241024_R_Básico_5

J.Ballesta

29/10/24

Resumen

En esta última publicación veremos de forma resumida las alternativas que ofrece R para hacer regresión polinomica y el package ggfortify para los graficos de análisis de los resultados de la regresión.

In this last publication, we will briefly review the alternatives that R offers for polynomial regression and the ggfortify package for visualizing the results of the regression analysis.

In dieser letzten Veröffentlichung werden wir kurz die Alternativen betrachten, die R für die polynomiale Regression bietet, sowie das Paket ggfortify zur Visualisierung der Ergebnisse der Regressionsanalyse.

Introducción

En la publicación anterior hemos visto de forma básica la regresión lineal mediante la función de *lm()* en dos casos :

- Regresión lineal simple: buscamos la relación entre dos variables con una ecuación de primer grado del tipo $y = a\,x + m + \epsilon$.
- Regresión lineal polinómica: buscamos la relación entre varias variables dependientes y una variable dependiente con una ecuación del tipo : $y = Coef_1 * Var_1 + Coef_2 * Var_2 + \ldots + Coef_n * Var_n + \epsilon$

R ((2024a)) mediante la función *summary()*, para cada tipo de regresión nos daba entre otros los resultados de :

- a : pendiente de la recta, variación de y por unidad de variación de x.
- m : intersección de la recta con el eje de ordenadas.
- $Coef_1, \ldots, Coef_n$ los distintos coeficientes para componer nuestra ecuación.

En determinados casos esta opción puede ser aceptable y dar el resultado que necesitamos, en otros casos debemos buscar la relación en forma de una ecuación de grado mayor que 1 o una alternativa en base de un spline.

En esta publicación veremos de forma resumida cómo empezar el estudio de este tipo de ecuaciones.

Librerías.

Como siempre, en primer lugar cargamos las librería necesarias que vamos a utilizar:

```
#
#
library(magrittr)   # para el uso del operador pipeline %>%
library(ggplot2)    # package de gráficos
theme_set(theme_bw()+
```

```
            theme(legend.position="bottom")) # por defecto los gráficos en B/N y leyenda en la parte inferior del gráfico
library(scales)     # ajustes en las escala de los gráficos
library (patchwork) # composición de los gráficos de  ggplot2
library(ggfortify)     # para gráficos del resultado del modelo con lm()
library(DataExplorer) # para análisis exploratorio de datos
library(splines)    # para el uso de la solución mediante splines
```

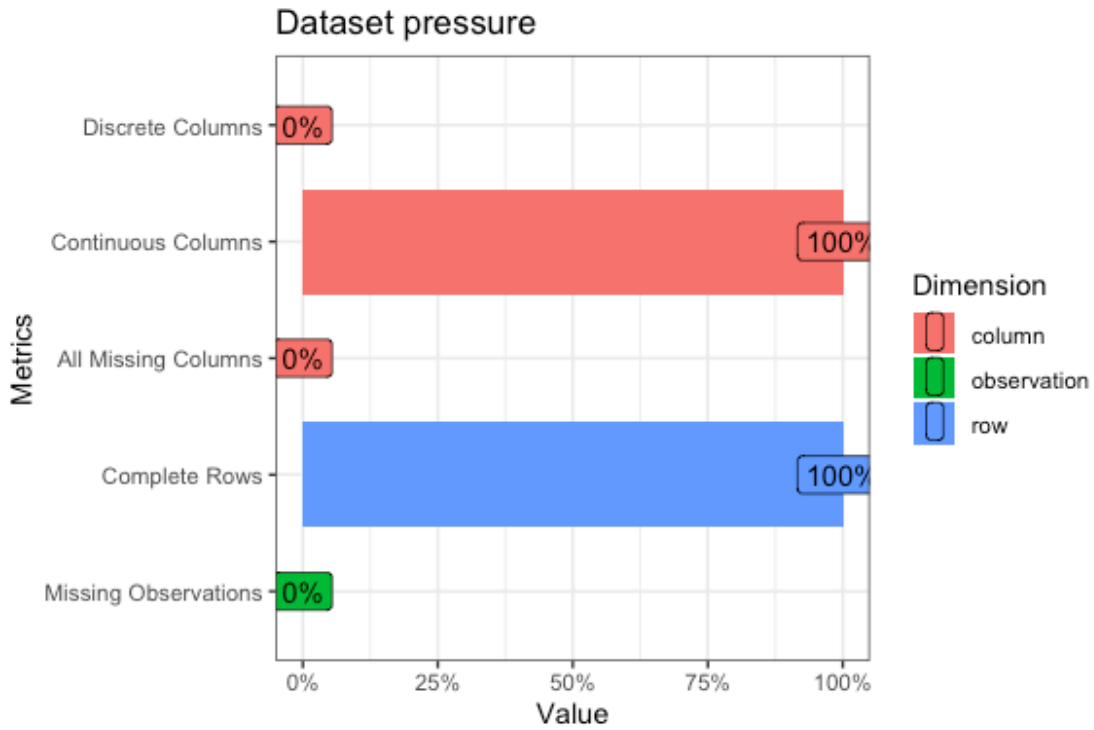
Conjuntos de datos.

Regresión polinómica y splines.

En ambos casos, haremos el estudio con el dataset *pressure* que contiene dos variables

- *$temperature* : la temperatura en grados celsius (ºC)
- *$presure* : la presión del vapor de mercurio en milímetros (de mercurio) (mm)

```
#
#    pasamos los datos del dataset a una variable de trabajo
datos_rp <- pressure
# mostramos el estado de los los datos
plot_intro(datos_rp,
           title = "Dataset pressure",
           ggtheme = theme_bw())
```




```
# su estructura
str(datos_rp)
```

```
'data.frame':   19 obs. of  2 variables:
 $ temperature: num  0 20 40 60 80 100 120 140 160 180 ...
 $ pressure   : num  0.0002 0.0012 0.006 0.03 0.09 0.27 0.75 1.85 4.2 8.8 ...
```

```
# un resumen de algunas magnitudes estadísticas
summary(datos_rp)
```

```
  temperature     pressure
 Min.   :  0   Min.   :  0.0002
 1st Qu.: 90   1st Qu.:  0.1800
 Median :180   Median :  8.8000
 Mean   :180   Mean   :124.3367
 3rd Qu.:270   3rd Qu.:126.5000
 Max.   :360   Max.   :806.0000
```

```
# mostramos los datos iniciales y finales
head(datos_rp, 5)
```

```
  temperature pressure
1           0   0.0002
2          20   0.0012
3          40   0.0060
4          60   0.0300
5          80   0.0900
```

```
tail(datos_rp, 5)
```

```
   temperature pressure
15         280      157
16         300      247
17         320      376
18         340      558
19         360      806
```

Regresión polinómica dos variables.

En primer lugar, presentamos de forma gráfica los datos disponibles en la variable de trabajo:

```
#
#    mediante el operador %>% pasamos los datos a ggplot
datos_rp%>%
    ggplot(aes(x=pressure, y=temperature))+
    geom_point(alpha=0.5)+
    labs (title = "Relación temperatura vs. presión de mercurio",
          subtitle = "Dataset: pressure")+
    xlab("Pressure Hg (mm)")+
    ylab("Temperature (ºC)")
```

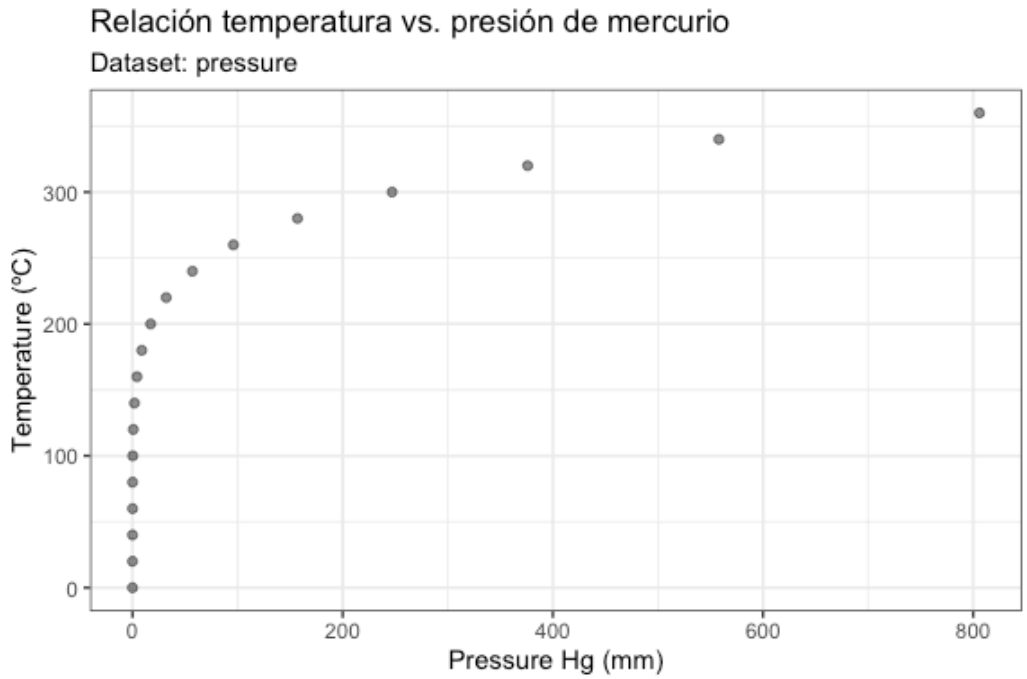


Figura 1: Relación presión y temperatura.

Aprovechando la capa *geom_smooth()* vamos a trazar varias curvas con polinomios de distintos grados para ver cual podría ajustarse mejor a nuestros datos en *pressure.*

```
#
#    pasamos los datos a ggplot y presentamos los gráficos
datos_rp%>%
    ggplot(aes(x=pressure, y=temperature))+
    geom_point(alpha=0.5)+
    geom_smooth(method = "lm", se=FALSE, aes(color="Reg.Lin"))+
    geom_smooth(method="lm", formula = "y~poly(x,2)", se=FALSE, aes(color="grad2"))+
    geom_smooth(method="lm", formula = "y~poly(x,3)", se=FALSE, aes(color="grad3"))+
    geom_smooth(method="lm", formula = "y~poly(x,4)", se=FALSE, aes(color="grad4"))+
    geom_smooth(method="lm", formula = "y~poly(x,5)", se=FALSE, aes(color="grad5"))+
    scale_color_manual(values=c("Reg.Lin"="blue",
                                "grad2"="red", "grad3"="green",
                                "grad4"="cyan", "grad5"="brown"))+
    labs (title = "Relación temperatura vs. presión de mercurio",
```

```
      subtitle = "Dataset: pressure")+
  xlab("Pressure Hg (mm)")+
  ylab("Temperature (ºC)")
```

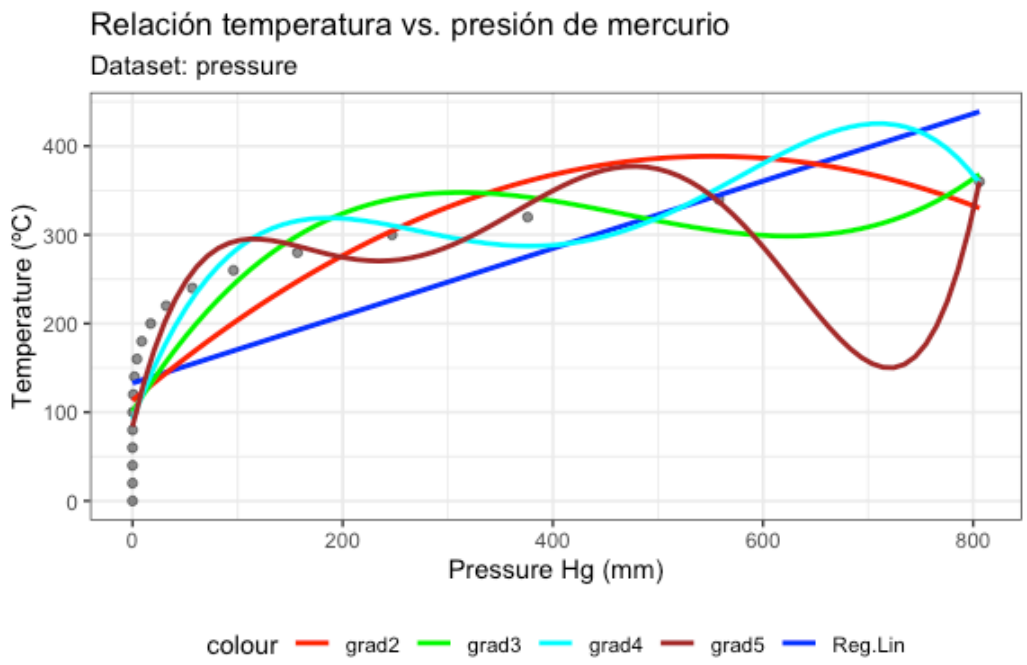


Figura 2: Ajuste polinomios diversos grados a dataset pressure.

Del gráfico Figura 2, podemos ver que el "mejor ajuste" a los datos originales se obtiene con un polinomio grado 4. Ahora usamos la función *lm()* para determinar el modelo:

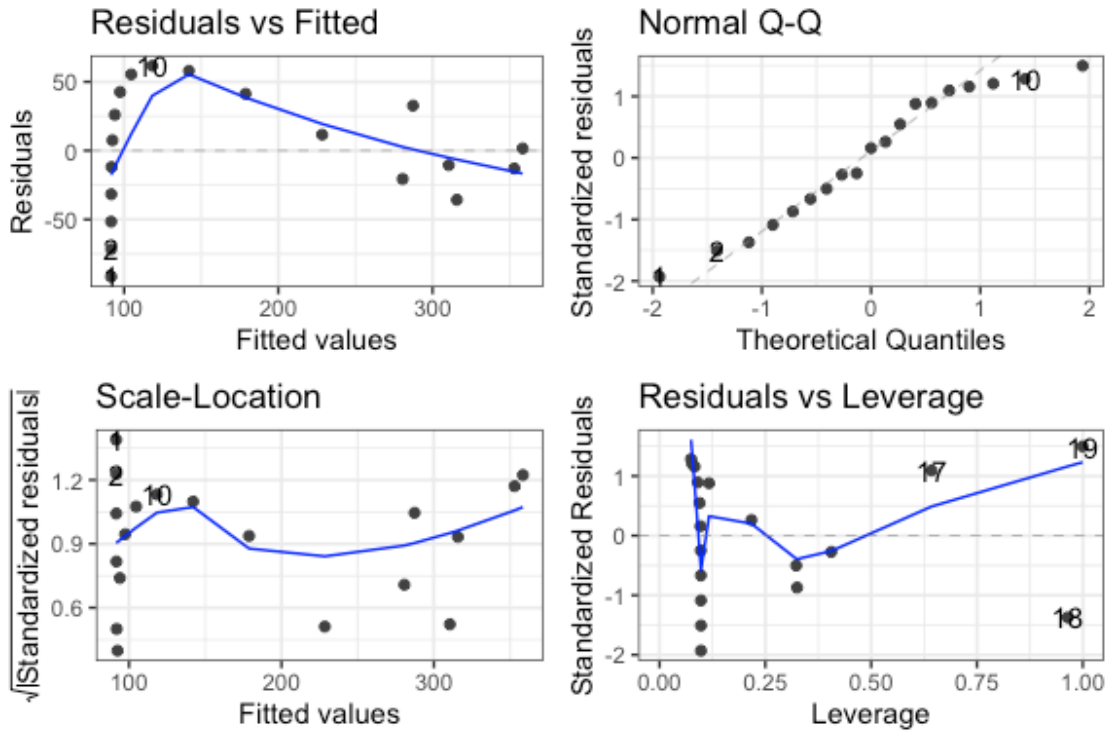
```
#
#    lanzamos con lm() el cálculo de la regresión con un polinomio grado 4
modelo_rlp <- lm (data=datos_rp, formula = temperature ~ poly(pressure, 4))
# vemos el resultado del calculo del modelo
summary (modelo_rlp)


Call:
lm(formula = temperature ~ poly(pressure, 4), data = datos_rp)

Residuals:
    Min      1Q  Median      3Q     Max
-91.606 -26.160   1.579  36.976  61.678

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           180.00      11.47  15.693 2.80e-10 ***
poly(pressure, 4)1    361.84      50.00   7.237 4.31e-06 ***
poly(pressure, 4)2   -186.66      50.00  -3.733  0.00223 **
poly(pressure, 4)3    129.90      50.00   2.598  0.02105 *
poly(pressure, 4)4   -101.78      50.00  -2.036  0.06117 .
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 50 on 14 degrees of freedom
Multiple R-squared:  0.8465,    Adjusted R-squared:  0.8027
F-statistic:  19.3 on 4 and 14 DF,  p-value: 1.39e-05
```

Vemos que el resultado de Adjusted R-squared es elevado (~0,80), recordemos que este valor es el porcentaje de varianza de la variable independiente que el modelo es capaz de detectar.

Como en la publicación anterior podemos optar por hacer nosotros los gráficos para el diagnóstico de modelo, o bien mediante el package ggfortify ((Tang, Horikoshi, y Li 2016)), usar la función *autoplot()*

```
#
#    lanzamos mediante la función autoplot, los gráficos del diagnóstico del modelo
# - gráfico de residuos vs. valores ajustados
# - gráfico para evaluar la normalidad de los residuos
# - escala ubicación para ver la homogeneidad de la varianza
# - residuos vs. leverage (distancia de cook) para identificar los puntos influyentes
autoplot(modelo_rlp)
```



Splines

En casos como este podemos buscar un mejor ajuste mediante la función *bs()* del package *splines* ((2024b)), que ajusta un spline sobre los datos disponibles. Los splines son funciones de interpolación que se usan para ajustar curvas a conjuntos de datos. A diferencia de la regresión polinómica, que utiliza un único polinomio para todo

el rango de datos, los splines usan múltiples polinomios conectados en puntos llamados “nudos” (en inglés knot).

Entre las ventajas de usar splines :

- flexibilidad: los splines permiten capturar cambios en la pendiente sin sobreajustar los datos.
- Control sobre la suavidad: puedes ajustar la suavidad de la curva al cambiar el número y la posición de los nudos

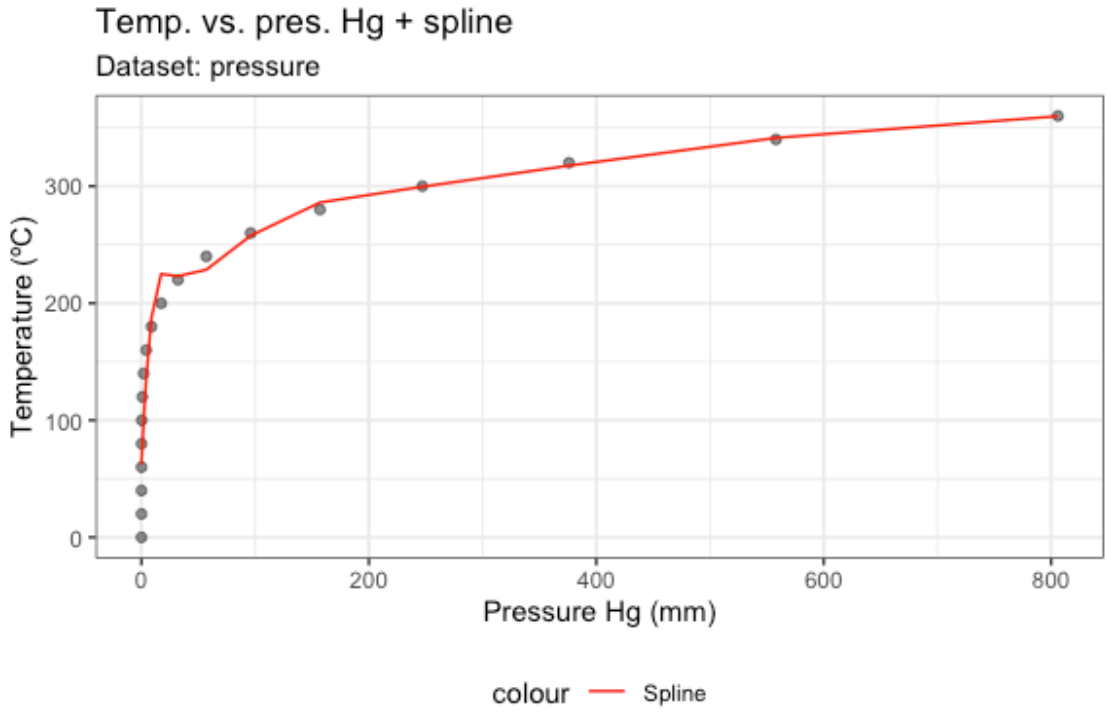
Teniendo en cuenta el gráfico Figura 1 , vamos a ajustar un modelo usando splines cúbicos.

```
#
#     ajustammos el modelo sobre la variable independiente y estimamos 3 nudos donde hacer
el
# "cambio" de spline.
modelo_spline <- lm(temperature ~ bs(pressure, knots=c(25, 100, 200)), data = datos_rp)
# a continuación vemos un resumen de los datos del modelo
summary(modelo_spline)


Call:
lm(formula = temperature ~ bs(pressure, knots = c(25, 100, 200)),
    data = datos_rp)

Residuals:
    Min      1Q  Median      3Q     Max
-62.224  -6.849   0.205  13.646  42.447

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                          62.22      11.93   5.217 0.000216 ***
bs(pressure, knots = c(25, 100, 200))1   175.84      38.96   4.513 0.000710 ***
bs(pressure, knots = c(25, 100, 200))2   141.62      49.02   2.889 0.013595 *
bs(pressure, knots = c(25, 100, 200))3   219.33      46.57   4.709 0.000506 ***
bs(pressure, knots = c(25, 100, 200))4   251.78      93.81   2.684 0.019895 *
bs(pressure, knots = c(25, 100, 200))5   292.72     100.70   2.907 0.013156 *
bs(pressure, knots = c(25, 100, 200))6   297.57      34.29   8.678 1.62e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 32.19 on 12 degrees of freedom
Multiple R-squared:  0.9455,    Adjusted R-squared:  0.9182
F-statistic: 34.68 on 6 and 12 DF,  p-value: 6.688e-07
```

Vemos que en este caso hemos pasado de un Adjusted R-squared de 0,80 a 0,91, el modelo con spline captura mejor la varianza de *$temperature*.

Gráficamente obtenemos el modelo spline añadiendo al gráfico de los datos originales en *pressure*, la capa *geom_line()*

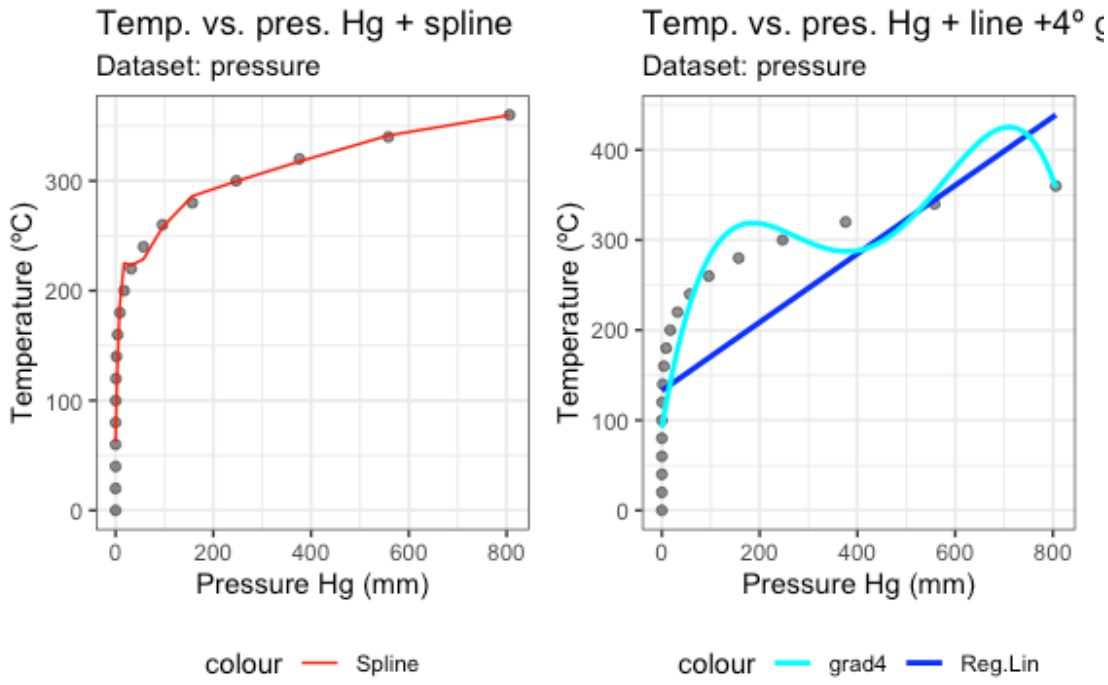
```
#
#          la función predict nos da los valores calculados en el modelo y para simplificar la
#presentación con ggplot, los añadimos como columna al dataframe original.
datos_rp$spline_predict <- predict(modelo_spline)
#
p1 <- datos_rp %>%
    ggplot(aes(x=pressure, y=temperature))+
    geom_point(alpha=0.5)+
    geom_line(aes(y=spline_predict, color="Spline"), linewidth=0.5)+
    scale_color_manual(values = c("Spline"="red"))+
    labs (title = "Temp. vs. pres. Hg + spline",
          subtitle = "Dataset: pressure")+
    xlab("Pressure Hg (mm)")+
    ylab("Temperature (ºC)")

# presentamos el gráfico en pantalla
p1
```



y ambos modelos :

```
#
#    hacemos un gráfico con los datos originales y los polinomio grado1 y grado4
p2 <- datos_rp%>%
    ggplot(aes(x=pressure, y=temperature))+
    geom_point(alpha=0.5)+
    geom_smooth(method = "lm", se=FALSE, aes(color="Reg.Lin"))+
    geom_smooth(method="lm", formula = "y~poly(x,4)", se=FALSE, aes(color="grad4"))+
    scale_color_manual(values=c("Reg.Lin"="blue", "grad4"="cyan"))+
    labs (title = "Temp. vs. pres. Hg + line +4º grad",
```

```
        subtitle = "Dataset: pressure")+
    xlab("Pressure Hg (mm)")+
    ylab("Temperature (ºC)")
# componemos mediante el package patchwork ambos gráficos uno al lado del otro.
p1+p2
```



Hemos visto en esta publicación de forma muy resumida como ajustar un modelo de regresión polinómico o mediante splines a un conjunto de datos dados. La elección final de la solución a nuestro problema es necesario estudiarla caso por caso, y decidir en consecuencia.

Referencias

R Core Team. 2024a. «R: A Language and Environment for Statistical Computing». https://www.R-project.org/.

———. 2024b. «R: A Language and Environment for Statistical Computing». https://www.R-project.org/.

Tang, Yuan, Masaaki Horikoshi, y Wenxuan Li. 2016. «ggfortify: Unified Interface to Visualize Statistical Result of Popular R Packages» 8. https://doi.org/10.32614/RJ-2016-060.