

# Using Regression and Oracle's Crystal Ball Software for Generating Probabilistic Forecasts

by Parag C. Pendharkar, Pennsylvania State University at Harrisburg

Oracle's Crystal Ball is a Microsoft Excel add-in component ([www.oracle.com/crystalball](http://www.oracle.com/crystalball)). It is a useful tool that can be used in undergraduate and graduate management science or information system classes. I use a software size estimation problem to illustrate the application of this add-in component.

Software project management often requires estimation of either software size or software effort for project budgeting and cost control. There are a variety of statistical, machine learning, and COCOMO models that aid in cost estimation. Unfortunately, most of these models provide a point estimate for software size/software effort given a set of known input variables, and do not provide managers to incorporate uncertainty in the set of input variables. There are a few Bayesian models that will allow a manager to incorporate uncertainty (Pendharkar et al, 2005), but these models are often complicated and require rigorous understanding of Bayesian statistics and graph theory.

A simple alternative to the use of Bayesian networks is to first use a linear regression to learn a software parameter estimation model, and then use a Monte Carlo simulation to incorporate uncertainty in input variables to a generate probabilistic forecast for a software parameter given uncertainty in inputs and software parameter estimation model. In this tutorial, I will show how this two-step approach can be easily accomplished using Microsoft Excel and Oracle's Crystal Ball software add-in for Microsoft Excel.

## Overview of a Real-World Software Size Estimation Problem and Its Modeling Using Crystal Ball Software

To illustrate the two step procedure, I use the dataset from Bielik (Bielak, 2000) study. This dataset consists of 152 components from a real life X/Motif-based C++ research and data-analysis application. The development time for the application was 24 months and a staff of four to nine full-time developers created the component-based application. There were four independent variables in the study that were shown to impact software size measured in source lines of code (S). The four independent variables were: the number of GUI elements (G) in a component; the number of events (E) and state changes handled by a dialog, window or an object; the number of member functions (F) per component; and the number of reused (R) system components used by a module. The linear regression relationship between the set of independent variables and dependent variable can be represented as follows:

$$S = \beta_0 + (\beta_1 \times G) + (\beta_2 \times E) + (\beta_3 \times F) + (\beta_4 \times R),$$

where parameters  $\beta_i, \forall i \in \{0, \dots, 4\}$  are regression error and coefficients of regression.

In the first step of the two step procedure, I obtain the values of these parameters by running the multiple regression in Microsoft Excel. The data on 152 components appears in cell range A153:E153 of a Microsoft Excel spreadsheet as shown in Figure 1. Using Data-> Data Analysis tab,



**Parag C. Pendharkar** is a professor of information systems at Pennsylvania State Harrisburg. He has published over 75 articles in several journals including Annals of Operations Research, Communications of

the ACM, Computers & Operations Research, Decision Sciences, Decision Support Systems, European Journal of Operational Research, among others.

[pxp19@psu.edu](mailto:pxp19@psu.edu)

<http://www.personal.psu.edu/pxp19/>

Figure 1.  
The Regression  
Dataset

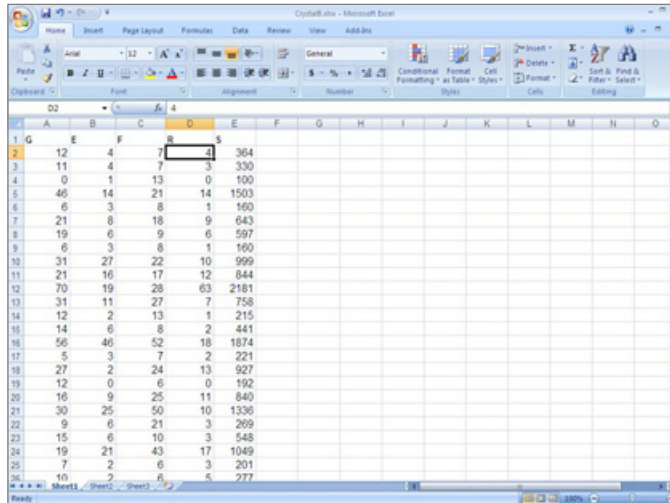


Figure 2.  
Regression Data  
Input and Output  
Specifications

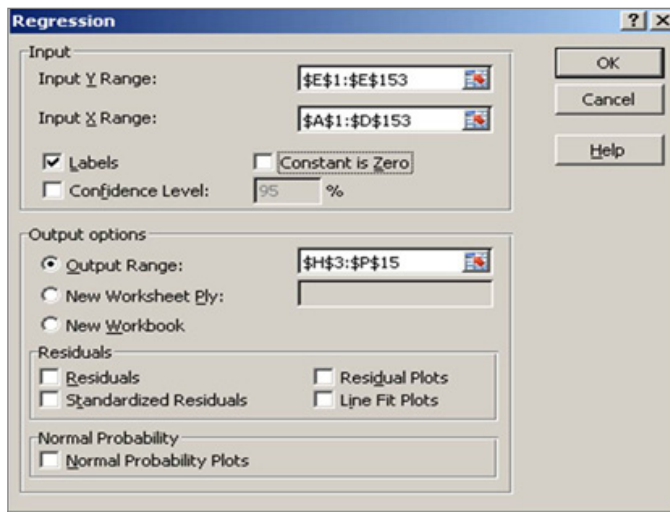
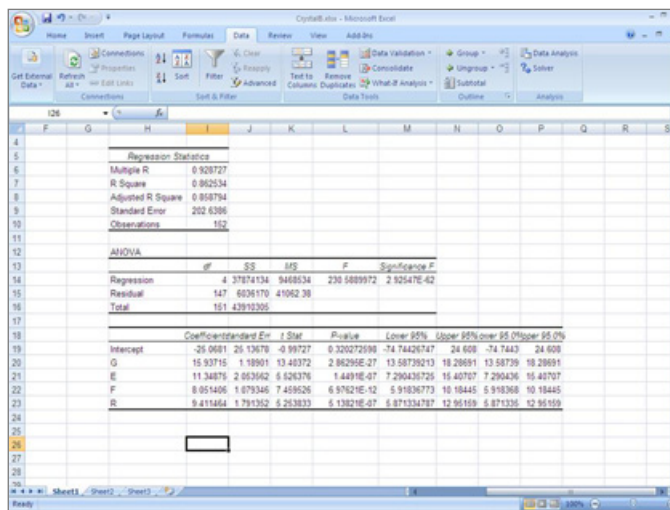


Figure 3.  
Microsoft Excel  
Regression Output



I choose Regression from the dialog box and click the OK button. I then enter the data range and output range as shown in Figure 2, and then click the OK button to run the regression procedure in Microsoft Excel.

Figure 3 illustrates the output of the regression, which is summarized in Table 1. The results indicate that the regression model is statistically significant with 99% statistical confidence, and variances in independent variables explain approximately 86% (Adj. R-Squared) of variance in dependent variable. The regression parameters for the regression model are summarized in Table 2. Table 2 indicates that all the independent variables (G, E, F, and R) are statistically significant in explaining variance in the dependent variable S.

Table 1: Regression Model Summary Table

	Deg. of Freedom	Sum of Sq.	Mean Sq.	F-Value	Sig.
<b>Regression</b>	4	37874134	9468534.00	230.50	0.000*
<b>Individual</b>	147	6036170	41062.38		
<b>Error</b>	151	43910306			

\*Significant at 99%; R-Squared 0.86; Adj. R-Squared 0.86

In the second step, I assume that a manager needs to estimate software size of two components where in the first component there is some uncertainty in the number of GUI elements; and in the second component, there is some uncertainty in the number of events. Table 3 illustrates input values for two components. The uncertainty in GUI elements for the first component, in Table 3, is represented as {11, 12, 13}, which implies that a manager is unsure about the number of GUI elements in the component as it could be either 11 or 12 or 13. Similarly, uncertainty in the number of events in second component is indicated as {35, 40, 43}.

A manager assigns a subjective belief (Bayesian probability) on the likely values of GUI elements in the first component and the number of events in the second component. These Bayesian probabilities are shown in Tables 4 and 5. Given that these uncertainties, a manager would like to get an estimate of software size for these two components.

Table 2: Regression Model Parameters

Parameter	Value	t-statistic	Sig.
$\beta_0$	-25.07	-0.99	0.320
$\beta_1$	15.94	13.40	0.000*
$\beta_3$	11.35	5.53	0.000*
$\beta_4$	9.41	5.25	0.000*

\*Significant at 99%

Table 3: Software Size Independent Variables with Uncertainty

G	E	F	R
{11,12,13}	3	13	3
38	{35, 40, 43}	55	20

Table 4: Bayesian Probabilities for GUI Element in the First Component

G	Bayesian Probability
11	0.2
12	0.6
13	0.2

Table 5: Bayesian Probabilities for the Number of Events in the Second Component

E	Bayesian Probability
35	0.3
40	0.5
43	0.2

Figure 4.  
New Worksheet  
Setup for Crystal  
Ball Analysis

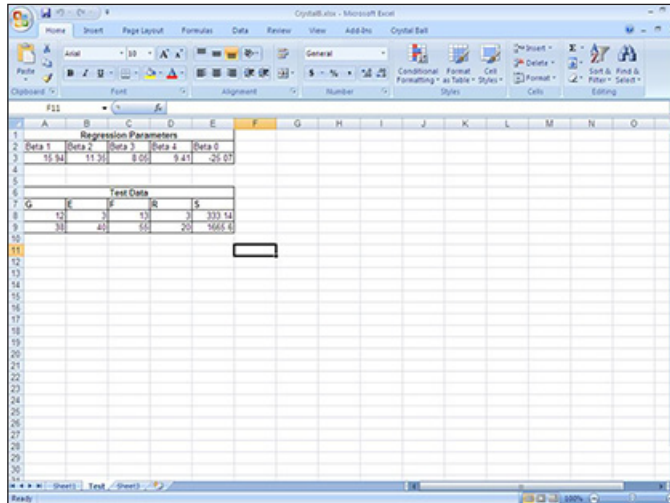


Figure 5.  
Data Distribution in  
Crystal Ball

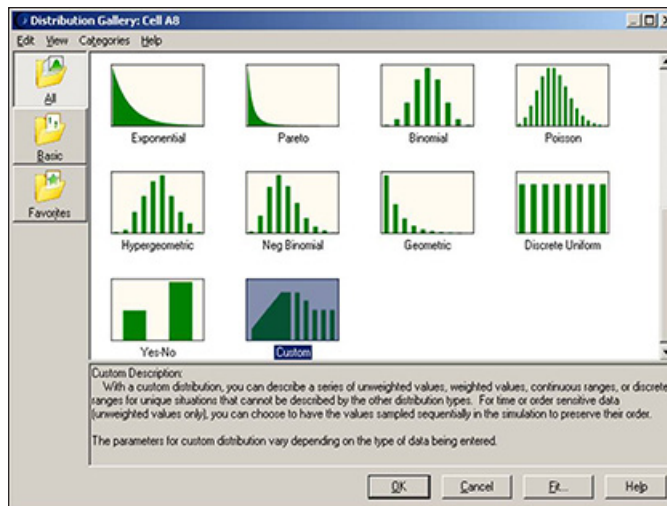
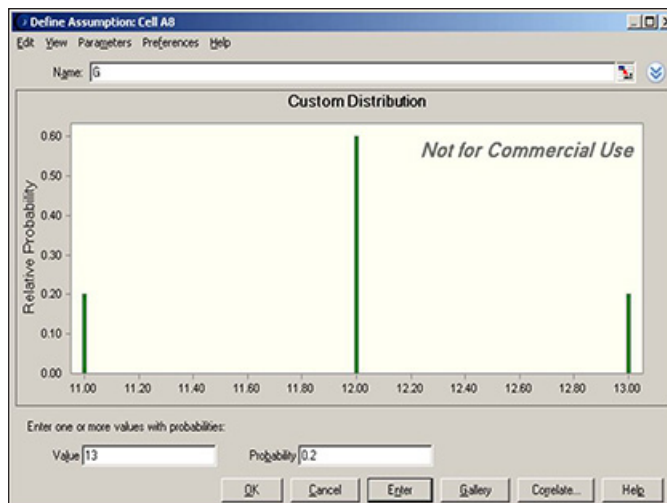


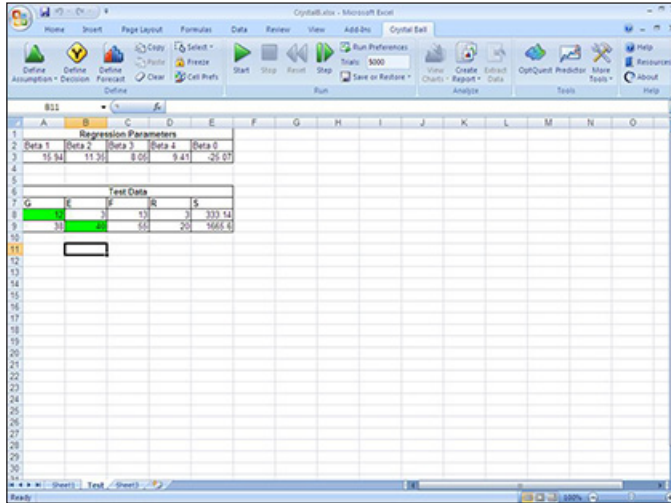
Figure 6.  
Entering Probability  
Distribution for  
the Number of GUI  
Elements



I will now show how the Crystal Ball will allow a manager to get an estimate of software size for these two components. However, before I open the Crystal Ball software, I save and close the original Excel workbook containing data and regression analysis from the first step. Next, I start the Crystal Ball and then open the original Excel workbook. In the second worksheet of the original Excel workbook, I enter data on regression parameters (A1:E3) obtained from regression analysis and data about two components (A6:D9) as shown in Figure 4. In cell E8, I enter following formula:  $=(A8*\$A\$3)+(B8*\$B\$3)+(C8*\$C\$3)+(D8*\$D\$3)+\$E\$3$ , copy it and paste it in cell E9. The values in cell E8 and E9 reflect an estimate of software size if there were no uncertainties in the number of GUI elements in the first component and the number of events in the second component.

I click on cell A8 and then click on Crystal Ball tab in Microsoft Excel (see Figure 4 for location of Crystal Ball tab on the Excel ribbon, and Figure 7 for various options for Crystal Ball tab). I then select Define Assumption->Custom to get a screen shown in Figure 5. I then click on OK button. Using the text boxes titled Value, Probability and the Enter button, I define the probability distribution for the number of GUI elements which is shown in Figure 6, and then I click the OK button. I then click on cell B9 and, using Table 5 values, define a custom probability distribution for the number of events for the second component.

Figure 7.  
The Excel Worksheet  
with Probabilistic  
Variables



After entering the probability distributions for the two components, my Excel spreadsheet looks similar to one shown in Figure 7. Crystal Ball highlights the variables with uncertainty in green color.

I select cells E8 and E9 one at a time and then click Define Forecast icon under the Define section of the Crystal Ball tab to get a dialog box shown in Figure 8. I make the selections as shown in Figure 8 and then click OK button. I repeat the procedure for cell E9 where I enter "Second Component" in the Name textfield.

After highlighting variables with uncertainty in inputs and dependent variables (using Define Forecast), my spreadsheet looks similar to the one shown in Figure 9. Crystal Ball highlights dependent variables with the light blue color.

Figure 8.  
Forecast Variable  
Dialog Box for  
Cell E8

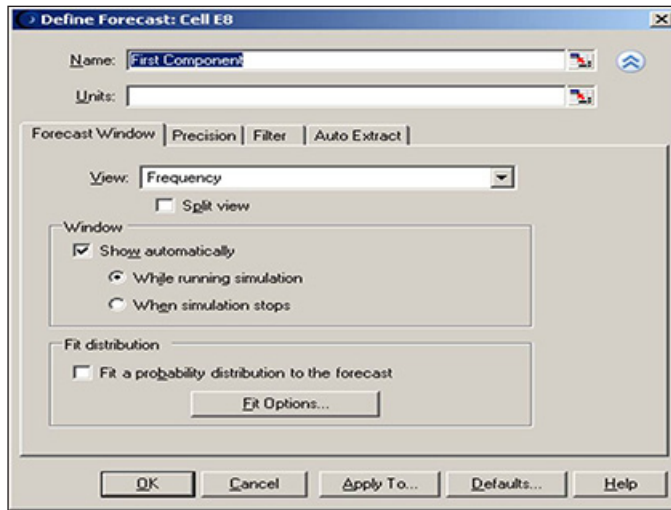


Figure 9.  
The Spreadsheet  
with Uncertain  
Input Variables and  
Dependent Variables

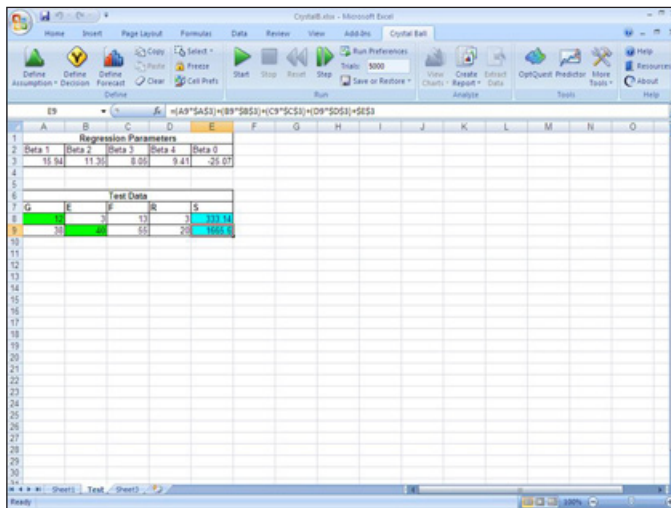
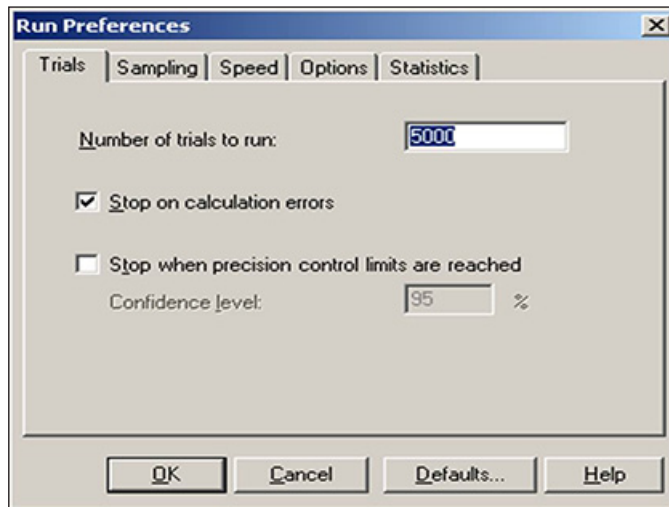


Figure 10.  
The Run Preferences  
Dialog Box



I then click on Run Preferences icon under Run section of Crystal Ball tab, and enter responses as shown in Figure 10. As per my preferences, I will run Monte Carlo simulation for 5000 random iteration where random value for input variables will be generated using the probability mass functions, and values of dependent variables will be computed using regression parameters.

To generate probabilistic forecasts of software size for two components, I click on Start icon under Run section of the Crystal Ball tab. Crystal Ball runs 5000 random iterations and generates the probability distribution for the dependent software size variables for the first and second component. Selecting View > Percentiles option shows percentiles for each component as shown in Figures 11a and 11b, and selecting View > Statistics option shows statistics for each component as shown in Figures 12a and 12b.

Figure 11a.  
The Software Size  
Percentiles for the  
First and the Second  
Component

Percentile	Forecast values
0%	317.20
10%	317.20
20%	317.20
30%	333.14
40%	333.14
50%	333.14
60%	333.14
70%	333.14
80%	333.14
90%	349.08
100%	349.08

Figure 11b.  
The Software Size  
Percentiles for the  
First and the Second  
Component

Percentile	Fit: Min Extreme	Forecast values
0%	-Infinity	1,608.85
10%	1,609.52	1,608.85
20%	1,630.16	1,608.85
30%	1,643.06	1,665.60
40%	1,652.94	1,665.60
50%	1,661.33	1,665.60
60%	1,669.01	1,665.60
70%	1,676.51	1,665.60
80%	1,684.50	1,699.65
90%	1,694.35	1,699.65
100%	Infinity	1,699.65

Figure 12a.  
The Software Size  
Statistics for the  
First and the Second  
Component

Statistic	Forecast values
Trials	5,000
Base Case	333.14
Mean	332.91
Median	333.14
Mode	333.14
Standard Deviation	10.14
Variance	102.77
Skewness	0.0121
Kurtosis	2.47
Coeff. of Variability	0.0305
Minimum	317.20
Maximum	349.08
Mean Std. Error	0.14

Figure 12b.  
The Software Size  
Statistics for the  
First and the Second  
Component

Statistic	Fit: Min Extreme	Forecast values
Trials	---	5,000
Base Case	---	1,665.60
Mean	1,655.54	1,655.58
Median	1,661.33	1,665.60
Mode	1,671.41	1,665.60
Standard Deviation	35.27	33.07
Variance	1,244.15	1,093.43
Skewness	-1.14	-0.3483
Kurtosis	5.40	1.83
Coeff. of Variability	0.0213	0.0200
Minimum	-Infinity	1,608.85
Maximum	Infinity	1,699.65
Mean Std. Error	---	0.47

Based on the Crystal Ball analysis, a manager can expect software size for first component to be approximately 333 source lines of code with 80% certainty that it will not exceed 333 source lines of code. For the second component, a manager can expect software size to be approximately 1656 source lines of code with an over 50% chance that it will exceed 1656 source lines of code and over 20% chance that it will exceed 1666 source line of code with a maximum of 1700 source lines of code.

### Conclusions and Discussion

In this tutorial, I have illustrated how Crystal Ball can be used to model uncer-

tainty in independent variables to generate probabilistic forecast of software size. I showed a two-step procedure that uses Microsoft Excel's regression model to learn regression parameters from historical data, and then uses the Crystal Ball's simulation capabilities to generate probabilistic forecast by incorporating a manager's beliefs about uncertainty in independent variables. I believe that the tool is simple for any practitioner to use. While I used discrete probability distribution, Crystal Ball offers wide range of probability distributions (see Figure 5 for some of these distributions) that a manager can use depending on the type of variables and problem type. I believe that most probabilistic software

estimation problems, including software cost, software effort and software size, can be easily handled by Microsoft Excel enhanced by Crystal Ball add in.

### Acknowledgements

I would like to thank Mr. James Bielak for providing the software size data for analysis.

### References

Bielak, J. (2000). Improving size estimates using historical data. *IEEE Software*, November/December, 27-35.

Pendharkar, P.C., Subramanian, G.H., Rodger, J.A. (2005). A probabilistic model for predicting software development effort. *IEEE Transactions on Software Engineering*, 17(10), 1379-1388. ■