Regresion Lineal Simple

Victor Lopez

2023-02-11

Error medio cuadrado

Se busca la recta $y = ax + b$ que mejor aproxime los puntos dados imponiendo que la suma de los cuadrados de las diferencias entre los valores y_i y sus aproximaciones $\tilde{y}_i = ax_i + b$ sea mínima. Es decir, que

$$MDE = \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{n}$$

sea mínima

Al analizar datos, siempre es recomendable empezar con una representación gráfica que nos permita hacernos a la idea de lo que tenemos.

Calcular la recta de regresion lineal

Se utiliza: `lm(y~x)`

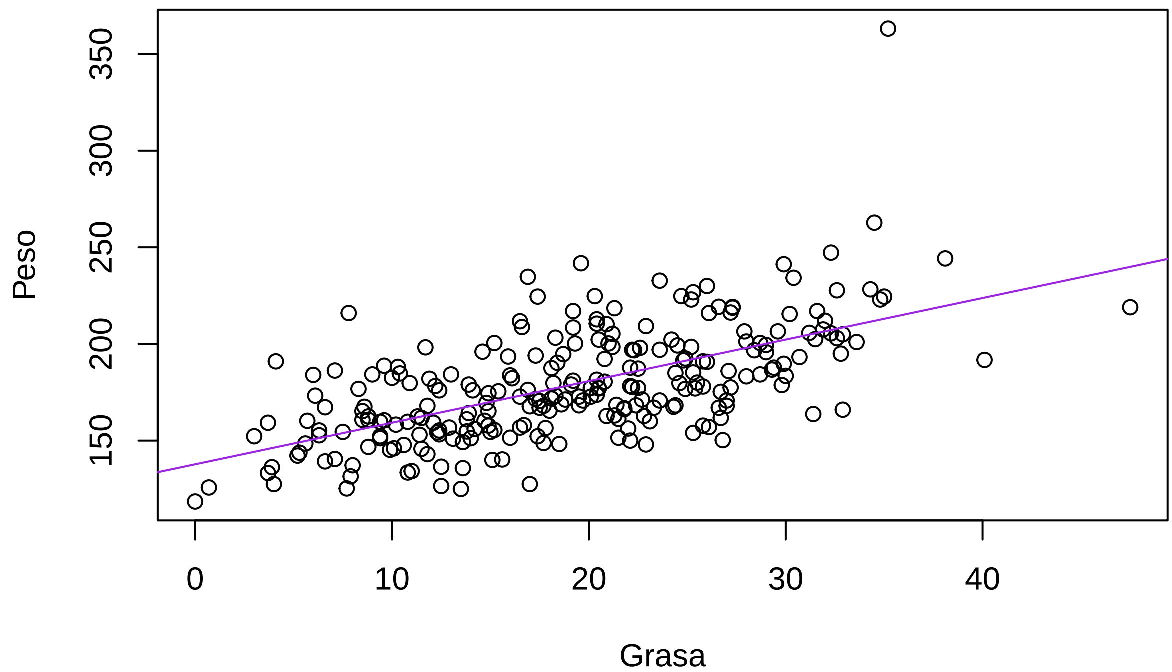
```r
body = read.table("../../data/bodyfat.txt", header = T)

body2 = body[, c(2, 4)]
names(body2) = c("Grasa", "Peso")
lm(Peso ~ Grasa, data = body2)
```

```
##
## Call:
## lm(formula = Peso ~ Grasa, data = body2)
##
## Coefficients:
## (Intercept)        Grasa
##     137.738        2.151
```

Esto significa que nuestra recta de regresion lineal sera:

$$y = 2.151x + 137.738$$

```r
plot(body2)
abline(lm(Peso ~ Grasa, data = body2), col = "purple")
```



Coeficiente de determinación

El coeficiente de determinación, R^2, nos es útil para evaluar numéricamente si la relación lineal obtenida es significativa o no.

No explicaremos de momento como se define. Eso lo dejamos para curiosidad del usuario. Por el momento, es suficiente con saber que este coeficiente se encuentra en el intervalo $[0, 1]$. Si R^2 es mayor a 0.9, consideraremos que el ajuste es bueno. De lo contrario, no.

La función `summary` aplicada a `lm` nos muestra los contenidos de este objeto. Entre ellos encontramos `Multiple R-squared`, que no es ni más ni menos que el coeficiente de determinación, R^2.

```
summary(lm(Peso ~ Grasa, data = body2))$r.squared
```

```
## [1] 0.3750509
```

Transformaciones de escala

No siempre encontraremos dependencias lineales. A veces nos encontraremos otro tipo de dependencias, como por ejemplo pontencias o exponenciales.

Diremos que un gráfico está en *escala semilogarítmica* cuando su eje de abcisas está en escala lineal y, el de ordenadas, en escala logarítmica.

Diremos que un gráfico está en escala *doble logarítmica* cuando ambos ejes están en escala logarítmica.

Ley aproximadamente exponencial

Si al representar unos puntos $(x_i, y_i)_{i=1,\ldots,n}$ en escala semilogarítmica observamos que siguen aproximadamente una recta, esto querrá decir que los valores $\log(y)$ siguen una ley aproximadamente lineal en los valores x, y, por lo tanto, que y sigue una *ley aproximadamente exponencial* en x.

En efecto, si $\log(y) = ax + b$, entonces,

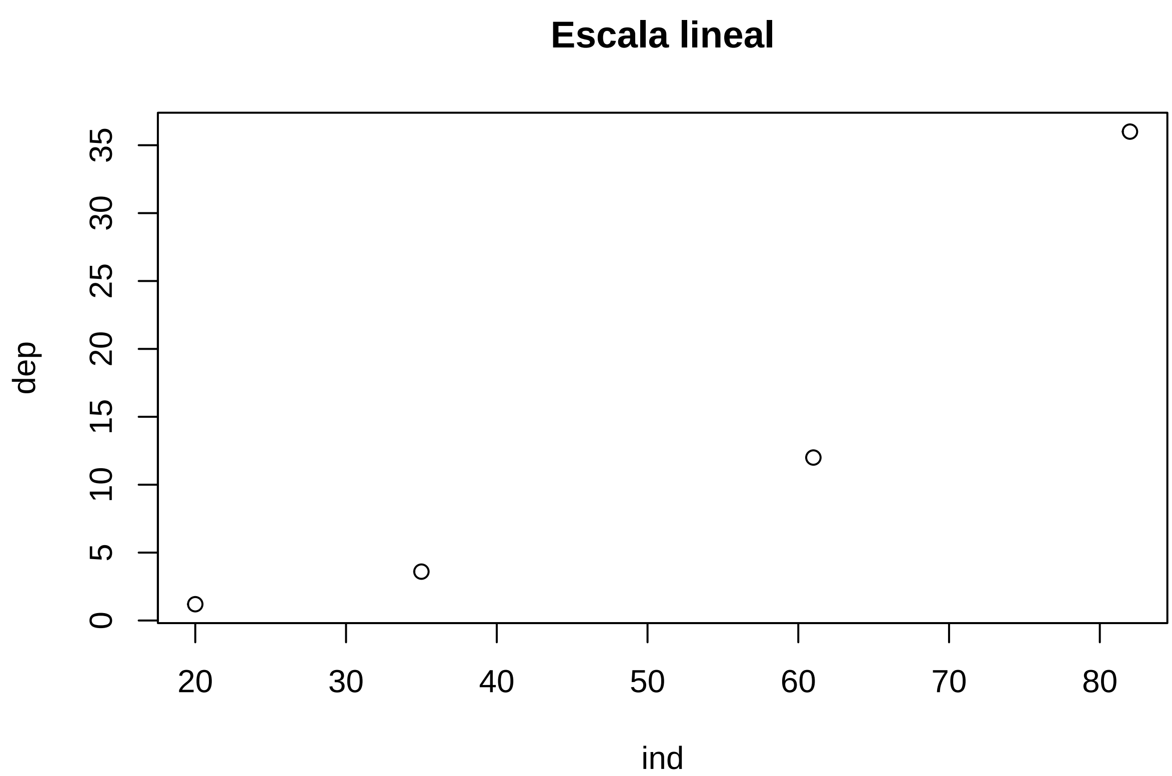
$$y = 10^{\log(y)} = 10^{ax+b} = 10^{ax} \cdot 10^{b} = \alpha^{x}\beta$$

con $\alpha = 10^a$ y $\beta = 10^b$

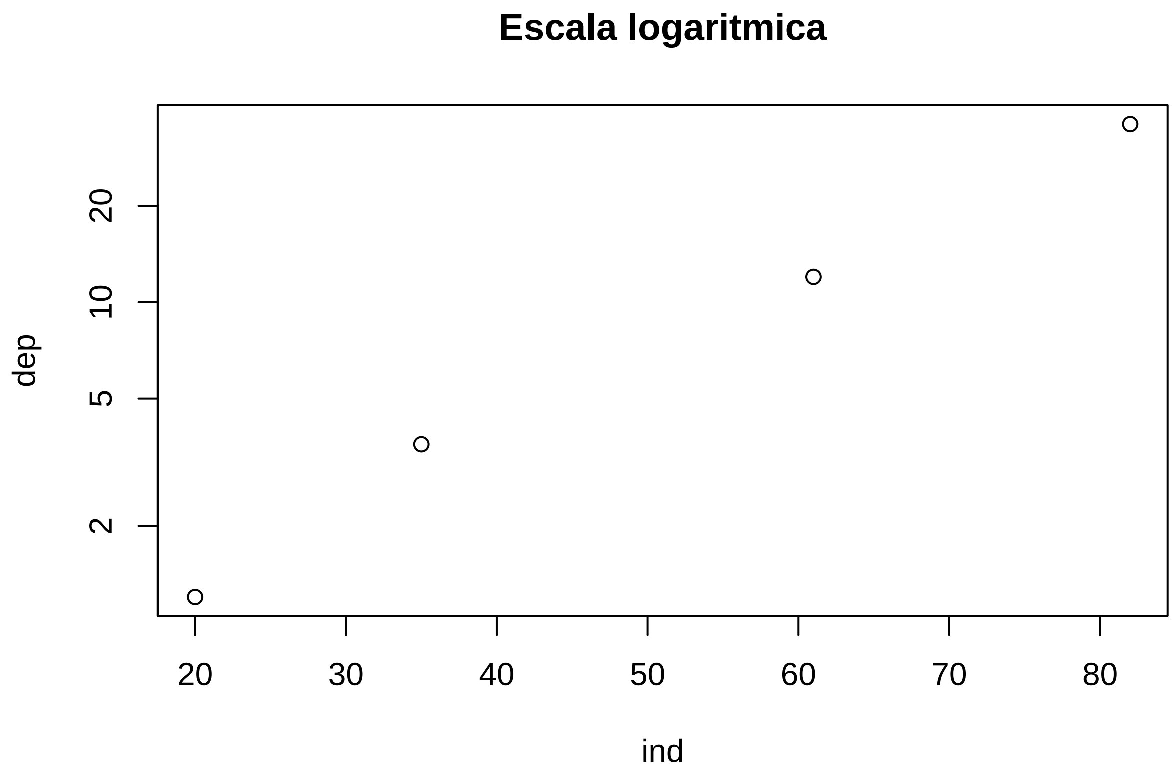
Por lo tanto el valor de "y" despejada es igual a la funcion logaritmica que se ajusta mejor a nuestros datos

Ejercicio:

```
dep = c(1.2, 3.6, 12, 36)
ind = c(20, 35, 61, 82)

plot(ind, dep, main = "Escala lineal")
```




```
plot(ind, dep, log = "y", main = "Escala logaritmica")
```




```
lm(log10(dep) ~ ind)
```

```
##
## Call:
## lm(formula = log10(dep) ~ ind)
##
## Coefficients:
## (Intercept)          ind
##    -0.32951      0.02318
```

```
summary(lm(log10(dep) ~ ind))$r.squared
```

```
## [1] 0.9928168
```

Lo que acabamos de obtener es que

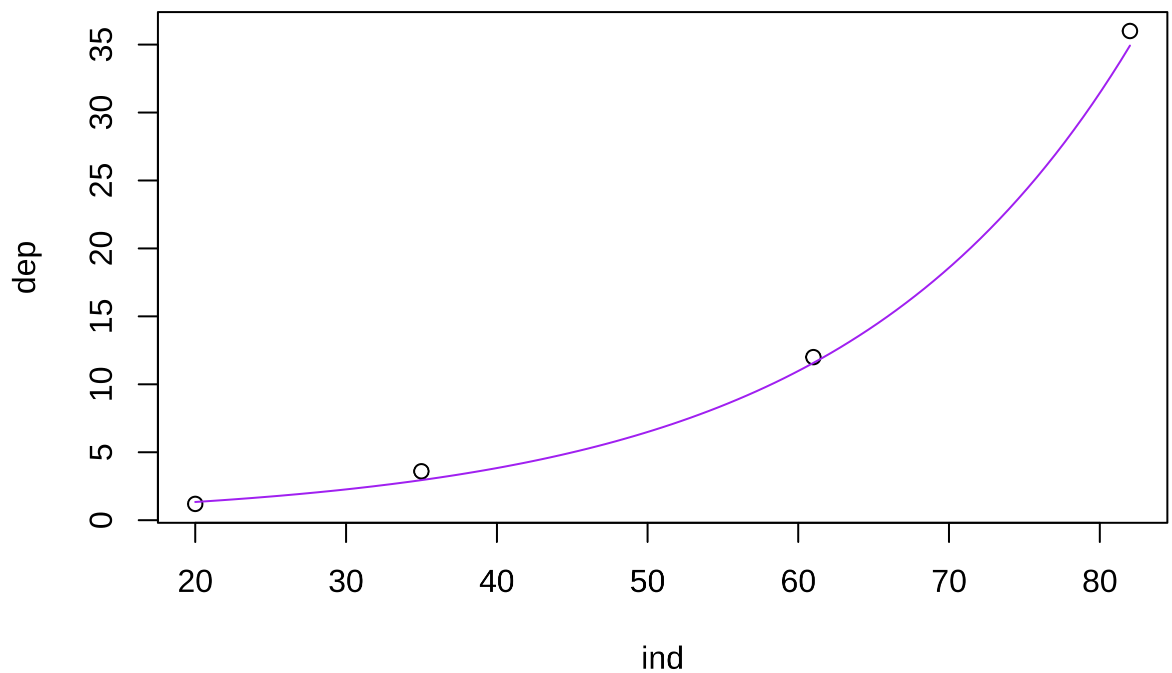
$$\log(dep) = 0.023 \cdot ind - 0.33$$

es una buena aproximación de nuestros datos.

Con lo cual

$$dep = 10^{0.023 \cdot ind} \cdot 10^{-0.33} = 1.054^{ind} \cdot 0.468$$

```
plot(ind, dep)
curve(1.054^x*0.468, add = T, col = "purple")
```



Ley aproximadamente potencial

Si al representar unos puntos $(x_i, y_i)_{i=1,\ldots,n}$ en escala doble logarítmica observamos que siguen aproximadamente una recta, esto querrá decir que los valores $\log(y)$ siguen una ley aproximadamente lineal en los valores $\log(x)$, y, por lo tanto, que y sigue una *ley aproximadamente potencial* en x.

En efecto, si $\log(y) = a\log(x) + b$, entonces, por propiedades de logaritmos

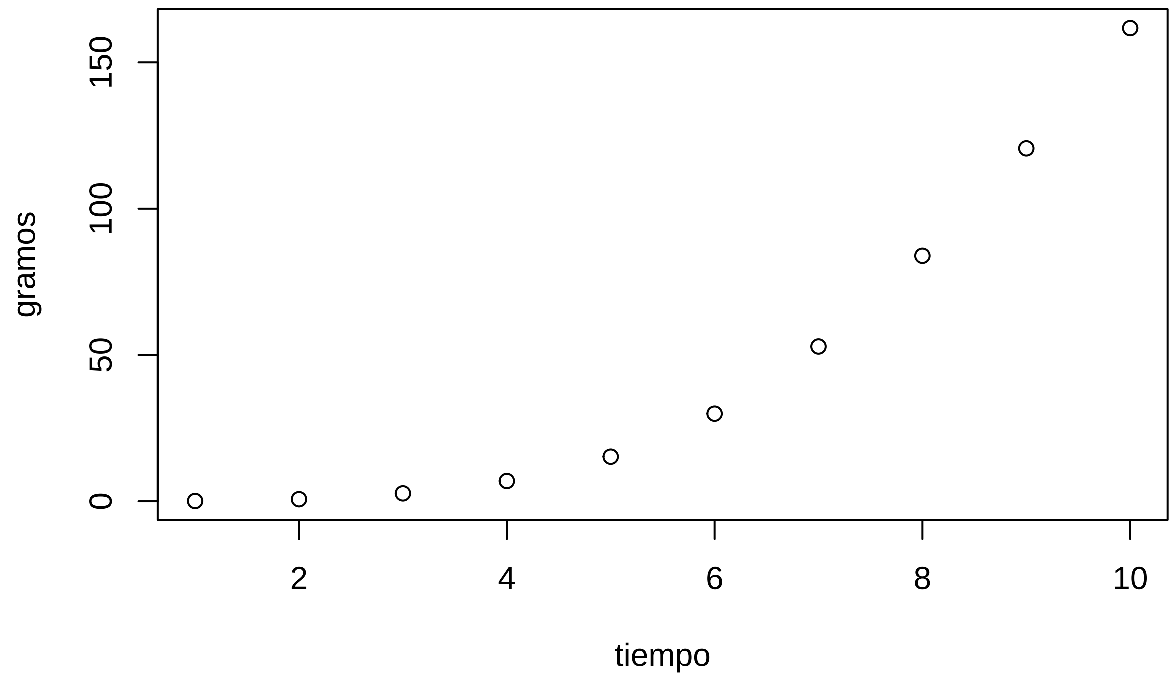
$$y = 10^{\log(y)} = 10^{a\log(x)+b} = (10^{\log(x)})^a \cdot 10^b = x^a\beta$$

con $\beta = 10^b$

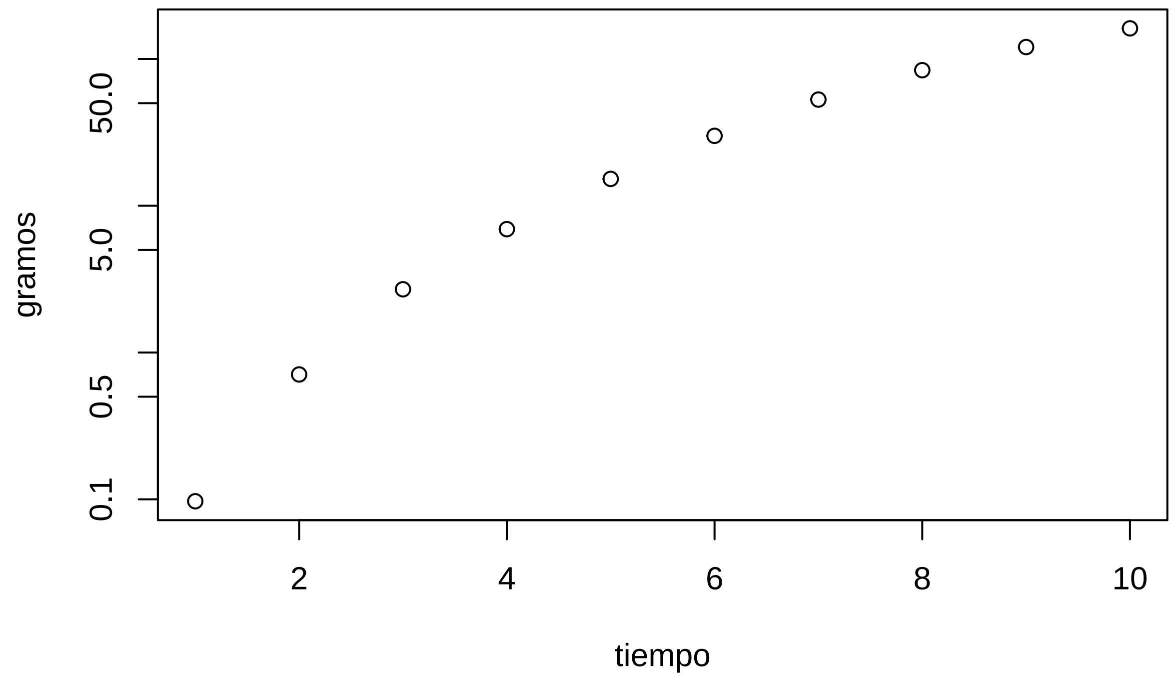
Por lo tanto el valor de "y" despejada es igual a la funcion logaritmica que se ajusta mejor a nuestros datos

Ejercicio:

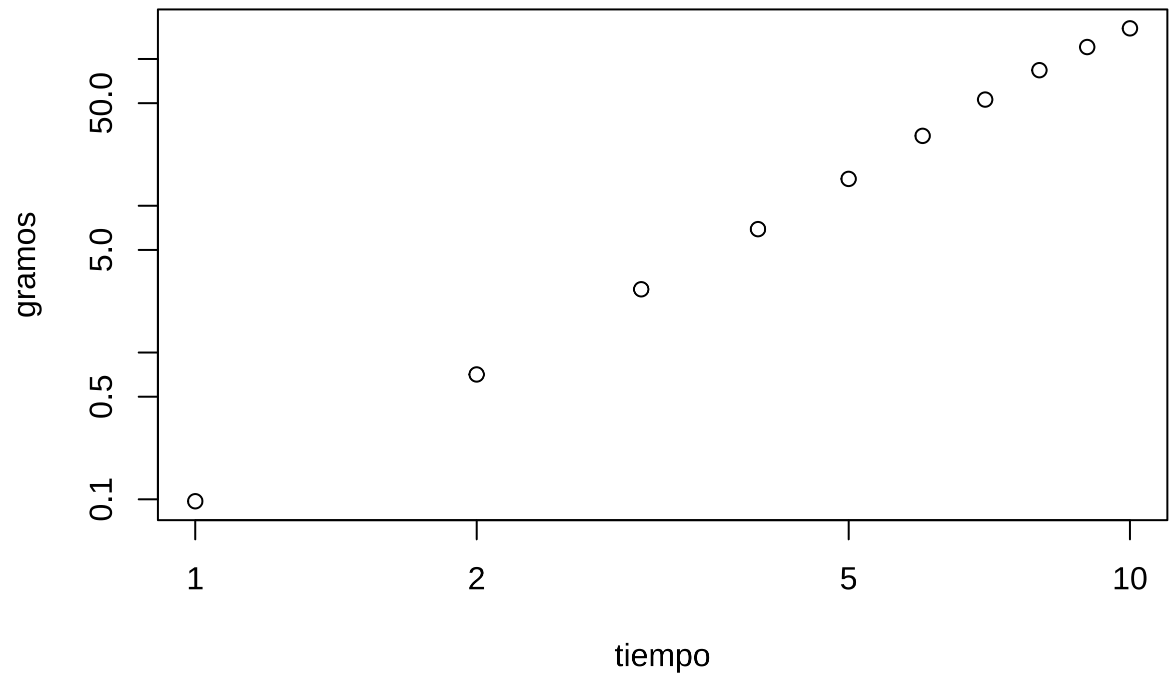
```
tiempo = 1:10
gramos = c(0.097,0.709,2.698,6.928,15.242,29.944,52.902,83.903,120.612,161.711)
d.f = data.frame(tiempo,gramos)

plot(d.f)
```




```
plot(d.f, log = "y")
```




```
plot(d.f, log = "xy")
```




```r
# Si observamos, es conveniente usar la escala logaritmica en ambos ejes cuando
# notamos que al usarla solo en "y", hace una curva pero hacia arriba en vez de abajo

lm(log10(gramos) ~ log10(tiempo), data = d.f)
```

```
##
## Call:
## lm(formula = log10(gramos) ~ log10(tiempo), data = d.f)
##
## Coefficients:
##   (Intercept)  log10(tiempo)
##        -1.093          3.298
```

```r
summary(lm(log10(gramos) ~ log10(tiempo), data = d.f))$r.squared
```

```
## [1] 0.9982009
```

Lo que acabamos de obtener es que

$$\log(gramos) = 3.298 \cdot \log(tiempo) - 1.093$$

es una buena aproximación de nuestros datos.

Con lo cual

$$gramos = 10^{3.298\cdot\log(tiempo)} \cdot 10^{-1.093} = tiempo^{3.298} \cdot 0.081$$

```
plot(d.f)
curve( x^(3.298)*0.081, add = T, col = "purple")
```