BIOS 511 Lab 6
Advanced Use of the DATA Step – Transposing Data

Please read the following instructions carefully before beginning this lab. You will need to start by downloading the VS dataset from the Lab-06 assignment on the Sakai site. This dataset will be the basis of all activites for this lab.

Instructions:

- All tasks should be completed in a single SAS program named lab-06-PID.sas where PID is your student PID number. Please make sure to include an appropriate header in your SAS program.

- For this lab you will not turn in an output file. You will turn in your program, log, and two SAS datasets which MUST be named BP4 and BP5 per the instructions. Failure to follow these instructions for naming the datasets will result in no credit being given for the related tasks.

- In your code, before starting a new task, include a block comment with the task number:

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

BIOS 511 Lab 6
Advanced Use of the DATA Step – Transposing Data

VS – Vital Sign data for the ECHO clinical trial (# obs = 15050)

In this lab you will work with the vital signs dataset (VS). The VS dataset contains data on vital signs that were collected during in the ECHO clinical trial. The variables contained in the VS dataset along with their types, lengths, and labels are included in the table below. A description of the values stored in the variable is also provided.

Variables contained in the VS dataset					
#	**Variable**	**Type**	**Len**	**Label**	**Description**
1	STUDYID	Char	10	Study Identifier	Values are always equal to "ECHO"
2	USUBJID	Char	30	Unique Subject Identifier	Values are of the form "ECHO-XXX-YYY" where XXX and YYY are integers
3	VSSEQ	Num	8	Sequence Number	A unique identifier for observations within a subject. The values of VSSEQ should start at 1 for each subject and should increment by 1 for each successive observation for the subject.
4	VSTESTCD	Char	8	Vital Signs Test Short Name	Short code for vital sign name: DIABP, HEIGHT, HR, SYSBP, WEIGHT
5	VSTEST	Char	40	Vital Signs Test Name	Vital sign name: Diastolic Blood Pressure, Height, Heart Rate, Systolic Blood Pressure, Weight
6	VSSTRESN	Num	8	Numeric Result/Finding in Standard Units	Numeric vital sign test result
7	VSSTRESU	Char	20	Standard Units	Vital sign unit
8	VSSTAT	Char	10	Completion Status	Values are NOT DONE or missing. This variable indicates why a vital sign was not done.
9	VSREASND	Char	50	Reason Test Not Done	Reason vital sign was not done
10	VSBLFL	Char	1	Baseline Flag	For each subject and vital sign, set to "Y" for the observation associated with the latest non-missing test result among those that occur on or before the date of first dose. Otherwise, VSBLFL will be missing.
11	VISITNUM	Num	8	Visit Number	Values: -1, 1, 2, 3, 4, 5
12	VISIT	Char	50	Visit Name	Values: Screening, Week 0, Week 8, Week 16, Week 24, Week 32
13	VSDTC	Char	20	Date/Time of Specimen Collection	Vital sign test date in YYYY-MM-DD format

The structure of the VS dataset is one observation per subject per vital sign and per visit.

BIOS 511 Lab 6
Advanced Use of the DATA Step – Transposing Data

Task 1: You do not have to turn in anything for this task other than your program and log. For this task, the goal is to simply visualize the VS data to gain an understanding of the data structure. The VS dataset is in what is referred to as "Long" format. This typically means that each subject has multiple observations in the dataset.

- Write a PROC PRINT step that prints the data for the subject with unique ID equal to ECHO-011-001.

Task 2: You do not have to turn in anything for this task other than your program and log. For this task you will create a dataset that contains one observation for each subject at each visit. The dataset will contain the variables USUBJID, VISITNUM, VISIT, DIABP, and SYSBP. The USUBJID, VISITNUM, and VISIT variables exist in the VS dataset. The DIABP and SYSBP variables will need to be created. You will explore three different ways of doing this to practice SAS programming skills. A depiction of what you are attempting to achieve is shown below:

Before

USUBJID	VISITNUM	VISIT	VSTESTCD	VSSTRESN
ECHO-011-001	-1	Screening	DIABP	112
ECHO-011-001	-1	Screening	SYSBP	139
ECHO-011-001	1	Week 0	DIABP	112
ECHO-011-001	1	Week 0	SYSBP	141
ECHO-011-001	2	Week 8	DIABP	109
ECHO-011-001	2	Week 8	SYSBP	140
ECHO-011-001	3	Week 16	DIABP	111
ECHO-011-001	3	Week 16	SYSBP	137
ECHO-011-001	4	Week 24	DIABP	109
ECHO-011-001	4	Week 24	SYSBP	134
ECHO-011-001	5	Week 32	DIABP	106
ECHO-011-001	5	Week 32	SYSBP	134

After

USUBJID	VISITNUM	VISIT	DIABP	SYSBP
ECHO-011-001	-1	Screening	112	139
ECHO-011-001	1	Week 0	112	141
ECHO-011-001	2	Week 8	109	140
ECHO-011-001	3	Week 16	111	137
ECHO-011-001	4	Week 24	109	134
ECHO-011-001	5	Week 32	106	134

Only a few variables are shown in the "Before" dataset but several more exist.

Only the data for one subject is shown.

BIOS 511 Lab 6
Advanced Use of the DATA Step – Transposing Data

Approach 1: Creating multiple datasets and merging them.

[1] For this approach, you will create two datasets, named WORK.DIABP and work.SYSBP, which contain the observations corresponding to diastolic blood pressure and systolic blood pressure measurements, respectively.

[2] You will then sort both the datasets by the variables necessary for the merge (in fact you can sort the ECHO.VS dataset as a first step as an alternative approach).

[2] Once you have created the two datasets, you will write a DATA step to merge them together using USUBJID and VISITNUM/VISIT as the merge BY variables. The data set should be named WORK.BP1.

A template SAS program is provided below.

```
data WORK.DIABP;
 set echo.vs;
 where <INSERT CONDITION TO SELECT DIABP RECORDS>;
  rename <RENAME RESULT VARIABLE TO DIABP>;
run;

proc sort data = WORK.DBP; by <VARIABLES TO MERGE BY>; run;

data WORK.SYSBP;
 set echo.vs;
 where <INSERT CONDITION TO SELECT SYSBP RECORDS>;
  rename <RENAME RESULT VARIABLE TO SYSBP>;
run;

proc sort data = WORK.SYSBP; by <VARIABLES TO MERGE BY>; run;

data WORK.BP1;
 merge <DATA SETS TO MERGE>;
  by <VARIABLES TO DEFINE RECORD MATCHING>;
  keep usubjid visitnum visit sysbp diabp;
run;
```

BIOS 511 Lab 6
Advanced Use of the DATA Step – Transposing Data

Approach 2: Using arrays and conditional output statements/subsetting IF statements.

[1] Write a PROC SORT step that orders the vital sign data by subject, visit, and then vital sign. Use a WHERE statement to select only the blood pressure vital signs. Name the newly created data set as WORK.VS.

[2] Write a DATA step that uses the newly created WORK.VS data set as input that creates a dataset named WORK.BP2.

- This DATA step my use a BY statement so that the temporary FIRST.XXXX and LAST.XXXX are initialized in the program data vector (PDV).
- Since the data for the systolic and diastolic blood pressure measurements are stored on two separate observations, a RETAIN statement is needed so that the vital sign value of the first observation can be pulled onto the second observation.
- To prevent data from one visit from being pulled onto observations for the next visit, use a conditional DO block to set both the SYSBP and DIABP variables to missing for the first observation processed for a given visit.
- In order for the work.BP2 dataset to have only a single observation for each subject and visit, use a subsetting IF statement to select only the last observation for each subject and visit.

A template SAS program is provided below.

```
proc sort data = echo.vs out = work.VS;
 by usubjid visitnum visit vstestcd;
 <INSERT WHERE STATEMENT>;
run;

data BP2;
 set work.VS;
 by <LIST OF VARIABLES BY WHICH DATA IS SORTED>;

 retain sysbp diabp;

 if <CONDITION TRUE FOR FIRST OBSERVATION PER SUBJECT AND VISIT> then do;
  sysbp = .;
  diabp = .;
 end;

 if vstestcd = 'SYSBP' then sysbp = <VARIABLE FROM ECHO.VS>;
 if vstestcd = 'DIABP' then diabp = <VARIABLE FROM ECHO.VS>;

 if <CONDITION TRUE FOR LAST OBSERVATION PER SUBJECT AND VISIT>;

 keep usubjid visitnum visit sysbp diabp;
run;
```

Hint: Use of FIRST.VARIABLE and LAST.VARIABLE can be challenging to new programmers. In order to use these *temporary* variables correctly, you need to be able to visualize how they are populated in the data step as observations are processed. You might use the following snippets of code to help understand these variables better. Your solution program can contain the below code but comment the code out using a block comment prior to final execution of your program.

One way of doing this is to use the PUTLOG statement to write out their values as in the following example:

```
option ls=150;
data _null_;
 set work.VS(obs=20);
 by usubjid visitnum visit vstestcd;
 putlog _n_= usubjid= first.usubjid= last.usubjid=
         visitnum= visit= first.visit= last.visit= vstestcd=  ;
run;
```

Note that the _NULL_ keyword on the DATA statement is an instruction for SAS to NOT create a new dataset (i.e., a dataset named WORK._NULL_ is NOT created). This is useful when one only wants to process a dataset but not create a new one. There are many instances where this is helpful and one is this case where we simply want to visualize the program data vector (PDV) which stores all values of the variables for the observation being processed. For each PUTLOG statement, the contents of the variables listed are printed to the SAS log.

Note: Setting the LS option adjusts the line size for the printed log in SDM so that the text written can extend longer across each line in the log. This is helpful when many variables are being printed to avoid wrapping the printed values over multiple lines (making the print out difficult to read).

Since FIRST.VARIABLE and LAST.VARIABLE are *temporary* they are not included in any dataset created by the DATA step. Thus, one cannot directly view the values of FIRST.VARIABLE and LAST.VARIABLE after the fact. A second approach to understanding how these variables are populated is to explicitly create new variables that are permanent so that we can then print those variables to view the data in a more easily readable way. See the following example:

```
data work.temp;
 set work.VS(obs=20);
 by usubjid visitnum visit vstestcd;
  F_usub = first.usubjid;
  L_usub = last.usubjid;

  F_vis = first.visit;
  L_vis = last.visit;
run;

title "Helpful display to understand FIRST.VARIABLE and LAST.VARIABLE";
proc print data = work.temp(obs=20);
 by usubjid;
 var usubjid F_usub L_usub visitnum visit F_vis L_vis vstestcd;
run;
```

BIOS 511 Lab 6
Advanced Use of the DATA Step – Transposing Data

Approach 3: Using PROC TRANSPOSE

The following code uses PROC TRANSPOSE to transform the data from long to wide format.

- The BY statement in PROC TRANSPOSE defines the unique observations that will exist in the newly created data set. The input data set must be sorted by those variables.
- The ID statement identifies a variable whose values will be used as the names of the newly created variables for the data once in "wide" format.
- The IDLABEL statement identifies the variable whose values define the labels for the newly created variables.
- The VAR statement defines the variable whose values are assigned to the newly created variables.

```
proc sort data = echo.vs out = VS;
 by usubjid visitnum visit vstestcd;
run;

proc transpose data = VS
      out = BP3(drop=<REMOVE VARIABLES CREATED DURING PROC TRANSPOSE>);
 by usubjid visitnum visit;
 where <CONDITION TO ONLY PROCESS BLOOD PRESSURE OBSERVATIONS>;
  id vstestcd;
  idlabel vstest;
  var vsstresn;
run;
```

Make the following modifications to the code above:

- By default, PROC TRANSPOSE adds several additional variables to the newly created data set. Use the DROP= data set option on the WORK.BP3 dataset to remove those variables from the WORK.BP3 dataset as it is created by PROC TRANSPOSE.
- Add a WHERE statement to PROC TRANSPOSE so that only the blood pressure observations are processed.

BIOS 511 Lab 6
Advanced Use of the DATA Step – Transposing Data

Task 3: For this task you will turn in your BP4 SAS dataset. Using Approach #2, create a dataset named BP4 that contains one record per subject and that stores all blood pressure measurements for the subject in separate variables on the observation. The variables storing the blood pressure measurements MUST be named as follows:

- Diastolic Blood Pressure: DBP_SCR, DBP_WK00, DBP_WK08, DBP_WK16, DBP_WK24, DBP_WK32
- Systolic Blood Pressure: SBP_SCR, SBP_WK00, SBP_WK08, SBP_WK16, SBP_WK24, SBP_WK32

Perhaps the most elegant way to complete this task using approach #2 is to use an ARRAY to store all the new variables and cleverly reference the variables in the array. A partially complete program is given below as a template to help get you going for this task.

```
data work.vs;
 set echo.vs;
 where vstestcd in ('DIABP' 'SYSBP');
run;
proc sort data = work.vs; by usubjid visitnum visit vstestcd; run;

data <LIBREF>.BP4;
 set work.vs;
 by usubjid visitnum visit vstestcd;

 retain DBP_SCR DBP_WK00 DBP_WK08 DBP_WK16 DBP_WK24 DBP_WK32
        SBP_SCR SBP_WK00 SBP_WK08 SBP_WK16 SBP_WK24 SBP_WK32;

 array bp[2,6] DBP_SCR DBP_WK00 DBP_WK08 DBP_WK16 DBP_WK24 DBP_WK32
               SBP_SCR SBP_WK00 SBP_WK08 SBP_WK16 SBP_WK24 SBP_WK32;

if first.usubjid then do;
 <*** WRITE CODE TO SET ALL BP ARRAY VARIABLES TO MISSING
  WHEN A NEW USUBJID VALUE IS ENCOUNTERED ***>
end;
      if vstestcd = 'DIABP' then array_row = 1;
 else if vstestcd = 'SYSBP' then array_row = 2;

 <WRITE CODE TO CREATE A NEW VARIABLE NAMED "ARRAY_COL" THAT IDENTIFIES THE
  CORRECT ARRAY COLUMN BASED ON VISIT/VISITNUM>

  bp[array_row,array_col] = vsstresn;

 <WRITE CODE TO KEEP ONLY ONE OBSERVATION PER SUBJECT AND
  MAKE SURE TO KEEP THE CORRECT OBSERVATION>

 keep usubjid visitnum visit dbp: sbp:;
run;
```

IMPORTANT NOTE: The code to set all BP array variables to missing when a new USUBJID is encountered should make use of the FIRST.USUBJID variable, as this should only happen the first time the USUBJID value is encountered.

To understand why, think about what would happen if one subject had an observation for diastolic blood pressure at week 24 (a value of 78) but the next subject did not have the week 24 observation. For the second subject, their week 24 value would be incorrectly inherited from the first subject (a value of 78). By setting all variables to missing when a new subject is encountered we ensure data are not pulled across subjects!

The placement of the code that sets all array variables to missing is also critical. The program would not function correctly if that code were beneath the line: `bp[array_row,array_col] = vsstresn;` ***? Can you see why? If not, ask for help!***

BIOS 511 Lab 6
Advanced Use of the DATA Step – Transposing Data

Task 4: For this task you will turn in your BP5 SAS dataset. Using Approach #3, create a dataset named BP5 using PROC TRANSPOSE. A partially complete program is given below as a template to help get you going for this task.

```
data <LIBREF>.BP5;
 set echo.VS;
 where vstestcd in ('DIABP' 'SYSBP');

 length varName $20.;

 <CREATE A VARIABLE NAMED "VARNAME" THAT HAS VALUES SUCH AS
  "DBP_SCR" OR "SBP_WK16">

run;
```

Once you have completed the DATA step, the VARNAME variable can be used in the ID statement for PROC TRANSPOSE (for this example you needn't worry about using the IDLABEL statement). The BP5 data sets MUST contain the same variables as the BP4 dataset (in fact they should be identical data sets).