6.2

```python
def calcular_c(err,x,y):

    #calculo w
    w = 1/np.square(err)

    #terminos suma
    a = np.sum(w*np.square(x))
    b = np.sum(w*y)
    c = np.sum(w*x)
    d = np.sum(w*x*y)

    #terminos delta
    e = np.sum(w)
    f = np.sum(w*np.square(x))
    g = np.square(np.sum(w*x))
    delta = f*e - g

    #calculo c
    c = (a*b - c*d)/delta

    return c
```

```python
def calcular_m(err,x,y):

    #calculo w
    w = 1/np.square(err)

    #terminos suma
    a = np.sum(w)
    b = np.sum(w*x*y)
    c = np.sum(w*x)
    d = np.sum(w*y)

    #terminos delta
    e = np.sum(w)
    f = np.sum(w*np.square(x))
    g = np.square(np.sum(w*x))
    delta = f*e - g

    #calculo m
    m = (a*b - c*d)/delta

    return m
```

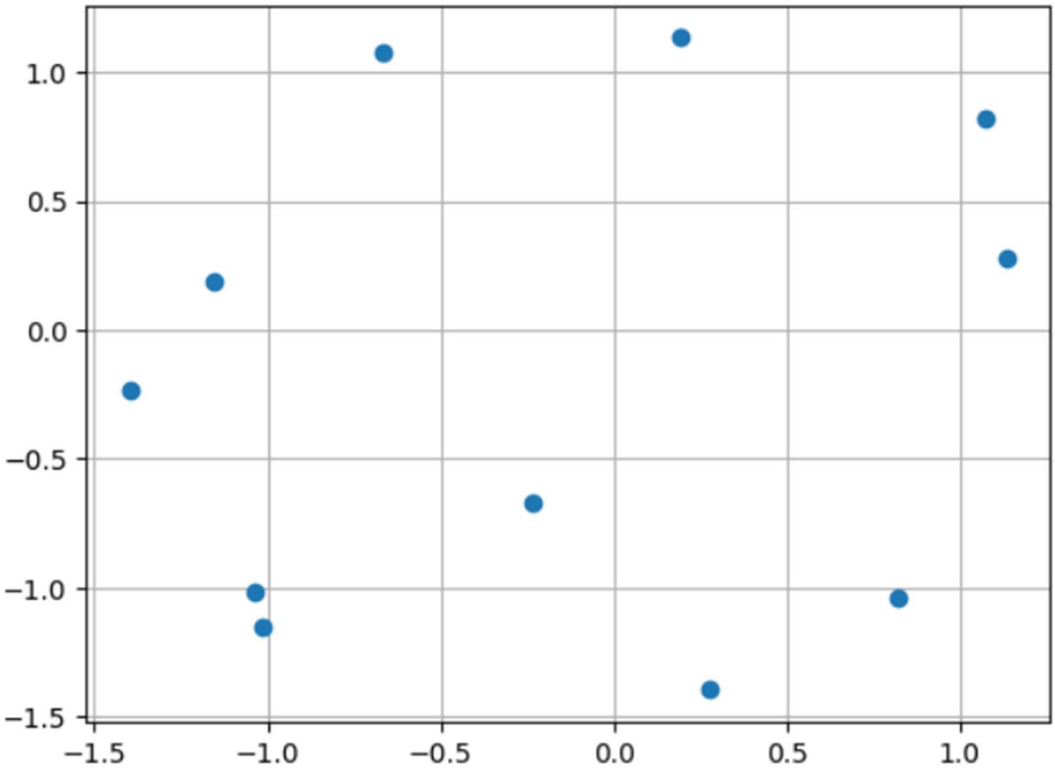
```python
def calcular_ac(err,x):

    #calculo w
    w = 1/np.square(err)

    #terminos suma
    a = np.sum(w*np.square(x))

    #terminos delta
    e = np.sum(w)
    f = np.sum(w*np.square(x))
    g = np.square(np.sum(w*x))
    delta = f*e - g

    #calculo ac
    ac = np.sqrt(a/delta)

    return ac
```

```python
def calcular_am(err,x):

        #calculo w
        w = 1/np.square(err)

        #terminos suma
        a = np.sum(w)

        #terminos delta
        e = np.sum(w)
        f = np.sum(w*np.square(x))
        g = np.square(np.sum(w*x))
        delta = f*e - g

        #calculo am
        am = np.sqrt(a/delta)

        return am
```

```
c
✓ 0.0s
-0.9474964662767381

m
✓ 0.0s
2.0284648216102745

ac
✓ 0.0s
3.386858673521736

am
✓ 0.0s
0.05197830827721802
```

```python
model = c + m*frecuencia
residuals = voltaje-model
normalized_residuals = residuals/error
lagged = np.roll(normalized_residuals, -1)
plt.plot(normalized_residuals, lagged, 'o')
plt.grid()
plt.show()
```

Lagged plot




```
def Durbin_Watson(residuals):
    return np.sum(np.square(normalized_residuals-lagged))/np.sum(normalized_residuals**2)
```

46] ✓ 0.0s

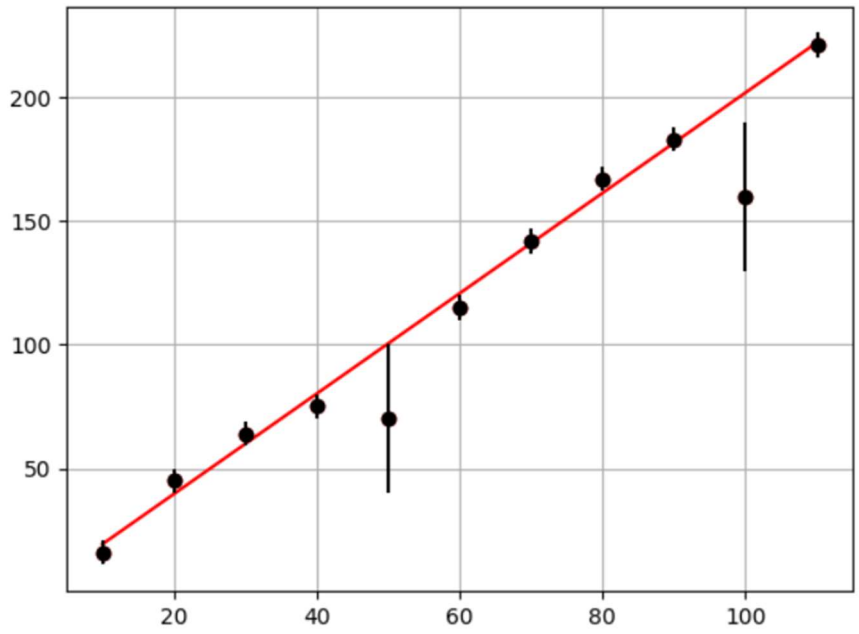
```
D = Durbin_Watson(residuals)
D
```

47] ✓ 0.0s

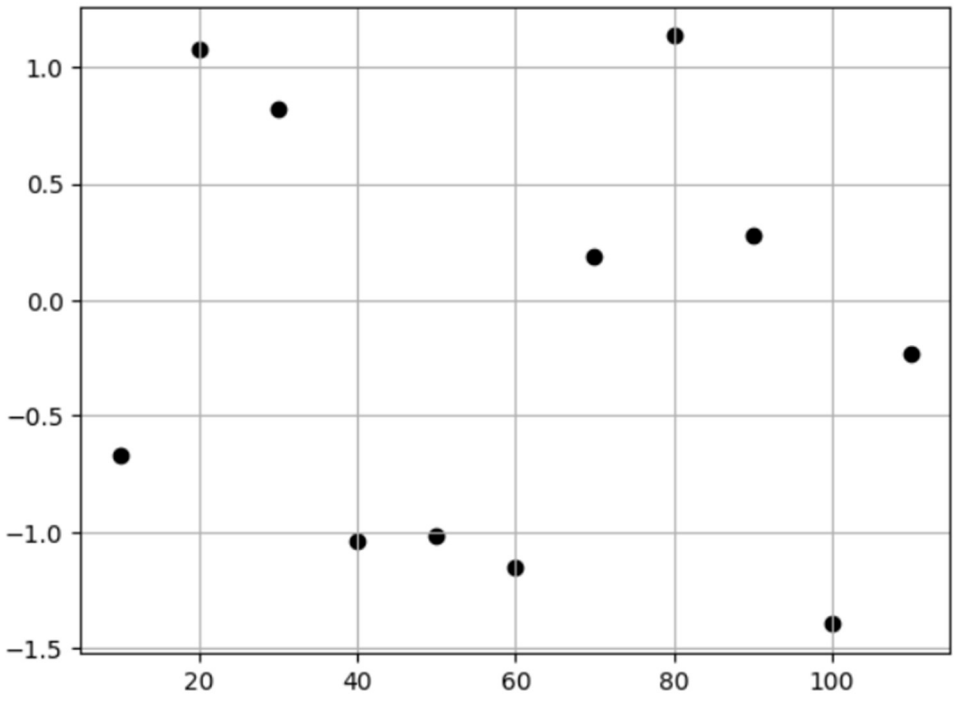
```
1.572140044519231
```

6.3

Datos con best-fit line



Normalized residuals



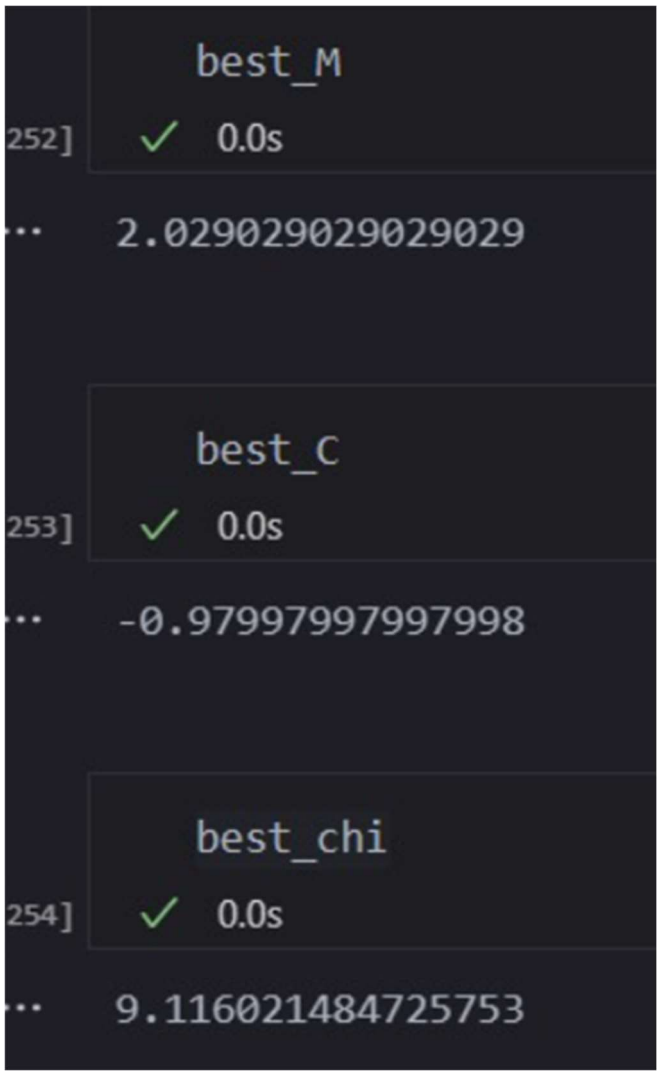
6.4

```python
def chi2(M, C):
    res = ((voltaje - (M * frecuencia + C)) / error) ** 2
    return np.sum(res)
```

✓ 0.0s

```python
M_values = np.linspace(1, 3, 1000)
C_values = np.linspace(-1, 1, 1000)
best_chi = np.inf
# Calculate chi-squared for each combination of parameters
chi2_grid = np.zeros((len(M_values), len(C_values)))
for i, M in enumerate(M_values):
    for j, C in enumerate(C_values):
        chi2_grid[i, j] = chi2(M, C)
        if chi2_grid[i, j] < best_chi:
            best_chi = chi2_grid[i, j]
            best_M = M
            best_C = C
```

✓ 8.0s




```python
def solve_Espacio_fase_c(m=m_min,c=c_min,x=x,y=y,contorno=1,alpha=dy,gmm=0.0001,infer=-1,tolerancia=1e-7):

    mjAnterior = m
    d = 1
    cj = c

    while d>tolerancia:
        #Loop que minimiza c

        cjAnterior = cj

        cj = nwt(f_c,cj+0.01*infer,args=(mjAnterior,x,y,alpha,contorno)) #Newton rapshon para minimizar c

        termino = 2*gmm*np.sum(x*(y-(mjAnterior)*x-cj)/(alpha)**2)

        mjAnterior += termino

        diff=np.abs(cj-cjAnterior)

        d = diff

    return cj
```

```python
def solve_Espacio_fase_m(m=m_min,c=c_min,x=x,y=y,contorno=1,alpha=dy,gmm=0.05,infer=-1,tolerancia=1e-7):

    cjAnterior = c
    d = 1
    mj = m

    while d>tolerancia:
        #loop que minimiza m

        mjAnterior = mj

        mj = nwt(f_m,mj+0.02*infer,args=(cjAnterior,x,y,alpha,contorno)) #Newton rapshon para minimizar m

        termino = 2*gmm*np.sum((y-(mj)*x-cjAnterior)/(alpha)**2)

        cjAnterior += termino

        diff=np.abs(mj-mjAnterior)

        d = diff

    return mj
```

Errores calculados de m y c:

Contorno 1:

```
Inferior:


m:
-0.017325659748360422


c:
-1.1289533645592529


Superior:


m:
0.017325677963016872


c:
1.1289523737901974
```

Contorno 4:

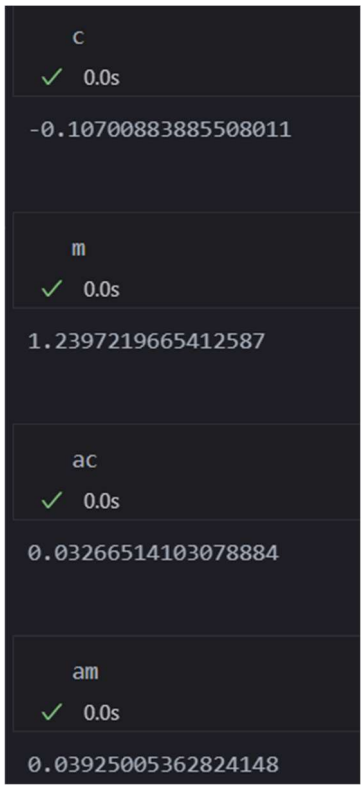
```
Inferior:


m:
-0.03465177823317589


c:
-2.2579062566317956


Superior:


m:
0.03465179644876434


c:
2.257905265862669
```

Contorno 9:

```
Inferior:


m:
-0.05197786414732053


c:
-3.3868591537285657


Superior:


m:
0.05197788236315951


c:
3.386858162959412
```

Esto se ajusta bastante bien a lo esperado.

6.5



M = 1.24 +- 0.04

C = -0.11 +- 0.03

Calculando la velocidad de la luz tenemos:
Vc = 304025095.50870967 +- 9807260.370254517

Ósea

Vc = 3.04*10^8 +- 0.10*10^8

El intercepto no es consistente, pero la velocidad de la luz da razonable.

6.6

i) Errores originales

ii) Errores ctes de 4

iii) errores 1 y final de 1, el resto de 8

A pesar de que no cambiamos los datos usados, tan solo cambiar los errores asociados a las mediciones impactó fuertemente el resultado. Esto, claramente se da porque en un cálculo con pesos de los mejores valores para la pendiente y corte, entre más preciso sea el dato, más contribuirá al resultado final. Por esto cambiar dichas precisiones afecta nuestros valores óptimos calculados.

8.2)

a) $\chi_\nu^2 = \frac{\chi^2_{min}}{\nu} = \frac{15.9}{10} = 1.59$

b) $X(\chi^2;\nu) = \frac{(\chi^2)^{(\frac{\nu}{2}-1)} \exp\{-\chi^2/2\}}{2^{\nu/2}\,\Gamma(\nu/2)}$; $P(\chi^2_{min};\nu) = \int_{\chi^2_{min}}^{\infty} X(\chi^2,\nu)\,d\chi^2$

$X(\chi^2;10) = \frac{(\chi^2)^4 \exp\{-\chi^2/2\}}{2^5\,4!}$

$\therefore P(\chi^2_{min};\nu) = \frac{1}{768}\int_{15.9}^{\infty} (\chi^2)^4 \exp\{-\chi^2/2\}\,d\chi^2$

D		I
$(\chi^2)^4$	+	$\exp\{-\chi^2/2\}$
$4(\chi^2)^3$	−	$-\frac{1}{2}$
$12(\chi^2)^2$	+	$\frac{1}{4}$
$24(\chi^2)$	−	$-\frac{1}{8}$
24	+	$\frac{1}{16}$
0		$-\frac{1}{32}$

$\therefore P(\chi^2_{min};\nu) = -\frac{1}{768}\exp\{-\chi^2/2\}\left[\frac{(\chi^2)^4}{2} + (\chi^2)^3 + \frac{3}{2}\left[(\chi^2)^2 + (\chi^2)\right] + \frac{3}{4}\right]\Big|_{15.9}^{\infty}$

$P(\chi^2_{min};\nu) = 0.1 \approx 10\%$

c) Para $\nu = 100$; $\chi^2_{min} = 159$

$\chi_\nu^2 = \frac{\chi^2_{min}}{\nu} = \frac{159}{100} = 1.59$

d) $P(\chi^2_{min};\nu) = \frac{1}{2^{50}\,49!}\int_{159}^{\infty} (\chi^2)^{49}\exp\{-\chi^2/2\}\,d\chi^2 =$

$\therefore P(\chi^2_{min};\nu) \approx 0.00016$

Vemos que al aumentar los grados de libertad, disminuye la probabilidad de obtener un valor de x^2_{min}, luego para 10 grados es un mejor fit. Además, $x^2_{min} > \nu + 3\sqrt{2\nu}$, $100 = \nu$ pero para $10 = \nu$ no.

8.3) $N = 2500$

Eq. 3.9 $\Rightarrow P(x_1 \leq x < x_2) = \frac{1}{\sigma\sqrt{2\pi}} \int_{x_1}^{x_2} \exp\left\{-\frac{(x-\bar{x})^2}{2\sigma^2}\right\} dx$

$E_i = N \cdot P$

$= E_{rf}(x_2; \bar{x}; \sigma) - E_{rf}(x_1; \bar{x}; \sigma)$, pero sea $\sigma = 1$; $\bar{x} = 0$

Usando Wolfram

x_1	x_2	O_i	P	E_i	$\frac{(O_i - E_i)^2}{E_i}$
$-\infty$	-2.5	9	0.0062	15.5	2.73
-2.5	-2	48	0.0165	41.25	1.02
-2	-1.5	142	0.044	110	9.418
-1.5	-1	154	0.0919	230	25.348
-1	-0.5	438	0.1498	374.5	10.76
-0.5	0	521	0.1914	478.5	3.775
0	0.5	405	0.1914	478.5	11.289
0.5	1	318	0.1498	374.5	8.524
1	1.5	299	0.0919	230	20.7
1.5	2	100	0.044	110	0.909
2	2.5	57	0.0165	41.25	6.07
2.5	∞	9	0.0062	15.5	2.73

i) V

ii) Podemos ver que todos los $E_i > 5$, luego no hace falta combinar

iii) $\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} = 103.213$

iv) $\nu = N - \mathcal{N}$

$\nu = 12 - 2$

$\nu = 10$

v) No es consistente con la hipótesis de una distribución Gaussiana.

8.5) $N = 101$

x_1	x_2	O_i	E_i	$\frac{(O_i - E_i)^2}{E_i}$
1	10	6	$11.\overline{2}$	2.43
11	20	11	$11.\overline{2}$	0.0044
21	30	8	$11.\overline{2}$	0.925
31	40	8	$11.\overline{2}$	0.925
41	50	14	$11.\overline{2}$	0.687
51	60	12	$11.\overline{2}$	0.0539
61	70	11	$11.\overline{2}$	0.0044
71	80	12	$11.\overline{2}$	0.0539
81	90	19	$11.\overline{2}$	5.391
			χ^2	10.475

i Como es uniforme, esperamos la misma cantidad de goles en cada intervalo

i $E_i = \frac{101}{9}$ ✓

ii) Podemos ver que $E_i > 5$ en todos

iii) $\chi^2 = 10,475$

iv) Como solo hay una restricción:

$\nu = 9 - 1 \Rightarrow \nu = 8$

v) $\chi^2_\nu = \frac{10.475}{8} = 7,309$

$\chi^2_\nu \approx 1$, luego es un buen Fit

8.7)

FV	1	2	3	4	5	6	
O_i	17	21	14	13	16	19	100
$\frac{(O_i - E_i)^2}{E_i}$	0.06$\overline{6}$	1.12$\overline{6}$	0.42$\overline{6}$	0.80$\overline{6}$	0.02$\overline{6}$	0.32$\overline{6}$	
χ^2	2.72						

Debería ser uniforme entonces debería ser $E_i = \frac{100}{6} = 16.\overline{6}$

; $\nu = 6 - 1 = 5$

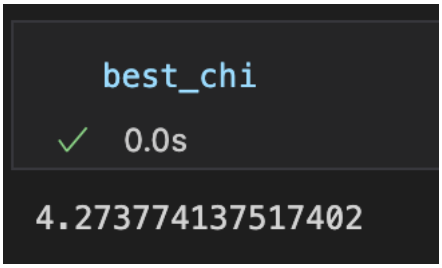
Viendo en una tabla, para 5 grados de libertad, vemos que a partir de 10% de probabilidad $\chi^2 = 9,24$, y entre más disminuye la probabilidad χ^2 aumenta.

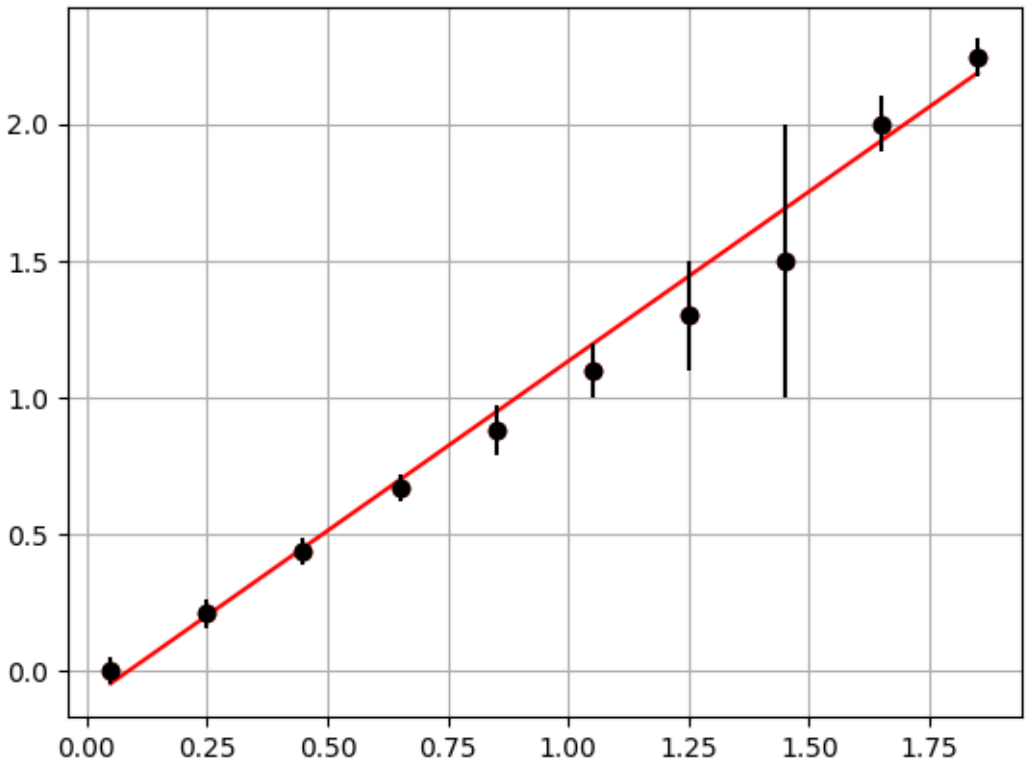
Luego $\chi^2 < 9.24$.

Osea, que un 90% de las veces que caiga el dado dará un valor correcto.

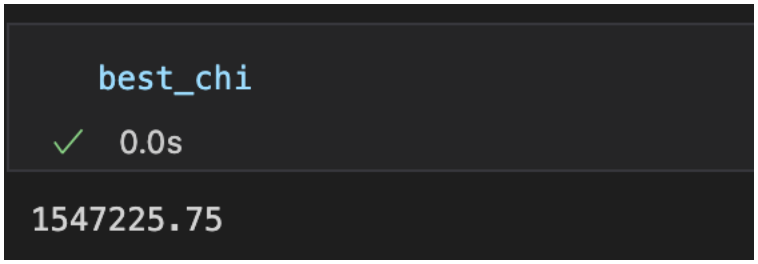
Luego el dado sí es justo.

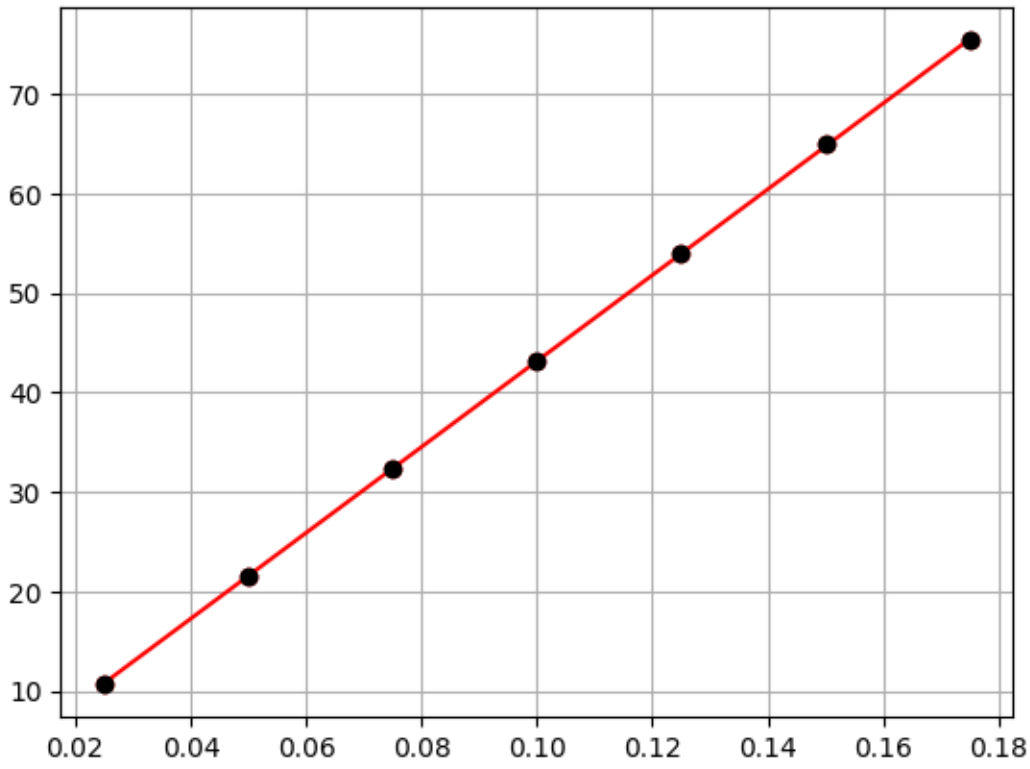
```
c
✓ 0.0s
-0.10700883885508011

m
✓ 0.0s
1.2397219665412587

ac
✓ 0.0s
0.032665141030788884

am
✓ 0.0s
0.03925005362824148
```


best_chi
0.0s
4.273774137517402


2.0
1.5
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75

```
c
✓ 0.0s
-0.02857142857142859

m
✓ 0.0s
431.71428571428567

ac
✓ 0.0s
0.0845154254728517

am
✓ 0.0s
0.7559289460184546
```


best_chi
0.0s
1547225.75


70
60
50
40
30
20
10
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18