Regresión logística simple y múltiple

Ejemplo de solución ejercicio prático N°10

Enunciado

Para este ejercicio usaremos los datos de medidas anatómicas recolectados por Heinz et al. (2003) que ya conocimos en el ejercicio práctico anterior (disponibles en el archivo "EP09 Datos.csv"). Como en este case se requiere de una variable dicotómica, vamos a realizar lo siguiente:

- **Crear la variable dicotómica `TRG` (¿tiene rodillas gruesas?) con valor "sí" cuando los diámetros de las rodillas sobrepasan los $19,0$ cm y "no" en caso contrario.**

Se pide construir un modelo de regresión logística para predecir la variable `TRG`, de acuerdo con las siguientes instrucciones:

1. **Definir la semilla a utilizar, que corresponde a los últimos cuatro dígitos del RUN (sin considerar el dígito verificador) del integrante de mayor edad del equipo.**
2. **Seleccionar una muestra de 150 mujeres (si la semilla es un número par) o 150 hombres (si la semilla es impar), asegurando que la mitad tenga rodillas gruesas y la otra mitad no. Dividir esta muestra en dos conjuntos: los datos de 100 personas (50 de ellas con rodillas gruesas) para utilizar en la construcción de los modelos y 50 personas (25 de ellas con rodillas gruesas) para poder evaluarlos.**
3. **Recordar las ocho posibles variables predictoras seleccionadas de forma aleatoria en el ejercicio anterior.**
4. **Seleccionar, de las otras variables, una que el equipo considere que podría ser útil para predecir la clase `TRG`, justificando bien esta selección (idealmente con literatura).**
5. **Usando el entorno R (pero no del paquete `caret`), construir un modelo de regresión logística con el predictor seleccionado en el paso anterior y utilizando la muestra obtenida.**
6. **Usando herramientas para la exploración de modelos del entorno R (pero no del paquete `caret`), buscar entre dos y cinco predictores de entre las variables seleccionadas al azar, recordadas en el punto 3, para agregar al modelo obtenido en el paso 5.**
7. **Evaluar la confiabilidad de los modelos (i.e. que tengan un buen nivel de ajuste y son generalizables) y "arreglarlos" en caso de que tengan algún problema.**
8. **Usando herramientas del entorno R (pero no del paquete `caret`), evaluar el poder predictivo de los modelos, con los datos que no se incluyeron en su construcción, en términos de sensibilidad y especificidad.**

Comencemos Incluyendo los paquetes que usaremos en este script.

```
library(car)
library(dplyr)
library(ggpubr)
library(gridExtra)
library(leaps)
library(tidyr)
```

Obtengamos los datos en formato ancho.

```
set.seed(1111)
src_dir <- "~/Downloads"
src_basename <- "EP09 Datos.csv"
src_file <- file.path(src_dir, src_basename)

datos <- read.csv2(file = src_file, stringsAsFactors = TRUE)
```

Generemos las variables nuevas requeridas para este ejercicio.

```
datos_ext <- datos |>
  mutate(TRG = ifelse(Knees.diameter < 19.0, "no", "sí"))
datos_ext[["Gender"]] <- factor(datos_ext[["Gender"]])
datos_ext[["TRG"]] <- factor(datos_ext[["TRG"]])
```

Obtenemos la muestra como indican las instrucciones 1 y 2, teniendo cuidado de *desordenar* los conjuntos de datos para que no queden juntos todos los casos con la misma clase, puesto que introduce artificialmente dependencia entre los datos.

```
muestra_a <- datos_ext |> filter(Gender == 1 & TRG == "no") |>
  sample_n(75, replace = FALSE)
muestra_b <- datos_ext |> filter(Gender == 1 & TRG == "sí") |>
  sample_n(75, replace = FALSE)

i_train <- sample(1:75, 50)
muestra_train <- rbind(muestra_a[i_train, ], muestra_b[i_train, ]) |>
  select(-Gender) |> sample_frac(1L)
muestra_test <- rbind(muestra_a[-i_train, ], muestra_b[-i_train, ]) |>
  select(-Gender) |> sample_frac(1L)
```

Verificamos que no cometimos algún error con las muestras

```
stopifnot(all(muestra_train$Id == unique(muestra_train$Id)))
stopifnot(all(muestra_test$Id == unique(muestra_test$Id)))
stopifnot(!any(muestra_train$Id %in% muestra_test))
```

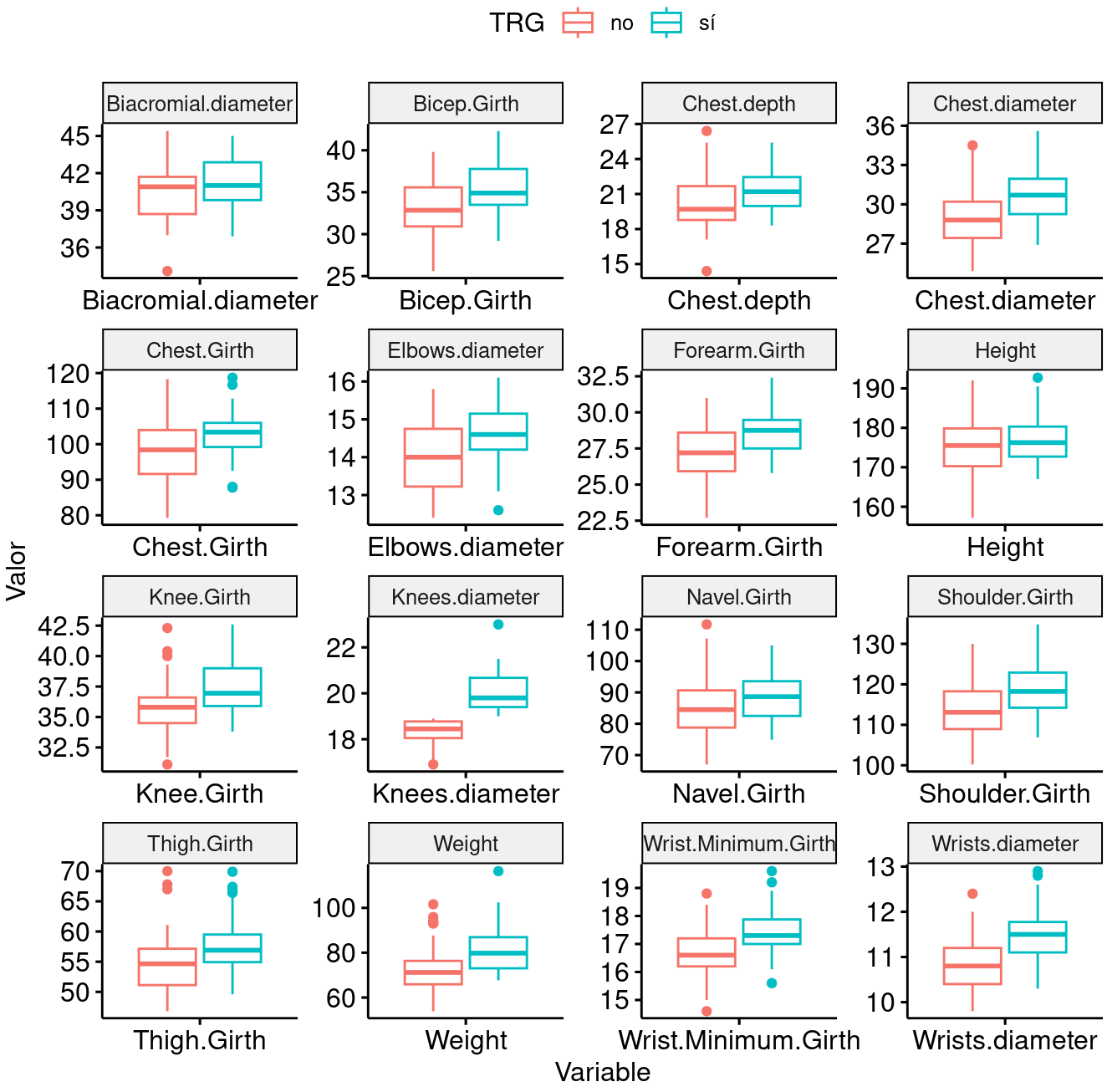
Siguiendo la instrucción 3, recordemos las ocho posibles variables predictoras seleccionadas de forma aleatoria en el ejercicio anterior.

```
nombre_respuesta <- "TRG"
predictores <- c("Ankles.diameter", "Calf.Maximum.Girth", "Waist.Girth", "Bitrochanteric.diamete
r",
                 "Ankle.Minimum.Girth", "Hip.Girth", "Biiliac.diameter", "Age")
```

Regresión logística simple

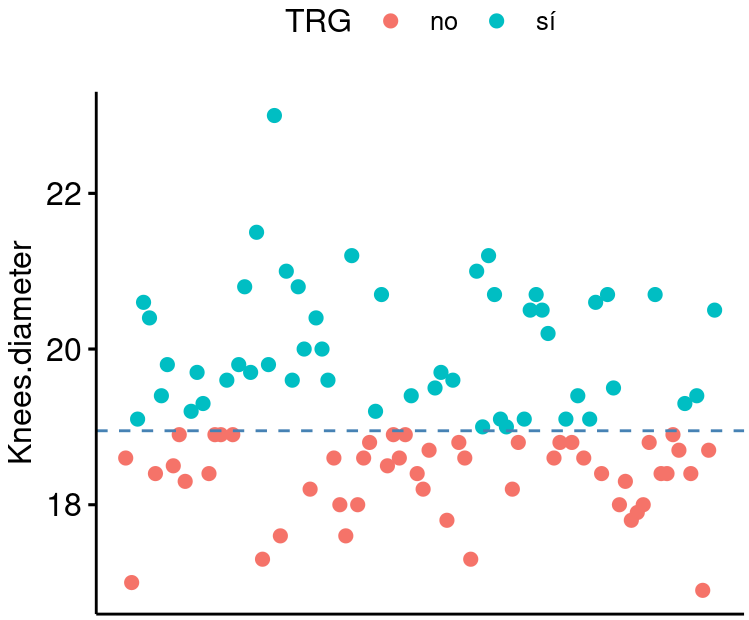
Corresponde seleccionar una de las otras variables (instrucción 4) que podría ser útil para predecir la variable respuesta. Para esto miremos cómo se relacionan las otras variables con la variable de respuesta, sin considerar la variable `Gender` que, por diseño, tiene solo un valor.

```
# Obtiene relaciones entre todos los pares de variables
otras <- colnames(muestra_train)[! colnames(muestra_train) %in% predictores]
p1_dfl <- muestra_train |> select(all_of(otras)) |>
  pivot_longer(-all_of(nombre_respuesta), names_to = "Variable", values_to = "Valor") |>
  mutate(Variable = factor(Variable))
p1 <- ggboxplot(p1_dfl, x = "Variable", y = "Valor", color = nombre_respuesta)
p1 <- p1 +  facet_wrap( ~ Variable, ncol = 4, scales = "free")
print(p1)
```



Por supuesto, la variable `Knees.diameter` es la que exhibe menor traslape entre las clases. Es más, no existe traslape para esta variable, por lo que nos permite clasificar los casos sin errores. Como vimos, esto presenta problemas si buscamos un modelo de regresión logística, ya que se trata de separación perfecta.

```
p2_dfl <- muestra_train |> select(Knees.diameter, TRG) |>
  mutate(Id = 1:n())
p2 <- ggscatter(p2_dfl, x = "Id", y = "Knees.diameter", color = nombre_respuesta)
p2 <- p2 + geom_hline(yintercept = 18.95, linetype = "dashed", color = "steelblue")
p2 <- p2 + theme(axis.title.x = element_blank(), axis.text.x = element_blank(),
                 axis.ticks.x = element_blank())
print(p2)
```



Veamos cómo falla la construcción del modelo.

```
rlogit_sep_perf <- glm(TRG ~ Knees.diameter, data = muestra_train,
                       family = binomial(link = "logit"))
```

```
Warning: glm.fit: algorithm did not converge
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

De este modo, tenemos que elegir otra variable para nuestro modelo de regresión logística simple (RLogitS). Mirando el gráfico de cajas, parece haber varias opciones: `Forearm.Girth`, `Knee.Girth`, `Shoulder.Girth`, `Weight`, `Wrist.Minimum.Girth`, y `Wrists.diameter` parecen tener niveles de solapamiento similares. Pero esta última variable parece tener la líneas de las medianas más separadas, por lo que la escogeremos para cumplir con la instrucción 5.

```
predictor <- "Wrists.diameter"
rlogits_fmla <- formula(paste(nombre_respuesta, predictor, sep = " ~ "))

rlogits <- glm(rlogits_fmla, data = muestra_train,
               family = binomial(link = "logit"))

cat("Modelo de regresión logística simple\n")
print(summary(rlogits))
```

```
Modelo de regresión logística simple

Call:
glm(formula = rlogits_fmla, family = binomial(link = "logit"),
    data = muestra_train)

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -18.2795     4.5749  -3.996 6.45e-05 ***
Wrists.diameter   1.6413     0.4107   3.996 6.44e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 138.63  on 99  degrees of freedom
Residual deviance: 117.48  on 98  degrees of freedom
AIC: 121.48

Number of Fisher Scoring iterations: 3
```

Regresión logística múltiple

Para cumplir con la instrucción 6, vamos a utilizar regresión escalonada hacia adelante.

```
add1(rlogits, scope = c(predictor, predictores))
```

```
Single term additions

Model:
TRG ~ Wrists.diameter
                        Df Deviance    AIC
<none>                       117.48 121.48
Wrists.diameter          0   117.48 121.48
Ankles.diameter          1   101.48 107.48
Calf.Maximum.Girth       1   111.17 117.17
Waist.Girth              1   117.46 123.46
Bitrochanteric.diameter  1   112.89 118.89
Ankle.Minimum.Girth      1   114.58 120.58
Hip.Girth                1   115.16 121.16
Biiliac.diameter         1   116.79 122.79
Age                      1   117.43 123.43
```

Podemos ver que la mejor opción es extender nuestro modelo simple es agregar la variable `Ankles.diameter` como predictor. Veamos el siguiente paso.

```
rlogitm <- update(rlogits, . ~ . + Ankles.diameter)

add1(rlogitm, scope = c(predictor, predictores))
```

```
Single term additions

Model:
TRG ~ Wrists.diameter + Ankles.diameter
                        Df Deviance    AIC
<none>                      101.478 107.48
Wrists.diameter          0  101.478 107.48
Ankles.diameter          0  101.478 107.48
Calf.Maximum.Girth       1   99.698 107.70
Waist.Girth              1  100.991 108.99
Bitrochanteric.diameter  1  100.294 108.29
Ankle.Minimum.Girth      1  101.399 109.40
Hip.Girth                1  101.293 109.29
Biiliac.diameter         1  101.472 109.47
Age                      1  100.855 108.86
```

En este paso podemos observar que la variable `Calf.Maximum.Girth` produce una leve disminución de la desviación, pero una pequeña alza en el AIC. Dado que se nos pide agregar al menos dos variables al modelo simple, la agregamos a los predictores del modelo.

```
rlogitm <- update(rlogitm, . ~ . + Calf.Maximum.Girth)

add1(rlogitm, scope = c(predictor, predictores))
```

```
Single term additions

Model:
TRG ~ Wrists.diameter + Ankles.diameter + Calf.Maximum.Girth
                        Df Deviance    AIC
<none>                      99.698 107.70
Wrists.diameter          0   99.698 107.70
Ankles.diameter          0   99.698 107.70
Calf.Maximum.Girth       0   99.698 107.70
Waist.Girth              1   97.693 107.69
Bitrochanteric.diameter  1   99.361 109.36
Ankle.Minimum.Girth      1   98.398 108.40
Hip.Girth                1   99.453 109.45
Biiliac.diameter         1   99.288 109.29
Age                      1   99.338 109.34
```

Ahora vemos que la variable `Waist.Girth` produce una pequeña baja en la desviación manteniendo el AIC casi intacto. Agreguémosla al modelo.

```
rlogitm <- update(rlogitm, . ~ . + Waist.Girth)

add1(rlogitm, scope = c(predictor, predictores))
```

```
Single term additions

Model:
TRG ~ Wrists.diameter + Ankles.diameter + Calf.Maximum.Girth +
    Waist.Girth
                       Df Deviance    AIC
<none>                      97.693 107.69
Wrists.diameter         0   97.693 107.69
Ankles.diameter         0   97.693 107.69
Calf.Maximum.Girth      0   97.693 107.69
Waist.Girth             0   97.693 107.69
Bitrochanteric.diameter 1   95.693 107.69
Ankle.Minimum.Girth     1   97.140 109.14
Hip.Girth               1   96.884 108.88
Biiliac.diameter        1   97.686 109.69
Age                     1   97.678 109.68
```

Ahora sucede algo similar con `Bitrochanteric.diameter`. Siguiendo el mismo criterio, la añadimos al modelo.

```
rlogitm <- update(rlogitm, . ~ . + Bitrochanteric.diameter)

add1(rlogitm, scope = c(predictor, predictores))
```

```
Single term additions

Model:
TRG ~ Wrists.diameter + Ankles.diameter + Calf.Maximum.Girth +
    Waist.Girth + Bitrochanteric.diameter
                       Df Deviance    AIC
<none>                      95.693 107.69
Wrists.diameter         0   95.693 107.69
Ankles.diameter         0   95.693 107.69
Calf.Maximum.Girth      0   95.693 107.69
Waist.Girth             0   95.693 107.69
Bitrochanteric.diameter 0   95.693 107.69
Ankle.Minimum.Girth     1   94.695 108.69
Hip.Girth               1   95.574 109.57
Biiliac.diameter        1   94.201 108.20
Age                     1   95.689 109.69
```

Vemos que ahora cualquier otro predictor del conjunto seleccionado al azar genera un aumento del AIC, por lo que detenemos la búsqueda. Veamos el modelo obtenido.

```
cat("Modelo de regresión logística múltiple con 5 predictores\n")
print(summary(rlogitm))
```

```
Modelo de regresión logística múltiple con 5 predictores

Call:
glm(formula = TRG ~ Wrists.diameter + Ankles.diameter + Calf.Maximum.Girth +
    Waist.Girth + Bitrochanteric.diameter, family = binomial(link = "logit"),
    data = muestra_train)

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -37.63732    8.19865  -4.591 4.42e-06 ***
Wrists.diameter           1.16015    0.52231   2.221  0.02634 *
Ankles.diameter           1.07774    0.36446   2.957  0.00311 **
Calf.Maximum.Girth        0.18931    0.13418   1.411  0.15828
Waist.Girth              -0.07325    0.03935  -1.861  0.06270 .
Bitrochanteric.diameter   0.25871    0.18852   1.372  0.16997
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 138.629  on 99  degrees of freedom
Residual deviance:  95.693  on 94  degrees of freedom
AIC: 107.69

Number of Fisher Scoring iterations: 5
```

Como era de esperarse, por las leves disminuciones en desviación, los últimos 3 predictores no aportan significativamente al modelo. Por el principio de parsimonia, deberíamos eliminar 2 de ellas para cumplir con el lo solicitado en el enunciado. Quitemos las últimas 2 variables agregadas.

```
rlogitm <- update(rlogitm, . ~ . - Waist.Girth - Bitrochanteric.diameter)

cat("Modelo de regresión logística múltiple con 3 predictores\n")
```

```
Modelo de regresión logística múltiple con 3 predictores
```

```
print(summary(rlogitm))
```

```
Call:
glm(formula = TRG ~ Wrists.diameter + Ankles.diameter + Calf.Maximum.Girth,
    family = binomial(link = "logit"), data = muestra_train)

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        -31.0752     6.6802  -4.652 3.29e-06 ***
Wrists.diameter      0.8920     0.4866   1.833  0.06675 .
Ankles.diameter      1.0832     0.3530   3.068  0.00215 **
Calf.Maximum.Girth   0.1483     0.1128   1.315  0.18839
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 138.629  on 99  degrees of freedom
Residual deviance:  99.698  on 96  degrees of freedom
AIC: 107.7

Number of Fisher Scoring iterations: 5
```

Confiabilidad de los modelos

Ajuste

Comencemos revisando la bondad de ajuste de los modelos.

```
rlogits_lrt <- anova(rlogits, test = "LRT")
rlogitm_lrt <- anova(rlogits, rlogitm, test = "LRT")

cat("Bondad de ajuste del modelo univariado:\n")
print(rlogits_lrt)
cat("\n")
cat("Bondad de ajuste del modelo multivariado:\n")
print(rlogitm_lrt)
```

```
Bondad de ajuste del modelo univariado:
Analysis of Deviance Table

Model: binomial, link: logit

Response: TRG

Terms added sequentially (first to last)

               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                                99     138.63
Wrists.diameter  1   21.145        98     117.48 4.258e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Bondad de ajuste del modelo multivariado:
Analysis of Deviance Table

Model 1: TRG ~ Wrists.diameter
Model 2: TRG ~ Wrists.diameter + Ankles.diameter + Calf.Maximum.Girth
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1        98    117.484
2        96     99.698  2   17.787 0.0001373 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Vemos que el modelo simple obtiene una reducción significativa de la devianza ($\chi^2(1) = 21{,}145$; $p < 0.001$) respecto del modelo nulo, y que el modelo múltiple logra reducir significativamente este estadístico respecto del modelo simple ($\chi^2(2) = 17{,}787$; $p < 0.001$). Bajo este criterio entonces, ambos modelos logran una buena bondad de ajsute.

Multicolinealidad

Aseguremos que esta falta de aporte no esté también introduciendo problemas de multicolinealidad.

```
cat("Factores de inflación de la varianza:\n")
print(vif(rlogitm))
cat("\n")
cat("Valores de tolerancia:\n")
print(1 / vif(rlogitm))
```

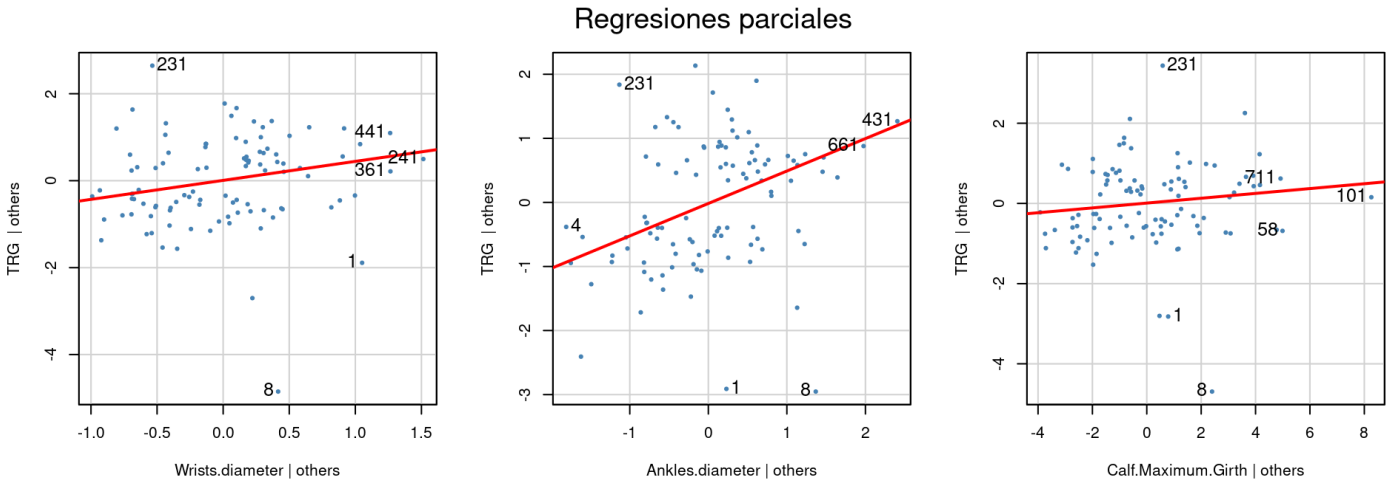
```
Factores de inflación de la varianza:
   Wrists.diameter    Ankles.diameter Calf.Maximum.Girth
          1.111214           1.094437           1.143815

Valores de tolerancia:
   Wrists.diameter    Ankles.diameter Calf.Maximum.Girth
         0.8999170          0.9137114          0.8742675
```

¡Fantástico! Podemos notar que todos los factores de inflación de la varianza están lejos del límite de 10 y ninguna tolerancia es menos a $0{,}2$, lo que indicaría que no hay presencia de multicolinealidad severa.

Relaciones lineales

Revisemos que se cumple la condición de relaciones lineales entre los predictores y la respuesta transformada, para lo que usaremos la función `avPlots()` del paquete `car`.

```
avPlots(rlogitm, layout = c(1, 3),
        col = "steelblue", pch = 20, col.lines = "red",
        main = "Regresiones parciales",
        id = list(n = 3, cex = 1.2, location = "lr"))
```

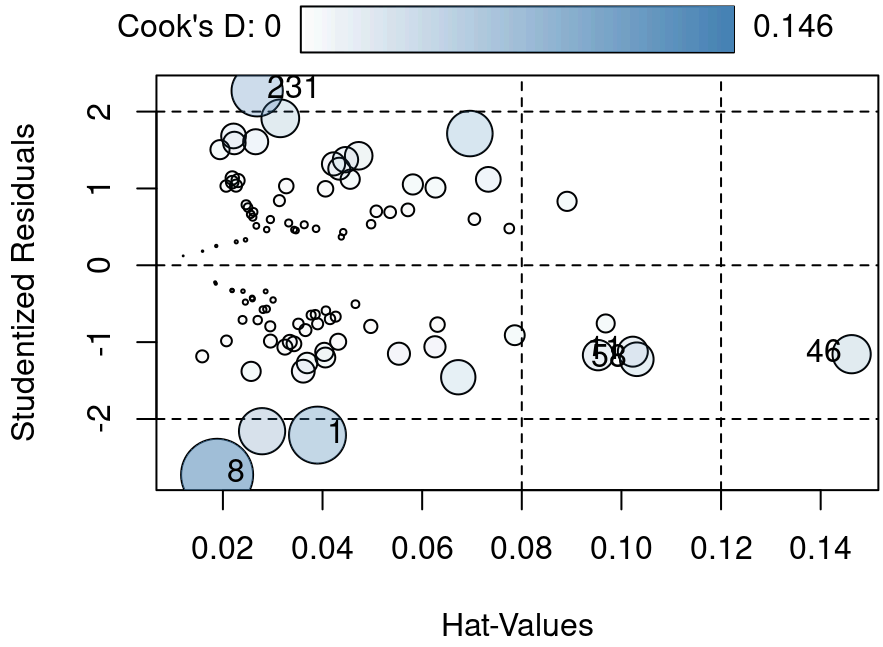


En estos gráficos podemos observar varias cosas interesantes. Primero, que las relaciones de la variable de salida con el diámetro de las muñecas (`Wrists.diameter`) y el grosor máximo de las pantorrillas (`Calf.Maximum.Girth`) parecen lineales, aunque con una pendiente bastante reducida. Por otro lado, la pendiente con el diámetro de los tobillos (`Ankles.diameter`) es más pronunciada, pero hay un comportamiento extraño de los residuos parciales que tienden a agruparse en dos nubes. La recta parece apalancada por los valores más extremos en esta variable, pues una línea prácticamente horizontal representaría mejor a la mayoría de los datos.

Casos sobre influyentes

Revisemos estas sospechas haciendo uso de la función `influencePlot()` provista por el paquete `car` que, recordemos, representa de forma gráfica tres métricas de influencia: residuos *studentizados* versus apalancamiento (*hat values*) y círculos cuyas áreas son proporcionales a la distancia de Cook.

```
rlogits_inf_estad <- influencePlot(rlogitm, fill.col = "steelblue",
                                   scale = 5, id = list(n = 3))
```




```
cat("Límites para el modelo de RLogitS:\n")
cat("Rango para 95% de los residuos studentizados: ")
cat("[", round(qt(0.05/2, nrow(muestra_train) - length(coef(rlogitm)) - 1), 3), ", ", sep = "")
cat(round(qt(1-0.05/2, nrow(muestra_train) - length(coef(rlogitm)) - 1), 3), "]\n", sep = "")
cat("Límite del apalancamiento:", round(2 * mean(hatvalues(rlogitm)), 3), "\n")
cat("Límite de la distancia de Cook:", round(3 * mean(cooks.distance(rlogitm)), 3), "\n")
cat("Casos notorios para el modelo de RLogitS:\n")
print(rlogits_inf_estad)
```

```
Límites para el modelo de RLogitS:
Rango para 95% de los residuos studentizados: [-1.985, 1.985]
Límite del apalancamiento: 0.08
Límite de la distancia de Cook: 0.033
Casos notorios para el modelo de RLogitS:
      StudRes        Hat      CookD
8   -2.726169 0.01883045 0.14601719
46  -1.156655 0.14621202 0.04110867
58  -1.222549 0.10308072 0.03184470
231  2.270636 0.02687771 0.07384157
1   -2.209810 0.03898680 0.09122136
11  -1.123885 0.10229049 0.02529247
```

Podemos observar que ninguno de los residuos destacados está fuera del rango seguro en todos los criterios. Tal vez el caso 1 podría considerarse algo problemático, pues exhibe una distancia de Cook muy superior (más de 4 veces) al del resto y tiene los valores más altos para los otros criterios (con un empate en apalancamiento con el caso 441). Pero en el gráfico de regresiones parciales se puede apreciar que este caso no parece realmente modificar la recta ajustada parcialmente a cada predictor, por lo que no parece que sea necesario sacarlo del ajuste. Por otro lado, los casos 431 y 661, que parecían preocupantes en la regresión parcial del diámetro de los tobillos, ni siquiera aparecen como preocupantes en términos de los criterios usados en la figura anterior.

Independencia de los residuos

Confirmemos que no existe dependencia entre los residuos generados por el modelo de RLogitS.

```
cat("Prueba de la independencia de los residuos para el modelo de RLogitS:\n")
print(durbinWatsonTest(rlogits))
```

```
Prueba de la independencia de los residuos para el modelo de RLogitS:
 lag Autocorrelation D-W Statistic p-value
   1      0.08844082      1.808878   0.278
 Alternative hypothesis: rho != 0
```

Vemos que no hay razones para sospechar que los residuos no sean independientes para este modelo.

Confirmemos que esto también se da para el modelo de RLogitM.

```
cat("Prueba de la independencia de los residuos para el modelo de RLogitM:\n")
print(durbinWatsonTest(rlogitm))
```

```
Prueba de la independencia de los residuos para el modelo de RLogitM:
 lag Autocorrelation D-W Statistic p-value
   1       0.1327261      1.725419   0.188
 Alternative hypothesis: rho != 0
```

¡Estupendo! No hay evidencia que nos indique falta de independencia de los residuos en este modelo tampoco.

Resultado

Concluimos que tanto el modelo de RLogitS como el de RLogitM son **relativamente confiables**, puesto que los predictores muestras asociaciones lineales con la variable de respuesta y no hay patrones visibles ni evidencia de dependencia entre los residuos. Tampoco se identificaron casos que estén ejerciendo demasiada influencia en el modelo, aunque hay dos o tres casos que podrían ser preocupantes.

Poder predictivo

La instrucción 8 nos pide evaluar la calidad predictiva de los modelos en términos de sensibilidad y especificidad (pero sin usar el paquete `caret`).

Comenzamos obteniendo las predicciones del modelo de RLogitS, tanto en los datos de entrenamiento como en los datos de prueba. Para esto, usaremos el umbral por defecto, y reordenamos las clases para que la clase positiva sea `sí`.

```
umbral <- 0.5

rlogits_probs_train <- fitted(rlogits)
rlogits_preds_train <- sapply(rlogits_probs_train,
function (p) ifelse (p < umbral, "no", "sí"))
rlogits_preds_train <- factor(rlogits_preds_train, levels = rev(levels(muestra_train[[nombre_res
puesta]])))

rlogits_probs_test <- predict(rlogits, muestra_test, type = "response")
rlogits_preds_test <- sapply(rlogits_probs_test,
function (p) ifelse (p < umbral, "no", "sí"))
rlogits_preds_test <- factor(rlogits_preds_test, levels = rev(levels(muestra_train[[nombre_respu
esta]])))
```

Teniendo las predicciones, podemos formar las matrices de confusión y calcular la sensibilidad y especificidad (teniendo cuidado de también dar vuelta las clases en los datos observados).

```
rlogits_obs_train <- factor(rlogits[["data"]][names(fitted(rlogits)), nombre_respuesta], levels
= rev(levels(muestra_train[[nombre_respuesta]])))
rlogits_obs_test <- factor(muestra_test[[nombre_respuesta]], levels = rev(levels(muestra_train
[[nombre_respuesta]])))

rlogits_train_conf_mat <- table(Predicho = rlogits_preds_train, Observado = rlogits_obs_train)
rlogits_test_conf_mat <- table(Predicho = rlogits_preds_test, Observado = rlogits_obs_test)

cat("Matriz de confusión del modelo de RLogitS en datos de entrenamiento:\n")
print(rlogits_train_conf_mat)
cat("\n")
cat("Matriz de confusión del modelo de RLogitS en datos de prueba:\n")
print(rlogits_test_conf_mat)
```

```
Matriz de confusión del modelo de RLogitS en datos de entrenamiento:
        Observado
Predicho sí no
      sí 34 15
      no 16 35

Matriz de confusión del modelo de RLogitS en datos de prueba:
        Observado
Predicho sí no
      sí 19  9
      no  6 16
```

Obtengamos la exactitud, sensibilidad y especificidad en cada caso y comparemos sus diferencias al pasar de datos vistos por el modelo a no vistos.

```
rlogits_train_exa <- (rlogits_train_conf_mat[1, 1] + rlogits_train_conf_mat[2, 2]) /
sum(rlogits_train_conf_mat)
rlogits_train_sen <- rlogits_train_conf_mat[1, 1] /
sum(rlogits_train_conf_mat[, 1])
rlogits_train_esp <- rlogits_train_conf_mat[2, 2] /
sum(rlogits_train_conf_mat[, 2])

rlogits_test_exa <- (rlogits_test_conf_mat[1, 1] + rlogits_test_conf_mat[2, 2]) /
sum(rlogits_test_conf_mat)
rlogits_test_sen <- rlogits_test_conf_mat[1, 1] /
sum(rlogits_test_conf_mat[, 1])
rlogits_test_esp <- rlogits_test_conf_mat[2, 2] /
sum(rlogits_test_conf_mat[, 2])

rlogits_cambio_exa <- (rlogits_train_exa - rlogits_test_exa) / rlogits_test_exa * 100
rlogits_cambio_sen <- (rlogits_train_sen - rlogits_test_sen) / rlogits_test_sen * 100
rlogits_cambio_esp <- (rlogits_train_esp - rlogits_test_esp) / rlogits_test_esp * 100

cat("Rendimiento del modelo de RLogitS en datos de entrenamiento:\n")
cat(sprintf("    Exactitud: %.2f\n", rlogits_train_exa))
cat(sprintf(" Sensibilidad: %.2f\n", rlogits_train_sen))
cat(sprintf("Especificidad: %.2f\n", rlogits_train_esp))
cat("\n")
cat("Rendimiento del modelo de RLogitS en datos de prueba:\n")
cat(sprintf("    Exactitud: %.2f\n", rlogits_test_exa))
cat(sprintf(" Sensibilidad: %.2f\n", rlogits_test_sen))
cat(sprintf("Especificidad: %.2f\n", rlogits_test_esp))
cat("\n")
cat("Cambio porcentual en el rendimiento del modelo de RLogitS:\n")
cat(sprintf("    Exactitud: %7.2f%%\n", rlogits_cambio_exa))
cat(sprintf(" Sensibilidad: %7.2f%%\n", rlogits_cambio_sen))
cat(sprintf("Especificidad: %7.2f%%\n", rlogits_cambio_esp))
```

```
Rendimiento del modelo de RLogitS en datos de entrenamiento:
    Exactitud: 0.69
 Sensibilidad: 0.68
Especificidad: 0.70

Rendimiento del modelo de RLogitS en datos de prueba:
    Exactitud: 0.70
 Sensibilidad: 0.76
Especificidad: 0.64

Cambio porcentual en el rendimiento del modelo de RLogitS:
    Exactitud:   -1.43%
 Sensibilidad:  -10.53%
Especificidad:    9.37%
```

Vemos que la exactitud no sufre un cambio importante, pero sí se observa un aumnento en la sensibilidad y una caída de la especificidad. En general, parece que el modelo se comporta bien con datos no vistos.

Repitamos el análisis con el modelo múltiple.

```
rlogitm_probs_train <- fitted(rlogitm)
rlogitm_preds_train <- sapply(rlogitm_probs_train,
function (p) ifelse (p < umbral, "no", "sí"))
rlogitm_preds_train <- factor(rlogitm_preds_train, levels = rev(levels(muestra_train[[nombre_res
puesta]])))

rlogitm_probs_test <- predict(rlogitm, muestra_test, type = "response")
rlogitm_preds_test <- sapply(rlogitm_probs_test,
function (p) ifelse (p < umbral, "no", "sí"))
rlogitm_preds_test <- factor(rlogitm_preds_test, levels = rev(levels(muestra_train[[nombre_respu
esta]])))

rlogitm_obs_train <- factor(rlogitm[["data"]][names(fitted(rlogitm)), nombre_respuesta],
                            levels = rev(levels(muestra_train[[nombre_respuesta]])))
rlogitm_obs_test <- factor(muestra_test[[nombre_respuesta]],
                           levels = rev(levels(muestra_train[[nombre_respuesta]])))

rlogitm_train_conf_mat <- table(Predicho = rlogitm_preds_train, Observado = rlogitm_obs_train)
rlogitm_test_conf_mat <- table(Predicho = rlogitm_preds_test, Observado = rlogitm_obs_test)

cat("Matriz de confusión del modelo de RLogitM en datos de entrenamiento:\n")
print(rlogitm_train_conf_mat)
cat("\n")
cat("Matriz de confusión del modelo de RLogitM en datos de prueba:\n")
print(rlogitm_test_conf_mat)
```

```
Matriz de confusión del modelo de RLogitM en datos de entrenamiento:
        Observado
Predicho sí no
      sí 39  9
      no 11 41

Matriz de confusión del modelo de RLogitM en datos de prueba:
        Observado
Predicho sí no
      sí 18 10
      no  7 15
```

Obtengamos las métricas de desempeño y comparémoslas al pasar de datos vistos a los no vistos.

```
rlogitm_train_exa <- (rlogitm_train_conf_mat[1, 1] + rlogitm_train_conf_mat[2, 2]) /
sum(rlogitm_train_conf_mat)
rlogitm_train_sen <- rlogitm_train_conf_mat[1, 1] /
sum(rlogitm_train_conf_mat[, 1])
rlogitm_train_esp <- rlogitm_train_conf_mat[2, 2] /
sum(rlogitm_train_conf_mat[, 2])

rlogitm_test_exa <- (rlogitm_test_conf_mat[1, 1] + rlogitm_test_conf_mat[2, 2]) /
sum(rlogitm_test_conf_mat)
rlogitm_test_sen <- rlogitm_test_conf_mat[1, 1] /
sum(rlogitm_test_conf_mat[, 1])
rlogitm_test_esp <- rlogitm_test_conf_mat[2, 2] /
sum(rlogitm_test_conf_mat[, 2])

rlogitm_cambio_exa <- (rlogitm_train_exa - rlogitm_test_exa) / rlogitm_test_exa * 100
rlogitm_cambio_sen <- (rlogitm_train_sen - rlogitm_test_sen) / rlogitm_test_sen * 100
rlogitm_cambio_esp <- (rlogitm_train_esp - rlogitm_test_esp) / rlogitm_test_esp * 100

cat("Rendimiento del modelo de RLogitM en datos de entrenamiento:\n")
cat(sprintf("    Exactitud: %.2f\n", rlogitm_train_exa))
cat(sprintf(" Sensibilidad: %.2f\n", rlogitm_train_sen))
cat(sprintf("Especificidad: %.2f\n", rlogitm_train_esp))
cat("\n")
cat("Rendimiento del modelo de RLogitM en datos de prueba:\n")
cat(sprintf("    Exactitud: %.2f\n", rlogitm_test_exa))
cat(sprintf(" Sensibilidad: %.2f\n", rlogitm_test_sen))
cat(sprintf("Especificidad: %.2f\n", rlogitm_test_esp))
cat("\n")
cat("Cambio porcentual en el rendimiento del modelo de RLogitM:\n")
cat(sprintf("    Exactitud: %7.2f%%\n", rlogitm_cambio_exa))
cat(sprintf(" Sensibilidad: %7.2f%%\n", rlogitm_cambio_sen))
cat(sprintf("Especificidad: %7.2f%%\n", rlogitm_cambio_esp))
```

```
Rendimiento del modelo de RLogitM en datos de entrenamiento:
    Exactitud: 0.80
 Sensibilidad: 0.78
Especificidad: 0.82

Rendimiento del modelo de RLogitM en datos de prueba:
    Exactitud: 0.66
 Sensibilidad: 0.72
Especificidad: 0.60

Cambio porcentual en el rendimiento del modelo de RLogitM:
    Exactitud:   21.21%
 Sensibilidad:    8.33%
Especificidad:   36.67%
```

¡Oh! Aquí sí hay una caída notoria de todas las métricas de desempeño cuando el modelo hace predicciones con datos no vistos.

Resultado

Ambos modelos muestran un **calidad predictiva moderada**, con una sensibilidad sobre 70% y una especificidad sobre 60% en datos no utilizados para construirlos.

El modelo simple muestra cierta estabilidad en el rendimiento al pasar de datos conocidos a desconocidos. Sin embargo, el modelo de RLogM parece tener **problemas de generalización** puesto que presenta una caída importante en el rendimiento al ser aplicado a datos no vistos. Esto es una indicación de sobreajuste y habría que explorar la eliminación de algún predictor, aunque eso nos haría incumplir con lo solicitado en el enunciado.

Referencias

Heinz, G., Peterson, L. J., Johnson, R. W., & Kerk, C. J. (2003). Exploring relationships in body dimensions. Journal of Statistics Education, 11(2).