Tarea 1

Mateo Perez

Octubre 2024

Ejercicio 1

Demostrar:

$$E\left[\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\beta}^T x_i)^2\right] \leq E\left[\frac{1}{m}\sum_{i=1}^{m}(y_i' - \hat{\beta}^T x_i')^2\right]$$

Primer paso:

$$E\left[\frac{1}{m}\sum_{i=1}^{m}(y_i' - \hat{\beta}^T x_i')^2\right] = \frac{1}{m}\sum_{i=1}^{m} E\left[(y_i' - \hat{\beta}^T x_i')^2\right]$$

Como:

$$E\left[(y_i' - \hat{\beta}^T x_i')^2\right] = E_t\left[E_v\left[(y_i' - \hat{\beta}^T x_i')^2|t\right]\right]$$

Entonces:

$$E\left[\frac{1}{m}\sum_{i=1}^{m}(y_i' - \hat{\beta}^T x_i')^2\right] = \frac{1}{m}\sum_{i=1}^{m} E_t\left[E_v\left[(y_i' - \hat{\beta}^T x_i')^2|t\right]\right]$$

Ahora tomo que:

$$E_v\left[(y_i' - \hat{\beta}^T x_i')^2|t\right] = E_v\left[(y_1' - \hat{\beta}^T x_1')^2|t\right]$$

Por lo tanto:

$$E\left[\frac{1}{m}\sum_{i=1}^{m}(y_i' - \hat{\beta}^T x_i')^2\right] = \frac{1}{m}\sum_{i=1}^{m} E_t\left[E_v\left[(y_1' - \hat{\beta}^T x_1')^2|t\right]\right]$$

Entonces:

$$E\left[\frac{1}{m}\sum_{i=1}^{m}(y_i' - \hat{\beta}^T x_i')^2\right] = E_t\left[E_v\left[(y_1' - \hat{\beta}^T x_1')^2|t\right]\right] = E\left[(y_1' - \hat{\beta}^T x_1')^2\right]$$

Segundo paso:

Se consideran las siguientes VA:

$$A = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\beta}^T x_i)^2$$

$$B = \frac{1}{n}\sum_{i=1}^{n}(y'_i - \tilde{\beta}^T x'_i)^2$$

Dado que y_i y y'_i, y x_i y x'_i son iid, entonces se puede afirmar que A y B tienen la misma distribución y por lo tanto $E[A] = E[B]$.

Tercer paso:

Como:

$$y'_i - \tilde{\beta}^T x'_i = (y'_i - \hat{\beta}^T x'_i) + (\hat{\beta}^T x'_i - \tilde{\beta}^T x'_i)$$

y $E[(a+b)^2] \leq E[a^2] + E[b^2]$, entonces:

$$(y'_i - \tilde{\beta}^T x'_i)^2 \leq (y'_i - \hat{\beta}^T x'_i)^2 + (\hat{\beta}^T x'_i - \tilde{\beta}^T x'_i)^2$$

Y por lo tanto:

$$\frac{1}{n}\sum_{i=1}^{n}(y'_i - \tilde{\beta}^T x'_i)^2 \leq \frac{1}{n}\sum_{i=1}^{n}(y'_i - \hat{\beta}^T x'_i)^2 + \frac{1}{n}\sum_{i=1}^{n}(\hat{\beta}^T x'_i - \tilde{\beta}^T x'_i)^2$$

Como el último término es cercano a 0, entonces:

$$B \leq \frac{1}{n}\sum_{i=1}^{n}(y'_i - \hat{\beta}^T x'_i)^2$$

Cuarto paso:

Siguiendo el paso anterior:

$$E[B] \leq E\left[\frac{1}{n}\sum_{i=1}^{n}(y'_i - \hat{\beta}^T x'_i)^2\right]$$

Por lo que queda demostrado que:

$$E\left[\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\beta}^T x_i)^2\right] \leq E\left[\frac{1}{n}\sum_{i=1}^{n}(y'_i - \hat{\beta}^T x'_i)^2\right]$$

Ejercicio 2

a) El tamaño muestral n es extremadamente grande y el número de pedictores p es pequeño.

Se espera que un método flexible funcione mejor ya que se puede ajustar mejor a la cantidad de datos.

b) El númeto de predictores p es extremadamente grande y el número de observaciones n es pequeño.

Se espera que un método inflexible funcione mejor ya que los flexibles seguramente generen sobreajuste.

c) La relación entre los predictores y la respuesta es marcadamente no lineal.

Se espera que un método flexible funcione mejor ya que este podría ajustar relaciones no lineales sin problemas.

d) La varianza del término de error $sigma^2 = V(\epsilon)$ es extremadamente alta.

Se espera que un método inflexible funcione mejor ya que los flexibles tenderán a sobreajustar.

Ejercicio 3

Modelos más flexibles tienen las ventajas de que pueden adaptarse mejor al tener mucha cantidad de variables y tienen una capacidad bastante alta para capturar relaciones complejas, por ejemplo no lineales, pero tienen las desventajas de que existe un mayor riesgo de producir sobreajustes y generalmente se precisa de una mayor cantidad de datos para su buen funcionamiento.

Por otro lado los modelos menos flexibles suelen usarse cuando existe una baja cantidad de datos, o cuando el interés principal está en poder interpretar los resultados, su desventaja principal es la baja capacidad para capturar relaciones complejas en los datos, además de que generalmente producen predictores con mayor sesgo a los flexibles.

Ejercicio 4

En clasificación se busca los “k” vecinos más cercanos al dato y se lo clasifica según la mayoría de etiquetas de esos vecinos, es decir si tiene 5 vecinos y 3 tienen una misma etiqueta y 2 otra, ese dato se clsifica como los 3 mencionados.

En regresión lo que sucede es que se toma el promedio de los “k” vecinos más cercanos para hacer una predicción sobre este.

La principal diferencia entonces es que en clasificación se usan valores discretos, mientras que en regresión se usan valores continuos.

Ejercicio 5

A)

Es probable que la SCR de la regresión cúbica sea menor o igual que la SCR de la regresión lineal ya que la regresión cúbica al ser más flexible, tiene la capacidad de ajustarse mejor a las variaciones en los datos de entrenamiento, aunque esas variaciones no representen la verdadera relación lineal.

B)

Se espera que la SCR sea menor para la regresión lineal que para la regresión cúbica ya que la regresión lineal va a captar mejor la verdadera relación entre X e Y, debido a que efectivamente esta relación es lineal.

C)

Se espera que la SCR de entrenamiento para la regresión cúbica sea menor que la SCR de entrenamiento para la regresión lineal, debido a la mayor flexibilidad de la regresión cúbica para ajustarse a un modelo no lineal.

D)

No existe suficiente información para asegurar que la SCR en el conjunto de test sea menor o mayor para la regresión cúbica que para la regresión lineal, debido a que aunque la regresión cúbica sea más flexible para captar esta relación no lineal, exite otro problema que viene por el lado del sobreajuste en el que muy seguramente caiga la regresión cúbica.

Ejercicio 6

```r
library(ggplot2)
library(tidymodels)
library(MASS)
library(dplyr)
library(parsnip)
library(splines)
library(mgcv)
library(ISLR)
library(rsample)
library(yardstick)
library(gridExtra)
library(broom)
# library(gratia)
```

```r
datos <- data.frame(
  ID = 1:20,
  Y = c(8, 9, 14, 10, 10, 15, 11, 6, 7, 8, 13, 11, 11, 10, 8, 15, 11, 4, 12, 8),
  X = c(6, 8, 12, 9, 9, 13, 11, 6, 5, 9, 13, 10, 11, 10, 8, 15, 11, 3, 11, 7)
)

pliegues <- list(
  pliegue_1 = c(4, 3, 19, 16),
  pliegue_2 = c(2, 15, 7, 18),
  pliegue_3 = c(9, 14, 12, 20),
  pliegue_4 = c(17, 6, 8, 10),
  pliegue_5 = c(1, 5, 13, 11)
)

# Paso a paso del procedimiento,
# en cada paso se deja afuera un pliegue distinto,
# se ajusta el modelos para los otros 4 pliegues
# y se testea con el que se deja afuera, calculando el MSE,
# en total cada pliegue se usa 4 veces para training y 1 vez para test

mse_list <- c()

for (i in 1:length(pliegues)) {
  # Test
  test_ids <- pliegues[[i]]

  # Training y test
  test_set <- datos[datos$ID %in% test_ids, ]
  train_set <- datos[!datos$ID %in% test_ids, ]

  # Modelo lineal
  mod <-
    parsnip::linear_reg() %>%
    parsnip::set_engine("lm")

  # Ajuste
```

```
  fit <-
    mod |>
    parsnip::fit(Y ~ X, data = train_set)

  # MSE para el pliegue
  rmse <- broom::augment(fit, new_data = test_set) %>%
                rmse(Y, .pred)

  mse <- (rmse$.estimate)^2
  mse_list <- c(mse_list, mse)
}

# Estimación del MSE por validación cruzada usando 5 pliegues (promedio de los 5 MSE's)
mse_cv <- mean(mse_list)
mse_cv
```

```
## [1] 0.8600236
```

Ejercicios ISLR

Capítulo 5 Ejercicio 8

a)

```
set.seed(1)
x <- rnorm(100)
y <- x - 2 * x^2 + rnorm(100)
```

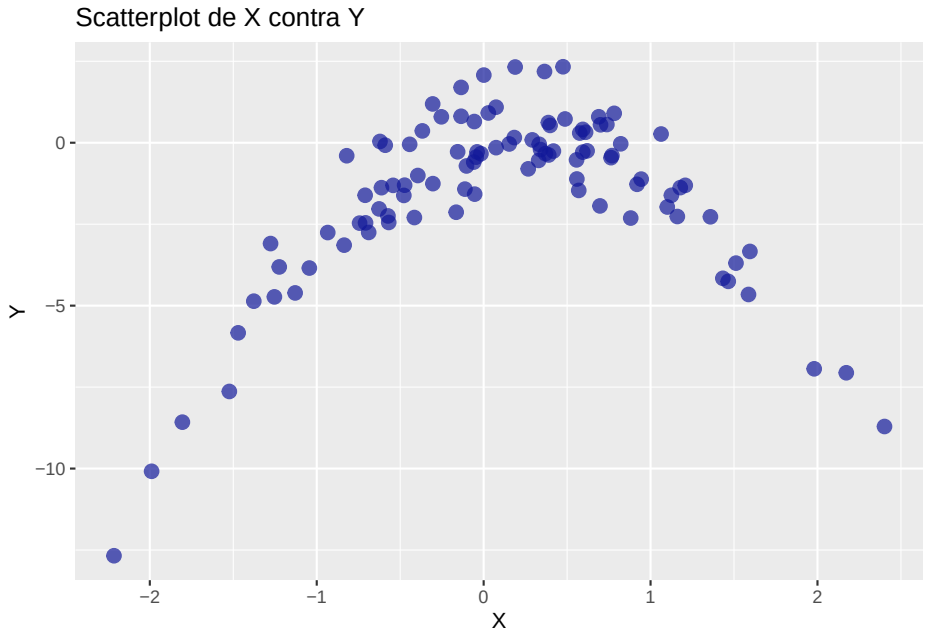
En este caso n = 100 y p = 1, y el modelo es:

$$Y = \beta_0 + \beta_1.X + \beta_2.X^2 + \epsilon$$

donde $\beta_0 = 0, \beta_1 = 1, \beta_2 = -2$ y ϵ es el error.

b)

```
ggplot(mapping = aes(x, y)) +
  geom_point(color = "#111899", size = 3, alpha = 0.7) +
  labs(title = "Scatterplot de X contra Y",
       x = "X",
       y = "Y")
```



Se observa una clara relación entre las variables, no lineal, se nota muy marcada una subida de los valores de Y en función de un aumento en X hasta un punto máximo donde la situación cambia y una vez pasado ese punto a medida que aumenta la X baja la Y. Destacar cerca del centro de los valores de X hay muchos puntos y no se aprecian tan claras estas relaciones.

c)

```
set.seed(1899)

data <- data.frame(x, y)

rmses <- c()

for (i in 1:4) {

  # Modelo lineal
  mod <- linear_reg() %>%
         set_engine("lm")

  # Ajuste
  fit <- mod %>%
         parsnip::fit(y ~ poly(x, degree = i, raw = TRUE), data = data)

  # RMSE´s
  rmses[[i]] <- augment(fit,
                        new_data = data) %>%
                rmse(y, .pred)
}

bind_rows(as.data.frame(rmses[[1]]),
          as.data.frame(rmses[[2]]),
          as.data.frame(rmses[[3]]),
          as.data.frame(rmses[[4]]))
```

```
##   .metric .estimator .estimate
## 1    rmse   standard 2.5740055
## 2    rmse   standard 0.9435524
## 3    rmse   standard 0.9431827
## 4    rmse   standard 0.9347677
```

d)

```
set.seed(2002)

rmses <- c()

for (i in 1:4) {

  # Modelo lineal
  mod <- linear_reg() %>%
         set_engine("lm")
  # Ajuste
  fit <- mod %>%
         parsnip::fit(y ~ poly(x, degree = i, raw = TRUE), data = data)
  # RMSE´s
  rmses[[i]] <- augment(fit,
                        new_data = data) %>%
                rmse(y, .pred)

}

bind_rows(as.data.frame(rmses[[1]]), as.data.frame(rmses[[2]]),
          as.data.frame(rmses[[3]]), as.data.frame(rmses[[4]]))
```

```
##   .metric .estimator .estimate
## 1    rmse   standard 2.5740055
## 2    rmse   standard 0.9435524
## 3    rmse   standard 0.9431827
## 4    rmse   standard 0.9347677
```

Son los mismos resultados, no hay aleatoriedad en esta parte.

e)

El modelo 2, el cuadrático, lo que tiene todo el sentido debido a lo que habíamos observado en el scatterplot, donde se veía una especie de parábola.

f)

```
modelo1 <- y ~ x
modelo2 <- y ~ x + I(x^2)
modelo3 <- y ~ x + I(x^2) + I(x^3)
modelo4 <- y ~ x + I(x^2) + I(x^3) + I(x^4)

summary(glm(modelo4, data = data))
```

Viendo el resumen se concuerda con la información vista en cv, los coeficientes extras asociados a los modelos 3 y 4 no son significativos, mientras que el modelo 2 ajusta mejor que el 1 siendo significativo en todos los parámetros.

Capítulo 7 Ejercicio 1

a)

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - \xi)_+^3$$

Como $x \leq \xi$, entonces:

$$f_1(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

donde $a_1 = \beta_0, b_1 = \beta_1, c_1 = \beta_2, d_1 = \beta_3$

b)

Como $x > \xi$, entonces:

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - \xi)_+^3$$

Ahora reescribimos $(x - \xi)^3$ como:

$$(x - \xi)^3 = x^3 - 3\xi x^2 + 3\xi^2 x - \xi^3$$

Entonces:

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x^3 - 3\xi x^2 + 3\xi^2 x - \xi^3)$$

Por lo tanto:

$$f(x) = (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2) x + (\beta_2 + 3\beta_4 \xi) x^2 + (\beta_3 + \beta_4) x^3$$

Donde $a_2 = (\beta_0 - \beta_4 \xi^3), b_2 = (\beta_1 + 3\beta_4 \xi^2), c_2 = (\beta_2 + 3\beta_4 \xi), d_2 = (\beta_3 + \beta_4)$

c)

$$f_1(\xi) = \beta_0 + \beta_1 \xi + \beta_2 \xi^2 + \beta_3 \xi^3$$
$$f_2(\xi) = (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2)\xi + (\beta_2 + 3\beta_4 \xi)\xi^2 + (\beta_3 + \beta_4)\xi^3$$
$$f_2(\xi) = \beta_0 - \beta_4 \xi^3 + \beta_1 \xi + 3\beta_4 \xi^3 + \beta_2 \xi^2 + 3\beta_4 \xi^3 + \beta_3 \xi^3$$
$$f_2(\xi) = \beta_0 + \beta_1 \xi + \beta_2 \xi^2 + \beta_3 \xi^3 + 2\beta_4 \xi^3$$
$$f_1(\xi) = f_2(\xi)$$

d)

$$f_1'(\xi) = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2$$
$$f_2'(\xi) = (\beta_1 + 3\beta_4 \xi^2) + 2(\beta_2 + 3\beta_4 \xi)\xi + 3(\beta_3 + \beta_4)\xi^2 = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2 + 12\beta_4 \xi^2$$
$$f_1'(\xi) = f_2'(\xi)$$

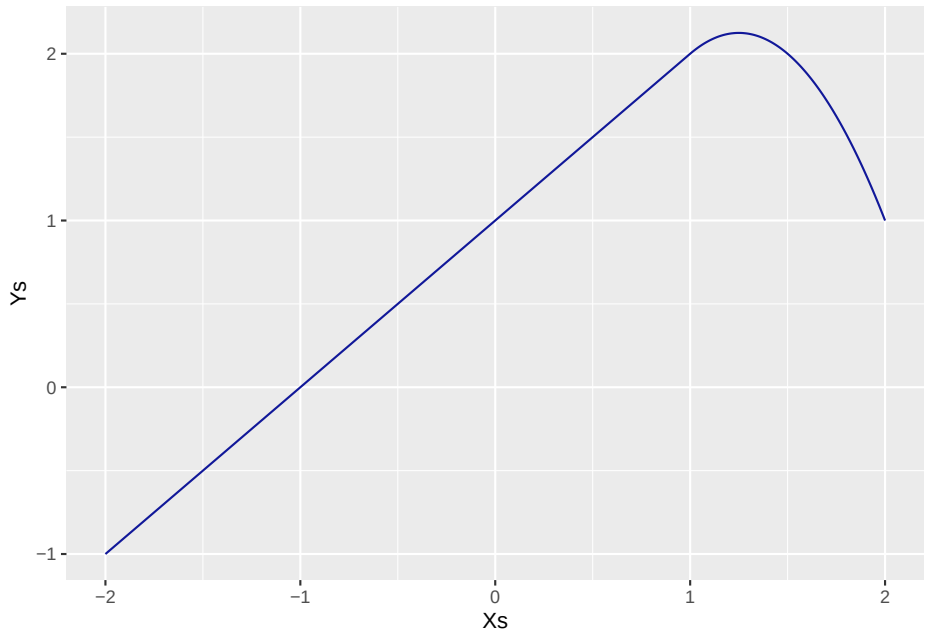
e)

$$f_1''(\xi) = 2\beta_2 + 6\beta_3 \xi$$
$$f_2''(\xi) = 2(\beta_2 + 3\beta_4 \xi) + 6(\beta_3 + \beta_4)\xi = 2\beta_2 + 6\beta_3 \xi + 6\beta_4 \xi$$
$$f_1''(\xi) = f_2''(\xi)$$

Capítulo 7 Ejercicio 2

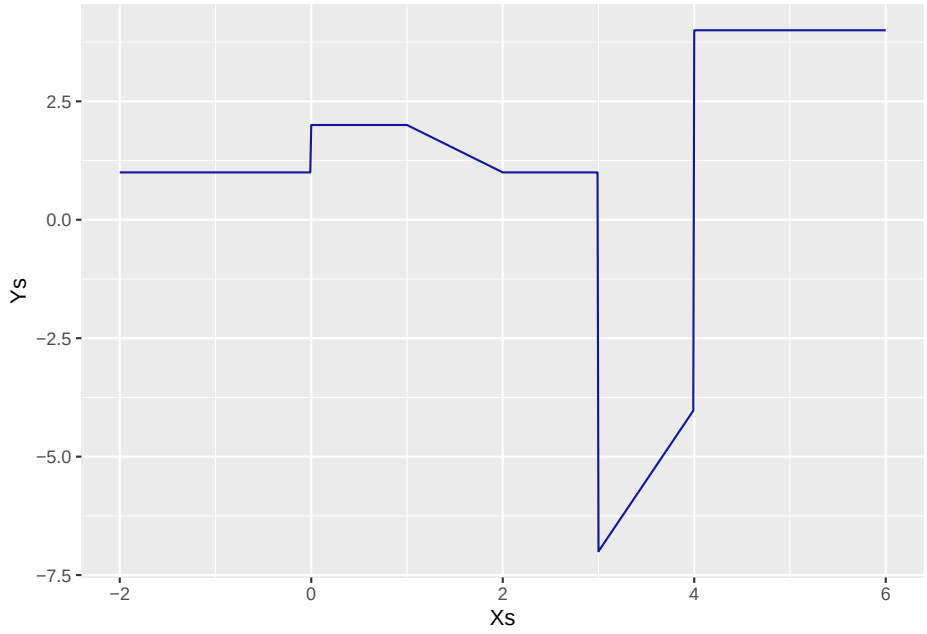
Capítulo 7 Ejercicio 3

```
b1 <- function(X) { X }
b2 <- function(X) { (X - 1)^2 * (X >= 1) }

Y_hat <- function(X) {
  ifelse(X < 1,
         1 + b1(X),
         1 + b1(X) - 2 * b2(X))
}

Xs <- seq(-2, 2, by = 0.01)
Ys <- Y_hat(Xs)

ggplot(mapping = aes(Xs, Ys)) +
  geom_line(col = "#111899")
```



Capítulo 7 Ejercicio 4

```
b1 <- function(X) { (0 <= X & X <= 2) - (X - 1) * (1 <= X & X <= 2) }
b2 <- function(X) { (X - 3) * (3 <= X & X <= 4) + (4 < X & X <= 5) }

Y_hat <- function(X) {
  ifelse(X < 0, 1,
         ifelse(X < 1, 2,
                ifelse(X <= 2,3 - X,
                       ifelse(X < 3, 1,
                              ifelse(X < 4, 3 * (X - 3) + 1 - 8,4))))
         )
}

Xs <- seq(-2, 6, by = 0.01)
Ys <- Y_hat(Xs)

ggplot(mapping = aes(Xs, Ys)) +
  geom_line(col = "#111899")
```



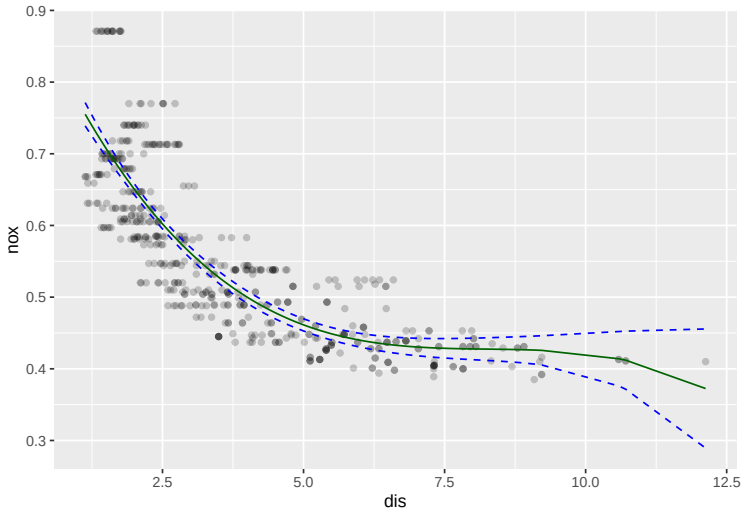
Capítulo 7 Ejercicio 9

a)

```
boston_data <- Boston

lm_fit <-
  parsnip::linear_reg() %>%
  parsnip::set_engine("lm") %>%
  parsnip::fit(nox ~ poly(dis, degree = 3, raw = TRUE),
               data = boston_data)

tidy(lm_fit)
```

```
## # A tibble: 4 x 5
##   term                                 estimate std.error statistic   p.value
##   <chr>                                   <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)                           0.934     0.0207       45.1  9.85e-179
## 2 poly(dis, degree = 3, raw = TRUE)1 -0.182     0.0147      -12.4  6.08e- 31
## 3 poly(dis, degree = 3, raw = TRUE)2  0.0219    0.00293       7.48 3.43e- 13
## 4 poly(dis, degree = 3, raw = TRUE)3 -0.000885  0.000173     -5.12 4.27e-  7
```

```
regression_lines <- bind_cols(
    augment(lm_fit, new_data = boston_data),
    predict(lm_fit, new_data = boston_data, type = "conf_int")
)

boston_data %>%
     ggplot(aes(dis, nox)) +
     geom_point(alpha = 0.2) +
     geom_line(aes(y = .pred), color = "darkgreen",
               data = regression_lines) +
     geom_line(aes(y = .pred_lower), data = regression_lines,
               linetype = "dashed", color = "blue") +
     geom_line(aes(y = .pred_upper), data = regression_lines,
               linetype = "dashed", color = "blue")
```

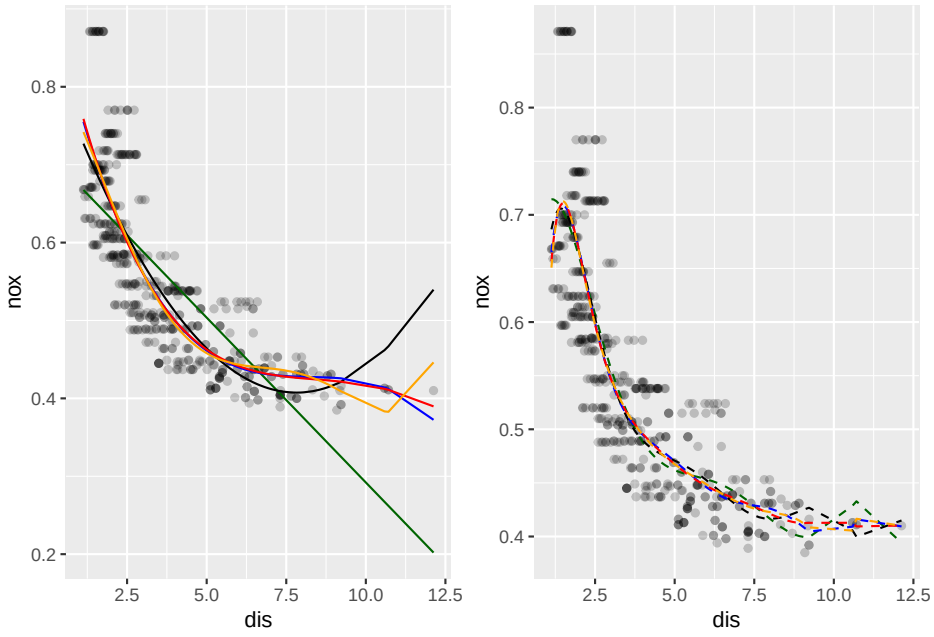


b)

```
results <- c()
regression_lines <- list()

for (degree in 1:10) {

  # Ajuste
  lm_fit <-
    parsnip::linear_reg() %>%
    parsnip::set_engine("lm") %>%
    parsnip::fit(nox ~ poly(dis, degree = degree, raw = TRUE),
                 data = boston_data)

  # MSE´s
  rmse <- broom::augment(lm_fit, new_data = boston_data) %>%
                 rmse(nox, .pred)

  mse <- (rmse$.estimate)^2
  results <- c(results, mse)

  regression_lines[[degree]] <- augment(lm_fit,
                                        new_data = boston_data)

}

results_df <- data.frame(Grados = 1:10, MSE = results)
results_df
```

```
##    Grados         MSE
## 1       1 0.005471468
## 2       2 0.004022257
## 3       3 0.003822345
## 4       4 0.003820121
## 5       5 0.003785158
## 6       6 0.003711971
## 7       7 0.003655106
## 8       8 0.003627727
## 9       9 0.003623183
## 10     10 0.003620892
```

```
g1 <- boston_data %>%
     ggplot(aes(dis, nox)) +
     geom_point(alpha = 0.2) +
     geom_line(aes(y = .pred), color = "darkgreen",
               data = regression_lines[[1]]) +
     geom_line(aes(y = .pred), color = "black",
               data = regression_lines[[2]]) +
     geom_line(aes(y = .pred), color = "blue",
               data = regression_lines[[3]]) +
     geom_line(aes(y = .pred), color = "red",
               data = regression_lines[[4]]) +
     geom_line(aes(y = .pred), color = "orange",
               data = regression_lines[[5]])

g2 <- boston_data %>%
     ggplot(aes(dis, nox)) +
     geom_point(alpha = 0.2) +
     geom_line(aes(y = .pred), color = "darkgreen", linetype = 2,
               data = regression_lines[[6]]) +
     geom_line(aes(y = .pred), color = "black", linetype = 2,
               data = regression_lines[[7]]) +
     geom_line(aes(y = .pred), color = "blue", linetype = 2,
               data = regression_lines[[8]]) +
     geom_line(aes(y = .pred), color = "red", linetype = 2,
               data = regression_lines[[9]]) +
     geom_line(aes(y = .pred), color = "orange", linetype = 2,
               data = regression_lines[[10]])

grid.arrange(g1, g2, ncol = 2)
```



c)

```
set.seed(1899)

grados <- 1:10
results <- tibble(degree = grados, rmse = numeric(10))
folds <- vfold_cv(boston_data, v = 5)

for (i in grados) {
  rmses <- c()

  for (f in folds$splits) {

    # Divido en train y test
    train_data <- training(f)
    test_data <- testing(f)

    # Modelo lineal
    lm_mod <-
      parsnip::linear_reg() %>%
      parsnip::set_engine("lm")

    # Ajuste
    lm_fit <-
      lm_mod %>%
      parsnip::fit(nox ~ poly(dis, i), data = train_data)

    # RMSE´s
    rmse_value <- broom::augment(lm_fit,
                                 new_data = train_data) %>%
                          rmse(nox, .pred)

    rmses <- c(rmses, rmse_value$.estimate)
  }

  results$rmse[i] <- mean(rmses)
}

results
```

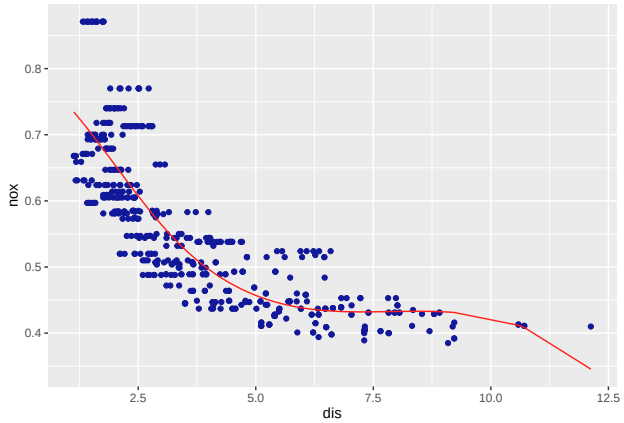
```
## # A tibble: 10 x 2
##    degree   rmse
##     <int>  <dbl>
##  1      1 0.0739
##  2      2 0.0634
##  3      3 0.0618
##  4      4 0.0618
##  5      5 0.0614
##  6      6 0.0608
##  7      7 0.0604
##  8      8 0.0602
##  9      9 0.0601
## 10     10 0.0601
```

d)

```
spline_fit <-
  parsnip::linear_reg() %>%
  parsnip::set_engine("lm") %>%
  parsnip::fit(nox ~ bs(dis, df = 4), data = boston_data)

spline_fit %>%
  extract_fit_engine() %>%
  summary()
```

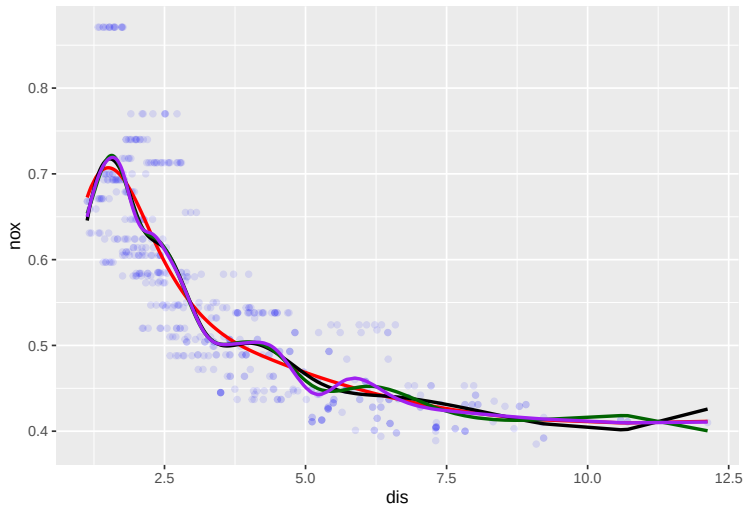
```
##
## Call:
## stats::lm(formula = nox ~ bs(dis, df = 4), data = data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124622 -0.039259 -0.008514  0.020850  0.193891
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.73447    0.01460  50.306  < 2e-16 ***
## bs(dis, df = 4)1 -0.05810    0.02186  -2.658  0.00812 **
## bs(dis, df = 4)2 -0.46356    0.02366 -19.596  < 2e-16 ***
## bs(dis, df = 4)3 -0.19979    0.04311  -4.634 4.58e-06 ***
## bs(dis, df = 4)4 -0.38881    0.04551  -8.544  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06195 on 501 degrees of freedom
## Multiple R-squared:  0.7164, Adjusted R-squared:  0.7142
## F-statistic: 316.5 on 4 and 501 DF,  p-value: < 2.2e-16
```

```
boston_data$predicts_sp <- predict(spline_fit, new_data = boston_data)$`.pred`

ggplot(boston_data, aes(x = dis, y = nox)) +
  geom_point(color = "#111899") +
  geom_line(aes(y = predicts_sp), color = "#FF2119")
```



e)

```
SCR_results <- data.frame(df = integer(), SCR = numeric())

for (df in 5:20) {
  spline_fit <-
    parsnip::linear_reg() %>%
    parsnip::set_engine("lm") %>%
    parsnip::fit(nox ~ bs(dis, df = df), data = boston_data)

  predictions <- predict(spline_fit, new_data = boston_data)$`.pred`

  model_summary <- glance(spline_fit)
  SCR <- sum(model_summary$sigma^2 * (model_summary$nobs - 1))

  SCR_results <- bind_rows(SCR_results, data.frame(df = df, SCR = SCR))

  boston_data[[paste0("predicts__", df)]] <- predictions
}

ggplot(boston_data, aes(x = dis, y = nox)) +
  geom_point(color = "blue", alpha = 0.1) +
  geom_line(aes(y = predicts__5), color = "red", size = 1) +
  geom_line(aes(y = predicts__10), color = "black", size = 1) +
  geom_line(aes(y = predicts__14), color = "darkgreen", size = 1) +
  geom_line(aes(y = predicts__18), color = "purple", size = 1)
```



```
SCR_results
```

```
##    df      SCR
## 1   5 1.858575
## 2   6 1.856018
## 3   7 1.855606
## 4   8 1.846242
## 5   9 1.858779
## 6  10 1.828748
## 7  11 1.837006
## 8  12 1.832545
## 9  13 1.829445
## 10 14 1.832644
## 11 15 1.837373
## 12 16 1.841903
## 13 17 1.841790
## 14 18 1.841475
## 15 19 1.843860
## 16 20 1.849994
```

f)

```
set.seed(1899)

df <- 5:20
results <- tibble(DF = df, RMSE = numeric(16))

folds <- vfold_cv(boston_data, v = 5)

for (i in seq_along(df)) {
  rmses <- c()

  for (f in folds$splits) {

    # Divido en train y test
    train_data <- training(f)
    test_data <- testing(f)

    # Modelo
    spline_fit <-
      parsnip::linear_reg() %>%
      parsnip::set_engine("lm") %>%
      parsnip::fit(nox ~ bs(dis, df = df[i]), data = train_data)

    # RMSE´s
    rmse_value <- broom::augment(spline_fit,
                                 new_data = train_data) %>%
                          rmse(nox, .pred)

    rmses <- c(rmses, rmse_value$.estimate)
  }
```

```
  results$RMSE[i] <- mean(rmses)
}

results
```

```
## # A tibble: 16 x 2
##       DF   RMSE
##    <int>  <dbl>
##  1     5 0.0603
##  2     6 0.0601
##  3     7 0.0601
##  4     8 0.0598
##  5     9 0.0599
##  6    10 0.0594
##  7    11 0.0595
##  8    12 0.0593
##  9    13 0.0592
## 10    14 0.0592
## 11    15 0.0592
## 12    16 0.0592
## 13    17 0.0591
## 14    18 0.0590
## 15    19 0.0590
## 16    20 0.0590
```

Capítulo 7 Ejercicio 10

a)

```
college_data <- College

set.seed(1899)

college_split <- initial_split(college_data,
                               prop = 3/4)

college_train <- training(college_split)
college_test <- testing(college_split)

mod <-
  parsnip::linear_reg() %>%
  parsnip::set_engine("lm")

fit_nulo <-
  mod |>
  parsnip::fit(Outstate ~ 1, data = college_train)

######## NO FUNCIONA ########
# modelo_fwd <- step(fit_nulo$fit,
#                    scope = list(lower = fit_nulo$fit,
#                                 upper = ~ .,
#                                 data = college_train),
#                    direction = "forward")
#
# modelo_fwd$fit %>%
#   summary()
#############################

fit <-
  mod |>
  parsnip::fit(Outstate ~ Private + Apps + Accept + Top10perc + Room.Board +
                        Terminal + perc.alumni + Expend + Grad.Rate, data = college_train)

fit$fit %>%
  summary()
```

```
##
## Call:
## stats::lm(formula = Outstate ~ Private + Apps + Accept + Top10perc +
##     Room.Board + Terminal + perc.alumni + Expend + Grad.Rate,
##     data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8083.2 -1287.5  -102.5  1356.0  5931.5
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -3.991e+03  5.635e+02  -7.084 4.14e-12 ***
## PrivateYes   2.791e+03  2.690e+02  10.377  < 2e-16 ***
## Apps        -3.248e-01  7.790e-02  -4.170 3.52e-05 ***
## Accept       5.211e-01  1.166e-01   4.471 9.40e-06 ***
## Top10perc    1.736e+01  7.553e+00   2.298   0.0219 *
## Room.Board   9.576e-01  1.002e-01   9.553  < 2e-16 ***
## Terminal     3.309e+01  7.383e+00   4.482 8.93e-06 ***
## perc.alumni  4.231e+01  9.008e+00   4.698 3.30e-06 ***
## Expend       2.172e-01  2.356e-02   9.219  < 2e-16 ***
## Grad.Rate    3.069e+01  6.297e+00   4.874 1.42e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2034 on 572 degrees of freedom
## Multiple R-squared:  0.7545, Adjusted R-squared:  0.7506
## F-statistic: 195.3 on 9 and 572 DF,  p-value: < 2.2e-16
```

```
rmse_linear <- augment(fit,
                       new_data = college_train) %>%
               rmse(Outstate, .pred)

rmse_linear
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       2017.
```

b)

```
rec_gam <- recipes::recipe(Outstate ~ Private + Apps + Accept + Top10perc + Room.Board +
                                      Terminal + perc.alumni + Expend + Grad.Rate,
                           data = college_train)

mgcv_spec <- parsnip::gen_additive_mod() %>%
  parsnip::set_engine("mgcv") %>%
  parsnip::set_mode("regression")

gam_wf <- workflows::workflow() %>%
  workflows::add_recipe(rec_gam) %>%
  workflows::add_model(mgcv_spec, formula = Outstate ~ Private + Apps + Accept + Top10perc + Room.Board
                                      Terminal + perc.alumni + Expend + Grad.Rate)

gam_fit <- parsnip::fit(gam_wf, data = college_train)
gam <- extract_fit_engine(gam_fit)

# gratia::draw(gam, residuals = T)
```

c)

```
rmse_gam_test <- augment(gam_fit,
                         new_data = college_test) %>%
                rmse(Outstate, .pred)

rmse_gam_test
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       1909.
```

d)

```
gam %>%
  summary()
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Outstate ~ Private + Apps + Accept + Top10perc + Room.Board +
##     Terminal + perc.alumni + Expend + Grad.Rate
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.991e+03  5.635e+02  -7.084 4.14e-12 ***
## PrivateYes   2.791e+03  2.690e+02  10.377  < 2e-16 ***
## Apps        -3.248e-01  7.790e-02  -4.170 3.52e-05 ***
## Accept       5.211e-01  1.166e-01   4.471 9.40e-06 ***
## Top10perc    1.736e+01  7.553e+00   2.298   0.0219 *
## Room.Board   9.576e-01  1.002e-01   9.553  < 2e-16 ***
## Terminal     3.309e+01  7.383e+00   4.482 8.93e-06 ***
## perc.alumni  4.231e+01  9.008e+00   4.698 3.30e-06 ***
## Expend       2.172e-01  2.356e-02   9.219  < 2e-16 ***
## Grad.Rate    3.069e+01  6.297e+00   4.874 1.42e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.751   Deviance explained = 75.4%
## GCV = 4.211e+06  Scale est. = 4.1386e+06  n = 582
```