

# Introducción al modelado con R

## Curso Básico R

CEPAL - Unidad de Estadísticas Sociales

2025-10-27

# Tabla de contenidos I

Introducción

Sección 2: Familia de modelos y modelo ajustado

# Introducción

# El Modelo: Señal y Ruido

El modelado es una herramienta para la Exploración de Datos (EDA).

## *Definición Fundamental*

- ▶ Un **modelo** es una representación simplificada de la realidad que captura patrones relevantes (señal) y deja fuera la variabilidad irrelevante (ruido).
- ▶ Propósitos: **describir, resumir y predecir.**

 George Box:

“Todos los modelos están equivocados, algunos son útiles.”

# Definición y propósito

Su propósito es distinguir:

- ▶ Señal ( $f(X)$ ): Los patrones verdaderos generados por el fenómeno de interés.
- ▶ Ruido ( $\epsilon$ ): Las variaciones aleatorias o el error que no nos interesa.

💡 Ecuación del Modelo:

$$Y = f(X) + \epsilon$$

*En este apartado, nos enfocamos en modelos predictivos (que generan  $\hat{Y}$ ).*

# El Rol Exploratorio del Modelo

El foco en esta sección es usar modelos para *construir intuición* y encontrar patrones sutiles.

## Función en EDA

- ▶ Se utiliza para extraer primero la Señal conocida, y luego examinar lo que queda, es decir, el Ruido ( $\epsilon$ ) o residuales.
- ▶ Si encontramos patrones en los residuales, hemos descubierto una nueva señal que el modelo inicial omitió.

# La Regla Crítica: Exploración vs. Confirmación

Es fundamental distinguir la fase de búsqueda de patrones de la fase de prueba formal.

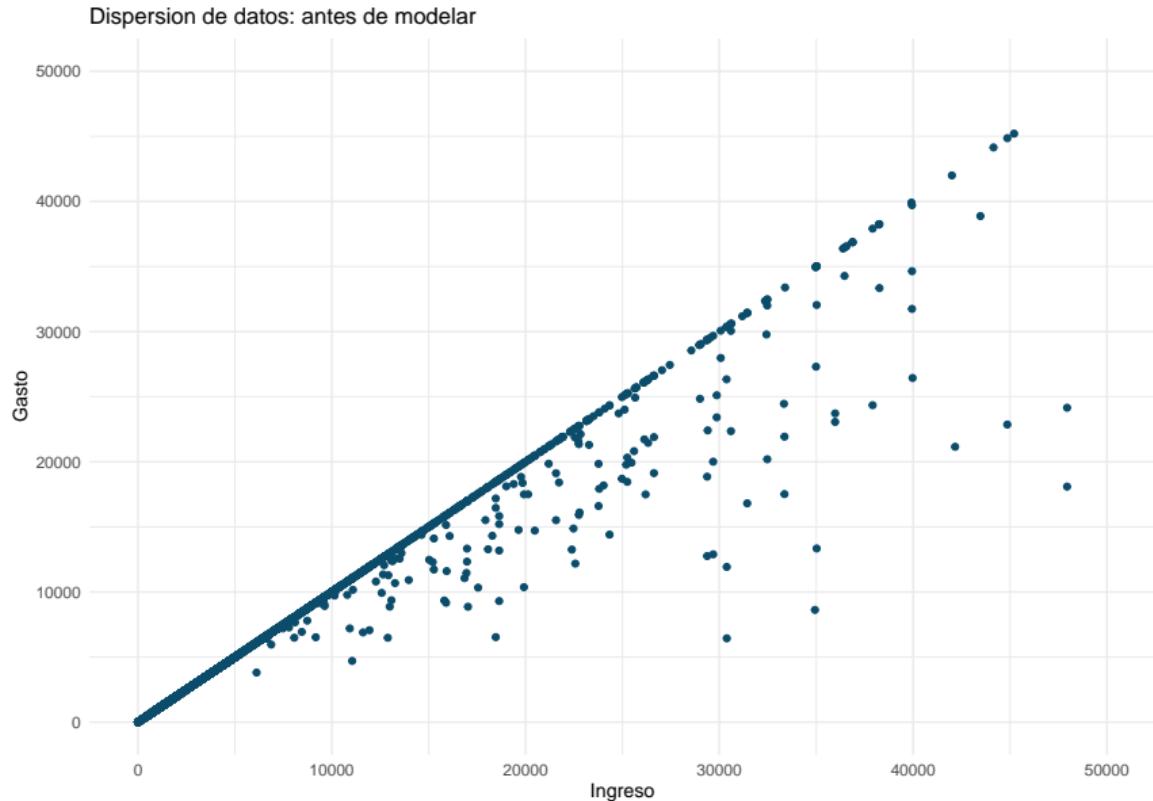
## Principio de la Independencia

- ▶ Exploración (Generación de Hipótesis): Puedes usar un dato muchas veces.
- ▶ Confirmación (Inferencia Formal): Solo puedes usar una observación **UNA SOLA VEZ**.

! Advertencia:

Si usas los mismos datos para generar y confirmar una hipótesis, serás **demasiado optimista**.

# Ejemplo práctico: visualización previa al ajuste



## Sección 2: Familia de modelos y modelo ajustado

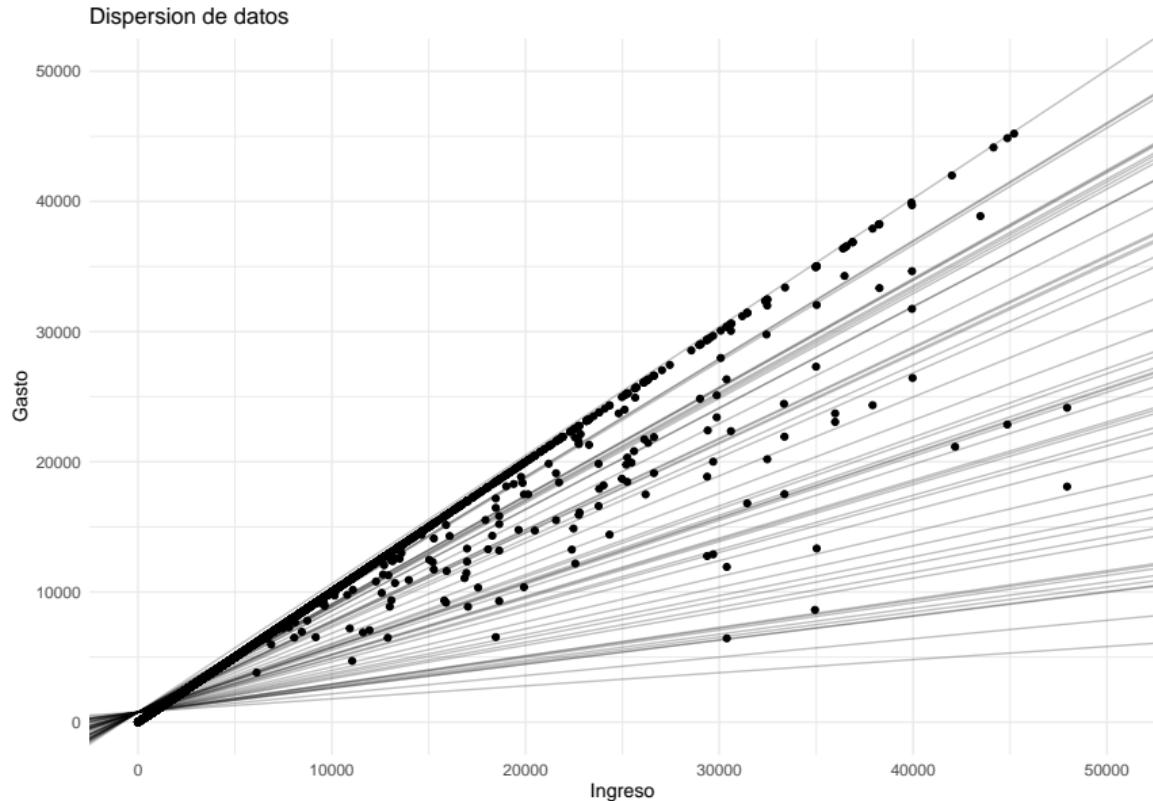
## Concepto: familia vs modelo ajustado

- ▶ **Familia de modelos:** conjunto de funciones posibles (ej. lineales, polinomiales, logísticas).
- ▶ **Modelo ajustado:** parámetros estimados que describen el miembro concreto de la familia.

! Importante:

Seleccionar familia adecuada según pregunta de interés y naturaleza de los datos.

# Modelos posibles



## Flujo de trabajo

- ▶ Flujo práctico: limpieza → visualización → modelado → diagnóstico → comunicación.
- ▶ Presentar la lógica del modelo simple antes de pasar a construcción y comparación.

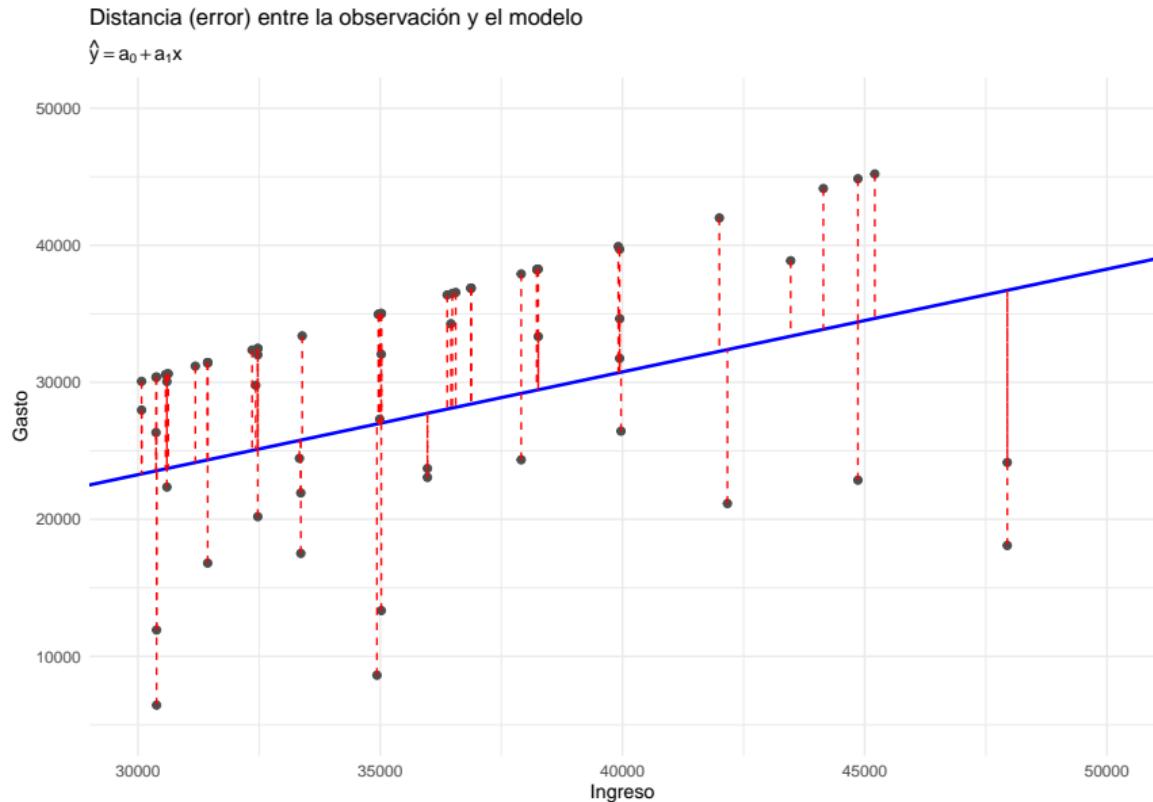
# ¿Cómo medimos qué tan bien se ajusta un modelo?

- ▶ Cada punto del gráfico representa un valor observado  $(x_i, y_i)$ .
- ▶ El modelo genera un valor predicho  $\hat{y}_i = a_0 + a_1 x_i$ .
- ▶ La **distancia vertical** entre el punto real y la recta del modelo es el **error de predicción**:

$$e_i = y_i - \hat{y}_i$$

- ▶ Visualmente, es la diferencia entre el punto y la línea del modelo

# Gráfico de los datos, la recta y los errores individuales



## Código ilustrativo:

```
model1 <- function(a, data) {  
  a[1] + data$ingreso * a[2]  
}  
  
datos$y_pred <- model1(c(762, 0.75), datos)  
datos %>% select(upm, id_hogar, id_pers, gasto,  
                   ingreso, y_pred) %>%  
  head()
```

upm	id_hogar	id_pers	gasto	ingreso	y_pred
1100100006	262	1	5391.527	5391.527	4805.645
1100100006	262	2	5391.527	5391.527	4805.645
1100100006	265	1	7077.083	7077.083	6069.813
1100100006	265	2	7105.557	7105.557	6091.167
1100100006	265	3	7077.083	7077.083	6069.813
1100100006	277	1	2418.750	2418.750	2576.062

## Medir la calidad del ajuste

- ▶ Tenemos 19427 distancias (una por observación).
- ▶ Para resumirlas en un único número, usamos la **raíz del error cuadrático medio (RMSE)**:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

- ▶ RMSE penaliza más los errores grandes y tiene las mismas unidades que la variable dependiente.

# Raíz del Error Cuadrático Medio con R

```
measure_distance <- function(mod, data) {  
  diff <- data$gasto - model1(mod, data)  
  sqrt(mean(diff^2))  
}
```

```
measure_distance(c(a0, a1), datos)
```

```
[1] 2225.108
```

```
datos %>% summarise(RMSE = sqrt(mean((gasto - y_pred)^2)))
```

$$\overline{\overline{\text{RMSE}}}$$
$$\overline{\overline{2225.115}}$$

## Selección del mejor modelo

- ▶ Se evalúan muchos modelos con diferentes parámetros ( $a_1, a_2$ ).
- ▶ Para cada uno se calcula la distancia (RMSE).
- ▶ Los modelos con menor distancia son los que **mejor ajustan los datos**.
- ▶ En el gráfico, los modelos más brillantes son los mejores.

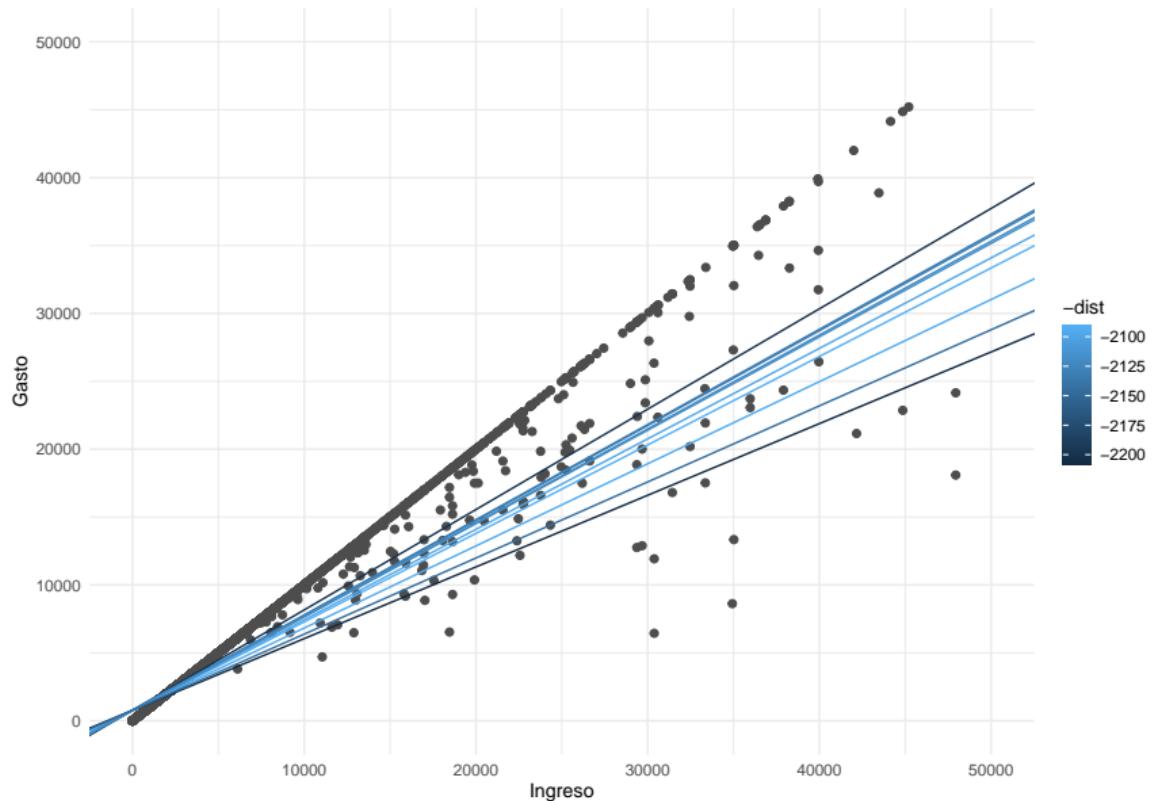
## Selección el modelo con menor RMSE

```
sim1_dist <- function(a1, a2) measure_distance(c(a1, a2), datos)

modelos <- modelos %>%
  mutate(dist = purrr::map2_dbl(a1, a2, sim1_dist),
         orden = rank(dist))
head(modelos)
```

a1	a2	dist	orden
758.6273	0.1412481	3922.724	49
773.6492	0.4979801	2280.083	13
762.2693	0.8190324	2424.019	22
776.4905	0.2097093	3533.166	44
778.2140	0.6048532	2095.540	2
751.3667	0.2858783	3147.861	38

# Los modelos más opcionados



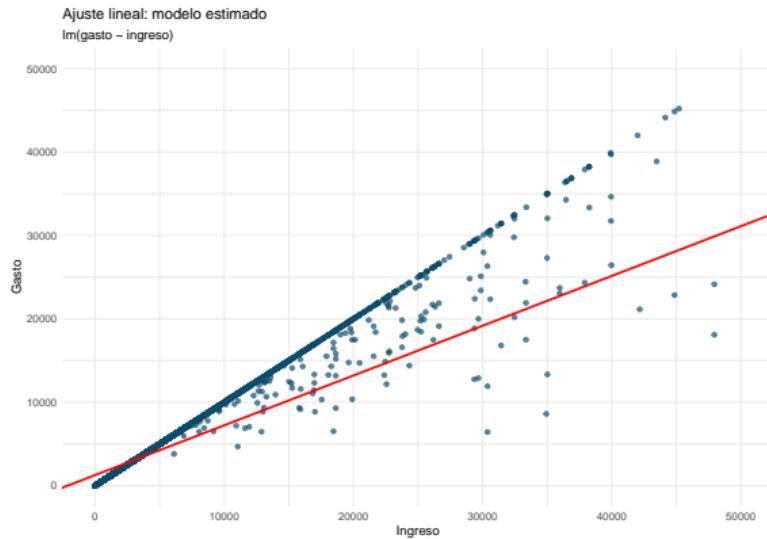
## Código: ajuste lineal y resumen

R cuenta con una herramienta diseñada especialmente para ajustar modelos lineales llamada `lm`

```
library(broom)
mod1 <- lm(gasto ~ ingreso, data = datos)
tidy(mod1)
```

term	estimate	std.error	statistic	p.value
(Intercept)	1272.4609772	17.0469044	74.64469	0
ingreso	0.5968975	0.0025406	234.94441	0

# Visualización del modelo ajustado



## Interpretación final:

*Los modelos con menor distancia representan la mejor combinación de parámetros para describir la relación entre (x) y (y).*

## Predicciones y errores

- ▶ Para modelos simples, podemos analizar el patrón que captura el modelo revisando sus coeficientes ajustados.
- ▶ Un enfoque más práctico es entender el modelo a través de las predicciones que genera sobre los datos.
- ▶ Los *residuos* —las diferencias entre los valores observados y los predichos— muestran lo que el modelo no logra capturar y son esenciales para evaluar la calidad del ajuste.

## Predicciones

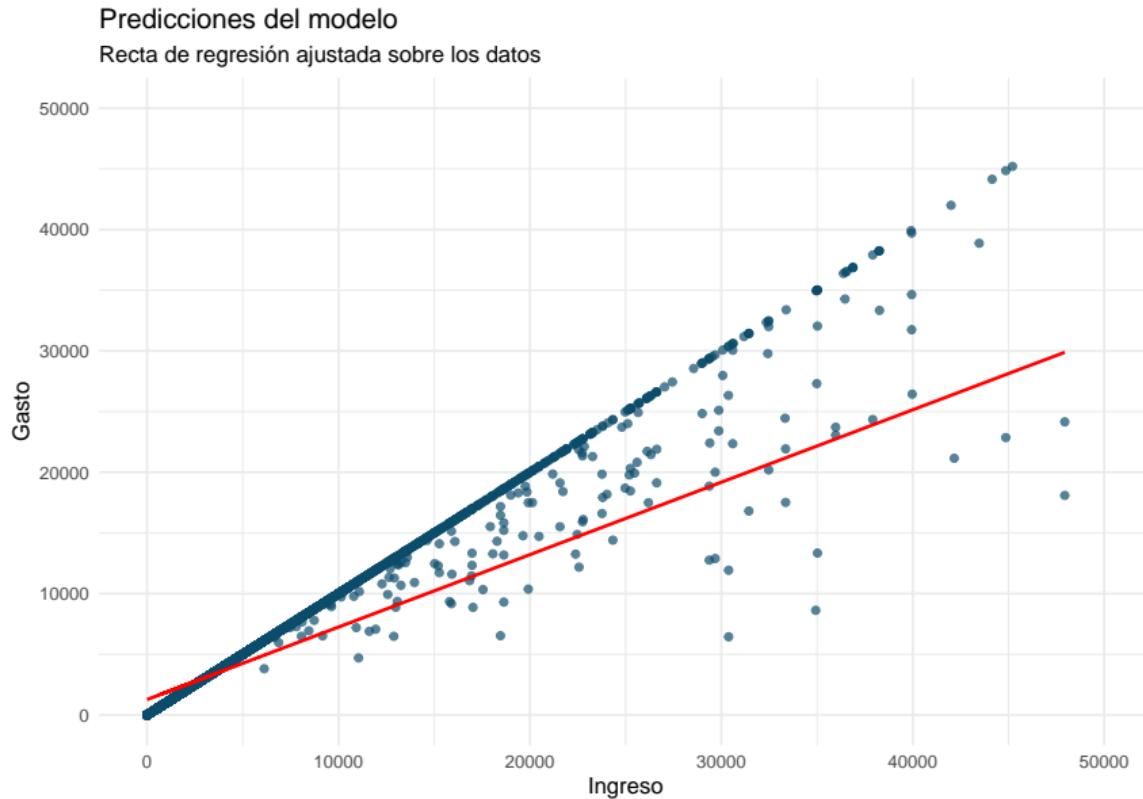
Primero generamos una **grilla de valores** del predictor ingreso y luego calculamos las predicciones del modelo.

```
library(modelr)
datos <- datos %>%
  add_predictions(mod1, var = "pred")

datos %>% select(upm, id_hogar, id_pers, gasto,
                   ingreso, pred) %>% head()
```

upm	id_hogar	id_pers	gasto	ingreso	pred
1100100006	262	1	5391.527	5391.527	4490.650
1100100006	262	2	5391.527	5391.527	4490.650
1100100006	265	1	7077.083	7077.083	5496.754
1100100006	265	2	7105.557	7105.557	5513.750
1100100006	265	3	7077.083	7077.083	5496.754
1100100006	277	1	2418.750	2418.750	2716.207

# Visualización de las predicciones



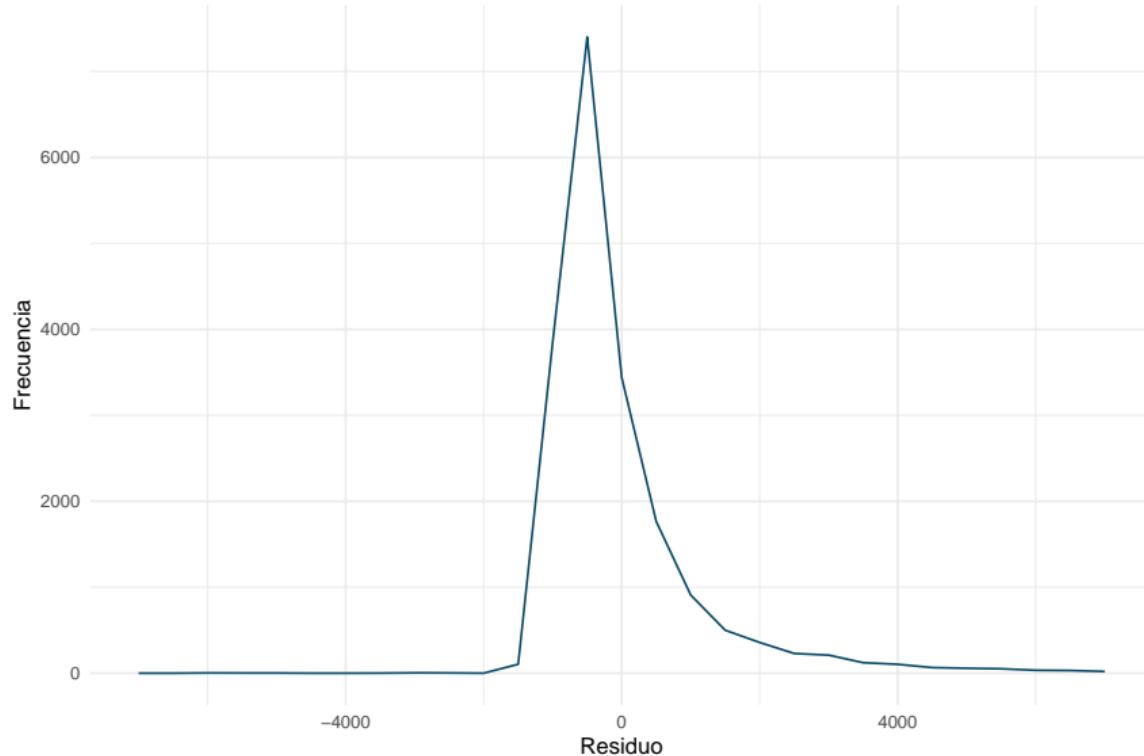
## Residuos

Agregamos los **residuos** (diferencias entre los valores observados y los predichos) al conjunto de datos:

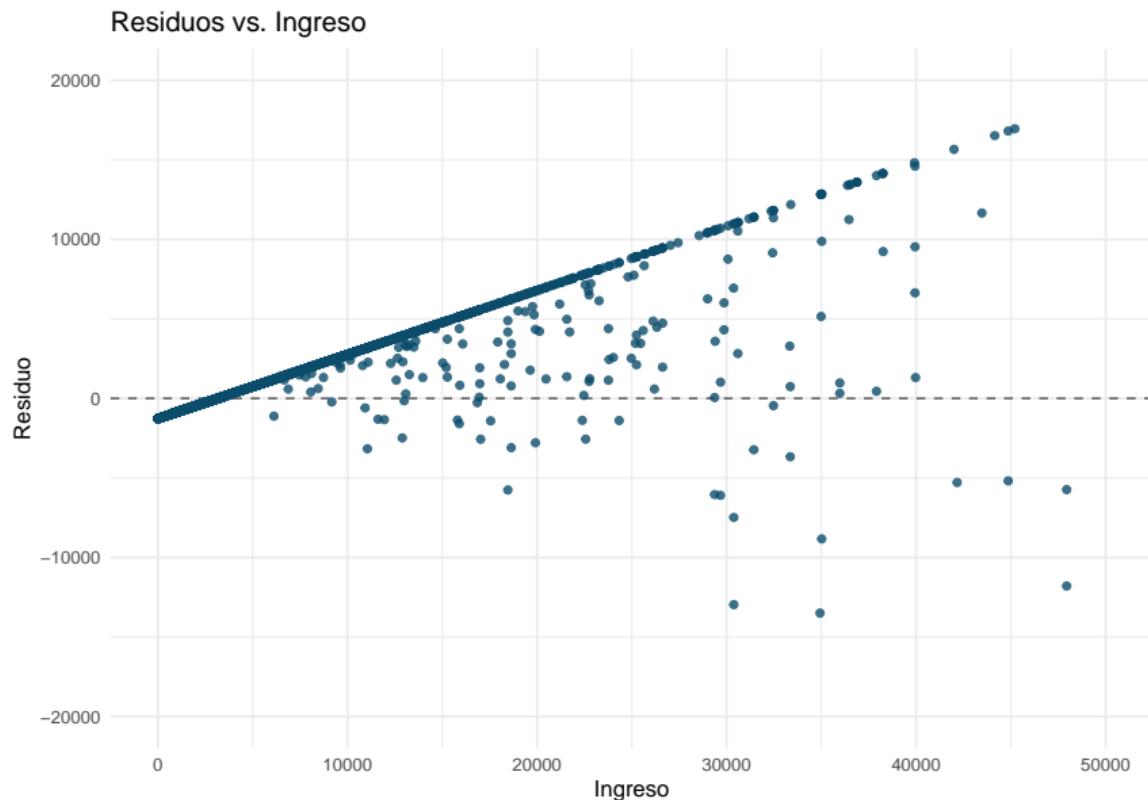
upm	id_hogar	id_pers	gasto	ingreso	pred	resid
1100100006	262	1	5391.527	5391.527	4490.650	900.8770
1100100006	262	2	5391.527	5391.527	4490.650	900.8770
1100100006	265	1	7077.083	7077.083	5496.754	1580.3292
1100100006	265	2	7105.557	7105.557	5513.750	1591.8068
1100100006	265	3	7077.083	7077.083	5496.754	1580.3292
1100100006	277	1	2418.750	2418.750	2716.207	-297.4568

# Distribución de los residuos

Distribución de los residuos



# Residuos vs. Ingreso



## Conclusión

- ▶ Las **predicciones** muestran qué patrón captura el modelo.
- ▶ Los **residuos** revelan lo que el modelo ignora.
- ▶ Residuos distribuidos aleatoriamente alrededor de cero indican un buen ajuste.

## Variables categóricas en modelos lineales

- ▶ Cuando el predictor es **categórico**, no tiene sentido multiplicar como en variables numéricas.
- ▶ Por ejemplo, para la fórmula  $y \sim \text{sexo}$  donde **sexo** puede ser Hombre o Mujer, R crea columnas dummy:

(Intercept)	sexoMujer
1	0
1	1
1	1
1	0
1	0
1	1

! Importante

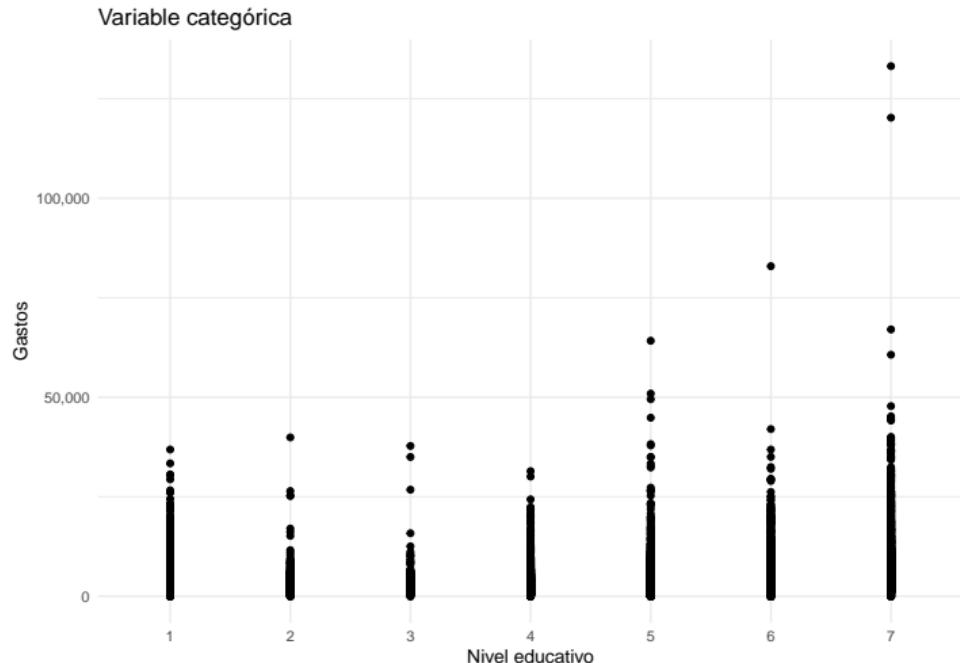
R genera solo **una columna dummy por categoría menos una** para evitar colinealidad:  $\text{sexoMujer} = 0$  si hombre, 1 si mujer.

## Ejemplo

```
library(scales)
library(haven)
datos2 <- datos %>%
  mutate(niveduc = as.character(niveduc_ee),
         niveduc_labels = as_factor(niveduc_ee)) %>%
  filter(!is.na(niveduc)) %>% ungroup()
datos2 %>%
  distinct(niveduc_labels,niveduc) %>% arrange(niveduc)
```

niveduc_labels	niveduc
Primaria incompleta	1
Primaria completa	2
Baja secundaria incompleta	3
Alta secundaria incompleta	4
Secundaria completa	5
Terciaria incompleta	6
Universitaria completa	7

# Visualización del los datos



## Ajustar un modelo lineal:

```
mod2 <- lm(gasto ~ niveduc, data = datos2)
```

```
tidy(mod2)
```

term	estimate	std.error	statistic	p.value
(Intercept)	2465.48541	51.87109	47.5310084	0.0000000
niveduc2	76.04747	121.86553	0.6240277	0.5326168
niveduc3	-26.73777	164.08917	-0.1629466	0.8705622
niveduc4	108.04682	78.36087	1.3788364	0.1679612
niveduc5	614.70302	74.74361	8.2241553	0.0000000
niveduc6	1914.95578	90.04565	21.2664988	0.0000000
niveduc7	6123.54811	112.32822	54.5147787	0.0000000

## Generar predicciones

```
datos2 <- datos2 %>%
  add_predictions(mod2, var = "y_pred")
datos2 %>% distinct(niveduc_labels,
  niveduc, y_pred) %>% head()
```

niveduc_labels	niveduc	y_pred
Universitaria completa	7	8589.034
Secundaria completa	5	3080.188
Terciaria incompleta	6	4380.441
Primaria incompleta	1	2465.485
Alta secundaria incompleta	4	2573.532
Baja secundaria incompleta	3	2438.748

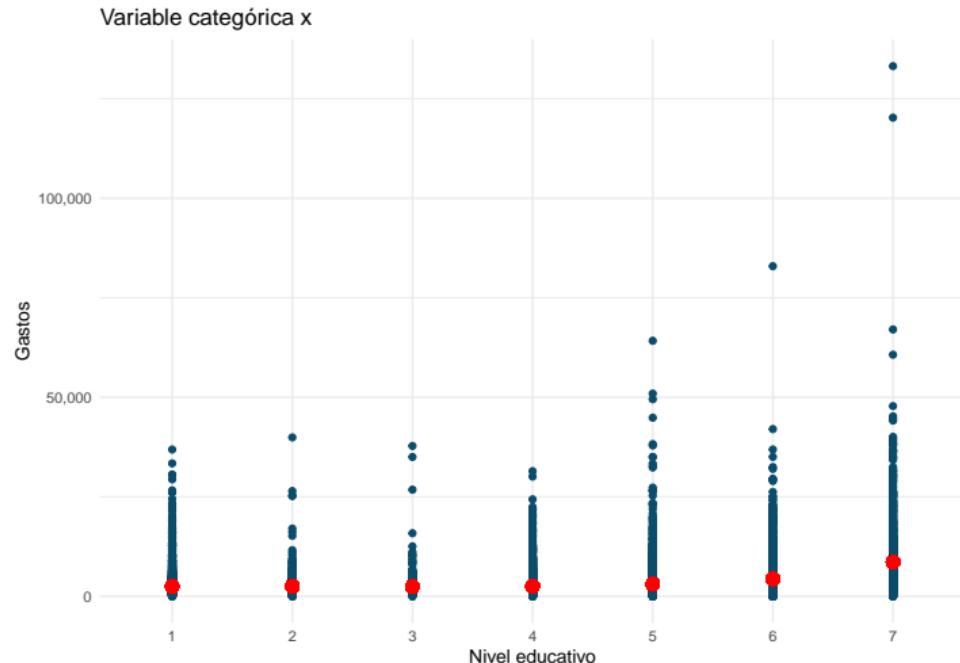
- ▶ El modelo predice el valor medio de  $y$  para cada categoría de  $x$ .
- ▶ No se pueden hacer predicciones para niveles **no observados**:

## Predicciones por categoría

```
datos2 %>% add_residuals(mod2, var = "residual") %>%
  distinct(niveduc_labels, niveduc,
           y_pred, residual) %>% head()
```

niveduc_labels	niveduc	y_pred	residual
Universitaria completa	7	8589.034	-3197.507
Secundaria completa	5	3080.188	2311.338
Universitaria completa	7	8589.034	-1511.950
Terciaria incompleta	6	4380.441	2725.115
Primaria incompleta	1	2465.485	4611.598
Terciaria incompleta	6	4380.441	-1961.691

# Visualización de predicciones por categoría

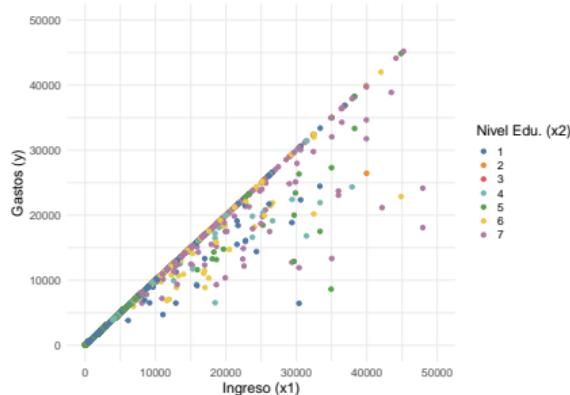


# Interacciones (continuas y categóricas)

Cuando combinamos una **variable continua** con una **variable categórica**, la relación entre la variable continua y la respuesta puede **cambiar entre categorías**.

Para ilustrarlo, usamos el conjunto de datos datos2, que contiene:

- ▶ x1: predictor continuo (ingreso)
- ▶ x2: predictor categórico (nivel educativo)



## Dos modelos posibles (I)

Ajustamos dos modelos lineales:

- $y \sim x_1 + x_2$ : modelo **aditivo**, sin interacción.

```
mod3 <- lm(gasto ~ ingreso + niveduc, data = datos2)
tidy(mod3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	1052.5625772	28.3988835	37.0635197	0.0000000
ingreso	0.5657842	0.0025694	220.1989982	0.0000000
niveduc2	44.1751925	64.9949005	0.6796717	0.4967205
niveduc3	-305.2357125	87.5230744	-3.4874885	0.0004887
niveduc4	50.6018528	41.7931400	1.2107693	0.2259987
niveduc5	241.0768970	39.8992209	6.0421455	0.0000000
niveduc6	814.0840150	48.2837177	16.8604253	0.0000000
niveduc7	2255.2404200	62.4307781	36.1238557	0.0000000

## Dos modelos posibles (II)

$y \sim x_1 * x_2$ : modelo con **interacción**, equivalente a

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_{12} (x_1 \times x_2)$$

```
mod4 <- lm(gasto ~ ingreso * niveduc, data = datos2)
```

term	estimate	std.error	statistic	p.value
(Intercept)	299.2481961	29.2681594	10.224360	0.0000000
ingreso	0.8674379	0.0077458	111.988415	0.0000000
niveduc2	-123.5454403	73.1447709	-1.689054	0.0912253
niveduc3	1597.4542299	74.0028572	21.586386	0.0000000
niveduc4	-68.6239103	46.4236375	-1.478211	0.1393678
niveduc5	675.6177473	40.6988346	16.600420	0.0000000
niveduc6	138.0827057	51.3160708	2.690828	0.0071336
niveduc7	3665.9555956	58.0839809	63.114744	0.0000000
ingreso:niveduc2	0.0590252	0.0203642	2.898483	0.0037540
ingreso:niveduc3	-0.6861225	0.0095227	-72.050903	0.0000000
ingreso:niveduc4	0.0340919	0.0126756	2.689561	0.0071608
ingreso:niveduc5	-0.2007009	0.0093876	-21.379272	0.0000000
ingreso:niveduc6	0.0200449	0.0100394	1.996621	0.0458804
ingreso:niveduc7	-0.3720818	0.0082739	-44.970740	0.0000000

## Generación de predicciones

Para visualizar ambos modelos, necesitamos generar una cuadrícula con todas las combinaciones de x1 y x2:

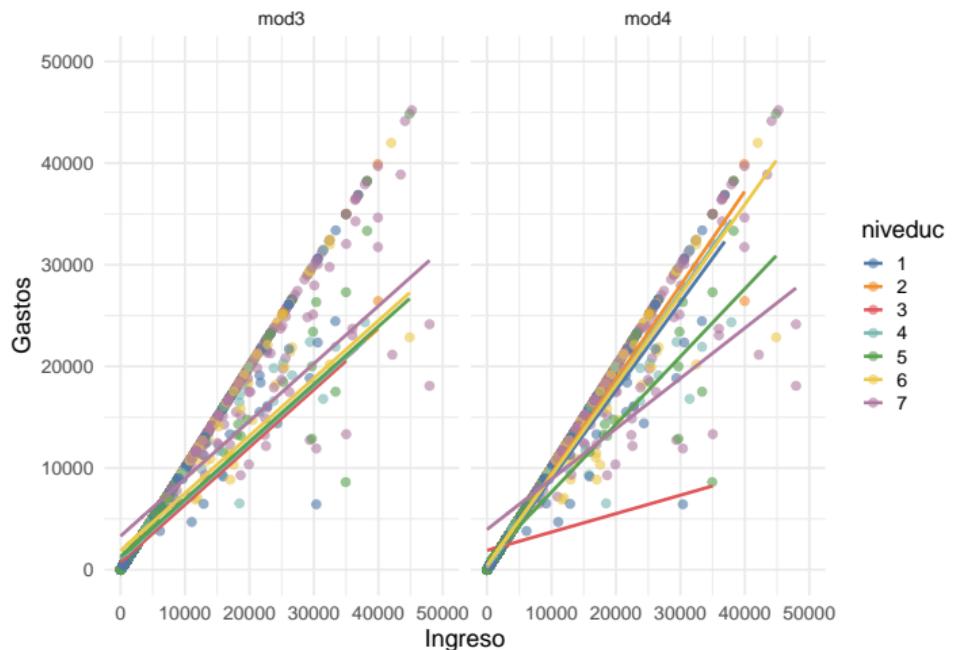
```
datos3 <- datos2 %>%
  select(gasto, ingreso, niveduc_labels , niveduc) %>%
  gather_predictions(mod3, mod4)

head(datos3, 5)
```

model	gasto	ingreso	niveduc_labels	niveduc	pred
mod3	5391.527	5391.527	Universitaria completa	7	6358.244
mod3	5391.527	5391.527	Secundaria completa	5	4344.080
mod3	7077.083	7077.083	Universitaria completa	7	7311.905
mod3	7105.557	7105.557	Terciaria incompleta	6	5886.859
mod3	7077.083	7077.083	Primaria incompleta	1	5056.665

# Visualización comparativa

Podemos representar los resultados de ambos modelos con facetas:

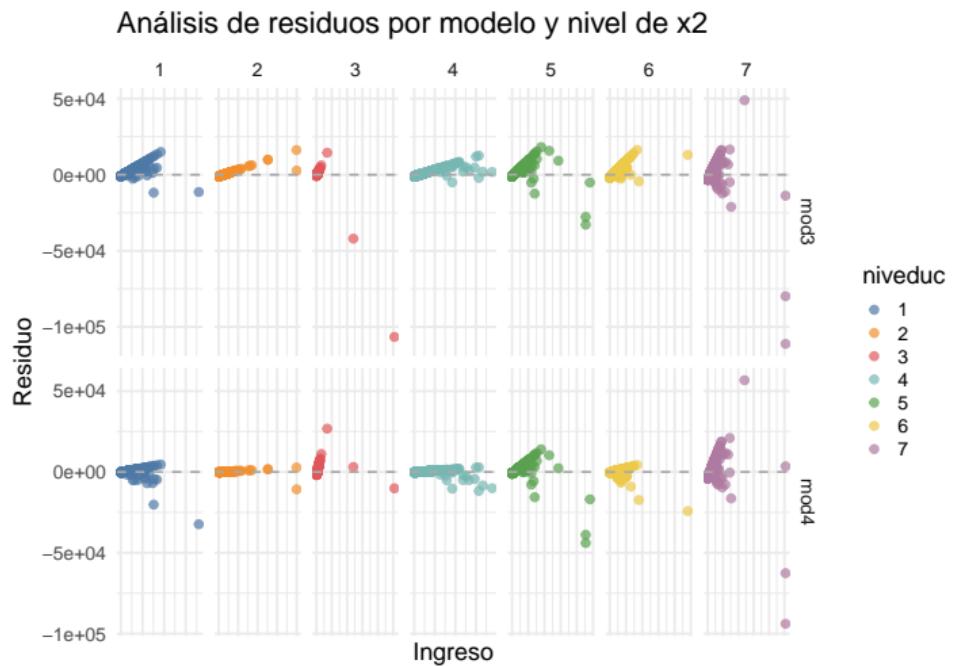


## Interpretación gráfica

- ▶ En el modelo **aditivo (+)**:
  - ▶ Misma pendiente para todas las categorías.
  - ▶ Diferentes interceptos.
- ▶ En el modelo **con interacción (\*)**:
  - ▶ Tanto la pendiente como el intercepto varían según  $x_2$ .

# Evaluación del ajuste

Para evaluar cuál modelo describe mejor los datos, analizamos los residuos:



## Observaciones

- ▶ El modelo con interacción permite capturar **cambios en la pendiente** según el nivel de la variable categórica.
- ▶ Aunque una comparación formal requiere herramientas estadísticas adicionales, el **análisis visual** de las pendientes y los residuos es suficiente para identificar mejoras sustanciales en el ajuste.
- ▶ **Las interacciones revelan cómo el efecto de una variable depende del valor de otra.**
- ▶ Ignorarlas puede llevar a interpretaciones engañosas, especialmente cuando las relaciones cambian entre grupos.

# Transformaciones dentro de la fórmula

Puedes hacer transformaciones directamente en `lm()`:

```
lm(log(y) ~ sqrt(x1) + x2)
```

Esto es equivalente a:

$$\log(y) = a_1 + a_2 \cdot \sqrt{x_1} + a_3 \cdot x_2$$

Si la transformación involucra `+`, `*`, `^` o `-`, usa `I()` para que R la interprete correctamente:

```
lm(y ~ x + I(x^2))
```

## ! Importante

Evita errores: `y ~ x ^ 2 + x` se traduce en interacción de `x` consigo mismo, lo que R simplifica automáticamente a `y ~ x`.

## Verificación con `model_matrix()`

```
df <- tribble(~y, ~x, 1, 1, 2, 2, 3, 3)
model_matrix(df, y ~ x^2 + x)
```

		(Intercept)	x
		1	1
		1	2
		1	3

```
model_matrix(df, y ~ I(x^2) + x)
```

		(Intercept)	I(x^2)	x
		1	1	1
		1	4	2
		1	9	3

`model_matrix()` permite ver **exactamente qué está ajustando `lm()`.**

## Aproximación polinomial

El teorema de Taylor permite aproximar funciones suaves con polinomios:

```
model_matrix(df, y ~ poly(x, 2))
```

	(Intercept)	poly(x, 2)1	poly(x, 2)2
1	-0.7071068	0.4082483	
1	0.0000000	-0.8164966	
1	0.7071068	0.4082483	

Fuera del rango de los datos, los polinomios pueden **explotar rápidamente**.

## Alternativa: splines naturales

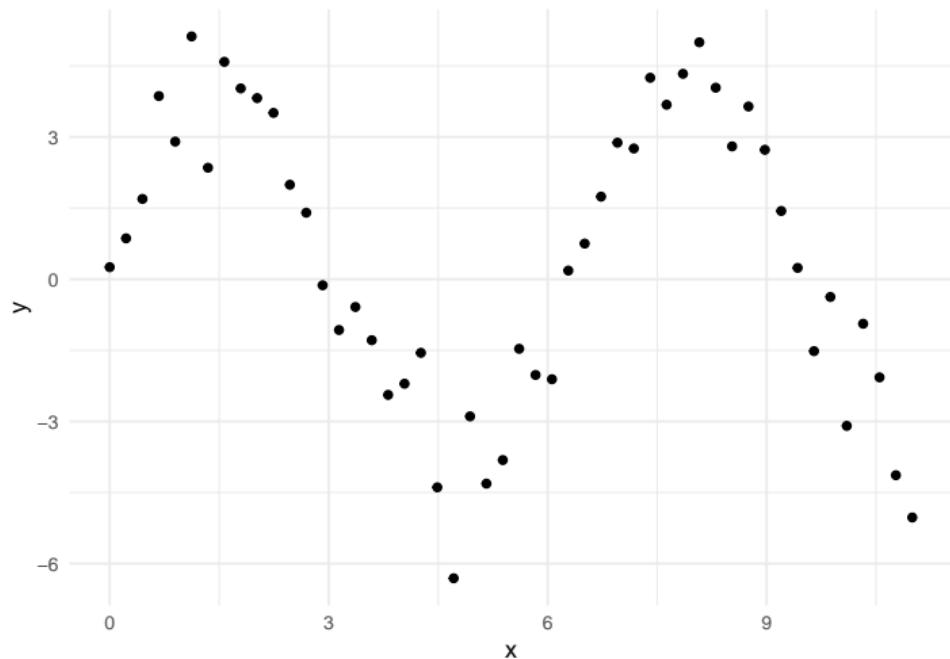
```
library(splines)
model_matrix(df, y ~ ns(x, 2))
```

	(Intercept)	ns(x, 2)1	ns(x, 2)2
1	0.0000000	0.0000000	
1	0.5662628	-0.2108419	
1	0.3440969	0.7706021	

- ▶ `ns(x, df)` genera splines naturales con `df` grados de libertad
- ▶ Mantiene estabilidad fuera del rango de los datos

## Ejemplo práctico

```
sim5 <- tibble(  
  x = seq(0, 3.5 * pi, length = 50),  
  y = 4 * sin(x) + rnorm(length(x))  
)
```



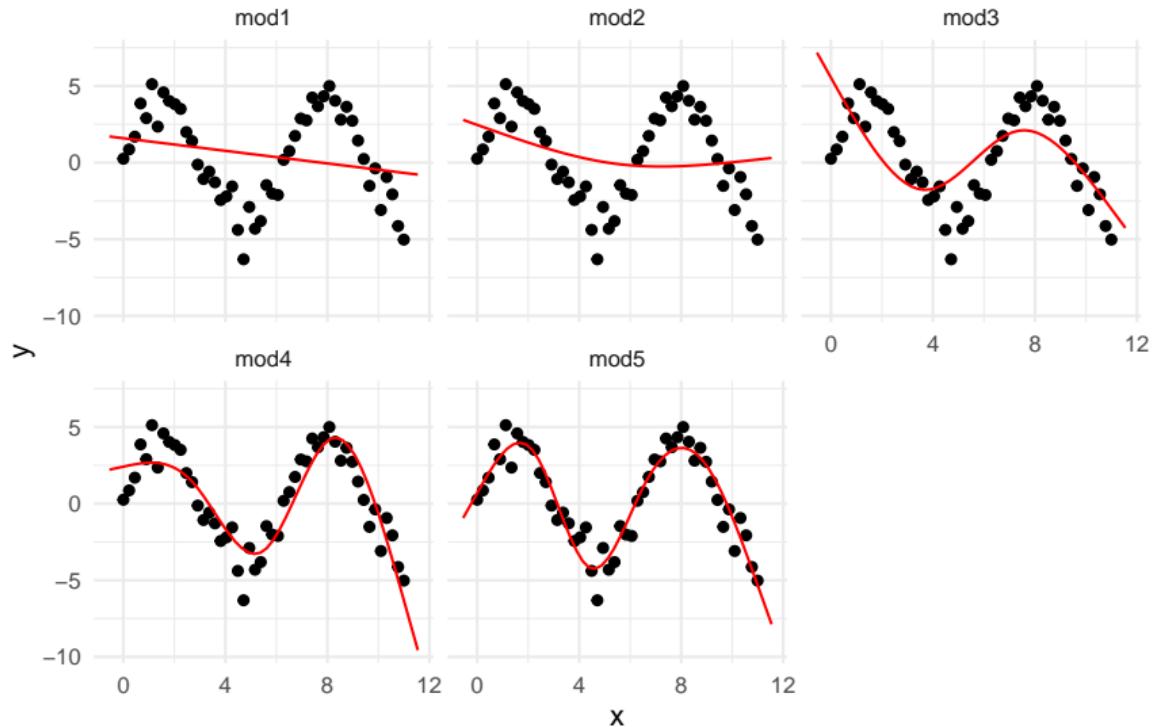
## Ajuste de múltiples modelos con splines

```
mod1 <- lm(y ~ splines::ns(x, 1), data = sim5)
mod2 <- lm(y ~ splines::ns(x, 2), data = sim5)
mod3 <- lm(y ~ splines::ns(x, 3), data = sim5)
mod4 <- lm(y ~ splines::ns(x, 4), data = sim5)
mod5 <- lm(y ~ splines::ns(x, 5), data = sim5)

grid <- sim5 %>%
  data_grid(x = seq_range(x, n = 50, expand = 0.1)) %>%
  gather_predictions(mod1, mod2, mod3, mod4, mod5, .pred = "y")
```

# Visualización comparativa

Ajuste de modelos spline con distintos grados de libertad



## Observaciones

- ▶ La extrapolación fuera del rango de los datos es peligrosa
- ▶ Los modelos nunca dicen si el comportamiento es verdadero fuera del rango observado

## Valores faltantes

Los valores faltantes no aportan información sobre la relación entre variables. Por defecto, `lm()` **elimina filas con NA** de manera silenciosa.

```
library(tidyverse)
df <- tribble(~x, ~y, 1, 2.2,
              2, NA, 3, 3.5, 4, 8.3, NA, 10
)
df
```

	x	y
1	2.2	
2	NA	
3	3.5	
4	8.3	
NA	10.0	

## Valores faltantes

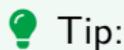
Para suprimir los mensajes de advertencia:

```
mod <- lm(y ~ x, data = df, na.action = na.exclude)
```

Podemos consultar cuántas observaciones se usaron:

```
nobs(mod)
```

```
[1] 3
```



Tip:

Ajustar `options(na.action = na.warn)` permite que R nos notifique automáticamente sobre la eliminación de filas con valores faltantes.

## Otros modelos

Hasta ahora nos centramos en **modelos lineales**, que suponen:

- ▶ Relación lineal:  $y = a_1x_1 + a_2x_2 + \cdots + a_nx_n$
- ▶ Residuos normalmente distribuidos

# Extensiones de modelos

- ▶ Modelos lineales generalizados
  - ▶ Permiten *respuestas no continuas* (binarias, conteos)
  - ▶ Basados en verosimilitud en lugar de solo mínimos cuadrados

```
glm(y ~ x, family = binomial, data = df)
```

- ▶ Modelos aditivos generalizados
  - ▶ Incorporan **funciones suaves arbitrarias**
  - ▶ Ajustan ( $y = f(x)$ ) con restricciones de suavidad

```
library(mgcv)  
gam(y ~ s(x), data = df)
```

# Extensiones de modelos

- ▶ Modelos lineales penalizados
  - ▶ Penalizan coeficientes grandes para **evitar sobreajuste**
  - ▶ Mejoran la **generalización** a nuevos datos

```
library(glmnet)
glmnet(x = as.matrix(df$x), y = df$y)
```

- ▶ Modelos robustos
  - ▶ Menos sensibles a **valores extremos**
  - ▶ Útiles cuando hay outliers, pero no siempre necesarios

```
library(MASS)
rlm(y ~ x, data = df)
```