Robyn Crook - 2008

Week 4 – Creating derived variables from existing data, making two-way frequency tables.

Last week we looked at how to make secondary data files using the 'data' and 'set' commands. In the Project Program Description pages, the last section of Component 1 deals with taking one of those secondary data files (either the random cws file, or your file containing first or last observations for cws) and forming derived variables from two existing columns of data. You will then conduct simple analyses to obtain descriptive statistics using 'proc sort', 'proc means' and 'by' statements.

After this week you will have all the information you need to work through to the end of Component 1.

To form a new variable from existing data (like a ratio or a product of data from two different columns), the code is simple. We need to make a secondary data file where we will add our new column of data, and specify how this new variable is calculated.

This is an example (i.e., fictional) dataset that we will use to demonstrate this procedure. Let's assume we have data for the body lengths, heights and weights for a population of cattle. Different individuals are in the first column of data, then the measurements in the next four columns, then colour and sex.

```
Title 'Forming a derived variable';
Data cattle;
Infile 'U:\cattle_measurements.egc';
Input indiv $1 length 3-5 weight 7-9 height 11-12 age 14 colour $16
sex $18;
run;
```

Now we can add the code to form a new variable. Perhaps we want to make a derived variable that is length divided by height for each individual. We will call it 'standard', as though we were using a standard measure of overall proportions.

```
data cattle2;
set cattle;
standard=length/height;
run;
```

Sometimes we may want to exclude certain columns from our secondary data set, for example if we wanted to have only measurements represented, and not the discrete variables 'colour' and 'sex'. In this case we add a line to the code before the 'run' statement.

```
data cattle2;
set cattle;
standard=length/height;
keep length weight height age standard;
run;
```

Now 'cattle2' includes the new variable 'standard' as well as the other measures we listed, but the columns 'colour' and sex' have been excluded.

Another way of making a new variable is to create a discrete variable from a continuous one. For example, in our data set we have 'weight' as a continuous measure. We could create a new variable that divides the weights into the discrete categories of light 'L', medium, 'M' and heavy, 'H'. We will call this new variable 'size'.

If we wanted to make sure we had a roughly even split between the three new categories, we need to decide where along the weight distribution we will separate into light and medium, and medium and heavy. Since we have three categories, we would like ideally to have a third of the individuals classified as light, a third as medium and a third as heavy. We can find the critical weights where we will make our split by using 'proc freq'.

```
proc freq data=cattle;
tables weight;
run;
```

This outputs a frequency table that lists cumulative frequencies and cumulative percentages of the weights. To decide where our cutoff weights will be for light, medium and heavy, we simply identify the weights at the 33rd and 66th percentiles. Now that we have these (I've made up the numbers here), we can form the new variable 'size' using if/then statements.

```
data cattle3;
set cattle;
if weight<235 then size='L';
if weight ge 235 and weight le 457 then size='M';
if weight>457 then size='H';
run;
```

So now we have an equal number of observations in each of the categories L, M, and H.

Making two-way frequency tables.

The last part of this section asks you to use proc freq to make two-way frequency tables. This means that rather than making a table (as we did above), of weights for the dataset, we might want to look at the frequencies of weights within each sex, or for each colour. The code for this is very similar to the normal proc freq code that we have already used.

```
Title 'Making a two-way frequency table of weights by sex';
proc freq data=cattle;
tables weight*sex;
run;
```

This creates a frequency table of the weights for each sex. We could go a step further and make a frequency table of weights, separated both by sex and by colour.

```
Title 'Making a two-way frequency table of weights by sex and colour';
proc freq data=cattle;
tables weight*sex*colour;
run;
```

Next week: Merging data files; catch up day.