Regresión lineal simple vs Polinomial

Rubén Pizarro Gurrola

2025-01-17

Objetivo

Construir y evaluar modelos de prediccion de regresión lieneal simple y polinomial entre la variable independiente estatura y el peso como variable dependiente de una persona.

Descripción

Regresión Lineal Simple: Relación lineal directa entre estatura y peso.

Regresión Polinómica: Modela relaciones no lineales.

La idea es construir modelos con un conjunto de entrenamiento, realizar predicciones sobre un conjunto de validación y evaluar el rendimiento utilizando el Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- n: Número total de observaciones.
- y_i : Valor observado (real).
- $\hat{y}_i$: Valor predicho o predicciones del modelo.
- RMSE: Raíz cuadrada del promedio de los errores al cuadrado.

Cargar librerías

```
library(readr)
library(ggplot2)
library(dplyr)


# library(mosaic)
library(ggplot2)  # Para gráficos
# library(cowplot) #Imágenes en el mismo renglón
# library("visualize")
```

Cargar funciones

Se cargan funciones preparadas pero antes se deben instalar estos paquetes en caso de que no se hayan instalado en el entorno R Studio:

- “dplyr”: para procesar datos extraer, filtrar, . . .
- “ggplot2”: para gráficos

```r
# install.packages("mosaic")
source("https://raw.githubusercontent.com/rpizarrog/Ciencia-de-los-Datos-Descriptivo-Predictivo/refs/hea
```

Cargar datos

Datos de entrenamiento (train)

```r
datos_entrenamiento <- read.csv("https://raw.githubusercontent.com/rpizarrog/Ciencia-de-los-Datos-Descri
```

Datos de validación (test)

```r
datos_validacion <- read.csv("https://raw.githubusercontent.com/rpizarrog/Ciencia-de-los-Datos-Descripti
```

Análisis descriptivo

```r
print ("Datos de entrenamiento")
```

```
## [1] "Datos de entrenamiento"
```

```r
summary(datos_entrenamiento)
```

```
##     estatura          peso
##  Min.   :150.3   Min.   : 49.83
##  1st Qu.:161.4   1st Qu.: 79.96
##  Median :174.8   Median : 87.88
##  Mean   :174.1   Mean   : 87.70
##  3rd Qu.:187.8   3rd Qu.: 96.09
##  Max.   :199.3   Max.   :118.43
```

```r
str(datos_entrenamiento)
```

```
## 'data.frame':    160 obs. of  2 variables:
##  $ estatura: num  156 195 184 173 166 ...
##  $ peso    : num  80.1 95.3 87.6 86.5 73.1 ...
```

```r
print ("Datos de validación")
```

```
## [1] "Datos de validación"
```

```r
summary(datos_validacion)
```

```
##     estatura          peso
##  Min.   :150.3   Min.   : 62.52
##  1st Qu.:161.8   1st Qu.: 80.03
##  Median :173.1   Median : 85.94
##  Mean   :174.6   Mean   : 88.09
##  3rd Qu.:187.0   3rd Qu.: 98.75
##  Max.   :199.3   Max.   :110.36
```

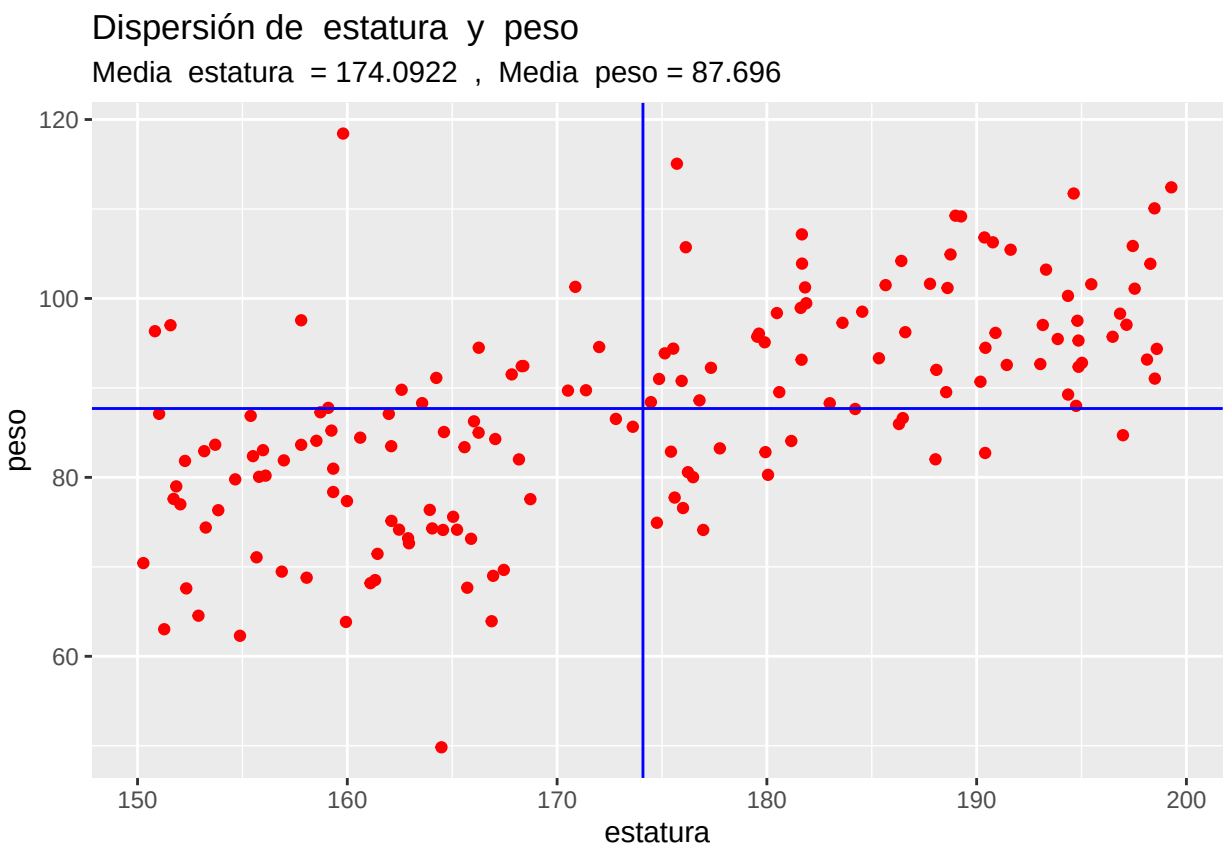
```r
str(datos_validacion)
```

```
## 'data.frame':    40 obs. of  2 variables:
##  $ estatura: num  175 159 180 162 150 ...
##  $ peso    : num  80.8 98.6 79.5 78.8 76.3 ...
```

Dispersión de los datos

Se muestra la dispersión de los datos de entrenamiento con ggplot

```
f_diag.dispersion(datos_entrenamiento)
```



Construir modelos

- *estatura* es la variable independiente
- *peso* es la variable dependiente

Modelo regresión lineal simple

$$Y = a + b \cdot x$$

$$Y = \beta_0 + \beta_1 \cdot x$$

```
modelo_rls <- lm(data = datos_entrenamiento, formula = peso~estatura)
summary(modelo_rls)
```

```
##
## Call:
## lm(formula = peso ~ estatura, data = datos_entrenamiento)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.861  -7.320   0.649   5.956  38.177
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
```

```
## (Intercept) -2.99573    8.96807  -0.334                0.739
## estatura     0.52094    0.05133  10.149 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.545 on 158 degrees of freedom
## Multiple R-squared:  0.3946, Adjusted R-squared:  0.3908
## F-statistic:   103 on 1 and 158 DF,  p-value: < 0.00000000000000022
```

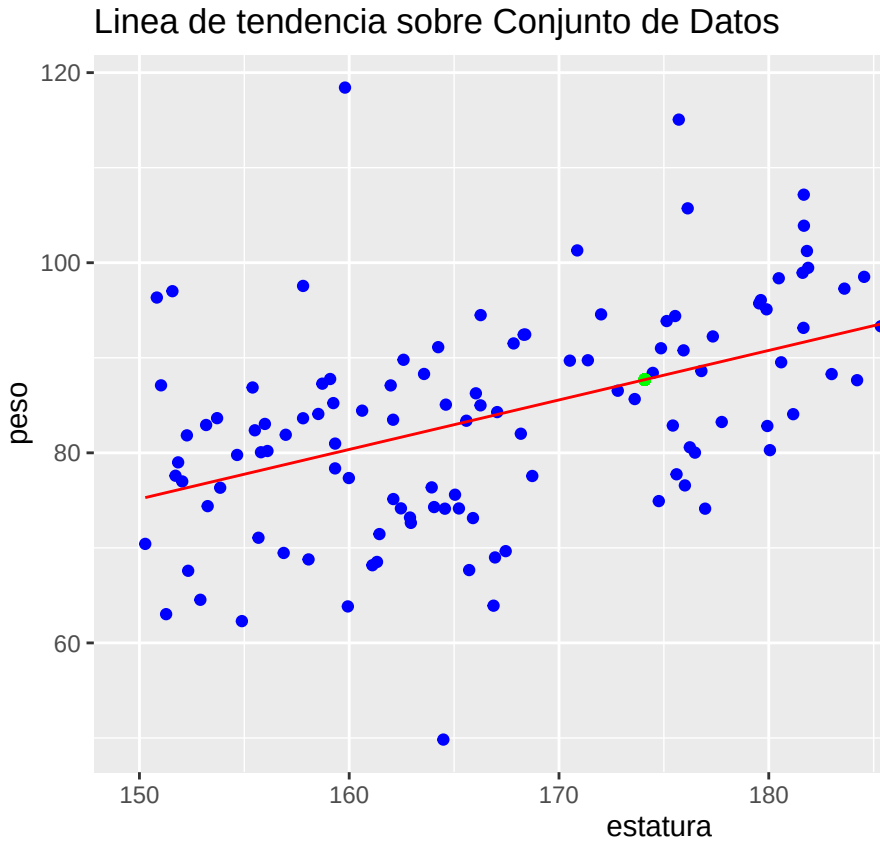
```
a <- modelo_rls$coefficients[1]
b <- modelo_rls$coefficients[2]
paste ("Valor de coeficiente a: ", a)
```

```
## [1] "Valor de coeficiente a:  -2.99573486847959"
```

```
paste ("Valor de coeficiente b: ", b)
```

```
## [1] "Valor de coeficiente b:  0.520940716085568"
```

```
f_linea_tendencia_reg_lineal(datos_entrenamiento, modelo_rls)
```



Linea de tendencia regresión lineal simple

```
predicciones <- predict(object = modelo_rls, datos_validacion)
predicciones
```

Hacer predicciones con regresión lineal simple y datos de validación

```
##         1         2         3         4         5         6         7         8
```

```
##  88.00728  79.92252  90.97012  81.33513  75.32645  97.84434 100.85085  92.79392
##         9        10        11        12        13        14        15        16
##  91.95021  92.40211  78.81603  99.23090  96.90253  84.48234  96.43159  85.21868
##        17        18        19        20        21        22        23        24
##  77.45034  78.91945  83.76429  77.08720  81.08241  83.06998  76.48624  89.76498
##        25        26        27        28        29        30        31        32
##  89.28100  85.26926  83.24554  96.04023  86.27026  78.26751  93.58856  86.39626
##        33        34        35        36        37        38        39        40
##  99.15746  94.13385  98.80301  98.78479  83.56385  93.45693  77.49715  95.23468
```

Evaluar las predicciones Vs reales Con las predicciones generadas, se pueden comparar contra los datos reales del conjunto de validación para determinar el *RMSE* y compararlo con otro modelo.

```
datos_comparar <- data.frame(reales = datos_validacion$peso, predicciones_rls = predicciones)
datos_comparar
```

```
##       reales predicciones_rls
## 1   80.81160         88.00728
## 2   98.55304         79.92252
## 3   79.47970         90.97012
## 4   78.76413         81.33513
## 5   76.30898         75.32645
## 6   82.48510         97.84434
## 7   80.99952        100.85085
## 8   83.75690         92.79392
## 9  102.71043         91.95021
## 10  93.10031         92.40211
## 11  73.66947         78.81603
## 12 107.86854         99.23090
## 13  92.77989         96.90253
## 14  76.65798         84.48234
## 15  99.55987         96.43159
## 16  90.95184         85.21868
## 17  82.36279         77.45034
## 18  78.41336         78.91945
## 19 104.71189         83.76429
## 20 100.01035         77.08720
## 21  82.68898         81.08241
## 22  80.15217         83.06998
## 23  67.99702         76.48624
## 24  85.68692         89.76498
## 25  86.19922         89.28100
## 26  94.25695         85.26926
## 27  62.52313         83.24554
## 28  95.87367         96.04023
## 29  79.67553         86.27026
## 30  99.32697         78.26751
## 31 100.57266         93.58856
## 32  76.90348         86.39626
## 33 103.18472         99.15746
## 34  82.44773         94.13385
## 35 105.17958         98.80301
## 36 110.35827         98.78479
## 37  88.96325         83.56385
## 38  86.34848         93.45693
```

```
## 39  84.07677          77.49715
## 40  87.12872          95.23468
```

```r
# Cálculo manual del RMSE
rmse <- sqrt(mean((datos_comparar$reales - datos_comparar$predicciones_rls)^2))

# Imprimir resultado
paste ("RMSE:", rmse)
```

Determinar RMSE

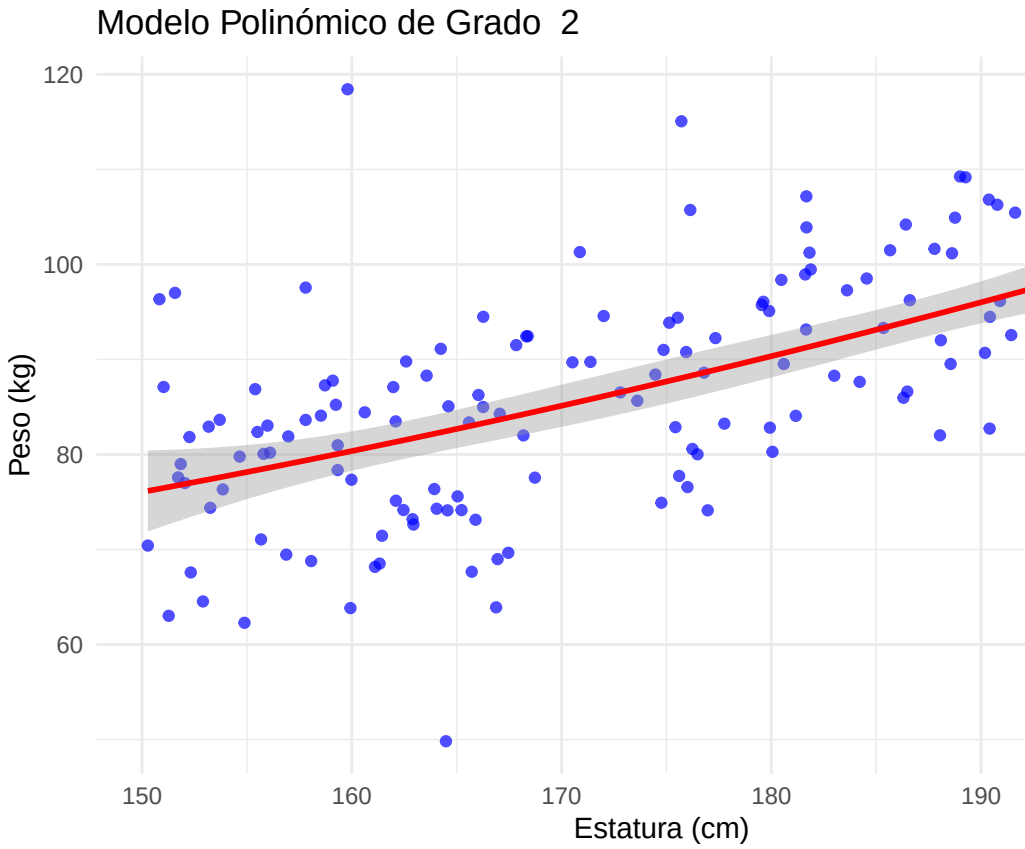
```
## [1] "RMSE: 10.4145276539336"
```

Significa que las predicciones se alejan aproximadamente *10.41* del valor real, este valor hay que compararlo contra otro modelo entre mas cercano a cero el modelo es mejor o predice de mejor manera.

Modelo polinomial segundo nivel

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 ... \beta_n X^n$$

```r
nivel <- 2
f_polinomial_curva(datos_entrenamiento, nivel)
```



Visualizar curva de tendencia

```r
# Ajustar modelo polinómico de grado 2
modelo_polinomico_2 <- lm(peso ~ poly(estatura, nivel, raw = TRUE), data = datos_entrenamiento)
```

```r
# Resumen del modelo
summary(modelo_polinomico_2)
```

```
##
## Call:
## lm(formula = peso ~ poly(estatura, nivel, raw = TRUE), data = datos_entrenamiento)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.614  -7.172   0.558   5.991  38.155
##
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        65.712726 126.360350   0.520    0.604
## poly(estatura, nivel, raw = TRUE)1 -0.270605   1.452940  -0.186    0.852
## poly(estatura, nivel, raw = TRUE)2  0.002264   0.004152   0.545    0.586
##
## Residual standard error: 9.567 on 157 degrees of freedom
## Multiple R-squared:  0.3958, Adjusted R-squared:  0.3881
## F-statistic: 51.42 on 2 and 157 DF,  p-value: < 0.00000000000000022
```

```r
predicciones <- predict(object = modelo_polinomico_2, datos_validacion)
predicciones
```

Hacer predicciones con regresión polinómico y datos de validación

```
##         1         2         3         4         5         6         7         8
##  87.51684  79.98817  90.54889  81.22500  76.19438  98.14775 101.71898  92.48812
##         9        10        11        12        13        14        15        16
##  91.58412  92.06684  79.04262  99.77602  97.06006  84.10029  96.52172  84.79686
##        17        18        19        20        21        22        23        24
##  77.90373  79.13014  83.42973  77.60613  81.00129  82.78952  77.11847  89.29793
##        25        26        27        28        29        30        31        32
##  88.80236  84.84504  82.95064  96.07717  85.80733  78.58146  93.35041  85.92964
##        33        34        35        36        37        38        39        40
##  99.68897  93.94821  99.27011  99.24864  83.24409  93.20685  77.94226  95.17017
```

Evaluar las predicciones Vs reales Con las predicciones generadas, se pueden comparar contra los datos reales del conjunto de validación para determinar el *RMSE* y compararlo con otro modelo.

```r
datos_comparar$predicciones_poly2 <-predicciones
datos_comparar
```

```
##       reales predicciones_rls predicciones_poly2
## 1   80.81160         88.00728           87.51684
## 2   98.55304         79.92252           79.98817
## 3   79.47970         90.97012           90.54889
## 4   78.76413         81.33513           81.22500
## 5   76.30898         75.32645           76.19438
## 6   82.48510         97.84434           98.14775
## 7   80.99952        100.85085          101.71898
## 8   83.75690         92.79392           92.48812
## 9  102.71043         91.95021           91.58412
## 10  93.10031         92.40211           92.06684
```

```
## 11  73.66947         78.81603          79.04262
## 12 107.86854         99.23090          99.77602
## 13  92.77989         96.90253          97.06006
## 14  76.65798         84.48234          84.10029
## 15  99.55987         96.43159          96.52172
## 16  90.95184         85.21868          84.79686
## 17  82.36279         77.45034          77.90373
## 18  78.41336         78.91945          79.13014
## 19 104.71189         83.76429          83.42973
## 20 100.01035         77.08720          77.60613
## 21  82.68898         81.08241          81.00129
## 22  80.15217         83.06998          82.78952
## 23  67.99702         76.48624          77.11847
## 24  85.68692         89.76498          89.29793
## 25  86.19922         89.28100          88.80236
## 26  94.25695         85.26926          84.84504
## 27  62.52313         83.24554          82.95064
## 28  95.87367         96.04023          96.07717
## 29  79.67553         86.27026          85.80733
## 30  99.32697         78.26751          78.58146
## 31 100.57266         93.58856          93.35041
## 32  76.90348         86.39626          85.92964
## 33 103.18472         99.15746          99.68897
## 34  82.44773         94.13385          93.94821
## 35 105.17958         98.80301          99.27011
## 36 110.35827         98.78479          99.24864
## 37  88.96325         83.56385          83.24409
## 38  86.34848         93.45693          93.20685
## 39  84.07677         77.49715          77.94226
## 40  87.12872         95.23468          95.17017
```

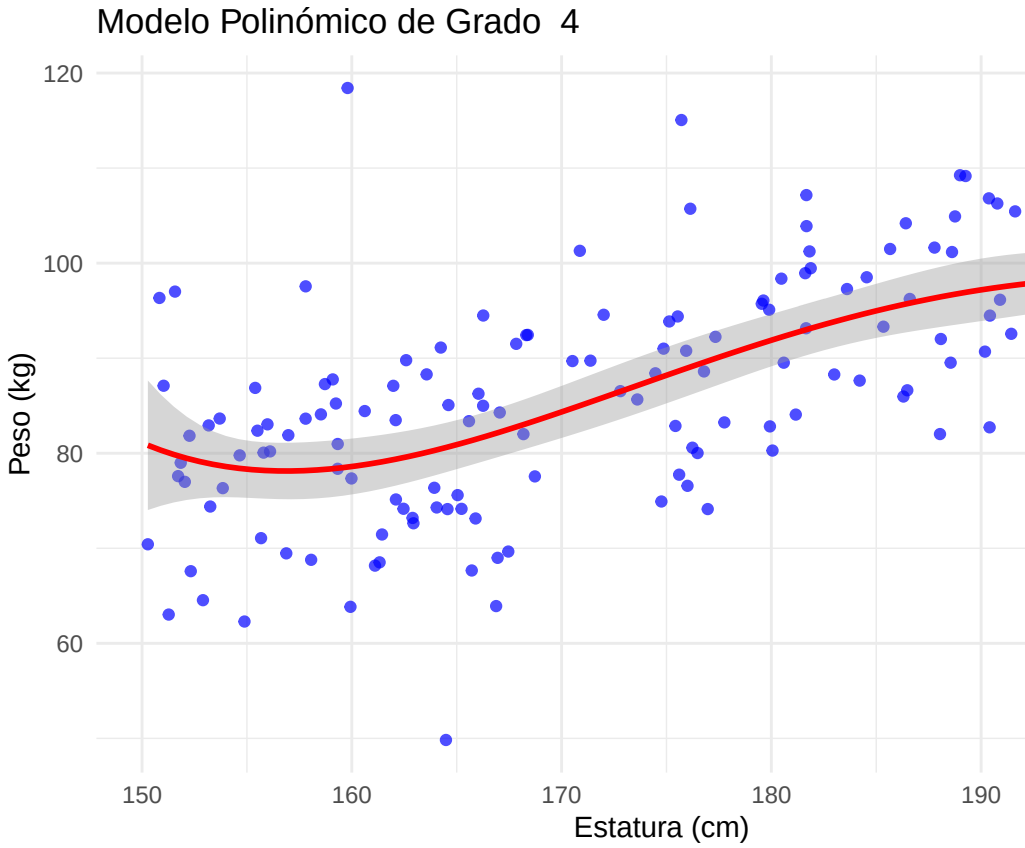
```r
# Cálculo manual del RMSE
rmse <- sqrt(mean((datos_comparar$reales - datos_comparar$predicciones_poly2)^2))

# Imprimir resultado
paste ("RMSE:", rmse)
```

Determinar RMSE

```
## [1] "RMSE: 10.3556596265271"
```

Modelo polinomial cuarto nivel

```r
nivel <- 4
f_polinomial_curva(datos_entrenamiento, nivel)
```



Visualizar curva de tendencia

```
# Ajustar modelo polinómico de grado 4
modelo_polinomico_4 <- lm(peso ~ poly(estatura, nivel, raw = TRUE), data = datos_entrenamiento)

# Resumen del modelo
summary(modelo_polinomico_4)
```

```
##
## Call:
## lm(formula = peso ~ poly(estatura, nivel, raw = TRUE), data = datos_entrenamiento)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.769  -6.644   0.093   5.787  39.882
##
## Coefficients:
##                                       Estimate     Std. Error t value
## (Intercept)                      12755.07976621 24557.89579943   0.519
## poly(estatura, nivel, raw = TRUE)1  -272.42770496   567.04711610  -0.480
## poly(estatura, nivel, raw = TRUE)2     2.17170279     4.89709441   0.443
## poly(estatura, nivel, raw = TRUE)3    -0.00761185     0.01874721  -0.406
## poly(estatura, nivel, raw = TRUE)4     0.00000991     0.00002684   0.369
##                                     Pr(>|t|)
## (Intercept)                            0.604
## poly(estatura, nivel, raw = TRUE)1     0.632
## poly(estatura, nivel, raw = TRUE)2     0.658
## poly(estatura, nivel, raw = TRUE)3     0.685
## poly(estatura, nivel, raw = TRUE)4     0.713
```

```
##
## Residual standard error: 9.486 on 155 degrees of freedom
## Multiple R-squared:  0.4135, Adjusted R-squared:  0.3984
## F-statistic: 27.32 on 4 and 155 DF,  p-value: < 0.00000000000000022
```

Hacer predicciones con regresión polinómico y datos de validación

```
predicciones <- predict(object = modelo_polinomico_4, datos_validacion)
predicciones
```

```
##        1        2        3        4        5        6        7        8
## 87.96135 78.38961 92.14898 79.28068 80.77518 98.14685 98.59845 94.35672
##        9       10       11       12       13       14       15       16
## 93.38117 93.91418 78.12981 98.51748 97.72883 82.82597 97.46879 83.84975
##       17       18       19       20       21       22       23       24
## 78.46456 78.13545 81.88036 78.69338 79.08117 81.03019 79.21530 90.51245
##       25       26       27       28       29       30       31       32
## 89.82522 83.92169 81.23839 97.22689 85.37608 78.16989 95.19485 85.56228
##       33       34       35       36       37       38       39       40
## 98.50502 95.72130 98.43372 98.42956 81.62780 95.06172 78.43960 96.65605
```

Evaluar las predicciones Vs reales

Con las predicciones generadas, se pueden comparar contra los datos reales del conjunto de validación para determinar el *RMSE* y compararlo con otro modelo.

```
datos_comparar$predicciones_poly4 <- predicciones
datos_comparar
```

```
##       reales predicciones_rls predicciones_poly2 predicciones_poly4
## 1   80.81160         88.00728           87.51684           87.96135
## 2   98.55304         79.92252           79.98817           78.38961
## 3   79.47970         90.97012           90.54889           92.14898
## 4   78.76413         81.33513           81.22500           79.28068
## 5   76.30898         75.32645           76.19438           80.77518
## 6   82.48510         97.84434           98.14775           98.14685
## 7   80.99952        100.85085          101.71898           98.59845
## 8   83.75690         92.79392           92.48812           94.35672
## 9  102.71043         91.95021           91.58412           93.38117
## 10  93.10031         92.40211           92.06684           93.91418
## 11  73.66947         78.81603           79.04262           78.12981
## 12 107.86854         99.23090           99.77602           98.51748
## 13  92.77989         96.90253           97.06006           97.72883
## 14  76.65798         84.48234           84.10029           82.82597
## 15  99.55987         96.43159           96.52172           97.46879
## 16  90.95184         85.21868           84.79686           83.84975
## 17  82.36279         77.45034           77.90373           78.46456
## 18  78.41336         78.91945           79.13014           78.13545
## 19 104.71189         83.76429           83.42973           81.88036
## 20 100.01035         77.08720           77.60613           78.69338
## 21  82.68898         81.08241           81.00129           79.08117
## 22  80.15217         83.06998           82.78952           81.03019
## 23  67.99702         76.48624           77.11847           79.21530
## 24  85.68692         89.76498           89.29793           90.51245
## 25  86.19922         89.28100           88.80236           89.82522
```

```
## 26  94.25695         85.26926          84.84504          83.92169
## 27  62.52313         83.24554          82.95064          81.23839
## 28  95.87367         96.04023          96.07717          97.22689
## 29  79.67553         86.27026          85.80733          85.37608
## 30  99.32697         78.26751          78.58146          78.16989
## 31 100.57266         93.58856          93.35041          95.19485
## 32  76.90348         86.39626          85.92964          85.56228
## 33 103.18472         99.15746          99.68897          98.50502
## 34  82.44773         94.13385          93.94821          95.72130
## 35 105.17958         98.80301          99.27011          98.43372
## 36 110.35827         98.78479          99.24864          98.42956
## 37  88.96325         83.56385          83.24409          81.62780
## 38  86.34848         93.45693          93.20685          95.06172
## 39  84.07677         77.49715          77.94226          78.43960
## 40  87.12872         95.23468          95.17017          96.65605
```

Determinar RMSE

```
# Cálculo manual del RMSE
rmse <- sqrt(mean((datos_comparar$reales - datos_comparar$predicciones_poly4)^2))

# Imprimir resultado
paste ("RMSE:", rmse)
```

```
## [1] "RMSE: 10.5667927983238"
```

Interpretación

Se construyeron modelos de regresión lineal simple y polnómico de segundo y cuarto nivel para hacer predicciones del peso de una persona en función de la estatura.

El modelo de regresión lineal simple tiene un valor aproximado de RMSE *10.41* que significa que las prediciones se desvían en promedio ese valor con respecto a los valores reales.

El modelo de polinomial de segundo nivel tiene un valor aproximado de RMSE *10.35* que sifgnifica que las prediciones se desvían en promedio ese valor con respecto a los valores reales.

El modelo de polinomial de cuarto nivel tiene un valor aproximado de RMSE *10.56* que significa que las prediciones se desvían en promedio ese valor con respecto a los valores reales.

Al final en términos de quién es mejor haciendo predicciones para estos datos es el modelo polinomial de segundo nivel.

Los valores *R Square* y *Multiple R Square* para los tres modelos andan aproximadamente en *39%* y *40%* siendo el de mejor rendimiento el modelo de **cuarto nivel**.

- Lineal Simple: Multiple R-squared: 0.3946, Adjusted R-squared: 0.3908
- Polinomial segundo nivel: Multiple R-squared: 0.3958, Adjusted R-squared: 0.3881
- Polinomial cuarto nivel: Multiple R-squared: 0.4135, Adjusted R-squared: 0.3984

¿que rendimiento tiene el modelo polinómico de otro nivel?