Seleccion de modelo Pt 1

Fernando Anorve

3/23/2020

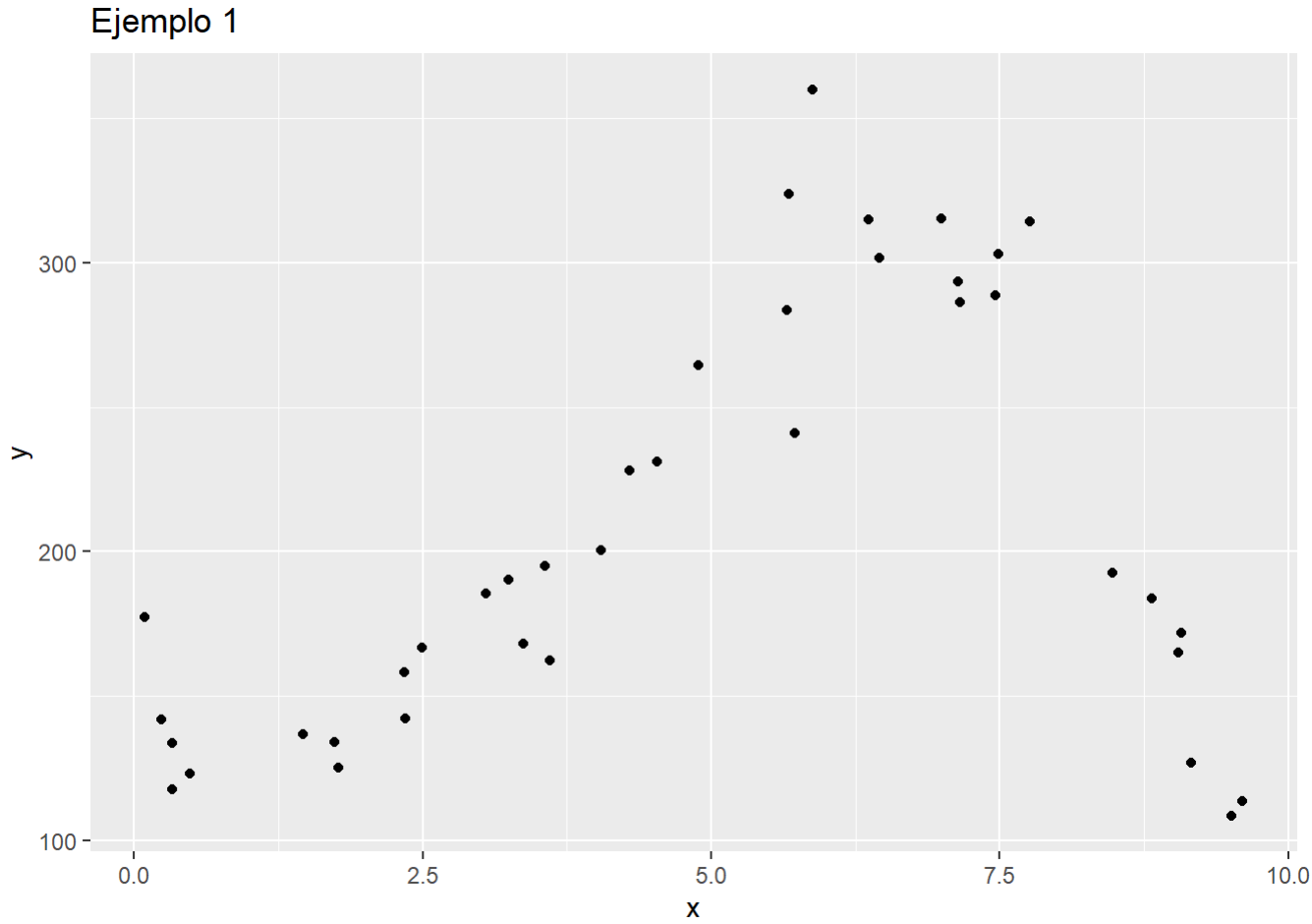
Ejemplo 1

Compensación entre sesgo y varianza

El primer ejemplo a estudiar es más bien hipotético, para entender la importancia de una selección adecuada del modelo.

El conjunto de datos en el archivo “ejemplo1.txt” contiene dos columnas, llamadas x, y. El objetivo en este caso es tratar de predecir la variable y utilizando el predictor x.

```
ej1 <- read.csv("ejemplo1.txt")

ggplot(data = ej1, aes(x,y)) + geom_point() + ggtitle("Ejemplo 1")
```



Evidentemente, este no es un modelo linear con respecto a x.

Probablemente habría que incorporar modelos cuadráticos o incluso de grados más altos.

$$\mathbb{E}[Y] = \beta_0 + \beta_1 x + \cdots + \beta_m x^m$$

¿Qué valor de m deberíamos usar?

- Agregamos términos de orden mayor:

```
ej1$x2 <- ej1$x^2
ej1$x3 <- ej1$x^3
ej1$x4 <- ej1$x^4
ej1$x5 <- ej1$x^5
ej1$x6 <- ej1$x^6
ej1$x7 <- ej1$x^7
```

- Tomamos provisionalmente un subconjunto aleatorio (luego veremos por qué):

```
n = length(ej1$x)
# set.seed(777)
# set.seed(8)
# set.seed(128)
set.seed(4096)
test_set <- sample(n,round(n/3))

ej1_test <- ej1[test_set,]
ej1 <- ej1[-test_set,]
```

- Ajustamos distintos modelos de diferente orden:

```
lm_1 <- lm(y ~ x , data = ej1)
lm_2 <- lm(y ~ x + x2 , data = ej1)
lm_3 <- lm(y ~ x + x2 + x3, data = ej1)
lm_4 <- lm(y ~ x + x2 + x3 + x4, data = ej1)
lm_5 <- lm(y ~ x + x2 + x3 + x4 + x5, data = ej1)
lm_6 <- lm(y ~ x + x2 + x3 + x4 + x5 + x6, data = ej1)
lm_7 <- lm(y ~ x + x2 + x3 + x4 + x5 + x6 + x7, data = ej1)
```

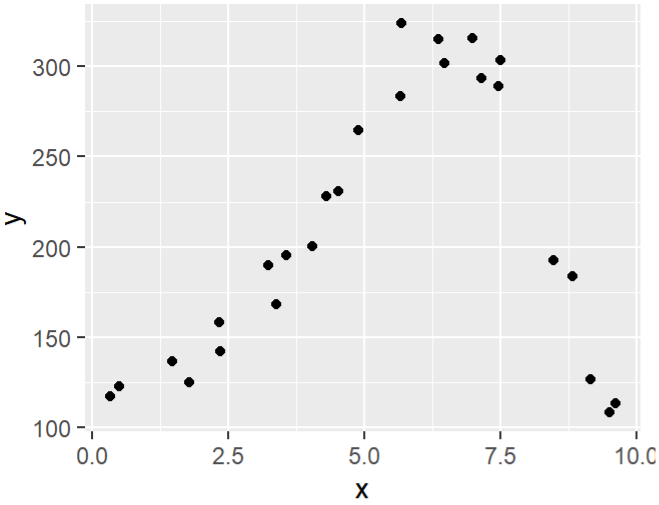
- Creamos la curva de ajuste de cada uno de los modelos, la x va de 0 a 10, en “saltos” de 0.1

```
fitted_line <- data.frame("x" = seq(0,10,by = 0.1))
fitted_line$x2 = fitted_line$x^2
fitted_line$x3 = fitted_line$x^3
fitted_line$x4 = fitted_line$x^4
fitted_line$x5 = fitted_line$x^5
fitted_line$x6 = fitted_line$x^6
fitted_line$x7 = fitted_line$x^7

fitted_line$y <- predict(lm_1,newdata = fitted_line)
fitted_line$y2 <- predict(lm_2,newdata = fitted_line)
fitted_line$y3 <- predict(lm_3,newdata = fitted_line)
fitted_line$y4 <- predict(lm_4,newdata = fitted_line)
fitted_line$y5 <- predict(lm_5,newdata = fitted_line)
fitted_line$y6 <- predict(lm_6,newdata = fitted_line)
fitted_line$y7 <- predict(lm_7,newdata = fitted_line)
```

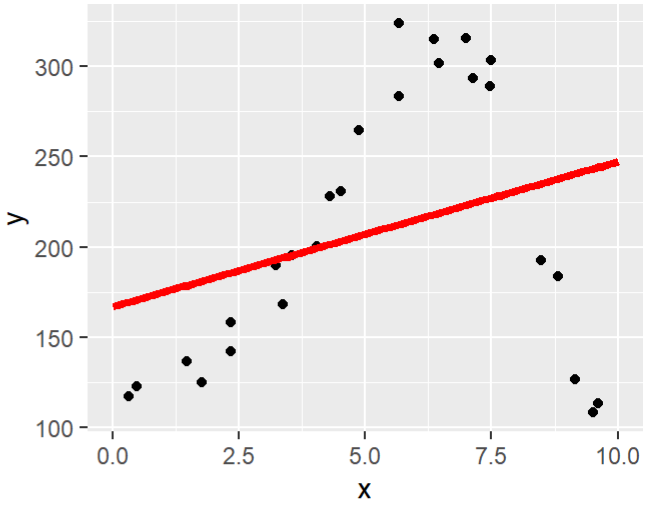
- Finalmente, graficamos los modelos

```
plt0 = ggplot(data = ej1, aes(x,y)) + ggtitle("Scatterplot") + geom_point()

plt1 = ggplot(data = ej1, aes(x,y)) + ggtitle("Modelo lineal") + geom_point() + geom_line(aes(y
= y), colour = "red", data = fitted_line, size = 1.5)

plt2 = ggplot(data = ej1, aes(x,y)) + ggtitle("Modelo cuadratico") + geom_point() + geom_line(ae
s(y = y2), colour = "red", data = fitted_line, size = 1.5)

plt3 = ggplot(data = ej1, aes(x,y)) + ggtitle("Modelo cubico") + geom_point() + geom_line(aes(y
= y3), colour = "red", data = fitted_line, size = 1.5)

plt4 = ggplot(data = ej1, aes(x,y)) + ggtitle("Modelo cuartico") + geom_point() + geom_line(aes
(y = y4), colour = "red", data = fitted_line, size = 1.5)

plt5 = ggplot(data = ej1, aes(x,y)) + ggtitle("Modelo m = 5") + geom_point() + geom_line(aes(y =
y5), colour = "red", data = fitted_line, size = 1.5)

plt6 = ggplot(data = ej1, aes(x,y)) + ggtitle("Modelo m = 6") + geom_point() + geom_line(aes(y =
y6), colour = "red", data = fitted_line, size = 1.5)

plt7 = ggplot(data = ej1, aes(x,y)) + ggtitle("Modelo m = 7") + geom_point() + geom_line(aes(y =
y7), colour = "red", data = fitted_line, size = 1.5)

grid.arrange(plt0,plt1, plt2, plt3 , plt4,plt5,plt6,plt7, ncol = 2)
```

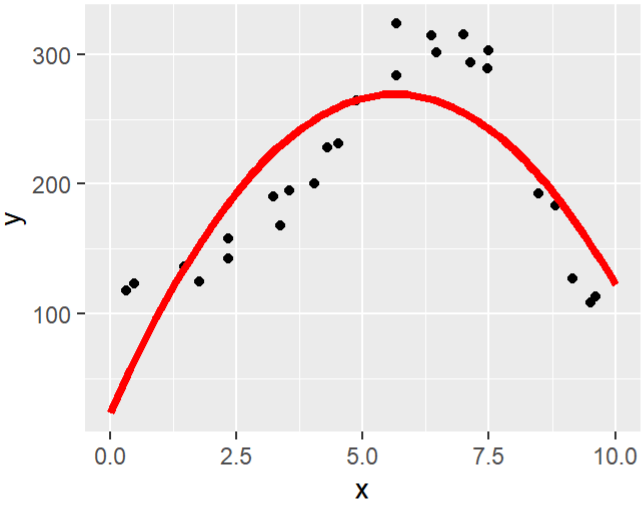
Scatterplot



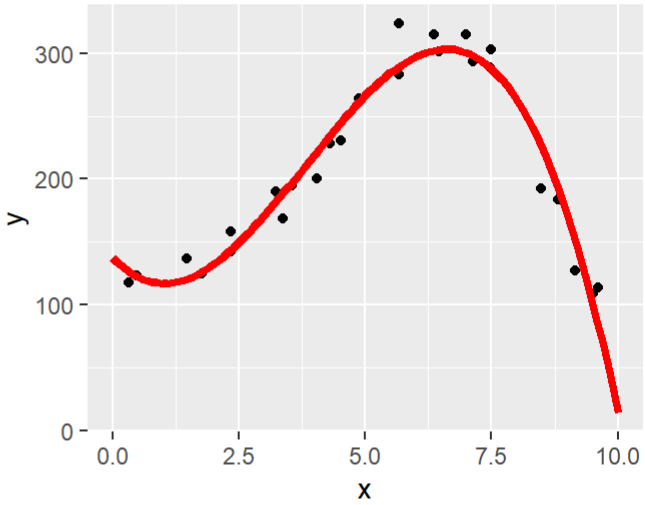
Modelo lineal



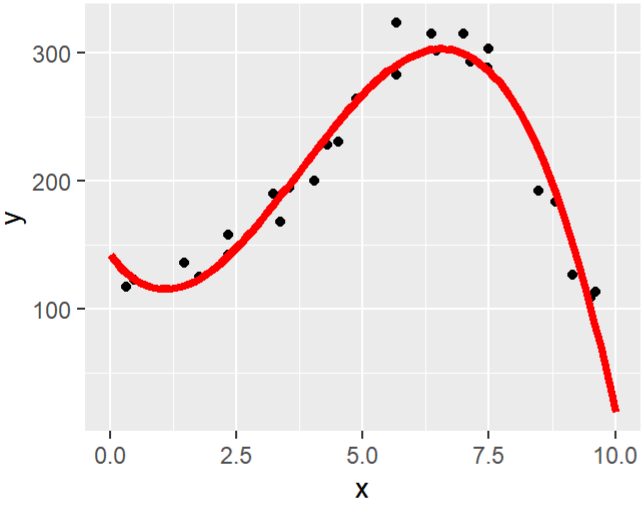
Modelo cuadratico



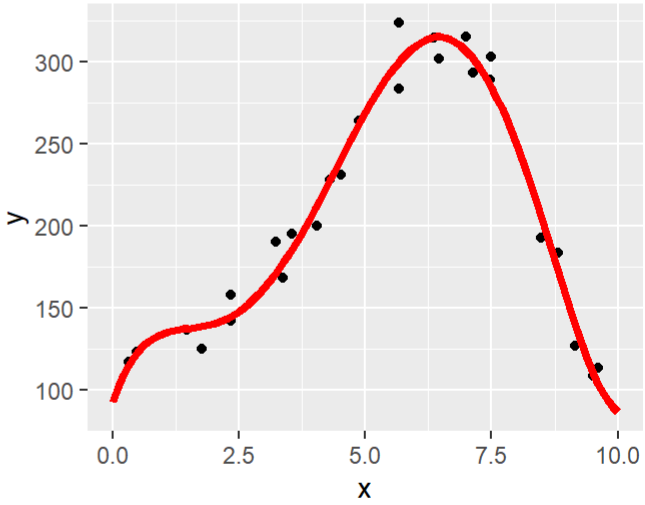
Modelo cubico



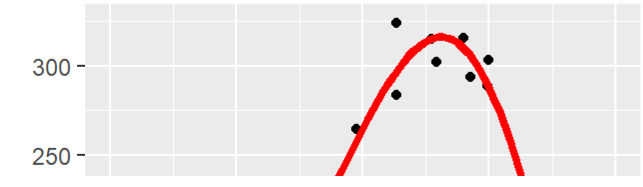
Modelo cuartico



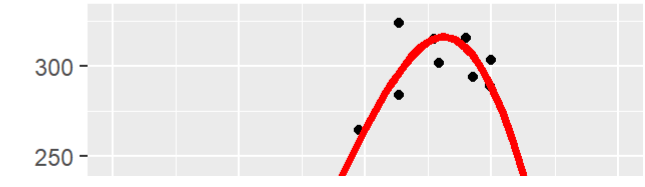
Modelo m = 5

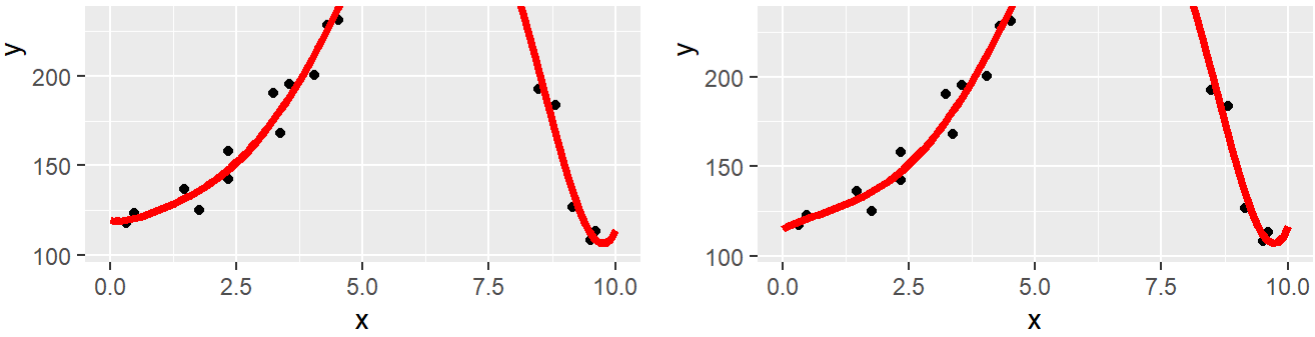


Modelo m = 6



Modelo m = 7





¿Qué puede observarse respecto a los valores de m?

El modelo cúbico parece explicar el fenómeno bastante bien. De hecho, los modelos empiezan a complicarse más, probablemente de forma innecesaria. Podemos notar algunas inflexiones en la curva que van cambiando conforme aumenta m. Al final, la curva se ajusta casi exactamente a los puntos de la gráfica de dispersión.

Esto parece ser bueno, ¿no?

No necesariamente. Ajustarse de esta manera a los datos (conocido como “sobreajuste”), puede ser malo porque el modelo depende demasiado de la muestra con la que se está trabajando.

En este caso, ¿qué pasa si agregamos más datos del mismo fenómeno? Desearíamos que el modelo de predicción también se ajustara a estos nuevos datos. Por lo tanto, quisieramos que al ajustar un modelo con estos nuevos datos, éste fuera más o menos parecido al anterior

Comparemos qué ocurre con $m = 3$ y $m = 7$

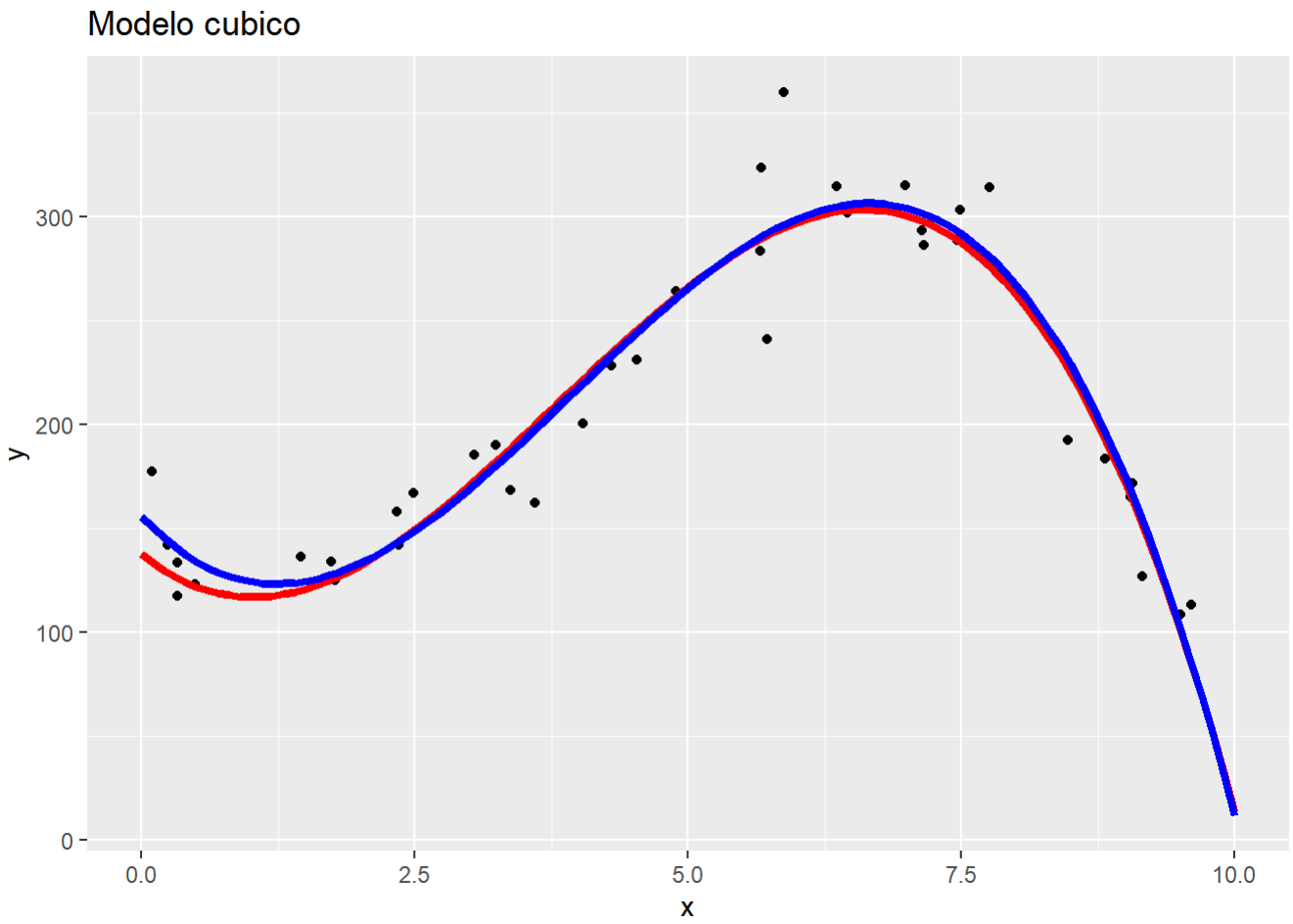
```
lm_3_new <- lm(y ~ x + x2 + x3, data = rbind(ej1,ej1_test))
summary(lm_3)
```

```
##
## Call:
## lm(formula = y ~ x + x2 + x3, data = ej1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.021  -7.946  -1.291  11.384  34.171
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 137.5840    14.6769   9.374 3.86e-09 ***
## x           -42.9064    12.0600  -3.558  0.00176 **
## x2           24.3884     2.7675   8.812 1.14e-08 ***
## x3           -2.1334     0.1813 -11.766 5.80e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.88 on 22 degrees of freedom
## Multiple R-squared:  0.9553, Adjusted R-squared:  0.9492
## F-statistic: 156.8 on 3 and 22 DF,  p-value: 5.352e-15
```

```
summary(lm_3_new)
```

```
##
## Call:
## lm(formula = y ~ x + x2 + x3, data = rbind(ej1, ej1_test))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.146 -12.074  -2.009  10.922  63.836
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 155.8966    11.8356  13.172 3.99e-15 ***
## x           -56.3884    10.9423  -5.153 1.01e-05 ***
## x2           27.1391     2.6991  10.055 7.34e-12 ***
## x3           -2.2939     0.1845 -12.433 2.13e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.34 on 35 degrees of freedom
## Multiple R-squared:  0.9151, Adjusted R-squared:  0.9079
## F-statistic: 125.8 on 3 and 35 DF,  p-value: < 2.2e-16
```

```
fitted_line_new = fitted_line

fitted_line_new$y3 <- predict(lm_3_new,newdata = fitted_line)

ggplot(data = rbind(ej1,ej1_test), aes(x,y)) + ggtitle("Modelo cubico") + geom_point() + geom_li
ne(aes(y = y3), colour = "red", data = fitted_line, size = 1.5) + geom_line(aes(y = y3), colour
= "blue", data = fitted_line_new, size = 1.5)
```



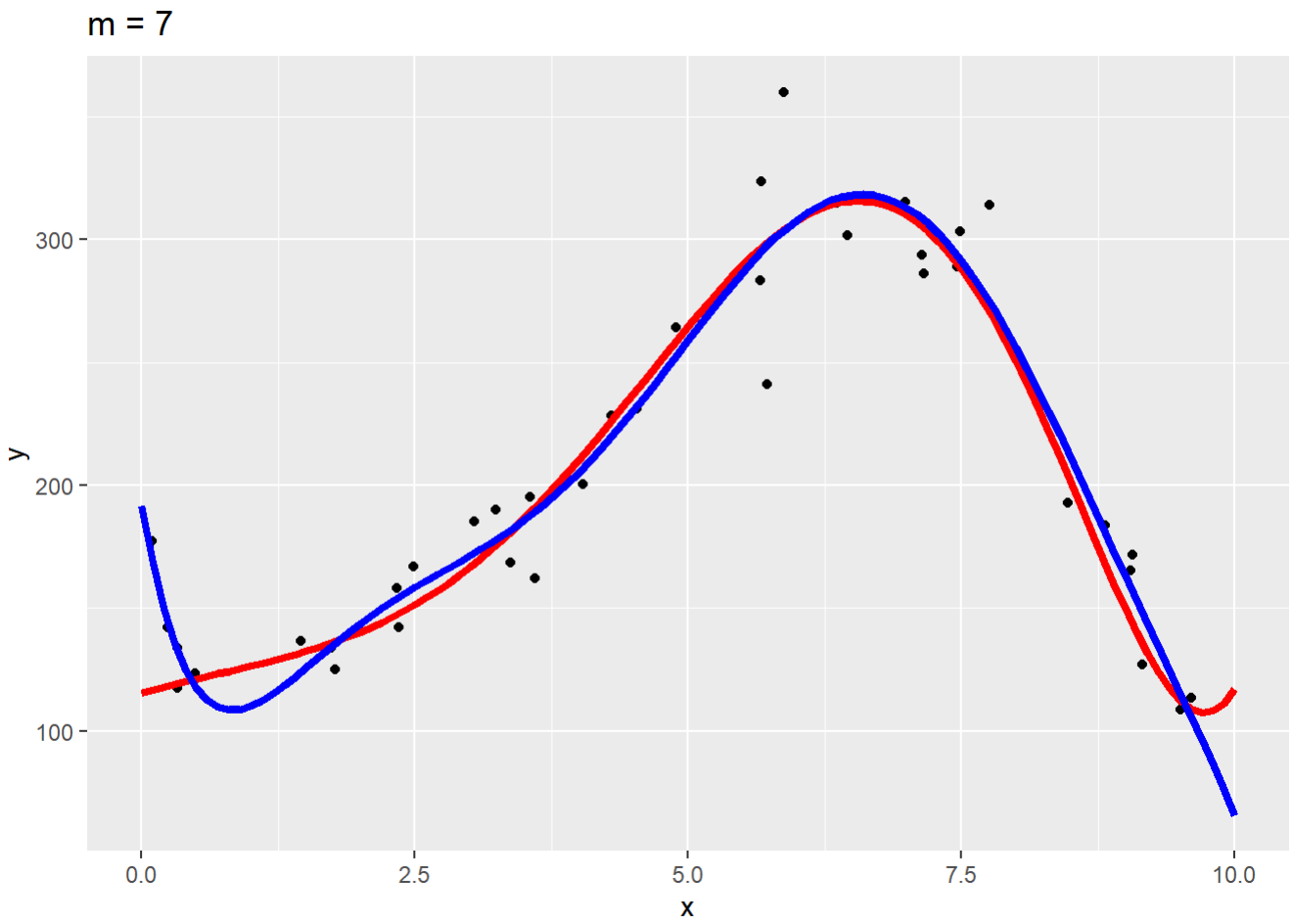

```
lm_7_new <- lm(y ~ x + x2 + x3 + x4 + x5 + x6 + x7, data = rbind(ej1,ej1_test))
summary(lm_7)
```

```
##
## Call:
## lm(formula = y ~ x + x2 + x3 + x4 + x5 + x6 + x7, data = ej1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.8419 -10.1225  -0.7259   5.6881  27.0844
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.154e+02  4.568e+01   2.526   0.0211 *
## x            1.221e+01  1.674e+02   0.073   0.9427
## x2          -1.709e+00  1.777e+02  -0.010   0.9924
## x3           5.164e-03  8.522e+01   0.000   1.0000
## x4           5.956e-01  2.139e+01   0.028   0.9781
## x5          -6.572e-02  2.914e+00  -0.023   0.9823
## x6          -5.351e-03  2.037e-01  -0.026   0.9793
## x7           6.012e-04  5.726e-03   0.105   0.9175
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.67 on 18 degrees of freedom
## Multiple R-squared:  0.9794, Adjusted R-squared:  0.9714
## F-statistic: 122.3 on 7 and 18 DF,  p-value: 7.211e-14
```

```
summary(lm_7_new)
```

```
##
## Call:
## lm(formula = y ~ x + x2 + x3 + x4 + x5 + x6 + x7, data = rbind(ej1,
##     ej1_test))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -56.42 -11.52   1.67   8.66  56.23
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.917e+02  2.669e+01   7.181  4.5e-08 ***
## x           -2.611e+02  1.241e+02  -2.103   0.0436 *
## x2           2.849e+02  1.443e+02   1.974   0.0574 .
## x3          -1.349e+02  7.338e+01  -1.838   0.0757 .
## x4           3.361e+01  1.924e+01   1.747   0.0906 .
## x5          -4.444e+00  2.711e+00  -1.640   0.1112
## x6           2.934e-01  1.948e-01   1.506   0.1421
## x7          -7.623e-03  5.599e-03  -1.361   0.1832
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.96 on 31 degrees of freedom
## Multiple R-squared:  0.9338, Adjusted R-squared:  0.9189
## F-statistic: 62.49 on 7 and 31 DF,  p-value: < 2.2e-16
```

```
fitted_line_new$y7 <- predict(lm_7_new,newdata = fitted_line)

ggplot(data = rbind(ej1,ej1_test), aes(x,y)) + ggtitle("m = 7") + geom_point() + geom_line(aes(y = y7), colour = "red", data = fitted_line, size = 1.5) + geom_line(aes(y = y7), colour = "blue", data = fitted_line_new, size = 1.5)
```



Nota: ¿En realidad la curva del modelo necesita tantos puntos de inflexión?

Conclusión

- Entre menos covariables se usen, más rígido es el modelo
- Entre más covariables se usen, el modelo es mucho más flexible y se ajusta “mejor” a los datos muestrales

$$\mathbf{Y} = \mathbf{X} \cdot \beta + \varepsilon$$

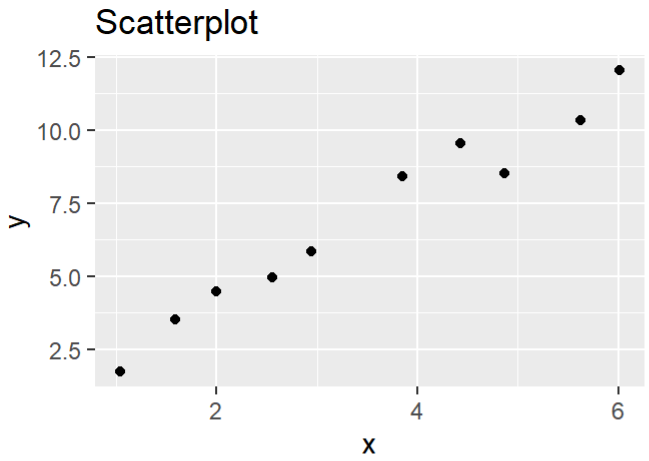
- De hecho, si n = p, y $\mathbf{X}$ tiene rango completo, la igualdad será exacta $\mathbf{Y} = \mathbf{X} \cdot \hat{\beta}$. Es decir, nuestro modelo “pasa por todos los puntos”
    - Si los datos cambian, el modelo tiene mucha libertad para cambiar y es inestable
    - No se puede estimar σ
- Si no incluimos suficientes predictores, no vamos a poder estimar bien $\mathbb{E}[Y]$ sin importar qué tan grande sea n (sesgo)
- Si incluimos predictores innecesarios, nuestra estimación se vuelve complicada y potencialmente inestable (varianza del estimador)

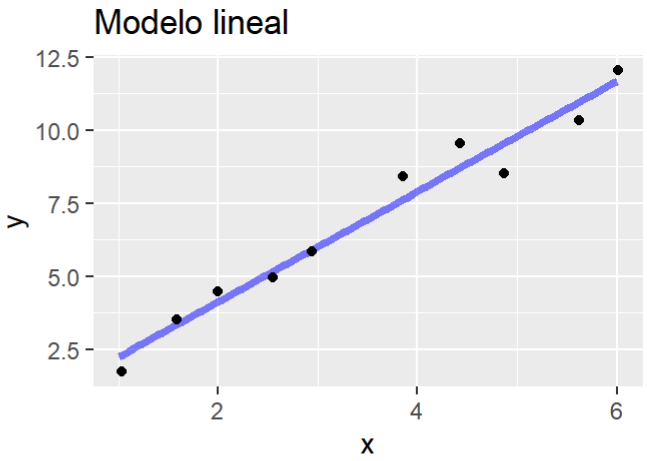
¡Tenemos que decidir los predictores en el modelo de manera inteligente!

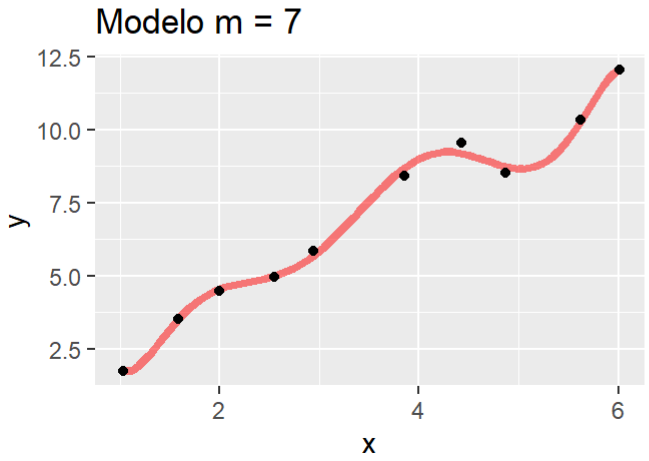
Un ejemplo más extremo

```
set.seed(2020)

x <- seq(1,6,length.out = 10) + rnorm(10,sd = 0.1)
y <- 2*x + rnorm(10,sd = 0.4)

plt0 <- ggplot(mapping = aes(x,y)) + geom_point() + ggtitle("Scatterplot")
```

```
x2 <- x^2
x3 <- x^3
x4 <- x^4
x5 <- x^5
x6 <- x^6
x7 <- x^7

lm_1 <- lm(y ~ x)

lm_7 <- lm(y ~ x + x2 + x3 + x4 + x5 + x6 + x7)

fitted_line <- data.frame("x" = seq(1,6,by = 0.05))

fitted_line$x2 = fitted_line$x^2
fitted_line$x3 = fitted_line$x^3
fitted_line$x4 = fitted_line$x^4
fitted_line$x5 = fitted_line$x^5
fitted_line$x6 = fitted_line$x^6
fitted_line$x7 = fitted_line$x^7

fitted_line$y1 <- predict(lm_1,newdata = fitted_line)
fitted_line$y7 <- predict(lm_7,newdata = fitted_line)

plt1 = ggplot(mapping = aes(x,y)) + ggtitle("Modelo lineal") + geom_line(aes(y = y1), colour =
"blue", data = fitted_line, size = 1.5, alpha = 0.5) + geom_point()

plt2 = ggplot(mapping = aes(x,y)) + ggtitle("Modelo m = 7") + geom_line(aes(y = y7), colour = "r
ed", data = fitted_line, size = 1.5, alpha = 0.5) + geom_point()

grid.arrange(plt0, plt1, plt2, ncol = 2)
```







Como podrán notar, el modelo lineal funciona bien y parece más estable. En cambio, es probable que el segundo modelo cambie drásticamente ante una fluctuación mínima de los datos, por lo que no sabemos si se comportará bien ante nuevos datos.

¿Cómo elegir “sabiamente” nuestro modelo?

Supongamos que tenemos una respuesta $\mathbf{Y}$ y potenciales predictores $X_1, X_2, \ldots, X_p$. Nuestro objetivo es que nuestro modelo sea lo más simple posible (i.e. que utilice la menor cantidad de predictores), sin que se ponga entredicho buen ajuste a los datos.

¿Cuál es el universo completo de modelos que se pueden ajustar? 2^p

¿Qué cosas sabemos hacer en este punto?

- Pruebas de hipótesis sobre los parámetros $H_0 : \beta_i = 0$
- Tablas ANOVA para evaluar un modelo
- Tablas ANOVA para comparar dos modelos
- Validación Cruzada

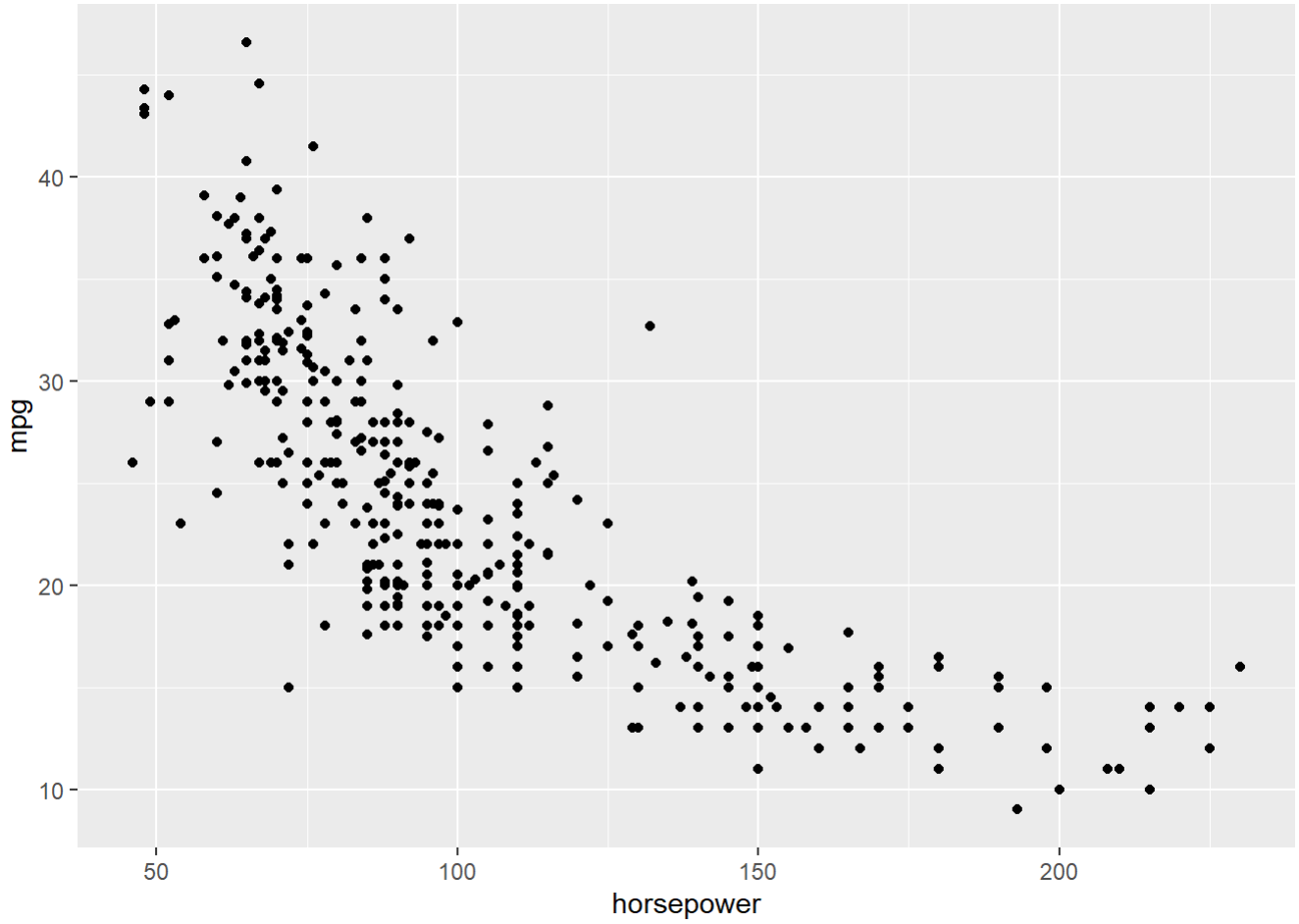
Las primeras tres herramientas en realidad sólo sirven para comparar *parejas de modelos*. Para hallar modelos convenientes dentro del universo de 2^p, hay que ser más hábiles.

Veamos primero un ejemplo bastante parecido al anterior, donde se puede usar fácilmente validación cruzada

Dataframe “Auto”

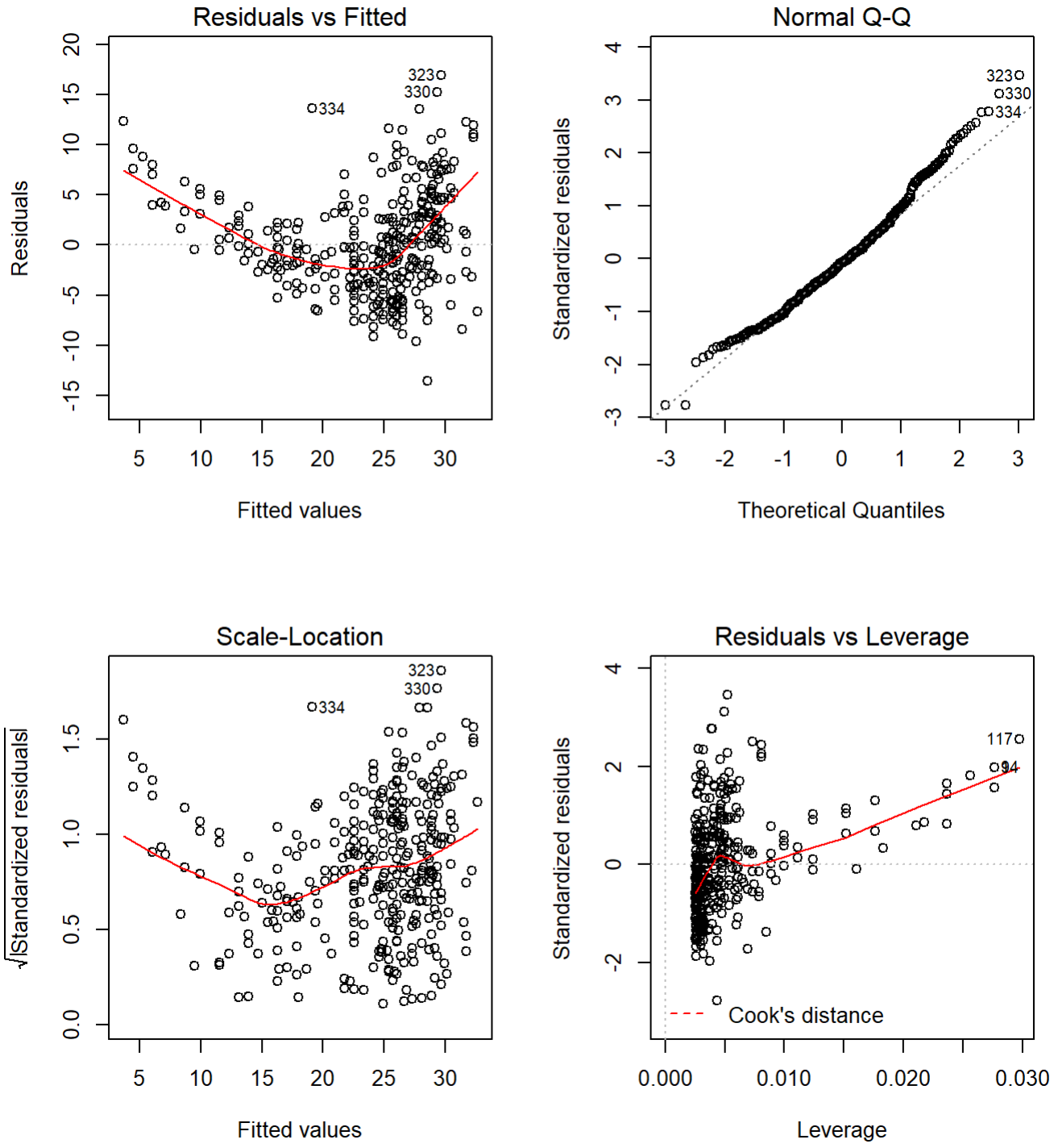
El conjunto de datos “Auto” en la librería ISLR contiene información de 392 vehículos, entre los que se incluye millas por galón de gasolina y caballos de fuerza.

```
data(Auto)

ggplot(data = Auto, aes(x = horsepower, y = mpg)) + geom_point()
```



Recordemos qué pasa cuando el modelo lineal no es suficiente (i.e. cuando hay que incluir términos no lineales)

```
linear_model <- lm(mpg ~ horsepower, data = Auto)
par(mfrow = c(2,2))
plot(linear_model)
```



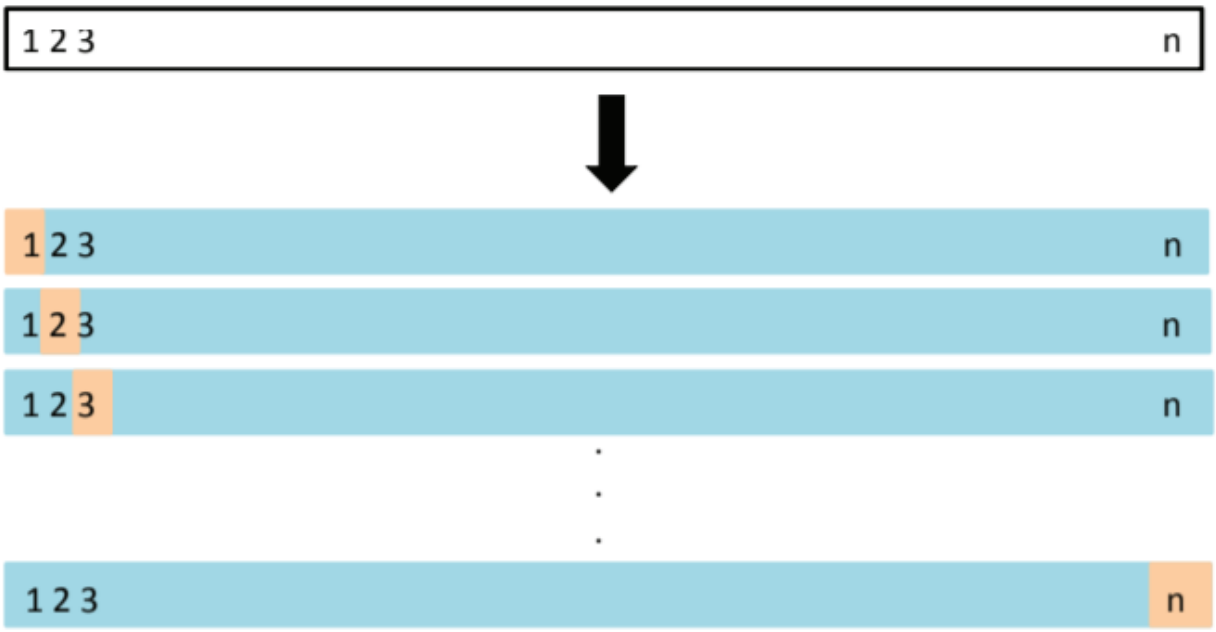
¿Qué hay que hacer?

Al igual que en el ejemplo anterior, es probable que haya que incorporar términos de orden mayor al lineal y hay que determinar un valor conveniente para m

$$\mathbb{E}[Y] = \beta_0 + \beta_1 x + \cdots + \beta_m x^m$$

LOOCV - Validación Cruzada Leave-one-out (dejar

uno fuera)



Leave-One-Out CV

Recordemos que esta estimación del error se obtiene como

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

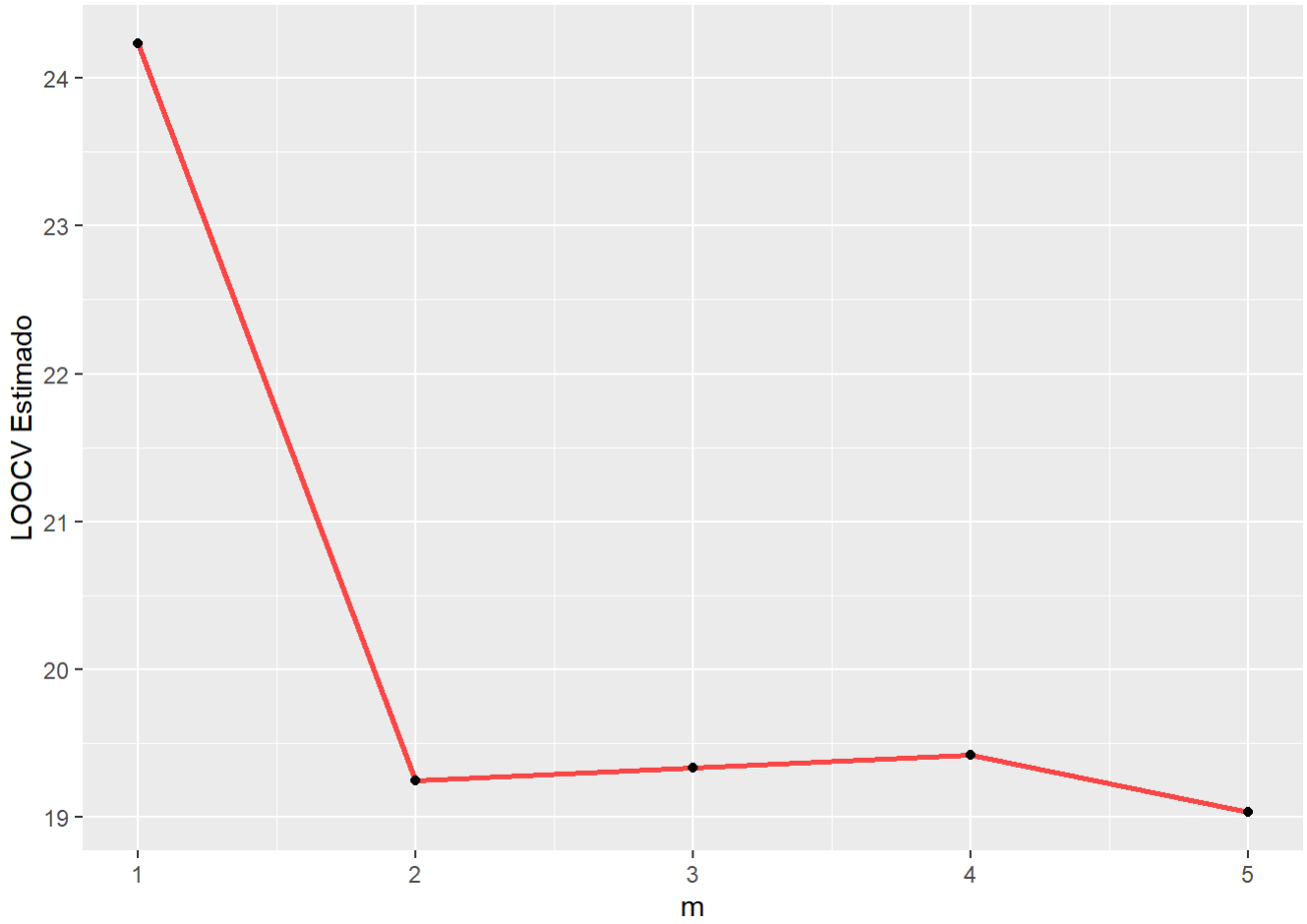
$\hat{y}_i$ es la estimación de y_i respecto a un modelo que sólo utilice las $n-1$ observaciones $\{(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \ldots, (x_n, y_n)\}$

La función cv.glm halla por default el LOOCV. Notamos que la función glm ajusta por default un modelo lineal cuando no se le indica una famlia de distribución.

```
cv.error = rep (0 ,5)
for(i in 1:5) {
  glm.fit = glm(mpg ~ poly(horsepower ,i) , data = Auto )
  cv.error[i]= cv.glm(Auto , glm.fit)$delta[1]
}
 cv.error
```

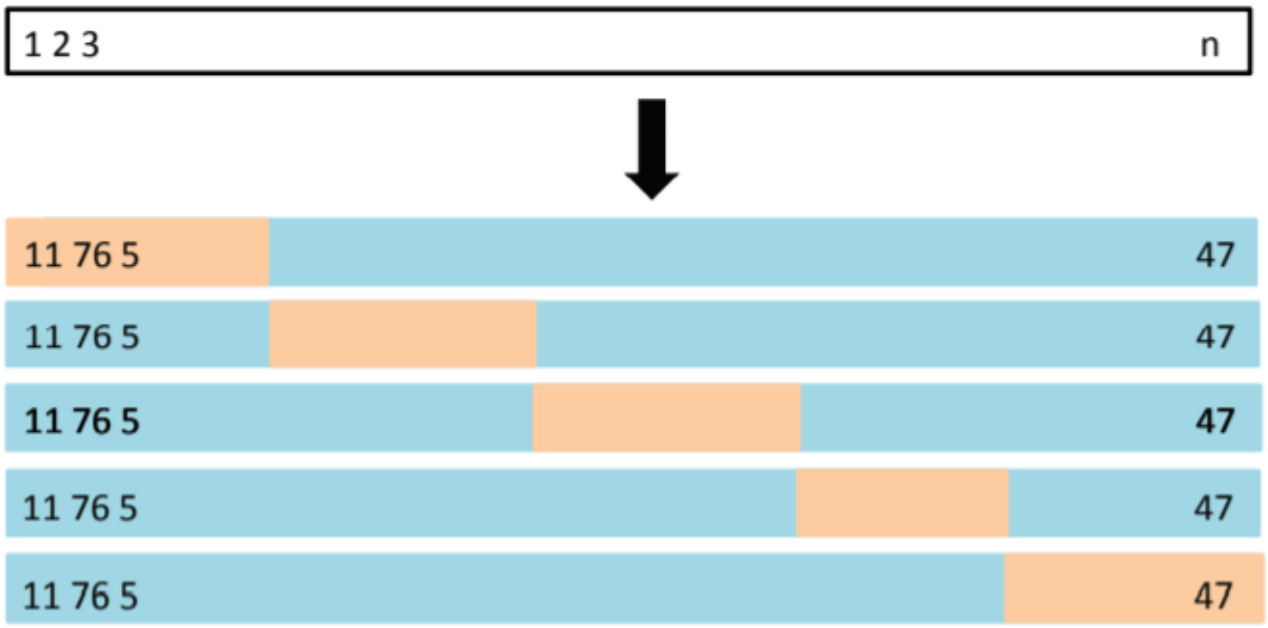
```
## [1] 24.23151 19.24821 19.33498 19.42443 19.03321
```

```
 ggplot(data = NULL, aes(x = 1:5,y = cv.error)) + geom_line(colour = "red", size = 1, alpha = 0.
7) + geom_point() + xlab("m") + ylab("LOOCV Estimado")
```



Hay una disminución considerable de usar un modelo lineal (m = 1) a uno cuadrático (m = 2), y posteriormente no existe un cambio significativo.

K-fold Cross Validation

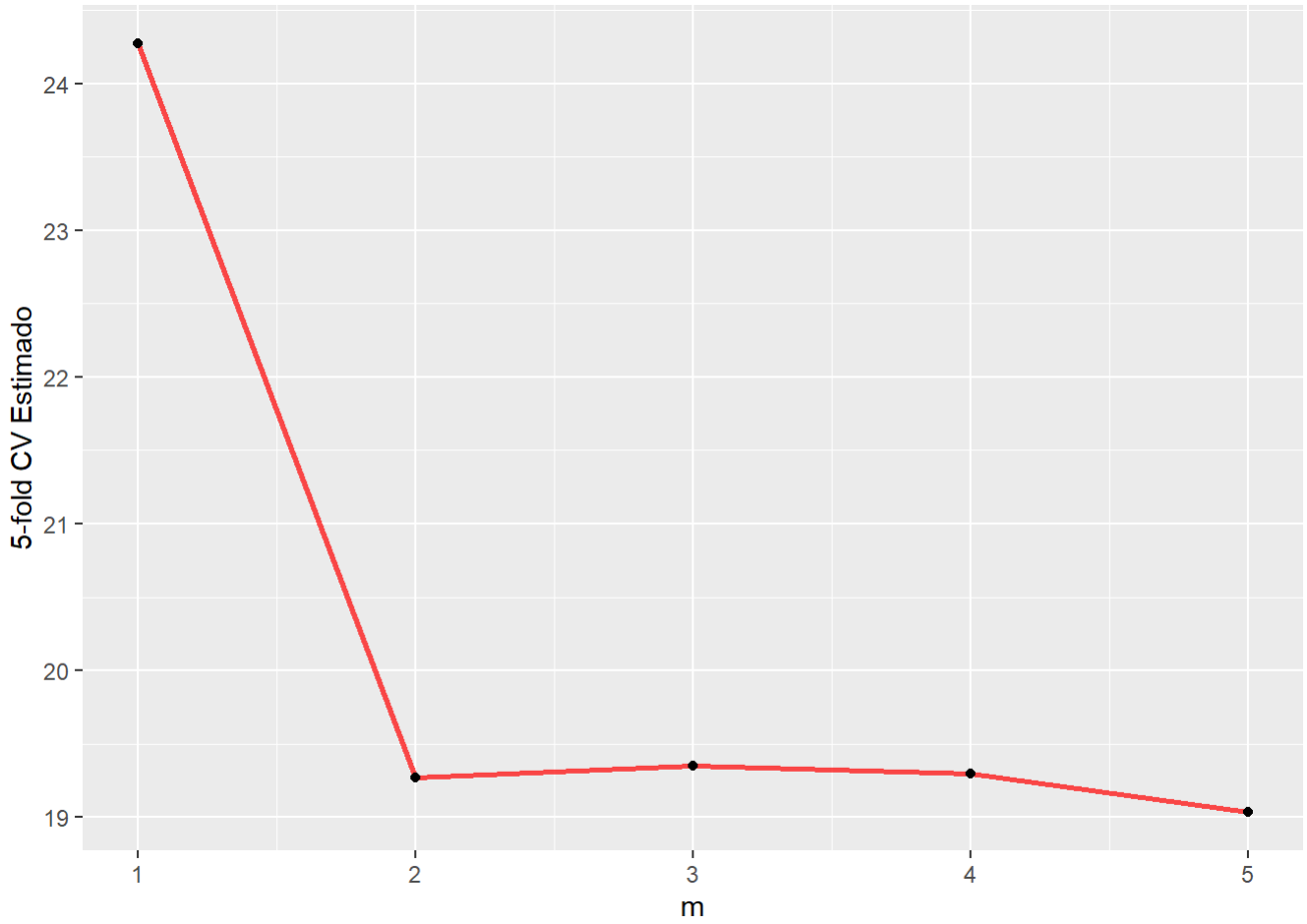


K-fold CV

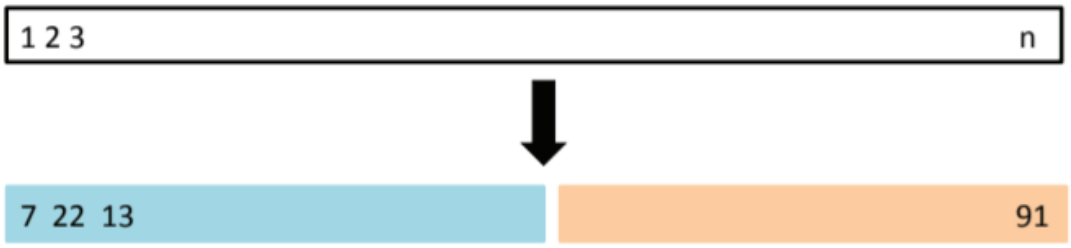
```
set.seed(17)
cv.error.5= rep (0 ,5)
for( i in 1:5) {
glm.fit = glm (mpg ~ poly( horsepower ,i) , data = Auto )
cv.error.5[ i ]= cv.glm( Auto , glm.fit ,K =10)$delta[1]
}
cv.error.5
```

```
## [1] 24.27207 19.26909 19.34805 19.29496 19.03198
```

```
ggplot(data = NULL, aes(x = 1:5,y = cv.error.5)) + geom_line(colour = "red", size = 1, alpha =
0.7) + geom_point() + xlab("m") + ylab("5-fold CV Estimado")
```



Conjunto de validación

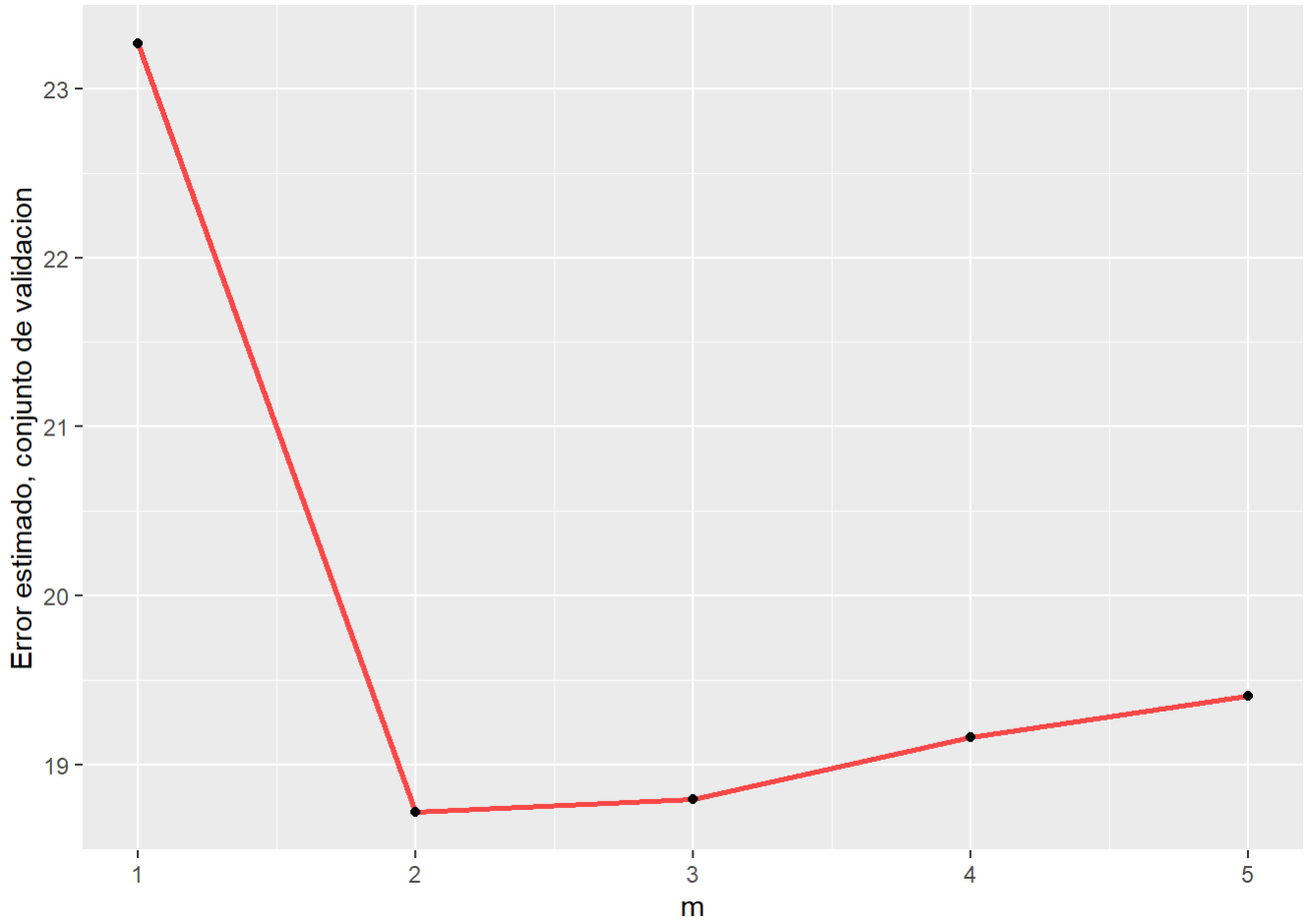


Conjunto de validación

```
set.seed (1)
train = sample (392 ,196)


validation_set_error = rep (0 ,5)

for(i in 1:5) {
  lm.fit = lm(mpg ~ poly(horsepower ,i) , data = Auto , subset = train)
  pred.vals <- predict(lm.fit, newdata = Auto[-train,])
  validation_set_error[i]= mean((Auto$mpg[-train] - pred.vals)^2)
}

validation_set_error
```

```
## [1] 23.26601 18.71646 18.79401 19.16017 19.40812
```

```
ggplot(data = NULL, aes(x = 1:5,y = validation_set_error)) + geom_line(colour = "red", size = 1,
alpha = 0.7) + geom_point() + xlab("m") + ylab("Error estimado, conjunto de validacion")
```



Por último, podemos calcular (con fines comparativos) el error medio in-sample.

```
in_sample_error = rep (0 ,5)

for(i in 1:5) {
  lm.fit = lm(mpg ~ poly(horsepower ,i) , data = Auto )
  in_sample_error[i]= mean(lm.fit$residuals^2)
}

in_sample_error
```

```
## [1] 23.94366 18.98477 18.94499 18.87633 18.42697
```

Los comparamos

```
data.frame("m" = 1:5,
           "validation" = validation_set_error,
           "K-fold" = cv.error.5,
           "LOOCV" = cv.error,
           "in_sample" = in_sample_error)
```

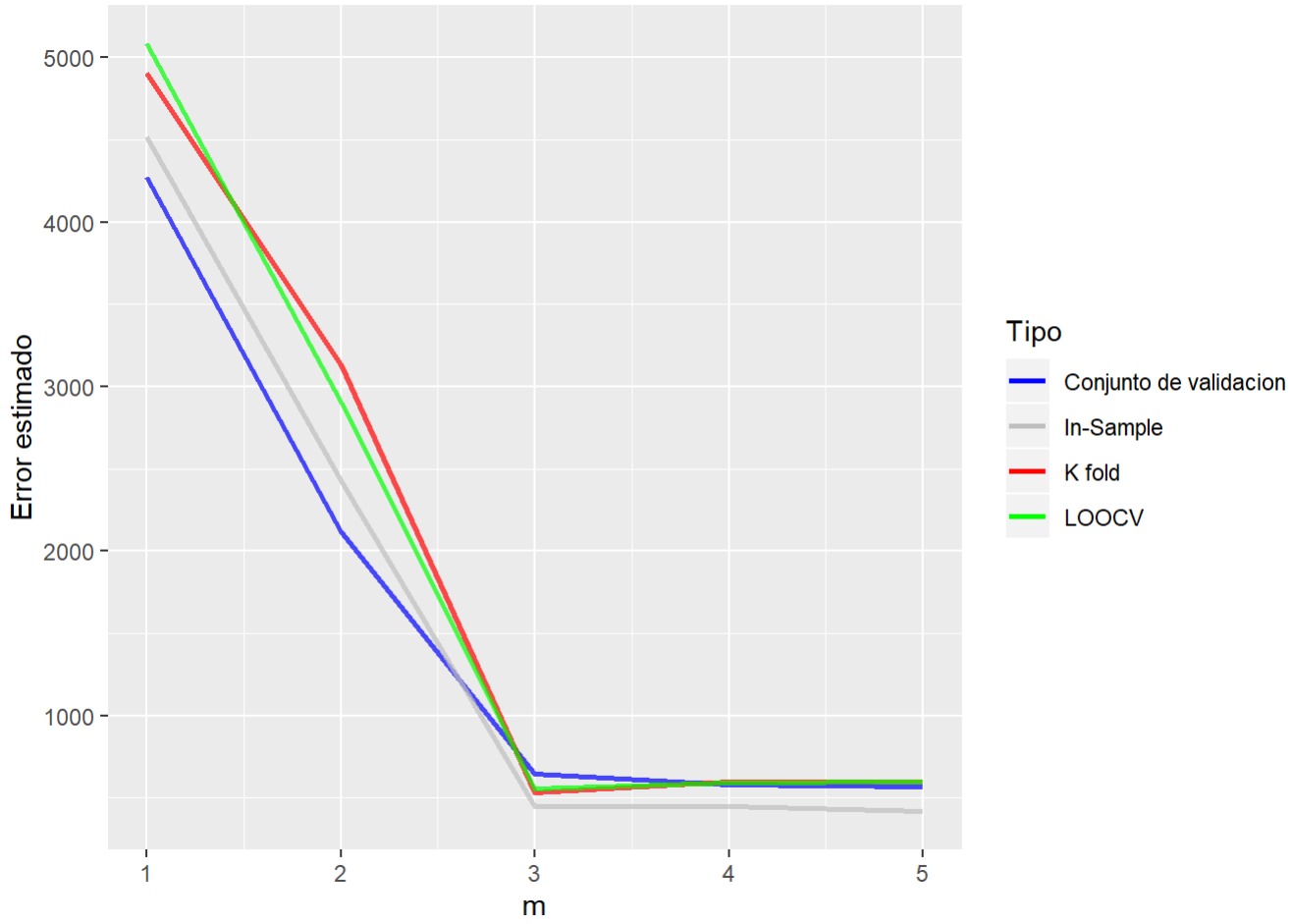
```
##   m validation   K.fold    LOOCV in_sample
## 1 1   23.26601 24.27207 24.23151  23.94366
## 2 2   18.71646 19.26909 19.24821  18.98477
## 3 3   18.79401 19.34805 19.33498  18.94499
## 4 4   19.16017 19.29496 19.42443  18.87633
## 5 5   19.40812 19.03198 19.03321  18.42697
```

Usualmente el set de validacion suele tener mayor error, sobre todo siendo tan grande 50%

La misma gráfica aplicada al primer ejemplo:

```
ej1 <- rbind(ej1,ej1_test)

cv.error = rep (0 ,5)
for(i in 1:5) {
  glm.fit = glm(y ~ poly(x ,i) , data = ej1 )
  cv.error[i]= cv.glm(ej1 , glm.fit)$delta[1]
}

set.seed(17)

cv.error.5= rep (0 ,5)
for( i in 1:5) {
glm.fit = glm (y ~ poly( x ,i) , data = ej1 )
cv.error.5[ i ]= cv.glm( ej1 , glm.fit ,K =10)$delta[1]
}

set.seed(4096)

train <- sample(n,round(n/3))

validation_set_error = rep (0 ,5)

for(i in 1:5) {
  lm.fit = lm(y ~ poly(x ,i) , data = ej1 , subset = train)

  pred.vals <- predict(lm.fit, newdata = ej1[-train,])
  validation_set_error[i]= mean((ej1$mpg[-train] - pred.vals)^2)
  validation_set_error[i]= mean(lm.fit$residuals^2)
}


in_sample_error = rep (0 ,5)

for(i in 1:5) {
  lm.fit = lm(y ~ poly(x ,i) , data = ej1 )
  in_sample_error[i]= mean(lm.fit$residuals^2)
}

errores <- data.frame("m" = 1:5,
           "validation" = validation_set_error,
           "K_fold" = cv.error.5,
           "LOOCV" = cv.error,
           "in_sample" = in_sample_error)


ggplot(data = errores, aes(x = m)) +
  geom_line(aes(y = validation,color = "Conjunto de validacion"), size = 1, alpha = 0.7) +
  geom_line(aes(y = K_fold,color = "K fold"), size = 1, alpha = 0.7) +
  geom_line(aes(y = LOOCV,color = "LOOCV"), size = 1, alpha = 0.7) +
  geom_line(aes(y = in_sample,color = "In-Sample"), size = 1, alpha = 0.7)+
  labs(y = "Error estimado", x = "m" , color = "Tipo") +
  scale_color_manual(values = c("Conjunto de validacion" = "blue",
```

```
                    "K fold" = "red",
                    "LOOCV" = "green",
                    "In-Sample" = "gray"))
```



Casos más complicados

En el caso anterior, era más o menos sencillo establecer una rutina para hallar un modelo adecuado, puesto que sólo teníamos un predictor que podía aparecer elevado a potencias mayores que 1 en el modelo.

Sin embargo, cuando tenemos p predictores distintos, habíamos mencionado que existen 2^p modelos distintos que probar (contando el naif).

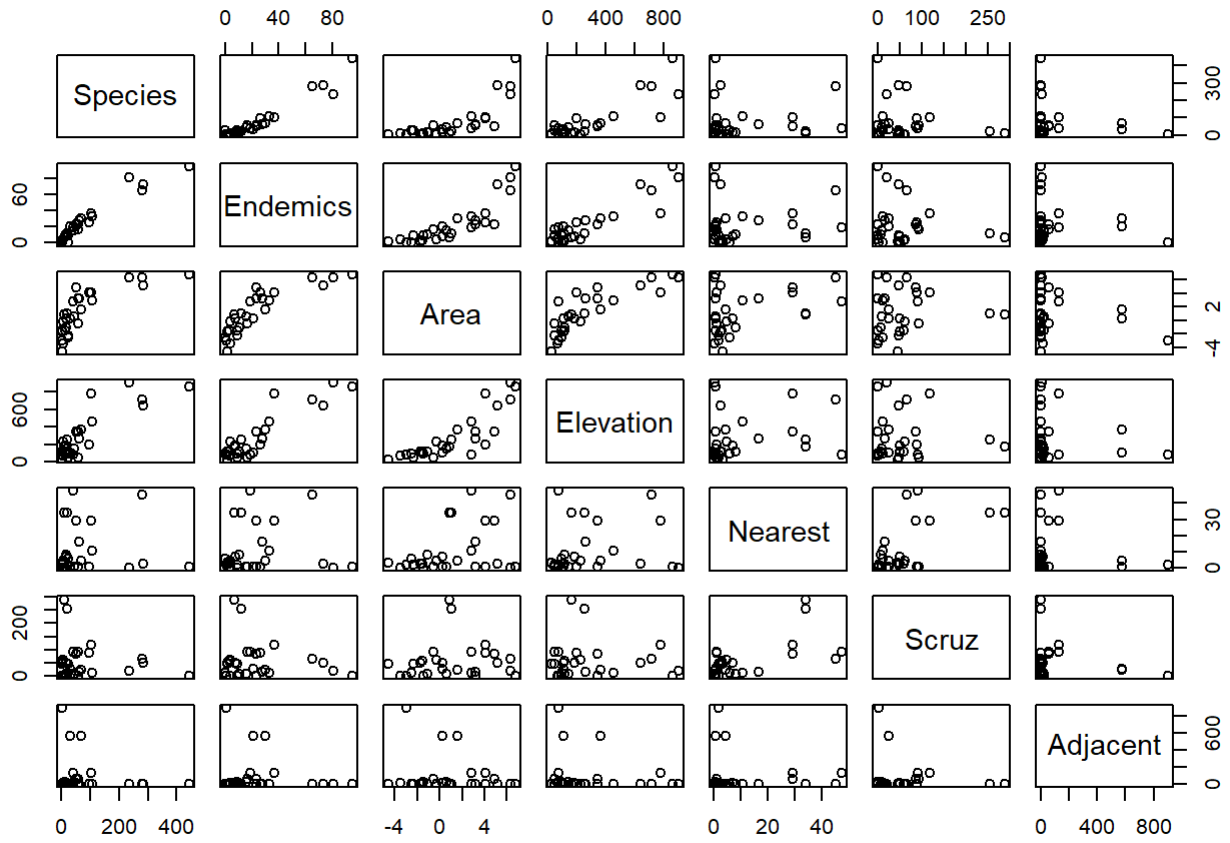
Consideraciones importantes

Hay algunos conceptos respecto a los que puede haber malentendidos, que vale la pena analizar

Galápagos

Recordemos el conjunto que habíamos analizado previamente, sobre especies de tortugas en las islas Galápagos. El propósito en este conjunto de datos era tratar de predecir “Species” a partir del resto de las variables, preferentemente variables geográficas

```
gala4 <- read.csv("transformed_gala.txt")
pairs(gala4)
```



Primera aclaración: Como ya se había mencionado, la tabla de anova no nos dice automáticamente qué incluir y qué no incluir en el modelo. Para el análisis anova depende mucho el orden en que ingresamos las covariables.

```
lm_full <- lm(Species ~ . , data = gala4)
anova(lm_full)
```

```
## Analysis of Variance Table
##
## Response: Species
##           Df Sum Sq Mean Sq  F value    Pr(>F)
## Endemics   1 288892  288892 361.6168 1.026e-14 ***
## Area       1   2544    2544   3.1842   0.08881 .
## Elevation  1    229     229   0.2868   0.59793
## Nearest    1    406     406   0.5076   0.48402
## Scruz      1     13      13   0.0157   0.90134
## Adjacent   1   1045    1045   1.3076   0.26570
## Residuals 21  16777     799
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lm_full_1 <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent + Endemics , data = gala
4)
anova(lm_full_1)
```

```
## Analysis of Variance Table
##
## Response: Species
##           Df Sum Sq Mean Sq  F value    Pr(>F)
## Area       1 184695  184695 231.1889 8.280e-13 ***
## Elevation  1  45023   45023  56.3567 2.248e-07 ***
## Nearest    1   8081    8081  10.1154  0.004503 **
## Scruz      1   1389    1389   1.7385  0.201526
## Adjacent   1    883     883   1.1058  0.304951
## Endemics   1  53057   53057  66.4134 6.113e-08 ***
## Residuals 21  16777     799
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

En cada renglón i, se comparan el modelo con las covariables $X_1, X_2, \ldots, X_{i-1}$ vs. $X_1, X_2, \ldots, X_i$. Por lo tanto, si por ejemplo X_1 aporta mucha información al modelo (i.e. explica mucha de la varianza), el resto de las covariables no aportarán mucha más información y por lo tanto la tabla ANOVA las catalogará como poco significativas.

La columna Sum Sq menciona cuánta varianza adicional explica la nueva variable: lo podemos corroborar en las siguientes comparaciones (la columna Sum of Sq coincide con las de arriba):

```
ej_lm1 <- lm(Species ~ Area + Elevation + Nearest  , data = gala4)
ej_lm2 <- lm(Species ~ Area + Elevation + Nearest + Scruz  , data = gala4)
ej_lm3 <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent  , data = gala4)
anova(ej_lm3,lm_full_1)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ Area + Elevation + Nearest + Scruz + Adjacent
## Model 2: Species ~ Area + Elevation + Nearest + Scruz + Adjacent + Endemics
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1     22 69834
## 2     21 16777  1     53057 66.413 6.113e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(ej_lm2,ej_lm3)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ Area + Elevation + Nearest + Scruz
## Model 2: Species ~ Area + Elevation + Nearest + Scruz + Adjacent
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1     23 70717
## 2     22 69834  1     883.4 0.2783 0.6031
```

```
anova(ej_lm1,ej_lm2)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ Area + Elevation + Nearest
## Model 2: Species ~ Area + Elevation + Nearest + Scruz
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1     24 72106
## 2     23 70717  1    1388.9 0.4517 0.5082
```

Segunda aclaración: Las pruebas de hipótesis en sobre los coeficientes $H_0 : \beta_i = 0$ tampoco nos dicen automáticamente qué estimadores omitir.

```
summary(lm_full)
```

```
##
## Call:
## lm(formula = Species ~ ., data = gala4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -73.112 -13.217  -1.016   9.355  64.014
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -27.605948  13.063527  -2.113   0.0467 *
## Endemics      5.098972   0.625684   8.149 6.11e-08 ***
## Area         -7.146263   4.014268  -1.780   0.0895 .
## Elevation    -0.032813   0.052240  -0.628   0.5367
## Nearest       0.327837   0.546231   0.600   0.5548
## Scruz        -0.005821   0.104674  -0.056   0.9562
## Adjacent     -0.029412   0.025721  -1.144   0.2657
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.26 on 21 degrees of freedom
## Multiple R-squared:  0.9459, Adjusted R-squared:  0.9304
## F-statistic: 61.15 on 6 and 21 DF,  p-value: 3.266e-12
```

El valor P para cierto estimador depende en gran parte de qué otros están presentes.

```
lm_elevation <- lm(Species ~ . , data = gala4[,-4])
anova(lm_elevation,lm_full)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ Endemics + Area + Nearest + Scruz + Adjacent
## Model 2: Species ~ Endemics + Area + Elevation + Nearest + Scruz + Adjacent
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1     22 17092
## 2     21 16777  1    315.18 0.3945 0.5367
```

```
lm_Scruz <- lm(Species ~ . , data = gala4[,-6])
anova(lm_Scruz,lm_full)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ Endemics + Area + Elevation + Nearest + Adjacent
## Model 2: Species ~ Endemics + Area + Elevation + Nearest + Scruz + Adjacent
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1     22 16779
## 2     21 16777  1    2.4703 0.0031 0.9562
```

El valor p del resumen de coeficientes equivale a comparar el modelo completo a un modelo donde sólo quitamos un predictor.

El problema con este tipo de pruebas, es que depende mucho de qué otras variables queden en el modelo. Por ejemplo, si dos variables (digamos X_i y X_j) están correlacionadas, aportarían más o menos la misma información y, por lo tanto, ambas tendrían un valor p alto.

¿Por qué? Porque X_i no aporta mucha información extra si en el modelo ya está X_j y vice versa.

Selección de subconjuntos de predictores

Consideremos el conjunto Hitters de la librería ISLR, que contiene datos de las ligas mayores de baseball de las temporadas de 1986 y 1987. El propósito de este ejemplo es tratar de predecir el salario de un jugador de baseball basado en varias estadísticas asociadas con el performance del año anterior.

```
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"
##  [6] "Walks"     "Years"     "CAtBat"    "CHits"     "CHmRun"
## [11] "CRuns"     "CRBI"      "CWalks"    "League"    "Division"
## [16] "PutOuts"   "Assists"   "Errors"    "Salary"    "NewLeague"
```

```
dim(Hitters)
```

```
## [1] 322  20
```

Podemos eliminar las observaciones que tienen NA en el campo de salario

```
sum(is.na(Hitters$Salary))
```

```
## [1] 59
```

```
Hitters = na.omit(Hitters)
```