In the previous section, we created a new dataset from within Stata. In this section, we will be importing a dataset from a spreadsheet program. Before I show you how to import a dataset, I just want to tell you that you can copy and paste data from Excel. Just select the cells in Excel, copy the data, and then open the data editor in Stata and paste. You will notice that the data are pasted in the proper order and that Stata has again taken it upon itself to label the variables. One difference though is in the type of the variables. If you click on the a cell with a numeric type, you will notice that the variable type is *byte*. This type allows for integer values only (no decimal points). By pasting the data, Stata identified that all occurrences of this variable do not take a decimal value, so it made the correct assumption that the variable represents an integer. There are several integer types in Stata. We have already used the *int* type. In this case however, the *byte* type makes more sense because data of type *byte* takes up less memory. If you want to know more about this, execute the command **help byte**. The help files shows a table that gives us the information that we need. A *byte* can have a minimum value of -127 and a maximum value of 100 while an *int* can have a minimum of -32,767 and a maximum of 32,740. Since no one will likely have more than 100 children, it is better to use the byte type and save memory. This is all done automatically for us when we copy and paste the data. When we were entering the data, Stata had no idea about the values that we were going to enter in the future. This is why when it first sees a numeric variable, it assigns the *float* type to it. A float type allows for decimal points. When we copy and paste the data however, Stata sees all possible values for each variable and assigns the most suitable data type.

Although copying and pasting is useful, it is still not the best way because it has to be done by hand. The best way to create a dataset is to import it via a spreadsheet program. First, let's tell Stata to clear the memory since we don't need the data that we were currently using:

```
clear all
```

Next, notice that there is a file named *dataset2.xlsx* in the working directory. This is the file that I want to import. To do that, I need to run the following command:

```
import excel "dataset2", sheet("Sheet1")
```

The "import" command is telling Stata that I want it to import an Excel file which is named *dataset2.xlsx*. The import command also specifies the worksheet from which I want to import the data, since each Excel file can contain more than once worksheet. When you run this command, you will notice that in the right-hand side of the screen there are four variables labelled "A", "B", "C", and "D". This is because Stata labels each column using the letters assigned to the columns in Excel. We can open the editor and rename the variables properly. There is however, a better way. In our Excel file, we can enter the names of the variables in the first row and then enter the observations below the names. We can then use the **firstrow** option in the **import** command to tell Stata that the names of the variables are listed in the first row. This way, when Stata imports the data, instead of naming the variables by the column letter used in Excel, it will use the entries in the first row. To do that, enter the following command:

```
import excel "dataset3", sheet("Sheet1") firstrow
```

Wait. Stat produced an error message which tells us that Stata will not complete the operation because the data in the memory will be lost. This is good practice by Stata because it recognizes that we have made changes to the data in memory but that these changes were not saved. Stata fears that we have forgotten about these changes. This is why we need to use the **clear** option with the command in order to tell Stata to import the Excel file even if there was unsaved data in memory:

import excel "dataset3", sheet("Sheet1") firstrow clear

If you open the editor, you will notice that the variables are properly named. Here, we can see why commands are much better than using the user interface. Imagine that there was an Excel file that contained 100 observations. Imagine also that I had imported this file into Stata and saved the dataset. Then one day, I receive an updated file from my colleague that contains 150 observations, and my colleague tell me that he managed to collect more data. Instead of me having to manually edit the data in Stata by adding the extra rows, and instead of having to open the Excel file, copy the data, and then paste it in Stata, all I have to do is to use the import command again. Just one line of code will tell Stata to get the new data and then I will be able to save the data.

Someone might raise the point that just like not everyone has Stata, not everyone has Excel. This is a fair point. Fortunately, Stata is well prepared for this situation because we can use it to import any file with the extension *csv*, which is short for comma-separated values. These files can be written by any spreadsheet program, so any user with any spreadsheet program can create these files. Let's see how this is done. First, I will create a *csv* file with the data. Next, I will tell Stata to import this file. Since I am no longer importing a file saved with the Excel extension, I cannot use the **import excel** command. Instead, I will have to use the following command:

import delimited "dataset4", varnames(1)

Once again, we are telling Stata that we want to import data. This time however, instead of specifying **excel** after the **import** command, we instead use **delimited** to tell Stata that we are importing from a file that delimits, or separates different values. The other difference between this command and the command used to import the Excel file is the option **varnames(1)**. This option is basically doing what the **firstrow** option did when we were importing the Excel file. The option **varnames(1)** is telling Stata that the names of the variables are contained in row number 1, which is the first row. If we open the data editor we can see that the data has been imported successfully. We can now save the data in Stata by typing:

save mydata1

Stata does not execute the command because we have already saved a dataset using this name in the same folder. Therefore, we need to pick another name:

save mydata2