TP2 Análisis Exploratorio de Datos

Gonzalo Barrera Borla, Octavio M. Duarte y Juan Manuel Berros

28 de Abril de 2018

Carga de librerías y datos

El siguiente análisis utiliza librerías del "pulcriverso", o *tidyverse*, como *dplyr* (para manipular *tibbles*, o *data frames*), *purrr* (para vectorizar el trabajo sobre data frames) y *magrittr* (para encadenar operaciones), junto con *knitr* para generar este informe.

Revisar el archivo `galtonMod.csv` en cualquier editor de texto nos revela que las dos primeras columnas contienen el número de observación de forma redundante, con lo cual podemos eliminar una. Adicionalmente, renombraremos las columnas con identificadores más descriptivos.

```
galton <- read_csv(
  "GaltonMod.csv", skip = 1,
  col_names = c("indice", "obs", "padre", "hijo"),
  col_types = cols(
    indice = col_integer(),
    obs = col_integer(),
    padre = col_double(),
    hijo = col_double()
  )) %>%
  select(-indice)
```

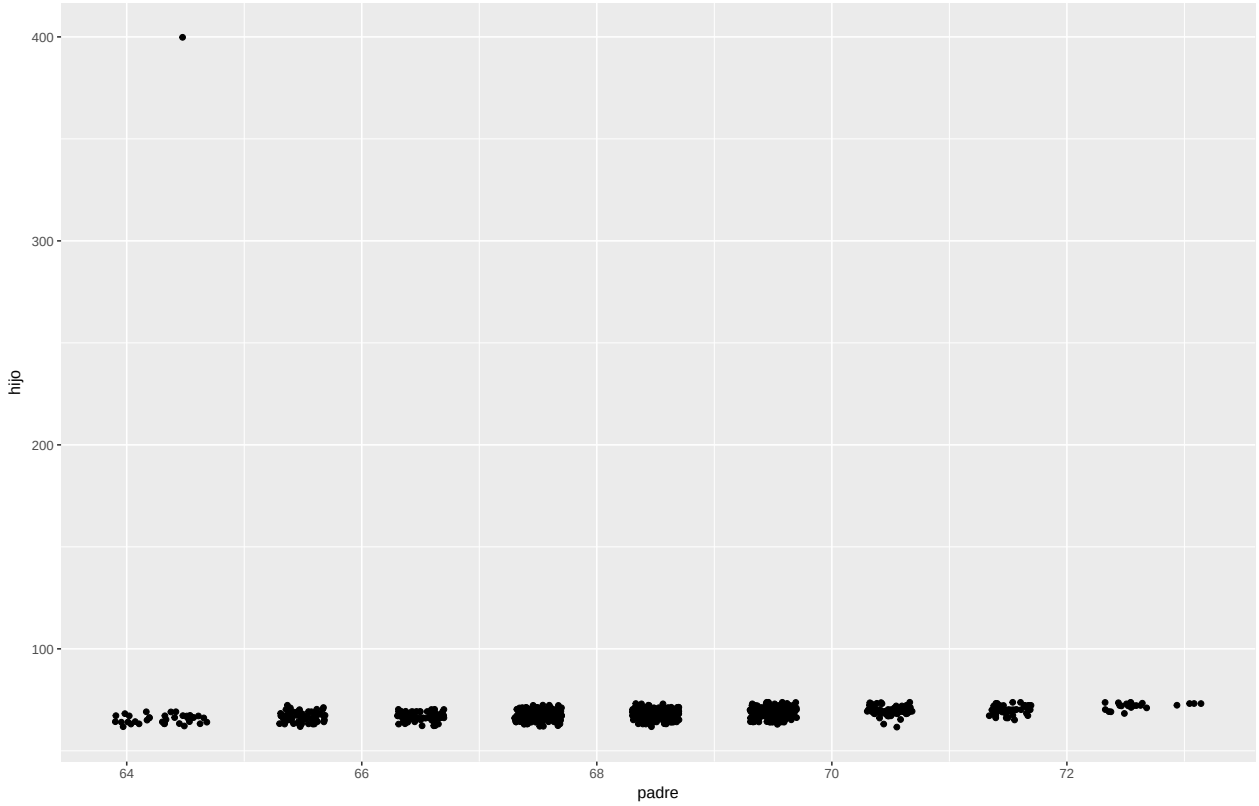
Limpieza de datos

Comenzamos graficando los datos crudos:

```
fig1 <- ggplot(data = galton, mapping = aes(y = hijo, x = padre)) +
  # position="jitter" les agrega un ligero "ruido" a los puntos para visualizarlos mejor
  geom_point(position="jitter") +
  labs(title = "Fig. 1: Alturas de padres e hijos, sin limpiar")

ggsave("fig1_dispersion_sin_limpiar.png", fig1, width = 12, height = 8)
fig1
```

Fig. 1: Alturas de padres e hijos, sin limpiar



Resultan evidente dos cosas: 1. Las alturas de padres e hijos están en una unidad extraña de medida, aparentemente pulgadas. Multiplicamos por 2,54 para transformarlas a centímetros. 2. Hay al menos 1 registro con valores imposiblemente altos para la altura de `hijo`. Como límite razonable, utilizaremos 272 cm, la máxima altura registrada para un ser humano según Wikipedia.

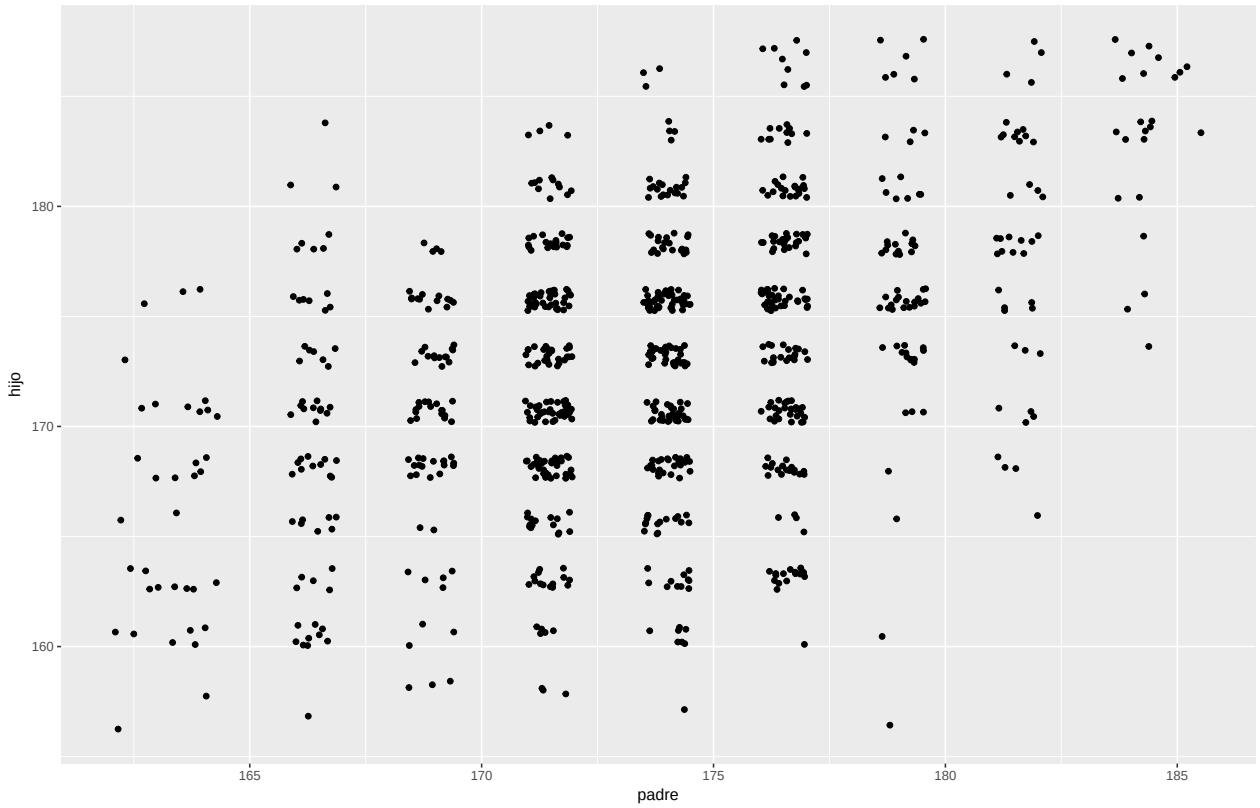
Hechas las correcciones, volvemos a graficar la dispersión de `padre` e `hijo`, para comprobar que los datos se vean coherentes:

```
galton <- galton %>%
  mutate(padre = padre * 2.54, hijo = hijo * 2.54) %>%
  filter(padre < 272, hijo < 272)

fig2 <- ggplot(data = galton, mapping = aes(y = hijo, x = padre)) +
  # position="jitter" les agrega un ligero "ruido" a los puntos para visualizarlos mejor
  geom_point(position="jitter") +
  labs(title = "Fig. 2: Alturas de padres e hijos, en centímetros, limpias")

ggsave("fig2_dispersion_limpia.png", fig2, width = 12, height = 8)
fig2
```

Fig. 2: Alturas de padres e hijos, en centímetros, limpias



¡Satisfactorio!

Ajuste y selección de modelos lineales

A continuación intentaremos ajustar distintas familias de modelos a los datos. Ya que el único predictor disponible para `hijo` es `padre`, intentaremos ajustar $m = 4$ polinomios de grados 0, 1, 2 y 3 inclusive sobre `padre` y ver cómo se comparan. Comenzamos declarando una lista de fórmulas para cada modelo:

Por otra parte, necesitamos un criterio de selección de modelos. para comparar los resultados. Para ello, utilizaremos un esquema de *k-pliegues*, asignando a cada observación un valor de k al azar:

```
formulas_modelos <- list(
  "constante" = hijo ~ 1,
  "lineal" = hijo ~ padre,
  "cuadratico" = hijo ~ poly(padre, 2),
  "cubico" = hijo ~ poly(padre, 3)
)

K <- 20

galton <- galton %>%
  mutate(k = sample(K, n(), replace = T))
```

El esquema de *k-pliegues* requiere entrenar a cada modelo k veces, usando en la i-ésima iteración los datos con $k \neq i$ para el entrenamiento, y los datos con $k = i$ para la evaluación. Es decir que si comparamos m modelos, tendremos que entrenar en total $m \times k$ regresiones lineales distintas.

Las siguientes funciones auxiliares nos permitirán recuperar los datos necesarios con facilidad, así como entrenar al modelo m-ésimo sobre la k-ésima partición, y calcular su error cuadrático medio (ECM/MSE)

```
galton_train <- function(k_test) { galton %>% filter(k != k_test) }
galton_test <-  function(k_test) { galton %>% filter(k == k_test) }

entrenar_m_en_k <- function(m, k) { lm(formulas_modelos[[m]], galton_train(k))}
predecir_m_en_k <- function(mod, k) { predict(mod, newdata = galton_test(k))}

calcular_ecm <- function(y, y_hat) { mean((y - y_hat)^2) }
```

Prepararemos ahora un *tibble* (una clase que hereda del *data frame* y se lleva mejor con las librerías de *tidyverse*) con los elementos relevantes de cada uno de los $m \times k$ ajustes lineales:

- `id_modelo`, el nombre del modelo a utilizar,
- k_{test}, el número de pliego sobre el cual calcular la pérdida del modelo ajustado,
- `modelo`, el ajuste del m-ésimo modelo a los k-ésimos datos de entrenamiento,
- `n_k_test` ($N(k_{test})$), la cantidad de elementos a predecir,
- `y_test` (Y_{test}), los valores a predecir,
- `predicciones_test` ($\hat{Y}_{test}$), las predicciones para Y_{test}, y
- `ecm` el error cuadrático medio

```
n_modelos <- length(formulas_modelos)

kfold <- tibble(
  id_modelo = rep(names(formulas_modelos), times = K),
  k_test = rep(1:K, each = n_modelos),
  modelo = map2(id_modelo, k_test, entrenar_m_en_k),
  y_test = map(k_test, galton_test) %>% map("hijo"),
  predicciones_test = map2(modelo, k_test, predecir_m_en_k),
  n_k_test = map_int(y_test, length),
  ecm = map2_dbl(y_test, predicciones_test, calcular_ecm))
```

Ahora necesitamos computar el ECM promedio para cada modelo, para lo cual habrá que ponderar los ECM por cada modelo en cada pliego:

```
resultados <- kfold %>%
  select(id_modelo, n_k_test, ecm) %>%
  group_by(id_modelo) %>%
  summarise(ecm_total = weighted.mean(ecm, n_k_test))

kable(resultados)
```

id_modelo	ecm_total
constante	40.71691
cuadratico	32.25040
cubico	32.14843
lineal	32.35751

Dado que el ECM para los modelos cuadrático y cúbico son prácticamente idénticos al del modelo lineal, podemos asumir con tranquilidad que un modelo lineal es el más parsimomioso para ajustar los datos. Veamos, sobre el gráfico de dispersión anterior, las 20 rectas que generamos con cada conjunto de entrenamiento:

```
rectas <- kfold %>%
  filter(id_modelo == "lineal") %>%
```

```
  mutate(coeficientes = map(modelo, coef),
         ordenada = map_dbl(coeficientes, 1),
         pendiente = map_dbl(coeficientes, 2)) %>%
  select(k_test, ordenada, pendiente)

kable(head(rectas))
```

k_test	ordenada	pendiente
1	61.90343	0.6405637
2	62.85621	0.6350028
3	62.12626	0.6388451
4	62.82226	0.6344065
5	60.00090	0.6508925
6	62.73654	0.6353168

Se puede observar de la tabla anterior que el rango de las ordenadas como de las pendientes es bastante acotado, con lo cual las 20 regresiones van a ser bastante similares:

```
fig3 <- fig2 +
  geom_abline(data = rectas, mapping = aes(intercept = ordenada, slope = pendiente),
              alpha = 0.2, color ="steelblue3") +
  labs(title = "Fig. 3: Alturas de padres e hijos, junto con curvas de ajuste lineal") +
  scale_y_continuous(breaks = seq(0, 300, 5), minor_breaks = seq(0, 300, 1)) +
  scale_x_continuous(breaks = seq(0, 300, 5), minor_breaks = seq(0, 300, 1))

ggsave("fig3_dispersion_curvas.png", fig3, width = 12, height = 8)
fig3
```

Fig. 3: Alturas de padres e hijos, junto con curvas de ajuste lineal