Supplemental Instructions for Penalized Likelihood Binary Regression

This document provides additional information about how to specify interaction and polynomial terms when using the program for penalized likelihood (Firth) binary regression. You need to use the formatting I describe here because I make use of the average marginal effects program in R and it relies on specialized R formatting. It will probably help if you watch the video for the average marginal effect program. Be sure to read the instructions in the 'Description Box' for the current program. I also assume you are familiar with average marginal effects (AMEs) as discussed in my book.

Specifying Interaction Terms and Polynomials

Consider a model that predicts an outcome Y from two predictors X (the cause of Y) and Z (the moderator of the effect of X on Y). Assume I have one covariate that I will call COV. Normally you would create a product term in your data, that I will call PROD, and then execute the following equation, specified here in R format:

```
Y~X+Z+PROD+COV
```

This format DOES NOT apply to the Firth program and will yield incorrect results if you use it for the AME analyses within the program. The reason is because the value of PROD is dependent on the values of X and Z (you can't change Z without changing PROD and you can't change X without changing PROD) and this dependency must be taken into account in the program calculations of marginal effects.

To put the equation in a format that the program can accommodate, enter the component parts of the product term and then create a product term *within the equation specification* per the following:

```
Y~X+Z+X*Z+COV
```

That is, you enter the X*Z term in the equation, ignoring the PROD variable you may have created in your data set.[1]

For a three way interaction involving Q, X, and Z, the traditional approach is to create all possible two way interaction terms in your data set (QX,QZ,XZ) and a three way term (TW) defined as the product of QX, QZ, and XZ and form the equation:

```
Y~Q+X+Z+QX+QZ+XZ+TW+COV
```

For the current program, you instead specify the three way term *within the equation*:

```
Y~Q+X+Z+Q*X*Z+COV
```

[1] If you enter X*Z and not the component parts, R will automatically add the component parts to the equation. However, you need to enter the component parts of the product term for my interface to work given the quirks of my interface.

The specification of Q*X*Z will automatically generate all of the two-way interactions and include them in the model in addition to the three way interaction term. The analysis will then make the necessary adjustments given all the dependencies.

For a quadratic polynomial predicting Y from X, you traditionally would create a product term in your data that squares X (perhaps calling it XX) and then enter both terms in the equation:

```
Y~X+XX+COV
```

However, this format DOES NOT apply to the program and will yield incorrect results for the margins portion of the program because of the dependencies between the terms, i.e., you can't change X without changing XX and this dependency must be taken into account. Here is the format you would use

```
 Y~X+I(X^2)+COV
```

where I is the capital letter i. You enter the component part of the square term and then the notation I(*your component variable*^2). The symbol ^ is traditional computer programming lingo for the phrase "raise to the power of." In this case, we are raising X to the power of 2. In other words, rather than adding a polynomial term from your data set, you specify the polynomial *within the equation* using R notation.

In contrast to the interaction analysis, if you use a higher order function, such as a cubic polynomial, you must enter all of the lower order terms, such as

```
 Y~X+I(X^2)+I(X^3)+COV
```

or for a quartic polynomial

```
Y~X+I(X^2)+I(X^3)+I(X^4)+COV
```

See my book and the video for an explanation of why the margins output for this program only shows results for the component terms despite the fact you specified the full equation with product or quadratic terms.

The Importance of Specifying Categorical/Nominal Variables as Factors

If your categorical/nominal variable only has two levels, you can treat it as quantitative without consequence. For categorical/nominal variables with three or more levels, like ethnicity, the tradition is to represent it in a regression model using k-1 dummy variables, where k is the number of levels of the variable. For k = 3, I create the dummy variables in the data set, D1, D2, and D3 and then, using the first group as the reference group, enter two of the dummy variables into a regression equation to represent the variable, as follows:

```
Y~D2+D3
```

This representation does not work in the margins section of the current program because of the dependency between D2 and D3; one cannot alter D2 without also altering the other dummy variables. The programs need to take this dependency into account. By formally specifying ethnicity as a factor, the programs will generate the relevant dummy variables internally (using the first group as the reference group) and then take the requisite dependencies into account.

You can change the reference group by reordering the levels of the variable before executing the syntax I generate in the ‘Run Syntax’ box. I flag with a comment line (that starts with #) the command you need to change. The line uses the R command ‘factor’ in conjunction with ‘levels.’ Suppose the factor variable is called m4 with levels scored 1, 2 and 3. Normally, level 1 is used as the reference group. The command to add to the auto-generated syntax by my program if I want to use 2 as the reference group is

```
z1$m4 <- factor(z1$m4,levels=c(2,1,3))
```

One key term in this expression you need to attend to is the variable name after the $ sign. In this case it is m4. You will change m4 to the variable name that you want to designate as a factor, i.e., as being nominal/categorical. The other term are the numbers between c(). This reorders the levels to be 2,1,3 so that level 2 will now be the reference group because R by default uses the first group listed as the reference group. Again, in the syntax I generate, there will always be comment lines that show you exactly where you do the above.