

# Análisis de encuestas de hogares con R

## Módulo 5: Análisis gráfico en R

CEPAL - Unidad de Estadísticas Sociales

# Tabla de contenidos I

Graficas en R

Gráficas de variables continuas.

Scaterplot

Diagrama de barras para variables categoricas

Creando mapas

# Graficas en R

# Introducción

- ▶ Este apartado se centra en mostrar cómo crear gráficos generales en R, lo cual es esencial en el análisis de encuestas para visualizar tendencias y verificar supuestos en el ajuste de modelos estadísticos.
- ▶ Se introduce el paquete ggplot2, una herramienta poderosa y flexible para la creación de gráficos elegantes en R. Fue desarrollado por Hadley Wickham y se basa en el concepto de “Grammar of Graphics.”
- ▶ La carga inicial de librerías y bases de datos es un paso común antes de comenzar cualquier análisis gráfico en R.

## Lectura de la base

```
encuesta <- readRDS("Imagenes/02_variable_continua/ENIGH_HND_Hogar.rds")

encuesta <- encuesta %>% # Base de datos.
  mutate(estrato = haven::as_factor(F1_A0_ESTRATO),
        TIPOVIVIENDA = haven::as_factor(F1_A1_P1_TIPOVIVIENDA),
        Area = haven::as_factor(F1_A0_AREA),
        TIENEVEHICULOS = haven::as_factor(F2_A2_P1_TIENEVEHICULOS))
```

## Definir diseño de la muestra con srvyr

Definiendo el diseño muestral, esto se hace de forma análoga a la anterior.

```
diseno <- encuesta %>% as_survey_design(
  strata = estrato, # Id de los estratos.
  ids = F1_A0_UPM,      # Id para las observaciones.
  weights = Factor,     # Factores de expansión.
  nest = TRUE           # Valida el anidado dentro del estrato
)
```

## Definir nuevas variables

Creando nuevas variables, para ello se hace uso de la función `mutate`.

```
diseno <- diseno %>% mutate(  
  ingreso_per = ifelse(YDISPONIBLE_PER < 0 , 0 , YDISPONIBLE_PER) ,  
  pobreza_LP = case_when(  
    ingreso_per < 3046 & Area == "1. Urbana" ~ 1,  
    ingreso_per < 1688 &  
      Area == "1. Rural" ~ 1,  
    TRUE ~ 0  
) ,  
  pobreza_LI = case_when(  
    ingreso_per < 1955 & Area == "1. Urbana" ~ 1,  
    ingreso_per < 1110 &  
      Area == "1. Rural" ~ 1,  
    TRUE ~ 0  
) ,  
  ingreso_hog = ingreso_per * CANTIDAD_PERSONAS ,  
  log_ingreso_per = log(ingreso_per + 500) ,  
  log_ingreso_hog = log(ingreso_hog + 500) ,  
  log_gasto = log(GASTO CORRIENTE HOGAR + 500)
```

# Aplicación en encuestas de hogares

CANTIDAD_PERSONAS	log_gasto	pobreza_LI	pobreza_LP	ingreso_per	ingreso_hog
1	9.841	0	0	1007333	1007333
3	10.192	0	0	311155	933465
2	12.248	0	0	377934	755868
4	11.752	0	0	188521	754083
4	12.551	0	0	182975	731900
11	11.424	0	0	54348	597830
4	11.204	0	0	137257	549027
2	12.400	0	0	270820	541640
2	11.310	0	0	225302	450604
8	12.392	0	0	51434	411475
4	12.066	0	0	101303	405213
4	11.512	0	0	94049	376197
1	11.039	0	0	374406	374406
6	11.421	0	0	62285	373708
6	10.950	0	0	61710	370258
4	11.572	0	0	92236	368943
7	12.607	0	0	52667	368667
4	10.561	0	0	90084	360334
5	12.784	0	0	69084	345422
6	11.241	0	0	56523	339139

## Sub-grupos

Dividiendo la muestra en sub grupos.

```
sub_Urbano <- diseno %>% filter(Area == "1. Urbana") #  
sub_Rural <- diseno %>% filter(Area == "2. Rural") #
```

## Creando tema para las gráficas

Para tener un estilo estándar las gráficas se define el siguiente tema.

```
theme_cepal <- function(...) {
  theme_light(10) +
  theme(
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    legend.position = "bottom",
    legend.justification = "left",
    legend.direction = "horizontal",
    plot.title = element_text(size = 20, hjust = 0.5),
    ...
  )
}
```

Gráficas de variables continuas.

## Histogramas para graficar variables continuas.

- ▶ Un histograma es una representación gráfica de los datos de una variable que utiliza barras, donde la altura de las barras representa la frecuencia de los valores y el ancho de las barras corresponde a la amplitud de los intervalos de clase.
- ▶ Los pasos para realizar el histograma incluyen definir la fuente de información, especificar la variable a graficar (x) y los pesos de muestreo (weight). Luego, se elige el tipo de gráfico, en este caso, un histograma (geom\_histogram). Además, se configuran los títulos deseados para el histograma y se aplica el tema de la CEPAL para mejorar su presentación.

# Histogramas

Las gráficas son realizadas principalmente con la librería ggplot2y nos apoyamos en la librería patchwork para organizar la visual de las gráficas.

```
library(ggplot2)
library(patchwork)
plot1_Ponde <- ggplot(
  data = diseno$variables,           # Fuente de datos.
  aes(x = log_ingreso_hog,
      weight = Factor) # Parámetros gráficos general.
) +
  geom_histogram(                   # Parámetro geométrico.
    aes(y = ..density..)) +        # Parámetros del gráfico
  ylab("") +                      # Nombre para el eje Y
  ggtitle("Ponderado") +          # Titulo.
  theme_cepal() # Aplicando tema
```

# Histogramas

De forma análoga se define el gráfico siguiente, note que en este caso se omitió el parámetro `weight`.

```
plot1_SinPonde <-
  ggplot( data = diseno$variables,
  aes(x = log_ingreso_hog)) +
  geom_histogram(aes(y = ..density..)) +
  ylab("") +
  ggtitle("Sin ponderar") +
  theme_cepal()
# Organizando la salida gráfica
plot1 <- plot1_Ponde | plot1_SinPonde
plot1
```

# Histograma

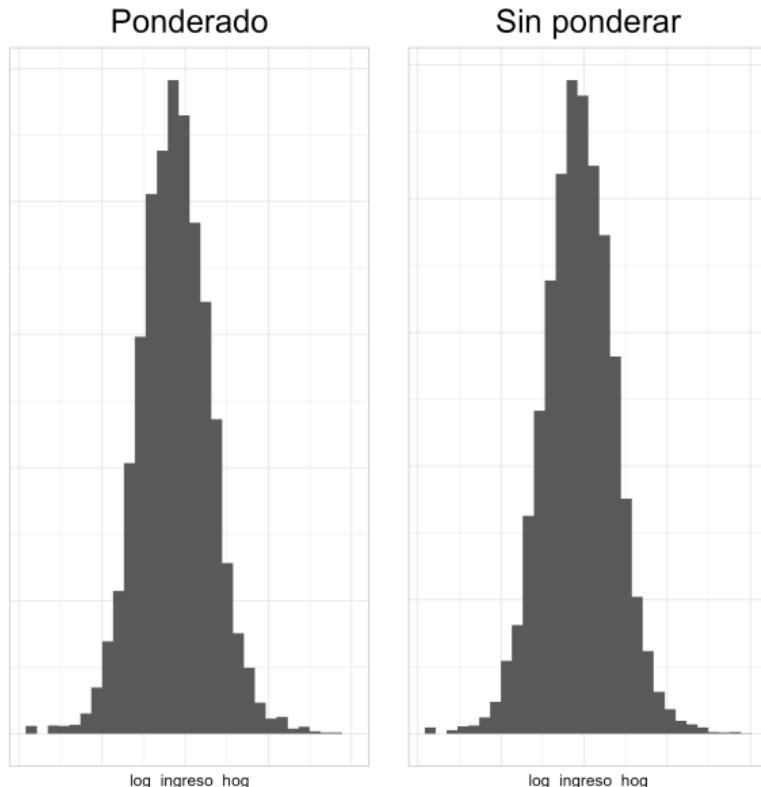


Figura 1: Histograma base para *log ingreso en eñ hogar*

## Histogramas

Ahora, repetimos la secuencia de gráficos para la variable *log\_gasto*

```
plot2_Ponde <- ggplot(  
  data = diseno$variables,  
  aes(x = log_gasto , weight = Factor)  
) +  
  geom_histogram(aes(y = ..density..)) +  
  ylab("") +  
  ggtitle("Ponderado") +  
  theme_cepal()
```

# Histogramas

```
plot2_SinPonde <- ggplot(data = diseno$variables,
  aes(x = log_gasto )) +
  geom_histogram(aes(y = ..density..)) +
  ylab("") +
  ggtitle("Sin ponderar") +
  theme_cepal()

plot2 <- plot2_Ponde | plot2_SinPonde
```

# Histogramas

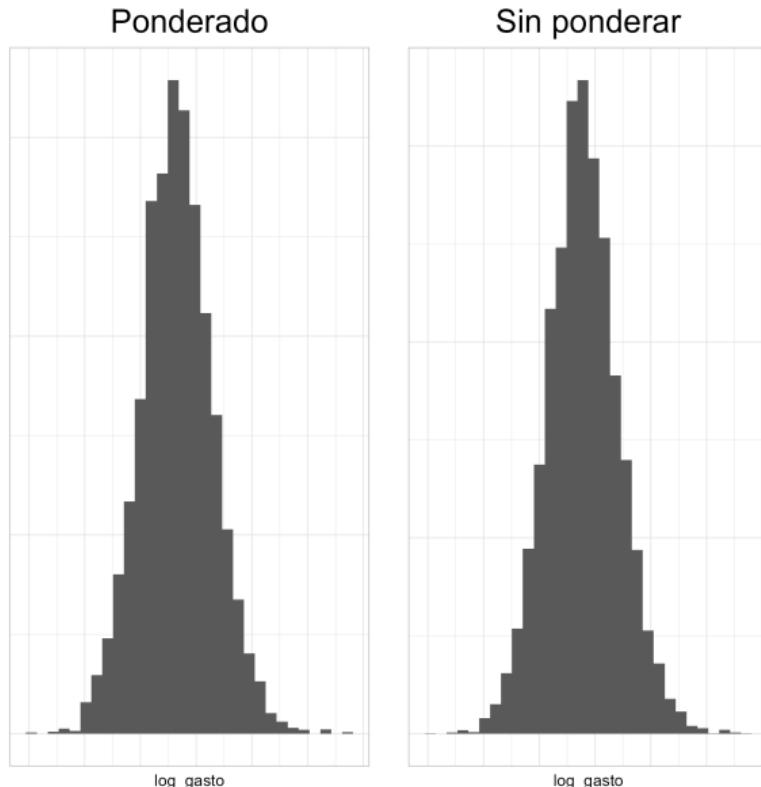


Figura 2: Histograma base para *log gasto corriente en el hogar*

## Histogramas por sub-grupos

Cuando el interés es realizar comparaciones entre dos o más agrupaciones, es posible hacer uso del parámetro `fill`, el cual “rellena” las barras del histograma con diferentes colores según sea el grupo.

```
plot3_Ponde <- ggplot(
  diseno$variables,
  aes(x = log_ingreso_hog , weight = Factor)
) +
  geom_histogram(
    aes(y = ..density.., fill = Area),
    alpha = 0.5,
    position = "identity" # Para que las barras no estén apiladas.
) +
  ylab("") +
  ggtitle("Ponderado") +
  theme_cepal()
```

## Histogramas por sub-grupos

La sintaxis es homologa a la anterior, sin embargo, se retiro el parámetro weight.

```
plot3_SinPonde <-
  ggplot(diseno$variables, aes(x = log_ingreso_hog )) +
  geom_histogram(aes(y = ..density.., fill = Area),
    alpha = 0.5, position = "identity"
  ) +
  ggtitle("Sin ponderar") +
  theme_cepal() +
  ylab("")
```

plot3 <- plot3\_Ponde | plot3\_SinPonde

# Histogramas por sub-grupos

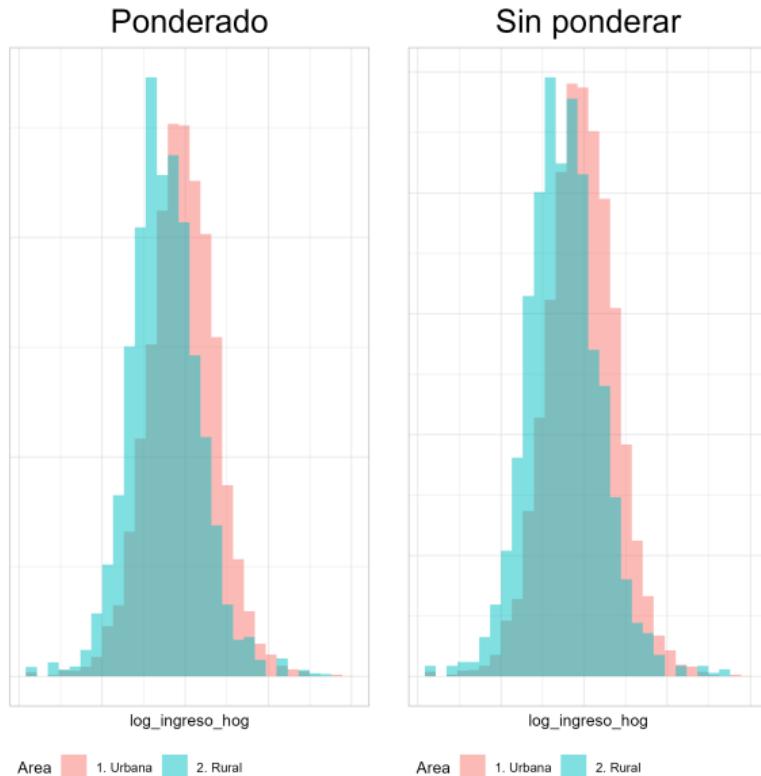


Figura 3: Histograma para *log. del ingreso del hogar*

## Histogramas por sub-grupos

Ahora, repetimos la secuencia de gráficos para la variable *log\_gasto*

```
plot4_Ponde <- ggplot(
  diseno$variables,
  aes(x = log_gasto, weight = Factor)
) +
  geom_histogram(aes(y = ..density.., fill = Area),
    alpha = 0.5, position = "identity"
) +
  ylab("") +
  ggtitle("Ponderado") +
  theme_cepal()
```

## Histogramas por sub-grupos

```
plot4_SinPonde <- ggplot(
  diseno$variables,
  aes(x = log_gasto)
) +
  geom_histogram(aes(y = ..density.., fill = Area),
    alpha = 0.5, position = "identity"
) +
  ggtitle("Sin ponderar") +
  theme_cepal() +
  ylab("")
plot4 <- plot4_Ponde | plot4_SinPonde
```

# Histogramas por sub-grupos

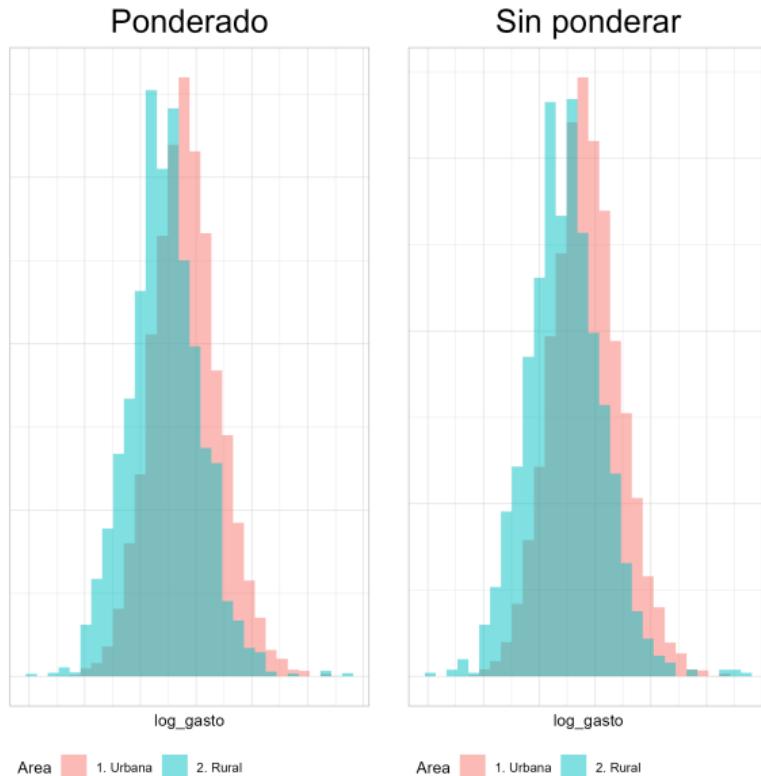


Figura 4: Histograma para  $\log_{10}$  gasto

## Histogramas por sub-grupos

Ahora, repetimos la secuencia de gráficos para la variable *log ingreo\_hog*, pero haremos el relleno por la variable *tiene vehículo*.

```
plot5_Ponde <-  
  ggplot(diseno$variables,  
          aes(x = log_ingreso_hog, weight = Factor)) +  
  geom_histogram(  
    aes(y = after_stat(density), fill = TIENEVEHICULOS),  
    alpha = 0.5,  
    position = "identity") +  
  ylab("") +  
  ggtitle("Ponderado") +  
  theme_cepal()
```

## Histogramas por sub-grupos

```
plot5_SinPonde <- ggplot(diseno$variables,
    aes(x = log_ingreso_hog)) +
geom_histogram(
    aes(y = after_stat(density), fill = TIENEVEHICULOS),
    alpha = 0.5,
    position = "identity") +
ggtitle("Sin ponderar") +
theme_cepal() +
ylab("")
plot5 <- plot5_Ponde | plot5_SinPonde
```

# Histogramas por sub-grupos

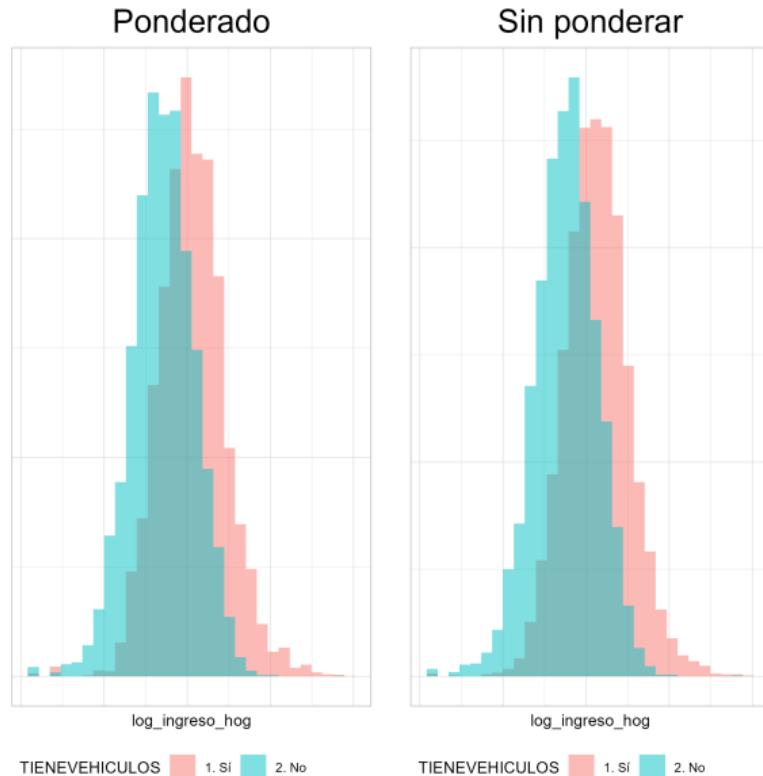


Figura 5: Histograma para *log ingreso del hogar*

## Histogramas por sub-grupos

Ahora, repetimos la secuencia de gráficos para la variable *log\_gasto* y el relleno por la variable *TIENEVEHICULOS*.

```
plot6_Ponde <- ggplot(diseno$variables,
                      aes(x = log_gasto, weight = Factor)) +
  geom_histogram(
    aes(y = after_stat(density),
        fill = TIENEVEHICULOS),
    alpha = 0.5,
    position = "identity") +
  ylab("") +
  ggtitle("Ponderado") +
  theme_cepal()
```

## Histogramas por sub-grupos

```
plot6_SinPonde <- ggplot(diseno$variables,
                           aes(x = log_gasto)) +
  geom_histogram(
    aes(y = after_stat(density),
        fill = TIENEVEHICULOS),
    alpha = 0.5,
    position = "identity") +
  ylab("") +
  ggtitle("Sin ponderar") +
  theme_cepal() +
  ylab("")  
plot6 <- plot6_Ponde | plot6_SinPonde
```

# Histogramas por sub-grupos

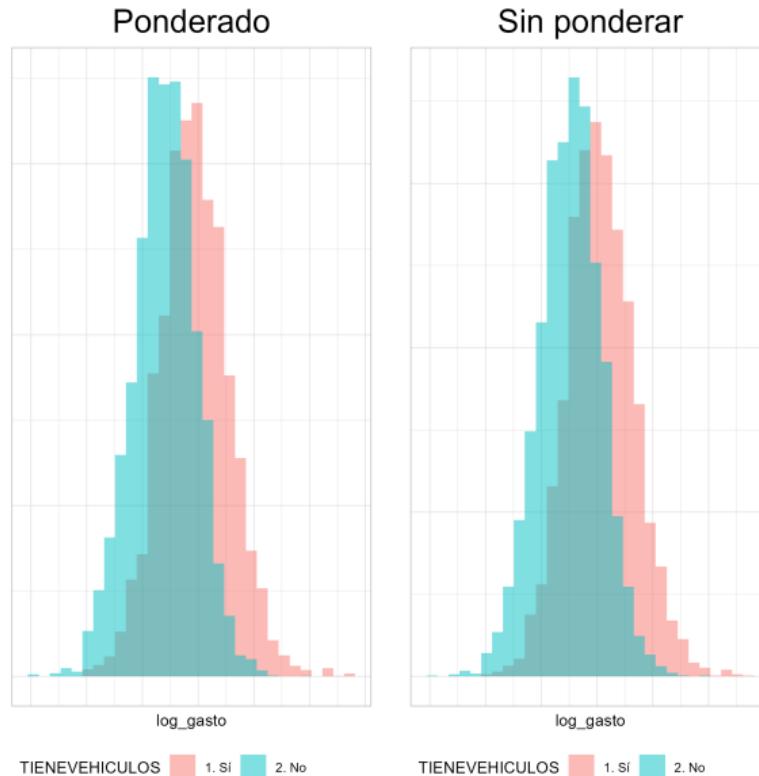


Figura 6: Histograma para *log\_gasto*

## Agregando densidad

Dadas las cualidades de la librería ggplot2, podemos agregar nuevas capas a la gráfica. Por ejemplo, la densidad con la función `geom_density` e incorporamos el parámetro `alpha` que regula la transparencia del relleno.

```
plot1_desy <- plot1_Ponde + geom_density(fill = "#ADD8E6", alpha = 0.3)  
plot2_Ponde + geom_density(fill = "#ADD8E6", alpha = 0.3)
```

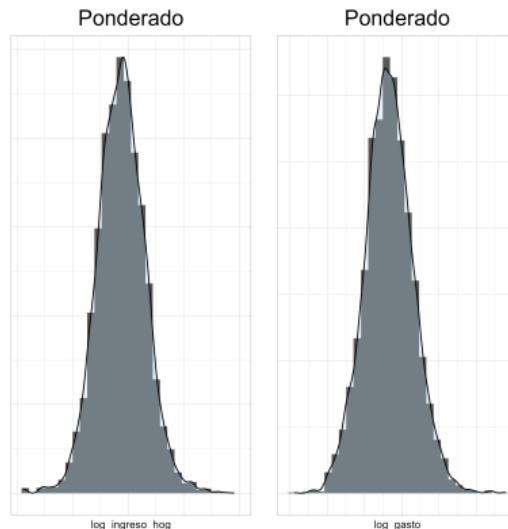


Figura 7: Desnsidad agregada al histograma

## Agregando densidad

Al hacer `aes(fill = Area)` permite que la densidad sea agregada para cada una de las agrupaciones.

```
plot3_densy <- plot3_Ponde + geom_density(aes(fill = Area), alpha = 0.3) |  
  plot4_Ponde + geom_density(aes(fill = Area), alpha = 0.3)
```

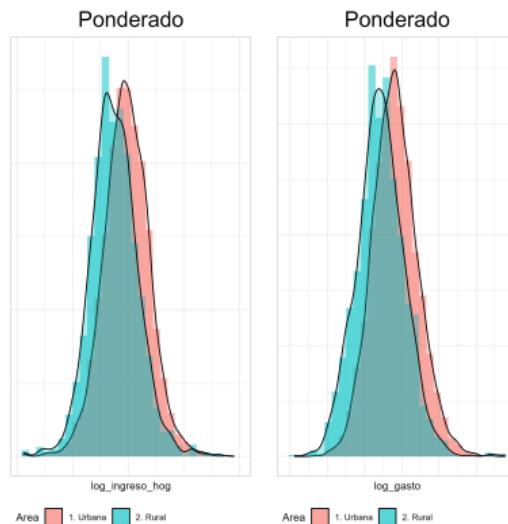


Figura 8: Desnsidad para agregada al histograma por área

## Agregando densidad

En esta oportunidad se agrega la densidad por sexo

```
plot5_densy <- plot5_Ponde +  
  geom_density(aes(fill = TIENEVEHICULOS),  
               alpha = 0.3) |  
  plot6_Ponde + geom_density(aes(fill = TIENEVEHICULOS),  
                           alpha = 0.3)
```

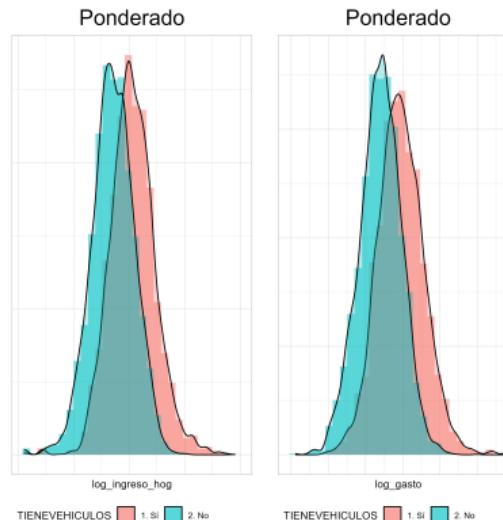


Figura 9: Densidad agregada al histograma por tiene vehículo

# Boxplot

- ▶ El boxplot es un gráfico ampliamente utilizado en estadísticas que fue presentado por John Tukey en 1977. Proporciona una representación resumida de un conjunto de datos utilizando cinco números clave.
- ▶ El boxplot consiste en un rectángulo llamado “caja” y dos segmentos llamados “bigotes”. Este gráfico muestra información sobre la relación entre los cuartiles (Q1, Q2 o mediana y Q3) y los valores mínimo y máximo del conjunto de datos, la presencia de valores atípicos y la simetría de la distribución.
- ▶ Para crear boxplots en R utilizando ggplot2, se emplea la función `geom_boxplot`.

## Boxplot

Otro gráfico que podemos hacer son los diagramas de caja, para esto deben emplear la función `geom_boxplot`.

```
plot7_Ponde <- ggplot( diseno$variables,
                      aes(x = log_ingreso_hog,
                          weight = Factor)) +
  geom_boxplot() + ggttitle("Ponderado") +
  coord_flip() + theme_cepal()

plot8_Ponde <- ggplot( diseno$variables,
                      aes(x = log_gasto, weight = Factor)
) + geom_boxplot() + ggttitle("Ponderado") + coord_flip() +
  theme_cepal()

plot_78 <- plot7_Ponde | plot8_Ponde
```

# Boxplot

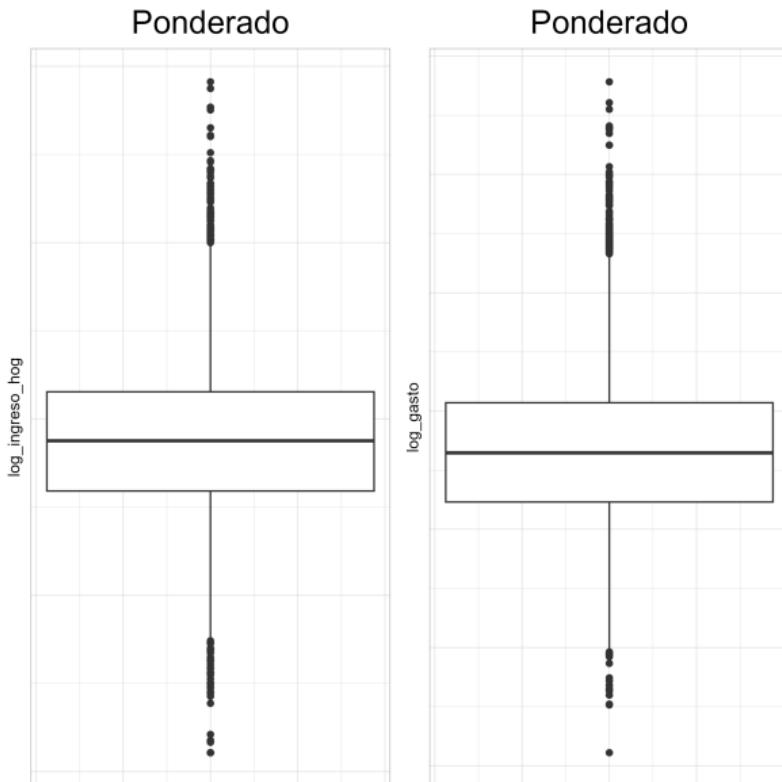


Figura 10: Boxplot para el log ingreso y log gasto

## Boxplot

Estos diagramas también permiten la comparación entre dos o más niveles de agrupamiento.

```
plot9_Ponde <- ggplot(diseno$variables,
                      aes(x = log_ingreso_hog,
                           weight = Factor)) +
  geom_boxplot(aes(fill = Area)) + ggttitle("Ponderado") +
  coord_flip() + theme_cepal()

plot10_Ponde <- ggplot( diseno$variables,
                        aes(x = log_gasto, weight = Factor) ) +
  geom_boxplot(aes(fill = Area)) + ggttitle("Ponderado") +
  coord_flip() + theme_cepal()

plot910_Ponde <- plot9_Ponde|plot10_Ponde
```

# Boxplot

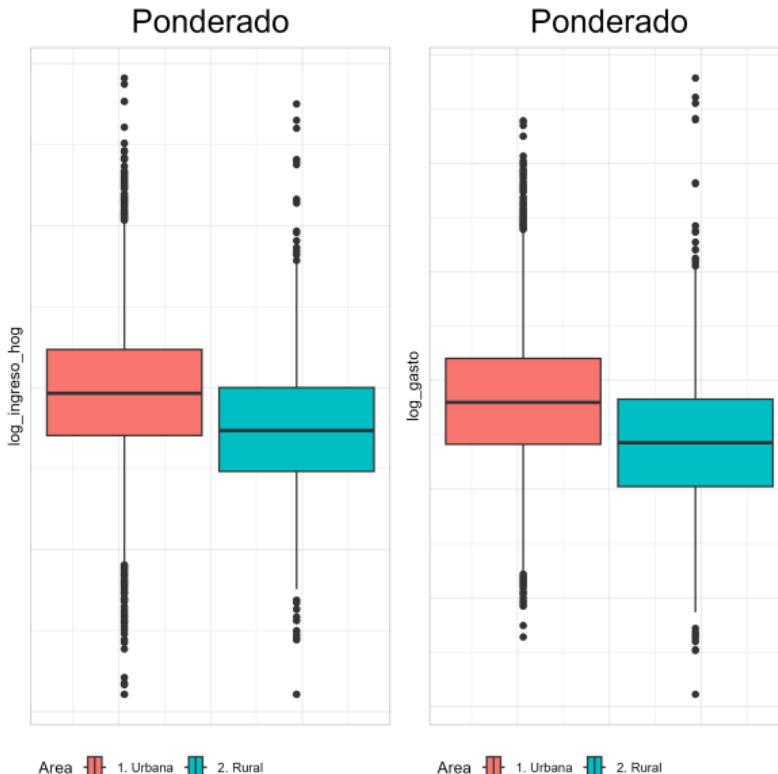


Figura 11: Boxplot para el ingreso y gasto por área

## Boxplot

Ahora, si desean personalizar los colores del relleno debe hacer uso de la función `scale_fill_manual`.

```
colorArea <- c("1. Urbana" = "#48C9B0", "2. Rural" = "#117864")
plot910_temp <-
  plot9_Ponde + scale_fill_manual(values = colorArea) |
  plot10_Ponde + scale_fill_manual(values = colorArea)
```

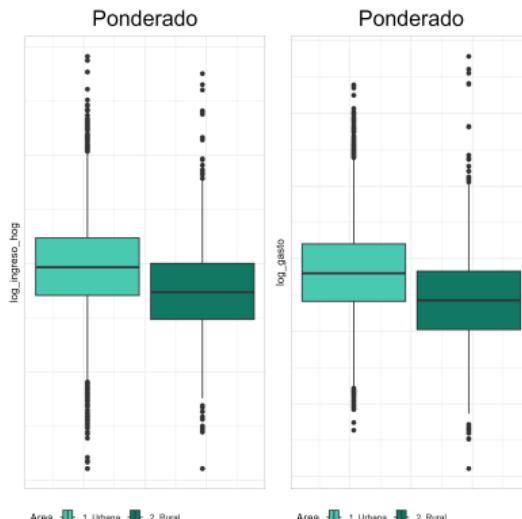


Figura 12: Boxplot para el ingreso y gasto por áreas

# Boxplot

Comparando los ingresos y gastos por tenencia de vehículo

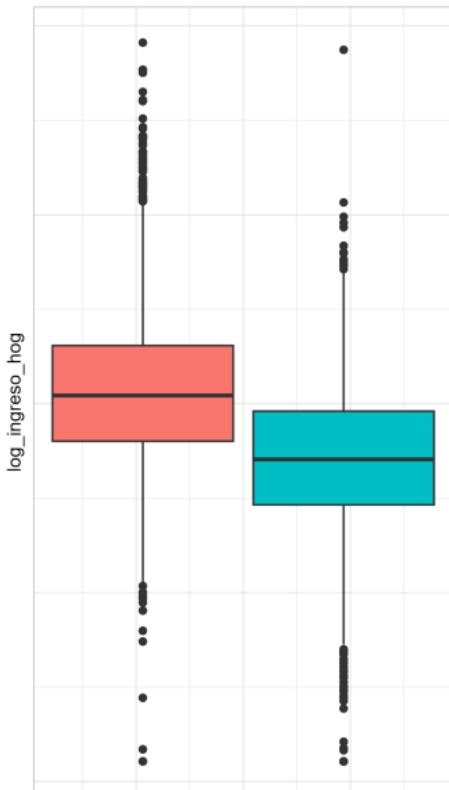
```
plot11_Ponde <- ggplot( diseno$variables,
  aes(x = log_ingreso_hog , weight = Factor)
) + geom_boxplot(aes(fill = TIENEVEHICULOS )) +
  ggtitle("Ponderado") +  coord_flip() +
  theme_cepal()

plot12_Ponde <- ggplot( diseno$variables,
  aes(x = log_gasto, weight = Factor)
) + geom_boxplot(aes(fill = TIENEVEHICULOS )) +
  ggtitle("Ponderado") +  coord_flip() +
  theme_cepal()

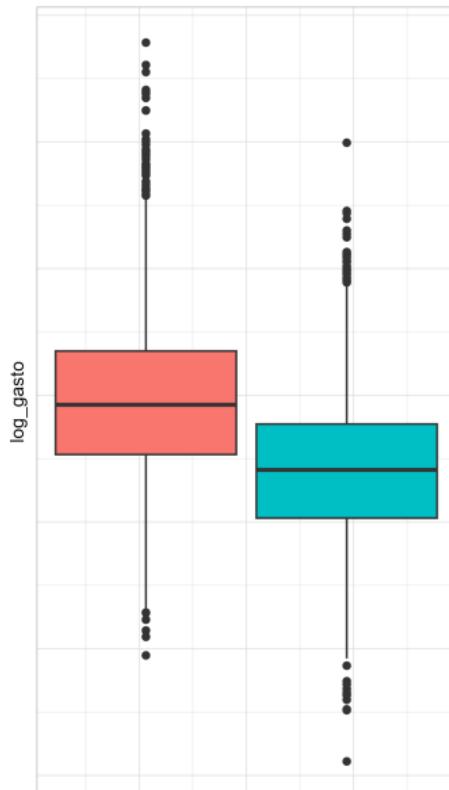
plot11_temp <- plot11_Ponde|plot12_Ponde
```

# Boxplot

Ponderado



Ponderado



TIENEVEHICULOS ■ 1. Sí ■ 2. No

TIENEVEHICULOS ■ 1. Sí ■ 2. No

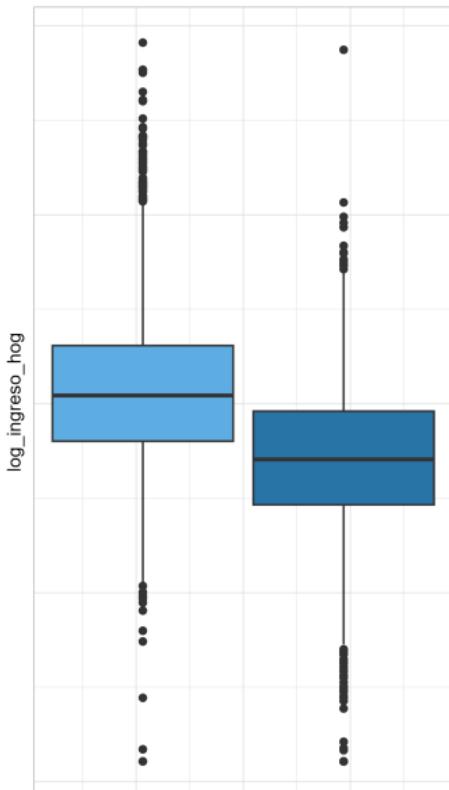
## Boxplot

Definiendo el color del relleno para Sí y No tiene vehículo.

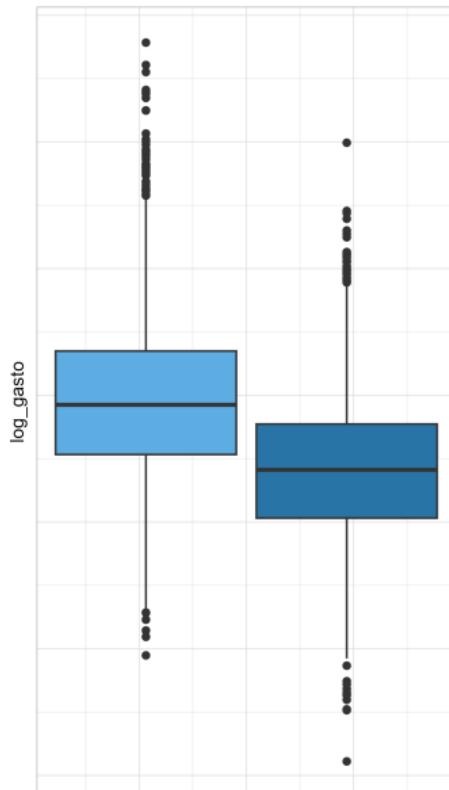
```
colorVehiculo <- c("1. Sí" = "#5DADE2", "2. No" = "#2874A6")  
  
plot11_temp <-  
plot11_Ponde + scale_fill_manual(values = colorVehiculo) |  
plot12_Ponde + scale_fill_manual(values = colorVehiculo)
```

# Boxplot

Ponderado



Ponderado



TIENEVEHICULOS    1. Sí    2. No

TIENEVEHICULOS    1. Sí    2. No

# Boxplot

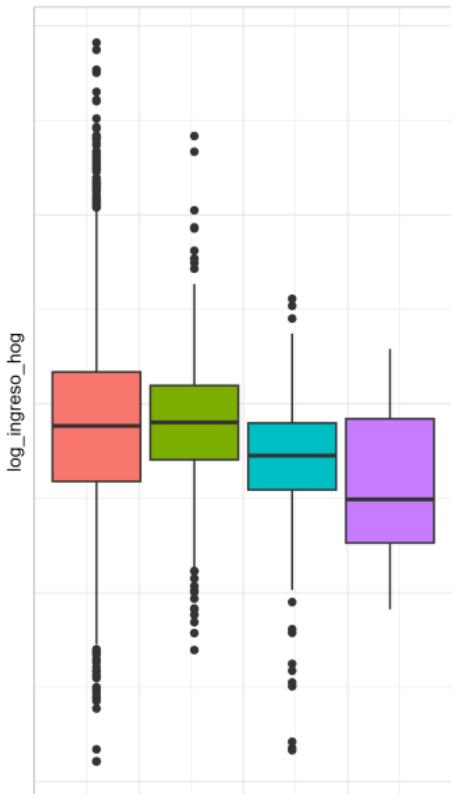
Realizando la comparación para más de dos categorías.

```
plot13_Ponde <- ggplot( diseno$variables,
  aes(x = log_ingreso_hog, weight = Factor)) +
  geom_boxplot(aes(fill = TIPOVIVIENDA )) +
  ggtitle("Ponderado") +    coord_flip() +
  theme_cepal()

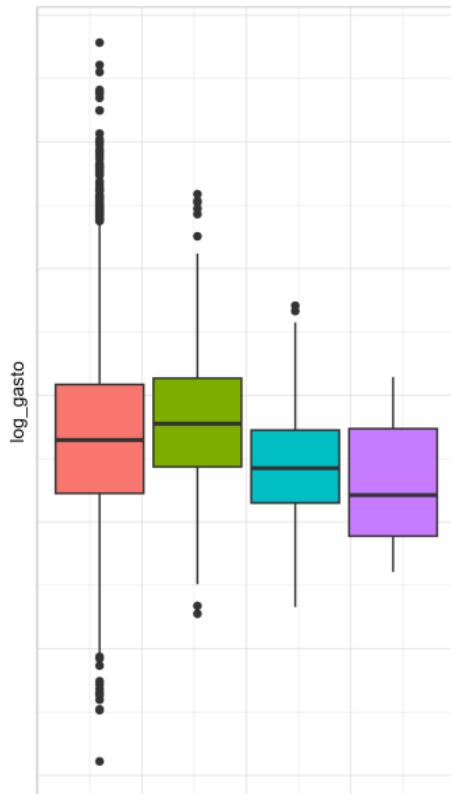
plot14_Ponde <- ggplot( diseno$variables,
  aes(x = log_gasto, weight = Factor)) +
  geom_boxplot(aes(fill = TIPOVIVIENDA)) +
  ggtitle("Ponderado") +    coord_flip() +
  theme_cepal()
```

# Boxplot

Ponderado



Ponderado



TIPOVIVIENDA ■ 1. Casa ■ 2. Apartamento ■ 3. TIPOVIVIENDA ■ 1. Casa ■ 2. Apartamento ■ 3. TIPOVIVIENDA ■ 1. Casa ■ 2. Apartamento ■ 3.

# Boxplot

Personalizando los coles cuando hay más de dos categorías.

```
colorvivienda <- c(  
  "1. Casa" = "#D6EAF8",  
  "2. Apartamento" = "#85C1E9",  
  "4. Local no construido para vivienda" = "#3498DB",  
  "3. Cuarto en mesón o cuartería" = "#2E86C1",  
  "5. Otro, especifique" = "#21618C"  
)  
pplot14_temp <-  
  plot13_Ponde + scale_fill_manual(values = colorvivienda) |  
  plot14_Ponde + scale_fill_manual(values = colorvivienda)
```

# Boxplot

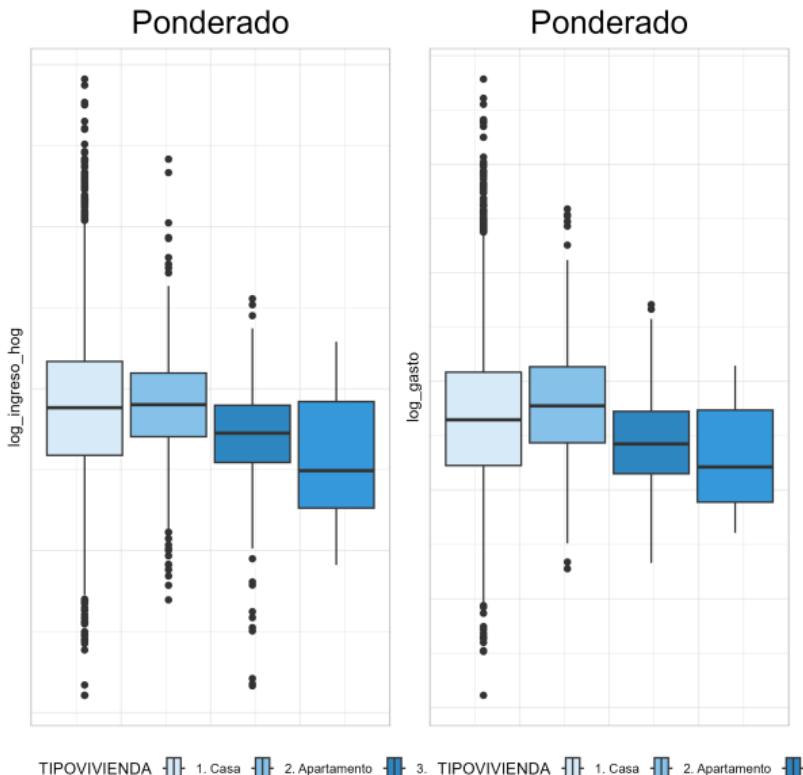


Figura 16: Boxplot para el ingreso y gasto por tipo de vivienda

## Boxplot

La función `geom_boxplot` permite realizar comparaciones con más de dos variables al tiempo. A continuación se compara el logaritmo ingresos por tenencia de vehículo en las diferentes áreas.

```
plot15_Ponde <-
  ggplot(
    diseno$variables,
    aes(x = log_ingreso_hog, y = Area, weight = Factor)
  ) +
  geom_boxplot(aes(fill = TIENEVEHICULOS)) +
  ggtitle("Ponderado") +
  scale_fill_manual(values = colorVehiculo) +
  coord_flip()
```

## Boxplot

De forma análoga podemos realizar la comparación de los gastos por sexo en las diferentes zonas.

```
plot16_Ponde <-
  ggplot(
    diseno$variables,
    aes(x = log_gasto, y = Area, weight = Factor)
  ) +
  geom_boxplot(aes(fill = TIENEVEHICULOS )) +
  ggtitle("Ponderado") +
  scale_fill_manual(values = colorVehiculo) +
  coord_flip()
```

# Boxplot

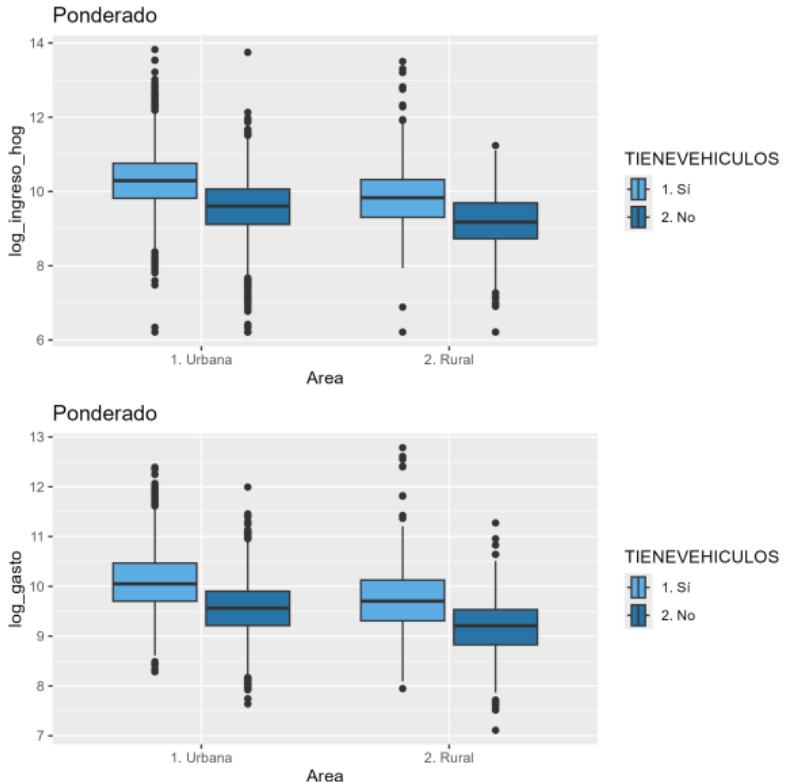


Figura 17: Boxplot para el logaritmo del ingreso y gasto

## Boxplot

Podemos extender las comparaciones variables que tienen más de dos categorías.

```
plot17_Ponde <-
  ggplot(
    diseno$variables,
    aes(x = log_ingreso_hog , y = TIPOVIVIENDA, weight = Factor)
  ) +
  geom_boxplot(aes(fill = TIENEVEHICULOS )) +
  ggtitle("Ponderado") +
  scale_fill_manual(values = colorVehiculo) +
  coord_flip()
```

## Boxplot

```
plot18_Ponde <-
  ggplot(
    diseno$variables,
    aes(
      x = log_gasto,
      y = TIPOVIVIENDA , weight = Factor
    )
  ) +
  geom_boxplot(aes(fill = TIENEVEHICULOS )) +
  ggtitle("Ponderado") +
  scale_fill_manual(values = colorVehiculo) +
  coord_flip()
```

```
plot17_Ponde / plot18_Ponde
```

# Boxplot

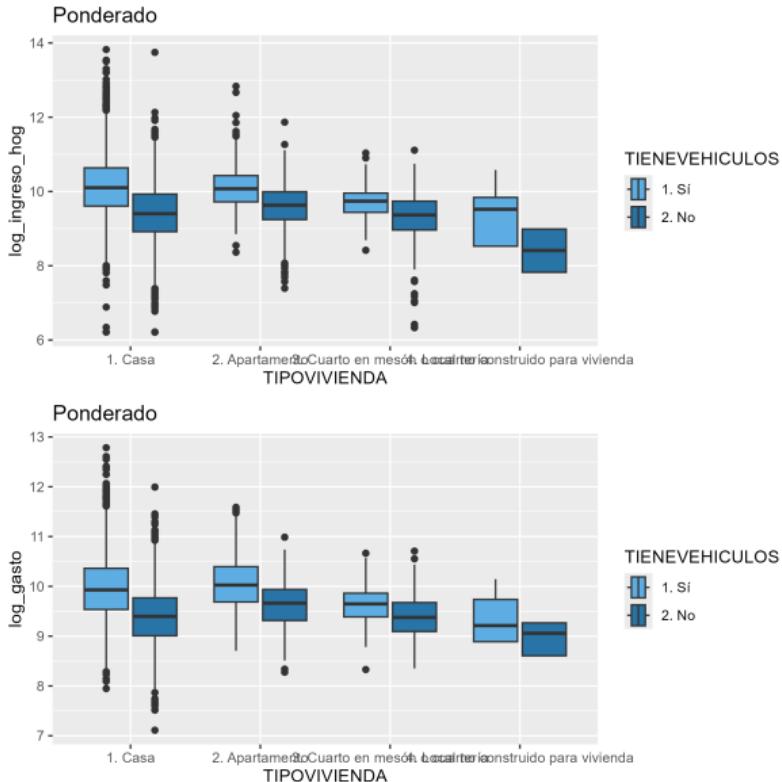


Figura 18: Boxplot para el ingreso y gasto por tipo de vivienda y tiene vehículo

Scaterplot

## Introducción

- ▶ Un diagrama de dispersión es una representación gráfica que muestra observaciones como puntos en un plano.
- ▶ Cada punto en el diagrama de dispersión se posiciona de acuerdo a los valores de dos variables.
- ▶ Los puntos pueden tener atributos como tamaño, color y forma, conocidos como estéticas.
- ▶ En R, se utiliza `geom_point` para crear un diagrama de dispersión.
- ▶ Es posible asignar estéticas a variables o establecer valores constantes para ellas en el gráfico.
- ▶ Se pueden utilizar diagramas de dispersión para visualizar la relación entre dos variables, identificar patrones o tendencias, y explorar datos.

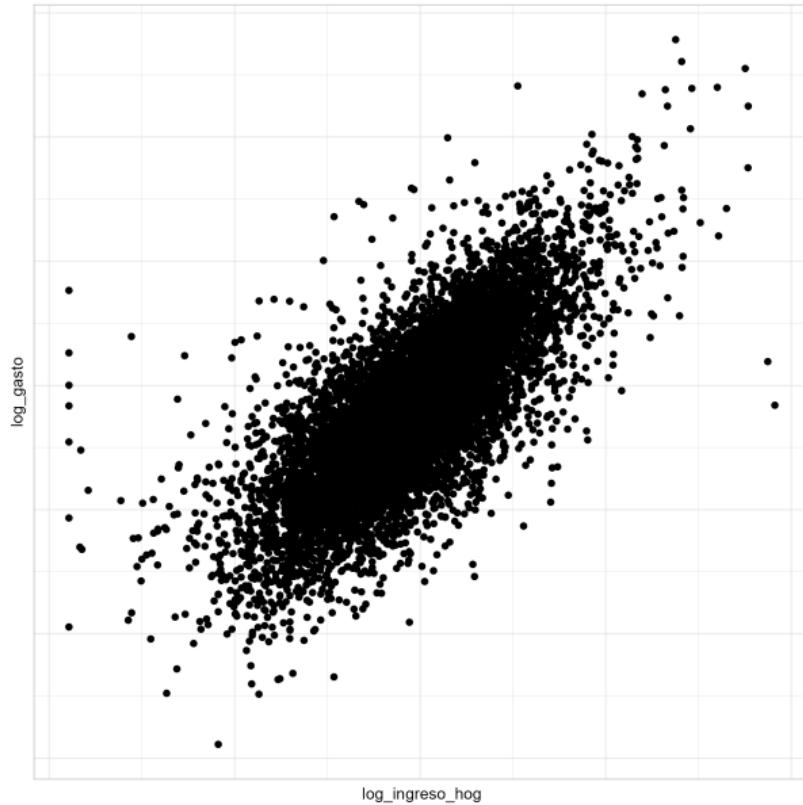
## Scaterplot

Para ejemplificar el uso de esta función, se graficarán las variables ingresos y gastos como se muestra a continuación:

```
plot19_Ponde <-
  ggplot(
    diseno$variables,
    aes( y = log_gasto, x = log_ingreso_hog,
        weight = Factor
    )
  ) +
  geom_point() +
  theme_cepal()
```

## Scaterplot

Note, que este caso el parámetro `weight` no esta aportando información visual al gráfico.



## Scaterplot

El parámetro `weight` lo podemos usar controlar el tamaño de los puntos de esa forma tener un mejor panorama del comportamiento de la muestra.

```
plot20_Ponde <-
  ggplot(
    diseno$variables,
    aes( y = log_gasto, x = log_ingreso_hog)
  ) +
  geom_point(aes(size = Factor), alpha = 0.3) +
  theme_cepal()
```

# Scaterplot

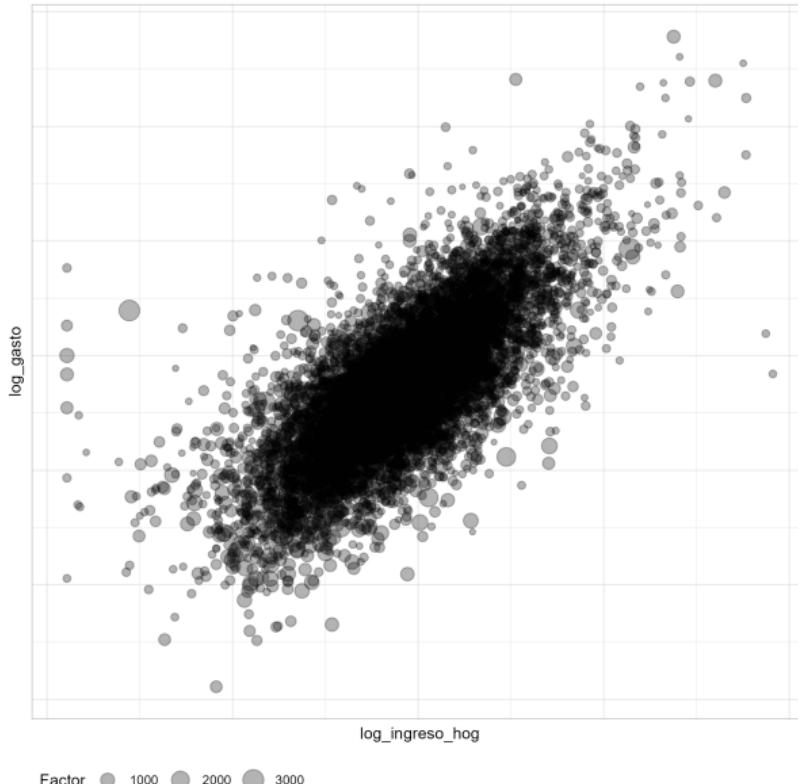


Figura 20: Diagramas de dispersión del ingreso frente al gasto con pesos

## Scaterplot

Otra forma de usar la variable Factor, es asignar la intensidad del color según el valor de la variable.

```
plot21_Ponde <-
  ggplot(
    diseno$variables,
    aes( y = log_gasto, x = log_ingreso_hog)
  ) +
  geom_point(aes(col = Factor), alpha = 0.3) +
  theme_cepal()
```

# Scaterplot

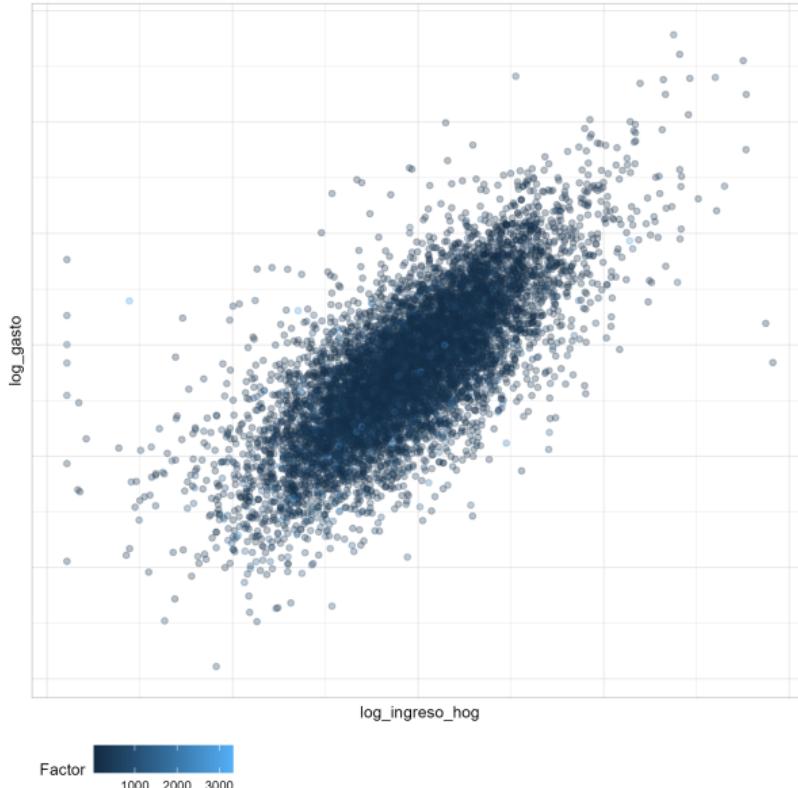


Figura 21: Diagramas de dispersión del ingreso frente al gasto con pesos

## Scaterplot

Podemos extender las bondades de los gráfico de ggplot2 para obtener mayor información de las muestra. Por ejemplo, agrupar los datos por área, para lograr esto se introduce el parámetro `shape`.

```
plot22_Ponde <-
  ggplot(
    diseno$variables,
    aes(
      y = log_gasto, x = log_ingreso_hog,
      shape = Area) # Formas por zona
  ) + geom_point(aes(
    size = Factor, color = Area
  ), alpha = 0.3) +
  labs(size = "Peso") +
  scale_color_manual(values = colorArea) +
  theme_cepal()
```

# Scaterplot

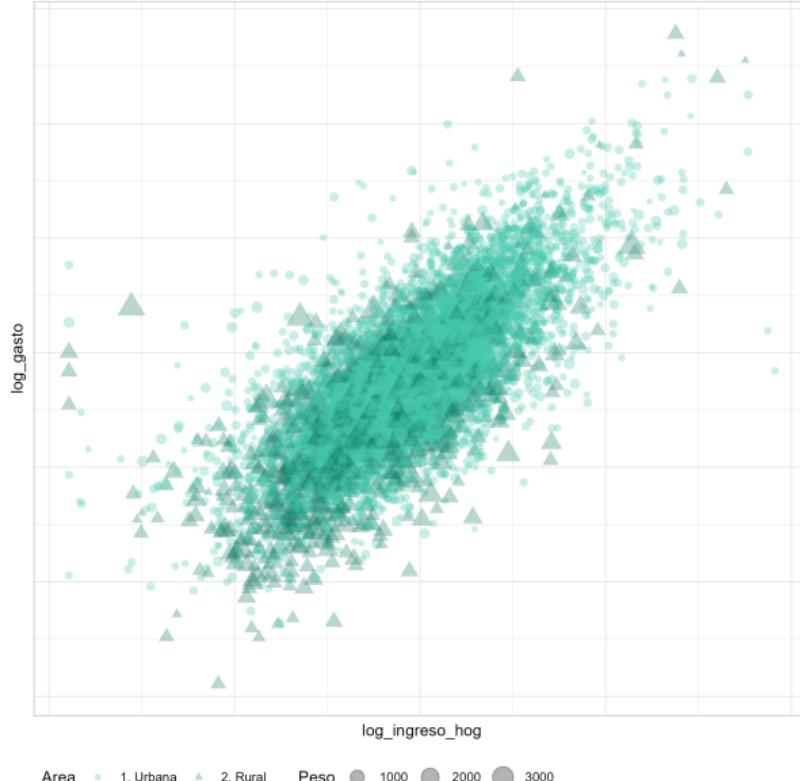


Figura 22: Diagramas de dispersión del ingreso frente al gasto con pesos por zona

## Scaterplot

De forma similar podemos obtener el resultado por tenencia de vehículo

```
plot23_Ponde <-
  ggplot( diseno$variables,
  aes(
    y = log_gasto, x = log_ingreso_hog,
    shape = TIENEVEHICULOS )) +
  geom_point(aes( size = Factor,
    color = TIENEVEHICULOS), alpha = 0.3 ) +
  labs(size = "Peso") +
  scale_color_manual(values = colorVehiculo) +
  theme_cepal()
```

# Scaterplot

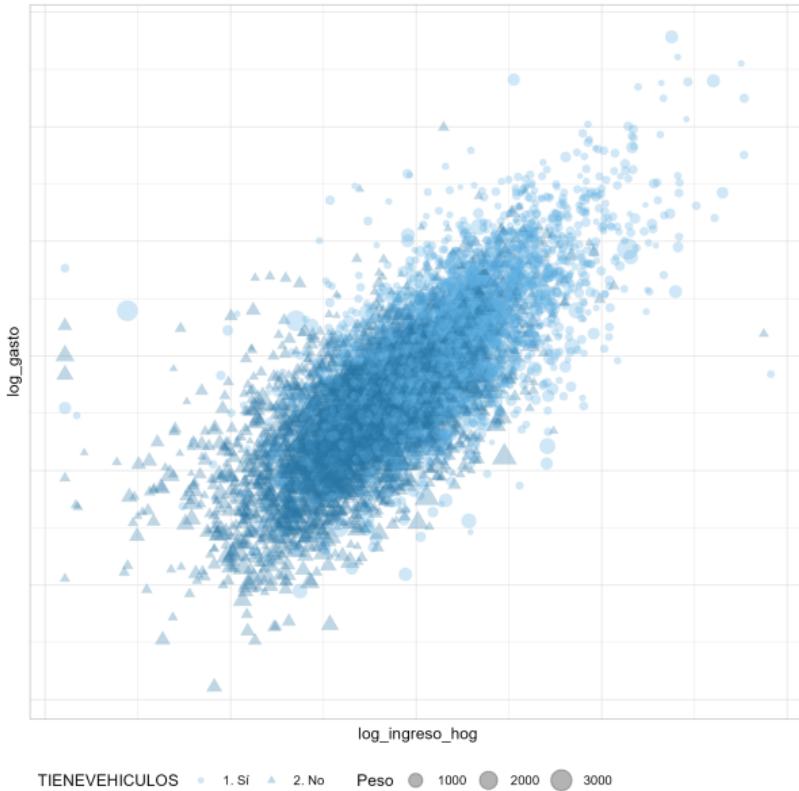


Figura 23: Diagramas de dispersión del ingreso frente al gasto con pesos por tenencia de vehículo

## Scaterplot

Un resultado equivalente se obtiene por región.

```
plot24_Ponde <-
  ggplot(diseno$variables,
  aes( y = log_gasto, x = log_ingreso_hog, shape = TIPOVIVIENDA ) ) +
  geom_point(aes( size = Factor, color = TIPOVIVIENDA ),
  alpha = 0.3 ) +    labs(size = "Peso") +
  scale_color_manual(values = colorvivienda) +
  theme_cepal()
```

# Scaterplot

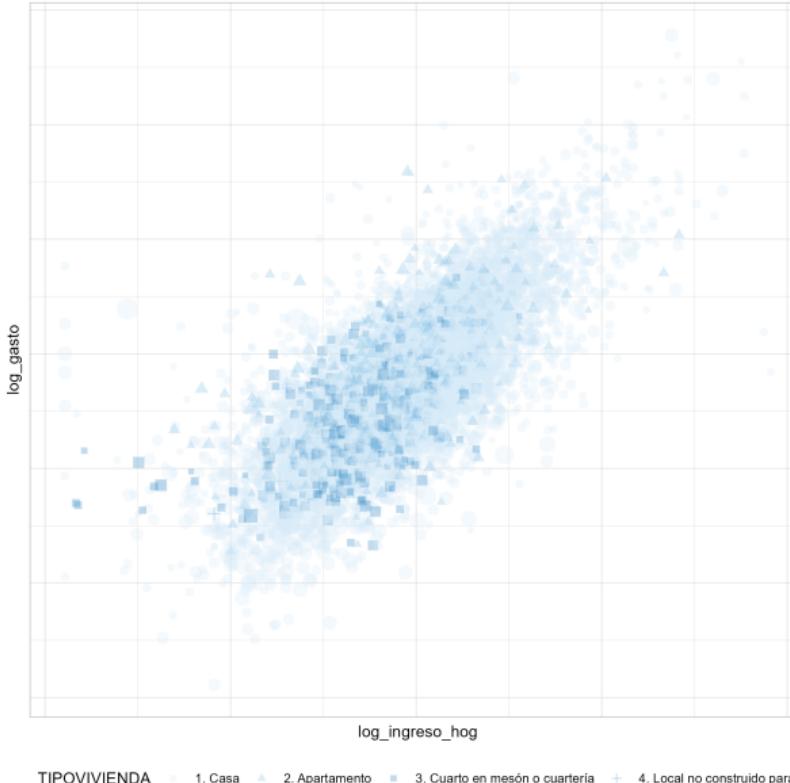


Figura 24: Diagramas de dispersión del ingreso frente al gasto con pesos por tenencia de vehículo

Diagrama de barras para variables categoricas

## Diagrama de barras

Para realizar estos gráfico un primer paso es realizar las estimaciones puntuales.

```
(tamano_area <- diseno %>%
  group_by(Area) %>%
  summarise(
    Nd = survey_total(vartype = c("se", "ci"))
  ))
```

Area	Nd	Nd_se	Nd_low	Nd_upp
1. Urbana	1508007	17896	1472874	1543141
2. Rural	1110294	42796	1026275	1194313

## Diagrama de barras

```
plot25_Ponde <- ggplot(  
  data = tamano_area,           # Fuente de los datos  
  aes(x = Area,                # Valores en el eje x  
      y = Nd,                  # Altura de la barras  
      ymax = Nd_upp,           # Limite superior del IC  
      ymin = Nd_low,           # Limite inferior del IC  
      fill = Area)             # Color del relleno  
) + geom_bar( stat = "identity",# Valor incluido en la tabla  
  position = "dodge") +  
geom_errorbar(      # Gráfica del IC.  
  position = position_dodge(width = 0.9),  
  width = 0.3  
) + theme_bw()
```

## Diagrama de barras

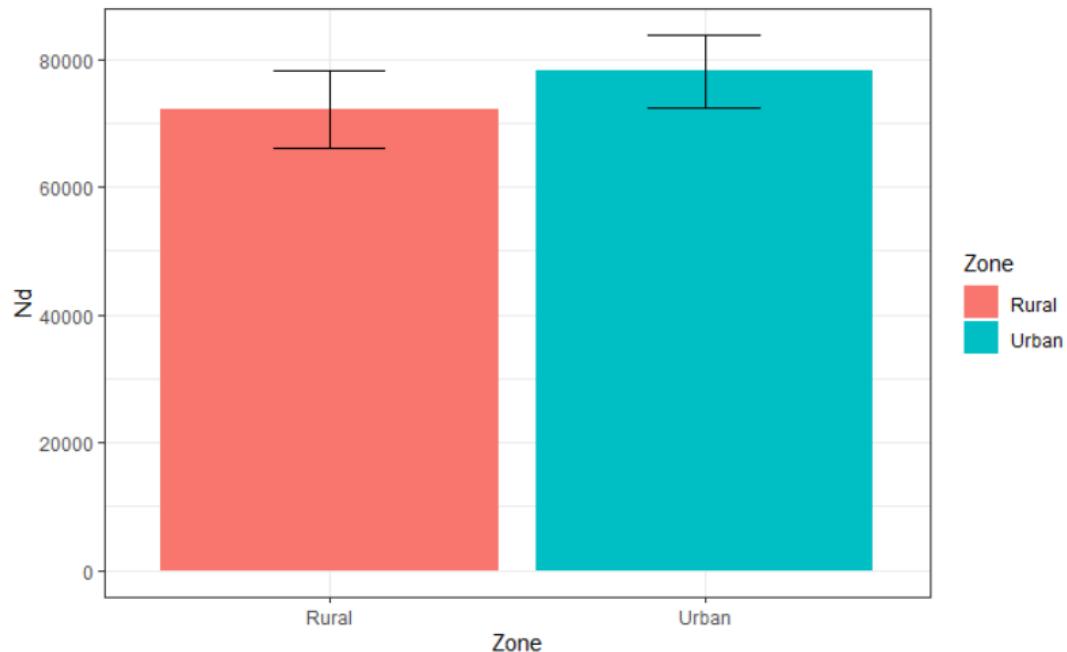


Figura 25: Diagramas de barras total de personas estimado por área

## Diagrama de barras

Como se ha visto en los gráficos anteriores podemos extender a muchas categorías.

```
(tamano_pobreza <- diseno %>%
  group_by(pobreza_LP = as.character(pobreza_LP)) %>%
  summarise(
    Nd = survey_total(vartype = c("se", "ci"))
  ))
```

pobreza_LP	Nd	Nd_se	Nd_low	Nd_upp
0	2346676	45883	2256598	2436754
1	271626	10855	250316	292936

## Diagrama de barras

El gráfico se obtiene con una sintaxis homologa a la anterior.

```
plot26_Ponde <- ggplot(
  data = tamano_pobreza,
  aes( x = pobreza_LP, y = Nd,
       ymax = Nd_upp, ymin = Nd_low,
       fill = pobreza_LP ) ) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar(
    position = position_dodge(width = 0.9),
    width = 0.3
  ) + theme_bw()
```

## Diagrama de barras

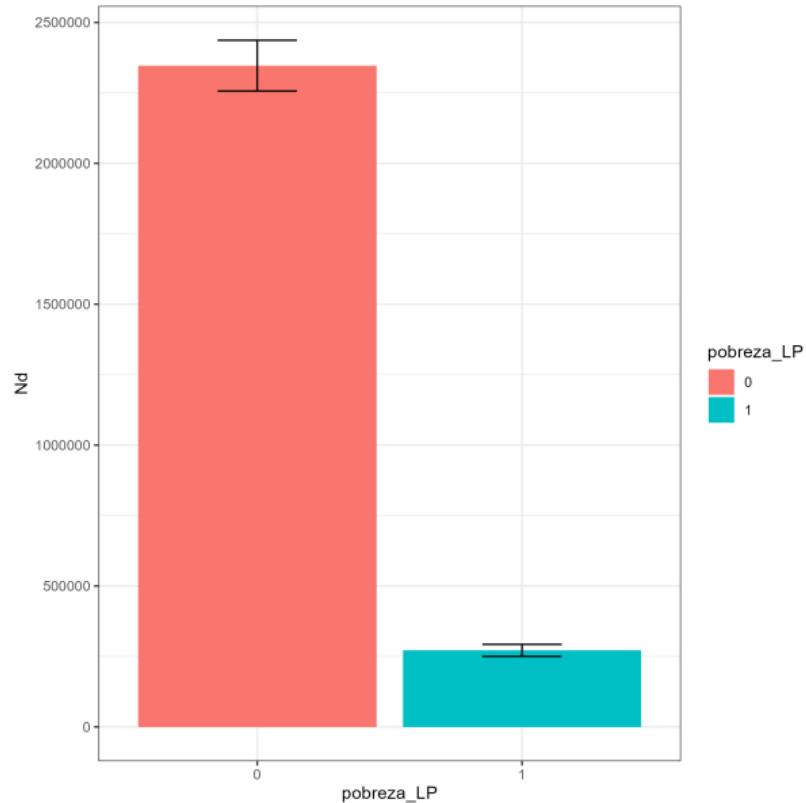


Figura 26: Diagramas de barras del total de personas condición de pobreza estimado

## Diagrama de barras

De forma similar a los gráficos de Caja es posible realizar comparaciones entre más dos variables.

```
tamano_vehiculo_pobreza <- diseno %>%
  group_by(TIENEVEHICULOS ,
           pobreza_LP = as.character(pobreza_LP)) %>%
  summarise(
    Nd = survey_total(vartype = c("se", "ci"))
  ) %>%   as.data.frame()
tamano_vehiculo_pobreza
```

## Diagrama de barras

El gráfico para la tabla anterior queda de la siguiente forma.

```
plot27_Ponde <-
  ggplot(
    data = tamano_vehiculo_pobreza,
    aes( x = pobreza_LP, y = Nd,
        ymax = Nd_upp, ymin = Nd_low,
        fill = as.factor(TIENEVEHICULOS) ) ) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar( position = position_dodge(width = 0.9),
    width = 0.3 ) + theme_bw() + labs(fill = "Tiene\nvehículo")
plot27_Ponde
```

## Diagrama de barras

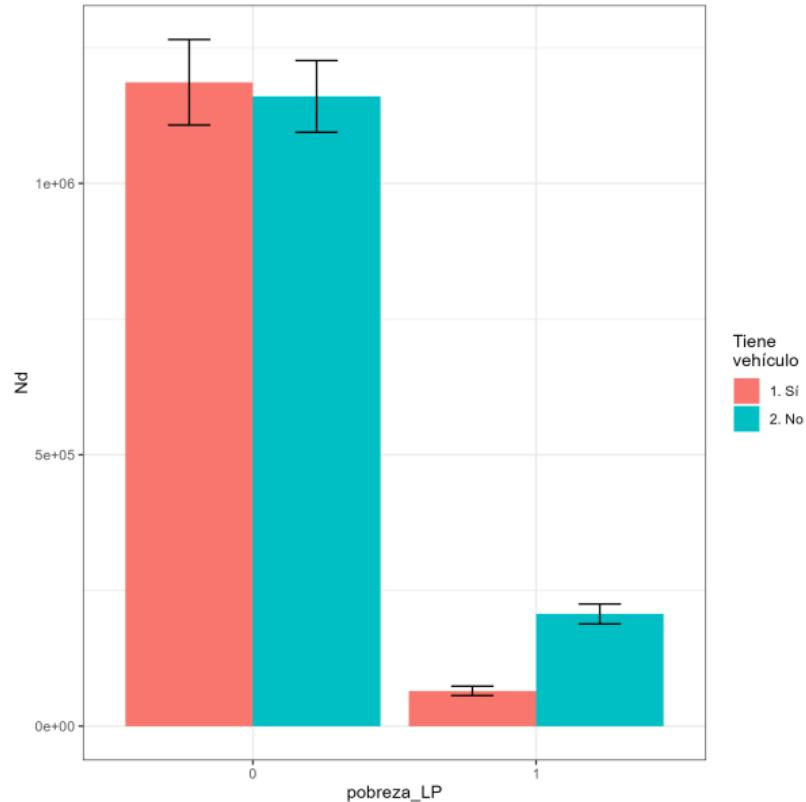


Figura 27: Diagramas de barras del total de personas condición de pobreza y tenencia de vehículo

## Diagrama de barras

En estos gráficos podemos presentar proporciones por variables.

```
(prop_vehiculo_Pobreza <- sub_Urbano %>%
  group_by(TIENEVEHICULOS,
           pobreza_LP = as.character(pobreza_LP) ) %>%
summarise(
  prop = survey_prop(
    vartype = c("se", "ci")
  )
) %>%
data.frame())
```

## Diagrama de barras

Después de tener la tabla con los valores a presentar el gráfico se realiza con la siguiente sintaxis.

```
plot28_Ponde <- ggplot(  
  data = prop_vehiculo_Pobreza,  
  aes(  
    x = pobreza_LP, y = prop,  
    ymax = prop_upp, ymin = prop_low,  
    fill = TIENEVEHICULOS  
) +  
  geom_bar(stat = "identity", position = "dodge") +  
  geom_errorbar(  
    position = position_dodge(width = 0.9),  
    width = 0.3  
) + scale_fill_manual(values = colorVehiculo) +  
  theme_bw()
```

## Diagrama de barras

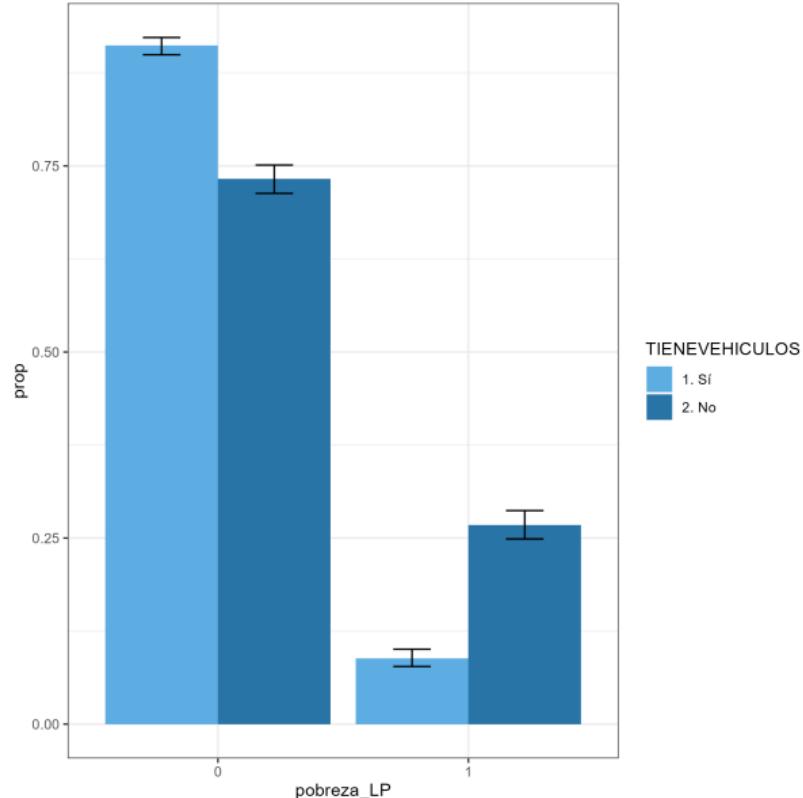


Figura 28: Diagramas de barras del total de personas condición de pobreza y tenencia de vehículo

## Diagrama de barras

Proporción de hogares con vehículo condicionado al tipo de vivienda en el área rural.

```
prop_vehiculo_vivienda <- sub_Rural %>%
  group_by(TIPOVIVIENDA , TIENEVEHICULOS) %>%
  summarise(
    prop = survey_prop(vartype = c("se", "ci")))
  ) %>%
  data.frame()
```

## Diagrama de barras

```
plot29_Ponde <- ggplot(data = prop_vehiculo_vivienda,
aes(
  x = TIPOVIVIENDA,
  y = prop,
  ymax = prop_upp,
  ymin = prop_low,
  fill = as.factor(TIENEVEHICULOS)
)) +
geom_bar(stat = "identity", position = "dodge") +
geom_errorbar(position = position_dodge(width = 0.9),
width = 0.3) + labs(fill = "Tiene vehículo") +
scale_fill_manual(values = colorVehiculo) +
theme_bw()
```

## Diagrama de barras

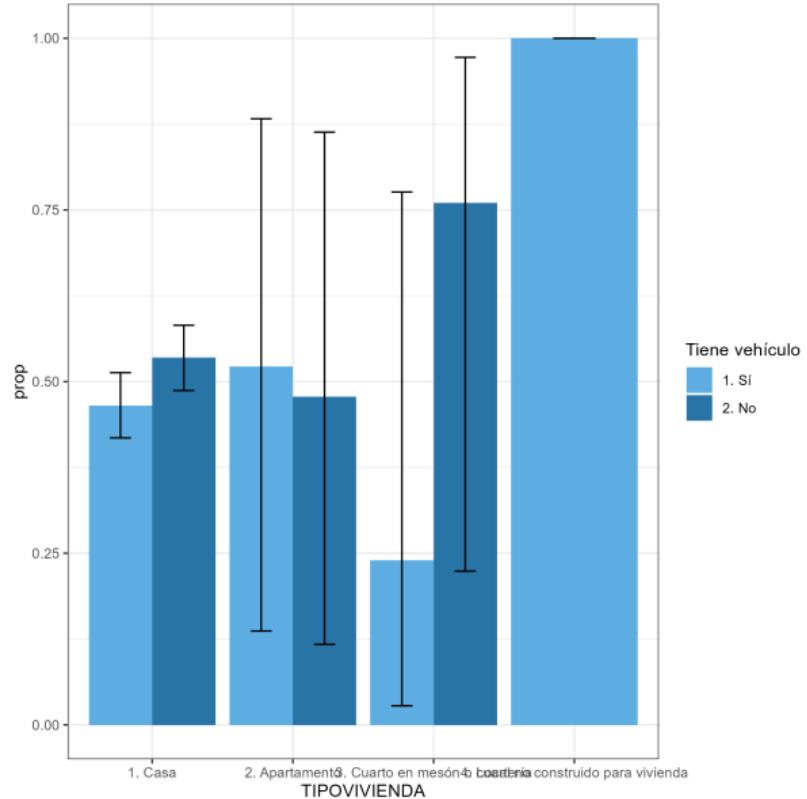


Figura 29: Diagramas de barras del total de hogares con vehículos y el tipo de vivienda

Creando mapas

# Introducción

- ▶ Los mapas son herramientas poderosas para visualizar datos, especialmente para indicadores sociales y demográficos.
- ▶ Para crear mapas en R, se requiere información geoespacial que contenga coordenadas o delimitaciones geográficas.
- ▶ Sitios web como <http://www.diva-gis.org/gdata> ofrecen bases de datos gratuitas con vectores geográficos.
- ▶ Estos conjuntos de datos contienen observaciones de longitud y latitud que permiten representar puntos y polígonos en un mapa.
- ▶ En R, existen varias bibliotecas para crear mapas, incluyendo `tmap` y `ggplot2`.

## Mapas con tmap.

Para realizar el mapa hay que contar con el archivo de *shepefile*

# Mapas con tmap.

El mapa resultante es:



Figura 30: Mapa por departamento

## Estimación de la pobreza por departamento

```
diseno <- diseno %>%
  mutate(
    dam = haven::as_factor(F1_A0_DEPARTAMENTO, levels = "values"),
    dam = stringi::stri_pad(str = dam, pad = "0", width = 2)
  )
prop_dam_pobreza <- diseno %>% group_by(dam) %>%
  summarise(prop = survey_mean(pobreza_LP, vartype = c("se"))) %>%
  data.frame()
prop_dam_pobreza %>% head(10)
```

## Tabla de estimación de la pobreza por departamento

dam	prop	prop_se
01	0.1338	0.0251
02	0.1183	0.0265
03	0.1078	0.0251
04	0.0935	0.0343
05	0.1112	0.0119
06	0.0464	0.0113
07	0.0600	0.0160
08	0.1301	0.0162
09	0.2262	0.0312
10	0.0696	0.0372

## Mapas con tmap.

```
brks <- c(0, .05, 0.1, 0.15, .2, 0.25, 1)
shape_temp <- inner_join(shapePais, prop_dam_pobreza) %>%
  tm_shape()

map2 <- shape_temp + tm_polygons(
  "prop",                      # Nombre de la columna
  breaks = brks,               # Puntos de corte
  title = "Pobreza",           # Titilo del labels.
  palette = "YlOrRd"          # Paleta y dirección de colores
)
```

# Mapas con tmap.

El mapa resultante es:

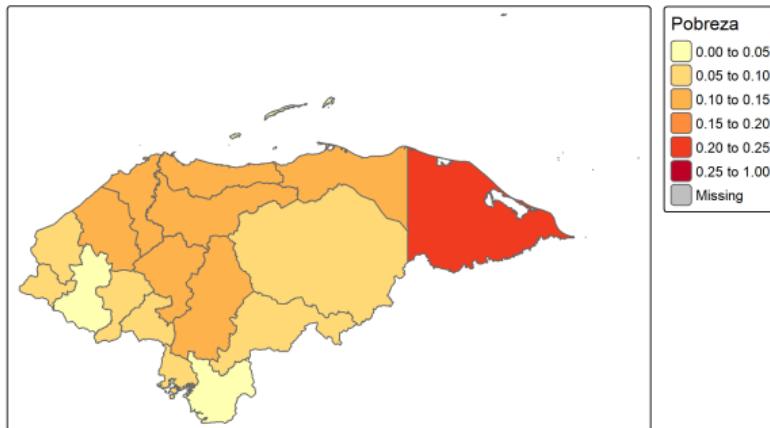


Figura 31: Mapa por departamento de la pobreza

## Estimación del ingreso medio por departamento

```
prom_dam <- svyby(~log_ingreso_hog, ~dam, diseno,
  svymean,
  na.rm = T, covmat = TRUE,
  vartype = c("cv")
) %>% mutate(cv = cv*100)
head(prom_dam)
```

	dam	log_ingreso_hog	cv
01	01	9.763	0.7662
02	02	9.565	0.7157
03	03	9.678	0.9578
04	04	9.831	1.6192
05	05	9.939	0.4026
06	06	9.916	0.8099

## Mapas con tmap.

```
brks <- c(0, 1, 3)
shape_temp <- inner_join(shapePais, prom_dam) %>%
  tm_shape()

map3 <- shape_temp + tm_polygons(
  "cv",
  breaks = brks,
  title = "cv",
  palette = c("#FFFFFF", "#000000"),
) + tm_layout(asp = 0)
```

## Mapas con tmap.

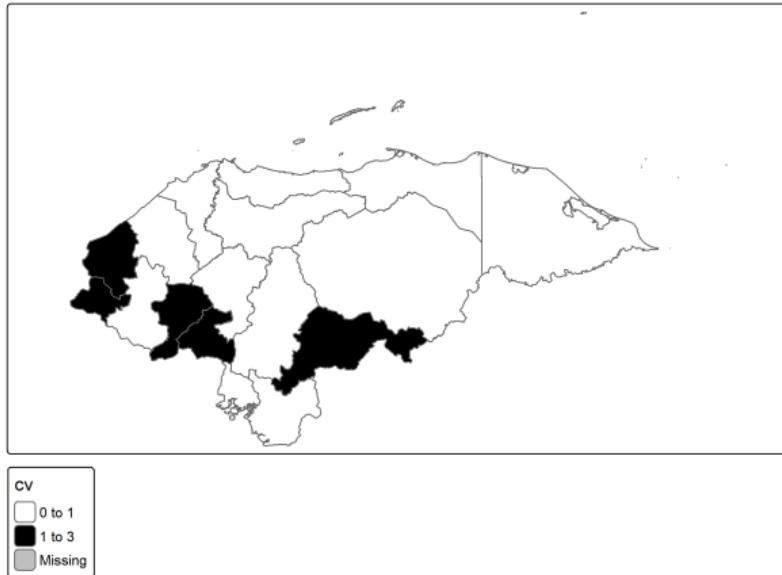


Figura 32: Mapa por departamento del Coeficiente de variación del log. ingreso medio

# Estimación de la pobreza por departamento.

Estimación de la pobreza por departamento y área

```
prom_dam_vehiculo <- diseno %>%
  group_by(dam, Area, TIENEVEHICULOS) %>%
  summarise(prop = survey_mean(pobreza_LP , vartype = "cv")) %>%
  filter(TIENEVEHICULOS == "1. Sí", Area == "1. Urbana")
data.frame(prom_dam_vehiculo) %>% head()
```

dam	Area	TIENEVEHICULOS	prop	prop_cv
01	1. Urbana	1. Sí	0.1094	0.2353
02	1. Urbana	1. Sí	0.1422	0.2303
03	1. Urbana	1. Sí	0.1331	0.1994
04	1. Urbana	1. Sí	0.1183	0.2734
05	1. Urbana	1. Sí	0.0387	0.2480
06	1. Urbana	1. Sí	0.0782	0.3025

## Mapas con tmap.

```
shape_temp <- inner_join(shapePais, prom_dam_vehiculo) %>%
  tm_shape()

map4 <- shape_temp + tm_polygons(
  "prop",
  title = "Pobreza",
) + tm_layout(asp = 0)
```

## Mapas con tmap.

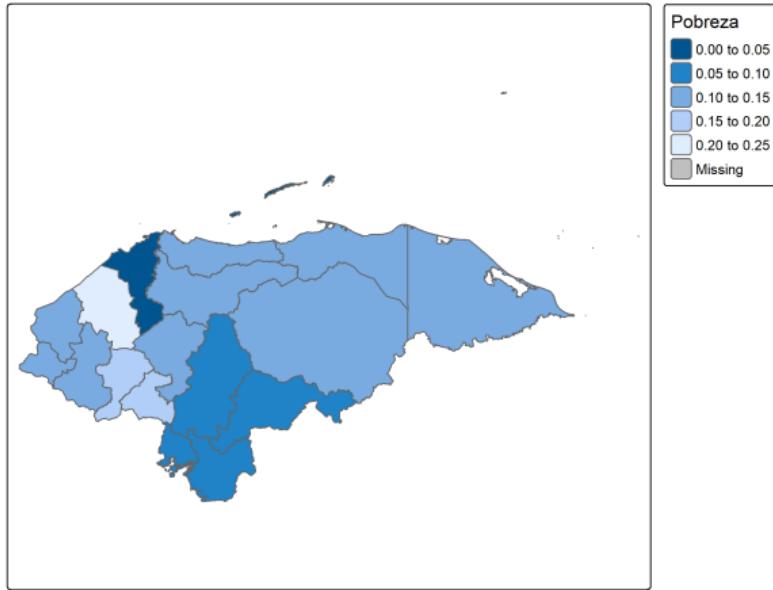


Figura 33: Mapa por departamento de la pobreza monetaria

## Mapas con tmap.

```
map5 <- shape_temp + tm_polygons(  
  "prop_cv",  
  title = "cv",  
  palette = c("#FFFFFF", "#000000"),  
  breaks = c(0, 0.2, 1)  
) + tm_layout(asp = 0)
```

## Mapas con tmap.

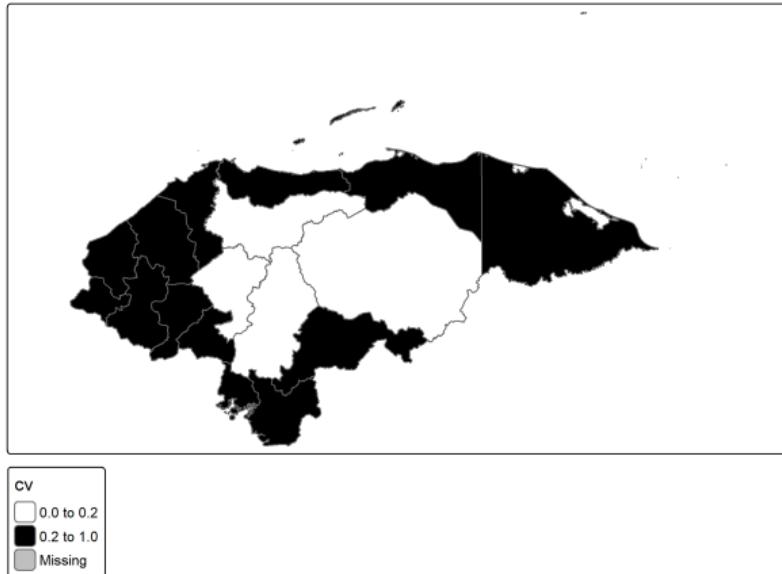


Figura 34: Mapa por departamento del CV para la estimación de la pobreza monetaria

¡Gracias!

*Email:* andres.gutierrez@cepal.org