BIOS 511 Lab 3
Basic Use of the DATA Step

Please read the following instructions carefully before beginning this lab. You will need to start by downloading the DM_INVNAM datasets from the LAB-03 assignment on the Sakai site. This dataset will be the basis of all activites for this lab. This dataset is nearly identical to the DM dataset used for LAB-02 but has two new variables added: SITEID (site id) and INVNAM (site investigator name).

Instructions:

- All tasks should be completed in a single SAS program named lab-03-PID.sas where PID is your student PID number. Please make sure to include an appropriate header in your SAS program.
- All the output that your SAS program produces in this lab should be delivered to a *single HTML file* named lab-03-PID-output.HTML.
- In your code, before starting a new task, include a block comment with the task number

```
/*************************************************************
                        SAS Code for Task # X
 *************************************************************/
```

- The output from the 3 tasks should be put on separate pages.
- All the output from a single task should be put on the same page.
    - Recommendation: While initially writing the SAS program, do not concern yourself with creating a permanent output file. Simply view results in the results window to verity that your program has created the desired output. Once your program is essentially complete, add the appropriate ODS statements to create the permanent HTML file based on the requirements above.
- You will upload the SAS program, SAS log, and HTML output file to document completion of the lab.

BIOS 511 Lab 3
Basic Use of the DATA Step

Task 1: For this task, using the ECHO trial DM dataset as input, you will create a dataset named WORK.DM_TASK1 that contains two additional new variables: AGECAT and AGECATN. These variables correspond to character and numeric age categories. After checking that these variables are created correctly, create a two-way frequency table and perform a Chi-square test. Use the following steps to complete the task:

Step 1: Create the data set WORK.DM_TASK1.

- Use a LENGTH statement to give the character AGECAT variable a length of 5.

  For more guidance, you can google "SAS 9.4 Documentation LENGTH statement" and the top link should provide detailed information on the LENGTH statement.

- The value of AGECAT should be missing if AGE is missing, equal to "<65" if AGE is non-missing and less than 65, and equal to ">=65" if AGE is at least 65.

  In order to create the AGECAT variable, use must ***conditional assignment statements*** (there are other ways but the ability to use conditional assignment statements is a critical skill).

  Template code for this exercise is given below:

```
if      condition1 then ageCat = "<65";
else if condition2 then assignment statement;
```

  You must write valid SAS code to replace *condition1*, *condition2,* and *assignment statement*.

  Note that a character variable will be missing unless it is assigned a non-missing value by an assignment statement.

  For reference, example assignment statements are given below:

  [1] X = "Y";
  [2] Z = 10 + W;

  Example [1] assigns the value Y to the variable X which is assumed to be character (how do you know?).

  Example [2] assigns the value 10 + <Value of W> to the variable Z. Both W and Z are assumed to be numeric (how do you know?).

- The value of the AGECATN variable should be equal to 99 if AGE is missing, equal to 1 if AGE is non-missing and less than 65, and equal to 2 if AGE is at least 65. Thus, the AGECATN and AGECAT variables should obey the following mapping:

AGE		AGECATN	AGECAT
Lowest Value	Highest Value		
.		99	
0 (min)	<65	1	<65
65	∞ (max)	2	>=65

- The variable AGECAT could be created using a similar approach to what is given above but for this set of conditional assignment statements you must use ***Boolean Numeric Expressions***.

- Template code for this exercise is given below:

```
if condition1 then ageCatn = 99;
else               ageCatn = 1 + ( boolean expression );
```

You must write valid SAS code to replace *condition1* and *boolean expression*.

Note that if the Boolean expression is true, a value of 1 will be added. Otherwise, a value of 0 will be added.

For example, consider the character variable SEX which has values M and F. One could create a variable SEXN which is equal to 1 when SEX is equal to F and 0 when SEX is equal to M using the following assignment statement:

SEXN = (SEX = "F");

Note that the above assignment statement is a complete and valid programming statement. If the value of SEX was missing (i.e " "), SEXN would be assigned a value of 0 based on the simple assignment statement in the example. That is why condition1 is included in the conditional assignment statement that you must complete for this task.

- Now, to create the WORK.DM_TASK1 dataset, you need to put all the instructions given above together. A template for the complete DATA step is provide below:

```
data DM_TASK1;
 set LIBREF.DM;
 /*** Insert length statement for AGECAT ***/

 /*** Insert programming statement(s) to create AGECAT ***/

 /*** Insert programming statement(s) to create AGECATN ***/
run;
```

Each of the three comments should be replaced with SAS code that does what the comment describes.

By convention, LENGTH statements which define the lengths of newly created variables are typically placed at the top of the data step just below the SET statement. If multiple new character variables are created in the DATA step, it is common to define their lengths with a single LENGTH statement.

It is uncommon to define the length of a numeric variable using a LENGTH statement. The only reason to define a numeric variable length to be anything other than 8 bytes is decrease the size (in terms of disk space) for what would be an otherwise massive dataset. Unless one is truly working with "BIG DATA", do not define the length of a numeric variable using a LENGTH statement; simply use the default length of 8 bytes.

Step 2: When you create new variables you will generally want to quickly verify that the created variables are defined appropriately. Check your derivation by running the following PROC FREQ step

```
proc freq data = DM_Task1;
  table age*agecatn*ageCat / list missing nocum nopercent;
run;
```

Before now we have not used the LIST option for the TABLE statement. That option is useful when one wants to view a two-way table in the format of a one-way table. Once you have verified that your derivation is correct with this PROC FREQ step, ***comment out*** the PROC FREQ step by placing it within a block-style comment (i.e., /* ….. */). It is common to write simple PROC steps like this to check derivations and comment them out or remove them from the program once you have ***debugged*** the code. There

should be no printed output from this PROC FREQ step in the lab solution you turn in, but the code (within a block-style comment) should be a part of the submitted SAS program.

Step 3: A common analysis performed in clinical trials uses a Chi-Square test to assess whether or not there are any imbalances between a categorical variable (i.e., AGECATN) between the two treatment groups. Write a PROC FREQ step that requests a two-way frequency analysis of the AGECAT variable by the ARMCD variable and use the CHISQ option on the TABLE statement to request the Chi-Square test for independence of the two variables (i.e., that the age distribution is the same for both treatment groups).

Use appropriate options to suppress cell and row percentages in the contingency table.

By using an ODS SELECT statement, only include the two-way frequency analysis and the ODS output object that contains the chi-square test table in your submitted PDF file. The title of your output should be: "Task 1 / Step 4: Two-Way Frequency Analysis of Treatment Group by Age Category".

Use a LABEL statement to add temporary labels to the AGECAT and ARMCD variables (make sure to use the ARMCD variable for this analysis instead of ARM). The labels should read "Age Category" and "Treatment Group", respectively.

Write a PROC FORMAT step to create a character format that maps the value "ECHOMAX" to "Intervention" and the value "PLACEBO" to "Placebo". Attach the format to the ARMCD variable in the PROC FREQ step so that the formatted values are displayed in the ODS table.

Step 4: Before moving on, ***temporarily*** comment out the LENGTH statement from your DATA step and run the code again. Examine the values of AGECAT. If you followed the template code provided, you will see that some of the values of AGECAT are now truncated.

This underscores the importance of using LENGTH statements and NOT letting SAS determine the length of newly created character variables. If you do not use a LENGTH statement, SAS will define the length of a newly created character variable as the length required to store data based on the *first* assignment statement where the variable is used in the DATA step (when viewing from top to bottom). If the template code provided was used for programming, this would be a length of 3 bytes which would be insufficient to store a value ">=65" without truncation (i.e., losing the "5").

Task 2: For this task, you will create a data listing of the primary investigators at each of the research sites in the clinical trial for each country where enrollment took place.

Step 1: First, use a PROC SORT step to create a *temporary* dataset named INVESTIGATORS1 that contains only the three variables COUNTRY, SITEID, and INVNAM.

This dataset should ordered by SITEID and INVNAM (there is one INVNAM per site), and there should be a single observation for each SITEID.

This can be done with a single PROC SORT step using the KEEP= dataset option on the newly created dataset to ensure only the desired variables are kept and using one option of the PROC SORT statement that requests duplicate observations be removed (see online documentation or previous lecture materials for assistance).

BIOS 511 Lab 3
Basic Use of the DATA Step

Template code is provided below:

```
proc sort data = LIBREF.DM
     out = Investigators1(keep=<list of variables to Keep>) <option>;
 by <list of variables to order by>;
run;
```

Thus far in the course we have used several dataset options. These include FIRSTOBS=, OBS=, and now the KEEP= option. One can also use a DROP= option to remove unwanted variables. Remember that dataset options are included in parentheses that are adjacent to the data set reference to which the options apply.

Step 2: Write a DATA step that reads from the newly created INVESTIGATORS1 dataset to create a new temporary data set named INVESTIGATORS2 that contains four new variables: (FIRSTNAME, LASTNAME, COUNTRY_ORDER, and COUNTRY_LONG).

- The FIRSTNAME and LASTNAME variables should have length 30 and the COUNTRY_LONG variable should have length 10. All variable lengths should be set with a single LENGTH statement included just below the SET statement by convention. You need not include the COUNTRY_ORDER variable in the LENGTH statement, as it is numeric.

- The values of the INVAM variable are in all caps and of the form "Last Name, First Name". You must use ***nested*** SAS DATA step function calls to extract the first/last name and change its case.

  - The SCAN function can be used to extract the first or last name since those values are consistently ***separated*** by a comma. The SCAN function is useful when you want to extract words or phrases that are separated by a common ***delimiter***. A delimiter is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text (i.e., words).

  - The PROPCASE function can convert from caps to title case (i.e. MATT → Matt).

  - The STRIP function can remove leading or trailing whitespace from any character variable.

  - A template assignment statement is provided for the LASTNAME variable.

    ```
    lastName = propcase(scan(INVNAM ,1,','));
    ```

    Write your own code for the FIRSTNAME variable. Note that the FIRSTNAME variable needs to use the STRIP function in addition to PROPCASE and SCAN. Can you see why?

    See the appendix at the end of this lab for information on how you can access the online SAS documentation for DATA step functions to help you understand their required and optional arguments.

  - Using techniques similar to the previous task and using the COUNTRY variable as the basis for the derivation, create the numeric COUNTRY_ORDER variable so that it is equal to 1 where COUNTRY is equal to *USA*, equal to 2 when COUNTRY is equal to *MEX*, and equal to 3 when COUNTRY is equal to *CAN*.

  - Create the COUNTRY_LONG variable so that it has values "USA", "Mexico", and "Canada". This variable will be used for display in the data listing.

- Both the COUNTRY_ORDER and COUNTRY_LONG variables can be created with one set of conditional assignment statements using a DO block. The basic syntax for a DO block is as follows:

```
if condition1 then do;
      assignment-statement-1;
      assignment-statement-2;
      .
      .
      .
      Assignment-statement-K;
end;
```

  In contrast, the following conditional assignment statement is syntactically correct but will produce incorrect results. Can you see why?

```
if condition1 then
      assignment-statement-1;
      assignment-statement-2;
      .
      .
      .
      Assignment-statement-K;
```

  This is a critical point and make sure to ask a question if you don't understand!

- Write a LABEL statement to attached permanent labels to the FIRSTNAME, LASTNAME, and COUNTRY_LONG variables. The labels should be "Investigator First Name", "Investigator Last Name", and "Country Name", respectively.

  The LABEL statement should be included in the DATA step so that the labels are permanently attached to the newly created variables. By convention the label statement will also go towards the top of the DATA step, often just below LENGTH and FORMAT statements but the LABEL and FORMAT statements can technically be included anywhere in the DATA step whereas a LENGTH statement must be used prior to any other reference of the variable in an assignment statement.

- An overall template for the INVESTIGATORS2 dataset is given below:

```
data Investigators2;
 set Investigators1;

 ** LENGTH statement for FIRSTNAME, LASTNAME, and COUNTRY_LONG;

 ** LABEL statement for FIRSTNAME, LASTNAME, and COUNTRY_LONG;

 lastName  = strip(propcase(scan(INVNAM ,1,',')));
 *** Assignment statement for FIRSTNAME variable;

 *** Conditional assignment statement DO block for COUNTRY_ORDER/COUNTRY_LONG;
      if COUNTRY = 'USA'  then do; COUNTRY_ORDER = 1; COUNTRY_LONG = COUNTRY;  end;
 else if condition2  then do; ..................................................... end;
 else if condition3  then do; ..................................................... end;

run;
```

Step 3: Once the INVESTIGATORS2 dataset has been created. Write a PROC SORT step to sort the data *in place* by the COUNTRY_ORDER variable *(e.g.; without an OUT= option on the PROC SORT statement).* See below for the code:

```
proc sort data = Investigators2;
 by COUNTRY_ORDER COUNTRY_LONG;
run;
```

This code will simply overwrite the existing temporary dataset replacing it with one that has been ordered based on the newly created variable. Does it matter whether include the COUNTRY_LONG variable in the BY statement for the PROC SORT step. Convince yourself that including that variable or not as the second variable in the BY statement (when COUNTRY_ORDER is the first) makes absolutely no difference in the output dataset. When would it make a difference?

Step 4: Write a PROC PRINT step that displays the data by country.

- Use a BY statement in the PROC PRINT step so that separate data listings are printed for the investigators from each country. The BY statement indicates to PROC PRINT which variables the data are sorted by, it does not sort the data. Thus, if one used the BY statement:

  ```
  by COUNTRY_LONG;
  ```

  the PROC PRINT step would produce an ERROR when executed and no output would be produced. Do you understand why?

- Only print site ID, investigator last name, and investigator first name.

- Use PROC statement options to suppress observation numbering and to ensure labels are printed.

- Use the following title statements:

  - Template code is provided below:

```
title1 "Task 2 / Step 4: Listing of ECHO Trial Investigators";
title2 "Country = #byval(COUNTRY_LONG)";
proc print data = Investigators2 <options>;
  by <variables that define the sort order of the data>;
  var <variables to print in display>;
run;
```

Note that the use of the #BYVAL() technique here is very powerful as it allows users to substitute values of BY statement variables into the title for different groups of observations. By default, the values of BY variables are printed as a less visually appealing title. This can be suppressed by using the NOBYLINE option on a global OPTIONS statement.

Task 3: Complete all of the following. For this task, create a data set named DM_TASK3 from the DM_INVNAM data set.

Step 1: Write a DATA Step from scratch (e.g., no template) that creates the following variables.

- Use the INDEX function to create a variable named COMMA_SPOT that stores the location of the comma in the values of INVNAM. Then, use the value of COMMA_SPOT along with the SUBSTR and PROPCASE functions

to create FIRSTNAME and LASTNAME as was done for task 2. Give the new variables lengths of 30. Compare the results here to those from the previous task to make sure you have done this correctly.

- Use the SCAN function to create character ICDAY, ICMONTH, and ICYEAR variables that are equal to the DAY, MONTH, and YEAR of informed consent (stored in the RFICDTC variable). If the value of RFICDTC is 2016-01-20, then ICDAY should be 20, ICMONTH should be 01, and ICYEAR should be 2016. *Make sure to set the LENGTH of each variable to 5 (even though it could be smaller)*.

- Create three new variables RFICDTC3- RFICDTC5 (that will be identical to RFICDTC), using ICDAY, ICMONTH, and ICYEAR as input and by concatenating their values using three different techniques: (1) by using the double-pipe concatenation operator (i.e., ||), (2) by using the CATS function, and (3) by using the CATX function. Give the new variables lengths of 20.

- Create the RACECAT variable from the RACE variable such that the RACECAT variable has values “White”, “Black or African American”, and “Other”. In the USA, such categorization are common due to small numbers of enrollees for other races. Note that the values of RACECAT should no longer be in all-caps like the race variable. Give the RACECAT variable a length of 30. The RACECAT variable can be created using many different techniques.

    - One can write conditional assignment statements that explicitly assign the values above (e.g., if RACE = “WHITE” then RACECAT = “White”).

    - Alternatively, one can use a combination of the PROPOCASE and TRANWRD functions. The TRANWRD function can be used to correct how the PROPCASE function deals with the word “OR”. Try and derive the RACECAT variable both ways for practice. It does not matter which approach you use for your solution. Comment the other approach out using a block-style comment but leave it in your program.

For this DATA step, write a single LENGTH statement and drop the COMMA_SPOT variable from the DM_TASK3 dataset using a DROP statement. The DROP statement can go anywhere in a DATA step by is generally placed at the end other the DATA step just before the RUN statement by convention.

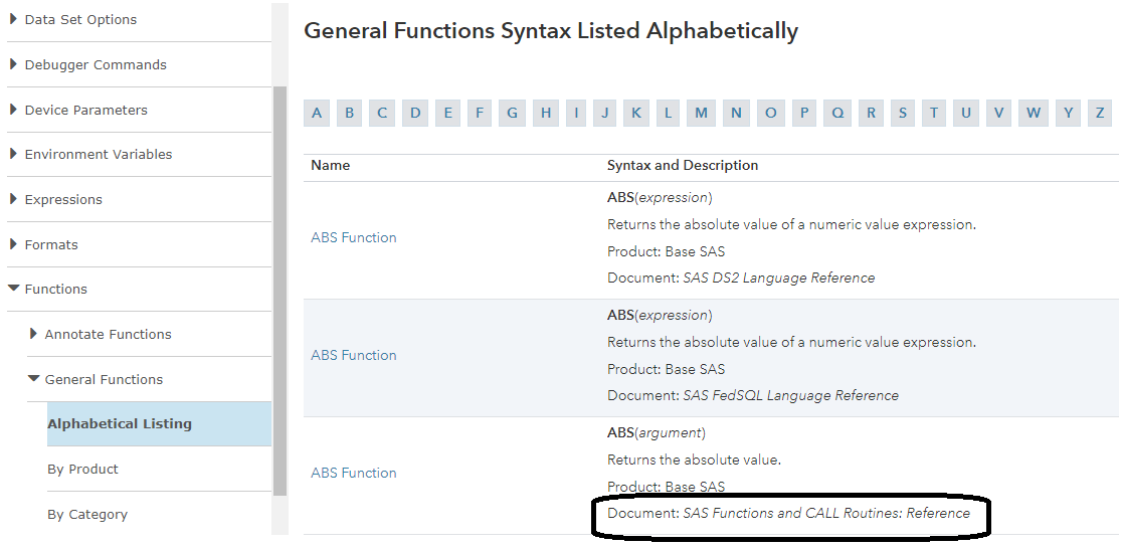
Step 2: Write a PROC PRINT step to that displays the newly created variables as well as the INVNAM, RFICDTC, and RACE variables to check your derivations.

- Suppress the default observation numbering
- Restrict the printed data to site 001 by using a where a WHERE statement.
- Print only the first 10 observations for that site using the OBS= dataset option on the PROC print step.

BIOS 511 Lab 3
Basic Use of the DATA Step

Appendix: Finding the documentation on SAS DATA step functions:

1. Google “SAS 9.4 Product Documentation” and select the product documentation web link.
2. Under “Syntax Shortcuts” click the hyperlink named “SAS Language Elements by Name, Product, and Category”.
3. On the panel that should now display on the left of your screen, click on “Functions” and then “General Functions”.
4. If you know the name of the function you want to look up, select “Alphabetical Listing” and then search to find it. Most functions have multiple entries and you will generally want the one corresponding to “Document: SAS Functions and CALL Routines: Reference” (see below).



5. Click on the name of the function to learn more about how to use the function in programming statements.