CHANGE to Separate Out self-determined
PHPM 672/677 Assignment #3: Arrays & Loops
Due date: Submit in E-Campus by 11:59pm Monday 2/17
Mid point check due date: Email by 11:59pm Monday 2/10 (next week)

Submission. Submit on E-Campus by 11:59pm the day before the class they are due.
There are TWO OPTIONS: structured or self-defined. Pick one.

- Structured: I planned the code (in comments). You just code. **Recommended for most (i.e., pick this if you are not sure).**
- Self-defined: You plan the code, and write the code

1. Midpoint email (1 point)
   - See below for details
2. Final Submission
   - Commented code (SAS: lnameN.sas, where N indicates the assignment number and lname is your last name): a total of four sas code (*_v1.sas, *_v2.sas, *_v3.sas, and lname.sas)
   - Output from your code (SAS: lnameN.log & **lnameN.lst**): for all code code
     - IMPORTANT NOTE: you must turn in a text output file with extension *.lst. html files are not acceptable because they are difficult to compare the differences
     - If the three output files are not identical in the content, you will get points off.
   - Readme.txt file answers to questions
   - Lab 3: different commented program, log, lst as appropriate for the option you chose

Late Assignments. Each student will be allowed one late assignment, due 7 days from the due date. NO other late assignments or make up will be accepted.

Plagiarism: If you consult any outside sources when doing your work, you are expect to further document these sources. Give credit where credit is due. Plagiarism will not be tolerated.

All handed in homework should state at the top any assistance with debugging and programming, as well as citations of any program segments copied from a website.

Elegance: There is always more than one way to say something, but some ways will be more "elegant" than others. You will learn to recognize elegant expressions as you become more familiar with a language and use the elegant idioms as you become more skilled.

Elegance will count in this assignment! It does not mean complicated programs. Remember KISS. The beauty of great programs is its SIMPILICITY. What you are trying to learn is how to break down complex tasks into simple modular tasks.

Required & recommended readings for this assignment

1. https://stats.idre.ucla.edu/sas/modules/working-across-variables/
2. The little SAS book (online book available from the library): 3.11 Simplifying programs with arrays & 3.12 Using Shortcuts to Lists of Variable Names

Guideline for assignment grading (Total of 8)

- Assignment (Total 4)
  - 1: Submitted code that does not run.
  - 2: Mostly running but incorrect.
  - 3: Correct and meets requirements (i.e use of arrays and loops)
  - 4: Correct & Elegant. Comments.

- Answers to questions on the assignment (Total 1)
- Midpoint Check email (Total 1)
- Lab (Total 2) – DON'T forget to submit this with this assignment if you have not made an early submission the week before.

Assignment 3: Arrays & Loops

In this assignment, you will learn to write elegant code. By the end of this assignment, you should be able to

- use `for` loops (iterative loops)
- use `while` loops (conditional loops)
- SAS: use one dimensional `arrays`

Setting Up

1. Determine the folder you will be working in. Create one if you need a new one, or use an existing one if that makes more sense.

Pick a problem: Pick two tasks (we will refer to these ask Task 1 & Task 2 below) that you need to do that would be good to use arrays and loops for (i.e. tasks that require repeated programming that is very similar). If at all possible, try to identify something you did in assignment 2 that can be improved by using arrays and loops. If you are having trouble identifying these, email me BEFORE the midpoint check due date for a meeting.

Getting Data: You should try to use the same dataset you used in assignment 2.

Midpoint check: Email Dr. Kum (please use numbers that are corresponding)

1. *A short description of your progress on lab3 (if you submitted lab3, just write "submitted")*
2. *Which option you will do for the assignment: structured or self-defined (if so, answer the following additional questions)*
3. *[self-defined ONLY] The dataset you will be using to do assignment 3, and the name of the SAS dataset. Note this MUST be a SAS dataset, as reading in files can be difficult. i.e. You need to have done the work required to read in the dataset you are interested in and have saved it out as a SAS dataset. This is to ensure that you can focus on practicing coding with loops and arrays for the assignment, rather than struggling with I/O. (if you are using the same as assignment2, just write "assignment 2 dataset")*
4. *[self-defined ONLY] Describe the two tasks you will program using arrays and loops. If either of the tasks are improving your program from assignment 2, identify them as "Improvement to assignment 2". This is your planning phase of the code. This does not need to be long, shorter the better, but you need to have thought about your task at hand and how you will be using arrays and loops for the task.*

For those using new data only. Describe the Data

- Describe the dataset, including the number of obs and variables that are in your subset of data
- Check your data. Examine each variable for its range (min to max) or number of categories. Discuss any anomalies you see in this data (such as an abnormally high or low values)

P1. Coding with Arrays and Loops

Your program should do the following. Please indicate BEFORE the code in comments where each of these items occurs in your code. Plan your code as follows:

- **P1.1 [No Loops limited task to 5 elements]**
  - Code Tasks 1 & 2 without using arrays or loops. If there are many elements to work with, you only need to code the first 5 elements. In this case, only process the first five elements in arrays and loops as well for P1.3 & P1.4 If this was done in assignment 2, you should be able to just copy & paste relevant sections of your code. Save this out as lname2_v1.sas. v1 here refers to version 1. The logs should have corresponding names.
- **P1.2 [Descriptive]**
  - Calculate appropriate descriptives for numerical and categorical variables in your dataset.
- **P1.3 [Using iterative loops]**
  - Now code Tasks 1 & 2 using arrays and/or loops. You may use either the `for` loop (iterative) or the `while` loop (conditional), but one of them must use the `for` loop. Remember to keep the same code for generating the descriptives at the end. These programs should be saved out with postfix "_v2" with corresponding logs.
- **P1.4 [Using `While` loops]**
  - Pick one task that you coded using the `for` loop, and code again. This time using the `while` loop. Remember to keep the same code for generating the descriptives at the end. These programs should be saved out with postfix "_v3" with corresponding logs.

NOTE: the more different syntax you try to use, the more you will learn. There are many different ways to do one task. You have to try different ways to fully understand the differences and find out which works best for you as well as understand what might work best in a given situation. Try to use three different things. The difference can be minor. One must be a while loop, one must be a for loop. There are many ways to specify arrays. You are writing at least three loops.

- **P1.5 [Testing]**
  - Now you should have 3 programs that do identical things. That means the output should also be identical. Parts of the log will not be identical, because this reflects the code being run. But the results of the descriptive analysis on the variables constructed using arrays and loops should be identical. Compare the output by either using Word/review/compare or diff (small program that will compare text files.) Confirm that the RESULTS are identical. Read P2 Readme file below to keep track of testing & debugging.
- **P1.6 [Debug]**
  - If anything is not identical, debug your code and fix. Retest the new code, and repeat until you pass the test (i.e. results are identical).
- **P1.7 [Extend to real task]**
  - If you only processed the first 5 elements as discussed in P1.1, now extend the arrays & loops to process ALL of your elements. If you have 5 or less elements, then use your imagination and extend your program to add one more element (i.e. make a fake additional variable that could apply. It is ok if the values are incorrect or is just a copy of an existing variable. The purpose is for you to learn to code.) Remember to keep the same code for generating the descriptives at the end. These programs should be saved out with no postfix (i.e., no _v) "lname3.sas" with corresponding logs, because this should be your final version. You might need to come back to this in future assignments.

[Option 1: Structured Assignment] I planned the code in the comments of Assignment 2 (posted under the Solution menu). The required dataset is ipstatw under the Data menu. Look at the comments and completed tasks 3.1-3.3 (required). There are also optional advanced exercise: 3.4-3.8; Use this to follow the description above from P1.1 to P1.7

[Option 2: Self defined] follow the guidelines above to write your own loops in your own code.

P2. Readme file

Create a text file called readme.txt in your working directory. If you are using the same directory as assignment 2, you might name the files readme3.txt to distinguish it from the readme file you created for assignment 2. Answer the following questions in the readme file and send as attachment with code and output. Start to think about how you might minimally format the text file for better usability (i.e. use things like * for bullets, ---- for lines, << >> for titles etc).

1. State what you were not able to complete, if any.
2. Copy & paste the email sent for mid point check
3. If you have any updates after completing the coding for Task 1 & 2 described in the first email, update the description under heading <Updated Task Description>
4. How many times did you have to test until everything was identical?
5. What program did you use to compare if the output was identical or not?
6. Write up at least one bug you fixed during testing by filling in the following
    - <<Buggy code>>: copy lines of buggy code
    - <<Working Code>>: copy lines of correct corresponding code