Manipulating the Regression Line with Xlisp-Stat

Jan de Leeuw
Jason Bond

November 7, 1994

Contents

Abstract

Two classical didactic devices for illustrating least squares fit of a regression line are programmed in Xlisp-Stat. The first one allows the student to move a candidate regression line around in the plot, while program updates the residuals both graphically and numerically. As an additional bonus we have incorporated different ways of weighting the axis associated with the errors-in-variables approach to least squares regression. Thus the program also illustrates the fact that least squares can be defined in various ways, with as a consequence different straight line fits. The second didactic device illustrates the effect of moving a single point around on the least squares regression line. This can be used to illustrate the notions of outliers and influence.

Chapter 1

Choosing the Loss

1.1 Introduction

Consider the following situation. We have a scatterplot of n point-pairs (x_i, y_i) and we want to draw a straight line $y = \alpha + \beta x$ in the plot which describes these points "as good as possible". We do this by defining a loss function $\Delta(\alpha, \beta)$ which is minimized over α and β. Many different choices of loss functions are possible. In this Xlisp-Stat module we illustrate a family of weighted least squares loss functions, associated with the errors-in-variables model.

1.2 Errors-in-variables

Consider the model

$$\begin{aligned} \underline{x} &= \underline{\chi} + \underline{\delta}, \qquad (1.1a)\\ \underline{y} &= \alpha + \beta\underline{\chi} + \underline{\epsilon}. \qquad (1.1b) \end{aligned}$$

We follow the *Dutch Convention* of underlining random variables. We also suppose that $\mathbf{E}(\underline{\delta}) = \mathbf{E}(\underline{\epsilon}) = 0$, and we define $\mathbf{E}(\underline{\chi}) = \mu$. Moreover the *disturbances* $\underline{\delta}$ and $\underline{\epsilon}$ are supposed to be independent of each other and independent of $\underline{\chi}$. Let us use the method of moments to try and find estimates of the parameters.

It follows that

$$\begin{aligned} \mathbf{E}(\underline{x}) &= \mu \qquad (1.2a)\\ \mathbf{E}(\underline{y}) &= \alpha + \beta\mu. \qquad (1.2b) \end{aligned}$$

Thus if we know β, the parameters μ and α can be computed uniquely from the first order moments, i.e. they are *identified*. It remains to identify β.

Define

$$\begin{aligned} \mathbf{V}(\underline{\chi}) &= \lambda^2, \qquad (1.3a)\\ \mathbf{V}(\underline{\delta}) &= \omega^2, \qquad (1.3b)\\ \mathbf{V}(\underline{\epsilon}) &= \theta^2. \qquad (1.3c) \end{aligned}$$

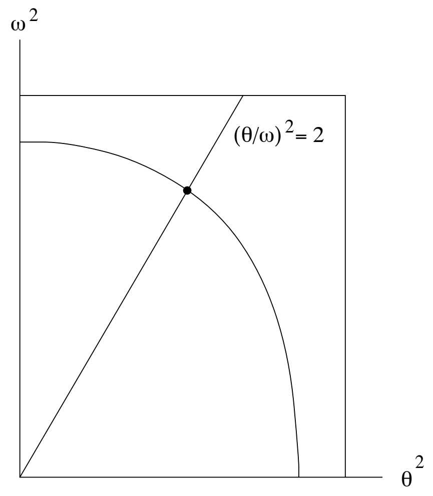


Figure 1.1: Frisch Hyperbola: $(\mathbf{V}(\underline{x}) - \omega^2)(\mathbf{V}(\underline{y}) - \theta^2) = \mathbf{C}(\underline{x},\underline{y})^2$

Then

$$\begin{aligned} \mathbf{V}(\underline{x}) &= \lambda^2 + \omega^2, & (1.4a)\\ \mathbf{V}(\underline{y}) &= \beta^2\lambda^2 + \theta^2, & (1.4b)\\ \mathbf{C}(\underline{x},\underline{y}) &= \beta\lambda^2. & (1.4c) \end{aligned}$$

The system 1.4 defines three equations in four unknowns. In general, there is an infinite number of solutions. We can picture them in (ω,θ) space, more precisely in the rectangle $\mathcal{R}$ defined by $0 \leq \omega^2 \leq \mathbf{V}(\underline{x})$ and $0 \leq \theta^2 \leq \mathbf{V}(\underline{y})$. Some, but not all, points in $\mathcal{R}$ correspond with solutions for β.

In fact, we see that

$$\beta = \frac{\mathbf{C}(\underline{x},\underline{y})}{\mathbf{V}(\underline{x}) - \omega^2} = \frac{\mathbf{V}(\underline{y}) - \theta^2}{\mathbf{C}(\underline{x},\underline{y})}, \tag{1.5}$$

which implies

$$\frac{\mathbf{C}(\underline{x},\underline{y})}{\mathbf{V}(\underline{x})} \leq \beta \leq \frac{\mathbf{V}(\underline{y})}{\mathbf{C}(\underline{x},\underline{y})}. \tag{1.6}$$

This is interesting, because it shows that β is bounded by the ordinary least squares regression coefficients for the regression of x and y and the regression of y on x.

It also follows that a pair (ω,θ) in the rectangle $\mathcal{R}$ corresponds with a solution to system 1.4 if and only if

$$(\mathbf{V}(\underline{x}) - \omega^2)(\mathbf{V}(\underline{y}) - \theta^2) = \mathbf{C}(\underline{x},\underline{y})^2, \tag{1.7}$$

which is known as the *Frisch hyperbola*. It defines a single equation in two unknowns. Thus all solutions are on the curved line in the rectangle in

Figure 1.1. Equation 1.7 can also be written as the determinantal equation

$$\begin{vmatrix} \mathbf{V}(\underline{x}) - \omega^2 & \mathbf{C}(\underline{x}, \underline{y}) \\ \mathbf{C}(\underline{x}, \underline{y}) & \mathbf{V}(\underline{y}) - \theta^2 \end{vmatrix} = 0. \tag{1.8}$$

If we want to find the intersection of the Frisch hyperbola with the line $\theta^2 = \eta\omega^2$, we must solve

$$\begin{vmatrix} \mathbf{V}(\underline{x}) - \omega^2 & \mathbf{C}(\underline{x}, \underline{y}) \\ \mathbf{C}(\underline{x}, \underline{y}) & \mathbf{V}(\underline{y}) - \eta\omega^2 \end{vmatrix} = 0, \tag{1.9}$$

It follows that (ω^2, β) is an eigenvalue-eigenvector pair of

$$\begin{pmatrix} \mathbf{V}(\underline{x}) & \mathbf{C}(\underline{x}, \underline{y}) \\ \mathbf{C}(\underline{x}, \underline{y}) & \mathbf{V}(\underline{y}) \end{pmatrix} \begin{pmatrix} \beta \\ -1 \end{pmatrix} = \omega^2 \begin{pmatrix} 1 & 0 \\ 0 & \eta \end{pmatrix} \begin{pmatrix} \beta \\ -1 \end{pmatrix} \tag{1.10}$$

Since the largest eigenvalue will be strictly larger than $\mathbf{V}(\underline{x})$, at least when $\mathbf{C}(\underline{x}, \underline{y}) \neq 0$, we need the smallest eigenvalue.

1.3 Weighted Least Squares

Let us consider a more geometrical approach. Define the loss function

$$\Delta(\alpha, \beta) = \min_{\chi_i} \frac{\frac{1}{n}\sum_{i=1}^n (x_i - \chi_i)^2}{\omega^2} + \frac{\frac{1}{n}\sum_{i=1}^n (y_i - \alpha - \beta\chi_i)^2}{\theta^2} \tag{1.11}$$

and minimize this over α and β.

Computing $\Delta(\alpha, \beta)$ means projecting (x_i, y_i) on the line $\alpha + \beta\chi$. The solution is

$$\begin{pmatrix} \hat{\chi}_i \\ \alpha + \beta\hat{\chi}_i \end{pmatrix} = \xi \begin{pmatrix} x_i \\ \alpha + \beta x_i \end{pmatrix} + (1 - \xi) \begin{pmatrix} \frac{y_i - \alpha}{\beta} \\ y_i \end{pmatrix}, \tag{1.12}$$

where

$$\xi = \frac{\theta^2}{\theta^2 + \beta^2\omega^2}. \tag{1.13}$$

Thus we see (compare Figure 1.2) that we project the point both horizontally and vertically on the line, and then interpolate. For a given line all projections are parallel.

It also follows from 1.12 that

$$\frac{(x_i - \chi_i)^2}{\omega^2} = \frac{\beta^2\omega^2(y_i - \alpha - \beta x_i)^2}{(\theta^2 + \beta^2\omega^2)^2}, \tag{1.14a}$$

$$\frac{(y_i - \alpha - \beta\chi_i)^2}{\theta^2} = \frac{\theta^2(y_i - \alpha - \beta x_i)^2}{(\theta^2 + \beta^2\omega^2)^2}. \tag{1.14b}$$

If we substitute this in 1.11 we find

$$\Delta(\alpha, \beta) = \frac{\frac{1}{n}\sum_{i=1}^n (y_i - \alpha - \beta x_i)^2}{\theta^2 + \beta^2\omega^2}. \tag{1.15}$$

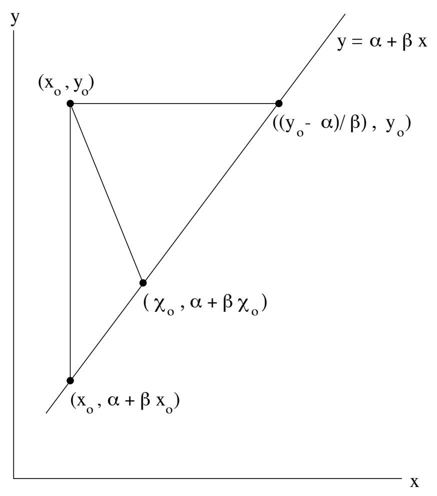


Figure 1.2: Interpolation of Regression Projections

The minimum for α is attained at $\hat{\alpha} = \overline{y} - \beta\overline{x}$, as usual, with $\overline{y}$ and $\overline{x}$ denoting sample means. This means we must still minimize

$$\Delta(\bullet, \beta) = \frac{S_{yy} - 2\beta S_{xy} + \beta^2 S_{xx}}{\theta^2 + \beta^2\omega^2} \tag{1.16}$$

over β. Here the S_{xx}, S_{xy}, S_{yy} are the sample variances and covariance.

Equation 1.16 is ratio of two quadratics. The minimum is given by the smallest eigenvalue of the matrix

$$\begin{pmatrix} \frac{S_{yy}}{\theta^2} & -\frac{S_{xy}}{\omega\theta} \\ -\frac{S_{xy}}{\omega\theta} & \frac{S_{xx}}{\omega^2} \end{pmatrix}, \tag{1.17}$$

and β can be found from the corresponding eigenvector.

Chapter 2

Pulling the Line

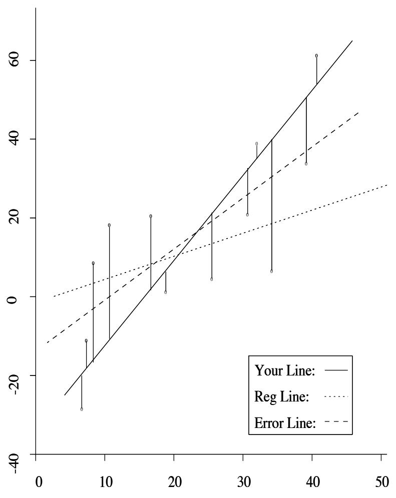
2.1 Introduction

In most undergraduate statistics courses, it is rarely explained what makes the Least Squares line "the best" line or even that there exist different best lines according to different optimization criteria. The usual regression line that students are introduced to is the regression line of y on x where the sum of squared vertical distances is the subject of minimization. The "pull line" program attempts to introduce students to a generalized definition of the regression line in a graphical environment.

The first of two illustrative devices in this paper graphically displays the effect of altering regression parameters on a regression line of the users choice. The user is also allowed to "grab" a "pull line", $y = a+bx$ and change the slope and intercept while residuals that are graphically projected onto this line are dynamically updated. The residuals that are to be updated and the specific regression line that the user chooses are dependent upon the values of $\mathbf{V}(\underline{\delta}) = \omega^2$ and $\mathbf{V}(\underline{\epsilon}) = \theta^2$. These parameters may also be changed by the user with the constraint that $k = \frac{\omega^2}{\omega^2+\theta^2}$. So the possible values of k lie in the interval $(0, 1)$. With ω^2 and θ^2 uniquely determined, β and α, the slope and intercept of the corresponding regression line are also uniquely determined. Equation 1.12 gives us $(\chi_i, a + b\chi_i)$, the projection onto the "pull line" line $y = a + bx$ and thus the residuals.

The plot menu provides the user with the option of plotting the regression line corresponding to any value of k. Sums of squares from equation 1.15 that are associated with this regression line as well as with the arbitrary "pull line" are displayed along with both line equations. Another regression line corresponding to a different value of k may be added to the plot in order to compare with the present regression line. This is accomplished by clicking on the menu button in the graph window and dragging down to the "Use Errors in Regressors" button. A slider will pop up and the user may change the value of k in increments of $1/100$.

The user may also supply any value of k in order to change the direction



```
Your  k-value is:   1e-11
Your Line Equation:   -35 + 2.64286x
Your Rss:  7.9534e-09
--------------------------
The LS k-value is:  .23
The LS Line:  -3.09056 +  .900182x
The LS RSS:   23.562
--------------------------
The Error K-value:  .98
The Error Line:  -17.6003 + 2.03582x
```

Figure 2.1: Pull Line Sample Screen and Output

in which the observed points are projected onto the "pull line". An example of a pull line session can be seen in Figure 2.1

One possible application of this program would be to allow a student to plot a regression line according to an arbitrary value of k, say $k = 1/5$. The value of k that determines the point of projection could be changed to this same value, $k = 1/5$ and as the "pull line" is moved around, it becomes apparent that the sums of squares is smallest when the "pull line" coincides with the plotted regression line.

2.2 The "Pull Line" Code

Once two lists x and y have been defined, a scatterplot is created where the line pulling will take place. The functions

```
(defun pull-line (x y &key (same nil))
 (setf pp (plot-points x y :title "Pull My Line"))
   (let* (
          (nn (send pp :num-points))
          (xx (send pp :scaled-range 0))
          (xn (send pp :x-axis))
          (yy (send pp :scaled-range 1))
          (k .00000000001)
         )

     (send pp :draw-mode 'xor)
     (send pp :add-slot 'reg-line)
     (send pp :add-slot 'your-coefs)
     (send pp :add-slot 'k1-val)
```

```
    (send pp :add-slot 'k-val)
    (send pp :add-slot 'xx xx)
    (send pp :add-slot 'yy yy)
    (send pp :add-slot 'k-val k)

    (send pp :add-mouse-mode 'pulling
             :title "Pulling Mode"
             :click :where-am-i
             :cursor 'hand)

    (send pp :add-mouse-mode 'point-moving
             :title "Point Moving"
             :cursor 'finger
             :click :do-point-moving)

    (send pp :mouse-mode 'point-moving)
    (send pp :mouse-mode 'pulling)
  )
)

(defmeth scatterplot-proto :begin-characteristics ()
  (let* (
         (params (send self :calc-params .00000000001 t))
         (coefs (first params))
         (rss (second params))
        )
    (send self :clear-lines :draw nil)
    (send self :abline (select coefs 0) (select coefs 1))
    (send self :your-coefs coefs)
    (send self :reg-coefs coefs)
    (send self :k-value .00000000001)
  )
)
```

define a scatterplot and various initial characteristics of the plot such as the range of the x and y axes, the number of points that are plotted, the different slots that wil be used for the plot, the initial regression coefficients, the ratio of the variances to be used, etc. Several mouse modes are also installed for use with line pulling and point moving. An initial line is drawn through the points and the cursor is changed into a hand indicating that the current mouse mode is the pulling mode. The pulling mode calls the method :where-am-i whenever the the mouse button is clicked and the mouse is dragged. The :where-am-i method locates the position of where the mouse was clicked and determines which endpoint of the line is "grabbed".

```
(defmeth scatterplot-proto :where-am-i (x y m1 m2)
(send self :while-button-down

      #'(lambda (x y)
          (let* (
                 (xx (send self :xx))
                 (yy (send self :yy))
                 (cc (send self :canvas-to-scaled x y))
                 (bg (list (first xx) (first yy)))
                 (ed (list (second xx) (second yy)))
                 (d1 (dist bg cc))
                 (d2 (dist ed cc))
                 )
             (cond ((< d1 d2) (setf (first xx) (first cc))
                   (setf (first yy) (second cc)))
              (t (setf (second xx) (first cc))
                 (setf (second yy) (second cc))))

          (send self :project-points))
 ))
)

(defun dist (x y)
(sqrt (sum (^ (- x y) 2))))
```

Once the line has been positioned, the points are projected onto the "pull line" by the method :project-points.

```
(defmeth scatterplot-proto :project-points ()
(let* (
       (xx (send self :xx))
       (yy (send self :yy))
       (n (send self :num-points))
       (x (send self :point-coordinate 0 (iseq n)))
       (y (send self :point-coordinate 1 (iseq n)))
       (coefs (send self :line-calculate xx yy))
       (k (send self :k-value))
       (params (send self :calc-params k nil))
       (ss (elt params 1))
       (xproj (elt params 2))
       (yproj (elt params 3))
      )
  (send self :clear-lines)
  (send self :adjust-to-data)
  (send self :add-lines xx yy)
  (dotimes (i n)
    (send self :add-lines (list (elt x i) (elt xproj i))
                          (list (elt y i) (elt yproj i))
    )
  )
))
```

It is also necessary to compute and store the equation of the “pull line” in a slot which is accomplished by the :line-calculate method.

```
(defmeth scatterplot-proto :line-calculate (xx yy)
  (let* (
         (m (/ (- (second yy) (first yy))
                  (- (second xx) (first xx))))
         (a (- (first yy) (* m (first xx))))
        )
   (send self :your-coefs (list a m))
   (list a m)
  )
)
```

The :project-points method calls the method :calc-params which computes the line parameters for the regression line and “pull line” for arguments t and nil respectively. The functions find-ss and find-proj are called to find the residual sums of squares and the points of projection $(\chi_i, a + b\chi_i)$. The argument test merely differentiates between the calculations of the regression line and the “pull line”.

```
(defmeth scatterplot-proto :calc-params (k test)
  (let* (
         (n (send self :num-points))
         (x (send self :point-coordinate 0 (iseq n)))
         (y (send self :point-coordinate 1 (iseq n)))
         (mux (mean x))
         (muy (mean y))
         (t-val (/ (- 1 k) k))
         (covmat (covariance-matrix x y))

         (beta (if test (/ (+ (aref covmat 1 1)
                              (- 0 (* (aref covmat 0 0) t-val))
                              (^ (+ (^ (- (* t-val (aref covmat 0 0))
                                          (aref covmat 1 1)) 2)
                                    (* 4 t-val (^ (aref covmat 0 1) 2)))
                               .5))
                           (* 2 (aref covmat 0 1)))

                        (second (send self :your-coefs))))
         (alpha (if test (- muy (* beta mux))
                         (first (send self :your-coefs))))

         (ss (find-ss alpha beta x y t-val covmat n test))
         (params (find-proj alpha beta x y n k))

       )
  (list (list alpha beta) ss (first params) (second params))
 )
)


(defun find-ss (a b x y t-val covmat n test)
   (let* (
          (fa (/ (- n 1) n))
          (ss
              (if test (/ (+ (* fa (aref covmat 1 1))
                             (- 0 (* 2 b (aref covmat 0 1) fa))
                             (* (^ b 2) (aref covmat 0 0) fa))
                       (+ t-val (^ b 2)))

                       (/ (mean (^ (- y a (* b x)) 2))
                          (+ t-val (^ b 2)))))
         )
     ss
   )
)

(defun find-proj (alpha beta x y n k)
 (let* (
        (xproj (list ))
        (yproj (list ))
        (tval (/ (- 1 k) k))
       )
   (dotimes (i n)
     (let* (
            (nume (+ (* x tval) (* beta (- y alpha))))
            (denom (+ tval (^ beta 2)))
            (xki (/ nume denom))
            (yki (+ alpha (* beta xki)))
           )
        (setf xproj (append xproj xki))
        (setf yproj (append yproj yki))
```

```
      )
    )
   (list xproj yproj)
  )
)
```

The error line, another regression line for a different value of k uses the same functions, :calc-params, find-ss and find-params in its computations. The slider is created by the method

```
(defmeth scatterplot-proto :create-slider ()
    (interval-slider-dialog '(0 1) :points 100
                                   :action #'(lambda (k1)
                                   (send self :k1-value k1)
                                   (send self :do-error-draw)))
)
```

The line is calculated by the method

```
(defmeth scatterplot-proto :do-error-draw ()
   (let (
         (k1 (send self :k1-value))
        )
    (send self :error-coefs (first (send self :calc-params k1 t)))
    (send self :draw-errorline)
   )
)
```

and is drawn by the method

```
(defmeth scatterplot-proto :draw-errorline ()
    (let* (
          (i (iseq 0 (- (send self :num-points) 1)))
          (x (send self :point-coordinate 0 i))
          (coefs (send self :error-coefs))
          )
     (send self :add-lines (list (list (min x) (max x))
                       (list (+ (first coefs) (* (min x) (second coefs)))
                             (+ (first coefs) (* (max x) (second coefs)))))
                        :type 'dashed :line-width 1)
    )
)
```

Accessor methods are needed for each of the slots in order to extract or insert information to or from them. Such an accessor method for the 'your-coefs slot is

```
(defmeth scatterplot-proto :your-coefs (&optional (val nil set))
(if set (setf (slot-value 'your-coefs) val))
(slot-value 'your-coefs))
```

A pull-line session is started by typing the command

```
> (pull-line x y)
```

where x and y are numerical lists of the same length.

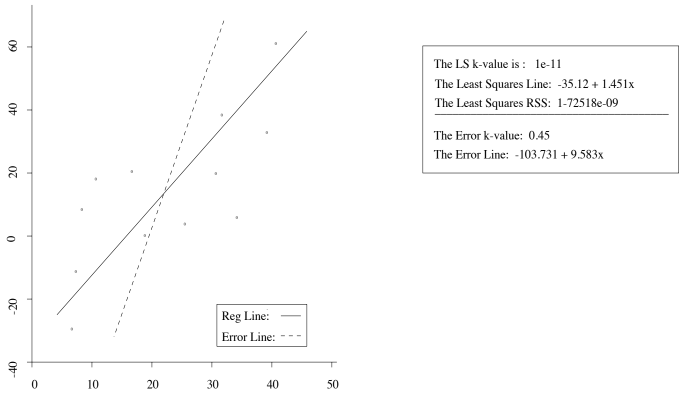
Chapter 3

Moving the Point

A second useful device to show the influence of points in a regression model can be accessed through the mouse modes option in the plot menu. This method was originally introduced in Tierney [1].

The user is allowed to grab a specific point and move it to a desired location in the plot. A sample point moving session can be seen in Figure 3.1. When the mouse button is released, the regression line according to $k = 0$ is plotted with the new coordinates of the point used in the regression computations. As in the point moving mode, another regression line can be added to the plot according to any other user specified value of k by clicking on the menu button and dragging down to "Use Errors in Regressors" button. A slider dialog will up, and the value of k may be changed in increments of 1/100. The equation of this line is also recomputed every time that a point is relocated.

This mode is accessed through the mouse mode point moving in the graph-plot menu and a finger cursor indicates the point moving mode is the current active mode. The point moving mode uses the same methods as were used in the pulling mode and moving between modes preserves the current plotted error line.



```
The LS k-value is :   1e-11
The Least Squares Line:  -35.12 + 1.451x
The Least Squares RSS:  1-72518e-09

The Error k-value:  0.45
The Error Line:  -103.731 + 9.583x
```

Figure 3.1: Point Move Sample Screen

Bibliography

[1] Tierney, Luke. *Lisp-Stat.* New York: John Wiley & Sons, 1990.

[2] Montfort, Moorijaart, and Deleeuw. "Regression with errors in variables: estimators based on third order moments." *Statistica Neerlandica*, 41 (1987).

[3] Frisch, R. Statistical Confluence Analysus by Means of Complete Regression systems, Publication No. 5, University of Oslo, Economic institute, 1934.