MODELIZACIÓN

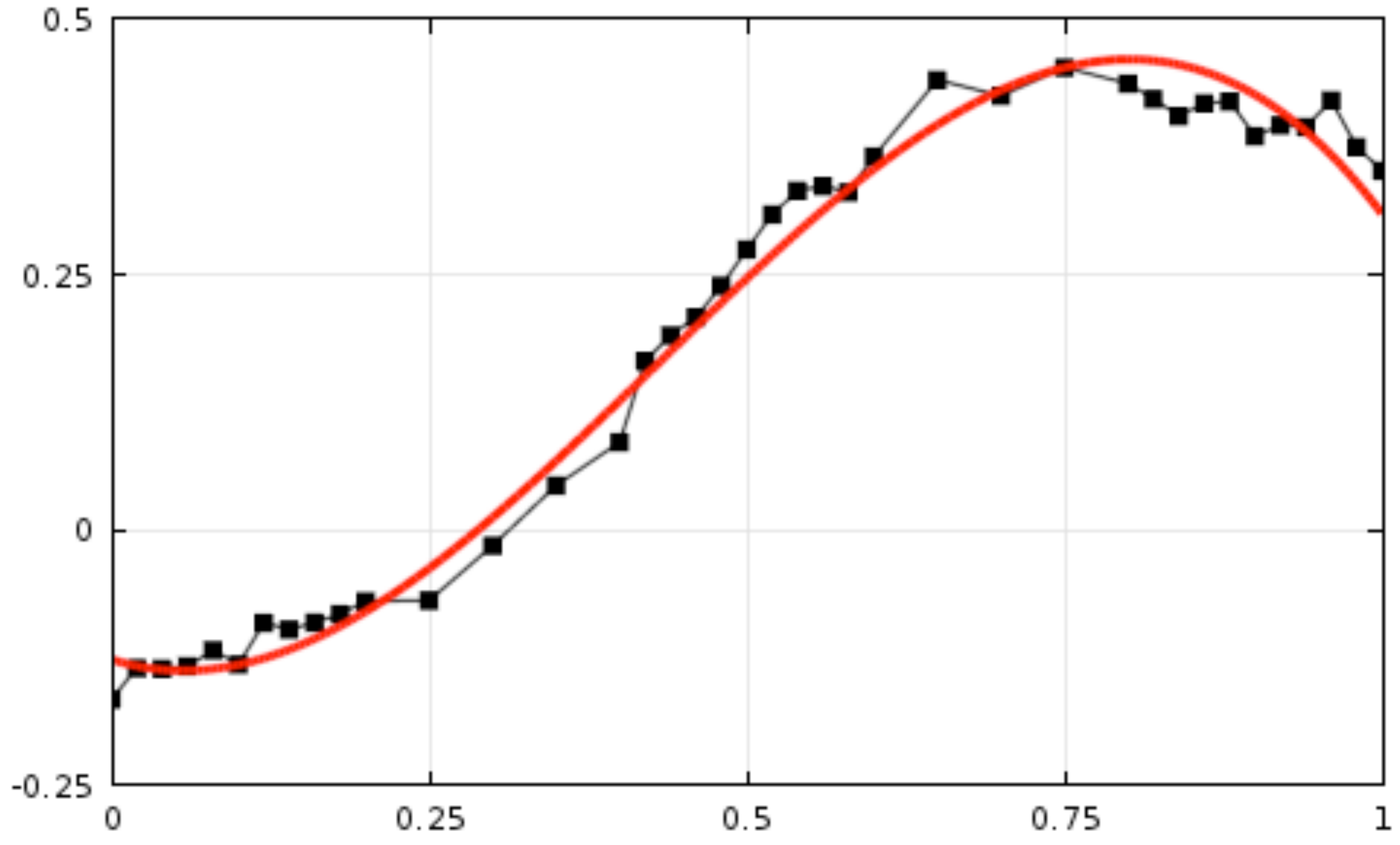
BIG DATA CON PYTHON

Biuse Casaponsa

https://github.com/biuse/2021Modelizacion

OUTLINE

1. Interpolación y extrapolación
2. Ajuste mínimos cuadrados
3. Ejemplos python (notebook)
4. **Lab**



https://github.com/biuse/2021Modelizacion

MODELIZAJE

Encontrar la relación entre dos o más variables.

Mediante una función matemática que se ajuste
a los datos que tenemos.

Nos será muy útil para poder hacer predicciones.

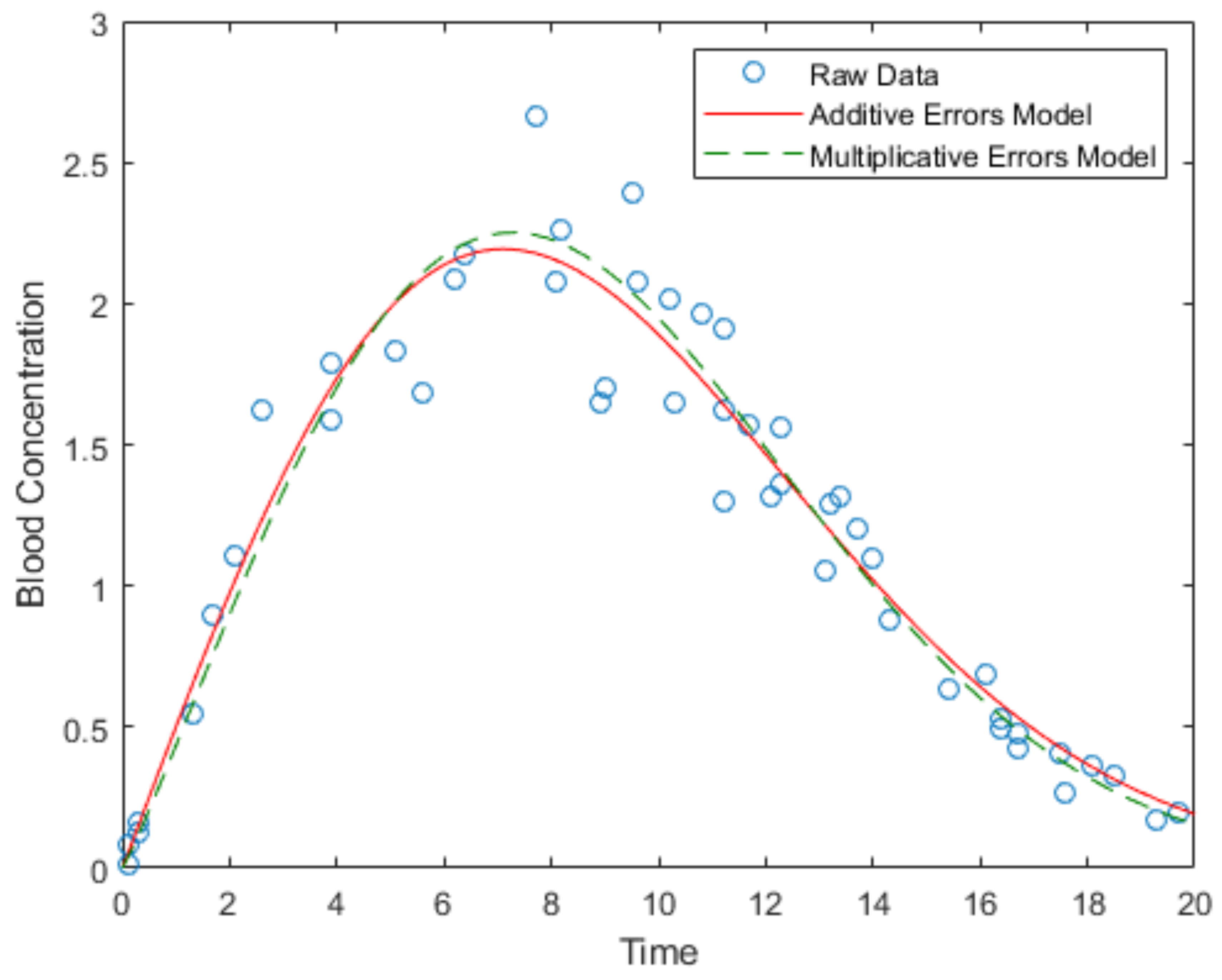
MODELIZAJE

Encontrar la relación entre dos o más variables.

Mediante una fun

a los datos que te

Nos será muy útil

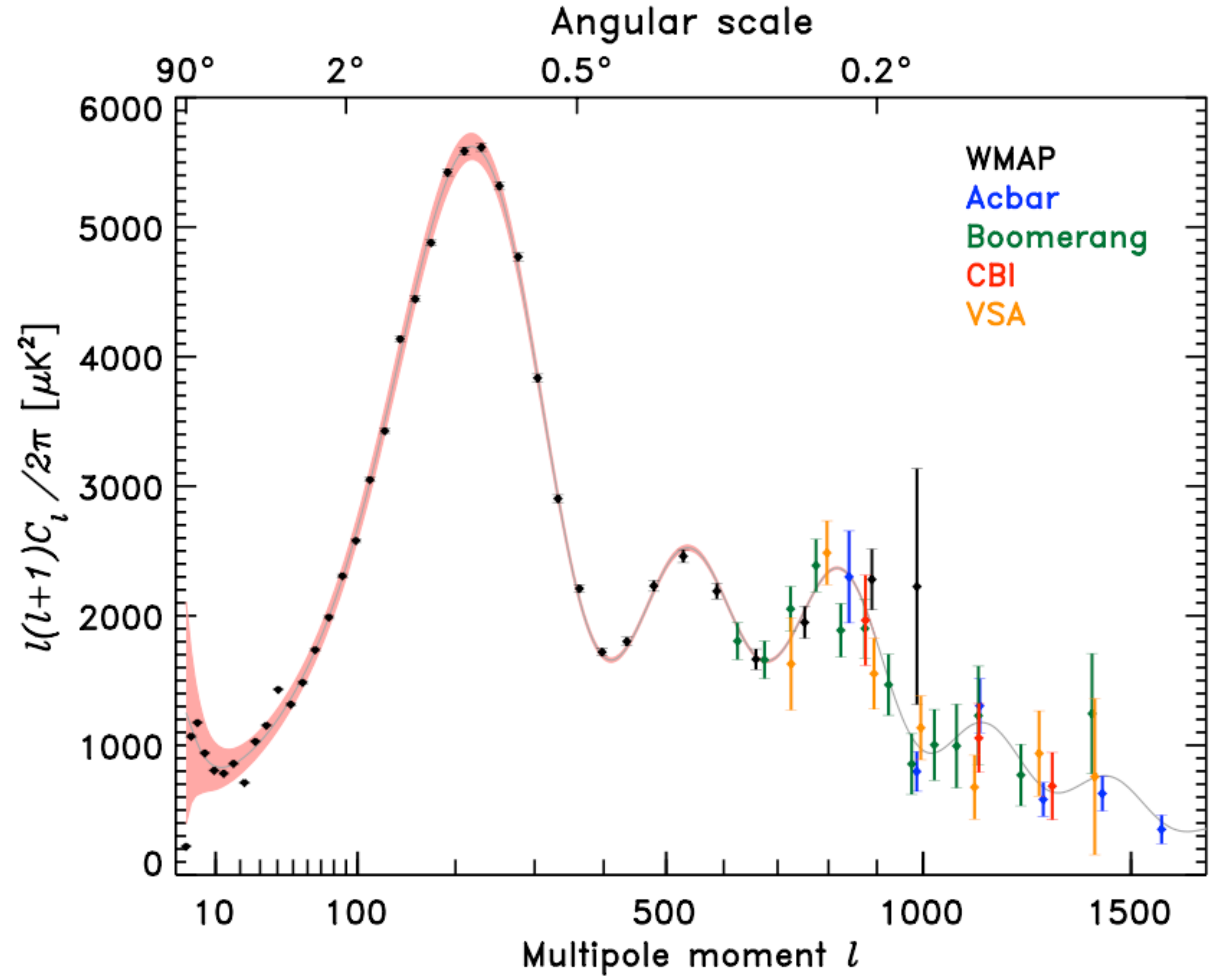


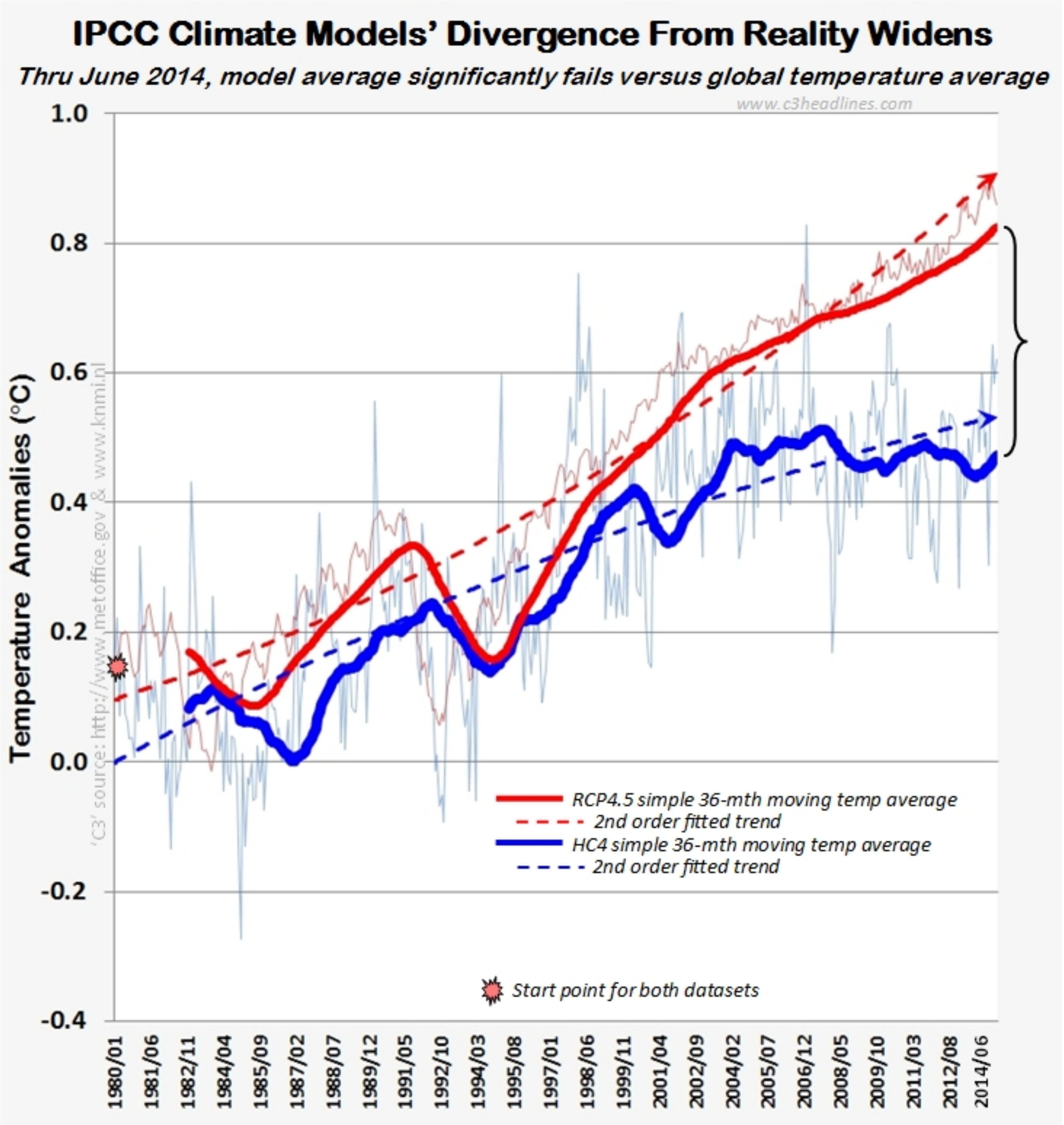
Concentración en sangre después de administrar un medicamento

Sueldo con la experiencia

MODELIZAR

Modelo téorico: Tenemos una relación matemática que puede estar dado por leyes conocidas o por hipotésis con una base teórica

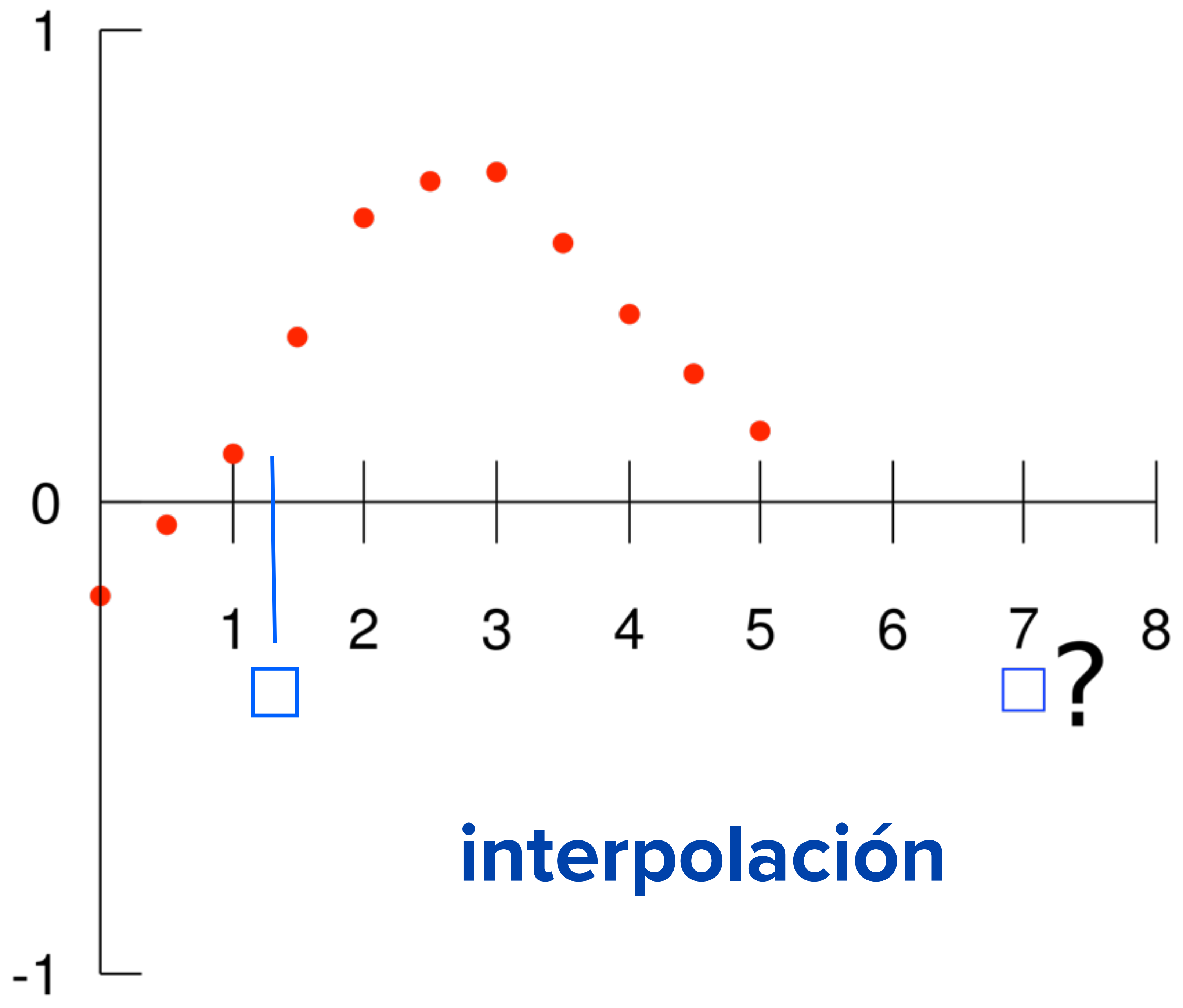




Modelo empírico: A partir de los datos intentamos encontrar una relación entre variables que nos permita predecir resultados con datos nuevos.

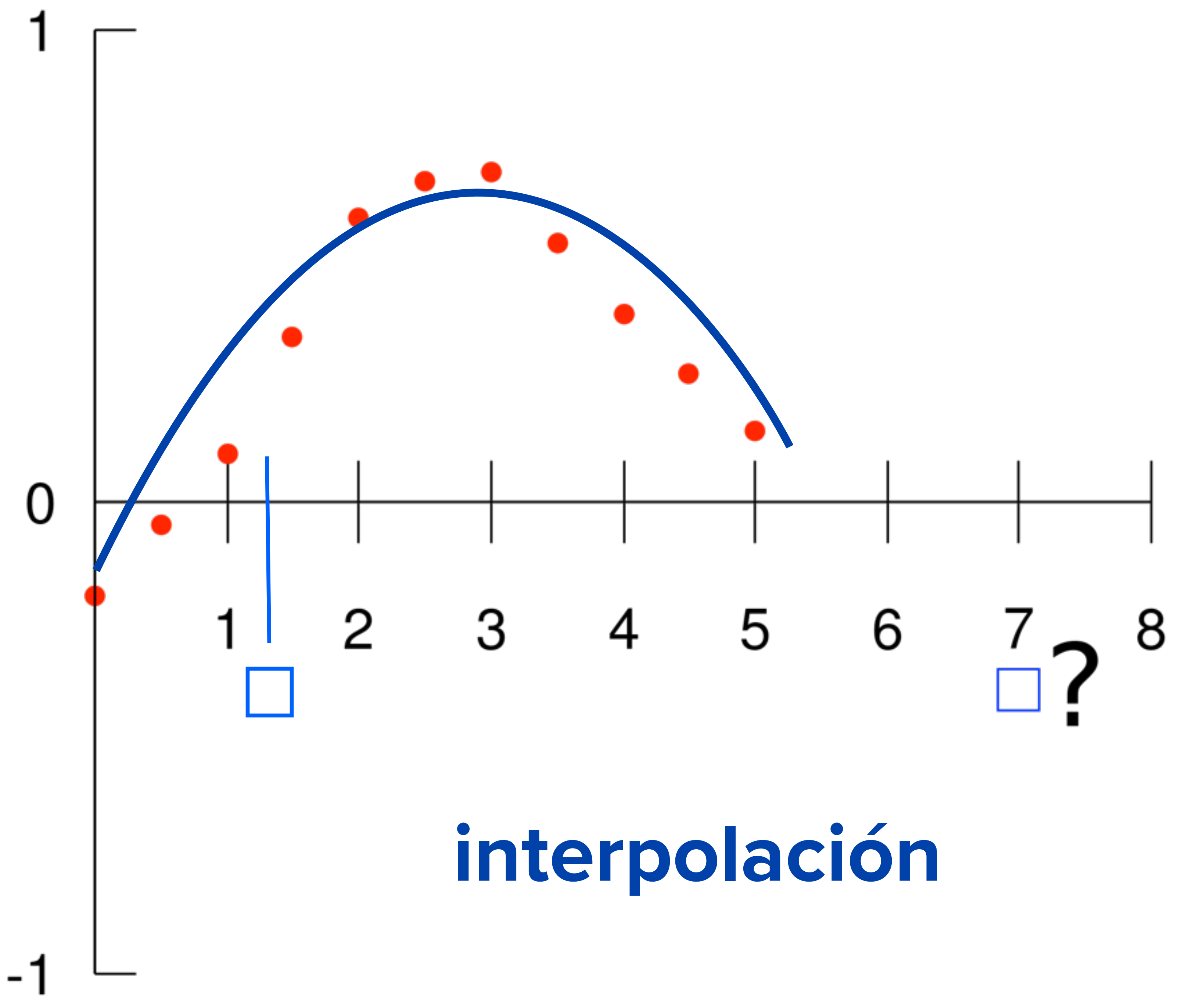
Ej : Ajuste a un polinomio (o **redes neuronales!)**

INTERPOLACIÓN Y EXTRAPOLACIÓN



extrapolación

INTERPOLACIÓN Y EXTRAPOLACIÓN



Modelo

extrapolación

INTERPOLACIÓN Y EXTRAPOLACIÓN

Ejemplo: Mareas

Entre las **9:00 y las 19:00** medimos a cada hora (t) la distancia que ha subido el mar (d).

Queremos saber a que altura ha estado el agua a las 9:30h. **Interpolación**

Queremos saber a que altura estará el agua a las 20 horas. **Extrapolación**

Queremos saber la altura del agua a cada minuto durante todos los días. Necesitamos un **Modelo**.

INTERPOLACIÓN Y EXTRAPOLACIÓN

Ejemplo: Mareas

Entre las **9:00 y las 19:00** medimos a cada hora (t) la distancia que ha subido el mar (d).

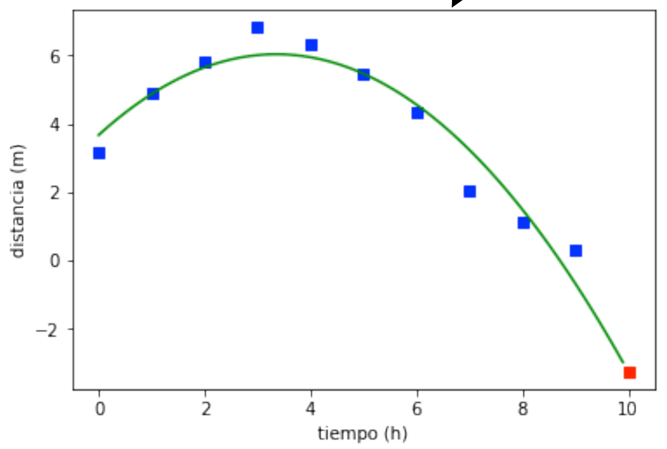
Queremos saber a que altura ha estado el agua a las 9:30h. **Interpolación**

Queremos saber a que altura estará el agua a las 20 horas. **Extrapolación**

Queremos saber la altura del agua a cada minuto durante todos los días. Necesitamos un **Modelo.**

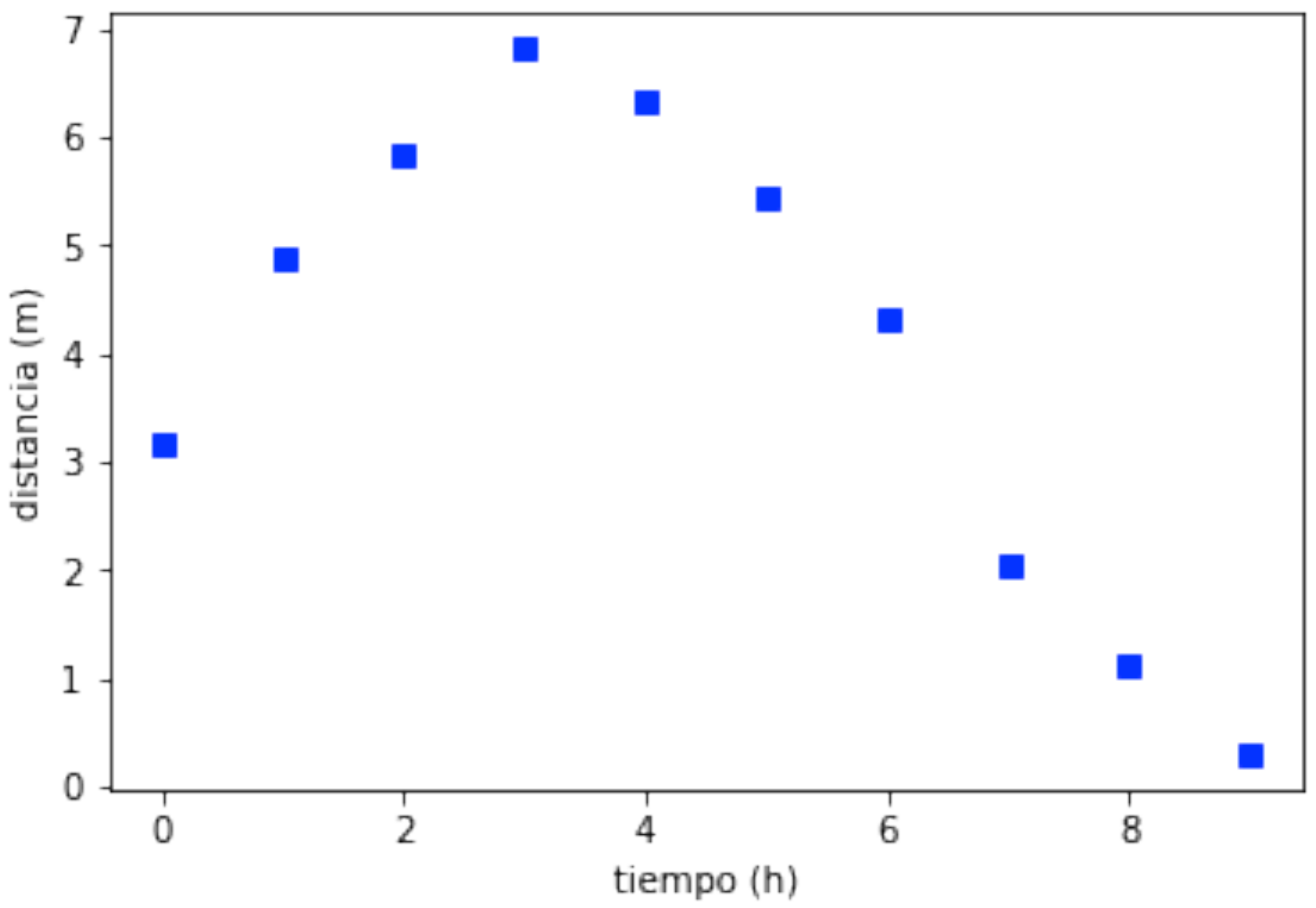
$$f_L = 2\frac{GM_L}{r^2}\frac{R}{r}$$

teórico



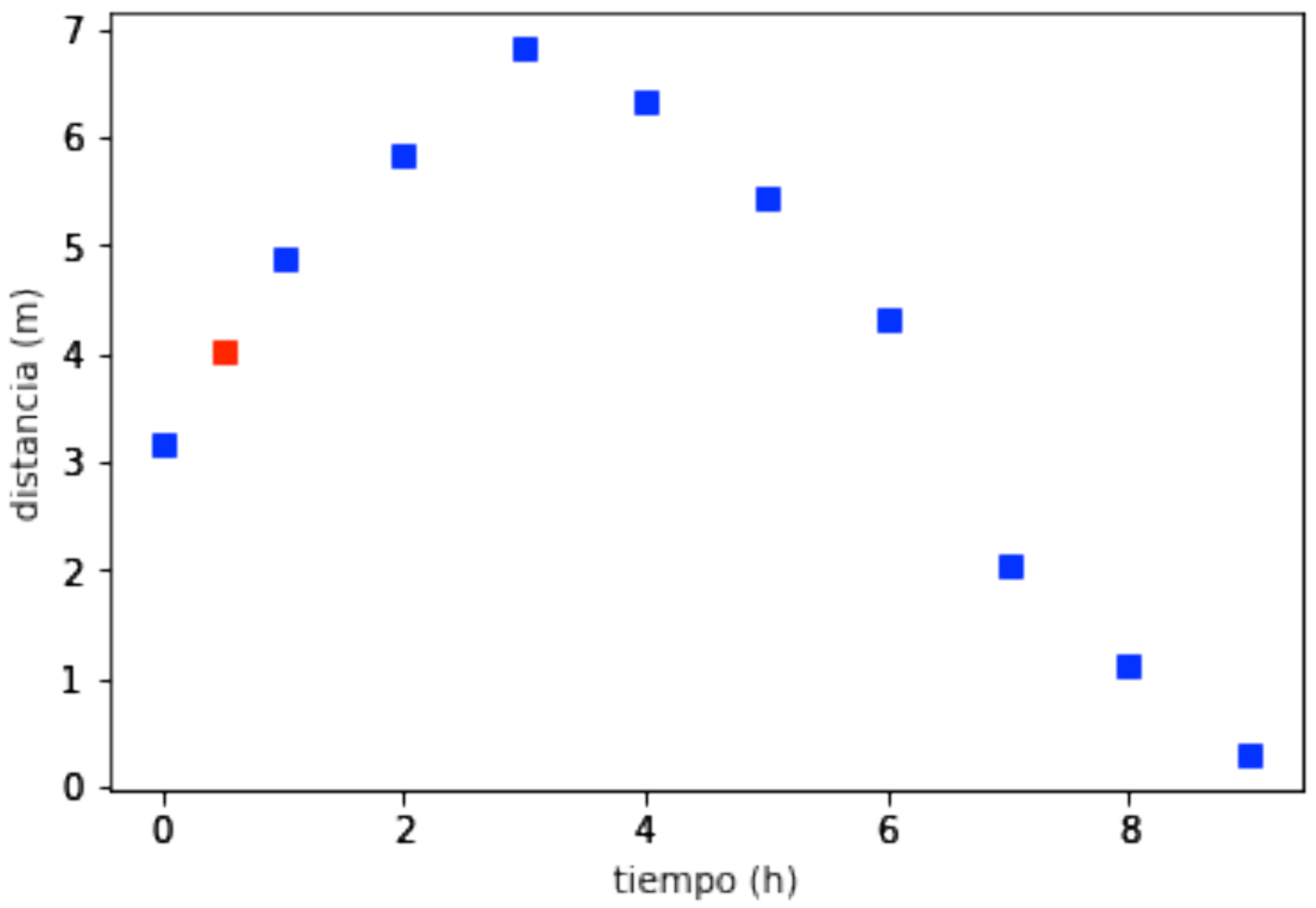
empírico

INTERPOLACIÓN Y EXTRAPOLACIÓN



Con estos datos, podemos intentar inferir que es lo que pasaba a 9:30 h

INTERPOLACIÓN Y EXTRAPOLACIÓN

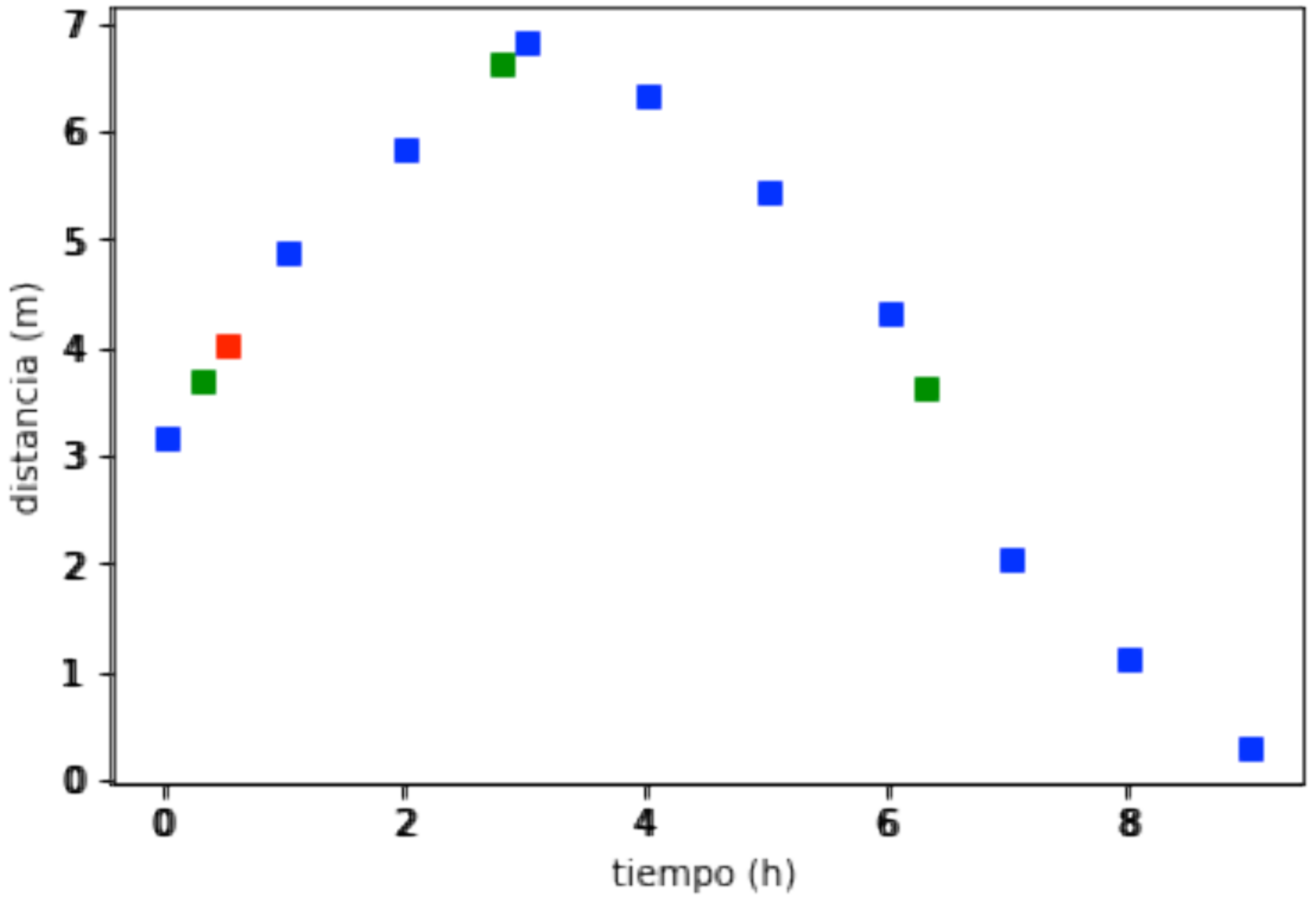


Con estos datos, podemos intentar inferir que es lo que pasaba a 9:30 h

por ejemplo: haciendo la media

t=0.5 --> $(d_0+d_1)/2$

INTERPOLACIÓN Y EXTRAPOLACIÓN



Con estos datos, podemos intentar inferir que es lo que pasaba a 9:30 h

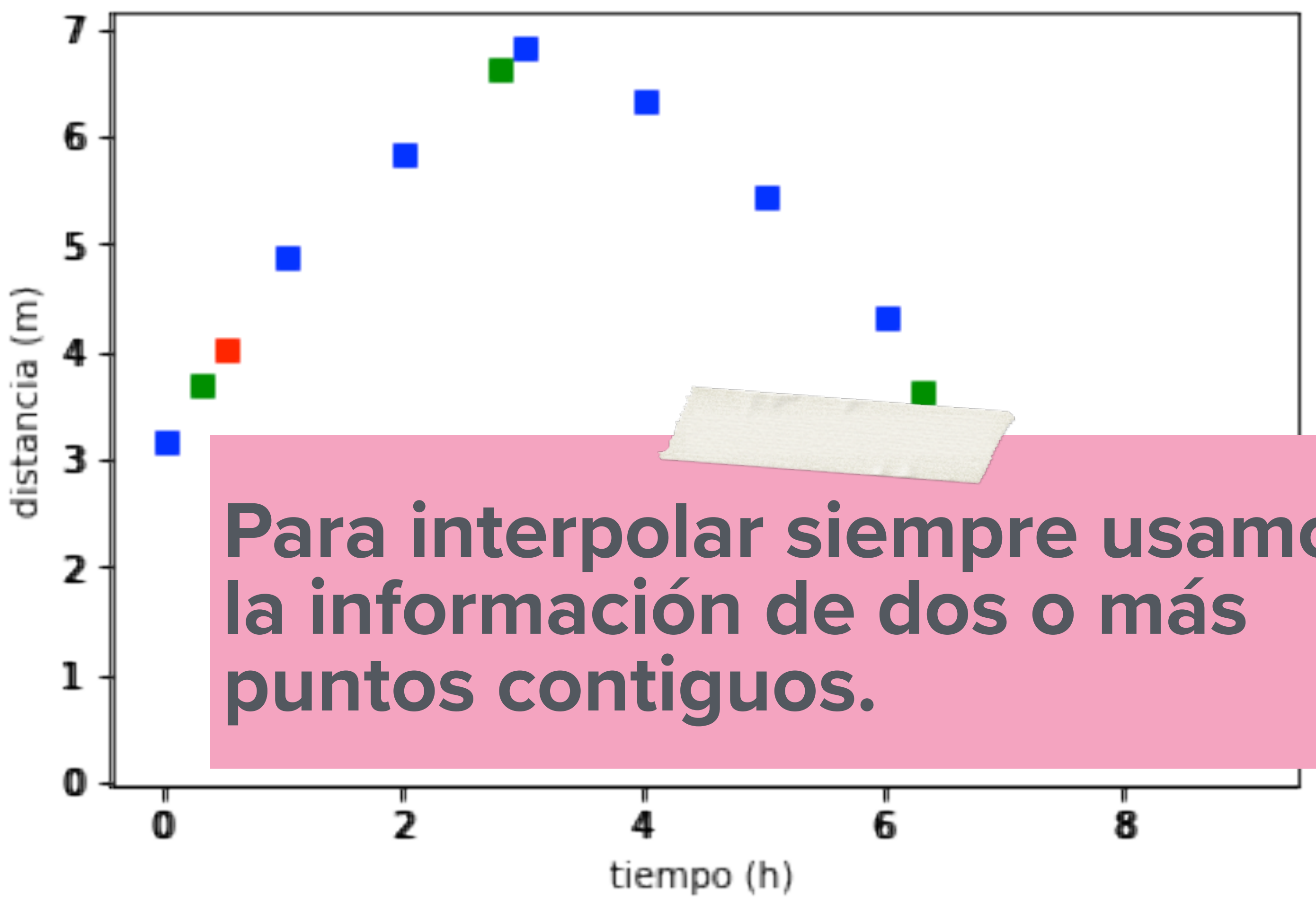
por ejemplo: haciendo la media

t=0.5 --> $(d_0+d_1)/2$

Para cualquier punto hay varios métodos de interpolación:

scipy.interpolate.interp1d(*x*, *y*, *kind='linear'*, ...)

INTERPOLACIÓN Y EXTRAPOLACIÓN



Para interpolar siempre usamos la información de dos o más puntos contiguos.

Con estos datos, podemos intentar inferir que es lo que pasaba a 9:30 h

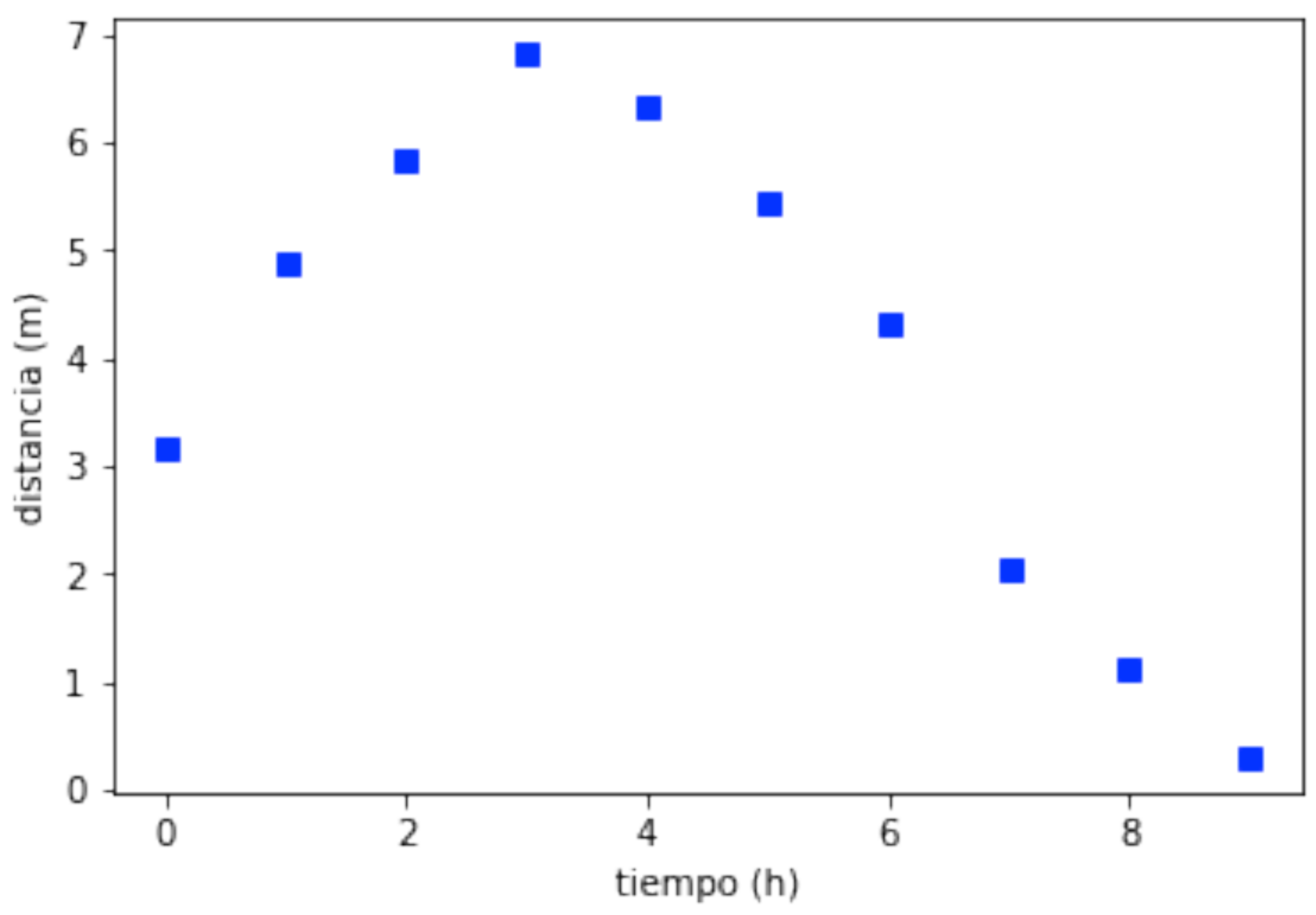
por ejemplo: haciendo la media

$t=0.5 \;\text{-->}\; (d_0+d_1)/2$

Para cualquier punto hay varios métodos de interpolación:

scipy.interpolate.interp1d(*x*, *y*, *kind='linear'*, ...)

INTERPOLACIÓN Y EXTRAPOLACIÓN

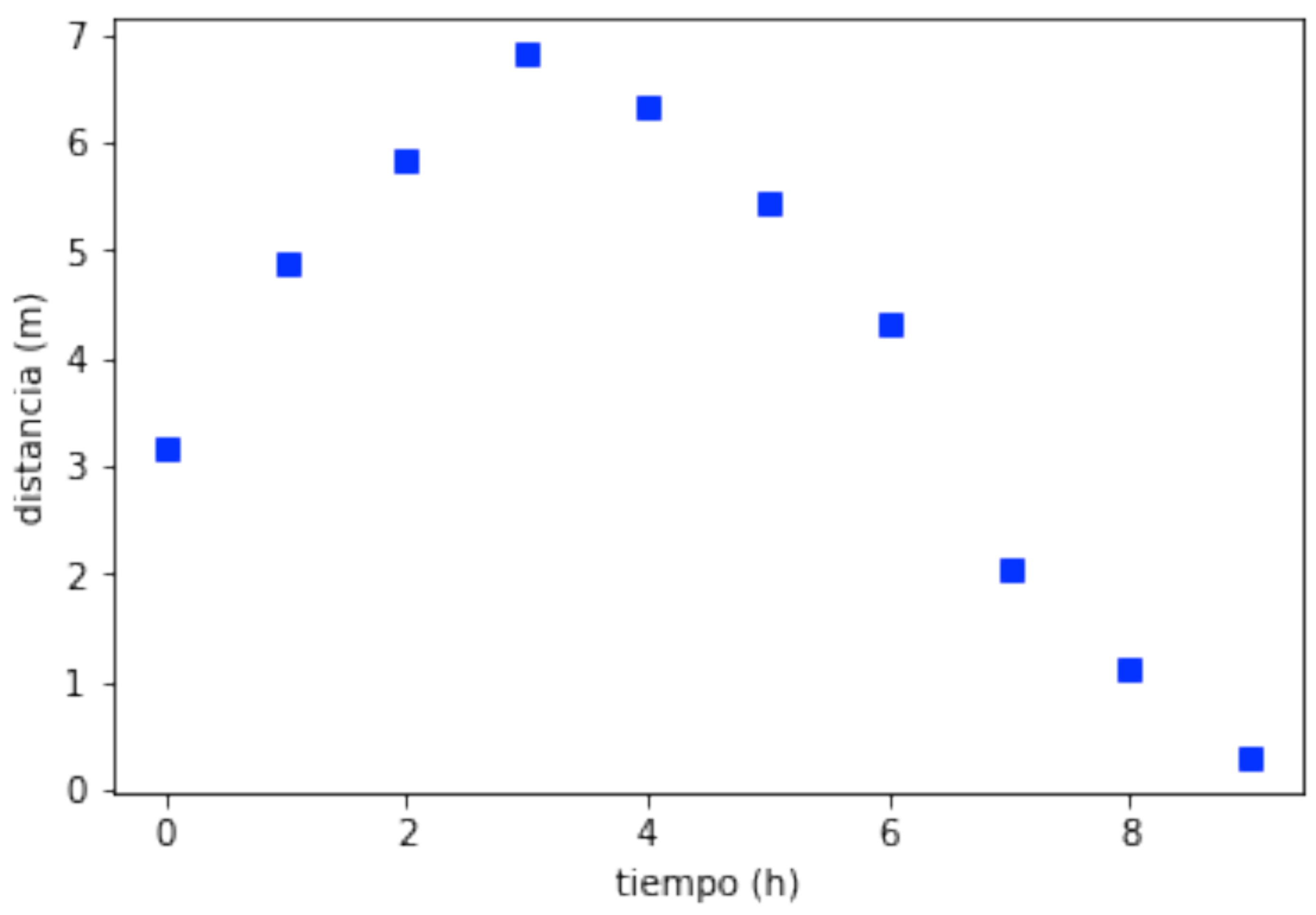


??

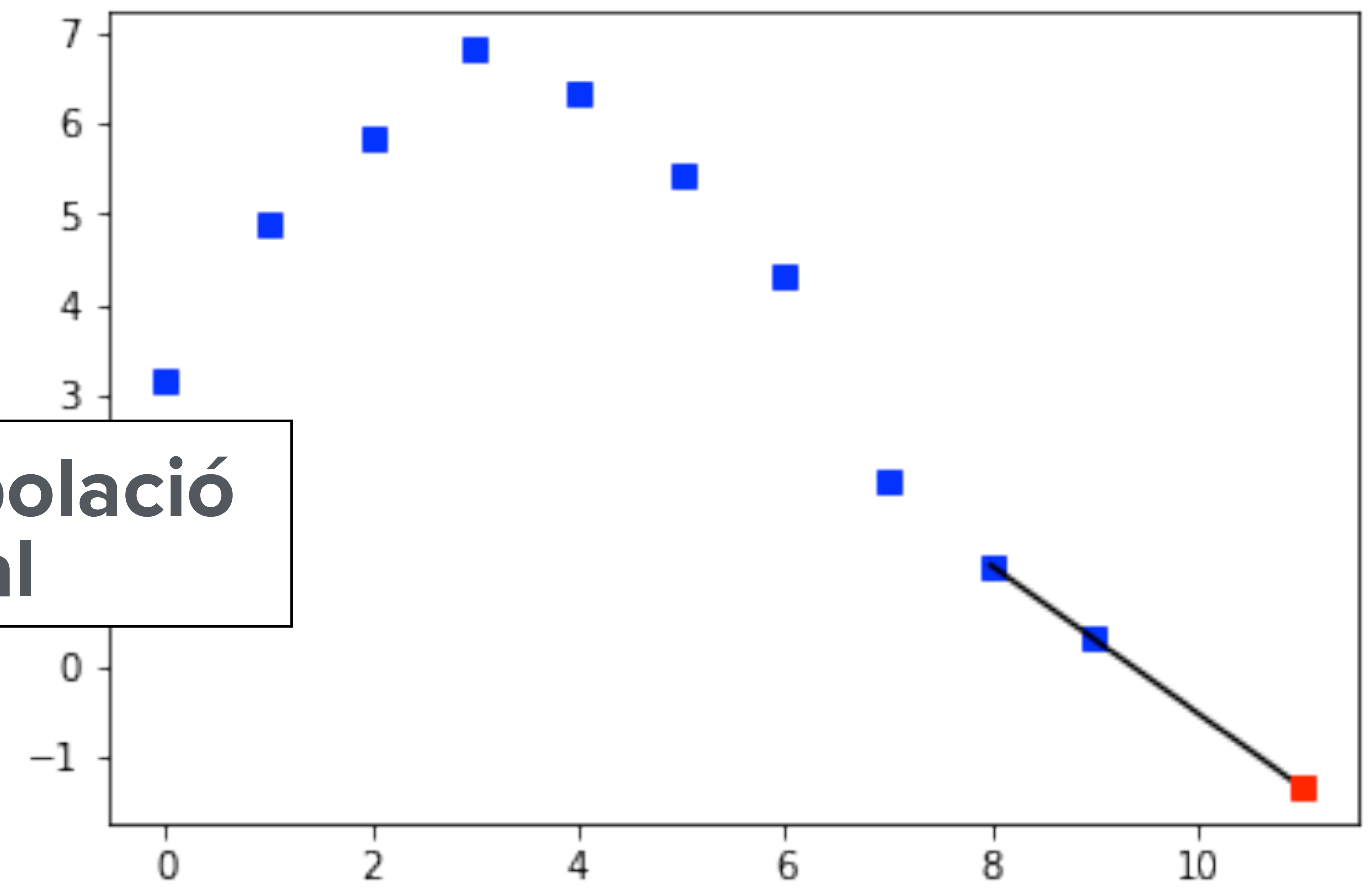
Si queremos saber lo distancia del agua a las a las 20h. Para extrapolar solo podemos usar los puntos anteriores.

INTERPOLACIÓN Y EXTRAPOLACIÓN

Si queremos saber lo distancia del agua a las a las 20h. Para extrapolar solo podemos usar los puntos anteriores.



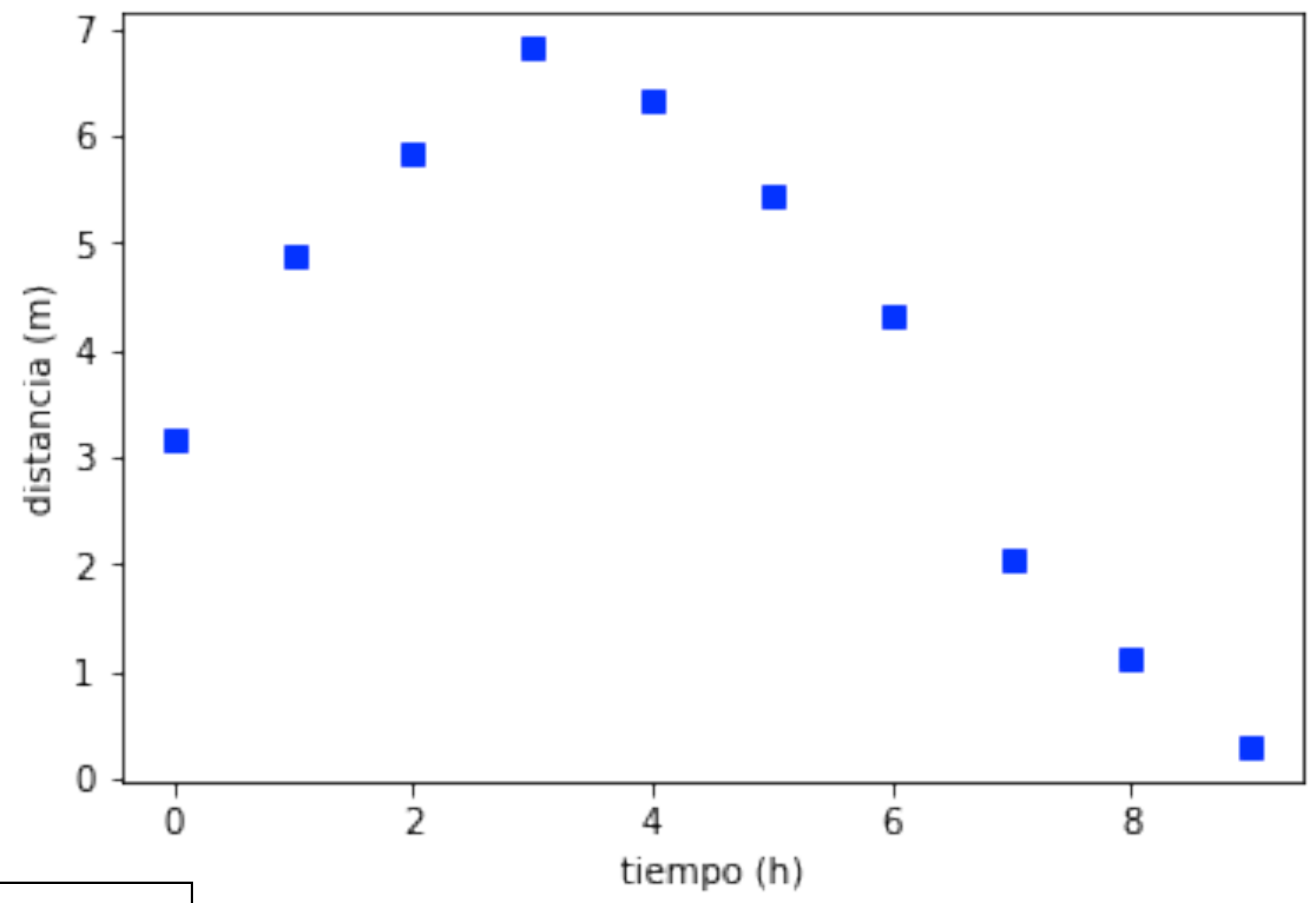
??



Extrapolación lineal

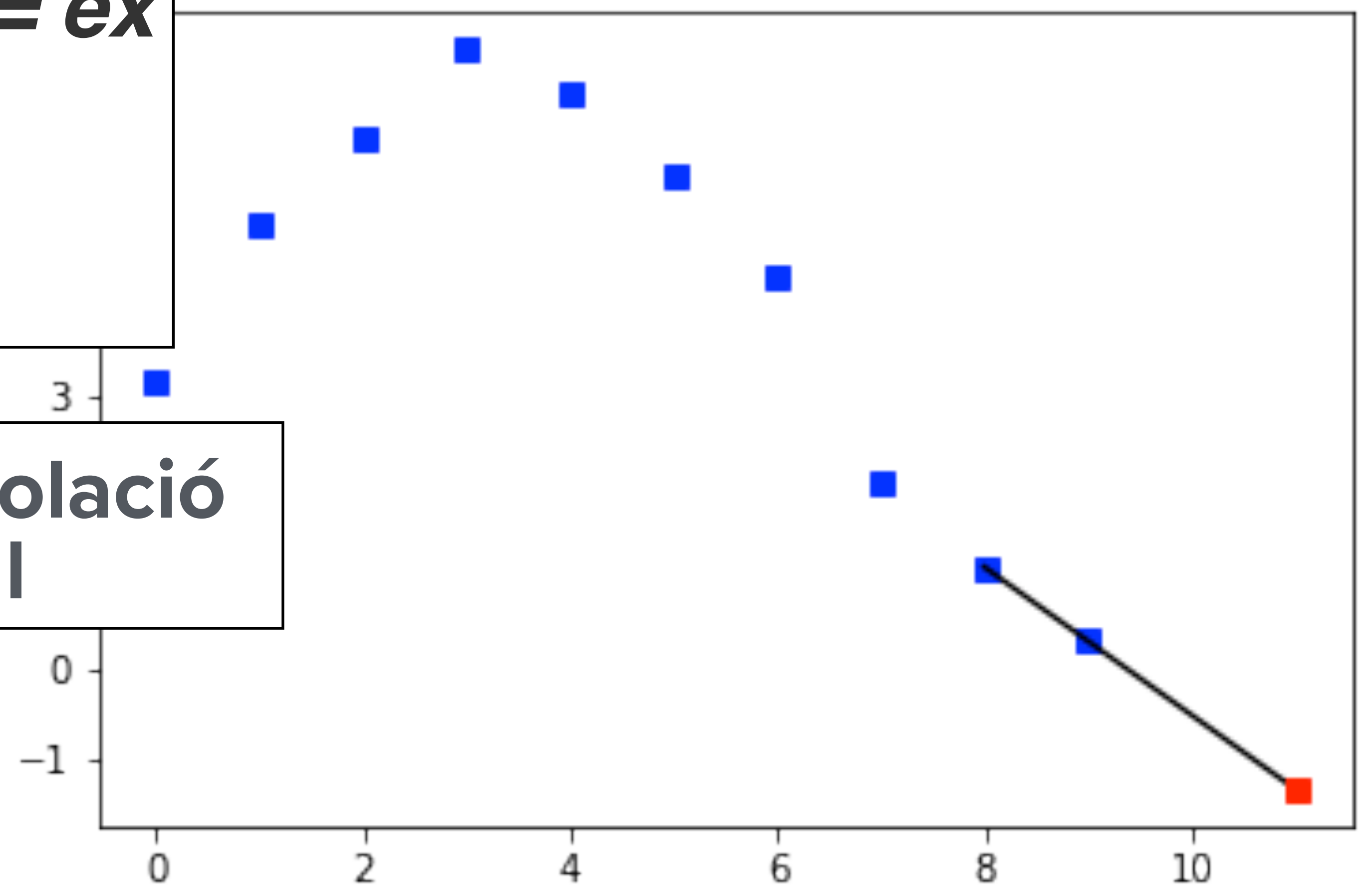
INTERPOLACIÓN Y EXTRAPOLACIÓN

Si queremos saber lo distancia del agua a las a las 20h. Para extrapolar solo podemos usar los puntos anteriores.



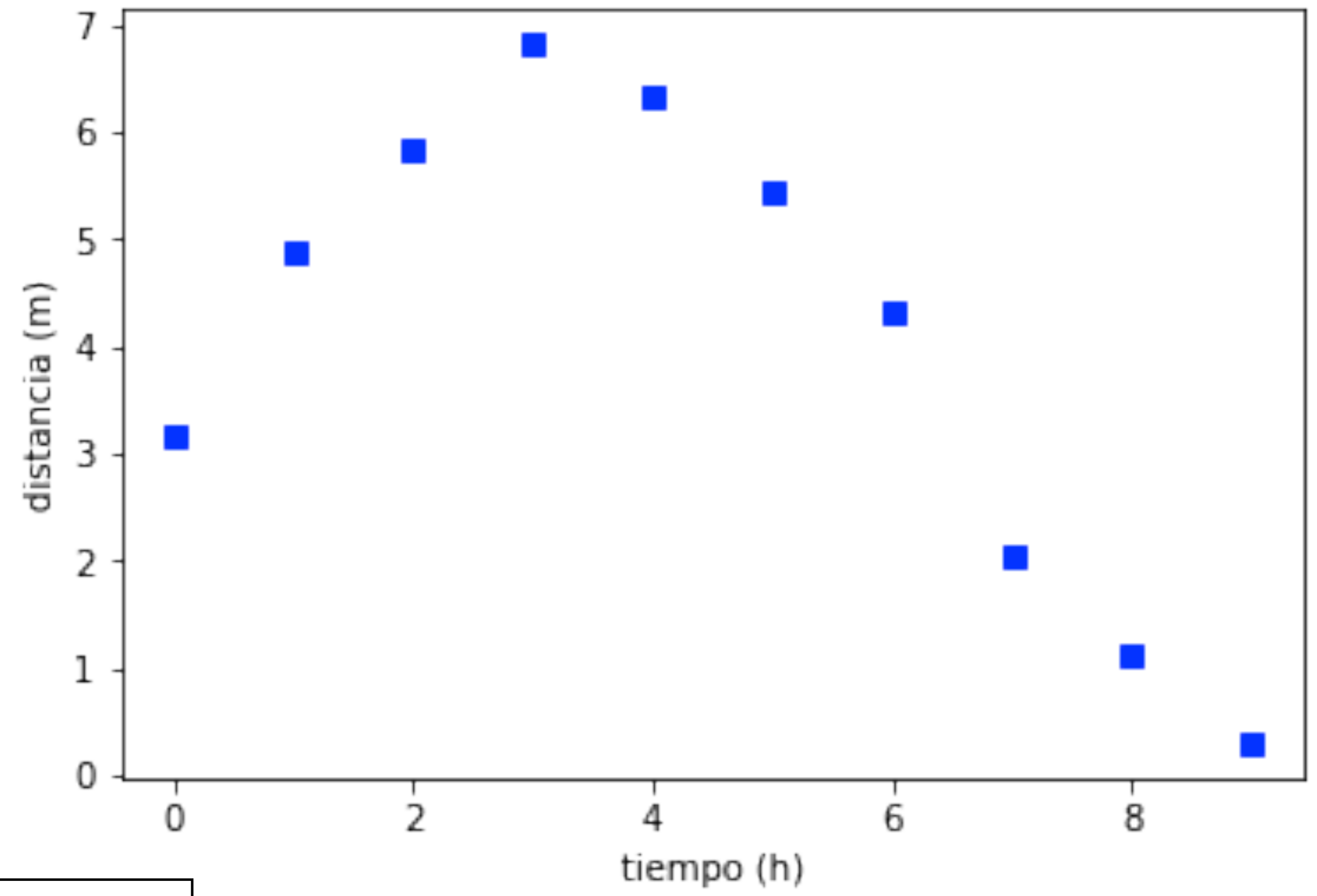
??

interp1d(*x*, *y*, *fill_val*=‘*extrapolate*’, ...)



Extrapolación lineal

INTERPOLACIÓN Y EXTRAPOLACIÓN

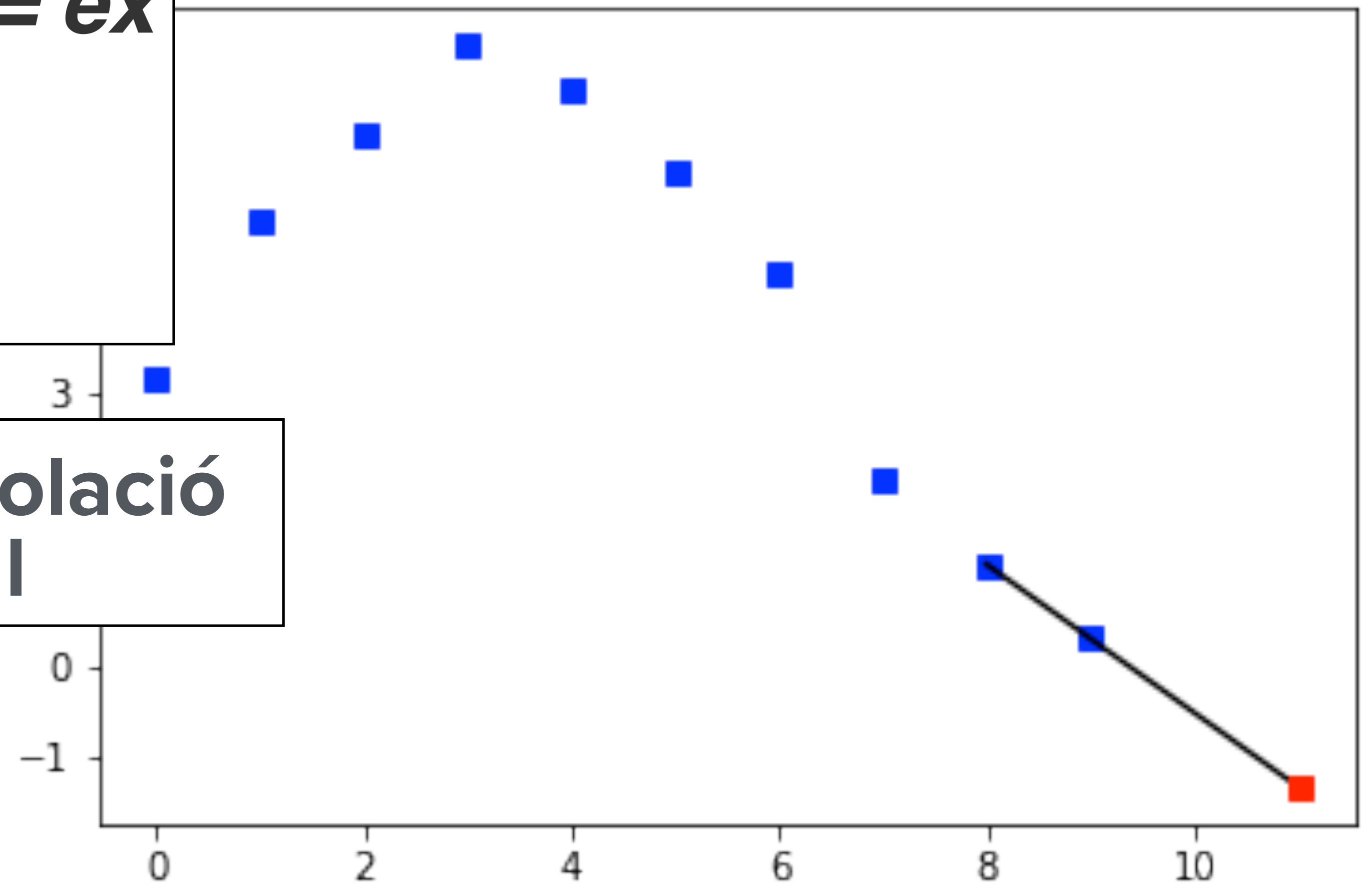


Si queremos saber lo distancia del agua a las a las 20h. Para extrapolar solo podemos usar los puntos anteriores.

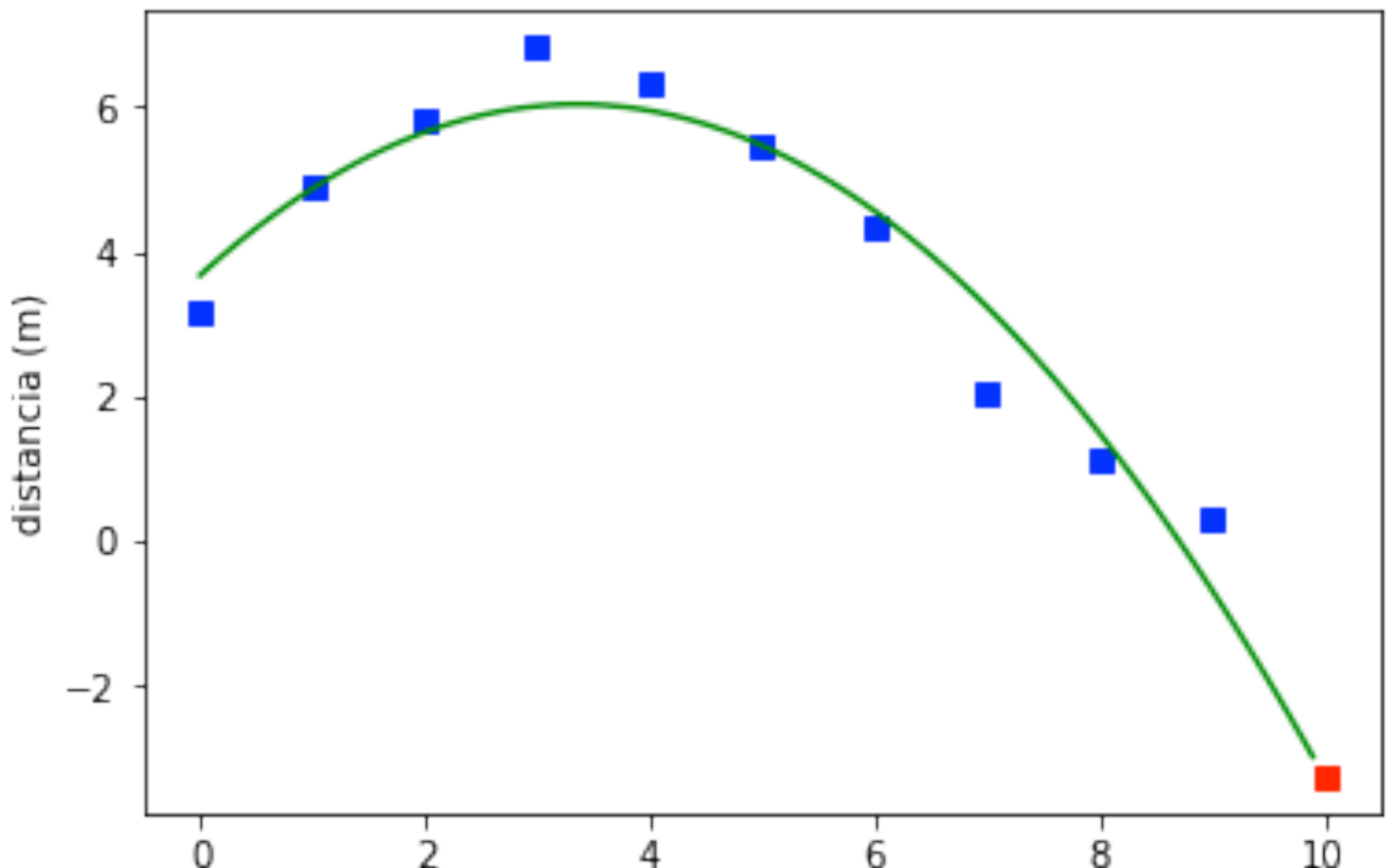
Como antes podemos usar otras funciones, parábola, y podemos usar dos o más puntos.

??

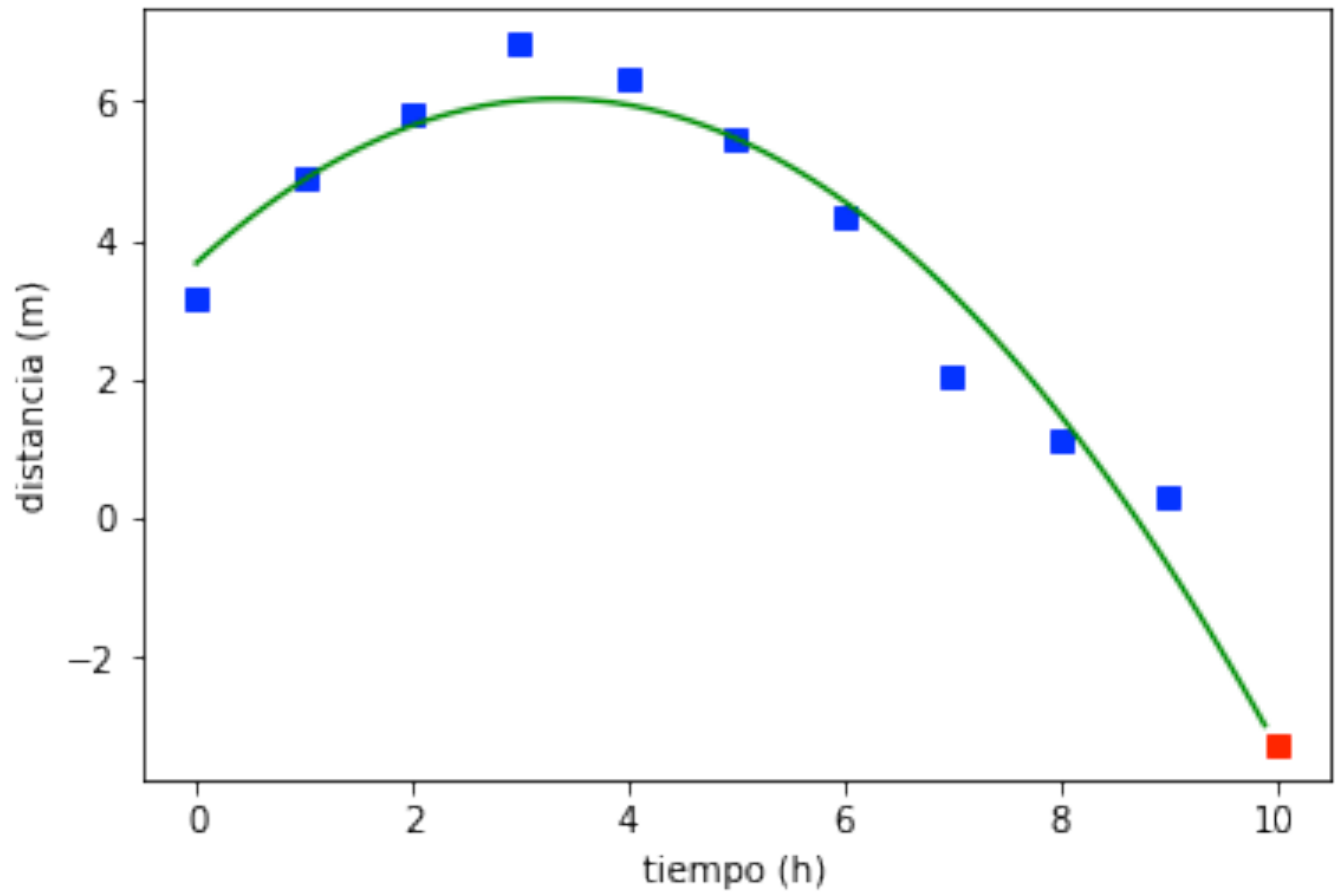
interp1d(*x*, *y*, *fill_val='extrapolate'*, ...)



Extrapolación lineal



INTERPOLACIÓN Y EXTRAPOLACIÓN

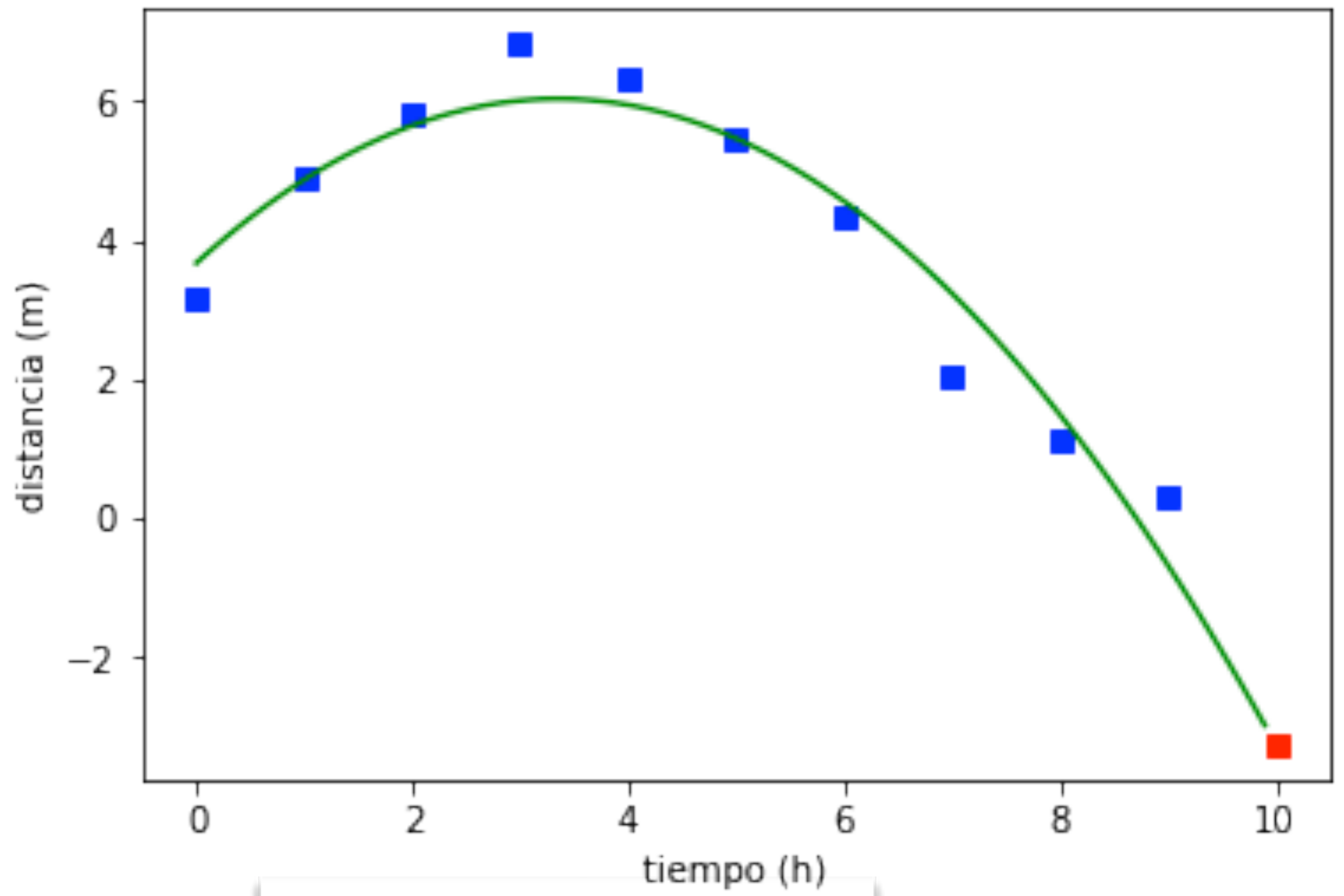


Si suponemos que es una parábola:

$$d = at^2 + bt + c$$

y hacemos un ajuste con **Python**: numpy.polyfit(t,d,2)

INTERPOLACIÓN Y EXTRAPOLACIÓN



Si suponemos que es una parábola:

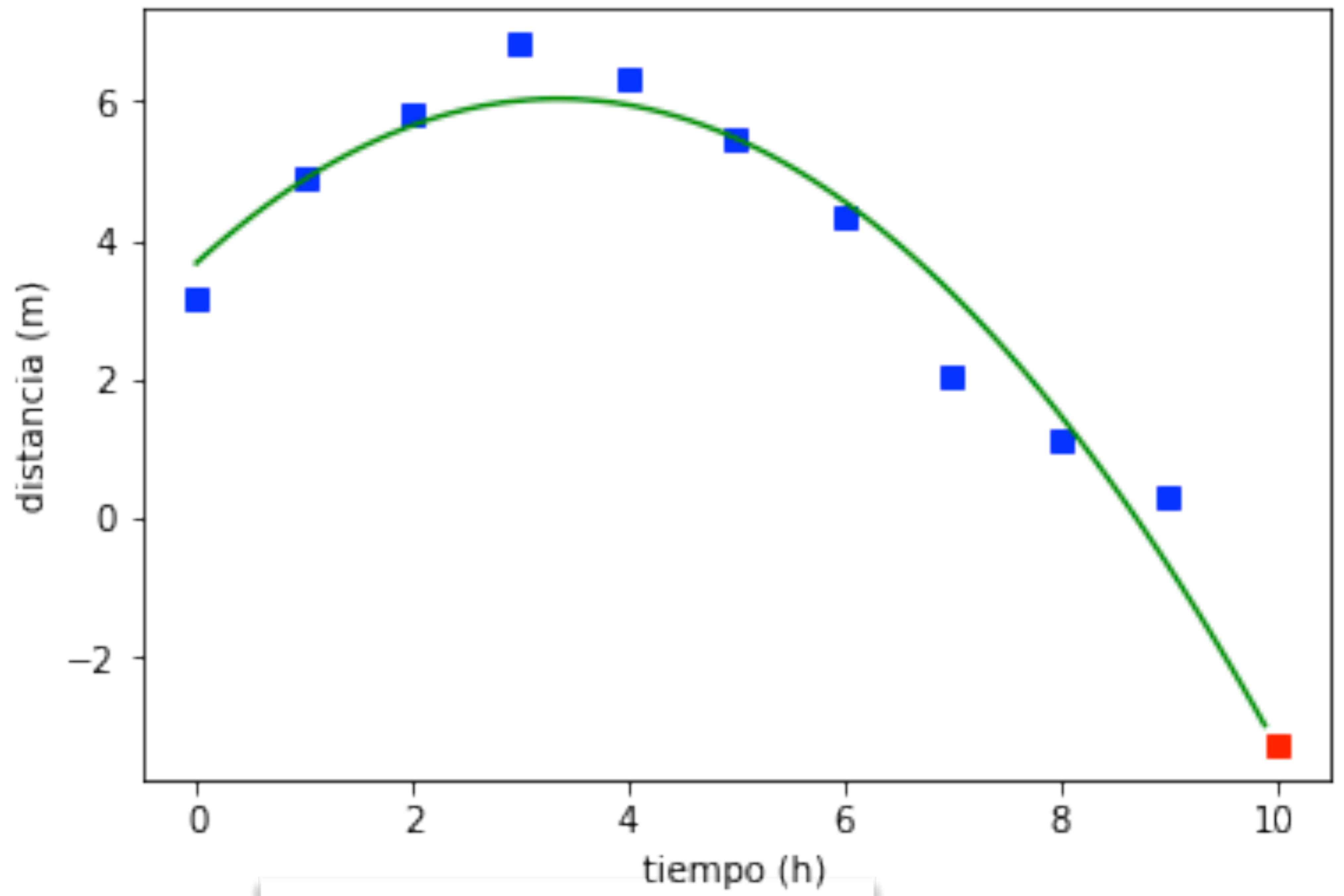
$$d = at^2 + bt + c$$

y hacemos un ajuste con **Python**:
numpy.polyfit(t,d,2)

np.polyfit me da los coeficientes

$$d = -0.2t^2 + 1.4t + 3.7$$

INTERPOLACIÓN Y EXTRAPOLACIÓN



np.polyfit me da los coeficientes

$$d = -0.2t^2 + 1.4t + 3.7$$

¡Ya está!

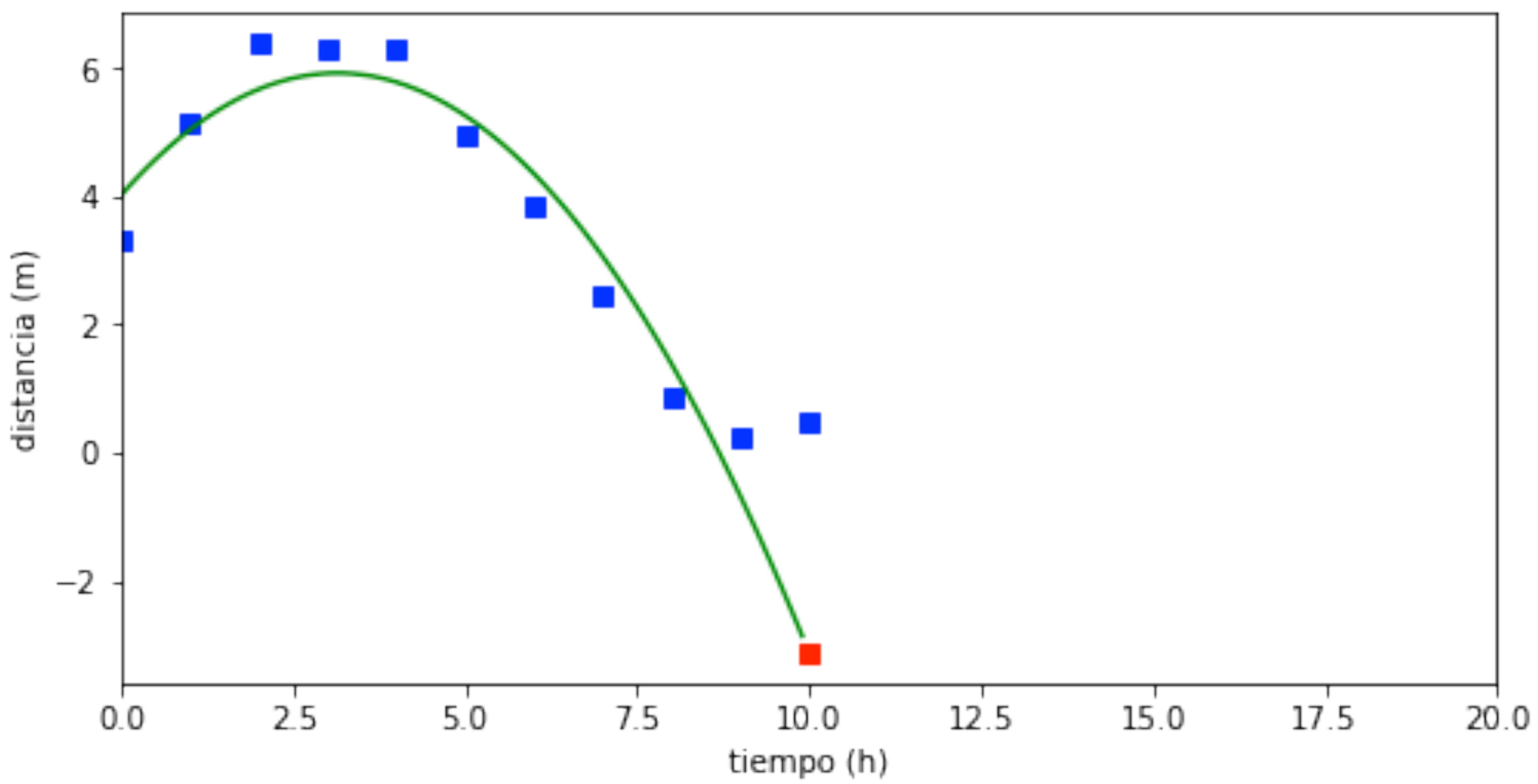
d=f(t)

A cada hora podemos saber la distancia que recorre el mar!

¿Ya podemos predecir la distancia a las 00 horas?

INTERPOLACIÓN Y EXTRAPOLACIÓN

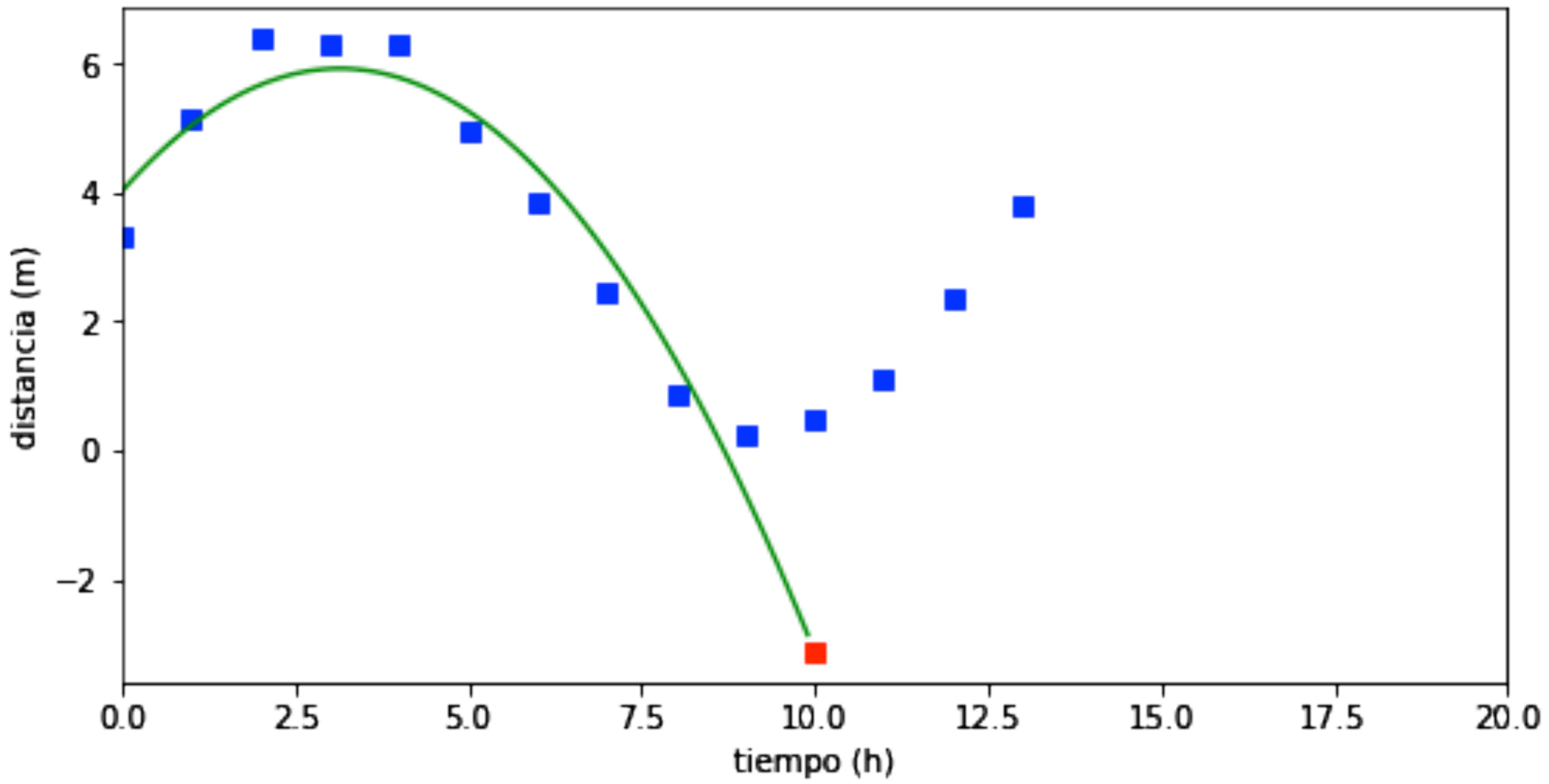
Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos



Tomamos más datos.

INTERPOLACIÓN Y EXTRAPOLACIÓN

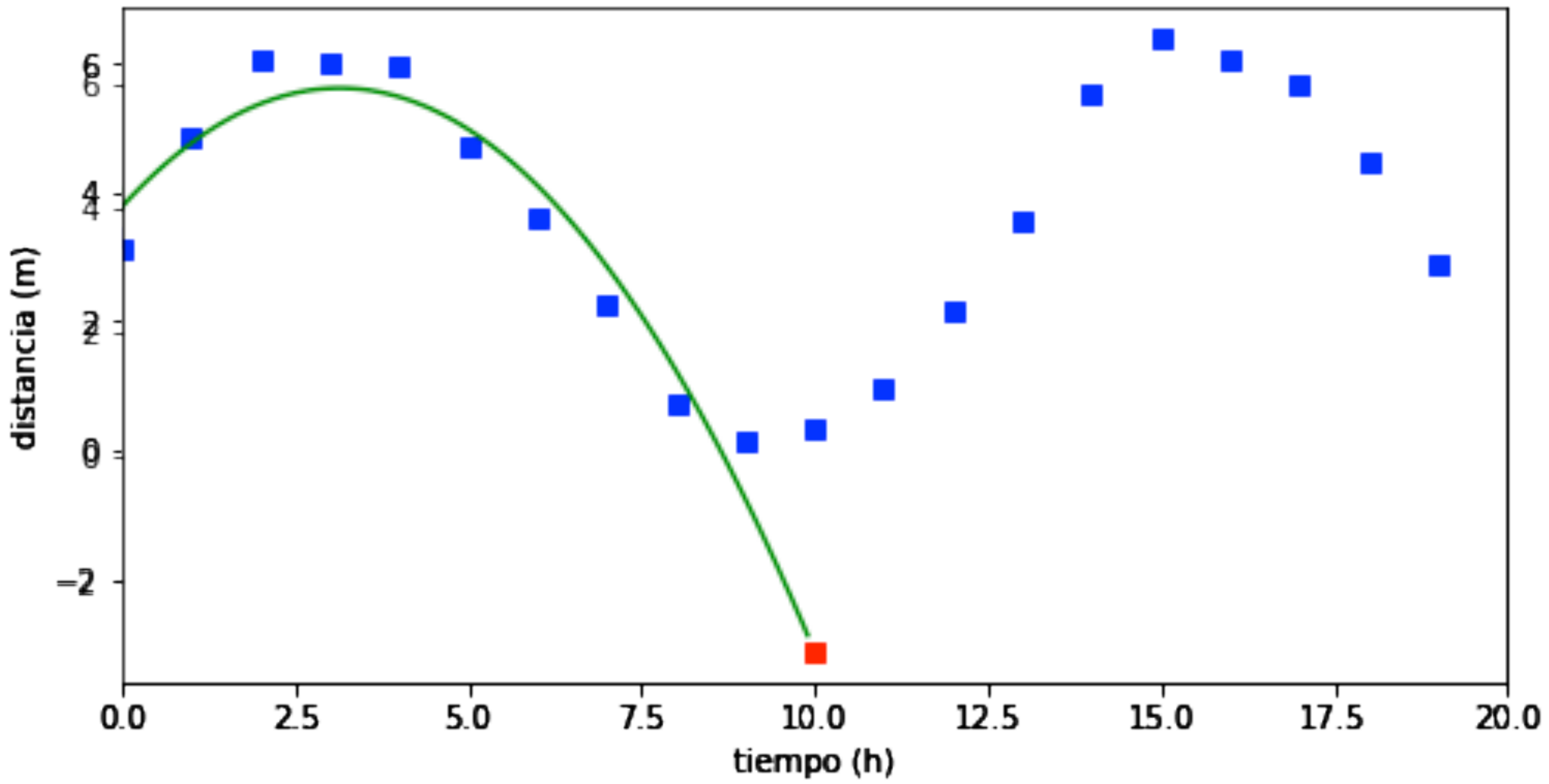
Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos



Tomamos más datos.

INTERPOLACIÓN Y EXTRAPOLACIÓN

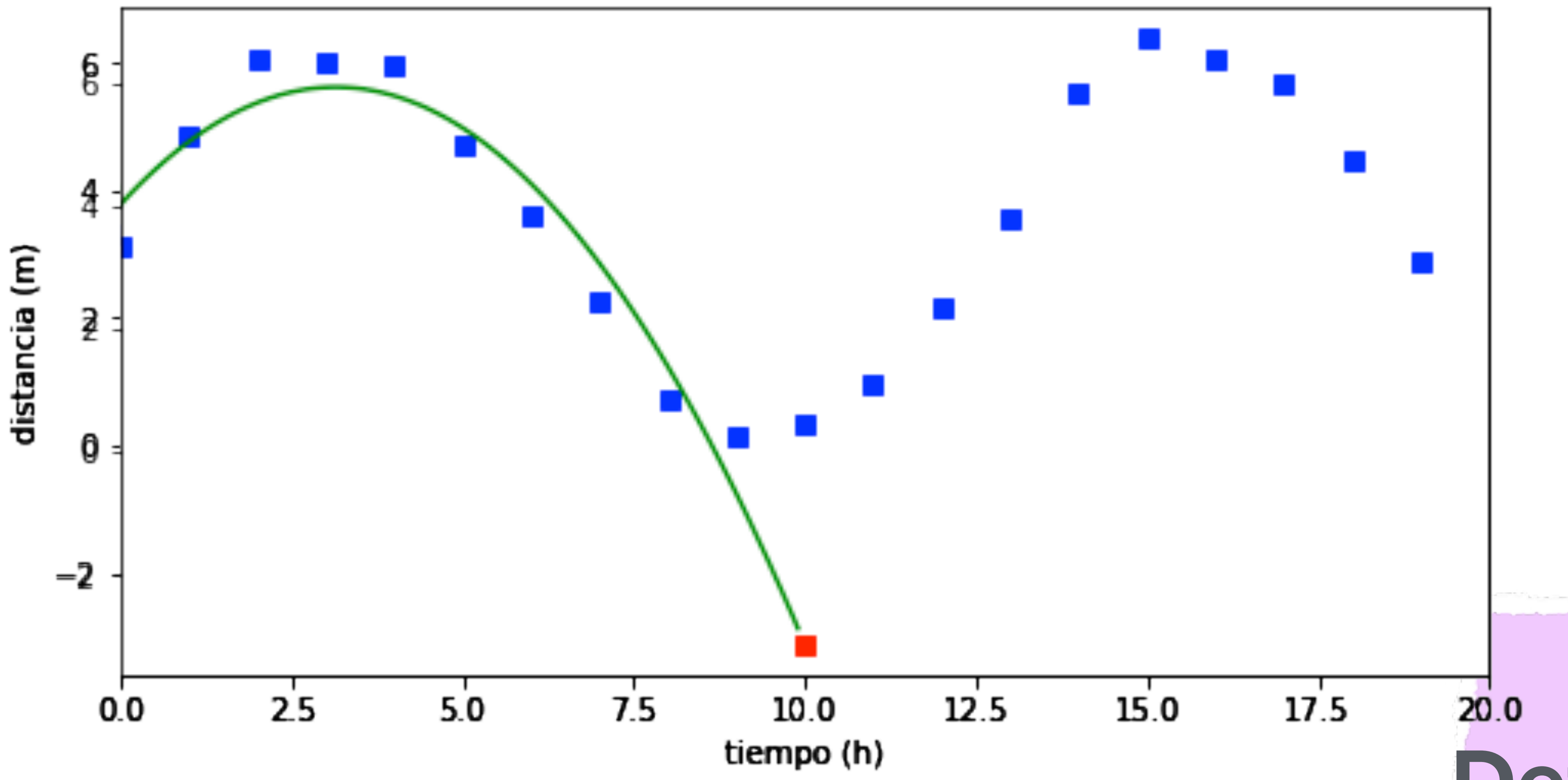
Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos



Tomamos más datos.

INTERPOLACIÓN Y EXTRAPOLACIÓN

Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos

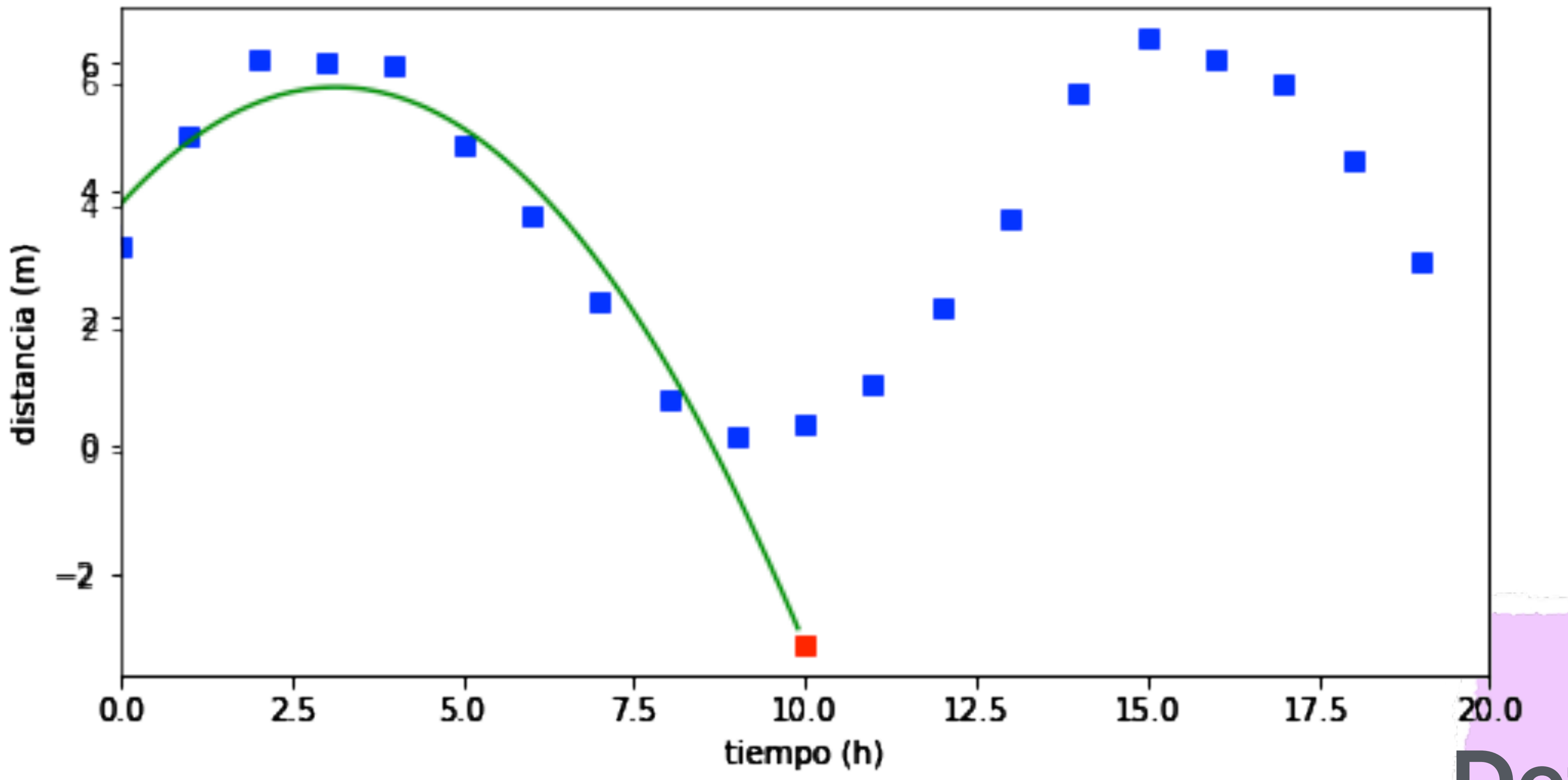


Tomamos más datos.

Definitivamente el modelo que hemos construido NO sirve

INTERPOLACIÓN Y EXTRAPOLACIÓN

Para que un modelo sea válido para extrapolar, tiene que ser correcto para nuevos datos



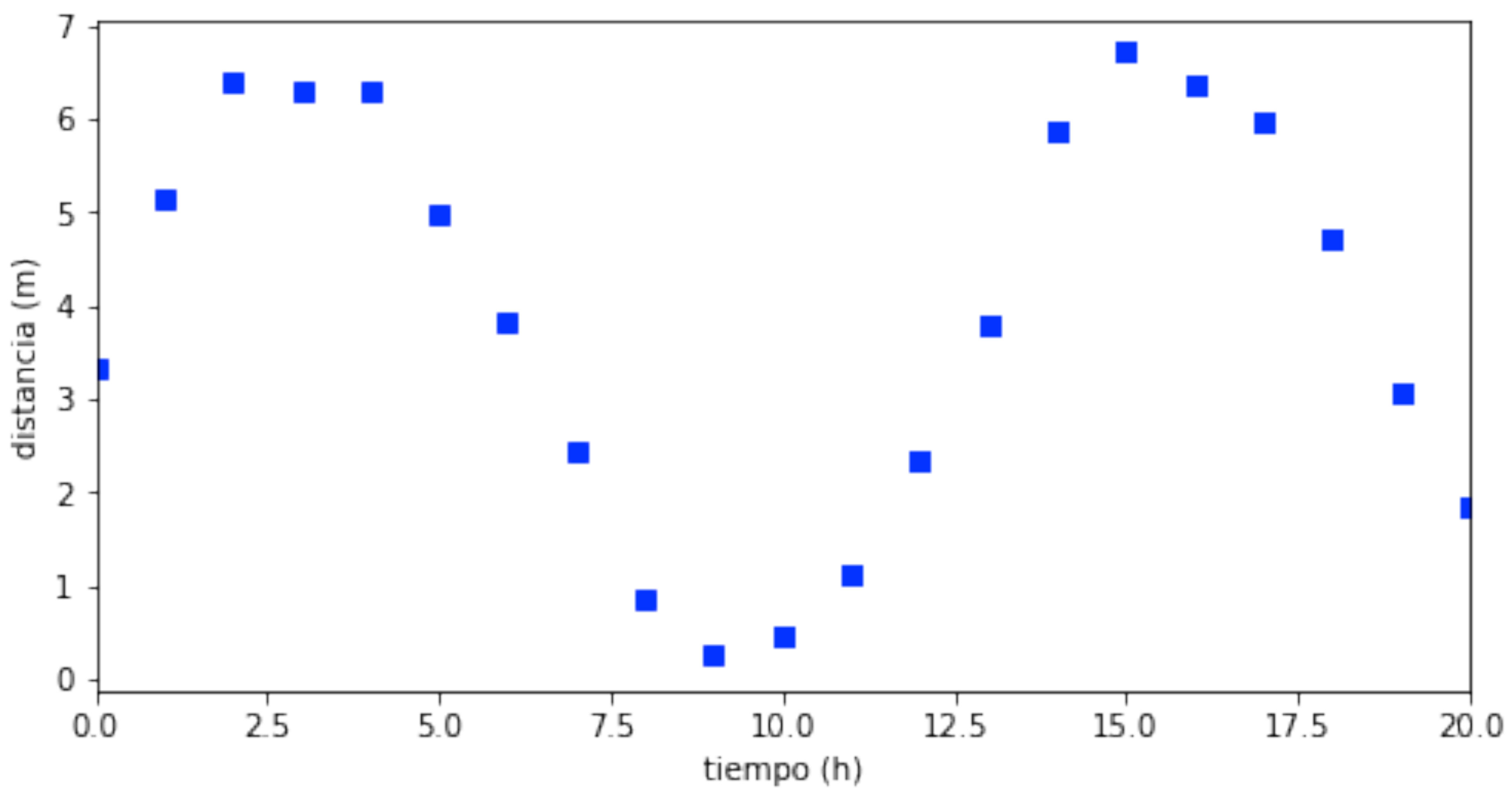
Tomamos más datos.

Definitivamente el modelo que hemos construido NO sirve

podríamos volver a buscar un polinomio con los nuevos datos, pero en general es solo válido para el rango del ajuste.

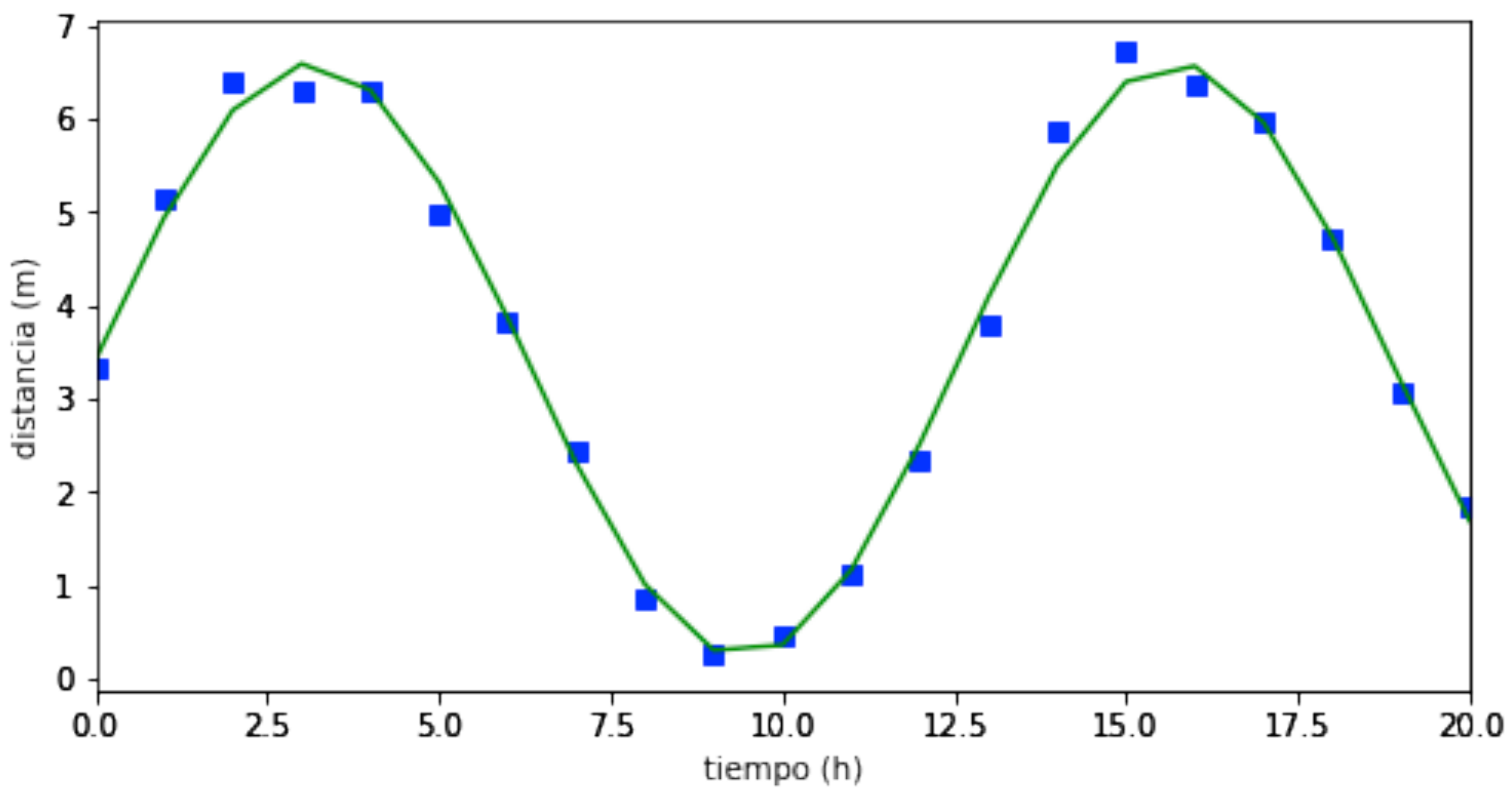
INTERPOLACIÓN Y EXTRAPOLACIÓN

Probamos otra función: $d = a\sin(bt) + c$



INTERPOLACIÓN Y EXTRAPOLACIÓN

Probamos otra función: $d = a\sin(bt) + c$



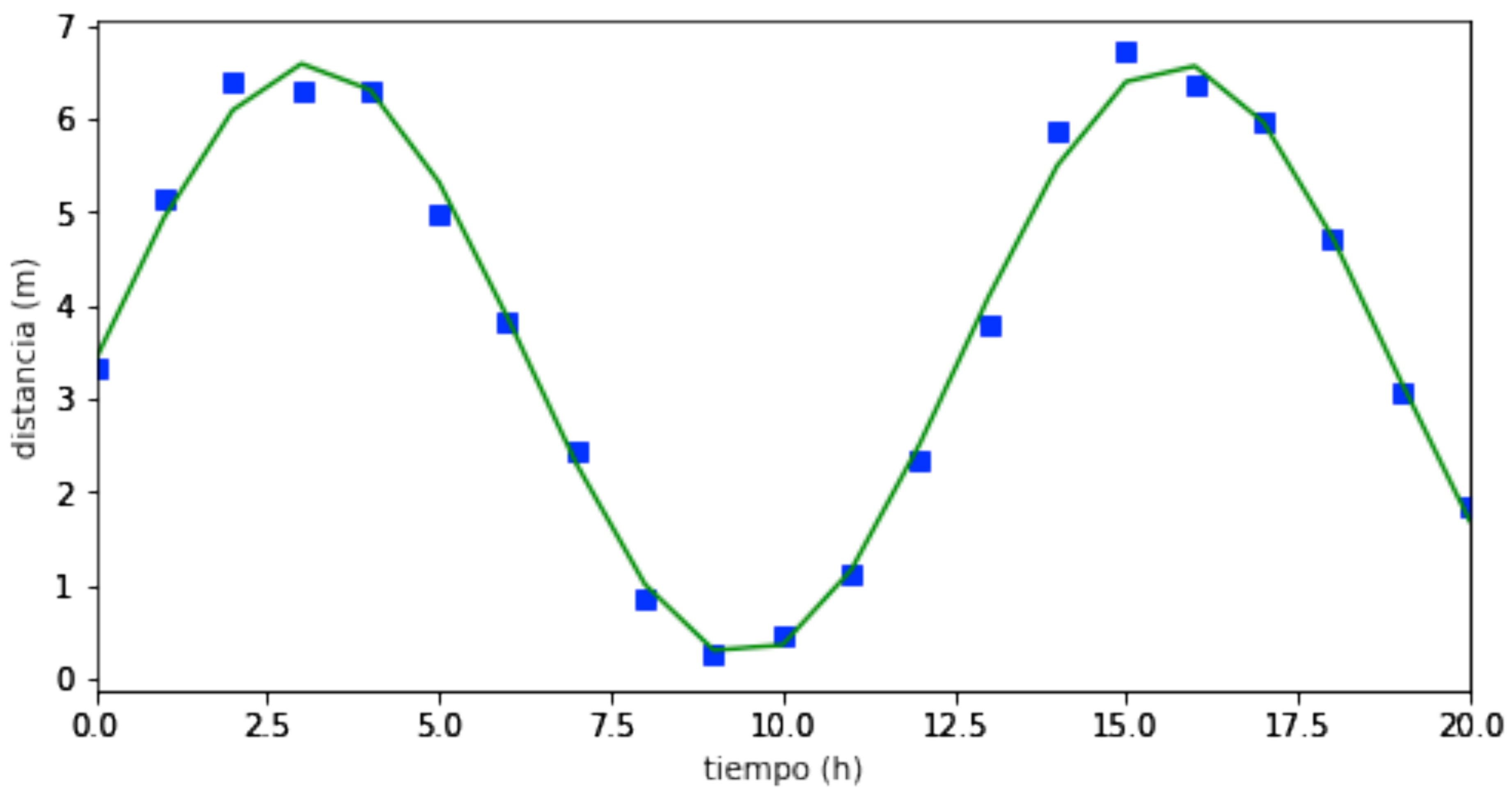
yo le he dicho la función que quiero probar y python nos da los parametros:

scipy.optimize.curve_fit(func, t, d)

$$d = 3.1\sin(0.5t) + 3.4$$

INTERPOLACIÓN Y EXTRAPOLACIÓN

Probamos otra función: $$d = a\sin(bt) + c$$



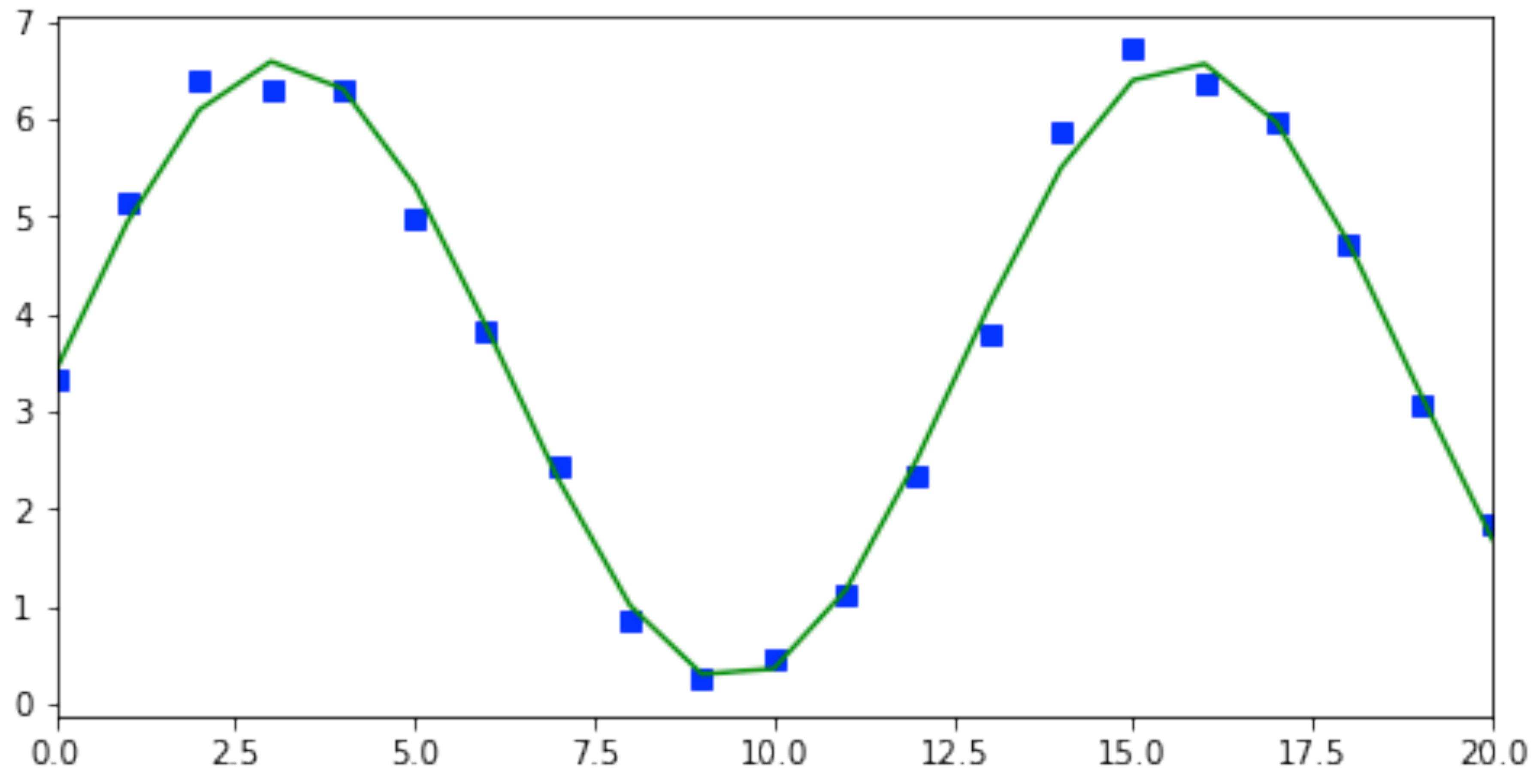
yo le he dicho la función que quiero probar y python nos da los parametros:

scipy.optimize.curve_fit(func, t, d)

$$d = 3.1\sin(0.5t) + 3.4$$

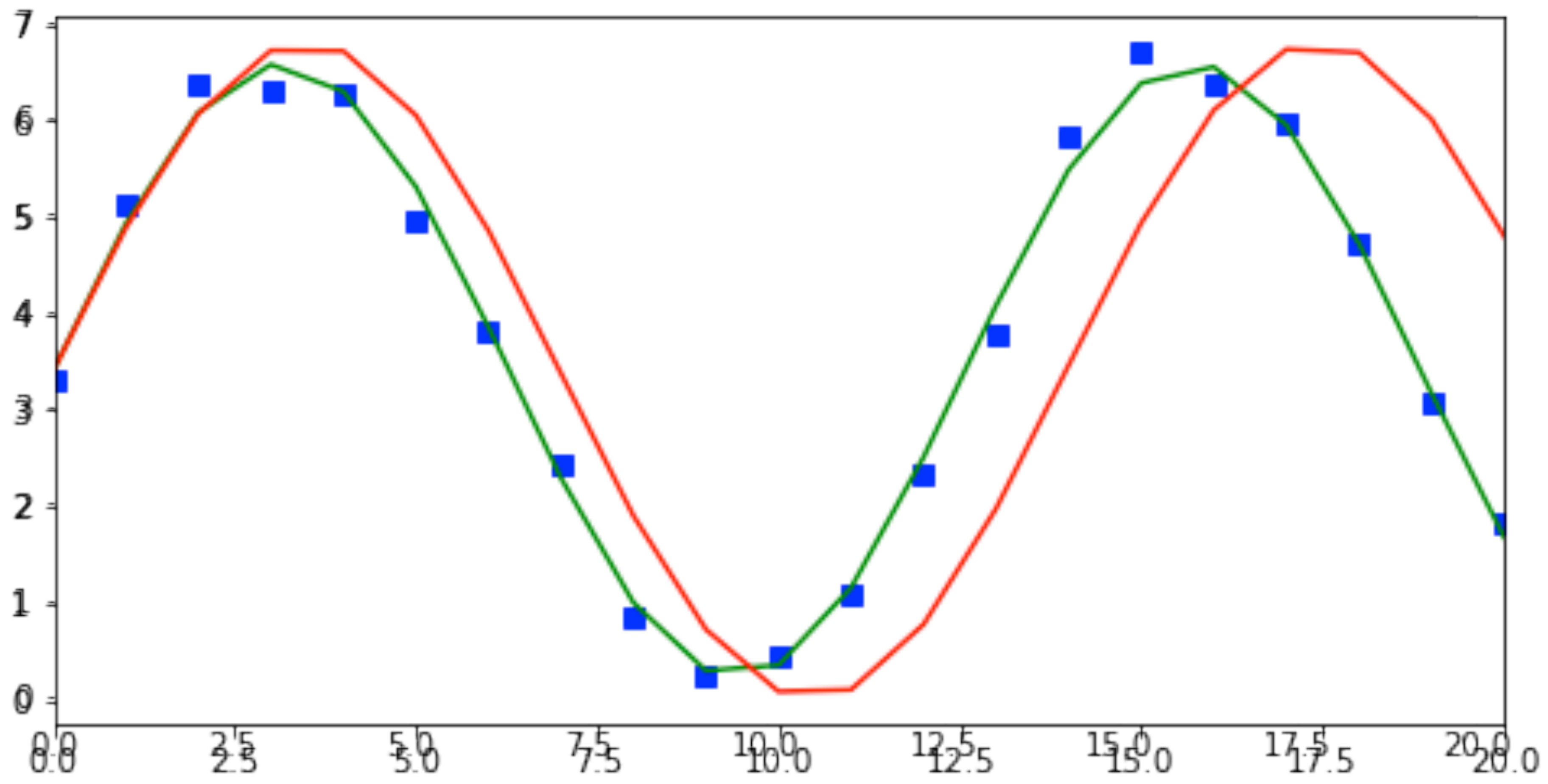
¡Este modelo parece que ajusta mejor a los datos!

INTERPOLACIÓN Y EXTRAPOLACIÓN



$$d = 3.1\sin(0.5t) + 3.4$$

INTERPOLACIÓN Y EXTRAPOLACIÓN

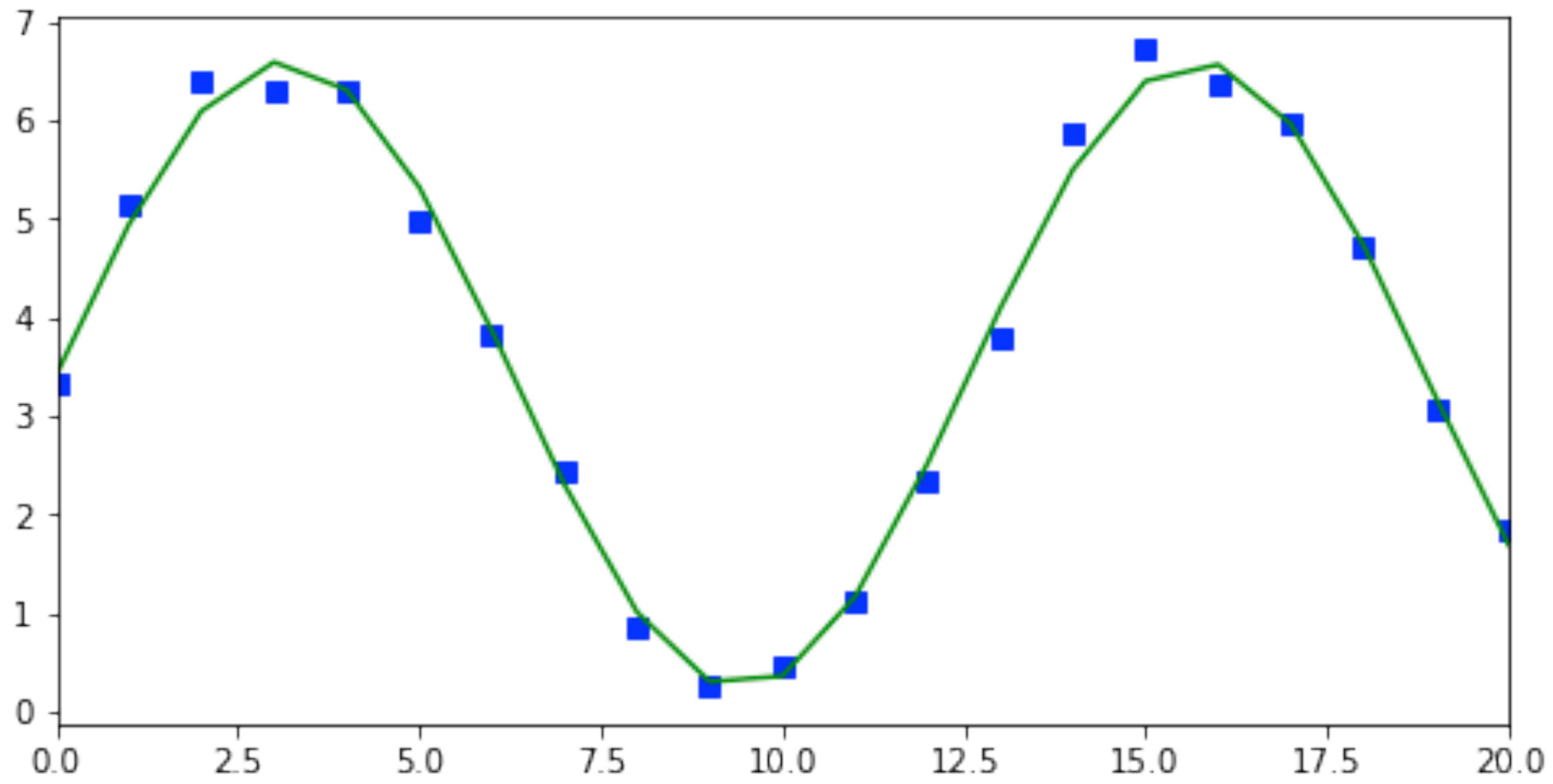


$$d = 3.1\sin(0.5t) + 3.4$$

$$d = 3\sin(0.3t) + 3$$

Modelo== Curva $\longrightarrow$ función + parametros. Cambiamos parámetros, cambiamos el modelo !!!

INTERPOLACIÓN Y EXTRAPOLACIÓN

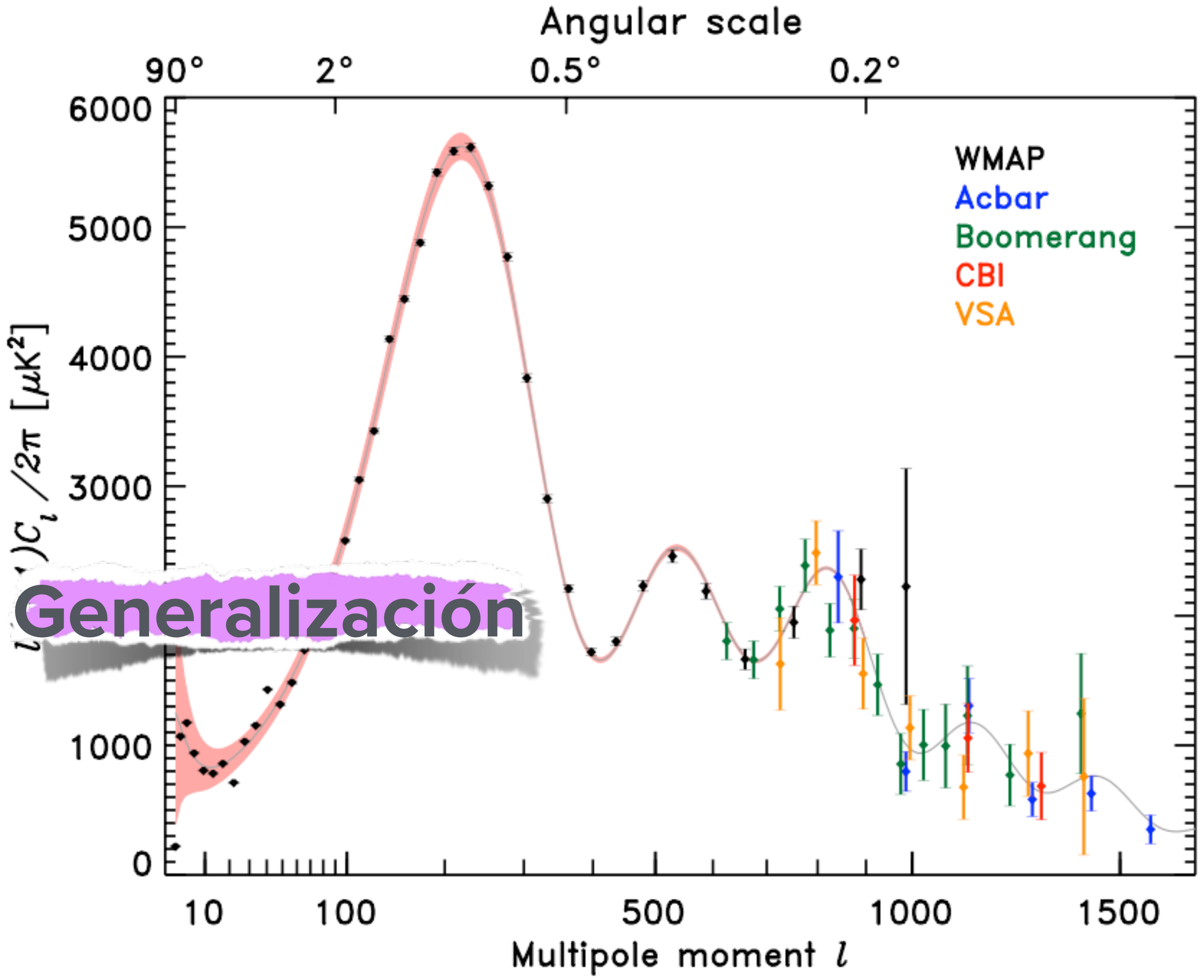


$$d = 3.1\sin(0.5t) + 3.4$$

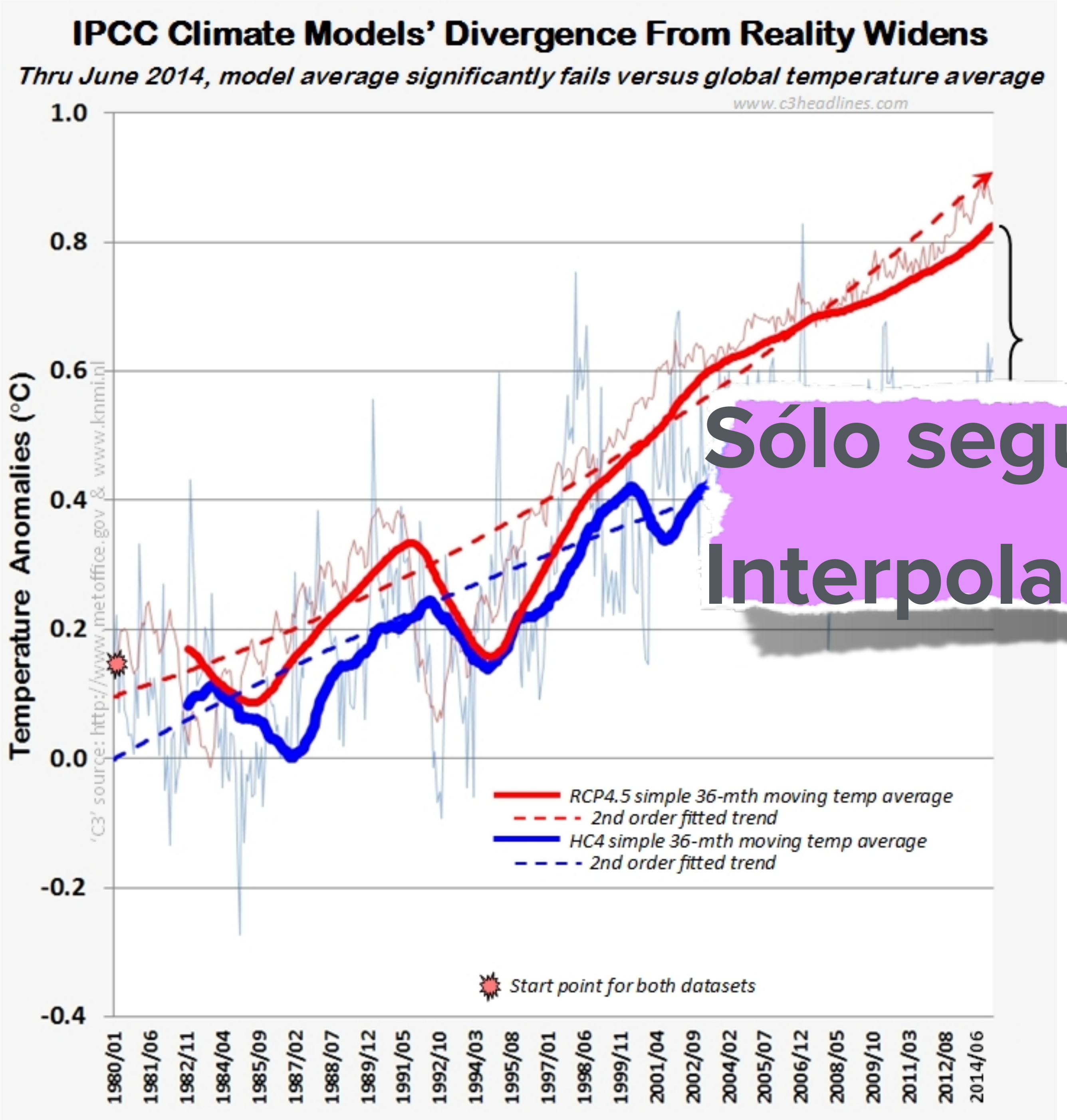
Pero si tomaremos el suficiente número de datos acabaría fallando también, necesitamos el **modelo teórico** para **generalizar**

MODELIZAR

Modelo téorico: Tenemos una relación matemática que puede estar dado por leyes conocidas o por hipotésis con una base teórica



Generalización



Sólo seguro en el rango de los datos.

Interpolación y extrapolación

intentamos ... que nos permita predecir resultados con datos nuevos.

Ej : Ajuste a un polinomio (o **redes neuronales!)**

¿Cómo obtenemos la curva?

MODELIZAR

Ajuste por mínimos cuadrados

- Minimizamos la distancia promedio entre los datos y la función matemática en el mismo punto

$$E = \sum_i \frac{(d - f(t))^2}{N}$$

En este caso f(t) $$f(t) = a\sin(bt) + c$$

MODELIZAR

Ajuste

Tiempo (h)	Distancia (m)
0	3
1	5
2	8
...	...

- Comparamos con una distancia el valor de los datos con el valor que me daría la función escogida:

$$f(t) = a\sin(bt) + c$$

- $\sum dist_i^2 = \sum ((d(t)_i - f(t)_i)^2 = (3 - asin(b0) + c)^2 + (5 - asin(b) + c)^2 + \ldots$

MODELIZAR

Función objetivo:

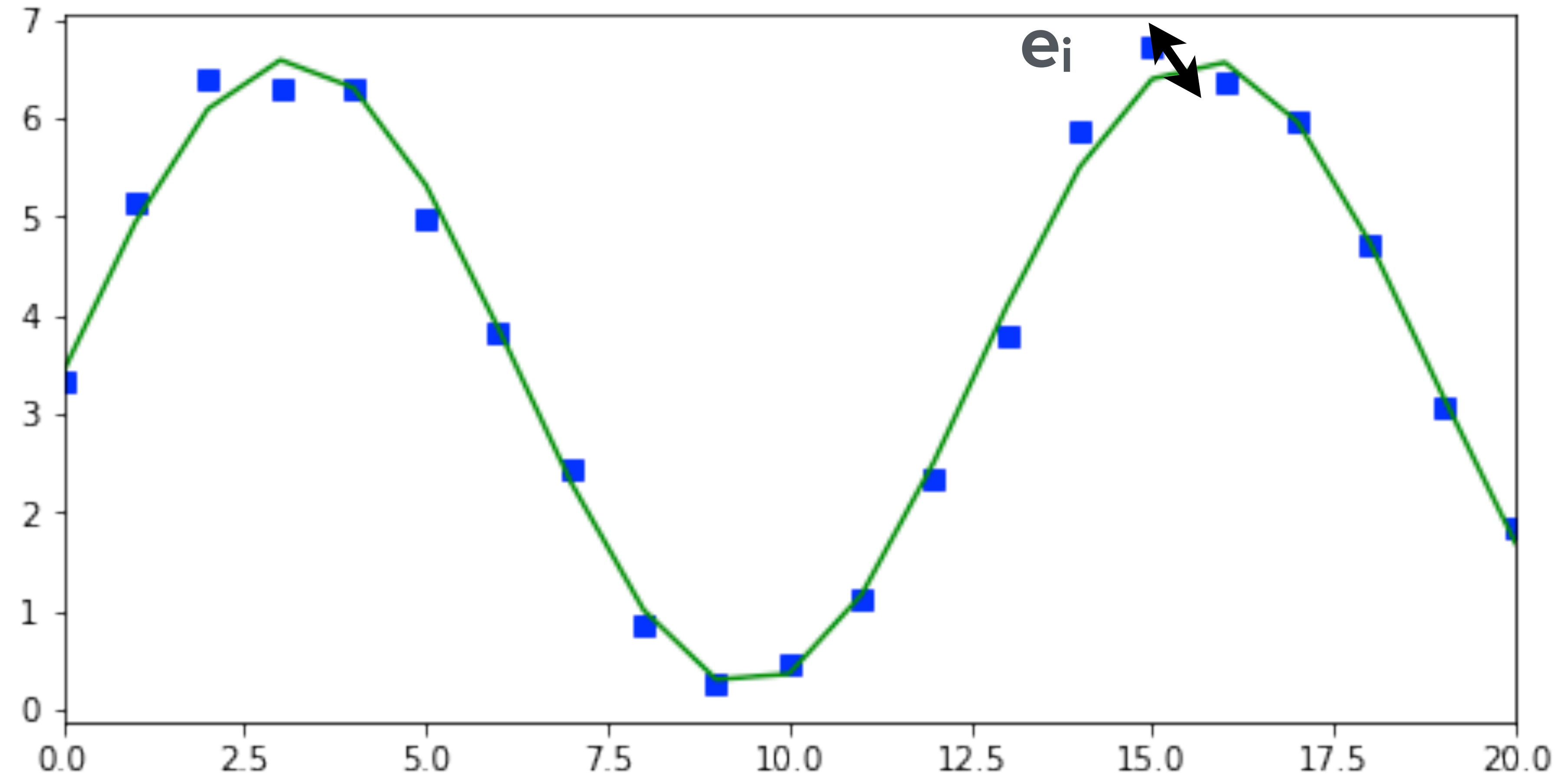
$$E = \sum_i \frac{(d_i - a\sin(bt_i) - c)^2}{N}$$

Variables:

$$a, b, c$$

* Buscamos la curva más cercana a los datos. La que minimiza la distancia promedio con los datos.

$$E = \sum_i e_i^2$$



* optimize.curve_fit() y numpy.polyfit()

les das f(t) y los datos y calculan E y buscan el mínimo

MODELIZAR

Función objetivo:

$$E = \sum_i \frac{(d_i - a\sin(bt_i) - c)^2}{N}$$

Variables:

$$a, b, c$$

```
# definimos el modelo en una función, con parámetros de entrada
# la variables y los parámetros libres
def func(x, a, b):
    return a * np.sin(b*x)

# hacemos ajuste por mínimos cuadrados usando curve_fit
popt, pcov = opt.curve_fit(func, x3, y3)#,p0=(0.5,0.5,0.5))
```

MODELIZAR

Función objetivo:

$$E = \sum_i \frac{(d_i - a\sin(bt_i) - c)^2}{N}$$

Variables:

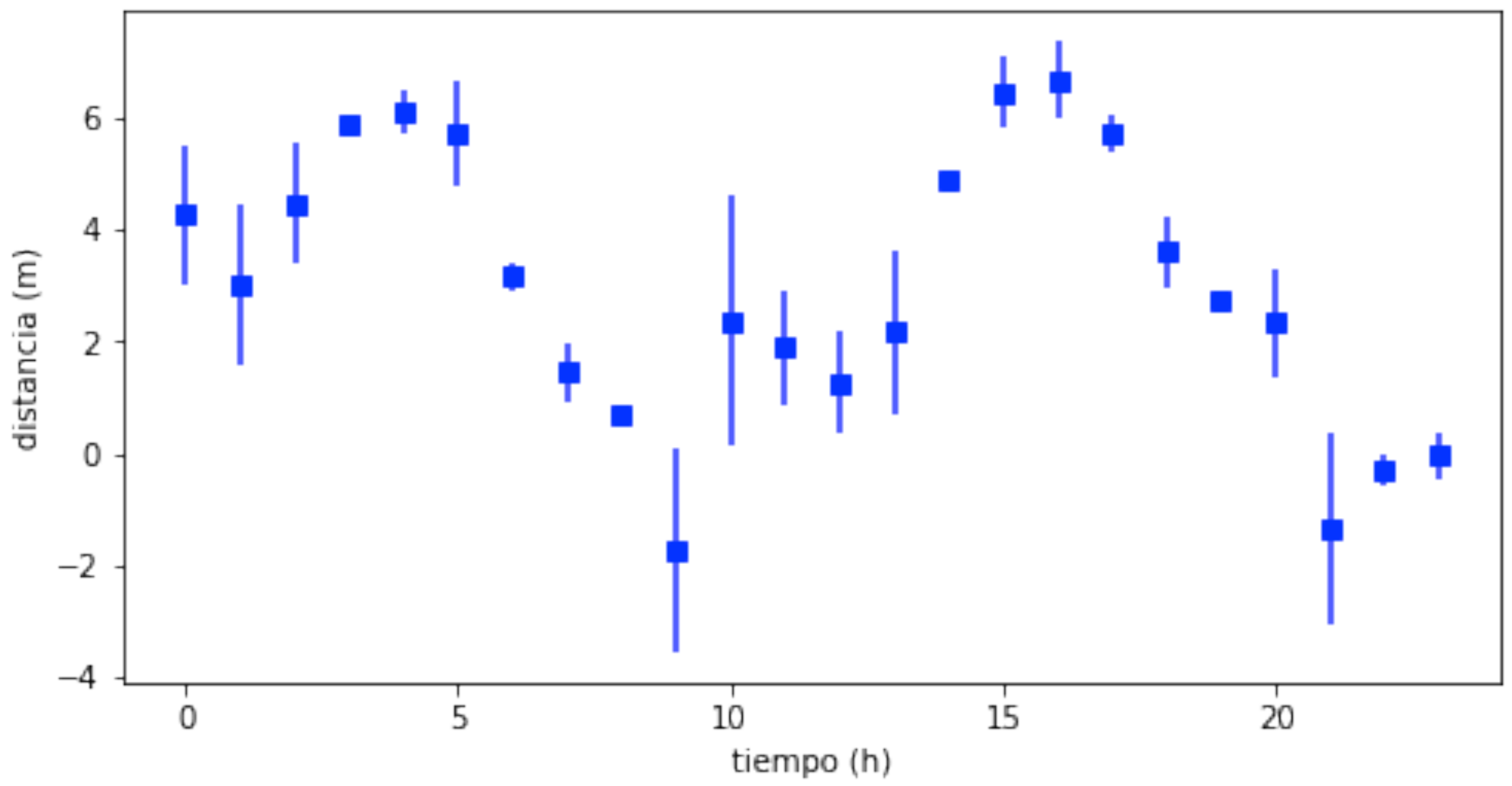
$$a, b, c$$

```python
# definimos el modelo en una función, con parámetros de entrada
# la variables y los parámetros libres
def func(x, a, b):
     return a * np.sin(b*x)

# hacemos ajuste por mínimos cuadrados usando curve_fit
popt, pcov = opt.curve_fit(func, x3, y3)#,p0=(0.5,0.5,0.5))
```

MODELIZAR

Ajuste. Mínimos cuadrados pesados

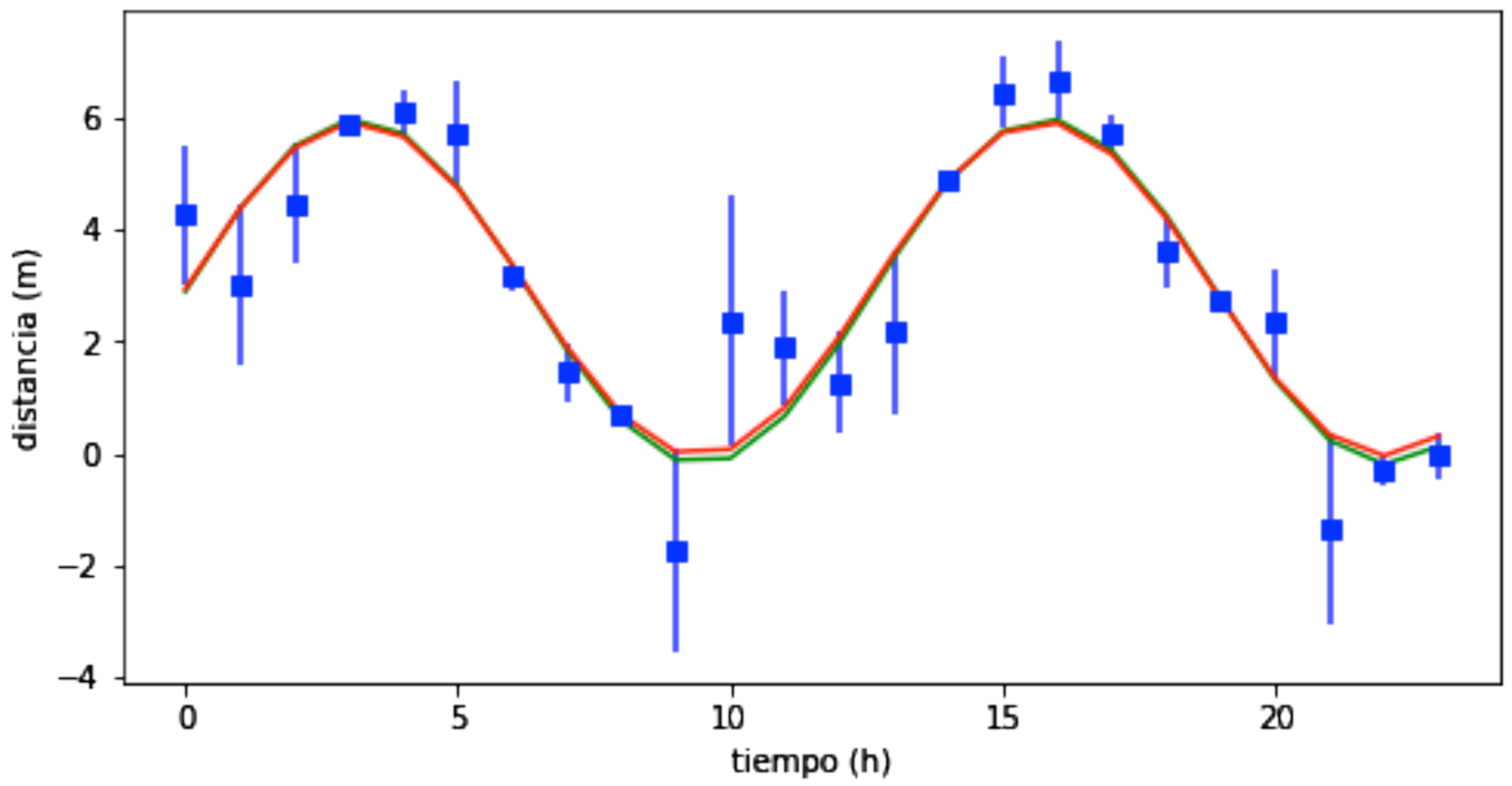


- Los datos suelen ser ruidosos, (errores al medir, ruido del aparato, incertidumbre, …)
- Weighted least squares —> si sabemos el error usamos la información del ruido

*

MODELIZAR

Ajuste. Mínimos cuadrados pesados



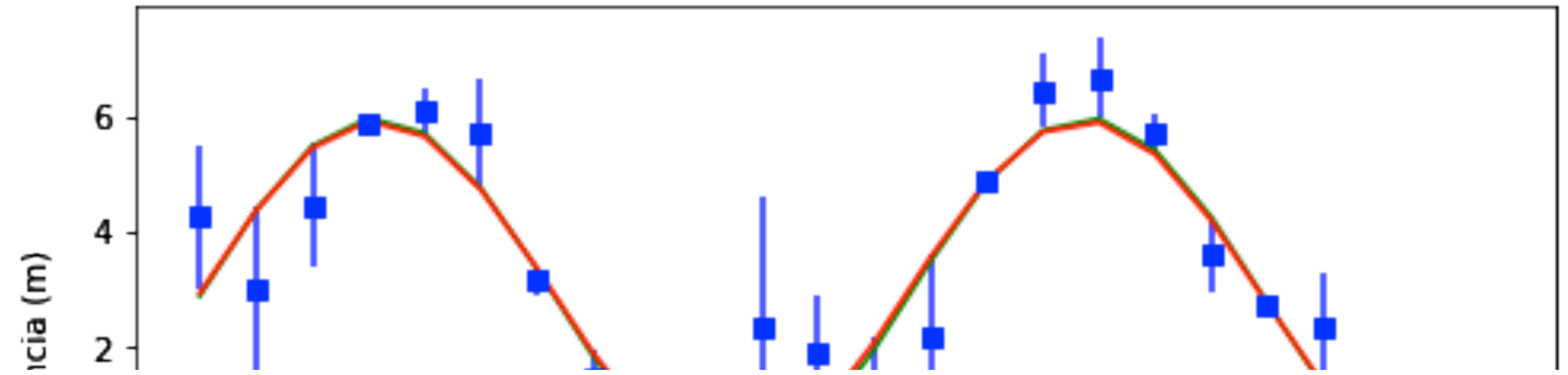
- Los datos suelen ser ruidosos, (errores al medir, ruido del aparato, incertidumbre, ...)
- Weighted least squares —> si sabemos el error usamos la información del ruido

* **Weighted least squares method**

$$\chi^2 = \sum_{i}^{N} \frac{(y_i - f(x_i))^2}{\sigma^2}$$

MODELIZAR

Ajuste. Mínimos cuadrados pesados




```
In [372]: Data2=np.loadtxt('Data2_modelling_noisy.txt')
x=Data2[:,0];y=Data2[:,1];yerr=Data2[:,2]
plt.figure(0,figsize=(8,4))


popt, pcov = opt.curve_fit(func, x, y,p0=(0.5,0.5,0.5))
poptErr, pcov = opt.curve_fit(func, x, y,p0=(0.5,0.5,0.5),sigma=yerr)
# esto nos devolverá los parametros a,b y c

plt.errorbar(x, y, yerr=yerr,  marker='s',color='b',label='data',linestyle='')
#popt, pcov = opt.curve_fit(func, x, y,method='lm')
a=popt[0];b=popt[1];c=popt[2]
#model=func(x,3,0.5,3)
model=func(x,a,b,c)
modelErr=func(x,*poptErr)
plt.plot(x,model,'-',color='g')
plt.plot(x,modelErr,'-',color='r')

plt.xlabel('tiempo (h)')
plt.ylabel('distancia (m)')
```

```
In [376]: print 'results     ',np.round(popt,2)
print 'results Err',np.round(poptErr,2)
print 'True val    ',[3.0,0.5,3.0]

results     [3.09 0.5  2.89]
results Err [2.99 0.5  2.94]
True val    [3.0, 0.5, 3.0]
```

error

least squares

$$\frac{(y_i - f(x_i))^2}{\sigma^2}$$

MODELIZAR

- Si la distribución de las variables es Gaussiana, los mínimos cuadrados siguen una distribución χ^2

- χ^2 tiene una función de densidad de probabilidad conocida que depende solo del número de **grados de libertad** del problema.

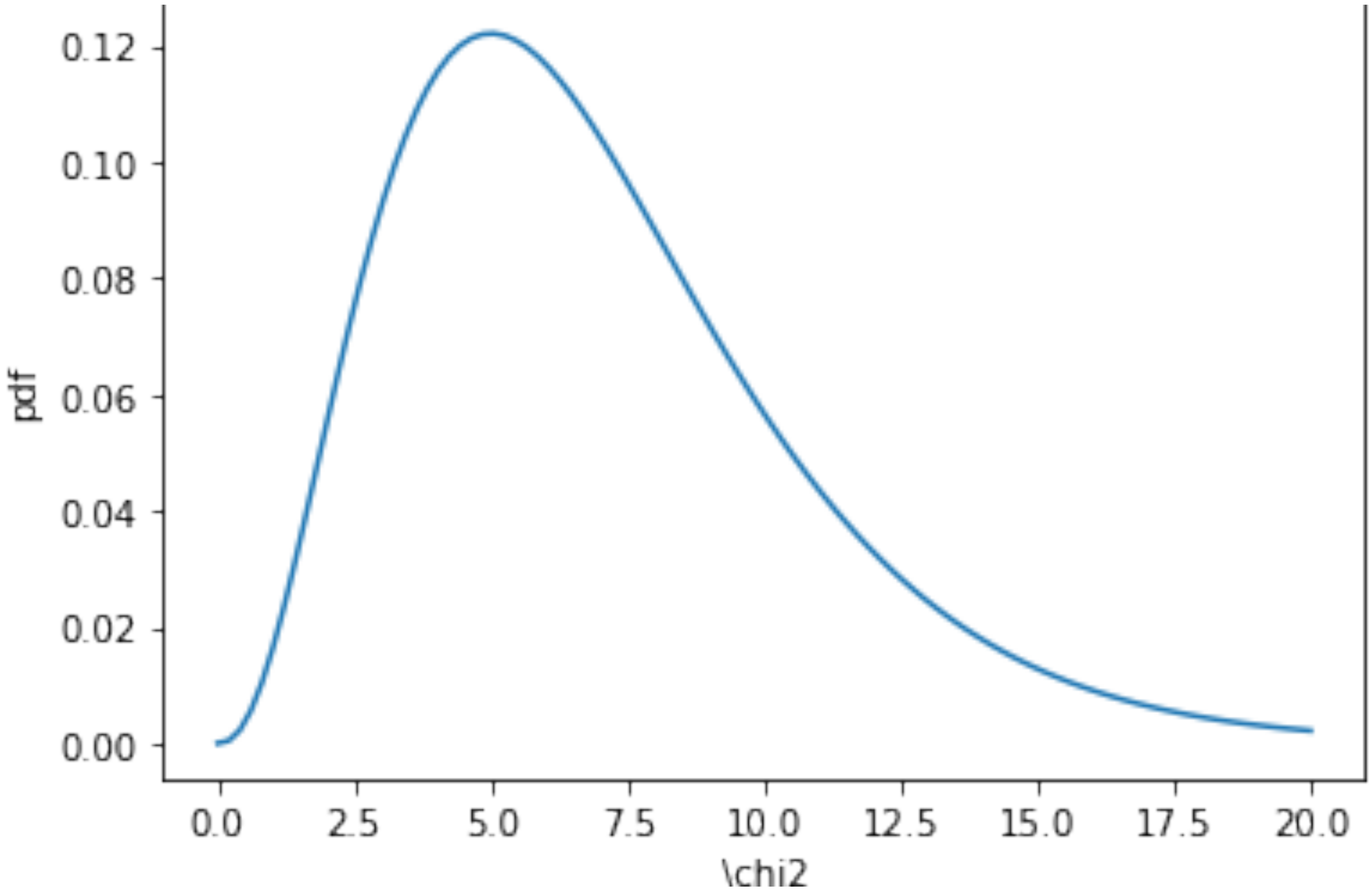
- Los grados de libertad son el número de muestras de la variable aleatoria.

- Perdemos un grado de libertad por cada parámetro del modelo.

- Dof = N - params

MODELIZAR

Distribuciones chi - square

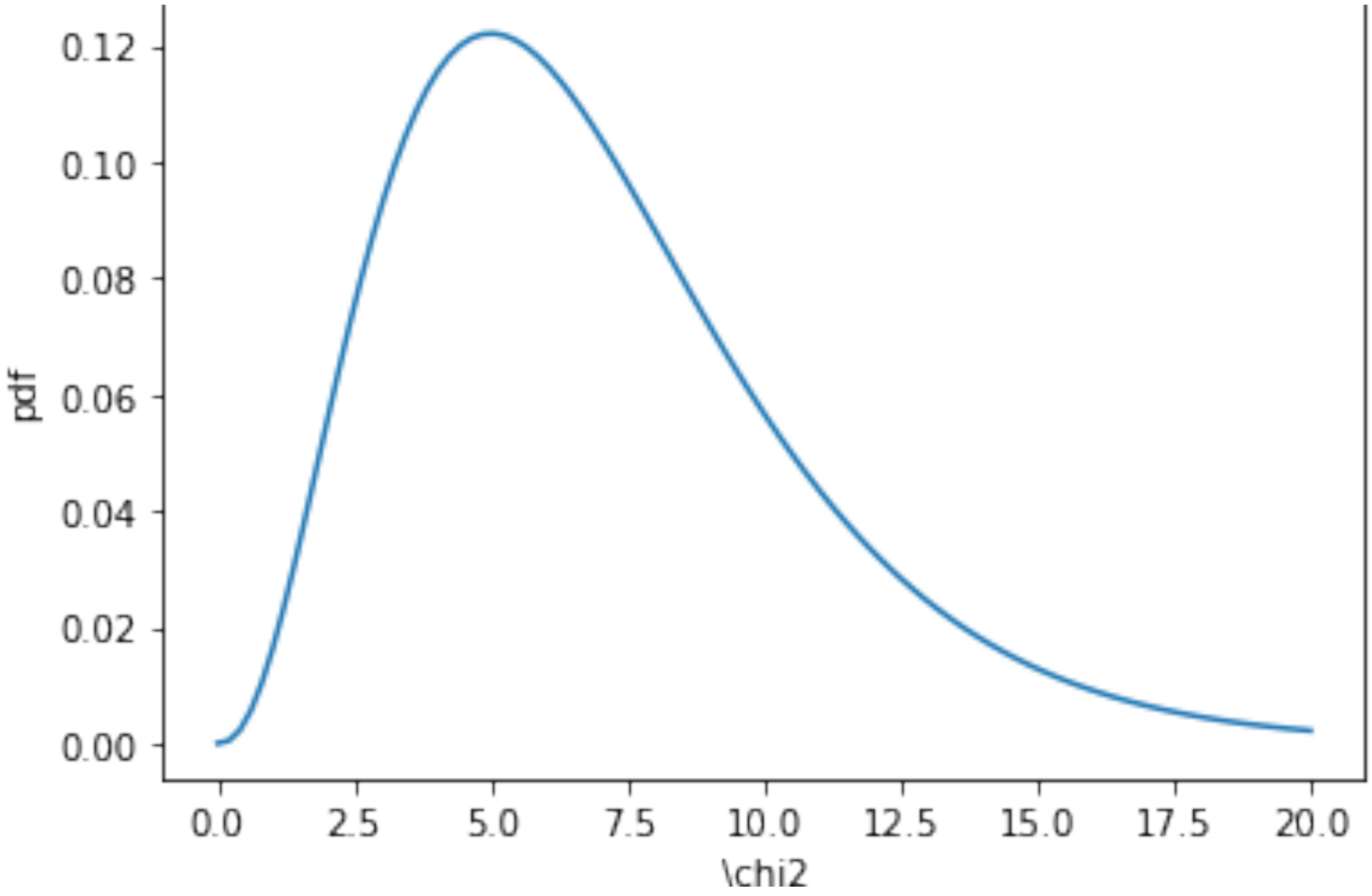


χ^2

Con 7 grados de libertad

MODELIZAR

Distribuciones chi - square

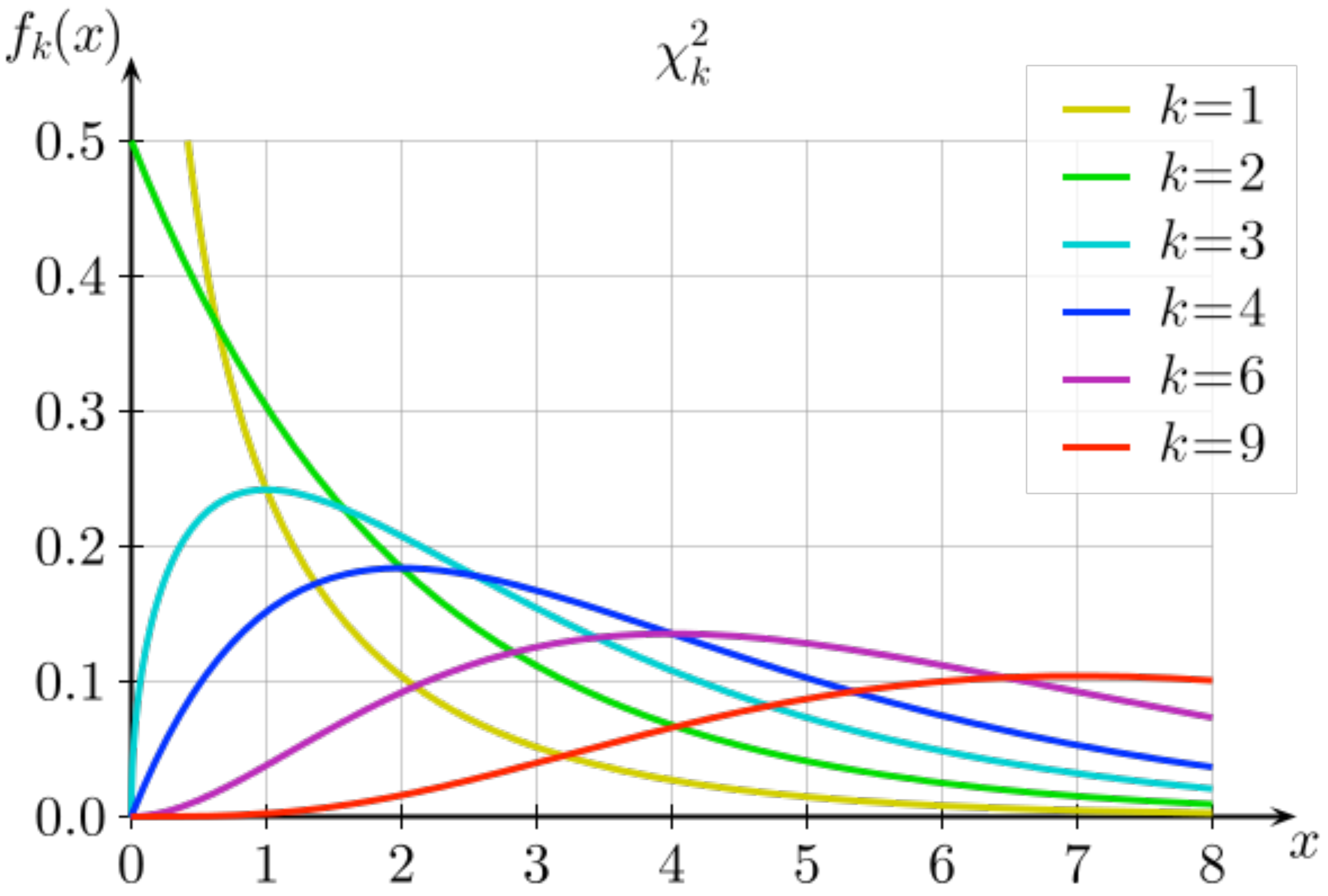


χ^2

Con 7 grados de libertad

MODELIZAR

Distribuciones chi - square

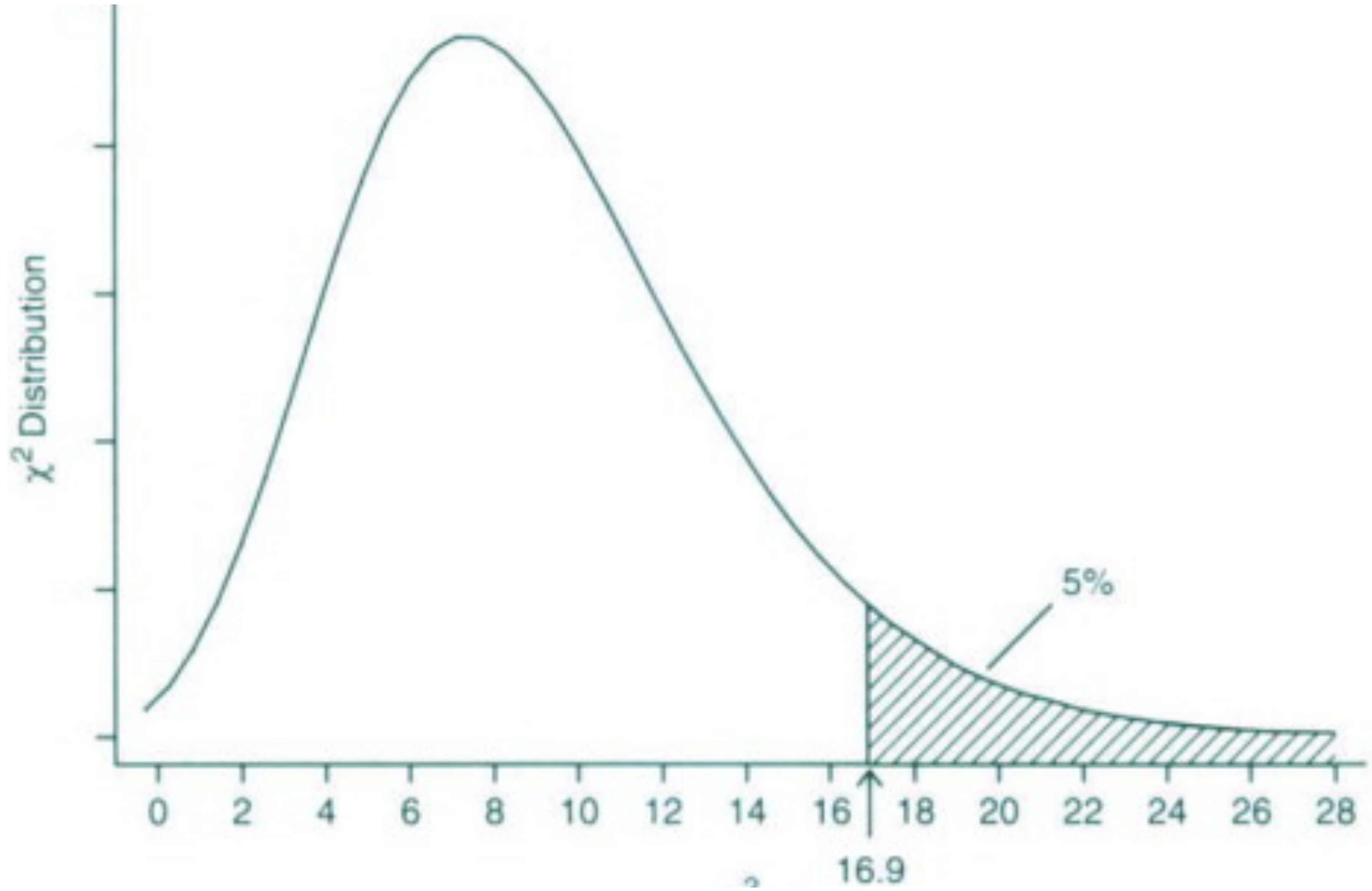


$$< \chi^2 > = k$$

$$\sigma(\chi^2) = \sqrt{2k}$$

MODELIZAR

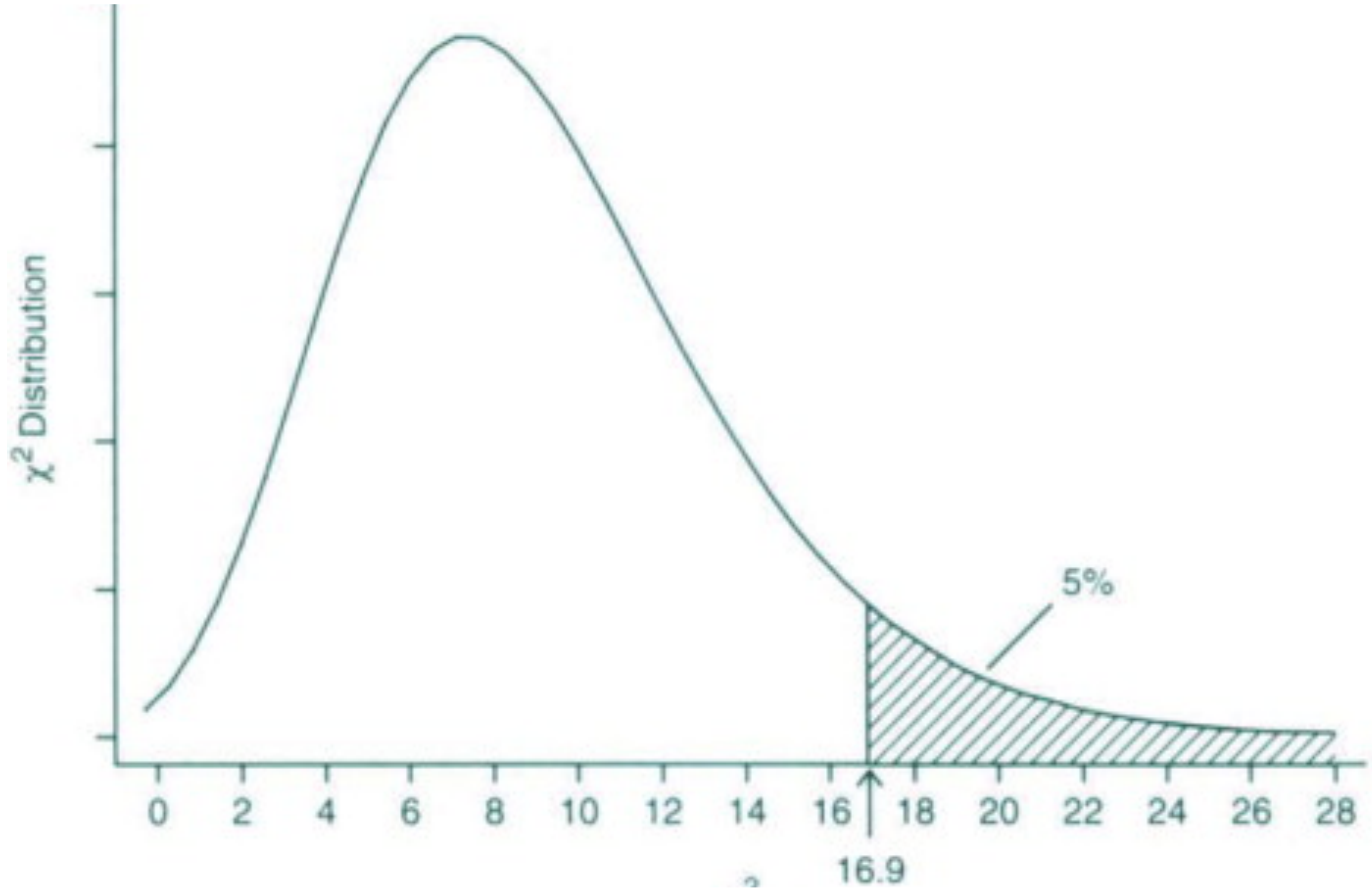
- Como conocemos la pdf y hemos calculado un χ^2 mínimo, podemos calcular cuál sería la probabilidad de tener un χ^2 mayor, tal como vimos en la notebook de distribuciones.



Función densidad de probabilidad de chi-square

MODELIZAR

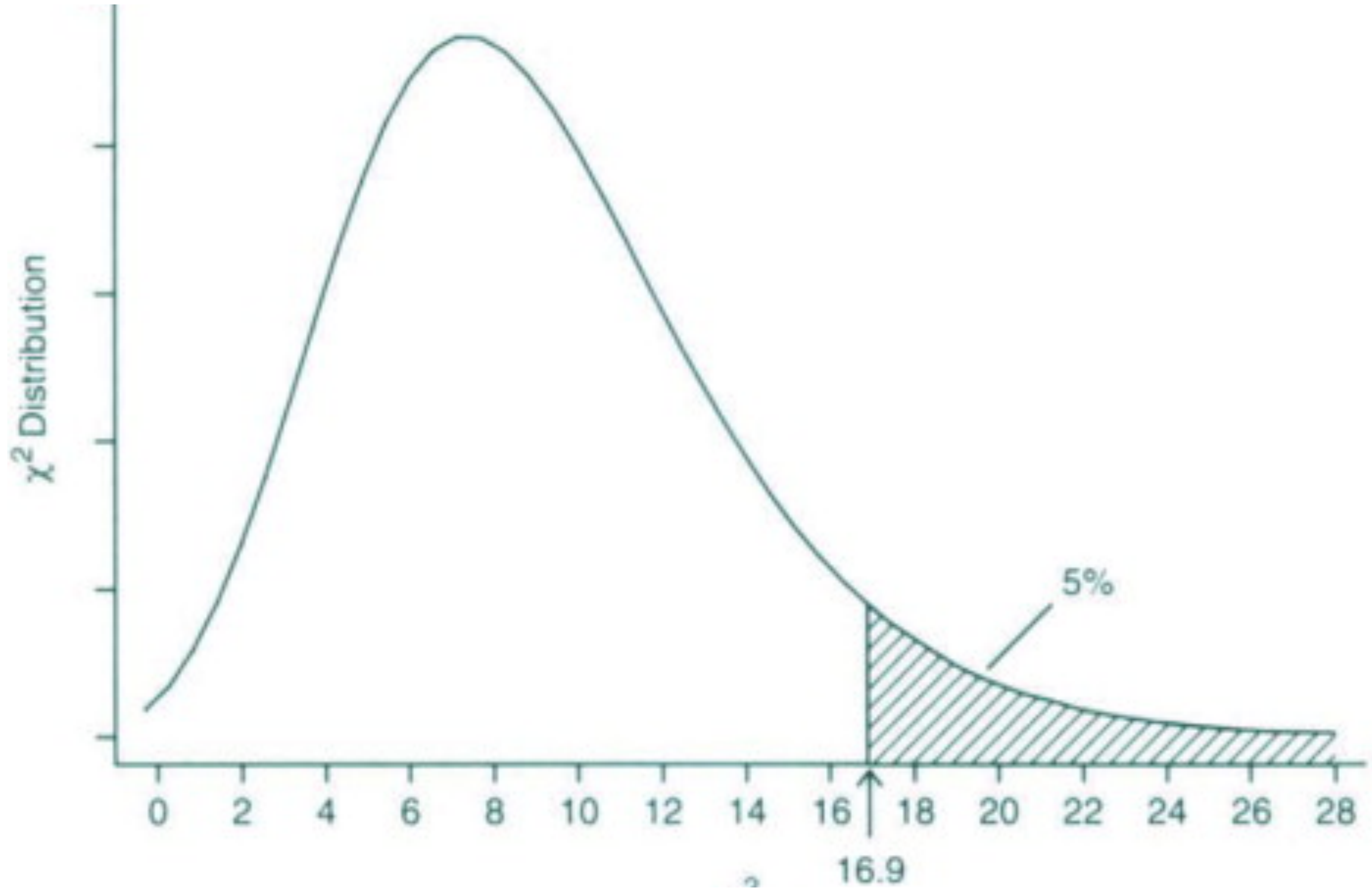
- **C**omo conocemos la pdf y hemos calculado un χ^2 mínimo, podemos calcular cuál sería la probabilidad de tener un χ^2 mayor, tal como vimos en la notebook de distribuciones.



Función densidad de probabilidad de chi-square

MODELIZAR

- Como conocemos la pdf y hemos calculado un χ^2 mínimo, podemos calcular cuál sería la probabilidad de tener un χ^2 mayor, tal como vimos en la notebook de distribuciones.



Función densidad de probabilidad de chi-square

CUANTO MENOR SEA ESA PROBABILIDAD MENOS VEROSÍMIL ES QUE NUESTROS DATOS SIGAN EL MODELO. SI $P(X>\chi^2) > 0.05$ SE DESCARTA EL FIT

MODELIZAR

Como sabemos si un ajuste está bien hecho?

Goodness of fit. **Hay varios tests que nos pueden decir cuan bueno es un ajuste. Dependiendo del método o estadísitco que estemos usando. Ejemplo:**

R-squared

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

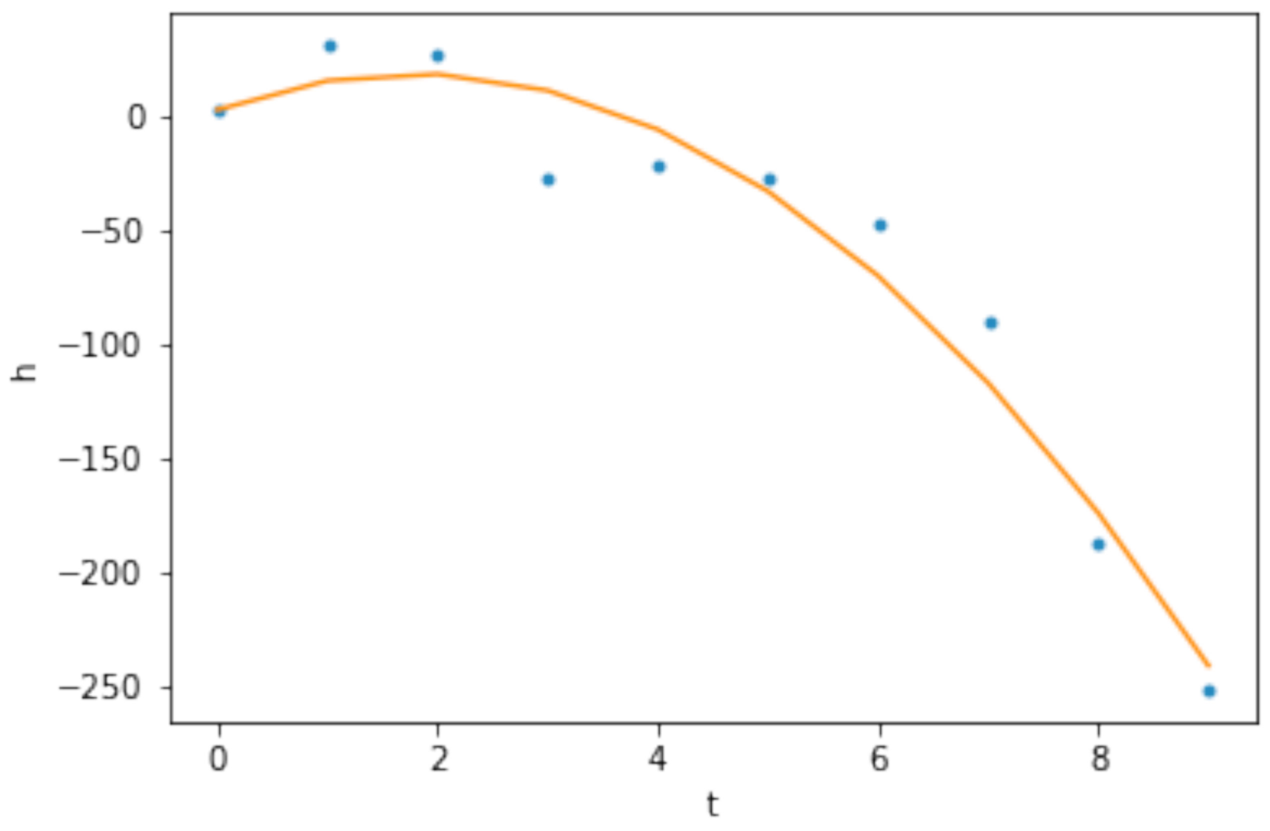
$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

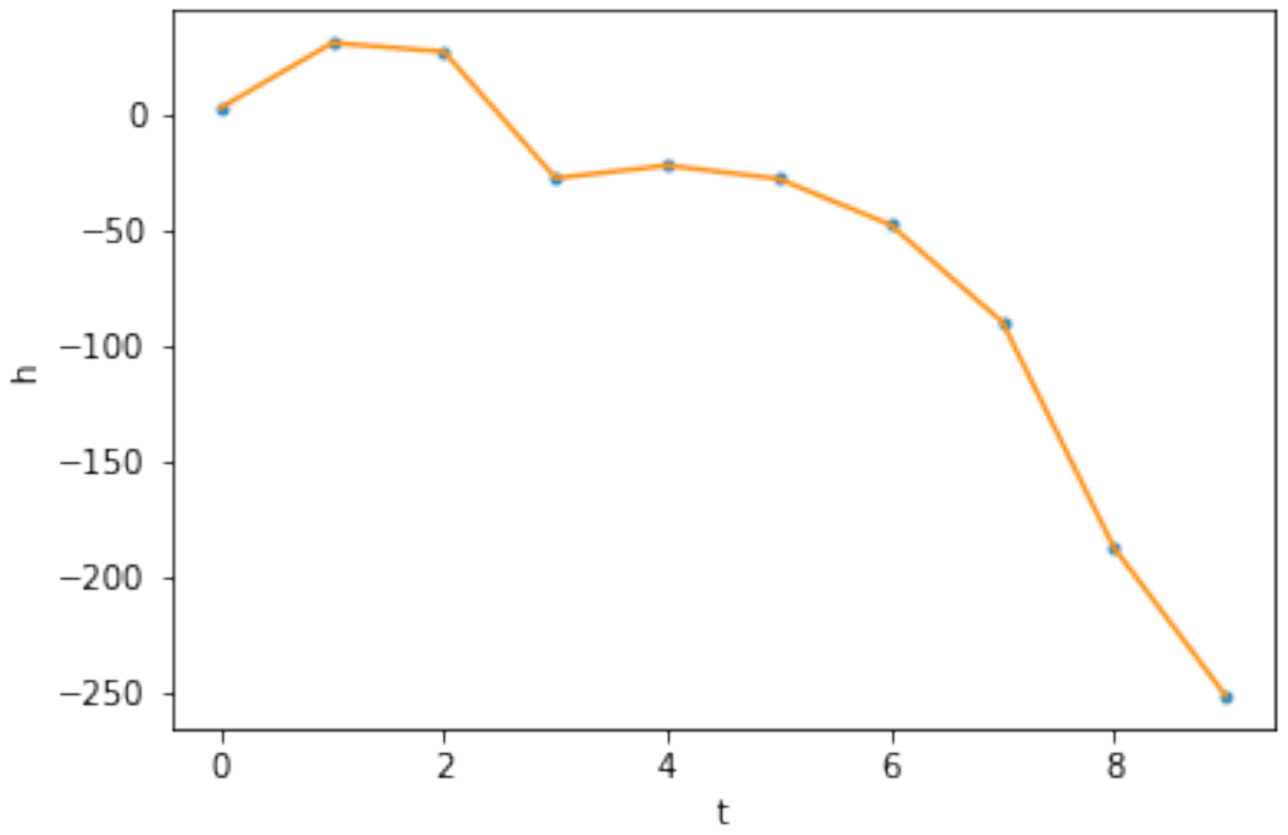
$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$

R-squared puede ir entre 0 y 1 . (Los errores en la estimación deberían ser menores a la varianza de la muestra)

MODELIZAR
SOBREAJUSTE

PERO ajustar bien a unos datos no implica que el modelo sea el correcto. El modelo tiene que estar bien motivado!!





Si el modelo no es del todo conocido es necesario tener muchos datos para hacer el ajuste con una muestra y luego probarlo en otra. Si tenemos muy buen fit en una y en la validación falla es que hemos sobreajustado (más en *machine learning*)

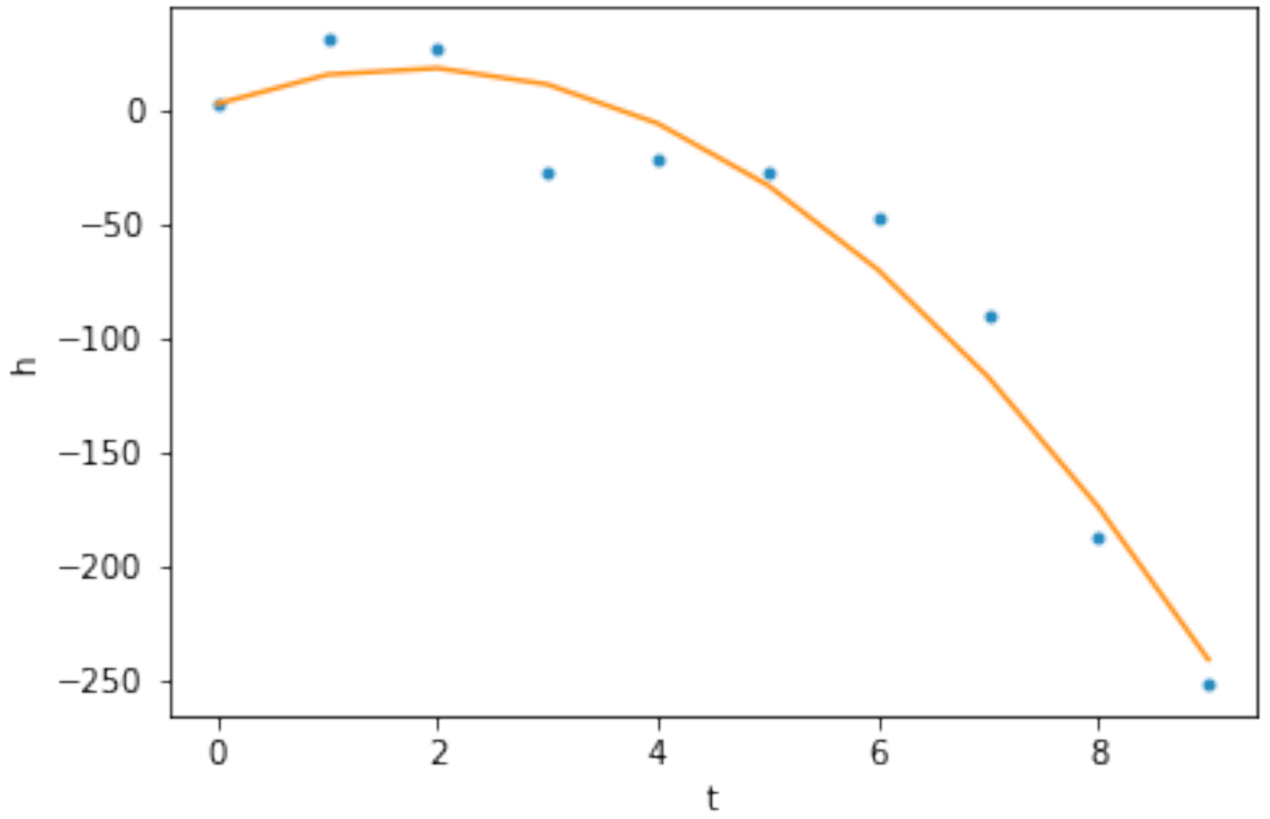
El ajuste de la derecha nos dará mejor R-squared y mejor chi-squared

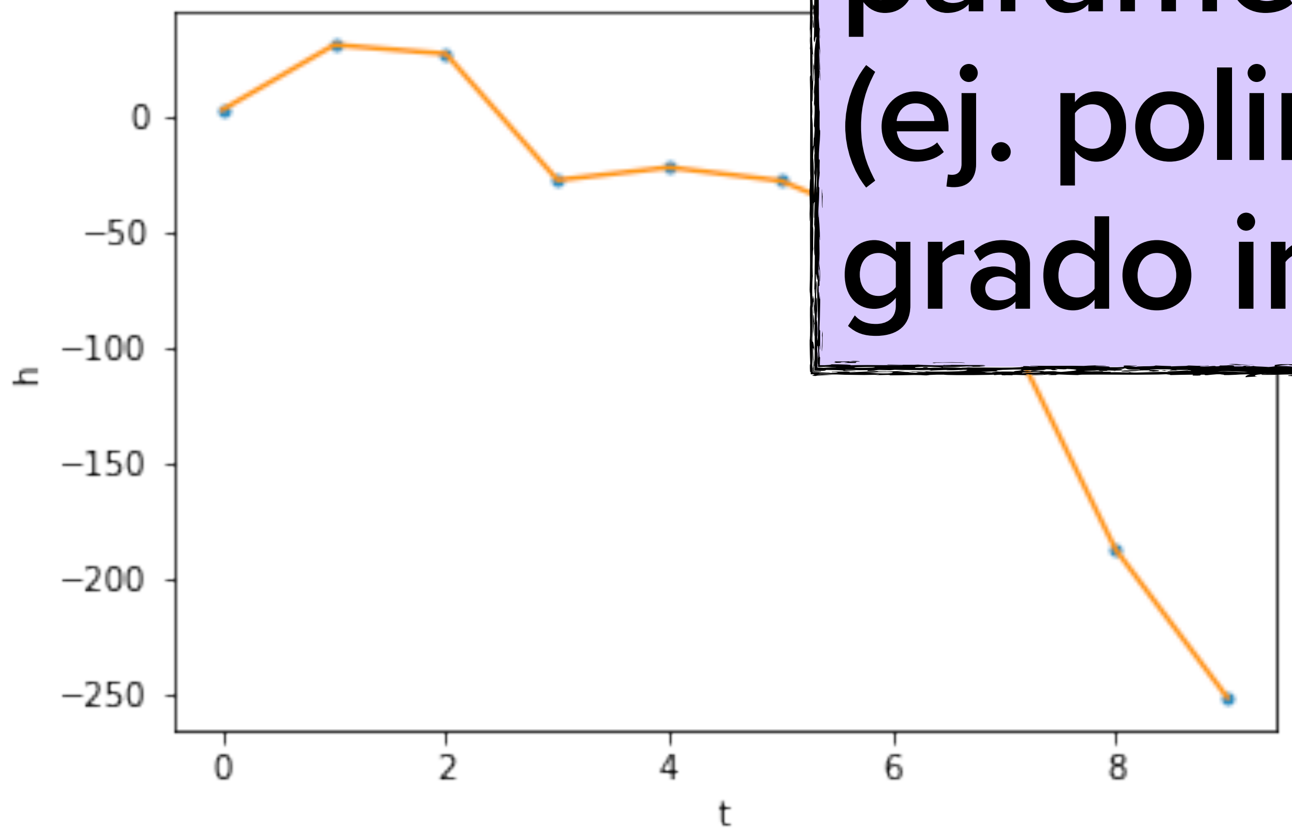
MODELIZAR

SOBREAJUSTE

PERO ajustar bien a unos datos no implica que el modelo sea el correcto. El modelo tiene que estar bien motivado!!

Modelo con menos parámetros preferido (ej. polinomio de grado inferior)



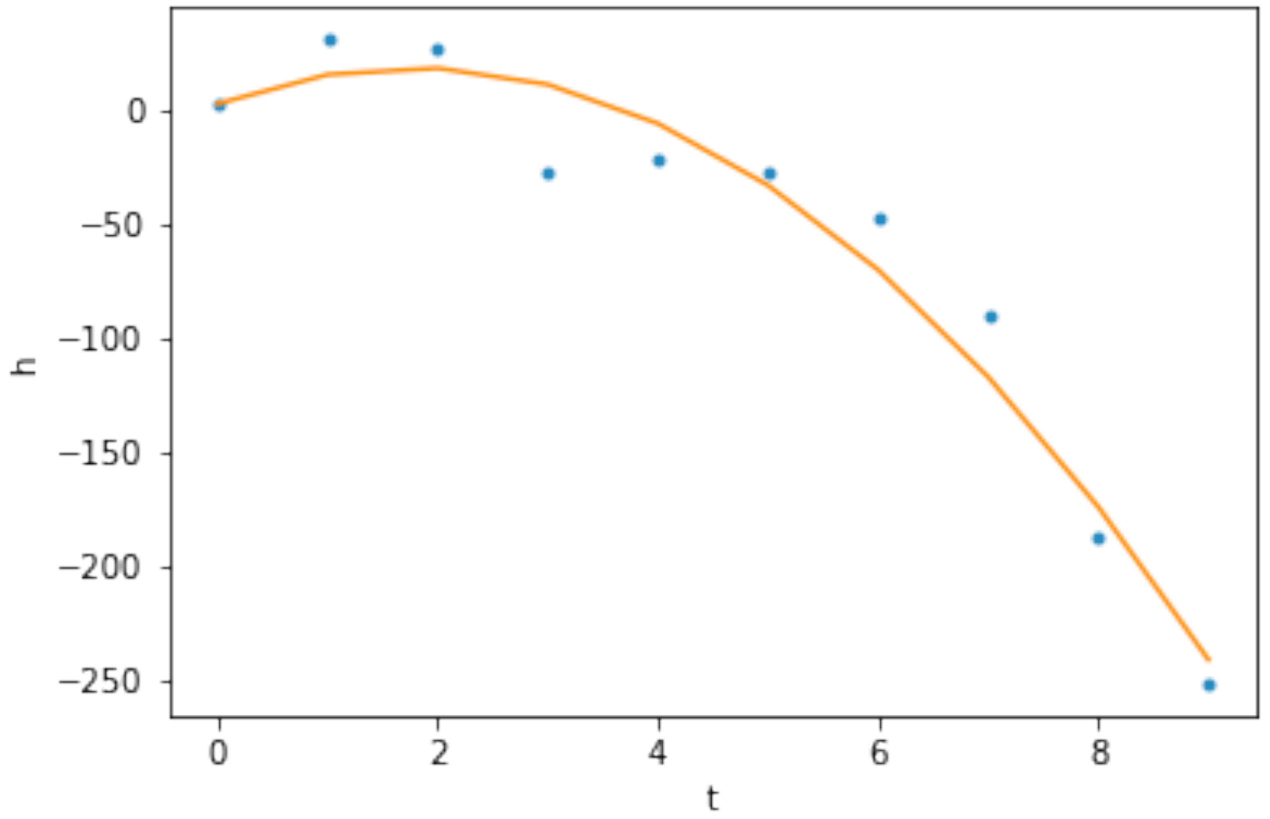


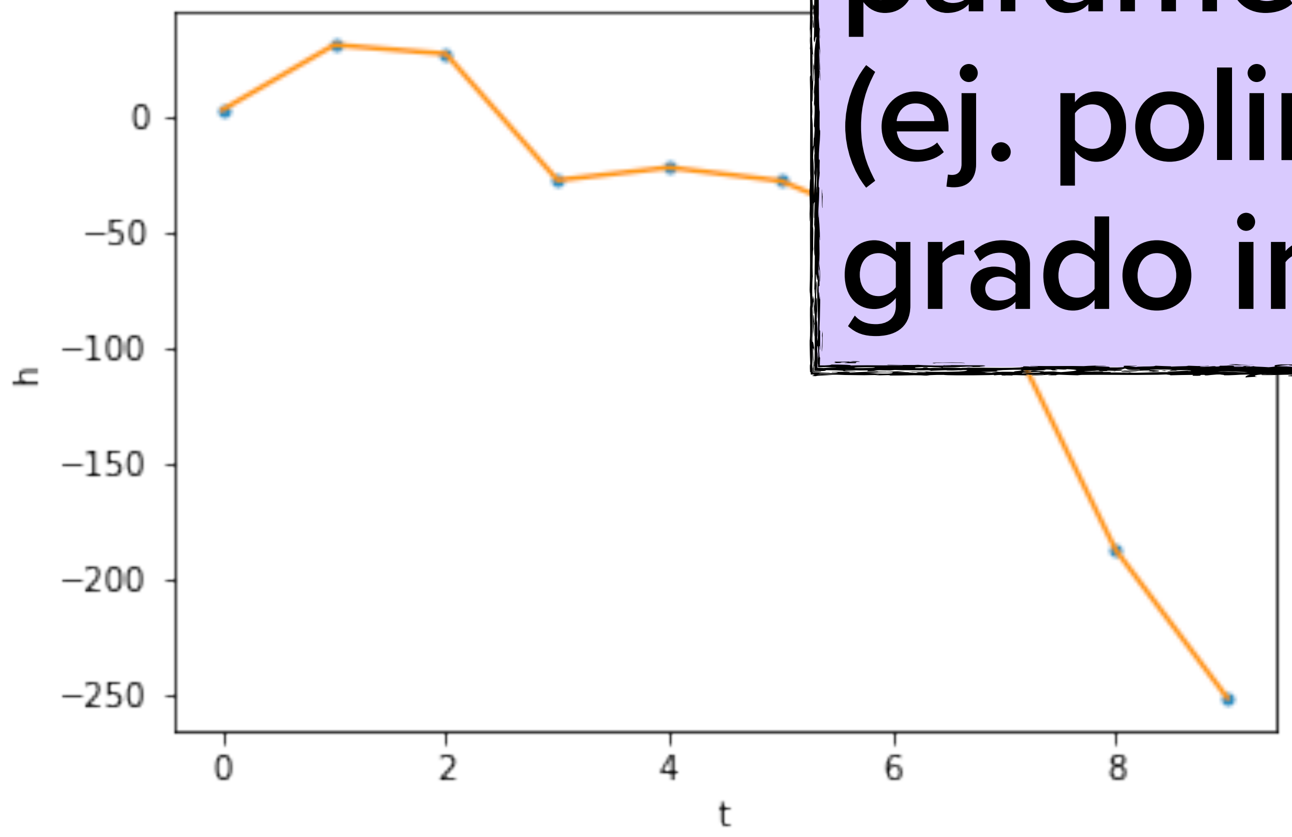
Si el modelo no es del todo conocido es necesario tener muchos datos para hacer el ajuste con una muestra y luego probarlo en otra. Si tenemos muy buen fit en una y en la validación falla es que hemos sobreajustado (más en *machine learning*)

El ajuste de la derecha nos dará mejor R-squared y mejor chi-squared

MODELIZAR
SOBREAJUSTE

PERO ajustar bien a unos datos no implica que el modelo sea el correcto. El modelo tiene que estar bien motivado!!





Modelo con menos parámetros preferido (ej. polinomio de grado inferior)

$$\chi^2_{red} = \frac{\chi^2}{dof}$$

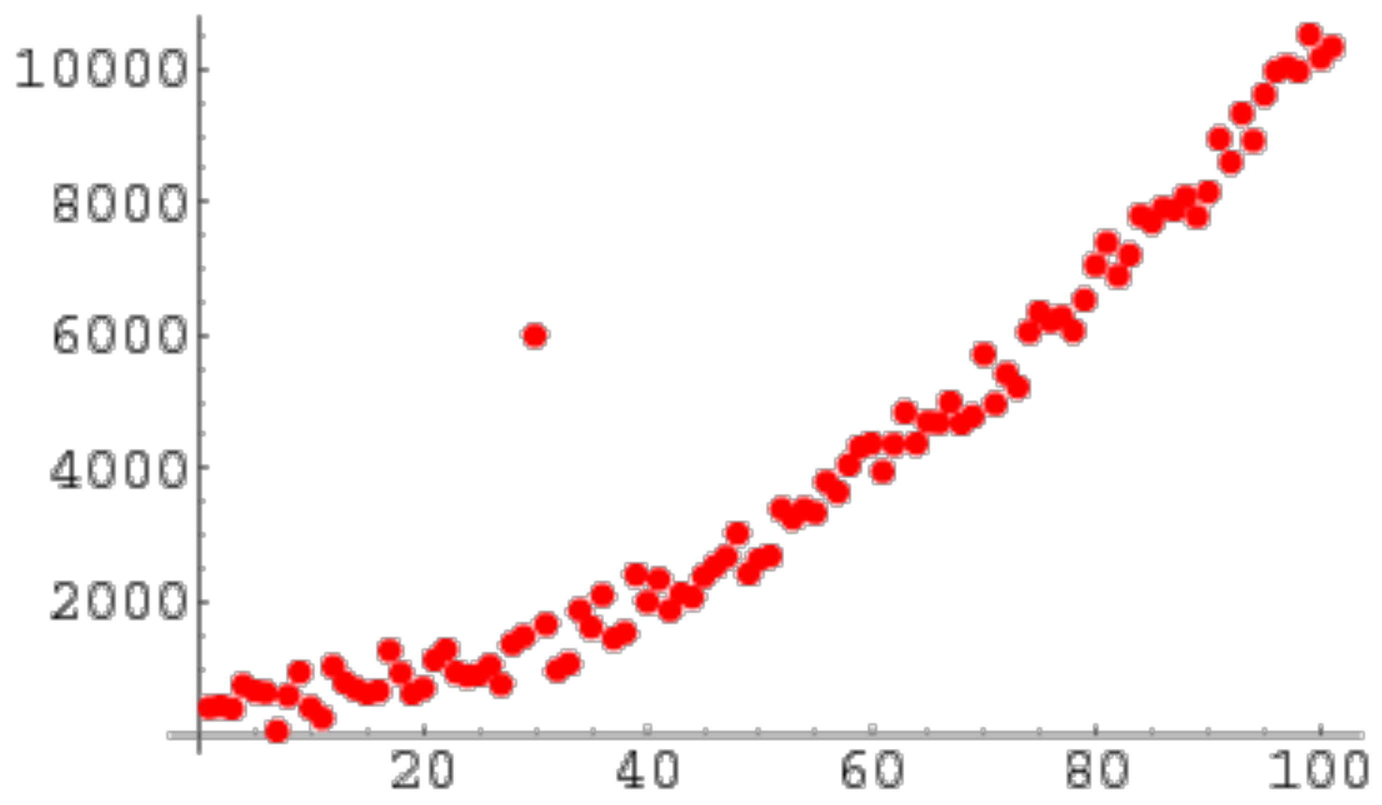
Si el modelo no es del todo conocido es necesario tener muchos datos para hacer el ajuste con una muestra y luego probarlo en otra. Si tenemos muy buen fit en una y en la validación falla es que hemos sobreajustado (más en *machine learning*)

El ajuste de la derecha nos dará mejor R-squared y mejor chi-squared

MODELIZAR

OUTLIERS

Vigilar con los Outliers



¡Mirar los datos antes de nada!

Valores muy extremos pueden pesar mucho en un ajuste. Hay que mirar los datos primero y descartar valores extremos, o al menos tenerlo en cuenta.

AJUSTES CON PYTHON
(VEMOS EJEMPLO CON NOTEBOOK)

SciPy.org

interp1d(x,y) interpolación

numpy.polyfit(x,y,d) ajuste con polinomio de grado d mediante minimos cuadrados

curve_fit(func,x,y) ajuste con función (func) mediante minimos cuadrados

minimize(ObjFunc,..) ajuste mediante cualquier función
