8 Importing data

Copying and pasting

One of the easiest ways to get data into Stata is often overlooked: you can copy data from most applications that understand the concept of a table and then paste the data into the Data Editor. This approach works for all spreadsheet applications, many database applications, some word-processing applications, and even some webpages. Just copy the full range of data, paste it into the Data Editor, and everything will probably work well. You can even copy a text file that has the pieces of data separated by commas and then paste it into the Data Editor.

Suppose that your friend has a small dataset about some very old cars.

```
VW Rabbit,4697,25,1930,3.78
Olds 98,8814,21,4060,2.41
Chev. Monza,3667,,2750,2.73
,4099,22,2930,3.58
Datsun 510,5079,24,2280,3.54
Buick Regal,5189,20,3280,2.93
Datsun 810,8129,,2750,3.55
```

You would like to put these data into Stata. Doing so is easier than you think:

1. Clear out your current dataset by typing `clear`.
2. Copy the data from the PDF documentation the way you would copy anything from any document. (For best results, use Adobe Reader.)
3. Open the Data Editor in edit mode.
4. Select **Edit > Paste special...**.
5. Stata sees that the column delimiters are commas and shows how the data would look.
6. Click on the **OK** button.

You can see that Stata has imported the data nicely.

Later in this chapter, we would like to bring these data into Stata without copying and pasting, so we would like to save them as a text file. Go back to the main Stata window, and click on the **Do-file Editor** button,  , to open a new Do-file Editor window. Paste the data in the Do-file Editor, then click on the **Save** button. Navigate to your working directory, and save the file as `a few cars.csv`. If you do not know what your working directory is, look in the status bar at the bottom of the main Stata window.

Be careful if you are copying data from a spreadsheet because spreadsheets can contain special formatting that ruins its rectangular form. Be sure that your spreadsheet does not contain blank rows, blank columns, repeated headers, or merged cells because these can cause trouble. As long as your spreadsheet looks like a table, you will be fine.

Commands for importing data

Copying and pasting is a great way to bring data into Stata, but if you need a clear audit trail for your data, you will need another way to bring data into Stata. The rest of this chapter will explain how to do this. You will also learn methods that lend themselves better to repetitive tasks and methods for importing data from a wide variety of sources.

Stata has various commands for importing data. The three main commands for reading non–Stata datasets in text are

- `import delimited`, which is made for reading text files created by spreadsheet or database programs or, more generally, for reading text files with clearly defined column delimiters such as commas, tabs, semicolons, or spaces;
- `infile`, which is made for reading simple data that are separated by spaces or rigidly formatted data aligned in columns; and
- `infix`, which is made for data aligned in columns but possibly split across rows.

Stata has other commands that can read other types of files and can even get data from external databases without the need for an interim file:

- The `import excel` command can read Microsoft Excel files directly, either as an `.xls` or as an `.xlsx` file.
- The `import sasxport` command can read any SAS XPORT file, so data can be transferred from SAS to Stata in this fashion.
- The `odbc` command can be used to pull data directly from any data sources for which you have ODBC (Open Database Connectivity) drivers.

Stata can import even more formats; see [D] **import** for the full list.

Each command expects the file that it is reading to be in a specific format. This chapter will explain some of those formats and give some examples. For the full description, consult the *Data Management Reference Manual*.

The import delimited command

The `import delimited` command was specially developed to read in text files that were created by spreadsheet or database programs because these are common formats for sharing datasets on the Internet. All spreadsheet programs and most database applications have an option to save the dataset as a text file with the columns delimited with either tab characters or commas. Some of these programs also save the column titles (variable names, in Stata) in the text file.

To read in such a file, you have only to type `import delimited` *filename*, where *filename* is the name of the text file. The `import delimited` command will figure out what the delimiter character is (tab or comma) and what type of data is in each column. As always, if *filename* contains spaces, put double quotes around the filename, and include the path if *filename* is not in the current working directory.

By default, the `import delimited` command understands files that use the tab or comma as the column delimiter automatically. If you have a file that uses another character as the delimiter, use `import delimited`'s `delimiters()` option.

Earlier in this chapter, you saved a file called `a few cars.csv` in *Copying and pasting*. These data correspond to the make, price, MPG, weight, and gear ratio of a few very old cars. The variable names are not in the file (so `import delimited` will assign its own names), and the fields are separated by commas. Clear out any existing data, then use `import delimited` to read the data in this file. Because there are spaces in the filename, it must be enclosed in double quotes.

```
. clear
. import delimited "a few cars.csv"
(5 vars, 7 obs)
```

You can look at the data in the Data Editor, and it will look just like the earlier result from copying and pasting. We will now list the data so that we can see them in the manual. The `separator(0)` option suppresses the horizontal separator line that is drawn after every fifth observation by default.

```
. list, separator(0)

     +-------------------------------------------+
     |          v1     v2   v3     v4      v5 |
     |-------------------------------------------|
  1. |   VW Rabbit   4697   25   1930    3.78 |
  2. |     Olds 98   8814   21   4060    2.41 |
  3. | Chev. Monza   3667    .   2750    2.73 |
  4. |               4099   22   2930    3.58 |
  5. |  Datsun 510   5079   24   2280    3.54 |
  6. | Buick Regal   5189   20   3280    2.93 |
  7. |  Datsun 810   8129    .   2750    3.55 |
     +-------------------------------------------+
```

If you want to specify better variable names, you can include the desired names in the command. When you specify variable names, you must also use the `using` keyword before the filename.

```
. import delimited make price mpg weight gear_ratio using "a few cars.csv"
(5 vars, 7 obs)
. list, separator(0)

     +-------------------------------------------+
     |       make   price   mpg   weight   gear_r~o |
     |-------------------------------------------|
  1. |   VW Rabbit    4697    25     1930      3.78 |
  2. |     Olds 98    8814    21     4060      2.41 |
  3. | Chev. Monza    3667     .     2750      2.73 |
  4. |                4099    22     2930      3.58 |
  5. |  Datsun 510    5079    24     2280      3.54 |
  6. | Buick Regal    5189    20     3280      2.93 |
  7. |  Datsun 810    8129     .     2750      3.55 |
     +-------------------------------------------+
```

As a side note about displaying data, Stata listed `gear_ratio` as `gear_r~o` in the output from `list`. `gear_r~o` is a unique abbreviation for the variable `gear_ratio`. Stata displays the abbreviated variable name when variable names are longer than eight characters.

To prevent Stata from abbreviating `gear_ratio`, you could specify the `abbreviate(10)` option:

```
. list, separator(0) abbreviate(10)

     +-------------------------------------------+
     |       make   price   mpg   weight   gear_ratio |
     |-------------------------------------------|
  1. |   VW Rabbit    4697    25     1930      3.78 |
  2. |     Olds 98    8814    21     4060      2.41 |
  3. | Chev. Monza    3667     .     2750      2.73 |
  4. |                4099    22     2930      3.58 |
  5. |  Datsun 510    5079    24     2280      3.54 |
  6. | Buick Regal    5189    20     3280      2.93 |
  7. |  Datsun 810    8129     .     2750      3.55 |
     +-------------------------------------------+
```

For more information on the ~ abbreviation and on `list`, see **[GSW] 10 Listing data and basic command syntax**.

We will use this dataset again in the next chapter, so we would like to save it. Type `save afewcars`, and press *Enter* in the Command window to save the dataset.

For this simple example, you could have copied the contents of the file and pasted it into the Data Editor by using **Paste special...** and choosing comma as the delimiter.

For text files that have no nice delimiters or for which observations could be spread out across many lines, Stata has two more commands: `infile` and `infix`. See [D] **import** for more information about how to read in such files.

Importing files from other software

Stata has some more specialized methods for reading data that were created by other applications and stored in their proprietary formats.

The `import excel` command is made for reading files created by Microsoft Excel. See `import excel` in [D] **import excel** for full details.

The `import sasxport` command can read and create SAS XPORT Transport files. See [D] **import sasxport** for full details.

If you have software that supports ODBC, you can read data by using the `odbc` command without the need to create interim files. See [D] **odbc** for full details.

Here is a brief summary of the choices:

- If you have a table, you could try copying it and pasting it into the Data Editor.
- If you have a Microsoft Excel `.xls` or `.xlsx` file, use `import excel`.
- If you have a file exported from a spreadsheet or database application to a tab-delimited or CSV file, use `import delimited`.
- If you have a fixed-format file, either use `infile` with a dictionary or use `infix`.
- If you have a database accessible with ODBC, use `odbc`.
- If you have a SAS XPORT file, use `import sasxport`.
- If you have economic data from the Federal Reserve Data, use `import fred`.
- If you subscribe to any Haver Analytics databases, use `import haver`.
- If you have a dBASE file, use `import dbase`.
- Finally, you can purchase a transfer program that will convert the other software's data file format to Stata's data file format.