

**Lab 2: Exploring data in SAS**  
**STT 421: Summer, 2004**  
**Vince Melfi**

Today we'll learn a bit about using SAS to explore distributions graphically and numerically. In the process we'll also learn about some of the data manipulation capabilities of SAS.

## Histograms

The data from Newcomb's experiment to determine the speed of light (from Table 1.1 in the text) are in the file `U:\msu\course\stt\421\summer04\newcomb.dat`. If you examine the file in Notepad, you'll see that the first number is 28. This represents the deviation of Newcomb's time measurement from 24800 nanoseconds. So Newcomb's first time measurement was 24828 nanoseconds, i.e., 0.000024828 seconds. Since the distance the light traveled was 7400 meters, this yields a measurement of

$$\frac{7400}{0.000024828} \approx 298050588.05 \text{ m/s.}$$

Although it takes a bit of work to get to the original data values, it's worth recoding the values, since we've seen (Exercise 1.67) that statistical packages sometimes have trouble with very large or very small numbers.

The following program reads in the data and then use SAS to draw a histogram. It also shows how SAS can manipulate data, and introduces **proc means**.

```
data newcomb;
  infile 'U:\msu\course\stt\421\summer04\newcomb.dat';
  input time1;
  time2 = (24800 + time1)/1000000000;
  sqtime = time1*time1;

proc print data = newcomb;
  var time1 time2 sqtime;
  title 'Newcomb speed of light data';

proc means data = newcomb;
  var time1 time2 sqtime;
  title 'Proc means applied to the Newcomb data';

proc univariate data = newcomb;
  var time1;
  histogram time1;
  title 'Histogram of the Newcomb data';

run;
```

## Explanation

1. In the **data** step we read in the data, calling the input variable `time1`. Then we created the actual time in seconds, `time2`, by transforming `time1`. We also obtained the square of `time1` and called it `sqtime`. You'll see why we did this below.
2. Then we used **proc print** to print out the three variables with a title.
3. Then we applied **proc means** to the three variables. Find this output in the **Output** window to learn what **proc means** does<sup>1</sup>. Verify that the mean of the original data `time2` can be obtained from the mean of the transformed data `time1` via the transformation we used to obtain `time2`. Verify that the mean of `sqtime` is *not* equal to the square of the mean of `time1`.
4. Then we used **proc univariate** with the additional request for a histogram. This should produce the usual spate of output that we saw in the previous lab plus a histogram.

Note that the default histogram is not very informative. The SAS algorithm for choosing the classes for the histogram is fooled by the outliers into providing too few classes. We can remedy this by using the **midpoints** option. In the above program, replace

```
histogram time1;
```

by

```
histogram time1 / midpoints = -55 to 55 by 5;
```

and see whether the histogram is better. You may have to scroll down in the graphics window to find the new histogram. Now replace the **histogram** line by

```
histogram time1 / midpoints = -55 to 55 by 1;
```

Is this better or worse? Again, you may have to scroll down to find the new histogram.

## Missing values

Another way to produce a more informative histogram of the non-outlier data is to ask SAS to ignore the outliers. We can do this by making them “missing” values, which SAS represents as a period. First add the following two lines in the data step after the line defining `sqtime` in order to have a variable `time3` that contains all the nonnegative (non-outlier in this case) observations.

```
time3 = time1;  
if time3 < 0 then time3 = .;
```

---

<sup>1</sup>Note that all the information from **proc means** is available from **proc univariate**, but the latter includes so much more output that it's sometimes hard to find the basic information that **proc means** provides.

The add the following lines just before the **run** line:

```
proc univariate data = newcomb;
  var time3;
  histogram time3;
  title 'Histogram of the Newcomb data without the outliers';
```

## Boxplots and sorting

We have seen that side-by-side boxplots provide a nice way to compare distributions of data. We'll use boxplots to compare self-concept scores for males and females from a dataset on 78 seventh grade students. (This data comes from Table 1.6 on p. 33 of the text.) We'll also learn about **proc sort**, and how to compute means separately for subsets of the data. Here is the SAS code. Please type it in and run the program.

```
data eddata;
infile 'U:\msu\course\stt\421\summer04\eddata.dat';
input obs gpa IQ gender self_concept;

proc print data = eddata;
  var self_concept gender;

proc sort;
  by self_concept;

proc print data = eddata;
  var self_concept gender;

proc sort;
  by gender;

proc print data = eddata;
  var self_concept gender;

proc means;
  var self_concept;
  by gender;

proc boxplot;
  plot self_concept*gender;
  title 'self concept by gender';

run;
```

## Explanation

1. In the **data step** we read in the data, which included five variables, `obs`, `gpa`, `IQ`, `gender`, and `self_concept`. Gender is coded “1” for female and “2” for male.
2. Then we printed `self_concept` and `gender`.
3. Then we used **proc sort** to sort the data by `self_concept` and printed `self_concept` and `gender` sorted this way.
4. Then we used **proc sort** to sort the data by `gender` and again printed the sorted data.
5. Then we used **proc means** to compute the mean of `self_concept`. But the **by gender** statement told SAS to compute the mean separately for the two genders. This wouldn’t have worked if we hadn’t sorted the data by gender. (In principle there are ways around this problem, but I find it best to sort first.)
6. Then we produced boxplots of the data. The line **plot self\_concept\*gender** told SAS to produce separate boxplots of `self_concept` for the two values of `gender`.

Look at the **Output** window and the **Graph** window to make sure your program did what was expected, and also to make sure you understand how the program works. In particular, can you figure out what the “+” symbol represents on the boxplot?