    5                0.01
  (8.1,16.2]                  45                   50                0.11
 (16.2,24.3]                 118                  168                0.29
 (24.3,32.4]                  98                  266                0.24
 (32.4,40.5]                  51                  317                0.13
 (40.5,48.6]                   6                  323                0.01
 (48.6,56.7]                  22                  345                0.06
 (56.7,64.8]                   5                  350                0.01
 (64.8,72.9]                  30                  380                0.07
   (72.9,81]                  20                  400                0.05
 Frecuencia_Relativa_Acumulada Frecuencia_Porcentual
                          0.01                  1.25
                          0.12                 11.25
                          0.42                 29.50
                          0.66                 24.50
                          0.79                 12.75
                          0.81                  1.50
                          0.86                  5.50
                          0.88                  1.25
                          0.95                  7.50
                          1.00                  5.00
 Frecuencia_Porcentual_Acumulada
                            1.25
                           12.50
                           42.00
                           66.50
                           79.25
                           80.75
                           86.25
                           87.50
                           95.00
                          100.00
```

```
Medidas de tendencia central:

Media: 31.78
Cuartil 1: 19
Mediana: 26.5
Cuartil 3: 35
Moda (frecuencia): 28
Moda (datos agrupados): 22.56

Medidas de variabilidad:

Minimo: 0
Maximo: 81
Rango: 81
Desviacion Estandar: 18.85
Varianza: 355.33
Coeficiente de Variacion (%): 59.31
```

```
#Nos arroja la tabla de frecuencia junto a sus medidas de tendencia central y de variabilidad
```