17.871
Spring 2012
How to Use the *STATA* **infile** and **infix** Commands

STATA is a very flexible program, allowing you to read-in and manipulate data in many different forms. This is good, because social science data come in various formats, requiring great flexibility among the statistical packages social scientists use. Unfortunately, the *STATA* manual we are using only covers how to input the simplest of data sets. The simplicity of the examples in that book border on the trivial. The purpose of this handout, therefore, is to introduce you to the use of the *STATA* **infile** and **infix** commands, going a little more in depth than the Hamilton book goes.

An easy case
Let us say that you have data about four students who have taken a standardized test. You have their first names, their ages, and their scores on two tests (Test 1 and Test 2). Here are the data in tabular form:

Name	Age	Test 1	Test 2
Bob	18	95	18
Carol	21	43	27
Ted	14	67	9
Alice	12	23	31

The easiest way to get these data into *STATA* is for you to fire up the STATA Data Editor and just type the data into the spreadsheet-like interface.

The next-easiest way to get the data into *STATA* is for you to type the data into a file and then let *STATA* read it in. Let's say your Athena[*] username is *janedoe*. You could create a data file using a text editor such as *EMACS*. Let's say you saved the data in a file in your directory named scores.dat. The file scores.dat looks like the following.[1]

Exhibit 1

```
Bob 18 95 18
Carol 21 43 27
Ted 14 67 9
Alice 12 23 31
```

Then, from within *STATA* you would type the following:

[1] Note that all files that *STATA* reads **must** end with a carriage return.

[*] Athena is MIT's UNIX-based computing environment. OCW does not provide access to it.

```
infile str5 name age test1 test2 using /mit/janedoe/scores.dat
```

The word **infile** is the command name. The words **name**, **age**, **test1**, and **test2** are the variable names. ***STATA* variable names must be 32 characters long,[2] or shorter, and begin with a letter or underscore (_).** *STATA* generally assumes that variables contain numbers. If the data are not numeric, *STATA* needs to be told that a variable is non-numeric (a text "string") and the longest the text string can be. That is the function of the word **str5** before the word **name**: to specify that **name** is a text string that may be no longer than 5 characters long.

After you have typed in the **infile** command, you should then issue the **compress** command. That is because *STATA* has some tricky memory management problems, and this command will convert all the variables to their most efficient internal representations.

An example with fixed field data

The above example is the simplest case of reading in data for use by *STATA.* In addition to being a short, narrow data set, we are able to express this data set using what we call a "free form" format: the data are just freely typed into the computer, with nothing but a space to separate variable values. Data sets are rarely this simple. For instance, if you had someone with a first name of "Mary Jane" you would have to get rid of the space (by typing in something like MaryJane or Mary_Jane). If you had thousands of observations (instead of four) and variables (instead of four) the spaces necessary to delimit individual observations might cause the data set to balloon beyond what is really necessary to contain the unique information among the data. For these, and other, reasons, data sets are typically organized using a "fixed format". With fixed format organization, each line begins a new observation[3] and each variable occupies the same column(s) on each line.

The fixed format version of the data would look something like the following:

Exhibit 2

```
Bob  189518
Carol214327
Ted  1467 9
Alice122331
```

To read in this data, you would use the *STATA* **infix** command. This is what you would type to read in the data from Exhibit 2:

```
infix str5 name 1-5 age 6-7 test1 8-9 test2 10-11 using scores.dat
```

[2] Earlier versions of *STATA* (i.e., versions 6 and earlier) limited variable names to 8 characters in length. I often maintain that convention, for compatibility reasons and for brevity.

[3] There are important exceptions that will be dealt with below.

A word about missing data
Sometimes data will be missing from a data set. There are three ways of indicating missing data in *STATA*: (1) the lone period, (2) missing value codes, and (3) blanks.

The lone period
STATA generally represents missing values with a lone period where the value of the variable should be. For instance, say that Ted would not tell us his age. We could account for this fact by placing a period where his age should go, either in free form:

```
Bob 18 95 18
Carol 21 43 27
Ted . 67 9
Alice 12 23 31
```

or in fixed format:

```
Bob  189518
Carol214327
Ted   .67 9
Alice122331
```

STATA would then exclude Ted from any calculations or procedures that required the use of the age variable.

Missing value codes
Most social science data sets use missing value codes to indicate missing values. It is most common to give someone an impossible value for that variable when the variable's true value is missing, and then to tell the statistical program about that value. So, for instance, ages must be positive. Therefore, we could make the value of -1 indicate a missing age, give Ted an age of -1, and then tell *STATA* what we've done. The data would look like this:

```
Bob  189518
Carol214327
Ted  -167 9
Alice122331
```

There are then two ways to change a missing value code into an actual missing value representation in *STATA*. The most general way is to use the **replace** command:

```
replace age=. if age== -1
```

The above command tells *STATA* to replace values of **age** with the missing value representation in those cases where **age** equals -1.

This technique can get tedious if you have lots of variables with the same missing value code. *STATA* has a command, **mvdecode**, which converts missing values to their proper representation. For instance, if **age**, **test1**, and **test2** all used -1 for missing data, you could issue the following single command to accommodate the missing values:

```
mvdecode age test1 test2, mv(-1)
```

You can use up to 27 different missing value codes. For instance, let's say you're working on a public opinion survey. One question asks if the person voted for president in the last election. The next question asks for whom the respondent voted, but only if the person reported in the previous question that she had voted for president. In this case, you will have missing data on the second question, for at least two reasons—the respondent may have not voted *or* the respondent may have voted, but then refused to tell you for whom. You might give the "not applicables" the missing value code of -8 and the "refusals" the missing value code of -9. STATA now allows you to collapse these different missing value cases into one, or maintain them in the data set. For instance, the command

```
mvdecode presvote, mv(-8 -9)
```

will turn values of -8 and -9 to the default missing value symbol "." (period). On the other hand, the command

```
mvdecode presvote, mv (-8 = .a -9=.b)
```

will translate values of -8 to the "extended missing value code" of .a and values of -9 to the extended missing value code of .b. For most statistical operations, the effect of the two command is the same. However, if you later want to go back and do analysis on the missing categories themselves (say, you're interested in why people refuse to state a preference for president), then you can recover the original data.

Sometimes, if you have a long list of missing value codes and you don't want to maintain the different codes internally, there's a less tedious **replace** command that will accomplish the same thing as the first **mvdecode** command. For instance, the command

```
replace presvote = . if presvote <= -8
```

produces precisely the same result as

```
mvdecode presvote, mv(-8 -9)
```

Blanks

If *STATA* encounters a blank in a fixed field formatted data set where a variable should be, it interprets that value of the variable as missing. (Question: Why won't this work with free format data?) So, in this example, the following fixed field data set would also indicate that Ted's age is missing:

```
Bob  189518
Carol214327
Ted    67 9
Alice122331
```

Which is the best way to indicate missing values?

You should get in the habit of indicating missing values with missing value codes, rather than using blanks or relying on the lone period. Using blanks is an invitation to sloppiness and errors. The lone period is fine in *STATA*, but not all statistical packages (or high-level programming languages) use the same missing value symbol. Also, the period may be mistaken for a legitimate decimal point, leading to further errors.

Multi-record data sets

Typically, data files contain only one line of data for each observation (or case). There are times, however, when you have so much data about each observation that it won't fit (legibly) on a single line. In those cases you must be explicit to *STATA* in telling it how many data lines constitute a case, or you will be in deep trouble.

Imagine that the data set we've been using put name, age, and the test scores on one line, but then put GPA on the second line, like this:

Exhibit 3

```
Bob  189518
3.35
Carol214327
2.97
Ted  -167 9
0.75
Alice122331
4.00
```

(Ted's age is still missing.) Notice that the variables still occupy the same columns each time, it's just that each individual's data also occupies two lines. You deal with reading "multiple records per case" by modifying the **infix** command in two ways. First, you need to tell **infix** how many lines of data constitute each observation. Second, you need to specify the line on which each variable may be found.

The following is the command you would issue to read in the data from Exhibit 3:

```
infix 2 lines 1: str5 name 1-5 age 6-7 test1 8-9 test2 10-11 2: gpa 1-4
```

The phrase "**2 lines**" indicates that each observation consists of two lines of data. "**1: str5 ... test2 10-11**" indicates the variables that are found on the first line. Likewise, "**2: gpa 1-4**" indicates that the variable gpa is on the second line, in columns 1-4.

Note that you *don't* have to read variables from each line. So, for instance, if you didn't have need for the gpa variable in the analysis you were doing, you could just type:

```
infix 2 lines 1: str5 name 1-5 age 6-7 test1 8-9 test2 10-11
```

and just skip over the second line.

17.871 Political Science Laboratory
Spring 2012