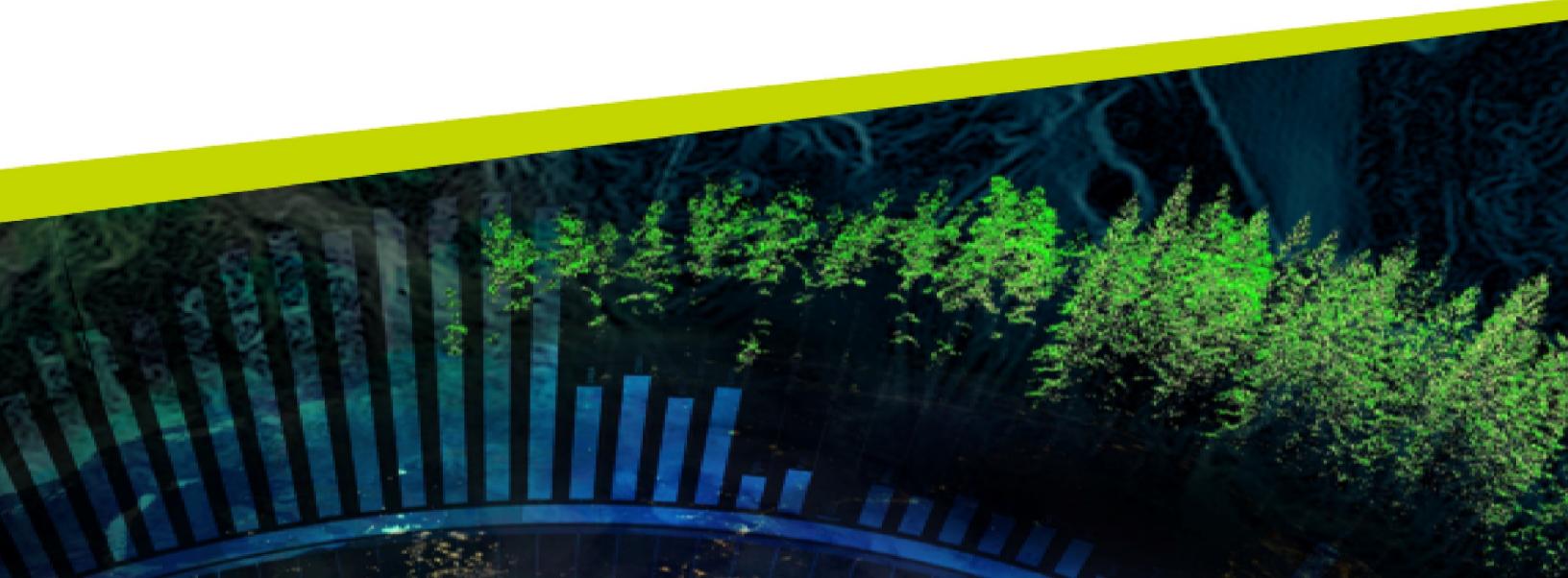




INFERENCIA Y MODELOS ESTADÍSTICOS

Jacqueline Köhler C. y José Luis Jara V.



CAPÍTULO 14. REGRESIÓN LINEAL MÚLTIPLE

En el capítulo anterior conocimos los principios detrás de la regresión lineal, considerando para ello una única variable predictora y una variable de respuesta. Sin embargo, en la vida real es más frecuente que un fenómeno pueda ser explicado por varias variables. En consecuencia, en este capítulo presentaremos un nuevo modelo lineal más complejo: la regresión lineal múltiple (RLM), correspondiente al caso de una única respuesta con múltiples predictores. Para ello tomaremos como base los textos de Field et al. (2012, pp. 245-311), Diez et al. (2017, pp. 372-385) y Fox y Weisberg (2018).

14.1 MODELO DE RLM

Una regresión lineal con múltiples variables tiene la forma que se presenta en la ecuación 14.1, donde:

- Cada x_i es un predictor.
- Cada β_i corresponde a un parámetro del modelo.
- k es la cantidad de predictores.
- \hat{y} es una estimación de la respuesta.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k \quad (14.1)$$

Una vez más, al ajustar el modelo mediante el método de mínimos cuadrados, buscamos minimizar la suma de los cuadrados de los residuos (ecuación 14.2), proceso que se vuelve más complejo a medida que aumenta la cantidad de variables por lo que suele hacerse mediante el uso de software.

$$\min \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (14.2)$$

Para los ejemplos de este capítulo usaremos una vez más el conjunto de datos `mtcars`, considerando vehículos entre 2 y 5 mil libras de peso, cuyas variables describimos en la tabla 13.1. Como punto de partida, recordemos la RLS para predecir la potencia del motor a partir del volumen útil de sus cilindros (figura 13.6).

Ahora consideraremos su extensión a una RLM agregando como un segundo predictor el peso del vehículo (columna `wt`), modelo que podemos obtener mediante el script 14.1 y que se muestra en la figura 14.2.

Vemos que el procedimiento para construir un modelo de RLM en R es el mismo que usamos en el capítulo anterior, pero ahora en el lado derecho de la fórmula para ajustar el modelo tenemos que combinar ambos predictores (línea 8 del script). Al desplegar el resumen del modelo en pantalla (línea 9), vemos que la salida también es muy similar a lo visto, solo que ahora se reporta una línea extra bajo el encabezado “Coefficients”, con el detalle para el nuevo predictor `wt`. Similarmente, para usar este modelo a fin de predecir valores para la respuesta a partir de un nuevo conjunto de datos, usamos una vez más la función `predict()`, asegurando que los nuevos datos contengan valores para **todas las columnas** incluidas como predictores en el modelo.

Script 14.1: regresión lineal para predecir predecir la potencia del motor (vehículos entre 2 y 5 mil libras) a partir de dos variables: el volumen útil de sus cilindros y el peso del vehículo.

```

1 library(dplyr)
2 library(scatterplot3d)
3
4 # Cargar y filtrar los datos.
5 datos <- mtcars |> filter(wt > 2 & wt < 5)
6
7 # Ajustar modelo de LRM
8 modelo <- lm(hp ~ disp + wt, data = datos)

```

```

Call:
lm(formula = hp ~ disp + wt, data = datos)

Residuals:
    Min      1Q  Median      3Q     Max 
-62.701 -38.975   0.804   7.635 147.997 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 41.1736   66.4604   0.620  0.54194    
disp         0.5205    0.1556   3.345  0.00293 **  
wt          -3.0407   28.0046  -0.109  0.91452    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 46.63 on 22 degrees of freedom
Multiple R-squared:  0.5537, Adjusted R-squared:  0.5131 
F-statistic: 13.65 on 2 and 22 DF,  p-value: 0.00014

Predicciones:
      disp     wt   hp_est
1 246.540 3.307 159.4539
2 185.015 2.965 128.4671
3 317.097 3.699 194.9902
4 403.338 4.178 238.4263
5 325.263 3.744 199.1042
6 336.128 3.804 204.5775
7 200.359 3.050 136.1960
8 327.478 3.756 200.2207
9 232.060 3.226 152.1627
10 382.015 4.059 227.6885

```

Figura 14.1: salida del script 14.1 con el ejemplo de un modelo lineal múltiple para predecir la potencia de un automóvil a partir de dos variables numéricas.

```

9 print(summary(modelo))
10
11 # Graficar modelo ajustado, diferencia valores sobre y bajo el plano.
12 i_color <- 1 + (resid(modelo) > 0)
13 g <- scatterplot3d(
14   datos[["disp"]], datos[["wt"]], datos[["hp"]], type = "p", angle = 20,
15   pch = 16, color = c("darkorange", "steelblue")[i_color],
16   xlab = bquote("Volumen útil de los cilindros" ~ group("[", "in"^-3, "]")),
17   ylab = "Potencia [hp]",
18   zlab = "Peso [lb x 1000]"
19 )
20 g$plane3d(modelo, draw_lines = TRUE, lty = "dotted")
21
22 # Definir valores de los predictores para vehículos no incluidos
23 # en el conjunto mtcars
24 disp <- c(246.54, 185.015, 317.097, 403.338, 325.263,
25           336.128, 200.359, 327.478, 232.06, 382.015)
26 wt <- c(3.307, 2.965, 3.699, 4.178, 3.744,
27           3.804, 3.050, 3.756, 3.226, 4.059)
28 datos_nuevos <- data.frame(disp, wt)
29

```

```

30 # Usar el modelo para predecir el rendimiento de otros modelos.
31 hp_est <- predict(modelo, newdata = datos_nuevos)
32 datos_nuevos <- cbind(datos_nuevos, hp_est)
33
34 # Mostrar los resultados
35 cat("Predicciones:\n")
36 print(datos_nuevos)

```

Como en este caso tenemos dos predictores, lo que se ajusta ya no es una recta, sino un plano. La figura 14.2 muestra el modelo ejemplo, marcando en azul-acero los valores con un residuo positivo y en naranja los que tienen residuos negativos. En este caso, el plano corresponde al modelo:

$$\begin{aligned}\hat{hp} &= b_0 + b_1 \text{disp} + b_2 \text{wt} \\ &= 41,174 + 0,521 \cdot \text{disp} - 3,041 \cdot \text{wt}\end{aligned}$$

De este modo, b_1 es el parámetro ajustado para el volumen útil de los cilindros del motor y b_2 es el parámetro ajustado para el peso del vehículo, que pueden interpretarse como las pendientes en la dirección del eje x e y , respectivamente, del plano de la figura 14.2. A su vez, la intercepción fija la posición del plano con respecto al origen.

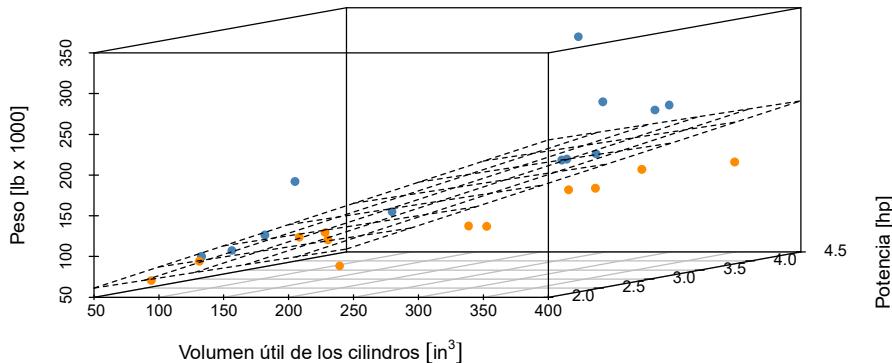


Figura 14.2: plano ajustado para la RLM ejemplo con dos predictores numéricos.

Así, ya no tiene sentido hablar de la pendiente de la recta al momento de interpretar los parámetros del modelo. Un análisis de regresión lineal con múltiples variables busca aislar la relación entre **cada predictor** y la respuesta, por lo que el coeficiente β_i del modelo, asociado al i -ésimo predictor, representa el cambio esperado que se produce en la respuesta al incrementar dicho predictor en una unidad, **manteniendo constantes todos los demás predictores**.

Desde luego, podemos extender esta idea para más de dos predictores. En tal caso ajustamos un **hiperplano** cuya forma y ubicación están dadas por los parámetros del modelo.

14.2 PREDICTORES CATEGÓRICOS NO DICOTÓMICOS

Al igual que con RLS, un modelo de RLM puede usar una variable categórica con dos niveles como predictor, haciendo la correspondiente transformación a una variable indicadora. Ahora vamos a extender esta idea para el caso de variables categóricas con k niveles, aplicando el siguiente algoritmo:

1. Crear $k - 1$ nuevas variables artificiales.

2. Para cada una de estas nuevas variables:

- a) Escoger un nivel diferente de la variable original.
- b) Asignar un 1 a todas las observaciones que tengan ese nivel y un 0 a las restantes.

Para entender mejor esta idea, supongamos que un conjunto de datos tiene la variable categórica `tipo`, con cuatro niveles: `A`, `B`, `C` y `D`. Creamos tres nuevas variables, por ejemplo, `tipo_B`, `tipo_C` y `tipo_D`. La variable `tipo_B` contiene tantas observaciones como la variable original, con un 1 para aquellas observaciones en que `tipo` toma el valor `B` y un 0 para las restantes. Para las variables `tipo_C` y `tipo_D` se procede de manera análoga. Puesto que solo se crean $k - 1$ variables artificiales, se descarta aquella correspondiente al nivel `A` de la variable `tipo`, cuyas observaciones se identifican porque todas las variables indicadoras tienen valor cero.

En R, podemos hacer esta tarea usando la función `dummy(x)` del paquete `dummy`, donde `x` es la matriz de datos (`data.frame`) original. La función devuelve una nueva matriz de datos con las variables indicadoras (*dummy*, en inglés) de todos los factores y vectores de strings en `x`. Existen otros argumentos que permiten obtener variables indicadoras para algunos valores categóricos, para obtener variables enteras (por defecto son strings), o considerando las categorías presentes en otro conjunto de datos. Esto último es útil cuando pueden existir diferencias en categorías encontradas en los datos que se están procesados y los datos usados para construir el modelo (datos de entrenamiento).

Por ejemplo, consideremos la tabla 14.1. A la izquierda se muestra una matriz de datos con una variable entera, dos variables categóricas, la primera dicotómica, y la segunda es la variable `tipo` descrita más arriba, y una columna numérica de tipo flotante. A la derecha se presenta la matriz de datos que resulta de aplicar la función `dummy()` a la primera. Podemos apreciar que aparecen variables indicadoras para cada uno de los niveles de las dos variables categóricas.

Matriz de datos original				Resultado de aplicar la función <code>dummy(x)</code>					
persona	sexo	tipo	valor	sexo_F	sexo_M	tipo_A	tipo_B	tipo_C	tipo_D
1	F	B	1.68	1	0	0	1	0	0
2	F	D	2.79	1	0	0	0	0	1
3	M	A	1.92	0	1	1	0	0	0
4	M	B	2.26	0	1	0	1	0	0
5	M	A	2.10	0	1	1	0	0	0
6	M	C	2.63	0	1	0	0	1	0
7	F	D	2.19	1	0	0	0	0	1
8	M	D	3.62	0	1	0	0	0	1
9	F	D	2.76	1	0	0	0	0	1

Tabla 14.1: creación de variables artificiales para una matriz de datos con variables categóricas.

El script 14.2 muestra el funcionamiento de `dummy()` para el ejemplo de la tabla ?? (línea 11). Como esta función crea una variable artificial por cada nivel de una variable categórica, debemos descartar una de ellas antes de utilizarlas para construir un modelo de RLM. Así, las líneas 12 y 13 eliminan dos de las variables indicadoras creadas, `sexo_F` y `tipo_A`, respectivamente, que son redundantes. Luego, la línea 14 agrega la variable `valor` a la matriz resultante. Con ello, es posible construir el modelo de RLM (línea 17) y mostrar los coeficientes obtenidos en pantalla (línea 18). El resultado puede verse en la figura 14.3

Como adelantamos al estudiar la RLS, la función `lm()` realiza internamente este proceso cuando recibe una variable categórica entre los predictores. Es más, las variables indicadoras eliminadas en el script 14.2 replican el criterio que utiliza la `lm()`, que descartada el primer nivel. La línea 21 ajusta el modelo de RLM usando el conjunto de datos original, dejando que la función `lm()` construya las variables indicadoras pertinentes. La línea 22 muestra el modelo que se obtiene con esta alternativa. Si nos fijamos en la figura 14.3, podemos ver que obtenemos el mismo modelo (con los mismos predictores, con iguales coeficientes).

```

Call:
lm(formula = valor ~ sexo_M + tipo_B + tipo_C + tipo_D, data = datos.dummy)

Coefficients:
(Intercept)      sexo_M1      tipo_B1      tipo_C1      tipo_D1
           1.154       0.856       0.388       0.620       1.472

Call:
lm(formula = valor ~ sexo + tipo, data = datos)

Coefficients:
(Intercept)      sexoM      tipoB      tipoC      tipoD
           1.154       0.856       0.388       0.620       1.472

```

Figura 14.3: resultado del script 14.2 que construye un modelo con variables indicadoras explícitas y de forma implícita.

Script 14.2: creación de variables artificiales para variables categóricas.

```

1 library(dummy)
2
3 # Crear una matriz de datos.
4 persona <- 1:9
5 sexo <- c("F", "F", "M", "M", "M", "M", "F", "M", "F")
6 tipo <- c("B", "D", "A", "B", "A", "C", "D", "D", "D")
7 valor <- c(1.68, 2.79, 1.92, 2.26, 2.1, 2.63, 2.19, 3.62, 2.76)
8 datos <- data.frame(persona, sexo, tipo, valor)
9
10 # Crear variables artificiales.
11 datos.dummy <- dummy(datos)
12 datos.dummy[["sexo_F"]] <- NULL
13 datos.dummy[["tipo_A"]] <- NULL
14 datos.dummy[["valor"]] <- datos[["valor"]]
15
16 # Crear y mostrar el modelo de RLM usando variables indicadoras
17 modelo <- lm(valor ~ sexo_M + tipo_B + tipo_C + tipo_D, datos.dummy)
18 print(modelo)
19
20 # Crear y mostrar el modelo de RLM dejando el trabajo a lm().
21 modelo_directo <- lm(valor ~ sexo + tipo, datos)
22 print(modelo_directo)

```

14.3 AJUSTE DE UN MODELO DE RLM

En el capítulo anterior introdujimos el coeficiente de determinación R^2 como instrumento para evaluar la bondad de ajuste de una regresión lineal simple, puesto que mide la proporción de la varianza total de la variable de salida que es explicada por el modelo. Sin embargo, cuando el modelo es multivariado, esta medida presenta un problema: R^2 aumenta siempre que se agregan más predictores al modelo, incluso si esos predictores no contribuyen significativamente a explicar la variabilidad de los datos, por lo que puede llevar a una falsa sensación de mejora del modelo.

Así, para evaluar una RLM tenemos que usar un **coeficiente de determinación ajustado**, que tenga en cuenta el número de predictores en el modelo. Diferentes autores han propuesto distintas maneras de efectuar este ajuste, una de las cuales presentamos en la ecuación 14.3, donde:

1. n es el número total de observaciones.
2. k es el número de predictores en el modelo (por lo que no se cuenta el intercepto).

$$R_{ajustado}^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - k - 1} \quad (14.3)$$

Si bien podemos usar este ajuste cuando tenemos un único predictor, la diferencia en este caso suele ser muy pequeña como para ser relevante. Pero en la RLM, al penalizar la **complejidad del modelo** (número de predictores), proporciona una métrica más realista, especialmente en modelos con muchos predictores. Un $R_{ajustado}^2 < R^2$ indica la presencia de predictores irrelevantes, que no han mejorado sustancialmente el ajuste del modelo. Por el contrario, un $R_{ajustado}^2 \approx R^2$ sugiere que las variables agregadas son relevantes para el modelo.

Preferir modelos más simples, con menos predictores, es conocido como el **principio de parsimonia**, o también como la navaja de Occam, es un concepto fundamental en la ciencia, la estadística y el aprendizaje automático (*machine learning*), que sostiene que, entre múltiples explicaciones o modelos que describen un fenómeno, se debe preferir el más simple que sea consistente con los datos disponibles, evitando así el exceso de supuestos o componentes si un modelo más sencillo es suficiente. Versiones del principio se puede rastrear hasta Aristóteles (Franklin, 2001), y siglos de uso han mostrado una y otra vez que la explicación más sencilla suele ser la mejor, puesto que un modelo más simple tiene menos probabilidad de ajustar ruido o variabilidad aleatoria en los datos, son más fáciles de entender, comunicar y justificar, y es más probable que un modelo parsimonioso generalice mejor a datos no vistos.

Existen otras alternativas para evaluar la bondad de ajuste de un modelo que también consideran el principio de parsimonia. Dos de ellas son el **criterio de información de Akaike**, abreviado AIC, y el **criterio bayesiano de Schwarz**, abreviado BIC o SBC, que penalizan el modelo por contener variables adicionales, por lo que mientras menor sea su valor, mejor será el modelo. Si bien el cálculo de estas medidas no se detalla aquí por ser un tópico más avanzado, podemos obtenerlas en R mediante las funciones `AIC(object)` y `BIC(object)`, donde `object` corresponde a un modelo lineal ya construido.

Para el modelo de RLS que usa únicamente el volumen útil de los cilindros del motor como predictor, obtenemos $AIC = 265,876$ y $BIC = 269,532$. Del mismo modo, para el modelo que usa como predictores esta variable y el peso del vehículo, en cambio, tenemos que $AIC = 267,862$ y $BIC = 272,738$. En consecuencia, usar el peso del automóvil como predictor parece no estar aportando a un mejor ajuste con los datos bajo estos criterios.

Esto no debería sorprendernos, ya que otra forma de conocer cuáles predictores aportan significativamente al ajuste del modelo es observar el estadístico t y los valores p asociados a cada predictor (columnas “`Pr(t value)`” y “`Pr(>|t|)`”). Para el ejemplo, estas se aprecian en la figura 14.2, donde vemos que nos indicaba que, al nivel $\alpha < 0,05$, no había evidencia de que la variable `wt` hiciera un aporte significativo ($t(23) = -0,109; p = 0,915$).

Si bien las métricas AIC y BIC nos pueden resultar útiles para comparar dos modelos de regresión lineal, considerando la regla general que un modelo es mejor mientras menor sea su valor, es importante notar que, al ser **medidas relativas**, hasta ahora no contamos con una prueba estadística que nos permita determinar si la diferencia es significativa, por lo que no tenemos cómo determinar a ciencia cierta si los modelos son similares o realmente uno de ellos es de mejor calidad.

Cuando los modelos son jerárquicos, es decir, el segundo incorpora nuevos predictores además de mantener los del primer modelo, podemos hacer una prueba de hipótesis usando los coeficientes de determinación. El estadístico de prueba se calcula mediante la ecuación 14.4, donde:

- n : cantidad de observaciones.
- k_1, k_2 : cantidad de predictores en el primer y segundo modelo, respectivamente, suponiendo $k_1 < k_2$.
- R_1^2, R_2^2 : coeficientes de determinación del primer y segundo modelo, respectivamente.

$$F_{cambio} = \frac{(n - k_2 - 1) (R_2^2 - R_1^2)}{(k_2 - k_1) (1 - R_2^2)} \quad (14.4)$$

Así, para los dos modelos ajustados hasta ahora tenemos:

$$F_{cambio} = \frac{(25 - 2 - 1) \cdot (0,5537 - 0,5534)}{1 \cdot (1 - 0,5537)} = 0,015$$

El estadístico de prueba sigue una distribución F con $(k_2 - k_1)$ y $n - k_2 - 1$ grados de libertad, a partir de lo cual podemos determinar el valor p correspondiente mediante la llamada `pf(0.015, 1, 22, lower.tail = FALSE)`, obteniendo como resultado $p = 0,904$. En consecuencia fallamos en rechazar la hipótesis nula y podemos concluir con 95 % de confianza que no hay evidencia de que el modelo de RLM construido se ajuste mejor a los datos que el modelo de LRS que ya teníamos.

Como ya es habitual, en R podemos hacer esta tarea de forma simple gracias a la función `anova(object, ...)`, que recibe como argumentos los diferentes modelos a comparar (pueden ser más de dos modelos jerárquicos). La interpretación del resultado de esta prueba es sencilla: si el valor p obtenido es significativo, entonces el modelo más complejo (con más predictores) se ajusta mejor a los datos (produce residuos significativamente menores).

El script 14.3 muestra cómo usar esta prueba para comparar el modelo nulo¹ con el modelo de un predictor, y luego este último con el modelo de dos predictores creados para el ejemplo que hemos seguido, todo con solo una llamada a la función `anova()`.

```

Modelo 0: AIC = 284.0302
Modelo 1: AIC = 265.8756
Modelo 2: AIC = 267.8622

Modelo 0: BIC = 286.468
Modelo 1: BIC = 269.5322
Modelo 2: BIC = 272.7377

Prueba de bondad de ajuste:
Analysis of Variance Table

Model 1: hp ~ 1
Model 2: hp ~ disp
Model 3: hp ~ disp + wt
  Res.Df   RSS Df Sum of Sq    F    Pr(>F)
  1     24 107174
  2     23  47859  1      59314 27.2802 3.075e-05 ***
  3     22  47834  1       26  0.0118   0.9145
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Figura 14.4: resultado del script 14.3 que compara el modelo nulo y los modelos de RLS y RLM ajustados como ejemplos.

Al ejecutar el script, se obtiene el resultado que presenta la figura 14.4, donde podemos ver que el modelo de RLS consigue reducir significativamente la varianza no explicada con respecto al modelo nulo ($F(1, 23) = 27,280$; $p < 0,001$) de 107.174 a 47.859, mientras que la reducción adicional que consigue el modelo de LRM es insignificante ($F(1, 22) = 0,012$; $p = 0,915$). Este resultado es similar al que obtuvimos anteriormente, con diferencias que se deben al redondeo.

¹Recordemos que el modelo nulo solo considera el intercepto, por lo que corresponde a la recta horizontal que pasa por la media de la variable de salida.

Script 14.3: comparación de los dos modelos lineales del ejemplo.

```

1 library(dplyr)
2
3 # Cargar y filtrar los datos.
4 datos <- mtcars |> filter(wt > 2 & wt < 5)
5
6 # Ajustar el modelo nulo, sin predictores,
7 # solo intercepto.
8 modelo_0 <- lm(hp ~ 1, data = datos)
9
10 # Ajustar un modelo con volumen de los cilindros
11 # como predictor.
12 modelo_1 <- lm(hp ~ disp, data = datos)
13
14 # Ajustar un modelo añadiendo el peso como predictor.
15 modelo_2 <- lm(hp ~ disp + wt, data = datos)
16
17 # Mostrar AIC y BIC de los modelos
18 cat("Modelo 0: AIC =", AIC(modelo_0), "\n")
19 cat("Modelo 1: AIC =", AIC(modelo_1), "\n")
20 cat("Modelo 2: AIC =", AIC(modelo_2), "\n")
21 cat("\n")
22 cat("Modelo 0: BIC =", BIC(modelo_0), "\n")
23 cat("Modelo 1: BIC =", BIC(modelo_1), "\n")
24 cat("Modelo 2: BIC =", BIC(modelo_2), "\n")
25
26 # Comparar los modelos.
27 comparacion <- anova(modelo_0, modelo_1, modelo_2)
28
29 cat("\n")
30 cat("Prueba de bondad de ajuste:\n")
31 print(comparacion)

```

14.4 SELECCIÓN DE PREDICTORES

Considerando el principio de parsimonia, es claro que debemos contar con alguna estrategia que nos permita determinar qué predictores incluir en un modelo de RLM, de manera que sean relevantes para mejorar su ajuste a los datos pero que, al mismo tiempo, no incluya más predictores de los necesarios para mantener su generalidad. Existen diversas alternativas para esta tarea que exploraremos a continuación.

14.4.1 Regresión jerárquica

El método más adecuado, aunque también el más complejo, es la **regresión jerárquica**. Es el que debemos considerar al momento de intentar probar una teoría y consiste en comenzar por incorporar en primer lugar aquellos predictores ya conocidos, en orden de importancia, en base a investigaciones previas (requiere una revisión bibliográfica). Una vez incorporados todos los predictores ya conocidos, podemos incorporar otros nuevos si creemos que existen **buenas y justificadas razones** para ello. Antes de la masificación de los computadores y de entornos como R, ésta era la única alternativa viable!

En R, podemos realizar este método con ayuda de la función `update(object, formula)`, que nos permite incorporar o quitar variables del modelo, donde:

- `object`: modelo previamente ajustado, en este caso con `lm()`.
- `formula`: actualización de la fórmula para el nuevo modelo. La especificación “`... . .`” indica “todo lo que hay al lado izquierdo de la fórmula” (el primer punto) y “todo lo que hay al lado derecho de la fórmula” (el segundo punto).

En este caso existe una teoría previa basada en la física, pero que es difícil de trasferir a modelos específicos. Podemos revisar esta teoría y complementarla con múltiples recursos documentales que discuten sobre la potencia y rendimiento de los motores, algunos de ellos se refieren a los motores de esa época, como por ejemplo:

- Aparicio Izquierdo, F., Vera Álvarez, C., & Díaz López, V. (2001). Universidad Politécnica de Madrid.
- Caliber. (s.f.). Car Design Through Time: From Classic to Contemporary. Caliber.com.
- Motor Trend Publications. Motor Trend Magazine collection. From the Collections of The Henry Ford. <https://www.thehenryford.org/collections-and-research/digital-collections>.
- Seger, E.E., & Brink, R.S. (1972). Trends of Vehicle Dimensions and Performance Characteristics from 1960 through 1970. Research Record, 1.

Revisando estos textos, encontramos que en los automóviles de los años 70, la potencia del motor estaba influenciada por varias variables mecánicas, de diseño y ambientales. A continuación, se describen las principales variables, más o menos en orden de importancia:

1. El volumen total de los cilindros del motor, pues motores con mayor volumen tienden a generar más potencia debido a su capacidad para quemar más mezcla de combustible y aire en cada ciclo.
2. Número de cilindros en el motor, por la misma razón anterior.
3. Relación de compresión entre el volumen del cilindro con el pistón en su posición más baja frente a su posición más alta, pues a mayor volumen más energía se extrae de la combustión.
4. Cantidad y tipo de carburadores, que influye en la capacidad del motor para mezclar aire y combustible de manera eficiente.
5. Eficiencia del sistema de escape, ya que una liberación rápida de gases mejora el rendimiento.
6. Peso del vehículo, ya que un auto pesado requiere más potencia, por lo que los fabricantes usan motores más potentes para autos grandes.
7. Configuración geométrica del motor (V, en línea, etc.), ya que influye en cómo se distribuye la potencia.
8. Ajuste de engranajes, pues relaciones de transmisión más ajustadas aumenta la entrega de potencia.
9. Leyes ambientales, porque en los años 70, especialmente en EE. UU., surgieron nuevas regulaciones que limitaron el uso de tecnologías que maximizaban la potencia en favor de reducir emisiones contaminantes.

Algunas de estas características no están incluidas en el conjunto de datos `mtcars`, otras están algo relacionadas a las variables registradas. Haciendo el análisis, podríamos seleccionar las siguientes variables, en orden de importancia, `disp`, `cyl`, `carb`, `wt` y `vs`. El script 14.4 presenta la aplicación de regresión jerárquica basada en esta información.

Script 14.4: aplicación de regresión jerárquica para construir un modelo de RLM para predecir la potencia del motor en automóviles que pesan entre 2 y 5 mil libras construidos en los años 70.

```

1 library(dplyr)
2
3 # Cargar y filtrar los datos.
4 datos <- mtcars |> filter(wt > 2 & wt < 5) |>
5   mutate_at(c("cyl", "vs", "am", "gear", "carb"), as.factor)
6
7 # Ajustar el modelo inicial con el volumen
8 # de los cilindros como predictor.
9 modelo_1 <- lm(hp ~ disp, data = datos)
10
11 # Incorporar al modelo el número de cilindros y
12 # verificar su utilidad.
13 modelo_2 <- update(modelo_1, . ~ . + cyl)
14 print(anova(modelo_1, modelo_2), signif.legend = FALSE)
15
16 # Como era esperable, la variable "cyl" no
17 # aporta al ajuste del modelo.
18
19 # Reemplazar el número de cilindros por el

```

```

20 # número de carburadores y verificar su utilidad.
21 modelo_3 <- update(modelo_2, . ~ . - cyl + carb)
22 cat("\n")
23 print(anova(modelo_1, modelo_3), signif.legend = FALSE)
24
25 # La variable "carb" sí genera un mejor ajuste,
26 # por lo que lo mantendremos en el modelo.
27
28 # Y en este último modelo, la variable "cyl"
29 # sigue siendo irrelevante? Veamos.
30 modelo_4 <- update(modelo_3, . ~ . + cyl)
31 cat("\n")
32 print(anova(modelo_3, modelo_4), signif.legend = FALSE)
33
34 # Ahora la variable "cyl" sí ayuda a obtener un
35 # mejor modelo!
36
37 # Incorporar al modelo el peso del vehículo y
38 # verificar su utilidad.
39 modelo_5 <- update(modelo_4, . ~ . + wt)
40 cat("\n")
41 print(anova(modelo_4, modelo_5), signif.legend = FALSE)
42
43 # Vemos que el peso no aporta a un mejor ajuste.
44 # Probablemente muy relacionado al número de
45 # cilindros y carburadores del motor?
46
47 # Reemplazar el peso del vehículo por el tipo de
48 # motor y verificar su utilidad.
49 modelo_6 <- update(modelo_5, . ~ . - wt + vs)
50 cat("\n")
51 print(anova(modelo_4, modelo_6), signif.legend = FALSE)
52
53 # Vemos que tipo de motor tampoco ayuda a conseguir
54 # un mejor modelo.
55
56 # Mostrar el modelo obtenido.
57 cat("\n\n")
58 cat("Modelo obtenido con regresión jerárquica:\n")
59 cat("-----\n")
60 print(summary(modelo_4), signif.legend = FALSE)

```

El resultado de ejecutar el script 14.4 se presenta en la figura 14.5. Comenzamos construyendo un modelo con un único predictor: el volumen de los cilindros (`disp`), el que, como sabemos, consigue un mejor ajuste que el modelo nulo (figura 14.4). En segundo lugar consideramos la variable categórica `cyl`, con el número de cilindros del vehículo. Como esta tiene directa relación con el predictor anterior, es poco probable que aporte información nueva, lo que es confirmado al comparar ambos modelos ($F(2, 21) = 2,536; p = 0,103$). Consideramos entonces la variable categórica `carb`, con el número de carburadores con que cuenta el automóvil. En este caso, la nueva variable sí aporta al modelo, mejorando su ajuste ($F(5, 18) = 14,080; p < 0,001$).

Ya tenemos un modelo de RLM con dos predictores. Como los coeficientes han cambiado, re-evaluamos que la variable `cyl` sigue siendo irrelevante, pero nos llevamos la sorpresa de que ahora esta *sí* aporta al modelo! ($F(2, 16) = 17,197; p < 0,001$), por lo que mantenemos el predictor. Luego de descartar la relevancia del peso del vehículo (`wt`; $F(1, 15) = 0,395; p = 0,539$) y del tipo de motor (`vs`; $F(1, 15) = 0,093; p = 0,765$), es con este modelo de tres predictores con el que nos quedamos en definitiva.

La línea 60 del script 14.4 despliega los detalles de este modelo en pantalla. Podemos ver que la variable `carb` aporta con cinco variables indicadoras al modelo, mientras que la variable `cyl` aporta con dos. Algo que también podríamos haber notado con las reducciones en los grados de libertad de las pruebas F que

Analysis of Variance Table							Analysis of Variance Table						
Model 1: hp ~ disp							Model 1: hp ~ disp + carb + cyl						
Model 2: hp ~ disp + cyl							Model 2: hp ~ disp + carb + cyl + vs						
Res.Df RSS Df Sum of Sq F Pr(>F)							Res.Df RSS Df Sum of Sq F Pr(>F)						
1 23 47859							1 16 3094.1						
2 21 38549 2 9310.7 2.5361 0.1031							2 15 3075.1 1 19.074 0.093 0.7645						
Analysis of Variance Table							Modelo obtenido con regresión jerárquica:						
Model 1: hp ~ disp							-----						
Model 2: hp ~ disp + carb							Call:						
Res.Df RSS Df Sum of Sq F Pr(>F)							lm(formula = hp ~ disp + carb + cyl, data = datos)						
1 23 47859							Residuals:						
2 18 9745 5 38114 14.08 1.096e-05 ***							Min 1Q Median 3Q Max						
Analysis of Variance Table							-30.422 -3.138 0.000 2.107 28.815						
Model 1: hp ~ disp + carb							Coefficients:						
Model 2: hp ~ disp + carb + cyl							Estimate Std. Error t value Pr(> t)						
Res.Df RSS Df Sum of Sq F Pr(>F)							(Intercept) 41.6148 15.7907 2.635 0.017997 *						
1 18 9745.3							disp 0.4762 0.1144 4.162 0.000734 ***						
2 16 3094.1 2 6651.2 17.197 0.0001033 ***							carb2 -19.0440 8.8794 -2.145 0.047671 *						
Analysis of Variance Table							carb3 27.9118 16.1811 1.725 0.103794						
Model 1: hp ~ disp + carb + cyl							carb4 57.1928 10.1001 5.663 3.53e-05 ***						
Model 2: hp ~ disp + carb + cyl + wt							carb6 120.9125 17.9545 6.734 4.80e-06 ***						
Res.Df RSS Df Sum of Sq F Pr(>F)							carb8 170.9128 18.4288 9.274 7.75e-08 ***						
1 16 3094.1							cyl6 -56.5693 13.9459 -4.056 0.000917 ***						
2 15 3014.7 1 79.433 0.3952 0.539							cyl8 -20.8492 29.1123 -0.716 0.484212						
Analysis of Variance Table							Residual standard error: 13.91 on 16 degrees of freedom						
Model 1: hp ~ disp + carb + cyl							Multiple R-squared: 0.9711, Adjusted R-squared: 0.9567						
Model 2: hp ~ disp + carb + cyl + vs							F-statistic: 67.28 on 8 and 16 DF, p-value: 7.366e-11						

Figura 14.5: resultado del script 14.4 con el ejemplo de regresión jerárquica.

realizamos. El modelo final logra un mucho mejor ajuste que el modelo nulo, disminuyendo la suma de los residuos cuadrados de 107.174 a 3.094, más de un 97 % de reducción.

14.4.2 Regresión paso a paso

Si en lugar de probar una teoría, lo que queremos es explorar los datos, podemos usar otras estrategias para seleccionar el conjunto de predictores para un modelo de LRM:

Selección hacia adelante A partir de un modelo nulo, se escoge la variable más prometedora para agregarla al modelo. Hay varias opciones para decidir esta variable más prometedora, que puede ser aquella que exhiba la correlación más alta con la variable de respuesta, o la que lleva el mayor aumento del coeficiente de determinación, o la que disminuye más el AIC del modelo, entre otras alternativas.

Si el predictor seleccionado incrementa la capacidad predictiva del modelo, se retiene de forma definitiva. En caso contrario, se descarta su inclusión. El procedimiento se repite y en cada iteración se escoge la siguiente variable más prometedora que aún no ha sido evaluada.

Si ninguno de los posibles predictores restantes promete mejorar el modelo que se tiene, se detiene la búsqueda de nuevos predictores.

Eliminación hacia atrás Es el proceso inverso a la selección hacia adelante, puesto que se comienza desde un modelo completo que incluye todas las variables disponibles, para luego ir eliminando predictores, uno a uno. Invirtiendo los criterios anteriores, se elige el predictor que menos aporta al modelo. Si la eliminación este último mejora el modelo de acuerdo al criterio utilizado, se elimina de forma definitiva y se reevalúa la contribución de los predictores que aún se encuentran en el modelo. Una vez más, el

proceso se repite hasta que no es posible eliminar un predictor sin perjudicar .

Regresión escalonada Combina los enfoques anteriores. Comienza desde un modelo vacío o completo y alterna entre agregar y eliminar predictores, evaluando en cada paso si es necesario incluir o excluir una variable según el criterio utilizado, de forma de reducir el efecto debido al orden de incorporación de los predictores.

Como estos métodos seleccionan un predictor en cada iteración, ya sea para agregarlo o para quitarlo del modelo de RLM, se les conocen como métodos de selección de predictores paso a paso o **regresión paso a paso** (*stepwise regression*, en inglés).

Es importante reiterar que solo debemos usar estos métodos si estamos **explorando datos**, pues de lo contrario incurriremos en **faltas a la ética**. Veamos por ejemplo el trabajo de Montero Muñoz et al. (2012), donde estudian algunas implicaciones clínicas en el uso de antibióticos en ancianos mayores de 80 años. Podemos ver que, en la recolección de datos, recopilaron variables que la medicina ha probado a lo largo de su historia que son importantes al evaluar la salud de un paciente y que pueden ser afectadas por el uso de medicamentos. Un equipo poco ético podría haber empleado otras variables registradas en las bases de datos de origen, por ejemplo, el monto de la pensión del paciente. ¿Qué consecuencias podrían existir si tuviéramos una correlación espuria de esta variable con la variable de respuesta? ¡Podríamos afectar las vidas de muchas personas si tal estudio concluyera que los antibióticos son más nocivos para ancianos con bajas pensiones!

Los métodos de regresión paso a paso están disponibles en R con ayuda de la ya conocida función `update()` y las funciones `add1(object, scope, test)` y `drop1(object, scope, test)`, donde:

- `object` : modelo de regresión base.
- `scope` : fórmula que especifica los potenciales predictores que se podrían agregar (o quitar), o un modelo de regresión creado con esta fórmula. También puede ser la matriz de datos con los potenciales predictores (que no incluya la variable de salida). En el caso de la función `drop1()`, puede omitirse para indicar que todos los predictores del modelo pueden ser candidatos a eliminación.
- `test` : tipo de la prueba de hipótesis a aplicar al comparar el modelo base y el potencial nuevo modelo. Por defecto tiene valor `"none"`, es decir no aplica prueba alguna. Se puede indicar el valor `"F"` para aplicar la prueba basada en la reducción de varianza no explicada que hemos usado con la función `anova()` al comparar modelos de RLM (y que es la que reporta la función `summary()` para un modelo de regresión lineal. Otra alternativa es `chisq` para aplicar la prueba de razón de verosimilitud (LRT, del inglés **likelihood ratio test**, que discutiremos más adelante (en el próximo capítulo).

Las funciones `add1()` y `drop1()` evalúan la incorporación o retirada, respectivamente, de cada predictor potencial (individualmente) al modelo base y entrega algunas métricas del efecto que su incorporación tendría en un nuevo modelo:

- Bajo el título “`Sum of Sq`” muestra en cuánto el nuevo modelo reduciría o aumentaría, respectivamente, la varianza no explicada respecto del modelo base.
- Bajo el título “`RSS`” muestra cuánta varianza no explicada quedaría en los residuos.
- Bajo el título “`AIC`” muestra el AIC que tendría el nuevo modelo.
- Si se ha especificado que se aplique una prueba F en cada caso, la función reporta el estadístico bajo el título “`F value`”.
- Cuando se ha aplicado alguna prueba de igualdad de ajuste, la función reporta el valor p para el estadístico encontrado bajo el título “`Pr(>F)`” o “`Pr(>Chi)`” según la prueba escogida.

Script 14.5: ejemplo de regresión paso a paso para construir modelos de RLM para predecir la potencia del motor.

```

1 library(dplyr)
2
3 # Cargar y filtrar los datos.
4 datos <- mtcars |> filter(wt > 2 & wt < 5) |>
5   mutate_at(c("cyl", "vs", "am", "gear", "carb"), as.factor)
6
7 # Ajustar el modelo nulo y el modelo completo.
8 nulo <- lm(hp ~ 1, data = datos)

```

```

9 completo <- lm(hp ~ ., data = datos)
10
11 cat("Selección hacia adelante:\n")
12 cat("-----\n\n")
13
14 # Evaluar variables para incorporar.
15 paso <- add1(nulo, scope = completo, test = "F")
16 print(paso, digits = 3, signif.legend = FALSE)
17
18 # Agregar la variable que logra la mayor reducción
19 # significativa de varianza no explicada.
20 modelo <- update(nulo, . ~ . + cyl)
21
22 # Evaluar variables para incorporar.
23 paso <- add1(modelo, scope = completo, test = "F")
24 cat("\n")
25 print(paso, digits = 3, signif.legend = FALSE)
26
27 # Agregar la variable que logra la mayor reducción
28 # significativa de varianza no explicada.
29 modelo <- update(modelo, . ~ . + carb)
30
31 # Mostrar los coeficientes del modelo conseguido.
32 cat("\nModelo obtenido:\n")
33 print(modelo[["coefficients"]])
34
35 cat("\n\n")
36 cat("Eliminación hacia atrás:\n")
37 cat("-----\n\n")
38
39 # Evaluar variables para eliminar.
40 paso <- drop1(completo, test = "F")
41 print(paso, digits = 3, signif.legend = FALSE)
42
43 # Quitar la variable con menor estadístico F.
44 modelo <- update(completo, . ~ . - wt)
45
46 # Evaluar variables para eliminar.
47 paso <- drop1(modelo, test = "F")
48 cat("\n")
49 print(paso, digits = 3, signif.legend = FALSE)
50
51 # Quitar la variable con menor estadístico F.
52 modelo <- update(modelo, . ~ . - drat)
53
54 # Mostrar los coeficientes del modelo conseguido.
55 cat("\nModelo obtenido:\n")
56 print(modelo[["coefficients"]])

```

El resultado de ejecutar el script 14.5 se presenta en la figura 14.6. En el lado izquierdo se aprecia lo obtenido con selección hacia adelante, mientras que en el lado derecho lo que obtenemos con eliminación hacia atrás.

Para el primer método comenzamos con un modelo nulo. La función `add1()` nos entrega información de los modelos con un predictor. Para este ejemplo, usamos como criterio elegir la variable que conlleva a la mayor reducción de varianza no explicada. Vemos que esta corresponde a la variable `cyl`, que hace un aporte significativo al ajuste del modelo, por lo que es seleccionada como primer predictor (línea 20 del script). En la segunda iteración (línea 23) encontramos que la variable `carb` disminuye significativamente la varianza del modelo con un anterior, por lo que es incluida (línea 29). Al mostrar los coeficientes de este modelo en pantalla (línea 33), vemos que se han ajustado coeficientes para siete variables indicadores, pues ambos predictores escogidos con el criterio mencionado, son factores.

Selección hacia adelante:

Single term additions

Model:

hp ~ 1

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
<none>		107174	211			
mpg	1	60602	46572	192	29.93	1.5e-05 ***
cyl	2	68625	38549	190	19.58	1.3e-05 ***
disp	1	59314	47859	193	28.50	2.0e-05 ***
drat	1	5585	101589	212	1.26	0.27240
wt	1	35016	72158	203	11.16	0.00284 **
qsec	1	61023	46151	192	30.41	1.3e-05 ***
vs	1	44410	62763	200	16.27	0.00052 ***
am	1	2	107171	213	0.00	0.98208
gear	2	44037	63137	202	7.67	0.00297 **
carb	5	61903	45271	200	5.20	0.00359 **

Single term additions

Model:

hp ~ cyl

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
<none>		38549	190			
mpg	1	4222	34327	189	2.58	0.1230
disp	1	0	38549	192	0.00	0.9964
drat	1	10241	28308	184	7.60	0.0118 *
wt	1	948	37600	191	0.53	0.4748
qsec	1	11620	26929	183	9.06	0.0067 **
vs	1	388	38161	191	0.21	0.6489
am	1	9686	28863	184	7.05	0.0148 *
gear	2	18503	20046	177	9.23	0.0014 **
carb	5	32105	6444	155	16.94	4.5e-06 ***

Modelo obtenido:

(Intercept)	cyl6	cyl8	carb2
99.90	-14.25	95.10	-21.57
carb3	carb4	carb6	carb8
-15.00	41.77	89.35	140.00

Eliminación hacia atrás:

Single term deletions

Model:

hp ~ mpg + cyl + disp + drat + wt + qsec + vs + am + gear + carb

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
<none>		368	101.2			
mpg	1	224	592	111.1	4.86	0.05848 .
cyl	2	1931	2299	143.0	20.99	0.00066 ***
disp	1	1005	1373	132.1	21.86	0.00159 **
drat	1	72	439	103.7	1.56	0.24763
wt	1	5	373	99.6	0.11	0.74709
qsec	1	253	621	112.3	5.51	0.04688 *
vs	1	445	813	119.1	9.68	0.01440 *
am	1	714	1081	126.2	15.52	0.00430 **
gear	2	579	947	120.9	6.30	0.02275 *
carb	5	9543	9911	173.6	41.51	1.6e-05 ***

Single term deletions

Model:

hp ~ mpg + cyl + disp + drat + qsec + vs + am + gear + carb

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
<none>		373	99.6			
mpg	1	219	592	109.1	5.28	0.04722 *
cyl	2	2175	2548	143.6	26.24	0.00018 ***
disp	1	1429	1802	136.9	34.48	0.00024 ***
drat	1	75	448	102.2	1.82	0.21030
qsec	1	250	623	110.4	6.03	0.03643 *
vs	1	482	855	118.3	11.63	0.00774 **
am	1	813	1186	126.5	19.62	0.00165 **
gear	2	620	993	120.0	7.47	0.01223 *
carb	5	10545	10918	174.0	50.89	2.5e-06 ***

Modelo obtenido:

(Intercept)	mpg	cyl6	cyl8	disp
-70.944601	-1.489398	-39.626433	16.251136	0.362381
5.999012	41.012233	50.258777	-50.073373	-13.167903
15.190355	53.043426	95.563457	132.291369	178.281305

Figura 14.6: resultado del script 14.5 con el ejemplo de regresión paso a paso.

Para la eliminación hacia atrás comenzamos con el modelo completo, e iremos eliminando los predictores que presenten el menor valor del estadístico F. En la primera iteración, la función `drop1()` nos informa que la eliminación más prometedora corresponde a la variable `wt`, la que se quita del modelo en la línea 44 del script. En la segunda iteración (línea 47) encontramos que la variable `drat` puede ser eliminada del modelo sin afectar significativamente su ajuste (línea 52). La línea 56 muestra los coeficientes resultantes para el resto de los predictores.

Si bien el script 14.5 muestra dos iteraciones de selección hacia adelante y otras dos de eliminación hacia atrás, en ambos casos podríamos seguir aplicando iteraciones hasta descartar que exista un paso que pueda mejorar el ajuste del modelo.

Por supuesto, R cuenta con funciones que implementan los métodos para seleccionar predictores paso a paso mediante la función `step(object, scope, direction, trace)`, que usa las funciones `add1()` y `drop1()` de manera iterativa, donde:

- `object`: modelo de regresión que es usado como punto de partida.
- `scope`: define el rango de búsqueda de los modelos. Esto puede ser una sola fórmula o una lista que

contenga los componentes `lower` y `upper`, ambas fórmulas (o modelos que contengan estas fórmulas). La parte derecha de la fórmula `lower` siempre está incluida en el modelo, mientras que la parte derecha del modelo siempre estará incluida en `upper`. Si solo se especifica una fórmula, esta se asigna al componente `upper` y se asume que el límite inferior es el modelo nulo. Si no se especifica scope, el modelo inicial se usa como el modelo `upper`.

- `direction`: indica el tipo de selección a realizar, donde `"forward"` corresponde a selección hacia adelante; `"backward"`, a eliminación hacia atrás, y `"both"`, a regresión escalonada.
- `trace`: argumento opcional que indica si se quiere ver por consola el proceso realizado.

La función `step()` utiliza el criterio de información de Akaike para guiar la búsqueda, seleccionando el predictor que conlleva al modelo con menor AIC, agregándolo al modelo cuando realiza un paso hacia adelante, o quitándolo de él cuando realiza un paso hacia atrás. Si uno quisiera que se usara el criterio de información Bayesiana para guiar la búsqueda, que tiende a seleccionar modelos más simples pero un poco menos precisos que el AIC, se puede lograr cambiando el valor de la penalización a la complejidad del modelo (el número de parámetros). Por defecto se usa el valor $k = 2$, que corresponde al usado por el AIC, que debe ser cambiado por el valor $k = \log n$, donde n corresponde al tamaño de la muestra de datos. La función `step()` acepta un argumento `k` para estos fines. Sin embargo, cualquiera sea el valor de k que especifiquemos, la función de todas forma lo reporta como “AIC”.

El script 14.6 muestra el funcionamiento de la función `step()` usando el BIC como criterio para seleccionar los predictores a incorporar en un modelo donde la respuesta es, una vez más, la potencia del motor de un automóvil construido en los años 70 que pesa entre 2 mil y 5 mil libras.

Script 14.6: regresión paso a paso (escalonada) para seleccionar los predictores a incluir en una RLM para predecir la potencia del motor en automóviles que pesan entre 2 y 5 mil libras construidos en los años 70.

```

1 library(dplyr)
2
3 # Cargar y filtrar los datos.
4 datos <- mtcars |> filter(wt > 2 & wt < 5) |>
5   mutate_at(c("cyl", "vs", "am", "gear", "carb"), as.factor)
6
7 # Ajustar el modelo nulo y el modelo completo.
8 nulo <- lm(hp ~ 1, data = datos)
9 completo <- lm(hp ~ ., data = datos)
10
11 # Realiza regresión escalonada usando el menor BIC
12 # como criterio (aunque se reporta como AIC), bajando
13 # (temporalmente) el número de cifras significativas
14 # y el ancho máximo de la pantalla al imprimir.
15 opt <- options(digits = 2, width = 54)
16 modelo <- step(nulo, scope = list(lower = nulo, upper = completo),
17                  direction = "both", k = log(nrow(datos)),
18                  test = "F", trace = 1)
19 options(digits = opt[[1]], width = opt[[2]])
20
21 # Mostrar los coeficientes del modelo conseguido
22 cat ("\nModelo obtenido:\n")
23 print (modelo[["coefficients"]])

```

Las líneas 8 y 9 del script ajustan los modelos nulo y completo para ser usados como los límites inferior y superior, respectivamente, de la regresión paso a paso. En la llamada a `step()` de las líneas 16–18, hacemos explícito el valor 1 para el argumento `trace` (que es el que tiene por defecto), por lo que la función muestra información de los modelos evaluados en cada paso. La figura 14.7 muestra la salida producida en las primeras cuatro iteraciones, y la figura 14.8 muestra el resto de la información desplegada por esta llamada².

²Las líneas inmediatamente anterior y posterior a esta llamada simplemente reducen y restauran al valor original, respectivamente, el número de cifras significativas y el número máximo de caracteres en cada línea a ser usado por las funciones y métodos de R que imprimen en pantalla. El único propósito de esto es obtener una salida con un formato apropiado para ser mostrado en dos columnas como en estas figuras.

```

Start: AIC=212
hp ~ 1

          Df Sum of Sq   RSS AIC F value    Pr(>F)
+ cyl    2   68625  38549 193  19.58 1.3e-05 ***
+ qsec   1   61023  46151 194  30.41 1.3e-05 ***
+ mpg    1   60602  46572 195  29.93 1.5e-05 ***
+ disp   1   59314  47859 195  28.50 2.0e-05 ***
+ vs     1   44410  62763 202  16.27 0.00052 ***
+ gear   2   44037  63137 206  7.67 0.00297 **
+ wt     1   35016  72158 206  11.16 0.00284 **
+ carb   5   61903  45271 207  5.20 0.00359 **
<none>
          107174 212
+ drat   1      5585 101589 214  1.26 0.27240
+ am     1      2 107171 216  0.00 0.98208
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Step: AIC=193
hp ~ cyl

          Df Sum of Sq   RSS AIC F value    Pr(>F)
+ carb   5   32105  6444 165  16.94 4.5e-06 ***
+ gear   2   18503  20046 183  9.23 0.0014 **
+ qsec   1   11620  26929 187  9.06 0.0067 **
+ drat   1   10241  28308 189  7.60 0.0118 *
+ am     1   9686  28863 189  7.05 0.0148 *
<none>
          38549 193
+ mpg    1   4222  34327 194  2.58 0.1230
+ wt     1   948  37600 196  0.53 0.4748
+ vs     1   388  38161 196  0.21 0.6489
+ disp   1      0 38549 196  0.00 0.9964
- cyl   2   68625 107174 212  19.58 1.3e-05 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Step: AIC=165
hp ~ cyl + carb

          Df Sum of Sq   RSS AIC F value    Pr(>F)
+ disp   1   3350  3094 149  17.32 0.00073 ***
+ mpg    1   1133  5311 163  3.41 0.08323 .
+ gear   2   1644  4800 164  2.57 0.10984
<none>
          6444 165
+ wt     1   463  5981 166  1.24 0.28199
+ vs     1   276  6168 167  0.72 0.41011
+ drat   1   161  6283 167  0.41 0.53064
+ am     1      8 6437 168  0.02 0.89263
+ qsec   1      6 6438 168  0.02 0.90112
- carb   5   32105 38549 193  16.94 4.5e-06 ***
- cyl   2   38827 45271 207  51.21 6.4e-08 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Step: AIC=149
hp ~ cyl + carb + disp

          Df Sum of Sq   RSS AIC F value    Pr(>F)
+ mpg    1      585  2510 147  3.49 0.08123 .
+ drat   1      511  2583 148  2.97 0.10540
<none>
          3094 149
+ am     1      274  2820 150  1.46 0.24588
+ gear   2      614  2480 150  1.73 0.21254
+ wt     1      79  3015 152  0.40 0.53902
+ qsec   1      71  3023 152  0.35 0.56263
+ vs     1      19  3075 152  0.09 0.76454
- disp   1      3350 6444 165  17.32 0.00073 ***
- cyl   2      6651 9745 172  17.20 0.00010 ***
- carb   5   35455 38549 196  36.67 3.3e-08 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Figura 14.7: resultado de las primeras cuatro iteraciones de la regresión escalonada implementada en el script 14.6.

Notemos que la función comienza por calcular el BIC (aunque reportado como AIC) para el modelo nulo, para luego determinar las métricas que se obtendría para los diferentes modelos que se obtendrían agregando solo una de las variables disponibles. El predictor que se escoge es aquel que genera el modelo con menor BIC, y que encabeza la lista pues se muestran ordenados por este valor, que corresponde al número de cilindros del motor (`cyl`).

En la siguiente iteración, vemos que explora los diferentes modelos que se obtendrían agregando solo una del resto de las variables disponibles al modelo que incluye el predictor `cyl`, así como el que se obtendría si se eliminara esta variable. La lista es encabezada por un paso hacia adelante agregando el número de carburadores del automóvil (`carb`).

En las siguientes iteraciones se repite este procedimiento: explora los modelos que se obtendrían agregando una de las variables no incluidas en el modelo base, y los que se obtendría eliminando uno de los predictores ya incluidos en él; ordena los modelos por su valor de BIC, y realiza el paso que encabeza la lista ordenada. De esta forma, agrega al modelo el volumen de los cilindros del motor (`disp`) en la iteración 3, el rendimiento del vehículo (`mpg`) en la cuarta y la relación de transmisión del eje trasero (`drat`) en la quinta.

En la iteración 6 resulta que el modelo que encabeza la lista de posibles pasos corresponde al modelo que ya se tiene, que la función `step()` marca como `<none >`. Esto detiene la búsqueda y la función reporta los coeficientes del modelo que ha conseguido.

```

Step: AIC=147
hp ~ cyl + carb + disp + mpg

  Df Sum of Sq  RSS AIC F value Pr(>F)
+ drat  1     1263 1246 133   14.20 0.00208 **
+ am    1      818 1691 141   6.77 0.02088 *
+ gear  2      891 1618 143   3.58 0.05770 .
+ wt    1      439 2071 146   2.97 0.10704
<none>
2510 147
+ qsec  1      233 2277 148   1.43 0.25149
- mpg   1      585 3094 149   3.49 0.08123 .
+ vs    1      17  2492 150   0.10 0.76087
- disp  1      2801 5311 163   16.74 0.00096 ***
- cyl   2      7119 9628 175   21.28 4.2e-05 ***
- carb  5      31804 34314 197   38.02 5.3e-08 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```



```

Step: AIC=133
hp ~ cyl + carb + disp + mpg + drat

  Df Sum of Sq  RSS AIC F value Pr(>F)
<none>
1246 133
+ am    1      143 1103 133   1.68 0.2170
+ wt    1      36  1210 136   0.39 0.5444
+ qsec  1      1  1245 136   0.01 0.9205
+ vs    1      1  1245 136   0.01 0.9269
+ gear  2      40 1206 139   0.20 0.8206
- drat  1      1263 2510 147   14.20 0.0021 **
- mpg   1      1337 2583 148   15.02 0.0017 **
- cyl   2      1722 2968 148   9.68 0.0023 **
- disp  1      3178 4424 162   35.70 3.4e-05 ***
- carb  5      22314 23560 190   50.14 1.9e-08 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```


Modelo obtenido:

(Intercept)	cyl6	cyl8	carb2
-16.81	-35.64	-7.93	-28.85
carb3	carb4	carb6	carb8
14.87	14.49	90.59	135.07
disp	mpg	drat	
0.47	-3.63	38.63	

Figura 14.8: resultado de las últimas dos iteraciones de la regresión escalonada implementada en el script 14.6 y el modelo obtenido.

14.4.3 Búsqueda exhaustiva

Si bien la regresión paso a paso implementada en R es bastante cómoda, la búsqueda que se realiza no es otra cosa que una implementación de una **búsqueda golosa**, por lo que no tenemos ninguna garantía de que consiga la combinación óptima de predictores, en el sentido de que permitan construir el modelo de RLM que maximiza o minimiza algún criterio.

Con el avance del poder de cómputo, se ha hecho posible una alternativa más exhaustiva que aplica un algoritmo de fuerza bruta en el que se exploran todos los subconjuntos de predictores que cumplan con algún criterio. Esto último es muy relevante, puesto que el conjunto completo de posibilidades corresponde a 2^k , siendo k la cantidad de potenciales predictores. Para nuestro ejemplo con el conjunto de datos `mtcars`, con 11 variables, el conjunto a explorar contiene $2^{10} = 1.024$ combinaciones, lo que es bastante abordable en estos tiempos. Sin embargo, para un conjunto de datos con 31 o 51 columnas habría que evaluar, respectivamente, 1.073.741.824 y 1.125.899.906.842.624 combinaciones distintas, lo que es altamente costoso en tiempo de ejecución.

El paquete `leaps` implementa en la función `regsubsets(formula, data, nbest, nvmax, force.in, force.out, method = "exhaustive")` este método de búsqueda exhaustiva, donde:

- `formula`: la especificación del modelo como se entregaría a la función `lm()`.
- `data`: matriz de datos.
- `nbest`: cantidad de modelos a reportar por cada tamaño de subconjunto. Si, por ejemplo, `nbest = 3`, entonces se reportan los tres mejores modelos con un único predictor, los tres mejores modelos con dos predictores, etc.
- `nvmax`: fija una cantidad máxima de predictores a considerar en el modelo.
- `force.in`: argumento opcional que toma la forma de un vector con los índices de las columnas que deben ser forzosamente consideradas en los modelos evaluados.
- `force.out`: argumento opcional que toma la forma de un vector con los índices de las columnas que deben ser forzosamente excluidas en los modelos evaluados.

El script 14.7 presenta el uso del método de todos los subconjuntos al ejemplo que hemos seguido con el conjunto de datos `mtcars`. En las líneas 9–11 se llama a la función `regsubsets()` indicando que reporte solo el

mejor modelo encontrado para cada tamaño del modelo, el que está permitido llegar a 16 predictores. Puede llamar la atención este número, puesto que la matriz de datos del ejemplo solo contiene 10 posibles predictores. Esto se debe a que la búsqueda se realiza con los factores ya transformados en variables indicadoras (sin las que son redundantes).

Script 14.7: regresión utilizando el método de todos los subconjuntos para construir un modelo de RLM para el ejemplo con el conjunto de datos `mtcars`.

```

1 library(dplyr)
2 library(leaps)
3
4 # Cargar y filtrar los datos.
5 datos <- mtcars |> filter(wt > 2 & wt < 5) |>
6   mutate_at(c("cyl", "vs", "am", "gear", "carb"), as.factor)
7
8 # Evaluar todos las combinaciones
9 combinaciones <- regsubsets(hp ~ ., data = datos,
10                           nbest = 1, nvmax = 16,
11                           method = "exhaustive")
12
13 # Graficar los resultados
14 plot(combinaciones)
15
16 # Extraer los mejores subconjuntos
17 comb_summary <- summary(combinaciones)
18 i_min_bic <- which.min(comb_summary[["bic"]])
19 i_max_r2a <- which.max(comb_summary[["adjr2"]])
20
21 mejor_comb_bic <- comb_summary[["which"]][i_min_bic, ]
22 mejor_comb_r2a <- comb_summary[["which"]][i_max_r2a, ]
23
24 # Extraer las variables seleccionadas
25 comb_mejor_bic <- names(mejor_comb_bic[mejor_comb_bic == TRUE])
26 comb_mejor_r2a <- names(mejor_comb_r2a[mejor_comb_r2a == TRUE])
27
28 # Eliminar variables indicadoras
29 nombres_mejor_bic <- unique(gsub("(.*)\d$", "\1", comb_mejor_bic))
30 nombres_mejor_r2a <- unique(gsub("(.*)\d$", "\1", comb_mejor_r2a))
31
32 # Obtener las fórmulas
33 pred_mejor_bic <- paste(nombres_mejor_bic[-1], collapse = " + ")
34 pred_mejor_r2a <- paste(nombres_mejor_r2a[-1], collapse = " + ")
35
36 fmla_mejor_bic <- as.formula(paste("hp", pred_mejor_bic, sep = " ~ "))
37 fmla_mejor_r2a <- as.formula(paste("hp", pred_mejor_r2a, sep = " ~ "))
38
39 # Construir y mostrar los mejores modelos
40 modelo_mejor_bic <- lm(fmla_mejor_bic, data = datos)
41 modelo_mejor_r2a <- lm(fmla_mejor_r2a, data = datos)
42
43 cat("Modelo que minimiza el BIC:\n")
44 cat("-----\n")
45 print(modelo_mejor_bic)
46 cat("\n")
47 cat("Modelo que maximiza el coeficiente de determinación ajustado:\n")
48 cat("-----\n")
49 print(modelo_mejor_r2a)

```

El objeto que retorna la función `regsubsets()` contiene mucha información de los subconjuntos encontrados y es un tanto complejo darle sentido. Por esta razón, es más recomendable revisar una representación gráfica

graficar de los resultados, que puede obtenerse con la función base de R `plot()`, como hace la línea 14 del script 14.7. La figura 14.9 muestra el gráfico generado por esta llamada. El eje *x* lista todas las variables disponibles para ser usadas en el modelo (de acuerdo al `scope` especificado e incluyendo las variables indicadoras que se necesiten) y el eje *y* corresponde a una lista ordenada de los valores del BIC registrado por cada uno de los mejores modelos evaluados. Así, la figura corresponde a una matriz en que cada fila representa a uno de los mejores subconjuntos de predictores encontrados para cada tamaño, donde se colorea el recuadro para las variables presentes en dicho modelo, con un tono más oscuro para menores valores de BIC. Notemos que el eje *y* se encuentra invertido, por lo que las mejores combinaciones aparecen en la parte superior de la matriz.

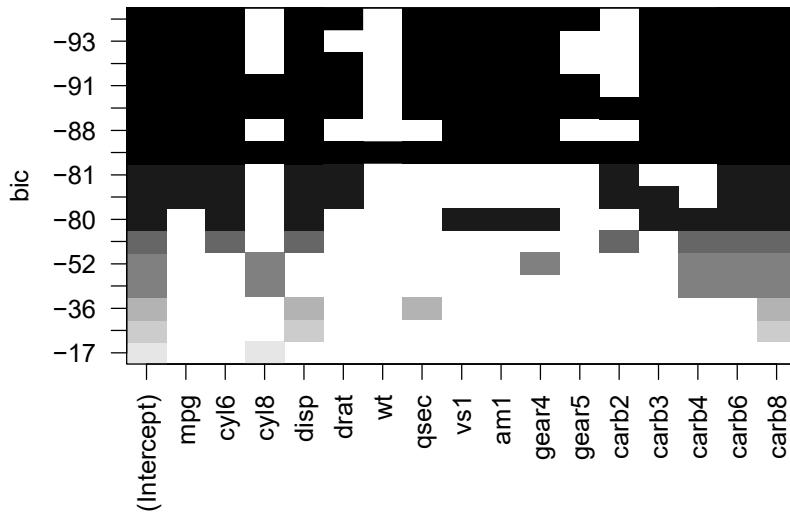


Figura 14.9: representación gráfica de las mejores combinaciones de predictores encontradas mediante el método de todos los subconjuntos para el ejemplo con el conjunto de datos `mtcars`.

Específicamente para el ejemplo, como se solicitó, se muestra la mejor combinación con un predictor, la mejor con dos predictores, la mejor con tres, y así sucesivamente hasta el modelo completo. La mejor combinación, al tope de la matriz, considera, además del intercepto, el rendimiento del vehículo (`mpg`), la variable indicadora de motor con 6 cilindros (`cyl6`), el volumen de los cilindros (`disp`), la relación de transmisión del eje trasero (`drat`), el tiempo necesario para recorrer un cuarto de milla desde el reposo (`qsec`), el tipo de motor (`vs1`), el tipo de transmisión (`am1`), el número de marchas hacia adelante (`gear4` y `gear5`), y cuatro variables indicadoras del número de carburadores (`carb3`, `carb4`, `carb6`, y `carb8`).

Si bien el conjunto de predictores seleccionados no incluye todos niveles de algunos predictores categóricos, es preferible reconstruir el modelo con los predictores categóricos completos. Así el modelo resultante es más generalizable a datos nuevos y más fácil de interpretar por quien lo use, aún cuando sea un poco menos parsimonioso que el encontrado por la función `regsubsets()`. En general, la tarea de reconocer las variables categóricas originales para esta reconstrucción es una tarea manual, pero en este caso, y solo porque todos los factores tenían niveles representados con números enteros, se pudo hacer de manera automática con las sentencias de las líneas 24–41 del script 14.7. El resultado, puesto en pantalla por las líneas 43–49, se muestra en la figura 14.10. Podemos ver que, usando el BIC o el coeficiente de determinación ajustado como criterio para seleccionar el conjunto de predictores, se llega al mismo modelo.

14.5 CONFIABILIDAD DE UN MODELO DE RLM

Al igual que en el caso de la regresión lineal simple (RLS), en la RLM las relaciones entre la variable de salida y los predictores requieren verificar condiciones importantes, alguna de las cuales son difíciles de evaluar. Afortunadamente, su cumplimiento puede transferirse a condiciones que debe cumplir el modelo que son más simples de comprobar. Haciendo explícitas otras consideraciones importantes para obtener un modelo de RLM confiable, podemos considerar:

Modelo que minimiza el BIC:

```
Call:  
lm(formula = fmla_mejor_bic, data = datos)
```

Coefficients:

Intercept)	mpg	cyl6	cyl8	disp	drat
-109.161	-1.918	-33.626	18.317	0.385	14.756
qsec	vs1	am1	gear4	gear5	carb2
5.815	36.075	45.616	-47.469	-18.402	9.595
carb3	carb4	carb6	carb8		
47.894	80.643	128.986	174.382		

Modelo que maximiza el coeficiente de determinación ajustado:

```
Call:  
lm(formula = fmla_mejor_r2a, data = datos)
```

Coefficients:

(Intercept)	mpg	cyl6	cyl8	disp	drat
-109.161	-1.918	-33.626	18.317	0.385	14.756
qsec	vs1	am1	gear4	gear5	carb2
5.815	36.075	45.616	-47.469	-18.402	9.595
carb3	carb4	carb6	carb8		
47.894	80.643	128.986	174.382		

Figura 14.10: modelos que resultan de seleccionar predictores con el método de todos los subconjuntos para para el ejemplo.

1. La variable de respuesta debe ser cuantitativa y continua, sin restricciones para su variabilidad.
 2. Los predictores deben ser cuantitativos o dicotómicos (de ahí la necesidad de variables indicadoras para manejar más de dos niveles).
 3. Los predictores deben tener algún grado de variabilidad (su varianza no debe ser igual a cero). En otras palabras, no pueden ser constantes.
 4. Cada predictor debe estar relacionado linealmente con la respuesta.
 5. La distribución de los residuos debe ser cercana a la normal centrada en cero.
 6. La variabilidad de los residuos debe ser aproximadamente constante (homocedasticidad).
 7. Los residuos deben ser independientes entre sí.
 8. No debe existir **multicolinealidad**. Esto significa que no deben darse relaciones lineales *fuertes* (coeficientes de correlación altos) entre dos o más predictores.
 9. Las estimaciones de los coeficientes del modelo no deben estar alterados por unos pocas observaciones influyentes.

En general, es sencillo comprobar las primeras tres condiciones, aunque a veces puede haber duda con variables no físicas, como puntuaciones que varían dentro de un intervalo dado. También puede haber alguna dificultad con decidir si un predictor tiene al menos una escala de intervalos iguales. Pero, como vimos, convertir factores en variables indicadoras es simple, al igual que comprobar que una columna de una matriz de datos no tiene un valor constante. /jljara/.texlive2021/texmf-var/fonts/pk/ljfou/jknappen/ec/ecbx1200.600pk></home/jljara/.texlive2021/texmf-var/fonts/pk/ljfou/jknappen/ec/ecti1000.600pk></home/jljara/.texlive2021/texmf-var/fonts/pk/ljfou/jknappen/ec/ecbx1000.600pk></home/jljara/.texlive202 Para verificar las condiciones 4–6, podemos aplicar el mismo procedimiento y herramientas que usamos para evaluar modelos de RLS. Así, la figura 14.11 presenta gráficos de residuos (arriba) y gráficos marginales (abajo) para el modelo obtenido con regresión exhaustiva. El primero se obtiene con la función `residualPlots()`, mientras que la segunda con la función `residualPlots()` del paquete `car`. Los parámetros gráficos utilizados se explican en 13.5.2. El único cambio es que hemos dado el valor `formula(~ mpg + disp + drat + qsec")` al argumento `terms`

en ambas llamadas, para que solo se consideren los predictores numéricos en los gráficos, ya que no tienen sentido para las variables categóricas.

Analizando los gráficos de residuos para cada predictor, podemos ver que no se observan patrones sistemáticos que pudieran ser problemáticos. Podría quedar alguna duda sobre la condición de homocedasticidad con la variable `qsec`, pero este efecto visual podría deberse a que no hay muchas observaciones con valores sobre 20. También debemos poner atención en la curvatura que se observa con valores altos en este predictor y para `drat`, pero esto también podría deberse únicamente a la falta de observaciones en esta región.

Revisando los valores predichos (*fitted values*), vemos que tampoco hay residuos muy lejos de la línea del cero, no hay patrones sistemáticos, parece mantenerse una dispersión constante, aunque también aparece una curvatura hacia valores altos de potencia, pero que podría deberse a la mencionada falta de datos. Si recordamos, la función `residualPlots()`, además de agregar las curvas en los gráficos, nos reporta pruebas de linealidad (t de Student para cada predictor y de no aditividad de Tukey para el modelo), las que en este caso dan:

	Test stat	Pr(> Test stat)
mpg	-0.0657	0.9492
disp	0.0266	0.9795
drat	1.1098	0.2993
qsec	0.7801	0.4578
Tukey test	-0.8035	0.4217

Como vimos en el capítulo de RLS, también podemos usar aplicar pruebas de homocedasticidad implementada en la función `ncvTest()` del mismo paquete, para descartar las dudas que tenemos.

```
Non-constant Variance Score Test
Variance formula: ~ mpg
Chisquare = 0.3489181, Df = 1, p = 0.55473
```

```
Non-constant Variance Score Test
Variance formula: ~ disp
Chisquare = 0.1457182, Df = 1, p = 0.70266
```

```
Non-constant Variance Score Test
Variance formula: ~ drat
Chisquare = 0.005625423, Df = 1, p = 0.94021
```

```
Non-constant Variance Score Test
Variance formula: ~ qsec
Chisquare = 0.4634622, Df = 1, p = 0.49601
```

```
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 0.3872895, Df = 1, p = 0.53373
```

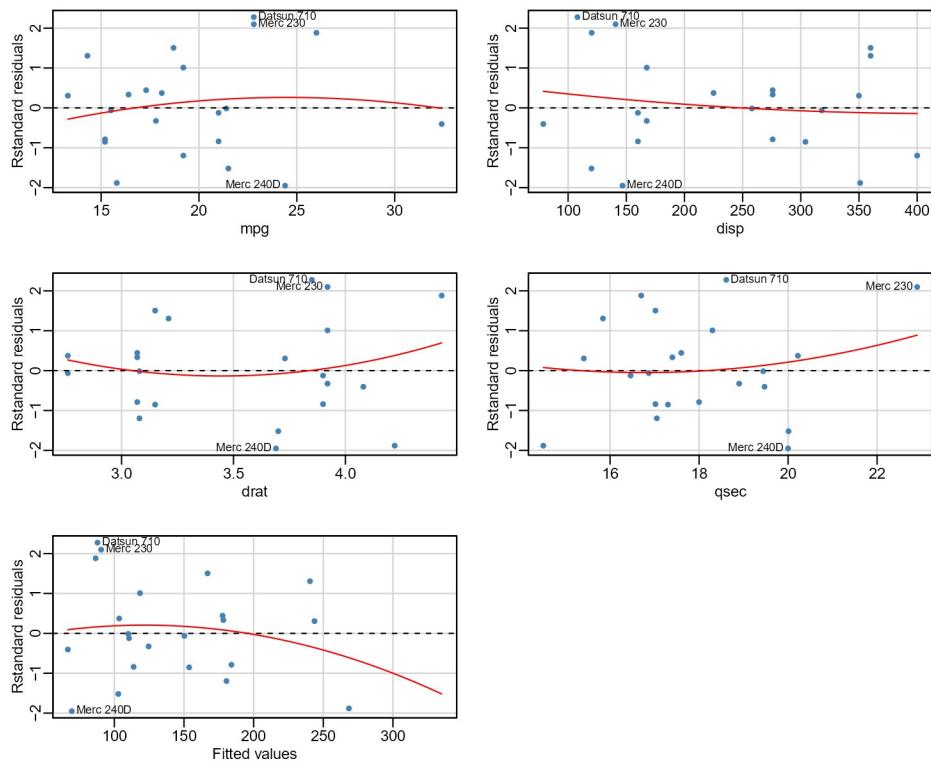
De esta forma, no podemos descartar que las relaciones entre la variable de salida y los predictores sean lineales, ni que la varianza de los residuos producidos por el modelo es constante.

Revisemos los gráficos marginales del modelo que aparecen en la parte inferior de la figura 14.11. Debemos recordar que esta figura muestra en colores distintos curvas que aproximan los valores observados de la variable de salida y los valores predichos por el modelo, además de comparar las desviaciones estándar.

A primera vista los gráficos son preocupantes. Para el predictor `mpg` observamos que una leve curvatura y una reducción de la dispersión a medida que crece el valor de la variable (rendimiento). Algo similar se observa para el predictor `qsec`. El predictor `disp` muestra una relación más lineal, pero su dispersión parece aumentar para volúmenes altos. El gráfico para el predictor `drat` parece incluso más preocupante, ya que claramente describe una curva, aunque la dispersión parece relativamente constante.

Sin embargo, como se mencionó anteriormente, estas curvaturas se dan en regiones donde no hay muchas observaciones, por lo que son más bien una suposición del método de aproximación usado para generar estas

Gráficos de residuos



Gráficos marginales

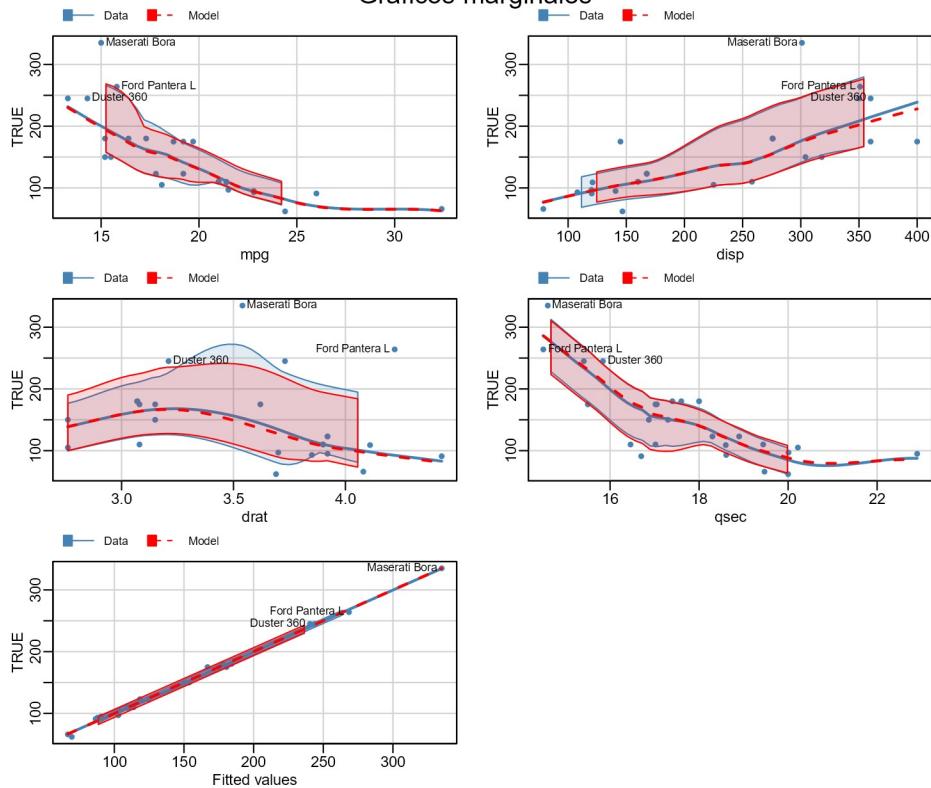


Figura 14.11: gráficos de residuos y marginales para el modelo de RLM de la figura 14.10.

curvas. En las regiones donde se concentran los datos ($15 \leq \text{mpg} \leq 25$; $75 \leq \text{disp} \leq 360$; $3 \leq \text{drat} \leq 4$); $15 \leq \text{qsec} \leq 20$), vemos que el comportamiento es aproximadamente lineal. Algo similar ocurre con la dispersión. En resumen, como comprobamos con las pruebas aplicadas, con los pocos datos que se tienen no es posible descartar que el modelo esté cumpliendo bien las condiciones para usar RLM.

El gráfico marginal para las predicciones, al estar en la escala de la variable de salida, hace más patente el buen ajuste que alcanza el modelo, pues los errores son tan pequeños que parecen estar sobre la línea de regresión.

Pasando a la condición 7, nuevamente podemos utilizar lo hecho para RLS y aplicar la prueba de Durbin-Watson implementada en la función `durbinWatsonTest` del paquete `car`, obteniendo:

```
lag Autocorrelation D-W Statistic p-value
 1      -0.2912021    2.581483   0.752
Alternative hypothesis: rho != 0
```

Con esto descartamos que exista autocorrelación entre los residuos, por lo que no hay evidencia de que no se esté cumpliendo la condición de independencia.

La condición 8 no es realmente una imposición de la RLM, pero evitar la multicolinealidad es importante porque el método para ajustar los coeficientes de un modelo asume que puede cambiar el valor para una variable predictora, *manteniendo los otros constantes*. Cuando las variables predictoras están correlacionadas, se hace imposible cambiar el valor de un coeficiente sin alterar también los de las demás variables, desestabilizando la estimación de cómo influye cada una en la variable de salida. Así, cuando existe multicolinealidad, los valores de los coeficientes varían enormemente si se agregan o quitan unos pocos datos de entrenamiento o, peor aún, se cambia el orden en que se agregan los predictores al modelo. En consecuencia, la multicolinealidad afecta la calidad del modelo al reducir su poder estadístico y su generalidad.

Por esta razón, necesitamos contar con alguna estrategia que nos permita escoger un conjunto de predictores lo más independientes entre sí para que puedan contribuir con información no correlacionada al modelo de RLM. La alternativa más usada se basa en el **factor de inflación de varianza** (*VIF*, por *variance inflation factor*) y el estadístico **tolerancia** ($1/VIF$).

El *VIF* para cada predictor i se calcula mediante la ecuación 14.5, donde R_i^2 corresponde al coeficiente de determinación de una regresión auxiliar que se obtiene usando todos los predictores restantes para ajustar una RLM con el predictor i como respuesta (Fox & Weisberg, 2018).

$$VIF_i = \frac{1}{1 - R_i^2} \quad (14.5)$$

Sin embargo, la definición de la ecuación 14.5 no es adecuada para modelos de regresión que incluyen predictores categóricos codificados con múltiples variables indicadoras. Para estos modelos se debe calcular **factores de inflación de varianza generalizados** (*GVIF*) utilizando la ecuación 14.5, donde \mathbf{C}_{full} corresponde a la matriz de covarianza del modelo completo y \mathbf{C}_{fullj} a la matriz de covarianza del modelo sin el predictor j (Fox & Weisberg, 2018).

$$GVIF_i = \frac{\det(\mathbf{C}_{full})}{\det(\mathbf{C}_{fullj})} \quad (14.6)$$

Además, los *VIF* generalizados pueden ajustarse de tal forma que queden en la misma escala que los factores de inflación de la varianza usuales usando la relación $GVIF_{ajustado} = GVIF^{1/(2 df)}$, donde df son los grados de libertad del predictor, igual al número de coeficientes que aporta al modelo (1 para variables continuas y $k - 1$ para variables categóricas con k niveles). Cuando se usa $GVIF_{ajustado}$, estos valores reemplazan a los valores normales de *VIF* al calcular la tolerancia de cada predictor.

Aunque no hay un acuerdo general, es común encontrar los siguientes criterios para interpretar los valores de *VIF* (o $GVIF_{ajustado}$):

- $VIF = 1$: no hay multicolinealidad.
- $1 < VIF \leq 5$: existe multicolinealidad moderada que podría afectar ligeramente los resultados, pero generalmente no es motivo de gran preocupación.
- $5 < VIF \leq 10$: la multicolinealidad es preocupante y podría afectar significativamente los resultados, por lo que sería recomendable investigar más a fondo y considerar acciones correctivas.
- $VIF > 10$: La multicolinealidad es severa y debe abordarse pues es probable que los coeficientes sean inestables y las inferencias estadísticas poco fiables.

En el caso de la tolerancia, la literatura sugiere que valores bajo 0,2 podrían ser problemáticos, pues esto puede interpretarse como que el 80 % de la varianza de ese predictor está explicada por los demás predictores del modelo, una clara señal de multicolinealidad. Valores cercanos a 0,4 también deberían ser revisados.

El paquete `car` de R incluye la función `vif(modelo)` para calcular los factores de inflación de la varianza (o los generalizados y ajustados, según corresponda) para un `modelo` de RLM. Al usarla para el modelo ejemplo (figura 14.10, obtenemos:

	GVIF	Df	GVIF^(1/(2*Df))
mpg	7.446394	1	2.728808
cyl	135.733676	2	3.413280
disp	23.904251	1	4.889197
drat	15.759580	1	3.969834
qsec	12.433984	1	3.526185
vs	16.194473	1	4.024236
am	14.742095	1	3.839544
gear	115.447392	2	3.277902
carb	854.968528	5	1.964242

Podría ser prudente revisar si un modelo que no considere la variable `disp`, que es la que muestra un valor cercano a 5, podría resultar más conveniente. Tarea que dejaremos pendiente para los ejercicios del capítulo.

La última condición en realidad tampoco es una restricción de la RLM pero, como vimos para la RLS, la presencia de observaciones sobreinfluyentes afectan las estimaciones del modelo y su generalización a datos no vistos.

Siguiendo lo realizado con modelos de RLS, podemos utilizar la función `influencePlot()` del paquete `car`, de la misma forma que hicimos en 13.5.3, para obtener una representación gráfica de posibles valores atípicos, el apalancamiento y distancia de Cook de los residuos generados por el modelo.

La figura 14.12 muestra el gráfico obtenido para el modelo ejemplo que estamos analizando. En pantalla se identifican los siguientes casos posiblemente problemáticos:

	StudRes	Hat	CookD
Datsun 710	3.293151	0.8805307	2.3858412
Merc 240D	-2.420409	0.6464100	0.4347061
Merc 230	2.766756	0.8900778	2.2271810
Ford Pantera L	-2.278866	0.8664138	1.4360599
Ferrari Dino	NaN	1.0000000	NaN
Maserati Bora	NaN	1.0000000	NaN
Volvo 142E	NaN	1.0000000	NaN

Vemos que hay tres casos con `japalancamiento igual a 1!` (y `distancia de Cook indefinida!`) y otros tres con distancias de Cook muy superiores al umbral 1. Es muy probable que la regresión esté dominada por estos casos. Sin embargo, el problema podría ser que el modelo está **sobreajustado** (*overfitting* en inglés), por lo que hace predicciones perfectas (cero error) o casi perfectas con estos datos, pero probablemente sea incapaz de generalizar a datos no visto. El problema también podría ser que existe una multicolinealidad severa entre los predictores del modelo, que la métrica $GVIF_{ajustado}$ no logra detectar. Claro que también podría tratarse simplemente de valores atípicos que están distorsionando el ajuste del modelo para la mayoría de los casos.

Como se dijo al discutir la RLS, eliminar las observaciones problemáticas podría constituir una acción reñida con la ética profesional si la decisión no se basa en un análisis riguroso de los datos.

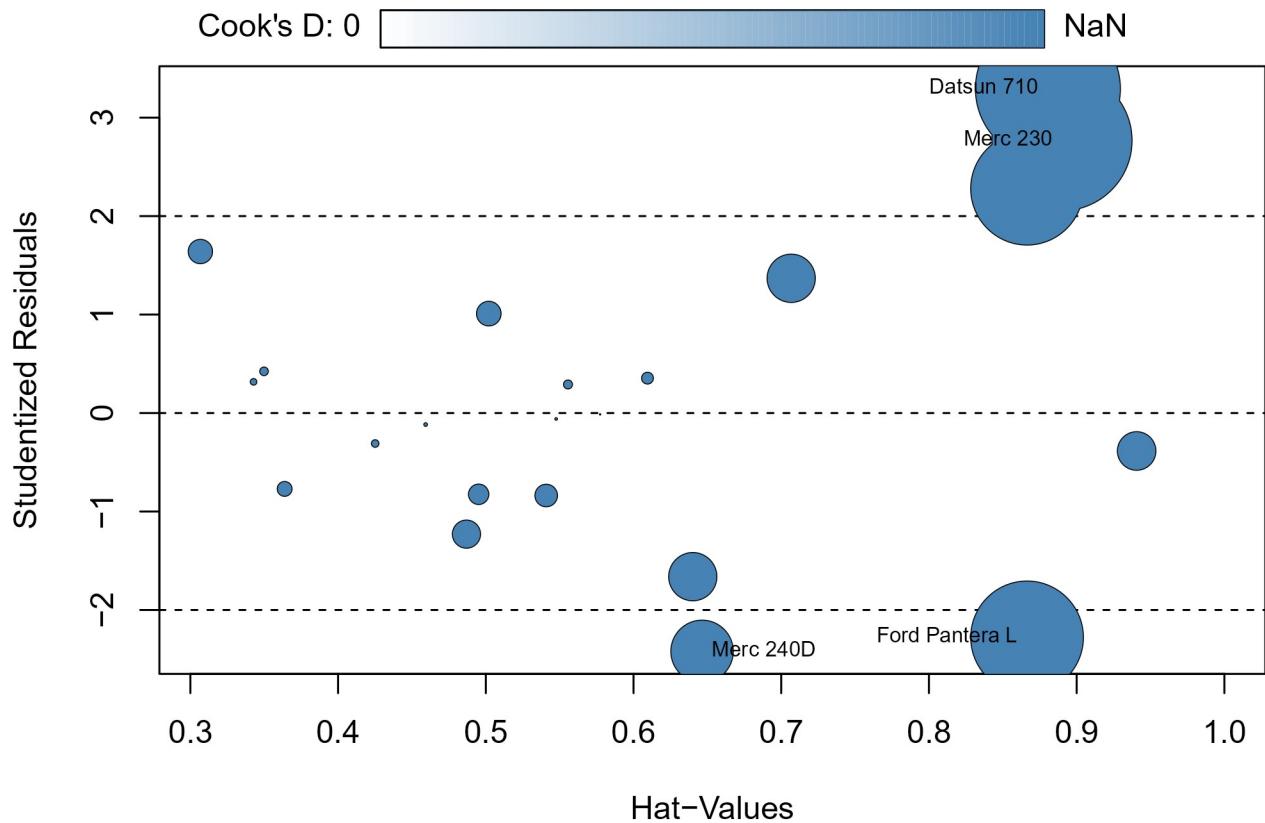


Figura 14.12: gráfico de influencia para el modelo de RLM de la figura 14.10.

De hecho, para el ejemplo, se da una combinación de sobreajuste, multicolinealidad y casos excepcionales. Se requieren varias iteraciones seleccionando predictores y evaluando modelos para conseguir un modelo más confiable. Queda como desafío para el lector o lectora.

Aquí vale la pena hacer un comentario sobre otro aspecto importante a tener en cuenta al momento de determinar si un modelo de RLM es confiable: el tamaño de la muestra. No existe un regla o criterio general que permita determinar el tamaño de la muestra que necesitamos, más que mientras más observaciones tengamos, mejor. Una regla simple, con cierto grado de aceptación, es verificar que se tengan al menos 10 o 15 observaciones por cada predictor numérico y cada nivel de las variables categóricas. De acuerdo a este criterio, la muestra del ejemplo cuenta con 25 observaciones, con lo que está muy lejos de ser suficiente para conseguir un modelo de 15 variables (considerando las variables indicadoras).

14.6 CALIDAD PREDICTIVA DE UN MODELO DE RLM

Al igual que en el caso de regresión lineal simple, también podemos usar validación cruzada como herramienta para mejorar la estimación del error cuadrado medio del modelo. No hay diferencia en la realización de este proceso con respecto a lo estudiado para la RLS, por lo que no se aborda aquí con mayor detalle.

Como ejercicio académico, y sabiendo que tiene problemas, el script ?? ajusta el modelo del ejemplo analizado en la sección anterior usando validación cruzada dejando uno fuera (líneas 16–28). La figura 14.13, a la izquierda, muestra el resultado que se obtiene. Notemos los mensajes de advertencia (*warnings*) que salen a pantalla al realizar esta operación, pues la función `predict()`, que internamente llama la función `train()`,

detecta que el modelo que se está utilizando tiene problemas.

Para comparar, en las líneas 36–48 del script se repite el proceso para un modelo con un predictor menos (`carb`). El resultado se muestra en el lado derecho de la figura 14.13. Vemos que ahora no aparecen las advertencias de que el modelo tiene problemas y que este exhibe una mejor calidad predictiva (es más generalizable que el anterior).

Modelo obtenido con completo `regsubset()`:

```
-----
Warning in predict.lm(modelFit, newdata) :
  prediction from rank-deficient fit; attr(*,
  "non-estim") has doubtful cases
Warning in predict.lm(modelFit, newdata) :
  prediction from rank-deficient fit; attr(*,
  "non-estim") has doubtful cases
```

```
hp ~ mpg + cyl + disp + drat + qsec + vs + am +
gear + carb
```

Predicciones en cada pliegue:

	pred	obs	rowIndex	intercept
Mazda RX4	111.08386	110	1	TRUE
Mazda RX4 Wag	117.61153	110	2	TRUE
Datsun 710	50.61264	93	3	TRUE
Hornet 4 Drive	110.14338	110	4	TRUE
Hornet Sportabout	163.36360	175	5	TRUE
Valiant	101.15911	105	6	TRUE
Duster 360	229.46875	245	7	TRUE
Merc 240D	83.11698	62	8	TRUE
Merc 230	54.26651	95	9	TRUE
Merc 280	113.79861	123	10	TRUE
Merc 280C	125.77044	123	11	TRUE
Merc 450SE	177.34866	180	12	TRUE
Merc 450SL	176.45313	180	13	TRUE
Merc 450SLC	186.36359	180	14	TRUE
Fiat 128	76.69676	66	15	TRUE
Toyota Corona	113.30605	97	16	TRUE
Dodge Challenger	150.60757	150	17	TRUE
AMC Javelin	158.09354	150	18	TRUE
Camaro Z28	242.04393	245	19	TRUE
Camaro Z28	242.04393	245	19	TRUE
Pontiac Firebird	185.74943	175	20	TRUE
Porsche 914-2	57.84787	91	21	TRUE
Ford Pantera L	297.15213	264	22	TRUE
Ferrari Dino	46.01351	175	23	TRUE
Maserati Bora	160.61755	335	24	TRUE
Volvo 142E	-65.38245	109	25	TRUE

Error estimado para el modelo:

intercept	RMSE	Rsquared	MAE
1	TRUE	47.14171	0.6214021
			34.24198

Modelo con un predictor menos:

```
-----
hp ~ mpg + cyl + disp + drat + qsec + vs + am +
gear
```

Predicciones en cada pliegue:

	pred	obs	rowIndex	intercept
Mazda RX4	101.73743	110	1	TRUE
Mazda RX4 Wag	97.09869	110	2	TRUE
Datsun 710	117.71142	93	3	TRUE
Hornet 4 Drive	102.18601	110	4	TRUE
Hornet Sportabout	176.99516	175	5	TRUE
Valiant	121.90452	105	6	TRUE
Duster 360	194.65422	245	7	TRUE
Merc 240D	78.85562	62	8	TRUE
Merc 230	-28.96070	95	9	TRUE
Merc 280	135.85810	123	10	TRUE
Merc 280C	143.56208	123	11	TRUE
Merc 450SE	178.62351	180	12	TRUE
Merc 450SL	170.90247	180	13	TRUE
Merc 450SLC	184.48076	180	14	TRUE
Fiat 128	-10.89509	66	15	TRUE
Toyota Corona	62.07997	97	16	TRUE
Dodge Challenger	197.87334	150	17	TRUE
AMC Javelin	196.45208	150	18	TRUE
Camaro Z28	203.51950	245	19	TRUE
Pontiac Firebird	177.12708	175	20	TRUE
Porsche 914-2	156.94908	91	21	TRUE
Ford Pantera L	353.64082	264	22	TRUE
Ferrari Dino	150.86278	175	23	TRUE
Maserati Bora	241.30126	335	24	TRUE
Volvo 142E	129.74939	109	25	TRUE

Error estimado para el modelo:

intercept	RMSE	Rsquared	MAE
1	TRUE	47.14171	0.6214021
			34.24198

Figura 14.13: resultado de aplicar validación cruzada dejando uno fuera para construir el modelo de RLM de la figura 14.10, y un modelo similar quitando un predictor.

Script 14.8: comparación de los dos modelos lineales del ejemplo.

```
1 library(caret)
2 library(dplyr)
3
4 # Imprimir mensajes de advertencia a medida que ocurre.
5 opt <- options(warn = 1)
6
7 # Cargar y filtrar los datos.
8 datos <- mtcars |> filter(wt > 2 & wt < 5) |>
9   mutate_at(c("cyl", "vs", "am", "gear", "carb"), as.factor)
```

```

10
11 cat("Modelo obtenido con regsubset():\n")
12 cat("-----\n\n")
13
14 # Ajustar y mostrar el modelo usando validación cruzada
15 # dejando uno fuera.
16 set.seed(111)
17 fmla <- formula("hp ~ mpg + cyl + disp + drat + qsec + vs + am + gear + carb")
18 entrenamiento <- train(fmla, data = datos, method = "lm",
19                         trControl = trainControl(method = "LOOCV"))
20 modelo <- entrenamiento[["finalModel"]]
21
22 # Mostrar la fórmula y las predicciones del modelo.
23 cat("\n")
24 print(fmla)
25 cat("\n")
26
27 cat("Predicciones en cada pliegue:\n")
28 print(entrenamiento[["pred"]])
29
30
31 cat("Error estimado para el modelo:\n")
32 print(entrenamiento[["results"]])
33
34
35 cat("\n\n")
36 cat("Modelo con un predictor menos:\n")
37 cat("-----\n\n")
38
39 # Ajustar y mostrar el modelo usando validación cruzada
40 # dejando uno fuera sin la variable "carb".
41 set.seed(111)
42 fmla <- formula("hp ~ mpg + cyl + disp + drat + qsec + vs + am + gear")
43 entrenamiento <- train(fmla, data = datos, method = "lm",
44                         trControl = trainControl(method = "LOOCV"))
45 modelo <- entrenamiento[["finalModel"]]
46
47 # Mostrar la fórmula y las predicciones del modelo modificado.
48 print(fmla)
49 cat("\n")
50
51 cat("Predicciones en cada pliegue:\n")
52 print(entrenamiento[["pred"]])
53
54 # Mostrar el resultado estimado para el modelo.
55 cat("\nError estimado para el modelo:\n")
56 print(entrenamiento[["results"]])
57
58 # Reestable opción para warnings
59 options(warn = opt[[1]])

```

14.7 EJERCICIOS PROPUESTOS

- 14.1** Tome una muestra de 100 observaciones del conjunto de datos `Prestige` del paquete `carData`, y construya un modelo de RLM con dos o tres predictores para la variable de salida `income` y evalúa su confiabilidad y calidad predictiva usando validación cruzada simple.
- 14.2** Tome una muestra de 100 observaciones del conjunto de datos `Prestige` del paquete `carData`, y cons-

truya un modelo de RLM con dos o tres predictores para la variable de salida `prestige` y evalúa su confiabilidad y calidad predictiva usando validación cruzada simple de 4 pliegues.

- [14.3] Tome una muestra de 100 observaciones del conjunto de datos `Prestige` del paquete `carData`, y construya un modelo de RLM con dos o tres predictores para la variable de salida `education` y evalúa su confiabilidad y calidad predictiva usando validación cruzada dejando uno fuera.
- [14.4] Tome una muestra de 250 observaciones del conjunto de datos `Boston` del paquete `MASS`, y construya un modelo de RLM con dos o tres predictores para la variable de salida `medv` y evalúa su confiabilidad y calidad predictiva usando validación cruzada simple.
- [14.5] Tome una muestra de 250 observaciones del conjunto de datos `Boston` del paquete `MASS`, y construya un modelo de RLM con dos o tres predictores para la variable de salida `rm` y evalúa su confiabilidad y calidad predictiva usando validación cruzada simple de 10 pliegues.
- [14.6] Tome una muestra de 250 observaciones del conjunto de datos `Boston` del paquete `MASS` que tengan valores menores a 400 en la variable de salida `tax`, y construya un modelo de RLM con dos o tres predictores para predecirla. Luego evalúa su confiabilidad y calidad predictiva usando validación cruzada dejando uno fuera.

14.8 BIBLIOGRAFÍA DEL CAPÍTULO

Diez, D., Barr, C. D., & Çetinkaya-Rundel, M. (2017). *OpenIntro Statistics* (3.ª ed.).
<https://www.openintro.org/book/os/>.

Field, A., Miles, J., & Field, Z. (2012). *Discovering statistics using R*. SAGE Publications Ltd.

Fox, J., & Weisberg, S. (2018). *An R companion to applied regression*. Sage publications.

Franklin, J. (2001). *The science of conjecture: Evidence and probability before Pascal*. Johns Hopkins University Press.

Montero Muñoz, J., Solla Suárez, P. E., & Gutiérrez Rodríguez, J. (2012). Estimación del filtrado glomerular en el paciente anciano. Implicaciones clínicas en el uso de antibióticos.
Revista Española de Geriatría y Gerontología, 56(5), 268-271-.