TONY DAVIES COLUMN

Back to basics: observing PLS

A.M.C. Davies[a] and Tom Fearn[b]
[a]Norwich Near Infrared Consultancy, 75 Intwood Road, Cringleford, Norwich NR4 6AA, UK
[b]Department of Statistical Science, University College London, Gower Street, London, UK

We will be writing a column next year giving some guidance on how to apply partial least squares regression, commonly known as PLS. Perhaps because PLS drops the word "regression", PLS tends to be regarded by newcomers (and some not so new users) as a magical piece of chemometrics which has no relationship to multiple linear regression (MLR) or principal component regression (PCR). So in this column we will attempt to persuade you that all three methods are closely related and that all rely on a "least squares" regression. An earlier column [**8(6),** 20 (1996), available on the *Spectroscopy Europe* web site] had the same intention but this time we will use diagrams generated by a program that TF wrote for one of our training courses. During use in the training course students see it being manipulated but the views needed for discussion are essentially static so we hope that it will work in this rather larger virtual classroom! The program is known as PFdemo (Project and Fit demonstration) and essential is comprised of two linked graphs and some buttons that instruct MATLAB to do some calculations and display the results. Figure 1 shows the layout with ten data points plotted in the left-hand graph. The control buttons will not be shown again so they will be described now. "Plot x" plots ten pairs of data points. These data points are from 10 samples for which we have values for two variables, x_1 and x_2 and a reference value, y. **(Note: this data stays the same throughout this example.)** "Project x" generates a blue line, z, that can be rotated around the centre of the figure on to which the positions of the samples are projected. This means that the position of a sample on the "z" line is determined by the intersection of a perpendicular which passes through the sample point as shown in Figure 2. "Clockwise" and "Anticlock" controls the angle of the projection line with respect to x_1 and "Step size" allows the operator a choice from course to fine changes in the angle each time a rotation button is pressed. Figure 3 shows a different choice of angle compared to Figure 2. We can calculate the values of z for each of the samples ($z=c_1x_1+c_2x_2$, where c_1 and c_2 are the cosines of the angles between the x_1 and x_2 axes): note that the data are centred to maintain the display. "Plot y, z" plots the projected data against the y values in the right-hand graph, Figure 4. You can see the blue dots on the z axis have been directly transferred from the left-hand graph. "Fit y, z" calculates and displays the line of best fit, i.e. a line giving the least sum of squared errors for the y, z data, Figure 5. The "Show V", "Show R^2" and "Show prod" buttons cause the program to display the variance of the z data, the value of R^2 for the y, z fit and the value of $V \times R^2$, Figure 6.

If we use R^2 to choose the angle of z, we can find the angle which gives a maximum value for R^2 as shown in Figures 7–

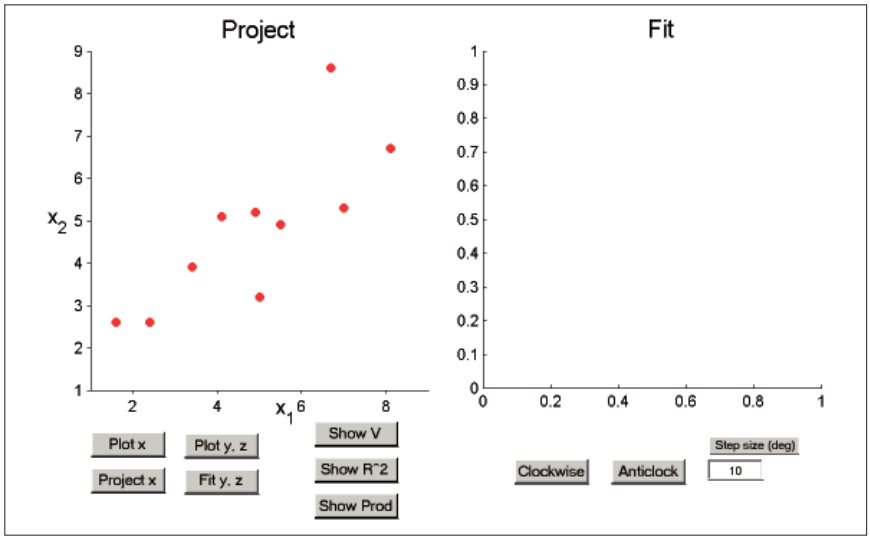

Figure 1. The complete PFdemo display with its control buttons.

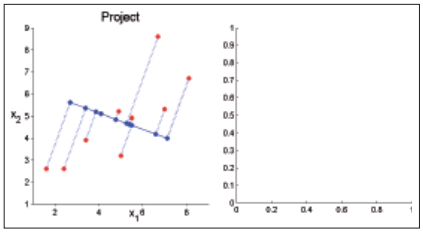

Figure 2. Projection on to z.

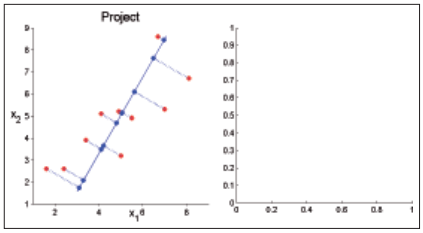

Figure 3. Projection on to another z.

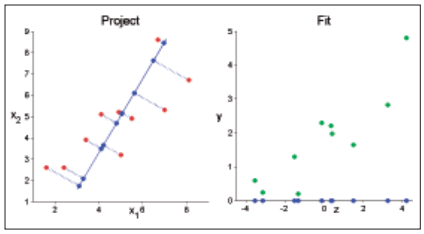

Figure 4. z values plotted against y.

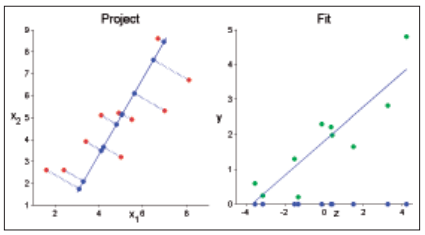

Figure 5. Best fit line for *y*, *z* data.

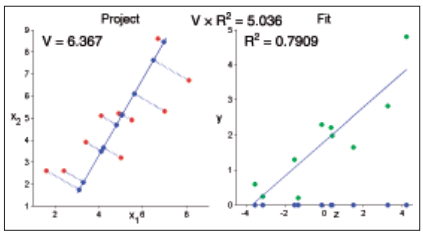

Figure 6. Display of statistics.

9. This is the value which corresponds to an MLR solution.

If we use *V* to choose the angle of *z*, we can find the angle which gives a maximum value for *V* as shown in Figures 10–12. This is the value which corresponds to a one-factor PCR solution. Note that we show the value of R^2 in Figure 12 but this was not used to guide the choice of *z*. By maximising the variance in the *z* data we hope to include the information which relates the original variables to the *y* data but we are protected from over-fitting because we do not start to test the least squares fit until after the *z* values have been determined.

If we use "Product" to choose the angle of *z*, we can find the angle which gives a maximum value for the product $V \times R^2$ as shown in Figures 13–15. This is the value which corresponds to a one-factor PLS solution. Note that we show the value of R^2 in Figure 15 but this was not used to guide the choice of *z*. In PLS we compromise between just using *V* or just using R^2. PCR and PLS tend to give similar results but PLS usually gets to an optimum solution with fewer factors than PCs in the PCR solution.

You may think that this demonstration shows that the MLR (R^2) solution is the best method. With two variables it may be best, but with many variables it is too easy to make $R^2=1$, even when there is noise in *y*. If there were only three samples in the plots here, we could make $R^2=1$, whatever the values for *y*. If we apply the MLR solution with 700 variables then we will produce a calibration which is grossly over-fitted and it will not give good predictions of unknown samples. If we use the PCR or PLS solutions we may not get such a good R^2 but we will be protected (if we follow the rules) against over-fitting, so our calibration will have a predictable performance. **It is very important to remember when doing calibrations that we should seek a calibration with good prediction performance.**

Of course, this is not how MLR, PCR or PLS are actually computed but it is an exactly equivalent method. It is not too difficult for you to imagine the graphs if we started with three variables and a three-dimensional cloud of datapoints (but very much more difficult to write the display program) but then we run out of human vision experience. Mathematically there is no such limit; the trick is to say 700 variables and think three!

Please note that we do NOT use the words "best calibration". We would not know which is the "best" calibration for many years; what we need is a reliable calibration which will give useful results.

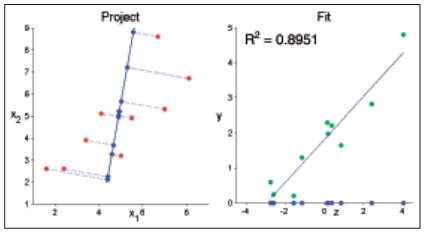

Figure 7. Rotating *z* to improve R^2.

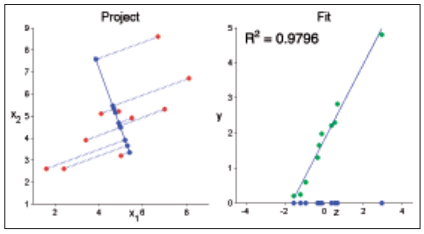

Figure 8. Better R^2.

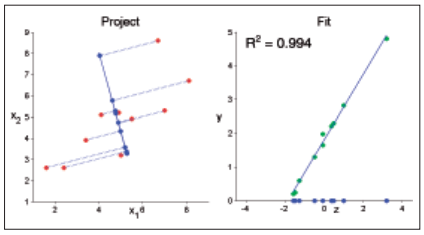

Figure 9. Best R^2.

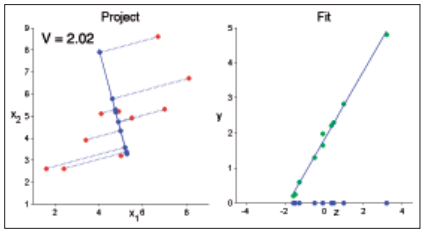

Figure 10. Low *V*.

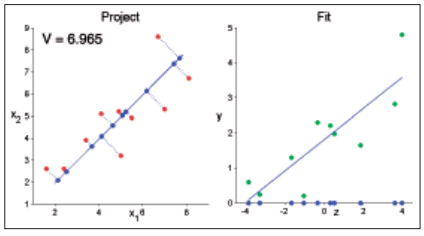

Figure 11. Better *V*.

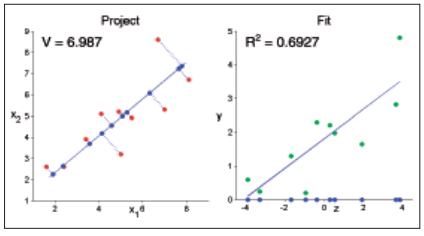

Figure 12. Best *V*.

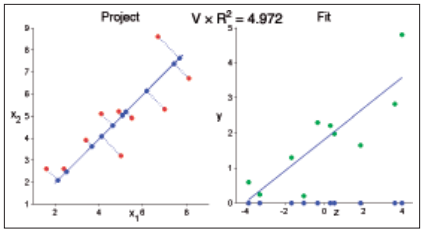

Figure 13. Looking at $V \times R^2$.

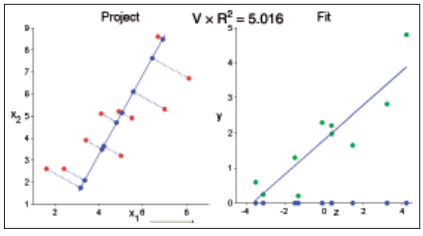

Figure 14. Improving $V \times R^2$.

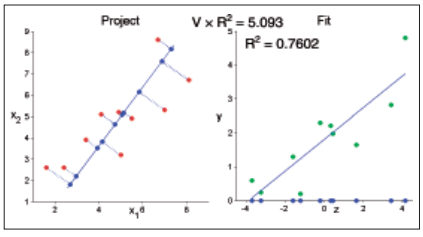

Figure 15. Best $V \times R^2$.