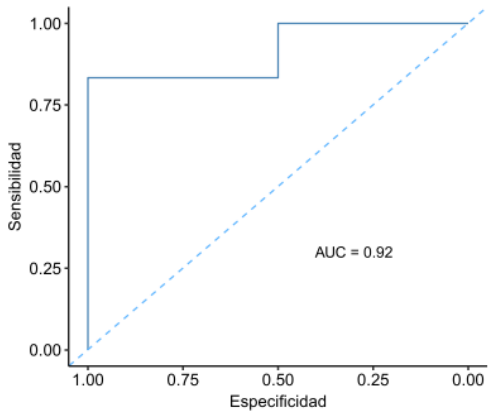



```
Evaluación del modelo (cjto. de prueba):
----------------------------------------
            Reference
Prediction   manual automático
  manual          1          1
  automático      1          5

   Exactitud: 0.750
 Sensibilidad: 0.833
Especificidad: 0.500
```

Figura 15.5: evaluación del clasificador con datos de prueba, no utilizados al construir el modelo.

En capítulos anteriores conocimos la validación cruzada como herramienta para conseguir una estimación más confiable y robusta del rendimiento de un modelo, la cual podemos usar de manera análoga para regresión logística.

El script 15.4 mejora la evaluación realizada en el script 15.3 incorporando el uso de validación cruzada de 4 pliegues (líneas 12–16). Notemos que la llamada a la función `train()` también solicita que "se guarden" los valores predichos, lo que nos permite calcular el rendimiento promedio del modelo, como si se repitiera la evaluación del script 15.3, seleccionando aleatoriamente un conjunto de entrenamiento y otro de prueba, cuatro veces. Los coeficientes del modelo que se consigue con este procedimiento se muestran en pantalla en las líneas 19–22 del script.

Las líneas 25–26 del script recuperan las predicciones hechas en cada pliegue. Luego se obtienen las respectivas matrices de confusión y métricas de evaluación para cada uno, utilizando la función `confusionMatrix()`, en las líneas 29–30. Como esta función devuelve muchas métricas, las líneas 33–36 extraen las que nos interesan en este caso: exactitud, sensibilidad y especificidad. Estas se ordenan en una matriz de datos, y se agregan las medias y desviaciones estándar de cada métrica en las lineas 39–42. En las líneas 45–47, esta tabla con valores numéricos se lleva a una tabla de strings para su presentación en pantalla, lograda en las líneas 51–58.

El resultado del script 15.4 puede verse en la figura 15.6. Debemos fijarnos en que el modelo obtenido, es idéntico al anterior (script 15.1). Esto se debe a que la función `train()` **recalcula** el modelo del pliegue que obtuvo mejor rendimiento con **todos los datos disponibles**. En el caso de la RLog (como con la lineal), los pliegues solo se diferencian en los datos que utilizan, **manteniendo** los predictores, por lo que siempre se llega al mismo modelo. Esto no sería así si la validación cruzada se usara, por ejemplo, para seleccionar las variables predictoras a incluir en el modelo, variando el conjunto de predictores usado en cada pliegue.

```
Coeficientes del modelo final:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -11.752      4.682  -2.510  0.01207
wt             3.935      1.487   2.646  0.00815

Detalle por pliegue:
 Exactitud  Sensibilidad  Especificidad  Pliegue
-------------------------------------------------
    1.000         1.000          1.000    Fold1
    0.857         0.667          1.000    Fold2
    0.667         0.500          0.750    Fold3
    1.000         1.000          1.000    Fold4
-------------------------------------------------
    0.881         0.792          0.938    Media
    0.158         0.250          0.125    D.E.
```

Figura 15.6: evaluación de un modelo de RLog usando validación cruzada.

Script 15.4: ajuste y evaluación de un modelo de regresión logística usando validación cruzada.

```
library(caret)
library(dplyr)
library(purrr)

# Cargar y filtrar los datos, teniendo cuidado de dejar "automático" como el
# 2do nivel de la variable "am" para que sea considerada como la clase positiva.
datos <- mtcars |> filter(wt > 2 & wt < 5) |>
  mutate(am = factor(am, levels = c(1, 0), labels = c("manual", "automático")))

# Ajustar modelo usando validación cruzada de 4 pliegues, asegurando que
# se guardan las predicciones de cada pliegue.
set.seed(113)
modelo_ent <- train(am ~ wt, data = datos, method = "glm",
                    family = binomial(link = "logit"),
                    trControl = trainControl(method = "cv", number = 4,
                                             savePredictions = TRUE))

# Mostrar los coeficientes del modelo obtenido
modelo_final <- modelo_ent[["finalModel"]]
modelo_final_str <- capture.output(print(summary(modelo_final), signif.stars = FALSE))
cat("Coeficientes del modelo final:\n")
write.table(modelo_final_str[6:9], quote = FALSE, row.names = FALSE, col.names = FALSE)

# Obtener las predicciones por pliegue
preds <- modelo_ent[["pred"]] |> mutate(pliegue = factor(Resample)) |>
  select(pred, obs, pliegue)

# Construir las matrices de confusión de cada pliegue
conf_mat_list <- preds |> group_split(pliegue) |>
  map(~ confusionMatrix(.x[["pred"]], .x[["obs"]]))

# Extraer las métricas de evaluación de interés
metricas_tab <- conf_mat_list |>
  map_df(~ data.frame(Exactitud = .$overall["Accuracy"],
                      Sensibilidad = .$byClass["Sensitivity"],
                      Especificidad = .$byClass["Specificity"]))

# Agregar el pliegue, los promedios y las desviaciones estándar
metricas_tab <- cbind(metricas_tab, Pliegue = levels(preds[["pliegue"]]))
medias_tab <- data.frame(t(apply(metricas_tab[, -4], 2, mean)), Pliegue = "Media")
desv_tab <- data.frame(t(apply(metricas_tab[, -4], 2, sd)), Pliegue = "D.E. ")
metricas_tab <- rbind(metricas_tab, medias_tab, desv_tab)

# Formatear las columnas
formatea_col <- function(cn) format(metricas_tab[[cn]], digits = 3,
                                    width = nchar(cn), justify = "right")
metricas_str_tab <- sapply(colnames(metricas_tab), formatea_col,
                           USE.NAMES = FALSE, simplify = TRUE)

# Mostrar las métricas obtenidas
encab <- paste(colnames(metricas_tab), collapse = "  ")
cat("Detalle por pliegue:\n", encab, "\n")
cat(strrep("-", nchar(encab)), "\n")
write.table(metricas_str_tab[1:4, ], sep = "  ",
            row.names = FALSE, col.names = FALSE, quote = FALSE)
cat(strrep("-", nchar(encab)), "\n")
write.table(metricas_str_tab[5:6, ], sep = "  ",
            row.names = FALSE, col.names = FALSE, quote = FALSE)
```

Analizando los resultados, se ve que en dos de los pliegues se obtuvo clasificación perfecta, mientras que los otros el rendimiento fue más parecido a lo observado en el modelo anterior (figura 15.5). Pero en promedio, repitiendo el experimento cuatro veces, el rendimiento del modelo es mayor que lo estimado anteriormente, por lo que con una única mirada lo estábamos subestimando.

15.5 MÚLTIPLES PREDICTORES

Cuando tenemos múltiples predictores potenciales, debemos decidir cuáles de ellos incorporar en el modelo de RLog. Una vez más, y tal como detallamos en el capítulo 14, el ideal es usar la regresión jerárquica para escoger los predictores de acuerdo a evidencia disponible en la literatura.

Sin embargo, si la intención es explorar los datos, podemos emplear los demás métodos ya descritos: selección hacia adelante, eliminación hacia atrás, regresión escalonada o todos los subconjuntos, usando para ello las mismas funciones de R descritas en el capítulo 14.

En páginas previas ajustamos un modelo de RLog para determinar el tipo de transmisión de un automóvil a partir de su peso. Sin embargo, el predictor fue seleccionado de manera arbitraria, simplemente para ilustrar el proceso, por lo que podríamos encontrar un mejor modelo usando algún método de selección de predictores.

Las líneas 22–23 del script 15.5 llevan a cabo esta tarea usando regresión escalonada, que busca golosamente el mejor modelo entre uno sin ningún predictor (el modelo nulo) hasta uno que considera todos los posibles predictores (el modelo completo). El resultado del script es presentado en la figura 15.7. En ella podemos ver que, además del peso del automóvil (`wt`), el mejor modelo encontrado considera además la potencia del motor (`hp`) como predictor.

Sin embargo, debemos notar que aparecen varios mensajes de advertencia indicando que se encontraron **problemas de convergencia** al realizar la búsqueda de este modelo. Es más, si nos fijamos en la evaluación de las estimaciones de los coeficientes, vemos que estos tienen ¡estadísticos gigantes y valores p igual a 1! Una clara indicación de que el modelo tiene problemas.

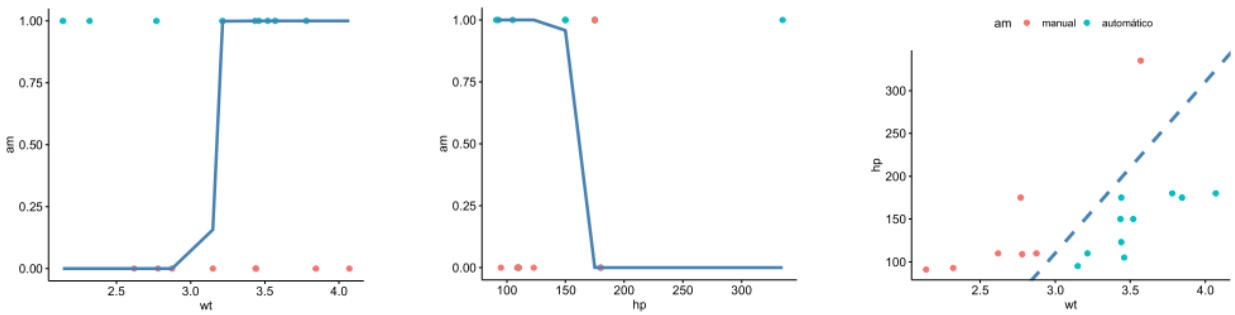
```
Modelo RLog conseguido con regresión escalonada:
------------------------------------------------

Call:
glm(formula = am ~ wt + hp, family = binomial(link = "logit"),
    data = datos_ent)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.730e+02  6.464e+05  -0.001        1
wt           1.409e+02  2.437e+05   0.001        1
hp          -4.994e-01  1.960e+03   0.000        1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2.3035e+01  on 16  degrees of freedom
Residual deviance: 4.6315e-10  on 14  degrees of freedom
AIC: 6

Number of Fisher Scoring iterations: 25

Warning messages:
1: glm.fit: fitted probabilities numerically 0 or 1 occurred
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
3: glm.fit: algorithm did not converge
4: glm.fit: fitted probabilities numerically 0 or 1 occurred
5: glm.fit: fitted probabilities numerically 0 or 1 occurred
6: glm.fit: fitted probabilities numerically 0 or 1 occurred
7: glm.fit: fitted probabilities numerically 0 or 1 occurred
8: glm.fit: fitted probabilities numerically 0 or 1 occurred
9: glm.fit: fitted probabilities numerically 0 or 1 occurred
10: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Figura 15.7: salida obtenida al buscar un modelo de RLog con regresión escalonada.

Script 15.5: búsqueda de un modelo de RLog usando regresión escalonada.

```
library(ggpubr)
library(dplyr)

# Cargar y filtrar los datos (solo predictores numéricos)
datos <- mtcars |> filter(wt > 2 & wt < 5) |>
  select(-c("cyl", "vs", "gear", "carb")) |>
  mutate(am = factor(am, levels = c(1, 0), labels = c("manual", "automático")))

# Separar los conjuntos de entrenamiento y prueba
set.seed(101)
n <- nrow(datos)
i_muestra <- sample.int(n = n, size = floor(0.7 * n), replace = FALSE)
datos_ent <- datos[i_muestra, ]
datos_pru  <- datos[-i_muestra, ]

# Construir los Modelos nulo y completo
nulo <- glm(am ~ 1, family = binomial(link = "logit"), data = datos_ent)
comp <- glm(am ~ ., family = binomial(link = "logit"), data = datos_ent)

# Ajustar y mostrar un modelo con regresión paso a paso escalonada
modelo <- step(nulo, scope = list(upper = comp),
               direction = "both", trace = FALSE)

cat("Modelo RLog conseguido con regresión escalonada:\n")
cat("-----------------------------------------------\n")
print(summary(modelo))
```

En un comienzo no es fácil determinar la raíz de los problemas. Si se grafican cómo serían los modelos de RLog asociados a cada predictor suponiendo constante el otro, obtenemos los gráficos (a) y (b) mostrados en la figura 15.8. Nada extraño es visible en ellos.



(a) `am ~ wt` (b) `am ~ hp` (c) `wt` vs. `hp`

Figura 15.8: gráficos de las predicciones parciales de los predictores peso (a) y potencia (b) y una visión de las clases en el gráfico de dispersión formado por estas variables (c).

Sin embargo, cuando vemos cómo interactúan estas variables en el gráfico (c), podemos darnos cuenta que una simple recta puede hacer una **separación perfecta** de los vehículos con transmisión manual de los que tienen transmisión automática. Es decir, ¡no necesitamos un clasificador logístico! El algoritmo de optimización iterativo descrito en la sección 15.3 no puede converger a coeficientes estables este caso.

El script 15.6 muestra una regresión paso a paso hacia adelante, de manera que podamos ir evitando caer en este problema de separación perfecta. Primero se construyen los modelos nulo y completo para ser usados, respectivamente, como modelo inicial y límite superior de la búsqueda. Notemos que el ajuste del modelo completo genera la primera advertencia de problemas de convergencia, como se aprecia en la figura 15.9 muestra la salida que genera esta búsqueda en pantalla.

Script 15.6: búsqueda de un modelo de RLog usando regresión paso a paso hacia adelante.

```
library(ggpubr)
library(dplyr)

# Imprimir mensajes de advertencia a medida que ocurren, más cortos
opt <- options(warn = 1, width = 26)

# Cargar y filtrar los datos (solo predictores numéricos)
datos <- mtcars |> filter(wt > 2 & wt < 5) |>
  select(-c("cyl", "vs", "gear", "carb")) |>
  mutate(am = factor(am, levels = c(1, 0), labels = c("manual", "automático")))

# Separar conjuntos de entrenamiento y prueba
set.seed(101)
n <- nrow(datos)
i_muestra <- sample.int(n = n, size = floor(0.7 * n), replace = FALSE)
datos_ent <- datos[i_muestra, ]
datos_pru  <- datos[-i_muestra, ]

# Definir modelos inicial y máximo
nulo <- glm(am ~ 1, family = binomial(link = "logit"), data = datos_ent)
maxi <- glm(am ~ ., family = binomial(link = "logit"), data = datos_ent)

# Revisar un paso hacia adelante
cat("\nPaso 1:\n------\n")
print(add1(nulo, scope = maxi))

# Actualizar el modelo
modelo1 <- update(nulo, . ~ . + wt)

# Revisar un paso hacia adelante
cat("\nPaso 2:\n------\n")
print(add1(modelo1, scope = maxi))

# Actualizar el modelo
modelo2 <- update(modelo1, . ~ . + mpg)

# Revisar un paso hacia adela
cat("\nPaso 3:\n------\n")
print(add1(modelo2, scope = maxi))

# Reestabler la opción para warnings y ancho de la pantalla
options(warn = opt[[1]], width = opt[[2]])

# Mostrar el modelo obtenido
modelo2_str <- capture.output(print(summary(modelo2)))
cat("\nModelo RLog conseguido con regresión hacia adelante:\n")
cat("-----------------------------------------------------\n")
write.table(modelo2_str[6:10], quote = FALSE, row.names = FALSE, col.names = FALSE)

# Comparar los modelos generados
cat("\nComparación de los modelos considerados:\n")
cat("----------------------------------------\n")
print(anova(nulo, modelo1, modelo2, test = "LRT"))
```

A partir del modelo nulo, en el primer paso (columna izquierda) vemos que la mayor reducción en AIC se obtiene agregando el peso del vehículo como predictor. La línea 29 del script actualiza el modelo en concordancia. En el segundo paso (columna central) aparecen varias advertencias sobre problemas de convergencia. La función `add1()` reporta desviación nula si agregamos como predictores las variables `hp` o `qsec`. Ya sabíamos de la primera, ahora sabemos que tampoco podemos usar la segunda. Además, debemos observar que la única reducción del AIC se consigue agregando el rendimiento del vehículo (`mpg`) como predictor. La línea 37 del script 15.6 realiza esta actualización. El siguiente paso (columna derecha de la figura 15.9) genera más advertencias de problemas de convergencia. Vemos que la mejor opción es detener la búsqueda y quedarnos con el modelo que incluye dos predictores.

```
Warning: glm.fit: fitted
probabilities numerically
 0 or 1 occurred
Paso 1:
------
Single term additions

Model:
am ~ 1
       Df Deviance    AIC
<none>     23.0348 25.035
mpg     1  18.9591 22.959
disp    1  15.1710 19.171
hp      1  23.0306 27.031
drat    1  12.5165 16.517
wt      1   9.9658 13.966
qsec    1  18.4951 22.495
```

```
Paso 2:
------
Warning: glm.fit: fitted
probabilities numerically
Warning: glm.fit: algorithm
did not converge
Warning: glm.fit: fitted
probabilities numerically
Single term additions

Model:
am ~ wt
       Df Deviance    AIC
<none>      9.9658 13.966
mpg     1   6.2682 12.268
disp    1   9.7704 15.770
hp      1   0.0000  6.000
drat    1   9.2023 15.202
qsec    1   0.0000  6.000
```

```
Paso 3:
------
Warning: glm.fit: fitted
probabilities numerically
Warning: glm.fit: algorithm
did not converge
Warning: glm.fit: fitted
probabilities numerically
Warning: glm.fit: algorithm
did not converge
Warning: glm.fit: fitted
probabilities numerically
Single term additions

Model:
am ~ wt + mpg
       Df Deviance    AIC
<none>      6.2682 12.268
disp    1   6.0578 14.058
hp      1   0.0000  8.000
drat    1   0.0000  8.000
qsec    1   0.0000  8.000
```

Figura 15.9: salida obtenida al construir un modelo de RLog manualmente con regresión hacia adelante.

La figura 15.10 presenta la salida a pantalla que genera la segunda parte del script 15.6. Vemos que, tanto al evaluar los coeficientes del modelo obtenido como al comparar el este modelo con los precedentes, pareciera que la inclusión del predictor `mpg` no hace un aporte importante al ajuste del modelo, y lo mejor sería evaluar su eliminación. Sin embargo, con lo pequeña que es esta muestra, no es tan aconsejable tomar decisiones cuando la evidencia está cerca del borde.

```
Modelo RLog conseguido con regresión hacia adelante:
---------------------------------------------------
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -66.4664    39.3397  -1.690   0.0911 .
wt           14.4710     8.0062   1.807   0.0707 .
mpg           1.0849     0.8049   1.348   0.1777

Comparación de los modelos considerados:
---------------------------------------
Analysis of Deviance Table

Model 1: am ~ 1
Model 2: am ~ wt
Model 3: am ~ wt + mpg
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1        16    23.0348
2        15     9.9658  1  13.0690 0.0003002 ***
3        14     6.2682  1   3.6976 0.0544915 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figura 15.10: modelo de RLog que resulta de aplicar manualmente regresión hacia adelante y su comparación con otros modelos evaluados durante la búsqueda.

15.6 CONFIABILIDAD DE UN MODELO DE RLOG

Aprovechando las dificultades encontradas con el ejemplo, es importante recordar que no basta con realizar una búsqueda automática de predictores y evaluar el desempeño de un clasificador, sino que también es

esencial verificar el cumplimiento de las condiciones necesarias para que un modelo de regresión logística sea válido:

1. Debe existir una relación lineal entre los predictores y la respuesta transformada.
2. Los residuos deben ser independientes entre sí.

Si bien son menos condiciones que las vistas para la regresión lineal, debemos además agregar situaciones en que puede ocurrir que el método de optimización no converja:

3. Multicolinealidad entre los predictores, que en este caso se evalúa y aborda del mismo modo que para RLM (por ejemplo, mediante el factor de inflación de la varianza o la tolerancia).
4. Información incompleta, que se produce cuando no contamos con observaciones suficientes para todas las posibles combinaciones de predictores, en especial para algún nivel de una variable categórica.
5. Separación perfecta, que ocurre cuando no hay superposición entre las clases (es decir, como vimos, cuando los predictores separan ambas clases completamente).

Y finalmente, también debemos descartar la presencia de casos problemáticos:

6. Las estimaciones de los coeficientes del modelo no están dominadas por casos influyentes.

Para verificar la condición 1 podemos usar los gráficos de residuos vistos para la RLM, provistos por la función `residualPlots()` del paquete `car`, aunque el gráfico de residuos versus los valores predichos pierde sentido en la RLog (y en general en los modelos de RLG), por lo que agregamos el argumento `fitted = FALSE`. El mismo paquete también permite revisar linealidad usando gráficos de los **residuos parciales**, es decir, cuánto contribuye a los residuos cada uno de los predictores, por medio de la función `crPlots(modelo)`.

Estos gráficos aplicados al ejemplo del modelo con dos predictores de la figura 15.10 se muestran, respectivamente, en la parte superior e inferior de la figura 15.11.

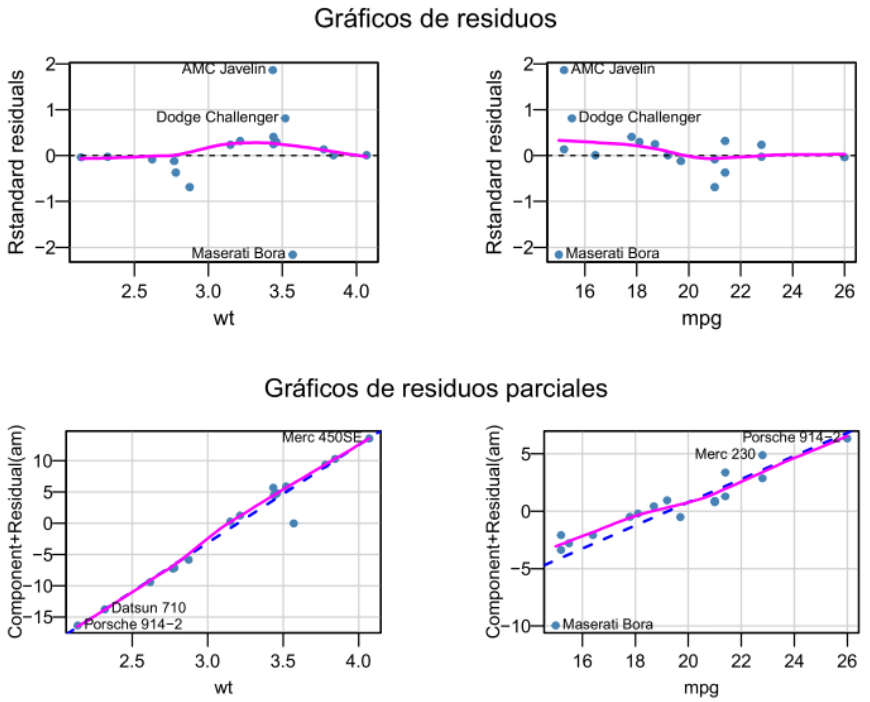


Figura 15.11: gráficos de residuos y gráficos de residuos parciales para evaluar la condición de linealidad en el modelo de RLog ejemplo.

En ambos gráficos, observamos que el supuesto de linealidad para cumplirse relativamente bien. De hecho, el resultado de las pruebas de curvatura que también realiza y reporta la función `residualPlots()` confirman esta idea:

```
    Test stat Pr(>|Test stat|)
wt     1.3077           0.2528
mpg    0.3709           0.5425
```

Pasando a la segunda condición, nuevamente podemos utilizar lo hecho para RLM y aplicar la prueba de Durbin-Watson implementada en la función `durbinWatsonTest` del paquete `car`. Para el ejemplo se obtiene:

```
 lag Autocorrelation D-W Statistic p-value
   1      -0.1794747      2.339983   0.532
 Alternative hypothesis: rho != 0
```

Con esto descartamos que exista evidencia para sospechar que no se esté cumpliendo la condición de independencia de los residuos.

Pasemos entonces a revisar que no existe no exista multicolinealidad fuerte, utilizando la ya conocida función `vif(modelo)` para obtener los factores de inflación de la varianza. Para el modelo ejemplo se consigue:

```
      wt      mpg
3.846939 3.846939
```

Vemos que reporta valores iguales (VIF = 3,847) para ambos predictores, lo que es esperable en modelos con dos predictores continuos sin interacción. Si bien son un poco altos, pues hay autores que sugieren que valores superiores a 3 pueden ser problemáticos, no sobrepasan el umbral más preocupante fijado en 5 ni el umbral crítico de 10. Si bien se mantienen estos umbrales, debemos notar que los valores de VIF provienen de modelos lineales auxiliares, y no directamente del modelo de regresión logística, por lo que debemos ser especialmente cuidadosos.

La siguiente condición es problemática para construir **cualquier modelo**. Recordemos que cuando estudiamos la RLM, se dijo que, si bien **no** existe un criterio único del tamaño de la muestra requerido, se ha recomendado contar con 10 o 15 observaciones por cada predictor numérico y nivel de las variables categóricas. Esto no suele ser simple, puesto que, por ejemplo, la variable asociada al número de carburadores presentes (`carb`) tiene 4 niveles, por lo que si se considera este potencial predictor, se necesitarían más de 50 datos distribuidos más o menos balanceadamente en estos niveles. Además, habría que considerar que estas observaciones, en cada nivel, cubriera el rango de valores de cada predictor numérico. En el ejemplo, tenemos dos predictores numéricos y, si bien los casos en la muestra se distribuyen bien a lo largo de sus valores, solo tenemos 17 observaciones, muy pocas como para conseguir un modelo confiable.

Por último, queda revisar la condición relacionada a casos influyentes. Cuando revisamos los residuos parciales respecto de la variable `car` (gráfico inferior derecho de la figura 15.11), debimos notar que la recta ajustada (línea punteada en color azul-acero) esta algo desviada de la curva de ajuste local de los datos (línea sólida en tono rojizo). Esto parece deberse por la influencia de un caso muy alejado hacia los valores negativos, correspondiente al (ya sospechoso de siempre) modelo Maserati Bora. Sin embargo, esta influencia no se observa en los gráficos de residuos, tal vez moderada por la presencia de otro caso atípico, pero positivo, correspondiente al modelo AMC Javelin.

Como con los diagnósticos anteriores, aquí también podemos aplicar las herramientas usadas para RLM, como el gráfico burbuja con tres medidas de influencia para el modelo que se obtiene con la función `influencePlot(modelo)` del paquete `car`, que para el ejemplo se presenta en la figura 15.12. Recordando que esta función también reporta los estadísticos para casos notorios, nos fijamos en lo que muestra en pantalla:

```
                StudRes       Hat      CookD
AMC Javelin    1.7553843 0.5185520 0.97532006
Mazda RX4 Wag -0.6005575 0.5005603 0.08357138
Maserati Bora -2.2497803 0.2826453 0.79483991
```

Vemos tres observaciones que sobrepasan dos veces el apalancamiento promedio (0,353 aproximadamente para el modelo de RLog del ejemplo) y que dos de ellos también presentan distancias de Cook altas (sobre el umbral $4/17 = 0{,}235$) pero que no sobrepasan el valor crítico de 1 (aunque el modelo AMC Javelin está muy cerca de hacerlo). El modelo Maserati Bora también constituye claramente un valor atípico.

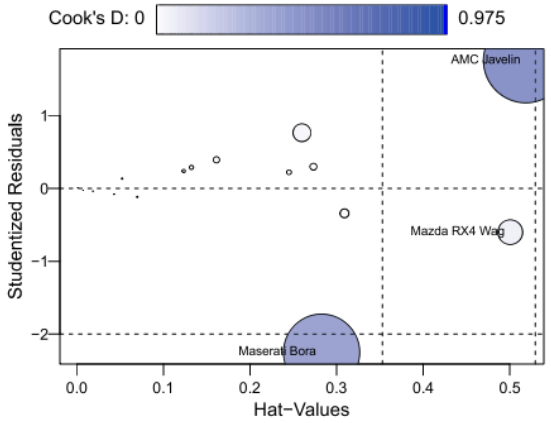


Figura 15.12: gráfico de influencia para el modelo de RLog del ejemplo.

Como hemos dicho anteriormente, correspondería hacer un análisis riguroso de los datos para decidir si eliminar estas observaciones problemáticas sin violar los principios de la ética profesional de un analista de datos. Si así se decidiera, se debe reconstruir (rehacer la búsqueda de predictores y coeficientes) y reevaluar el modelo que se consiga sin datos influyentes. Y si vuelven a aparecer otros casos con demasiada influencia, se debe iterar nuevamente en la búsqueda de un modelo confiable. En este ejemplo, el problema puede deberse al tamaño insuficiente de la muestra de observaciones, por lo que si el estudio fuera nuestro, los esfuerzos deberían concentrarse en extender este conjunto.

15.7 EJERCICIOS PROPUESTOS

15.1 Investiga cómo solicitar a la función `train()` del paquete `caret` que use una métrica distinta a la exactitud en su búsqueda de un modelo de regresión logística, y adapta el script 15.4 para evaluar el poder predictivo de un modelo que optimice el AUC.

15.2 Explica con mayor detalle el código presentado en el script 15.4 y adáptalo para conseguir una evaluación con 20 repeticiones de validación cruzada de 4 pliegues.

15.3 Evalúa la calidad predictiva del clasificador mostrado en la figura 15.10 en datos no vistos y compara el desempeño con los datos de entrenamiento. ¿Hay indicios de sobreajuste?.

15.4 Evalúa la calidad predictiva del clasificador mostrado en la figura 15.10 usando 20 repeticiones de validación cruzada de 4 pliegues con los datos de entrenamiento y compáralo con el desempeño en el conjunto de prueba. ¿Hay indicios de sobreajuste?.

15.5 Evalúa la calidad predictiva del clasificador mostrado en la figura 15.10 usando validación cruzada dejando uno fuera con los datos de entrenamiento y compáralo con el desempeño en el conjunto de prueba. ¿Hay indicios de sobreajuste?.

15.6 Investiga cómo se puede indicar a la función `train()` que utilice bootstrapping en la búsqueda de un modelo. Evalúa la calidad predictiva del clasificador con dos predictores de la figura 15.10 usando 1.000 repeticiones de bootstrapping. ¿Cómo evalúas la generalidad del modelo?.

15.7 Grafica la ROC asociada al clasificador de la figura 15.10 en los datos de entrenamiento. Determina el AUC y el umbral óptimo. Con este último, aplica el modelo a los datos de prueba y agrega al gráfico la ROC correspondiente. ¿Cómo se comparan los desempeños?

15.8 Investiga cómo usar la función `roc.test()` del paquete `pROC` y verifica si la curva ROC generada por el clasificador de la figura 15.10 para los datos de entrenamiento es significativamente mejor que la generada para los datos de prueba.