Supplemental Instructions for Average Marginal Effects and for Profile Analysis Programs

This document provides information about (a) how to specify interaction and polynomial terms when using the average marginal effects and profile analysis programs, (b) the need to tell these programs that a given variable is categorical/nominal when the variable has three or more levels, (c) how to use the ‘At function’ in the programs, and (d) some “bugs” in the average marginal effects program. I assume you have watched the video for the programs and read the initial instructions in the ‘Description Box.’ I also assume you are familiar with average marginal effects (AMEs) and average adjusted predictions (AAPs).

Specifying Interaction Terms and Polynomials

The material in this section applies to both the AME and profile analysis programs. 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 two programs and will yield incorrect results if you use it. 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.

To put the equation in a format that the programs 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, contrary to this, 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 for AMEs 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 two programs and will yield incorrect results 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 output for interactions and polynomials for the AME program only shows results for the individual variables despite the fact you specified them in a larger equation with interaction effects or polynomials.

The Importance of Specifying Categorical/Nominal Variables as Factors

If your categorical/nominal variable only has two levels, you can treat it as quantitative in these programs 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 AME and PAA programs 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.

Using the ‘At function’ Command

I will first explain the structure of the ‘At function’ box for the profile analysis program. Much of it generalizes to the AME program.

Suppose a logistic regression with the outcome y has four predictors, two continuous mediators (m1 and m2, each ranging from -2 to +2), and three covariates, biological sex (0=female, 1=male), annual income (measured in dollars), and age. I treat sex as quantitative because it is binary. To calculate the predicted probability for people who are 30, you enter into the ‘At function’ box

```
age=30
```

The program then sets each case's age score to 30 and calculates the predicted probability of the outcome but using the original logistic equation, letting the other predictors take on whatever values they did for any given individual. Then, the program averages the probabilities. This is called the average adjusted prediction and is the predicted probability of people who are 30 years old.

You can have the program calculate several profiles in the same run. If I want to see three profiles for age (one for 30 years old, one for 40 years old, and one for 50 years old), I use the command

```
age=c(30,40,50)
```

The c( ) function tells R I want to operate on multiple values and the values occur within the parentheses, comma delimited.

I can specify a profile by setting the value of as many predictors as I want. For example, for age and sex, I might set age to 30 years old and sex to females. The program will then report the adjusted average prediction for 30 year old females. Here is the expression I use:

```
age=30,sex=0
```

I can use the c() function as well, such as

```
age=c(30,40,50),sex=(0,1)
```

This will produce results for six profiles: 30 year old females, 40 year old females, 50 year old females, 30 year old males, 40 year old males, and 50 year old males.

You can use any R mathematical operation when specifying a profile. To see the predicted probability when income and age are at theor mean values, use

income=mean(income),age=mean(age)

A function I find useful is the quantile function. For example,

income=quantile(income,0.50),age=quantile(age,0.50)

defines the profile setting income to the median income and age to the median age, i.e., the 0.50 quantile. If I want to define a profile based on the 90th quantile, I would use

income=quantile(income,0.90),age=quantile(age,0.90)

The general notation described above applies to nominal/qualitative variables that are declared as factors, but you must put the desired values in quotes. For example, if ethnicity is a three level nominal variable with the levels 1, 2, and 3, a profile for group 3 would be

ethnicity=”3”

and to generate three profiles

ethnicity=c(“1”,”2”,”3”)

Returning to the prior example, I can, of course, specify a profile for all predictors in the equation, such as

m1=0,m2=0,age=30,income=30000,sex=0

which will tell me the predicted probability for individuals who score 0 on m1, 0 on m2, who are 30 years old, female and who earn $30,000 per year.

Entries in the ‘At function’ box can get lengthy and I apologize for having to use a small display textbox for it to conform to my Input box format. You can use the left arrow and right arrow to scroll through the contents of the textbox. Clunky but workable.

Bugs in the ‘At function’ Command for the AME Program

The above description of the ‘At function’ works fine for the profile analysis program. However, there are some bugs in the R package ‘margins’ when applied to AMEs. Perhaps the maintainer of the program will fix these quirks in later releases of the package, so the bugs I am about to describe may not occur if this happens and you update your version of margins.

For quantitative variables, the ‘At function’ works fine. The bugs only occur for predictor variables that are declared as factors.

You will often get an error message unless you specify all the levels of the factor in the ‘At function’ box. For example,

ethnicity=”1”

will not work, but if I specify

ethnicity=c(“1”,”2”,”3”), the program will run and give me results for each level of ethnicity, even though I might only be interested in one of them.

Another bug is the numbering of the factor levels ***in the output***. For example, if the ethnicity level is scored 0, 1 and 2, in some cases, the program will renumber these in the output as 1, 2, and 3, i.e., the first category will always start with 1. If you use the ‘at function’, still use the categories 0, 1, and 2 if this is how the factor is scored – it is just that it will appear with the 1, 2, 3 notation on the output.

As best as I can tell, all the reported results in the AME program are accurate. It is only these specification and labeling quirks that occur and that you must be careful of. If you have no factor variables in the model, the bugs are irrelevant.