Module 6
Doing least squares regression analysis in practice

In this session we going to make some final comments on the least squares regression fitting method of data

and we going to look at how we going really do this in real world.
using Computational tools like matlab & Python or R

Few Comments to make before we move on; in reality there are a huge number of solvers for nonlinear least squares problems

But we can observe, that if we do a Taylor series expansion for $\beta_1 x^2$ then, the 2nd term, the 2nd derivative is the Hessian.

which gives us information about the
curvature, or gradient of gradient,
gradient of the function
and therefore we can shoot directly from
where the function is zero, just as in NR,
using that 2nd derivative

Now using the Hessian will be faster, than simply
taking steps along the steepest descent
algorithm
Effectively, we would be using the Hessian
to give us a guess as to the size of the
step we should take in gradient descent

The problem is that often, the Hessian is
not very stable, especially for finding
the minimum
The L.-M. Marker method uses steepest
descent for finding minimum, and
then switches to using the Hessian

as it gets close to the minimum, based on the criteria of whether X^2 is getting better or not.

If its getting better, it uses the Hessian, if its in trouble, it uses steepest descent.

There are also the Gauss Newton method and BFGS method amongst many others that either use the Hessian directly or build up information about the Hessian over successive iterations

and depending on the convergence, different methods may be better than other

Robust fitting is another topic you should be aware of in case you need to look it up later

If we come back to Anscombe quartet, we see

Notice the bottom left data set in our problem, has just that 1 flying data point. A truly Robust fitting method, will be unbothered by such a data point

One approach to robust fitting, is minimizing, instead of the least squares, the absolute, or the square deviation

So it does not weight the points that are far away from line as strongly

It means it fits a little bit more visually, looks a bit better.

Now, lets turn to look at how you do this in real world.

In MathLab...

Matlab

Import data
Apps... Curve fitting toolbox

Also see histfit function

In Python is very nearly a simple:
Deals in the Scientific python, Scipi set of modules
the optimizer module, includes a least squares
fitting minimizer, curve_fit
"Scipy.optimize.curve_fit"...

So example provided in Course.

3 lines to do:
⇒ 2 lines to do the function
⇒ 1 line to do the fit.

```
# From https://docs.scipy.org/
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Define func of parameters a,b,c and variable x
def func(x, a, b, c):
    return a * np.exp(-b * x) + c

# generate some noisy data and plot it
xdata = np.linspace(0, 4, 50)
y = func(xdata, 2.5, 1.3, 0.5)
np.random.seed(1729)
y_noise = 0.2 * np.random.normal(size=xdata.size)
ydata = y + y_noise
plt.plot(xdata, ydata, 'b-', label='data')

#fit a,b,c and overplot
popt, pcov = curve_fit(func, xdata, ydata)
plt.plot(xdata, func(xdata, *popt), 'r-',
         label='fit: a=%5.3f, b=%5.3f, c=%5.3f' % tuple(popt))

# Constrain the optimization to the region of 0 <= a <= 3, 0 <= b <= 1 and 0 <= c <= 0.5
popt, pcov = curve_fit(func, xdata, ydata, bounds=(0, [3., 1., 0.5]))
plt.plot(xdata, func(xdata, *popt), 'g--',
         label='fit: a=%5.3f, b=%5.3f, c=%5.3f' % tuple(popt))

#add labels and show the plot
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

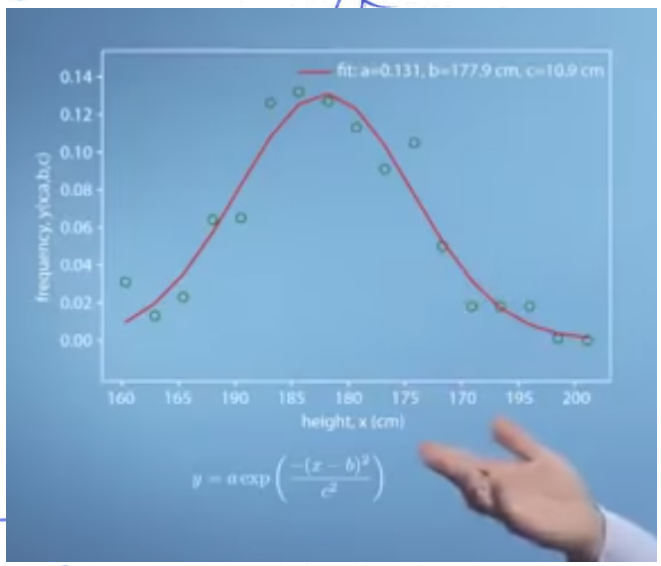
Rest is about plotting, and importing the data

```
def func(x, a, b, c):
    return a * np.exp(-b * x) + c

popt, pcov = curve_fit(func, xdata, ydata)
```

So only a few lines in python to bring it to life

Now, into a python Code block, to plot the Gaussian distribution Shown here (pic)



You will need to give it a Starting guess, and we will give you the input data for the height distribution in the population

It's important to note here, why we need to have a Starting guess

If we started with a guess here for mean of 100cm, the model curve would not overlap with the data at all.

So when we did a little more to it, we get no change.

and therefore the gradient of χ^2 w.r.t mean b would be zero

So the algorithm would not know what direction to go in

we would not get a sensible answer, for the function or for grad

and therefore our algorithm would not know where to go to find the minimum.

So in doing any of the line data fitting, it's vital to come up with a good means for generating a starting guess

like its easy, you pick the biggest value

So what we have done in the session, we have finished our little discussion in using vectors and multivariate calculus together to help us do optimizations of functions and to fit data to functions

~~Thread~~ turned out to be very easy computation.

Remember in Python, it's just a few lines to achieve what we have learned in these sessions.

But Now, you understand something of how those algorithms work under the hood.

That means we will be much better off at figuring out how to fix them when they go wrong