Week 3 – Making secondary data-files and review of descriptive statistics.

So far we have looked at how to import data into SAS and make basic corrections to the entire dataset. Sometimes we will want to only perform analyses on some part of the dataset, not the whole thing. For example, in the first paragraph of the project description you are asked to conduct simple analyses on blue males only. To do this, we can form a secondary data file containing only the observations we need, as a subset of our original file. We do this by using the SET and IF statements.

Using the same sample code as last week:

```
Title 'Spatial memory code';
*Animals are A-F, trialno is 1-5, time is in seconds, ten replicates
per animal per trialno;
Data spatmem;
Infile 'U:\spatialtimes.egc';
Input animal $1 trialno 3-4 time 6-9 replicate 11-12;
run;
```

Now we have the data imported (I've excluded options and corrections to save space, your programs will have these). Next we can form a secondary dataset containing only a subset of our observations, using the DATA and SET commands. I have named the secondary dataset 'subset1', you can choose what you like for your programs.

```
Title 'Forming a secondary dataset containing only data for animal A
that has no missing values for trialno';
data subset1;
set spatmem;
if animal='A';
if trialno=' ' or trialno>5 then delete;
run;
```

Now we have a secondary dataset called 'subset1' that contains only data for animal A where 'trialno' is neither missing nor greater than 5. Our original dataset is unchanged, and we can still work with it if we choose to. We can now obtain descriptive stats for the continuous variable 'time' in subset1.

```
Title 'Descriptive Stats for subset1';
proc means data=subset1 mean median min max stderr maxdec=2;
var time;
run;
```

This gives us the mean, median, minimum, maximum and standard error for the variable 'time', pooled across all trial numbers. Note that now we have two datasets, we must specify after proc means which one we want to use by writing data=subset1.

We could also ask for means to be calculated separately for each trialno by using a 'by' statement under the 'var' statement. First we need to sort the file by our chosen 'by' variable. If this step is skipped SAS will generate an error.

```
proc sort data=subset1;
by trialno;
run;
```

```
proc means data=subset1 mean median min max stderr maxdec=2;
var time;
by trialno;
run;
```

This will produce separate descriptive stats of 'time' for each trialno (trial 1 through trial 5).

For discrete variables (like trialno), taking a mean or median is not what we want. Instead we want to look at frequencies. In your program you would look at frequencies of things like gosling colour, who, gosling condition, etc.

```
proc freq data=subset1;
tables trialno;
run;
```

This will give us a table (since we specified 'tables' under the proc freq command) in the output window showing how many observations there are (the frequencies) for each trialno (we are supposed to have ten replicates for each trial number).

Proc means and Proc freq are important code fragments for finding descriptive stats for your gosling data.

Forming secondary files by selecting one observation from several replicates.

In this hypothetical dataset we have multiple replicates for each animal for each trial number (i.e., Animal A has 10 different replicate times for trial numbers 1 through 5, so 50 in total). In some analyses, we only want one of these replicates per 'trialno' represented to avoid pseudo-replication, or we might be interested in the effect of replication on our results. We can make secondary data files to select one replicate per trial for us. We can ask the program to select one of the ten replicates based on its order, or at random. You need to do both of these for your gosling programs.

First we need to sort the file by trial number (trialno) and then by replicate using the command Proc sort. If you have more than one dataset you must specify which one you want to sort. Under the proc sort line you specify the sorting order using 'by'.

```
Title 'Selecting the first of multiple replicates';
proc sort data=subset1;
by trialno replicate;
run;
```

Now we can select the first replicate for each trial number to be placed in another secondary file, which will give us a data set containing only data from the first replicate for each trial. This has been called 'subset2'.

```
data subset2;
set subset1;
by trialno replicate;
if first.trialno;
run;
```

Because we have sorted by trial number then by replicate, 'if first.trialno' will select the first observation for each trialno, which will be replicate 1.

We could do exactly the same to take the tenth replicate.

```
data subset3;
set subset1;
by trialno replicate;
if last.trialno;
run;
```

The other approach to this is to sort the replicates randomly by placing a column of random numbers in the dataset, then taking whichever one ends up being first into the secondary data set. This is a three-step process. First we make a new subset file using some random-number generating code.

```
Title 'Selecting a random replicate for each trialno';
data subset4;
set subset1;
retain seed1 987654321;
call ranuni (seed1,randomn);
run;
```

This code is very specific – you will need to transfer it exactly to make your program do this (remember to change the variable names). 'Retain seed1' means we are specifying where our random number generator should start (we 'seed' the random number generator with the value 987654321, you could choose any value). 'Call ranuni' ('ranuni' is short for random, uniform distribution) means that a random number is allocated (called) to each row in our dataset, and is placed in a new column called 'randomn' (for 'random number'). You could call this column anything you like.

Now we sort this new dataset by trialno, then by the random numbers we just allocated to each replicate.

```
proc sort data=subset4;
by trialno randomn;
run;
```

Now, since the replicates are still in trialno order (1-5), but the replicates are randomly ordered, we just select whichever replicate is first (or last, whichever).

```
data subset5;
set subset4;
by trialno randomn;
if first.trialno;
run;
```

We could now conduct analyses on these secondary files using proc means and proc freq as previous.

Next week: Forming derived variables and ordinal variables. Using proc freq to make two-way frequency tables.