Modelado Estadístico: Regresión Ordinal

Barragán (LU: 1472/21), Fontana (LU: 1530/21), Gandolfo (LU: 169/21).

2025-06-07

Trabajo Práctico 1

1.

Filtramos nulos y valores sin sentido de la columna age, que se usará más adelante.

```
datos_train <- datos_train %>%
  filter(!is.na(Q43), Q43 %in% 1:5, !is.na(Q9), !is.na(age)) %>%
  mutate(age_numeric = as.numeric(as.character(age))) %>%
  filter(age_numeric >= 13, age_numeric <= 100) %>%
  mutate(Q43_ord = factor(Q43, levels = 1:5, ordered = TRUE))

datos_test <- datos_test %>%
  filter(!is.na(Q43), Q43 %in% 1:5, !is.na(Q9), !is.na(age)) %>%
  mutate(age_numeric = as.numeric(as.character(age))) %>%
  filter(age_numeric >= 13, age_numeric <= 100) %>%
  mutate(Q43_ord = factor(Q43, levels = 1:5, ordered = TRUE))
```

2.

Elegimos la pregunta 43:

Q43: I think a natural disaster would be kind of exciting.

3.

La regresión lineal asume que la variable de respuesta es continua y no acotada en $\mathbb{R}$. En cambio, Q (por ejemplo, Q_{43}) es una escala Likert ordinal con valores enteros $\{1, 2, 3, 4, 5\}$, por lo que un modelo lineal puede predecir valores no enteros o fuera del rango $[1, 5]$ (por ejemplo, 5.8 o 0.3), lo cual no tendría sentido.

El modelo multinomial (para distribución categórica con $K = 5$ posibles resultados) estima $K - 1 = 4$ ecuaciones de log-odds independientes:

$$\log(\text{odds}(Y = k \mid X)) \;=\; \beta_{0k} + \beta_{1k} X_1 + \cdots, \quad k = 2, 3, 4, 5,$$

tomando, por ejemplo, la categoría 1 como referencia. Este planteo ignora completamente el orden intrínseco entre las categorías $1 < 2 < 3 < 4 < 5$. Tratar un "5" como clase independiente no aprovecha que un "5" es "más cercano" a un "4" que a un "1". Como no se impone ningún criterio de orden, puede dar probabilidades que no respeten la secuencia, lo que hace difícil interpretarlo cuando sí hay un sentido creciente entre niveles.

4.

La regresión ordinal resuelve los problemas anteriores aprovechando el orden de las categorías al modelar la probabilidad acumulada de que la respuesta caiga "en o por debajo" de cada nivel, en lugar de tratar cada categoría como independiente. Con un solo conjunto de coeficientes, el modelo estima cómo las variables predictoras desplazan ese puntaje hacia niveles más bajos o más altos. Así, las probabilidades resultantes siempre respetan el orden natural: si un predictor se modifica, la probabilidad acumulada de estar en categorías superiores crece/disminuye de manera ordenada, sin saltos incoherentes entre niveles.

5.

Primero entrenamos el modelo.

```
modelo_ordinal <- polr(Q43_ord ~ age_numeric, data = datos_train, Hess = TRUE)

summary(modelo_ordinal)
```

```
## Call:
## polr(formula = Q43_ord ~ age_numeric, data = datos_train, Hess = TRUE)
##
## Coefficients:
##                Value Std. Error t value
## age_numeric -0.02113  0.0004392  -48.12
##
## Intercepts:
##     Value    Std. Error t value
## 1|2  -1.0399   0.0111   -93.4350
## 2|3  -0.4683   0.0109   -42.7698
## 3|4   0.0773   0.0109     7.0742
## 4|5   1.2140   0.0115   105.9068
##
## Residual Deviance: 674882.48
## AIC: 674892.48
```

El coeficiente es negativo y muy pequeño en valor absoluto (-0.021). Eso significa que, a medida que la edad aumenta en 1 año, la probabilidad de pertenecer a categorías "más altas" disminuye muy suavemente.

Evaluamos el modelo en nuestro conjunto de test.

```
pred_test_ordinal <- predict(modelo_ordinal,
                     newdata = datos_test,
                     type    = "class")
#matriz de confusion
cm <- table(Predicho = pred_test_ordinal, Real = datos_test$Q43_ord)
print(cm)
```

```
##         Real
## Predicho     1     2     3     4     5
##        1 34945 13225 12547 19515 14951
##        2     0     0     0     0     0
##        3     0     0     0     0     0
##        4     0     0     0     0     0
##        5     0     0     0     0     0
```

```r
pred_num <- as.numeric(as.character(pred_test_ordinal))
real_num <- as.numeric(as.character(datos_test$Q43_ord))
accuracy_num <- mean(pred_num == real_num)
cat("Accuracy (comparando como números):", accuracy_num, "\n")
```

```
## Accuracy (comparando como números): 0.3671349
```

Dado que las edades oscilan entre 13 y 100 (filtrado propio basándonos en la información de la encuesta), ese cambio tan gradual apenas mueve las probabilidades: para prácticamente toda la franja de edades, la categoría 1 sigue siendo la más probable. Por eso, al pedirle `type="class"`, el modelo "elige siempre 1" como la categoría de máxima probabilidad, aun para edades grandes o chicas.

```r
real_num <- as.numeric(as.character(datos_test$Q43_ord))
pred_num <- as.numeric(as.character(pred_test_ordinal))

MAE <- mean(abs(pred_num - real_num))
cat("Mean Absolute Error (MAE) en el test set:", MAE, "\n")
```

```
## Mean Absolute Error (MAE) en el test set: 1.645966
```

Un MAE de ~1.65, junto a que la mayoría de las predicciones quedaron en "1", sugiere que la edad por sí sola tiene muy poca capacidad para diferenciar las respuestas 1–5.

6.

Para estimar esa probabilidad, primero filtramos el conjunto de datos para quedarnos solo con las personas que tienen `age_numeric == 25` y un valor válido en Q9; luego contamos cuántas de esas personas tienen Q9 4 ("al menos de acuerdo") y dividimos ese conteo por el total de la submuestra de 25 años; la proporción resultante se usa como estimador puntual de la probabilidad buscada.

```r
sub25 <- datos_train %>%
  filter(age_numeric == 25, !is.na(Q9))

prob_25_acuerdo <- mean(sub25$Q9 >= 4, na.rm = TRUE)
prob_25_acuerdo
```

```
## [1] 0.346872
```

7.

```r
loss_L <- function(y, y_pred) {
  y_pred_cat <- pmin(pmax(round(y_pred), 1), 5)
  mean(abs(y - y_pred_cat))
}
```

8.

Entrenamos el modelo lineal.

```
y_train_num   <- as.integer(datos_train$Q43_ord)
modelo_lineal <- lm(y_train_num ~ age_numeric, data = datos_train)

pred_cont <- predict(modelo_lineal, newdata = datos_test)

# redondeamos al rango deseado
pred_cat <- pmin(pmax(round(pred_cont), 1), 5)
```

9.

```
y_test_num <- as.integer(datos_test$Q43_ord)

# perdida del modelo ordinal
pred_test_ordinal <- predict(modelo_ordinal, newdata = datos_test, type = "class")
pred_ord_num      <- as.numeric(as.character(pred_test_ordinal))
loss_ord          <- loss_L(y_test_num, pred_ord_num)

# ajustar y predecir con el modelo lineal
y_train_num   <- as.integer(datos_train$Q43_ord)
modelo_lm     <- lm(y_train_num ~ age_numeric, data = datos_train)
pred_lm_cont  <- predict(modelo_lm, newdata = datos_test)
loss_lm       <- loss_L(y_test_num, pred_lm_cont)


cat("Pérdida L (ordinal):", loss_ord, "\n")
```

```
## Pérdida L (ordinal): 1.645966
```

```
cat("Pérdida L (lineal+round):", loss_lm, "\n")
```

```
## Pérdida L (lineal+round): 1.37244
```

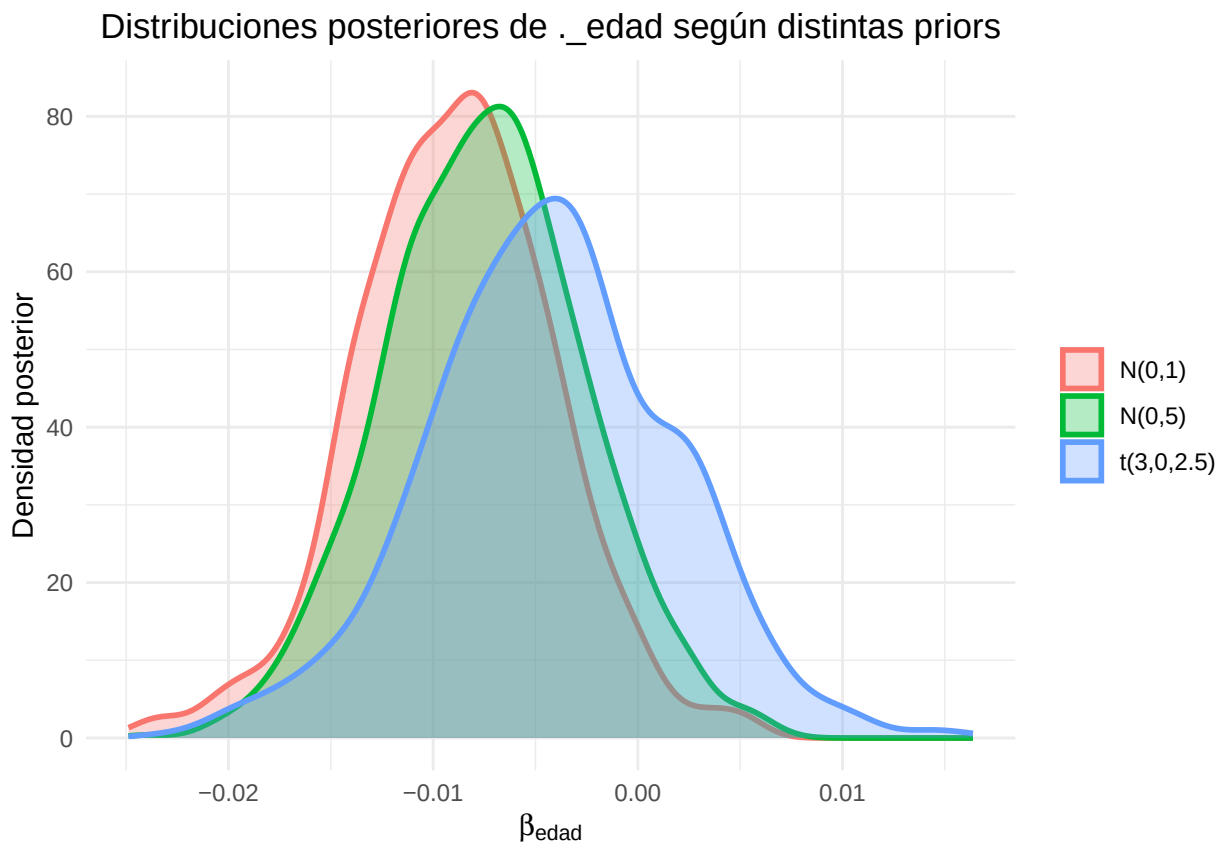
Que el modelo lineal (con redondeo) obtiene una pérdida L=1.3724, que es menor que la del modelo ordinal (L=1.6460). En otras palabras, en promedio la predicción entera del modelo lineal queda 1.37 categorías lejos de la real, mientras que la del ordinal se queda 1.65 categorías fuera. Por lo tanto, bajo la función de pérdida L definida, el modelo lineal + redondeo es más preciso y, en este sentido, preferible al modelo ordinal.

#10.

```
library(ggplot2)
ggplot(posterior_all, aes(x = beta_age, color = prior, fill = prior)) +
  geom_density(alpha = 0.3, size = 1) +
  labs(
    x     = expression(beta[edad]),
```

```
    y     = "Densidad posterior",
    title = "Distribuciones posteriores de  _edad según distintas priors"
  ) +
  theme_minimal() +
  theme(
    plot.title   = element_text(hjust = 0.5),
    legend.title = element_blank()
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Se podría concluir que la posterior de los β casi no cambia al variar la prior, lo que indica que la información de los datos es tan fuerte que domina el resultado y la prior apenas influye.