Linear Model Selector

Forward Selection Algorithm

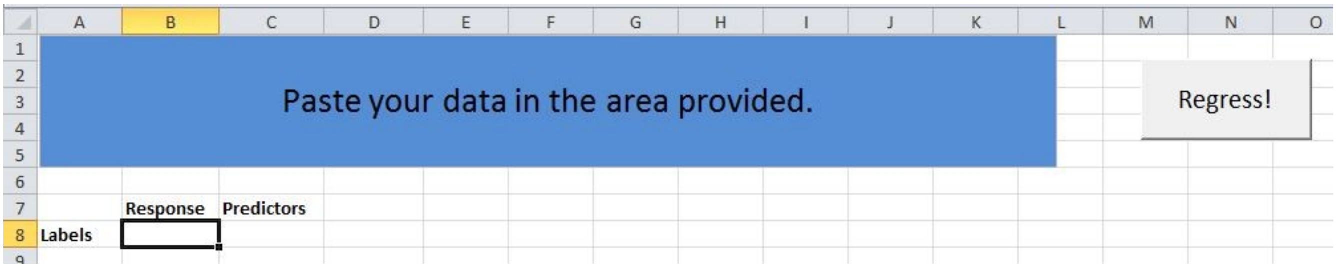
Richard Wilde
April 12, 2011

Executive Summary

Knowing what to make of data can be very difficult. A linear model allows an analyst to predict the value of an important but unknown quantity with by plugging known values into a linear equation. Different combinations of variables have different levels of effectiveness in predicting the values of the response variable. One way to measure the effectiveness of a model is SSE, or sum of squared error. Deciding which predictors to include in a model and which to leave out involves a trade-off between how much that predictor decreases the SSE (its added sum of squares) on the one hand and the risk of over-fitting the model by adding too many predictors on the other. This program implements an automatic model selector that employs the forward selection algorithm. This provides a great starting point for developing a model and eliminates the tedium of searching manually.

Data Input

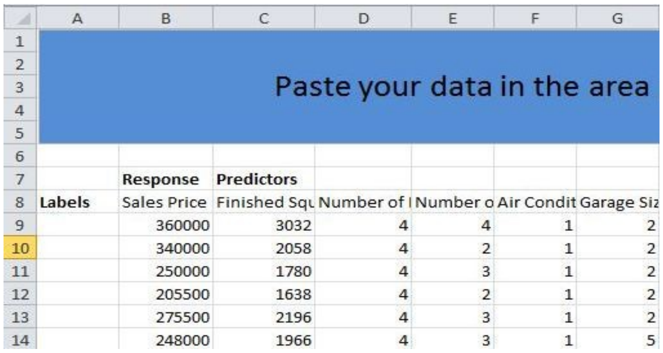
I found that the only data input form necessary was a worksheet in which the user could paste in a dataset with any number of variables and observations. All the user has to do is make sure the response variable is listed first, followed by any potential predictors. Varying numbers of predictors and rows, multicollinearity (linearly dependent predictors), and ineligible predictors are dealt with automatically by the program. The following figure shows the easy input form.



Paste your data in the area provided.

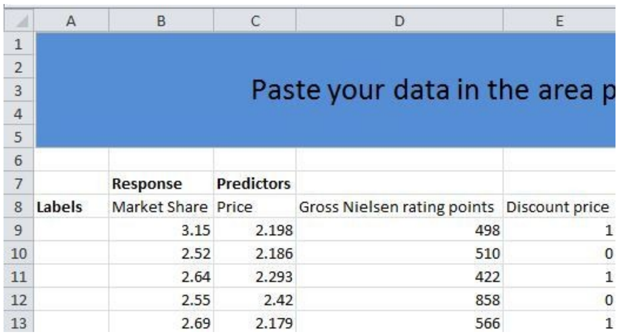
	A	B	C
7		**Response**	**Predictors**
8	**Labels**		

In these examples the form has been filled in with different datasets.



Paste your data in the area

	A	B	C	D	E	F	G
7		**Response**	**Predictors**				
8	**Labels**	Sales Price	Finished Squ	Number of I	Number o	Air Condit	Garage Siz
9		360000	3032	4	4	1	2
10		340000	2058	4	2	1	2
11		250000	1780	4	3	1	2
12		205500	1638	4	2	1	2
13		275500	2196	4	3	1	2
14		248000	1966	4	3	1	5



Paste your data in the area p

	A	B	C	D	E
7		**Response**	**Predictors**		
8	**Labels**	Market Share	Price	Gross Nielsen rating points	Discount price
9		3.15	2.198	498	1
10		2.52	2.186	510	0
11		2.64	2.293	422	1
12		2.55	2.42	858	0
13		2.69	2.179	566	1

Once the data has been pasted into the provided space, all the user has to do is press the “Regress!” button and wait for the program to do all the work.

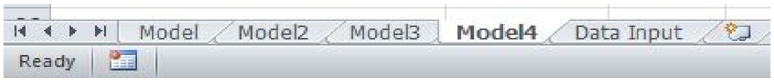
Processing

The program first separates the data from the form and constructs the matrices necessary for linear regression. Further matrix formulas arrive at the SSE. The recursive algorithm follows like this:

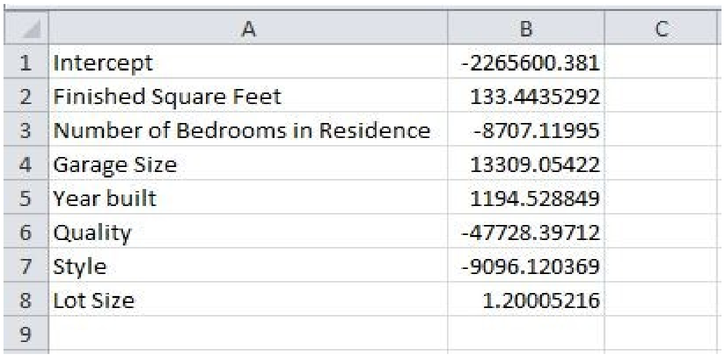
1. Fit the base model (just the intercept with no predictors)
2. Make a model with each predictor and see which lowers the SSE the most.
3. Perform a hypothesis test. If the SSE is lowered significantly, the predictor is added. Otherwise, program terminates.
4. Search for the next predictor to add by the same criteria.

Results

Once a model has been developed, a new worksheet is created. The default name is "Model." If that name is taken, the program checks the names "Model2," "Model3," and so forth until a name is found.



This way the user can develop multiple models without name conflicts just by inputting new data into the "Data Input" tab and pressing the "Regress!" button. The model tab that is created has the label of each predictor which has been selected followed by the estimate for its coefficient in the linear model. The result looks like this:



	A	B	C
1	Intercept	-2265600.381	
2	Finished Square Feet	133.4435292	
3	Number of Bedrooms in Residence	-8707.11995	
4	Garage Size	13309.05422	
5	Year built	1194.528849	
6	Quality	-47728.39712	
7	Style	-9096.120369	
8	Lot Size	1.20005216	
9			

Discussion

I began work on this project with the intention of having a series of user forms that would help the user manipulate their data input, choose variables, and identify the response. I learned a great deal about user forms. I had to look up a lot of different things, such as how to prompt the user to input a range and how to handle possible errors. Ultimately, I found that these forms were a lot of work and they didn't actually do very much. About all they did was to allow the user to select which variables they wanted to use, and since that decision is the point of the program anyway, I decided it was unnecessary and made input more straightforward. Throughout this process I learned to make sure what I'm programming actually has a purpose and does not just distract from the real point of the project.

Processing the data requires a lot of matrix algebra and recursion. I needed the following matrix functionality:

- Matrix Multiplication
- Matrix Inversion
- Matrix Subtraction

After some research, I found that the matrix multiplication function MMULT() and the matrix inversion function MINVERSE() are comparable to similar functions in statistical programs. Both are worksheet functions used in array formulas. I found that they could be used in VBA on the .value property of ranges using the implementation of each in the WorksheetFunction object. Surprisingly, I had more trouble with subtraction. The “-” operator cannot be used with the .value property of a range object in VBA, so I ended up setting the range's .FormulaArray property with the addresses of the source matrices and the “-” operator, which worked fine. I did all the computations in worksheets, so I made sure to turn off screen updating and alerts.

To determine if a new predictor's added sum of squares was significant, I needed to get a p-value. The F-statistic can be determined with a simple equation, and WorksheetFunction.F_Dist is an implementation of the cumulative distribution function for the F distribution, so I used that to find the p-value.

Overall, I learned that while Excel is nowhere near the best program for statistical analysis (it's actually quite stigmatized by the discipline), it does have some very useful statistical functions if a programmer is willing to put some work into it.