PAC 2 Regresión Lineal

Maria Lucas

2023-06-22

Table of Contents

# Ejercicio 1

# Set viasulization number  
options(scipen = 999)  
  
# Set the file path  
file\_path <- "D:/Antiguos estudios/MASTER2/Sem2/Regresion/PAC2/P2/pancreas\_biomarkers.txt"  
  
# Load the data into a data frame  
data <- read.table(file\_path, header = TRUE, sep = "\t")  
  
# Display the first few rows of the data frame  
head(data)

## sample\_id sample\_origin age age\_cat sex diagnosis stage  
## 1 S1 BPTB 33 26-35 F 1   
## 2 S10 BPTB 81 75+ F 1   
## 3 S100 BPTB 51 46-55 M 1   
## 4 S101 BPTB 61 56-65 M 1   
## 5 S102 BPTB 62 56-65 M 1   
## 6 S103 BPTB 53 46-55 M 1   
## benign\_sample\_diagnosis creatinine LYVE1 REG1B TFF1  
## 1 1.83222 0.89321920 52.94884 654.2822  
## 2 0.97266 2.03758500 94.46703 209.4882  
## 3 0.78039 0.14558890 102.36600 461.1410  
## 4 0.70122 0.00280488 60.57900 142.9500  
## 5 0.21489 0.00085956 65.54000 41.0880  
## 6 0.84825 0.00339300 62.12600 59.7930

# Saving variables as factors  
data$diagnosis <- factor(data$diagnosis, levels = 1:3)  
data$age\_cat <- factor(data$age\_cat)

### (a) Modelo de regresión logística

#### Diagnóstico de todos los casos

model <- glm(diagnosis ~ age\_cat + creatinine + LYVE1 + REG1B + TFF1, data = data, family = binomial)

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

summary(model)

##   
## Call:  
## glm(formula = diagnosis ~ age\_cat + creatinine + LYVE1 + REG1B +   
## TFF1, family = binomial, data = data)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.27174931 0.47650815 0.570 0.568479   
## age\_cat36-45 0.27524151 0.52389015 0.525 0.599319   
## age\_cat46-55 -0.17244569 0.49309008 -0.350 0.726545   
## age\_cat56-65 -0.57047219 0.49120108 -1.161 0.245487   
## age\_cat66-75 0.34683938 0.51381181 0.675 0.499655   
## age\_cat75+ 0.00001974 0.59791940 0.000 0.999974   
## creatinine -0.84736599 0.21311812 -3.976 0.000070073 \*\*\*  
## LYVE1 0.21942153 0.06138360 3.575 0.000351 \*\*\*  
## REG1B 0.00069921 0.00143069 0.489 0.625041   
## TFF1 0.00251348 0.00050567 4.971 0.000000668 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 730.70 on 589 degrees of freedom  
## Residual deviance: 567.16 on 580 degrees of freedom  
## AIC: 587.16  
##   
## Number of Fisher Scoring iterations: 7

Tal y como podemos ver en los resultados, tan solo la creatinina, la LYVE1 y el TFF1 son variables que permiten predecir el riesgo de adenocarcinoma ductal pancreático. Se puede observar porque en todas ellas el pvalor es menos a 0.01. En el contexto de regresión logística un pvalor de 0.01 permite rechazar la hipótesis nula de no relación entre la variable predictora y la variable respuesta. En otras palabras, la variable tiene un impacto significativo sobre la clase.

#### Sólo adenocarcinoma y otro

# Subset the data to include only levels 2 and 3 of the "diagnosis" variable  
subset\_data <- subset(data, diagnosis != 1)  
  
# Recode values 2 and 3 to 0 and 1, respectively  
subset\_data$diagnosis <- ifelse(subset\_data$diagnosis == 2, 0, 1)  
  
# Fit the logistic regression model using the subsetted data  
model <- glm(diagnosis ~ age\_cat + creatinine + LYVE1 + REG1B + TFF1, data = subset\_data, family = binomial)  
  
# View the model summary  
summary(model)

##   
## Call:  
## glm(formula = diagnosis ~ age\_cat + creatinine + LYVE1 + REG1B +   
## TFF1, family = binomial, data = subset\_data)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.66048523 1.18806983 -3.081 0.00206 \*\*   
## age\_cat36-45 1.15688820 1.22969696 0.941 0.34681   
## age\_cat46-55 1.66227729 1.18000260 1.409 0.15892   
## age\_cat56-65 3.03244805 1.16788522 2.597 0.00942 \*\*   
## age\_cat66-75 2.70033855 1.16861309 2.311 0.02085 \*   
## age\_cat75+ 3.44294729 1.21523972 2.833 0.00461 \*\*   
## creatinine -0.30213039 0.22625232 -1.335 0.18176   
## LYVE1 0.31400141 0.05280717 5.946 0.00000000274 \*\*\*  
## REG1B 0.00280367 0.00109942 2.550 0.01077 \*   
## TFF1 -0.00005978 0.00021201 -0.282 0.77798   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 564.02 on 406 degrees of freedom  
## Residual deviance: 382.52 on 397 degrees of freedom  
## AIC: 402.52  
##   
## Number of Fisher Scoring iterations: 5

Tal y como podemos ver en los resultados, la edad a partir de 56 años es un indicador significativo. La LYVE1 y el REG1B también son variables que permiten predecir el riesgo de adenocarcinoma ductal pancreático. Se puede observar porque en todas ellas el pvalor es menos a 0.05. En el contexto de regresión logística un pvalor de 0.01 permite rechazar la hipótesis nula de no relación entre la variable predictora y la variable respuesta. En otras palabras, la variable tiene un impacto significativo sobre la clase.

Que el intercepto también sea significativo sugiere que parte de la variable respuesta no es explicada por las variables independientes estudiadas. En regresión logística el intercepto captura la probabilidad de ocurrencia de un suceso cuando todas las variables predictoras están en su nivel de referencia. En otras palabras, un intercepto significativo implica que aunque no tengamos ningun predictor, hay diferencias significativas entre las clases.

### (b) Interpretación de coeficientes

Los coeficientes en un modelo de regresión logística indican el cambio estimado en el log-odds del evento ocurriendo (codeado como 1 = adenocarcinoma ductal pancreático) asociado a una unidad de cambio en la variable predictora, sin variar el resto de variables.

Un estimador positivo sugiere que el incremento de la variable está asociado a un incremento de la probabilidad (log-odds) del evento ocurriendo. Si el estimador es 0.5 significa que al aumentar en 1 el valor del predictor, esto se asopcia a un 0.5 aumento del log-odds del evento ocurriendo. Este es el caso del LYVE1, al aumentar LYVE1 es más probable tener adenocarcinoma (0.3140). Lo mismo sucede con la edad, parece ser que tener más de 56 años aumenta el log-odds de tener adenocacinoma. Lo hace de forma distinta dependiendo del rango de edad, de 56-65 (3.032), de 66-75 (2.7) y a partir de 75 años (3.443).

Contrariamente, un estimador negativo significa que un aumento de la variable disminuye el log-odds del evento ocurriendo (tener adenocarcinoma).

### (c) Modelo reducido

Para comparar ambos modelos podemos realizar un anova aplicando el test Chi cuadrado. Con esto estamos comparando el ajuste del modelo bajo las siguientes hipótesis:

* **H0:** Desviación del modelo reducido = Desviación del modelo completo. No hay diferencia de ajuste entre los modelos.
* **H1:** Desviación del modelo reducido > Desviación del modelo completo. El modelo reducido se ajusta peor que el modelo completo.

La desviación del modelo se refiere a la diferencia entre los valores reales de los datos y los valores predichos por el modelo.

# Fit the logistic regression model using the subsetted data  
model\_simple <- glm(diagnosis ~ age\_cat + LYVE1 + REG1B, data = subset\_data, family = binomial)  
  
summary(model\_simple)

##   
## Call:  
## glm(formula = diagnosis ~ age\_cat + LYVE1 + REG1B, family = binomial,   
## data = subset\_data)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.0311243 1.1897669 -3.388 0.000704 \*\*\*  
## age\_cat36-45 1.2607495 1.2590860 1.001 0.316672   
## age\_cat46-55 1.8352456 1.2038156 1.525 0.127378   
## age\_cat56-65 3.2169977 1.1924837 2.698 0.006981 \*\*   
## age\_cat66-75 2.8828093 1.1890190 2.425 0.015328 \*   
## age\_cat75+ 3.6627647 1.2328682 2.971 0.002969 \*\*   
## LYVE1 0.2924045 0.0495417 5.902 0.00000000359 \*\*\*  
## REG1B 0.0025692 0.0009132 2.813 0.004901 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 564.02 on 406 degrees of freedom  
## Residual deviance: 384.83 on 399 degrees of freedom  
## AIC: 400.83  
##   
## Number of Fisher Scoring iterations: 5

# Compare the two models  
anova(model, model\_simple, test = "Chi")

## Analysis of Deviance Table  
##   
## Model 1: diagnosis ~ age\_cat + creatinine + LYVE1 + REG1B + TFF1  
## Model 2: diagnosis ~ age\_cat + LYVE1 + REG1B  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)  
## 1 397 382.52   
## 2 399 384.83 -2 -2.312 0.3147

# Install and load the "lmtest" package  
# install.packages("lmtest")  
library(lmtest)

## Warning: package 'lmtest' was built under R version 4.3.1

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.3.1

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

# Perform the likelihood ratio test  
lrtest(model, model\_simple)

## Likelihood ratio test  
##   
## Model 1: diagnosis ~ age\_cat + creatinine + LYVE1 + REG1B + TFF1  
## Model 2: diagnosis ~ age\_cat + LYVE1 + REG1B  
## #Df LogLik Df Chisq Pr(>Chisq)  
## 1 10 -191.26   
## 2 8 -192.41 -2 2.312 0.3147

Dado que el pvalor no es significativo (0.31) aceptamos la hipótesis nula, el modelo reducido (sin creatinina y sin TFF1) es igual de bueno que el complejo. Adicionalmente, vemos que el AIC del modelo reducido es 2 unidades menor que el del modelo complejo, esto sugiere que el modelo reducido tiene un ajuste mejor teniendo en cuenta la complejidad de ambos modelos.

### (d) Funcion cuadrática

# Cuadratic LYVE1  
model\_simple\_LYVE1 = glm(diagnosis ~ age\_cat + LYVE1 + I(LYVE1^2) + REG1B, data = subset\_data, family = binomial)  
  
summary(model\_simple\_LYVE1)

##   
## Call:  
## glm(formula = diagnosis ~ age\_cat + LYVE1 + I(LYVE1^2) + REG1B,   
## family = binomial, data = subset\_data)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.2017396 1.2069541 -3.481 0.000499 \*\*\*  
## age\_cat36-45 1.3193891 1.2661132 1.042 0.297375   
## age\_cat46-55 1.8948671 1.2104349 1.565 0.117479   
## age\_cat56-65 3.2671843 1.1997369 2.723 0.006464 \*\*   
## age\_cat66-75 2.9251655 1.1953280 2.447 0.014398 \*   
## age\_cat75+ 3.7143104 1.2402500 2.995 0.002746 \*\*   
## LYVE1 0.3878762 0.1016950 3.814 0.000137 \*\*\*  
## I(LYVE1^2) -0.0104317 0.0090294 -1.155 0.247965   
## REG1B 0.0025524 0.0009032 2.826 0.004715 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 564.02 on 406 degrees of freedom  
## Residual deviance: 383.93 on 398 degrees of freedom  
## AIC: 401.93  
##   
## Number of Fisher Scoring iterations: 5

# Compare the two models  
anova(model\_simple, model\_simple\_LYVE1, test = "Chi")

## Analysis of Deviance Table  
##   
## Model 1: diagnosis ~ age\_cat + LYVE1 + REG1B  
## Model 2: diagnosis ~ age\_cat + LYVE1 + I(LYVE1^2) + REG1B  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)  
## 1 399 384.83   
## 2 398 383.93 1 0.89906 0.343

# Cuadratic REG1B  
model\_simple\_REG1B = glm(diagnosis ~ age\_cat + LYVE1 + REG1B + I(REG1B^2), data = subset\_data, family = binomial)  
  
summary(model\_simple\_REG1B)

##   
## Call:  
## glm(formula = diagnosis ~ age\_cat + LYVE1 + REG1B + I(REG1B^2),   
## family = binomial, data = subset\_data)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.956734568 1.152437904 -3.433 0.000596 \*\*\*  
## age\_cat36-45 1.139264885 1.231123197 0.925 0.354765   
## age\_cat46-55 1.740771095 1.169448508 1.489 0.136608   
## age\_cat56-65 3.084592019 1.165415260 2.647 0.008126 \*\*   
## age\_cat66-75 2.757009540 1.160754339 2.375 0.017540 \*   
## age\_cat75+ 3.536822433 1.205876020 2.933 0.003357 \*\*   
## LYVE1 0.285412846 0.050292662 5.675 0.0000000139 \*\*\*  
## REG1B 0.003777035 0.001875858 2.013 0.044062 \*   
## I(REG1B^2) -0.000001654 0.000002161 -0.765 0.444018   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 564.02 on 406 degrees of freedom  
## Residual deviance: 384.31 on 398 degrees of freedom  
## AIC: 402.31  
##   
## Number of Fisher Scoring iterations: 5

# Compare the two models  
anova(model\_simple, model\_simple\_REG1B, test = "Chi")

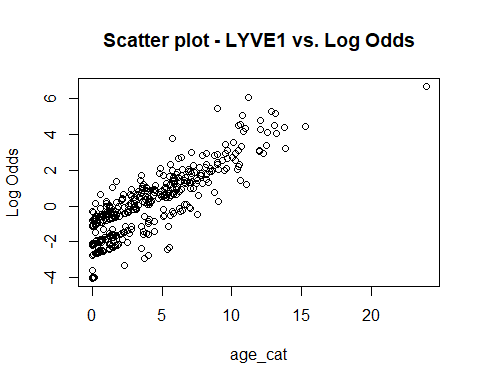
## Analysis of Deviance Table  
##   
## Model 1: diagnosis ~ age\_cat + LYVE1 + REG1B  
## Model 2: diagnosis ~ age\_cat + LYVE1 + REG1B + I(REG1B^2)  
## Resid. Df Resid. Dev Df Deviance Pr(>Chi)  
## 1 399 384.83   
## 2 398 384.31 1 0.52118 0.4703

En ninguno de los dos casos añadir el término cuadrático ha mejorado el ajuste del modelo. Siguiendo la explicación del apartado anterior: pv = 0.3 y 0.4, aceptamos H0 los dos modelos tienen el mismo ajuste.

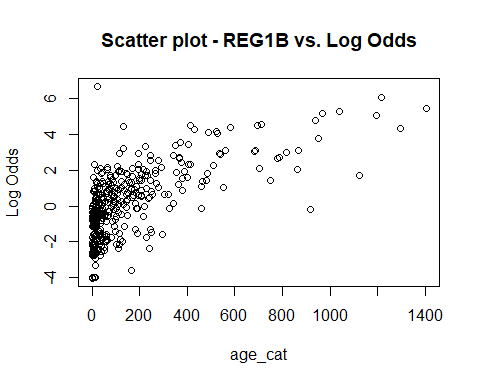
Adicionalmente, si examinamos el AIC veremos que en los modelos con el término cuadrático este es superior. Un valor mayor AIC sugiere que el modelo no se ajusta mejor para la complejidad que presenta.

En conclusión, no deberíamos incluir ninguno de los dos términos cuadráticos. Es importante comentar que, aunque estos tests aparezcan no-significativos, hay que valorarlos siempre junto con el contexto del análisis. En este caso, se ha realizado adicionalmente una inspección visual de la variable vs el log odds de la variable respuesta (disponible en el ANEXO). Estos gráficos nos confirman que efectivamente, existe una relación lineal, dado que se observa un patrón lineal sin observar otros patrones como curvas, forma de U, etc.

# Obtain predicted log odds from the model  
predicted\_logodds <- predict(model\_simple, type = "link")  
  
plot(subset\_data$LYVE1, predicted\_logodds, xlab = "age\_cat", ylab = "Log Odds", main = "Scatter plot - LYVE1 vs. Log Odds")



plot(subset\_data$REG1B, predicted\_logodds, xlab = "age\_cat", ylab = "Log Odds", main = "Scatter plot - REG1B vs. Log Odds")



### (e) Predicción de caso

# Create a new data frame for the new case  
new\_case = data.frame(age\_cat = "66-75", LYVE1 = 6, REG1B = 140)  
  
# Predict the outcome using the model  
prediction = predict(model\_simple, newdata = new\_case, type = "response")  
  
print(prediction)

## 1   
## 0.7242816

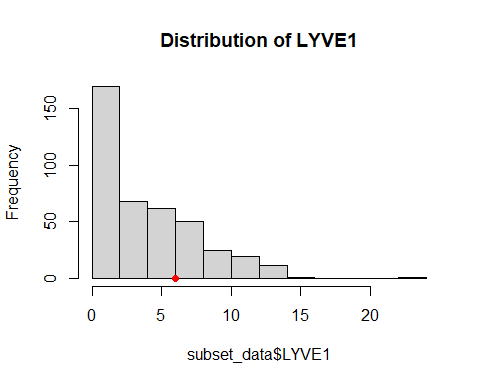
El modelo predice la presencia de adenocarcinoma ductal pancreático con un 72,42% de probabilidad. Recordemos que los valores fueron recodificados a 0 = afecciones pancreáticas no cancerosas y 1 = adenocarcinoma ductal pancreático.

La extrapolación ocurre cuando se hacen predicciones de para datos de la variable predictora fuera del rango de los datos usados para contruir el modelo.

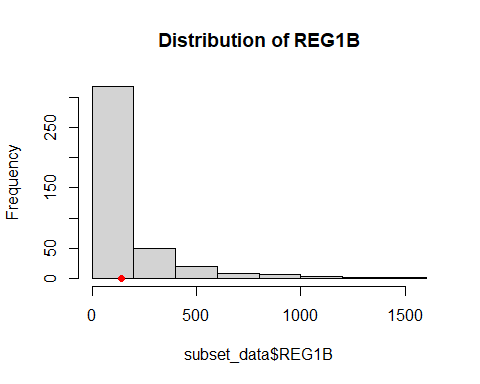
# Check predictor variable ranges  
range\_data <- sapply(subset\_data[, c("LYVE1", "REG1B")], range)  
range\_new\_observation <- c(6, 140) # Replace with the values of the new observation  
  
# Compare new observation values with observed range  
is\_extrapolation <- any(range\_new\_observation < range\_data[1, ] | range\_new\_observation > range\_data[2, ])  
  
# Print results  
cat("Extrapolation:", is\_extrapolation, "\n")

## Extrapolation: FALSE

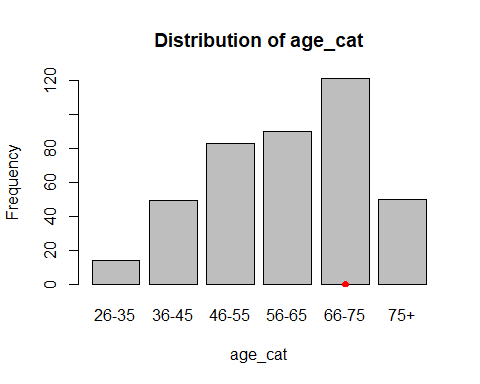
# Assess distribution of predictor variables  
# LYVE1  
hist(subset\_data$LYVE1, main = "Distribution of LYVE1")  
# Add new observation to LYVE1 plot  
points(6, 0, col = "red", pch = 16)



# REGB1  
hist(subset\_data$REG1B, main = "Distribution of REG1B")  
# Add new observation to REGB1 plot  
points(140, 0, col = "red", pch = 16)



# age\_cat  
barplot(table(subset\_data$age\_cat), main = "Distribution of age\_cat", xlab = "age\_cat", ylab = "Frequency")  
# Add new observation to age\_cat plot  
points(5.5, 0, col = "red", pch = 16)



Como podemos ver, los valores del caso a predecir se encuentra entre el rango utilizado para construir el modelo.

# Ejercicio 2

set.seed(123)  
  
# Data import  
# Note: I changed the variable names to avoid problems with symbols  
  
import.data <-  
"http://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/telemonitoring/parkinsons\_updrs.data"  
data <- read.table(url(import.data), sep=",", skip=1)  
names(data) <- c("subject#","age","sex","test\_time","motor\_UPDRS","total\_UPDRS",  
"Jitter\_p","Jitter\_Abs","Jitter\_RAP","Jitter\_PPQ5","Jitter\_DDP",  
"Shimmer","Shimmer\_dB","Shimmer\_APQ3","Shimmer\_APQ5","Shimmer\_APQ11", "Shimmer\_DDA","NHR","HNR","RPDE","DFA","PPE")  
  
# Select predictor variables and response  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

set.seed(123)  
parkinson <- data %>% dplyr::select(6:22)  
  
# Split the data into train and test  
library(caret)

## Warning: package 'caret' was built under R version 4.3.1

## Loading required package: ggplot2

## Loading required package: lattice

set.seed(123)  
train\_indices <- createDataPartition(parkinson$total\_UPDRS, p = 0.8, list = FALSE)  
train\_data <- parkinson[train\_indices, ]  
test\_data <- parkinson[-train\_indices, ]

### (a) Regresión lineal

Al no utilizar el factor “sujeto” no tenemos en cuenta las posibles variaciones de cada individuo. Las muestras deberían ser apareadas, ya que se ace un seguimiento a lo largo del tiempo. Una estimación. No sé lo que digo ya lo mirare.

# Fit the model  
model\_lineal = lm(total\_UPDRS ~ ., data = train\_data)  
  
# Extract R-squared  
r\_squared = summary(model\_lineal)$r.squared  
  
# Extract adjusted R-squared  
adj\_r\_squared = summary(model\_lineal)$adj.r.squared  
  
# Predict on the training data  
predictions\_train = predict(model\_lineal, train\_data)  
# Predict on the test data  
predictions\_test <- predict(model\_lineal, test\_data)  
  
# Calculate residuals train  
residuals\_train = train\_data$total\_UPDRS - predictions\_train  
# Calculate residuals test  
residuals\_test <- test\_data$total\_UPDRS - predictions\_test  
  
# Calculate RMSE train  
lineal\_rmse\_train = sqrt(mean(residuals\_train^2))  
# Calculate RMSE test  
lineal\_rmse\_test <- sqrt(mean(residuals\_test^2))  
  
# Create results table  
# Extract the variable names from the linear regression model  
variables <- names(coef(model\_lineal))[-1]  
  
# Create a data frame for the results  
results\_table <- data.frame(  
 Metric = c("Número de variables", "R-squared", "Adjusted R-squared", "RMSE\_train", "RMSE\_test"),  
 Value = c(length(variables), r\_squared, adj\_r\_squared, lineal\_rmse\_train, lineal\_rmse\_test)  
)  
  
# Print the result table  
print(results\_table)

## Metric Value  
## 1 Número de variables 16.0000000  
## 2 R-squared 0.1036576  
## 3 Adjusted R-squared 0.1005964  
## 4 RMSE\_train 10.0855940  
## 5 RMSE\_test 10.4046756

cat("Variables usadas en el modelo: ")

## Variables usadas en el modelo:

cat(variables, sep=", ")

## Jitter\_p, Jitter\_Abs, Jitter\_RAP, Jitter\_PPQ5, Jitter\_DDP, Shimmer, Shimmer\_dB, Shimmer\_APQ3, Shimmer\_APQ5, Shimmer\_APQ11, Shimmer\_DDA, NHR, HNR, RPDE, DFA, PPE

# Robust RMSE for exercise 3  
# Number of observations to trim  
n\_trim <- round(0.1 \* length(residuals\_test))  
# Sort the residuals in ascending order  
sorted\_residuals\_test <- sort(residuals\_test)  
# Trim the specified percentage of observations from both ends  
trimmed\_residuals\_test <- sorted\_residuals\_test[(n\_trim + 1):(length(residuals\_test) - n\_trim)]  
# Calculate the trimmed mean of the residuals  
trimmed\_mean\_test <- mean(trimmed\_residuals\_test)  
# Calculate the squared residuals using the trimmed mean  
trimmed\_squared\_residuals\_test <- (trimmed\_residuals\_test - trimmed\_mean\_test)^2  
# Calculate the robust RMSE using the trimmed mean  
robust\_lineal\_rmse\_test <- sqrt(mean(trimmed\_squared\_residuals\_test))

Al no considerar el “sujeto” se viola la suposición de independencia. En este tipo de modelos, se asume que todas las observaciones son independientes. Al no serlo, se construirá un modelo que incluirá métricas erróneas. De manera similar, se pierde precisión al no tener en cuenta la correlación entre los datos. El error estándar de los coeficientes estimados puede subestimarse, dando intervalos de confianza más estrechos.

Adicionalmente, se aumenta el error de Tipo 1 (rechazar incorrectamente H0 haciendo que una variable sea significativa cuando no lo es). Esto sucede porque los datos de un mismo individuo tienden a ser más similares, inflando así la significación de los resultados.

Finalmente, se hace más complicado detectar las variaciones entre en un mismo individuo.

### (b) Regresión lineal con AIC

library(MASS)

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

set.seed(123)  
  
# Perform stepwise variable selection based on AIC  
model\_stepwise <- stepAIC(model\_lineal, direction = "both", trace = FALSE)  
  
# Extract R-squared  
r\_squared = summary(model\_stepwise)$r.squared  
  
# Extract adjusted R-squared  
adj\_r\_squared = summary(model\_stepwise)$adj.r.squared  
  
# Predict on the training data  
predictions\_train = predict(model\_stepwise, train\_data)  
# Predict on the test data  
predictions\_test <- predict(model\_stepwise, test\_data)  
  
# Calculate residuals  
residuals\_train = train\_data$total\_UPDRS - predictions\_train  
# Calculate residuals  
residuals\_test <- test\_data$total\_UPDRS - predictions\_test  
  
# Calculate RMSE train  
step\_rmse\_train = sqrt(mean(residuals\_train^2))  
# Calculate RMSE test  
step\_rmse\_test <- sqrt(mean(residuals\_test^2))  
  
# Results  
# Extract the variable names from the linear regression model  
variables <- names(coef(model\_stepwise))[-1]  
  
# Create a data frame for the results  
results\_table <- data.frame(  
 Metric = c("Número de variables", "R-squared", "Adjusted R-squared", "RMSE\_train", "RMSE\_test"),  
 Value = c(length(variables), r\_squared, adj\_r\_squared, step\_rmse\_train, step\_rmse\_test)  
)  
  
# Print the result table  
print(results\_table)

## Metric Value  
## 1 Número de variables 11.0000000  
## 2 R-squared 0.1033935  
## 3 Adjusted R-squared 0.1012906  
## 4 RMSE\_train 10.0870795  
## 5 RMSE\_test 10.4071343

cat("Variables usadas en el modelo: ")

## Variables usadas en el modelo:

cat(variables, sep=", ")

## Jitter\_p, Jitter\_Abs, Shimmer, Shimmer\_APQ3, Shimmer\_APQ5, Shimmer\_APQ11, NHR, HNR, RPDE, DFA, PPE

# Robust RMSE  
# Number of observations to trim  
n\_trim <- round(0.1 \* length(residuals\_test))  
# Sort the residuals in ascending order  
sorted\_residuals\_test <- sort(residuals\_test)  
# Trim the specified percentage of observations from both ends  
trimmed\_residuals\_test <- sorted\_residuals\_test[(n\_trim + 1):(length(residuals\_test) - n\_trim)]  
# Calculate the trimmed mean of the residuals  
trimmed\_mean\_test <- mean(trimmed\_residuals\_test)  
# Calculate the squared residuals using the trimmed mean:  
trimmed\_squared\_residuals\_test <- (trimmed\_residuals\_test - trimmed\_mean\_test)^2  
# Calculate the robust RMSE using the trimmed mean:  
robust\_step\_rmse\_test <- sqrt(mean(trimmed\_squared\_residuals\_test))

Removing a variable solely based on its lack of significance may affect the overall fit of the model and the relationships between other variables. A variable that is not individually significant may contribute to the model’s overall predictive power when combined with other variables or interacted with other predictors. Therefore, it’s crucial to assess the model’s overall performance, such as through measures like R-squared or adjusted R-squared, and consider the context and theoretical significance of the variables

The significance of a variable can be influenced by multicollinearity, which occurs when predictor variables are highly correlated with each other. In the presence of multicollinearity, the individual coefficients and their significance can be unstable or misleading. It’s important to check for multicollinearity among the variables and consider its potential impact on the significance of individual predictors.

In this example, model\_linear is the initial linear regression model you built using all the variables of interest. The stepAIC() function from the MASS package is used to perform the stepwise variable selection based on AIC.

The direction argument specifies the direction of the stepwise procedure. “both” allows variables to be added or removed from the model. Other options include “backward” for variable removal only and “forward” for variable addition only.

### (c) Regresión por componentes principales

Performing regression using principal components involves transforming the predictor variables into a set of principal components and then using these components as predictors in the regression model.

# install.packages('pls')  
library(pls)

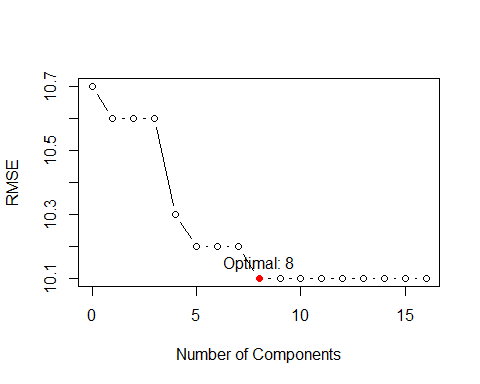
## Warning: package 'pls' was built under R version 4.3.1

##   
## Attaching package: 'pls'

## The following object is masked from 'package:caret':  
##   
## R2

## The following object is masked from 'package:stats':  
##   
## loadings

set.seed(123)  
  
PCA\_model <- pcr(total\_UPDRS ~ ., data = train\_data, validation="CV", scale = TRUE)  
  
# Calculate RMSE values  
mpcCV <- RMSEP(PCA\_model, estimate = "CV")  
rmse\_values <- round(mpcCV$val, 1) # Rounded to first decimal to account for complexity  
  
# Plot the graph  
plot(mpcCV$comp, rmse\_values, type = "b", xlab = "Number of Components", ylab = "RMSE")  
  
# Find the optimal number of components  
optimal\_components <- mpcCV$comp[which.min(rmse\_values)]  
points(optimal\_components, min(rmse\_values), col = "red", pch = 16)  
text(optimal\_components, min(rmse\_values), paste("Optimal:", optimal\_components), pos = 3)

 El minimo absoluto es 11 pero el minimo razonable sin hacerlo muy complejo es 8 -1 = 7

In the code you provided, the line (numpredcp <- which.min(mpcCVval). This line calculates the RMSEP for different numbers of components using cross-validation and then identifies the index of the minimum value using the which.min function.

The code performs principal component analysis (PCA) on the predictor variables using the prcomp() function. The resulting principal components (pcs) are then used to construct regression models with different numbers of components (ranging from 1 to the total number of components).

For each model, the code predicts the outcome variable on the test data, calculates the residuals, and computes the root mean squared error (RMSE). The RMSE values are stored in the rmse\_values vector.

The code then plots the RMSE values against the number of components to visualize the relationship. The number of components that yields the minimum RMSE is identified, and the corresponding results are printed.

Yes, in general, a lower RMSE indicates a better-fitting model. RMSE (Root Mean Squared Error) is a commonly used measure of the average prediction error of a regression model. It represents the square root of the average squared differences between the predicted values and the actual values of the outcome variable.

Since RMSE measures the magnitude of the prediction errors, a smaller RMSE implies that the model’s predictions are, on average, closer to the actual values.

# Predict on the training data using the model  
predictions\_train <- predict(PCA\_model, train\_data, ncomp = optimal\_components-1)  
# Predict on the test data using the model  
predictions\_test <- predict(PCA\_model, test\_data, ncomp = optimal\_components-1)  
   
# Calculate residuals for training and test data  
residuals\_train <- train\_data$total\_UPDRS - predictions\_train  
residuals\_test <- test\_data$total\_UPDRS - predictions\_test  
  
# Calculate R-squared for training data  
r2\_train <- 1 - sum(residuals\_train^2) / sum((train\_data$total\_UPDRS - mean(train\_data$total\_UPDRS))^2)  
  
# Calculate adjusted R-squared for training data  
n\_train <- nrow(train\_data)  
p\_train <- optimal\_components - 1 # Number of predictors (components) used  
r2\_adj\_train <- 1 - (1 - r2\_train) \* ((n\_train - 1) / (n\_train - p\_train - 1))  
   
# Calculate RMSE for training data  
PCA\_rmse\_train <- sqrt(mean(residuals\_train^2))  
# Calculate RMSE for test data  
PCA\_rmse\_test <- sqrt(mean(residuals\_test^2))  
  
# Create a data frame for the results  
results\_table <- data.frame(  
 Metric = c("Número de componentes", "R-squared", "Adjusted R-squared", "RMSE\_train", "RMSE\_test"),  
 Value = c((optimal\_components - 1), r2\_train, r2\_adj\_train, PCA\_rmse\_train, PCA\_rmse\_test)  
)  
  
# Print the result table  
print(results\_table)

## Metric Value  
## 1 Número de componentes 7.00000000  
## 2 R-squared 0.08079001  
## 3 Adjusted R-squared 0.07941922  
## 4 RMSE\_train 10.21343614  
## 5 RMSE\_test 10.55268400

cat("Variables usadas en el modelo: ")

## Variables usadas en el modelo:

cat(variables, sep=", ")

## Jitter\_p, Jitter\_Abs, Shimmer, Shimmer\_APQ3, Shimmer\_APQ5, Shimmer\_APQ11, NHR, HNR, RPDE, DFA, PPE

# Robust RMSE for ex3  
# Number of observations to trim  
n\_trim\_best <- round(0.1 \* length(residuals\_test))  
# Sort the residuals in ascending order  
sorted\_residuals\_best <- sort(residuals\_test)  
# Trim the specified percentage of observations from both ends  
trimmed\_residuals\_best <- sorted\_residuals\_best[(n\_trim\_best + 1):(length(residuals\_test) - n\_trim\_best)]  
# Calculate the trimmed mean of the residuals for the best model  
trimmed\_mean\_best <- mean(trimmed\_residuals\_best)  
# Calculate the squared residuals using the trimmed mean for the best model  
trimmed\_squared\_residuals\_best <- (trimmed\_residuals\_best - trimmed\_mean\_best)^2  
# Calculate the robust RMSE using the trimmed mean for the best model  
robust\_PCA\_rmse\_test <- sqrt(mean(trimmed\_squared\_residuals\_best))

Nota: el mejor modelo según el RMSEP con CV, que no se que es.

### (d) Ridge regression

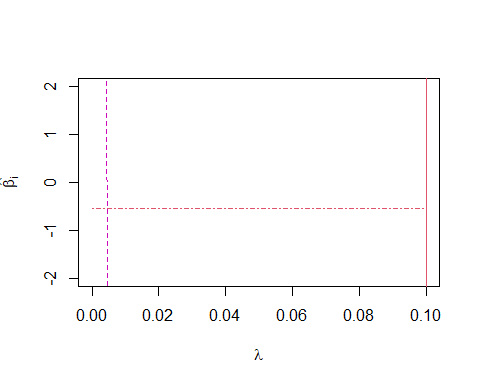
Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values being far away from the actual values.

Yes, you can adjust the model using Ridge regression. Ridge regression is a regularization technique that introduces a penalty term to the least squares objective function, helping to reduce the impact of multicollinearity and potentially improve the model’s performance.

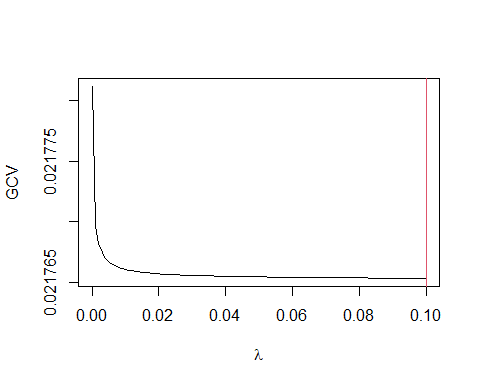
library(MASS)  
  
set.seed(123)  
  
mr <- lm.ridge(total\_UPDRS ~ ., data = train\_data, lambda=(seq(0, 0.1, 0.001)))  
(nGCV <- which.min(mr$GCV))

## 0.100   
## 101

lGCV <- mr$lambda[nGCV]  
matplot(mr$lambda,coef(mr),type="l", ylim=c(-2,2), xlab=expression(lambda),ylab=expression(hat(beta[i])))  
abline(v=lGCV,col=2)



plot(mr$lambda,mr$GCV,type="l",xlab=expression(lambda),ylab="GCV")  
abline(v=lGCV,col=2)



mr <- lm.ridge(total\_UPDRS ~ ., data = train\_data, lambda=lGCV)

# Make predictions (no y this time)  
predictions\_test = cbind(1,as.matrix(test\_data[,-1])) %\*% coef(mr)  
predictions\_train = cbind(1,as.matrix(train\_data[,-1])) %\*% coef(mr)  
  
# Calculate residuals for training and test data  
residuals\_train <- train\_data$total\_UPDRS - predictions\_train  
residuals\_test <- test\_data$total\_UPDRS - predictions\_test  
  
# Calculate R-squared for training data  
r2\_train <- 1 - sum(residuals\_train^2) / sum((train\_data$total\_UPDRS - mean(train\_data$total\_UPDRS))^2)  
  
# Calculate adjusted R-squared for training data  
n\_train <- nrow(train\_data)  
p\_train <- optimal\_components - 1 # Number of predictors (components) used  
r2\_adj\_train <- 1 - (1 - r2\_train) \* ((n\_train - 1) / (n\_train - p\_train - 1))  
   
# Calculate RMSE for training data  
ridge\_rmse\_train <- sqrt(mean(residuals\_train^2))  
# Calculate RMSE for test data  
ridge\_rmse\_test <- sqrt(mean(residuals\_test^2))  
  
# Extract the variable names  
variables <- colnames(test\_data[,-1])  
  
# Create a data frame for the results  
results\_table <- data.frame(  
 Metric = c("Número de componentes", "R-squared", "Adjusted R-squared", "RMSE\_train", "RMSE\_test"),  
 Value = c((optimal\_components - 1), r2\_train, r2\_adj\_train, ridge\_rmse\_train, ridge\_rmse\_test)  
)  
  
# Print the result table  
print(results\_table)

## Metric Value  
## 1 Número de componentes 7.0000000  
## 2 R-squared 0.1035475  
## 3 Adjusted R-squared 0.1022107  
## 4 RMSE\_train 10.0862133  
## 5 RMSE\_test 10.4070174

cat("Variables usadas en el modelo: ")

## Variables usadas en el modelo:

cat(variables, sep=", ")

## Jitter\_p, Jitter\_Abs, Jitter\_RAP, Jitter\_PPQ5, Jitter\_DDP, Shimmer, Shimmer\_dB, Shimmer\_APQ3, Shimmer\_APQ5, Shimmer\_APQ11, Shimmer\_DDA, NHR, HNR, RPDE, DFA, PPE

# Robust RMSE for ex 3  
# Number of observations to trim  
n\_trim <- round(0.1 \* length(residuals\_test))  
# Sort the residuals in ascending order  
sorted\_residuals\_test <- sort(residuals\_test)  
# Trim the specified percentage of observations from both ends  
trimmed\_residuals\_test <- sorted\_residuals\_test[(n\_trim + 1):(length(residuals\_test) - n\_trim)]  
# Calculate the trimmed mean of the residuals  
trimmed\_mean\_test <- mean(trimmed\_residuals\_test)  
# Calculate the squared residuals using the trimmed mean:  
trimmed\_squared\_residuals\_test <- (trimmed\_residuals\_test - trimmed\_mean\_test)^2  
# Calculate the robust RMSE using the trimmed mean:  
robust\_ridge\_rmse\_test <- sqrt(mean(trimmed\_squared\_residuals\_test))

In the context of ridge regression, lambda (λ) is a regularization parameter that controls the amount of shrinkage applied to the regression coefficients. It is also referred to as the penalty parameter or the tuning parameter.

Ridge regression is a technique used to address the issue of multicollinearity (high correlation) among predictor variables in a regression model. When multicollinearity is present, the ordinary least squares (OLS) estimates become unstable, leading to overfitting and unreliable coefficient estimates.

Lambda plays a crucial role in ridge regression by introducing a penalty term to the loss function that the model tries to minimize. The penalty term is proportional to the square of the magnitude of the coefficients. By increasing the value of lambda, the ridge regression model imposes a stronger penalty, shrinking the coefficient estimates towards zero.

By tuning the value of lambda, you can control the degree of shrinkage applied to the coefficients. A larger lambda value corresponds to stronger regularization and more pronounced shrinkage of the coefficients. Conversely, a smaller lambda value reduces the amount of shrinkage and allows the model to closely approximate the OLS estimates.

### (e) motor\_UPDRS como respuesta

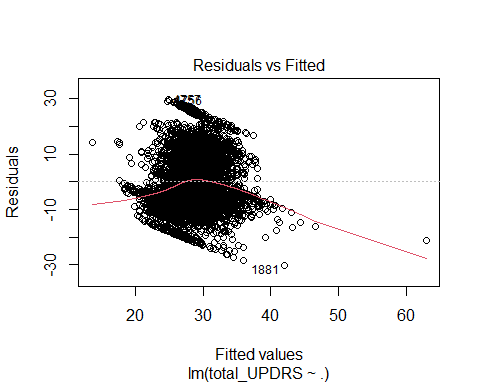
A value of “0.1” for both R2 and R2 adjusted means that the predictors included in the model explain approximately 10% of the variance in the dependent variable. This indicates a relatively weak relationship between the predictors and the response variable. Keep in mind that the interpretation of the R2 and R2 adjusted values depends on the specific context and the nature of the data being modeled.

Yo diria que si porque es puta mierda. No cumpliria el objetivo principal: El principal objetivo es predecir el UPDRS total a partir de las 16 medidas de voz. Pero esque este tampoco lo hace porque es basssurrra.

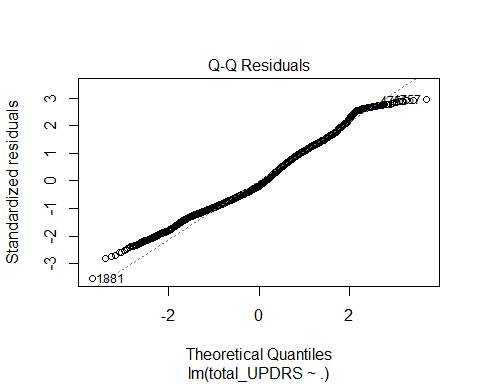
### (f) Análisis de residuos

In statistics, ordinary least squares (OLS) is a type of linear least squares method for choosing the unknown parameters in a linear regression model (with fixed level-one effects of a linear function of a set of explanatory variables) by the principle of least squares: minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being observed) in the input dataset and the output of the (linear) function of the independent variable.

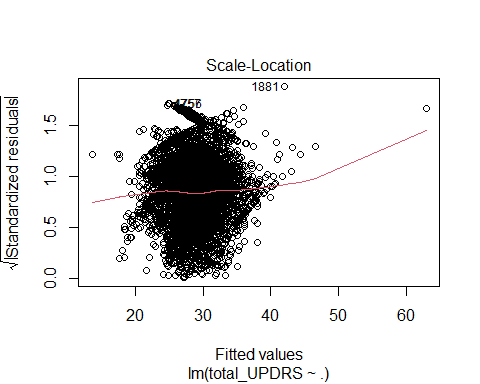
# Residual plot  
plot(model\_lineal, which = 1)



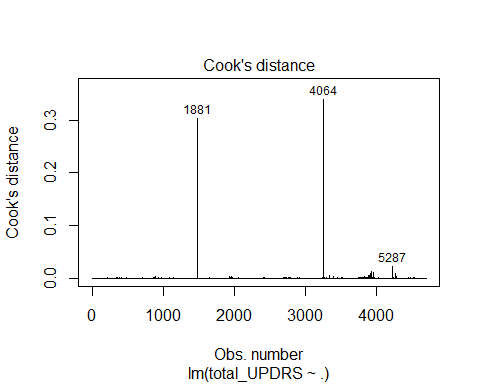
# Normal Q-Q plot  
plot(model\_lineal, which = 2)



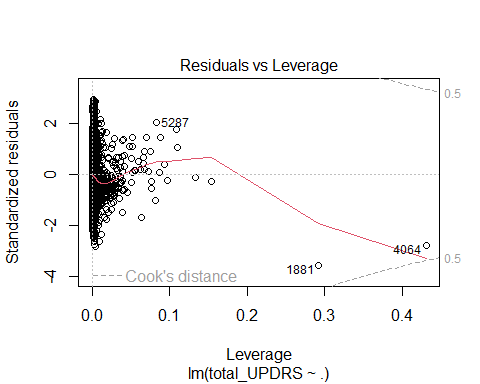
# Scale-location plot (square root of standardized residuals)  
plot(model\_lineal, which = 3)



# Cook's distance  
plot(model\_lineal, which = 4)



# Residuals vs. fitted values plot  
plot(model\_lineal, which = 5)

 which = 1: Residuals vs. Fitted: Alrededor de 0, mas o menos lineal which = 2: Normal Q-Q plot: No son muy normal en linea recta which = 3: Scale-Location plot: homocedasticidad mas o menos, no es cono which = 4: Cook’s distance plot: NO Hay puntos influyentes. Más grande cook más influye. Más 1 es influyente which = 5: Residuals vs. Leverage plot: No Existen puntos influyentes

Multicollinearity refers to a situation where independent variables in a regression model are highly correlated with each other. It can cause issues in the interpretation of coefficients and affect the stability and reliability of the regression model. To study multicollinearity in R, you can use the following approaches:

cor\_matrix <- cor(train\_data[, c("Jitter\_p","Jitter\_Abs","Jitter\_RAP","Jitter\_PPQ5","Jitter\_DDP",  
"Shimmer","Shimmer\_dB","Shimmer\_APQ3","Shimmer\_APQ5","Shimmer\_APQ11", "Shimmer\_DDA","NHR","HNR","RPDE","DFA","PPE")])  
  
# install.packages("corrplot")  
library(corrplot)

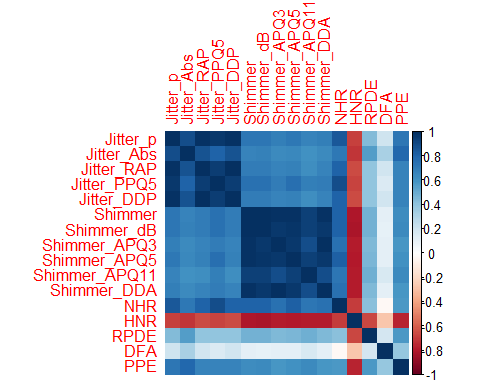
## Warning: package 'corrplot' was built under R version 4.3.1

## corrplot 0.92 loaded

##   
## Attaching package: 'corrplot'

## The following object is masked from 'package:pls':  
##   
## corrplot

corrplot(cor\_matrix, method = "color")



library(car)

## Warning: package 'car' was built under R version 4.3.1

## Loading required package: carData

##   
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
##   
## recode

vif\_values <- vif(model\_lineal)  
  
tolerance\_values <- 1/vif\_values  
  
# Extract the variable names from the linear regression model  
variables <- names(coef(model\_lineal))[-1]  
  
# Create a data frame for the results  
results\_cor <- data.frame(  
 VIF = vif\_values,  
 Tolerance = tolerance\_values  
)  
  
print(results\_cor)

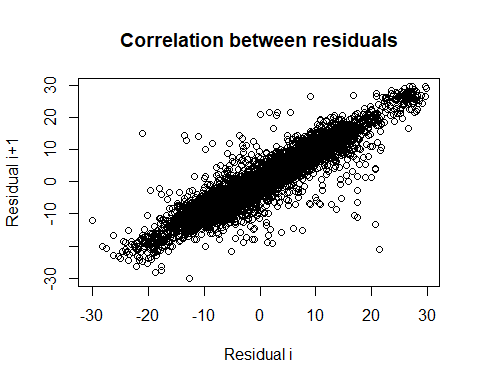
## VIF Tolerance  
## Jitter\_p 91.315313 0.01095106582675  
## Jitter\_Abs 7.597483 0.13162253129251  
## Jitter\_RAP 1310444.658918 0.00000076309976  
## Jitter\_PPQ5 31.062590 0.03219306517109  
## Jitter\_DDP 1310728.804146 0.00000076293433  
## Shimmer 173.202176 0.00577359951296  
## Shimmer\_dB 78.875425 0.01267822005257  
## Shimmer\_APQ3 24909348.382192 0.00000004014557  
## Shimmer\_APQ5 51.970530 0.01924167411572  
## Shimmer\_APQ11 14.388984 0.06949761174304  
## Shimmer\_DDA 24909297.237443 0.00000004014565  
## NHR 9.389968 0.10649663282808  
## HNR 5.533415 0.18072023874422  
## RPDE 2.096565 0.47697068283926  
## DFA 1.591038 0.62852060695795  
## PPE 4.447827 0.22482888938855

Todos los tipos de Jitter están muy relacionados con todos los tipos de Jitter y los Shimmer con los Shimmer.Osea mucho kek. Normal que el modelo se una basurrra.

Tolerance is the reciprocal of the VIF. It indicates the proportion of variance in an independent variable that is not explained by other independent variables. Variables with low tolerance values (close to 0) indicate high multicollinearity.

The VIF measures how much the variance of the estimated regression coefficient is inflated due to multicollinearity.

# Calculate residuals  
residuals <- residuals(model\_lineal)  
  
# Plot residuals  
plot(residuals[-length(residuals)], residuals[-1], xlab = "Residual i", ylab = "Residual i+1", main = "Correlation between residuals")



If your data is randomly ordered and does not have a time component, then the concept of “consecutive residuals” in the temporal sense may not be directly applicable. The assumption of independence between residuals in linear regression typically assumes that the order of the data points does not matter, as long as the observations are independent and identically distributed.

In this case, you can still check for correlation between the residuals, but the interpretation would be different. Instead of investigating temporal dependence, you would be examining whether there is a pattern or relationship between the residuals regardless of their order. The correlation between residuals can provide insights into potential systematic patterns or relationships in the errors that may affect the validity of your linear model.

Therefore, while the concept of consecutive residuals in a temporal sense may not be relevant to your randomly ordered data, you can still use the code you provided to check for correlation between the residuals and investigate whether there are correlated errors present in your linear model.

# Ejercicio 3

### (a) Comparación modelos con y sin puntos influyentes

# Calculate cooks distance  
cooksd <- cooks.distance(model\_lineal)  
  
# Get 3 most influential points  
top3\_indices <- order(cooksd, decreasing = TRUE)[1:3]  
  
# Remove the points  
train\_data <- train\_data[-top3\_indices, ]

Re-execute everything but with clean data

# Fit the model  
model\_lineal = lm(total\_UPDRS ~ ., data = train\_data)  
  
# Extract R-squared  
r\_squared = summary(model\_lineal)$r.squared  
  
# Extract adjusted R-squared  
adj\_r\_squared = summary(model\_lineal)$adj.r.squared  
  
# Predict on the training data  
predictions\_train = predict(model\_lineal, train\_data)  
# Predict on the test data  
predictions\_test <- predict(model\_lineal, test\_data)  
  
# Calculate residuals train  
residuals\_train = train\_data$total\_UPDRS - predictions\_train  
# Calculate residuals test  
residuals\_test <- test\_data$total\_UPDRS - predictions\_test  
  
# Calculate RMSE train  
lineal\_rmse\_train = sqrt(mean(residuals\_train^2))  
# Calculate RMSE test  
lineal\_clean\_rmse\_test <- sqrt(mean(residuals\_test^2))  
  
# Create results table  
# Extract the variable names from the linear regression model  
variables <- names(coef(model\_lineal))[-1]  
  
# Create a data frame for the results  
results\_table <- data.frame(  
 Metric = c("Número de variables", "R-squared", "Adjusted R-squared", "RMSE\_train", "clean\_rmse\_test"),  
 Value = c(length(variables), r\_squared, adj\_r\_squared, lineal\_rmse\_train, lineal\_clean\_rmse\_test)  
)  
  
# Print the result table  
print(results\_table)

## Metric Value  
## 1 Número de variables 16.0000000  
## 2 R-squared 0.1080147  
## 3 Adjusted R-squared 0.1049665  
## 4 RMSE\_train 10.0586529  
## 5 clean\_rmse\_test 10.4840466

cat("Variables usadas en el modelo: ")

## Variables usadas en el modelo:

cat(variables, sep=", ")

## Jitter\_p, Jitter\_Abs, Jitter\_RAP, Jitter\_PPQ5, Jitter\_DDP, Shimmer, Shimmer\_dB, Shimmer\_APQ3, Shimmer\_APQ5, Shimmer\_APQ11, Shimmer\_DDA, NHR, HNR, RPDE, DFA, PPE

# Robust RMSE for exercise 3  
# Number of observations to trim  
n\_trim <- round(0.1 \* length(residuals\_test))  
# Sort the residuals in ascending order  
sorted\_residuals\_test <- sort(residuals\_test)  
# Trim the specified percentage of observations from both ends  
trimmed\_residuals\_test <- sorted\_residuals\_test[(n\_trim + 1):(length(residuals\_test) - n\_trim)]  
# Calculate the trimmed mean of the residuals  
trimmed\_mean\_test <- mean(trimmed\_residuals\_test)  
# Calculate the squared residuals using the trimmed mean  
trimmed\_squared\_residuals\_test <- (trimmed\_residuals\_test - trimmed\_mean\_test)^2  
# Calculate the robust RMSE using the trimmed mean  
robust\_lineal\_clean\_rmse\_test <- sqrt(mean(trimmed\_squared\_residuals\_test))

library(MASS)  
  
set.seed(123)  
  
# Perform stepwise variable selection based on AIC  
model\_stepwise <- stepAIC(model\_lineal, direction = "both", trace = FALSE)  
  
# Extract R-squared  
r\_squared = summary(model\_stepwise)$r.squared  
  
# Extract adjusted R-squared  
adj\_r\_squared = summary(model\_stepwise)$adj.r.squared  
  
# Predict on the training data  
predictions\_train = predict(model\_stepwise, train\_data)  
# Predict on the test data  
predictions\_test <- predict(model\_stepwise, test\_data)  
  
# Calculate residuals  
residuals\_train = train\_data$total\_UPDRS - predictions\_train  
# Calculate residuals  
residuals\_test <- test\_data$total\_UPDRS - predictions\_test  
  
# Calculate RMSE train  
step\_rmse\_train = sqrt(mean(residuals\_train^2))  
# Calculate RMSE test  
step\_clean\_rmse\_test <- sqrt(mean(residuals\_test^2))  
  
# Results  
# Extract the variable names from the linear regression model  
variables <- names(coef(model\_stepwise))[-1]  
  
# Create a data frame for the results  
results\_table <- data.frame(  
 Metric = c("Número de variables", "R-squared", "Adjusted R-squared", "RMSE\_train", "clean\_rmse\_test"),  
 Value = c(length(variables), r\_squared, adj\_r\_squared, step\_rmse\_train, step\_clean\_rmse\_test)  
)  
  
# Print the result table  
print(results\_table)

## Metric Value  
## 1 Número de variables 11.0000000  
## 2 R-squared 0.1074522  
## 3 Adjusted R-squared 0.1053575  
## 4 RMSE\_train 10.0618235  
## 5 clean\_rmse\_test 10.4817952

cat("Variables usadas en el modelo: ")

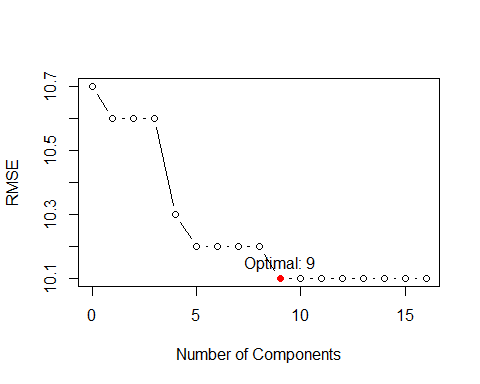
## Variables usadas en el modelo:

cat(variables, sep=", ")

## Jitter\_p, Jitter\_Abs, Shimmer, Shimmer\_APQ3, Shimmer\_APQ5, Shimmer\_APQ11, NHR, HNR, RPDE, DFA, PPE

# Robust RMSE  
# Number of observations to trim  
n\_trim <- round(0.1 \* length(residuals\_test))  
# Sort the residuals in ascending order  
sorted\_residuals\_test <- sort(residuals\_test)  
# Trim the specified percentage of observations from both ends  
trimmed\_residuals\_test <- sorted\_residuals\_test[(n\_trim + 1):(length(residuals\_test) - n\_trim)]  
# Calculate the trimmed mean of the residuals  
trimmed\_mean\_test <- mean(trimmed\_residuals\_test)  
# Calculate the squared residuals using the trimmed mean:  
trimmed\_squared\_residuals\_test <- (trimmed\_residuals\_test - trimmed\_mean\_test)^2  
# Calculate the robust RMSE using the trimmed mean:  
robust\_step\_clean\_rmse\_test <- sqrt(mean(trimmed\_squared\_residuals\_test))

# install.packages('pls')  
library(pls)  
  
set.seed(123)  
  
PCA\_model <- pcr(total\_UPDRS ~ ., data = train\_data, validation="CV", scale = TRUE)  
  
# Calculate RMSE values  
mpcCV <- RMSEP(PCA\_model, estimate = "CV")  
rmse\_values <- round(mpcCV$val, 1) # Rounded to first decimal to account for complexity  
  
# Plot the graph  
plot(mpcCV$comp, rmse\_values, type = "b", xlab = "Number of Components", ylab = "RMSE")  
  
# Find the optimal number of components  
optimal\_components <- mpcCV$comp[which.min(rmse\_values)]  
points(optimal\_components, min(rmse\_values), col = "red", pch = 16)  
text(optimal\_components, min(rmse\_values), paste("Optimal:", optimal\_components), pos = 3)



# Predict on the training data using the model  
predictions\_train <- predict(PCA\_model, train\_data, ncomp = optimal\_components-1)  
# Predict on the test data using the model  
predictions\_test <- predict(PCA\_model, test\_data, ncomp = optimal\_components-1)  
   
# Calculate residuals for training and test data  
residuals\_train <- train\_data$total\_UPDRS - predictions\_train  
residuals\_test <- test\_data$total\_UPDRS - predictions\_test  
  
# Calculate R-squared for training data  
r2\_train <- 1 - sum(residuals\_train^2) / sum((train\_data$total\_UPDRS - mean(train\_data$total\_UPDRS))^2)  
  
# Calculate adjusted R-squared for training data  
n\_train <- nrow(train\_data)  
p\_train <- optimal\_components - 1 # Number of predictors (components) used  
r2\_adj\_train <- 1 - (1 - r2\_train) \* ((n\_train - 1) / (n\_train - p\_train - 1))  
   
# Calculate RMSE for training data  
PCA\_rmse\_train <- sqrt(mean(residuals\_train^2))  
# Calculate RMSE for test data  
PCA\_clean\_rmse\_test <- sqrt(mean(residuals\_test^2))  
  
# Create a data frame for the results  
results\_table <- data.frame(  
 Metric = c("Número de componentes", "R-squared", "Adjusted R-squared", "RMSE\_train", "clean\_rmse\_test"),  
 Value = c((optimal\_components - 1), r2\_train, r2\_adj\_train, PCA\_rmse\_train, PCA\_clean\_rmse\_test)  
)  
  
# Print the result table  
print(results\_table)

## Metric Value  
## 1 Número de componentes 8.00000000  
## 2 R-squared 0.08562868  
## 3 Adjusted R-squared 0.08406898  
## 4 RMSE\_train 10.18409080  
## 5 clean\_rmse\_test 10.52010556

cat("Variables usadas en el modelo: ")

## Variables usadas en el modelo:

cat(variables, sep=", ")

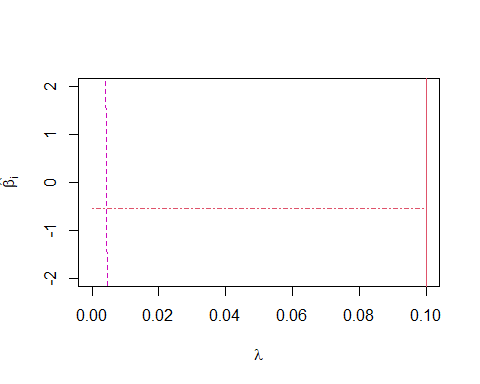
## Jitter\_p, Jitter\_Abs, Shimmer, Shimmer\_APQ3, Shimmer\_APQ5, Shimmer\_APQ11, NHR, HNR, RPDE, DFA, PPE

# Robust RMSE for ex3  
# Number of observations to trim  
n\_trim\_best <- round(0.1 \* length(residuals\_test))  
# Sort the residuals in ascending order  
sorted\_residuals\_best <- sort(residuals\_test)  
# Trim the specified percentage of observations from both ends  
trimmed\_residuals\_best <- sorted\_residuals\_best[(n\_trim\_best + 1):(length(residuals\_test) - n\_trim\_best)]  
# Calculate the trimmed mean of the residuals for the best model  
trimmed\_mean\_best <- mean(trimmed\_residuals\_best)  
# Calculate the squared residuals using the trimmed mean for the best model  
trimmed\_squared\_residuals\_best <- (trimmed\_residuals\_best - trimmed\_mean\_best)^2  
# Calculate the robust RMSE using the trimmed mean for the best model  
robust\_PCA\_clean\_rmse\_test <- sqrt(mean(trimmed\_squared\_residuals\_best))

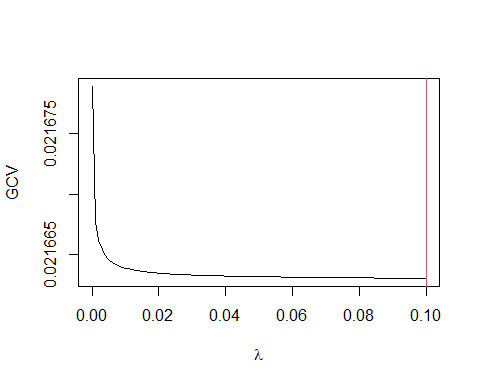
library(MASS)  
  
set.seed(123)  
  
mr <- lm.ridge(total\_UPDRS ~ ., data = train\_data, lambda=(seq(0, 0.1, 0.001)))  
(nGCV <- which.min(mr$GCV))

## 0.100   
## 101

lGCV <- mr$lambda[nGCV]  
matplot(mr$lambda,coef(mr),type="l", ylim=c(-2,2), xlab=expression(lambda),ylab=expression(hat(beta[i])))  
abline(v=lGCV,col=2)



plot(mr$lambda,mr$GCV,type="l",xlab=expression(lambda),ylab="GCV")  
abline(v=lGCV,col=2)



mr <- lm.ridge(total\_UPDRS ~ ., data = train\_data, lambda=lGCV)

# Make predictions (no y this time)  
predictions\_test = cbind(1,as.matrix(test\_data[,-1])) %\*% coef(mr)  
predictions\_train = cbind(1,as.matrix(train\_data[,-1])) %\*% coef(mr)  
  
# Calculate residuals for training and test data  
residuals\_train <- train\_data$total\_UPDRS - predictions\_train  
residuals\_test <- test\_data$total\_UPDRS - predictions\_test  
  
# Calculate R-squared for training data  
r2\_train <- 1 - sum(residuals\_train^2) / sum((train\_data$total\_UPDRS - mean(train\_data$total\_UPDRS))^2)  
  
# Calculate adjusted R-squared for training data  
n\_train <- nrow(train\_data)  
p\_train <- optimal\_components - 1 # Number of predictors (components) used  
r2\_adj\_train <- 1 - (1 - r2\_train) \* ((n\_train - 1) / (n\_train - p\_train - 1))  
   
# Calculate RMSE for training data  
ridge\_rmse\_train <- sqrt(mean(residuals\_train^2))  
# Calculate RMSE for test data  
ridge\_clean\_rmse\_test <- sqrt(mean(residuals\_test^2))  
  
# Extract the variable names  
variables <- colnames(test\_data[,-1])  
  
# Create a data frame for the results  
results\_table <- data.frame(  
 Metric = c("Número de componentes", "R-squared", "Adjusted R-squared", "RMSE\_train", "clean\_rmse\_test"),  
 Value = c((optimal\_components - 1), r2\_train, r2\_adj\_train, ridge\_rmse\_train, ridge\_clean\_rmse\_test)  
)  
  
# Print the result table  
print(results\_table)

## Metric Value  
## 1 Número de componentes 8.0000000  
## 2 R-squared 0.1079086  
## 3 Adjusted R-squared 0.1063869  
## 4 RMSE\_train 10.0592508  
## 5 clean\_rmse\_test 10.4868157

cat("Variables usadas en el modelo: ")

## Variables usadas en el modelo:

cat(variables, sep=", ")

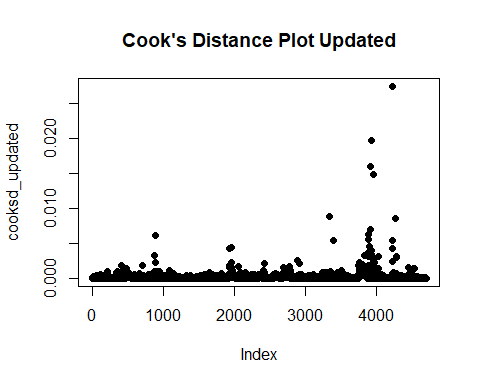
## Jitter\_p, Jitter\_Abs, Jitter\_RAP, Jitter\_PPQ5, Jitter\_DDP, Shimmer, Shimmer\_dB, Shimmer\_APQ3, Shimmer\_APQ5, Shimmer\_APQ11, Shimmer\_DDA, NHR, HNR, RPDE, DFA, PPE

# Robust RMSE for ex 3  
# Number of observations to trim  
n\_trim <- round(0.1 \* length(residuals\_test))  
# Sort the residuals in ascending order  
sorted\_residuals\_test <- sort(residuals\_test)  
# Trim the specified percentage of observations from both ends  
trimmed\_residuals\_test <- sorted\_residuals\_test[(n\_trim + 1):(length(residuals\_test) - n\_trim)]  
# Calculate the trimmed mean of the residuals  
trimmed\_mean\_test <- mean(trimmed\_residuals\_test)  
# Calculate the squared residuals using the trimmed mean:  
trimmed\_squared\_residuals\_test <- (trimmed\_residuals\_test - trimmed\_mean\_test)^2  
# Calculate the robust RMSE using the trimmed mean:  
robust\_ridge\_clean\_rmse\_test <- sqrt(mean(trimmed\_squared\_residuals\_test))

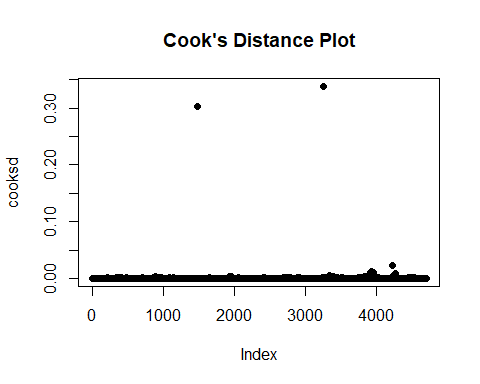
# Create a data frame for the results  
results <- data.frame(  
 Modelo = c("OLS", "AIC", "RCP", "Ridge"),  
 RMSE\_Con = c(lineal\_rmse\_test, step\_rmse\_test, PCA\_rmse\_test, ridge\_rmse\_test),  
 RMSE\_Sin = c(lineal\_clean\_rmse\_test, step\_clean\_rmse\_test, PCA\_clean\_rmse\_test, ridge\_clean\_rmse\_test)  
)  
  
print(results)

## Modelo RMSE\_Con RMSE\_Sin  
## 1 OLS 10.40468 10.48405  
## 2 AIC 10.40713 10.48180  
## 3 RCP 10.55268 10.52011  
## 4 Ridge 10.40702 10.48682

# Checking if the points were correctly deleted  
# Re-calculate cooks distance  
cooksd\_updated <- cooks.distance(model\_lineal) # Clean data  
  
# Plot both  
plot(cooksd\_updated, pch = 20, cex = 1.5, main = "Cook's Distance Plot Updated")



plot(cooksd, pch = 20, cex = 1.5, main = "Cook's Distance Plot")



### (b) Cálculo del RMSE robusto

The trimmed mean is less sensitive to outliers and can provide a more robust estimate of the central tendency.

# Create a data frame for the results  
results2 <- data.frame(  
 Modelo = c("OLS", "AIC", "RCP", "Ridge"),  
 RMSE\_Con = c(lineal\_rmse\_test, step\_rmse\_test, PCA\_rmse\_test, ridge\_rmse\_test),  
 RMSE\_Sin = c(lineal\_clean\_rmse\_test, step\_clean\_rmse\_test, PCA\_clean\_rmse\_test, ridge\_clean\_rmse\_test),  
 Robusto\_Con = c(robust\_lineal\_rmse\_test, robust\_step\_rmse\_test, robust\_PCA\_rmse\_test, robust\_ridge\_rmse\_test),  
 Robusto\_Sin = c(robust\_lineal\_clean\_rmse\_test, robust\_step\_clean\_rmse\_test, robust\_PCA\_clean\_rmse\_test, robust\_ridge\_clean\_rmse\_test)  
)  
  
print(results2)

## Modelo RMSE\_Con RMSE\_Sin Robusto\_Con Robusto\_Sin  
## 1 OLS 10.40468 10.48405 7.021287 7.044118  
## 2 AIC 10.40713 10.48180 7.031638 7.063658  
## 3 RCP 10.55268 10.52011 7.106559 7.100806  
## 4 Ridge 10.40702 10.48682 7.028084 7.050813

### (c) LTS o Huber

set.seed(123)  
  
# Data import  
# Note: I changed the variable names to avoid problems with symbols  
  
import.data <-  
"http://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/telemonitoring/parkinsons\_updrs.data"  
parkinson <- read.table(url(import.data), sep=",", skip=1)  
names(parkinson) <- c("subject#","age","sex","test\_time","motor\_UPDRS","total\_UPDRS",  
"Jitter\_p","Jitter\_Abs","Jitter\_RAP","Jitter\_PPQ5","Jitter\_DDP",  
"Shimmer","Shimmer\_dB","Shimmer\_APQ3","Shimmer\_APQ5","Shimmer\_APQ11", "Shimmer\_DDA","NHR","HNR","RPDE","DFA","PPE")  
  
# Select predictor variables and response  
library(dplyr)  
set.seed(123)  
parkinson <- parkinson %>% dplyr::select(6:22)  
  
# Split the data into train and test  
library(caret)  
set.seed(123)  
train\_indices <- createDataPartition(parkinson$total\_UPDRS, p = 0.8, list = FALSE)  
train\_data <- parkinson[train\_indices, ]  
test\_data <- parkinson[-train\_indices, ]  
  
# Huber  
require(MASS)  
hub <- rlm(total\_UPDRS ~ ., data = train\_data)  
summary(hub)

##   
## Call: rlm(formula = total\_UPDRS ~ ., data = train\_data)  
## Residuals:  
## Min 1Q Median 3Q Max   
## -38.708 -7.026 -1.416 7.516 31.287   
##   
## Coefficients:  
## Value Std. Error t value   
## (Intercept) 60.3923 3.4288 17.6133  
## Jitter\_p 372.3992 251.1458 1.4828  
## Jitter\_Abs -38611.9982 11264.0447 -3.4279  
## Jitter\_RAP -22944.9819 54363.4404 -0.4221  
## Jitter\_PPQ5 -234.0636 217.5246 -1.0760  
## Jitter\_DDP 7754.6094 18122.8676 0.4279  
## Shimmer 179.2708 73.5378 2.4378  
## Shimmer\_dB -7.3858 5.5745 -1.3249  
## Shimmer\_APQ3 -19115.0831 54687.0579 -0.3495  
## Shimmer\_APQ5 -155.1232 62.3563 -2.4877  
## Shimmer\_APQ11 96.5758 27.6603 3.4915  
## Shimmer\_DDA 6304.1068 18229.0061 0.3458  
## NHR -43.1735 7.2424 -5.9612  
## HNR -0.4856 0.0802 -6.0527  
## RPDE 10.9901 2.1184 5.1880  
## DFA -46.7154 2.6448 -17.6633  
## PPE 19.0560 3.4049 5.5966  
##   
## Residual standard error: 10.62 on 4685 degrees of freedom

# Predict on train and test data  
train\_pred <- predict(hub, newdata = train\_data)  
test\_pred <- predict(hub, newdata = test\_data)  
  
# Calculate R-squared  
train\_r2 <- 1 - sum((train\_data$total\_UPDRS - train\_pred)^2) / sum((train\_data$total\_UPDRS - mean(train\_data$total\_UPDRS))^2)  
test\_r2 <- 1 - sum((test\_data$total\_UPDRS - test\_pred)^2) / sum((test\_data$total\_UPDRS - mean(train\_data$total\_UPDRS))^2)  
  
# Calculate adjusted R-squared  
n <- nrow(train\_data)  
p <- length(coef(hub))  
train\_r2\_adj <- 1 - (1 - train\_r2) \* ((n - 1) / (n - p - 1))  
test\_r2\_adj <- 1 - (1 - test\_r2) \* ((n - 1) / (n - p - 1))  
  
# Calculate RMSE  
train\_rmse <- sqrt(mean((train\_data$total\_UPDRS - train\_pred)^2))  
test\_rmse <- sqrt(mean((test\_data$total\_UPDRS - test\_pred)^2))  
  
# Create results table  
# Extract the variable names from the linear regression model  
variables <- names(coef(hub))[-1]  
  
# Create a data frame for the results  
results\_table <- data.frame(  
 Metric = c("Número de variables", "R-squared", "Adjusted R-squared", "RMSE\_train", "RMSE\_test"),  
 Value = c(length(variables), train\_r2, train\_r2\_adj, train\_rmse, test\_rmse)  
)  
  
# Print the result table  
print(results\_table)

## Metric Value  
## 1 Número de variables 16.00000000  
## 2 R-squared 0.09922548  
## 3 Adjusted R-squared 0.09595623  
## 4 RMSE\_train 10.11049817  
## 5 RMSE\_test 10.46585726

cat("Variables usadas en el modelo: ")

## Variables usadas en el modelo:

cat(variables, sep=", ")

## Jitter\_p, Jitter\_Abs, Jitter\_RAP, Jitter\_PPQ5, Jitter\_DDP, Shimmer, Shimmer\_dB, Shimmer\_APQ3, Shimmer\_APQ5, Shimmer\_APQ11, Shimmer\_DDA, NHR, HNR, RPDE, DFA, PPE

# ANEXO

### Código