MWSUG 2016 - Paper BB23

Automating the Process of Generating Publication Quality Regression Tables through SAS® Base Programming

Ji Qi, University of Michigan Health System, Ann Arbor, MI

ABSTRACT

Regression modeling is a widely used statistical method for scientific investigation. A series of rules on how to report regression result have also been established to standardize the way it is presented in formal settings. The default output from major SAS/STAT® regression procedures, however, does not usually meet such requirement and is therefore not directly usable for final presentation. While it is acceptable to manually edit the raw output, such task can easily become tedious and error-prone when a large number of regression models are to be presented. This paper describes several SAS® Base programming techniques for automating the process of generating publication quality regression tables. Using the birth weight data set offered through the SAS.HELP library, we provide a detailed illustration on creating nicely-constructed output under the context of linear and logistic regression model. Key steps discussed here include applying formats to easily manipulate the reference group and displaying order for categorical predictors; Using Output Delivery System (ODS) to extract key component(s) of regression modeling results; and making use of various DATA step functions and statements to format regression statistics and/or transform regression tables. A macro program is further provided that integrates all the necessary steps to achieve complete automation. All the techniques mentioned in this paper can be easily extended to other commonly used regression procedures.

INTRODUCTION

Regression modeling, in its various forms, is a commonly used statistical method in scientific investigation. Depending on the specific research objective and hypothesis, regression analysis can be used to describe associations between different variables, perform predictions of outcome, or reveal causal relationship. Accurate reporting of results from regression models is therefore critical for researchers to effectively show significant findings to their audiences, and for the audiences to quickly gain insight.

Towards that end, a series of requirements on reporting regression models have been established to standardize the communication of statistical analysis results. While the exact format and requirement could vary by discipline, one way of summarizing analytic results can be illustrated in Table 1 below:

Variable		OR	95% CI	p-Value
Age		1.09	(1.06,1.13)	<.001
Years of education		1.01	(0.99,1.03)	0.497
Gender				
	Female		-Reference-	
	Male	1.59	(1.06, 2.39)	0.024
Household annual income				
	<50k		-Reference-	
	50k-99k	1.07	(0.72, 1.60)	0.721
	>=100k	3.14	(1.13, 8.68)	0.028

Table 1. A Possible Layout for Presenting Logistic Regression Model Results

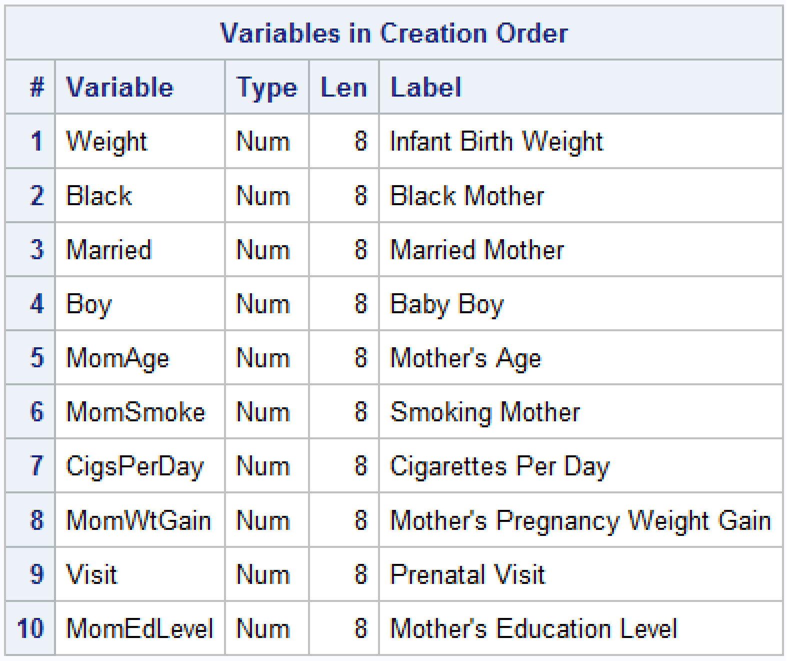
In this example, the goal is to present the result from a logistic regression model. The convention is to list exponentiated parameter estimates, such as odds ratios (ORs), rather than the parameter estimates themselves. The inclusion of 95% confidence intervals is also necessary because it reflects the precision of each parameter estimate. When confidence intervals are included, it is not necessary to also show p-values. But when presented, the actual observed values should be rounded to a certain number of decimal points. In terms of the general layout of the regression table, you usually just need one row for each continuous variable (e.g., age). For categorical variable, you need one row for the name of that variable (e.g., gender), and also additional rows for each level of the variable (e.g., female, male), including the reference category. It is also a good idea to clearly indicate which level is used as reference category. For linear regression model, you present the actual beta coefficient rather than OR, but other parts remain the same.

For most SAS/STAT® regression procedures, however, the default output is not organized in the way described above. To prepare for final presentation, SAS® users might have to export the raw output to an Excel sheet, and manually go through a series of steps to create a nicely formatted table. While such task is doable, it can easily become tedious and error-prone when you are to present a large number of regression models. Finding a solution to automate such process of generating regression tables will save the analyst considerable amount of time, enabling them to focus on other important parts of the research.

This paper attempts to build the connection between what SAS users need and what the software provides when it comes to presenting results from regression models. We discuss a series of data manipulation techniques or tips that help you turn a raw regression output into a nicely constructed table such as the one displayed in Table 1. Linear and logistic regression models are used as examples throughout the paper, as they are the two most commonly used statistical methods among researchers. However, all the techniques mentioned here can be easily extended to other regression context.

DATA SET AND REGRESSION MODEL

The data set used throughout this paper is the BWeight data provided in the SAS.HELP library. It contains information on 50,000 live, singleton births to mothers between the ages of 18 and 45 in the United States. Figure 1 below summarizes the variables included in this data set.



Variables in Creation Order

#	Variable	Type	Len	Label
1	Weight	Num	8	Infant Birth Weight
2	Black	Num	8	Black Mother
3	Married	Num	8	Married Mother
4	Boy	Num	8	Baby Boy
5	MomAge	Num	8	Mother's Age
6	MomSmoke	Num	8	Smoking Mother
7	CigsPerDay	Num	8	Cigarettes Per Day
8	MomWtGain	Num	8	Mother's Pregnancy Weight Gain
9	Visit	Num	8	Prenatal Visit
10	MomEdLevel	Num	8	Mother's Education Level

Figure 1. Variable List for BWeight Data Set

Linear and logistic regression models are used here as examples to demonstrate the process of generating regression tables. For linear regression model, the dependent variable is baby's weight. For logistic regression model, the dependent variable is an indicator on whether the baby has low birth weight (defined as having a birth weight less than 2,500 grams). All the other variables contained in the BWeight data set are included as independent variables for both models. These two models are implemented using the GLM procedure and the LOGISTIC procedure, respectively, as shown below:

```
proc glm data=sashelp.bweight;
    class Black Married Boy MomSmoke Visit MomEdLevel;
    model weight = MomAge CigsPerDay MomWtGain Black Married Boy MomSmoke
Visit MomEdLevel / solution;
run;
```

```
data bw;
   set sashelp.bweight; if Weight<2500 then low=1; else low=0;
run;

proc logistic data=bw;
    class Black Married Boy MomSmoke Visit MomEdLevel / param=ref;
    model low(event=”1”) = MomAge CigsPerDay MomWtGain Black Married Boy
MomSmoke Visit MomEdLevel;
run;
```

Below we first walk through each step of generating regression tables within the setting of linear regression model (PROC GLM), during which we provide detailed explanation on the issues to consider and the corresponding solutions. Later we move on to logistic regression (PROC LOGISTIC) and highlight the parts that are different and need special attention.

MANIPULATING CATEGORICAL VARIABLES

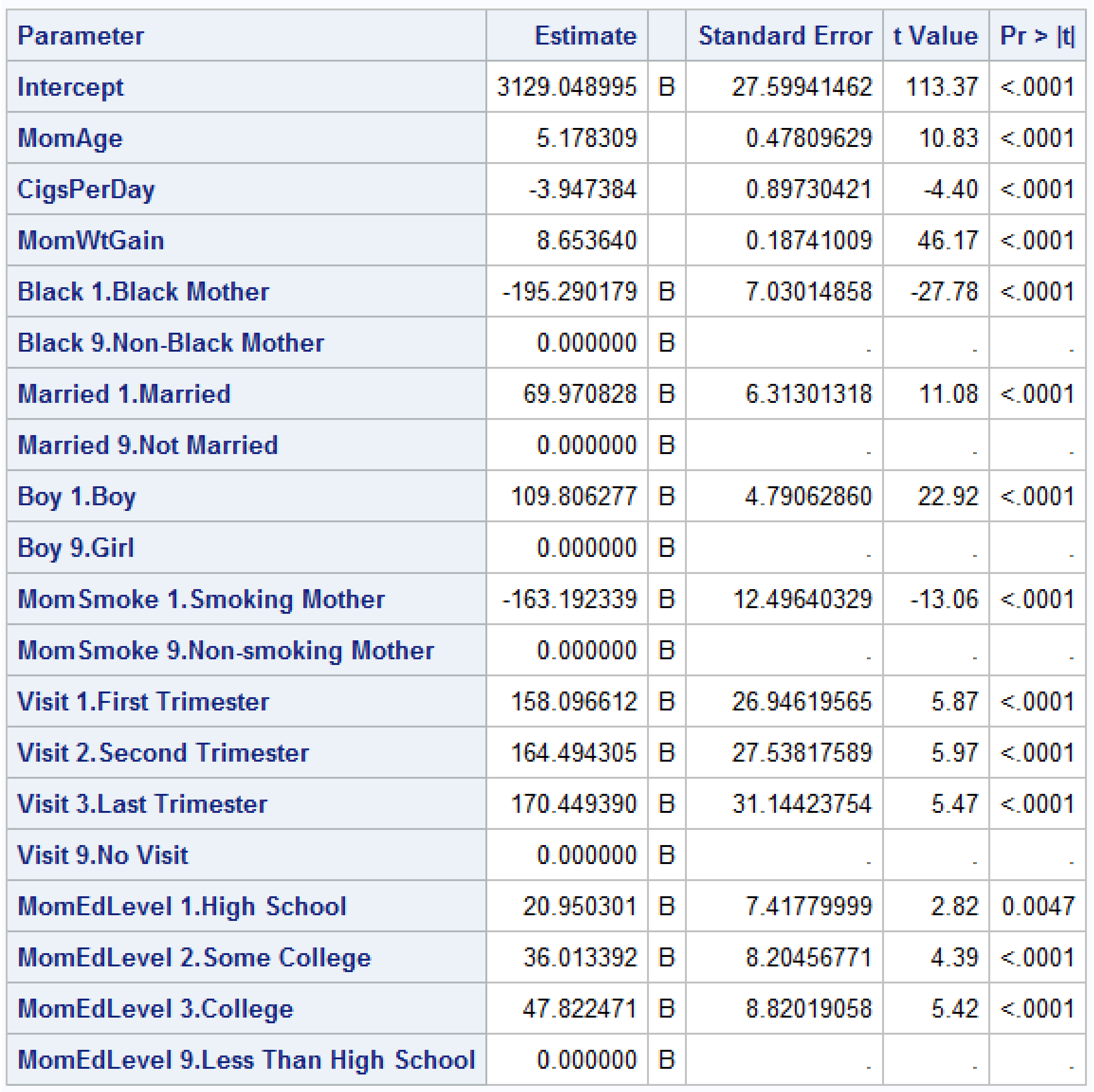
For categorical variables included in the regression model, you need to decide which level of each variable is used as the reference group. For variables having more than two levels, i.e. nominal or ordinal variables, another potential issue to consider is the order in which each level of the variable is listed in the table. By default, SAS determines the reference level of a variable by first sorting the values of that variable in ascending order, and assigning the last ordered value as the reference group. For character variables, the sorting is based on alphabetical order. For example, in the BWeight data set, the default reference group for variable Visit is the one coded as 3 (corresponds to “First Trimester”), because it is the largest value across all levels.

Such default setting might not be ideal when the users have specific preference on which level of a variable should be used as reference. A simple way to easily manipulate this is to assign a format to each categorical variable through the FORMAT procedure and FORMAT statement, and then use ORDER=FORMATTED option in PROC GLM statement:

```
proc format;
   value Black 1="1.Black Mother" 0="9.Non-Black Mother" ;
   value Married 1="1.Married" 0="9.Not Married";
   value Boy 1="1.Boy" 0="9.Girl";
   value MomSmoke 1="1.Smoking Mother" 0="9.Non-smoking Mother";
   value Visit 0="9.No Visit" 1="2.Second Trimester" 2="3.Last Trimester"
               3="1.First Trimester";
   value MomEdLevel 0="1.High School" 1="2.Some College" 2="3.College"
                    3="9.Less Than High School";
run;

proc glm data=sashelp.bweight order=formatted;
    class Black Married Boy MomSmoke Visit MomEdLevel;
    model weight = MomAge CigsPerDay MomWtGain Black Married Boy MomSmoke
Visit MomEdLevel / solution;
    format Black Black. Married Married. Boy Boy. MomSmoke Momsmoke. Visit
Visit. MomEdLevel MomEdLevel.;
run;
```

Here ORDER=FORMATTED option tells SAS to sort each categorical variable based on their formatted value instead of the original coded value, with the largest formatted value being the reference group. Under this setup, you just need to make sure that your choice of the reference level for each variable is assigned the largest value label in PROC FORMAT. As shown in the code above, a simple trick is to include a numeric prefix in each value label, with the reference level getting a relatively large number such as 9. For example, for variable MomEdLevel, you want “Less than high school” as the reference level, so you include number 9 at the beginning of the value label. For other levels of MomEdLevel, you assign a smaller number, such as 1-3, as the prefix of the value label. In this particular example, creating a format or not won't make a difference, because the initial coding of MomEdLevel is in a way that "Less than high school" gets the largest numeric number. However, if later for whatever reason you decide to change the reference group to "College", all you need to do is changing the numeric prefix. Therefore, having a format ready for each variable can help you quickly switch reference group.

Figure 2 shows the partial output PROC GLM generated after running the code above. Note that the reference group for each variable is assigned in the same way as we expect. Moreover, by including this numeric prefix to the value label, you further have explicit control over how other non-reference levels are displayed in the output table. Again take variable MomEdLevel as an example, "Less than high school" is used as the reference category, and the output table first displays the row for "High school", followed by "Some college" and finally "College". If you want a different order of these three levels, just play with the numeric prefix and the output will reflect such change accordingly.



Parameter	Estimate		Standard Error	t Value	Pr > |t|
Intercept	3129.048995	B	27.59941462	113.37	<.0001
MomAge	5.178309		0.47809629	10.83	<.0001
CigsPerDay	-3.947384		0.89730421	-4.40	<.0001
MomWtGain	8.653640		0.18741009	46.17	<.0001
Black 1.Black Mother	-195.290179	B	7.03014858	-27.78	<.0001
Black 9.Non-Black Mother	0.000000	B	.	.	.
Married 1.Married	69.970828	B	6.31301318	11.08	<.0001
Married 9.Not Married	0.000000	B	.	.	.
Boy 1.Boy	109.806277	B	4.79062860	22.92	<.0001
Boy 9.Girl	0.000000	B	.	.	.
MomSmoke 1.Smoking Mother	-163.192339	B	12.49640329	-13.06	<.0001
MomSmoke 9.Non-smoking Mother	0.000000	B	.	.	.
Visit 1.First Trimester	158.096612	B	26.94619565	5.87	<.0001
Visit 2.Second Trimester	164.494305	B	27.53817589	5.97	<.0001
Visit 3.Last Trimester	170.449390	B	31.14423754	5.47	<.0001
Visit 9.No Visit	0.000000	B	.	.	.
MomEdLevel 1.High School	20.950301	B	7.41779999	2.82	0.0047
MomEdLevel 2.Some College	36.013392	B	8.20456771	4.39	<.0001
MomEdLevel 3.College	47.822471	B	8.82019058	5.42	<.0001
MomEdLevel 9.Less Than High School	0.000000	B	.	.	.

Figure 2. Partial output from PROC GLM

Another helpful trick during this step is to make sure that you specify the format name to be the same as the variable name in the PROC FORMAT. As we show later in the paper, this can greatly facilitate the automation of generating regression tables.

OUTPUT DELIVERY SYSTEM

When you submit a statistical modeling procedure such as PROC GLM, SAS not only displays key results in the output window, but also creates a series of tables behind the scene. These tables store different parameters or statistics calculated from the regression model you are building, and you can select these tables and output them into separate data sets. Once a data set is created from such table, you can apply many DATA step programming techniques to perform data transformation. This makes it possible to automatically create a nicely formatted regression tables.

You can use the Output Delivery System (ODS) to select a given table and create a corresponding output data set. SAS assigns a name to each table it creates, and you can use this name to refer to the table you are targeting. In our example, we want a table that contains the information on regression coefficient and hopefully other related information such as p-value and 95% confidence interval (CI) associated with each estimate. This table is named ParameterEstimates in PROC GLM, so here is the code:

```
proc glm data=sashelp.bweight order=formatted;
    class Black Married Boy MomSmoke Visit MomEdLevel;
    model weight = MomAge CigsPerDay MomWtGain Black Married Boy MomSmoke
Visit MomEdLevel / solution CLPARM;
    format Black Black. Married Married. Boy Boy. MomSmoke Momsmoke. Visit
Visit. MomEdLevel MomEdLevel.;
    ods output ParameterEstimates=Beta;
run;
```

In the ODS OUTPUT statement, you put the name of the table, ParameterEstimates, on the left side of the equal sign. Then you specify a valid SAS data set name on the right side. In our example, the information contained in the table ParameterEstimates is output to the data set called Beta. You might wonder how we know the name ParameterEstimates. Such information can be obtained by referring to the SAS online documentation. In practice, simply search for “ods output PROC GLM” in Google, for example, and you will get a full list of names and contents of tables SAS creates under PROC GLM.

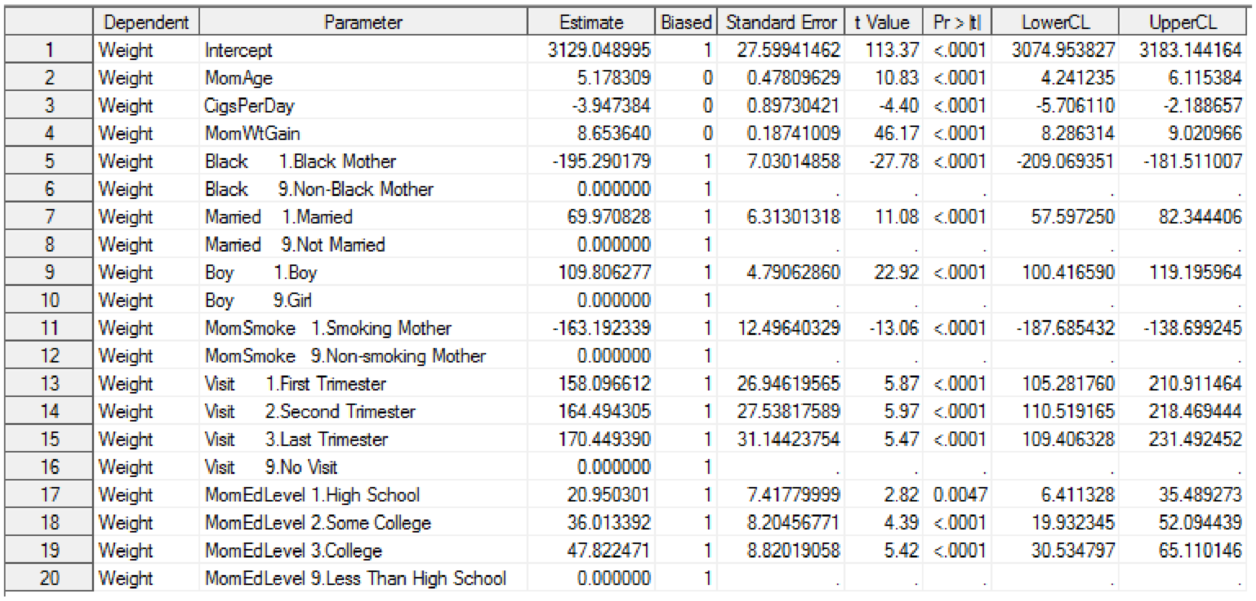
Note that in order to successfully create the output data set from ODS, you might need to specify additional options within other statements of the procedure. In the example above, we have specified two options in the GLM MODEL statement. The first one is SOLUTION, which refers to the estimated regression coefficients of the linear model we build. Without this option, you will not see the corresponding output in the results window, and the ODS fails to create data set Beta. The second option is CLPARM, which tells SAS to include the lower and upper limit of the 95% CIs to the output. This information is necessary for final result presentation.

DATA STEP PROGRAMMING ON TABLE FORMATTING

The output data set (Beta) obtained through ODS has very similar structure as the default regression output SAS displays in the output window, neither of which is directly usable for final presentation. However, because such information is now stored in a SAS data set, we can use a series of DATA step programming to change its layout. Figure 3 shows the initial configuration of data set Beta.

Given how the information is organized in this raw output data set, we can start with the following code to format the table:

```
data Beta1(drop=LowerCL UpperCL);
  set Beta(firstobs=2 keep=Parameter Estimate Probt--UpperCL);

  if LowerCL ne . then
  CI="("||strip(put(LowerCL,7.2))||","||" "||strip(put(UpperCL,7.2))||")";
  else CI="-Reference-";
run;
```



	Dependent	Parameter	Estimate	Biased	Standard Error	t Value	Pr > |t|	LowerCL	UpperCL
1	Weight	Intercept	3129.048995	1	27.59941462	113.37	<.0001	3074.953827	3183.144164
2	Weight	MomAge	5.178309	0	0.47809629	10.83	<.0001	4.241235	6.115384
3	Weight	CigsPerDay	-3.947384	0	0.89730421	-4.40	<.0001	-5.706110	-2.188657
4	Weight	MomWtGain	8.653640	0	0.18741009	46.17	<.0001	8.286314	9.020966
5	Weight	Black 1.Black Mother	-195.290179	1	7.03014858	-27.78	<.0001	-209.069351	-181.511007
6	Weight	Black 9.Non-Black Mother	0.000000	1	.	.	.	.	.
7	Weight	Married 1.Married	69.970828	1	6.31301318	11.08	<.0001	57.597250	82.344406
8	Weight	Married 9.Not Married	0.000000	1	.	.	.	.	.
9	Weight	Boy 1.Boy	109.806277	1	4.79062860	22.92	<.0001	100.416590	119.195964
10	Weight	Boy 9.Girl	0.000000	1	.	.	.	.	.
11	Weight	MomSmoke 1.Smoking Mother	-163.192339	1	12.49640329	-13.06	<.0001	-187.685432	-138.699245
12	Weight	MomSmoke 9.Non-smoking Mother	0.000000	1	.	.	.	.	.
13	Weight	Visit 1.First Trimester	158.096612	1	26.94619565	5.87	<.0001	105.281760	210.911464
14	Weight	Visit 2.Second Trimester	164.494305	1	27.53817589	5.97	<.0001	110.519165	218.469444
15	Weight	Visit 3.Last Trimester	170.449390	1	31.14423754	5.47	<.0001	109.406328	231.492452
16	Weight	Visit 9.No Visit	0.000000	1	.	.	.	.	.
17	Weight	MomEdLevel 1.High School	20.950301	1	7.41779999	2.82	0.0047	6.411328	35.489273
18	Weight	MomEdLevel 2.Some College	36.013392	1	8.20456771	4.39	<.0001	19.932345	52.094439
19	Weight	MomEdLevel 3.College	47.822471	1	8.82019058	5.42	<.0001	30.534797	65.110146
20	Weight	MomEdLevel 9.Less Than High School	0.000000	1	.	.	.	.	.

Figure 3. Data Set Beta

To begin with, we read in the Beta data set by specifying the FIRSTOBS=2 data set option. This prevents SAS from reading the very first row of the data, which contains the regression estimate for the intercept — something that is usually omitted in the final presentation.

The first thing you can do is generating a formatted 95% CI for each regression estimate. As shown in Figure 3, the output data set stores such information in two separate columns, one for the lower limit (LowerCL) and one for the upper limit (UpperCL). Using string functions is a good way to combine information that spreads across different columns. More specifically, it involves the following steps:

1) using the PUT function to perform explicit numeric to character conversion of variable LowerCL and UpperCL, with the second argument (format) of the function being 7.2, so that two decimal points are displayed after the conversion;

2) using the STRIP function to remove any leading and trailing blanks of the character string obtained from the PUT function, which prevents extra blanks in the final confidence interval string;

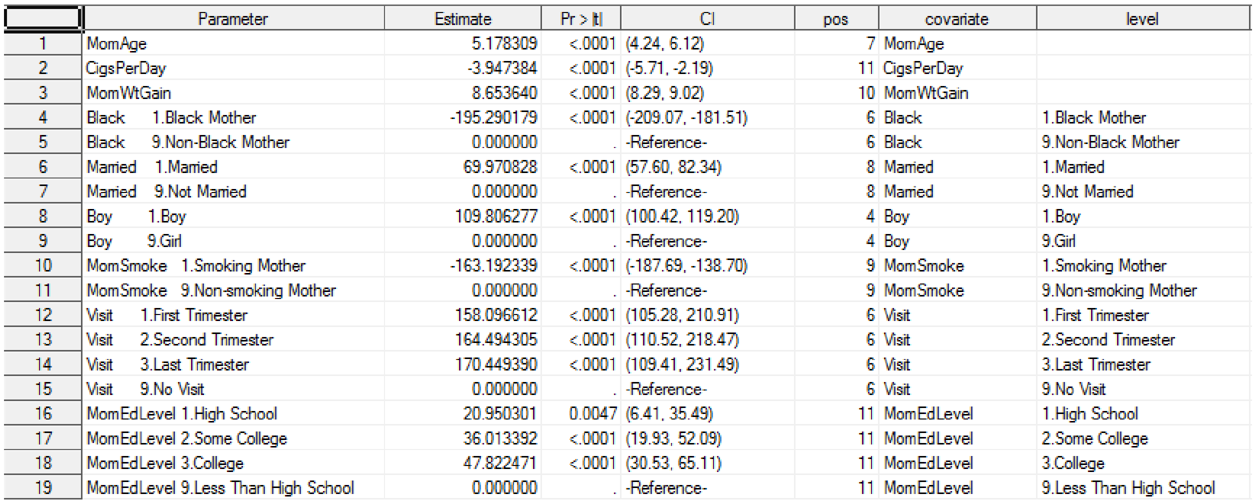
3) using the concatenation operator (a pair of vertical bars) to combine the aforementioned strings and other symbols (parenthesis, comma, blank) in order to construct the final CI;

For reference levels, instead of generating a corresponding CI, we just specify it to be the string "-Reference-". This will make the final table easier to read.

Note that in Figure 3, variable Parameter stores information on the variables used in the regression model. For categorical variables, both the variable name and the formatted value label of the variable are included in the string. For the final presentation, we prefer to have only the labels because they are more descriptive. We can use the following code to separate variable name from its label:

```
data Beta1(drop=LowerCL UpperCL);
  set Beta(firstobs=2 keep=Parameter Estimate Probt--UpperCL);

  if LowerCL ne . then
  CI="("||strip(put(LowerCL,7.2))||","||" "||strip(put(UpperCL,7.2))||")";
  else CI="-Reference-";

  pos=index(Parameter,"");
  covariate=substr(Parameter,1,pos-1);
  level=strip(substr(Parameter,pos));
run;
```

The trick here is to realize that for each string of the Parameter variable, there exists at least one blank between variable name and its label. We can use this feature to extract both the variable name and the label by first figuring out the position of the first blank within the string. This is accomplished by using the INDEX function. The INDEX function searches a character value (the first argument) for a specified string (the second argument), and returns the position of the string's first character. In our case, we just want to search each string of Parameter for blank, so the second argument of the function only contains quotation marks. The position of the first blank in each string is stored in variable pos, which is further used in the SUBSTR function to extract the information on both the variable name and the value label. Figure 4 shows the new variables created during this step (CI, pos, covariate, level).



	Parameter	Estimate	Pr > |t|	CI	pos	covariate	level
1	MomAge	5.178309	<.0001	(4.24, 6.12)	7	MomAge	
2	CigsPerDay	-3.947384	<.0001	(-5.71, -2.19)	11	CigsPerDay	
3	MomWtGain	8.653640	<.0001	(8.29, 9.02)	10	MomWtGain	
4	Black 1.Black Mother	-195.290179	<.0001	(-209.07, -181.51)	6	Black	1.Black Mother
5	Black 9.Non-Black Mother	0.000000	.	-Reference-	6	Black	9.Non-Black Mother
6	Married 1.Married	69.970828	<.0001	(57.60, 82.34)	8	Married	1.Married
7	Married 9.Not Married	0.000000	.	-Reference-	8	Married	9.Not Married
8	Boy 1.Boy	109.806277	<.0001	(100.42, 119.20)	4	Boy	1.Boy
9	Boy 9.Girl	0.000000	.	-Reference-	4	Boy	9.Girl
10	MomSmoke 1.Smoking Mother	-163.192339	<.0001	(-187.69, -138.70)	9	MomSmoke	1.Smoking Mother
11	MomSmoke 9.Non-smoking Mother	0.000000	.	-Reference-	9	MomSmoke	9.Non-smoking Mother
12	Visit 1.First Trimester	158.096612	<.0001	(105.28, 210.91)	6	Visit	1.First Trimester
13	Visit 2.Second Trimester	164.494305	<.0001	(110.52, 218.47)	6	Visit	2.Second Trimester
14	Visit 3.Last Trimester	170.449390	<.0001	(109.41, 231.49)	6	Visit	3.Last Trimester
15	Visit 9.No Visit	0.000000	.	-Reference-	6	Visit	9.No Visit
16	MomEdLevel 1.High School	20.950301	0.0047	(6.41, 35.49)	11	MomEdLevel	1.High School
17	MomEdLevel 2.Some College	36.013392	<.0001	(19.93, 52.09)	11	MomEdLevel	2.Some College
18	MomEdLevel 3.College	47.822471	<.0001	(30.53, 65.11)	11	MomEdLevel	3.College
19	MomEdLevel 9.Less Than High School	0.000000	.	-Reference-	11	MomEdLevel	9.Less Than High School

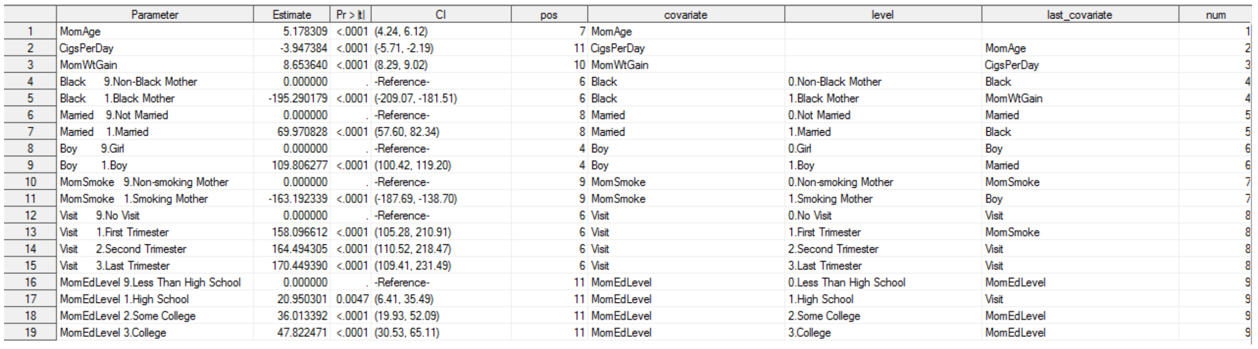
Figure 4. Data Set Beta1

Up until this point, we pretty much have gathered everything we need for the regression table: variable name and its label, coefficient estimate, 95% CI and p-value. However, as shown in Figure 4, one potential issue is that for each categorical variable, the reference group is listed in the end. For final presentation, the convention is to list the reference group first within each predictor. The code below is used to modify the relative order of reference level, while still maintaining the overall predictor sequence:

```
data Beta1(drop=LowerCL UpperCL);
  set Beta(firstobs=2 keep=Parameter Estimate Probt--UpperCL);

  if LowerCL ne . then
  CI="("||strip(put(LowerCL,7.2))||","||" "||strip(put(UpperCL,7.2))||")";
  else CI="-Reference-";

  pos=index(Parameter,"");
  covariate=substr(Parameter,1,pos-1);
  level=strip(substr(Parameter,pos));

  level = tranwrd(level,"9","0");
  last_covariate=lag(covariate);
  if covariate ne last_covariate then num+1;
run;

proc sort data=Beta1; by num level; run;
```

Here we first use the TRANWRD function, which replaces certain characters within a string. The first argument specifies the string you are working on, the second argument specifies the targeted string you want to search for and get replaced, the third argument specifies the string that replaces the target. Earlier when creating formats for categorical variables, we assigned a large numeric prefix, 9, to each reference group so that PROC GLM correctly treats it as the baseline level under the ORDER=FORMATTED option. What we do here is to change that prefix from 9 to 0, so that later when we sort the data by its label, each reference group will move on to the top row because now it has the smallest formatted value.

Next we create another variable called num, which serves as a way of maintaining the relative order of predictor variables (i.e. each unique value in the variable covariate) listed in the regression table. The trick here is to number each unique value of covariate in an ascending order. As is shown in the code above, this is accomplished by comparing the current observation of covariate with its previous observation (obtained using the LAG function), and add 1 to the value of num (through the SUM statement) if the two observations do not match— the place where the value of covariate changes. Then once you sort the data by num and level, the reference group for each categorical variable is listed first, and the order of predictor variables remains unchanged (Figure 5).



	Parameter	Estimate	Pr > |t|	CI	pos	covariate	level	last_covariate	num
1	MomAge	5.178309	<.0001	(4.24, 6.12)	7	MomAge			1
2	CigsPerDay	-3.947384	<.0001	(-5.71, -2.19)	11	CigsPerDay		MomAge	2
3	MomWtGain	8.653640	<.0001	(8.29, 9.02)	10	MomWtGain		CigsPerDay	3
4	Black 9.Non-Black Mother	0.000000	.	-Reference-	6	Black	0.Non-Black Mother	Black	4
5	Black 1.Black Mother	-195.290179	<.0001	(-209.07, -181.51)	6	Black	1.Black Mother	MomWtGain	4
6	Married 9.Not Married	0.000000	.	-Reference-	8	Married	0.Not Married	Married	5
7	Married 1.Married	69.970828	<.0001	(57.60, 82.34)	8	Married	1.Married	Black	5
8	Boy 9.Girl	0.000000	.	-Reference-	4	Boy	0.Girl	Boy	6
9	Boy 1.Boy	109.806277	<.0001	(100.42, 119.20)	4	Boy	1.Boy	Married	6
10	MomSmoke 9.Non-smoking Mother	0.000000	.	-Reference-	9	MomSmoke	0.Non-smoking Mother	MomSmoke	7
11	MomSmoke 1.Smoking Mother	-163.192339	<.0001	(-187.69, -138.70)	9	MomSmoke	1.Smoking Mother	Boy	7
12	Visit 9.No Visit	0.000000	.	-Reference-	6	Visit	0.No Visit	Visit	8
13	Visit 1.First Trimester	158.096612	<.0001	(105.28, 210.91)	6	Visit	1.First Trimester	MomSmoke	8
14	Visit 2.Second Trimester	164.494305	<.0001	(110.52, 218.47)	6	Visit	2.Second Trimester	Visit	8
15	Visit 3.Last Trimester	170.449390	<.0001	(109.41, 231.49)	6	Visit	3.Last Trimester	Visit	8
16	MomEdLevel 9.Less Than High School	0.000000	.	-Reference-	11	MomEdLevel	0.Less Than High School	MomEdLevel	9
17	MomEdLevel 1.High School	20.950301	0.0047	(6.41, 35.49)	11	MomEdLevel	1.High School	Visit	9
18	MomEdLevel 2.Some College	36.013392	<.0001	(19.93, 52.09)	11	MomEdLevel	2.Some College	MomEdLevel	9
19	MomEdLevel 3.College	47.822471	<.0001	(30.53, 65.11)	11	MomEdLevel	3.College	MomEdLevel	9

Figure 5. Data Set Beta1 after Re-ordering of Rows

At this point, the Beta1 data set is already a fairly good candidate to use for final presentation. However, before printing it out, we would like to do some final adjustments. This includes:

1) Separating each categorical variable with a blank line;

2) Removing the numeric prefix in the categorical variable label;

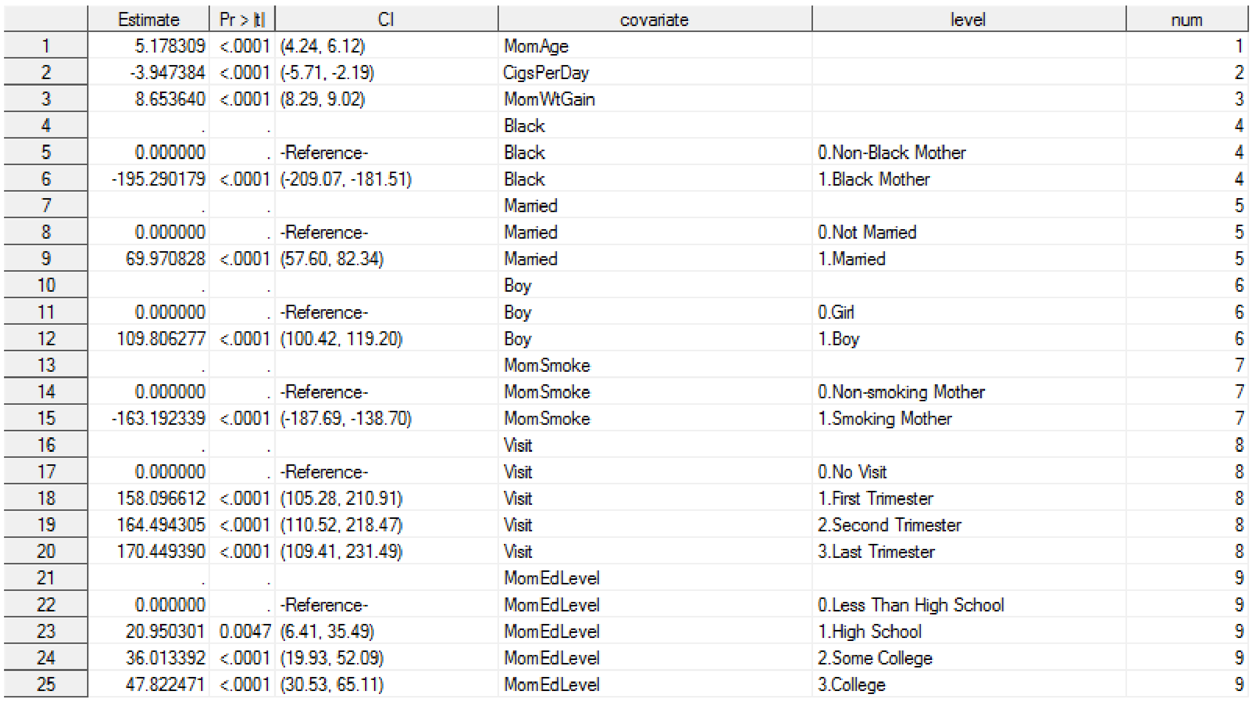
3) Formatting regression coefficient, p-value and the column header;

The reason for 1) is to make the final table look less crowded and therefore easier to read. Earlier when creating formats, we deliberately include a numeric prefix to the value label so that we can easily manipulate the reference group. For the final presentation, however, we do not want to show this number, so 2) is necessary. As for 3), depending on the situation the requirement can be different. In our example, we choose to retain two decimal points for the regression coefficient, and three decimal points for the p-value. Moreover, for p-values that are smaller than 0.001, we would like to display it as "<.001".

To create the blank line, we use the code below:

```
proc sql;
   create table blank as
         select covariate, num from Beta1
               where probt=.;
   quit;

data combine(keep=Estimate Probt CI covariate level num);
   set Beta1 blank;
run;

proc sort data=combine; by num level; run;
```

In the SQL procedure above, we select only those rows with missing p-value, which gives us a list of all the categorical variables. Then we append this blank data set to the previous Beta1 data set. During the appending process, because data set blank only contains two variables (covariate and num), variable level is set to missing for all the observations coming from that data set. As a result, when we sort the appended data set again by num and level, the additional blank line is listed first within each categorical variable (Figure 6), because missing value is considered the smallest in SAS.



	Estimate	Pr > |t|	CI	covariate	level	num
1	5.178309	<.0001	(4.24, 6.12)	MomAge		1
2	-3.947384	<.0001	(-5.71, -2.19)	CigsPerDay		2
3	8.653640	<.0001	(8.29, 9.02)	MomWtGain		3
4	.	.		Black		4
5	0.000000	.	-Reference-	Black	0.Non-Black Mother	4
6	-195.290179	<.0001	(-209.07, -181.51)	Black	1.Black Mother	4
7	.	.		Married		5
8	0.000000	.	-Reference-	Married	0.Not Married	5
9	69.970828	<.0001	(57.60, 82.34)	Married	1.Married	5
10	.	.		Boy		6
11	0.000000	.	-Reference-	Boy	0.Girl	6
12	109.806277	<.0001	(100.42, 119.20)	Boy	1.Boy	6
13	.	.		MomSmoke		7
14	0.000000	.	-Reference-	MomSmoke	0.Non-smoking Mother	7
15	-163.192339	<.0001	(-187.69, -138.70)	MomSmoke	1.Smoking Mother	7
16	.	.		Visit		8
17	0.000000	.	-Reference-	Visit	0.No Visit	8
18	158.096612	<.0001	(105.28, 210.91)	Visit	1.First Trimester	8
19	164.494305	<.0001	(110.52, 218.47)	Visit	2.Second Trimester	8
20	170.449390	<.0001	(109.41, 231.49)	Visit	3.Last Trimester	8
21	.	.		MomEdLevel		9
22	0.000000	.	-Reference-	MomEdLevel	0.Less Than High School	9
23	20.950301	0.0047	(6.41, 35.49)	MomEdLevel	1.High School	9
24	36.013392	<.0001	(19.93, 52.09)	MomEdLevel	2.Some College	9
25	47.822471	<.0001	(30.53, 65.11)	MomEdLevel	3.College	9

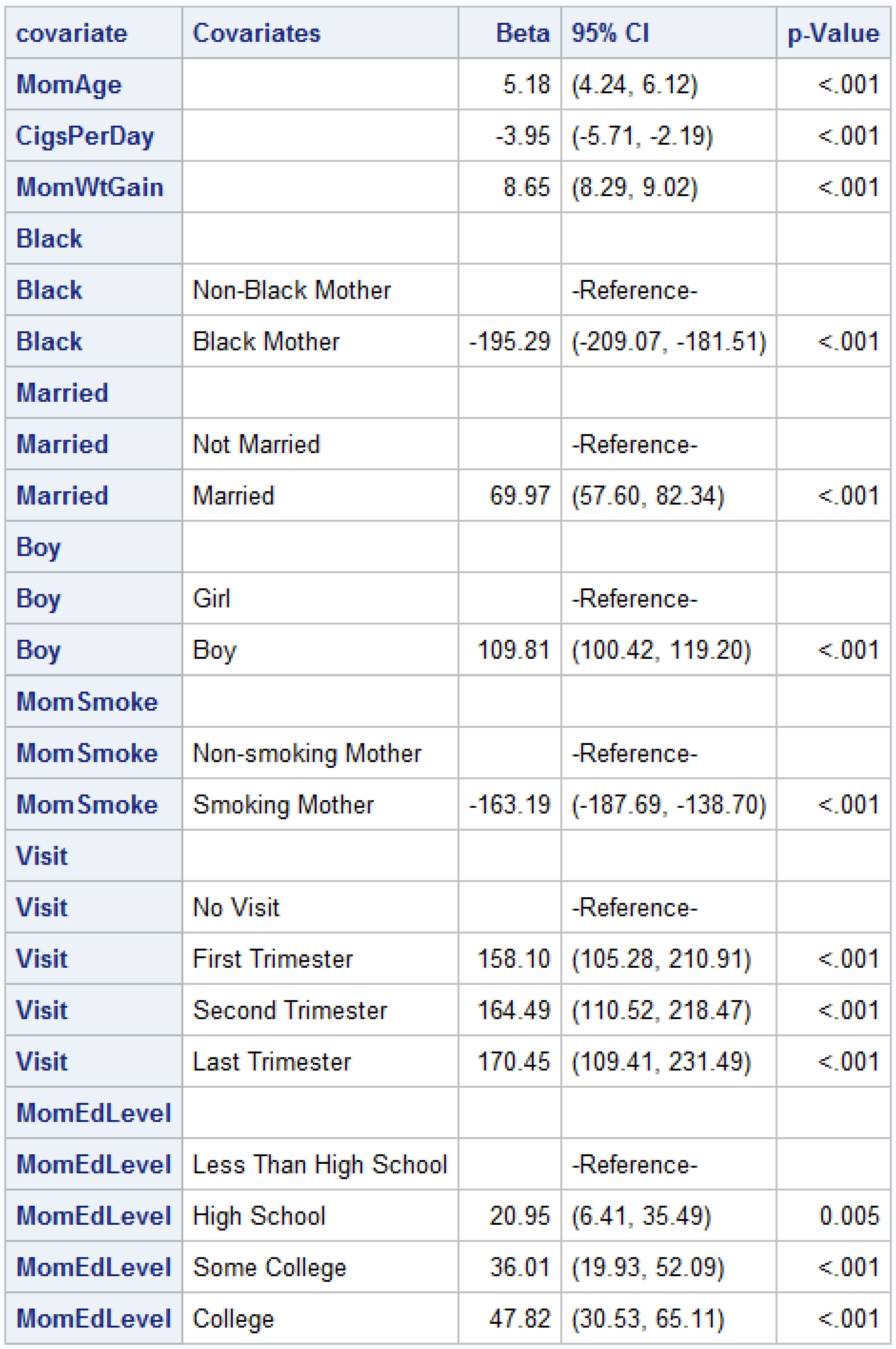
Figure 6. Data Set Combine

The code below does some final formatting tasks. The SUBSTR function is used to get rid of the numeric prefix for variable level. The LABEL statement is used to assign a more interpretable name to each column. When it comes to formatting regression coefficient and p-value, one thing to note here is that both variables contain some special values. As is shown in Figure 6, regression coefficient for each reference group is listed as 0.000000, with a missing p-value represented by a dot. For each blank line we just add, both numbers are also missing and listed as dot. To avoid displaying such undesired numbers/symbols in the final table, we can combine both user-defined formats and SAS system formats. For example, for regression coefficient, we assign a label with no content (a blank within the quotation mark) to 0 and missing value (denoted by the keyword OTHER). For other values, we apply a SAS system format with length 7 and two decimal points. Note that when you are referring to another existing format within the VALUE statement, make sure to enclose that format with square bracket. Otherwise SAS will treat the format, which in this case is 7.2, as a value label, resulting in all the 0s and missing values being displayed as “7.2”.

```
data combine1;
    set combine;
    x=substr(level,3);
    level=x;
    label level="Covariates" estimate="Beta" CI="95% CI" probt="p-Value";
run;

proc format;
    value estimate low-<0 = [7.2] 0 = " " 0<-high = [7.2] other = " ";
    value p 0-<0.001 = "<.001" 0.001-1 = [5.3] other = " " ;
run;
```

Now we are finally ready to print the table. Running the code below, we get the output in Figure 7.

```
proc print data=combine1 label;
    id covariate;
    var level estimate CI probt;
    format estimate estimate. probt p.;
run;
```



covariate	Covariates	Beta	95% CI	p-Value
MomAge		5.18	(4.24, 6.12)	<.001
CigsPerDay		-3.95	(-5.71, -2.19)	<.001
MomWtGain		8.65	(8.29, 9.02)	<.001
Black				
Black	Non-Black Mother		-Reference-	
Black	Black Mother	-195.29	(-209.07, -181.51)	<.001
Married				
Married	Not Married		-Reference-	
Married	Married	69.97	(57.60, 82.34)	<.001
Boy				
Boy	Girl		-Reference-	
Boy	Boy	109.81	(100.42, 119.20)	<.001
MomSmoke				
MomSmoke	Non-smoking Mother		-Reference-	
MomSmoke	Smoking Mother	-163.19	(-187.69, -138.70)	<.001
Visit				
Visit	No Visit		-Reference-	
Visit	First Trimester	158.10	(105.28, 210.91)	<.001
Visit	Second Trimester	164.49	(110.52, 218.47)	<.001
Visit	Last Trimester	170.45	(109.41, 231.49)	<.001
MomEdLevel				
MomEdLevel	Less Than High School		-Reference-	
MomEdLevel	High School	20.95	(6.41, 35.49)	0.005
MomEdLevel	Some College	36.01	(19.93, 52.09)	<.001
MomEdLevel	College	47.82	(30.53, 65.11)	<.001

Figure 7. Final Formatted Regression Table

You can then copy and paste the above output to Excel, after which all you need to do is adding the name of each predictor to the table. In the end, you will get something like **Error! Reference source not found.** below.

Covariates		Beta	95% CI	p-Value
Mom's Age		5.18	(4.24, 6.12)	<.001
Cigarettes per Day		-3.95	(-5.71, -2.19)	<.001
Mom's Weight Gain		8.65	(8.29, 9.02)	<.001
Race				
	Non-Black Mother		-Reference-	
	Black Mother	-195.29	(-209.07, -181.51)	<.001
Marital Status				
	Not Married		-Reference-	
	Married	69.97	(57.60, 82.34)	<.001
Baby's Gender				
	Girl		-Reference-	
	Boy	109.81	(100.42, 119.20)	<.001
Smoking Status				
	Non-smoking Mother		-Reference-	
	Smoking Mother	-163.19	(-187.69, -138.70)	<.001
Prenatal Visit				
	No Visit		-Reference-	
	First Trimester	158.10	(105.28, 210.91)	<.001
	Second Trimester	164.49	(110.52, 218.47)	<.001
	Last Trimester	170.45	(109.41, 231.49)	<.001
Education				
	Less Than High School		-Reference-	
	High School	20.95	(6.41, 35.49)	0.005
	Some College	36.01	(19.93, 52.09)	<.001
	College	47.82	(30.53, 65.11)	<.001

Table 2. Final Regression Table after Minimal Adjustment

MACRO FOR GENERATING REGRESSION TABLE

It is a good idea to create a macro program that stores all the SAS codes we just talked about on making regression tables, so that you can easily accomplish similar tasks in the future. The macro glm_table listed below is one such attempt.

There are four parameters included in the macro definition: dataset parameter specifies the data set on which the regression model is running; y parameter specifies the dependent variable for the model; cont_covar parameter specifies all the continuous predictors used in the model; cat_covar parameter specifies all the categorical predictors used in the model. For the last two parameters, if multiple variables are to be used, specify them in the order you want each variable to be presented in the final regression table, and separate each variable with a blank. With the current setup, all the continuous predictors are listed first in the final table, followed by all the categorical variables.

```
*===========Macro for Making Regression Tables in PROC GLM===============*;
%macro glm_table(dataset=, y=, cont_covar=, cat_covar=);

/*count the number of categorical covariates in the model and store
  this information in macro count*/
    %local count;
    %let count=0;
    %do %while (%scan(&cat_covar, &count+1) ne %str());
```

```
          %let count=%eval(&count+1);
    %end;

/*run regression model*/
    proc glm data=&dataset order=formatted;
          class &cat_covar;
          model &y = &cont_covar &cat_covar / solution CLPARM;

/*use do loop to generate a list of variable followed by its format*/
          format
                %local i;
                %do i=1 %to &count;
                      %scan(&cat_covar, &i) %scan(&cat_covar, &i).
                %end;
          ;
          ods output ParameterEstimates=Beta;
    run;

/*construct regression table*/
    data Beta1(drop=LowerCL UpperCL);
          set Beta(firstobs=2 keep=Parameter Estimate Probt--UpperCL);

          if LowerCL ne . then
CI="("||strip(put(LowerCL,7.2))||","||" "||strip(put(UpperCL,7.2))||")";
          else CI="-Reference-";

/*separate variable name from its label*/
          pos=index(Parameter,"");
          covariate=substr(Parameter,1,pos-1);
          level=strip(substr(Parameter,pos));

/*re-order reference group*/
          last_covariate=lag(covariate);
          if covariate ne last_covariate then num+1;
          level = tranwrd(level,"9","0");
    run;

/*blank line separating each variable*/
    proc sql;
          create table blank as
                select covariate, num
                      from Beta1
                            where probt=.;
    quit;

    data combine(keep=Estimate Probt CI covariate level num);
          set Beta1 blank;
    run;

    proc sort data=combine; by num level; run;

    data combine1;
          set combine;
          /*get rid of number prefix in format*/
          x=substr(level,3);
          level=x;
```

```
            /*label*/
            label level="Covariates" estimate="Beta" CI="95% CI"
                  probt="p-Value";
      run;

      proc format;
         value estimate low-<0 = [7.2] 0 = " " 0<-high = [7.2] other = " ";
         value p 0-<0.001 = "<.001" 0.001-1 = [5.3] other = " ";
      run;

   /*PRINT the final table!*/
      proc print data=combine1 label;
            id covariate;
            var level estimate CI probt;
            format estimate estimate. probt p.;
      run;
   %mend;
```

For example, to create the regression table shown in Figure 7, submit the following code:

```
   %glm_table(dataset=sashelp.bweight,
                y=weight,
                cont_covar=MomAge CigsPerDay MomWtGain,
                cat_covar=Black Married Boy MomSmoke Visit MomEdLevel)
```

One thing to note is that at the very beginning of this macro, instead of jumping right into PROC GLM, we first spend some time trying to count the total number of categorical variables to be used in the model. The reason for doing this is to avoid manually specifying all the variables and their formats in the FORMAT statement within PROC GLM section. As is shown above, the trick is to use macro function %SCAN to extract each word contained in the macro variable cat_covar. A local macro named count is created to accumulate the total number of words scanned in cat_covar, and the accumulation process continues until the %SCAN function detects no more words. Then in the FORMAT statement in PROC GLM, we use %DO loop to iteratively generate a series of text, each of which contains one categorical variable followed by its format name. The local macro count helps determine the number of iteration to perform. Earlier we mentioned that it is important to make sure the format name is the same as the variable name. Now you see why this is the case. Without such setup, it will be difficult to automate the format-assigning process within the FORMAT statement.

LOGISTIC REGRESSION EXAMPLE

We now apply what we have discussed before to a logistic regression example. Below is the macro for creating a regression table based on results from PROC LOGISTIC. The setup is very similar to that of PROC GLM. There are four parameters to be specified in the macro definition: data set, dependent variable, continuous predictors and categorical predictors. The macro starts by counting the number of categorical variables to be included in the model, then uses PROC LOGISTIC to generate the initial output, later employs a series of DATA step programming to format the ODS output tables before finally printing it out.

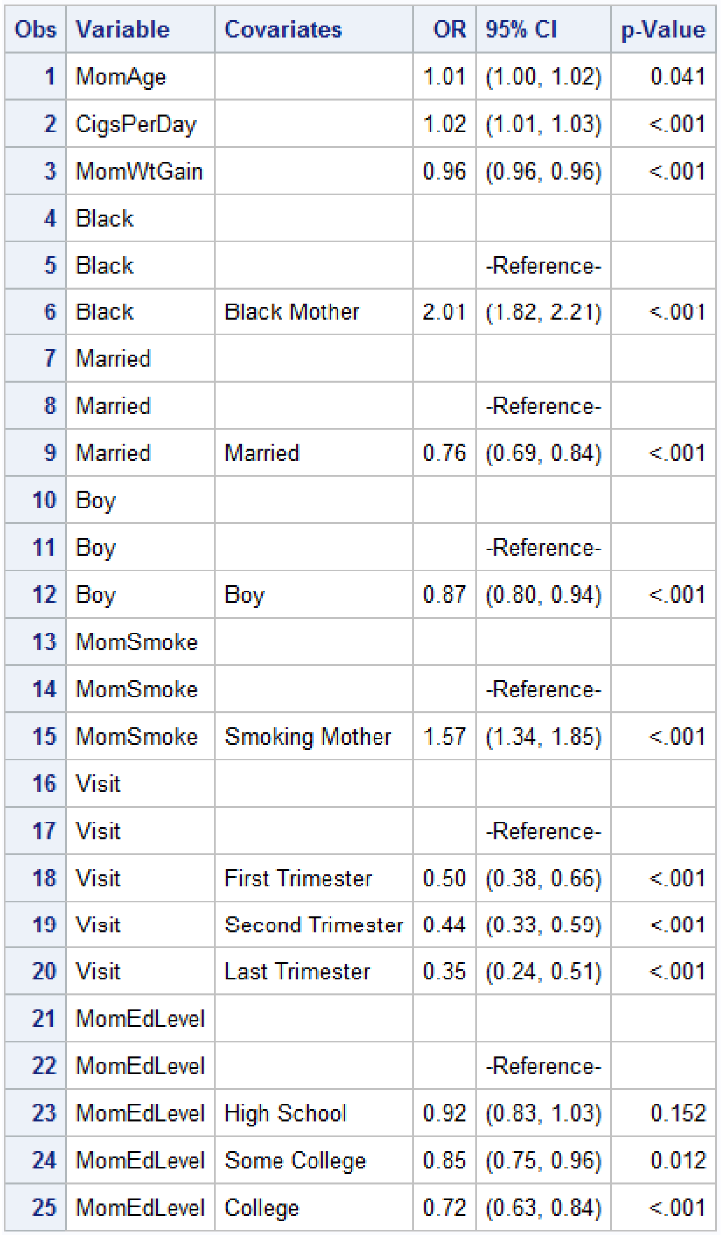
One thing to note is that for PROC LOGISTIC, information on estimated odds ratio and the corresponding p-value for each predictor is stored in two separate ODS tables (called OddsRatios and ParameterEstimates). You need to output both tables and later combine them together as shown below. Another potential issue is that PROC LOGISTIC suppresses the row for reference category in the output table. You might need to create it on your own in the later DATA step.

```
   *=========Macro for Making Regression Tables in PROC LOGISTIC============*;
   %macro logistic_table(dataset=, y=, cont_covar=, cat_covar=);
      *count the number of categorical covariates in the model;
```

```
    %local count;
    %let count=0;
    %do %while (%scan(&cat_covar, &count+1) ne %str());
          %let count=%eval(&count+1);
    %end;

    *regression model;
    proc logistic data=&dataset order=formatted;
          class &cat_covar / param=ref;
          model &y(event="1") = &cont_covar &cat_covar;
          format
                %local i;
                %do i=1 %to &count;
                      %scan(&cat_covar, &i) %scan(&cat_covar, &i).
                %end;
          ;
          ods output OddsRatios=OR ParameterEstimates=Beta;
    run;

    *construct regression table;
    data combined;
          set Beta(firstobs=2 keep=Variable ClassVal0 ProbChiSq);
          set OR(keep=OddsRatioEst LowerCL UpperCL);
CI="("||strip(put(LowerCL,5.2))||","||" "||strip(put(UpperCL,5.2))||")";

          last_var=lag(variable);
          if variable ne last_var then num+1;
    run;

    *select all the categorical variables;
    proc sql;
          create table ref as
          select distinct variable, num from combined
                      where ClassVal0 ne "";
    quit;

    *add two rows, one for reference group, one for the blank;
    data combined2; set combined ref ref; run;

    proc sort data=combined2; by num ClassVal0; run;

    data combined3;
          set combined2;
          by num ClassVal0;
          if first.num=0 and ClassVal0="" then CI="-Reference-";

          *get rid of the number in the format;
          x=substr(ClassVal0,3);
          ClassVal0=x;

          label ClassVal0="Covariates" OddsRatioEst="OR" CI="95% CI"
                ProbChiSq="p-Value";
    run;

    proc format;
          value OR low-high = [5.2] other = " ";
          value p 0-<0.001 = "<.001" 0.001-1 = [5.3] other = " " ;
    run;
    proc print data=combined3 label;
          var variable ClassVal0 OddsRatioEst CI ProbChiSq;
          format OddsRatioEst OR. ProbChiSq p.;
    run;
%mend;
```

Assuming that you want to run a logistic regression model investigating the factors associated with low birth weight of baby. The code below is used to generate a final table. Figure 8 displays the final output. The only thing for you to do manually is adding the variable names and the label for each reference group.

```
%logistic_table(dataset=bw, y=low,
                cont_covar=MomAge CigsPerDay MomWtGain,
                cat_covar=Black Married Boy MomSmoke Visit MomEdLevel)
```



Obs	Variable	Covariates	OR	95% CI	p-Value
1	MomAge		1.01	(1.00, 1.02)	0.041
2	CigsPerDay		1.02	(1.01, 1.03)	<.001
3	MomWtGain		0.96	(0.96, 0.96)	<.001
4	Black				
5	Black			-Reference-	
6	Black	Black Mother	2.01	(1.82, 2.21)	<.001
7	Married				
8	Married			-Reference-	
9	Married	Married	0.76	(0.69, 0.84)	<.001
10	Boy				
11	Boy			-Reference-	
12	Boy	Boy	0.87	(0.80, 0.94)	<.001
13	MomSmoke				
14	MomSmoke			-Reference-	
15	MomSmoke	Smoking Mother	1.57	(1.34, 1.85)	<.001
16	Visit				
17	Visit			-Reference-	
18	Visit	First Trimester	0.50	(0.38, 0.66)	<.001
19	Visit	Second Trimester	0.44	(0.33, 0.59)	<.001
20	Visit	Last Trimester	0.35	(0.24, 0.51)	<.001
21	MomEdLevel				
22	MomEdLevel			-Reference-	
23	MomEdLevel	High School	0.92	(0.83, 1.03)	0.152
24	MomEdLevel	Some College	0.85	(0.75, 0.96)	0.012
25	MomEdLevel	College	0.72	(0.63, 0.84)	<.001

Figure 8. Logistic Regression Table

CONCLUSION

This paper introduces several SAS Base programming techniques that can be used to automate the process of generating high quality regression tables for presentation purposes. Through examples on both linear and logistic regression, we demonstrate issues to be considered along the way and the corresponding solutions. These include: using format to easily manipulate the reference group and displaying orders for categorical variables; using ODS to extract tables that contain key information on regression results; using a series of DATA step functions and statements to transform and format tables and numbers. To further help automate the entire process, a macro program is further provided.

Comparing two examples used in this paper, we see that the general steps required to build a regression table are similar regardless of the regression procedure we are using. With slight adjustment, all the aforementioned techniques can be easily applied to many other commonly used regression settings, such as PROC GENMOD, PROC MIXED, PROC GLIMMIX, PROC PHREG and PROC LIFEREG.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ji Qi
University of Michigan Health System, Ann Arbor
qiji@umich.edu