Procesamiento y Análisis de Datos Astronómicos

9.- Modelado de datos y determinación de parámetros II

R. Gil-Hutton

Marzo 2020

Errores en el modelo:

- Si $x_1, x_2, \cdots, x_k$ son datos independientes para los cuales tenemos una $f(x|\mathbf{a})$, la función de verosimilitud es:

$$\begin{aligned} \mathcal{L}(x_1, x_2, \cdots, x_k) &= f(x_1, x_2, \cdots, x_k, \mathbf{a}) = \\ &= f(x_1|\mathbf{a}) f(x_2|\mathbf{a}) \cdots f(x_k|\mathbf{a}) = \\ &= \prod_{i=1}^{k} f(x_i|\mathbf{a}) \end{aligned}$$

- La idea básica es elegir la aproximación para $\mathbf{a}$ para la cual el valor de $\mathcal{L}$ sea tan grande como sea posible:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}} = 0 \qquad \frac{\partial \ln \mathcal{L}}{\partial \mathbf{a}} = 0$$

Errores en el modelo:

- Una forma de estimar el error en el ajuste de los datos con el modelo Y es calcular el siguiente estimador de máxima verosimilitud que asume una distribución normal para los errores:

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - Y(x_i; \mathbf{a})}{\sigma_i} \right)^2$$

donde tenemos $\nu = N - M$ grados de libertad para M parámetros del modelo, y se utilizan las funciones gamma incompletas $\mathcal{P}(\chi^2|\nu)$ y $\mathcal{Q}(\chi^2|\nu) = 1 - \mathcal{P}(\chi^2|\nu)$ para su evaluación.

Errores en el modelo:

- Es posible utilizar métodos más robustos para el ajuste.
- Con un estimador de tipo M la función de verosimilitud es función de:

$$f(x_i|\mathbf{a}) = \exp[-\rho(y_i; Y(x_i, \mathbf{a}))]$$

donde $\rho(z)$ es función de $z = [y_i - Y(x_i, \mathbf{a})]/\sigma_i$.

- Si definimos $\phi(z) = d\rho(z)/dz$, las ecuaciones de condición para cada parámetro del modelo tienen la forma:

$$\sum_i \frac{1}{\sigma_i}\phi(z)\frac{\partial Y(x_i, \mathbf{a})}{\partial a_j} = 0$$

Errores en el modelo:

- Para errores con distribución normal, tenemos que:

$$\rho(z) = \frac{1}{2}z^2 \quad \phi(z) = z \quad p(x_i|\mathbf{a}) \sim exp\left(-\frac{[y_i - Y(x_i, \mathbf{a})]^2}{2\sigma_i^2}\right)$$

- Si los errores se distribuyen según una doble exponencial:

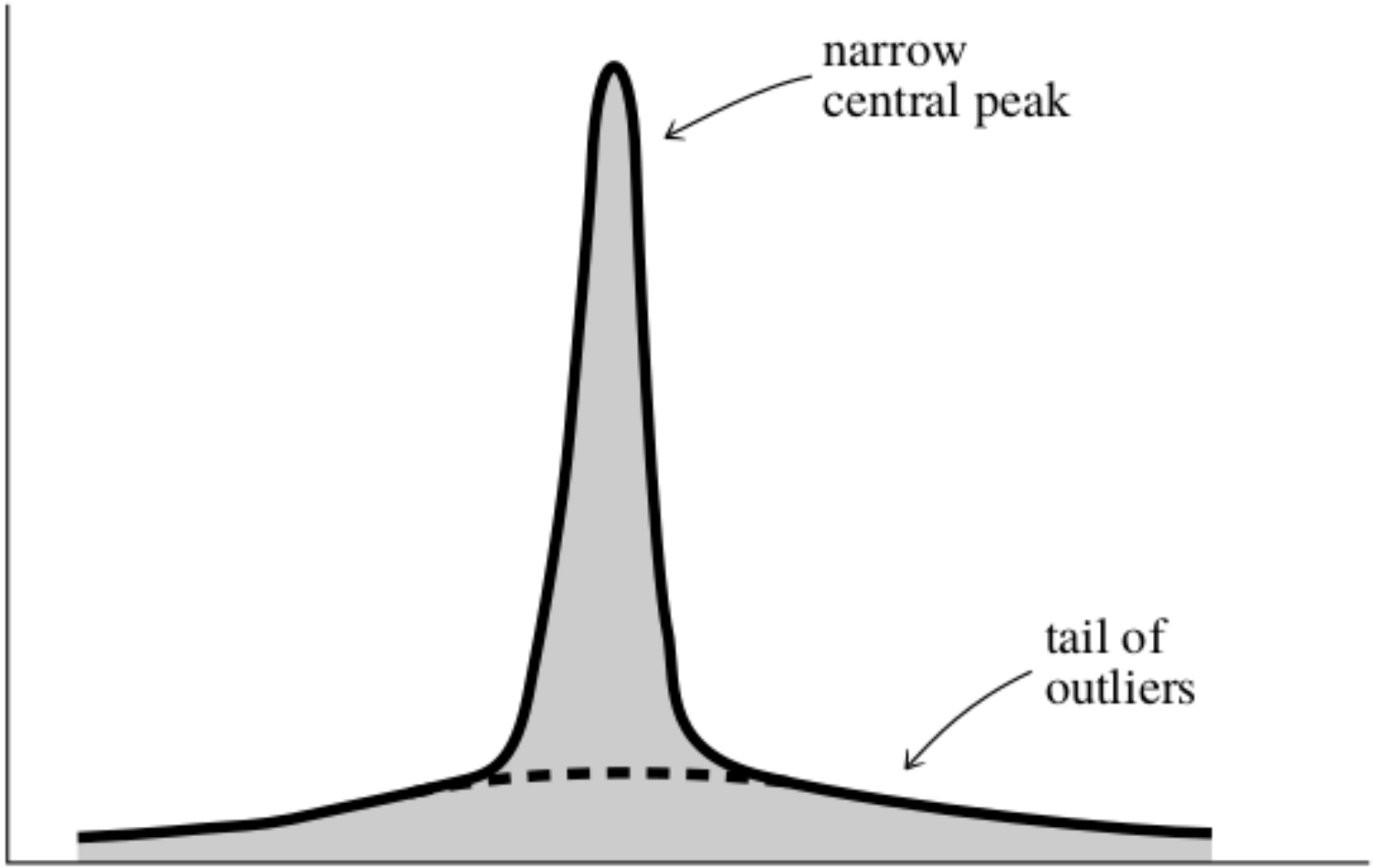
$$\rho(z) = |z| \quad \phi(z) = sgn(z) \quad p(x_i|\mathbf{a}) \sim exp\left(-\left|\frac{y_i - Y(x_i, \mathbf{a})}{\sigma_i}\right|\right)$$

- En el caso de una distribución de Lorentz:

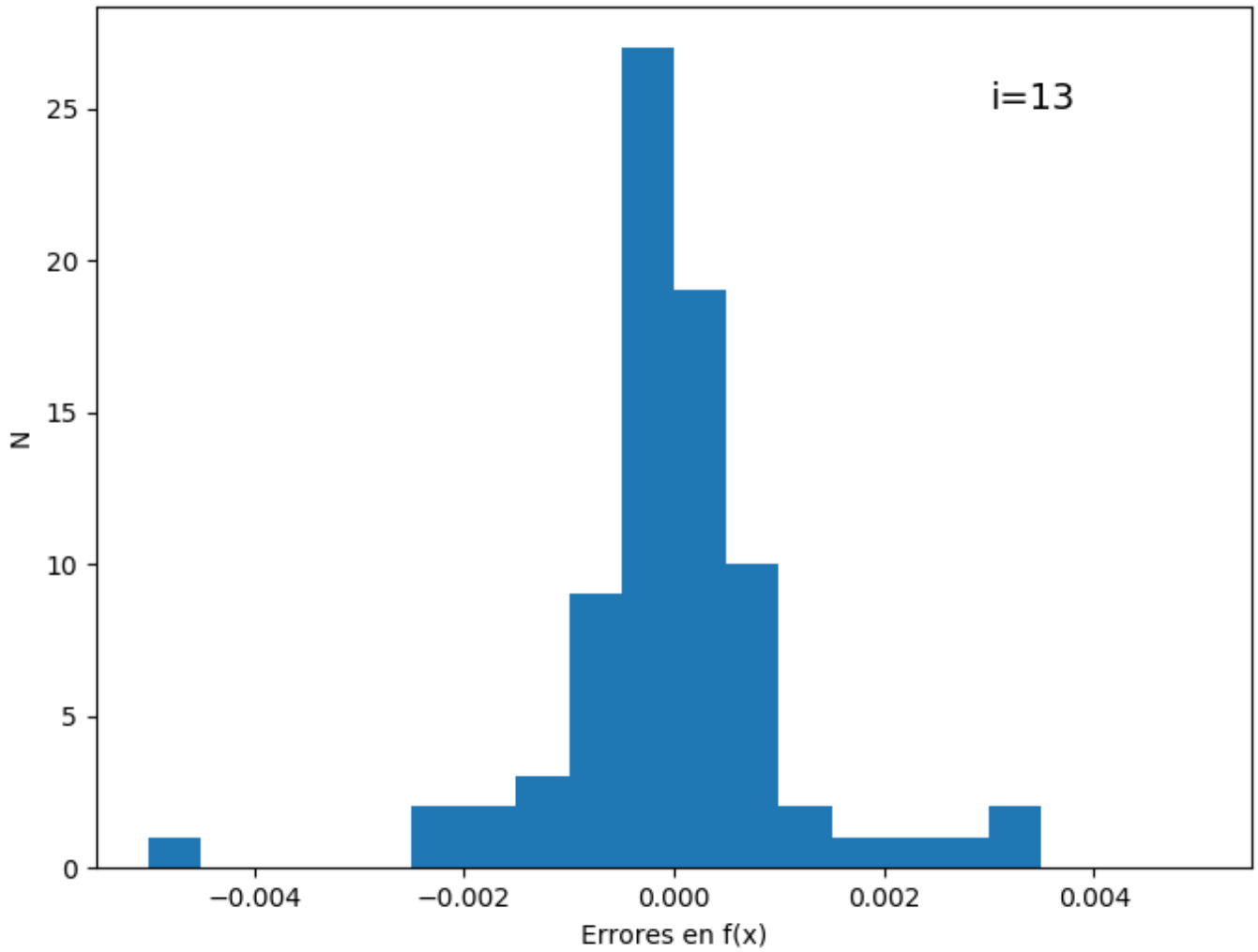
$$\rho(z) = \log\left(1 + \frac{1}{2}z^2\right) \quad \phi(z) = \frac{z}{1 + \frac{1}{2}z^2}$$

$$p(x_i|\mathbf{a}) \sim \frac{1}{1 + \frac{1}{2}\left[\frac{y_i - Y(x_i, \mathbf{a})}{\sigma_i}\right]^2}$$

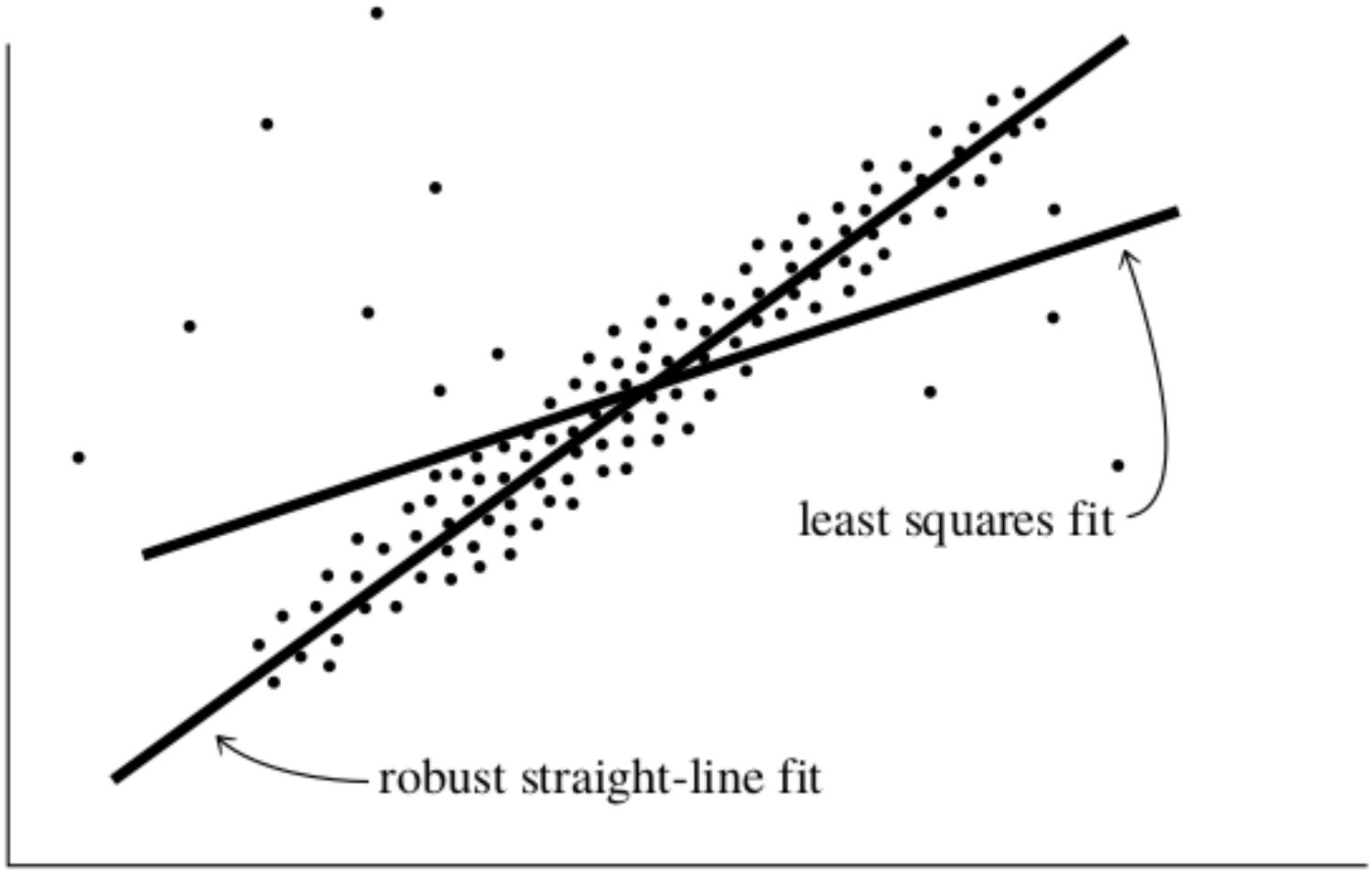
Errores en el modelo:



Errores en el modelo:



Errores en el modelo:



Errores en el modelo:

Ejemplo: Supongamos que tenemos los siguientes valores para una variable aleatoria x y su $f(x)$ observada:

xx0=[8.42, 4.85, 2.55, 8.06, 4.81, 4.77, 7.93, 9.28, 2.47, 3.09, 0.14, 7.58, 0.42, 8.13, 2.99, 5.44, 7.72, 8.67, 3.8 , 9.22]

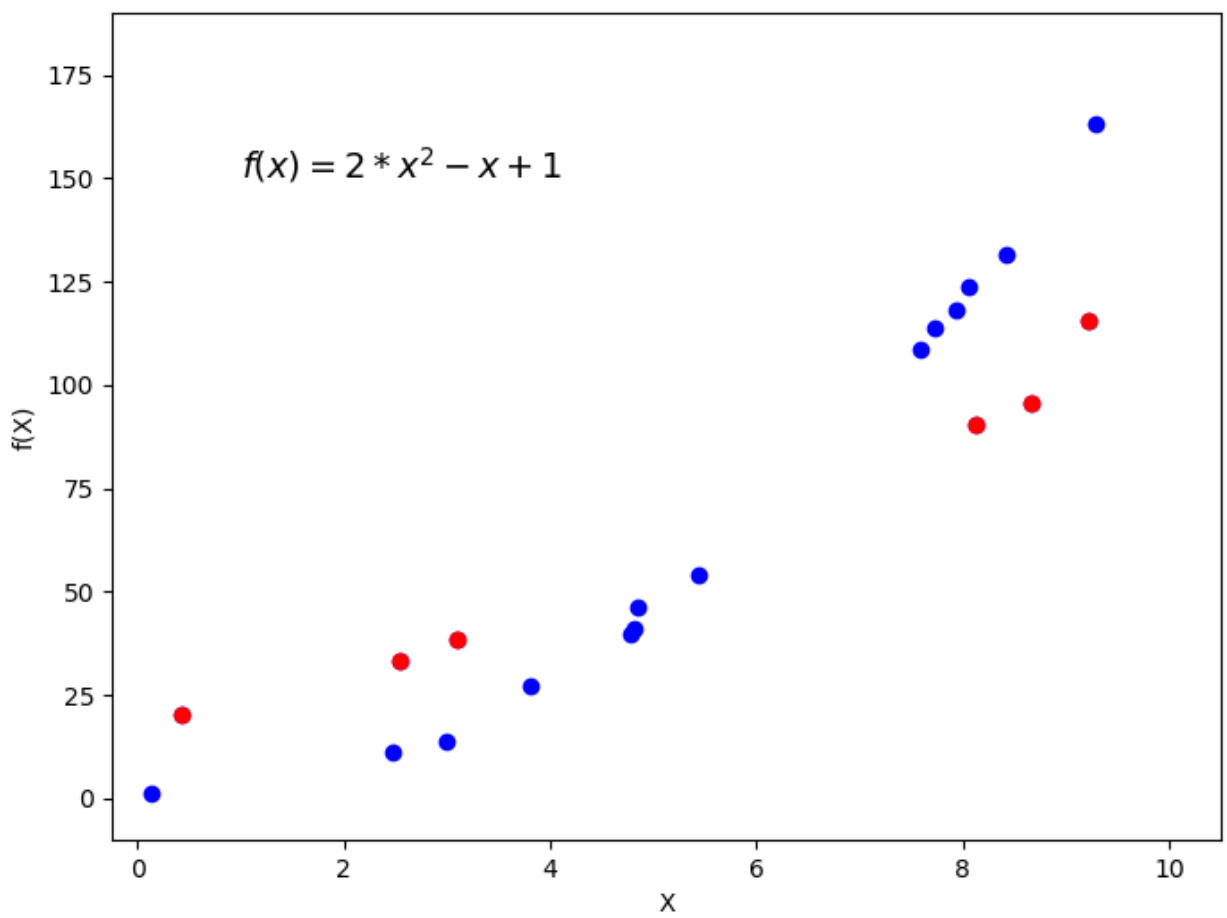
yy0=[131.77, 46.42, 33.21, 123.89, 41.21, 39.68, 118.11, 163.31, 11.41, 38.43, 1.24, 108.68, 20.21, 90.31, 13.99, 54.02, 113.63, 95.6 , 27.39, 115.48]

con errores:

er0=[2.59, 3.23, 2.10, 1.02, 1.24, 2.04, 0.72, 0.64, 0.68, 0.49, 0.34, 0.34, 0.18, 2.18, 1.89, 0.72, 1.16, 0.82, 1.31, 0.60]

y queremos ajustar un modelo $Y = a * x^2 + b * x + c$.

Errores en el modelo:



Errores en el modelo:

las ecuaciones de condición para cada parámetro del modelo tienen la forma:

$$\sum_i \frac{1}{\sigma_i}\phi(z)\frac{\partial Y(x_i, \mathbf{a})}{\partial a_j} = 0$$

siendo las derivadas del modelo $Y = a * x^2 + b * x + c$:

$$\frac{\partial Y}{\partial a} = x^2 \quad \frac{\partial Y}{\partial b} = x \quad \frac{\partial Y}{\partial c} = 1$$

Usaremos dos distribuciones para los errores: una normal y otra doble exponencial.

Errores en el modelo:

las ecuaciones de condición para una distribución normal son:

$$\sum_i \frac{1}{\sigma_i} \left(\frac{y_i - Y(x_i)}{\sigma_i} \right) x_i^2 = 0$$

$$\sum_i \frac{1}{\sigma_i} \left(\frac{y_i - Y(x_i)}{\sigma_i} \right) x_i = 0$$

$$\sum_i \frac{1}{\sigma_i} \left(\frac{y_i - Y(x_i)}{\sigma_i} \right) = 0$$

Si bien estas ecuaciones tienen solución analítica vamos a utilizar un procedimiento numérico para facilitar el cálculo.

Errores en el modelo:

Vamos a usar la función scipy.optimize.minimize() para encontrar los parámetros, pero antes necesitamos una función de mérito:

```
In [163]: def func0(p,xx,yy,er):
     ...:     nn=len(xx)
     ...:     s1=0.
     ...:     s2=0.
     ...:     s3=0.
     ...:     for ii in range(nn):
     ...:         zz=yy[ii]-(p[0]*xx[ii]**2+p[1]*xx[ii]+p[2])
     ...:         s1+=zz/er[ii]**2*xx[ii]**2
     ...:         s2+=zz/er[ii]**2*xx[ii]
     ...:         s3+=zz/er[ii]**2
     ...:     return s1**2+s2**2+s3**2
     ...:
```

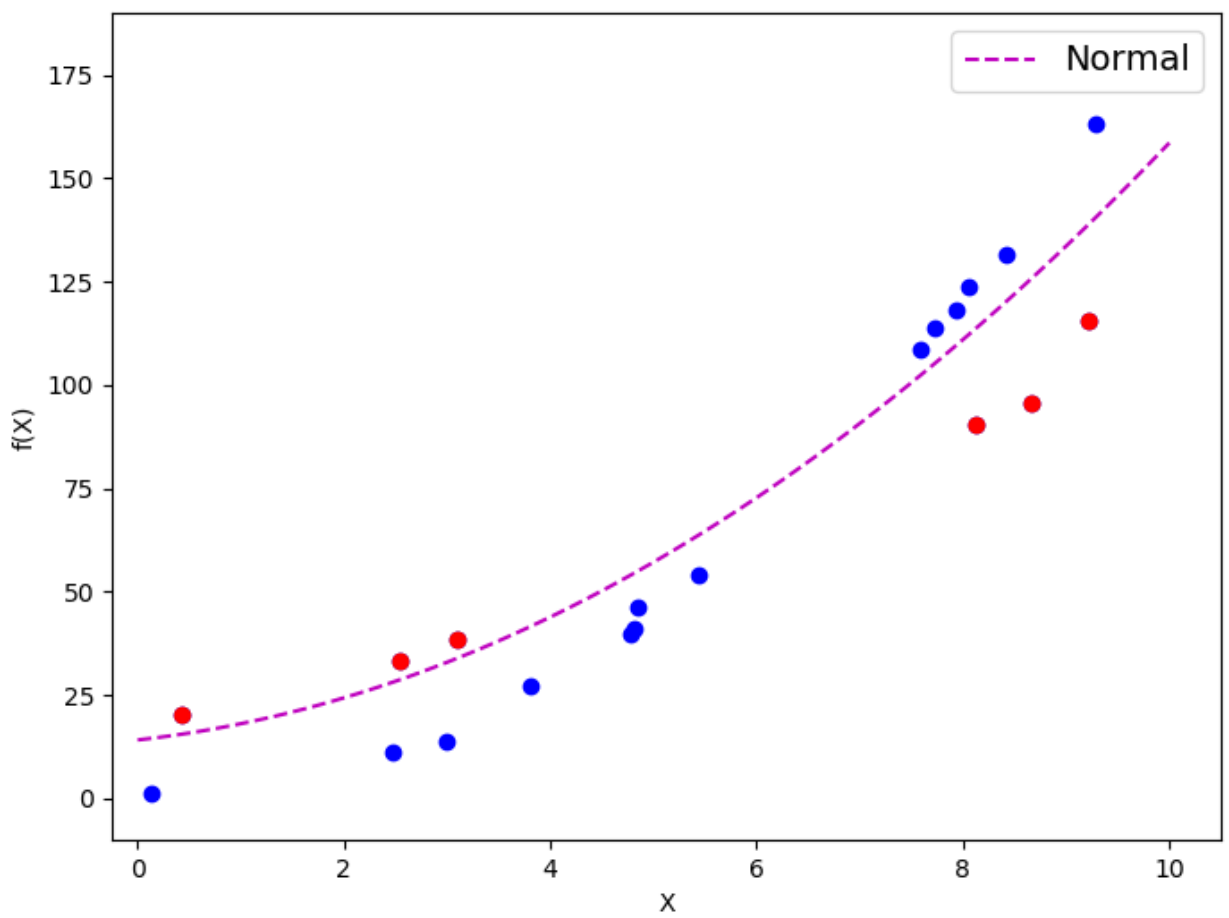
que devuelve un escalar como medida del ajuste para el vector de parámetros **p**.

Errores en el modelo:

Ahora con scipy.optimize.minimize() se minimiza la función de mérito usando un método quasi-Newtoniano:

```
In [405]: import scipy.optimize as opt

In [406]: opt.minimize(func0,[2.,-1.,1.],args=(xx0,yy0,er0),method='BFGS',tol=1.
     ...: e-3)
Out[406]:
      fun: 0.05753833262836908
 hess_inv: array([[ 8.66189920e-06, -8.04678029e-05,  7.63084529e-05],
       [-8.04678029e-05,  7.50900243e-04, -7.35322988e-04],
       [ 7.63084529e-05, -7.35322988e-04,  8.79993762e-04]])
      jac: array([-1.92318112e-05, -1.43144280e-05,  6.26547262e-06])
  message: 'Optimization terminated successfully.'
     nfev: 40
      nit: 4
     njev: 8
   status: 0
  success: True
        x: array([ 1.16846535,  2.76036752, 14.16439131])
```

Errores en el modelo:



Errores en el modelo:

las ecuaciones de condición para una distribución doble exponencial son:

$$\sum_i \frac{1}{\sigma_i} sgn\left[\frac{y_i - Y(x_i)}{\sigma_i}\right] x_i^2 = 0$$

$$\sum_i \frac{1}{\sigma_i} sgn\left[\frac{y_i - Y(x_i)}{\sigma_i}\right] x_i = 0$$

$$\sum_i \frac{1}{\sigma_i} sgn\left[\frac{y_i - Y(x_i)}{\sigma_i}\right] = 0$$

Estas ecuaciones tienen discontinuidades por lo que es estrictamente necesario utilizar un procedimiento numérico.

Errores en el modelo:

La función de mérito en este caso es:

```
In [170]: def func(p,xx,yy,er):
     ...:     nn=len(xx)
     ...:     s1=0.
     ...:     s2=0.
     ...:     s3=0.
     ...:     for ii in range(nn):
     ...:         zz=(yy[ii]-(p[0]*xx[ii]**2+p[1]*xx[ii]+p[2]))/er[ii]
     ...:         s1+=np.sign(zz)/er[ii]*xx[ii]**2
     ...:         s2+=np.sign(zz)/er[ii]*xx[ii]
     ...:         s3+=np.sign(zz)/er[ii]
     ...:     return s1**2+s2**2+s3**2
     ...:
```

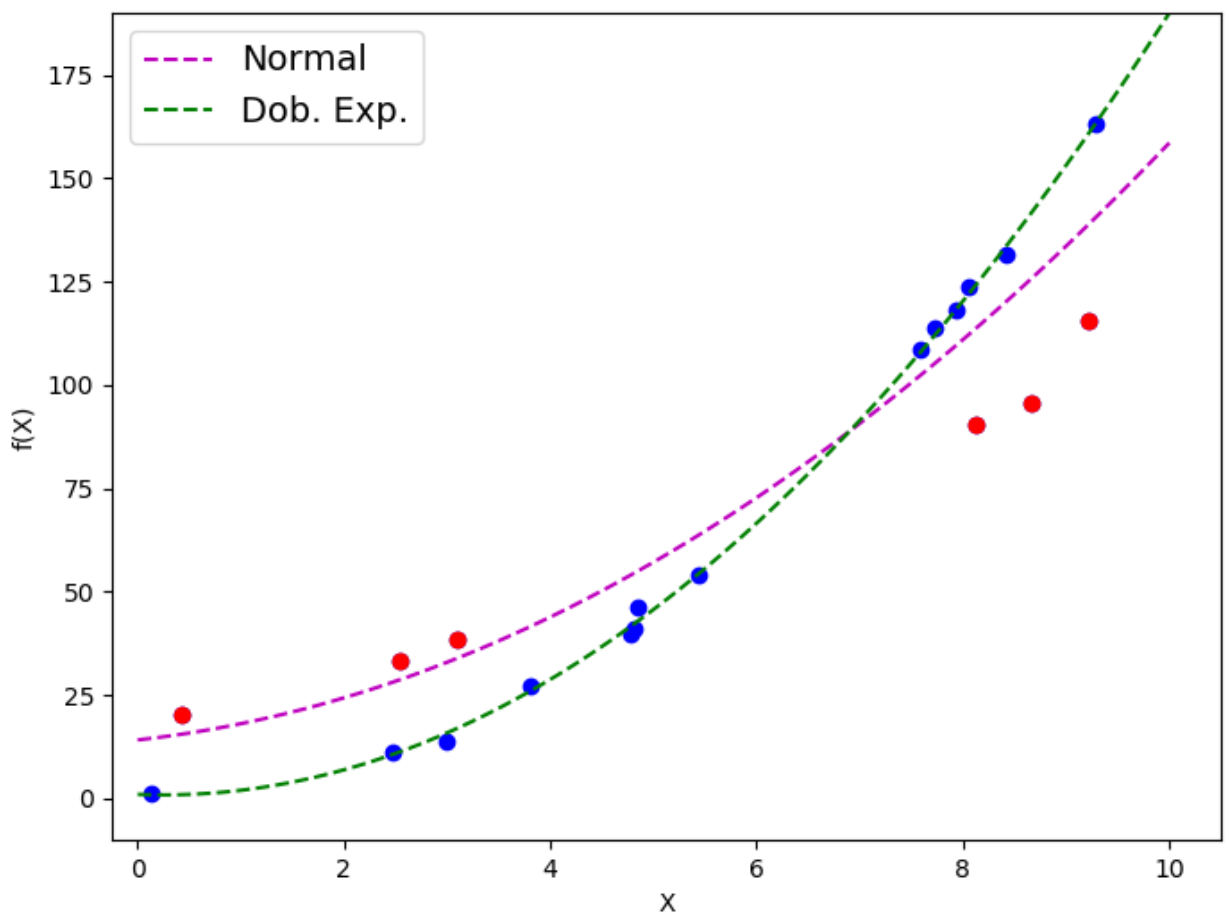
que también devuelve un escalar como medida del ajuste para el vector de parámetros **p**.

Errores en el modelo:

Ahora con scipy.optimize.minimize() se minimiza la función de mérito usando el método de Nelder-Mead o método Simplex:

```
In [408]: opt.minimize(func,[2.,-1.,1.],args=(xx0,yy0,er0),method='Nelder-Mead',
     ...: tol=1.e-3)
Out[408]:
 final_simplex: (array([[ 1.99085648, -1.01417824,  1.00677083],
       [ 1.99072989, -1.01391963,  1.0066569 ],
       [ 1.99074195, -1.01353504,  1.007449  ],
       [ 1.99078836, -1.0136559 ,  1.00686578]]), array([481.08422886, 481.08422
886, 481.08422886, 481.08422886]))
           fun: 481.0842288649228
       message: 'Optimization terminated successfully.'
          nfev: 49
           nit: 16
        status: 0
       success: True
             x: array([ 1.99085648, -1.01417824,  1.00677083])
```

Errores en el modelo:



Errores en los parámetros:

- Una muestra tomada de una población dependerá del modelo empleado para ajustar la distribución de probabilidades con sus parámetros reales y errores al azar cometidos en el proceso de adquisición:

$$\mathcal{D}_0 = Y(\mathbf{x}, \mathbf{a}_{ver}) + \mathbf{e}$$

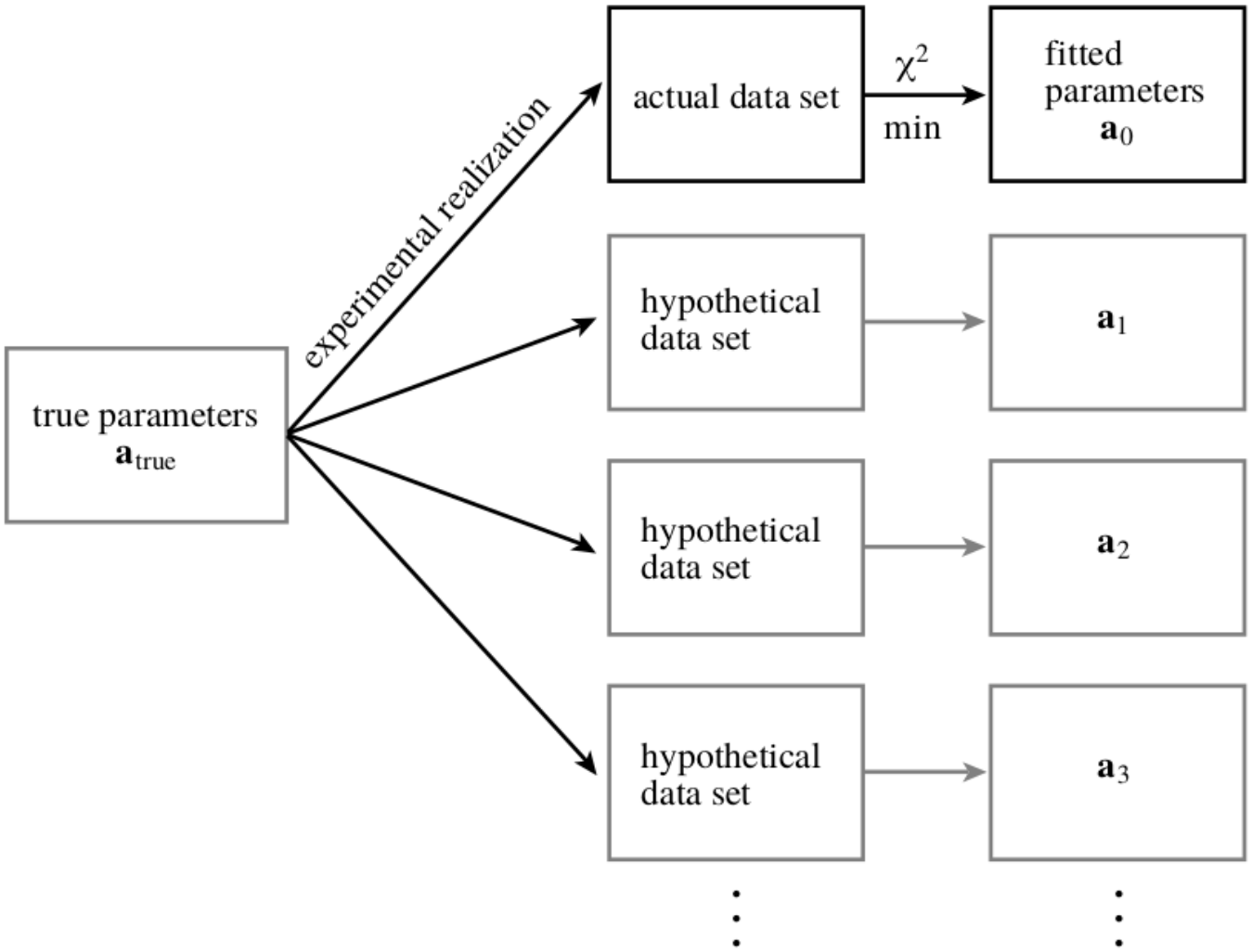
donde $\mathcal{D}_0$ es la muestra, $Y(\mathbf{x}, \mathbf{a}_{ver})$ es el modelo para la distribución real de la población con parámetros $\mathbf{a}_{ver}$, $\mathbf{x}$ corresponde a la variable aleatoria y $\mathbf{e}$ son errores de medición al azar.

- Con la muestra $\mathcal{D}_0$ el investigador ajustará su modelo y por alguno de los procedimientos posibles encontrará un vector de parámetros $\mathbf{a}_0$.

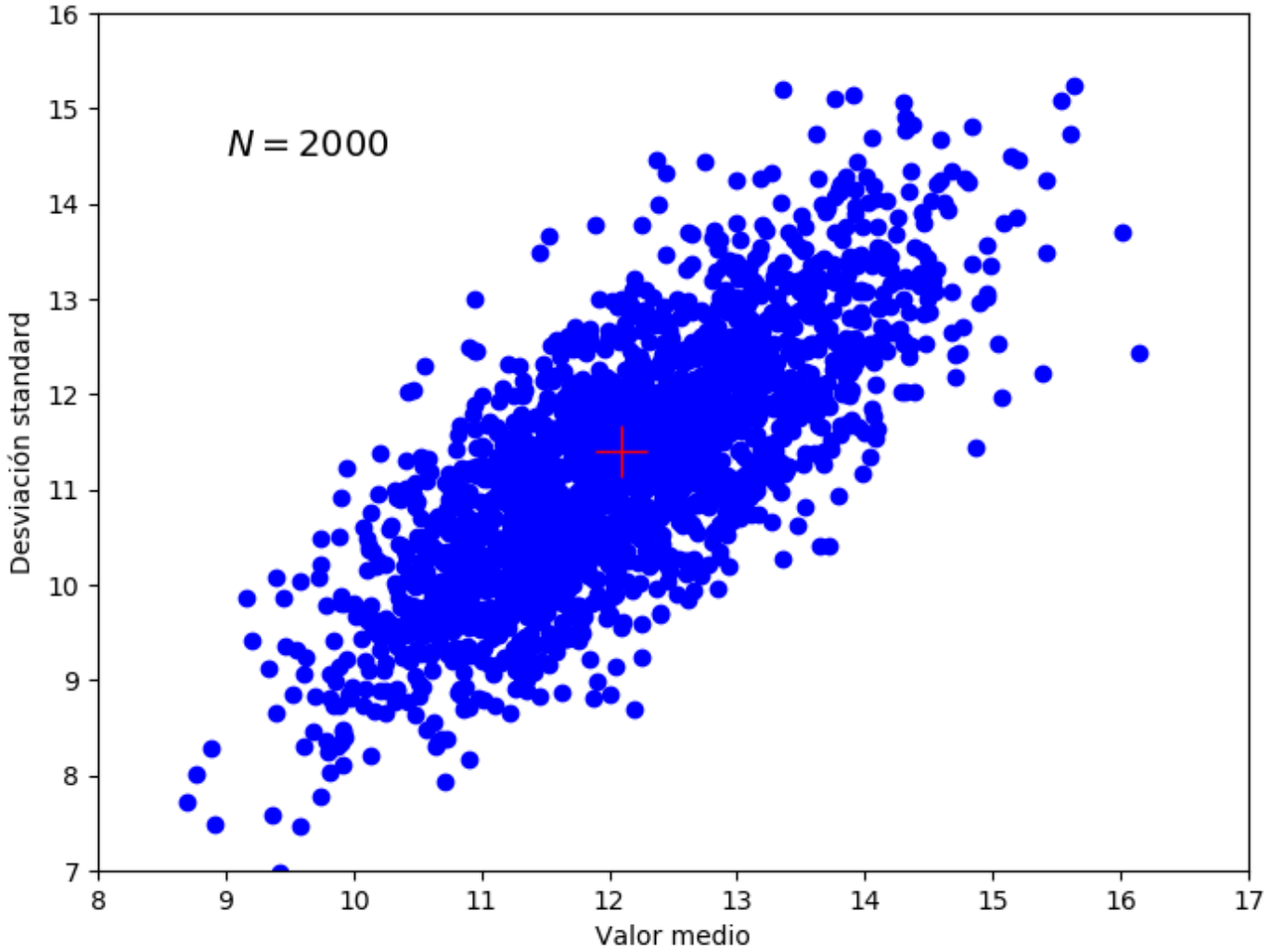
Errores en los parámetros:

- Debido a la naturaleza de los errores $\mathbf{e}$, la muestra $\mathcal{D}_0$ es una de las infinitas muestras posibles, $\mathcal{D}_1, \mathcal{D}_2, \cdots$, las cuales resultan en un conjunto de parámetros diferentes, $\mathbf{a}_1, \mathbf{a}_2, \cdots$.
- Los posibles vectores de paŕametros $\mathbf{a}_i$ se distribuyen con una cierta probabilidad en el espacio multidimensional de todos los posibles vectores de parámetros $\mathbf{a}$.
- Más importante que como se distribuyen los vectores $\mathbf{a}_i$ es la distribución de la diferencia $\mathbf{a}_i - \mathbf{a}_{ver}$, que es una traslación de la distribución respecto de los valores reales y nos permite conocer las incertezas en $\mathbf{a}_0$.

Errores en los parámetros:



Errores en los parámetros:



Simulación Monte Carlo:

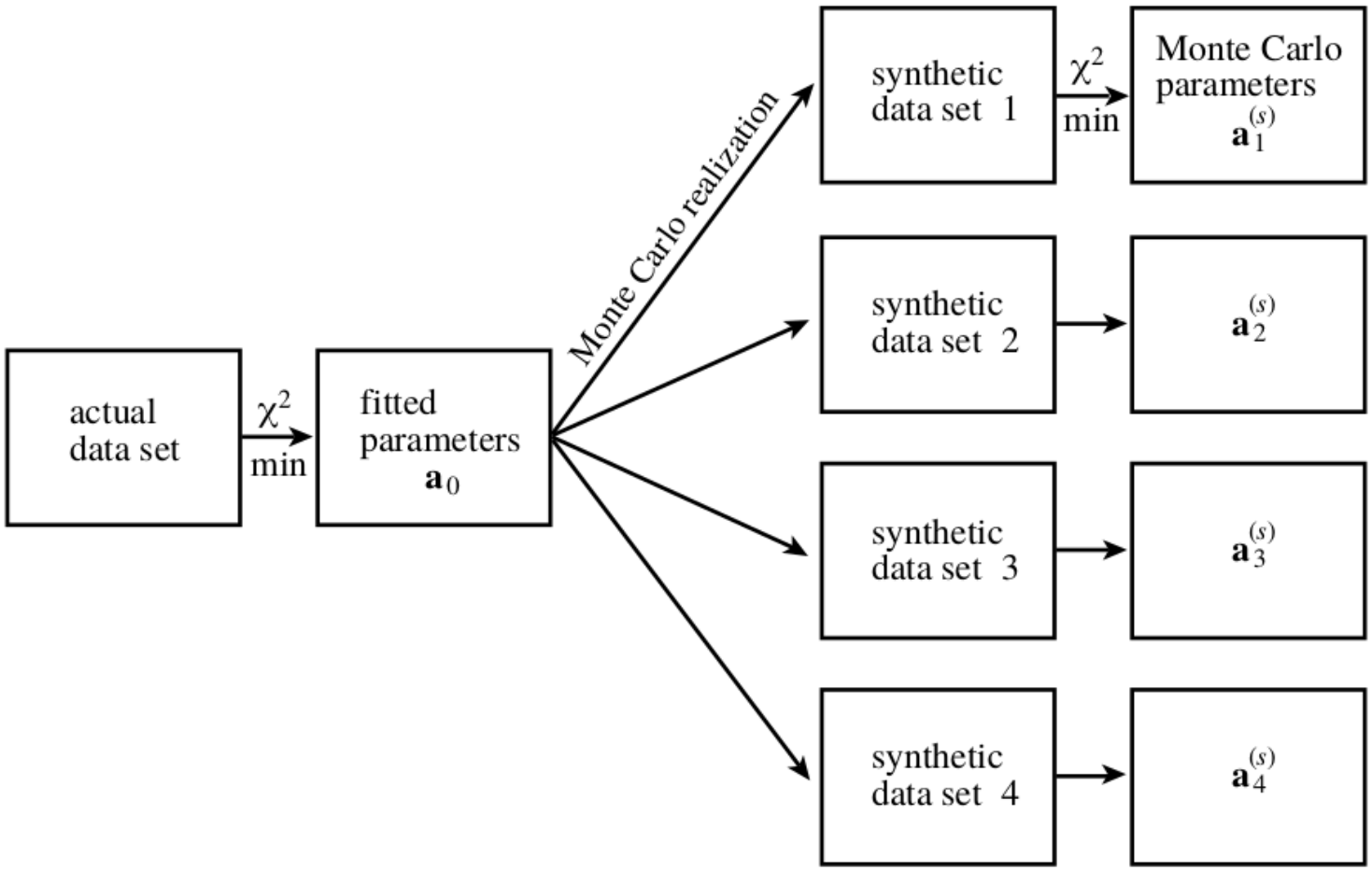
La manera de estimar las incertezas en $\mathbf{a}_0$ es mediante una simulación Monte Carlo:

- Se asume que $\mathbf{a}_0$ es el vector de parámetros verdadero, $\mathbf{a}_{ver}$.
- Entonces, se puede simular la adquisición de muestras sintéticas a partir de $\mathbf{a}_0$ y con el mismo número de elementos:

$$\mathcal{D}s_j = Y(\mathbf{x}, \mathbf{a}_0) + \mathbf{e}$$

- Estas muestras sintéticas tienen exactamente la misma relación estadística con $\mathbf{a}_0$ que las muestras $\mathcal{D}_i$ tienen con $\mathbf{a}_{ver}$.
- Si de $\mathcal{D}s_j$ se obtienen parámetros sintéticos $\mathbf{a}_j^s$, se puede obtener la distribución de probabilidades de $(\mathbf{a}_j^s - \mathbf{a}_0)$, que tendrá las mismas propiedades que la distribución de $(\mathbf{a}_i - \mathbf{a}_{ver})$.

Simulación Monte Carlo:



Simulación Monte Carlo:

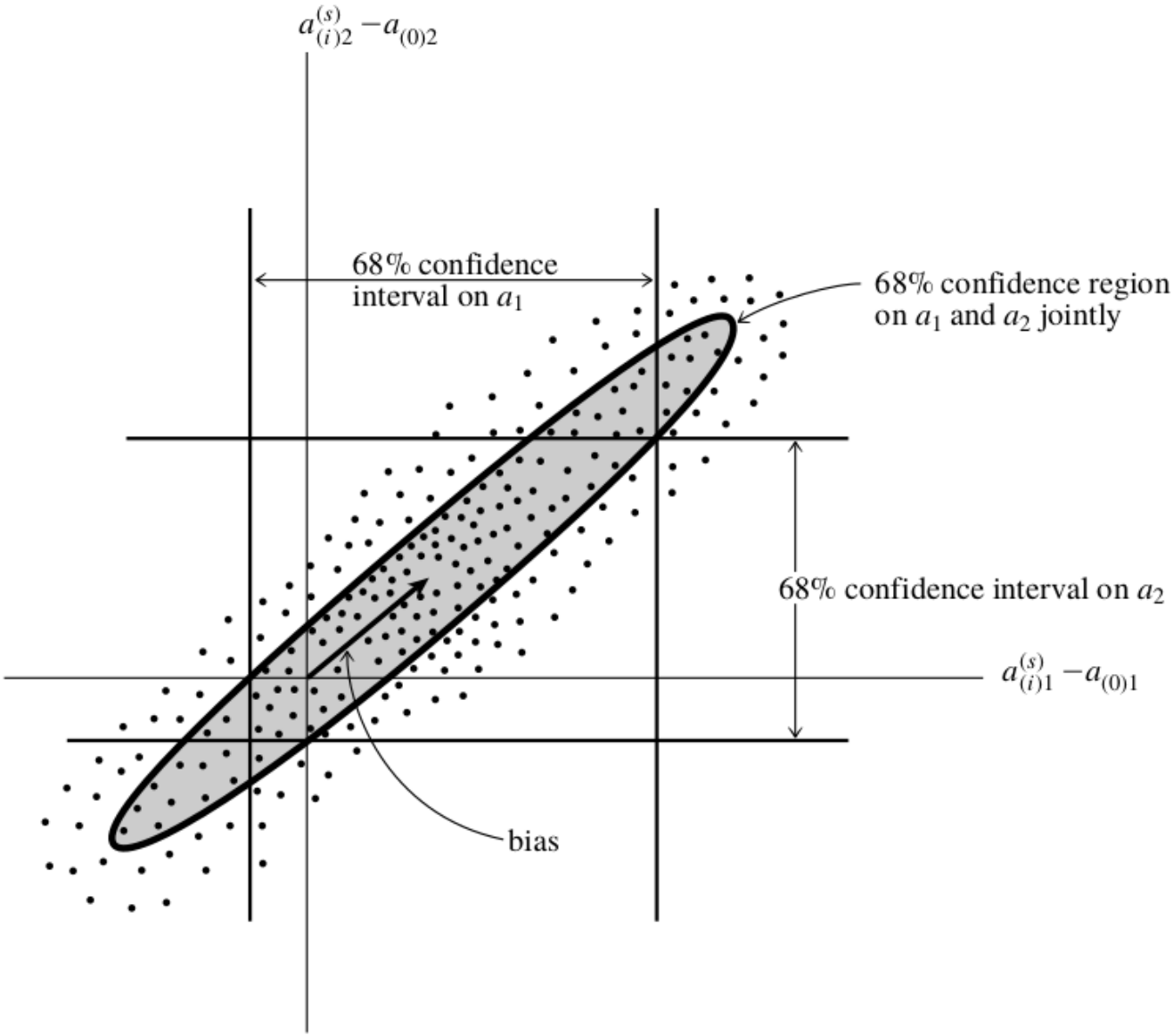
Para simular la adquisición de parámetros sintéticos a partir de $\mathbf{a}_0$ hay varios métodos:

1. A partir de la muestra original $\mathcal{D}_0$, se obtienen muestras sintéticas del mismo número de elementos tomando valores al azar con reposición de $\mathcal{D}_0$ de donde se calculan los vectores $\mathbf{a}^s$ (bootstrap).
2. A partir de la muestra original $\mathcal{D}_0$ de N elementos de donde se obtuvo $\mathbf{a}_0$ se crean N submuestras de $N-1$ elementos eliminando un elemento cada vez. Para cada muestra sintética se obtiene una estimación $\hat{\mathbf{a}}$ y los paŕametros sintéticos $\mathbf{a}^s = N\mathbf{a}_0 - (N-1)\hat{\mathbf{a}}$ (jackknife).

Intervalos de confianza:

- Es común caracterizar los errores en los parámetros de un modelo mediante regiones o intervalos de confianza en el espacio multidimensional de parámetros **a**.
- Una región de confianza es una región del espacio multidimensional de parámetros que contiene un cierto porcentaje de los valores experimentales. Valores típicos para este porcentaje son 68,3 %, 95,5 % y 99,7 % (relacionados con una distribución normal), o 90 %, 95 % y 99 % (relacionados con criterios estadísticos).
- La región de confianza de parámetros individuales en un espacio multidimensional se denominan intervalo de confianza proyectado.

Intervalos de confianza:



Intervalos de confianza:

Si el modelo se ajusta mediante χ^2 hay un modo sencillo de definir los elipsoides de la región de confianza.

- para $\mathbf{a}_0$ tendremos el valor mínimo para χ^2.
- si se elije otro vector de parámetros $\mathbf{a}_i$ el valor de χ^2 aumentará.
- la región en la cual χ^2 aumenta menos que un cierto $\Delta\chi^2$ define una región de confianza que encierra un cierto porcentaje de los $\mathbf{a}_i$.
- los valores de $\Delta\chi^2$ se eligen de tal modo de abarcar uno de los valores porcentuales típicos.

Intervalos de confianza:

Ejemplo: Se adquiere una muestra de 10000 elementos y se la ajusta a un modelo lineal $Y(x_i; a, b) = a * x + b$:

```
In [184]: xx=np.random.random(10000)*10.

In [185]: er=np.random.standard_normal(10000)*0.5

In [186]: yy=2.*xx+3.+er

In [187]: ss=np.sum(1./er**2)

In [188]: sx=np.sum(xx/er**2)

In [189]: sy=np.sum(yy/er**2)

In [190]: sxx=np.sum(xx**2/er**2)

In [191]: sxy=np.sum(xx*yy/er**2)

In [192]: dd=ss*sxx-sx**2

In [193]: aa0=(ss*sxy-sx*sy)/dd

In [194]: bb0=(sxx*sy-sx*sxy)/dd

In [195]: chisq0=np.sum((yy-(aa0*xx+bb0))**2/er**2)
```

Intervalos de confianza:

Se obtienen muestras simuladas usando bootstrap y se obtienen los parámetros del ajuste al modelo:

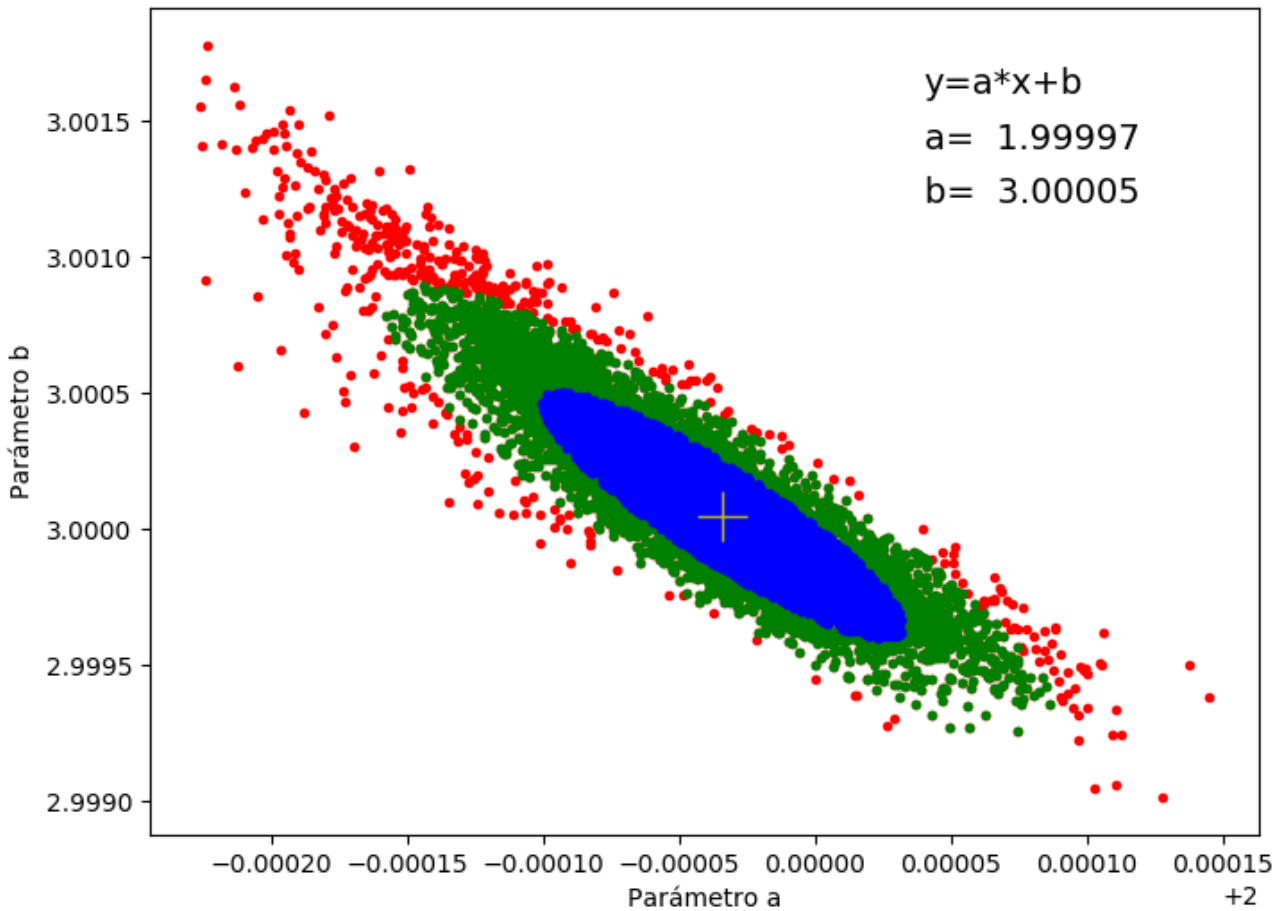
```
In [197]: va=np.zeros(10000)

In [198]: vb=np.zeros(10000)

In [199]: chi2=np.zeros(10000)

In [200]: for ii in range(10000):
     ...:   inx=np.random.randint(0,10000,10000)
     ...:   mx=xx[inx]
     ...:   me=er[inx]
     ...:   my=yy[inx]
     ...:
     ...:   ss=np.sum(1./me**2)
     ...:   sx=np.sum(mx/me**2)
     ...:   sy=np.sum(my/me**2)
     ...:   sxx=np.sum(mx**2/me**2)
     ...:   sxy=np.sum(mx*my/me**2)
     ...:
     ...:   dd=ss*sxx-sx**2
     ...:   aa=(ss*sxy-sx*sy)/dd
     ...:   bb=(sxx*sy-sx*sxy)/dd
     ...:   chisq=np.sum((yy-(aa*xx+bb))**2/er**2)
     ...:   va[ii]=aa
     ...:   vb[ii]=bb
     ...:   chi2[ii]=chisq-chisq0
     ...:
```

Intervalos de confianza:

Ordeno el vector con los valores de $(\chi^2 - \chi_0^2)$ y extraigo los que están dentro del 68,3 % y 95,5 % de la muestra:

```
In [201]: inx=np.argsort(chi2)

In [202]: aa683=va[inx[:6830]]

In [203]: bb683=vv[inx[:6830]]

In [204]: aa955=va[inx[:9550]]

In [205]: bb955=vb[inx[:9550]]
```

Intervalos de confianza:



Intervalos de confianza:

Si el ajuste del modelo se hace mediante SVD los errores formales del ajuste vienen dados en las matrices que produce la descomposición:

- la descomposición en valores singulares de una matriz A con dimensiones $N \times M$ y $N > M$ resulta en:

$$\mathbf{A} = \mathbf{U} * \begin{pmatrix} w_1 & \cdots & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & \cdots & w_N \end{pmatrix} * \mathbf{V^T}$$

donde $\mathbf{U}$ tiene las mismas dimensiones que $\mathbf{A}$, $\mathbf{W}$ y $\mathbf{V^T}$ son matrices cuadradas $M \times M$, siendo las columnas de $\mathbf{U}$ y $\mathbf{V^T}$, y las filas de $\mathbf{V^T}$, ortonormales:

$$\mathbf{U^T} * \mathbf{U} = \mathbf{V^T} * \mathbf{V} = \mathbf{V} * \mathbf{V^T} = \mathbf{1}$$

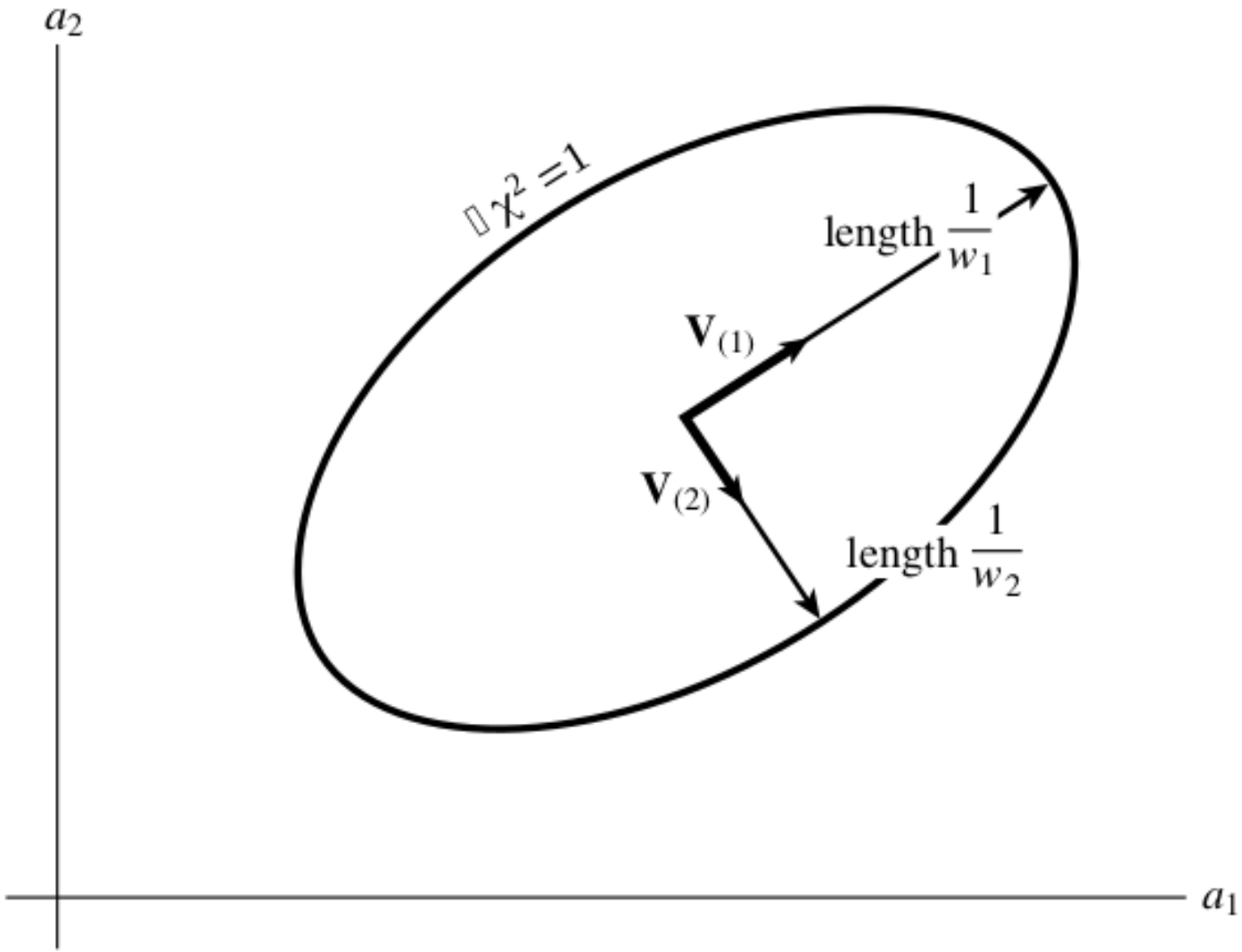
Intervalos de confianza:

- las columnas de $\mathbf{V}$ son un conjunto ortonormal de vectores que son los ejes principales de los elipsoides de $\delta\chi^2$ constante.
- Si llamamos $\mathbf{v_1}, \mathbf{v_2}, \cdots$ a las columnas de $\mathbf{V}$ y $\delta\mathbf{a}$ es una variación en los parámetros, tenemos que:

$$\Delta\chi^2 = w_1^2(\mathbf{v_1} * \delta\mathbf{a})^2 + \cdots + w_M^2(\mathbf{v_M} * \delta\mathbf{a})^2$$

- la longitud de los ejes de los elipsoides de $\delta\chi^2$ constante son inversamente proporcional a los correspondientes valores singulares $w_1, w_2, \cdots, w_M$.

Intervalos de confianza:



Covarianza:

La varianza mide la variación de una sola variable aleatoria:

$$\sigma_x^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

mientras que la covarianza mide en que medida dos variables aleatorias varían juntas:

$$Cov(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

De las expresiones anteriores resulta que $\sigma_x^2 = Cov(x, x)$.

Covarianza:

Si un modelo tiene parámetros $a_1, a_2, \cdots, a_M$, la matriz de covarianza $\mathcal{C}$ es una matriz cuadrada $M \times M$ cuyos elementos son:

$$C_{i,j} = Cov(a_i, a_j)$$

Esta matriz es simétrica ya que $Cov(a_i, a_j) = Cov(a_j, a_i)$ y en su diagonal están las varianzas de cada parámetro $Cov(a_i, a_i)$.

Si $Cov(a_i, a_j) = Cov(a_j, a_i) = 0$ se dice que los parámetros a_i y a_j son independientes entre sí. Si $Cov(a_i, a_j) = Cov(a_j, a_i) < 0$ los parámetros a_i y a_j están anti correlacionados.

Covarianza:

En el caso de descomposición en valores singulares los elementos de la matriz de covarianza se obtienen operando con la matriz **V** y los valores singulares w:

$$C_{j,k} = \sum_{i=1}^{M} \frac{1}{w_i^2} v_{ji} v_{ki}$$

En Numpy es sencillo calcular la matriz de covarianza utilizando numpy.cov(). Si tenemos tres parámetros con cinco mediciones para cada uno:

```
In [242]: va = [45,37,42,35,39]

In [243]: vb = [38,31,26,28,33]

In [244]: vc = [10,15,17,21,12]

In [245]: np.cov(np.array([va,vb,vc]),ddof=1)
Out[245]:
array([[ 15.8 ,   9.6 , -12.  ],
       [  9.6 ,  21.7 , -17.25],
       [-12.  , -17.25,  18.5 ]])
```

Covarianza:

En Pandas también es sencillo:

```
In [248]: import pandas as pd

In [249]: data = {'A': [45,37,42,35,39],
     ...:          'B': [38,31,26,28,33],
     ...:          'C': [10,15,17,21,12]
     ...:          }

In [250]: df = pd.DataFrame(data,columns=['A','B','C'])

In [251]: pd.DataFrame.cov(df)
Out[251]:
      A      B      C
A  15.8   9.60 -12.00
B   9.6  21.70 -17.25
C -12.0 -17.25  18.50
```

Actividades:

Práctica 9:

- Para los dos ajustes realizados en el práctico anterior calcule el intervalo de confianza proyectado y la varianza de cada uno de los parámetros del modelo.

Entrega

Para la próxima clase

Por consultas: ricardo.gil-hutton@conicet.gov.ar
Grupo de Ciencias Planetarias - CUIM 2