PRÁCTICA 2A: REGRESIÓN LINEAL MÚLTIPLE

ANÁLISIS ESTADÍSTICO MULTIVARIANTE

GRADO EN CIENCIA E INGENIERÍA DE DATOS

Sumario: En esta práctica veremos cómo llevar a cabo un análisis de Regresión Lineal Múltiple (RLM) usando R. La estimación de los parámetros del modelo la haremos directamente usando la función *lm()* de R, que resuelve el problema de mínimos cuadrados a través del método de factorización QR (descomposición de una matriz como producto de una matriz ortogonal por una triangular superior). La práctica contempla aspectos fundamentales en RLM como son: la transformación de los datos (familia Box-Cox), la selección del modelo (o selección de regresores), la validación del modelo y la realización de predicciones. Además, se muestra cómo introducir en el modelo potencias de los predictores e interacciones entre los mismos (modelos polinómicos).

1. Conjunto de datos y análisis descriptivo previo

Los datos que usaremos en esta práctica han sido simulados con el fin de poder comparar el resultado del análisis RLM con el verdadero modelo simulado. Las variables empleadas y las simulaciones generadas están inspiradas en datos reales.

En el tratamiento de aguas residuales, el oxígeno (OXIGENO) consumido por unos microorganismos en el proceso de depuración puede venir influenciado por las siguientes variables: BOD (*biological oxigen demand*), TKN (*total Kjeldahl nitrogen*), TS (*total solids*), TVS (*total volatile solids*) y COD (*chemical oxygen demand*).

Objetivo del estudio: averiguar qué variables influyen principalmente en la cantidad de oxígeno consumida por los microorganismos y proporcionar una ecuación que modelice la relación con el fin de realizar predicciones para nuevas observaciones de las variables predictoras. Por tanto, el OXIGENO es la variable respuesta, mientras que el resto son los regresores o predictores del modelo.

Datos simulados:

- Los datos de los regresores TS, BOD, COD y TKN se han simulado usando distribuciones independientes. Sin embargo, el regresor TVS se ha simulado usando una relación lineal estrecha con TS, concretamente:

$$TVS = 0.65 + 0.003 \cdot TS + Normal(0, 0.01)$$

- Los tres modelos que hemos simulado (tres variables respuesta), se han obtenido mediante la siguiente relación en cada caso:

$$\begin{aligned} OXIGENO\_lineal &= -0.3 + 0.04 \cdot TS + 0.02 \cdot COD + Normal(0, 0.3) \\ OXIGENO\_cubica &= (-0.3 + 0.04 \cdot TS + 0.02 \cdot COD + Normal(0, 0.3))^3 \\ OXIGENO\_exponencial &= exp(-0.3 + 0.04 \cdot TS + 0.02 \cdot COD + Normal(0, 0.3)) \end{aligned}$$

- En la última fila (caso número 33), se ha introducido un dato atípico (outlier) para cada variable respuesta.

En una situación real, no dispondremos de la ecuación que relaciona las variables en estudio sino que debemos determinar qué variables influyen y estimar la ecuación que las relaciona.

Comenzaremos cargando el fichero de datos **moore_simulado_def.xlsx**. Hemos descargado el fichero previamente en la subcarpeta "data" del proyecto de R.

Importante: Recordar que, si trabajamos con un script de R-Markdown (.Rmd), el directorio de trabajo parte de la carpeta donde se encuentra ubicado dicho script. Sin embargo, cuando trabajamos con un script de R (fichero .R), el directorio de trabajo parte de la carpeta donde se encuentra ubicado el proyecto, de manera que no necesitaríamos iniciar el **path** con "../".

```
library("readxl")
datos <- read_xlsx("../data/moore_simulado_def.xlsx")

#Si usamos script de R en lugar de script de R-Markdown pondremos lo siguiente:
#datos <- read_xlsx("data/moore_simulado.xlsx")
```

Como hemos indicado anteriormente, en realidad se han simulado 3 modelos diferentes (uno lineal y dos no lineales). Creamos 3 dataframes, uno por cada variable respuesta:

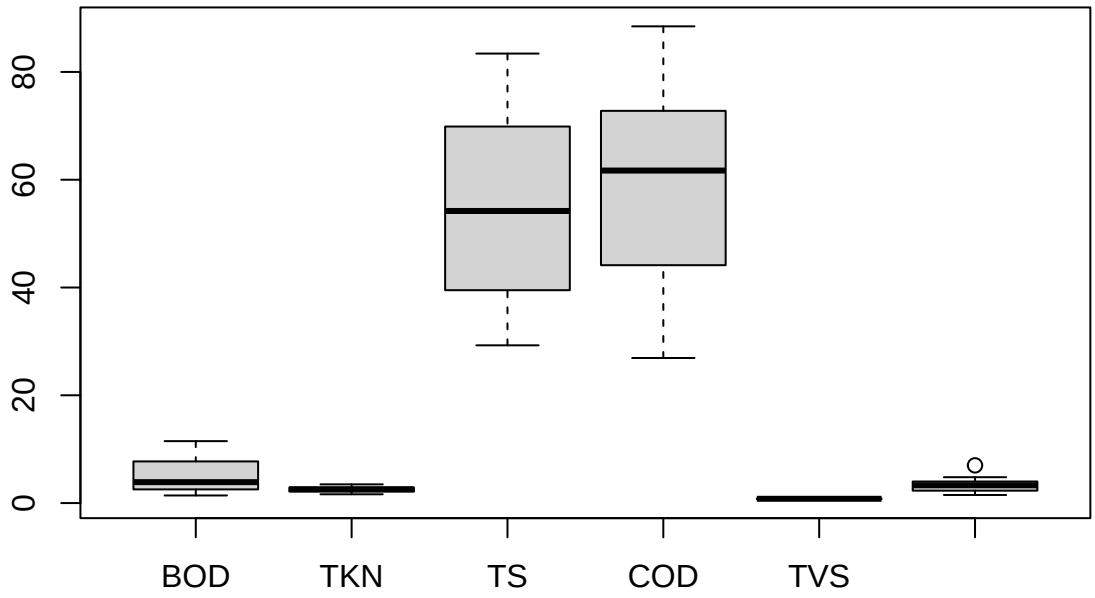
```
datos_lin <- datos[ , -c(7,8)]
datos_cub <- datos[ , -c(6,8)]
datos_exp <- datos[ , -c(6,7)]
```

Comenzamos considerando la variable respuesta OXIGENO_lineal, que no requiere ninguna transformación previa en los datos, como veremos más adelante. En una sección posterior, veremos cómo abordar los casos para usar a OXIGENO_cubica y a OXIGENO_exponencial como variables respuesta, que sí requieren una transformación en los datos.

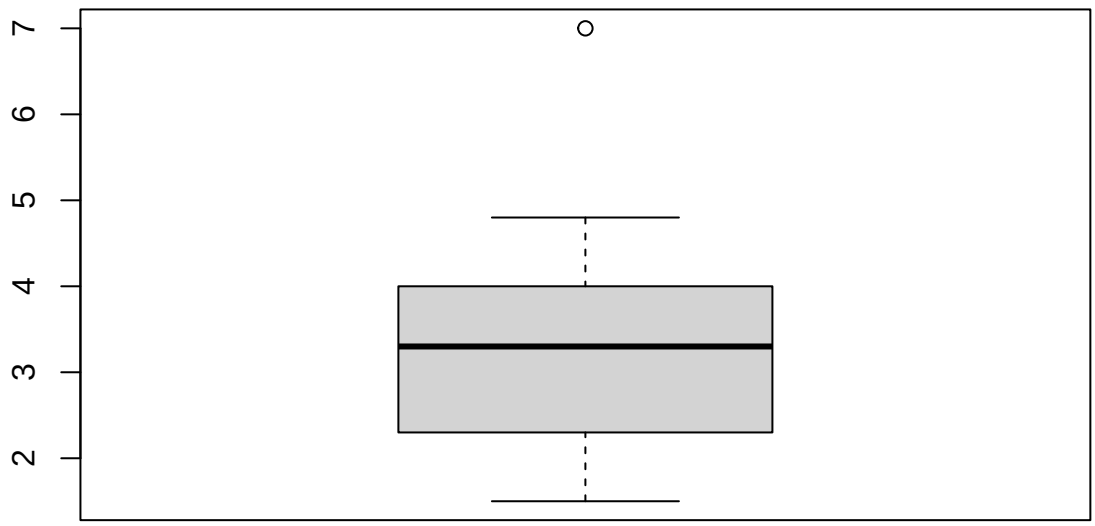
Por tanto, **en adelante usaremos sólo el dataframe "datos_lin"**. En la sección dedicada a transformación de los datos veremos cómo analizar los dos modelos no lineales correspondientes a los dataframes "datos_cub" y "datos_exp".

El **análisis descriptivo inicial** debe contemplar, entre otros, diagramas de caja para cada variable por separado, con el fin de identificar outliers, y diagramas de dispersión por pares, con el fin de identificar relaciones lineales entre predictores o con la variable respuesta. También debemos hacer pruebas de normalidad de la variable respuesta para determinar si es necesario una transformación en los datos.

```
#Diagramas de caja
boxplot(datos_lin)
```




```
#Podemos realizar cada boxplot por separado para visualizarlo mejor.
#Para la variable respuesta tenemos:
boxplot(datos_lin$OXIGENO_lineal, xlab = "OXIGENO_lineal")
```

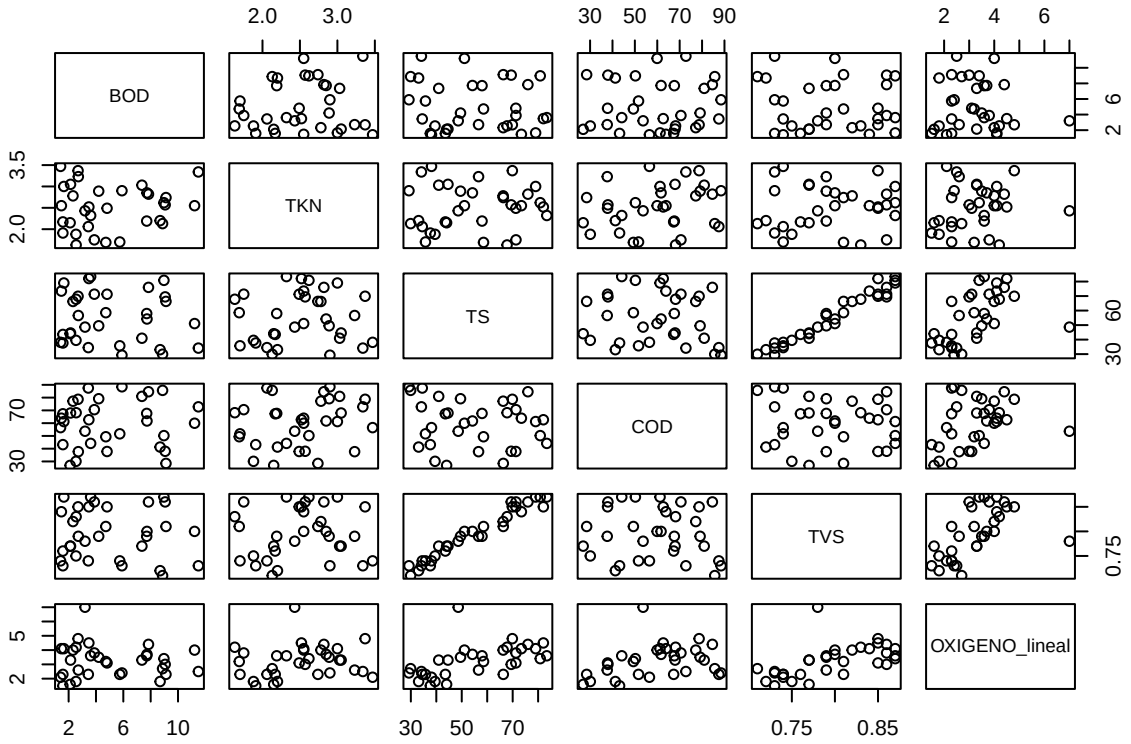


OXIGENO_lineal

```
#Para detectar la observación atípica haremos:
which.max(datos_lin$OXIGENO_lineal)
```

```
## [1] 33
```

```
#Nube de puntos por pares de variables
plot(datos_lin)
```

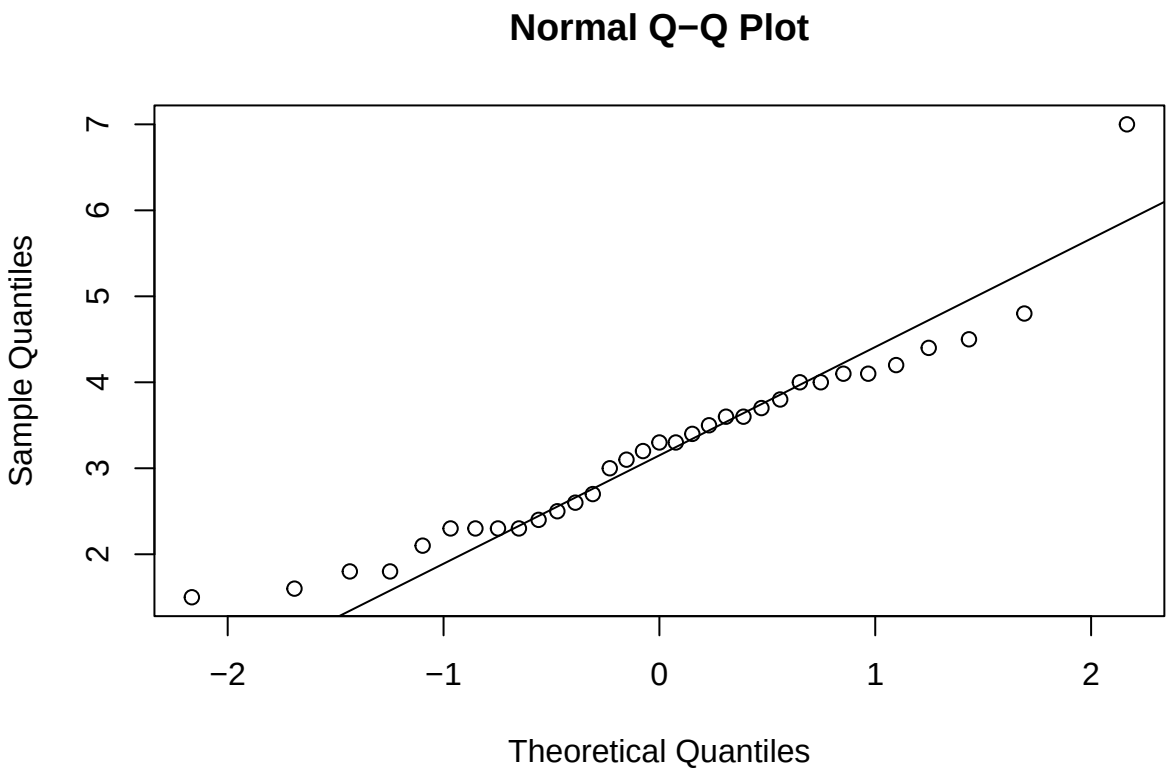


Podemos ver relaciones lineales claras entre algunas variables predictoras (TS y TVS), pero la mayoría parecen independientes. En general, hay una baja relación lineal con la variable respuesta, de manera individual. En las gráficas de la variable respuesta aparece la observación atípica de la línea 33.

```
#Pruebas de normalidad de la variable respuesta
shapiro.test(datos_lin$OXIGENO_lineal)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datos_lin$OXIGENO_lineal
## W = 0.93158, p-value = 0.03882
```

```
qqnorm(datos_lin$OXIGENO_lineal)
qqline(datos_lin$OXIGENO_lineal)
```

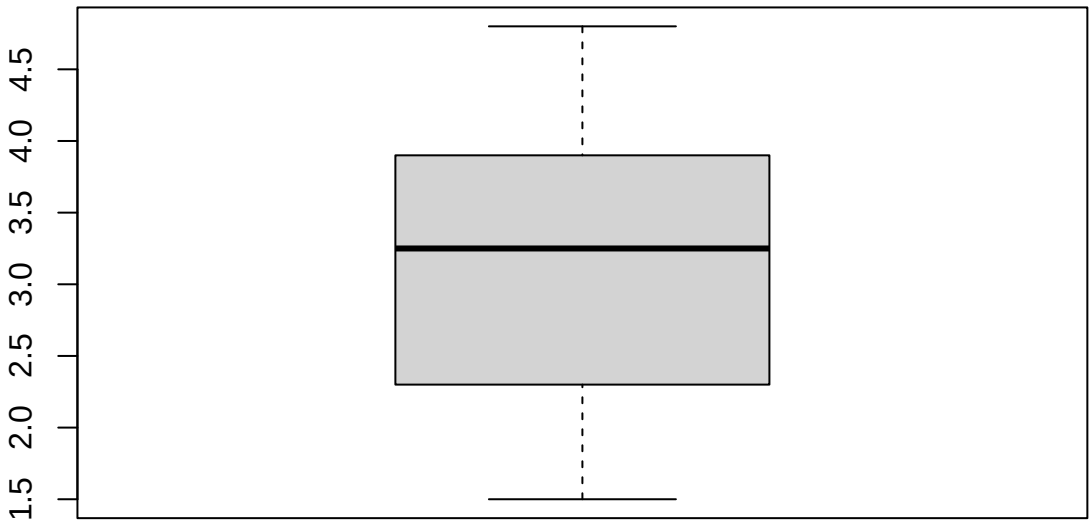


Los gráficos muestran que la variable respuesta toma un valor especialmente alto (outlier) para la observación número 33. Además la hipótesis de Normalidad para la variable respuesta está en entredicho, quizás provocado por dicho valor.

En esta situación, optaremos por eliminar la observación número 33 y repetir el análisis descriptivo previo, comprobando que ahora sí se puede asumir la hipótesis de Normalidad para la variable respuesta.

Importante: En general, los datos atípicos no deben eliminarse de forma automática, sino que debemos realizar el análisis con y sin dichos datos para evaluar su efecto sobre el modelo, si se trata verdaderamente de observaciones influyentes o no.

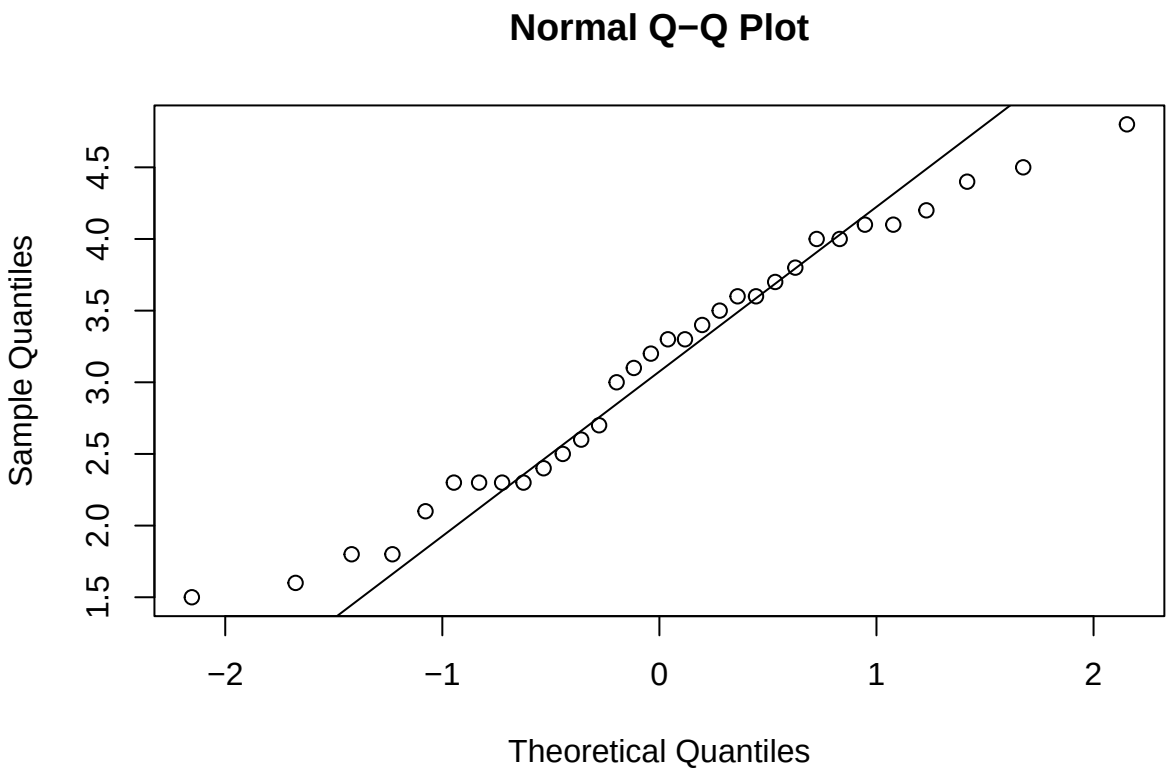
```
datos_lin <- datos_lin[-33, ]
boxplot(datos_lin$OXIGENO_lineal)
```




```
shapiro.test(datos_lin$OXIGENO_lineal)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datos_lin$OXIGENO_lineal
## W = 0.9645, p-value = 0.3628
```

```
qqnorm(datos_lin$OXIGENO_lineal)
qqline(datos_lin$OXIGENO_lineal)
```

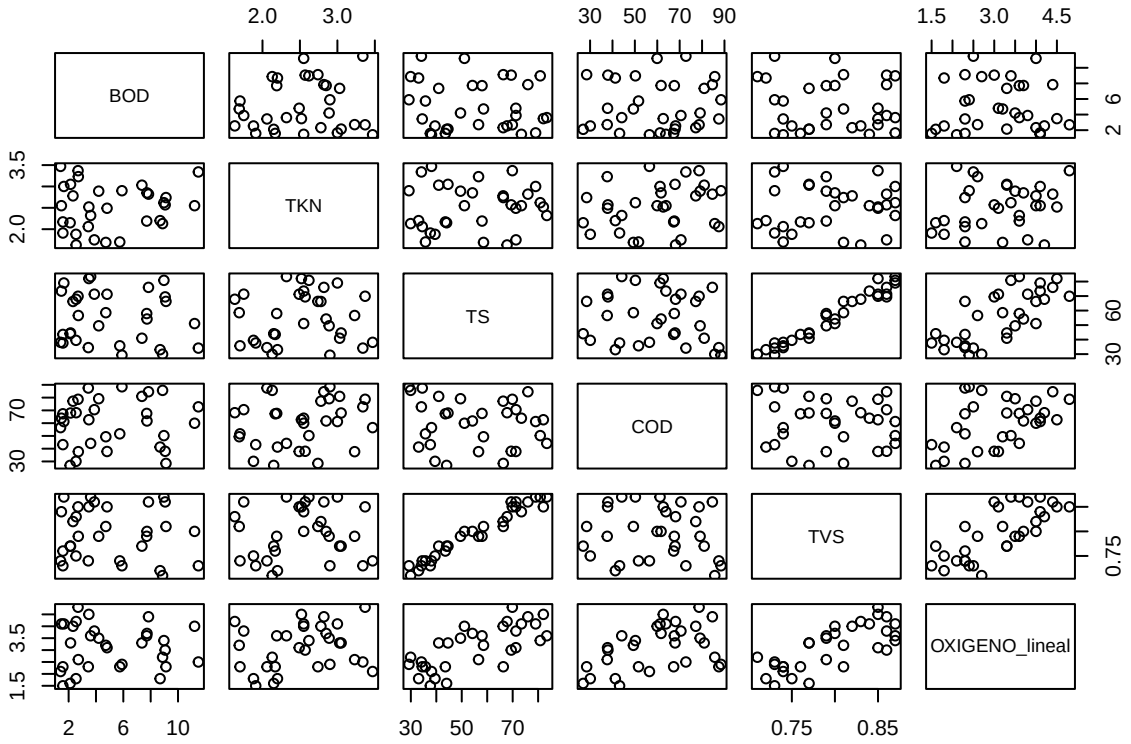



```
#Para ver las primeras filas de datos haremos View(datos_lin) o (datos_lin)
(datos_lin)
```

```
## # A tibble: 32 x 6
##      BOD   TKN    TS   COD   TVS OXIGENO_lineal
##    <dbl> <dbl> <dbl> <dbl> <dbl>          <dbl>
##  1  2.54  1.88  39.5  30.1  0.75            1.8
##  2  5.74  1.7   35.7  51.6  0.74            2.3
##  3  7.73  2.85  54.2  61.7  0.8             3.7
##  4  4.2   2.89  49.5  79.1  0.79            3.5
##  5  2.71  3.23  56.6  37.7  0.79            2.6
##  6  8.69  2.2   33.1  41.3  0.72            1.8
##  7  4.81  2.49  71.3  37.8  0.85            3.1
##  8  5.9   2.9   29.3  88.5  0.73            2.4
##  9  1.58  1.91  37.7  43.2  0.73            1.5
## 10  2.14  3.05  44.6  68.0  0.77            3.3
## # ... with 22 more rows
```

Para comprobar que ya no hay observaciones atípicas haremos:

```
plot(datos_lin)
```



2. Estimación de los parámetros del modelo RLM y métodos de selección de regresores

Para realizar la estimación de los coeficientes del modelo RLM podemos usar la función *lm()* de R. Esta función requiere indicar la variable respuesta y la fórmula que la relaciona con los predictores. Indicar que si se aplica la función *lm()* a un dataframe sin especificar fórmula, se toma como variable respuesta a la primera columna.

Comenzamos realizando el ajuste del modelo RLM usando todos los predictores:

```
modelo_completo <- lm(OXIGENO_lineal ~ BOD + TKN + TS + COD + TVS ,
                      data = datos_lin)

summary(modelo_completo)
```

```
##
## Call:
## lm(formula = OXIGENO_lineal ~ BOD + TKN + TS + COD + TVS, data = datos_lin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.64258 -0.25655  0.03202  0.24172  0.86486
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.284482   3.999749  -1.321    0.198
```

```
## BOD          0.014937   0.022649   0.659    0.515
## TKN          0.056763   0.137235   0.414    0.683
## TS           0.023260   0.019034   1.222    0.233
## COD          0.026681   0.003996   6.677 4.39e-07 ***
## TVS          6.651270   6.367052   1.045    0.306
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3871 on 26 degrees of freedom
## Multiple R-squared:  0.8505, Adjusted R-squared:  0.8218
## F-statistic: 29.59 on 5 and 26 DF,  p-value: 6.004e-10
```

```
#También se puede usar la siguiente fórmula para indicar que consideramos como regresores todas las var
# modelo_completo <- lm(OXIGENO_lineal ~. , data = datos_lin)
```

Por lo tanto, la variable respuesta se predice con la fórmula: $OXIGENO\_lineal = -5.28448 + 0.01494 \cdot BOD + 0.05676 \cdot TKN + 0.02326 \cdot TS + 0.02668 \cdot COD + 6.65127 \cdot TVS$

La bondad del ajuste se mide con el valor de R-cuadrado (multiple R-squared) o el R-cuadrado ajustado (adjusted R-squared), que al ser superiores a 0.8 nos indican un buen ajuste. Recordemos que es preferible usar el R-cuadrado ajustado cuando pretendemos comparar la bondad del ajuste de modelos con distinto número de predictores (regresores).

Además de la estimación de los coeficientes de regresión, obtenemos los p-valores (última columna) correspondientes a los contrastes sobre si los coeficientes son significativos o no,

$$\left.\begin{array}{l} H_0 : \theta_i = 0 \\ H_1 : \theta_i \neq 0 \end{array}\right\}$$

Observamos que existen varios predictores con p-valores altos (superiores a 0.10), indicativo de que el modelo es reducible.

Aplicaremos los métodos de selección de regresores *backward*, *fordward* y *stepwise*. Por ejemplo, en el método *backward* (regresión hacia atrás), se parte del modelo completo y en cada iteración se saca del modelo el predictor que más mejora la bondad del ajuste atendiendo al criterio de Akaike (menor AIC).

```
#Backward Regression
modelo_backward <- step(modelo_completo, direction = "backward")
```

```
## Start:  AIC=-55.38
## OXIGENO_lineal ~ BOD + TKN + TS + COD + TVS
##
##        Df Sum of Sq     RSS     AIC
## - TKN   1    0.0256  3.9219 -57.173
## - BOD   1    0.0652  3.9614 -56.852
## - TVS   1    0.1635  4.0598 -56.068
## - TS    1    0.2238  4.1200 -55.596
## <none>               3.8962 -55.383
## - COD   1    6.6814 10.5776 -25.424
##
## Step:  AIC=-57.17
## OXIGENO_lineal ~ BOD + TS + COD + TVS
##
##        Df Sum of Sq     RSS     AIC
## - BOD   1    0.0742  3.9961 -58.574
## - TVS   1    0.1689  4.0907 -57.824
## - TS    1    0.2218  4.1437 -57.413
## <none>               3.9219 -57.173
```

```
## - COD   1    7.2445 11.1664 -25.690
##
## Step:  AIC=-58.57
## OXIGENO_lineal ~ TS + COD + TVS
##
##        Df Sum of Sq     RSS     AIC
## - TS    1    0.1921  4.1882 -59.071
## - TVS   1    0.1981  4.1941 -59.026
## <none>               3.9961 -58.574
## - COD   1    7.2652 11.2613 -27.420
##
## Step:  AIC=-59.07
## OXIGENO_lineal ~ COD + TVS
##
##        Df Sum of Sq     RSS     AIC
## <none>               4.1882 -59.071
## - COD   1    7.0916 11.2798 -29.367
## - TVS   1   16.6672 20.8554  -9.700
```

Observamos que primero se elimina la variable TKN (la que mayor p-valor tenía), después BOD y finalmente TS, quedándose con solo dos variables COD y TVS. Ahora los coeficientes serían:

```
modelo_backward$coefficients
```

```
## (Intercept)         COD         TVS
## -9.80420736  0.02615863 14.25740459
```

Para aplicar los modelos forward y stepwise, tendremos que indicar que partiremos de un modelo que contempla sólo la constante y que llegaremos a lo sumo al modelo completo con todos los predictores.

```
#Ajuste usando solo la cte
modelo_cte <- lm(OXIGENO_lineal ~ 1 , data = datos_lin)

#Forward Regression
modelo_forward <- step(modelo_cte, direction = "forward",
                       scope = formula(modelo_completo))
```

```
## Start:  AIC=-4.56
## OXIGENO_lineal ~ 1
##
##        Df Sum of Sq    RSS      AIC
## + TVS   1   14.7890 11.280 -29.3672
## + TS    1   13.8740 12.195 -26.8713
## + COD   1    5.2133 20.855  -9.6999
## <none>              26.069  -4.5600
## + TKN   1    1.2842 24.785  -4.1765
## + BOD   1    0.0000 26.069  -2.5600
##
## Step:  AIC=-29.37
## OXIGENO_lineal ~ TVS
##
##        Df Sum of Sq     RSS     AIC
## + COD   1    7.0916  4.1882 -59.071
## <none>              11.2798 -29.367
## + TKN   1    0.6433 10.6365 -29.246
## + BOD   1    0.1050 11.1748 -27.666
```

```
## + TS    1    0.0185 11.2613 -27.420
##
## Step:  AIC=-59.07
## OXIGENO_lineal ~ TVS + COD
##
##        Df Sum of Sq    RSS     AIC
## <none>              4.1882 -59.071
## + TS    1  0.192147 3.9961 -58.574
## + BOD   1  0.044543 4.1437 -57.413
## + TKN   1  0.030380 4.1578 -57.304
```

```
modelo_forward$coefficients
```

```
## (Intercept)         TVS         COD
## -9.80420736 14.25740459  0.02615863
```

Observamos que se obtiene el mismo modelo con dos variables que se obtuvo con el método backward.

Finalmente, el método stepwise permite ir en ambas direcciones (añadiendo o quitando variables).

```
#Stepwise Regression
modelo_stepwise <- step(modelo_cte, direction = "both",
                        scope = formula(modelo_completo))
```

```
## Start:  AIC=-4.56
## OXIGENO_lineal ~ 1
##
##        Df Sum of Sq    RSS      AIC
## + TVS   1   14.7890 11.280 -29.3672
## + TS    1   13.8740 12.195 -26.8713
## + COD   1    5.2133 20.855  -9.6999
## <none>              26.069  -4.5600
## + TKN   1    1.2842 24.785  -4.1765
## + BOD   1    0.0000 26.069  -2.5600
##
## Step:  AIC=-29.37
## OXIGENO_lineal ~ TVS
##
##        Df Sum of Sq     RSS     AIC
## + COD   1    7.0916  4.1882 -59.071
## <none>              11.2798 -29.367
## + TKN   1    0.6433 10.6365 -29.246
## + BOD   1    0.1050 11.1748 -27.666
## + TS    1    0.0185 11.2613 -27.420
## - TVS   1   14.7890 26.0688  -4.560
##
## Step:  AIC=-59.07
## OXIGENO_lineal ~ TVS + COD
##
##        Df Sum of Sq     RSS     AIC
## <none>               4.1882 -59.071
## + TS    1    0.1921  3.9961 -58.574
## + BOD   1    0.0445  4.1437 -57.413
## + TKN   1    0.0304  4.1578 -57.304
## - COD   1    7.0916 11.2798 -29.367
## - TVS   1   16.6672 20.8554  -9.700
```

```
modelo_stepwise$coefficients
```

```
## (Intercept)         TVS         COD
## -9.80420736 14.25740459  0.02615863
```

Obsérvese que en este caso se ha llegado al mismo modelo por los tres métodos, auque esto no ocurre siempre. El modelo final propuesto por los tres métodos contempla sólo a los predictores TVS y COD. Teniendo en cuenta la estrecha relación lineal entre los regresores TS y TVS, propondremos como alternativa el modelo que contempla los predictores TS y COD, para comparar con el modelo teórico usado para simular los datos (véase sección 1).

```
modelo_final_1 <- lm(OXIGENO_lineal ~ TVS + COD, data = datos_lin)
summary(modelo_final_1)
```

```
##
## Call:
## lm(formula = OXIGENO_lineal ~ TVS + COD, data = datos_lin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.73515 -0.24227  0.00034  0.25749  0.83060
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.804207   1.103059  -8.888 8.91e-10 ***
## TVS         14.257405   1.327161  10.743 1.27e-11 ***
## COD          0.026159   0.003733   7.007 1.05e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.38 on 29 degrees of freedom
## Multiple R-squared:  0.8393, Adjusted R-squared:  0.8283
## F-statistic: 75.75 on 2 and 29 DF,  p-value: 3.06e-12
```

```
modelo_final_2 <- lm(OXIGENO_lineal ~ TS + COD, data = datos_lin)
summary(modelo_final_2)
```

```
##
## Call:
## lm(formula = OXIGENO_lineal ~ TS + COD, data = datos_lin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70013 -0.27357 -0.05773  0.19435  1.06548
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.910691   0.342239  -2.661   0.0126 *
## TS           0.042535   0.003963  10.733 1.29e-11 ***
## COD          0.027929   0.003755   7.438 3.39e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3803 on 29 degrees of freedom
## Multiple R-squared:  0.8391, Adjusted R-squared:  0.828
## F-statistic: 75.63 on 2 and 29 DF,  p-value: 3.123e-12
```

Obsérvese que todos los regresores del modelo final son significativos. Por ejemplo, para el modelo_final_1 la significación (p-valor del contraste) correspondiente a las variables TVS y COD es prácticamente cero, y lo mismo ocurre con TS y COD para el modelo_final_2. Por tanto, estos modelos ya no serían reducibles.

La **bondad del ajuste** del modelo se puede medir a través del "Multiple R-squared" (coeficiente de determinación o R-cuadrado). Observamos bondades del ajuste altas en ambos casos (0.8393 para modelo_final_1 y 0.8393 para modelo_final_2), es decir, los modelos de regresión obtenidos explican aproximadamente el 84% de la variabilidad del oxígeno consumido por los microorganismos.

3. Inferencias en el modelo RLM. Predicciones

Una vez validado el modelo RLM, lo cual se realizará en una sección posterior, tenemos garantías para realizar inferencias con dicho modelo. Estas inferencias incluyen la obtención de intervalos de confianza para los parámetros de regresión, intervalos de confianza para la media de la variable respuesta e intervalos de predicción para la respuesta.

```
#Intervalos de confianza al 95% para los parámetros del modelo_final_1
confint(modelo_final_1, level=0.95)
```

```
##                    2.5 %     97.5 %
## (Intercept) -12.06021650 -7.5481982
## TVS          11.54305532 16.9717539
## COD           0.01852376  0.0337935
```

```
#Intervalos de confianza al 99% para los parámetros del modelo_final_2
confint(modelo_final_2, level=0.99)
```

```
##                   0.5 %     99.5 %
## (Intercept) -1.85403516 0.03265277
## TS           0.03161208 0.05345884
## COD          0.01757861 0.03827909
```

Por tanto, cualquier modelo teórico con coeficientes incluidos en los intervalos anteriores serían también válidos teniendo en cuenta nuestros datos muestrales. En particular, fijado un nivel de confianza del 99%, el modelo simulado con el que se generaron los datos de nuestro fichero debería aceptarse como posible modelo que explica la relación entre el oxígeno consumido y el resto de variabes químicas, ya que cada uno de los tres coeficientes del modelo (θ_0 = -0.3, θ_1 = 0.04 y θ_2 = 0.02) están contenidos en el correspondiente intervalo de confianza del **modelo_final_2**.

Para la predicción de nuevos valores utilizaremos la función **predict** que permitirá obtener tanto valores de predicción como intervalos de confianza para el valor medio de la respuesta y para el valor de la respuesta utilizando la opción en el argumento **interval = "confidence"** e **interval = "prediction"**, respectivamente. Así por ejemplo, para los valores TVS = 0.85 y COD = 70,

```
#Predicciones
nuevos <- data.frame(TVS = 0.85, COD = 70)
predict(modelo_final_1, newdata = nuevos, interval = "confidence", level = 0.95)
```

```
##        fit     lwr      upr
## 1 4.145691 3.92644 4.364941
```

```
predict(modelo_final_1, newdata = nuevos, interval = "prediction", level = 0.95)
```

```
##        fit      lwr      upr
## 1 4.145691 3.338115 4.953266
```

Una vez validado el modelo, podemos decir que el valor esperado para el oxígeno (cuando el agua tiene esos valores de TVS y COD) estaría entre 3.92644 y 4.364941, y su predicción entre 3.338115 4.953266, con una

confianza del 95%.

4. Validación del modelo

De forma resumida, para que la inferencia y las predicciones realizadas anteriormente sean válidas, debemos verificar:

1) Hipótesis de Normalidad: los residuos siguen una distribución Normal. Podemos analizarlo con el test de Shapiro o bien con gráfico de cuantiles.

2) Hipótesis de Homocedasticidad: varianza constante en las perturbaciones aleatorias. Podemos analizarlo realizando un gráfico de residuos frente a valores ajustados. Dicho gráfico debe presentar un comportamiento aleatorio con dispersión aproximadamente constante.

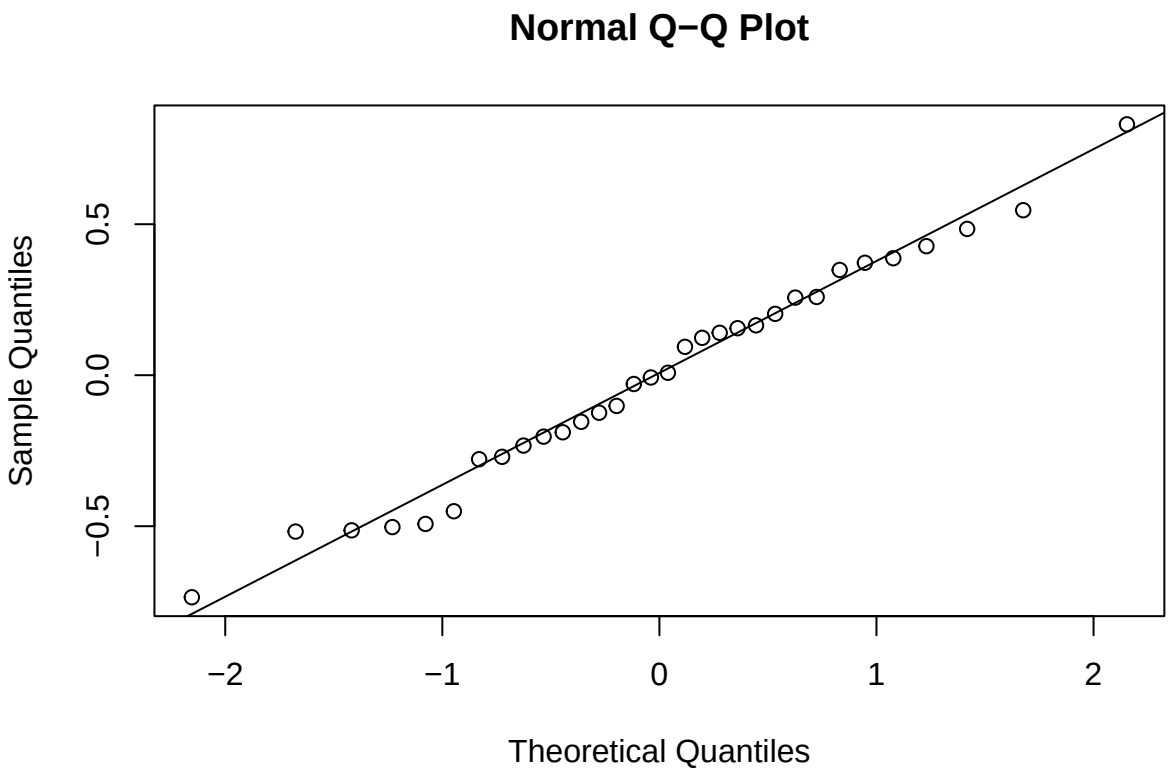
3) Hipótesis de Independencia: los residuos son independientes. Podemos analizarlo representando la serie temporal de residuos o con el test de Durbin-Watson.

Además, debemos verificar que no se da el problema de multicolinealidad entre los predictores del modelo y que no haya observaciones influyentes.

```
# 1) Hipótesis de Normalidad
shapiro.test(modelo_final_1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  modelo_final_1$residuals
## W = 0.98468, p-value = 0.9182
```

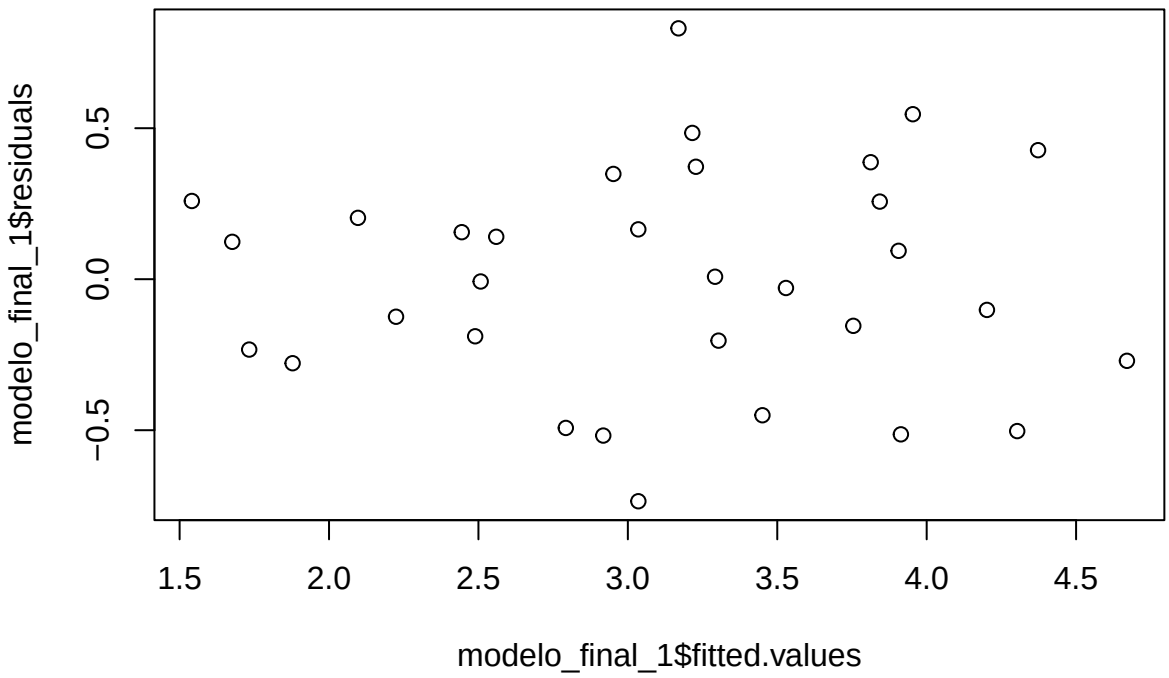
```
qqnorm(modelo_final_1$residuals)
qqline(modelo_final_1$residuals)
```



Así comprobamos que los residuos pasan el test de normalidad. Para ver los residuos podemos hacer:

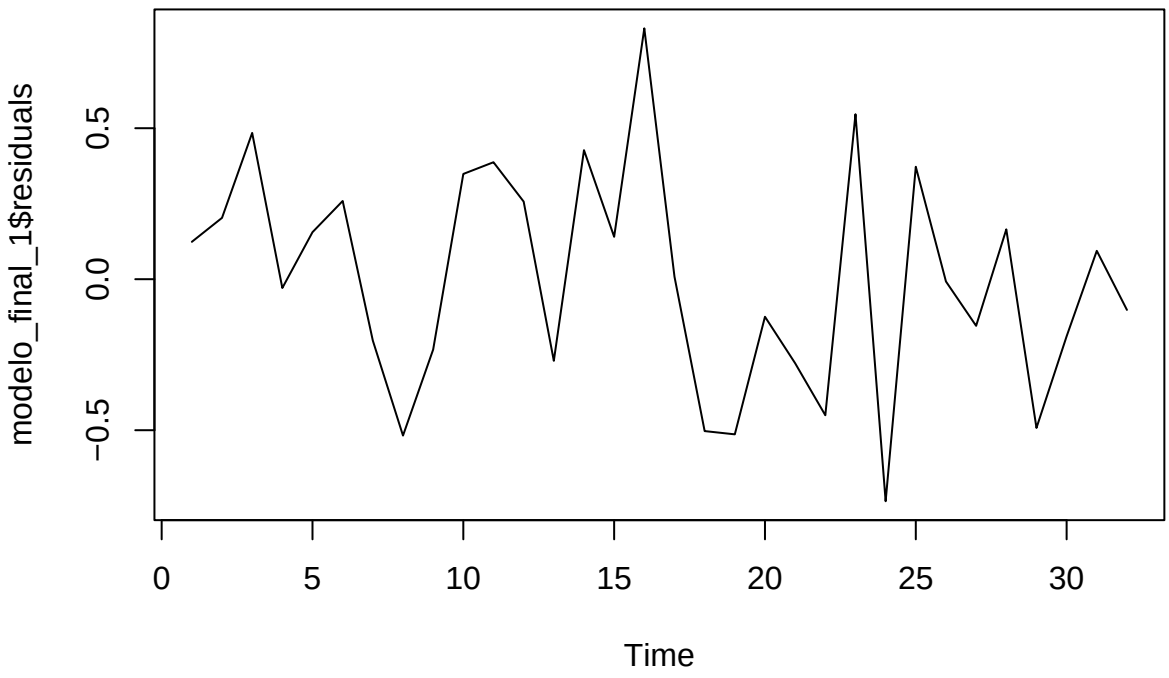
```
# plot(modelo_final_1$residuals)
```

```
# 2) Hipótesis de Homocedasticidad
plot(modelo_final_1$fitted.values, modelo_final_1$residuals)
```



Con el gráfico anterior observamos que los residuos no dependen de la magnitud de la variable predicha.

```
# 3) Hipótesis de Independencia
ts.plot(modelo_final_1$residuals)
library("lmtest")
```




```
dwtest(modelo_final_1, alternative="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  modelo_final_1
## DW = 2.0014, p-value = 0.9628
## alternative hypothesis: true autocorrelation is not 0
```

Con el gráfico anterior observamos que los residuos no dependen de la fila que ocupan en el dataframe y el p-valor alto (superior a 0.10) del contraste de Durbin-Watson nos indica que podemos suponer independencia.

Para comprobar la multicolinealidad, calcularemos el factor de varianza inflada (VIF) para cada predictor. Pretende medir la relación lineal de cada predictor con el resto de predictores. Como regla empírica, valores del VIF superiores a 7 son indicativos de existencia de multicolinealidad.

```
#Comprobamos que no hay multicolinealidad en el modelo final calculando los VIFs
library("rms")
vif(modelo_final_1)
```

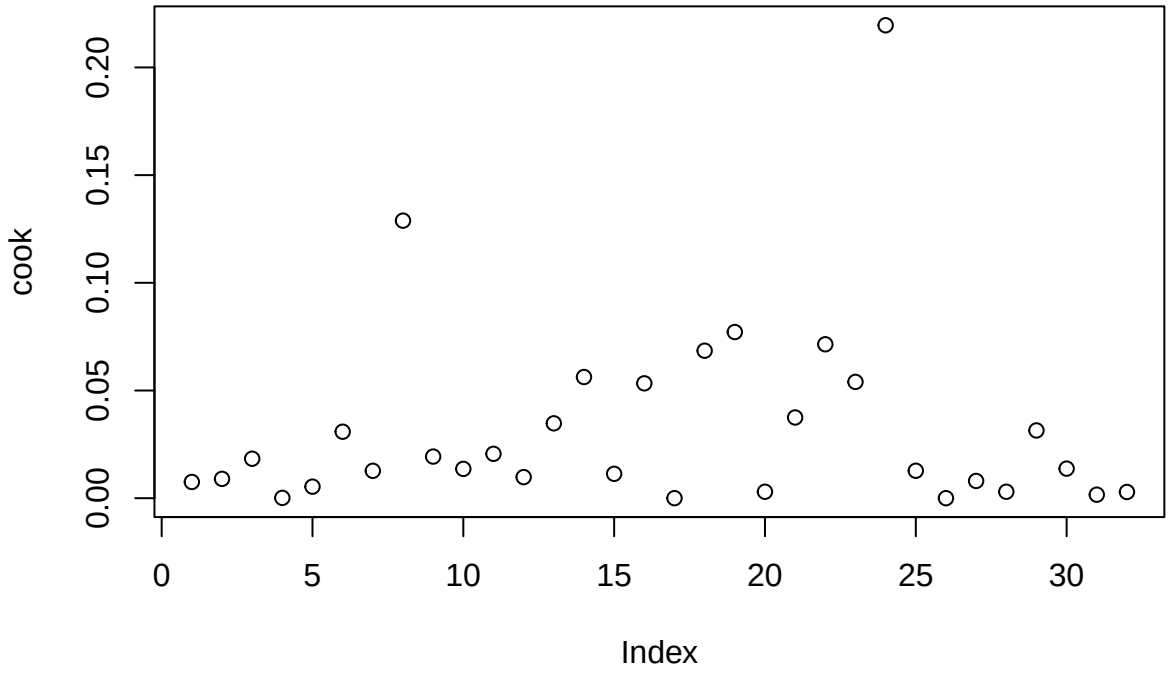
```
##      TVS      COD
## 1.009218 1.009218
```

```
#Sí existe multicolinealidad en el modelo completo
#(recordar que TS y TVS tienen fuerte relación lineal)
vif(modelo_completo)
```

```
##       BOD       TKN        TS       COD       TVS
##  1.037482  1.079131 22.701974  1.114348 22.385844
```

Por último, para comprobar que no hay observaciones influyentes, podemos calcular la distancia de Cook para cada observación. Como regla empírica, valores por encima de 1 indican que se trata de una observación influyente. En general, conviene representar los valores de la distancia de Cook con el fin de identificar si hay alguna observación con valores especialmente altos comparados con el resto.

```
cook <- cooks.distance(modelo_final_1)
plot(cook)
```



En resumen, viendo los resultados de los procedimientos anteriores, podemos asumir la validez del modelo.

Nota: Se deja al lector como ejercicio comprobar que si no se hubiera eliminado la observación número 33 al comienzo del análisis, entonces se obtendría una distancia de Cook elevada para dicha observación indicando que es influyente. Recordar que las observaciones influyentes sí deben eliminarse del modelo con el fin de disponer de modelos fiables.

5. Transformaciones de los datos (familia Box-Cox)

Para identificar la transformación más adecuada sobre la variable respuesta, usaremos la función *boxcox()* del paquete MASS. Indicar que la familia de transformaciones de Box-Cox consiste en elevar la variable respuesta a un exponente $\lambda > 0$, o bien tomar logaritmos neperianos si resulta $\lambda = 0$.

Primero eliminaremos la fila número 33 de los otros dataframes a analizar:

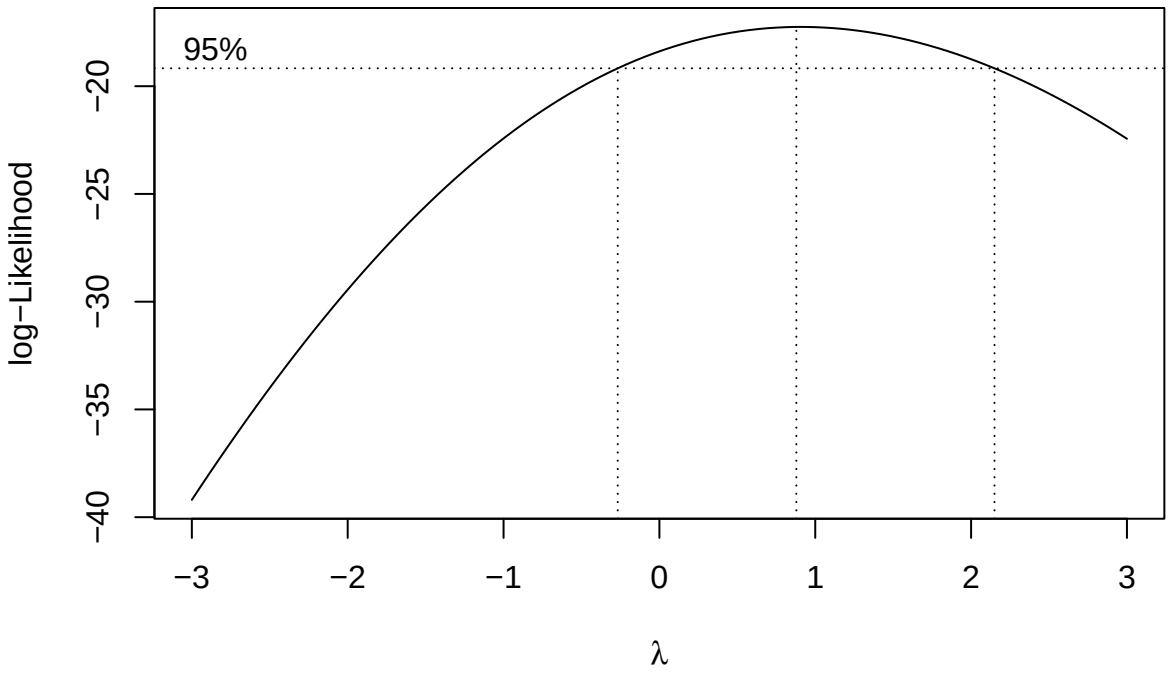
```
datos_cub <- datos_cub[-33, ]
datos_exp <- datos_exp[-33, ]
```

Para el dataframe **"datos_lin"**, representamos la verosimilitud para diferentes valores del exponente lambda (λ) en la familia de transformaciones de Box-Cox:

```r
library("MASS")
```

```
## Warning: package 'MASS' was built under R version 4.1.3
```

```r
boxcox(lm(OXIGENO_lineal ~ 1, data = datos_lin), lambda = seq(-3, 3, 1/10))
```



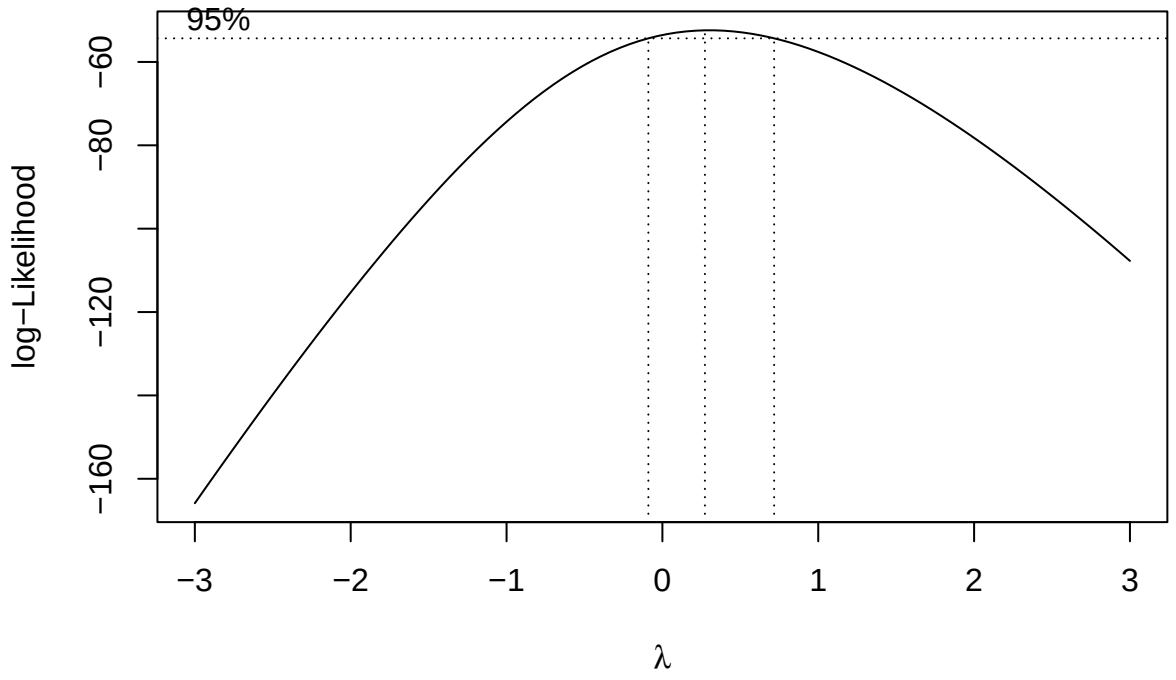

```r
boxcox(lm(OXIGENO_lineal ~ 1, data = datos_lin), lambda = seq(-3, 3, 1/10),
       plotit = FALSE)
```

```
## $x
##  [1] -3.0 -2.9 -2.8 -2.7 -2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -1.9 -1.8 -1.7 -1.6
## [16] -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1
## [31]  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3  1.4
## [46]  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9
## [61]  3.0
##
## $y
##  [1] -39.19413 -38.10813 -37.04540 -36.00646 -34.99183 -34.00199 -33.03746
##  [8] -32.09869 -31.18617 -30.30034 -29.44162 -28.61043 -27.80714 -27.03212
## [15] -26.28571 -25.56819 -24.87986 -24.22096 -23.59169 -22.99224 -22.42274
## [22] -21.88332 -21.37405 -20.89498 -20.44610 -20.02740 -19.63881 -19.28025
## [29] -18.95160 -18.65270 -18.38338 -18.14343 -17.93261 -17.75067 -17.59733
## [36] -17.47229 -17.37524 -17.30584 -17.26374 -17.24857 -17.25996 -17.29753
## [43] -17.36088 -17.44961 -17.56331 -17.70156 -17.86396 -18.05008 -18.25951
## [50] -18.49182 -18.74660 -19.02342 -19.32187 -19.64153 -19.98200 -20.34286
## [57] -20.72372 -21.12418 -21.54383 -21.98230 -22.43920
```

Para el modelo lineal, se observa que la trasformacion identidad es adecuada (para lambda igual a 1 se alcanza aproximadamente la máxima verosimilitud). Por tanto, no hay que transformar la variable respuesta OXIGENO_lineal.

Repetimos el proceso para el dataframe **"datos_cub"**:

```
boxcox(lm(OXIGENO_cubica ~ 1, data = datos_cub), lambda = seq(-3, 3, 1/10))
```




```
boxcox(lm(OXIGENO_cubica ~ 1, data = datos_cub), lambda = seq(-3, 3, 1/10),
       plotit = FALSE)
```

```
## $x
##  [1] -3.0 -2.9 -2.8 -2.7 -2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -1.9 -1.8 -1.7 -1.6
## [16] -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1
## [31]  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3  1.4
## [46]  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9
## [61]  3.0
##
## $y
##  [1] -165.86359 -160.55043 -155.28402 -150.06753 -144.90443 -139.79856
##  [7] -134.75411 -129.77574 -124.86861 -120.03841 -115.29146 -110.63478
## [13] -106.07612 -101.62405  -97.28802  -93.07839  -89.00643  -85.08431
## [19]  -81.32498  -77.74210  -74.34972  -71.16206  -68.19305  -65.45593
## [25]  -62.96273  -60.72379  -58.74728  -57.03892  -55.60169  -54.43585
## [31]  -53.53898  -52.90626  -52.53084  -52.40416  -52.51647  -52.85716
## [37]  -53.41510  -54.17901  -55.13759  -56.27977  -57.59479  -59.07230
## [43]  -60.70242  -62.47574  -64.38340  -66.41700  -68.56868  -70.83106
## [49]  -73.19724  -75.66076  -78.21562  -80.85622  -83.57736  -86.37419
```

```
## [55]  -89.24221  -92.17723  -95.17538  -98.23303 -101.34682 -104.51364
## [61] -107.73055
```

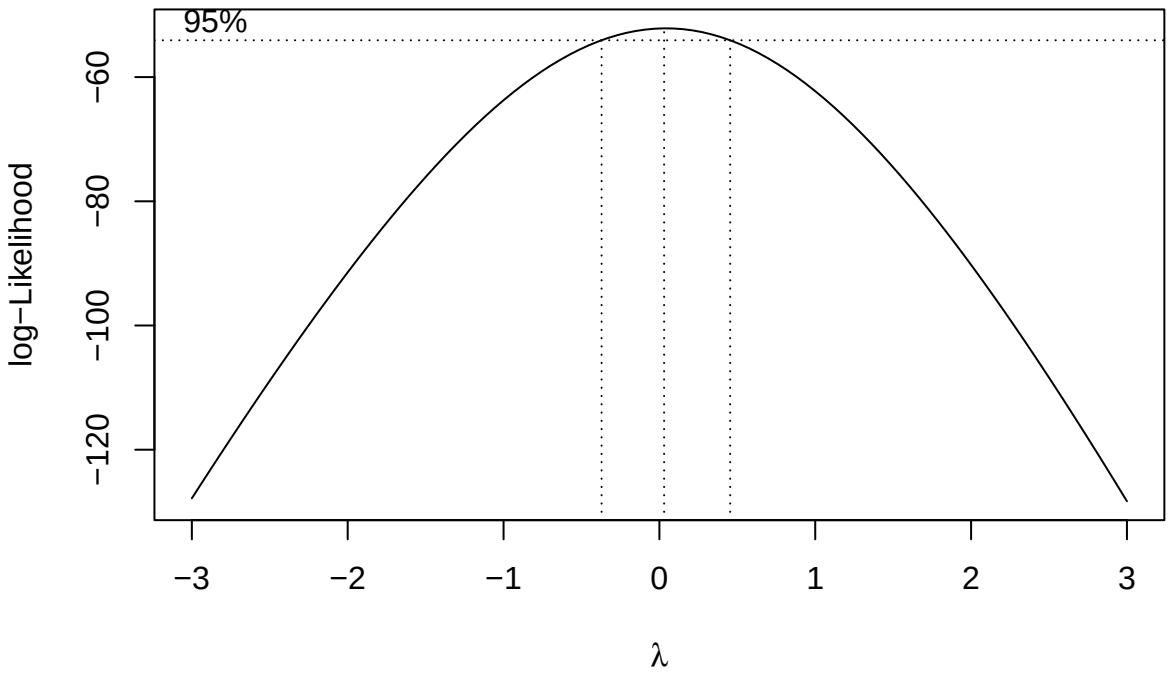
En este caso se observa que la trasformacion adecuada es tomar raíz cúbica sobre la variable respuesta (para lambda igual a 1/3 aproximadamente se alcanza la máxima verosimilitud). Por tanto, hay que transformar la variable respuesta OXIGENO_cubica elevándola a 1/3. Esa variable transformada (que llamaremos por ejemplo Ynew), será nuestra nueva variable respuesta para ajustar el modelo RLM del dataframe **"datos_cub"**. Lógicamente, tras esta transformación se obtiene el mismo modelo del caso lineal anterior (véase la sección 1).

```
datos_cub$Ynew <- datos_cub$OXIGENO_cubica ^ (1/3)
modelo_cubico <- lm(Ynew ~ TVS + COD, data = datos_cub)
summary(modelo_cubico)
```

```
##
## Call:
## lm(formula = Ynew ~ TVS + COD, data = datos_cub)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.73515 -0.24227  0.00034  0.25749  0.83060
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.804207   1.103059  -8.888 8.91e-10 ***
## TVS         14.257405   1.327161  10.743 1.27e-11 ***
## COD          0.026159   0.003733   7.007 1.05e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.38 on 29 degrees of freedom
## Multiple R-squared:  0.8393, Adjusted R-squared:  0.8283
## F-statistic: 75.75 on 2 and 29 DF,  p-value: 3.06e-12
```

Repetimos el proceso para el dataframe "datos_exp":

```
boxcox(lm(OXIGENO_exponencial ~ 1, data = datos_exp), lambda = seq(-3, 3, 1/10))
```




```
boxcox(lm(OXIGENO_exponencial ~ 1, data = datos_exp), lambda = seq(-3, 3, 1/10),
       plotit = FALSE)
```

```
## $x
##  [1] -3.0 -2.9 -2.8 -2.7 -2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -1.9 -1.8 -1.7 -1.6
## [16] -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1
## [31]  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3  1.4
## [46]  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9
## [61]  3.0
##
## $y
##  [1] -127.82646 -123.94369 -120.10585 -116.31592 -112.57713 -108.89302
##  [7] -105.26744 -101.70456  -98.20895  -94.78555  -91.43970  -88.17718
## [13]  -85.00421  -81.92743  -78.95394  -76.09126  -73.34729  -70.73032
## [19]  -68.24891  -65.91190  -63.72824  -61.70694  -59.85697  -58.18708
## [25]  -56.70571  -55.42083  -54.33979  -53.46920  -52.81475  -52.38110
## [31]  -52.17180  -52.18910  -52.43398  -52.90603  -53.60350  -54.52328
## [37]  -55.66101  -57.01117  -58.56722  -60.32173  -62.26661  -64.39324
## [43]  -66.69265  -69.15573  -71.77331  -74.53634  -77.43596  -80.46360
## [49]  -83.61106  -86.87051  -90.23454  -93.69617  -97.24886 -100.88650
## [55] -104.60339 -108.39423 -112.25410 -116.17844 -120.16303 -124.20396
## [61] -128.29760
```

En este caso se observa que la trasformacion adecuada es tomar logaritmos neperianos sobre la variable respuesta (para lambda igual a cero aproximadamente se alcanza la máxima verosimilitud). Por tanto, hay que transformar la variable respuesta OXIGENO_exponencial tomando logaritmos neperianos. Esa variable transformada (que llamaremos por ejemplo Ynew), será nuestra nueva variable respuesta para ajustar el

modelo RLM del dataframe **"datos_exp"**. Lógicamente, tras esta transformación se obtiene el mismo modelo del caso lineal (véase la sección 1).

```
datos_exp$Ynew <- log(datos_exp$OXIGENO_exponencial)
modelo_exponencial <- lm(Ynew ~ TVS + COD, data = datos_exp)
summary(modelo_exponencial)
```

```
##
## Call:
## lm(formula = Ynew ~ TVS + COD, data = datos_exp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.73515 -0.24227  0.00034  0.25749  0.83060
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.804207   1.103059  -8.888 8.91e-10 ***
## TVS         14.257405   1.327161  10.743 1.27e-11 ***
## COD          0.026159   0.003733   7.007 1.05e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.38 on 29 degrees of freedom
## Multiple R-squared:  0.8393, Adjusted R-squared:  0.8283
## F-statistic: 75.75 on 2 and 29 DF,  p-value: 3.06e-12
```

6. Consideraciones adicionales

En ocasiones, necesitamos ajustar modelos con alguna o varias variables predictoras elevadas a una potencia, así como considerar interacciones entre las variables predictoras.

A continuación se muestran algunos ejemplos de este tipo de ajustes. Poniendo I(COD^3) se incluye esa potencia de la variable COD y poniendo pol(TVS,3) se incluye un polinomio de grado hasta tres de TVS.

```
#Modelo con predictores elevados a potencias
modelo_polinomial <- lm(OXIGENO_lineal ~ I(COD^3) + COD + pol(TVS,3),
                        data = datos_lin )

summary(modelo_polinomial)
```

```
##
## Call:
## lm(formula = OXIGENO_lineal ~ I(COD^3) + COD + pol(TVS, 3), data = datos_lin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.54546 -0.19858  0.00838  0.15891  0.45435
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        9.680e+02  2.978e+02   3.250  0.00318 **
## I(COD^3)          -3.220e-06  9.331e-07  -3.451  0.00192 **
## COD                5.965e-02  1.031e-02   5.786 4.28e-06 ***
## pol(TVS, 3)TVS    -3.747e+03  1.130e+03  -3.316  0.00270 **
```

```
## pol(TVS, 3)TVS^2  4.806e+03  1.426e+03   3.370  0.00236 **
## pol(TVS, 3)TVS^3 -2.043e+03  5.989e+02  -3.411  0.00213 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2731 on 26 degrees of freedom
## Multiple R-squared:  0.9256, Adjusted R-squared:  0.9113
## F-statistic: 64.71 on 5 and 26 DF,  p-value: 7.757e-14
```

Introduciendo el término COD:TVS en la fórmula, estamos indicando que se incluya interacción a través del predictor producto de COD por TVS.

```
#Modelo con interacción
modelo_interaccion <- lm(OXIGENO_lineal ~ COD + TVS + COD:TVS,
                         data = datos_lin)
summary(modelo_interaccion)
```

```
##
## Call:
## lm(formula = OXIGENO_lineal ~ COD + TVS + COD:TVS, data = datos_lin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.61088 -0.26012  0.01349  0.23259  0.81982
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.54088    3.60087  -0.983   0.3339
## COD         -0.07629    0.05639  -1.353   0.1870
## TVS          6.28440    4.56249   1.377   0.1793
## COD:TVS      0.13068    0.07179   1.820   0.0794 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3657 on 28 degrees of freedom
## Multiple R-squared:  0.8563, Adjusted R-squared:  0.8409
## F-statistic: 55.63 on 3 and 28 DF,  p-value: 6.431e-12
```

Una forma alternativa de ajustar el mismo modelo que antes sería:

```
#Fórmula alternativa
modelo_interaccion_2 <- lm(OXIGENO_lineal ~ COD*TVS,
                           data = datos_lin)
summary(modelo_interaccion_2)
```

```
##
## Call:
## lm(formula = OXIGENO_lineal ~ COD * TVS, data = datos_lin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.61088 -0.26012  0.01349  0.23259  0.81982
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.54088    3.60087  -0.983   0.3339
## COD         -0.07629    0.05639  -1.353   0.1870
```

```
## TVS          6.28440    4.56249   1.377   0.1793
## COD:TVS      0.13068    0.07179   1.820   0.0794 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3657 on 28 degrees of freedom
## Multiple R-squared:  0.8563, Adjusted R-squared:  0.8409
## F-statistic: 55.63 on 3 and 28 DF,  p-value: 6.431e-12
```

Obsérvese que la introducción de varias potencias de un mismo regresor puede dar lugar a problemas de multicolinealidad.

```
vif(modelo_polinomial)
```

```
##         I(COD^3)              COD   pol(TVS, 3)TVS pol(TVS, 3)TVS^2
##     1.534931e+01     1.490360e+01     1.416596e+06     5.718479e+06
## pol(TVS, 3)TVS^3
##     1.445463e+06
```

Por último, comentar que los paquetes *MASS*, *olsrr* y *StepReg* contemplan funciones para ajuste y selección del modelo. En particular, el paquete *olsrr* resulta bastante didáctico y completo para la validación y diagnosis del modelo. El uso de estos paquetes no se exigirá en la evaluación de esta práctica.

Importante: (1) Siempre se puede llegar a un ajuste perfecto sobre los datos aumentado los grados de los polinomios, pero que estos modelos darán peores predicciones que modelos más sencillos (sobreajuste). (2) No debemos realizar predicciones para valores muy alejados de nuestro rango de valores observados de los predictores (regresores).