Class Exercise 11

This exercise is based upon Chapters 11 and 13 of the SAS Advanced Certification Prep Guide, including a macro review and an investigation of alternative methods for sampling records from a data set.

We will first start out by revisiting the meddb data, and using macro variables identified with Type of Service to process data in groups. The basic idea here is to close the loop on the process we started earlier. If we have a straightforward way of saving each Type of Service as a macro variable, then we ought to be able to use %DO-%END to generate separate analyses for each Type of Service. The following macro TOTCLAIM allows the user to specify the permanent library where meddb is saved as a permanent SAS data set (so make sure you have saved meddb as a permanent SAS data set before proceeding). Then it uses PROC SQL to acquire all possible values of TOS, find the number of such values (&NREC), save each of those values as a macro variable (&TOS1-&TOS4), and then plot histograms of claims for each Type of Service. Note SGPLOT’s complicated WHERE clause. I thought I might need to use some of our advanced methods to have TOS1-TOS4 interpreted correctly by either the WHERE clause or the TITLE statement, but that didn’t prove necessary.

options mlogic mprint symbolgen;

%macro totclaim(lib);

libname perm "&lib";

data meddb;

set perm.meddb;

run;

proc sql;

create table tosclaim as select tos, sum(claim) as totalclaim

from meddb

group by tos;

proc sql;

select count(\*) into :nrec from tosclaim;

%let nrec=&nrec;

select tos label="Type of Service"

into :tos1-:tos&nrec

from tosclaim;

%do i=1 %to &nrec;

proc sgplot data=meddb (where=(tos="&tos&i"));

title "Claims for Service Category &tos&i";

histogram claim/scale=count fillattrs=(color=red);

density claim/type=kernel(c=3);

run;

%end;

%mend;

Now run the macro using the appropriate directory (i.e.):

%totclaim(lib=/home/grego1/STAT 541)

Once I’ve constructed a macro, I always think of generalizations or refinements. What other refinements might you make to the above code?

Now let’s look at a couple alternatives to sampling methods we just studied. We will work once again with Fall08. Read Fall08 into the WORK directory and then run the code below to extract 400 records.

proc sql inobs=400; create table original as select noobs, major, class, cltotgpa, gender, race, regstat from fall08; quit;

The code below should sample 5 records from each combination of gender and race. We assign a uniform random variable to each record, and then sort each Gender/Race subgroup on the uniform random variable—this should effectively shuffle the records. At this point, we then extract the first 5 records for each group using some commands (FIRST.variablename) that you likely have not seen since STAT 540. What do you think would happen if a subgroup had fewer than 5 records? Would the program still work?

proc sort data=original;

by gender race;

run;

data original;

set original;

shuffling=ranuni(0);

run;

proc sort data=original;

by gender race shuffling;

run;

data sample;

set original;

by gender race;

if first.race

then counter=1;

else counter+1;

if counter<=5;

run;

Inspect SAMPLE to confirm that it worked as planned—how many combinations of Gender and Race were created? Graduate students should convert the above code to a macro program that allows the user to specify the sample size as a variable and to specify a single BY group variable. Test the program with a sample size of 6 and BY group variable RACE.