R para el Análisis Econométrico

Módulo 1: Fundamentos Estadísticos en R

José Burgos | ASOECO

07-09-2025

Section 1

Parte I: Fundamentos de R y RStudio

R y RStudio

Bienvenida a R y RStudio

- ¿Qué es R? Un lenguaje de programación para computación estadística y gráficos.
- ¿Qué es RStudio? El Entorno de Desarrollo Integrado (IDE).
- Filosofía de R: código abierto, extensible a través de paquetes y con una gran comunidad.

R y RStudio

Hacer las cosas bien desde el inicio

- Buenas prácticas desde el inicio: Investigación Reproducible.
- Introducción al concepto de proyectos en RStudio para mantener el trabajo organizado.
- Entorno de trabajo y creación de un primer proyecto.

Manejo del entorno y flujo de trabajo

- Scripts de R: Cómo escribir y ejecutar código desde un archivo .R.
- Atajos de teclado clave en RStudio para agilizar el trabajo (ejecutar código, asignación, pipe).

Section 2

Primeros pasos en la programación con R

Operadores aritméticos

Tal como sugiere su nombre, estos operadores se emplea específicamente para realizar cálculos aritméticos. Es factible efectuar estas operaciones utilizando datos tanto enteros como numéricos.

Operador	Operación	Ejemplo	Resultado
+	Suma	5+3	8
-	Resta	5-3	2
*	Multiplicación	5*3	15
/	División	5/3	1.666667
^	Potencia	5^3	125
%%	División entera	5%%3	2

El orden de las operaciones es similar al que se usa en matemáticas: primero se resuelven las potencias, luego las multiplicaciones y divisiones, y finalmente las sumas y restas.

Operadores relacionados

Se usan los operadores lógicos para hacer comparaciones y su resultado siempre será `TRUE` o `FALSE` (verdadero o flaso, respectivamente)

Operador	Comparación	Ejemplo	Resultado
<	Menor que	`5 < 3`	`FALSE`
<=	Menor o igual que	`5 <= 3`	`FALSE`
>	Mayor que	`5 > 3`	`TRUE`
>=	Mayor o igual que	`5 >= 3`	`TRUE`
==	Exactamente igual que	`5 == 3`	`FALSE`
!=	No es igual que	`5 != 3`	`TRUE`

Casi siempre cuando se hacen comparaciones siempre obtendremos TRUE o FALSE.

Operadores lógicos

Los operadores lógicos son usados para describir relaciones lógicas, expresadas como verdadero o falso.

Operador	Comparación
`x	y`	`x` ó `y` es verdadero
`x & y`	`x` Y `y` son verdaderos
`!x`	`x` no es verdadero

Advertencia

- `|` devuelve `TRUE` si alguno de los datos es `TRUE`
- `&` solo devuelve `TRUE` si ambos datos es `TRUE`
- `|` solo devuelve `FALSE` si ambos datos son `FALSE`
- `&` devuelve `FALSE` si alguno de los datos es `FALSE`

Operadores de asignación

Los operadores de asignación son los más importantes, nos permiten asignar datos a variables. Se recomienda utilizar <- a la hora de asignar.

```
juan <- 5
jose =  5
```

Consideraciones:

- Los nombres de las variables deben comenzar con una letra (a-z, A-Z).
- No pueden comenzar con un número, contener espacios o caracteres especiales (%, $, &, etc.), excepto el guion bajo (_).
- R es sensible a mayúsculas y minúsculas (por ejemplo, `edad` y `Edad` son variables diferentes).

Section 3

Tipos de Datos y Estructuras

Tipos de datos

Los tipos de datos más común en R son:

Tipo	Ejemplo	Nombre en inglés
Entero	1	integer
Numérico	1.3	numeric
Cadena de texto	"uno"	character
Factor	uno	factor
Lógico	TRUE	logical
Perdido	`NA`	NA
Vacio	`NULL`	null

Para consultar el tipo de datos, usaremos la función `class()`

Coerción

Podemos manipular los datos en R, o más precisamente, realizar conversiones forzadas de un tipo a otro utilizando las funciones de la familia as: `as.interger()`, `as.numeric()`, `as.character()`, `as.factor()`, `as.logical()`, `as.null()` . Todas estas funciones trabajan con datos y vectores.

Ejemplos:

```
as.character(5)
```

```
## [1] "5"
```

```
as.logical(1)
```

```
## [1] TRUE
```

Coerción

Advertencia

Se debe utilizar esta funciones con cuidado, ya que pueden generar resultados inesperados. Siempre se debe de evaluar antes de convertir un tipo de dato a otro. Por ejemplo: si pasas un texto a numérico, el resultado será `NA`. Si pasas un número a lógico, el resultado será `TRUE` si el número es distinto de cero y `FALSE` si es cero, pero si es mayor de 1, el resultado será `TRUE`.

Estructuras de datos

Vectores

Un vector es una colección de uno o más datos del mismo tipos. Los vectores siempre tienen tres propiedades:

1. **Tipo:** El tipo de dato que contiene el vector (numérico, carácter, lógico, etc.)
2. **Longitud:** El número de elementos en el vector.
3. **Atributos:** Metadatos adicionales que describen el vector (como nombres de elementos). Por ejemplo, un vector puede tener nombres para cada uno de sus elementos: `c(a = 1, b = 2, c = 3)`.

Vectores

```
vec <- 1 # Crear un vector
# Vectores de más de un elemento usar c()
vector1 <- c(1,3,4,5,6)
vector2 <- 1:10

vector2 + 2 # Vectorización
```

```
##  [1]  3  4  5  6  7  8  9 10 11 12
```

Data frames

Un data frame es una colección de vectores de igual longitud, donde cada vector representa una columna y cada fila representa una observación. Los data frames son especialmente útiles para manejar conjuntos de datos tabulares, como los que se encuentran comúnmente en análisis estadísticos y científicos.

```
head(iris, n = c(3,3))
```

```
##   Sepal.Length Sepal.Width Petal.Length
## 1          5.1         3.5          1.4
## 2          4.9         3.0          1.4
## 3          4.7         3.2          1.3
```

Creando un data frames

```r
# Crear vectores
nombre <- c("Ana", "Luis", "Marta")
edad <- c(23, 30, 25)
peso <- c(55.5, 70.2, 60.3)

# Crear data frame
estudiantes <- data.frame(nombre, edad, peso)
```

Acceder a los elementos de un data frame

```
# Acceder a una columna
estudiantes$nombre
```

```
## [1] "Ana"   "Luis"  "Marta"
```

```
# Acceder a una fila
estudiantes[1, ]
```

```
##   nombre edad peso
## 1    Ana   23 55.5
```

Acceder a los elementos de un data frame

```
# Acceder a un elemento específico
estudiantes[1, 2]
```

```
## [1] 23
```

```
# Acceder a múltiples filas y columnas
estudiantes[1:2, c("nombre", "edad")]
```

```
##   nombre edad
## 1    Ana   23
## 2   Luis   30
```

Section 4

Ejercicios practicos

Ejercicios

1. Crea un vector llamado `edades` que contenga las edades de cinco personas. Ejemplo: 25, 30, 22, 28, 35. Luego, muestra el vector en la consola.
2. Utiliza operadores aritméticos para calcular la suma, resta, multiplicación y división de dos números. Asigna el resultado de cada operación a una variable diferente y muestra los resultados en la consola.
3. Crea un data frame llamado `estudiantes` con las siguientes columnas: `nombre` (carácter), `edad` (numérico) y `aprobado` (lógico). Llena el data frame con datos de cinco estudiantes.
4. Utiliza la función `class()` para verificar el tipo de datos de cada columna en el data frame `estudiantes`.

Resultados

```
# Ejercicio 1
edades <- c(25, 30, 22, 28, 35)
edades
```

```
## [1] 25 30 22 28 35
```

Resultados

```r
# Ejercicio 2
num1 <- 10
num2 <- 5
suma <- num1 + num2
resta <- num1 - num2
multiplicacion <- num1 * num2
division <- num1 / num2
suma; resta; division; multiplicacion
```

```
## [1] 15
```

```
## [1] 5
```

```
## [1] 2
```

```
## [1] 50
```

Resultados

```r
# Ejercicio 3
nombre <- c("Ana", "Luis", "Marta", "Carlos", "Sofia")
edad <- c(23, 30, 25, 28, 22)
aprobado <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
estudiantes <- data.frame(nombre, edad, aprobado)
estudiantes
```

```
##   nombre edad aprobado
## 1    Ana   23     TRUE
## 2   Luis   30    FALSE
## 3  Marta   25     TRUE
## 4 Carlos   28     TRUE
## 5  Sofia   22    FALSE
```

Resultados

```
# Ejercicio 4
class(estudiantes$nombre)
```

```
## [1] "character"
```

```
# Ejercicio 4
class(estudiantes$aprobado)
```

```
## [1] "logical"
```

Section 5

Funciones y paquetes

Funciones

Las funciones son bloques de código reutilizables que realizan tareas específicas. En R, las funciones se definen utilizando la palabra clave `function()`. Las funciones pueden aceptar argumentos (entradas) y devolver valores (salidas).

```r
# Estructura básica de una función
mi_funcion <- function(arg1, arg2) {
  # Código que realiza una tarea
  resultado <- arg1 + arg2
  return(resultado)
}
```

Si queremos acceder a la documentación de una función en especial, podemos ejecutar `?mean` o `help(mean)`

Paquetes

Los paquetes son colecciones de funciones, datos y documentación que extienden las capacidades básicas de R. Existen miles de paquetes disponibles para diversas tareas, desde análisis estadísticos hasta visualización de datos.

Instalación y carga de paquetes

```r
# Para instalar un paquete, usamos la función
install.packages("tidyverse")

# Una vez instalado, debemos cargar el paquete
# en nuestra sesión de R utilizando:
library(tidyverse)
```

Funciones más usadas en dplyr

Función	Descripción
`filter()`	Filtra filas según condiciones específicas.
`select()`	Selecciona columnas específicas de un data frame.
`group_by()`	Agrupa datos según una o más columnas.
`mutate()`	Crea nuevas columnas o modifica existentes.
`arrange()`	Ordena filas según una o más columnas.
`summarise()`	Resume múltiples valores en un solo valor.

Ejemplos

```
library(tidyverse)
filter(estudiantes, edad > 25)
```

```
##   nombre edad aprobado
## 1   Luis   30    FALSE
## 2 Carlos   28     TRUE
```

```
select(estudiantes, nombre, edad)
```

```
##   nombre edad
## 1    Ana   23
## 2   Luis   30
## 3  Marta   25
## 4 Carlos   28
## 5  Sofia   22
```

Ejemplos

```
summarise(
  group_by(estudiantes, aprobado),
  edad_media = mean(edad), edad_max = max(edad)
  )
```

```
## # A tibble: 2 x 3
##   aprobado edad_media edad_max
##   <lgl>          <dbl>    <dbl>
## 1 FALSE           26         30
## 2 TRUE            25.3       28
```

Pipe

El operador pipe (%>% o |>) es una herramienta poderosa en R que permite encadenar múltiples operaciones de manera legible y concisa. Facilita la lectura del código al expresar una secuencia de transformaciones de datos.

```r
# Ejemplo usando el operador pipe
estudiantes %>%
  filter(edad > 25) %>%
  select(nombre, edad) %>%
  arrange(desc(edad))
```

```
##   nombre edad
## 1   Luis   30
## 2 Carlos   28
```

Section 6

Importación de datos

Importación de datos

Podemos cargar datos de diferentes formatos, como CSV, Excel, entre otros. Vamos a cargar la base de datos de la Encuesta Nacional de Fuerza de Trabajo (ENFCT) que se encuentra en formato excel.

Con el paquete `readxl` podemos leer archivos de Excel (.xlsx, .xls).

```
library(readxl)
enfct_2019 <- read_excel("enfct_2019.xlsx")
```

Evaluación de datos

Una vez que hemos importado los datos, es importante evaluar su estructura y contenido para entender qué tipo de análisis podemos realizar. Algunas funciones útiles para esta tarea son:

```
# Ver las primeras filas del data frame
head(enfct_2019)
# La más recomendables es glimpse() del paquete dplyr
glimpse(enfct_2019)
```

Section 7

Manipulación de datos con dplyr

Manipulación de datos con dplyr

El paquete `dplyr` es parte del ecosistema `tidyverse` y proporciona una gramática coherente para manipular datos. A continuación, se presentan algunas de las funciones más comunes y útiles de `dplyr` para transformar y analizar datos.

Tips opcional

Se recomienda limpiar los nombres de las variables, para evitar problemas al momento de manipular los datos. Para esto, usaremos la función `clean_names()` del paquete `janitor`.

Manipulación de datos con dplyr

```
library(janitor)
enfct_2019 <- clean_names(enfct_2019)
```

```
enfct_2019  %>%
  select(1:3) %>%
  tail(3) %>%
  glimpse()
```

```
## Rows: 3
## Columns: 3
## $ ano               <dbl> 2019, 2019, 2019
## $ id_provincia      <dbl> 12, 12, 12
## $ factor_expansion <dbl> 173.7651, 173.7651, 173.7651
```

Manipulación de datos con dplyr

Primer paso:

Definimos las variables que nos interesa y evaluamos su contenido. En este ejercicio vamos a selecionar las variables `edad`, `sexo`, `des_provincia`, `sueldo_bruto_ap_monto`, `estado_civil` y `nivel_ultimo_ano_aprobado`.

```
enfct_2019 <- enfct_2019 %>%
  select(edad, sexo, des_provincia,
         sueldo_bruto_ap_monto, estado_civil,
         nivel_ultimo_ano_aprobado)
```

Manipulación de datos con dplyr

Evaluamos el contenido de las variables e identificamos los cambios necesarios para nuestros datos.

```
enfct_2019 %>%
  tail(3) %>%
  glimpse()
```

```
## Rows: 3
## Columns: 6
## $ edad                          <dbl> 26, 4, 32
## $ sexo                          <chr> "F", "F", "M"
## $ des_provincia                 <chr> "LA ROMANA", "LA ROMANA",
## $ sueldo_bruto_ap_monto         <dbl> 2500, NA, 2000
## $ estado_civil                  <chr> "Union libre", NA, "Separ
## $ nivel_ultimo_ano_aprobado     <chr> "Secundario", "Ninguno",
```

Manipulación de datos con dplyr

Con count, podemos identificar varios detalles: los valores dentro de la variables y la frecuencia de cada valor. Una segunda opción es la función `distinct`, que nos muestra los valores únicos dentro de una variable.

```
enfct_2019 %>%
  count(sexo)
```

```
## # A tibble: 2 x 2
##   sexo      n
##   <chr> <int>
## 1 F     41872
## 2 M     41159
```

Manipulación de datos con dplyr

```
enfct_2019 %>%
  distinct(estado_civil)
```

```
## # A tibble: 7 x 1
##   estado_civil
##   <chr>
## 1 Separado(a)
## 2 <NA>
## 3 Casado(a)
## 4 Soltero(a)
## 5 Union libre
## 6 Viudo(a)
## 7 Divorciado(a)
```

Manipulación de datos con dplyr

Vamos a realizar algunos cambios en las variables que seleccionamos. Primero, vamos a cambiar los valores de la variable `sexo`, que actualmente tiene los valores `F` y `M`, por `Femenino` y `Masculino`. Para esto, usaremos la función `mutate()` junto con recode.

```
enfct_2019 <- enfct_2019 %>%
  mutate(sexo = recode(sexo,
                       "F" = 1,
                       "M" = 0))
enfct_2019 %>% distinct(sexo)
```

```
## # A tibble: 2 x 1
##    sexo
##   <dbl>
## 1     1
## 2     0
```

Manipulación de datos con dplyr

Vamos a cambiar los valores de la variable `estado_civil`, que actualmente tiene los valores Separado(a), Casado(a), Soltero(a), Union libre, Viudo(a), Divorciado(a), por `Casado` y `Soltero`. Para esto, usaremos la función `mutate()` junto con `case_when()`.

```
enfct_2019 <- enfct_2019 %>%
  mutate(estado_civil = case_when(
    estado_civil %in% c(
      "Casado(a)", "Union libre") ~ "Casado",
    estado_civil %in% c(
      "Soltero(a)", "Viudo(a)", "Divorciado(a)",
      "Separado(a)") ~ "Soltero"
  ))
```

Manipulación de datos con dplyr

```
enfct_2019 %>% distinct(estado_civil)
```

```
## # A tibble: 3 x 1
##   estado_civil
##   <chr>
## 1 Soltero
## 2 <NA>
## 3 Casado
```

Section 8

Ejercicios prácticos

Ejercicios

1. Carga el conjunto de datos `mtcars` que viene incluido en R. Utiliza la función `head()` para ver las primeras filas del conjunto de datos.
2. Selecciona las columnas `mpg`, `cyl`, y `hp` del conjunto de datos `mtcars` y crea un nuevo data frame llamado `mtcars_subset`.
3. Filtra las filas del data frame `mtcars_subset` para incluir solo los autos con más de 6 cilindros (`cyl > 6`). Guarda el resultado en un nuevo data frame llamado `mtcars_filtered`.
4. Crea una nueva columna en el data frame `mtcars_filtered` llamada `power_to_weight` que sea el resultado de dividir la potencia (`hp`) por el peso (`wt`). Utiliza la función `mutate()` para esto.

Section 9

Resultados

Resultados

```r
data(mtcars)
# Ejercicio 1
head(mtcars, 6)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am
## Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1
## Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0  1
## Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1
## Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0
## Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0  0
## Valiant          18.1   6  225 105 2.76 3.460 20.22  1  0
```

Resultados

```
# Ejercicio 2
mtcars %>%
  select(mpg, cyl, hp) %>%
  head(6)
```

```
##                    mpg cyl  hp
## Mazda RX4         21.0   6 110
## Mazda RX4 Wag     21.0   6 110
## Datsun 710        22.8   4  93
## Hornet 4 Drive    21.4   6 110
## Hornet Sportabout 18.7   8 175
## Valiant           18.1   6 105
```

Resultados

```r
# Ejercicio 3
mtcars %>%
  filter(cyl > 6) %>%
  head(6)
```

```
##                    mpg cyl  disp  hp drat   wt  qsec vs am
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.44 17.02  0  0
## Duster 360        14.3   8 360.0 245 3.21 3.57 15.84  0  0
## Merc 450SE        16.4   8 275.8 180 3.07 4.07 17.40  0  0
## Merc 450SL        17.3   8 275.8 180 3.07 3.73 17.60  0  0
## Merc 450SLC       15.2   8 275.8 180 3.07 3.78 18.00  0  0
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.25 17.98  0  0
```

Resultados

```r
# Ejercicio 4
mtcars %>%
  mutate(
    power_to_weight = hp / wt,
    .before = mpg
    ) %>%
  head(6)
```

```
##                   power_to_weight  mpg cyl disp  hp drat
## Mazda RX4                41.98473 21.0   6  160 110 3.90 2.
## Mazda RX4 Wag            38.26087 21.0   6  160 110 3.90 2.
## Datsun 710               40.08621 22.8   4  108  93 3.85 2.
## Hornet 4 Drive           34.21462 21.4   6  258 110 3.08 3.
## Hornet Sportabout        50.87209 18.7   8  360 175 3.15 3.
## Valiant                  30.34682 18.1   6  225 105 2.76 3.
##                   carb
## Mazda RX4            4
```

Section 10

Exportación de datos

Exportación de datos

Una vez que hemos manipulado y analizado nuestros datos, es importante guardar los resultados para su uso futuro o para compartirlos con otros. R ofrece varias funciones para exportar datos a diferentes formatos, como CSV, Excel y RDS.

Exportar a excel con openxlsx

El paquete `openxlsx` permite exportar data frames a archivos de Excel (.xlsx) de manera sencilla y eficiente.

```
# Pasos para exportar un data frame a Excel
library(openxlsx)
write.xlsx(enfct_2019, file = "enfct_2019_new.xlsx")
```

Section 11

Estadística descriptiva

Medidas de tendencia central

Las medidas de tendencia central son estadísticas que describen el centro o la ubicación típica de un conjunto de datos. Las tres medidas más comunes son la media, la mediana y la moda.

$$\text{Media} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\text{Mediana} = \begin{cases} \frac{x_{(n/2)} + x_{(n/2+1)}}{2}, & \text{si } n \text{ es par} \\ x_{((n+1)/2)}, & \text{si } n \text{ es impar} \end{cases}$$

Medidas de tendencia central

Medida	Descripción	Función en R
Media	El promedio aritmético de los datos.	`mean()`
Mediana	El valor central cuando los datos están ordenados.	`median()`
Moda	El valor que aparece con mayor frecuencia.	No hay función nativa, pero se puede calcular con `table()` y `which.max()`

Medidas de tendencia central

```
enfct_2019 %>%
  summarise(
    # na.rm = TRUE para ignorar los valores perdidos
    media = mean(edad, na.rm = TRUE),
    mediana = median(edad, na.rm = TRUE)
  )
```

```
## # A tibble: 1 x 2
##   media mediana
##   <dbl>   <dbl>
## 1  31.3      28
```

Medidas de dispersión

Las medidas de dispersión describen la variabilidad o dispersión de un conjunto de datos. Las dos medidas más comunes son la desviación estándar y la varianza.

Varianza (s^2)

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

Desviación Estándar (s)

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Medidas de dispersión

Medida	Descripción	Función en R
Varianza	La media de las desviaciones al cuadrado respecto a la media.	`var()`
Desviación Estándar	La raíz cuadrada de la varianza, que mide la dispersión en las mismas unidades que los datos.	`sd()`

Medidas de dispersión

```
enfct_2019 %>%
  summarise(
    varianza = var(edad, na.rm = TRUE),
    desviacion_estandar = sd(edad, na.rm = TRUE)
  )
```

```
## # A tibble: 1 x 2
##   varianza desviacion_estandar
##      <dbl>               <dbl>
## 1     466.                21.6
```

Medidas de posición

Las medidas de posición describen la ubicación relativa de un valor dentro de un conjunto de datos. Las medidas más comunes son los cuartiles, percentiles y cuantiles.

Percentil: El percentil P_k es el valor que divide el conjunto de datos en (k) partes iguales. Por ejemplo, el percentil 25 (P25) es el valor por debajo del cual se encuentra el 25% de los datos.

$$P_k = \text{Valor que divide el conjunto de datos en } k \text{ partes iguales}$$

Medidas de posición

Cuartiles: Los cuartiles Q_1, Q_2 y Q_3 son valores que dividen el conjunto de datos en cuatro partes iguales. Q_1 es el percentil 25, Q_2 es la mediana (percentil 50) y Q_3 es el percentil 75.

$Q_1, Q_2, Q_3 =$ Valores que dividen el conjunto de datos en cuatro partes iguale

Cuantiles: Los cuantiles q_p son valores que dividen el conjunto de datos en p partes iguales. Por ejemplo, el cuantil 0.25 (q0.25) es el valor por debajo del cual se encuentra el 25% de los datos.

$$q_p = \text{Valor que divide el conjunto de datos en } p \text{ partes iguales}$$

Resumiendo nuestras variables

```
enfct_2019 %>%
  summarise(
    q1 = quantile(edad, 0.25, na.rm = TRUE),
    q2 = quantile(edad, 0.50, na.rm = TRUE),
    q3 = quantile(edad, 0.75, na.rm = TRUE)
  )
```

```
## # A tibble: 1 x 3
##       q1    q2    q3
##    <dbl> <dbl> <dbl>
## 1     13    28    47
```

Section 12

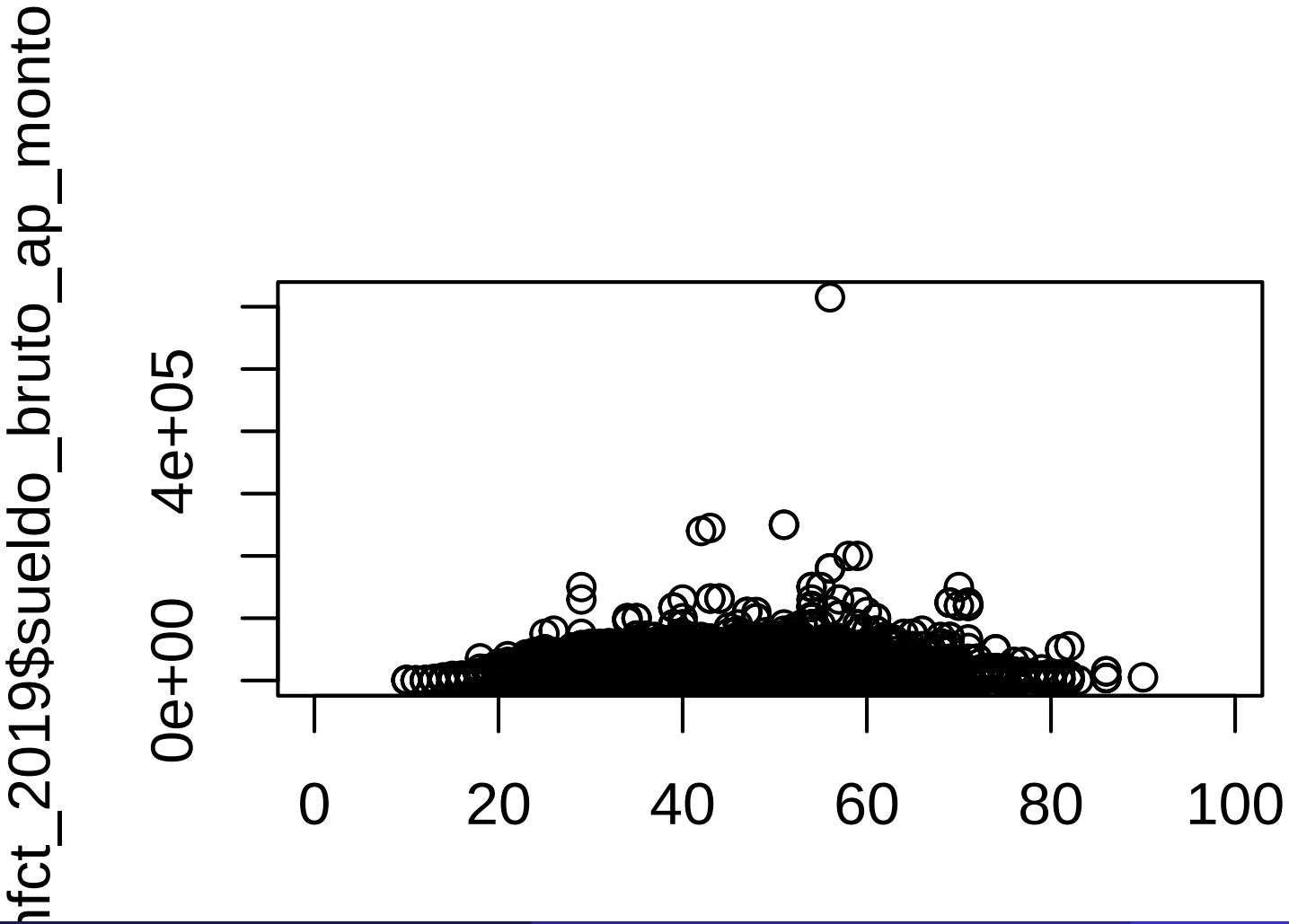
Gráficos en R con ggplot2

Gráficos de R base

R tiene funciones nativas para crear gráficos básicos, como `plot()`, `hist()`, `boxplot()`, entre otras. Estas funciones son fáciles de usar y permiten crear gráficos rápidamente. Su desventaja principal es que la personalización y la creación de gráficos complejos pueden ser limitadas.

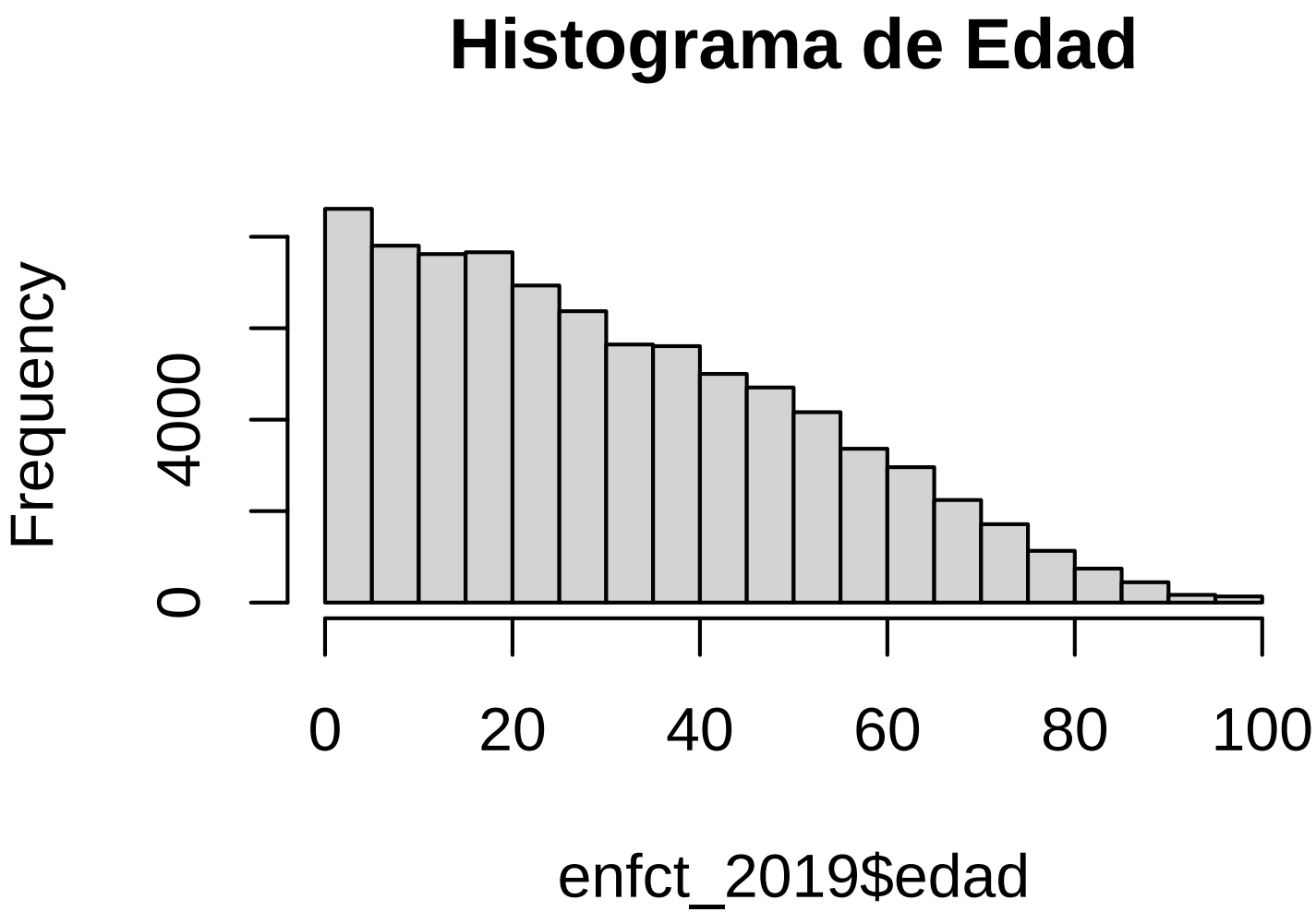
Dispersión en R base

```r
# Gráfico de dispersión
plot(enfct_2019$edad, enfct_2019$sueldo_bruto_ap_monto)
```



Histograma en R base

```r
hist(enfct_2019$edad, main = "Histograma de Edad")
```



Gráficos con ggplot2

Los gráficos con `ggplot2` se basan en la "gramática de gráficos", que permite construir gráficos capa por capa. Esto facilita la creación de gráficos complejos y altamente personalizados. Las capas principales incluyen:

- `ggplot()`: El lienzo inicial.
- `aes()`: Mapeo de variables a estéticas (ejes, color, forma).
- Geoms: Las representaciones visuales de los datos (puntos, barras, líneas).

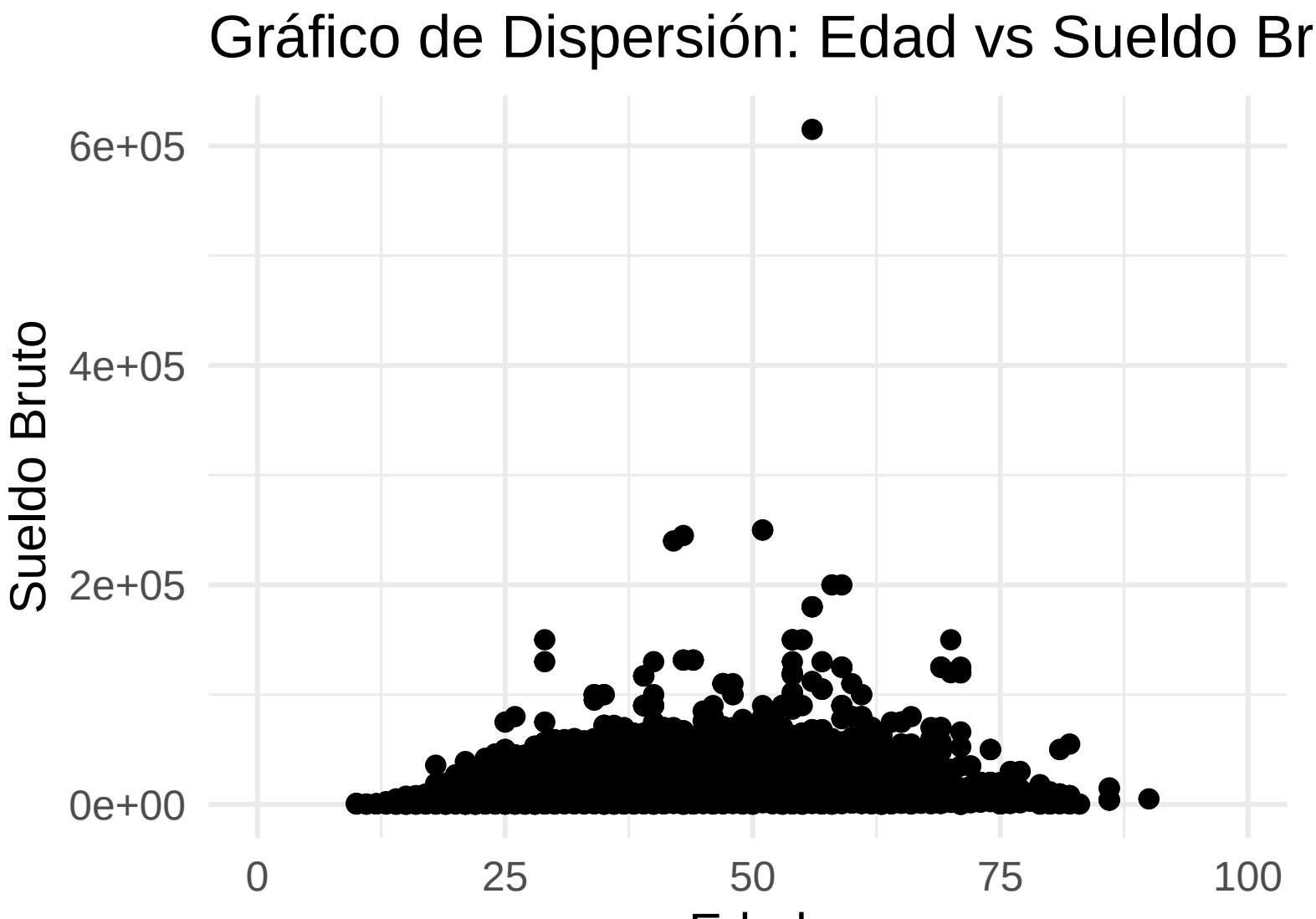
Dispersión con ggplot2

```
library(ggplot2)

plot_dispersion <- enfct_2019 %>%
  ggplot(aes(x = edad, y = sueldo_bruto_ap_monto)) +
  geom_point() +
  labs(title = "Gráfico de Dispersión: Edad vs Sueldo Bruto",
       x = "Edad",
       y = "Sueldo Bruto") +
  theme_minimal()
```

Dispersión con ggplot2

```
plot_dispersion
```



Histograma con ggplot2

```
plot_histograma <- enfct_2019 %>%
  ggplot(aes(x = edad)) +
  geom_histogram(
    binwidth = 5, fill = "blue",
    color = "black", alpha = 0.7) +
  labs(title = "Histograma de Edad",
       x = "Edad",
       y = "Frecuencia") +
  theme_minimal()
```

Histograma con ggplot2

```
plot_histograma
```

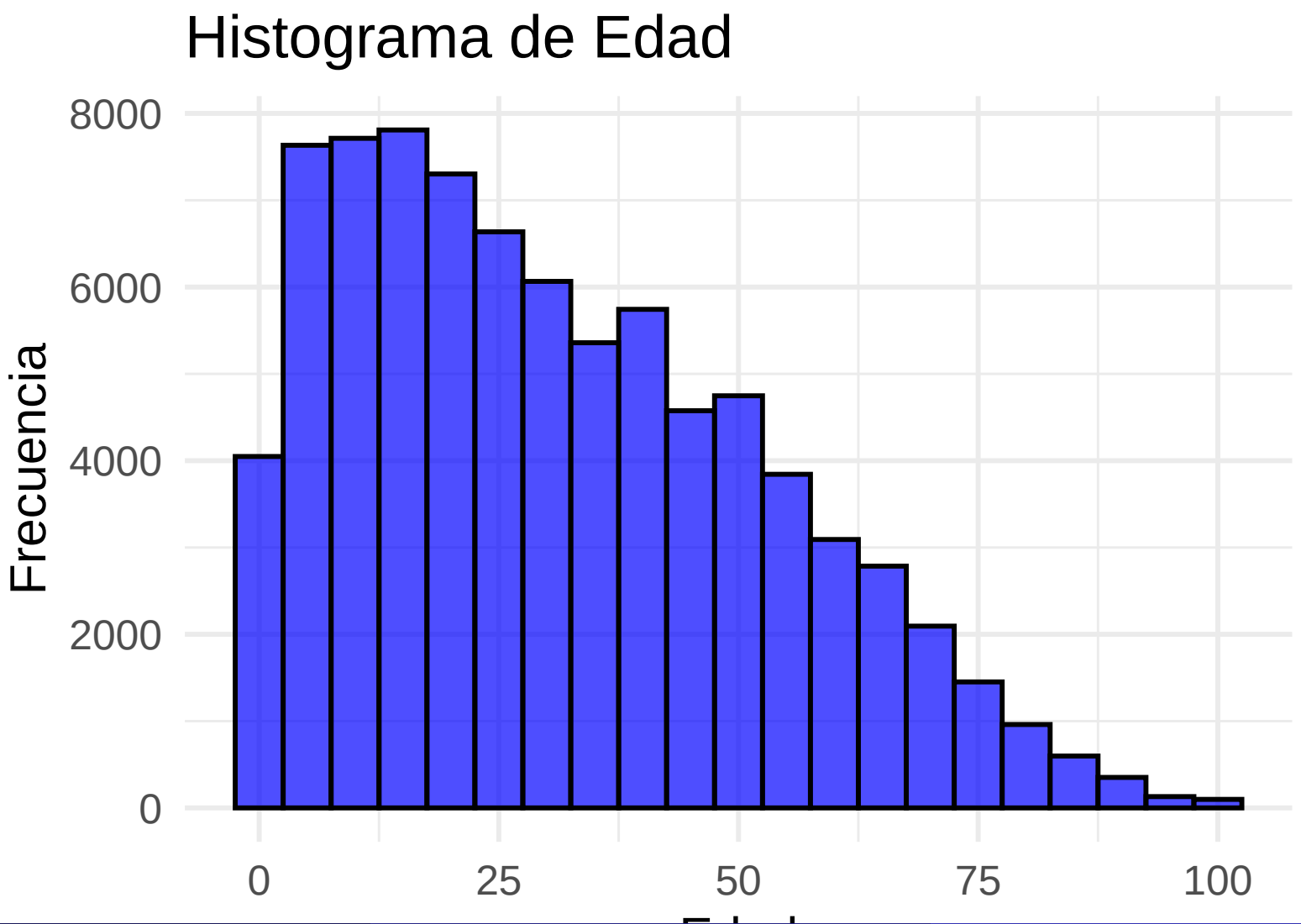
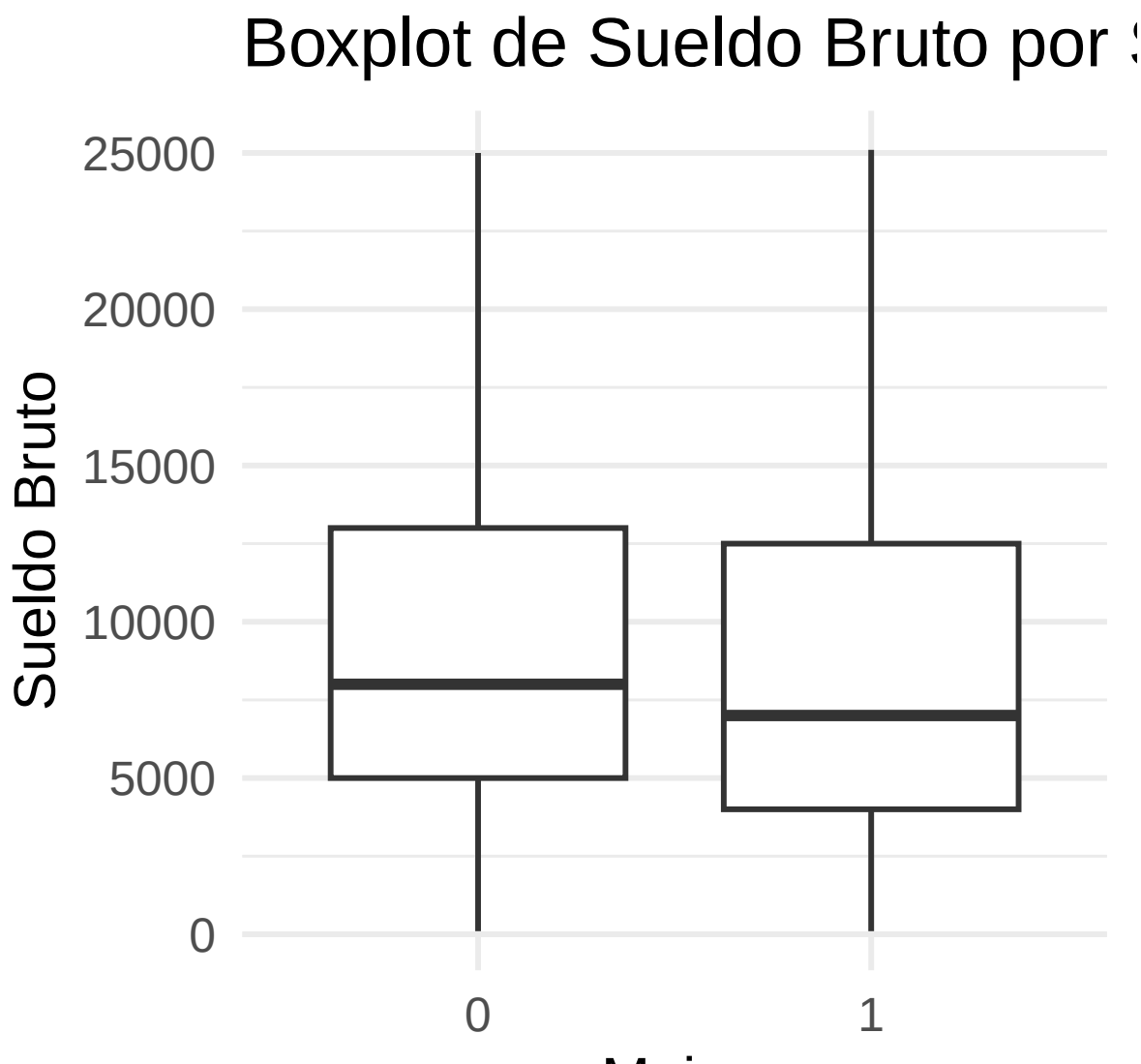


Gráfico de boxplot

```
plot_boxplot <- enfct_2019 %>%
  ggplot(aes(x = factor(sexo), y = sueldo_bruto_ap_monto)) +
  geom_boxplot(outliers = FALSE) +
  labs(title = "Boxplot de Sueldo Bruto por Sexo",
       x = "Mujer",
       y = "Sueldo Bruto") +
  theme_minimal()
```

Gráfico de boxplot

```
plot_boxplot
```



Section 13

Ejercicios prácticos

Ejercicios

1. Calcula los estadisticos aprendidos aquí para el ingreso de los individuos, edad.
2. Calcula los estadisticos aprendidos aquí para el ingreso de los individuos, segmentando por sexo, y luego por estado civil.
3. Realiza el ejercicio anterior, pero ahora segmentando por sexo, y luego por estado civil.
4. Realiza un gráfico de dispersión entre edad y sueldo bruto, coloreando por sexo.

Ejercicios

5. Calcula el promedio de sueldo bruto por provincia y realiza un gráfico de barras con los resultados. Sientete libre en personalizar el gráfico.
6. ¿Los ingresos más bajo a que provincia pertenecen?
7. ¿Son los solteros más joven que los casados?
8. ¿Los hombres ganan más que las mujeres?

Section 14

Referencia

Referencia

Mendoza Vega, J. B. (2020). *R para principiantes: Introducción al análisis de datos con R y RStudio* (4a ed.). Editorial Académica Española. https://bookdown.org/jboscomendoza/r-principiantes4/

Wickham, H., & Grolemund, G. (2017). *R para ciencia de datos* (1a ed.). Ediciones Díaz de Santos. https://r4ds.hadley.nz/index.html

Villarroel-Riquelme, F. (2020). *Introducción a la Estadística con R y RStudio*. https://rpubs.com/franciscovillarroel/estadistica_r

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer. https://www.econometrics-with-r.org/1-introduction.html