Pragmatic demonstration of the CLT or, why errors are normally distributed

Note: before each case, start with "New experiment", and copy and paste the following lines into the command window. You can also do it line by line, or use the menu commands for display and fitting.

Case 1

```
// Even noise 0.995 to 1.005 10000 values binned in 100 bin centred
histogram
SetRandomSeed 0.1
Make/N=10000 fakeY=enoise(0.005)+1
Make/N=100/O fakeY_Hist;DelayUpdate
Make/N=100/O fakeY_Hist;DelayUpdate
Histogram/C/B=1 fakeY,fakeY_Hist
Display fakeY_Hist
ModifyGraph mode=1
SetAxis/A/N=1/E=1 left;DelayUpdate
SetAxis/A/N=1 bottom
```

Case 2

```
//Now the same, but two variables with even noise 0.995 to 1.005 multiplied
together

SetRandomSeed 0.1
Make/N=10000 fakeY1=enoise(0.005)+1
Make/N=10000 fakeY2=enoise(0.005)+1
Make/N=10000 fakeY=fakeY1*fakeY2
Make/N=100/O fakeY_Hist;DelayUpdate
Make/N=100/O fakeY_Hist;DelayUpdate
Histogram/C/B=1 fakeY,fakeY_Hist
Display fakeY_Hist
ModifyGraph mode=1
SetAxis/A/N=1/E=1 left;DelayUpdate
SetAxis/A/N=1 bottom
```

You can fit this already to a Gaussian function via the menu item Analysis – Curve fitting – tab Function and Data – Function - Gauss tab Coefficients y0 Hold, initial guess "0"
Or by typing:

```
K0 = 0;
CurveFit/H="1000"/TBOX=769 gauss fakeY_Hist /D
```

Then press "OK" to display fit. You can see it already is not far from Gaussian, except in the wings where the values descend more quickly to the baseline. Now let's try the same with six variables multiplied together:

Case 3

```
//Six variables with even noise 0.995 to 1.005 multiplied together with fit
to Gaussian
SetRandomSeed 0.1
Make/N=10000 fakeY1=enoise(0.005)+1
Make/N=10000 fakeY2=enoise(0.005)+1
Make/N=10000 fakeY3=enoise(0.005)+1
Make/N=10000 fakeY4=enoise(0.005)+1
Make/N=10000 fakeY5=enoise(0.005)+1
Make/N=10000 fakeY6=enoise(0.005)+1
Make/N=10000 fakeY=(fakeY1*fakeY2*fakeY3*fakeY4*fakeY5*fakeY6)
Make/N=100/O fakeY_Hist;DelayUpdate
Histogram/C/B=1 fakeY,fakeY_Hist
```

```
Display fakeY_Hist
ModifyGraph mode=1
SetAxis/A/N=1/E=1 left;DelayUpdate
SetAxis/A/N=1 bottom
K0 = 0;
CurveFit/H="1000"/TBOX=769 gauss fakeY_Hist /D
```

You can see now that with just 6 independently fluctuating variables the resulting product displays normally distributed or Gaussian noise. Real life experimental measurements fluctuate as a result of many independent error sources.

You can try the same exercise but instead of calculating the product, try calculating the sum fakeY=(fakeY1+fakeY2+fakeY3+fakeY4+fakeY5+fakeY6)
Does this work too ?

Finally, if you are feeling adventurous, you could try to write a procedure to calculate an arbitrary number (say, 100) for products or sums – you will need to use the Igor manual to help you with procedures.