Eric Doviak
original: 22 Feb 2014

Quantitative Analysis
OLS regression

The notebook uses Ordinary Least Squares to provide an example of optimization with a single variable.

First, load the "distrib" package and turn off unnecessary printing.

```
(%i1) load(distrib) $
      ratprint: false $
```

Now, randomly draw 20 values from the standard normal.

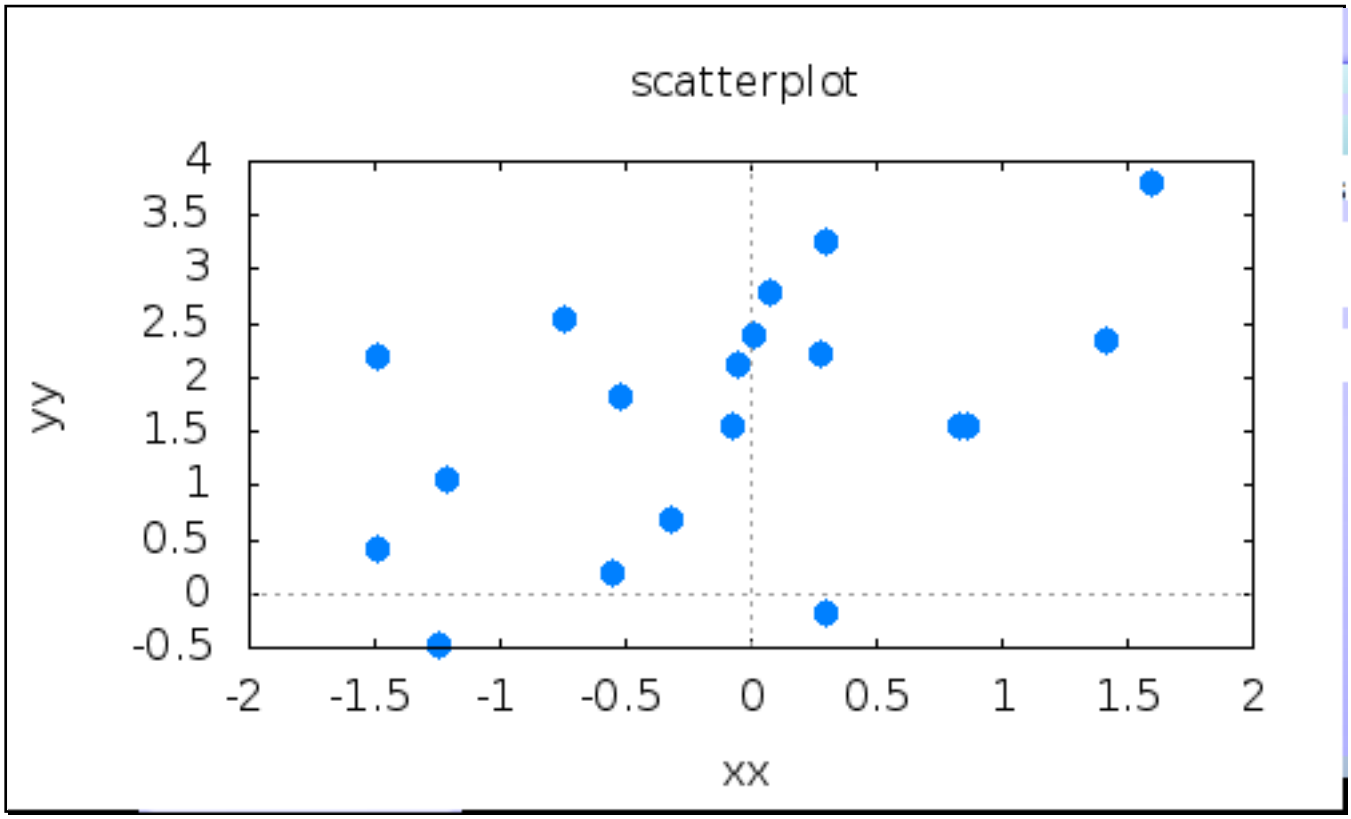
y = alpha + beta*x + u

let the "true" parameter values be: alpha=2 , beta=1

```
(%i3) N:20$
      xx:random_normal(0,1,N)$
      uu:random_normal(0,1,N)$
      yy:2+1*xx+uu$
```

```
(%i7) print("")$
      wxplot2d([discrete,xx,yy],[x,-2,2],[style,points],
               [xlabel,"xx"],[ylabel,"yy"],
               [gnuplot_preamble,"set title 'scatterplot'"])$
      print("")$
```

(%t8)



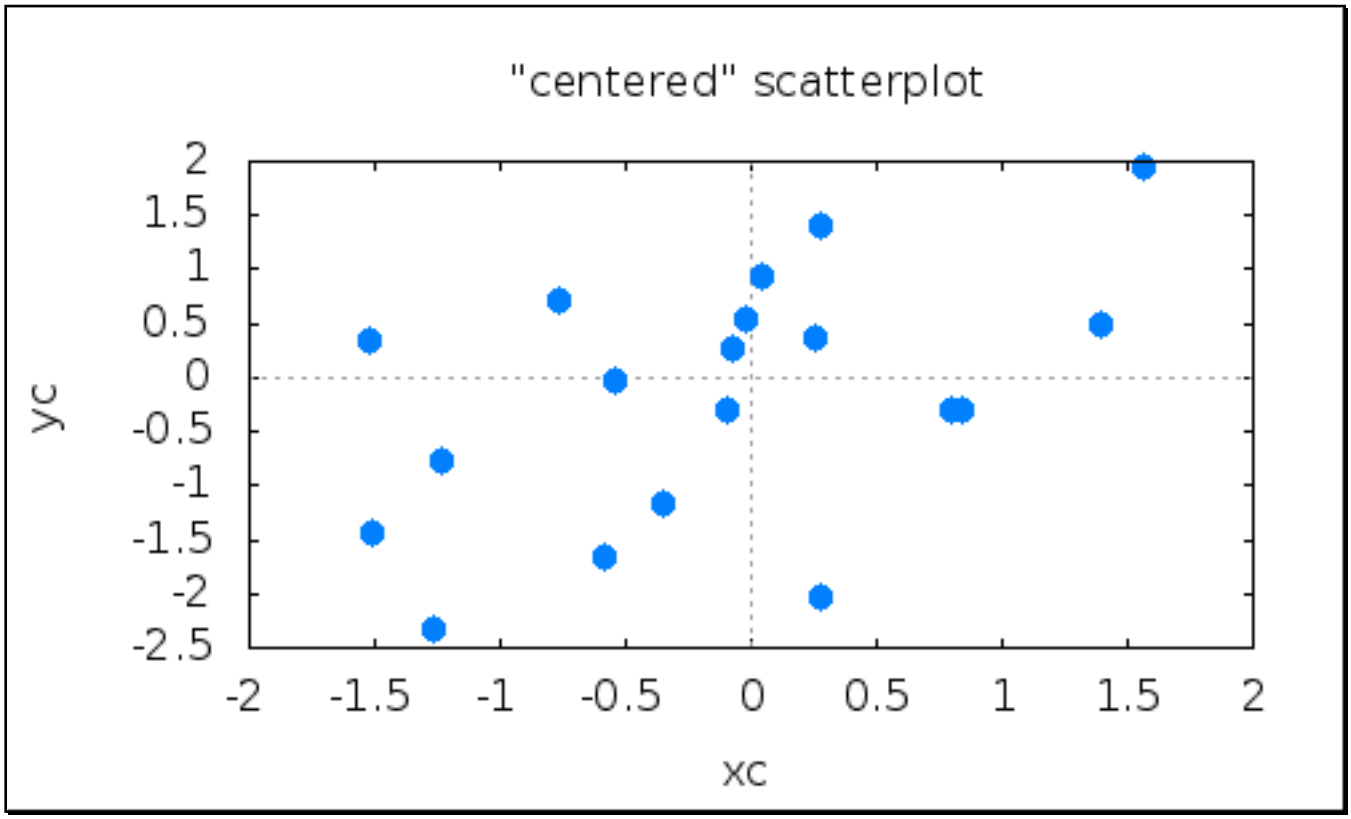

```
To estimate the intercept and slope (i.e. "alpha" and "beta"),
OLS minimizes the Sum of Squared Errors (SSE).

We'll use the simpler, but equivalent approach of subtracting
the means from "x" and "y," which sets the intercept to zero.
We will then use our estimate of "beta" to calculate the
estimate of "alpha."

       yy  =   alpha  +  beta*xx         +  uu
     - ym  =  -alpha  -  beta*xm         -  um
     ----     ----------------------------------------
(yy - ym)  =             beta*(xx - xm)  +  (uu - um)


where "ym," "xm" and "um" are the means of "yy," "xx" and "uu"
```

```
For convenience, let's define:
    yc  ==  (yy - ym)
    xc  ==  (xx - xm)
    uc  ==  (uu - um)

so that:
    yc  =  beta*xc + uc

Assuming that the error terms average to zero (i.e. um = 0),
we can calculate the estimate of "alpha" as:

    alpha  = ym - beta*xm
```

```
(%i10) ym : mean(yy)  $  xm : mean(xx)  $  um : mean(uu)  $
       yc : yy - ym   $  xc : xx - xm   $  uc : uu - um   $
```

```
(%i16) print("")$
       wxplot2d([discrete,xc,yc],[x,-2,2],[style,points],
                [xlabel,"xc"],[ylabel,"yc"],
                [gnuplot_preamble,"set title '\"centered\" scatterplot'"])$
       print("")$
       print("Because we subtracted means, our estimate of the intercept")$
       print("will be equal to zero.")$
       print("")$
```

(%t17)

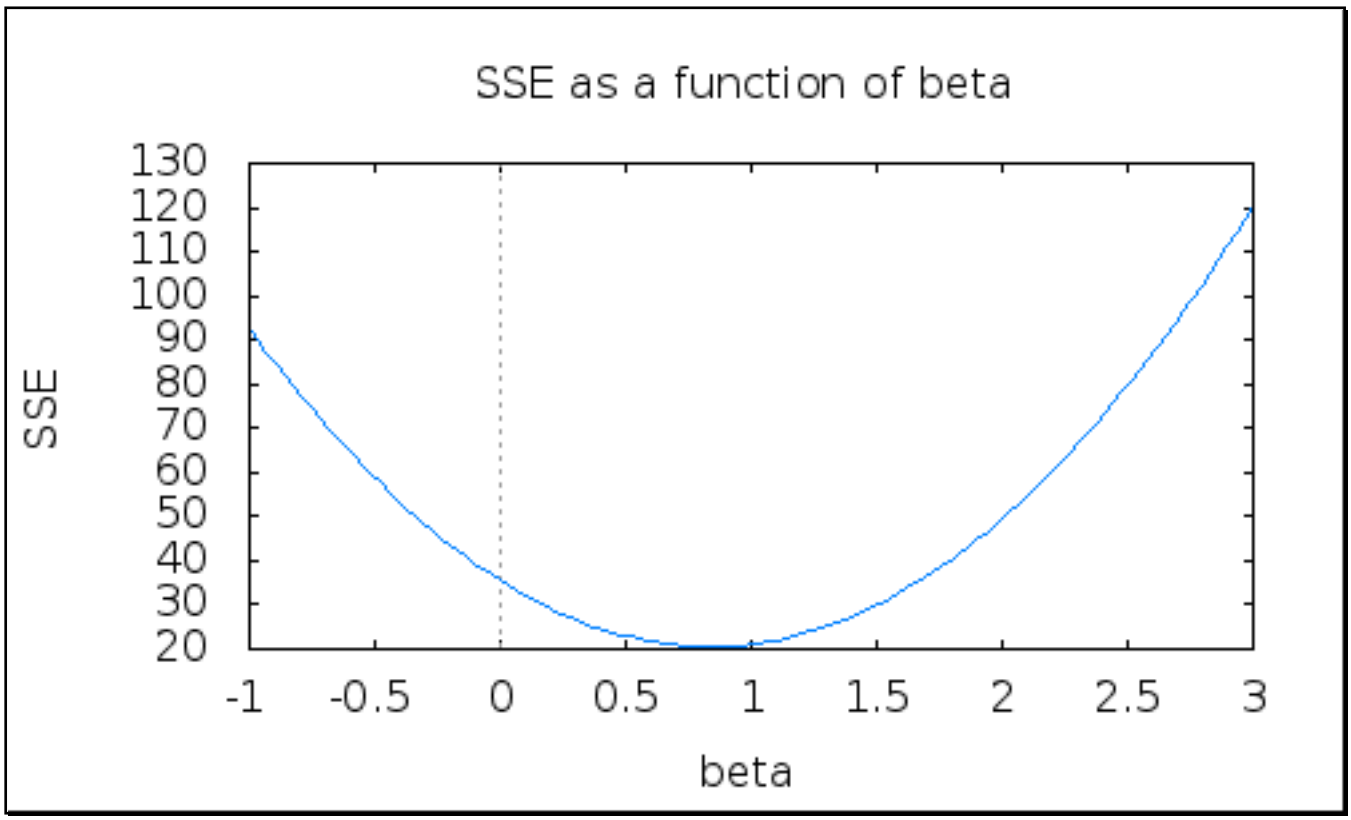


Because we subtracted means, our estimate of the intercept will be equal to zero.

```
We only observe "xx" and "yy." We do not observe the errors.
So let's plot the Sum of Squared Errors as a function of "beta."

Rearranging terms yields the following expression for the
"centered" errors:

     uc   =  yc - beta*xc

Squaring both sides:

    uc^2  =  (yc - beta*xc)^2
          =   yc^2  - 2*beta*xc*yc + (beta^2)*(xc^2)
```

```
(%i22) sse(beta) := sum( (yc[i]-(beta*xc[i]))^2 ,i,1,N)$

       betahat : lbfgs(sse(beta), [beta], [1.0], 1e-4, [-1,0]) $
       betahat : subst( betahat[1] , beta) $

       alphahat : ym - betahat*xm $

       print("")$
       wxplot2d(sse(beta),[beta,-1,3],[xlabel,"beta"],[ylabel,"SSE"],
                [gnuplot_preamble,"set title 'SSE as a function of beta'"])$
       print("")$
       print("SSE minimized when beta = ", round(100*betahat)/100.0,
              " (the \"true\" value was 1)")$
       print("which implies that alpha = ", round(100*alphahat)/100.0,
             " (the \"true\" value was 2)")$
       print("")$
```

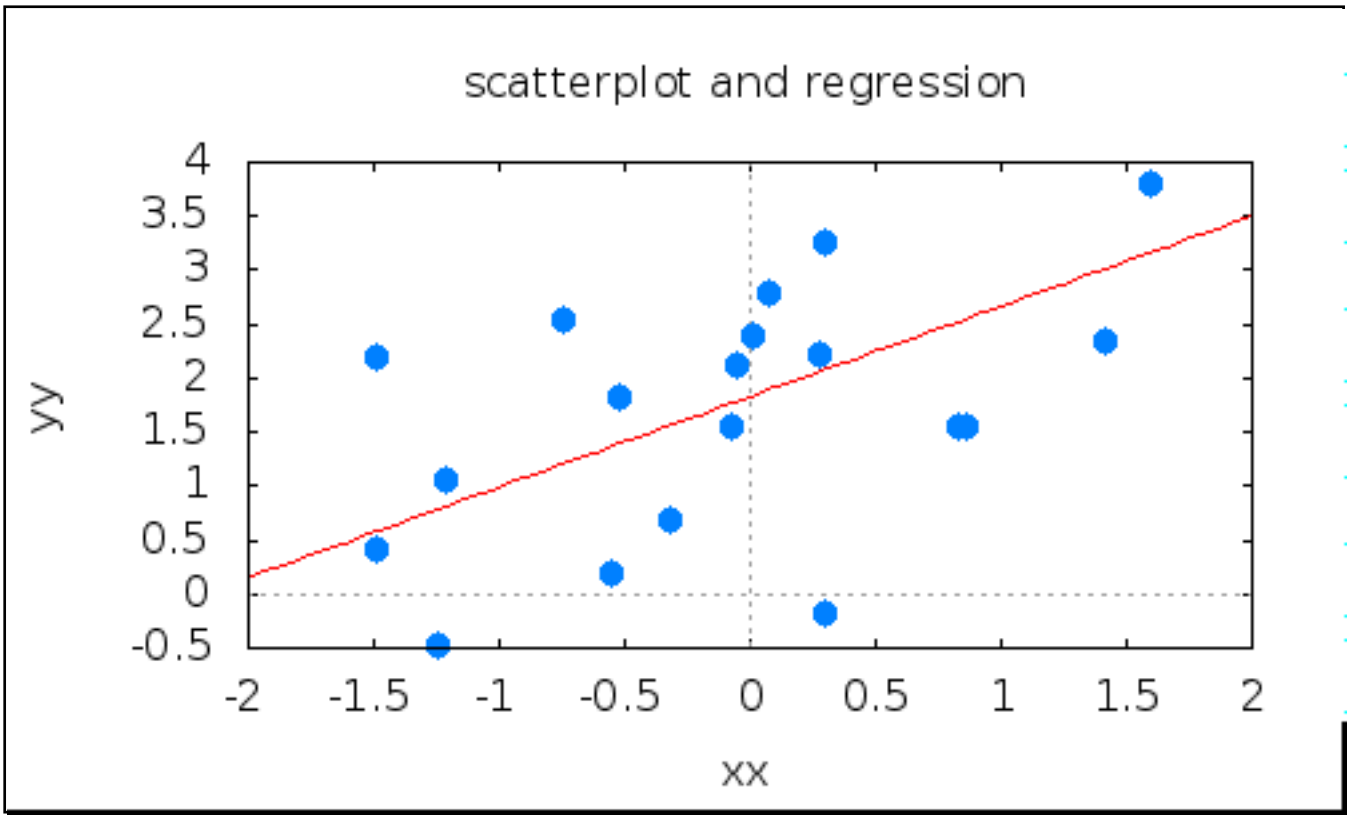
(%t27)



SSE minimized when beta = 0.84 *(the "true" value was 1)*
which implies that alpha = 1.83 *(the "true" value was 2)*

```
(%i32) ols(x) := alphahat + betahat*x $

       print("")$
       wxplot2d([[discrete,xx,yy],ols(x)],[x,-2,2],[style,points,lines],
                [xlabel,"xx"],[ylabel,"yy"],[legend,false],
                [gnuplot_preamble,"set title 'scatterplot and regression'"])$
       print("")$
```

(%t34)




```
Now let's use calculus to obtain the same result. Recall that:

         uc^2  =       yc^2  - 2*beta*xc*yc + (beta^2)*(xc^2)

Summing both sides:

    sum(uc^2)  =  sum( yc^2  - 2*beta*xc*yc + (beta^2)*(xc^2) )

Now, take the derivative with respect to beta.
```

```
(%i36) /* skip straight to the result */
       newbetahat : sum(xc[i]*yc[i],i,1,N) / sum(xc[i]^2,i,1,N) $

       /* now do the work */
       kill(xc,yc,N)$
       dsse(beta) := diff(sse(beta),beta)$

       print("")$
       print(("d SSE"/"d beta")," = ",dsse(beta))$
       print("")$
       print("At the minimum, the slope is zero, which implies that:")$
       print("")$
       print("beta_hat = ", (sum(xc[i]*yc[i],i,1,N)/sum(xc[i]^2,i,1,N)))$
       print("")$
       print("the formula above estimates beta at: ",
             round(100*newbetahat)/100.0)$
       print("which is equal to our old estimate: ",
             round(100*betahat)/100.0)$
       print("")$
```

$$\frac{d\ SSE}{d\ beta} = -2 \sum_{i=1}^{N} xc_i \left(yc_i - \beta\, xc_i\right)$$

At the minimum, the slope is zero, which implies that:

$$beta\_hat = \frac{\sum_{i=1}^{N} xc_i\, yc_i}{\sum_{i=1}^{N} xc_i^2}$$

the formula above estimates beta at: 0.84
which is equal to our old estimate: 0.84