Procedimiento y análisis de datos

Amparo Galvez Vilar

2025-02-20

Contents

1 CARGA DE LIBRERÍAS **1**

2 CARGA DE DATOS E INSPECCION INICIAL **2**

3 RECODIFICAR LAS VARIABLES **3**

4 LIMPIEZA DE DATOS **4**
4.1 Distribución variables sociodemográficas . . . 7
4.2 Equilibrio Escenario . . . 11
4.3 Tasa global Showrooming . . . 12
4.4 Distribución Escalas Likert . . . 19

5 PROPIEDADES PSICOMETRICAS **20**
5.1 Realismo escenario . . . 20
5.2 Chequeo de confianza percibida . . . 22
5.3 Chequeo de servicio percibido . . . 22
5.4 Análisis Factorial Confirmatorio y propiedades psiconometricas . . . 23

6 ANÁLISIS EXPLORATORIO **27**
6.1 Estadísticos descriptivos y matriz de correlaciones . . . 27

7 MODELO DE REGRESIÓN LOGÍSTICA **34**
7.1 Modelo inicial: sólo las cinco variables clave . . . 34
7.2 Ampliar con variables sociodemográficas . . . 35
7.3 Probar interacciones entre variables clave y demográficas . . . 36
7.4 Probar interacciones entre variables clave . . . 37
7.5 Modelo Final . . . 39

8 BONDAD DEL AJUSTE DEL MODELO FINAL **40**

9 INTERPRETACIÓN DE RESULTADOS **42**
9.1 Global . . . 42
9.2 Orientación al precio x Grupo de edad . . . 44
9.3 Gratificación x Tamaño de municipio . . . 45

1 CARGA DE LIBRERÍAS

```
library(car)
library(dplyr)
library(emmeans)
```

```
library(GGally)
library(gmodels)
library(grid)
library(lavaan)
library(likert)
library(pROC)
library(psych)
library(RColorBrewer)
library(readr)
library(reshape2)
library(ResourceSelection)
library(scales)
library(semTools)
library(tibble)
library(tidyverse)
```

2 CARGA DE DATOS E INSPECCION INICIAL

```
df <- read_delim("~/uni/CUARTO/segundi_cuatri/TFG/data/Datos_Amparo_FINAL_COPIA.csv",
    delim = ";", escape_double = FALSE, trim_ws = TRUE)
```

```
glimpse(df)   # Muestra tipos de cada columna y primeros valores
```

```
## Rows: 720
## Columns: 24
## $ REGISTRO   <dbl> 77, 79, 106, 109, 112, 115, 117, 119, 120, 130, 132, 133, 1~
## $ DURACION   <dbl> 379, 347, 220, 524, 337, 368, 551, 284, 562, 468, 248, 203,~
## $ ESTADO     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ S2         <dbl> 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 2, 2, 2, 1, 2, 2,~
## $ AGE        <dbl> 57, 47, 64, 43, 48, 55, 62, 52, 64, 56, 61, 61, 62, 56, 55,~
## $ CUOTA_EDAD <dbl> 3, 2, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
## $ S3         <dbl> 7, 7, 7, 6, 7, 7, 6, 5, 7, 7, 7, 7, 7, 6, 7, 7, 7, 7, 7, 7,~
## $ P1         <dbl> 3, 4, 4, 4, 3, 3, 4, 4, 4, 2, 3, 2, 4, 3, 3, 3, 4, 4, 4, 3,~
## $ P2         <dbl> 2, 3, 4, 4, 2, 2, 4, 4, 4, 2, 4, 2, 4, 2, 3, 2, 3, 4, 2, 2,~
## $ ESCENARIO  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ P3         <dbl> 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,~
## $ P4         <dbl> 7, 5, 5, 5, 5, 7, 7, 6, 7, 7, 6, 5, 6, 4, 4, 6, 5, 4, 7, 7,~
## $ P5_1       <dbl> 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2,~
## $ P5_2       <dbl> 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2,~
## $ P6_1       <dbl> 6, 4, 5, 5, 5, 5, 4, 5, 5, 6, 5, 6, 4, 5, 5, 5, 5, 6, 3, 3,~
## $ P6_2       <dbl> 2, 3, 4, 4, 3, 3, 2, 4, 4, 6, 5, 6, 5, 4, 5, 4, 4, 5, 2, 2,~
## $ P6_3       <dbl> 6, 5, 5, 4, 4, 5, 2, 5, 6, 6, 7, 6, 5, 4, 5, 6, 5, 6, 6, 3,~
## $ P7_1       <dbl> 7, 6, 5, 4, 6, 6, 4, 6, 6, 6, 7, 5, 7, 7, 5, 5, 5, 7, 5, 7,~
## $ P7_2       <dbl> 5, 5, 4, 3, 5, 6, 4, 6, 6, 5, 7, 5, 7, 7, 4, 5, 5, 6, 3, 7,~
## $ P7_3       <dbl> 3, 5, 4, 5, 5, 5, 5, 6, 5, 6, 7, 4, 7, 5, 6, 5, 5, 6, 5, 7,~
## $ P7_4       <dbl> 4, 5, 4, 6, 4, 6, 4, 6, 4, 6, 7, 5, 7, 6, 6, 5, 4, 6, 6, 7,~
## $ P8_1       <dbl> 1, 1, 4, 4, 3, 2, 3, 4, 4, 2, 3, 4, 4, 4, 3, 5, 4, 5, 5, 1,~
## $ P8_2       <dbl> 1, 1, 4, 4, 2, 4, 3, 5, 4, 2, 3, 4, 6, 4, 6, 5, 5, 3, 4, 3,~
## $ P8_3       <dbl> 1, 1, 4, 2, 3, 4, 3, 5, 4, 2, 3, 4, 5, 4, 6, 5, 5, 3, 5, 3,~
```

```
summary(df)   # Estadísticos básicos de cada variable
```

```
##     REGISTRO         DURACION         ESTADO           S2           AGE
```

```
##  Min.   :  77.0   Min.   :180.0   Min.   :1.000   Min.   :1.0   Min.   :18.00
##  1st Qu.: 306.8   1st Qu.:216.8   1st Qu.:1.000   1st Qu.:1.0   1st Qu.:31.00
##  Median : 567.5   Median :274.5   Median :1.000   Median :1.5   Median :43.00
##  Mean   : 781.5   Mean   :307.7   Mean   :1.144   Mean   :1.5   Mean   :42.64
##  3rd Qu.:1247.2   3rd Qu.:358.0   3rd Qu.:1.000   3rd Qu.:2.0   3rd Qu.:55.00
##  Max.   :1880.0   Max.   :926.0   Max.   :5.000   Max.   :2.0   Max.   :65.00
##    CUOTA_EDAD          S3              P1              P2          ESCENARIO
##  Min.   :1.000   Min.   :5.000   Min.   :2.000   Min.   :2.000   Min.   :1.00
##  1st Qu.:2.000   1st Qu.:6.000   1st Qu.:3.000   1st Qu.:2.000   1st Qu.:1.75
##  Median :2.000   Median :7.000   Median :3.000   Median :3.000   Median :2.50
##  Mean   :2.122   Mean   :6.675   Mean   :3.072   Mean   :2.678   Mean   :2.50
##  3rd Qu.:3.000   3rd Qu.:7.000   3rd Qu.:4.000   3rd Qu.:3.000   3rd Qu.:3.25
##  Max.   :3.000   Max.   :7.000   Max.   :4.000   Max.   :4.000   Max.   :4.00
##        P3              P4            P5_1            P5_2            P6_1
##  Min.   :1.000   Min.   :1.00   Min.   :1.000   Min.   :1.000   Min.   :1.00
##  1st Qu.:1.000   1st Qu.:5.00   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:5.00
##  Median :1.000   Median :6.00   Median :1.000   Median :1.000   Median :5.00
##  Mean   :1.401   Mean   :5.86   Mean   :1.449   Mean   :1.446   Mean   :5.25
##  3rd Qu.:2.000   3rd Qu.:7.00   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:6.00
##  Max.   :2.000   Max.   :7.00   Max.   :2.000   Max.   :2.000   Max.   :7.00
##       P6_2            P6_3            P7_1            P7_2
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:3.000   1st Qu.:4.000   1st Qu.:5.000   1st Qu.:4.750
##  Median :4.000   Median :5.000   Median :6.000   Median :5.000
##  Mean   :4.322   Mean   :5.306   Mean   :5.806   Mean   :5.286
##  3rd Qu.:5.000   3rd Qu.:7.000   3rd Qu.:7.000   3rd Qu.:6.000
##  Max.   :7.000   Max.   :7.000   Max.   :7.000   Max.   :7.000
##       P7_3            P7_4            P8_1            P8_2
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:4.750   1st Qu.:4.000   1st Qu.:2.000   1st Qu.:2.000
##  Median :5.000   Median :5.000   Median :4.000   Median :4.000
##  Mean   :5.258   Mean   :5.319   Mean   :3.375   Mean   :3.431
##  3rd Qu.:6.000   3rd Qu.:6.000   3rd Qu.:5.000   3rd Qu.:5.000
##  Max.   :7.000   Max.   :7.000   Max.   :7.000   Max.   :7.000
##       P8_3
##  Min.   :1.000
##  1st Qu.:1.750
##  Median :4.000
##  Mean   :3.364
##  3rd Qu.:5.000
##  Max.   :7.000
```

3 RECODIFICAR LAS VARIABLES

```
df <- df %>%
  # 1. Recodificar variables de entorno y chequeos
  mutate(
    S2    = factor(S2,
                   levels = c(1, 2),
                   labels = c("Masculino", "Femenino")),
    CuotaEdad = factor(CUOTA_EDAD,
                   levels = c(1, 2, 3),
                   labels = c("18-30", "31-50", "51-65")),
```

```
    TamPob    = factor(S3,
                       levels = 1:8,
                       labels = c("<5.000", "5.001-10.000", "10.001-25.000",
                                  "25.001-50.000", "50.001-100.000",
                                  "100.001-500.000", ">500.000", "Desconocido")),
    Escenario = factor(ESCENARIO,
                       levels = 1:4,
                       labels = c("BajaConf+Auto", "AltaConf+Auto",
                                  "BajaConf+Asist", "AltaConf+Asist")),
    CompraBin = factor(P3,
                       levels = c(1, 2),
                       labels = c("Tienda", "Online")),
    CheckConf = factor(P5_1,
                       levels = c(1, 2),
                       labels = c("PercibióAltaConf", "PercibióBajaConf")),
    CheckServ = factor(P5_2,
                       levels = c(1, 2),
                       labels = c("PercibióAsist", "PercibióAuto"))
  ) %>%
  # 2. Crear variables derivadas para el análisis
  mutate(
    CONFIANZA = if_else(ESCENARIO %in% c(1, 3), 1, 2),  # 1=Baja, 2=Alta
    SERVICIO  = if_else(ESCENARIO %in% c(1, 2), 1, 2),   # 1=Auto, 2=Asistido

    # Showrooming: 1 = visitó tienda y compró online, 0 = compró en tienda
    SHOWROOMING = if_else(P3 == 2, 1, 0)
  ) %>%
  # 3. Convertir las nuevas variables en factores con etiquetas
  mutate(
    CONFIANZA   = factor(CONFIANZA,
                         levels = c(1, 2),
                         labels = c("BajaConfianza", "AltaConfianza")),
    SERVICIO    = factor(SERVICIO,
                         levels = c(1, 2),
                         labels = c("Autoservicio", "Asistido")),
    SHOWROOMING = factor(SHOWROOMING,
                         levels = c(0, 1),
                         labels = c("No", "Sí"))
  )
```

4 LIMPIEZA DE DATOS

```
# Calcular % de NA en las escalas principales y P3_2
vars_na <- c(paste0("P6_", 1:3),
             paste0("P7_", 1:4),
             paste0("P8_", 1:3)
             )

df <- df %>%
  rowwise() %>%
  mutate(pct_na = sum(is.na(c_across(all_of(vars_na)))) / length(vars_na)) %>%
  ungroup()
```

```r
# Resumen de pct_na
summary(df$pct_na)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

```r
table(cut(df$pct_na, breaks = c(-Inf, 0, .1, 1), labels = c("0%", "0-10%", ">10%")))
```

```
##
##    0% 0-10%  >10%
##   720     0     0
```

```r
# Filtrar
df_step2 <- df %>% filter(pct_na <= 0.10)
cat("Casos tras filtrar >10% NA:", nrow(df_step2),
    " (se eliminaron", nrow(df) - nrow(df_step2), "casos)\n")
```

```
## Casos tras filtrar >10% NA: 720  (se eliminaron 0 casos)
```

```r
# Función para imputar media por variable
imputar_media <- function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x)

df_step3 <- df_step2

for (v in vars_na) {
  mean_v <- mean(df_step3[[v]], na.rm = TRUE)
  df_step3[[v]] <- imputar_media(df_step3[[v]])
  cat("Imputada media en", v, ":", round(mean_v,2), "\n")
}
```

```
## Imputada media en P6_1 : 5.25
## Imputada media en P6_2 : 4.32
## Imputada media en P6_3 : 5.31
## Imputada media en P7_1 : 5.81
## Imputada media en P7_2 : 5.29
## Imputada media en P7_3 : 5.26
## Imputada media en P7_4 : 5.32
## Imputada media en P8_1 : 3.38
## Imputada media en P8_2 : 3.43
## Imputada media en P8_3 : 3.36
```

```r
# Marcar qué filas tienen outliers en cada escala
outlier_flag <- function(x) {
  q <- quantile(x, c(.25, .75))
  iqr <- diff(q)
  x < (q[1] - 1.5 * iqr) | x > (q[2] + 1.5 * iqr)
}

# Para cada participante, contar en cuántas escalas es outlier
df_step3 <- df_step3 %>%
  rowwise() %>%
  mutate(
    n_outliers = sum(sapply(across(all_of(vars_na)), outlier_flag))
  ) %>%
  ungroup()
```

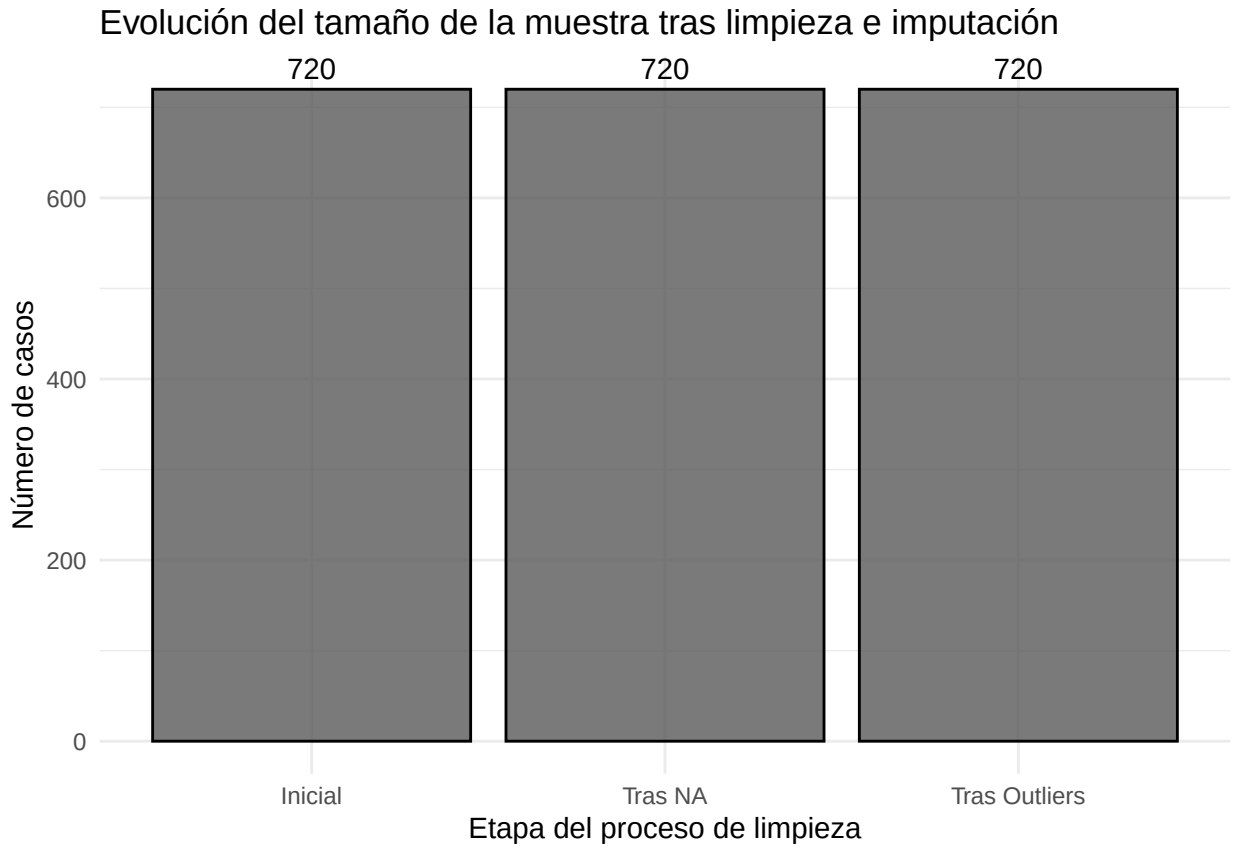
```r
table(df_step3$n_outliers)
```

```
##
##   0
## 720
```

```r
df_clean <- df_step3 %>% filter(n_outliers <= 2)
cat("Casos tras eliminar >2 outliers:", nrow(df_clean),
    " (se eliminaron", nrow(df_step3) - nrow(df_clean), "casos)\n")
```

```
## Casos tras eliminar >2 outliers: 720  (se eliminaron 0 casos)
```

```r
df_summary <- tibble(
  Etapa   = c("Inicial", "Tras NA", "Tras Outliers"),
  Casos   = c(nrow(df), nrow(df_step2), nrow(df_clean))
)

ggplot(df_summary, aes(x = Etapa, y = Casos)) +
  geom_col(color = "black", alpha = 0.8) +
  geom_text(aes(label = Casos), vjust = -0.5) +
  labs(
    x = "Etapa del proceso de limpieza",
    y = "Número de casos",
    title = "Evolución del tamaño de la muestra tras limpieza e imputación"
  ) +
  theme_minimal()
```



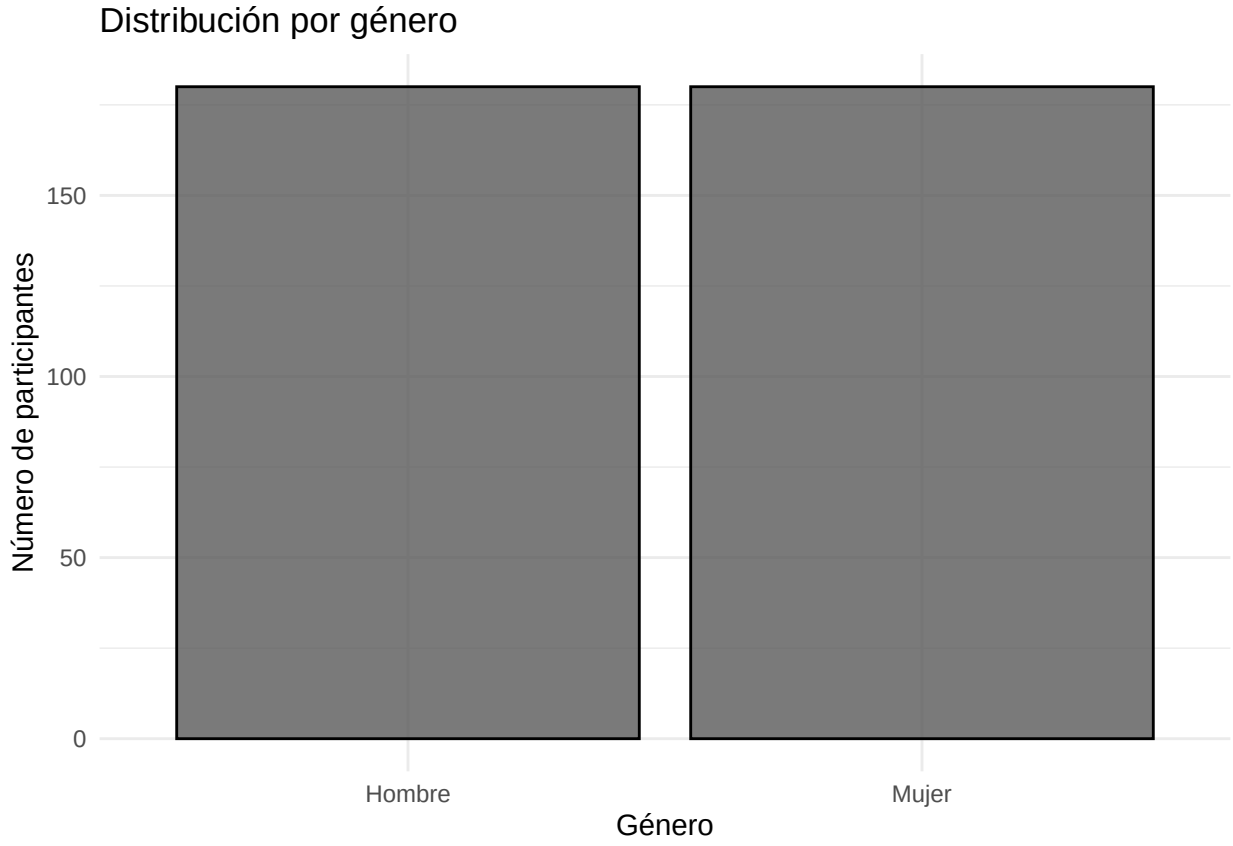
ESTADÍSTICOS DESCRIPTIVOS DE LA MUESTRA

```r
df_unicos <- df %>%
  distinct(REGISTRO, .keep_all = TRUE)
```

4.1 Distribución variables sociodemográficas

4.1.1 Género

```r
# GÉNERO
ggplot(df_unicos, aes(x = factor(S2, labels = c("Hombre","Mujer")))) +
  geom_bar(color = "black", alpha = 0.8) +
  labs(x = "Género", y = "Número de participantes",
       title = "Distribución por género") +
  theme_minimal()
```

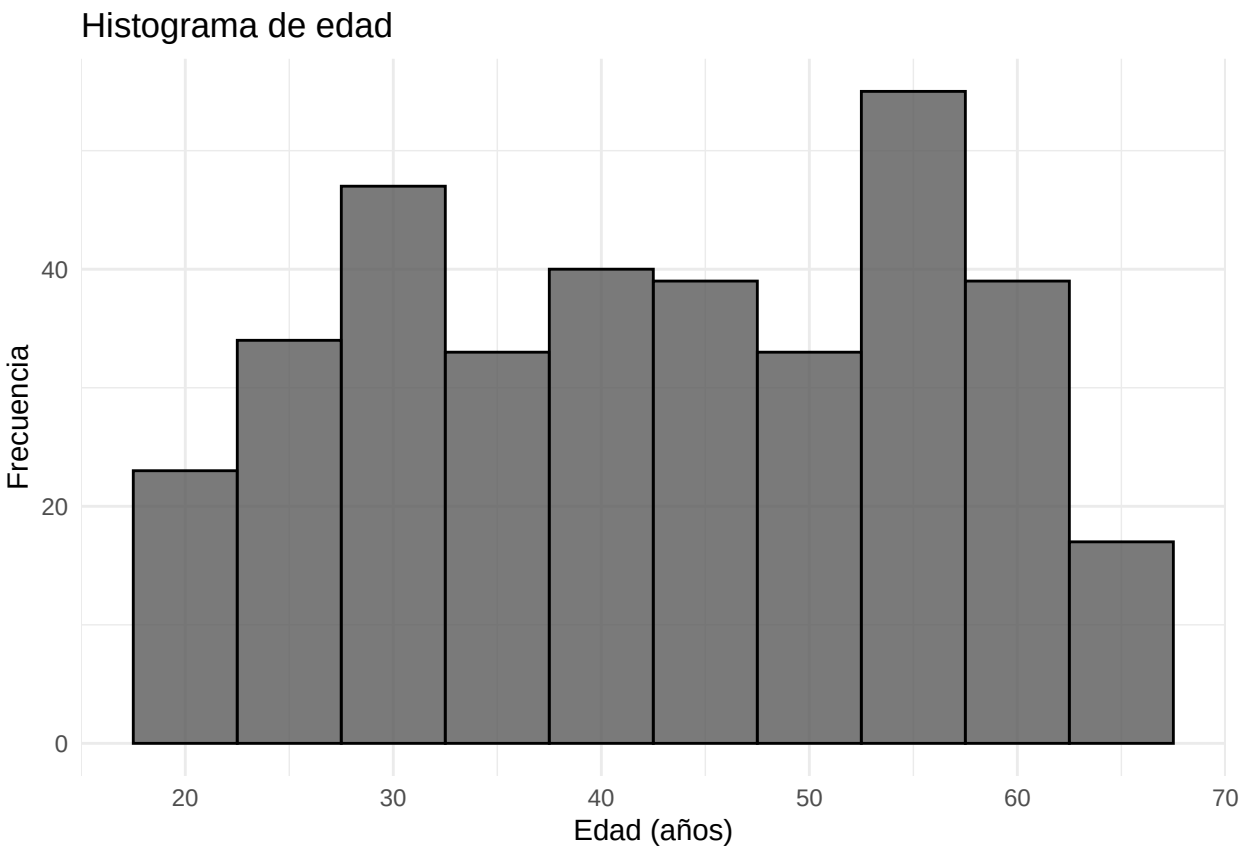


Edad

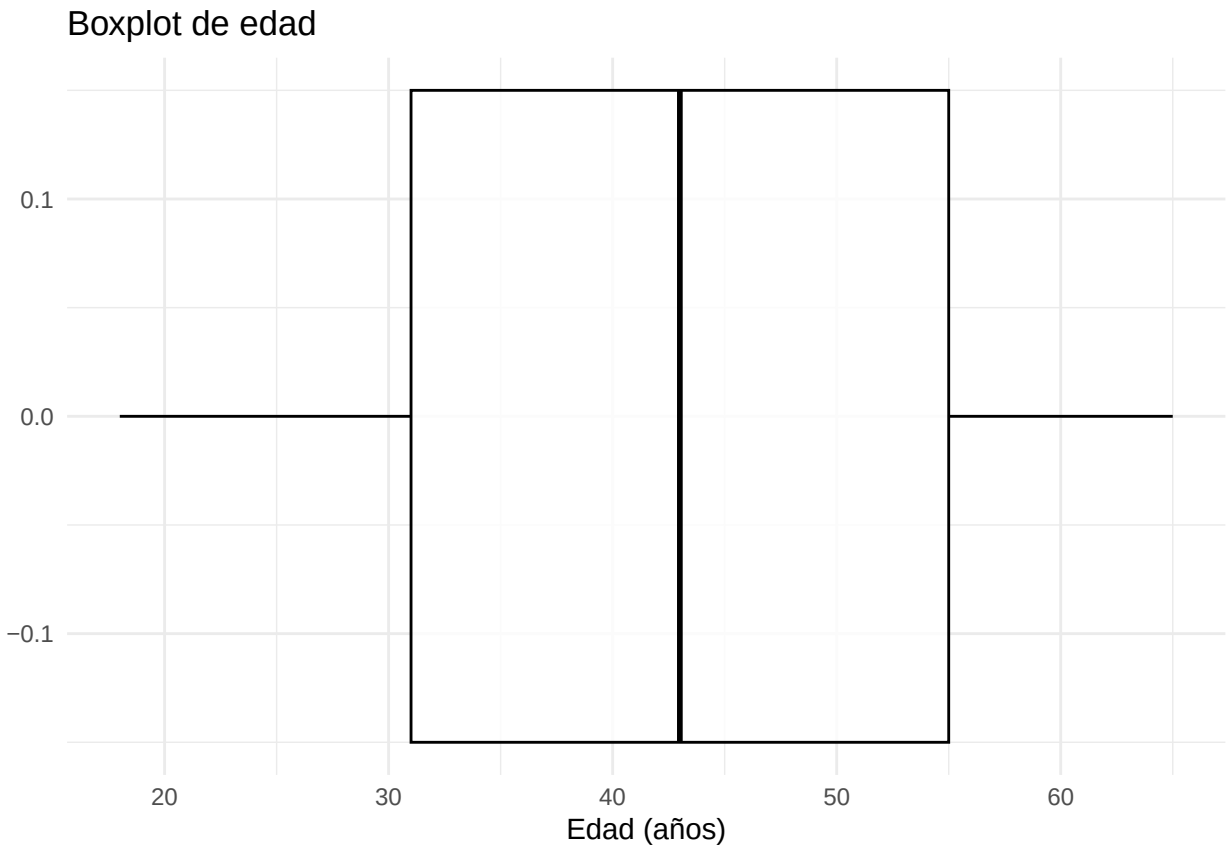
```r
# Estadísticos descriptivos de AGE
edad_stats <- df_unicos %>%
  summarise(
    media               = mean(AGE, na.rm = TRUE),
    mediana             = median(AGE, na.rm = TRUE),
    desviacion_tipica   = sd(AGE, na.rm = TRUE),
    varianza            = var(AGE, na.rm = TRUE),
    min                 = min(AGE, na.rm = TRUE),
    max                 = max(AGE, na.rm = TRUE),
  )
edad_stats
```

```
## # A tibble: 1 x 6
##   media mediana desviacion_tipica varianza   min   max
##   <dbl>   <dbl>             <dbl>    <dbl> <dbl> <dbl>
## 1  42.6      43              13.2     175.    18    65
```

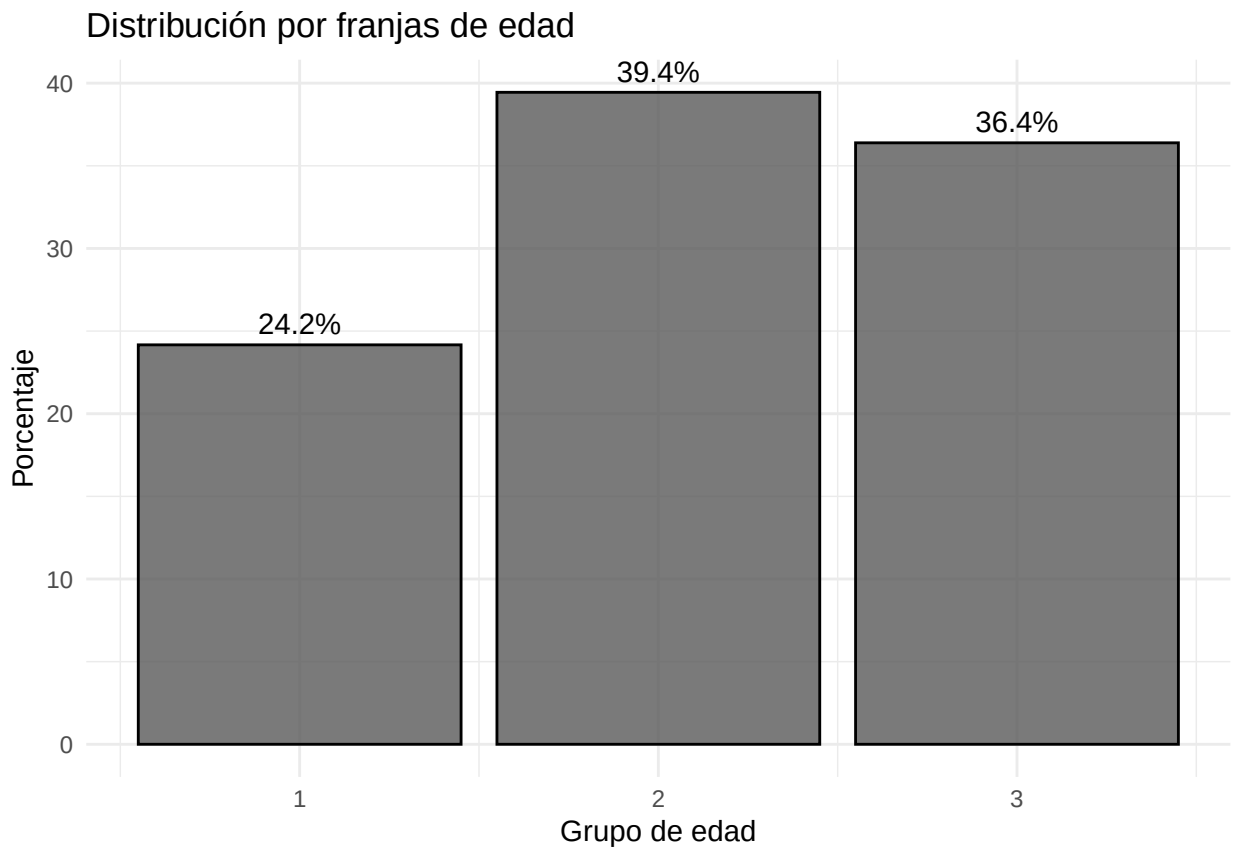
```r
# 1. Histograma y boxplot de AGE
p_hist <- ggplot(df_unicos, aes(x = AGE)) +
  geom_histogram(binwidth = 5, color = "black", alpha = 0.8) +
  labs(x = "Edad (años)", y = "Frecuencia",
       title = "Histograma de edad") +
  theme_minimal()

p_box <- ggplot(df_unicos, aes(x = AGE)) +
  geom_boxplot(width = 0.3, color = "black", alpha = 0.8) +
  labs(x = "Edad (años)",
       title = "Boxplot de edad") +
  theme_minimal()
print(p_hist)
```




```r
print(p_box)
```




```r
# 2. Gráfico de barras de cuotas de edad
cuota_stats <- df_unicos %>%
  count(CUOTA_EDAD) %>%
  mutate(
    pct = n / sum(n) * 100,
    grupo = case_when(
      CUOTA_EDAD == 1 ~ "18-30",
      CUOTA_EDAD == 2 ~ "31-50",
      CUOTA_EDAD == 3 ~ "51-65"
    )
  )

p_bar <- ggplot(cuota_stats, aes(x = CUOTA_EDAD, y = pct)) +
  geom_col(color = "black", alpha = 0.8) +
  geom_text(aes(label = sprintf("%1.1f%%", pct)),
            vjust = -0.5) +
  labs(x = "Grupo de edad", y = "Porcentaje",
       title = "Distribución por franjas de edad") +
  theme_minimal()

print(p_bar)
```

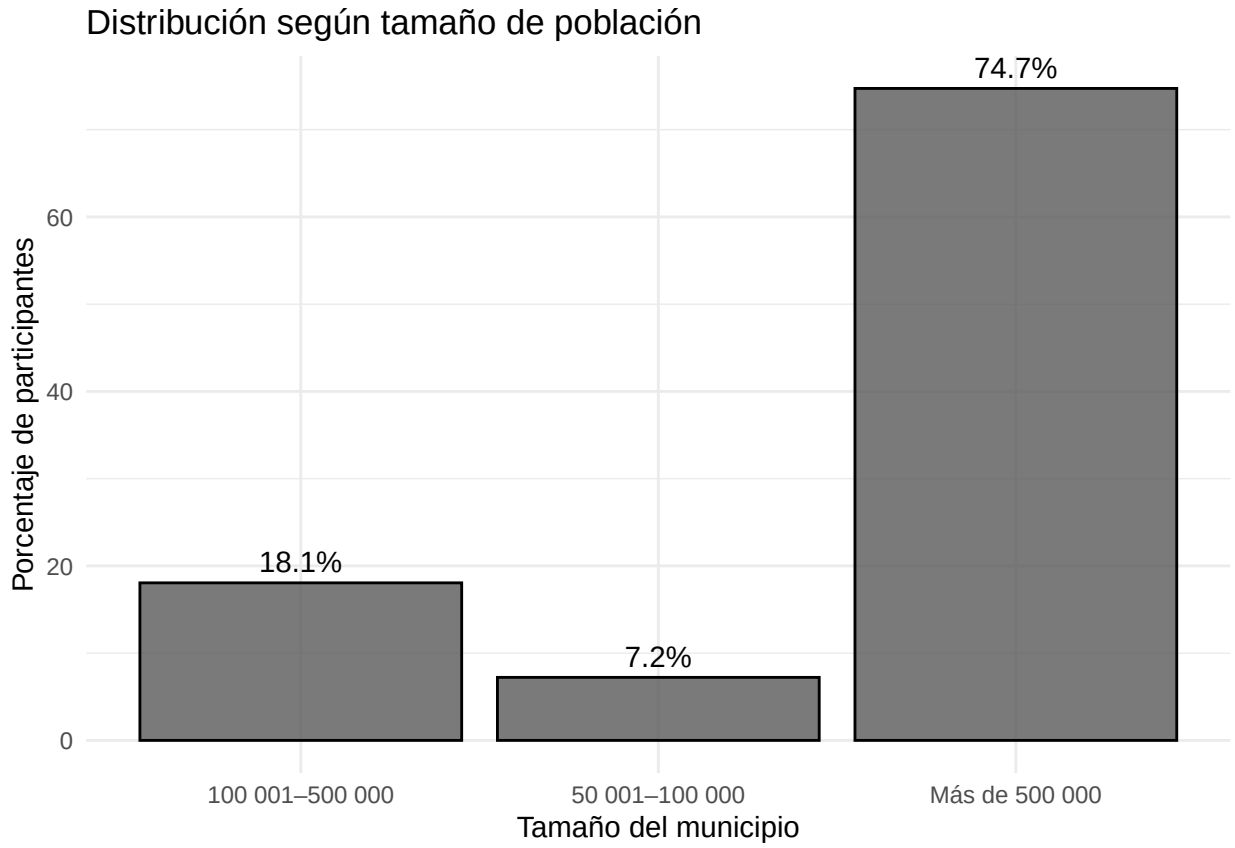


4.1.2 Tamaño municipio

```
# 1. Calcular frecuencias y porcentajes para todas las categorías de S3
pob_stats <- df_unicos %>%
  count(S3) %>%
  mutate(
    grupo = case_when(
      S3 == 1 ~ "Menos de 5 000",
      S3 == 2 ~ "5 001-10 000",
      S3 == 3 ~ "10 001-25 000",
      S3 == 4 ~ "25 001-50 000",
      S3 == 5 ~ "50 001-100 000",
      S3 == 6 ~ "100 001-500 000",
      S3 == 7 ~ "Más de 500 000",
      S3 == 8 ~ "Desconozco"
    ),
    pct = n / sum(n) * 100
  )
print(pob_stats)
```

```
## # A tibble: 3 x 4
##      S3     n grupo             pct
##   <dbl> <int> <chr>           <dbl>
## 1     5    26 50 001-100 000   7.22
## 2     6    65 100 001-500 000 18.1
## 3     7   269 Más de 500 000  74.7
```

```r
# 3. Gráfico de barras para todas las categorías
ggplot(pob_stats, aes(x = grupo, y = pct)) +
  geom_col(color = "black", alpha = 0.8) +
  geom_text(aes(label = sprintf("%1.1f%%", pct)),
            vjust = -0.5)+
  labs(
    x = "Tamaño del municipio",
    y = "Porcentaje de participantes",
    title = "Distribución según tamaño de población"
  ) +
  theme_minimal()
```



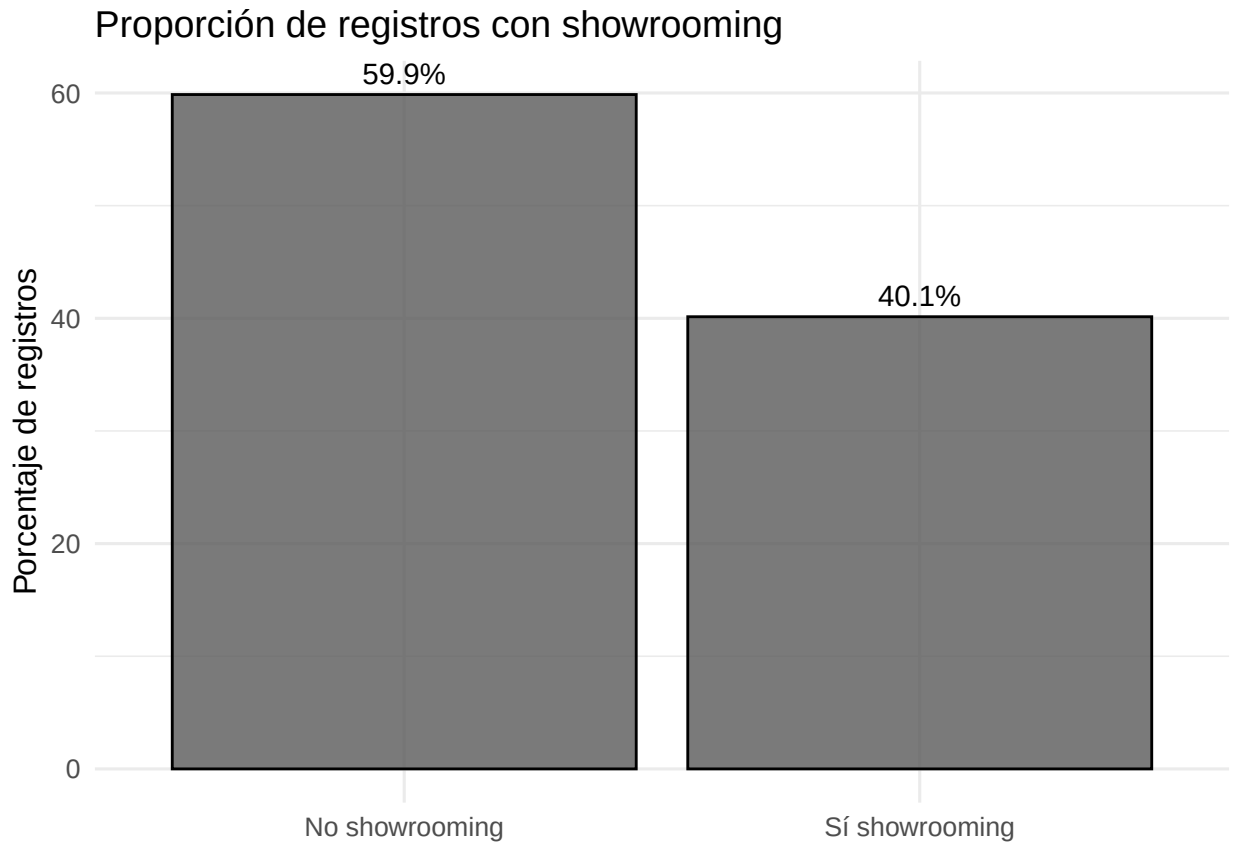
4.2 Equilibrio Escenario

```r
df_clean %>% group_by(ESCENARIO) %>% count()
```

```
## # A tibble: 4 x 2
## # Groups:   ESCENARIO [4]
##   ESCENARIO     n
##       <dbl> <int>
## 1         1   180
## 2         2   180
## 3         3   180
## 4         4   180
```

4.3 Tasa global Showrooming

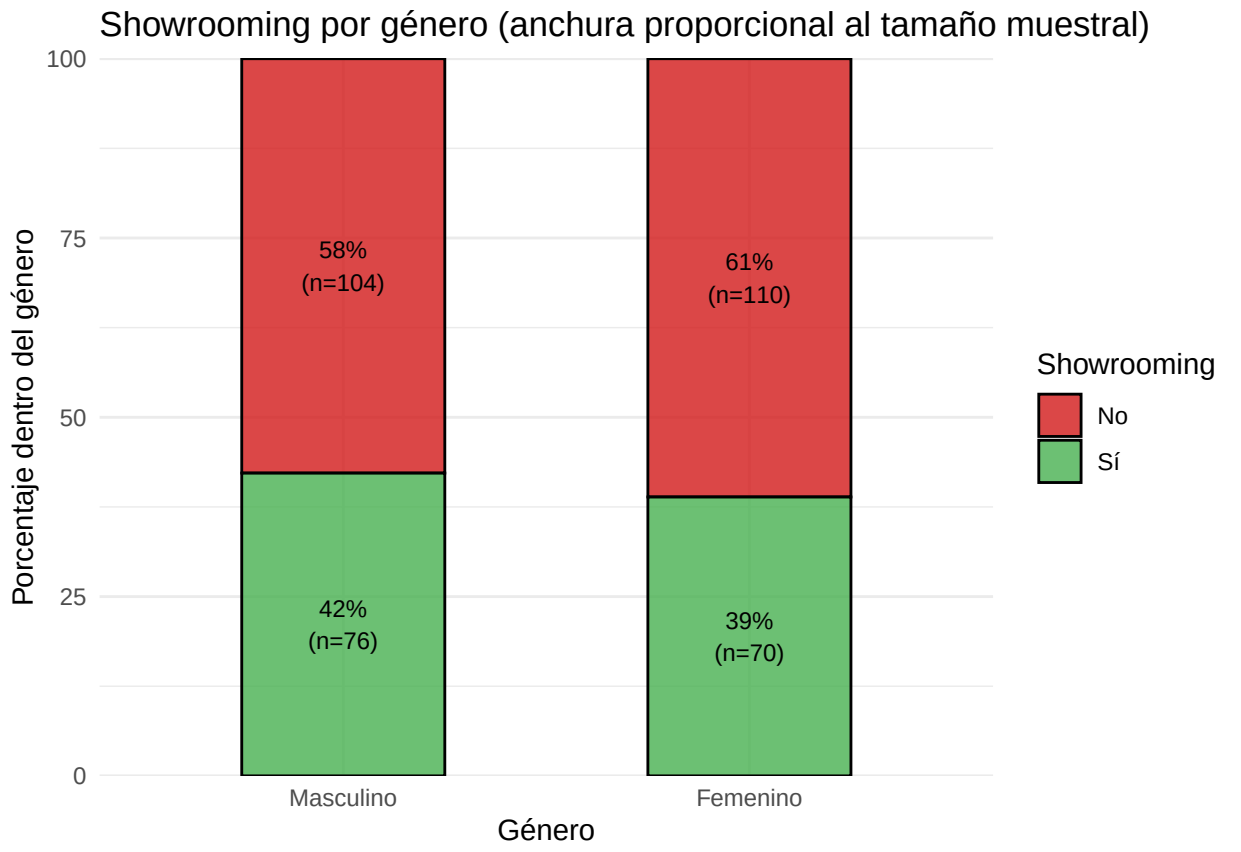
```r
# 1. Calcular frec. y % de showrooming
show_stats <- df_clean %>%
  count(SHOWROOMING) %>%
  mutate(
    pct = n / sum(n) * 100,
    etiqueta = if_else(SHOWROOMING == "Sí", "Sí showrooming", "No showrooming")
  )

# 2. Graficar
ggplot(show_stats, aes(x = etiqueta, y = pct)) +
  geom_col(color = "black", alpha = 0.8) +
  geom_text(aes(label = sprintf("%1.1f%%", pct)),
            vjust = -0.5) +
  labs(
    x = NULL,
    y = "Porcentaje de registros",
    title = "Proporción de registros con showrooming"
  ) +
  theme_minimal(base_size = 12)
```



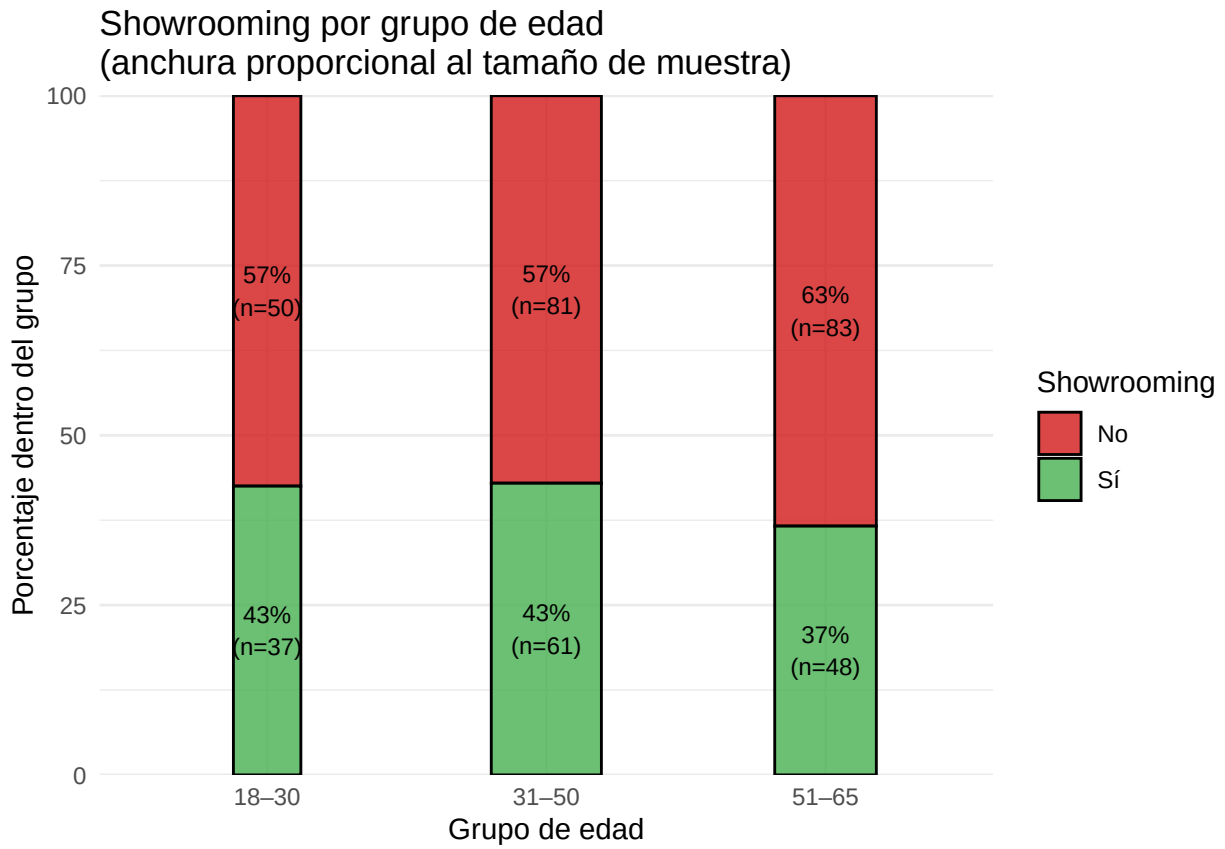
Distribución showrooming según variables sociodemográficas

```r
# 1) calculamos N y peso de cada género
peso_genero <- df_unicos %>%
  count(S2) %>%
```

```r
  rename(N_gen = n) %>%
  mutate(w_gen = N_gen / sum(N_gen))

# 2) calculamos % showrooming dentro de cada género
pct_genero <- df_unicos %>%
  count(S2, SHOWROOMING) %>%
  group_by(S2) %>%
  mutate(pct = n / sum(n) * 100) %>%
  ungroup() %>%
  left_join(peso_genero, by = "S2")

# 3) graficamos, usando width = w_gen en geom_col
ggplot(pct_genero, aes(x = S2, y = pct, fill = SHOWROOMING, width = w_gen)) +
  geom_col(color = "black", alpha = 0.8, position = "stack") +
  geom_text(aes(label = sprintf("%1.0f%%\n(n=%d)", pct, n)),
            position = position_stack(vjust = 0.5), size = 3) +
  scale_y_continuous(expand = c(0, 0)) +
  scale_fill_manual(values = c("No" = "#d31919", "Sí" = "#48b150")) +
  labs(
    x     = "Género",
    y     = "Porcentaje dentro del género",
    fill  = "Showrooming",
    title = "Showrooming por género (anchura proporcional al tamaño muestral)"
  ) +
  theme_minimal()
```

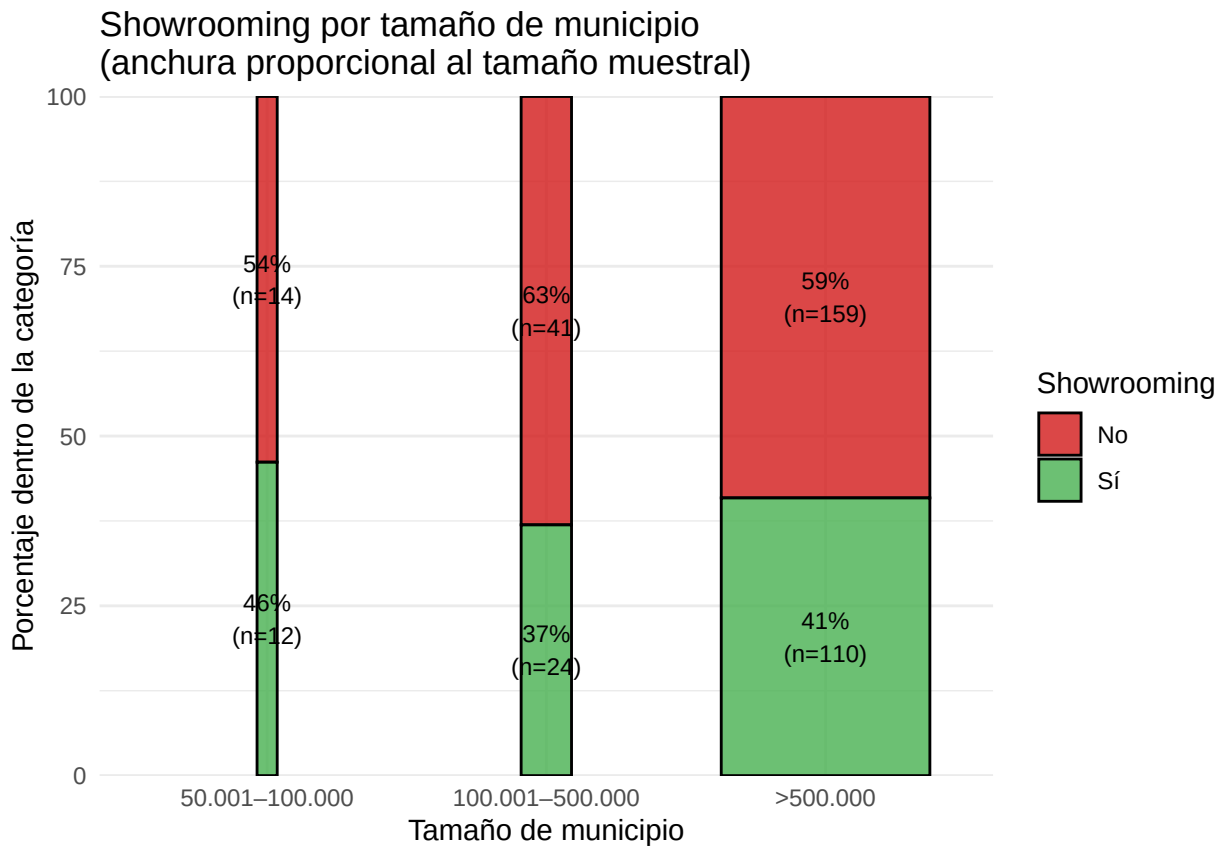


```r
# Edad
# 1) Calculamos N y peso de cada grupo de edad
peso_edad <- df_unicos %>%
  count(CuotaEdad) %>%
  rename(N_edad = n) %>%
  mutate(w_edad = N_edad / sum(N_edad))

# 2) Calculamos % showrooming dentro de cada grupo de edad
pct_edad <- df_unicos %>%
  count(CuotaEdad, SHOWROOMING) %>%
  group_by(CuotaEdad) %>%
  mutate(pct = n / sum(n) * 100) %>%
  ungroup() %>%
  left_join(peso_edad, by = "CuotaEdad")

# 3) Graficamos: barras apiladas cuya anchura es proporcional al tamaño muestral
ggplot(pct_edad, aes(x = CuotaEdad, y = pct, fill = SHOWROOMING, width = w_edad)) +
  geom_col(position = "stack", color = "black", alpha = 0.8) +
  geom_text(aes(label = sprintf("%1.0f%%\n(n=%d)", pct, n)),
            position = position_stack(vjust = 0.5), size = 3) +
  scale_y_continuous(expand = c(0, 0)) +
  scale_fill_manual(values = c("No" = "#d31919", "Sí" = "#48b150")) +
  labs(
    x     = "Grupo de edad",
    y     = "Porcentaje dentro del grupo",
    fill  = "Showrooming",
    title = "Showrooming por grupo de edad\n(anchura proporcional al tamaño de muestra)"
  ) +
  theme_minimal()
```



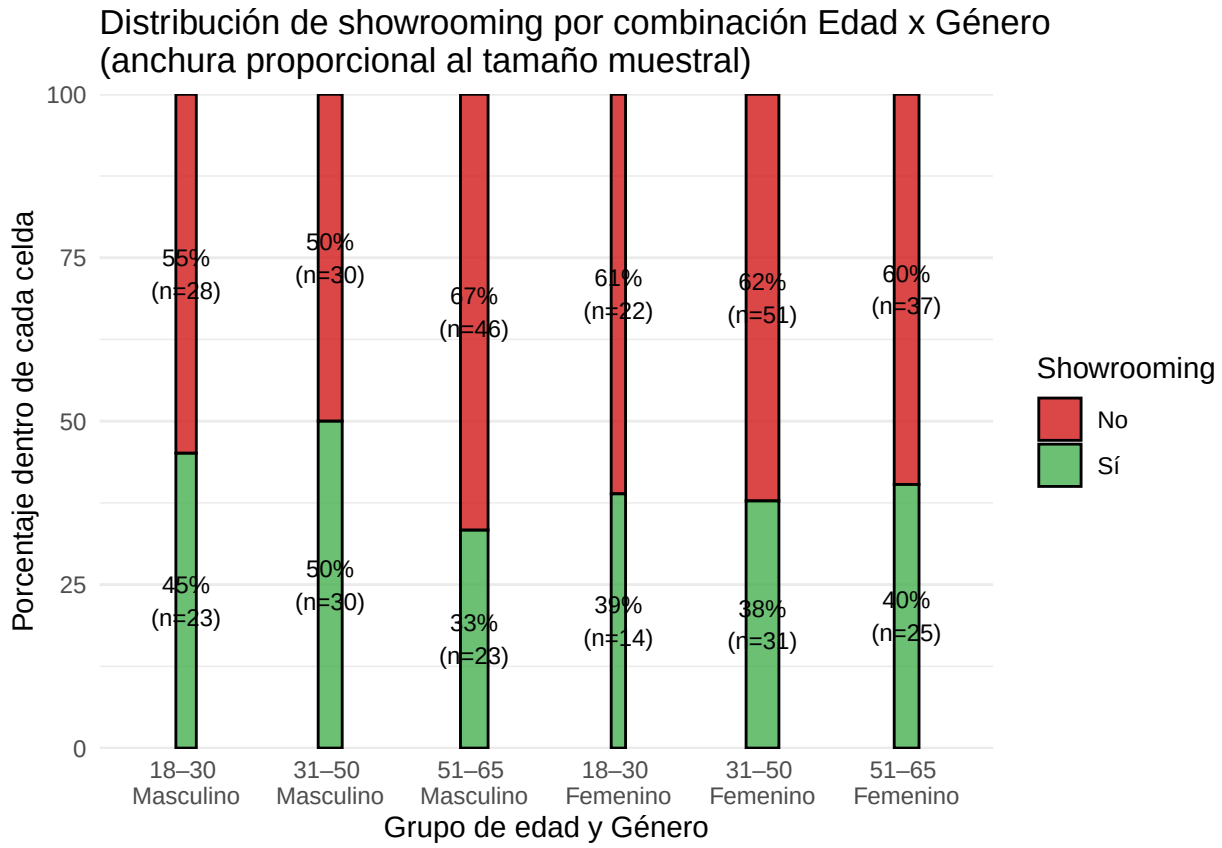

```r
# 1) Calculamos N y peso de cada categoría de tamaño de municipio
peso_pob <- df_unicos %>%
  count(TamPob) %>%
  rename(N_pob = n) %>%
  mutate(w_pob = N_pob / sum(N_pob))

# 2) Calculamos % showrooming dentro de cada categoría
pct_pob <- df_unicos %>%
  count(TamPob, SHOWROOMING) %>%
  group_by(TamPob) %>%
  mutate(pct = n / sum(n) * 100) %>%
  ungroup() %>%
  left_join(peso_pob, by = "TamPob")

# 3) Graficamos: barras apiladas cuya anchura (width) es w_pob
ggplot(pct_pob, aes(x = TamPob, y = pct, fill = SHOWROOMING, width = w_pob)) +
  geom_col(position = "stack", color = "black", alpha = 0.8) +
  geom_text(aes(label = sprintf("%1.0f%%\n(n=%d)", pct, n)),
            position = position_stack(vjust = 0.5), size = 3) +
  scale_y_continuous(expand = c(0, 0)) +
  scale_fill_manual(values = c("No" = "#d31919", "Sí" = "#48b150")) +
  labs(
    x     = "Tamaño de municipio",
    y     = "Porcentaje dentro de la categoría",
    fill  = "Showrooming",
    title = "Showrooming por tamaño de municipio\n(anchura proporcional al tamaño muestral)"
```

```
  ) +
  theme_minimal()
```



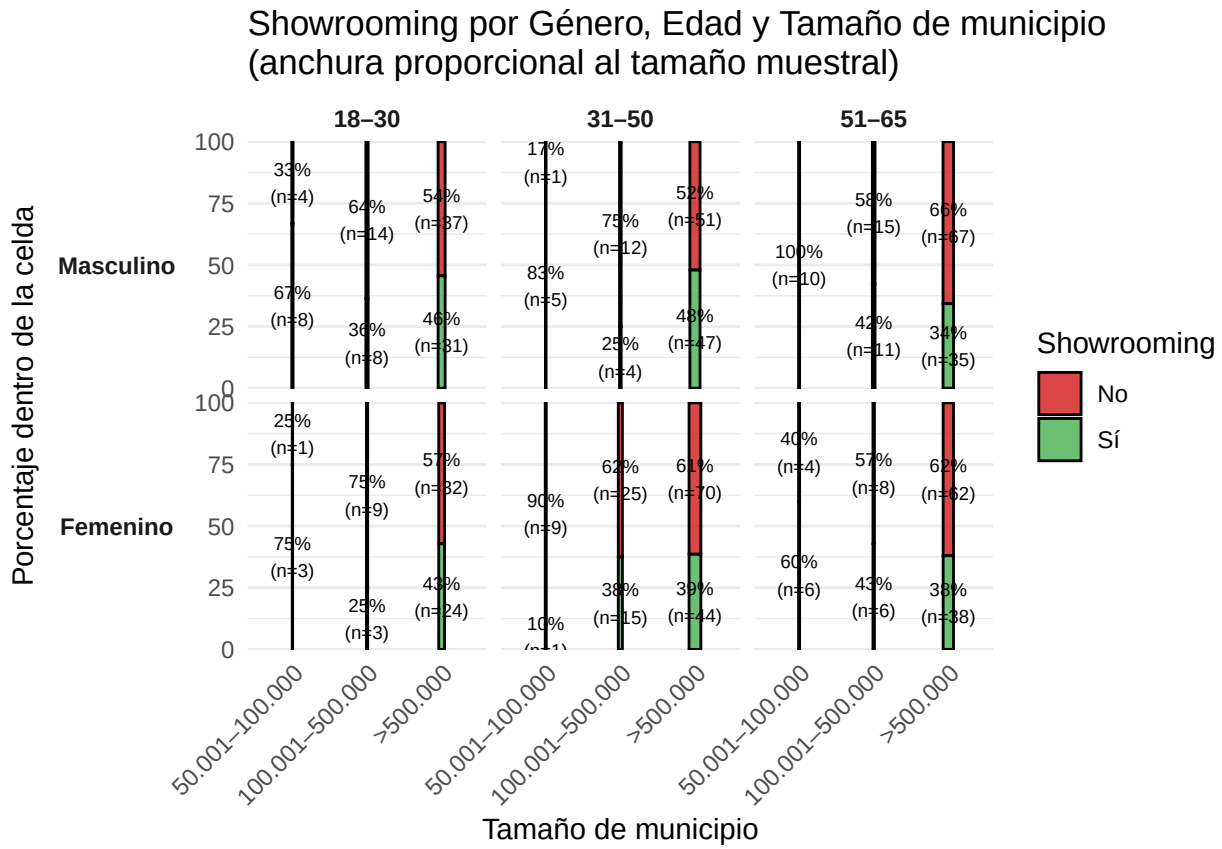

```
# 1) Peso de cada combinación Género x Edad
peso_ge <- df_unicos %>%
  count(S2, CuotaEdad) %>%
  rename(N_ge = n) %>%
  mutate(w_ge = N_ge / sum(N_ge))

# 2) % showrooming dentro de cada combinación
pct_ge <- df_unicos %>%
  count(S2, CuotaEdad, SHOWROOMING) %>%
  group_by(S2, CuotaEdad) %>%
  mutate(pct = n / sum(n) * 100) %>%
  ungroup() %>%
  left_join(peso_ge, by = c("S2", "CuotaEdad"))

# 3) Gráfico
ggplot(pct_ge, aes(x = interaction(CuotaEdad, S2, sep = " \n"),
                   y = pct,
                   fill = SHOWROOMING,
                   width = w_ge)) +
  geom_col(position = "stack", color = "black", alpha = 0.8) +
  geom_text(aes(label = sprintf("%1.0f%%\n(n=%d)", pct, n)),
            position = position_stack(vjust = 0.5), size = 3) +
  scale_y_continuous(expand = c(0, 0)) +
```

```
  scale_fill_manual(values = c("No" = "#d31919", "Sí" = "#48b150")) +
  labs(
    x     = "Grupo de edad y Género",
    y     = "Porcentaje dentro de cada celda",
    fill  = "Showrooming",
    title = "Distribución de showrooming por combinación Edad x Género\n(anchura proporcional al tamaño
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(hjust = 0.5),
    panel.grid.major.x = element_blank()
  )
```




```
# 1) Peso de cada combinación GéneroxEdadxMunicipio
peso_tripleta <- df_clean %>%
  count(S2, CuotaEdad, TamPob) %>%
  rename(N_trip = n) %>%
  mutate(w_trip = N_trip / sum(N_trip))

# 2) % showrooming dentro de cada combinación
pct_tripleta <- df_clean %>%
  count(S2, CuotaEdad, TamPob, SHOWROOMING) %>%
  group_by(S2, CuotaEdad, TamPob) %>%
  mutate(pct = n / sum(n) * 100) %>%
  ungroup() %>%
  left_join(peso_tripleta, by = c("S2", "CuotaEdad", "TamPob"))
```

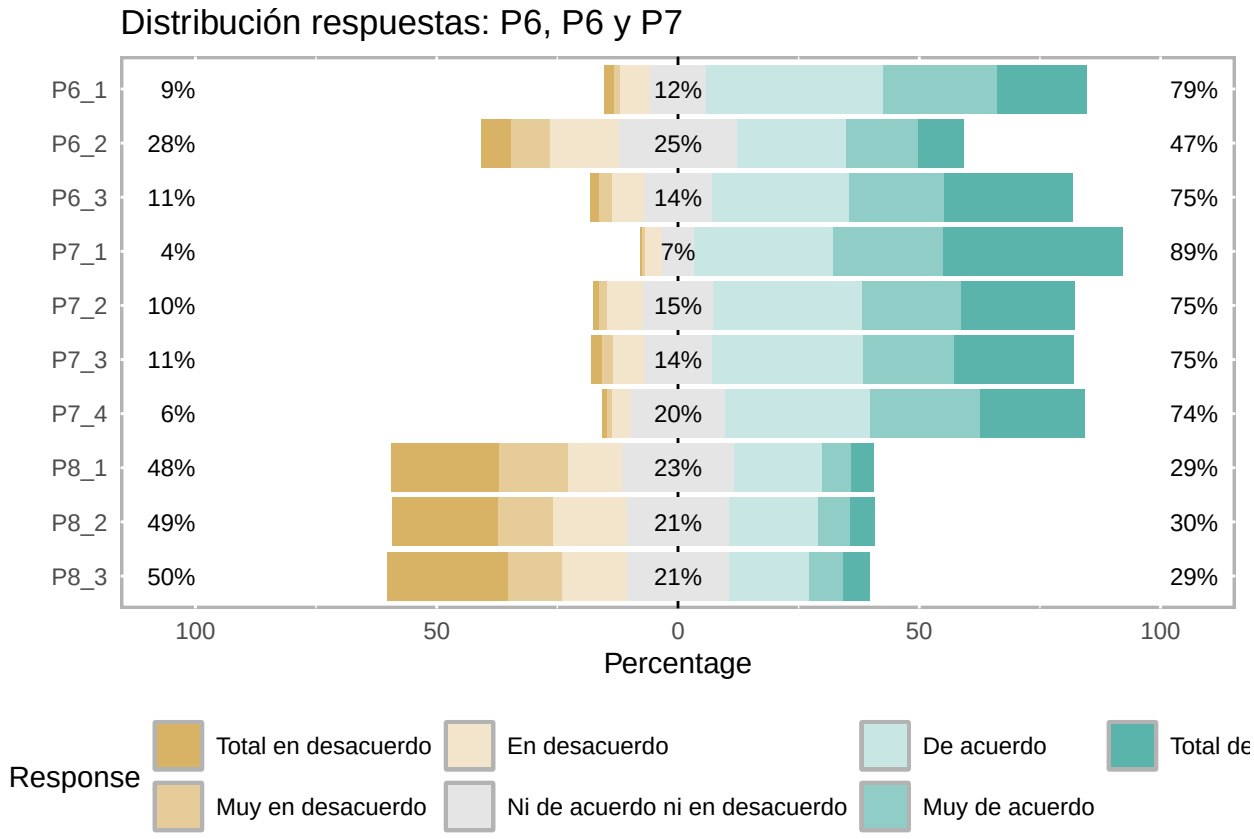
```r
# 3) Gráfico con facet_grid Género vs Edad
ggplot(pct_tripleta,
       aes(x = TamPob,
           y = pct,
           fill = SHOWROOMING,
           width = w_trip)) +
  geom_col(position = "stack", color = "black", alpha = 0.8) +
  geom_text(aes(label = sprintf("%1.0f%%\n(n=%d)", pct, n)),
            position = position_stack(vjust = 0.5),
            size = 2.5) +
  facet_grid(rows = vars(S2), cols = vars(CuotaEdad),
             scales = "free_x", space = "free_x",
             switch = "y") +
  scale_y_continuous(expand = c(0, 0)) +
  scale_fill_manual(values = c("No" = "#d31919", "Sí" = "#48b150")) +
  labs(
    x     = "Tamaño de municipio",
    y     = "Porcentaje dentro de la celda",
    fill  = "Showrooming",
    title = "Showrooming por Género, Edad y Tamaño de municipio\n(anchura proporcional al tamaño muestra
  ) +
  theme_minimal() +
  theme(
    axis.text.x        = element_text(angle = 45, hjust = 1),
    strip.placement    = "outside",
    strip.background.x = element_blank(),
    strip.background.y = element_blank(),
    strip.text.x       = element_text(face = "bold"),
    strip.text.y.left  = element_text(face = "bold", angle = 0),
    panel.grid.major.x = element_blank()
  )
```



4.4 Distribución Escalas Likert

```
niveles <- c(
  "Total en desacuerdo",
  "Muy en desacuerdo",
  "En desacuerdo",
  "Ni de acuerdo ni en desacuerdo",
  "De acuerdo",
  "Muy de acuerdo",
  "Total de acuerdo"
)

to_factor_block <- function(cols) {
  lapply(df_clean[ , cols], function(x)
    factor(x, levels = 1:7, labels = niveles, ordered = TRUE)
  )
}

P6_fact <- as.data.frame(to_factor_block(c("P6_1","P6_2","P6_3")))
P7_fact <- as.data.frame(to_factor_block(c("P7_1","P7_2","P7_3","P7_4")))
P8_fact <- as.data.frame(to_factor_block(c("P8_1","P8_2","P8_3")))

all_items <- likert(cbind(P6_fact, P7_fact, P8_fact))

plot(all_items,
```

```
    group.order = c(
      "P6_1", "P6_2", "P6_3",
      "P7_1", "P7_2", "P7_3", "P7_4",
      "P8_1", "P8_2", "P8_3"
    )
) +
  ggtitle("Distribución respuestas: P6, P6 y P7")
```



5 PROPIEDADES PSICOMETRICAS

5.1 Realismo escenario

5.1.1 Levene's Test (homogeneidad de varianzas)

```
leveneTest(P4 ~ Escenario, data = df_clean)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value  Pr(>F)
## group   3   2.437 0.06352 .
##       716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

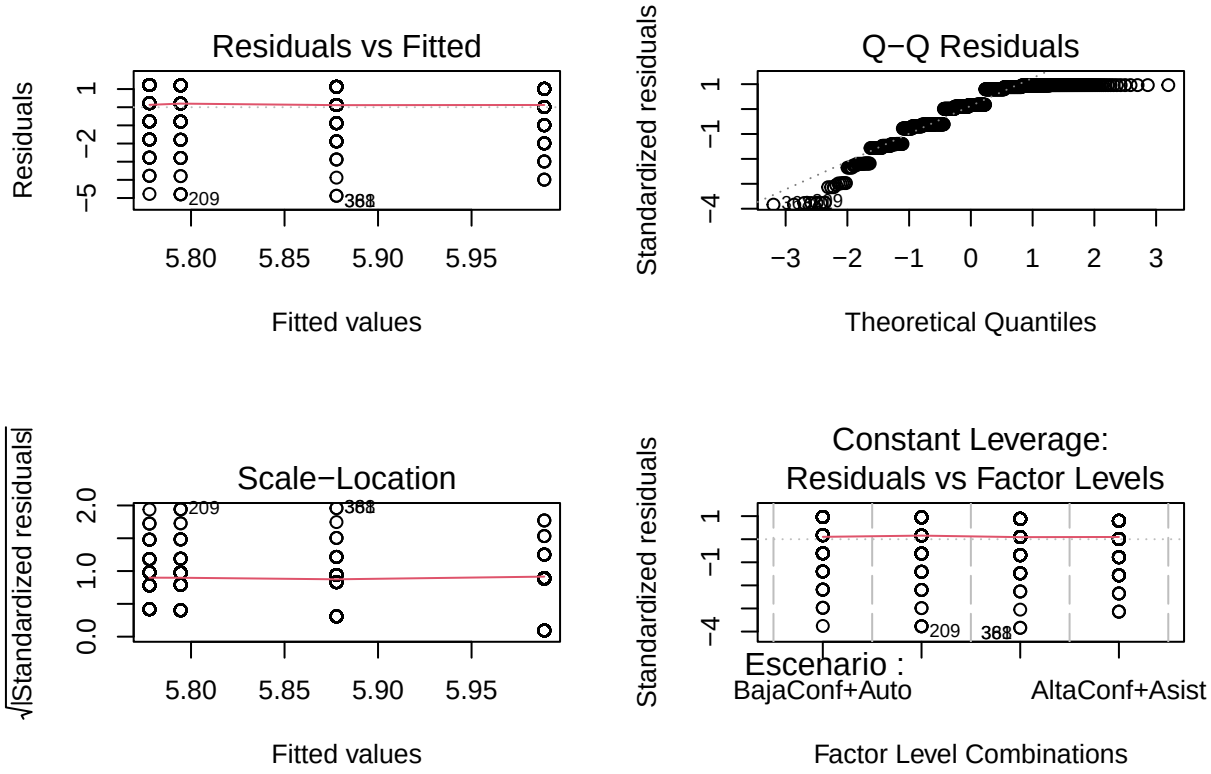
5.1.2 ANOVA unifactorial

```
aov_p4 <- aov(P4 ~ Escenario, data = df_clean)
summary(aov_p4)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Escenario     3      5   1.679   1.033  0.377
## Residuals   716   1164   1.625
```

5.1.3 Diagnóstico residuos

```
par(mfrow = c(2,2))
plot(aov_p4)
```



5.1.4 Comparaciones post-hoc (Tukey)

```
emmeans(aov_p4, "Escenario") %>%
  contrast(method = "pairwise", adjust = "tukey")
```

```
##  contrast                              estimate    SE  df t.ratio p.value
##  (BajaConf+Auto) - (AltaConf+Auto)      -0.0167 0.134 716  -0.124  0.9993
##  (BajaConf+Auto) - (BajaConf+Asist)     -0.1000 0.134 716  -0.744  0.8792
##  (BajaConf+Auto) - (AltaConf+Asist)     -0.2111 0.134 716  -1.571  0.3960
##  (AltaConf+Auto) - (BajaConf+Asist)     -0.0833 0.134 716  -0.620  0.9257
##  (AltaConf+Auto) - (AltaConf+Asist)     -0.1944 0.134 716  -1.447  0.4705
##  (BajaConf+Asist) - (AltaConf+Asist)    -0.1111 0.134 716  -0.827  0.8418
```

```
##
## P value adjustment: tukey method for comparing a family of 4 estimates
```

5.2 Chequeo de confianza percibida

```
# 1 Construir tabla de contingencia
tab_conf <- table(df_clean$Escenario, df_clean$P5_1,
                  dnn = c("Escenario", "Percibió confianza"))

# 2 Ver tabla observada
print(tab_conf)
```

```
##                 Percibió confianza
## Escenario          1   2
##   BajaConf+Auto   56 124
##   AltaConf+Auto  147  33
##   BajaConf+Asist  45 135
##   AltaConf+Asist 149  31
```

```
# 3 Test X^2
chisq_conf <- chisq.test(tab_conf, correct = FALSE)
chisq_conf
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab_conf
## X-squared = 214.91, df = 3, p-value < 2.2e-16
```

```
# 4 Tabla de frecuencias esperadas
chisq_conf$expected
```

```
##                 Percibió confianza
## Escenario            1     2
##   BajaConf+Auto  99.25 80.75
##   AltaConf+Auto  99.25 80.75
##   BajaConf+Asist 99.25 80.75
##   AltaConf+Asist 99.25 80.75
```

```
# 5 Residuos estandarizados
round(chisq_conf$stdres, 2)
```

```
##                 Percibió confianza
## Escenario            1     2
##   BajaConf+Auto  -7.48  7.48
##   AltaConf+Auto   8.26 -8.26
##   BajaConf+Asist -9.39  9.39
##   AltaConf+Asist  8.61 -8.61
```

5.3 Chequeo de servicio percibido

```
# 1 Construir tabla de contingencia
tab_serv <- table(df_clean$Escenario, df_clean$P5_2,
                  dnn = c("Escenario", "Percibió servicio"))
```

```r
# 2 Ver tabla observada
print(tab_serv)
```

```
##                 Percibió servicio
## Escenario          1   2
##   BajaConf+Auto   49 131
##   AltaConf+Auto   43 137
##   BajaConf+Asist 155  25
##   AltaConf+Asist 152  28
```

```r
# 3 Test X2
chisq_serv <- chisq.test(tab_serv, correct = FALSE)
chisq_serv
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab_serv
## X-squared = 260.36, df = 3, p-value < 2.2e-16
```

```r
# 4 Tabla de frecuencias esperadas
chisq_serv$expected
```

```
##                 Percibió servicio
## Escenario            1     2
##   BajaConf+Auto  99.75 80.25
##   AltaConf+Auto  99.75 80.25
##   BajaConf+Asist 99.75 80.25
##   AltaConf+Asist 99.75 80.25
```

```r
# 5 Residuos estandarizados
round(chisq_serv$stdres, 2)
```

```
##                 Percibió servicio
## Escenario            1     2
##   BajaConf+Auto  -8.79  8.79
##   AltaConf+Auto  -9.83  9.83
##   BajaConf+Asist  9.57 -9.57
##   AltaConf+Asist  9.05 -9.05
```

5.4 Análisis Factorial Confirmatorio y propiedades psiconometricas

5.4.1 Definición del modelo de medida

```r
modelo_cfa <- '
  # 1) Definición de los factores latentes y sus indicadores
  GratInmed  =~ 1*P6_1 + P6_2 + P6_3
  Precio     =~ 1*P7_1 + P7_2 + P7_3 + P7_4
  MalaConsc  =~ 1*P8_1 + P8_2 + P8_3

  # 2) Liberar las covarianzas entre los tres factores
  GratInmed ~~ Precio
  GratInmed ~~ MalaConsc
  Precio    ~~ MalaConsc
'
```

5.4.2 Estimación del CFA

```r
# 1) Ajuste del CFA
fit <- cfa(
  model     = modelo_cfa,
  data      = df_clean,
  std.lv    = FALSE,           # escalamos fijando cargas, no varianza latente
  estimator = "MLM"            # ML robusto
)

# 2) Resumen con medidas de ajuste y estandarizados
summary(fit,
        fit.measures = TRUE,
        standardized = TRUE)
```

```
## lavaan 0.6-19 ended normally after 38 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        23
##
##   Number of observations                           720
##
## Model Test User Model:
##                                              Standard      Scaled
##   Test Statistic                              201.881     146.022
##   Degrees of freedom                               32          32
##   P-value (Chi-square)                          0.000       0.000
##   Scaling correction factor                                 1.383
##     Satorra-Bentler correction
##
## Model Test Baseline Model:
##
##   Test statistic                             3007.034    2378.339
##   Degrees of freedom                               45          45
##   P-value                                       0.000       0.000
##   Scaling correction factor                                 1.264
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                   0.943       0.951
##   Tucker-Lewis Index (TLI)                      0.919       0.931
##
##   Robust Comparative Fit Index (CFI)                        0.947
##   Robust Tucker-Lewis Index (TLI)                           0.925
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)             -11639.506  -11639.506
##   Loglikelihood unrestricted model (H1)     -11538.565  -11538.565
##
##   Akaike (AIC)                               23325.011   23325.011
##   Bayesian (BIC)                             23430.334   23430.334
##   Sample-size adjusted Bayesian (SABIC)      23357.303   23357.303
```

```
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.086       0.070
##   90 Percent confidence interval - lower         0.075       0.061
##   90 Percent confidence interval - upper         0.097       0.080
##   P-value H_0: RMSEA <= 0.050                    0.000       0.000
##   P-value H_0: RMSEA >= 0.080                    0.812       0.056
##
##   Robust RMSEA                                                0.083
##   90 Percent confidence interval - lower                      0.069
##   90 Percent confidence interval - upper                      0.097
##   P-value H_0: Robust RMSEA <= 0.050                          0.000
##   P-value H_0: Robust RMSEA >= 0.080                          0.645
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.047       0.047
##
## Parameter Estimates:
##
##   Standard errors                             Robust.sem
##   Information                                   Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   GratInmed =~
##     P6_1             1.000                               1.063    0.812
##     P6_2             0.975    0.079   12.402    0.000    1.037    0.645
##     P6_3             0.774    0.077   10.113    0.000    0.823    0.568
##   Precio =~
##     P7_1             1.000                               0.856    0.731
##     P7_2             1.233    0.065   18.851    0.000    1.056    0.778
##     P7_3             1.250    0.084   14.858    0.000    1.071    0.745
##     P7_4             1.067    0.070   15.205    0.000    0.914    0.727
##   MalaConsc =~
##     P8_1             1.000                               1.458    0.823
##     P8_2             1.051    0.044   24.141    0.000    1.533    0.862
##     P8_3             1.113    0.044   25.382    0.000    1.622    0.884
##
## Covariances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   GratInmed ~~
##     Precio           0.266    0.052    5.121    0.000    0.292    0.292
##     MalaConsc        0.186    0.074    2.526    0.012    0.120    0.120
##   Precio ~~
##     MalaConsc       -0.237    0.059   -3.980    0.000   -0.190   -0.190
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .P6_1             0.584    0.095    6.170    0.000    0.584    0.341
##    .P6_2             1.510    0.142   10.628    0.000    1.510    0.584
##    .P6_3             1.418    0.105   13.459    0.000    1.418    0.677
```

```
##    .P7_1             0.640    0.056   11.444    0.000    0.640    0.466
##    .P7_2             0.728    0.079    9.262    0.000    0.728    0.395
##    .P7_3             0.918    0.112    8.190    0.000    0.918    0.445
##    .P7_4             0.744    0.062   11.932    0.000    0.744    0.471
##    .P8_1             1.014    0.121    8.403    0.000    1.014    0.323
##    .P8_2             0.812    0.114    7.099    0.000    0.812    0.257
##    .P8_3             0.738    0.107    6.933    0.000    0.738    0.219
##     GratInmed        1.131    0.123    9.167    0.000    1.000    1.000
##     Precio           0.733    0.076    9.659    0.000    1.000    1.000
##     MalaConsc        2.125    0.151   14.096    0.000    1.000    1.000
```

5.4.3 Valoración de la validez y fiabilidad del instrumento de medida

```
reliability(fit)[1,]
```

5.4.3.1 Fiabilidad individual: Alpha de Cronbach

```
## GratInmed    Precio MalaConsc
## 0.6991493 0.8312931 0.8914530
```

```
reliability(fit)[4,]
```

5.4.3.2 Fiabilidad compuesta (CR)

```
## GratInmed    Precio MalaConsc
## 0.7132338 0.8333141 0.8926309
```

```
reliability(fit)[5,]
```

5.4.3.3 Validez convergente (AVE)

```
## GratInmed    Precio MalaConsc
## 0.4507367 0.5581757 0.7348126
```

```
lavInspect(fit, what="cor.lv")^2
```

5.4.3.4 Validez discriminante

```
##           GrtInm Precio MlCnsc
## GratInmed  1.000
## Precio     0.085  1.000
## MalaConsc  0.014  0.036  1.000
```

```
htmt(modelo_cfa, df_clean)
```

```
##           GrtInm Precio MlCnsc
## GratInmed  1.000
## Precio     0.280  1.000
## MalaConsc  0.122  0.190  1.000
```

6 ANÁLISIS EXPLORATORIO

6.1 Estadísticos descriptivos y matriz de correlaciones

```
# 1 Crear composites con nombres explícitos
df_clean <- df_clean %>%
  mutate(
    gratificacion    = rowMeans(select(., P6_1:P6_3), na.rm = TRUE),
    orient_precio    = rowMeans(select(., P7_1:P7_4), na.rm = TRUE),
    culpa            = rowMeans(select(., P8_1:P8_3), na.rm = TRUE)
  )

# 2 Recodificar variables de manipulación
df_clean <- df_clean %>%
  mutate(
    confianza_bin = if_else(ESCENARIO %in% c(2,4), 1, 0),
    servicio_bin  = if_else(ESCENARIO %in% c(3,4), 1, 0)
  )

# 3. Estadísticos descriptivos
est <- describe(df_clean %>%
           select(confianza_bin, servicio_bin, gratificacion, orient_precio, culpa))

# 4. Matriz de correlaciones (punto-biserial para binarios y Pearson para continuos)
#    Tratamos los binarios como numéricos 0/1
vars_modelo <- c("confianza_bin", "servicio_bin",
                 "gratificacion", "orient_precio", "culpa")
cor_mat <- df_clean %>%
  select(all_of(vars_modelo)) %>%
  cor(use = "pairwise.complete.obs", method = "pearson")

print(cor_mat)
```

```
##               confianza_bin servicio_bin gratificacion orient_precio
## confianza_bin    1.00000000   0.00000000    0.01926146    0.02277940
## servicio_bin     0.00000000   1.00000000   -0.02166915    0.03970123
## gratificacion    0.01926146  -0.02166915    1.00000000    0.23231746
## orient_precio    0.02277940   0.03970123    0.23231746    1.00000000
## culpa           -0.01422256   0.04949450    0.12039252   -0.16846323
##                      culpa
## confianza_bin -0.01422256
## servicio_bin   0.04949450
## gratificacion  0.12039252
## orient_precio -0.16846323
## culpa          1.00000000
```

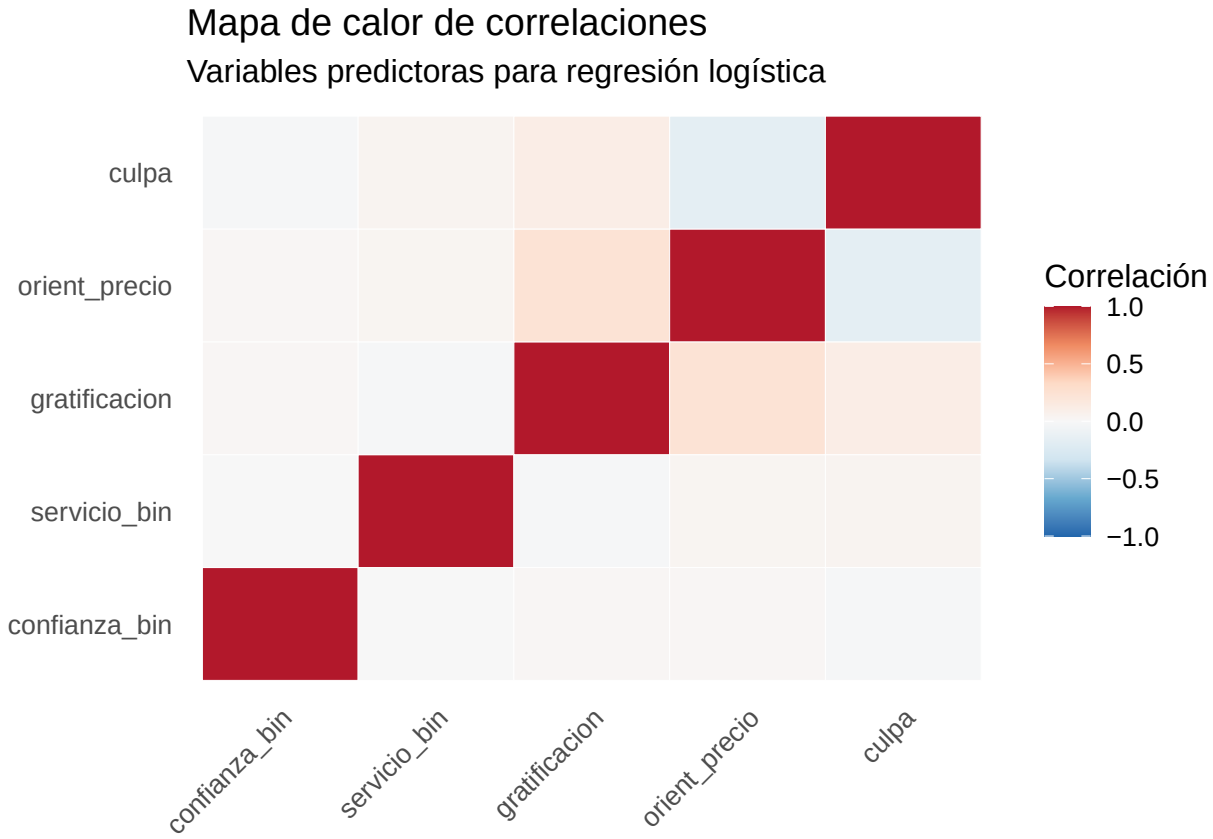
```
# 5. Significancia de correlaciones
corr.test(
  df_clean %>% select(all_of(vars_modelo)),
  use    = "pairwise",
  method = "pearson",
  adjust = "none"
)
```

```
## Call:corr.test(x = df_clean %>% select(all_of(vars_modelo)), use = "pairwise",
```

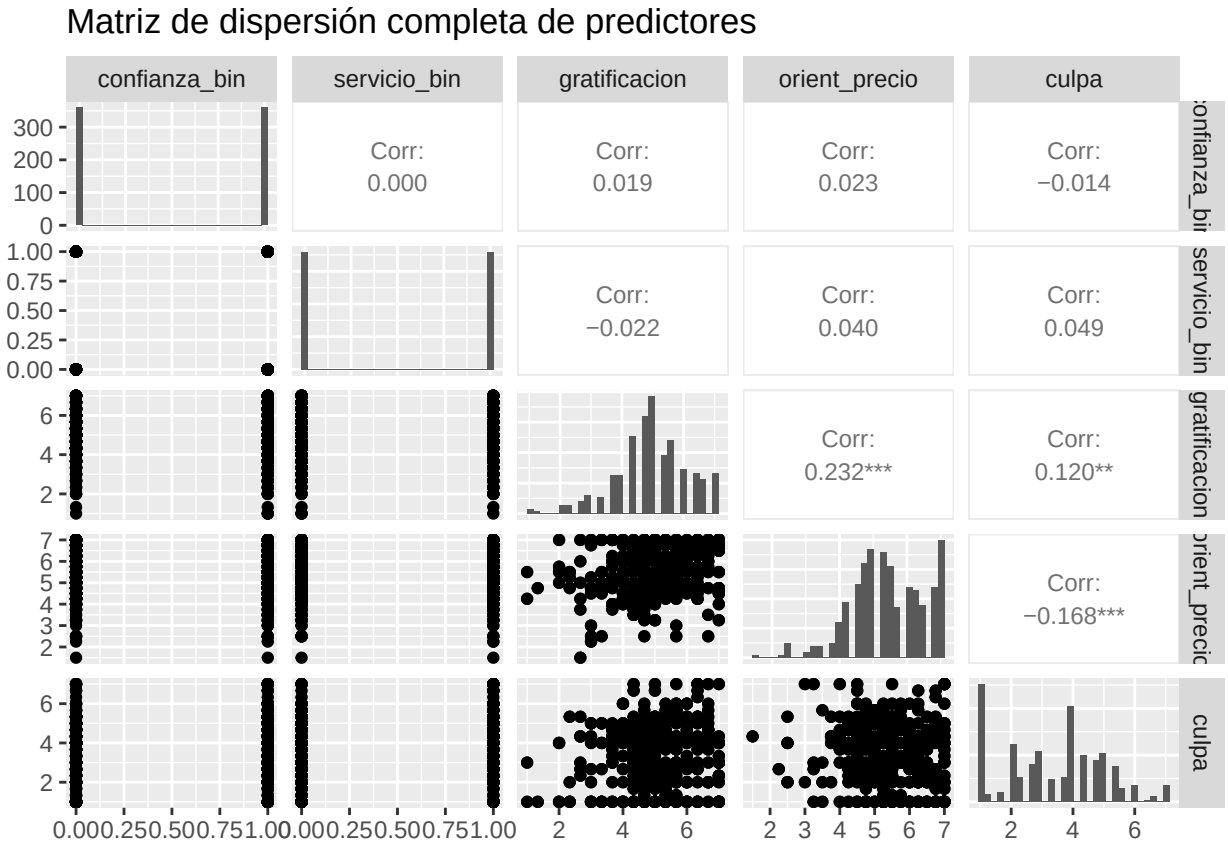
```
##     method = "pearson", adjust = "none")
## Correlation matrix
##               confianza_bin servicio_bin gratificacion orient_precio culpa
## confianza_bin          1.00         0.00          0.02          0.02 -0.01
## servicio_bin           0.00         1.00         -0.02          0.04  0.05
## gratificacion          0.02        -0.02          1.00          0.23  0.12
## orient_precio          0.02         0.04          0.23          1.00 -0.17
## culpa                 -0.01         0.05          0.12         -0.17  1.00
## Sample Size
## [1] 720
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##               confianza_bin servicio_bin gratificacion orient_precio culpa
## confianza_bin          0.00         1.00          0.61          0.54  0.70
## servicio_bin           1.00         0.00          0.56          0.29  0.18
## gratificacion          0.61         0.56          0.00          0.00  0.00
## orient_precio          0.54         0.29          0.00          0.00  0.00
## culpa                  0.70         0.18          0.00          0.00  0.00
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

6.1.1 Mapa de calor y matriz de dispersión

```r
# Mapa de calor
corr_long <- melt(
  cor_mat,
  varnames   = c("x", "y"),
  value.name = "r"
)

# 6.2 Dibujar
ggplot(corr_long, aes(x = x, y = y, fill = r)) +
  geom_tile(color = "white") +
  scale_fill_distiller(
    palette = "RdBu",
    limit   = c(-1, 1),
    name    = "Correlación"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    panel.grid  = element_blank()
  ) +
  labs(
    x         = NULL,
    y         = NULL,
    title     = "Mapa de calor de correlaciones",
    subtitle  = "Variables predictoras para regresión logística"
  )
```

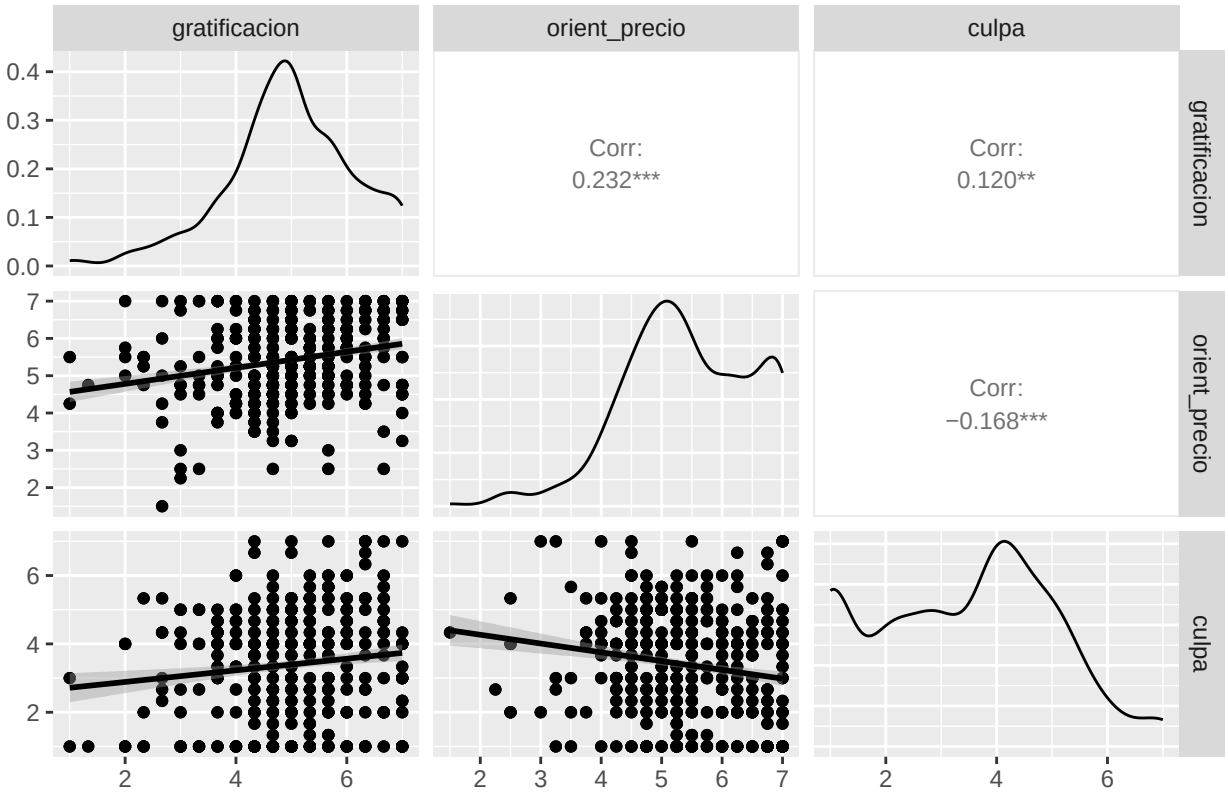



```
# Matriz de dispersión

# Matriz completa (incluyendo los binarios)
ggpairs(
  df_clean[, vars_modelo],
  upper = list(continuous = wrap("cor", size = 3)),
  lower = list(continuous = "points"),
  diag  = list(continuous = "barDiag")
  ) +
  ggtitle("Matriz de dispersión completa de predictores")
```

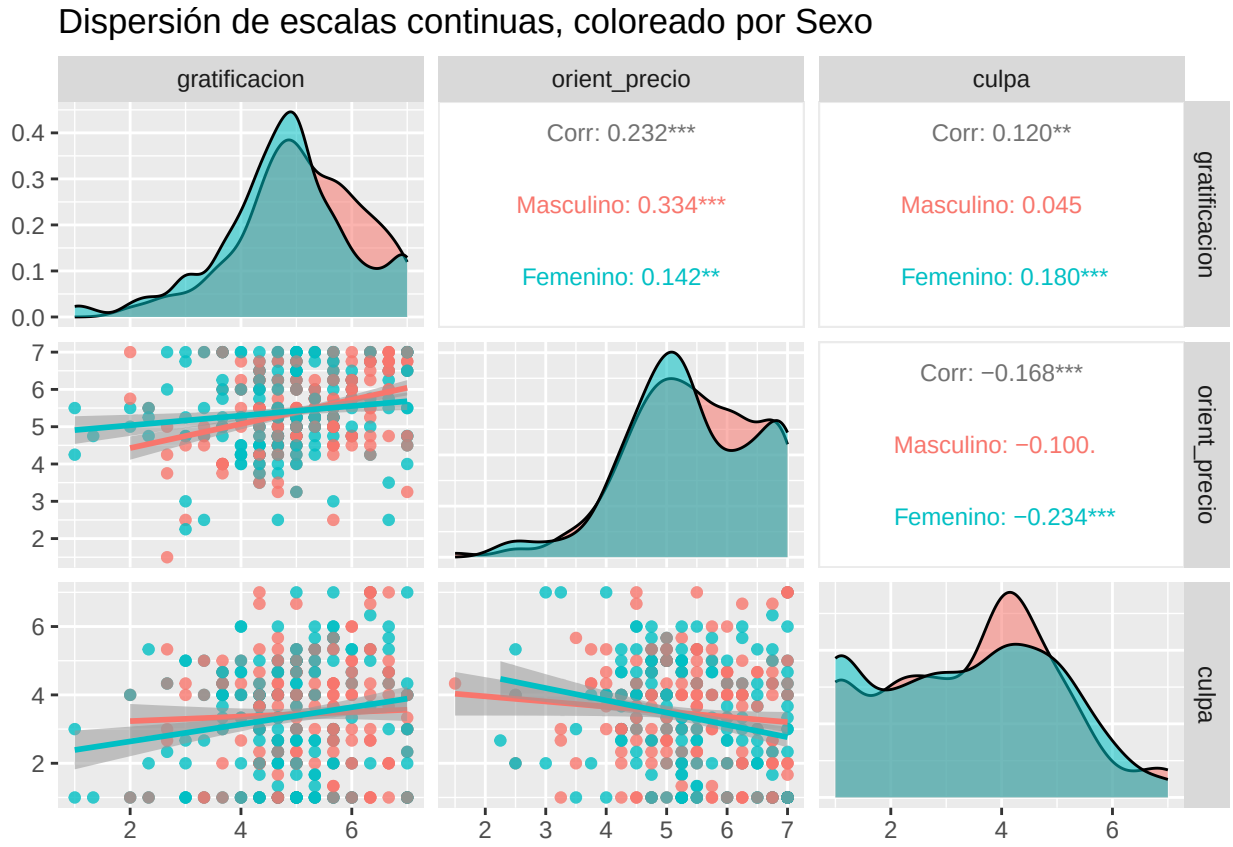



```r
# Solo continuas
ggpairs(
  df_clean[, c("gratificacion", "orient_precio", "culpa")],
  upper = list(continuous = wrap("cor", size = 3)),
  lower = list(continuous = "smooth"),
  diag  = list(continuous = "densityDiag")
) +
  ggtitle("Matriz de dispersión de las escalas continuas")
```

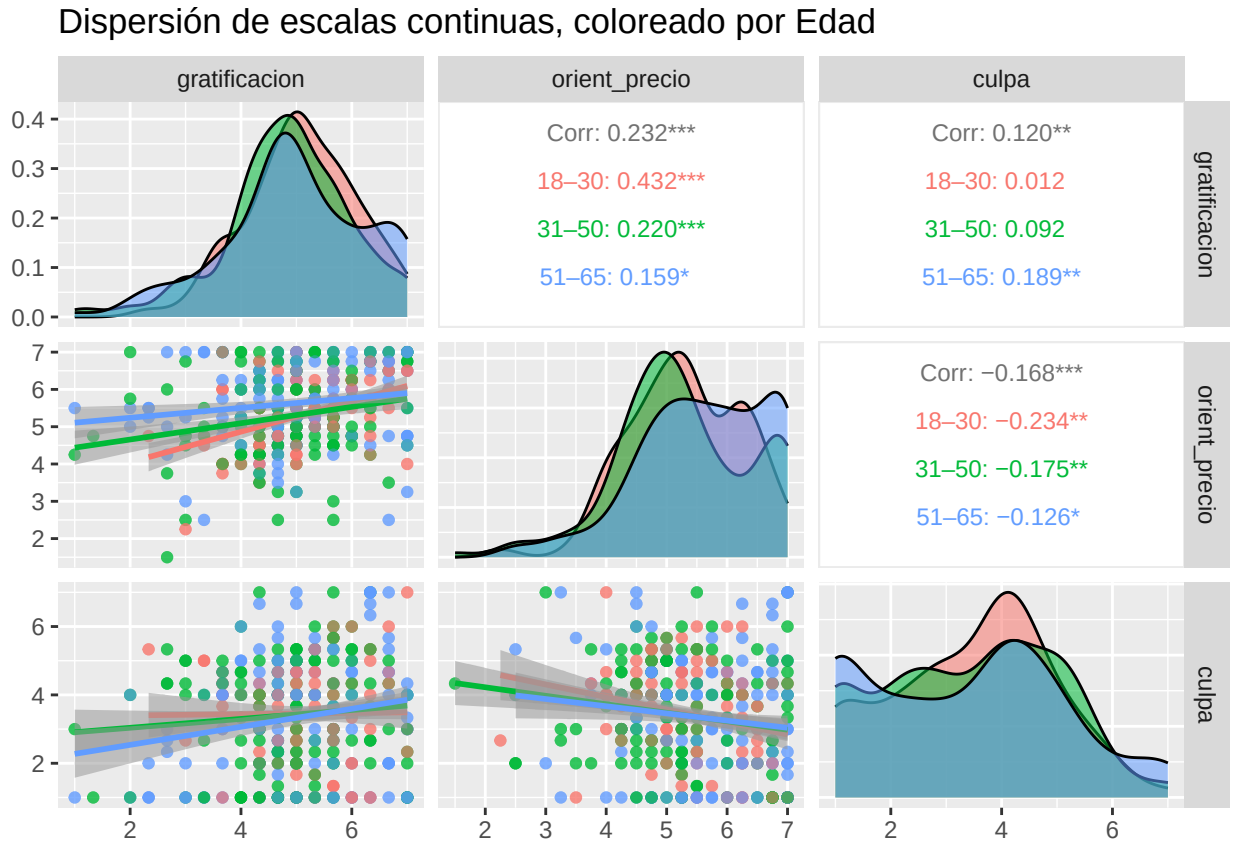
Matriz de dispersión de las escalas continuas



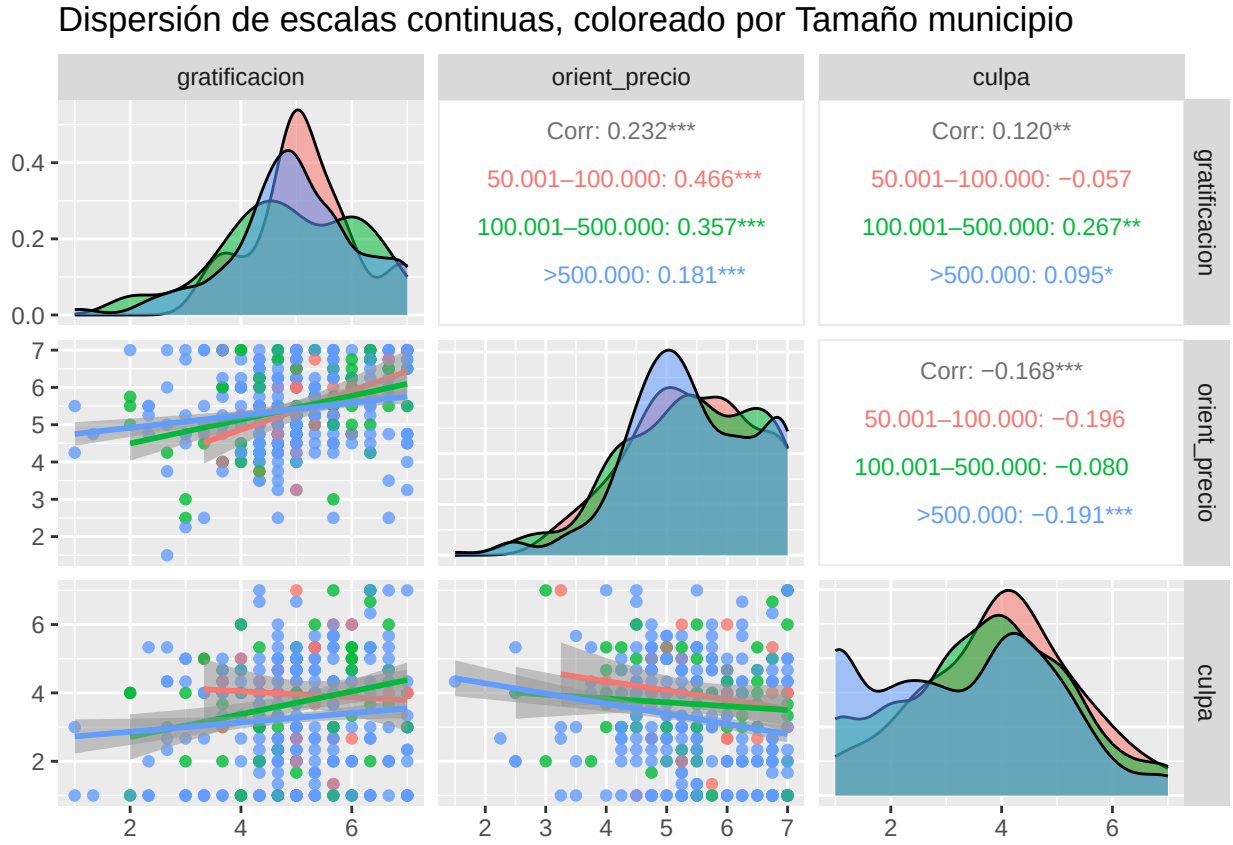

```
ggpairs(
  df_clean[, c("gratificacion", "orient_precio", "culpa")],
  mapping = aes(color = df_clean$S2, alpha=0.08),
  upper   = list(continuous = wrap("cor", size = 3)),
  lower   = list(continuous = "smooth"),
  diag    = list(continuous = "densityDiag")
) +
  ggtitle("Dispersión de escalas continuas, coloreado por Sexo")
```




```
ggpairs(
  df_clean[, c("gratificacion", "orient_precio", "culpa")],
  mapping = aes(color = df_clean$CuotaEdad, alpha=0.08),
  upper   = list(continuous = wrap("cor", size = 3)),
  lower   = list(continuous = "smooth"),
  diag    = list(continuous = "densityDiag")
) +
  ggtitle("Dispersión de escalas continuas, coloreado por Edad")
```




```
ggpairs(
  df_clean[, c("gratificacion", "orient_precio", "culpa")],
  mapping = aes(color = df_clean$TamPob, alpha=0.08),
  upper   = list(continuous = wrap("cor", size = 3)),
  lower   = list(continuous = "smooth"),
  diag    = list(continuous = "densityDiag")
) +
  ggtitle("Dispersión de escalas continuas, coloreado por Tamaño municipio")
```



7 MODELO DE REGRESIÓN LOGÍSTICA

7.1 Modelo inicial: sólo las cinco variables clave

```
modelo_base <- glm(
  SHOWROOMING ~ confianza_bin + servicio_bin +
                gratificacion + orient_precio + culpa,
  data   = df_clean,
  family = binomial(link = "logit")
)
summary(modelo_base)
```

```
##
## Call:
## glm(formula = SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa, family = binomial(link = "logit"),
##     data = df_clean)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.14517    0.53415  -0.272  0.78580
## confianza_bin -0.21343    0.15889  -1.343  0.17918
## servicio_bin   0.22989    0.15923   1.444  0.14880
## gratificacion -0.14988    0.07167  -2.091  0.03651 *
## orient_precio  0.26111    0.07994   3.266  0.00109 **
```

```
## culpa         -0.28528    0.05150  -5.539 3.03e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 969.94  on 719  degrees of freedom
## Residual deviance: 910.18  on 714  degrees of freedom
## AIC: 922.18
##
## Number of Fisher Scoring iterations: 4
```

```
AIC(modelo_base)
```

```
## [1] 922.1821
```

7.2 Ampliar con variables sociodemográficas

```
modelo_demo <- update(
  modelo_base,
  . ~ . + CuotaEdad + S2 + TamPob
)
summary(modelo_demo)
```

```
##
## Call:
## glm(formula = SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob, family = binomial(link = "logit"),
##     data = df_clean)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            0.53806    0.65303   0.824   0.4100
## confianza_bin         -0.21903    0.16025  -1.367   0.1717
## servicio_bin           0.21392    0.16032   1.334   0.1821
## gratificacion         -0.18051    0.07367  -2.450   0.0143 *
## orient_precio          0.29854    0.08257   3.616   0.0003 ***
## culpa                 -0.29494    0.05279  -5.588  2.3e-08 ***
## CuotaEdad31-50        -0.14938    0.20555  -0.727   0.4674
## CuotaEdad51-65        -0.49149    0.21224  -2.316   0.0206 *
## S2Femenino            -0.20363    0.16363  -1.244   0.2133
## TamPob100.001-500.000 -0.49700    0.34886  -1.425   0.1543
## TamPob>500.000        -0.36262    0.30676  -1.182   0.2372
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 969.94  on 719  degrees of freedom
## Residual deviance: 900.64  on 709  degrees of freedom
## AIC: 922.64
##
## Number of Fisher Scoring iterations: 4
```

```r
AIC(modelo_demo)
```

```
## [1] 922.644
```

```r
anova(modelo_base, modelo_demo, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa
## Model 2: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       714     910.18
## 2       709     900.64  5   9.5382  0.08943 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

7.3 Probar interacciones entre variables clave y demográficas

```r
#    - orient_precio x CuotaEdad
#    - gratificacion x TamPob
#    - orient_precio x S2

mod_int_op_age <- update(modelo_demo, . ~ . + orient_precio:CuotaEdad)
mod_int_gra_mun <- update(modelo_demo, . ~ . + gratificacion:TamPob)
mod_int_op_sex <- update(modelo_demo, . ~ . + orient_precio:S2)

# Comparar AIC y pruebas Chi-cuadrado
modelos_demo_int <- list(
  demo       = modelo_demo,
  op_x_age   = mod_int_op_age,
  gra_x_mun  = mod_int_gra_mun,
  op_x_sex   = mod_int_op_sex
)
sapply(modelos_demo_int, AIC)
```

```
##      demo  op_x_age gra_x_mun  op_x_sex
##  922.6440  913.0752  916.0611  917.6910
```

```r
anova(modelo_demo, mod_int_op_age, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob
## Model 2: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob + orient_precio:CuotaEdad
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       709     900.64
## 2       707     887.08  2   13.569 0.001131 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(modelo_demo, mod_int_gra_mun, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob
## Model 2: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob + gratificacion:TamPob
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       709     900.64
## 2       707     890.06  2   10.583 0.005035 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(modelo_demo, mod_int_op_sex, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob
## Model 2: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob + orient_precio:S2
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       709     900.64
## 2       708     893.69  1   6.9529 0.008368 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

7.4 Probar interacciones entre variables clave

```r
#    - confianza_bin x culpa
#    - servicio_bin x orient_precio
#    - servicio_bin x gratificacion
#    - gratificacion x orient_precio
#    - orient_precio x culpa
mod_int_conf_culp <- update(modelo_demo, . ~ . + confianza_bin:culpa)
mod_int_serv_prec <- update(modelo_demo, . ~ . + servicio_bin:orient_precio)
mod_int_serv_gra  <- update(modelo_demo, . ~ . + servicio_bin:gratificacion)
mod_int_gra_prec  <- update(modelo_demo, . ~ . + gratificacion:orient_precio)
mod_int_prec_culp <- update(modelo_demo, . ~ . + orient_precio:culpa)

# Comparar AIC y Chi-cuadrado
modelos_key_int <- list(
  demo            = modelo_demo,
  conf_x_culp     = mod_int_conf_culp,
  serv_x_prec     = mod_int_serv_prec,
  serv_x_gra      = mod_int_serv_gra,
  gra_x_prec      = mod_int_gra_prec,
  prec_x_culp     = mod_int_prec_culp
)
sapply(modelos_key_int, AIC)
```

```
##        demo conf_x_culp serv_x_prec  serv_x_gra  gra_x_prec prec_x_culp
##    922.6440    923.2155    924.2523    924.2083    924.5149    922.3084
```

```r
anova(modelo_demo, mod_int_conf_culp, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob
## Model 2: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob + confianza_bin:culpa
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       709     900.64
## 2       708     899.22  1   1.4284    0.232
```

```r
anova(modelo_demo, mod_int_serv_prec,  test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob
## Model 2: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob + servicio_bin:orient_precio
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       709     900.64
## 2       708     900.25  1  0.39162   0.5314
```

```r
anova(modelo_demo, mod_int_serv_gra,   test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob
## Model 2: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob + servicio_bin:gratificacion
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       709     900.64
## 2       708     900.21  1  0.43568   0.5092
```

```r
anova(modelo_demo, mod_int_gra_prec,   test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob
## Model 2: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob + gratificacion:orient_precio
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       709     900.64
## 2       708     900.51  1  0.12902   0.7195
```

```r
anova(modelo_demo, mod_int_prec_culp,  test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob
## Model 2: SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob + orient_precio:culpa
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
```

```
## 1       709     900.64
## 2       708     898.31  1   2.3356   0.1265
```

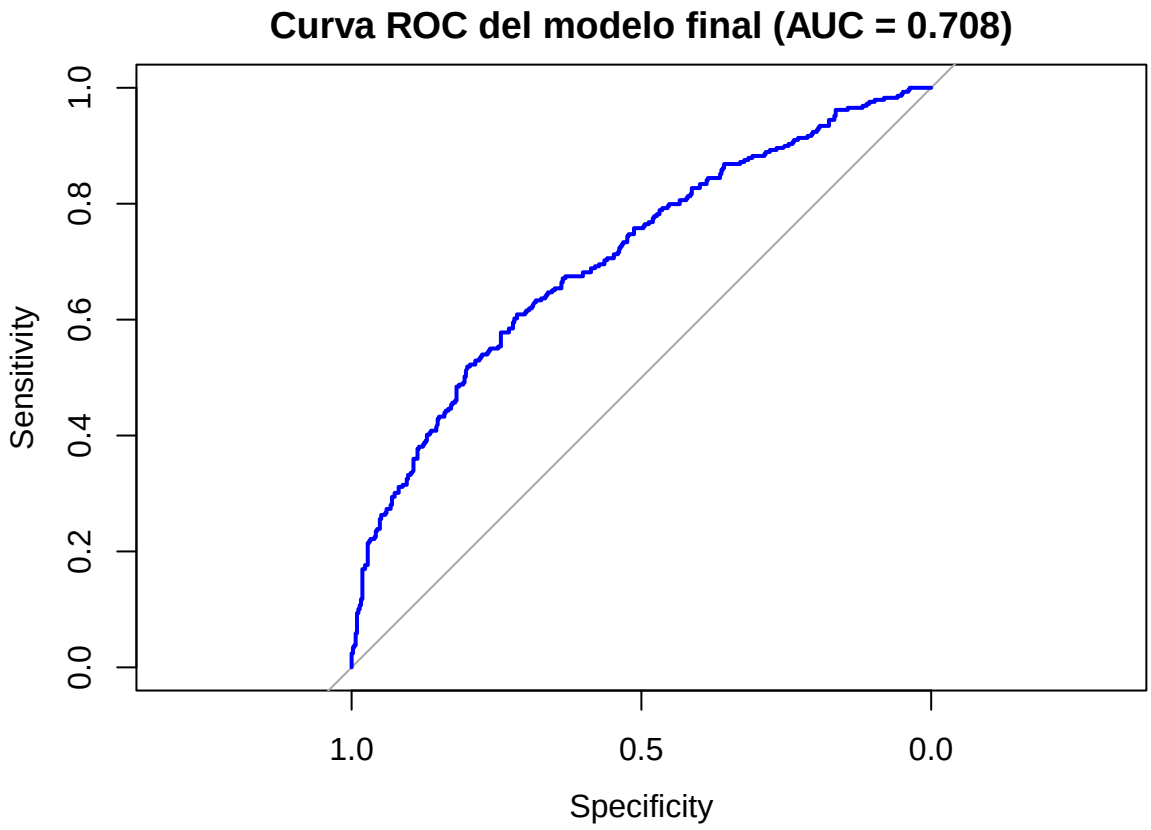
7.5 Modelo Final

```
modelo_final <- glm(
  SHOWROOMING ~ confianza_bin + servicio_bin +
                gratificacion + orient_precio + culpa +
                CuotaEdad + S2 + TamPob +
                orient_precio:CuotaEdad +
                gratificacion:TamPob +
                orient_precio:S2,
  family = binomial(link="logit"),
  data   = df_clean
)

summary(modelo_final)
```

```
##
## Call:
## glm(formula = SHOWROOMING ~ confianza_bin + servicio_bin + gratificacion +
##     orient_precio + culpa + CuotaEdad + S2 + TamPob + orient_precio:CuotaEdad +
##     gratificacion:TamPob + orient_precio:S2, family = binomial(link = "logit"),
##     data = df_clean)
##
## Coefficients:
##                                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)                          5.84220    2.17782   2.683 0.007306 **
## confianza_bin                       -0.24386    0.16375  -1.489 0.136425
## servicio_bin                         0.22729    0.16384   1.387 0.165363
## gratificacion                       -1.06786    0.37433  -2.853 0.004334 **
## orient_precio                        0.18648    0.20172   0.924 0.355252
## culpa                               -0.33243    0.05459  -6.090 1.13e-09 ***
## CuotaEdad31-50                      -1.16726    1.14713  -1.018 0.308893
## CuotaEdad51-65                      -4.57388    1.33151  -3.435 0.000592 ***
## S2Femenino                           1.95606    0.90969   2.150 0.031536 *
## TamPob100.001-500.000               -6.58705    2.09797  -3.140 0.001691 **
## TamPob>500.000                      -4.80944    1.97453  -2.436 0.014861 *
## orient_precio:CuotaEdad31-50         0.20024    0.21300   0.940 0.347167
## orient_precio:CuotaEdad51-65         0.73291    0.23651   3.099 0.001943 **
## gratificacion:TamPob100.001-500.000  1.18775    0.40516   2.932 0.003373 **
## gratificacion:TamPob>500.000         0.85111    0.38116   2.233 0.025553 *
## orient_precio:S2Femenino            -0.40258    0.16410  -2.453 0.014159 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 969.94  on 719  degrees of freedom
## Residual deviance: 870.77  on 704  degrees of freedom
## AIC: 902.77
##
## Number of Fisher Scoring iterations: 4
```

8 BONDAD DEL AJUSTE DEL MODELO FINAL

```
# 1) Obtener probabilidades y vector 0/1
df_clean$prob_final <- predict(modelo_final, type = "response")
df_clean$show_bin_num <- ifelse(df_clean$SHOWROOMING == "Sí", 1, 0)

# 2) Curva ROC y AUC
library(pROC)
roc_obj <- roc(
  response = df_clean$show_bin_num,
  predictor = df_clean$prob_final
)
plot(roc_obj, col  = "blue",
  lwd  = 2,
  main = paste0(
    "Curva ROC del modelo final (AUC = ",
    round(auc(roc_obj), 3),
    ")"
  )
)
```




```
auc_val <- auc(roc_obj)

cat("AUC =", round(auc_val, 4), "\n")
```

```
## AUC = 0.7079
```

```r
# 3) Hosmer-Lemeshow goodness-of-fit
hl <- hoslem.test(
  x = df_clean$show_bin_num,
  y = df_clean$prob_final,
  g = 10
)
print(hl)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  df_clean$show_bin_num, df_clean$prob_final
## X-squared = 10.73, df = 8, p-value = 0.2175
```

```r
# 4) Matriz de confusión
# Definir clase predicha con umbral 0.5
df_clean$pred_class <- factor(
  ifelse(df_clean$prob_final >= 0.5, "Sí", "No"),
  levels = c("No","Sí")
)
CrossTable(
  x         = df_clean$show_bin_num,
  y         = df_clean$pred_class,
  prop.chisq= FALSE,
  prop.t    = FALSE,
  prop.r    = FALSE,
  dnn       = c("Actual","Predicho")
)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  720
##
##
##              | Predicho
##       Actual |        No |        Sí | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       357 |        74 |       431 |
##              |     0.692 |     0.363 |           |
## -------------|-----------|-----------|-----------|
##            1 |       159 |       130 |       289 |
##              |     0.308 |     0.637 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       516 |       204 |       720 |
##              |     0.717 |     0.283 |           |
## -------------|-----------|-----------|-----------|
##
```

```
##
# 5) Métricas (accuracy, recall, precision, F1)
cm <- table(
  Actual   = df_clean$show_bin_num,
  Predicho = ifelse(df_clean$pred_class=="Sí", 1, 0)
)
TN <- cm["0","0"]; FP <- cm["0","1"]
FN <- cm["1","0"]; TP <- cm["1","1"]

accuracy    <- (TP + TN) / sum(cm)
sensitivity <- TP / (TP + FN)
specificity <- TN / (TN + FP)
precision   <- TP / (TP + FP)
f1_score    <- 2 * precision * sensitivity / (precision + sensitivity)

metrics <- data.frame(
  Metric        = c("Accuracy","Sensitivity","Specificity","Precision","F1 Score"),
  Value         = c(accuracy, sensitivity, specificity, precision, f1_score)
)
print(metrics)
```

```
##        Metric     Value
## 1    Accuracy 0.6763889
## 2 Sensitivity 0.4498270
## 3 Specificity 0.8283063
## 4   Precision 0.6372549
## 5    F1 Score 0.5273834
```

9 INTERPRETACIÓN DE RESULTADOS

9.1 Global

```
# 1) Lista de continuas
vars_cont <- c("gratificacion", "orient_precio", "culpa")

# 2) Probabilidades base
p_base <- predict(modelo_final, type = "response")

# 3) Para cada variable, subimos en +1 y computamos p2
ame_cont <- sapply(vars_cont, function(var){
  newdata <- df_clean
  newdata[[var]] <- newdata[[var]] + 1
  p2 <- predict(modelo_final, newdata = newdata, type = "response")
  mean(p2 - p_base)
})

ame_cont
```

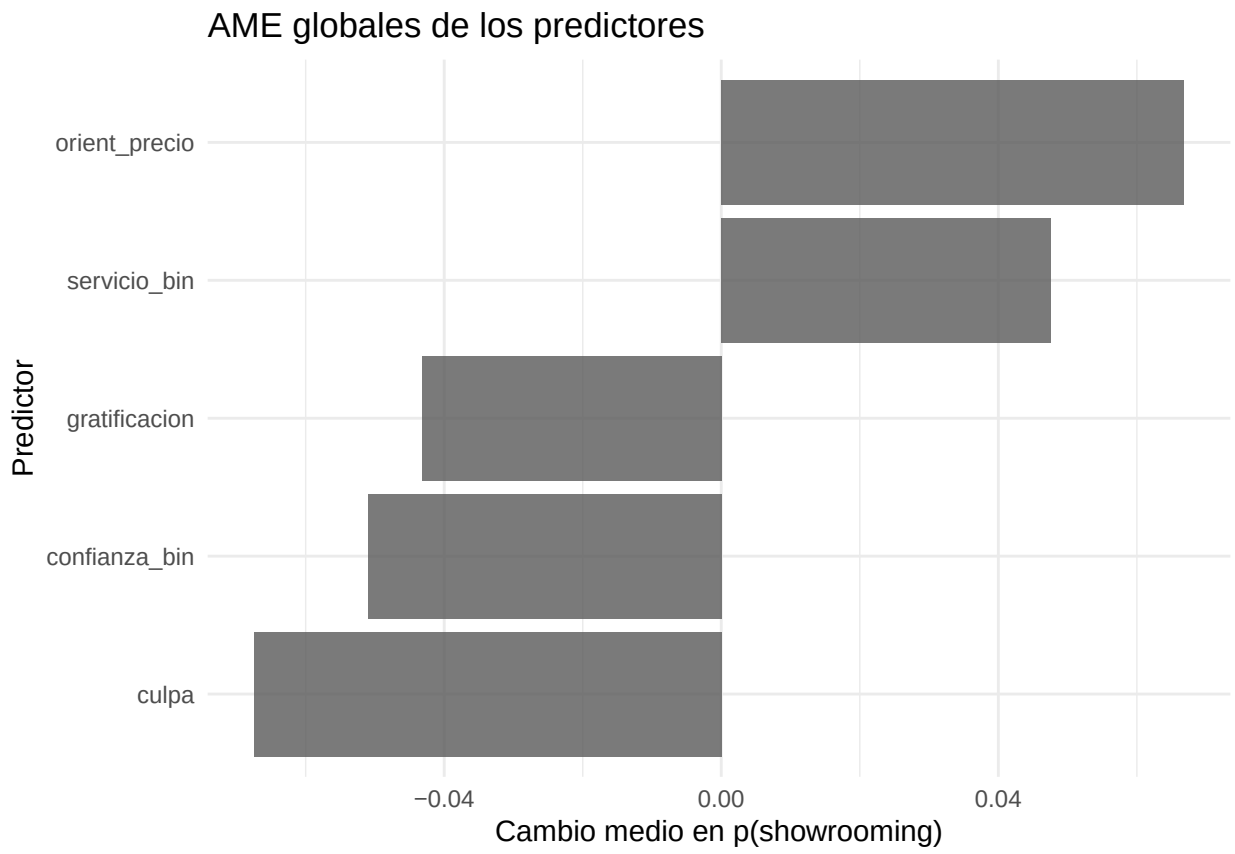
```
## gratificacion orient_precio         culpa
##   -0.04322129    0.06679170   -0.06744264
```

```
vars_bin <- c("confianza_bin","servicio_bin")

ame_bin <- sapply(vars_bin, function(var){
```

```
  new1 <- df_clean; new1[[var]] <- 1
  new0 <- df_clean; new0[[var]] <- 0
  p1 <- predict(modelo_final, newdata = new1, type = "response")
  p0 <- predict(modelo_final, newdata = new0, type = "response")
  mean(p1 - p0)
})

ame_bin
```

```
## confianza_bin  servicio_bin
##   -0.05104284    0.04760867
```

```
df_global <- tibble(
  predictor = c(names(ame_cont), names(ame_bin)),
  ame       = c(ame_cont, ame_bin)
)

ggplot(df_global, aes(x = reorder(predictor, ame), y = ame)) +
  geom_col(alpha = 0.8) +
  coord_flip() +
  labs(
    title = "AME globales de los predictores",
    x     = "Predictor",
    y     = "Cambio medio en p(showrooming)"
  ) +
  theme_minimal()
```

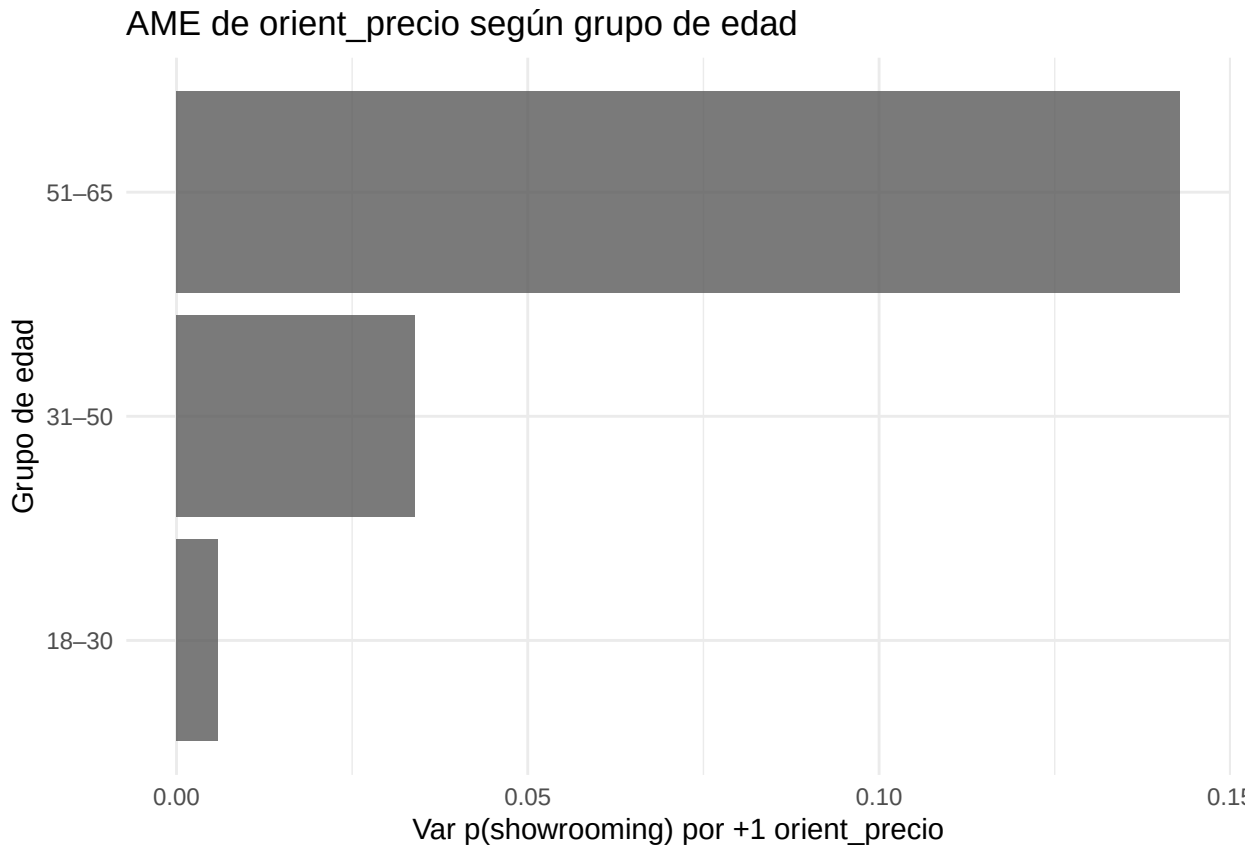


9.2 Orientación al precio x Grupo de edad

```
niv_edad <- levels(df_clean$CuotaEdad)
ame_op_by_age <- sapply(niv_edad, function(lv){
  sub <- df_clean; sub <- sub[sub$CuotaEdad == lv, ]
  p0 <- predict(modelo_final, newdata = sub, type = "response")
  sub2 <- sub; sub2$orient_precio <- sub2$orient_precio + 1
  p1 <- predict(modelo_final, newdata = sub2, type = "response")
  mean(p1 - p0)
})
ame_op_by_age
```

```
##       18-30       31-50       51-65
## 0.005863655 0.033944510 0.142860708
```

```
# Orient_precio x Edad
df_op_age <- tibble(
  CuotaEdad = niv_edad,
  ame       = ame_op_by_age
)

ggplot(df_op_age, aes(x = CuotaEdad, y = ame))  +
  geom_col(alpha = 0.8) +
  coord_flip() +
  labs(
    title = "AME de orient_precio según grupo de edad",
    x     = "Grupo de edad",
    y     = "Var p(showrooming) por +1 orient_precio"
  ) +
  theme_minimal()
```



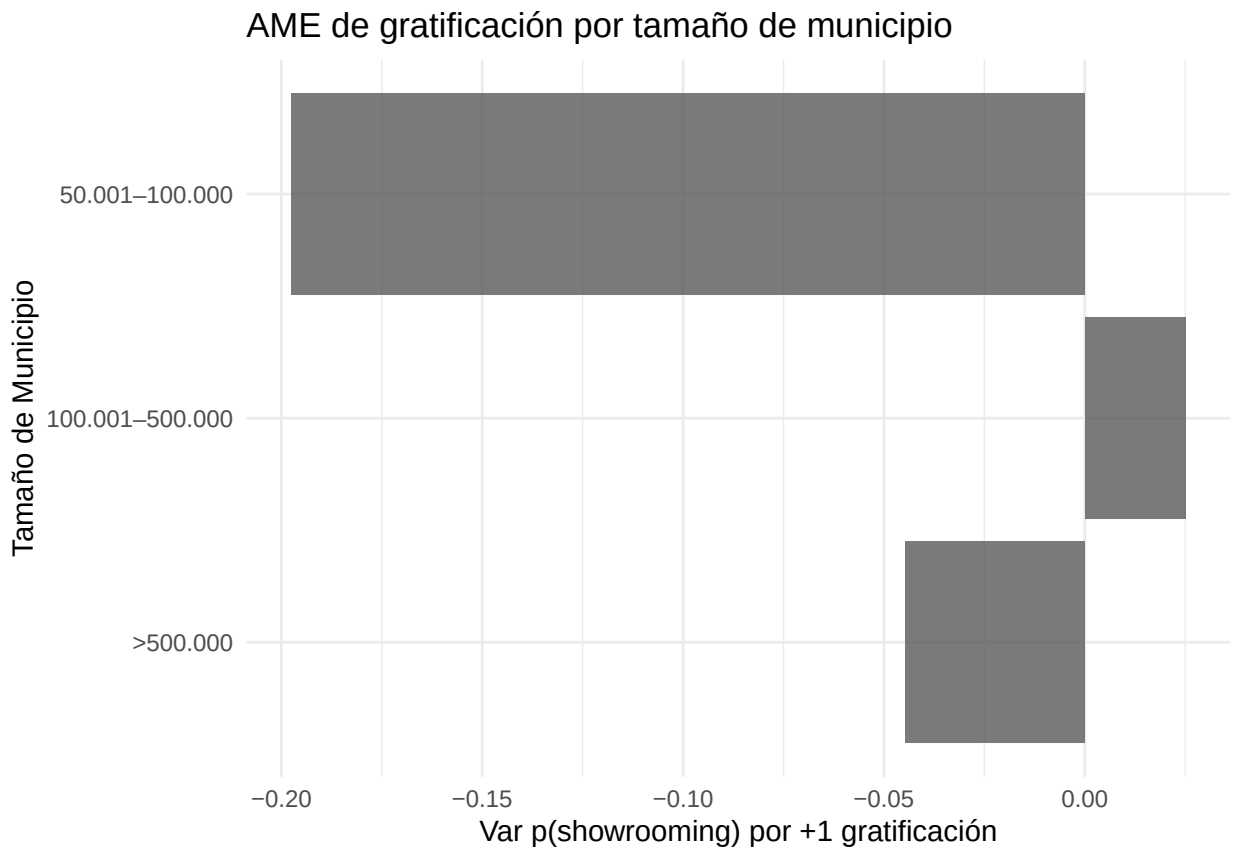
9.3 Gratificación x Tamaño de municipio

```
mun_counts <- table(df_clean$TamPob)
niveles_presentes <- names(mun_counts)[mun_counts > 0]
ame_gr_by_mun <- sapply(niveles_presentes, function(lv) {
  sub0 <- df_clean[df_clean$TamPob == lv, ]
  # Probabilidad base
  p0 <- predict(modelo_final, newdata = sub0, type = "response")
  # Subimos gratificación en +1 y predecimos de nuevo
  sub1 <- sub0
  sub1$gratificacion <- sub1$gratificacion + 1
  p1 <- predict(modelo_final, newdata = sub1, type = "response")
  # AME para ese nivel
  mean(p1 - p0)
})

ame_gr_by_mun
```

```
##  50.001-100.000 100.001-500.000        >500.000
##     -0.19758349      0.02511274     -0.04481346
```

```
df_gr_mun <- tibble(
  TamPob = niveles_presentes,
  ame    = ame_gr_by_mun
)
```

```
ggplot(df_gr_mun, aes(x = TamPob, y = ame)) +
  geom_col(alpha = 0.8) +
  coord_flip() +
  labs(
    title = "AME de gratificación por tamaño de municipio",
    x     = "Tamaño de Municipio",
    y     = "Var p(showrooming) por +1 gratificación"
  ) +
  theme_minimal()
```



Orientación al precio x Sexo

```
niv_sex <- levels(df_clean$S2)
ame_op_by_sex <- sapply(niv_sex, function(lv){
  sub <- df_clean; sub <- sub[sub$S2 == lv, ]
  p0 <- predict(modelo_final, newdata = sub, type = "response")
  sub2 <- sub; sub2$orient_precio <- sub2$orient_precio + 1
  p1 <- predict(modelo_final, newdata = sub2, type = "response")
  mean(p1 - p0)
})
ame_op_by_sex
```

```
##  Masculino   Femenino
## 0.11004765 0.02353574
```

```
# Orient_precio x Sexo
df_op_sex <- tibble(
  Sexo = niv_sex,
```

```
  ame  = ame_op_by_sex
)

ggplot(df_op_sex, aes(x = Sexo, y = ame)) +
  geom_col(alpha = 0.8) +
  coord_flip() +
  labs(
    title = "AME de orient_precio según sexo",
    x     = "Sexo",
    y     = "Var p(showrooming) por +1 orient_precio"
  ) +
  theme_minimal()
```