Como hacer un ajuste en Python (y no morir en el intento)

Llegó el ajuste al taller de Python de la FIFA

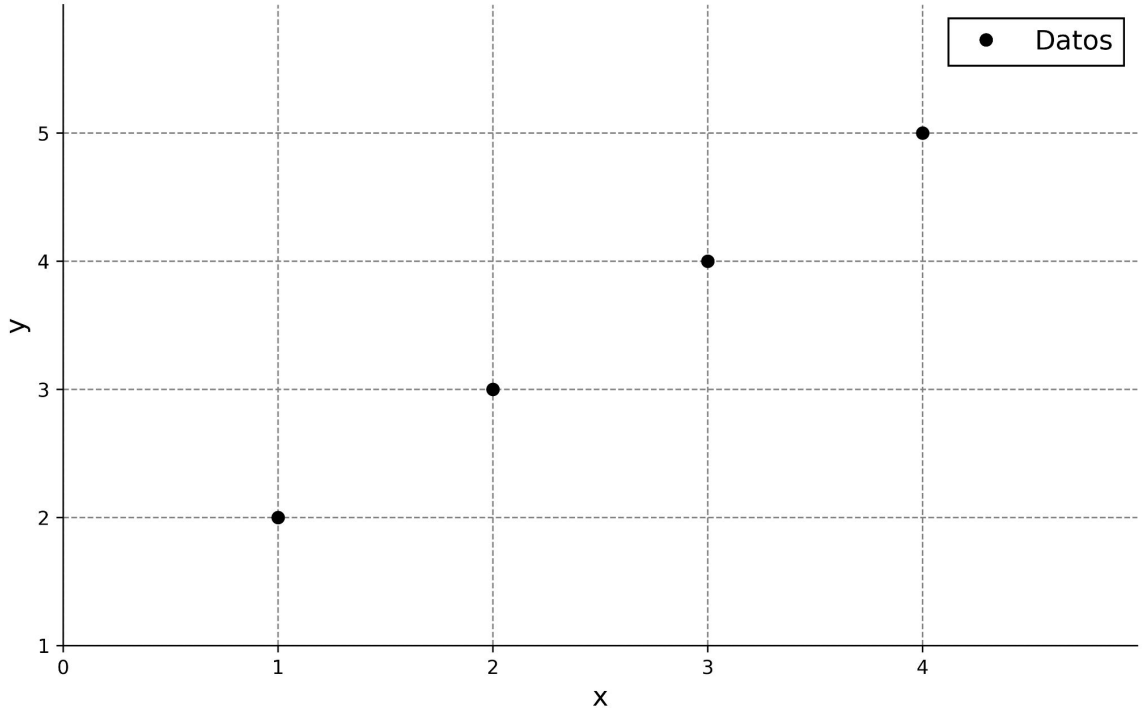
El problema a resolver

El problema a resolver

```
x = [1, 2, 3, 4]

y = [2, 4, 5, 6]

def lineal(x, a, b):
    return a*x+b
```

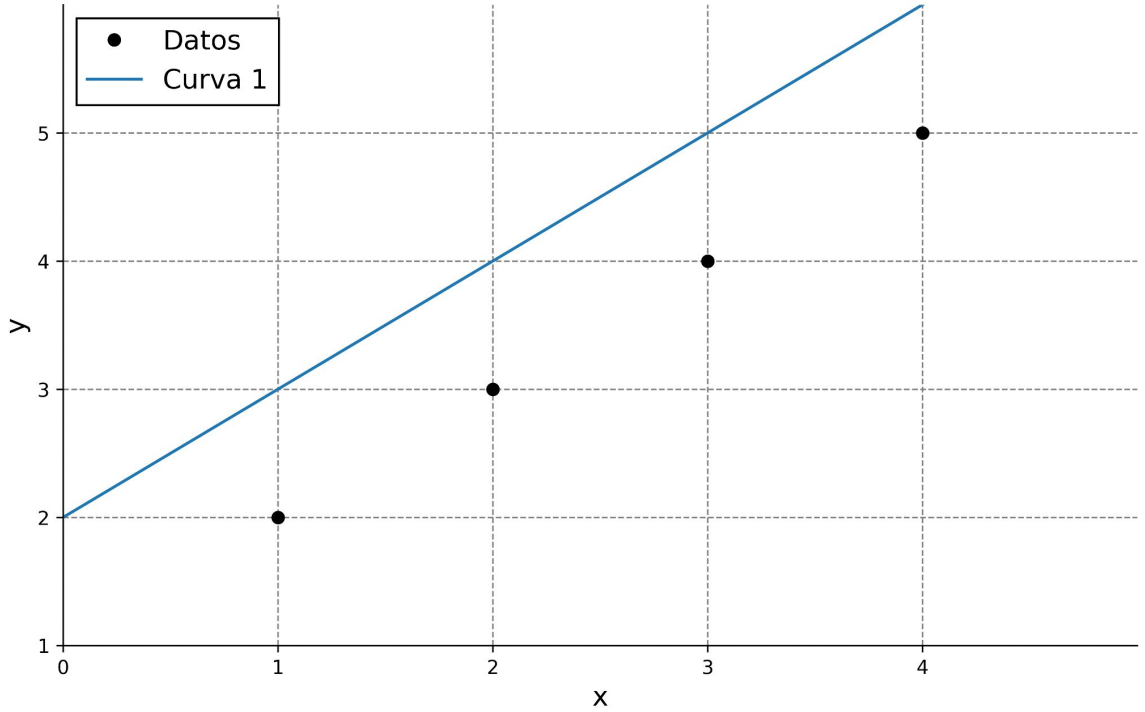
El problema a resolver

```
x = [1, 2, 3, 4]

y = [2, 4, 5, 6]

def lineal(x, a, b):
    return a*x+b
```



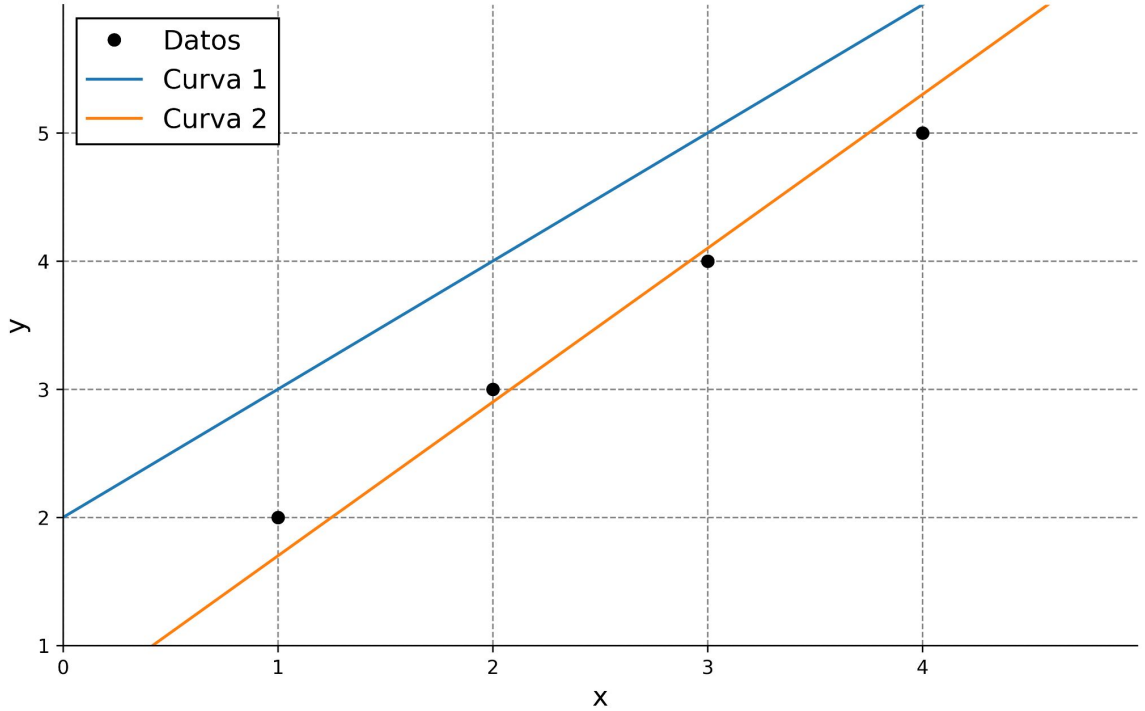
El problema a resolver

```
x = [1, 2, 3, 4]

y = [2, 4, 5, 6]

def lineal(x, a, b):
    return a*x+b
```



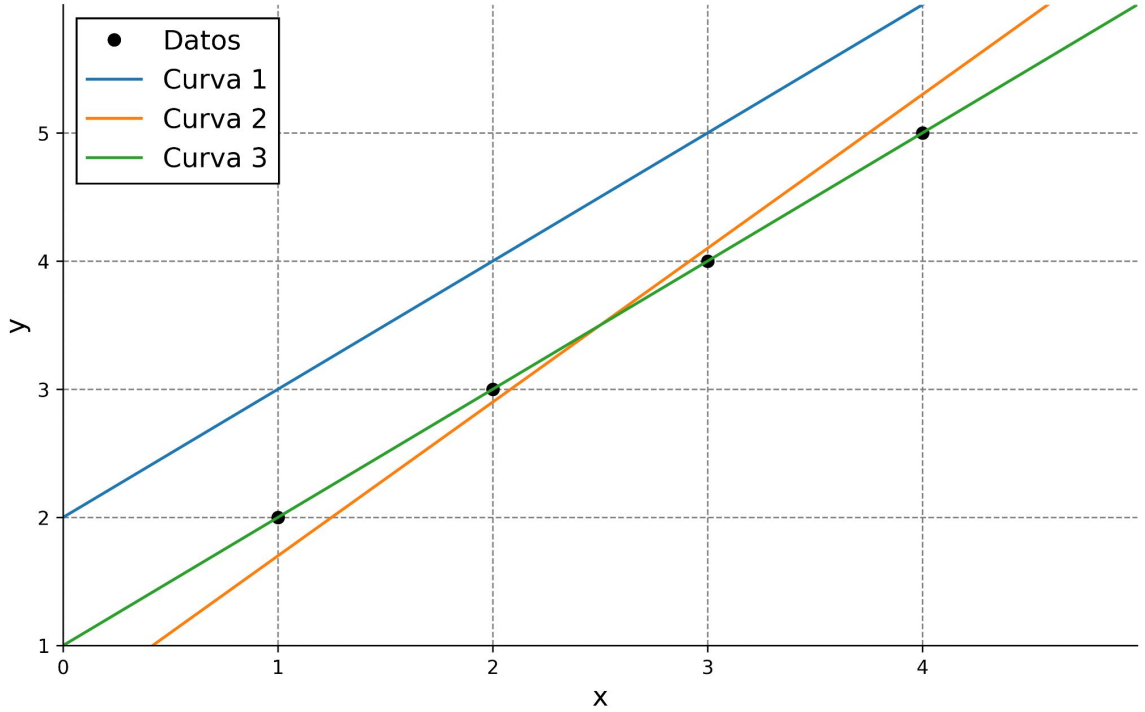
El problema a resolver

```
x = [1, 2, 3, 4]

y = [2, 4, 5, 6]

def lineal(x, a, b):
    return a*x+b
```



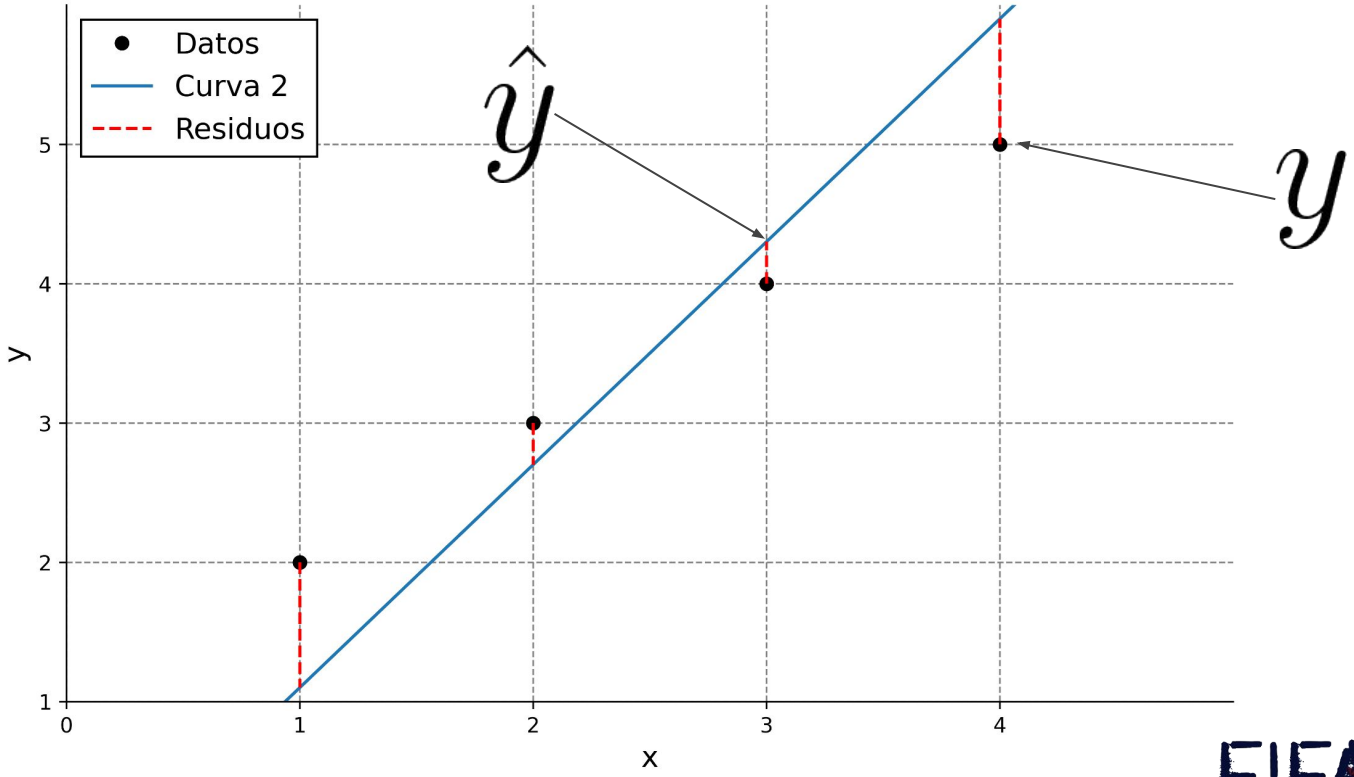
El problema a resolver

```
x = [1, 2, 3, 4]

y = [2, 4, 5, 6]

def lineal(x, a, b):
    return a*x+b
```

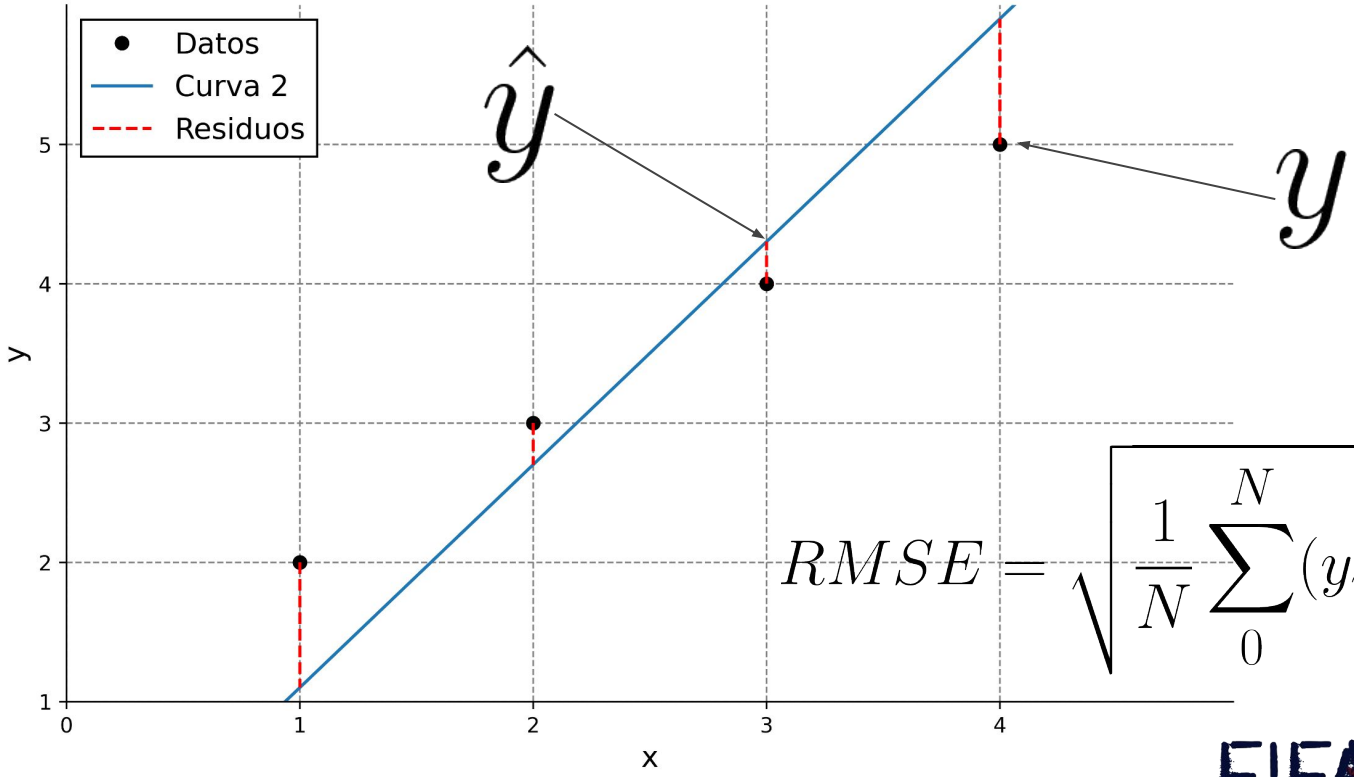


Cómo mido cual es la mejor curva?

Cómo mido cual es la mejor curva?



Cómo mido cual es la mejor curva?



Y ahora?

Quiero el valor de “a” y “b” que minimiza estas diferencias…

Hay una función que nos devuelve esto!

```
from scipy.optimize import curve_fit

x = [1, 2, 3, 4]
y = [2, 4, 5, 6]

def lineal(x, a, b):
    return a*x+b

popt, pcov = curve_fit(lineal, x, y)
```

Que le tenemos que pasar a curve_fit?

Que le tenemos que pasar a curve_fit?

```
popt, pcov = curve_fit(
    lineal,
    x,
    y,
    sigma=y_err,
    absolute_sigma=True
)
```

Que le tenemos que pasar a curve_fit?

```
popt, pcov = curve_fit(
    lineal,
    x,
    y,
    sigma=y_err,
    absolute_sigma=True
)
```

Función a ajustar

```
def lineal(x, a, b):
    return a*x+b
```

Que le tenemos que pasar a curve_fit?

```
popt, pcov = curve_fit(
    lineal,
    x,
    y,
    sigma=y_err,
    absolute_sigma=True
)
```

Valores del eje x

Que le tenemos que pasar a curve_fit?

```
popt, pcov = curve_fit(
    lineal,
    x,
    y,
    sigma=y_err,
    absolute_sigma=True
)
```

Valores del eje y

Que le tenemos que pasar a curve_fit?

```
popt, pcov = curve_fit(
    lineal,
    x,
    y,
    sigma=y_err,
    absolute_sigma=True
)
```

→ Errores del eje y (opcional)

Que le tenemos que pasar a curve_fit?

```
popt, pcov = curve_fit(
    lineal,
    x,
    y,
    sigma=y_err,
    absolute_sigma=True
)
```

Sirve para que nos de el error de “a” y “b” como en Labo 1

Que le tenemos que pasar a curve_fit?

```
popt, pcov = curve_fit(
    lineal,
    x,
    y,
    sigma=y_err,
    absolute_sigma=True
)
```

Hay mas opciones posibles, pero las importantes son estas 5

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- popt: Parámetros óptimos (Los coeficientes que queremos)

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- popt: Parámetros óptimos (Los coeficientes que queremos)

```
print(popt)
```

```
>> [1, 2]
```

Qué es 1 y que es 2?

```
def lineal(x, a, b):
    return a*x+b
```

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- popt: Parámetros óptimos (Los coeficientes que queremos)

```
print(popt)
>> [1, 2]
```

Qué es 1 y que es 2?

```
def lineal(x, a, b):
    return a*x+b
```

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- popt: Parámetros óptimos (Los coeficientes que queremos)

```
print(popt)
```

>> [1, 2]

```
def lineal(x, a, b):
    return a*x+b
```

Qué es 1 y que es 2?

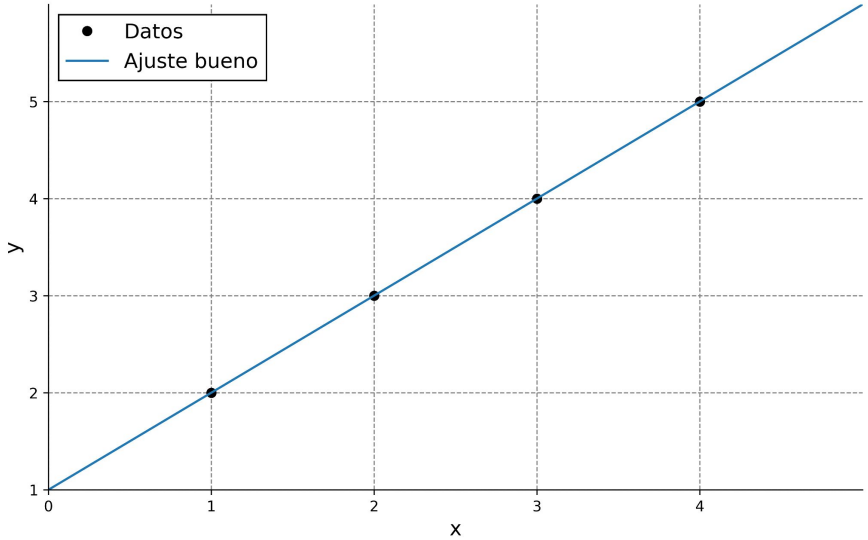
Van en orden con la función que nosotros le pasamos!
(El primer parámetro de la función es la variable independiente del ajuste)

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- popt: Parámetros óptimos (Los coeficientes que queremos)

```
a, b = popt

print(a)
>> 1
print(b)
>> 2
```



Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- pcov: Matriz de correlación (Nos da los errores de “a” y “b”)

```
print(pcov)
>> [[0.02, 0.05],
    [0.05, 0.03]]
```

Cuáles son los valores que me interesan?

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- pcov: Matriz de correlación (Nos da los errores de “a” y “b”)

```
print(pcov)
>> [[0.02, 0.05],
    [0.05, 0.03]]
```

```
def lineal(x, a, b):
    return a*x+b
```

Cuáles son los valores que me interesan?

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- pcov: Matriz de correlación (Nos da los errores de “a” y “b”)

```
print(pcov)
>> [[0.02, 0.05],
    [0.05, 0.03]]
```

```
def lineal(x, a, b):
    return a*x+b
```

Cuáles son los valores que me interesan?

Estos son los errores al cuadrado!

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- pcov: Matriz de correlación (Nos da los errores de “a” y “b”)

```
print(np.sqrt(np.diag(pcov)))
```

```
>> [0.141, 0.173]
```

```
def lineal(x, a, b):
    return a*x+b
```

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- pcov: Matriz de correlación (Nos da los errores de “a” y “b”)

```
print(np.sqrt(np.diag(pcov)))
```

```
>> [0.141, 0.173]
```

```
def lineal(x, a, b):
    return a*x+b
```

Qué devuelve curve_fit?

```
popt, pcov = curve_fit(...)
```

- pcov: Matriz de correlación (Nos da los errores de “a” y “b”)

```
a_err, b_err = np.sqrt(np.diag(pcov))
```

```
print(a_err)
>> 0.141
print(b_err)
>> 0.173
```

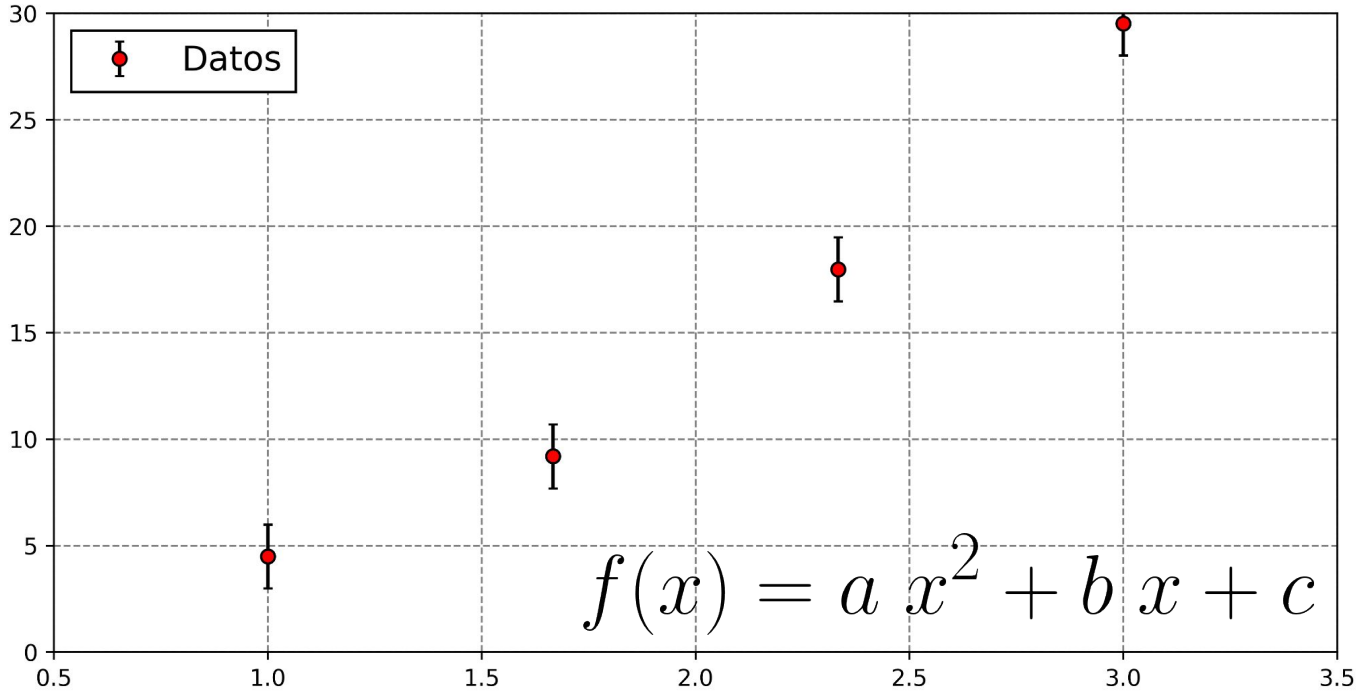
a = 1.00 ± 0.14
b = 2.00 ± 0.17

Ojo con como grafican el ajuste

Si bien no hay una forma correcta de hacerlo, hay formas que me evitan el sangrado de ojos

Ojo con como grafican el ajuste

Si bien no hay una forma correcta de hacerlo, hay formas que me evitan el sangrado de ojos



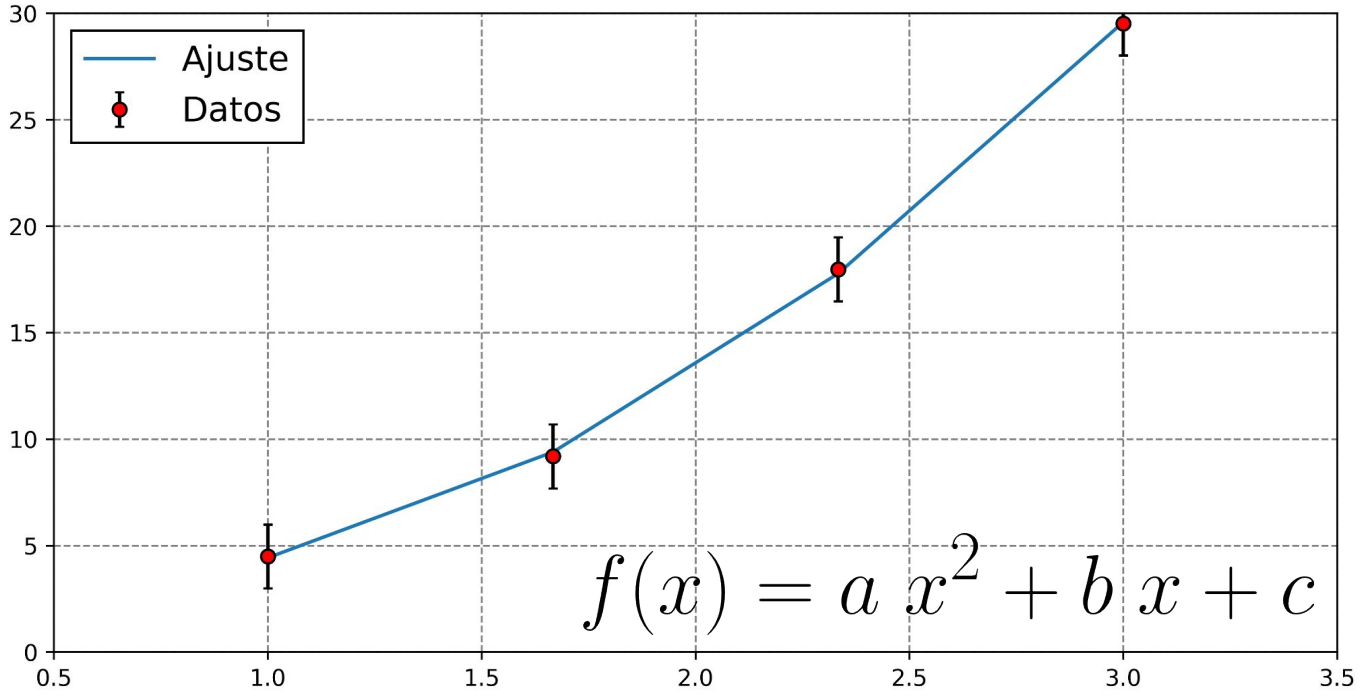
Ojo con como grafican el ajuste

Si bien no hay una forma correcta de hacerlo, hay formas que me evitan el sangrado de ojos

```
x = ...
y = ...

popt, pcov = curve_fit(cuadratica, x, y)

a, b, c = popt

plt.plot(x, cuadratica(x, a, b, c))
```

Ojo con como grafican el ajuste

Si bien no hay una forma correcta de hacerlo, hay formas que me evitan el sangrado de ojos



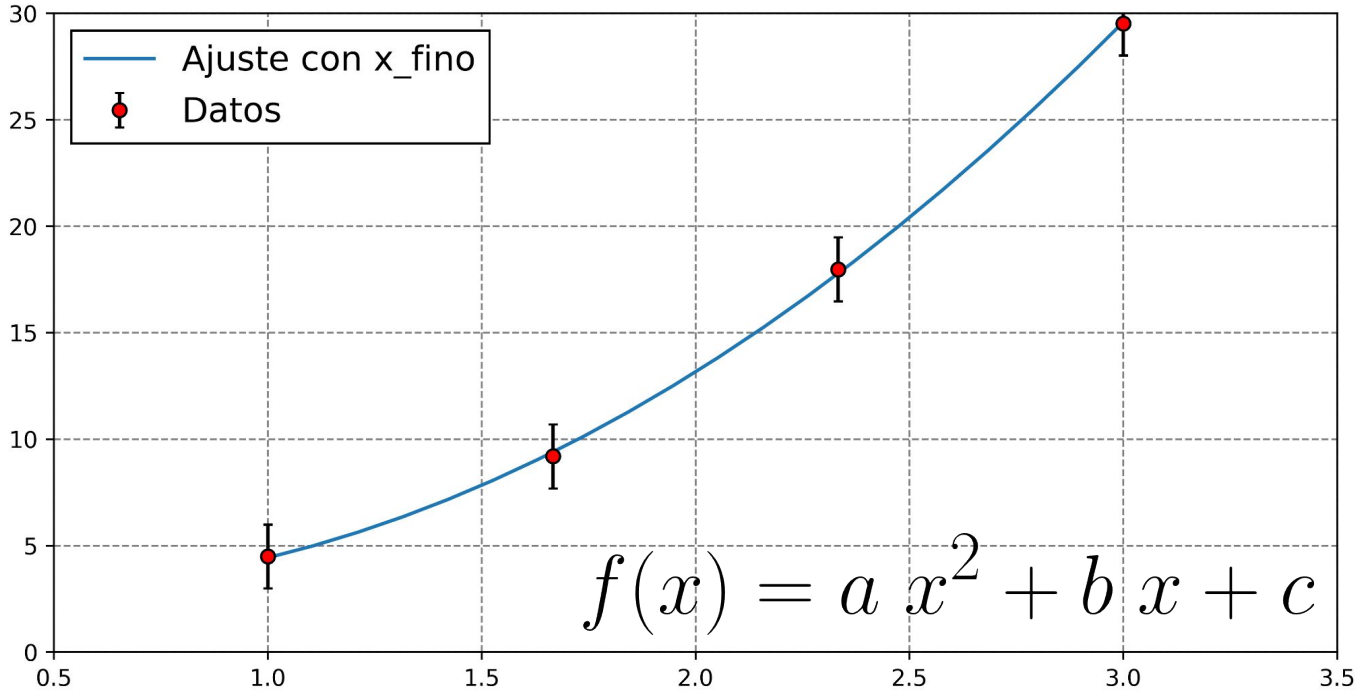
Ojo con como grafican el ajuste

Si bien no hay una forma correcta de hacerlo, hay formas que me evitan el sangrado de ojos

```
x_fino = np.linspace(min(x), max(x), len(x) * 5)

plt.plot(x_fino, cuadratica(x_fino, a, b, c))
```

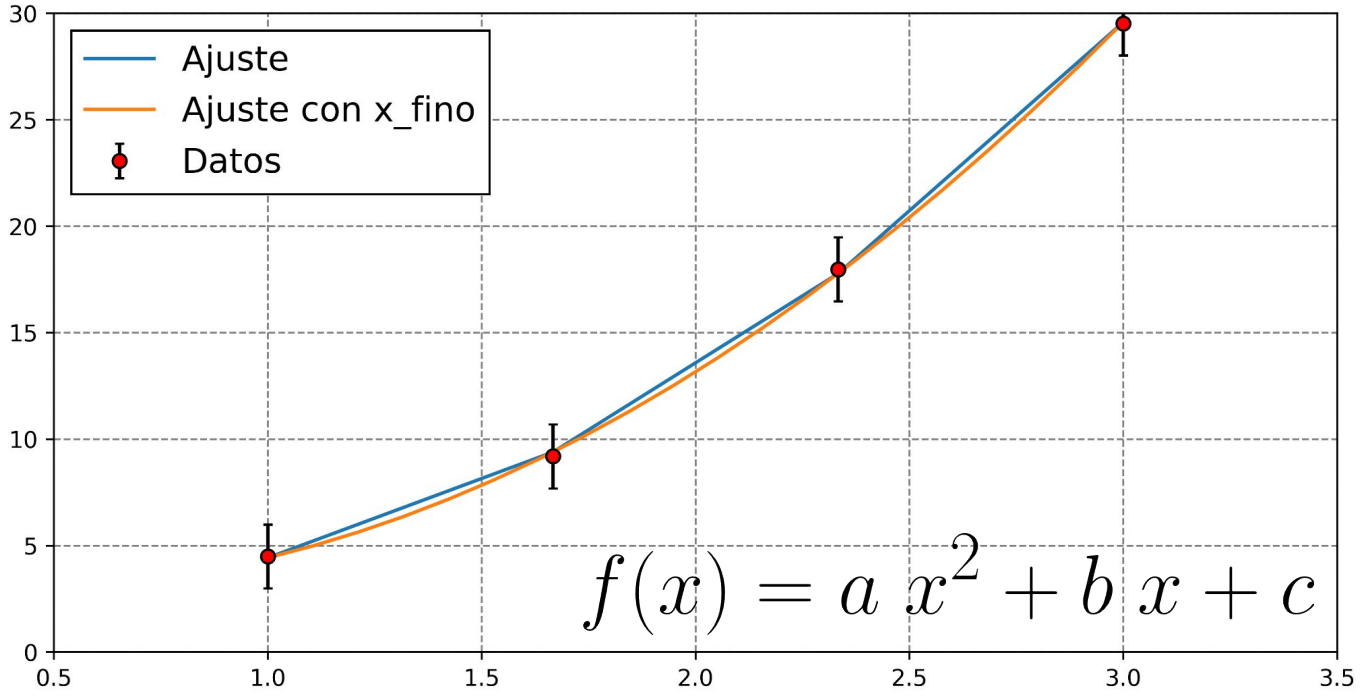
Ojo con como grafican el ajuste

Si bien no hay una forma correcta de hacerlo, hay formas que me evitan el sangrado de ojos



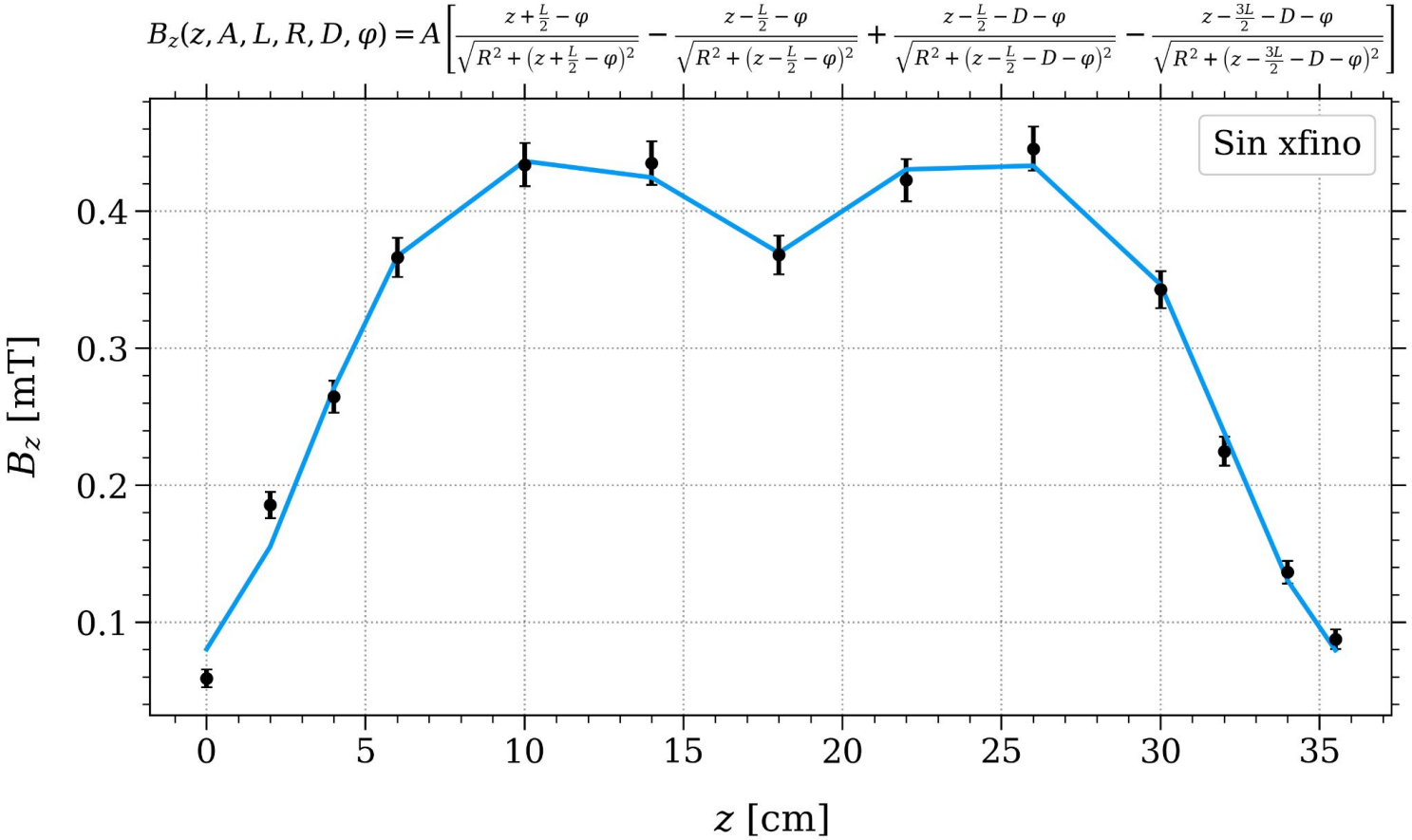
Ojo con como grafican el ajuste

Si bien no hay una forma correcta de hacerlo, hay formas que me evitan el sangrado de ojos

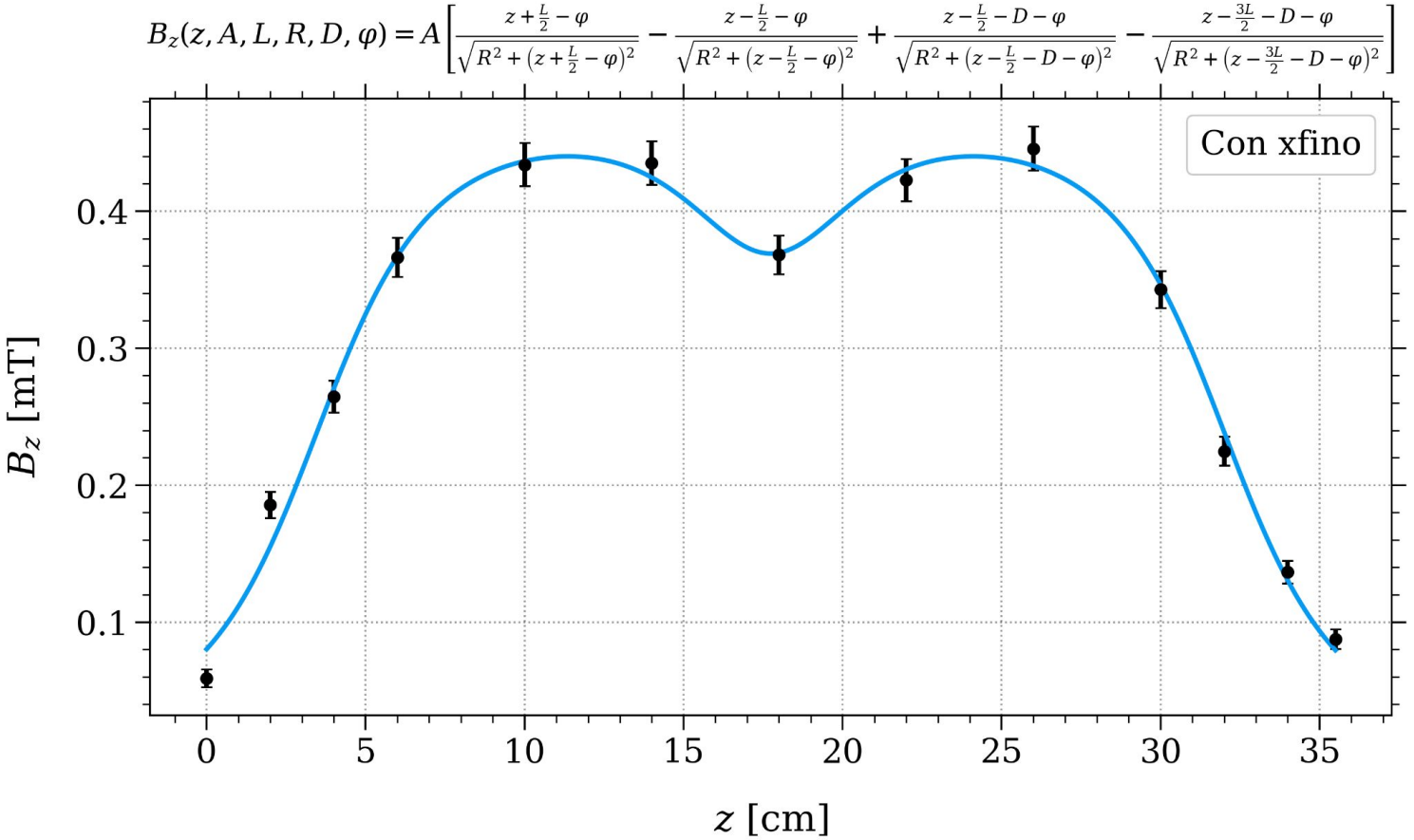


Mucho mejor el x_fino, no?

Otro ejemplo...



Otro ejemplo...



Fin del camino

Después hay otros parámetros que les pueden llegar a ser útiles en su vida de laboratorios, pero creemos que con esto está más que bien para sus primeras prácticas…

Los que tengan dudas nos pueden consultar que más se puede hacer!

Ya son expertos en *Machine Learning*