MAE: ENTREGA II

Jon Zorrilla Gamboa

1. Ejercicio 1

Leemos el dataset:

```
natalidad <- read.table("https://verso.mat.uam.es/~joser.berrendero/datos/natalidad.txt", hea
mutate(log_pnb = log(pnb))
```

Este dataset tiene la siguiente pinta:

```
   nat mort mortinf esph espm  pnb  log_pnb
1 24.7  5.7    30.8 69.6 75.5  600 6.396930
2 12.5 11.9    14.4 68.3 74.7 2250 7.718685
3 13.4 11.7    11.3 71.8 77.7 2980 7.999679
4 11.6 13.4    14.8 65.4 73.8 2780 7.930206
5 14.3 10.2    16.0 67.2 75.7 1690 7.432484
6 13.6 10.7    26.9 66.5 72.4 1640 7.402452
```

En primer lugar, se ajusta un modelo de regresión lineal múltiple, en función de la natalidad, la tasa de mortalidad infantil y el logaritmo del producto nacional bruto del país:

```
reg <- lm(esph ~ nat  + mortinf + log_pnb, data = natalidad)
summary(reg)
```

```
Call:
lm(formula = esph ~ nat + mortinf + log_pnb, data = natalidad)

Residuals:
    Min      1Q  Median      3Q     Max
-8.4893 -2.1660  0.1581  2.0663  7.9084
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 65.95088    3.25642  20.253  < 2e-16 ***
nat         -0.14621    0.04762  -3.071  0.00285 **
mortinf     -0.13312    0.01537  -8.663  2.2e-13 ***
log_pnb      0.94478    0.32989   2.864  0.00524 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.128 on 87 degrees of freedom
Multiple R-squared:    0.9, Adjusted R-squared:  0.8966
F-statistic: 261.1 on 3 and 87 DF,  p-value: < 2.2e-16
```

1.1.

¿De cuántos países consta la muestra utilizada?

La muestra utilizada consta de 91 países.

1.2.

¿Cuánto vale la suma de cuadrados que se utiliza para medir la variabilidad explicada por las tres variables regresoras?

La variabilidad explicada se mide mediante la siguiente fórmula:

$$SCE = \sum_{i=1}^{n} (\hat{Y}_i - \bar{Y})^2 \tag{1}$$

Entonces, aplicando esta fórmula al modelo lineal, obtenemos $SCE = 7665{,}254$

1.3.

¿Cuánto vale la varianza muestral de la variable respuesta $\sum_{i=1}^{n} \frac{(Y_i - \bar{Y})^2}{(n-1)}$?

Aplicando la fórmula, obtenemos 94,62798.

1.4.

Contrasta a nivel $\alpha = 0{,}05$ la hipótesis nula $H_0 : \beta_1 = 0$

Con los datos obtenidos en:

```
reg <- lm(esph ~ nat  + mortinf + log_pnb, data = natalidad)
summary(reg0)
```

```
Call:
lm(formula = esph ~ mortinf + log_pnb, data = natalidad)

Residuals:
    Min      1Q  Median      3Q     Max
-8.4484 -1.9316  0.0083  1.7038  8.4861

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 61.84674    3.10843  19.896  < 2e-16 ***
mortinf     -0.16449    0.01201 -13.691  < 2e-16 ***
log_pnb      1.14861    0.33827   3.396  0.00103 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.275 on 88 degrees of freedom
Multiple R-squared:  0.8892,        Adjusted R-squared:  0.8867
F-statistic: 353.1 on 2 and 88 DF,  p-value: < 2.2e-16
```

Obtenemos un p valor $p < 2{,}2 \cdot 10^{-16}$

1.5.

Contrasta a nivel $\alpha = 0{,}05$ la hipótesis nula $H_0 : \beta_1 = \beta_2 = \beta_3 = 0$

```
reg1 <- lm(esph ~ 1, data = natalidad)
summary(reg1)
```

```
Call:
```

```
lm(formula = esph ~ 1, data = natalidad)

Residuals:
    Min      1Q  Median      3Q     Max
-23.281  -5.981   2.019   7.119  14.519

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    61.38       1.02   60.19   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.728 on 90 degrees of freedom
```

En este caso, como tenemos un modelo lineal, no obtenemos ningún dato correspondiente al p valor. Con lo cual podemos suponer que este modelo no es óptimo.

1.6.

Estima la correlación entre $\hat{\beta}_1$ y $\hat{\beta}_2$. (Indicación: usa el comando *vcov*)

Escribimos lo siguiente:

```
vcov(reg)

            (Intercept)           nat       mortinf      log_pnb
(Intercept) 10.60424770 -0.0636441427 -0.0154732224 -1.033908908
nat         -0.06364414  0.0022673659 -0.0004865478  0.003160908
mortinf     -0.01547322 -0.0004865478  0.0002361416  0.002230265
log_pnb     -1.03390891  0.0031609079  0.0022302652  0.108830676
```

Como podemos ver, obtenemos una correlación entre β_1 y β_2 de $-0{,}0004865478$. Lo que implica una correlación muy baja.

1.7.

Calcula intervalos de confianza al nivel 90 % para todos los β_i del modelo. (Indicación: usa el comando *confint*)

Hacemos uso del siguiente código:

```
confint(object = reg, level=0.9)
```

```
                   5 %        95 %
(Intercept) 60.5369009 71.36485887
nat         -0.2253786 -0.06704702
mortinf     -0.1586652 -0.10756848
log_pnb      0.3963072  1.49324572
```

1.8.

Predice el valor de la esperanza de vida de los hombres en un país para el que el índice de natalidad es 29, la mortalidad infantil vale 50 y el logaritmo de su pnb vale 7. Calcula un intervalo de confianza del 95 % para el valor esperado de dicha variable.

```
nuevo.dato <- data.frame(29, 50, 7)
names(nuevo.dato) <- names(natalidad)[c(1, 3, 7)]
nuevo.dato
predict(reg, nuevo.dato, interval='prediction')
```

```
      fit      lwr      upr
1 61.6683 55.40147 67.93514
```

Como podemos observar, se predice un valor de 61,6683, con un intervalo de confianza del 95 % entre los siguientes valores: (55.40147, 67.93514).

2. Ejercicio 2

En primer lugar, generaremos una variable regresora aleatoria X, un vector aleatorio ϵ y la variable respuesta $Y = X + X^2 + X^3 + \epsilon$ de la siguiente manera:

```
library(GGally)
library(leaps)
```

```
library(glmnet)


set.seed(100)
n <- 100


x <- rnorm(n)
eps <- rnorm(n, sd=1)
y = x + x**2 + x**3 + eps
```

2.1.

En primer lugar, definimos la matriz de datos con todas las variables:

```
datos <- data.frame(y=y, x1=x, x2=x**2, x3=x**3, x4=x**4, x5=x**5, x6=x**6, x7=x**7,
x8=x**8, x9=x**9, x10=x**10)
```

Ahora, estudiaremos el modelo:

```
modelo_todos <- leaps::regsubsets(y ~ ., data=datos)
resumen_todos <- summary(modelo_todos)
resumen_todos$outmat
```

```
         x1  x2  x3  x4  x5  x6  x7  x8  x9  x10
1  ( 1 ) " " " " "*" " " " " " " " " " " " " "
2  ( 1 ) " " "*" "*" " " " " " " " " " " " " "
3  ( 1 ) "*" "*" "*" " " " " " " " " " " " " "
4  ( 1 ) "*" "*" "*" "*" " " " " " " " " " " "
5  ( 1 ) "*" "*" "*" "*" "*" " " " " " " " " "
6  ( 1 ) "*" "*" "*" " " "*" " " " " "*" " " "*"
7  ( 1 ) "*" "*" "*" "*" " " "*" " " "*" " " "*"
8  ( 1 ) "*" "*" "*" "*" "*" "*" " " "*" " " "*"
```

De esta manera, hemos creado el conjunto de datos mediante el modelo exhaustivo, donde, los asteriscos indican las variables más importantes a la hora de ajustar el modelo. Ahora, crearemos una función que nos de el mejor resultado en función de cada criterio C_p, BIC y R_a^2, y hallaremos que variables son necesarias para obtener el mejor resultado.

```
optimal <- data.frame(
  CP = which.min(resumen_todos$cp),
  BIC = which.min(resumen_todos$bic),
  Adj.R2 = which.max(resumen_todos$adjr2)
)

cat("Mejor modelo según criterio Cp:\n")
resumen_todos$outmat[optimal[[1]],]
cat("Mejor modelo según criterio BIC:\n")
resumen_todos$outmat[optimal[[2]],]
cat("Mejor modelo según criterio R^2:\n")
resumen_todos$outmat[optimal[[3]],]
```

Y, las variables indicadas con asteriscos son las que crean el mejor modelo para los distintos criterios:

```
> cat("Mejor modelo según criterio Cp:\n")
Mejor modelo según criterio Cp:
> resumen_todos$outmat[optimal[[1]],]
 x1  x2  x3  x4  x5  x6  x7  x8  x9 x10
"*" "*" "*" " " " " " " " " " " " " " "
> cat("Mejor modelo según criterio BIC:\n")
Mejor modelo según criterio BIC:
> resumen_todos$outmat[optimal[[2]],]
 x1  x2  x3  x4  x5  x6  x7  x8  x9 x10
"*" "*" "*" " " " " " " " " " " " " " "
> cat("Mejor modelo según criterio R^2:\n")
Mejor modelo según criterio R^2:
> resumen_todos$outmat[optimal[[3]],]
 x1  x2  x3  x4  x5  x6  x7  x8  x9 x10
"*" "*" "*" "*" "*" "*" " " "*" " " "*"
```

Como se puede observar, pese a que hallamos definido $Y = X + X^2 + X^3 + \epsilon$, solo para los criterios Cp y BIC se hace uso de esas variables, no ocurre lo mismo para el criterio R^2, el cual hace uso de muchas más variables. Cuando sucede esto, regularizar el modelo es buena idea.

2.2.

Ahora, realizaremos lo anterior pero con el método iterativo hacia adelante:

```
modelo_forward <- regsubsets(y ~ ., data=datos, method='forward', nvmax=10)
resumen_forward <- summary(modelo_forward)
resumen_forward$outmat

optimal_forward <- data.frame(
  CP = which.min(resumen_forward$cp),
  BIC = which.min(resumen_forward$bic),
  Adj.R2 = which.max(resumen_forward$adjr2)
)

cat("Mejor modelo según criterio Cp:\n")
resumen_todos$outmat[mej_forward[[1]],]
cat("Mejor modelo según criterio BIC:\n")
resumen_todos$outmat[mej_forward[[2]],]
cat("Mejor modelo según criterio R^2 ajustado:\n")
resumen_todos$outmat[mej_forward[[3]],]
```

Y, obtenemos:

```
> cat("Mejor modelo según R^2 ajustado:\n")
Mejor modelo según R^2 ajustado:
> resumen_todos$outmat[mej_forward[[1]],]
 x1  x2  x3  x4  x5  x6  x7  x8  x9 x10
"*" "*" "*" "*" " " " " " " " " " " " "
> cat("Mejor modelo según Cp:\n")
Mejor modelo según Cp:
> resumen_todos$outmat[mej_forward[[2]],]
 x1  x2  x3  x4  x5  x6  x7  x8  x9 x10
"*" "*" "*" " " " " " " " " " " " " " "
> cat("Mejor modelo según BIC:\n")
Mejor modelo según BIC:
> resumen_todos$outmat[mej_forward[[3]],]
```

```
 x1  x2  x3  x4  x5  x6  x7  x8  x9 x10
"*" "*" "*" " " " " " " " " " " " " " "
```

Para el método iterativo hacia adelante sí que obtenemos unos buenos resultados, pues se hace uso únicamente de las variables que describen el modelo.

2.3.

Ahora, aplicaremos la regularización Lasso al modelo, haremos uso de validación cruzada para seleccionar el parámetro de regularización y compararemos el ajuste con lo obtenido previamente.

```
x_l <- as.matrix(datos[,-1])
y_l <- datos[,1]
modelo_lasso <- cv.glmnet(x_l, y_l, alpha = 1)
lambda_lasso <- modelo_lasso$lambda.1se
modelo_final_lasso <- glmnet(x_l, y_l, alpha = 1, lambda = lambda_lasso)
cat("Lambda elegido:", lambda_lasso)
coef(modelo_final_lasso)
```

Donde, se ha obtenido un $\lambda = 0{,}2230922$ por validación cruzada y los siguiente coeficientes:

```
11 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept) 0.1636166178
x1          0.7050171039
x2          0.7028241595
x3          0.9843437309
x4          0.0497512674
x5          0.0001965399
x6          .
x7          .
x8          .
x9          .
x10         .
```

Este caso también obtenemos un resultado bueno, pues hace uso principalmente de x, x^2 y x^3, con diferentes pesos, mientras que el resto de variables regresoras tienen un peso muy pequeño en el modelo, lo cual nos da a entender que prácticamente no aparecen.

2.4.

Ahora, volveremos a aplicar la regularización Lasso al siguiente modelo: $Y = X^7 + \epsilon$.

```
y7 = x**7 + eps
modelo_lasso_7 = cv.glmnet(x_l, y7, alpha = 1)
lambda_lasso_7 <- modelo_lasso_7$lambda.1se
modelo_final_lasso_7 <- glmnet(x_l, y7, alpha = 1, lambda = lambda_lasso_7)
cat("Lambda elegido:", lambda_lasso_7)
coef(modelo_final_lasso_7)
```

En este caso, se hace uso de $\lambda = 4{,}157775$ y se obtiene lo siguiente:

```
11 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept) 0.3940213278
x1          .
x2          .
x3          .
x4          .
x5          .
x6          .
x7          0.9557000590
x8          .
x9          0.0009552257
x10         .
```

De nuevo, el resultado obtenido mediante lasso es óptimo porque hace uso del elemento $x7$, correspondiente a x^7, que es la componente principal de la función y. También se puede observar que hace uso de la variable $x10$, pero con un peso muy pequeño, lo cual es prácticamente insignificante.

Con esto, podemos concluir que es necesaria algún tipo de regularización para obtener los resultados deseados. En caso contrario, si el modelo no tiene restricciones, sobreajustará demasiado.

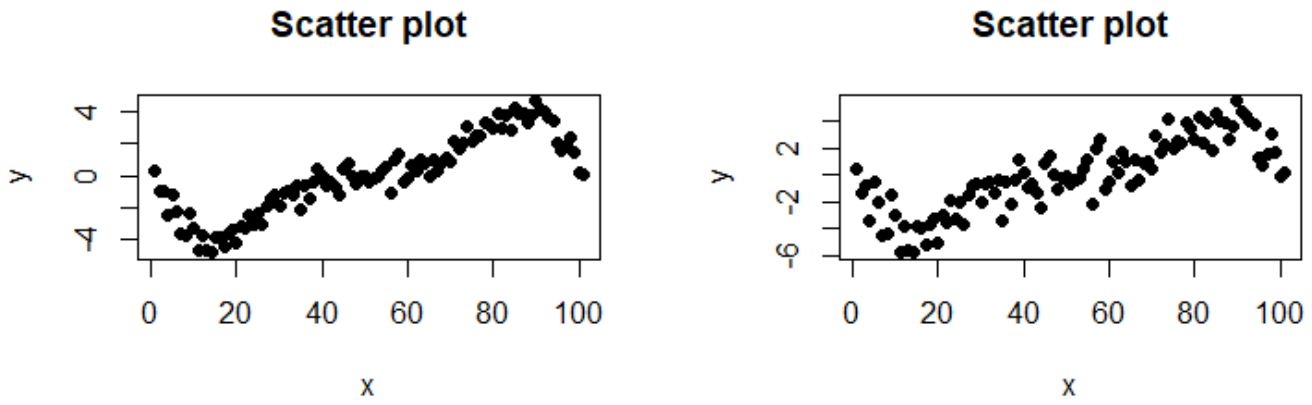
3. Ejercicio 3

Generamos dos conjuntos de datos de la siguiente manera:

```
library(ggplot2)


x <- seq(-pi,pi, (2*pi/100))
eps <- rnorm(101)
sigma1 <- 0.5
sigma2 <- 1.0



y <- function(x, sigma){
  res <- x**2*sin(x) + sigma*eps
  return(res)
  }
```

3.1.

Representamos los resultados obtenidos a continuación:

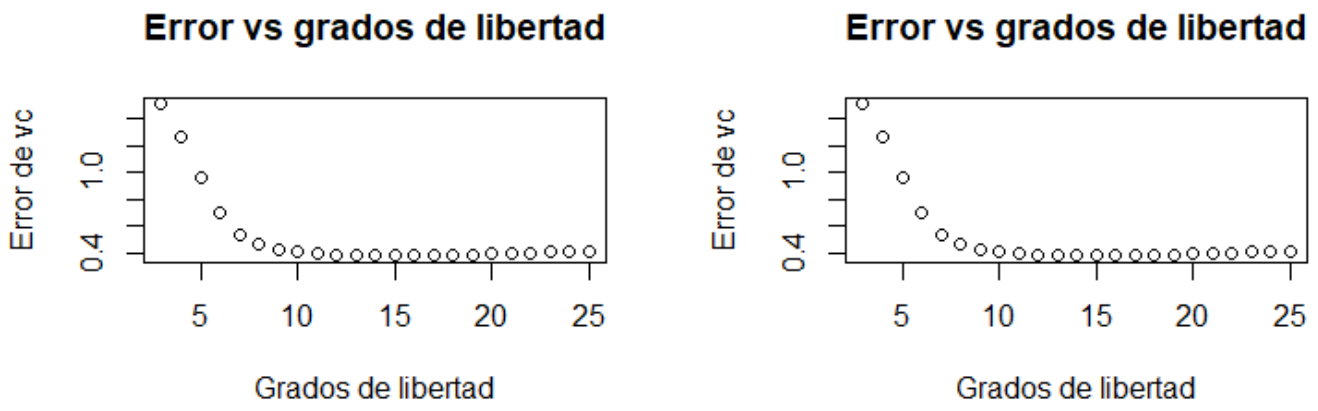


(a) $Y(x, \sigma = 0{,}5)$ (b) $Y(x, \sigma = 1{,}0)$

Figura *1*

3.2.

Ahora, ajustaremos una regresión de mínimos cuadrados penalizada prefijando un conjunto de grados de libertad efectivos $3, ..., 25$. Para ello, ejecutaremos el siguiente código:



(a) Error en función de los grados de libertad para $\sigma = 0{,}5$ (b) [Error en función de los grados de libertad para $\sigma = 1$

Figura *2*

```
gl <- c(3:25)
error <- c()
sum <- 0
for (i in (3:25)){
  splines <- smooth.spline(x, y(x, sigma), df = i)
  error <- append(error, splines$cv.crit)
}
plot(gl, error, main="Error vs grados de libertad", xlab="Grados de libertad", ylab ="Error
```

Como se puede observar en *2*, el error de validación cruzada es muy alto para muy pocos grados de libertad y empieza a reducirse a medida aumentamos los grados de libertad. Esto ocurre hasta un número determinado de grados de libertad, donde, como podemos observar, el error de validación cruzada empieza a aumentar de nuevo. Esto puede darse debido al overfitting, pues con tantos grados de libertad sobreajustamos el modelo.