Herramientas avanzadas para modelos de regresión

Ejemplo de solución ejercicio prático N°11

Enunciado

Para este ejercicio usaremos los datos de medidas anatómicas recolectados por Heinz et al. (2003) que ya hemos utilizado, con la adición de la variable `TRG` considerada en el ejercicio práctico anterior.

En este contexto realizaremos las siguientes actividades:

1. **Definir la semilla a utilizar, que corresponde a los primeros cinco dígitos del RUN del integrante de mayor edad del equipo.**
2. **Seleccionar una muestra de 100 personas, asegurando que la mitad tenga rodillas gruesas (`TRG == "sí"`) y la otra mitad no (`TRG == "no"`).**
3. **Usando las herramientas del paquete `leaps` , realizar una búsqueda exhaustiva para seleccionar entre dos y ocho predictores que ayuden a estimar el diámetro (promedio) de las rodillas (`Knees.diameter`), obviamente sin considerar la nueva variable `TRG` , y luego utilizar las funciones del paquete `caret` para construir un modelo de regresión lineal múltiple con los predictores escogidos y evaluarlo usando bootstrapping.**
4. **Haciendo un poco de investigación sobre el paquete `caret` , en particular cómo hacer Recursive Feature Elimination (RFE), construir un modelo de regresión lineal múltiple para predecir la variable `Knees.diameter` que incluya entre 5 y 15 predictores, seleccionando el conjunto de variables que maximice R^2 y que use cinco repeticiones de validación cruzada de cinco pliegues para evitar el sobreajuste (obviamente no se debe considerar la variable `TRG`).**
5. **Usando RFE, construir un modelo de regresión logística múltiple para la variable `TRG` que incluya el conjunto de predictores, entre cuatro y doce, que entregue la mejor curva ROC y que utilice validación cruzada dejando uno fuera para evitar el sobreajuste (obviamente no se debe considerar la variable `Knees.diameter`).**
6. **Pronunciarse sobre la confiabilidad y el poder predictivo de los modelos obtenidos.**

Comencemos Incluyendo los paquetes que usaremos en este script.

```
library(car)
library(caret)
library(dplyr)
library(ggpubr)
library(leaps)
library(pROC)
library(psych)
```

Obtengamos los datos en formato ancho.

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

datos <- read.csv2(file = src_file, stringsAsFactors = TRUE)
datos[["Gender"]] <- factor(datos[["Gender"]])
```

Generemos las variables nuevas requeridas para este ejercicio.

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

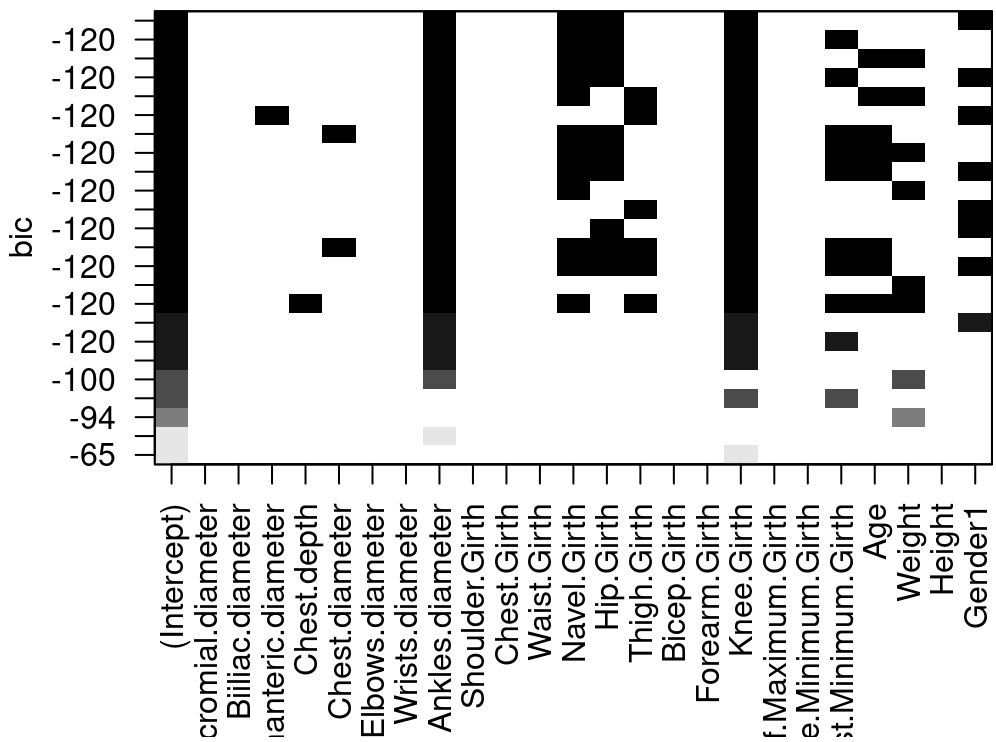
Obtenemos la muestra como indican las instrucciones 1 y 2, teniendo cuidado de *desordenar* los conjuntos de datos para que no queden juntos todos los casos con la misma clase, puesto que introduce artificialmente dependencia entre los datos.

```
set.seed(11111)
muestra_a <- datos_ext |> filter(TRG == "no") |> sample_n(50, replace = FALSE)
muestra_b <- datos_ext |> filter(TRG == "sí") |> sample_n(50, replace = FALSE)
muestra_ext <- rbind(muestra_a, muestra_b) |> sample_frac(1L)
```

Regresión lineal múltiple usando el paquete `leaps`

Para cumplir la instrucción 3, buscaremos los predictores de forma *exhaustiva*, teniendo cuidado de indicar la variable prohibida.

```
respuesta_lineal <- "Knees.diameter"
respuesta_binaria <- "TRG"

rlm1_df <- muestra_ext |> select(-all_of(respuesta_binaria))
rlm1_fmla <- formula(paste(respuesta_lineal, ".", sep = " ~ "))
rlm1_sets <- regsubsets(rlm1_fmla, data = rlm1_df, nbest = 3, nvmax = 8, method = "exhaustive")
rlm1_sets_summ <- summary(rlm1_sets)
rlm1_sets_i_mejor <- which.min(rlm1_sets_summ[["bic"]])
rlm1_seleccion <- names(which(rlm1_sets_summ[["which"]][rlm1_sets_i_mejor, ])[-1])

plot(rlm1_sets, main = "Subconjuntos modelo de RLM 1")
```




```
cat("Mejores predictores para el modelo de RLM 1:\n")
print(rlm1_seleccion)
```

```
Mejores predictores para el modelo de RLM 1:
[1] "Ankles.diameter" "Navel.Girth"     "Hip.Girth"       "Knee.Girth"
[5] "Gender1"
```

Vemos que hay varios subconjuntos que llevan a un BIC de alrededor de -120. El mejor subconjunto considera una variable indicadora (`Gender1`) que en realidad no aparece en la matriz de datos. Debemos tener cuidado de cambiarla por el nombre verdadero antes de usar este conjunto para construir el modelo. Para ello usaremos la función `train()` del paquete caret, indicando que use bootstrapping con `B` repeticiones para evitar sobreajuste, teniendo cuidado de definir una semilla para poder reproducir el mismo resultado cada vez que se ejecute el código.

```
rlm1_seleccion[5] <- "Gender"
rlm1_sel_text <- paste(rlm1_seleccion, collapse = " + ")
rlm1_fmla <- formula(paste(respuesta_lineal, rlm1_sel_text, sep = " ~ "))

B = 1999
set.seed(11 * 11111)
rlm1_train <- train(rlm1_fmla, data = rlm1_df, method = "lm",
                    trControl = trainControl(method = "boot", number = B))
rlm1 <- rlm1_train[["finalModel"]]

cat("Modelo de RLM 1:\n")
print(summary(rlm1))
```

```
Modelo de RLM 1:

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.38769 -0.28241 -0.04487  0.37080  1.49998 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)      2.93227    1.13043   2.594 0.011006 *  
Ankles.diameter  0.42012    0.08343   5.036 2.29e-06 ***
Navel.Girth     -0.03489    0.01191  -2.928 0.004274 ** 
Hip.Girth        0.07532    0.01890   3.985 0.000133 ***
Knee.Girth       0.15367    0.03934   3.906 0.000177 ***
Gender1          0.61814    0.17182   3.598 0.000514 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6185 on 94 degrees of freedom
Multiple R-squared:  0.7815,    Adjusted R-squared:  0.7698 
F-statistic: 67.23 on 5 and 94 DF,  p-value: < 2.2e-16
```

Multicolinealidad

Cuando los modelos tienen muchos predictores, la probabilidad de que exista multicolinealidad aumenta. Por eso, es bueno que descartemos este potencial problema tempranamente.

```
cat("Factores de inflación de la varianza:\n")
print(vif(rlm1))
cat("\n")
cat("Valores de tolerancia:\n")
print(1 / vif(rlm1))
```

```
Factores de inflación de la varianza:
Ankles.diameter     Navel.Girth       Hip.Girth      Knee.Girth         Gender1
       2.405305        2.787130        4.327124        2.699718        1.891284

Valores de tolerancia:
Ankles.diameter     Navel.Girth       Hip.Girth      Knee.Girth         Gender1
      0.4157478       0.3587920       0.2311004       0.3704091       0.5287412
```

Vemos que el predictor `Hip.Girth` está relativamente cerca del límite para declarar un problema de multicolinealidad. Para jugar seguro, mejor quitemos este predictor del modelo.

```
rlm1_seleccion <- rlm1_seleccion[-3]
rlm1_sel_text <- paste(rlm1_seleccion, collapse = " + ")
rlm1_fmla <- formula(paste(respuesta_lineal, rlm1_sel_text, sep = " ~ "))

set.seed(11 * 11111)
rlm1_train <- train(rlm1_fmla, data = rlm1_df, method = "lm",
                    trControl = trainControl(method = "boot", number = B))
rlm1<- rlm1_train[["finalModel"]]

cat("Modelo de RLM 1 con cuatro predictores:\n")
print(summary(rlm1))
cat("Factores de inflación de la varianza:\n")
print(vif(rlm1))
cat("\n")
cat("Valores de tolerancia:\n")
print(1 / vif(rlm1))
```

```
Modelo de RLM 1 con cuatro predictores:

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.4967 -0.3223  0.0071  0.3946  2.3309 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)      3.985153   1.182097   3.371  0.00108 ** 
Ankles.diameter  0.428360   0.089700   4.775 6.50e-06 ***
Navel.Girth     -0.005373   0.010037  -0.535  0.59365    
Knee.Girth       0.255146   0.032258   7.909 4.67e-12 ***
Gender1          0.497535   0.181896   2.735  0.00744 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6652 on 95 degrees of freedom
Multiple R-squared:  0.7445,    Adjusted R-squared:  0.7338 
F-statistic: 69.22 on 4 and 95 DF,  p-value: < 2.2e-16

Factores de inflación de la varianza:
Ankles.diameter     Navel.Girth      Knee.Girth         Gender1 
       2.403826        1.710125        1.569042        1.832618 

Valores de tolerancia:
Ankles.diameter     Navel.Girth      Knee.Girth         Gender1 
      0.4160036       0.5847525       0.6373316       0.5456675 
```

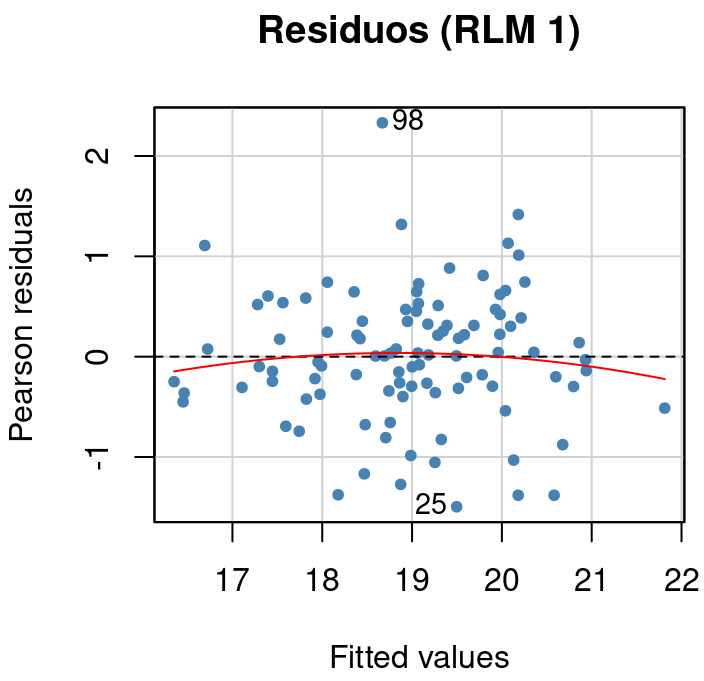
¡Bien! Ahora el modelo presenta niveles de multicolinealidad aceptables.

Ajuste y linealidad

En la salida a pantalla anterior, podemos observar que el modelo obtenido consigue una reducción significativa de la varianza no explicada ($F(4, 95) = 69{,}22;\ p < 0.001$) respecto del modelo nulo.

Comprobemos ahora que los residuos cumplen las condiciones necesarias usando la función `residualPlots()` del paquete `car`. Sin embargo, las funciones de este paquete tienen problemas encontrando información usada por la función `train()` del paquete `caret` en la construcción del modelo. Por esta razón, primero creamos un modelo de la manera tradicional que es **equivalente** al modelo final obtenido por `train()`.

```
rlm1_equiv <- lm(rlm1_fmla, rlm1_df)

cat("Prueba de curvatura para los predictores del modelo de RLM 1:\n")
residualPlots(rlm1_equiv, terms = ~ 1,
              col = "steelblue", pch = 20, col.quad = "red",
              id = list(cex = 0.9, location = "lr"))
title("Residuos (RLM 1)")
```

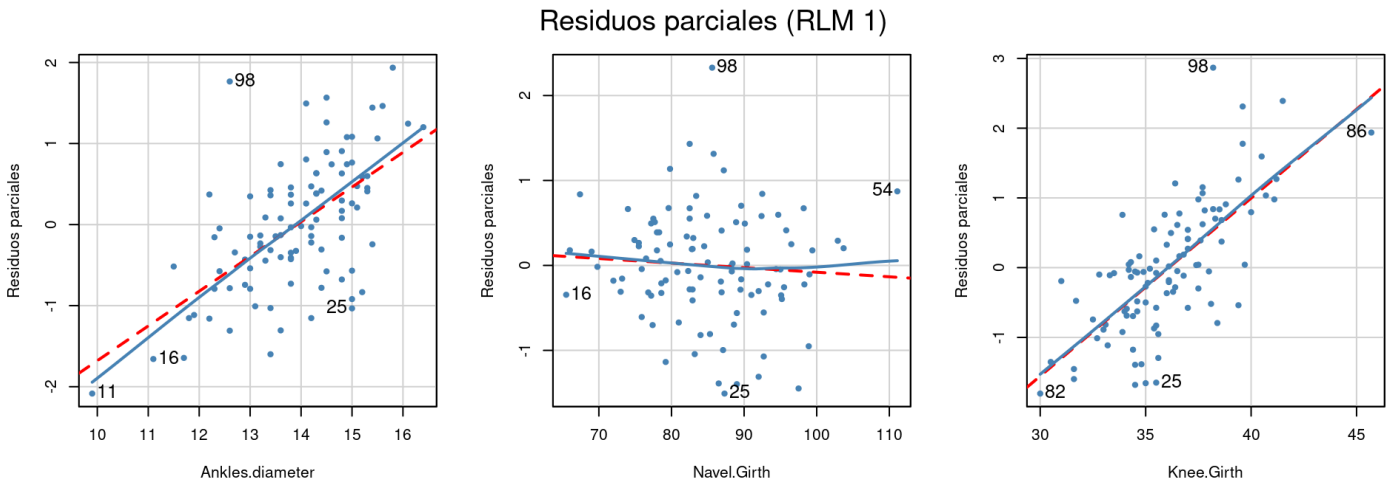



```
Prueba de curvatura para los predictores del modelo de RLM 1:
           Test stat Pr(>|Test stat|)
Tukey test   -0.7522           0.452
```

Vemos que, si bien hay un caso atípico (98), no se observan patrones problemáticos, lo que es confirmado por las pruebas de curvatura aplicadas. Así, no hay evidencia para sospechar que los residuos no siguen una distribución normal centrada en cero para cada predictor (aunque se ven algunos posibles valores atípicos).

Revisemos que la variable de salida se relaciona linealmente con los predictores por medio del gráfico de residuos parciales que entrega la función `crPlots()` del paquete `car`.

```
crPlots(rlm1_equiv, terms = ~ . - Gender, layout = c(1, 3),
        col = "steelblue", pch = 20, col.lines = c("red", "steelblue"),
        smooth = list(smoother = loessLine, span = 1),
        id = list(cex = 1.2, location = "lr"),
        main = "Residuos parciales (RLM 1)", ylab = "Residuos parciales")
```



Primero, notamos que las relaciones entre cada predictor y la variable respuesta son aproximadamente lineales. Segundo, el modelo (línea segmentada roja) se ajusta bien a las relaciones observadas (líneas continua azul-acero), con unas leves desviaciones en los datos más extremos que evita apalancamiento.

Tampoco se ven cambios en notorios en la varianza, lo que podemos confirmar con la prueba de varianza del error no constante.

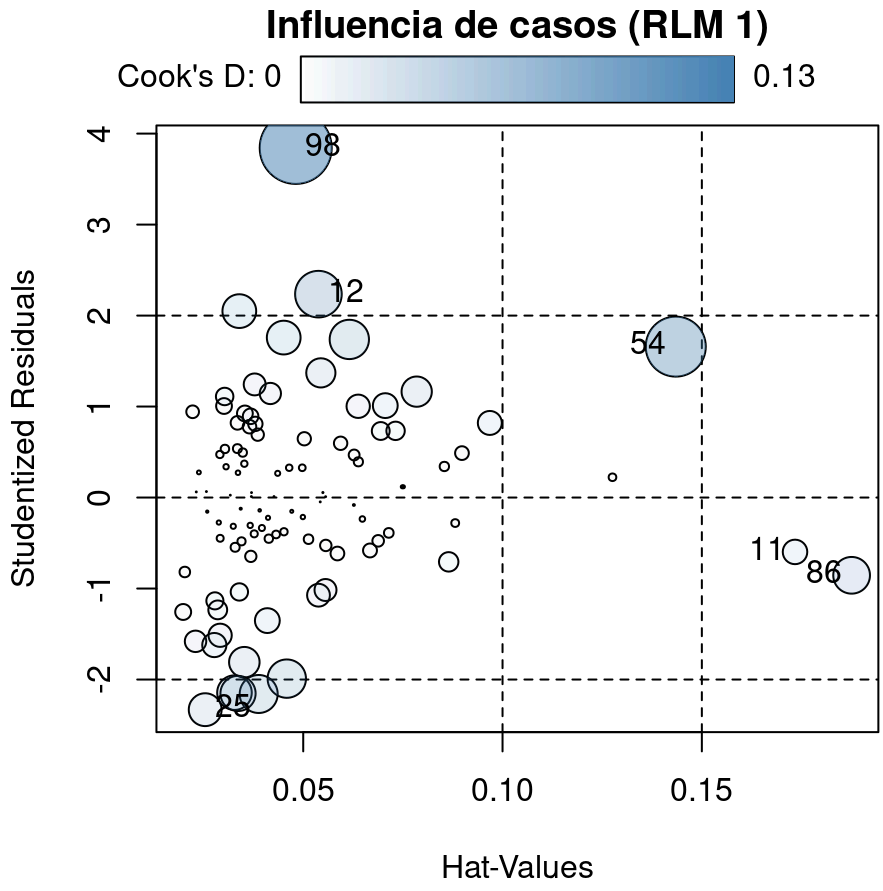
```
cat("Prueba de varianza del error no constante:\n")
ncvTest(rlm1_equiv)
```

```
Prueba de varianza del error no constante:
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 0.8433325, Df = 1, p = 0.35845
```

Casos sobreinfluyentes

Usemos el gráfico de diagnóstico disponible en el paquete `car` entregado por la función `influencePlot()` que ya hemos usado en ejercicios prácticos anteriores.

```
rlm1_inf_estad <- influencePlot(rlm1_equiv, fill.col = "steelblue",
                                scale = 5, id = list(n = 3),
                                main = "Influencia de casos (RLM 1)\n")
```

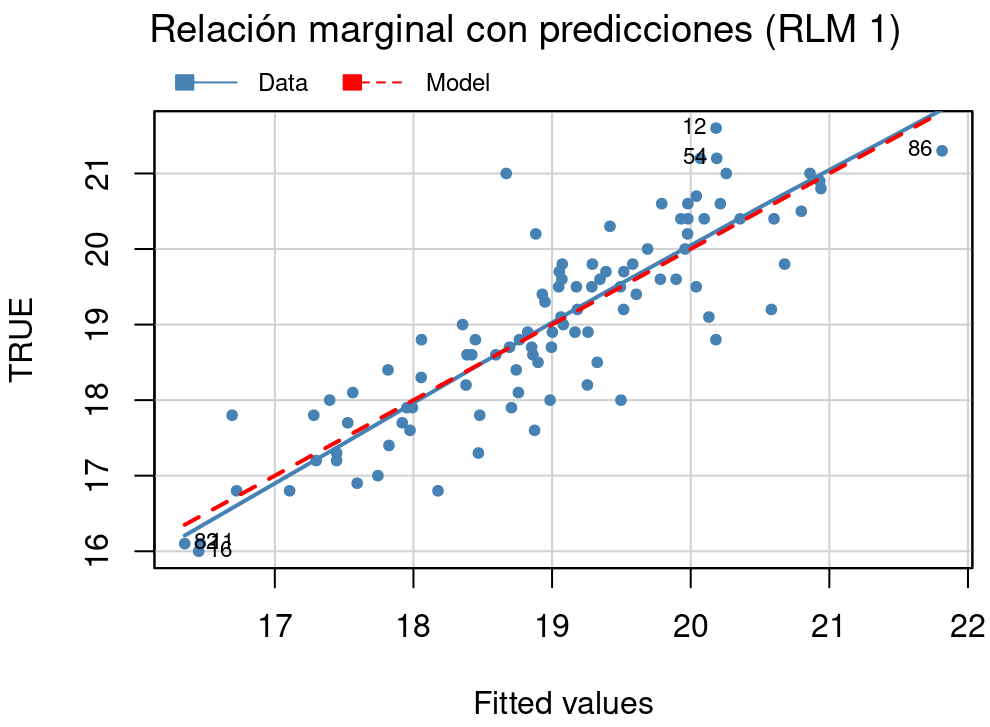


```
cat("Límites para el modelo de RLM 1:\n")
cat("Rango para 95% de los residuos studentizados: ")
cat("[", round(qt(0.05/2, nrow(rlm1_df) - length(predictors(rlm1)) - 2), 3), ", ", sep = "")
cat(round(qt(1-0.05/2, nrow(rlm1_df) - length(predictors(rlm1)) - 2), 3), "]\n", sep = "")
cat("Límite del apalancamiento:", round(2 * mean(hatvalues(rlm1)), 3), "\n")
cat("Límite de la distancia de Cook:", round(3 * mean(cooks.distance(rlm1)), 3), "\n")
cat("\nCasos notorios para el modelo de RLM 1:\n")
print(rlm1_inf_estad)
```

```
Límites para el modelo de RLM 1:
Rango para 95% de los residuos studentizados: [-1.986, 1.986]
Límite del apalancamiento: 0.1
Límite de la distancia de Cook: 0.029

Casos notorios para el modelo de RLM 1:
      StudRes        Hat      CookD
11 -0.5976916 0.17337516 0.01508729
12  2.2353713 0.05381179 0.05454203
25 -2.3317219 0.02540675 0.02708227
54  1.6589126 0.14345937 0.09051523
86 -0.8546471 0.18757376 0.03382407
98  3.8428609 0.04811658 0.13039905
```

Ninguno de los casos notorios reportados por la función `influencePlot()` está fuera de rango en las tres métricas. Los casos 12 y 98 están alejados y exhiben una distancia de Cook alta, mientras que las observaciones 54 y 86 están fuera de los límites del apalancamiento y la distancia de Cook. Revisemos el impacto de estos casos en el modelo.

```
mmps(rlm1_equiv, terms = ~ 1,
        col = "steelblue", pch = 20, col.line = c("steelblue", "red"),
        smooth = list(smoother = loessLine, span = 1),
        id = list(n = 6, cex = 0.7, location = "lr"),
        main = "Relación marginal con predicciones (RLM 1)", sub = " ")
```



Podemos ver, en esta figura y en los gráficos de residuos parciales, que ninguno de los casos potencialmente problemáticos distorsiona la línea del modelo, por lo que no es necesario eliminar ninguna de estas observaciones.

Independencia de los residuos

Confirmemos que no existe dependencia entre los residuos generados por el modelo de RLM 1.

```
cat("Prueba de la independencia de los residuos para el modelo de RLM 1:\n")
print(durbinWatsonTest(rlm1))
```

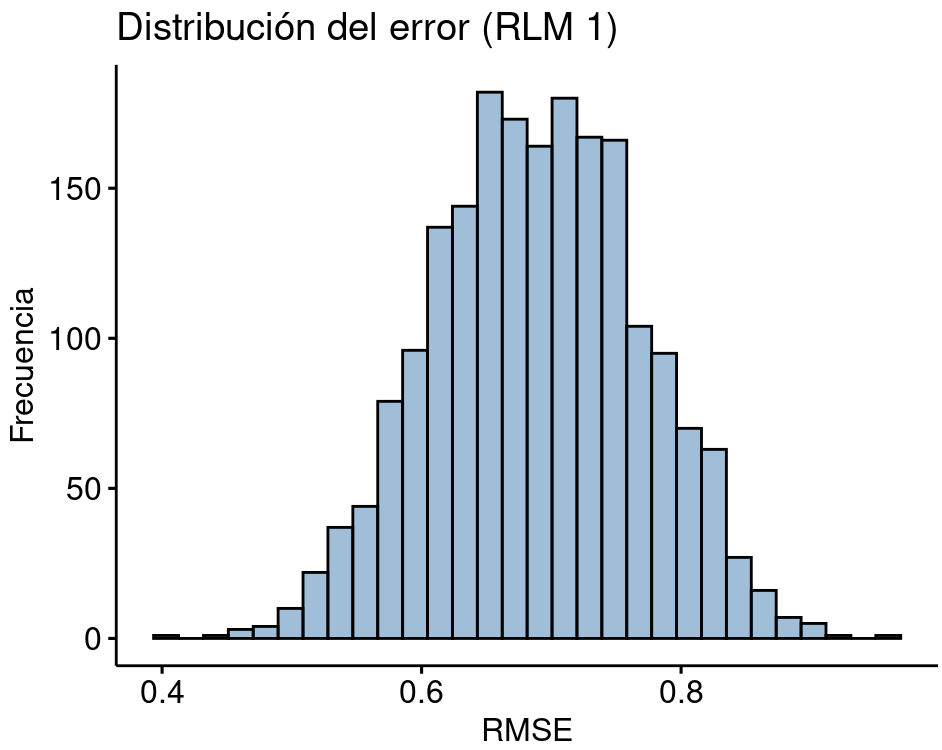
```
Prueba de la independencia de los residuos para el modelo de RLM 1:
 lag Autocorrelation D-W Statistic p-value
   1    -0.002855788      1.979999   0.912
 Alternative hypothesis: rho != 0
```

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

Desempeño

Veamos los niveles de error cometidos por el modelo de RLM 1 que hemos conseguido, analizando un histograma de los errores (RMSE) en cada repetición del bootstrapping y el reporte del error promedio generado por la función `train()`.

```
rlm1_err_df <- data.frame(RMSE = rlm1_train[["resample"]][["RMSE"]])
rlm1_err_p <- gghistogram(rlm1_err_df, x = "RMSE", bins = 30,
                          fill = "steelblue", ylab = "Frecuencia",
                          title = "Distribución del error (RLM 1)")
print(rlm1_err_p)
```




```
cat("Rendimiento del modelo de RLM 1:\n")
print(rlm1_train[["results"]], digits = 3)
cat("\nMás detalle del raíz del error cuadrático medio:\n")
print(describe(rlm1_err_df, trim = 0, quant = c(0.25, 0.75),
               skew = FALSE, IQR = TRUE), digits = 3)
```

```
Rendimiento del modelo de RLM 1:
  intercept  RMSE Rsquared   MAE RMSESD RsquaredSD  MAESD
1      TRUE 0.689    0.723 0.531   0.08     0.0736 0.0621

Más detalle del raíz del error cuadrático medio:
     vars    n  mean   sd median   min   max range    se   IQR Q0.25 Q0.75
RMSE    1 1999 0.689 0.08  0.688 0.408 0.965 0.557 0.002 0.111 0.634 0.745
```

Vemos que el error promedio que el modelo comete en sus estimaciones es de $0{,}689 \pm 0{,}080$ cm, lo que es bastante bueno si consideramos que la variable de respuesta varía entre $16{,}0$ y $21{,}6$ cm, con una media de $18{,}95$ cm. También podemos observar que la distribución del error es relativamente simétrica con un rango que va desde $0{,}408$ y $0{,}965$ cm con un rango intercuantil de $0{,}111$ ($[0{,}634;\ 0{,}745]$).

Regresión lineal múltiple usando Recursive Feature Elimination

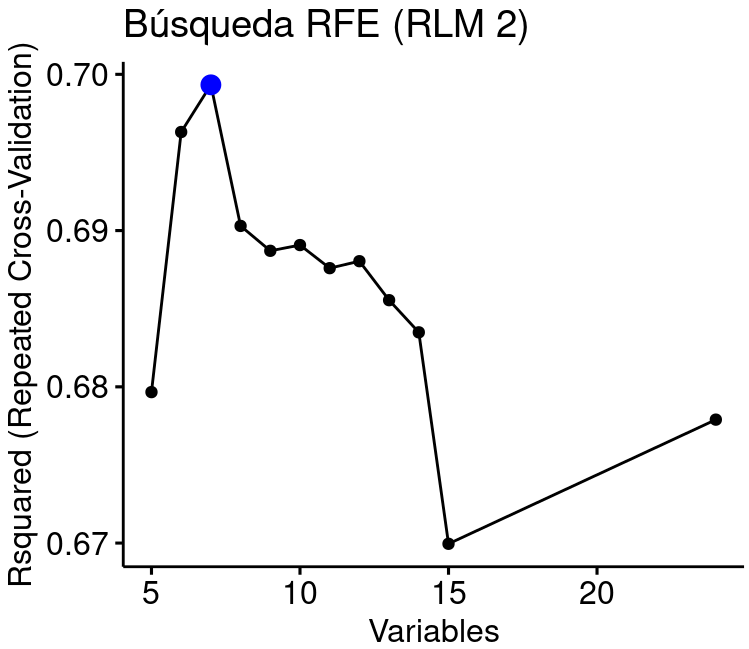
El paquete `caret` implementa la *regresión escalonada hacia atrás* bajo el nombre de Recursive Feature Elimination (RFE), mediante la función `rfe()`. Se pueden definir varias alternativas de control para guíar la búsqueda, incluyendo funciones *wrapper* para varios tipos de modelo. En particular, `caret` proporciona la función *wrapper* `lmFuncs` para trabajar modelos de regresión lineal.

La instrucción 4 nos indica buscar, mediante cinco repeticiones de validación cruzada de cinco pliegues, un modelo de RLM que consiga el mayor valor del coeficiente de determinación R^2 y que incluya entre 5 y 15 predictores. Esto podemos hacerlo con el siguiente código. Como la validación cruzada divide los datos de forma aleatoria, vamos a tener el cuidado de definir una semilla para su reproducibilidad.

```
rlm2_df <- muestra_ext |> select(-all_of(respuesta_binaria))
rlm2_fmla <- formula(paste(respuesta_lineal, ".", sep = " ~ "))
rlm2_control <- rfeControl(functions = lmFuncs, method = "repeatedcv",
                           number = 5, repeats = 5, verbose = FALSE)

set.seed(13 * 11111)
rlm2_rfe <- rfe(rlm2_fmla, data = rlm2_df, rfeControl = rlm2_control,
                sizes = 5:15, metric = "Rsquared")
rlm2 <- rlm2_rfe[["fit"]]
```

Veamos una representación gráfica del proceso de búsqueda realizado.

```
rlm2_rfe_p <- ggplot(rlm2_rfe) + theme_pubr()
rlm2_rfe_p <- ggpar(rlm2_rfe_p, title = "Búsqueda RFE (RLM 2)")
print(rlm2_rfe_p)
```



Podemos apreciar que la búsqueda obtuvo el valor del R^2 más alto con un modelo que considera 7 variables. Veamos el modelo obtenido.

```
cat("Modelo de RLM 2 obtenido con RFE:\n")
print(summary(rlm2))
```

```
Modelo de RLM 2 obtenido con RFE:

Call:
lm(formula = y ~ ., data = tmp)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.09011 -0.57630 -0.02443  0.50732  3.01450 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)         5.29544    1.64110   3.227 0.001735 ** 
Gender1            -0.33824    0.30246  -1.118 0.266355    
Wrist.Minimum.Girth 0.15075    0.15929   0.946 0.346440    
Ankles.diameter     0.43945    0.12635   3.478 0.000773 ***
Wrists.diameter     0.05675    0.21346   0.266 0.790929    
Elbows.diameter     0.03287    0.13747   0.239 0.811555    
Chest.depth         0.04839    0.05063   0.956 0.341794    
Forearm.Girth       0.12670    0.07477   1.695 0.093544 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8204 on 92 degrees of freedom
Multiple R-squared:  0.6237,    Adjusted R-squared:  0.595 
F-statistic: 21.78 on 7 and 92 DF,  p-value: < 2.2e-16
```

Multicolinealidad

Revisemos los niveles de multicolinealidad del modelo obtenido.

```
cat("Factores de inflación de la varianza:\n")
print(vif(rlm2))
cat("\n")
cat("Valores de tolerancia:\n")
print(1 / vif(rlm2))
```

```
Factores de inflación de la varianza:
            Gender1 Wrist.Minimum.Girth     Ankles.diameter     Wrists.diameter 
           3.330997            6.796051            3.135140            5.706823 
    Elbows.diameter         Chest.depth       Forearm.Girth 
           4.820786            2.189152            6.295889 

Valores de tolerancia:
            Gender1 Wrist.Minimum.Girth     Ankles.diameter     Wrists.diameter 
          0.3002104           0.1471443           0.3189651           0.1752288 
    Elbows.diameter         Chest.depth       Forearm.Girth 
          0.2074350           0.4567980           0.1588338 
```

Vemos que hay varios predictores con valores de inflación de la varianza cercanos o sobre 5. La variable `Wrist.Minimum.Girth` es la que presenta el valor más alto, por lo que es mejor quitarla del modelo.

```
rlm2_seleccion <- predictors(rlm2)[-2]
rlm2_seleccion[1] <- "Gender"
rlm2_sel_text <- paste(rlm2_seleccion, collapse = " + ")
rlm2_fmla <- formula(paste(respuesta_lineal, rlm2_sel_text, sep = " ~ "))

set.seed(13 * 11111)
rlm2_train <- train(rlm2_fmla, data = rlm2_df, method = "lm",
                    trControl = trainControl(method = "repeatedcv", number = 5, repeats = 5))
rlm2<- rlm2_train[["finalModel"]]

cat("Nuevos factores de inflación de la varianza:\n")
print(vif(rlm2))
cat("\n")
cat("Nuevos valores de tolerancia:\n")
print(1 / vif(rlm2))
```

```
Nuevos factores de inflación de la varianza:
        Gender1 Ankles.diameter Wrists.diameter Elbows.diameter     Chest.depth
       3.329586        3.099705        4.519014        4.819889        2.153033
  Forearm.Girth
       4.884168

Nuevos valores de tolerancia:
        Gender1 Ankles.diameter Wrists.diameter Elbows.diameter     Chest.depth
      0.3003376       0.3226114       0.2212872       0.2074737       0.4644610
  Forearm.Girth
      0.2047432
```

Podemos apreciar que mejoran los valores de inflación de la varianza, aunque la variable `Forearm.Girth` sigue presentando un valor alto. Mejor quitarlo del modelo.

```
rlm2_seleccion <- rlm2_seleccion[-6]
rlm2_sel_text <- paste(rlm2_seleccion, collapse = " + ")
rlm2_fmla <- formula(paste(respuesta_lineal, rlm2_sel_text, sep = " ~ "))

set.seed(13 * 11111)
rlm2_train <- train(rlm2_fmla, data = rlm2_df, method = "lm",
                    trControl = trainControl(method = "repeatedcv", number = 5, repeats = 5))
rlm2 <- rlm2_train[["finalModel"]]

cat("Nuevos factores de inflación de la varianza (2):\n")
print(vif(rlm2))
cat("\n")
cat("Nuevos valores de tolerancia (2):\n")
print(1 / vif(rlm2))
```

```
Nuevos factores de inflación de la varianza (2):
        Gender1 Ankles.diameter Wrists.diameter Elbows.diameter     Chest.depth
       3.200859        3.094765        4.053677        4.168391        1.987696

Nuevos valores de tolerancia (2):
        Gender1 Ankles.diameter Wrists.diameter Elbows.diameter     Chest.depth
      0.3124161       0.3231263       0.2466896       0.2399007       0.5030950
```

Vemos que ahora los predictores presentan niveles de multicolinealidad más o menos aceptables. Como el enunciado nos exige un mínimo de 5 predictores, detenemos esta poda aquí, aunque es probable que todavía haya espacio para reducir más el modelo.

Ajuste y linealidad

Revisemos el modelo conseguido.

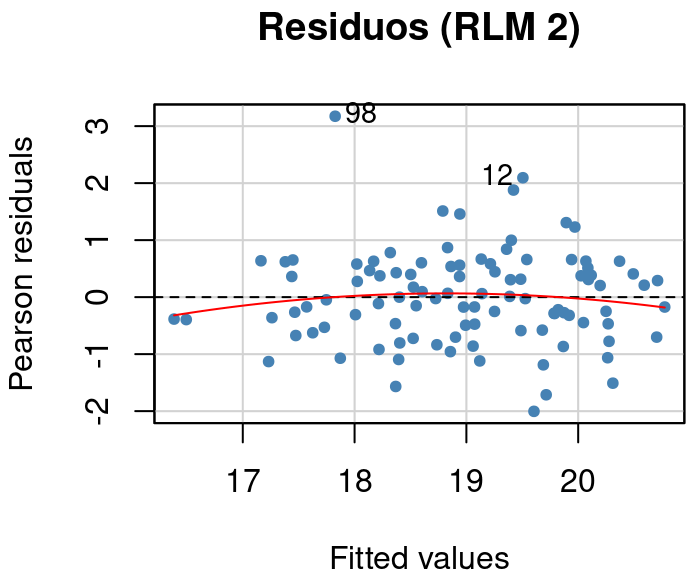
```
cat("Modelo de RLM 2 con cinco predictores:\n")
print(summary(rlm2), digits = 3)
```

```
Modelo de RLM 2 con cinco predictores:

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
   Min     1Q Median     3Q    Max
-2.004 -0.502 -0.026  0.519  3.174

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       6.0237     1.6372    3.68  0.00039 ***
Gender1          -0.1995     0.3040   -0.66  0.51329
Ankles.diameter   0.4400     0.1287    3.42  0.00093 ***
Wrists.diameter   0.2972     0.1845    1.61  0.11048
Elbows.diameter   0.1540     0.1311    1.18  0.24285
Chest.depth       0.0884     0.0495    1.79  0.07719 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.841 on 94 degrees of freedom
Multiple R-squared:  0.596, Adjusted R-squared:  0.574
F-statistic: 27.7 on 5 and 94 DF,  p-value: <2e-16
```

Observamos que el modelo consigue una reducción significa de la varianza no explicada en comparación al modelo nulo ($F(5, 94) = 27{,}71; p < 0.001$), aunque confirmamos que hay variables que no contribuyen significativamente a este ajuste y que podrían quitarse del modelo.

Revisemos el gráfico de diagnóstico de los residuos que genera este modelo (usando un modelo equivalente creado con las funciones base).

```
rlm2_equiv <- lm(rlm2_fmla, rlm2_df)

cat("Prueba de curvatura para los predictores del modelo de RLM 2:\n")
residualPlots(rlm2_equiv, terms = ~ 1,
              col = "steelblue", pch = 20, col.quad = "red",
              id = list(cex = 0.9, location = "lr"))
title("Residuos (RLM 2)")
```




```
Prueba de curvatura para los predictores del modelo de RLM 2:
           Test stat Pr(>|Test stat|)
Tukey test   -0.9153             0.36
```

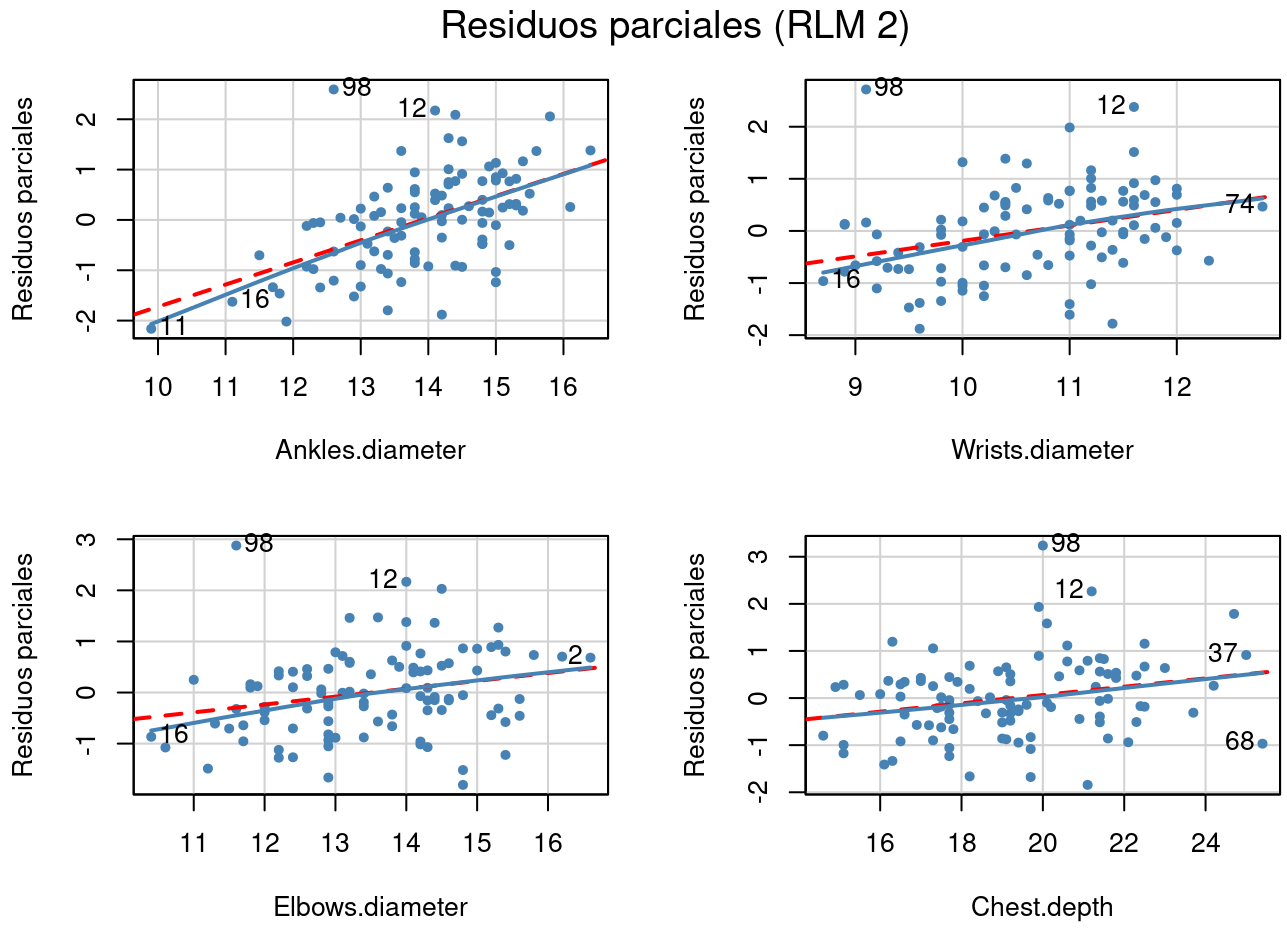
Vemos que los residuos muestran el comportamiento esperado, con el mismo caso atípico observado con el modelo anterior. Esto es confirmado por la prueba de curvatura, por lo que no tenemos evidencia para creer que los residuos no siguen una distribución normal con varianza constante. Si tuviéramos dudas, podríamos confirmar con gráficos y pruebas auxiliares, aunque deberíamos quitar este *único* caso atípico del análisis para mayor robustez cuando sea posible.

```
cat("Normalidad de los residuos generados por el modelo (RLM 2):\n")
shapiro.test(resid(rlm2)[-98])

cat("\nPrueba de varianza del error no constante (RLM 2):\n")
ncvTest(rlm2)
```

```
Normalidad de los residuos generados por el modelo (RLM 2):

    Shapiro-Wilk normality test

data:  resid(rlm2)[-98]
W = 0.98996, p-value = 0.6677


Prueba de varianza del error no constante (RLM 2):
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 0.04152027, Df = 1, p = 0.83854
```

Revisemos ahora la condición de linealidad entre predictores y variable de salida.

```
crPlots(rlm2_equiv, terms = ~ . - Gender,
        col = "steelblue", pch = 20, col.lines = c("red", "steelblue"),
        smooth = list(smoother = loessLine, span = 1),
        id = list(cex = 1.0, location = "lr"),
        main = "Residuos parciales (RLM 2)", ylab = "Residuos parciales")
```

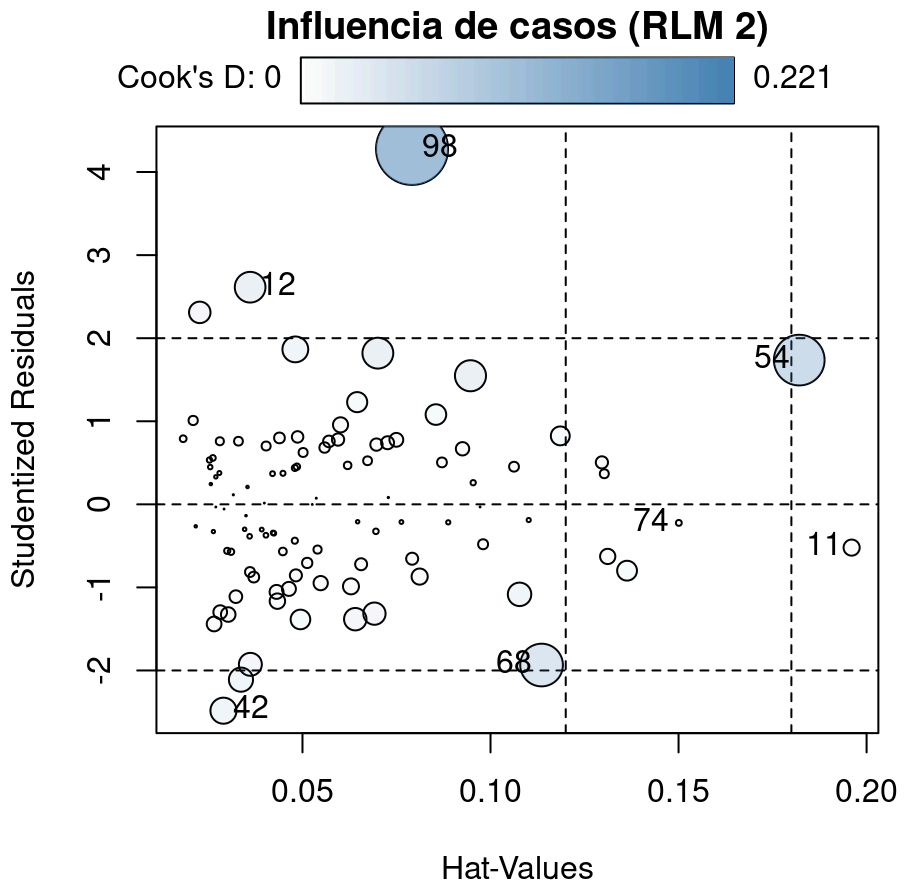


Observamos que las relaciones entre cada predictor y la variable respuesta son aproximadamente lineales y que el modelo logra ajustarse bien a estos datos, incluso evitando el apalancamiento que podría ejercer algunos valores en el extremo inferior de estas variables.

Casos sobreinfluyentes

Revisemos el gráfico de influencia y los casos notorios que se identifican en él.

```
rlm2_inf_estad <- influencePlot(rlm2_equiv, fill.col = "steelblue",
                                scale = 5, id = list(n = 3),
                                main = "Influencia de casos (RLM 2)\n")
```

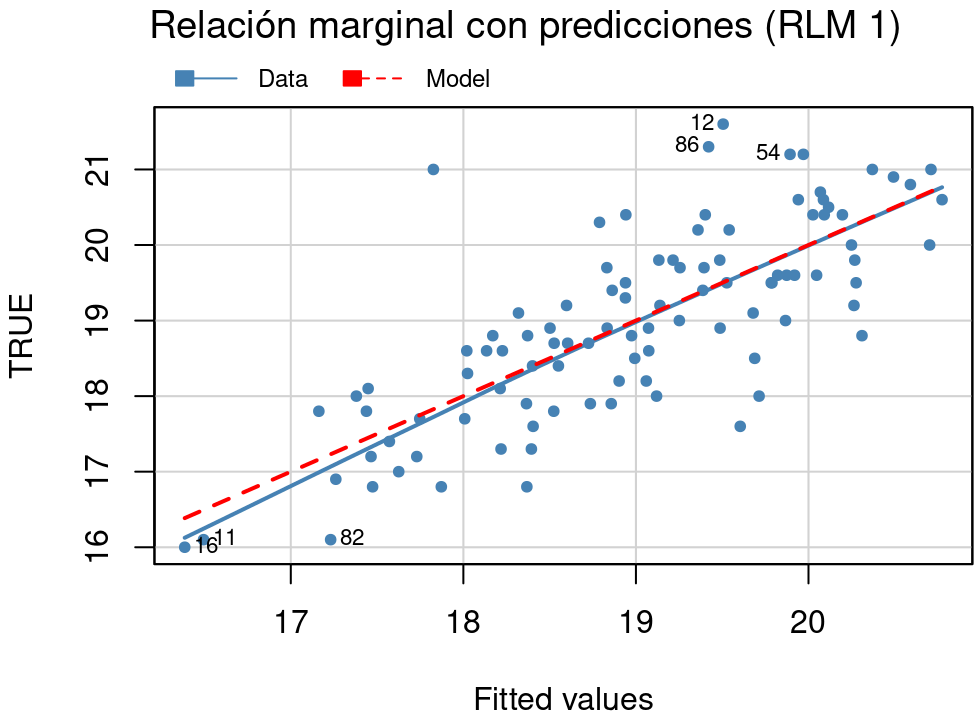



```
cat("Límites para el modelo de RLM 2:\n")
cat("Rango para 95% de los residuos studentizados: ")
cat("[", round(qt(0.05/2, nrow(rlm2_df) - length(predictors(rlm2)) - 2), 3), ", ", sep = "")
cat(round(qt(1-0.05/2, nrow(rlm2_df) - length(predictors(rlm2)) - 2), 3), "]\n", sep = "")
cat("Límite del apalancamiento:", round(2 * mean(hatvalues(rlm2)), 3), "\n")
cat("Límite de la distancia de Cook:", round(3 * mean(cooks.distance(rlm2)), 3), "\n")
cat("\nCasos notorios para el modelo de RLM 2:\n")
print(rlm2_inf_estad)
```

```
Límites para el modelo de RLM 2:
Rango para 95% de los residuos studentizados: [-1.986, 1.986]
Límite del apalancamiento: 0.12
Límite de la distancia de Cook: 0.033

Casos notorios para el modelo de RLM 2:
      StudRes        Hat       CookD
11 -0.5208844 0.19601905 0.011111271
12  2.6135436 0.03608069 0.040124218
42 -2.4837976 0.02898390 0.029091246
54  1.7361259 0.18209073 0.109493062
68 -1.9342825 0.11357271 0.077630916
74 -0.2233511 0.15007108 0.001483035
98  4.2782016 0.07913052 0.221379876
```

A priori, ningún residuo esta fuera de rango en los tres criterios. Los casos 12 y 98 son atípicos y con distancia de Cook alta, mientras que el caso 54 presenta apalancamiento y distancia de Cook fuera de los límites. Sin embargo, ninguno de estos casos parece influir demasiado en las rectas de regresiones parciales de arriba. Veamos su impacto en las predicciones del modelo.

```
mmps(rlm2_equiv, terms = ~ 1,
    col = "steelblue", pch = 20, col.line = c("steelblue", "red"),
    smooth = list(smoother = loessLine, span = 1),
    id = list(n = 6, cex = 0.7, location = "lr"),
    main = "Relación marginal con predicciones (RLM 1)", sub = " ")
```



Se puede observar que ninguno de los casos identificados como potencialmente problemático ejerce una influencia indebida en el modelo, que se ajusta bien a los datos, evitando incluso el apalancamiento que ejercen los casos 11, 16 y 82 en la parte baja de las predicciones.

Independencia de los residuos

Confirmemos que no existe dependencia entre los residuos generados por el modelo de RLM 2.

```
cat("Prueba de la independencia de los residuos para el modelo de RLM 1:\n")
print(durbinWatsonTest(rlm2))
```

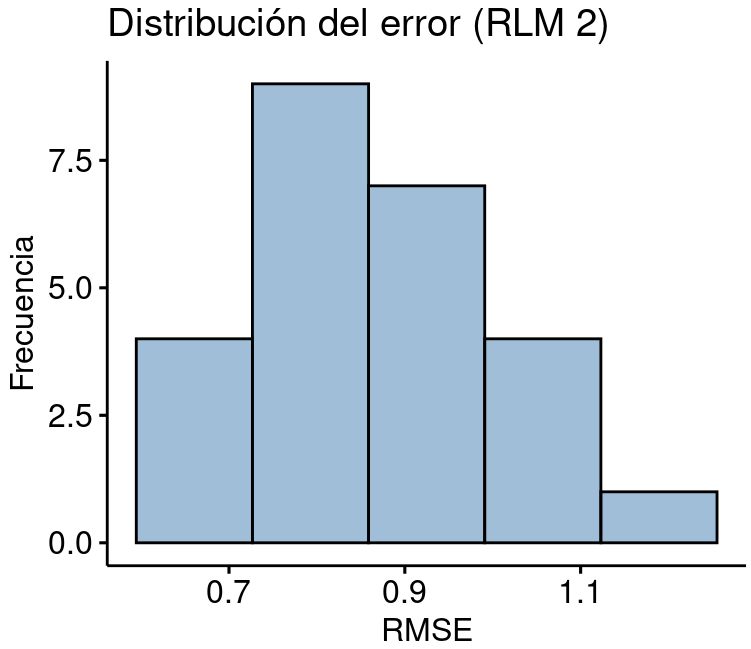
```
Prueba de la independencia de los residuos para el modelo de RLM 1:
 lag Autocorrelation D-W Statistic p-value
   1     -0.02315018      2.038852   0.824
 Alternative hypothesis: rho != 0
```

Vemos que no hay razones para rechazar la hipótesis de que los residuos de este modelo son independientes.

Desempeño

Veamos los niveles de error cometidos por el modelo de RLM 2 que hemos conseguido. Como antes, analizando un histograma de los errores (RMSE) en cada repetición, esta vez de la validación cruzada, además del reporte generado por la función `train()`.

```
rlm2_err_df <- data.frame(RMSE = rlm2_train[["resample"]][["RMSE"]])
rlm2_err_p <- gghistogram(rlm2_err_df, x = "RMSE", bins = 5,
                          fill = "steelblue", ylab = "Frecuencia",
                          title = "Distribución del error (RLM 2)")
print(rlm2_err_p)
```




```
cat("Rendimiento del modelo de RLM 2:\n")
print(rlm2_train[["results"]], digits = 3)
cat("\nMás detalle de la raíz del error cuadrático medio:\n")
print(describe(rlm2_err_df, trim = 0, quant = c(0.25, 0.75),
               skew = FALSE, IQR = TRUE), digits = 3)
```

```
Rendimiento del modelo de RLM 2:
  intercept  RMSE Rsquared   MAE RMSESD RsquaredSD  MAESD
1      TRUE 0.859    0.568 0.677  0.146      0.161 0.0881

Más detalle de la raíz del error cuadrático medio:
     vars  n  mean    sd median   min   max range    se   IQR Q0.25 Q0.75
RMSE    1 25 0.859 0.146  0.859 0.623 1.152 0.528 0.029 0.206 0.767 0.973
```

El modelo comete errores que van desde $0{,}623$ y $1{,}152$ cm ($0{,}859 \pm 0{,}146$ cm en promedio). Este resultado no es malo si consideramos que la variable de respuesta varía entre $16{,}0$ y $21{,}6$ cm.

Regresión logística múltiple usando RFE

La instrucción 5 nos pide usar RFE para conseguir un modelo de regresión logística múltiple (RLogitM), que incluya de 4 a 12 predictores, utilizando validación cruzada dejando uno fuera para evitar el sobreajuste.

Esto podemos hacerlo con el siguiente código, que indica a la función `rfe()` que utilice la función `twoClassSummary()` para medir el rendimiento del modelo, la que calcula las métricas de sensibilidad, especificidad y área bajo la curva ROC. Nuevamente definimos una semilla para poder reproducir la validación cruzada.

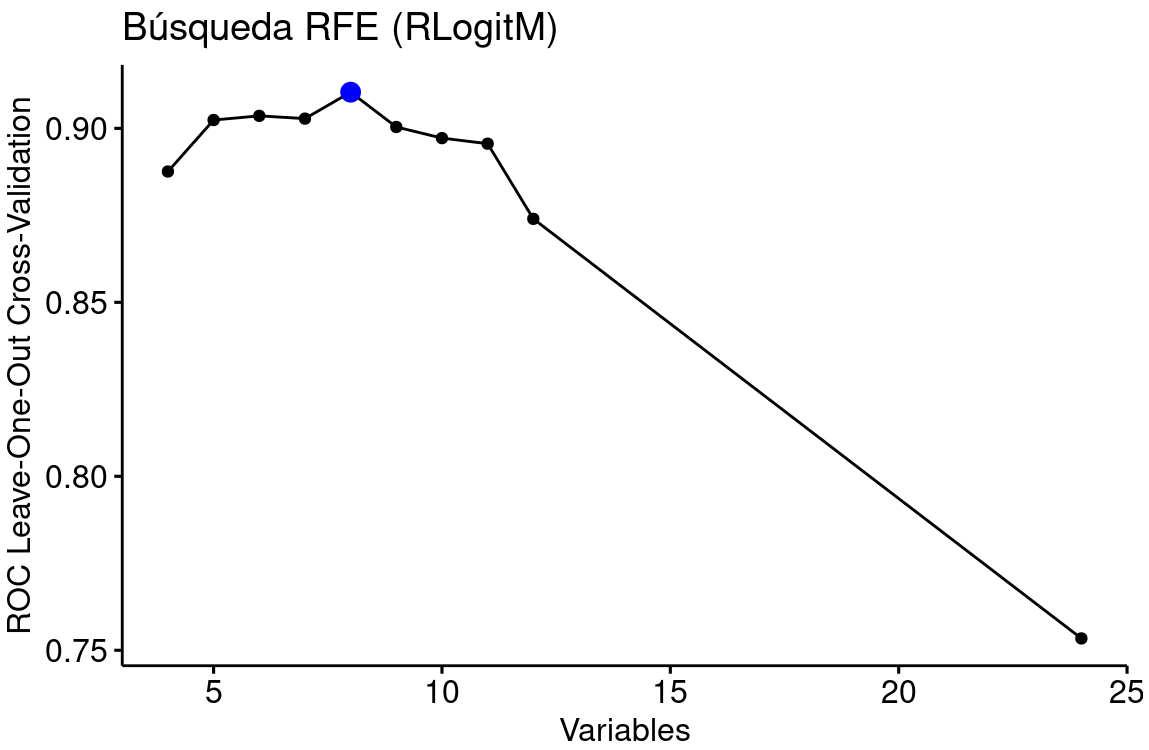
Notemos que se suprimen los *warnings* puesto muchas combinaciones podrían tener problemas para converger y se nos llenaría la pantalla con estos mensajes.

```
rlogitm_df <- muestra_ext |> select(-all_of(respuesta_lineal))
rlogitm_fmla <- formula(paste(respuesta_binaria, ".", sep = " ~ "))

lrFuncs[["summary"]] <- twoClassSummary
rlogitm_rfe_control <- rfeControl(functions = lrFuncs, method = "LOOCV", saveDetails = TRUE, ret
urnResamp = "all", verbose = FALSE)
rlogitm_train_control <- trainControl(method = "none", classProbs = TRUE,
                                      summaryFunction = twoClassSummary)

set.seed(17 * 11111)
rlogitm_rfe <- suppressWarnings(
  rfe(rlogitm_fmla, data = rlogitm_df, sizes = 4:12, metric = "ROC",
      rfeControl = rlogitm_rfe_control, trControl = rlogitm_train_control)
)
rlogitm <- rlogitm_rfe[["fit"]]

cat("Modelo de RLogitM obtenido con RFE:\n")
print(summary(rlogitm))
```

```
Modelo de RLogitM obtenido con RFE:

Call:
glm(formula = Class ~ ., family = "binomial", data = tmp)

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -61.16686   14.20889  -4.305 1.67e-05 ***
Wrist.Minimum.Girth       1.08980    0.70420   1.548  0.12173
Ankle.Minimum.Girth      -0.70453    0.40231  -1.751  0.07991 .
Height                   -0.03355    0.06375  -0.526  0.59863
Bitrochanteric.diameter   0.19892    0.24482   0.813  0.41650
Ankles.diameter           1.74282    0.65897   2.645  0.00817 **
Forearm.Girth            -0.23535    0.34006  -0.692  0.48889
Shoulder.Girth            0.09991    0.07296   1.369  0.17087
Knee.Girth                0.80926    0.34578   2.340  0.01926 *
---
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:  56.589  on 91  degrees of freedom
AIC: 74.589

Number of Fisher Scoring iterations: 7
```

Podemos ver el proceso de búsqueda realizado por RFE.

```
rlogitm_rfe_p <- ggplot(rlogitm_rfe) + theme_pubr()
rlogitm_rfe_p <- ggpar(rlogitm_rfe_p, title = "Búsqueda RFE (RLogitM)")
print(rlogitm_rfe_p)
```



Observemos que usando la función `twoClassSummary()` para medir el rendimiento del modelo, la búsqueda de predictores intenta maximizar el área bajo la curva ROC obtenida.

Aprovechemos de notar que por la naturaleza de RFE, que intenta ir eliminar predictores, siempre se evalúa el modelo con todos los posibles predictores, que en este caso resulta con menor desempeño que usando 4 a 12 variables. Si bien los mensajes de *warnings* que se generan por dificultades de convergencia pueden ser molestos, en general esto no es problemático, a menos que este modelo inicial converja y obtenga el mejor resultado. En ese caso la función `rfe()` retorna este modelo y hay que *bucear* en las opciones y el objeto que retorna para recuperar algún modelo con el tamaño solicitado.

Multicolinealidad

Revisemos los niveles de multicolinealidad del modelo inicial.

```
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:
    Wrist.Minimum.Girth     Ankle.Minimum.Girth                  Height
               3.953721                3.099090                1.981193
Bitrochanteric.diameter         Ankles.diameter           Forearm.Girth
               1.363920                1.679955                5.003949
         Shoulder.Girth              Knee.Girth
               2.936470                3.089147

Valores de tolerancia:
    Wrist.Minimum.Girth     Ankle.Minimum.Girth                  Height
              0.2529263               0.3226754               0.5047464
Bitrochanteric.diameter         Ankles.diameter           Forearm.Girth
              0.7331810               0.5952541               0.1998421
         Shoulder.Girth              Knee.Girth
              0.3405450               0.3237139
```

Apreciamos que solo la variable `Forearm.Girth` muestra valores de inflación de la varianza preocupantes, por lo que procedemos a sacarla del modelo.

```
rlogitm_seleccion <- predictors(rlogitm)[-6]
rlogitm_sel_text <- paste(rlogitm_seleccion, collapse = " + ")
rlogitm_fmla <- formula(paste(respuesta_binaria, rlogitm_sel_text, sep = " ~ "))
rlogitm_train_control <- trainControl(method = "LOOCV", classProbs = TRUE,
                                      summaryFunction = twoClassSummary)

set.seed(17 * 11111)
rlogitm_train <- train(rlogitm_fmla, data = rlogitm_df, method = "glm", metric = "ROC",
                       trControl = rlogitm_train_control)
rlogitm <- rlogitm_train[["finalModel"]]

cat("Nuevos factores de inflación de la varianza:\n")
print(vif(rlogitm))
cat("\n")
cat("Nuevos valores de tolerancia:\n")
print(1 / vif(rlogitm))
```

```
Nuevos factores de inflación de la varianza:
    Wrist.Minimum.Girth     Ankle.Minimum.Girth                  Height
               2.137769                2.841019                1.818548
Bitrochanteric.diameter         Ankles.diameter          Shoulder.Girth
               1.345249                1.502785                1.901734
             Knee.Girth
               2.821926

Nuevos valores de tolerancia:
    Wrist.Minimum.Girth     Ankle.Minimum.Girth                  Height
              0.4677775               0.3519864               0.5498892
Bitrochanteric.diameter         Ankles.diameter          Shoulder.Girth
              0.7433570               0.6654311               0.5258359
             Knee.Girth
              0.3543679
```

Con esto hemos conseguido un modelo que incluye siete predictores con niveles de multicolinealidad aceptables.

Ajuste

Revisemos el modelo conseguido y realicemos una comparación con el modelo nulo usando la prueba de la razón de verosimilitud (y un modelo tradicional equivalente para que funcione con las funciones del paquete `car`).

```
rlogitm_equiv <- glm(rlogitm_fmla, data = rlogitm_df, family = binomial(link = "logit"))

rlogitm_nulo_fmla <- formula(paste(respuesta_binaria, "1", sep = " ~ "))
rlogitm_nulo <- glm(rlogitm_nulo_fmla, data = rlogitm_df, family = binomial(link = "logit"))

cat("Modelo de RLogitM con cinco predictores:\n")
print(summary(rlogitm))
cat("\n")
cat("Comparación con el modelo nulo:\n")
print(anova(rlogitm_nulo, rlogitm_equiv, test = "LRT"))
```

```
Modelo de RLogitM con cinco predictores:

Call:
NULL

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -58.80135   13.41116  -4.385 1.16e-05 ***
Wrist.Minimum.Girth       0.77293    0.51501   1.501  0.13340
Ankle.Minimum.Girth      -0.64039    0.38654  -1.657  0.09757 .
Height                   -0.02186    0.06135  -0.356  0.72163
Bitrochanteric.diameter   0.19583    0.24499   0.799  0.42409
Ankles.diameter           1.61034    0.61120   2.635  0.00842 **
Shoulder.Girth            0.07036    0.05881   1.196  0.23152
Knee.Girth                0.76333    0.32939   2.317  0.02048 *
---
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:  57.078  on 92  degrees of freedom
AIC: 73.078

Number of Fisher Scoring iterations: 7


Comparación con el modelo nulo:
Analysis of Deviance Table

Model 1: TRG ~ 1
Model 2: TRG ~ Wrist.Minimum.Girth + Ankle.Minimum.Girth + Height + Bitrochanteric.diameter +
    Ankles.diameter + Shoulder.Girth + Knee.Girth
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1        99    138.629
2        92     57.078  7   81.552 6.644e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

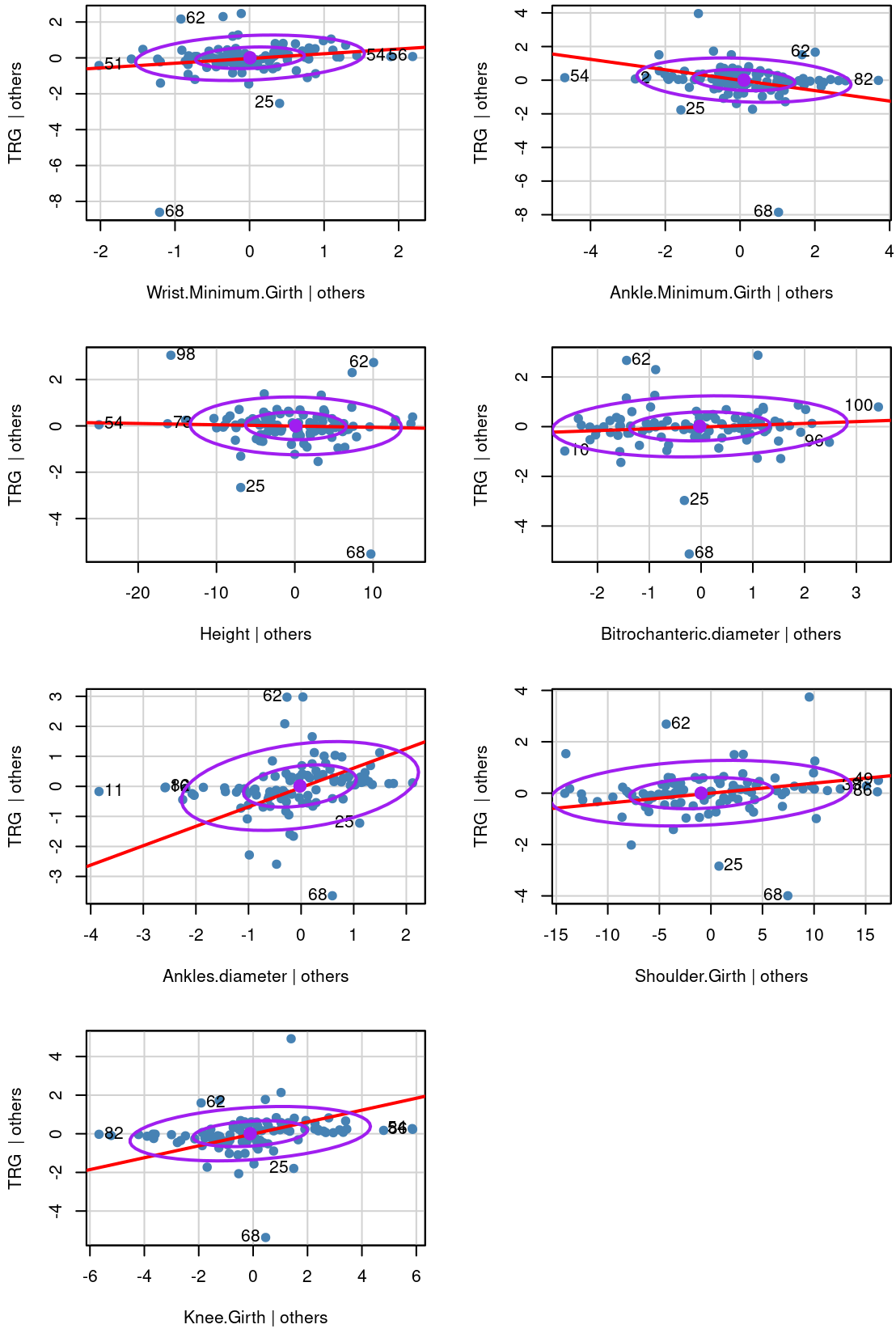
Observamos que el modelo consigue una reducción importante y significativa de la devianza ($\chi^2(6) = 81{,}426, p < 0.001$) respecto del modelo nulo.

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` . En este ocasión también marcaremos, con elipses, las nubes de puntos con 50% y 95% de los casos para que nos ayuden a identificar casos influyentes.

```
avPlots(rlogitm_equiv, layout = c(4, 2),
        col = "steelblue", pch = 20, cex = 1.5, lty = 2, col.lines = "red",
        main = "Regresiones parciales",
        id = list(n = 3, cex = 1, location = "lr"),
        ellipse = list(levels=c(0.50, 0.95), col = "purple"))
```

Regresiones parciales



En estos gráficos podemos observar características relevantes. Primero, que todas las relaciones de entre los predictores y la variable de salida transformada parecen lineales, sin que se vean patrones que se podrían atribuir a relaciones de otro tipo. Segundo, que la nube central de puntos domina el ajuste de las rectas de regresión parciales, que no parecen estar afectadas por los pocos valores que se alejan hacia los extremos (que tenderían a hacerlas más horizontales). Por último, es claro que las relaciones de la respuesta con la estatura (`Height`) y el

diámetro bitrocantérico (`Bitrochanteric.diameter`) son prácticamente nulas, mientras que con el grosor mínimo de las muñecas (`Wrist.Minimum.Girth`) y el grosor a la altura de los hombros (`Shoulder.Girth`) también se ven bastante débiles. Vemos que el ajuste es muy bueno, con alguna desviación en los valores extremos del predictor `Ankle.Minimum.Girth` , pero que no parece importante. Recordemos que el último subgráfico representa la distribución condicional de la variable respuesta dado el modelo ajustado. Vemos que esta estimación también es de muy buena calidad.

En consecuencia, y dado que se nos pide un modelo con al menos cinco predictores, es mejor que quitemos, uno a uno, los que contribuyen menos al ajuste del modelo, comenzando con `Height` ($t(92) = -0{,}356$).

```
rlogitm_seleccion <- rlogitm_seleccion[-3]
rlogitm_sel_text <- paste(rlogitm_seleccion, collapse = " + ")
rlogitm_fmla <- formula(paste(respuesta_binaria, rlogitm_sel_text, sep = " ~ "))
rlogitm_train_control <- trainControl(method = "LOOCV", classProbs = TRUE,
                                      summaryFunction = twoClassSummary)

set.seed(17 * 11111)
rlogitm_train <- train(rlogitm_fmla, data = rlogitm_df, method = "glm", metric = "ROC",
                       trControl = rlogitm_train_control)
rlogitm <- rlogitm_train[["finalModel"]]

cat("Nuevo modelo de RLogitM:\n")
print(summary(rlogitm))
```

```
Nuevo modelo de RLogitM:

Call:
NULL

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -59.97019   13.10765  -4.575 4.76e-06 ***
Wrist.Minimum.Girth       0.72259    0.49504   1.460  0.14438
Ankle.Minimum.Girth      -0.57509    0.33619  -1.711  0.08716 .
Bitrochanteric.diameter   0.18262    0.24214   0.754  0.45073
Ankles.diameter           1.56872    0.59377   2.642  0.00824 **
Shoulder.Girth            0.06306    0.05497   1.147  0.25124
Knee.Girth                0.72447    0.30492   2.376  0.01751 *
---
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:  57.203  on 93  degrees of freedom
AIC: 71.203

Number of Fisher Scoring iterations: 6
```

La variable `Bitrochanteric.diameter` sigue siendo la que menos contribuye al ajuste del modelo ($t(92) = -0{,}356$), por lo que procedemos a eliminarla.

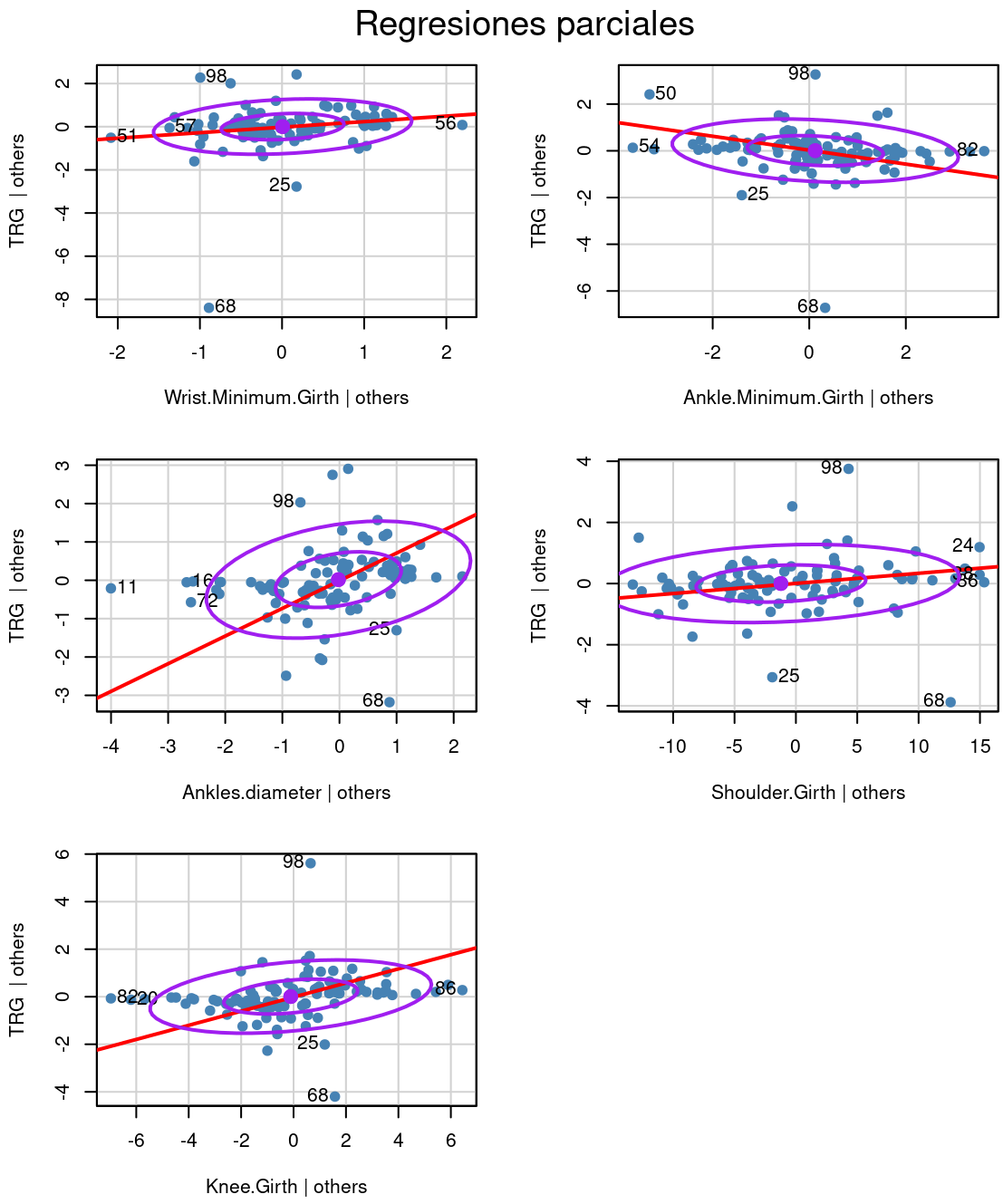
```
rlogitm_seleccion <- rlogitm_seleccion[-3]
rlogitm_sel_text <- paste(rlogitm_seleccion, collapse = " + ")
rlogitm_fmla <- formula(paste(respuesta_binaria, rlogitm_sel_text, sep = " ~ "))
rlogitm_train_control <- trainControl(method = "LOOCV", classProbs = TRUE,
                                      summaryFunction = twoClassSummary)

set.seed(17 * 11111)
rlogitm_train <- train(rlogitm_fmla, data = rlogitm_df, method = "glm", metric = "ROC",
                       trControl = rlogitm_train_control)
rlogitm <- rlogitm_train[["finalModel"]]

cat("Nuevo modelo de RLogitM con 5 predictores:\n")
print(summary(rlogitm))
```

```
Nuevo modelo de RLogitM con 5 predictores:

Call:
NULL

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         -58.29107   12.89091  -4.522 6.13e-06 ***
Wrist.Minimum.Girth   0.75119    0.49568   1.515  0.12965
Ankle.Minimum.Girth  -0.58747    0.33055  -1.777  0.07553 .
Ankles.diameter       1.63111    0.58798   2.774  0.00554 **
Shoulder.Girth        0.06042    0.05391   1.121  0.26241
Knee.Girth            0.81847    0.28271   2.895  0.00379 **
---
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:  57.782  on 94  degrees of freedom
AIC: 69.782

Number of Fisher Scoring iterations: 6
```

Vemos que este modelo más simple consigue prácticamente la misma reducción de desviación que el modelo con dos predictores extras: 57,782 vs. 57,078 respectivamente.

Hagamos una revisión rápida que todo va bien con este modelo.

```
rlogitm_equiv <- glm(rlogitm_fmla, data = rlogitm_df, family = binomial(link = "logit"))

cat("Nuevos factores de inflación de la varianza:\n")
print(vif(rlogitm))

avPlots(rlogitm_equiv, layout = c(3, 2),
        col = "steelblue", pch = 20, cex = 1.5, lty = 2, col.lines = "red",
        main = "Regresiones parciales",
        id = list(n = 3, cex = 1, location = "lr"),
        ellipse = list(levels=c(0.50, 0.95), col = "purple"))
```



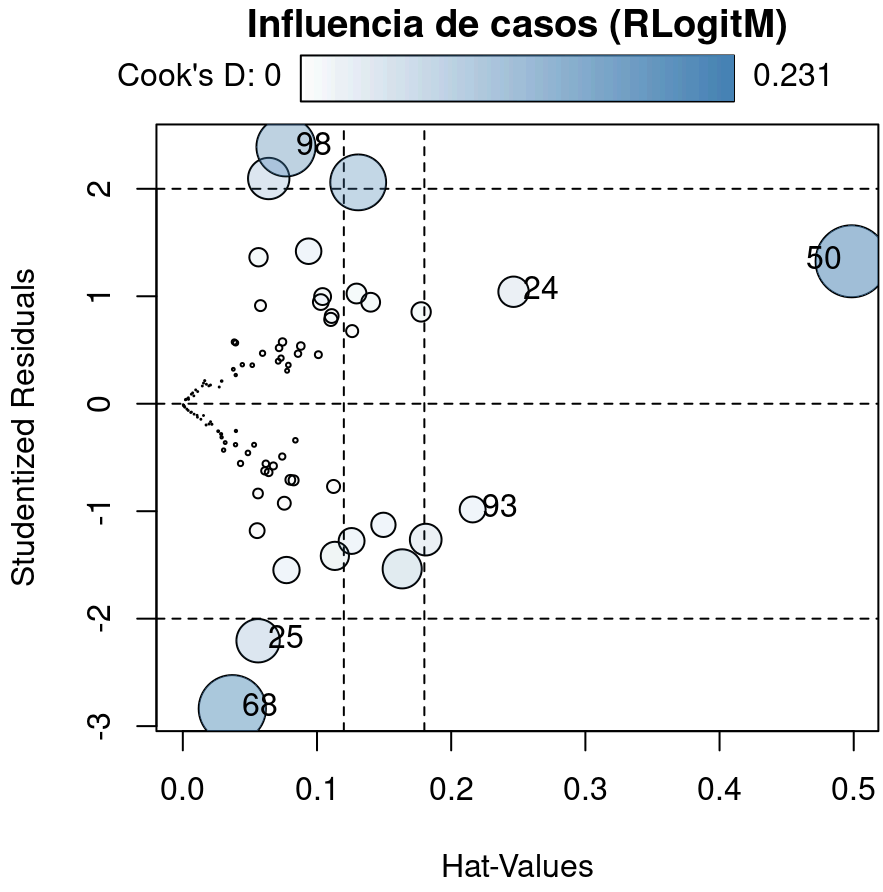
```
Nuevos factores de inflación de la varianza:
Wrist.Minimum.Girth Ankle.Minimum.Girth     Ankles.diameter      Shoulder.Girth
           1.932528            2.099071            1.395360            1.666613
         Knee.Girth
           2.092049
```

Si bien hay predictores que parecen irrelevantes, por las restricciones del enunciado no podemos quitar más variables y detenemos este proceso aquí.

Casos sobreinfluyentes

Confirmemos que no hay casos con sobre influencia en el modelo.

```
rlogitm_inf_estad <- influencePlot(rlogitm_equiv, , fill.col = "steelblue",
                                   scale = 5, id = list(n = 3),
                                   main = "Influencia de casos (RLogitM)\n")
```



```
cat("Límites para el modelo de RLogitM:\n")
cat("Rango para 95% de los residuos studentizados: ")
cat("[", round(qt(0.05/2, nrow(rlogitm_df) - length(predictors(rlogitm)) - 2), 3), ", ", sep =
"")
cat(round(qt(1-0.05/2, nrow(rlogitm_df) - length(predictors(rlogitm)) - 2), 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("\nCasos notorios para el modelo de RLogitM:\n")
print(rlogitm_inf_estad)
```

```
Límites para el modelo de RLogitM:
Rango para 95% de los residuos studentizados: [-1.986, 1.986]
Límite del apalancamiento: 0.12
Límite de la distancia de Cook: 0.042

Casos notorios para el modelo de RLogitM:
      StudRes        Hat      CookD
24  1.0418291 0.24647971 0.04109543
25 -2.2054816 0.05609974 0.08374029
50  1.3252436 0.49842024 0.23104555
68 -2.8377327 0.03687325 0.20109492
93 -0.9839328 0.21600404 0.02999544
98  2.3895710 0.07685464 0.15498576
```

Observamos que el residuo 98 esta fuera de rango en los tres criterios, pero que sin embargo no parece desviar ninguna de las rectas de regresión parciales. Algo similar ocurre con el caso 68. Los casos 62 y 93, ni siquiera aparecen destacados en las regresiones parciales.

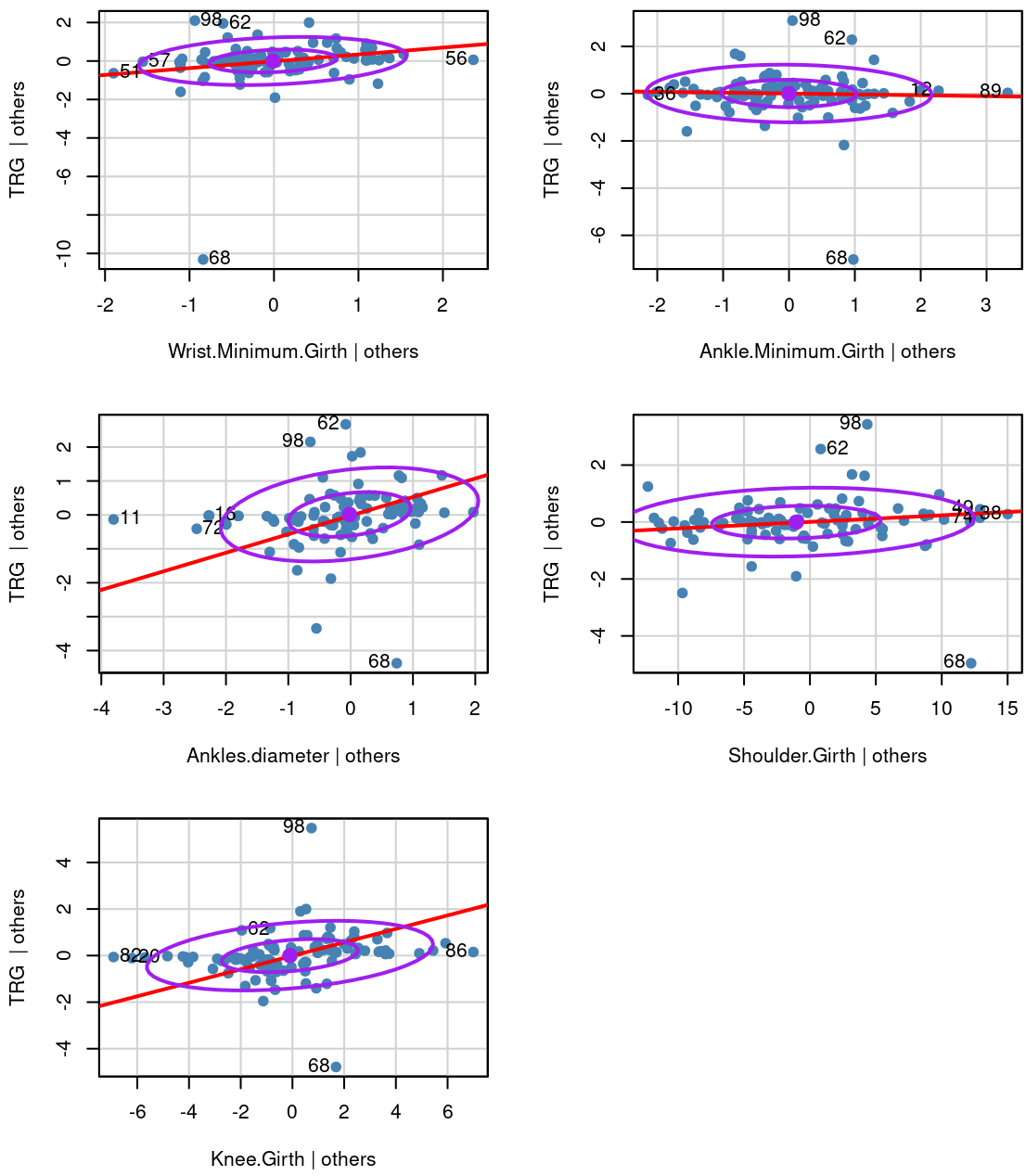
Sin embargo, el caso 50 podría estar tirando la pendiente asociada al grosor mínimo de los tobillos (`Ankle.Minimum.Girth`) hacia valores negativos; mientras que el caso 24 podría estar aumentando espuriamente la pendiente asociada al grosor a la altura de los hombros (`Shoulder.Girth`). Es poco probable que estos dos casos dominen el ajuste del modelo, pero para hacer el ejercicio interesante, procedemos a eliminarlos.

```
rlogitm_df_2 <- rlogitm_df[-c(24, 50), ]

set.seed(17 * 11111)
rlogitm_train_2 <- train(rlogitm_fmla, data = rlogitm_df_2, method = "glm", metric = "ROC",
                         trControl = rlogitm_train_control)
rlogitm_2 <- rlogitm_train_2[["finalModel"]]

cat("Modelo de RLogitM actualizado\n")
print(summary(rlogitm_2))
```

```
Modelo de RLogitM actualizado

Call:
NULL

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         -63.22329   14.27889  -4.428 9.52e-06 ***
Wrist.Minimum.Girth   0.88049    0.54221   1.624   0.1044
Ankle.Minimum.Girth  -0.19549    0.41141  -0.475   0.6347
Ankles.diameter       1.34676    0.58924   2.286   0.0223 *
Shoulder.Girth        0.04292    0.05964   0.720   0.4717
Knee.Girth            0.81627    0.28438   2.870   0.0041 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 135.816  on 97  degrees of freedom
Residual deviance:  54.348  on 92  degrees of freedom
AIC: 66.348

Number of Fisher Scoring iterations: 7
```

Claramente los coeficientes para estos predictores estaban inflados, y ahora resulta más evidente que no aportan al ajuste del modelo. Hagamos una revisión rápida que todo va bien con este modelo.

```
rlogitm_equiv_2 <- glm(rlogitm_fmla, data = rlogitm_df_2, family = binomial(link = "logit"))

cat("Nuevos factores de inflación de la varianza:\n")
print(vif(rlogitm_2))

avPlots(rlogitm_equiv_2, layout = c(3, 2),
        col = "steelblue", pch = 20, cex = 1.5, lty = 2, col.lines = "red",
        main = "Regresiones parciales",
        id = list(n = 3, cex = 1, location = "lr"),
        ellipse = list(levels=c(0.50, 0.95), col = "purple"))
```




```
Nuevos factores de inflación de la varianza:
Wrist.Minimum.Girth Ankle.Minimum.Girth     Ankles.diameter      Shoulder.Girth
           2.124653            1.352894            1.404387            1.868007
         Knee.Girth
           1.405886
```

Se deja como ejercicio revisar si no han aparecido otros casos con sobreinfluencia para este nuevo modelo de RLogitM.

Independencia de los residuos

Confirmemos que el modelo de RLogitM conseguido no genera dependencia en los residuos.

```
cat("Prueba de la independencia de los residuos para el modelo de RLogitM:\n")
print(durbinWatsonTest(rlogitm_2))
```

```
Prueba de la independencia de los residuos para el modelo de RLogitM:
 lag Autocorrelation D-W Statistic p-value
   1        0.109564      1.762728   0.208
 Alternative hypothesis: rho != 0
```

Vemos que no hay razones para rechazar la independencia de los residuos de este modelo.

Desempeño

Recordemos que el método de de validación cruzada dejando uno fuera evalúa solo una observación en cada iteración. Por lo tanto, al concluir las iteraciones, solo tiene una tabla de confusión de donde calcular las métricas de desempeño, es decir, no hay varias estimaciones del rendimiento del modelo como teníamos en las preguntas anteriores. Podemos conocer el desempeño del modelo de forma directa.

```
cat("Rendimiento del modelo de RLogitM actualizado:\n")
print(rlogitm_train_2[["results"]][, 2:4], digits = 2)
```

```
Rendimiento del modelo de RLogitM actualizado:
   ROC Sens Spec
1 0.92 0.86  0.9
```

Vemos que el modelo obtenido tiene un rendimiento relativamente bueno, con un área bajo la curva ROC de $0{,}92$ (sensibilidad = $0{,}86$, especificidad = $0{,}90$).

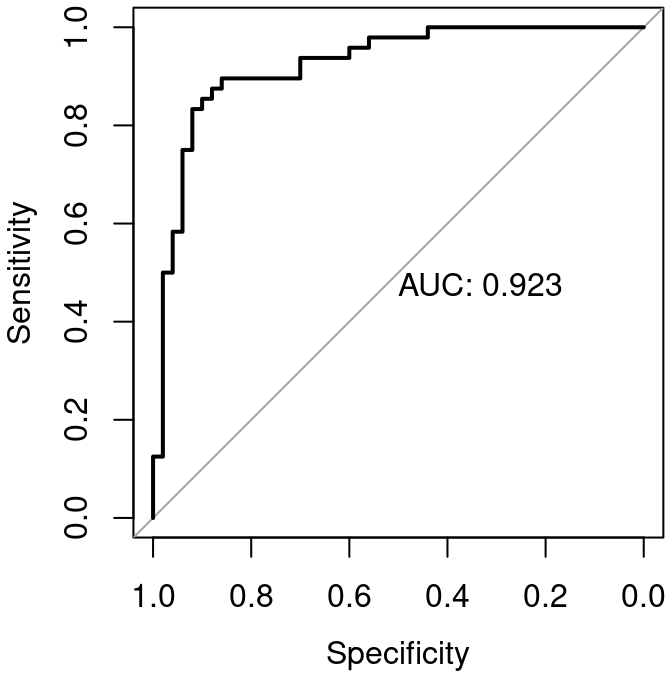
Por supuesto podemos tener más detalles de estos resultados mirando, por ejemplo, la matriz de confusión resultante.

```
rlogitm_mat_conf <- confusionMatrix(rlogitm_train_2[["pred"]][["pred"]],
                                    rlogitm_train_2[["pred"]][["obs"]])

cat("Matriz de confusión del modelo de RLogitM:\n")
print(rlogitm_mat_conf)
```

```
Matriz de confusión del modelo de RLogitM:
Confusion Matrix and Statistics

          Reference
Prediction no sí
        no 43  5
        sí  7 43

               Accuracy : 0.8776
                 95% CI : (0.7959, 0.9351)
    No Information Rate : 0.5102
    P-Value [Acc > NIR] : 1.325e-14

                  Kappa : 0.7552

 Mcnemar's Test P-Value : 0.7728

            Sensitivity : 0.8600
            Specificity : 0.8958
         Pos Pred Value : 0.8958
         Neg Pred Value : 0.8600
             Prevalence : 0.5102
         Detection Rate : 0.4388
   Detection Prevalence : 0.4898
      Balanced Accuracy : 0.8779

       'Positive' Class : no
```

También podemos obtener una gráfica de la curva ROC conseguida.

```
rlogitm_2_roc <-roc(rlogitm_train_2[["pred"]][["obs"]],
                    rlogitm_train_2[["pred"]][["sí"]],
                    direction = "<", levels=c("no", "sí"))
plot(rlogitm_2_roc, print.auc = TRUE)
```



Conclusión

La instrucción 6 nos solicita que nos pronunciarse sobre la confiabilidad y la calidad predictiva de los modelos obtenidos. Veamos.

Los tres modelos son confiables en términos de ajuste, generando residuos sin patrones y sin indicios de falta de independencia o que no se cumpla la linealidad de las relaciones entre predictores y la variable de respuesta. En el caso de los modelos de RLM, además, no se halló evidencia para dudar que se cumple la normalidad y homocedasticidad de los residuos. Además, los tres modelos consiguen niveles aceptables de multicolinealidad.

Sin embargo, los tres modelos incluyeron predictores que no apotaban al buen ajuste alcanzado, en especial los modelos obtenidos con RFE. También fue necesario eliminar un par de observaciones con demasiada influencia que alteraba de forma indebida los coeficientes del modelo de RLogitM.

Los modelos de RLM consiguieron una calidad predictiva relativamente buena, aunque el modelo obtenido con RFE exhibe mayor error ($0{,}859 \pm 0{,}146$ cm) que el modelo obtenido con el método de todos los subconjuntos ($0{,}689 \pm 0{,}080$ cm), aunque el primero fue evaluado en 25 conjuntos de datos mientras que el segundo en casi 2.000, por lo que esta comparación no es completamente definitiva.

El modelo de RLogitM consiguió una muy buena calidad predictiva para detectar rodillas gruesas, alcanzando un área bajo la curva ROC sobre $0{,}92$ estimada con validación cruzada dejando uno fuera.

Declaración importante

Es importante notar que no hemos sido *atarantados* al minuto de remover datos al construir los modelos. De hecho, uno **no elimina simultáneamente** todos los casos sospechosos. Metodológicamente, uno tendría que eliminar **un caso** sobreinfluyente solo si se llega a la conclusión de que se trata de un **dato erróneo** o de **una excepción** en la **población (no la muestra)** que no se debería incluir en un modelo que pretende describir un fenómeno general. Si no es un error, una excepción o no se busca un modelo que describa la mayoría de la población, entonces el dato **no debe ser eliminado**.

Además, luego de eliminar un dato, se **debe revisar** el efecto que esto tuvo en el modelo, cómo cambiaron los coeficientes y el ajuste, y volver a examinar si aparecen otros casos sobreinfluyentes. Por razones pedagógicas (evitar complejizar demasiado el ejemplo) no hemos seguido exactamente este procedimiento en este script.

Es probable que en la vida laboral, algún "jefe/a" nos pida "*quitar algunos datitos*" (¡o variables!) de un modelo. Manipular los datos para conseguir un modelo que confirme lo que nos gustaría concluir es **profundamente antiético** y ningún profesional, menos uno de la Universidad de Santiago de Chile, debería cometer este tal acto deshonesto.

Por supuesto, como todo dilema moral, esto es más fácil decirlo que hacerlo cuando la estabilidad laboral está en juego. Cada estudiante debe prepararse para estas situaciones, aprovechando al máximo las instancias y asignaturas que apuntan a desarrollar y mejorar sus habilidades personales (y que a veces desatendemos por no comprender la relevancia que tienen).

Referencias

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