Chapter 3

List manipulation; curve fitting

2D projectile revisited

- $x(t) = x_0 + v_0 t \cos\theta; y(t) = y_0 + v_0 t \sin\theta + \frac{g}{2} t^2.$
- *T*: time of flight; *T* =-$2(y_0 + v_0 \sin\theta)/g.$
- $g = -9.81;$

2D projectile using list

- Revisit 2D projectile motion: Using **ListPlot** instead.
- Generate a list containing the coordinates of a 2D projectile for a time from 0 to T=-$2(y_0 + v_0 \sin\theta)/g$ at an time interval of Detalt=0.01T
- See sample code 1 and 2 in C3_listprojectile.nb.
- **list=Table[{x[t,theta],y[t,theta]},{t,0,T,Deltat}]** comprises of a list of the coordinates of the projectile at discrete values of *t*.
- Abstract information of this list: see sample code 3 in C3_listprojectile.nb
- **Syntax: Length[list]; list[[2]]; list[[2,1]]; list[[2,2]; list[[-1]]**

Manipulating list (case study)

- See C3_listprojectile.nb as a case study on how we manipulate the list of a 2D projectile to find the maximum values of y and the corresponding value of x at which ymax occurs.
- Syntax: **Max[];Min[];Sort;Ordering;**
- Exercise: Use your code to tell you when does ymax occurs.

Measuring speed in a simulation: case study 1

- As a case study, recycle your code for simulating a single pendulum.
- By creating a list for the locations of the pendulum at different time step,
- (*i*) “measure” the speed of the pendulum in each time step.
- (*ii*) Plot the speed of the pendulum as a function of time.
- (*ii*) Plot the square of speed of the pendulum as a function of displacement from the equilibrium position. You should obtain a quadratic relation between these two dynamical variables.
- See C3_measure_pendulum_speed.nb

Directory hopping, **Export**, **Import**

- Try to generate the coordinates of a semicircle (in the upper half of the x-y plane) of radius *R* that is centered at (0,0)
- **y[t_]:=R*Cos[t]; y[t_]:=R*Sin[t];**
- **datasemicircle=Table[{x[t],y[t]},{t,0,Pi,0.05Pi}];**
- **Export[“datasemicircle.dat”,datasemicircle];**
- **Import[“datasemicircle.dat”];**
- See C3_datasemicircle.nb.
- Syntax: **SetDirectory[];NotebookDirectory[];Import[];Export[];AspectRatio->1; FileExistsQ[]**

Simple curve fitting

- Download the data "datasemicircle.dat" online. It is supposed to have been generated by your friend who decline to disclose what value of *R* she used when generating the data, except notifying that the center of the circle was located at (0,0).
- Now, can you write a code to decipher what value of *R* she uses to generate the data?
- To this end, you need to quantify the error of the trial values of R with respect to the "true" R value of datasemicircle.dat.
- See C3_decipher_R_semicircle.nb

"Merit function"

- A circle centered at (*h*,*k*) with radius *R* is described by the equation

$$(x-h)^2+(y-k)^2=R^2$$

- Say you are given a set of coordinates of a circle with known center (*h*,*k*) but unknown radius, $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_i, y_i), \ldots, (x_N, x_N)\}$.
- How do we find out what value of R is exactly, using numerical method?
- Consider an equation based on the circle equation, defined as

$$\Delta^2=(R^2-[(x-h)^2+(y-k)^2])^2$$

- Since we do not know what the true value of *R* is, let's make a guess, say, *r*=1. Slotting the guessed value *r* the values of any pair of (*x*,*y*) values from the data set into the equation Δ^2, we have

$$\Delta^2(r,x_i,y_i)=\left(r^2-[(x_i-h)^2+(y_i-k)^2]\right)^2$$

- The function $\Delta^2(r,x_i,y_i)$ dictates the discrepancy between the guessed value *r* and the true value of *R* as contributed by the data point (x_i, y_i).
- With the guessed value *r*, $\Delta^2(r,x_i,y_i)$ generally does not equal zero for all *i*.

“Merit function”, cont.

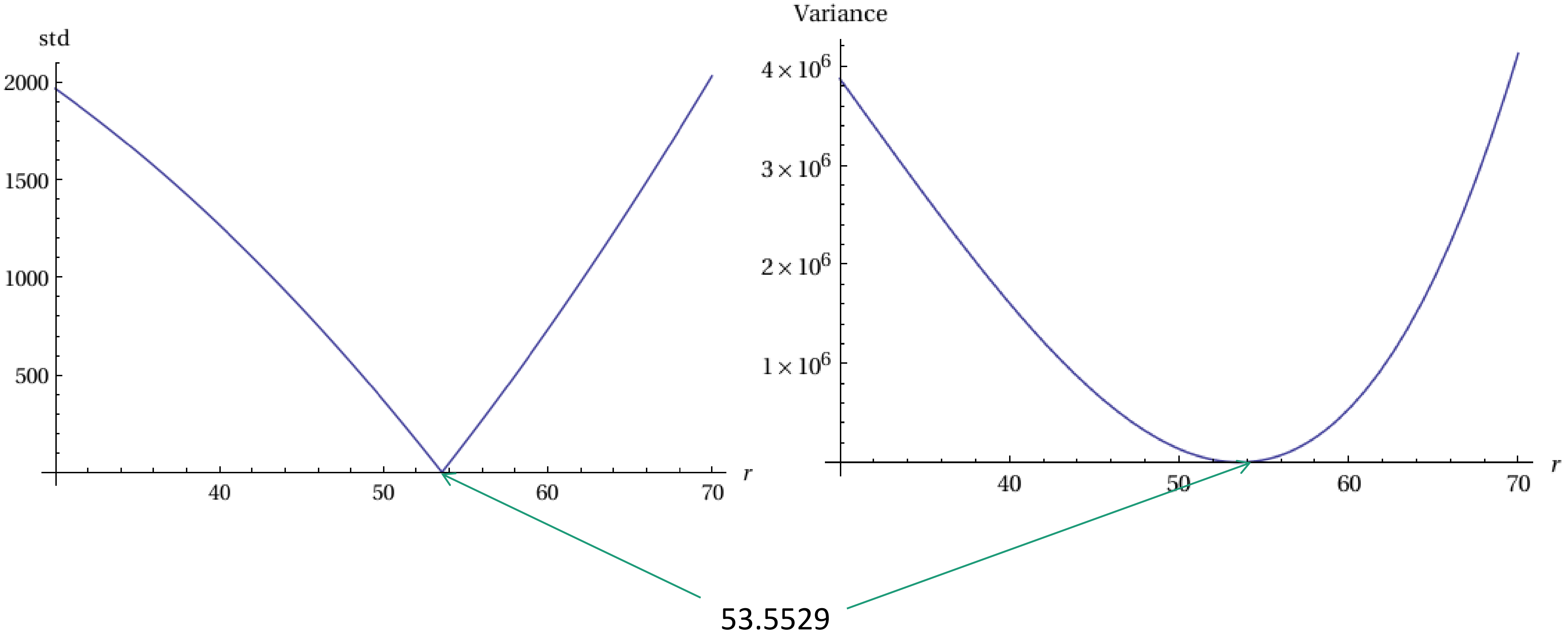
- Since every data point $\{x_i, y_i\}$ contribute differently to $\Delta^2(r, x_i, y_i)$, we should sum up these contribution. To these end, we define the variance and standard deviation,

$$\sigma^2(r) = \frac{\sum_{i=1}^{i=N} \Delta^2(r, x_i, y_i)}{N} = \frac{\sum_{i=1}^{i=N} (r^2 - [(x_i - h)^2 + (y_i - k)^2])^2}{N} \quad \text{(variance)}$$

$$\sigma(r) = \sqrt{\sigma^2(r)}, \quad \text{(standard deviation)}$$

- If the guessed value *r* happens to hit upon the “true” value *R*, then both the variance or standard deviation will become zero. Standard error and variance are both examples of “merit functions”, functions that when minimized will tell what the true value of *R* is.

Standard deviation and variance of a semicircle data with R= 53.5529



Minimise you merit function

- Merit function is system-dependent.
- To find out the values for the unknown parameter describing a set of data points, you must form a suitable merit function.
- Upon minimizing the merit function numerically, these unknown values could be obtained.

Exercise: Decipher the radius of a full circle

- Download the data "datacircle.dat" online. It is supposed to have been generated by your friend who decline to disclose what value of *R* she used when generating the data, except notifying that the center of the circle was located at (0,0).
- Now, modify the code decipher_R_semicircle.nb to decipher what value of *R* she uses to generate the data.
- Note: Be aware!! You got to redesign the function used to quantify the error between the trial value y and the true y.
- Solution: see C3_decipher_R_circle.nb.

Example of a two-variables curve fitting

- In previous exercises, you was fitting a data set with an equation with only a single unknown.
- Say if you were given a data set of a circle with a known radius *R* but unknown center (*h*,*k*), can you still able to write a code to figure out what values of (*h*,*k*) are?
- Download the data file, data_circle_unknown_center.dat. It contains data of a 2-D circle with radius *R*=0.75. The center of the circle (*h*,*k*) is not known. It was generated using the formula:

$$(x-h)^2+(y-k)^2=R^2$$

Or equivalently, in parametrized form,

$$x=R\cos\theta+h, y=R\sin\theta+k, \theta\in[0,2\pi].$$

Write a code to find out the (*h*,*k*). See C3_decipher_R_circle_2D.nb

The merit function for two unknown parameters, $\{h, k\}$.

$$(x-h)^2 + (y-k)^2 = R^2$$

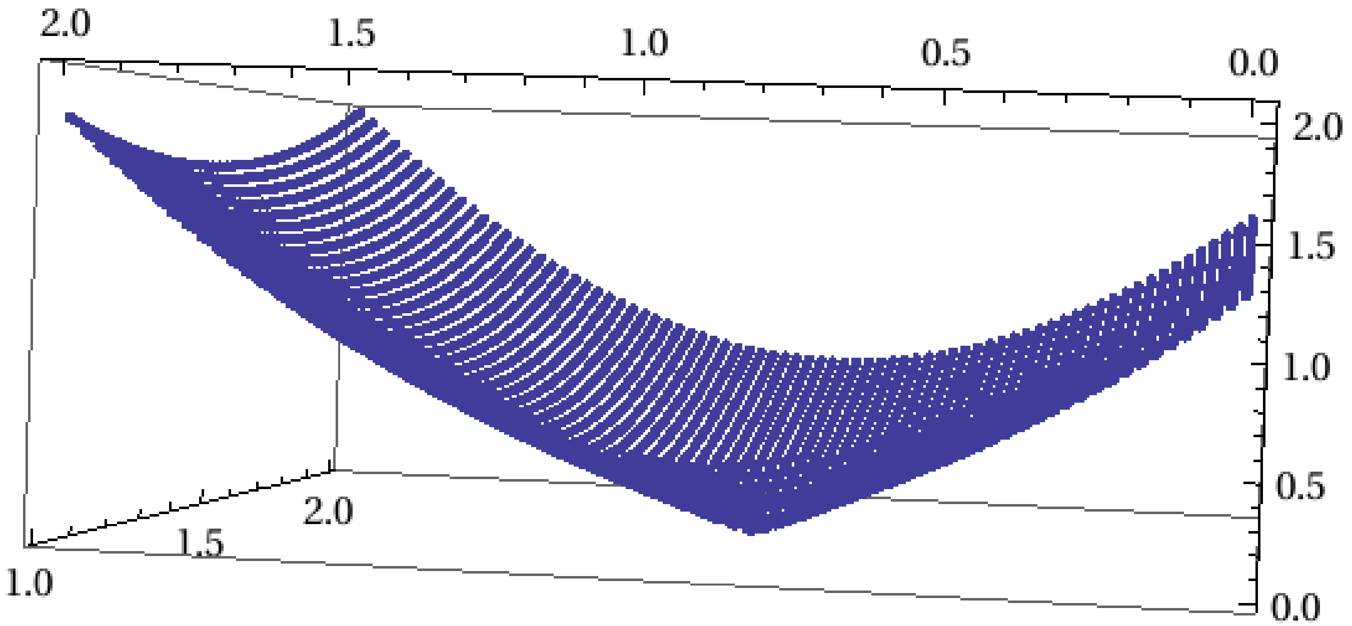
$$\Delta^2(h,k,x_i,y_i) = \left(R^2 - \left[(x_i-h)^2 + (y_i-k)^2\right]\right)^2$$

$$\sigma^2(h,k) = \frac{1}{N}\sum_i^N \Delta^2(h,k,x_i,y_i)$$

$$\sigma_s(h,k) = \sqrt{\sigma^2(h,k)}$$

In this case, you have to scan the two-dimensional parameter space spanned by *h* and *k* to search for which values of $\{h,k\}$ are such that σ^2 (or σ_s) is minimised.

Minimised standard deviation in (h,k) parameter space

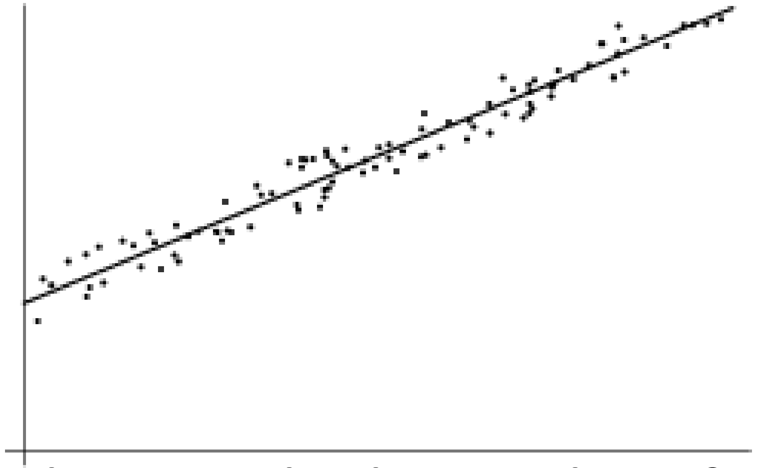


Least Squares Fitting

Least Squares Fitting

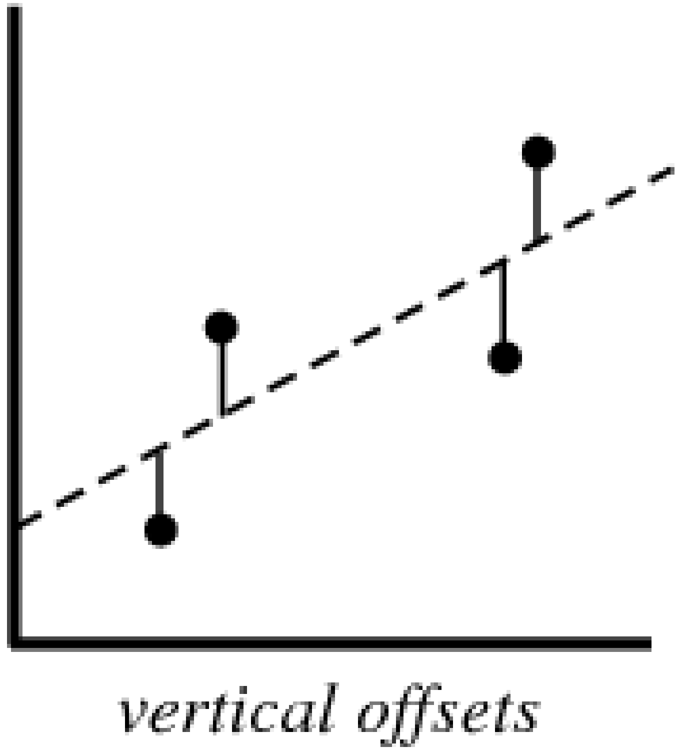
http://mathworld.wolfram.com/LeastSquaresFitting.htm

You have measured a set of data points, $\{x_i, y_i\}, i = 1,2, \dots, N$; and you know that they should approximately lie on a straight line of the form *y* = *a* x + *b* if the y_i's are plotted against x_i's.



We wish to know what are the best values for *a* and *b* that make the best fit for the data set.

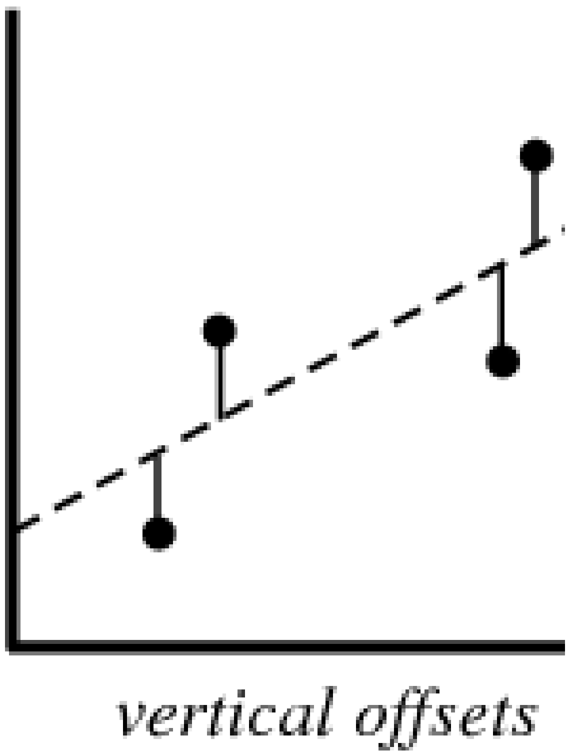
Vertical offset



Let $f(x_i, a, b) = b\,x_i + a$, in which we are looking for the best values for *b* and *a*.

Vertical least squares fitting proceeds by minimizing the sum of the *squares* of the *vertical* deviations R^2 of a set of *n* data points

Brute force minimisation



The values of a and b for which R^2 is minimized are the best fit values.

You can either find these best fit values using "brute force method", i.e. minimize R^2 by scanning the parameter spaces of a and b (see sample code C3_brute_force_linearfit.nb), or be smarter by using a more intelligent approach.

Least Squared Minimisation

$$R^2(a, b) \equiv \sum_{i=1}^{n} [y_i - (a + b\,x_i)]^2$$

$$\frac{\partial(R^2)}{\partial a} = -2 \sum_{i=1}^{n} [y_i - (a + b\,x_i)] = 0$$

$$\frac{\partial(R^2)}{\partial b} = -2 \sum_{i=1}^{n} [y_i - (a + b\,x_i)]\,x_i = 0.$$

$$n\,a + b \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$$

$$a \sum_{i=1}^{n} x_i + b \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i\,y_i.$$

In matrix form

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i\, y_i \end{bmatrix},$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i\, y_i \end{bmatrix}.$$

Eq. (1)

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \begin{bmatrix} \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i\, y_i \\ n \sum_{i=1}^{n} x_i\, y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i \end{bmatrix},$$

$$a = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i\, y_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$= \frac{\bar{y}\left(\sum_{i=1}^{n} x_i^2\right) - \bar{x} \sum_{i=1}^{n} x_i\, y_i}{\sum_{i=1}^{n} x_i^2 - n\,\bar{x}^2}$$

Eq. (2)

$$b = \frac{n \sum_{i=1}^{n} x_i\, y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

Eq. (3)

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

$$\bar{x} = \frac{1}{N}\sum_{i}^{N} x_i\,, \bar{y} = \frac{1}{N}\sum_{i}^{N} y_i$$

standard errors

$$\mathrm{SS}_{x\,x} = \sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad \mathrm{SS}_{x\,y} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

$$\mathrm{SS}_{y\,y} = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

$$s = \sqrt{\frac{\mathrm{ss}_{y\,y} - b\,\mathrm{ss}_{x\,y}}{n-2}} = \sqrt{\frac{\mathrm{ss}_{y\,y} - \frac{\mathrm{ss}_{x\,y}^2}{\mathrm{ss}_{x\,x}}}{n-2}}$$

$$\mathrm{SE}(a) = s\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\mathrm{ss}_{x\,x}}} \qquad \mathrm{SE}(b) = \frac{s}{\sqrt{\mathrm{ss}_{x\,x}}}. \qquad \text{Eq. (4,5)}$$

Exercise

Write a Mathematica code to calculate a, b and their standard errors
based on Eqs. (2,3,4,5).
Use the data file:
data_for_linear_fit.dat

Matrix Manipulation

The best values for a and b in the fitting equation can also be obtained by solving the matrix equation, Eq. (1).

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i \, y_i \end{bmatrix}.$$

Develop a Mathematica code to implement the matrix calculation.
See sample code: C3_least_sq_fit_matrix.nb.

Mathematica's built-in functions for data fitting

See sample code: C3_Math_built_in_linearfit.nb.

Syntax: **Take[]**

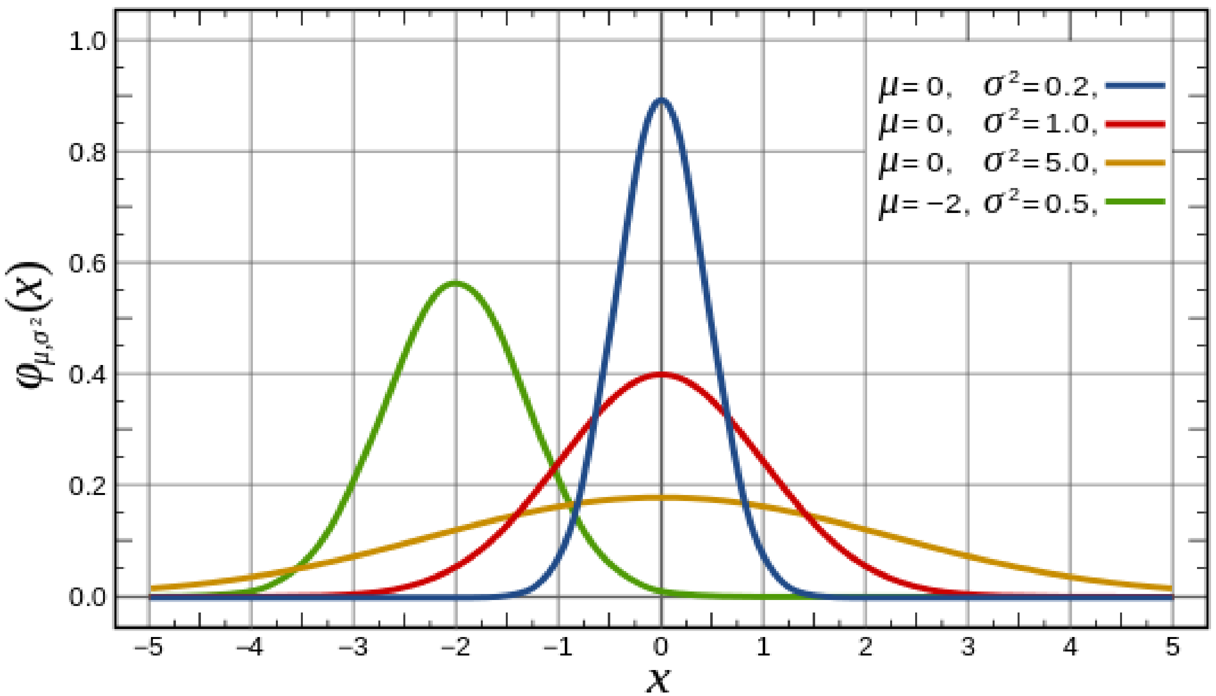
Syntax: **FindFit[ ]**, **LinearModelFit**,**Normal** "**BestFit**", "**ParameterTable**"

These are Mathematica's built in functions to fit a set of data against a linear formula, such as $y = a + b\,x$, and at the same time automatically provide errors of the best fit parameters – very handy way to fit a set of data against any linear formula.

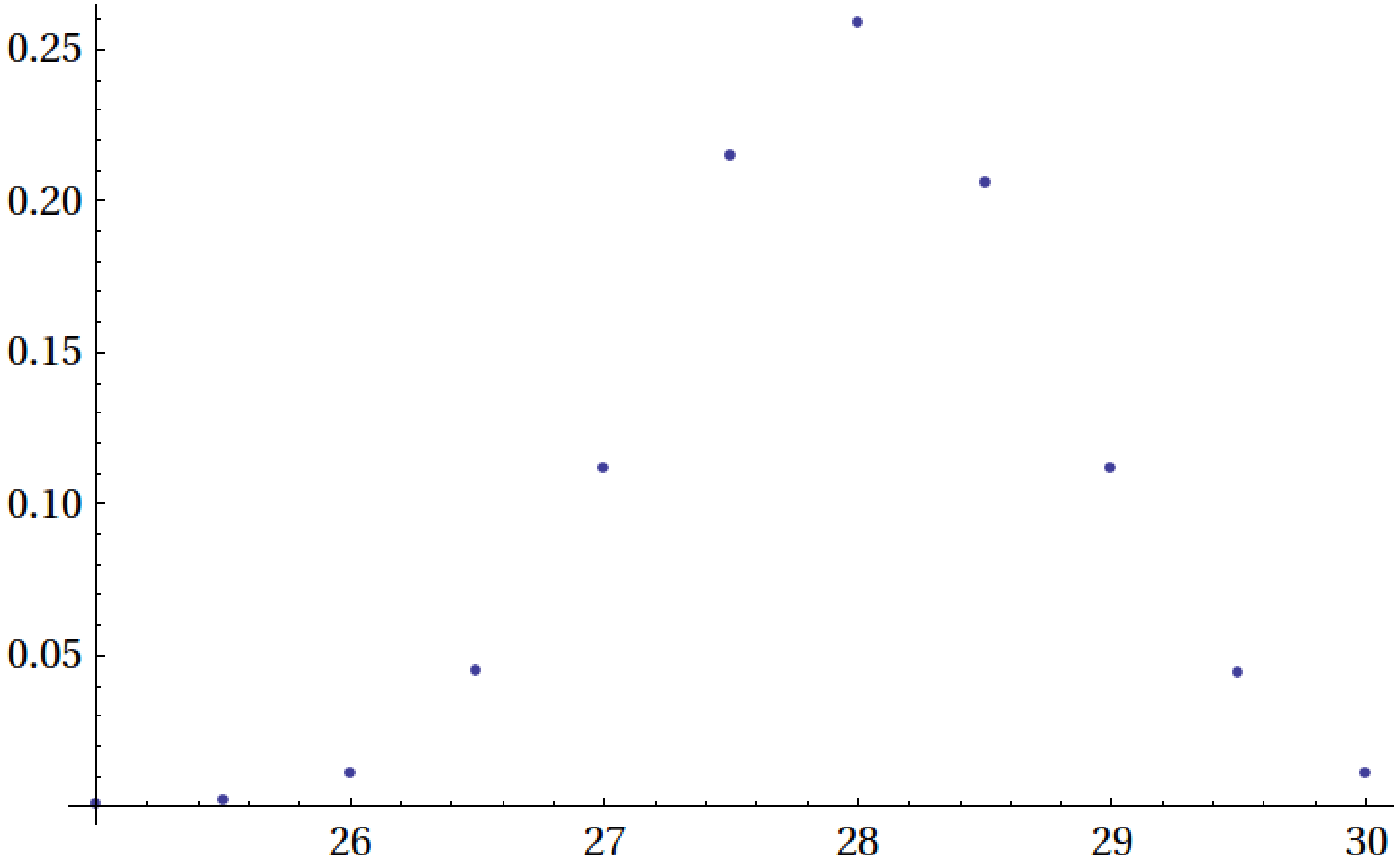
Gaussian function

A Gaussian function has the form:It $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}$,

It is parametrised by two parameters, μ (average) and σ (width, or squared root of variance).



Download and **ListPlot** the data file, gaussian.dat.



Interpolation[]

These data points can be automatically linked up by a best curve using the Mathematica built-in function **Interpolation.**

See sample code: C3_interpolation_gaussian_data.nb

NonlinearModelFit

Now, how would you ask Mathematica to find out the values of σ and μ that best fit the data point against a gaussian function?

Use **NonlinearModelFit**

See sample code: C3_nonlinearfit_gaussian.nb for the use of built-in function to fit a set of data points onto a non-linear function, such as the gaussian distribution.