Class Exercise 8

This exercise is based upon Chapters 3 and 4 of Delwiche and Slaughter's "The Little SAS Book". We will learn a little bit more about the `ARRAY` statement in the `DATA` step.

Instead of using the `DO OVER` syntax, I will start off by using the less-preferred (but more common) `DO` loop syntax with an indexing variable. For this particular study, we will not be inputting any data, but instead creating it on our own. Note that the `ARRAY` statement allows you to include variables in the array (`X1-X10`) that have not yet been input or otherwise defined.

```
DATA A;
ARRAY X X1-X10;
DO I=1 to 10;
  X(I)=RAND('NORMAL');
END;
RUN;
```

Run the code listed above; SAS selects a start value for the random number sequence based on the current clock time. Be careful if copying and pasting this code–the single quotes in the PDF document may be interpreted as backward carats–so you may have to restore them. In this particular example, our ultimate objective is to generate random samples of size 10 from a standard normal distribution. We use the array format to simplify the assignment of the random outcomes to the variables `X1-X10`.
As you'll see when you run the code, this particular set of commands would only generate a single random sample. To generate multiple random samples, we need to place our `DO` loop inside another loop, and then save the results from each sample. We use the `OUTPUT` statement to tell SAS to create a new record in data set `A`; this record contains the current value of all variables referenced in the data step, and hence can be used to save as many normal random samples as we would like.

```
DATA A;
ARRAY X X1-X10;
DO J=1 to 100;
  DO I=1 to 10;
    X(I)=RAND('NORMAL');
  END;
  OUTPUT;
END;
RUN;
```

You can look at the datasheet for `A` and confirm that you now have a normal random sample. You should also try to run the above code without the `OUTPUT` statement (perhaps try commenting it out); you'll see that your single random sample simply gets written over 99 times, and you're left with a single sample at the end of the 100 iterations (the value of J in the single-row data set is a tip-off). If you needed to compute the mean, you could

include the command `XBAR=MEAN(OF X1-X10)` immediately before the `OUTPUT` statement; re-run the code after doing this and report your results.

We can save the file using `FILE` and `PUT` as we did in class. Often, we don't want to save the variables without formatting–the output variables will have too many significant digits for easy inspection. As a final series of steps, save your random file to an appropriate external directory, e.g.

```
DATA B; SET A;
FILE '/home/grego1/STAT 540/EXER8.TXT';
PUT (X1-X10) (5.2 +2) +4 XBAR 5.2;
RUN;
```

You may need to enclose the filename in double quotes or enter the single quotes yourself. Even though `XBAR` has the same format as `X1-X10`, we separated out `XBAR` to add 2 leading spaces and eliminate any trailing spaces. How does your file look?

NOTE: The `PUT` command seemed to be sensitive to the operating system; either one of the following commands may work better for you:

```
PUT (X1-X10 XBAR) (5.2 +2);
PUT (X1-X10) (5.2 +2) (XBAR) (+4 5.2);
```