Análisis Exploratorio de Datos: Práctica 3

Gonzalo Barrera Borla

15-17 Abril, 2018

Acondicionamiento Inicial

Carga de librerías

```
library(tidyverse)
library(scales)
set.seed(1337)
```

Ejercicio 1: `autos.txt`

Se pretende analizar la relación entre el precio y la calidad de una muestra con $n = 100$ autos. EL siguiente código destaca cómo aplicar varios modelos a los mismos datos:

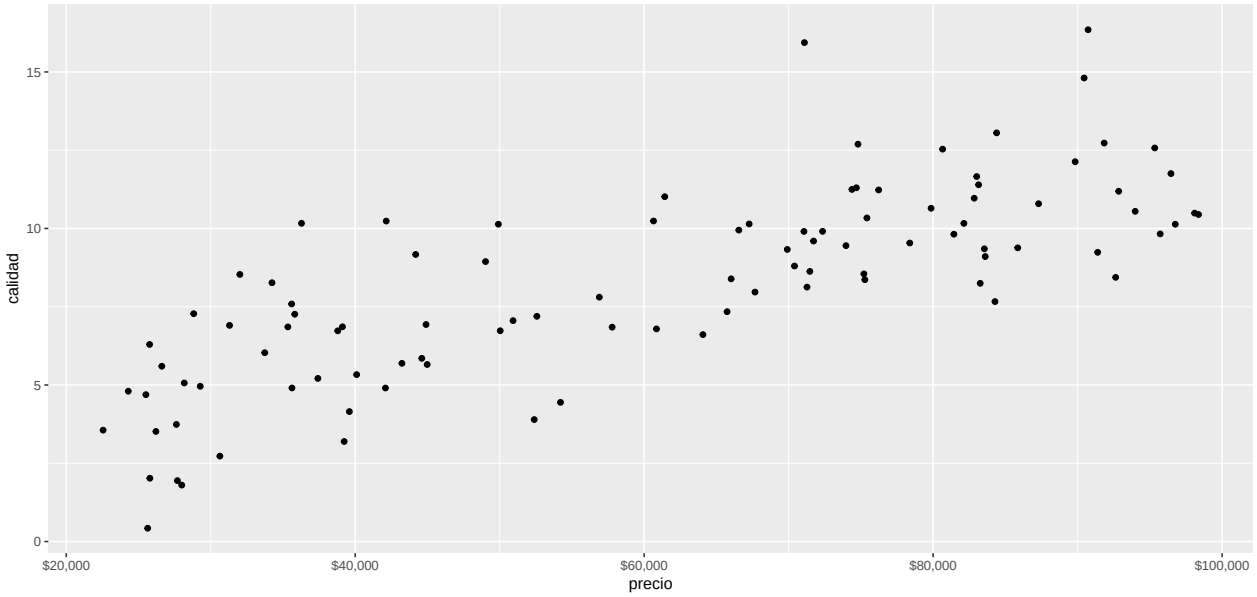
Carga de datos

```
autos <- read_delim(file = "../Datos/autos.txt", delim = " ",
                    col_types =  cols(precio = col_double(), calidad = col_double()))
```

Diagrama de dispersión precio-calidad

```
autos %>%
  ggplot(mapping = aes(y = calidad, x = precio)) +
  geom_point() +
  scale_x_continuous(labels = dollar) +
  labs(title = "Fig. 1: Dispersión precio-calidad para n = 100 autos.") -> fig1

ggsave("fig1_dispersion_precio_calidad.png", plot = fig1, width = 12, height = 8)
fig1
```

Fig. 1: Dispersión precio−calidad para n = 100 autos.



La lista `formulas_modelos` contendrá las "fórmulas" de R que intentaremos ajustar a nuestros datos.

```
# Formulas por modelo
formulas_modelos <- list(
  # Solo ordenada
  formula(calidad ~ 1),
  # Lineal en precio sin ordenada
  formula(calidad ~ 0 + precio),
  # Lineal en precio con ordenada
  # equivalente a "calidad ~ poly(precio, 1)"
  formula(calidad ~ precio),
  # Varios polinomios de distinto grado
  calidad ~ poly(precio, 2),
  calidad ~ poly(precio, 3),
  calidad ~ poly(precio, 4)
)
# es equivalente a formula(calidad ~ 1 + precio)
ajustar_reglin <- function(datos, formula) { lm(formula, datos) }

predecir_n_miles <- function(n) {
  function(modelo) {
    precios <- tibble(precio = seq(n)*1000)
    precios %>%
      mutate(calidad = predict(modelo, precios))
    #tibble(precio = seq(50)*1000,
    #       calidad = predict(modelo, newdata = seq(50)*1000))
  }
}

# No sé como llamar a esta variable expresivamente
tibble(id = 1:length(formulas_modelos),
       formula = formulas_modelos,
       nombre = map_chr(formulas_modelos, deparse),
  # Ajusto una regresion lineal sobre `datos` segun `formula` y guardo el objeto resultado en `resultad
```

```
      resultados = map(formula, ~lm(., autos)),
  # Utilizo `predict` para predecir sobre un dataset con una sola fila de precio = 50000
  predicciones = map(resultados, predecir_n_miles(100))) -> df
```

Divido el conjunto de datos en train y test y ajusto el modelo nuevamente

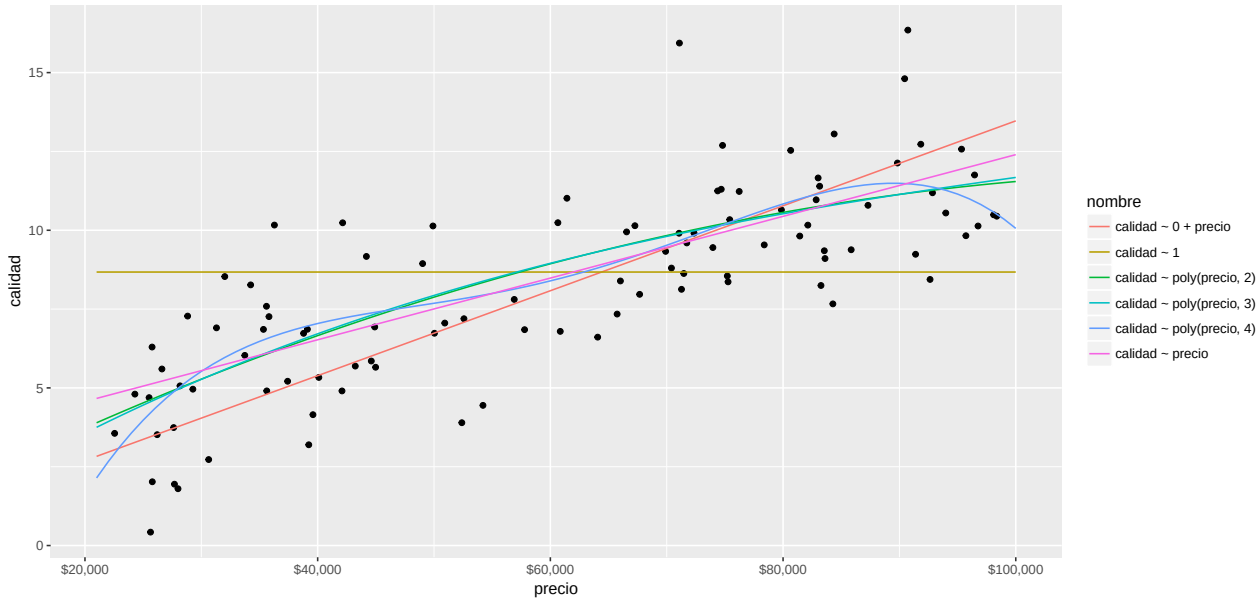
```
test_frac <- 0.3
autos <- autos %>%
  mutate(is_test = sampling::srswor(test_frac*n(), n()) == 1)
autos_test <- autos %>% filter(is_test)
autos_train <- autos %>% filter(!is_test)

df %>%
  # Entreno el modelo sobre los datos de entrenamiento
  mutate(resultados_train = map(formula, ~lm(., autos_train)),
  # Genero predicciones para los datos de prueba
  predicciones_test = map(resultados_train, ~broom::augment(., newdata = autos_test))) -> df

# Calculo RSE por modelo
calcular_rse <- function(df) { mean((df$calidad-df$.fitted)^2) }
df %>% mutate(rse_test = map_dbl(predicciones_test, calcular_rse)) -> df

df[c("rse_test", "nombre")] %>%
  arrange(rse_test) %>% mutate(ranking = 1:n()) -> resumen_modelos

kable(resumen_modelos)
```

rse_test	nombre	ranking
3.245565	calidad ~ 0 + precio	1
3.447511	calidad ~ poly(precio, 4)	2
3.535645	calidad ~ poly(precio, 2)	3
3.543309	calidad ~ precio	4
3.554307	calidad ~ poly(precio, 3)	5
10.485602	calidad ~ 1	6

El resumen de los modelos no nos dice demasiado: veámoslos graficados sobre el scatterplot.

```
df %>%
  select(nombre, resultados_train) %>%
  mutate(predicciones_test_full = map(resultados_train, predecir_n_miles(100))) %>%
  select(-resultados_train) %>%
  unnest() %>% filter(precio > 20000) -> predicciones_por_modelo

fig2 <- fig1 +
  geom_line(predicciones_por_modelo, mapping = aes(precio, calidad, colour=nombre)) +
  labs(title ="Fig. 2: Dispersión precio-calidad y curvas de ajuste para distintos modelos")

ggsave("fig2_dispersion_modelos.png", plot = fig2, width = 12, height = 8)
fig2
```

Fig. 2: Dispersión precio−calidad y curvas de ajuste para distintos modelos



Ejercicio 2: simulación y regresión

Generación de datos

El enunciado especifica que:

$$X \sim N(\mu = 0, \sigma^2 = 1)$$
$$\epsilon \sim N(\mu = 0, \sigma^2 = 0.25)$$
$$Y = -1 + 0.5X + \epsilon$$

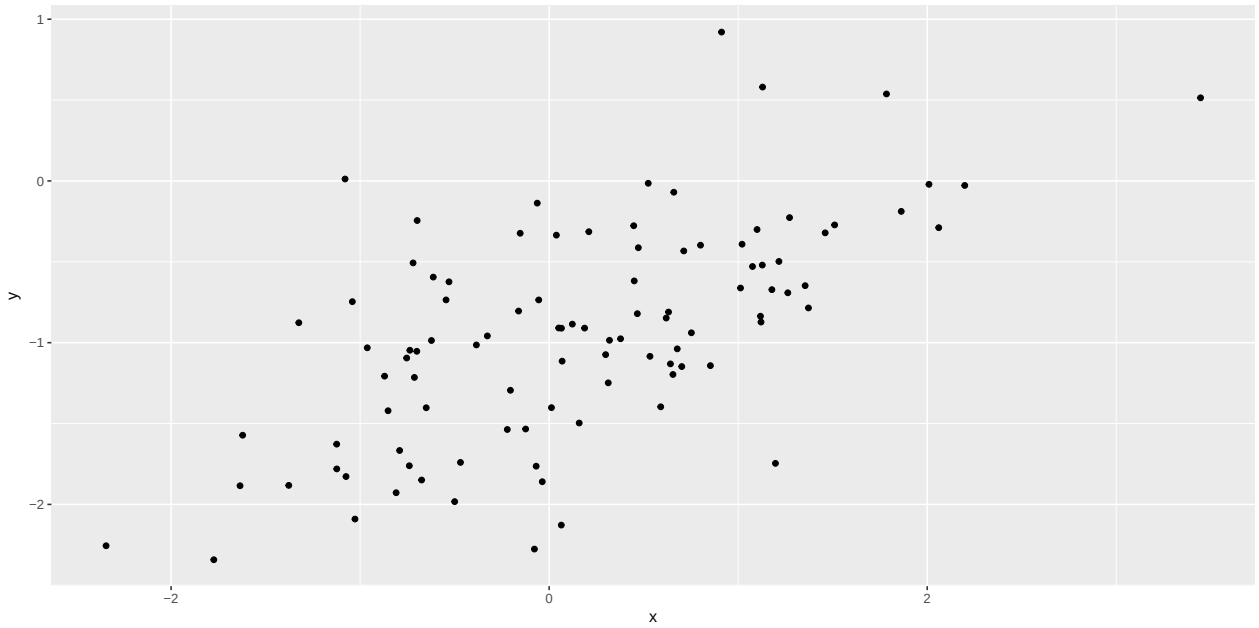
de forma que $\beta_0 = -1, \beta_1 = 0.5$

```
b0 <- -1
b1 <- 0.5
```

```
n_sims <- 100

datos_ej3 <- tibble(
  x = rnorm(n = n_sims),
  # entiendo que sigma^2 es la _varianza_, no el desvío estándar del error epd
  eps = rnorm(n = n_sims, sd = sqrt(0.25)),
  y = - 1 + 0.5*x + eps)
```

Scatterplot y-x

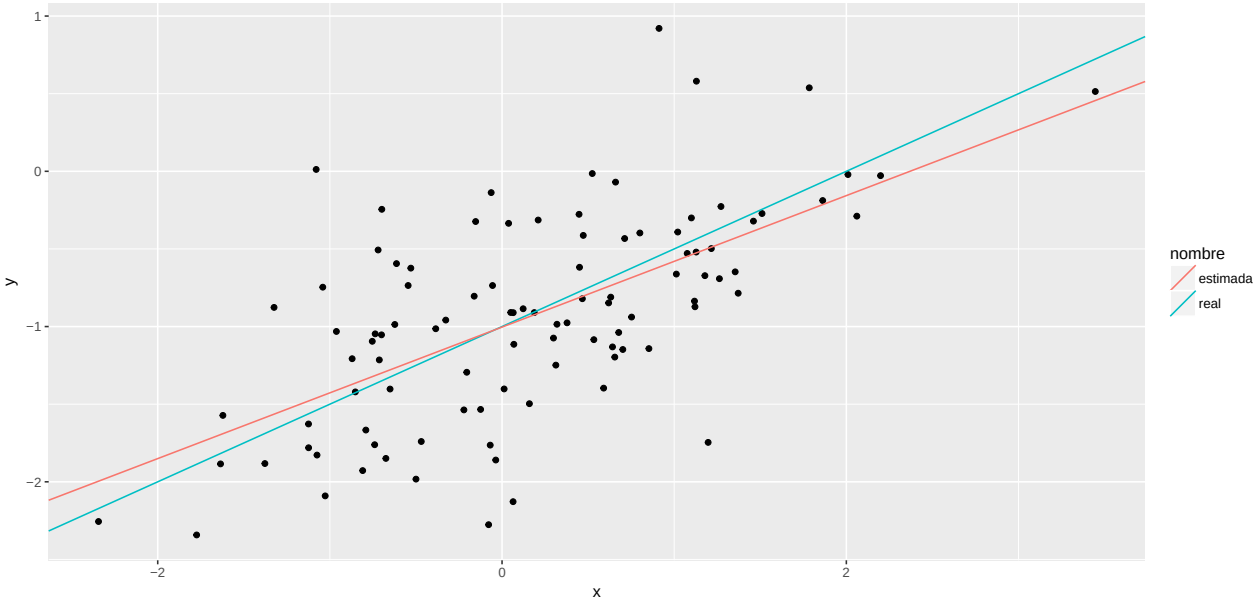
```
datos_ej3 %>%
  ggplot(aes(x, y)) + geom_point() -> fig3

ggsave("fig3_dispersion_simulaciones.png", plot = fig3, width = 12, height = 8)
fig3
```



Ajuste cuadrados mínimos

```
ajuste_lm <- lm(y ~ x, datos_ej3)
b0_hat <- ajuste_lm$coefficients["(Intercept)"]
b1_hat <- ajuste_lm$coefficients["x"]

lineas <- tribble(
  ~nombre, ~b0, ~b1,
  "real", b0, b1,
  "estimada", b0_hat, b1_hat
)

fig3 +
      geom_abline(data = lineas, mapping = aes(slope = b1, intercept = b0, color = nombre)) +
        labs(title = "Fig. 4: Línea de ajuste estimada contra real") -> fig4

ggsave("fig4_dispersion_simulaciones_ajuste.png", plot = fig4, width = 12, height = 8)
fig4
```

Fig. 4: Línea de ajuste estimada contra real



Probando niveles de contaminación

¡Nuevo modelo! Sea $\tilde{Y_\delta}$ la nueva variable con grado de contaminacion δ. Recordamos la definición anterior:

$$
\begin{aligned}
X &\sim N(\mu = 0, \sigma^2 = 1) \\
\epsilon &\sim N(\mu = 0, \sigma^2 = 0.25) \\
Y &= -1 + 0.5X + \epsilon
\end{aligned}
$$

Ahora,

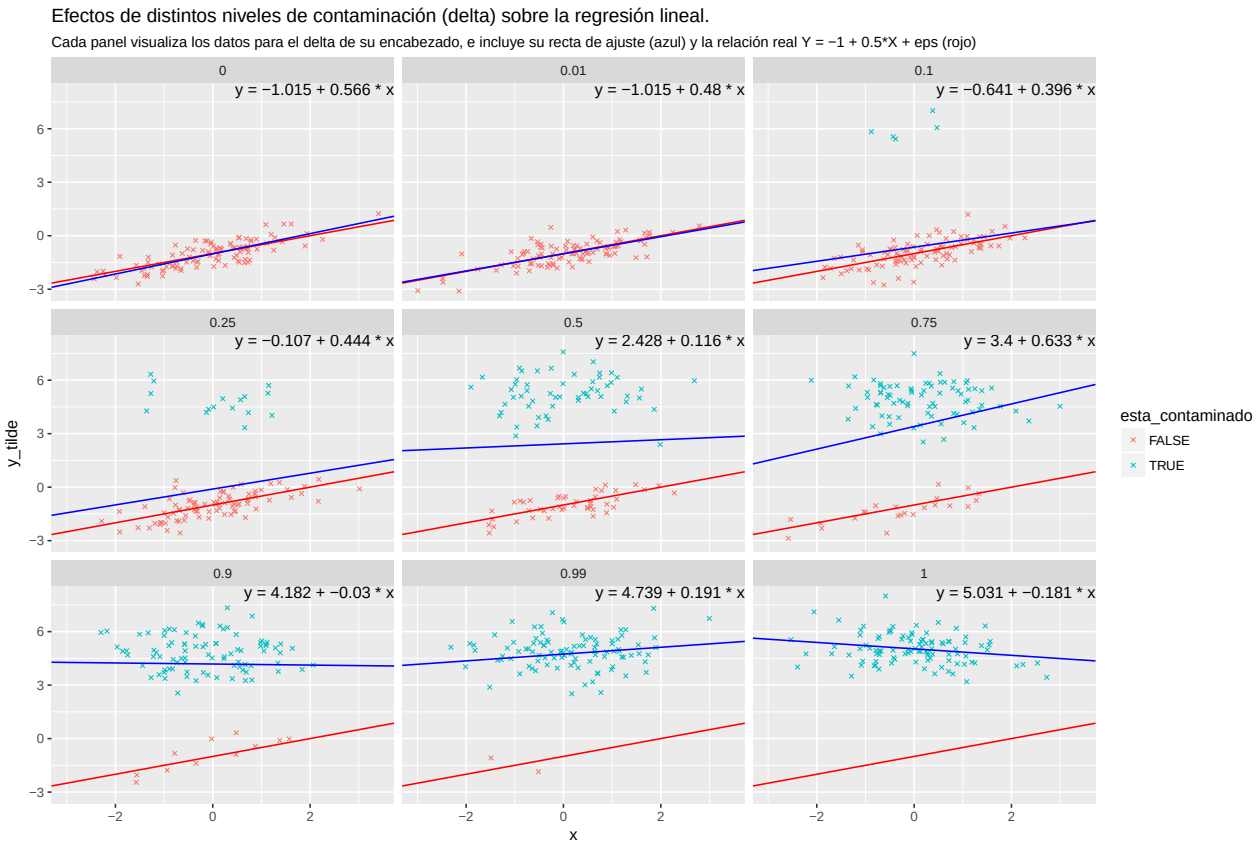
$$
\begin{aligned}
V_\delta &\sim Bernoulli(p = 1 - \delta) \\
W &\sim N(\mu = 5, \sigma^2 = 1) \\
\tilde{Y_\delta} &= V_\delta \times Y + (1 - V_\delta) \times W
\end{aligned}
$$

En el enunciado original la formulación es un poco distinta. Dice que $V_\delta \sim Binom(1, p = \delta)$, lo cual es equivalente a una Bernoulli, entiendo. A la vez, toma $V = 1$ como "contaminado", salvo que la ecuación de $\tilde{Y_\delta}$ lo plantea al revés: V=1 es no contaminado, y V=0 es contaminado. Por último, $W \sim N(50, 1)$, pero una contaminación tan brutal de los datos hace poco claros los gráficos siguientes.

Además, agregamos el parámetro $0 \leq \delta \leq 1$ para controlar el grado de contaminación.

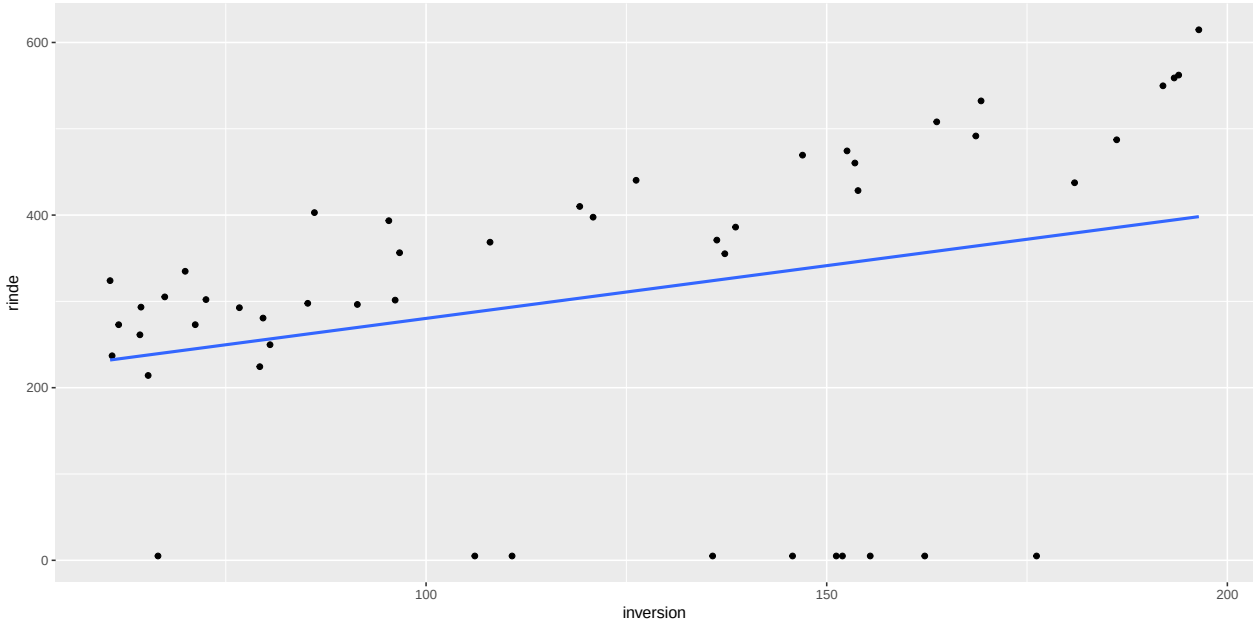
```
generar_datos_contaminados <- function(n, delta) {
  tibble(
    x = rnorm(n = n, mean = 0, sd = 1),
    eps = rnorm(n, 0, sqrt(0.25)),
    y = - 1 + 0.5*x + eps,
    v = rbinom(n, 1, 1-delta),
    esta_contaminado = v == 0,
    w = rnorm(n, 5, 1),
    y_tilde = v * y + (1-v) * w)
}

grados_cont <- c(0, 0.01, 0.1, 0.25, 0.5, 0.75, 0.9, 0.99, 1)
```

```
efectos_cont <- tibble(
  delta = grados_cont,
  datos = map(delta, ~generar_datos_contaminados(100, .)),
  modelo = map(datos, ~(lm(y_tilde ~ x, .))),
  ordenada = map_dbl(modelo, ~coef(.)[1]),
  pendiente = map_dbl(modelo, ~coef(.)[2]))

efectos_cont %>% select(delta, datos) %>% unnest() -> observaciones
efectos_cont %>% select(delta, ordenada, pendiente) -> regresiones

observaciones %>%
  ggplot(aes(x, y_tilde, group=delta)) +
  geom_point(size = 1, shape=4, mapping = aes(color = esta_contaminado)) +
  geom_abline(slope = b1, intercept = b0, color = "red") +
  geom_abline(regresiones, mapping = aes(slope = pendiente, intercept = ordenada), color = "blue") +
  geom_text(regresiones, vjust = 1, hjust = 1, mapping = aes(
    Inf, Inf, label=paste("y =", round(ordenada,3), "+", round(pendiente,3), "* x"))) +
  facet_wrap(~delta, nrow=3, ncol=3) +
  labs(
    title = "Efectos de distintos niveles de contaminación (delta) sobre la regresión lineal.",
    subtitle = paste(
      "Cada panel visualiza los datos para el delta de su encabezado, e incluye",
      "su recta de ajuste (azul) y la relación real Y = -1 + 0.5*X + eps (rojo)")) -> fig5

ggsave("fig5_efectos_contaminacion.png", plot = fig5, width = 18, height = 12)
fig5
```



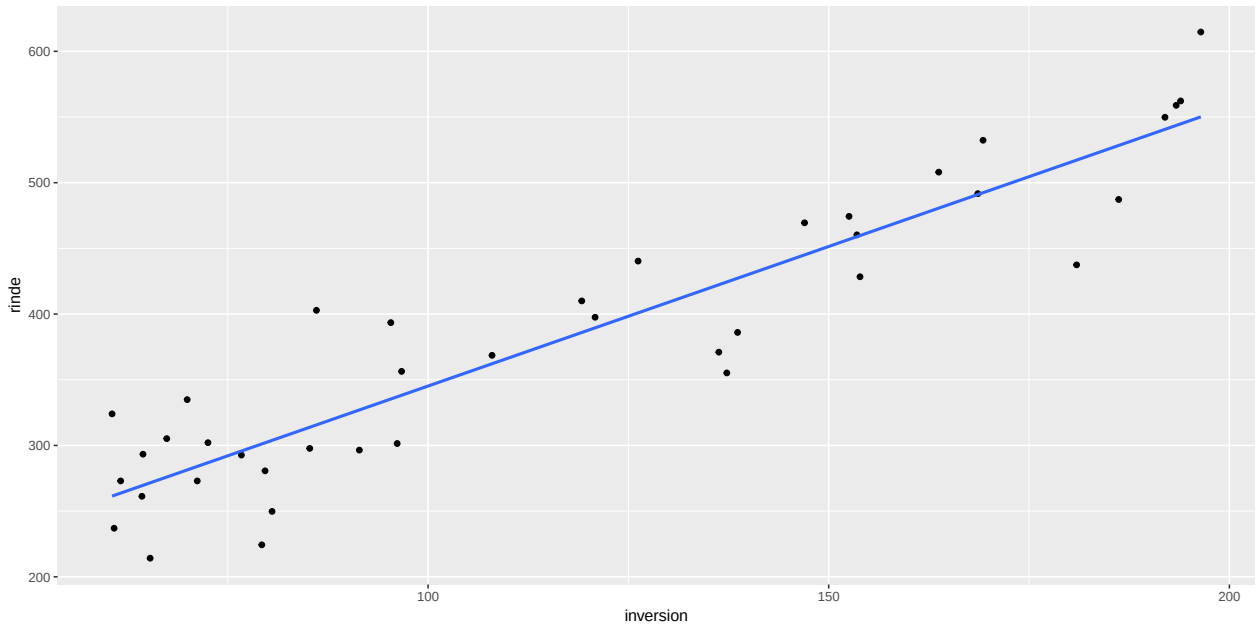
Ejercicio 3: `girasol.txt`

```
girasol <- read_delim("../Datos/girasol.txt", delim = " ",
                      col_types = cols(inversion = col_double(), rinde = col_double()))

girasol %>%
  ggplot(aes(inversion, rinde)) +
  geom_point() +
  geom_smooth(method = 'lm', formula = y ~ x, se = F)
```



Hmmm, para varios puntos el rinde es == 5 independientemente de la inversión. Limpiamos y reajustamos:

```
girasol %>%
  filter(rinde != 5) %>%
  ggplot(aes(inversion, rinde)) +
  geom_point() +
  geom_smooth(method = 'lm', formula = y ~ x, se = F)
```



¡Mucho mejor!

Ejercicio 4: `abalone.txt`

```
nombres_columnas <- c(
  "sexo", "longitud", "diametro", "altura", "p_completo",
  "p_carne", "p_visceras", "p_caparazon", "anillos")
tipos_columnas <-  cols(
  .default = col_double(),
  sexo = readr::col_factor(levels = c("M", "F", "I")),
  anillos = col_integer()
)
abalone <- read_csv("../Datos/abalone.txt", col_names = nombres_columnas, col_types = tipos_columnas)


# Repetimos la técnica usada en `autos`
test_frac <- 0.3
abalone <- abalone %>%
  mutate(is_test = sampling::srswor(test_frac*n(), n()) == 1)
abalone_test <- abalone %>% filter(is_test)
abalone_train <- abalone %>% filter(!is_test)

sin_test <- lm(diametro ~ longitud, abalone)
con_test <- lm(diametro ~ longitud, abalone_train)
broom::augment(con_test, newdata = abalone_test)%>%
  summarise(rse = mean((longitud - .fitted)^2)) -> rse_test
```

El RSE de prueba es 0.0138713.