Chapter 2
The data analysis process

2.0 Introduction

The topic of chapter 2 is the collection of data. Interpretation of data depends critically on how it was gathered. In some observational studies we may be interested only in describing the characteristics of a sample. For this purpose, the graphic portrayals of data you have seen in Chapter 1 of POD, and other techniques which you will see in succeeding chapters are perfectly fine. In other circumstances the goal of the data gathering is to acquire a sample for the purpose of generalizing to a population. As an example, we may take a sample of high school students and ask the number of hours they spend studying in a typical week. Our purpose is not just to tabulate how many hours the students in the particular sample studied; we wish to generalize beyond the sample to the population of students. In order to make statements about the population, we must select the sample so that it has a good chance of "reflecting" the characteristics of the population. The critical aspect of sampling that enables us to generalize is that the sample is a "random" sample. There are different methods of random sampling, and each of them involves generating random numbers – it is at this stage the TI-83 enters the data gathering picture.

2.1 The random number generating capabilities of the TI-83.

It seems odd to talk about computers or calculators generating "random" numbers – everybody knows that calculating machines operate by executing a step of pre-defined instructions. How can calculators generate random numbers? It turns out that calculators don't actually produce truly random numbers; they produce what are known as "pseudo-random" numbers. The generation of pseudo-random numbers is accomplished by creating a sequence of numbers using a starting number, called a "seed." The seed is built into the calculator in the factory, and everyone who has a TI-83 will start out in the same place with respect to pseudo-random numbers. (If you have a new TI-83 and have not generated any random numbers yet, find someone else with a new one and check it out.) The process works something like this….

$$seed \times (magicNumber_1) + magicNumber_2 \longrightarrow pseudoRandom\#_1$$
$$pseudoRandom\#_1 \times (magicNumber_1) + magicNumber_2 \longrightarrow pseudoRandom\#_2$$
$$pseudoRandom\#_2 \times (magicNumber_1) + magicNumber_2 \longrightarrow pseudoRandom\#_3$$

and this sequence of pseudo-numbers continues generating numbers for a very long time before finally repeating itself. For our practical purposes the pseudo-random numbers generated by this process are just as good as actual random numbers. The "magic"

numbers are not, of course, magic – they are numbers carefully chosen by mathematicians and computer science experts for the purpose of generating well shuffled and unpredictable numbers. This process as implemented in most calculators will generate pseudo-random numbers in the interval, $0 \le r < 1$.

It is possible to mathematically transform these random numbers into different random numbers as desired. For example, to generate random integers from 1 to 6, such as in a dice game, the following mathematical procedure could be used:

Step 1: Generate a pseudo-random number between 0 and 1
Step 2: Multiply that number by 6.
Step 3: Add 1.
Step 4: Round down.

A different method of transformation might be to build in functions that perform operations such as those multiplication and rounding steps; and this is what the TI-83 has done. The TI-83 has a built-in function to get that initial pseudo-random number, and also has some mathematical procedures that generate random numbers in forms commonly used in statistics. To use the random number generation features of the TI-83, press

MATH > PRB

and you will see a screen that looks like the one at right:

```
MATH NUM CPX PRB
1:rand
2:nPr
3:nCr
4:!
5:randInt(
6:randNorm(
7:randBin(
```

Choice #1, "rand," generates a pseudo-random number in the interval from 0 to 1 as described above. Choice #5, "randInt," generates random integers in an interval determined by the user. If you are at this screen now, choose **rand** and then press **Enter > Enter > Enter.** You should see three random numbers, all between 0 and 1. On my calculator the numbers are: .9435974025, .908318861, and .1466878292. (Gee, they sure look random to me!)

With this information as background, we will now tackle Example 2.3.

Example 2.3: Selecting a Random Sample of Glass Soda Bottles

Breaking strength is an important characteristic of glass soda bottles. If the strength is too low, a bottle may burst—not a desirable outcome. Suppose that we want to measure the breaking strength of each bottle in a random sample of size n = 3 selected from four crates containing a total of 100 bottles (the population). Each crate contains five rows of five bottles each. We can identify each bottle with a number from 1 to 100 by numbering across the rows, starting with the top row of crate 1, as pictured in POD.

Generating random integers with the TI-83

To solve the problem of sampling in Example 2.3, we will generate 3 random integers, each with values in the range from 1 to 100. The 3 random integers will correspond to random choices from the 4 crates. Since we are sampling without replacement we might have to generate more than 3 random integers. If we do get repeat random integers, we will just keep on going until we have 3 different integers. To see how to do this we will repeat our sequence of strokes above, and then go one step farther.

MATH > `PRB` > `randInt(`

Well, now we're in a pickle. The calculator is sitting there with a blinking cursor, acting like it is expecting you to give it some information. In fact, that's exactly what it is waiting for. At the end of the above sequence of keystrokes is, essentially, a mathematical function. This particular mathematical function is named "**`randInt`**" which is certainly more creative than the usual *f* and *g* names commonly seen in mathematics classes. In those math classes, *f* and *g* need values before they can be used. Those values are usually "*x*" values, giving us *f(x)* and *g(x)*. Like the *f* and *g* functions, "**`randInt`**" needs some information in order to do its calculations. That's why it has that open parenthesis, "(", at the end of its option in the list. When the TI-83 needs more information to perform a function evaluation, it signals this with a single open parenthesis – your task is then to supply the right information in the right order, and – oh, yes – supply the closing right parenthesis.

It is not always obvious what information is desired, and it is seldom obvious what order the calculator wants you to supply the information. That's why the calculator comes with a manual. (You may have the manual on a CD.) Looking up `randInt` in the TI manual's index and then flipping over to the recommended page reveals the following cryptic syntax:

randInt(*lower, upper* [,*numtrials*])

This sort of thing looks at worst like gibberish, and at best like some ancient dialect nobody speaks any more. It turns out that it isn't gibberish, it is some ancient dialect, but it is spoken by one group of people: computer programmers. This style of presentation is the traditional way of telling computer types – or in our case, calculator types – how to use functions. Let's take this statement apart and look at its parts:

randInt -- the name of the function, and is followed by a "("

lower, upper [,*numtrials*] -- the information the function needs. The "[ ]" means that the information is optional.

The information that is needed by the calculator to evaluate a function depends on the function, and will make sense when you use it. For **`randInt`**,

lower – the smallest random integer you want
upper – the largest random integer you want
numtrials – how many random integers you want.

Recall, we want 3 random integers between 1 and 100. Let's complete the calculator sequence using this information:

MATH > `PRB` > `randInt(1,100,3)` > ENTER

Our calculator returns {38 , 12, 46}. Your calculator will, of course, return different numbers. What happens if your calculator gives you 2 numbers that repeat? In that case, you can generate 3 more random integers, and if needed – not likely. – 3 more. After executing the sequence above, you don't need to redo the whole sequence of keystrokes – just press **ENTER** again to execute the whole sequence of keystrokes on the TI-83.

At this point we have accomplished our goal – generating a random sample of size 3, from our population of crates. We will take Crate #38, Crate #12, and Crate #46.

2.2 Afterword

The TI-83 has a whole suite of random number generation capabilities. As you study different parts of probability and statistics, these functions can be very helpful. You might want to just browse in that chapter of your TI manual to see what's there.