SOCIETY OF ACTUARIES

Article from:

CompAct

July 2009 – Issue 32

R Corner[i]—Model Formula Framework

By Steven Craighead

Steve Craighead, ASA, MAAA, is an actuarial consultant at TowersPerrin in Atlanta, Ga. He can be reached at steven.craighead@towersperrin.com.

Editor's note: R Corner[i] is a series by Steve Craighead introducing readers to the "R" language used for statistics and modeling of data. The first column was published in the October 2008 issue, and explains how to download and install the package, as well as providing a basic introduction to the language. Refer to each CompAct issue since then for additional articles in the series. The introductory article can be found on p. 24 of the October 2008 on the SOA Web site: *http://soa.org/library/newsletters/compact/2008/october/com-2008-iss29.pdf*

In this article we will examine the Model Formula Framework within R[1] for linear and generalized linear models. You may use this framework to set up a large number of different statistical models.

Since we are going to concentrate on linear-like models, we will assume that both the predictors and the resultant variables are continuous. You may also use the framework to model Analysis of Variance (ANOVA) models on factor or categorical data (variables that take on discrete values), but that will be discussed in a future column.

The simplest formula will look like this:

observed ~ predictor1 + predictor2 + predictor3 + ... + predictorN,

where *observed* is the variable that you wish to model, by determining some relationship with the various predictor variables. Note: The "+" convention is used to include a variable in the model in a linear fashion. For a linear regression, the actual model that is fit is of this form:

observed = (C1)predictor1 + (C2)predictor2 +...+ (CN)predictorN + residual_error.

Here the *Cn* denote the separate coefficients of this model.

If you want to model the observed against all variables (except itself) in a dataset, you may use the "." convention, such as:

observed ~ .

If you want to eliminate a specific term within a model, you may use the "-" convention,

observed ~ -1 + predictor1 + predictor2 +...+ predictorN

Here the "-1" indicates that you do not want to have an intercept term calculated within your model.

For instance, if you want to model observed against all predictors except for predictor3, you can use the "." and "-" in this way:

observed ~ . – predictor3

The symbol ":" with continuous variables indicates the actual product of variables. The symbol "*" denotes a factor crossing. So that *predictor1*predictor2* denotes *predictor1 + predictor2 + predictor1:predictor2*. Note how "*" is not a true product of the variable in the same way that "+" is used above. The "^" symbol denotes a factor crossing to a specific degree. For instance, *(predictor1 + predictor2 + predictor3)^2* is the same as *(predictor1 + predictor2 + predictor3)*(predictor1+predictor2+predictor3)*, which is a formula of the form:

(predictor1 + predictor2+predictor3 + predictor1:predictor2+ predictor1:predictor3+predictor2:predictor3)

Usually, you will just use the actual variable names, but you can use functions of the variables as well. For instance, *exp(observed) ~ exp(predictor1)* is a valid formula. Now, note however, that there appears to be a contradiction to this format, however, if you need to create formulae that actually need the normal arithmetic meaning of the operators. You overcome this by

FOOTNOTES

[1] R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

using a *I()* format to surround the model components that need to actually use these arithmetic operators. So if you wanted to create a formula that actually adds two predictors together, before creating the model, you would use a form like this:

observed ~ I(predictor1+predictor2)

Now, to further clarify the idea of the ":" format from above, *predictor1:predictor2* is the same as *I(predictor1*predictor2)*.

If you want to create transformed observed and predictor models, you can use a formula like this:

I(1/Observed) ~ I(1/predictor1^2) + sqrt(predictor2)

Here the multiplicative inverse of the observed variable is fit against the multiplicative inverse of *predictor1* squared and the square root of *predictor2*.

If you wish to create a polynomial model of a predictor, you may use the *poly(,)* convention. For instance, suppose you want to model *predictor1* as a cubic equation and *predictor2* as quadratic and predictor3 as linear. The formula would look like this:

observed ~ poly(predictor1,3)+poly(predictor2,2) + predictor3

In generalized linear models, you may use the *s()* convention. When you surround a variable by this convention, it tells R to fit a smoothed model. For instance:

observed ~ s(predictor1) + predictor2 + s(predictor3)

would fit *observed* by creating non-parametric smoothed models of *predictor1* and *predictor3* and use the actual values of *predictor2*.

Let's look at a couple of examples of a linear regression model using some of the formulas above. Define a dataset containing three predictors (say the variable names are x, y and z. In our model, we will let x contain 100 samples of the standard normal distribution, y will contain 100 samples of the continuous uniform distribution on the interval (2,5), and z will be (x+y)^3. The observed variable will be r. Let r take on the values of x^2 + 1/y + z. To generate these, use the following commands:

```
x <- rnorm(100,mean=0,sd=1)
y <-runif(100,min=2,max=5)
z <- (x + y)^3
r <- x^2 + 1/y + z
```

Now, create a dataframe called RTest containing these variables by executing this command:

RTest <- data.frame(cbind(r,x,y,z))

Here *cbind()* will concatenate the four variables into a matrix whose columns are *r, x, y* and *z*. *The data.frame()* function then converts the matrix into a dataframe. If you type

(names(RTest))

R will display the separate column names:

[1] "r" "x" "y" "z"

Using the "." convention, create the following simple regression model:

(Model <- lm(r ~ ., data=RTest))

Note how the *data* input parameter is used to reference the dataframe *RTest*. Now, R will display:

Call:
lm(formula = r ~ ., data = RTest)

Coefficients:

(Intercept)	x	y	z
4.217	-1.175	-1.303	1.028

So the linear model is:

r = 4.217 + -1.175x -1.303y+1.028z

CONTINUED ON **PAGE 16**

To observe additional information regarding the model, use this command:

(summary(Model))

R will display:

Call:
lm(formula = r ~ ., data = RTest)

Residuals:

Min	1Q	Median	3Q	Max
-1.5187	-0.6738	-0.3520	0.4036	3.1998

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	4.217229	0.537224	7.850	5.89e-12	***
x	-1.175125	0.195919	-5.998	3.52e-08	***
y	-1.303079	0.199134	-6.544	2.93e-09	***
z	1.027829	0.003924	261.915	< 2e-16	***

Signif. codes: 0 '*' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1**

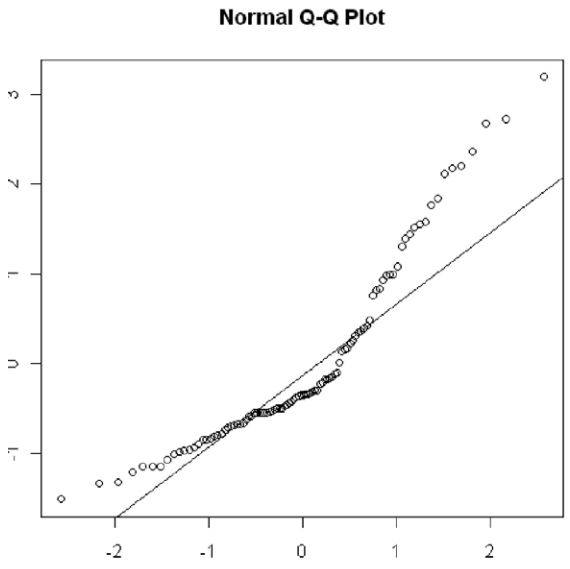
Residual standard error: 1.044 on 96 degrees of freedom
Multiple R-squared: 0.9998, Adjusted R-squared: 0.9998
F-statistic: 1.354e+05 on 3 and 96 DF, p-value: < 2.2e-16

The F-statistic is very significant as well as each coefficient and the R-squared values are almost 1, so you may be tempted to just stop here and just use this simpler model. However, if you use the *plot()* function, you will see that the model's behavior is not that good. There are four separate graphs produced with the *plot()* function, but below we will only display the Quantile-Quantile (Q-Q) plot, to demonstrate that the residuals are not normal. We will use these commands to plot just Q-Q plot:

qqplot(resid(Model))
qqline(resid(Model))

R will display the graph above (right):

If the residuals were actually normal, the sorted residuals would follow the 45-degree line. However, this graph indicates that the left tails are actually heavier than a normal curve and the right tail is lighter, so since linear regression models require the residuals to be normal, we can say that this model is faulty. Also, note how the tails are not symmetric.



Let's examine the results of the actual formula that we used to model r, as a regression model:

(Model2 <- lm(r ~ I(x^2) + I(1/y) + z,data=RTest))

R displays this:

Call:
lm(formula = r ~ I(x^2) + I(1/y) + z, data = RTest)

Coefficients:

(Intercept)	I(x^2)	I(1/y)	z
6.879e-15	1.000e+00	1.000e+00	1.000e+00

Notice how the coefficients are all equal to one, but there is an intercept term, which isn't in the original model, so revise the regression to eliminate the intercept term:

(Model3 <- lm(r ~ -1 + I(x^2) + I(1/y) + z,data=RTest))

R displays:

Call:
lm(formula = r ~ -1 + I(x^2) + I(1/y) + z, data = RTest)

Coefficients:
I(x^2) I(1/y) z
1 1 1

Now examine the results of summary:

(summary(Model3))

Call:
lm(formula = r ~ -1 + I(x^2) + I(1/y) + z, data = RTest)

Residuals:
Min 1Q Median 3Q Max
-1.298e-13 -3.347e-15 -1.798e-16 4.946e-15 3.037e-14

Coefficients:

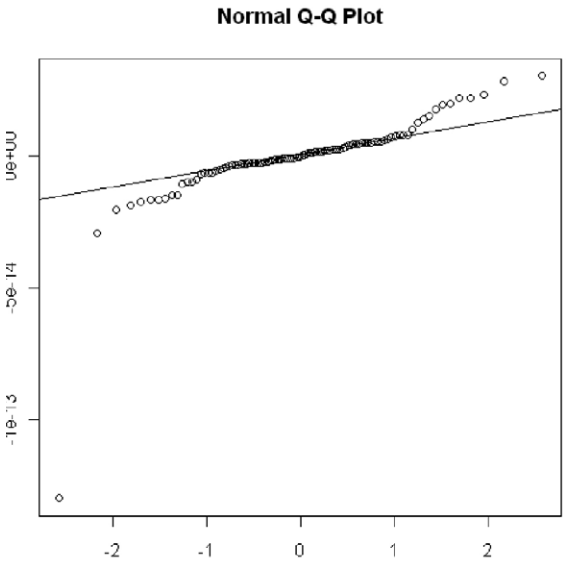
	Estimate	Std. Error	t value	Pr(>|t|)	
I(x^2)	1.000e+00	1.372e-15	7.289e+14	<2e-16	***
I(1/y)	1.000e+00	7.584e-15	1.319e+14	<2e-16	***
z	1.000e+00	2.278e-17	4.389e+16	<2e-16	***

Signif. codes: 0 '*' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1**

Residual standard error: 1.655e-14 on 97 degrees of freedom
Multiple R-squared: 1, Adjusted R-squared: 1
F-statistic: 1.148e+33 on 3 and 97 DF, p-value: < 2.2e-16

Note how the F-statistic is larger than the first model above and that the t values have greater significance as well. Note that the R-square statistic is also equal to one. Now, look at the Q-Q plot:

qqnorm(resid(Model3))
qqline(resid(Model3))



Notice how the tails are symmetric like a normal distribution. Also, both tails are slightly lighter than normal. But, the largest residual in absolute value is *-1.298e-13,* which is effectively zero, so we can say that this model is definitely very good.

If you want to see how R explains how to use the formulae format, please use the help command:

help(formula)

Another good resource on how to use the formulae format is "The R Book" by Michael J. Crawley. This book is published by Wiley and its ISBN is 978-0-470-51024-7.

In the next article, I will actually use R to create an efficient actuarial modeling technique by using my two most favorite non-parametric models with R. These models are the CLARA clustering algorithm and the Projection Pursuit Regression (PPR) predictive model. I will use CLARA to extract a small set of representative scenarios from a collection of 10,000 scenarios, and then I will use PPR to create a predictive model of specific corporate surplus results. I will also demonstrate the effectiveness of this combined approach when trying to quickly model the Conditional Tail Expectation (CTE) on the surplus. ■